Universidad de la República
Facultad de Ingeniería

Leticia Cirlinas Martins
Francisco Manfredi Jardi
Andrés Spaggiari Bernardi

Tutor
Michel Hakas

23 de octubre de 2008

Resumen

El presente documento describe el diseño, construcción y caracterización de un juego de tatetí formado por un sistema robótico ubicado en el Museo Participativo de Ciencia Viva.

El sistema consiste en un brazo mecánico encargado de mover las fichas del juego, el mueble con el tablero de tatetí donde se encuentra montado el brazo, el hardware eléctrico para su control formado por seis placas de electrónica, un tablero de interfaz compuesto por luces y botones que permiten al usuario interactuar con el sistema y el software implementado para un microprocesador Rabbit encargado de la lógica del juego y movimientos de los motores.

Además adjunta los esquemáticos y PCBs diseñados así como los archivos fuente del software. Se incluye también un manual de mantenimiento con los detalles del montaje mecánico, instalación eléctrica y sus cuidados.

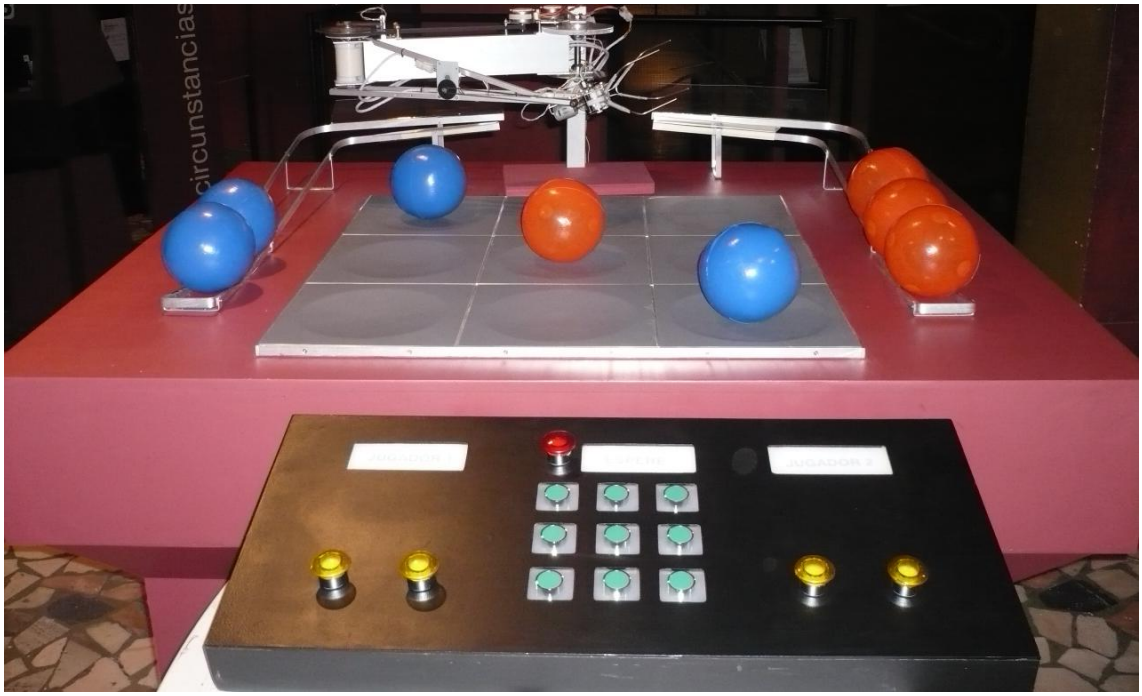


Figura 1: Foto del juego terminado.

Estructura de la documentación

A continuación se describe el contenido de la presente documentación.

- **Capítulo 1 – Introducción.** Incluye la motivación, los antecedentes y el contenido de la documentación del proyecto.
- **Capítulo 2 – Definición del Sistema y Prediseño.** Detalla el proceso de definición inicial del sistema y prediseño.
- **Capítulo 3 – Diseño y construcción Mecánica.** Describe el proceso de diseño y construcción del dispositivo mecánico implementado.
- **Capítulo 4 – Hardware de control de Motores.** Detalla el diseño del hardware eléctrico construido para el control del dispositivo mecánico.
- **Capítulo 5 – Diseño e implementación de la Interfaz de Usuario.** Describe el proceso de diseño y construcción de la interfaz de usuario del juego.
- **Capítulo 6 – Control del Sistema.** Detalla la implementación del hardware necesario para lograr la interacción del microprocesador con el sistema.
- **Capítulo 7 – Desarrollo de Software.** Se presenta todo lo referente al diseño e implementación del software de control del juego.
- **Capítulo 8 – Prototipo y caracterización del Sistema.** Se presenta el prototipo del juego, las pruebas realizadas para su terminación y la caracterización del juego.
- **Capítulo 9 – Conclusiones.** Resume los conceptos y puntos más importantes presentados a lo largo de la documentación.

Agradecimientos

Los integrantes del grupo quisiéramos agradecer especialmente a las siguientes personas:

A Alberto "Tito"Levy, Daniel Bergara, Fernando, Gerardo, Michel Hakas y todo el personal de Ciencia Viva por su valiosa colaboración en el proyecto.

A Raúl Arbiza por su excelente disposición y apoyo en la construcción de las placas de electrónica.

Al Instituto de Física de la Facultad de Ingeniería, a la empresa Prodie S.A. por haber donado materiales.

A Alfredo Spaggiari por su invaluable ayuda en la construcción del brazo y a María Bernardi y toda la familia Spaggiari por su gran hospitalidad y paciencia.

A Mario Cirlinas y Lucy Martins por sus amables consejos técnicos y por colaborar con la compra de materiales y con los costos del proyecto.

A Roberto Manfredi por sus generosos consejos técnicos y al resto de la familia Manfredi por donar diversos materiales y herramientas y traer desde Buenos Aires algunos integrados.

A Andrés Silva, Damián Pintos, Eduardo Perera, Federico Izmirlian, Fernando Hortiz, Gerardo Capucho, Santiago Conde, Sergio Boccardi, Sergio Blanco y Silvia Koziol, por su colaboración en este emprendimiento.

A todos los familiares y amigos que con sus amables aportes ayudaron a que se realizara este proyecto.

Contenido

Tabla de figuras

Capítulo 1

Introducción

1.1. Motivación

La historia de la robótica moderna tiene su punto de partida en 1954 con la patente de George C. Devol, Jr., seguida de la instalación en 1959 del primer modelo de prueba"Unimate" en la planta de fundición inyectada de General Motors en Turnstead y la creación en 1961 de Unimation Inc. El tiempo transcurrido desde entonces ha contemplado un intenso desarrollo de la robótica y, en concreto, de la denominada robótica industrial, de tal forma que los robots, que llegaron a ser considerados el paradigma de la automatización industrial, se han convertido en nuestros días en un elemento más, aunque importante, de dicha automatización. [1]

De acuerdo a lo que expone la Real Academia Española, Robot es una "Máquina o ingenio electrónico programable, capaz de manipular objetos y realizar operaciones antes reservadas solo a las personas".[2]

La palabra fue empleada por primera vez en la obra "Rossum's Universal Robot" del autor checo Karel Capek en 1921. Asimismo, el término Robótica como disciplina, no fue abordado hasta 1942, cuando el escritor de origen Ruso, Isaac Asimov la utilizó en su obra "Runaround".[3].

A mediados del siglo XX, el estadounidense George Devol inventó un brazo primitivo que se podía programar para realizar algunas tareas. Más tarde en 1975, el también estadounidense Victor Scheinman desarrolló un manipulador polivalente flexible conocido como Brazo Manipulador Universal Programable, PUMA por sus siglas en inglés. Éste mecanismo era capaz de mover un objeto y colocarlo en un lugar deseado que estuviera a su alcance. A grandes rasgos, el concepto básico de multiarticulación empleado por el PUMA es la base de la mayoría de los robots actuales. [3]

El proyecto TATETI surge de la inquietud del grupo a explorar un campo en desarrollo, y de la satisfacción de generar un aporte y acercamiento de la robótica a la sociedad. La Robótica, como área de aplicación de la ingeniería, se encuentra en constante crecimiento y plantea un nuevo desafío con muchas incertidumbres, donde los límites están en la propia imaginación. Este desafío nos motivó a crear algo nuevo y creativo aplicando ingeniería, donde se utilizaran herramientas del área de la electrónica, diseño, mecánica, control, al igual que de la informática.

Otro de los objetivos de los integrantes del grupo es poder combinar todo el aprendizaje adquirido en los años de formación en la Universidad para lograr la completitud del proyecto.

Este proyecto surge de la necesidad del Museo Participativo Ciencia Viva quien desea disponer de una nueva experiencia interactiva que tenga una fuerte base tecnológica. Tiene como objetivo principal lograr atraer la atención de niños y adolescentes, para despertar el interés por la ciencia y la tecnología.

A su vez, se pretende poder cumplir con los objetivos generales de Ciencia Viva [4] que implican la creación de un ámbito permanente de orden socio-cultural,

1. que acreciente y mantenga vivo el interés de la Comunidad en la Ciencia y sus logros
2. que estimule particularmente en los adolescentes y jóvenes, una participación activa a través de muestras, concursos, intercambios y toda tarea que conduzca al desarrollo de la creatividad,
3. que propugne el establecimiento de canales eficientes para una interacción de la ciencia con otras esferas de la vida cultural del país,
4. que favorezca la creación y desarrollo de proyectos multidisciplinarios.

La propuesta planteada frente a esta necesidad, fue la creación de un dispositivo capaz de jugar al tatetí utilizando un brazo robótico para el movimiento de sus fichas. El brazo debe ser tal que sus partes eléctricas y mecánicas se encuentren a la vista del público, de ser posible.

1.2. Antecedentes

Se investigó y estudió proyectos de similares características que se pudieron encontrar en Internet y Facultad. Allí se encontró una diversa gama de brazos robóticos diseñados e implementados, cada uno de ellos con una anatomía y funcionalidades diferentes.

En particular, dentro de la Facultad de Ingeniería de la Universidad del a República se realizaron algunos proyectos de fin de carrera que implementan un brazo robótico, como ser el Proyecto UMANIS realizado en el año 2005[5].

A pesar que se encontraron tanto modelos comerciales como prototipos de estudiantes, se observó que todos ellos se basan en los mismos principios mecánicos y eléctricos. Por ejemplo, los motores utilizados eran en su mayoría servomotores o motores paso a paso y todos eran comandados mediante un microcontrolador o un microprocesador acompañados de un driver. [6], [7], [8], [9], [10].

Visto esto, se profundizó en la investigación y estudio de los diferentes tipos de motores y drivers utilizados en la automatización de procesos, consultando además diferentes catálogos de proveedores y hojas de datos[11], [12].

En paralelo, se investigó el comportamiento y la disponibilidad de diferentes tipos de sensores, microswitches, encoders y pulsadores necesarios para la implementación del sistema.

De este modo, se pudo tener una noción más clara de los componentes que se tienen disponibles en plaza.

Además, se estudió la posibilidad de la utilización de uno de dos microprocesadores particulares que el grupo tiene a su disposición: el Rabbit 2000 junto con el módulo RCM2200 y el Rabbit 3000 junto con el módulo RCM3365. Para ello, se adquirieron sus manuales de usuario y sus hojas de datos que disponen de toda la información necesaria para su utilización[13].

Lo que diferencia este proyecto del resto de los proyectos existentes conocidos es por una parte su estructura mecánica y tamaño diferente y por otra su aplicación: este brazo formará parte de un juego interactivo para niños.

1.3. Requerimientos

Se fijaron los siguientes requerimientos y algunas restricciones por parte del cliente para este sistema.

- El comportamiento del brazo debe asemejarse en lo posible al de un brazo humano.
- La tecnología y estructura del brazo deben ser visibles en la medida de lo posible.
- El juego debe tener una presentación amigable y sencilla al usuario, de modo que los niños puedan operarlo sin la ayuda de un mayor. Por este motivo también, debe tener un diseño robusto.
- El juego deberá poder funcionar en forma continua durante 8 horas por día durante 6 días a la semana.
- El mantenimiento no debe ser complicado, frecuente o caro. Los repuestos deben ser fáciles de conseguir en plaza. El grupo debe brindar al personal de Ciencia Viva una buena información técnica sobre el mantenimiento general del dispositivo.
- Económicas: Se dispone de un presupuesto acotado. Ciencia Viva aporta hasta U$S 1000 para la realización de este proyecto.
- El espacio físico del dispositivo es limitado y está determinado.
- Se deberán cumplir mínimos en las dimensiones del brazo, fichas y tablero de juego.
- En la mecánica: El brazo deberá ser capaz de cumplir todas las restricciones de comportamiento indicadas por el cliente como tiempo máximo de posicionamiento de las fichas (ocho segundos), limpieza del tablero y tiempos de respuesta a diferentes comandos entre otras. Estos tiempos se comentarán más adelante.
- El mantenimiento debe ser sencillo, de bajo presupuesto y debe estar claramente explicado por el grupo a los funcionarios de Ciencia Viva.
- Se debe garantizar que no se necesite realizar sustitución de piezas o reparaciones significativas al juego en un plazo de 3 años.

Capítulo 2

Definición del Sistema y Prediseño

2.1. Introducción

El proyecto TATETI es innovador, novedoso y no depende de proyectos previos. Sólo se disponía de una idea, que debía bajarse a tierra y pulirse, surgida en una charla en un bar y del interés de crear algo interesante y diferente. Desde esa idea de un juego de tatetí comandado por un brazo robótico se comenzó a trabajar y se terminó creando un complejo sistema mecánico y eléctrico comandado por un microprocesador.

El primer paso fue definir específicamente qué es lo que se pretendía hacer y cómo se podría hacerlo. Para ello, fue imprescindible pasar por una etapa de definición del sistema y prediseño.

El grupo dedicó buena parte del año que duró el proyecto al diseño del sistema. Fue en esos meses donde se definió el comportamiento general que iba a tener el juego y se presentó al cliente un diseño de punto de partida que fuera aceptado.

Este capítulo pretende introducir al lector en el proceso de diseño seguido en la primera etapa del proyecto. También se detallan las funcionalidades básicas del sistema, su comportamiento y estructura.

2.2. Descripción funcional del juego

El juego está compuesto por un mueble y un tablero. En el mueble, que no está accesible al usuario, va montado el brazo, un tablero de tatetí y dos rampas donde se almacenan las fichas de juego. En el tablero están instalados los botones e indicaciones luminosas con los cuales el usuario interactúa con el juego.

El usuario, al enfrentarse con el juego, puede decidir jugar una partida contra otro jugador, contra la máquina u observar un modo “demo” (máquina contra máquina). A partir de ese momento cada jugador va haciendo sus jugadas hasta que haya un ganador o empate.

En caso de tener que jugar la máquina, el juego estudia la partida y decide en qué posición ubica la ficha.

Cuando sea el turno del usuario, se le habilita un teclado donde puede indicar en qué posición quiere colocar su ficha dentro del tablero.

En todos los casos, el brazo es el encargado de tomar la ficha del jugador, que está ubicada a un lado del tablero, y colocarla en el lugar que se le indicó. Una vez que termine de realizar ese movimiento, queda a la espera de una nueva jugada.

Se indica mediante alguna señal luminosa si hubo o no ganador. Una vez culminada la partida, el brazo se encarga de retirar las fichas del tablero y devolverlas al lugar correspondiente.

2.3. Definición inicial del sistema y proceso de diseño

Se definió que el tamaño del tablero de juego sería de aproximadamente $90cm \times 90cm$ siendo cada casillero de $30cm \times 30cm$. En base a esto, se estudiaron las posibles estructuras y dimensiones del brazo, determinando que debía medir alrededor de un metro de largo.

Se decidió que las fichas serían pelotas de unos $14cm$ de diámetro. La definición de la forma y tamaño de las fichas, así como el modo y el lugar para depositar y retirar las fichas, no fue tarea trivial debido a que dependía directamente de la estructura de la mano y su implementación.

Seleccionadas las fichas de juego, quedó definida la forma que deberían tener las pinzas de la mano, o al menos, cómo se deberían tomar las fichas. A partir de allí se entró en el detalle del diseño del brazo y de la mano al igual que del mueble de juego.

Se decidió que el brazo pudiera moverse únicamente dentro de un plano horizontal paralelo al tablero de juego. Para ello, se utilizaron cuatro articulaciones cilíndricas, ubicadas en el hombro (base del brazo), el codo (ubicado en la mitad del brazo), la muñeca (conexión del brazo con la mano) y en las pinzas de la mano. El brazo puede rotar sus articulaciones de hombro y codo para mover la mano dentro del plano del tablero de juego, mientras que la mano, además de poder abrirse y cerrarse, puede girar de arriba a abajo gracias a la articulación de la muñeca. En definitiva, el movimiento del conjunto de brazo y mano tiene cuatro grados de libertad.

En la figura 2.1 se muestra un primer esquema de cómo se vería el conjunto del brazo y de la mano. Se agregó además una ficha de juego para tener una noción del tamaño del dispositivo mecánico.

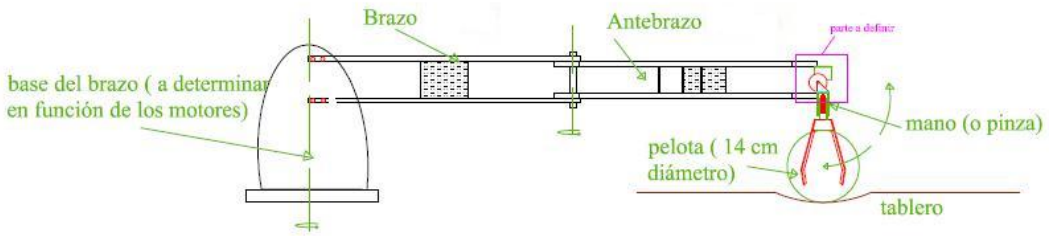


Figura 2.1: Detalle del brazo.

A continuación, se tuvo que realizar un estudio aproximado de los movimientos posibles del brazo sobre el tablero de juego para verificar que los mismos no interfirieran con las fichas colocadas en los casilleros, en particular, al retirar o depositar una ficha de juego.

Se realizaron algunos modelos del sistema en AutoCAD que oficiaron no sólo como método de estudio de los movimientos, sino también como que base para el diseño del mueble de juego[1].

En la figura 2.2 se muestra el primer diseño de cómo se verían el brazo y el tablero de juego montados en el mueble del juego. También se puede apreciar las canaletas inclinadas donde quedarían colocadas las fichas que no han sido utilizadas en la partida.

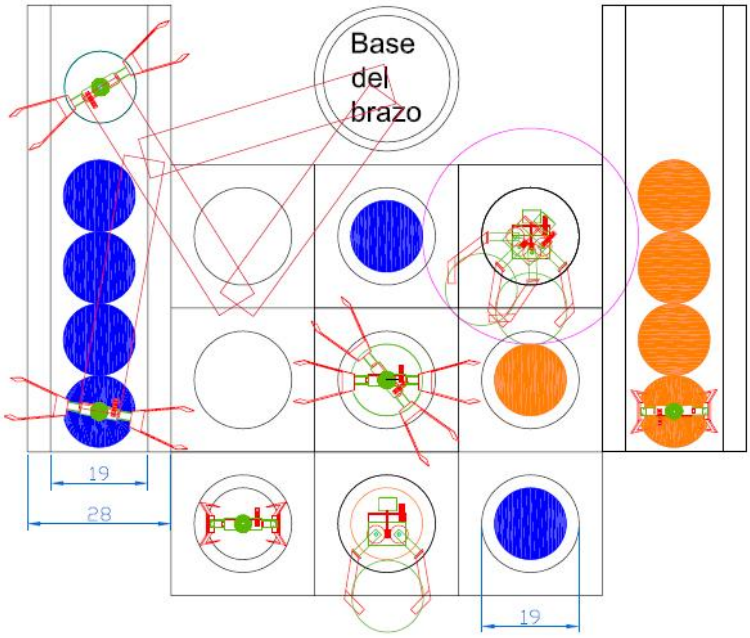


Figura 2.2: Bosquejo del brazo y el tablero de juego.

A su vez, en la figura 2.3 se muestra el detalle de la mano en sus posiciones de abierto y cerrado. Estos dibujos fueron el punto de partida para el diseño de la mano y pinzas.

[1]Si bien el grupo estuvo involucrado en parte del diseño, el mueble fue construido por el personal de Ciencia Viva

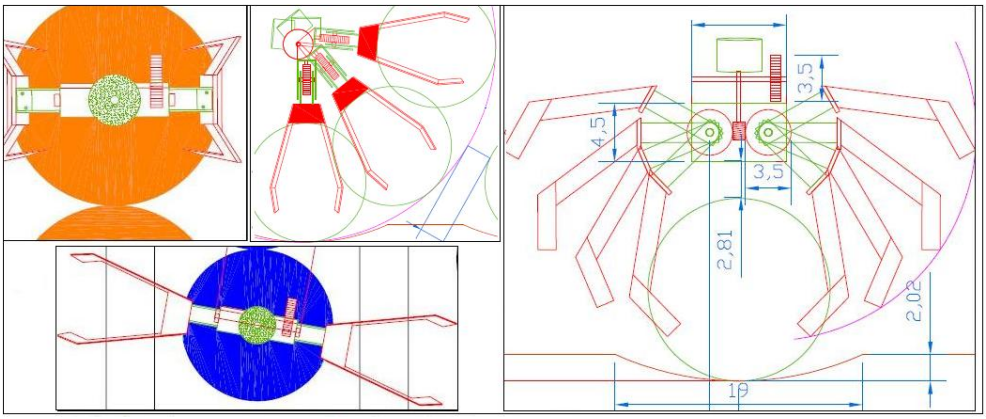


Figura 2.3: Detalle de la mano.

Definidas las articulaciones, se propuso inicialmente implementar los movimientos del brazo con motores paso a paso y los de la mano con motores de continua. Luego de evaluar la forma en la que se comandarían los motores se decidió utilizar un motor de continua únicamente para el movimiento de las pinzas, utilizando motores paso a paso para el resto de las articulaciones.

Luego se debía definir quién sería el encargado de comandar estos motores y llevar la lógica del juego. Debido a que el grupo disponía de algunos microprocesadores Rabbit, que son adecuados para implementar sistemas embebidos de esta índole, se decidió que el encargado de comandar estos motores sería un módulo de Rabbit RCM3365 mostrado en la figura 2.4, que contiene al microprocesador Rabbit 3000.

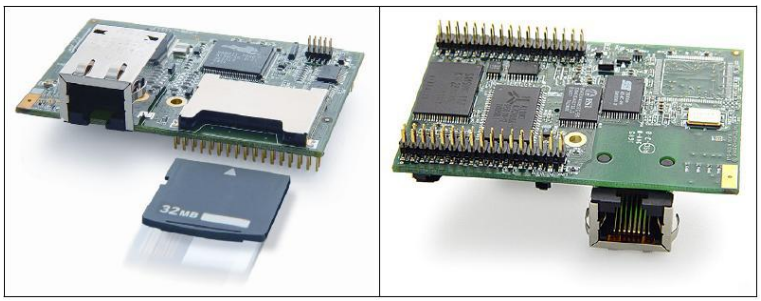


Figura 2.4: Módulo Rabbit RCM3365.

A continuación, se definió qué apariencia tendría el juego y cómo iban a interactuar con él los usuarios. Se tuvo en cuenta que los principales usuarios del juego serían niños y adolescentes, por lo que se buscó una interfaz sencilla y amigable.

La interfaz de usuario estaría compuesta por un tablero de nueve botones dispuestos como una matriz de $3x3$, cinco botones de mayor tamaño para comenzar y seleccionar los jugadores de una partida y un conjunto de señales luminosas para guiar al usuario en el transcurso del juego. Todos estos elementos, junto con el tablero de juego y el brazo robótico, estarían instalados en un mueble de madera. Inicialmente, la interfaz de usuario formaría parte del mismo mueble en donde se encontraría el brazo. Luego, por motivos

prácticos y de seguridad, se decidió separar la interfaz del mueble de juego y montarlo sobre un soporte independiente. En la figura 2.5 se muestra un primer diseño en AutoCAD de la interfaz de usuario.

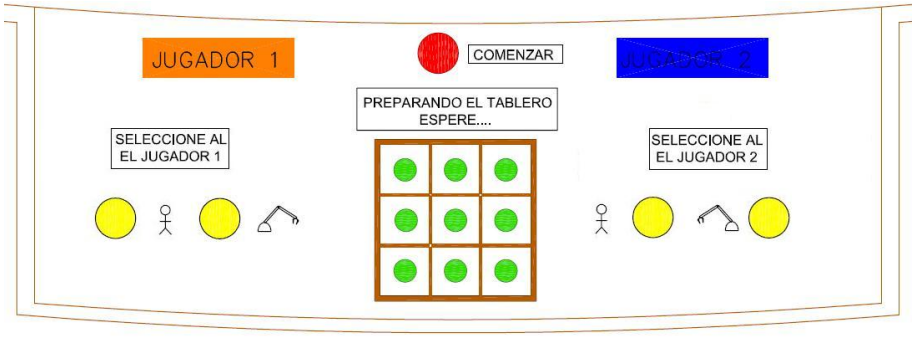


Figura 2.5: Bosquejo de la interfaz del juego.

Se propuso utilizar pulsadores de hongo para el tablero y señales luminosas con indicaciones de las diferentes situaciones en que se encuentre el juego. Para el control de todos estos dispositivos de interfaz con el usuario, en un principio se decidió utilizar un PLD que estaría en constante comunicación con el módulo Rabbit. Lamentablemente, luego de varias pruebas con PLDs del curso de *Diseño Lógico 1* de la Facultad de Ingeniería de la Universidad de la República, se dio de baja a esta idea debido a que los PLDs disponibles resultaron demasiado pequeños. Se decidió entonces que el microprocesador se encargue del control de la interfaz de usuario.

2.4. Estructura del sistema

Una vez definida la estructura general del juego y sus componentes, se profundizó en la definición y diseño del tipo y sistema de control a implementar. Se determinó que para el control de los motores sería necesaria una etapa de potencia independiente por motor a ser comandada por el microprocesador. A su vez, para poder tener una realimentación de la posición de los motores, se decidió utilizar encoders ópticos en cada articulación. Esta decisión se comentará en capítulos posteriores. La electrónica necesaria se diseñó en etapas posteriores, que se comentarán en los próximos capítulos.

En el esquema 2.6 se muestra la estructura implementada para el control del sistema.

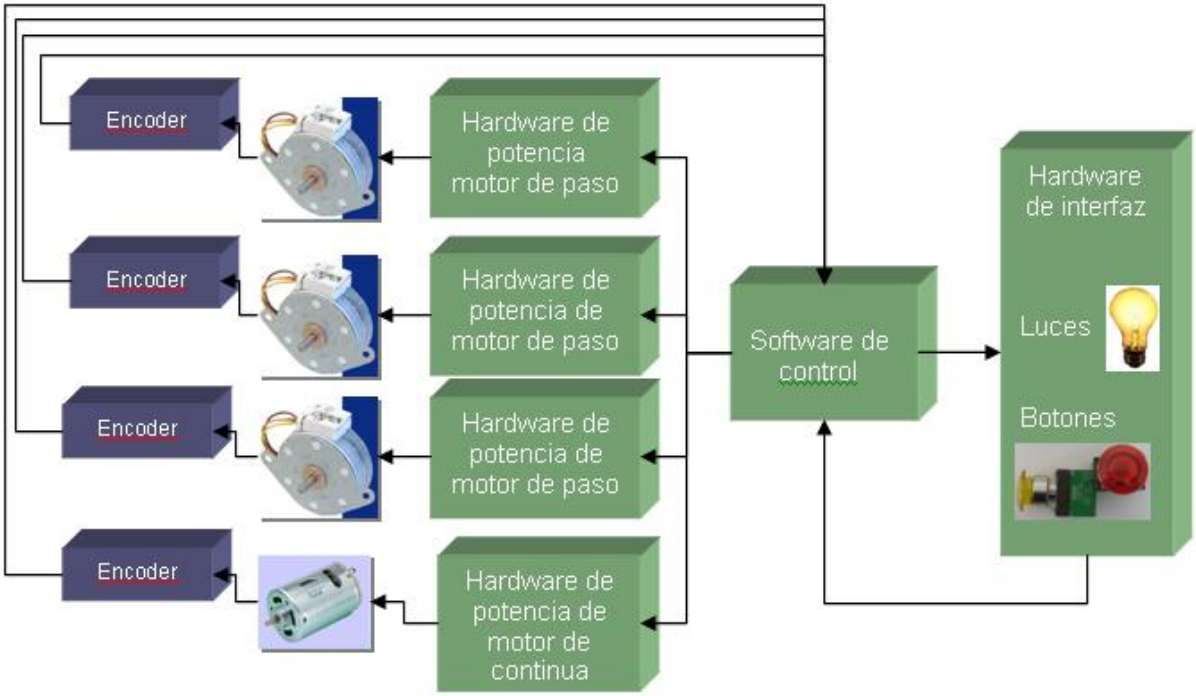


Figura 2.6: Esquema del sistema de control

Capítulo 3

Diseño y construcción mecánica

3.1. Introducción

En este capitulo se dará una descripción completa de la estructura y funcionamiento del dispositivo mecánico construido, sus componentes y como interactúan entre sí para lograr que el mecanismo se comporte de forma adecuada. También se harán comentarios del porqué de la estructura realizada y los inconvenientes encontrados.

Hay que destacar que todo el mecanismo fue construido con piezas encontradas en el mercado local, ya sea por partes retiradas de equipamientos en desuso, como por piezas hechas a medida. Este es un punto interesante dado que interesaba mostrar que no se necesitan piezas traídas del extranjero y de valor elevado para la construcción de la mecánica, sino que alcanza con el ingenio y la imaginación.

La primera dificultad encontrada fue el hecho que el grupo nunca se había enfrentado a un problema de estas características, donde el diseño mecánico representa un gran desafío. Se requirió de un tiempo considerable de investigación de proyectos similares en Internet y en la propia facultad. Con esta información, y luego de evaluar diferentes propuestas, se optó por una estructura que se adaptara a la función particular del juego, decidiéndose que el brazo se mueva dentro de un plano horizontal paralelo al tablero de juego y pudiera mover su mano hacia arriba y hacia abajo para manejar las fichas de juego.

Dentro de las propuestas descartadas, estaban la colocar una articulación cilíndrica adicional en el hombro o en el codo para elevar desde allí el resto del brazo. Estas opciones fueron rápidamente descartadas debido a la fuerza que debería soportar la articulación y su motor, así como a las limitaciones del mueble donde estaría ubicado.

También fue descartada la estructura del puente móvil pues su comportamiento y forma no se asemeja a la de un brazo humano.

A partir de la idea de la estructura, se realizó un bosquejo localizando sus componentes. Para una mejor comprensión, se hará referencia a las partes del mecanismo usando los nombres correspondientes al cuerpo humano. En la tabla siguiente se listan los principales componentes del dispositivo.

Articulación	Componente
Hombro	Soporte
Codo	Brazo
Muñeca	Antebrazo
Nudillos[1]	Mano

Desde un principio se consideró realizar un diseño "liviano" en peso y con el menor momento de inercia posible debido a que de ello depende directamente el esfuerzo que deben realizar los motores[2].

La figura 3.1 muestra la estructura del dispositivo terminada, donde a su vez se indica la nomenclatura utilizada.

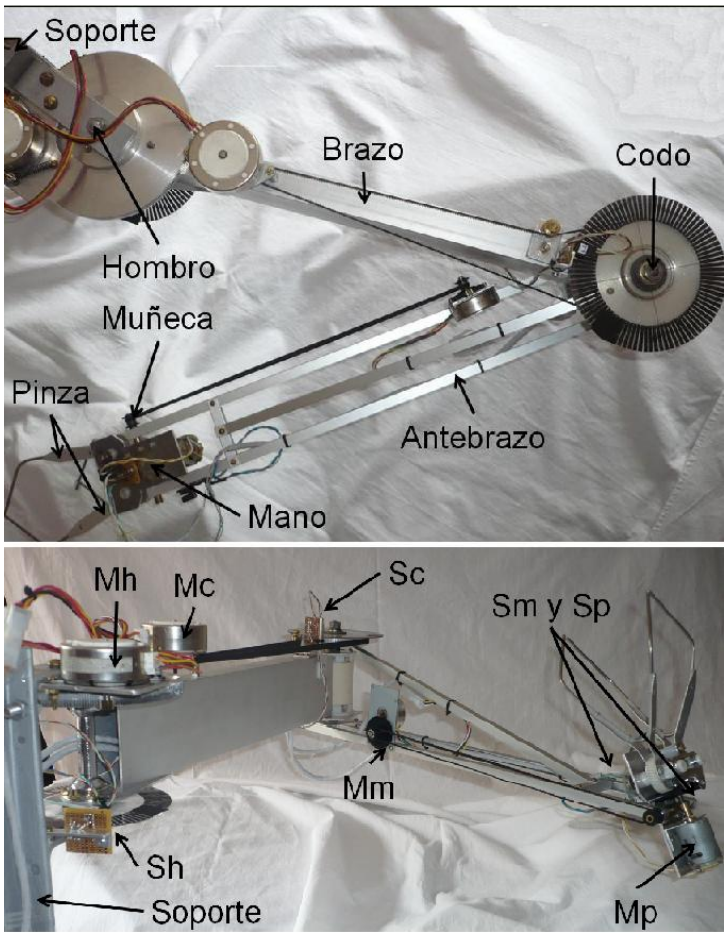


Figura 3.1: Estructura y articulaciones del brazo.

[2]Los motores fueron retirados de impresoras en desuso. El proceso de selección de motores se detalla en la sección siguiente

Mc	motor que genera el movimiento del hombro
Mc	motor que genera el movimiento del codo.
Mm	motor que genera el movimiento de la muñeca.
Mp	motor que genera el movimiento de las pinzas.
Sh	Sensor utilizado para el posicionamiento del hombro.
Sc	Sensor utilizado para el posicionamiento del codo.
Sm	Sensor utilizado para el posicionamiento del muñeca.
Sp	Sensor utilizado para el posicionamiento del pinzas.

3.2. Descripción de la estructura

A continuación se describe la construcción de los distintos componentes mecánicos, la selección de materiales, las restricciones de movimiento y dimensiones.

El mecanismo se compone de dos tramos de aluminio de $45cm$, teniendo un largo total de $90cm$. Por su gran tamaño y cantidad de partes independientes, se decidió construirlo de forma tal que fuera fácil de desarmar en partes bien diferenciadas. Basado en este principio, se llegó a un diseño más modular pero con mayor complejidad para su montaje.

Tal como se puede observar en las figura 3.1, el dispositivo cuenta con dos articulaciones cilíndricas verticales que permiten el movimiento sobre un plano horizontal. Una tercera articulación en sentido horizontal es la encargada del movimiento vertical de la mano mientras que la articulación de los dedos permite sujetar o soltar las fichas.

Estas articulaciones se encuentran ubicadas en el hombro (conexión soporte-brazo), codo (conexión brazo-antebrazo), muñeca (conexión antebrazo-mano) y pinzas (articulación de las dedos de la mano).

3.2.1. Soporte

El soporte está compuesto por una lámina de hierro y un parante. La lámina, que tiene $5mm$ de espesor, $13,3cm$ de ancho y $22cm$ de largo, sirve hacer de asiento de toda la estructura sobre el mueble, al que se fija con cuatro bulones. El parante, que es un perfil de hierro en forma de "T" de $2cm$ x $4cm$, $35cm$ de largo y $0{,}5cm$ de espesor, va soldado a la lámina y sirve para mantener el dispositivo a la altura deseada. Sobre él van atornilladas dos planchuelas de hierro en forma de "L" de $8cm$ x $11cm$ x $3cm$ que sujetan la articulación y el motor del hombro.

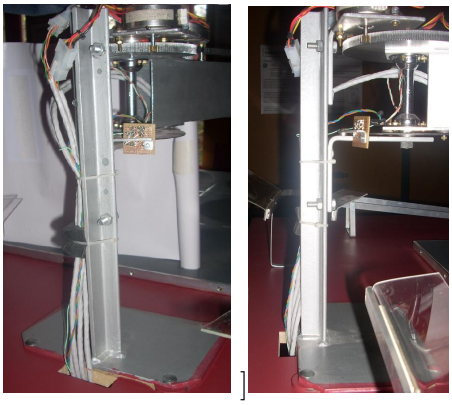


Figura 3.2: Vista del soporte

3.2.2. Hombro

La articulación del hombro es la encargada de realizar el giro del brazo con respecto al soporte. Para la selección de esta pieza, se tuvo en cuenta que debe soportar todo el peso del dispositivo mecánico, por lo que se utilizó una masa de bicicleta de acero rodado 26. El eje de la misma se encuentra solidario al soporte mientras que su parte móvil es solidaria al brazo. Sobre este último se encuentra el engranaje que mueve la articulación.

Este engranaje de aluminio fue hecho a medida y tiene un diámetro de $14cm$ por $1cm$ de espesor. Las dimensiones de esta pieza surgen del estudio realizado al motor[3] y de las restricciones de fuerza-tiempos que se nos plantearon, utilizando una reducción de $14:1$.

Dada la forma en que se implementó la articulación y se colocó el motor, el brazo tiene 225^o de apertura en su movimiento. La realimentación de la posición del brazo es realizada por el conjunto disco-encoder, donde el disco se graduó cada 2^o y se ubicó solidario a la masa de bicicleta.

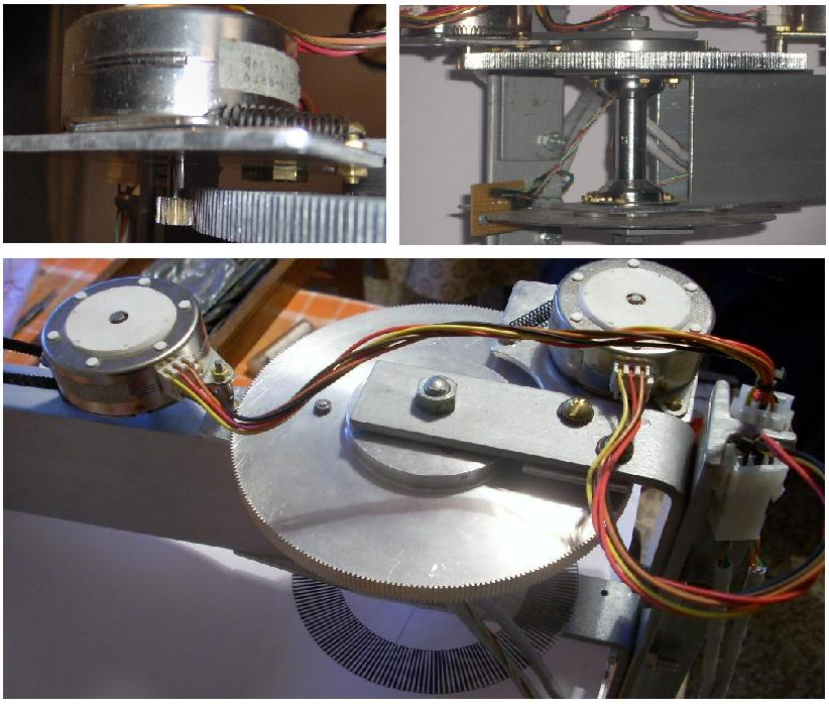


Figura 3.3: Estructura y articulaciones del hombro.

3.2.3. Brazo

El brazo se extiende desde la articulación del hombro hasta el codo. Está formado por un perfil de aluminio hueco de sección rectangular de $8cm$ x $2,5cm$ y $39cm$ de largo. Se utilizó una estructura rígida de estas características para garantizar que la misma no cediera o se deformara al encontrarse el brazo en movimiento. Se debe tener especial cuidado cuando el ángulo entre el brazo y antebrazo es 90^o, debido a que en esta situación la torsión sobre el brazo es máxima.

Sobre el brazo se encuentra montado el motor utilizado para generar el movimiento del codo[4], el cual transmite el movimiento de su eje a la articulación del codo por medio de una correa dentada de goma de $6mm$ de espesor.

[3]Por detalles sobre el modelo del motor utilizado, referirse a las figuras 3.9 y 3.10
[4]Por detalles sobre el modelo del motor utilizado, referirse al apéndice B

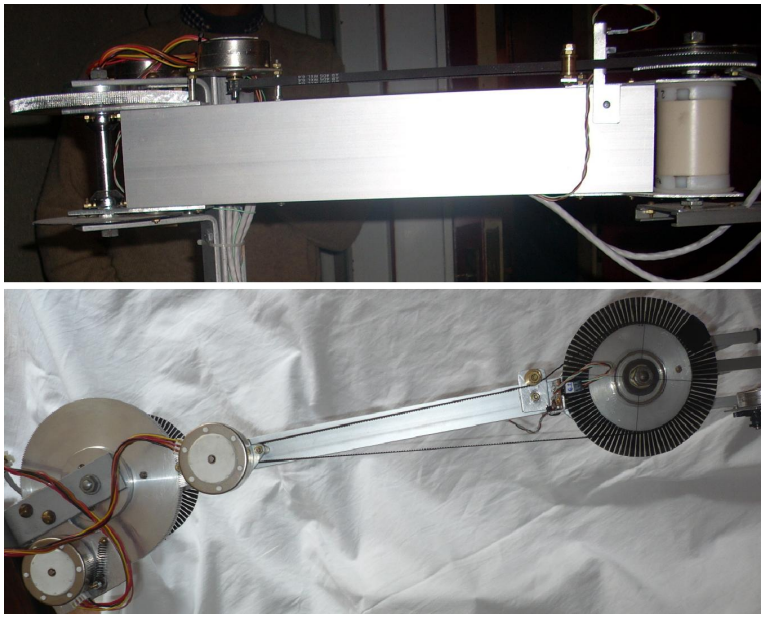


Figura 3.4: Estructura del brazo.

3.2.4. Codo

El codo está formado por una masa de bicicleta rodado 14. Su eje es solidario al antebrazo mientras que su parte móvil al brazo. Para mover esta articulación, el motor colocado en el brazo hace girar, por medio de una correa dentada retirada de un escáner, una polea de aluminio de 8,5cm de diámetro por 1cm de espesor solidaria al antebrazo, como se mencionó anteriormente. Las dimensiones de esta polea surgen del estudio de toque realizado al motor, utilizando una reducción de 8,5 : 1.

