UNIVERSIDAD DE  VALLADOLID

E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN, MENCIÓN EN SISTEMAS DE TELECOMUNICACIÓN

Adaptación y Diseño de un Nuevo Sistema de Interfaz para una Silla de Ruedas

Autor:
D. Alvaro Riol Triviño
Tutor:
D. Alonso Alonso Alonso
Valladolid, 10 de Septiembre de 2015

TÍTULO:	**Adaptación y Diseño de un Nuevo Sistema de Interfaz para una Silla de Ruedas**
AUTOR:	**D. Alvaro Riol Triviño**
TUTOR:	**D. Alonso Alonso Alonso**
DEPARTAMENTO:	**Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE:	**D. Alonso Alonso Alonso**
VOCAL:	**D. Ramón de la Rosa Steinz**
SECRETARIO	**D. Jaime Gómez Gil**
SUPLENTE	**D. Juan Pablo de Castro Fernández** **D. Alberto Izquierdo Fuente**

Resumen

La discapacidad física es un problema que afecta a un número de personas más alto del que nos imaginamos. Estas discapacidades se pueden producir por accidentes o incluso por enfermedades degenerativas que van dejando al individuo sin la capacidad de moverse. Por ello, en el laboratorio de electrónica y bioingeniería de la Universidad de Valladolid se desarrollan proyectos para proporcionar esas ayudas técnicas.

El trabajo consiste en el estudio de las interfaces de control de sillas de ruedas para discapacitados. Se ha desarrollado un software adecuado para el control de una silla real. También se ha construido una interfaz novedosa para el control con los pies basada en *ARDUINO*. Además, se han optimizado diversos componentes necesarios para el control mecánico indirecto del joystick. Asimismo, y para terminar, se han realizado pruebas con discapacitados y con sujetos de control para estudiar la viabilidad y prestaciones del nuevo interfaz.

PALABRAS CLAVE: discapacidad, sensor, joystick, *Arduino*, silla de ruedas, enfermedad degenerativa, interfaces aferentes.

Abstract

Physical disability is a problem that affects a number of people higher than we imagine. These disabilities can be produced by accidents or even by degenerative diseases which leave the person without the capacity to move. Thus, in the electronic and bioengineering of the University of Valladolid projects are developed to provide those technical aids.

Work consists of the study of control interfaces wheelchair for disabled people. It has developed a suitable software for the control of one real wheelchair. Also, it has built a new interface to handle the wheelchair with the foot based on *ARDUINO*. It has optimized many necessary components to the indirect mechanical control of the joystick. We have also been tested with disabled people and with control subjects to study the viability and perfomance of the new interface.

KEYWORDS: disabilities, sensor, A*rduino*, wheelchair, degenerative disease, afferent interfaces.

A toda mi familia,
Quienes tras cada caída,
Me enseñaron a levantarme

Agradecimientos

En primer lugar, darles las gracias a Alonso y Albano por toda la paciencia y ayuda prestada durante el proyecto. Especialmente en Alonso por depositar la confianza en mí para llevar a cabo las tareas que me pidió.

No podía faltar agradecer a Justo y Eva su paciencia conmigo durante la realización de todas las pruebas.

Tampoco podía faltar el agradecimiento a mi amiga Paula García por su enorme interés a la hora de realizar toda la memoria del proyecto. A mi amigo Sergio Quintero por su interés tanto en la memoria del proyecto como en el trabajo llevado a cabo. También darles las gracias a todas las personas que me han ayudado y se han interesado por el transcurso del proyecto: Kike, Jorge, Rodri, Samuel, Tino, Mario, Ismael y Pastrana. Darles las gracias también a todos los que me han acompañado en la carrera, en especial a Benito por todos los buenos momentos, a Tito por lo locos que estamos, a Rebeca por su “paciencia”, a Juanpa y a Alber, con los que he pasado largas jornadas nocturnas estudiando y tengo muy buenos recuerdos.

Sobre todo a mi familia por todo el cariño que he recibido por parte de ellos a lo largo de mi vida, y a mi hermano, por ser la persona más buena que conozco.

Por último, agradecerle al lector el tiempo que esta dedicando en leer estas líneas y se ha interesado por este proyecto.

Índice abreviado

Índice general

Lista de figuras

Lista de tablas

Capítulo 1: Introducción

Las interfaces aferentes son en la actualidad un instrumento eficaz para proporcionar una mejora en la autonomía personal de los individuos que padecen alguna lesión. Hoy en día, el número de personas que padece alguna lesión que le imposibilita llevar una vida normal, va en aumento, debido al progresivo envejecimiento de la población y a la presencia de enfermedades degenerativas en personas de todas las edades que las impiden realizar determinados movimientos. Por desgracia, estas causas, a veces naturales y otras no, imposibilitan al individuo para moverse, llegando incluso en algunos casos a dejarlo prácticamente inmóvil. Estas situaciones se producen por amputaciones, por accidentes que hayan sufrido, o incluso por enfermedades degenerativas que, poco a poco, van dejando sin movilidad a la persona que la padece. Esto hace que algunas de las tareas más corrientes resulten auténticos retos, haciendo muy difícil la vida diaria. Por ello, en el Laboratorio de Electrónica y Bioingeniería (LEB) de la Universidad de Valladolid, ubicado en la Escuela Técnica Superior de Ingenieros de Telecomunicación (ETSIT), se desarrollan prototipos destinados a aplicaciones de bioingeniería para conseguir mejorar la calidad de vida de aquellas personas que padecen alguna discapacidad. Estas investigaciones han cobrado una especial importancia para ayudar en la medida de lo posible a dicha población.

1.1 Tecnologías de Rehabilitación en el LEB y ámbito del proyecto

Los prototipos que se desarrollan en el área de las tecnologías de rehabilitación del Laboratorio de Electrónica y Bioingeniería (LEB) [1] están enfocados a las aplicaciones médicas y a la rehabilitación. Entre las investigaciones podemos destacar el desarrollo de las interfaces aferentes, dispositivos diseñados para captar señales de una determinada naturaleza y llevarlas hacia un sistema electrónico. Estos dispositivos pueden captar señales muy diferentes pudiendo distinguirlas según la señal que capten. Entre las investigaciones que se han realizado se pueden enumerar los siguientes.

1.1.1 Adquisición de señales biomecánicas

Este campo esta centrado en la captación de movimientos residuales en personas que tengan una fuerte discapacidad para que puedan controlar diferentes dispositivos en el ámbito de la robótica y la domótica. Uno de estos prototipos, es el reconocimiento de guiños mediante unas gafas que poseen unos sensores para captar las señales y poder controlar sistemas como una silla de ruedas eléctrica [2] o un robot de vigilancia [3]. Para reconocer estos guiños se han utilizado una variedad de sensores: Ópticos de reflexión [4], de detección de patrones y de vibración.

- Sensores ópticos de reflexión

Los sensores ópticos de reflexión empleados para el reconocimiento de guiños son los CNY70, dispositivos que incorporan un fotodiodo y un led infrarrojo para detectar la superficie que tienen enfrente. Estos dos componentes se colocan paralelos para que la luz infrarroja

emitida por el led se refleje en la superficie que tiene enfrente y la reciba el fotodiodo. Esta superficie consiste en una tira con dos colores (blanco y negro) que se le pega al usuario a cada lateral del ojo pudiendo detectar el guiño por el movimiento que se produce en el músculo orbicular al guiñar. De esta forma, en el momento del guiño, el sensor pasa de detectar un color a otro. En las siguientes figuras podemos observar donde se ubica el músculo orbicular y un esquema del sensor CNY70. En las figuras 1-1 y 1-2 podemos observar tanto la ubicación del músculo orbicular como un diagrama del funcionamiento del sensor CNY70.

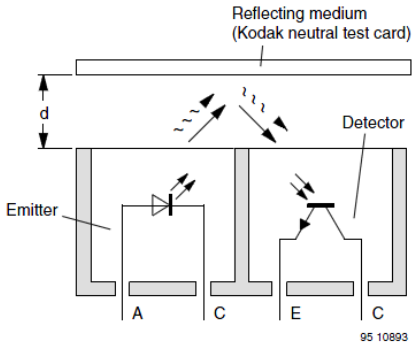
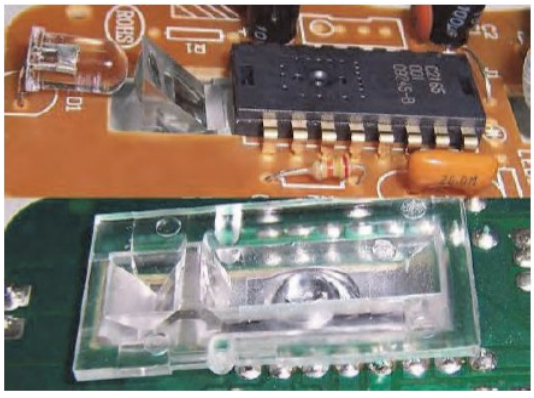


Figura 1-1 y 1-2: Músculo orbicular señalado por flechas [6] y CNY70 [5]

- Sensores de detección de patrones

Para detectar los guiños mediante este sensor se emplean los circuitos que se usan en los ratones ópticos para los ordenadores, la cámara óptica integrada, el LED y el prisma que dirige la luz (Figuras 1-3 y 1-4). Tiene incorporado un fotosensor que mediante un procesador digital de señales (DSP) se analiza el patrón observado en un momento concreto y se compara los obtenidos anteriormente para determinar el guiño. El procesado se realiza con una plataforma electrónica *Arduino*.



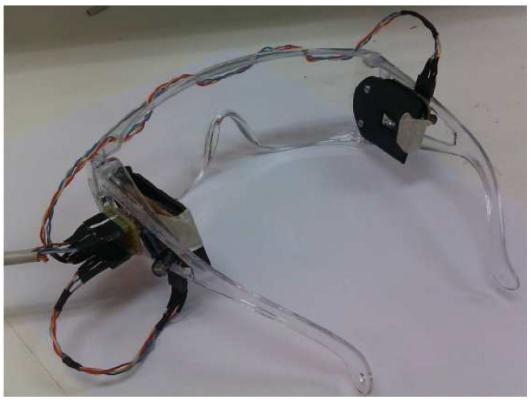


Figura 1-3 y 1-4: Circuito ratón óptico y ratones ópticos atados a unas gafas

La ventaja de este sensor es que no se necesita ninguna tira blanca y negra, simplemente basta con colocar el sensor en la posición correcta con la ayuda de unas gafas.

- Sensores de vibración[4]

Con estos sensores se detecta el guiño mediante la vibración del entorno ocular. Para ello se emplea un piezoeléctrico dimorfo que responde ante las deformaciones mecánicas mediante una variación de potencial. Para el procesado en este caso, será necesario aparte de la plataforma *Arduino*, un circuito de acondicionamiento empleando el circuito integrado 555 y un amplificador inverso, como puede verse en la figura 1-5.

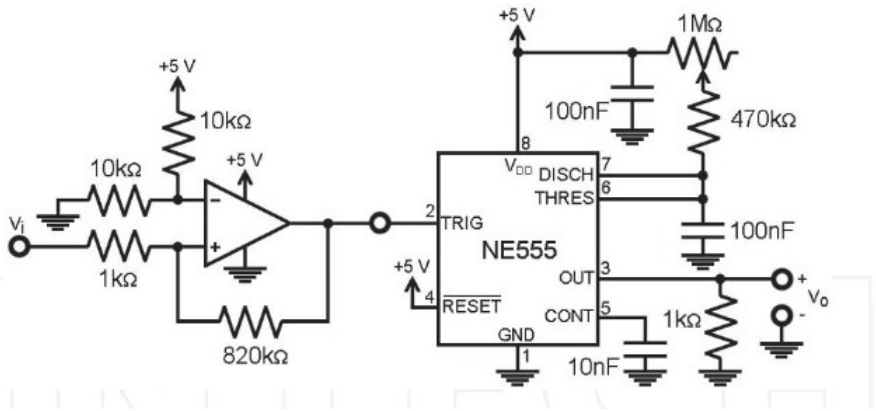


Figura 1-5: Circuito acondicionamiento

1.1.2 Adquisición de señales bioeléctricas

La interfaz de señales bioeléctricas permite recoger diferentes bipotenciales en el cuerpo humano. Además, también se puede utilizar para controlar elementos con el fin de realizar aplicaciones de domótica y sistemas de rehabilitación. Para ello se emplea una interfaz basada en electrodos y un sistema electrónico que sea capaz de recoger las variaciones que miden estos electrodos (Figura 1-6).

Figura 1-6: Electrodos

Éste sistema electrónico se basa en un amplificador de instrumentación que recoge la señal de los electrodos, un filtro paso alto y uno paso bajo para acomodar la señal al conversor analógico-digital, siendo éste el último bloque que digitaliza la señal.

El bloque ESD colocado a la entrada tiene el propósito de proteger al circuito electrónico contra sobrevoltajes que se pueden generar por electricidad estática, como se puede ver en la figura 1-7.

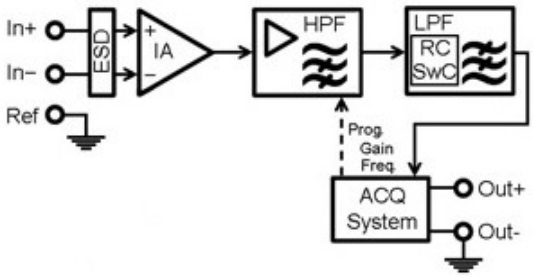


Figura 1-7: Sistema electrónico

1.1.3 Robot de vigilancia

Este robot de vigilancia (Figura 1-8) se controla mediante las gafas mencionadas anteriormente. De esta forma, el usuario que posea alguna discapacidad motora podrá desplazarse virtualmente por las dependencias de su hogar. También se ha incorporado un modo de manejar este robot de forma remota para que un amigo o familiar pueda mover el robot y ver como se encuentra el discapacitado.

El control del robot se realiza de forma inalámbrica y mediante un procesado el robot decodificará estas ordenes y realizará el movimiento.

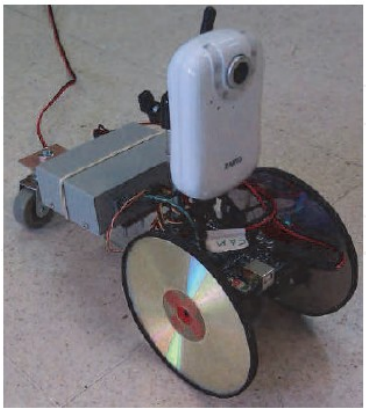


Figura 1-8: Robot vigilancia

Para que el usuario discapacitado pueda visualizar todo lo que filma el robot en tiempo real se usa una plataforma *Arduino* y la aplicación propia de la cámara inalámbrica basada en servicio Web (Figura 1-9). Al acceder a la pagina Web podemos ver en el centro lo que esta grabando el robot y a la izquierda una interfaz para controlar el robot de forma que un amigo o familiar pueda controlarlo si quiere ver el estado del discapacitado.

Figura 1-9: Servicio Web

1.1.4 Ámbito del proyecto

Otro de los proyectos que se ha realizado en el Laboratorio de Electrónica y Bioingeniería de la Universidad, es el de una silla de ruedas con motor eléctrico que se mueve utilizando interfaces aferentes de adquisición de señales biomecánicas que permiten al usuario tener un completo control de la silla mediante guiños voluntarios. En la figura 1-10 podemos ver los elementos de los que se compone todo el sistema de movilidad.

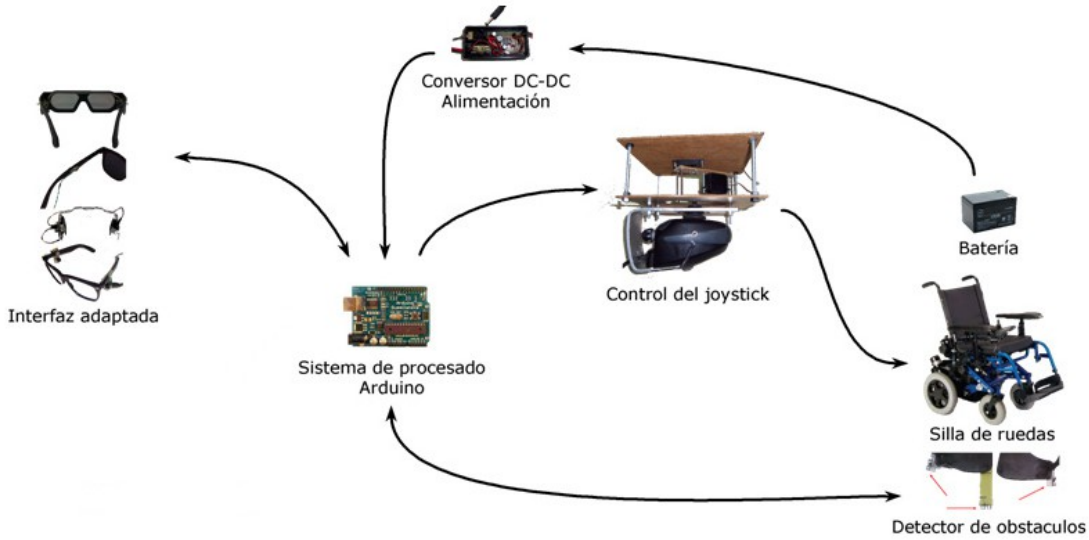


Figura 1-10: Elementos del sistema de movilidad

El sistema incorpora una plataforma *Arduino* que realiza el procesado, se encarga de la detección de obstáculos y gestiona las órdenes que se dan para el control de la silla. Para la detección de objetos se usan tres sensores de ultrasonidos que paran la silla en caso de colisión.

En cuanto al control del joystick, se emplea un sistema mecánico mediante servomotores que mueven el joystick según las ordenes que se reciban de la interfaz de entrada. Esta interfaz de entrada consiste en unas gafas que detectan los guiños voluntarios de forma que las órdenes que recibe la plataforma electrónica *Arduino* dependen de como se ejecuten estos guiños. En la tabla 1-1 se puede comprobar que instrucciones son necesarias para desplazar de la forma que se desee la silla.

INSTRUCCIÓN		MOVIMIENTO
1º Guiño	2º Guiño	
Izquierdo	Izquierdo	Avanzar
Izquierdo	Derecho	Giro a la derecha
Derecho	Izquierdo	Giro a la izquierda
Derecho	Derecho	Marcha atrás

Tabla 1-1: Instrucciones para realizar el movimiento

1.2 Objetivos

El objetivo de este proyecto es la realización de una interfaz de entrada distinta al sistema anterior de guiños para poder controlar el joystick que dirige la silla de ruedas mencionada en el apartado 1.1.4. Este joystick se mueve gracias a un mecanismo diseñado mediante una biela y dos servomotores que son controlados por la plataforma electrónica *Arduino*.

Por lo tanto, el fin de este proyecto es el diseño de una interfaz de entrada que consista en tres sensores de contacto (botones) colocados en los reposapiés con el objetivo de desplazar la silla, la elaboración de un mecanismo para controlar con dos servomotores un joystick de forma indirecta y una serie de pruebas que se realizarán con unos voluntarios para comprobar el correcto funcionamiento de la silla.

Esta interfaz de entrada se diseñó especialmente para un hombre que no poseía movilidad en las extremidades superiores ni inferiores, exceptuando unos leves movimientos que era capaz de realizar con los pies.

El proyecto esta destinado a todas aquellas personas que, por razones naturales o no, se ven en una situación en la que son incapaces de desplazarse, ya sea por ellos mismos o mediante una silla de ruedas. A través de este proyecto, es posible proporcionar un modo de controlar una silla de ruedas de motor eléctrico a aquellas personas que no podían desplazarse en una silla de ruedas controlándola con brazos o simplemente con un joystick.

1.3 Fases y Métodos

Para la realización de este proyecto dividimos el mismo para facilitar su organización.

1. Primeramente, se debe documentar y conseguir una serie de habilidades para la realización y desarrollo de este proyecto. A esta fase la denominaremos **fase de documentación** y consistirá en:

    - Estudiar el anterior proyecto que tengan relación con el trabajo que será llevado a cabo.
    - Adquirir las habilidades necesarias para el diseño de la nueva interfaz así como los conocimientos necesarios en programación Arduino.

2. **Fase de desarrollo**, esta fase se podrá dividir a su vez en diferentes grupos:

- PRIMERA FASE:

    - Desarrollo de la nueva interfaz de entrada.
    - Realización de pruebas con un voluntario para verificar la viabilidad de la silla y solucionar cualquier problema que se presente.

- SEGUNDA FASE:

    - Elaboración de un nuevo programa Arduino para optimizar su funcionamiento.
    - Diseño de un diagrama de flujo para visualizar de una forma mas sencilla todo el código.
    - Solución de los problemas que se presenten con éste nuevo código.
    - Realización de pruebas con estudiantes.

- TERCERA FASE:

    - Fabricación del mecanismo que controla la silla de forma indirecta mediante servomotores.

3. Para finalizar, **la fase de análisis y documentación de los resultados obtenidos**, constará de las siguientes fases:

    - Realización de pruebas básicas para comprobar el buen funcionamiento del sistema.
    - Realización de la memoria del proyecto.

1.4 Medios disponibles

Para la realización de este proyecto fue necesario la utilización de distintos medios, tanto software como hardware.

1.4.1 Medios Software

En la siguiente tabla (tabla 1-2) se pueden ver los distintos programas que se han utilizado para llevar a cabo dicho proyecto:

Nombre	Tipo	Uso
Arduino	Entorno desarrollo	Utilizado para la programación de la plataforma electrónica
Mozilla Firefox	Navegador Web	Usado para la documentación mediante Internet
Adobe Acrobat	Paquete Ofimática	Utilizado para convertir la memoria de formato Word a PDF.
Fritzing	Automatización de diseño electrónico	Utilizado para dibujar el circuito electrónico.

Tabla Software usado 1-2: Software usado

1.4.1.1 Entorno desarrollo Arduino

El lenguaje de programación *Arduino* se basa en C/C++. Para la programación de la plataforma electrónica *Arduino*, se proporciona un entorno de desarrollo, en la página Web [7].

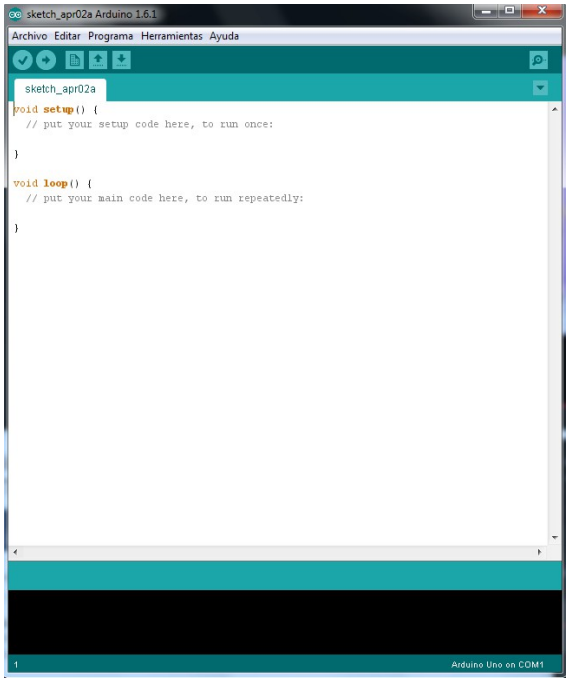


Figura 1-11: Entorno de desarrollo

El entorno de desarrollo *Arduino*, Figura 1-11, consta de un editor de texto para escribir el código deseado, una barra de herramientas y un área de mensajes. Este entorno de desarrollo permite la conexión con el hardware de *Arduino* para cargar los programas y comunicarse con ellos.

En el entorno de *Arduino* se utiliza la palabra '*sketch*' para dirigirse al programa. Estos *sketches* son escritos en el editor de texto. En el área de mensajes se muestra la información mientras los *sketches* son cargados y también se muestran los errores.

Los *sketches* se componen esencialmente de dos funciones básicas, la función *setup ()* y la función *loop ()*. Un *sketch* siempre tiene que contener estas dos funciones aunque no sea necesario definirlas. La función *setup ()* se ejecutará una sola vez cuando se conecte la placa de *Arduino* a una fuente de alimentación externa o se pulse el botón de Reset. Esta función se emplea para iniciar variables, librerías, establecer la configuración de los pines, etc. La función *loop ()* se ejecutará a modo de bucle durante todo el tiempo que este alimentada la plataforma o hasta que se pulse el botón Reset. En el caso de pulsarse, el programa volvería a correr la función *setup ()* para después recorrer en bucle la función *loop ()*.

1.4.1.2 Fritzing

Fritzing [8] se trata de un programa de automatización de diseño electrónico muy útil, tanto en el ámbito educativo como en el profesional, que permite crear prototipos en una placa virtual de pruebas generando el esquema del circuito. Este programa permite también agregar notas a los diseños.

En cuanto a la vista, puedes acceder a tres modos diferentes de vista:

- Vista protoboard: Permite visualizar el circuito que diseñes con sus componentes y conexiones de forma real.

- Vista de esquema: Muestra los componentes y conexiones del prototipo mediante símbolos.

- Vista de PCB: Es una representación de como quedarían los componentes y las conexiones en una placa de circuito impreso.

Además, puedes emplear una gran cantidad de elementos electrónicos de todo tipo. En la figura 1-12, podemos ver el diseño que tiene *Frytzing* para diseñar circuitos.

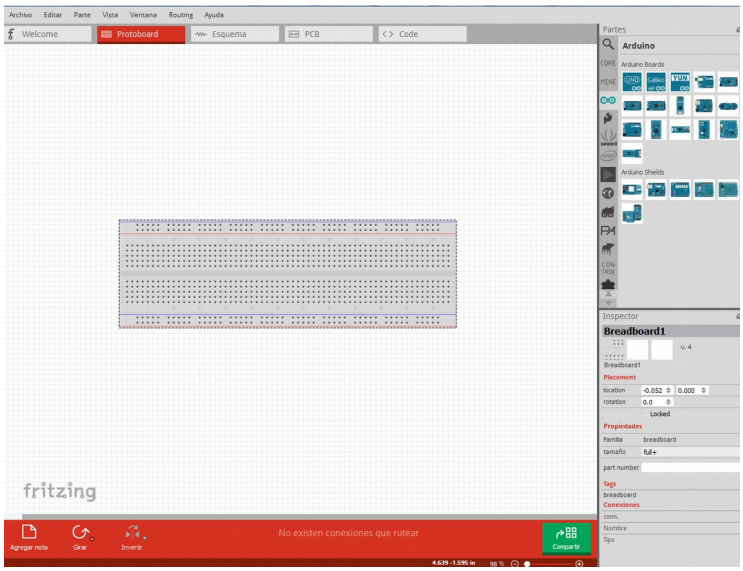


Figura 1-12: Fritzing

1.4.2 Medios Hardware

Para la realización del proyecto se ha usado la placa de *Arduino Duemilanove* (Figura 1-13). Arduino consiste en una plataforma electrónica abierta para la creación de prototipos basados en software o hardware. Puede obtener información de diferentes sensores por medio de algunos de sus pines, y puede actuar en consecuencia a ello con motores, PC, etc.

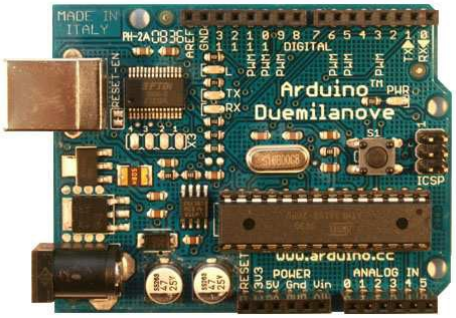


Figura 1-13: Arduino Duemilanove

Arduino [8] posee un microcontrolador, memoria, interfaces de entrada y salida y demás electrónica, todo ello concentrado en una misma placa para que pueda ser utilizado solamente, añadiendo los elementos que conforman el circuito que se desee realizar y desarrollando la programación necesaria para que ese circuito funcione. De esta forma, se consigue un ahorro bastante significativo en tiempo y en diseño.

Existen varias versiones de *Arduino*, aquí detallaremos algunos de ellos.

Tipos de *Arduino*:

• *Arduino UNO*: Se desarrolló en 2010, esta placa posee 14 pines de entrada y salida contiene un micro controlador *ATmega* 328.

• *Arduino nano*: Tiene el mismo funcionamiento que el *Arduino Duemilanove* que posee un microcontrolador *ATmega* 328 o un *ATmega* 168 dependiendo del modelo escogido. Posee 20 pines entrada salida.

• *Arduino Duemilanove*: El *Arduino Duemilanove* ("2009") es una placa electrónica basada en el *ATmega168* o *ATmega328*. Cuenta con 14 pines digitales de entrada / salida, 6 entradas analógicas, un oscilador de 16MHz, una conexión USB, un conector de alimentación, una cabecera ICSP, y un botón de reinicio.

• *Arduino pro mini*: Placa que tiene 22 pines de entrada salida (14 digitales y 8 analógicas) con microcontrolador *ATmega168* que posee un oscilador de cristal a 8MHz o 16 MHz (el que tiene los otros) según el modelo escogido.

