Proyecto: Juego Tres en Raya

Asignatura: Sistemas Robotizados
Fecha de entrega: 10/12/2017

Carlos Prados Sesmero 70830147z

Índice

Introducción

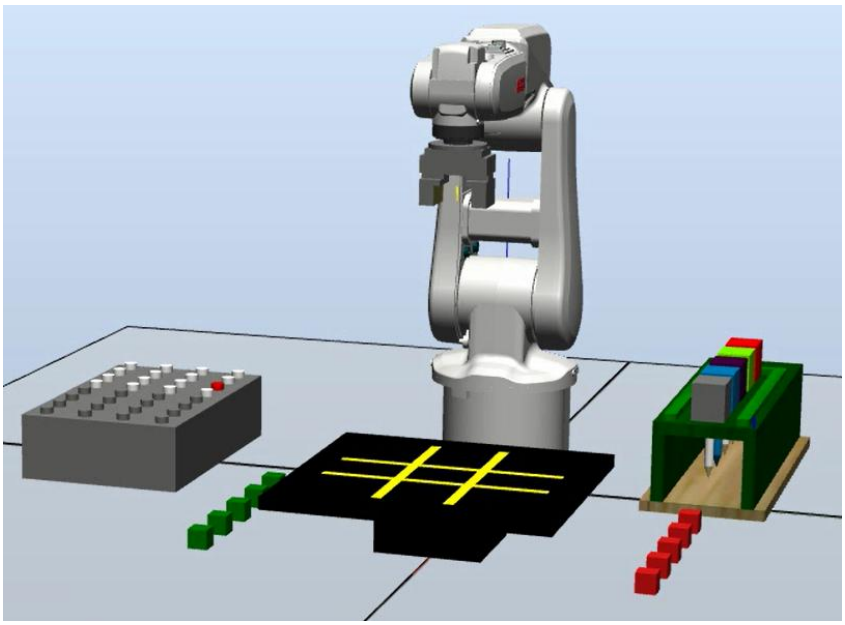
Se presenta el proyecto llamado "Tres en Raya" como el resultado del siguiente enunciado: Realización de un programa en RAPID que permita jugar al "tres en raya", jugando un máximo de 2 partidas.

Para la realización de dicho proyecto deberemos considerar las siguientes pautas dentro de la flexibilidad del proyecto:

- Todas las posiciones (robtarget) a las que se mueve el robot en un plano de trabajo (wobjdata) deben estar referidas a determinados puntos, por lo que definiremos algunos puntos (como pueden ser inicio y final de búsqueda) como puntos de referencia.
- Se deben definir tantos planos de trabajo (wobjdata), como "zonas de trabajo" en las que el robot realice sus movimientos. Solo será necesario describir un único plano de trabajo (mesanegra).
- Cada posición (robtarget) que se utilice debe estar definida con respecto al plano de trabajo (wobjdata) al que pertenece.
- Es obligatorio estructurar el código RAPID mediante la utilización de rutinas (PROC o FUNC) con paso de parámetros.
- Es obligatorio utilizar una o varias rutinas de tratamiento de interrupción.

Hemos desarrollado dos programas en RAPID distintos, los cuales se incluyen junto a este documento en el archivo entregado. Uno de ellos será el que podremos subir al robot real y funcione correctamente en dicho robot, el otro será un archivo de simulación, el cual en el robot real no funcionaria puesto que los puntos de referencia tomados son distintos y produciríamos choques en el robot. De manera que este segundo archivo nos valdrá para comprobar el funcionamiento en la estación de Robot Studio. También cabe destacar que el programa de simulación tiene alguna otra modificación para que funcione correctamente a la hora de simularlo.

Para el desarrollo del proyecto hemos partido de la siguiente estación en Robot Studio:



Definición del proyecto

Antes de describir el proyecto completo parte por parte, vamos a establecer algunas consideraciones necesarias para comprender el funcionamiento del proyecto y su estructura. Estas consideraciones serán las siguientes:

1. En el juego "tres en raya" como ya sabemos es un juego en el que hay 2 participantes. De esta manera vamos a distinguir el participante "Robot" que jugará con las fichas verdes y el participante "Usuario" que jugará con las fichas rojas, colocadas a la derecha e izquierda del robot respectivamente.

