Informe de Diseño y Construcción de MechChess

Sebastián Bensusan 50607
Sebastián Aranega 50620
Nazareno Maggi 50649

10 de Abril de 2014

Contents

1 Resumen

Se diseñó y fabricó un producto llamado MechChess que permite jugar al ajedrez interactivamente. En este informe se detallan los objetivos principales del mismo y las decisiones tomadas para cumplirlos. Luego se recorren los distintos subsistemas del producto (incluyendo el software) y las justificaciones de cada punto del diseño. Además se provee documentación para la construcción del producto. Finalmente se compara el costo final con el presupuesto inicial y se extraen conclusiones sobre el proceso de diseño y producción.

2 Objetivos y Lineamientos Principales

El objetivo principal del producto se definió en el Brief (Apéndice A) como:

- Seguir un partido de ajedrez desde el comienzo indicándole a los jugadores si los movimientos realizados son legales o no.

Los objetivo de máxima principales se definieron cómo:

- Agregar un reloj para seguir el tiempo de cada jugador.
- Indicarle a los jugadores que movimientos son legales una vez que se elige una pieza.
- Reproducción de Partida

De los objetivos de mínima se desprende que el producto debe tener tres capacidades básicas: INPUT, entender los movimientos de los jugadores, ("movimientos realizados"), OUTPUT, comunicarle al usuario información ("indicándole a los jugadores..."), y capacidad de capacidad de decisión ("movimientos legales o no").

Para poder seguir los movimientos se decidió colocar sensores Hall debajo de cada casillero e imanes en cada una de las piezas. Las reglas del ajedrez implican que el jugador puede tomar dos tipos de decisiones además de movimientos: ofrecerle tablas al oponente o rendirse, y elegir el tipo de pieza una vez que se corona algún peón. Como la segunda decisión es crítica para seguir el juego, se decidió agregar cuatro pulsadores (uno por cada tipo de pieza) al producto. Para permitirle a los jugadores terminar un partido y comenzar uno nuevo se agregó un interruptor de reset.

Para poder comunicarle al usuario información se decidió colocar un LED rojo y un LED azul debajo de cada casillero. Eso implica que el tablero (la superficie donde se apoyan las piezas) debe tener cierta transparencia. Inicialmente no se tenía un uso definido para cada color, sino que eligió en función de permitir flexibilidad al momento de implementar la interfaz final.

En el caso de los botones, se agregó un LED en paralelo con cada uno, para que el usuario reciba feedback si los presiona. Para el reloj se agregó un display LCD de dieciséis caracteres por dos líneas. Para unir estos sistemas y tomar

decisiones se eligió utilizar un microcontrolador Arduino Mega por su cantidad de puertos digitales (54) y facilidad de uso y programación.

El producto final se encuentra en la Figura 1, donde también se muestran las piezas utilizadas.

Figure 1: Producto Final

3 Diseño y Construcción

3.1 Hardware - Problemas y Soluciones

3.1.1 Diseño de la Caja y el Tablero

Para el tamaño del tablero se tomaron como referencia las piezas mecanizadas a lo largo del curso de 30mm de diámetro. Al no estar el set de 32 piezas completo, finalmente se utilizaron otras piezas pero el tamaño de cada casillero se eligió de 35mm para acomodar las piezas originales. Las dimensiones de la caja son de 350mm x 300mm x 40mm. Se eligió acrílico translúcido blanco para toda la caja y el tablero ya que además de tener muy buena deja pasar la luz de los LEDs pero no deja que se vean los circuitos de los mismos. El problema es que la luz de cada LED no queda contenida a su tablero si no que se esparce a los

demás. Por eso se mandó a armar una cuadrícula de madera cortada a láser que separe los casilleros.

El tablero sigue un diseño tradicional y agrega símbolos para los pulsadores de la coronación y reset y un logo. Además se dejó lugar para el display de LCD en el centro. El diseño del tablero (Figura 2) se imprimió en una capa de vinilo que luego se pegó sobre la cara superior.

