Escola Tècnica Superior d'Enginyeries
Industrial i Aeronàutica de Terrassa
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Titulación:

INGENIERÍA INDUSTRIAL

Alumno:

RAFAEL LARROSA ESPEJO

Titulo PFC:

DISEÑO DE LA ADAPTACIÓN DE UN JUEGO DE AJEDREZ PARA PERSONAS CON MOVILIDAD REDUCIDA

Director del PFC:

DAVID GONZÁLEZ DIEZ

Convocatoria de entrega del PFC:

SEPTIEMBRE DE 2014

Contenido de este volumen: **-MEMORIA-**

ÍNDICE

ÍNDICE DE IMÁGENES

ÍNDICE DE TABLAS

1. OBJETO Y JUSTIFICACIÓN

1.1. Objeto

El proyecto consiste en el diseño para adaptar un juego de ajedrez común a personas con movilidad reducida o algún tipo de discapacidad que les impida jugar con normalidad a partir de la automatización del juego mediante placas de programación de Arduino.

1.2. Justificación

La discapacidad debido a movilidad insuficiente cada día es más común en el mundo. Cerca de un 8,5% de la población en España (unos 4 millones de personas) y un 12% de la población mundial (unos 900 millones de personas) es discapacitada. De este porcentaje, cuatro de cada diez personas mayores de seis años tienen deficiencia en los huesos y articulaciones.

Tabla 1 - Personas mayores de seis años con discapacidad según el grupo

Fuente: Instituto Nacional de Estadística (2008)

	Ambos sexos		Varones		Mujeres	
	Nº de personas	Tasa por 1.000	Nº de personas	Tasa por 1.000	Nº de personas	Tasa por 1.000
TOTAL	3.787,4	89,70	1.510,9	72,58	2.276,5	106,35
Visión	979,0	23,19	371,3	17,84	607,7	28,39
Audición	1.064,1	25,20	455,7	21,88	608,5	28,43
Comunicación	734,2	17,39	336,6	16,17	397,5	18,57
Aprendizaje realización tareas	630,0	14,92	264,5	12,70	365,5	17,07
Movilidad	2.535,4	60,05	881,5	42,34	1.653,9	77,27
Autocuidado	1.824,5	43,21	645,0	30,98	1.179,5	55,10
Vida doméstica	2.079,2	49,24	605,8	29,10	1.473,4	68,83
Relaciones personales	621,2	14,71	291,7	14,01	329,5	15,39

* Numero de personas en miles y tasa por 1000 habitantes.

Es decir, en España hay 2,5 millones de personas con discapacidad de movilidad a las cuales les resultaría, de algún modo, difícil jugar a cualquier juego de mesa por si solos o participar en torneos profesionales. Por ello se deben crear soluciones y servicios innovadores que permitan el acceso a la tecnología y a la integración social de las personas con alguna discapacidad.

Con el presente proyecto se pretende **fomentar la inclusión social** de las personas discapacitadas en los deportes, en este caso en el ajedrez, manteniendo la motivación de los mismos creando un método distinto de juego para elevar la posibilidad de jugar con normalidad.

Un jugador profesional de ajedrez discapacitado tiene en contra el tiempo de juego, ya que no hay excepción en su tiempo de jugada y debe ser el mismo que su oponente. Estos jugadores necesitan a un acompañante al cual deben transmitirle que ficha quieren mover y donde. Según el tipo de discapacidad la comunicación puede ser una gran perdida de tiempo debido a la dificultad de habla y ocasionaría la perdida de la partida por tiempo. Por ello se propone el diseño de un tablero de ajedrez donde el propio jugador escoja mediante un mando (ya sea mediante un *joystick* o a través de un acelerómetro) la pieza que quiere mover y donde, y una vez seleccionado el tiempo de este jugador se detendrá unos segundos para permitir al ayudante mover la ficha. No solo evitará perder más tiempo del necesario sino que mejorará la comunicación entre jugador y ayudante facilitando la tarea de este ultimo.

1.3. Motivación

La idea del titulo de este proyecto surgió de David González, tutor de este proyecto. Escogí este proyecto y descarté otros títulos debido a que me resultó motivador la posibilidad de crear un ajedrez donde el jugador discapacitado pudiera jugar sin aumentar su tiempo de jugada, sin necesitar de perder tiempo comunicándole que fichas quería mover a su ayudante, etc. A parte de estos motivos, tenía la necesidad de seguir aprendiendo en el mundo de la electrónica y de la programación en C, por esto he decidido hacer el proyecto con el entorno de programación Arduino ya que es una herramienta *Open Source* donde crear proyectos fascinantes es bastante sencillo.

2. ALCANCE Y ESPECIFICACIONES BÁSICAS

2.1. Alcance

Este proyecto tendrá como objetivo principal diseñar un juego de ajedrez donde se facilite al máximo el juego de una persona con alguna discapacidad física que le impida moverse con normalidad y le dificulte el juego.

Toda la parte de programación se controlará con la plataforma Arduino, una herramienta de código abierto para elaborar proyectos de una manera más sencilla.

Se incluirán y comentarán las características de todos los componentes del proyecto ya sea la pantalla, el mando de Nunchuk, los servomotores, los circuitos integrados que se necesiten,…

Se diseñará el primer prototipo del ajedrez con el brazo robot en 3D.

Se incluirá un presupuesto del proyecto y un presupuesto para la fabricación de 100 unidades.

En el anexo se incluirá un manual de instrucciones sobre el modo de uso del ajedrez.

2.2. Especificaciones básicas

A continuación se enumeran las distintas especificaciones básicas que se quieren lograr con el presente proyecto:

- Facilitar el juego de personas con movilidad reducida.
- Mejorar la dinámica de la forma de juego.
- Programación del código integro en C para el temporizador y para la matriz de LED.
- Diseño y fabricación del tablero de ajedrez.
- Diseño y fabricación de la matriz de LED que se utilizarán como indicadores.
- Diseño del brazo robot así como de la pinza en 3D.
- Diseño y fabricación de las placas electrónicas para el control del ajedrez.

3. EL AJEDREZ

3.1. Historia y origen del ajedrez

El ajedrez tiene más de mil doscientos años de historia. Hoy en día existen muchas leyendas sobre el origen real del ajedrez, pero lo que es cierto es que numerosos juegos de mesa se atribuyen a su invención. El ajedrez actual se cree que proviene del juego llamado *shatranj,* que a su vez proviene de del *chaturanga*, inventado en la India el siglo VI.

En el 1300 a.C. se encontraron, junto a la tumba de Tutankamon, piezas y un tablero cuadriculado con una significativa semejanza al ajedrez que conocemos, aunque con un número menor de piezas.

Numerosos expertos aseguran que el ancestro más antiguo del ajedrez es el *chaturanga* y, se cree, que se utilizaba para idear estrategias en el campo. El nombre puede significar "juego de cuatro partes" haciendo referencia a las partes en las que se dividía el ejército en el juego.

La expansión hacia el occidente del ajedrez se debe a los musulmanes con la conquista de la India hacia la segunda mitad del siglo VIII. La primera mención expresa el ajedrez en un texto persa en el año 600. Esta influencia persa se rastrea en algunas palabras como "alfil", que deriva de la palabra árabe "al-fil" elefante que era lo que antiguamente representaba, y la expresión "dar jaque" viene de Sha, rey de los persas y Sha mat significa "el rey ha muerto".

En esta época las reglas del ajedrez eran distintas de las actuales. La dama y el alfil sólo podían avanzar dos casillas, por otro lado el alfil podía saltar, en lugar del enroque[1] existía el "salto del rey" que permitía saltar por encima de una casilla, y los peones solo se movían una casilla. De este modo el juego era muy lento y para darle dinamismo al juego se inventaron las tabiyas, posiciones simétricas con las que por acuerdo de los jugadores comenzaba la partida), el alfil saltaba ya que en su cultura estaba representado por un camello y representaba la superioridad sobre todos los animales.

Con la invasión de los árabes llega el ajedrez a la península ibérica en el siglo VIII. Los primeros indicios del ajedrez en Europa se encontraron en el año 900 en Peñalba de Santiago. En esa época la dama no existía, era la alferza la que ocupaba esa casilla con un movimiento más limitado. El ajedrez se fue transformando hasta día de hoy donde el alfil ahora es un obispo aunque conserva su nombre en recuerdo de su nombre originario.

[1] El enroque es una jugada del ajedrez. Se explica en el apartado de Reglas del juego y movimientos especiales.

En el ajedrez contemporáneo fue donde Philidor Stamma de Aleppo fijó definitivamente las reglas actuales del juego en su libro "El noble juego del ajedrez" , donde también definía la actual notación algebraica.

3.2. Introducción al ajedrez

El ajedrez se considera un deporte en la versión de competición donde dos jugadores intentan eliminar al rey. Existen las variantes de un jugador y un ordenador o dos ordenadores que juegan entre sí.

El juego consta de dieciséis piezas móviles para cada jugador (blancas o claras para un jugador y negras u oscuras para el otro) que se colocan en un tablero de 8x8 casillas.

El ajedrez no es un juego de azar, sino de un juego racional ya que cada jugador puede escoger el movimiento de sus piezas cuando es su turno.

El jugador que empieza siempre es el que juega las fichas blancas, aunque parece una pequeña ventaja, en las competiciones de niveles altos es muy importante quien empieza, por lo tanto se sortea quien será las blancas y quien las negras. También se intenta que cada jugador, en una misma competición, haya jugado similares veces con las piezas blancas y negras.

Una vez decidido quien es el jugador 1 y quien el jugador 2, ambos se turnan para mover alguna de sus piezas. El objetivo es tanto eliminar piezas como irse posicionando en el tablero para dejar al rey del otro jugador sin ninguna pieza que pueda defenderle, esta última jugada se conoce como "jaque mate".

Otras situaciones que también definen el fin del juego puede ser el abandono de la partida de cualquier jugador o si se agota el tiempo de alguno de ellos. En los torneos profesionales se considera perdedor al jugador con una conducta antideportiva ya sea negándose a saludar al jugador contrario o si le suena el teléfono móvil durante una partida.

Por último pueden quedar empate (o también llamado tablas), que se producen en estos casos:

- Por acuerdo común de los jugadores.
- Cuando ningún jugador tiene piezas suficientes para dar jaque mate.
- Si se repite tres veces la misma posición de las piezas en el tablero.
- Cuando un jugador, sin estar en jugada de "jaque mate", no puede realizar en su turno ninguna jugada legal, lo que se denomina "tablas por ahogado".

- Cuando después de cincuenta jugadas consecutivas no se ha hecho ninguna captura o se ha avanzado un peón.

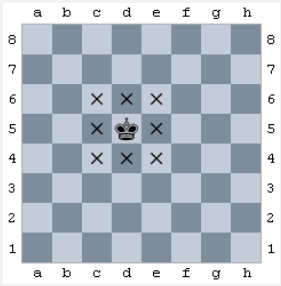
3.3. Elementos del ajedrez

Los elementos básicos de un juego de ajedrez son el tablero y las piezas. Por último, si se trata de una competición, también es imprescindible un temporizador para cada jugador.

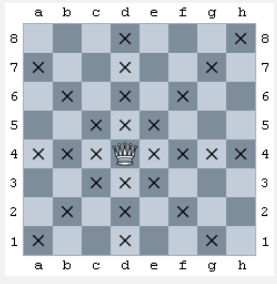
3.3.1. Las piezas

Existen las mismas piezas para cada jugador, siendo unas blancas o de un color claro y otras negras u oscuras. Cada jugador dispone de dieciséis piezas de seis tipos distintos, ocho peones, dos torres, dos caballos, dos alfiles, una dama y un rey.

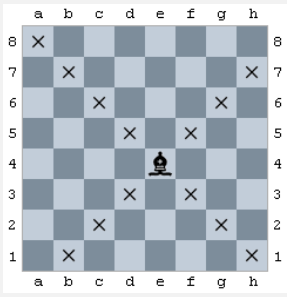
- **Rey**

Descripción	Figura	Movimientos permitidos
El rey puede moverse en cualquier dirección ya sea vertical, horizontal o diagonal avanzando o retrocediendo una sola casilla.		

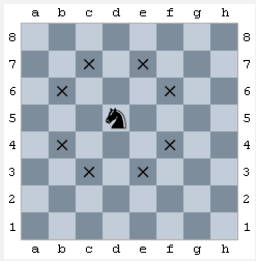
- **Dama**

Descripción	Figura	Movimientos permitidos
La dama se puede mover en cualquier dirección avanzando o retrocediendo en el tablero el numero de casillas que desee, hasta topar con otra pieza o el borde del tablero.		

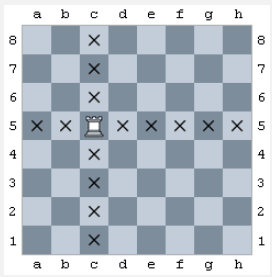
- **Alfil**

Descripción	Figura	Movimientos permitidos
El alfil sólo se puede mover en diagonal hasta topar con otra pieza o con el borde del tablero		

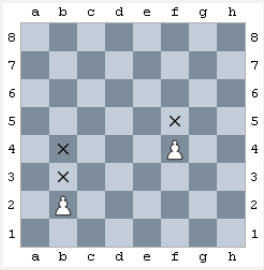
- **Caballo**

Descripción	Figura	Movimientos permitidos
El caballo puede moverse dos casillas en vertical y una en horizontal o viceversa. Es la única pieza que puede saltar a las demás.		

- **Torre**

Descripción	Figura	Movimientos permitidos
La torre sólo puede moverse en dirección vertical y horizontal, nunca en diagonal, hasta topar con otra pieza del tablero o el borde.		

- **Peón**

Descripción	Figura	Movimientos permitidos
El peón puede avanzar una o dos casillas en dirección vertical en su primer movimiento, después solo una. No puede retroceder y solo puede matar en diagonal. Si llega a la última fila podrá transformarse en la pieza que el jugador desee.		

3.3.2. El tablero

El tablero de ajedrez es un cuadrado subdividido en sesenta y cuatro casillas o escaques iguales. Su dimensión es de ocho casillas en vertical y ocho casillas en horizontal, están coloreadas alternativamente en un color claro y otro oscuro. El tablero se coloca de forma que cada jugador tenga una esquina blanca en su parte derecha. A continuación se ilustra una imagen de un tablero con las piezas colocadas:

Figura 1 - Tablero de ajedrez

La colocación de las figuras es la siguiente:

Tabla 2 - Colocación de figuras

Figura	Ubicación	
	Blancas	Negras
Rey	1-d	8-d
Dama	1-e	8-e
Alfil	1-c , 1-f	8-c, 8-f
Caballo	1-b , 1-g	8-b , 8-g
Torre	1-a , 1-h	8-a , 8-h
Peón	Toda la fila 1	Toda la fila 7

Los elementos básicos del tablero son los siguientes:

- **Fila** : Cada una de las ocho líneas de ocho casillas que se forman alineando éstas horizontalmente respecto a los jugadores. Van numeradas del 1 al 8 dónde la fila 1 es la más cercana del jugador con las piezas blancas.
- **Columna** : Cada una de las ocho líneas de ocho casillas que se forman alineando éstas verticalmente respecto a los jugadores. Se nombran con letras minúsculas de la *a* a la *h*, comenzando desde la primera columna izquierda con respeto al jugador de las piezas blancas.
- **Diagonal** : Cada una de las 16 líneas que se forman agrupando las casillas diagonalmente. Las diagonales más grandes tienen ocho casillas.
- **Centro** : El centro del tablero son los cuatro escaques centrales, aunque a veces se tienen en cuenta las doce casillas que rodean a estas cuatro centrales.
- **Esquinas** : Cada una de las cuatro casillas ubicadas en las esquinas del tablero.
- **Bordes** : Las dos columnas (a y h) y las dos filas (1 y 8) situadas en los extremos del tablero.

La finalidad de numerar el tablero es la facilidad de poder reproducir y comentar las partidas. Su función más importante es el registro del desarrollo de la partida mediante la notación algebraica.

3.3.3. El temporizador

El temporizador del juego consiste en un doble cronómetro que controla el tiempo que le queda a cada jugador para realizar sus movimientos. Cada vez que un jugador presiona su pulsador se detiene su tiempo y automáticamente se activa el del jugador contrario.

El temporizador analógico tiene una tuerca en la parte posterior para darle cuerda antes de empezar la partida. Algunos de estos incluyen una bandera que cae cuando el jugador ha perdido la partida. Por otro lado, el temporizador digital funciona con pilas y avisa del fin del tiempo con un pitido.

Figura 2 - Temporizador analógico

Figura 3 - Temporizador digital

Hay diferentes ritmos de partida según el tipo de juego que se quiera realizar y se clasifican según la duración de ésta:

- **Partida Blitz o relámpago:** Aquella en la cual cada jugador dispone de un máximo de 15 minutos para terminar la partida.
- **Partida rápida:** Aquella en la cual cada jugador está entre 15 y 60 minutos. Es la más usada en desempates de torneos y su tiempo se fija en 25 minutos.
- **Partida estándar (ritmo clásico) :** Aquella en la cual el tiempo es superior a 60 minutos. Es el ritmo más usado a nivel magistral. En los torneos el tiempo suele ser de 90 minutos.

Las partidas sin reloj no tienen un nombre específico, se suelen llamar partidas amistosas o de café.

4. LA DISCAPACIDAD

4.1. Introducción

Según la OMS (Organización Mundial de la Salud), el término discapacidad abarca las deficiencias, las limitaciones de las actividades y las restricciones de la participación.

Estas deficiencias son problemas que afectan a una estructura o función corporal; las limitaciones de la actividad son dificultades para ejecutar acciones o tareas, y las restricciones de la participación son problemas para participar en situaciones vitales.

Una **deficiencia** es toda pérdida o anormalidad de una estructura o función psicológica, fisiológica o anatómica. Una **discapacidad** es toda restricción o ausencia, debida a una deficiencia, de la capacidad de realizar una actividad en la forma o dentro del margen que se considera normal para el ser humano. Una **minusvalía** es una situación desventajosa para un individuo determinado consecuencia de una deficiencia o una discapacidad, que limita o impide el desempeño de un rol.

Hay cerca de mil millones de personas con alguna tipo de discapacidad, y de éstas, casi doscientos millones experimentan dificultades considerables en su funcionamiento.

La discapacidad será un motivo de preocupación aún mayor en el futuro, ya que su prevalencia está aumentando. Esto se debe a la población que está envejeciendo y el riesgo de discapacidad es superior entre los adultos mayores. También aumentara debido a las enfermedades crónicas como la diabetes, enfermedades cardiovasculares, el cáncer y los trastornos de la salud mental.

4.2. Tipos de discapacidad

Hay distintos tipos de discapacidades. En general la población cree que es una condición permanente. Sin embargo, hay discapacidades temporales y discapacidades permanentes. Por ejemplo, la fractura de un pie o de un brazo, o la perdida de visión por algún agente químico nocivo, es una discapacidad temporal normalmente. Por lo tanto la mayoría de la población se ha encontrado en situación de discapacidad alguna vez en la vida.

Por otro lado, también hay distintos grados de discapacidad: leve, moderada o severa. Según el tipo de discapacidad, habrán uno o varios especialistas que evalúen a través de distintas pruebas específicas el nivel que presenta.

Una vez conocidos los grados, estos son los tipos de discapacidad:

- **Psíquica:** Ya puede ser mental, intelectual, cognitiva,… Es una disminución en las habilidades cognitivas e intelectuales del individuo. Son del tipo Retraso Mental, Síndrome de Down.
- **Física:** Del tipo motriz, motora,… Quienes padecen esta discapacidad se ven afectadas sus habilidades motrices. Son del tipo Parálisis Cerebral, Espina Bífida,…
- **Sensorial:** En este grupo se encuentran aquellas discapacidades relacionadas con la disminución de uno o varios sentidos. Son del tipo auditiva, visual o multisensorial.

El proyecto se centra en discapacidades físicas. Dentro de este tipo se encuentran éstas, aunque hay más tipos, pero éstas son las más conocidas:

- **Lesión medular:** Daño presentado en la medula espinal debido a una enfermedad, accidente,… Según la zona afectada puede ser paraplejía (afecta a las piernas, no a los brazos) o tetraplejía (debilidad en los brazos y parálisis completa de las piernas)
- **Parálisis cerebral:** Conjunto de desórdenes cerebrales que afecta el movimiento y la coordinación muscular. Es causada por el daño de áreas específicas del cerebro, generalmente durante el desarrollo fetal.
- **Mal de Parkinson:** Enfermedad neurológica de las más comunes y causa una lenta pérdida de la capacidad física.
- **Espina bífida:** Malformación congénita debida a la falta de cierre o fusión de uno o varios arcos posteriores de la columna vertebral. Suele manifestarse en el primer mes de embarazo y la causa es la falta de ácido fólico en el organismo de la madre.
- **Distonia muscular:** Síndrome que consiste en contracciones musculares sostenidas en el tiempo. Suele provocar torsiones, movimientos repetitivos y posturas anómalas. También presenta tics regulares o irregulares en varias partes del cuerpo.

