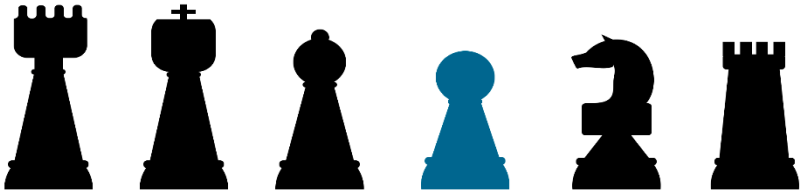

AJEDREZ MÁGICO

PROYECTO FIN DE GRADO

Ingeniería de computadores

Rubén Cerrillo Muñoz

Índice

Objetivo

Comenzamos esta andadura con una idea clara en mente, construir un ajedrez. Se quiere dar una vuelta de tuerca al ajedrez clásico y construir uno que, utilizando la tecnología actual, permita jugar mediante órdenes de voz. A su vez tiene que ser autónomo, es decir, que utilizando inteligencia artificial sea capaz de tomar decisiones y hacer frente a un jugador. Como se desprende de la idea del control por voz, no será necesario que el jugador mueva las fichas con sus manos, será el propio ajedrez el que muevas las fichas respondiendo a las órdenes del jugador.

Uno de los objetivos secundarios es utilizar para su construcción piezas accesibles para la mayoría de la gente. Todas las piezas a utilizar se podrán encontrar en tiendas físicas y por supuesto en tiendas online donde se comprarán la mayoría de los componentes. Se utilizarán piezas genéricas y bien conocidas en el panorama actual de la tecnología, módulos prefabricados y probados por la comunidad, software y hardware libre, así como piezas impresas en 3D que nos servirán de gran ayuda a la hora de ajustar los componentes a nuestras necesidades.

La idea es que todo el software y el diseño hardware sea público y accesible para todo el mundo, así, quien decida construirse su propio ajedrez o mejorarlo o simplemente aprender construyendo, tendrá disponible la documentación necesaria para hacerlo sin que ello suponga una dificultad muy grande.

Orientar el proyecto hacia terrenos comerciales nunca ha sido un objetivo, con lo que éste estará orientado a la educación. Uno de los grandes objetivos es aprender. Se van a utilizar herramientas con las que nunca se ha tenido contacto, lenguajes de programación nuevos (Python) y componentes hardware tales como la Raspberry Pi 3, que será usada como cerebro y controlador del ajedrez, que habrá que aprender cómo usar el sistema GPIO de la misma, instalación de entornos para desarrollar, etc. Con todo esto, el componente de investigación previo y durante el desarrollo del proyecto será importante. Para la finalización del proyecto se espera haber aprendido en multitud de áreas, en diseño de componentes, planificación, documentación, lenguajes de programación nuevos, sistemas de control, control por voz y algunas pinceladas de inteligencia artificial.

Resumen

Durante siglos de existencia del ser humano, éste ha desarrollado todo tipo de juegos para su entretenimiento. La mayoría de ellos se basan en un tablero con fichas donde desarrollan una estrategia para derrotar al contrario. Antiguamente el azar no tenía cabida en estos juegos y el desarrollo de este únicamente dependía a la mente del jugador y su capacidad para llevar a cabo estrategias ganadoras. Estos juegos han ido evolucionando y han llegado hasta nuestros días donde la esencia de aquellos divertimentos se conserva. El ajedrez es uno de ellos, un tablero, piezas y dos jugadores dispuestos a pensar para ganar. Es muy popular actualmente, mucha gente sabe jugar e incluso, muchos de ellos son profesionales.

Este proyecto le da una pequeña evolución a este juego de toda la vida, con la inclusión de control por voz y movimiento automático de las piezas que permitirá que el juego pueda llegar a muchas más personas que por motivos de discapacidad, por ejemplo, no podían utilizarlo.

Se ha optado por una solución que a su vez incorpore inteligencia artificial para poder jugar contra la máquina. Se va a desarrollar sobre una Raspberry Pi 3 donde se albergará el sistema de control, sistemas para el reconocimiento de voz y el motor de inteligencia. Todo esto se desarrollará en Python utilizando diversas librerías existentes. El reconocimiento de voz correrá a cargo de Google Cloud Platform con su API de Speech Recognition y la inteligencia artificial estará soportada por Stockfish Chess Engine. El uso de librerías para manejar dichas APIs hará que la complejidad del proyecto se reduzca para permitir futuras mejoras y mejor claridad de funcionamiento para todo aquel que quiera aprender, mejorar y desarrollar su propio juego.

Abstract

For centuries, humans have developed all kind of games for entertainment. Most of them are played on a board with some pieces where they play a strategy to beat the opponent. In the past the only thing that matters in the game was the ability of the player to display great winner strategies. This games have been evolving and they have arrived to the present where the essence of the game remains. Chess is one of this games. It has a board, pieces and two players ready to think how to win. It is very popular nowadays, lot of people know how to play and most of them are even professional players.

This project gives a tiny evolution to this game, with the inclusion of voice control and piece automatic movement, it allows to access the game to many people that could not play before, like people with disabilities.

A solution has been developed that includes artificial intelligence to be able to play against the machine. All this will be developed with a Raspberry Pi 3 which will include the control system, speech recognition and the intelligence engine. All will be developed in Python using existing libraries. Speech recognition will be driven by Google Cloud Platform with the Speech Recognition API and the artificial intelligence will be done by Stockfish Chess Engine. By using libraries to manage this APIs the project complexity will be reduced and it allows that anyone that want to learn, improve or develop his own project can do it easily.

Capítulo 1. El Ajedrez

El ajedrez es un juego de mesa para dos jugadores. Dentro de esta simple definición se esconde el que es uno de los mayores juegos de la historia. Lleva siglos con nosotros, es uno de los juegos clásicos de mesa más conocido, la mayoría de las personas lo conocen o han jugado alguna vez o, incluso, poseen algún ejemplar en sus casas. El conjunto de piezas y reglas que usamos actualmente no ha sido siempre así. A lo largo de los años ha ido cambiando y adaptándose hasta lo que hoy en día conocemos como Ajedrez. Los historiadores datan las primeras partidas del ajedrez en el siglo V en India (Ortega). Con las conquistas musulmanas, el juego se extendió por Europa y en el siglo XV sufrió muchos cambios, ya que antes se jugaba con dados y las fichas eran diferentes. El tiempo ha ido moldeándolo y evolucionándolo hasta asemejarlo a lo que actualmente conocemos. Hoy en día es uno de los juegos de mesa más populares, aún con la aparición de innumerables juegos modernos tanto de mesa como videojuegos, el ajedrez se corona como uno que gran multitud de personas conocen y en cierto modo saben jugar. Existen jugadores profesionales de ajedrez que se baten en torneos oficiales en varias modalidades, con o sin tiempo. Así mismo es jugado por miles de personas en el mundo simplemente para su entretenimiento. Este juego requiere de la mente del jugador para llevarse a cabo, no hay azar. Ambos empiezan con las mismas fichas y únicamente la estrategia y la pericia de los contrincantes a la hora de moverlas los llevará a la victoria o a la derrota.

1.1. Reglas

La partida se disputa en un tablero de 8x8 casillas pintadas de manera alterna con colores contrastados, comúnmente blanco y negro. Existen seis tipos de fichas: peón, torre, caballo, alfil, reina y rey. Cada jugador dispone de dieciséis fichas, ocho peones, dos torres, dos caballos, dos alfiles, una reina y un rey. Cada uno coloca sus fichas en dos filas en los bordes opuestos del tablero. La primera fila la forman los ocho peones y la fila de detrás la forman las ocho fichas restantes. En cada esquina se sitúan las torres, a continuación, se van completando desde cada esquina hacia el centro con dos caballos, dos alfiles y terminamos con el rey y la reina, situando a esta última en la casilla de su mismo color. Las fichas de cada jugador se distinguen por el color, unas negras y otras blancas. En la siguiente figura se puede ver la disposición del tablero listo para empezar.

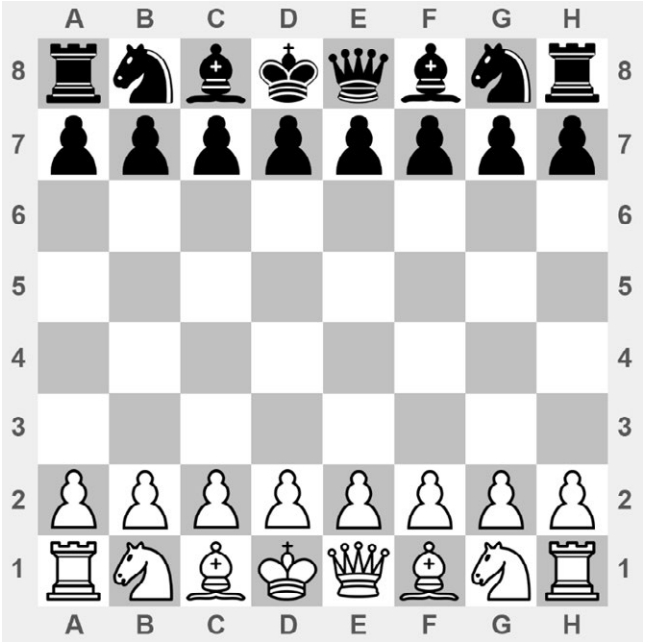

Figura 1: Tablero de ajedrez con sus piezas.

El tablero se sitúa entre los dos jugadores y el juego comienza con el jugador que posee las fichas blancas. Éste realiza el primer movimiento y se continúa alternativamente con el jugador de fichas negras haciendo cada uno un solo movimiento. El objetivo es capturar al rey (jaque mate). El jugador que haga jaque mate al rey del jugador contrario ha ganado y finaliza la partida. Si ninguno de los jugadores tiene la oportunidad de hacer jaque mate al rey del contrario entonces el juego finaliza quedando en tablas.

1.2. Movimientos

Cada pieza tiene un movimiento determinado, a continuación, se verán en detalle cada uno (Federación Internacional De Ajedrez). Hay que tener en cuenta que nunca puede haber dos piezas del mismo color en una misma casilla. Si una pieza llega a una casilla donde se encuentra otra del otro color, significará que esa pieza ha sido capturada y será eliminada del tablero como parte del movimiento. y que las piezas no pueden pasar por encima de otras en su trayectoria, a excepción, del caballo.

Peón

El peón se puede mover una casilla hacia adelante si ésta está desocupada o una casilla en diagonal, también hacia adelante, para capturar una pieza del oponente. En el primer movimiento del peón, puede avanzar dos casillas hacia adelante.

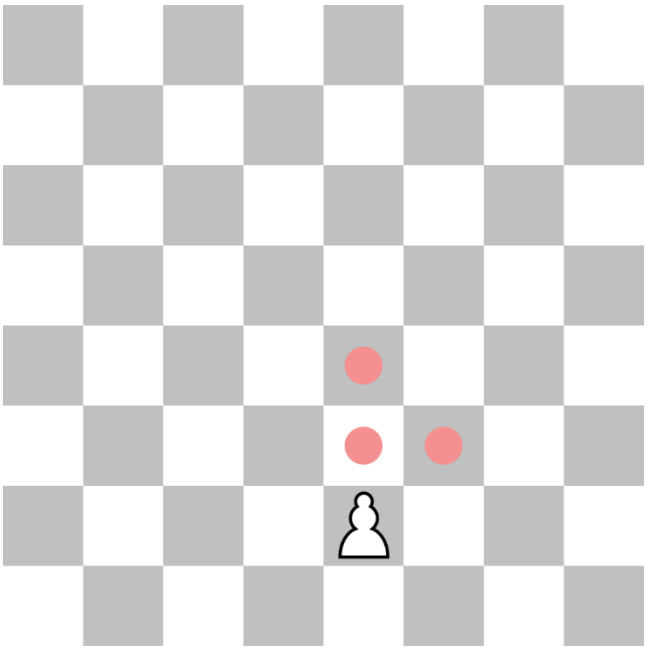

Figura 2: Movimientos del peón

Torre

La torre puede moverse un número indefinido de casillas, siempre que estén desocupadas en su misma fila o columna.

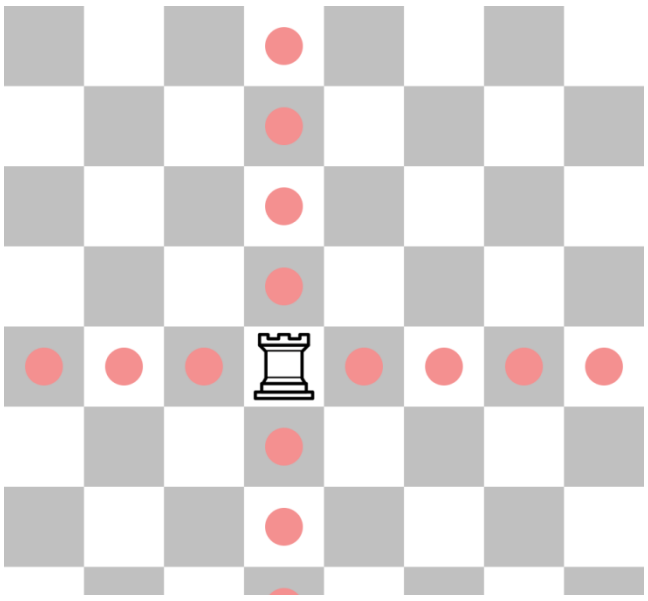

Figura 3: Movimientos de la torre

Caballo

El caballo se moverá a cualquiera de los cuadrados que tenga más cerca exceptuando los de su misma fila, columna o diagonal. Comúnmente se dice que el caballo se mueve formando una L. Esta pieza es la única que tiene la capacidad de saltar piezas que se encuentran en su trayectoria.

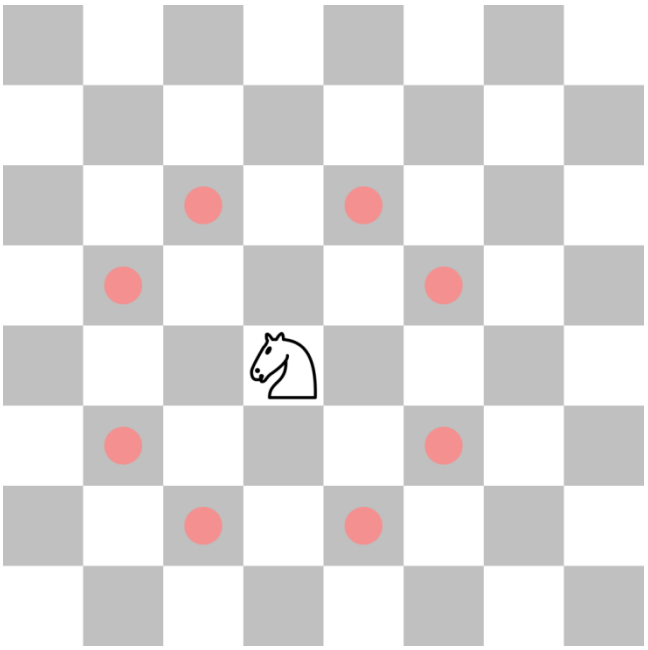

Figura 4: Movimientos del caballo

Alfil

El alfil se moverá un número indefinido de casillas no ocupadas a lo largo de sus diagonales.

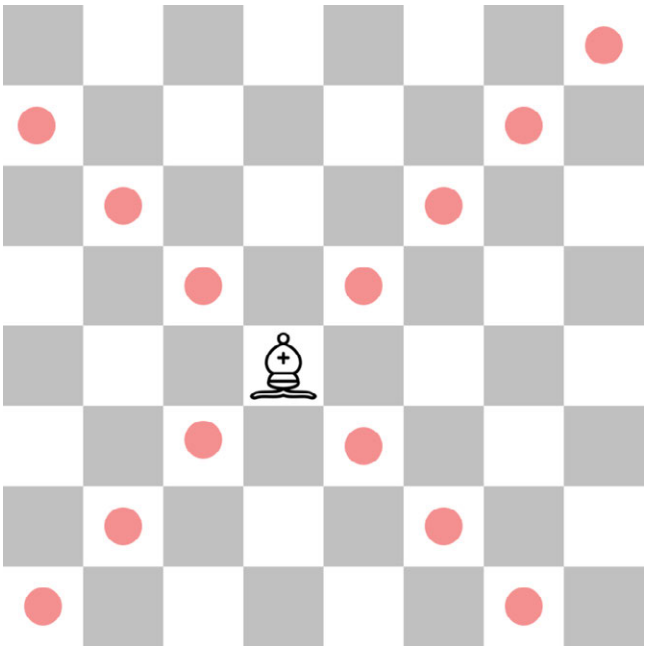

Figura 5: Movimientos del alfil

Reina

La reina se moverá un número indefinido de casillas no ocupadas a lo largo de su fila, columna y diagonales.

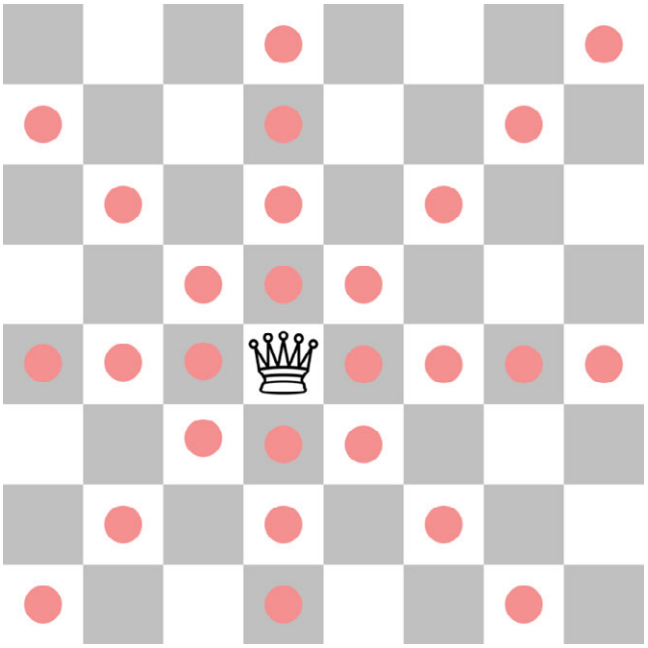

Figura 6: Movimientos de la reina

Rey

El rey se moverá una sola casilla en su misma fila, columna o diagonales.

