DISEÑO ELECTRÓNICO

Capítulo Octavo

INTRODUCCIÓN

En este capítulo veremos lo que es todo el aspecto electrónico de la linea transportadora inteligente. Así como se especificó anteriormente, nos vamos a enfocar en la electrónica de un solo módulo y de su integración con todos los demás. Y dejando al lector, que le dé la aplicación que más crea conveniente para esta línea transportadora.

La parte electrónica será la encargada de realizar las siguientes funciones:

- Conmutar las bobinas
- Dirigir el tráfico dentro del módulo
- Comunicar el módulo con su entorno

Como vimos en el marco teórico, existen muchos sistemas de procesamiento que son capaces de realizar estas funciones sin ningún problema. Unos tienen mayor capacidad que otros, pero una gran verdad es que la aplicación es lo bastante lenta como para que cualquier procesador pueda realizar todas las operaciones necesarias. Hay que recordar que un pallet puede viajar a 20, 30 o tal vez hasta 40Km/hr. Si el paso del motor es de 5cm. Estaríamos viajando a 222.23 pasos por segundo, lo que nos llevaría a una conmutación cada 4.5ms, y eso a la velocidad más alta. Y recordemos que un procesador típico de 8 bits, de los más baratos tiene un ciclo de máquina de 3 a 1µs. Es decir que el procesador puede realizar hasta 4,500 operaciones antes de requerir mover un pallet.

Ahora bien, el punto que sí hay que cuidar es el de la comunicación, puesto que es el proceso más lento, ya que la inductancia y capacitancia de las lineas de transmisión no permite que existan velocidades más altas. Y tal vez esa sea nuestra limitante. Aunque siempre existen posibilidades que mejoran la transmisión, ya sea incrementando la velocidad, cambiando de tipo comunicación o protocolo o añadiendo canales extra.

CONMUTACIÓN

Como ya hemos visto, los módulos están llenos de bobinas, y estas bobinas tienen inductancias mutuas. Pero para evitar eso vamos a utilizar corriente directa para polarizar los inducidos, y que lo único que se induzca sea el transitorio.

Además de que se nos simplifique un poco la electrónica. Y ya que estamos hablando de electrónica de potencia, de conmutaciones, de bobinas y microcontroladores. Hay un circuito que es el ideal para este propósito, que maneja niveles TTL y saca niveles aptos para la alimentación de la bobina. Este circuito es el Puente H.

Este puente funciona de una manera muy sencilla. Pero antes de explicar el funcionamiento de dicho circuito, hay que describir las condiciones que debemos de satisfacer para que el motor realice sus funciones, así como vemos en la tabla siguiente:

Movimiento Requerido	**Diferencia de Potencial en sus terminales**	**Voltaje en la Terminal A**	**Voltaje en la Terminal B**
Gire a la derecha	Positiva (+10v)	10 volts	0 volts
Gire a la izquierda	Negativa (-10v)	0 Volts	10 Volts
Que esté parado	Cero	0 Volts	0 Volts
		10 Volts	10 Volts

Como se ve en la tabla, hay solo tres movimientos requeridos y cuatro posibles condiciones para reproducir esos movimientos. Hay que recordar que todos los voltajes son medidos con respecto a tierra y que para que exista una diferencia de potencial entre las terminales del motor, debe de haber dos potenciales diferentes. Por eso es que para parar el motor no debe de existir diferencia de potencial en sus terminales, pero sí pueden tener las dos terminales el mismo potencial CON RESPECTO A TIERRA.

Para explicar el principio de funcionamiento del puente H vamos a suponer que tenemos un motor que se alimenta con una fuente de 10 volts. Y para hacer que gire a la derecha necesitamos conectar el el positivo de la fuente a la terminal A del motor y el negativo a la terminal B del motor. Si necesitamos hacer que gire el motor a la izquierda, habria que conectar el positivo de la fuente al terminal B y el negativo al A. Pero si quicieramos hacer que se pare el motor simplemente debemos de conectar los dos terminales a un solo borne de la fuente. El puente H hace exactamente lo mismo. Y podemos ejemplificar su funcionamiento en la figura:

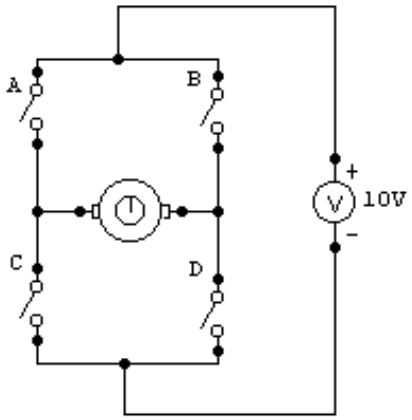

Figura 8.1 - Principio de operación del puente H

En ella se ve que para que este motor gire en un sentido se tiene que cerrar los interruptores A y D y para que gire en otro sentido se tienen que cerrar los interruptores B y C. Y para pararlo se abren los interruptores C y D o A y B. PERO JAMÁS DEBE DE CERRARSE LOS INTERRUPTORES A Y C ni BY C por que causarían un corto circuito en la fuente de alimentación.

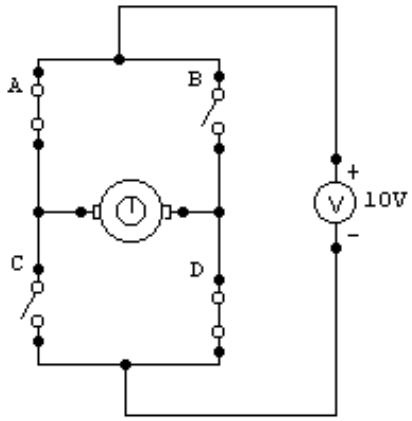

Figura 8.2 - Configuración de los interruptores para que gire en un sentido

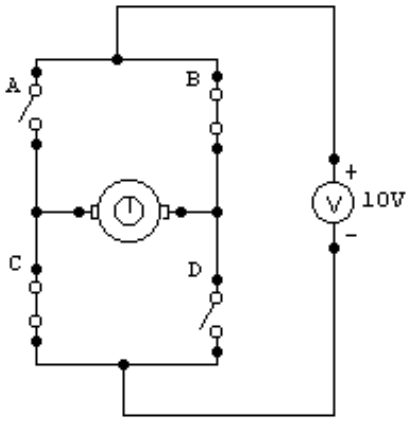

Figura 8.3 - Configuración de los interruptores para que gire en sentido contrario

Ahora bien, el puente H es un circuito electrónico de potencia y en vez de usar interruptores o relevadores utiliza transistores especiales que son llamados de potencia. Ya sean BJTs o FETs los transistores deben de trabajar en modo de corte y saturación. Como todos recordaremos un transistor en corte es igual que un interruptor abierto, por que no deja pasar corriente por sus terminales. Y un transistor en saturación es como un interruptor cerrado que deja pasar toda la corriente que circula por la maya y además no consume voltaje. Claro que esto no es estrictamente cierto refiriendonos a los transistores, por que siempre van a consumir un pequeño voltaje y que es al rededor de los 0.3 volts.

La ventaja de utilizar transistores en vez de interruptores es que los podemos controlar con señales electricas de baja potencia y por la que circulará una corriente minúscula. Así podemos hacer una interfase entre dispositivos lógicos y dispositivos de potencia. Para que un puente H funcione y pueda realizar todos los movimientos requeridos de la tabla anterior, necesita dos señales lógicas que le van a indicar que movimiento debe hacer el motor. Y en la siguiente tabla de verdad tenemos descrito el funcionamiento de un Puente H.

SÑ1	SÑ2	A	B	C	D	Comentario	Estado del Motor
0	0	Cerrado	Cerrado	Abierto	Abierto	No hay diferencia de potencial en el motor	El motor está parado
0	1	Cerrado	Abierto	Abierto	Cerrado	Existe diferenica de potencial (+) en A y (-) en D	El motor gira para la izquierda
1	0	Abierto	Cerrado	Cerrado	Abierto	Existe diferencia de potencial (+) en B y (-) en C	El motor gira para la derecha
1	1	Abierto	Abierto	Cerrado	Cerrado	No hay diferencia de potencial en el motor	El Motor está parado

Como podemos ver en la tabla de verdad anterior el interruptor A se activa cuando la señal SÑ1 esta activa y el interruptor C se activa cuando la señal SÑ1 está en cero. En otras palabras el interruptor A es activado por la señal SÑ1 y el interruptor A es activado por la negación de la señal SÑ1.

A=SÑ1 B=SÑ2 C=Neg(SÑ1) D=Neg(SÑ2)

Teniendo en cuenta las ecuaciones anteriores podemos observar que el puente H consta de dos partes. La parte que está controlada por la señal SÑ1 y la parte controlada por la señal SÑ2. Y que las dos partes son exactamente iguales. Por tanto si analizamos una mitad del puente H habremos entendido bien su funcionamiento.

Entonces vamos a analizar el puente H en la mitad que es controlada por la señal SÑ1 y que está formada por los interruptores A y C. Vamos a substituir los interruptores por transistores que funcionen en corte y saturación. Así hemos remplazado el interruptor A por el transistor QA y el interruptor C por el transistor QC. Hay que recordar que donde tenemos el riesgo de hacer un corto circuito, es si abrimos los interruptores A y B al mismo tiempo. Y para que esto nunca pase se le añadió un transistor QI1 que tiene la función de invertir la señal de entrada (SÑ1). Así jamás entraran en saturación los dos transistores al mismo tiempo y aseguramos el buen funcionamiento del circuito. Vease la figura a continuación:

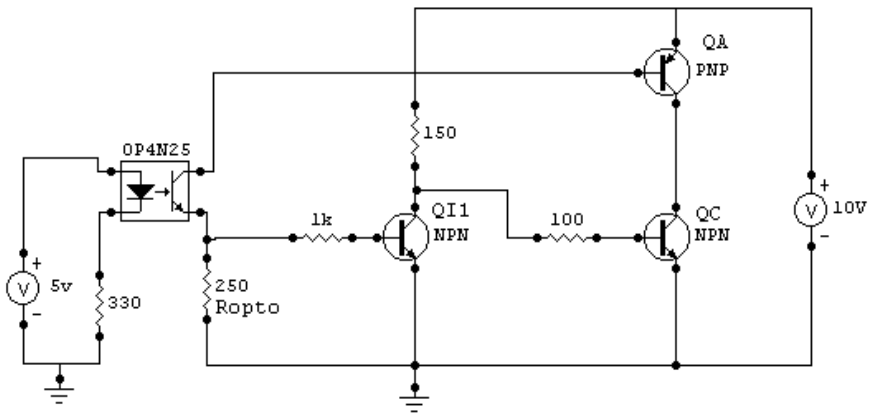

Figura 8.4 - Medio puente H

Empecemos nuestro analisis por definir nuestras entradas y nuestras salidas. La entrada está dado por la fuente de poder de 5v SÑ1, que simula la salida digital de nuestro microprocesador o dispositivo de control TTL. Y la salida es el punto donde están conectados los dos colectores de los transistores QA y QB (ver figura siguiente).