4.3. Campeonatos para discapacitados

La mayoría de campeonatos [2] para personas con discapacidad son para deficientes visuales y/o ciegos.

Una de las federaciones españolas que apuestan por el deporte como forma de participación e integración social es ASEMCAN [3]. Esta federación organiza campeonatos para gente con enfermedades neuromusculares para disfrutar del ocio y del deporte que proporciona el ajedrez.

A nivel mundial otro campeonato que se celebró en Croacia (Split) en 2012 fue la 12va edición del Campeonato Mundial de Ajedrez para personas con Discapacidades Físicas, un torneo jugado en formato suizo de 9 rondas con 90 minutos + 30 segundos por jugada de ajedrecista.

[2] Para ver los campeonatos organizados puede entrar en la web de la Federación Española de Deportes para Ciegos http://www.fedc.es .
[3] Federación Española de Enfermedades Neuromusculares (www.asem-esp.org) .

5. HARDWARE

5.1. Arduino

Arduino es una plataforma de *hardware* libre (*Open source)* basada en el microcontrolador ATmega 328 (en el caso de Arduino UNO Rev3). Su uso más común son los proyectos de electrónica. Según las necesidades del proyecto existen diferentes versiones más complejas donde el numero de entradas y salidas es mayor para poder controlar más sensores, interruptores, actuadores,…

En el proyecto se utilizan dos placas de la versión UNO Rev3, donde el número de entradas digitales configurables es de catorce (de la 0 a la 13) y operan a 5V (se puede aumentar el valor utilizando los contactos AREF y modificando el código de la placa) y como máximo puede proporcionar o recibir 40 mA. De estas catorce E/S digitales hay seis que proporcionan salidas por modulación del ancho de pulso (PWM), estos contactos son el 3, 5, 6, 8, 10 y 11. Los contactos 0 y 1 controlan la comunicación serie y si hay algún dispositivo conectado interferirá en la escritura por el puerto USB. En el proyecto se comunican las dos placas mediante dos cables que las unen, por ello, cada vez que se necesite cargar un código a cualquiera de las dos placas es necesario desconectar la unión entre ellas. Los pines para la conexión de I^2C son el A4 para el SDA y el pin A5 para el SCL. Si necesitamos crear interrupciones en el código, los pines son el 2 para la interrupción 0 y el pin 3 para la interrupción 1.

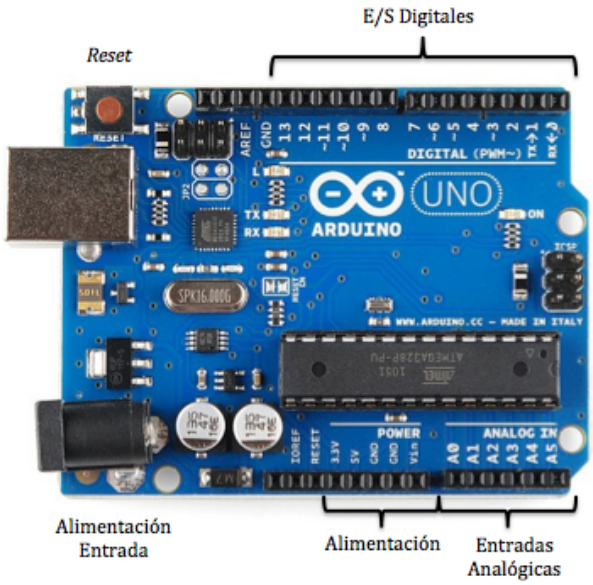


Figura 4 - Placa Arduino UNO Rev3

Las especificaciones técnicas de la placa Arduino son las siguientes:

Tabla 3 - Especificaciones técnicas Arduino

Modelo:	Arduino UNO – Rev 3
Microcontrolador:	ATmega 328
Voltaje de entrada:	7-12 V
Voltaje del sistema:	5 V
Frecuencia del reloj:	16 MHz
E/S Digitales:	14
Salidas PWM:	6
Entradas Analógicas:	6
UART[4]:	1
Memoria *Flash*:	32 Kb
Cargador:	Optiboot
Interfaz programación	USB

5.2. Mando de control (*Nunchuk*)

Nunchuk es un accesorio de expansión para el mando inalámbrico Wii de Nintendo. Este mando controla todos los movimientos del jugador para mover un personaje, un objeto…

Figura 5 - Mando Nunchuk

[4] UART : Universal Asynchronous Reciver-Transmitter (Transmisor – Receptor Asíncrono Universal).

Utiliza un bus de transmisión de datos I^2C. La principal característica de esta comunicación serie es que solo necesita dos líneas para transmitir la información de los sensores. Una de ellas es la señal de reloj (SCL) y la otra señal es la de datos (SDA). Cada dispositivo que se conecta al bus tiene una dirección única y se define un Maestro (es el que inicia la transmisión de datos y genera la señal de reloj) y los demás como Esclavos (reciben la señal de reloj y envían datos al maestro).

El mando Nunchuk incluye dos pulsadores, un *joystick* analógico en su exterior y un acelerómetro digital en su interior. En la parte izquierda se puede observar el Microcontrolador FNURVL 405 849KM de 36-pin que controla toda la electrónica del mando y envía todos los datos por el bus mediante la comunicación I^2C. En la figura 6 se muestra la parte posterior de la placa:

Figura 6 - Parte posterior Nunchuk

Joystick

En la parte izquierda (Figura 7) encontramos el *joystick* formado por dos potenciómetros axiales de 30KΩ, estos detectan la posición mediante la resistencia de cada uno de los dos ejes.

En la figura 7 se muestra la placa por la parte anterior:

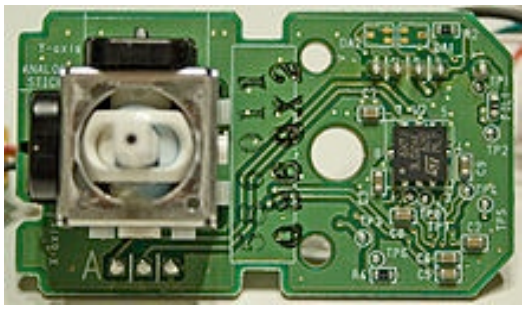


Figura 7 – Parte anterior Nunchuk

Mediante el puerto serie del software de Arduino y la librería ArduinoNunchuk.h podemos leer los datos de estos potenciómetros para saber su valor en cada posición. Para los potenciómetros se utilizan 8 bits así que el rango aproximado será de 0 a 255.

Para el eje horizontal (eje X) estos son los valores de los extremos:

Tabla 4 - Valores del eje X del potenciómetro

	Izquierda	Centro	Derecha
Valor aproximado	32	131	224

Para el eje vertical (eje Y) estos son los valores registrados en los extremos:

Tabla 5 – Valores del eje Y del potenciómetro

	Abajo	Centro	Arriba
Valor aproximado	23	121	216

Acelerómetro

En la parte derecha de la parte anterior de Nunchuk (Figura 7) se encuentra el acelerómetro de la marca STMicroelectronics y es el modelo LIS3L02AL. Según su *datasheet* estos son los ejes del acelerómetro.

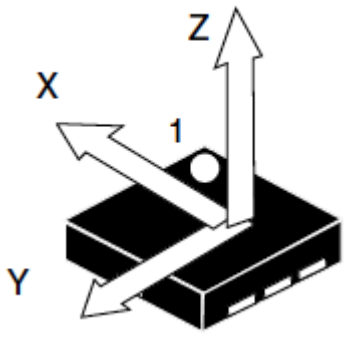


Figura 8 - Ejes del acelerómetro LIS3L02AL

Dentro del mando así quedarán los ejes:

Figura 9 - Ejes acelerómetro Nunchuk

La aceleración se envía con 10 bits por lo que el rango de valores será de 0 a 1023.

Para definir más adelante los valores para mover el cursor por el ajedrez, se irán cambiando los valores del acelerómetro para ajustar el movimiento de muñeca y no dar falsos movimientos, ni errores en la lectura de un movimiento bueno.

Para hacerse una idea del rango de valores del acelerómetro procedemos a la lectura de esos valores con el puerto serie de Arduino y la librería utilizada anteriormente, en este proyecto solo se utilizaran los ejes X e Y del sensor.

Para el eje X estos son los valores:

Tabla 6 - Valores eje X acelerómetro

	Izquierda	Centro	Derecha
Valor aproximado	350	500	650

Para el eje Y estos son los valores registrados:

Tabla 7 - Valores eje Y acelerómetro

	Atrás	Centro	Adelante
Valor aproximado	350	500	650

Botones C y Z

Por último los valores de los botones C (botón superior) y Z (botón inferior) los cuales utilizan solo un bit para enviar su estado, por lo tanto el valor será 0 o 1.

Son dos pulsadores de membrana y en la siguiente figura se pueden observar desde el exterior.

Figura 10 - Botones C y Z

Según la librería utilizada en el entorno de programación estos son los estados de los pulsadores.

Tabla 8 - Valores botones C y Z

	Reposo	Pulsado
Botón C	0	1
Botón Z	0	1

Conector

El conector del mando dispone de seis conectores aunque realmente solo utilizan cuatro. En la figura siguiente se muestra en la parte superior el conector del mando Wii y en la parte inferior el conector de Nunchuk.

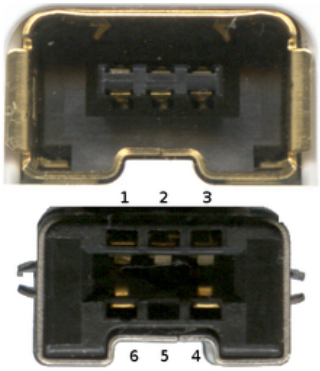


Figura 11 - Conector Nunchuk

En la siguiente tabla se muestran la conexión de cada pin:

Tabla 9 - Conexiones Nunchuk

	Conexión
Pin 1	Data
Pin 2	-
Pin 3	+3,3V
Pin 4	Clock
Pin 5	-
Pin 6	Masa

Para conectar el mando a Arduino se ha utilizado un adaptador que simula el puerto de entrada de la consola Wii.

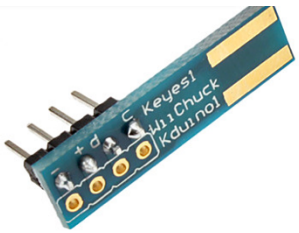


Figura 12 - Adaptador Nunchuk

5.2.1. Conexión Nunchuk en Arduino

A continuación se muestra el conexionado del mando de Nunchuk con Arduino.

Tabla 10 - Conexionado Nunchuk - Arduino

Pin Arduino	Pin Nunchuk
A4	1
-	2
5V	3
A5	4
-	5
GND	6

Los puertos A5 y A6 hacen referencia a las entradas analógicas 4 y 5.

Por último se muestra el esquema de conexión:

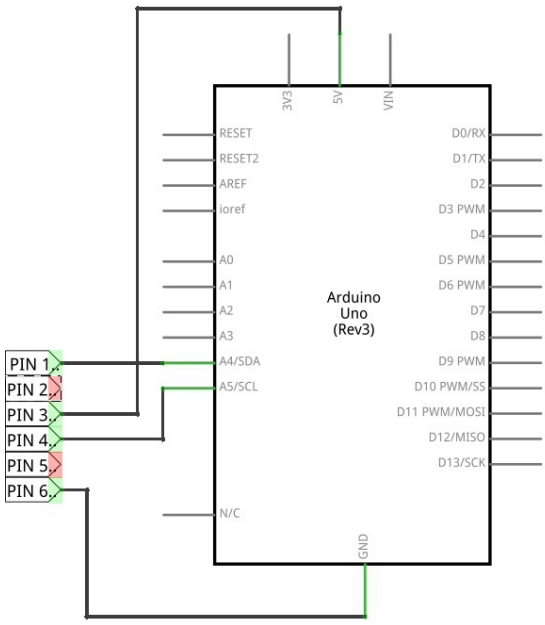


Figura 13 - Conexión Arduino – Nunchuk

5.3. Pulsadores

Para el proyecto se han utilizado pulsadores para controlar la pantalla LCD. Se utilizan cuatro para el botón UP, DOWN, OK y NEXT.

5.3.1. Conexión pulsadores en Arduino

La conexión de los pulsadores o interruptores es la misma para todos:

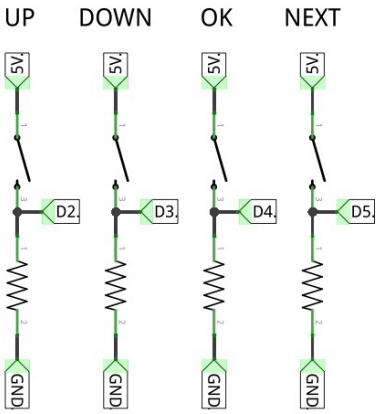


Figura 14 - Conexión de un pulsador a Arduino

Las etiquetas D2, D3, D4 y D5 corresponden al pin digital donde se conectaran los distintos botones (2, 3, 4 y 5).

Se detecta pulsación cuando se lee un valor digital alto, ya que en reposo los pines digitales leen un valor booleano de 0 (pin conectado directamente a tierra), en el caso de estar pulsado se detectará un valor booleano de 1 (pin conectado a 5V).

5.4. Matriz de LEDS 8x8

El tablero de ajedrez está formado por ocho columnas y ocho filas. Para simular el tablero de ajedrez se hará toda la parte práctica con una matriz de LED de 8x8 ya fabricada, cuando todo funcione se diseñara la matriz de 8x8 adaptada para el tablero de ajedrez.

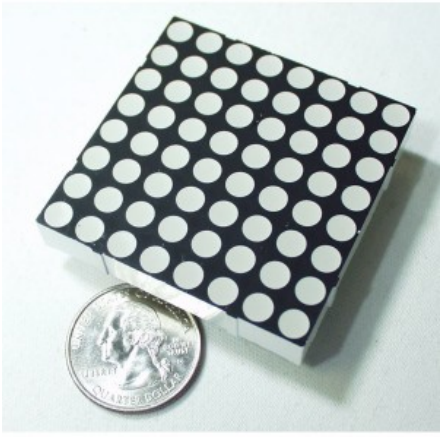


Figura 15 - Matriz LED 8x8

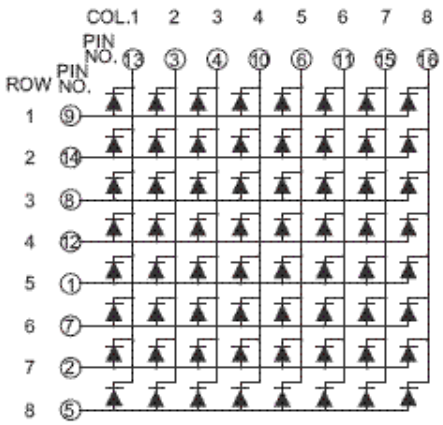


Figura 16 - Esquema matriz 8x8

Para poder controlar una matriz de 64 diodos se necesitarían 16 pines digitales para controlar el estado de cada uno. La solución es utilizar el circuito integrado MAX7219 o MAX7221. Este chip es capaz de controlar hasta 64 diodos independientemente. Para lograrlo utiliza un *decoder* BCD code-B, también utiliza el multiplexado y una memora RAM interna para almacenar el valor de cada dígito. Solo utiliza una resistencia externa de 100 Ω, un condensador cerámico de 100 nF y un condensador electrolítico de 10 μF. La resistencia es utilizada para no quemar los diodos LED de la matriz (se verifica el valor en el apartado de 7.1. Diseño y fabricación de la matriz de LED).

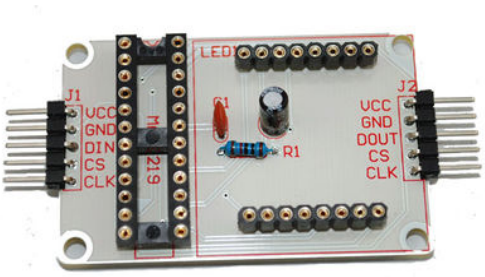


Figura 17 - Placa MAX7219 sin montar

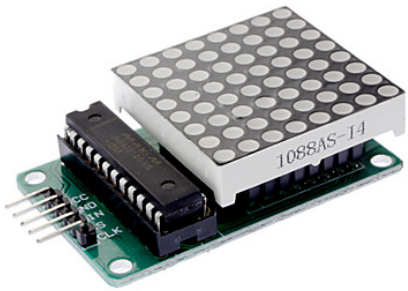


Figura 18 – Placa MAX7219 montada

Se controla con cinco cables, a continuación se muestra una tabla con las entradas y salidas de la placa .

Tabla 11 - Conexiones placa montada y pines MAX7219

Pin placa montada	Nombre	PIN MAX7219
1	VCC → +5V	19
2	GND →Tierra	4 y 9
3	*Data In* (DIN)	1
4	*Load Chip Select* (CS)	12
5	*Clock* (CLK)	13

Para controlar este circuito desde Arduino se necesita una librería que sepa comunicarse con este chip, para ello se ha decidido utilizar la librería LedControl.h[5].

Esta librería tiene funciones muy interesantes para el proyecto ya que se puede controlar cada LED mediante coordenadas , aquí se muestra un ejemplo del código de Arduino:

```
lc.setLed(0,row,col,true);
```

Con el comando *setLed* se envía un 0 seguido de la fila , la columna y un *true* o un *false* para definir el estado de ese LED.

[5] Más adelante, en el apartado de software, se explica detenidamente todas las funciones de esta librería.

5.4.1. Conexión matriz en Arduino

A continuación se muestra el conexionado de la matriz con Arduino.

Tabla 12 - Conexionado placa montada MAX7219 - Arduino

Pin Arduino	Pin placa montada
5V	VCC
GND	GND
2	DIN
3	CS
4	CLK

Por último se muestra el esquema del conexionado:

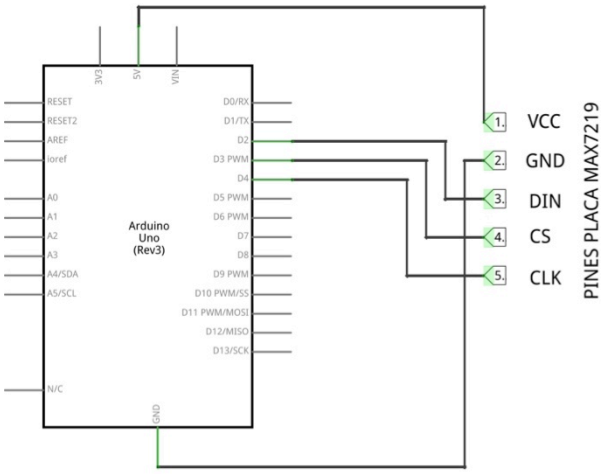


Figura 19 - Conexión Arduino – Placa MAX7219

5.5. Pantalla LCD

Las pantallas de cristal líquido o LDC (*Liquid Crystal Display*) es una pantalla delgada y plana formada por un número determinado de píxeles ya sea de color o monocromo colocadas delante de una fuente de luz o reflectora. Se utilizan para dispositivos electrónicos ya que consumen unas cantidades muy pequeñas de energía.

Hay distintos tamaños de pantallas según las necesidades del proyecto. A continuación se muestran dos tamaños de pantallas:

Figura 20 - Pantalla LCD 2x16

Figura 21 - Pantalla LCD 4x20

Para el proyecto se ha escogido una pantalla de 4 filas y 20 *pixel* por fila. Se he escogido una pantalla de 4 filas para poder mostrar toda la información de los tiempos de cada jugador y por lo tanto tendrá 20 *pixels* por fila.

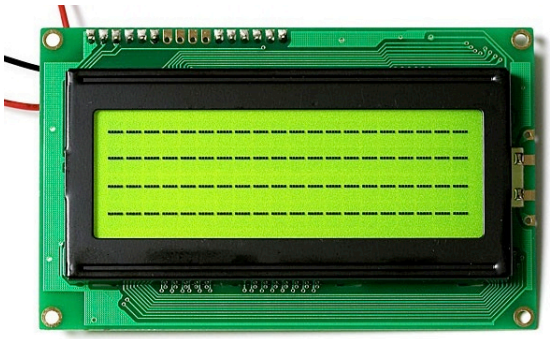


Figura 22 – Pantalla LCD 4x20

El módulo de la pantalla dispone de 16 pines que se describen a continuación :

Tabla 13 – Conexiones pantalla LCD

Pin Pantalla LCD	Símbolo	Descripción
1	V_{SS}	Tierra (GND)
2	V_{DD}	Alimentación 5V (+5V)
3	V_0	Ajuste del contraste
4	RS	Señal del selector del registro
5	R/W	Señal de Lectura/Escritura
6	E	Disponibilidad de Lectura/Escritura
7-10	DB0-DB3	No usados
11-14	DB4-DB7	Control de los valores
15	LED+	Alimentación para contraste
16	LED-	Tierra para contraste (GND)

La señal de alimentación corresponde a los pines 1, 2 y 3. El pin 1 corresponde a la masa, el 2 a la alimentación positiva donde normalmente es de 5V en continua, y el pin 3 ajusta el contraste de la pantalla, cuanto más cercano sea el valor a 0 más contraste tendrá la pantalla.