Esta articulación tiene un ángulo de apertura de 230º, simétrico con respecto al brazo. La realimentación de la posición del antebrazo también es realizada por el conjunto disco-encoder, donde el disco se graduó cada 2º y se ubicó solidario a la polea mencionada.

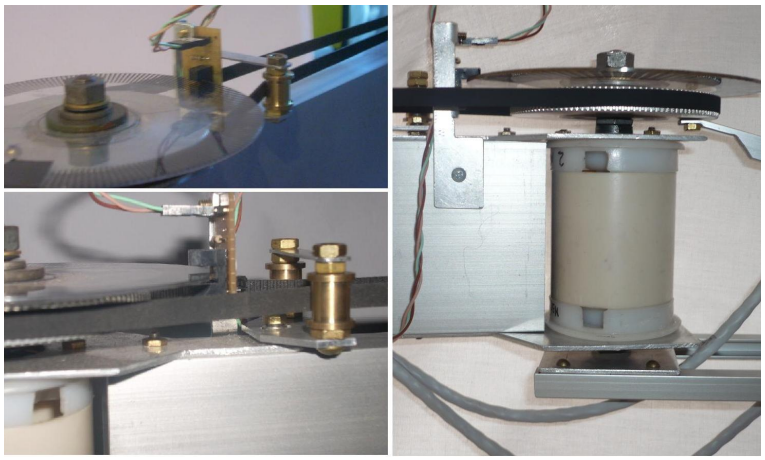


Figura 3.5: Estructura y articulación del codo.

3.2.5. Antebrazo

Para el antebrazo se busco un diseño que fuera "liviano" pero que no comprometiera la rigidez del mismo. Por este motivo, se armó con una estructura de "esqueleto", formado por tres perfiles de aluminio en forma de "U" de $1cm$ de lado y $48cm$ de largo para conseguir una distancia entre ejes de $45cm$.

El antebrazo se extiende desde la articulación del codo hasta la muñeca y sobre éste se encuentra colocado el motor que genera el movimiento de la muñeca[5].

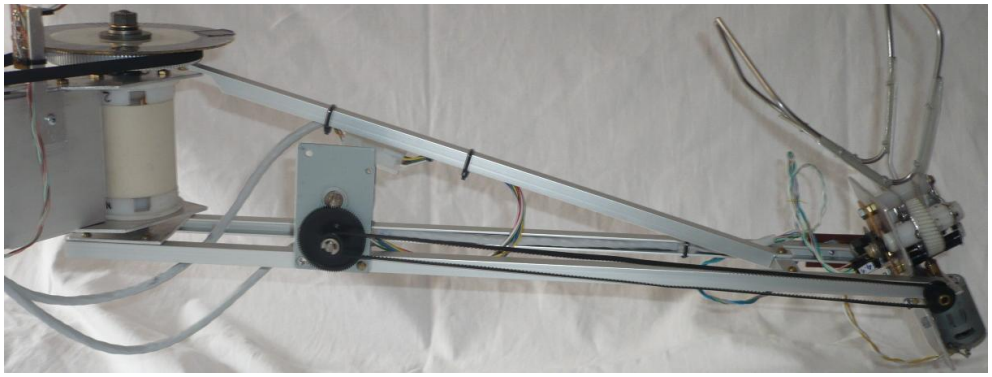


Figura 3.6: Estructura del antebrazo.

3.2.6. Muñeca

El motor de la muñeca se encarga de generar el giro de la mano con respecto al antebrazo. Este movimiento se logra por medio de una correa dentada de goma colocada entre la polea de $1cm$ de diámetro utilizada para la reducción del motor y una polea de plástico de $1cm$ de diámetro ubicada en el eje de la mano. Estos componentes fueron encontrados en el juego de transmisión mecánica de un escáner y fueron adaptados para poder montarlos al antebrazo. La reducción total que se logra entre el eje del motor y la articulación es de 5,4 : 1. Para esta reducción no fue necesario realizar un estudio detallado, pues no se requería de grandes esfuerzos para poder mover la mano dada la forma en la que se optó construirla.

El ángulo de giro obtenido para la articulación de la muñeca fue de 200^o. Este movimiento está controlado por la combinación de un encoder óptico ubicado en el antebrazo y un disco graduado con tres posiciones de interés ("mano hacia abajo", "mano en posición horizontal" y "mano hacia arriba").

[5]Por detalles sobre el motor de la muñeca, referirse al apéndice B

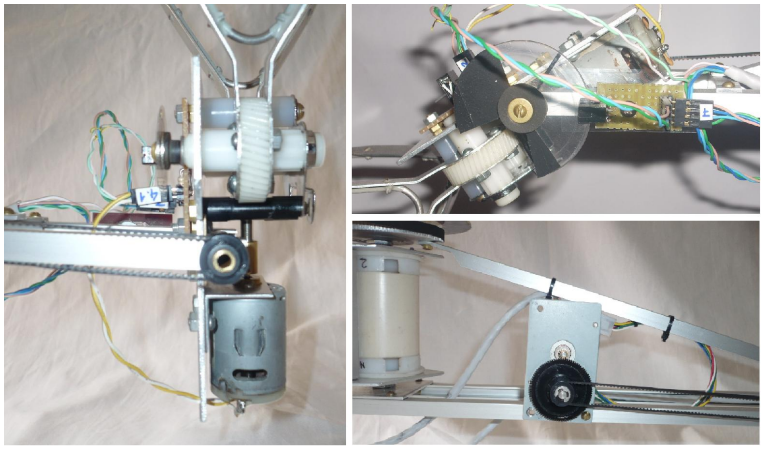


Figura 3.7: Estructura y articulación de la muñeca.

3.2.7. Mano

La mano esta formada por un motor de continua, que por medio de dos engranajes y un tornillo sinfín, realiza el movimiento de apertura y cierre de las pinzas. Su estructura se basa en una lámina de aluminio de $2mm$ de espesor sobre la cual se montan todos los componentes anteriormente mencionados. Es interesante destacar que el juego de engranajes fue extraído de una licuadora en desuso y adaptado a la aplicación mientras que el motor se retiró de un sacador de pelo. La figura 3.8 muestra esta estructura en detalle.

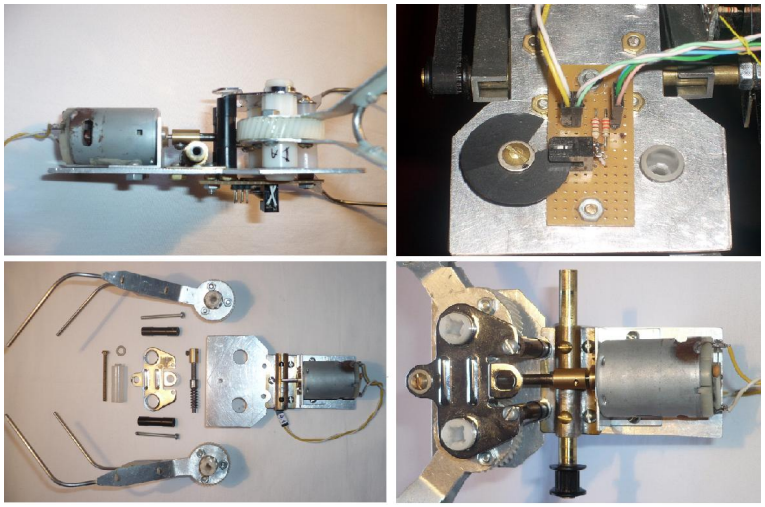


Figura 3.8: Estructura de la mano.

Para su construcción se partió de la forma de las fichas que debe mover y la necesidad de poseer una rigidez tal capaz de soportar la presión que se realizará sobre las mismas. Debido a esto se optó por armarlas con alambre de acero de $3{,}75mm$ de diámetro. La determinación de su forma fue obtenida tras varios ensayos con las fichas de juego para asegurar un correcto agarre de las mismas.

El recorrido total de las pinzas es de 120º. Esta articulación, al igual que el resto, se controla por medio de un conjunto disco-encoder utilizado para indicar si la pinzas se encuentran abiertas o cerradas.

La tabla 3.1 muestra las características más importantes de cada articulación.

Articulación	Motor	Alimentación (V) Marcha/Hold	Transmisión
Hombro	PM55L-048	24/12	14:1
Codo	PM55L-048	24/12	8.5:1
Muñeca	FDK SB40	12/5	5.4:1
Pinzas	Thomson	12	40:1

Tabla 3.1: Tabla de características para cada articulación.

3.3. Selección y prueba de motores

Antes de comenzar con la búsqueda de los motores a utilizar, se estudió el comportamiento de los diferentes tipos de motores eléctricos que se utilizan para este tipo de dispositivos. Se pudo averiguar que los motores de continua son más livianos y presentan mejores torques, pero los motores paso permiten controlar su velocidad más fácilmente via microprocesador.

Basados en esto, se decidió colocar motores de paso en las articulaciones del hombro y codo, que tenían cotas mínimas de precisión mayores. Se optó poner también este tipo de motores en la muñeca pues se consiguió una pieza que ya contenía la reducción. Para las pinzas el motor es de continua pues interesa hacer que la mano sea liviana y no es necesario tener gran precisión en el movimiento.

A continuación se consiguió una buena cantidad de motores de paso sacados de impresoras y escáneres rotos o en desuso, se listaron, se buscaron sus hojas de datos de los mismos y se relevó su comportamiento del torque en función de la frecuencia.

Para poder relevar esta característica, se implementó dentro del PLD de la placa del curso de Diseño Lógico 1, un controlador sencillo con frecuencia y sentido de giro configurable, que fue conectado a una etapa de potencia Darlington que el grupo del proyecto ITDOAS (del año 2007) cedió amablemente. Luego, con la ayuda de un "Torquímetro", que consiste en acoplar al eje del motor un cilindro plástico con una cuerda, se colgaron pesas de diferente masa y se testeó si el motor era capaz de levantarlas. Se realizó una aproximación lineal de las curvas obtenidas experimentalmente y se compararon con la hoja de datos del fabricante, para verificar que el motor estuviera funcionando correctamente y efectivamente tuviera el torque nominal indicado.

Una vez terminado estos estudios, se decidió utilizar motores de paso PM 55L-048[6] para el hombro y codo, pues poseen una respuesta torque-velocidad-consumo adecuada para nuestra aplicación. Con la ayuda de estos datos y restringiendo cada articulación a los

[6]Referirse al apéndice B por más detalles

parámetros correspondientes (tiempo-torque) se obtuvieron los valores de las reducciones para cada articulación.

La gráfica 3.9 muestra la curva característica del motor seleccionado para el hombro y codo. Por su parte, la figura 3.10 muestra las curvas características de cada articulación relevadas con la reducción.

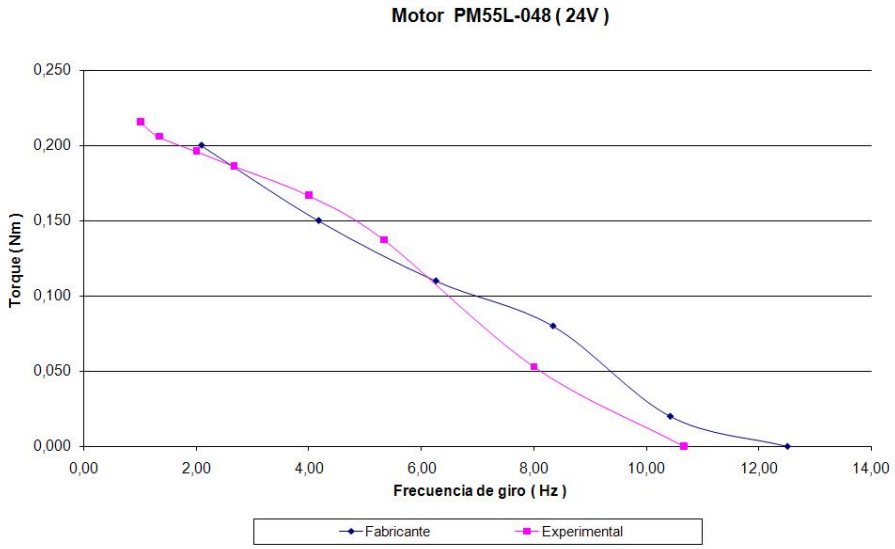


Figura 3.9: Curva Torque-Frecuencia de giro del motor elegido.

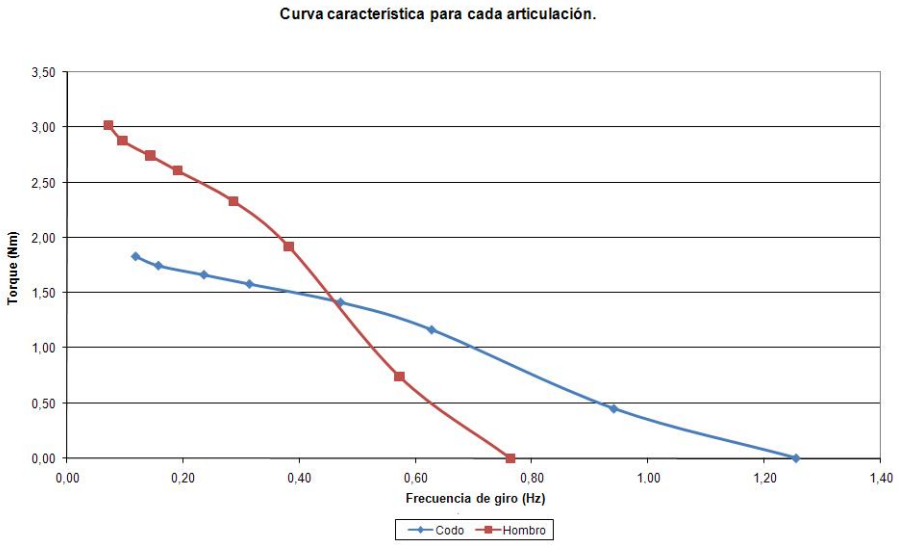


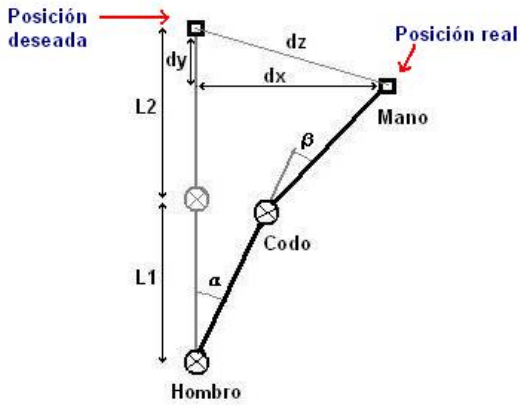
Figura 3.10: Curva resultante luego de la reducción para cada articulación.

Dado que las articulaciones del hombro y codo son las encargadas del posicionamiento del dispositivo dentro del tablero de juego, son las que determinan la exactitud del movimiento. Analizando los diferentes componentes de la implementación mecánica de estas articulaciones es posible determinar la máxima precisión alcanzable por el mecanismo sin considerar la utilización de los encoders.

Los motores PM55L-048 giran un ángulo de 7,5º por paso. Dividiendo este valor por la reducción colocada se obtiene el ángulo resultante por paso para cada articulación. En la siguiente tabla se detallan los valores obtenidos.

Articulación	Paso del motor (º)	Reducción	Paso de la articulación (º)
Hombro	7.5	14:1	0.54
Codo	7.5	8.5:1	0.98

Como resultado, la articulación de hombro logra una precisión de 0,54º mientras que el hombro 0,98º. Los efectos de esta limitación en los movimientos se aprecian en mayor medida cuando el mecanismo se encuentra totalmente estirado. En la figura 3.3 se pretende ejemplificar el efecto descrito y a continuación se presenta el cálculo de la desviación máxima experimentada por la mano.



Se deduce que

$$dx = L_1 sen(\alpha) + L_2 sen(\alpha + \beta)$$

$$dy = L_1 + L_2 - [L_1 cos(\alpha) + L_2 cos(\alpha + \beta)]$$

y sabiendo

$$L_1 = L_2 = 45cm$$

$$\alpha = 0{,}54^o$$

$$\beta = 0{,}98$$

se tiene que $dx = 1{,}62cm$ y $dy = 0{,}02cm$ por lo que $dz = 1{,}62cm$. Esto indica que en el peor caso el mecanismo se moverá de a pasos que distan $1{,}62cm$ entre si.

En el funcionamiento real del juego, a este error se le suman otros factores como los juegos entre los engranajes y poleas, las vibraciones y las transmisiones entre otros.

Capítulo 4

Hardware de control de motores

4.1. Selección de los circuitos de control de los motores

Una vez determinados los motores involucrados en el movimiento del brazo y sus características, se pasó a diseñar el hardware de potencia necesario para el control y alimentación de los mismos. Se investigó en Internet qué etapa de potencia y protección era necesaria para poder conectar a un microprocesador un motor paso a paso de cuatro hilos y controlarlo. Se pudo observar que dados los requerimientos eléctricos de nuestra aplicación, era más práctico trabajar con integrados ya diseñados para realizar este tipo de controles que armar circuitos con componentes discretos. Se encontró que existen integrados especialmente diseñados para controlar motores paso a paso los cuales implementan la lógica de control internamente.

En particular, se decidió trabajar con los integrados L297 y L298N, que son un controlador de motor paso a paso y un puente H doble respectivamente. El circuito base en el cual fueron montados estos integrados fue el sugerido en sus hojas de datos, lo que resultó ideal para nuestra aplicación. Al mismo se le agregaron algunas funcionalidades, como se detalla en la subsección 4.2.

El único inconveniente fue que estos circuitos integrados no se encontraban disponibles en plaza, y a pesar de que se pudo ubicar un comercio en Buenos Aires, Argentina[1] que los vendía a precio razonable, conseguirlos y comprarlos atrasó un poco la tarea de armado de la electrónica de control de motores.

Luego, se estudió qué circuitos eran necesarios para controlar motores de continua. La idea general era conectar el motor a un puente H comandado por una señal PWM, cuyo valor de continua determinara la velocidad de giro.

Luego de un poco de investigación y estudio de algunos integrados que realizaran este tipo de control, se determinó que el integrado LMD18200T era ideal para esta aplicación porque, además de realizar el control esperado posee un sensado de sobrecorriente para la protección del motor. Al igual que para los motores de paso, se partió del circuito sugerido en la hoja de datos del integrado al que se le realizaron algunas modificaciones, como se detalla en la sección 4.3.

Este integrado también se tuvo que traer desde Buenos Aires debido a que no se en-

[1] Casa de componentes electrónicos, **GM Electrónica S.A.**, sitio web http://www.gmelectronica.com.ar/

contró en el mercado local. Por este motivo se compraron dos chips para prever posibles errores.

Definidos los integrados para el control de los motores, se pasó a la etapa de diseño de los circuitos e implementación de los PCB para luego enviar a fabricar las placas.

4.2. Placa de potencia para el control de motores de paso

Como se mencionó anteriormente, para el diseño del circuito de control de motores paso a paso, se partió del circuito recomendado en la hoja de datos de los integrados L297 y L298N mostrado en la figura 4.1.

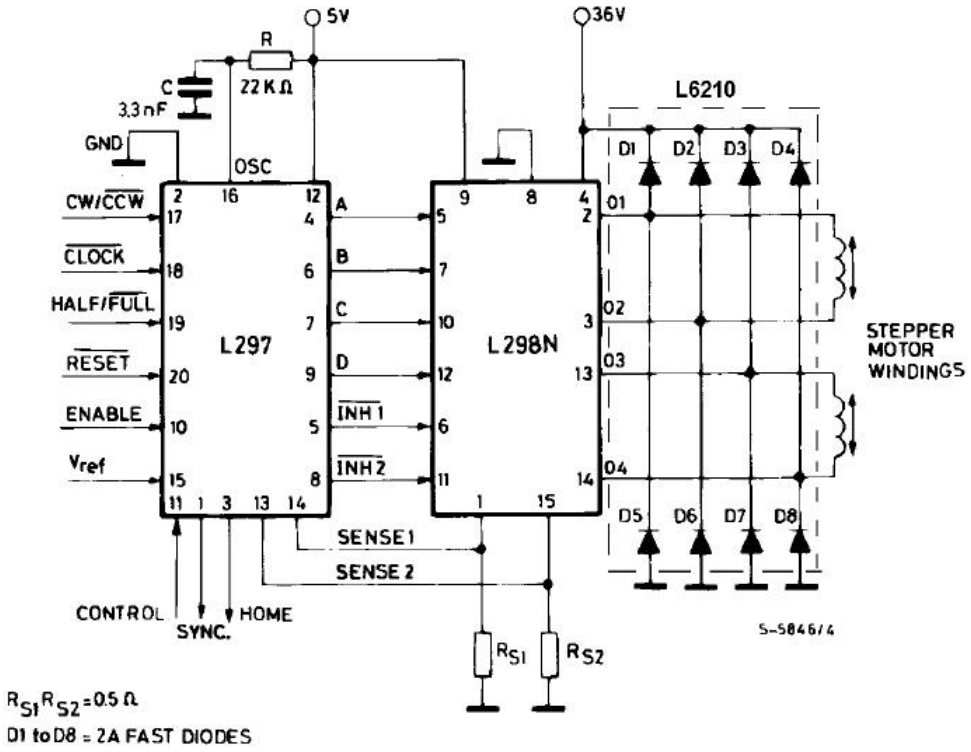


Figura 4.1: Circuito sugerido en la hoja de datos del integrado L297.

Se puede observar en la figura anterior que el integrado L297 es quien recibe las señales de control y comando desde el exterior y envía los "bits" de control A, B, C, D al integrado L298N quien se encarga luego de entregar la corriente y polaridad necesarias para el accionamiento de las bobinas del motor que se encuentre conectado. Como se detalla en el apéndice B, los motores se utilizan conectados de forma bipolar lo que implica que se dispone de dos bobinas por motor.

4.2.1. Integrado L297

El controlador de motores de paso L297[14] genera cuatro señales de control para motores bipolares de dos fases y motores unipolares de cuatro fases en aplicaciones de control.

Para comprender mejor el funcionamiento del controlador L297[2], se resume a continuación el significado de las señales de control más importantes.

- **CW/CCW**\: Señal de control que indica la dirección de giro del motor.
- **CLOCK**: Reloj de paso. Un flanco de subida en esta señal hace que motor se mueva un paso.
- **HALF/FULL**\: Cuando se encuentra activa selecciona la operación de medio paso, mientras que si se encuentra en nivel bajo selecciona la operación de paso entero.
- **RESET**\: Entrada de reset. Un nivel bajo en esta entrada lleva las señales A, B, C y D a un valor conocido (ABCD=0101).
- **ENABLE**: Entrada de habilitación del chip. Cuando se encuentra inactiva (nivel bajo) las señales de salida se se llevan a cero.
- **VREF**: Referencia de voltaje para el circuito de chopeado. Un votaje aplicado en esta entrada determina el pico de corriente de carga.
- **CONTROL**: Define la acción del chopeado. En nivel bajo, el chopeado actúa sobre INH1 e INH2, mientras que en nivel alto sobre las fases A, B, C y D.

El controlador recibe las señales de reloj, dirección y modo de un microcontrolador o similar, y genera las señales de control para la etapa de potencia posterior. Esto implica que debe trabajar siempre en conjunto con una etapa de potencia.

Las funciones principales son un traductor que genera la secuencia de fases para los motores, y un circuito de chopeado de PWM dual que regula la corriente en los bobinados del motor. El traductor genera tres secuencias diferentes que se seleccionan con la entrada **HALF/FULL**\. Estas son la secuencia *Normal* (dos fases energizadas), *Wave Drive* (una fase energizada) y *Half-Step* (alternadamente las otras dos). A su vez el L297 genera dos señales de inhibición que se utilizan para implementar los últimos dos modos[3].

4.2.2. Integrado L298N

El circuito integrado L298N[15], con encapsulado del tipo MULTIWATT15, es un puente H doble diseñado para aceptar niveles lógicos TTL estándar y manejar cargas inductivas como relés, solenoides y motores de paso de corriente continua. Puede ser conectado a tensiones de hasta $46VDC$ y puede entregar hasta $2A$ por canal. Posee dos señales de habilitación que permiten habilitarlo o deshabilitarlo en forma independiente a las señales de control. A su vez, dispone de una segunda entrada de tensión permitiendo a la lógica de control funcionar a un voltaje inferior.

La figura 4.2 muestra un diagrama de bloques del integrado junto con las señales de control.

[2]Por más detalles de las señales de control del integrado, referirse a la hoja de datos [14]

[3]Más detalles del comportamiento de estos modos de funcionamiento se puede encontrar en [14]

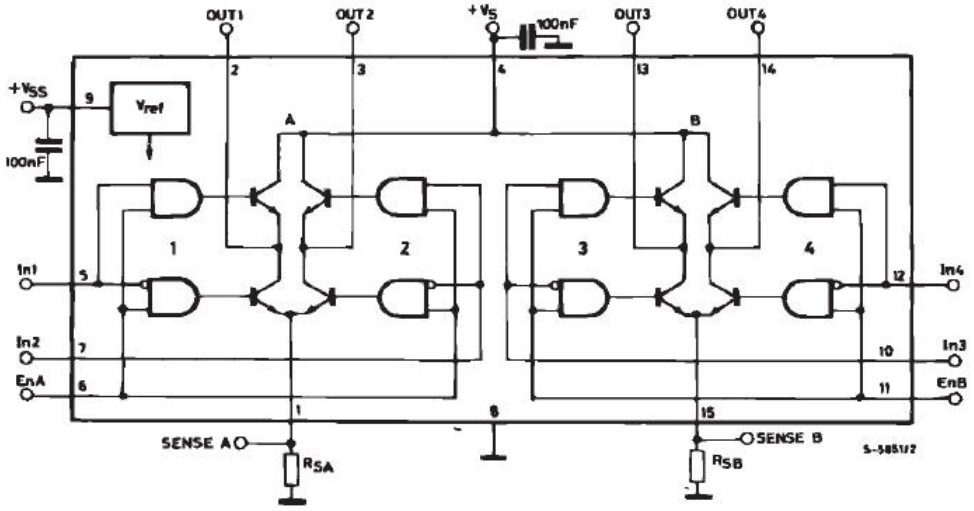


Figura 4.2: Diagrama de bloques del integrado L298.

Cada puente es manejado por cuatro compuertas cuyas entradas son las señales $In1$, $In2$, EnA, y $In3$, $In4$, EnB. Las entradas In setean el estado del puente cuando la entrada En se encuentra activa. Un nivel bajo en la entrada En deshabilita el puente.

4.2.3. Descripción del circuito implementado

El circuito de esta placa fue diseñado de manera de permitir a su usuario determinar el valor de las entradas **CONTROL** y **HALF/FULL** a través de jumpers. Para nuestra aplicación se mantendrá a la señal **HALF/FULL** en nivel bajo y **CONTROL** en nivel alto seleccionando de esta manera el modo *Normal* de funcionamiento, donde siempre se encuentran energizadas dos fases del motor y se controlan solo las fases ABCD. En este modo, las señales **INH1** e **INH2** se mantienen inactivas (en nivel alto) lo que se corresponde con la habilitación permanente de los puentes H del integrado L298.

El comportamiento del ciclo en el traductor se muestra en la figura 4.3 donde la secuencia de cuatro bits indica el estado de las salidas A, B, C y D. Este modo se selecciona con la señal **HALF/FULL** en cero y cuando el traductor se encuentra en un estado impar.

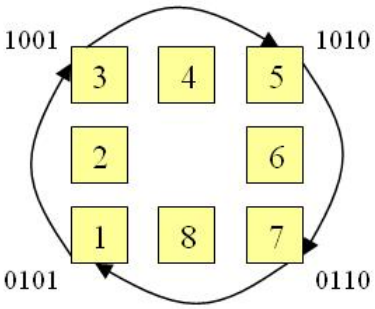


Figura 4.3: Ciclo lógico del traductor.

La entrada **ENABLE** se mantiene siempre habilitada con un *pull-up*. El control de habilitación y conmutación de niveles de tensión se realiza en un circuito externo para mantener al motor siempre alimentado logrando trabarlo cuando no se encuentra en movimiento. De esta manera se evita sobrecargar y en consecuencia recalentar los bobinados del motor.

A pesar de que no resulta necesario controlar la señal de **RESET**, se mantuvo accesible para poder ser manejada externamente. En el cableado definitivo esta señal quedó conectada a GND.

En síntesis, tres son las señales de control que se comandan desde el microprocesador:

1. **CLK** :Una señal conectada a la entrada de **CLOCK** del L297. Con cada pulso que se da a esta señal se mueve al motor en un paso.

2. **DIR** : Una señal de dirección conectada a la entrada **CW/CCW** para indicar el sentido en que se debe mover al motor.

3. **HAB** : Una señal de habilitación que permite conmutar la tensión de alimentación del motor entre dos valores: uno alto para el funcionamiento del motor y uno bajo para mantenerlo energizado pero en estado de reposo.

Para lograr el objetivo de la señal **HAB**, se implementó una realimentación para el control del voltaje de alimentación. Cuando el motor se encuentra deshabilitado pero debe quedar trabado en una posición, se reduce su tensión de alimentación a un valor menor (en el caso de estos motores se seleccionó en $5V$) para evitar que se recaliente y dañe[4].

Las hojas de datos recomiendan utilizar un puente de diodos externo a la salida del L298 cuando se manejan cargas inductivas y diodos Schottky rápidos de $2A$ preferentemente. Esta solución permite manejar hasta $3A$ en DC y hasta $3{,}5A$ para picos de corriente repetitivos. Esto es más que suficiente pues los motores utilizados consumen en promedio, alrededor de $1A$.

[4]Los motores de paso recalientan sus bobinados si los mismos se mantienen energizados por mucho tiempo y con un voltaje alto

El circuito implementado se puede observar en la figura 4.4 donde se muestra una simulación en Multisim del mismo.

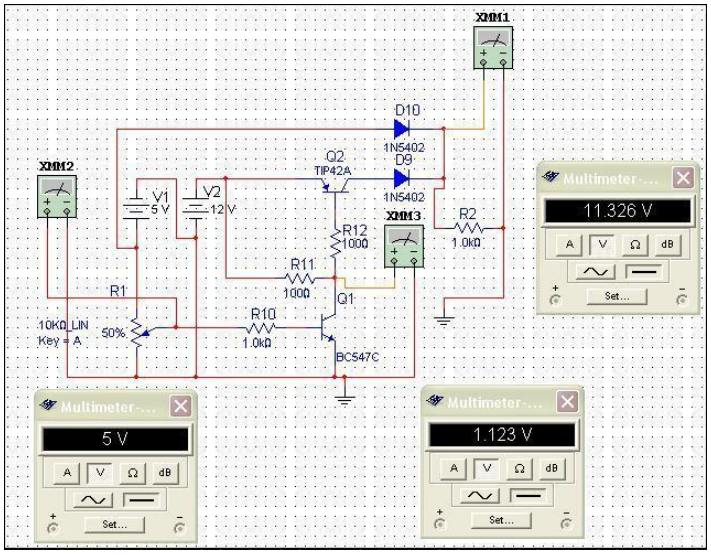


Figura 4.4: Simulación del circuito de conmutación de tensión de alimentación.

Cabe aclarar que las tensiones $V1$ y $V2$ del diagrama anterior deben cumplir la relación $V1 < V2$ pero no tiene que ser necesariamente $5V$ y $12V$ respectivamente. Para el motor del hombro, motor que debe realizar el mayor esfuerzo, se utilizarán $V2 = 24V$ y $V1 = 5V$, mientras que para los motores del codo y de la muñeca se utilizarán $V2 = 12V$ y $V1 = 5V$.

Por más detalles del circuito diseñado, referirse al esquemático C.3 ubicado en el apéndice C.

4.2.4. Diseño del PCB

Definido y revisado el circuito, se procedió a diseñar el PCB necesario para la fabricación de las placas. Este diseño se realizó y modificó varias veces tratando de llegar a un diseño óptimo que facilitará su armado y manejo. También se procuró que ocupara el menor espacio posible para abaratar los costos, pero cumpliendo con las restricciones de distancias y tamaños mínimas para que funcione correctamente.

Se decidió utilizar borneras con tornillos para la entrada y salida de las señales para asegurar la sencilla interconexión con el resto del sistema. El diseño del PCB se armó en dos capas (layers *bottom* y *top*), y se tuvieron que realizar algunos cruces de lado a lado para poder pasar y continuar las pistas (en estos cruces se suelda un pequeño cable de cada lado para conectar eléctricamente las pistas). También se tuvo que tener especial cuidado para evitar que las pistas que llegan a los pads de las borneras quedaran en la capa superior, al igual que los pads de otros componentes. Como se fabricaron las placas más económicas, que no implementan la unión eléctrica en los agujeros de la misma, las pistas que llegan a los pads de componentes que no pueden ser soldados de ambos lados deben mantenerse en la capa inferior. Esto llevó a que se compraran zócalos en tiras independientes para integra-

dos con encapsulados tipo DIP20, y que los mismos se soldaran de ambos lados de la placa.

El PCB final de la placas de potencia para el control de los motores de paso se puede observar en la figura C.4 del apéndice C.

4.2.5. Armado y pruebas de las placas diseñadas

A pesar de que se necesitan tres placas de potencia para controlar a los tres motores de paso que conforman el brazo, una vez diseñado y revisado el PCB, se mandó a fabricar solo una de ellas. El impreso se fabricó con fibra de vidrio, se le agregó un material protector antioxidante para cubrir las pistas y la serigrafía del impreso teniendo así bien identificados los componentes. Se consideró conveniente armar una, probarla completamente y realizar las modificaciones necesarias en el diseño antes de mandar fabricar las dos restantes.

Previo a comprar y montar los componentes, se verificó que sus terminales entraran en los agujeros hechos en la placa. Una vez verificado esto, se realizó una compra grande de componentes para armar las tres placas de potencia y se montaron en la placa terminada.

Se realizó una prueba intensiva de la placa armada utilizando un motor de paso de $24V$ que se tenía de repuesto. Se probó con diferentes tensiones de alimentación, con el motor colocado a la salida y sin carga. Para poder realizar un estudio más detallado de las señales de entrada y salida de interés y su respuesta, se utilizó un osciloscopio digital y un generador de señales con el cual se pudo generar señales de diferente tensión, frecuencia y forma de onda.

De estas pruebas se concluyó que la placa funcionaba correctamente y que dos de las señales de salida entregadas por el L297 para el control (señales **ERROR** y **HOME**[14]), no serían utilizadas en nuestra aplicación. A partir de estas modificaciones se realizó un nuevo diseño del esquemático del circuito y consecuentemente de su PCB y se mandaron fabricar las dos placas restantes.

Para verificar el correcto comportamiento de estas placas, se procedió de la misma manera que para la anterior, salvo que en este caso los motores utilizados para las pruebas fueron los que estaban colocados en el brazo. Esto permitió determinar las tensiones de alimentación para cada motor y detectar que dos de los integrados (el puente L298N y el transistor TIP42 utilizado en el control de tensión de alimentación) calentaban mucho, por lo que se les tuvo que colocar disipadores que fueron obtenidos de fuentes de computadora rotas.

La figura siguiente muestra una de las placas de potencia terminada y armada.

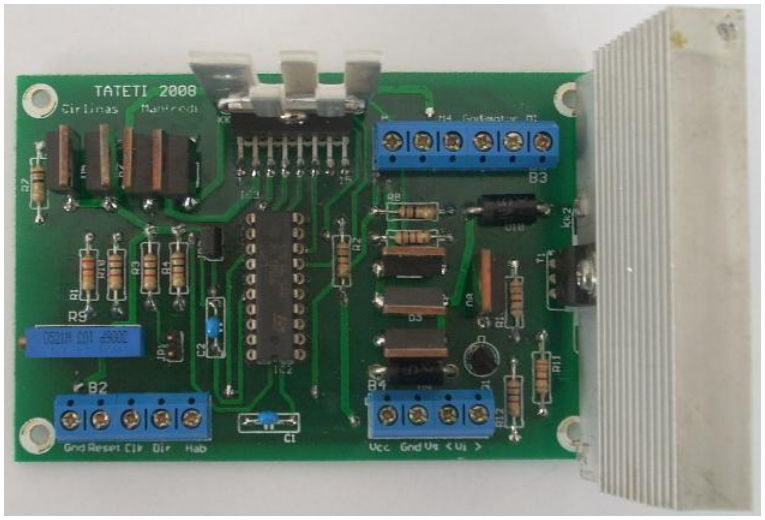


Figura 4.5: Foto de una de las tres placas de potencia para el control y comando de los motores de paso.

La figura 4.6 muestra la ubicación de los componentes en la placa diseñada y los nombres de las señales. El preset se utiliza para regular la entrada de voltaje **VREF** comentada anteriormente. Las entradas de voltaje indicadas como Vs y Vi se corresponden con $V1$ y $V2$ de la figura 4.4 respectivamente. Las borneras M1, M2, M3 y M4 corresponden a las salidas para los motores.

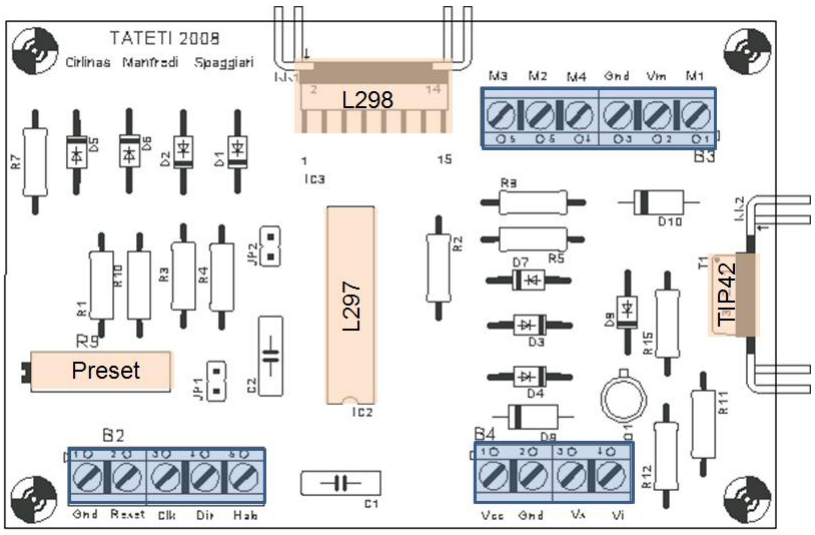


Figura 4.6:

4.3. Placa de potencia para control del motor de continua

4.3.1. Integrado LMD18200

El integrado LMD18200T seleccionado para realizar el control del motor de continua, es un puente H diseñado especialmente para el control de movimiento de motores de continua y de paso. Puede entregar hasta $3A$ de corriente continua, opera a tensiones de alimentación de hasta $55VDC$ y sus entradas son TTL y CMOS compatibles, entre otros. Contiene un circuito innovador que implementa un sensado de corriente a la salida de baja pérdida. Se adquirieron con el modelo de encapsulado MULTIWATT11 (o TO-220). La figura 4.7 muestra su diagrama de bloques funcional.

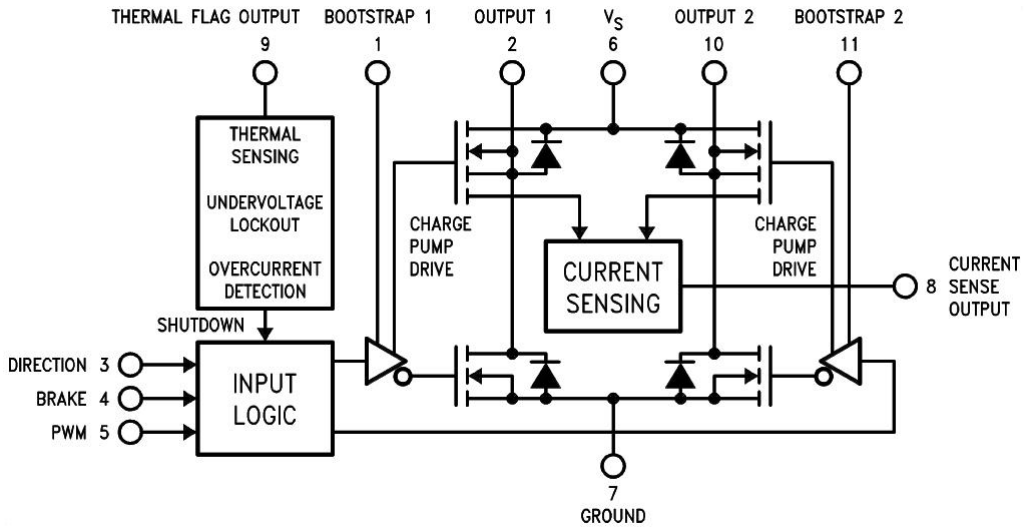


Figura 4.7: Diagrama de bloque funcional del integrado LMD18200T

Como se puede observar, las tres señales de control son **PWM**, **DIRECTION** Y **BRAKE**. Estas tres señales actúan en conjunto para comandar las salidas del puente H.

- **DIRECTION**: Controla la dirección del flujo de corriente entre las salidas OUTPUT1 y OUTPUT2, lo que implica que controla la dirección de giro de un motor.
- **BRAKE**: Se utiliza para frenar al motor y debe trabajar en combinación con **PWM**.
- **PWM**: La tabla de verdad indica cómo se debe utilizar esta señal junto con **BRAKE** y **DIRECTION** para realizar el control deseado. Esta tabla se muestra en la figura 4.8.

PWM	Dir	Brake	Active Output Drivers
H	H	L	Source 1, Sink 2
H	L	L	Sink 1, Source 2
L	X	L	Source 1, Source 2
H	H	H	Source 1, Source 2
H	L	H	Sink 1, Sink 2
L	X	H	NONE

Figura 4.8: Tabla lógica de verdad para el LMD18200.

Debido a la diversidad de controles que se pueden lograr, existen varios circuitos implementados con diferentes tipos de señales PWM. La hoja de datos del integrado LMD18200 posee varios circuitos de aplicación recomendados [5]. Se partió de uno de estos circuitos para el diseño del esquemático, al que se le realizaron las modificaciones necesarias para nuestra aplicación. El circuito recomendado se muestra en la figura 4.9.