En la siguiente tabla (tabla 1-3) se detallaran las características del *Arduino* empleado para este proyecto, el *Arduino Duemilanove*:

Microcontrolador	ATmega 168
Voltaje de funcionamiento	5V
Voltaje de entrada(recomendado)	7-12V
Voltaje de entrada(límite)	6-20V

Pines E/S digitales	14
Pines analógicos	6
Intensidad por pin	40mA
Intensidad por pin 3.3 V	50mA
Memoria Flash	16 KB (ATmega168); 2 KB para bootloader
SRAM	1 KB (ATmega168)
EEPROM	512 bytes (ATmega168)
Velocidad de reloj	16 MHz
Longitud	68.6 mm
Anchura	53.4 mm
Peso	25 g

Tabla 1-3: Características Arduino

Hay que aclarar también algunas características que posee el *Arduino* [8]:

-La memoria Flash: La memoria flash es el espacio que posee el *Arduino* para almacenar el propio programa que se le haya introducido y de los cuales, en el *Arduino Duemilanove*, usa 2KB para el bootloader.

-Bootloader (gestor de arranque): También llamado *firmware*, (debido a que casi nunca se modifica) se encarga de recibir nuestro programa de parte del entorno de desarrollo *Arduino* para más adelante almacenarlo correctamente en la memoria flash. Una vez realizado el proceso de grabación, el bootloader termina su ejecución y el microcontrolador dispone a procesar de inmediato y de forma permanente (mientras este encendido) las instrucciones recientemente grabadas.

-Memoria SRAM: Es el espacio donde los *sketches* almacenan y manipulan variables al ejecutarse. Ésta información se eliminará cuando el A*rduino* pierda la alimentación.

-Memoria EEPROM: Es un espacio para almacenar variables a largo plazo, incluso si el *Arduino* pierde la alimentación. Los programadores pueden usarla de ser necesario si quieren guardar alguna variable importante.

En *el Arduino Duemilanove* hay patillas que tienen funciones especiales:

• PWM: 3, 5, 6, 9, 10, y 11: Proporciona una salida PWM (Pulse Wave Modulación) de 8 bits de resolución.

• Pines 0 y 1: Para comunicaciones TTL. El pin 0 es para recibir y el 1 es para transmitir.

• I2C: Los pines analógicos 4 y 5, dan soporte al protocolo I2C.

• Pines 2 y 3 para interrupciones externas: Estos pines se configuran para lanzar una interrupción al micro controlador cuando la señal de entrada de estos pines cambie o valga un determinado valor.

1.4.3 Material de laboratorio

Para la realización del proyecto se ha hecho uso del material disponible en el laboratorio 2L011 de la ETS de ingenieros de Telecomunicación. Entre el material pueden destacarse los siguientes elementos:

- Osciloscopios.
- Fuentes de alimentación.
- Multímetros digitales.
- Taladro de columna.

1.5 Motivación del proyecto

Una vez que se han expuesto las investigaciones que se elaboran en el LEB y los medios hardware y software empleados para la realización de este proyecto, se presentará en primer lugar que es la discapacidad y se expondrán datos estadísticos relacionados con la discapacidad. Cabe decir, que muchas de las siguientes discapacidades que se comentarán a continuación, no disponen de soluciones tecnológicas para cubrir las necesidades de aquellas personas que las padecen. Ni tan siquiera una fase de desarrollo de prototipos. Además, a la hora de realizar estos nuevos dispositivos, es necesario tener en mente que el coste económico no debe ser muy elevado para posibilitar un uso de estas tecnologías a los usuarios finales. Por lo tanto, es necesario demostrar a la población, que gracias a la tecnología actual, es posible conseguir una mejora notable en la calidad de vida y en la autonomía de los discapacitados.

Para terminar, se explicarán las enfermedades degenerativas detallando el problema que conlleva para la persona que la padece.

1.5.1 La discapacidad

Según la organización mundial de la salud (OMS) [10], la discapacidad es un término general que abarca las deficiencias, las limitaciones de la actividad y las restricciones de la participación. Las deficiencias son problemas que afectan a una estructura o función corporal; las limitaciones de la actividad son dificultades para ejecutar acciones o tareas, y las restricciones de la participación son problemas para participar en situaciones vitales. A continuación se expondrán una serie de datos y cifras para hacer entrar en situación al lector de la situación actual de la población discapacitada.

Se calcula que más de mil millones de personas —es decir, un 15% de la población mundial— están aquejadas por la discapacidad en alguna forma. Tienen dificultades

importantes para funcionar entre 110 millones (2,2%) y 190 millones (3,8%) personas mayores de 15 años. Eso no es todo, pues las tasas de discapacidad están aumentando debido en parte al envejecimiento de la población y al aumento de las enfermedades crónicas.

Todas las personas con discapacidad tienen las mismas necesidades de salud que la población en general y, en consecuencia, necesitan tener acceso a los servicios corrientes de asistencia sanitaria. Pero a menudo, la vida estas personas se ve afectada por vulnerabilidades a enfermedades relacionadas con la edad, a afecciones secundarias, y una frecuencia mas elevada a una muerte prematura.

En el artículo 25 de la Convención sobre los derechos de las personas con discapacidad se reconoce que las personas con discapacidad tienen derecho a gozar del más alto nivel posible de salud sin discriminación.

1.5.2: La discapacidad en España

Según el boletín informativo del Instituto Nacional de Estadística [11], en 2008 había 3,85 millones de personas residentes en hogares que afirman tener discapacidad o limitación. Esto supone una tasa de 85.5 por mil habitantes.

El estudio de las características de la discapacidad se ha centrado en la población de 6 o más años, ya que para los menores el pronóstico de evolución es incierto y solo se analizan las limitaciones adaptadas a su edad. Para las personas de 6 o más años la tasa de discapacidad se sitúa en 89,7 por mil habitantes, como puede verse en la figura 1-14.

La discapacidad mas habitual es la movilidad, con casi el 67.2 % de las personas con limitaciones para moverse, el 55.3 % problemas para realizar tareas domésticas y el 48.4 % con las tareas del cuidado e higiene personal.

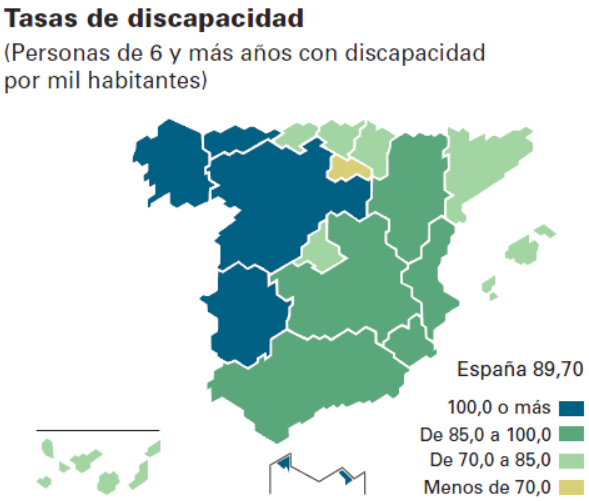


Figura 1-14: Mapa de España con personas discapacitadas

La discapacidad se puede dividir en varios grupos de discapacidad como puede ser la movilidad, visión, audición, comunicación, etc. En la figura 1-15 se muestra el porcentaje de personas de más de 6 años con algún tipo de discapacidad según su grupo de discapacidad.

Se hace notar, que una persona puede tener más de un tipo de discapacidad por lo que la suma no tiene que dar necesariamente 100%.

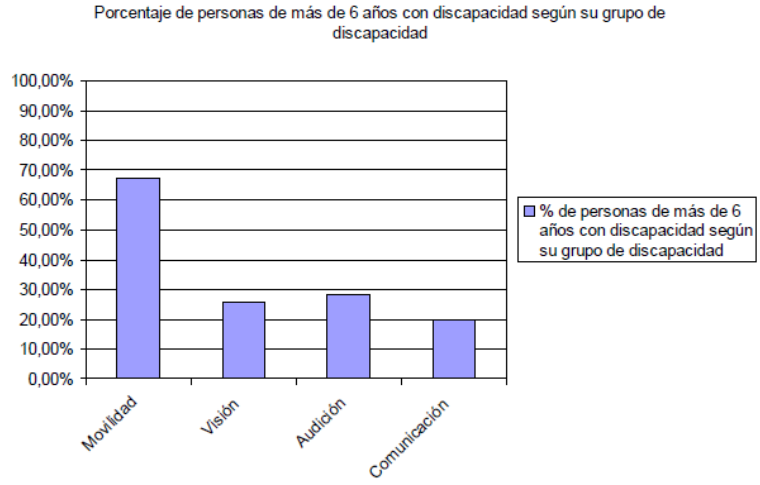


Figura 1-15: Porcentaje de personas con discapacidad

Se puede extraer claramente de la figura 1-15, que la mayor parte de las personas discapacitadas presenta problemas de movilidad, siendo con creces el mayor problema encontrado dentro de la población discapacitada.

1.5.3: Dispositivos y tecnologías de apoyo a las personas con discapacidad

Los dispositivos y las tecnologías de apoyo como sillas de ruedas, prótesis, ayudas para la movilidad, audífonos, dispositivos de ayuda visual y equipos y programas informáticos especializados aumentan la movilidad, la audición, la visión y las capacidades de comunicación.

Con la ayuda de estas tecnologías, las personas con discapacidad pueden mejorar sus habilidades y, por tanto, están más capacitados para vivir de forma autónoma y participar en la vida social.

Este tipo de tecnologías depende mucho del tipo de discapacidad que se posea [12]:

-Para las personas con discapacidad física (Figura 1-16), se usan mesas regulables en altura, teclas de gran tamaño, ratones virtuales o ergonómicos, etc.

Figura 1-16: Teclado especial para personas con discapacidad física

-Para personas con discapacidad visual (Figura 1-17), pantallas de gran formato, lectores de pantalla para invidentes, impresoras de braile, magnificadores de pantalla o lupas aumentativas, etc.

Figura 1-17: Teclado especial para personas con dificultad de visión

-Para personas con discapacidad auditiva (Figura 1-18): Intérpretes de lengua de signos, emisoras de frecuencia modulada, prótesis auditivas, etc.

Figura 1-18: Prótesis auditivas

Muchas de estas tecnologías sirven sobre todo para conseguir una adaptación del minusválido para el empleo, pero también hay muchas más que no solo sirven para el trabajo, sino para mejorar su calidad de vida.

A continuación se reflejan algunos ejemplos de respuestas que demuestran cómo las tecnologías han influido en un incremento de la calidad de vida de los encuestados:

- *"Gracias a la invención del audífono, puedo escuchar".*
- *"Las Nuevas Tecnologías me permiten comunicarme mucho mejor".*
- *"Ahora puedo escribir en el ordenador sin emplear las manos: sólo necesito la voz".*
- *"Tengo movilidad e independencia para desplazarme, gracias a mi vehículo adaptado".*
- *"Puedo acceder al ocio gracias a subtítulos y audífonos digitales. Antes era imposible".*
- *"Gracias a las Nuevas Tecnologías puedo estar informada y comunicada con el mundo exterior*

sin necesidad de desplazarme".
- "*Gracias a la prótesis que tengo en la columna, puedo moverme sin dolores*" .

Existen múltiples proyectos tecnológicos que han conseguido ayudar a muchas personas con diferentes signos de discapacidad para ayudarlos a ser más dependientes. Estas discapacidades pueden ser del tipo física, mental, auditiva o múltiple.

Todos los avances tecnológicos de ayuda conseguidos podrían agruparse en cinco grupos: [13]

- Sistemas alternativos y aumentativos de acceso a la información.
- Sistemas de acceso.
- Sistemas alternativos y aumentativos de comunicación.
- Sistemas de movilidad.
- Sistemas de control de entornos.

El primer grupo engloba toda tecnología destinada a ayudar a personas con discapacidad visual y/o auditiva. Entre ellas se pueden destacar las tecnologías del habla, rehabilitación cognitiva o comunicaciones avanzadas.

El segundo grupo corresponde a las tecnologías que habilitan a una persona que tiene alguna discapacidad física o sensorial para utilizar un ordenador. Podemos destacar en este grupo el sintetizador Braile, pizarras electrónicas, bastones digitales, pantallas táctiles, etc.

El tercer grupo engloba las tecnologías para ayudar a la persona que por su discapacidad no puede establecer una comunicación oral-verbal. Dentro de este grupo podemos distinguir la comunicación alternativa, que se basa en un modo de comunicarse cara a cara distinta al habla, y la comunicación aumentativa que es la comunicación de apoyo o ayuda.

Los sistemas de movilidad son dispositivos relacionados con la movilidad personal que se destinan a personas con discapacidad física. Ejemplos de estos sistemas podrían ser los brazos o soportes articulados, conmutadores adosados a sillas de ruedas etc.

Por último, los sistemas de control de entornos permiten la manipulación de dispositivos que como su propio nombre indica, ayudan a controlar el entorno. Se distinguen dos tipos: El control ambiental, que permiten a gente que tiene alguna discapacidad motora controlar algún dispositivo de uso doméstico, y la realidad virtual, dispositivos de entrada y salida como guantes sensitivos

1.5.4: Discapacidades físicas debido a enfermedades degenerativas

La discapacidad física [14] se puede definir como una desventaja, resultante de una imposibilidad que limita o impide el desempeño motor de la persona afectada. Esto significa que las partes afectadas son los brazos y/o las piernas.

Las causas de la discapacidad física muchas veces están relacionadas a problemas durante la gestación, a la condición de prematuro del bebé o a dificultades en el momento del nacimiento. También pueden ser causadas por lesión medular en consecuencia de accidentes (zambullido, por ejemplo) o problemas del organismo (derrame, por ejemplo).

Esta discapacidad también puede ser producida por enfermedades degenerativas, algunas de las cuales son:

- Ataxia de Friedreich: enfermedad hereditaria que ocasiona un daño progresivo del sistema nervioso con síntomas que van entre debilidad muscular y problemas de dicción, por un lado, y enfermedad cardiaca por otro. En general el primer síntoma que aparece es la dificultad para caminar y se va propagando progresivamente a los brazos y al tronco. Otros síntomas asociados son: pérdida de reflejos en rodillas, tobillos y muñecas, escoliosis, dolor de pecho, dificultad para respirar, palpitaciones, dificultad para hablar y en general suelen padecer profundas depresiones. El cerebro y la inteligencia no se ven alterados.

- Esclerosis múltiple: se trata de una enfermedad del sistema nervioso central que se produce cuando se destruye o deteriora la mielina perdiendo los nervios la capacidad de conducir los impulsos eléctricos. Los síntomas de esta enfermedad varían entre diferentes personas e incluso en un mismo individuo según los mementos.

- Distrofia muscular progresiva: se conoce a un conjunto de enfermedades, todas hereditarias, caracterizadas por una debilidad progresiva y un deterioro de los músculos esqueléticos o voluntarios que son los que se encargan del movimiento. La forma más frecuente y grave es la distrofia muscular de Duchenne, con una expectativa de vida de 20 años. Estas personas son muy sensibles a las lesiones por lo que hay que tener cuidado en los cambios de posiciones y los movimientos bruscos.

- Corea de Hungtitnton: también popularmente conocida como "Baile de San Vito". Es una enfermedad neurológica degenerativa caracterizada por movimientos involuntarios incontrolados, desarreglos psíquicos y pérdida de las funciones intelectuales (demencia).

Estos tipos de enfermedades tienen la particularidad de que en muchos casos, pueden controlar determinadas partes de su cuerpo como un movimiento parcial de rodillas o de pies pero aún así, son incapaces de ponerse de pie y de mover los brazos. Éstas personas no podrían controlar una silla de ruedas eléctrica convencional con los brazos, pero sí con los pies. De esta manera, con pequeños movimientos del pie, se pueden tocar unos botones colocados en los reposapiés, para que el minusválido pueda hacer uso de una silla de ruedas eléctrica.

1.6 Organización de la memoria

La memoria constará de cinco capítulos incluido esté. A continuación se detallan brevemente:

- En el capítulo 2 se hablará de como estaba la silla antes de empezar con este proyecto, las modificaciones que se realizaron en ella. Se continuará con la interfaz de entrada que se diseño en este proyecto y con el nuevo mecanismo elaborado para controlar el joystick de otra silla de ruedas con motor eléctrico.

- En el capítulo 3 se expondrá el diagrama de flujo realizado para comprender el primer programa desarrollado, el motivo de volver a modificar dicho programa

para obtener el máximo rendimiento de la silla y, como puede entenderse, el programa final. El diagrama de flujo final se puede encontrar en el apéndice de éste

- En el capítulo 4 se expondrán las pruebas y los tiempos que hicieron los alumnos de la Universidad al recorrer un circuito de pruebas y se comentará cuales fueron sus impresiones acerca de la silla.

- En el capítulo 5 se hará una conclusión general del proyecto y se expondrán algunas mejoras para el sistema.

1.7 Conclusiones

Al empezar este capítulo se habló de un modo general sobre las investigaciones realizadas en el LEB, el ámbito de éste proyecto, el objetivo a conseguir y los medios software y hardware utilizados.

Después se expuso la minusvalía en España y como la discapacidad física es la que mayor presencia tiene en nuestro país. También se explicaron diferentes tecnologías que han conseguido ayudar a las personas que padecen algún tipo de minusvalía.

Más adelante se comentaron las enfermedades degenerativas que imposibilitan al individuo a mover el cuerpo en su totalidad pero aún así, siendo incapaces de ponerse de pie, son capaces de mover ciertas partes de su cuerpo. Esto dio la idea de crear una interfaz con la que poder dirigir una silla de ruedas mediante los pies.

Capítulo 2: Hardware de la silla

En este capítulo se explicará el estado de la silla al principio del proyecto, cuando fue modificada por última vez por Juan José Ortega, [4] en su proyecto final de carrera. Se detallará cómo se modificó una silla convencional de motor eléctrico para que pudiera ser activada mediante guiños. Seguidamente, se expondrán los sensores de ultrasonido SRF-08 que llevaba la silla para evitar choques frontales y para dirigir la silla paralela a la pared. El mecanismo ideado para poder controlar el joystick con guiños, fue diseñado mediante dos servomotores de rotación 180 º y una biela.

Más adelante, se pasará a comentar las modificaciones realizadas para instalar la nueva interfaz de entrada diseñada específicamente para poder controlar la silla con los tobillos y finalmente, se realizó un mecanismo para poder accionar el joystick de otra silla de ruedas.

2.1 El guiño

En un principio, la silla estaba diseñada para poder ser controlada mediante guiños, que es un movimiento que no se suele perder a pesar de sufrir discapacidades físicas muy severas. El mecanismo que se ideó para ello fue el de colocar en unas gafas unos sensores de infrarrojos, concretamente, se utilizó el sensor de reflexión MSE S110.2, como puede verse en la figura 2-1. [15]

2.1.1 El sensor de reflexión MSE S110.2

El funcionamiento del sensor MSE S110.2 (Figura 2-1) se basa en dos sensores CNY70 (Figura 2-2). Cada dispositivo se compone de un diodo emisor de luz infrarroja (transmisor) y un fototransistor (receptor). Cuando la luz emitida por el diodo infrarrojo es absorbida por una superficie oscura, el fototransistor recoge este reflejo y la salida del sensor, convenientemente acondicionada, genera una señal lógica de nivel “1”. En el caso de reflejarse sobre una superficie clara, se genera por la salida una señal de nivel “0”.

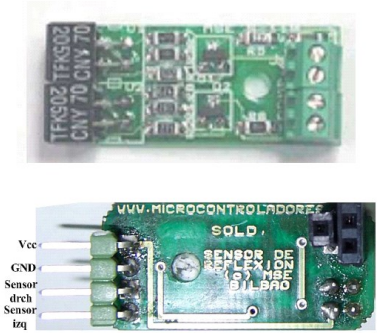


Figura 2-1: Sensor MSE S110.2

Este sensor posee cuatro conexiones, las conexiones 1 y 2 se reservan para alimentación y tierra mientras que las conexiones 3 y 4 corresponden a las salidas lógicas: Una salida por cada dispositivo CNY70.

Figura 2-2: CNY70

2.1.2 Las gafas

Para poder captar los guiños, se utilizaron unas gafas como soporte, para colocar los sensores de reflexión en las patillas. Se fijaron dos tiras de color blanco y negro en cada lateral de la cara del usuario, como se muestra en la figura 2-3, de tal forma que sin producirse ningún guiño, el sensor detecte siempre el color blanco. Al producirse un guiño voluntario, el sensor de reflexión detectará el color negro, consiguiendo captar el guiño. Esto produce una variación de tensión, la cual, se aprovecha para indicar a la plataforma *Arduino* que el ojo ha sido guiñado.

Esta placa de tiras además se sujeta en una de las patillas de las gafas mediante un tornillo y en la otra patilla se coloca el sensor MSE S110.2. En la placa de tiras se soldaron conectores de tipo hembra para luego unir las dos placas mediante un cable.

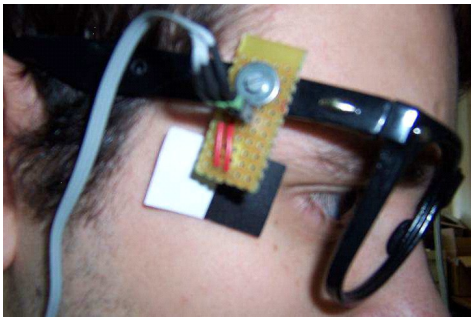


Figura 2-3: Colocación de los sensores MSE S110.2

Cabe decir, que la colocación de las tiras es fundamental, porque una mala sujeción de éstas, dejaría a los sensores inservibles. Este mecanismo presenta una desventaja y es el uso de las tiras. Es imprescindible la correcta colocación de las mismas y esto supone una dificultad para el usuario de la silla.

Otro problema encontrado es el de la disposición de las gafas, ya que si no se adaptan bien a la forma de la cara, pueden moverse de su correcta posición de tal forma que tras un tiempo de uso se pierda la funcionalidad de los sensores.

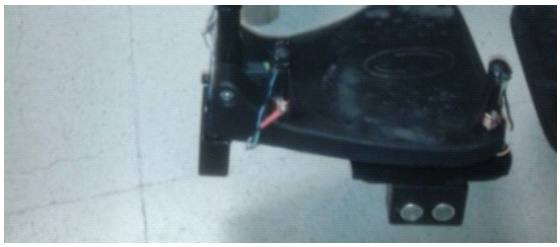
Tras unas pruebas realizadas con un paciente, se pudo comprobar que este sistema no funcionaba correctamente por lo que se pensó en desarrollar otra interfaz que fuera apta para

él y que pudiera controlar sin ninguna dificultad. De ahí surgió la idea que se presentará en el apartado 2.4.

2.2 Sistemas de detección de obstáculos

Para evitar cualquier tipo de choque, se incorporó a la silla un sistema de detección de obstáculos mediante sensores de ultrasonidos con el que disminuía la velocidad de la silla en caso de detectarse un objeto cercano, llegando a pararla de ser necesario.

Este sistema está compuesto por tres sensores, dos laterales para detectar los posibles objetos con los que colisionar al realizar un giro y otro frontal, para detectar objetos frontales. Los sensores laterales además permitían otra función: El auto-guiado.



Figuras 2-4 y 2-5: Sensores de ultrasonido colocados en los reposapiés

La fijación de los tres sensores se llevó a cabo mediante un sistema de adhesión a la silla por medio de velcros para sujetarlos en la parte inferior de cada reposapiés de la silla, como se puede ver en las figuras 2-4 y 2-5. Para comprobar si hay riesgo de colisión no se mide la distancia que existe entre el sensor y el obstáculo, sino que se realiza un algoritmo para determinar el tiempo restante para que se produzca la colisión a la velocidad que lleva la silla en ese momento. Con la velocidad de la silla, y la distancia que existe entre el sensor y el obstáculo, se puede calcular el tiempo que queda para que se produzca la colisión. Para el sensor central, la silla va disminuyendo su velocidad si el tiempo que existe para la colisión es menor a 2 segundos. Para los sensores laterales el tiempo es de 3 segundos.

En cuanto a la activación del sistema de sensores, se realizaba guiñando uno de los dos ojos. Una vez realizado, los sensores quedaban activados funcionando con el algoritmo anteriormente explicado. Cuando uno de los sensores laterales detectaba un objeto mientras se avanzaba, el sistema ponía en marcha el auto-guiado de la silla dirigiéndola en dirección paralela al objeto.

2.2.1 Sensores de ultrasonido SRF-08

El sensor SRF-08 (Figura 2-6) [17] es un medidor ultrasónico que posee dos cápsulas ultrasónicas de 40 KHz, una célula LDR para proporcionar una medida de luz y un microcontrolador PIC16F872 que realiza las funciones de control e interfaz.

Figura 2-6: Sensor ultrasónico SRF08

Las características principales de este sensor son:

- La distancia que es capaz de medir, hasta 11 metros.
- Se controla desde un bus I2C.
- La medida puede darse en cm, pulgadas, o microsegundos.
- El Led que tiene en la parte posterior expresa la dirección del bus I2C.

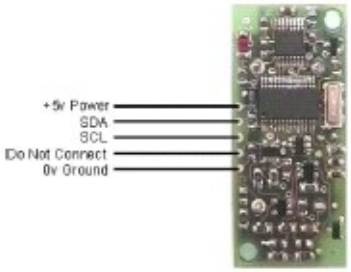
Para interconectar más sensores SRF-08 en el mismo bus, es necesario cambiar la dirección I2C del bus que viene por defecto y tener conectado solo un sensor SRF08 al bus. Después, se debe escribir la siguiente secuencia en orden: 0XA0, 0XAA, 0XA5, (dirección deseada). En el momento de probar estos sensores se comprobó que para enviar un comando se utiliza la dirección del trabajo pero para cambiar la dirección se usa la dirección de cambio, ambas direcciones se muestran en la tabla 2-1.

Dirección para el cambio	Dirección de trabajo	Descripción
0xF8	0xF0	Sensor izquierda
0xF9	0xF2	Sensor centro
0xFA	0xF4	Sensor derecha

Tabla 2-: Dirección para el cambio y para el trabajo

Por lo tanto, si se quiere cambiar la dirección al sensor izquierda por ejemplo, se escribirá lo siguiente: 0XA0, 0XAA, 0XA5, 0XF8. Si se quiere cambiar al sensor central o al sensor de la derecha basta con cambiar el código 0XF8 mencionado antes por el 0XF9 y 0XFA, respectivamente.

En las siguientes figuras (Figura 2-7 y figura 2-8) viene reflejado las conexiones del sensor SRF08 y su ángulo de detección cónica que abarca 55°.



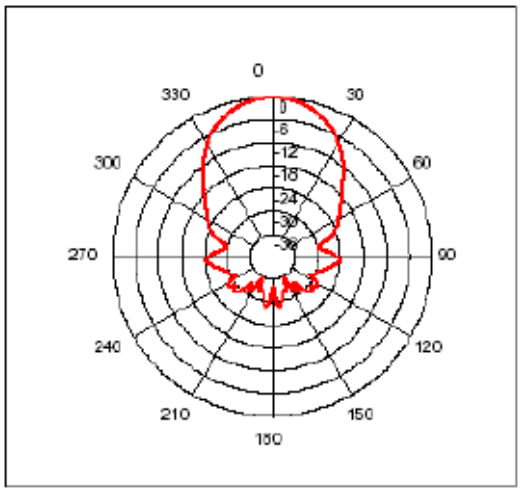


Figura 2-7 y 2-8: Conexiones y ángulo de detección del sensor SRF08.

2.3 La plataforma Arduino y el joystick

La plataforma electrónica *Arduino* elegida para este proyecto es la *Duemilanove*. Esta plataforma electrónica [2] consigue recibir las variaciones de tensión provenientes de los sensores de reflexión (las gafas) y a partir de ahí, controlar el joystick, gracias a un mecanismo diseñado mediante unos servomotores. Por lo tanto, serán los servomotores los responsables del movimiento del joystick y por consiguiente, de la silla.

Para realizar esta tarea, se consiguió una silla de ruedas convencional que poseía un motor eléctrico para desplazarla pudiendo ser controlada por el ocupante mediante un joystick situado en el apoya brazos. Pero para el objetivo del proyecto, había que cambiar la forma de controlar el joystick, puesto que la silla está destinada a personas sin movilidad en las extremidades superiores. Para lograr esto, se modificó esta parte y se cambió la posición del joystick para que pudiera ser controlada mediante guiños en lugar de usar las manos.

Para esta parte, se pensó en una primera opción: Sustituir las señales generadas por el joystick por las mismas señales pero generadas por el sistema de guiños. Las señales del joystick se generan gracias a cuatro bobinas fijas y una móvil que se acciona de acuerdo al movimiento del joystick, es decir, al moverlo (el joystick), esta bobina móvil induce señales a las demás bobinas [18] pero esta opción se descartó puesto que reduce considerablemente la vida útil del dispositivo y la tarea de introducir señales externas es muy delicada.

Debido a esto, se optó por la segunda opción viable: Elaborar un mecanismo mediante dos servomotores controlados por la plataforma *Arduino*, que controlen el joystick y así producir el movimiento de la silla. El mecanismo que se ideó se muestra en la figura 2-9.

Figura 2-9: Joystick de la silla

Para poder desplazar el joystick en todas las direcciones se necesitan dos servomotores: Uno que realiza el movimiento de rotación, que es el que se encuentra atornillado a la placa superior en la figura 2-9 y otro de empuje, que se mueve de forma lineal. Como estos dos servomotores proporcionan solo un movimiento de rotación, se empleó una biela (Figura 2-10) para proporcionar el movimiento de empuje que esta enganchada entre el aspa del servomotor y el joystick.

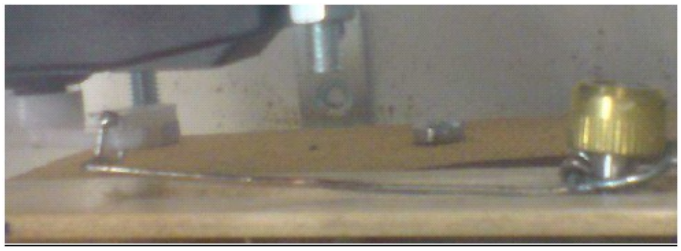


Figura 2-10: Biela para proporcionar movimiento lineal

Como puede verse en la figura 2-10, la biela consiste en un alambre que se engancha por un lado, al aspa del servomotor (Figura 2-11) y por otro lado, al joystick de la silla que tiene además una tuerca dorada para impedir que esta biela se desenganche.