La señal del control proviene de los pines 4, 5 y 6. El pin 4 selecciona el registro del dato enviado, el pin 5 controla si se puede escribir o leer en la pantalla (vale 1 cuando se puede leer y 0 cuando se puede escribir) y el pin 6 permite habilitar (E=1) o deshabilitar (E=0) la comunicación con la pantalla.

Por último, la señal de datos pertenece a los pines del 7 al 14. Estos forman un bus de doble dirección de 8 bits por donde se pueden enviar instrucciones o datos al microcontrolador o conocer su estado.

Por falta de pines disponibles en la placa Arduino se usará otra placa independiente de la primera que controlará solo los tiempos de juego.

La función de la pantalla será la siguiente:

- Dar la bienvenida a los jugadores.
- Ajustar el tiempo total de la partida mediante pulsadores.
- Ajustar el tiempo incrementado por jugada.
- Elección del modo de juego (*JOYSTICK* O ACELERÓMETRO)
- Mostrar el tiempo restante de cada jugador.
- Mostrar aviso cuando el brazo robot esté actuando (o el ayudante esté moviendo la ficha).
- Mostrar un aviso cuando se agote el tiempo de algún jugador.

La pantalla ya incluye un procesador para poderlo controlar de manera sencilla con Arduino y una librería especifica para pantallas de cristal líquido. Esta librería se llama LiquidCrystal.h y está explicada en el apartado 6.2.5. Librería LiquidCrystal.

Bienvenida a los jugadores

La pantalla mostrará el mensaje "BIENVENIDOS AL AJEDREZ ADAPTADO"

Figura 23 - Mensaje bienvenida

Este mensaje será visible durante 4 segundos (4000 ms) y desaparecerá.

Ajuste del tiempo de partida

Una vez mostrado el mensaje de bienvenida, la pantalla mostrará lo siguiente:

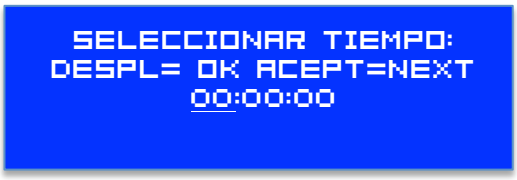


Figura 24 - Mensaje tiempo de partida

El tiempo se escogerá con los pulsadores "Subir" y "Bajar" (*Up* y *Down*) para incrementar o reducir las horas, minutos o segundos en una unidad. Para desplazar el cursor a la siguiente casilla se pulsará el botón "OK" y una vez configurado el tiempo total del juego se pulsará "Siguiente" (*Next*) para aceptar el tiempo.

Incremento del tiempo por jugada

En la mayoría de partidas suele haber un tiempo que se añade después de cada jugada al tiempo total del jugador que acaba de mover. En este menú se seleccionará los minutos y/o segundos que se añadirán a cada jugada. Como en la pantalla anterior, se escogerá con los pulsadores "Subir" y "Bajar" (*Up* y *Down*) el tiempo de incremento y se desplazará con el botón "OK". Para aceptar el tiempo se pulsará "Siguiente" (*Next*).

Figura 25 - Mensaje incremento de tiempo

Elección del modo de juego

Una vez escogido el tiempo añadido se deberá seleccionar el modo de juego del mando. El modo *Joystick* solo deja activo el mando físico del Nunchuk, por otro lado, el modo Acelerómetro deja activo el sensor de aceleración interno del mando. Para escoger un modo u otro se deben utilizar los botones "Subir" y "Bajar" para mover el cursor entre las dos opciones. Una vez seleccionado se pulsará el botón "NEXT" para aceptar el modo.

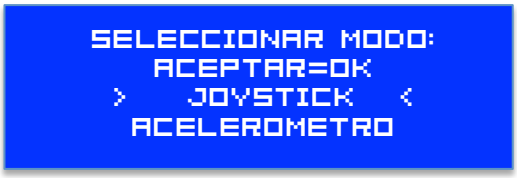


Figura 26 - Elección del modo de juego

Una vez seleccionado el modo se enviará por el puerto serie el modo escogido para que el otro Arduino recoja datos solo con el *joystick* o con el acelerómetro, después nos muestra la elección por pantalla. A continuación sonará un pitido para indicar que empieza la partida.

Figura 27 - Confirmación modo de juego

Tiempo restante de cada jugador

Después del pitido de inicio de partida se escuchara otro pitido para indicar que está jugando el jugador numero 1 (un pitido) y el tiempo empezará a descontara del total.

Figura 28 - Mensaje tiempo restante

La primera fila muestra de que jugador es el turno, la segunda muestra el mensaje "tiempo restante" y en las dos filas inferiores los tiempos de cada jugador para finalizar. A cada cambio de jugador se escucharan uno o dos pitidos dependiendo del jugador que le toque mover.

Pieza en movimiento

Una vez iniciada la partida y el temporizador esté contando, desaparecerá el mensaje de TURNO J1 cuando el brazo robot esté moviendo una pieza y el tiempo dejará de descontar. Una vez finalizado el movimiento y este Arduino reciba por el puerto serie que se ha finalizado el movimiento, automáticamente se cambiará al jugador 2 y su tiempo empezará a descontar. Hay que recordar

que a cada cambio de jugador se incrementa el tiempo de jugada una cantidad "x" definida al principio del juego como tiempo *add.*

Figura 29 – Mensaje pieza en movimiento

Aviso tiempo agotado

Cuando el tiempo de alguno de los dos jugadores finaliza, en la pantalla se muestra el mensaje siguiente:

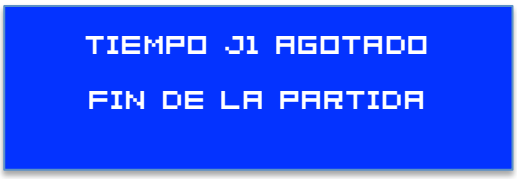


Figura 30 - Mensaje tiempo agotado

Una vez finalizado el tiempo se oirá un pitido indefinido hasta que se reinicien las placas de Arduino.

Para empezar una nueva partida es necesario reiniciar la placa Arduino con el interruptor ON-OFF.

5.5.1. Conexión pantalla LCD en Arduino

A continuación se muestra el conexionado de la pantalla LCD a Arduino.

Tabla 14 – Conexionado pantalla LCD - Arduino

Pin Pantalla LCD	Pin Arduino
1, 5 y 16	GND
2 y 15	+5V
3	Centro potenciómetro
4	7
6	8
11	9
12	10
13	11
14	12

Por último se muestra el gráfico de conexionado:

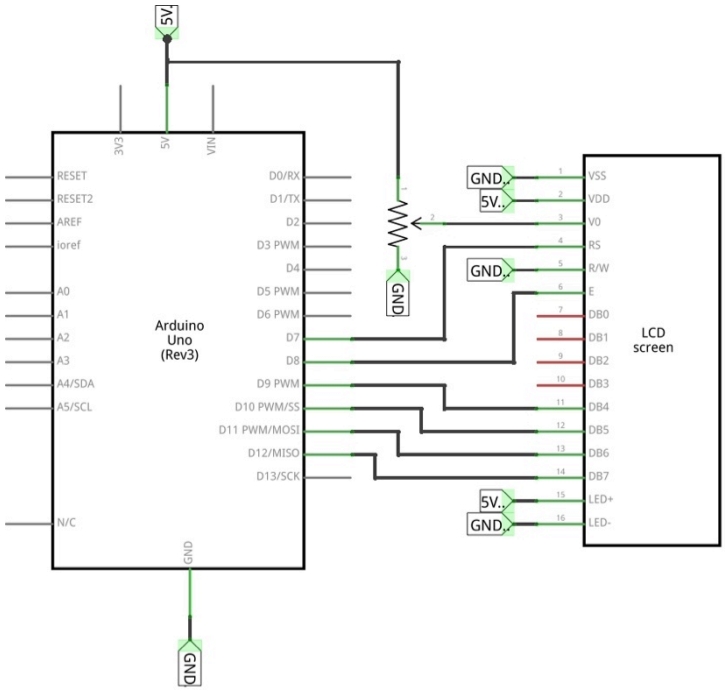


Figura 31 – Conexión Arduino – pantalla LCD

5.6. Servomotor

Un servomotor es un dispositivo muy utilizado en proyectos de robótica o en proyectos de radiocontrol. La característica principal de un servomotor es que podemos controlar la posición de su eje enviando una señal codificada mediante un tren de pulsos. Mientras esta señal sea existente el circuito de control mantendrá la posición fija que se le haya asignado. En la siguiente figura se observa como es un servomotor.

Figura 32 - Servomotor

Estos dispositivos tienen un gran rendimiento y son muy poderosos en proporción a su tamaño tan reducido. Para conseguir tal potencia se transmite la potencia del motor al eje mediante engranajes metálicos o de plástico.

Figura 33 - Interior de un servomotor

Para conocer la posición del eje de salida, este está acoplado directamente a un potenciómetro. Es por este motivo que nunca podrá dar la vuelta entera el eje ya que un potenciómetro tiene sus limitaciones en cuanto al giro. De todos modos abarca aproximadamente 180 grados.

El conexionado es sumamente simple, está formado por tres cables, uno para la alimentación, comúnmente de color rojo (Vcc), otro para conectar a masa de color negro (GND) y uno para el control que suele ser de diferentes colores según el fabricante.

El control de posición es proporcional, es decir, la cantidad de voltaje que aplica al motor es proporcional a la distancia que necesita recorrer. Por lo tanto, si tiene que barrer una distancia muy larga, la velocidad será muy elevada, en cambio, si es una distancia pequeña lo hará de manera más lenta. Una vez recibida el amplificador de error calcula el valor entre la posición en que se encuentra el motor y el valor de referencia. Si el valor es distinto de 0 aplicará un voltaje al motor hasta que el error sea nulo.

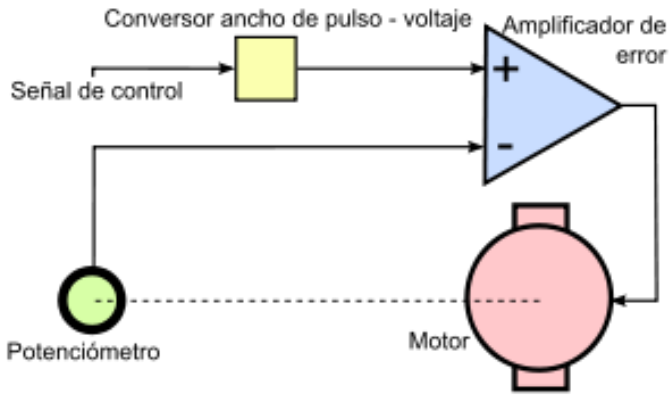


Figura 34 - Circuito de control de un servomotor

La señal de control es un tren de pulsos para definir que posición es la marcada. El ancho del pulso es proporcional al ángulo de salida. En la siguiente figura se compara el ángulo con el pulso de salida.

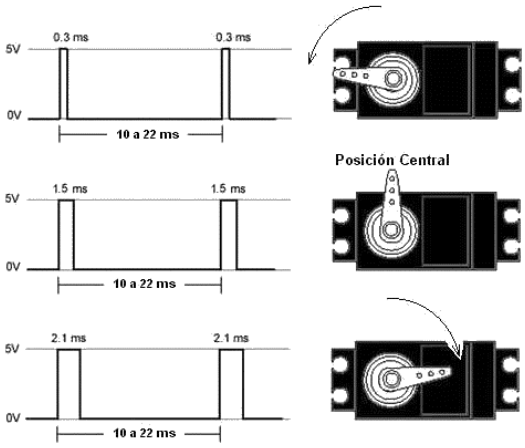


Figura 35 - Pulso-Ángulo servomotor

El pulso varía según el fabricante pero siempre está entre 1 ms y 2 ms. Si el ángulo de salida es 0º el pulso será aproximadamente de 1 ms, si el ángulo de salida es de 180º rondará los 2 ms y cualquier valor entre 0º y 180º es el valor proporcional entre 1 ms y 2 ms.

En la siguiente tabla se comparan los valores de cada servomotor de los fabricantes más utilizados:

Tabla 15 - Características servomotor según fabricante

Fuente: roboticapy.com

	Ancho pulso			Color cables		
Fabricante	**Mín.**	**Medio**	**Máx.**	**positivo**	**negativo**	**PWM**
Futaba	0.9	1.5	2.1	rojo	negro	blanco
Hitech	0.9	1.5	2.1	rojo	negro	amarillo
aRC-Universe	0.8	1.5	2.2	rojo	marrón	naranja
Multiplex	1.05	1.6	2.15	rojo	negro	amarillo
Robbe	0.65	1.3	1.95	rojo	negro	blanco
Simprop	1.2	1.7	2.2	rojo	azul	negro

5.6.1. Brazo robot

Para diseñar el brazo robot se utilizarán cuatro servomotores, uno para el giro del brazo en el eje x (hombro), otro para el giro del eje y (codo), otro para el giro del eje z (muñeca) y uno de menor potencia para la apertura/cierre de la pinza.

- 3 x Servo aRC-Universe MG946R:
  - Tipo: Analógico
  - Ángulo máximo: 180º
  - Peso: 55 g
  - Tensión: 4,8 V
  - Par: 12.1 Kg·cm

- 1 x Servo Turnigy TG9e:
  - Tipo: Analógico
  - Ángulo máximo: 180º
  - Peso: 9 g
  - Tensión: 4,8 V
  - Par: 1.5 kg·cm

En el siguiente esquema observaremos la colocación de cada servo en el brazo:

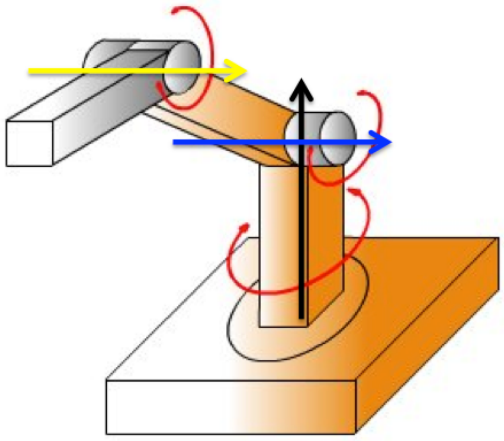


Figura 36 - Ejes del brazo robot

El eje negro representa el giro del hombro, el eje azul el giro del codo y el eje amarillo el giro de la muñeca. Falta incluir el servomotor que se encarga de abrir y cerrar la pinza que no se muestra en la figura anterior.

En el apartado de "7.3. Diseño del brazo robot" se especifican todas las partes que componen el brazo robot de tres ejes y en el apartado de planos se detallan todas las medidas para un futuro montaje.

5.6.2. Conexión Servomotor-Arduino

A pesar de no construir el brazo robot se especifican todas las conexiones necesarias para su correcto funcionamiento.

Para la conexión de los servomotores con Arduino se necesitará más potencia de la que nos da la propia placa, si se conectan los 4 servomotores a la placa podría destruirla ya que la demanda de corriente de los servomotores es muy elevada respecto a la que la placa puede proporcionar, por lo tanto se necesitará una alimentación externa para los servomotores:

Los seis cables que van hacia la placa y hacia el porta pilas (cuatro de señal, VCC y la tierra) se les soldará un conector para poder desacoplarlo del circuito en el caso de no querer utilizar el brazo robot en las partidas.

En este caso la placa Arduino utilizada es la que controla las luces LED del tablero ya que es la que dispone de más salidas PWM libres.

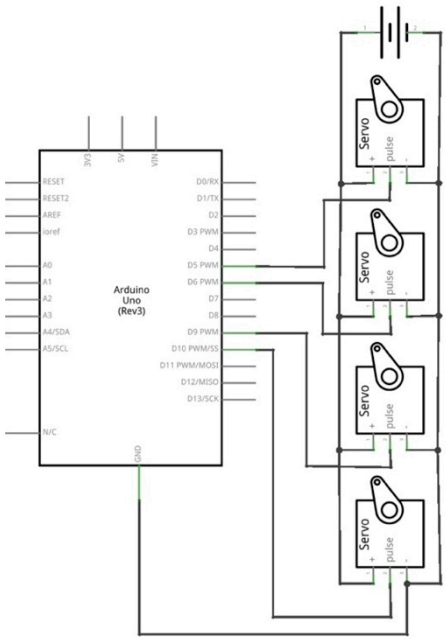


Figura 37 – Conexión Servomotores – Arduino

El encargado del giro del brazo se conectará al pin 5, el encargado de elevar el primer brazo al pin 6, el que eleva el brazo dos al pin 9 y el que abre y cierra la pinza al pin 10. Se deben conectar las tierras entre Arduino y el porta pilas de 6 V.

5.7. Comunicación serie entre Arduino

Existen varios métodos de unir dos placas de Arduino mediante la comunicación *serial.* A través de esta comunicación podemos enviar datos a nuestro

dispositivo, y de igual modo recibirlos. Hay que tener en cuenta que si conectamos el puerto serie entre dos placas la conexión USB con el ordenador queda anulada y no podemos cargar código a menos que desconectemos la conexión serie que interrumpe la escritura.

Para configurar el dispositivo se tiene que definir la velocidad de transmisión de datos, normalmente se utilizan 9600 baudios [6], aunque en el proyecto utilizaremos 19200. Esta velocidad se define en el *setup* del programa del siguiente modo:

```
Serial.begin(19200);
```

Una vez definido la velocidad de transmisión de datos solo debe utilizarse la función Serial.*print* o Serial.*println* (esta segunda crea un salto de línea) para enviar datos y la función Serial.*read* para leer los datos.

La comunicación serie de Arduino envía y recibe datos codificados en forma ASCII :

Tabla 16 - Equivalencias ASCII

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Por lo tanto si se envía el número 1 por el puerto serial, se recibirá el valor 49 ya que este es el valor que se le corresponde en la tabla ASCII.

[6] Unidad de medida usada en telecomunicaciones que representa el número de símbolos por segundo en un medio de transmisión digital.

Para enviar un número sin confusiones se debe definir que se está enviando un valor numérico entero (tipo decimal), por lo tanto se definirá así el código del emisor:

```
Serial.print(81,DEC); // Envía un 81
De igual manera se puede enviar un valor en hexadecimal (HEX), octal (OCT),
binario (BIN) o un byte donde no se usa la función Serial.print sino
Serial.write
Serial.print(81,HEX); // Envía un 51
Serial.print(81,OCT); // Envía un 121
Serial.print(81,BIN); // Envía un 1010001
Serial.write(81); // Envía una "Q". Es el equivalente en ASCII
```

Para recibir los datos en el otro Arduino se utiliza la función Serial.*read*. Este es el código necesario para leer los datos recibidos:

```
if(Serial.available()>0){ // Si hay datos en el buffer
       int dato = Serial.read(); //Guardamos el dato
       Serial.print(dato) // Imprimimos el valor
}
```

Si deseamos enviar una palabra, debemos crear un vector tipo *char* para que guarde cada letra en una posición del vector. También se debe usar la función *atoi()* que convierte la cadena de caracteres en números enteros y la función *memset()* para borrar el contenido del vector. Este sería el código utilizado:

```
char cadena[30];
byte posicion = 0;
int valor;
void loop(){
if(Serial.available()>0){
     memset(cadena,0,sizeof(cadena));
     while(Serial.available()>0){
         delay(5);
          cadena[posicion] = Serial.read();
          posicion++;
     }
     valor=atoi(cadena);
     posicion=0;
}
```

Una vez tenemos conocimiento de cómo conectar los dispositivos, éstas son las formas más comunes de conectarlos entre si:

Bluetooth

Los módulos *bluetooth* son la mejor forma de agregar una conectividad inalámbrica a los proyectos. Existen unos módulos llamados XBee[7] que solo hay que configurar y en pocos minutos la conexión está realizada.

Figura 38 - Módulo inalámbrico XBee

No utilizaremos este método debido a que estos dispositivos son bastante caros ya que, para conectarlos a la placa, se necesita un adaptador especial que encarece el precio y, además, no necesitamos conectividad inalámbrica entre los Arduinos, ya que el que controla el ajedrez y el que controla el temporizador estarán uno al lado del otro.

Cableado

Ésta es la forma más sencilla de conectar dos Arduinos. Lo único necesario es conectar el puerto 0 (TX) al puerto 1 (RX) del otro dispositivo y el puerto 1 (RX) al puerto 0 (TX) del otro. En la siguiente figura se muestra la conexión.