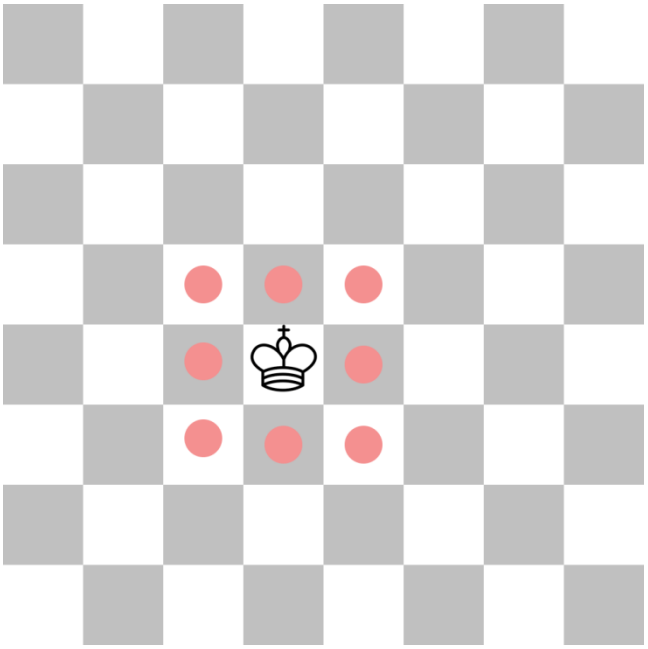

Figura 7: Movimiento del rey

1.3. Movimientos especiales

Existen varios movimientos especiales que se salen de lo que son las reglas comunes del ajedrez.

Enroque

El enroque es un movimiento que involucra al rey y a una de las torres. El rey se puede mover dos posiciones en dirección a la torre de la izquierda y la torre se mueve a la casilla que el rey ha cruzado (Figura 8). Así mismo, podría moverse el rey 3 casillas hacia la torre. Éste último se conoce como enroque largo y el primero como enroque corto.

Este movimiento solo puede hacerse bajo las siguientes condiciones:

- El rey nunca se ha movido.
- La torre a usar nunca se ha movido.
- El rey no está en jaque.
- Las casillas entre el rey y la torre están desocupadas.
- El rey no termina en jaque.

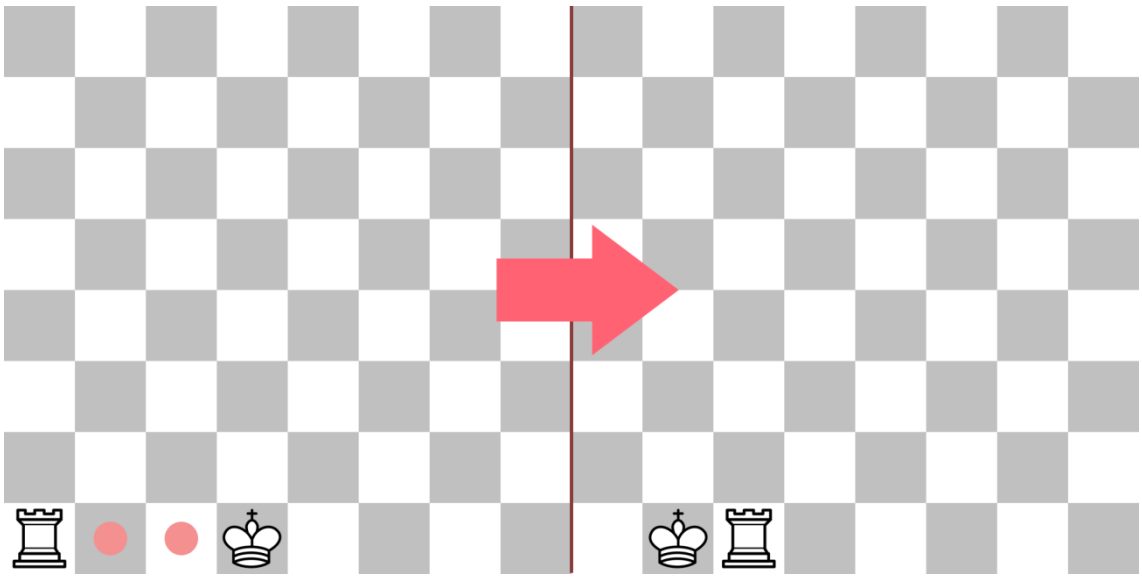

Figura 8: Enroque

Coronación

La coronación es la capacidad que tiene un peón de convertirse en cualquier pieza que el jugador desee, normalmente, en reina, ya que es la que tiene la mayor libertad de trayectorias. Este movimiento es posible cuando el peón alcanza el borde del equipo contrario.

Jaque

El rey está en jaque cuando está siendo atacado por una pieza adversaria, es decir, que el siguiente movimiento de dicha pieza puede ser capturarle. En este momento, es obligatorio para el jugador sacar al rey del jaque. Esto es posible de diferentes formas, ya sea moviendo el rey hasta una posición donde no esté amenazado, capturando la pieza atacante o disponiendo una pieza en la trayectoria de la atacante.

Jaque mate

Si el jugador no puede defender el rey de un jaque, entonces estaría en jaque mate y perdería la partida.

La partida podría quedar en empate, en tablas. Esto es así por varios motivos, cuando se repite tres veces la misma posición sobre el tablero, cuando ninguno de los jugadores tiene piezas suficientes para hacer jaque mate o si el jugador que tiene el turno no puede realizar ningún movimiento (Ajedrez - Online).

Existen multitud de reglas que definen todas y cada una de las posibilidades del juego, desde cómo mover las fichas con una sola mano hasta las reglas de competición. Todas ellas están recogidas en el *handbook* de la FIDE (Federación Internacional De Ajedrez).

Capítulo 2. Raspberry Pi

Las Raspberry Pi son una serie de placas de computación de bajo coste cuyo objetivo es difundir la tecnología y el mundo digital (Raspberry Pi : About). Está destinada principalmente a la educación, se puede considerar como un ordenador del tamaño de una tarjeta de crédito, como se puede ver más abajo. El modelo que se va a usar para este proyecto es la Raspberry Pi 3 Model B, es la última versión de una serie de placas que la Raspberry Pi Fundation, empresa detrás del producto, ha ido desarrollando a lo largo de los años.

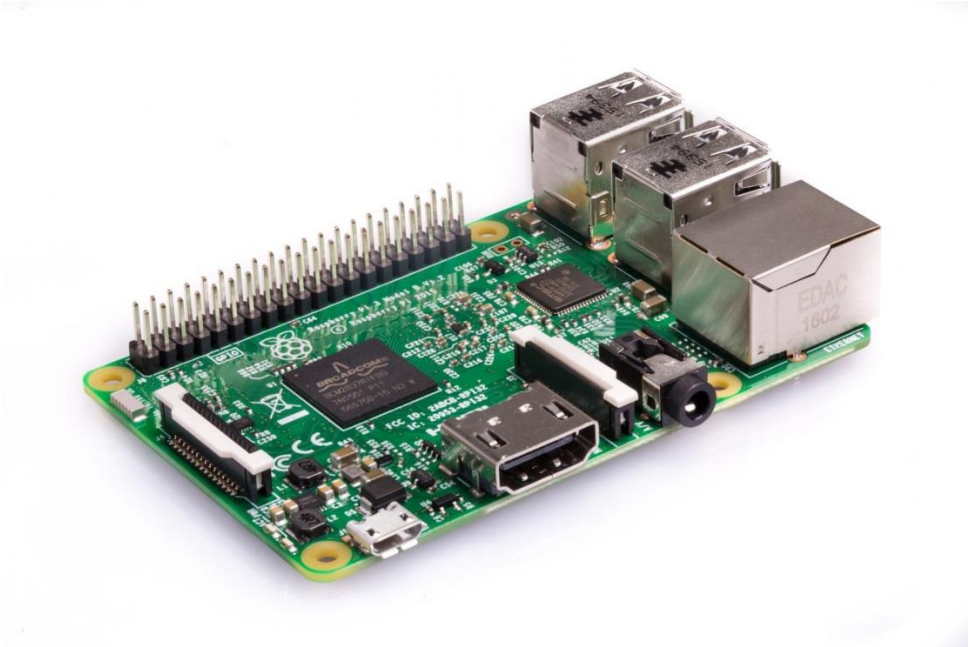

Figura 9: Raspberry Pi 3 Model B (raspberrypi.org)

Su uso es muy variado, aunque el objetivo principal era enseñar a programar a un coste muy bajo, sus características han permitido que sea usada en muchos otros terrenos como la robótica, la automatización o incluso la investigación.

Estas son las especificaciones de la placa que podemos ver en la web oficial:

- CPU Broadcom BCM2837 64bit
- 1GB RAM
- BCM43438 wireless LAN y Bluetooth Low Energy (BLE) integrado
- 40-pin extended GPIO
- 4 puertos USB 2.0
- Puerto Ethernet
- Puerto compuesto de video y audio
- HDMI
- Puerto CSI camera para la conexión de la Raspberry Pi camera
- Puerto DSI display para la conexión de Raspberry Pi touchscreen display
- Puerto Micro SD para la carga del SO
- Puerto de carga Micro USB hasta 2.5A

La versatilidad de esta placa reside básicamente en dos aspectos, su pequeño tamaño y los pines GPIO.

Para el funcionamiento de la placa debe contar con un sistema operativo, almacenado en la tarjeta SD que hará las veces de disco duro de un ordenador convencional. El sistema operativo más utilizado es Raspbian, una variante de Debian optimizada para el hardware de la Raspberry Pi (Raspbian). Se pueden usar multitud de sistemas operativos, desde más específicos como algunos para montar servidores hasta otros más generales como Windows 10.

La salida de video por HDMI junto con los cuatro puertos USB, dan la posibilidad de usar la pequeña Raspberry Pi como si de un ordenador común se tratase, pudiendo conectar el monitor y los periféricos sin que por ello ocupemos todos los puertos de la placa.

Alimentarla es realmente sencillo, al contar con un puerto micro USB. Aunque actualmente la tendencia es ir cambiando al USB-C, los puertos micro USB eran el estándar en la telefonía móvil, con lo cual, con casi cualquier cable y cargador de móvil, siempre y cuando la intensidad de salida sea de unos 2.5A, se puede alimentar sin problemas.

Uno de los puntos fuertes con los que cuenta esta placa es la inclusión en su tercera generación de conectividad WiFi y Bluetooth. Gracias a la conexión inalámbrica es mucho más sencillo acceder a internet sin depender de un cable y sin la necesidad de estar próximos al punto de acceso. Se adquiere mucha libertad para usar el aparato en proyectos en los que la placa vaya montada sobre algún dispositivo en movimiento.

2.1. GPIO

Los pines GPIO son unos puertos de entrada y salida, la Raspberry Pi cuenta con 40, se pueden ver en dos filas situados en un lateral de la placa. La versatilidad de estos pines es que son programables, podemos establecer valores de salida o leer de ellos valores de entrada. Gracias a ellos podemos manejar multitud de periféricos como luces, motores, otros microcontroladores y permiten llevar a cabo multitud de proyectos.

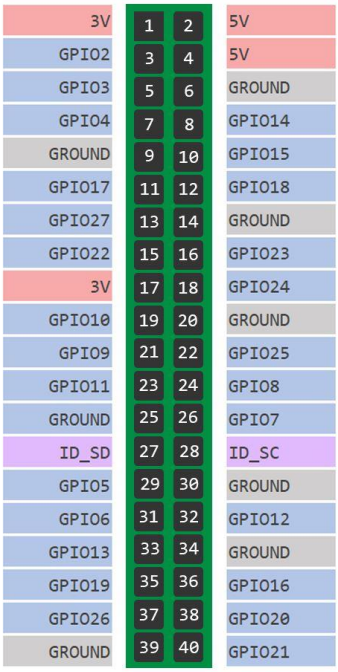

Figura 10: Esquema Pines GPIO

Cuenta con varios tipos de pines. Entre ellos nos encontramos con pines de alimentación, programables y de comunicación. Los primeros, permiten alimentar la electrónica que vamos a conectar a la Raspberry Pi, contamos con varias salidas de 3.3V, 5V y tierra. La mayoría son pines programables, es decir, que podemos establecer en ellos niveles altos o bajos o configurarlos como entrada, para leer valores tanto digitales, como analógicos. No todos son iguales, algunos soportan ciertas funcionalidades como modulación por ancho de pulsos

(PWM) o protocolos de comunicación como SPI o I2C. Por último, hay dos pines dedicados a la interfaz de la EEPROM.

2.2. Alternativas

Tras el éxito cosechado por la Raspberry Pi han surgido multitud de placas con características similares. Entre ellas podemos encontrar modelos que superan con creces la potencia de la Raspberry u otros con un precio imbatible, todas con unas características comunes: tamaño reducido, bajo coste, salidas programables y potencia suficiente para ejecutar un sistema operativo completo.

- PINE A64
  La PINE A64 (PINE A64) es una placa de computación con un procesador Quad-Core ARM Cortex A53 de 64 bits. Cuenta con 2GB de memoria RAM. Cuenta con puertos de todo tipo, desde los clásicos USB, HDMI y Ethernet, hasta receptor IR, puerto para batería externa, o puerto para una batería para el reloj, así como diferentes buses. También, viene con conectividad WiFi y Bluetooth.
  Es posible instalar multitud de distribuciones de Linux, Android e incluso Windows 10.

- Orange Pi
  Orange pi cuenta con diversas placas de computación. Cada una con unas especificaciones ligeramente diferentes. Entre ellas, por sus buenas especificaciones, destaca la Orange Pi Plus 2. (Orange Pi) Este modelo cuenta con un procesador Quad-Core ARM Cortex A7 y 2 GB de memoria RAM. Incluye diferentes puertos, USB, GPIO, Ethernet, interfaz SATA y HDMI. Conexiones inalámbricas Wifi y Bluetooth. Además, contiene 16GB de memoria directamente en la placa, con lo que no sería necesaria una memoria externa para grabar el sistema operativo, los cuales pueden ser Windows, Ubuntu o Debian.

- Asus Tinker Board
  Esta placa del conocido fabricante de ordenadores Asus, (Asus) monta un procesador Rockchip Quad-Core RK3288 junto a 2GB de memoria RAM. Al igual que las anteriores, dispone de varios puertos, entre ellos, USB, HDM, Ethernet, GPIO, entre otros. También, cuenta con conectividad WiFi y Bluetooth. El sistema operativo corre a cargo de una distribución de Debian llamada TinkerOS. Esta placa presume de tener un gran rendimiento, doblando en algunos casos la potencia de sus competidores.

La elección de la Raspberry Pi para este proyecto es debida básicamente a la gran comunidad de desarrolladores que tiene detrás. El rendimiento de la placa cumple perfectamente ya que los programas que van a correr sobre ella no requieren gran potencia de cálculo. En general, con los puertos básicos de la placa se suplen las necesidades, ya que principalmente se usarán el USB y los pines GPIO. Por último, se buscaba resolver los posibles problemas que se fueran encontrando de forma rápida, por lo que la comunidad que ampara la Raspberry Pi ofrece la posibilidad de encontrar respuestas a las dificultades en un tiempo razonablemente rápido.

Capítulo 3. Reconocimiento de voz

El reconocimiento de voz por software es la tecnología que permite a las máquinas entender el lenguaje humano. Poco a poco, las formas de interaccionar con los aparatos tecnológicos van cambiando. Gradualmente, escribir, hacer click o tocar, van siendo sustituidas por reconocimiento de voz. Obviamente, no todas las interacciones humanas con las máquinas pueden ser sustituidas por conversaciones pues algunas tareas se volverían más complicadas. Sin embargo, la tecnología actual permite que muchas de las funciones que se hacían antes puedan ser realizadas hablando directamente con la máquina.

Hoy en día, existen multitud de situaciones donde el reconocimiento de voz se ha establecido como sistema de interacción por omisión. Desde dictar una nota o un documento, dar instrucciones al móvil o al GPS mientras conducimos, llamadas a servicios de atención al cliente, hasta decirle al sistema de sonido de una casa que cambie de canción o haga la compra. Todas estas tareas todavía resultan un tanto extrañas para la mayoría de las personas, pues no hay costumbre de ello, pero, en un futuro no muy lejano, estaremos completamente rodeados de aparatos capaces de mantener una conversación totalmente fluida, lo que permitirá que las personas tomen de forma natural usar la tecnología con la voz.

En un principio, los sistemas de reconocimiento de voz se basaban únicamente en la habilidad de distinguir palabras. Los primeros sistemas distinguían desde simples números hasta unas pocas palabras. El funcionamiento consistía en procesar el audio de manera que, dividiéndolo en fonemas, coincidiera con los fonemas almacenados y así, devolver una posible palabra. Con los años, el vocabulario fue creciendo.

Hasta entonces, para su correcto funcionamiento, los mecanismos existentes necesitaban recoger un audio claro y con pausas entre palabras. La posibilidad de distinguir un vocabulario fluido y sin pausas se fue desarrollando hasta alcanzar lo que se conoce como “Lenguaje Natural”. Con la llegada de los móviles modernos y las conexiones a internet de gran ancho de banda desde cualquier parte, el reconocimiento de voz se ha ido estandarizando, pues el procesado del audio no se hace en el propio dispositivo si no que se envía hasta un centro de procesamiento que devuelve el texto. Con la gran potencia de cálculo de los servidores donde se trata la señal de audio, es posible desarrollar tecnologías basadas en sistemas mucho más complejos donde el aprendizaje juega un papel fundamental. Permitiendo afinar con el uso el índice de acierto al reconocer las palabras y aprender patrones del lenguaje.

Gracias a esto, desde hace pocos años hasta hoy, la proliferación de asistentes de voz ha ido en aumento. Los asistentes de voz son software instalado en todo tipo de aparatos electrónicos, desde teléfonos móviles, altavoces o televisores, que permiten, mediante órdenes de voz, manejar el aparato, o incluso, mantener conversaciones más o menos fluidas para preguntar por datos que nos interesen. O incluso hacer chistes.

Con los avances en inteligencia artificial, sistemas de aprendizaje automático y la creciente capacidad de cálculos de cualquier artilugio electrónico de la actualidad, las posibilidades que pueden llegar son innumerables.

Los sistemas de conversión de texto a voz, a su vez, están siendo ampliamente investigados, llegando a lograr una capacidad de habla prácticamente indistinguible al habla humano.

Con todo esto, lo que se puede esperar es que, en un periodo no muy largo de tiempo, la forma principal de interacción con las máquinas se haga de forma natural mediante la voz.

Capítulo 4. Análisis de Requisitos y Especificaciones

En este capítulo, se analizará lo que quiere construir, qué requisitos básicos ha de cumplir, algunas restricciones con las que contará para no desbordar el tiempo de implementación y desarrollo, así como el detalle de partes cruciales como lo serán los comandos de voz.

4.1. Objetivo general

Se quiere construir un ajedrez donde el jugador mueva las fichas mediante comandos de voz. El ajedrez será capaz de jugar de forma autónoma haciendo frente a un jugador.

4.2. Requisitos funcionales

Se han definido una serie de requisitos que el prototipo ha de cumplir. Estos requisitos han sido concretados la mayoría al inicio del proyecto, pero otros han sido dispuestos a medida que se avanzaba sobre la idea inicial.

Requisito 1: El sistema será capaz de mover las fichas realizando los movimientos legales del ajedrez.