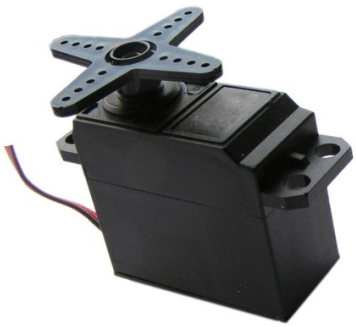


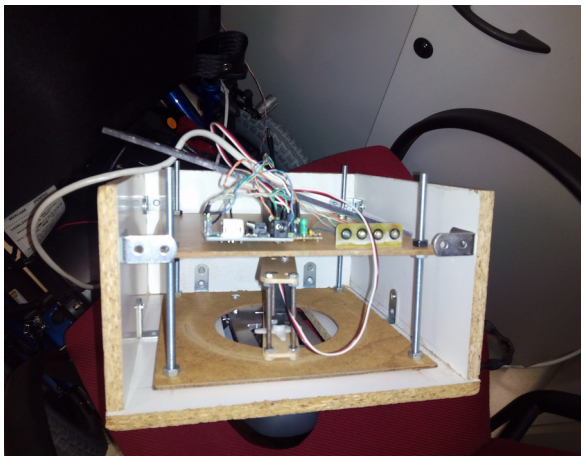
Figura 2-11: Servomotor empleado

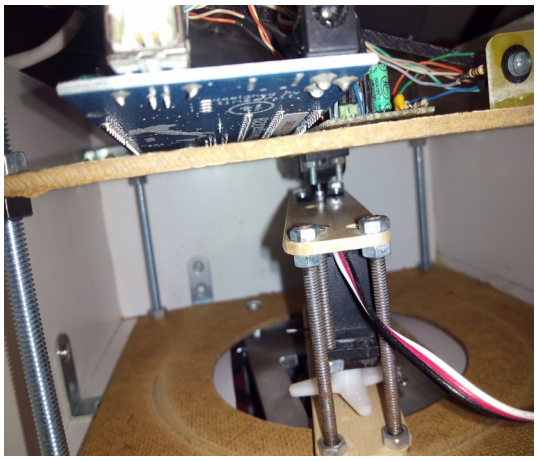
En vista de que la silla pertenece a la Universidad, se realizó otro mecanismo para controlar el joystick, parecido al anterior, pero para poder ser instalado en otra silla.

Por ello, se expondrán los pasos a seguir mediante imágenes y como se creó el mecanismo desde cero hasta su puesta en funcionamiento.

2.3.1 Construcción del mecanismo

Antes de empezar a realizar el mecanismo para controlar el joystick, se realizaron las medidas que tenía el antiguo mecanismo, como puede verse en las figuras 2-12 y 2-13. En las figuras, puede verse como el mecanismo se encuentra entre dos placas que están sujetas por cuatro varillas situadas en los vértices de las placas. Justo en el centro, se encuentran los dos servomotores que serán los encargados de controlar el joystick que desplazará la silla.





Figuras 2-12 y 2-13: Mecanismo completo de la silla

Para realizar esta operación, se tuvo que quitar la placa que tiene en la parte superior para poder hacer todas las medidas necesarias. Una vez realizadas las medidas, se comenzó por trazar una semi-circunferencia a una de las placas y cortarla dejando al joystick libertad suficiente para realizar todos los movimientos posibles. Después, en la otra placa, se trazó un rectángulo de las mismas medidas que tiene uno de los servomotores y se atornillo a ella para que estuviera sujeta (Figura 2-14). A continuación se cortaron dos tablas de madera. Estas tablas contendrán el otro servomotor que irá atornillado a estas y será el encargado de realizar la acción de empuje del joystick. También se calcularon las posiciones donde se debía agujerear la placa para colocar las varillas. La longitud de las varillas definirá la altura de la caja resultante (Figura 2-15). El servomotor atornillado a una de las placas será el responsable de los giros.

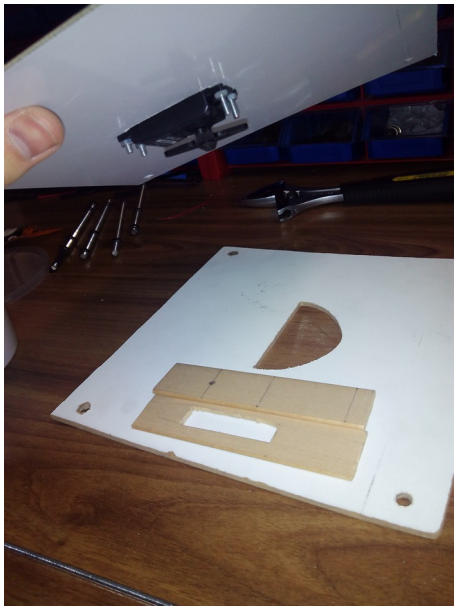


Figura 2-14: Semicircunferencia realizada

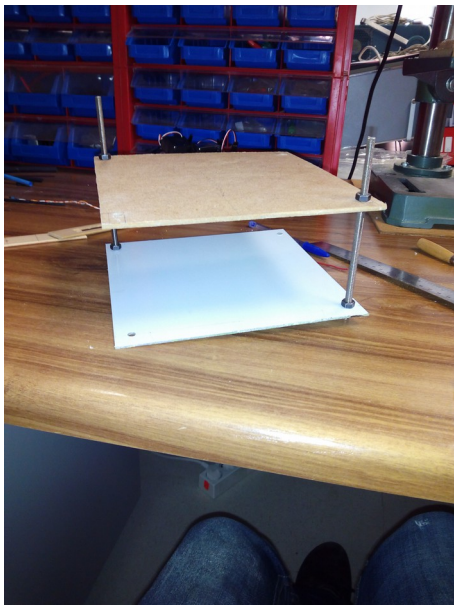


Figura 2-15: Forma de la caja

Una vez atornillado el servomotor encargado de los giros (Figura 2-14) a una de las placas, se continúo colocando el servomotor responsable del empuje entre dos tablas de madera (Figura 2-17). En un extremo irá este servomotor y en el otro, el servomotor encargado del giro (Figura 2-16). En la figura 2-16, el servomotor responsable del giro se encuentra enganchado a una de las tablas de madera a las que esta adherido mediante varillas al servomotor de empuje.

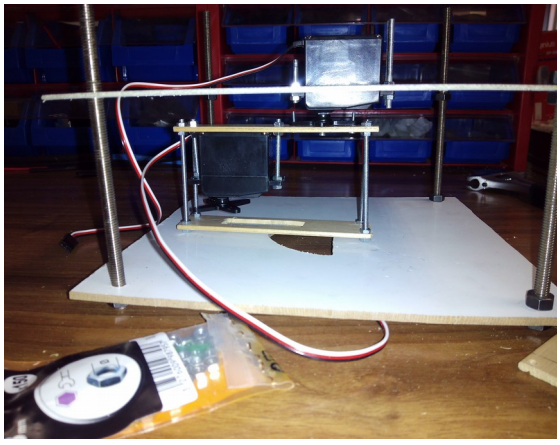


Figura 2-16: Mecanismo final

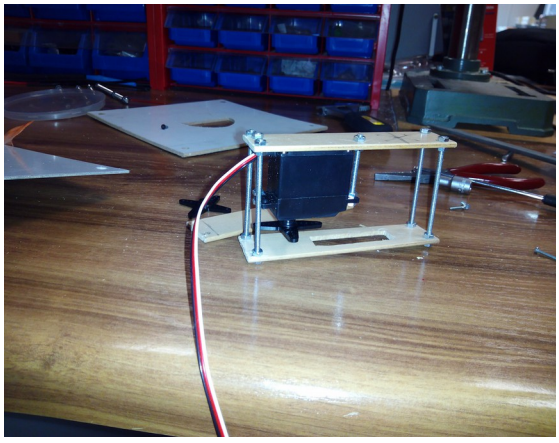
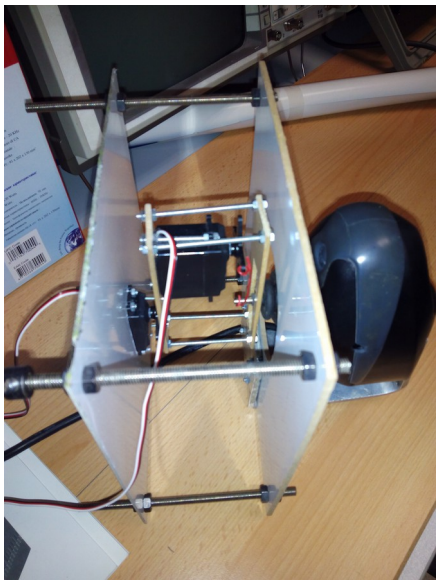


Figura 2-17: Servomotor de empuje

Cuando se tuvo la estructura hecha, solo faltaba colocar el joystick justo en el centro de la semicircunferencia y mediante un alambre enganchado al aspa del servomotor de empuje por un extremo y por el otro al joystick, poder controlarlo mediante los sensores de contacto que se encuentran en los reposapiés de la silla. En vista de que esta semicircunferencia era demasiado pequeña como para que el joystick tuviera libertad suficiente para realizar todos los movimientos, se cortó más placa para tener como resultado una circunferencia. Los dos servomotores se conectaron a un circuito realizado en una protoboard para comprobar su

funcionalidad. Este circuito se explica en el apartado 2.4.1. En las figuras 2-18 y 2-19 se muestra el dispositivo completo con los dos servomotores y la biela enganchada al joystick. En la figura 2-20 se puede apreciar un alambre enganchado entre el aspa del servomotor de empuje y el joystick.



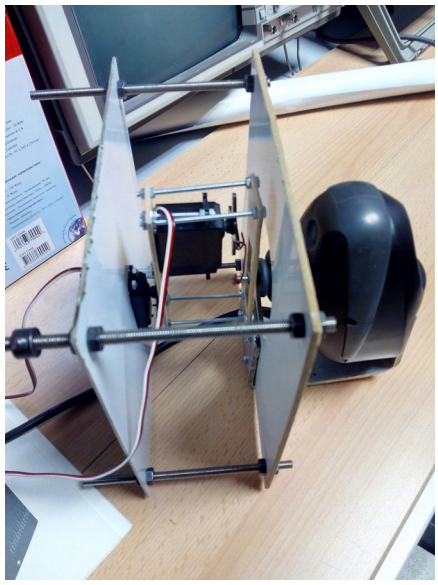


Figura 2-18 y 2-19: Dispositivo completo

Figura 2-20: Alambre enganchado al joystick

Una vez explicado la elaboración del nuevo mecanismo, se expondrá la nueva interfaz de entrada que se realizó como novedad en este proyecto.

2.4 Nueva interfaz de entrada

Esta nueva interfaz de entrada se ideó para que pudiera ser usada por Justo, un paciente que padecía una enfermedad degenerativa que le incapacitaba de extremidades superiores e inferiores pero aún así, era capaz de realizar movimientos con los tobillos.

La interfaz consiste en tres botones situados en los dos reposapiés de la silla de ruedas para poder activarlos mediante los tobillos. Se han colocado dos botones en dos pivotes del reposapiés derecho y el otro botón en otro pivote del reposapiés izquierdo. En estas imágenes se muestra como están colocados.

Figura 2-21 y 2-22: Reposapiés derecho e izquierdo con los botones para controlar la silla

Figura 2-23: Imagen del sensor de contacto

Como se puede ver en las figuras 2-21,2-22 y 2-23, se ha enrollado cobre en la parte inferior de un pivote que ya tenia la silla y se ha colocado una lámina de aluminio, sacada de una cinta VHS, en la parte superior del pivote, de tal forma que al mover el tobillo en dirección al pivote, hagan contacto la lámina de aluminio con el cobre enrollado, cerrando un circuito que se describirá a continuación.

Se colocaron leds en la caja de protección donde irá el circuito para comprobar cuando se activaba cada uno de los sensores de contacto.

Antes de diseñar esta interfaz, hay que entender que la silla estaba hecha para ser controlada mediante guiños, y el movimiento de parada estaba programado para que fuera con un parpadeo muy fuerte. Para realizar el movimiento de parada con los dos botones, habría que pulsarlos al mismo tiempo, algo imposible si los botones están en el mismo reposapiés. Al pensar en esta forma de dirigir la silla, se diseño un tercer botón en el otro reposapiés, que tuviera la función de parada exclusivamente. Para hacer esto, se utilizó un codificador que se detallará en este apartado.

En la siguiente tabla (tabla 2-2), se vuelve a recordar el modo de dirigir la silla mediante el anterior sistema de guiños.

Guiños		Movimiento
Primer guiño	Segundo guiño	
Izquierdo	Izquierdo	Movimiento hacia delante
Izquierdo	Derecho	Giro derecha
Derecho	Izquierdo	Giro izquierdo
Derecho	Derecho	Atrás
Parpadeo voluntario fuerte		Parada

Tabla 2-2: Posibles movimientos con el sistema de guiños

Aparte de los posibles movimientos descritos en la tabla 2-2, si estas desplazándote hacia delante, puedes activar un modo especial: Virar la silla. Para virar la silla a voluntad, se activa primero el modo de avance y a continuación, se guiña uno de los dos ojos. Una vez guiñado, tan solo tienes que guiñar el ojo derecho para virar a la derecha, o el ojo izquierdo si quieres virar a la izquierda.

Cuando se guiña el ojo izquierdo con las gafas puestas, el pin 2 de la plataforma electrónica *Arduino* recibe 0 voltios, momento en el cual la plataforma electrónica Arduino interpreta un guiño izquierdo, funcionando de la misma forma con el ojo derecho (pin 3). Con los botones funciona de forma similar, el botón izquierdo está conectado al pin 2 del *Arduino* y el pin 3 del *Arduino* al botón derecho.

En resumen, si se quiere desplazar la silla hacia delante, se tendrá que pulsar dos veces el botón izquierdo, lo que hará llegar 0 voltios al pin 2, de tal forma que el código del *Arduino* interpretará este comando y los servomotores activarán el joystick para desplazar la silla hacia delante. Para girar a la izquierda, se pulsa el botón derecho y después el izquierdo, para girar a

la derecha, el botón izquierdo y luego el derecho y para dar marcha atrás, dos veces el botón derecho.

Ahora bien, para realizar la parada, no se puede pulsar con el mismo pie los dos botones, por lo que habrá que instalar un tercer botón en el otro reposapiés y conseguir que al pulsarlo, le lleguen 0 voltios tanto al pin 2 como al pin 3. Para esto, se usa el codificador N74LS148 [19].

2.4.1 Codificador N74LS148

El codificador N74LS148 es un circuito integrado de 16 patillas [20] que permite codificar hasta 8 líneas de entrada en 3 líneas de salida, como se puede ver en la tabla 2-3.

Para entender mejor esta parte se recurre al programa *Fritzyng* que permitirá dibujar el circuito con elementos electrónicos.

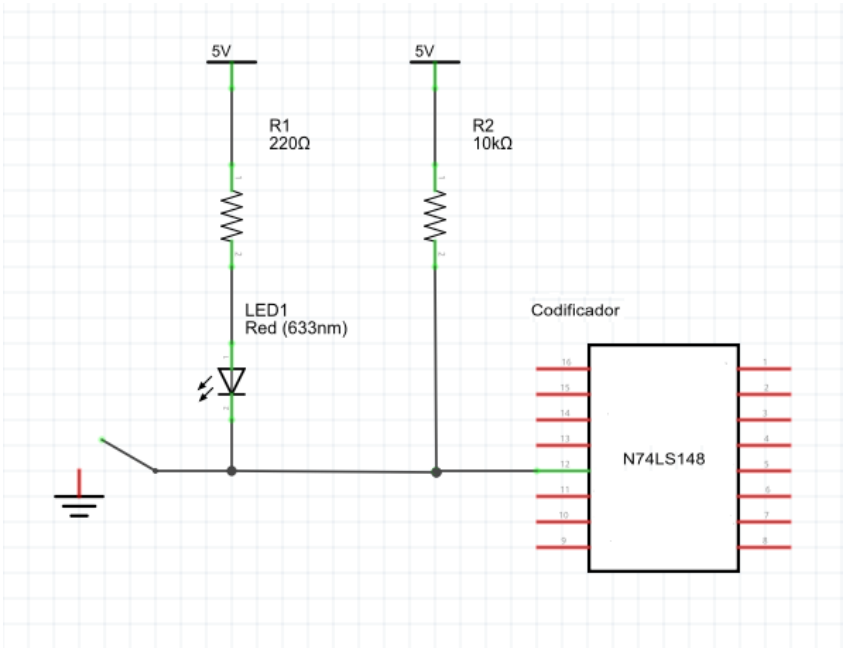


Figura 2-24: Esquema de conexión

La parte que esta en circuito abierto de la figura 2-24 correspondería al botón que se explicó con anterioridad, de forma que al pulsar este botón, el circuito se cerrará, lucirá el LED y la patilla del codificador recibirá 0 voltios.

En la figura anterior, la nota "*Codificador*" indica la conexión que irá conectado a una patilla del codificador que se explicará mas adelante, pero claro, ¿Por qué se quiere tener 0 voltios (L en la tabla 2-3) por una patilla del codificador cuando se pulse el botón? Esto se explica mejor mostrando la tabla del codificador empleado, el N74LS148.

SN54/74LS148
SN54/74LS748
FUNCTION TABLE

INPUTS									OUTPUTS				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

Tabla 2-3: Tabla de verdad del codificador N74LS148

Según esta tabla, si conectamos el pin 1 del codificador al botón de la derecha del reposapiés derecho, el pin 2 al botón de la izquierda del reposapiés derecho, y el pin 3 al botón de paro, se conseguiría obtener por las salidas A1 y A0 los valores deseados. Si la salida A1 se conecta al pin 2 del *Arduino Duemilanove* (el botón izquierdo) y A0 al pin 3 (el botón derecho), se obtienen las siguientes posibilidades:

INPUTS			OUTPUTS	
1(Botón derecho)	2(Botón izquierdo)	3(Botón parada)	A1	A0
X	X	L	L	L
X	L	H	L	H
L	X	X	H	L

Tabla 2-4: Entradas y salidas utilizadas del codificador

Según la tabla 2-4, al pulsar el botón 3, que es el botón de parada, (Pulsar el botón equivale a tener una entrada LOW que en la tabla viene indicada como L) se consigue una tensión LOW en las salidas A1 y A0 del codificador, que llega a los pines 2 y 3 del *Arduino* haciendo que la silla se pare en caso de estar en movimiento. En el caso de pulsar el botón 2, se obtendrá una señal LOW por la salida A1 que recibirá el pin 2 del *Arduino* contabilizándose una pulsación del botón izquierdo. Si se pulsa el botón 1, aparecerá una tensión LOW por la salida A0 del codificador y al estar conectado al pin 3 del *Arduino*, se anotará una pulsación del botón derecho.

A1	A0	Botones	Reposapiés
L	L	Parada	Izquierdo
L	H	Izquierdo	Derecho
H	L	Derecho	Derecho

Tabla 2-5: Función de cada botón con su localización

En la tabla 2-5, podemos ver que se consigue al pulsar cada uno de los tres botones. En el caso de apretar el botón de parada, que se encuentra en el reposapiés izquierdo, se generan dos tensiones LOW por las dos salidas del codificador. Al presionar el botón izquierdo, se produce una tensión LOW por la salida del codificador A1 y de pulsar el botón derecho, se origina una tensión LOW por la salida A0. Para obtener por salidas del codificador lo expuesto en la tabla anterior, hay que realizar una serie de conexiones a mayores. Las salidas "*GS*" y "*EO*" hay que conectarlas a tierra y a tensión, respectivamente y la entrada "*EI*" a tierra. La función de cada patilla puede verse en la figura 2-25.

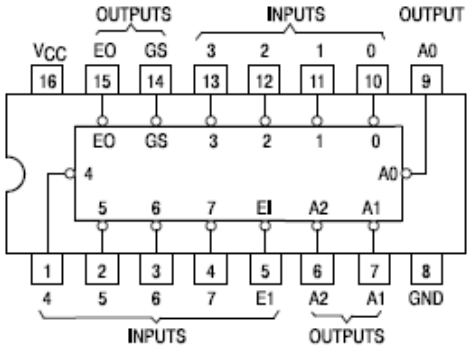
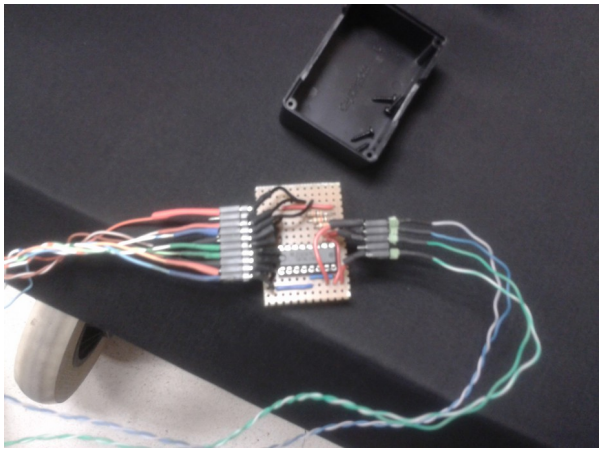
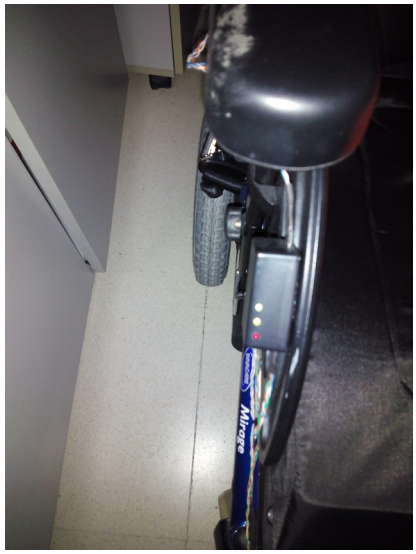


Figura 2-25: Patillas del codificador N74LS148

Una vez entendido este procedimiento, se empezó a realizar el circuito y a colocarlo dentro de una caja de protección como muestra la figura 2-26 y 2-27.





Figuras 2-26 y 2-27: Circuito final con su correspondiente caja

2.5 Conclusiones

En este capítulo se expuesto el hardware completo de la silla. Se ha explicado el sistema de los guiños mediante los sensores de reflexión y los sensores de ultrasonido, hasta su puesta en funcionamiento con la nueva interfaz instalada para su manejo con los tobillos. Se ha aclarado también, como se realizó el nuevo mecanismo de control del joystick para ser instalado en otra silla de ruedas con motor eléctrico

Una vez realizado todo el diseño hardware, se detallará la parte software de la silla, explicando su funcionamiento con diagramas de flujo y exponiendo gran parte del código que posee la plataforma *Arduino*.

Autor: Alvaro Riol Triviño

Capítulo 3: Software de la silla

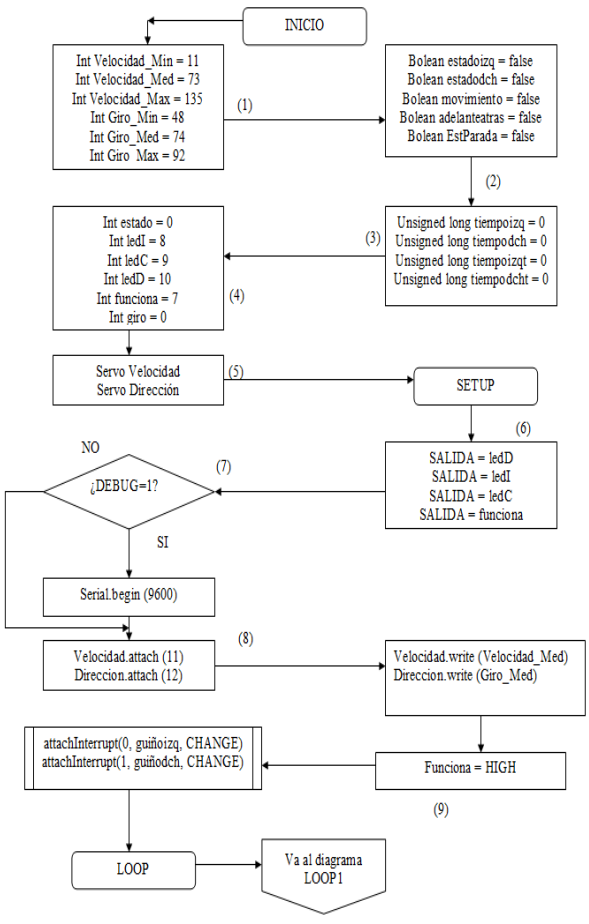
El capítulo 3: Software de la silla, refleja todo el trabajo realizado respecto a la programación *Arduino* para mejorar el rendimiento del sistema (desactivando los sensores de ultrasonido). El esquema que va a seguir este capítulo es el siguiente: en primer lugar, se expondrán los diferentes diagramas de flujo que se desarrollaron al comienzo para visualizar el programa y comprender qué parte debía modificarse; en segundo lugar, se comentará el código que se modificó al comienzo paso a paso, con las líneas comentadas, para posibilitar en la mayor medida su comprensión; en tercer lugar, se explicará el motivo del fallo que tenía este programa, puesto que tras un tiempo de funcionamiento, el sistema se bloqueaba, dejando a los botones de los reposapiés sin utilidad. Para solucionar este problema se realizaron unas modificaciones, las cuáles, también se explicarán en este capítulo.

3.1 Diagrama de flujo y programa Arduino

Antes de empezar a modificar el programa *Arduino* se realizó un diagrama de flujo para visualizar el programa con los sensores de ultrasonido desactivados.

3.1.1 Diagrama de flujo

Una vez explicado en qué consiste el programa *Arduino* en su conjunto (ver capítulo 1.4), se expondrá el diagrama de flujo con su código. Hay que aclarar que **éste no es el código final ni el diagrama de flujo final** pero es necesario documentarlo para explicar por qué se tuvo que volver a modificar. Se expondrá en primer lugar la función "*setup ()*":



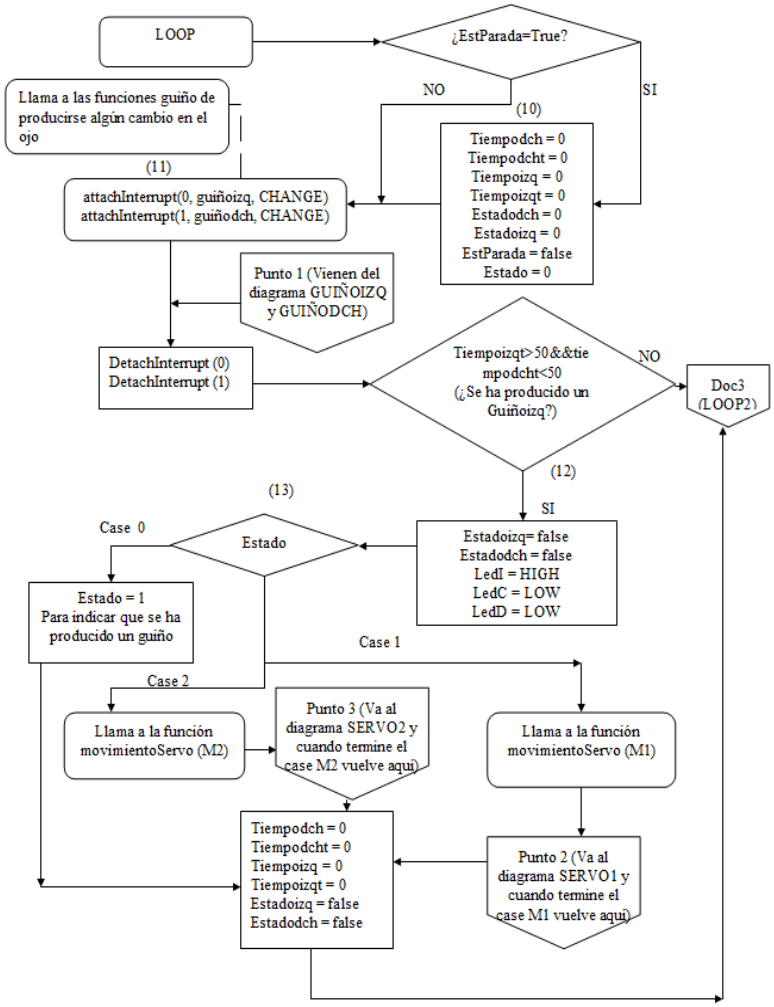
En la primera parte, antes de incluir la función "*setup ()*", se declaran una serie de valores necesarios para detectar los guiños, contabilizar el tiempo que permanece el botón presionado, etc.

Ya en la función "*setup ()*" se expresa en qué pines irán conectados los servos y se definen dos interrupciones para los guiños izquierdo y derecho en los pines 2 y 3. En el *Arduino Duemilanove*, para utilizar las interrupciones hay que conectar con los pines digitales 2 y 3 [20]: El número 0 indica el pin 2 y el número 1 indica el pin 3.

La ejecución *"Serial.begin (9600)"* es necesario escribirla en todos los sketches, siempre en la función "*setup ()*". Con dicha línea, se define la velocidad de transferencia de datos entre el ordenador y la plataforma electrónica.

Las dos siguientes partes corresponden al diagrama de flujo de la función "*loop ()*".

