UNIVERSIDAD DE SALAMANCA
ESCUELA POLITÉCNICA SUPERIOR DE ZAMORA

GRADO EN INGENIERÍA MECÁNICA
DEPARTAMENTO DE INFORMÁTICA Y AUTOMÁTICA
AREA DE LENGUAJES Y SISTEMAS INFORMÁTICOS

PROGRAMA DE CONTROL DEL BRAZO ROBOT MITSUBISHI RV-E2 Y SU APLICACIÓN AL PROBLEMA DE LAS TORRES DE HANÓI

AUTOR: VÍCTOR GAGO COCO
TUTOR: JOSÉ LUÍS PÉREZ IGLESIAS

FECHA DE ADJUDICACIÓN: NOVIEMBRE 2016
FECHA DE PRÓRROGA: OCTUBRE 2017
FECHA DE PRESENTACIÓN: FEBRERO 2018

RESUMEN

La elección de este Trabajo Fin de Grado se ha considerado por la creciente importancia en el mundo de la robótica, con una notable presencia en el ámbito industrial. También se ha considerado la presencia de un brazo robot marca Mitsubishi y modelo MOVEMASTER RV-E2 en las instalaciones de la Escuela Politécnica Superior de Zamora.

Con todo ello se decidió la elaboración de este trabajo para dotar a dicho robot de un programa de control que pudiese comunicarse con su unidad de control. Este programa debe ser de fácil acceso al usuario y compatible con los ordenadores a los que se pueda conectar el robot.

El lenguaje escogido ha sido Visual Basic en su versión 6.0, debido a que se trata de un lenguaje de programación sencillo y con alta compatibilidad con los equipos de los que se dispone.

También se ha considerado la importancia de realizar un trabajo con dicho robot, no solo para comprobar la capacidad del programa de conexión y control, sino también para dotar al robot de una función práctica. Esta función debe poder ser estudiada y servir como demostración tanto de las capacidades del robot, como de las posibilidades del programa de control. La tarea que se ha encomendado realizar al robot ha sido la resolución física del problema de las torres de Hanói.

En este documento se definirán las características básicas de los robots industriales, centrándose en los denominados brazos robot por ser el caso particular que se va a tratar. Además se pondrá atención en el robot Mitsubishi con el que se va a conectar y el lenguaje de programación Visual Basic con el que se desarrollará la aplicación para el control del mismo. La conexión por medio de un puerto serie RS-232 tendrá también un espacio importante debido a que se trata del medio de conexión PC-Robot en este caso.

Por último, se considerarán las características principales del programa de control del brazo robot y la aplicación sobre el problema de las torres de Hanói.

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

1. INTRODUCCIÓN

1.1 Historia de la robótica

Los robots realizan las tareas que se les encomiendan de manera más rápida, precisa y sin sufrir fatiga por el tiempo de trabajo. De esta manera han llegado a sustituir a los operarios humanos en numerosas tareas, de forma más notable en aquellas que resultan monótonas, repetitivas o peligrosas. Debido a su incidencia en el último siglo se ha pasado de una producción eminentemente artesanal a una producción industrial, pasando de producirse pocos productos en mucho tiempo (con un gran coste por unidad) a una producción en cadena que ha facilitado la fabricación de grandes lotes en poco tiempo, reduciendo el coste del producto.

Con la aparición de los robots también llego la necesidad de programarlos para la realización de las tareas que se les encomendaban. Los medios por los que se puede realizar la programación de los robots son muy diversos, desde cuadros eléctricos con elementos como relés o interruptores; autómatas programables o desde computadoras especificas o personales a través de los lenguajes de programación más comunes para la programación general en estos sistemas.

Por consiguiente podemos decir que un robot es una máquina con algún tipo de controlador capaz de programarse para realizar un trabajo determinado, con la capacidad de moverse, interaccionar o manipular objetos, interaccionando así con su entorno. Una de sus definiciones más aceptada es la siguiente:

"Dispositivo multifuncional reprogramable diseñado para manipular y/o transportar material a través de movimientos programados para la realización de tareas variadas" (Robot Institute of América, 1979)

La idea más común es que el robot esté asociado con un dispositivo de control digital con una memoria en la que se puedan almacenar distintos programas y con la ejecución de este se muevan los elementos mecánicos que interactúan con su entorno y realizan la tarea encomendada. Cambiando el programa del robot cambian los movimientos y la interacción de este, realizando una tarea diferente.

Por otra parte podríamos definir la robótica como la ciencia que estudia el diseño y construcción de los robots. El término robótica fue acuñado por Isaac Asimov, el cual

también formuló las tres famosas leyes de la robótica que han de regir las programaciones de las máquinas:

1. Un robot no puede dañar a un ser humano ni, por inacción, permitir que un ser humano sufra daño.
2. Un robot debe cumplir las órdenes de los seres humanos, excepto si dichas órdenes entran en conflicto con la Primera Ley.
3. Un robot debe proteger su propia existencia en la medida en que ello no entre en conflicto con la Primera o la Segunda Ley.

Estas leyes fueron enunciadas en *"Círculo vicioso"* publicado en 1942, más tarde Asimov incidió de nuevo en esta idea en un conjunto de relatos publicados con el título *"Yo, robot"* (1950) donde planteaba paradojas, dilemas y/o conflictos resultantes de tener que cumplir con estas leyes.

Aunque la aparición de los robots, tal y como los conocemos hoy en día, es relativamente reciente, ahonda en ideas y motivaciones planteadas a lo largo de varios siglos de historia humana. Se conocen casos de la época griega que intentan recrear dispositivos con un movimiento sin fin, que no tengan que ser controlados por personas. En los siglos XVII y XVIII se desarrollaron sistemas mecánicos humanoides controlados con mecanismos de relojería que tenían alguna de las características de los robots.

Durante la revolución industrial se crearon nuevas máquinas, que sin dejar de ser puramente mecánicas, empezaron a sentar las bases de lo que posteriormente sería la aparición de la robótica. Entre estos hechos debe considerarse la "Hiladora Spinning Jenny" de Hargreaves en 1764. Esta invención reducía notablemente el tiempo de producción facilitando a un solo operario el trabajo que debían de realizar seis hiladoras ordinarias.

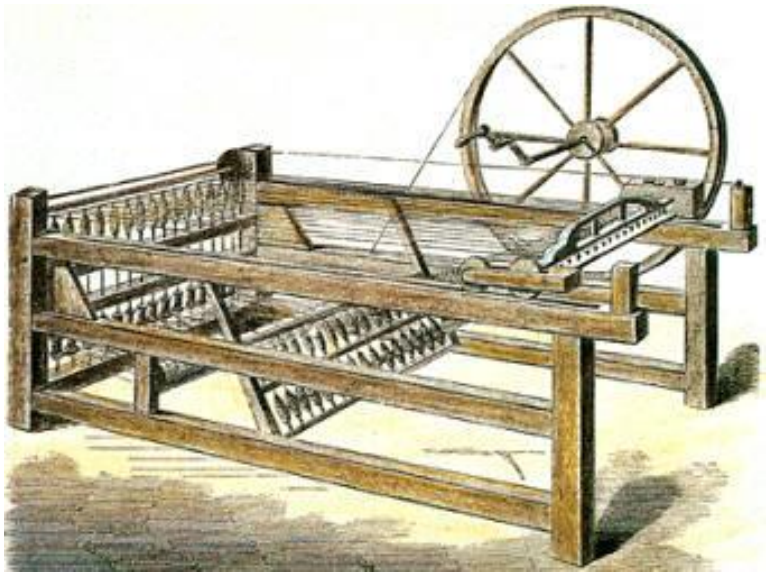


Fig. 1: Hiladora Spinning Jenny de Hargreaves

Aunque en 1948 Goertz inventa un manipulador maestro-esclavo germen de la robótica industrial, planteando la necesidad de maquinaria flexible que pueda realizar distintos procesos; no es hasta la década siguiente cuando aparecen los primeros robots con las características que hoy reconocemos.

C. W. Kenward solicitó la primera patente de un sistema robótico en 1954 en Reino Unido, siéndole concedida en 1957. No obstante está considerado como el primer robot al producido por la empresa Unimation, fundada por G. Devol y J. Engelberger. Debido a su éxito la compañía General Motors los instalaría en sus fábricas.

Fig. 2: Robot Unimation

Durante la década de los 70 la Universidad de Stanford y el MIT (Instituto tecnológico de Massachusetts) se involucran en la tarea de controlar un robot mediante un

computador. A partir de 1975 la aparición de los microprocesadores redujo la dificultad y el coste económico de esta tarea.

Debido al aumento destacable de fabricación de robots durante 1980 y los grandes avances producidos en la década siguiente, se conoce a este año como el "primer año de la época robótica"

En 1982 IBM crea un lenguaje de programación específico para la robótica: AML (A Manufacturing Lenguage). También en esta década se produce el primer robot SCARA (acrónimo de sus siglas en inglés: Selective Compliant Assembly Robot Arm) de accionamiento directo, es decir, el robot realizaba el movimiento mecánico a través de motores eléctricos colocados directamente en las articulaciones, sin necesidad de cadenas de engranajes u otros medios de transmisión del movimiento de forma mecánica.

Otra de las características que hay que atender a la hora de conocer el desarrollo de la robótica, es la forma en la cual se realizan los movimientos. Desde los primeros prototipos con pocos grados de libertad hasta el movimiento en ejes cartesianos se han desarrollado distintas formas de los mecanismos. La aparición de los robots cilíndricos o la fabricación de las primeras unidades que se movían sobre pórticos, han sido importantes innovaciones que responden a la solicitud de los requerimientos de la industria.

Uno de los principales problemas que se le planteaban a los operarios encargados de los determinados robots que tenían que supervisar era la comunicación hombre-máquina. Para ello en 1992 la empresa ABB desarrolla un sistema de control abierto, el controlador S4.

Fig. 3: Controlador S4.

El robot industrial, favorecido enormemente por las técnicas de control por computadora, ha sido uno de los mayores impulsores de la automatización industrial. Este hecho ha sido clave en el desarrollo de la industria y en los procesos de fabricación en serie. La fabricación de pequeñas series de producción había quedado en un inicio excluida de la utilización de robots industriales. No obstante la capacidad que han adquirido con el tiempo para ser fácilmente reprogramables y multifuncionales, terminó por incluir a estas máquinas en todos los procesos, sin importar que el trabajo programado, ya que eran fácilmente programables y podían emplearse en muy distintas tareas.

La capacidad de cambiar de herramienta y reprogramar los movimientos y las interacciones con el entorno de una forma rápida y sencilla, ha terminado por hacer del robot industrial parte fundamental en cualquier tipo de ámbito de la industria.

1.2 Brazos mecánicos

Ya centrándonos en el robot que nos ocupa, una gran parte de los robots industriales presentan la forma de brazo, mediante el cual pueden utilizarse distintos tipos de herramientas situadas en el extremo del brazo. Además presentan un gran número de sensores capaces de recoger la información procedente del exterior, la cual es empleada en su programación informática para la toma de decisiones.

La denominación de brazo robot se debe a su semejanza de movimientos con un brazo humano. Esta similitud también le otorga a las articulaciones del robot los nombres de las articulaciones humanas a las que corresponderían, comúnmente utilizado el inglés para este hecho por ser el idioma más extendido en estos ámbitos.