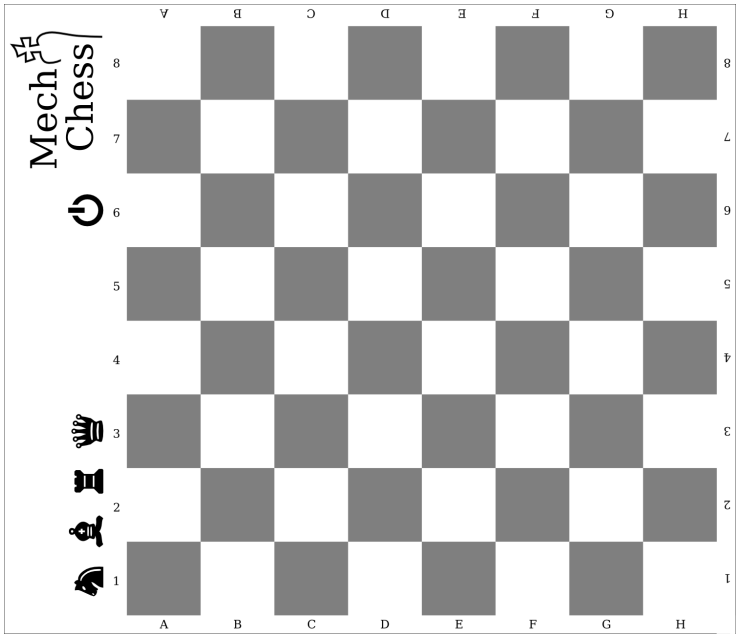


Figure 2: Diseño del Tablero impreso en vinilo

3.1.2 Diseño de los casilleros

Dado que hay 64 casilleros y cada uno tiene tres componentes (sensor, led azul, y led rojo), en principio se necesitarian $64 \cdot 3 = 192$ puertos del microprocesador para sensar y actuar. Para resolver ese problema, se decidió utilizar un sistema igual al teclado matricial para los sensores Hall. Para leer el casillero $[R, C]$ se alimenta el puerto de la columna C con 5V y se lee el puerto de la fila R. Como el sensor se encuentra a 5V cuando no detecta campo magnetico y funciona como un switch (pasando a 0V cuando detecta), se debe agregar una resistencia pull up entre la salida y una alimentación independiente a la de control. El costo del sistema matricial es que no se pueden leer sensores de dos columnas distintas en simultáneo. Como los usuarios se lento respecto al microcontrolador esto no es un problema.

Para los LEDs se implementa un sistema muy similar, pero como sólo necesitan alimentación y tierra para funcionar, se les agrega un transistor que le abre el paso a la corriente cuando recibe la señal de control. Entonces para

prender el LED azul del casillero $[R, C]$ se sube la alimentación de la columna C, y se sube la señal de control de los LEDs azules de la fila R. En este caso el costo del sistema matricial es mayor: no se pueden prender LEDs de columnas distintas al mismo tiempo sin prender otros casilleros no deseados. La solución es prenderlos por columna lo suficientemente rápido como para que parezca que estan prendidos en simultaneo. El resultado final es que baja la intensidad de la luz cuando se prenden varias columnas en simultáneo.

Como los casilleros comparten señales se reduce la cantidad de puertos totales necesarios pero se aumenta la cantidad de señales que cada casillero debe manejar. Esto implica que cada casillero debe tener:

1. Alimentación controlada para el sensor (VH).
2. Señal de salida del sensor (HO).
3. Señal de alimentación continua para el pull up de los sensores (VC).
4. Alimentación controlada para los LEDs (VL).
5. Señal de control para el LED azul (VA).
6. Señal de control para el LED rojo (VR).
7. Tierra común a los tres componentes (GND).

Con esas 7 señales por casillero se puede controlar cada uno de los tres elementos por separado. Si se desea sensar, se sube VH a 5V y se lee HO. Si se desea prender alguno de los LEDs se sube VL a 5V y luego se activa VA o VR o ambas. El esquemático final del circuito necesario en un casillero se puede ver en la Figura 3.