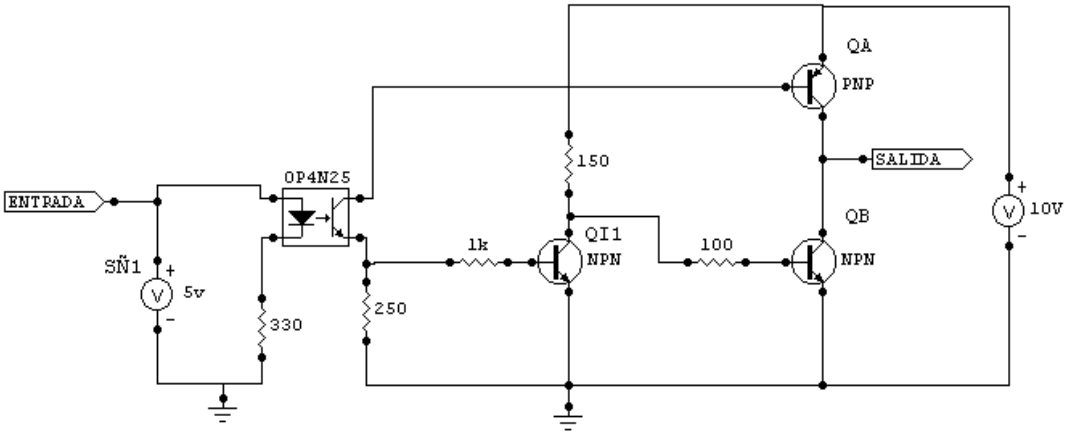

Figura 8.5 - Entradas y salidas del medio puente H

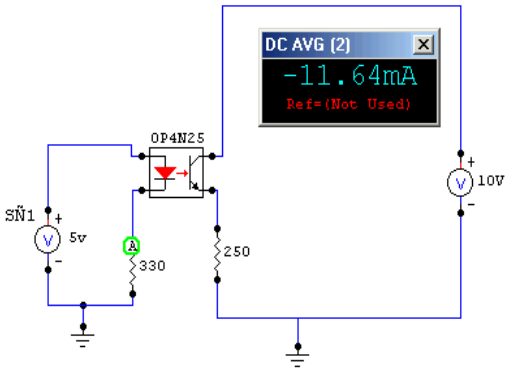

Figura 8.6 - Corriente que circula a la entrada del optoacoplador

La señal de entrada SÑ1, que está representada por la fuente poder de la izquierda, tiene la característica de que va a simular los valores TTL. Que son 5 volts para un uno lógico y 0 volts para un cero lógico. Esta señal se va a conectar a un optoacoplador 4N25 y la cual hará circular una corriente a través del diodo emisor de luz que tiene integrado el optoacoplador y activará el fototransistor del mismo integrado. Hay que recalcar que lo que se está consiguiendo con este optoacoplador es aislar electricamente a estos dos circuitos. El digital que solamente activara al optoacoplador y el puente H que maneja la potencia y los ruidos generados por el motor.

Para no dañar el optoacoplador 4N25, el fabricante recomienda que por el LED no se haga circular una corrient mayor a 60mA y también nos dice que el LED consume un voltaje de 1.15 Volts. Para limitar la corriente que circula sobre el LED se debe de poner una resistencia en su catodo y aterrizarla a tierra como se muestra en la figura. Si le ponemos una resistencia de 330 ohms verificamos que se cumplen las recomendaciones del fabricante

$$I_{LED} = \frac{V_{in} - V_{LED}}{R} = \frac{5 - 1.15}{330} = 0.011667A$$

Y si se fijan el resultado de la simulación (pagina anterior) son los mismos 11.6 mA que nos dieron en el cálculo que hicimos. Ahora bien, cuando el LED es polarizado emite fotones en la base del transistor y hace que el transistor entre en saturación y que el voltaje de emisor a colector sea practicamente cero. Así como lo demuestra la simulación (ver imágen de la izquierda), en donde el transistor está consumiendo 10-9.674=0.327volts. Cuando el transistor entra en saturación permite que casi la totalidad del voltaje de la fuente de 10v sea aplicado directamente a la resistencia del emisor con lo cual podemos calcular que existe una corriente de 10/250=40mA (V/R). Muy parecido a lo que sale en la simulación (imagen de la derecha).

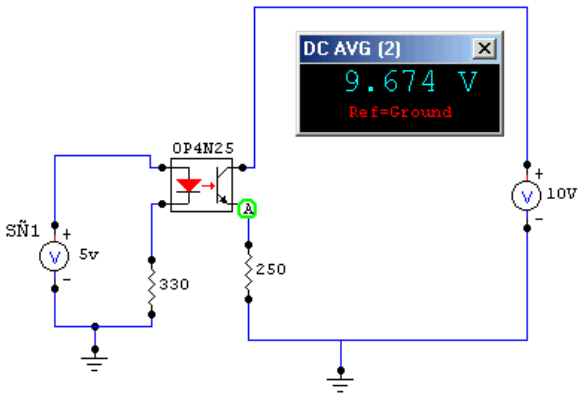

Figura 8.7 - Voltaje que se queda en la resistencia, prueba de saturación del transistor

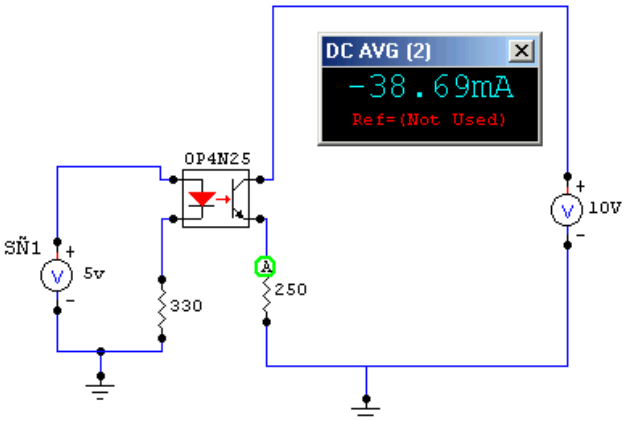

Figura 8.8 - Corriente que circula en la salida del optoacoplador

Ahora bien, usted se estará preguntando ¿y de donde salio esa resistencia de 250 Ohms? ¿Por que se escogió ese valor? Ah pues la respuesta es por que esa corriente de 40mA la necesitaremos para polarizar el transistor QA el cual necesitamos que pase una corriente máxima de 4 Amps. y estimamos que su beta es de 100 lo que nos da el requerimiento de tener una corriente de base de (Ic/Beta = 4/100 = 40mA). En resumen, el valor de la resistencia de emisor del optoacoplador está dado por la siguiente fórmula:

$$R_{Opto} = \frac{V}{I_{cQA} / \beta_{QA}} \qquad R_{Opto} = \frac{10v}{4\,Amps / 100} = 250\Omega$$

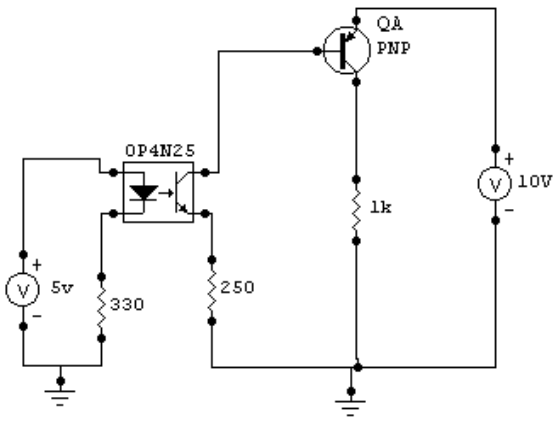

Figura 8.9 - Forma de conectar el primer transistor.

Cuando el valor de la señal de entrada (SÑ1) es cero (cero volts) el LED no se polariza y no induce una corriente en el transistor del optoacoplador. Por tanto entrará en corte, no circulará ninguna corriente a través del mismo y no se polarizará el transistor QA. Cuando el transistor QA no es polarizado entra en corte y no permite el paso de corriente y será el equivalente a un interruptor abierto.

Para que el puente H funcione correctamente hay que hacer que el transistor QC entre en saturación cuando la señal SÑ1 sea cero. Para eso hay que invertir esa señal con el transistor QI1. El transistor Q1 está en configuración emisor común y la característica especial de esa señal es que invierte la señal. La resistencia de base de ese transistor debe tener un valor mayor que la resistencia del optoacoplador para que cuando el transistore del optoacoplador entre en saturación no pase demasiada corriente por él. Solo hay que cuidar que la corriente de base del QI1 no sea tan pequeña como para que el transistor no se sature. Si ponemos una resistencia de 1K el sistema funciona muy bien.

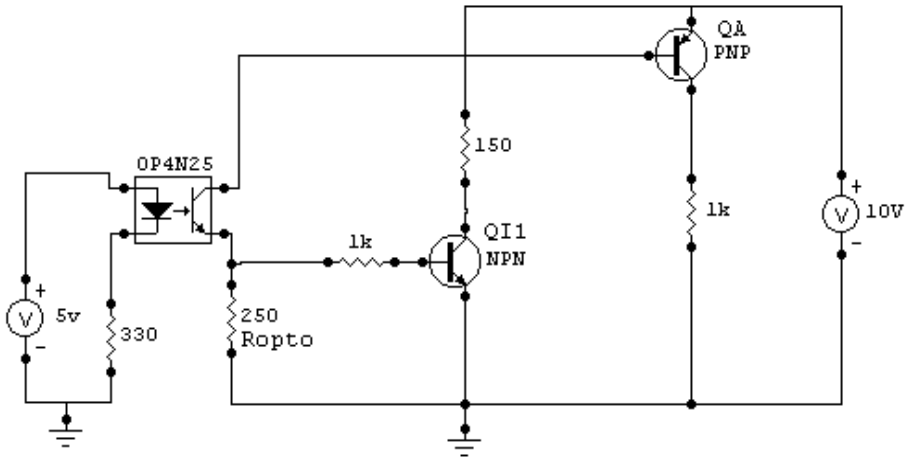

Figura 8.10 - Forma de conectar el segundo transistor del medio puente

Enseguida hay que acoplar el transistor QC a la salida del transistor QI1. Para ello hay que interconectar la base del transistor QC al colector del QI1 por medio de una resistencia. Hay que considerar que el transistor QC se polariza cuando el transistor QI1 está en corte y pasa la corriente a través de la resistencia del colector del transistor

QI1 (150 ohms) y por la resistencia de base del QC (100 ohms) y son los que nos van a dar la corriente de base. Para obtener los valores de las resistencias hay que tomar en cuenta la siguiente equación y después se fijan los valores arbitrariamente. Tenga en cuenta que entre mayor sea la resistencia de colector del transistor QI1 menor será la corriente de base necesaria para saturar el transistor.

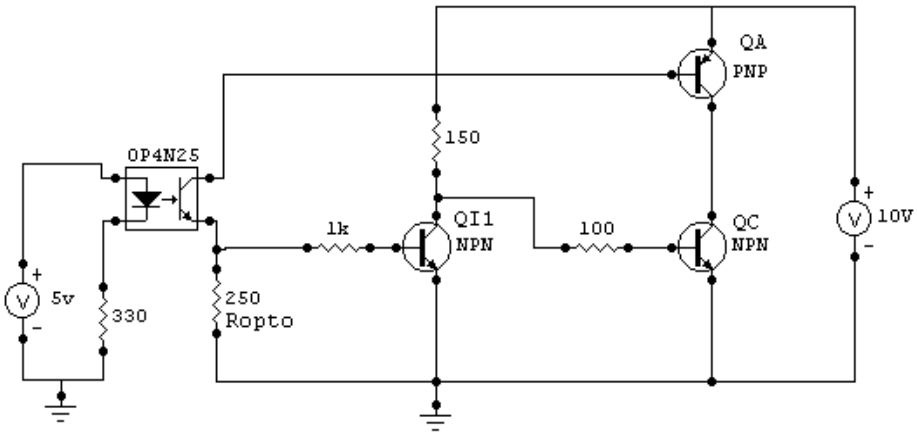

Figura 8.11 - Forma de conectar el tercer transistor del puente H

$$(R_{CQI1} + R_{BQC}) = \frac{V\beta_{QC}}{I_{cQC}}$$

$$(R_{CQI1} + R_{BQC}) = \frac{10v \bullet 100}{4A} = 250\Omega$$

Y ahora vamos a ver el resultado en la simulación. Cuando la señal de entrada SÑ1 vale un uno (5v) en la salida tenemos aproximadamente 10v:

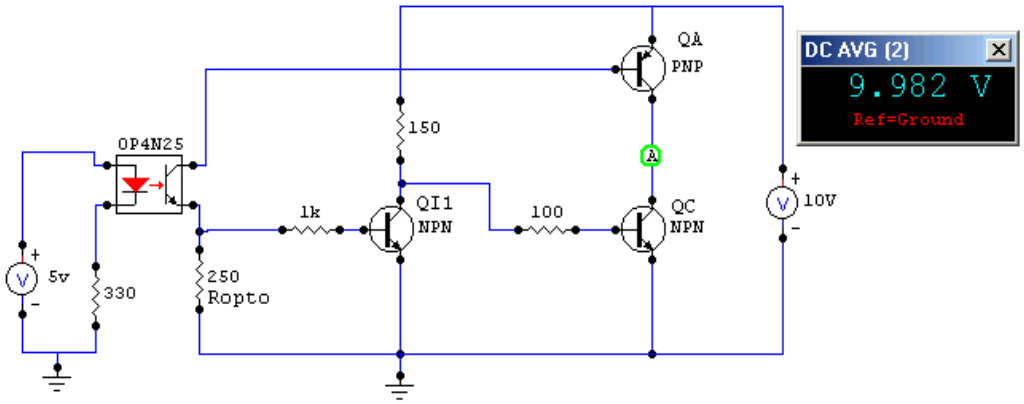

Figura 8.12 - Prueba simulada de que el circuito está operando correctamente

Cuando la señal de entrada SÑ1 vale cero (0v) en la salida tenemos cero volts:

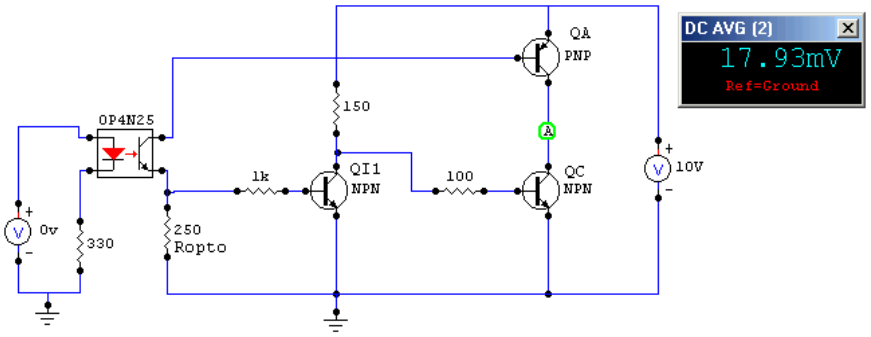

Figura 8.13 - Prueba simulada de que el circuito conmuta correctamente

Una vez que ya tenemos la mitad del puente H funcionando. Lo único que queda hacer es la otra mitad. Exactamente igual pero en espejo. Así como se muestra a continuación:

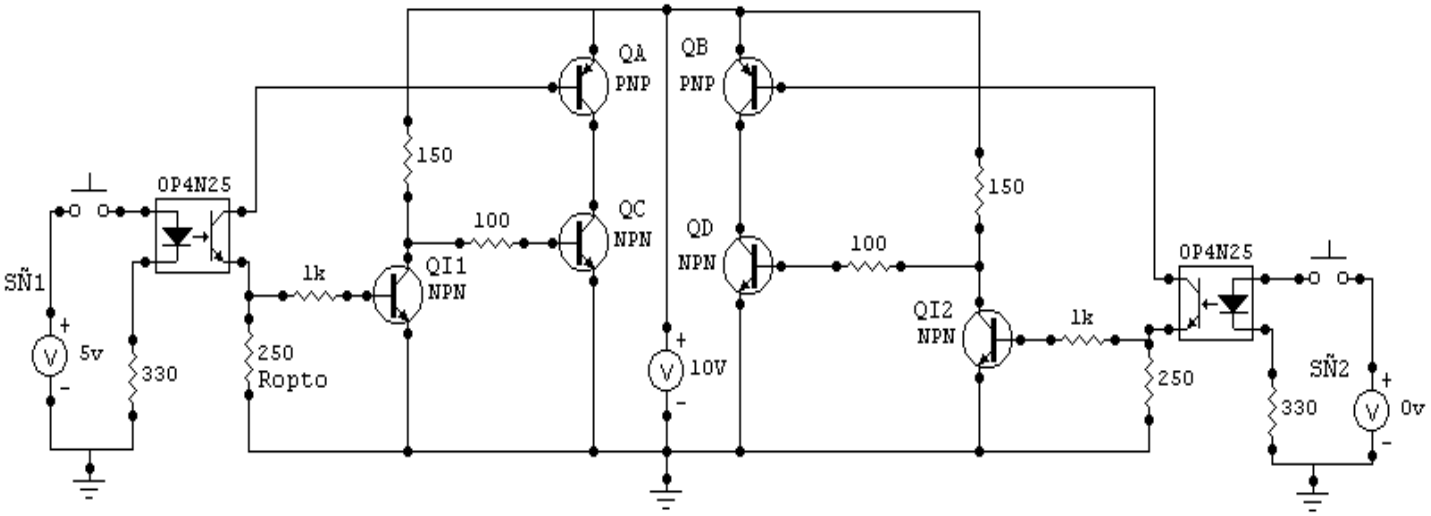

Figura 8.14 - Diagrama final del Puente H completo

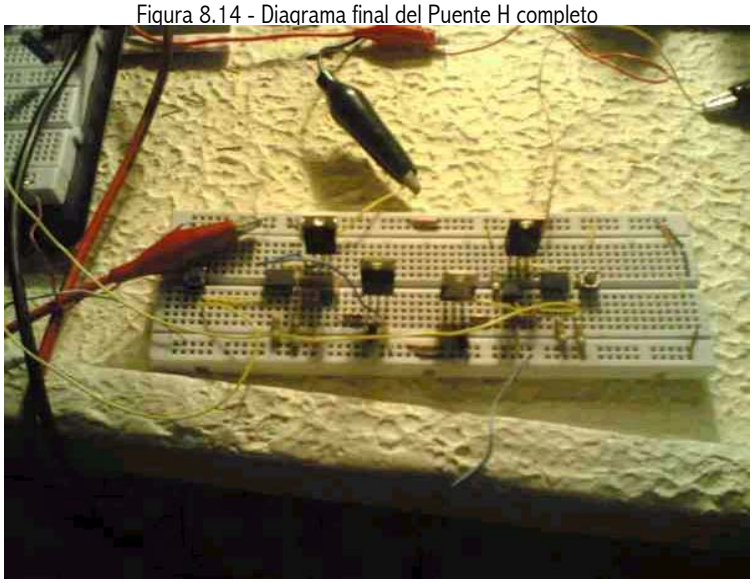

Figura 8.15 - Fotografía del puente H implementado en el protoboard

Despúes de haber probado el circuito y de estar seguro que funcina correctamente hay que agregarle cuatro diodos de protección. Puesto que un motor está formado por bobinados ó inductores que cuando se les corta la corriente generan grandes picos de voltaje que pueden destruir los transistores del circuito. Los diodos deben de quedar de la siguiente manera. Tenga mucho cuidado de ponerlos tal y como se indica en la siguiente figura y si los ponen al revés harán un gran corto circuito.

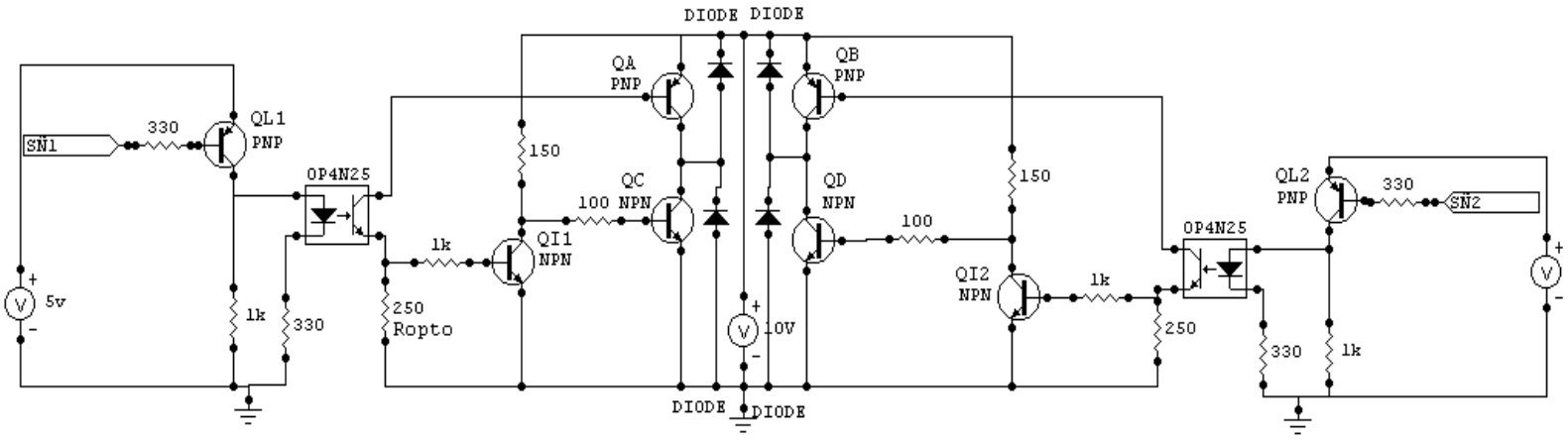

Figura 8.16 - Diagrama final del puente H, con los diodos incluidos y con las entradas digitales

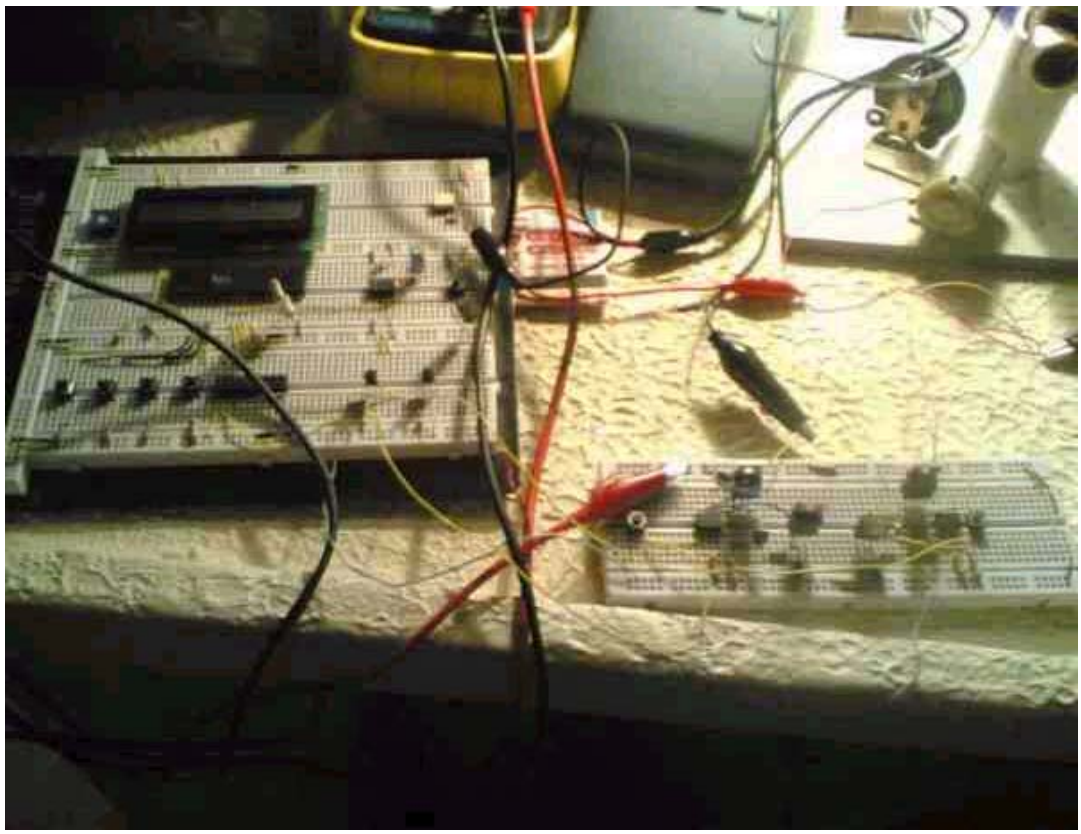

Figura 8.17 - Fotografía del puente H completo, con sus diodos, la interconexión con el microcontrolador At89C51 (protoboard doble de la izquierda).

