UNIVERSIDAD NACIONAL AUTONOMA

DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES
CUAUTITLAN

DIGITALIZADOR DE VOZ

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A N:

ERNESTO AGUILAR RODRÍGUEZ
GILBERTO LÓPEZ ZAMUDIO
NOEMÍ HERNÁNDEZ DOMÍNGUEZ

ASESOR: ING. JOSE UBALDO RAMÍREZ URIZAR

CUAUTITLAN IZCALLI. EDO. DE MEX. **2007**

GRACIAS

A Dios:

Por haberme permitido llegar a este punto, en donde culmino una de mis metas en la vida, por darme la oportunidad de aprender un poco de cada persona que puso en mi camino y por estar conmigo en todo momento.

A mis Padres:

Por darme la vida y enseñarme a vivir
Mamá gracias por ser el oído, el abrazo, el consejo en el momento adecuado.
Papá† gracias por ayudarme a ver que si quiero conseguir algo es posible si me esfuerzo.
A ambos por ver por mí desde que nací y enseñarme a amarlos y a amar la vida.

A mis Hermanas:

Por las discusiones, gritos y peleas, pero más que nada por demostrarme que siempre van a estar conmigo por más locas que sean mis ideas, que siempre puedo apoyarme en ellas por más erróneas que sean mis decisiones.

A mi chaparra Hermosa:

Por que definitivamente ha sido el esfuerzo que me faltaba, por que llego en el momento adecuado a mi vida , porque para mi fue parte fundamental en la culminación de este trabajo, por que me ayuda a crecer día a día como persona y por escucharme Gracias.

A mis Amigos:

Por que convivieron y crecieron conmigo aceptándonos con nuestras diferencias y similitudes, por que discutimos y reímos juntos, por que nos criticamos y ayudamos pero más por que con ellos aprendí a valorar la amistad.

A mis Maestros y Amigos:

Por que a cada uno de ellos le debo gran parte de mi formación tanto profesional como personal, por que resolvieron muchas de mis dudas y por que me orientaron.

A Mi Facultad:

Por que me albergo durante varios años y fungió como mi segunda casa, y no se para los demás pero yo siempre diré que esta Facultad de Cuautitlán tiene ese algo, un “no se que” que la hace muy especial.

A:

Ines, Gilberto, Tere, Linda, Maribel, Noemí, Ernesto, Mario, Alejandro, Juan Carlos, Verónica, Sonia, Humberto, Lalo, Ana, Javier, Marcelo, Blanca, Ubaldo, Juan, José Luís …

GRACIAS……

GIL.

A DIOS

Por darme la oportunidad de vivir y darme inteligencia y sabiduría.

A MIS PADRES

Gregorio y Margarita que son las personas que más quiero, y que desde pequeña me han brindado todo el apoyo moral y económico para realizar mis estudios.

A MI ABUELO

Cele por ser la persona más sabia que he conocido y porque siempre confió en mi.

A MIS HERMANOS

Vero, Jose, Ceci y Feli por ser una parte muy importante en mi vida y porque de una u otra manera siempre me han echado la mano.

A NETO

Por todo el cariño que me ha dado, porque en cada momento que se hacia difícil el trabajo siempre estuvo apoyándome para sacarlo adelante y no darnos por vencidos y por haberme brindado su apoyo y comprensión cuando más lo necesitaba.

A MIS PROFESORES Y AMIGOS

A todos mis profesores y amigos que conocí en la FESC por haberme compartido sus conocimientos día con día ya que aprendí experiencias buenas y malas que serán inolvidables.

NOEMÍ

A DIOS Y A MIS PADRES

Ernesto y Emma por su ejemplo, apoyo y sacrificio a lo largo de toda mi vida.

A MI HERMANA Y MI FAMILIA

Anabel, mis abuelitos, tíos y primos por estar conmigo en las buenas y en las malas.

A MI NOVIA

Noemí por su amor y comprensión Durante tanto tiempo.

A MI AMIGO

Gilberto por su apoyo.

A MIS AMIGOS Y PROFESORES

Miguel, Rene, Raúl, Mario, Marcos, Carlos, Héctor, Alejandra, Daniel, Alejandro, por su amistad y a los Ingenieros Ubaldo, Marcelo, Blanca, Lourdes, Jorge Por su ayuda en la elaboración de este trabajo.

ERNESTO.

OBJETIVOS

GENERALES

Construir un sistema de control por medio de la voz.

PARTICULARES

Que el sistema sea capaz de realizar tareas especificas por el usuario.

Que tenga la habilidad de diferenciar entre una señal de voz y un silencio.

Mover a través de la voz un motor de corriente directa.

INTRODUCCIÓN

Desde la antigüedad la voz ha sido el medio de comunicación más importante entre los seres humanos, posteriormente con el avance tecnológico no se ha detenido en un medio de comunicación sino que ha permitido que éste haya sido medio de investigación para crear áreas de estudio y análisis, así surge el área de reconocimiento de voz, el principal objetivo que persigue es proporcionar una forma apropiada de interacción hombre – máquina mediante el uso de voz.

El primer sistema de reconocimiento de voz fue desarrollado en 1952 sobre una computadora analógica usando voz discretizada para reconocer los dígitos del 0 al 9 con un algoritmo de plantilla de concordancia dependiente de la persona que habla. Una exactitud de 98% en el reconocimiento fue reportada. Mas tarde en esa misma década, un sistema con atributos similares fue desarrollado que reconoció consonantes y vocales. En los años sesenta la investigación en reconocimiento de voz se movió a las computadoras digitales. Esta plataforma proporcionó las bases para la tecnología de reconocimiento de voz como se conoce hoy en día. Todo esto ha sido posible gracias a la evolución de la tecnología electrónica la cual ha permitido desarrollar sistemas complejos que puedan simular y generar, casi a la perfección señales acústicas de la voz humana.

En los sistemas de reconocimiento de voz no se intenta, reconocer lo que el usuario dice, sino identificar una serie de sonidos y sus características para decidir si el usuario es quien dice ser. Para identificar a un usuario utilizando un reconocedor de voz se debe disponer de ciertas condiciones para el correcto registro de los datos, como ausencia de ruidos o ecos y tener la constancia en el tono de voz; idealmente, porque desafortunadamente estas condiciones en ocasiones pueden estar cambiando de forma indeseable.

Los sistemas independientes de la persona que habla pueden reconocer voz de cualquier persona. Los sistemas dependientes de la persona que habla deben ser entrenados para cada usuario individual, pero típicamente tienen más altas tasas de exactitud. Los sistemas adaptables a la persona que habla, inicia con frases independientes de la persona que habla. Los sistemas de voz continuos pueden reconocer palabras habladas en un ritmo natural mientras que los sistemas de palabras aisladas requieren de una pausa deliberada entre cada palabra. No obstante más deseable, la voz continua es más difícil de procesar por la dificultad en detectar los límites de cada palabra.

Un sistema de control de acceso por medio de la voz es una aplicación de los sistemas de reconocimiento de voz que utiliza forzosamente un digitalizador de voz, en donde se comparan los datos en formato digital con la base de datos que existen en la PC, previamente grabados; aunque actualmente en la mayoría de sistemas basta con una sola frase, es habitual que el usuario se vea obligado a repetirla porque el sistema le niega el acceso (una simple congestión hace variar el tono de voz, aunque sea levemente, y el sistema no es capaz de decidir si el acceso ha de ser autorizado o no; incluso el estado anímico de una persona varía su timbre...). A su favor, el reconocimiento de voz posee la cualidad de una reconocer entre los usuarios, siempre y cuando su funcionamiento sea correcto y éstos no se vean obligados a repetir lo mismo varias veces, o se les niegue un acceso porque no se les reconoce correctamente.

Un sistema de reconocimiento de voz realiza mediante un digitalizador de voz interfaces dirigidas para efectuar acciones como aplicación de control por computadora que ha hecho posible el movimiento “inteligente”en robots industriales, la optimización de economía de combustible en automóviles, sistemas de ayuda a discapacitados, operaciones y transacciones comerciales y control de acceso entre otras.

.

Es por esto que el presente diseño pretende hacer uso de estas características del sistema de reconocimiento para el control de un motor de corriente directa.

En este trabajo se describen las diferentes etapas de funcionamiento del sistema y este será implementado en los laboratorios de electrónica. Además se pretende que el proyecto quede como antecedente o base para una mejora a futuro, para fines académicos y/o industriales.

El siguiente diagrama de bloques (Figura I), describe la estructura general:

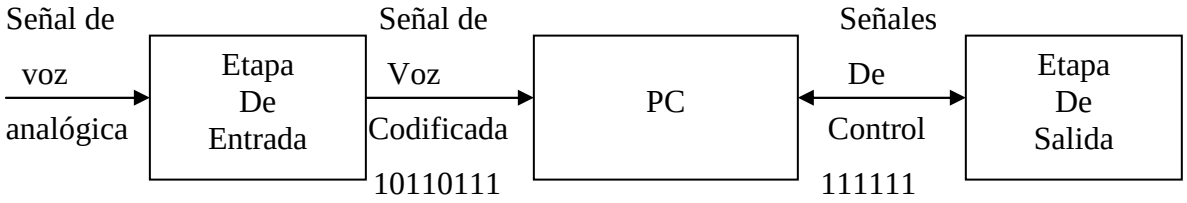


Fig. I. Diagrama a bloques del sistema de control por medio de la voz. .

CAPÍTULO I

1.1 SEÑALES.

Una señal x(t) es una función real o escalar de la variable de tiempo t. Por real se quiere decir que para cada valor fijo de la variable de tiempo t, el valor de la señal en el tiempo es un número real.

Existen diferentes tipos de señales que se clasifican en analógicas y digitales, periódicas o no periódicas, continuas o discontinuas, etc.

SEÑAL ANALÓGICA: Es aquella que esta definida en un intervalo continuo de tiempo cuya amplitud puede adoptar un intervalo continuo de valores. La figura 1.1 muestra una señal analógica en tiempo continuo. La señal en tiempo continuo es un caso especial de una señal analógica .

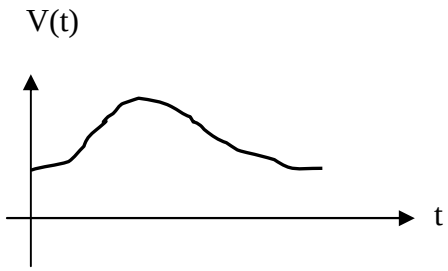


Fig.1.1 Señal analógica

SEÑAL DISCRETA o EN TIEMPO DISCRETO. Esta se encuentra definida sólo en valores discretos de tiempo (esto es, aquellos en los que la variable independiente t está cuantificada). En una señal en tiempo discreto, la amplitud puede adoptar valores en un

intervalo continuo, entonces la señal se denomina de datos muestreados, esta se puede generar muestreando una señal analógica en valores discretos de tiempo.

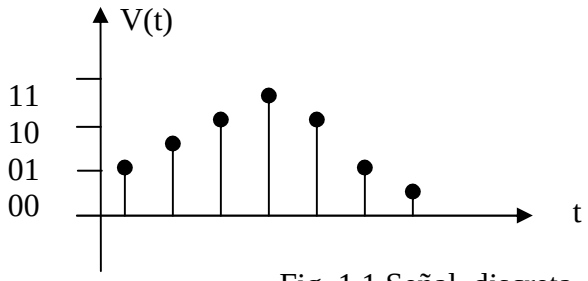


Fig. 1.1 Señal discreta

SEÑAL DIGITAL. Es una señal en tiempo discreto con amplitud cuantificada, que se puede representar mediante una secuencia de números, por ejemplo, en la forma de números binarios. En la práctica, muchas señales digitales se obtienen mediante el muestreo de señales analógicas para que así sean leídas como palabras binarias finitas.
La figura 1.3 muestra una señal digital cuantificada en amplitud.

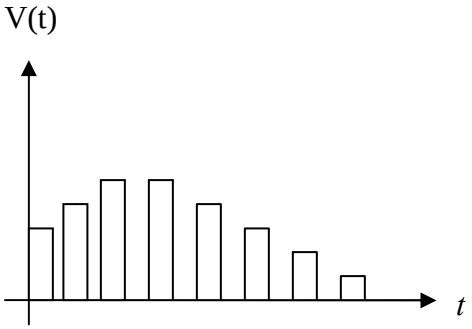


Fig. 1.2 Señal digital

1.2 SEÑAL DE VOZ

Se define formalmente como la función de una o más variables, que transportan información acerca de la naturaleza de un fenómeno físico. Cuando la función depende de una sola variable, se dice que es unidimensional. Una señal de voz es un ejemplo de una señal unidimensional cuya amplitud varía en el tiempo, dependiendo de la palabra hablada y de quien la pronuncie.

La voz es un flujo continuo de ondas sonoras en combinación con los silencios (Figura 1.4).

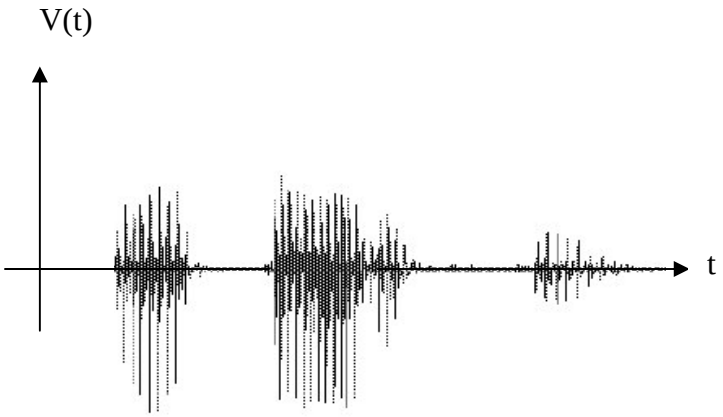


Fig. 1.4 Señales y silencios en la señal de voz.

La voz humana es analógica, aunque presenta una serie de curvas complejas, en forma simplificada podría representarse por una señal senoidal cambiante en amplitud y frecuencia (Figura 1.5).

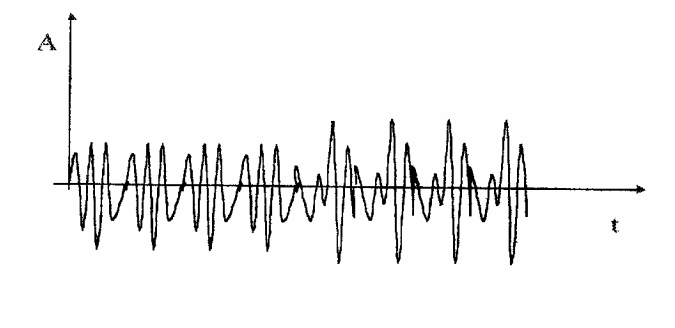


Fig. 1.5 Señal de voz

La voz es continua en el tiempo pero variante en amplitud y frecuencia. Para que pueda ser procesada por hardware y software es necesario convertirla a formato digital el cual se explica posteriormente en el proceso de digitalización.

Cuatro características importantes del análisis de la señal de voz son:

- Frecuencia
- Amplitud
- Estructura Armónica
- Resonancia.

La **frecuencia** representa al número de vibraciones (ciclos) por segundo. Los patrones de sonidos más simples son los sonidos puros, y se pueden representar gráficamente por una onda senoidal.

La mayoría de los sonidos incluyendo la voz tienen una frecuencia dominante llamada **frecuencia fundamental.** Esta al oído se percibe como el tono dicha frecuencia es la velocidad a la que vibran las cuerdas vocales al producir un sonido.

La frecuencia fundamental es característica de cada persona y no se puede cambiar. Este parámetro puede encontrarse atenuado, ser irregular, ausente, entrecortado, de acuerdo a las distintas características del sistema fonético.

La frecuencia fundamental es medida en Hz, y varia según se trate de un niño, mujer u hombre y de acuerdo a su sistema emisor de voz, como el que se muestra en la figura 1.6. La frecuencia fundamental nos permite obtener más información de la voz de la persona a la cual se desea reconocer, para esto es necesario tomar en consideración que el ancho de banda (BW) de la voz esta entre los 300 Hz y los 3.3 KHz

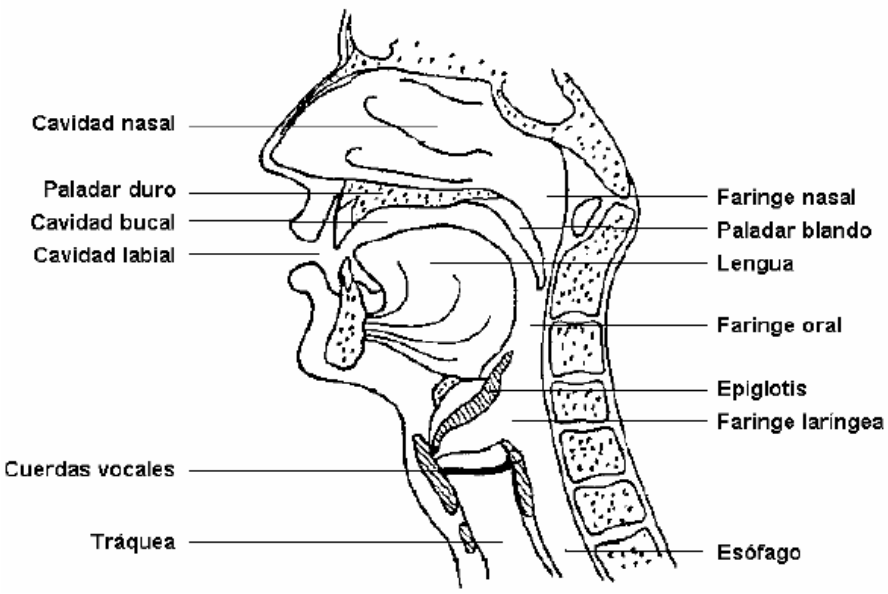


Fig. 1.6 Sistema emisor de voz

Los tonos altos equivalen a tener mayor frecuencia y viceversa tonos bajos dan como resultado una menor frecuencia. El volumen de un sonido refleja la cantidad de aire que es forzada a salir cuando se pronuncia una palabra. Se describe y representa como **amplitud** de la onda y se mide en decibeles dB.

Dentro de la frecuencia fundamental existen otras frecuencias que contribuyen al timbre del sonido. (Son las que nos permiten distinguir las voces de diferentes personas). Algunas de juegan un rol muy importante en la distinción entre un fonema y otro, las cuales son producidas por la **resonancia,** ésta se define como la habilidad que tiene una fuente de sonido de causar que otro objeto vibre de lo anterior cabe destacar que la señal de voz no es un sonido puro, es continuación de múltiples frecuencias y se representa como una onda compleja.

1.3 SISTEMA DE RECONOCIMIENTO DE VOZ.

Un sistema de reconocimiento automático de voz es un sistema de cierta complejidad capaz de decodificar los sonidos de la voz y generar un conjunto de patrones binarios (asociados a partes de la voz) que puedan ser comparados con la señal acústica de entrada analógica (a reconocer) devolviendo la secuencia de estos patrones que con mayor probabilidad "representan" a la misma.

La tecnología del habla voz se estructura en cuatro tecnologías básicas principales:

- **El reconocimiento de voz:** Es el proceso de conversión de un mensaje hablado, que permita al usuario una comunicación con la máquina.
- **La conversión texto – voz:** Se ocupa de la generación de mensajes hablados mediante la simulación del proceso de lectura de un texto escrito almacenado en formato electrónico.
- **El reconocimiento de locutores:** Es el proceso de identificación o verificación de la identidad del hablante de forma automática a partir de la señal de voz.
- **La codificación de voz:** Su objetivo es la búsqueda de representaciones eficientes en formato digital de la señal de voz para su almacenamiento y/o transmisión, persiguiendo obtener la mayor calidad posible, para el menor número de bits por muestra.

Es importante para este proyecto **Codificar la voz,** ya que precisamente uno de los primeros pasos a realizar es la captura y codificación de una señal de voz para después almacenar estos datos en la computadora mediante el uso de un software.

La tecnología del habla se sitúa como receptora de un amplio conjunto de conocimientos y procedimientos de actuación sobre la información representada en la señal de voz. Conocimientos que se articulan con un alto grado de dificultad y especialización, ya que pertenecen a un marco científico-técnico multidisciplinarlo, donde se dan cita diferentes ramas del conocimiento como son: fisiología, acústica, lingüística, procesado de señal,

inteligencia artificial, teoría de la comunicación y de la información, y ciencia de la computación.

1.4 PROBLEMAS EN EL RECONOCIMIENTO DE VOZ.

Cuando un sistema de reconocimiento de voz funciona en situaciones reales se encuentra con condiciones adversas, como ruidos, distorsiones y otros factores que degradan la calidad de la voz; la capacidad del sistema para hacer frente a estos cambios en las condiciones del entorno se denomina robustez.

La cuestión principal ha sido determinar cuales son las posibles causas que hacen tan difícil llevar a cabo un reconocimiento de voz en condiciones generales, de forma que se puedan buscar soluciones parciales a cada uno de los problemas y conseguir una solución global lo mas óptima posible. Algunas dificultades a las que se enfrenta el sistema son:

- **Las variaciones de fonación, debidas a los hablantes:** Ninguna persona habla igual, es decir los sonidos producidos por diferentes personas no suenan igual.
- **Variaciones fonéticas:** Las frecuencias de los formantes y las duraciones de las transiciones pueden cambiar a lo largo del tiempo, lo que produce un cierto alejamiento de los patrones o reglas utilizadas durante el reconocimiento.
- **Variaciones temporales:** La duración de una palabra e incluso de los sonidos puede cambiar, generando la necesidad de realizar alineamientos dinámicos, que permitan tener en cuenta a estas posibles variaciones.
- **Ruidos e interferencias:** Una persona puede reconocer el habla en condiciones adversas, es decir, con baja relación señal/ruido e incluso en presencia de otros sonidos interferentes, gracias a las propias características del oído.

Para reconocer el habla de forma automática se requiere una representación paramétrica de la voz que retenga sus características relevantes.

Todo sistema de reconocimiento de voz generalmente es utilizado como una interfaz entre humano y computadora para realizar una tarea especifica. (Figura 1.7)

Fig. 1.7 Proceso de reconocimiento

Este debe de cumplir:

- **Pre-procesamiento:** Convierte la señal de voz de tal manera que pueda ser procesada en forma digital.

- **Reconocimiento:** En esta etapa básicamente se reconoce al patrón de voz. Este sistema es dependiente del locutor, por lo que si un locutor no corresponde al patrón establecido no realizará las funciones que tiene predeterminadas.

- **Comunicación:** Envía lo reconocido al sistema (Software/Hardware) que lo requiere.
- **Ejecución:** Mediante órdenes del software el sistema realiza tareas específicas.

1.5 DIGITALIZACIÓN.

Las características más importantes del contenido de la voz se presentan al extraer y representar los contenidos espectrales (frecuencias), de la señal de voz por segmentos fonéticos. Por lo tanto es necesario trabajar con los conceptos de filtrado, muestreo y cuantización. Todos estos conceptos y técnicas de procesamiento son clásicos en el análisis de señales, por lo que la voz por su característica de señal física analógica se ajusta a este principio. Estos conceptos se definen a continuación:

1.5.1 FILTRADO

Para este proceso es necesario saber que es un filtro y como funciona. En la vida cotidiana es difícil tener una señal eléctrica pura o aislada, por lo regular las señales se entremezclan o distorsionan, por ello es necesario contar con algún tipo de dispositivo que nos permita separar las señales unas de otras o simplemente eliminar el ruido de ellas. Los filtros son aquellos dispositivos de los cuales nos servimos para estos fines. Estos sistemas se entienden como redes de dos puertos, uno de entrada y uno de salida, que funcionan en el dominio de la frecuencia. Estos circuitos permiten el paso de una determinada banda de frecuencias mientras atenúa todas las señales que no estén comprendidas dentro de esta banda. Existen filtros activos y pasivos. Los filtros pasivos sólo tienen resistencias, inductores y capacitores. En los filtros activos se utilizan amplificadores operacionales además de resistencias y capacitores.

Así mismo, dependiendo de la aplicación y funcionamiento deseado, se deberá seleccionar el tipo de filtro adecuado. Existen cuatro tipos de filtros, estos son: filtro pasa bajas, este tipo de filtro trabaja dejando pasar las frecuencias bajas y rechazando las frecuencias altas, (figura 1.8), filtro pasa altas (Figura 1.9), este filtro trabaja dejando pasar las frecuencias altas y rechazando las frecuencias bajas, filtro pasa banda (Figura 1.10), este filtro trabaja dejando pasar una banda o intervalo de frecuencias definido y rechazando las frecuencias mayores y menores a los limites de dicha banda y por último el filtro supresor de banda

(Figura 1.11), este filtro se encarga de atenuar un rango determinado de frecuencias, permitiendo el paso de frecuencias más altas o más bajas del rango determinado.

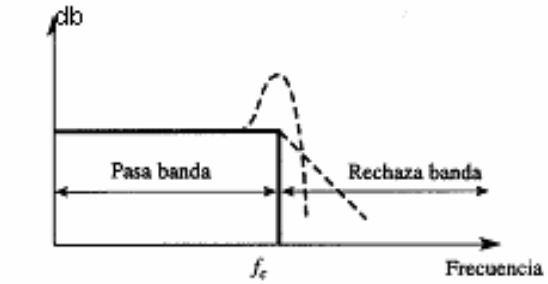


Fig. 1.8 Filtro pasa bajas

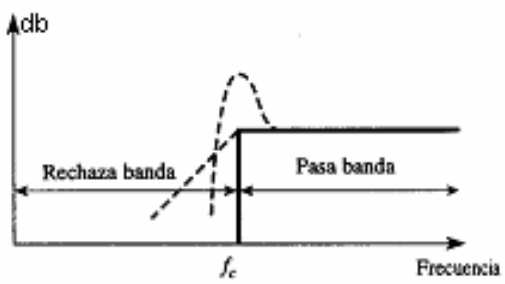


Fig. 1.9 Filtro pasa altas

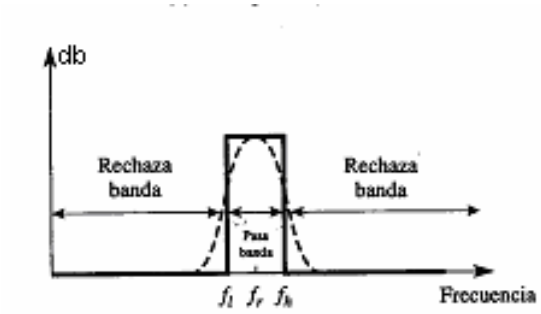


Fig. 1.10 Filtro pasa banda

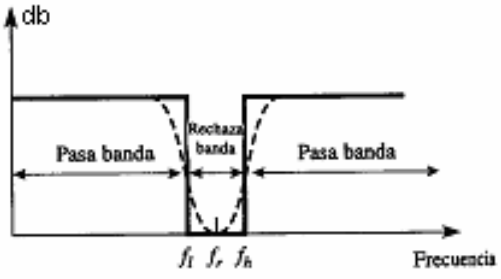


Fig. 1.11 Filtro supresor de banda

Para este trabajo se utiliza un filtro pasa banda debido a que solo se requiere un rango de frecuencias, el cual esta determinado por el ancho de banda de la señal de voz. Para poder utilizar la señal analógica de voz en el digitalizador de voz, esta debe pasar por un filtro ***pasa-banda*** a la frecuencia de la voz, como se muestra en la Figura 1.12. Esto con la finalidad de suprimir las frecuencias que se encuentran fuera del ancho de banda (BW) y para eliminar los ruidos que pueden estar presentes en el ambiente.

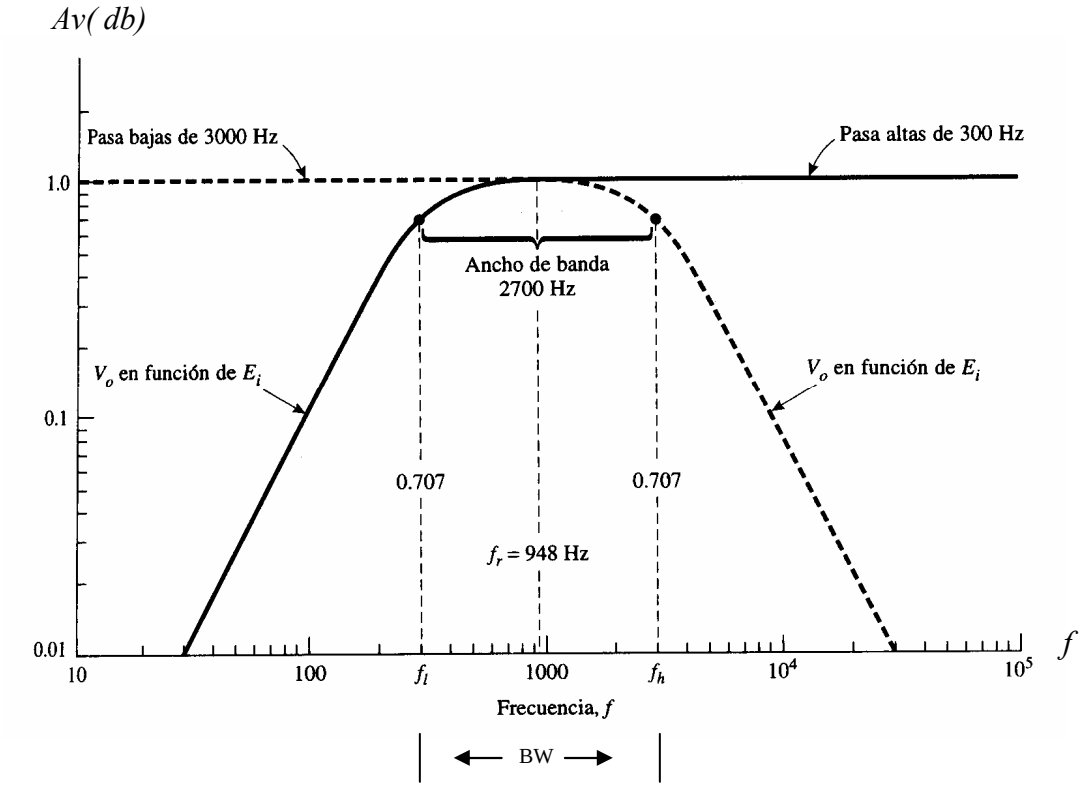


Fig. 1.12 Filtro pasa banda

1.5.2 MUESTREO

La digitalización consiste, básicamente en realizar de forma periódica, medidas de la amplitud de la señal y presentarlas en formato binario. Para esto se requiere tomar en cuenta el siguiente parámetro.

FRECUENCIA DE MUESTREO: Es un término que esta íntimamente relacionado con la conversión de una señal analógica a digital, ya que una señal analógica puede tomar valores infinitos de amplitud en un momento dado y la señal digital solo podrá adoptar un número determinado y finito de valores en ese momento. La frecuencia de muestreo es el número de veces por segundo que se muestrea la señal analógica para pasarla a formato digital como puede observarse en la figura 1.13.

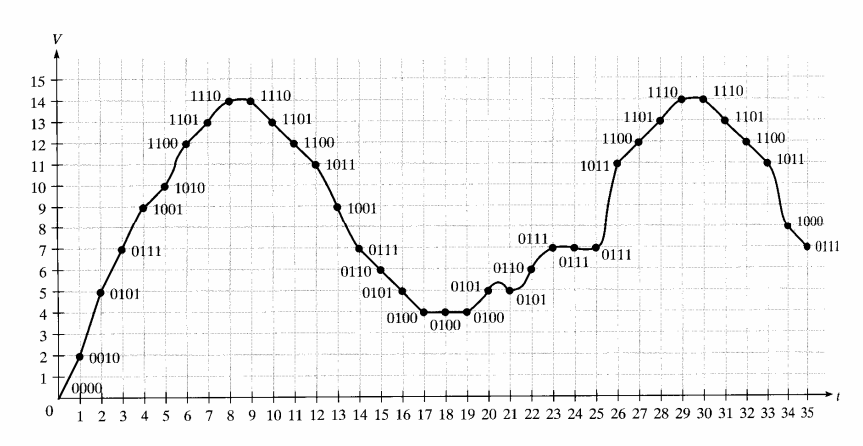


Fig. 1.13 Muestreo

TEOREMA DE NYQUIST: afirma que cuando se muestrea una señal, la frecuencia de muestreo debe ser mayor o igual que dos veces el ancho de banda de la señal de entrada, para poder reconstruir la señal original a partir de las muestras. Si BW es el ancho de banda de la señal y F_S es la frecuencia de muestreo, el teorema puede expresarse del siguiente modo:

$$\boldsymbol{F_S \geq 2BW}$$

.

Es decir, para poder reconstruir una señal analógica, se requiere que la frecuencia con que se toman las muestras sea igual o mayor que el doble de la máxima frecuencia que pueda contener la señal.

Tanto el aumento de la frecuencia de muestreo, como el aumento de la resolución, generan una mayor fidelidad en la señal digitalizada. Aunque también, lógicamente requiere unos volúmenes mayores de almacenamiento, así como unos mayores anchos de banda en los dispositivos de almacenamiento y distribución.

1.5.3 CUANTIFICACIÓN

El proceso de representar una variable por medio de un conjunto de valores distintos se denomina cuantización o cuantificación; y los valores distintos resultantes se denominan valores cuantizados o cuantificados. La variable cuantificada solo cambia en un conjunto finito de valores distintos, la figura 1.14) es una señal cuantificada en tiempo continuo (cuantificada sólo en amplitud).

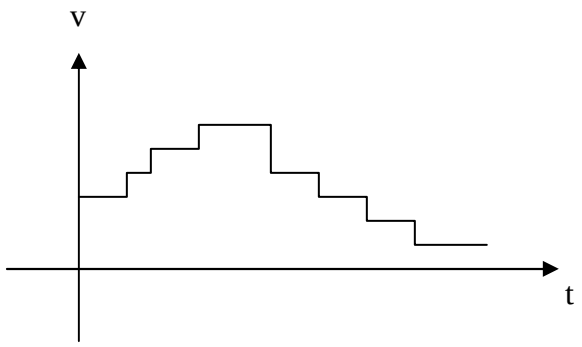


Fig. 1.14 Señal cuantificada

Se puede deducir de manera práctica, que es imposible convertir la señal de forma que cada valor analógico tenga un valor digital equivalente, ya que se pierde parte de la información. En resumen toda conversión analógica digital genera pérdidas, estas son imposibles de eliminar, ya que es una desventaja de las señales digitales. Una manera de no generar tantas perdidas es aumentando la resolución en el convertidor que realiza este proceso de cuantificación.

1.6 CONVERTIDOR ANALÓGICO DIGITAL (ADC)

Un convertidor analógico digital, también conocido como codificador, es un dispositivo que convierte una señal analógica en una señal digital, usualmente una señal codificada numéricamente. Dicho convertidor se utiliza como una interfaz entre un componente analógico y uno digital. Con frecuencia un circuito de muestreo y retención es una parte

integral de un ADC disponible comercialmente. La conversión de una señal analógica en la señal digital correspondiente (numero binario) es una aproximación, ya que la señal analógica puede adoptar un número infinito de valores, mientras que la variedad de números diferentes que se pueden formar mediante un conjunto finito de dígitos esta limitada. Este proceso de aproximación se denomina cuantificación.

1.7 TRANSDUCTOR

Un transductor es un dispositivo que convierte una señal de entrada en una señal de salida de naturaleza diferente, tal como los dispositivos que convierten una señal de presión en una salida de voltaje, las magnitudes de estas señales pueden tomar cualquier valor dentro de las limitaciones físicas del sistema.

1.8 ENTRADAS Y SALIDAS (E/S) DE LA COMPUTADORA (PC)

Durante la ejecución de un programa, normalmente la computadora lee de la memoria o escribe en ésta, y a su vez solicitar al procesador que lea o escriba en uno de los dispositivos periféricos, que generalmente se encuentran en los puertos de E / S, o a través de las ranuras de expansión que están interconectadas por medio de los buses internos con que pueda contar la PC, y a cada uno de estos le corresponde una dirección de puerto.
Puede haber cualquier número de estos dispositivos conectados al sistema de la computadora, este proceso se realiza normalmente por medio de una interfaz.

1.9 INTERFAZ

Las interfaces son componentes de un sistema en tiempo real que comunican el CPU con el exterior para recibir y enviar información necesaria para ejecutar una o varias tareas específicas. Las interfaces deben tener características deseables para trabajar en tiempo real:

-Funcionamiento eficiente.
-Capacidad de conexión con diversos dispositivos.

Además dependiendo del tipo de señal con la que se trabaje las interfaces pueden ser:

Digitales. Tratan con señales digitales (abierto/cerrado)
Analógicas. Tratan con señales continuas en el tiempo, las convierten a digitales y viceversa.
Todas estas características son indispensables para este proyecto, ya que se requiere que la interfaz tenga la capacidad de adquirir la señal de voz, procesar y controlar diferentes dispositivos de manera eficiente.

1.9.1 INTERFAZ PARALELA PROGRAMABLE (PPI)

Uno de los integrados universalmente usados en los sistemas basados en microprocesadores es sin duda la interfaz paralela programable (el circuito 8255). Este circuito fue inicialmente diseñado por Intel Corporation como parte del juego de integrados de apoyo a sus primeros sistemas de 16 bits (8086 y 8088).