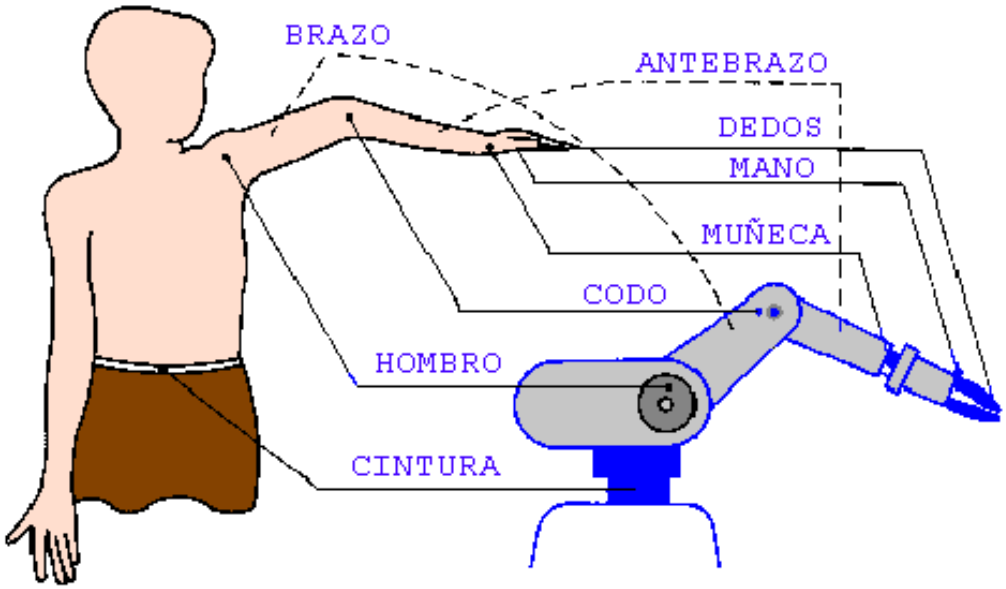


Fig. 4: Correspondencia de las articulaciones entre brazo humano y mecánico

La mayoría de los robots más sofisticados de la industria o incluso de la ciencia en general tienen esta estructura de brazo robot. Esto les otorga una gran versatilidad para sostener, mover o reorientar objetos. La estructura en forma de brazo los hace más parecido a los humanos.

Esta tipología de robot puede emplearse de manera estacionaria (muy común en la industria) o como apéndice de un sistema móvil mayor. Al hablar de brazo robot normalmente se excluye la herramienta o mano, ya que esta se centra en la función específica para realizar el trabajo y no influye en el movimiento general del robot o su comportamiento mecánico.

Una de las características que definen un brazo robot es la forma del área que puede alcanzar el extremo del mecanismo (la herramienta). A esta zona accesible se le conoce como espacio de trabajo y es el conjunto de todos los puntos donde puede actuar de forma efectiva la herramienta.

Distintos mecanismos y herramientas hacen posible el movimiento y funcionamiento de un brazo robot. Por lo general consta de una base que está unida a un elemento rígido (que puede ser móvil o estático) donde se encuentra el hardware principal que lo controla y hará funcionar. En esta base se encuentra la placa base, circuitería principal o incluso puede situarse una computadora que gobierne el funcionamiento del robot. Además presenta un brazo principal que permite el movimiento en las tres dimensiones del espacio. Y en la mayoría de los casos suele encontrarse una segunda parte o antebrazo, el cual está unido por otra articulación que le otorga mayor versatilidad en los movimientos que realiza y suele aportar precisión en su movimiento.

Una forma de clasificar los brazos robot es mediante el tipo de coordenadas que definen su movimiento. No obstante, hoy en día los avanzados sistemas de control permiten versatilidad en la forma de definir las coordenadas con las cuales podemos mover el robot. De una forma sencilla esta clasificación podría ser la siguiente:

- **Coordenadas angulares o de revolución:** Este tipo de brazo define la posición por medio del valor angular que presentan las distintas articulaciones del mismo.
- **Coordenadas polares o esféricas:** El espacio de trabajo tiene forma esférica. Este tipo de robot presenta dos grados de libertad definidos por ejes rotacionales, igual que en el caso anterior, pero el tercer grado de libertad queda definido por un carro que varía la longitud del antebrazo, un eje lineal.
- **Coordenadas cilíndricas:** Su espacio de trabajo se asemeja a un cilindro. Presenta también un eje rotacional en la base, pero los dos grados de libertad siguientes son lineales, normalmente uno vertical y otro horizontal.
- **Coordenadas rectangulares:** En este caso los tres grados de libertad que presenta son de forma lineal. El movimiento se realiza a través de distintos carros que mueven el robot de forma lineal en distintos ejes.

Está clasificación es una forma sencilla, en la realidad los robots pueden combinar distintas formas y aunar distintos grados de libertad haciéndolos más complejos. No

obstante lo más común es encontrarse brazos robot de coordenadas angulares, por ser uno de los que mayor versatilidad presentan. No obstante mediante los controladores se puede definir el movimiento en distinto tipo de coordenadas atendiendo a la resolución del problema de posición inverso, es decir, definir la posición en un sistema de coordenadas distinto al sistema que otorga el movimiento al robot.

1.3 El robot Mitsubishi RV-E2

Una vez introducida la historia de la robótica y conocidos las principales características y tipos de robots, toca hablar del que va a ser el protagonista principal de este proyecto: el robot de la empresa Mitsubishi y la serie MOVEMASTER con denominación RV-E2.

Fig. 5: Robot Mitsubishi RV-E2

Se trata de un robot con 6 grados de libertad, todos ellos de rotación, que se controlan a través de distintos motores eléctricos de corriente alterna. Además está capacitado con encoders de posición absoluta (realmente son relativos pero una memoria RAM, alimentada mediante pilas, almacena las posiciones absolutas). También puede acoplarse un grado más de libertad, ya que tiene la capacidad de controlar un eje lineal adicional. Está equipado con una herramienta en forma de pinza, pero puede equiparse con otros medios al tener disponible el control de dos manos neumáticas.

Esta máquina tiene una capacidad de carga efectiva de 2kg, un alcance máximo de 706mm en el punto más alejado de su espacio de trabajo, una velocidad máxima de 3500mm/s y una repetitividad de ±0.04mm. El peso del brazo en su conjunto es de 36kg.

La unidad de control (CR-E116) gobierna y calcula las trayectorias del brazo. Está equipada con un microprocesador tipo RISC de 32 bits y un DSP encargado de enviar las órdenes a los servoamplificadores que equipan el robot. El peso de la unidad de control es de 27kg y no se encuentra unida al brazo. Esta unidad cuenta con 30 entradas y 16

salidas digitales, con las que recibir información de un operario o del entorno, mediante las cuales se puede intervenir en el programa que esté ejecutando.

La memoria de 62Kb puede almacenar hasta 999 posiciones y hasta 30 programas. Posee dos interfaces de comunicación: un RS-232 para comunicar con un PC y un RS-422 con el que se conecta a un mando con el que se puede controlar el robot, denominado Teaching Box.

Los comandos empleados para trasmitir las órdenes al robot se han de proporcionar en un lenguaje tipo BASIC. Una vez trasmitidos los programas mediante el puerto serie desde el PC, no es necesaria mantener este tipo de conexión. La memoria interna de la unidad de control mantiene los programas almacenados, al igual que las posiciones definidas y la posición actual del robot.

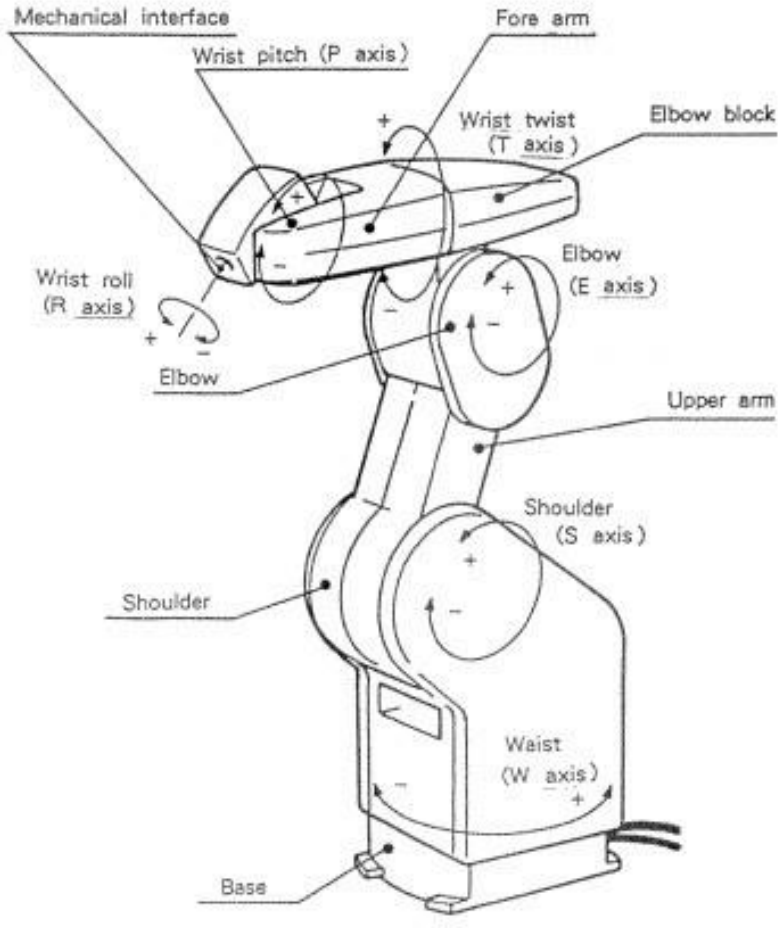


Fig. 6: Articulaciones y sentido de giro robot Mitsubishi RV-E2

Las iniciales de las articulaciones (en inglés) definen los ejes que proporcionan los distintos grados de libertad al brazo.

En la siguiente tabla se muestran las longitudes de las distintas partes del brazo:

Altura del pie	350mm
Distancia hombro-eje del pie(cadera)	100mm
Longitud hombro-codo	250mm
Longitud segundo tramo del brazo	130mm
Longitud del tramo perpendicular	250mm
Longitud de la mano	85mm

Tabla 1: Longitudes del brazo. Robot Mitsubishi RV-E2.

En la siguiente figura se muestran las distancias que alcanza el brazo y las longitudes del mismo antes descritas.

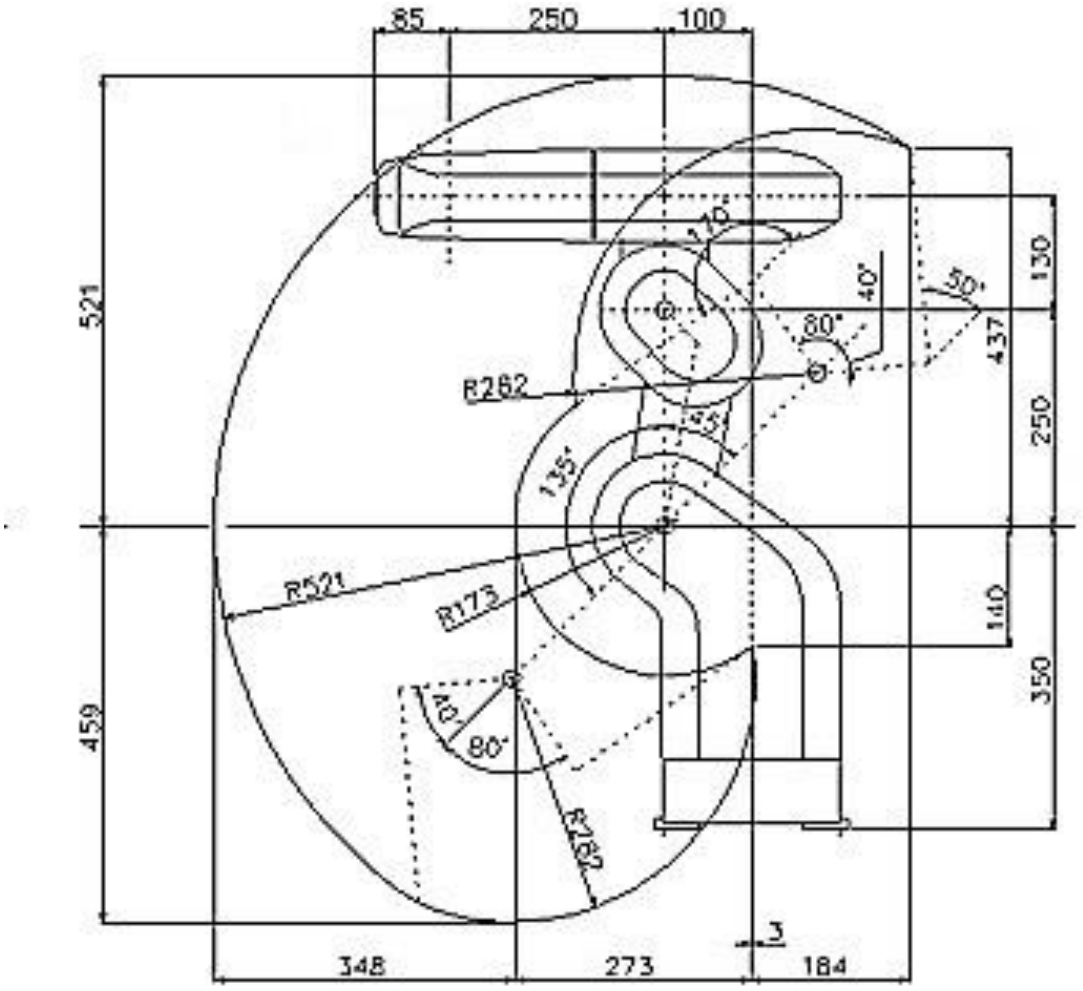


Fig. 7: Distancias y longitudes robot Mitsubishi RV-E2

Los límites para los ángulos de los distintos ejes son:

Waist	±160
Shoulder	-45 a +135
Elbow	-40 a +80
Wrist Twist	±160
Twist Pitch	±120
Wrist Roll	±200

Tabla 2: Límite de grados para los distintos ejes del robot Mitsubishi RV-E2

Con estos datos se puede definir la posición del robot como un mecanismo de 6 barras (una barra entre cada articulación más la longitud de la herramienta) con 6 grados de libertad, o incluso se puede reducir a un mecanismo de 4 barras con una articulación con 3 grados de libertad (suponiendo que confluyen en el mismo punto los tres giros de la muñeca o Wrist).

1.4 Visual Basic 6.0

Para realizar el programa encargado del control del brazo robot, que se ejecutará desde el PC y se comunicará con la unidad de control mediante un puerto serie tipo RS-232, se ha optado por realizar tal programación por medio de Visual Basic en su versión 6.0.

Visual Basic es un lenguaje de programación desarrollado por Microsoft. Su característica principal es que se trata de un lenguaje orientado a objetos. Este lenguaje de programación está basado en el tipo BASIC aunque presentó importantes novedades y agregados que facilitaban de manera notable el trabajo de los programadores.

Se trata de un lenguaje de programación sencillo e intuitivo. El entorno de desarrollo integrado integra un editor de interfaces gráficas donde es sencillo elaborar la presentación de los distintos formularios o ventanas donde insertar los objetos y un editor de texto para la edición del código fuente. Además presenta dentro del entorno de desarrollo integrado una función para ejecutar el programa que se está desarrollando, es decir tiene su propio intérprete sin necesidad de compilar y generar la aplicación entera, donde presenta su propio tratamiento de errores para comprobar el funcionamiento del programa.

La versión 6 fue la última liberada por Microsoft en 1998 y se extendió su soporte hasta el año 2008. Posteriormente la compañía lanzó sus lenguajes de programación en basado en otros tipos de lenguaje (C#) que fueron los sucesores de Visual Basic. No obstante el desarrollo de estos tipos de lenguaje presentaba varias similitudes con Visual Basic y un entorno de desarrollo muy parecido.

2. OBJETIVOS DEL PROYECTO

2.1 El programa de comunicación PC-Robot

El objetivo principal de este proyecto es elaborar un programa para poder realizar de manera sencilla y cómoda la comunicación entre un ordenador personal y la unidad de control del robot Mitsubishi RV-E2 presente en la Escuela Politécnica Superior de Zamora de la Universidad de Salamanca.

La comunicación ha de establecerse mediante un puerto serie y un canal de comunicación tipo RS-232. Para ello el programa ha de poder transferir los datos de manera que sean interpretables para el robot y según las condiciones que necesita para establecer la conexión de manera eficiente.

Para este aspecto lo primero que hay que considerar son las instrucciones recogidas en los distintos manuales proporcionados por Mitsubishi. Ante este hecho es importante considerar que el tipo de lenguaje al que se refieren en dichos manuales es un lenguaje BASIC. Por consiguiente es necesario interpretar las explicaciones que recogen y traducirlas al lenguaje de programación utilizado en nuestro caso. También hay que considerar que algunos de estos procesos se ejecutan de manera distinta o automática.

Otra consideración importante es el cable de conexión. Este cable ha de ser de tipo RS-232, sin embargo esta conexión se puede hacer por dos tipos de puertos: DB25 (25 pines) o DB9 (9 pines). La unidad de control presenta un puerto tipo DB25, pero lo más común de encontrar en los ordenadores (siendo un sistema que hoy en día está obsoleto) es del tipo DB9. Esta diferencia de puertos, o de cantidad de pines, no interfiere en este caso, ya que con las conexiones presentes en el segundo tipo el robot es perfectamente funcional (los pines restantes no son empleados por la unidad). Por tanto es importante establecer la correlación entre los pines de uno y otro tipo y en caso de ser necesario una conexión mediante un cable con dos extremos diferentes que la correspondencia entre una conexión y otra sea la correcta.

Entrando en detalle de las funciones que va a tener que realizar el programa de comunicación, hay que distinguir que aspectos ha de presentar y que funciones debe de tener para el usuario. Estas funciones deben ser acordes con las opciones que ofrece el robot en sus posibilidades de comunicación con un PC.

Lo primero que hay que considerar es que se pueda establecer una conexión eficiente con la unidad de control. Para ello el programa ha de posibilitar la apertura del puerto al que se conecte el cable de conexión dentro del hardware del propio PC. En caso de que el ordenador presente distintos puertos ha de ser necesario poder elegir entre ellos y confirmar que el puerto seleccionado es mediante el cual se trabaja.

Es importante considerar que la comunicación entre el ordenador y el robot se hace siempre por caracteres, bien sea a través de caracteres individuales o cadenas. Por ello hay que considerar que la variable a tratar para este objetivo ha de ser siempre de este tipo.

Una vez se ha establecido una conexión de manera correcta el robot presenta dos tipos de comunicación que pueden realizarse: se pueden enviar órdenes de manera directa y ejecución inmediata o se puede realizar a través del envío de programas que se almacenan en la memoria.

Al diferenciar estas dos posibilidades de comunicación hay que considerar también las singularidades que se presentan en cada caso. El envío de una orden directa no requiere de más rutinas que las referidas a establecer una comunicación efectiva a la hora del envío de los distintos caracteres (estas características se definen en los próximos apartados). Por el contrario el envío de un programa definido requiere el envío de otros aspectos, como el número de programa o la numeración de las órdenes.

Estas singularidades en la comunicación nos advierten de la necesidad de establecer dos protocolos distintos para el envío de datos. Uno mediante el cual se envíen programas enteros que puedan ser almacenados en la memoria del robot y ejecutados cuando se le ordene y otro por el cual se envíen órdenes simples de ejecución inmediata.

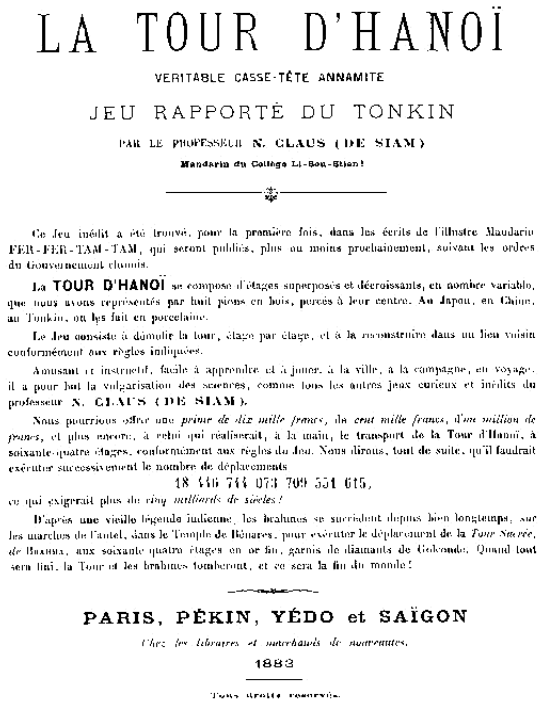
Por otro lado, la comunicación entre el PC y el robot es bidireccional, es decir, que la unidad de control también tiene la capacidad de enviar datos al ordenador al que se encuentre conectado. Para ello es necesario establecer un medio por el cual puedan leerse los datos emitidos por el robot sin interferir en el trabajo que esté realizando el usuario.

Para el reconocimiento de la solicitud de envío de datos desde el robot al ordenador al cual está conectado, ha de establecerse un protocolo que detecte dicha solicitud y un medio por el cual puedan leerse los caracteres enviados.

Por último es necesario comprobar el correcto funcionamiento del programa y la correcta actuación del robot. Para ello es necesario demostrar que el robot es capaz de obedecer órdenes directas y realizar un programa completo. El trabajo elegido para que el robot desarrolle un programa completo ha sido el problema de las torres de Hanói.

2.2 Torres de Hanói

Las instrucciones del rompecabezas conocido como "La torre de Hanói" fueron publicadas en 1883 por el "Profesor N. Claus de Siam", pseudónimo del matemático francés Édouard Lucas d'Amiens (1842-1891), que trabajó en el observatorio de París y como profesor de matemáticas en varios liceos, a él se le atribuyen varios juegos matemáticos entre los que destaca el que nos ocupa en este proyecto.



LA TOUR D'HANOÏ

VERITABLE CASSE-TÊTE ANNAMITE

JEU RAPPORTÉ DU TONKIN

PAR LE PROFESSEUR N. CLAUS (DE SIAM)

Mandarin du Collège Li-Sou-Stian!

Ce Jeu inédit a été trouvé, pour la première fois, dans les écrits de l'illustre Mandarin FER-FER-TAM-TAM, qui seront publiés, plus ou moins prochainement, suivant les ordres du Gouvernement chinois.

La TOUR D'HANOÏ se compose d'étages superposés et décroissants, en nombre variable, que nous avons représentés par huit pions en bois, percés à leur centre. Au Japon, en Chine, au Tonkin, on les fait en porcelaine.

Le Jeu consiste à démolir la tour, étage par étage, et à la reconstruire dans un lieu voisin conformément aux règles indiquées.

Amusant et instructif, facile à apprendre et à jouer, à la ville, à la campagne, en voyage, il a pour but la vulgarisation des sciences, comme tous les autres jeux curieux et inédits du professeur N. CLAUS (DE SIAM).

Nous pourrions offrir une *prime de dix mille francs*, de *cent mille francs*, d'*un million de francs*, et plus encore, à celui qui réaliserait, à la main, le transport de la Tour d'Hanoï, à soixante-quatre étages, conformément aux règles du Jeu. Nous dirons, tout de suite, qu'il faudrait exécuter successivement le nombre de déplacements

18 446 744 073 709 551 615,

ce qui exigerait plus de *cinq milliards de siècles!*

D'après une vieille légende indienne, les brahmes se succèdent depuis bien longtemps, sur les marches de l'autel, dans le Temple de Bénarès, pour exécuter le déplacement de la *Tour Sacrée, de Brahma*, aux soixante-quatre étages en or fin, garnis de diamants de Golconde. Quand tout sera fini, la Tour et les brahmes tomberont, et ce sera la fin du monde!

PARIS, PÉKIN, YÉDO et SAÏGON

Chez les libraires et marchands de nouveautés.

1883

Tous droits réservés.

Fig. 8: Instrucciones originales del juego, publicadas bajo el pseudónimo de: "Profesor N. Claus de Siam" formado con las mismas letras que el suyo propio.