2. Únicamente realizará movimientos de fichas el Robot ABB IRB 120, que será el encargado de realizar todo el proceso de colocación de fichas en los diferentes puntos de su espacio de trabajo.

3. Todas y cada una de las trayectorias del Robot para coger y dejar las fichas, deberán comenzar y terminar en una posición cómoda para nuestro IRB 120, denominada "reposo". Esta posición será tal que los ejes del robot sean 0º salvo el eje 5 que se encontrará a 90º.

4. Distinguiremos dos formas de trabajo: El movimiento y colocación de las fichas verdes y el de las fichas rojas. Cada uno de ellos se caracteriza por las siguientes pautas:

    a. Colocación de las fichas verdes: Como hemos mencionado anteriormente las fichas verdes se encuentran a la derecha del robot (vistas desde el robot). Para conocer la localización de dichas fichas disponemos de un fichero llamado "verdes.txt" (variable y a modificar a nuestro antojo) en el que se definirán mediante letras (A, B, C, D y E) dichas posiciones. Dependiendo de cuales sean las letras encontradas en este fichero, el robot deberá ir a diferentes posiciones para coger cada una de las fichas. La posición A es la más próxima al robot y serán las letras posteriores las que definan el resto de piezas, estando cada una de las piezas separadas por una distancia de 80mm (distancia variable a nuestro antojo). Esto quiere decir que si, por ejemplo, nuestro fichero tiene el contenido "A B D E", habrá 4 fichas verdes colocadas en las posiciones: primera, segunda, cuarta y quita. De esta manera y como es lógico al inicio del juego podrá haber 3, 4 o 5 fichas verdes, que estarán situadas en alguna de las 5 posiciones posibles. Podremos encontrar y modificar nuestro fichero "verdes.txt" en la carpeta "HOME".
    b. Colocación de las fichas rojas: Como hemos mencionado anteriormente las fichas rojas se encuentran a la izquierda del robot (vistas desde el robot). Al inicio del juego podremos tener 3, 4 o 5 fichas rojas que estarán colocadas a nuestro antojo en una línea definida por la que pasará nuestro robot a buscar cada una de ellas. Para ello el robot necesitará tener un proceso de búsqueda, comenzando por una posición inicial definida (la más cercana al robot) y terminando en una posición final también definida (que será la más lejana al robot). Para realizar dicha búsqueda será necesaria la ayuda de la pinza ubicada en el robot (Será nuestra

herramienta de trabajo durante toda la práctica) y la ayuda de un sensor detector de objetos unido a la entrada digital “di16” de la que el robot tendrá conocimiento en todo momento.

5. El comienzo del juego lo realizará o el Usuario o el Robot. Esta premisa quedará por determinar dependiendo de la elección del Usuario, al que se le mostrará por la pantalla de la Flexpendant un diálogo que le dará a elegir quién empezará la partida.

6. Para la localización de piezas también podemos distinguir entre Usuario y Robot:

    a. Cuando sea Usuario el encargado de poner una ficha el procedimiento será el siguiente: Una vez encontrada la pieza roja a localizar nos desplazaremos a la posición de reposo. Detendremos el movimiento del robot hasta que el Usuario indique qué posición es la deseada donde dejar la ficha roja, para lo cual será necesario un diálogo mediante la Flexpendant, la cual nos mostrará las posiciones a elegir. En este momento el robot se desplazará a dicha posición, en caso de no estar ocupada por otra ficha, y dejará la pieza (a no ser que presionemos la entrada digital di1, de la que hablaremos más adelante). En los casos en los que la posición indicada por el Usuario esté ocupada, aparecerá un mensaje por la Flexpendant indicándolo, de tal manera que el Usuario deberá elegir otra posición a dejar la ficha. Para una mayor facilidad para Usuarios que no conocen de una manera elevada nuestro programa he decidido realizar la elección de la posición mediante un menú en el que primero se deberá indicar la fila a dejar la ficha (Arriba, Medio o Abajo) y después se indicará la columna (Izquierda, Medio o Derecha), ya que indicando mediante número el Usuario es más propenso a errores.