La fuerte evolución en el diseño de computadoras ha convertido a gran parte del chipset del 8086 en piezas de museo debido a que muchas de las funciones que realizaban no tienen hoy día utilidad alguna (carece de sentido emplear viejos controladores de disco). Sin embargo existen una serie de componentes que conservan todavía hoy después de 20 años, toda su utilidad. En concreto refiriéndose al controlador de interfaz paralelo PPI 8255, un versátil y económico integrado de fácil conexión a cualquier sistema basado en microprocesador o microcontrolador, que proporciona de un modo elegante y sencillo puertos E / S disponibles.

1.9.2 BUS ISA

El bus es una ranura que se encuentra en la tarjeta madre, y permite la transmisión de datos del Microprocesador a otros dispositivos. En el bus se pueden insertar tarjetas o interfaces para ampliar la capacidad de la PC.

El BUS ISA, desde su origen fue un Bus de 8 bits y, como tal, estuvo en uso durante unos diez años, después fue ampliado a 16 bits y la frecuencia de operación del Bus de expansión, era inicialmente de 4.77 MHz, posteriormente esta velocidad fue aumentada a 6 MHz y luego a 8 MHz debido a la evolución de las computadoras.

1.10 OPTOACOPLADORES

Los componentes optoelectrónicos son ampliamente utilizados dentro de la electrónica actual, para la transmisión y aislamiento de señales electrónicas a través del medio óptico, además de su utilización como indicadores visuales (diodos led, paneles, etc).

Dentro de la electrónica de potencia se utilizan para el aislamiento de señales entre las etapas de control y potencia, proporcionando un alto aislamiento entre ambas.

La utilización de optoacopladores frente a otros sistemas de transmisión de las señales de control, presenta el inconveniente de necesitar de una fuente de alimentación externa al circuito de control, por el contrario presentan la gran ventaja de permitir trabajar a frecuencias elevadas.

CAPÍTULO II

En la figura 2.1 se muestra de forma general las diferentes etapas por las que pasa la señal de voz para ser procesada dentro de la PC, y finalmente controlar el motor de C.D.

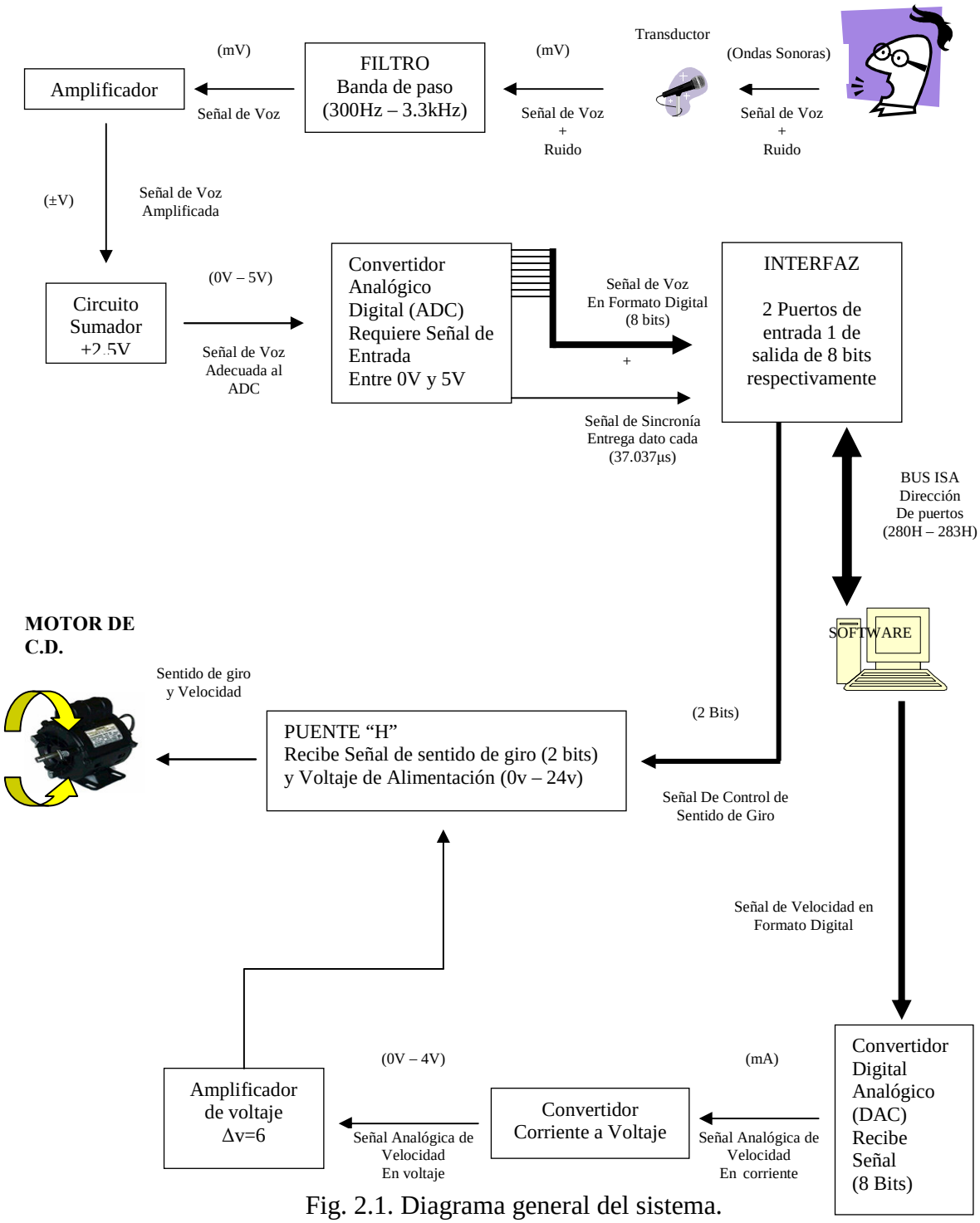


Fig. 2.1. Diagrama general del sistema.

2.2 DESARROLLO DEL SISTEMA DE LA ETAPA DE ENTRADA

Durante este capítulo se explica el diseño de la tarjeta de adquisición de la señal de voz así como su funcionamiento, se utilizarán los conceptos descritos en el capitulo I adaptándolos a los requerimientos del proyecto.

2.3 ADQUISICIÓN DE LA SEÑAL DE VOZ

Es necesario el uso de un transductor (micrófono) para convertir la señal de voz en una señal eléctrica. El micrófono transforma las ondas de presión en el aire (es decir, el sonido) en una señal de voltaje.

Se debe considerar la fidelidad en un micrófono para indicar la variación de sensibilidad con respecto a la frecuencia. Así mismo, la fidelidad viene definida como la propia respuesta en frecuencia del micrófono, si el sonido real fuese igual al sonido captado, ésta respuesta sería plana y su representación gráfica sería una línea recta donde la desviación sobre la horizontal seria de 0 dB. Cuanto más lineal sea la respuesta en frecuencia, mayor fidelidad tendrá el micrófono, puesto que el sonido captado por un micrófono nunca va a ser exactamente igual al real habrá frecuencias que han sido atenuadas, mientras que otras habrán sido incrementadas.

En la práctica la mayoría de los micrófonos ofrecen mejor sensibilidad ante unos tonos que ante otros, y de hecho, se comercializan así, divididos para los distintos sonidos que se desean grabar. Además, hay micrófonos de ínfima calidad que ofrecen una respuesta irregular.

Para este proyecto el micrófono modelo ACER (fig. 2.2) fue seleccionado debido a que presento buena fidelidad para la adquisición de la voz, mediante pruebas realizadas en los laboratorios, aunque pueden existir otros modelos mas recientes con mejor respuesta.

Fig. 2.2 Micrófono modelo ACER

Para visualizar las señales que se generaban en la entrada del micrófono se utilizó un osciloscopio como se puede observar en la figura 2.3, es una señal muy pequeña aproximadamente 500mVpp de amplitud.

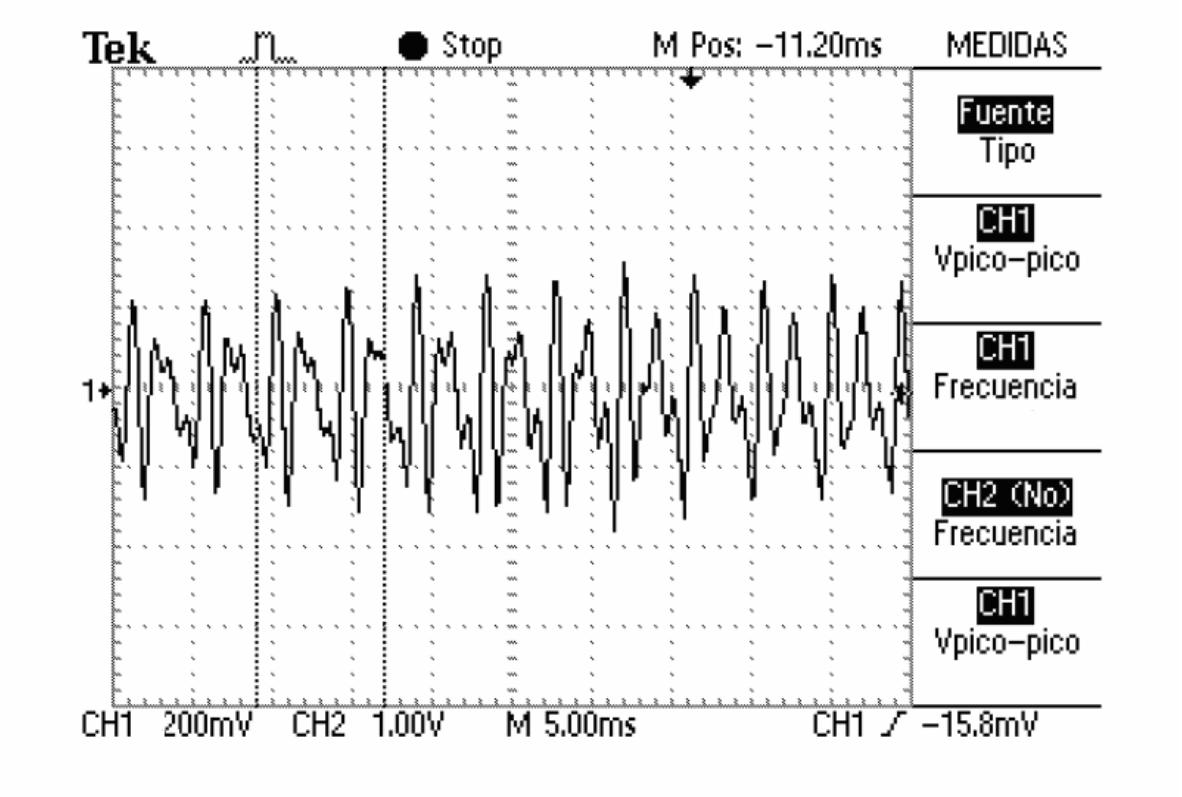


Fig. 2.3 Señal de voz tomada del micrófono.

2.4 FILTRADO DE LA SEÑAL DE VOZ

Para poder utilizar la señal analógica de voz que proviene del micrófono es necesario eliminar el ruido del medio ambiente con un filtro adecuado que permita trabajar exclusivamente con el ancho de banda de la voz de 300 Hz a 3.3 KHz.

2.4.1 RESPUESTA EN FRECUENCIA.

Los filtros pasa banda son selectores de frecuencia. Permiten elegir una determinada banda de frecuencias de entre todas las frecuencias que puede haber en un circuito. Cuando se conecta en serie la salida de un circuito con la entrada de un segundo circuito, se dice que las etapas de ganancia están en cascada.

En general, para construir un filtro pasa banda se conecta en cascada un filtro pasa bajas y un filtro pasa altas. Es importante que las frecuencias de las secciones de ambos filtros se traslapen y que ambas tengan la misma ganancia en su banda de paso (fig. 2.4).

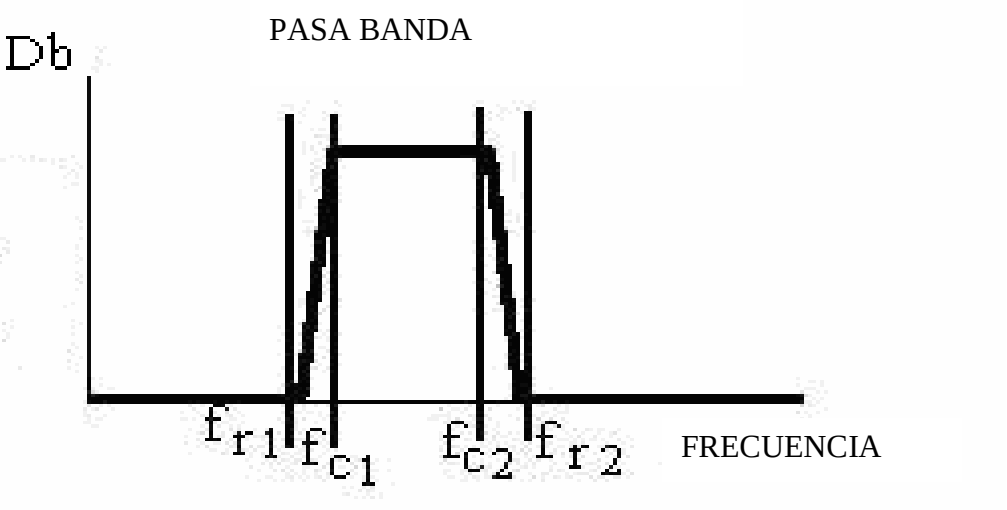


Fig. 2.4 Respuesta en frecuencia filtro pasa banda

Para este caso se acopló en la primera etapa un filtro pasa bajas de 3000 Hz y en la salida un filtro pasa altas de 300 hz con la finalidad de solo dejar pasar la señal de voz.

El filtro pasa banda obtenido mediante los filtros pasa bajas y pasa altas conectados en cascada tiene las siguientes características:

1. La frecuencia de corte inferior F1, esta determinada sólo por el filtro pasa altas y es de 300Hz.

2. La frecuencia de corte superior, F2, está definida exclusivamente por el filtro pasa bajas y es de 3000 Hz.

3. La ganancia tendrá su valor máximo en la frecuencia resonante, esto es dentro del ancho de banda.

Para determinar la selección del tipo de filtro se realizaron pruebas mediante el uso del software de simulación incluido en el paquete de Orcad.

Dentro de las pruebas que se realizaron, se observo que el filtro pasabanda Butterworth de 10º orden, con una banda de paso entre los 300 Hz y 3.3 KHz, respondió satisfactoriamente a las necesidades requeridas en las frecuencias de corte y la estabilidad en el ancho de banda, por lo que se decidió diseñar éste, ya que la finalidad es obtener una señal con el menor ruido posible.

La figura 2.5 muestra la respuesta que tiene el filtro Butterworth que se utiliza en la tarjeta de adquisición de la señal de voz.

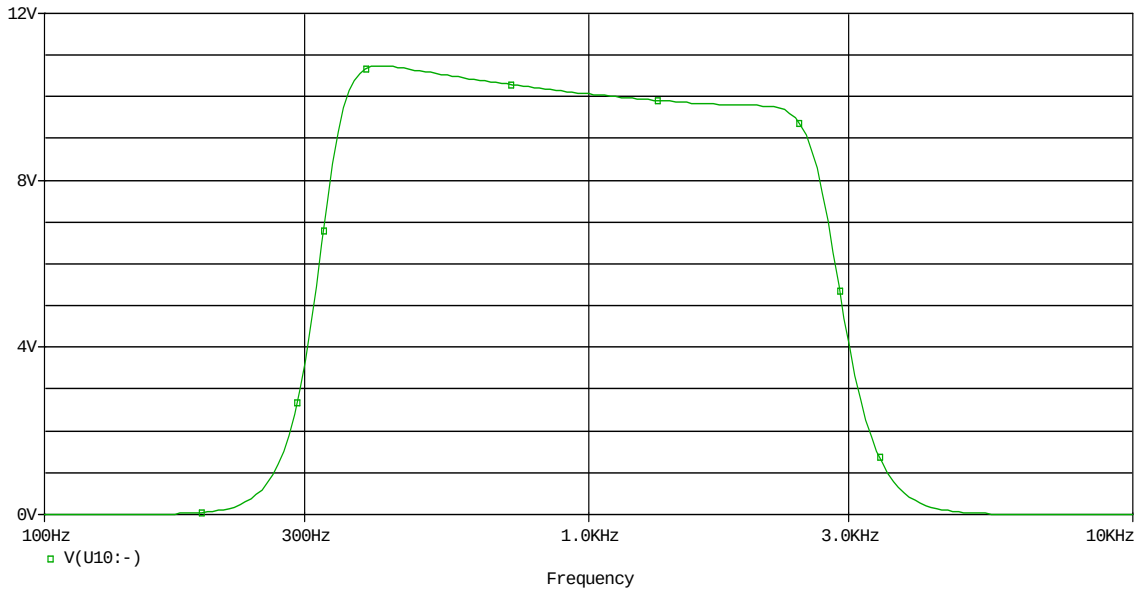


Fig. 2.5. Respuesta del filtro Butterworth

Así mismo un filtro de tipo Chebyshev del mismo orden y características tenía un mejor comportamiento en las frecuencias de corte ya que se asemeja mucho al corte de un filtro ideal, pero durante la banda de paso presentaba bastante inestabilidad, lo que no paso con el filtro Butterworth (como se muestra en las figura 2.6).

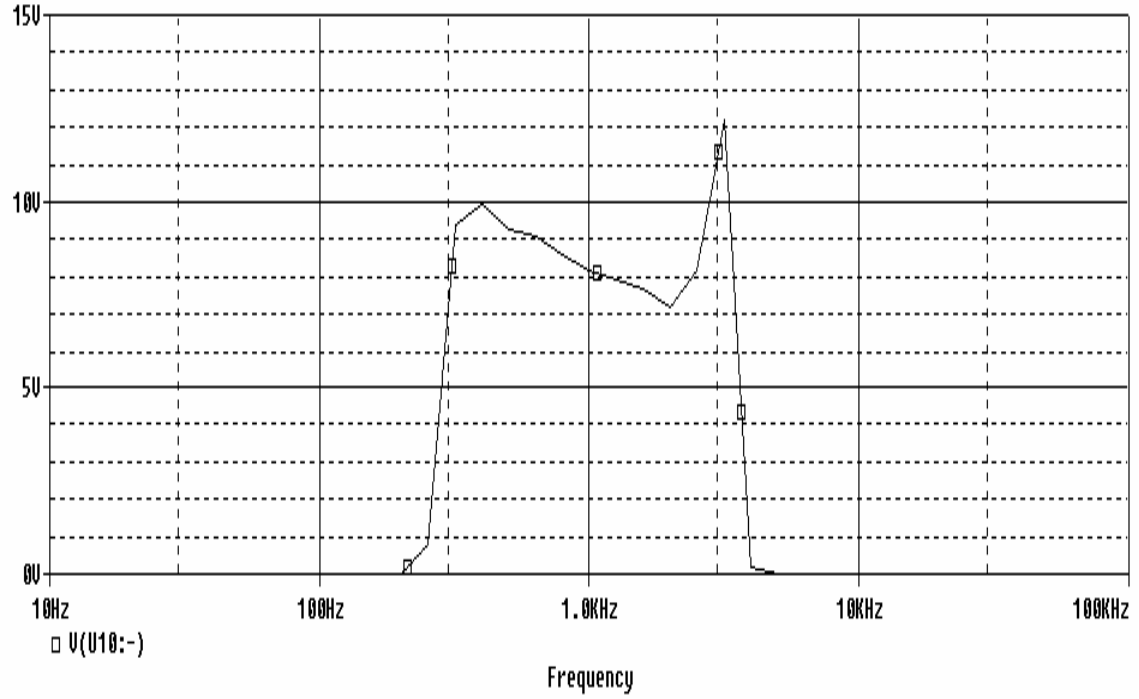


Fig.2.6 Respuesta del Filtro Chebyshev

Por lo tanto, tomando en consideración el comportamiento de ambos filtros se decidió utilizar el filtro pasabanda Butterworth de 10º orden (fig. 2.7), que fue implementado con circuitos integrados LM837 que consta de 4 amplificadores operacionales especiales para señales de audio, que se caracterizan principalmente por ser amplificadores de bajo ruido.

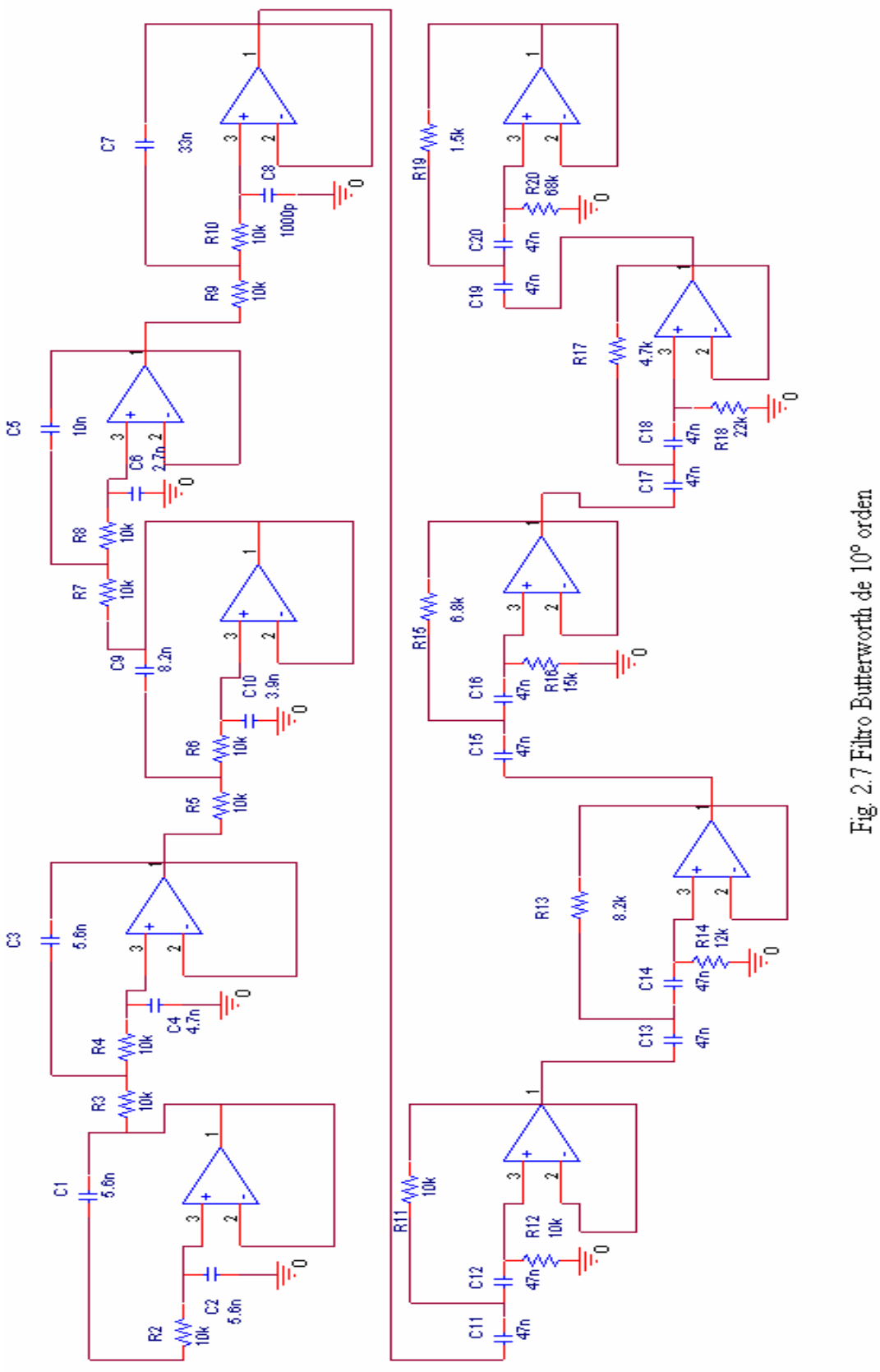


Fig. 2.7 Filtro Butterworth de 10º orden

En la figura 2.8 se muestra la señal de voz una vez que ha pasado por el filtro Butterworth, ahora esta señal será introducida a una etapa amplificadora.

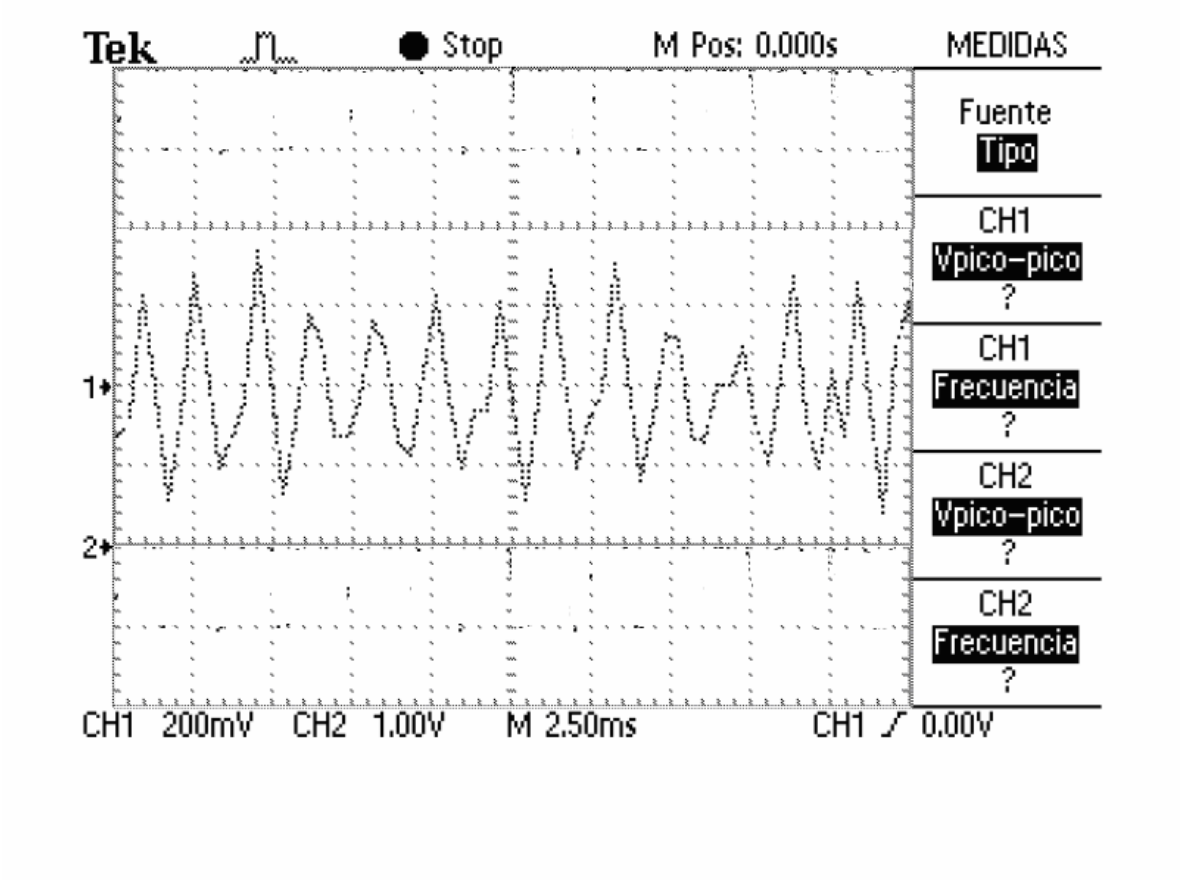


Fig. 2.8 Señal de voz Filtrada

2.5 AMPLIFICACION DE LA SEÑAL DE VOZ

Se obtuvo una señal libre de ruido, pero de amplitud muy baja, por lo cual se decidió implementar una etapa de amplificación con un C.I. LF356 optimo para señales de audio, ya que tiene la característica de presentar bajo ruido a la salida. Y alta impedancia a la entrada como se muestra en la figura 2.9.

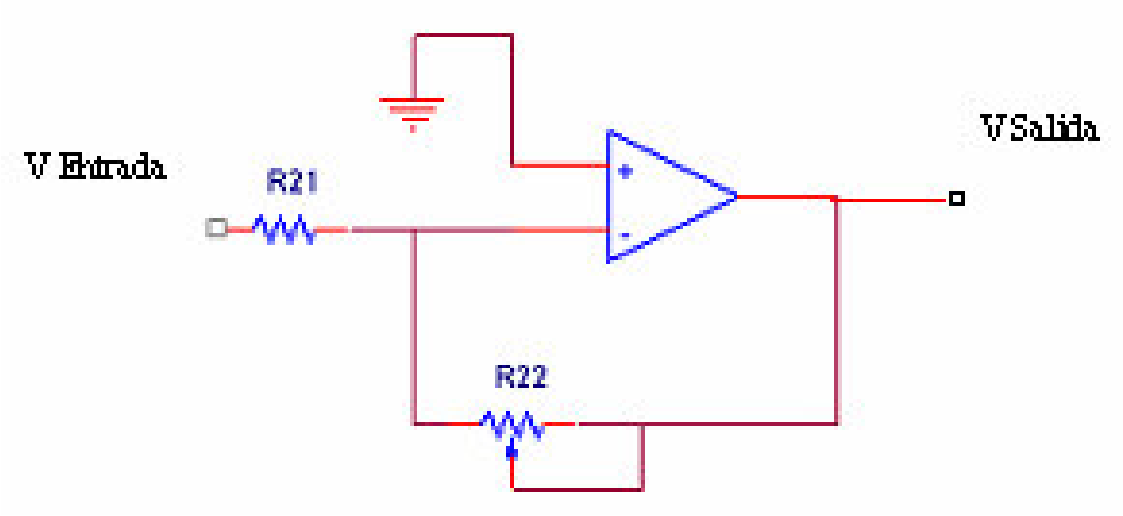


Fig. 2.9 Amplificador con ganancia de 4

La ganancia fue determinada de tal forma que a la salida se obtuviera un voltaje de 4 V para aprovechar al máximo la resolución del convertidor, ya que como se describirá mas adelante, este requiere de cierto voltaje para trabajar en óptimas condiciones.

La figura 2.10 muestra la señal de voz filtrada y amplificada, ésta señal es parte de la entrada de un circuito sumador que a continuación se describe.

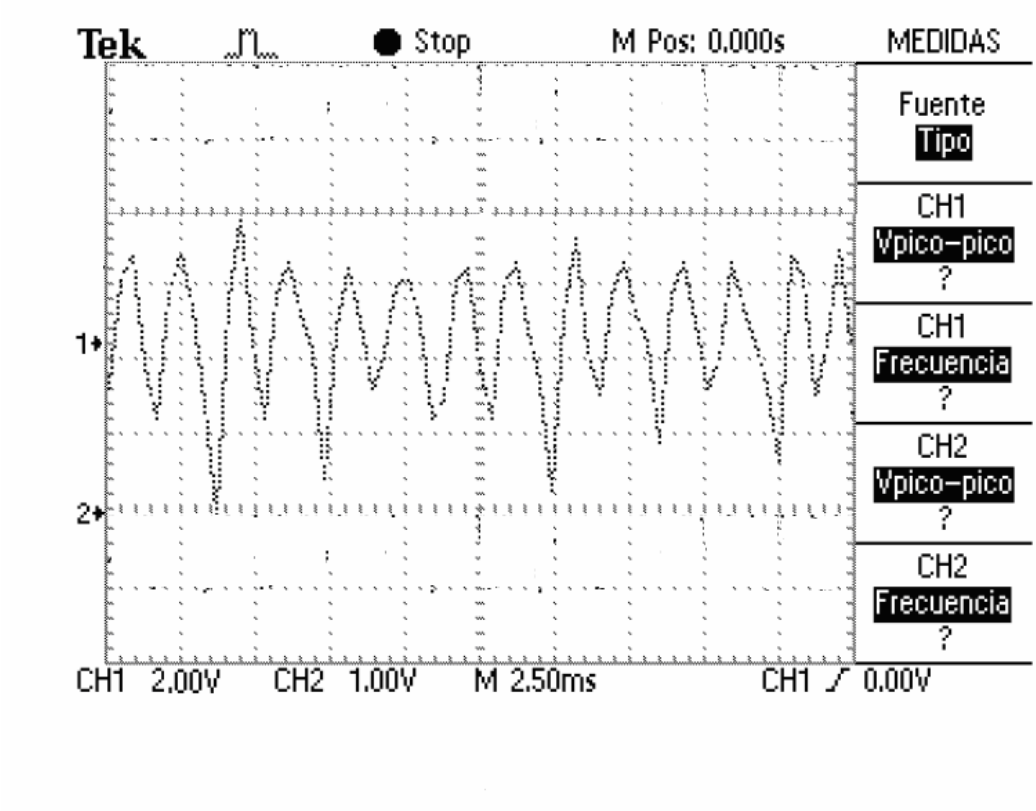


Fig. 2.10 Señal de voz filtrada y amplificada

2.6 CIRCUITO SUMADOR.

Este circuito es simplemente un amplificador sumador que utiliza un CI. LF356 con la finalidad de añadirle a la señal de voz proveniente del amplificador un voltaje de corriente directa de 2.5 volts para poder acoplarlo al ADC(figura 2.11), esto se hace para aprovechar toda la señal de la voz, de lo contrario se perdería toda la parte negativa de la señal y esto no conviene, porque no se obtendrían todos los datos para poder hacer la conversión a digital.

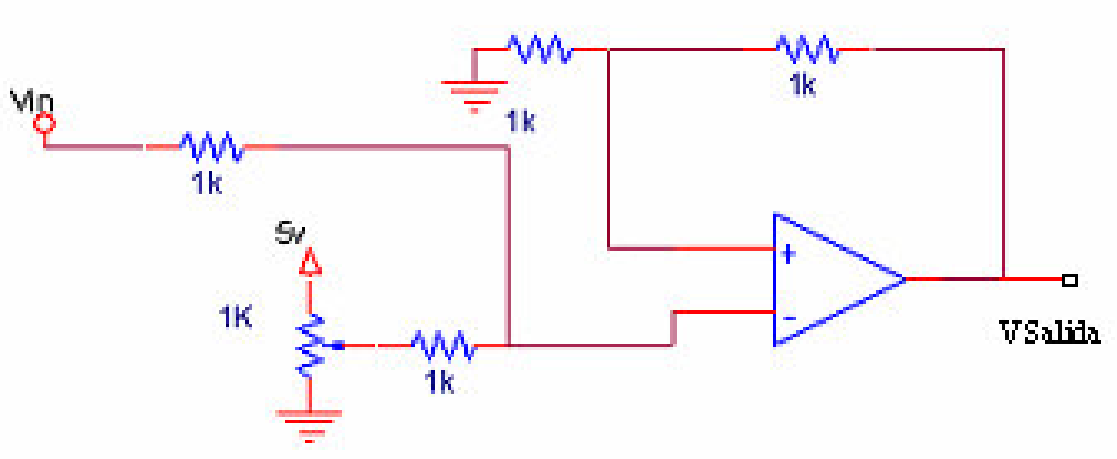


Fig. 2.11 Sumador de 2.5 V

La figura 2.12 muestra la señal de voz filtrada, amplificada con la suma de los 2.5v obteniendo la señal adecuada para el acoplo con el ADC.

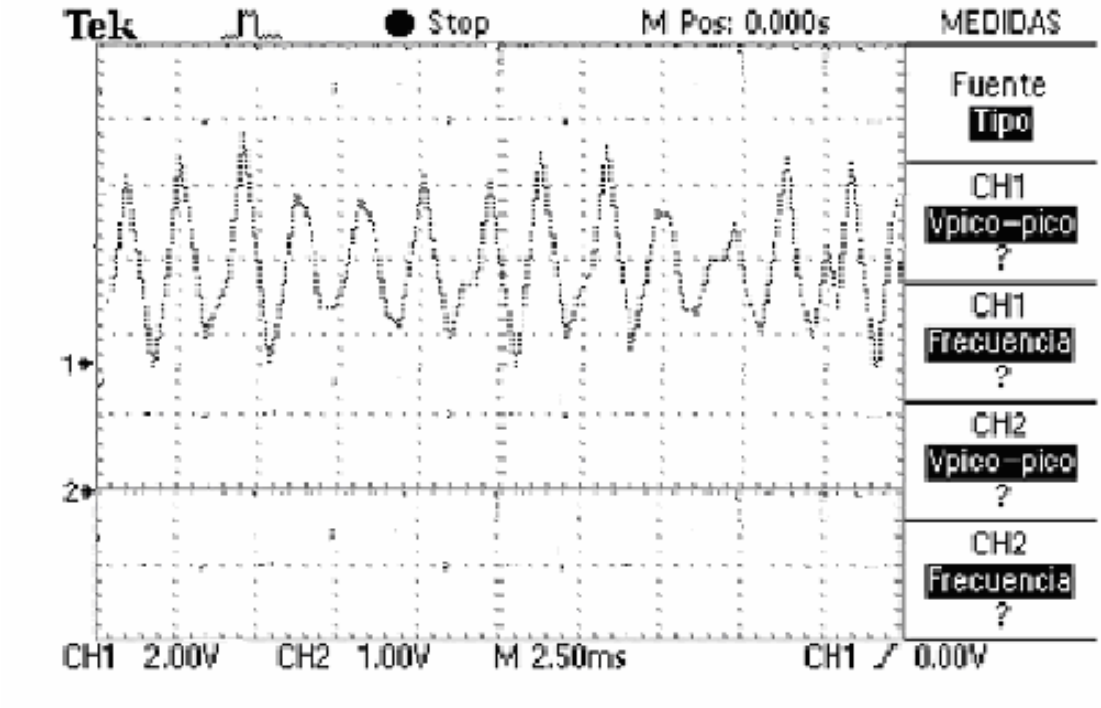


Fig. 2.12 Señal de voz filtrada, amplificada y sumada con 2.5v.