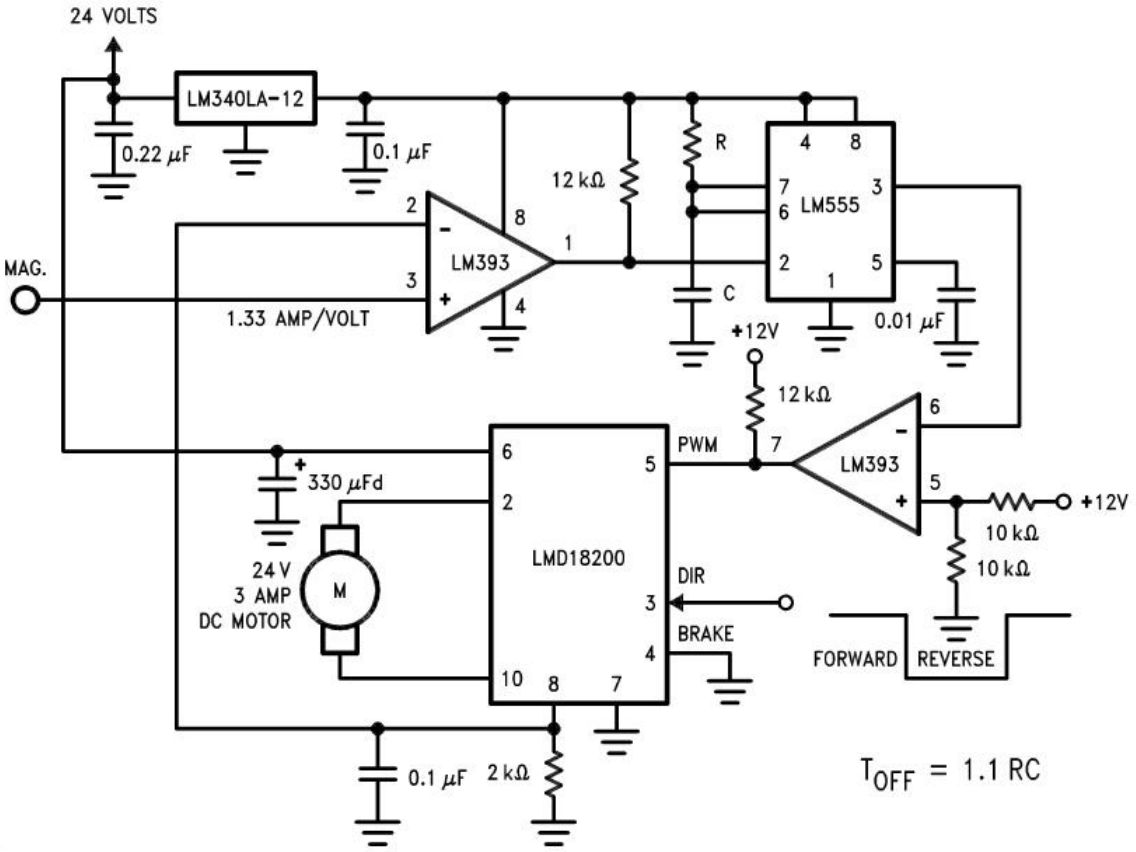


Figura 4.9: Circuito de control recomendado para utilizar con motores de continua.

El circuito anterior controla la corriente por el motor al aplicar un voltaje promedio de $0V$ en los terminales del mismo durante un período de tiempo fijo. Esta acción logra que la corriente por el motor varíe muy poco con respecto al valor promedio controlado

[5] Ver hoja de datos del integrado [16] por diferentes circuitos de aplicación.

externamente. La duración del tiempo en OFF es controlada por la combinación de la resistencia y el condensador a la entrada del timer LM555 [17] siendo $T_{OFF} = 1{,}1RC$. Los comparadores se utilizan para conmutar las señales del circuito y lograr el comportamiento deseado.

La figura 4.10 muestra un ejemplo del comportamiento de este circuito donde se puede observar la modulación realizada sobre la corriente por el motor.

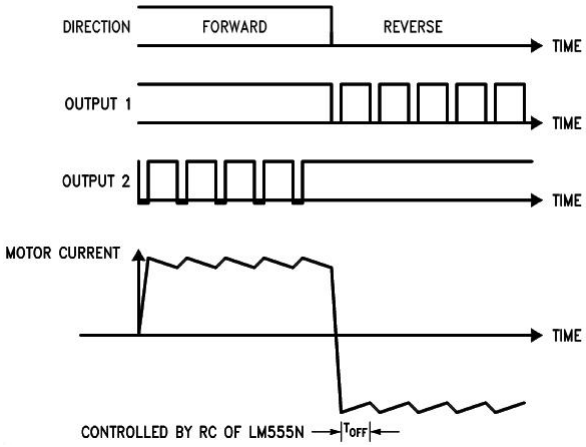


Figura 4.10: Ejemplo de comportamiento del circuito sugerido en la hoja de datos del LMD18200. Cabe observar que el motor se encontrará siempre en movimiento debido a que la señal de **BRAKE** está conectada GND.

4.3.2. Descripción del circuito implementado

El circuito implementado varía con respecto al anterior en los siguientes puntos:

1. Se trabajó todo el circuito con $12V$ para el motor y $5V$ para la señales de control, lo que implicó no colocar el regulador LM340LA-12 a la entrada.

2. La señal analógica **MAG** permite controlar el valor medio de la corriente que puede consumir el motor, haciendo que el mismo se detenga si la corriente supera el máximo. Debido a que el microprocesador no controla señales analógicas y a que no resultó necesario el control de esta señal para el movimiento de la pinza, se colocó un preset multivuelta que se ajustó en la etapa de prueba de la placa.

3. Existe una segunda señal de entrada **INPUT** que se podría controlar externamente[6]. Se dejó prevista en una bornera de entrada para su control, aunque se determinó que para esta aplicación se debe conectar a $5V$ para que el circuito funcione correctamente.

4. La señal de **BRAKE** se pasó a comandar por el microprocesador en lugar de en nivel bajo, permitiendo de esta manera poder frenar al motor.

[6]En la figura 4.9 aparece conectada a $+12V$

Una vez realizadas estas modificaciones, se prosiguió con el diseño del esquemático[7] donde resultaron ser dos las señales necesarias para controlar el motor de continua desde el microprocesador:

1. **DIR**: Señal que indica el sentido de giro del motor.
2. **BRAKE**: Señal utilizada para detener y trabar al motor.

Diseñado y verificado el esquemático se pasó al diseño del PCB. Este diseño se implementó en una sola capa (layer *bottom*) logrando abaratar costos pero debiendo realizar algunos puentes con cables externos[8].

4.3.3. Armado y pruebas de la placa diseñada

Una vez armada la placa, se montaron los componentes y se pasó a la etapa de pruebas. Primero se probó con un motor de continua cualquiera sin carga en su eje, para ajustar la velocidad de giro y torque. Se vio que los valores de algunas resistencias y condensadores sugeridos en la hoja de datos del chip [16] no funcionaban para nuestro motor, debido a que el circuito sugerido refiere a un motor de $24V$ de alimentación. Mediante un proceso de prueba y error se probaron diferentes valores hasta dar con los valores correctos. En particular se ajustaron los valores del condensador y resistencia que determinan el tiempo de T_{OFF}.

Las figuras 4.11 y 4.12 muestran una foto de la placa terminada, y un esquema de la misma respectivamente. En esta última se indican los componentes y los nombres de las señales en las borneras.

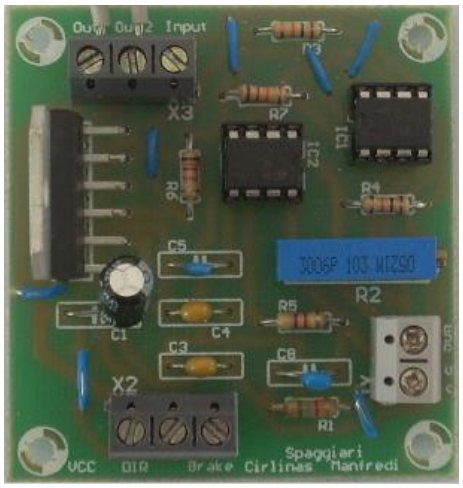


Figura 4.11: Placa de control del motor de continua.

[7] Ver figura C.5 del apéndice C por detalles del esquemático diseñado
[8] Por detalles del esquemático y PCB diseñado, ver figuras C.5 y C.6 del apéndice C.

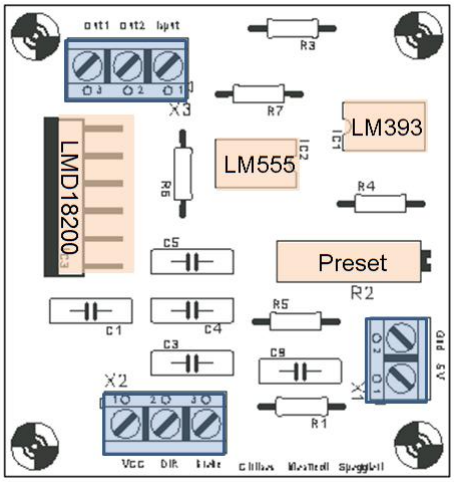


Figura 4.12: Esquema de la placa de control del motor de continua.

Como ya fue mencionado, en esta etapa de pruebas se ajustó el valor de **MAG** para que el motor funcionara correctamente. En definitiva, como **MAG** controla el valor medio de la intensidad que puede circular por el motor, está controlando el torque máximo que puede realizar. Frente al primer ajuste todo quedó funcionando correctamente hasta que se le colocó una carga en el eje. Se estuvo bastante tiempo para estudiar en detalle el comportamiento del circuito. Luego de varias pruebas se llegó al valor de **MAG** y a la forma correcta de comandar las señales para obtener el comportamiento deseado.

Al controlar el motor de continua colocado en la mano se vio, como era de esperar, que los movimientos de apertura y cierre de la pinza eran distintos; cuando las pinzas miran hacia abajo (esto sucede cuando se quiere tomar una ficha), se requiere de mayor torque para abrir la mano que para cerrarla. Si bien el valor de **MAG** seleccionado resulta adecuado para abrir la mano en esa posición, hace que las pinzas se cierren muy bruscamente. Como este valor no podía ser comandado desde el microprocesador, se decidió enlentecer el cierre de la pinza comandando la señal de **BRAKE** desde el microprocesador, tal como se explicará más adelante7.

4.4. Realimentación de posición: Encoders ópticos

En un principio, no se pensaba utilizar una realimentación para el control de la posición de los motores en ninguna de las articulaciones del brazo. Luego de analizar el comportamiento de los sistemas de estas características y apelando a los conocimientos de Control adquiridos, se vio la necesidad de implementar una realimentación para no perder el control de la posición del brazo por trabajar en loop abierto. Utilizando en cada articulación un disco graduado cada cierta cantidad de grados, se puede ir contando pulsos en la señal del encoder y determinar en base a ello la posición del brazo.

Cuando se desarmaron impresoras para obtener motores para el brazo, también se retiraron varias placas con encoders que las propias impresoras utilizaban para controlar la posición de sus motores. Después de realizar algunas pruebas para estudiar su comportamiento, se determinó que estos encoders, alimentados con $5V$ y acompañados de un par de resistencias, servían para nuestra aplicación.

El circuito implementado para los encoders se muestra en la figura siguiente.

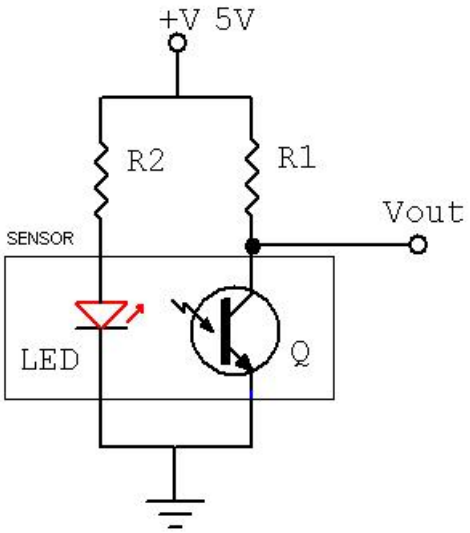


Figura 4.13: Circuito eléctrico implementados para los sensores.

Como se puede observar, el sensor se compone de un LED infrarrojo y un fototransistor. Tiene una ranura en el centro para que se pueda cortar el haz infrarrojo con algún elemento, que en nuestro caso, será el disco transparente graduado.

Cuando hay algún objeto opaco en la ranura, el fototransistor no se excita y por lo tanto no conduce, determinando que $Vout = VCC$.

Cuando no hay ningún objeto cortando la trayectoria del haz, el fototransistor toma corriente de su colector, determinando que el voltaje de salida sea $Vout = VCC - R1I$, donde I es proporcional a la intensidad del haz y por lo tanto dependiente de $R2$. Si se seleccionan los valores de $R1$ y $R2$ correctos, se puede lograr que este voltaje sea interpretado como un cero lógico.

El bloqueo del haz infrarrojo no fue tan trivial. El material debe ser totalmente opaco

para que funcione correctamente. Inicialmente se armaron discos graduados cada 2 grados impresos con impresora láser sobre una lámina de plástico para transparencias. Estos discos se pegaron a discos de CD transparentes para darle rigidez, lo cual dio bastante trabajo porque no cualquier pegamento servía.

Tampoco resultó problema trivial el dimensionado de las resistencias del circuito. Al probarlo con los discos mencionados se detectaron espurios en la señal enviada por encoder lo que provocaba, que cada cierto tiempo, el microprocesador detectara una señal inválida. Se testeó este circuito con un osciloscopio digital y se determinó que estos espurios eran causados por la combinación de una corriente de polarización del LED muy alta con una impresión de toner no lo suficientemente opaca como para evitar el pasaje de la luz.

Ajustando los valores de las resistencias para disminuir la potencia en el LED, se logró obtener un comportamiento uniforme sin espurios considerables.

Una vez encontrados los valores de $R1$ y $R2$ óptimos para la aplicación, se armaron cuatro placas de diferente forma y tamaño determinando previamente su ubicación en el brazo. Se probaron una por una con diferentes objetos para estudiar su funcionamiento.

La figura 4.14 muestra una foto del osciloscopio donde se pueden observar las señales de dirección **DIR** en la parte superior y el voltaje de salida V_{OUT} del encoder en la parte inferior.

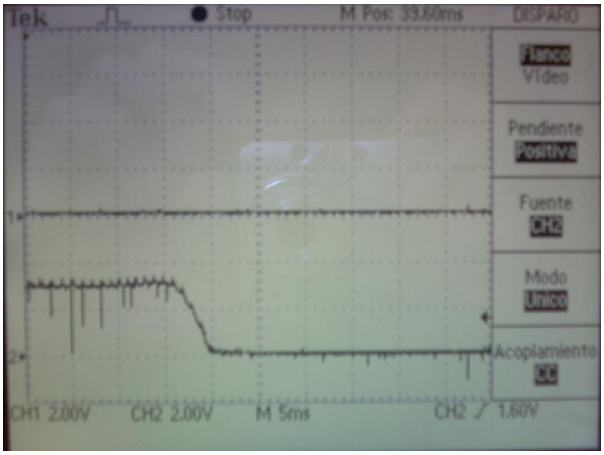


Figura 4.14: Observación de espurios en la salida del encoder.

Tal como fue mencionado, los discos de la articulación del codo y hombro se graduaron cada dos grados, mientras que el de la muñeca se graduó únicamente para indicar los dos finales de carrera y la posición intermedia y el de la mano se graduó para marcar los dos finales de carrera.

Las figuras siguientes muestran un acercamiento a los encoders ubicados en cada una de las articulaciones con sus respectivos discos.

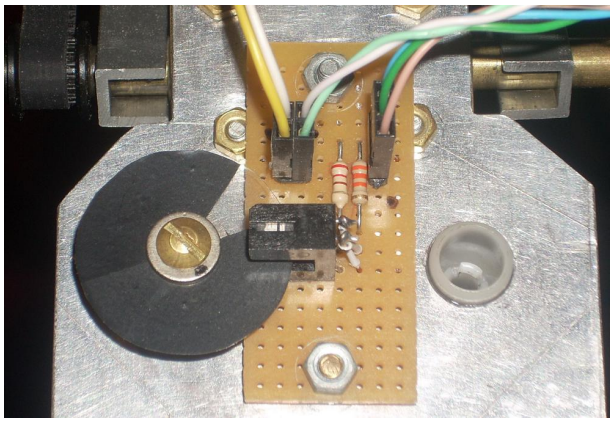


Figura 4.15: Encoder ubicado en la mano utilizado para controlar la posición de apertura de las pinzas.

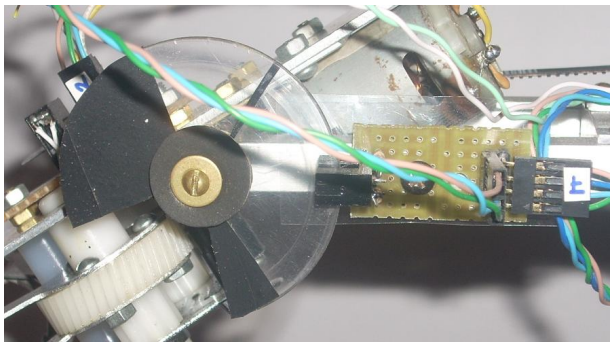


Figura 4.16: Encoder ubicado en la muñeca utilizado para controlar la posición de la mano.

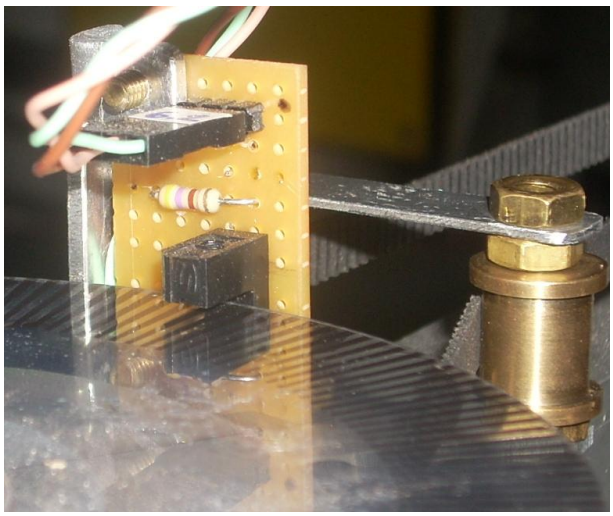


Figura 4.17: Encoder ubicado en el codo utilizado para controlar la posición del antebrazo con respecto al brazo y a la posición de *Home*.

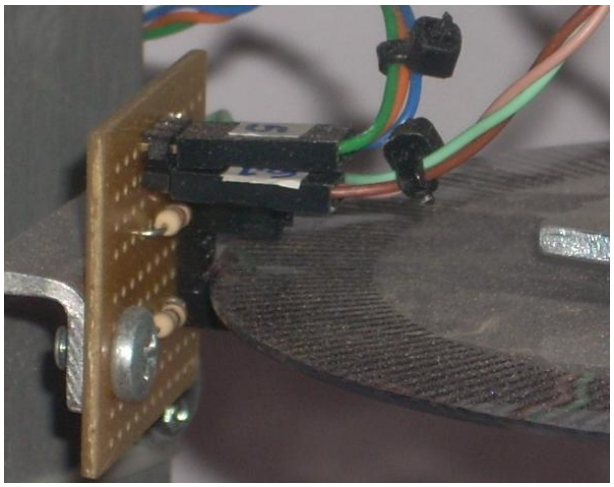


Figura 4.18: Encoder ubicado en el hombro utilizado para controlar la posición del brazo con respecto a la posición de *Home* para esta articulación.

Capítulo 5

Diseño e implementación de la Interfaz de Usuario

5.1. Introducción

El presente capítulo describe el diseño y construcción de la interfaz de usuario, que como ya fue mencionado, permitirá al usuario interactuar con el juego utilizando botones y carteles luminosos indicadores que lo guiarán a lo largo de la partida. A través de la misma, el usuario podrá seleccionar el modo de juego y realizar sus jugadas.

La misma está compuesta por dos partes bien diferenciadas. Por un lado se encuentra el mueble de la interfaz (o tablero, donde están montados las luces y los botones) y por el otro el hardware de control (encargado de controlar la eléctrica del sistema de luces y botones). En las siguientes subsecciones se detallará el diseño y construcción de ambas partes.

5.2. Diseño y construcción del mueble de interfaz

En la primera instancia del diseño se definió completamente la apariencia del tablero en el cual irían montados los botones y luces. Se evaluaron varias propuestas de interfaz en las cuales variaba tanto el número de botones y luces como el tipo de interacción del usuario con el juego. Se trató de minimizar el número de botones para la selección del modo de juego y así simplificar la lógica de control, volviéndolo comparable a los clásicos juegos de maquinitas que los niños y adolescentes conocen. Cabe aclarar que el diseño y armado del mueble fue una tarea en conjunto con el personal de Ciencia Viva.

Luego, se pulió esta propuesta inicial con el cliente, reconociendo sus necesidades y preferencias, y se presentó el diseño final en AutoCAD mostrado en la figura 5.1. La construcción del tablero se basó en este diseño.

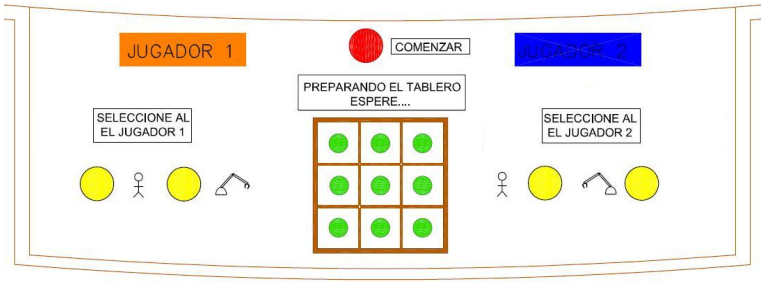


Figura 5.1: Primer diseño en AutoCAD de la interfaz de usuario.

Se decidió tener un botón **COMENZAR**, con el cual se inicia una partida, dos botones de selección de modo de juego por jugador y los nueve botones que indican al brazo en qué casillero del tablero de juego se debe colocar la ficha de la jugada.

El tablero también contiene tres carteles luminosos; el cartel de *ESPERE...* para indicar que se debe esperar dado que el brazo está realizando movimientos, y los carteles *JUGADOR 1* y *JUGADOR 2* para indicar el turno de cada jugador y el resultado de la partida.

Definida la cantidad de botones y luces a ser comandados por el microprocesador, se seleccionó y compró los materiales a colocar en el mueble y los utilizados para el hardware de control. Es importante aclarar que la definición del número de entradas y salidas al microprocesador no fue tarea trivial, dado que fue necesario conocer, desde el inicio de la etapa de diseño, la totalidad de las señales del sistema para asegurar la disponibilidad de puertos de entrada-salida necesarios para la su implementación.

Se decidió utilizar pulsadores industriales para la implementación de los botones de interfaz de usuario, debido a que los mismos no sólo son fáciles de encontrar en plaza sino que son bastante robustos frente a malos tratos.

Los pulsadores seleccionados para el botón **COMENZAR** y para la selección del modo de juego son de tipo hongo iluminables de diámetro $\phi 40mm$ y base de diámetro $\phi 22mm$ de colores amarillo y rojo. Los nueve pulsadores para seleccionar la posición en el tablero son del tipo rasantes de diámetro $\phi 22mm$ de aro metálico y color verde. Cada uno de ellos se compró con un contacto normal abierto (NO) implementando así un *pull-down* a la entrada de la placa de interfaz. Estos pulsadores, a pesar de su elevado costo, son de muy buena calidad. Cabe aclarar que no fue fácil encontrar pulsadores de hongo iluminables de estos tamaños en el mercado local[1].

Las luces que iluminan los tres carteles y los casilleros de los nueve botones son un conjunto de LEDs blancos de alto brillo, mientras que las luces de los cuatro pulsadores de selección de modo de juego son lamparitas de hasta $30VDC$ que se colocan en el pulsador como si fueran otro contacto.

El modelo, marca y código de los materiales adquiridos se detallan el la tabla 5.1.

[1]La empresa de eléctrica e iluminación BERON dispone de este tipo de pulsadores de marca LOVATO.

Material	Marca	Modelo	cantidad
Pulsador tipo hongo iluminable color rojo	LOVATO	8LM2T BL6144	1
Pulsador tipo hongo iluminable color amarillo	LOVATO	8LM2T BL6145	4
Pulsador rasante de aro metálico color verde	ANU	LA139M	9
Porta lámpara para pulsador iluminable	LOVATO	8LM2TEL400	5
Leds blancos de alto brillo	ENEKA	-	60

Tabla 5.1: Materiales eléctricos que componen la interfaz de usuario.

En la figura 5.2 se pueden observar los tres tipos de pulsadores adquiridos y sus contactos.

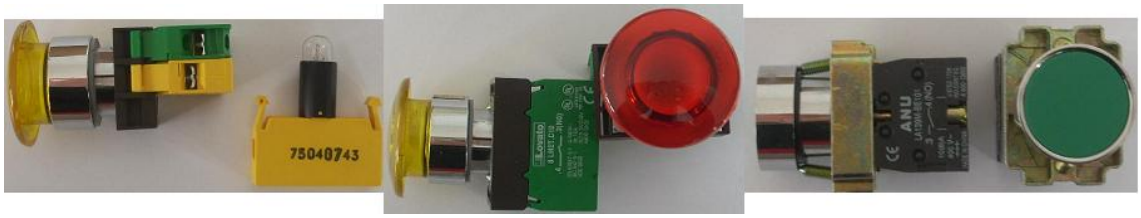


Figura 5.2: Pulsadores que componer la interfaz de usuario.

Como se mencionó, los casilleros del tablerito también deben poder iluminarse desde adentro según lo acordado en la etapa de diseño. Para ello se diseñaron placas de electrónica de tamaño tal que cubran el casillero ($4,5cm \times 4,5cm$) los cuales poseen cuatro LEDs blancos de alto brillo conectados en serie. El objetivo de cada una de estas placas es iluminar desde adentro los acrílicos de color blanco que se colocaron alrededor de los pulsadores verdes. El lado interior de este tablerito está formado por una grilla de madera con tapa sobre la cual se montaron las placas de LEDs, donde su objetivo es aislar cada casillero de la luz proveniente de las placas vecinas.

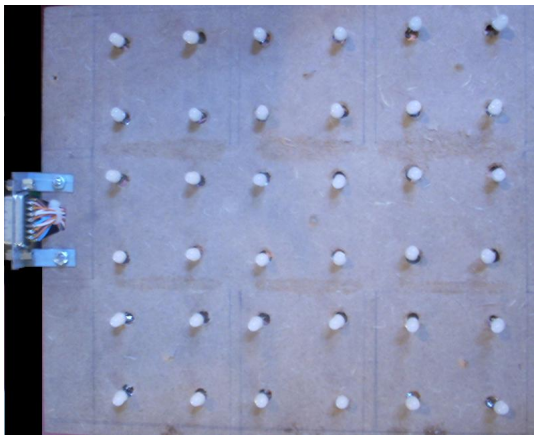


Figura 5.3: Tapa de la grilla con las placas de LEDs.

Para los carteles, se armaron tres placas de mayor tamaño conteniendo ocho LEDs blancos de alto brillo conectados en serie. Las placas están colocadas dentro de cajas de madera que cubren por completo a cada uno de los carteles.

En la figura 5.4 se muestra el tablero de juego en su versión final, pintado y con los carteles colocados, donde se observan claramente los acrílicos de color blanco mencionados anteriormente.

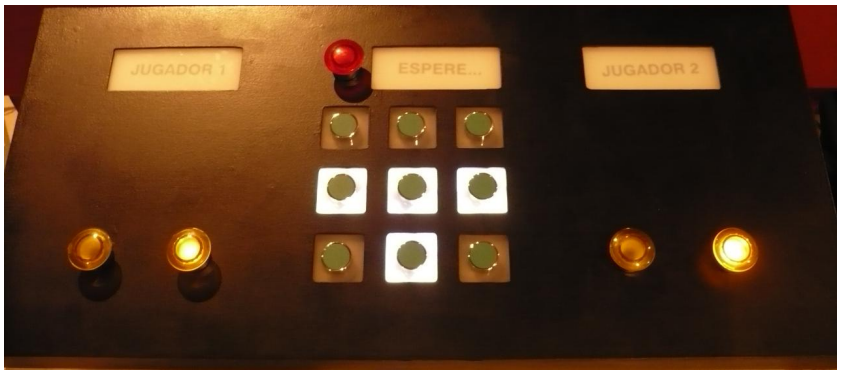


Figura 5.4: Vista frontal del tablero de interfaz de usuario.

Las figuras 5.5 y 5.6 muestran el interior del mueble de interfaz de usuario donde se pueden visualizar las cajas y grilla mencionadas. Se observan las placas de LEDs colocadas hacia el interior de cada casillero y cartel con sus respectivos conectores.

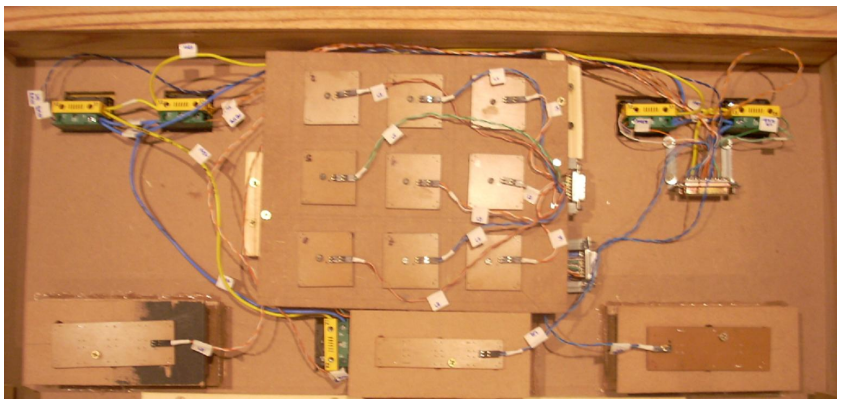


Figura 5.5: Vista interior de la interfaz de usuario.

La interfaz de usuario está armada de forma tal que cada una de sus partes se puede desconectar y retirar sin comprometer al resto.

Para interconectar las señales provenientes de la placa de control de interfaz (que se detalla en la siguiente subsección) con los componentes de la interfaz de usuario (botones, placas de LEDs y luces), se armaron varios cables con conectores $DB25$ y $DB15$ para facilitar su conexión y desconexión. Se encuentran separados en tres grupos: las luces de los nueve botones del tablerito se conectan en uno de los conectores hembra, los botones se conectan en el otro DB15 hembra y el resto de luces y botones del tablero se conectan en el conector hembra DB25. En las figuras 5.7, 5.8 y 5.9 se muestran los mismos al igual que los conectores hembra ubicados en la parte interior del tablero.

Por detalles sobre la conexión de las señales, referirse al Manual de Mantenimiento en el apéndice D.

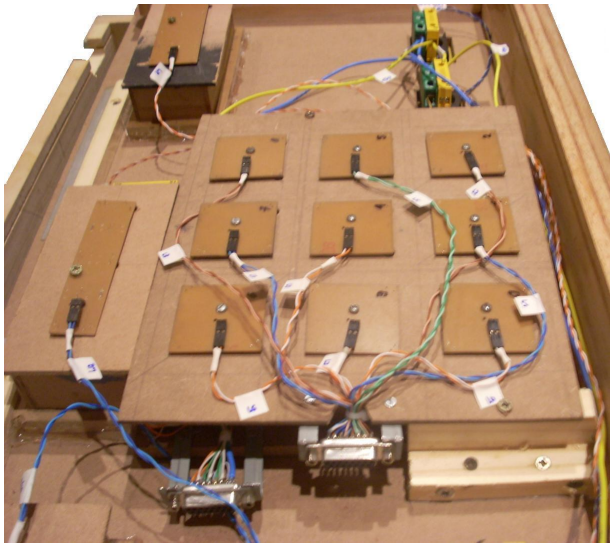


Figura 5.6: Vista interior de la interfaz de usuario.

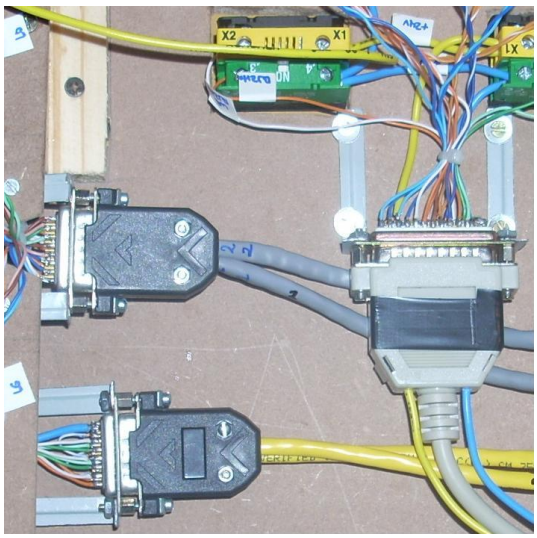


Figura 5.7: Conectores DB15 y DB25 utilizados para conectar la interfaz de usuario con la placa de interfaz.

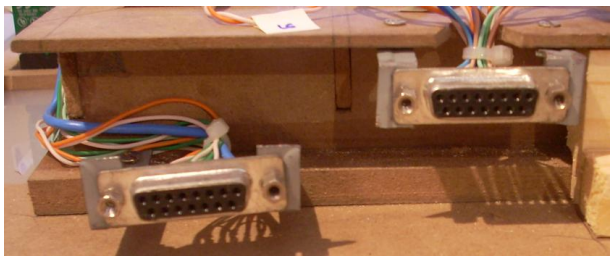


Figura 5.8: Conectores DB15 hembra utilizados para las señales de luces y botones del tablerito.

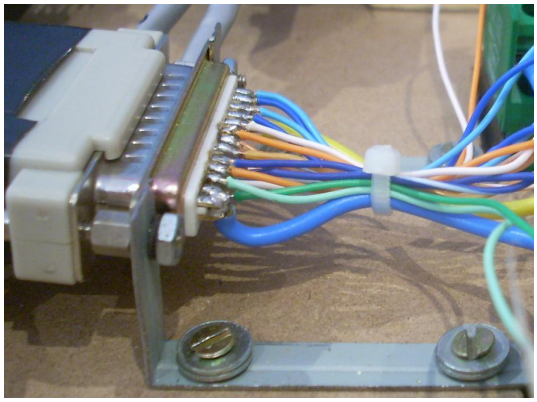


Figura 5.9: Detalle del conector DB25.

5.3. Diseño e implementación del hardware de control

Para que el microprocesador reciba señales de los botones desde el tablero de interfaz y envíe señales de control a las luces sin problemas, es necesario protegerlo frente a cortocircuitos y sobrecorrientes con buffers de entrada y salida. Para ello se decidió utilizar buffers 74LS541[25] (de ocho entradas) que permiten la interacción adecuada del microprocesador Rabbit 3000 con señales de $5VDC$. Como el microprocesador trabaja con una tensión de alimentación de $3,3VDC$ una señal de salida en nivel alto tendrá a lo sumo este valor de tensión. Por ello,es imprescindible verificar que el voltaje en el cual conmutan sea menor a $3,3VDC$. Como se puede observar en la hoja de datos, el buffer mencionado detecta un nivel alto si su entrada supera los $2VDC$.

A su vez, para poder encender las luces desde el microprocesador, se necesita una etapa de potencia de salida que entregue la corriente necesaria para su correcto funcionamiento. Conocido el funcionamiento de los integrados ULN2803 [26], se decidió utilizarlos para armar esta etapa alimentándolo con $24VDC$. El integrado contiene ocho etapas Darlington formadas con transistores NPN las cuales se comandan con señales lógicas TTL, CMOS o PMOS y pueden entregar hasta $500mA$ por salida.

La interfaz se compone de catorce botones, doce placas de LEDs y cuatro lámparas. Para comandarlos se utilizan cuatro buffers y dieciséis etapas Darlington (dos integrados ULN2803) para las luces.

Frente a esta necesidad se diseñó y armó una placa de control de interfaz que contiene los componentes mencionados exceptuando los buffers de salida necesarios para proteger al microprocesador Rabbit en el comando de las luces. Por simplicidad en el armado de esta placa, se decidió colocarlos directamente en la placa donde se encuentra el módulo RCM3365 del microprocesador. Las salidas de estos buffers se conectarán luego a las entradas de los dos integrados ULN2803 mencionados.

Todos los componentes de iluminación del tablero se alimentan con $24VDC$ mientras que los botones manejan contactos entre $0VDC$ y $5VDC$. Se implementó un *pull-down* para la entrada de cada pulsador, lo que implica que se recibirá un cero lógico si el botón

no se encuentra pulsado. Para el manejo de las luces, mediante la señal digital que recibe el integrado ULN2803 se cierra o abre el circuito encendiendo o apagando las mismas. Para una mejor comprensión del funcionamiento de estos circuitos, en las figuras 5.10 y 5.11 se representa su esquema de funcionamiento.

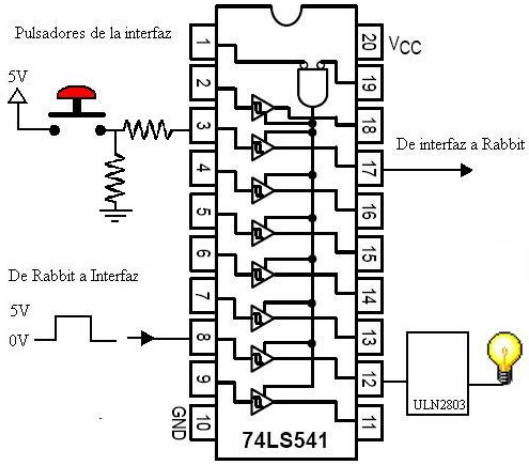


Figura 5.10: Diagrama de conexión del buffer 74LS541.

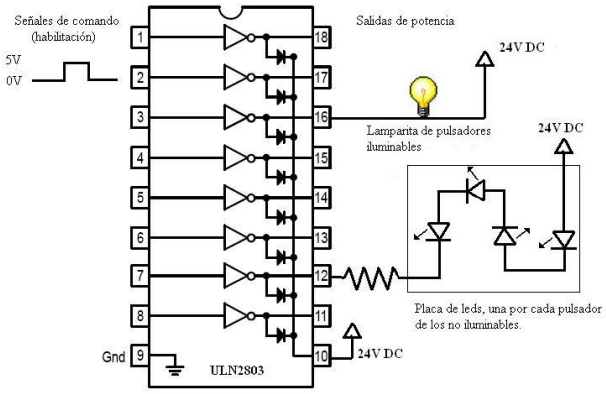


Figura 5.11: Diagrama de conexión al integrado ULN2803.

Los LEDs utilizados tienen un voltaje V γ del orden de los $2V$. Al colocar varios LEDs en serie se debe mantener en un valor adecuado la corriente que va a circular por el circuito. Para ello se colocó una resistencia en serie con la cual se ajusta la corriente. Se tomó como corriente de partida $10mA$. El cálculo de la misma se detalla a continuación donde n es el número de LEDs conectados en serie. En la mayoría de los casos, la potencia determinada en los cálculos excedía los $0,25W$ por lo que fue necesario colocar resistencias de $2W$.

$$R_{salida} = \frac{24 - 2{,}0 \times n}{10mA} \tag{5.1}$$

$$P_R = \frac{((24 - 2,0 \times n)^2}{R_{salida}} \tag{5.2}$$

Para limitar la corriente de entrada al microprocesador al presionar un pulsador, se colocó una resistencia en serie con el botón y otra oficiando de *pull-down*, tal como se observa en el esquema 5.10.

Una vez realizado el diseño de la placa en papel se procedió a realizar algunas pruebas de estos circuitos para corroborar su correcto funcionamiento. Los resultados obtenidos de las pruebas fueron satisfactorios por lo que se continuó con el diseño del PCB. Este diseño se realizó en una sola capa y se le colocaron borneras para cada una de las entradas y salidas con el objetivo de facilitar su conexión al resto del sistema.

La siguiente figura muestra la placa de interfaz terminada.

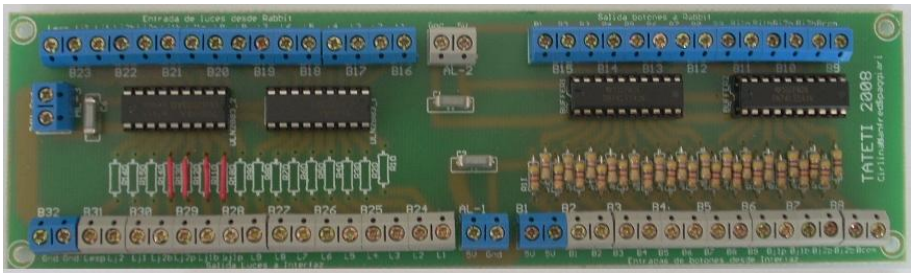


Figura 5.12: Placa de control de interfaz terminada. Faltan algunas de las resistencias de salida debido a que su valor se determinó junto con la definición de las luces a colocar.

En el esquema mostrado en la figura 5.13 se indican los componentes de la placa al igual que la señal que se conecta a cada una de las borneras.

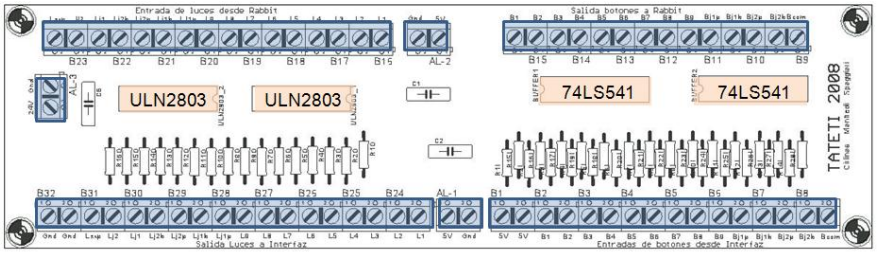


Figura 5.13: Esquema de la placa de control de interfaz.

Armada la placa, se probó con todas las placas de LEDs en serie y con los pulsadores verificando que el comando de los mismos con señales provenientes del microprocesador fuera el esperado. Estas pruebas resultaron satisfactorias dando así por aprobada la placa de control de interfaz.

Capítulo 6

Control del Sistema

6.1. Descripción del Rabbit Core RCM 3365

Como se explica en el capítulo 2, se analizó y luego decidió utilizar un microprocesador Rabbit para la implementación del control del sistema. Esto se debió, no solo a que se disponía de varios ejemplares sin costo sino a que luego de estudiar las características y ventajas, se vio que su inserción en un sistema embebido de estas características resultaría sencilla.