    b. Cuando sea Robot el encargado de poner una ficha el procedimiento se basará en la lógica programada, la cual consiste en la generación de números aleatorios. No he dedicado gran cantidad de tiempo para el desarrollo de la lógica puesto que el proyecto se centra en la programación de RAPID y las características típicas de la programación de robots, no de la programación lógica de un juego como puede ser el tres en raya. Una vez generado el número aleatorio el robot moverá la ficha, previamente cogida, a la posición generada.

7. El juego continuará hasta que se produzcan diferentes eventos, como los siguientes:

    a. Error de algún tipo en el programa (piezas no encontradas, error en la lectura/escritura de archivos, superación del máximo de interrupciones permitidas, etc.).

    b. No sea posible alojar más fichas en el tablero (9 fichas como máximo). En cuyo caso la partida termina en empate.

c. No haya más fichas a colocar, puesto que ya hemos emplazado todas las disponibles. En cuyo caso la partida termina en empate.

d. Usuario o Robot ganan la partida.

8. Después de que cualquiera de estos eventos se haya producido, el robot recogerá las piezas del tablero y las desplazará a la posición donde las ha cogido inicialmente (en los casos de que haya piezas sobre el tablero). De la misma manera, el robot recogerá todas las piezas localizadas en el tablero cuando termine cada una de las partidas.

9. Como hemos mencionado anteriormente disponemos de una interrupción. Poniéndonos en situación: Una vez que el robot tiene una ficha roja cogida y situada en la posición de reposo el Usuario decidirá la posición donde dejar la pieza. Dicho Usuario puede equivocarse a la hora de elegir donde dejar la pieza y seleccionar una posición del tablero errónea. En este caso dispondrá de un tiempo comprendido entre la elección de la posición del tablero y el momento en el que el robot llega a dicha posición para decidir si quiere cambiar de opinión. De esta manera la interrupción consiste en lo siguiente: Si en ese periodo de tiempo, en el que el Usuario puede cambiar de opinión, presiona la entrada digital “di1”, el robot retornará a la posición de reposo y solicitará una nueva posición del tablero donde queremos dejar la ficha. Este proceso se repetirá tantas veces como el Usuario desee.

10. En la finalización del programa (tras cumplir las dos partidas), se mostrará por la pantalla de la Flexpendant un informe de los datos del programa, donde se incluye: El día y la hora de la partida, la duración de cada una de las dos partidas (en minutos y segundos) y el resultado, es decir, quién ha ganado la partida o si ha resultado en empate. Para tener registro de todas las partidas, incluiremos dichos datos en un fichero de texto llamado “Resultados.txt” localizado en la carpeta “HOME”, de manera que podemos visualizar todas las partidas realizadas con su fecha, tiempo y resultado.

Catálogo de Usuario

En este apartado definiremos las pautas a seguir por el Usuario para una correcta utilización y comprensión de la ejecución del programa en el robot IRB 120. Por lo tanto es necesaria la lectura de esta descripción previamente de la ejecución del programa. Los pasos a seguir durante la ejecución del programa serán los siguientes:

1. En primer lugar deberemos subir el programa al robot, mediante una memoria USB podemos cargarlo a través de la Flexpendant.

2. Deberemos colocar el archivo “verdes.txt” en la carpeta “HOME” del robot, para ello podemos apoyarnos en el explorador de archivos de la Flexpendant y cargarlo mediante una memoria USB. Este archivo deberá contener letras a elegir entre “A, B, C, D y E” en el orden que deseemos (no es necesario colocarlas en orden).

