Índice

Introducción

Este trabajo tiene como objetivo el reconocimiento en tiempo real de piezas de ajedrez, así como su posición en el tablero, que cambia durante la partida.

Podemos dividir el trabajo en dos bloques, un primer bloque consiste en el desarrollo de una aplicación en C++ y un dispositivo hardware para el escaneo de cada una de las distintas piezas y utilizar esas fotografías para el entrenamiento de una red neuronal. El segundo bloque consiste en el desarrollo de una aplicación en Java para el sistema operativo Android que sea capaz de reconocer un tablero de ajedrez en una imagen tomada de la cámara del dipositivo para después separar la imagen por casillas y detectar en que casilla se encuentra cada pieza con el uso de la red neuronal mencionada anteriormente.

Dispositivo de escaneo

1. *Introducción:*

El dispositivo de escaneo de piezas de ajedrez está diseñado utilizando hardware y software libre.

El código fuente del microcontrolador, del ordenador y los modelos STL para impresión 3d se encuentran disponibles para su descarga en el repositorio: ***http://www.github.com/angmolin/ChessPieceScanner.***

Los planos de las bases inferior y superior del dispositivo están disponibles en los anexos 1 y 2 respectivamente.

Debido a que usamos el driver V4L2 para manejar la webcam, es necesario que el ordenador ejecute un sistema operativo basado en Linux y que sea compatible con esta librería, además la webcam utilizada debe ser soportada por este driver y permitir la captura de imágenes en formato MJPEG.

2. *Hardware:*

- Arduino Uno
- 2 Controladoras de motores paso a paso A4988
- Motor paso a paso NEMA-17
- Motor paso a paso NEMA-14
- Servomotor miniatura
- Dispositivos ópticos de final de carrera

3. *Software:*

- Librería V4L2
- Librerías *nix

4. *Funcionamiento:*

El microcontrolador es el encargado de gestionar los drivers de los motores, el servomotor y los dispositivos ópticos de final de carrera, además, con el fin de evitar inconsistencia, guarda las coordenadas de cada eje. La comunicación con el ordenador se realiza por medio de puerto serie a una velocidad de 115.200 baudios. Véase el anexo 3 para consultar la lista de comandos disponibles.

El ordenador controla el microcontrolador y la webcam de forma simultánea, posicionando primero las coordenadas de los ejes y después capturando la imagen de la webcam en formato JPG, después recorta la imagen y finalmente la guarda con el nombre $1y$2z.jpg donde $1 es la coordenada del eje Y y $2 es la coordenada del eje Z. El comportamiento de la webcam (brillo, luminosidad, saturación...) puede ser configurado desde el archivo de configuración ‘./conf/webcam.conf’. El recorte y la posición de cada imagen puede configurarse en ‘./conf/images.conf’.

5. *Compilación:*

La compilación del código del microcontrolador puede hacerse desde el IDE de Arduino abriendo el fichero '.ino'. También podemos subir el código a la placa desde aquí.

La compilación del código del ordenador está automatizado mediante un Makefile y puede realizarse yendo a la capeta raíz del proyecto y ejecutando la orden 'make' en un terminal.

6. *Ejemplo de ejecución:*

Al ejecutar la aplicación se abre la webcam, el puerto serie y se calibra el dispositivo posicionando los ejes J, Y y Z en la posición 0. Después aparece el menú principal. *fig. 1.*