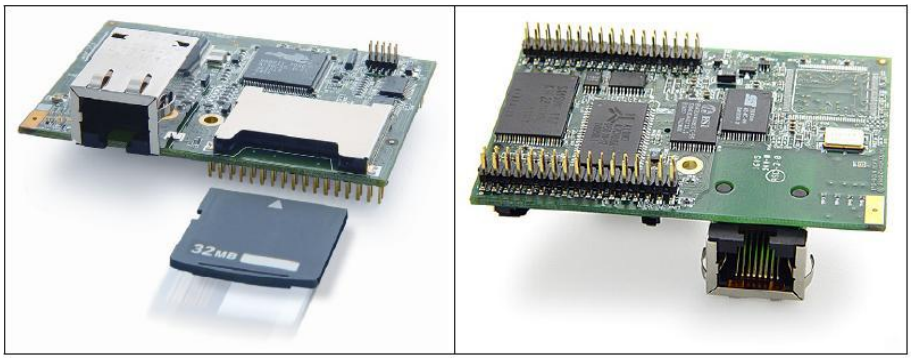


Figura 6.1: Foto del módulo de Rabbit RCM3365 con el microprocesador Rabbit 3000

El módulo Rabbit RCM3365 utilizado en este sistema está compuesto por un microprocesador Rabbit 3000 y todos lo elementos necesarios para su funcionamiento, permitiendo utilizarse directamente en cualquier sistema casi sin ningún hardware adicional. Posee dos relojes (el oscilador principal y un reloj en tiempo real) y la circuitería necesaria para el respaldo por batería de la RAM estática y reloj. Se coloca en el *motherboard* del diseño realizado por un usuario y actúa como el microprocesador del sistema. Puede interactuar con todo tipo de dispositivos digitales CMOS-compatible.

El lenguaje utilizado para programarlo es el Dynamic C, una variación del ANSI C. Los programas se desarrollan dentro del entorno de programación Dynamic C versión 9.24 o superior. Este entorno es muy amigable y permite editar, compilar, depurar, monitorear en tiempo real y descargar los programas al microprocesador a través de un cable de programación[13].

Características principales del RCM3365

- Tamaño: (47 mm x 69 mm x 22 mm).
- Microprocesador Rabbit 3000 corriendo a 44,2MHz.
- 16MB Nand Flash.
- 512K Memoria Flash / 512K SRAM.
- Puerto RJ-45 para Ethernet.
- 52 I/O digitales paralelas: 44 configurables como I/O, 4 entradas fijas y 4 salidas fijas. Posibilidad alternada de configurar algunos puertos con bus de datos y direcciones como señales de control.
- 10 timers de 8 bit y un timer de 10 bits.
- Reset externo.
- Reloj en tiempo real.
- Dos headers de 34 pines para acceder a las señales y puertos.
- 3, 3VDC de alimentación.
- 6 puertos seriales configurables.
- 5VDC tolerante para I/O.

Especificaciones más detalladas del módulo se pueden encontrar en el apéndice A del Manual de Usuario [18], *RCM3365/RCM3375 Specifications*.

La figura 6.2 muestra los subsistemas en los que se basa el Rabbit, diseñados en el módulo RCM3365.

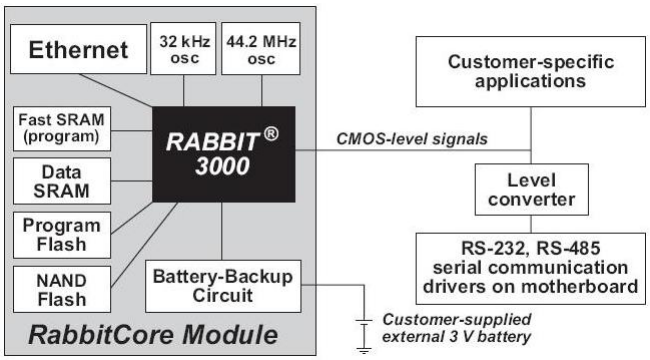


Figura 6.2: Subsistemas del módulo de Rabbit

6.2. Diseño e implementación del hardware para el control del juego

Rabbit TM dispone de varios kits de desarrollo los cuales incluyen placas de prototipado que permiten simular y realizar una gran variedad de pruebas y diseños. A su vez, estas placas facilitan la conexión de los módulos RCM a una tensión de alimentación, poseen algunos periféricos básicos de entrada-salida al igual que un área libre de prototipado para implementar otros diseños de hardware.

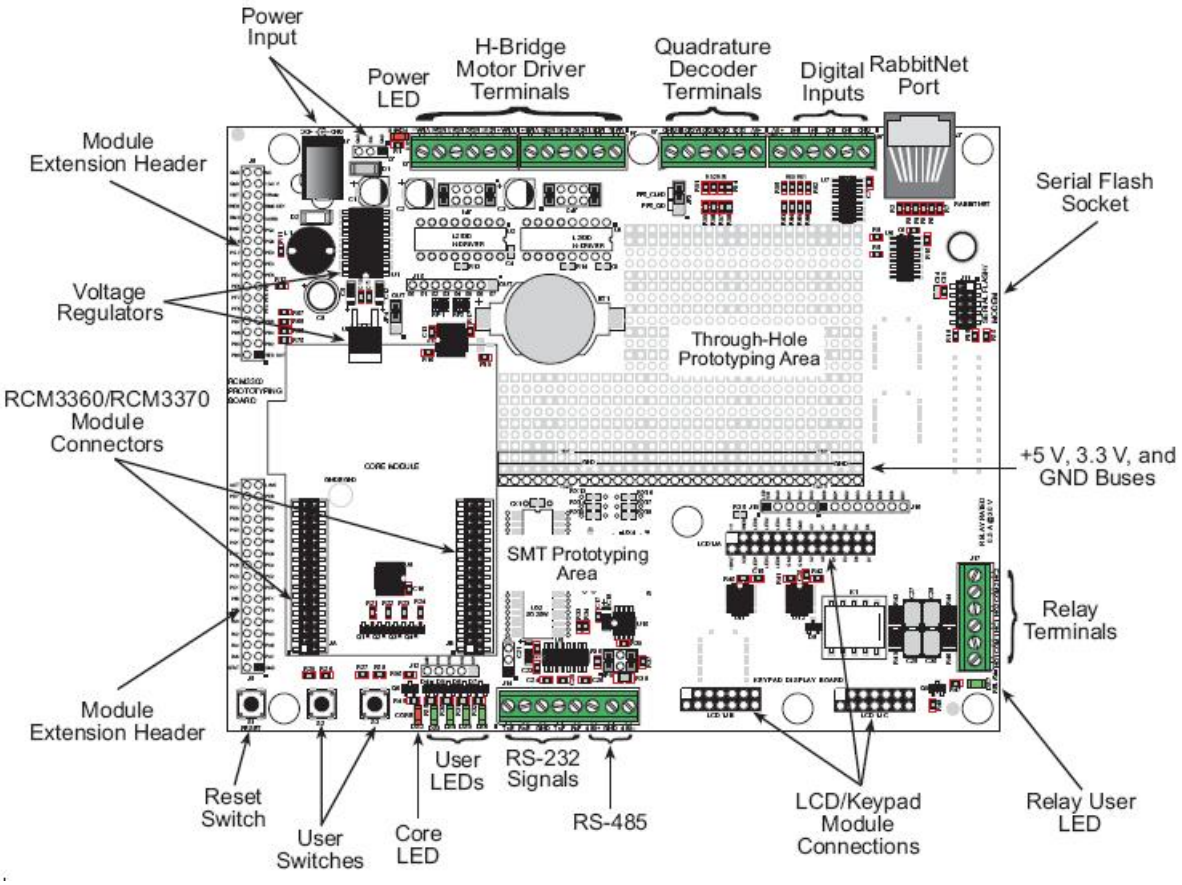


Figura 6.3: Esquema de la placa de prototipado para el RCM3365.

Sin embargo, esta placa no está pensada para formar parte de un sistema con funcionalidades particulares como lo es este proyecto. Por lo tanto, a pesar de que se han podido realizar una gran cantidad de pruebas, fue necesario diseñar una placa independiente donde colocar al módulo de Rabbit y donde poder colocar los periféricos de entrada-salida necesarios para este sistema.

Previo al diseño de esta placa, se leyó con especial cuidado el manual de usuario del RabbitCore RCM3365 para asegurarse de entender su funcionamiento, protecciones y dimensiones[1].

[1]Por detalles del módulo referirse al apéndice A del Manual de Usuario[18].

La primera dificultad encontrada refiere a los zócalos para los headers necesarios para poder conectar el módulo a una placa externa. Los mismos son de montaje superficial y sus dimensiones no son las estándar de 2,54mm de separación entre conectores sino que son de 2,00mm.

Esto en un principio resultó ser un problema para el diseño del PCB debido a que los programas de diseño utilizados no disponen de footprints de este tipo. Se crearon entonces los diseños en el Eagle[19] para estos conectores de montaje superficial respetando las distancias especificadas en el manual como muestra la figura 6.4. A partir de esta especificación se creó el layout para el módulo con headers para montaje superficial.

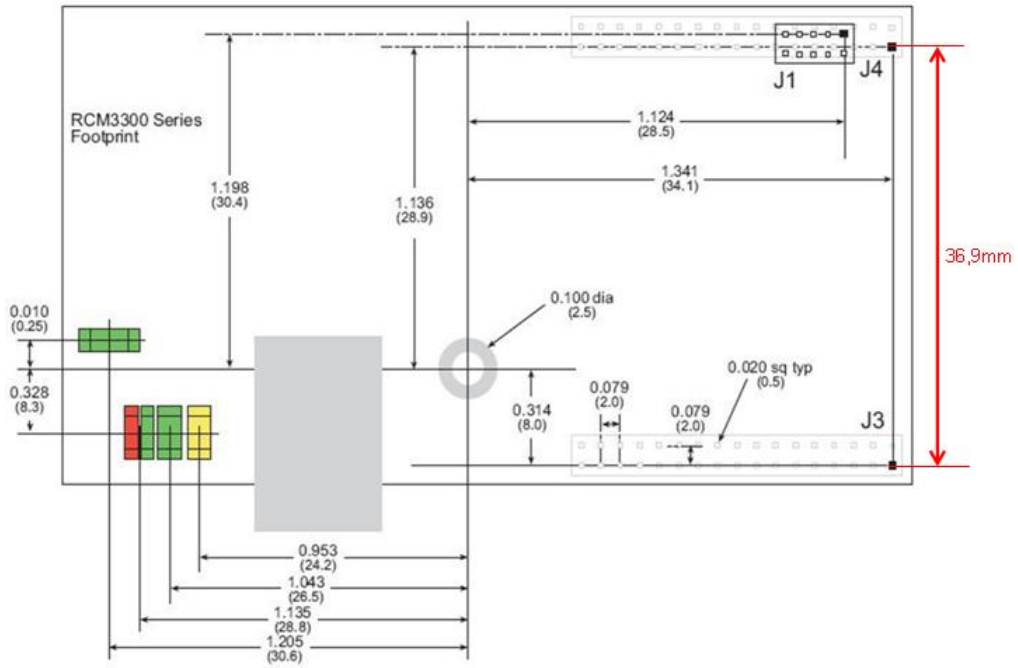


Figura 6.4: Footprint

A pesar de que estos zócalos no se venden en plaza, lo que resultaba en otro inconveniente, se disponía de los mismos gracias a una donación.

Diseñados los footprints para el microprocesador, se continuó con el diseño del resto de la placa. Se agregaron buffers de entrada-salida para todos los puertos paralelos utilizados para interactuar con el sistema. Previo a este diseño se debieron determinar cuáles puertos trabajarían como entradas y cuáles como salida para conectarlos en el sentido correcto a los buffers. Para ello fue necesario definir el número de señales a comandar.

El módulo RCM3365 dispone de puertos paralelos de 8 bits nombrados de la A a la G. Los mismos presentan algunas restricciones para su utilización en cualquier sistema que son propias del correcto funcionamiento del microprocesador. Se tuvo que evaluar en detalle el comportamiento de cada uno de los puertos, su posible utilización y la existencia de los mismos en los 72 headers accesibles. La figura 6.5 muestra un esquema del uso de

los puertos del microprocesador Rabbit 3000 en el módulo RCM3365.

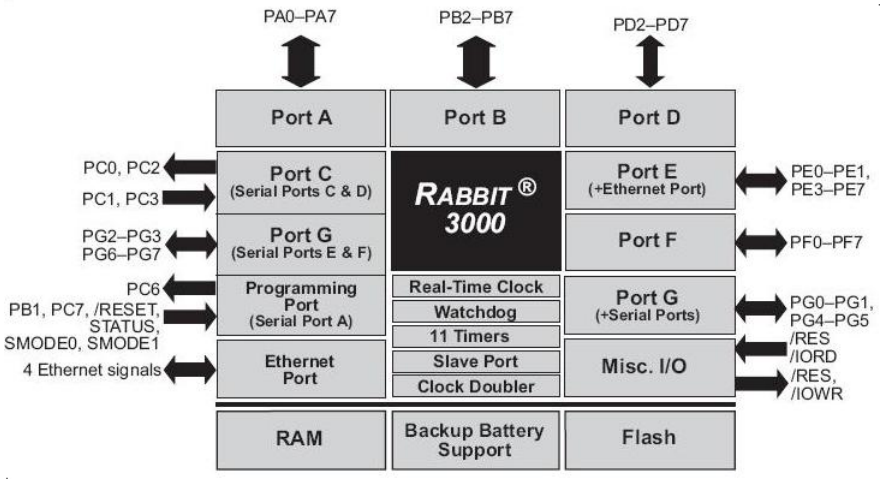


Figura 6.5: Puertos del módulo de Rabbit.

A su vez, la figura 6.6 muestra el pinout de los headers J3 y J4 de la placa de prototipado que se corresponden con los headers de 34 pines ubicados en el módulo. Como se puede observar, los mismos poseen la mayoría de los puertos mencionados al igual que señales de control, puertos de alimentación y señales de entrada-salida. Se partió de este esquema para el diseño y conexionado de señales en el PCB.

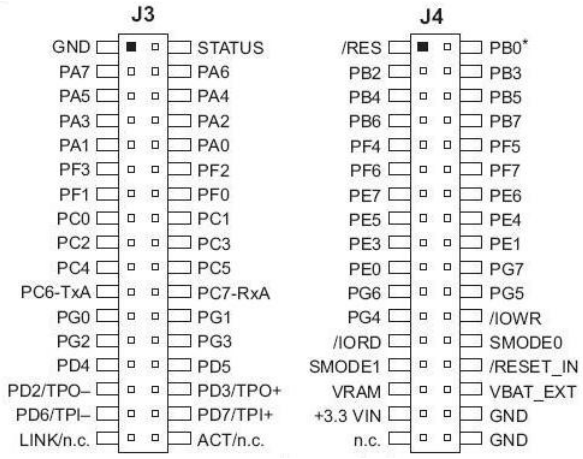


Figura 6.6: Pinout de los headers del módulo RCM3365.

A continuación se enumeran algunos de los problemas encontrados para la configuración de los puertos que se tuvieron que tener en cuenta para el diseño[2]. También se muestra un cuadro con la configuración realizada para el sistema.

1. Los puertos PC6 y PC7 se corresponden con el puerto serial A y se utilizan para realizar la programación del microprocesador. Por este motivo, si estos puertos a su

[2]La configuración de los puertos por software se detalla en el capítulo 7

vez se utilizan como entrada-salida de señales del sistema, no es posible realizar un depurado del programa en tiempo real.

2. Si el puerto A se utiliza como puerto de entrada, el puerto F solo puede ser puerto de salida. Esto se debe a un conflicto propio del microprocesador.

3. Los puertos PD0 y PD1 se utilizan para llevar las señales de *LINK* y *ACT* y encender unos leds en el módulo. Se debe retirar un resistencia de la placa para desconectarlos de estas señales.

4. El puerto PB1 no se encuentra accesible en los headers mencionados.

La tabla siguiente resume la configuración realizada para los puertos paralelos del microprocesador.

	Bit del puerto							
Puerto	7	6	5	4	3	2	1	0
A	I	I	I	I	I	I	I	I
B	O	I	O	O	O	O	X	O
C	I	O	I	O	I	O	I	O
D	O	O	O	O	O	O	X	X
E	O	O	O	X	I	X	O	O
F	O	O	O	O	O	O	O	O
G	O	O	I	I	I	I	I	I

Tabla 6.1: Configuración de los puertos paralelos de entrada y salida

Se colocó una batería de litio de $3V$ modelo CR2032 tipo moneda recomendada en el Manual de Usuario, para respaldar la memoria RAM estática (el estado de las variables) frente a una caída de la alimentación del microprocesador. La misma se conecta al pin indicado como VBAT_EXT en la figura 6.6. A su vez, se agregó un pulsador de RESET disponible en el pin indicado como RESET_IN en la figura 6.6, que permite resetear el programa. También se tienen disponibles como entradas dos señales de status, SMODE0 y SMODE1. El estado lógico de las mismas determina el procedimiento de arranque del sistema luego de un reset. Se previó la posibilidad de seleccionar la combinación del estado colocando jumpers externos. De todas maneras, se realizó un *pull-down* de las mismas debido a que son manejadas por una PC al conectar el cable de programación y compilar un programa.

Debido a que el microprocesador se alimenta con $3{,}3VDC$ y las señales de entrada y salida manejadas por los buffers trabajan con tensiones de $5VDC$, se colocó una bornera para la entrada de alimentación con leds indicadores de presencia de tensión. No fue necesario implementar el circuito regulador de tensión de $3{,}3VDC$ recomendado en la hoja de datos, debido a que las tensiones necesarias se obtuvieron de una fuente de computadora de $300W$. Los valores mínimos y máximos permitidos por el módulo son $3{,}0VDC$ y $3{,}6VDC$ respectivamente, por lo que no se precisa de una regulación exacta de la tensión de alimentación.

Se utilizaron cinco buffers 74LS541 que permiten la protección y correcta interacción del microprocesador con el resto del sistema[3]. No se utilizó la señal de **ENABLE** para el control, dejándolos siempre habilitados, debido a que se colocaron solo a modo de protección. A la placa se soldaron zócalos para los mismos previendo la posibilidad de sustituirlos en caso de falla o rotura.

Los puertos no utilizados en el sistema y que pueden ser configurados como puertos de entrada o salida, se conectaron a los buffers con la posibilidad de ser configurados mediante *jumpers* en la entrada y salida de los buffers.

El PCB para esta placa se diseñó en dos capas, pero fue necesario realizar, de todas formas, varios puentes para unir pistas de ambos lados de la placa. También se le colocaron borneras de tornillo para facilitar su conexión con el resto del sistema. El detalle del PCB diseñado se puede observar en la figura C.7 del apéndice C. La figura siguiente muestra un esquema de la placa diseñada donde se detallan tanto los componentes como las señales involucradas.

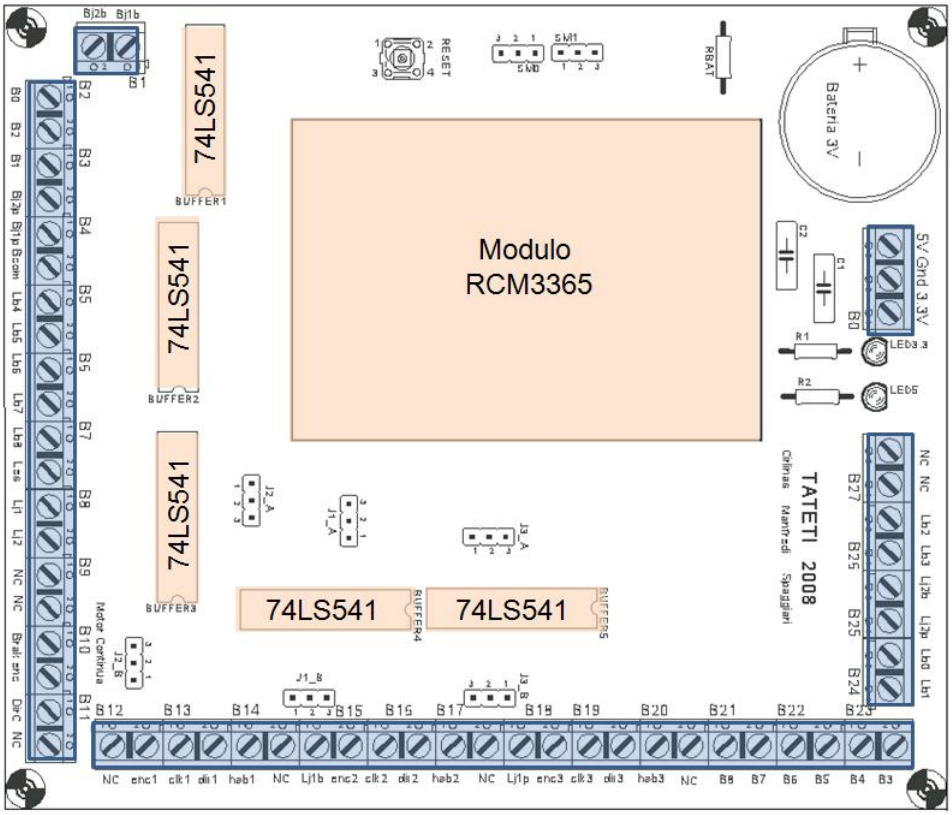


Figura 6.7: Esquema de la placa diseñada.

Fabricada la placa en fibra de vidrio, se montaron los componentes con especial cuidado en los de montaje superficial. Se realizaron las pruebas necesarias para verificar que el conjunto de componentes colocados funcionara correctamente. Cabe aclarar que algunos

[3]Referirse al capítulo 5 para obtener una descripción más detallada de los mismos

de los problemas mencionados anteriormente referentes a la configuración de los puertos, aparecieron en esta etapa de pruebas pero resultaron fácilmente solucionables.

La figura 6.8 muestra una foto de la placa armada con el módulo de Rabbit RCM3365 colocado.

Figura 6.8: Placa para el módulo RCM3365.

Capítulo 7

Desarrollo de Software

En este capítulo se estudiará todo lo vinculado con el software de control del juego, que como ya fue mencionado, deberá comandar todos los dispositivos hardware estudiados anteriormente (luces, botones y motores) y dotar al brazo con la inteligencia suficiente para jugar una partida de tatetí contra una persona.

Primero, se presentarán las condiciones generales de juego que se tuvieron que tener en cuenta al diseñar e implementar la arquitectura del software y las diferentes rutinas que terminaron componiendo el programa principal.

Luego, se expondrá y analizará el proceso de diseño, implementación y depuración de las diferentes rutinas. Primero se estudiará la rutina de inteligencia artificial del brazo, luego las rutinas del control de interfaz y de motores.

Finalmente, se describirá el comportamiento del programa principal del juego.

Todos los análisis se centrarán en los aspectos cualitativos de las rutinas implementadas intentando entrar en detalles particulares o vinculados con el lenguaje de programación solo cuando sea estrictamente necesario.

Para estudiar los detalles de cada rutina, el lector interesado puede consultar los comentarios de los archivos fuente del programa. También puede utilizar la herramienta *Library Function Lookup*, a la que se puede acceder presionando `CTRL+H` en el entorno de programación de Dynamic C, que permite observar rápidamente las descripciones de cada rutina de cada biblioteca particular[1].

Además, podrá encontrar los detalles relacionados con el lenguaje en el Manual de Usuario de Dynamic C [20].

[1] en caso que las bibliotecas del proyecto estén en el archivo `LIB.DIR`. Por más información al respecto consultar la página 42 del Manual de Usuario de Dynamic C [20]

7.1. Estudio de los requerimientos de software

El primer paso que se tomó en el proceso de diseño del software del juego fue reconocer las condiciones impuestas por la mecánica y la electrónica del juego, así como los requerimientos particulares del cliente y sus consecuencias.

Con esto se intentó llegar al momento de diseñar la arquitectura del programa con una visión más formada de cuál sería el comportamiento general que debería tener el software y cuáles serían las dificultades más importantes que debería superar. También permitió al grupo ya ir imaginando diferentes maneras de implementarlo.

A continuación se mencionan las diferentes condiciones que tuvieron que ser consideradas en la implementación del programa.

Se puede observar que a todos estos requerimientos se les suma el hecho de que el programa va a ser ejecutado en un módulo Rabbit RCM3365. Las consecuencias de esto son varias, y a pesar de que varias de ellas no fueron de importancia en este proyecto, vale la pena mencionarlas.

- El programa debe estar escrito en el lenguaje Dynamic C y debe ser compilado en el entorno de desarrollo Dynamic C 9.21 y no en algún otro.

  Esto, que resulta inevitable si se utiliza un módulo Rabbit, determinó que el grupo tuvo que familiarizarse primero con el lenguaje de programación ANSI C[2] y luego estudiar algunas de las diferencias y extensiones que presenta el Dynamic C.

  Si bien a primera vista podría pensarse que el costo en tiempo de adaptarse a este lenguaje propietario pudo haber sido alto, no fue el caso. Si bien Dynamic C no es ANSI C, es casi idéntico y muchas de las extensiones del lenguaje ofrecidas fueron valiosas herramientas en el desarrollo del programa.

- El software debe atender diferentes dispositivos (luces, botones y motores) de manera simultánea y cumpliendo determinados requisitos en la velocidad de procesamiento y de lectura/escritura de las diferentes señales.

  A modo de adelanto, la solución que se encontró para lograr manejar varios dispositivos a la vez con rutinas relativamente sencillas fue utilizar un esquema de multitasking cooperativo, cuyas características más relevantes se explicarán luego.

  Respecto a las necesidades de velocidad de procesamiento y de manejo de señales, se observa que la cota impuesta por la arquitectura del módulo y las características de los diferentes componentes es muy superior a la necesaria para el proyecto. La lógica del sistema es muy sencilla y los requerimientos de velocidad más exigentes entre lecturas o escrituras de señales caen dentro del rango de los milisegundos, muy poco estrictos si se considera que el microprocesador del RCM3365 corre a 44,2 MHz. Sin embargo, en caso que se hubiese estado en una situación comprometida al respecto, el grupo podría haber evaluado implementar parte del código en assembler.

[2] Una excelente referencia sobre este lenguaje de programación se puede encontrar en [21].

- El tamaño del código final no debe ser mayor a los 512 kBytes de la memoria flash del módulo.

  Este requerimiento no fue limitante para el proyecto. Desde un principio se estimó que esa capacidad era más que suficiente para el programa pues se supo de proyectos similares cargados en módulos con menor capacidad de memoria. En caso de que esa memoria no fuera suficiente, parte del programa podría ser almacenado en los 512 kBytes adicionales de la RAM respaldada por batería, si bien no sería una solución ideal pues almacenar todo en el sector de memoria flash hace al sistema más confiable.

  Afortunadamente este supuesto se cumplió y solo fueron necesarios aproximadamente unos 85 kBytes.

A estos requerimientos generales se les suman algunas condiciones particulares, cuyo cumplimiento ya no tiene tanto que ver con el módulo Rabbit sino con una programación inteligente de la lógica del juego y del control de dispositivos.

Entre las más importantes se encuentran:

- Condiciones en el manejo de motores.

  Es deseable que el software realice un buen control de la velocidad de los motores y coordine adecuadamente los movimientos conjuntos de motores para lograr buena precisión en las trayectorias del brazo y no producir desgastes innecesarios en los engranajes y correas. También se pretende que los movimientos sean lo más rápido posible, para evitar que el juego pierda dinámica. Hay que recordar que estos motores mueven objetos que pueden llegar a tener bastante inercia.

  Además, se debe lograr que en ningún movimiento alguna parte del brazo salga del área de juego pues se va a rodear la mesa con paneles de acrílico. En caso que se saliera, se podrían producir golpes que dañen el brazo o los acrílicos.

- Condiciones en el manejo de la interfaz y en la lógica del juego.

  Entre otras cosas, el programa debe guiar al usuario durante el transcurso de cada partida mediante un manejo agradable de las luces y botones. Se debe intentar que el proceso de comienzo de una partida sea lo más intuitivo posible y que estén bien señalizados los momentos donde se espere que el usuario realice una jugada o donde se haya terminado una partida.

- Condiciones en el manejo de señales.

  Como se va a trabajar con señales externas, que viajan hacia el módulo por un cable largo desde algunas de las placas de control de motores o la placa de control de interfaz, al leerlas va a ser importante tener cuidado con los ruidos eléctricos que puedan aparecer y al escribirlas se deberá respetar los mínimos en los tiempos de setup y hold que pudiera tener cada dispositivo.

  Un claro ejemplo del problema en la lectura es el sensado de botones, al que se le debe agregar una eliminación de rebotes. Otro ejemplo que será explicado detalladamente es el de la lectura de encoders, donde hay que tener algunos recaudos más.

Por su parte, se puede tomar como ejemplo de condición en la escritura de señales, a la generación de flancos en las señales de CLK de las placas de control de motores stepper. Como los mínimos en los tiempos de setup y hold del controlador L297 son mayores al tiempo que le toma al RCM3365 realizar dos escrituras consecutivas, es necesario agregar un delay entre escrituras.

- Condiciones en el manejo de situaciones de falla.

  El programa debe manejar las variables importantes inteligentemente y ser lo suficientemente robusto como para poder recuperar el estado del juego y limpiar el tablero cuando se produzca una falla en la alimentación o en el manejo de los motores o la interfaz. Con esto se busca evitar que sea necesario que una persona acomode el tablero de juego cuando ocurra alguna falla.

Por supuesto, se intenta que todo esto se cumpla manteniendo un buen estilo de programación, obteniendo un código fuente bien comentado y fácil de mantener al que se le pueda agregar o quitar elementos de manera sencilla.

Como ya fue mencionado, una vez reconocidos y analizados los ítems antes expuestos, el grupo pudo ver qué debería hacer y cómo se debería comportar a grandes rasgos el programa, pudiendo así comenzar a diseñar la arquitectura del programa.

El proceso de diseño e implementación de esta arquitectura se describirá en detalle en la siguiente sección.

7.2. El esquema de multitasking cooperativo

El multitasking es un esquema de programación mediante el cual diferentes tareas pueden compartir el uso del microprocesador así como del resto de los recursos hardware de un sistema (en nuestro caso son las luces, los botones y los motores). Utilizándolo dentro de un programa, se pueden controlar con rutinas sencillas varios dispositivos externos simultáneamente, lo que lo hace ideal para la implementación de software para sistemas embebidos de nuestras características.

El lenguaje Dynamic C dispone de dos maneras de implementarlo, dentro de las varias que existen. A saber:

- Multitasking cooperativo (o *cooperative*)
  Dentro de este esquema, cada tarea cede voluntariamente el control del microprocesador cuando completa su ejecución o tiene que esperar algún evento.
- Multitasking preferente (o *preemptive*)
  En este caso las tareas no ceden el control voluntariamente, sino que se les asigna un tiempo fijo de ejecución después del cual se les quita el control del microprocesador y se le cede a una tarea diferente.

Antes de diseñar la arquitectura del software, se analizaron cada una de las posibilidades, evaluando sus ventajas y desventajas, decidiéndose implementar el programa utilizando el multitasking cooperativo.

Primero, su implementación resulta más intuitiva. En este esquema, la interacción entre las tareas y los recursos del módulo es más fácil de controlar pues cada rutina puede mantener el control del microprocesador todo el tiempo necesario para hacer lo que tengan que hacer y no son interrumpidas, como sucede en el esquema preferente.

Además, dado que los dispositivos externos del juego necesitan ser atendidos en momentos bien definidos en el juego (por ejemplo, no se sensan botones mientras se están moviendo motores) y además no se van a controlar más de dos o tres dispositivos por vez (a lo sumo se moverán un par de motores y parpadeará una luz), lo más conveniente es dejar que las rutinas de control se ejecuten sin interrupciones y cedan el control del microprocesador cuando lo consideren conveniente. Si se conmutara estas tareas con otras que ya se sabe que no necesitan ser atendidas, se estaría agregando al programa un overhead totalmente innecesario.

En este sentido, el multitasking preferente es más adecuado cuando se debe garantizar que todas las tareas tengan la oportunidad de ejecutarse dentro de un período razonable de tiempo o sea necesario establecer un esquema de prioridades dentro de las tareas existentes. En el caso del proyecto TATETI, estas características no eran necesarias.

Dentro del lenguaje Dynamic C, este esquema se implementa con los *costates* y las *cofunctions*.

Los *costate* sirven para delimitar bloques de código que se comportan como una tarea independiente. Los *costates* deben implementarse dentro del loop principal del juego. Mientras todos los *costates* se comporten correctamente y cedan en algún punto el control del microprocesador, la ejecución de las tareas se dará casi en paralelo.

Por su parte, las *cofunctions* son rutinas que además de poder recibir argumentos y devolver resultados, pueden utilizar cualquiera de las sentencias de control utilizadas para

los *costates*. Dentro de un bloque `waitfordone` se pueden instanciar una o más de estas rutinas. Análogamente a los *costates*, mientras estas *cofunctions* se comporten correctamente, la ejecución de las rutinas se dará en paralelo.

Por más información sobre este tema, se puede consultar el capítulo 8 del libro - Embedded Systems Design using the Rabbit 3000 Microprocessor[22] o el capítulo 5 del Manual de Usuario de Dynamic C[20].

7.3. La arquitectura del software

Al momento de diseñar la arquitectura del software se intentó dividir el programa en bloques o elementos bien diferenciados y con funcionalidades diferentes. De esta manera se pretende obtener un código más claro y conciso, que permita implementar y depurar cada rutina por separado, así como agregar, quitar o modificar de manera sencilla las diferentes funcionalidades del sistema. También se procuró no dividir innecesariamente el sistema, pues eso obligaría al grupo a mantener más archivos fuente y manejar con más cuidado las diferentes variables y rutinas del programa.

Para ello, se definieron cuatro elementos, que terminaron formando tres bibliotecas (con rutinas, constantes y variables propias) y un programa. A saber:

El primer elemento del programa es la lógica del juego de tatetí. Dentro de esta lógica se encuentran todas las rutinas de la biblioteca `tatetiLib.LIB`, necesarias para hacer que el Rabbit pueda realizar una jugada dado el estado del tablero y las rutinas que permitan controlar si una partida aún no terminó o si lo hizo con ganador o con empate.

Si bien las constantes que se definen aquí tienen caracter auxiliar, no sucede lo mismo con las variables. Aquí se incluyen todas las vinculadas con la partida, como son el tablero o el ganador.

Luego, están los controles software de las luces, los botones y los motores. Dadas las grandes diferencias en la lógica de control y los requerimientos de cada uno, las rutinas de control de estos dispositivos externos se dividieron en dos grupos diferentes.

Se puede decir entonces que el segundo elemento del programa son las pertenecientes a la interfaz del usuario (luces y botones), condensadas en la biblioteca `ioLib.LIB`. La lógica de estas rutinas es sencilla y en caso de falla no se compromete ningún aspecto del juego.

Las constantes de esta biblioteca sirven para describir las características de cada elemento del hardware de la interfaz, como puede ser el nivel activo de las luces y de los botones. Por su parte, las variables sirven para almacenar el estado de estos elementos, como puede ser el estado de cada luz.

Las rutinas vinculadas con el control de los motores, agrupadas en la biblioteca `motoresLib.LIB`, componen el tercer elemento. Su lógica es sensiblemente más complicada y en caso de error se puede llegar a comprometer no solo al transcurso normal de una partida sino a la integridad física de los motores y del brazo.

Como es natural, las constantes definidas en esta biblioteca sirven en general para definir diferentes parámetros del control de motores, mientras que las variables incluidas se relacionan con la posición del brazo (se guarda la posición en que se encuentra cada una de las articulaciones en cada momento del juego) y de los casilleros del tablero (se guardan las coordenadas de hombro y codo en que se encuentra cada casillero así como las partes altas y bajas de cada rampa).

El cuarto y último elemento del software es el programa principal `main.C`. Esta rutina es la que se encarga de utilizar convenientemente las rutinas y variables de las tres bibliotecas para hacer funcionar al juego y recuperar al sistema en caso de fallas. Se puede decir que funciona como núcleo de toda la estructura de software del juego.

En la siguiente figura se muestra un pequeño diagrama de la interacción que tienen los diferentes elementos del software.

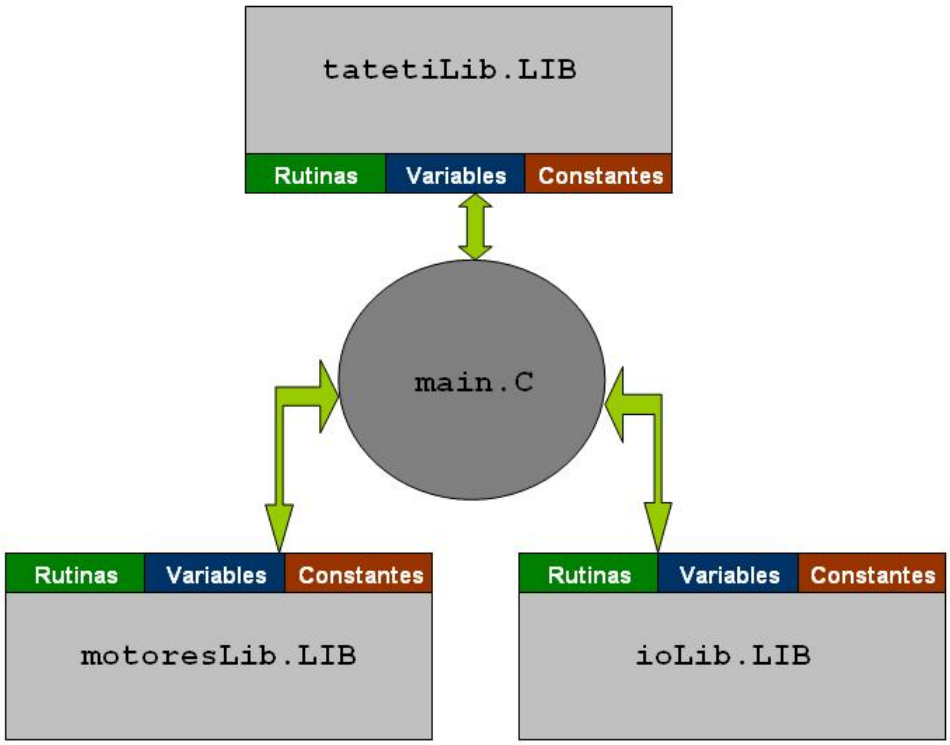


Figura 7.1: Arquitectura del software

Es importante aclarar que para que se puedan manejar las condiciones de falla correctamente, todas las variables de importancia de las diferentes rutinas deben ser almacenadas en el sector de memoria RAM con respaldo de batería agregándoles los atributos `bbram static`[3] para poder recuperarlas después de un reset.

Se puede observar en la figura que este esquema logra cumplir todos los objetivos mencionados en el primer párrafo de esta sección, agrupando de buena manera todas las funcionalidades. Por ello, el grupo decidió que esta fuera la arquitectura sobre la que iba a trabajar a lo largo del proyecto.

Se puede decir que en general se logró respetar este esquema, si bien en algunas rutinas aparecen interacciones débiles con elementos de otras bibliotecas. Cuando en la implementación del sistema surgió la necesidad de que rutinas de una biblioteca manejen variables o utilicen rutinas de otra biblioteca, se cuidó que la interacción sea mínima y controlada, tal como se mostrará más adelante en la documentación.

Una vez diseñada la arquitectura del sistema, se comenzó a diseñar, implementar y depurar cada una de las rutinas necesarias para el control del juego. En las siguientes secciones se explicará en detalle el procedimiento seguido.

[3]Por más información consultar las páginas 177 y 199 del Manual de Usuario del Dynamic C [20]

7.4. Respecto al proceso de depuración del software

Como es natural, el diseño y la implementación primaria de las diferentes rutinas del programa estuvieron cargadas de errores tanto de lógica como de sintaxis de la programación. Estos errores tuvieron que ser detectados y corregidos en un proceso iterativo de implementación y ensayo, que terminó convirtiéndose en la parte más complicada del desarrollo del software.

Para la depuración de detalles más vinculados con el hardware fue útil la placa de desarrollo[4] del RCM3365. Esta placa permitía acceder a todos los pines del módulo de manera sencilla y así poder conectar pulsadores o al generador de señal para simular en cada rutina el comportamiento de señales de entrada así como luces o puntas de osciloscopio para observar el comportamiento de las diferentes señales de salida.

Por otra parte, para lograr superar las dificultades vinculadas con el lenguaje de programación fue fundamental el uso de las herramientas de compilación y depuración del entorno de programación Dynamic C. Gracias a ellas se pudo detectar rápidamente los errores más finos en la implementación de las rutinas.

Por ejemplo, sólo con el PC se pudo depurar completamente la rutinas de inteligencia del brazo. Un caso similar es el de las rutinas de control de interfaz, para las que no fue necesario tener el tablero terminado, sino que bastó conectar a la placa de desarrollo la pequeña placa con luces y botones que se muestra en la siguiente figura.

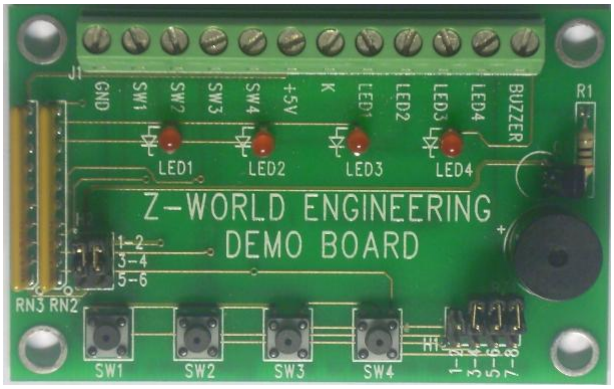


Figura 7.2: Placa con luces y botones utilizada en la depuración del software.

Lamentablemente, las herramientas del Dynamic C no se pudieron utilizar en las rutinas de control de motores por diferentes razones.

Primero, como las rutinas tenían requerimientos estrictos en el timing de las señales, no era adecuado obligar al módulo a comunicarse con el PC via instrucciones de debug. Esto enlentecía la ejecución y hacía que las señales no tuvieran el comportamiento esperado.

Segundo, dado que no se tenían las herramientas necesarias para simular las señales de entrada y observar las salidas de todos los puertos que utilizaban estas rutinas, la mayoría de su proceso de depuración se tuvo que hacer con el Rabbit conectado al juego y por ello no se pudo utilizar el PC.