3. Deberemos colocar las fichas verdes en las posiciones estipuladas por el fichero de texto “verdes.txt”. En caso contrario el robot no podrá coger las fichas correctamente y nos dará mensaje de error al ir a una posición vacía. Las piezas rojas deberán estar localizadas en la línea de búsqueda y será necesario que haya un mínimo de fichas igual al número de fichas verdes definidas. Ambas fichas verdes y rojas deberán estar localizadas a una distancia máxima del robot, para poder localizar las últimas fichas dentro del espacio de trabajo del robot.

4. Una vez realizado todo esto estaremos en posibilidad de ejecutar el programa.

5. Cuando ejecutamos el programa nos aparece por pantalla “Inicio del Programa” y empezará la ejecución normalizada del mismo.

6. Una vez iniciado el programa y cada vez que iniciemos una partida nos aparecerá por pantalla “Partida nueva” y nos saltará un diálogo preguntándonos si deseamos empezar o si cedemos el turno al Robot.

7. Independientemente de quién empiece, comenzará el juego. Durante el turno del Robot, todo movimiento queda ajeno al Usuario y será el robot el que realice todas las funciones, cogiendo fichas verdes y dejándolas en la posición del tablero que el propio robot haya decidido. Durante nuestro turno, el robot cogerá la ficha verde y la llevará a posición de reposo esperando a nuestra decisión. Deberemos elegir una posición del tablero (la que deseemos). En el caso de que dicha posición se encuentre ocupada por otra ficha nos dará error y nos volverá a pedir una nueva posición. Si la posición del tablero está vacía se procederá a dejar la ficha, donde tenemos dos opciones:

    a. Hemos decidido bien donde dejar la ficha: En cuyo caso no deberemos realizar ninguna acción y dejaremos que el robot continúe con su movimiento.

    b. Hemos introducido mal la posición a dejar la ficha: En cuyo caso tendremos un tiempo para pulsar la entrada digital “di1”. Si accionamos la entrada digital “di1” el robot retornará a la posición de reposo y volverá a pedir los datos de la posición a dejar la ficha. El tiempo que tenemos para decidir queda definido por dos instantes: El momento en el que introducimos erróneamente la posición y el momento en el que ha llegado a dicha posición, por lo que deberemos cerciorarnos rápidamente de la posición indicada. Si no pulsamos la entrada “di1” antes del tiempo máximo la ficha no podrá moverse, “ficha en la mesa presa”.

8. Transcurrirá con normalidad la partida hasta que no haya más piezas o posiciones disponibles, o que uno de los dos participantes gane. En este último caso se indicará por pantalla de la Flexpendant el ganador y se indicará “Fin de partida”. En todos estos casos el robot recogerá todas las fichas del tablero y las colocará en sus posiciones iniciales.

9. Podremos jugar un máximo de dos partidas.

10. Cuando finalicen las dos partidas se podrá observar por pantalla la fecha, el tiempo de cada partida y los resultados de ambas partidas. Podremos revisar los detalles de todas las partidas realizadas en el archivo llamado “Resultados.txt” localizado en la carpeta “HOME”, la cual la podremos encontrar mediante el explorador de archivos de la Flexpendant.

¡¡ Cuidado !! ¡¡ No se puede modificar la posición de las fichas en el tablero durante la ejecución del programa, puesto que el robot no conoce dichos cambios y produciríamos un error fatal !!

Desarrollo del proyecto

En este apartado vamos a explicar en detalle cada uno de los aspectos de la programación. Ambos programas (aplicable al robot real y de simulación) se encontrarán adjuntos en el archivo subido en el Campus Virtual por lo que no incluiré nada imágenes del código en RAPID.

Para la realización de este programa he desarrollado una serie de módulos que abarcan las funciones, procesos e interrupciones necesarias para la correcta ejecución del mismo. Estos módulos son los siguientes (con sus procesos, funciones e interrupciones asociadas):

- M_Main: Se incluyen todas las variables globales a utilizar durante la ejecución del programa. Contiene el proceso principal y la interrupción:
  - main
  - int1

- Asignacion_Lectura: Contiene lo relacionado con la lectura del fichero “verdes.txt” y la asignación de posiciones del tablero:
  - Asigna_Tablero
  - LeeFichero