2.7 CONVERTIDORTES ANALOGICO DIGITAL (ADC)

Dado que el número de bits que se tienen de un convertidor es finito, el código de salida deberá ser siempre el correspondiente al valor mas cercano que pueda representarse mediante los bits, la conversión digital efectúa una cuantificación en la entrada analógica, acotándola entre dos niveles consecutivos cuya distancia es precisamente el grado de resolución obtenido. El grado de resolución se refiere al valor mínimo de voltaje que el convertidor analógico/digital necesita para cambiar al siguiente número binario.

$$q = \frac{F.S.}{2^n - 1}$$

Donde:

F.S. = Escala completa

q = Resolución

n = Número de bits

Existen diversos tipos de convertidores en igual forma utilizados para efectuar la conversión, en unos casos se efectúa la conversión directa, por comparación contra una tensión de referencia, en otros casos se efectúa una transformación a una variable intermedia, como puede ser el tiempo, también puede efectuarse la conversión analógico/digital efectuando una conversión inversa digital/analógico.

El proceso de conversión analógico/digital es generalmente mas complejo y largo que el proceso inverso digital/analógico, se han creado y utilizado muchos métodos de conversión analógico/digital como son:

El método de rampa digital, aunque una de las desventajas de este es que el tiempo de conversión es lento, ya que el contador se reestablece después de cada nueva conversión.

El método de pendiente dual, en este el método de conversión es de los más lentos ya que oscila entre los 10ms y 100ms.

El método de conversión analógico digital de voltaje a frecuencia, aunque este es un método de conversión simple es difícil alcanzar un alto grado de precisión debido a la dificultad en el diseño.

El método de aproximaciones sucesivas, los convertidores contienen un valor fijo en su tiempo de conversión que no depende del valor de la entrada analógica, el convertidor de aproximaciones sucesivas no utiliza ningún contador para dar la entrada en el bloque del convertidor digital/analógico, pero en cambio usa un registro con lógica de control que modifica el contenido del registro bit a bit hasta que los datos del registro son el equivalente digital de la entrada analógica.

El método de comparación, es el único caso en que los procesos de cuantificación y codificación están claramente separados. El primer paso se lleva a cabo mediante comparadores que discriminan entre un número finito de niveles de tensión. Estos comparadores reciben en sus entradas la señal analógica de entrada junto con una tensión de referencia, distinta para cada uno de ellos. Al estar las tensiones de referencia escalonadas, es posible conocer si la señal de entrada está por encima o por debajo de cada una de ellas, lo cual permitirá conocer el estado que le corresponde como resultado de la cuantificación.

Es por esto que se utilizo el ADC0820 (figura 2.13), ya que uno de sus beneficios es que su mecanismo de entrada tiene la capacidad de medir señales que estén cambiando a una gran velocidad como es la voz, sin ayuda de un mecanismo externo de muestreo y retensión.

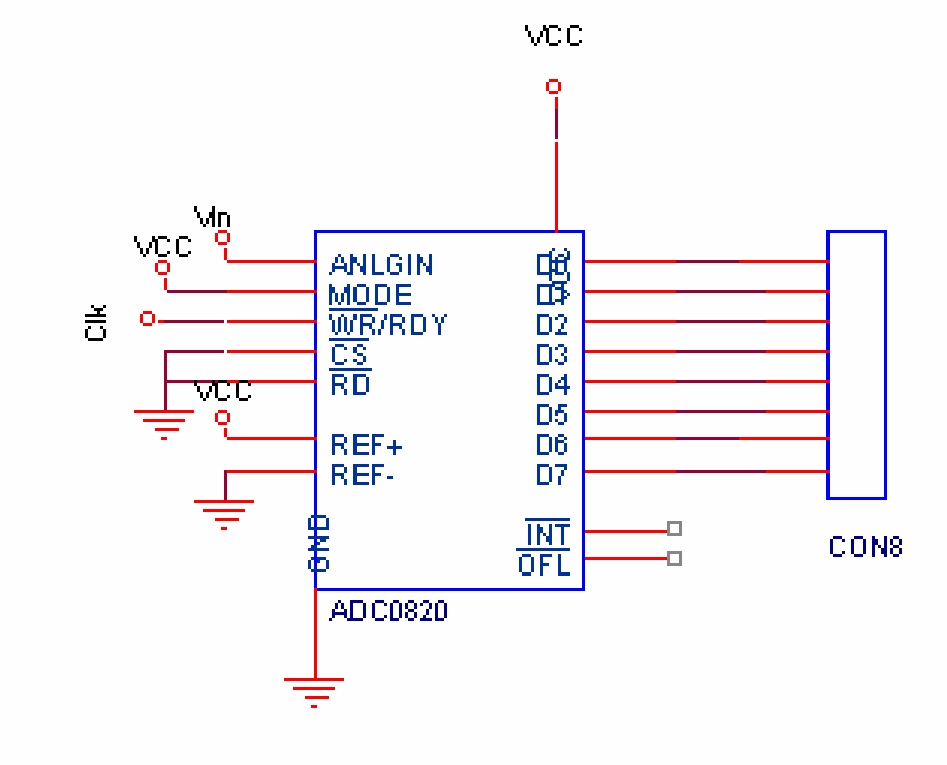


Fig. 2.13 Convertidor Analógico Digital

2.8 CIRCUITO DE RELOJ

Como se necesitaba que la señal de reloj fuera lo mas estable posible se selecciono el circuito XR2207CP ya que este circuito (figura 2.14) trabaja como un generador de onda cuadrada, triangular, diente de sierra y pulsos, y entre sus principales características ofrece un ciclo de trabajo ajustable de 0.1% a 99.9%, además de un amplio rango de frecuencias de trabajo.

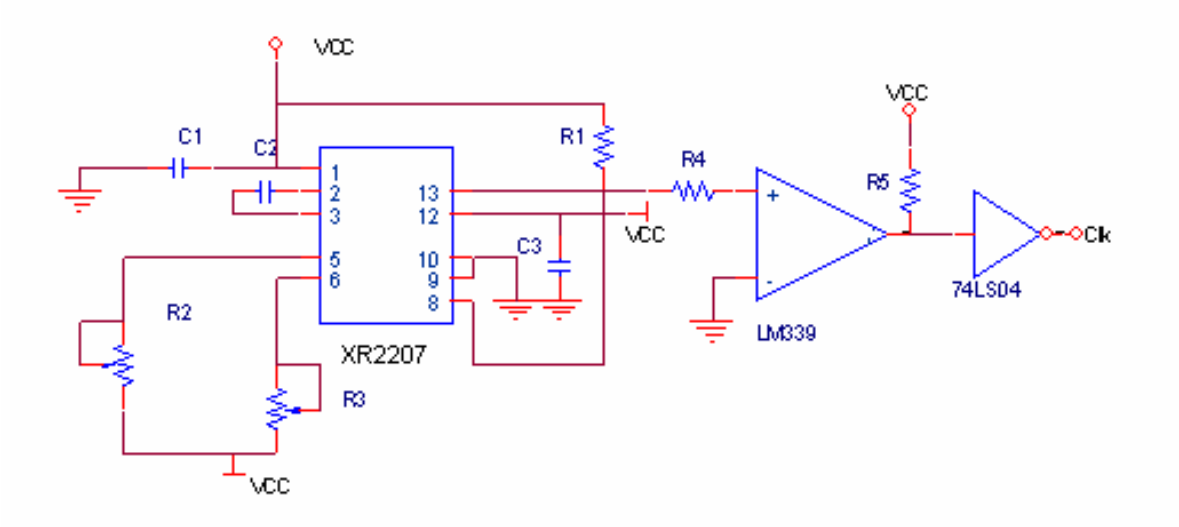


Fig. 2.14 Circuito de reloj

Este arreglo genera una señal de reloj de 27 KHz que es la frecuencia a la que se requiere muestrear la señal analógica.

CAPÍTULO III

3.1 DISEÑO DE LA INTERFAZ DE ENTRADA Y SALIDA.

Para la interacción del digitalizador de voz con el CPU se diseño una tarjeta de entrada y salida de manera externa, esta tarjeta cuenta con la capacidad para transportar los datos hacia el CPU de manera que su funcionamiento no interfiera en la ejecución de otras tareas, esto es posible gracias al circuito mostrado en la figura 3.1 (ver apéndice A), el cual es un diseño que utiliza un C.I 74l38 como multiplexor para direccionar y un circuito programable C.I. 8255 que son conectados a través de un bus ISA (Industry Standard Architecture).

3.1.1 INTERFAZ PARALELO PROGRAMABLE

El circuito PPI (Interfaz Periférica Programable) 8255A de Intel es un dispositivo programable de entrada/salida de propósito general diseñado para ser usado con microprocesadores, tiene 24 terminales de entrada/salida las cuales se pueden programar individualmente en 2 grupos de 12 y se puede utilizar en 3 diferentes modos de operación.

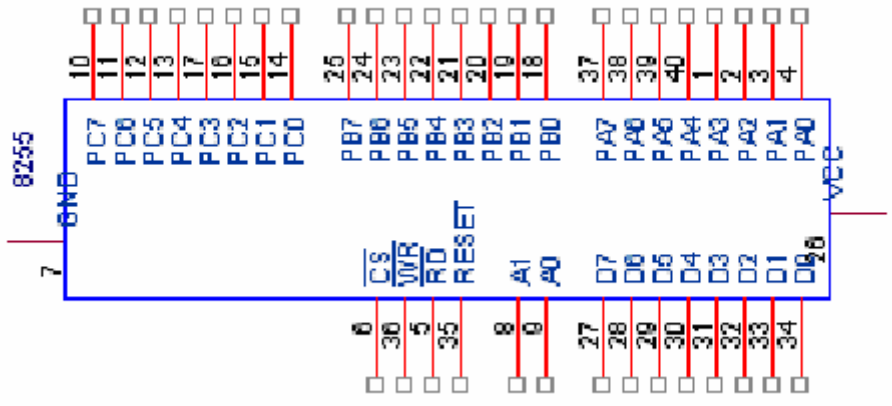


Fig. 3.1 PPI

- *Grupo A:* PA0..PA7 + PC4..PC7
- *Grupo B:* PB0..PB7 + PC0..PC3

Puertos A, B, C

El 8255A contiene 3 puertos (A, B, C) de 8 bits cada uno. Todos pueden ser configurados en una amplia variedad de características funcionales a través del software del sistema, pero cada uno tiene sus características especiales.

Puerto A. Una salida de datos de 8 bits latch/buffer y una entrada de datos de 8 bits latch.
Puerto B. Una entrada/salida de datos de 8 bits latch/buffer y una entrada de datos de 8 bits buffer.
Puerto C. Una salida de datos de 8 bits latch/buffer y una entrada de datos de 8 bits buffer únicamente. Este puerto puede ser dividido en dos puertos de 4 bits bajo el modo de control. Cada puerto de 4 bits contiene un latch de 4 bits y este puede ser usado para el control de las señales de estado de entrada en conjunto con los puertos A y B.

Selección del modo

Existen tres modos básicos de operación que pueden ser seleccionados por el software del sistema:

- *Modo 0.* Modo básico de entrada/salida.

- *Modo 1.* Modo de "Strobe" de entrada/salida. Un puerto de ocho bits programable como entrada o salida y un puerto de cuatro bits de control.

- *Modo 2.* Modo de bus bidireccional. Este modo configura al puerto A como un puerto bidireccional dejando los cinco bits más significativos del puerto C

De los tres modos de funcionamiento, se utilizó el modo 0, ya que se configuró el sistema con entradas de 8 bits y salidas de 8 bits para control de una etapa de potencia, que se

describirá en el siguiente capitulo. Los modos de funcionamiento se programan a través de una palabra de control de 8 bits.

La palabra de control de programación de modo 0, consta de 8 bits que van desde D0 hasta D7, donde D0 es el bit menos significativo (LSB) y D7 es el bit más significativo (MSB). La figura 3.2 muestra la estructura que conforma la palabra de control de programación del PPI 8255A.

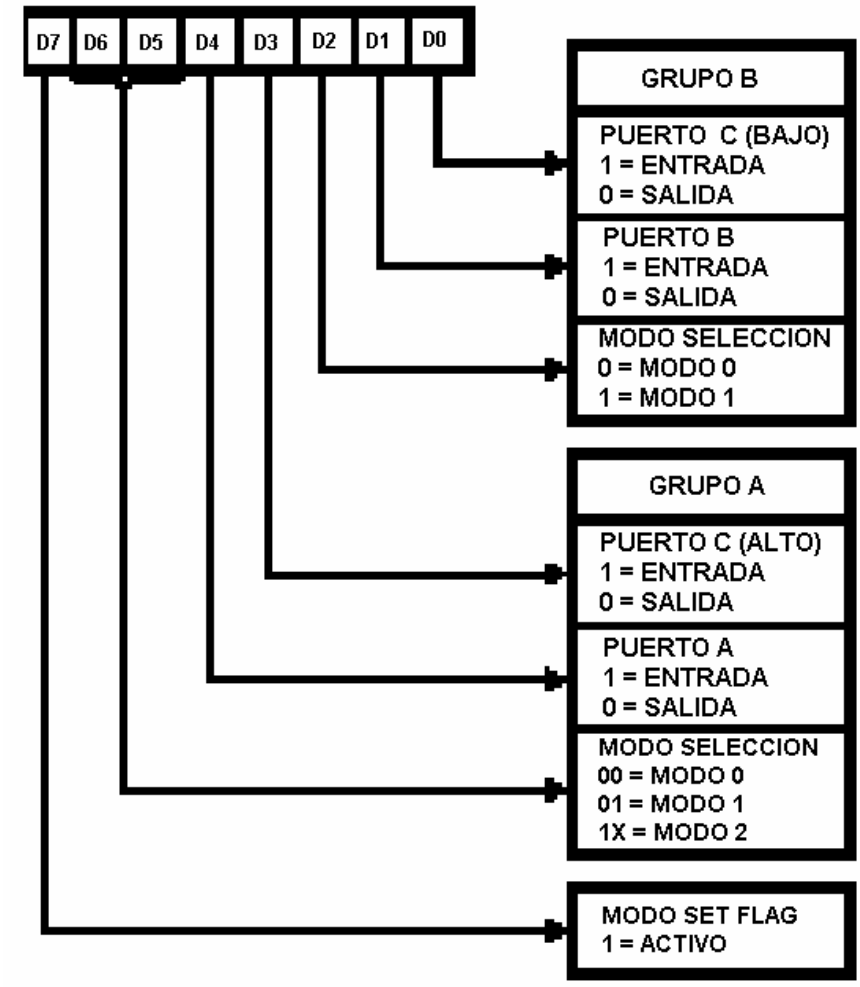


Fig. 3.2 Palabra de control para programar el PPI.

La palabra de control usada en este proyecto es 10011001 (que expresada en forma hexadecimal equivale a 0x99), esta indica el modo de programación del PPI:

D0	Activa el puerto C bajo de entrada	(1)
D1	Activa el puerto B de salida	(0)
D2	Activa el modo 0	(0)
D3	Activa el puerto C alto de entrada	(1)
D4	Activa el puerto A de entrada	(1)
D5	Activa el modo 0	(0)
D6	Activa el modo 0	(0)
D7	Activa la bandera del PPI	(1)

Existen cinco grupos de señales en el PPI 8255, como se muestra en la figura 3.3, además de los dos terminales de alimentación:

- Grupo de control
- Bus de datos
- Puerto A
- Puerto B
- Puerto C

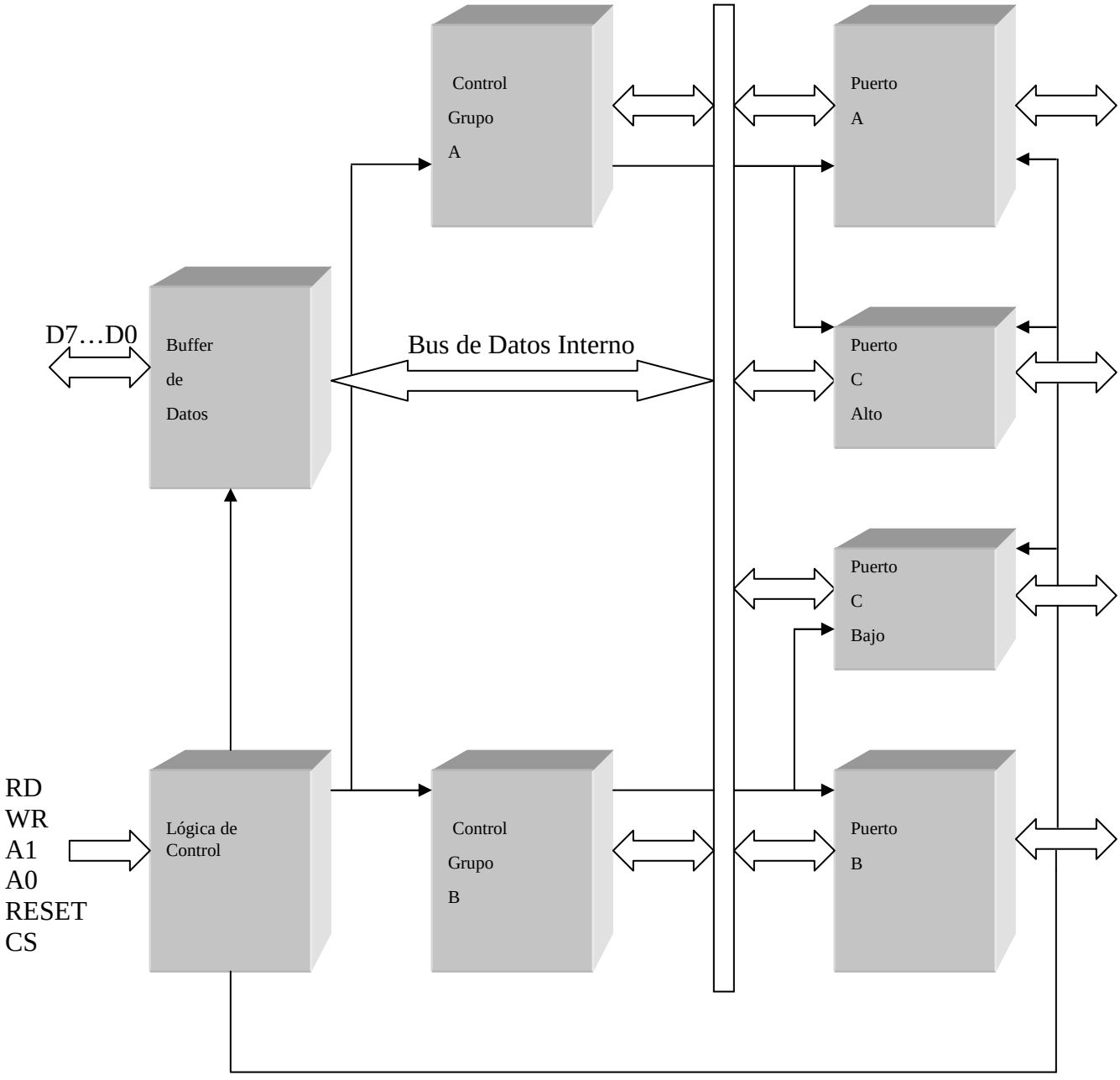


Fig. 3.2 Descripción de las señales que proporciona el PPI 8255.

Buffer del Bus de Datos

Este buffer bidireccional de 8 bits y con la característica de tener tercer estado en sus terminales se utiliza para conectar el PPI al bus de datos del sistema. Los datos son transmitidos o recibidos por el buffer a través de las instrucciones del CPU , siendo estas de entrada o de salida. Las palabras de control y el estado de la información también se transmiten a través del buffer.

R/W y Control Lógico

La función de este bloque es manejar todas las transferencias internas y externas relacionadas con los Datos, Control o Palabras de Estado. Acepta entradas provenientes de los buses de Control y Direcciones del CPU y emite comandos a los grupos de control.

CS

Selección del chip. Un nivel bajo en la entrada de esta terminal, habilita la comunicación entre el 8255A y el CPU.

RD

Lectura. Un nivel bajo en la entrada de esta terminal, habilita al 8255A para enviar el dato o estado de la información al CPU a través del bus de datos. Esto permite al CPU leer del 8255A.

WR

Escritura. Un nivel bajo en la entrada de esta terminal, habilita al CPU para escribir datos o palabras de control al 8255A.

A0 y A1 Puertos de selección 0 y 1. Estas señales de entrada, en conjunto con las señales RD y WR, controlan la selección de uno de los tres puertos o los registros de la palabra de control. Normalmente se conectan a los bits menos significativos del bus de direcciones (A0 y A1).

Reset Reinicialización. Un nivel alto en la entrada de esta terminal limpia el registro de control y todos los puertos (A, B, C) son colocados en el modo de entrada.

Control de Grupo A y B

La configuración funcional de cada puerto es programado por el software del sistema. Es decir, las salidas del CPU van a la palabra de control del 8255A. La palabra de control contiene información tal como "modo", "bit de habilitación", "bit de rehabilitación", etc., que inicia la configuración funcional del 8255A.

Cada uno de los bloques de control (Grupo A y Grupo B) acepta comandos del control lógico R/W, recibe palabras de control del bus de datos interno y envía los comandos apropiados para los puertos asociados.

Entre las aplicaciones actuales podemos comprobar como el 8255 se encuentra con facilidad en tarjetas de expansión de puertos para el bus ISA de la PC, ya que simplifica enormemente la elaboración de la placa de circuito impreso aportando suficiente potencia de control. Es también ideal para expansión de puertos E/S en monoplacas o gestión de periféricos como conversores analógico / digital entre otros.

3.1.2 BUS ISA Y MULTIPLEXOR DE DIRECCIÓN

El CPU con el cual se realiza este proyecto cuenta precisamente con un BUS de expansión ISA y la tarjeta que se utiliza (ver apéndice A), es precisamente la interfaz que se encarga de realizar la transferencia de los datos de la etapa de digitalización de la voz hacia la etapa de control y potencia.

El Bus de expansión contempla líneas de datos, líneas de control y líneas de direcciones (Ver figura 3.4). La polarización para los equipos periféricos es proporcionada en las líneas B1 y B31 (tierras) y en las líneas B3 y B29 (+5V). Todas las líneas del bus de expansión son compatibles con niveles TTL. El bus de expansión contiene los siguientes elementos:

- 1 bus de datos bidireccional de 8 bits.
- 20 líneas de dirección
- 6 niveles de interrupción
- 3 canales para líneas de control DMA
- 1 canal para pruebas de línea
- Líneas de control para lectura de memoria y lectura o escritura de entrada/salida
- Energía y tierra para los adaptadores

A continuación se dará una descripción de las señales involucradas en el bus de expansión.

+[A0-Al9]

Son las direcciones de los bits 0 al 19 que se utilizan para direccionar la memoria y los dispositivos de entrada/salida del sistema. Estas líneas pueden ser generadas por el procesador o el controlador DMA (Direct Memory Access).

+ALE

(Address Latch Enable). Esta señal habilita los acondicionadores de las direcciones de salida del 8088.

CLOCK

Reloj del sistema con una frecuencia de 4.77 MHz y un ciclo de trabajo del 33 %, producida por el generador de reloj 8284 (la frecuencia de 14.318 MHz del cristal se divide entre tres).

+AEN

(Address Enable). Inhibe al controlador del bus 8288 y a los acondicionadores de las direcciones para permitir que el DMA asuma el control del bus.

+[DO-D7]

Líneas de datos que constituyen el bus bidireccional de 8 bits a transferencia de datos, comandos e información.

[DACK 0-3]

(DMA Acknowledge Channels). Estos canales de reconocimiento del DMA activos bajos, notifican a dispositivos periféricos individuales cuando se ha otorgado un servicio del DMA.

[DRQ 1-3]

(DMA Request Channels). Un dispositivo periférico puede obtener servicio del DMA a través de estos canales de petición de entrada.

+T/C

Señal de salida que representa la terminación de un servicio de DMA.

-I/O CH CK

(I/O Channel Check). Señal generada por una tarjeta externa en el bus de expansión que informa al CPU de una falla catastrófica por medio de la circuitería del NMI (No Mascarable Interruption).

+I/O CH RDY

(I/O Channel Ready). Señal de "handshake" (protocolo) para una, tarjeta externa en el bus de expansión empleada cuando se insertan estados de espera a los ciclos de reloj del CPU o un DMA.

-IOR

(I/O Read). El CPU activa en nivel bajo esta señal para realizar una lectura hacia un dispositivo periférico del sistema.

-IOW

(I/O Write). EL CPU activa en nivel bajo esta señal para realizar una escritura hacia un dispositivo periférico del sistema.

+[IRQ 2-7]

(Interrupt Request Channels). Estas señales de entrada son canales de petición de interrupción pertenecientes al 8259; los niveles de prioridad de los mismos son establecidos por el diseñador del sistema.

-MEMR

(Memory Read). Señal que el CPU habilita en bajo cuando realiza una lectura a memoria.

-MEMW

(Memory Write). Señal que el CPU habilita en bajo cuando realiza una escritura a memoria.

+OSC

Señal cuya frecuencia es equivalente a la frecuencia del cristal de 14.318 MHz, usada exclusivamente por la circuiteria de video.

+RESET

Señal para reestablecer o inicializar el sistema lógico en el encendido de la máquina o durante bajas de voltaje.

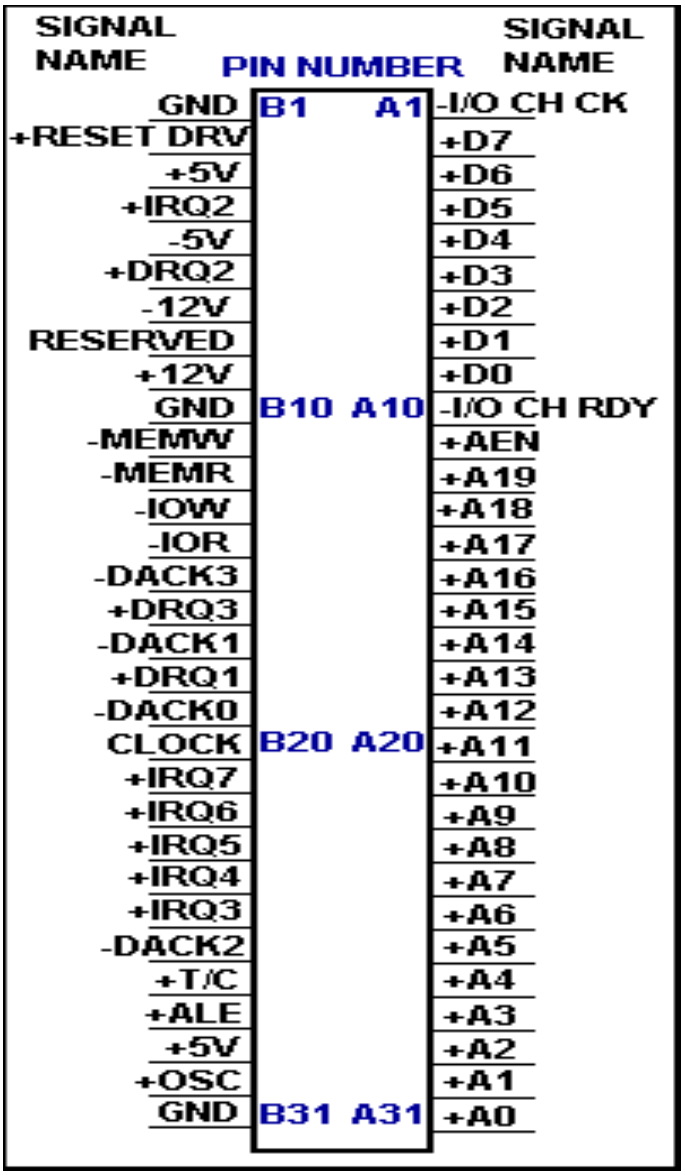


Fig. 3.4Bus de expansión ISA.

Esta tarjeta además de servirse del bus ISA para interconectarse con el CPU, necesita un multiplexor, el cual ofrece una gama de direccionamientos y por medio de un DIP switch se determina la dirección de cada uno de los puertos utilizados para así indicarle al CPU las acciones que ejecutará a través de un software, como se vera mas adelante. La dirección utilizada para el proyecto es la 280h, 281h, 282h, 283h, las cuales no interfieren con las direcciones que ocupa el CPU. (ver Figura 3.5)

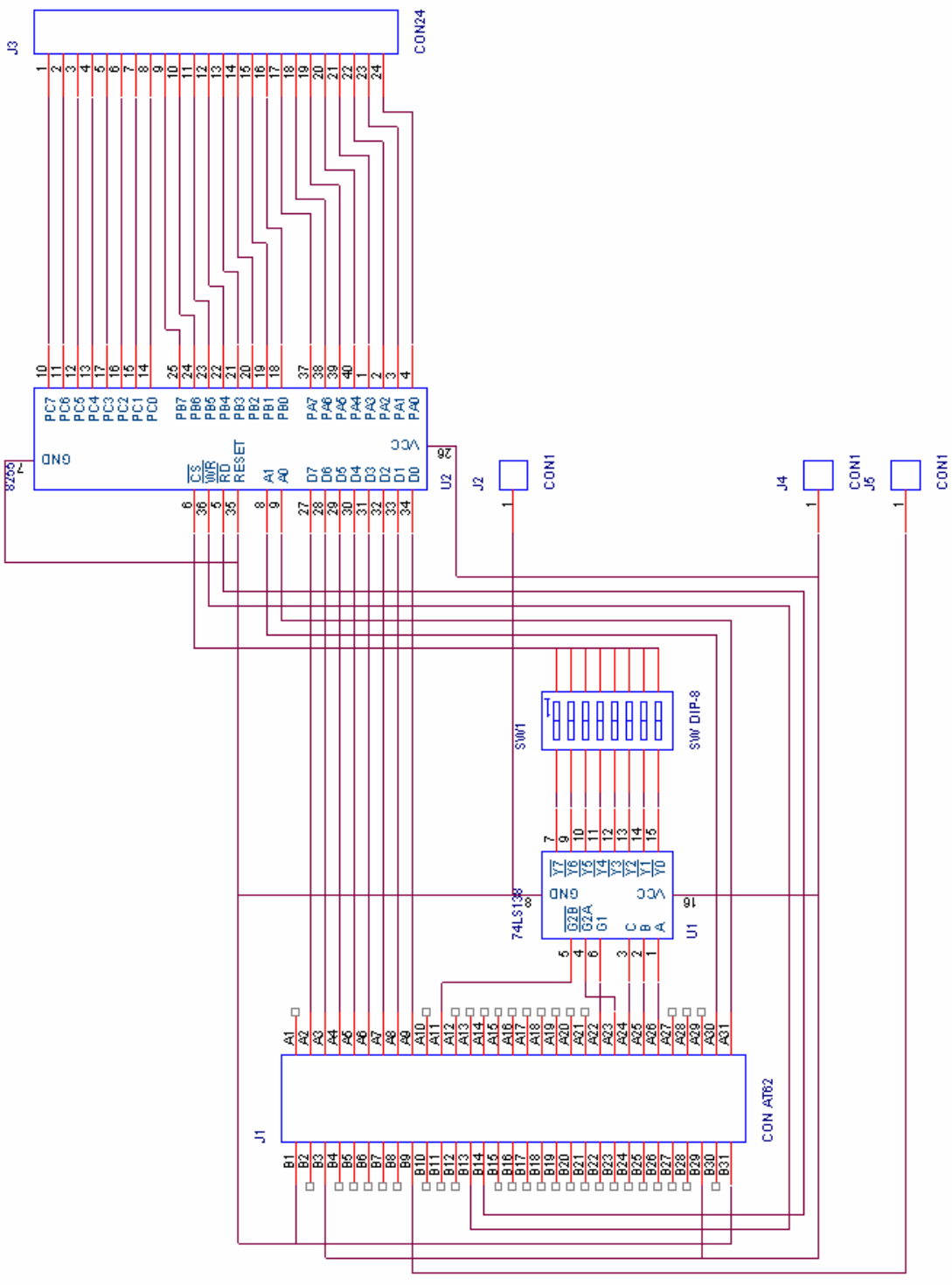


Fig. 3.5 Interfaz de comunicación de entrada y salida

CAPITULO IV

4.1 SOFTWARE DE APLICACIÓN.

El programa se desarrollada en lenguaje C, mediante el cual se programa el PPI que se encuentra en la tarjeta utilizada como interfaz para la adquisición, almacenamiento y manipulación de los datos, así como el control de la entrada y salida de señales para un óptimo control del sistema.

Para una mayor comprensión del programa, este se encuentra seccionado por etapas, las cuales son:

- Adquisición y almacenamiento.
- Lectura y manipulación de los archivos para el reconocimiento.
- Señales de control de los diferentes dispositivos de salida las cuales están dentro del programa principal.

4.1.1. ADQUISICIÓN Y ALMACENAMIENTO.

En cuanto a la adquisición de datos, se realizaron diversas pruebas a través de las cuales se determinó el nivel de la señal de silencio (ruido del medio ambiente), y la señal de voz como se muestra en la figura 4.1, esto con la finalidad de que el software por sí solo pudiera diferenciar entre ellas y así únicamente almacenar los datos correspondientes a la señal de voz, es decir, cualquier señal generada por algún ruido, eco u otra perturbación no pueda activar el sistema, porque no es una señal de voz.

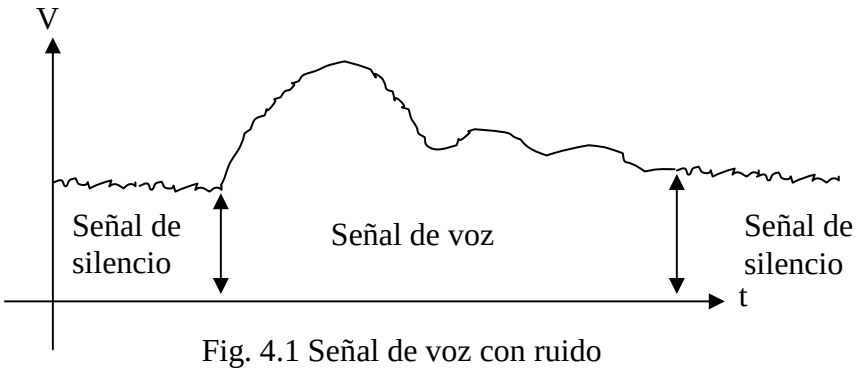


Fig. 4.1 Señal de voz con ruido

Una vez identificada la señal de voz (figura 4.2), se almacena en un archivo; este proceso se realiza diez veces para tener una base de datos y realizar posteriormente una comparación.

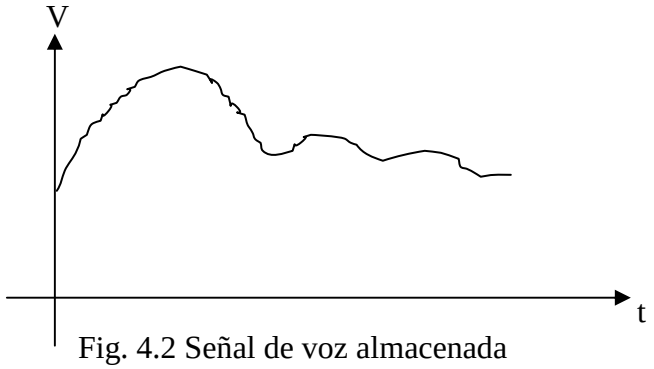


Fig. 4.2 Señal de voz almacenada

4.1.2 LECTURA Y MANIPULACIÓN DE LOS ARCHIVOS PARA EL RECONOCIMIENTO

En esta parte del programa se almacena una señal de voz en un archivo temporal, que permite hacer una comparación con las muestras ya almacenadas, sin embargo esta señal se encuentra desfasada, como se muestra en la figura 4.3.

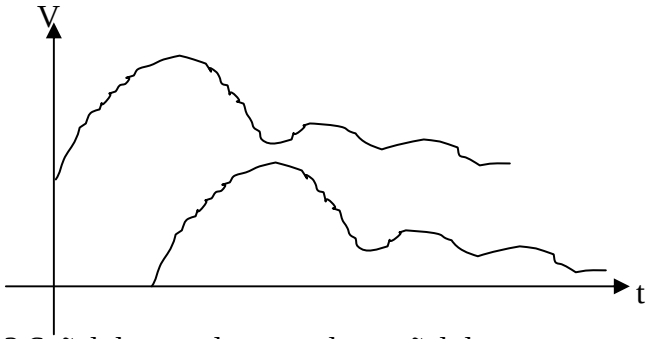


Fig. 4.3 Señal de voz almacenada y señal de voz a comparar

Para una mejor comparación, el programa realiza un desplazamiento de la señal de voz del archivo temporal, con la finalidad de poner en fase las dos señales a comparar (figura 4.4).