Ahora que ya sabemos como funciona y para que sirve un Puente H y la mitad de un puente H. Podemos conectar las bobinas. Sustituyendo el motor por el embobinado. Hay que recordar que los diodos sirven para suprimir el transitorio. Y por tanto no habrá que cuidar mucho de él. Hay dos maneras para conectar las bobinas, y todo depende de el número de bobinas a controlar por microprocesador:

Un puente H por cada bobina
Una Matriz de Bobinas

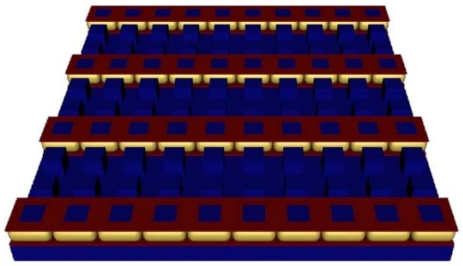

Figura 8.18 - Modulo tipo Bus

En el primero, lo único que vamos a hacer es conectar una bobina a cada puente H. Y por tanto vamos a requerir dos pines del micro controlador, o bien dos salidas digitales, por cada bobina. Si nuestro módulo fuera del tipo BUS y supongamos que de 40 bobinas, como el que se ejemplifica en la figura de la izquierda. No tendríamos ningún problema, ya que fácilmente podemos implementar 80 salidas digitales, con 10 circuitos integrados 74ls373 o 74ls377. Asignamos un puerto del microcontrolador como bus de datos y la mitad de otro puerto (4 bits) como direcciones, o bien utilizamos diez pines de otros puertos. Y si estamos utilizando un At89C51 todavía nos sobrarían pines para seguir conectando dispositivos. A continuación se muestra un diagrama ejemplificando esto mismo:

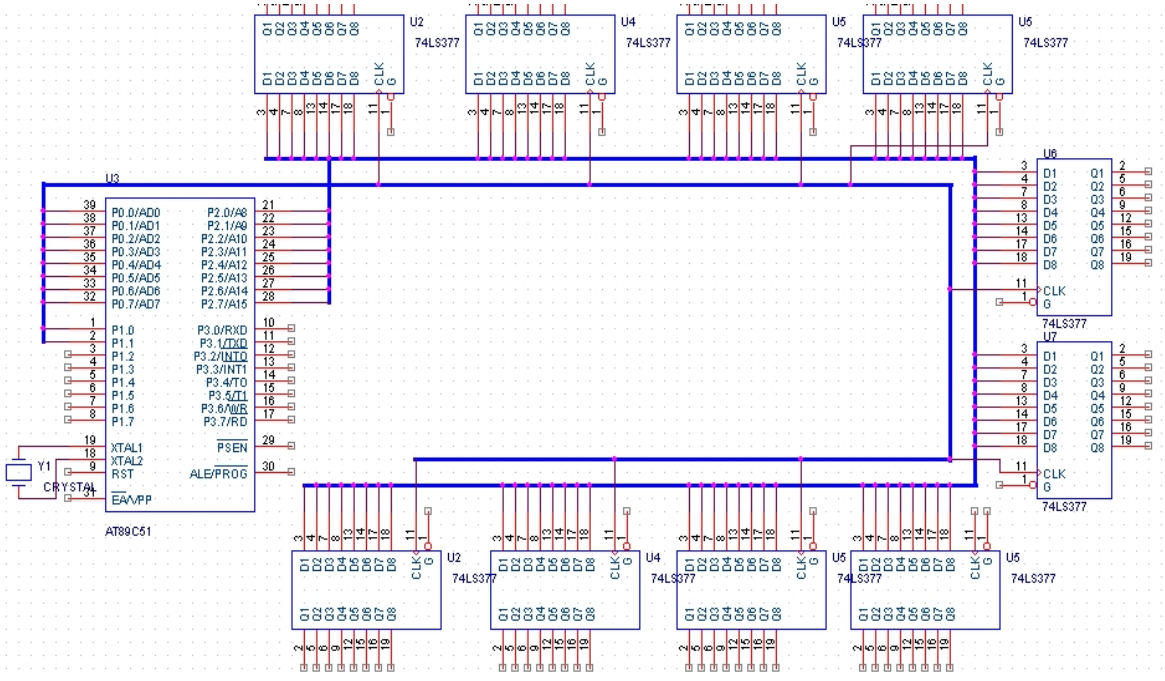

Figura 8.19 - Esquema general de como tener 80 salidas digitales, para ser conectadas a 40 bobinas. Utilizando el At89c51

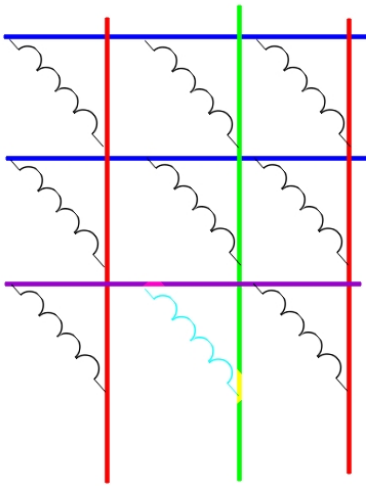

Figura 8.20 - Esquema de conexiones propuesto para una matriz de bobinas

La otra manera de conectar conmutar las bobinas, es haciendo una matriz de bobinas, donde habrá renglones y columnas. Cada bobina estará será la intersección de un renglón con una columna. Una terminal del electroimán se conectará a un renglón y la otra terminal se conectará a una columna. Así como se muestra en el diagrama de la izquierda. Allí vemos que los renglones están en azul, las columnas en rojo, pero el renglón violeta y la columna verde son las únicas que están activadas y juntas activarán solo y únicamente a la bobina azul claro. Aquí cabe aclarar que reducimos la electrónica y ya no necesitaremos el puente H, sino que solamente necesitaremos un transistor funcionando en corte y saturación, para cada línea, ya sea renglón o columna. Hay que mencionar que las líneas renglones deben tener distinta polaridad que las líneas columnas. También conviene señalar, que con este método, sólo se puede activar una bobina a la vez, pero podemos estar activando y desactivando varias bobinas a una alta frecuencia, lo que a la vista humana aparenta que se están activando varias bobinas al mismo tiempo. Otra ventaja de esta conexión es que se requieren menos salidas digitales y menos electrónica de potencia, ya que solo necesitamos controlar las columnas y los renglones, y el número de bobinas controladas, es el producto de ellos dos. Si por ejemplo, quisiéramos controlar 100 bobinas en matriz, solamente tendríamos que controlar 20 salidas digitales (10 renglones, 10 columnas).

FUNCIONES DE CADA MÓDULO

Para que esta línea transportadora sea inteligente, hay que dotar de capacidad de decisión a la misma. Las líneas transportadoras actuales utilizan un PLC o una computadora, para programar el movimiento de cada pallet. Y como generalmente utilizan bandas, pues no pueden avisar de fallos, pallets trabados, ni tienen capacidad de reconfigurar la trayectoria de los pallets. Mi propuesta es que se ponga a trabajar a cada microcontrolador, PLC o computadora constituyente de la línea, en equipo. Aportando cada una de las partes de la línea transportadora con una función específica. Así no recaerá toda la responsabilidad del control en un solo dispositivo de control, también con esto estaremos dando una cierta tolerancia contra fallos. Como vimos en el capítulo anterior. Cada módulo tiene su propia función y de acuerdo a eso clasificamos a los módulos en tres tipos:

Módulo BUS
Módulo Terminal o Acumulador
Módulo Ruteador

MODULO BUS

MODULO BUS

El objetivo principal del Módulo Bus es el de llevar un pallet de un extremo del módulo al otro. El microcontrolador que controla ese módulo debe de llevar la programación necesaria para realizar dicha tarea y demás que son necesarias para el buen funcionamiento de la línea. A continuación hacemos mención de las características inherentes al procesamiento del módulo y cada una de las funciones que debe realizar el procesador de este módulo:

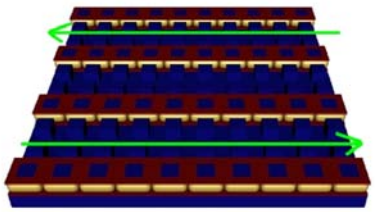

Figura 8.21 - Esquema de un módulo tipo bus. Se observa que los pallets viajan en una sola dirección, pero en dos sentidos diferentes

Características:

Este módulo debe de transportar los pallets de un extemo al otro.

Este módulo sólo puede transportar pallets en una sola dirección, pero puede hacerlo en dos sentidos.

Éste es un módulo de unión, por tanto forzosamente debe de tener un módulo vecino en cada extremo

Funciones del módulo:

Deberá tener la capacidad de conmutar las bobinas adecuadamente para que pueda llevar el pallet hasta el otro extremo

Deberá tener la capacidad de dirigir varios pallets a en el mismo módulo

Deberá tener la capacidad de informar a la computadora central de la localización de los pallets y del estado del módulo.

Deberá comunicar al siguiente módulo de que se tiene un pallet en frontera

Deberá tener una subrutina que acepte o retrase el paso de un pallet del módulo anterior.

Deberá ser capaz de retransmitir a la computadora central, la información del pallet

Deberá ser capaz de detectar fallos en la transportación de un pallet

Deberá ser capaz de comunicar a la computadora central sobre algún fallo u obstáculo en el módulo

MODULO RUTEADOR

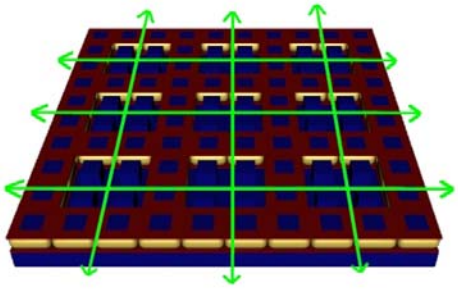

Figura 8.22 - Esquema de un módulo tipo ruteador.

MODULO RUTEADOR

El objetivo principal del Módulo Ruteador es el de dirigir los pallets al extremo del módulo que le corresponda, de acuerdo con su dirección . El microcontrolador que controla ese módulo debe de llevar la programación necesaria para realizar dicha tarea y demás que son necesarias para el buen funcionamiento de la línea. A continuación hacemos mención de las características inherentes al procesamiento del módulo y cada una de las funciones que debe realizar el procesador de este módulo:

Características:

Este módulo debe de transportar los pallets de acuerdo con la dirección del mismo

Este módulo puede transportar pallets multiples direcciones, y en los dos sentidos.

Éste es un módulo puede o no tener conectado un módulo vecino en cada extremo

Este módulo debe de ser capaz de leer la dirección del pallet

Funciones del módulo:

Deberá tener la capacidad de conmutar las bobinas adecuadamente para que pueda llevar el pallet hasta el extremo adecuado

Deberá tener la capacidad de dirigir varios pallets a en el mismo módulo

Deberá tener la capacidad de informar a la computadora central de la localización de los pallets y del estado del módulo.

Deberá comunicar al siguiente módulo de que se tiene un pallet en frontera

Deberá tener una subrutina que acepte o retrase el paso de un pallet del módulo anterior.

Deberá ser capaz de retransmitir a la computadora central, la información del pallet

Deberá ser capaz de detectar fallos en la transportación de un pallet

Deberá ser capaz de comunicar a la computadora central sobre algún fallo u obstáculo en el módulo

MODULO TERMINAL O ACUMULADOR

MODULO TERMINAL O ACUMULADOR

El objetivo principal del Módulo Acumulador es el de dirigir el pallet al extremo del módulo para que sea tomado por la estación de trabajo. Otro objetivo es el de acumular los pallets que están esperando ser atendidos por la estación de trabajo. El microcontrolador que controla ese módulo debe de llevar la programación necesaria para realizar dicha tarea y demás que son necesarias para el buen funcionamiento de la línea. A continuación hacemos mención de las características inherentes al procesamiento del módulo y cada una de las funciones que debe realizar el procesador de este módulo:

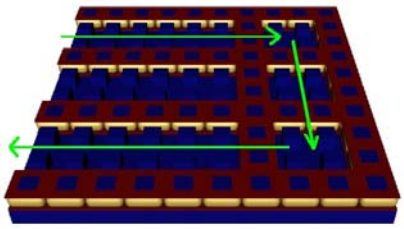

Figura 8.23 - Esquema de un módulo tipo acumulador

Características:

- Este módulo debe de dirigir los pallets al terminal del módulo para que sean tomados por la estación de trabajo
- Este módulo debe de coordinar la acumulación de pallets, para que sean atendidos por la estación de trabajo
- Este módulo deberá de sacar el pallet usado a el router para ser atendido por la siguiente estación de trabajo
- Éste es un módulo forzosamente debe tener conectado módulo ruteador en su extremo
- Este módulo debe de ser capaz de leer la dirección del pallet

Funciones del módulo:

- Deberá tener la capacidad de conmutar las bobinas adecuadamente para que pueda llevar el pallet hasta el extremo adecuado
- Deberá tener la capacidad de dirigir varios pallets a en el mismo módulo
- Deberá tener la capacidad de informar a la computadora central de la localización de los pallets y del estado del módulo.
- Deberá comunicar al siguiente módulo de que se tiene un pallet en frontera
- Deberá tener una subrutina que acepte o retrase el paso de un pallet del módulo anterior.
- Deberá ser capaz de retransmitir a la computadora central, la información del pallet
- Deberá ser capaz de detectar fallos en la transportación de un pallet
- Deberá ser capaz de comunicar a la computadora central sobre algún fallo u obstáculo en el módulo

LAS COMUNICACIONES

Ahora que ya sabemos las funciones que van a realizar los módulos, y la computadora central. Podemos darnos una idea de la información que deberá de fluir en la línea transportadora inteligente. Básicamente será la localización de los pallets, pallets en frontera, y redirección de los pallets. Eventualmente se tendrán comunicaciones de falla, volver a asignar tabla de direcciones para los ruteadores.