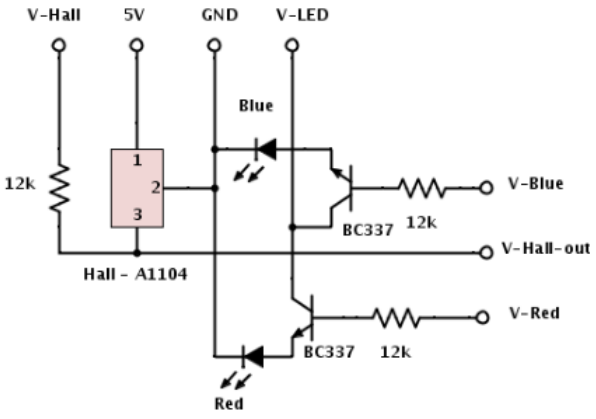
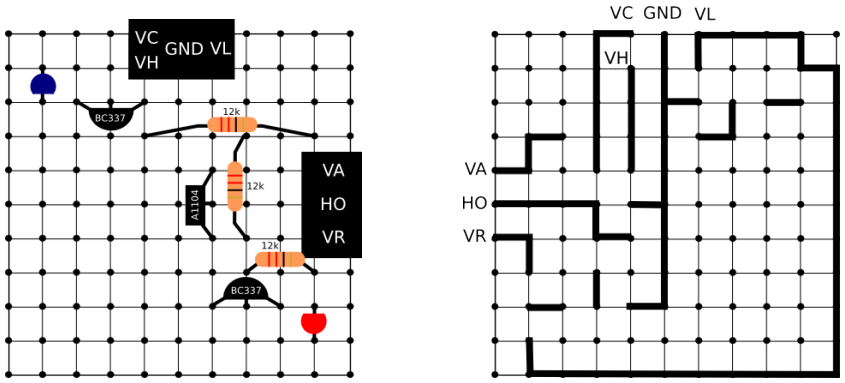


Figure 3: Esquematico de la Placa de los casilleros

Se decidió implementar el circuito de cada casillero en una placa separada y conectar las filas y columnas con cables planos. Al ser unidades separadas, en caso de que exista alguna falla es mucho más facil identificarla, diagnosticarla, y solucionarla que si las placas estuviesen unidas. Las placas se pueden reemplazar unas por otras y además se pueden probar individualmente. Por falta de experiencia con el proceso de planchado de placas y el tamaño de las placas necesarias se decidió soldarlas todas manualmente en placas multiperforadas. Cada placa incluye dos conectores de pines, uno para el cable de fila y uno para el cable de columna, que fijan la posición de la placa en el tablero. El diagrama de soldadura y presentación de componentes se encuentra en la Figura 4 y ejemplos de placas construidas en la Figura 5.



(a) Presentación de componentes (b) Lineas de soldadura

Figure 4: Placa implementada en una placa perforada

(a) *Presentación de componentes* (b) Soldadura

Figure 5: Placa construida

3.1.3 Botones

Cuando el usuario presiona un boton es importante darle feedback para que entienda que funcionó y no lo siga presionando. Por eso se le conectó en paralelo a la salida de cada boton un LED azul que se prende junto al pulsador. Los

pulsadores sólo tienen función durante la inicialización y durante la coronación de algún peón. El resto del tiempo no sirven. Con el LED conectado en paralelo, si el usuario presiona algún pulsador cuando éste no sirve, recibe un feedback positivo aunque el pulsador no tiene efecto alguno. Por eso es deseable que la alimentación de los pulsadores sea controlada por un puerto. De esa manera los pulsadores se "activan" sólo cuando se los necesita y el usuario sólo recibe feedback cuando los botones cumplen alguna función. Por otro lado, el interruptor de reset, que también cuenta con un LED, se alimenta directamente ya que el usuario puede utilizarlo en cualquier momento.

3.1.4 Conexiones

Todos los casilleros de una fila tienen en común VA, VR, y HO, mientras que todos los casilleros de una columna tienen VH y VL en común. VC y GND se transmiten a lo largo de los cables de las columnas, pero todos los casilleros comparten esos dos puertos. Se utilizó un cable plano de 6 tanto para conectar todos casilleros de cada fila como los casilleros de cada columna. En la posición de cada casillero se le agregó al cable un conector hembra y se lo pegó sobre la caja. Hay 16 cables, 8 con tres señales (filas), y 8 con cuatro (columnas). Los conectores pegados en la base de la caja fijan la posición de las placas de los casilleros (Figura 6).