> Este movimiento ha de realizarse por cada ficha siguiendo las directrices del reglamento oficial de ajedrez (Federación Internacional De Ajedrez). Para fichas que requieren de un movimiento complejo como es el caballo, se realizará posicionando la ficha por el borde de las casillas para poder saltar las piezas. Cuando una pieza captura a otra ésta se llevará de igual forma por el borde de las casillas hasta el exterior del tablero.

Requisito 2: El sistema será capaz de reconocer y ejecutar comandos de voz referentes a una notación estándar de ajedrez.

> Los comandos de voz serán de la forma:
>
> *<Ficha><Numero de ficha><Coordenada X><Coordenada Y>*
>
> Las fichas irán numeradas del 1 al 8 para poder identificar cada una por separado. Es decir, los peones se numeran del 1 al 8 de izquierda a derecha. Las torres, los caballos y los alfiles del 1 al 2 de izquierda a derecha también y por último la reina y rey con el 1.

Requisito 3: Se requiere un sistema de inteligencia capaz de generar movimientos legales de ajedrez en consonancia con el estado del tablero con el objetivo de ganar el juego.

> En el turno del ajedrez, la inteligencia artificial, conociendo el estado del tablero tiene que ser capaz de generar un movimiento válido y coherente con el objetivo de ganar al jugador.

Requisito 4: El sistema contendrá un pulsador para lanzar el reconocimiento de voz.

> Debido a la restricción número 4 es necesaria la inclusión de un pulsador para dar comienzo a la escucha que desembocará en el reconocimiento del comando dado por el jugador.

Requisito 5: El sistema contará con una interfaz de usuario para informar del turno en curso.

> Unos indicadores deberán dar a conocer el turno en el que se encuentra la partida en cada momento siendo fácil para el jugador reconocer cuando le toca.

4.3. Restricciones

Se han definido una serie de restricciones que aseguran la viabilidad, en su mayor parte temporal, del proyecto. Esto es así puesto que la implementación de algunas de ellas requeriría de mucho tiempo de investigación y desarrollo y se distanciaría del objetivo.

1. El sistema debe estar alimentado por una misma fuente.
2. El sistema debe estar conectado a internet para llevar a cabo el reconocimiento de voz. Implementar un sistema de reconocimiento in situ es harto complejo, se aprovecharán servicios externos que son bastante fiables.
3. No se realizarán movimientos especiales de ajedrez tales como el enroque y la promoción de los peones. Las trayectorias para realizar estos movimientos no son triviales, la carencia de éstos no impide jugar.
4. No se implementará un sistema de escucha permanente. Pocos servicios de reconocimiento de los que se dispone permiten esta funcionalidad.

Capítulo 5. Diseño de la solución

Con las especificaciones dadas en el apartado anterior se ha diseñado un producto que se ajusta perfectamente y que cumple con lo descrito. El producto será un ajedrez de tamaño normal fabricado en madera con todas las piezas, blancas y negras.

Interiormente para mover las piezas se dispone de un cabezal intersecado por dos ejes de metal que se mueven de forma perpendicular uno a otro. El cabezal de madera incorporará un motor unido a un imán permanente que, accionando el motor hacia un lado o a otro, moverá el imán arriba y abajo, pudiendo sujetar por debajo del tablero la pieza deseada, dotada de un pequeño trozo de metal. El movimiento de los ejes se producirá a través de cuatro motores, dos para cada eje.

La encargada del cerebro del ajedrez será una Raspberry Pi 3. En ella estará el programa principal que controlará el movimiento de las piezas mediante los ejes y el imán, recibirá las órdenes del jugador, mostrará mediante dos LED el turno de la partida y albergará la inteligencia artificial que dará vida al oponente del jugador.

También se utilizarán módulos electrónicos, para manejar los motores a través de la Raspberry. Estos elementos (Puentes H) permiten operar mayor cantidad de Intensidad de corriente ya que la salida que proporcionan los pines GPIO de la Raspberry Pi no son suficiente para hacer funcionar los motores. Por tanto, para el motor del imán se ha elegido un integrado que contiene dos puentes-H y para los motores un módulo con otros dos puentes-H. La naturaleza de todos estos componentes incluida la Raspberry Pi y componentes pasivos como resistencias y diodos será explicado en capítulos posteriores con más detalle.

El lenguaje utilizado para programar el proyecto ha sido Python, es un lenguaje con muchas posibilidades que, mediante librerías de terceros, permiten una programación agradable y sencilla. Se utilizarán básicamente tres librerías de terceros. La primera será la encargada de controlar los pines de salida y entrada GPIO de la Raspberry PI. La segunda, ayudará con el manejo del sistema de reconocimiento de voz. Ésta recogerá el audio a través de un micrófono y lo devolverá en forma de texto para ser tratado. La tercera proporciona una base de un sistema de ajedrez donde poder realizar movimientos, comprobaciones y guardar el estado del tablero, a su vez proporciona una interfaz para comunicarse con la inteligencia artificial que generará los movimientos.

Esto es a grandes rasgos como está planteado el proyecto. En los siguientes capítulos se detallará el funcionamiento de cada parte y se verá cómo se une todo para dar con el producto final.

Capítulo 6. Herramientas usadas

Para la realización del proyecto se han utilizado diversas herramientas software. Entre ellas se encuentran editores de código, programas de comunicación e incluso aplicaciones de modelado 3D.

Putty

Putty es un cliente SSH que permite conectarse a sistemas Unix (como el usado en la Raspberry Pi) desde Windows mediante este protocolo (Chiark Greenend). Es de código libre y su uso no entraña ninguna complejidad.

El proyecto ha sido desarrollado sobre una Raspberry PI con Raspbian como sistema operativo. Ésta no ha sido conectada a dispositivos de salida de video ni dispositivos de entrada como teclados o ratones, con lo cual para desarrollar sobre ella es necesario conectarse remotamente mediante SSH. Al utilizar un ordenador con Microsoft Windows para conectarse es necesario el uso de este cliente SSH. Su funcionamiento es sencillo, basta con iniciar la aplicación e Introducir la dirección IP de la Raspberry. Una vez conectado, pedirá usuario y contraseña y ya se tendría acceso total mediante un terminal remoto. Existen otras alternativas con muchas más opciones como el uso de varias sesiones en pestañas, exploración de archivos con copia *drag and drop* y muchas más, pero para el proyecto a desarrollar, la única necesidad es abrir una sesión en la máquina de destino con la que poder ejecutar código y explorar a grandes rasgos ciertas carpetas del sistema de ficheros, con lo cual, Putty resuelve a la perfección todo esto.

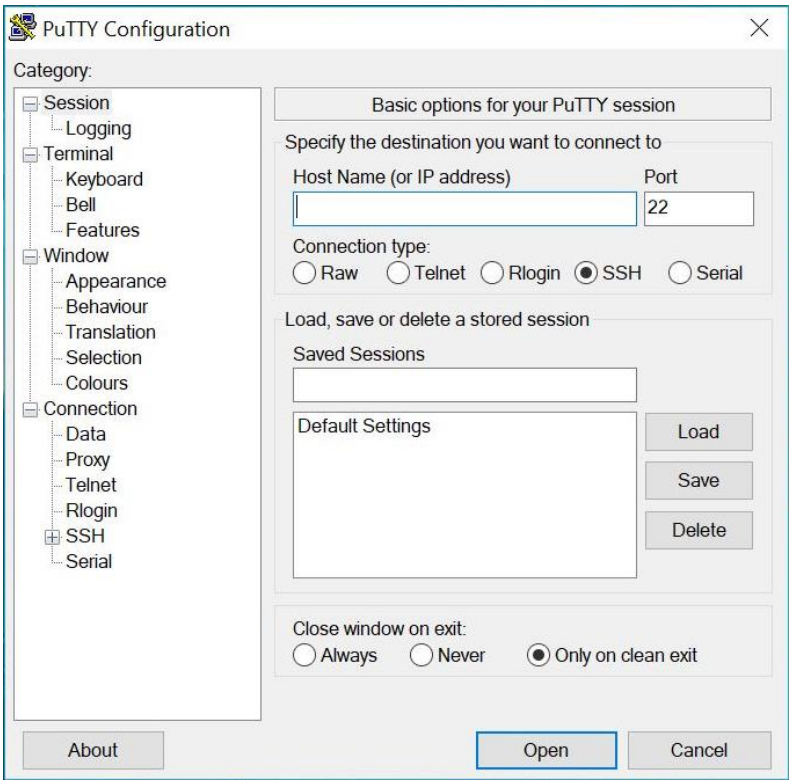

Figura 11: Pantalla principal de Putty

SSHFS Manager

SSHFS Manager es un cliente SSHFS para Windows (Code Google Archive). SSHFS es un protocolo que permite tener acceso seguro a un servidor remoto. Permite montar un sistema de archivos remoto en una máquina local.

De este modo es posible tener acceso a todos los ficheros de la Raspberry en una máquina de forma remota, permitiendo copiar, cortar, borrar y modificar cualquier archivo de forma totalmente transparente, como si de un disco propio conectado a la misma máquina se tratara.

Al igual que la herramienta anterior, su funcionamiento es muy similar, se introduce la dirección IP de la Raspberry, usuario, contraseña y el directorio a partir del cual se montará la unidad, permitiendo incluso a partir del directorio raíz.

Para el propósito del proyecto, esta herramienta cumple a la perfección, es rápida, sencilla y ligera. Otras alternativas ofrecen experiencias similares con lo que la elección de dicha aplicación se debe únicamente a su sencillez.

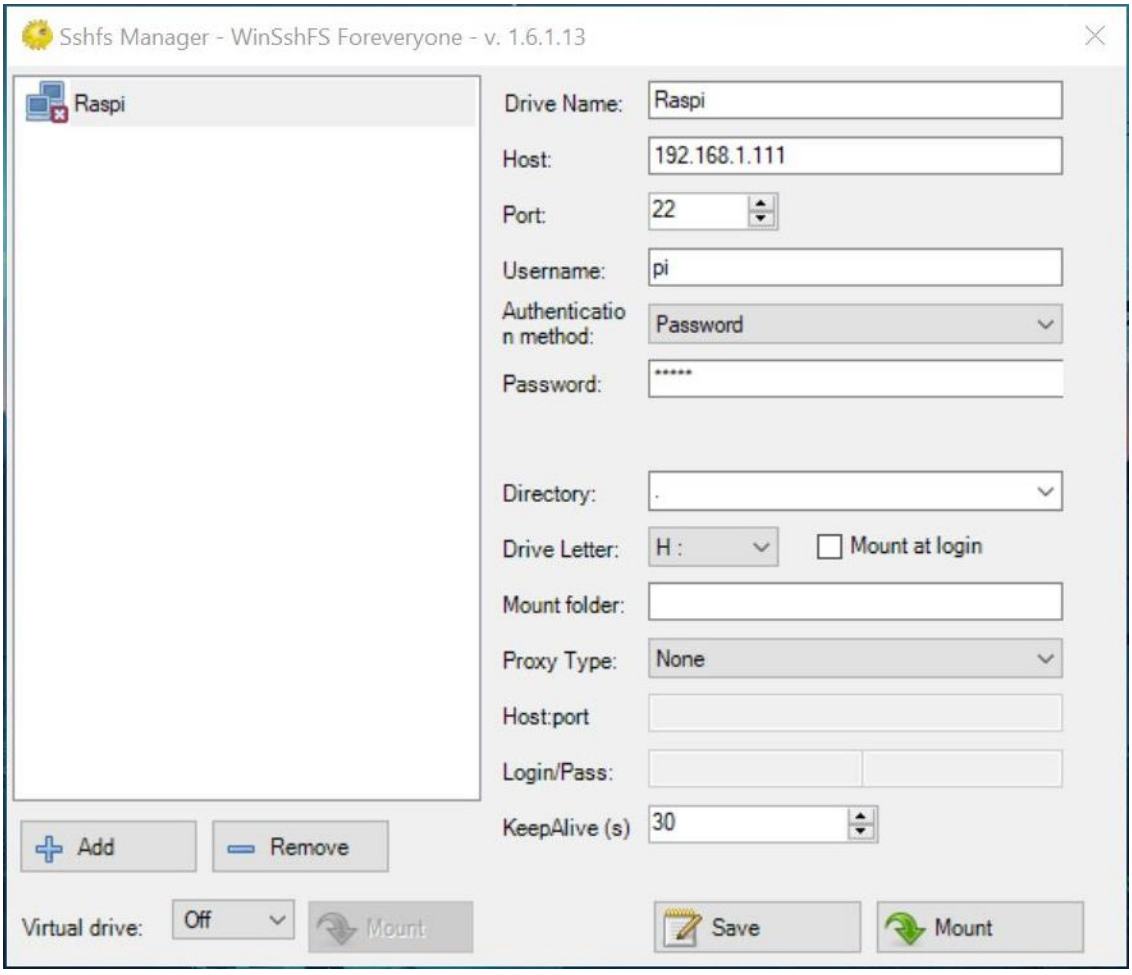

Figura 12: Pantalla principal SSHFS Manager

Sublime Text 3

Sublime Text 3 es un editor de texto para código (Sublime Text). Está disponible para múltiples plataformas y admite una gran cantidad de *plugins* a través de paquetes que extienden su funcionalidad.

La característica principal por la cual se ha usado este editor es la paleta de color del código. Como todo editor de código ofrece un coloreado del código en correspondencia con el tipo de lenguaje. En este caso Python. Esto posibilita un desarrollo con una interfaz muy cómoda de leer. Se podrían haber usado editores más enfocados al lenguaje con el que se ha desarrollado el proyecto o incluso algún entorno de desarrollo completo, pero dada la baja complejidad del código no era necesario.

Se han usado además varios *plugins* como son el Bracket Highlighter para mostrar el principio y el final de los paréntesis, corchetes y demás signos de apertura y cierre que contienen los lenguajes.

Se ha elegido este editor por su sencillez y rapidez, es capaz de abrir grandes archivos de texto en poco tiempo y su interfaz carece de distracciones.

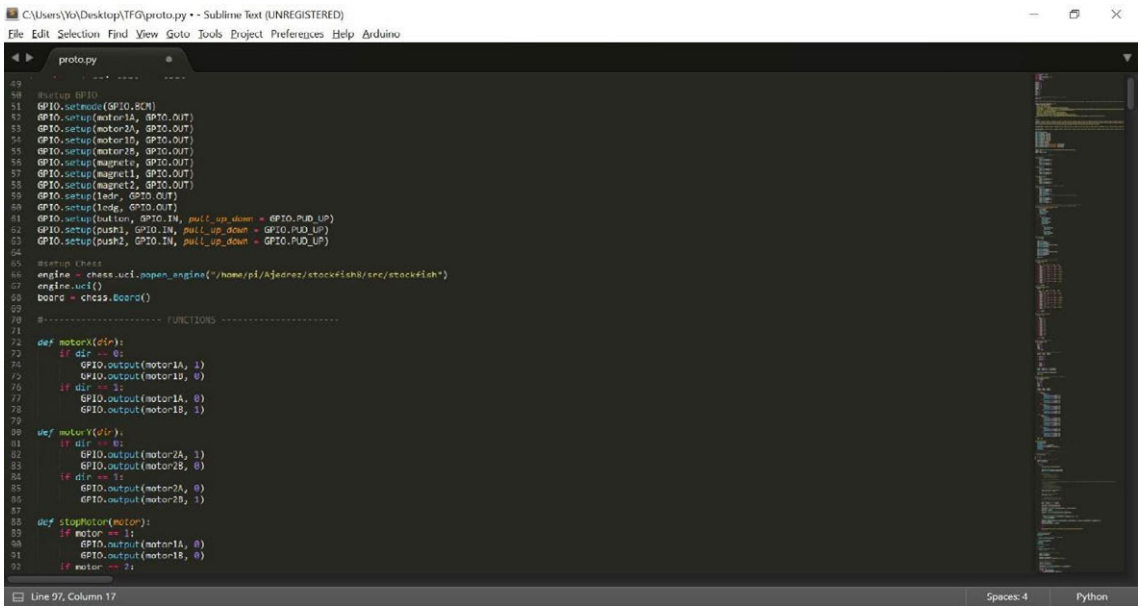

Figura 13: Editor de código SublimeText

Python

El Bundle de Python para Windows (Docs Python WIndows) incluye múltiples programas como el IDLE y una Shell. Permite el desarrollo de programas o script Python sobre Windows. El uso que se le ha dado a esta herramienta es la de comprobar el código ya que éste se ha desarrollado en Windows y posteriormente copiado mediante sshfs. Aunque el programa no se pudiera ejecutar, debido a que las librerías propias de la Raspberry, tales como la del GPIO, no funcionarían en Windows, es muy útil al menos comprobar que el código está bien escrito y tiene coherencia antes de pasarlo definitivamente a la Raspberry.

El paquete de Python para la Raspberry ya venía preinstalado en la imagen de Raspbian. Para hacer uso de las librerías empleadas, se ha utilizado la herramienta incluida PIP. Es un gestor de paquetes que permite descargar de una manera sencilla cualquier librería del repositorio de software de Python. Basta con escribir en el terminal:

pip install nombre-libreria

o, si se utiliza la versión 3:

pip3 install nombre-librería

Una vez hecho esto ya está la librería disponible para usar en cualquier programa.

Vim

Vim es un editor de texto integrado en la mayoría de sistemas Unix. El editor se ejecuta en modo texto y dispone de varios modos de funcionamiento, se pueden ejecutar multitud de comandos que permiten realizar multitud de operaciones. Se puede desde modificar texto hasta ejecutar comandos en una Shell. Una característica destacada es que mediante un fichero de configuración tener la posibilidad de modificar el comportamiento del editor y ejecutar una serie de configuraciones por defecto. Además, permite el uso de *plugins*.

En este caso, este editor se ha usado para crear y modificar pequeños programas de prueba y rápidas modificaciones en el código del programa principal, por lo tanto, se han incluido varias configuraciones adaptando el editor a dichas necesidades. Primero se ha activado la sintaxis ya que mejora de forma notable su uso a la hora de escribir código. También se ha añadido la opción para que al pulsar el tabulador lo convierta en 4 espacios. Hay multitud de opciones más, que podrían ser útiles para desarrollar Python en Vim, aun así, como el uso que se le ha dado es más para hacer modificaciones en los archivos de código copiados desde otro entorno y no para desarrollar todo el código del proyecto, con estas dos opciones es más que suficiente.

SketchUp

Para el diseño de la estructura física del ajedrez se ha usado SketchUp (SketchUp). Es un programa de modelado 3D. Su uso es muy sencillo y, al contrario de las demás herramientas de modelado que existen, la curva de aprendizaje es muy pequeña, permitiendo un manejo fluido de la herramienta al poco tiempo de conocerla. Al contrario de lo que se pueda pensar, se pueden crear modelos bastante complejos con este editor.

Con el auge de la impresión 3D, este software acerca a la gente menos experimentada a esta tecnología. Su sencillez permite que personas con poca experiencia en la creación de objetos para imprimir, puedan realizar sus proyectos sin ninguna complicación.