Para comunicar todos los módulos con la computadora central se propone utilizar una comunicación serial convencional, como RS485 o RS232. La elección del tipo de comunicación estará dado en función del número de módulos a conectar, ya que RS485 puede ser más veloz y abarcar distancias más grandes. Pero para una aplicación común, con RS232 está bien.

Se propone utilizar un protocolo maestro esclavo, donde la computadora central sea el maestro y los módulos sean los esclavos. La computadora central deberá de coordinar las comunicaciones, ya que será la maestra, también tendrá la facultad de otorgar el token a los módulos para que informen a otros módulos adyacentes sobre algún pallet en frontera. Para ello se puede seguir el protocolo Bitbus, que se describió en el marco teórico.

UNA PEQUEÑA PRUEBA / SIMULACIÓN

Dado a que esta es una tesis y aquí todo debe de estar sustentado, los sinodales nos propusieron que hiciéramos una pequeña prueba / simulación de la comunicación de varios microprocesadores en un mismo canal. Dado este requerimiento se planteó el siguiente experimento:

PLANTEAMIENTO

Vamos a hacer que tres microcontroladores despliegen en tres displays de cristal líquido, lo que se esté tecleando en la computadora en tiempo real. Pero, para hacerlos trabajar en equipo, el mensaje se fragmentará para ser desplegado en los tres displays.

Así por ejemplo, si los displays que serán de 16 caracteres de largo, y la frase es de 45 caracteres, los primeros 16 caracteres se desplegarán en el primer display, los segundos 16 en el segundo y los últimos 13 caracteres se desplegarán en el tercer display.

Figura 8.24 - Ejemplo del experimento propuesto. Primer display

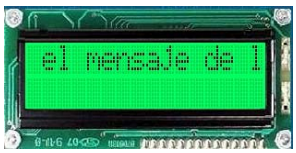

Figura 8.25 - Ejemplo del experimento propuesto. Segundo display

Figura 8.26 - Ejemplo del experimento propuesto. Tercer display

Con esto lograremos demostrar que existe la posibilidad de que un sistema en el que colaboren varios procesadores para un solo fin, opere correctamente, a pesar de que no exista una computadora central que los dirija. Ni que tampoco necesitamos una super computadora que lleve el control de toda la línea inteligente. Ya que si dividimos las responsabilidades, podemos obtener un excelente resultado, sin el gasto de una super computadora.

MANOS A LA OBRA Bueno, para completar esta demostración, lo primero que hicimos fue armarnos con todos los los módulos electrónicos necesarios para dar soporte a nuestro desarrollo. El primer módulo que hicimos, fue el de la interfase entre el puerto serial de la computadora y el UART de los microcontroladores.

Este módulo consta de dos partes, la primera es para engañar a la computadora de que tenemos vamos a conectar un dispositivo DCE (Data Comunication Equipment) que usa la norma RS232 y que va a hacer un Handshake para inicializar la comunicación. Si no se realiza este engaño a la máquina, simplemente no podríamos enviar ni recibir ningún dato, a través del programa Hyperterminal de la máquina.

Este engaño es muy utilizado por los electrónicos, se usa frecuentemente en periféricos que desean ahorrar un poco de hardware y programación. Para realizar dicho engaño hay que implementar una conexión Null Modem con loop back handshaking. Que no es otra cosa, más que conectar los cables adecuadamente para que el puerto de la máquina haga el control de flujo él solito.

Para hacer la conexión Null Modem con loop back handshaking. Hay que interconectar entre si los pines 1,4, y 6, y de forma separada los pines 7 y 8. Hay que recordar que los pines 2,3, y 5 son los que vamos a mandar a los otros dispositivos.

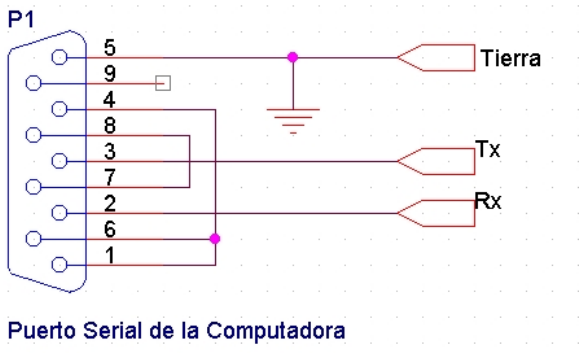

Figura 8.27 - Conexión Null Modem con Loop Back HandShaking

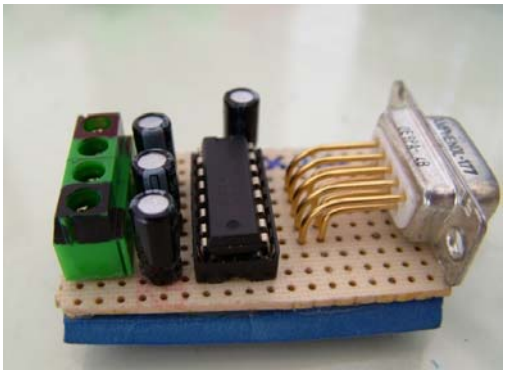

Figura 8.29 - Fotografía del módulo de interfase entre RS232 y TTL

Ahora que ya separamos los tres cables que requerimos de los nueve iniciales, hay que cambiar de los niveles de voltaje RS232 que utiliza la computadora a los niveles TTL que utilizan los microcontroladores. Para ello existen muchos circuitos integrados que realizan esa conversión como el Max232 de Maxim o el DS14C232 de National Semiconductors o el HIN232 de Intersil, etc.

Estos circuitos integrados convierten los niveles de voltaje sin la necesidad de añadir una fuente de 10 o 12 volts al circuito. Esta conversión de voltaje de DC a DC la hacen utilizando la energía de cuatro capacitores de 1µF. El integrado viene en un encapsulado DIP de 16 patitas, y la forma de conectarlo es la siguiente:

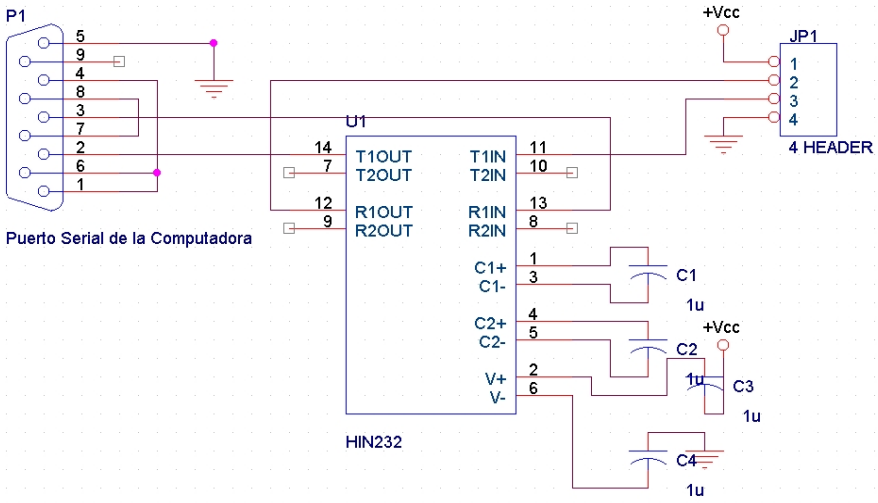

Figura 8.28 - Diagrama del módulo de Interfase entre RS232 y TTL construido para comunicar la computadora con los microprocesadores utilizando el puerto serial de la computadora

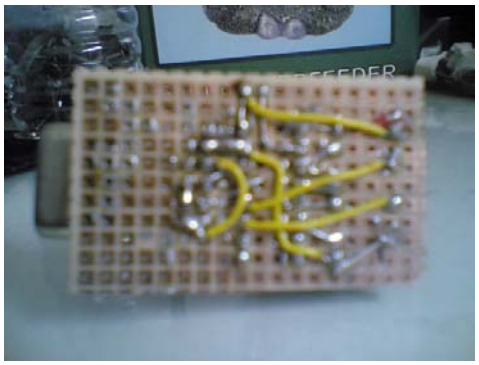

Figura 8.30 - Fotografía de las conexiones hechas para la creación del módulo de interfase entre RS232 y TTL

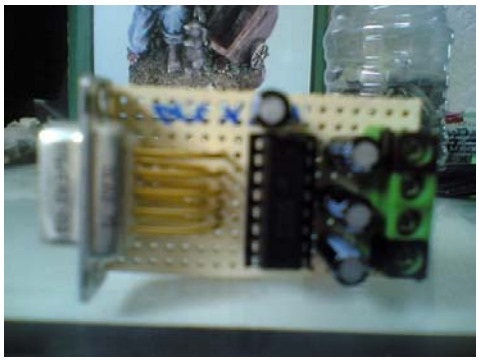

Figura 8.31 - Fotografía de la vista superior del módulo de interfase entre RS232 y TTL. Aqui podemos observar el integrado MAX232, los 4 capacitores y los dos conectores.

Ahora lo que sigue es armar los microprocesadores con sus respectivos displays de LCD. Los LCD que vamos a utilizar son los de texto de 16x8, ya que son los más baratos y simples de utilizar. Solamente permiten visualizar mensajes cortos de texto. El controlador Hitachi HD44780 se ha convertido en un estándar de industria cuyas especificaciones funcionales son imitadas por la mayoría de los fabricantes. Este controlador cuenta con las siguientes terminales eléctricas:

< **D0-D7**: ocho señales eléctricas que componen un bus de datos.
< **R/W**: una señal que indica si se desea leer o escribir en la pantalla (generalmente solamente se escribe).
< **RS**: una señal que indica si los datos presentes en D0-D7 corresponden bien a una instrucción, bien a sus parámetros.
< **E**: una señal para activar o desactivar la pantalla.
< **Vee**: señal electrica para determinar el contraste de la pantalla. Generalmente en el rango de cero a cinco voltios. Cuando el voltaje es de cero voltios se obtienen los puntos más oscuros.
< **Vss** y **Vdd**: señales de alimentación. Generalmente a cinco voltios.
< **LED+** y **LED-**: Señales para encender la luz trasera de la pantalla en algunos modelos.

A continuación veremos una tabla con la asignación estándar de pines para displays menores de 80 caracteres:

Pin	Symbolo	Nivel	I/O	Función
1	Vss	-	-	Tierra (GND)
2	Vcc	-	-	+Vcc (+5V)
3	Vee	-	-	Contraste
4	RS	0/1	I	Selección de Registro 0 = Instrucción 1 = Datos
5	R/W	0/1	I	0 = Escribimos Dato 1 = Leemos Dato
6	E	1, 1->0	I	Enable signal
7	DB0	0/1	I/O	Datos (LSB)
8	DB1	0/1	I/O	Datos
9	DB2	0/1	I/O	Datos
10	DB3	0/1	I/O	Datos
11	DB4	0/1	I/O	Datos
12	DB5	0/1	I/O	Datos
13	DB6	0/1	I/O	Datos
14	DB7	0/1	I/O	Datos (MSB)

Y para controlarlo hay que usar la siguiente tabla de instrucciones:

Command	Code										Description	Execution Time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears the display and returns the cursor to the home position (address 0).	82μs~1.64ms
Return Home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position (address 0). Also returns a shifted display to the home position. DD RAM contents remain unchanged.	40μs~1.64ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and enables/disables the display.	40μs
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Turns the display ON/OFF (D), or the cursor ON/OFF (C), and blink of the character at the cursor position (B).	40μs
Cursor & Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing the DD RAM contents.	40μs
Function Set	0	0	0	0	1	DL	N$	F	*	#	Sets the data width (DL), the number of lines in the display (L), and the character font (F).	40μs
Set CG RAM Address	0	0	0	1	A_{CG}						Sets the CG RAM address. CG RAM data can be read or altered after making this setting.	40μs
Set DD RAM Address	0	0	1	A_{DD}							Sets the DD RAM address. Data may be written or read after making this setting.	40μs
Read Busy Flag & Address	0	1	BF	AC							Reads the BUSY flag (BF) indicating that an internal operation is being performed and reads the address counter contents.	1μs
Write Data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	46μs
Read Data from CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM.	46μs
	I/D = 1: Increment S = 1: Accompanies display shift. S/C= 1: Display shift R/L= 1: Shift to the right. DL = 1: 8 bits N = 1: 2 lines F = 1: 5x10 dots BF = 1: Busy *# Set to 1 on 24x4 modules* *$ With KS0072 is Address Mode.*						I/D = 0: Decrement S/C = 0: cursor move R/L= 0: Shift to the left. DL = 0: 4 bits N = 0: 1 line F = 0: 5 x 7 dots BF = 0: Can accept data				DD RAM: Display data RAM CG RAM: Character generator RAM A_{CG}: CG RAM Address A_{DD}: DD RAM Address Corresponds to cursor address. AC: Address counter Used for both DD and CG RAM address.	Execution times are typical. If transfers are timed by software and the busy flag is not used, add 10% to the above times.