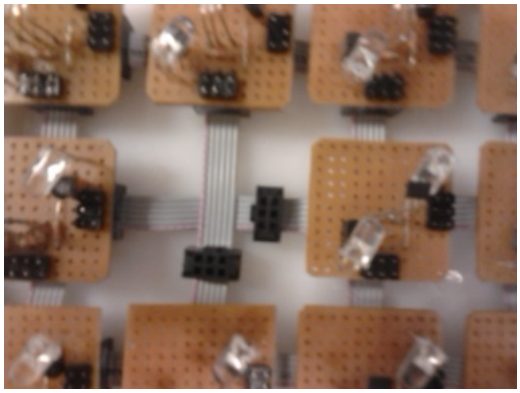


Figure 6: Conectores y posición de las placas

Los 16 cables de las filas y columnas se conectan a una placa intermedia que reorganiza las conexiones para que luego se conecten al microcontrolador. Se agruparon las señales de la fila N con las señales de la columna N. De esa manera se tienen 8 cables de 5 señales (VA, VR, VH, HO, VL), cada uno con toda la información de un par fila-columna. Como VC y GND son común a todos los casilleros, hay un único cable doble yendo desde el microcontrolador a la placa intermedia. La ubicación de la placa intermedia respecto a los casilleros se puede ver en la Figura 8. Luego de la placa intermedia, se conectan los cables al microcontrolador como se indica en la Figura 7. En el diagrama se omiten

los cables fila-columna 3, 4, y 5 por claridad, pero estos se conectan de manera simetrica a los cables 6,7, y 8 respectivamente siguiendo los puertos de conexión de la Tabla 1.

Fila-Columna	VL	HO	VA	VR	VH
1	23	22	14	15	16
2	17	18	19	20	21
3	24	26	28	30	32
4	34	36	38	40	42
5	44	46	48	50	52
6	25	27	29	31	33
7	35	37	39	41	43
8	45	47	49	51	53

Table 1: Puertos de Cables Fila-Columna

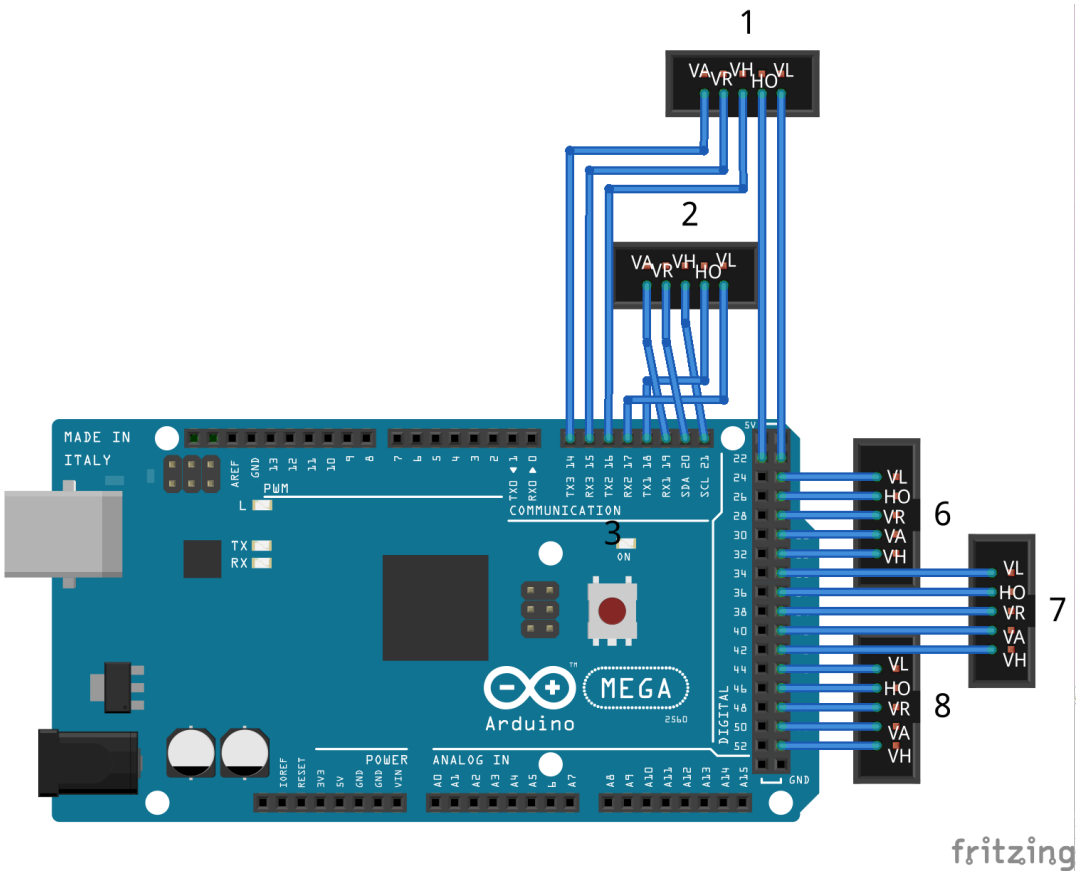


Figure 7: Conexión de placas al microcontrolador *(sin cables 3, 4, y 5)*

Figure 8: Placa Intermedia

El microcontrolador tiene puertos con conexión hembra. Unir un cable plano directamente a una tira de pines es muy difícil y propenso a cortos y roturas. Por eso se unió a unas pequeñas placas perforadas intermedias con pines y cables soldados. Estas uniones aunque no tienen problemas de cortos, son muy sensibles y deben ser manipuladas con cuidado. Para las filas y columnas se utilizaron todos los puertos del 22 al 52.