El programa, ofrece una serie de herramientas que permiten dibujar objetos de manera cómoda. Además, permite aplicar colores y texturas a las superficies, crear puntos de vista y multitud de opciones expandibles gracias a un centro de control de componentes.

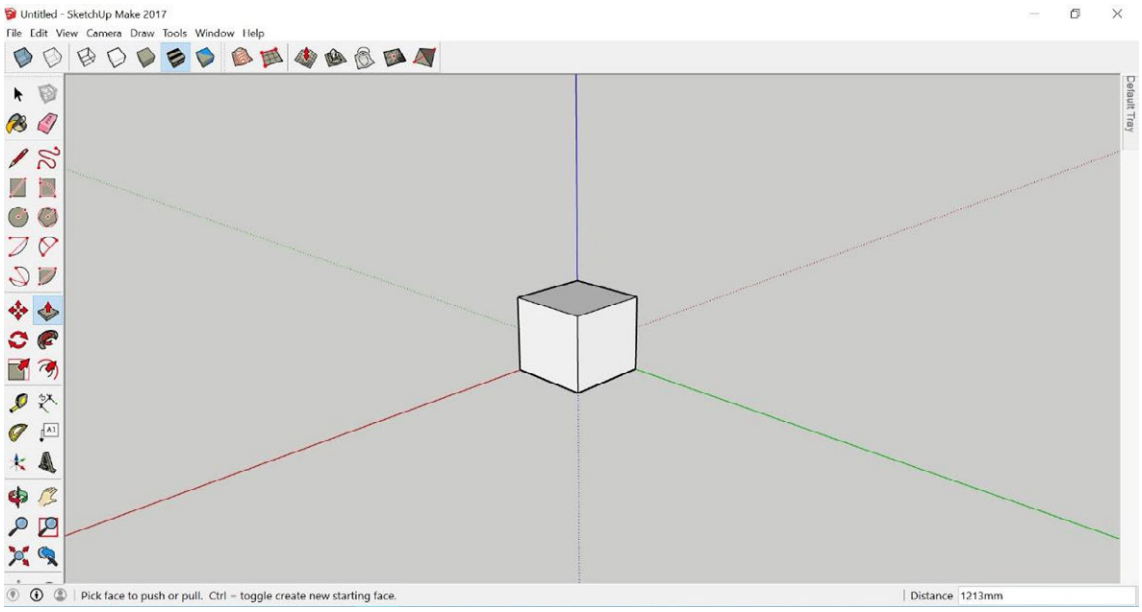

Figura 14: Pantalla principal de Sketchup

Capítulo 7. Implementación

Este capítulo se va a dividir en dos partes, por un lado, la construcción física del prototipo y por otro el desarrollo del software que lo controla. En la siguiente figura se puede ver el modelo 3D del producto completo. Este modelo se ha usado como referencia virtual a la hora de construir el prototipo real, tanto para fijar las medidas que ha de tener como para la construcción de pequeñas piezas que se han impreso con impresora 3D.

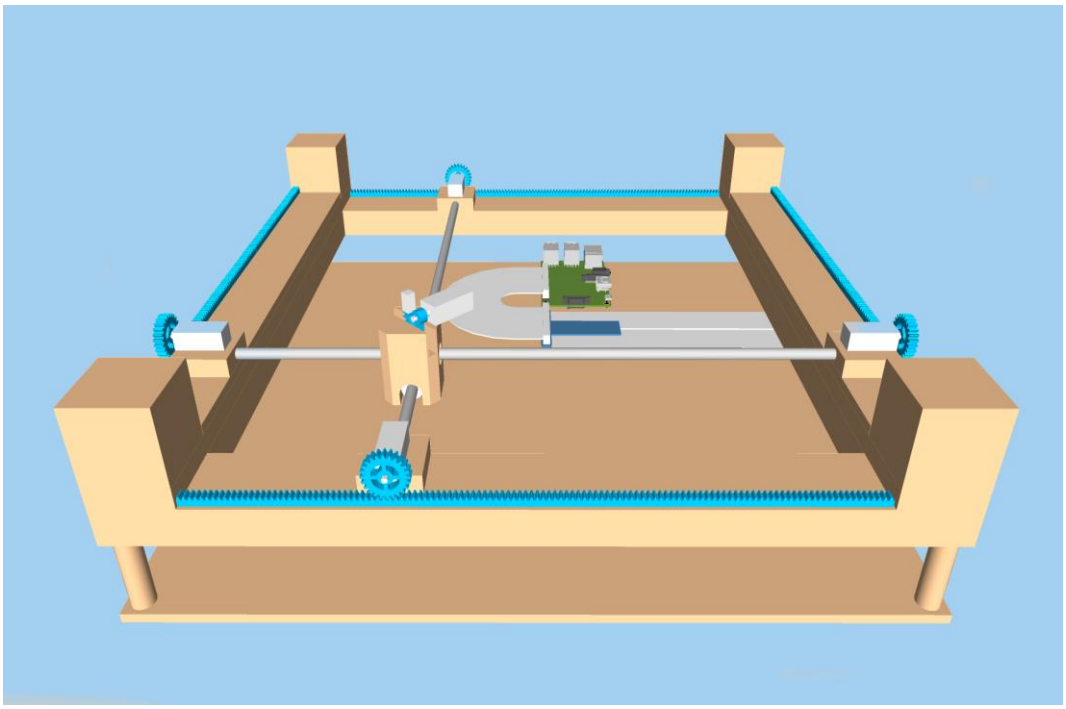

Figura 15: Modelo 3D del ajedrez

7.1. Construcción

La construcción del prototipo es lo que más tiempo ha requerido. Esto es debido a diversos factores, por un lado, las dificultades que han ido apareciendo a lo largo de la construcción, materiales que no sirven, estructuras deficientes, etc. Y por otro lado los problemas a la hora de adquirir los productos. Más adelante se verá más en detalle todos estos problemas y como se han solucionado.

7.1.1. Estructura

La estructura del ajedrez se ha intentado simplificar lo máximo posible pasa así poder reducir costes en material manteniendo la funcionalidad. Para ello se ha utilizado únicamente dos tipos de listones de madera de haya de 40mmx10mm y de 40mmx20mm de sección. A partir de segmentos de dichos listones se ha construido la mayor parte de la estructura (Figura 16).

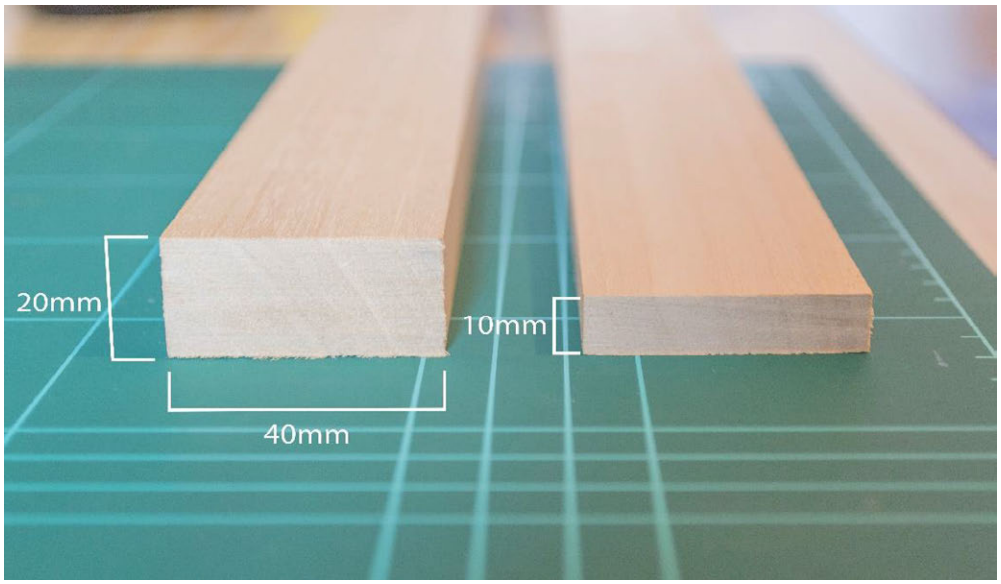

Figura 16: Listones de madera de la estructura

Sobre este armazón irán los ejes que moverán el cabezal, a su vez se sostendrá el tablero sobre el que irán las fichas. La estructura que conseguida es bastante robusta, no es demasiado ligera ni estética, pero para la construcción de un prototipo cumple su función perfectamente. Debido al volumen de la electrónica se ha optado por elevar el conjunto ligeramente, para ello se han colocado 4 pilares en las esquinas de la estructura que irán encima de una base donde se sujetará toda la electrónica. Estos pilares están construidos por una varilla de madera de 12 mm de diámetro. El motivo de la elección de las varillas es para poder dejar hueco suficiente para poder manipular sin impedimentos la electrónica del conjunto. Si bien ésta se ha ensamblado y probado antes de introducirla en la estructura, cualquier problema surgido del movimiento o cualquier rotura de algún componente podrá ser subsanada sin problema. El conjunto finalizado (Figura 17) provee al ajedrez de un buen soporte para todos sus componentes.

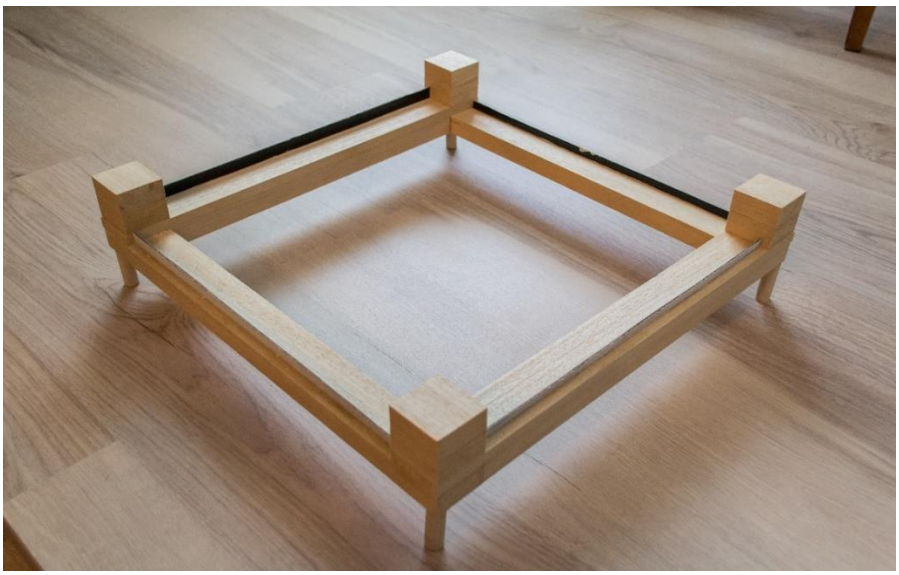

Figura 17: Estructura principal del prototipo

7.1.2. Ejes

Una de las partes más importantes del ajedrez es el sistema que permite mover las fichas. Este sistema está compuesto por los ejes y el cabezal con el imán (Figura 18), que se detallará más adelante en el apartado de electrónica. Los ejes forman un conjunto que permite al cabezal moverse libremente por todo el tablero, permitiendo alcanzar todas y cada una de las posiciones del tablero de juego y realizar movimientos por separado, es decir, un solo eje por movimiento, o en conjunto, moviendo a la vez ambos ejes.

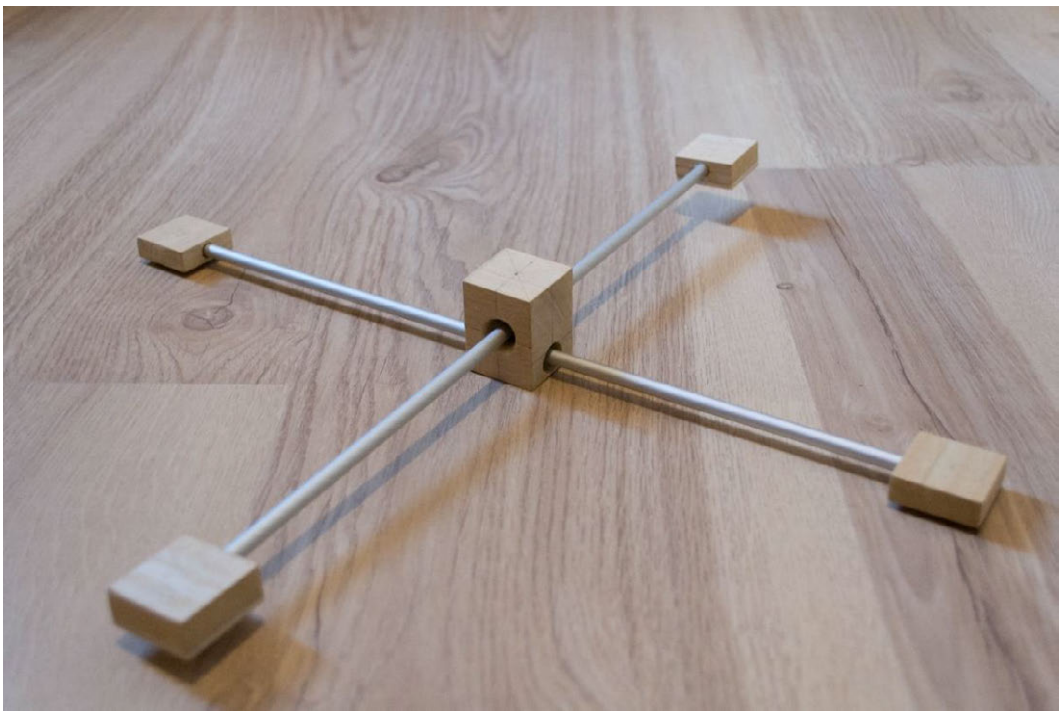

Figura 18: Ejes y pivote central

Los ejes están formados por dos varillas de aluminio huecas de 6 mm de diámetro. Las varillas han de ser de metal, pues necesitan aguantar una carga bastante grande sin deformarse, además, al tratarse de aluminio anodizado, tiene una superficie lo suficientemente deslizante, permitiendo que el cabezal se deslice sin problema sobre ellas. En cada extremo, hay un pequeño taco de madera en los que se insertará la varilla, éstos servirán como rail, permitirán deslizar todo el conjunto sobre la estructura. Debajo de cada taco de madera se encuentran unas pequeñas bolitas de metal (Figura 19).

Deslizar el taco de madera directamente sobre la estructura genera una fricción demasiado grande. Se pensó en barnizar y pulir ambas superficies, pero los resultados no eran esperanzadores ya que el barniz también generaba demasiada fricción. Al final, el uso de estas bolitas extraídas de los raíles de un cajón, dieron mucho mejor resultado.

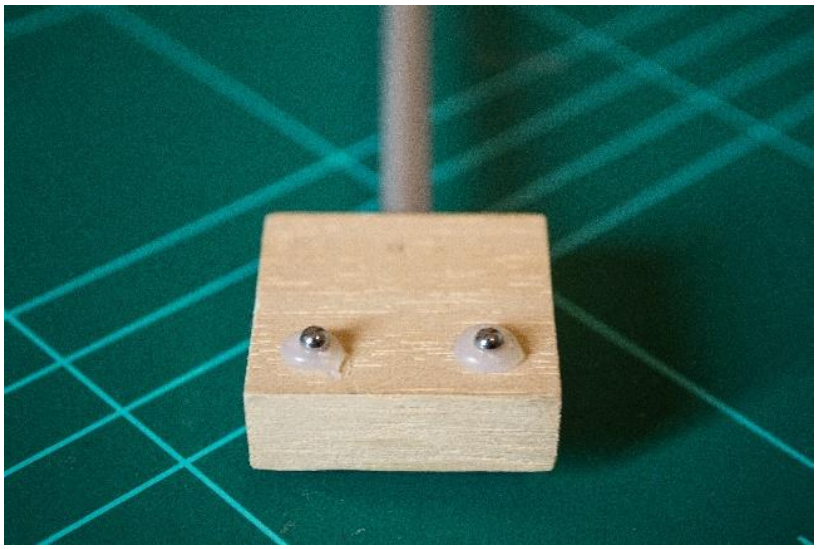

Figura 19: Parte inferior de los apoyos de los ejes

Esto ayudará a disminuir la fricción de la madera y permitirán al eje desplazarse con soltura a lo largo del rail. A su vez, proveerá al conjunto de los ejes una buena base de apoyo.

El cabezal se compone de una pieza de madera de 30x30mm y 40mm de alto (Figura 20) En ella hay perforados dos orificios cruzados en perpendicular, uno más arriba del otro para que los ejes no se toquen. En dichos orificios se introducirán las varillas permitiendo que el cabezal se deslice por ellas.

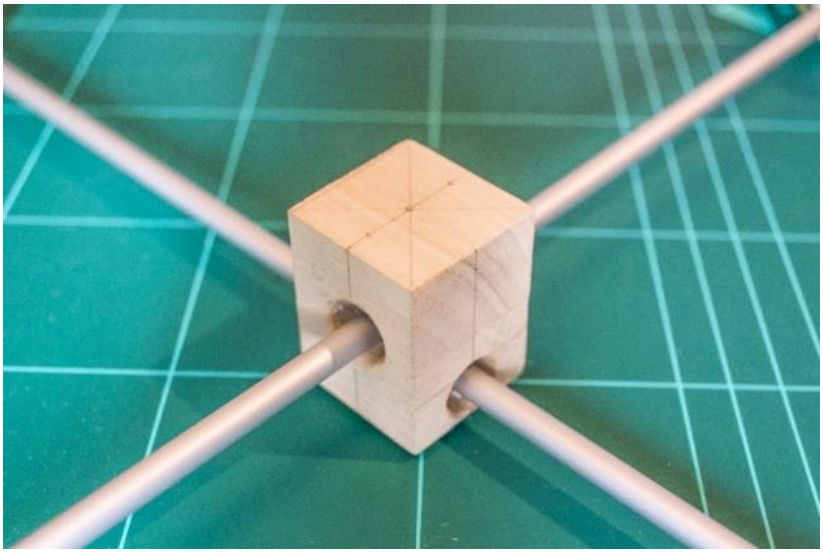

Figura 20: Cabezal

La fricción entre la madera y el metal ocasionaba un mal deslizamiento del cabezal a lo largo del eje, generando paradas bruscas y ocasionando un movimiento muy poco fluido. Para que las varillas se deslicen sin problema se han dispuesto en cada orificio un rodamiento lineal (Figura 21) de 12mm de grosor y con el orificio del mismo diámetro que las varillas.

Figura 21: Rodamiento lineal

Este sistema permite a cada eje moverse de forma independiente en cada dirección mientras que el cabezal se situará en la intersección de ambos con lo que, al moverse, éste podrá alcanzar cualquier posición del plano formado por la longitud de ambos ejes.

Encima del cabezal se situará el motor que hará subir y bajar el imán permanente solidario a su eje. Para permitir este movimiento se ha hecho un corte en el cabezal (Figura 22) facilitando el movimiento de la estructura que sujeta el imán. Más adelante se verá por qué se ha usado este sistema, algo rudimentario, y no uno más elegante, como un electroimán.