Figura 8.32 - Hoja de instrucciones para programar un display de LCD. Tabla obtenida de la hoja de datos de Hantronix. No 5 de la bibliografía

Estos módulos de Cristal Líquido, aparte de tener la función de display, también son una memoria RAM, donde se guardan los caracteres enviados, y leer dichos valores. Aunque no es común que se haga eso. Ahora bien, cada localidad de memoria, es un lugar para desplegar un caracter. Y cada lugar para poner un caracter tiene su dirección de memoria, así como lo muestra la figura siguiente:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	← DISPLAY POSITION
FIRST LINE	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	DD RAM ADDRESS
SECOND LINE	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	

Figura 8.33 - Mapa de las direcciones de memoria que ocupa cada caracter en el display de LCD

Por ejemplo, para posicionar el cursor en la localidad 40h (principio del segundo renglón), hay que ingresar la instrucción Set DD Ram Address con la dirección 40h (Add= 40 = 100 000). Con lo cual tendremos el siguiente byte:

Instrucción	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Set DD Ram Address	0	0	1	*1*	*0*	*0*	*0*	*0*	*0*	*0*

COMUNICACIÓN DE LA PC A UN DISPLAY DE LCD

Ahora que hemos mostrado, como se utiliza un display de LCD. Hay que hacer que despliege la información transmitida por la computadora. Así como se muestra en el esquema general de conexión. Para ello utilizamos el puerto serial, que utiliza la norma RS-232 y el UART del AT89C51 que es de la familia MCS51.

Figura 8.34 - Esquema general de conexión para desplegar la información transmitida del puerto serial de la computadora a un display de LCD.

La computadora utiliza la norma RS232, en donde los voltajes son de 12 volts, y además están negados. El UART del microcontrolador utiliza niveles de voltaje TTL. Por eso es que tuvimos que utilizar el pequeño circuito MAX232, que nos convierte esos dos niveles de voltajes.

Como el estándar RS232 utiliza muchas señales, y el microcontrolador utiliza solamente tres, hubo que hacer una conexión Null Modem para que la computadora pudiera mandar la información al microcontrolador. Y esto se implementó con el módulo descrito anteriormente que convierte de RS232 en TTL.

El módulo que se construyo específicamente para este experimento, fue el módulo del display. El cual consta de un display de LCD de 16 caracteres por 2 renglones, un microcontrolador AT89C51 que le indica al display lo que tiene que desplegar, además de comunicarse con la PC. En la figura siguiente podemos ver el diagrama eléctrico del módulo:

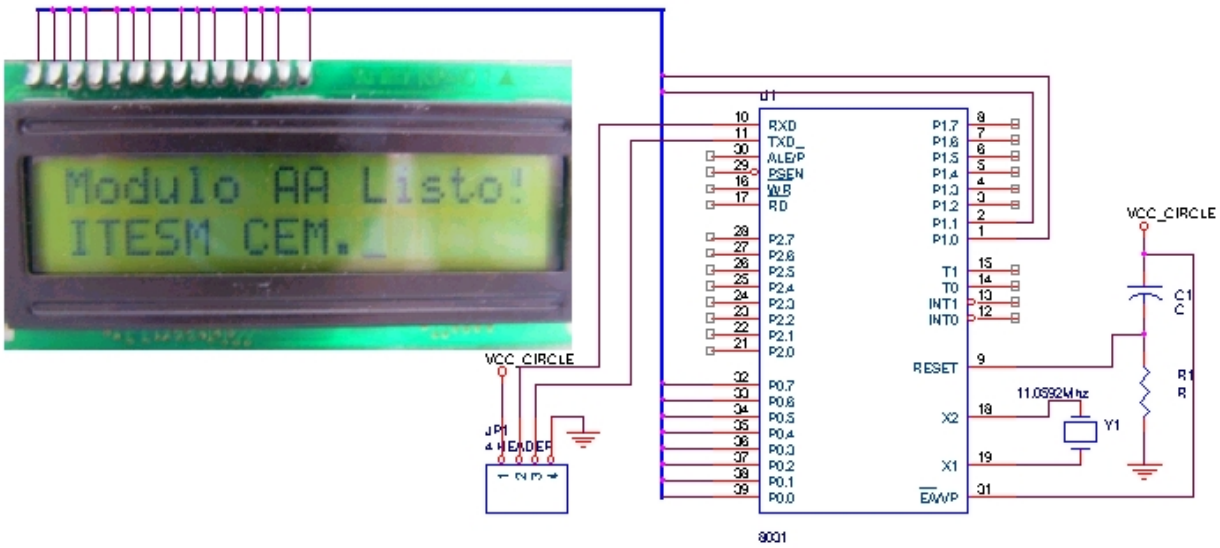

Figura 8.35 - Diagrama eléctrico del módulo del display

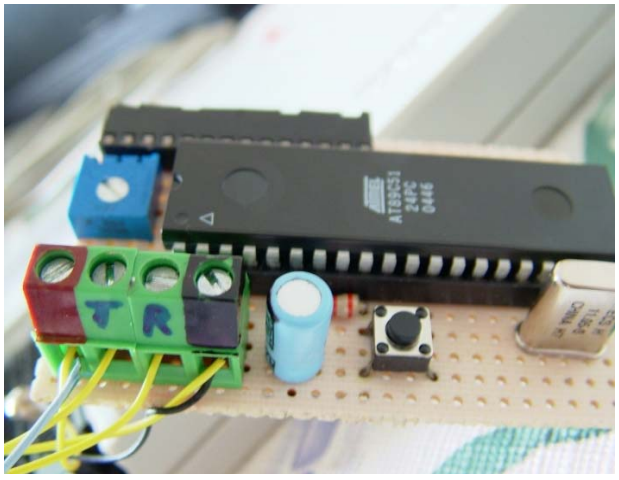

Figura 8.36 - Fotografía de la base del módulo de LCD, se observa el microcontrolador AT89C51, su conector de alimentación y de comunicación serial.

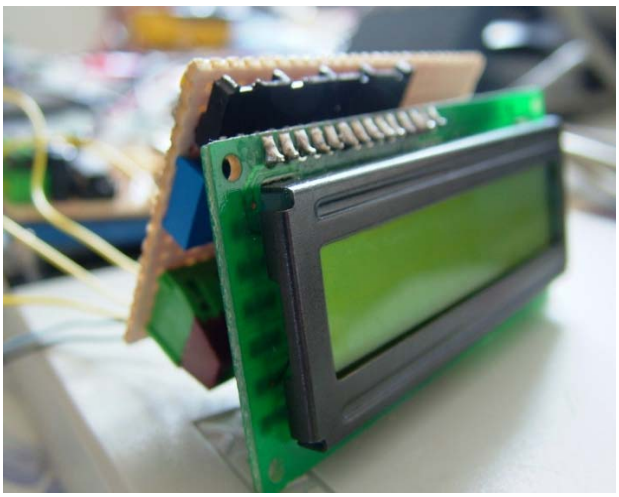

Figura 8.37 - Fotografía del módulo completo de LCD.

El programa que se utilizó fue se muestra a continuación. Se omiten las subrutinas de tiempo, de inicialización del puerto serial y de LCD , si requieren conocer estas subrutinas, pueden accesar al sitio http://smh.famaher.com/ . Se muestra la subrutina de interrupción serial, puesto que es parte medular de este programa.

LTI002.asm

```
LCD_DI equ P1.0              ;Bit del microcontrolador que se
                              ;conecta al pin DI del Display de LCD

LCD_En equ P1.1              ;Bit del microcontrolador que se
                              ;conecta al pin EN del Display de LCD

SERIALBUFFER equ 0FF00h


org 0000h
   sjmp INICIO

org 0023h
   jmp INTERRUPCION_SERIAL


INICIO:
   call INICIALIZA_LCD

   mov DPTR,#CADENA1          ;Muestra mensaje de Bienvenida
   call MUESTRA_MENSAJE_LCD
   call BAJA_RENGLON_LCD

   setb EA
   setb ES
   mov P3,#0FFh
   call INICIALIZA_9600_8_1SP_11Mhz

   mov SBUF,#'S'

FIN: sjmp FIN
```

```
;================================================
;                  CADENAS DE CARACTÉRES
;================================================

CADENA0: db '1234567890123456'
CADENA1: db 'Modulo AA Listo!'

;-------------------------------------------------------------------------------
;INTERRUPCION_SERIAL -- Transmite cadena de caracteres, Recibe
; una cadena de caracteres y la manda a
; un buffer serial para su futura
; operacion
;
;USA: PSW.5 Acc MODIFICA: DPTR
;
;Variables de Entrada:
; DPTR -- Dirección del Mensaje
; SERIALBUFFER -- Dirección de inicio del Buffer Serial
;Variables de Salida:
; NINGUNA
;
;Observaciones: Puerto serial a 9600bps, Hay que definir la
; direccion del buffer serial, el separador de
; comandos es enter 13 decimal = 0Dh
;
;Ejemplo:
; SERIALBUFFER equ 0FF00h
;
;
;MENSAJE1:
; DB 'Salvador Macias Hernandez'
; DB 00h
;-------------------------------------------------------------------------------

INTERRUPCION_SERIAL:

  JB TI, DATOENVIADO        ;Verificando Mótivo de Interrupción
  JB RI, DATORECIBIDO

DATOENVIADO:
  clr TI
reti
```

```
DATORECIBIDO:
   clr RI
   mov A,SBUF
   mov SBUF,A

   clr LCD_En
   setb LCD_DI          ;Decimos que es dato ;)
   mov P0,Acc
   setb LCD_En
   call ESPERA_50uS
   clr LCD_En
reti
end
```

Ahora hay que correr el programa y probar el circuito, al encender el módulo apareció lo que se muestra en la figura 8.38. Ya que lo primero que hace nuestro programa es mandar un mensaje (figura 8.38) diciendo la dirección del módulo (AA) y el estado (Listo). Y en el renglón de abajo se pone el cursor, esperando que la computadora le mande la información a escribir en ese mismo renglón. Para comunicar a los dos aparatos, es necesario tener una velocidad de 9600bps, ser de 8 bits, tener un bit de paro y ningún bit de paridad. A continuación vemos lo que se mandó escribir (figura 8.39):

Figura 8.38 - Mensaje de inicio del módulo de display. Muestra la dirección del módulo y el estado del módulo. También deja el cursor parpadeando en espera de recibir información a desplegar.

Figura 8.39 - El módulo de LCD resultó un exito, ya que se tecleo en la computadora el texto que aparece en el display. El texto fue apareciendo sin retraso alguno al ojo humano.

COMUNICACIÓN DE TRES MICROCONTRADORES

El objetivo de este proyecto es demostrar que es posible comunicar varios procesadores para que colaboren entre sí para llegar a cumplir su misión. El experimento que se propone, es el de comunicar una PC con tres microcontroladores para que lo que se vaya escribiendo en la PC se vaya observando en tiempo real, en los displays de los microcontroladores.