Los botones tienen una placa propia de conexión donde se incluyen las resistencias pull-down, la conexión a tierra, las entradas, y la alimentación a los botones. Se utilizaron los puertos del 8 al 13.

La pantalla LCD puede ser cualquier pantalla de 16x2 caracteres compatible con el driver Hitachi HD44780. El LCD tiene también su placa de conexión que alimenta, conecta a tierra, y anula los puertos innecesarios (D0-D3). El puerto que maneja el contraste necesita una señal de alrededor de 1V que se consiguió con un divisor de tensiones implementado en la misma ($5k\Omega$ y $1k\Omega$). Aunque la pantalla tiene 16 puertos, el microcontrolador sólo necesita 6 para controlarla. Como se puede ver en la Figura 9, se utilizaron los puertos del 2 al 7.

La conexión de USB (tipo B hembra) y de alimentación del microcontrolador se encuentran en un costado de la caja para que el usuario pueda programar y/o alimentarlo. De esa manera se permite que se carguen nuevos programas al sistema y se agregue funcionalidad. La conexión de alimentación tiene un conector barril de 2.1mm con el centro positivo y necesita una fuente regulada de 5V con un minimo de 1A. Es la misma ficha del Arduino Mega 2560 y cualquier fuente compatible con el Arduino Mega funciona con MechChess.