El nombre ha ido derivando a lo largo de los años a "torres de Hanói" en plural, debido a que en el desarrollo del juego se van generando varias torres. Aunque también es llamado "Torre de Brahma" o "Las torres del fin del mundo" debido a la leyenda que el propio autor mencionó en las instrucciones del juego, no se sabe muy bien si porque se inspiró en alguna historia oriental o porque se la inventó como reclamo comercial al juego. Es en 1884, en la publicación de un artículo en la revista "La Nature" en el que el divulgador científico Henri de Parville (1838-1909), desarrolla la leyenda del juego:

"En el gran templo de Benarés, debajo de la cúpula que marca el centro del mundo, yace una base de bronce, en donde se encuentran acomodadas tres agujas de diamante,

cada una del grueso del cuerpo de una abeja y de una altura de 50 cm aproximadamente. En una de estas agujas, Dios, en el momento de la Creación, colocó sesenta y cuatro discos de oro -el mayor sobre la base de bronce, y el resto de menor tamaño conforme se va ascendiendo-. Día y noche, incesantemente, los sacerdotes del templo se turnan en el trabajo de mover los discos de una aguja a otra de acuerdo con las leyes impuestas e inmutables de Brahma, que requieren que siempre haya algún sacerdote trabajando, que no muevan más de un disco a la vez y que deben colocar cada disco en alguna de las agujas de modo que no cubra a un disco de radio menor. Cuando los sesenta y cuatro discos hayan sido transferidos de la aguja en la que Dios los colocó, en el momento de la Creación, a otra aguja, el templo y los brahmanes se convertirán en polvo y, junto con ellos, el mundo desaparecerá"

Normas:

El juego original se compone de tres postes verticales y una torre de ocho discos apilados en uno de los postes de mayor a menor diámetro.

La finalidad del juego consiste en trasladar la torre de discos de un poste a otro quedando ordenados de la misma manera y siguiendo estas tres simples normas:

- Sólo se puede mover un disco en cada movimiento.

- No se puede colocar un disco sobre otro de menor diámetro.

- Solo se puede mover el disco superior de cada varilla.

Son numerosos los matemáticos que han estudiado este particular problema. Para cualquiera de ellos ha resultado evidente que el problema original, con los ocho discos, se descompone a su vez en solventar el mismo problema con un disco menos, quedando así liberado el último disco para poder moverse y teniendo otra varilla libre a la cual moverlo. Esto quiere decir que el problema se basa en la recursividad de sucesivos problemas más sencillos.

Con este planteamiento podemos empezar por determinar los movimientos necesarios para un número menor de discos. El problema más sencillo sería el caso de tener que mover una torre de un único disco. Esté problema es sencillo y se resuelve moviendo el disco de una varilla a otra. Para el caso de dos discos debemos realizar la resolución para un solo disco, mover el segundo disco y volver a realizar la resolución de la torre de un único disco colocando dicha torre sobre el segundo disco.

Aunque esta resolución parece demasiado sencilla, puede generalizarse para cualquiera que sea el número de discos. Por consiguiente para resolver el problema con *n* discos hay que resolver el problema para *(n-1)* discos, mover el disco *n* y repetir el proceso para *(n-1)*.

Esto es un claro ejemplo de recursividad. Por consiguiente puede determinarse el número de movimientos mínimos que se necesitan para resolver este problema con el número de discos que se establezca en cada caso.

Según lo enunciado hasta ahora podemos definir una función para determinar el número de movimientos como; el número de movimientos para resolver la torre con un disco menos, más el movimiento del disco; más volver a mover la torre con un disco menos. Empezando a estudiarlo desde el caso de un solo disco hasta la generalización para *n* discos:

$$f(1) = 0 + 1 + 0 = 1$$

$$f(2) = 1 + 1 + 1 = 3$$

$$f(3) = 3 + 1 + 3 = 7$$

$$f(4) = 7 + 1 + 7 = 15$$

...

$$f(n) = f(n-1) + 1 + f(n-1) = 2f(n-1) + 1$$

Mediante la realización de un proceso recursivo es fácil demostrar que:

$$f(n) = 2f(n-1) + 1 = \sum_{k=1}^{n-1} 2^k + 1$$

Que por otro lado puede escribirse como:

$$f(n) = 2^n - 1$$

Esta determinación del número de movimientos necesarios no solo nos indica el número de movimientos totales que han de realizarse para resolver este problema, sino que también nos llega a indicar en que paso estamos dentro del problema. Resolviendo este caso de manera inversa podemos definir a partir del paso en el que nos encontramos que disco hemos de mover.

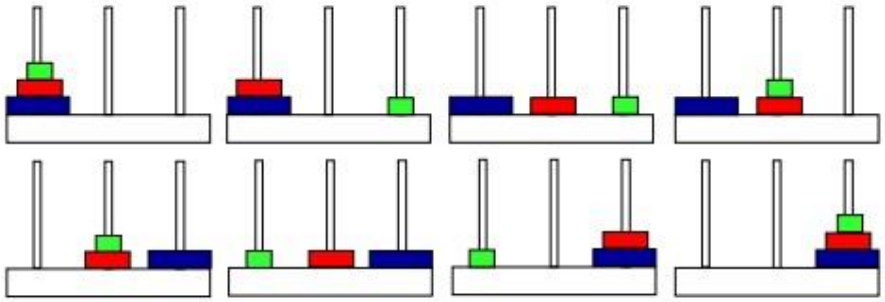


Fig. 9: Resolución del problema de las torres de Hanói para 3 discos en 7 pasos

Una vez conocido que disco vamos a mover y sabiendo en qué posición lo hemos colocado, es fácil determinar de qué a qué varilla hay que mover el disco. Para esté propósito hace falta una última consideración. La varilla a la que vamos a mover un disco no puede bloquearnos el movimiento siguiente, ni interferir con la segunda regla.

Para elaborar un método que defina a qué lugar mover los discos de manera sencilla, uno de las primeras consideraciones es que la varilla a la que se mueva un disco tiene que ser distinta a la que se pretenda mover el disco siguiente. Esto quiere decir que (enumerando los discos de menor a mayor del 1 al 3 y las varillas de izquierda a derecha: A, B y C respectivamente) si el disco 1 se mueve a la varilla B, el 2 irá a la C y el 3 volverá a ir a la B en su primer movimiento. Por consiguiente podemos decir que los discos impares se mueven a la varilla siguiente y los pares de dos en dos. Esto es solo un criterio que se toma para poder elaborar la resolución paso a paso que requieren las órdenes del robot.

Para facilitar el desarrollo del trabajo del robot, se elaborará un programa que determine los movimientos que ha de realizar el robot, donde se introduzcan el número de discos que se llevarán a cabo, de manera que sea independiente el número de discos de la resolución del problema.

También deberá presentar una opción para determinar la posición en la cual se presentan las torres, la altura de las varillas y la altura de los discos. Datos con los cuales se puede definir la posición de los discos en todo momento y los movimientos que ha de realizar el robot.

3. CONEXIÓN RS-232

3.1 El estándar de conexión RS-232-C

Todos los sistemas automáticos e incluso los ordenadores personales necesitan un método por el cual puedan intercambiar información con el exterior (con otros sistemas).

Uno de los sistemas más comunes para el intercambio de información (hasta la aparición de los USB o los sistemas inalámbricos de alta velocidad) fue el protocolo RS-232-C o simplemente RS-232. Esta comunicación tipo serie se establece a partir de un puerto serie con el que se equipan tanto a emisor como a receptor.

Uno de los aspectos más atractivos para usar una comunicación tipo serie es que requiere menos cableado que la comunicación en paralelo. Este hecho cobra especial importancia cuando las distancias son mayores. Por tanto se ha extendido y generalizado notablemente entre los fabricantes y desarrolladores.

Los equipos de comunicación en serie pueden dividirse en tres grupos:

- Simplex: La información se envía en una sola dirección.
- Half-duplex: Los datos pueden ser enviados en ambas direcciones pero en una sola dirección al mismo tiempo.
- Full-duplex: Los dos sistemas conectados pueden enviar información en ambos sentidos y al mismo tiempo.

También puede distinguirse entre la conexión síncrona o asíncrona. En nuestro caso trabajaremos con una conexión de tipo asíncrona. En este caso la información se trasmite carácter a carácter en cadenas de bits. Los caracteres se envían en cadenas de 5 a 8 bits y la sincronización se mantiene solamente durante el envío de cada carácter.

Cuando no hay comunicación entre emisor y receptor la comunicación se mantiene en estado de reposo. Esto se identifica con un elemento de señalización que responde a un 1 en binario. Por tanto se distingue cuando no se está trasmitiendo la información con una tensión negativa en la línea, así aunque no haya trasmisión de datos los equipos mantienen la conexión.

Para la detección de errores a la hora del envío de información se puede añadir un bit de paridad. Debido a la importancia de este bit en concreto, este ocupa siempre la posición más significativa. Este bit se determina por parte del emisor, de forma que el número de

unos dentro de la cadena de bits sea par (paridad par) o impar (paridad impar), dependiendo del criterio que se elija.

También es necesario establecer un elemento de parada. Este elemento corresponde al elemento de reposo (1 binario) y debe especificarse la duración mínima de envío de este elemento que puede ser 1, 1.5 o 2 veces la duración de un bit normal. No es necesario la determinación máxima ya que si el elemento queda en reposo después de la cadena se mantiene la tensión correspondiente al bit de reposo.

Otro elemento que debe considerarse en la conexión es la tasa de baudios (baud rate), este dato define el número de señales por segundo que se trasmiten en el intercambio de información.

Con todo esto la comunicación queda definida de manera sencilla con cuatro datos que son los ajustes principales que hay que definir para este tipo de comunicación: tasa de baudios, cantidad de bits que componen la cadena, tipo de paridad y longitud de parada.

En el caso particular de nuestro robot la sincronización queda definida por una tasa de baudios de 9600, 8 bits por cadena, una paridad de tipo par y 2 bits de parada.

El estándar de conexión RS-232 fue diseñado en la década de los 60 para la conexión entre un equipo terminal de datos (habitualmente un PC) y un equipo de comunicación de datos (normalmente un módem). El estándar RS-232-C es la tercera revisión de la norma RS-232 propuesta por la EIA (Electronic Industries Asociation). Posteriormente el CCITT (Comité Consultivo Internacional Telegráfico y Telefónico) creó una versión internacional conocida como V.24.

Dependiendo de la velocidad de baudios puede conseguirse una trasmisión efectiva con cables de hasta 15 metros de longitud, no obstante se aconsejan distancias menores para evitar posibles problemas.

Este estándar de comunicación indica que el conector debe tener un total de 25 pines donde el conector del terminal de datos debe ser macho y el conector del equipo de comunicación debe ser hembra. No obstante muchos de los equipos que realizan esta conexión no necesitan de la utilización de todos los pines. Por tanto muchos de los ordenadores integran un tipo de conector de únicamente 9 pines conocido como DB9.

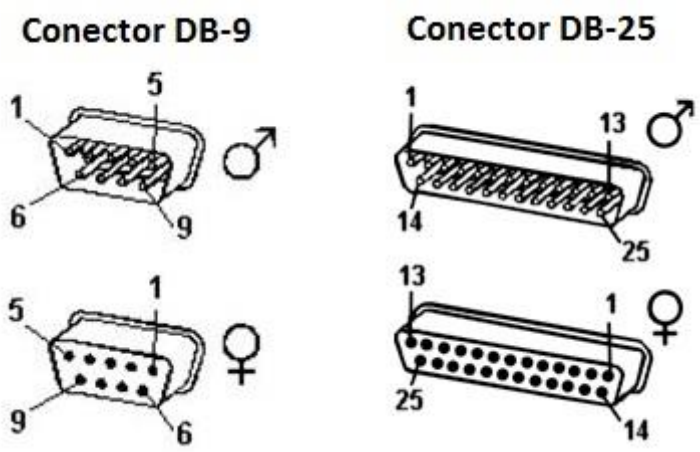


Fig. 10: Conectores tipo DB-9 y DB-25 para la comunicación serie RS-232

Las comunicaciones de este tipo se realizan entre + 12 voltios y -12 voltios. Un cero lógico corresponde con una tensión de entre +9 y +12 voltios y un uno lógico a una tensión de entre -9 y -12 voltios. Por tanto el estado de reposo mantiene una tensión negativa situada entre estos valores.