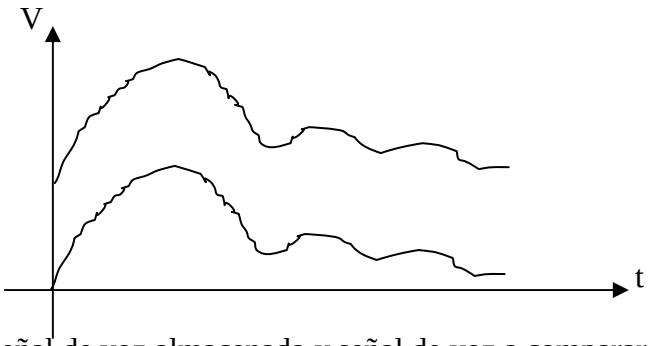


Fig. 4.4 Señal de voz almacenada y señal de voz a comparar en fase

Posteriormente, el programa realiza una lectura byte a byte considerando un barrido el cuál proporciona un margen de tolerancia que considera que las señales son iguales, este proceso es realizado contra las diez muestras de la señal de voz, con el objetivo de tener una mayor precisión respecto al hablante.

4.1.3 SEÑALES DE CONTROL DE SALIDA

Las señales de control son enviadas a los puertos de salida en función de la instrucción dada por el usuario; para ejemplificar, en el proyecto se hace girar un motor de CD mediante una etapa de potencia que se describirá en el siguiente capítulo. El programa realiza mediante los comandos de voz las siguientes instrucciones (tabla 4.1).

PALABRA	INSTRUCCIÓN
HOLA	Activación del sistema
DOS	Giro de motor izquierda
NUEVE	Giro de motor derecha
HOLA	Velocidad rápida
A	Velocidad lenta
B	Parar motor

Tabla 4.1 Instrucciones de voz

Las figuras 4.5 y 4.6 representan muestras de la palabra "HOLA" y "DOS" respectivamente, de la forma en que quedaron almacenadas para así poder llevar a cabo la comparación y reconocimiento, la primera identifica el usuario y la segunda gira el motor a la izquierda.

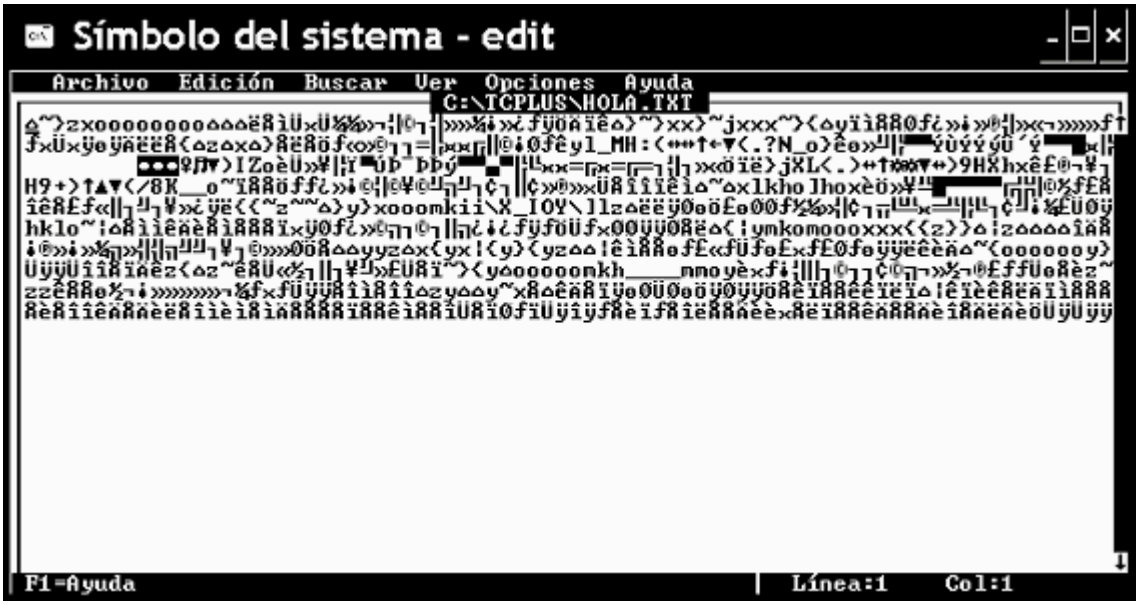


Fig. 4.5. Representación de la palabra "HOLA"

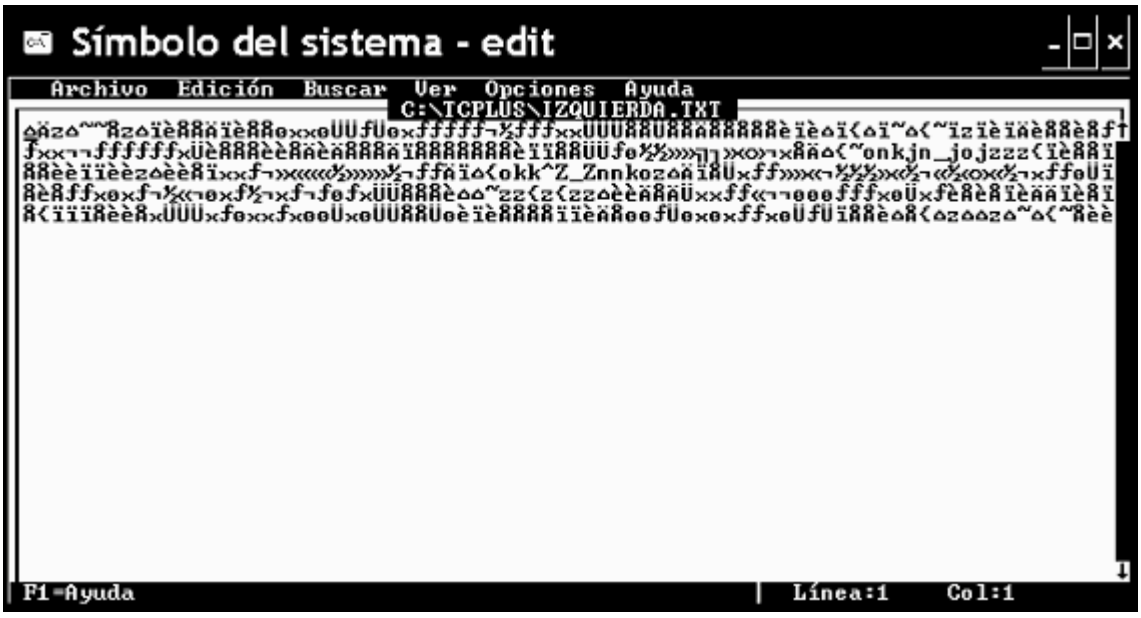


Fig. 4.6 Representación de la palabra "DOS"

4.2 PROCESO DEL SOFTWARE

Para iniciar el software se muestra un diagrama de flujo (Figura 4.7) que representa las características mas relevantes en su ejecución.

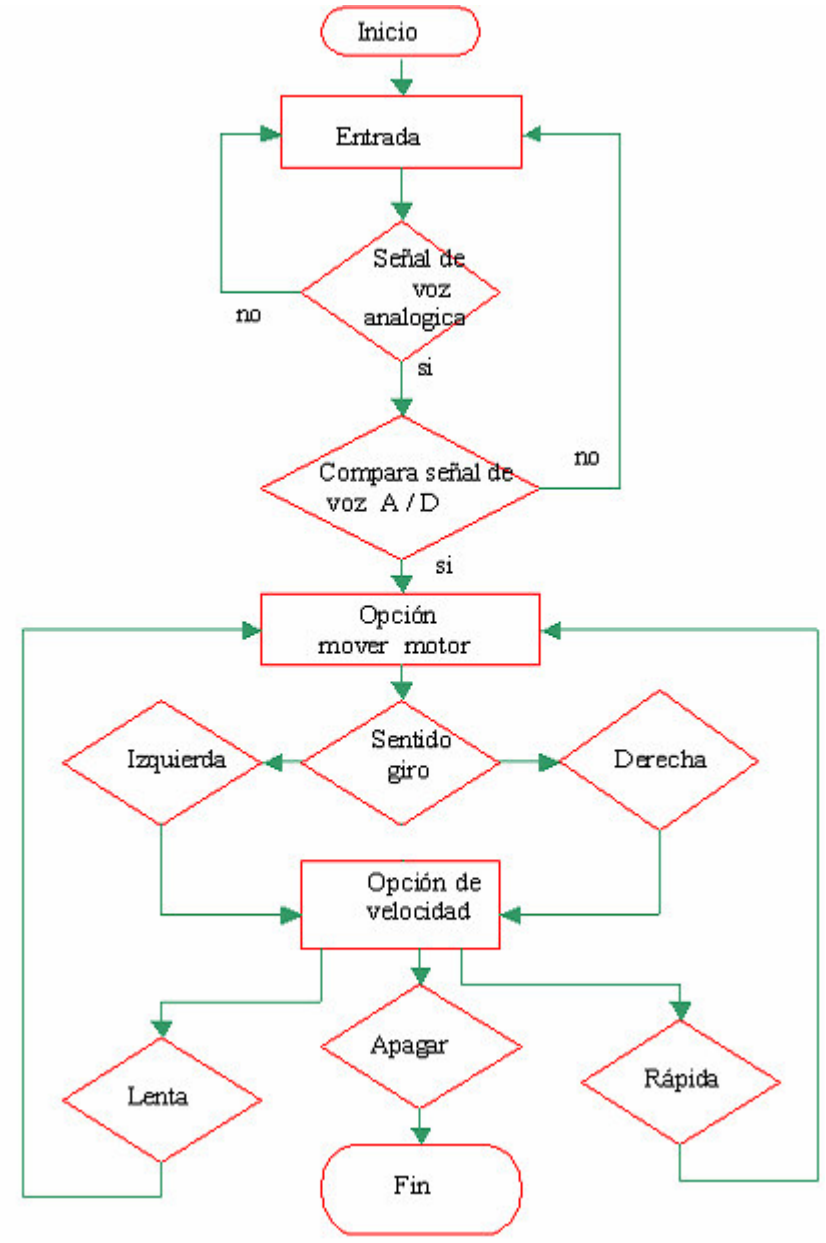


Figura 4.7 Diagrama de flujo

El diagrama de la figura 4.8, muestra más claramente el proceso que sigue el programa. El sistema permanece en espera de una señal de voz, la ingresa en formato digital para crear un archivo temporal que permitirá compararlo con los archivos previamente almacenados de la señal de voz que corresponde a un único usuario, esta comparación servirá para identificar la señal de voz correcta, es entonces cuando el sistema activa las señales de control hacia los puertos para que finalmente realice las tareas correspondientes, en este caso el movimiento del motor de CD.

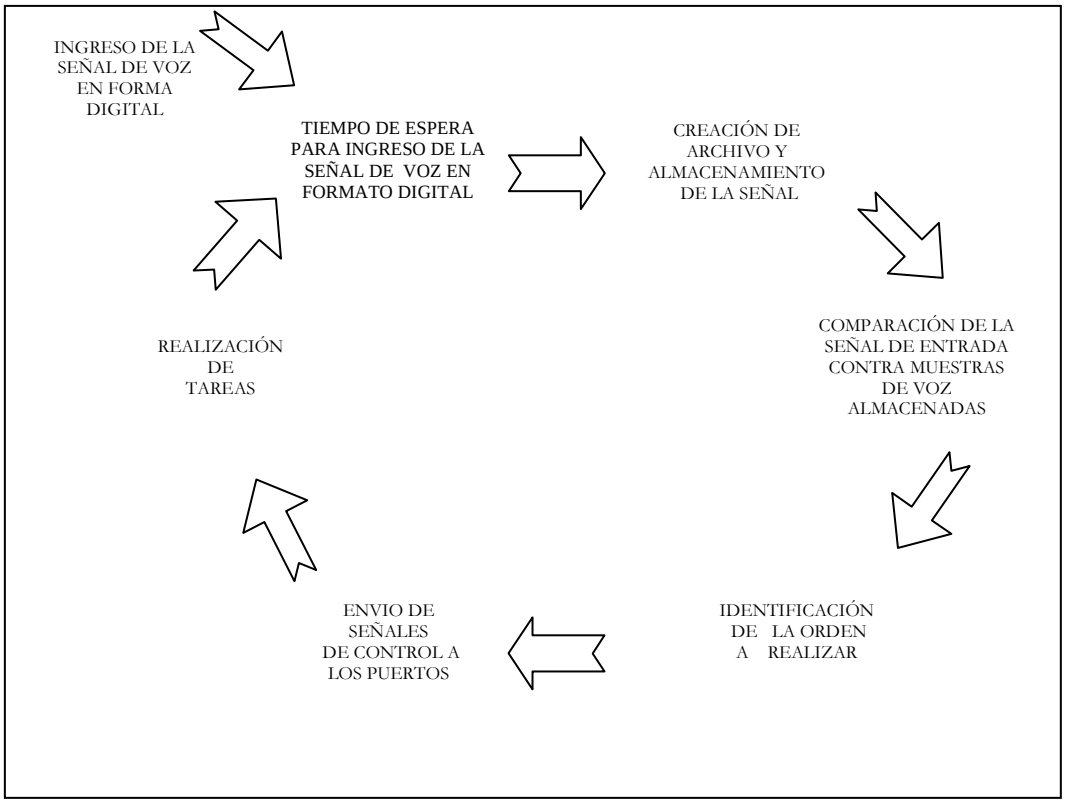


Fig.4.8 Proceso del software.

A continuación se describe el proceso de control del programa:

El programa principal es el encargado de realizar la programación del PPI y llamar a las funciones o subrutinas encargadas del proceso de control del sistema.

A continuación se muestra el código del programa principal

```
void main (void)
{   outportb (puerto_ctrl, palabra_ctrl);
    outportb (puerto_b, 0xFF);
    clrscr();
    printf("Hola\n");
    printf("este programa tiene como finalidad\n");
    printf("accesar al sistema por medio del reconocimiento de voz\n");
    printf("ademas te da la opcion de mover un motor de CD\n");
    printf("despues de que se quite esta pantalla el CPU esta listo para recibir tu orden\n");
    printf("despues de que presiones una tecla\n");
    printf("di    ' HOLA'    por favor\n");
  while (t<8)
    { t=0;
       clrscr ();
       escribe();          */subrutina de adquisición y almacenamiento/*
       compara();          */subrutina de comparación/*
        }   if (t>=8)    {        clrscr();
       printf("Hola\n");
       printf("ahora que quieres hacer\n");
       printf("decir DOS gira el motor a la derecha\n");
       printf("decir NUEVE gira el motor a la izquierda\n");
       printf("despues de que se quite esta pantalla el CPU esta listo para recibir tu orden\n");
       t=0;
       while(t!=1) {      clrscr();
                          printf("quieres cambiar el sentido de giro\n");
                           u=0;  v=0;
          while(u<3&&v<3){   clrscr();
                                escribe();
                          motori();         */subrutina de comparación para sentido de giro/*
```

```
                    motord();
                    }if (u>=3){
                    printf("gira motor izquierda");
        outportb(puerto_b,izquierda);          // orden de salida de datos de control
                    velocidad();    //subrutina de velocidad
                }else if(v>=3) {
                    printf("gira motor derecha\n");
        outportb(puerto_b,derecha);            //orden  de salida de datos de control
                    velocidad();
}}}}
```

La subrutina de adquisición y almacenamiento, realiza la sincronización de la tarjeta de adquisición de datos mediante el PPI con la PC, posteriormente determina si la señal entrante es voz o silencio (ruido ambiental). Mientras exista silencio el programa entra a un ciclo de espera, en el instante en que detecta que la señal entrante es voz, sale de este ciclo de espera, ingresa los datos a la PC y los guarda en un archivo temporal. Esta rutina se realiza cada vez que se le da una instrucción a la PC.

Rutina para almacenar los datos en un archivo temporal

```
void escribe(){  unsigned char putc;
   char sens;                    //declara el sensor de silencios
   char sen;
   FILE *muestra1;
   muestra1 = fopen ("71.txt","wb+");    //abre archivo para escritura
   clrscr();                     //limpia pantalla
   conteo=0;
   while (!kbhit())    {  }
   sens= inportb(puerto_a);          //control de silencios
   while (sens >= '„' && sens<= '¥') {  sens= inportb(puerto_a);    }
```

```
 while (conteo <=200)                    //ciclo para ingreso de muestras
       {    sensor= inportb(puerto_c);              //checa si hay dato
    check = sensor & 1;
          while (check == 1) {   dato = inportb (puerto_a);
       if(dato!=0x1a)
                  {  fputc (dato,muestra1);          //escribe datos
                  while (check == 1)
                        {sensor = inportb (puerto_c);
                           check = sensor & 1;}
                  sen= inportb(puerto_a);
                  if (sen >= '‡' && sen<= '¥')
                        {  conteo = conteo + 1;
                           if (conteo==199){    dato='$';
                                    fputc(dato,muestra1);
                                    dato='$';
                                    fputc(dato,muestra1);
                                    dato='$';
                                    fputc(dato,muestra1);
                        }}
                  else  { conteo = 0;
} } } }
fclose(muestra1);                        //cierra archivo
return;
}
```

La subrutina de lectura y manipulación de archivos, realiza la apertura de dos archivos; un archivo temporal y uno de los diez de la base de datos, realiza una lectura byte a byte de ambos archivos y los compara para determinar el grado de coincidencia, en base a la comparación determina la igualdad de los archivos, posteriormente cierra el archivo de la base de datos y abre el siguiente para realizar este mismo procedimiento, y así sucesivamente hasta comparar el ultimo archivo.

En el momento que la comparación determina que hay coincidencia ejecuta la orden indicada, en caso contrario el programa regresa a la rutina de adquisición y almacenamiento para brindar al usuario nuevamente la oportunidad de ingresar la orden por medio de la voz. Este proceso se realiza cada que el programa recibe un de comando voz.

Rutina para comparación de archivo.

```
void compara(){  limpia =1;
   limpia1 =1;
        clrscr();
        if (((muestra= fopen("1.txt", "r"))==NULL) || ((muestra2=
fopen("71.txt","r"))==NULL))
        {       perror ("archivo nulo");
                exit (1);}
        resultado= &muestra;
        while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) &&
!feof(muestra2)))
        {      while(limpia!=0)   {    limpia=fflush(muestra);       }
      while(limpia1!=0)   { limpia1=fflush(muestra2);     }
      buffer [i]=c;
                c=fgetc (muestra);
                i++;
                buffer2 [j]=d;
                d=fgetc (muestra2);
           j++;
                di1=d+1;di2=d+2;df1=d-1;df2=d-2;
                while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
      {        buffer2 [j]=d;
                        d=fgetc (muestra2);
```

```
                j++;     }
          while (s<=3 || b<=3)
          {buffer [i]=c;
                c=fgetc (muestra);
                i++;
                if (c== '$'){s++;}
                else{   s=0;}
                if (s==3){buffer [i--]= '\0';
          if (ferror (muestra))
               perror ("error");
               buffer2 [j--]= '\0';
          if (ferror (muestra2))
               perror ("error");
               fclose (muestra);
               fclose (muestra2);
               printf("%d,%d\n",M,N);
          if (M>=1200) {   printf("si\n");
       t++; }
          else  {     printf("no\n");    }
    printf("%d\n",limpia);
    printf("%d\n",limpia1);
                return;}
else   {        buffer2 [j]=d;
                d=fgetc (muestra2);
                j++;
                if (d== '$')   {    b++;    }
                else   {    b=0;    }
                if (b==3)    {   buffer [i--]= '\0';
          if (ferror (muestra))
               perror ("error");
```

```
                buffer2 [j--]= '\0';
            if (ferror (muestra2))
                perror ("error");
                fclose (muestra);
                fclose (muestra2);
                printf("%d,%d\n",M,N);
            if (M>=1200)   {    printf("si\n");
                                t++;    }
                        else    {   printf("no\n");      }
        printf("%d\n",limpia);
        printf("%d\n",limpia1);
                    return;}
      else  {    rango();
} } } }
  return;
  }
```

El programa completo se muestra en el apéndice B.

Una vez integradas las etapas de adquisición y de potencia mediante el software de control se obtiene el funcionamiento completo del sistema, que resumiendo consta de adquisición de voz, su reconocimiento y ejecución de tareas especificadas por el usuario.

CAPITULO V

5.1 FUNCIONAMIENTO.

La etapa final del proyecto es la aplicación del digitalizador de voz, esta consiste particularmente en ejecutar órdenes de voz que ingresan a la PC y que por medio de la interfaz de potencia se controla el movimiento de un motor en sentido de giro izquierda o derecha, y con la opción de variar la velocidad, lenta ó rápida.

Al inicializar el programa, éste ejecuta una rutina la cual muestra la pantalla de la figura 5.1, que al mismo tiempo, da la bienvenida e indica al usuario que es un sistema basado en voz para el control de un motor de CD.

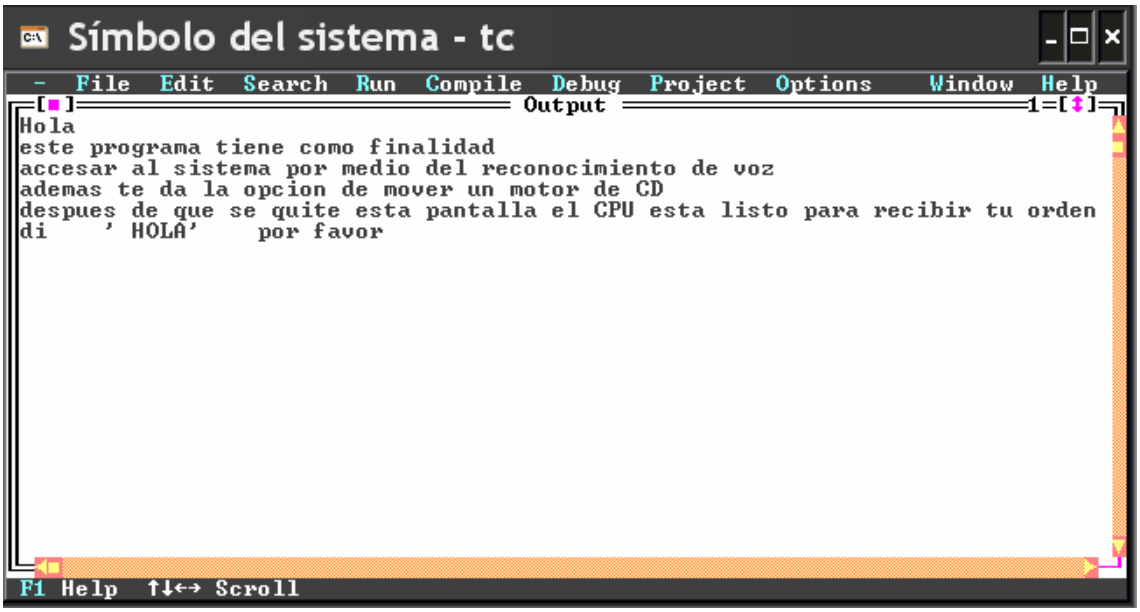


Fig. 5.1 Pantalla de inicio

Al momento de quitarse la pantalla el programa permanece en espera de la palabra y persona adecuada, mientras esto no suceda el programa permanece en ese ciclo de espera, como ya fue descrito en el capitulo anterior, al momento en que el software reconoce al hablante presenta la siguiente pantalla (Figura 5.2):

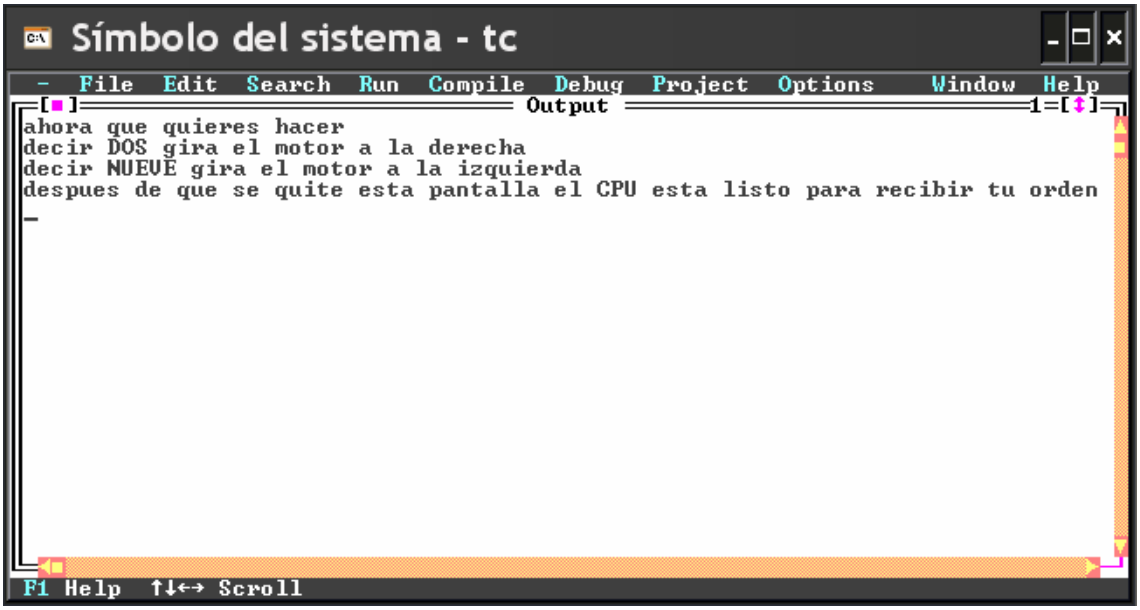


Fig. 5.2 Opciones de sentido de giro

En este punto el software espera una orden ya sea dos o nueve, éste tiene la capacidad de determinar por medio de el proceso de comparación anteriormente descrito cual es la instrucción deseada, y en función de dicha comparación emite una de las siguientes dos ventanas(Figuras 5.3a y 5.3b).

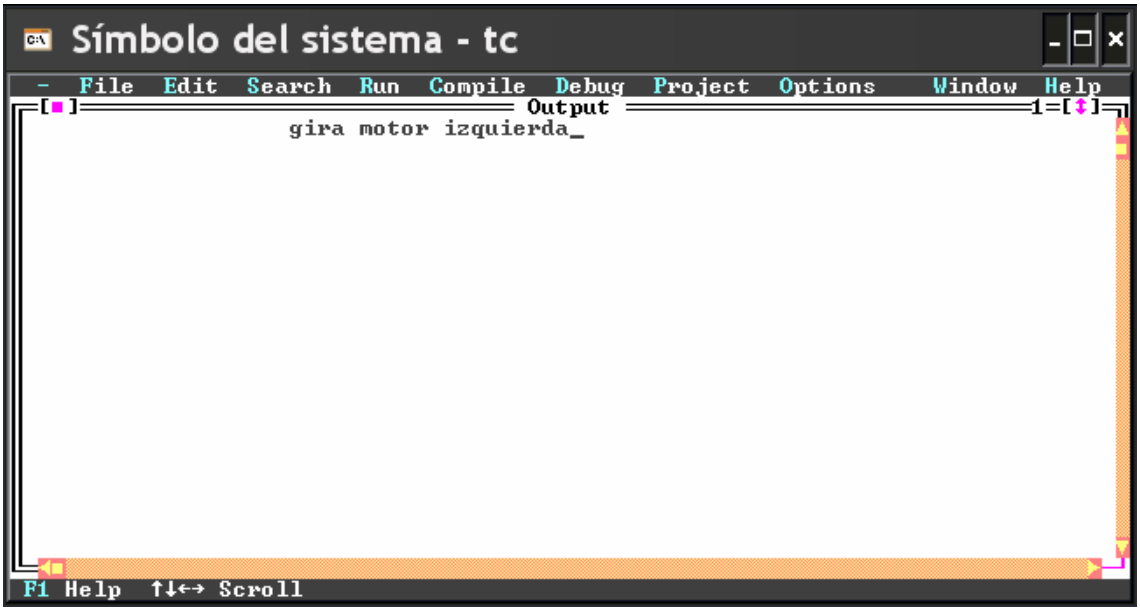


Fig. 5.3a Sentido de giro izquierda

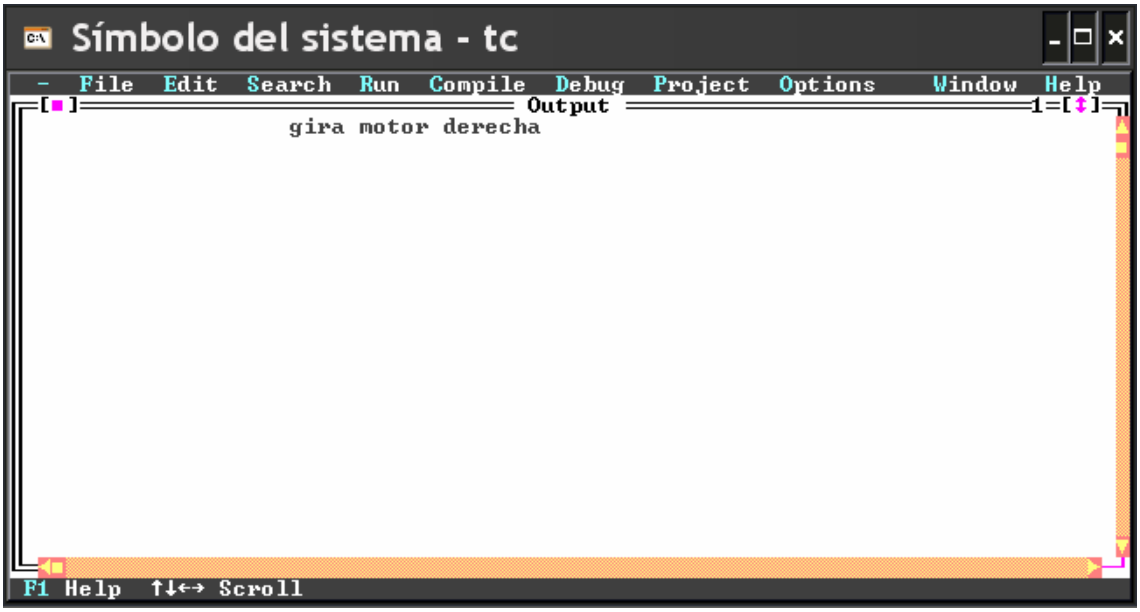


Fig. 5.3b Sentido de giro derecha.

Al mismo tiempo que presenta esta pantalla el programa envía dos bits por uno de los puertos de salida (Puerto B), conectado directamente a la etapa de potencia. Para esta etapa, se diseño la tarjeta que esta construida con los siguientes dispositivos electrónicos:
Opto-acopladores, DAC, amplificadores operacionales, transistores y un driver para controlar un motor de CD.

5.2 CONSTRUCCION DE LA ETAPA DE POTENCIA.

Se utilizan opto-acopladores, ya que son dispositivos de especial interés en las aplicaciones que requieren voltajes, corrientes o potencias diferentes entre la lógica digital y los elementos a controlar. Los opto-acopladores excitan al componente de salida (habitualmente un foto transistor) con la luz emitida por el LED en la entrada, de manera que conectando los componentes externos en forma adecuada, se obtiene el mismo resultado que si se utilizara un transistor común (con la ventaja mencionada en el capitulo I de aislamiento eléctrico entre etapas).

Se utiliza el C.I. PC817, que contiene internamente opto-acopladores para protección de la PC y mantener la corriente en niveles adecuados de tal manera que no se tengan perdidas en las señales de salida debidas al consumo de corriente al acoplar la etapa de potencia.

Para controlar el sentido de giro del motor de corriente directa se utiliza el C.I. L293B que internamente contiene un puente H, por medio del cual se determina el sentido de giro del motor (izquierda ó derecha). Este arreglo consiste básicamente de cuatro transistores acomodados de la siguiente manera:

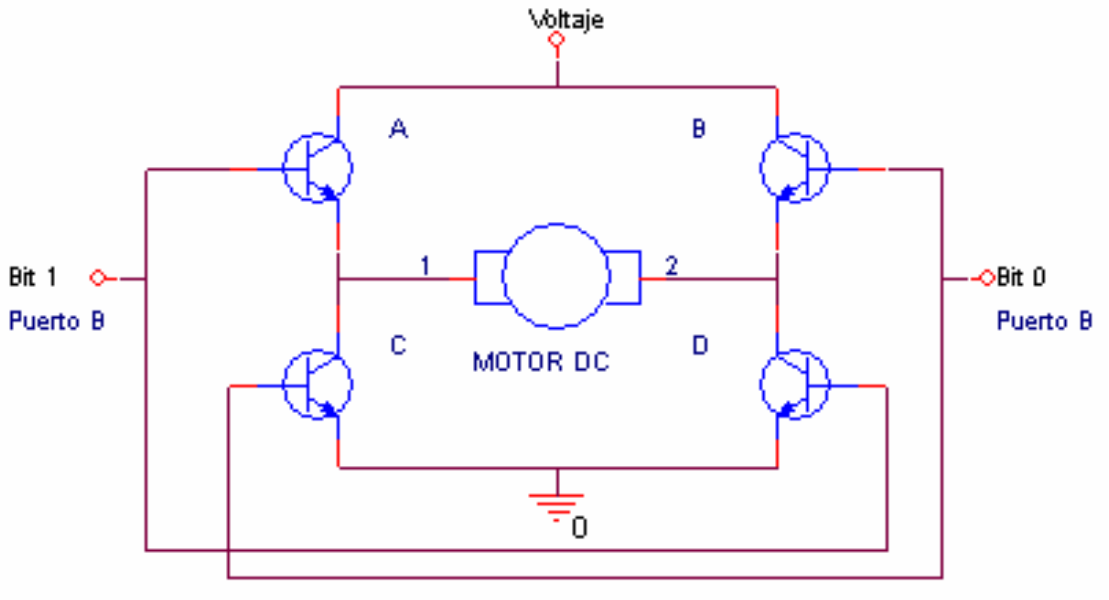


Fig. 5.4 Puente H

Los transistores (A, B, C y D) se conectan como se muestra en la figura 5.4 y funcionan de la siguiente manera:

Si se cierran solamente los contactos A y D la corriente circulará en un sentido a través del motor y si se cierran solamente los contactos B y C la corriente circulará en sentido contrario.

Continuando con el funcionamiento del sistema, el programa después de haber enviado estos dos bits por el puerto B muestra las siguientes pantallas (figura 5.5).

Fig. 5.5 Opciones de velocidad.

Con las cuales indican que esta listo para recibir la orden de velocidad, además de mostrar las opciones tanto para velocidad lenta o rápida como para apagar el sistema, al quitarse esta pantalla el sistema esta listo para recibir una orden y así determinar, dependiendo de la instrucción, cual es la operación a realizar, si el software entiende lento o rápido muestra una de las dos pantallas siguientes respectivamente (figura 5.6).