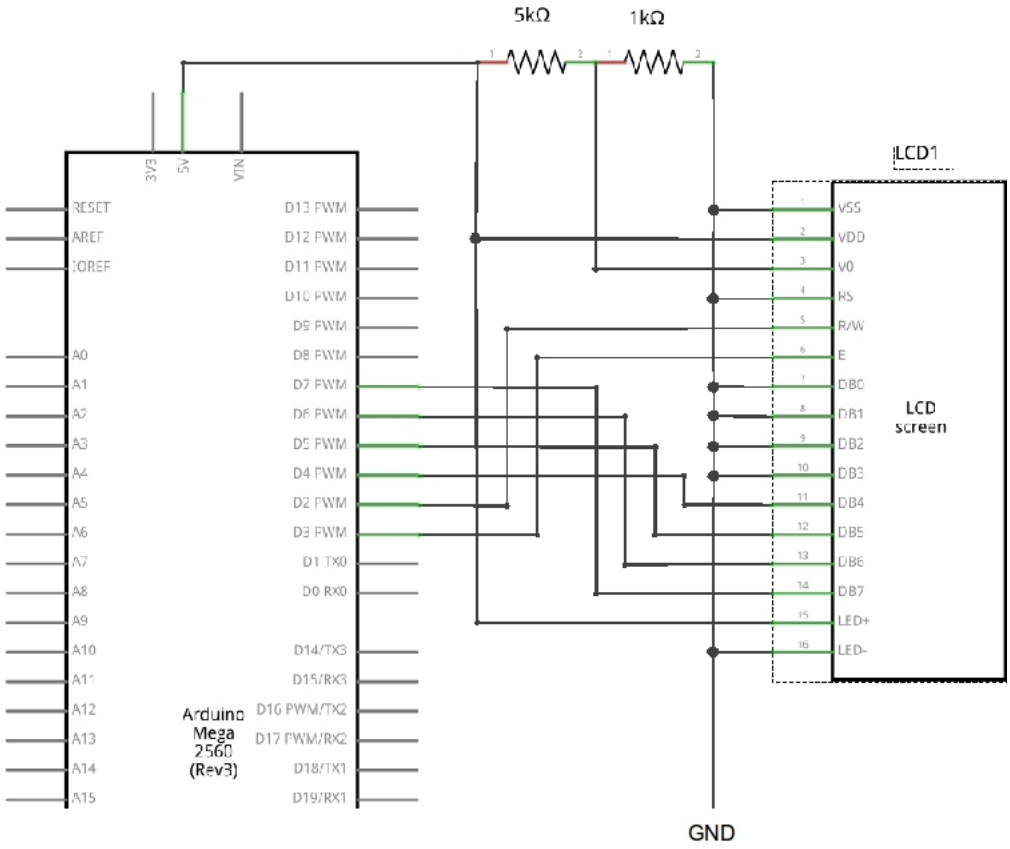


Figure 9: Esquematico LCD

3.2 Software - Problemas y Soluciones

3.2.1 Control de LEDs y Sensores Hall

El sensor lee las columnas y guarda las lecturas (0 si no hay pieza, y 1 si hay) en una matriz binaria de 8x8 (llamada *out*). Cuando percibe un cambio respecto a la lectura anterior guarda en un buffer el movimiento leído con tres datos: fila, columna, y si se levantó o se apoyó una pieza (valor lógico). El buffer es FIFO y es el único recurso que ofrece el driver. Se puede acceder al mismo mediante dos funciones: *any_moves_left* que devuelve un valor lógico y *next_move* que devuelve el próximo movimiento en el buffer en caso de haber alguno. Para ignorar lecturas ruidosas se implementa un filtro pasa bajo en la medición: para registrar un cambio, se debe leer N veces el nuevo valor antes de darlo como válido.

Como el hardware de ambos sistemas es similar, los drivers y las estructuras de datos son también similares. El driver de LEDs cuenta con dos matrices bi-

narias de 8x8 en las cuales se indica que casilleros deben estar prendidos. Luego, el driver se encarga de activar los puertos apropiados y recorrer las columnas lo suficientemente rápido. Estas matrices se pueden modificar mediante las funciones *set_square* (que toma un fila, una columna y un color y prende/apaga el casillero correspondiente), *grid_off*, que toma un color y apaga todos los casilleros de ese color, y *grid_on* que toma una matriz binaria y un color como argumento y prende esos casilleros de ese color.

3.2.2 Pantalla LCD

La plataforma de Arduino incluye un driver para pantallas llamado LiquidCrystal[1]. Se utilizaron las funciones *clear* que limpia la pantalla, *setCursor* que ubica el cursor en un punto, y *print* que escribe el texto adecuado.

Enviarle información al LCD lleva tiempo y a lo largo del programa hay una serie de flags (*display_on*) y estructuras de control que complican el código pero se aseguran de que se mande la información al LCD sólo cuando es necesaria y no en cada ciclo del programa.

3.2.3 Botones

El driver de los botones es muy sencillo. Lee los botones periódicamente si están activados y guarda los cambios en un buffer. Ofrece una forma de activar y desactivar los pulsadores (*activate_buttons*), una forma de saber si hay algo en el buffer (*any_buttons_left*), acceso al buffer (*next_button*), y una forma de saber si está prendido el interruptor de reset (*is_power_on*).

3.2.4 Reloj

El reloj tiene dos registros que guardan el tiempo utilizado por cada jugador (*clock_white, clock_black*) y un parámetro (*duration*) que indica cual es el tiempo de juego (seleccionado en la inicialización). Ofrece dos funciones, *update_player* que actualiza el reloj del jugador seleccionado, *get_players_clock* que devuelve una lista de caracteres con el tiempo del jugador seleccionado, y *is_done* que comprueba que ambos jugadores tengan tiempo para jugar. Las implementa con una función del sistema Arduino, *millis*, que devuelve la cantidad de milisegundos desde que el microcontrolador se inicializó.

3.2.5 Reglas del Ajedrez

Las reglas del movimiento de las piezas se encuentran en la clase Piece. Las piezas pueden ser de tipo Pawn, Knight, Bishop, Rook, Queen, o King y de color White o Black. Los casilleros vacios son de tipo Empty y color None. La clase Piece provee la función *is_move_legal* que toma un posible movimiento (dos pares fila-columna) y un determinado tablero y devuelve un valor lógico.

[1] http://arduino.cc/en/Reference/LiquidCrystal

La posición de las piezas entre cada movimiento (tableros) se guardan en la clase Board. Esta guarda los tipos y colores de piezas (objetos Piece) en una matriz de 8x8. Además tiene una serie de flags para recordar si todavía es posible para cada jugador enrocar y capturar en passant. Este set de variables tiene la "historia" del juego y se modifica cada vez que se realiza una movimiento. La clase Board ofrece las funciones *is_in_check* (evalua si el jugador está en jaque), y *cant_move* (evalua si el jugador tiene capacidad de movimiento).