Una vez definidos los valores para establecer la conexión es importante determinar a que equivalen cada uno de los pines presentes en un conector para la conexión RS-232.

- DTR: El emisor indica mediante una señal por este cable que está encendido y listo para enviar datos.
- DSR: El receptor recibe la señal de que el emisor está listo.
- RTS: El emisor recibe la señal de que el receptor está listo.
- CTS: El receptor indica que está listo para recibir datos.
- TXD: Por este canal el emisor trasmite los datos al receptor.
- RXD: Por este canal el receptor recibe los datos que han sido enviados.
- GND: Esta cable es el encargado de mantener una señal de tierra común para un correcto funcionamiento y lectura de los datos.

Al tratarse de una conexión bidireccional es importante destacar que las señales enviadas por el emisor deben llegar por el canal correspondiente al receptor. Esto indica que los pines que corresponden en cada caso a una señal de salida deben de conectarse mediante el cable con el pin de entrada correspondiente del receptor.

Considerando que el PC empleado para la conexión con el robot tiene un puerto tipo DB-9 y la unidad de control un puerto DB-25. Por ello en la siguiente tabla podemos

observar la correspondencia en el cable de conexión. Para ello se ha tenido en cuenta la relación entre la conexión de un puerto de 25 pines y uno de 9 y la relación emisor receptor que debe establecerse en cada caso.

Pin DB-25	**Señal robot**	**Señal PC**	**Pin DB-9**
2	TXD	RXD	2
3	RXD	TXD	3
4	RTS	CTS	8
5	CTS	RTS	7
6	DTR	DSR	4
20	DSR	DTR	6
7	GND	GND	5

Tabla 3: Relación de los extremos del cable de conexión según el protocolo RS-232.

3.2 Programación de una conexión efectiva PC-Robot

Establecer una comunicación efectiva y correcta ha sido una de las principales dificultades encontradas a la hora de desarrollar este trabajo. Las complicaciones en este aspecto se deben a las singularidades que se presentan a la hora de desarrollar una comunicación que presente las cualidades necesarias para este robot y la falta de conocimiento de los procesos y rutinas que se deben seguir. Para este aspecto se han tenido en cuenta los siguientes medios para recabar la información necesaria:

- Consultar los manuales ofrecidos por el fabricante para este modelo. Donde encontramos las características básicas de comunicación necesarias para que esta se lleve a cabo.
- Revisar otros programas para la manipulación de este robot. Aunque se presenten en otro lenguaje de programación (Mitsubishi en sus manuales realiza ejemplos en BASIC) podemos extraer numerosa información útil de datos de conexión.
- Comprobar las características del administrador de puertos por parte del sistema operativo que nos ocupe. Para evitar posibles fallos de control provocados por el tratamiento del sistema.
- Si es necesario, emplear un software que monitorice la línea RS-232 sin intervenir en el programa para comprobar en tiempo real el estado de esta y las actuaciones que sobre ella se llevan a cabo. También se puede emplear este método para comprobar la actuación que realiza un programa existente que conecta con el robot, si no se dispone del código fuente de dicho programa. En este caso se ha utilizado el programa Serial Port Monitor para comprobar las acciones llevadas a cabo por el ejecutable en base MS-DOS ofrecido por Mitsubishi.

También cabe destacar que, para comprobar el estado correcto de funcionamiento, es conveniente realizar en un primer tiempo una aplicación lo más sencilla posible y mediante la cual se envíe una información fácil de chequear. De esta manera en los primeros compases de la elaboración de este trabajo se realizó un programa sencillo que se conectara con el puerto y pudiera enviar una cadena de caracteres.

El método de comprobación escogido para conocer si era efectiva la comunicación fue el envío de dos órdenes sencillas: ER (mediante el cual el robot devuelve el error actual

presente a través del puerto serie) y OR (el robot se mueve a la posición de origen). Como veremos más adelante se trata de tipos de órdenes distintas, la primera es un comando de comunicación y la segunda de posición.

Mediante este medio y leyendo e interpretando los errores producidos e indicados por la unidad de control se ha podido comprobar los fallos que se presentaban y conocer que aspectos se debían introducir para la elaboración de una conexión efectiva.

Una vez definidos los aspectos más importantes de la conexión es importante destacar los aspectos básicos necesarios en la programación en el lenguaje de Visual Basic para establecer la conexión.

Dentro de nuestro lenguaje de programación hay un componente específico para el control de puertos series de este tipo. Este componente que vamos a utilizar es MSComm.

Este objeto está dotado de las propiedades y eventos necesarios para operar con los puertos serie del ordenador. En la tabla 4 mostramos las propiedades principales del componente MSComm.

Para que el programa reconozca si está conectado o no a un puerto se utilizará una variable contenida en un módulo para que pueda ser utilizada por los distintos fragmentos del código. Esta variable será de tipo Boolean y se denominará “Conectado”. Cuando el puerto se active ésta debe pasar a True y cuando no se encuentre activado a False.

Una vez establecidas que propiedades se van a emplear hay que definir los procedimientos que se tienen que llevar a cabo para trabajar con el puerto.

En primer lugar hay que diseñar una ventana donde actuar para establecer una conexión con el puerto. Para este diseño se ha optado por una ventana sencilla con un ComboBox donde escoger el puerto y un CommandButton para llamar al procedimiento de conexión.

Propiedad	**Descripción**
CommPort	Determina que puerto serie del hardware del equipo va a controlar.
Settings	Establece los ajustes de conexión (tasa de baudios, bit de paridad, número de bits que componen la cadena y bits de parada).
PortOpen	Abre el puerto para su funcionamiento.
RThreshold	Indica el número de caracteres que debe estar presente en el buffer para activar el proceso OnComm.
Handshaking	Determina el medio por el cual se llevará a cabo el control del puerto.
Interval	Indica el tiempo de intervalo entre las comprobaciones del estado de recepción.
CTSTimeout	Tiempo que permanece esperando la señal CTS desde que se activa la señal RTS.
CDTimeout	Tiempo máximo de espera desde que se activa la señal DTR hasta la señal CD (Carrier Detect).
DSRTimeout	Tiempo máximo de espera desde que se activa la señal DTR hasta la señal DSR.
DTREnabled	Activa la salida DTR.
RTSEnabled	Activa la salida RTS.
Output	Envía por la línea TXD los caracteres determinados por el usuario y los bits establecidos según los ajustes.

Tabla 4: Propiedades utilizadas de MSComm

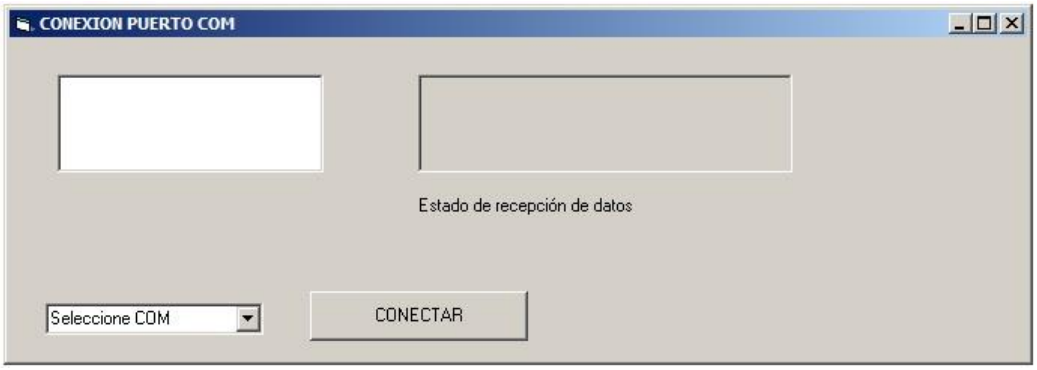


Fig. 11: Aplicación para comprobar la conexión del puerto serie

Para el procedimiento de conexión hay que establecer los valores que determinan como se va a definir la conexión. Esto se llevará a cabo con las propiedades del objeto MSComm encargadas de definir valores de conexión.

También es necesario establecer ciertas comparaciones de seguridad para evitar problemas e informar si ya se ha establecido una conexión con alguno de los puertos serie. Además debe comprobarse si se ha escogido un puerto entre los disponibles en el ComboBox.

El procedimiento utilizado para la conexión queda definido en la siguiente porción de código:

```
Public Sub Conexion()

    If Conectado Then

        MsgBox "Ya está conectado al puerto COM" + CStr(NumeroPuerto), , "Aviso"

    ElseIf cmbsel.ListIndex >= 0 Then

        frmprincipal.puerto.CommPort = Val(cmbsel.ListIndex + 1) ' Selección del puerto a través del index del ComboBox habilitado para selección del puerto
        frmprincipal.puerto.Settings = "9600,e,8,2" ' Establecimiento de los Settings de conexión
        frmprincipal.puerto.PortOpen = True ' Apertura del puerto
        frmprincipal.puerto.RThreshold = 1 ' Indica la iniciación del proceso OnComm() cuando haya al menos un byte en el buffer de entrada
        frmprincipal.puerto.Handshaking = comRTS ' Determinación del medio por el cual se controla el puerto
        frmprincipal.puerto.Interval = 1000
        frmprincipal.puerto.CTSTimeout = 6000 ' Tiempo que permanece esperando la señal CTS desde que se activa la señal RTS
        frmprincipal.puerto.DSRTimeout = 6000 ' Tiempo máximo de espera desde que se activa la señal DTR hasta la señal DSR
        frmprincipal.puerto.CDTimeout = 0 ' Tiempo máximo de espera desde que se activa la señal DTR hasta la señal CD

        Conectado = True ' Establece la variable que indica que está conectado como True indicando que está conectado
        NumeroPuerto = cmbsel.ListIndex + 1 ' Variable que nos va a indicar a qué puerto estamos conectados

        MsgBox "Se ha conectado al puerto COM " + CStr(NumeroPuerto), , "Controlador Mitsubishi RV-E2"

    Else ' Se produce un mensaje en caso de no haber seleccionado ningún puerto del ComboBox
```

```
        MsgBox "Seleccione un puerto COM", , "Aviso"
    End If
End Sub
```

Por último cabe destacar también la importancia de los procedimientos para el envío de datos a través del puerto.

Cómo nos indica la propiedad Handshaking definida, el control de flujo de información se realizará a través de señales auxiliares del puerto mediante el hardware. Este procedimiento se realizará empleando la señal RTS que habrá que activar previamente al envío de información.

Consecuentemente, en el procedimiento por el cual se desee enviar datos entre la unidad de control y el ordenador al que estará conectada, habrá que hacer un procedimiento previo que es la activación de la señal RTS (para verificar que se puede enviar la información) y la señal DTR (para indicar que se desea enviar información a la unidad). Y tras el envío de información desactivar la señal DTR, de esta manera se evita que la unidad espere recibir más información cuando no va a enviarse. En el código aparecerá de esta manera:

```
frmprincipal.puerto.DTREnable = True
frmprincipal.puerto.RTSEnable = True
```

Por otro lado el tratamiento de la información por parte de la unidad del robot RV-E2 que estamos tratando en este caso, requiere que después de un bloque de información (una orden completa o una línea de programa) se envíe el carácter que indica un retorno de carro (OD en sistema hexadecimal o el carácter 13 en código ASCII). De esta forma puede llevar un correcto tratamiento de la información. El código escribirá:

```
puerto.Output = Chr(13)
```

4. INTERFAZ DE USUARIO PARA EL CONTOL DE ROBOT MITSUBISHI RV-E2

Una vez determinadas las propiedades para establecer la conexión es necesario incluirlas en una interfaz donde un usuario pueda tratar de una manera clara, sencilla y cómoda con el envío y recepción de datos. De esta manera se podrán aprovechar al máximo las posibilidades del robot.

Para determinar una manera fácil y sencilla de llevar a cabo nuestros objetivos es necesario conocer las características de Visual Basic y ajustarse a las posibilidades de la comunicación PC-Robot.