El diagrama de la función "*loop ()*" se ha dividido en dos partes. A la primera parte se la llamará "*LOOP1*" y a la segunda parte *"LOOP2*". Nada más empezar, en la función "*loop ()*", hay un "*if*", para ver si se ha pulsado al botón de parada o no. Al presionar el botón de parada, la variable "*EstParada*", que antes poseía el valor "*false*", se convierte en "*true*", por lo que accede al "*if*" (Punto 10 del diagrama), inicializando las variables del programa.

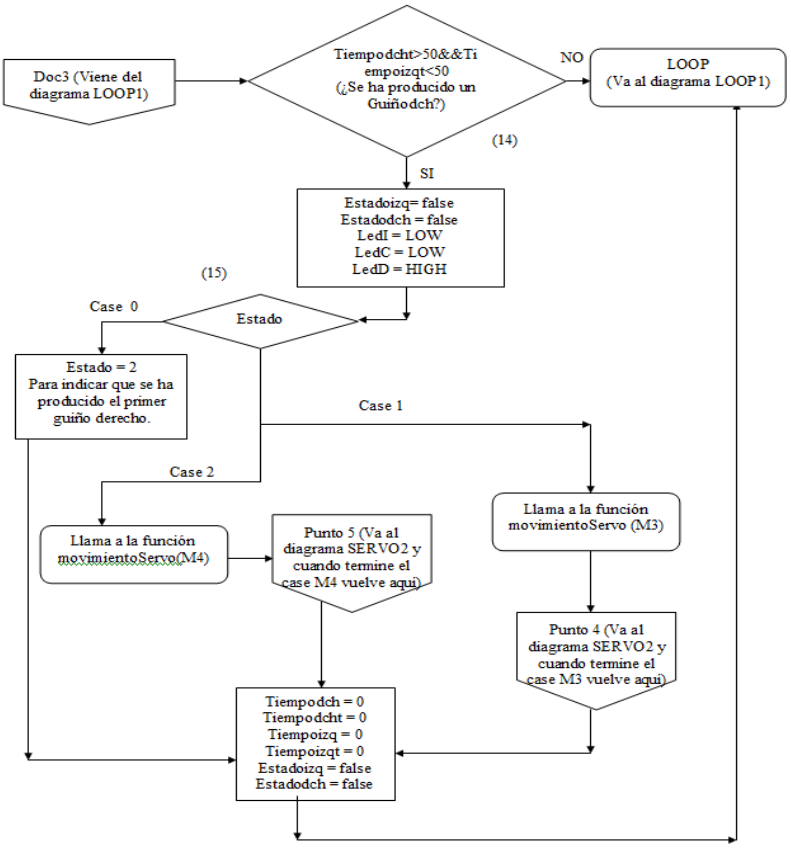


El siguiente punto a importantes a tratar son las interrupciones. En este caso, si se detecta alguna interrupción en algunos de esos pines, es decir, si se pulsa algún botón del reposapiés derecho, se llama a una de esas funciones, "*Guiñoizq*" o "*Guiñodch*" (Hay que entender que en este código que se desarrolló, se dejaron los nombres de estas funciones pero cuando escribimos *Guiñoizq*, nos referimos a pulsar el botón izquierdo del reposapiés derecho y cuando indicamos *Guiñodch*, nos referimos al botón derecho del reposapiés derecho).

De no producirse ningún cambio en esos botones, el programa no haría nada más, ya que solo entra en el "*if*" siguiente (el marcado en el punto 12 del diagrama) si se ha pulsado el botón izquierdo y en el "*if*" del diagrama de abajo (Punto 14 del diagrama) si se ha presionado el botón derecho.

En el caso de obtener un cambio en el botón izquierdo, se llama a la función "*Void Guiñoizq ()*" (Punto 16 del diagrama) donde se realizan una serie de operaciones para calcular el tiempo que permanece el botón apretado para comprobar que ciertamente se ha pulsado dicho botón. De haberse presionado correctamente, accedería al "*if*" (Punto 12 del diagrama) y entraría en el "*Case 0*" de ser la primera vez que se ha pulsado. En los otros dos *"Cases*" depende de si el botón que se ha apretado la primera vez es el izquierdo (llamada a M1) o el derecho (llamada a M2).

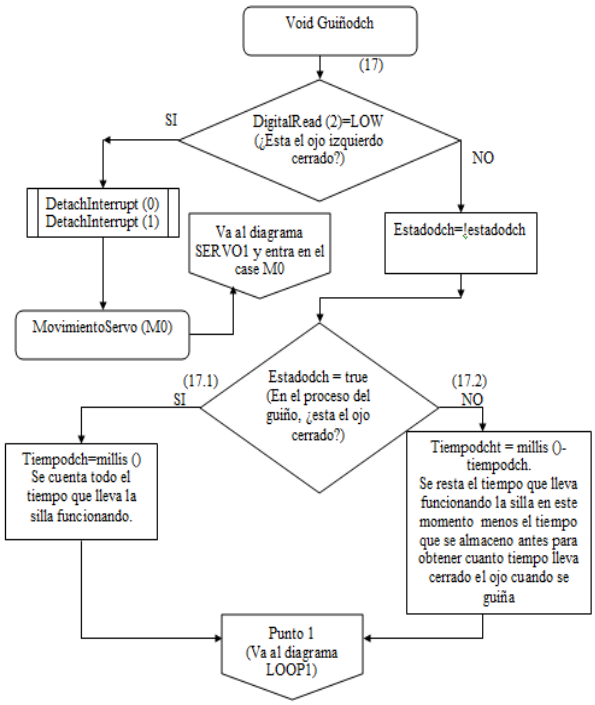
El siguiente diagrama de flujo (*"LOOP2"*) de la función "*loop ()*", resulta prácticamente similar a la parte del "*switch*" anteriormente explicado para el botón izquierdo, pero esta vez se aplica al botón derecho.



En el caso de haberse presionado el botón derecho, se accedería al "*if*" (Punto 14 del diagrama) y una vez dentro de este "*if*", si es la primera vez que se ha apretado el botón, accedería al "*Case 0*" y se guardaría una variable "*Estado*", reflejando que se ha pulsado el botón derecho. Si es la segunda vez que se presiona, y la primera vez se apretó el botón izquierdo, accedería al "*Case 1*", llamando a la función "*movimientoServo (M3)*". Si fue el botón derecho el que se pulsó la primera vez, entraría al "*Case 2*" para llamar a la función "*movimientoServo (M4)*".

Ahora se explicará la parte del diagrama de flujo que tiene que ver con las funciones "*Void Guiñoizq* y *Guiñodch*". Se empezará con el diagrama de flujo de la función *"Void Guiñodch ()"*.

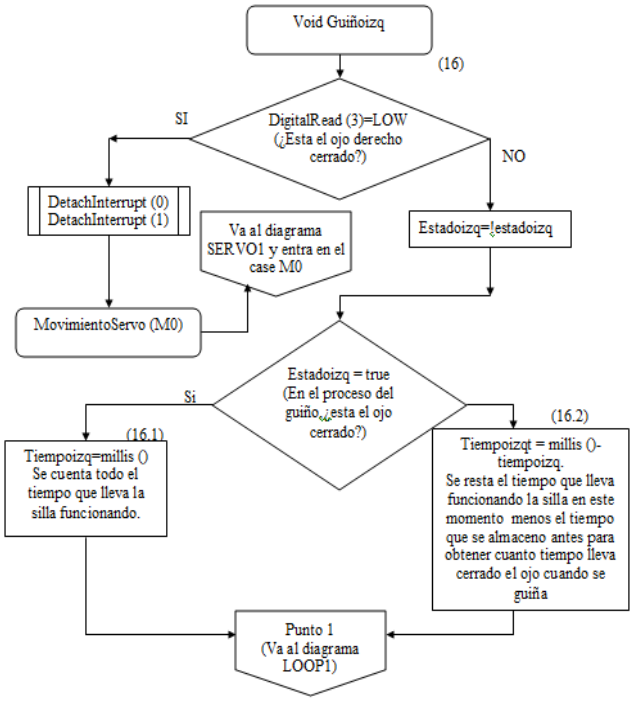
En el caso de producirse una interrupción en el pin 3 se accede a esta función, es decir, cuando se aprieta el botón derecho. Al introducirse en la función, hay un "*if*" para comprobar si se está pulsando el botón izquierdo. Si está presionado, se interpreta como que no se ha apretado el botón de la derecha, sino que se ha pulsado el botón de parada en el reposapiés izquierdo. Si es así, entra en el "*if*" y llama a la función "*movimientoServo*" accediendo al "*Case 0*" para detener la silla.



Si se ha pulsado al botón de la derecha del reposapiés derecho entonces no accede a ese "*if*", y

mediante el estado de la variable "*Estadodch*" podemos obtener el tiempo que se ha mantenido presionado el botón derecho. Para ello, entra en un "*if*" si detecta que se está pulsando el botón, y se iguala el tiempo que lleva encendida la plataforma *Arduino* a una variable llamada "*Tiempodch*".

Cuando soltemos el botón, se volverá a llamar a la función "*Guiñodch ()*", puesto que se la llama siempre que se produzca un cambio. Pero esta vez, no accederá al "*if*" y realizará la operación que hay en el diagrama (Punto 17.2 del diagrama) para obtener el tiempo total que se ha mantenido apretado el botón derecho.

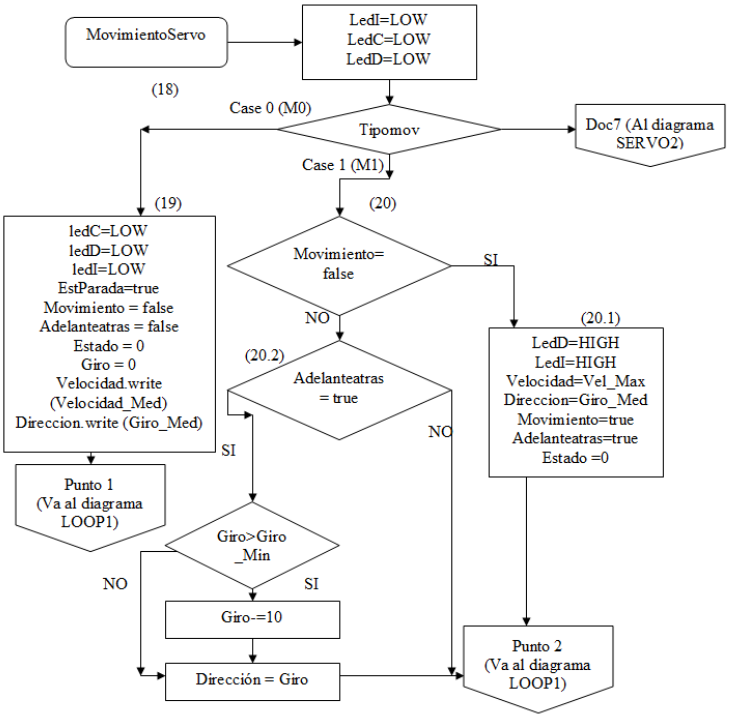


A continuación, se pasará a exponer el diagrama de flujo correspondiente al botón izquierdo que funcionaría del mismo modo que el derecho.

El diagrama de esta función es prácticamente idéntica a la función guiño derecho. En este caso, en vez de examinar el pin 2, que es el del botón izquierdo, comprobamos el pin 3, para determinar si se ha apretado el botón de parada o el botón de la izquierda. En el caso de que se pruebe que se ha pulsado solo el botón izquierdo, no entra en el "*if*" (Punto 16 del diagrama) y siguiendo el método anterior se mide el tiempo que se ha mantenido presionado el botón.

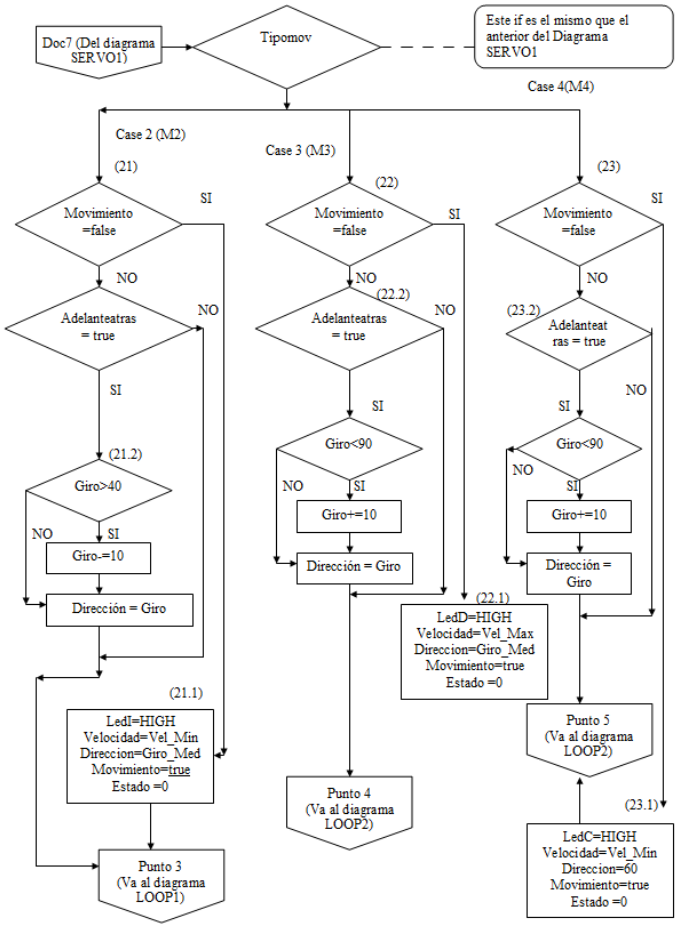
Una vez explicado el funcionamiento de las funciones a las que se llama al pulsar los botones, pasaremos a describir qué hace cada llamada a la función "*movimientoServo*" que vimos en el diagrama de flujo de la función "*loop ()*". Como también es bastante extenso, se volverá a estructurarlo en dos partes: Al primer diagrama se le llamará *"SERVO1"* y al segundo *"SERVO2"*.

En esta primera parte podemos ver un "*switch (tipomov)*" que entra en cada "*Case*" dependiendo de los botones que se hayan pulsado y su orden. Si accedemos al "*Case 0*", significa que se ha presionado el botón de parada y es en esta parte del programa (Punto 19 del diagrama), donde se guardan los valores necesarios en las variables que controlan los servomotores para dejar el joystick en posición de parada. Además, se inicializan otras variables necesarias para el funcionamiento del programa. Se declara "*Movimiento=false*" y *"Adelanteatras=false"* para indicar que la silla esta parada.



En el siguiente, "*Case*" (*Case 1*), se ejecuta el movimiento de avance estableciendo los valores necesarios a las variables servos (Velocidad y Dirección). Una vez se esté avanzando, si se pulsa uno de los botones del reposapiés derecho, se activa la posibilidad de virar la silla hacia la derecha o hacia la izquierda. En esta parte del diagrama (Punto 20.2 del diagrama) se viraría un poco a la izquierda. Esto corresponde con la línea "*Giro-=10*" y se iguala a "*Dirección*", que es el nombre de uno de los servos. Una vez realizadas las operaciones indicadas, el programa vuelve al Punto 2 indicado en el diagrama de flujo "*LOOP1*".

Esta parte del diagrama de flujo relacionado con la función "*movimientoServo*" es en realidad el "*switch*" anterior con otras tres posibilidades: Giro hacia la izquierda ("*Case 2*"), hacia la derecha ("*Case 3*") y marcha atrás ("*Case 4*").



Como se explicó antes, al terminar cada uno de estos "*Cases*", vuelve a los puntos que se indican en el diagrama de flujo.

Se accede al "*Case 2*" cuando se aprieta el botón derecho por primera vez y luego el izquierdo, girando la silla hacia la izquierda. Si no se está avanzando, se detecta con el "*if Movimiento=false*" ejecutando en los servomotores los valores indicados en el punto 21.1 del diagrama. Una vez realizado, vuelve al Punto 3, indicado en el diagrama "*LOOP1*". En el caso de estar avanzando, "*Movimiento=true*" y "*Adelanteatras = true*". Se accede al punto 21.2 del diagrama y se vira la silla un poco a la izquierda con las líneas "*Giro-=10*" y "*Dirección = Giro*".

Para los dos "*Cases*" restantes el funcionamiento es muy similar a los ya explicados. El "*Case 3*" tiene la

función de girar la silla a la derecha presionando en primer lugar el botón izquierdo y luego el derecho y el "*Case 4*" de desplazarla hacia atrás, apretando dos veces el botón derecho.

Una vez terminado el diagrama de flujo, se instalaron los drivers necesarios en el ordenador del laboratorio para poder transferir el programa desde el ordenador a la plataforma *Arduino* mediante un cable USB. Como la plataforma de la silla es la *Duemilanove* [21], se instalaron los drivers propios de esta plataforma en el ordenador para lograr la transferencia.

3.1.2 Programa Arduino

Una vez realizado el diagrama de flujo necesario para entender todo el código, se modifica el código inicial de acuerdo al diagrama.

```
#include <Servo.h>      //Librería necesaria para trabajar con los dos servomotores
de rotación

int DEBUG = 1;

int Velocidad_Min = 11;
int Velocidad_Med = 73;    // Velocidades (posición de los servos.)  Diagrama= (1)-
>Declaración variables velocidad y giro
int Velocidad_Max = 135;
int Velocidad_Atras = 45;
int Velocidad_Izq = 31;
int Velocidad_Drch = 125;

int Giro_Min = 48;
int Giro_Med = 74;
int Giro_Max = 92;

boolean estadoizq = false;
boolean estadodch = false;
boolean movimiento = false;      // Diagrama = (2)->Declaración variables Booleanas
boolean adelanteatras = false;
boolean EstParada = false;  //Variable que pasa a true cuando apretamos el botón de
parada

unsigned long tiempoizq = 0;
unsigned long tiempodch = 0;      //Diagrama = (3)->Variables declaradas para indicar
el tiempo en el que permanece el botón presionado.
unsigned long tiempoizqt = 0;
unsigned long tiempodcht = 0;

int estado=0;                  //estado 0= primera detección; 1= guiñoizq 1ª
detección; 2= guiñodrch 2ª detección  (4)
int avanza = 0;   //Variable puesta para que al llegar a 3 o mas se reinicie la silla
al pararla
int ledI=8;  //indicador luminoso izquierdo
int ledC=9;  //indicador luminoso central              //Diagrama = (4)-> Declaración
de variables usadas para los led
int ledD=10;  //indicador luminoso derecho
int funciona=7; //indicador luminoso de funcionamiento

int giro=0;      //Diagrama -> (4)        Variable que se utilizara para indicar el
valor del Servo dirección

Servo velocidad; //Servo de control de la velocidad
Servo direccion; //Servo de control de la dirección de movimiento      //Diagrama=
(5)->Servos
```

Como se procedió antes, se dividirá el código en partes y se irá explicando una a una para poder entenderlo mejor. Además, en los comentarios se específica a qué parte del diagrama corresponde cada línea.

En la primera parte del código, inicializamos las variables que vamos a utilizar. Cada una de ellas viene comentada para entender su funcionamiento.

Esta primera parte del código es solo declaración de variables. La siguiente parte del código a explicar será la función "*setup ()*".

```
void setup() {                              //Diagrama (S)->Inicio

        pinMode(ledD, OUTPUT);      //Salida de los LEDs
        pinMode(ledI, OUTPUT);
        pinMode(ledC, OUTPUT);
        pinMode(funciona,OUTPUT);    //Diagrama = (6)-> Establecer los 4 leds como
salidas
        if (DEBUG){
                Serial.begin(9600);    //Diagrama = (7) -> DEBUG
        }
        Serial.println("Iniciando configuración");
        velocidad.attach(11);    //Configuración de los servos  //Diagrama = (8)
        direccion.attach(12);
        velocidad.write(Velocidad_Med);
        direccion.write(Giro_Med);


    digitalWrite(funciona,HIGH);  //Enciende el indicador de funcionamiento

    Serial.println("Configuración Finalizada");  //Diagrama = (9) -> Interrupciones
para cualquier cambio en pines 2 y 3
    attachInterrupt(0, guinoizq, CHANGE);  //Habilitar interrupciones por los pines 2
y 3; Para más información-> http://arduino.cc/en/Reference/attachInterrupt
    attachInterrupt(1, guinodch, CHANGE);
}
```

Esta función se encarga de declarar los pines *ledD* (pin 10), *ledI* (pin 8), *ledC* (pin 9) y *funciona* (pin 7) como salidas. El resto del código que se encuentra en la función "*setup ()*" sirve para la correcta configuración de los servomotores.

Ahora, se continuará con una pequeña parte de la función "*loop ()*". La plataforma *Arduino* recorrerá, con un bucle infinito, esta función hasta que se pulse el botón *Reset*, lo que hará que se reinicie el programa y vuelva a ejecutarse la función *"setup ()"*, o hasta que se quite la alimentación de la plataforma.

```
void loop() {    //Diagrama = (loop)

          if(EstParada ==true){                    //Entra aquí de haberse pulsado
          el botón de parada para resetear la silla (RT)
               tiempodch=0;
               tiempodcht=0;
               tiempoizq=0;
               tiempoizqt=0;          //Diagrama = (10) -> Entra aquí cuando se
               presiona el botón de parada y reinicia variables. ESTPARADA = TRUE en
               el momento del parpadeo
               estadodch=false;
               estadoizq=false;
               EstParada = false;
               estado = 0;

               Serial.println("RESETEO DE GUIÑOS!!");
        }


        attachInterrupt(0, guinoizq, CHANGE);        //Interrupciones puestas para
        determinar si ha existido algún cambio tanto en el botón izquierdo como en el
        derecho (RT)

        attachInterrupt(1, guinodch, CHANGE);  //Diagrama (11)

        detachInterrupt(0);           //Detiene la interrupción(RT)

        detachInterrupt(1);
```

En esta parte de la función *"loop ()"*, se puede dividir en dos apartados: El correspondiente a todo lo contenido en el *"if"* y las cuatro sentencias relacionadas con las interrupciones.

El primero de ellos, es un "*if*", al cual, se accede cuando se ha pulsado el botón de parada, aunque antes de acceder aquí llama a la función "*movimientoServo* (0)" (como se expondrá más adelante) para parar la silla, declarar la variable "*EstParada*" a "*true*" y de esa forma entrar en el "*if*".

En cuanto a las interrupciones que suceden en los pines 2 y 3, estos pines son los botones que dirigen la silla, los botones derecha e izquierda que se encuentran en el reposapiés derecho. En cuanto detecte un cambio en el botón derecho o izquierdo, se llamará a las funciones *guinodch* o *guinoizq*.

La siguiente parte de la función "*loop* ()" consiste en dos "*ifs*" en los que accede dependiendo de si se toma como válido la pulsación de uno de estos dos botones. La validez de una pulsación depende del tiempo que se mantenga apretado el botón.

```
if(tiempoizqt > 50 && tiempodcht < 50)       //Una vez que en guinoizq hemos
establecido si se ha presionado el botón izquierdo o no, entramos en este if
de haber existido una pulsación(RT)
{
        //Guiñoizq
        estadoizq = false;
        estadodch = false;         //Diagrama = (12) ->Si se pulsa el botón
        izquierdo; Es decir, la variable tiempoizqt > 50
        Serial.print("Se detecto guiño izq\tTiempoizqt:");
        Serial.print(tiempoizqt);
        Serial.print("\tTiempodcht");
        Serial.println(tiempodcht);
        digitalWrite(ledI, LOW);
        digitalWrite(ledC, LOW);      //Se establece el valor de los LED para
        saber en que modo estamos (RT)
        digitalWrite(ledD, LOW);
        digitalWrite(ledI, HIGH);
        switch(estado){           //Diagrama = (13)-> Llamadas a movimiento
        servo según la variable estado
        case 0:
                estado = 1;       //Entra en este case de ser la primera vez
                que pulsamos el botón izquierdo (RT)
                break;            //
        case 1:
                movimientoservo(1);  //Llama a la función movimiento servo
                para establecer el joystick en movimiento avance; Se accede al
                apretar dos veces al botón izquierdo(RT)
                break;                //[M1]
        case 2:
                movimientoservo(2);  //Llama a la función movimiento servo
                para establecer el joystick en giro izquierda; Se accede al
                pulsar en primer lugar el botón derecho y después el
                izquierdo(RT)
                break;                //[M2]
        }
        tiempodch=0;
        tiempodcht=0;
        tiempoizq=0;
        tiempoizqt=0;
        estadodch=false;
        estadoizq=false;
        delay(200);
}
```

```
        if(tiempoizqt < 50 && tiempodcht > 50)              //Una vez que en guinoder
        hemos establecido si se ha pulsado el botón derecho, entramos en este if (RT)
        {
                //Guiño Derecho o Botón derecho
                estadoizq = false;      //Diagrama de flujo->(14) ->Si se presiona
                botón derecho. Es decir, la variable tiempodcht > 50
                estadodch = false;
                Serial.print("Se detecto guiño derecho\tTiempoizqt:");
                Serial.print(tiempoizqt);
                Serial.print("\tTiempodcht");
                Serial.println(tiempodcht);
                digitalWrite(ledI, LOW);      //Se establece el valor de los LED para
                saber en que modo estamos (RT)
                digitalWrite(ledC, LOW);
                digitalWrite(ledD, LOW);
                digitalWrite(ledD, HIGH);
                switch(estado){         //Diagrama de flujo->(15)
                case 0:
                        estado = 2;         //Entra en este case de ser la primera vez
                        que apretamos el botón derecho (RT)
                        break;
                case 1:
                        movimientoservo(3); //Llama a la función movimiento servo para
                        establecer el joystick en giro derecha; Se accede al pulsar el
                        botón izquierdo y después el derecho(RT)
                        break;              //[M3]
                case 2:
                        movimientoservo(4); //Llama a la función movimiento servo para
                        establecer el joystick en movimiento hacia atrás. Se accede
                        pulsando dos veces el botón derecho  (RT)
                        break;              //[M4]
                }
                tiempodch=0;
                tiempodcht=0;
                tiempoizq=0;
                tiempoizqt=0;
                estadodch=false;
                estadoizq=false;
                delay(150);
        }
}
```

Para entrar en estos dos "*ifs*", el del botón izquierdo o el del botón derecho, antes tiene que entrar en dos funciones: guinodch y guinoizq (que se explicarán más adelante), pero en esencia, en esas funciones se calcula el tiempo que transcurre mientras se está pulsando el botón, y de pulsarse, se entra en el "*if*" correspondiente.

En cualquiera de los dos "*ifs*", al entrar, se encienden unos determinados leds para poder saber en qué estado nos encontramos según la siguiente tabla:

LEDs	Significado
	Botón Izquierdo
	Botón derecho

Figuras 3-1 y 3-2: Estado de los leds según la orden recibida

LEDs	Significado
	Parado
	Avance
	Atrás
	Giro izquierdo
	Giro derecha

El estado de los leds nos indicará en qué estado estamos del programa, como podemos ver en las figuras 3-1 y 3-2

Ya se han visto la función "*setup ()*" y la función "*loop ()*", ahora explicaran las funciones restantes: "*guinoizq*", "*guinodch*" y "*movimientoServo*".

```
void guinoizq()
{
    if(digitalRead(3)==LOW){              //Diagrama->(16)
              detachInterrupt(0);    //Accede aquí solo para parar la silla en el
              caso de haberse pulsado el botón de parada. Entra porque detecta tanto
              el pin 3 como el 2 en LOW, y eso solo se produce si se presiona dicho
              botón. (RT)
              detachInterrupt(1);
              movimientoservo(0);     //[M0]
              return;
    }
    else{
              estadoizq = !estadoizq;
              if(estadoizq == true)       //Diagrama (16.1)
              {
                     tiempoizq = millis();         //Establece el tiempo que ha
                     transcurrido desde que se encendió la silla en tiempoizq y sale
                     de la función.
                     //El estado de izq es LOW
              }
              else     //Diagrama (16.2)
              {
                     tiempoizqt = millis() - tiempoizq; //Cuando lo soltemos,
                     vuelve a llamar a millis para saber el tiempo que lleva
                     encendida la silla y lo resta del valor anterior para saber el
                     tiempo que ha permanecido el botón apretado. (RT)
                     //El estado izq es HIGH
                 Serial.print("tiempoizqt:");
                 Serial.println(tiempoizqt);
                 delay(50);
              }
    }
}
```

Esta función es llamada justo cuando se produce alguna interrupción en el botón izquierdo del reposapiés derecho o cuando se ha pulsado el botón de parada. Cuando se llama a la función, se comprueba si por el pin 3 hay cero voltios, si es así, es que se ha presionado el

botón de parada y llama a la función "*movimientoServo*" para parar la silla. De lo contrario, se ha apretado el botón izquierdo.

La siguiente parte de esta función es la que se encarga de contabilizar el tiempo que se mantiene pulsado el botón. La primera vez que entra, guarda el tiempo que lleva encendida la silla con la función "*millis ()*" en "*tiempoizq*". En este momento se ha presionado el botón.