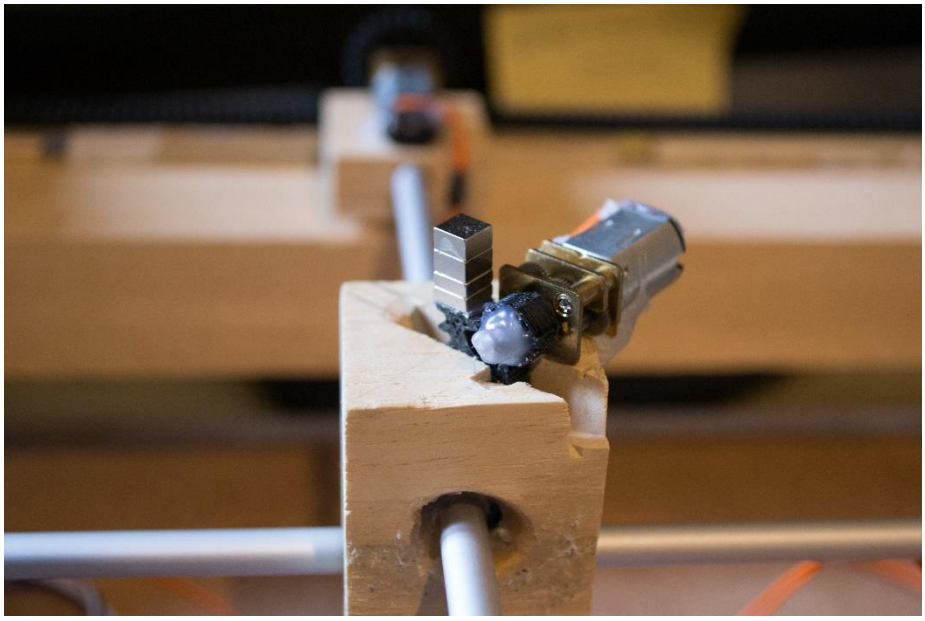

Figura 22: Cabezal con el mecanismo de sujeción de las piezas

7.1.3. Movimiento

Una vez está la estructura montada sólo queda el actuador con el que se moverán los ejes. Para ello se han dispuesto dos motores (Figura 23) en cada extremo de los mismos. Se trata de unos pequeños motores de corriente continua. Solidario a su eje se encuentra una reductora que disminuye la velocidad y aporta más fuerza. Alimentado con 6V desarrolla 100 RPM. En este caso la alimentación se realiza con una fuente de 5V con lo que la velocidad será ligeramente menor, pero suficiente para mover sin problema el conjunto.

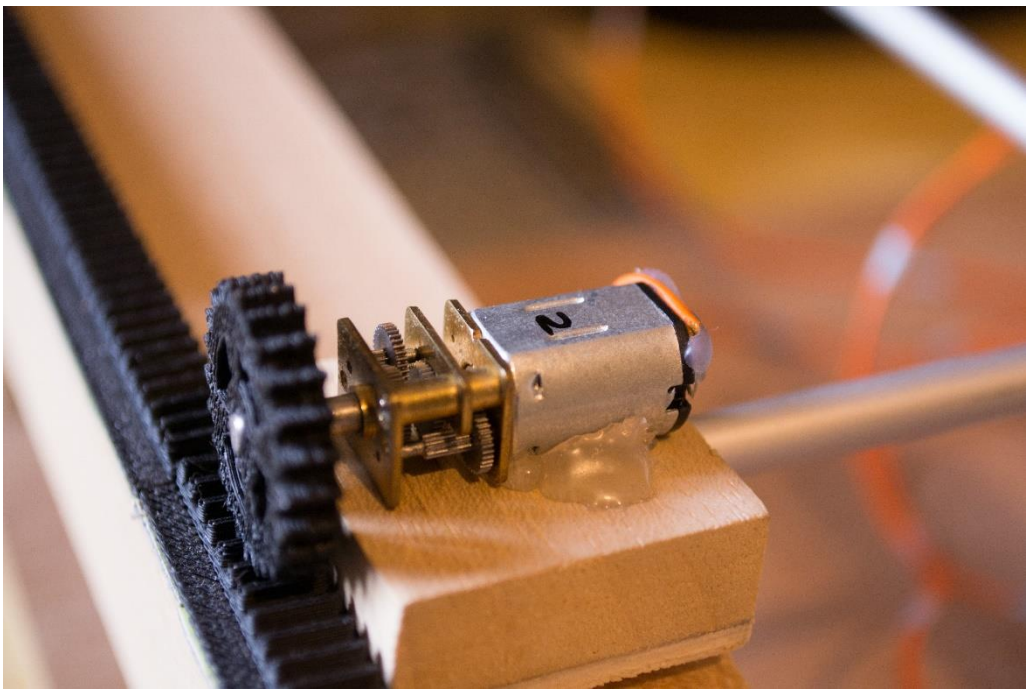

Figura 23: Motor de los ejes

En un principio se dispusieron únicamente dos motores, uno por eje, sin embargo, al tener una estructura tan tosca las fricciones generadas en los raíles y el peso de los ejes hacían que el extremo donde no estaba situado el motor se quedara estático con lo cual no conseguía mover el eje correctamente. Por ello se optó por poner un motor más por eje, así, teniendo uno en cada extremo el movimiento es totalmente uniforme.

Disponiendo de un presupuesto más grande se podría mejorar el diseño de los raíles, por ejemplo, con varillas de metal en cada extremo, similar a los escáneres de papel. Con este diseño más depurado y preciso, trabajando con bajas tolerancias, se podría prescindir de uno de los motores por eje.

En los ejes del motor se ha dispuesto un engranaje de aproximadamente 22mm de diámetro que rodará por una cremallera de 320mm a lo largo de cada lado (Figuras 24-25), de este modo será posible el movimiento. Tanto el engranaje como la cremallera estarán impresas con una impresora 3D. Esto permite adaptar a la perfección la pieza a las necesidades concretas del proyecto a la par que se ahorran costes, ya que la fabricación de un engranaje a medida por otro método sería mucho más costosa. En el

caso de la cremallera se ha tenido que cortar en 3 segmentos, dos de 120mm y uno de 80mm puesto que la cama de la impresora utilizada es mucho más pequeña que los 320mm de largo de la cremallera.

Los modelos se han obtenido de la página web de Thingiverse (jjnm5). Esto simplifica la tarea de modelar a mano un engranaje, que si bien no sería una tarea complicada, permite ahorrar algo de tiempo y muestra la facilidad con la que se pueden encontrar todo tipo de piezas que, con pocos conocimientos, son fácilmente modificables para adaptarlas a necesidades específicas. En este caso se ha elegido un pack del usuario *jjnm5* que tiene un paquete de engranajes de diferentes tamaños y modelos, una vez descargado se ha procedido a hacer modificaciones para adaptarlas al proyecto como, por ejemplo, modelar el centro del engranaje para que encaje perfectamente con el buje

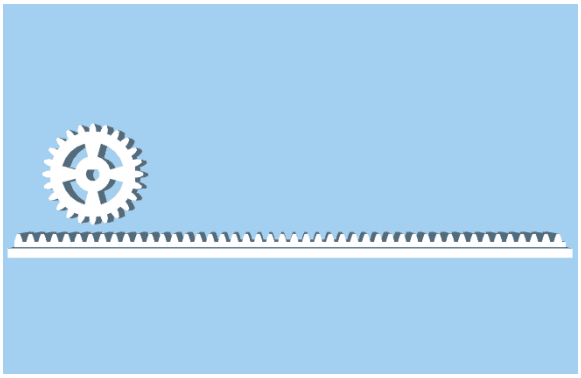

Figura 24: Modelo 3D de los engranajes

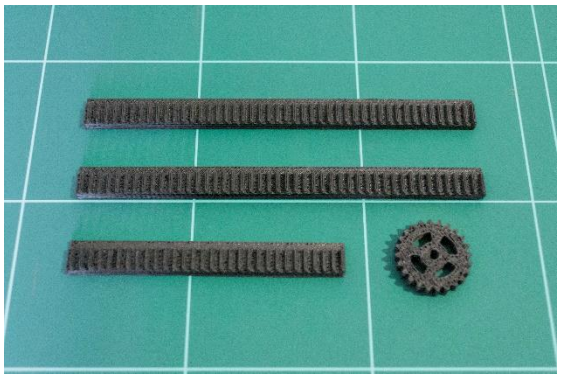

Figura 25: Impresión 3D de los engranajes

del motor.

7.1.4. Electrónica

Otra de las partes esenciales del proyecto es la electrónica ya que sin ella nada podría funcionar. La parte electrónica está compuesta por diversos componentes, dada la naturaleza del proyecto de servir para que cualquiera pueda construirlo se ha intentado hacer uso de diferentes módulos prefabricados, que son fáciles de entender y de coste reducido. El componente principal es el cerebro del juego, éste está formado por una Raspberry Pi 3 Model B. Este modelo tiene una CPU de 64 bits con 4 núcleos a 1,2GHz y 1 GB de memoria RAM. Incluye conectividad integrada WiFi y Bluetooth. Dispone de múltiples puertos como un HDMI, cuatro USB, así como salida compuesta de audio y video. La parte más interesante son los pines de entrada y salida, GPIOs, que servirán para ejecutar el control de los actuadores. Éste es el punto clave del componente ya que sus 40 pines dan la posibilidad de trabajar con multitud de componentes externos. A través de software se pueden activar y desactivar todos estos pines o incluso leer de ellos.

Figura 26: Raspberry Pi 3 Model B

Se ha elegido este componente ya que, gracias a la posibilidad de instalar un sistema operativo basado en Unix en ella, es posible programarla en multitud de lenguajes y ejecutar programas como el motor de inteligencia artificial del ajedrez que en otros sistemas no se podría. Además, dado su reducido tamaño es posible montarla en el interior sin que ocupe demasiado espacio. Otro punto a favor de este modelo es la conectividad inalámbrica WiFi con lo que la conexión a la red requerida se puede realizar sin cables y de manera sencilla.

3V	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GROUND
GPIO4	7	8	GPIO14
GROUND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GROUND
GPIO22	15	16	GPIO23
3V	17	18	GPIO24
GPIO10	19	20	GROUND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GROUND	25	26	GPIO7
ID_SD	27	28	ID_SC
GPIO5	29	30	GROUND
GPIO6	31	32	GPIO12
GPIO13	33	34	GROUND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GROUND	39	40	GPIO21

Figura 27: Pines GPIO

En el esquema de los pines GPIO (figura 27) se puede ver que no todos de ellos pueden ser usados como entradas y salidas. Hay pines de alimentación que dan 5V o 3.3V así como pines de tierra. También hay pines de comunicación mediante I2C o pines de conexión con la UART, los cuales sirven para comunicar sensores, módulos, o cualquier tipo de electrónica que soporte estos protocolos. Por último, dispone de multitud de pines entrada/salida. Estos pines son los que se pueden manipular mediante software, bien configurándolos como salida y sacar un nivel bajo o alto, o bien configurándolos como entrada y leer el nivel lógico de los pines. En el apartado del desarrollo del software se verá cómo se configura y se programa esta interfaz.

A parte del componente principal se han usado dos componentes más. El primer componente es un módulo HG7881 (ic2ic datasheet) que contiene dos puentes-H, L9110.

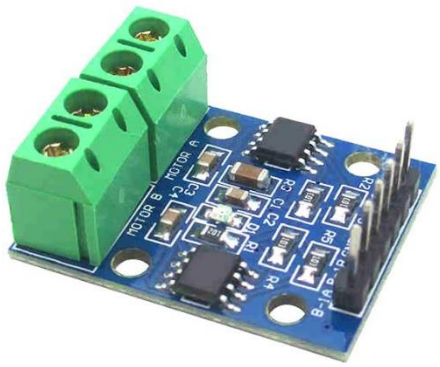

Figura 28: Módulo HG7881

Un puente H es un circuito electrónico que permite manejar un motor de corriente continua, haciendo posible el giro en ambos sentidos. Su uso con transistores es muy útil ya que permite manejar motores con señales provenientes de un microcontrolador. Las salidas de un integrado de control proporcionan como mucho 5V y, dependiendo del modelo, una intensidad de corriente muy baja. En concreto, la Raspberry Pi 3 puede drenar hasta 50mA de sus GPIO con 3.3V (FAQ: Power Req. Raspberry Pi) mientras que un motor puede consumir picos de varios amperios. Con lo cual, su uso está más que justificado si no se pretende dañar la placa.

Estos serán los intermediarios entre las salidas de la Raspberry Pi y en este caso, los cuatro motores de los ejes. Este módulo permite controlar dos canales con lo que podemos controlar los dos ejes por separado. Cuenta con dos integrados que se controlan por separado mediante dos entradas. Este módulo también permite controlar un solo motor paso a paso de 4 entradas. En este caso lo se utilizará para controlar los 4 motores DC normales, dos por cada canal, van conectados en paralelo dos a dos. La tabla de verdad de las entradas es la que se muestra en la siguiente figura.

Input		Output		Descripción
IA	IB	OA	OB	
0	0	0	0	Off
1	0	1	0	Adelante
0	1	0	1	Atrás
1	1	1	1	Off

Figura 29: Tabla de verdad del HG7881

Como se puede ver el primer pin de cada canal se utiliza para encender o apagar el motor y el segundo para controlar la dirección. En el caso de que se quiera controlar la velocidad, podemos hacerlo mediante PWM (Pulse Width Modulation, Modulación por Ancho de Pulsos) en el primer pin.

El segundo componente es un integrado L298N (StMicroelectronics) (Figura 28) que incluye dos puentes H para controlar dos motores de forma independiente. Se ha usado

este integrado debido a una necesidad surgida de un problema a la hora de utilizar el electroimán para mover las fichas.

Figura 30: Integrado L298N

El planteamiento inicial era usar un pequeño electroimán alimentado a 5V junto con un módulo con un transistor MOS IRF520N (infineon datasheet) con el que se podría encender y apagar el imán desde la Raspberry. Tras probar esta configuración la fuerza del electroimán resultó ser muy deficiente sin llegar a poder mover una ficha de unos escasos gramos de peso. Ante tal situación se decidió usar un motor, el cual se tenía de un envío erróneo, igual al utilizado para mover los ejes, pero con una relación de velocidad menor, unido a un imán permanente.

Debido a que se necesitaba mover el motor en los dos sentidos para subir y bajar el imán, el módulo MOS utilizado para el electroimán no servía para mover el motor en ambos sentidos, por lo cual se tuvo que adquirir el integrado con el puente H. Éste quizás no es el más adecuado, puesto que es un integrado que soporta una potencia muy superior a la que se requiere y además necesita de circuitería adicional para hacerlo funcionar, pero debido al poco tiempo restante de proyecto y al escaso stock de productos en la tienda de electrónica se decidió usarlo ya que el funcionamiento es el mismo. Lo ideal sería usar un módulo como el utilizado para los motores de los ejes, o incluso alguno más simple que disponga de un solo canal.

El electroimán, que inicialmente estaba planeado utilizar, estaba alimentado por 5V. En teoría tiene fuerza suficiente para sujetar 2.5Kg de peso. Esto es sólo en teoría ya que, según las pruebas realizadas, la fuerza que genera deja mucho que desear. Con una alimentación de 5V no es capaz de sostener ni 1kg de peso y el campo magnético que genera es bastante débil en cuanto el objeto se aleja a pocos milímetros del electroimán. Por este motivo se decidió no usarlo, la fuerza generada no era suficiente para mover las fichas a través del tablero.

Con esto se tienen todos los componentes necesarios, faltaría añadir componentes pasivos. Se han utilizado dos resistencias de 220Ω para los LED, cuatro diodos y dos condensadores electrolíticos de 100nF para el integrado que controla el motor que mueve el imán, un pulsador y dos diodos LED, uno rojo y otro verde.

Adicionalmente se ha incluido un micrófono USB de pequeño tamaño conectado directamente a los puertos de la Raspberry Pi. Posteriormente, se decidió sustituir el

micrófono por otro de mayor tamaño y calidad ya que el sonido grabado era prácticamente ruido.

7.1.5. Tablero y piezas

El tablero (Figura 31) es una plancha de madera de densidad media. Tiene unas medidas de 400x400mm y un grosor de 2.5mm. Este grosor junto con la rigidez del material permite mantener el tablero recto para facilitar el movimiento del cabezal por debajo del mismo. A su vez, al ser tan fino, la fuerza del imán del cabezal es suficiente para mover las fichas sin problema.

Figura 31: Tablero

Está posicionado justo encima de la estructura. Para calibrar la altura de este y no tener problemas de rozamiento con el imán, el tablero descansa sobre unos tacos de cartón para regular la altura. La calibración de la altura se ha realizado poniendo el cabezal en las cuatro esquinas de tablero y añadiendo o quitando láminas de cartón hasta que el imán deje de tocar la madera.

Las líneas del ajedrez están calculadas posicionando el cabezal en sus límites máximos y dividiendo la longitud en 18 partes, las 16 partes centrales corresponden a las 8 casillas del juego y las 2 restantes, para una zona límite a ambos lados del tablero donde se posicionarán las fichas capturadas.

Las piezas del juego han sido reutilizadas de un viejo juego de ajedrez. Para adaptarlas al tamaño de las casillas, se han recortado la amplia base de la que disponían tal y como se ve en la figura. Así, la base de las piezas cabe dentro de las casillas y además posibilita que otras piezas pasen entre medias como el alfil o el caballo.

Figura 32: Proceso de recorte de la base

Para poder moverlas se les ha incorporado una pieza de metal en la parte de abajo. Se ha optado por usar tornillos pequeños pegados en el interior de la pieza ya que éstos son fáciles de conseguir, baratos y ofrecen una superficie suficiente para que el imán consiga fijar las piezas y desplazarlas a lo largo del tablero.

7.2. Desarrollo software

El software del juego ha sido pensado para ser lo más sencillo posible. Se ha optado por desarrollar la totalidad del código en lenguaje Python. El uso de este lenguaje tiene su motivo en el gran auge que está teniendo en la actualidad. Es un lenguaje muy potente con multitud de librerías. La sintaxis y la estructura hace que sea de fácil comprensión y el código sea totalmente legible para cualquiera. Además, al no haber utilizado antes este lenguaje supone una oportunidad de aprender, conociendo otras formas de escribir código favoreciendo la adaptación a futuros retos similares.

La estructura del código se basa en un único fichero con el programa principal. Éste depende de varias librerías externas que se pasará a analizar en profundidad más adelante.

El programa principal implementa un sistema de turnos, empezando por el jugador, seguido de la máquina y así en un bucle hasta que alguno de los dos gane el juego.

Ambos jugadores tienen similitudes en sus códigos, la diferencia reside en la forma en la que se recoge el movimiento. En el caso del jugador se obtiene mediante reconocimiento de voz y en el caso de la máquina es la parte de la inteligencia artificial quien devuelve el siguiente movimiento a realizar. Después de hallar dicho movimiento,

el procedimiento es prácticamente el mismo para los dos: traducciones, comprobaciones, movimiento virtual, movimiento físico.