Dicho problema surgió de la imposibilidad del Rabbit de correr el programa principal

[4]Las características de esta placa se pueden observar rápidamente en la figura 8.1 del capítulo 6

y a la vez comunicarse con el PC. Como el juego ocupa casi la totalidad de los puertos disponibles, incluso aquellos que usa para comunicarse con el Dynamic C (correspondientes al puerto serie A)[5], no se pudieron utilizar las herramientas de depuración del entorno de programación.

Como regla general, dado que resulta mucho más complicado detectar y diagnosticar errores en tiempo de ejecución con el Rabbit conectado al hardware del juego que hacerlo en un ambiente más controlado, como es el del Dynamic C, primero se intentó depurar lo más posible las rutinas con programas auxiliares que corrieran con el Rabbit en la placa de desarrollo y se comunicaran con la entrada/salida estándar de la PC.

Una vez que se tenía confianza suficiente en la lógica implementada, se migraba de la placa de desarrollo al juego cada rutina por separado. Allí se testeaba nuevamente y se le hacían los ajustes necesarios. Todas las grandes modificaciones que fueran necesarias se hacían llevando de nuevo al Rabbit a la placa de desarrollo.

Como el Rabbit no tenía la posibilidad de comunicarse con el PC cuando estaba conectado al juego, y por lo tanto no se podían utilizar ninguna de las herramientas de depuración del Dynamic C, se fue agregando código extra que comunicara a través de las luces del tablero en qué parte de la ejecución se encontraba el programa. De ese modo, se pudo localizar cada error dentro de sectores de código reducidos. Cabe aclarar que esto no fue la panacea; los errores más complejos que llevaban la ejecución a lock-ups obligando al watchdog del Rabbit a resetear el sistema fueron corregidos cuando se revisaba el código en su totalidad.

Con esta política de desarrollo incremental y depuración se logró minimizar el tiempo dedicado al estudio y resolución de errores.

[5]Tal como se explicó en el capítulo 6

7.5. Rutina de inteligencia artificial del brazo

El primer paso en el proceso de diseño de la lógica que permitiera al microprocesador jugar al tatetí fue decidir qué algoritmo utilizar. Se intentó buscar un algoritmo sencillo y que presentara un nivel de dificultad agradable al jugador pues se pretende que el juego, además de despertar el interés por la tecnología a niños y jóvenes, resulte divertido.

Si se implementa un algoritmo de juego perfecto, no hay posibilidad de que el primer jugador pierda, y el segundo jugador solo puede aspirar a empatar la partida. Esto no es deseable pues puede resultar un tanto frustrante para los niños.

Por otra parte, si el algoritmo utilizado es demasiado sencillo, el jugador le ganaría al brazo con mucha facilidad, haciendo al juego un tanto aburrido.

Se estudiaron en Internet diferentes algoritmos utilizados en la implementación de juegos de características similares. Dentro de las variantes encontradas, merece mención el algoritmo de poda alfa-beta [23], cuyo estudio puede resultar interesante al lector.

Luego de una extensa investigación, se encontró un juego de tatetí [24] programado en C++ que tenía un algoritmo de juego que proporcionaba la inteligencia de juego justa para nuestra aplicación.

Este programa se utilizó como punto de partida en la implementación de la lógica de control de juego. Cabe aclarar que si bien las características generales y el algoritmo utilizado terminaron siendo muy similares, el código fue completamente reescrito y se agregaron varios elementos originales.

El algoritmo utilizado en la lógica del juego se muestra a continuación:

1. Se verifica si el brazo puede ganar en esa jugada. Si puede hacerlo, gana la partida. En caso contrario, se va al paso 2.

2. Se verifica si el rival puede ganar en esa jugada. En caso afirmativo, se bloquea la jugada. En caso que el rival tampoco pueda ganar, se va al paso 3.

3. Se elige un casillero al azar.

Como se puede observar, el brazo va a ganar si lo puede hacer y va a bloquear cualquier jugada ganadora que tenga su contrincante, pero en las primeras jugadas de la partida selecciona el casillero al azar, evento que el rival puede aprovechar para ganar el juego.

Esta inteligencia fue implementada en la rutina principal `jugadaRabbit` que hace uso de diferentes rutinas auxiliares de la biblioteca `tatetiLib.LIB`. A su vez, esta rutina es invocada dentro del programa principal del juego en los momentos que se necesita que el brazo realice una jugada.

Para verificar si un jugador dado puede ganar la partida, se fija si hay en el tablero filas, columnas o diagonales donde se hayan colocado fichas de ese jugador en dos de sus casilleros y esté libre el tercer casillero. Siguiendo este mismo procedimiento para cada jugador, implementa tanto el primer paso del algoritmo como el segundo.

Por su parte, para el tercer paso del algoritmo genera al azar números enteros correspondientes a casilleros del tablero hasta encontrar uno que corresponda a un casillero vacío.

Tal como fue mencionado anteriormente, en caso de querer obtener más detalles respec-to a la implementación de estas rutinas, el lector puede consultar los fuentes de la biblioteca `tatetiLib.LIB`. También puede consultar la *Library Function Lookup* y buscar esta biblioteca.

Para depurar y verificar esta rutina, se creó un programa auxiliar que toma del teclado del PC jugadas de una persona y e imprime en su pantalla la jugada del Rabbit. Cabe destacar que el esqueleto de este programa está basado en el del programa principal (que se explicará más adelante), dado que su comportamiento es prácticamente idéntico; solo se tiene que capturar la jugada desde el tablero en vez de hacerlo desde el teclado y colocar fichas en vez de mostrar en pantalla.

7.6. Rutinas de control de interfaz

Dentro del control software de interfaz tenemos dos rutinas básicas, la de comando de luces y la de sensado de botones, implementadas como *cofunctions*. A partir de ellas se construyen rutinas más complejas y dedicadas a atender diferentes necesidades que surgen en la implementación del programa principal, como son el proceso de selección de jugadores, la captura de jugadas en el teclado o la señalización de los diferentes momentos de juego.

Por ello, se explicará primero el funcionamiento de estas dos rutinas básicas y luego se explicará cómo se utilizaron para construir el resto de las rutinas que componen el control de la interfaz.

La rutina de control de luces `setLight` es muy sencilla; como ya fue explicado en el capítulo 5, para encender o apagar una de las luces del tablero solo es necesario colocar en nivel alto o bajo la señal correspondiente.

También es muy simple la rutina de sensado de botones `getBoton`. En este caso se lee la señal correspondiente y se le hace una eliminación de rebotes. En caso de detectar que la señal está activa, se hace una segunda lectura un tiempo después. Si ambas lecturas son iguales, se espera a que el botón se suelte y se valida. Esto evita que el sistema considere que se ha presionado y soltado varias veces el botón cuando aparece algún tren de pulsos espurios causado por el rebote en los contactos del botón o cuando el usuario lo mantiene apretado durante mucho tiempo.

En base a estas dos rutinas se crearon el resto de las rutinas que componen la biblioteca `ioLib.LIB` de control de interfaz. Estas rutinas se pueden dividir en dos grandes grupos según su funcionalidad. Por un lado tenemos las de entrada de información, como son la rutina de selección de jugador `getJugadores` y la de captura de jugadas `getJugadaHombre`. Por el otro están las de salida de información o de señalización, como la que indica cuál fue el ganador `mostrarGanador`.

Respecto a la implementación de estas rutinas, aparece como importante el uso de timeout en las rutinas de entrada de información via tablero, para evitar que en el caso de que el usuario se haya ido el juego permaneciera a la espera indefinidamente. Esto sirve para que el programa principal, que es quien las instancia, pueda tomas las medidas correspondientes en caso de que se haya detectado un timeout.

La depuración de estas rutinas se realizó incluyendo cada rutina en programas auxiliares sencillos que permitían sensar botones y comandar luces desde la placa de desarrollo tal como se haría en el tablero del juego. Una vez que el tablero del juego estuvo listo, se conectó el Rabbit a la placa de control de interfaz y se volvieron a probar.

7.7. Rutinas de control de motores

Las primeras rutinas de control de motores en implementarse fueron `driveMotorHombro`, `driveMotorCodo`, `driveMotorMuneca` y `driveMotorMano`. Ellas son las encargadas de controlar las placas de potencia de cada motor del brazo de manera de llevar la articulación correspondiente desde la posición inicial en que se encuentre a la posición destino que se le indique. En base a ellas se creó la rutina `goToCasillero`, que es la encargada de llevar el brazo a cada uno de los casilleros del juego controlando la trayectoria recorrida para no golpear las rampas o la mampara de acrílico del juego.

También se crearon las rutinas de *homing* de los motores del hombro y del codo `goHomeHombro` y `goHomeCodo` y en base a ellas se creó `goHome`, que lleva al brazo a su posición de *home* controlando al igual que `goToCasillero` la trayectoria seguida.

Es importante aclarar que todas estas rutinas se implementaron como *cofunctions* y como estamos trabajando con multitasking cooperativo, estas rutinas se pueden instanciar simultáneamente y por lo tanto mover varias articulaciones en conjunto.

En las siguientes subsecciones se explicará en detalle el comportamiento de todas estas rutinas

Además, se implementaron `colocarFicha` que permite retirar una ficha de una rampa y colocarla en un casillero dado, `retirarFicha` que permite retirar una ficha del tablero y devolverla a su rampa y `limpiarTablero` que retira todas las fichas del tablero.

Como el único cometido de estas últimas rutinas es el de hacer el código del programa principal, compactando en una sola sentencia el código necesario para lograr que el brazo realice operaciones sencillas de posicionamiento y de manejo de fichas, no serán estudiadas en este capítulo. En caso de considerarlo necesario, el lector podrá estudiarlas directamente del archivo fuente de la biblioteca `motoresLib.LIB`.

7.7.1. Rutinas de control de los motores del hombro y del codo

Las rutinas `driveMotorHombro` y `driveMotorCodo` son las encargadas de controlar los motores de paso de las articulaciones del hombro y del codo respectivamente. Consultan en memoria la posición actual de la articulación y la posición del casillero destino[6] y en base a ello, controlan las señales **DIR**, **HAB** y **CLK** de las placas de potencia que fueron descritas en el capítulo 4.

A pesar de que los motores de paso se pueden controlar en loop abierto, determinando la posición de su rotor en función de la cantidad de pasos dados desde el *home*, se decidió controlarlos en loop cerrado, utilizando la realimentación de los encoders.

Esta decisión se tomó durante las primeras pruebas de movimientos de motores, donde se evaluó utilizar sensores ópticos para determinar los finales de carrera y utilizarlos para el posicionamiento de las articulaciones. En ese momento se consideró que dado el estado del brazo y de las rutinas, con este tipo de realimentación se podría lograr mejor precisión en los movimientos.

[6] Cabe aclarar que el valor de las coordenadas de cada posición de interés dentro del tablero de juego depende del montaje del brazo y de la posición de la franja de *home* de cada articulación, por lo que tuvieron que ser determinadas experimentalmente.

Es importante aclarar que esto no quita que el control se pueda hacer en loop abierto. Incluso, el grupo implementará rutinas alternativas de control por loop abierto para comparar resultados y eventualmente migrar el diseño a este tipo de sistema.

Para ello, sigue un procedimiento muy sencillo. A saber:

1. Se setea **DIR** de manera que el motor gire según el sentido deseado, que se determina consultando en memoria el valor de la coordenada de la posición destino y comparándolo con la posición actual.
2. Se sube **HAB** para alimentar al motor con una tensión superior.
3. Se dan pulsos en **CLK** a una velocidad controlada mientras se va sensando el estado de la señal del encoder y actualizando la posición de la articulación.
4. Cuando se detecta que se ha llegado al destino, se deja de dar pulsos en **CLK** y se baja **HAB** para dejar al motor en reposo con una tensión menor.

Si bien el código necesario para implementar este comportamiento es mínimo, se deben agregar diversos controles para evitar problemas en el posicionamiento.

Es posible que estas rutinas sean las más importantes de todo el software del juego. Como dependen de ellas todos los movimientos del brazo y el control de las fichas, elementos centrales del juego, los requerimientos de precisión son mayores a los del resto de los motores. Es de vital importancia que el posicionamiento de las articulaciones del hombro y del codo cumplan cotas mínimas de precisión. Si esto no sucediera, el brazo podría presentar serios problemas en el manejo de fichas, obligando al personal de Ciencia Viva a abrir el juego y corregir el error manualmente.

En los siguientes apartados se explican los diversos controles agregados en la lectura de encoders y en la actualización de la posición de las articulaciones durante el movimiento como maneras de alcanzar estas cotas de precisión. También se estudiará cómo y por qué se hace volver al brazo a su posición de *home* entre movimientos. Finalmente, se mencionarán las características más relevantes del control de velocidad implementado.

7.7.1.1. Lectura de los encoders

Tal como fue mencionado en el capítulo 4, el pasaje del sensor entre una franja negra y una transparente se traduce en un flanco en la señal de salida. Como es con estos flancos que el sistema controla la rutina puede ubicar al brazo en cada momento del juego.

Se implementó un doble control en la validación de flancos detectados en las señales del encoder.

Primero, cada flanco se valida después de leer en pasos consecutivos el nuevo valor; se deben leer un cero y dos unos cuando se pasa de una franja transparente a una negra y un uno y dos ceros cuando se pasa de una negra a una transparente. Como las franjas comunes en los discos tienen un ancho en pasos mayor a 4 o 5, si se detectara que la señal no se mantiene constante dos pasos después de un flanco, se podría saber que no se trata de una franja del disco. Esto provee cierta inmunidad al sistema frente a mugre o imperfecciones en el disco del encoder.

Además, para hacer un poco más robusta la lectura, en cada paso se repite dos veces el sensado con unos pocos milisegundos de diferencia (este tiempo es configurable). De esta manera se puede verificar en la mayoría de los casos si el valor detectado es válido o fue causado por ruido eléctrico, un resplandor en el disco o algún otro factor extraño.

Cabe aclarar que esta lógica se utiliza en la lectura de las señales de los encoders de todas las articulaciones y no solo en las del hombro y de la muñeca.

7.7.1.2. Control de posición

Como ya fue explicado anteriormente, los discos de los encoders del hombro y del codo están graduados cada dos grados. Esta información la utilizan las rutinas de movimiento para actualizar en cada momento del movimiento la posición de la articulación. En particular, con cada flanco detectado en la señal del encoder, las variables que almacenan la posición de cada articulación se incrementan o decrementan de a dos según nos alejamos o nos acercamos al *home* respectivamente (girando en sentido horario en el hombro o antihorario en el codo).

En un principio, pareció razonable definir a la condición

$$posicionBrazo = posicionDestino$$

como final de cada movimiento (cuidando de definir la posición destino como un número par).

Si bien esta lógica es sencilla y parece funcionar correctamente, trajo algunos problemas de precisión que se explican en el siguiente ejemplo.

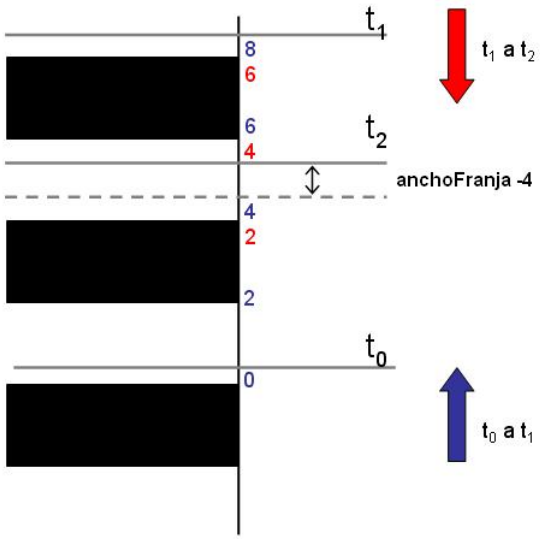


Figura 7.3: Posicionamiento con la primer lógica propuesta para la actualización de posición de las articulaciones del hombro y del codo.

Considere que en el instante inicial la articulación se encuentra en la posición de *home*, que en la figura se puede reconocer como la posición 0 azul. Si ahora se obliga mover hasta

llegar al ángulo 8, la rutina irá dando pulsos en la señal de **CLK** hasta detectar cuatro flancos en la señal del encoder. Esto dejaría al brazo posicionado en el 8 azul, dos pasos después del final de la segunda franja negra (los dos pasos que se usan para validar el flanco).

Si ahora se desea ir al ángulo 4, se movería la articulación hasta detectar dos flancos en la señal del encoder. Esto ubicaría la articulación sobre el 4 rojo dos pasos antes del inicio de la segunda franja negra.

Como se puede observar, existe una distancia igual al ancho equivalente en pasos de una franja menos cuatro entre la posición 4 azul, a la que se llega moviéndonos desde el *home*, y la posición 4 rojo, a la que se llega utilizando esta lógica de actualización de posición si nos movemos desde una posición mayor.

En principio se consideró que esta diferencia era tolerable, durante los ensayos se observó que esta diferencia en algunos movimientos se llegaba a traducir varios centímetros, mucho más de la tolerancia máxima que se pretendía alcanzar.

El problema de las diferencias reales de la posición del brazo cuando se llega a un destino desde una posición menor o mayor se resolvió modificando la lógica de actualización de la posición de cada articulación.

Si se determina la condición de final del movimiento se determina con la condición

$$posicionDestino \leq posicionBrazo < posicionDestino + 2$$

si nos movemos desde una posición menor y

$$posicionDestino - 2 \leq posicionBrazo < posicionDestino$$

cuando nos movemos de una posición mayor.

Repitiendo la experiencia anterior, partiendo desde el *home* la rutina moverá la articulación hasta cumplir la primer inecuación, llegando a la misma posición 8 azul. Sin embargo, cuando se va desde allí a la posición 4, se intentará cumplir la segunda condición y se moverá el motor hasta posicionar la articulación sobre el 2 rojo.

Como se puede observar, la diferencia entre las posiciones a las que se llega desde cada lado del disco es de solo cuatro pasos. Si bien esto no resulta en una gran mejora en la articulación del codo, donde una franja común tiene un ancho de alrededor unos 6 pasos, mejoró sensiblemente el posicionamiento de la articulación del hombro, cuyas franjas tienen un ancho de poco más de 10 pasos. En los dos casos, ese error de dos pasos se corrige invirtiendo señal de dirección y dando dos pulsos más una vez determinado el final del movimiento.

Esta lógica fue la que quedó implementada en la versión final de las rutinas de control de los motores del hombro y del codo.

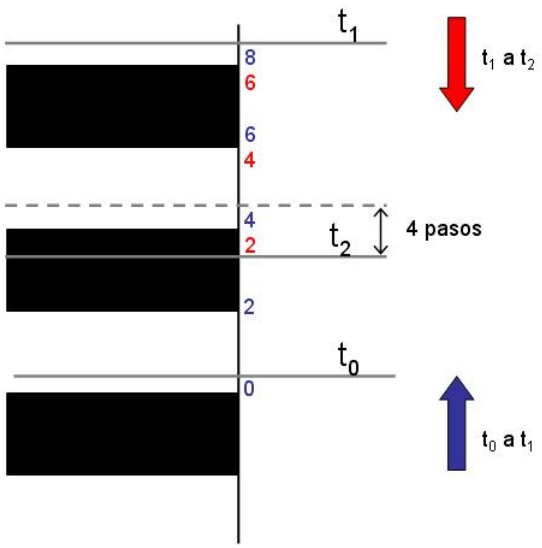


Figura 7.4: Posicionamiento la lógica final de actualización de posición de las articulaciones del hombro y del codo.

7.7.1.3. *Homing*

Además del fenómeno descrito en el apartado anterior, existen otras fuentes de error que determinan que el brazo vaya perdiendo precisión en la posición de las articulaciones con motores stepper cuando realiza secuencias largas de movimientos; por ejemplo, se pueden ir perdiendo pasos o se pueden hacer lecturas erradas en los encoders.

Por ello, para eliminar el arrastre y la acumulación de estos errores es necesario colocar todas las articulaciones del brazo en su posición de home después de haber realizado algunos movimientos. En particular, en el programa se manda a *home* todos los motores del brazo después de colocar o retirar cada ficha, operación que toma dos movimientos.

Para llevar las articulaciones del hombro y del codo a esa posición, se implementaron las rutinas dedicadas `goHomeHombro` y `goHomeCodo` que se incluyen en la biblioteca `motoresLib.LIB`.

Primero setean la dirección de manera de moverse hacia el *home*.

Luego, dan pasos en la señal de **CLK** y van contando el ancho en pasos de cada franja del disco del encoder que pasan. Cuando se llega a la franja negra gruesa que marca el *home* y la cuenta de pasos de franja supera el valor normal, frenan el motor. El control de velocidad de este movimiento es prácticamente idéntico al de las rutinas de posicionamiento

Finalmente, cambian de dirección y llevan la articulación al inicio de la primer franja blanca del disco. En ese momento se setea la posición en cero.

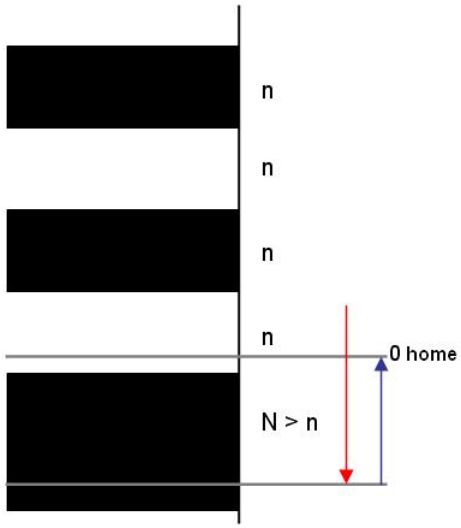


Figura 7.5: *Homing* de los motores de las articulaciones del hombro y del codo.

7.7.1.4. Control de velocidad

La regulación de velocidad constituye una parte esencial de las rutinas de control de motores pues de ella depende en gran medida el buen funcionamiento de la articulación y la preservación de los componentes mecánicos del brazo. En la siguiente figura se muestra el comportamiento de la velocidad en función de la posición que se implementó en las rutinas de control de los motores de paso.

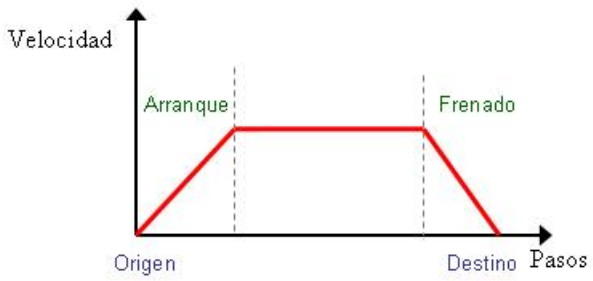


Figura 7.6: Comportamiento de la velocidad en función de la posición en las articulaciones del hombro y del codo.

El movimiento comienza con poca velocidad para que el motor trabaje en su zona de mayor torque[7] y pueda vencer todos los pares de fricción estática que la articulación ejerce en su rotor.

Una vez que se pudo vencer esa resistencia, se va incrementando paulatinamente la velocidad a lo largo de una distancia fija hasta llegar a un máximo.

La selección de esta velocidad máxima debe hacerse con cuidado. Por un lado, con una velocidad máxima alta se reducen tiempos de los movimientos del brazo y también se reduce el ruido emitido por el perfil de aluminio que conforma el hombro. Sin embargo, con

[7]Esta característica fue explicada en el capítulo 3

velocidades demasiado altas el par aplicado por el motor cae por debajo del necesario para vencer los rozamientos dinámicos y el rotor empieza a patinar, causando serios problemas de precisión en el movimiento.

Sobre el final del trayecto se reduce lentamente la velocidad para obtener un frenado parejo y lograr que la articulación llegue con la menor energía cinética posible al destino. Esto reduce el estrés sufrido por las correas y los engranajes, a la vez que mejora la precisión del movimiento. La duración de esta etapa se determina según la distancia a la posición destino, al igual que en el arranque.

Como cada articulación tiene comportamientos estáticos y dinámicos diferentes, fue fundamental ajustar correctamente los diferentes parámetros del control de velocidad para obtener una buena precisión en los movimientos y evitar dañar los engranajes y correas. Para ello se realizaron pruebas intensivas de movimiento de cada motor por separado y del brazo en su conjunto.

7.7.2. Rutina de control del motor de la muñeca

La rutina encargada del control del motor de paso de la muñeca es la `driveMotorMuneca`. Si bien la lógica que controla las señales de la placa de potencia del motor de la muñeca es idéntico al explicado para el hombro y el codo, existen algunas diferencias en el control de posicionamiento y el control de velocidad.

La primera diferencia es que la muñeca puede tomar solamente tres posiciones diferentes. Una posición hacia arriba para transportar las fichas a través del tablero de juego sin golpear las que ya estén colocadas. Luego, una posición al medio para una vez recogidas del tablero, el brazo pueda devolver las fichas en la parte alta de la rampa. Finalmente, una posición hacia abajo para tomar o soltar fichas a nivel del tablero.

En función de la posición inicial y la detección de flancos de subida se va actualizando la variable de posición de la articulación y se comanda el movimiento. Por ejemplo, si estamos en la posición de abajo y deseamos ir a la de arriba, se dan pulsos en la señal de **CLK** hasta detectar dos flancos de subida (el primero corresponde a la posición del medio y el segundo a la de arriba).

En este caso aparecería en la franja del medio el mismo problema de posicionamiento discutido para el hombro y el codo. Sin embargo, como la franja negra del medio es bien fina[8], la diferencia entre la posición del medio cuando se viene desde abajo o cuando se viene desde arriba es despreciable. Debido a esto, no fue necesario agregar lógica que detenga el movimiento en el mismo lado de la franja.

El control de velocidad es también diferente. Como no tenemos en el disco del encoder franjas intermedias que nos permitan saber aproximadamente en qué parte de un trayecto nos encontramos, la rampa de velocidad de este motor se implementó en función de los pasos y no de la posición.

Además del control de velocidad y de posición, se agregó para esta articulación un control de timeout para el movimiento. Esta protección es útil para situar al motor en una posición conocida partiendo de una posición inicial desconocida (sin tener que implementar una complicada rutina de *homing*), tal como sucede al principio del programa principal.

7.7.3. Rutina de control del motor de la mano

La rutina que maneja el motor de la articulación de la mano es la `driveMotorMano`. Como esta articulación tiene la particularidad de tener un motor de continua en vez de uno de paso, fue necesario crear en esta rutina una lógica diferente a la implementada para los motores del hombro, el codo y la muñeca.

En este caso se controlan las señales **DIR** y **BRAKE** descritas en el capítulo 4. El procedimiento seguido es también muy sencillo:

1. Se setea **DIR** según el sentido de giro deseado.
2. Se baja **BRAKE** para que el motor se mueva.
3. Se sensa el estado de la señal del encoder y cuando se detecta un flanco se detiene el movimiento subiendo **BRAKE**.

[8] Como se puede observar en la figura 4.16

Análogamente a la rutina de control de la muñeca, como la mano solamente tiene dos posiciones, se actualiza la posición de la articulación cuando se detecta un flanco válido en la señal del encoder.

Como ya fue explicado, la velocidad de giro normal del motor está impuesta por el valor de la señal **MAG**, que no puede ser manipulada por software. Al momento de la calibración, se puso el mínimo valor necesario para hacer que el motor pudiera vencer el peso de las pinzas y las levantase a una velocidad razonable cuando la muñeca está hacia abajo.

Sin embargo, cuando se cierran las pinzas con la muñeca hacia abajo la velocidad resultó demasiado alta y se tuvo que agregar pequeños pulsos en la señal de **BRAKE** cuando la mano se cierra. Con esto se pudo reducir el valor medio del voltaje que la placa de potencia impone en bornes del motor durante el movimiento y por lo tanto la velocidad de cerrado.

Al igual que en la rutina de control de motor de la muñeca, se implementó un control de timeout al igual que en la muñeca, que permite devolver la mano a una posición conocida al inicio del programa. Además de esto, permite reintentar atrapar una ficha cuando no se logra hacerlo en el primer intento.

Si detecta un timeout cuando está cerrando la mano, cosa que sucede cuando no puede tomar bien una ficha, abre la mano, espera un tiempo y las vuelve a cerrar. En general, cuando una ficha está mal ubicada, esa apertura hace que la ficha se mueva y vuelva al centro del casillero, permitiendo que la mano la pueda agarrar cuando cierren nuevamente las pinzas.

7.7.4. Control de trayectorias

Como ya fue mencionado, se debe asegurar que el brazo nunca salga del perímetro de la mesa (y por lo tanto golpear la mampara de acrílico) o toque alguna de las rampas o las fichas que puedan estar sobre ellas. Para lograrlo, se implementó una lógica de control de trayectorias en las rutinas `goToCasillero` y `gHome`, que son las encargadas de controlar los movimientos hacia los casilleros y hacia el *home*.

Como solo se dispone de rutinas que permiten mover las articulaciones entre punto y punto (sin ningún tipo de control de trayectoria), se tuvo que descomponer cada movimiento conflictivo en dos o más partes, obligando al brazo a pasar por puntos intermedios en el tablero. Seleccionando convenientemente estos puntos, y luego de varias pruebas, se pudo lograr que la posición del brazo nunca salga de la zona permitida.

7.7.5. Programas auxiliares de calibración de posiciones

Una vez que se tuvo implementadas todas las rutinas de control de motores, fue necesario determinar las coordenadas de cada uno de los casilleros y del inicio y del final de cada una de las rampas para que el brazo pudiera colocar o retirar las fichas en el tablero de juego durante una partida.

En un principio, se cargaron en el código fuente valores aproximados para cada una de las coordenadas y luego se fueron probando con el programa principal y modificando. El grupo rápidamente detectó que este método era muy poco práctico pues no solo la

primer aproximación era en general mala, sino que cada vez que se cambiaba el valor de una coordenada se tenía que volver a grabar en la memoria flash del Rabbit el programa principal, conectar el módulo al juego y comenzar una nueva partida.

Debido a esto, pareció razonable crear algunas herramientas software que aliviaran el trabajo de la calibración. En particular, se implementaron dos programas auxiliares, `calibracionMotores.C` y `testCalibracionMotores.C`, que serán descritos a continuación.

El programa `calibracionMotores.C` permite al usuario conocer las coordenadas de cualquier punto al que pueda acceder el brazo. Para ello, tiene que utilizar los botones del tablero para llevar las articulaciones del hombro y del codo al lugar deseado. En caso de considerarlo necesario, también puede comandar la muñeca y la mano para verificar que el brazo puede tomar y soltar una ficha correctamente en esa posición. Una vez hecho esto, debe presionar el botón **COMENZAR** para observar en las luces del tablero las coordenadas del hombro y del codo.

Haciendo uso de esta aplicación, se pudo obtener un valor aproximado de cada una de las veintiséis coordenadas de interés dentro del plano de juego y se las introdujo dentro de la biblioteca de control de motores.

Una vez hecho esto, se afinó con `testCalibracionMotores.C` la calibración de cada posición por separado. Este programa permite al usuario presionar un botón del tablero y lograr que el brazo vaya allí y vuelva rápidamente, reduciendo drásticamente el tiempo que insumía cada prueba de calibración. Cuando se detectaba que algún valor no era correcto, se corregía directamente en el código fuente del programa y se volvía a ejecutar esta aplicación. Para lograr determinar los valores óptimos, bastó en general con un par de iteraciones.

En el código fuente de estos programas se pueden encontrar las instrucciones de uso y comentarios sobre los detalles más importantes de su comportamiento y su implementación.

7.8. Programa principal del juego

Como ya fue mencionado, el programa principal es el núcleo de la estructura software del juego. Maneja todos los dispositivos hardware haciendo uso de la rutina de inteligencia y las diferentes rutinas de control para lograr que el brazo pueda jugar al tatetí contra una persona. Además, se encarga de recuperar al sistema frente a fallas en la alimentación.

Como fue mencionado anteriormente, este comportamiento se logró implementando el programa dentro del esquema del multitasking cooperativo. En la siguientes figuras se muestran los esquemas de las diferentes tareas que ejecuta simultáneamente el programa principal.

El *costate* de inicialización `inicializacion` es el encargado de inicializar todas las variables del sistema, limpiar el tablero en caso que sea necesario y finalmente dejar al juego en espera de una nueva partida. Su comportamiento general se puede observar en el diagrama de flujo 7.7. En caso que el lector desee obtener más detalles, puede consultar los comentarios del código fuente que se incluyen en el CD.

El *costate* `checkBotonComenzar`, es el que sensa en cada momento del juego el estado del botón **COMENZAR**. En caso de detectar que se presionó ese botón, detiene el juego, limpia el tablero y deja el juego en espera de una nueva partida. Como política, se decidió que sólo atienda una petición en este botón cuando el brazo se encuentre detenido.
En el diagrama de flujo 7.8 se muestra su comportamiento.

El *costate* `controlJuego` define el comportamiento del brazo durante cada partida. Se encarga de la seleccion de jugadores, atiende las jugadas de cada uno, coloca las fichas y verifica si hubo ganador o empate. El diagrama de flujo de este bloque se puede observar en la figura 7.9

Finalmente, el *costate* `idle` realiza pequeñas demostraciones del movimiento del brazo a la vez que enciende y apaga luces del tablero si pasó una cantidad de minutos grande (este tiempo es configurable) desde la última vez que alguien interactuó con el juego. Esto sirve para atraer visitantes al juego. El diagrama de flujo de este bloque se puede observar en la figura 7.10

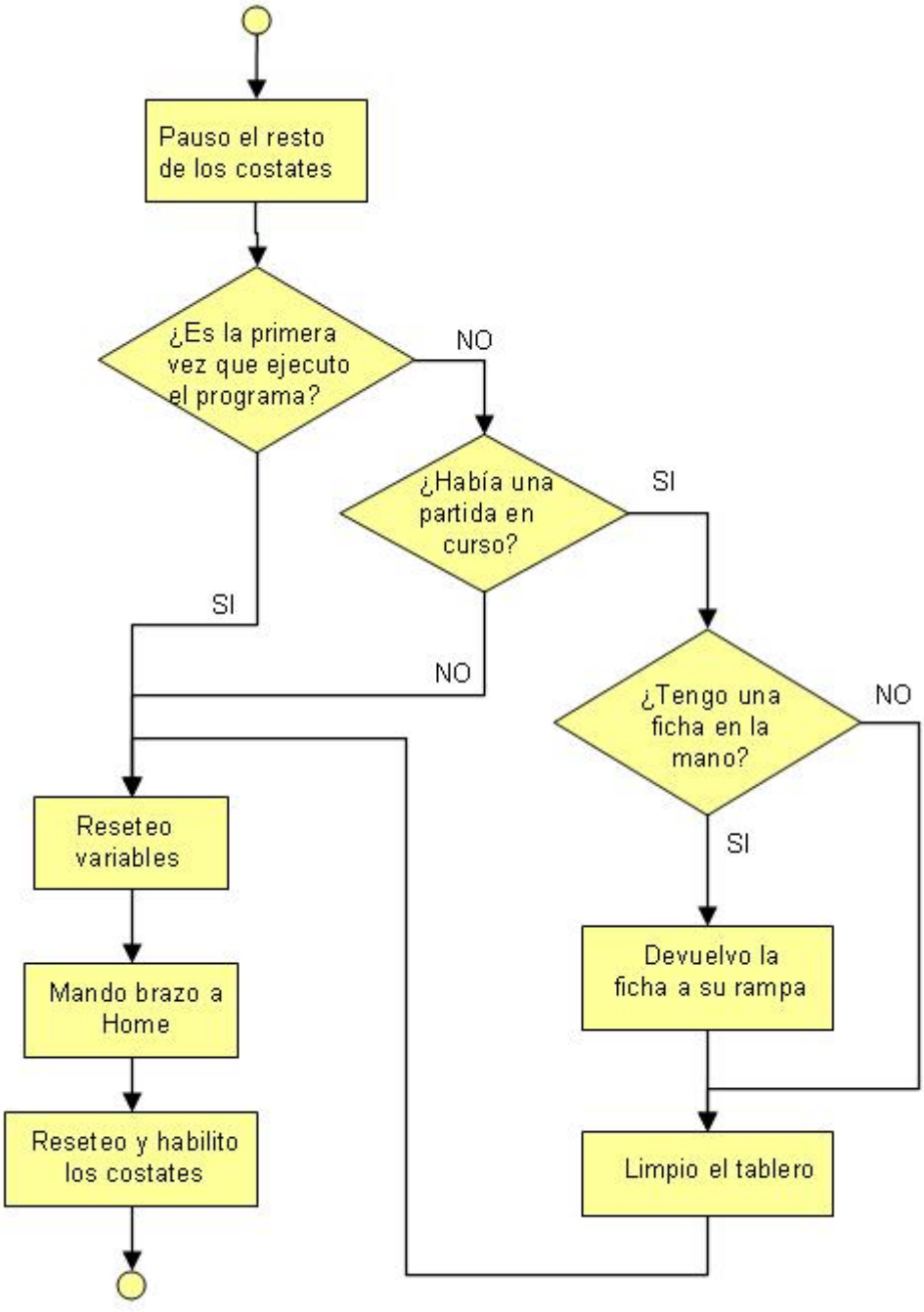


Figura 7.7: Diagrama de flujo del *costate* `inicializacion`.

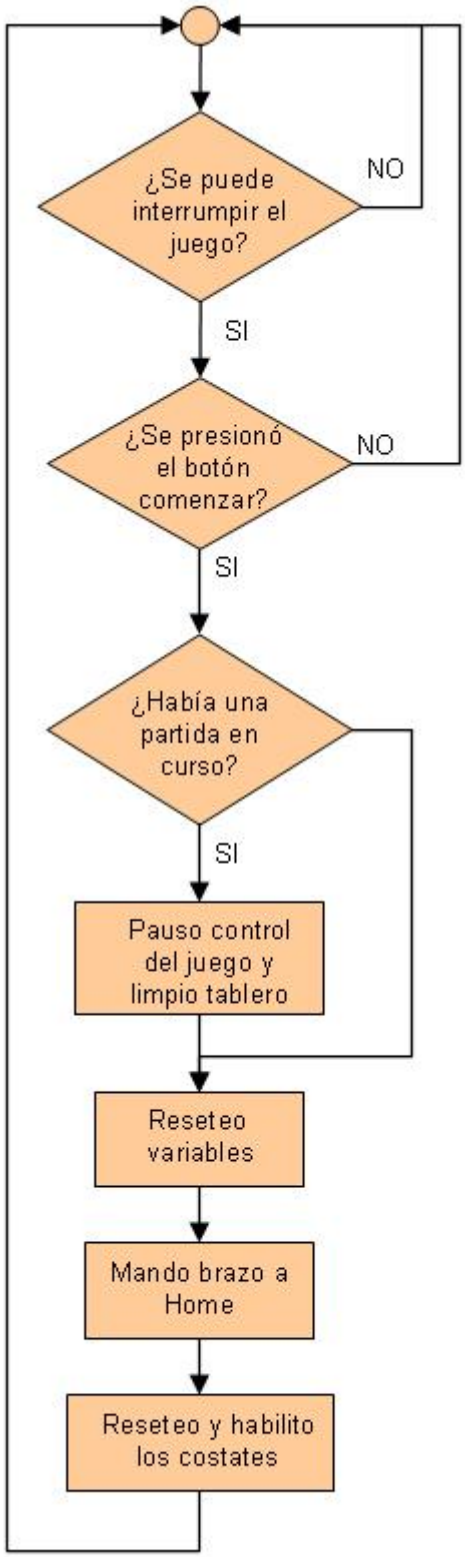


Figura 7.8: Diagrama de flujo del *costate* `checkBotonComenzar`.

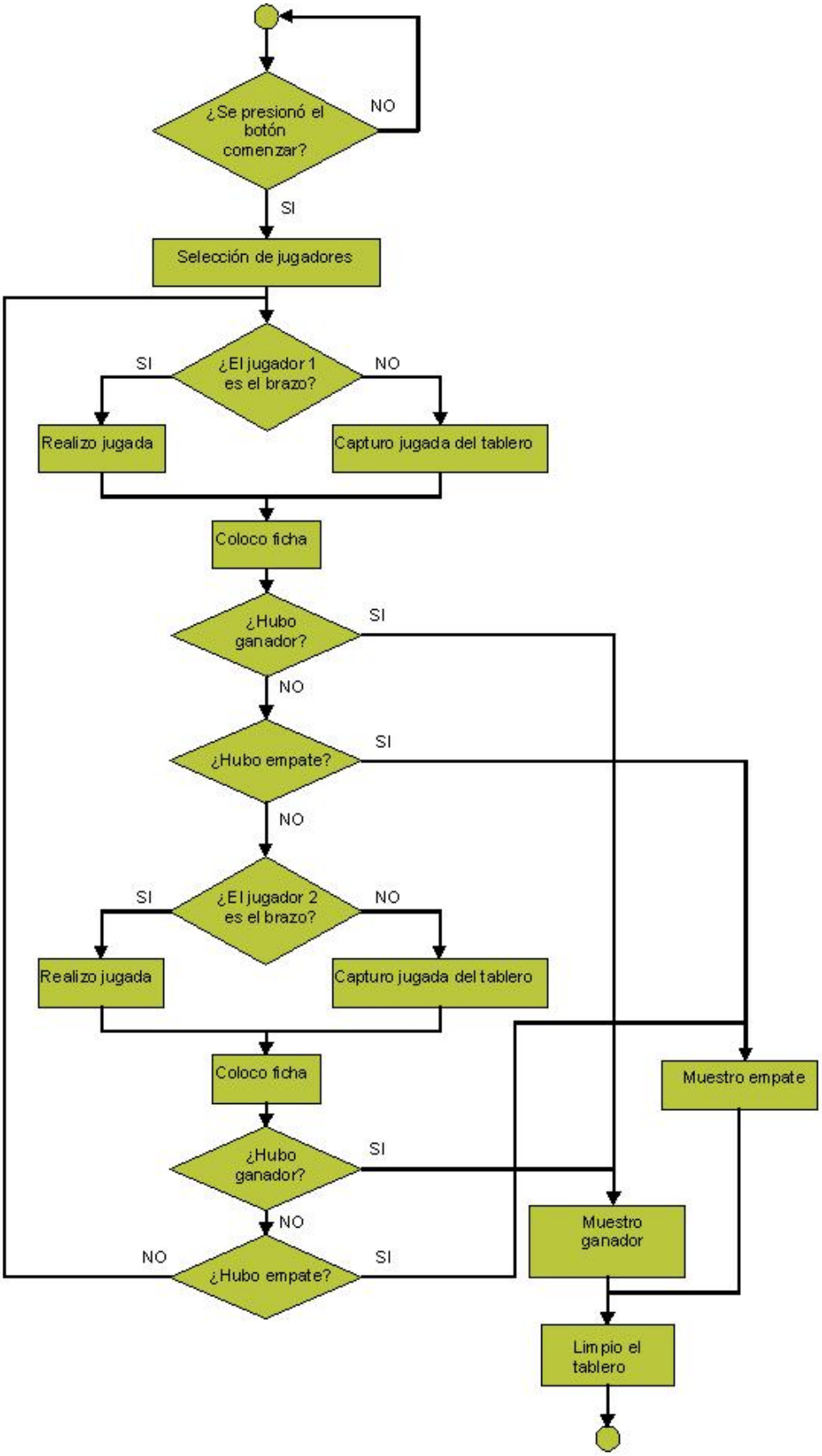


Figura 7.9: Diagrama de flujo del *costate* `controlJuego`.