Lo primero que tenemos que tener en cuenta es que el lenguaje de programación Visual Basic se centra en la creación de formularios (ventanas) que contienen los objetos con los que puede interaccionar el usuario. Por tanto es necesario determinar que objetos y que propiedades particulares deben estar presentes para llevar a cabo las distintas tareas. También es importante determinar que tareas se desean llevar a cabo y una vez determinadas decidir si han de compartir formulario o se presenta de forma más sencilla si se encuentran en formularios distintos.

Uno de los aspectos más interesantes de realizar una conexión PC-Robot es la posibilidad de crear programas que se almacenan en la memoria de la unidad de control y pueden ser ejecutados a posteriori por el robot. Por ello se ha considerado que la ventana principal contenga un cuadro de texto donde pueda escribirse este programa. También debe contener un objeto que llame al procedimiento por el cual se envía el programa.

Además, considerando que el formulario principal es la ventana donde se producirá mayor interacción con el usuario, se establecerá allí la información que se considera más interesante para ser observada. En dicho formulario se establecerá una lista donde se recoja la información enviada por la unidad al ordenador y se colocará también un texto que indique en que puerto se ha establecido la conexión o si no está conectado.

También por ser la ventana principal se colocarán en ella los distintos menús y submenús que den acceso a las demás operaciones y posibilidades que presente la aplicación.

El listado de los menús es el siguiente:

Menú	**Submenú**	**Descripción**
Proyecto	Nuevo Proyecto	Activa el cuadro de texto para escribir un programa y/o lo deja en blanco.
	Guardar	Guarda un archivo de texto con el programa escrito.
	Abrir	Escribe en el cuadro de texto un archivo de texto previamente guardado en la memoria del ordenador.
	Trabajar Online	Muestra el formulario para trabajar online.
	Torres de Hanói	Muestra el formulario para crear un programa tipo que permita al robot resolver el problema de las torres de Hanói.
Editar	Borrar Tabla de Lectura de Datos	Borra los datos recogidos en la tabla que muestra la información enviada por la unidad de control.
	Numerar Programa	Numera con paso 10 las líneas escritas en el cuadro de texto del programa.
Serial	Conexión del Puerto Serie	Muestra el formulario que permite abrir un determinado puerto.
	Desconexión del Puerto Serie	Cierra el puerto en caso de que esté alguno abierto.
Información		Muestra un formulario con información referente al programa.

Tabla 5: Menús y submenús del formulario principal

También es importante destacar que deben colocarse otros dos objetos que han de estar presentes para el funcionamiento de la aplicación pero que no serán visibles para el usuario:

- MSComm: Es el objeto que posibilita la manipulación del puerto serie presente en el hardware del equipo.
- CommonDialog: Permite trabajar con cuadros de diálogo mediante los cuales se puede entre otras opciones abrir y guardar archivos presentes en el equipo.

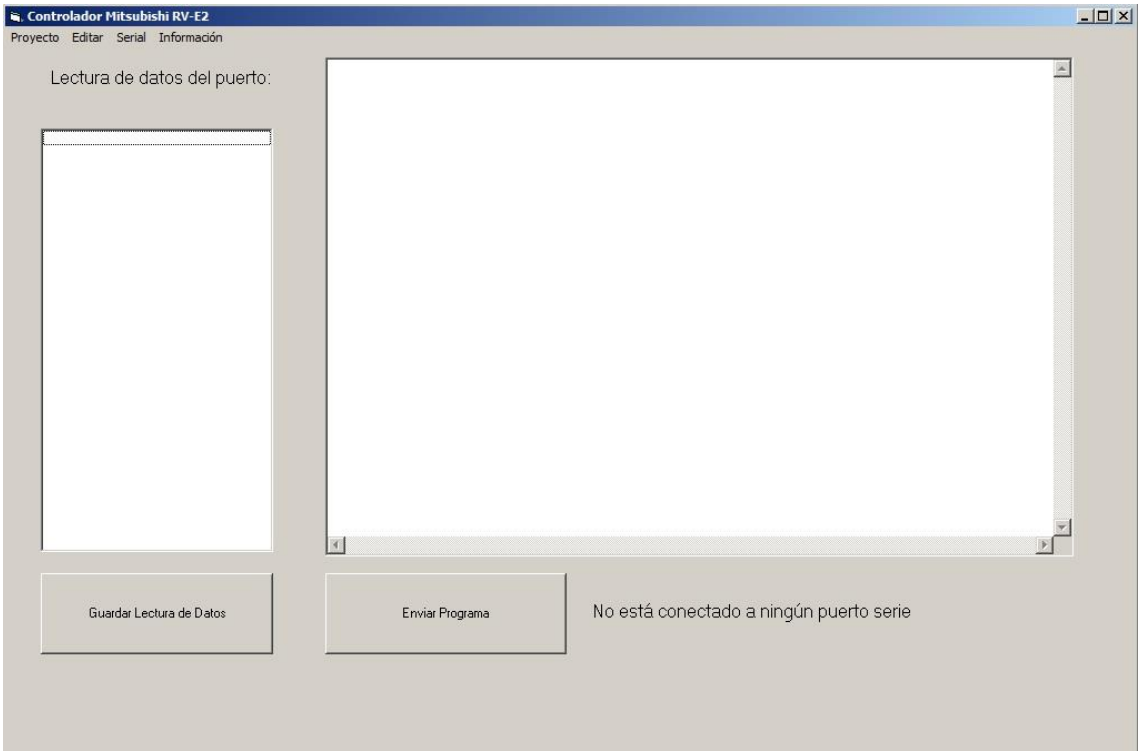


Fig. 12: Aspecto del formulario principal

Los principales procesos que se van a realizar a través de este formulario son:

- Abrir el resto de formularios presentes en la aplicación.
- Leer los datos de entrada del puerto serie al que se conecte.
- Guardar un archivo de texto con los datos recibidos.
- Borrar los datos presentes en la tabla de lectura de datos recibidos.
- Enviar el programa escrito en el cuadro de texto al robot a través del puerto serie.
- Abrir un archivo de texto en el cuadro de texto donde se escriben los programas.

- Guardar en un archivo de texto lo contenido en el cuadro de texto.
- Numerar las distintas líneas que estén escritas de programa.

También se ha considerado que cerrando esta ventana se debe salir de la aplicación y por tanto finalizar el resto de formularios que estén activos. Para evitar la pérdida de datos de un posible programa que no haya sido guardado en un cierre accidental del formulario, se abre un cuadro de diálogo para confirmar que el cierre es voluntario o si se desean guardar los datos antes de cerrar.

El siguiente formulario que se debe considerar es el que realiza la conexión al puerto serie. Se ha optado por separar esta función del formulario principal por dos razones. La primera es que se puede desear trabajar en la realización de un programa o en la revisión de este, sin necesidad de estar conectado por el puerto serie. El segundo caso es que así se dificulta que se pueda realizar un cambio de puerto de manera indeseada.

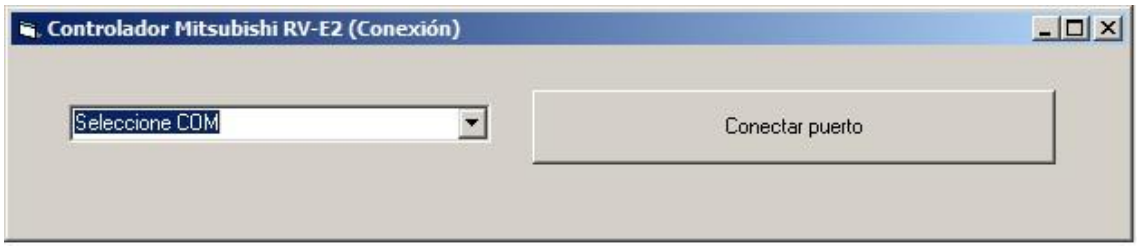


Fig. 13: Aspecto del formulario de conexión

El proceso que se realiza en esta ventana es la conexión, el cual quedó definido anteriormente al tratar las propiedades de la conexión. Además se han incluido dos métodos para llamar a este procedimiento: por medio de la tecla enter del teclado o a través del CommandButton presente en el formulario.

Otro de las posibilidades de la conexión PC-Robot es el envío de órdenes directas. Para trabajar de esta manera se ha insertado otro formulario centrado en esta función. El formulario presenta un cuadro de texto donde se escribe la orden que se desea enviar y un botón para su envío (también se puede enviar a través de la tecla enter). Además de estos dos objetos se ha introducido una lista donde se registran los datos (caracteres) que se envían a través del puerto serie, así como dos botones para poder guardar este registro o borrarlo.

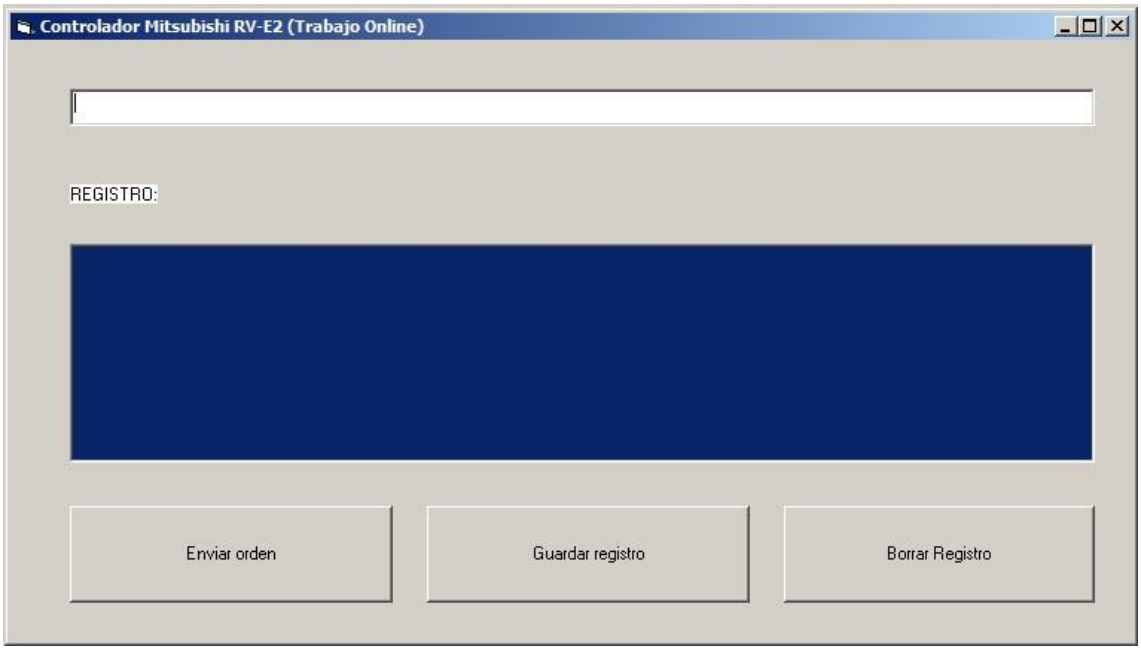


Fig. 14: Aspecto del formulario de envío de órdenes directas

Cabe destacar que el registro numera los datos guardados. Dicha numeración se realiza con paso 10 para que tenga aspecto de programa en caso de que quiera abrirse como tal directamente, reproduciendo un programa todos los pasos realizados por el usuario de manera directa.

Dentro de las posibilidades del robot, en el caso de este trabajo en concreto se ha decidido que con el fin de demostrar el funcionamiento y las capacidades de esta unidad, sea capaz de realizar el problema de las torres de Hanói, como ya se ha mencionado. Para esta aplicación se hará un programa de funcionamiento del robot Mitsubishi mediante el cual se realice este problema. Para ello se ha elaborado un formulario que determina el programa del robot en base a unos parámetros iniciales. Estos parámetros se recogen en la siguiente tabla:

Posición 1 (Posición de inicio)	Es la posición en la que la herramienta del robot se encuentra centrada sobre la varilla donde en el estado inicial se encontrará la torre.
Posición 2	Es la posición en la que la herramienta del robot se encuentra centrada sobre la segunda varilla.
Posición 3	Es la posición en la que la herramienta del robot se encuentra centrada sobre la tercera varilla.
Altura de varilla	Es la altura desde la posición centrada que se determine en cada varilla al pie de la varilla (esta distancia debe coincidir en los tres casos).
Altura del disco	Es la altura que tienen los discos que se están empleando.
Número de discos	Indica el número de discos en la torre inicial, es decir, para los que se realizará el problema.