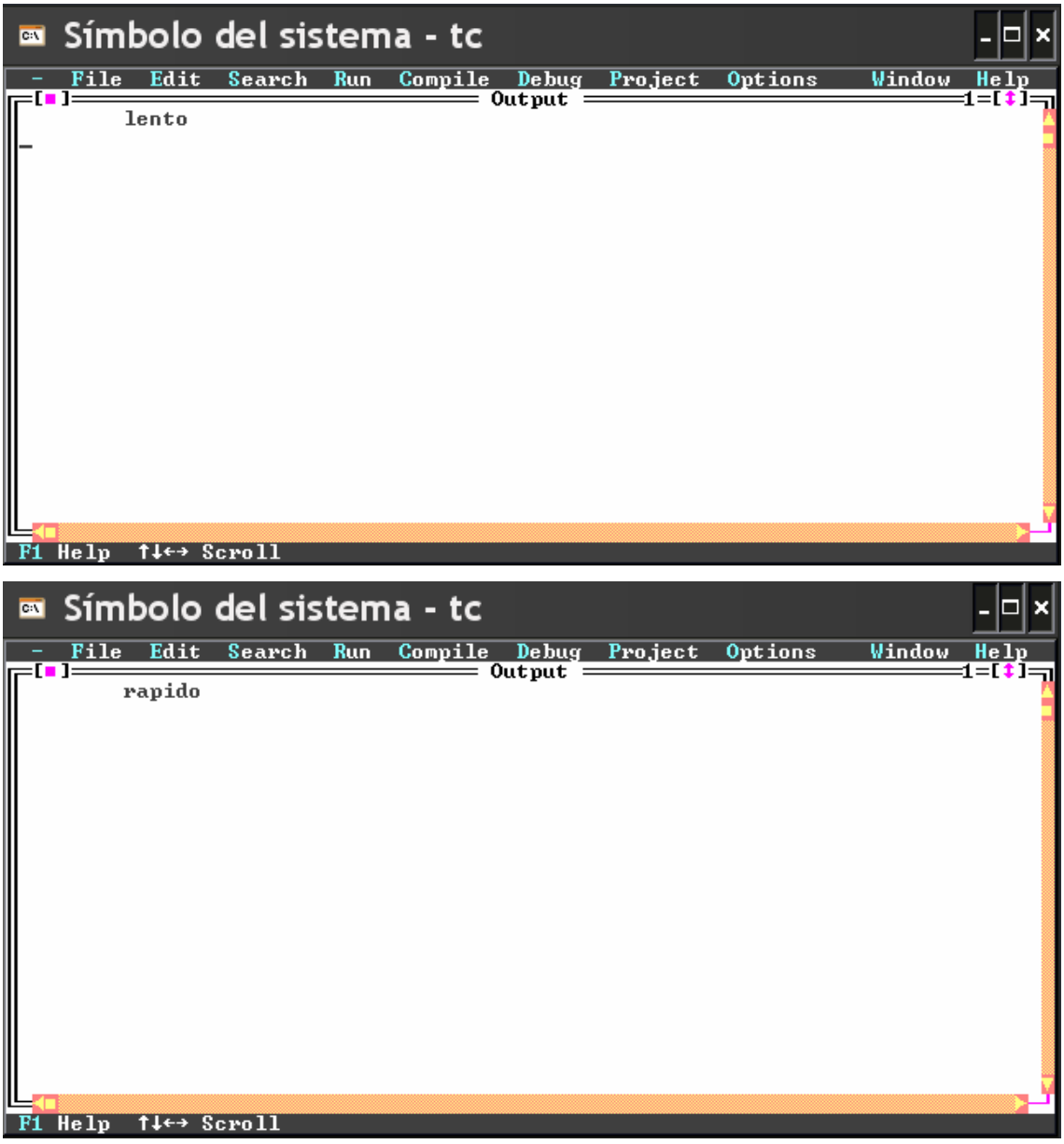


Fig. 5.6 Velocidad del motor

Con una de estas dos opciones seleccionadas, el sistema envía por medio del puerto paralelo una señal de 8 bits, la cual determina la velocidad a la cual se quiere que gire el motor. El control de velocidad se realiza mediante una tarjeta que recibe los datos del puerto paralelo, la cual esta constituida por un DAC, amplificadores operacionales y un transistor principalmente (figura 5.7), con la finalidad de proporcionar un mejor control del motor.

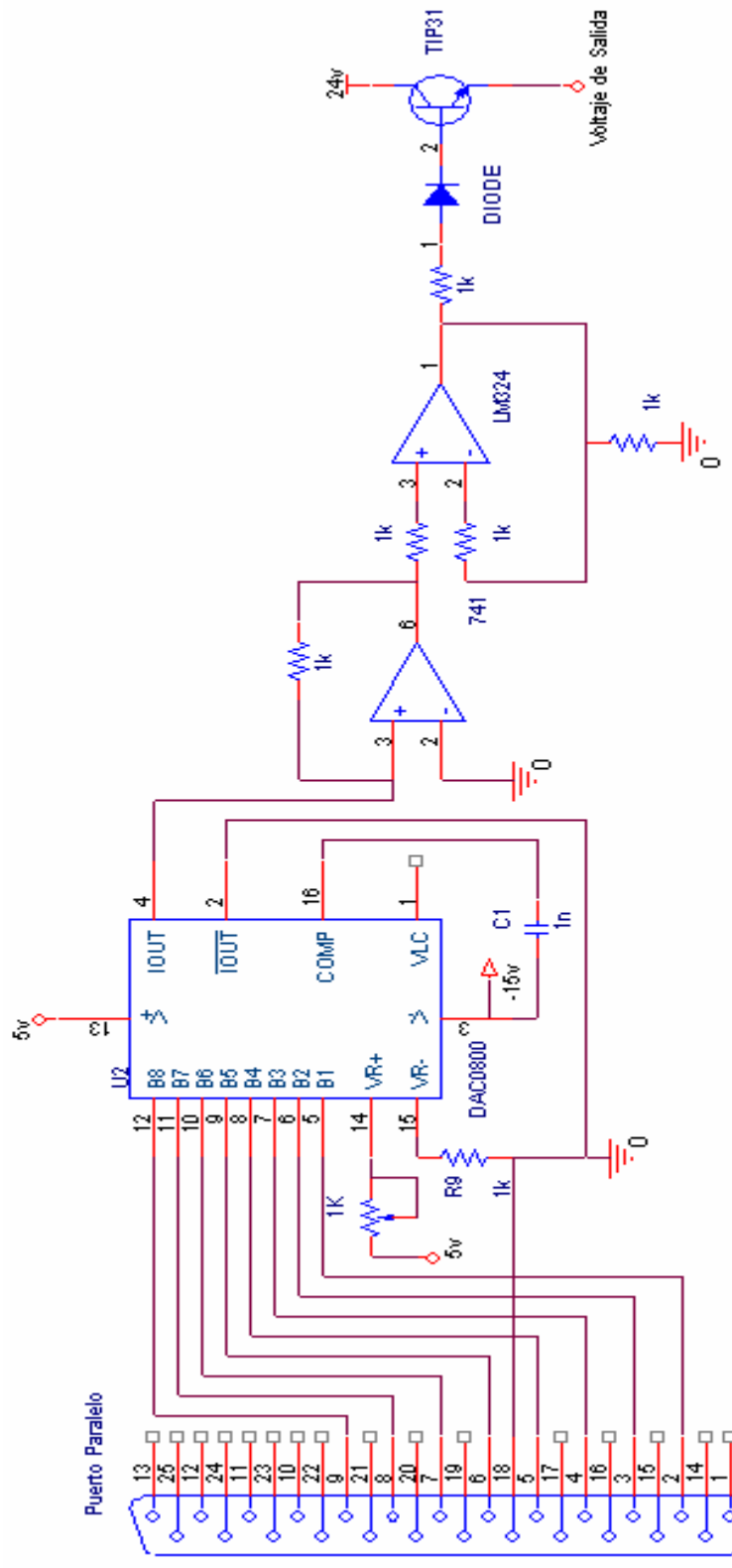


Fig. 5.2 Interfaz de potencia para el control de la velocidad

Ésta tarjeta permite que el motor tenga la velocidad y el sentido de giro deseado, posteriormente el programa muestra la siguiente ventana (figura 5.8) que permite tener la opción de cambiar el sentido de giro y el proceso se repite.

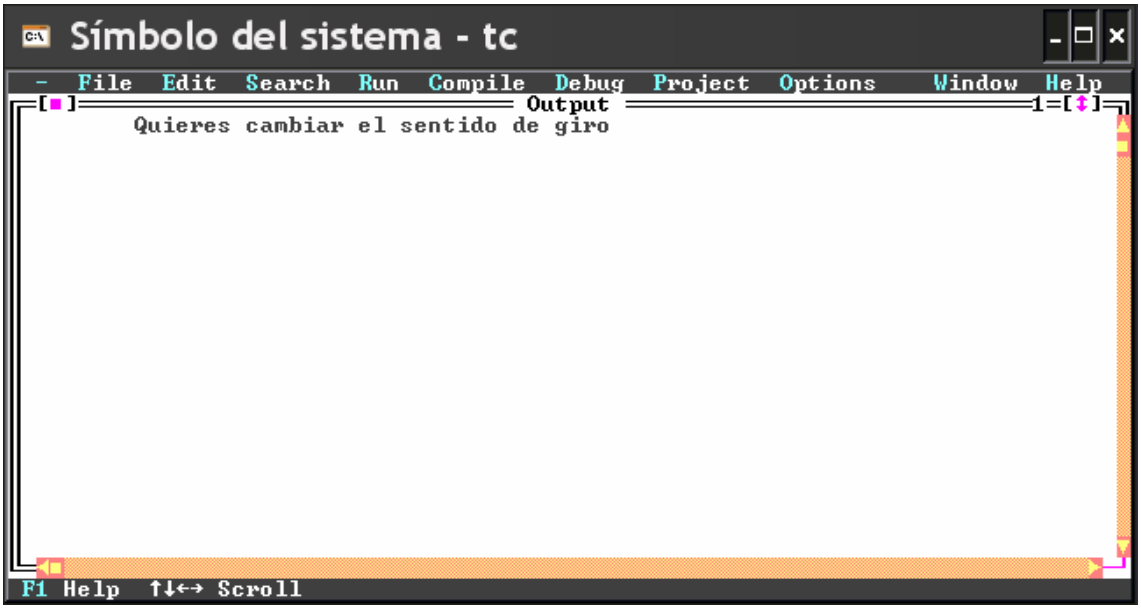


Fig. 5.8 Opción para cambio de sentido de giro

Solo si se elige la opción apagar el programa muestra la siguiente pantalla (figura 5.9).

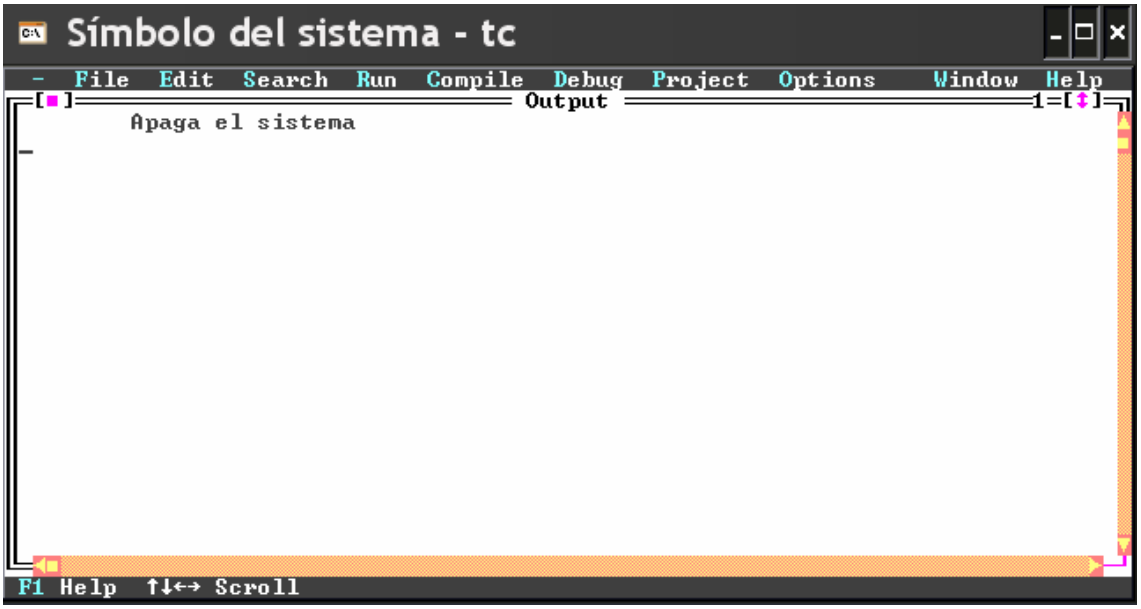


Fig. 5.9 Fin del programa

Y finaliza el programa.

La interfaz de salida tiene la posibilidad de adoptar otras señales para ser ejecutadas a través del software, tales como el control de un sistema de iluminación con control de intensidad, sin embargo en este caso se dejan como opción para la ampliación del proyecto.

La figura 5.10 muestra la etapa de potencia, como se puede observar solo se utilizan dos bits del puerto B del PPI, quedando libres seis señales de salida para los dispositivos que se quieran integrar al sistema, además de ilustrar que el sistema no obstruye los demás componentes periféricos de la PC, es decir si así se quisiera, mediante la repetición ó modificación de algunas rutinas en el software siguiendo el mismo proceso se pueden controlar más dispositivos.

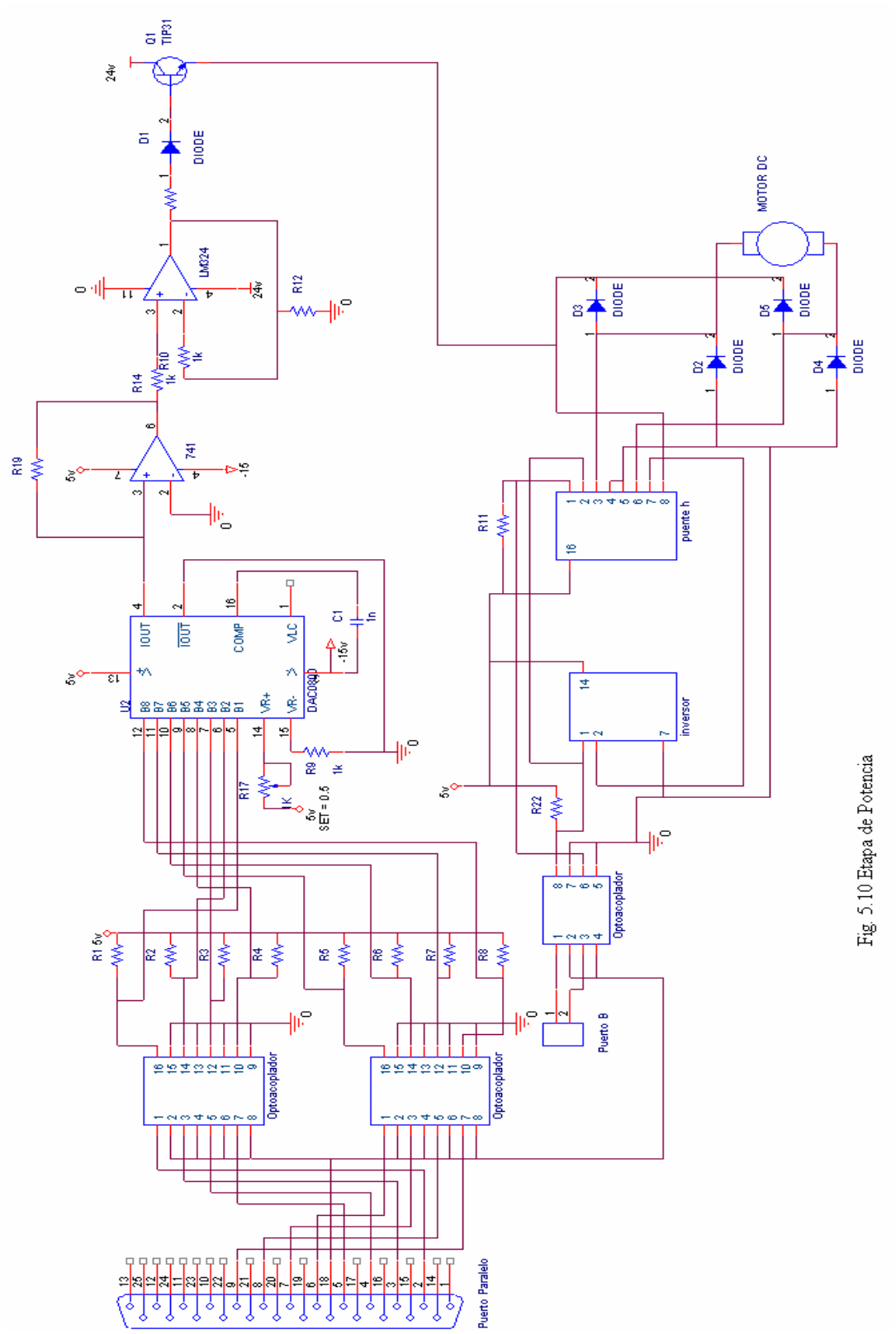


Fig. 5.10 Etapa de Potencia

Todas la pruebas necesarias para la construcción del digitalizador de voz se realizaron en el laboratorio de electronica L 912 iniciando con pruebas de simulación en el software ORCAD versión 9.2 posteriormente se experimento en las tabletas protoboard (figura A.) hasta realizar todas las placas de circuito impreso mostradas en los capítulos anteriores.

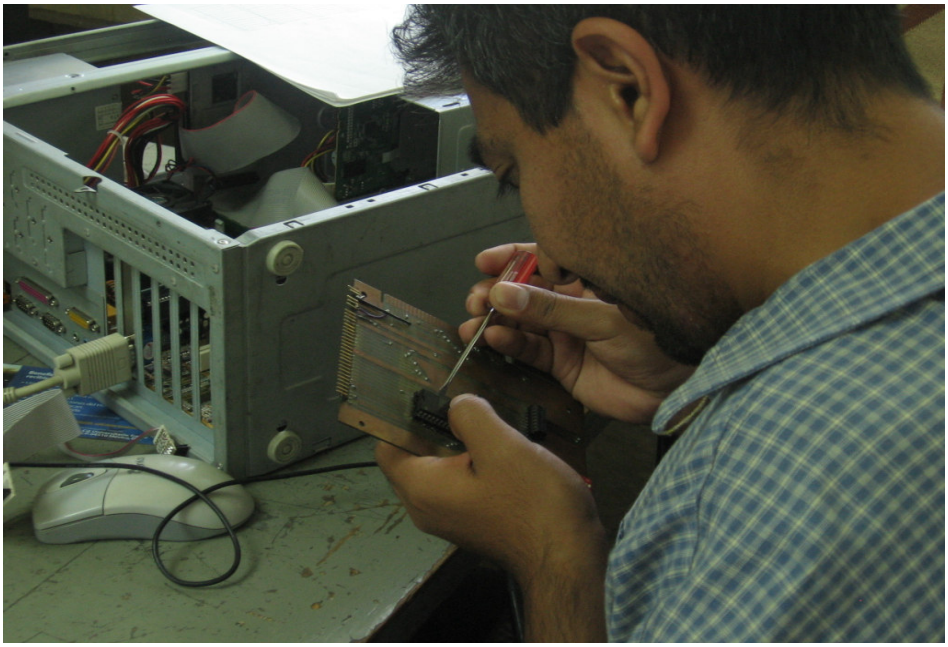


Fig. A Pruebas realizadas en el laboratorio de electrónica

Cabe mencionar que durante las pruebas se modificaron varias veces la integración de los componentes debido a los requerimientos que se iban presentado en el desarrollo del sistema, por mencionar algunos los voltajes de referencia fueron ajustados en varias ocasiones, se hicieron cambios en el convertidor analógico digital (CAD) del cual se tuvo que hacer una selección que se adecuara con el sistema, ya que en el mercado existen diferentes convertidores, pero cada uno esta diseñado para diversas aplicaciones.

En la actualidad existen diferentes métodos para desarrollar sistemas de adquisición de voz para emplearlos en sistemas de alta tecnología, de hecho en el mercado se pueden encontrar

tarjetas que realizan la función de interfaz entre el usuario y la PC, sin embargo son de un costo muy elevado.

Dos razones importantes son las que motivaron la realización de este proyecto:

1. Aplicar parte de los conocimientos adquiridos a lo largo de la carrera de ingeniería, especialmente la teoría y práctica de las materias de electrónica.
2. Emplear dispositivos que reduzcan el costo del sistema.

De lo primero, realizar el proyecto aplicando los conceptos de materias básicas: diseño lógico, amplificación de señales, microcontroladores, programación y electrónica analógica entre otras. Tal vez este sistema en un tiempo sea obsoleto, sin embargo pueden utilizarse otras alternativas para realizar las interfaces, ya sea por medio de alguna tarjeta que incluya algún dispositivo lógico programable o en cuestión de conexión a través del puerto USB.

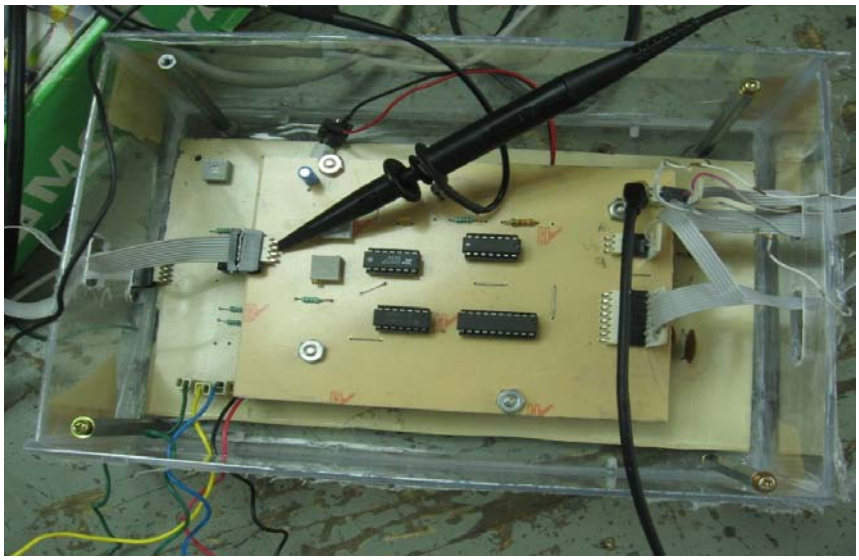


Fig. B. Digitalizador de voz.

De lo segundo, todos los dispositivos empleados en este trabajo de tesis son los que respondieron bien a las necesidades del digitalizador (Figura B)y que de alguna manera no son muy costosos.

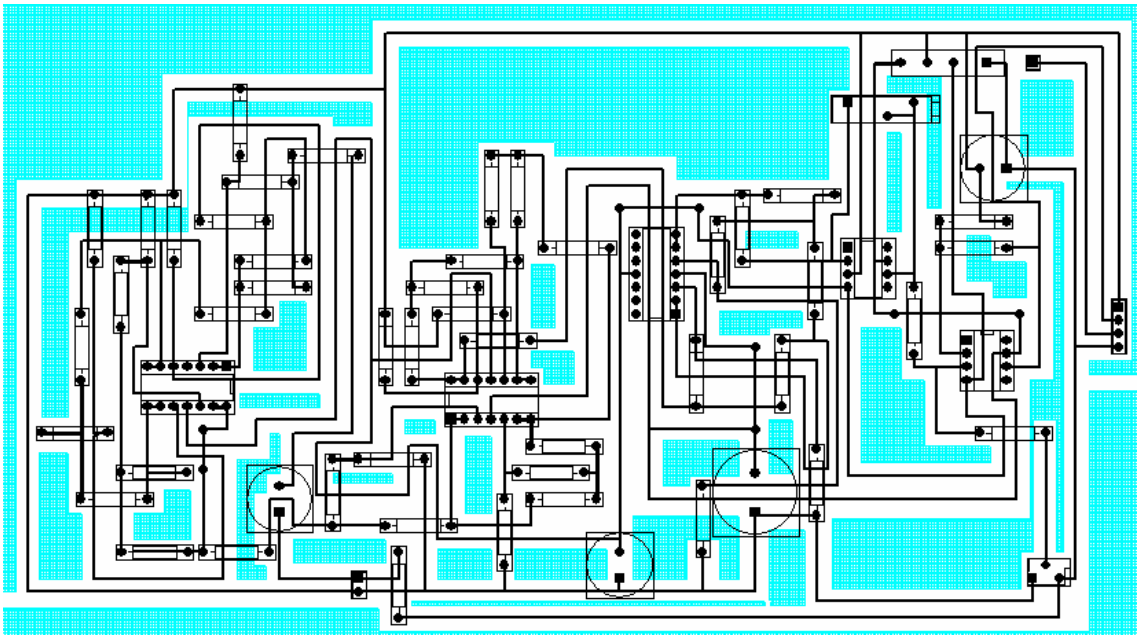
APÉNDICE A

Este Apéndice contiene los negativos de los circuitos impresos de las interfaces de la etapa de entrada, etapa de control y etapa de salida así como los diagramas electrónicos y los componentes que lo integran.

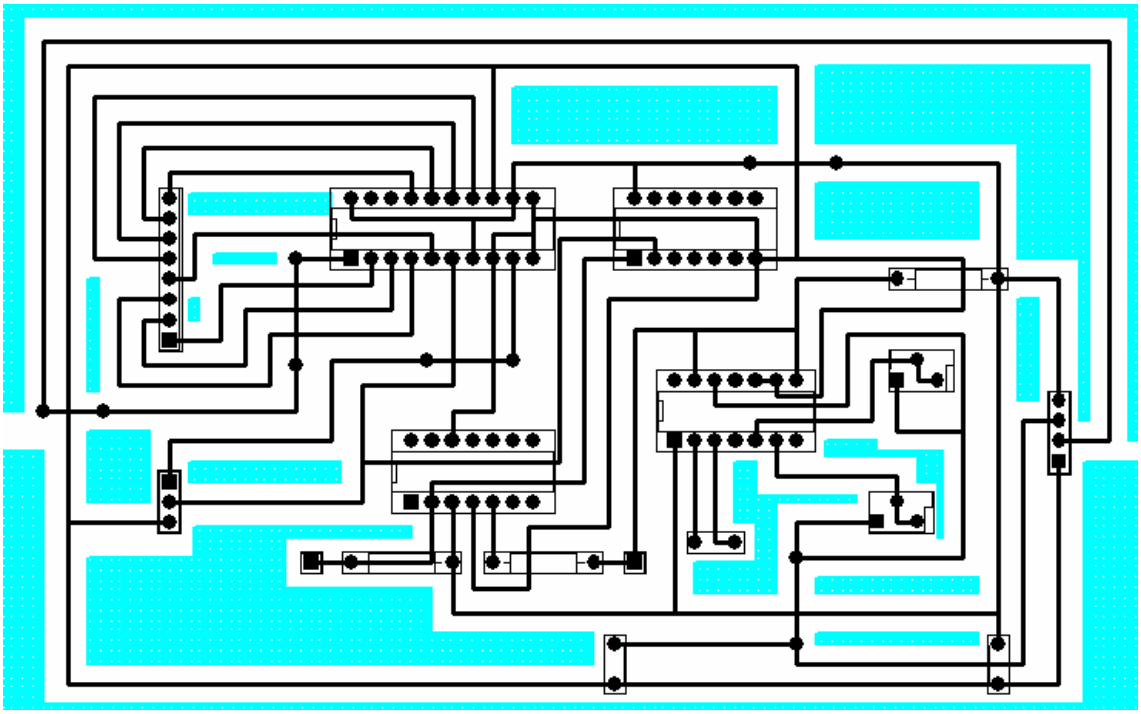
Para la etapa de entrada, que es la etapa de adquisición y digitalización de voz, se utilizan voltajes de +12V d.C., -12 V d.C., 5 V d.C. y tierra.

En la etapa intermedia, que es la etapa de control por medio de software, se utiliza la tierra interna de la PC

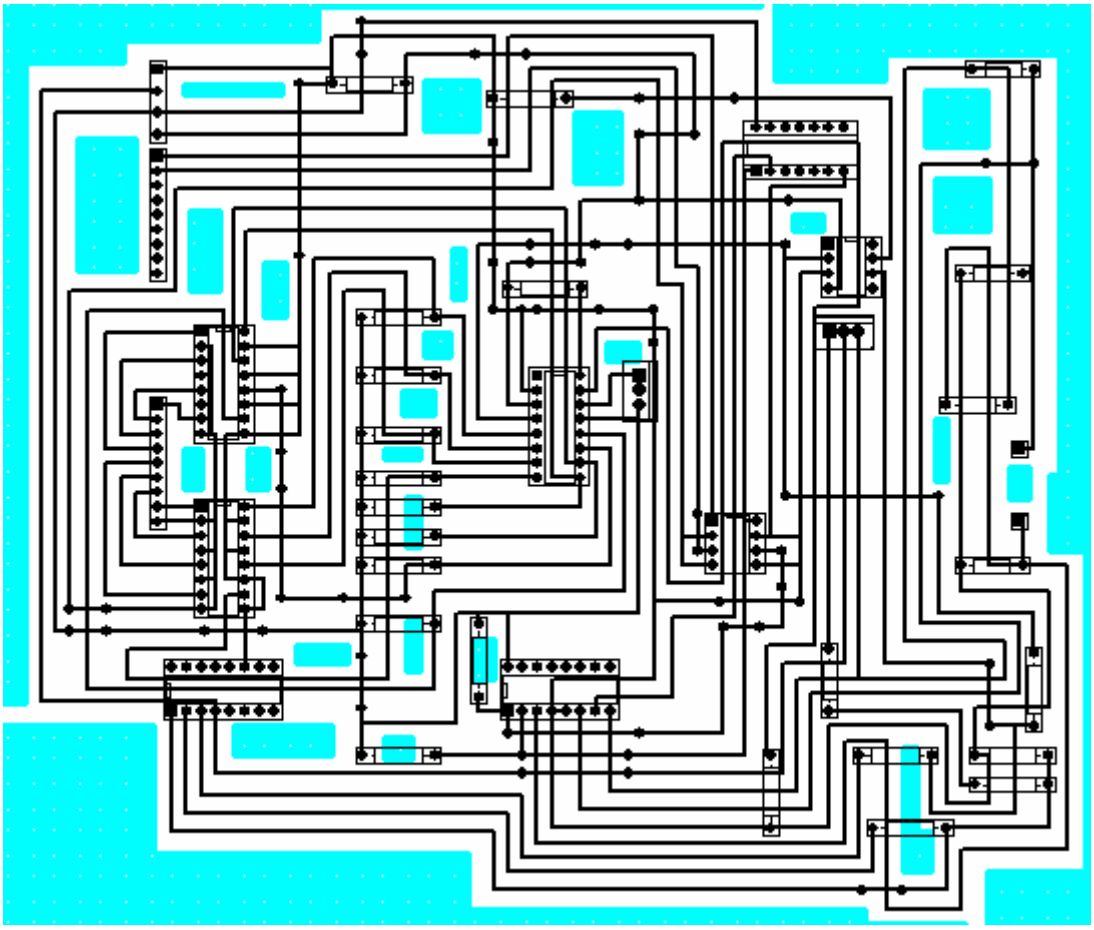
En la etapa de salida, que es la etapa de potencia se utiliza un voltaje de 24V.d.C. además de los voltajes analógicos de +12Vc.d., -12Vcd.



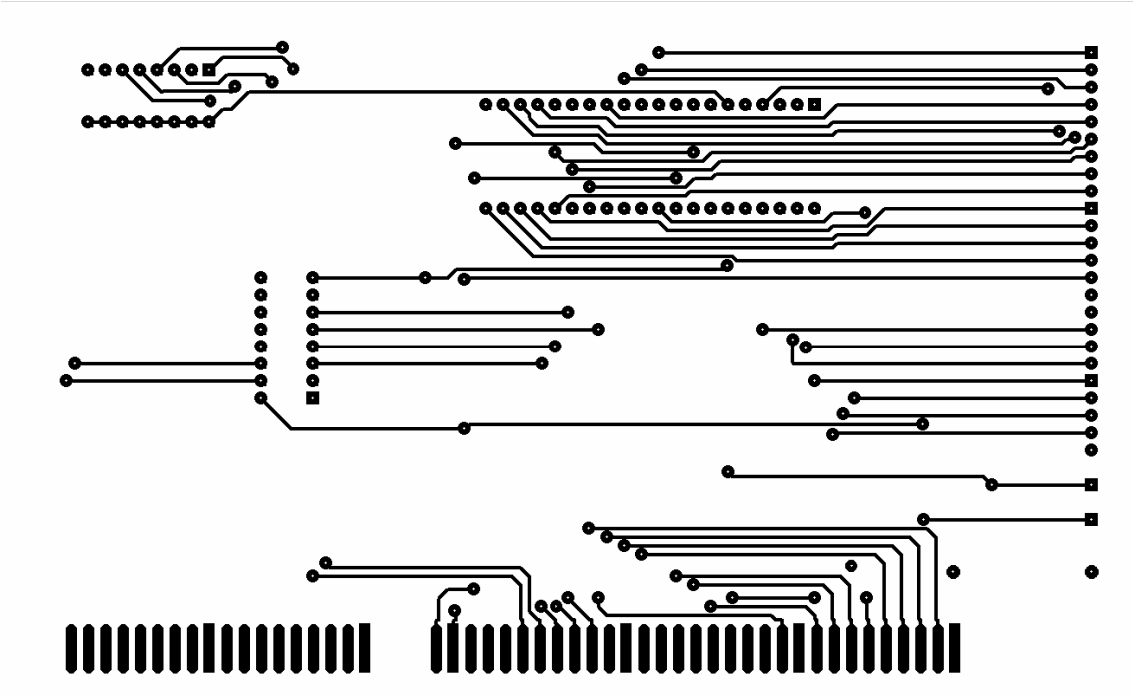
Tarjeta del micrófono, filtro, amplificador y sumador.