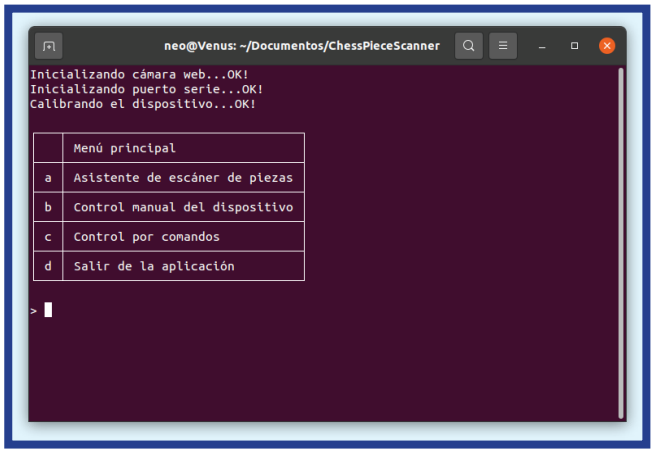

fig. 1

Desde este menú podemos iniciar el asistente de escaneo de piezas, controlar el dispositivo manualmente o enviar comandos al microcontrolador. Si seleccionamos la primera opción, introduciendo la letra 'a' y pulsando 'Enter' podemos ver que la interfaz cambia.

Ahora podemos seleccionar cómo deseamos realizar las fotografías, un test para comprobar que el recorte de las imágenes está configurado correctamente, realizar fotografías de 180 grados para la pieza ó realizar fotografías de 360 grados *fig. 2. Una vez seleccionado el modo, nos pregunta por el nombre de la pieza (Con este nombre se creará una carpeta en el directorio './photos/' donde se guardarán las imágenes) fig. 3.*

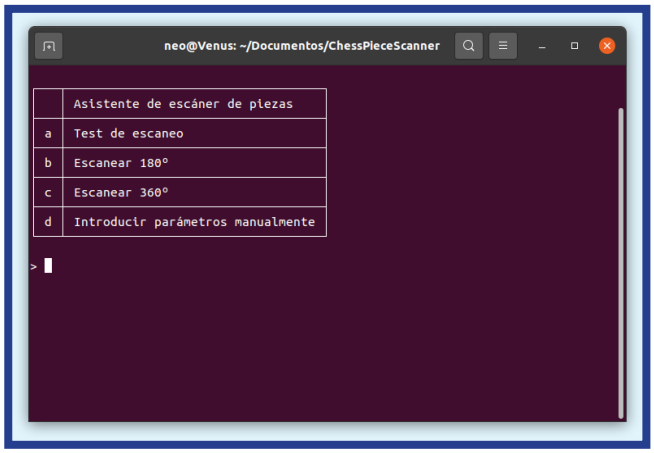

fig. 2

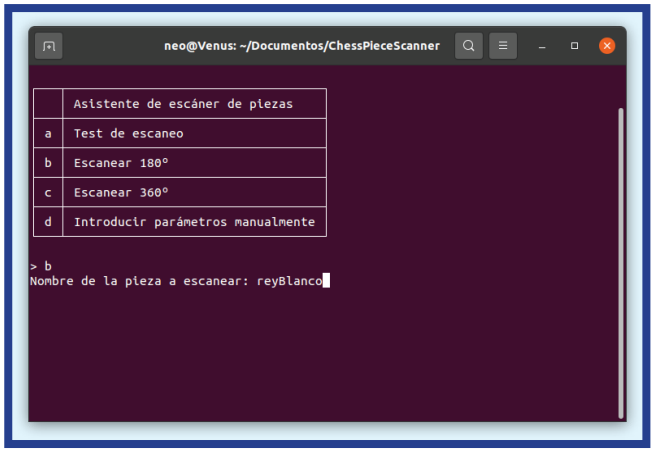

fig. 3

La aplicación comenzará a mandar posiciones al microcontrolador que posicionará cada eje en la posición solicitada y realizará la fotografía. Finalmente la aplicación volverá al menú principal y esperará a que el usuario seleccione una acción. *fig. 1.*

Si desde el menú principal seleccionamos la opción 'b' se entrará en el modo de control manual del dispositivo. Desde aquí podemos mover cada uno de los ejes a la posición que deseemos y realizar una fotografía de prueba (esta fotografía será guardada en la ruta './photos/preview.jpg'. *fig. 4.*