Para realizar este experimento, hay que tener las siguientes dispositivos:

Una computadora con puerto serial
Un módulo de Interfase entre RS-232 y TTL
Tres módulos de Microcontrolador con Display

Una vez que pudimos comunicar la PC con un microcontrolador, y desplegar en tiempo real la información que se manda de la PC a través del Hyperterminal. Ahora es tiempo de conectar dos microcontroladores que despliegen en forma coordinada el texto que se está tecleando en la máquina. Vamos a utilizar el mismo esquema general de conexión que en el experimento anterior:

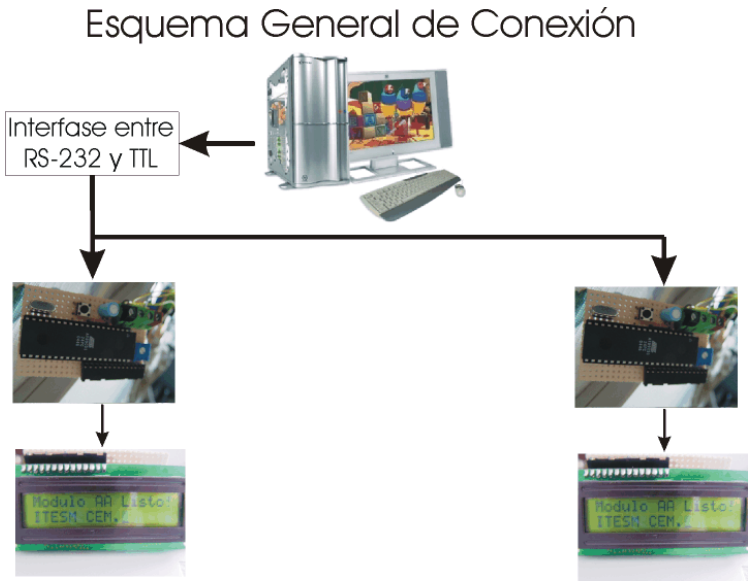

Figura 8.40 - Esquema general de conexión para el segundo experimento.

Lo novedoso aquí, es que, ahora hay que coordinar el trabajo de la PC y de los dos microcontroladores. Puesto que lo único que comparten es el canal de comunicación. Para ello se debieron de imponer las siguientes reglas:

Las terminales de recepción y transmisión, se unirán eléctricamente para hacer un solo canal de comunicación donde todos escuchen lo mismo.
La PC es la única que podrá mandar datos.
Los microcontroladores son los únicos que podrán mandar códigos de control.
El usuario deberá mandar únicamente texto. No se podrán mandar códigos extendidos del sistema ascii.
Cada microcontrolador tendrá una dirección, comprendida desde la 01 hasta la 31 decimal.
La PC no requiere de ninguna dirección

Ahora bien, partiendo de lo que tenemos (experimento anterior), vimos que el display despliega los primeros 16 caracteres de lo que se está tecleando, pero si el usuario sigue tecleando más caracteres, el display seguirá guardando en su memoria los caracteres insertados. Ahora lo que requerimos es que muestre esos primeros 16 caracteres, y que después ceda el turno al siguiente display, para que éste despliegue otros 16 caracteres.

Pora eso propusimos que el microcontrolador con la dirección 01 cuente los caracteres enviados y desplegados. Y cuando ya haya desplegado los 16 caracteres que tiene capacidad de desplegar (primeros 16 caracteres enviados por la PC), mande inmediatamente un código de control, que indique al siguiente microcontrolador que es su turno. Este código de control, va a ser la dirección del microcontrolador. En este caso mandará un 02, que en código ascii es una carita en negro. Y cuando ese microcontrolador acabe de desplegar sus 16 caracteres (segundos 16 caracteres enviados por la PC), inmediatamente enviará un 03, y así suecesivamente. Hasta que se llegue a activar al último módulo conectado.

Bueno, y tal vez se estará preguntando: ¿Como puede funcionar este minúsculo protocolo, si el usuario de la PC esta tecleando constantemente, y la PC está ocupando el canal constantemente? y la respuesta es simple. La PC ocupa el canal constantemente, pero no en todo momento. La PC manda un byte cada que el usuario pulsa una tecla. Y la velocidad de transmisión es muchísimo mayor que la velocidad de mecanografiado de un humano.

Una excelente secretaria, con muchísima experiencia, llega a mecanografiar hasta 350 pulsaciones por minuto, lo que equivale a decir que teclea 5.8333 caracteres por segundo, lo que quiere decir que pulsa una tecla cada 0.17143 segundos. Pero la PC tarda en enviar un byte con su bit de comienzo y su bit de parada (10 bits en total) 0.00104166 segundos a una velocidad de 9600 Bits por segundo. La diferencia de tiempos es enorme casi 100 veces, por tanto, no podemos asegurar que no ocurrirán colisiones, pero sí podemos asegurar que serán muy escasas, o tal vez nulas.

Una vez aclarado esto, y armado el circuito. Vamos comenzar el experimento. Lo primero que hay que hacer es iniciar una conexión serial, utilizando el Hypreterminal. Que lo puedes encontrar en: Inicio \ Programas\ Accesorios\Comunicaciones. Y hay configurarlo para que tengamos una conexión a:

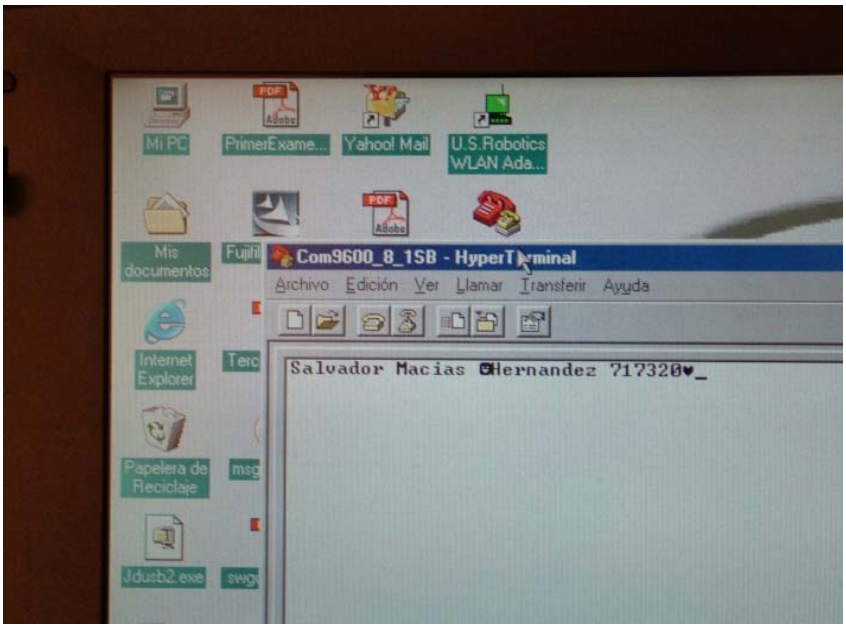

Figura 8.41 - Fotografía de la pantalla de la computadora al realizar el segundo experimento. Se debe notar que se está realizando una comunicación a 9600 Baudios, 8 bits de dato, 1 bit de parada. Y también se puede observar que cada 16 caracteres hay un código de control. Que en este caso es la carita negrita y el corazon.

9600 Bps
8 bits de datos
1 bit de parada
Sin paridad

Luego hay que alimentar los microcontroladores, y esperar a que muestren su dirección y su estado. Van a desplegar algo así: Modulo 02 Listo! Cuando ya hayan desplegado eso. Entonces sí, lo que el usuario tendrá que hacer es teclear una frase, oración, o simplemente su nombrey número de matrícula, como lo fue en este caso. Observaremos que en el Hyperterminal, se observará lo que se está tecleando, y al mismo instante también se observará en los displays de los módulos. Así como lo muestran las fotografías siguientes:

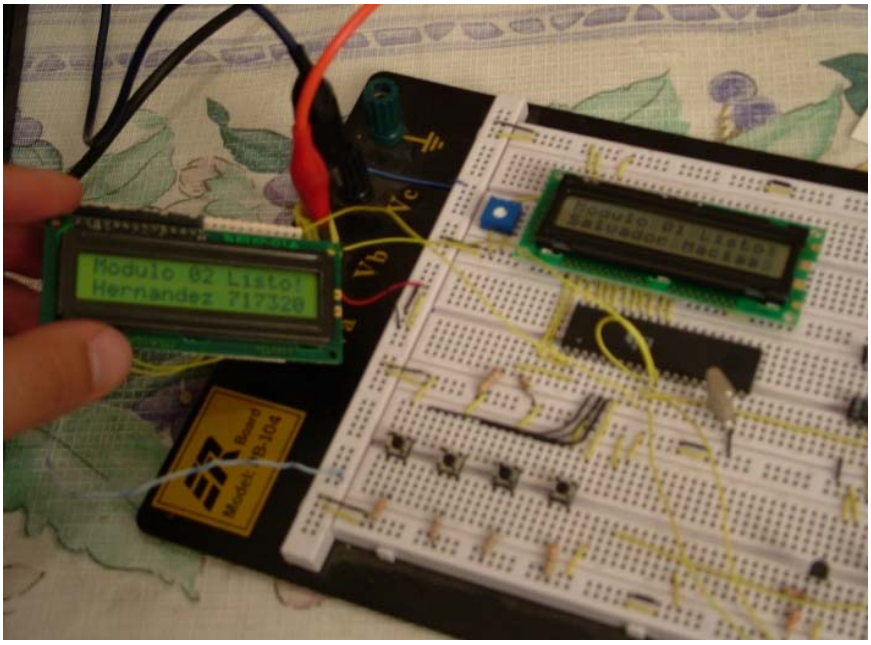

Figura 8.42 - Fotografía de el segundo experimento. Se observa que en el módulo con dirección 01 está mi nombre y mi primer apellido, y en el módulo con dirección 02 está mi apellido materno y mi matrícula.

En la fotografía anterior (Figura 8.41), que es de la ventana del Hyperterminal, se nota mi nombre y en el caracter 17 se ve una carita en negro, esa carita es el código de control (02h) que le indica al siguiente módulo, que es su turno. Por eso es que en el módulo con dirección 01h se observa desplegado "Salvador Macias" y en el siguiente módulo con dirección 02h se observa "Hernandez 717320"

Como se observa en las fotos, se logró una muy buena coordinación entre los procesadores. No existe ningún procesador central que controle absolutamente todo, sino que se hizo una cooperación entre los diferentes módulos. Cada procesador le tocó hacer una tarea en específico, y se despreocupó de la de los demás. Esto nos permitió, tener varios microcontroladores sencillos, en vez de un gran y COSTOSO procesador, que quiera controlar todo.

A continuación se presenta el programa que se integró en el primer microcontrolador (dirección 01h). Solamente se presenta la subrutina de recepción serial, puesto que es parte activo de este programa. Las demás subrutinas las puedes encontrar en la página de internet http://smh.famaher.com/.