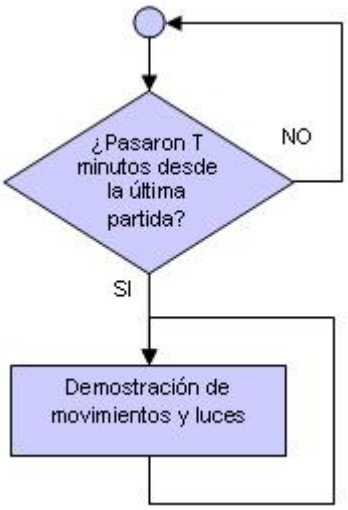


Figura 7.10: Diagrama de flujo del *costate* `idle`.

Capítulo 8

Prototipo y caracterización del Sistema

Es este capítulo se describe brevemente el proceso de interconexión y prueba de todas las partes involucradas en el sistema, así como algunos de los problemas que surgieron y los resultados obtenidos. A su vez, se presentan las características más relevantes del comportamiento del sistema terminado.

8.1. Puesta en marcha

El proceso de interconexión de los elementos estudiados anteriormente se dio en etapas bien definidas, cada una destinada a lograr integrar un elemento particular a un bloque ya validado.

En una primera instancia, se conectaron y probaron los motores del brazo con sus placas de potencia. Para ello, se manejaron las señales de control de las placas con un generador de señal de manera de probar cada articulación, su correcta conexión, el movimiento brazo y de la mano. En esta instancia se ajustó el valor de **MAG** para el motor de continua.

Figura 8.1: Generador de señales.

Una vez verificado el correcto funcionamiento de las placas de control en conjunto con los motores, se sustituyó al generador de señales por el microprocesador Rabbit y se probaron las diferentes rutinas de control de motores. Esto permitió ajustar los tiempos en las rutinas para que las velocidades de los movimientos fueran las deseadas y visualizar el

movimiento de las articulaciones en su conjunto.

En esta etapa de pruebas surgieron varios problemas y detalles a ajustar. Uno de los problemas más importantes encontrados fue que los circuitos de los encoders no trabajaban acorde a lo esperado, provocando el comportamiento errático que fue comentado en el capítulo 4.

En esta etapa el grupo logró verificar que el comportamiento mecánico y eléctrico del brazo así como del control con el Rabbit era el esperado. También se pudo realizar un primer ajuste de las velocidades y movimientos de los motores.

El siguiente paso fue verificar el comportamiento de la placa de control de interfaz en conjunto con los pulsadores y las luces. Al igual que en las pruebas de los motores, para comandar las señales de la placa se utilizó primero el generador de señal y luego el Rabbit. El desarrollo de estas pruebas fue particularmente tranquilo, pues no surgió ningún problema técnico de interés.

Con estas pruebas se pudo verificar el adecuado comportamiento de la placa de control de interfaz, de los botones, las luces y de sus rutinas de control.

Una vez instalado y probado por separado cada componente hardware del sistema, se probó el comportamiento de todo el sistema, es decir, la interacción de las cuatro grandes partes que componen el juego: el dispositivo mecánico, el tablero de interfaz de usuario, el software de juego y el hardware electrónico. En un principio no se contaba con el tablero de juego terminado, por lo que estas pruebas se realizaron moviendo al brazo a ciertas posiciones cualesquiera sobre la mesa donde se encontraba montando. Como era de esperar, surgieron algunos problemas de esta puesta en marcha que permitieron ir puliendo el software de juego y fueron solucionados sin inconvenientes.

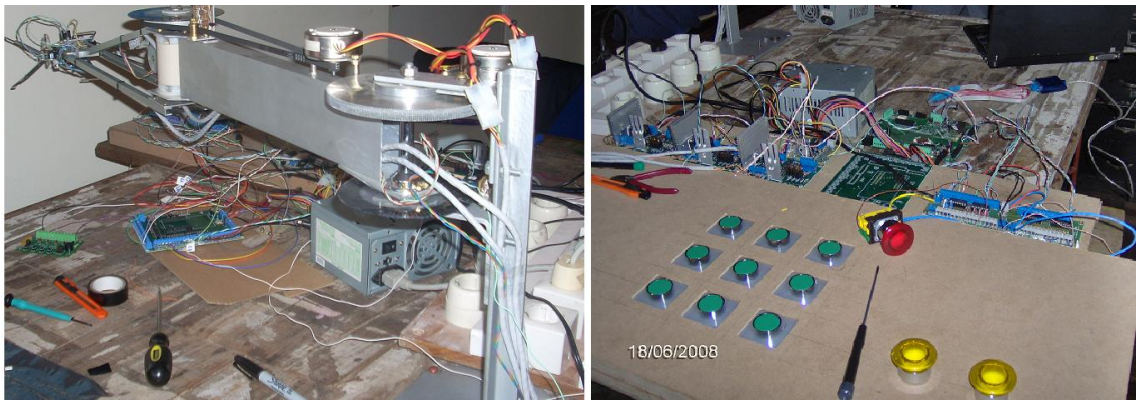


Figura 8.2: Fotos de la sala donde se realizaron las primeras pruebas del prototipo.

Una vez que el personal de Ciencia Viva terminó de construir el mueble, se trasladó el brazo y el tablero al Planetario para instalarlo en el mueble y comenzar con los ajustes de posiciones y pruebas definitivas. Primero, se montaron todas las placas y fuentes en el mueble y se interconectaron. Todos estos elementos se atornillaron a una tabla de madera lo que permite tener mayor control y comodidad a la hora de realizar el cableado. También

se instalaron algunos tomacorrientes manejados con un interruptor unipolar para alimentar las fuentes. En la figuras 8.3 y 8.4 se muestra este montaje en el interior del mueble.

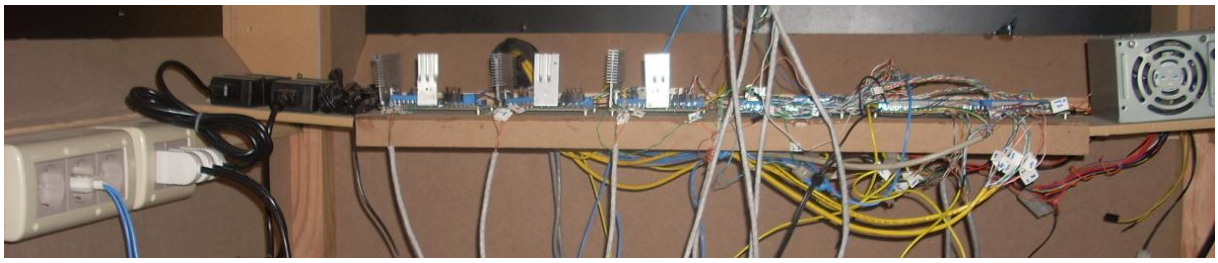


Figura 8.3: Interior del mueble de juego.

Las fuentes de alimentación utilizadas son una fuente de computadora de $300W$ que dispone de las tensiones de interés ($3{,}3V$, $12V$ y $5V$) y dos fuentes de $24V$ que pueden entregar hasta $1{,}7A$, obtenidas de unos terminales POS.

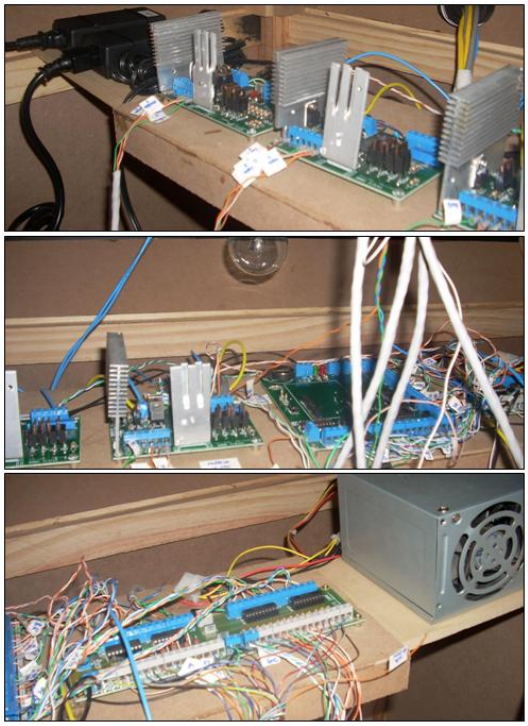


Figura 8.4: Placas de electrónica y fuentes de alimentación.

Al mismo tiempo se colocó y atornilló el dispositivo mecánico al mueble en la posición definitiva realizando los ajustes de nivel y altura necesarios. Con el brazo y los casilleros de juego colocados se calibró la posición de cada casillero utilizando programas auxiliares desarrollados especialmente con ese fin[1].

[1] En el capítulo 7 se detallan las programas implementados

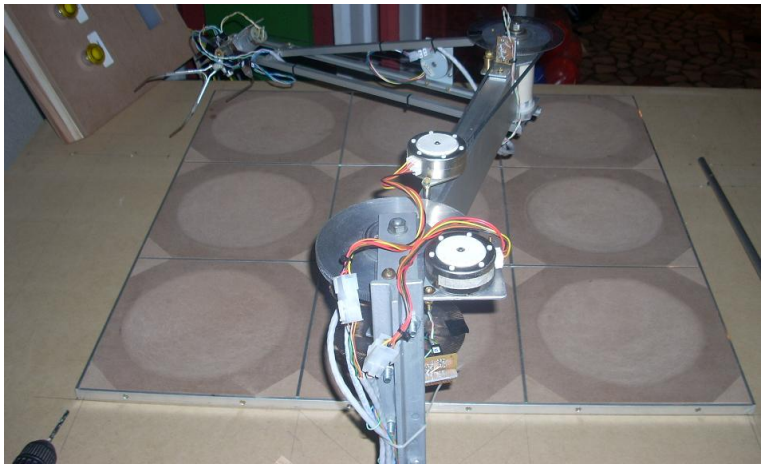


Figura 8.5: Mueble de juego con el brazo y casilleros colocados.

Una vez que estaban calibradas las posiciones de los casilleros, se determinaron la posiciones de donde se retiran y depositan las fichas de juego. Para poder calibrar estas cuatro últimas era necesario disponer de las canaletas colocadas.

La definición de la forma de las canaletas no fue tarea trivial. En un principio se probó con dos barras cilíndricas de aluminio paralelas colocadas con cierta pendiente para que las fichas deslizaran desde la parte superior donde son depositadas, a la parte inferior desde donde son retiradas. Esta canaleta presentó algunos inconvenientes por no tener la pendiente suficiente para que las pelotas deslizaran siempre y por no ser lo suficientemente amplias en la parte superior como para atajar la pelota. Existía un compromiso entre la altura y tamaño de la rampa con la posición de la mano al realizar estos movimientos, para evitar que se tocaran.

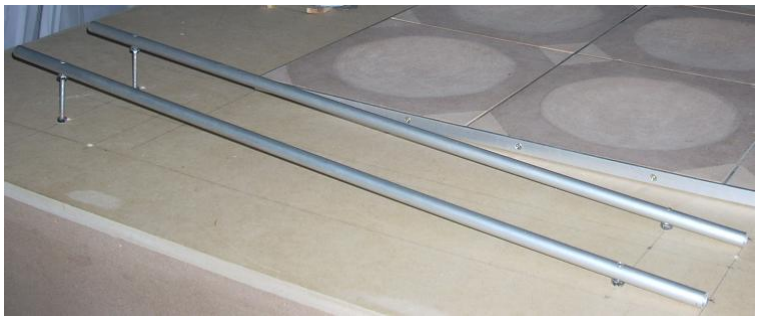


Figura 8.6: Primera versión de las canaletas con tubos de aluminio huecos.

Luego de varios ensayos se determinó que las rampas serían de perfiles de aluminio de forma rectangular dobladas y levantadas para dar una pendiente en forma de “L”. Las fichas se colocan en un costado de la “L” y ruedan a la parte inferior. Esto requirió colocar paredes de acrílico para detener el impulso de la pelota cuando es depositada.

Definidas y colocadas las canaletas, se calibraron las cuatro posiciones culminando de esta manera con el proceso de calibración.

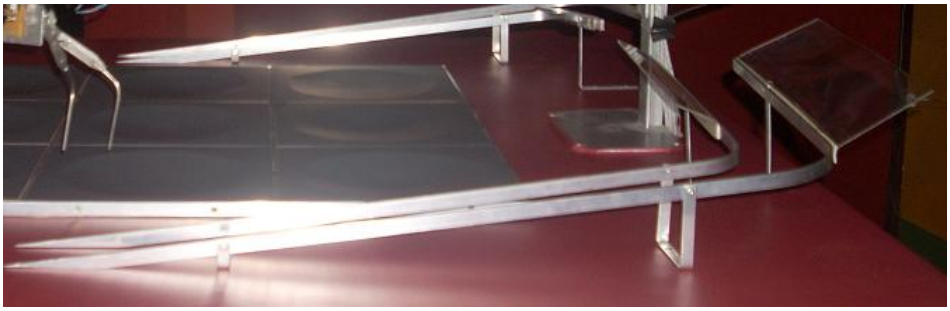


Figura 8.7: Rampas finales sin los topes de la parte inferior.

Por último y una vez pintado el mueble, se probó reiteradas veces el comportamiento del juego dentro del funcionamiento normal como en las condiciones de falla. En este proceso también se realizaron algunos retoques menores al software. Culminados estos ensayos de manera exitosa, se dio por terminado el juego. La figura xxx muestra una foto del juego terminado.

Figura 8.8: Mueble de juego.

8.2. Caracterización del sistema

Las etapas y comportamiento normal del brazo al comenzar y terminar un juego se realizan como se describe en la tabla a continuación.

Secuencias de un juego de Tatetí	
Acción del usuario	Reacción del sistema
Encender el juego con la llave que se encuentra dentro del mueble	El dispositivo mecánico se coloca en la posición *home* y aguarda que se presione el botón **COMENZAR**.
Se presiona el botón **COMENZAR**	Las tres luces asociadas con el jugador 1 (luz del cartel de jugador 1 y luces de los botones de selección de jugador) tintinean aguardando que se seleccione al mismo.
Se presiona uno de los dos botones de selección de jugador (Brazo o Persona)	Queda encendida únicamente la luz del botón seleccionado. Comienzan a tintinear las luces correspondientes al jugador 2, aguardado que se seleccione al mismo.
Se presiona uno de los dos botones de selección de jugador	Queda encendida únicamente la luz del botón seleccionado. Tintinea la luz del botón del jugador 1 que había quedado encendida indicando que es su turno. Si el jugador 1 corresponde al Brazo, inmediatamente realiza su jugada encendiendo la luz en el tablerito del casillero a donde moverá su ficha y realiza el movimiento. De lo contrario, queda a la espera de una jugada.
Se presiona el pulsador de un casillero cualquiera	Se enciende la luz correspondiente al casillero seleccionado y el dispositivo mecánico coloca la ficha. Cuando termina el movimiento deja encendida la luz del jugador que acaba de jugar y pasa a tintinear la luz del botón del otro jugador indicando su turno.
Termina el juego	
Se determina que es un empate	Las luces de los dos carteles de **JUGADOR 1** y **JUGADOR 2** tintinean al mismo tiempo.
Gana uno de los jugadores	Las tres luces correspondientes al jugador se prenden y apagan de forma circular.
En todos los casos, luego de un tiempo de indicar el resultado de la partida, se comienza a retirar las fichas del tablero y depositarlas en la rampa correspondiente. Se retiran en orden comenzando por el casillero inferior derecho y a medida que las deposita apaga la luz del casillero. Durante todo este proceso se encuentra la luz de espera encendida.	

8.2.1. Características del comportamiento general del sistema

- Cuando se dice que el brazo coloca una ficha, implica que retira la ficha de la rampa correspondiente al jugador del turno, la coloca en el casillero seleccionado y vuelve a la posición de *home*. Al retirar una ficha del tablero, realiza el procedimiento inverso: retira la ficha del casillero, la deposita en la parte superior de la rampa correspondiente y vuelve a la posición de *home*.

- *Estado de Espera:* Cada vez que el brazo se encuentre en movimiento, se encenderá el

cartel de **ESPERE...** hasta que el mismo deje de moverse. En estos períodos de tiempo, el juego hará caso omiso a que se presione cualquier botón del tablero.

- *Culminación de una partida:* Si en cualquier momento del transcurso de una partida y estando el cartel de **ESPERE...** apagado, se presiona el botón **COMENZAR**, se culminará la partida y el dispositivo mecánico pasará inmediatamente a retirar las fichas del tablero para comenzar un nuevo juego.

- *Tiempo "Idle":* Luego de culminada una partida y retiradas las fichas del tablero, el sistema queda a la espera de la indicación de un nuevo juego. Si luego de cierto tiempo no se comienza una nueva partida, sistema realiza una serie de movimientos y encendido de luces hasta que se presione el botón **COMENZAR**.

- *Corte de energía:* El sistema, luego que se restablece de un corte de energía, evalúa la situación en la que se encontraba y retira todas las fichas del tablero. No continúa con el juego anterior.

- *Condición de arranque particular:* Cuando el programa que contiene el Rabbit es cargado por primera vez desde una PC o cuando por algún motivo, se levantaron las fichas del tablero y se modificó la posición del brazo manualmente estando el sistema apagado, se debe indicar al microprocesador de esta situación. Para ello, cuando se enciende el sistema y durante el parpadeo inicial de las luces del tablero, se debe mantener presionados el botón **COMENZAR** y el pulsador de la posición **CERO** ubicado en el extremo inferior derecho.

- *Culminación de tiempos de espera:* Para poder determinar si se abandonó la partida, el sistema contabiliza el tiempo transcurrido entre cada etapa del juego y, si el tiempo de espera por una jugada supera un tiempo dado[2], inmediatamente culmina la partida retirando todas las fichas del tablero.

8.2.2. Tiempos característicos del sistema

La tabla siguiente detalla alguno de los tiempos más importantes del sistema que se deben tener en cuenta a la hora de jugar.

Descripción	Tiempo
Tiempo de espera para seleccionar un jugador una vez que comienzan a tintinear las luces, configurado por software	$30s$
Tiempo de espera para que se realice una jugada, configurado por software	$30s$
Tiempo de espera para entrar en estado "Idle", configurado por software	$5min$
Tiempo promedio en colocar una ficha en un casillero y volver a *home*	$20s$
Tiempo promedio en colocar una ficha en un casillero	$16s$
Tiempo promedio en retirar una ficha del tablero y volver a *home*	$21s$
Tiempo promedio en retirar una ficha del tablero hasta colocarla en la rampa	$14s$
Tiempo total en limpiar todo el tablero (nueve fichas colocadas)	$3:20min$

Los tiempos de posicionamiento se calcularon como el promedio de los tiempos de cada una de las posiciones del tablero.

[2]Este tiempo es ajustable en el programa

Capítulo 9

Conclusiones

9.1. Conclusiones generales

Se logró diseñar y construir exitosamente un sistema electromecánico capaz de jugar partidas de tatetí y cuyo comportamiento, aspecto e interfaz con el usuario cumplieran con las especificaciones del cliente[1]. A saber:

- El comportamiento del brazo se asemeja al de un brazo humano; si bien sus movimientos se dan dentro de un plano, su estructura y el comportamiento de sus articulaciones es muy similar al del brazo de una persona.
- Los componentes mecánicos de la estructura del brazo son completamente visibles al usuario y además provienen de elementos de uso común.
- El dispositivo es de fácil mantenimiento y desarme modular. Los motores y la mayoría de los repuestos son fáciles de conseguir en plaza. A su vez, el grupo brindó a Ciencia Viva documentación sobre cómo mantener y reparar cada uno de los componentes mecánicos del brazo, así como de toda la electrónica del juego.
- Se cumplieron con los mínimos exigidos en las dimensiones del brazo, fichas y tablero de juego. En particular, el brazo mide $90cm$ de largo, las fichas $14cm$ de diámetro y cada casillero del tablero $26cm$ de lado.
- La interfaz con el usuario además de estar hecha con componentes robustos, tiene un manejo sencillo, amigable e interactivo. El sistema responde de forma inmediata a los estímulos externos y guía al usuario a lo largo del juego.
- El sistema tiene una correcta reacción frente a fallas en la alimentación eléctrica, minimizando la necesidad de que personal de Ciencia Viva intervenga en estos casos.
- El costo en efectivo de este proyecto para Ciencia Viva fue menor a los $U\$S1000$ liberados en el presupuesto inicial.

[1] Estas especificaciones fueron mencionadas en el capítulo 1.

A pesar de ello, hubo algunos requerimientos que no pudieron ser cumplidos. En particular:

- No se cumplió con el plazo de entrega a Ciencia Viva especificado en el Plan de Proyecto (finales del mes de mayo de este año). Sin embargo, se cumplió con el plazo establecido por la Facultad de Ingeniería de la Universidad de la República.

- No se pudo cumplir con la restricción impuesta de 8 segundos sobre el tiempo medio de posicionamiento de fichas.

  Hoy en día este tiempo ronda los 15 segundos y el grupo está trabajando para poder reducirlo.

- No se pudo verificar que el juego pudiera funcionar en forma continua durante 8 horas por día y 6 días a la semana. Tampoco se pudo comprobar si el comportamiento de la interfaz con el usuario era adecuada y aceptada por los niños, usuarios finales de la experiencia interactiva.

  Como la culminación del proyecto se atrasó un poco respecto a lo planificado, no se pudieron realizar pruebas intensivas sobre el dispositivo que permitieran verificar estas características. El grupo continúa trabajando en ello para poder verificar estos requerimientos en el corto plazo.

9.1.1. Sobre la mecánica

Se logró un diseño mecánico que responde a las especificaciones y requerimientos del cliente así como a las necesidades de esta aplicación.

Si bien el grupo no contaba con conocimiento o experiencia previa en el área, gracias a las horas de estudio y a ayuda externa, se pudo implementar el diseño sin mayores problemas.

A pesar de que el mecanismo está formado en su mayoría por piezas de fácil adquisición en el mercado, los engranajes de las articulaciones del hombro y codo tuvieron que ser hechos a medida. Como su forma y tamaño no son estándar sino que se construyeron especialmente para esta aplicación, encontrar a alguien que los construyera fue difícil. Afortunadamente, luego de consultar en varias tornerías, se dio con una logró hacer el trabajo.

9.1.2. Sobre el diseño del hardware

Se lograron diseñar, construir y poner en funcionamiento todas las placas de electrónica necesarias para el control de los motores, la interfaz con el usuario y el control del sistema.

En general, no se presentaron grandes inconvenientes en el proceso de diseño, construcción y pruebas de estas placas. Esto se pudo lograr gracias a que el grupo fue muy cauteloso a la hora de decidir qué componentes utilizar y se basó en diseños conocidos.

9.1.3. Sobre la interfaz de usuario

Se pudo implementar una interfaz con el usuario que es robusta, sencilla y amigable.

Tras varias idas y vueltas en el diseño, se logró llegar a una versión que fue aceptada por el cliente. Se utilizaron componentes adecuados para la aplicación que garantizan la durabilidad del mismo.

9.1.4. Sobre el software

Se logró implementar un software que pudiera controlar todos los dispositivos hardware del sistema (luces, botones y motores) y dotara al brazo con la inteligencia suficiente para jugar una partida de tatetí contra una persona.

El programa implementado permite configurar de manera sencilla varias de las características del manejo de la interfaz con el usuario y del control de los motores. De existir la necesidad, se pueden ajustar estos parámetros para adaptar el comportamiento a los deseos del cliente.

Si bien la versión actual del programa cumple con los requerimientos básicos del sistema, existe la posibilidad de realizarle algunas mejoras para lograr obtener una mejor performance en cuanto al posicionamiento y a los tiempos de manejo de fichas. El grupo continuará trabajando en estos detalles.

9.1.5. Sobre el prototipo

Se logró integrar exitosamente todos los componentes hardware y software del juego.

Tal como fue mencionado, luego de diseñar e implementar cada una de las partes que integran el juego, se verificaron por separado. Hecho esto, se fueron integrando los componentes progresivamente. Gracias a esta política, el grupo pudo mantener bajo control todos los inconvenientes que surgieron y logró integrar todas las partes del sistema en poco tiempo y sin tener que realizar modificaciones de importancia.

9.2. Conclusiones sobre el desarrollo del Proyecto

El proyecto TATETI no solo fue el primer acercamiento de los integrantes del grupo a la robótica, sino que fue el primer proyecto de ingeniería que dirigieron y gestionaron.

Como ya fue mencionado, el proyecto involucra diversas disciplinas de la Ingeniería. Esto implicó que cada uno de los integrantes del grupo no solo tuviera que utilizar todas las herramientas adquiridas durante la carrera, sino que necesitara investigar y estudiar más a fondo temas de mecánica, teoría de control, electrónica y programación.

Además de estos aspectos técnicos, el grupo pudo adquirir importantes herramientas de trabajo como son el trato con clientes y proveedores, confeccionar presupuestos, administrar gastos, así como gestionar y manejar fechas y plazos de un proyecto.

También se aprendió a trabajar en grupo en un emprendimiento de mayor porte y duración. Se tuvo que dividir y coordinar esfuerzos para poder cumplir con los objetivos propuestos en tiempo y forma, sorteando tanto las dificultades técnicas como personales que pudieran surgir en el trayecto, tal como sucede en la vida profesional de un Ingeniero.

Lograr cumplir con el objetivo de este proyecto fue un gran desafío para el grupo, el cual tuvo que estudiar y trabajar arduamente para lograrlo. En lo personal, esto generó una gran emoción y satisfacción con la labor llevada a cabo.

Apéndice A

Evaluación del Plan de Proyecto

Previo al diseño y construcción del sistema, el grupo realizó un Plan de Proyecto donde se establecieron claramente los objetivos, restricciones y supuestos. A su vez, se realizó una planificación inicial con el objetivo de organizar mejor el trabajo tanto en tiempo como en tareas.

También se evaluaron los riesgos a los que el grupo se podía enfrentar y se establecieron criterios de éxito para evaluar el proyecto. En una etapa posterior, se realizó la WBS donde se dividió el objetivo principal en sub-objetivos bien definidos.

Finalmente, se realizó un estudio de costos para evaluar si el presupuesto con el que se contaba era aceptable para la realización del mismo.

A continuación se resume el Plan de Proyecto y se realiza una breve evaluación de algunos puntos que el grupo considera de interés.

A.1. Definición de objetivos específicos y WBS

El objetivo del proyecto es construir una experiencia interactiva para el Museo Participativo Ciencia Viva que funciona en la planta inferior del Planetario Municipal de Montevideo. Se trata de un juego de Tatetí donde las fichas serán movidas por un brazo robótico y la interfaz con el usuario se hará por medio de un tablero auxiliar.

Se dividió el objetivo final, en los objetivos específicos que se enumeran a continuación. Se identificó a cada uno de ellos con un nombre corto, el cual se conservó lo largo del proyecto. Cada uno de estos objetivos se subdividió en tareas específicas que se fueron realizando a lo largo del proyecto. No se entrará en el detalle de las mismas debido a que no aportará datos significativos al lector.

1. **Objetivo *Definición***
   El objetivo *Definición* es el de definir específicamente cómo será el funcionamiento y aspecto físico del juego a construir, en base a las especificaciones y lineamientos acordados con el cliente.

2. **Objetivo *Brazo***
   El objetivo *Brazo* es el de construir y controlar un brazo robótico de manera que sea capaz de realizar movimientos sencillos y pueda tomar y colocar una ficha en posiciones que se le indiquen.

3. **Objetivo *Interfaz***
   El objetivo *Interfaz* es el de definir y construir en su totalidad el entorno de juego y los elementos que serán utilizados para que el jugador interactúe con el dispositivo.

4. **Objetivo *Juego***
   El objetivo *Juego* es el de implementar el software que controle el juego en su totalidad.

5. **Objetivo *Prototipo***
   El objetivo *Prototipo* es el de integrar todas las partes construidas hasta ahora en un dispositivo prototipo para poder ser puesto a prueba en planta.

6. **Objetivo *Versión Final***
   El objetivo *Versión Final* es el de culminar el proyecto, cumpliendo con las especificaciones dadas por el cliente y colmando las expectativas de los usuarios. Incluye además la realización de la documentación.

Cumplidos todos estos objetivos, se puede dar por terminado el proyecto de fin de carrera.

A.2. Criterios de éxito

Para definir los criterios de éxito del proyecto, se planteó realizar tres tipos de pruebas funcionales. Primero se comprobaría el buen funcionamiento mecánico y lógico del juego, luego se verificaría que la respuesta del juego sea la esperada frente a los escenarios típicos de falla y finalmente se verificaría que se cumplan las condiciones pedidas de interfaz con el usuario.

Mediante las pruebas de funcionamiento mecánico y lógico del juego se planteó comprobar:

- Que el brazo responda correctamente a todos los posibles comandos que el usuario ingrese, colocando las fichas en los lugares correctos y no interfiriendo con el resto de las fichas colocadas en el tablero.
- Que al encender el dispositivo el brazo vaya a una posición conocida y quede en estado de espera de una nueva partida.
- Que el juego sea capaz de definir mediante un timeout que el usuario abandonó el juego y retire todas las fichas del tablero y espere por una nueva partida.
- Comprobar que todos los movimientos de fichas cumplan con las restricciones de temporización, teniendo una duración de no más de 8 segundos.
- Bajo ninguna circunstancia normal de juego las fichas deben soltarse de la mano del brazo o quedar ubicadas en posiciones erróneas.

Mediante las pruebas de respuesta a escenarios típicos de falla se planteó:

- Verificar que luego de un corte de electricidad, el brazo retire todas las fichas de juego del tablero, quedando en espera de un nuevo juego sin importar en qué momento se presentó la falla. De esta manera se podrá garantizar que el juego quedará funcionando correctamente sin la asistencia de un funcionario de Ciencia Viva.

Finalmente, mediante las pruebas de interfaz con el usuario se planteó comprobar:

- Que el público habitual del Museo Participativo Ciencia Viva sea capaz de operar correctamente el juego sin la necesidad de apoyo de los guías. En particular, se verificará que el juego cumple con las expectativas de los niños. Es decir, que sean capaces de jugar correctamente sin tener que explicarles cómo y les resulte divertido e interesante.

A.2.1. Evaluación de los criterios de éxito

Luego de las pruebas realizadas en la etapa de Prototipo, se llega a que el proyecto cumple con todos los criterios de éxito mencionados anteriormente salvo por el último punto que aun no ha podido ser evaluado. Esto se debe a que para poder evaluar la respuesta del público es necesario que el juego se encuentre instalado, con el acrílico de protección colocado y con los carteles indicativos necesarios para poder operarlo sin ayuda externa.

A pesar de que el grupo considera importante realizar esta evaluación, las terminaciones mencionadas corresponden a tareas asociadas a Ciencia Viva. De todas maneras, como pasos a seguir, se realizará la prueba con niños previo a su instalación para todo público.

A.3. Análisis de riesgos

Se reconocieron las diferentes categorías de elementos del proyecto y en los cuales puede ocurrir algún evento que pueda arriesgar el éxito del emprendimiento. A continuación se describen brevemente y se comentan los resultados.

- Elementos vinculados con el aspecto técnico del proyecto.

  Dentro de esta categoría aparecen el no encontrar motores, correas, microprocesadores, drivers y el resto de los elementos de hardware que sean útiles para la construcción del brazo o que no puedan conseguirse en tiempo o a un costo razonable. También puede ser que se estropeen los que ya se tienen.

  Resultado: Se previó desde un principio disponer de varios motores y correas de repuesto para evitar grandes inconvenientes. Con respecto a los motores, se tenían dos de repuesto que no fue necesario utilizar. Pero con respecto a las correas, la utilizada para la articulación de la muñeca se quebró ya terminado el brazo. Para solucionar el problema, se sustituyó por una de igual forma pero más corta. Esto requirió correr el motor de lugar lo que no dio grandes problemas.

- Elementos vinculados con el diseño del juego, cumplimiento de los objetivos propuestos y satisfacción del usuario final.

  Son los riesgos de que existan errores en el diseño y luego de construida la parte el comportamiento no sea el esperado, que lo que se haya implementado no fuera lo que se pretendía o que las evaluaciones hechas por los usuarios finales no sean satisfactorias y se tenga que volver atrás en el proyecto.

  Resultado: No hubo inconvenientes por desconformidades del cliente con el dispositivo implementado. Esto se debió a que se mantuvo durante todo el proyecto un buen contacto con el cliente. Tampoco surgieron errores en el diseño debido a que

cada parte construida se estudiaba, diseñaba y probaba de forma independiente. Si embargo, debido a inconvenientes tanto del grupo como de Ciencia Viva, aun no se ha probado el juego con los usuarios finales por lo que no se puede concluir acerca de su reacción. Estas pruebas serán los pasos a seguir para culminar con el proceso de evaluación.

- Elementos vinculados con actores externos al grupo de proyecto.

  Los proveedores pueden no poder conseguir ciertas partes necesarias para la construcción del brazo y se tengan que traer del exterior, con los costos y demoras asociadas. Por otra parte, el cliente puede no poder cumplir con los acuerdos hechos con el grupo.

  Resultado: Alguno de estos inconvenientes sucedieron tanto por parte del cliente como por tener que traer materiales del exterior, lo que atrasó el proyecto con respecto a la fecha acordada inicialmente: Tener un prototipo terminado para finales del mes de Mayo. De todas maneras, se pudo culminar con el proyecto dentro del plazo establecido por la Facultad de Ingeniería de la Universidad de la república.

- Elementos vinculados con el grupo de proyecto.

  Aquí se ubicarían los problemas personales, enfermedad o necesidad de viajar de alguno o varios de los miembros del grupo.

  Resultado: El grupo no tuvo inconvenientes de este tipo por lo que estos riesgos no fueron un problema.

- Elementos vinculados con la mala decisión al momento de determinar los supuestos del proyecto.

  Se pueden haber tomado supuestos errados, que resulten en que el trabajo a realizar tenga que ser diferente, más arduo o más costoso, con los riesgos que ello conlleva.

  Resultado: Si bien los supuestos considerados no estaban errados, el trabajo resultó más arduo de lo que se esperaba implicando que algunas tareas llevaran más tiempo que el estipulado y atrasaran la fecha de finalización del proyecto.

A.4. Análisis de costos

Otra parte importante del proceso de estudio del proyecto fue la realización de un estudio de costos, para asegurar que el proyecto pudiera ser completado dentro de un presupuesto razonable.

Primero, el grupo intentó reconocer los diferentes rubros en los cuales pueden ubicarse cada uno de los costos de los materiales del proyecto, como ser la Electrónica o la Mecánica. Dentro de estos rubros y los de mano de obra y Varios, para contemplar imprevistos, se ubicarán cada uno de los elementos de costos que el grupo reconozca.

A continuación, se intentó estudiar y estimar la cantidad y el costo de todos los componentes que serían necesarios dentro de cada rubro para poder construir el dispositivo. Cuando fue posible, el costo asignado fue el que se pudo averiguar en el mercado. Todo esto aparece en la columna *Presupuesto* de la figura A.1.

De todos lo ítems mencionados se consideró que el costo de mano de obra sería nulo, que se podrían reciclar algunos componentes del depósito de Ciencia Viva y que el módulo

Rabbit era una donación. De esta manera, el costo en pesos que le significaría a Ciencia Viva este proyecto sería de $28877.
Como era esperado en etapas anteriores del análisis del proyecto, el estimativo superaba los $U\$S1000$ que Ciencia Viva dio como subsidio. Cabe aclarar que el precio del dólar a Agosto del 2007 era de aproximadamente $23,50.

Presupuesto disponible:		U$S 1000			
				Totales en $ (pesos)	
Imputación		Cantidad	Unidad	Costo unitario	Presupuesto
Costos					
Materiales					
Mecanica	Perfil de aluminio grueso	1,5	metro	300	450
	Perfil de aluminio fino	2	metro	150	300
	Láminas de aluminio	0,5	m^2	400	200
	Partes mecanicas de transmision [1]	1	un	800	800
	Ejes pesados	2	un	200	400
	Motores	6	un	300	1800
	Varios [2]	1	s/u	500	500
	Sub total 1				4450
Electronica	Componentes electrónicos [3]	1	un	700	700
	Componentes de potencia [4]	1	un	1000	1000
	Mecanismos de control [5]	2	un	200	400
	Fuente de alimentación	2	un	600	1200
	Integrados	1	un	1000	1000
	Drivers de motores	6	un	200	1200
	Cables de potencia	15	m	30	450
	Cables de control (baja tensión)	20	m	30	600
	Placas de electrónica	4	un	400	1600
	Microprocesador Rabbit	1	un	2400	2400
	Switches	10	un	60	600
	Varios	1	s/u	500	500
	Sub total 2				11650
Interfaz	Pulsadores y botoneras	20	un	75	1500
	Luces o leds	20	un	15	300
	Mueble de madera y acrílico	1	un	3500	3500
	Fichas	30	un	20	600
	Varios	1	un	500	500
	Sub total 3				6400
Otros	Zapatilla de tomas	1	un	200	200
	Fichas de adaptacion	4	un	100	400
	Cables de conexión de tension	3	un	100	300
	Varios	1	s/u	500	500
	Sub total 4				1400
Total Costos Materiales					23900
Mano de Obra	Leticia Cirlinas	500	horas	50	25000
	Andres Spaggiari	500	horas	50	25000
	Francisco Manfredi	500	horas	50	25000
	Viáticos totales [6]	864	viajes	15	12960
	Personal carpintería	20	horas	30	600
	Personal soldadura	20	horas	30	600
	Personal de pintura	20	horas	30	600
	Personal de cartelería	20	horas	30	600
	Otros Ciencia viva	20	horas	30	600
	Varios	1	s/u	500	500
Total Costos Mano de Obra					91460
Costos totales del Proyecto					115360

[1] Engranajes, ejes livianos, poleas
[2] Remaches, tornillos, tuercas, cemento, material extra.
[3] Incluye condensadores, resistencias, conectores, estaño, y otros
[4] Incluye relés, contactos, interruptores y otros
[5] Sensores de posición (potenciómetros, encoders)
[6] Cantidad de viajes que realicemos en total

Figura A.1: Estudio de costos para el proyecto.

A lo largo de todo el proyecto el grupo llevó un registro detallado de todas las compras realizadas. Sin considerar la carpintería realizada en Ciencia Viva, el grupo gastó $22700 en los componentes para el proyecto. Este costo queda por debajo tanto de lo estipulado como de lo disponible.

Apéndice B

Motores paso a paso

B.1. Introducción a motores paso a paso

El motor de paso a paso es un dispositivo electromecánico que convierte una serie de impulsos eléctricos en desplazamientos angulares discretos, lo que significa es que es capaz de avanzar una serie de grados (paso) dependiendo de sus entradas de control. El motor paso a paso se comporta de la misma manera que un convertidor digital-analógico y puede ser gobernado por impulsos procedentes de sistemas lógicos.

Existen tres tipos fundamentales de motores paso a paso: de reluctancia variable, de magnetización permanente, y el híbrido.

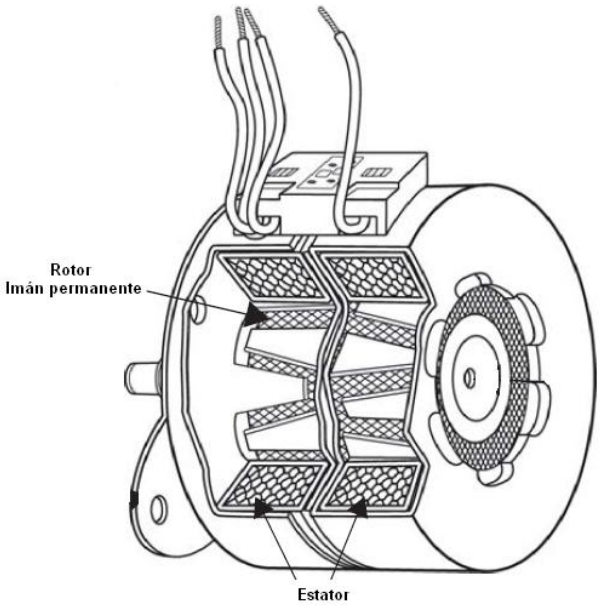


Figura B.1: Corte del motor mostrando los componentes internos.

El motor paso a paso está constituido, como la mayoría de motores eléctricos, esencialmente de dos partes:

- Una parte fija llamada "estator", construida a base de cavidades en las que van

depositadas las bobinas que excitadas convenientemente formarán los polos norte-sur de forma que se cree un campo magnético giratorio.

- Una parte móvil, llamada "rotor" construida bien con un imán permanente o bien por un inducido ferromagnético, con el mismo número de pares de polos que el contenido en una sección de la bobina del estator; este conjunto va montado sobre un eje soportado por dos cojinetes que le permiten girar libremente.

Si por el medio del control que sea (electrónico, informático, etc..), se consigue excitar el estator creando los polos N-S, y se hace variar dicha excitación de modo que el campo magnético formado efectúe un movimiento giratorio, la respuesta del rotor será seguir el movimiento de dicho campo, produciéndose de este modo el giro del motor.

Estos motores poseen la habilidad de poder quedar enclavados en una posición o bien totalmente libres. Si una o más de sus bobinas están alimentadas, el motor estará enclavado en la posición correspondiente y por el contrario quedará completamente libre si no circula corriente por ninguna de sus bobinas [27].