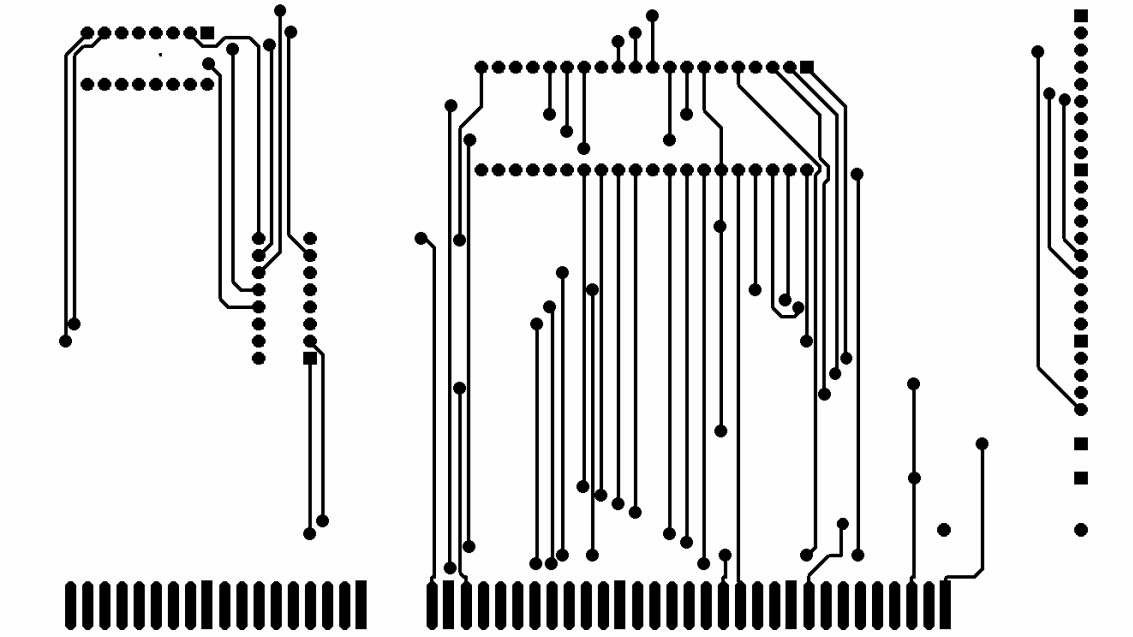
Tarjeta del ADC



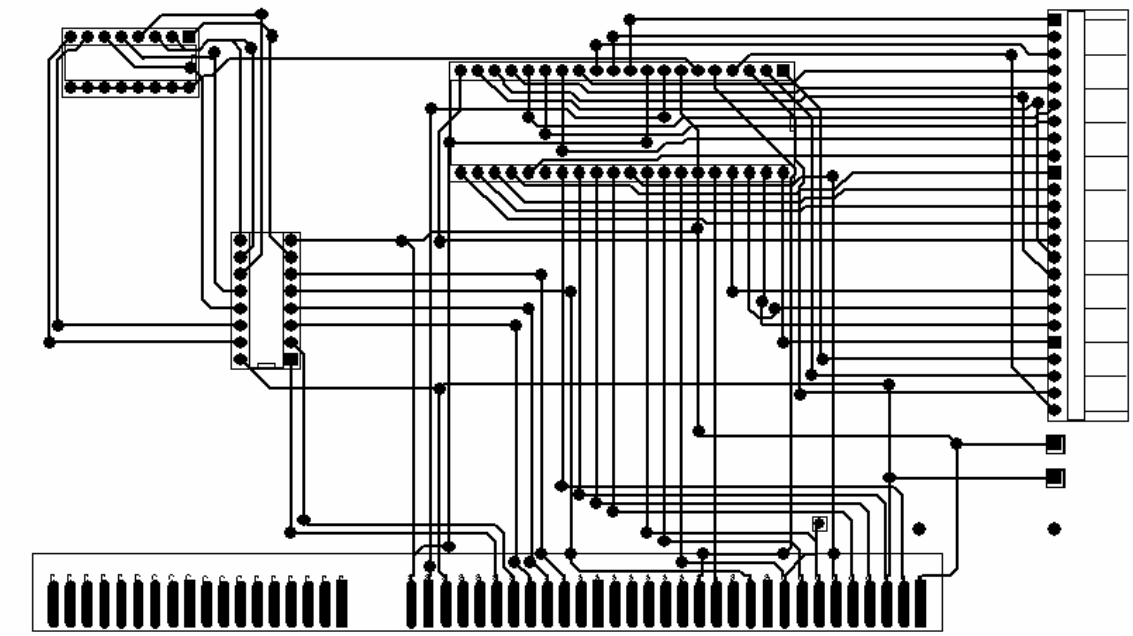
Tarjeta de etapa de potencia



Tarjeta del PPI (1 cara)



Tarjeta del PPI (2 cara)



Tarjeta del PPI (1,2 caras)