La totalidad del juego se guarda en ChessGame. Allí se incluye además el jugador de turno en cada instante (*turn*) y el reloj de ambos jugadores. Es allí donde se implementa la maquina de estado que sigue el juego. Un sólo movimiento se captura con tres estados principales y dos estados secundarios. Una vez que se encuentra que el movimiento es válido, se lo guarda en Board, se cambia el turno y el reloj y se inicia el proceso de nuevo. Los estados son:

Waiting: se esperan movimientos del jugador en turno y todas las piezas están apoyadas. No se prende ninguna luz, se desactivan los botones, se actualiza el reloj del jugador de turno, y se muestra el reloj de ambos. Si alguien a ganado (o hay tablas) se pasa a Ended. Si se encuentra un movimiento se lo evalúa. Si es de una pieza que se puede mover se procede al estado Moving, si no a Error.

Moving: el jugador de turno levantó una pieza y todavía no la apoyó. Se muestran con LEDs azules los casilleros donde puede apoyar la pieza, se desactivan los botones, se actualiza el reloj del jugador de turno, y se muestra el reloj de ambos. Si se encuentra un movimiento se lo evalúa. Si se apoyó una pieza en uno de los casilleros legales, se guarda el movimiento en Board, se cambia de turno, y se pasa a Waiting. Si se levanta una pieza del color opuesto que puede ser capturada, se deja sólo a ese casillero como movimiento legal en azul. Si es un peón para coronar se pasa a Promotion. Si es cualquier otra pieza se cambia a Error.

Promotion: se activan los pulsadores, y se le indica al usuario que presione alguno y elija una pieza. Cuando lo hace, se guarda el tipo de pieza en Board, se cambia de turno y se pasa a Waiting.

Ended: el juego a terminado, se prende en azul el casillero del rey ganador y en rojo el del perdedor. Se muestra el ganador en el display y el sistema espera a ser reseteado.

Error: se escanea continuamente el tablero. Cualquier diferencia leída por los sensores con Board (último tablero válido en memoria) se guarda en *diff* y se indica con LEDs rojos. Una vez que no hay diferencia se procede al último estado, Moving o Waiting.

3.2.6 Inicialización de Parámetros

Al iniciar el sistema se selecciona el modo de juego y la duración del mismo y luego se espera a que todas las piezas estén bien posicionadas. Las etapas de selección se implementaron con una máquina de estados secuencial muy sencilla:

SetMode: se activan los pulsadores, y se le indica al usuario que es lo que puede seleccionar en la pantalla. El botón del caballo selecciona el modo Learning, y el de la reina el modo Pro. Una vez que se tiene la selección se pasa al estado SetClock.

SetClock: se activan los pulsadores, y se le indica al usuario en la pantalla que es lo que puede seleccionar (5m, 30m, 60m, e infinito). Una vez que se tiene la selección se pasa al estado SetPieces.

SetPieces: se escanea continuamente el tablero. Cualquier diferencia leída por los sensores con el tablero inicial *Board_init* se guarda en *diff* y se indica con LEDs rojos. Una vez que no hay diferencia se inicializan los relojes, se guarda el turno como White, y se comienza la máquina de estados principal en Waiting.

4 Costos

4.1 Presupuesto Inicial

Junto al Brief (Apéndice A) se presentó un presupuesto inicial (Tabla 2). En el mismo se estimó que los mayores costos vendrían de los componentes principales. Para el resto de los componentes como cables y conectores se asumió un margen del 20%. Como originalmente se iba a usar un HSC08-JM60 del departamento de mecatrónica se omitió el costo del microcontrolador. Se asumió que otros componentes electrónicos comunes como resistencias y transistores serían encontrados en el pañol del ITBA.