Cuando soltemos el botón entrará en el "*else*"(Punto 16.2 del diagrama), guardando en la variable "*Tiempoizqt*" el tiempo que se ha mantenido pulsado el botón, ya que resta el tiempo que lleva encendida la silla menos el tiempo que llevaba encendida justo al pulsar el botón. Esto nos dará como resultado el tiempo que llevamos presionando el botón.

```
void guinodch()       //Diagrama->(17)
{
        if(digitalRead(2)==LOW){
                detachInterrupt(0);    //Accede aquí solo para parar la silla en el
                caso de haberse pulsado el botón de parada. Entra porque detecta tanto
                el pin 3 en LOW como el pin 2, y eso solo se produce si se presiona
                dicho botón. (RT)
                detachInterrupt(1);
                movimientoservo(0);     //[M0]
                return;
        }
        else{
                estadodch = !estadodch;
                if(estadodch == true)  //Diagrama (17.1)
                {
                        tiempodch = millis();         //Al presionar el botón,
                        establece el tiempo que ha transcurrido desde que se encendió
                        la silla en tiempodch.
                        //El estado de dch es LOW
                }
                else     //Diagrama (17.2)
                {
                        tiempodcht = millis() - tiempodch; //Cuando lo soltemos,
                        vuelve a llamar a millis para saber el tiempo que lleva
                        encendida la silla y lo resta del valor anterior para saber el
                        tiempo que ha permanecido el botón presionado. (RT)
                        //El estado dch es HIGH
                          Serial.print("tiempodcht:");
                          Serial.println(tiempodcht);
                          delay(50);
                }
        }
}
```

La función "*void guinodch*" es similar a la anterior, lo único que cambia son los nombres de las variables. La función anterior se encargaba de comprobar si se había pulsado el botón izquierdo o no. Esta se encarga del botón derecho.

Ahora pasaremos a comentar la última función de todo el código, la función "*movimientoServo*".

```
void movimientoservo(int tipomov){   //Diagrama->(18)
      digitalWrite(ledI, LOW);
       digitalWrite(ledC, LOW);
       digitalWrite(ledD, LOW);
       switch(tipomov){
              case 0:    //Parada  Diagrama->(19)      //[M0]
                     Serial.println("Parada");
                     velocidad.write(Velocidad_Med);
                     direccion.write(Giro_Med);
                     digitalWrite(ledC, LOW);  //Apaga el led indicador
                     digitalWrite(ledD, LOW);  //Apaga el led indicador
                     digitalWrite(ledI, LOW);  //Apaga el led indicador
                     EstParada = true;         //EstParada=True para entrar en el if
                     del principio nada mas empezar la función loop y parar la silla
                     movimiento = false;       //Para indicar que no hay movimiento
                     adelanteatras = false;

                     Velocidad_Max = 135;
                     Velocidad_Min = 11;
                     estado = 0;
                     giro = 0;
                     break;

              case 1:    //Avanza    Diagrama->(20)        //[M1]
                     if(movimiento == false){            //Si no hay movimiento
                     entra aquí y pone la silla en funcionamiento avanzando hacia
                     delante->(RT)
                            Serial.println("Avanza");   //Diagrama(20.1)
                            velocidad.write(Velocidad_Med);
                            delay(50);
                            direccion.write(Giro_Med);
                            delay(50);
                            velocidad.write(Velocidad_Max);
                            digitalWrite(ledD,HIGH);  //Enciende el led derecho
                            para indicar el movimiento que se esta ejecutando->(RT)
                            digitalWrite(ledI,HIGH);  //Enciende el led izquierdo
                            para indicar el movimiento que se esta ejecutando->(RT)
                            giro = Giro_Med;
                            movimiento = true;
                            adelanteatras = true;
                            estado = 0;

                     }else{

                            if(adelanteatras == true){  //Si la silla esta
                            avanzando entonces entra en este if para girar un poco
                            a la izquierda->(RT)
                                   if(giro > Giro_Min){  //Diagrama(20.2)
                                          Serial.println("resto 10");
                                          giro -= 10;
                                 direccion.write(giro);
                                 delay(10);
                                //Gira un poco a la izquierda
                                   }
                            }
                     }
                     break;
```

La función "*movimientoServo*", como su propio nombre indica, es la que se encarga de mover los servomotores para realizar todas las maniobras de la silla. En concreto, esta parte del código, se centra en el movimiento de parada y avance. Cuando se pulse el botón de parada y se entre en esta función, accederá directamente al "*Case 0*" apagando los *leds* para indicarnos que se ha parado la silla y moviendo el joystick para detener la silla de ruedas.

Si apretamos dos veces el botón izquierdo del reposapiés derecho entraremos en el "*Case 1*" del "*switch*" y se encenderán los leds de la forma que se describió en la tabla 3-2 para indicar el estado de avance.

```
        case 2:   //Giro izquierda  Diagrama->(21)     [M2]
                        if(movimiento == false){            //Si no hay
        movimiento entra aquí y pone la silla en
        funcionamiento para girar a la izquierda->(RT)

                                Serial.println("Giro izquierda");  //Diagrama (21.1)
                                digitalWrite(ledI,HIGH);
                                velocidad.write(Velocidad_Med);
                                delay(50);
                                direccion.write(150);
                                delay(50);
                                velocidad.write(Velocidad_Izq);
                                movimiento = true;
                                estado = 0;

                                Serial.print(" Lectura servo velocidad ");  //Segunda
                                verificación para el servo velocidad
                                Serial.println( velocidad.read());

                        }else{
                                if(adelanteatras == true){     //Si la silla esta
                                avanzando entonces entra en este if para girar un poco
                                a la izquierda->(RT)
                                        if(giro > 40){       //Diagrama (21.2)
                                                Serial.println("resto 10");
                                                giro -= 10;  //Antes estaba a 5
                                                direccion.write(giro);
                                              delay(10);

                                        }
                                        Serial.print("Giro un poco a la izquierda\tGiro:
");
                                        Serial.println(giro);


                                }
                        }
                        break;
```

```
                case 3:    //Giro Derecha  Diagrama->(22)  //[M3]
                        if(movimiento == false){          //Si no hay movimiento entra
                        aquí y pone la silla en funcionamiento para girar a la derecha-
                        >(RT)

                                Serial.println("Giro derecha");  //Diagrama (22.1)
                                digitalWrite(ledD,HIGH);
                                //Cambia la velocidad del giro derecha; A ver si lo
                                coge.
                                velocidad.write(Velocidad_Med);
                                Serial.print("Velocidad_Med Giro Der: ");
                                //Velocidad_Med giro der
                                Serial.println(Velocidad_Med);
                                delay(50);
                                direccion.write(150);
                                   Velocidad_Drch = 125;
                                delay(50);
                                velocidad.write(Velocidad_Drch);  //Antes estaba a
                                Vel_Max=135

                                Serial.print("Velocidad_Max Giro Der: ");  //Comprobar
                                el valor de velocidad_Max que tenia antes
                                Serial.println(Velocidad_Drch);
                                movimiento = true;
                                estado = 0;
                                Serial.print(" Lectura servo velocidad ");  //Lectura
                                servo velocidad
                                Serial.println( velocidad.read());
                        }else{
                                if(adelanteatras == true){  //Si la silla esta
                                avanzando entonces entra en este if para girar un poco
                                a la derecha->(RT)
                                        if(giro < 90){      //Diagrama (22.2)
                                                Serial.println("sumo 10");
                                                giro += 10;
                                              direccion.write(giro);
                                              delay(10);

                                        }
                                        Serial.print("Giro un poco a la derecha\tGiro:
");
                                        Serial.println(giro);

                                }
                        }
                break;
```

```
                case 4:   //Atrás    Diagrama->(23)    [M4]

                        if(movimiento == false){       //Si no hay
        movimiento entra aquí y pone la silla en
        funcionamiento para dar marcha atrás->(RT)
                                         //Diagrama (23.1)
                                Serial.println("Atras");
                                digitalWrite(ledC,HIGH);
                                velocidad.write(Velocidad_Med);
                                delay(50);
                                direccion.write(60);
                                delay(50);
                                velocidad.write(Velocidad_Atras);
                                movimiento = true;
                                estado = 0;
                        }else{                          //Diagrama (23.2)
                                if(adelanteatras == true){      //Si la
                                silla esta avanzando entonces entra en este
        if para girar un poco a la derecha->(RT)
                                        if(giro < 90){
                                                Serial.println("sumo 10");
                                                giro += 10;
                                              direccion.write(giro);
                                              delay(10);

                                        }
                                        Serial.print("Giro un poco a la derecha\tGiro:
");
                                        Serial.println(giro);

                                }
                        }
                break;
        }
}
```

Estos dos cases del *switch* sirven para mover el joystick de forma que se rote la silla hacia la izquierda (en el "*Case 2*") y hacia la derecha (en el "*Case 3*").

Como en el movimiento de avance, tiene también otra parte en la que solo entra cuando esta en modo avance y se ha pulsado a uno de los dos botones del reposapiés derecho. Una vez hecho eso, se puede virar un poco la silla mientras se avanza: si se pulsa al botón derecho viras a la derecha y si se presiona el izquierdo viras a la izquierda.

El ultimo "*Case*" es el del movimiento hacia atrás.

En el último "*Case*" se encienden los *leds* para indicar el estado en el que se encuentra la silla y mover la silla de ruedas hacia atrás. Éste movimiento se logra pulsando dos veces el botón derecho.

3.1.3 Resultados

Una vez concluido, se subió este programa a la plataforma *Arduino* de la silla para comprobar el funcionamiento de ésta. La primera prueba que se realizó con la silla no fue muy alentadora puesto que los servomotores se bloqueaban tras ordenar a la silla cualquier movimiento. Una vez bloqueados, la silla dejaba de responder, puesto que la primera medida a tomar fue la de aumentar los retrasos unos milisegundos más, para dar tiempo suficiente a los servomotores a realizar el movimiento necesario y conseguir desplazar el joystick sin bloqueo. Esto se consigue escribiendo entre los servomotores la línea "*delay (ms)*" y entre los paréntesis, se establece el valor en milisegundos que se desea para el retraso.

Esta medida fue eficaz y la silla dejo de bloquearse respondiendo con eficacia y rapidez a cada una de las órdenes que se la mandaban. Pero tras un tiempo continuado manejando la silla, ésta dejaba de reaccionar y se volvía a bloquear. Esto suponía un grave problema que había que solucionar puesto que este bloqueo podía producirse mientras la silla estaba en modo de avance de forma que si deseabas parar la silla accionando el botón de parada no era posible, creando una situación bastante peligrosa para el usuario.

Se realizaron más pruebas con este programa y se corroboró que tras un tiempo aproximado de media hora, la silla dejaba de responder, bloqueándose y ejecutando sin pausa el movimiento anterior que se la había mandado realizar.

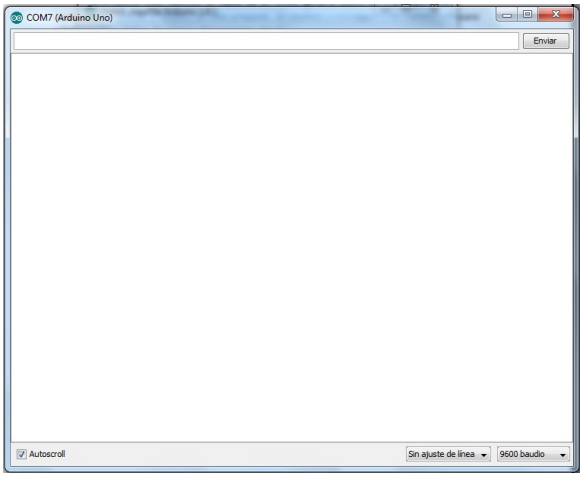


Figura 3-3: Monitor Serial del entorno de desarrollo *Arduino*

Para solventar este problema, se ejecutaron las órdenes en el laboratorio con el motor de la silla apagado y se visualizó lo que sucedía en el monitor *Serial* del entorno de desarrollo *Arduino* (Figura 3-3) para comprobar en qué punto del programa quedaba bloqueado.

Si se quiere acceder a este monitor, hay que conectar la plataforma *Arduino* de la silla al ordenador mediante un cable USB. Una vez realizado, y abierto el entorno de desarrollo, en la parte superior a la derecha se visualiza un icono con forma de lupa que, al dejar el ratón sobre él, se puede leer *Monitor Serie*. Al hacer clic sobre él, se abre el *Monitor* y presionando los botones que tiene la silla en los reposapiés se puede ir viendo en que parte del programa se encuentra gracias a los *Serial.print (“”)* colocados a lo largo del programa. Dentro de los paréntesis se escribe lo que se quiera visualizar en el monitor cuando el programa pasa por esa línea.

Mediante este método, se verificó el momento exacto donde el programa se bloqueaba. Cuando realizaba el algoritmo para calcular el tiempo que se mantiene pulsado el botón, el programa se quedaba justo en ese punto y a partir de ahí, era inútil apretar cualquier botón porque el sistema era incapaz de responder. Por ello, se procedió a simplificar el programa que se explicó con anterioridad, eliminando este algoritmo para evitar el bloqueo del sistema.

3.2 Programa definitivo

Al igual que en el primer programa, se irá comentando cada parte pero de una forma más breve puesto que gran parte del código es similar al anterior.

```
#include <Servo.h>


int DEBUG = 1;

int Velocidad_Min = 11;
int Velocidad_Med = 73;    // Velocidades (posición de los servos.)  Diagrama= (1)
->Declaración variables velocidad y giro
int Velocidad_Max = 135;
int Velocidad_Atras = 45;
int Velocidad_Izq = 31;
int Velocidad_Drch = 125;

int Giro_Min = 48;
int Giro_Med = 74;
int Giro_Max = 92;

boolean estadoizq = false;
boolean estadodch = false;
boolean movimiento = false;      // Diagrama = (2)->Declaración variables Booleanas
boolean adelanteatras = false;
boolean EstParada = false;  //Variable que pasa a true cuando realizamos el parpadeo
para parar la silla

unsigned long tiempoizq = 0;
unsigned long tiempodch = 0;      //Diagrama = (3)->Variables declaradas para indicar
el tiempo en el que permanece el ojo cerrado para cada guiño
unsigned long tiempoizqt = 0;
unsigned long tiempodcht = 0;

int estado=0;                  //estado 0= primera detección; 1= guiñoizq 1ª
detección; 2= guiñodrch 2ª detección  (4)

int ledI=8;  //indicador luminoso izquierdo
int ledC=9;  //indicador luminoso central              //Diagrama = (4)-> Declaracion
de variables usadas para los led
int ledD=10;  //indicador luminoso derecho
int funciona=7; //indicador luminoso de funcionamiento

int giro=0;      //Diagrama -> (4)        Variable que se utilizara para indicar el
valor del Servo direccion

Servo velocidad; //Servo de control de la velocidad
Servo direccion; //Servo de control de la direccion de movimiento      //Diagrama=
(5)->Servos
```

Esta parte del código corresponde a la inicialización de las variables. En el nuevo programa se han eliminado variables que eran innecesarias para su desarrollo.

```
void setup() {                    //Diagrama (S)->Inicio Void

        pinMode(ledD, OUTPUT);      //Salida de los LEDs
        pinMode(ledI, OUTPUT);
        pinMode(ledC, OUTPUT);
        pinMode(funciona,OUTPUT);    //Diagrama = (6)-> Establecer los 4 leds como
salidas
        if (DEBUG){
                Serial.begin(9600);    //Diagrama = (7) -> DEBUG

        }
        Serial.println("Iniciando configuración");

        velocidad.attach(11);    //Configuración de los servos  //Diagrama = (8)
        direccion.attach(12);
        velocidad.write(Velocidad_Med);
        direccion.write(Giro_Med);


    digitalWrite(funciona,HIGH);  //Enciende el indicador de funcionamiento

    Serial.println("Configuración Finalizada");  //Diagrama(9) ->
}
```

La función "*void setup ()*" del nuevo programa establece los pines donde irán los *leds* de señalización como salidas y configura los servomotores. Además, puede comprobarse cómo se han eliminado las líneas correspondientes a las interrupciones de los pines 2 y 3 puesto que el problema encontrado estaba relacionado con las funciones a las que llamaba. Concretamente, con el algoritmo realizado para calcular el tiempo que se mantenía pulsado cada botón. Por ello, se eliminó y se sustituyo por tres *"ifs"*: el primero para detectar si se ha pulsado el botón de parada, el segundo para comprobar si se ha producido algún cambio en el botón izquierdo, y el tercero, y último, para comprobar el botón derecho.

En lugar de utilizar las interrupciones para llamar a las funciones que realizaban el algoritmo, se detecta mediante un *"if"* que botones se ha pulsado. En este caso concreto, si detecta por los pines 2 y 3 una tensión de cero voltios (LOW), se llama a la función encargada de parar la silla.

```
void loop() {    //Diagrama = (loop)

       if(EstParada ==true){                //Accede aquí de presionar el botón de
parada para resetear la silla (RT)

              EstParada = false;
              estado = 0;
              Serial.println("RESETEO DE GUIÑOS!!"); //Diagrama = (10) -> Entra aquí
cuando se produce un parpadeo y reinicia variables. ESTPARADA = TRUE al pulsar el
botón de parada.

       }

       if(digitalRead(2)==LOW && digitalRead(3)==LOW)
       {
          movimientoservo(0); //Diagrama (11) -> En cuanto pulses al botón de parada
          ( o se realice un parpadeo) se llama a la función movimiento servo(0) que
          parará la silla (RT)

       }
```

```
if(digitalRead(2)==LOW && digitalRead(3) == HIGH)      // Diagrama (12) -> Entras
en este if en el momento que  pulses el botón izquierdo(RT)
   {
   delay(20);
      if(digitalRead(2) == HIGH && digitalRead(3)==HIGH)
      {
                  // Entras en este if en el momento que sueltes el botón
                  izquierdo (RT)
                  //Guiñoizq o botón izquierdo
            digitalWrite(ledI, LOW);
            digitalWrite(ledC, LOW);      //Se establece el valor de los LED para
            saber en que modo estamos (RT)
            digitalWrite(ledD, LOW);
            digitalWrite(ledI, HIGH);
            switch(estado){          //Diagrama = (13)-> Llamadas a movimiento
            servo según la variable estado
            case 0:
                  estado = 1;      //Entra en este case de ser la primera vez que
                  pulsamos el botón izquierdo (RT)
                  break;           //
            case 1:
                  movimientoservo(1);  //Llama a la función movimiento servo para
                  establecer el joystick en movimiento avance; Se accede pulsando
                  dos veces al botón izquierdo(RT)
                  break;               //[M1]
            case 2:
                  movimientoservo(2);  //Llama a la función movimiento servo para
                  establecer el joystick en movimiento izquierda; Se accede
                  pulsando al botón derecho y después al izquierdo(RT)
                  break;               //[M2]
            }
            delay(200);
      }
   }
```

Esta primera parte de la función "*void loop ()*" consiste en "*ifs*" a los que el programa accede en función del botón que se pulse. El primero de ellos accede solo después de presionar el botón de parada y es similar al que había en el código anterior, solo que inicializa muchas menos variables. El siguiente "*if*" es el relacionado con el botón de parada: Solo entra en él nada mas pulsar dicho botón, llamando al "*case 0*" que se encuentra en la función "*movimientoServo*". Para contabilizar una pulsación con el botón izquierdo, se emplean dos "*ifs*": El primero, para indicar que se esta pulsando el botón, el cual se puede dejar apretado todo el tiempo que se quiera, y el segundo, al que se accede solo cuando se ha soltado el botón, contabilizándose una pulsación y llamando a la función "*movimientoServo*" dependiendo de los botones pulsados.

```
      if(digitalRead(3) == LOW && digitalRead(2) == HIGH)   //Una vez que en guinoder
      hemos establecido si hubo guiño o no, entramos en este if de haber existido un
      guiño der (RT)
        {
            delay(20);
          if(digitalRead(3) == HIGH && digitalRead(2) == HIGH )
          {

               //Diagrama de flujo->(14) ->Si se produce un guiño derecho; Es decir,
               la variable tiempodcht > 50

               digitalWrite(ledI, LOW);      //Se establece el valor de los LED para
               saber en que modo estamos (RT)
               digitalWrite(ledC, LOW);
               digitalWrite(ledD, LOW);
               digitalWrite(ledD, HIGH);
               switch(estado){        //Diagrama de flujo->(15)
               case 0:
                       estado = 2;        //Entra en este case de ser la primera vez
                       que presionamos el botón derecho. (RT)
                       break;
               case 1:
                       movimientoservo(3); //Llama a la función movimiento servo para
                       establecer el joystick en movimiento derecha; Se accede
                       pulsando el botón izquierdo y luego el derecho. (RT)
                       break;              //[M3]
               case 2:
                       movimientoservo(4); //Llama a la función movimiento servo para
                       establecer el joystick en movimiento hacia atrás. Se accede
                       pulsando dos veces el botón derecho. (RT)
                       break;              //[M4]
               }
               delay(200);
          }
        }
}       //Cierre de la función LOOP
```

La siguiente parte de la función "*loop ()*" sirve para contabilizar una pulsación del botón derecho y llamar a la función "*movimientoServo*". Es igual al anterior bloque explicado salvo porque aquí se lee si el pin 3 recibe una tensión de 0 voltios.

Como se puede ver, con este código no llamamos a ninguna función para contabilizar tiempo, y con solo dos "*ifs*" se consigue contabilizar si se ha pulsado correctamente el botón.

El resto del código corresponde a la función *"MovimientoServo ()"*. Esta función es igual que en el primer programa explicado y su código viene detallado en el apéndice final (Código final de la silla).

Una vez terminado el código, se subió el programa al *Arduino* de la silla para comprobar su funcionamiento.

3.2.1 Resultados con el programa definitivo

Los resultados obtenidos con el nuevo programa fueron muy satisfactorios. La silla respondía con eficacia a cada una de las ordenes recibidas y era capaz de estar en funcionamiento todo el tiempo que la batería le suministrase energía sin temor a que el sistema pudiera bloquearse.

A pesar del buen funcionamiento del sistema, existen situaciones en las que la silla aumenta la velocidad sin explicación. Este suceso ocurre al girar a la izquierda y luego establecer a la silla en modo de avance. En este momento la velocidad de la silla aumenta. Cuando se detiene y se gira a la izquierda, al detenerla y volver a avanzar, la velocidad vuelve a

la normalidad. Este hecho puede deberse a un factor mecánico en el movimiento del joystick puesto que, como se ha podido comprobar en el código, los valores que se establecen a los servomotores para avanzar son siempre los mismos y no cambian nunca.

Una posible solución, podría ser la de disminuir el valor del servomotor de empuje justo después de efectuar un giro hacia la izquierda para intentar ajustar la velocidad a la que posee la silla por defecto.

3.3 Conclusiones

Este capítulo ha servido como guía y aclaración acerca de todas las dudas que pudieran tenerse acerca de los dos códigos empleados. Se ha realizado un diagrama de flujo detallando cada punto y relacionándolo con el código para servir de aclaración lo mejor posible. También se ha visto cómo fue el desarrollo desde el primer programa implementado en este proyecto y cómo se fueron solucionando los problemas que se presentaban hasta su puesta en funcionamiento, viendo cómo el sistema respondía perfectamente y sin ningún bloqueo.

Capítulo 4: Pruebas realizadas

Una vez que finalizado el desarrollo hardware y software del sistema, se procedió a realizar pruebas con voluntarios para determinar la fiabilidad de la silla. Es comprensible entender, que durante la realización de dichas pruebas, si se encontraba algún fallo en el sistema, se solucionaba después de terminar con la prueba como sucedió con el primer voluntario en probar la silla, Justo. Las pruebas realizadas con los estudiantes se llevaron a cabo sin ningún contratiempo y el sistema funcionó correctamente.

En este capítulo se comentarán las pruebas realizadas una vez terminado todo el desarrollo hardware. Al observar problemas en el funcionamiento del sistema, se optó por modificar el software, como se vio en el capítulo 3. Tras dicha modificación, se volvieron a realizar mas pruebas, esta vez con estudiantes de la escuela.

4.1 Pruebas

En este apartado se expondrán las primeras pruebas realizadas con la silla y los fallos encontrados durante el transcurso de dichas pruebas.

4.1.1 Preparación

Una vez se terminó la interfaz de entrada, se sacó la silla del laboratorio y se realizo un primer recorrido por la segunda planta de la Escuela Superior de Telecomunicaciones para comprobar su funcionamiento. Tras esta primera prueba, se tomó nota de lo siguiente: Al intentar virar la silla mientras se avanzaba, había ocasiones en las que no recibía la orden porque la plataforma *Arduino* estaba demasiado ocupada comprobando los sensores de ultrasonido y no tenía tiempo para controlar el joystick. El antiguo programa que se encargaba del control de los sensores de ultrasonido era muy denso y cuando la plataforma *Arduino* recorría toda esta función, no tenía tiempo para encargarse del control del joystick por lo que no reaccionaba al recibir una orden del sensor de contacto. Como el botón de parada funcionaba perfectamente y la silla se dirigía bien, solicitamos la ayuda de un voluntario para realizar una prueba.

4.1.2 Primera prueba

La primera prueba llevada a cabo fue con Justo, el voluntario que se ofreció a probar la silla. El recorrido se realizó por la segunda planta de la Escuela Superior de Ingenieros de Telecomunicación. Se le explicó la forma de dirigir la silla, y se le dio total libertad para manejarla, con el objetivo de que nos comentara sus impresiones y sus opiniones acerca de como mejorarla.

La forma de explicar el funcionamiento de la silla era el siguiente: Se realizaban los movimientos con la silla mientras se le explicaba que botones pulsar. Se comenzaba con el

movimiento hacia adelante, siguiendo por el giro izquierda, giro derecha y marcha atrás. Cuando se terminaba de explicar un movimiento, se le solicitaba que presionase el botón de parada, haciendo especial énfasis sobre esta orden puesto que es la más importante.

Tras esta primera prueba, el voluntario nos expresó su deseo de desactivar los sensores de ultrasonido, puesto que el botón de parada funcionaba perfectamente y el quería parar la silla por propia voluntad. Además, tras otra serie de pruebas que se realizaron sin el voluntario por la segunda planta, se verificó que la plataforma *Arduino* se encargaba de controlar demasiados sensores por lo que en determinadas ocasiones no recibía bien las órdenes. Mientras se pensaba en el modo de desactivar los sensores de ultrasonido, se realizó una prueba en una pista de baloncesto y se trazó un circuito como se muestra en la siguiente figura (Figura 4-1). Durante esta prueba se le dejó total libertad para que pudiera adaptarse a las órdenes de la silla.

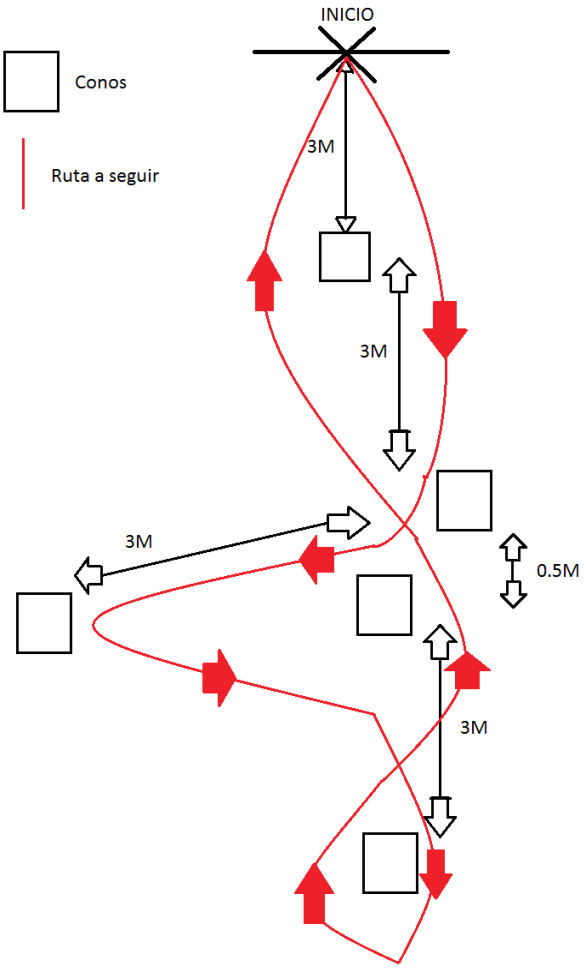


Figura 4-1: Esquema del primer circuito

En esta prueba se detectó un nuevo fallo: Las conexiones con los sensores de ultrasonido fallaban, puesto que al intentar encender la silla, había muchas ocasiones en las que ésta no llegaba a encenderse completamente porque detectaba algún sensor de ultrasonido sin conexión. El programa que llevaba incorporado la plataforma *Arduino* necesitaba detectar los tres sensores de ultrasonidos conectados, por lo que si algún sensor estuviera desconectado, no se iniciaría el programa.

El resto de movimientos que se realizaban eran satisfactorios y la silla se movía sin mayores problemas, pero aún así, era necesario solucionar los fallos anteriormente comentados.

Al tener que quitar los sensores de ultrasonido, también habría que modificar parte del código que tenía la silla por lo que antes de realizar ésta labor se realizó un diagrama de flujo del código para entenderlo mejor, esto se vio en el apartado 3.1.

Antes de la supresión de los sensores de ultrasonido y la simplificación del programa de control, se realizó otra prueba más en un garaje, alejado de las paredes, donde los sensores no causaran ningún conflicto y se simplificó el recorrido del circuito (Figura 4-2).