APENDICE B

Este apéndice contiene la versión completa de el software aplicado al sistema que se encuentra realizado en lenguaje C.

```
/**************************************INICIALIZACION DEL PROGRAMA*****************************************/

#include<stdio.h>
#include<conio.h>
#include<dos.h>
#include<stdlib.h>
int puerto_base = 0x280;
int puerto_a = 0x280;
int puerto_b = 0x281;
int puerto_c = 0x282;
int puerto_ctrl = 0x283;
int palabra_ctrl = 0x99;
int puerto_vel =0x378;
int conteo = 0;
FILE *muestra;
FILE *muestra2;


int *resultado;
char buffer [12000];
char buffer2 [12000];

char c;
char d;
char e;
char h;
int i=0;
int j=0;
int z=0;
int ci7;
int ci8;
int ci9;
int cs7;
int cs8;
int cs9;
int cs6;
int ci;
int ci6;
int ci1;
int ci2;
int ci3;
int ci4;
int ci5;
int cs1;
int cs2;
int cs3;
int cs4;
int cs5;
int di1;
int di2;
int df1;
int df2;
int k=0;
int M=0;/*cuenta los caracteres dentro de rango*/
int N=0;/*cuenta los caracteres fuera de rango*/
int b=0;/*contadores de fin de archivo*/
int s=0;
int t=0;
int u=0;
int v=0;
int O=0;
int P=0;
int Q=0;
int salida=0;
int limpia;
int limpia1;
unsigned char dato = 0;
unsigned char sensor;
unsigned char check;
unsigned char lent = 50;
unsigned char rapid = 255;
unsigned char apagado = 0;
unsigned char izquierda = 254;
unsigned char derecha = 127;

void escribe();
```

```
void compara();
void compara1();
void compara2();
void compara3();
void compara4();
void compara5();
void compara6();
void compara7();
void compara8();
void compara9();
void motori();
void motori1();
void motori2();
void motori3();
void motord();
void motord1();
void motord2();
void motord3();
void velocidad();
void rapido();
void rapido1();
void rapido2();
void lento();
void lento1();
void lento2();
void apaga();
void apaga1();
void apaga2();
void rango();

/*************************************************PROGRAMA PRINCIPAL*********************************************/

void main (void)
{

   outportb (puerto_ctrl, palabra_ctrl);
   outportb (puerto_b, 0xFF);
   clrscr();
   printf("Hola\n");
   printf("este programa tiene como finalidad\n");
   printf("accesar al sistema por medio del reconocimiento de voz\n");
   printf("ademas te da la opcion de mover un motor de CD\n");
   printf("despues de que se quite esta pantalla el CPU esta listo para recibir tu orden\n");
   printf("despues de que presiones una tecla\n");
   printf("di    ' HOLA'    por favor\n");
   delay(5000);



   while (t<8)
   {
      t=0;
      clrscr();
      escribe();
      compara();
      compara1();
      compara2();
      compara3();
      compara4();
      compara5();
      compara6();
      compara7();
      compara8();
      compara9();
   }
   if (t>=8)
   {
      clrscr();
      printf("Hola\n");
      printf("ahora que quieres hacer\n");
      printf("decir DOS gira el motor a la derecha\n");
      printf("decir NUEVE gira el motor a la izquierda\n");
      printf("despues de que se quite esta pantalla el CPU esta listo para recibir tu orden\n");
      delay(5000);
      t=0;
      while(t!=1)
      {
         clrscr();
```

```
                                printf("quieres cambiar el sentido de giro\n");
                                delay(5000);

        u=0;
        v=0;
        while(u<3&&v<3)
                                {
                                        clrscr();
                                        escribe();
                                        motori();
                                        motori1();
                                        motori2();
                                        motori3();
                                        motord();
                                        motord1();
                                        motord2();
                                        motord3();
                                }
                                if (u>=3)
                                {
                                        printf("gira motor izquierda");
           outportb(puerto_b,izquierda);                        // orden de salida de datos de control
                                        velocidad();
                                }
                                else if(v>=3)
                                {
                                        printf("gira motor derecha\n");
           outportb(puerto_b,derecha);                          //orden  de salida de datos de control
                                        velocidad();
                                }

     }
   }

}

/*************************************************************************************************/
/******************RUTINA DE ADQUISICION Y ALMACENAMIENTO*********************************/
void escribe()
{

   unsigned char putc;
   char sens;                   //declara el sensor de silencios
   char sen;
   FILE *muestra1;
   muestra1 = fopen ("71.txt","wb+");    //abre archivo para escritura
   clrscr();                        //limpia pantalla
   conteo=0;


   while (!kbhit())
   {
   }
   sens= inportb(puerto_a);             //control de silencios
   while (sens >= ',„' && sens<= '¥')
   {
      sens= inportb(puerto_a);
   }

   while (conteo <=200)                 //ciclo para ingreso de muestras
              {
                 sensor= inportb(puerto_c);             //checa si hay dato
      check = sensor & 1;
                 while (check == 1)
                           {
                              dato = inportb (puerto_a);
          if(dato!=0x1a)
                              {
                              fputc (dato,muestra1);          //escribe datos
                              while (check == 1)
                                        {
                                           sensor = inportb (puerto_c);
                                           check = sensor & 1;
                                        }
                              sen= inportb(puerto_a);
                              if (sen >= '‡' && sen<= '¥')
                                        {
                                           conteo = conteo + 1;
```

```
                                        if (conteo==199)
                                                {
                                                   dato='$';
                                                   fputc(dato,muestra1);
                                                   dato='$';
                                                   fputc(dato,muestra1);
                                                   dato='$';
                                                   fputc(dato,muestra1);
                                                }
                                        }
                                else
                                        {
                                           conteo = 0;
                                        }
                                }
                        }
   }
fclose(muestra1);                       //cierra archivo

return;
}
/*****************************************************************************************************************/
/******RUTINAS DE  LECTURA Y MANIPULACION DE ARCHIVOS PARA SU RECONOCIMIENTO*****************************/
void compara()
{
   limpia =1;
   limpia1 =1;
            clrscr();
            if (((muestra= fopen("1.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
            {
                        perror ("archivo nulo");
                        exit (1);
            }
            resultado= &muestra;
            while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
            {
    while(limpia!=0)
    {
    limpia=fflush(muestra);
    }
    while(limpia1!=0)
    {
    limpia1=fflush(muestra2);
    }
    buffer [i]=c;
                        c=fgetc (muestra);
                        i++;

                        buffer2 [j]=d;
                        d=fgetc (muestra2);
                j++;

                        di1=d+1;
                        di2=d+2;
                        df1=d-1;
                        df2=d-2;
                        while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
    {
       buffer2 [j]=d;
                                d=fgetc (muestra2);

                                j++;
    }


                        while (s<=3 || b<=3)
                        {
                                buffer [i]=c;
                                c=fgetc (muestra);
                                i++;
                                if (c== '$')
                                {
                                        s++;
                                }
                                else
                                {
                                s=0;
                                }
```

```
                                        if (s==3)
                                        {
                                                buffer [i--]= '\0';
                        if (ferror (muestra))
                                perror ("error");
                                buffer2 [j--]= '\0';
                        if (ferror (muestra2))
                                perror ("error");
                                fclose (muestra);
                                fclose (muestra2);
                                printf("%d,%d\n",M,N);
                        if (M>=1200)
                        {
                                                printf("si\n");
                t++;
                        }
                        else
                        {
                                                printf("no\n");
                                        }
                printf("%d\n",limpia);
                printf("%d\n",limpia1);

                                        return;
                                }

        else
        {
                                        buffer2 [j]=d;
                                        d=fgetc (muestra2);

                                        j++;
                                        if (d== '$')
                                        {
                                                b++;
                                        }
                                        else
                                        {
                                                b=0;
                                        }
                                        if (b==3)
                                        {
                                                buffer [i--]= '\0';
                        if (ferror (muestra))
                                perror ("error");
                                buffer2 [j--]= '\0';
                        if (ferror (muestra2))
                                perror ("error");
                                fclose (muestra);
                                fclose (muestra2);
                                printf("%d,%d\n",M,N);
                        if (M>=1200)
                        {
                                                printf("si\n");
                                                        t++;
                                                }
                                                else
                                                {
                                                        printf("no\n");
                                                }

                printf("%d\n",limpia);
                printf("%d\n",limpia1);

                                        return;
                                                }
                else
                {
                rango();
                }
        }
    }
  }
  return;
}

void compara1()
```

```
{
   i=0;
   j=0;
   M=0;
   N=0;
   s=0;
   b=0;
   printf("%d\n",limpia);
   printf("%d\n",limpia1);
   limpia =1;
   limpia1 =1;
            clrscr();
            if (((muestra= fopen("2.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
            {
                     perror ("archivo nulo");
                     exit (1);
            }
            resultado= &muestra;
            while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
            {

   while(limpia!=0)
   {
   limpia=fflush(muestra);
   }
   while(limpia1!=0)
   {
   limpia1=fflush(muestra2);
   }
                     buffer [i]=c;
                     c=fgetc (muestra);
                     i++;

                     buffer2 [j]=d;
                     d=fgetc (muestra2);
            j++;

                     di1=d+1;
                     di2=d+2;
                     df1=d-1;
                     df2=d-2;
             while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
   {
                     buffer2 [j]=d;
                        d=fgetc (muestra2);

                        j++;
   }


                     while (s<=3 || b<=3)
                     {
                              buffer [i]=c;
                              c=fgetc (muestra);
                              i++;
                        if (c== '$')
                              {
                              s++;
                              }
                              else
                              {
                                       s=0;
                              }
                              if (s==3)
                              {
                                       buffer [i--]= '\0';
                           if (ferror (muestra))
                                       perror ("error");
                                       buffer2 [j--]= '\0';
                                 if (ferror (muestra2))
                                       perror ("error");
                                       fclose (muestra);
                                       fclose (muestra2);
                                       printf("%d,%d\n",M,N);
                     if (M>=1200)
                     {
                                          printf("si\n");
            t++;
```

```
                }
                else
                {
                                printf("no\n");
                }

                printf("%d\n",limpia);
        printf("%d\n",limpia1);
                return;
            }

        else
        {
                buffer2 [j]=d;
                d=fgetc (muestra2);
        j++;
                if (d== '$')
                {
                                b++;
                }
                else
                {
                   b=0;
                }
                if (b==3)
                {
                                buffer [i--]= '\0';
                        if (ferror (muestra))
                           perror ("error");
                                buffer2 [j--]= '\0';
            if (ferror (muestra2))
                                perror ("error");
                                fclose (muestra);
                                fclose (muestra2);
                                printf("%d,%d\n",M,N);
                                if (M>=1200)
                                {
                                        printf("si\n");
               t++;
                               }
                               else
                               {
                                        printf("no\n");
                               }

                printf("%d\n",limpia);
            printf("%d\n",limpia1);
                return;
                }
            else
            {
            rango();
            }
        }
            }
   }
   return;
}

void compara2()
{
   i=0;
   j=0;
   M=0;
   N=0;
   s=0;
   b=0;
   printf("%d\n",limpia);
   printf("%d\n",limpia1);
   limpia =1;
   limpia1 =1;
            clrscr();
            if (((muestra= fopen("3.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
            {
                        perror ("archivo nulo");
                        exit (1);
            }
            resultado= &muestra;
```

```
        while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
        {

while(limpia!=0)
{
limpia=fflush(muestra);
}
while(limpia1!=0)
{
limpia1=fflush(muestra2);
}
                        buffer [i]=c;
                        c=fgetc (muestra);
                        i++;

                        buffer2 [j]=d;
                        d=fgetc (muestra2);
            j++;

                        di1=d+1;
                        di2=d+2;
                        df1=d-1;
                        df2=d-2;
            while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
{
                        buffer2 [j]=d;
                           d=fgetc (muestra2);

                           j++;
}


                        while (s<=3 || b<=3)
                        {
                                    buffer [i]=c;
                                    c=fgetc (muestra);
                                    i++;
                           if (c== '$')
                                    {
                                    s++;
                                    }
                                    else
                                    {
                                                s=0;
                                    }
                                    if (s==3)
                                    {
                                                buffer [i--]= '\0';
                               if (ferror (muestra))
                                                perror ("error");
                                                buffer2 [j--]= '\0';
                                       if (ferror (muestra2))
                                                perror ("error");
                                                fclose (muestra);
                                                fclose (muestra2);
                                                printf("%d,%d\n",M,N);
                     if (M>=1200)
                     {
                                                   printf("si\n");
          t++;
                     }
                     else
                     {
                                                   printf("no\n");
                                       }

                                       printf("%d\n",limpia);
       printf("%d\n",limpia1);
                                       return;
                                    }

     else
     {
                                       buffer2 [j]=d;
                                       d=fgetc (muestra2);
       j++;
                                       if (d== '$')
                                       {
```

```
                                        b++;
                                }
                                else
                                {
                                   b=0;
                                }
                                if (b==3)
                                {
                                        buffer [i--]= '\0';
                        if (ferror (muestra))
                                   perror ("error");
                                        buffer2 [j--]= '\0';
        if (ferror (muestra2))
                                        perror ("error");
                                        fclose (muestra);
                                        fclose (muestra2);
                                        printf("%d,%d\n",M,N);
                                        if (M>=1200)
                                        {
                                                printf("si\n");
          t++;
                                    }
                                    else
                                    {
                                            printf("no\n");
                                    }

                                printf("%d\n",limpia);
        printf("%d\n",limpia1);
                                return;
                                }
          else
          {
          rango();
          }
       }
            }
   }
   return;
}
void compara3()
{
   i=0;
   j=0;
   M=0;
   N=0;
   s=0;
   b=0;
   printf("%d\n",limpia);
   printf("%d\n",limpia1);
   limpia =1;
   limpia1 =1;
            clrscr();
            if (((muestra= fopen("4.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
            {
                    perror ("archivo nulo");
                    exit (1);
            }
            resultado= &muestra;
            while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
            {

   while(limpia!=0)
   {
   limpia=fflush(muestra);
   }
   while(limpia1!=0)
   {
   limpia1=fflush(muestra2);
   }
                    buffer [i]=c;
                    c=fgetc (muestra);
                    i++;

                    buffer2 [j]=d;
                    d=fgetc (muestra2);
            j++;
```

```
                    di1=d+1;
                    di2=d+2;
                    df1=d-1;
                    df2=d-2;
          while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
{
                    buffer2 [j]=d;
                       d=fgetc (muestra2);

                       j++;
}


                    while (s<=3 || b<=3)
                    {
                              buffer [i]=c;
                              c=fgetc (muestra);
                              i++;
                       if (c== '$')
                              {
                              s++;
                              }
                              else
                              {
                                        s=0;
                              }
                              if (s==3)
                              {
                                        buffer [i--]= '\0';
                          if (ferror (muestra))
                                        perror ("error");
                                        buffer2 [j--]= '\0';
                                  if (ferror (muestra2))
                                        perror ("error");
                                        fclose (muestra);
                                        fclose (muestra2);
                                        printf("%d,%d\n",M,N);
                    if (M>=1200)
                    {
                                           printf("si\n");
          t++;
                    }
                    else
                    {
                                           printf("no\n");
                                  }

                                  printf("%d\n",limpia);
      printf("%d\n",limpia1);
                                  return;
                              }

    else
    {
                                  buffer2 [j]=d;
                                  d=fgetc (muestra2);
      j++;
                                  if (d== '$')
                                  {
                                        b++;
                                  }
                                  else
                                  {
                                     b=0;
                                  }
                                  if (b==3)
                                  {
                                        buffer [i--]= '\0';
                            if (ferror (muestra))
                                     perror ("error");
                                        buffer2 [j--]= '\0';
          if (ferror (muestra2))
                                        perror ("error");
                                        fclose (muestra);
                                        fclose (muestra2);
                                        printf("%d,%d\n",M,N);
                                        if (M>=1200)
                                        {
```

```
                                                                printf("si\n");
            t++;
                                                }
                                                else
                                                {
                                                        printf("no\n");
                                                }

                                        printf("%d\n",limpia);
            printf("%d\n",limpia1);
                                        return;
                                        }
          else
          {
          rango();
          }
        }
            }
   }
   return;
}


void compara4()
{
   i=0;
   j=0;
   M=0;
   N=0;
   s=0;
   b=0;
   printf("%d\n",limpia);
   printf("%d\n",limpia1);
   limpia =1;
   limpia1 =1;
            clrscr();
            if (((muestra= fopen("5.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
            {
                    perror ("archivo nulo");
                    exit (1);
            }
            resultado= &muestra;
            while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
            {

    while(limpia!=0)
    {
    limpia=fflush(muestra);
    }
    while(limpia1!=0)
    {
    limpia1=fflush(muestra2);
    }
                    buffer [i]=c;
                    c=fgetc (muestra);
                    i++;

                    buffer2 [j]=d;
                    d=fgetc (muestra2);
            j++;

                    di1=d+1;
                    di2=d+2;
                    df1=d-1;
                    df2=d-2;
            while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
    {
                    buffer2 [j]=d;
                       d=fgetc (muestra2);

                       j++;
    }


                    while (s<=3 || b<=3)
                    {
                            buffer [i]=c;
                            c=fgetc (muestra);
```

```
                        i++;
                    if (c== '$')
                        {
                        s++;
                        }
                        else
                        {
                                s=0;
                        }
                        if (s==3)
                        {
                                buffer [i--]= '\0';
                      if (ferror (muestra))
                                perror ("error");
                                buffer2 [j--]= '\0';
                          if (ferror (muestra2))
                                perror ("error");
                                fclose (muestra);
                                fclose (muestra2);
                                printf("%d,%d\n",M,N);
            if (M>=1200)
            {
                                    printf("si\n");
    t++;
            }
            else
            {
                                    printf("no\n");
                          }

                          printf("%d\n",limpia);
  printf("%d\n",limpia1);
                          return;
                        }

else
{
                          buffer2 [j]=d;
                          d=fgetc (muestra2);
  j++;
                          if (d== '$')
                          {
                                b++;
                          }
                          else
                          {
                             b=0;
                          }
                          if (b==3)
                          {
                                buffer [i--]= '\0';
                      if (ferror (muestra))
                             perror ("error");
                                buffer2 [j--]= '\0';
    if (ferror (muestra2))
                                perror ("error");
                                fclose (muestra);
                                fclose (muestra2);
                                printf("%d,%d\n",M,N);
                                if (M>=1200)
                                {
                                        printf("si\n");
      t++;
                                }
                                else
                                {
                                    printf("no\n");
                                }

                          printf("%d\n",limpia);
    printf("%d\n",limpia1);
                          return;
                          }
  else
  {
  rango();
  }
}
```

```
                }
    }
    return;
}

void compara5()
{
    i=0;
    j=0;
    M=0;
    N=0;
    s=0;
    b=0;
    printf("%d\n",limpia);
    printf("%d\n",limpia1);
    limpia =1;
    limpia1 =1;
                clrscr();
                if (((muestra= fopen("6.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
                {
                                perror ("archivo nulo");
                                exit (1);
                }
                resultado= &muestra;
                while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
                {

       while(limpia!=0)
       {
       limpia=fflush(muestra);
       }
       while(limpia1!=0)
       {
       limpia1=fflush(muestra2);
       }
                                buffer [i]=c;
                                c=fgetc (muestra);
                                i++;

                                buffer2 [j]=d;
                                d=fgetc (muestra2);
                  j++;

                                di1=d+1;
                                di2=d+2;
                                df1=d-1;
                                df2=d-2;
                   while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
       {
                                buffer2 [j]=d;
                                  d=fgetc (muestra2);

                                  j++;
       }


                                while (s<=3 || b<=3)
                                {
                                                buffer [i]=c;
                                                c=fgetc (muestra);
                                                i++;
                                  if (c== '$')
                                                {
                                                s++;
                                                }
                                                else
                                                {
                                                                s=0;
                                                }
                                                if (s==3)
                                                {
                                                                buffer [i--]= '\0';
                                     if (ferror (muestra))
                                                                perror ("error");
                                                                buffer2 [j--]= '\0';
                                        if (ferror (muestra2))
                                                                perror ("error");
                                                                fclose (muestra);
```

```
                                                fclose (muestra2);
                                                printf("%d,%d\n",M,N);
                        if (M>=1200)
                        {
                                                    printf("si\n");
                t++;
                        }
                        else
                        {
                                                    printf("no\n");
                                        }

                                        printf("%d\n",limpia);
            printf("%d\n",limpia1);
                                        return;
                                    }

            else
            {
                                        buffer2 [j]=d;
                                        d=fgetc (muestra2);
               j++;
                                        if (d== '$')
                                        {
                                                    b++;
                                        }
                                        else
                                        {
                                           b=0;
                                        }
                                        if (b==3)
                                        {
                                                    buffer [i--]= '\0';
                                    if (ferror (muestra))
                                           perror ("error");
                                                    buffer2 [j--]= '\0';
               if (ferror (muestra2))
                                                    perror ("error");
                                                    fclose (muestra);
                                                    fclose (muestra2);
                                                    printf("%d,%d\n",M,N);
                                                    if (M>=1200)
                                                    {
                                                                printf("si\n");
                   t++;
                                                    }
                                                    else
                                                    {
                                                                printf("no\n");
                                                    }

                                        printf("%d\n",limpia);
               printf("%d\n",limpia1);
                                        return;
                                        }
               else
               {
               rango();
               }
            }
                }
    }
    return;
}

void compara6()
{
    i=0;
    j=0;
    M=0;
    N=0;
    s=0;
    b=0;
    printf("%d\n",limpia);
    printf("%d\n",limpia1);
    limpia =1;
    limpia1 =1;
                clrscr();
```

```
        if (((muestra= fopen("7.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
        {
                perror ("archivo nulo");
                exit (1);
        }
        resultado= &muestra;
        while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
        {

while(limpia!=0)
{
limpia=fflush(muestra);
}
while(limpia1!=0)
{
limpia1=fflush(muestra2);
}
                buffer [i]=c;
                c=fgetc (muestra);
                i++;

                buffer2 [j]=d;
                d=fgetc (muestra2);
          j++;

                di1=d+1;
                di2=d+2;
                df1=d-1;
                df2=d-2;
          while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
{
                buffer2 [j]=d;
                   d=fgetc (muestra2);

                   j++;
}


                while (s<=3 || b<=3)
                {
                        buffer [i]=c;
                        c=fgetc (muestra);
                        i++;
                   if (c== '$')
                        {
                        s++;
                        }
                        else
                        {
                                s=0;
                        }
                        if (s==3)
                        {
                                buffer [i--]= '\0';
                     if (ferror (muestra))
                                perror ("error");
                                buffer2 [j--]= '\0';
                          if (ferror (muestra2))
                                perror ("error");
                                fclose (muestra);
                                fclose (muestra2);
                                printf("%d,%d\n",M,N);
              if (M>=1200)
              {
                                   printf("si\n");
        t++;
              }
              else
              {
                                   printf("no\n");
                          }

                          printf("%d\n",limpia);
    printf("%d\n",limpia1);
                          return;
                        }

   else
```

```
        {
                                        buffer2 [j]=d;
                                        d=fgetc (muestra2);
          j++;
                                        if (d== '$')
                                        {
                                                b++;
                                        }
                                        else
                                        {
                                           b=0;
                                        }
                                        if (b==3)
                                        {
                                                buffer [i--]= '\0';
                                    if (ferror (muestra))
                                           perror ("error");
                                                buffer2 [j--]= '\0';
             if (ferror (muestra2))
                                                perror ("error");
                                                fclose (muestra);
                                                fclose (muestra2);
                                                printf("%d,%d\n",M,N);
                                                if (M>=1200)
                                                {
                                                        printf("si\n");
                t++;
                                              }
                                              else
                                              {
                                                        printf("no\n");
                                              }

                                        printf("%d\n",limpia);
             printf("%d\n",limpia1);
                                        return;
                                        }
             else
             {
             rango();
             }
        }
            }
   }
   return;
}

void compara7()
{
   i=0;
   j=0;
   M=0;
   N=0;
   s=0;
   b=0;
   printf("%d\n",limpia);
   printf("%d\n",limpia1);
   limpia =1;
   limpia1 =1;
             clrscr();
             if (((muestra= fopen("8.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
             {
                        perror ("archivo nulo");
                        exit (1);
             }
             resultado= &muestra;
             while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
             {

    while(limpia!=0)
    {
    limpia=fflush(muestra);
    }
    while(limpia1!=0)
    {
    limpia1=fflush(muestra2);
    }
                        buffer [i]=c;
```

```
                    c=fgetc (muestra);
                    i++;

                    buffer2 [j]=d;
                    d=fgetc (muestra2);
          j++;

                    di1=d+1;
                    di2=d+2;
                    df1=d-1;
                    df2=d-2;
          while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
{
                    buffer2 [j]=d;
                      d=fgetc (muestra2);

                      j++;
}


                    while (s<=3 || b<=3)
                    {
                              buffer [i]=c;
                              c=fgetc (muestra);
                              i++;
                      if (c== '$')
                              {
                              s++;
                              }
                              else
                              {
                                        s=0;
                              }
                              if (s==3)
                              {
                                        buffer [i--]= '\0';
                          if (ferror (muestra))
                                        perror ("error");
                                        buffer2 [j--]= '\0';
                                  if (ferror (muestra2))
                                        perror ("error");
                                        fclose (muestra);
                                        fclose (muestra2);
                                        printf("%d,%d\n",M,N);
              if (M>=1200)
              {
                                            printf("si\n");
      t++;
              }
              else
              {
                                            printf("no\n");
                                  }

                                  printf("%d\n",limpia);
   printf("%d\n",limpia1);
                                  return;
                                }

  else
  {
                                  buffer2 [j]=d;
                                  d=fgetc (muestra2);
    j++;
                                  if (d== '$')
                                  {
                                        b++;
                                  }
                                  else
                                  {
                                    b=0;
                                  }
                                  if (b==3)
                                  {
                                        buffer [i--]= '\0';
                            if (ferror (muestra))
                                    perror ("error");
                                        buffer2 [j--]= '\0';
```

```
            if (ferror (muestra2))
                                                perror ("error");
                                                fclose (muestra);
                                                fclose (muestra2);
                                                printf("%d,%d\n",M,N);
                                                if (M>=1200)
                                                {
                                                        printf("si\n");
                t++;
                                                }
                                                else
                                                {
                                                        printf("no\n");
                                                }

                                        printf("%d\n",limpia);
            printf("%d\n",limpia1);
                                        return;
                                        }
          else
          {
          rango();
          }
        }
            }
   }
   return;
}

void compara8()
{
   i=0;
   j=0;
   M=0;
   N=0;
   s=0;
   b=0;
   printf("%d\n",limpia);
   printf("%d\n",limpia1);
   limpia =1;
   limpia1 =1;
            clrscr();
            if (((muestra= fopen("9.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
            {
                        perror ("archivo nulo");
                        exit (1);
            }
            resultado= &muestra;
            while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
            {

   while(limpia!=0)
   {
   limpia=fflush(muestra);
   }
   while(limpia1!=0)
   {
   limpia1=fflush(muestra2);
   }
                        buffer [i]=c;
                        c=fgetc (muestra);
                        i++;

                        buffer2 [j]=d;
                        d=fgetc (muestra2);
            j++;

                        di1=d+1;
                        di2=d+2;
                        df1=d-1;
                        df2=d-2;
             while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
   {
                        buffer2 [j]=d;
                           d=fgetc (muestra2);

                           j++;
   }
```

```
                while (s<=3 || b<=3)
                {
                        buffer [i]=c;
                        c=fgetc (muestra);
                        i++;
                   if (c== '$')
                        {
                        s++;
                        }
                        else
                        {
                                s=0;
                        }
                        if (s==3)
                        {
                                buffer [i--]= '\0';
                     if (ferror (muestra))
                                perror ("error");
                                buffer2 [j--]= '\0';
                          if (ferror (muestra2))
                                perror ("error");
                                fclose (muestra);
                                fclose (muestra2);
                                printf("%d,%d\n",M,N);
          if (M>=1200)
          {
                                    printf("si\n");
    t++;
          }
          else
          {
                                    printf("no\n");
                          }

                          printf("%d\n",limpia);
  printf("%d\n",limpia1);
                          return;
                        }

else
{
                          buffer2 [j]=d;
                          d=fgetc (muestra2);
  j++;
                          if (d== '$')
                          {
                                b++;
                          }
                          else
                          {
                             b=0;
                          }
                          if (b==3)
                          {
                                buffer [i--]= '\0';
                     if (ferror (muestra))
                             perror ("error");
                                buffer2 [j--]= '\0';
    if (ferror (muestra2))
                                perror ("error");
                                fclose (muestra);
                                fclose (muestra2);
                                printf("%d,%d\n",M,N);
                                if (M>=1200)
                                {
                                        printf("si\n");
      t++;
                                }
                                else
                                {
                                    printf("no\n");
                                }

                          printf("%d\n",limpia);
    printf("%d\n",limpia1);
                          return;
```

```
                                        }
            else
            {
            rango();
            }
          }
              }
    }
    return;
}


void compara9()
{
    i=0;
    j=0;
    M=0;
    N=0;
    s=0;
    b=0;
    printf("%d\n",limpia);
    printf("%d\n",limpia1);
    limpia =1;
    limpia1 =1;
                clrscr();
                if (((muestra= fopen("10.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
                {
                            perror ("archivo nulo");
                            exit (1);
                }
                resultado= &muestra;
                while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
                {

      while(limpia!=0)
      {
      limpia=fflush(muestra);
      }
      while(limpia1!=0)
      {
      limpia1=fflush(muestra2);
      }
                            buffer [i]=c;
                            c=fgetc (muestra);
                            i++;

                            buffer2 [j]=d;
                            d=fgetc (muestra2);
                  j++;

                            di1=d+1;
                            di2=d+2;
                            df1=d-1;
                            df2=d-2;
                   while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
      {
                            buffer2 [j]=d;
                               d=fgetc (muestra2);

                               j++;
      }


                            while (s<=3 || b<=3)
                            {
                                        buffer [i]=c;
                                        c=fgetc (muestra);
                                        i++;
                               if (c== '$')
                                        {
                                        s++;
                                        }
                                        else
                                        {
                                                    s=0;
                                        }
                                        if (s==3)
                                        {
```

```
                                        buffer [i--]= '\0';
                                if (ferror (muestra))
                                        perror ("error");
                                        buffer2 [j--]= '\0';
                                if (ferror (muestra2))
                                        perror ("error");
                                        fclose (muestra);
                                        fclose (muestra2);
                                        printf("%d,%d\n",M,N);
                        if (M>=1200)
                        {
                                                printf("si\n");
                    t++;
                        }
                        else
                        {
                                                printf("no\n");
                                }

                                printf("%d\n",limpia);
            printf("%d\n",limpia1);
                                return;
                            }

            else
            {
                                buffer2 [j]=d;
                                d=fgetc (muestra2);
               j++;
                                if (d== '$')
                                {
                                        b++;
                                }
                                else
                                {
                                   b=0;
                                }
                                if (b==3)
                                {
                                        buffer [i--]= '\0';
                            if (ferror (muestra))
                                   perror ("error");
                                        buffer2 [j--]= '\0';
               if (ferror (muestra2))
                                        perror ("error");
                                        fclose (muestra);
                                        fclose (muestra2);
                                        printf("%d,%d\n",M,N);
                                        if (M>=1200)
                                        {
                                                printf("si\n");
                  t++;
                                      }
                                      else
                                      {
                                                printf("no\n");
                                      }

                                printf("%d\n",limpia);
               printf("%d\n",limpia1);
                                return;
                                }
               else
               {
               rango();
               }
            }
                }
    }
    return;
}

void motori()
{
    i=0;
    j=0;
    M=0;
    N=0;
```

```
s=0;
b=0;
printf("%d\n",limpia);
printf("%d\n",limpia1);
limpia =1;
limpia1 =1;
        clrscr();
        if (((muestra= fopen("dos.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
        {
                perror ("archivo nulo");
                exit (1);
        }
        resultado= &muestra;
        while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
        {

  while(limpia!=0)
  {
  limpia=fflush(muestra);
  }
  while(limpia1!=0)
  {
  limpia1=fflush(muestra2);
  }
                buffer [i]=c;
                c=fgetc (muestra);
                i++;

                buffer2 [j]=d;
                d=fgetc (muestra2);
          j++;

                di1=d+1;
                di2=d+2;
                df1=d-1;
                df2=d-2;
          while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
  {
                buffer2 [j]=d;
                  d=fgetc (muestra2);

                  j++;
  }


                while (s<=3 || b<=3)
                {
                        buffer [i]=c;
                        c=fgetc (muestra);
                        i++;
                  if (c== '$')
                        {
                        s++;
                        }
                        else
                        {
                                s=0;
                        }
                        if (s==3)
                        {
                                buffer [i--]= '\0';
                     if (ferror (muestra))
                                perror ("error");
                                buffer2 [j--]= '\0';
                           if (ferror (muestra2))
                                perror ("error");
                                fclose (muestra);
                                fclose (muestra2);
                                printf("%d,%d\n",M,N);
            if (M>=1200)
            {
                                    printf("si\n");

        u++;
            }
            else
            {
                                    printf("no\n");
```

```
                                        }

                                        printf("%d\n",limpia);
        printf("%d\n",limpia1);
                                        return;
                                    }

      else
      {
                                        buffer2 [j]=d;
                                        d=fgetc (muestra2);
        j++;
                                        if (d== '$')
                                        {
                                                b++;
                                        }
                                        else
                                        {
                                           b=0;
                                        }
                                        if (b==3)
                                        {
                                                buffer [i--]= '\0';
                                if (ferror (muestra))
                                           perror ("error");
                                                buffer2 [j--]= '\0';
        if (ferror (muestra2))
                                                perror ("error");
                                                fclose (muestra);
                                                fclose (muestra2);
                                                printf("%d,%d\n",M,N);
                                                if (M>=1200)
                                                {
                                                        printf("si\n");

          u++;
                                                }
                                                else
                                                {
                                                        printf("no\n");
                                                }

                                        printf("%d\n",limpia);
        printf("%d\n",limpia1);
                                        return;
                                        }
        else
        {
        rango();
        }
      }
          }
  }
  return;
}

void motori1()
{
  i=0;
  j=0;
  M=0;
  N=0;
  s=0;
  b=0;
  printf("%d\n",limpia);
  printf("%d\n",limpia1);
  limpia =1;
  limpia1 =1;
          clrscr();
          if (((muestra= fopen("dos1.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
          {
                  perror ("archivo nulo");
                  exit (1);
          }
          resultado= &muestra;
          while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
          {
```

```
while(limpia!=0)
{
limpia=fflush(muestra);
}
while(limpia1!=0)
{
limpia1=fflush(muestra2);
}
                        buffer [i]=c;
                        c=fgetc (muestra);
                        i++;

                        buffer2 [j]=d;
                        d=fgetc (muestra2);
            j++;

                        di1=d+1;
                        di2=d+2;
                        df1=d-1;
                        df2=d-2;
            while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
{
                        buffer2 [j]=d;
                           d=fgetc (muestra2);

                           j++;
}


                        while (s<=3 || b<=3)
                        {
                                    buffer [i]=c;
                                    c=fgetc (muestra);
                                    i++;
                           if (c== '$')
                                    {
                                    s++;
                                    }
                                    else
                                    {
                                                s=0;
                                    }
                                    if (s==3)
                                    {
                                                buffer [i--]= '\0';
                              if (ferror (muestra))
                                                perror ("error");
                                                buffer2 [j--]= '\0';
                                       if (ferror (muestra2))
                                                perror ("error");
                                                fclose (muestra);
                                                fclose (muestra2);
                                                printf("%d,%d\n",M,N);
                  if (M>=1200)
                  {
                                                   printf("si\n");

          u++;
                  }
                  else
                  {
                                                   printf("no\n");
                                       }

                                       printf("%d\n",limpia);
      printf("%d\n",limpia1);
                                       return;
                                    }

   else
   {
                                       buffer2 [j]=d;
                                       d=fgetc (muestra2);
      j++;
                                       if (d== '$')
                                       {
                                                b++;
                                       }
```

```
                                        else
                                        {
                                           b=0;
                                        }
                                        if (b==3)
                                        {
                                                buffer [i--]= '\0';
                                    if (ferror (muestra))
                                           perror ("error");
                                                buffer2 [j--]= '\0';
            if (ferror (muestra2))
                                                perror ("error");
                                                fclose (muestra);
                                                fclose (muestra2);
                                                printf("%d,%d\n",M,N);
                                                if (M>=1200)
                                                {
                                                        printf("si\n");

               u++;
                                                }
                                                else
                                                {
                                                        printf("no\n");
                                                }

                                        printf("%d\n",limpia);
            printf("%d\n",limpia1);
                                        return;
                                        }
           else
           {
           rango();
           }
        }
              }
   }
   return;
}


void motori2()
{
   i=0;
   j=0;
   M=0;
   N=0;
   s=0;
   b=0;
   printf("%d\n",limpia);
   printf("%d\n",limpia1);
   limpia =1;
   limpia1 =1;
              clrscr();
              if (((muestra= fopen("dos2.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
              {
                        perror ("archivo nulo");
                        exit (1);
              }
              resultado= &muestra;
              while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
              {

      while(limpia!=0)
      {
      limpia=fflush(muestra);
      }
      while(limpia1!=0)
      {
      limpia1=fflush(muestra2);
      }
                        buffer [i]=c;
                        c=fgetc (muestra);
                        i++;

                        buffer2 [j]=d;
                        d=fgetc (muestra2);
              j++;
```

```
                    di1=d+1;
                    di2=d+2;
                    df1=d-1;
                    df2=d-2;
          while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
{
                    buffer2 [j]=d;
                      d=fgetc (muestra2);

                      j++;
}


                    while (s<=3 || b<=3)
                    {
                              buffer [i]=c;
                              c=fgetc (muestra);
                              i++;
                      if (c== '$')
                              {
                              s++;
                              }
                              else
                              {
                                        s=0;
                              }
                              if (s==3)
                              {
                                        buffer [i--]= '\0';
                          if (ferror (muestra))
                                        perror ("error");
                                        buffer2 [j--]= '\0';
                                  if (ferror (muestra2))
                                        perror ("error");
                                        fclose (muestra);
                                        fclose (muestra2);
                                        printf("%d,%d\n",M,N);
                    if (M>=1200)
                    {
                                          printf("si\n");

        u++;
                    }
                    else
                    {
                                        printf("no\n");
                                  }

                                  printf("%d\n",limpia);
      printf("%d\n",limpia1);
                                  return;
                                }

   else
   {
                                  buffer2 [j]=d;
                                  d=fgetc (muestra2);
      j++;
                                  if (d== '$')
                                  {
                                        b++;
                                  }
                                  else
                                  {
                                    b=0;
                                  }
                                  if (b==3)
                                  {
                                        buffer [i--]= '\0';
                            if (ferror (muestra))
                                    perror ("error");
                                        buffer2 [j--]= '\0';
        if (ferror (muestra2))
                                        perror ("error");
                                        fclose (muestra);
                                        fclose (muestra2);
                                        printf("%d,%d\n",M,N);
```

```
                                                if (M>=1200)
                                                {
                                                        printf("si\n");

                u++;
                                                }
                                                else
                                                {
                                                        printf("no\n");
                                                }

                                        printf("%d\n",limpia);
                printf("%d\n",limpia1);
                                        return;
                                        }
                else
                {
                rango();
                }
            }
                }
    }
    return;
}

void motori3()
{
    i=0;
    j=0;
    M=0;
    N=0;
    s=0;
    b=0;
    printf("%d\n",limpia);
    printf("%d\n",limpia1);
    limpia =1;
    limpia1 =1;
                clrscr();
                if (((muestra= fopen("dos3.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
                {
                        perror ("archivo nulo");
                        exit (1);
                }
                resultado= &muestra;
                while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
                {

      while(limpia!=0)
      {
      limpia=fflush(muestra);
      }
      while(limpia1!=0)
      {
      limpia1=fflush(muestra2);
      }
                        buffer [i]=c;
                        c=fgetc (muestra);
                        i++;

                        buffer2 [j]=d;
                        d=fgetc (muestra2);
                j++;

                        di1=d+1;
                        di2=d+2;
                        df1=d-1;
                        df2=d-2;
                while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
      {
                        buffer2 [j]=d;
                           d=fgetc (muestra2);

                           j++;
      }


                        while (s<=3 || b<=3)
                        {
```

```
                        buffer [i]=c;
                        c=fgetc (muestra);
                        i++;
                if (c== '$')
                        {
                        s++;
                        }
                        else
                        {
                                s=0;
                        }
                        if (s==3)
                        {
                                buffer [i--]= '\0';
                  if (ferror (muestra))
                                perror ("error");
                                buffer2 [j--]= '\0';
                     if (ferror (muestra2))
                                perror ("error");
                                fclose (muestra);
                                fclose (muestra2);
                                printf("%d,%d\n",M,N);
          if (M>=1200)
          {
                                    printf("si\n");

    u++;
          }
          else
          {
                                    printf("no\n");
                     }

                                printf("%d\n",limpia);
  printf("%d\n",limpia1);
                                return;
                        }

else
{
                                buffer2 [j]=d;
                                d=fgetc (muestra2);
  j++;
                                if (d== '$')
                                {
                                        b++;
                                }
                                else
                                {
                                   b=0;
                                }
                                if (b==3)
                                {
                                        buffer [i--]= '\0';
                        if (ferror (muestra))
                                perror ("error");
                                        buffer2 [j--]= '\0';
    if (ferror (muestra2))
                                        perror ("error");
                                        fclose (muestra);
                                        fclose (muestra2);
                                        printf("%d,%d\n",M,N);
                                        if (M>=1200)
                                        {
                                                printf("si\n");

      u++;
                                        }
                                        else
                                        {
                                                printf("no\n");
                                        }

                                printf("%d\n",limpia);
    printf("%d\n",limpia1);
                                return;
                                }
  else
```

```
            {
            rango();
            }
         }
               }
   }
   return;
}




void motord()
{
   i=0;
   j=0;
   M=0;
   N=0;
   s=0;
   b=0;
   printf("%d\n",limpia);
   printf("%d\n",limpia1);
   limpia =1;
   limpia1 =1;
               clrscr();
               if (((muestra= fopen("nueve.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
               {
                              perror ("archivo nulo");
                              exit (1);
               }
               resultado= &muestra;
               while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
               {

      while(limpia!=0)
      {
      limpia=fflush(muestra);
      }
      while(limpia1!=0)
      {
      limpia1=fflush(muestra2);
      }
                              buffer [i]=c;
                              c=fgetc (muestra);
                              i++;

                              buffer2 [j]=d;
                              d=fgetc (muestra2);
                  j++;

                              di1=d+1;
                              di2=d+2;
                              df1=d-1;
                              df2=d-2;
                  while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
      {
                              buffer2 [j]=d;
                                 d=fgetc (muestra2);

                                 j++;
      }


                              while (s<=3 || b<=3)
                              {
                                          buffer [i]=c;
                                          c=fgetc (muestra);
                                          i++;
                                 if (c== '$')
                                          {
                                          s++;
                                          }
                                          else
                                          {
                                                      s=0;
                                          }
```

```
                                        if (s==3)
                                        {
                                                buffer [i--]= '\0';
                                    if (ferror (muestra))
                                                perror ("error");
                                                buffer2 [j--]= '\0';
                                      if (ferror (muestra2))
                                                perror ("error");
                                                fclose (muestra);
                                                fclose (muestra2);
                                                printf("%d,%d\n",M,N);
                        if (M>=1200)
                        {
                                                    printf("si\n");

                  v++;
                        }
                        else
                        {
                                                    printf("no\n");
                                      }

                                      printf("%d\n",limpia);
            printf("%d\n",limpia1);
                                      return;
                                    }

          else
          {
                                      buffer2 [j]=d;
                                      d=fgetc (muestra2);
            j++;
                                      if (d== '$')
                                      {
                                                b++;
                                      }
                                      else
                                      {
                                         b=0;
                                      }
                                      if (b==3)
                                      {
                                                buffer [i--]= '\0';
                                  if (ferror (muestra))
                                         perror ("error");
                                                buffer2 [j--]= '\0';
              if (ferror (muestra2))
                                                perror ("error");
                                                fclose (muestra);
                                                fclose (muestra2);
                                                printf("%d,%d\n",M,N);
                                                if (M>=1200)
                                                {
                                                        printf("si\n");

                 v++;
                                              }
                                              else
                                              {
                                                        printf("no\n");
                                              }

                                      printf("%d\n",limpia);
              printf("%d\n",limpia1);
                                      return;
                                      }
              else
              {
              rango();
              }
          }
                }
    }
    return;
}

void motord1()
{
```

```
i=0;
j=0;
M=0;
N=0;
s=0;
b=0;
printf("%d\n",limpia);
printf("%d\n",limpia1);
limpia =1;
limpia1 =1;
        clrscr();
        if (((muestra= fopen("nueve1.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
        {
                perror ("archivo nulo");
                exit (1);
        }
        resultado= &muestra;
        while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
        {

  while(limpia!=0)
  {
  limpia=fflush(muestra);
  }
  while(limpia1!=0)
  {
  limpia1=fflush(muestra2);
  }
                buffer [i]=c;
                c=fgetc (muestra);
                i++;

                buffer2 [j]=d;
                d=fgetc (muestra2);
          j++;

                di1=d+1;
                di2=d+2;
                df1=d-1;
                df2=d-2;
          while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
  {
                buffer2 [j]=d;
                  d=fgetc (muestra2);

                  j++;
  }


                while (s<=3 || b<=3)
                {
                        buffer [i]=c;
                        c=fgetc (muestra);
                        i++;
                  if (c== '$')
                        {
                        s++;
                        }
                        else
                        {
                                s=0;
                        }
                        if (s==3)
                        {
                                buffer [i--]= '\0';
                     if (ferror (muestra))
                                perror ("error");
                                buffer2 [j--]= '\0';
                          if (ferror (muestra2))
                                perror ("error");
                                fclose (muestra);
                                fclose (muestra2);
                                printf("%d,%d\n",M,N);
              if (M>=1200)
              {
                                  printf("si\n");

        v++;
```

```
                }
                else
                {
                                printf("no\n");
                }

                printf("%d\n",limpia);
        printf("%d\n",limpia1);
                return;
        }

        else
        {
                buffer2 [j]=d;
                d=fgetc (muestra2);
          j++;
                if (d== '$')
                {
                                b++;
                }
                else
                {
                   b=0;
                }
                if (b==3)
                {
                                buffer [i--]= '\0';
                        if (ferror (muestra))
                           perror ("error");
                                buffer2 [j--]= '\0';
          if (ferror (muestra2))
                                perror ("error");
                                fclose (muestra);
                                fclose (muestra2);
                                printf("%d,%d\n",M,N);
                                if (M>=1200)
                                {
                                        printf("si\n");

            v++;
                                }
                                else
                                {
                                        printf("no\n");
                                }

                printf("%d\n",limpia);
          printf("%d\n",limpia1);
                return;
                }
          else
          {
          rango();
          }
        }
            }
   }
   return;
}

void motord2()
{
   i=0;
   j=0;
   M=0;
   N=0;
   s=0;
   b=0;
   printf("%d\n",limpia);
   printf("%d\n",limpia1);
   limpia =1;
   limpia1 =1;
            clrscr();
            if (((muestra= fopen("nueve2.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
            {
                        perror ("archivo nulo");
                        exit (1);
            }
```

```
        resultado= &muestra;
        while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
        {

while(limpia!=0)
{
limpia=fflush(muestra);
}
while(limpia1!=0)
{
limpia1=fflush(muestra2);
}
                        buffer [i]=c;
                        c=fgetc (muestra);
                        i++;

                        buffer2 [j]=d;
                        d=fgetc (muestra2);
            j++;

                        di1=d+1;
                        di2=d+2;
                        df1=d-1;
                        df2=d-2;
             while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
{
                        buffer2 [j]=d;
                           d=fgetc (muestra2);

                           j++;
}


                        while (s<=3 || b<=3)
                        {
                                    buffer [i]=c;
                                    c=fgetc (muestra);
                                    i++;
                           if (c== '$')
                                    {
                                    s++;
                                    }
                                    else
                                    {
                                                s=0;
                                    }
                                    if (s==3)
                                    {
                                                buffer [i--]= '\0';
                               if (ferror (muestra))
                                                perror ("error");
                                                buffer2 [j--]= '\0';
                                      if (ferror (muestra2))
                                                perror ("error");
                                                fclose (muestra);
                                                fclose (muestra2);
                                                printf("%d,%d\n",M,N);
                     if (M>=1200)
                     {
                                                   printf("si\n");

          v++;
                     }
                     else
                     {
                                                   printf("no\n");
                                      }

                                      printf("%d\n",limpia);
       printf("%d\n",limpia1);
                                      return;
                                    }

     else
     {
                                      buffer2 [j]=d;
                                      d=fgetc (muestra2);
       j++;
```

```
                                        if (d== '$')
                                        {
                                                b++;
                                        }
                                        else
                                        {
                                           b=0;
                                        }
                                        if (b==3)
                                        {
                                                buffer [i--]= '\0';
                                if (ferror (muestra))
                                        perror ("error");
                                                buffer2 [j--]= '\0';
        if (ferror (muestra2))
                                                perror ("error");
                                                fclose (muestra);
                                                fclose (muestra2);
                                                printf("%d,%d\n",M,N);
                                                if (M>=1200)
                                                {
                                                        printf("si\n");

          v++;
                                                }
                                                else
                                                {
                                                        printf("no\n");
                                                }

                                        printf("%d\n",limpia);
        printf("%d\n",limpia1);
                                        return;
                                        }
      else
      {
      rango();
      }
    }
        }
  }
  return;
}

void motord3()
{
  i=0;
  j=0;
  M=0;
  N=0;
  s=0;
  b=0;
  printf("%d\n",limpia);
  printf("%d\n",limpia1);
  limpia =1;
  limpia1 =1;
        clrscr();
        if (((muestra= fopen("nueve3.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
        {
                perror ("archivo nulo");
                exit (1);
        }
        resultado= &muestra;
        while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
        {

  while(limpia!=0)
  {
  limpia=fflush(muestra);
  }
  while(limpia1!=0)
  {
  limpia1=fflush(muestra2);
  }
                buffer [i]=c;
                c=fgetc (muestra);
                i++;
```

```
                buffer2 [j]=d;
                d=fgetc (muestra2);
        j++;

                di1=d+1;
                di2=d+2;
                df1=d-1;
                df2=d-2;
        while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
{
                buffer2 [j]=d;
                  d=fgetc (muestra2);

                  j++;
}


                while (s<=3 || b<=3)
                {
                        buffer [i]=c;
                        c=fgetc (muestra);
                        i++;
                  if (c== '$')
                        {
                        s++;
                        }
                        else
                        {
                                s=0;
                        }
                        if (s==3)
                        {
                                buffer [i--]= '\0';
                     if (ferror (muestra))
                                perror ("error");
                                buffer2 [j--]= '\0';
                          if (ferror (muestra2))
                                perror ("error");
                                fclose (muestra);
                                fclose (muestra2);
                                printf("%d,%d\n",M,N);
            if (M>=1200)
            {
                                    printf("si\n");

      v++;
            }
            else
            {
                                    printf("no\n");
                                }

                                printf("%d\n",limpia);
   printf("%d\n",limpia1);
                                return;
                                }

  else
  {
                                buffer2 [j]=d;
                                d=fgetc (muestra2);
    j++;
                                if (d== '$')
                                {
                                        b++;
                                }
                                else
                                {
                                   b=0;
                                }
                                if (b==3)
                                {
                                        buffer [i--]= '\0';
                             if (ferror (muestra))
                                   perror ("error");
                                        buffer2 [j--]= '\0';
      if (ferror (muestra2))
                                        perror ("error");
```

```
                                                fclose (muestra);
                                                fclose (muestra2);
                                                printf("%d,%d\n",M,N);
                                                if (M>=1200)
                                                {
                                                        printf("si\n");

                v++;
                                                }
                                                else
                                                {
                                                        printf("no\n");
                                                }

                                        printf("%d\n",limpia);
            printf("%d\n",limpia1);
                                        return;
                                        }
            else
            {
            rango();
            }
          }
              }
    }
    return;
}


void velocidad()
{
            t=0;
            salida=0;
            clrscr();
            printf("selecciona la velocidad del motor\n");
            delay(5000);
            while(salida!=1)
            {
                        clrscr();
      printf("lento A  rapido B apaga C\n");
                        delay(5000);
                        O=0;
                        P=0;
                        Q=0;
                        while(O<2&&P<2&&Q<2)
                        {
                                    escribe();
                                    rapido();
                                    rapido1();
                                    rapido2();
                                    lento();
                                    lento1();
                                    lento2();
                                    apaga();
                                    apaga1();
                                    apaga2();
                        }
                        if(O>=2)
                        {
                                    outportb(puerto_vel,rapid);          // SALIDA DE DATOS DE CONTROL
                                    printf("rapido\n");
                                    salida=1;
                                    t=0;
                        }
                        else if(P>=2)
                        {
                                    outportb(puerto_vel,lent);          //SALIDA DE DATOS DE CONTROL
                                    printf("lento\n");
                                    salida=1;
                                    t=0;
                        }
                        else if(Q>=2)
                        {
                                    outportb(puerto_vel,apagado);       //SALIDA DE DATOS DE CONTROL
                                    printf("apaga\n");
                                    salida=1;
                                    t=1;
                        }
```

```
                        else
                        {
                                salida=0;
                                t=0;
                        }
                }
                return;
}

void rapido()
{
   i=0;
   j=0;
   M=0;
   N=0;
   s=0;
   b=0;
   printf("%d\n",limpia);
   printf("%d\n",limpia1);
   limpia =1;
   limpia1 =1;
                clrscr();
                if (((muestra= fopen("mas.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
                {
                        perror ("archivo nulo");
                        exit (1);
                }
                resultado= &muestra;
                while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
                {

    while(limpia!=0)
    {
    limpia=fflush(muestra);
    }
    while(limpia1!=0)
    {
    limpia1=fflush(muestra2);
    }
                        buffer [i]=c;
                        c=fgetc (muestra);
                        i++;

                        buffer2 [j]=d;
                        d=fgetc (muestra2);
              j++;

                        di1=d+1;
                        di2=d+2;
                        df1=d-1;
                        df2=d-2;
              while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
    {
                        buffer2 [j]=d;
                          d=fgetc (muestra2);

                          j++;
    }


                        while (s<=3 || b<=3)
                        {
                                buffer [i]=c;
                                c=fgetc (muestra);
                                i++;
                          if (c== '$')
                                {
                                s++;
                                }
                                else
                                {
                                        s=0;
                                }
                                if (s==3)
                                {
                                        buffer [i--]= '\0';
                             if (ferror (muestra))
                                        perror ("error");
```

```
                                        buffer2 [j--]= '\0';
                                    if (ferror (muestra2))
                                        perror ("error");
                                        fclose (muestra);
                                        fclose (muestra2);
                                        printf("%d,%d\n",M,N);
                    if (M>=1200)
                    {
                                            printf("si\n");

            O++;
                    }
                    else
                    {
                                            printf("no\n");
                                    }

                                    printf("%d\n",limpia);
        printf("%d\n",limpia1);
                                    return;
                                }

    else
    {
                                    buffer2 [j]=d;
                                    d=fgetc (muestra2);
        j++;
                                    if (d== '$')
                                    {
                                        b++;
                                    }
                                    else
                                    {
                                       b=0;
                                    }
                                    if (b==3)
                                    {
                                        buffer [i--]= '\0';
                                if (ferror (muestra))
                                    perror ("error");
                                        buffer2 [j--]= '\0';
            if (ferror (muestra2))
                                        perror ("error");
                                        fclose (muestra);
                                        fclose (muestra2);
                                        printf("%d,%d\n",M,N);
                                        if (M>=1200)
                                        {
                                                printf("si\n");

                O++;
                                    }
                                    else
                                    {
                                            printf("no\n");
                                    }

                                    printf("%d\n",limpia);
            printf("%d\n",limpia1);
                                    return;
                                    }
            else
            {
            rango();
            }
        }
              }
   }
   return;
}

void rapido1()
{
   i=0;
   j=0;
   M=0;
   N=0;
   s=0;
```

```
b=0;
printf("%d\n",limpia);
printf("%d\n",limpia1);
limpia =1;
limpia1 =1;
            clrscr();
            if (((muestra= fopen("mas1.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
            {
                        perror ("archivo nulo");
                        exit (1);
            }
            resultado= &muestra;
            while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
            {

   while(limpia!=0)
   {
   limpia=fflush(muestra);
   }
   while(limpia1!=0)
   {
   limpia1=fflush(muestra2);
   }
                        buffer [i]=c;
                        c=fgetc (muestra);
                        i++;

                        buffer2 [j]=d;
                        d=fgetc (muestra2);
               j++;

                        di1=d+1;
                        di2=d+2;
                        df1=d-1;
                        df2=d-2;
               while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
   {
                        buffer2 [j]=d;
                           d=fgetc (muestra2);

                           j++;
   }


                        while (s<=3 || b<=3)
                        {
                                    buffer [i]=c;
                                    c=fgetc (muestra);
                                    i++;
                           if (c== '$')
                                    {
                                    s++;
                                    }
                                    else
                                    {
                                                s=0;
                                    }
                                    if (s==3)
                                    {
                                                buffer [i--]= '\0';
                              if (ferror (muestra))
                                                perror ("error");
                                                buffer2 [j--]= '\0';
                                       if (ferror (muestra2))
                                                perror ("error");
                                                fclose (muestra);
                                                fclose (muestra2);
                                                printf("%d,%d\n",M,N);
                     if (M>=1200)
                     {
                                                   printf("si\n");

            O++;
                     }
                     else
                     {
                                                   printf("no\n");
                                       }
```

```
                                        printf("%d\n",limpia);
        printf("%d\n",limpia1);
                                        return;
                                    }

      else
      {
                                        buffer2 [j]=d;
                                        d=fgetc (muestra2);
        j++;
                                        if (d== '$')
                                        {
                                                b++;
                                        }
                                        else
                                        {
                                           b=0;
                                        }
                                        if (b==3)
                                        {
                                                buffer [i--]= '\0';
                                    if (ferror (muestra))
                                           perror ("error");
                                                buffer2 [j--]= '\0';
          if (ferror (muestra2))
                                                perror ("error");
                                                fclose (muestra);
                                                fclose (muestra2);
                                                printf("%d,%d\n",M,N);
                                                if (M>=1200)
                                                {
                                                        printf("si\n");

            O++;
                                                }
                                                else
                                                {
                                                        printf("no\n");
                                                }

                                        printf("%d\n",limpia);
          printf("%d\n",limpia1);
                                        return;
                                        }
          else
          {
          rango();
          }
      }
            }
  }
  return;
}

void rapido2()
{
  i=0;
  j=0;
  M=0;
  N=0;
  s=0;
  b=0;
  printf("%d\n",limpia);
  printf("%d\n",limpia1);
  limpia =1;
  limpia1 =1;
          clrscr();
          if (((muestra= fopen("mas2.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
          {
                  perror ("archivo nulo");
                  exit (1);
          }
          resultado= &muestra;
          while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
          {

    while(limpia!=0)
```

```
{
limpia=fflush(muestra);
}
while(limpia1!=0)
{
limpia1=fflush(muestra2);
}
                        buffer [i]=c;
                        c=fgetc (muestra);
                        i++;

                        buffer2 [j]=d;
                        d=fgetc (muestra2);
            j++;

                        di1=d+1;
                        di2=d+2;
                        df1=d-1;
                        df2=d-2;
            while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
{
                        buffer2 [j]=d;
                           d=fgetc (muestra2);

                           j++;
}


                        while (s<=3 || b<=3)
                        {
                                    buffer [i]=c;
                                    c=fgetc (muestra);
                                    i++;
                           if (c== '$')
                                    {
                                    s++;
                                    }
                                    else
                                    {
                                                s=0;
                                    }
                                    if (s==3)
                                    {
                                                buffer [i--]= '\0';
                                 if (ferror (muestra))
                                                perror ("error");
                                                buffer2 [j--]= '\0';
                                       if (ferror (muestra2))
                                                perror ("error");
                                                fclose (muestra);
                                                fclose (muestra2);
                                                printf("%d,%d\n",M,N);
                  if (M>=1200)
                  {
                                                   printf("si\n");

         O++;
                  }
                  else
                  {
                                                   printf("no\n");
                                       }

                                       printf("%d\n",limpia);
      printf("%d\n",limpia1);
                                       return;
                                    }

   else
   {
                                       buffer2 [j]=d;
                                       d=fgetc (muestra2);
      j++;
                                       if (d== '$')
                                       {
                                                b++;
                                       }
                                       else
```

```
                                        {
                                          b=0;
                                        }
                                        if (b==3)
                                        {
                                                buffer [i--]= '\0';
                                    if (ferror (muestra))
                                          perror ("error");
                                                buffer2 [j--]= '\0';
            if (ferror (muestra2))
                                                perror ("error");
                                                fclose (muestra);
                                                fclose (muestra2);
                                                printf("%d,%d\n",M,N);
                                                if (M>=1200)
                                                {
                                                        printf("si\n");

               O++;
                                                }
                                                else
                                                {
                                                        printf("no\n");
                                                }

                                        printf("%d\n",limpia);
            printf("%d\n",limpia1);
                                        return;
                                        }
          else
          {
          rango();
          }
        }
            }
   }
   return;
}

void lento()
{
   i=0;
   j=0;
   M=0;
   N=0;
   s=0;
   b=0;
   printf("%d\n",limpia);
   printf("%d\n",limpia1);
   limpia =1;
   limpia1 =1;
            clrscr();
            if (((muestra= fopen("menos.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
            {
                        perror ("archivo nulo");
                        exit (1);
            }
            resultado= &muestra;
            while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
            {

    while(limpia!=0)
    {
    limpia=fflush(muestra);
    }
    while(limpia1!=0)
    {
    limpia1=fflush(muestra2);
    }
                        buffer [i]=c;
                        c=fgetc (muestra);
                        i++;

                        buffer2 [j]=d;
                        d=fgetc (muestra2);
            j++;

                        di1=d+1;
```

```
                    di2=d+2;
                    df1=d-1;
                    df2=d-2;
          while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
{
                    buffer2 [j]=d;
                      d=fgetc (muestra2);

                      j++;
}


                    while (s<=3 || b<=3)
                    {
                              buffer [i]=c;
                              c=fgetc (muestra);
                              i++;
                      if (c== '$')
                              {
                              s++;
                              }
                              else
                              {
                                        s=0;
                              }
                              if (s==3)
                              {
                                        buffer [i--]= '\0';
                          if (ferror (muestra))
                                        perror ("error");
                                        buffer2 [j--]= '\0';
                                  if (ferror (muestra2))
                                        perror ("error");
                                        fclose (muestra);
                                        fclose (muestra2);
                                        printf("%d,%d\n",M,N);
                if (M>=1200)
                {
                                           printf("si\n");

        P++;
                }
                else
                {
                                           printf("no\n");
                                  }

                                  printf("%d\n",limpia);
     printf("%d\n",limpia1);
                                  return;
                              }

   else
   {
                                  buffer2 [j]=d;
                                  d=fgetc (muestra2);
     j++;
                                  if (d== '$')
                                  {
                                        b++;
                                  }
                                  else
                                  {
                                     b=0;
                                  }
                                  if (b==3)
                                  {
                                        buffer [i--]= '\0';
                            if (ferror (muestra))
                                     perror ("error");
                                        buffer2 [j--]= '\0';
        if (ferror (muestra2))
                                        perror ("error");
                                        fclose (muestra);
                                        fclose (muestra2);
                                        printf("%d,%d\n",M,N);
                                        if (M>=1200)
                                        {
```

```
                                                                printf("si\n");

                P++;
                                                        }
                                                        else
                                                        {
                                                                printf("no\n");
                                                        }

                                                printf("%d\n",limpia);
                printf("%d\n",limpia1);
                                                return;
                                                }
        else
        {
        rango();
        }
      }
          }
  }
  return;
}

void lento1()
{
  i=0;
  j=0;
  M=0;
  N=0;
  s=0;
  b=0;
  printf("%d\n",limpia);
  printf("%d\n",limpia1);
  limpia =1;
  limpia1 =1;
          clrscr();
          if (((muestra= fopen("menos1.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
          {
                  perror ("archivo nulo");
                  exit (1);
          }
          resultado= &muestra;
          while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
          {

    while(limpia!=0)
    {
    limpia=fflush(muestra);
    }
    while(limpia1!=0)
    {
    limpia1=fflush(muestra2);
    }
                  buffer [i]=c;
                  c=fgetc (muestra);
                  i++;

                  buffer2 [j]=d;
                  d=fgetc (muestra2);
          j++;

                  di1=d+1;
                  di2=d+2;
                  df1=d-1;
                  df2=d-2;
          while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
    {
                  buffer2 [j]=d;
                    d=fgetc (muestra2);

                    j++;
    }


                  while (s<=3 || b<=3)
                  {
                          buffer [i]=c;
                          c=fgetc (muestra);
```

```
                        i++;
                    if (c== '$')
                        {
                        s++;
                        }
                        else
                        {
                                s=0;
                        }
                        if (s==3)
                        {
                                buffer [i--]= '\0';
                      if (ferror (muestra))
                                perror ("error");
                                buffer2 [j--]= '\0';
                          if (ferror (muestra2))
                                perror ("error");
                                fclose (muestra);
                                fclose (muestra2);
                                printf("%d,%d\n",M,N);
            if (M>=1200)
            {
                                    printf("si\n");

    P++;
            }
            else
            {
                                    printf("no\n");
                          }

                          printf("%d\n",limpia);
  printf("%d\n",limpia1);
                          return;
                        }

else
{
                          buffer2 [j]=d;
                          d=fgetc (muestra2);
  j++;
                          if (d== '$')
                          {
                                b++;
                          }
                          else
                          {
                             b=0;
                          }
                          if (b==3)
                          {
                                buffer [i--]= '\0';
                      if (ferror (muestra))
                             perror ("error");
                                buffer2 [j--]= '\0';
    if (ferror (muestra2))
                                perror ("error");
                                fclose (muestra);
                                fclose (muestra2);
                                printf("%d,%d\n",M,N);
                                if (M>=1200)
                                {
                                        printf("si\n");

      P++;
                                }
                                else
                                {
                                        printf("no\n");
                                }

                          printf("%d\n",limpia);
    printf("%d\n",limpia1);
                          return;
                          }
  else
  {
  rango();
```

```
            }
        }
            }
    }
    return;
}

void lento2()
{
    i=0;
    j=0;
    M=0;
    N=0;
    s=0;
    b=0;
    printf("%d\n",limpia);
    printf("%d\n",limpia1);
    limpia =1;
    limpia1 =1;
            clrscr();
            if (((muestra= fopen("menos2.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
            {
                    perror ("archivo nulo");
                    exit (1);
            }
            resultado= &muestra;
            while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
            {

    while(limpia!=0)
    {
    limpia=fflush(muestra);
    }
    while(limpia1!=0)
    {
    limpia1=fflush(muestra2);
    }
                    buffer [i]=c;
                    c=fgetc (muestra);
                    i++;

                    buffer2 [j]=d;
                    d=fgetc (muestra2);
            j++;

                    di1=d+1;
                    di2=d+2;
                    df1=d-1;
                    df2=d-2;
             while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
    {
                    buffer2 [j]=d;
                      d=fgetc (muestra2);

                      j++;
    }


                    while (s<=3 || b<=3)
                    {
                            buffer [i]=c;
                            c=fgetc (muestra);
                            i++;
                      if (c== '$')
                            {
                            s++;
                            }
                            else
                            {
                                    s=0;
                            }
                            if (s==3)
                            {
                                    buffer [i--]= '\0';
                        if (ferror (muestra))
                                    perror ("error");
                                    buffer2 [j--]= '\0';
                                if (ferror (muestra2))
```

```
                                        perror ("error");
                                        fclose (muestra);
                                        fclose (muestra2);
                                        printf("%d,%d\n",M,N);
                if (M>=1200)
                {
                                            printf("si\n");

            P++;
                }
                else
                {
                                        printf("no\n");
                                    }

                                    printf("%d\n",limpia);
        printf("%d\n",limpia1);
                                    return;
                                }

        else
        {
                                    buffer2 [j]=d;
                                    d=fgetc (muestra2);
          j++;
                                    if (d== '$')
                                    {
                                        b++;
                                    }
                                    else
                                    {
                                       b=0;
                                    }
                                    if (b==3)
                                    {
                                        buffer [i--]= '\0';
                            if (ferror (muestra))
                                perror ("error");
                                        buffer2 [j--]= '\0';
            if (ferror (muestra2))
                                        perror ("error");
                                        fclose (muestra);
                                        fclose (muestra2);
                                        printf("%d,%d\n",M,N);
                                        if (M>=1200)
                                        {
                                                printf("si\n");

              P++;
                                    }
                                    else
                                    {
                                            printf("no\n");
                                    }

                                    printf("%d\n",limpia);
            printf("%d\n",limpia1);
                                    return;
                                    }
            else
            {
            rangou();
            }
        }
              }
    }
    return;
}

void apaga()
{
    i=0;
    j=0;
    M=0;
    N=0;
    s=0;
    b=0;
    printf("%d\n",limpia);
```

```
printf("%d\n",limpia1);
limpia =1;
limpia1 =1;
        clrscr();
        if (((muestra= fopen("apaga.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
        {
                perror ("archivo nulo");
                exit (1);
        }
        resultado= &muestra;
        while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
        {

  while(limpia!=0)
  {
  limpia=fflush(muestra);
  }
  while(limpia1!=0)
  {
  limpia1=fflush(muestra2);
  }
                buffer [i]=c;
                c=fgetc (muestra);
                i++;

                buffer2 [j]=d;
                d=fgetc (muestra2);
          j++;

                di1=d+1;
                di2=d+2;
                df1=d-1;
                df2=d-2;
          while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
  {
                buffer2 [j]=d;
                  d=fgetc (muestra2);

                  j++;
  }


                while (s<=3 || b<=3)
                {
                        buffer [i]=c;
                        c=fgetc (muestra);
                        i++;
                  if (c== '$')
                        {
                        s++;
                        }
                        else
                        {
                                s=0;
                        }
                        if (s==3)
                        {
                                buffer [i--]= '\0';
                     if (ferror (muestra))
                                perror ("error");
                                buffer2 [j--]= '\0';
                          if (ferror (muestra2))
                                perror ("error");
                                fclose (muestra);
                                fclose (muestra2);
                                printf("%d,%d\n",M,N);
            if (M>=1200)
            {
                                  printf("si\n");

      Q++;
            }
            else
            {
                                  printf("no\n");
                          }

                          printf("%d\n",limpia);
```

```
        printf("%d\n",limpia1);
                                        return;
                                    }

      else
      {
                                        buffer2 [j]=d;
                                        d=fgetc (muestra2);
        j++;
                                        if (d== '$')
                                        {
                                                b++;
                                        }
                                        else
                                        {
                                           b=0;
                                        }
                                        if (b==3)
                                        {
                                                buffer [i--]= '\0';
                                if (ferror (muestra))
                                           perror ("error");
                                                buffer2 [j--]= '\0';
          if (ferror (muestra2))
                                                perror ("error");
                                                fclose (muestra);
                                                fclose (muestra2);
                                                printf("%d,%d\n",M,N);
                                                if (M>=1200)
                                                {
                                                        printf("si\n");

            Q++;
                                                }
                                                else
                                                {
                                                        printf("no\n");
                                                }

                                        printf("%d\n",limpia);
          printf("%d\n",limpia1);
                                        return;
                                        }
          else
          {
          rango();
          }
      }
          }
  }
  return;
}

void apaga1()
{
  i=0;
  j=0;
  M=0;
  N=0;
  s=0;
  b=0;
  printf("%d\n",limpia);
  printf("%d\n",limpia1);
  limpia =1;
  limpia1 =1;
          clrscr();
          if (((muestra= fopen("apaga1.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
          {
                  perror ("archivo nulo");
                  exit (1);
          }
          resultado= &muestra;
          while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
          {

    while(limpia!=0)
    {
    limpia=fflush(muestra);
```

```
}
while(limpia1!=0)
{
limpia1=fflush(muestra2);
}
                        buffer [i]=c;
                        c=fgetc (muestra);
                        i++;

                        buffer2 [j]=d;
                        d=fgetc (muestra2);
            j++;

                        di1=d+1;
                        di2=d+2;
                        df1=d-1;
                        df2=d-2;
            while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
{
                        buffer2 [j]=d;
                           d=fgetc (muestra2);

                           j++;
}


                        while (s<=3 || b<=3)
                        {
                                    buffer [i]=c;
                                    c=fgetc (muestra);
                                    i++;
                           if (c== '$')
                                    {
                                    s++;
                                    }
                                    else
                                    {
                                                s=0;
                                    }
                                    if (s==3)
                                    {
                                                buffer [i--]= '\0';
                                 if (ferror (muestra))
                                                perror ("error");
                                                buffer2 [j--]= '\0';
                                       if (ferror (muestra2))
                                                perror ("error");
                                                fclose (muestra);
                                                fclose (muestra2);
                                                printf("%d,%d\n",M,N);
                  if (M>=1200)
                  {
                                                   printf("si\n");

          Q++;
                  }
                  else
                  {
                                                   printf("no\n");
                                       }

                                       printf("%d\n",limpia);
      printf("%d\n",limpia1);
                                       return;
                                    }

    else
    {
                                       buffer2 [j]=d;
                                       d=fgetc (muestra2);
      j++;
                                       if (d== '$')
                                       {
                                                   b++;
                                       }
                                       else
                                       {
                                          b=0;
```

```
                                        }
                                        if (b==3)
                                        {
                                                buffer [i--]= '\0';
                                if (ferror (muestra))
                                        perror ("error");
                                                buffer2 [j--]= '\0';
        if (ferror (muestra2))
                                                perror ("error");
                                                fclose (muestra);
                                                fclose (muestra2);
                                                printf("%d,%d\n",M,N);
                                                if (M>=1200)
                                                {
                                                        printf("si\n");

          Q++;
                                                }
                                                else
                                                {
                                                        printf("no\n");
                                                }

                                        printf("%d\n",limpia);
        printf("%d\n",limpia1);
                                        return;
                                        }
          else
          {
          rango();
          }
        }
          }
   }
   return;
}

void apaga2()
{
   i=0;
   j=0;
   M=0;
   N=0;
   s=0;
   b=0;
   printf("%d\n",limpia);
   printf("%d\n",limpia1);
   limpia =1;
   limpia1 =1;
          clrscr();
          if (((muestra= fopen("apaga2.txt", "r"))==NULL) || ((muestra2= fopen("71.txt","r"))==NULL))
          {
                  perror ("archivo nulo");
                  exit (1);
          }
          resultado= &muestra;
          while ((!ferror (muestra) && !feof(muestra)) || (!ferror (muestra2) && !feof(muestra2)))
          {

   while(limpia!=0)
   {
   limpia=fflush(muestra);
   }
   while(limpia1!=0)
   {
   limpia1=fflush(muestra2);
   }
                  buffer [i]=c;
                  c=fgetc (muestra);
                  i++;

                  buffer2 [j]=d;
                  d=fgetc (muestra2);
          j++;

                  di1=d+1;
                  di2=d+2;
                  df1=d-1;
```

```
                df2=d-2;
        while  (c!=d&&c!=di1&&c!=di2&&c!=df1&&c!=df2)
{
                buffer2 [j]=d;
                  d=fgetc (muestra2);

                  j++;
}


                while (s<=3 || b<=3)
                {
                        buffer [i]=c;
                        c=fgetc (muestra);
                        i++;
                  if (c== '$')
                        {
                        s++;
                        }
                        else
                        {
                                s=0;
                        }
                        if (s==3)
                        {
                                buffer [i--]= '\0';
                     if (ferror (muestra))
                                perror ("error");
                                buffer2 [j--]= '\0';
                          if (ferror (muestra2))
                                perror ("error");
                                fclose (muestra);
                                fclose (muestra2);
                                printf("%d,%d\n",M,N);
            if (M>=1200)
            {
                                  printf("si\n");

      Q++;
            }
            else
            {
                                  printf("no\n");
                          }

                          printf("%d\n",limpia);
   printf("%d\n",limpia1);
                          return;
                        }

  else
  {
                          buffer2 [j]=d;
                          d=fgetc (muestra2);
    j++;
                          if (d== '$')
                          {
                                b++;
                          }
                          else
                          {
                             b=0;
                          }
                          if (b==3)
                          {
                                buffer [i--]= '\0';
                     if (ferror (muestra))
                             perror ("error");
                                buffer2 [j--]= '\0';
      if (ferror (muestra2))
                                perror ("error");
                                fclose (muestra);
                                fclose (muestra2);
                                printf("%d,%d\n",M,N);
                                if (M>=1200)
                                {
                                        printf("si\n");
```

```
                Q++;
                                                }
                                                else
                                                {
                                                        printf("no\n");
                                                }

                                        printf("%d\n",limpia);
            printf("%d\n",limpia1);
                                        return;
                                        }
            else
            {
            rango();
            }
          }
              }
   }
   return;
}

void rango()
{
                                                ci=c;
                                                ci5=c-5;
                                                ci4=c-4;
                                                ci3=c-3;
                                                ci2=c-2;
                                                ci1=c-1;
                                                cs5=c+5;
                                                cs4=c+4;
                                                cs3=c+3;
                                                cs2=c+2;
                                                cs1=c+1;
                                                cs6=c+6;
                                                ci6=c-6;
                                                ci7=c-7;
                                                ci8=c-8;
                                                ci9=c-9;
                                                cs7=c+7;
                                                cs8=c+8;
                                                cs9=c+9;

                                                if (cs6==d|| ci6==d|| ci==d|| ci1==d|| ci2==d || ci3==d || ci4==d ||ci5==d ||cs1==d ||cs2==d ||cs3==d
||cs4==d ||cs5==d ||cs7==d||cs8==d||cs9==d||ci7==d||ci8==d||ci9==d)
                                                {
                                                        e=d;
                                                   M++;
                                                }
                                                else
                                                {
                                                        N++;
                                                }
            return;
}
```

APENDICE C

Este apéndice incluye las hojas de especificaciones técnicas de los componentes de cada una de las etapas que integran el sistema.