Componente	Cantidad	Precio Unitario $	Subtotal $
Sensores Hall UGN3503U	64	13.7256	878.5
L-5BW/12000 LED Azul	64	3.272	209.4
L-5BW/12000 LED Rojo	64	3.272	209.4
Tablero + Caja	1	500	500
Placas Perforadas	1	100	100
		Subtotal	1897.25
Margen por Miscelaneos		20.00%	379.5
		Total	**2276.7**

Table 2: Presupuesto Inicial

4.2 Costo final del Producto

Como se ve en la Tabla 3, el costo final del producto fue un 60% más de lo presupuestado. Esto indica que las hipótesis del presupuesto inicial estaban equivocadas y que no se cumplió el objetivo de costos. Se puede ver que los precios presupuestados son iguales o menores a los finalmente obtenidos *para los*

items tenidos en cuenta; se consiguieron LEDs a la mitad del precio presupuestado, sensores Hall levemente más baratos, casi todas las placas perforadas provinieron del pañol, y el costo extra del tablero y vinilo se tuvo en cuenta con el 20% de margen.

La diferencia viene de todas las cosas auxiliares que se ignoraron en el presupuesto inicial a las que se les adjudicó el margen de 20%. Los conectores y cables terminaron costando más que los sensores. Los imanes que se habían elegido al empezar el diseño resultaron ser inadecuados y se necesitaron imanes especiales. Además, al rediseñarse el producto para cumplir los objetivos de máxima se agregó la categoría de Botones y LCD. Finalmente, también se compró un microcontrolador cuando la cantidad de puertos necesarios superó las capacidades del HS08.

En otras palabras, cuando se está diseñando un producto, hace falta elegir un margen de seguridad mucho mayor en el presupuesto inicial. Las compras programadas y hechas pueden resultar totalmente equivocadas y nuevos costos aparecen con cada nueva decisión. Además se debe planificar mejor el presupuesto inicial ya que alguno de los costos sorpresa eran fácilmente verificables (conectores y cables) a pesar de no ser intuitivos. En este caso, habiendo tenido en cuenta los costos de los conectores, un margen del 20% hubiese alcanzado. Sin embargo, para proyectos futuros se tendrá en cuenta un margen del 50%.

El valor presentado es el costo del proceso de diseño y no el del producto. Hay compras repetidas, costo de repuestos, y componentes sobrantes. El costo de materiales para la fabricación se estima en $3000. No se están teniendo en cuenta las horas hombre necesarias para soldar y ensamblar el producto. Durante el diseño no se tuvo en cuenta el proceso de producción más allá de la capacidad del equipo para llevarlo a cabo. Algunas decisiones que resultaron ser muy efectivas (ej: construcción de placas individuales) no son buenas para un proceso de producción estandarizado.

Concepto	Subtotal $
Sensores Hall	853.3
LEDs	276.44
Conectores y Cables	1042.07
Tablero y Vinilo	771
Microcontrolador	289
Imanes	237
Botones y LCD	205
Placas Multiperforadas	32
Total	**3705.81**

Table 3: Costo Final

5 Conclusión

Se cumplieron todos los objetivos de mínima y algunos de máxima. Los que no se cumplieron pueden ser todos implementados sobre el hardware diseñado. En ese sentido el producto es extensible.

A lo largo del diseño, tanto de software como de hardware, se tuvo en cuenta la modularidad de los subsistemas y se trató de minimizar su interdependencia. Es por eso que el producto es fácilmente mantenible. La única excepción son los conectores entre la placa intermedia y el microprocesador que son proclives a la rotura y difíciles de reemplazar.

El costo final fue un 60% mayor al inicialmente presupuestado. En ese sentido, se fracasó en cumplir el objetivo de costos.

Apéndice A : Brief

Nombre del Proyecto: MechChess

Descripción: Tablero de Ajedrez Electronico - Interactivo

Integrantes: Sebastián Aranega, Nazareno Maggi, Sebastián Bensusan.

Periodo: 2013/10/21 – 2013/12/01

Objetivos de mínima:

- Seguir un partido de ajedrez desde el comienzo indicandole a los jugadores si los movimientos realizados son legales o no.
- Tablero con 64 sensores de efecto Hall montados en la base.
- 64 leds mostrando las posiciones de las piezas y potenciales movidas.
- En principio el SW solo registra las movidas.

Objectivos donde falta precisión:

- Microcontrolador con interfaz USB que reciba la información de los sensores Hall y que encienda los LEDs. *¿Interfaz USB a una computadora? ¿Qué funcionalidad ofrece la computadora?*

Objetivos de máxima

- Agregar un reloj para seguir el tiempo de cada jugador.
- Indicar a los jugadores que movimientos son legales una vez que se elige una pieza.
- Análisis de movida permitida.

- Presentación de movidas.
- Reproducción de Partida.
- Incorporar un motor de ajedrez para partidos contra Inteligencia Artificial.