Las traducciones son pasos intermedios necesarios para traducir el formato del movimiento que se recibe, como, por ejemplo, el texto plano reconocido de la voz del jugador, en un comando reconocible para el motor de ajedrez. Más adelante veremos las peculiaridades de dichos formatos.

Después de tener el movimiento en el formato correcto, se procede a comprobar si hay alguna ficha en la casilla destino donde se moverá la ficha elegida. Si la hay se procederá a expulsarla del tablero. También se comprueba qué tipo de ficha es la elegida, pues en el caso del caballo, el movimiento a realizar es diferente de los que se realizan de forma habitual. En este punto, es donde en un futuro podría implementarse la comprobación de si el movimiento es o no válido.

A continuación, se efectúan los movimientos virtuales, es decir, se realiza el movimiento usando el motor del ajedrez para actualizar la posición en el tablero virtual y después, se actualizan las posiciones en los registros internos del programa.

Por último, se procede a realizar el movimiento físico de la ficha por el tablero.

En los siguientes apartados se verá cómo se realizan ambos tipos de movimientos.

7.2.1. Motor de ajedrez

Para llevar la lógica del juego se ha optado por hacer uso de una librería de Python que ofrece multitud de posibilidades para manejar un ajedrez.

La librería se llama Python-chess y está disponible en el repositorio de software de Python (Python Package Index : python-chess, s.f.).

Incorpora un motor de ajedrez completo, es totalmente configurable y proporciona multitud de funciones para interactuar con el tablero, incorporando, además, un sistema de comunicación mediante el protocolo UCI (*Universal Chess Interface*) que servirá para comunicarse con la inteligencia artificial.

Los métodos de la librería que se han usado son:

`Chess.Board()`

Inicializa el tablero, devuelve un objeto Board.

`chess.uci.popen_engine(ruta_fisica_del_motor)`

Abre un proceso del motor de la inteligencia artificial.

`Engine.uci()`

Le dice al motor que use la interfaz UCI.

`Board.piece_at(square)`

Devuelve la pieza situada en la casilla indicada.

chess.square(file_index,row_index)

Devuelve un objeto tipo *square* según la fila y la columna indicada.

Board.push(movement)

Efectúa un movimiento.

Board.is_game_over()

Devuelve *True* o *False* dependiendo si ha habido jaque mate o no.

Engine.position(Board)

Se le indica el estado del tablero al motor de inteligencia artificial.

Engine.go(movetime=5000)

Devuelve el mejor movimiento posible. En este caso se usará un parámetro para indicar el tiempo máximo de cálculo. A más tiempo de cálculo, más preciso el resultado, es decir, más posibilidades de que el movimiento elegido sea el óptimo.

engine.stop()

engine.quit()

Detiene el motor de inteligencia artificial.

Únicamente con estos métodos y funciones, sería posible poner en marcha un juego de ajedrez completo, la cantidad de funciones, parámetros y configuraciones de las que dispone la librería, las cuales se pueden ver en la documentación de la misma, son añadidos que enriquecerían la experiencia y servirían para hacer un juego más robusto, además, permitiría jugar con ellas para desarrollar experimentos de todo tipo con el ajedrez.

Con todo esto, se pueden realizar las comprobaciones de si hay o no ficha en una casilla, hacer movimientos y calcular movimientos con la inteligencia artificial. La librería contiene de forma interna registros de las posiciones, estado del tablero, así como histórico de movimientos para poder hacer y deshacer.

A parte de los registros de la librería, se han implementado unos diccionarios para uso propio donde se guardan las fichas junto con la posición en la que se encuentran (Código 1).

```
whiteChessboard = {"torre 1":(1,1), "caballo 1":(2,1), "alfil
1":(3,1), "reina 1":(4,1), "rey 1":(5,1), "alfil 2":(6,1), "caballo
2":(7,1), "torre 2":(8,1), "peón 1":(1,2), "peón 2":(2,2), "peón
3":(3,2), "peón 4":(4,2), "peón 5":(5,2), "peón 6":(6,2), "peón
7":(7,2), "peón 8":(8,2)}
```

```
blackChessboard = {"torre 1":"a8", "caballo 1":"b8", "alfil 1":"c8",
"reina 1":"d8", "rey 1":"e8", "alfil 2":"f8", "caballo 2":"g8", "torre
2":"h8", "peón 1":"a7", "peón 2":"b7", "peón 3":"c7", "peón 4":"d7",
"peón 5":"e7", "peón 6":"f7", "peón 7":"g7", "peón 8":"h7"}
```

Código 1

La inteligencia artificial que que se ha usado es un motor de ajedrez *open source* llamado Stockfish. Utiliza el protocolo UCI y está disponible para múltiples plataformas.

Para utilizarlo basta con descargarlo y situarlo en una carpeta cualquiera. Después para configurarlo junto con la librería de ajedrez, basta con invocar el método *chess.uci.popen_engine(ruta_fisica_del_motor)* pasándole la ruta donde hemos dejado el motor.

7.2.2. Reconocimiento de voz

El apartado del reconocimiento de voz corre a cargo de una librería externa llamada SpeechRecognition. Al igual que la anterior está disponible en el repositorio de Python (Python Package Index: SpeechRecognition, s.f.).

Esta librería proporciona una interfaz para llevar a cabo reconocimiento de voz. Soporta múltiples motores de reconocimiento tanto online como locales.

Además de la librería, se requieren una serie de funcionalidades:

- La librería de Python PyAudio. Necesaria para usar el micrófono.
- La librería Google API Client Library for Python.
- FLAC encoder.

En la documentación de la librería se explica detalladamente cómo instalar estos requerimientos, así como ejemplos de todo tipo.

En este caso se ha usado la API de Google Cloud Platform (Google). En un principio se usó la API de Microsoft Cognitive Services debido a que era más sencillo de configurar. Pero, el reconocimiento de voz era poco preciso y Google ofrecía ciertas características que hacen que reconozca los comandos de voz a la perfección.

Google proporciona un servicio de reconocimiento de voz gratuito con límites. Hay que registrarse en Google Cloud Platform, abrir un proyecto de reconocimiento de voz y conseguir una clave. Esta clave, contenida en un fichero json, dará acceso mediante autenticación a los servicios de Google.

Una vez obtenida la clave, utilizar la librería es realmente sencillo A continuación, se puede ver un ejemplo de uso.

```
import speech_recognition as sr
```

```
r = sr.Recognizer()
with sr.Microphone() as source:
    print("Say something!")
    audio = r.listen(source)

GOOGLE_CLOUD_SPEECH_CREDENTIALS = r"""INSERT THE CONTENTS OF THE
GOOGLE CLOUD SPEECH JSON CREDENTIALS FILE HERE"""
try:
    print("Google Cloud Speech thinks you said " +
r.recognize_google_cloud(audio,
credentials_json=GOOGLE_CLOUD_SPEECH_CREDENTIALS, preferred_phrases=
commands ))
except sr.UnknownValueError:
    print("Google Cloud Speech could not understand audio")
except sr.RequestError as e:
    print("Could not request results from Google Cloud Speech service;
{0}".format(e))
```

Código 2

Como se puede ver lo único que hay que hacer es, iniciar el reconocedor y el micrófono, capturar un mensaje de audio e instanciar la llamada que desencadenará la petición al servidor de Google, pasando como parámetro el audio, el lenguaje y la clave.

A su vez, hay otro parámetro adicional en la llamada llamado *preferred_phrases*. Este parámetro contiene una lista de posibles frases que puede contener el audio a reconocer (Código 3). Esto es muy útil ya que, en este proyecto, las frases que se van a pronunciar son acotadas. Con lo cual, se le proporciona una lista con todos los posibles comandos que se pueden utilizar en el juego. De esta forma se asegura que el reconocimiento va a ser satisfactorio casi el 100% de las veces.

```
commands = ["peón 1","torre 1","caballo 1","alfil 1","reina 1","rey
1","peón 2","torre 2","caballo 2","alfil 2","peón 3","peón 4","peón
5","peón 6","peón 7","peón 8","a1","a2","a3","a4","a5","a6",
"a7","a8","b1","b2","b3","b4","b5","b6","b7","b8","c1","c2","c3","c4",
"c5","c6","c7","c8","d1","d2","d3","d4","d5","d6","d7","d8","e1","e2",
"e3","e4","e5","e6","e7","e8","f1","f2","f3","f4","f5","f6","f7","f8",
"g1","g2","g3","g4","g5","g6","g7","g8","h1","h2","h3","h4","h5","h6",
"h7","h8"]
```

Código 3

7.2.3. GPIO

Manejar los pines GPIO de la Raspberry Pi es una tarea sencilla gracias a la librería RPi.GPIO (Python Package Index: RPi.GPIO, s.f.).

Esta librería permite manejar de forma sencilla los puertos de la Raspberry. A continuación, se explicarán todos los métodos utilizados en el proyecto.

Primero se importa la librería:

```
import RPi.GPIO as GPIO
```

Una vez importada se dispone de multitud de funciones para controlar los pines. Al inicio, hay que hacer un *setup* de los pines que se van a utilizar, para se declaran unas variables con el número de pin, para así después, poder llamar a las entradas o salidas por un nombre más legible (Código 4).

```
motor1A = 5
motor1B = 6
motor2A = 13
motor2B = 19
magnete = 26
magnet1 = 16
magnet2 = 12
ledr = 20
ledg = 21
button = 24
push1 =23
push2 =18
```

Código 4

Ahora se hace el *setup* de los pines (Código 5):

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(motor1A, GPIO.OUT)
GPIO.setup(motor2A, GPIO.OUT)
GPIO.setup(motor1B, GPIO.OUT)
GPIO.setup(motor2B, GPIO.OUT)
GPIO.setup(magnete, GPIO.OUT)
GPIO.setup(magnet1, GPIO.OUT)
GPIO.setup(magnet2, GPIO.OUT)
GPIO.setup(ledr, GPIO.OUT)
GPIO.setup(ledg, GPIO.OUT)
GPIO.setup(button, GPIO.IN, pull_up_down = GPIO.PUD_UP)
GPIO.setup(push1, GPIO.IN, pull_up_down = GPIO.PUD_UP)
GPIO.setup(push2, GPIO.IN, pull_up_down = GPIO.PUD_UP)
```

Código 5

Se puede apreciar que, la primera llamada que se realiza es la siguiente:

GPIO.setmode(GPIO.BCM)

Esta función indica el modo en que se numeran los pines, en este caso, el modo BCM se toman como referencia el número de pines del sistema. En el modo BOARD se refiere a los números físicos de la placa (Figura 33).

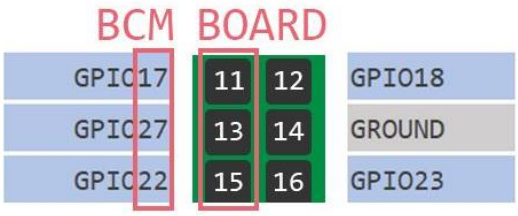

Figura 33: Numeración de los pines GPIO

Si se necesita un pin de salida se configura como *GPIO.OUT* y si se necesita una entrada se configura como *GPIO.IN*. En esta última configuración se pueden añadir más parámetros que son muy útiles, en este caso se añade *pull_up_down = GPIO.PUD_DOWN*. Esto activa una resistencia *pull-down* en la entrada seleccionada. Servirá para conectar el pulsador a dicha entrada.

Una vez configurado sólo queda leer de las entradas y activar o desactivar salidas. Para ello se usan las siguientes funciones:

GPIO.output(motor1A, salida)

GPIO.input(pulsador)

En la primera función se usan dos parámetros, el primero es el pin sobre el que va a actuar y el segundo el valor lógico a poner, '0' o '1'.

La segunda función devuelve el valor del pin del parámetro, '0' o '1'.

También se dispone de la siguiente función:

GPIO.wait_for_edge(pin, GPIO.FALLING)

Esta función ofrece la posibilidad de esperar a que la entrada cambie de voltaje. En el caso del ejemplo, se activa cuando habiendo un nivel alto, se produce una caída a nivel bajo. Se dispone de varias variables:

GPIO.FALLING

GPIO.RISING

GPIO.BOTH

Cada una de estas variables representa una o bien una caída en el nivel lógico, cuando pasa de nivel alto a nivel bajo, o una subida, cuando pasa de nivel bajo a nivel alto y la tercera cuando se producen los dos cambios, es decir, se recibe un pulso.

Una de las entradas del pulsador se conectará al pin de entrada y la otra, dependiendo de si se quiere detectar una bajada o una subida de voltaje, se conectará a GND o a una patilla de alimentación de 3,3V. A su vez, dependiendo de donde se conecte habrán de activarse la resistencia *pull-up* o *pull-down*.

Como se puede ver, trabajar con las entradas y salidas de la Raspberry Pi es muy sencillo gracias a esta librería, pues básicamente con dos funciones las posibilidades son infinitas.

7.2.4. Movimientos

Por último, los movimientos, que en esencia, es una de las partes más importantes del proyecto.

Como se ha visto en apartados anteriores, para poner a funcionar los actuadores únicamente se hace uso de una función, donde se activa o desactiva el pin de los GPIO correspondiente. Con esto y con la función *time.sleep()* se realizará la totalidad de los movimientos, calculando previamente una serie de parámetros para saber qué pines activar o desactivar y durante cuánto tiempo esperar.

Se comienza definiendo tres funciones básicas (Código 6) que servirán de interfaz para controlar los motores.

```
def motorX(dir):
    if dir == 0:
        GPIO.output(motor1A, 1)
        GPIO.output(motor1B, 0)
    if dir == 1:
        GPIO.output(motor1A, 0)
        GPIO.output(motor1B, 1)

def motorY(dir):
    if dir == 0:
        GPIO.output(motor2A, 1)
        GPIO.output(motor2B, 0)
    if dir == 1:
        GPIO.output(motor2A, 0)
        GPIO.output(motor2B, 1)

def stopMotor(motor):
    if motor == 1:
        GPIO.output(motor1A, 0)
        GPIO.output(motor1B, 0)
    if motor == 2:
        GPIO.output(motor2A, 0)
        GPIO.output(motor2B, 0)
```

Código 6

Las dos primeras funciones sirven para mover o bien los motores del eje X o los del eje Y. Como parámetro se le introduce la dirección en la que se quiere mover. La última función sirve para detener los motores del eje elegido.

Se continúa definiendo una función que sirve de intermedio entre las funciones básicas y las funciones más elaboradas (Código 7).

```
def moveMotor (x,y,dirX,dirY,stepX,stepY):
    if x and y and stepX==stepY:
        motorX(dirX)
        motorY(dirY)
        time.sleep(stepX)
        stopMotor(1)
        stopMotor(2)
    else:
        if x and y:
            motorX(dirX)
            time.sleep(stepX)
            stopMotor(1)
            motorY(dirY)
            time.sleep(stepY)
            stopMotor(2)
        else:
            if x:
                motorX(dirX)
                time.sleep(stepX)
                stopMotor(1)
            if y:
                motorY(dirY)
                time.sleep(stepY)
                stopMotor(2)
```

Código 7

Esta función tiene seis parámetros obligatorios. Los dos primeros son dos parámetros booleanos donde se selecciona qué eje se quiere mover, el X, el Y o ambos. Los dos siguientes la dirección en la que se quiere mover cada eje. Por último, los dos siguientes se les dice el número de pasos a dar.

Los pasos (Figura 34) equivalen al tiempo necesario para moverse el número de medias casillas que necesitamos.

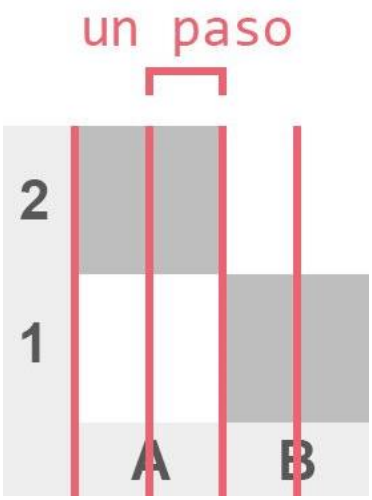

Figura 34: Esquema de los pasos

Se ha calculado el tiempo necesario para moverse desde 1 paso hasta el máximo de 18 pasos ya que debido al uso de motores DC en vez de paso a paso y las holguras producidas en los desplazamientos entre los engranajes, hacían que al tener un único tiempo por paso, según aumentaba el número de pasos, disminuía la distancia que en teoría debería recorrer el cabezal, con lo que calculando el tiempo exacto desde la posición 0 hasta cada una de las posiciones hasta la 18, se consigue una exactitud bastante precisa en el posicionamiento del cabezal. Adicionalmente, para evitar aún más esta problemática, por cada movimiento completo que se hace, se vuelve siempre a la posición origen.

Dentro de la función lo único que se hace es, teniendo en cuenta los parámetros de entrada, encender los motores, esperar el tiempo y apagar los motores.

Por último, se ha desarrollado la función de desplazar el cabezal hasta una posición indicada (Código 8) en un formato de coordenadas (x,y).

```
def placeMagnet (pos):
    x = False
    y = False
    dirX = 0
    dirY = 0
    stepX = 0
    stepY = 0
    global mPos

    #calculate the parameters to move the magnet
    restaX = pos[0] - mPos[0]
    restaY = pos[1] - mPos[1]

    if restaX != 0:
        x = True
    if restaY != 0:
        y = True
```

```
    if restaX < 0:
        dirX = 1
    if restaY < 0:
        dirY = 1

    stepX = times[str(2 * abs(restaX))]
    stepY = times[str(2 * abs(restaY))]

    moveMotor(x,y,dirX,dirY,stepX,stepY)

    mPos = pos
```

Código 8

En esta función se calcula qué motores han de moverse y cuantos pasos ha de efectuar cada uno. Como se puede ver, esto se realiza calculando la diferencia de la posición nueva con la posición actual del cabezal. Si esta es negativa, hacia un lado, si es positiva, hacia el otro. Si la diferencia es 0 en alguna de las coordenadas, el motor correspondiente a ese eje no se mueve. Para calcular el número de pasos únicamente basta con hacer el valor absoluto de la diferencia y multiplicarlo por 2 debido a que cada paso es la mitad de una casilla.

Hay dos funciones que realizan una serie de movimientos especiales. Estas dos funciones se encargan de mover el caballo (Código 9) y de mover las fichas al borde del tablero cuando han sido eliminadas (Código 10).