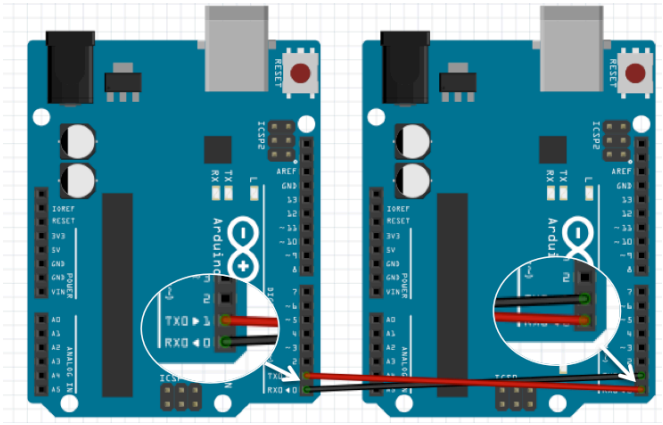


Figura 39 - Puerto serie cableado

[7] Basados en la comunicación Zigbee. Más información en www.xbee.cl

Es necesario conectar ambas tierras (GND) entre sí para tomar referencia a una tierra común. Las placas se comunicaran para:

- Una vez decidido el modo de juego (*joystick* o acelerómetro) se enviará por puerto serie el valor 0 si es *joystick* o 1 si es acelerómetro.
- Cuando el brazo robot empiece a moverse enviará el valor 2 para indicar que ha empezado el movimiento y una vez finalizado enviará el valor 3 para indicar que puede continuar la partida. Como el proyecto no abarca el diseño del código del brazo robot ni de su implementación en el tablero, una vez recibido el valor 2 por el puerto serie se detendrá el tiempo un periodo definido (unos 5 segundos) para permitir que el ayudante mueva la ficha por el tablero y a continuación se recibirá el valor 3 por el puerto serie para el cambio al jugador 2.

A pesar de enviar números enteros, la comunicación se realiza con caracteres ASCII por lo tanto las placas recibirán los valores 48, 49, 50 y 51 respectivamente.

Tabla 17 - Datos enviados y recibidos por puerto serie

Arduino con pantalla LCD	Arduino con matriz y brazo robot
Modo *Joystick* envía valor 0	Recibe valor 48
Modo Acelerómetro envía valor 1	Recibe valor 49
Recibe valor 50	Parar tiempo envía valor 2
Recibe valor 51	Iniciar tiempo envía valor 3

6. SOFTWARE

6.1. Arduino

La plataforma de Arduino se programa mediante el uso de un lenguaje propio basado en el lenguaje de programación de alto nivel *Processing.* Sin embargo, es posible utilizar otros lenguajes y aplicaciones populares para Arduino debido a que soporta la transmisión de datos por puerto serie utilizada en la mayoría de lenguajes. Está basado en la programación en C y soporta todas las funciones de ésta y algunas de C++. A continuación se muestra un resumen de las estructuras y sintaxis del lenguaje de Arduino:

- Delimitadores: { }
- Comentarios: //, /* */
- Cabeceras: #define, #include
- Operadores aritméticos: +, -, *, /, %
- Asignación : =
- Operadores de comparación: ==, !=, <, >, <=, >=
- Operadores Booleanos: &&, ||, !
- Operadores de acceso a punteros: *, &
- Operadores de bits: &, |, ^, <<,>>
- Operadores de incremento y decremento de variables: ++, --
- Operadores de asignación y operación: +=, -=, *=, /=, &=, !=

Las estructuras de control son las siguientes:

- Condicionales: *if, if…else, switch case*
- Bucles*: for, while, do…while*
- Bifurcaciones y saltos: *break, continue, return, goto*

Las constantes ya definidas en el código son éstas:

- ***HIGH/LOW***: Representan los niveles alto y bajo de las señales, los altos son a partir de los 3V.
- ***INPUT/OUTPUT***: Entradas o salidas.
- ***false***: Señal que representa al cero lógico, se escribe en minúscula.
- ***true***: Señal que representa al uno lógico, se escribe en minúscula.

Los tipos de datos que se pueden utilizar son los siguientes:

- ***void****:* vacío, no devuelve nada.
- ***boolean****:* del tipo *true/false*.
- ***char****:* un carácter.
- ***byte****:* almacena un número sin signo de 8-bit desde el 0 hasta el 255.

- ***int:*** valor entero guardado en 2 bytes. Entre el -32.768 hasta el 32.767.
- ***unsigned int:*** valor entero positivo entre el 0 y 65.535.
- ***long***: valor entero de 4 bytes del -2.147.483.648 al 2.147.483.647.
- ***unsigned long***: valor entero de 4 bytes del 0 al 4.294.967.295.
- ***float***: valor decimal $3{,}4028235 \cdot 10^{38}$ al $-3{,}4028235 \cdot 10^{38}$. Ocupan 4 bytes.
- ***double***: Ocupa 4 bytes y tiene la misma precisión que *float.*
- ***string***: Representan cadenas de caracteres que terminan con un *NULL*.
- ***array***: Matriz con una colección de variables donde se accede mediante un número de índice.

Las funciones básicas utilizadas en el proyecto son:

- Digitales:
  - **pinMode(pin, modo):** define un pin como entrada o salida.
  - **digitalWrite(pin, valor):** define un valor digital a un pin de salida.
  - **digitalRead(pin):** devuelve el valor digital del pin (0 o 1).
- Analógicas
  - **analogWrite(pin, valor):** define un valor analógico entre 0-255 a una salida PWM digital.
  - **analogRead(pin):** devuelve el valor analógico entre 0-1024.
- Tiempo
  - **delay(x) :** Detiene el programa "x" milisegundos.

Las funciones para comunicación con el puerto serie son:

- **begin(x):** Empieza la conexión a x baudios.
- **available():** Devuelve si el puerto serie está disponible.
- **read():** Lee lo que recibe por el puerto serie.
- **print():** Envía datos por el puerto serie.
- **println():** Envía datos y hace un intro por el puerto serie.

Por último éstas son las bibliotecas más utilizadas por Arduino:

- **EEPROM:** Controla la lectura y escritura de almacenamiento permanente.
- **Servo**: Controla servo motores.
- **LiquidCrystal**: Controla una pantalla LCD.
- **Stepper**: Controla motores paso a paso unipolares o bipolares.
- **Wire**: Envía y datos sobre dispositivos de dos cables I2C.

Programación de Arduino

Para programar una placa Arduino se necesita el *software* gratuito de la página web www.arduino.cc. El software está disponible para Windows, GNU/Linux y Macintosh.

El entorno de programación es el siguiente:

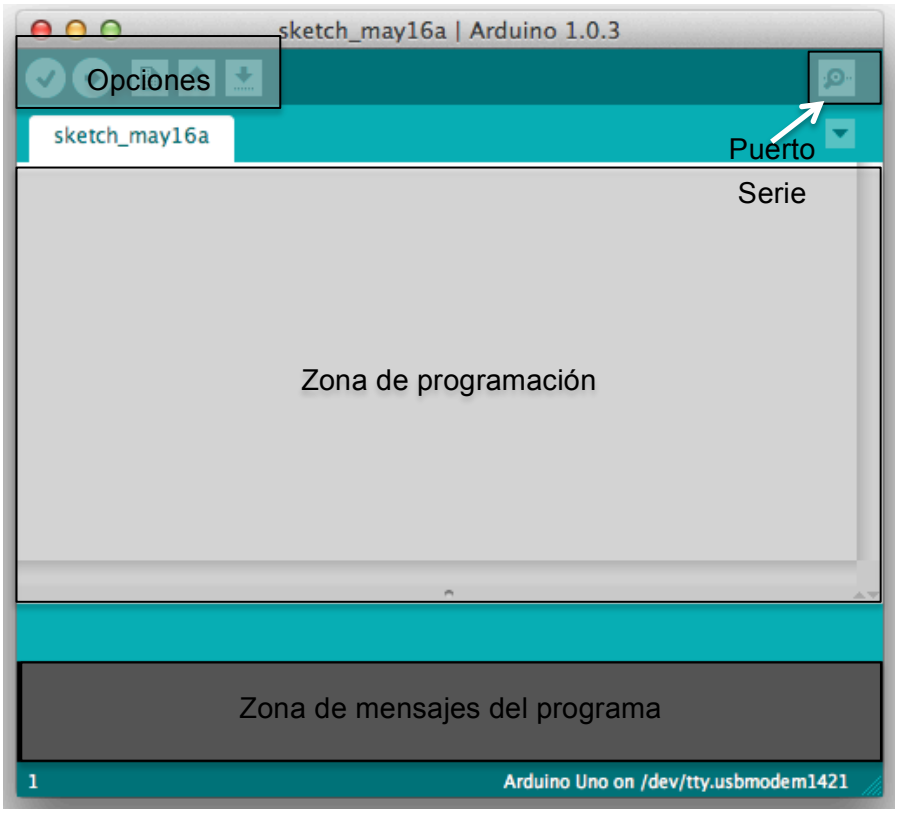
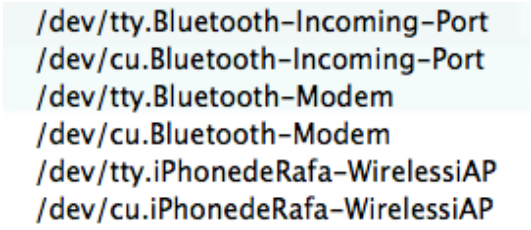


Figura 40 - Entorno de programación

- La zona de opciones dispone de los siguientes botones:
  - Verificar : Busca errores en el código, ya sean de sintaxis o variables que no se han definido.
  - Cargar: Envía el código a la placa conectada por USB.
  - Nuevo: Crea un nuevo documento de código.
  - Abrir: Abre un código ya existente.
  - Guardar: Guarda el código.
- La zona de programación recoge todo el código del programa.
- La zona de mensajes del programa muestra lo que está haciendo en cada momento. También muestra errores de sintaxis del código.
- El botón del puerto serie nos abre una ventana con los datos que envía o recibe el puerto.

Para empezar a programar se debe especificar que versión de la placa Arduino se ha conectado:

Por último se define el puerto al que se ha conectado la placa:



6.2. Librerías

Las librerías de Arduino sirven de ayuda para facilitar la comunicación con los diferentes *interfaces* que vamos a usar para el proyecto. Se han utilizado librerías para la matriz de diodos LED, para el mando Nunchuk, la librería *wire* necesaria para controlar el mando y una última para la pantalla LCD.

6.2.1. Librería LedControl

Esta librería de Arduino ha sido creada para controlar los circuitos integrados MAX7221 y MAX7219. Ambos sirven para controlar una matriz de 64 LED o *displays* de 7 segmentos y permite poner en cascada diferentes circuitos como en la figura siguiente:

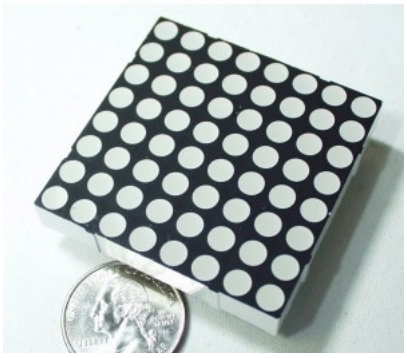


Figura 41 - Matriz LED 8x8

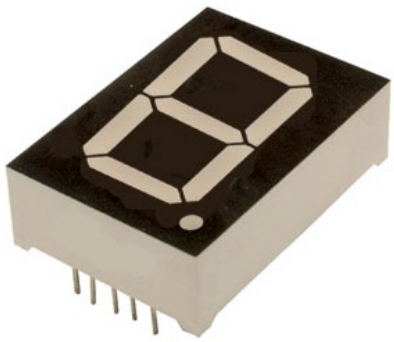


Figura 42 - *Display* de 7 segmentos

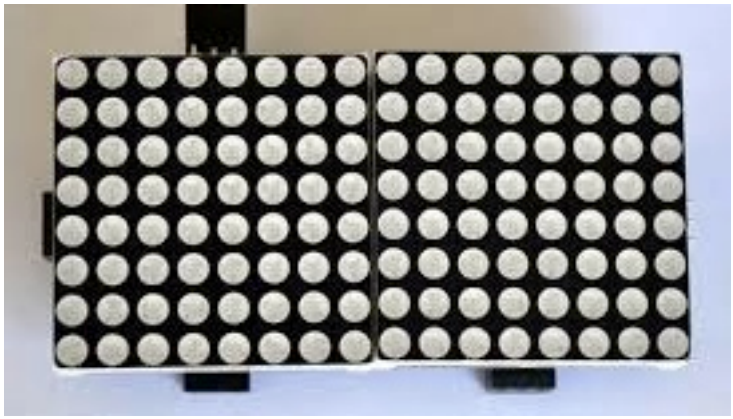


Figura 43 - Matriz LED en cascada

Instalar librería

Para instalar la librería de LedControl.h en nuestro entorno de programación debemos descargar el paquete de la web de Arduino y guardarlo en la carpeta raíz del programa. Esta carpeta raíz depende del sistema operativo del ordenador de trabajo:

Carpeta raíz en MAC : User/NOMBRE_USUARIO/Documents/Arduino/libraries

Carpeta raíz en WINDOWS: C:/Documents and settings/Administrator/Mis documentos/Arduino/libraries

Dentro del paquete encontramos estos archivos:

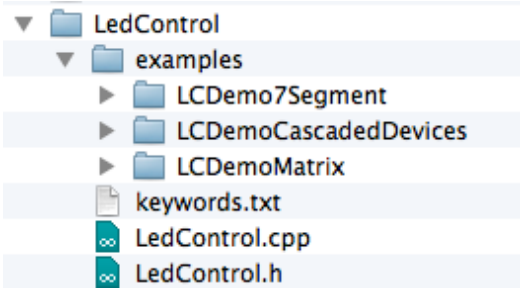


Figura 44 - Contenido LedControl.h

Una carpeta con tres ejemplos diferentes, y dos archivos. Los ejemplos son uno para un *display* de 7 segmentos, uno para diferentes matrices en cascada y el último para una única matriz. Los dos archivos son los encargados de almacenar la estructura de la librería.

Una vez añadida la carpeta en su directorio debemos cerrar y volver a abrir el entorno de programación de Arduino para que detecte que se ha añadido una nueva librería.

Definir la librería

Primero de todo se sebe incluir la librería en el código, a continuación se deben de definir las conexiones que se han realizado en Arduino.

Para nuestro proyecto la conexión ha sido la siguiente:

Tabla 18 - Conexión Matriz - Arduino

Pin Arduino	Pin Matriz
5V	VCC
GND	GND
2	DIN
3	CS
4	CLK

Se deberá crear un nuevo objeto para llevar el control de esa matriz y después definir donde está conectado el *data in* (DIN), el *clock* (CLK), el *chip select* (CS) y el numero de matrices que se conectan. El código quedará así:

```
#include <LedControl.h>
LedControl lc=LedControl(2,4,3,1);
```

- LedControl lc → Nos crea un objeto llamado "lc". Siempre que queramos referirnos a la matriz para enviarle algún comando será mediante las letras "lc".

- LedControl(2,4,3,1) → Define las conexiones de la matriz. El 2 define el DIN, el 4 el CLK, el 3 el CS y el 1 define que solo hay una matriz conectada, no hay matrices en cascada (el valor máximo es 8).

Si quisiéramos conectar otra matriz a Arduino (no en cascada, sino independiente) deberíamos definir nuevos CS, CLK y DIN diferentes de los pines 2, 4 y 3.

A continuación debemos definir una serie de comandos para inicializar la matriz de LED dentro del void setup().

```
void setup(){
lc.shutdown(0,false);
lc.setIntensity(0,8);
lc.clearDisplay(0);
}
```

- lc.shutdown(0, false) → Activa el MAX7219 en modo de ahorro de energía. Todos los LEDS brillaran al mínimo si hay alguno encendido.
- lc.setIntensity(0, 8) → Define la intensidad de brillo de los diodos, en este caso brillaran a la mitad de su limite ya que el máximo es 15.
- lc.clearDisplay(0) → Borra toda la matriz y deja todos los diodos apagados.

Funciones

A partir de aquí ya funcionará nuestra librería en el programa. Por último mostramos como se utiliza el comando para controlar los diodos independientemente:

- lc.setLed(0,row,col,true) → Permite activar/desactivar un LED enviando la fila, la columna y un *true* o *false* según necesitemos encender o apagar el diodo.

Solo se ha explicado la parte necesaria para el proyecto ya que esta librería también controla *displays* y matrices en cascada para reproducir letras o textos.

Licencia

Esta librería se distribuye bajo los términos de *General Menor GNU Public License version* 2.1.

6.2.2. Librería ArduinoNunchuk

Esta librería de Arduino ha sido creada para tratar los datos que llegan del mando Nunchuk, así como los valores del *joystick*, acelerómetro o los dos pulsadores.

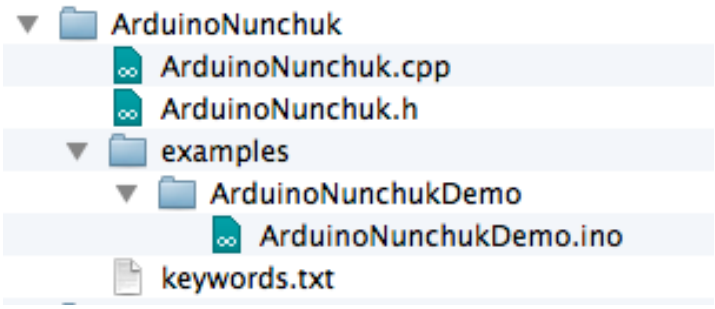
Instalar librería

Para instalar la librería de ArduinoNunchuk.h en nuestro entorno de programación debemos descargar el paquete de la web de Arduino y guardarlo en la carpeta raíz del programa. Esta carpeta raíz depende del sistema operativo del ordenador de trabajo:

Carpeta raíz en MAC : User/NOMBRE_USUARIO/Documents/Arduino/libraries

Carpeta raíz en WINDOWS: C:/Documents and settings/Administrator/Mis documentos/Arduino/libraries

Dentro del paquete encontramos estos archivos:



La carpeta incluye un ejemplo y los dos archivos necesarios para utilizar la librería. Los dos últimos ejemplos sirven para controlar un servo con el *joystick* o con el acelerómetro. El ejemplo ArduinoNunchukDemo.ino lo utilizaremos para leer los datos de los sensores ya que muestra por el puerto serie todos los datos de todos los sensores. A continuación se muestra un ejemplo de lo que se lee por el puerto serie:

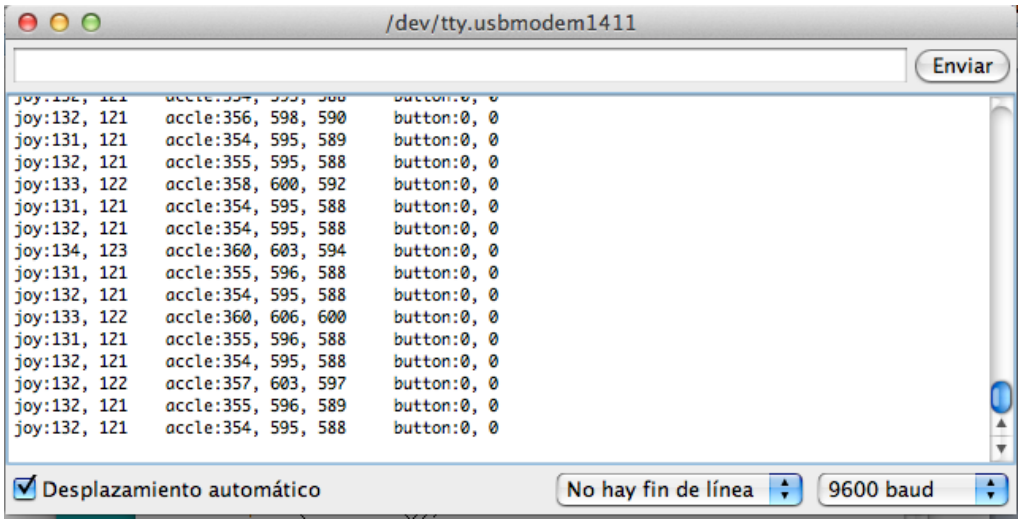


Figura 45 - Datos Nunchuk

Nos muestra:

- La posición de los potenciómetros (*joy*: x, y).
- La posición del acelerómetro (*accle*: x, y, z).
- El estado de los botones (*button*: C, Z).

Definir la librería

El primer paso en todas las librerías es incluirla en las primeras líneas de código. A continuación se harán las conexiones tal y como se definieron en el apartado de funcionamiento de Nunchuk.

Tabla 19 - Conexión Nunchuk-Arduino

Pin Arduino	Pin Nunchuk
A4	1
-	2
5V	3
A5	4
-	5
GND	6

```
#include <ArduinoNunchuk.h>
ArduinoNunchuk wii = ArduinoNunchuk();
```

Este es el código que se ha utilizado para definir el mando Nunchuk, solo se debe escribir el comando ArduinoNunchuk seguido del nombre que queramos ponerle al dispositivo dentro del entorno de programación, en nuestro caso se ha llamado "wii".