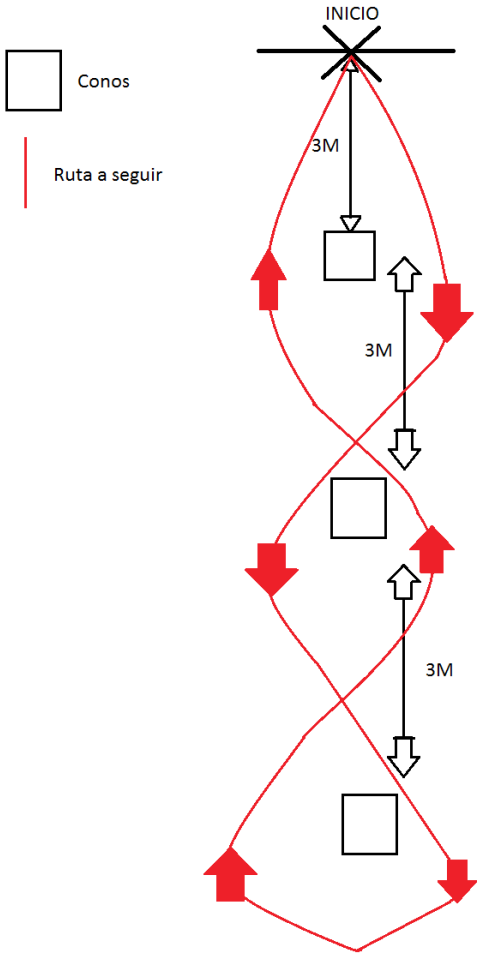


Figura 4-2: Esquema del segundo circuito

Ésta vez, en lugar de los habituales conos se dibujó con tiza los puntos que había que sortear para que los sensores de ultrasonido no molestaran al pasar cerca de ellos.

Aún así, había algunos momentos en los que la silla no recibía las órdenes para virar hacia la izquierda o derecha mientras se avanzaba, además de que en algunas ocasiones la silla se bloqueaba y no recibía ninguna orden por lo que se decidió que hasta que no estuviera perfecta, no se volvería a realizar otra prueba.

Se desactivaron los sensores de ultrasonido. Esto, solucionó dos problemas: La activación del sistema y la acción de virar mientras se avanzaba.

Cuando se completó el nuevo software de la silla y se verificó que la silla funcionaba al 100% sin ningún fallo, se realizaron otra serie de pruebas cronometradas con alumnos de la Universidad para comprobar cuanto tiempo tardaban en adaptarse a los comandos de la silla.

4.2 Pruebas con el nuevo programa

Antes de salir del laboratorio y empezar a manejar la silla con el motor encendido, se realizaron las pruebas visualizando en el monitor *Serial* como respondía el sistema y ver si seguía bloqueándose. Se comprobó que mediante el nuevo programa, el sistema fluía con eficacia y rapidez sin bloquearse en ningún momento por lo que se empezó a probarla en el exterior ya con el motor encendido y se verificó que la silla funcionaba al 100% sin ningún error.

Una vez comprobada su eficacia, se realizaron las pruebas. Para comprobar esto, se realizó el circuito de la figura 4-2 y se cronometró el tiempo que tardaban en realizarlo.

La primera prueba la llevaron a cabo Lázaro Batuecas Rodríguez y Samuel Nafría de Blas que se adaptaron con rapidez a los comandos de la silla. Se les explicó el funcionamiento de ésta y se puso mucho énfasis en el botón que realizaba la acción de parada para que lo tuvieran perfectamente claro. Una vez hecho esto, Lázaro inició la prueba y se le concedieron cinco minutos para entrenarse libremente. Después realizó una vuelta completa al circuito de la figura anterior. Cuando terminó, se sentó el otro voluntario siguiendo el mismo procedimiento: Se le dejó un tiempo para adaptarse y luego recorrió el circuito por completo. Para realizar el resto de vueltas se fueron turnando uno a uno. Los tiempos realizados por Lázaro y Samuel se muestran en la siguiente tabla:

	Samuel	**Lázaro**
Tiempo 1° Vuelta	1’22’’	1’21’’
Tiempo 2° Vuelta	1’20’’	1’16’’
Tiempo 3° Vuelta	1’14’’	1’12’’

Tabla 4-1: Primera prueba realizada con estudiantes

Ambos voluntarios tardaron un tiempo muy reducido en realizar las vueltas y en cada vuelta nueva realizada el tiempo mejoraba considerablemente.

La siguiente prueba se realizo con 5 voluntarios: Alberto de la Peña, Justino Rodríguez, Rubén Benito, Ismael Sanz, Jorge Jiménez y Mario Curiel. Se procedió de la misma forma que la primera vez: se les enseñó a manejarla haciendo énfasis en el botón de parada y se cronometró el tiempo que tardaba cada uno en completar el circuito. Los tiempos en esta prueba figuran en la tabla 4-2.

	Alberto	Justino	Rubén	Ismael	Jorge	Mario
Tiempo 1° Vuelta	1’37	2’49’’	2’40’’	2’20’’	1’30’’	3’00’’
Tiempo 2° Vuelta	1’15’’	1’10’’	1’17’’	1’06’’	1’25’’	1’49’’
Tiempo 3° Vuelta	1’15’’	1’20’’	1’18’’	1’07’’	1’23’’	1’12’’

Tabla 4-2: Segunda prueba realizada con estudiantes

En esta prueba los voluntarios tardaron una vuelta en acostumbrarse a los comandos de la silla. En el caso de Mario, la primera vuelta le llevó más tiempo porque tendía a pulsar muy rápido los botones y movía mucho el pie realizando una acción que el no quería. Ya en la segunda vuelta, los pulsó con más calma y tardó mucho menos en realizar el circuito aunque hubo alguna vez que volvió a tener el problema anteriormente comentado. Ya en la tercera vuelta, la manejó perfectamente sin mayores contratiempos. El resto de voluntarios necesitaban una vuelta para adaptarse a la silla, a partir de la primera vuelta, la manejaban sin problema.

La tercera prueba realizada fue con: Miguel Galindo Pérez, Jorge Hernández de Benito y Rodrigo Pérez. Los tiempos en realizar el circuito (Tabla 4-3) fueron parecidos a los anteriores: Tardaron en adaptarse a la silla una vuelta. A Rodrigo le costó una vuelta más que al resto pero en definitiva, se puede llegar a la conclusión de que no se necesita mucho tiempo para manejar los comandos y por lo general la gente se adapta bastante rápido a la silla.

	Miguel	Jorge	Rodrigo
Tiempo 1° Vuelta	1’47’’	2’56’’	2’25’’
Tiempo 2° Vuelta	1’00’’	1’25’’	2’00’’
Tiempo 3° Vuelta	1’00’’	1’06’’	1’28’’

Tabla 4-3: Tercera prueba realizada con estudiantes

Con estos estudiantes, se dio por finalizado este apartado, aún así, hubo mas gente que se ofreció voluntaria a probar la silla expresando su opinión acerca de la misma. En conclusión, no hubo nadie insatisfecho con el sistema y todos se adaptaron a la silla en la primera vuelta dada y el tiempo libre de entrenamiento que se les ofrece. En la siguiente tabla

(Tabla 4-4) se muestra los resultados obtenidos por las nuevas personas invitadas a probar la silla.

	Javier	Sergio
Tiempo 1° Vuelta	2’10’’	1’30’’
Tiempo 2° Vuelta	1’10’’	1’25’’
Tiempo 3° Vuelta	1’17’’	1’16’’

Tabla 4-4: Pruebas restantes con voluntarios

En la figura 4-3 se puede ver un ejemplo de las pruebas realizadas con los estudiantes durante la ejecución del recorrido. Durante el circuito, el sujeto recorría los conos en zig-zag. Al llegar al tercer cono, volvía al punto de inicio realizando de nuevo el zig-zag entre los conos.

Figura 4-3: Recorrido del circuito

En la siguiente figura (Figura 4-4), podemos ver la media de cada sujeto en realizar las tres vueltas. De los 11 sujetos, 6 se encuentran en un tiempo entre los 1’20’’que es el tiempo que tardan todos en realizar la vuelta tras el periodo de adaptación. Las personas que la primera vuelta les ha costado el aprendizaje de los comandos de la silla, hace que se eleve un poco mas su media pero al final, en la tercera vuelta, acaban realizando un tiempo aproximado de 1’20’’, lo que indica una rápida adaptación a las ordenes de la silla, como puede verse en la figura 4-5. En la tercera vuelta todos los voluntarios consiguen realizar el recorrido en 1’20’’ aproximadamente.

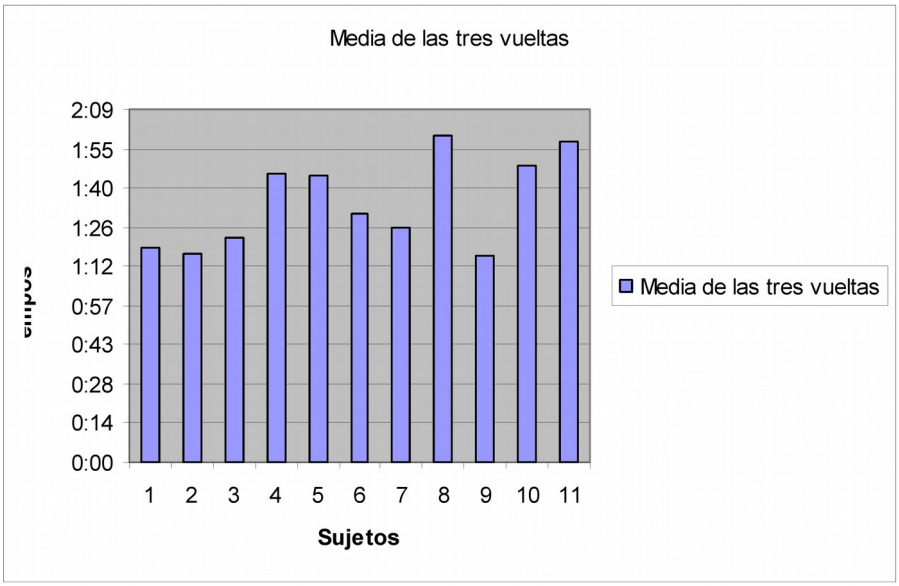


Figura 4-4: Media en segundos de los primeros 11 individuos

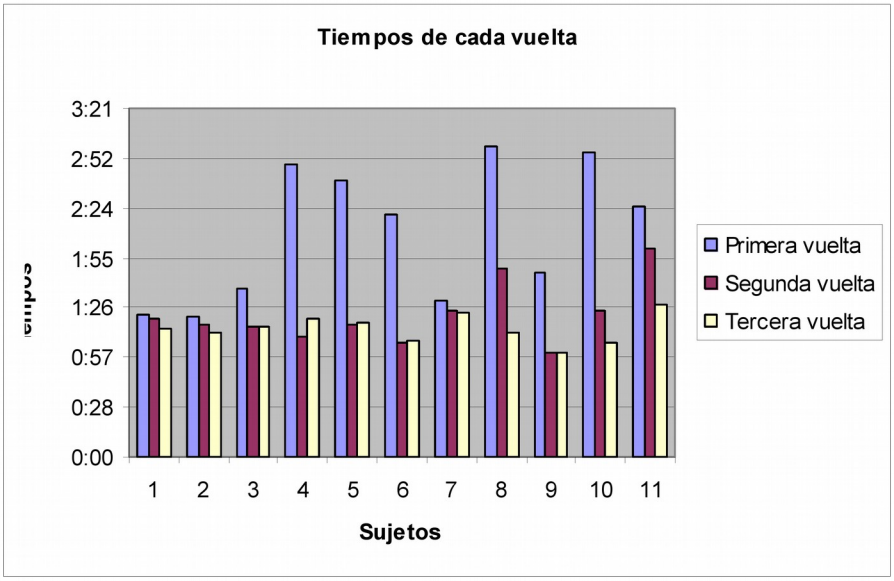


Figura 4-5: Gráfico de los tiempos de cada vuelta

Como puede observarse en la figura 4-6, existe una clara mejoría en los tiempos entre la primera vuelta y la segunda vuelta de casi un minuto. Ésta gráfica, muestra la media de los tiempos realizados en la primera vuelta, la segunda y la tercera entre los once primeros voluntarios que se presentaron a probar la silla.

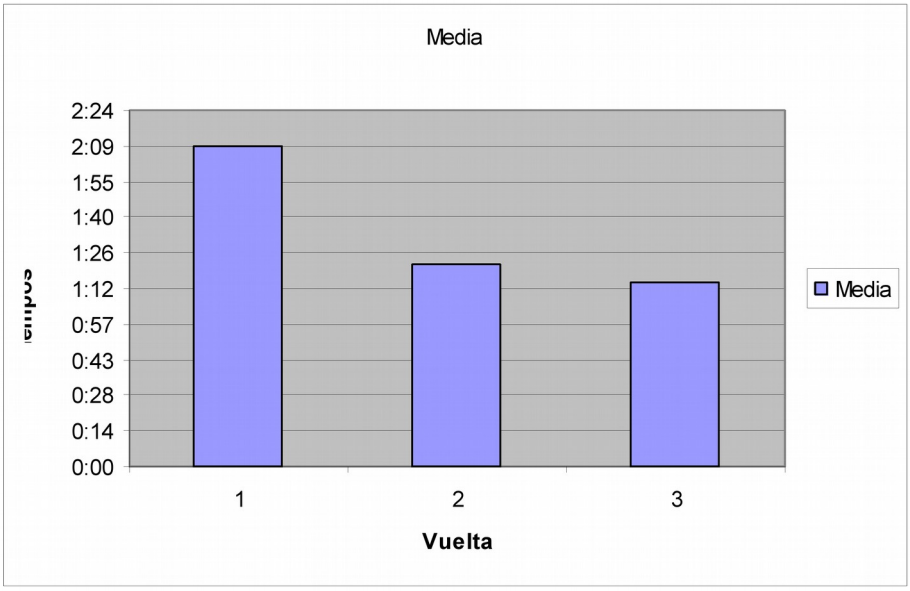


Figura 4-6: Media del tiempo

Podemos establecer como conclusión final que todos los voluntarios consiguen adaptarse correctamente a los comandos de la silla en la realización de la segunda vuelta.

4.3 Conclusiones

En este capítulo se ha visto el desarrollo de las pruebas realizadas: Desde las primeras pruebas realizadas antes de modificar el software de la silla hasta las últimas pruebas realizadas con estudiantes.

Mas detalladamente, se han comentado los fallos detectados tras las primeras prueba y como en las pruebas finales con los estudiantes se solventaron completamente.

Finalmente, se realizaron una serie de pruebas con estudiantes de la universidad y se estudió el tiempo que tardaban en adaptarse a los comandos de la silla realizando las medias de estos tiempos para verificar la conclusión obtenida.

Capítulo 5: Conclusiones y líneas futuras

5.1 Conclusiones

Al empezar con este proyecto, se realizó una introducción donde se puede sacar como conclusión que existe una población elevada que no puede mover las extremidades superiores ni inferiores por lo que resulta bastante interesante para poder ayudar a estas personas. Se verificó como existen también individuos con enfermedades degenerativas que les impiden mover extremidades superiores e inferiores pero aun así, son capaces de mover los tobillos. Este proyecto esta destinado para éstos individuos.

Entrando en capítulos posteriores, se ha expuesto el diseño de la nueva interfaz de entrada que posee la silla y como se procedió para completar su desarrollo. También se explicó el modo de diseñar un nuevo mecanismo para poder controlar el joystick de otra silla de ruedas diferente.

En cuanto al software, es el capítulo mas extenso puesto que sirve de aclaración de dudas acerca de como se desarrollo todo el programa, como se realizo para empezar un diagrama de flujo de aclaración y como se prosiguió con el primer programa, y el porque de desarrollar un segundo programa para mejorar el rendimiento del sistema y solucionar los fallos que poseía el primer programa.

Para terminar, se invito a una serie de personas a familiarizarse con este sistema para que nos expresaran su opinión acerca del proyecto y comprobar cuan rápido se adaptaban a la silla. La conclusión obtenida fue satisfactoria puesto que todas las personas que la manejaron se adaptaron con bastante rapidez y les gustó su manejo.

5.2 Líneas futuras

Para mejorar todo lo realizado en este proyecto, se ha pensado en la posibilidad de incorporar un modulo *GSM* de *Arduino* para posibilitar llamar a emergencias en el caso de ser necesario. Se podría instalar otro botón para ello en el reposapiés izquierdo o también, se podría incorporar una función por la que si lo mantienes pulsado el botón de parada durante un determinado tiempo (Por ejemplo 5 segundos), llamar a emergencias.

En el aspecto de mejorar la silla que se encuentra en el laboratorio de Bioingeniería y Electrónica, se producen situaciones en las que durante el manejo de la silla, la velocidad de esta a la hora de avanzar se incrementa. Esto suele suceder después de realizar algún giro hacia la izquierda o hacia la derecha. Tras verificar que en el programa *Arduino* se le concede el mismo valor al servomotor de empuje para desplazar la silla hacia delante, se pensó que este incremento de velocidad puede deberse a un suceso mecánico propio del diseño de ese mecanismo por el cual, después de un giro, el joystick tendría mas libertad para realizar el movimiento de avance. De esta forma, al indicar la orden de avance, el joystick tiene mas libertad para moverse hacia delante por lo que la velocidad de la silla se incrementa.

Para solucionar este problema, seria interesante proporcionar una orden para disminuir el valor del servomotor de empuje y con ello, la velocidad de la silla. De hacerse, se podría realizar instalando otro botón en el reposapiés izquierdo. Si se quiere continuar con la idea de llamar a emergencias, lo más razonable sería utilizar el botón de parada para realizarlo.

Por último, otra idea para mejorar la silla que además sería bastante simple de realizar: Proporcionar un direccionamiento a la silla mientras se esta dando marcha atrás porque al ir hacia atrás, no hay ningún comando en la silla para dirigirla y realizar esto no sería mas que copiar el código que tiene el programa *Arduino* para virar mientras se avanza y darle forma.

REFERENCIAS

[1] “Laboratorio de electrónica y bioingeniería. Universidad de Valladolid”. http://www.biolab.tel.uva.es/index.html.

[2] Juan Jose Ortega. “Avances en tecnologías de rehabilitación: Comunicación aumentativa y apoyo a la movilidad. Proyecto fin de carrera de ingeniero de telecomunicación. E.T.S.I Telecomunicación, Universidad de Valladolid.

[3] Pablo Luis Mayo Herguedas. “Desarrollo de un sistema de ayudas y soluciones en tecnologías de rehabilitación para ambientes controlados (astrac) mediante el empleo de sensores y tecnología inalámbrica wi-fi“. Proyecto fin de carrera de ingeniero técnico de telecomunicación: Sistemas de telecomunicación. E.T.S.I. Telecomunicación, Universidad de Valladolid. Marzo 2011.

[4] Alonso Alonso, Ramón de la Rosa, Albano Carrera, Alfonso Bahillo, Ramón Durán and Patricia Fernández. “A control system for robots and wheelchairs: its application for people with severe motor disability”, Mobile Robots/Book 3, Ed.: In Tech. pp. 105-126.

[5] Vishay Semiconductors. “Reflective Optical Sensor with Transistor Output”. Datasheet.

[6]”Músculos de la expresión facial” http://es.slideshare.net/sindyvargas/msculos-de-la-expresin-facial

[7] “Fritzing” http://sourceforge.net/projects/fritzing.mirror/

[8] “Arduino” http://arduino.cc/en/Main/Software

[9] “Tipos de memoria Arduino” http://www.ajpdsoft.com/modules.php?name=News&file=article&sid=571

[10] “Organización mundial de la salud” http://www.who.int/mediacentre/factsheets/fs352/es/

[11] “Boletín Informativo del Instituto Nacional de Estadística” http://www.ine.es/revistas/cifraine/1009.pdf

[12] “Las tecnologías como ayuda” http://www.compromisorse.com/rse/2014/07/10/las-tecnologias-adaptadas-permiten-trabajar-al-52-de-las-personas-con-discapacidad-/

[13] “Tecnologías para el apoyo a discapacitados” http://www.uco.es/grupos/eatco/automatica/ihm/descargar/discapacitados.pdf

[14] “Discapacidad física debido a enfermedad degenerativa” http://www.efdeportes.com/efd98/discap.htm

[15] MSE-S110.2.pdf

[16] Philips Semiconductors. “HEF40106B gates Hex inverting Schmitt trigger” Datasheet.

[17] Microsystems engineering. “SRF08 Ultra sonic range finder”. Datasheet

[18] Enrique Palacios y otros, “Microcontrolador PIC16F84, Desarrollo de proyectos”, Ed Ra-Ma

[19] Tesis doctoral de Albano: Innovaciones en sistemas e interfaces humano-máquina: aplicación a las tecnologías de rehabilitación

[20] “Codificador N74LS148”. Datasheet

[21] “Atach Interrupt Arduino” http://arduino.cc/en/pmwiki.php?n=Reference/AttachInterrupt

[22] “Comunicación Arduino-PC en Windows” http://www.arduino.cc/en/pmwiki.php?n=Guide/Windows