```
def placeMagnetKnight(pos):
    global mPos
    x = False
    y = False
    dirX = 0
    dirY = 0
    stepX = 0
    stepY = 0

    restaX = pos[0] - mPos[0]
    restaY = pos[1] - mPos[1]

    if restaX > 1:
        if restaX < 0:
            if restaY < 0:
                moveMotor(0,1,0,1,0,times["1"])
                moveMotor(1,0,1,0,times["4"],0)
                moveMotor(0,1,0,1,0,times["1"])
```

Código 9

El caballo, al ser una pieza que tiene la posibilidad de saltar y pasar por encima a otras piezas, supone un problema a la hora de moverlo. La solución implementada para este caso ha sido la de mover el caballo por el borde de las casillas, pudiendo así pasar entremedias de las fichas (Figura 36) sin llevarse por delante ninguna. En la función se calcula la trayectoria de la ficha dependiendo de su nueva posición (Figura 35).

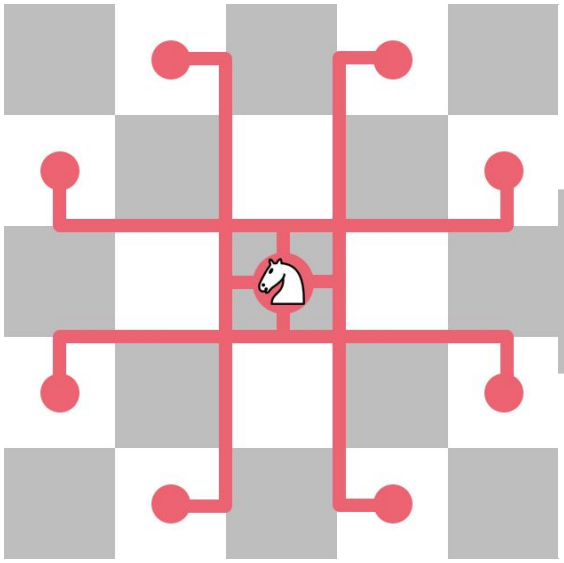

Figura 35: Trayectoria del caballo detallada

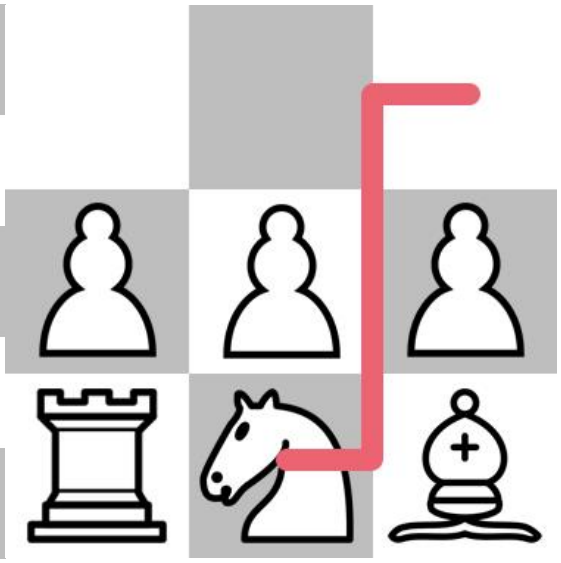

Figura 36: Trayectoria del caballo

Esta trayectoria, si bien no siempre queda exenta de colisiones con otras piezas debido a la precisión del mecanismo de movimiento, resulta un método bastante efectivo, pudiendo desplazar el caballo sin dificultad por el tablero mientras que las demás piezas no se ven afectadas.

Para el movimiento de las fichas eliminada se ha optado por una estrategia similar a la ficha del caballo. La figura se moverá hasta la línea borde más próxima inferior y desde allí será llevada hasta el borde del tablero sin tocar, en la medida de lo posible, a ninguna otra ficha. Al igual que el caballo, la trayectoria no es perfecta debido a la precisión, pero resulta efectiva para eliminar las fichas del tablero como se ve en la siguiente figura.

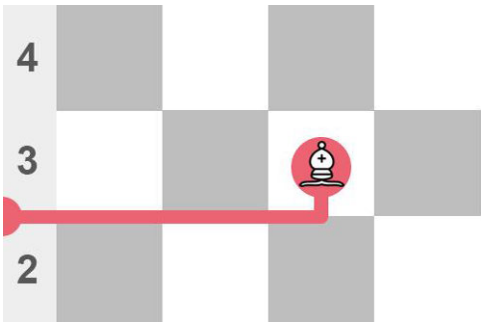

Figura 37: Trayectoria de la eliminación

```
def moveToKill(pos):
    placeMagnet(pos)
    eMagnet(1)
    moveMotor(0,1,0,1,0,times["4"])
    moveMotor(1,0,1,0,times[str(2*pos[0])],0)
    eMagnet(0)
    calibrate()
```

Código 10

Como se ha mencionado anteriormente, por cada movimiento completo el cabezal vuelve al a posición inicial. Para ello se ha dispuesto dos pulsadores de final de carrera en los límites del recorrido de los ejes (Figura 38). Para este movimiento se cuenta con la siguiente función.

```
def calibrate ():
    global mPos

    time.sleep(0.5)
    GPIO.output(motor1A,0)
    GPIO.output(motor1B,1)
    GPIO.wait_for_edge(push1,GPIO.FALLING)
    GPIO.output(motor1A,0)
    GPIO.output(motor1B,0)

    GPIO.output(motor2A,0)
    GPIO.output(motor2B,1)
    GPIO.wait_for_edge(push2,GPIO.FALLING)
    GPIO.output(motor2A,0)
    GPIO.output(motor2B,0)
    time.sleep(0.5)

    mPos=(0,0)
```

Código 11

En ella, se encienden los motores de un eje y se espera hasta que topen con los pulsadores. Se hace lo mismo con el otro eje. De esta forma el cabezal siempre estará situado en la misma posición al inicio del movimiento.

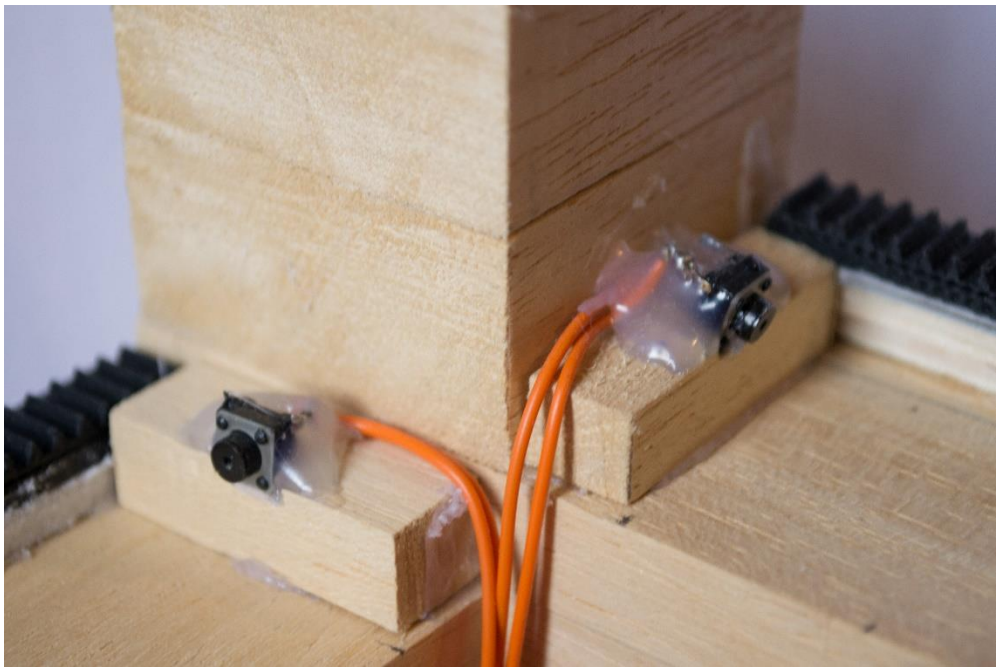

Figura 38: Pulsadores fin de recorrido

Por último, se cuenta con una función para accionar el imán del cabezal (Código 12). Para ello mediante un parámetro se indica la dirección a la que mover el imán, arriba o abajo. Esta función se usa una vez realizado el primer movimiento, cuando el cabezal se posiciona sobre la pieza. A continuación, se mueve el imán hacia arriba. Una vez realizado el trazo del movimiento de la ficha, se vuelve a activar hacia abajo.

```
def eMagnet(dir):
    if dir == 1:
        GPIO.output(magnet1, 0)
        GPIO.output(magnet2, 1)
        GPIO.output(magnete, 1)
        time.sleep(0.5)
        GPIO.output(magnete, 0)
        time.sleep(0.5) #Wait a little bit to fix the piece well to
the magnet
    if dir == 0:
        GPIO.output(magnet1, 1)
        GPIO.output(magnet2, 0)
        GPIO.output(magnete, 1)
        time.sleep(0.5)
        GPIO.output(magnete, 0)
```

Código 12

Capítulo 8. Pruebas

En este capítulo se va a tratar de explicar las pruebas realizadas, tanto previas al desarrollo de las partes que componen el ajedrez como posteriores a la finalización. Así mismo, como se verá en el siguiente capítulo sobre la planificación, la metodología obliga a ir corrigiendo progresivamente defectos y descubriendo mejoras con lo que durante el desarrollo también han tenido que efectuarse pruebas, es este caso, denominadas, pruebas individuales.

Las primeras pruebas realizadas son las relacionadas con el software. Las diferentes librerías usadas requieren un estudio previo, así como una batería de pruebas para comprobar su funcionamiento.

Tras obtener cualquier placa de desarrollo como una Raspberry Pi o un Arduino, una de las pruebas iniciales es el famoso *Blinking Led* (Código 13). Esta prueba trata de hacer parpadear un LED conectado a una de las salidas de la placa. Por supuesto, esto ayuda a empezar desde una base muy sencilla para comprender el funcionamiento de la librería de control de los GPIO.

```
import RPi.GPIO as GPIO
import time

#setup
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)

#run
def blink():
        print "Inicio..."
        iteracion = 0
        while iteracion < 10:
                GPIO.output(17, True)  #Se enciende led
                time.sleep(1)          #duerme 1 seg
                GPIO.output(17, False) #Se apaga led
                time.sleep(1)
                iteracion += 1
        print "Fin."
        GPIO.cleanup()

blink()
```

Código 13

Además de la librería, se practica la generación de funciones en Python, que será esencial para el desarrollo del proyecto.

Como se puede ver, lo que hace es iterar diez veces en un bucle mientras que se enciende el LED, se espera un segundo, se apaga el mismo LED, se espera otro segundo y se vuelve al principio. Con esto se tiene el control sobre cualquier pin de salida de la Raspberry Pi.

Para controlar el módulo de los motores se genera otro código de prueba (Código 14).

```
#=======================================
#Ejemplo de control del motor del ajedrez
#=======================================

import RPi.GPIO as GPIO
import time

motor1A = 13
motor1B = 19

GPIO.setmode(GPIO.BCM)
GPIO.setup(motor1A, GPIO.OUT)
GPIO.setup(motor1B, GPIO.OUT)

print "setup completo"

print "Motor: adelante"
GPIO.output(motor1A, True)
GPIO.output(motor1B, False)
time.sleep(5)

print "Motor: atras"
GPIO.output(motor1A, False)
GPIO.output(motor1B, True)
time.sleep(5)

print "Motor: parado"
GPIO.output(motor1A, False)
GPIO.output(motor1B, False)
time.sleep(1)

print "Finalizando"
GPIO.cleanup()
```

Código 14

En este código se encienden los motores hacia ambos lados y seguido se paran. Así se obtienen los pines que hay que encender y apagar para moverlo hacia un lado u otro, comprobando que se han realizado las conexiones correctamente en el circuito.

Para poner a prueba los pulsadores y comprobar el funcionamiento del modo entrada de los pines de la placa, se ha usado el siguiente programa (Código 15):

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(24, GPIO.IN, pull_up_down = GPIO.PUD_UP)

print("dale al boton")

GPIO.wait_for_edge(24,GPIO.FALLING)
print("pulsado")

time.sleep(1)
print("fin")

GPIO.cleanup()
```

Código 15

Aquí, se conecta el pulsador al pin elegido y se espera hasta obtener una caída de voltaje en el pin de entrada, esto significa que se ha pulsado el botón. Análogamente, podría configurarse la entrada con la resistencia *pull-down*, conectar el pulsador a 3.3V y esperar una subida de voltaje en el pin.

Los tiempos de espera del motor entre que se enciende y se apaga, dependiendo del número de pasos, se extrajo probando con el siguiente código (Código 16).

```
import RPi.GPIO as GPIO
import time

motor1A =6
motor1B =5

STEP = 0.1934

GPIO.setmode(GPIO.BCM)
GPIO.setup(motor1A, GPIO.OUT)
GPIO.setup(motor1B, GPIO.OUT)

while True:

    inp = input("parametro: ")

    if inp=="1":
        GPIO.output(motor1A,0)
        GPIO.output(motor1B,1)
        time.sleep(STEP)
        GPIO.output(motor1A,0)
        GPIO.output(motor1B,0)
```

```
    if inp=="0":
        break

GPIO.cleanup()
```

Código 16

Lo que se hace es variar el valor de la variable STEP y a continuación se ejecuta el programa indicándole que avance, es decir, enciende el motor, espera el tiempo para recorrer el paso o los pasos y lo para. Así, con el imán del cabezal en posición y con una pieza del ajedrez, se va midiendo para cada número de pasos el tiempo necesario que hay que esperar. Es un trabajo tedioso y poco elegante, pero, debido a las holguras y a los tiempos de arranque y parada del motor, es necesario realizar este paso para conseguir un movimiento lo más preciso posible.

En el caso de la librería de reconocimiento de voz, se ha de comprobar con un simple ejemplo que lo que se recibe en el micrófono, se traduce en texto satisfactoriamente. Cómo se comentaba en capítulos anteriores, inicialmente se usó la API de Microsoft para efectuar el reconocimiento. Posteriormente, debido a la poca precisión del texto devuelto se optó por usar la API de Google. Para probar el funcionamiento se han realizado unos ejemplos simples que se encuentran en la referencia de la librería. Éste se puede encontrar en el apartado *9.2.2 Reconocimiento de voz* (Código 2).

Simplemente se trata de ejecutar dicho código y ver qué devuelve. Para cambiar entre una API u otra, basta con cambiar la llamada, pues la librería cuenta con diferentes llamadas para cada servicio de reconocimiento. Probando con la de Microsoft (Código 17), se obtuvieron unos resultados muy pobres y poco precisos.

```
print("Microsoft thinks you said: " + r.recognize_bing(audio,
key=BING_KEY, language="es-ES"))
```

Código 17

Lo que llevó a descartarla por completo era la imposibilidad de reconocer la palabra “peón”. Al tratarse de comandos de voz muy específicos, el contexto no contenía las condiciones suficientes para que dicha palabra fuera reconocida pues, por ejemplo, si recibe: ‘*El peón de obra*’, la frase era devuelta correctamente. Por el contrario, al decir: ‘*Peón C4*’, el servicio nunca reconocía la palabra. También, inducía a error las coordenadas de las fichas, confundiendo las letras. Por ejemplo, al decir: ‘*Torre B2*’, el servicio podía devolver: ‘*Torre D dos*’ o ‘*Torre B dos*’ ya que fonéticamente las dos letras son similares, en parte debido a la calidad de la grabación que puede convertir el sonido suave de la letra *B* en un pico fuerte como contiene la letra *D*.

Con todo esto se decidió optar por la opción de Google. Su servicio ofrecía la posibilidad de incluir una lista de posibles resultados. Así pues, pasando la lista de las posibles coordenadas de cada ficha, el resultado es muy satisfactorio, obteniendo casi en la totalidad de casos el comando hablado en texto. En ciertas ocasiones, tiene predilección por devolver una frase con sentido en vez de un comando de ajedrez. Por ejemplo, sigue teniendo el mismo problema con las letras *B* y *D*, en muchos casos devuelve la palabra '*de*' en vez de la letra. Con lo que, si le envía '*Peón 3 B2*' devuelve: '*Peón 3 de 2*'.

Con más tiempo, se podrían hacer muchas más pruebas y codificar una función que analice el texto devuelto y lo convierta a un comando entendible para el programa, ya sea, por ejemplo, transformando la palabra '*de*' en la letra *D* o *B* según la ficha pueda o no moverse a esa fila.

Con esto concluyen las pruebas iniciales. En el transcurso del desarrollo surgirá la necesidad de adaptar estos programas para probar por ejemplo las conexiones de los diferentes módulos con la Raspberry Pi, el buen emplazamiento de los motores, así como pulsadores, indicadores luminosos, etc.

Por lo tanto, durante todo el proyecto han sido necesarias pruebas intermedias en las que según va avanzando la construcción del software se han ido probando las funcionalidades desarrolladas inmediatamente y así mismo, retocar o incluir mejoras en las mismas.

Para estas pruebas se desactivaron ciertas partes del código para facilitar su ejecución. Se ha sustituido el sistema de reconocimiento de voz por un sistema de entrada de texto, así no es necesario lanzar todo el proceso de detección de voz, envío a servidores y posterior traducción a texto, escribiendo por consola el comando es más sencillo realizar pruebas.

Las pruebas intermedias tratan básicamente dos módulos, el módulo del motor de ajedrez y el módulo de los movimientos.

El módulo del motor de ajedrez ha necesitado de innumerables pruebas. Para ello se ha activado la opción de imprimir por pantalla el tablero, de esta forma se puede ver "gráficamente" las posiciones de las fichas y si efectivamente se han actualizado o no.

Las diferentes conversiones entre coordenadas de diferente aspecto, por ejemplo, de la notación de ajedrez *a1c3* a coordenadas numéricas a1 = (1,1) c3 = (3,3) y adaptar estas coordenadas a la notación que usa el motor de ajedrez ha requerido bastante tiempo para acomodar todas las interfaces y unir el circuito *Comando de voz – Ajedrez – IA*.

El módulo de movimiento ha requerido pruebas para su calibración, como se explicó anteriormente, debido a la naturaleza de los motores usados y del mecanismo interno, la precisión a la hora de realizar movimientos no lo suficientemente grande como para crear trayectorias de las longitudes que se pretenden. Para lo cual se creó la lista de tiempo por pasos en la que se guarda el tiempo necesario que tiene que estar el motor encendido para recorrer x pasos.

Para esta prueba se situó el imán en posición de captura junto con una pieza del juego fija. Se ha desarrollado un programa en el que, introduciendo las coordenadas de una posición del tablero, de (0,0) hasta (18,18) el cabezal mueve la ficha hasta esta posición. Con esto, se puede medir y ajustar de manera aproximada el tiempo que tarda en moverse la ficha desde la posición 0 hasta cualquiera de las otras. En principio la calibración de los valores se ha realizado siempre sobre el mismo eje, al ser un tablero cuadrado se presupone que las distancias son las mismas en ambos lados. Una vez teniendo todos los tiempos, se ha probado a introducir coordenadas y a comprobar si efectivamente la ficha se sitúa donde se le indica.