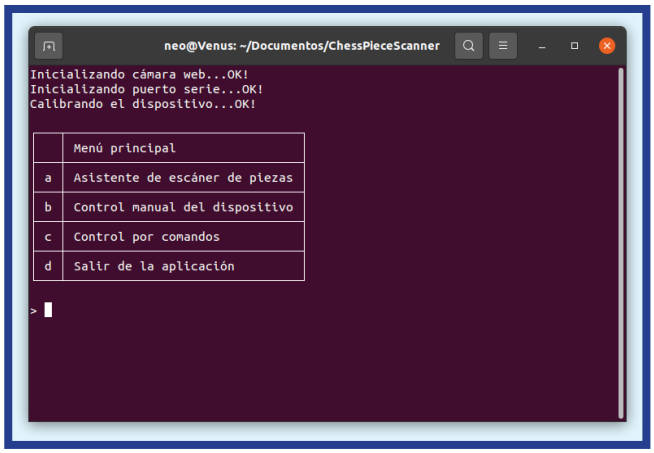

fig. 4

También podemos realizar un control totalmente manual del dispositivo por medio de comandos, para ello seleccionaremos la opción 'c'. Al seleccionar esta opción se abrirá una terminal que permitirá la comunicación mediante comandos. Además se muestra un pequeño resumen de los comandos disponibles. *fig. 5.*

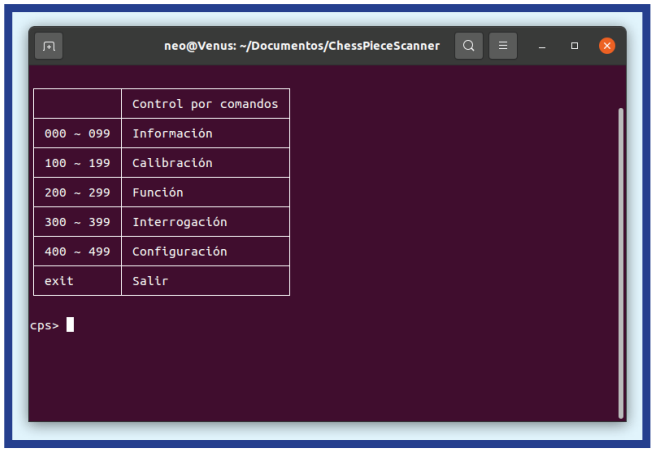

fig. 5

Para finalizar la aplicación debemos volver al menú principal y pulsar la opción 'd', de esta manera el dispositivo volverá a las coordenadas iniciales y liberará los motores para su desconexión segura y se devolverá al usuario a la línea de comandos. *fig. 6.*

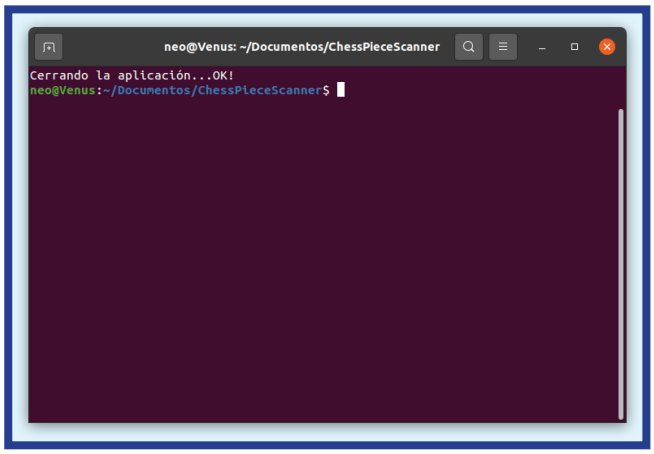

fig. 6

7. Imagenes de muestra

Las imágenes guardadas en la nueva carpeta creada con el nombre dado en el directorio './photos/' están listas para realizar la inferencia de la red neuronal.

Para el entrenamiento de nuetra red hemos realizado fotografías de las piezas con fondo blanco y con fondo negro. *fig. 7, 8, 9, 10.*

fig. 7

fig. 8

fig. 9

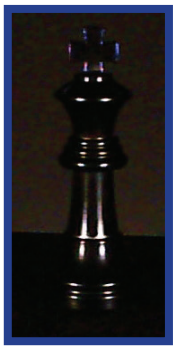

fig. 10