Tabla 6: Parámetros iniciales para resolver el problema de las torres de Hanói.

Cabe destacar que los parámetros de posición deben estar escritos en el formato por el cual quedan definidos para la posición del robot en ese punto. También es importante una correcta medición de la altura de las varillas desde la posición vertical definida en los tres casos, ya que esto puede ocasionar una colisión en algún punto del recorrido o un fallo a la hora de recoger un disco.

La aplicación que aquí se ha creado define un programa que permite al robot realizar esta tarea. Esto es posible ya que la aplicación determina según los parámetros iniciales los movimientos que debe realizar el robot para la resolución del problema y los escribe en un formato de órdenes que pueden convertirse en un programa para su envío al robot.

Esta forma de definir la aplicación nos permite una gran versatilidad a la hora de realizar la tarea ya que no se basa en unos movimientos cerrados para un caso particular,

sino que nos permite resolver el problema en cualquier caso que entre dentro de las posibilidades del robot. Esto quiere decir que no es exclusivo de unas posiciones concretas de la varilla o un número determinado de discos, aunque esto nos exige una mayor exactitud de las condiciones iniciales.

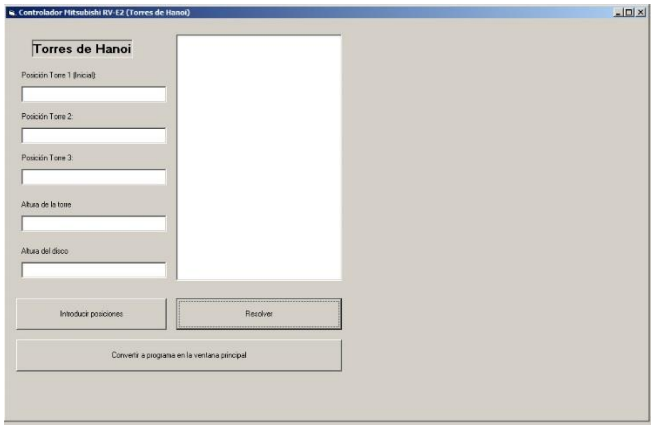


Fig. 15: Aspecto del formulario para la resolución de las torres de Hanói

Una vez definido el programa del robot, escrito en la lista de este formulario, el programa puede ser enviado al cuadro de texto del formulario principal mediante un botón. De esta forma ya podrá ser tratado, introducirse alguna modificación y/o enviarse a la unidad del robot.

Por último también se ha introducido un formulario con información referente a este Trabajo Fin de Grado, a la cual se puede acceder a través del menú información del formulario principal.

Fig. 16: Aspecto del formulario de información

5. CONCLUSIONES

Este apartado nos sirve como epilogo de todo el documento y se podrá evaluar si se ha cumplido con los objetivos establecidos en este Trabajo Fin de Grado. Vamos a enumerar las distintas conclusiones que se puedan extraer.

En primer lugar se ha podido crear una aplicación a través del código de Visual Basic 6.0 que permite establecer una comunicación bidireccional y funcional entre el PC y el robot Mitsubishi RV-E2 a través del puerto serie. Esto se ha podido llevar a cabo gracias al tratamiento de la comunicación serie ofrecido por el objeto de Visual Basic MSComm.

También cabe destacar que el tratamiento para el intercambio de información se realiza de forma visual y sencilla por la elaboración de distintos formularios mediante los que se intercambia información. En estos formularios podemos realizar el envío de órdenes que gobiernan el robot por los dos medios que admite: programas numerados y órdenes directas. Además se ha introducido un espacio donde se puede observar la información emitida desde la unidad de control del robot que no interfiere con el trabajo del usuario.

Por otra parte se ha añadido un formulario a la aplicación que permite la elaboración de una lista de órdenes del robot que permiten la resolución física del problema de las torres de Hanói. Este formulario permite la introducción de unas condiciones iniciales que le otorgan flexibilidad y la posibilidad de cumplir con esta tarea en diversas condiciones (cambio de posición de las varillas, cambio de discos, cambio de número de elementos, etc.).

Por último se puede afirmar que la aplicación se ha testado con éxito, incluida la comunicación con en el robot Mitsubishi RV-E2 presente en las instalaciones de la Escuela Politécnica Superior de Zamora perteneciente a la Universidad de Salamanca.

Con todo esto se puede concluir que los objetivos previstos para este caso han sido cumplidos de forma satisfactoria.

6. BIBLIOGRAFÍA

Libros de consulta

Mitsubishi Electronic Corporation (1994). *Mitsubishi Industrial Robot RV-E2 Movemaster Super User's Manual.*

Mitsubishi Electronic Corporation (1995). *Movemaster Industrial Robot Specification Manual.*

Mitsubishi Electronic Corporation (1994). *Mitsubishi Industrial Robot RV-E2 Movemaster Super Reference Manual.*

Stalling, W. (2004). *Comunicación y Redes de Computadores* (Séptima Edición). Madrid. Prentice Hall.

Atiken, Peter G. (1999). *Visual Basic 6: Manual Completo de Programación*. Madrid. Paraninfo.

Páginas Web

- Comunicación serie. Disponible en: http://digital.ni.com/public.nsf/allkb/039001258CEF8FB686256E0F005888D1
- Estandar RS-232C. Disponible en: http://www.euskalnet.net/shizuka/rs232.htm
- Servicios web de Microsoft. Disponible en: https://msdn.microsoft.com/es-es/
- Propiedades y eventos del objeto MSComm. Disponible en: http://uttinfor.tripod.com/index/id4.html
- Robótica. Disponible en: www.monografías.com

7. ANEXOS

ANEXO I: Código del Formulario Principal

```
Private Sub CmdGuardarLectura_Click()

    Call GuardarLectura

End Sub


Private Sub cmdenviarprograma_Click()

    If Conectado Then

        Call EnviarPrograma

    Else

        MsgBox "El programa no se pudo enviar", , "Aviso"

    End If

End Sub

Private Sub mnuabrir_Click()

    If txtprograma.Visible = False Then

        Call Abrir

    ElseIf txtprograma.Text <> "" And Not Guardado Then

        Dim ConfirmarAbrir As Integer
        ConfirmarAbrir = MsgBox("Los datos existentes se perderán", vbOKCancel,
"Aviso")

        If ConfirmarAbrir = vbOK Then

            Call Abrir

        End If

    Else

        Call Abrir

    End If

End Sub

Private Sub mnuBorrarLectura_Click()

    Dim confirmarBorrar As Integer
    confirmarBorrar = MsgBox("Se borraran los datos de lectura de puerto",
vbOKCancel, "Aviso")

    If confirmarBorrar = vbOK Then
```

```
        List1.Clear

    End If

End Sub

Private Sub mnuconexion_Click()

    frmConectar.Show 'Abre la ventana para la conexión del puerto

End Sub


Private Sub mnuDesconexion_Click()

    Call Desconexión

    If Conectado Then

        lblestadoCOM.Caption = "Conectado al puerto COM" + CStr(NumeroPuerto)

    Else

        MsgBox "Se ha desconectado el puerto serie", , "Aviso"
        lblestadoCOM.Caption = "No está conectado a ningún puerto serie"

    End If

End Sub

Private Sub mnuHanoi_Click()

    FormHanoi.Show

End Sub

Private Sub mnuinformacion_Click()

    frmInformacion.Show

End Sub

Private Sub mnuNuevo_Click()

    If txtprograma.Text <> "" And Not Guardado Then

        Dim ConfirmarAbrir As Integer
        ConfirmarAbrir = MsgBox("Los datos existentes se perderán", vbOKCancel,
"Aviso")

        If ConfirmarAbrir = vbOK Then

            txtprograma.Visible = True
            txtprograma.Text = ""
            cmdenviarprograma.Visible = True
            lblestadoCOM.Visible = True
```

```
        End If

    Else

        txtprograma.Visible = True
        txtprograma.Text = ""
        cmdenviarprograma.Visible = True
        lblestadoCOM.Visible = True

    End If

End Sub

Private Sub mnuNumerar_Click()

    Dim DatoDeNumeracion() As String
    Dim TextoNumerado As String
    Dim LineaANumerar As Integer
    DatoDeNumeracion = Split(txtprograma, vbCrLf)

    TextoNumerado = ""

    For LineaANumerar = 0 To UBound(DatoDeNumeracion)

        If DatoDeNumeracion(LineaANumerar) <> "" Then

            TextoNumerado = TextoNumerado + CStr((LineaANumerar + 1) * 10) + "
" + DatoDeNumeracion(LineaANumerar) + vbCrLf

        End If

    Next

    txtprograma.Text = TextoNumerado

End Sub

Private Sub mnuOnline_Click()

    If Conectado Then

        frmOnline.Show

    Else

        MsgBox "No está conectado", , "Aviso"

    End If

End Sub

Private Sub smnuguardar_Click()

    If Guardado Then

        MsgBox "El programa ya está guardado"

    Else
```

```
        Call Guardar

    End If

End Sub

Public Sub txtprograma_change()

    Guardado = False

End Sub

Private Sub txtprograma_GotFocus()

    If Conectado Then

        lblestadoCOM.Caption = "Conectado al puerto COM" + CStr(NumeroPuerto)

    Else

        lblestadoCOM.Caption = "No está conectado a ningún puerto serie"

    End If

End Sub

Private Sub puerto_oncomm()

    Dim Textin As String

    Select Case puerto.CommEvent
        Case comEvReceive
            Textin = puerto.Input ' Define la cadena de caracteres que se va a
recibir
            If Textin <> "" Then ' Condiciona que la cadena de caracteres no
sea nula

                List1.AddItem Textin

            End If
    End Select

End Sub
Private Sub Form_Unload(Cancel As Integer)

    Dim SalirSinGuardar As Integer

    If txtprograma.Text = "" Then

    ElseIf Guardado Then

    Else

        SalirSinGuardar = MsgBox("¿Desea guardar los cambios?", vbYesNoCancel,
"Aviso")

        If SalirSinGuardar = vbYes Then

            Call Guardar
```

```
            If Guardado Then

            Else

                Cancel = 1 'Si ha habido un problema con el guardado no cierra
el programa

            End If

        ElseIf SalirSinGuardar = vbNo Then

        Else

            Cancel = 1 ' Si se presiona Cancelar o cerrar la ventana no sale
del programa

        End If

    End If

    If Cancel = 0 Then

        Unload frmConectar 'Cierra si está abierta la ventana de conexión
        Unload frmOnline ' Cierra si está abierta la ventana de Trabajo Online
        Unload FormHanoi ' Cierra si está abierta la ventana de Torres de Hanoi

    End If


End Sub

Private Sub GuardarLectura()

    If List1.List(0) = "" Then

        MsgBox "No hay datos en el registro", , "Aviso"

    Else

        CommonD.Flags = cdlOFNOverwritePrompt + cdlOFNHelpButton
        CommonD.Filter = "Archivos de texto |*.txt"
        CommonD.ShowSave

        Dim LecturaGuardar As String
        Dim BucleLectura As Integer
        Dim NumeroDeLecturas As Integer
        NumeroDeLecturas = List1.ListCount

        If CommonD.FileName <> "" Then

            Open CommonD.FileName For Output As #4

            For BucleLectura = 0 To NumeroDeLecturas

                LecturaGuardar = List1.List(BucleLectura)
                Print #4, LecturaGuardar

            Next
```

```
            Close (4)
            LecturaGuardar = ""

        End If

    End If

End Sub

Private Sub Desconexión()

    puerto.PortOpen = False ' Cierra el puerto
    NumeroPuerto = 0
    Conectado = False '  Establece la variable que indica que está conectado
como False indicando que no está conectado.

End Sub

Private Sub Abrir()

    txtprograma.Visible = True
    txtprograma.Text = ""
    cmdenviarprograma.Visible = True
    lblestadoCOM.Visible = True

    CommonD.Flags = cdlOFNHelpButton
    CommonD.Filter = "Archivos de texto |*.txt"
    CommonD.ShowOpen
    If CommonD.FileName <> "" Then
        Open CommonD.FileName For Input As #2
        Do While Not EOF(2)
            Line Input #2, linea
            txtprograma.Text = txtprograma.Text + linea + vbCrLf
        Loop
        Close (2)
    End If


End Sub
Private Sub Guardar()

    If txtprograma.Text = "" Then

        MsgBox "El proyecto está en blanco", , "Aviso"

    Else

        CommonD.Flags = cdlOFNOverwritePrompt + cdlOFNHelpButton
        CommonD.Filter = "Archivos de texto |*.txt"
        CommonD.ShowSave

        If CommonD.FileName <> "" Then

            Open CommonD.FileName For Output As #1
            Print #1, txtprograma.Text
            Close (1)

             Guardado = True
```

```
        End If

    End If

End Sub
Private Sub EnviarPrograma()

    If Conectado Then

        Dim Np As String ' Valor que se introduce para definir el número de
programa
        Dim Npi As Integer ' Valor para la comprobación del número de programa
dentro del bucle for
        Dim Npc As Boolean  ' Valor para comprobar que el valor introducido
coincide con un valor válido

        Np = InputBox("Número de programa: ", "Número de programa", "1")

        If Np = "" Then ' Si no hay nada escrito o se presiona Cancelar

            MsgBox "Programa no enviado", , "Aviso"
            Npc = True

        Else  ' En caso de que haya algo escrito y se acepta se conprueba que
sea un dato válido

            For Npi = 1 To 99

                If Np = CStr(Npi) Then

                    Npc = True
                    frmprincipal.puerto.DTREnable = True
                    frmprincipal.puerto.RTSEnable = True

                    puerto.Output = "N" + Np + Chr(13)
                    puerto.Output = "DL" + Chr(32) + "1," + Chr(32) + "9999" +
Chr(13)

                    Dim DatoDeEnvio() As String
                    Dim LineaDePrograma As Integer
                    DatoDeEnvio = Split(txtprograma, vbCrLf)

                    For LineaDePrograma = 0 To UBound(DatoDeEnvio)

                        If DatoDeEnvio(LineaDePrograma) <> "" Then

                            puerto.Output = DatoDeEnvio(LineaDePrograma)
                            puerto.Output = Chr(13)

                        End If

                    Next

                End If

            Next

        End If
```

```
        If Not Npc Then

            MsgBox "El número de programa introducido no es un valor válido" +
vbCrLf + "1 a 99", , "Aviso"
            Call EnviarPrograma

        End If

    End If

End Sub
```