La figura B.2 muestra un ejemplo de su funcionamiento en cuatro pasos para un motor de cuatro bobinas.

Figura B.2: Ejemplo de funcionamiento de un motor de paso de cuatro bobinas.

B.2. Motores de paso NMB - Permanent Magnet

Los motores de paso utilizados para las articulaciones del hombro y codo, son marca NMB modelo **PM55L-048** [29]. Esto indica que son de la serie de motores *Permanent Magnet (PM)*, de $55mm$ de diámetro, tamaño *large* y de 48 pasos por vuelta. Esto último implica que los mismos giran $7,5^o$ por paso.

La tabla siguiente resume las especificaciones del modelo.

Reference Characteristics		
Motor Size	PM55L-048	
No. of Steps per Rotation	48 (7.5° / Step)	
Drive Method	2-2 PHASE	
Drive Circuit	UNIPOLAR CONST. VOLT.	BIPOLAR CHOPPER
Drive Voltage	24 [V]	24 [V]
Current / PHASE		600 [mA]
Coil Resistance / PHASE	30 [Ω]	6 [Ω]
Drive IC	SMDT - 002	UDN2916B-V
Magnet Material	Ferrite plastic magnet, Polar anisotropy ferrite sintered magnet, Nd-Fe-B bonded magnet	

Figura B.3: Datos técnicos del motor PM55L-048

Estos motores se pueden utilizar conectados de forma unipolar o bipolar. En esta aplicación se utilizaron de forma bipolar lo que implica que se conectaron cuatro de los seis cables disponibles para su control. La figura siguiente detalla este modo de conexión y los colores de los conductores mostrando a su vez cómo se deben comandar para lograr hacer girar al motor.

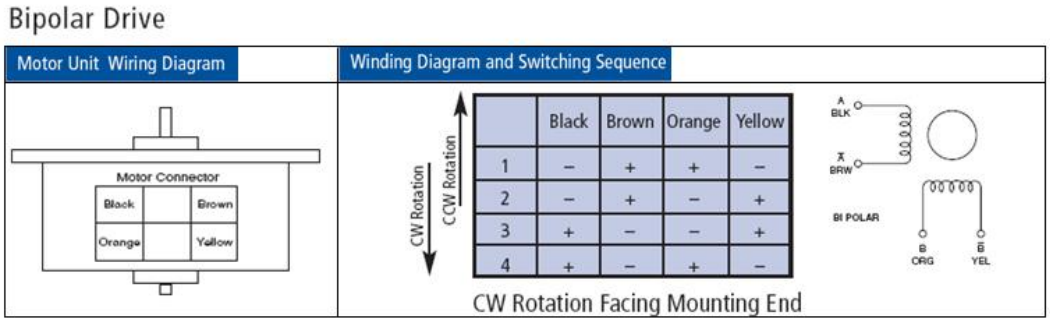


Figura B.4: Detalle de conexión de los motores PM

La curva *torque-velocidad* y las dimensiones de los motores se pueden encontrar en la hoja de datos del fabricante [29], junto con algunas especificaciones generales de este tipo de motores.

B.3. Motores de paso FDK

El motor de paso utilizado para la articulación de la muñeca es de marca FDK y modelo **SMB40-48** [28]. Esto indica que es de la serie SM (stepping motor), de $14,4mm$ de altura, $42mm$ de diámetro y de 48 pasos por vuelta al igual que los motores de NMB. La tensión de alimentación no debe exceder los $30VDC$.

También se utilizaron en modo bipolar como se detalla en la figura siguiente, donde los colores de los conductores se detallan en la figura siguiente.

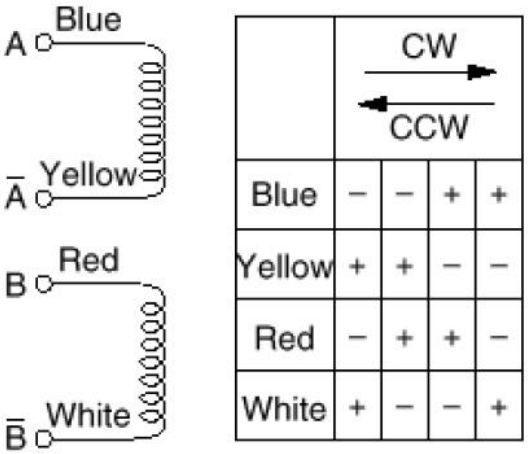


Figura B.5: Detalle de conexión del motor FDK en modo bipolar

Apéndice C

Esquemáticos y PCBs de las placas diseñadas

A continuación se presentan los esquemáticos y el layout definitivos de las placas diseñadas para el proyecto. Para todos los diseños se utilizó el software Eagle versión 4.11. Del layout se dan las capas de cobre de la cara inferior y superior (en caso de que corresponda) en un mismo PCB.

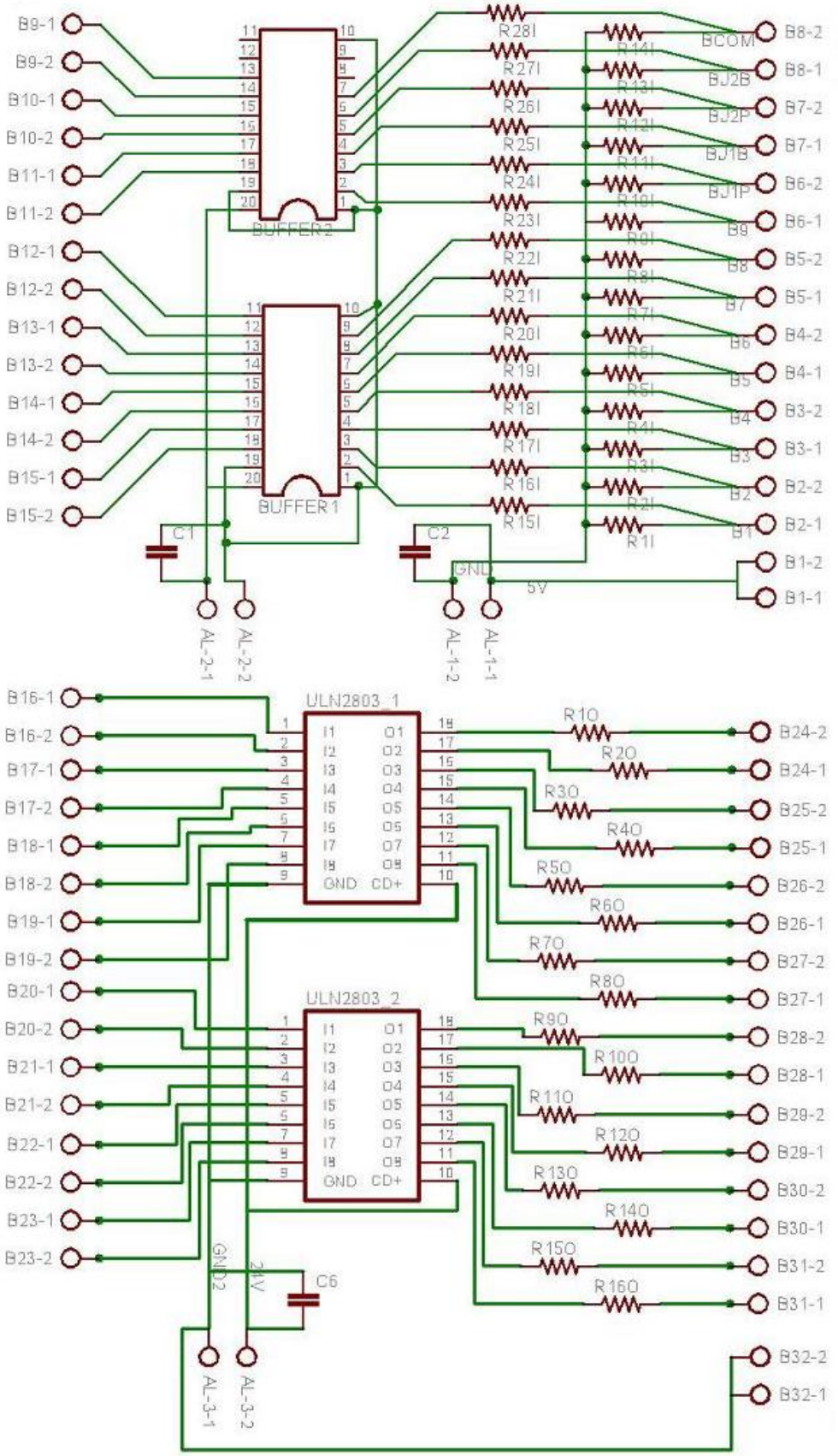


Figura C.1: Esquemático del circuito de la placa de interfaz de usuario.

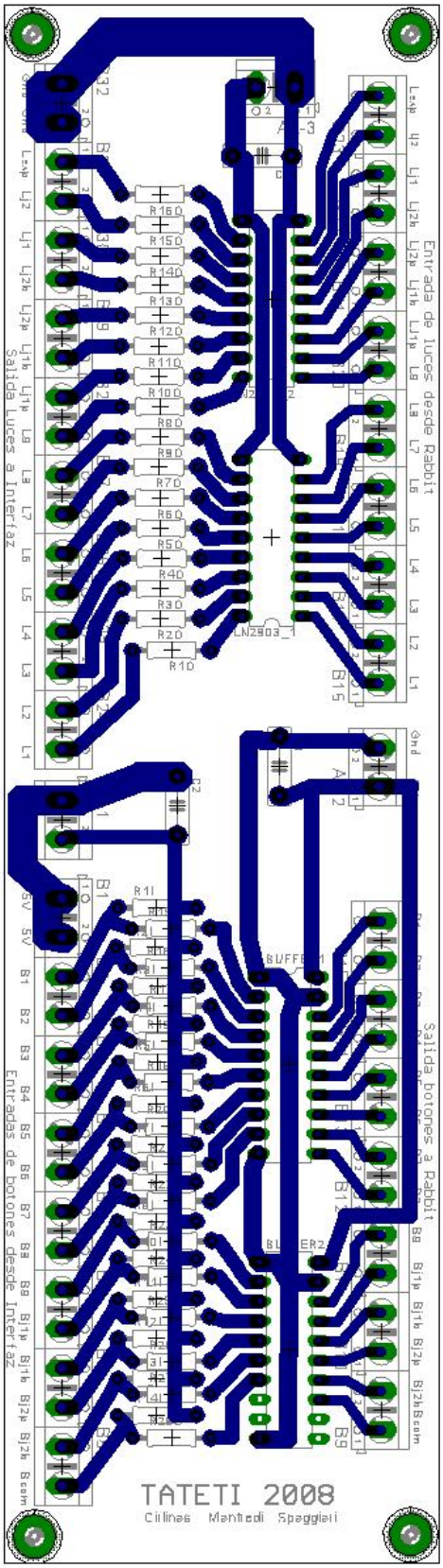


Figura C.2: PCB de la placa de control de interfaz diseñada en una sola capa.

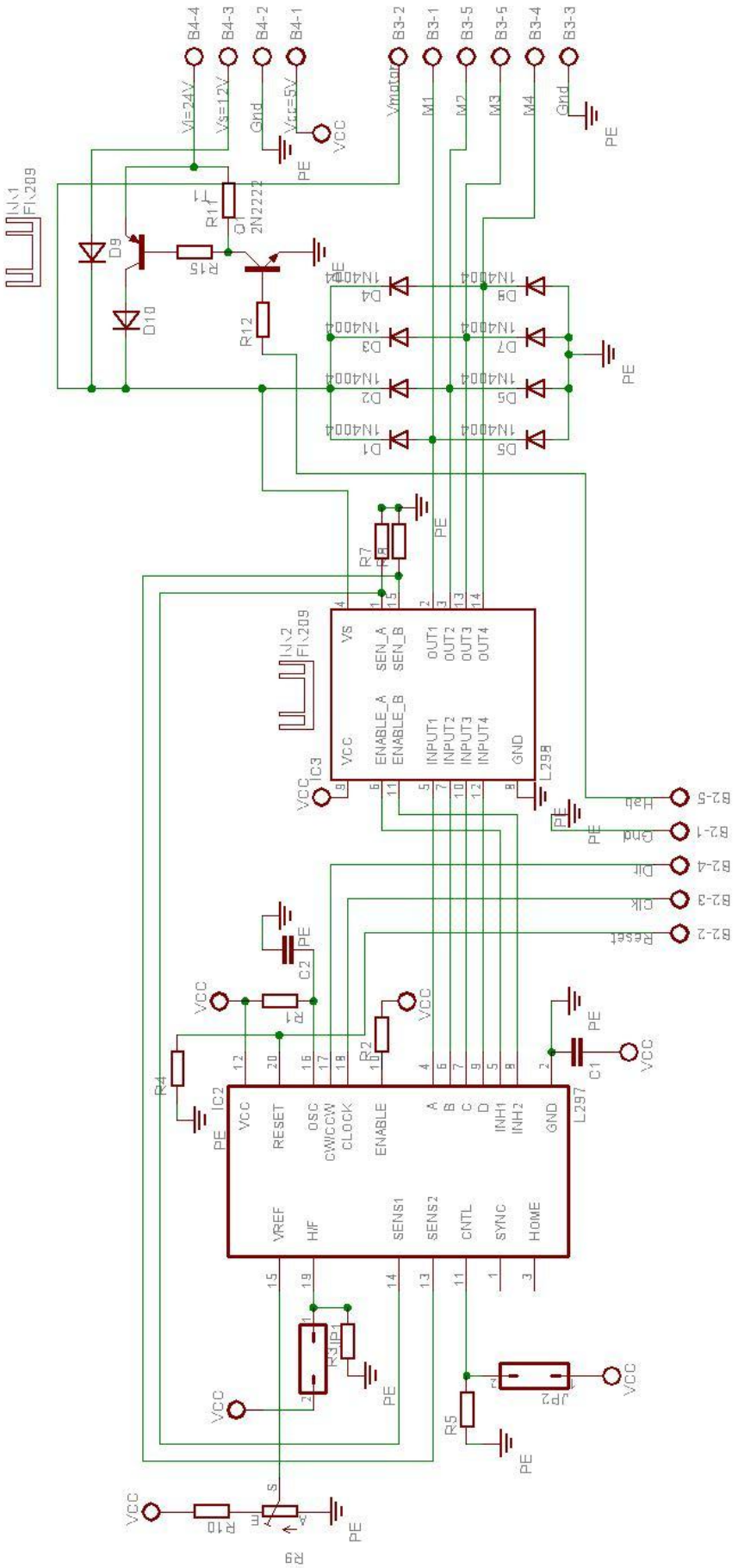


Figura C.3: Esquemático del circuito de potencia para el control de los motores de paso.

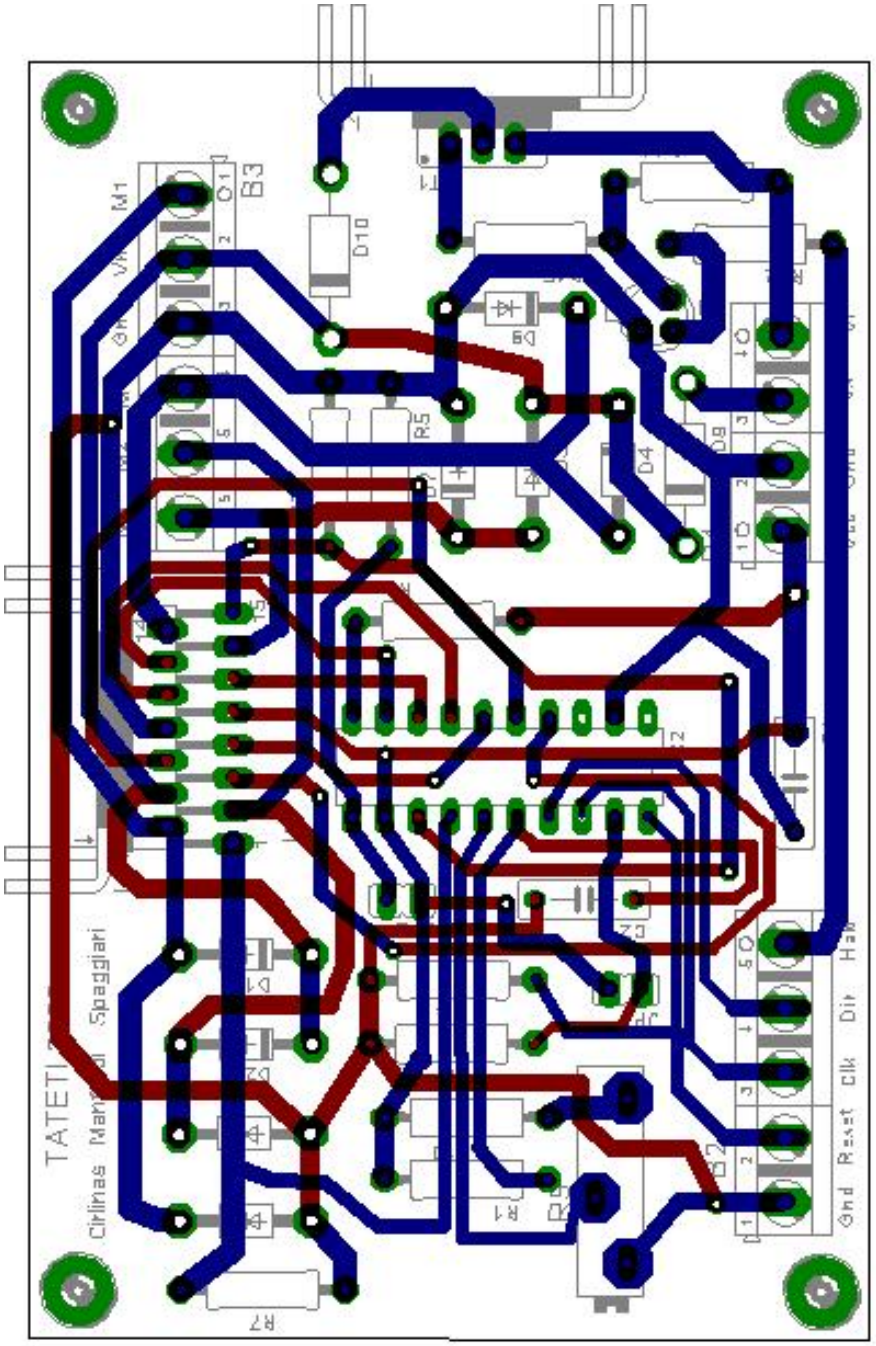


Figura C.4: Layout del circuito de potencia para en control de los motores de paso.

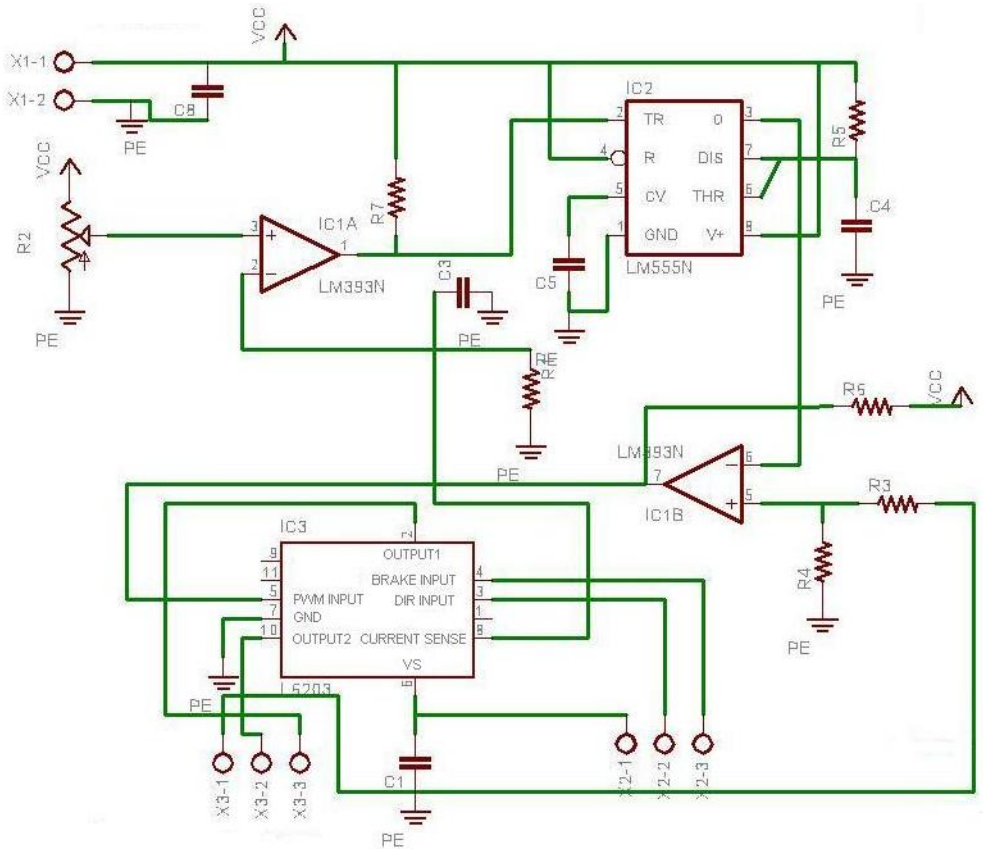


Figura C.5: Esquemático del circuito de potencia para en control de los motores de paso.

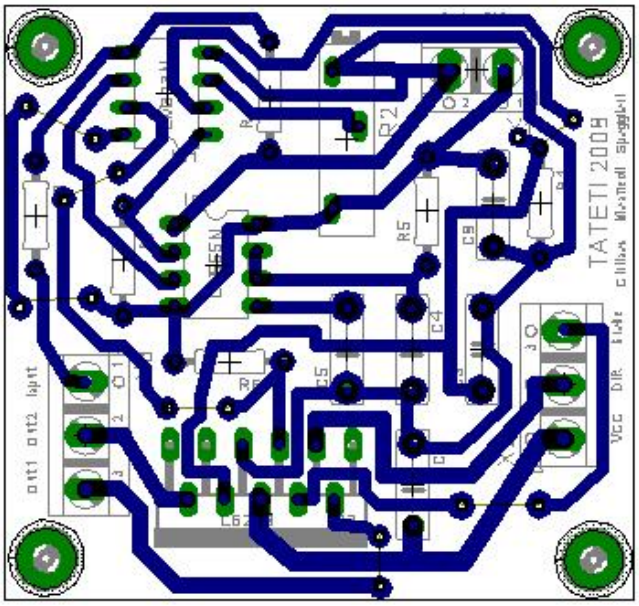


Figura C.6: Layout del circuito de potencia para en control de los motores de paso.

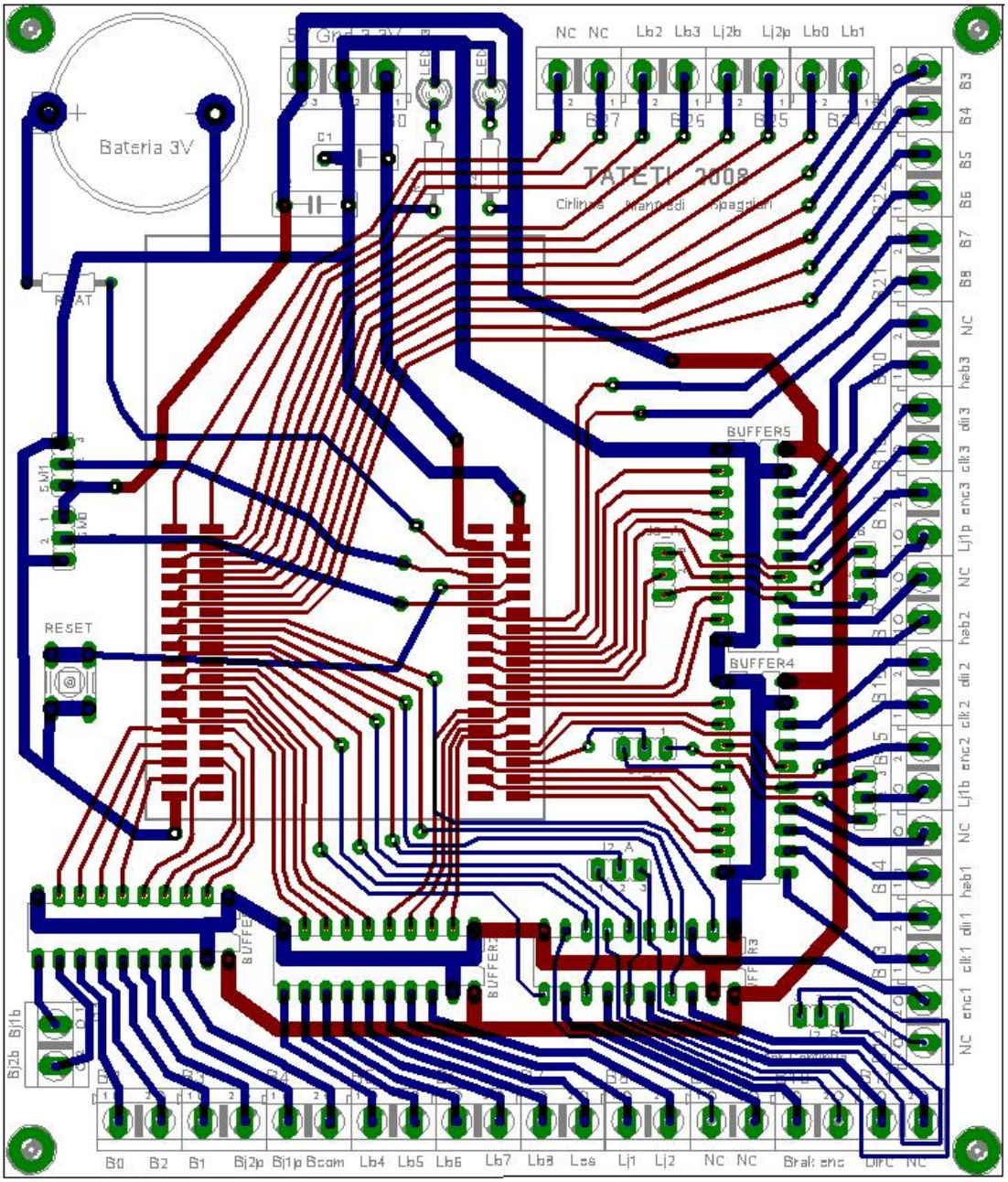


Figura C.7: Layout del circuito del microprocesador.

Apéndice D

Manual de Mantenimiento

D.1. Introducción

El juego TATETI es un equipamiento electro-mecánico, compuesto por un brazo robótico, un sistema de control electrónico y una interfaz. El mismo es capaz de jugar partidas de tatetí entre uno o dos usuarios o contra si mismo. Todo el sistema es alimentado por dos fuentes conectadas a la red eléctrica de 220V y 50Hz.

El equipamiento incluye:

Equipamiento			
Mecánico	Hardware Eléctrico	Mueble	Otros
Brazo Robótico	Control del Sistema Control de Motores Control Interfaz	Tablero de Interfaz Tablero de Juego	Fuentes de Alimentación Cableado del Sistema

D.2. Familiarización con el Sistema

El brazo cuenta con cuatro grados de libertad. Nos referiremos a sus partes usando los nombres correspondientes a las articulaciones del cuerpo humano para una mayor comprensión. Está compuesto por el soporte, brazo, antebrazo y mano, donde las articulaciones son las del hombro, codo, muñeca y pinzas. Este conjunto de articulaciones permite al brazo moverse en un plano horizontal hacia cierta posición y luego subir y bajar la mano al igual que abrir y cerrar las pinzas. En las figuras D.1 y D.2 se muestran las partes mencionadas.

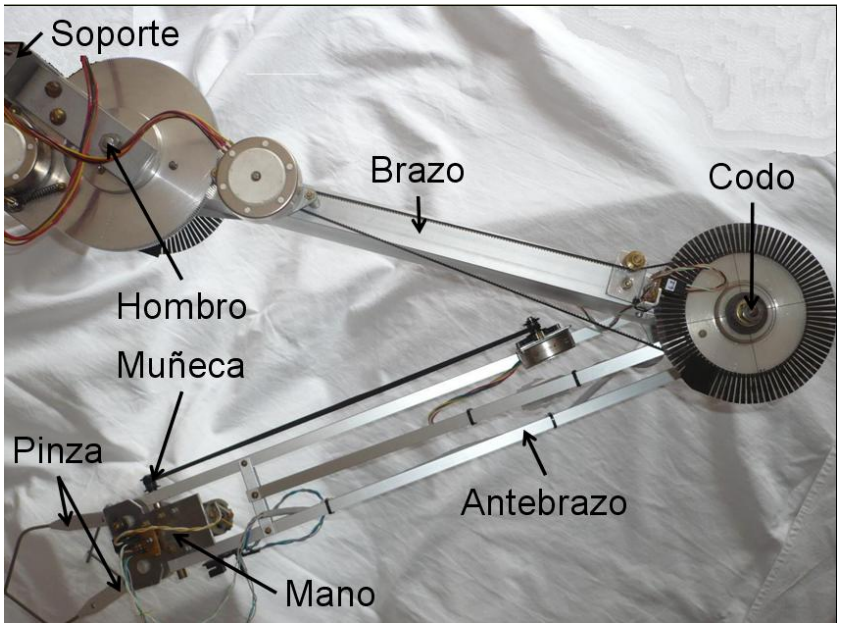


Figura D.1: Estructura y articulaciones del brazo.

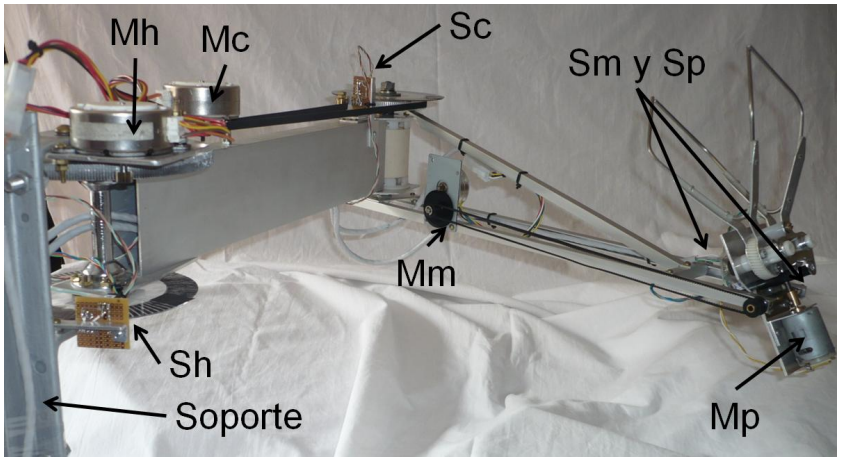


Figura D.2: Elementos de control y movimiento del brazo.

Referencia:

- **Mh**: motor que genera el movimiento del hombro.
- **Mc**: motor que genera el movimiento del codo.
- **Mm**: motor que genera el movimiento de la muñeca.
- **Mp**: motor que genera el movimiento de las pinzas.
- **Sh**: Sensor utilizado para el posicionamiento del hombro.
- **Sc**: Sensor utilizado para el posicionamiento del codo.
- **Sm**: Sensor utilizado para el posicionamiento del muñeca.
- **Sp**: Sensor utilizado para el posicionamiento del pinzas.

D.3. Mantenimiento

Es imprescindible mantener al brazo en buenas condiciones de limpieza para evitar deterioro de sus partes por acumulación de polvo y suciedad. A continuación se detallan los pasos a seguir para el mantenimiento de cada una de las partes.

- **General**
  Limpie periódicamente la parte exterior del brazo con un paño húmedo y detergente (evite dejar pelusa sobre el mismo); no utilice abrasivos ni solventes. Limpie y seque según sea necesario. Si no va ser utilizado durante un periodo largo de tiempo, almacenarlo en un lugar seco sin polvo, tapado y fuera del alcance directo de la luz solar. En caso de querer mover la mano, asegúrese que la misma se encuentra desconectada, nunca fuerce las articulaciones ni intente detener su movimiento. Evite todo tipo de golpes y malos tratos.

- **Limpieza de articulaciones y engranajes**
  El brazo cuenta con cuatro articulaciones que deben ser limpiadas periódicamente [1].

  1. *Pinzas*
     Los asientos de los engranajes son directamente sobre la chapa de aluminio y sobre la lámina de hierro cromada. Para que esta articulación funcione adecuadamente es necesario mantener limpios y engrasados estos puntos de apoyo, para esto utilice un paño humedecido con alcohol y un compresor de aire. Los engranajes pueden ser limpiados de igual forma y teniendo las mismas precauciones de limpieza (no es necesario engrasar por completo ambos engranajes que hacen juego con el sin fin, dado que este movimiento esta restringido a unos 90°)[2].

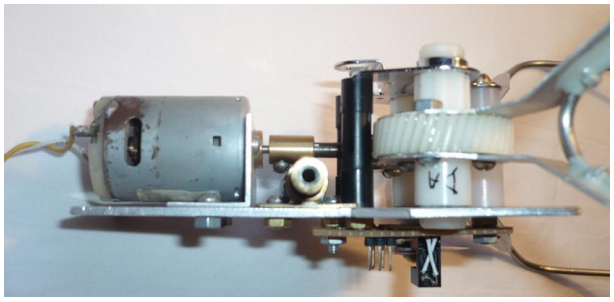


Figura D.3: Vista lateral de la mano.

[1] por mas información del periodo de limpieza ver capitulo: Especificaciones Generales
[2] Por más detalles respecto al lubricante a usar ver capítulo: Especificaciones Generales

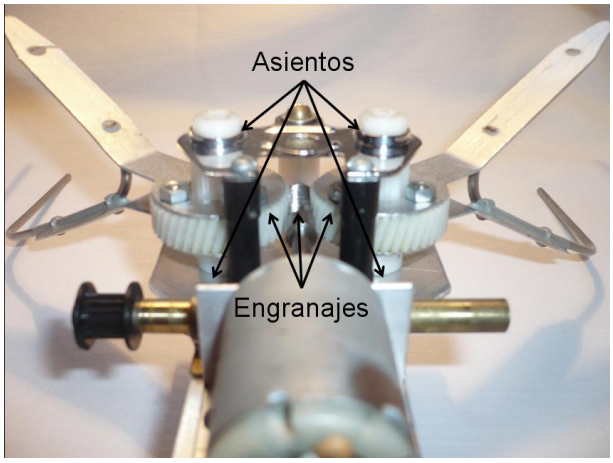


Figura D.4: Conexión motor-sinfin-engranajes-pinzas.

2. *Muñeca*
   Esta articulación descansa entre los orificios en los perfiles de aluminio y el tubo de bronce que funciona como eje de giro. Es importante evitar cualquier tipo de golpe sobre estos perfiles sino la articulación quedará forzada. El motor que mueve esta articulación se encuentra en la base del codo. Mantener estas articulaciones limpias y engrasadas.

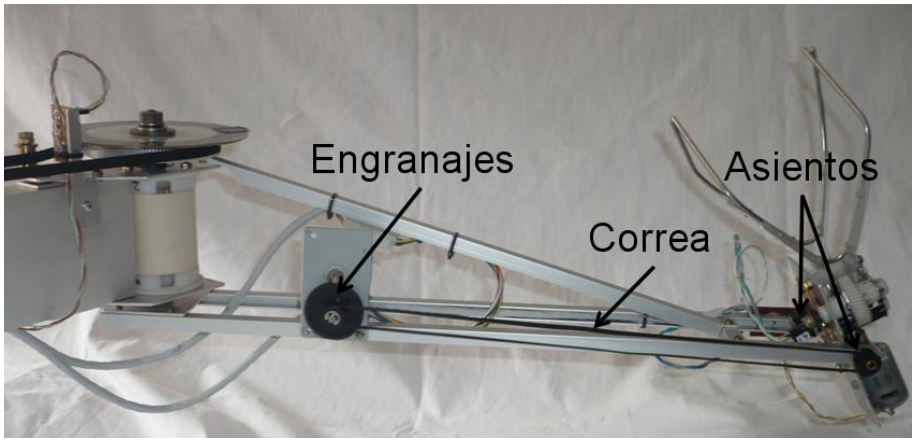


Figura D.5: Vista lateral del antebrazo.

3. *Codo*
   Esta articulación se compone de una masa de bicicleta de plástico puesta en forma vertical, cuyo eje se encuentra solidario al antebrazo (apretado con tuercas en ambos extremos). En la parte superior se encuentra la polea de aluminio que genera el giro. Es importante mantenerla limpia pero no engrasarla. Verificar que la correa esté lo suficientemente tensa pero no en exceso. Para esto, ajuste el tensor hasta ver que la correa no vibre cuando se mueve manualmente la articulación.

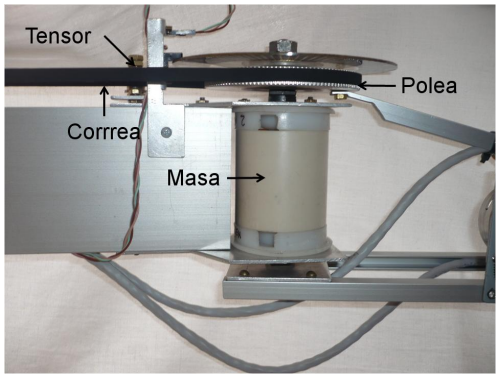


Figura D.6: Articulación del codo.

4. *Hombro*
   Articulación formada por una masa de acero de bicicleta en forma vertical. El eje se encuentra solidario al soporte mientras que la parte móvil al brazo. Sobre la cara superior del brazo asienta el engranaje de aluminio de $14cm$ de diámetro enfrentado al engranaje del motor de $1cm$ de diámetro. Esta articulación es la más delicada de todas por su exigencia de fuerza y velocidad. Es importante entonces mantenerla limpia y engrasada, así como también llevar un control periódico del agarre del motor (chequear que el motor pueda pivotear sobre el eje de giro).

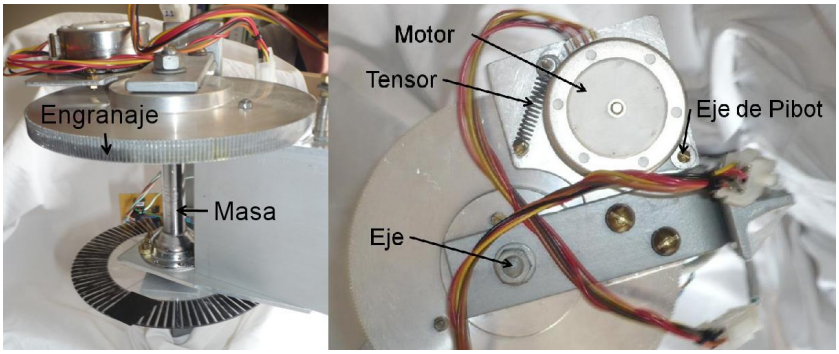


Figura D.7: Articulación del hombro.

Otros cuidados:

1. Para una limpieza más a fondo, es posible desmontar las partes del brazo que se desean [3].
2. Es importante que todas las articulaciones estén bien engrasadas o aceitadas con lubricante de consistencia adecuada [4].
3. Evitar el exceso de grasa en las articulaciones y engranajes. Esto evita la acumulación de polvo y mugre, y por lo tanto un posible mal funcionamiento del sistema. Con el sistema bien lubricado se consigue un mejor funcionamiento y durabilidad del mismo.

[3] ver capitulo: Montaje de Brazo
[4] Ver capítulo: Especificaciones Generales

- **Limpieza de discos y encoders**
  Todos los controles de posición son llevados a cabo por los cuatro conjuntos disco-encoder. Mantener los discos limpios evitando que se rayen. Pueden limpiarse con un paño humedecido en alcohol (no usar solventes). En caso de moverse, cada uno tiene una marca de referencia que indica su lugar.

- **Conexiones eléctricas**
  Todas las conexiones eléctricas del sistema se encuentran numeradas y tienen una única forma de conectarse. El sentido de conexión de cada par conector-header está indicado mediante una flecha en el conector que se corresponde con un punto dibujado en el placa donde se encuentra el header [5].

[5] ver capítulo: Hardware y Conexiones

D.4. Montaje del Brazo

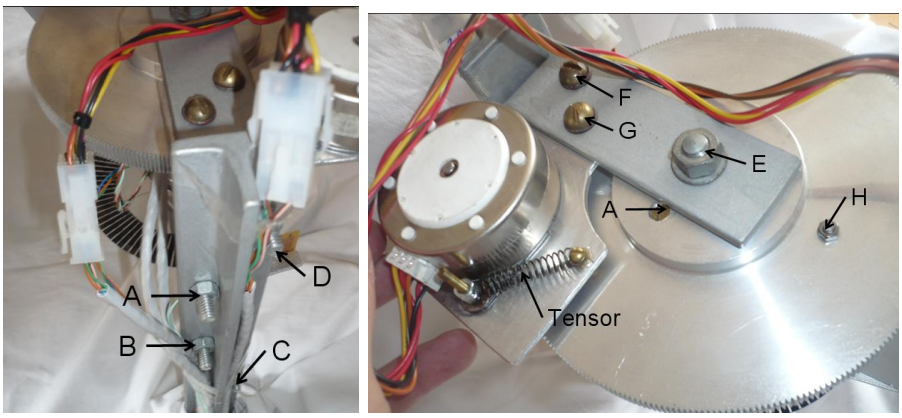
Recomendación importante: se aconseja no desarmar completamente el brazo ya que cuenta con muchas piezas pequeñas y puede dificultarse su reconstrucción. Muchas de estas piezas son hechas a medida y poseen una única forma de colocarse. Para el proceso de armado, siga las instrucciones en el sentido inverso.

Tabla de herramientas		
Herramienta	Tipo	Descripción
Destornillador	Philips Plano	6 mm diámetro 8 mm diámetro 4 mm diámetro
Llaves	Francesa Tubo	se necesitan dos dado de 1/4”
Pinzas	Plana Pico fino	

D.4.1. Articulación del Hombro

Para desarmar esta articulación seguir los siguientes pasos:

1. Desmontar el brazo del soporte. Para esto retirar los tornillos indicados como A, B, C y D en la figura D.8(a).
2. Retirar la tuerca E y la correspondiente a la parte inferior del eje indicadas en la figura D.8(b). Sacar las dos planchuelas de hierro en forma de “L”.
3. Para retirar el soporte de aluminio con el motor retirar los tornillos F y G.
4. Retirar los tornillos A, A’y A”(estos últimos no se aprecian en la fotografía pero se pueden observar al sacar las planchuelas en forma de “L”). Por último sacar el bulón H logrando liberar completamente el engranaje de aluminio.
5. Retirar los conos de la masa de bicicleta.