Debemos definir algún parámetro más dentro del void setup() :

```
void setup(){
wii.init();
}
```

A continuación se inicializa el mando con la función wii.init(), ahora la placa ya está preparada para recibir datos del mando.

Cuando iniciemos el void loop() utilizaremos la función wii.update() que actualiza los valores de los sensores.

Funciones

Ahora ya se puede utilizar el mando Nunchuk en nuestro programa, solo falta definir las funciones que utilizaremos para leer los valores de cada sensor.

- wii.analogX → Lee el valor del potenciómetro X, un valor entre 0 y 255.
- wii.analogY → Lee el valor del potenciómetro Y, un valor entre 0 y 255.

- wii.accelX → Lee el valor del acelerómetro X, un valor entre 0 y 1023.
- wii.accelY → Lee el valor del acelerómetro Y, un valor entre 0 y 1023.
- wii.accelZ → Lee el valor del acelerómetro Z, un valor entre 0 y 1023.

- wii.cButton() → Lee el estado del pulsador C, toma el valor 0 o 1.
- wii.zButton() → Lee el estado del pulsador Z, toma el valor 0 o 1.

6.2.3. Librería Wire

Esta librería de Arduino sirve para comunicar los dispositivos I^2C/TWI, es muy simple de utilizar, solo se debe de saber que en la mayoría de placas Arduino el SDA o línea de datos está en el pin analógico 4 (A4) y el SCL o línea de reloj está en el pin analógico 5 (A5). Hay que tener en cuenta que en Arduino Mega se encuentran en el pin 20 y 21 respectivamente.

Instalar librería

Esta librería viene ya instalada en el entorno de programación Arduino por defecto ya que es muy utilizada para comunicar diferentes dispositivos con la placa.

Definir la librería

Solo debemos definir la librería al principio del código y ya está listo para ser usado.

```
#include <Wire.h>
```

Funciones

La función más utilizada de esta librería es *begin* para empezar la conexión con los dispositivos que estén conectados por I^2C, como el mando Nunchuk.

6.2.4. Librería Servo

Esta librería de Arduino ha sido creada para controlar la posición de los servo motores mediante sencillas funciones.

Instalar librería

Esta librería viene ya instalada en el entorno de programación Arduino por defecto ya que es muy utilizada en todos los proyectos.

Definir la librería

Solo debemos definir la librería al principio del código y ya está listo para ser usado.

```
#include <Servo.h>
Servo miservo;
```

La función Servo crea una nuevo servo motor con el nombre miservo.

Debemos definir algún parámetro más dentro del void setup() :

```
void setup(){
miservo.attach(9); }
```

La función attach define en el pin que está conectado el servo motor (pin PWM).

Funciones

Las funciones más utilizadas en esta librería son las siguientes:

```
miservo.write(posición);
miservo.read();
```

Donde *write* coloca el servo en la posición que definamos, este valor ira de 0 a 179. Y la función *read* devuelve el ángulo actual del servo.

6.2.5. Librería LiquidCrystal

Esta biblioteca permite a la placa Arduino controlar *displays* LCD basados en el *chipset* Hitachi HD44780 o compatibles. Gracias a esta librería se pueden controlar la mayoría de pantallas del mercado.

Instalar librería

No es necesario instalar esta librería ya que viene instalada de serie con el software de Arduino.

Definir la librería

El primer paso en todas las librerías es incluirla en las primeras líneas de código. A continuación se harán las conexiones tal y como se definieron anteriormente.

Tabla 20 - Conexión Arduino - pantalla LCD

Pin Pantalla LCD	Pin Arduino
1	GND
2	+5V
3	Centro potenciómetro
4	7
5	GND
6	8
7-10	-
11	9
12	10
13	11
14	12
15	5V
16	GND

Una vez conectado de este modo se define en la primera línea de código la librería para poderla utilizar.

```
#include <LiquidCrystal.h>
```

Acto seguido se declaran las conexiones y se define una nueva pantalla que la llamaremos "lcd".

```
LiquidCrystal lcd (7, 8, 9, 10, 11, 12);
```

Estos son los parámetros en orden : RS , ENABLE (E), D4, D5, D6, D7.

Debemos definir algún parámetro más dentro del void setup() :

```
void setup(){
lcd.begin(20,4);
lcd.clear();
}
```

La primera línea define el tamaño de la pantalla (columnas, filas), la segunda línea borra todos los datos de la pantalla.

En este punto ya se puede utilizar la pantalla para mostrar mensajes, a continuación se explican las funciones más utilizadas de la librería LiquidCrystal.h:

- lcd.clear() → Borra todos los datos de la pantalla.
- lcd.setCursor(columna,fila) → Coloca el cursor donde vamos a escribir. Los valores de las columnas van del 0 al 19 y los valores de las filas van del 0 al 3.

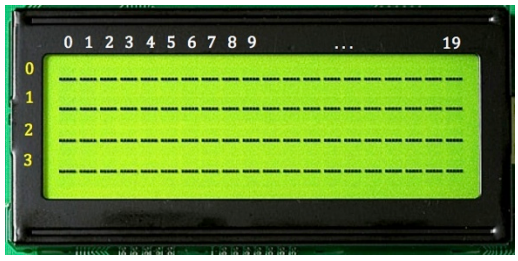


Figura 46 – Coordenadas del LCD

- lcd.print("mensaje") → Escribe donde este colocado el cursor el mensaje entre comillas.
- lcd.createChar(num,data) → Crea caracteres especiales definidos previamente con vector del tipo *byte.* El numero es el identificador del carácter creado (del 0 al 7) y el *data* es el nombre con el que se ha definido anteriormente el vector. A continuación se muestra un ejemplo:

```
byte barra[8] = {
       B11111,
       B00000,
       B00000,
       B00000,
       B00000,
       B00000,
       B00000,
       B00000};
```

Es realmente sencillo definir un carácter en binario, cada BXXXXXX define una fila de un carácter de la pantalla y cada valor identifica si este pixel está encendido o apagado.

Existen numerosas webs como quinapalus.com donde te ayudan a definir los caracteres de forma visual, basta con pintar encima del carácter los píxel que queremos encendidos y la aplicación devuelve el código en binario.

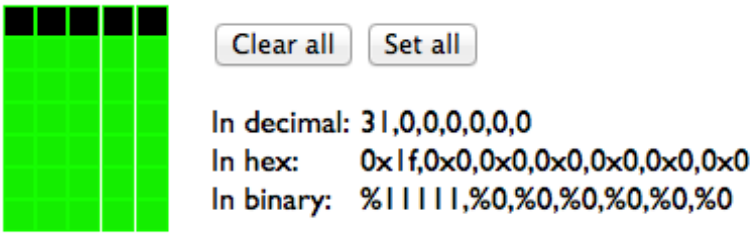


Figura 47 - Definir carácter LCD visualmente

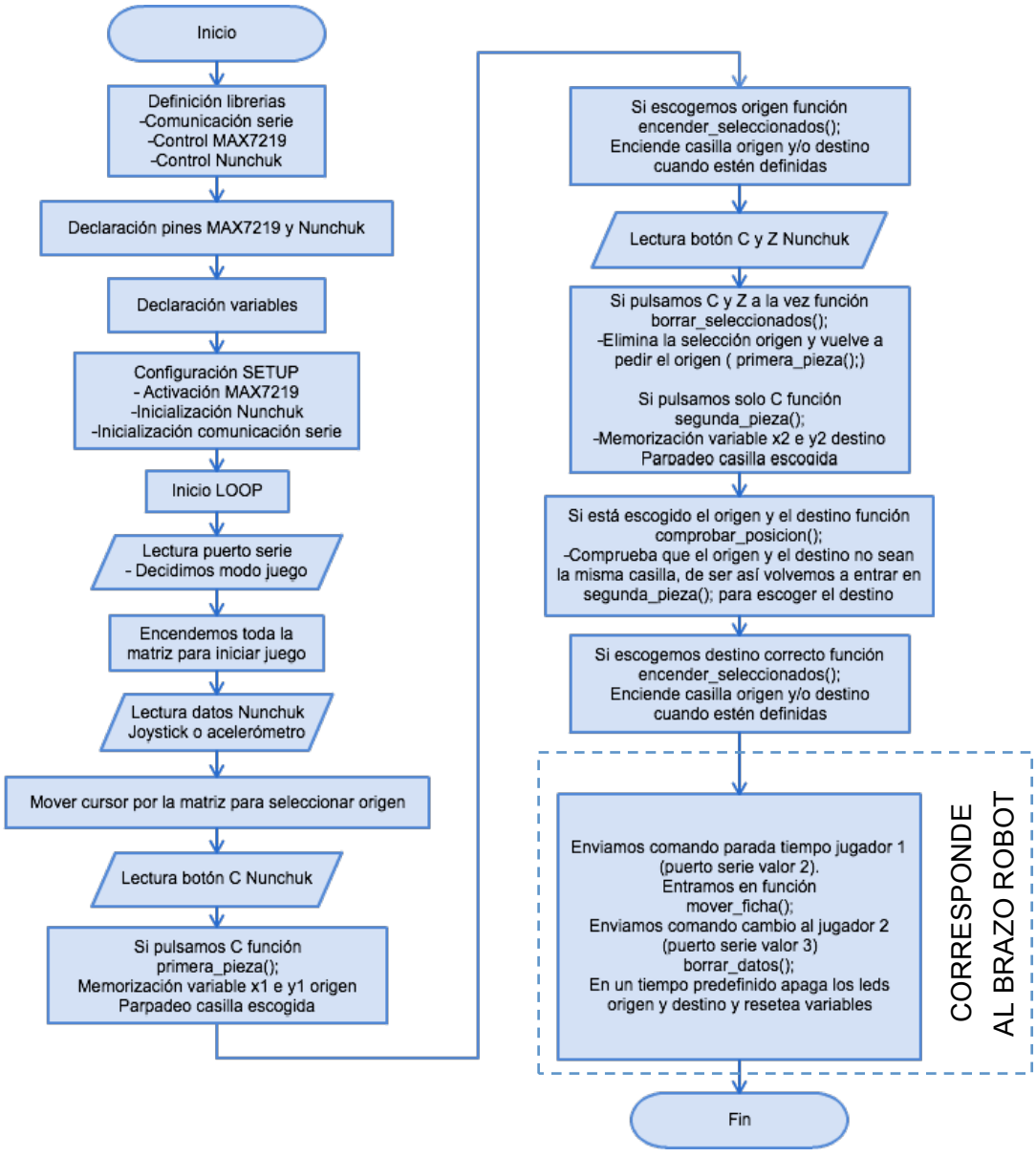
Una vez generado el vector solo hay que definirlo en el *setup* del programa:

```
lcd.createChar(1,barra);
```

6.3. Diagramas de flujo simples y explicación del código

En los siguientes apartados se definirán los diagramas de flujo simples y los códigos de cada Arduino.

6.3.1. Diagrama simple de la matriz LED y Nunchuk



Para calibrar la sensibilidad del mando a distintos jugadores se debe modificar el código de la placa de Arduino. Las variables a modificar son para el *joystick*: jderecha, jizquierda, jarriba, jabajo, y para el acelerómetro: aderecha, aizquierda, aarriba, aabajo.

6.3.2. Explicación del código de la matriz LED y Nunchuk

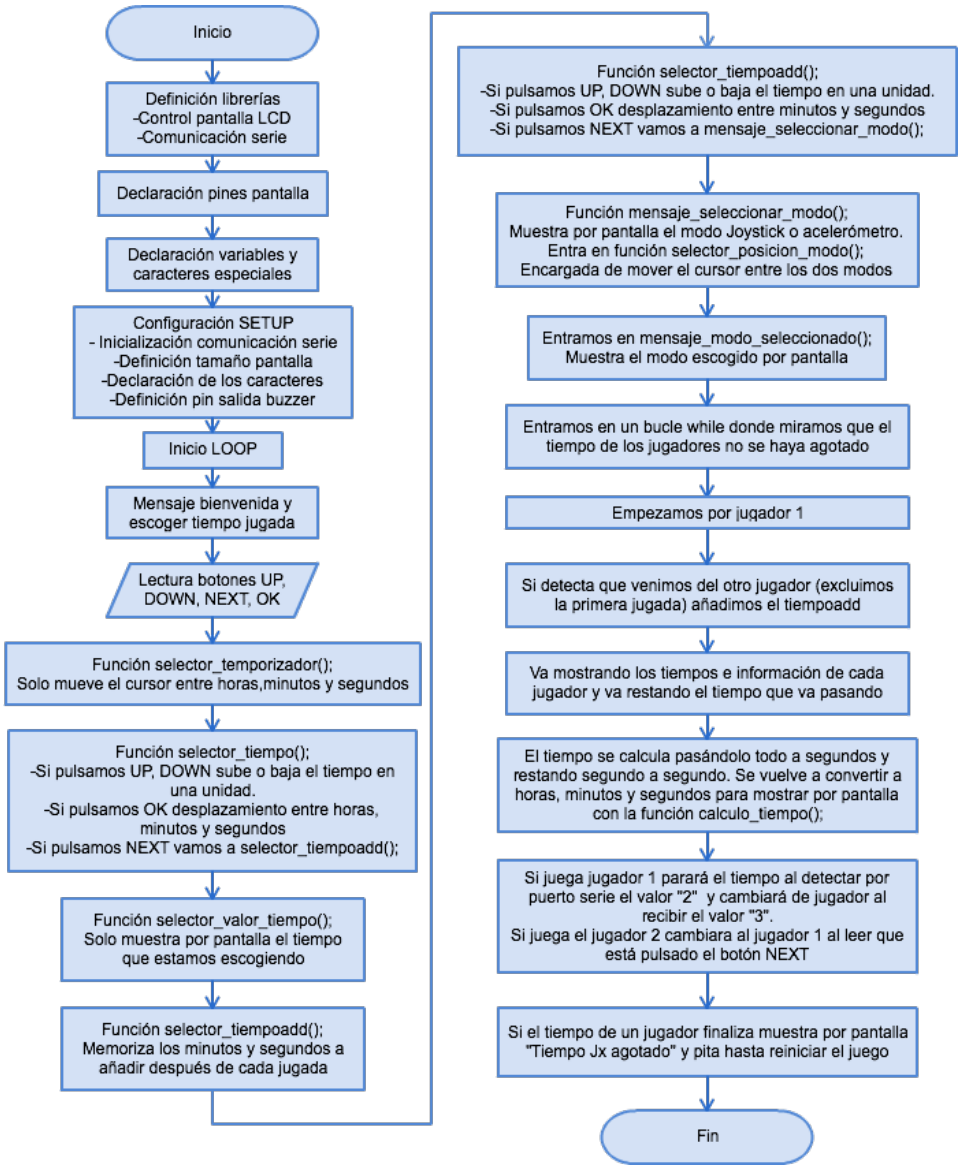
```
#include <LedControl.h> //Librería control del multiplexor MAX7219
#include <Wire.h> //Librería control de la comunicación
#include <ArduinoNunchuk.h> //Librería control del mando Nunchuk

LedControl lc=LedControl(2,4,3,1); //Declaración de los pines del MAX7219
ArduinoNunchuk wii = ArduinoNunchuk(); //Definimos el mando como wii

boolean seq_inicio=true; //Enciende todos los leds al comenzar
int joyx=100,joyy=100,accelx,accely; //guardan valores del joystick y
acelerómetro
int modo=0;
int x=0, y=0 ,x1=-1, x2=-1, y1=-1, y2=-1, destino=0; //Guardan la posición
actual, posición inicial y posición final
int botonC, botonZ; //Guardan el estado de los botones del Nunchuk
byte incomingbyte; //Almacena el modo de juego recibido por puerto serie
boolean ini_selec=false, fin_selec=false; //true cuando seleccionamos pieza
inicial y pieza final
int figura_movida; //Guarda la figura que vamos a mover
int inicio=1; //Evitara leer el puerto serie todas las veces

//*****************************************************************************
//***********VALORES MODIFICABLES PARA AJUSTAR SENSIBLIDAD MANDO**************
//*****************************************************************************
int jderecha=210,jizquierda=50,jarriba=210,jabajo=30;
int aderecha=550,aizquierda=450,aarriba=550,aabajo=450;
//*****************************************************************************
//*****************************************************************************

void setup(){ //Configuración SETUP
  lc.shutdown(0,false); //Enciende la matriz de leds
  lc.setIntensity(0,8); //Define la intensidad de la matriz a 8
  lc.clearDisplay(0); //Borra todos los leds si hay encendidos

  wii.init(); //Iniciamos el mando Nunckuk

  Serial.begin(19200); //Iniciamos a comunicación a 19200 baudios

} //Fin SETUP

void loop(){ //Configuración LOOP

while(inicio==1){
if(Serial.available()){ //Si recibe datos por el puerto serie entramos
    incomingbyte= Serial.read(); //Guardamos el dato en incomingbyte
    Inicio=0;
    }
}
  wii.update(); //Actualizamos valores del mando
  while(seq_inicio==true){ //Enciende todos los leds del tablero
        for(int col=0;col<8;col++){
          for(int row=0;row<8;row++){
            delay(100);
       lc.setLed(0,row,col,true); //Va encendiendo cada led
          }
        }
        lc.clearDisplay(0); //Apaga todos los leds
        seq_inicio=false; //Sale de la condición if
 }
     obtener_datos(); //Función encargada de guardar valores el Nunchuk
    if(incomingbyte==48){ //Si recibe por serie el 48 el modo es Joystick
```

```
    if(joyx>jderecha){ //Joystick a la derecha
      lc.setLed(0,x,y,false); //apaga el led encendido
      if(x<7){ //Si estamos en cualquier casilla menos la última
        x++; //Añadimos uno a x
        lc.setLed(0,x,y,true); //Encendemos el led de la derecha
      }else{ //Si se trata de la columna 8, enciende el de la columna 0
        x=0;
        lc.setLed(0,x,y,true);
      }
    }
if(joyx<jizquierda ){  // Izquierda ídem
      lc.setLed(0,x,y,false);
      if(x>0){
        x--;
        lc.setLed(0,x,y,true);
      }else{
        x=7;
        lc.setLed(0,x,y,true);
      }
    }
    if(joyy>jarriba){ //Arriba ídem
      lc.setLed(0,x,y,false);
      if(y<7){
        y++;
        lc.setLed(0,x,y,true);
      }else{
        y=0;
        lc.setLed(0,x,y,true);
      }
    }
if(joyy<jabajo){  //Abajo ídem
      lc.setLed(0,x,y,false);
      if(y>0){
        y--;
        lc.setLed(0,x,y,true);
      }else{
        y=7;
        lc.setLed(0,x,y,true);
      }
    }
  }
  if(incomingbyte==49){ //Si recibe por serie el 49 el modo es acelerómetro
    delay(500);
     if(accelx>aderecha){ //Derecha ídem
      lc.setLed(0,x,y,false);
      if(x<7){
        x++;
        lc.setLed(0,x,y,true);
      }else{
        x=0;
        lc.setLed(0,x,y,true);
      }
    }
    if(accelx<aizquierda){  //Izquierda ídem
      lc.setLed(0,x,y,false);
      if(x>0){
        x--;
        lc.setLed(0,x,y,true);
      }else{
        x=7;
        lc.setLed(0,x,y,true);
```

```
      }
    }

    if(accely>aarriba){ //Arriba ídem
      lc.setLed(0,x,y,false);
      if(y<7){
        y++;
        lc.setLed(0,x,y,true);
      }else{
        y=0;
        lc.setLed(0,x,y,true);
      }
    }

    if(accely<aabajo){  //Abajo ídem
      lc.setLed(0,x,y,false);
      if(y>0){
        y--;
        lc.setLed(0,x,y,true);
      }else{
        y=7;
        lc.setLed(0,x,y,true);
      }
    }
  }
obtener_datos(); //Obtiene datos del Nunchuk
primera_pieza(); //Función para escoger primera pieza
encender_seleccionados(); //Enciende la ficha a mover inicial

obtener_datos(); //Vuelve a obtener datos por si queremos anular la selección
borrar_seleccionados();  //Si pulsamos C y Z anula la selección

obtener_datos();  //Vuelve a obtener datos
segunda_pieza(); //Escoge la segunda pieza
encender_seleccionados();  //Enciende el led del destino
comprobar_posicion(); //Comprueba que no son iguales el origen y el destino
} //Fin LOOP


void primera_pieza(){
  if(botonC==true && ini_selec==false && fin_selec==false){ //Si pulsamos el
botón C y no hay piezas aún seleccionadas
      x1=x; //Guardamos la posición de la pieza a mover
      y1=y;
      for(int c=0; c<5;c++){ //Parpadeo de la casilla
      lc.setLed(0,x1,y1,true);
      delay(100);
      lc.setLed(0,x1,y1,false);
      delay(100);
      ini_selec=true; //Fin de la función
      }
     }
  }
void encender_seleccionados(){ //Enciende las casillas origen y destino
  if(ini_selec==true || fin_selec==true){
      lc.setLed(0,x1,y1,true);
      lc.setLed(0,x2,y2,true);
  }
}
void borrar_seleccionados(){ //Función elimina la casilla seleccionada
if(botonC==true && botonZ==true){
    lc.clearDisplay(0); //Borra la matriz de leds entera
```

```
      x1=-1; //Reset de las variables a un valor inexistente en la matriz
      x2=-1;
      y1=-1;
      y2=-1;
      ini_selec=false; //Para volver a entrar a la función de escoger origen
      fin_selec=false; //Para volver a entrar a la función de escoger destino
      delay(1000);
}
  }
void segunda_pieza(){ //Escoge casilla destino
  if(botonC==true && ini_selec==true && fin_selec==false){ //Si pulsamos el
botón C y ya se ha escogido el origen pero no el destino entra

       x2=x; //Guarda las coordenadas para encender el led
       y2=y;
       for(int c=0; c<5;c++){ //Parpadeo de la casilla
       lc.setLed(0,x2,y2,true);
       delay(100);
       lc.setLed(0,x2,y2,false);
       delay(100);
       fin_selec=true; //Sale de la condición
      }
     }
}
void comprobar_posicion(){ //Comprueba que la casilla de destino no es la misma
que la de origen
  if (fin_selec==true){ //Si se ha elegido ya el destino
     if(x1==x2 && y1==y2){ //Si el origen es igual al destino
       fin_selec=false; //Deja entrar en la función escoge destino
     }else{  //Si son distintas coordenadas
     Serial.print(2); //Envia al otro arduino la orden de parar el temporizador
del jugador 1
      delay(2000);
 // AQUÍ IRIA EL PROGRAMA DEL BRAZO ROBOT
      Serial.print(3);   //Envia al otro arduino la orden de cambiar el jugador
      borrar_datos(); //Apaga todos los leds
      }
     }
}

void obtener_datos(){ //Obtiene estado de botones, joystick y acelerometro
    wii.update();
    botonC=wii.cButton;
    botonZ=wii.zButton;
    joyx=wii.analogX;
    joyy=wii.analogY;
    accelx=wii.accelX;
    accely=wii.accelY;
}
void borrar_datos(){ //Apaga los leds en 2 segundos
  for(int cnt=2; cnt>-1;cnt--){
        delay(1000);
        Serial.println(cnt);
       }
        lc.clearDisplay(0); //Apaga leds
        x1=-1;
        x2=-1;
        y1=-1;
        y2=-1;
        ini_selec=false;
        fin_selec=false;
}
```