- Busca_Pieza: Contiene la función asociada a la búsqueda de piezas:
  - Rastreo

- CalibData: Incluye todos los puntos, objetos de trabajo, herramientas de trabajo y posiciones del robot de interés. Además incluye los procesos de abrir y cerrar pinza y la generación y asignación de posiciones del tablero mediante la generación de números aleatorios:
  - AbrirPinza
  - CerrarPinza
  - Rand
  - Random

- Comprobacion: Contiene los procesos asociados a la comprobación de ganador del juego y funciones de mostrar datos del juego:
  - Comprueba
  - Ganado
  - Ganador_Tiempo

- Juego: Contiene los procesos asociados al juego tanto de Usuario como de Robot:
  - Juega_Rojo
  - Juega_Verde

- M_Menu: Contiene la función destinada a la decisión sobre las posiciones del tablero:
  - Menu

- Mover_Piezas: Contiene los procesos asociados con la colocación y retirada de fichas del tablero:
  - PonerDado
  - Recoge_piezas

Explicaré una a una la función de cada uno de los procesos, funciones e interrupciones para la mejor comprensión del programa.

PROC main()

Es el proceso principal de la ejecución del programa. En primer lugar conectará la variable de interrupción a la rutina de interrupción. Hará una llamada al proceso “LeeFichero” que nos indicará las posiciones de las fichas verdes.

Será el encargado de llevar el control de las partidas, de manera que en cada partida inicializará las variables necesarias y hará una llamada al proceso “Asigna_Tablero”. Al inicio de cada partida colocará al robot en la posición de equilibrio y solicitará al Usuario su deseo de empezar la partida en su turno. Inicializará en ese momento el control del tiempo de juego y hará las llamadas necesarias a los procesos “Juega_Rojo” y “Juega_Verde” para el trascurso de la partida. Será también el encargado de detener el tiempo de juego y hacer una llamada al proceso “Recoge_piezas”.

Cuando terminen las dos partidas llamará al proceso “Ganador_Tiempo” y eliminará la interrupción.

En este proceso se incluyen los siguientes tratamientos de errores:

- Saturación de la cola de interrupciones: Dejará la pieza que porta en su posición y recogerá el resto de piezas, con su posterior finalización del programa.
- Pérdida de contacto con la unidad: Finalizará el programa.

- Tiempo de programa sobrepasado: Recogerá todas las piezas del tablero para posteriores partidas y finalizará el programa.

TRAP int1

Será la rutina de la interrupción que ya hemos definido. Su propósito consiste en vaciar la posición del tablero donde íbamos a colocar la pieza y modificar la variable "repetir" que nos ayudará a tratar la interrupción a lo largo del programa. Moverá el robot a la posición de equilibrio cuando termine la instrucción que se estaba ejecutando.

PROC LeeFichero()

Será la función encargada de leer el fichero "verdes.txt" y rellenar un vector de 5 posiciones con "unos" en las posiciones indicadas por el fichero.

Abrirá el fichero, lo leerá, y asignará a una variable la posición de la letra a buscar (A, B, C, D y E). Posteriormente comprobará si se encontró la letra a buscar y rellenará el vector nombrado anteriormente. Tendrá los siguientes tratamientos de errores:

- En el caso de que el contenido del fichero no sea el requerido lanzará error de "Parámetros del fichero erróneos".
- Error al acceder al archivo.
- Error al abrir el archivo.
- Error al buscar el archivo.

En todos los errores anteriores se terminará la ejecución del programa.

PROC Asigna_Tablero()

Este proceso será el encargado de establecer las diferentes posiciones del tablero (robtarget), teniendo en cuenta la separación entre celdas, y será el encargado de vaciar la matriz de fichas posicionadas en el tablero (para que inicialmente se encuentre vacía dicha matriz.

FUNC robtarget Rastreo (robtarget inicio)

Esta función devolverá la posición de la pieza encontrada por rastreo. Como argumento a dicha función tenemos la posición de inicio de búsqueda.