(a) Montaje del brazo (b) Montaje del motor y engranaje.

Figura D.8: Montaje de componentes del hombro.

D.4.2. Articulación del Codo

Para desarmar esta articulación seguir los siguientes pasos:

1. Desmontar el antebrazo del brazo. Para ello sacar la tuerca A indicada en la figura D.9. Retirar el disco graduado junto con las arandelas y desmontar la correa.
2. Sacar el tornillo B y luego la polea de aluminio.
3. Retirar los bulones C y D logrando con ello liberar el antebrazo.
4. Sacar los bulones E, E'y E"(Este último no se aprecia en la fotografía).
5. Retirar los conos de la masa de bicicleta.

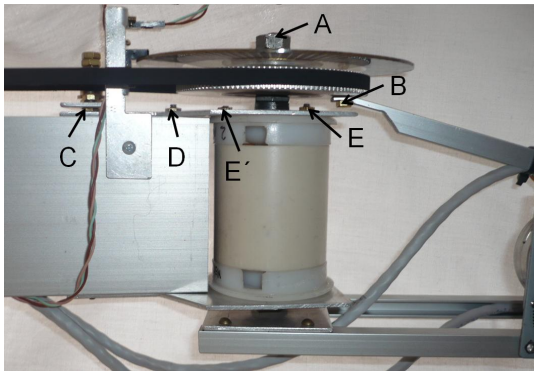


Figura D.9: Montaje del antebrazo.

D.4.3. Muñeca

1. Sacar el tornillo A mostrado en la figura D.11 y retirar el disco de acrílico graduado[6].
2. Aflojar la correa que sostiene la muñeca. Para esto sacar el tornillo B mostrado en la figura D.10 y hacer pivotear el soporte del motor sobre el tornillo C.
3. Sacar los tornillos D y E de la mano que la sujetan al eje. Estos tornillos se muestran en la figura D.12.
4. Sacar completamente la correa y hacer deslizar el eje.

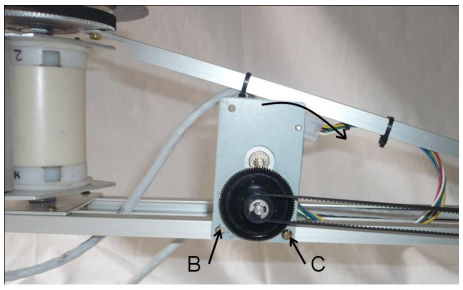


Figura D.10: Montaje del motor de la muñeca.

[6]En esta imagen no se encuentra graduado aún

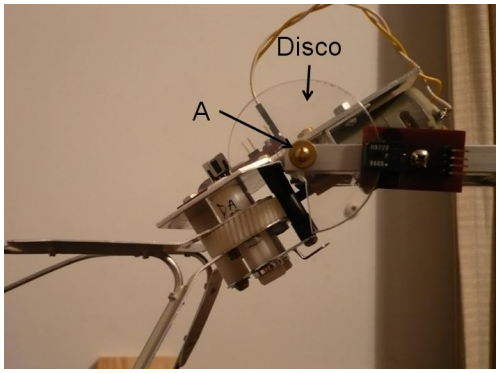


Figura D.11: Vista lateral de la mano.

D.4.4. Pinzas

1. Sacar los tornillos F, G y H que aparecen en la figura D.12.
2. Retirar la lamina de hierro cromada.
3. Retirar las pinzas.

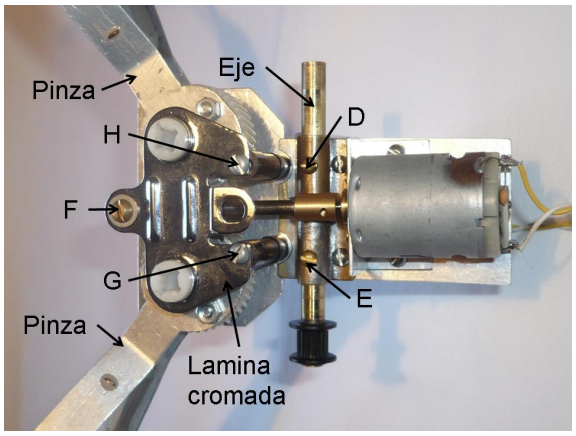


Figura D.12: Estructura de la mano.

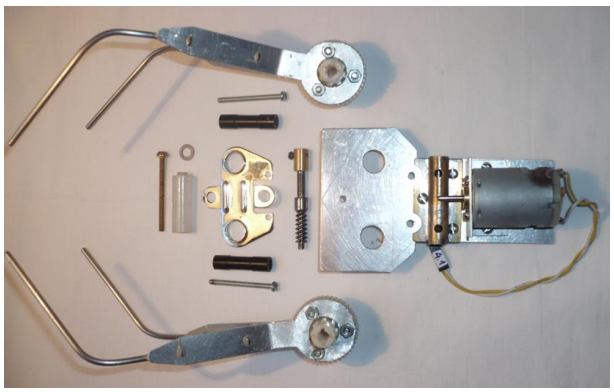


Figura D.13: Mano desarmada.

D.5. Especificaciones Generales

Alimentación:
Tres fuentes de alimentación: una fuente de computadora de $300W$ que entrega $3{,}3V$, $5V$ y $12V$ y dos fuentes de $24V$ y $24W$. Todas con conexión a 220V - 50Hz.

Dimensiones:

- Mesa: (1,30 x 1,30 x 1,50) metros (LxWxH).
- Tablero interfaz: 0,80 x 0,40 x 0,15 metros (LxWxH).
- Brazo: 0,90 metros (desde el eje del hombro a la articulación de la muñeca).
  1,20 metros (desde el eje del hombro a la punta de la pinza cuando esta extendidas).

Ángulos de desplazamiento:
Para la medida del ángulo del hombro se tiene como referencia la línea paralela a la cara posterior de la mesa que pasa por la base del brazo.
El ángulo del codo se mide respecto a la prolongación del brazo. De esta forma si el brazo esta estirado el ángulo es 0^o.
Para la muñeca, la posición más baja corresponde al 0^o y cuando está levantado a 180^o.
Para las pinzas, cuando están cerradas su posición es 0^o y abiertas en 90^o.

Articulación	Ángulo
Hombro	0 / 225
Codo	-115 / 115
Muñeca	0 / 200
Pinza	0 / 120

Lubricante:
Cada articulación debe ser engrasada con un lubricante de consistencia adecuada, se aconseja usar como referencia la siguiente tabla.

Articulación	Tipo de lubricante
Hombro	grasa N^o 2
Codo	grasa N^o 2
Muñeca	grasa N^o 2
Pinzas	grasa N^o 2

Componentes mecánicos:
A continuación se detallan los componentes mecánicos retirados de equipamiento en desuso, se indica el modelo y marca para facilitar su reemplazo en caso de ser necesario.

Componente	Modelo	Repuesto
Motores	PM55L048	impresora: HP Deskjet 9800
	FDK SB40	escáner: Genius ColorPage Vivid II
	Thomson	secador de pelo: GAMA Mystere 3000
Correas	Muñeca	escáner: Microtek V300
	Hombro	escáner: Microtek SCSI
Engranajes	Codo	hecho en: Grosso S.A.
	Hombro	hecho en: Grosso S.A.

Condiciones ambientales:

Temperatura	5 a 40 °C
Humedad	menor a 90 %

Periodo de mantenimiento:
Se debe limpiar periódicamente todo el sistema para garantizar un buen funcionamiento, se aconseja realizarla según la siguiente tabla (estos tiempos pueden variar según el lugar donde esté operando el sistema).

Parte	Tiempo
Interfase	1 mes
Placas control	6 mes
Mecánica	6 meses
Software	no requiere

D.6. Tablero de interfaz de usuario

El tablero de interfaz con el usuario se compone de pulsadores industriales tipo hongo y rasantes, de placas compuestas por LEDs blancos de alto brillo y lámparas que vienen en conjunto con los pulsadores tipo hongo. Todos estos componentes eléctricos se encuentran disponibles en plaza por lo que será fácil su reposición en caso de rotura.
La interfaz de usuario está armada de forma tal que cada una de sus partes se puede desconectar y retirar sin comprometer al resto. Como se muestra en secciones posteriores, su desconexión del resto del sistema es sencilla dado que se utilizan conectores bien identificados.

La figura D.14 muestra la parte interior y exterior del tablero de interfaz de usuario sin las terminaciones de apariencia pero con todas las conexiones realizadas.

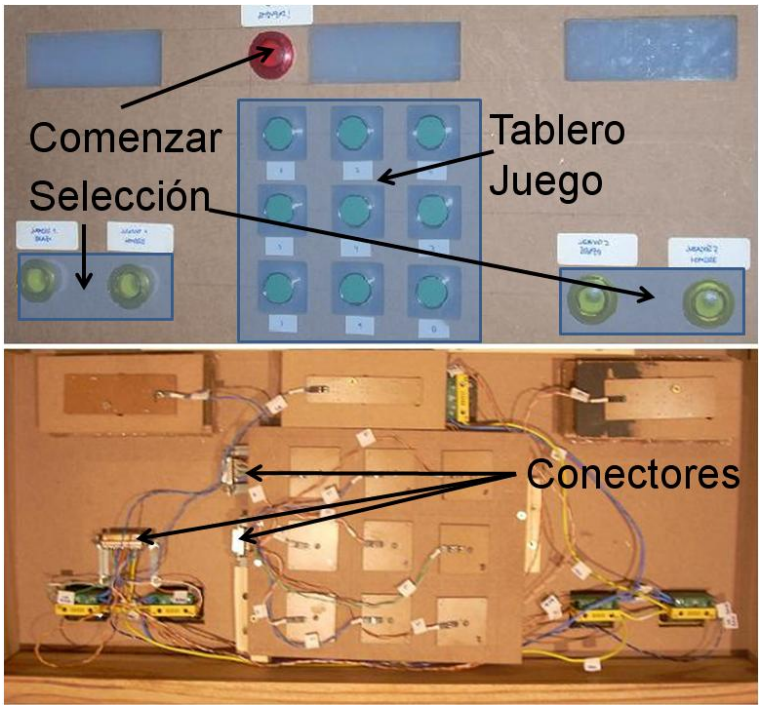


Figura D.14: Vista interior y exterior del tablero de interfaz de usuario.

D.7. Caracterización del sistema

Las etapas y comportamiento normal del brazo al comenzar y culminar un juego se realizan como se describe en la tabla a continuación.

Secuencias de un juego de Tatetí	
Acción del usuario	Reacción del sistema
Encender el juego con la llave que se encuentra dentro del mueble	El dispositivo mecánico se coloca en la posición *Home* y aguarda por que se presione el botón **COMENZAR**.
Se presiona el botón **COMENZAR**	Las tres luces asociadas con el jugador 1 (luz del cartel de jugador 1 y luces de los botones de selección de jugador) tintinean aguardando que se seleccione al mismo.
Se presiona uno de los dos botones de selección de jugador (Brazo o Persona)	Queda encendida únicamente la luz del botón seleccionado. Comienzan a tintinear las luces correspondientes al jugador 2, aguardado que se seleccione al mismo.
Se presiona uno de los dos botones de selección de jugador	Queda encendida únicamente la luz del botón seleccionado. Tintinea la luz del botón del jugador 1 que había quedado encendida indicando que es su turno. Si el jugador 1 corresponde al Brazo, inmediatamente realiza su jugada encendiendo la luz en el tablerito del casillero a donde moverá su ficha y realiza el movimiento. De lo contrario, queda a la espera de una jugada.
Se presiona el pulsador de un casillero cualquiera	Se enciende la luz correspondiente al casillero seleccionado y el dispositivo mecánico coloca la ficha. Cuando termina el movimiento deja encendida la luz del jugador que acaba de jugar y pasa a tintinear la luz del botón del otro jugador indicando su turno.
Termina el juego	
Se determina que es un empate	Las luces de los dos carteles de **JUGADOR 1** y **JUGADOR 2** tintinean al mismo tiempo.
Gana uno de los jugadores	Las tres luces correspondientes al jugador se prenden y apagan de forma circular.
En todos los casos, luego de un tiempo de indicar el resultado de la partida, se comienza a retirar las fichas del tablero y depositarlas en la rampa correspondiente. Se retiran en orden comenzando por el casillero inferior derecho y a medida que las deposita apaga la luz del casillero. Durante todo este proceso se encuentra la luz de espera encendida.	

D.7.1. Características del comportamiento general del sistema

- Cuando se dice que el brazo coloca una ficha, implica que retira la ficha de la rampa correspondiente al jugador del turno, la coloca en el casillero seleccionado y vuelve a la posición de *Home*. Al retirar una ficha del tablero, realiza el procedimiento inverso: retira la ficha del casillero, la deposita en la parte superior de la rampa correspondiente al jugador del turno y vuelve a la posición de *Home*.

- *Estado de Espera:* Cada vez que el brazo se encuentre en movimiento, se encenderá el

cartel de **ESPERE...** hasta que el mismo deje de moverse. En estos períodos de tiempo, el juego hará caso omiso a que se presione cualquier botón del tablero.

- *Culminación de una partida:* Si en cualquier momento del transcurso de una partida y estando el cartel de **ESPERE...** apagado, se presiona el botón **COMENZAR**, se culminará la partida y el dispositivo mecánico pasará inmediatamente a retirar las fichas del tablero para comenzar un nuevo juego.

- *Tiempo "Idle":* Luego de culminada una partida y retiradas las fichas del tablero, el sistema queda a la espera de la indicación de un nuevo juego. Si luego de cierto tiempo no se comienza una nueva partida, sistema realiza una serie de movimientos y encendido de luces hasta que se presione el botón **COMENZAR**.

- *Corte de energía:* El sistema, luego que se restablece de un corte de energía, evalúa la situación en la que se encontraba y retira todas las fichas del tablero. No continúa con el juego anterior.

- *Condición de arranque particular:* Cuando el programa que contiene el Rabbit es cargado por primera vez desde una PC o cuando por algún motivo, se levantaron las fichas del tablero y se modificó la posición del brazo manualmente estando el sistema apagado, se debe indicar al microprocesador de esta situación. Para ello, cuando se enciende el sistema y durante el parpadeo inicial de las luces del tablero, se debe mantener presionados el botón **COMENZAR** y el pulsador de la posición **CERO** ubicado en el extremo inferior derecho.

- *Culminación de tiempos de espera:* Para poder determinar si se abandonó la partida, el sistema contabiliza el tiempo transcurrido entre cada etapa del juego y, si el tiempo de espera por una jugada supera un tiempo dado[7], inmediatamente culmina la partida retirando todas las fichas del tablero.

D.7.2. Tiempos característicos del sistema

La tabla siguiente detalla alguno de los tiempos más importantes del sistema que se deben tener en cuenta a la hora de jugar.

Descripción	Tiempo
Tiempo de espera para seleccionar un jugador una vez que comienzan a tintinear las luces, configurado por software	$30s$
Tiempo de espera para que se realice una jugada, configurado por software	$30s$
Tiempo de espera para entrar en estado "Idle", configurado por software	$5min$
Tiempo promedio en colocar una ficha en un casillero y volver a *home*	$20s$
Tiempo promedio en colocar una ficha en un casillero	$16s$
Tiempo promedio en retirar una ficha del tablero y volver a *home*	$21s$
Tiempo promedio en retirar una ficha del tablero hasta colocarla en la rampa	$14s$
Tiempo total en limpiar todo el tablero (nueve fichas colocadas)	$3:20min$

Los tiempos de posicionamiento se calcularon como el promedio de los tiempos de cada una de las posiciones del tablero.

[7] Este tiempo es ajustable en el programa

D.8. Detalle de Hardware eléctrico y conexiones

El sistema de control del brazo está formado por seis placas de electrónica, las cuales deben ser limpiadas periódicamente para evitar la acumulación de polvo o mugre. La limpieza es importante por la gran disipación que tienen las mismas cuando el brazo esta en funcionamiento.

Hardware de Control	
Placa	Cantidad
Control Sistema	1
Control Interfaz	1
Motor DC	1
Motor Paso	3

D.8.1. Descripción del hardware de control

Las figuras siguientes pretenden mostrar de forma sencilla la estructura de cada una de las placas mencionadas en la tabla anterior. Para ello se realizaron esquemáticos representativos de las mismas para indicar especialmente las borneras de conexión con el resto del sistema y la nomenclatura que se utilizó para referirse a las mismas. En todas ellas se indican en azul las borneras de conexión y en naranja los componentes mas significativos de cada una de las placas.

Descripción de componentes	
Chip	Descripción
74LS541	octal buffer 3-state
ULN2803	octal Darlington
L297	controlador Stepper-Motor
L298	doble puente H de potencia
LMD18200	puente H de potencia
TIP42	transistor de potencia (PNP)
LM393	doble comparador-diferencial
LM555	Timer
RCM3365	modulo que contiene el Rabbit

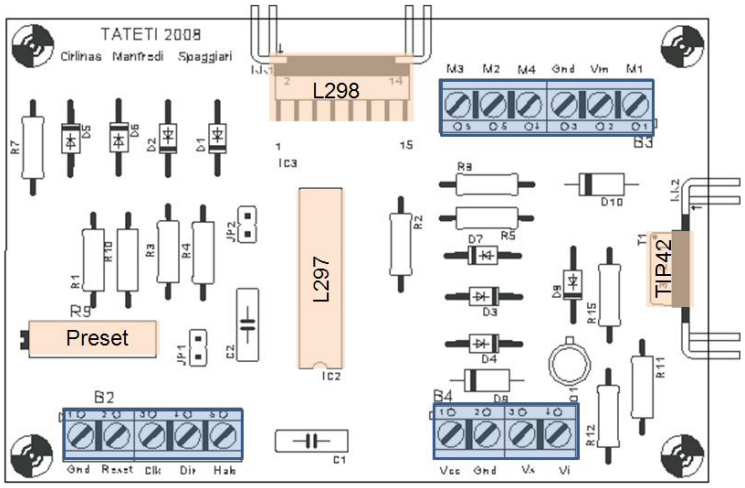


Figura D.15: **Motor Paso**. Placas de potencia utilizadas para el control de los motores de paso.

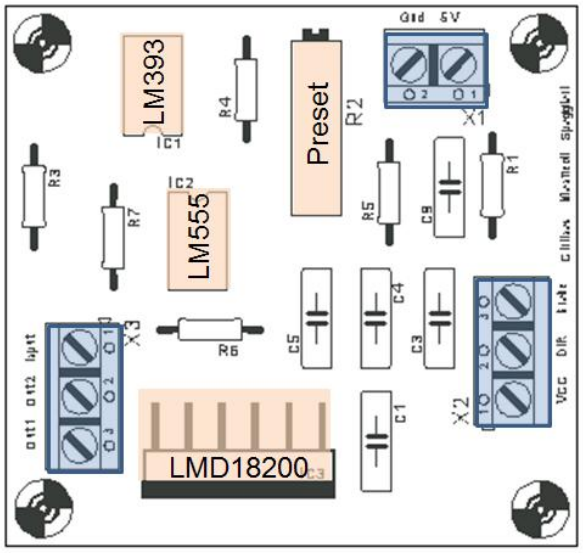


Figura D.16: **Motor DC**. Placa de potencia utilizada para el control del motor de continua de la mano.

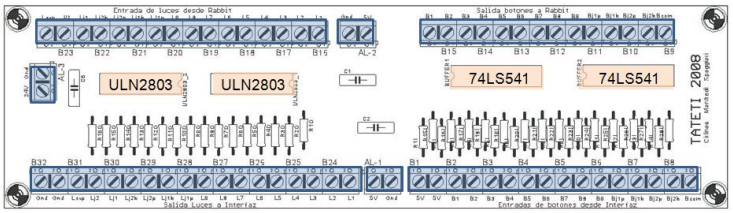


Figura D.17: **Control Interfaz**. Placa de control de interfaz que contiene la etapa de entrada-salida de los componentes de la interfaz de usuario.

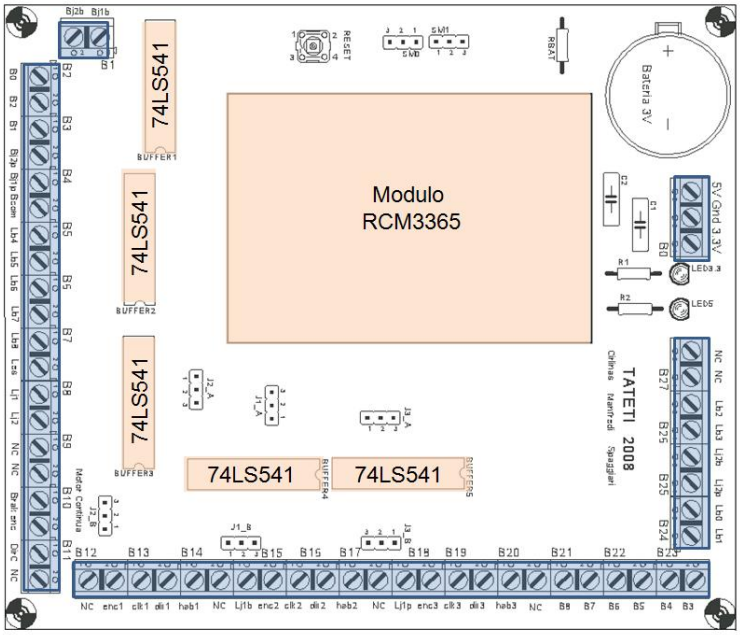


Figura D.18: **Control Sistema**.Placa principal del control del sistema que contiene al microprocesador Rabbit.

D.8.2. Tablas de Conexión entre los componentes del sistema

	Señal	Borneras											
Alimentación	Gnd	Placa Rabbit	0.2	Placa Interfaz	4.2	Placa Hombro	2.1 - 3.3 - 4.2	Placa Codo	2.1 - 3.3 - 4.2	Placa Muñeca	4.2	Placa Mano	1.2
	3.3 V		0.1		---		---		---		---		---
	5 V		0.3		4.1		2.2 - 4.1		2.2 - 4.1 - 4.3		2.2 - 4.1 - 4.3		1.1
	12 V		---		4.3		4.3		4.4		4.4		2.1
	24 V		---		4.4		4.4		---		---		---

Figura D.19: Borneras de alimentación para cada placa.

Para interconectar las señales provenientes de la placa de control de interfaz mostrada en la figura D.17 con los componentes de la interfaz de usuario (botones, placas de LEDs y luces), se armaron varios cables con conectores DB25 y DB15 para facilitar su conexión y desconexión. En la figura D.20 se muestran los conectores mencionados conectados al conector hembra ubicados en la parte interior del tablero de interfaz.

La tabla D.21 detalla el armado y código de colores de cada cable, al igual que las señales de cada uno de los pines de los conectores.

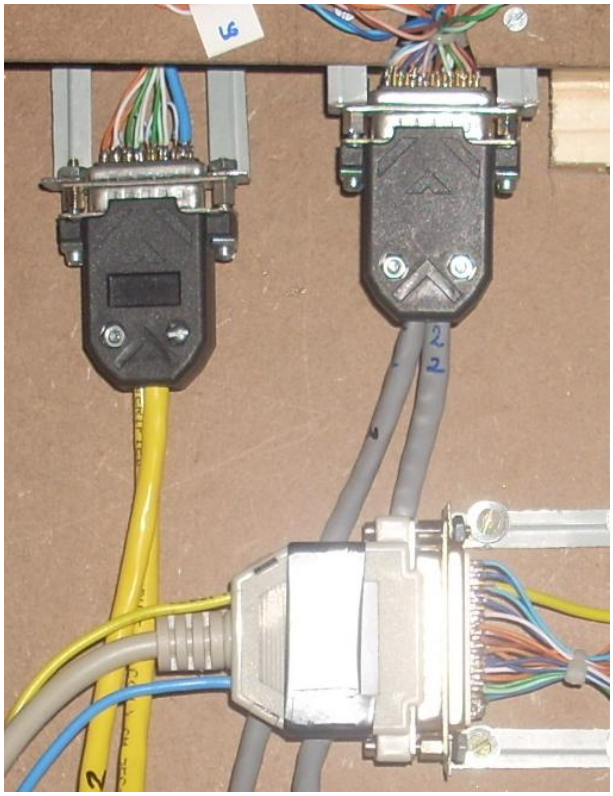


Figura D.20: Conectores DB15 y DB25 utilizados para conectar la interfaz de usuario con la placa de interfaz.

Los cables que interconectan a los componentes eléctricos del brazo con las placas de control del mismo se detallan en la tabla siguiente donde se especifica el nombre de la bornera de origen, el número de conector y el color de los cables en cada extremo. Esta tabla permite la correcta conexión de los conectores una vez que fueron retirados.

Conector	Parte	Pin	Color cable	señal	
DB15 Gris	1		Naranja	luz botón	5
			Naranja blanco	luz botón	2
			Verde	Común	24V
			Verde blanco	luz botón	7
			Marrón	Común	24V
			Marrón blanco	luz botón	8
			Azul	Común	24V
			Azul blanco	luz botón	1
	2		Naranja	luz botón	3
			Naranja blanco	Común	24V
			Verde	luz botón	0
			Verde blanco	Común	24V
			Marrón	NC	
			Marrón blanco	Común	24V
			Azul	luz botón	4
			Azul blanco	luz botón	6
DB15 amarillo	1		Naranja	botón	8
			Naranja blanco	botón	7
			Verde	botón	4
			Verde blanco	botón	5
			Marrón	botón	2
			Marrón blanco	Común	5V
			Azul	botón	1
			Azul blanco	Común	5V
	2		Naranja	botón	3
			Naranja blanco	botón	6
			Verde	Común	5V
			Verde blanco	Común	5V
			Marrón	botón	0
			Marrón blanco	Común	5V
			Azul	Común	5V
			Azul blanco	Común	5V
DB25 gris	1	21 a 25	Amarillo grueso	Común	24V
		1-2-14-15	Azul grueso	Común	5V
		3	Marrón	botón	Bj2p
		4	Marrón blanco	botón	Bj2b
		5	Rojo	botón	Bj1b
		6	Rojo blanco	botón	Bj1p
		7	Naranja	botón	Bcom
		8	Naranja blanco	luz	Lj1
		9	Rosado	luz	24V
		10	Amarillo	luz	Lj2
		11	Verde	luz	24V
		12	Verde blanco	luz	Lesp
		13	Verde claro	luz	24V
		16	Violeta	luz botón	Lj2p
		17	Gris	luz botón	Lj2b
		18	Negro	luz botón	Lj1p
		19	Azul	luz botón	Lj1b
		20	Azul blanco	luz botón	Lcom

Figura D.21: Cables de conexión entre los conectores del tablero de interfaz de usuario y la placa de control de interfaz.

Hardware	Conector	Color cable 1	Color cable 2	Señal placas
Motor Hombro	1.1 y 1.2	Rosado	Marrón	M4
		Verde	Naranja	M2
		Marrón	Negro	M1
		Naranja	Amarillo	M3
Motor Codo	2.1 y 2.2	Rosado	Marrón	M2
		Verde	Naranja	M4
		Marrón	Negro	M1
		Naranja	Amarillo	M3
Motor Muñeca	3.1 y 3.2	Rosado	Azul	M3
		Verde	Blanco	M1
		Marrón	Rojo	M2
		Naranja	Amarillo	M4
Motor Mano	4.1 y 4.2	Blanco	Blanco	Out 2
		Amarillo	Verde claro	Out 1
Encoder Hombro	5	Naranja		GND
		Verde		VCC
		Azul		enc1
Encoder codo	6.1 y 6.2	Marrón	Marrón	VCC
			Verde claro	enc2
		Rosado	Rosado	GND
Encoder muñeca	7	Rosado		puente
		Verde		GND
		Celeste		Vcc
		Azul		enc3
Encoder mano	8.1 y 8.2	Rosado	Rosado	encC
		Verde	Verde	GND
		Celese	Celeste	Vcc

Figura D.22: Detalle de cables y conectores de los headers del brazo.

	Descripción	Etiqueta	Origen	I/O	Bornera	Destino	I/O	Bornera
Botones	Comenzar	BCom	Placa Rabbit	I	4.2	Placa Interfaz	O	9.1
	J1 Brazo	Bj1b		I	1.1		O	10.2
	J1 Hombre	Bj1p		I	4.1		O	11.1
	J2 Brazo	Bj2b		I	1.2		O	9.2
	J2 Hombre	Bj2p		I	3.2		O	10.1
	0	B0		I	2.1		O	15.2
	1	B1		I	3.1		O	15.1
	2	B2		I	2.2		O	14.2
	3	B3		I	23.2		O	14.1
	4	B4		I	23.1		O	13.2
	5	B5		I	22.2		O	13.1
	6	B6		I	22.1		O	12.2
	7	B7		I	21.2		O	12.1
	8	B8		I	21.1		O	11.2
Luces	Espera	Les	Placa Rabbit	O	7.2	Placa Interfaz	I	23.2
	J1	Lj1		O	8.1		I	22.2
	J1 Hombre	Lj1p		O	18.1		I	20.2
	J1 Brazo	Lj1b		O	15.1		I	21.1
	J2	Lj2		O	8.2		I	23.1
	J2 Brazo	Lj2b		O	25.2		I	22.1
	J2 Hombre	Lj2p		O	25.1		I	21.2
	0	Lb0		O	24.2		I	16.1
	1	Lb1		O	24.1		I	16.2
	2	Lb2		O	26.2		I	17.1
	3	Lb3		O	26.1		I	17.2
	4	Lb4		O	5.1		I	18.1
	5	Lb5		O	5.2		I	18.2
	6	Lb6		O	6.1		I	19.1
	7	Lb7		O	6.2		I	19.2
	8	Lb8		O	7.1		I	20.1
Motor hombro	Dir	Dir1	Placa Rabbit	O	13.2	Placa Hombro	I	2.4
	Hab	Hab1		O	14.1		I	2.5
	Clk	clk1		O	13.1		I	2.3
	Encoder	enc1		I	12.2			
Motor Codo	Dir	Dir2	Placa Rabbit	O	16.2	Placa Codo	I	2.4
	Hab	Hab2		O	17.1		I	2.5
	Clk	clk2		O	16.1		I	2.3
	Encoder	enc2		I	15.2			
Motor Muñeca	Dir	Dir3	Placa Rabbit	O	19.2	Placa Muñeca	I	2.4
	Hab	Hab3		O	20.1		I	2.5
	Clk	clk3		O	19.1		I	2.3
	Encoder	enc3		I	18.2			
Motor Mano	Dir	DirC	Placa Rabbit	O	11.1	Placa Mano	I	2.2
	Brake	Brake		O	10.1		I	2.3
	Encoder	enc		I	10.2			
NC			Placa Rabbit		14.2			
					9.1			
					9.2			
					12.1			
					11.2			
					27.2			
					27.1			
					17.2			
					20.2			

Apéndice E

Contenido del CD

E.1. Estructura del CD

A continuación se presenta el árbol de directorios correspondiente al CD que se adjunta con la presente documentación.

```
/
├── TATETI.PDF
├── Archivos fuente del software
├── Diseño de hardware
├── Hojas de datos de integrados
├── Hojas de datos de motores
├── Manuales de Rabbit Semiconductor
├── Planillas de conexionado
└── Planos de AutoCAD
```

E.2. Descripción de los archivos

En esta sección se describe el contenido de cada uno de los archivos contenidos en las diferentes carpetas del CD.

Viendo la estructura del CD en la sección anterior se puede ver que en el primer nivel (junto con los directorios) se encuentra el archivo que contiene el presente documento, llevando el nombre de `TATETI.PDF`.

E.2.1. Archivos fuente del software

Nombre del archivo	Descripción
`calibracionMotores.C`	Programa auxiliar de calibración de posiciones de los casilleros del tablero de juego en Dynamic C
`calibracionMotores.DCP`	Archivo del proyecto asociado a `calibracionMotores.C`
`ioLib.LIB`	Biblioteca de rutinas de control de interfaz en Dynamic C
`main.C`	Programa principal del juego en Dynamic C
`motoresLib.LIB`	Biblioteca de rutinas de control de motores en Dynamic C
`tateti.DCP`	Archivo del proyecto asociado a `main.C`
`tatetiLib.LIB`	Biblioteca de rutinas de la inteligencia artificial del brazo
`tatetiLibDir.DIR`	Directorio de bibliotecas de los proyectos `calibracionMotores.DCP`, `tateti.DCP` y `testCalibracionMotores.DCP`.
`testCalibracionMotores.C`	Programa auxiliar para verificar la correcta calibración de las posiciones de los casilleros del tablero de juego en Dynamic C
`testCalibracionMotores.DCP`	Archivo del proyecto asociado a `testCalibracionMotores.C`

Tabla E.1: Contenido del directorio `Archivos fuente de software`.

Es importante recordar que la versión de Dynamic C en la cual se implementaron, verificaron y compilaron estos archivos es la 9.21, que se encuentra disponible en Internet. No se garantiza que estos fuentes funcionen correctamente en una versión diferente.

Para que los archivos `.C` incluidos en esta carpeta puedan ser compilados y la funcionalidad del *Library Function Lookup* pueda ser utilizada, primero se debe copiar el archivo `tatetiLibDir.DIR` en la carpeta raíz del Dynamic C (`C:\DCRABBIT_9.21` es la por defecto) y guardar todos los fuentes del proyecto en la carpeta `C:\DCRABBIT_9.21\TATETI`. Además, abriendo los archivos de proyecto (de extensión `.DCP`) se puede configurar automáticamente todas las preferencias de compilación del Dynamic C necesarias para hacer que los archivos `.C` asociados trabajen sobre el módulo RCM3365 conectado al juego.

E.2.2. Diseño de Hardware

Nombre del archivo	Descripción
`Interfaz.BRD`	PCB en Eagle de la placa de potencia de control de intefaz
`Interfaz.SCH`	Esquemático en Eagle de la placa de potencia de control de interfaz
`motor_continua.BRD`	PCB en Eagle de la placa de potencia para el motor de continua
`motor_continua.SCH`	Esquemático en Eagle de la placa de potencia para el motor de continua
`Potencia_final.BRD`	PCB en Eagle de la placa de potencia para los motores paso a paso
`Potencia_final.SCH`	Esquemático en Eagle de la placa de potencia para los motores paso a paso
`rabbit.BRD`	PCB en Eagle de la placa para el módulo RCM3365

Tabla E.2: Contenido del directorio `Diseño de hardware`.

Todos los esquemáticos y PCBs fueron creados en Eagle versión 4.11.

E.2.3. Hoja de datos de integrados

Nombre del archivo	Descripción
`74LS541.PDF`	Hoja de datos del integrado *buffer/line driver* 74LS541
`L297.PDF`	Hoja de datos del integrado *Stepper motor controler* L297.
`L298N.PDF`	Hoja de datos del integrado *Dual Full bridge driver* L298N.
`LM393.PDF`	Hoja de datos del comparador doble LM393.
`LM555.PDF`	Hoja de datos del timer LM555.
`LMD18200.PDF`	Hoja de datos del integrado puente H LMD18200.
`ULN2803.PDF`	Hoja de datos del integrado *Octal high voltage, high current Darlington transistor arrays* ULN2803

Tabla E.3: Contenido del directorio `Hoja de datos de integrados`.

E.2.4. Hoja de datos de motores

Nombre del archivo	Descripción
`FDK-SM-SME001.PDF`	Hoja de datos de los motores paso a paso FDK
`PermanentMagnetSeriesStepMotors.PDF`	Hoja de datos de toda la serie de motores PM (Permanent Magnet) de NMB (Minebea Motor Manufacturing Corporation).
`PM55L048.PDF`	Hoja de datos del motor de paso PM55L-048 de NMB.

Tabla E.4: Contenido del directorio `Hoja de datos de motores`.

E.2.5. Manuales de Rabbit Semiconductor

Nombre del archivo	Descripción
`DCPUM.PDF`	Manual de usuario de Dynamic C.
`090-0188.PDF`	Esquemático de la placa de prototipado de la serie de Rabbit Core RCM3300
`090-0214.PDF`	Esquemáticos del módulo Rabbit Core RCM3365.
`R3000UM.PDF`	Manual de usuario del microprocesador Rabbit 3000.
`RCM3365UM.PDF`	Manual de usuario del Rabbit Core RCM3365.

Tabla E.5: Contenido del directorio `Manuales de Rabbit Semiconductor`.

Estos manuales y esquemáticos se pueden encontrar también en la página web de Rabbit Semiconductor `www.rabbitsemiconductor.com`.

E.2.6. Planillas de conexionado

Nombre del archivo	Descripción
`asignacion de puertos.PDF`	Planilla con asignación de los puertos del Rabbit y su conexión al resto del sistema.
`cables y conectores.PDF`	Planilla donde se detallan los conectores y cables utilizados para la alimentación de los componentes sobre el brazo.
`conectores de interfaz.PDF`	Planilla que detalla el armado de los conectores que llevan las señales desde y hacia la interfaz con el usuario.

Tabla E.6: Contenido del directorio `Planillas de conexionado`.

E.2.7. Planos de AutoCAD

Nombre del archivo	Descripción
`interfaz tablero 3.PDF`	Planos en AutoCAD del diseño de interfaz de usuario
`modelo mano 1.PDF`	Modelo de la mano en AutoCAD vista 1
`modelo mano 2.PDF`	Modelo de la mano en AutoCAD vista 2
`modelo mano 3.PDF`	Modelo de la mano en AutoCAD vista 3
`modelo mano 4.PDF`	Modelo de la mano en AutoCAD vista 4
`modelo mano 5.PDF`	Modelo de la mano en AutoCAD vista 5
`ubicacion brazo y mano 1.PDF`	Modelo del mueble de juego, brazo y mano en AutoCAD vista 1
`ubicacion brazo y mano 2.PDF`	Modelo del mueble de juego, brazo y mano en AutoCAD vista 2
`ubicacion brazo y mano 3.PDF`	Modelo del mueble de juego, brazo y mano en AutoCAD vista 3
`ubicacion brazo y mano 4.PDF`	Modelo del mueble de juego, brazo y mano en AutoCAD vista 4
`ubicacion brazo y mano 5.PDF`	Modelo del mueble de juego, brazo y mano en AutoCAD vista 5

Tabla E.7: Contenido del directorio `Planos de AutoCAD`.

Todos estos planos fueron creados en AutoCAD 2000 Educational Version.

E.3. Requerimientos del sistema

- **Sistema operativo:** Windows 98/XP o superior
- **Memoria RAM (mínima):** 64 MB
- **Resolución de pantalla:** 800 x 600 o superior
- **Velocidad de lectora de CD-ROM:** 16x o superior
- **Visor de PDF:** Acrobat Reader 5 o superior
- **Eagle:** Eagle versión 4.11 o superior
- **Dynamic C:** Dynamic C versión 9.21
- **AutoCAD:** AutoCAD 2000 Educational Version o compatible

Bibliografía

[1] *Institut dÓrganització i Control de Sistemes Industrials (UPC)*, Luis Basañez Villaluenga - Coordinador Nacional Español en la IFR
http://www.interempresas.net/MetalMecanica/Articulos/Articulo.asp?A=1464

[2] Real Academia Española
http://buscon.rae.es/draeI/SrvltConsulta?TIPO_BUS=3&LEMA=robot

[3] Mundo Robot, *Entre esteriotipos y verdades*
http://hypatia.morelos.gob.mx/reportajesmundo/_robot_1.htm

[4] Ciencia Viva – Una propuesta para aprender juntos.
http://cienciaviva.fcien.edu.uy

[5] Martín Alcalá Rubí, Nicolás de Mula y Andrés Pietrafesa, Proyecto: *UMANIS, Un manipulador inteligente*, 2005.

[6] "How to build a robotic arm", *robotclothes.com*,
http://www.robotclothes.com/insideout/archives/2005/08/big_humanoid_ar.html

[7] "Robot arm tutorial", *Society of Robots*,
http://www.societyofrobots.com/robot_arm_tutorial.shtml

[8] "Robotic Arm", *Giandomenico´s Page*,
http://gidesa.altervista.org/roboticarm.php

[9] *Robots, pasión por la robótica en Argentina*,
http://robots-argentina.com.ar/robots.htm

[10] "Taller de robótica", *Robótica en Mendoza*,
http://www.roboticajoven.mendoza.edu.ar/taller.htm

[11] "Mecánica - Motores servos", *X - Robotics*
http://www.x-robotics.com/motorizacion.htm

[12] "SureEstep", *Sistema de motores paso a paso*
http://www.automationdirect.com/static/manuals/surestepmanualsp/surestepmanualsp.html

[13] Rabbit Semiconductor,
http://www.rabbit.com/

[14] Hoja de datos del integrado L297, ST.

[15] Hoja de datos del integrado L298N, ST.

[16] Hoja de datos del LMD18200T, National Semiconductor.

[17] Hoja de datos del integrado LM555.

[18] RC3365UM, *Rabit Core RCM3365/RCM3375 - User's Manual* http://www.rabbit.com/documentation/docs/manuals/RCM3365/UsersManual/RC3365UM.pdf

[19] Cadsoft – Eagle versión 5, http://www.cadsoft.de/

[20] DCPUM, *Dynamic C - User's Manual* http://www.rabbit.com/documentation/docs/manuals/RCM3365/UsersManual/DCPUM.pdf

[21] Brian W. Kernighan, *El lenguaje de programación C*, Mexico, 1991.

[22] Kamal Hyder, "Embedded Systems Design using the Rabbit 3000 Microprocessor", *Interfacing, Networking, and Application Development*, Rabbit Semiconductor, 2005.

[23] Alpha-beta pruning, *Wikipedia* http://en.wikipedia.org/wiki/Alpha-beta_pruning

[24] "SMB Source Code Site", http://www.angelfire.com/blues/smb/cpp.htm

[25] Hoja de datos del buffer 74LS541, Motorola.

[26] Hoja de datos del ULN2803, Motorola.

[27] "Motores paso a paso", *Wikipedia* http://es.wikipedia.org/wiki/Motor_paso_a_paso

[28] "Stepper Motors" *FDK – Technology creating a better future* http://www.fdkamerica.com/motor.htm

[29] *NMB Permanent magnet (PM) Step Motors* http://www.nmbtech.com/step_stepping_motor_hybrid_permanent.htm

[30] LaTeX, http://www.latex-project.org/