6.3.3. Diagrama simple del temporizador



6.3.4. Explicación del código del temporizador

```
#include <Wire.h>
#include <LiquidCrystal.h>

LiquidCrystal lcd(7, 8, 9, 10, 11 , 12); //Definición pines pantalla

int msg=0; //variable mensaje de bienvenida
boolean seq_inicio=true;
int botonUP, botonDOWN,botonOK,botonNEXT; //guardan estado botones
int selector_posicion=0, selector_modo; //Cursor posición 0
int parpadeo1=0,parpadeo2=0,parpadeo3=0; //ayuda para parpadeo cursor
int updown,updownmodo; //define si sube o baja el joystick Y
int ahoras=0, aminutos=0, asegundos=0, ahoras2=0, aminutos2=0, asegundos2=0;
//guardan las horas, minutos y segundos de los jugadores
int aminutosadd=0,asegundosadd=0; //guarda minutos y segundos añadidos
int segundostotal, segundostotal2, segundostotaladd, segundostotal2add; //guarda
tiempo jugadores y añadido en segundos
int tiempo_seleccionado=0, modo_seleccionado=0,tadd_seleccionado=0,
fin_configuracion=0; //fin de cada configuración
int modo=2; //Inicializamos diferente de 0 y 1
int buzzer=6; // Pin del buzzer
int jugadoractivo=0,jugadoractivoantiguo=0; //guarda el jugador activo y el
antiguo
int beepJ1=0,beepJ2=0; //variable para sonido de aviso cambio jugador
int x=0, y=0; //cursor del led encendido
int finpartidaJ1=0, finpartidaJ2=0; //Vale 1 fin del tiempo de un jugador
int botoncambio=0, botonchange=5; //botonchange es el mismo que el boton NEXT en
el pin 5
int estado; //Memoriza el estado del pulsador NEXT para cambiar del jugador 2 al
1
byte flecha[8] = {
    B11111,
    B00000,
    B00000,
    B00000,
    B00000,
    B00000,
    B00000,
    B00000
}; //simbolo cursor

byte flecha2[8]={
  B00000,
  B00100,
  B00010,
  B00001,
  B00010,
  B00100,
  B00000,
}; //símbolo izquierdo selector joystick o acelerometro

byte flecha3[8]={
  B00000,
  B00100,
  B01000,
  B10000,
  B01000,
  B00100,
  B00000,
}; //símbolo derecho selector joystick o acelerómetro
```

```
void setup(){ //CONFIGURACION SETUP
  Serial.begin(19200); //Velocidad de transmisión de datos

  lcd.begin(20, 4); //Configura el largo y ancho de caracteres lcd
  lcd.clear(); //Borra la pantalla si tiene algún carácter

  lcd.createChar(1,flecha); //Crea los símbolos
  lcd.createChar(3,flecha2);
  lcd.createChar(4,flecha3);

  pinMode(buzzer, OUTPUT); //Define el pin del buzzer como salida

}

void loop(){ //CONFIGURACION DEL LOOP

  if(msg==0){ //Mensaje de bienvenida
       lcd.setCursor(3,0);
       lcd.print("BIENVENIDOS AL");
       lcd.setCursor(1,2);
       lcd.print("AJEDREZ ADAPTADO");
       delay(2000);
       msg = 1; //Sale de la condicion if
       lcd.clear(); //Borra pantalla
     }

   botonUP=digitalRead(2); //Lectura botones UP DOWN y OK
   botonDOWN=digitalRead(3);
   botonOK=digitalRead(4);

   if (tiempo_seleccionado==0){ //Condición para seleccionar tiempo
     mensaje_seleccionar_tiempo(); //Función definida abajo

     selector_temporizador(); //Función definida abajo

     selector_tiempo(); //Función definida abajo

     selector_valor_tiempo(); //Función definida abajo
   }

   if(tiempo_seleccionado==1 && tadd_seleccionado==0){ //Una vez seleccionado el
tiempo se escoge tiempo añadido

    selector_tiempoadd(); //Función definida abajo

   }

   if(tadd_seleccionado==1 && modo_seleccionado==0){ //Una vez seleccionado el
tiempo añadido se escoge el modo de juego
    mensaje_seleccionar_modo(); //Función definida abajo

    selector_posicion_modo(); //Función definida abajo

   }

  if(modo_seleccionado==1 && fin_configuracion==0){ //Una vez configurado el
modo de juego se muestra por pantalla el modo

    mensaje_modo_seleccionado(); //Función definida abajo
    delay(2000);
    fin_configuracion=1;
    lcd.clear(); //Borra pantalla
```

```
    beep(); //Función definida abajo
  }
  while (segundostotal > 0 && segundostotal2 >0 && fin_configuracion==1) {
//Mientras el tiempo de los jugadores sea diferente de 0

    while(jugadoractivo==0 && segundostotal>0){ //Si juega el jugador 1
        if(jugadoractivoantiguo!=jugadoractivo){ //Si venimos del otro jugador
añade el tiempo añadido, esto evita añadir nada más comenzar
          segundostotal2=segundostotal2+segundostotal2add;
          jugadoractivoantiguo=jugadoractivo; //Guarda el jugador antiguo y sale
de la condición if

        }

        lcd.setCursor(2,0); //Coloca el cursor y muestra el mensaje
        lcd.print("Turno JUGADOR 1");
        if(beepJ1==0){ //Hace uno o dos pitidos según a quien le toca
          beep();
          beepJ2=0; //Sale de la condición if
          beepJ1=1;
        }

        //wii.update();
        delay(1000); //Descontamos en periodos de 1 segundo
        lcd.clear(); //Borra pantalla para actualizar tiempo
        segundostotal--; //Resta un segundo al tiempo
        calculo_tiempo(); //Función definida abajo

if(Serial.available()){//Si recibimos datos por serie

          estado=Serial.read();//Memorizamos el dato obtenido

          if(estado==50){ //Si recibimos el valor 50 (el otro arduino manda un 2
para parar el tiempo mientras mueve ficha)
            while(estado==50){
              if(Serial.available()){ //Si recibe otro dato por el puerto serie
                estado=Serial.read(); //Recibirá el valor 51 (un 3 del otro
arduino) cuando haya movido la ficha
              }
            }
            jugadoractivo=1; //Cambio jugador cuando ya se ha movido la ficha

        }
        }

        if( segundostotal==0){ //Si acaba el tiempo del jugador fin partida
          finpartidaJ1=1;
        }

     }

   while(jugadoractivo==1 && segundostotal2>0){ //Todo idéntico para jugador 2
       botoncambio=digitalRead(botonchange); //lee el estado del boton de cambio
de jugador de 2 a 1.



      if(jugadoractivoantiguo!=jugadoractivo){
        segundostotal=segundostotal+segundostotaladd;
        jugadoractivoantiguo=jugadoractivo;

      }
```

```
      if(botoncambio==0){ //Si el botón NEXT es pulsado cambia al jugador 1
        jugadoractivo=0;
      }else{ //sino sigue con el jugador 2
        jugadoractivo=1;
      }


      lcd.setCursor(2,0);
      lcd.print("Turno JUGADOR 2");
      if(beepJ2==0){
        beep();
        delay(500);
        beep();
        beepJ2=1;
        beepJ1=0;
      }
      delay (1000);
      lcd.clear();
      segundostotal2--;
      calculo_tiempo();

      if( segundostotal2==0){
        finpartidaJ2=1;
      }
   }
 }
    if( finpartidaJ1==1){ //Si acaba tiempo jugador 1 muestra mensaje y pita
  lcd.clear();
  lcd.setCursor(1,0);
  lcd.print("TIEMPO J1 AGOTADO");

  lcd.setCursor(1,2);
  lcd.print("FIN DE LA PARTIDA");
  beep();
  }
    if( finpartidaJ2==1){ //Si acaba tiempo jugador 2 muestra mensaje y pita
      lcd.clear();
      lcd.setCursor(1,0);
      lcd.print("TIEMPO J2 AGOTADO");

      lcd.setCursor(1,2);
      lcd.print("FIN DE LA PARTIDA");
      beep();
    }

} //FIN LOOP


void selector_temporizador(){ //Solo mueve el cursor
   botonUP=digitalRead(2);  //Lectura de los botones
   botonDOWN=digitalRead(3);
   botonOK=digitalRead(4);

   if(botonOK==0){ //Mueve el cursor al que estemos configurando
     selector_posicion++;
     delay(250);
   }
   if(selector_posicion==3){ //Si estamos en los segundos y pulsamos vuelve a
las horas
     selector_posicion=0;
   }
```

```
   switch(selector_posicion){ //Según la posición muestra el cursor parpadenado
    case 0:
            if(parpadeo1==0){
            for(int contador=0; contador<20; contador++)
            {
              lcd.setCursor(contador,3);
              lcd.print(" ");
            }
            parpadeo1=1;
            parpadeo2=0;
            parpadeo3=0;
            }
            lcd.setCursor(6,3);
            lcd.write(1); //Muestra símbolo 1
            lcd.write(1);
            break;
   case 1:
             if(parpadeo2==0){
             for(int contador=0; contador<20; contador++)
            {
              lcd.setCursor(contador,3);
              lcd.print(" ");
            }
             }
            parpadeo1=0;
            parpadeo2=1;
            parpadeo3=0;
            lcd.setCursor(9,3);
            lcd.write(1);
            lcd.write(1);
            break;
   case 2:
             if(parpadeo3==0){
             for(int contador=0; contador<20; contador++)
            {
              lcd.setCursor(contador,3);
              lcd.print(" ");
            }
             }
            parpadeo1=0;
            parpadeo2=0;
            parpadeo3=1;
            lcd.setCursor(12,3);
            lcd.write(1);
            lcd.write(1);
            break;
  }

}



void selector_tiempo(){ //Selecciona el tiempo de jugada
        delay(200);
        botonUP=digitalRead(2); //Lectura botones
        botonDOWN=digitalRead(3);
        if(botonUP==0 && selector_posicion==0) //Si el boton ha sido pulsado,
aumentamos las horas en una unidad
             {
              ahoras = ahoras + 1 ;
              delay(500);
             }
```

```
        if(botonDOWN==0 && ahoras>0 && selector_posicion==0) //Si el botón ha
sido pulsado, disminuimos las horas en una unidad
        {
          ahoras=ahoras-1;
          delay(500);
        }

        if(botonUP==0 && selector_posicion==1) //Si el botón ha sido pulsado,
aumentamos los minutos en una unidad
       {
         aminutos = aminutos + 1;
         delay(250);
       }
       if(botonDOWN==0 && aminutos>0 && selector_posicion==1) //Si el botón ha
sido pulsado, disminuimos los minutos en una unidad
          {
            aminutos=aminutos-1;
            delay(250);
          }


          if(botonUP==0 && selector_posicion==2) //Si el botón ha sido pulsado,
aumentamos los segundos en una unidad
       {
         asegundos = asegundos + 1;
         delay(250);
       }
       if(botonDOWN==0 && asegundos>0 && selector_posicion==2) //Si el botón ha
sido pulsado, disminuimos los segundos en una unidad
          {
            asegundos=asegundos-1;
            delay(250);
          }

}


void mensaje_seleccionar_tiempo(){ //Muestra instrucciones pantalla
       lcd.setCursor(0,0);
       lcd.print("Seleccionar tiempo:");
       lcd.setCursor(1,1);
       lcd.print("DESPL=OK ACEPT=NEXT");
}

void selector_valor_tiempo(){ // Muestra el tiempo por pantalla en hh:mm:ss
      lcd.setCursor(6,2);

      if (ahoras < 10) lcd.print("0");    // Si las horas son menor que 10, pone
un "0" delante.
       lcd.print(ahoras);       // Sin este código, se muestra así: H:M:S
(1:M:S)
       lcd.print(":");

      if (aminutos < 10) lcd.print("0");  // Si los minutos son menor que 10,
pone un "0" delante.
       lcd.print(aminutos); // Sin este código, se muestra así: H:M:S  (H:1:S)

       lcd.print(":");
      if (asegundos < 10) lcd.print("0"); // Si los segundos son menor que 10,
pone un "0" delante.
       lcd.print(asegundos); // Sin este código, se muestra así: H:M:S  (H:M:1)
```

```
       segundostotal = asegundos + (aminutos * 60) + (ahoras * 60 * 60);
//Convierte el tiempo elegido en segundos
       segundostotal2= segundostotal; //Los copia para el jugador 2


       botonNEXT=digitalRead(5);
       if(botonNEXT==0){ //Si pulsamos NEXT vamos al siguiente menú
         tiempo_seleccionado=1;
         botonNEXT==1;
         lcd.clear();
       }
}


void selector_tiempoadd(){ //Escoger tiempo añadido función idéntica a la
anterior
   delay(200);
   lcd.setCursor(0,0);
   lcd.print("Selecc tiempo add:");  //Muestra mensaje y las HH:MM:SS que
vayamos aumentando
   lcd.setCursor(1,1);
   lcd.print("DESPL=OK ACEPT=NEXT");


   botonUP=digitalRead(2);
   botonDOWN=digitalRead(3);
   botonOK=digitalRead(4);

   if(botonOK==0){
     selector_posicion++;
     delay(250);
   }
   if(selector_posicion==2){
     selector_posicion=0;
   }
   switch(selector_posicion){
    case 0:
            if(parpadeo1==0){
            for(int contador=0; contador<20; contador++)
            {
              lcd.setCursor(contador,3);
              lcd.print(" ");
            }
            parpadeo1=1;
            parpadeo2=0;
            parpadeo3=0;
            }
            lcd.setCursor(6,3);
            lcd.write(1);
            lcd.write(1);
            break;
   case 1:
             if(parpadeo2==0){
             for(int contador=0; contador<20; contador++)
            {
              lcd.setCursor(contador,3);
              lcd.print(" ");
            }
             }
            parpadeo1=0;
            parpadeo2=1;
```

```
            parpadeo3=0;
            lcd.setCursor(9,3);
            lcd.write(1);
            lcd.write(1);
            break;

  }


    botonUP=digitalRead(2);
    botonDOWN=digitalRead(3);
    if(botonUP==0 && selector_posicion==0) //Si el botón ha sido pulsado,
aumentamos las horas en una unidad
         {
          aminutosadd = aminutosadd + 1 ;
          delay(500);
         }
    if(botonDOWN==0 && aminutosadd>0 && selector_posicion==0)
    {
      aminutosadd=aminutosadd-1;
      delay(500);
    }

    if(botonUP==0 && selector_posicion==1){ //Si el botón ha sido pulsado,
aumentamos los minutos en una unidad
     asegundosadd = asegundosadd + 1;
     delay(250);
   }
   if(botonDOWN==0 && asegundosadd>0 && selector_posicion==1)
      {
        asegundosadd=asegundosadd-1;
        delay(250);
      }



      lcd.setCursor(6,2);

  if (aminutosadd < 10) lcd.print("0"); // Si las horas son menor que 10, pone
un "0" delante.
   lcd.print(aminutosadd); // Sin este código, se muestra así: H:M:S  (1:M:S)
   lcd.print(":");

  if (asegundosadd < 10) lcd.print("0");  // Si los minutos son menor que 10,
pone un "0" delante.
   lcd.print(asegundosadd);// Sin este código, se muestra asi: H:M:S  (H:1:S)

   segundostotaladd = asegundosadd + (aminutosadd * 60);  //Convierte el tiempo
elegido en segundos
   segundostotal2add= segundostotaladd;

   botonNEXT=digitalRead(5);
   if(botonNEXT==0){ //Si pulsamos NEXT vamos al siguiente menú
     tadd_seleccionado=1;
     botonNEXT==1;
     lcd.clear();
   }
}
void mensaje_seleccionar_modo(){ //Mensaje seleccionar modo
       lcd.setCursor(1,0);
       lcd.print("Seleccionar modo:");
       lcd.setCursor(3,1);
```

```
       lcd.print("ACEPTAR=NEXT");
       lcd.setCursor(6,2);
       lcd.print("Joystick");
       lcd.setCursor(4,3);
       lcd.print("Acelerometro");
}

void selector_posicion_modo(){ //Mueve la flecha de selección
       delay(200);
       botonUP=digitalRead(2);
       botonDOWN=digitalRead(3);
       botonNEXT=digitalRead(5);
        if(botonUP==0 || botonDOWN==0){
             selector_modo++;
             delay(500);
   }

   if(selector_modo==2){ //Si estamos en el modo acelerómetro bajando podemos
acceder al modo joystick
     selector_modo=0;
   }
   switch(selector_modo){
    case 0:
            if(parpadeo1==0){

              lcd.setCursor(2,2);
              lcd.print(" ");
              lcd.setCursor(2,3);
              lcd.print(" ");
              lcd.setCursor(17,2);
              lcd.print(" ");
              lcd.setCursor(17,3);
              lcd.print(" ");
            }
            parpadeo1=1;
            parpadeo2=0;

            lcd.setCursor(2,2);
            lcd.write(3);
            lcd.setCursor(17,2);
            lcd.write(4);
            break;
   case 1:
             if(parpadeo2==0){
              lcd.setCursor(2,2);
              lcd.print(" ");
              lcd.setCursor(2,3);
              lcd.print(" ");
              lcd.setCursor(17,2);
              lcd.print(" ");
              lcd.setCursor(17,3);
              lcd.print(" ");

             }
            parpadeo1=0;
            parpadeo2=1;
            lcd.setCursor(2,3);
            lcd.write(3);
            lcd.setCursor(17,3);
            lcd.write(4);
            break;
  }
```

```
  if(botonNEXT==0){ //Si pulsamos NEXT vamos al siguiente menú
    modo_seleccionado=1;
    lcd.clear();
  }else{
    modo_seleccionado=0;
  }

}


void mensaje_modo_seleccionado(){ //Muestra el modo seleccionado por pantalla
   lcd.setCursor(0,0);
   lcd.print("Modo seleccionado:");
   if(selector_modo==0){
     Serial.print(0);
     lcd.setCursor(6,2);
     lcd.print("JOYSTICK");
   }
   if(selector_modo==1){
     Serial.print(1);
     lcd.setCursor(4,2);
     lcd.print("ACELEROMETRO");
   }
}

void beep(){ //Hace un pitido de 0,5s
        digitalWrite(buzzer,HIGH);
        delay(500);
        digitalWrite(buzzer,LOW);
        delay(500);
}

void calculo_tiempo(){ //Convierte el tiempo de segundos a h,m,s
  ahoras = ((segundostotal / 60)/ 60);   //Convertimos los segundos totales en
horas
  aminutos = (segundostotal / 60) % 60;  //Convertimos los segundos totales en
minutos
  asegundos = segundostotal % 60;        //Convertimos los segundos totales en
periodos de 60 segundos

  lcd.setCursor(3,1);
  lcd.print("Tiempo restante:");        //Mostramos mensaje de tiempo restante
  lcd.setCursor(0,2);
  lcd.print("JUGADOR 1");
  lcd.setCursor(0,3);
  if (ahoras < 10) lcd.print("0");     // Si las horas son menor que 10, pone un
"0" delante
  lcd.print(ahoras);                   // Sin este código, se muestra asi: H:M:S
(1:M:S)
  lcd.print(":");
  if (aminutos < 10) lcd.print("0");   // Si los minutos son menor que 10, pone
un "0" delante.
  lcd.print(aminutos); // Sin este codigo, se muestra asi: H:M:S  (H:1:S)
  lcd.print(":");
  if (asegundos < 10) lcd.print("0");  // si el valor de segundo esta por debajo
de 9 (unidad) antepone un cero
  lcd.print(asegundos);
  lcd.setCursor(11,2);
  lcd.print("JUGADOR 2");
  ahoras2 = ((segundostotal2 / 60)/ 60);   //Convertimos los segundos totales en
horas
```

```
  aminutos2 = (segundostotal2 / 60) % 60;  //Convertimos los segundos totales en
minutos
  asegundos2 = segundostotal2 % 60; //Convertimos los segundos totales en
periodos de 60 segundos
  lcd.setCursor(12,3);
  if (ahoras2 < 10) lcd.print("0");     // Si las horas son menor que 10, pone
un "0" delante.
  lcd.print(ahoras2);                   // Sin este código, se muestra asi:
H:M:S  (1:M:S)
  lcd.print(":");
  if (aminutos2 < 10) lcd.print("0");   // Si los minutos son menor que 10, pone
un "0" delante.
  lcd.print(aminutos2);                 // Sin este código, se muestra asi:
H:M:S  (H:1:S)
  lcd.print(":");
  if (asegundos2 < 10) lcd.print("0");  // si el valor de segundo esta por
debajo de 9 (unidad) antepone un cero
  lcd.print(asegundos2);
}
```