ANEXO II: Código Formulario de Conexión

```
Private Sub cmbsel_KeyPress(KeyAscii As Integer) ' Llama al proceso de
determinar la conexión si se aprieta el ENTER en el teclado

    If KeyAscii = 13 Then

        Call Conexion

    End If

End Sub

Private Sub cmdConectar_Click() ' Llama al proceso de determinar la conexión
si se hace click en el botón

    Call Conexion

End Sub

Public Sub Conexion() ' Determina si se puede conectar un puerto, que puerto
se ha de conectar o avisa si ya está conectado

    If Conectado Then

        MsgBox "Ya está conectado al puerto COM" + CStr(NumeroPuerto), ,
"Aviso"

    ElseIf cmbsel.ListIndex >= 0 Then

        frmprincipal.puerto.CommPort = Val(cmbsel.ListIndex + 1) ' Selección
del puerto a traves del index del Combobox habilitado para selección del
puerto
        frmprincipal.puerto.Settings = "9600,e,8,2" ' Establecimiento de los
Settings de conexión
        frmprincipal.puerto.PortOpen = True ' Apertura del puerto
        frmprincipal.puerto.RThreshold = 1 ' Indica la iniciación del proceso
OnComm() cuando haya al menos un byte en el buffer de entrada
        frmprincipal.puerto.Handshaking = comRTS ' Determinación del medio
por el cual se controla el puerto
        frmprincipal.puerto.Interval = 1000
        frmprincipal.puerto.CTSTimeout = 6000 ' Tiempo que permanece
esperando la señal CTS desde que se activa la señal RTS
        frmprincipal.puerto.DSRTimeout = 6000 ' Tiempo máximo de espera desde
que se activa la señal DTR hasta la señal DSR
        frmprincipal.puerto.CDTimeout = 0 ' Tiempo máximo de espera desde que
se activa la señal DTR hasta la señal CD

        Conectado = True ' Establece la variable que indica que está
conectado como True indicando que está conectado
        NumeroPuerto = cmbsel.ListIndex + 1 ' Variable que nos va a indicar a
qué puerto estamos conectados

        MsgBox "Se ha conectado al puerto COM " + CStr(NumeroPuerto), ,
"Controlador Mitsubishi RV-E2"

    Else ' Se produce un mensaje en caso de no haber seleccionado ningún
puerto del ComboBox
```

```
        MsgBox "Seleccione un puerto COM", , "Aviso"
    End If
End Sub
```

ANEXO III: Código Formulario de Trabajo Online

```
Private Sub cmdBorrarRegistro_Click()

    Call BorrarRegistro

End Sub

Private Sub cmdEnviar_Click()

    Call EnviarOnline

End Sub

Private Sub cmdGuardarRegistro_Click()

    Call GuardarRegistro

End Sub

Private Sub txtenviar_KeyPress(KeyAscii As Integer)

    If KeyAscii = 13 Then

        Call EnviarOnline

    End If

End Sub
Private Sub Form_Unload(Cancel As Integer)

    registro = 0

End Sub
Public Sub EnviarOnline()

    If Conectado Then

        Dim TextOut As String
        Dim l As Integer
        Dim m As Integer
        TextOut = txtenviar.Text ' Define la cadena de caracteres que se va a
enviar a traves de lo escrito en el TextBox
        l = Len(TextOut)
        frmprincipal.puerto.DTREnable = True
        frmprincipal.puerto.RTSEnable = True

        For m = 1 To l

            frmprincipal.puerto.Output = Mid(TextOut, m, 1) ' Envía la cadena
de caracteres definida

        Next

        frmprincipal.puerto.Output = Chr(13)
        frmprincipal.puerto.RTSEnable = False

        registro = registro + 1
        lstRegistro.AddItem (CStr(registro * 10) + " " + TextOut)
```

```
        txtenviar.Text = ""

    Else

        MsgBox "No se ha conectado a un puerto serie", , "Aviso"

    End If

End Sub

Private Sub GuardarRegistro()

    If lstRegistro.List(0) = "" Then

        MsgBox "No hay datos en el registro", , "Aviso"

    Else

        frmprincipal.CommonD.Flags = cdlOFNOverwritePrompt + cdlOFNHelpButton
        frmprincipal.CommonD.Filter = "Archivos de texto |*.txt"
        frmprincipal.CommonD.ShowSave
        Dim RegistroGuardar As String
        Dim BucleRegistro As Integer

        If frmprincipal.CommonD.FileName <> "" Then

            Open frmprincipal.CommonD.FileName For Output As #3

            For BucleRegistro = 0 To registro

                RegistroGuardar = lstRegistro.List(BucleRegistro)
                Print #3, RegistroGuardar

            Next

            Close (3)
            RegistroGuardar = ""

        End If

    End If

End Sub

Private Sub BorrarRegistro()

    Dim PreguntarSiBorrar As Integer
    PreguntarSiBorrar = MsgBox("Se perderán los datos del registro ¿Borrar
datos?", vbYesNo, "Aviso")

    If PreguntarSiBorrar = vbYes Then

        lstRegistro.Clear
        registro = 0

    End If

End Sub
```

ANEXO IV: Código Formulario Torres de Hanói

```
Private Sub cmdConvertir_Click() ' Pasa el texto al cuadro de programa de la
ventana principal

    Dim BucleTexto As Integer
    Dim NumerodeLineas As Integer
    Dim Texto As String


    NumerodeLineas = List1.ListCount

        For BucleTexto = 0 To NumerodeLineas

            Texto = Texto & List1.List(BucleTexto) & vbCrLf

        Next

    frmprincipal.txtprograma.Visible = True

    If frmprincipal.txtprograma.Text <> "" And Not Guardado Then ' Evita que
se borren los datos si existiaran de un programa en la ventana principal que
no esté guardado

        Dim ConfirmarEnviarHanoi As Integer
        ConfirmarEnviarHanoi = MsgBox("Los datos del programa de la ventana
principal se perderán. ¿Desea continuar?", vbYesNo, "Aviso")

        If ConfirmarEnviarHanoi = vbYes Then

        frmprincipal.txtprograma.Text = Texto

        End If

    Else

        frmprincipal.txtprograma.Text = Texto

    End If

End Sub

Private Sub Command1_Click()

    List1.Clear ' Elimina los datos de posibles casos anteriores

    List1.AddItem "PD 1," + Text1.Text
    List1.AddItem "PD 2," + Text2.Text
    List1.AddItem "PD 3," + Text3.Text

End Sub

Private Sub Resolver_Click()
    'definición de parametros: a-número de discos, x-contador de movimientos
realizados, j-disco a mover

    Dim a As Integer, x As Integer, j As Integer
    'posi() matriz de memoria; posi(j,0) posición origen disco j; posi(j,1)
posición destino; posi(j,2) número de veces que se ha movido el disco j
```

```
    a = InputBox("número de discos")

    Dim posi() As Integer
    ReDim posi(a, 2)

    For x = 1 To (2 ^ a) - 1

        Call recursividad(x, j, a, posi())

    Next x

    List1.AddItem "ED"

End Sub

Sub recursividad(y As Integer, ByRef i As Integer, b As Integer, ByRef
posii() As Integer)
    i = 0
    Do
        i = i + 1

    Loop Until ((2 ^ (i - 1)) + posii(i, 2) * 2 ^ i = y) ' queda definida i
como el disco que corresponde al movimento que se va a realizar

    posii(i, 2) = posii(i, 2) + 1 ' Contador de movimientos realizados por
dicho disco

    ' posii (i,0) es la posición desde la que parte el disco
    ' posii (i,1) es la posición a la que se moverá el disco
    ' al empezar este proceso el disco se encuentra en la posición definida
por posii(i,1) ya que es donde se llevó en el movimiento anterior

    Dim DiscosenZ As Integer
    Dim p As Integer
    DiscosenZ = 0

    For p = b To i Step -1 ' Define el número de discos que hay en la torre
de donde se va a tomar el disco en este paso

        If posii(p, 1) = posii(i, 1) Then

            DiscosenZ = DiscosenZ + 1

        End If

    Next p

    posii(i, 0) = posii(i, 1)

    If i Mod 2 = 0 Then

        posii(i, 1) = posii(i, 0) + 2

    Else

        posii(i, 1) = posii(i, 0) + 1

    End If
```

```
    If posii(i, 1) >= 3 Then

        posii(i, 1) = posii(i, 1) - 3

    End If

    Dim MovimientoZ As Single
    MovimientoZ = CSng(txtAlturaTorre.Text) - (CSng(DiscosenZ) - 1) *
CSng(txtAlturaDisco.Text)

    List1.AddItem "MO " + CStr(posii(i, 0) + 1) + ",O"
    List1.AddItem "DS 0,0,-" + CStr(MovimientoZ)
    List1.AddItem "GC"
    List1.AddItem "MO " + CStr(posii(i, 0) + 1) + ",C"
    List1.AddItem "MO " + CStr(posii(i, 1) + 1) + ",C"
    List1.AddItem "DS 0,0,-4"
    List1.AddItem "GO"
    List1.AddItem "MO " + CStr(posii(i, 1) + 1) + ",O"

End Sub
```