[23] “Tutorial Arduino” http://www.arduino.cc/es/pmwiki.php?n=

APÉNDICE CÓDIGO ANTIGUO ARDUINO

```
#include <Servo.h>
#include <Wire.h>
#include <Sonar_srf08.h>

Sonar_srf08 MySonar;
#define CommandRegister 0x00
int Direccion_izq = 248;
int Direccion_drch = 249;          // Variables para la comunicación
con los
sensores SRF08
int Direccion_centro = 250;
#define ResultRegister 0x02
int DEBUG = 1;
char unit = 'c';              // Centimetros
float anterior = 0;     // Variable que almacenará la información
recogida en la muestra anterior del sensor menos la muestra actual.
//Array para almacenar las lecturas de los sensores
float lectura_izq[5] = {0, 0, 0, 0, 0};
float lectura_drch[5] = {0, 0, 0, 0, 0};
float lectura_centro[5] = {0, 0, 0, 0, 0};
//Array para almacenar el tiempo entre los sensores
float tiempo_izq[4] = {0, 0, 0, 0};
float tiempo_drch[4] = {0, 0, 0, 0};
float tiempo_centro[4] = {0, 0, 0, 0};
//Variable de tiempo auxiliares
float tiempo_izq_aux = 0;
float tiempo_drch_aux = 0;
float tiempo_centro_aux = 0;

//Variable de velocidad media hacia el obstaculo
float vel_izq_media = 0;
float vel_drch_media = 0;
float vel_centro_media= 0;
//Variables de velocidad instantanea hacia el obstaculo
float vel_izq_ins = 0;
float vel_drch_ins = 0;
float vel_centro_ins = 0;
boolean comp_lat = false;          //Variable que indicará si el
autoguiado está
activado
boolean paralelo = false;     //Variable que indicá si la silla ya está
paralela a la pared.
char direccion_mov;     //Indica que movimiento es, 'd' adelante, 'l'
izquierda y 'r' derecha
int Velocidad_Min = 11;
int Velocidad_Med = 73;       // Velocidades (posición del servo de
velocidad.)
int Velocidad_Max = 135;
int Giro_Min = 48;
int Giro_Med = 74;                 //Giro(Posición del servo de giro)
int Giro_Max = 92;
boolean estadoizq = false;    //Variables para el estado de los CNY70
boolean estadodch = false;
boolean movimiento = false;   //Indica si la silla está en movimiento o
no
boolean adelanteatras = false;      //Indca si el movimiento es de
avance
boolean EstParada = false;    //Indica si acaba de parar la silla
//Variable para indicar el tiempo de los guiños
```

```
unsigned long tiempoizq = 0;
unsigned long tiempodch = 0;
unsigned long tiempoizqt = 0;
unsigned long tiempodcht = 0;
int estado=0;                  //estado 0= primera detección; 1=
guiñoizq 1ª
detección; 2= guiñodrch 2ª detección
int ledI=8;                          //indicador luminoso izquierdo
int ledC=9;                          //indicador luminoso central
int ledD=10;                         //indicador luminoso derecho

int funciona=7;               //inciador luminoso de funcionamiento
int giro=0;                          //Variable para enviar a los
servos.
int NVelocidad = 0;
Servo velocidad;              //Servo de control de la velocidad
Servo direccion;  //Servo de control de la direccion de movimiento
void setup() {
      pinMode(ledD, OUTPUT);         //Salida de los LEDs
      pinMode(ledI, OUTPUT);
      pinMode(ledC, OUTPUT);
      pinMode(funciona,OUTPUT);
      if (DEBUG){
            Serial.begin(9600);
      }
      Serial.println("Iniciando configuración");
      velocidad.attach(11);          //Configuración de los servos
      direccion.attach(12);
      velocidad.write(Velocidad_Med);
      direccion.write(Giro_Med);
      NVelocidad = Velocidad_Med;
      MySonar.connect();                  //Conexión con los sensores
      delay(200);
      Lectura_Sensores();
      Lectura_Sensores();
      Lectura_Sensores();
      Lectura_Sensores();
      Lectura_Sensores();
      Serial.println("Configuración Finalizada");
      digitalWrite(funciona,HIGH);        //Enciende el indicador de
      funcionamiento
      attachInterrupt(0, guinoizq, CHANGE); //Habilitar interrupciones
      por los
      pines 2 y 3
      attachInterrupt(1, guinodch, CHANGE);
}


/* La función loop() comprueba los guiños que se ha producido
y llama a las funciones encargadas del movimiento de los servos
además llama a las funciones para enviar el comando de lectura de
sensores
y comprobación de un posible choque.
*/


void loop() {
//Comprueba si se ha producido una parada(parpadeo) y resetea algunas
variables.
```

```
if(EstParada ==true){

      tiempodch=0;
      tiempodcht=0;

      tiempoizq=0;
      tiempoizqt=0;
      estadodch=false;
      estadoizq=false;
      EstParada = false;
      estado = 0;
      Serial.println("Reset");

}
Lectura_Sensores();
comprobacion_choque(direccion_mov);
attachInterrupt(0, guinoizq, CHANGE);
attachInterrupt(1, guinodch, CHANGE);
delay(50);
detachInterrupt(0);
detachInterrupt(1);
if(tiempoizqt > 50 && tiempodcht < 50)
{
      //Guiñoizq
      estadoizq = false;
      estadodch = false;
      Serial.println("Se detecto guiño izquierdo");
      digitalWrite(ledI, LOW);
      digitalWrite(ledC, LOW);
      digitalWrite(ledD, LOW);
      digitalWrite(ledI, HIGH);
      switch(estado){
      case 0:
            estado = 1;
            comp_lat = true;
            break;
      case 1:
            movimientoservo(1);
            break;
      case 2:
            movimientoservo(2);
            break;
      }
      tiempodch=0;
      tiempodcht=0;
      tiempoizq=0;
      tiempoizqt=0;
      estadodch=false;
      estadoizq=false;
      delay(50);
}
if(tiempoizqt < 50 && tiempodcht > 50)

{
      //Guiño Derecho
      estadoizq = false;
      estadodch = false;
      Serial.println("Se detecto guiño derecho");
      digitalWrite(ledI, LOW);
      digitalWrite(ledC, LOW);
      digitalWrite(ledD, LOW);
```

```
	digitalWrite(ledD, HIGH);
	switch(estado){
	case 0:
		estado = 2;
		comp_lat = true;
		break;
	case 1:
		movimientoservo(3);
		break;
	case 2:
		movimientoservo(4);
		break;

	}
	tiempodch=0;
	tiempodcht=0;
	tiempoizq=0;
	tiempoizqt=0;
	estadodch=false;
	estadoizq=false;
	delay(50);
	}
}


/* Las funciones guinoizq() y guinodch () son
equivalente, son lllamadas cuando hay un cambio en
los sensores, primero comprueban si hay parpadeo, si es que
si se llama a la funcion movimientoservo() para parar si no
se procede a controlar el tiempo de guiño
*/


void guinoizq()
{
	if(digitalRead(3)==LOW){
		detachInterrupt(0);
		detachInterrupt(1);
		movimientoservo(0);
		return;
	}
	else{
		estadoizq = !estadoizq;

		if(estadoizq == true)
		{
			tiempoizq = millis();
		}
		else
		{
			tiempoizqt = millis() - tiempoizq;
		}
	}
}
void guinodch()
{
	if(digitalRead(2)==LOW){
		detachInterrupt(0);
		detachInterrupt(1);
```

```
            movimientoservo(0);
            return;
      }
      else{
            estadodch = !estadodch;
            if(estadodch == true)
            {
                  tiempodch = millis();
                  //Serial.println("El estado de dch es LOW");
            }
            else
            {
                  tiempodcht = millis() - tiempodch;
                  //Serial.println("El estado dch es HIGH");
            }
      }
}



/*
movimientoservo se encarga de comprobar la instrucción que ha llegado
y
mueve los servos de forma conveniente
*/



void movimientoservo(int tipomov){
digitalWrite(ledI, LOW);
digitalWrite(ledC, LOW);
digitalWrite(ledD, LOW);
switch(tipomov){

case 0: //Parada
      Serial.println("Parada");
      velocidad.write(Velocidad_Med);
      direccion.write(Giro_Med);
      digitalWrite(ledC, LOW); //Apaga el led indicador central

      digitalWrite(ledD, LOW); //Apaga el led indicador derecho
      digitalWrite(ledI, LOW); //Apaga el led indicador izquierdo
      EstParada = true;
      movimiento = false;
      adelanteatras = false;
      direccion_mov = 'p';
      Velocidad_Max = 135;
      Velocidad_Min = 11;
      estado = 0;
      giro = Giro_Med;
      NVelocidad = Velocidad_Med;
      vel_izq_ins = 0;
      vel_drch_ins = 0;
      vel_centro_ins = 0;
      break;
case 1: //Avanza
      if(movimiento == false){
      if(lectura_centro[4]>9.0){
      Serial.println("Avanza");
      velocidad.write(Velocidad_Med);
      delay(50);
```

```
        direccion.write(Giro_Med);
        delay(50);
        velocidad.write(Velocidad_Max);
        digitalWrite(ledD,HIGH); //Enciende el led derecho para indicar
        el
        movimiento que se esta ejecutando
        digitalWrite(ledI,HIGH); //Enciende el led izquierdo para
        indicar
        el movimiento que se esta ejecutando
        giro = Giro_Med;
        NVelocidad = Velocidad_Max;
        direccion_mov = 'd';
        movimiento = true;
        adelanteatras = true;
        comp_lat = false;
        }
        estado = 0;
        }else{
        if(adelanteatras == true){
        if(giro > Giro_Min){
        Serial.println("resto 5");
        giro -= 5;
        }
        Serial.print("Giro un poco a la izquierda\tGiro: ");
        Serial.println(giro);
        direccion.write(giro);
        }


        }
        break;
case 2: //Giro izquierda
        if(movimiento == false){
        if(lectura_izq[4] < 9.0){
        break;
        }
        Serial.println("Giro izquierda");
        digitalWrite(ledI,HIGH);
        velocidad.write(Velocidad_Med);
        delay(50);
        direccion.write(150);
        delay(50);
        velocidad.write(Velocidad_Min+20);
        movimiento = true;
        direccion_mov = 'l';
        estado = 0;
        comp_lat = false;
        NVelocidad = Velocidad_Min+20;
        }else{
        if(adelanteatras == true){
        if(giro > 40){
        Serial.println("resto 5");
        giro -= 5;
        }
        Serial.print("Giro un poco a la izquierda\tGiro: ");
        Serial.println(giro);
        direccion.write(giro);
        }
        }
        break;
case 3: //Giro Derecha
```

```
        if(movimiento == false){
        if(lectura_drch[4] < 9.0){
        break;
        }
        Serial.println("Giro derecha");
        digitalWrite(ledD,HIGH);
        velocidad.write(Velocidad_Med);
        delay(50);
        direccion.write(150);
        delay(50);
        velocidad.write(Velocidad_Max-10);
        movimiento = true;
        direccion_mov = 'r';
        estado = 0;

        comp_lat = false;
        NVelocidad = Velocidad_Max-10;
        }else{
        if(adelanteatras == true){
        if(giro < 90){
        Serial.println("sumo 5");
        giro += 5;
        }
        Serial.print("Giro un poco a la derecha\tGiro: ");
        Serial.println(giro);
        direccion.write(giro);
        }
        }
break;
case 4: //Atrás
        if(movimiento == false){
        Serial.println("Atras");
        digitalWrite(ledC,HIGH);
        velocidad.write(Velocidad_Med);
        delay(50);
        direccion.write(60);
        delay(50);
        velocidad.write(Velocidad_Min+30);
        movimiento = true;
        estado = 0;
        comp_lat = false;
        }else{
        if(adelanteatras == true){
        if(giro < 90){
        Serial.println("sumo 5");
        giro += 5;
        }
        Serial.print("Giro un poco a la derecha\tGiro: ");
        Serial.println(giro);
        direccion.write(giro);
        }
        }
        break;
        }
}


/*
Envia los comando necesarios a los sensores srf08 para realizar
una medida y lo guarda en el array de medidas.
*/
```

```
void Lectura_Sensores(){
int l=0;

float lectura_aux_izq;
float lectura_aux_drch;
float lectura_aux_centro;
// Unidades en las que se va a medir la distancia
for(int i=0;i<4;i++){
      //lectura_izq[i]=lectura_izq[i+1];
      //lectura_drch[i]=lectura_drch[i+1];
      lectura_centro[i]=lectura_centro[i+1];
}
for(int i=0;i<3;i++){
      //tiempo_izq[i]=tiempo_izq[i+1];
      //tiempo_drch[i]=tiempo_drch[i+1];
      tiempo_centro[i]=tiempo_centro[i+1];
}
//Sensor de la izquierda
do{
      l++;
      MySonar.setUnit(CommandRegister, Direccion_izq, unit);
      delay(70);
      MySonar.setRegister(Direccion_izq, ResultRegister);
      // Lectura de datos.
      lectura_aux_izq = MySonar.readData(Direccion_izq, 2);
}while(lectura_aux_izq==0 || l==4 );
//if(lectura_aux_izq < 62.0 || direccion_mov == 'c'){
if(true){
      for(int i=0;i<4;i++){
            lectura_izq[i]=lectura_izq[i+1];
      }
      for(int i=0;i<3;i++){
            tiempo_izq[i]=tiempo_izq[i+1];
      }
      lectura_izq[4] = lectura_aux_izq;
      tiempo_izq[3] = millis() - tiempo_izq_aux;
      calculo_velocidad('i');
      tiempo_izq_aux=millis();
}
if(direccion_mov=='l'){
      comprobacion_choque(direccion_mov);
}
//Sensor de la derecha
l = 0;
do{

      l++;
      MySonar.setUnit(CommandRegister, Direccion_drch, unit);
      delay(70);
      MySonar.setRegister(Direccion_drch, ResultRegister);
      // Lectura de datos.
      lectura_aux_drch = MySonar.readData(Direccion_drch, 2);
      //Serial.println(lectura_drch[4]);
}while(lectura_aux_drch==0 || l==4);
//if(lectura_aux_drch < 62.0 || direccion_mov == 'c'){
if(true){
      for(int i=0;i<4;i++){
            lectura_drch[i]=lectura_drch[i+1];
      }
```

```
      for(int i=0;i<3;i++){
            tiempo_drch[i]=tiempo_drch[i+1];
      }
      lectura_drch[4] = lectura_aux_drch;
      tiempo_drch[3] = millis() - tiempo_drch_aux;
      calculo_velocidad('d');
      tiempo_drch_aux=millis();
}
if(direccion_mov=='r'){
      comprobacion_choque(direccion_mov);
}
//Sensor del centro
l=0;
do{
      MySonar.setUnit(CommandRegister, Direccion_centro, unit);
      delay(70);
      MySonar.setRegister(Direccion_centro, ResultRegister);
      // Lectura de datos.
      lectura_centro[4] = MySonar.readData(Direccion_centro, 2);
      //Serial.println(lectura_centro[4]);
}while(lectura_centro[4]==0 || l==4);
tiempo_centro[3] = millis() - tiempo_centro_aux;
calculo_velocidad('c');
tiempo_centro_aux=millis();
}
/*
La función calculo_velociad calcula la velocidad a la que nos
acercamos al
obstaculo.
*/
void calculo_velocidad(char pos){
switch(pos){
case 'i':

      vel_izq_media = 0;
      vel_izq_ins = 0;
for(int i=0;i<4;i++){
      vel_izq_media += vel_izq_media + (lectura_izq[i]-
      lectura_izq[i+1]);
}
vel_izq_ins = lectura_izq[3]-lectura_izq[4];
tiempo_izq_aux = 0;
for(int i=0;i<4;i++){
      tiempo_izq_aux += tiempo_izq[i];
}
vel_izq_media = vel_izq_media / (tiempo_izq_aux/1000);
vel_izq_ins = vel_izq_ins / (tiempo_izq[3]/1000);
return;
break;
case 'c':
vel_centro_media = 0;
vel_centro_ins = 0;
for(int i=0;i<4;i++){
      vel_centro_media += vel_centro_media + (lectura_centro[i]-
      lectura_centro[i+1]);
}
vel_centro_ins = lectura_centro[3]-lectura_centro[4];
tiempo_centro_aux = 0;
for(int i=0;i<3;i++){
      tiempo_centro_aux += tiempo_centro[i];
}
```

```
vel_centro_media = vel_centro_media / (tiempo_centro_aux/1000);
vel_centro_ins = vel_centro_ins / (tiempo_centro[3]/1000);
return;
break;
case 'd':
vel_drch_media = 0;
vel_drch_ins = 0;
for(int i=0;i<4;i++){
      vel_drch_media += vel_drch_media + (lectura_drch[i]-
      lectura_drch[i+1]);
}
vel_drch_ins = lectura_drch[3]-lectura_drch[4];
tiempo_drch_aux = 0;
for(int i=0;i<3;i++){
      tiempo_drch_aux += tiempo_drch[i];
}
vel_drch_media = vel_drch_media / (tiempo_drch_aux/1000);
vel_drch_ins = vel_drch_ins / (tiempo_drch[3]/1000);
return;
break;

default:
return;
break;
}
}
/*
comprobacion_choque comprueba si existe un posible choque y siminuye
la
velocidad de la silla
llegando incluso a pararse y tambien realiza el autoguiado.
*/
void comprobacion_choque(char tipo_movimiento){
float tiempo_choque;
switch(tipo_movimiento){
case 'd':
// Detectamos el choque por delante
if(lectura_centro[4] < 5.0 || lectura_izq[4] <= 3.0 || lectura_drch[4]
<= 3.0 ){
      movimientoservo(0);
      break;
}
tiempo_choque = lectura_centro[4]/vel_centro_ins;
if(abs(tiempo_choque) < 2 && vel_centro_ins > 0 && lectura_centro[4]
< 30){
      Velocidad_Max = (Velocidad_Max-73)*tiempo_choque/2+73;
if(Velocidad_Max<95){
      movimientoservo(0);
      return;
}
velocidad.write(Velocidad_Max);
}
if(comp_lat == true){
      if(lectura_izq[4] < 50.0){
            if(lectura_drch[4] < 50.0){
                  giro = Giro_Med;
            }
      else{
            if(lectura_izq[3] <= lectura_izq[4] && paralelo == true){
                  giro = Giro_Med;
                  paralelo = false;
```

```
            }
      if(lectura_izq[3] > lectura_izq[4]){
            paralelo = true;
            if(giro < Giro_Max){
                  giro += 5;
            }
      }

      }
}
direccion.write(giro);
delay(50);
}
if(comp_lat == true){
      if(lectura_drch[4] < 50.0){
            if(lectura_izq[4] < 50.0){
                  giro = Giro_Med;
            }
            else{
                  if(lectura_drch[3] <= lectura_drch[4] && paralelo ==
            true){
                  giro = Giro_Med;
                  paralelo = false;
            }
            if(lectura_drch[3] > lectura_drch[4]){
                  paralelo = true;
                  if(giro > Giro_Min){
                        giro -= 5;
                  }
            }
      }
}
direccion.write(giro);
delay(50);
}
break;
case 'l':
// Detectamos el choque en el giro hacia la izq
if(lectura_izq[4] < 9.0){
      movimientoservo(0);
      break;
}
if(vel_izq_ins == 0){
      vel_izq_ins = 0.01;
}
tiempo_choque = lectura_izq[4]/vel_izq_ins;
if(abs(tiempo_choque) < 2 && vel_izq_ins > 0 && lectura_izq[4]<
50.0){
      NVelocidad = (NVelocidad-73)*tiempo_choque/3+73;
if(NVelocidad > 56 || lectura_izq[4] < 30.0){
      movimientoservo(0);
return;
}
velocidad.write(NVelocidad);
}


break;
case 'r':
// Detectamos el choque en el giro hacia la derecha
if(lectura_drch[4] < 9.0){
```

```
      movimientoservo(0);
      break;
}
if(vel_drch_ins == 0){
      vel_drch_ins = 0.01;
}
tiempo_choque = lectura_drch[4]/vel_drch_ins;
if(abs(tiempo_choque) < 2 && vel_drch_ins > 0 && lectura_drch[4] <
50.0){
      NVelocidad = (NVelocidad-73)*tiempo_choque/3+73;
if(NVelocidad < 90 || lectura_drch[4] < 30.0){
      movimientoservo(0);
      return;
}
velocidad.write(NVelocidad);
}
break;
default:
break;
}
}
```

APÉNDICE CODIGO FINAL

```
#include <Servo.h>  //Librería necesaria para manejar los servos

int DEBUG = 1;

int Velocidad_Parada = 110;    // Velocidades (posición de los
servos.)  Diagrama= (1)->Declaración variables velocidad y giro
int Velocidad_avance = 20;
int Velocidad_Izq = 31;
int Velocidad_Drch = 40;
int Velocidad_Atras = 180;

int Giro_Min = 40;
int Giro_Med = 90;  //Valor dado para la parada, movimiento de avance,
atras, izquierda y derecha
int Giro_Izq = 180;
int Giro_Drch = 10;

boolean movimiento = false;      // Diagrama = (2)->Declaración
variables Booleanas
boolean adelanteatras = false;
boolean EstParada = false;  //Variable que pasa a true cuando
realizamos el parpadeo para parar la silla

int estado=0;                   //estado 0= primera detección; 1=
guiñoizq 1ª detección; 2= guiñodrch 2ª detección  (3)

int ledI=8;  //indicador luminoso izquierdo
int ledC=9;  //indicador luminoso central              //Diagrama =
(3)-> Declaración de variables usadas para los led
int ledD=10;  //indicador luminoso derecho
int funciona=7; //indicador luminoso de funcionamiento

int giro=0;      //Diagrama -> (3)        Variable que se utilizara
para indicar el valor del Servo dirección

Servo velocidad; //Servo de control de la velocidad
Servo direccion; //Servo de control de la dirección de movimiento
//Diagrama= (4)->Servos

void setup() {                              //Diagrama (S)->Inicio Void
Setup

      pinMode(ledD, OUTPUT);      //Salida de los LEDs
      pinMode(ledI, OUTPUT);
      pinMode(ledC, OUTPUT);
      pinMode(funciona,OUTPUT);    //Diagrama = (6)-> Establecer los 4
leds como salidas
      if (DEBUG){
            Serial.begin(9600);    //Diagrama = (7) -> DEBUG

      }
      Serial.println("Iniciando configuración");
        //delay(500);
        //Serial.print("  SRAM Libre: "); Serial.println(freeRam());
      velocidad.attach(11);    //Configuración de los servos
//Diagrama = (8)
      direccion.attach(12);
      velocidad.write(Velocidad_Parada);
      direccion.write(Giro_Med);
```

```
    digitalWrite(funciona,HIGH);  //Enciende el indicador de
funcionamiento

    Serial.println("Configuración Finalizada");  //Diagrama = (9) ->
Interrupciones para cualquier cambio en pines 2 y 3

}

void loop() {    //Diagrama = (loop)

      if(EstParada ==true){               //Entra aquí de haber
existido un parpadeo para resetear la silla (RT)
                    //Diagrama = (10) -> Entra aquí cuando se produce
un parpadeo y reinicia variables. ESTPARADA = TRUE en el momento del
parpadeo
            EstParada = false;
            estado = 0;
                //delay(500);
            Serial.println("RESETEO DE GUIÑOS!!");

      }
        if(digitalRead(2)==LOW && digitalRead(3)==LOW)  //Diagrama
(11) -> En cuanto pulses al botón de parada ( o se realice un
parpadeo) se llama a la función movimiento servo(0) que parará la
silla (RT)
        {
          movimientoservo(0);
        }
      if(digitalRead(2)==LOW && digitalRead(3)==HIGH)// Diagrama (12)
-> Entras en este if en el momento que pulses el botón izquierdo(RT)
      {
          //Serial.println("Estas pulsando");
          delay(20);
          if(digitalRead(3) == HIGH && digitalRead(2) == HIGH)  //
Entras en este if en el momento que sueltes el botón izquierdo(RT)
          {
            //Guiñoizq o botón izquierdo
                Serial.println("Guiñoizq");
            digitalWrite(ledI, LOW);
            digitalWrite(ledC, LOW);      //Se establece el valor de
los LED para saber en que modo estamos (RT)
            digitalWrite(ledD, LOW);
            digitalWrite(ledI, HIGH);
            switch(estado){          //Diagrama = (13)-> Llamadas a
movimiento servo según la variable estado
            case 0:
                  estado = 1;      //Entra en este case de ser la
primera vez que guiñamos el ojo izquierdo (RT)
                break;
            case 1:
                  movimientoservo(1);  //Llama a la función movimiento
servo para establecer el joystick en movimiento avance; Se interpreta
como dos guiños izqs (RT)
                  break;               //[M1]
            case 2:
                  movimientoservo(2);  //Llama a la función movimiento
servo para establecer el joystick en movimiento izquierda; Se
interpreta como guiño der.
                                              //y luego guiño izq. (RT)
                  break;               //[M2]
            }
```

```
            delay(200);
          }
      }

      if(digitalRead(3) == LOW && digitalRead(2)==HIGH)  //Diagrama
(14) -> Entras en este if cuando pulses el botón derecho (RT)
      {
          //Serial.println("Estas pulsando");
          delay(20);
          if(digitalRead(2) == HIGH && digitalRead(3)==HIGH)  //Entras
en este if cuando sueltes el botón derecho (RT)
          {

            //Guiño Derecho o botón derecho
                Serial.println("Guiñodch");
            digitalWrite(ledI, LOW);      //Se establece el valor de
los LED para saber en que modo estamos (RT)
            digitalWrite(ledC, LOW);
            digitalWrite(ledD, LOW);
            digitalWrite(ledD, HIGH);
            switch(estado){        //Diagrama de flujo->(15)
            case 0:
                  estado = 2;     //Entra en este case de ser la
primera vez que guiñamos el ojo derecho (RT)
                  break;
            case 1:
                  movimientoservo(3); //Llama a la función movimiento
servo para establecer el joystick en movimiento derecha; Se interpreta
como guiño dizq y luego derecha     (RT)
                  break;              //[M3]
            case 2:
                  movimientoservo(4); //Llama a la función movimiento
servo para establecer el joystick en movimiento hacia atrás      (RT)
                  break;              //[M4]
            }
            delay(200);
          }
      }
}

void movimientoservo(int tipomov){   //Diagrama->(16)
        digitalWrite(ledI, LOW);
      digitalWrite(ledC, LOW);
      digitalWrite(ledD, LOW);
      switch(tipomov){
            case 0:    //Parada  Diagrama->(17)      //[M0]
                  Serial.println("Parada");
                  velocidad.write(Velocidad_Parada);
                        delay(50);
                  direccion.write(Giro_Med);
                        delay(50);
                  digitalWrite(ledC, LOW);  //Apaga el led indicador
central
                  digitalWrite(ledD, LOW);  //Apaga el led indicador
derecho
                  digitalWrite(ledI, LOW);  //Apaga el led indicador
izquierdo
                  EstParada = true;         //EstParada=True para
entrar en el if del principio nada mas empezar la función loop y parar
la silla
                  movimiento = false;       //Para indicar que no hay
```

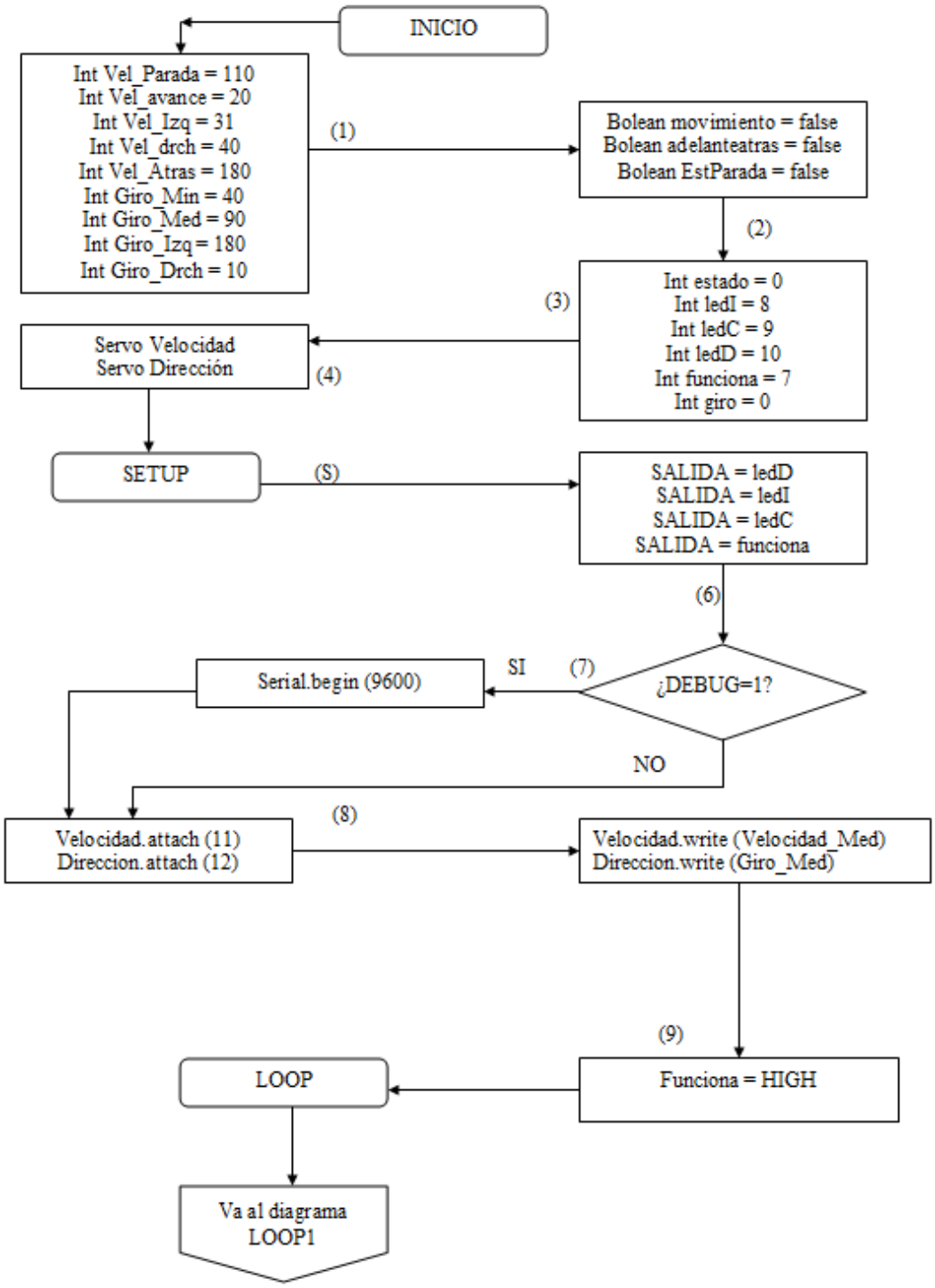
```
movimiento
                  adelanteatras = false;
                  estado = 0;
                  giro = Giro_Med;

                  break;
            case 1:    //Avanza    Diagrama->(18)        //[M1]
                  if(movimiento == false){            //Si no hay
movimiento entra aquí y pone la silla en funcionamiento avanzando
hacia delante->(RT)
                        Serial.println("Avanza");   //Diagrama(18.1)

                        velocidad.write(Velocidad_Parada);
                        delay(100);  //Estos delay pueden variar pero
se recomienda no poner mas de un segundo para que no tarde tanto (RT)
                        direccion.write(Giro_Med);
                        delay(100);
                        velocidad.write(Velocidad_avance);

                        digitalWrite(ledD,HIGH);  //Enciende el led
derecho para indicar el movimiento que se esta ejecutando->(RT)
                        digitalWrite(ledI,HIGH);  //Enciende el led
izquierdo para indicar el movimiento que se esta ejecutando->(RT)
                        giro = Giro_Med;

                        movimiento = true;
                        adelanteatras = true;

                        estado = 0;
                  }else{
                        if(adelanteatras == true){  //Si la silla esta
avanzando entonces entra en este if para girar un poco a la izquierda-
>(RT)
                              if(giro > Giro_Min){  //Diagrama(18.2)
                                    Serial.println("resto 10");
                                    giro -= 10;    //Antes estaba a 5
                                          direccion.write(giro);
                                          delay(50);

                              }
                              Serial.print("Giro un poco a la
izquierda\tGiro: ");
                              Serial.println(giro);

                        }
                  }
                  break;
            case 2:   //Giro izquierda  Diagrama->(19)     [M2]
                  if(movimiento == false){            //Si no hay
movimiento entra aquí y pone la silla en funcionamiento para girar a
la izquierda->(RT)

                        Serial.println("Giro izquierda");  //Diagrama
(19.1)
                        digitalWrite(ledI,HIGH);
                        velocidad.write(Velocidad_Parada);

                        direccion.write(Giro_Izq);
                        delay(200);
                        velocidad.write(Velocidad_Izq);
```

```
                        movimiento = true;
                        estado = 0;

                        Serial.print(" Lectura servo velocidad ");
//Segunda verificación para el servo velocidad
                        Serial.println( velocidad.read());

                  }else{
                        if(adelanteatras == true){    //Si la silla
esta avanzando entonces entra en este if para girar un poco a la
izquierda->(RT)
                              if(giro > Giro_Min){        //Diagrama
(19.2)
                                    Serial.println("resto 10");
                                    giro -= 10;  //Antes estaba a 5
                                          direccion.write(giro);
                                          delay(50);
                              }
                              Serial.print("Giro un poco a la
izquierda\tGiro: ");
                              Serial.println(giro);
                        }
                  }
                  break;
            case 3:    //Giro Derecha  Diagrama->(20)  //[M3]
                  if(movimiento == false){          //Si no hay
movimiento entra aquí y pone la silla en funcionamiento para girar a
la derecha->(RT)

                        Serial.println("Giro derecha");  //Diagrama
(20.1)
                        digitalWrite(ledD,HIGH);
                              //Cambia la velocidad del giro
derecha;
                        velocidad.write(Velocidad_Parada);
                              delay(100);
                        direccion.write(Giro_Drch);
                        delay(100);
                        velocidad.write(Velocidad_Drch);

                        movimiento = true;
                        estado = 0;

                        Serial.print(" Lectura servo velocidad ");
//Lectura servo velocidad
                        Serial.println( velocidad.read());

                  }else{
                        if(adelanteatras == true){  //Si la silla esta
avanzando entonces entra en este if para girar un poco a la derecha-
>(RT)
                              if(giro < 120){      //Diagrama (20.2)
                                    Serial.println("sumo 10");
                                    giro += 10;
                                          direccion.write(giro);
                                          delay(50);

                              }
                              Serial.print("Giro un poco a la
derecha\tGiro: ");
                              Serial.println(giro);
```

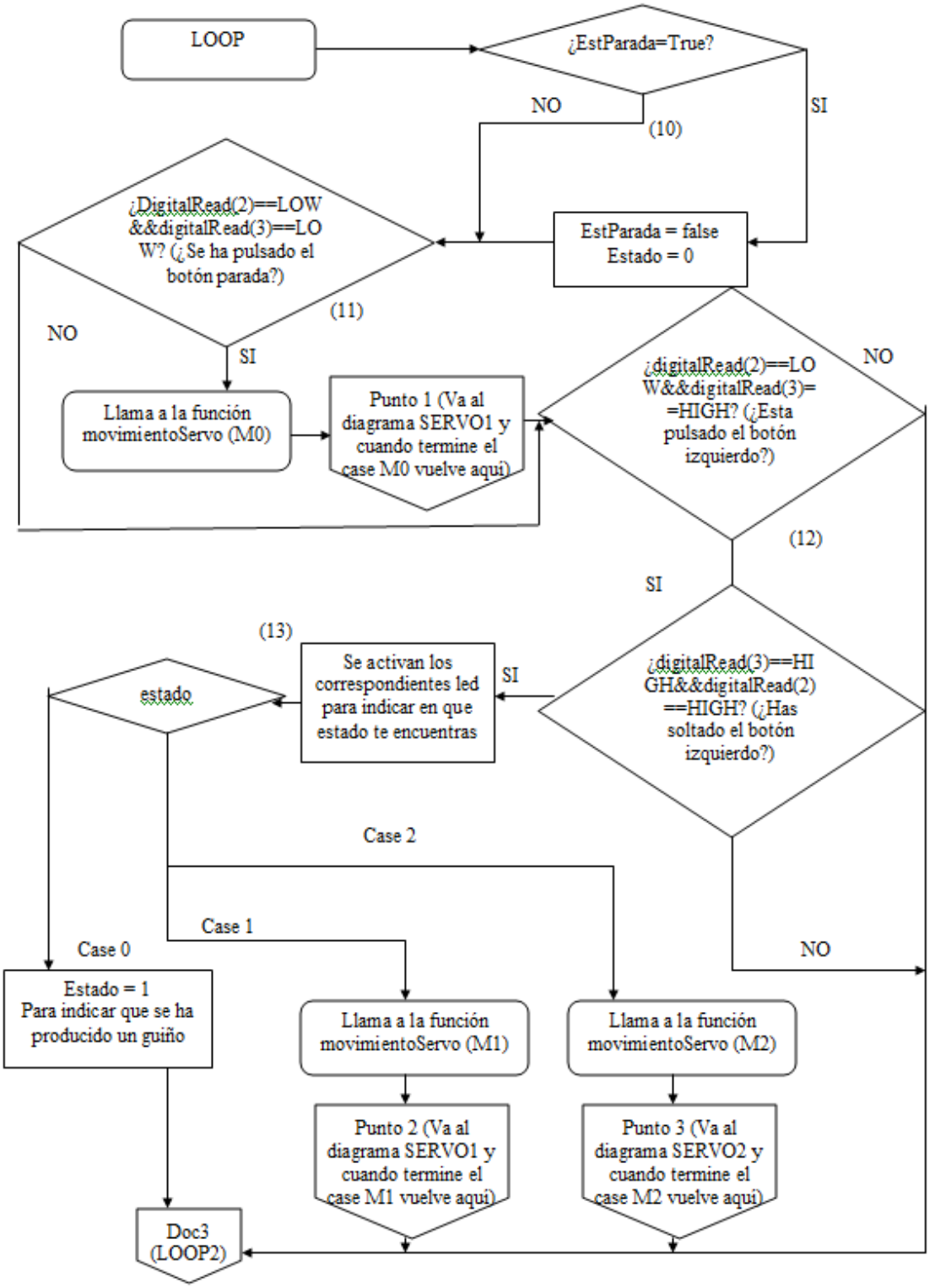
```
                        }
                  }
            break;
            case 4:   //Atrás    Diagrama->(21)    [M4]

                  if(movimiento == false){       //Si no hay movimiento
entra aquí y pone la silla en funcionamiento para dar marcha atrás-
>(RT)
                                                      //Diagrama
(21.1)
                        Serial.println("Atras");
                        digitalWrite(ledC,HIGH);


                        velocidad.write(Velocidad_Parada);
                        delay(100);
                        direccion.write(Giro_Med);
                        delay(100);
                        velocidad.write(Velocidad_Atras);

                        movimiento = true;
                        estado = 0;
                  }else{                        //Diagrama (21.2)
                        if(adelanteatras == true){      //Si la silla
esta avanzando entonces entra en este if para girar un poco a la
derecha->(RT)
                              if(giro < 120){
                                    Serial.println("sumo 10");
                                    giro += 10;
                                          direccion.write(giro);
                                          delay(50);

                              }
                              Serial.print("Giro un poco a la
derecha\tGiro: ");
                              Serial.println(giro);

                        }
                  }
            break;
      }
}
```