De las pruebas ejecutadas se obtuvieron los siguientes resultados, agrupados en un diccionario. El tiempo está medido en segundos:

```
times =
{"0":0.0,"1":0.18,"2":0.36,"3":0.54,"4":0.84,"5":1.00,"6":1.22,"
7":1.40,"8":1.64,"9":1.81,"10":2.04,"11":2.25,"12":2.46,"13":2.6
8,"14":2.88,"15":3.11,"16":3.30,"17":3.46,"18":3.68}
```

Con este procedimiento se ha conseguido una precisión bastante decente, si bien no se sitúa exactamente en su posición en ciertas longitudes. La distancia de error fluctúa entorno a los 3mm.

Capítulo 9. Costes y planificación

A continuación, se muestra una tabla que detalla el presupuesto aproximado para el análisis y construcción del prototipo totalmente funcional.

Los costes son aproximados, pues el coste de cada producto varía en función del proveedor escogido. También, los apartados de materiales varios, herramientas varias y PC están calculados con un precio medio del coste de productos básicos.

En cuanto a los salarios, son datos publicados en el Boletín Oficial del Estado para el año 2017 (BOE-A-2017-542) donde aparecen sueldos anuales de Programador y Analista, se han calculado a jornada completa durante 6 meses.

Producto	Proveedor	Coste	Cantidad	Importe	Totales
Raspberry Pi 3 Model B	Element14	31,69 €	1	31,69 €	
Cargador 5V 3A	Amazon	10,99 €	1	10,99 €	
Tarjeta Memoria MicroSD 16GB	Amazon	7,50 €	1	7,50 €	
Micrófono USB	Aliexpress	1,78 €	1	1,78 €	
Expansión GPIO	Aliexpress	3,61 €	1	3,61 €	
Motor DC 6V 100RPM	Aliexpress	3,45 €	5	17,25 €	
Modulo Puente-H L9110	DealExtreme	2,56 €	1	2,56 €	
Puente-H L298N	Electrónica Embajadores	4,59 €	1	4,59 €	Electrónica
Conector MicroUSB hembra	DealExtreme	1,36 €	1	1,36 €	
Placa de prototipo	DealExtreme	4,08 €	1	4,08 €	
Pulsador pequeño	Electrónica Embajadores	0,50 €	2	1,00 €	
Pulsador	Electrónica Embajadores	0,82 €	1	0,82 €	
Diodos LED	Electrónica Embajadores	0,15 €	2	0,30 €	
Resistencias 10 unidades	Electrónica Embajadores	0,25 €	1	0,25 €	
Condensador Electrolítico 100nF	RS	0,04 €	2	0,08 €	
Diodo 1N4007	Electrónica Embajadores	0,09 €	4	0,36 €	
Cable Unipolar Ø 0,07mm 10m	Electrónica Embajadores	1,99 €	1	1,99 €	
Soldadura de estaño 1mm 100g	Electrónica Embajadores	5,29 €	1	5,29 €	Total: 95,50€
Tablero de madera 2,5mm 600x400mm	MaxiChina	2,25 €	2	4,50 €	
Listón de madera 2250x40x10mm	Bricomart	2,10 €	1	2,10 €	
Listón de madera 2250x40x20mm	Bricomart	3,39 €	1	3,39 €	Materiales
Varilla de aluminio Ø 6mm	Bricomart	1,00 €	1	1,00 €	
Rodamientos lineales Ø 14-6mm	Amazon	4,68 €	1	4,68 €	
Materiales varios (Tornillos, clavos, cola...)	Cualquier proveedor	5,00 €	1	5,00 €	Total: 20,67€
Soldador 26W	Cualquier proveedor	35,00 €	1	35,00 €	
Pistola para silicona termofusible	Cualquier proveedor	12,00 €	1	12,00 €	
Multímetro Digital	Cualquier proveedor	12,00 €	1	12,00 €	Herramientas
PC (Cualquier pc con acceso a red)	Cualquier proveedor	400,00 €	1	400,00 €	
Ratón USB (opcional)	Cualquier proveedor	5,00 €	1	5,00 €	

Teclado USB (opcional)	Cualquier proveedor	5,00 €	1	5,00 €	
Herramientas varias (Utensilios de taller...)	Cualquier proveedor	10,00 €	1	10,00 €	Total: 479,00€
Salario Programador 6 meses	BOE - 2017	8.463,80 €	1	8.463,80 €	Salarios
Salario Analista 6 meses	BOE - 2017	9.415,50 €	1	9.415,50 €	Total: 17.879,30€
					Total 18.474,47 €

Presupuesto

En cuanto a la planificación, inicialmente el proyecto iba a tener una duración de 6 meses. Se puede ver en el siguiente diagrama de Gantt (Figura 39).

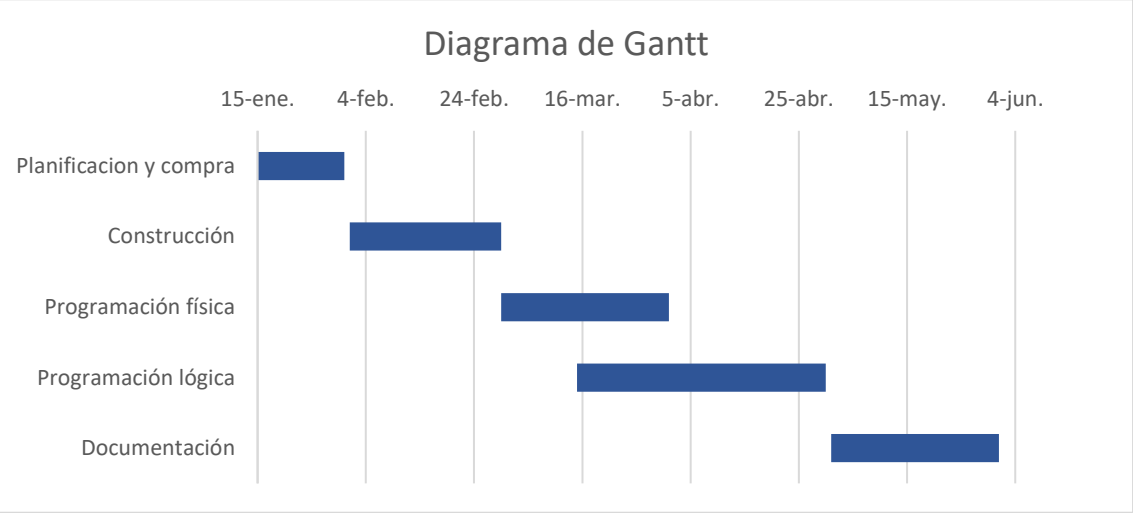

Figura 39: Diagrama de Gantt planificado

La realidad es que el proyecto se ha extendido tres meses más. El porqué de este retraso es debido principalmente a la etapa de compra. Los proveedores se retrasaron en sus pedidos, enviaron productos defectuosos e incluso el transportista perdió algunos. Obviamente no solo es debido a esto, la construcción del prototipo fue totalmente experimental y se iban descubriendo fallos a medida que avanzaba, solucionar estos fallos ha extendido el tiempo previsto y modificado la planificación.

En el siguiente gráfico se puede ver más o menos la realidad de las distintas fases.

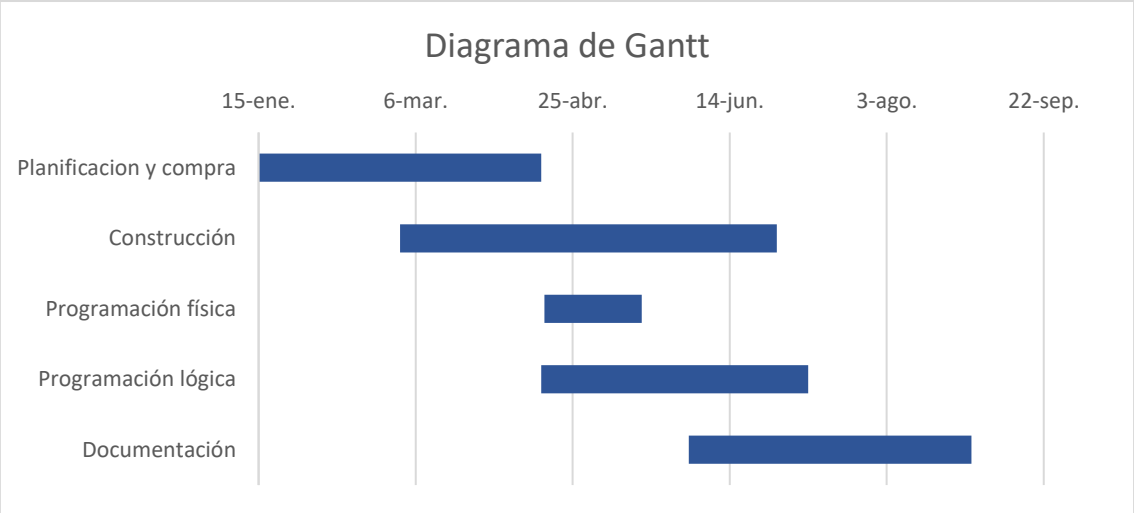

Figura 40: Diagrama de Gantt real

Se puede ver que, no solamente las fases son más largas y no siguen el mismo orden que lo planificado, si no que la línea temporal se ha expandido, evidenciando que el proyecto se ha alargado más de lo previsto.

La metodología usada para este proyecto, aunque en principio es clásica, la realidad es que va evolucionando a medida que se avanzaba, pues según surgían problemas, el proyecto volvía a una fase anterior hasta que todos los requisitos planteados al principio, incluso modificados sobre la marcha, han sido cumplidos.

Esto pone en evidencia la poca efectividad de la metodología clásica aplicada al mundo de la informática y explica el auge de nuevas metodologías como por ejemplo las ágiles, donde requisitos y prototipos se van mezclando en el tiempo, volviendo hacia atrás o hacia delante hasta tener un desarrollo completo de un producto, incluso entonces, hasta el ciclo de vida completo del mismo se siguen produciendo actualizaciones para mantener o mejorar partes del producto. La metodología clásica funciona bien en otras áreas donde los requisitos están claros y todo el proceso está completamente cerrado. En el mundo de la informática, el cliente y el desarrollador están en permanente contacto, comunicando posibles necesidades, mejoras y abordando multitud de problemas que surgen en un desarrollo. Por ello, las metodologías ágiles ofrecen esa libertad y permiten desligarse de plazos cerrados y procedimientos sin capacidad de reacción ante posibles incidencias o cambios

Capítulo 10. Aspectos éticos, sociales y ambientales

Desde el primer momento el proyecto se ha construido pensando en el enfoque educacional del mismo. El proyecto está enfocado para que sirva de referencia de futuras implementaciones, para fomentar el aprendizaje. Las personas que quieran podrían de forma sencilla acceder a todas y cada una de las partes del proyecto, tanto modelos en 3D como código interno. El hardware utilizado se ha intentado hacerlo modular, es decir, que la mayoría de la electrónica utilizada sea en forma de módulos que se pueden adquirir en tiendas con lo que la complejidad se reduce drásticamente y permitiría, que personas que no tienen soltura con la electrónica ni con componentes que desconocen, puedan empezar con una base sencilla sobre la que construir un conocimiento más sólido.

Referente al software, se ha utilizado software de terceros que no es software libre como es el reconocimiento de voz. Aunque este sea software privativo sí que facilita el acceso a una tecnología que, de tener que ser implementada por nosotros sería mucho más costoso, en este sentido, con el uso de librerías y comunicándose con la plataforma que facilita Google, el acceso a reconocimiento de voz se torna una tarea nada complicada.

Los materiales utilizados son amigables con el medio ambiente, la estructura es de madera y ciertas partes son de aluminio. No se han utilizado ningún tipo de sustancia nociva como disolventes o pinturas y no se han generado residuos peligrosos más allá de serrín. Los adhesivos utilizados son cola blanca y silicona termofusible, se ha descartado el uso de otro tipo de sustancias más nocivas como el pegamento de contacto.

Varios de los componentes son reutilizados de otros productos electrónicos antiguos. Las resistencias, los diodos y los condensadores han sido extraídos de antiguas placas de circuitos, los diodos por ejemplo pertenecen a un antiguo puente rectificador y los condensadores a una radio. Así aprovechamos componentes que son perfectamente usables de desechos electrónicos que de otra forma acabarían en la basura contaminando.

En definitiva, se trata de un proyecto pensado para servir de ayuda a gente que quiera aprender, dando todas las facilidades posibles a la hora de acceder a él y permitiendo que el conocimiento adquirido en la realización de este pueda servir de guía a otras personas interesadas. Conjugando materiales reciclados con otros de bajo coste, haciendo uso de la impresión 3D y de software libre aseguramos que, para todos aquellos interesados en el juego como en su construcción encuentren todas las facilidades posibles.

Capítulo 11. Conclusiones

Los resultados obtenidos son satisfactorios. Se ha conseguido llevar a cabo la totalidad del proyecto planteado inicialmente. Se ha logrado construir un ajedrez controlado por la voz perfectamente funcional, si bien con algún detalle a pulir y a mejorar en un futuro, el prototipo elaborado cumple todos los requisitos que se plantearon en un inicio.

Se ha conseguido un sistema de reconocimiento de voz, que, reconoce casi completamente todos los comandos de voz. Se fabricado un mecanismo de movimiento de piezas que después de muchas horas de prueba y error y calibración, encuentra y mueve con bastante acierto las fichas encima del tablero.

El objetivo está conseguido. En el transcurso del proyecto se han adquirido multitud de aptitudes, se ha probado la habilidad de resolver problemas durante el desarrollo, se han invertido muchas horas en investigación y otras tantas en el desarrollo. Se ha buscado la manera de resolver y mejorar cualquier defecto que pudiera sacar un mejor funcionamiento del prototipo.

En definitiva, todo lo aprendido y desarrollado en este proyecto podría servir de guía para todo aquel que busque un comienzo no demasiado complejo en el mundo de la electrónica, controladores y programación.

Capítulo 12. Líneas Futuras

Las mejoras y modificaciones que se pueden implementar en este proyecto son de diversa índole. Aunque se puede considerar como un producto terminado, hay muchas líneas donde se podrían hacer pequeños cambios para ofrecer funcionalidades nuevas o mejorar las existentes. Algunos de los aspectos que se podrían mejorar son:

- **Escucha permanente**.

Esta funcionalidad permitiría al ajedrez escuchar de forma permanente y, mediante un comando de voz especifico, por ejemplo “Ajedrez”, se empezaría la captura de audio para dar la orden del movimiento de la pieza. Este método sería mucho más directo, sin interfaces de usuario como pulsadores de por medio. Permitiría una fluidez conversacional mejorada que ayudaría a llevar a cabo una mejor partida.

- **Implementación de movimientos especiales**.

Una de las restricciones era que el ajedrez no realizará movimientos especiales. Se podría implementar la realización de estos movimientos. Obviamente estos movimientos son bastante más complejos que los que se realizan normalmente ya que implican mover dos piezas, incluso en el caso de la coronación habría que tener en cuenta si la reina sigue o no en el tablero para sustituir la pieza y tratar a la nueva con un nuevo nombre.

- **Calidad de materiales**.

Este proyecto, al contar con un presupuesto limitado y al tratarse de un prototipo, se ha tenido que prescindir de elementos que podrían convertirlo en un producto comercializable al 100%. Hay dos elementos básicos de mejora, la estructura y la electrónica. La estructura actual es de madera y metal, para mejorarla habría que emplear un diseño de plástico por inyección o en su defecto piezas impresas en 3D dejando un acabado mucho más estético. En la parte de los componentes, la placa de prototipo y los módulos podrían sustituirse por una placa impresa personalizada que contenga todos los componentes necesarios, así como conectores que pudieran permitir su ensamblaje junto con la Raspberry Pi como si de un *shield* se tratara, permitiendo reducir el espacio que ocupa el conjunto de la electrónica.

Referencias

Ajedrez - Online. (s.f.). Obtenido de https://ajedrez-online.es/reglas-ajedrez

Amazon Echo. (s.f.). Obtenido de Amazon: https://www.amazon.com/Amazon-Echo-Bluetooth-Speaker-with-Alexa-Black/dp/B00X4WHP5E/

Asus. (s.f.). *Asus Tinker Board*. Obtenido de https://www.asus.com/us/Single-Board-Computer/Tinker-Board/overview/

Chiark Greenend. (s.f.). Obtenido de https://www.chiark.greenend.org.uk/~sgtatham/putty/

Code Google Archive. (s.f.). Obtenido de https://code.google.com/archive/p/win-sshfs/

Docs Python WIndows. (s.f.). Obtenido de Using Python on Windows: https://docs.python.org/3/using/windows.html

FAQ: Power Req. Raspberry Pi. (s.f.). Obtenido de Raspberry Pi: https://www.raspberrypi.org/help/faqs/#powerReqs

Federación Internacional De Ajedrez. (s.f.). *Fide Laws of Chess.* Obtenido de World Chess Federation: https://www.fide.com/fide/handbook.html?id=208&view=article

Google. (s.f.). *Google Cloud Platform*. Obtenido de Google Cloud: https://cloud.google.com/

Google Home. (s.f.). Obtenido de Google: https://support.google.com/googlehome/answer/7029281?hl=en&ref_topic=7029677

ic2ic datasheet. (s.f.). Obtenido de http://www.ic2ic.com/search.jsp?sSearchWord=HG7881

infineon datasheet. (s.f.). Obtenido de https://www.infineon.com/dgdl/irf520n.pdf?fileId=5546d462533600a4015355e339ee1983

jjnm5, J. (s.f.). *Thingiverse*. Obtenido de Thingiverse: https://www.thingiverse.com/thing:633265

MICROSOFT ESPAÑA. (12 de 12 de 17). Obtenido de www.microsoft.com/es

Orange Pi. (s.f.). *Orange Pi* . Obtenido de Orange Pi - Orange Pi Plus 2: http://www.orangepi.org/orangepiplus2/

Ortega, J. (s.f.). *Breve historia del ajedrez*. Obtenido de Universo Ajedrez: http://www.universodelajedrez.com/2012/08/breve-historia-del-ajedrez/

PINE A64. (s.f.). *PINE A64*. Obtenido de PINE A64: https://www.pine64.org/?page_id=1194

Python Package Index : python-chess. (s.f.). Obtenido de Python Package Index: https://pypi.python.org/pypi/python-chess

Python Package Index: RPi.GPIO. (s.f.). Obtenido de Python Package Index: https://pypi.python.org/pypi/RPi.GPIO

Python Package Index: SpeechRecognition. (s.f.). Obtenido de Python Package Index: https://pypi.python.org/pypi/SpeechRecognition

Raspberry Pi : About. (s.f.). Obtenido de Raspberry Pi: https://www.raspberrypi.org/about/

Raspbian. (s.f.). Obtenido de Raspbian: https://www.raspbian.org/

SketchUp. (s.f.). Obtenido de https://www.sketchup.com/es

StMicroelectronics. (s.f.). Obtenido de http://www.st.com/resource/en/datasheet/l298.pdf

Sublime Text. (s.f.). Obtenido de https://www.sublimetext.com/

Wikipedia. (s.f.). *Wikipedia*. Obtenido de https://es.wikipedia.org/wiki/Ajedrez

Romano F. (2005). Learning Python. Packt Publising.