Moveremos el robot a la posición de reposo y dependiendo de donde se localice la posición inicial de búsqueda distinguiremos entre piezas rojas y verdes. El robot se moverá (con las precauciones necesarias) al punto de inicio de búsqueda y buscará la primera pieza que encuentre para devolver su posición. Para evitar errores, si en el punto de inicio de la búsqueda detectamos una pieza, la cogemos y no realizamos la búsqueda.

Como tratamiento de errores, tenemos uno: Si no encontramos ninguna ficha en la búsqueda concluimos que las piezas están mal colocadas o que el sensor de la pinza está estropeado, por lo que indicamos al Usuario la mala colocación de las fichas. Posteriormente nos vamos a la posición de reposo con precaución y recogeremos las piezas del tablero.

FUNC num rand() y FUNC num random()

La primera de ellas generará un número aleatorio de magnitud 1, y la segunda devolverá un valor entre 1 y 3 dependiendo del valor de dicho número aleatorio.

PROC AbrirPinza() y PROC CerrarPinza()

Abrirán y cerrarán la pinza respectivamente, manteniendo una espera de 2 segundos para garantizar que se abre y cierra completamente.

PROC Comprueba()

El objetivo de este proceso será el comprobar, cada vez que se coloca una ficha sobre el tablero, si la partida se ha terminado por el hecho de que uno de los participantes haya ganado la partida. Para ello deberá comprobar en filas, columnas y en diagonales.

PROC Ganado()

Este proceso únicamente mostrará por pantalla, cada vez que se coloca una ficha sobre el tablero, si uno de los dos participantes ha ganado la partida, en caso contrario no intervendrá en la ejecución del programa.

PROC Ganador_Tiempo()

Este proceso se ejecutará únicamente al finalizar las dos partidas del programa. Su objetivo consiste en mostrar la información relacionada con la ejecución de cada partida. Para ello es necesario calcular la duración de cada programa en minutos y segundos y acceder a determinados datos de control de la ejecución.

En primer lugar se almacenarán los datos en un fichero de texto y posteriormente (con una estructura distinta debido a su menor flexibilidad) mostraremos toda la información por la pantalla de la Flexpendant. Esta información consiste en lo siguiente: Fecha y hora de la partida, tiempo de juego en minutos y segundos cada una de las dos partidas y resultado final de cada partida.

Tenemos una serie de rutinas de error relacionadas a este proceso, las cuales son similares a las del proceso “Lee_Fichero”, las cuales son:

- Error al acceder al archivo de escritura Resultados.txt
- Error al intentar abrir el archivo

- Error en la búsqueda del fichero.

PROC Juega_Rojo()

Este proceso consiste en la parte de programa que se debe ejecutar cada vez que el turno es el de Usuario, es decir, se muevan las fichas rojas. Para poder cumplir todas las especificaciones explicadas al principio de este informe, he distinguido entre dos casos:

1. Si no se ha producido ninguna interrupción: En este caso el programa funcionará en su estado nominal y la variable "repetir" valdrá "FALSE". Como no podría ser de otro modo, también he distinguido entre dos casos: en los casos en los que se empiece y en los que no. La diferencia entre ambos casos consiste en el trato sobre la variable auxiliar "saltar", la cual nos ayudará a ignorar la ejecución del turno del Usuario en los casos de que una pieza verde no se encuentre descrita en el fichero.

2. Si se ha producido una interrupción: En este caso la variable "repetir" valdrá "TRUE" (modificada en la rutina de interrupción) por lo que debemos inicializarla a "FALSE" para poder detectar interrupciones posteriores. En este caso únicamente se ejecutará la parte de programa correspondiente, que consistirá en el fragmento de instrucciones en las cuales el robot ya parte de la posición de reposo con una ficha agarrada por la pinza.

PROC Juega_Verde()

Este proceso se debe ejecutar cuando el turno de movimiento corre por parte del participante Robot. Si estamos ante una situación de que estamos iniciando la partida y no hemos colocado ninguna pieza verde, buscaremos mediante rastreo la primera pieza verde disponible (almacenando en una variable el valor de su posición y tomando como referencia dicho punto).