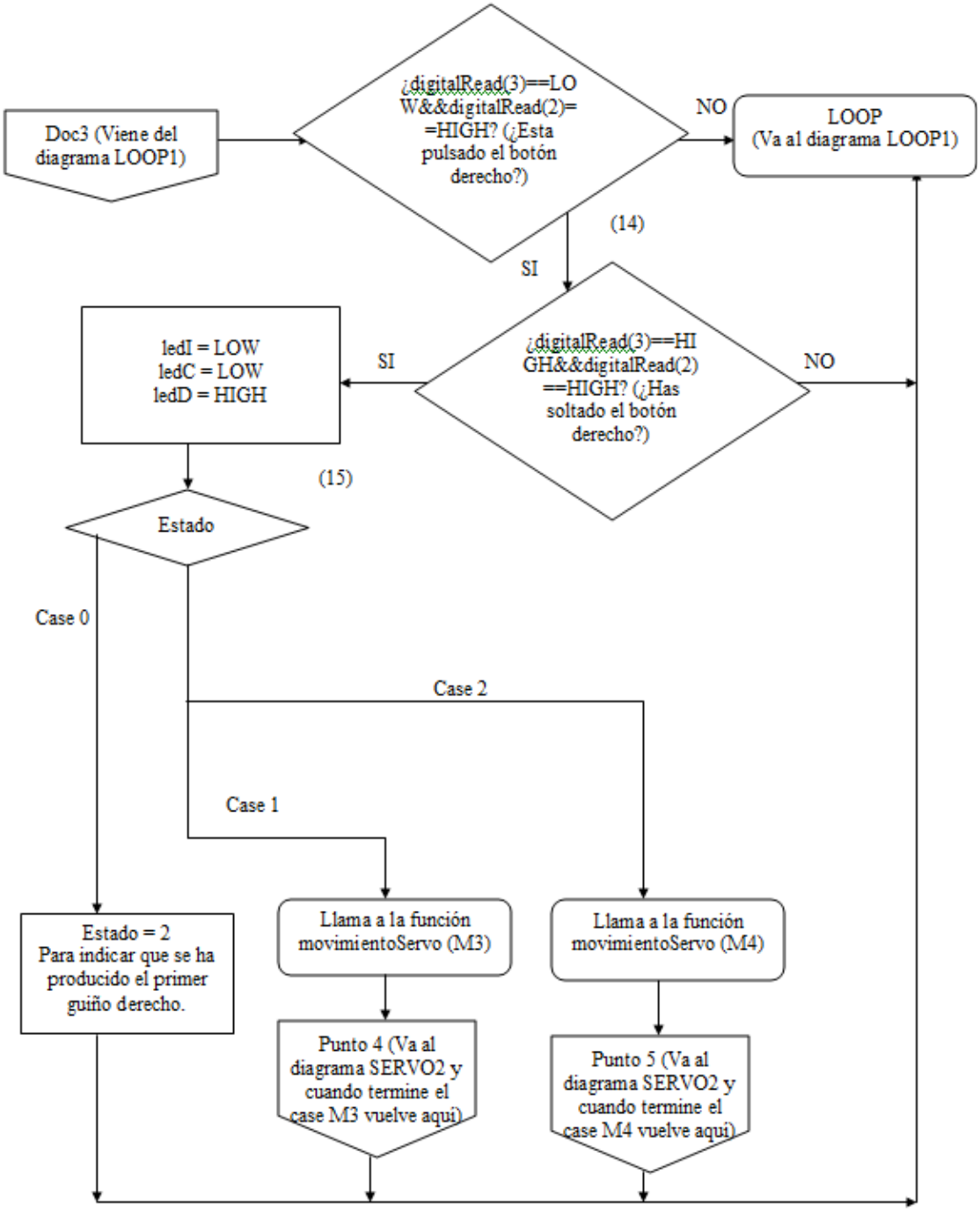
APÉNDICE DIAGRAMA DE FLUJO FINAL

Función void setup ()

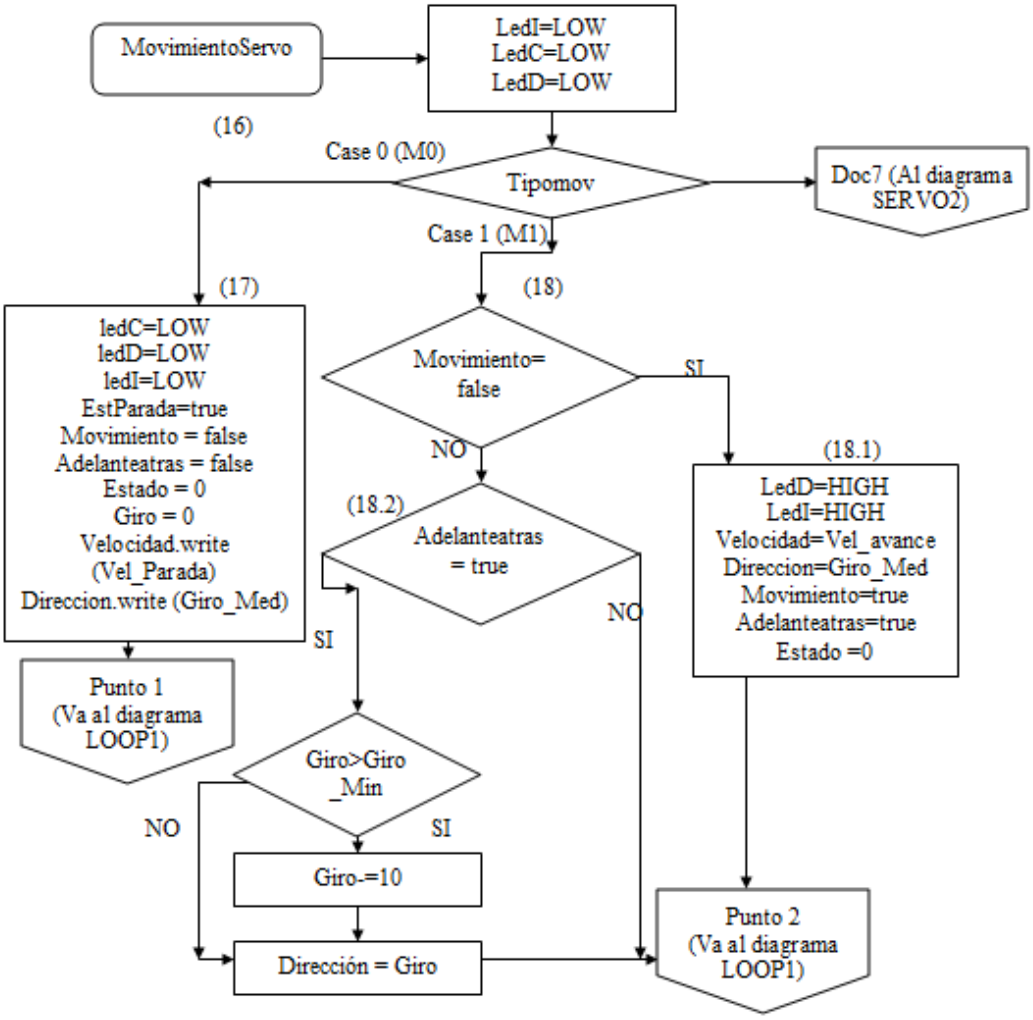


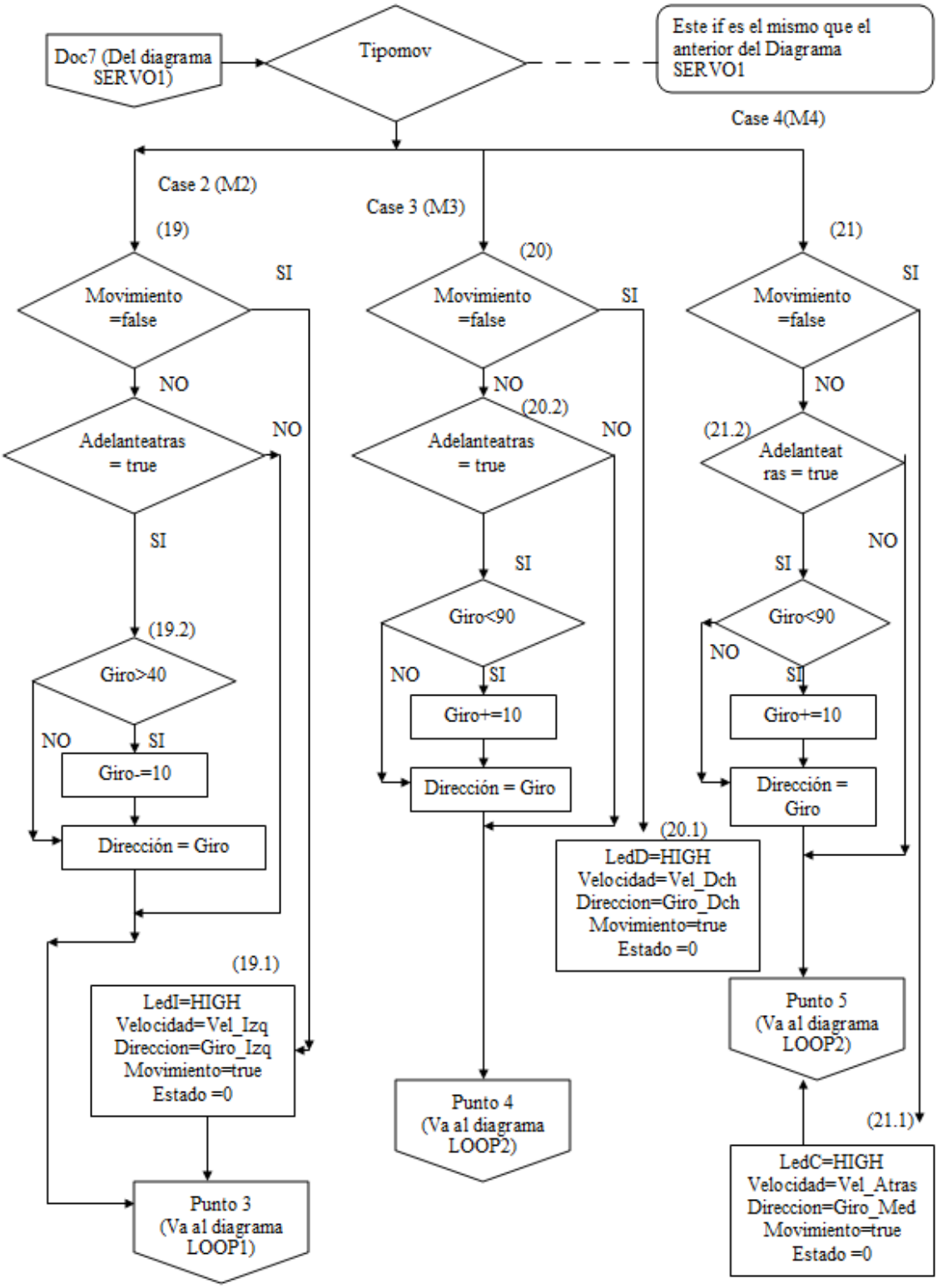
Función void loop ()


LOOP
¿EstParada=True?
NO
SI
(10)
¿DigitalRead(2)==LOW &&digitalRead(3)==LO W? (¿Se ha pulsado el botón parada?)
EstParada = false
Estado = 0
(11)
NO
SI
Llama a la función movimientoServo (M0)
Punto 1 (Va al diagrama SERVO1 y cuando termine el case M0 vuelve aqui)
¿digitalRead(2)==LO W&&digitalRead(3)= =HIGH? (¿Esta pulsado el botón izquierdo?)
NO
(12)
SI
(13)
Se activan los correspondientes led para indicar en que estado te encuentras
SI
estado
¿digitalRead(3)==HI GH&&digitalRead(2) ==HIGH? (¿Has soltado el botón izquierdo?)
Case 2
Case 1
Case 0
NO
Estado = 1
Para indicar que se ha producido un guiño
Llama a la función movimientoServo (M1)
Llama a la función movimientoServo (M2)
Punto 2 (Va al diagrama SERVO1 y cuando termine el case M1 vuelve aqui)
Punto 3 (Va al diagrama SERVO2 y cuando termine el case M2 vuelve aqui)
Doc3 (LOOP2)


Doc3 (Viene del diagrama LOOP1)
¿digitalRead(3)==LOW&&digitalRead(2)==HIGH? (¿Esta pulsado el botón derecho?)
NO
LOOP (Va al diagrama LOOP1)
(14)
SI
¿digitalRead(3)==HIGH&&digitalRead(2)==HIGH? (¿Has soltado el botón derecho?)
NO
SI
ledI = LOW
ledC = LOW
ledD = HIGH
(15)
Estado
Case 0
Case 2
Case 1
Estado = 2
Para indicar que se ha producido el primer guiño derecho.
Llama a la función movimientoServo (M3)
Llama a la función movimientoServo (M4)
Punto 4 (Va al diagrama SERVO2 y cuando termine el case M3 vuelve aquí)
Punto 5 (Va al diagrama SERVO2 y cuando termine el case M4 vuelve aquí)

Función MovimientoServo




Doc7 (Del diagrama SERVO1)
Tipomov
Este if es el mismo que el anterior del Diagrama SERVO1
Case 4(M4)
Case 2 (M2)
Case 3 (M3)
(19)
(20)
(21)
Movimiento =false
SI
NO
Adelanteatras = true
(20.2)
(21.2)
Adelanteat ras = true
(19.2)
Giro>40
Giro<90
Giro-=10
Giro+=10
Dirección = Giro
(19.1)
LedI=HIGH
Velocidad=Vel_Izq
Direccion=Giro_Izq
Movimiento=true
Estado =0
(20.1)
LedD=HIGH
Velocidad=Vel_Dch
Direccion=Giro_Dch
Movimiento=true
Estado =0
(21.1)
LedC=HIGH
Velocidad=Vel_Atras
Direccion=Giro_Med
Movimiento=true
Estado =0
Punto 3 (Va al diagrama LOOP1)
Punto 4 (Va al diagrama LOOP2)
Punto 5 (Va al diagrama LOOP2)

Datasheet Codificador N74LS148

MOTOROLA

10-LINE-TO-4-LINE AND 8-LINE-TO-3-LINE PRIORITY ENCODERS

The SN54/74LS147 and the SN54/74LS148 are Priority Encoders. They provide priority decoding of the inputs to ensure that only the highest order data line is encoded. Both devices have data inputs and outputs which are active at the low logic level.

The LS147 encodes nine data lines to four-line (8-4-2-1) BCD. The implied decimal zero condition does not require an input condition because zero is encoded when all nine data lines are at a high logic level.

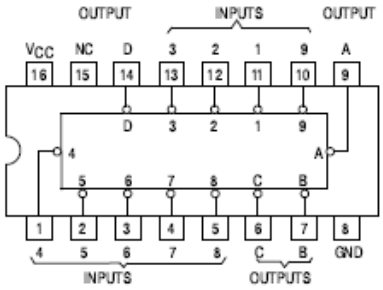
The LS148 encodes eight data lines to three-line (4-2-1) binary (octal). By providing cascading circuitry (Enable Input EI and Enable Output EO) octal expansion is allowed without needing external circuitry.

The SN54/74LS748 is a proprietary Motorola part incorporating a built-in deglitcher network which minimizes glitches on the GS output. The glitch occurs on the negative going transition of the EI input when data inputs 0–7 are at logical ones.

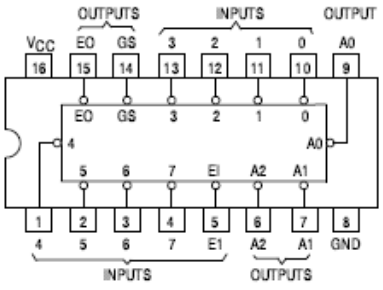
The only dc parameter differences between the LS148 and the LS748 are that (1) Pin 10 (input 0) has a fan-in of 2 on the LS748 versus a fan-in of 1 on the LS148; (2) Pins 1, 2, 3, 4, 11, 12 and 13 (inputs 1, 2, 3, 4, 5, 6, 7) have a fan-in of 3 on the LS748 versus a fan-in of 2 on the LS148.

The only ac difference is that t_{PHL} from EI to EO is changed from 40 to 45 ns.

SN54/74LS147
(TOP VIEW)



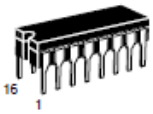
SN54/74LS148
SN54/74LS748
(TOP VIEW)



SN54/74LS147
SN54/74LS148
SN54/74LS748

10-LINE-TO-4-LINE
AND 8-LINE-TO-3-LINE
PRIORITY ENCODERS

LOW POWER SCHOTTKY



J SUFFIX
CERAMIC
CASE 620-09

N SUFFIX
PLASTIC
CASE 648-08

D SUFFIX
SOIC
CASE 751B-03

ORDERING INFORMATION

SN54LSXXXJ	Ceramic
SN74LSXXXN	Plastic
SN74LSXXXD	SOIC

SN54/74LS147
FUNCTION TABLE

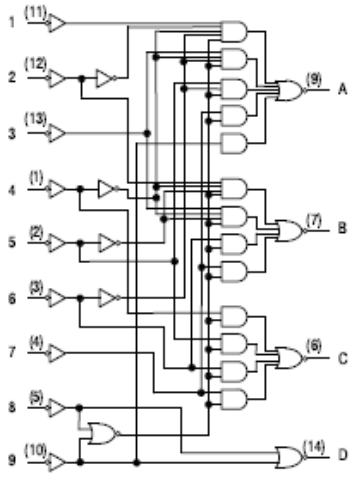
INPUTS									OUTPUTS			
1	2	3	4	5	6	7	8	9	D	C	B	A
H	H	H	H	H	H	H	H	H	H	H	H	H
X	X	X	X	X	X	X	X	L	L	H	H	L
X	X	X	X	X	X	X	L	H	L	H	H	H
X	X	X	X	X	X	L	H	H	H	L	L	L
X	X	X	X	X	L	H	H	H	H	L	L	H
X	X	X	X	L	H	H	H	H	H	L	H	L
X	X	X	L	H	H	H	H	H	H	L	H	H
X	X	L	H	H	H	H	H	H	H	H	L	L
X	L	H	H	H	H	H	H	H	H	H	L	H
L	H	H	H	H	H	H	H	H	H	H	H	L

H = HIGH Logic Level, L = LOW Logic Level, X = Irrelevant

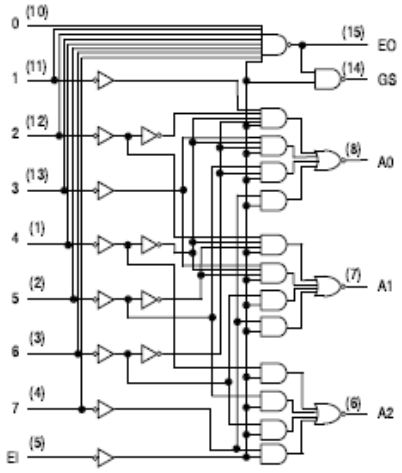
SN54/74LS148
SN54/74LS748
FUNCTION TABLE

INPUTS									OUTPUTS				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

FUNCTIONAL BLOCK DIAGRAMS

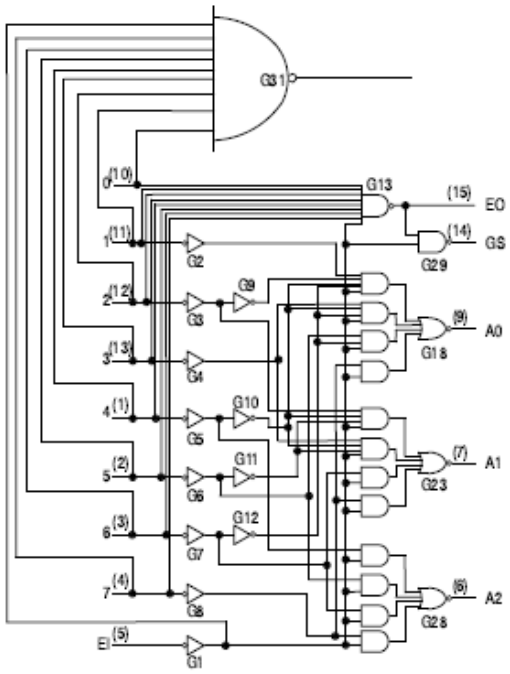


SN54/74LS147



SN54/74LS148

FUNCTIONAL BLOCK DIAGRAMS (continued)



SN54/74LS748

GUARANTEED OPERATING RANGES

Symbol	Parameter		Min	Typ	Max	Unit
V_{CC}	Supply Voltage	54 74	4.5 4.75	5.0 5.0	5.5 5.25	V
T_A	Operating Ambient Temperature Range	54 74	−55 0	25 25	125 70	°C
I_{OH}	Output Current — High	54, 74			−0.4	mA
I_{OL}	Output Current — Low	54 74			4.0 8.0	mA

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

Symbol	Parameter		Limits Min	Limits Typ	Limits Max	Unit	Test Conditions	
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs	
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs	
		74			0.8			
V_{IK}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA	
V_{OH}	Output HIGH Voltage	54	2.5	3.5		V	V_{CC} = MIN, I_{OH} = MAX, V_{IN} = V_{IH} or V_{IL} per Truth Table	
		74	2.7	3.5		V		
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 4.0 mA	V_{CC} = V_{CC} MIN, V_{IN} = V_{IL} or V_{IH} per Truth Table
		74		0.35	0.5	V	I_{OL} = 8.0 mA	
I_{IH}	Input HIGH Current All Others Input 0 (LS748) Inputs 1–7 (LS148) Inputs 1–7 (LS748)				20 40 40 60	µA	V_{CC} = MAX, V_{IN} = 2.7 V	
	All Others Input 0 (LS748) Inputs 1–7 (LS148) Inputs 1–7 (LS748)				0.1 0.2 0.2 0.3	mA	V_{CC} = MAX, V_{IN} = 7.0 V	
I_{IL}	Input LOW Current All Others Input 0 (LS748) Inputs 1–7 (LS148) Inputs 1–7 (LS748)				−0.4 −0.8 −0.8 −1.2	mA	V_{CC} = MAX, V_{IN} = 0.4 V	
I_{OS}	Short Circuit Current (Note 1)		−20		−100	mA	V_{CC} = MAX	
I_{CCH}	Power Supply Current Output HIGH				17	mA	V_{CC} = MAX, All Inputs = 4.5 V	
I_{CCL}	Output LOW				20	mA	V_{CC} = MAX, Inputs 7 & E1 = GND All Other Inputs = 4.5 V	

Note 1: Not more than one output should be shorted at a time, nor for more than 1 second.

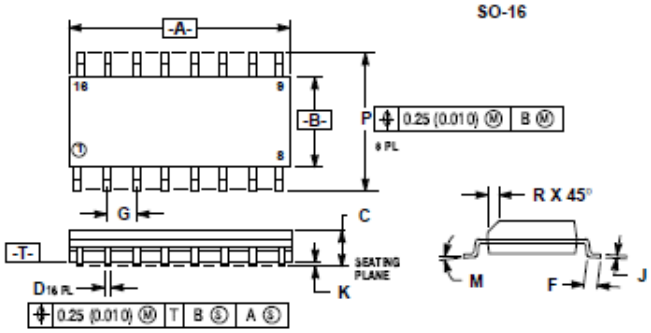
AC CHARACTERISTICS (V_{CC} = 5.0 V, T_A = 25°C)
SN54/74LS147

Symbol	From (Input)	To (Output)	Waveform	Limits			Unit	Test Conditions
				Min	Typ	Max		
t_{PLH}	Any	Any	In-phase output		12	18	ns	C_L = 15 pF, R_L = 2.0 kΩ
t_{PHL}					12	18		
t_{PLH}	Any	Any	Out-of-phase output		21	33	ns	
t_{PHL}					15	23		

SN54/74LS148
SN54/74LS748

Symbol	From (Input)	To (Output)	Waveform	Limits			Unit	Test Conditions
				Min	Typ	Max		
t_{PLH}	1 thru 7	A0, A1, or A2	In-phase output		14	18	ns	C_L = 15 pF, R_L = 2.0 kΩ
t_{PHL}					15	25		
t_{PLH}	1 thru 7	A0, A1, or A2	Out-of-phase output		20	36	ns	
t_{PHL}					16	29		
t_{PLH}	0 thru 7	EO	Out-of-phase output		7.0	18	ns	
t_{PHL}					25	40		
t_{PLH}	0 thru 7	GS	In-phase output		35	55	ns	
t_{PHL}					9.0	21		
t_{PLH}	EI	A0, A1, or A2	In-phase output		16	25	ns	
t_{PHL}					12	25		
t_{PLH}	EI	GS	In-phase output		12	17	ns	
t_{PHL}					14	36		
t_{PLH}	EI	EO	In-phase output		12	21	ns	(LS148) (LS748)
t_{PHL}					28 30	40 45		

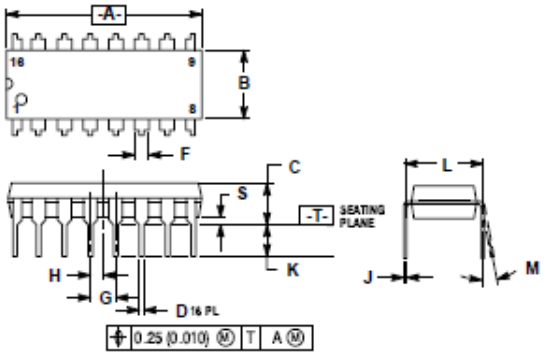
Case 751B-03 D Suffix
16-Pin Plastic
SO-16



NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DIMENSION A AND B DO NOT INCLUDE MOLD PROTRUSION.
4. MAXIMUM MOLD PROTRUSION 0.15 (0.006) PER SIDE.
5. 751B-01 IS OBSOLETE, NEW STANDARD 751B-03.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	9.80	10.00	0.386	0.393
B	3.80	4.00	0.150	0.157
C	1.35	1.75	0.054	0.068
D	0.35	0.49	0.014	0.019
F	0.40	1.25	0.016	0.049
G	1.27 BSC		0.050 BSC	
J	0.19	0.25	0.008	0.009
K	0.10	0.25	0.004	0.009
M	0°	7°	0°	7°
P	5.80	6.20	0.229	0.244
R	0.25	0.50	0.010	0.019

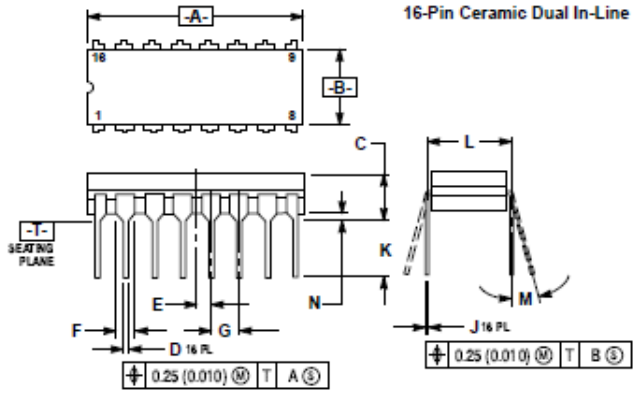
Case 648-08 N Suffix
16-Pin Plastic



NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION "L" TO CENTER OF LEADS WHEN FORMED PARALLEL.
4. DIMENSION "B" DOES NOT INCLUDE MOLD FLASH.
5. ROUNDED CORNERS OPTIONAL.
6. 648-01 THRU -07 OBSOLETE, NEW STANDARD 648-08.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	18.80	19.55	0.740	0.770
B	6.35	6.85	0.250	0.270
C	3.69	4.44	0.145	0.175
D	0.39	0.53	0.015	0.021
F	1.02	1.77	0.040	0.070
G	2.54 BSC		0.100 BSC	
H	1.27 BSC		0.050 BSC	
J	0.21	0.38	0.008	0.015
K	2.80	3.30	0.110	0.130
L	7.50	7.74	0.295	0.305
M	0°	10°	0°	10°
S	0.51	1.01	0.020	0.040

Case 620-09 J Suffix
16-Pin Ceramic Dual In-Line



NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION L TO CENTER OF LEAD WHEN FORMED PARALLEL.
4. DIM F MAY NARROW TO 0.76 (0.030) WHERE THE LEAD ENTERS THE CERAMIC BODY.
5. 620-01 THRU -08 OBSOLETE, NEW STANDARD 620-09.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	19.05	19.55	0.750	0.770
B	6.10	7.36	0.240	0.290
C	—	4.19	—	0.165
D	0.39	0.53	0.015	0.021
E	1.27 BSC		0.050 BSC	
F	1.40	1.77	0.055	0.070
G	2.54 BSC		0.100 BSC	
J	0.23	0.27	0.009	0.011
K	—	5.08	—	0.200
L	7.62 BSC		0.300 BSC	
M	0°	15°	0°	15°
N	0.39	0.88	0.015	0.035