National Semiconductor

August 2000

LM837
Low Noise Quad Operational Amplifier

General Description

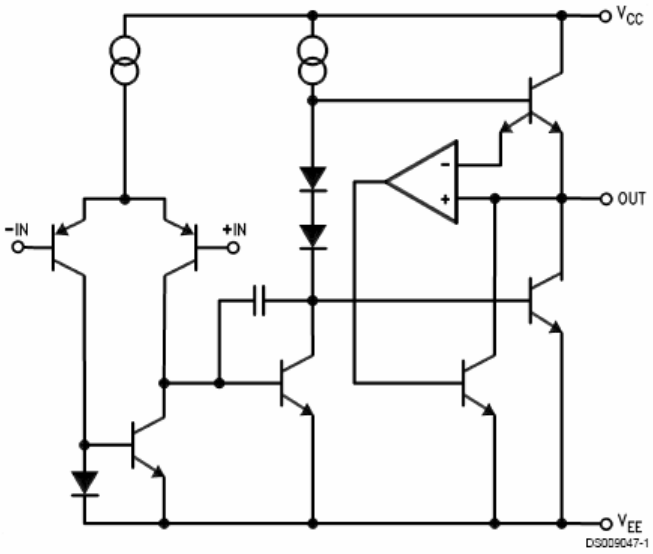
The LM837 is a quad operational amplifier designed for low noise, high speed and wide bandwidth performance. It has a new type of output stage which can drive a 600Ω load, making it ideal for almost all digital audio, graphic equalizer, preamplifiers, and professional audio applications. Its high performance characteristics also make it suitable for instrumentation applications where low noise is the key consideration.

The LM837 is internally compensated for unity gain operation. It is pin compatible with most other standard quad op amps and can therefore be used to upgrade existing systems with little or no change.

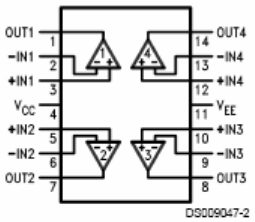
Features

■ High slew rate	10 V/μs (typ); 8 V/μs (min)
■ Wide gain bandwidth product (min)	25 MHz (typ); 15 MHz
■ Power bandwidth	200 kHz (typ)
■ High output current	±40 mA
■ Excellent output drive performance	>600Ω
■ Low input noise voltage	4.5 nV/$\sqrt{Hz}$
■ Low total harmonic distortion	0.0015%
■ Low offset voltage	0.3 mV

Schematic and Connection Diagrams



Dual-In-Line Package



Top View
Order Number LM837M, LM837MX or LM837N
See NS Package Number M14A or N14A

XR-2207
Voltage-Controlled Oscillator

EXAR ...the analog plus company™

June 1997–3

FEATURES

- Excellent Temperature Stability (20ppm/°C)
- Linear Frequency Sweep
- Adjustable Duty Cycle (0.1% to 99.9%)
- Two or Four Level FSK Capability
- Wide Sweep Range (1000:1 Minimum)
- Logic Compatible Input and Output Levels
- Wide Supply Voltage Range (±4V to ±13V)
- Low Supply Sensitivity (0.1% /V)
- Wide Frequency Range (0.01Hz to 1MHz)
- Simultaneous Triangle and Squarewave Outputs

APPLICATIONS

- FSK Generation
- Voltage and Current-to-Frequency Conversion
- Stable Phase-Locked Loop
- Waveform Generation
  - Triangle, Sawtooth, Pulse, Squarewave
- FM and Sweep Generation

GENERAL DESCRIPTION

The XR-2207 is a monolithic voltage-controlled oscillator (VCO) integrated circuit featuring excellent frequency stability and a wide tuning range. The circuit provides simultaneous triangle and squarewave outputs over a frequency range of 0.01Hz to 1MHz. It is ideally suited for FM, FSK, and sweep or tone generation, as well as for phase-locked loop applications.

The XR-2207 has a typical drift specification of 20ppm/°C. The oscillator frequency can be linearly swept over a 1000:1 range with an external control voltage; and the duty cycle of both the triangle and the squarewave outputs can be varied from 0.1% to 99.9% to generate stable pulse and sawtooth waveforms.

ORDERING INFORMATION

Part No.	Package	Operating Temperature Range
XR-2207M	14 Lead 300 Mil CDIP	-55°C to +125°C
XR-2207CP	14 Lead 300 Mil PDIP	0°C to +70°C
XR-2207D	16 Lead 300 Mil JEDEC SOIC	0°C to +70°C
XR-2207ID	16 Lead 300 Mil JEDEC SOIC	-40°C to +85°C

BLOCK DIAGRAM

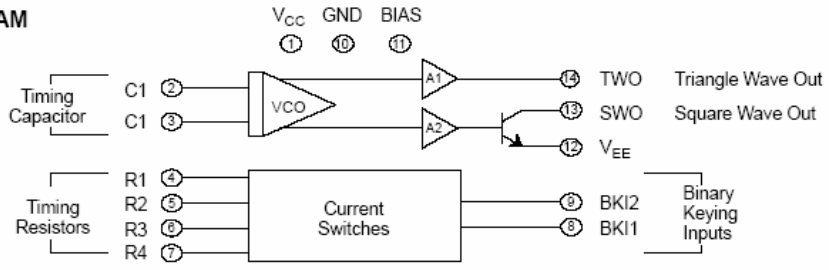


Figure 1. Block Diagram

Rev. 2.02

EXAR Corporation, 48720 Kato Road, Fremont, CA 94538 ◆ (510) 668-7000 ◆ FAX (510) 668-7017

HARRIS Semiconductor

82C55A

CMOS Programmable Peripheral Interface

June 1998

Features

- Pin Compatible with NMOS 8255A
- 24 Programmable I/O Pins
- Fully TTL Compatible
- High Speed, No "Wait State" Operation with 5MHz and 8MHz 80C86 and 80C88
- Direct Bit Set/Reset Capability
- Enhanced Control Word Read Capability
- L7 Process
- 2.5mA Drive Capability on All I/O Ports
- Low Standby Power (ICCSB) 10µA

Ordering Information

PART NUMBERS		PACKAGE	TEMPERATURE RANGE	PKG. NO.
5MHz	8MHz			
CP82C55A-5	CP82C55A	40 Ld PDIP	0°C to 70°C	E40.6
IP82C55A-5	IP82C55A		-40°C to 85°C	E40.6
CS82C55A-5	CS82C55A	44 Ld PLCC	0°C to 70°C	N44.65
IS82C55A-5	IS82C55A		-40°C to 85°C	N44.65
CD82C55A-5	CD82C55A	40 Ld CERDIP	0°C to 70°C	F40.6
ID82C55A-5	ID82C55A		-40°C to 85°C	F40.6
MD82C55A-5/B	MD82C55A/B		-55°C to 125°C	F40.6
8406601QA	8406602QA	SMD#		F40.6
MR82C55A-5/B	MR82C55A/B	44 Pad CLCC	-55°C to 125°C	J44.A
8406601XA	8406602XA	SMD#		J44.A

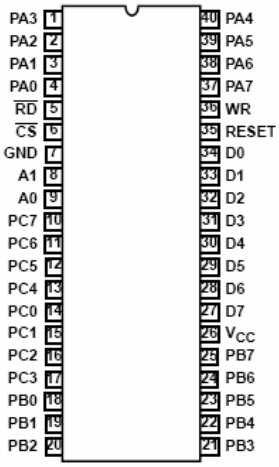
Description

The Harris 82C55A is a high performance CMOS version of the industry standard 8255A and is manufactured using a self-aligned silicon gate CMOS process (Scaled SAJI IV). It is a general purpose programmable I/O device which may be used with many different microprocessors. There are 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. The high performance and industry standard configuration of the 82C55A make it compatible with the 80C86, 80C88 and other microprocessors.

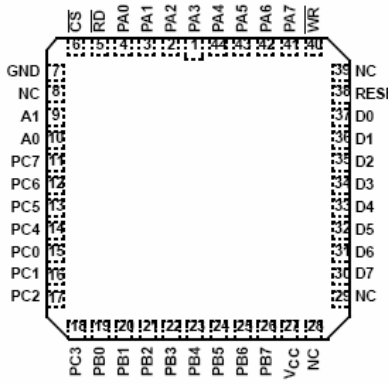
Static CMOS circuit design insures low operating power. TTL compatibility over the full military temperature range and bus hold circuitry eliminate the need for pull-up resistors. The Harris advanced SAJI process results in performance equal to or greater than existing functionally equivalent products at a fraction of the power.

Pinouts

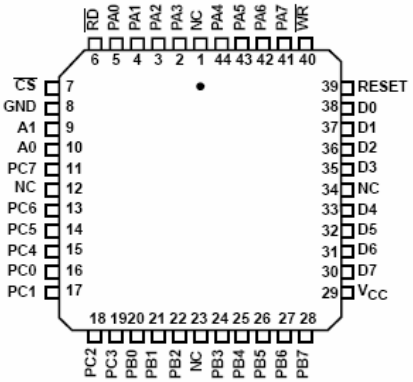
82C55A (DIP)
TOP VIEW



82C55A (CLCC)
TOP VIEW



82C55A (PLCC)
TOP VIEW



CAUTION: These devices are sensitive to electrostatic discharge. Users should follow proper IC Handling Procedures.

File Number **2969.2**

82C55A

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance: Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.

The mode definitions and possible mode combinations may seem confusing at first, but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 82C55A has taken into account things such as efficient PC board layout, control signal definition vs. PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

Single Bit Set/Reset Feature (Figure 5)

Any of the eight bits of Port C can be Set or Reset using a single Output instruction. This feature reduces software requirements in control-based applications.

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were output ports.

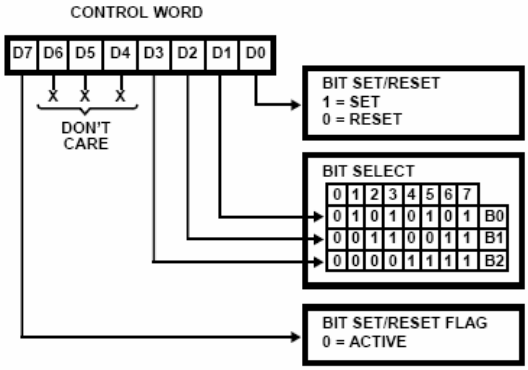


FIGURE 5. BIT SET/RESET FORMAT

Interrupt Control Functions

When the 82C55A is programmed to operate in mode 1 or mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the bit set/reset function of port C.

This function allows the programmer to enable or disable a CPU interrupt by a specific I/O device without affecting any other device in the interrupt structure.

INTE Flip-Flop Definition

(BIT-SET)-INTE is SET - Interrupt Enable

(BIT-RESET)-INTE is Reset - Interrupt Disable

NOTE: All Mask flip-flops are automatically reset during mode selection and device Reset.

Operating Modes

Mode 0 (Basic Input/Output). This functional configuration provides simple input and output operations for each of the three ports. No handshaking is required, data is simply written to or read from a specific port.

Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports
- Any Port can be input or output
- Outputs are latched
- Input are not latched
- 16 different Input/Output configurations possible

MODE 0 PORT DEFINITION

A		B		GROUP A			GROUP B	
D4	D3	D1	D0	PORT A	PORT C (Upper)	#	PORT B	PORT C (Lower)
0	0	0	0	Output	Output	0	Output	Output
0	0	0	1	Output	Output	1	Output	Input
0	0	1	0	Output	Output	2	Input	Output
0	0	1	1	Output	Output	3	Input	Input
0	1	0	0	Output	Input	4	Output	Output
0	1	0	1	Output	Input	5	Output	Input
0	1	1	0	Output	Input	6	Input	Output
0	1	1	1	Output	Input	7	Input	Input
1	0	0	0	Input	Output	8	Output	Output
1	0	0	1	Input	Output	9	Output	Input
1	0	1	0	Input	Output	10	Input	Output
1	0	1	1	Input	Output	11	Input	Input
1	1	0	0	Input	Input	12	Output	Output
1	1	0	1	Input	Input	13	Output	Input
1	1	1	0	Input	Input	14	Input	Output
1	1	1	1	Input	Input	15	Input	Input

National Semiconductor

June 1999

ADC0820
8-Bit High Speed µP Compatible A/D Converter with Track/Hold Function

General Description

By using a half-flash conversion technique, the 8-bit ADC0820 CMOS A/D offers a 1.5 µs conversion time and dissipates only 75 mW of power. The half-flash technique consists of 32 comparators, a most significant 4-bit ADC and a least significant 4-bit ADC.

The input to the ADC0820 is tracked and held by the input sampling circuitry eliminating the need for an external sample-and-hold for signals moving at less than 100 mV/µs.

For ease of interface to microprocessors, the ADC0820 has been designed to appear as a memory location or I/O port without the need for external interfacing logic.

Key Specifications

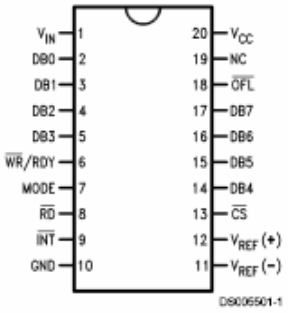
- Resolution — 8 Bits
- Conversion Time — 2.5 µs Max (RD Mode); 1.5 µs Max (WR-RD Mode)
- Low Power — 75 mW Max
- Total Unadjusted Error — ±½ LSB and ± 1 LSB

Features

- Built-in track-and-hold function
- No missing codes
- No external clocking
- Single supply — 5 V_{DC}
- Easy interface to all microprocessors, or operates stand-alone
- Latched TRI-STATE® output
- Logic inputs and outputs meet both MOS and T^2L voltage level specifications
- Operates ratiometrically or with any reference value equal to or less than V_{CC}
- 0V to 5V analog input voltage range with single 5V supply
- No zero or full-scale adjust required
- Overflow output available for cascading
- 0.3" standard width 20-pin DIP
- 20-pin molded chip carrier package
- 20-pin small outline package
- 20-pin shrink small outline package (SSOP)

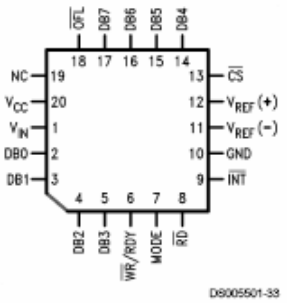
Connection and Functional Diagrams

Dual-In-Line, Small Outline and SSOP Packages



Top View

Molded Chip Carrier Package



TRI-STATE® is a registered trademark of National Semiconductor Corporation.

National Semiconductor

May 2000

LF155/LF156/LF355/LF356/LF357 JFET Input Operational Amplifiers

General Description

These are the first monolithic JFET input operational amplifiers to incorporate well matched, high voltage JFETs on the same chip with standard bipolar transistors (BI-FET™ Technology). These amplifiers feature low input bias and offset currents/low offset voltage and offset voltage drift, coupled with offset adjust which does not degrade drift or common-mode rejection. The devices are also designed for high slew rate, wide bandwidth, extremely fast settling time, low voltage and current noise and a low 1/f noise corner.

Features

Advantages

- Replace expensive hybrid and module FET op amps
- Rugged JFETs allow blow-out free handling compared with MOSFET input devices
- Excellent for low noise applications using either high or low source impedance — very low 1/f corner
- Offset adjust does not degrade drift or common-mode rejection as in most monolithic amplifiers
- New output stage allows use of large capacitive loads (5,000 pF) without stability problems
- Internal compensation and large differential input voltage capability

Applications

- Precision high speed integrators
- Fast D/A and A/D converters
- High impedance buffers
- Wideband, low noise, low drift amplifiers
- Logarithmic amplifiers
- Photocell amplifiers
- Sample and Hold circuits

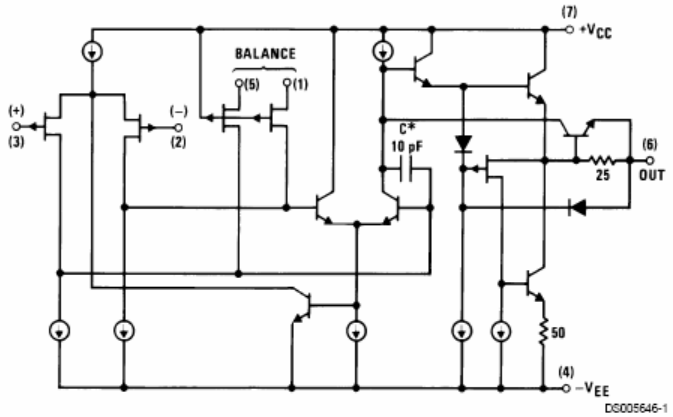
Common Features

- Low input bias current: 30pA
- Low Input Offset Current: 3pA
- High input impedance: $10^{12}\Omega$
- Low input noise current: $0.01\ pA/\sqrt{Hz}$
- High common-mode rejection ratio: 100 dB
- Large dc voltage gain: 106 dB

Uncommon Features

	LF155/ LF355	LF156/ LF356	LF357 (A_V=5)	Units
■ Extremely fast settling time to 0.01%	4	1.5	1.5	µs
■ Fast slew rate	5	12	50	V/µs
■ Wide gain bandwidth	2.5	5	20	MHz
■ Low input noise voltage	20	12	12	$nV/\sqrt{Hz}$

Simplified Schematic



*3 pF in LF357 series.

BI-FET™, BI-FET II™ are trademarks of National Semiconductor Corporation.

Microsemi
21201 Itasca St.
Chatsworth, CA 91311
Phone: (818) 701-4933
Fax: (818) 701-4939

1N4933 thru 1N4937

1 Amp Fast Recovery Rectifier 50 - 600 Volts

Features

- Low Leakage Current
- Metallurgical Bonded Construction
- Low Cost
- Fast Switching

Maximum Ratings

- Operating Temperature: -65°C to +150°C
- Storage Temperature: -65°C to +150°C
- Maximum Thermal Resistance; 30°C/W Junction To Lead

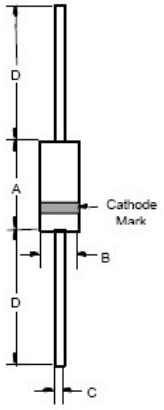
Microsemi Part Number	Maximum Recurrent Peak Reverse Voltage	Maximum RMS Voltage	Maximum DC Blocking Voltage
1N4933	50V	35V	50V
1N4934	100V	70V	100V
1N4935	200V	140V	200V
1N4936	400V	280V	400V
1N4937	600V	420V	600V

Electrical Characteristics @ 25°C Unless Otherwise Specified

Average Forward Current	$I_{F(AV)}$	1.0A	T_A =55°C
Peak Forward Surge Current	I_{FSM}	30A	8.3ms, half sine
Maximum Instantaneous Forward Voltage	V_F	1.2V	I_{FM} = 1.0A; T_J = 25°C*
Maximum DC Reverse Current At Rated DC Blocking Voltage	I_R	5.0µA 100µA	T_J = 25°C T_J = 125°C
Maximum Reverse Recovery Time	T_{rr}	200ns	I_F=0.5A, I_R=1.0A, I_{rr}=0.25A
Typical Junction Capacitance	C_J	15pF	Measured at 1.0MHz, V_R=4.0V

*Pulse test: Pulse width 300 µsec, Duty cycle 1%

DO-41



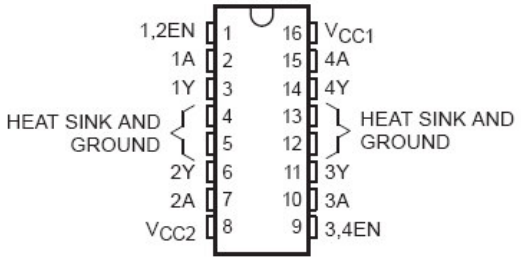
DIMENSIONS					
	INCHES		MM		
DIM	MIN	MAX	MIN	MAX	NOTE
A	.166	.205	4.10	5.20	
B	.080	.107	2.00	2.70	
C	.028	.034	.70	.90	
D	1.000	---	25.40	---	

L293, L293D
QUADRUPLE HALF-H DRIVERS

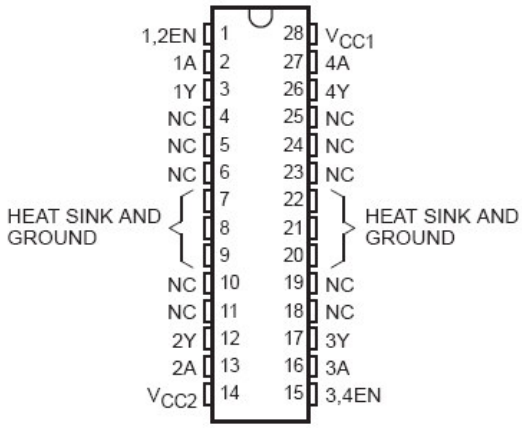
SLRS008C – SEPTEMBER 1986 – REVISED NOVEMBER 2004

- **Featuring Unitrode L293 and L293D Products Now From Texas Instruments**
- **Wide Supply-Voltage Range: 4.5 V to 36 V**
- **Separate Input-Logic Supply**
- **Internal ESD Protection**
- **Thermal Shutdown**
- **High-Noise-Immunity Inputs**
- **Functionally Similar to SGS L293 and SGS L293D**
- **Output Current 1 A Per Channel (600 mA for L293D)**
- **Peak Output Current 2 A Per Channel (1.2 A for L293D)**
- **Output Clamp Diodes for Inductive Transient Suppression (L293D)**

L293 . . . N OR NE PACKAGE
L293D . . . NE PACKAGE
(TOP VIEW)



L293 . . . DWP PACKAGE
(TOP VIEW)



description/ordering information

The L293 and L293D are quadruple high-current half-H drivers. The L293 is designed to provide bidirectional drive currents of up to 1 A at voltages from 4.5 V to 36 V. The L293D is designed to provide bidirectional drive currents of up to 600-mA at voltages from 4.5 V to 36 V. Both devices are designed to drive inductive loads such as relays, solenoids, dc and bipolar stepping motors, as well as other high-current/high-voltage loads in positive-supply applications.

All inputs are TTL compatible. Each output is a complete totem-pole drive circuit, with a Darlington transistor sink and a pseudo-Darlington source. Drivers are enabled in pairs, with drivers 1 and 2 enabled by 1,2EN and drivers 3 and 4 enabled by 3,4EN. When an enable input is high, the associated drivers are enabled, and their outputs are active and in phase with their inputs. When the enable input is low, those drivers are disabled, and their outputs are off and in the high-impedance state. With the proper data inputs, each pair of drivers forms a full-H (or bridge) reversible drive suitable for solenoid or motor applications.

ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	HSOP (DWP)	Tube of 20	L293DWP	L293DWP
	PDIP (N)	Tube of 25	L293N	L293N
	PDIP (NE)	Tube of 25	L293NE	L293NE
		Tube of 25	L293DNE	L293DNE

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

National Semiconductor

August 2000

LM124/LM224/LM324/LM2902
Low Power Quad Operational Amplifiers

General Description

The LM124 series consists of four independent, high gain, internally frequency compensated operational amplifiers which were designed specifically to operate from a single power supply over a wide range of voltages. Operation from split power supplies is also possible and the low power supply current drain is independent of the magnitude of the power supply voltage.

Application areas include transducer amplifiers, DC gain blocks and all the conventional op amp circuits which now can be more easily implemented in single power supply systems. For example, the LM124 series can be directly operated off of the standard +5V power supply voltage which is used in digital systems and will easily provide the required interface electronics without requiring the additional ±15V power supplies.

Unique Characteristics

- In the linear mode the input common-mode voltage range includes ground and the output voltage can also swing to ground, even though operated from only a single power supply voltage
- The unity gain cross frequency is temperature compensated
- The input bias current is also temperature compensated

Advantages

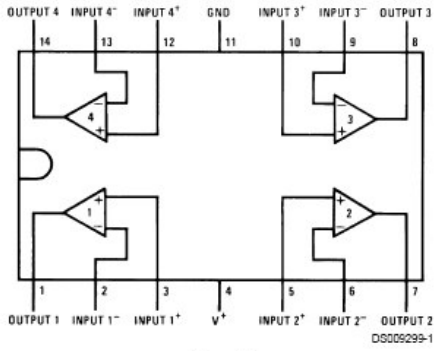
- Eliminates need for dual supplies
- Four internally compensated op amps in a single package
- Allows directly sensing near GND and V_{OUT} also goes to GND
- Compatible with all forms of logic
- Power drain suitable for battery operation

Features

- Internally frequency compensated for unity gain
- Large DC voltage gain 100 dB
- Wide bandwidth (unity gain) 1 MHz (temperature compensated)
- Wide power supply range:
  Single supply 3V to 32V
  or dual supplies ±1.5V to ±16V
- Very low supply current drain (700 µA)—essentially independent of supply voltage
- Low input biasing current 45 nA (temperature compensated)
- Low input offset voltage 2 mV and offset current: 5 nA
- Input common-mode voltage range includes ground
- Differential input voltage range equal to the power supply voltage
- Large output voltage swing 0V to V+ − 1.5V

Connection Diagram

Dual-In-Line Package



Top View

Order Number LM124J, LM124AJ, LM124J/883 (Note 2), **LM124AJ/883** (Note 1), **LM224J, LM224AJ, LM324J, LM324M, LM324MX, LM324AM, LM324AMX, LM2902M, LM2902MX, LM324N, LM324AN, LM324MT, LM324MTX or LM2902N LM124AJRQML and LM124AJRQMLV**(Note 3)
See NS Package Number J14A, M14A or N14A

Note 1: LM124A available per JM38510/11006

Note 2: LM124 available per JM38510/11005

National Semiconductor

August 2000

LM741
Operational Amplifier

General Description

The LM741 series are general purpose operational amplifiers which feature improved performance over industry standards like the LM709. They are direct, plug-in replacements for the 709C, LM201, MC1439 and 748 in most applications.

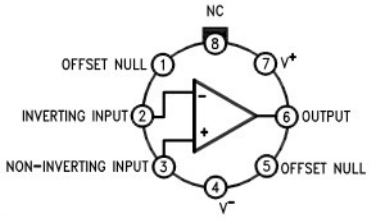
The amplifiers offer many features which make their application nearly foolproof: overload protection on the input and output, no latch-up when the common mode range is exceeded, as well as freedom from oscillations.

The LM741C is identical to the LM741/LM741A except that the LM741C has their performance guaranteed over a 0˚C to +70˚C temperature range, instead of −55˚C to +125˚C.

Features

Connection Diagrams

Metal Can Package

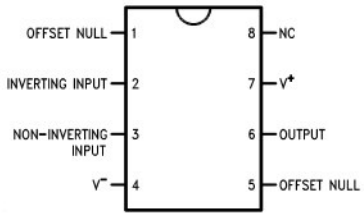


00934102

Note 1: LM741H is available per JM38510/10101

Order Number LM741H, LM741H/883 (Note 1), **LM741AH/883 or LM741CH**
See NS Package Number H08C

Dual-In-Line or S.O. Package



00934103

Order Number LM741J, LM741J/883, LM741CN
See NS Package Number J08A, M08A or N08E

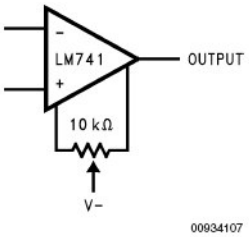
Ceramic Flatpak

00934106

Order Number LM741W/883
See NS Package Number W10A

Typical Application

Offset Nulling Circuit



00934107

8-bit multiplying D/A converter MC1408-8

DESCRIPTION

The MC1408-8 is an 8-bit monolithic digital-to-analog converter which provides high-speed performance with low cost. It is designed for use where the output current is a linear product of an 8-bit digital word and an analog reference voltage.

FEATURES

- Fast settling time: 70 ns (typ)
- Relative accuracy ±0.19% (max error)
- Non-inverting digital inputs are TTL and CMOS compatible
- High-speed multiplying rate 4.0 mA/µs (input slew)
- Output voltage swing +0.5 V to –5.0 V
- Standard supply voltages +5.0 V and –5.0 V to –15 V

APPLICATIONS

- Tracking A-to-D converters
- 2 1/2-digit panel meters and DVMs
- Waveform synthesis
- Sample-and-Hold
- Peak detector
- Programmable gain and attenuation
- CRT character generation
- Audio digitizing and decoding
- Programmable power supplies
- Analog-digital multiplication
- Digital-digital multiplication
- Analog-digital division
- Digital addition and subtraction
- Speech compression and expansion
- Stepping motor drive modems
- Servo motor and pen drivers

PIN CONFIGURATIONS

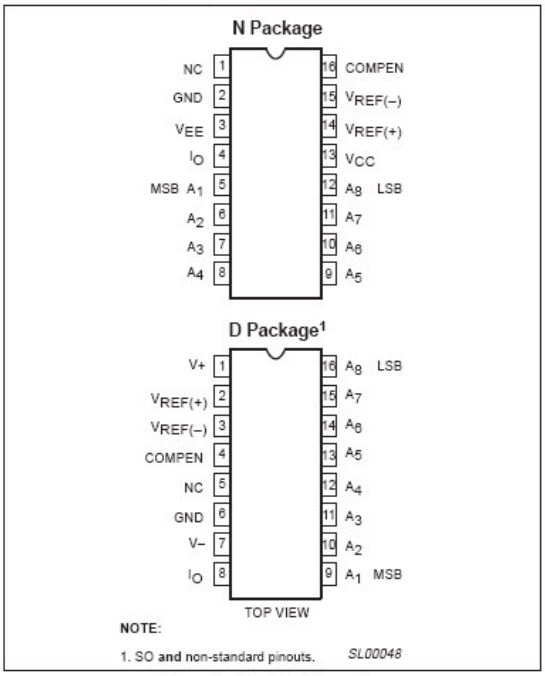


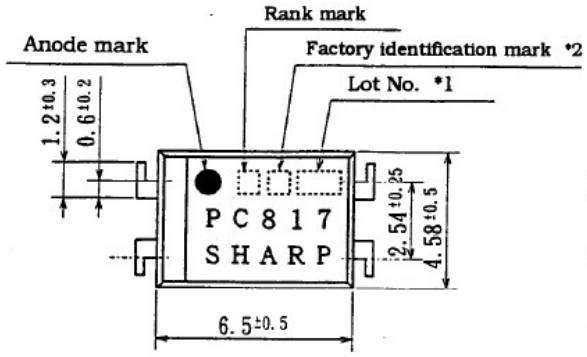
Figure 1. Pin Configurations

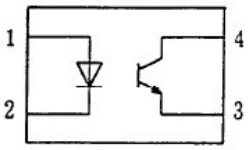
ORDERING INFORMATION

DESCRIPTION	TEMPERATURE RANGE	ORDER CODE	DWG #
16-Pin Plastic Dual In-Line Package (DIP)	0 °C to +70 °C	MC1408-8N	SOT38-4
16-Pin Small Outline (SO) Package	0 °C to +70 °C	MC1408-8D	SOT109-1

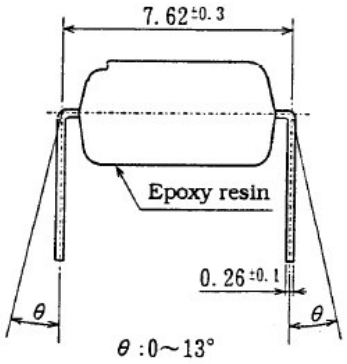
SHARP CORPORATION

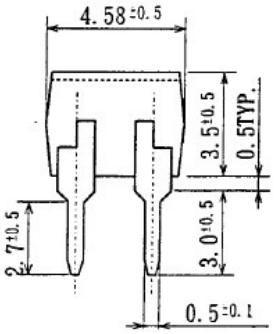
ED-94054D	June 14, 1996
MODEL No. PC817X	PAGE 5/8





Pin Nos. and internal connection diagram





*1) 2-digit number shall be marked according to DIN standard.

*2) Factory identification mark shall be or shall not be marked.

*3) Marking is laser marking

UNIT : 1/1 mm	
Name	PC817 Outline Dimensions (Business dealing name : PC817X)
Drawing No.	CY6961K02

Señales y sistemas
Oppenheim Willsky
Ed. Prentice Hall

Instrumentación virtual
Adquisición, procesado y análisis
Francesc J. Sánchez
Ed. Grupo Alfaomega.

Comunicaciones I.
Herrera
ED. Limusa.

Circuitos en Ingeniería Eléctrica.
Hugh Hildreth Skilling
Ed. C.E.C.S.A.

Aplicaciones de Procesamiento Digital de Señales.
Alan V. Oppenheim.
Ed. Prentice Hall

Introducción a las señales y los sistemas
Douglas K. Linder.
Mc Graw Hill.

Lenguaje C. algoritmos para procesamiento digital de señales.
Paul M Embree
Ed. Prentice Hall

Microelectrónica circuitos y dispositivos.
Mark N Horenstein
Ed. Prentice Hall.

La Biblia del C

Teoría de circuitos con Orcad Pspice
Blas Ogayar
Ed. Alfaomega.

www.departamentodeinformatica.com

http://mailweb.udlap.mx/~ingrid/ingrid/RV/Proc_de_Voz.html

www.agelectronica.com.mx

www.analogdevices.com

www.national.com

www.alek.pucp.edu.pe/~dflores/INDEX.html

www.geocities.com/CapeCanaveral/Runway/3015/inicial.html