En los casos posteriores, si la ficha se encuentra descrita en el fichero definiremos la posición de las siguientes piezas por desfase.

Posteriormente colocaremos las piezas disponibles en el tablero, aumentando el número de piezas en el mismo.

FUNC robtarget Menu (num tipo)

Esta función es la encargada de definir las posiciones del tablero a ocupar en cada caso. Se pasará como argumento el tipo de la pieza (roja o verde) y solicitará datos de donde dejar la ficha o generará una posición del tablero en cada caso.

Una vez que se ha determinado cual será la posición del tablero a dejar la ficha, rellenaremos la matriz "tablero" en la que se indican las posiciones ocupadas. Dicha función devolverá una posición en donde deberemos dejar la ficha.

PROC PonerDado (robtarget inicio, bool búsqueda)

El objetivo de este proceso será la colocación de las fichas sobre el tablero. Los argumentos de dicha función serán una posición llamada “inicio” que describe donde se encuentra situada la pieza a coger y una variable “bool búsqueda” que indica si se ha obtenido el valor de la posición de la ficha mediante búsqueda o mediante desfase (TRUE y FALSE respectivamente).

En los casos de que no se haya producido mediante búsqueda, deberemos desplazar el robot a una posición superior (en el eje z) para coger la ficha cómodamente. Posteriormente, y sin distinguir entre casos, moveremos el robot a la posición de la pieza, cerraremos la pinza y nos desplazaremos a la posición de reposo con las precauciones necesarias. En este proceso, he incluido un tratamiento de error que consiste en lo siguiente: Si nos desplazamos a la posición de la pieza y el sensor detector de piezas no detecta ninguna, entonces se producirá un error debido a que no hay ninguna pieza en esa posición.

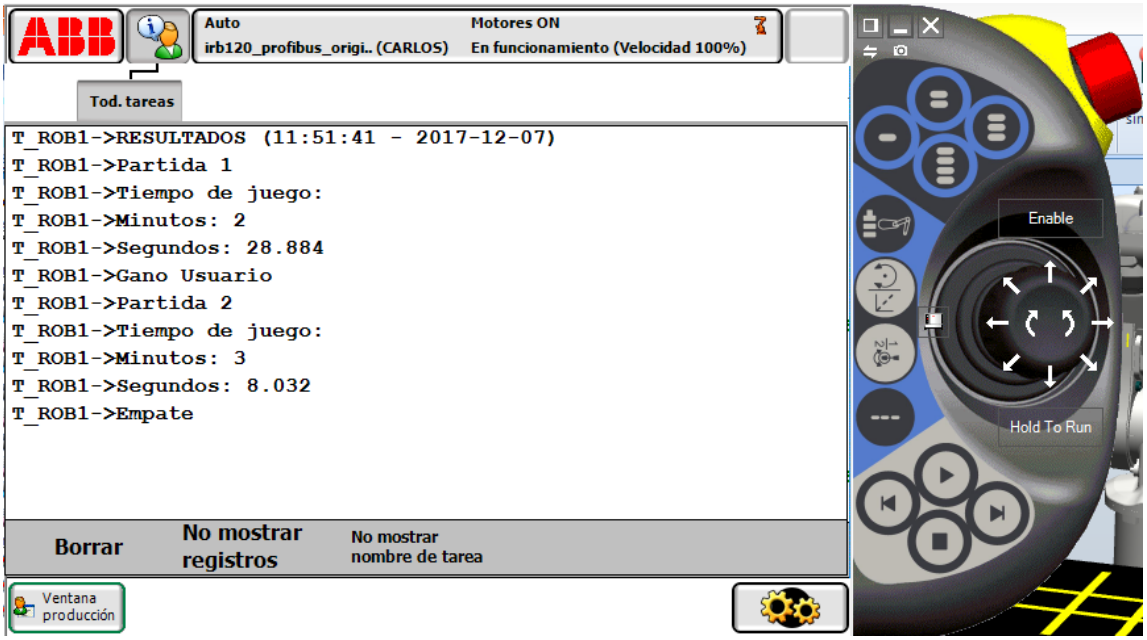
Posteriormente, haremos una llamada a la función “Menu” y tras conocer en qué posición deberá dejar la ficha el robot activaremos la interrupción si la pieza a mover es roja. Después de ello distinguiremos entre dos situaciones: Si se produce una interrupción nos saldremos de esta función; Si no se produce localizaremos al robot en una posición determinada del tablero y dejaremos la ficha con su posterior comprobación de ganador.