7. PROCESO DE DISEÑO Y FABRIACIÓN

7.1. Diseño y fabricación de la matriz de LED

La matriz LED estará formada por 64 diodos colocados en ocho filas y ocho columnas. Según las características del tablero de cristal la separación entre diodos será de 3,7 cm tanto en horizontal como en vertical y estarán dispuestos sobre un tablero de madera de 3mm de grosos perforado con 64 agujeros de 5 mm de diámetro. Cada diodo irá centrado dentro de cada casilla y el esquema es el siguiente:

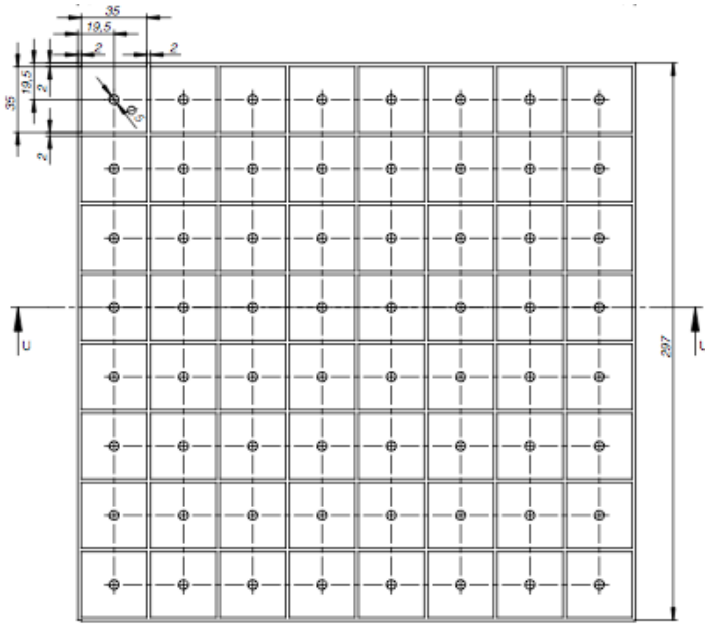


Figura 48 - Esquema del tablero perforado

La conexión de los diodos, como ya se ha explicado antes es la siguiente: las columnas para los ánodos y las filas para los cátodos (según las especificaciones del integrado MAX7219). Los diodos tienen las siguientes características:

Tabla 21 - Características diodos LED

Características	Valor
Color	Azul
Diámetro	5 mm
Intensidad luminosa	10.000 mcd
Intensidad nominal	20 mA
Tensión nominal	3,2 V
Vida útil	100.000 horas

Teniendo en cuenta que la alimentación de los diodos es de 5V aproximadamente se calculará la resistencia necesaria para no dañar los componentes:

$$R = \frac{V}{I} = \frac{5{,}00\ V - 3{,}20\ V}{0{,}02\ A} = 90\ \Omega$$

La resistencia que se utilizará será de 100 Ω para asegurar que no sobrepasamos la intensidad nominal de cada diodo.

El primer paso es colocar todos los diodos dentro de su agujero en el tablero que los contiene y una vez colocados se irán soldando primero todos los ánodos por columnas y después todos los cátodos por filas.

Para que la soldadura sea eficaz se deben poner en contacto las patillas a soldar y calentar con el soldador hasta que, acercando el estaño, se funda por si solo sin tener contacto con el soldador, de este modo evitamos soldaduras frías que podrían romperse con facilidad.

Una vez finalizado el proceso de soldadura, se deben soldar los cables que unirán las filas y columnas con el integrado MAX7219. En la siguiente imagen se observan los cables verdes que conectan la matriz con el integrado:

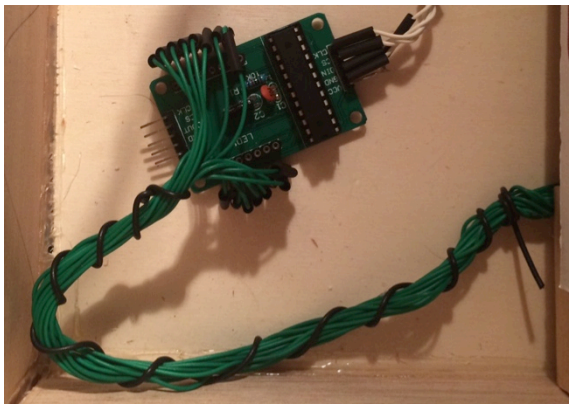


Figura 49 - Conexión de la matriz LED con el integrado MAX7219

A continuación se diseñarán unas separaciones entre casilla y casilla para evitar que los diodos iluminen fuera de su casilla, el diseño es el siguiente:

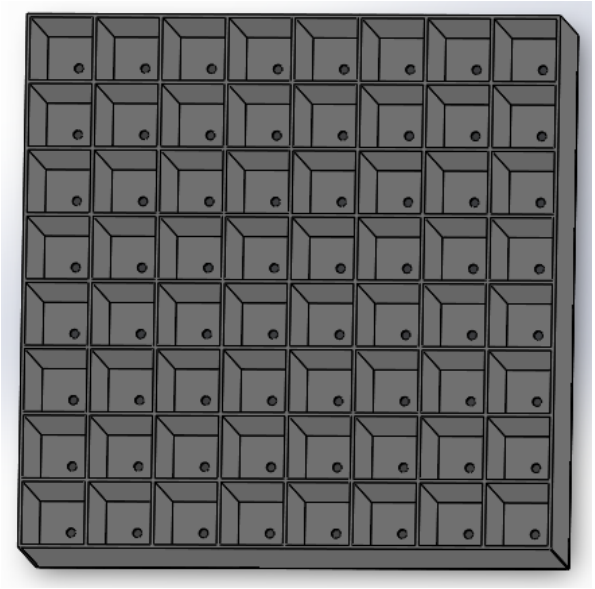


Figura 50 - Rejilla de separación de los diodos LED

Esta rejilla se construirá con cartón para que el tablero tenga un acabado más ligero. Se utilizará cartón blanco para una mayor reflexión de la luz con las paredes y conseguir una mayor iluminación de la casilla.

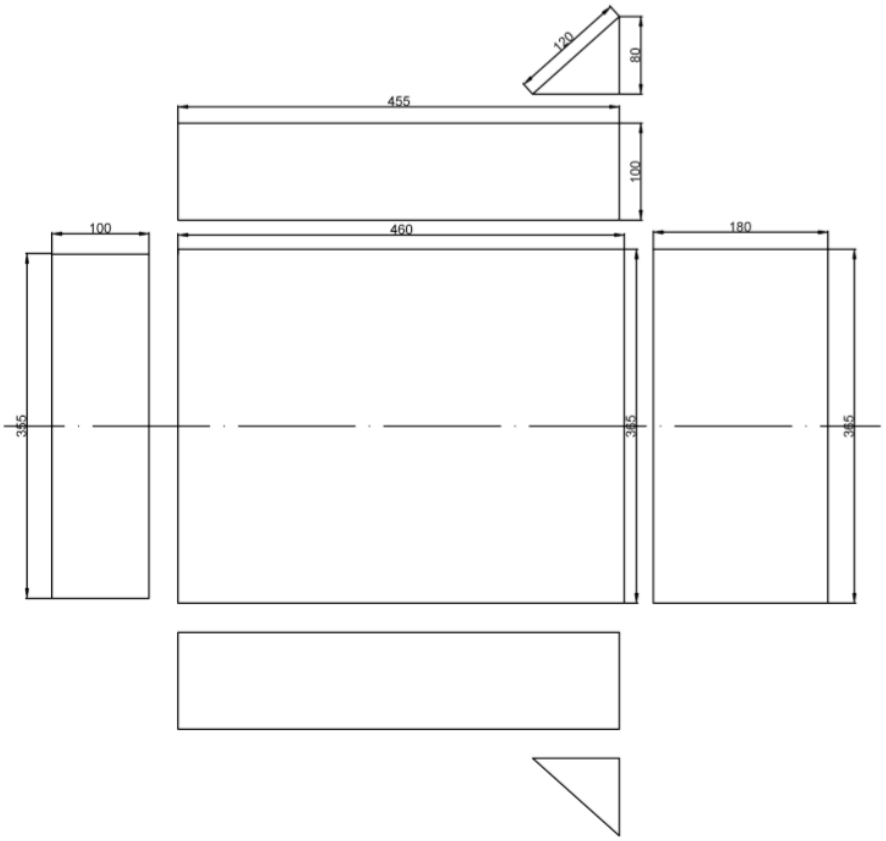
Se ha escogido una altura de 7 cm para la rejilla y así poder iluminar toda la casilla entera con un diodo.

Para diseñar la rejilla verticales se utilizarán tiras de cartón de 7 cm de alto y de 29,6 cm de largo (ancho y largo del tablero). Para las rejillas horizontales se cortarán trozos de cartón de 7 cm de alto y 3,5 cm de ancho.

Por último, para ver la casilla iluminada, se colocará papel vegetal o un papel normal en la parte superior de la rejilla para que la luz ilumine la superficie blanca del papel.

7.2. Diseño y fabricación del tablero

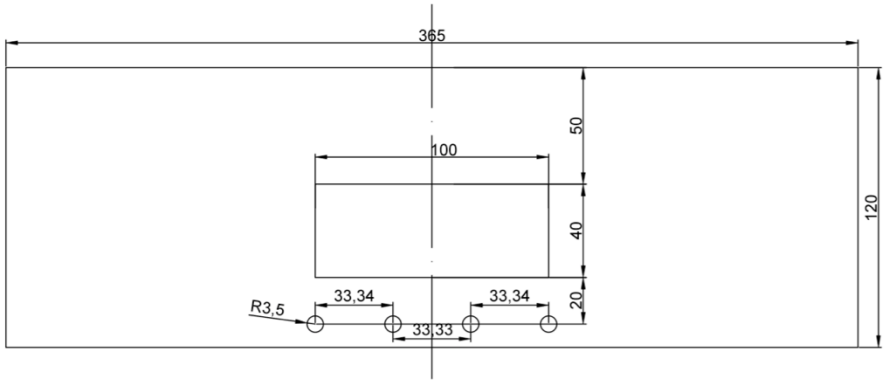
Se empezará construyendo la base del tablero con madera contrachapada de 3 mm de espesor, un ancho de 365 mm y 460 mm de largo. Después se colocaran las paredes del tablero construidas en madera contrachapada de 5 mm de espesor con un alto de 10 cm para poder introducir dentro la matriz de LED. A continuación se detallan los planos de diseño y fabricación del tablero de ajedrez.



La pantalla LCD y los cuatro botones irán colocados en la parte inclinada del tablero para tener una mejor visualización por parte de los dos jugadores.

El orden de los botones será de izquierda a derecha: UP – DOWN – OK – NEXT.

Es importante colocar el botón NEXT a la derecha del todo ya que el jugador 2 pasará de turno cuando mantenga pulsado el botón NEXT y oiga el pitido de cambio de jugador. A continuación se muestra un plano simplificado del panel que contiene la pantalla y los botones (el grosor de la madera es de 5 mm) :



El tamaño de la pantalla es de 100 mm de largo por 40 mm de ancho y los botones tienen un diámetro de 7 mm.

Ésta será la forma final de la caja que contendrá todo el conjunto:

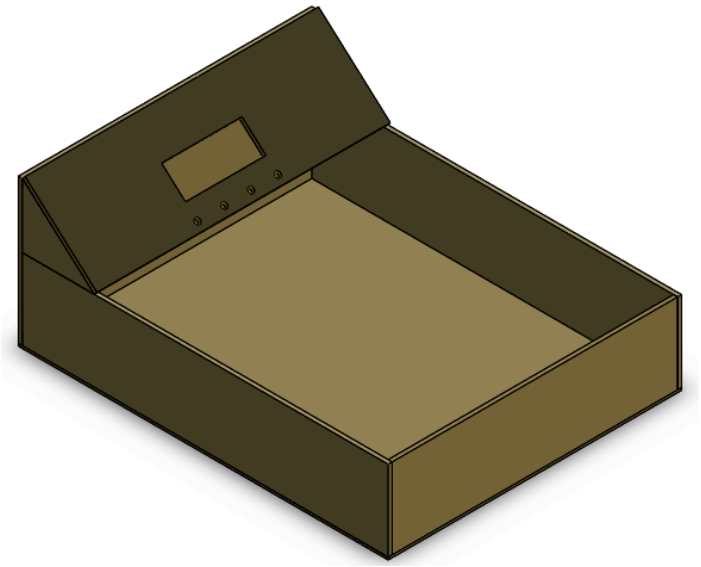


Figura 51 - Cuerpo del ajedrez

Todo el cuerpo se ha encolado con silicona termo fusible debido a la velocidad de secado de ésta.

La matriz preparada anteriormente irá encajada y debe quedar de la siguiente manera:

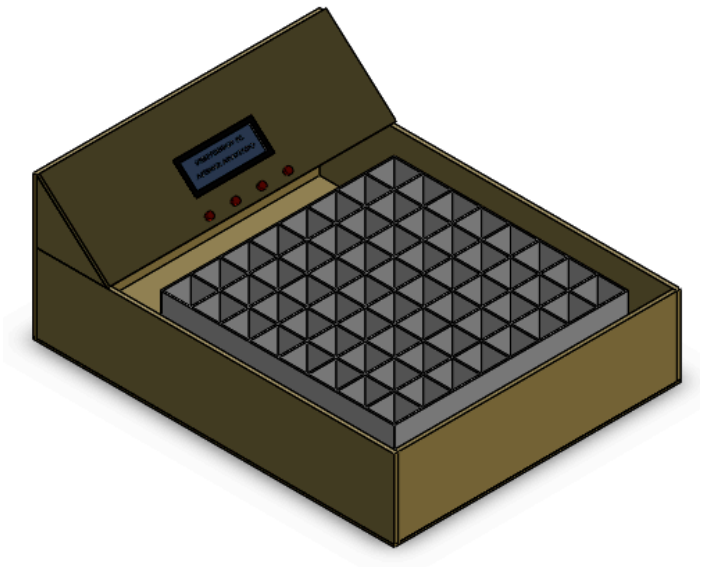


Figura 52 - Cuerpo del ajedrez con la matriz

7.3. Diseño del brazo robot

Para diseñar el brazo robot necesitaremos construir la base donde irá montado todo el brazo y la pinza para coger las fichas.

El diseño final será muy similar a la siguiente reproducción en 3D:

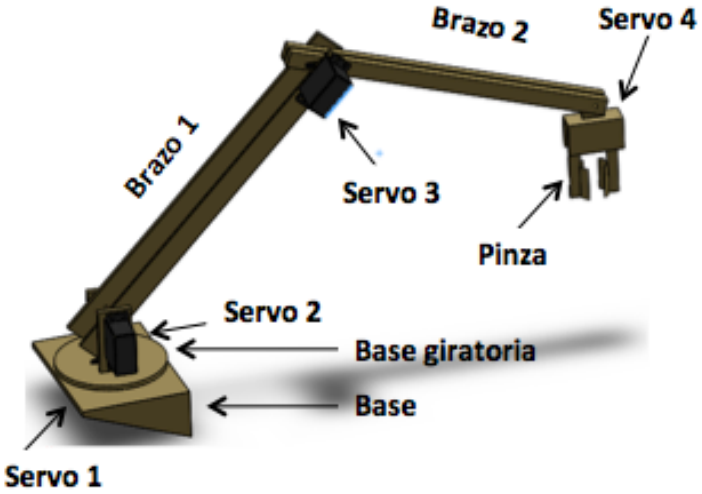


Figura 53 - Diseño del brazo robot

Primero se montará la superficie donde irá el primer servomotor que permitirá el giro del brazo. Esta pieza mide 100 mm de ancho por 140 mm de largo y 5 mm de espesor y el eje del servomotor irá centrado, sobre esta superficie irán dos resaltos hechos con madera de 3 mm de grosor para ajustar la base giratoria:

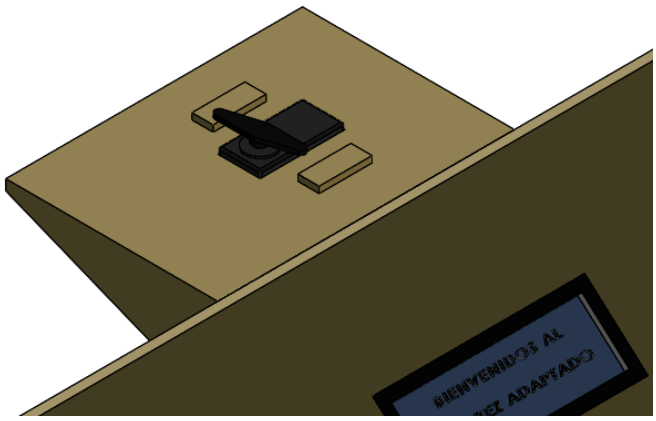


Figura 54 - Servomotor 1 montado

A continuación se acopla la base giratoria de 100 mm de diámetro y 5 mm de espesor. Encima irán montadas dos placas de madera de 30 mm de ancho por 60 mm de alto y de 5mm de espesor donde se montará el segundo servomotor que controlará el ángulo del primer brazo:

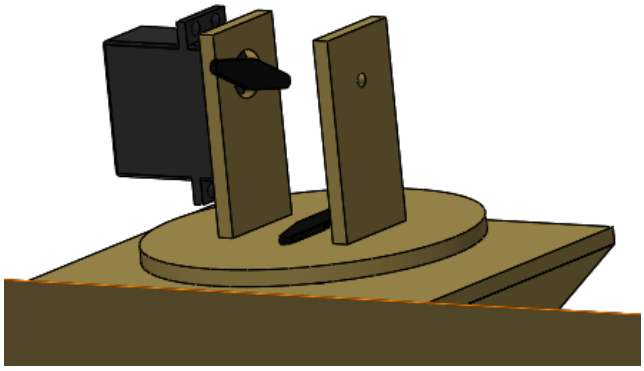


Figura 55 - Base giratoria y soporte servomotor 2

Acto seguido se montan las dos partes del primer brazo, uno cuya sujeción es un tornillo de M4 con una tuerca autoblocante y el otro irá pegado sobre el accesorio que se acopla al eje del servomotor (el servo también irá encolado con silicona a su soporte) :