PROC Recoge_piezas()

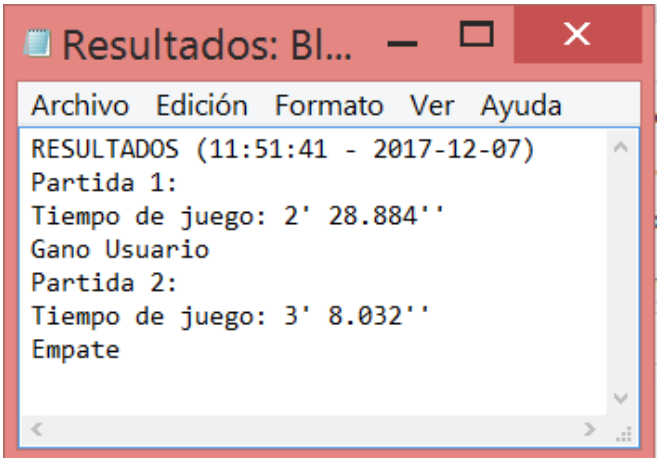
Este proceso será el encargado de tomar las fichas del tablero y moverlas a sus posiciones iniciales. Para ello comprobamos posición por posición si hay una pieza en la misma, en cuyo caso nos desplazaremos a dicha posición y cogeremos la pieza. Posteriormente, dependiendo de qué tipo de ficha sea la movemos a la posición inicial de cada una de ellas, almacenada en las variables “pieza_v” y “pieza_r”. En los casos de que la pieza no esté descrita en el fichero a leer, saltaremos dicha posición.

Ejemplo de resultados

Para poder observar los resultados obtenidos en la ejecución del programa se muestra el siguiente ejemplo, en el que al terminar las dos partidas disponibles se ha mostrado por la pantalla de la Flexpendant lo siguiente:



Donde se encuentra lo que he descrito anteriormente. Para poder comprobar el resultado de todas las partidas generadas se almacenarán dichos datos en un fichero mencionado previamente. Aunque se almacenarán de manera diferente debido a la mayor flexibilidad que nos permite realizar las escrituras en fichero, se muestran los mismos datos:



Conclusiones y resultados

Un detalle de la práctica que no he mencionado en ocasiones anteriores es que para el inicio de la búsqueda de una nueva pieza roja, toma como punto de partida la última pieza roja encontrada, de esta manera se reducen los tiempos de búsqueda y de esta manera el tiempo de espera y ejecución.

Al finalizar el proyecto y con su correspondiente comprobación mediante simulación se concluye que el programa desarrollado cumple con todas las especificaciones propuestas en el enunciado del proyecto, cumpliendo tanta las especificaciones previas como las consideraciones posteriores.

Para comprobar dicha simulación, se adjunta una carpeta llamada “Proyecto_Simulacion” en la que se incluye el programa desarrollado para dicha simulación (el cual sufre alguna pequeña modificación sobre el programa del proyecto a subir al robot IRB 120. Dicho programa se encuentra en la carpeta llamada “Proyecto_Tres_En_Raya”.

Para la correcta ejecución de ambos programas será necesario que exista un archivo de texto llamado “verdes.txt” en la carpeta HOME que cumpla las especificaciones requeridas. He incluido un archivo de texto con ese nombre como modo de ejemplo.

He encontrado el desarrollo de la práctica muy entretenido debido a que programar el funcionamiento de un robot es algo divertido y ameno. Si existe alguna pregunta o consideración sobre el programa desarrollado, estoy dispuesto a resolver cualquier duda y de esta manera facilitar la comprensión del código.