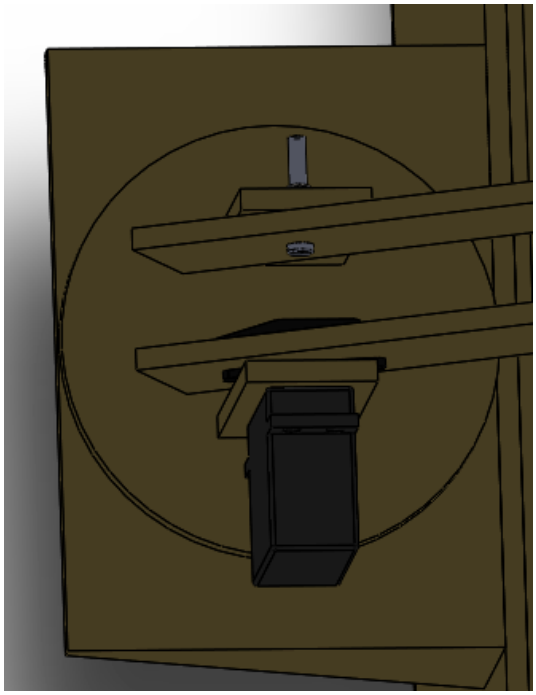


Figura 56 - Sujeción primer brazo

En el otro extremo del brazo que contenía el segundo servomotor se pega con silicona termo fusible el tercer servo y en el accesorio del eje se encola una pieza del segundo brazo. En el otro extremo del brazo se atornilla, con un tornillo del M4 y otra rosca autoblocante, la otra pieza del segundo brazo:

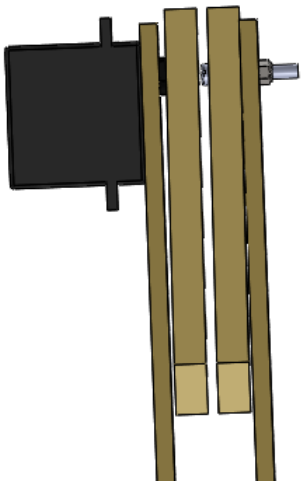


Figura 57 - Sujeción segundo brazo y servomotor 3

En el extremo del segundo brazo se atornilla el cabezal de la pinza dejando que el giro sea libre y sin fricción para que siempre esté orientado hacia abajo:

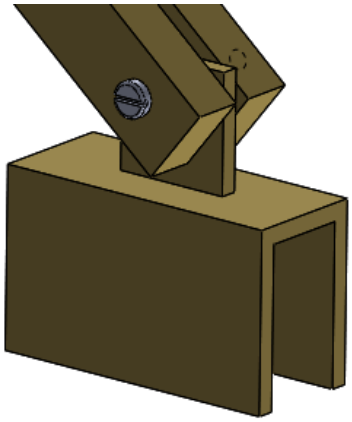
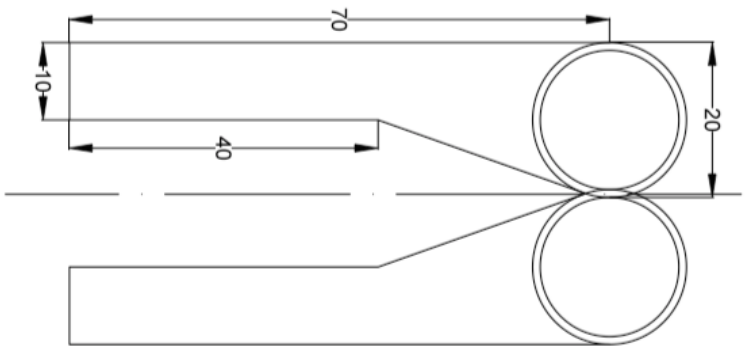


Figura 58 - Sujeción cabezal pinza

Por último para diseñar la pinza utilizaremos un servomotor de poca fuerza para abrir y cerrar y dos engranajes para transmitir el movimiento. La pinza tendrá estas medidas:



La parte circular irá pegada sobre los engranajes para abrir y cerrar las dos extremidades a la vez, sobre el eje de una de ellas irá acoplado el servomotor que controlará el ángulo de apertura:

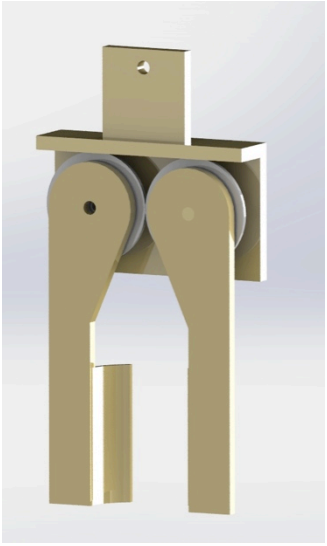


Figura 59 - Detalle del mecanismo de la pinza

Este es el resultado del ajedrez con el brazo montado:

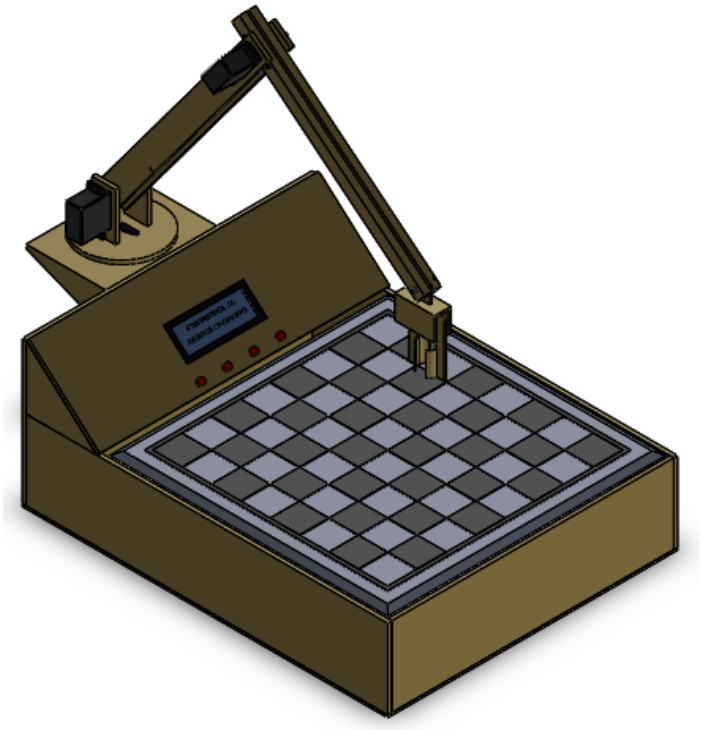


Figura 60 - Resultado final del ajedrez montado (3D)

7.4. Diseño y fabricación de los circuitos

Para la fabricación de los circuitos se pueden utilizar placas vírgenes para soldadura, donde el acabado es muy profesional pero el precio es más elevado o placas para soldar ya perforadas que tienen un aspecto menos profesional pero el precio de las placas es más reducido y para un prototipo es suficiente.

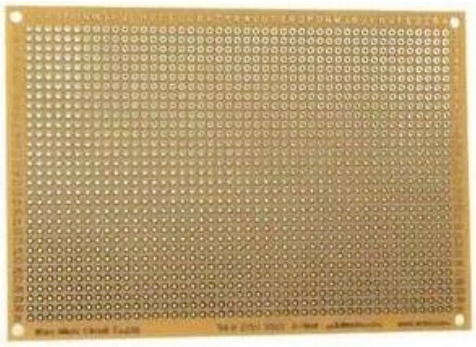


Figura 61 - Placa perforada para soldar

Circuito del temporizador

La placa del circuito del temporizador deberá contener los siguientes elementos:

- *Display* LCD.
- Cuatro botones para escoger las opciones del juego.
- Un potenciómetro para el contraste de la pantalla LCD.
- Buzzer.
- Cuatro resistencias.

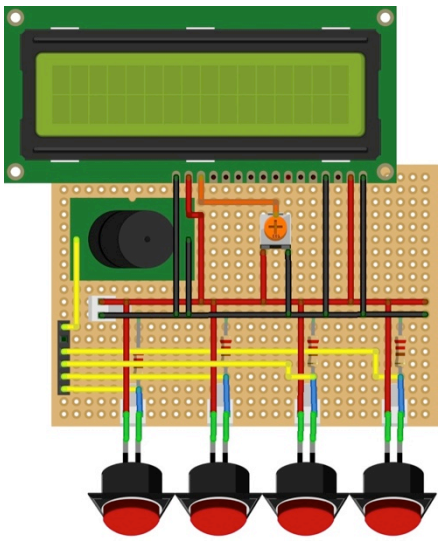


Figura 62 - Diseño placa de pantalla y pulsadores

La pantalla LCD irá soldada en la parte superior de la placa agujereada. Solo se soldarán a la placa los pines de la pantalla: 1, 2, 4, 14, 15 y 16 (referentes a tierras, 5V y común del potenciómetro), los demás pines (no se muestran en la figura anterior) irán cableados directamente hacia la placa de Arduino.

Como entradas a la placa perforada se encuentran los cuatro pulsadores (UP, DOWN, NEXT y OK de izquierda a derecha) de los cuales saldrá un cable de cada uno (cableado de color amarillo) hacia la placa Arduino para leer si están pulsados o no.

Por último el *buzzer* llevará el extremo negativo soldado al cable común de tierra (cablado negro) y el otro extremo (cableado de color amarillo) irá directo a la placa de Arduino.

Este es el resultado real del circuito del temporizador:

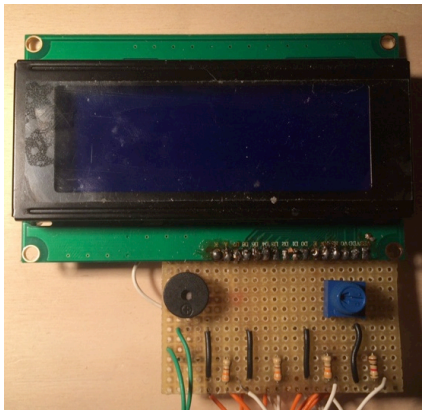


Figura 63 - Circuito real de la pantalla LCD y los pulsadores

Circuito de la matriz de LED

En este caso se ha utilizado la placa que venía fabricada por defecto con el integrado MAX7219:

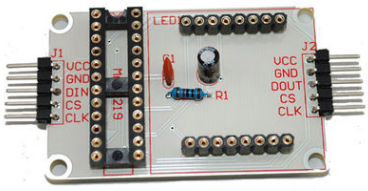


Figura 64 - Placa MAX7219

Se ha colocado el integrado en su sitio y se han cableado las 16 salidas hacia las filas y columnas de la matriz de 8x8. Los extremos de la parte derecha no se utilizan ya que no hay otra matriz en cascada y los extremos de la parte

izquierda (VCC, GND, DIN CS y CLK) van cableados directamente hacia la placa de Arduino.

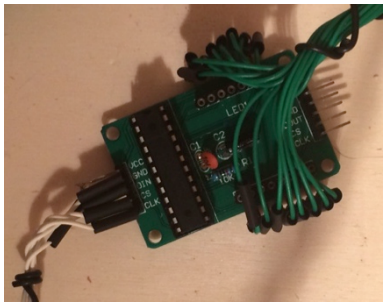


Figura 65 - Placa MAX7219 real

Los cables verdes son los que van hacia la placa de Arduino, los cables blancos van a parar a este circuito:

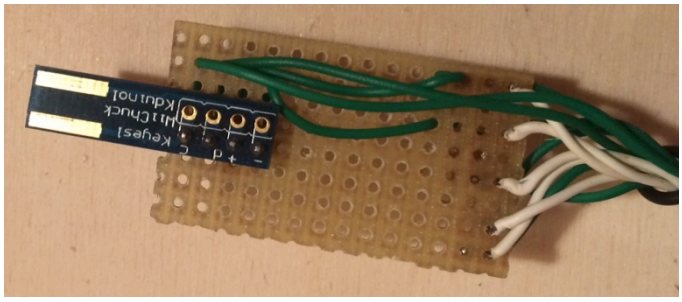


Figura 66 - Conector Nunchuk real

Los cinco cables blancos son los que vienen de la placa con el MAX7219 y los siete cables verdes van hacia la placa de Arduino (GND y VCC común para Nunchuk y la matriz, CLOCK y DATA de Nunchuk, DIN, CS, CLK de la matriz).

8. VIABILIDAD ECONÓMICA

Como se puede ver el presupuesto del proyecto el precio fabricar un juego de ajedrez es de 454,50€ la unidad. En este apartado se pretende estudiar la viabilidad económica durante su vida útil.

Se tiene en cuenta que el ciclo de vida del ajedrez es de 5 años suponiendo que en este periodo la tecnología avanzará y se creará un prototipo mejor y más económico.

También se debe contemplar un coste de reparación estimado debido a el cambio de diodos fundidos, piezas rotas, desgaste del tablero y/o fichas, pilas, cambio de botones o del mando Nunchuk… Se supondrá un coste de reparación medio anual de 20€.

Se considera que un ayudante de ajedrez cobra 10€ la hora y cada torneo trabaja durante 5 horas, también se considera una media de 5 torneos de ajedrez anuales, es decir cada año se ahorraría 250€ en el ayudante, por lo tanto:

Tabla 22 - Coste acumulado - ahorro acumulado

Año	Coste acumulado	Ahorro acumulado
0	454,5	0
1	474,5	250,0
2	494,5	500,0
3	514,5	750,0
4	534,5	1.000,0
5	554,5	1.250,0

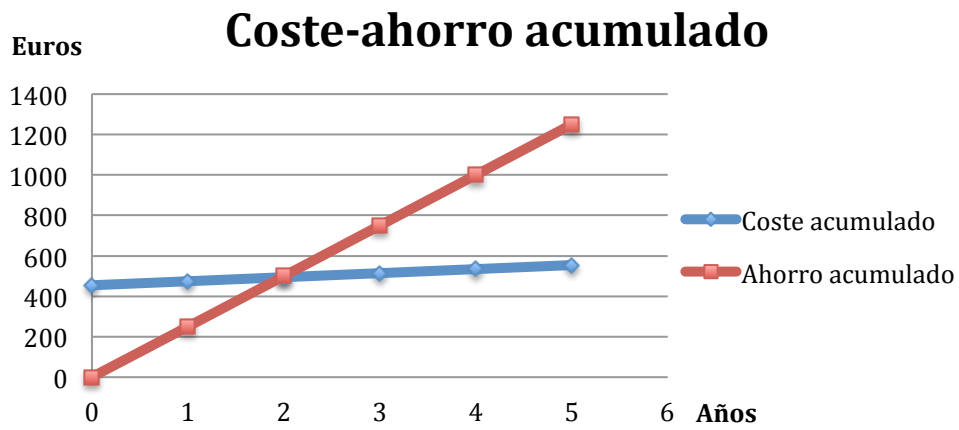


Figura 67 - Coste acumulado - Ahorro acumulado

El proyecto se amortizará a finales del año dos. Por lo tanto, el proyecto es viable económicamente ya que se amortiza antes de la mitad de su vida útil.

9. CONCLUSIONES

El proyecto ha concluido cumpliendo todas las especificaciones indicadas al principio del documento. El ajedrez es capaz de reducir el tiempo de comunicación entre jugador-asistente, por lo que tendrá más tiempo para pensar y no será una gran preocupación agotar el tiempo de manera innecesaria.

Se han programado dos códigos distintos, a la vez comunicados, eficientes que cumplen con sus funciones a la perfección y no se han detectado errores en el código en todas las pruebas que se han realizado.

En un futuro sería conveniente ampliar el estudio sobre la fabricación del brazo robot para dar más independencia al jugador. Además de la fabricación, se deberá diseñar un programa en C robusto que permita el manejo eficiente del brazo robot.

También se podría mejorar el diseño final del ajedrez, modificar el acabado en madera por un acabado ligero en plástico o algún material de similares características. Se podría utilizar alguna lente que ampliara el ángulo de iluminación de los diodos LED y así reducir el grosor del tablero y minimizar el peso final del mismo.

En caso de su futura comercialización se deberían diseñar placas impresas (PCB) para asegurar que funcionará durante años sin problemas en las soldaduras. Se podría diseñar una placa que agrupara todas las placas separadas que contiene este proyecto (placas del MAX7219, del conector del Nunchuk, de la pantalla LCD...). Se reduciría el precio programando sobre un PIC en lugar de sobre una placa de Arduino.

Otras características interesantes sería la extrapolación del código para implementar un segundo mando de juego para poder jugar dos oponentes con discapacidad física, se podría memorizar la posición de cada pieza y validar si un movimiento es correcto o no. También se podría programar una segunda pantalla LCD para que cada jugador tuviera una delante suyo y facilitar la lectura de los tiempos restantes, turnos…

Por último se debería estudiar, diseñar y construir distintos *interfaces* de control de cada jugador para abarcar distintas discapacidades como un mando para mover el joystick con la lengua, mover el acelerómetro con la cabeza, escoger las casillas mediante reconocimiento ocular para personas que no pueden mover ninguna parte del cuerpo… Esto ampliaría el volumen de personas que podrían utilizar un ajedrez de estas características.

Con la realización de este proyecto se ha aprendido sobre los siguientes aspectos:

- Elevar el nivel de programación en C hasta ahora adquirido en la universidad.
- Aprender a controlar dispositivos electrónicos tales como:
  - Mostrar texto en pantallas de cristal líquido (LCD), controlar la manera en que aparecen, crear símbolos nuevos para mostrar,…
  - Utilizar librerías para controlar los datos que llegan de un dispositivo tal como Nunchuk, interpretar sus datos y experimentar sobre los valores para detectar un nuevo evento como mover el *joystick*, inclinar el mando o pulsar algún botón.
  - Aprender a controlar un integrado para controlar 8 dígitos que utiliza el multiplexado como el MAX7219, entender sus conexiones y utilizar una amplia librería para gobernar 64 diodos LED.
  - Aprender a depurar códigos para solucionar errores en la programación, construir el código de un temporizador, crear distintos menús para la configuración,…
- Implementar la comunicación entre dos placas de Arduino, poder detener el código de un código a partir de la recepción de un dato por el puerto serie, enviar la configuración del modo de juego de una placa a otra,…
- Recordar el funcionamiento del *software* de diseño Solidworks y aprender nuevas características tales como dar apariencia a los objetos, crear relaciones de posición,…
- Adquirir nuevos conocimientos en *software* para mi nuevos como el programa de diseño de esquemas electrónicos Fritzing, web de diseño de esquemas Cacoo…
- Aprender a soldar componentes electrónicos en placas perforadas y crear las pistas metálicas mediante cables.
- Aprender a organizar el tiempo y ajustarse a la planificación establecida.

Este proyecto podrá ayudar a aumentar el interés en este campo donde no se oye mucho hablar o no se han creado objetos adaptados para personas que no pueden jugar con normalidad a juegos tan comunes como el ajedrez.

10. BIBLIOGRAFÍA

- Michael Margolis, *Arduino Cookbook.* Año: 2012. Cap. 4: *Serial Comunications* y Cap. 11: *Using Displays.*
- Óscar Torrente, *Arduino: Curso práctico de formación.* Año: 2013. Cap. 5: Librerías Arduino.
- INDRA (http://www.tecnologiasaccesibles.com) Tecnologías accesibles (Consulta: 20 de abril de 2014)
- Instituto Nacional de Estadística (http://www.ine.es/prensa/np524.pdf) Encuesta de Discapacidad, Autonomía personal y situaciones de Dependencia Año 2008 (Consulta: 20 de abril de 2014)
- FEDERACIÓN ESPAÑOLA DE DEPORTES PARA CIEGOS (http://www.fedc.es) Reglas del ajedrez para ciegos (Consulta:21 de abril de 2014)
- IPADNANO, Temporizador: Arduino + LCD (Consulta: 4 de Mayo de 2014) (https://electronicavm.wordpress.com/2011/06/22/temporizador-arduino-lcd/)
- QUINAPLUS (http://www.quinapalus.com/hd44780udg.html) HD44780 LCD User-Defined Graphics (Consulta: 9 de Mayo de 2014)
- XBEE (http://www.xbee.cl) Información sobre comunicación XBee (Consulta: 14 de Mayo de 2014)
- ARDUINO (http://arduino.cc/es/Main/Hardware) Características Arduino UNO (Consulta: 16 de Mayo de 2014)
- OLGA SIERRA, Discapacidades: Tipos y características (Consulta 22 de Mayo de 2014) (http://atendiendonecesidades.blogspot.com.es/2012/11/distintos-tipos-de-discapacidad-y-sus-caracteristicas.html)
- JOSE IGNACIO SUÁREZ MARCELO, Como gobernar un *display* LCD (Consulta 27 de Mayo de 2014) (http://eii.unex.es/profesores/jisuarez/descargas/ip/lcd_alfa.pdf)
- INFO-AB(http://www.info-ab.uclm.es/labelec/solar/electronica/elementos/servomotor.htm) El Servomotor (Consulta 6 de Junio de 2014)
- HOBBYKING (http://www.hobbyking.com/hobbyking/store/__9549__turnigy_tg9e_9g_1_5kg_0_10sec_eco_micro_servo.html) Datasheet servomotor Turnigy TG9e (Consulta: 1 de Septiembre de 2014)
- CACOO (https://cacoo.com/) Diseño de diagramas de flujo (Consulta 10 Junio de 2014)
- *Software* FRITZING. Diseño de esquemas electrónicos (enlace descarga: http://fritzing.org).