Lti003.asm

```
LCD_DI equ P1.0              ;Bit del microcontrolador que se
                             ;conecta al pin DI del Display de LCD
LCD_En equ P1.1              ;Bit del microcontrolador que se
                             ;conecta al pin EN del Display de LCD
SERIALBUFFER equ 0FF00h


org 0000h
   sjmp INICIO

org 0023h
   jmp INTERRUPCION_SERIAL

INICIO:
  call INICIALIZA_LCD

  mov DPTR,#CADENA1 ;Muestra mensaje de Bienvenida
  call MUESTRA_MENSAJE_LCD
  call BAJA_RENGLON_LCD

  setb EA
  setb ES
  mov R2,#00h
  mov R3,#01h
  mov P3,#0FFh
  call INICIALIZA_9600_8_1SP_11Mhz

FIN: sjmp FIN

;=================================================
; CADENAS DE CARACTÉRES
;=================================================
CADENA0: db '1234567890123456'
CADENA1: db 'Modulo 01 Listo!'
```

```
;-------------------------------------------------------------------------------
;INTERRUPCION_SERIAL -- Transmite cadena de caracteres, Recibe
; una cadena de caracteres y la manda a
; un buffer serial para su futura
; operacion
;
;USA: PSW.5 Acc MODIFICA: DPTR
;
;Variables de Entrada:
; DPTR -- Dirección del Mensaje
; SERIALBUFFER -- Dirección de inicio del Buffer Serial
;Variables de Salida:
; NINGUNA
;
;Observaciones: Puerto serial a 9600bps, Hay que definir la
; direccion del buffer serial, el separador de
; comandos es enter 13 decimal = 0Dh
;
;Ejemplo:
; SERIALBUFFER equ 0FF00h
;
;MENSAJE1:
; DB 'Salvador Macias Hernandez'
; DB 00h
;-------------------------------------------------------------------------------
INTERRUPCION_SERIAL:
   JB TI, DATOENVIADO        ;Verificando Mótivo de Interrupción
   JB RI, DATORECIBIDO

DATOENVIADO:
   clr TI
reti

DATORECIBIDO:
   clr RI
   cjne R3, #01h, SALIR_INT_SERIAL        ;Verifica si es el turno
                                          ;de este micro
   DESPLEGAR_INFO:
      mov Acc, SBUF
      clr LCD_En
      setb LCD_DI                         ;Decimos que es dato ;)
      mov P0,Acc
      setb LCD_En
      call ESPERA_50uS
      clr LCD_En
```

```
        inc R2
        cjne R2, #16, SALIR_INT_SERIAL   ;Verifica que no se hayan
        mov SBUF, #02h                   ;escrito los 16 caracteres de
        mov R3,#02h                      ;este display, si sí pasa turno a 02h
        sjmp SALIR_INT_SERIAL

SALIR_INT_SERIAL:
   reti
end
```

El siguiente programa es el que se cargó en el segundo microcontrolador (dirección 02h)

Lti003_02.asm

```
LCD_DI equ P1.0                  ;Bit del microcontrolador que se
                                 ;conecta al pin DI del Display de LCD
LCD_En equ P1.1                  ;Bit del microcontrolador que se
                                 ;conecta al pin EN del Display de LCD
SERIALBUFFER equ 0FF00h

org 0000h
   sjmp INICIO

org 0023h
   jmp INTERRUPCION_SERIAL

INICIO:
   call INICIALIZA_LCD

   mov DPTR,#CADENA1 ;Muestra mensaje de Bienvenida
   call MUESTRA_MENSAJE_LCD
   call BAJA_RENGLON_LCD

   setb EA
   setb ES
   mov R2,#00h
   mov R3,#01h
   mov P3,#0FFh
   call INICIALIZA_9600_8_1SP_11Mhz

FIN: sjmp FIN
;=================================================
; CADENAS DE CARACTÉRES
;=================================================
CADENA0: db '1234567890123456'
CADENA1: db 'Modulo 02 Listo!'
```

```
;-------------------------------------------------------------------------------
;INTERRUPCION_SERIAL -- Transmite cadena de caracteres, Recibe
; una cadena de caracteres y la manda a
; un buffer serial para su futura
; operacion
;
;USA: PSW.5 Acc MODIFICA: DPTR
;
;Variables de Entrada:
; DPTR -- Dirección del Mensaje
; SERIALBUFFER -- Dirección de inicio del Buffer Serial
;Variables de Salida:
; NINGUNA
;
;Observaciones: Puerto serial a 9600bps, Hay que definir la
; direccion del buffer serial, el separador de
; comandos es enter 13 decimal = 0Dh
;
;Ejemplo:
; SERIALBUFFER equ 0FF00h
;
;MENSAJE1:
; DB 'Salvador Macias Hernandez'
; DB 00h
;-------------------------------------------------------------------------------
INTERRUPCION_SERIAL:
   JB TI, DATOENVIADO          ;Verificando Mótivo de Interrupción
   JB RI, DATORECIBIDO

DATOENVIADO:
   clr TI
reti

DATORECIBIDO:
   clr RI
   mov Acc,SBUF
   cjne A,#02h, VERIFICA_TURNO
   mov R3, #02h
   sjmp SALIR_INT_SERIAL

VERIFICA_TURNO:
   cjne R3, #02h, SALIR_INT_SERIAL          ;Verifica si es el turno
                                            ;de este micro
   DESPLEGAR_INFO:
      mov Acc, SBUF
      clr LCD_En
```

```
        setb LCD_DI ;Decimos que es dato ;)
        mov P0,Acc
        setb LCD_En
        call ESPERA_50uS
        clr LCD_En

        inc R2
        cjne R2, #16, SALIR_INT_SERIAL      ;Verifica que no se hayan
        mov SBUF, #03h                      ;escrito los 16 caracteres de
        mov R3,#03h                         ;este display, si sí pasa turno a 03h
        sjmp SALIR_INT_SERIAL

SALIR_INT_SERIAL:
reti

end
```

Ya que aprendimos a comunicar los dos microprocesadores, y además hacer que trabajen conjuntamente. Es tiempo de completar nuestro proyecto e incorporarle el tercer módulo. Para ello vamos a seguir el mismo esquema general de conexión que se ha seguido:

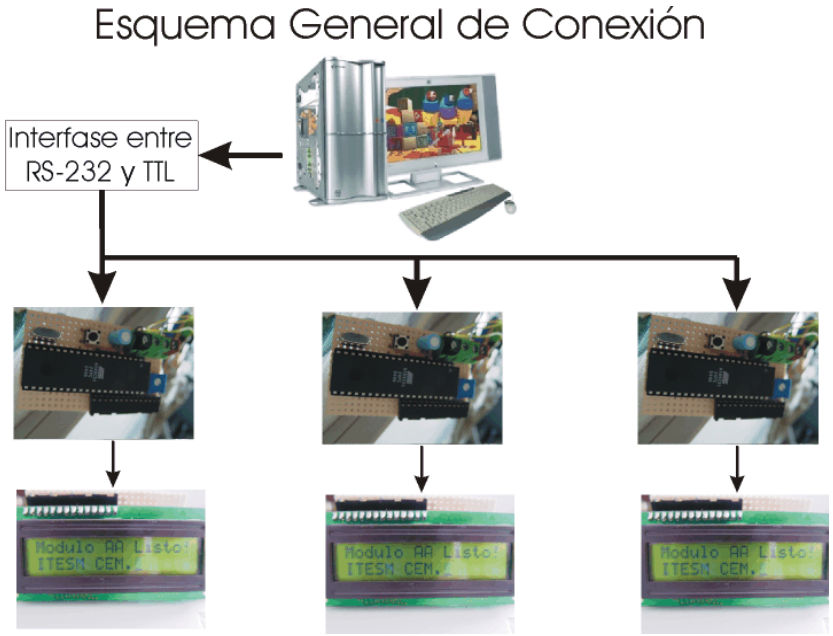

Figura 8.43 - Esquema general de conexión del experimento que demuestra que es posible conectar varios microprocesadores en un mismo canal, para que trabajen colaborativamente.

El programa que utilizaremos en el tercer microcontrolador (con dirección 03h) es el siguiente:

Lti003_03.asm

```
LCD_DI equ P1.0                 ;Bit del microcontrolador que se
                                ;conecta al pin DI del Display de LCD
LCD_En equ P1.1                 ;Bit del microcontrolador que se
                                ;conecta al pin EN del Display de LCD
SERIALBUFFER equ 0FF00h

org 0000h
   sjmp INICIO

org 0023h
   jmp INTERRUPCION_SERIAL

INICIO:
   call INICIALIZA_LCD

   mov DPTR,#CADENA1            ;Muestra mensaje de Bienvenida
   call MUESTRA_MENSAJE_LCD
   call BAJA_RENGLON_LCD

   setb EA
   setb ES
   mov R2,#00h
   mov R3,#01h
   mov P3,#0FFh
   call INICIALIZA_9600_8_1SP_11Mhz

FIN: sjmp FIN
;=================================================
; CADENAS DE CARACTÉRES
;=================================================
CADENA0: db '1234567890123456'
CADENA1: db 'Modulo 03 Listo!'



;-------------------------------------------------------------------------------
;INTERRUPCION_SERIAL -- Transmite cadena de caracteres, Recibe
; una cadena de caracteres y la manda a
; un buffer serial para su futura
; operacion
;
;USA: PSW.5 Acc MODIFICA: DPTR
;
```

```
;Variables de Entrada:
; DPTR -- Dirección del Mensaje
; SERIALBUFFER -- Dirección de inicio del Buffer Serial
;Variables de Salida:
; NINGUNA
;
;Observaciones: Puerto serial a 9600bps, Hay que definir la
; direccion del buffer serial, el separador de
; comandos es enter 13 decimal = 0Dh
;
;Ejemplo:
; SERIALBUFFER equ 0FF00h
;
;MENSAJE1:
; DB 'Salvador Macias Hernandez'
; DB 00h
;-------------------------------------------------------------------------------
INTERRUPCION_SERIAL:
   JB TI, DATOENVIADO          ;Verificando Mótivo de Interrupción
   JB RI, DATORECIBIDO

DATOENVIADO:
   clr TI
reti

DATORECIBIDO:
   clr RI
   mov Acc,SBUF
   cjne A,#03h, VERIFICA_TURNO
   mov R3, #03h
   sjmp SALIR_INT_SERIAL

VERIFICA_TURNO:
   cjne R3, #03h, SALIR_INT_SERIAL          ;Verifica si es el turno
                                            ;de este micro
   DESPLEGAR_INFO:
      mov Acc, SBUF
      clr LCD_En
      setb LCD_DI                          ;Decimos que es dato ;)
      mov P0,Acc
      setb LCD_En
      call ESPERA_50uS
      clr LCD_En
```

```
        inc R2
        cjne R2, #16, SALIR_INT_SERIAL      ;Verifica que no se hayan
        mov SBUF, #04h                      ;escrito los 16 caracteres de
        mov R3,#04h                         ;este display, si sí pasa turno a 04h
        sjmp SALIR_INT_SERIAL

SALIR_INT_SERIAL:
reti

end
```

Como era de esperarse, se obtuvieron muy buenos resultados. Se conectaron los tres módulos a un canal común, junto con la computadora. Así como lo indica el esquema general de conexión (figura 8.43), y que puede observarse también en la siguiente fotografía:

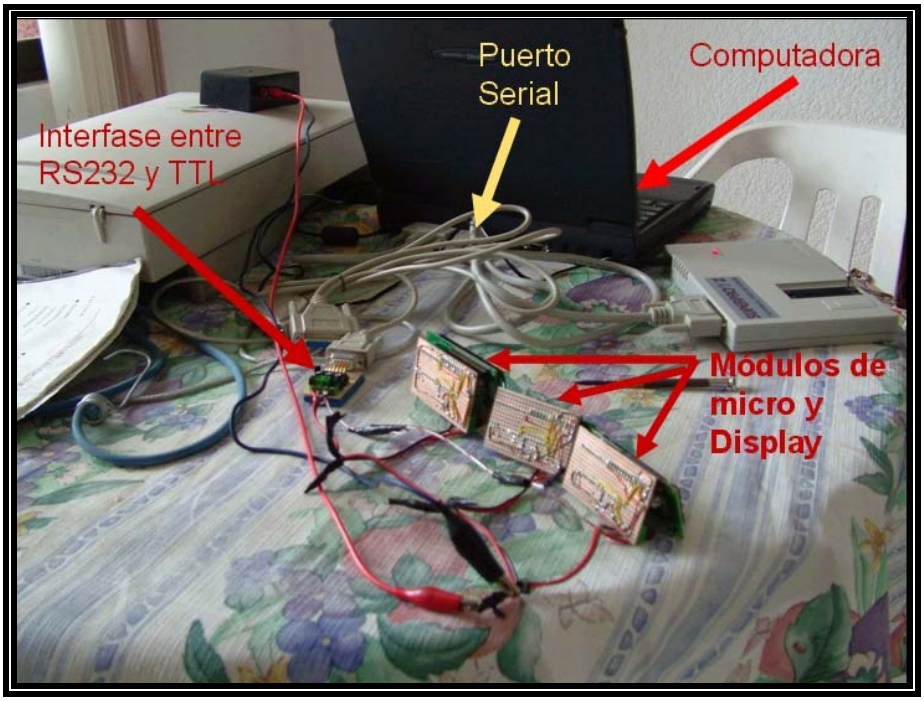

Figura 8.44 - Conexión realizada en la práctica, para el tercer experimento.

La foto que se muestra a continuación, es la evidencia de que el proyecto funcionó adecuadamente. En éste se puso como texto "Salvador macias Hernandez 717320 ITESM C.E.M. " Y así fue como apareció en los displays:

Figura 8.45 - Fotografía que muestra el resultado satisfactorio del experimento que demuestra que es viable el hacer que varios microcontroladores trabajen colaborativamente en la línea transportadora inteligente. También demuestra que no es necesaria una computadora central que coordine absolutamente todo. En este experimento se tecleó mi nombre, matricula y escuela donde estudio. Y lo cual fue desplegado en tiempo real y en los tres displays que se muestran.

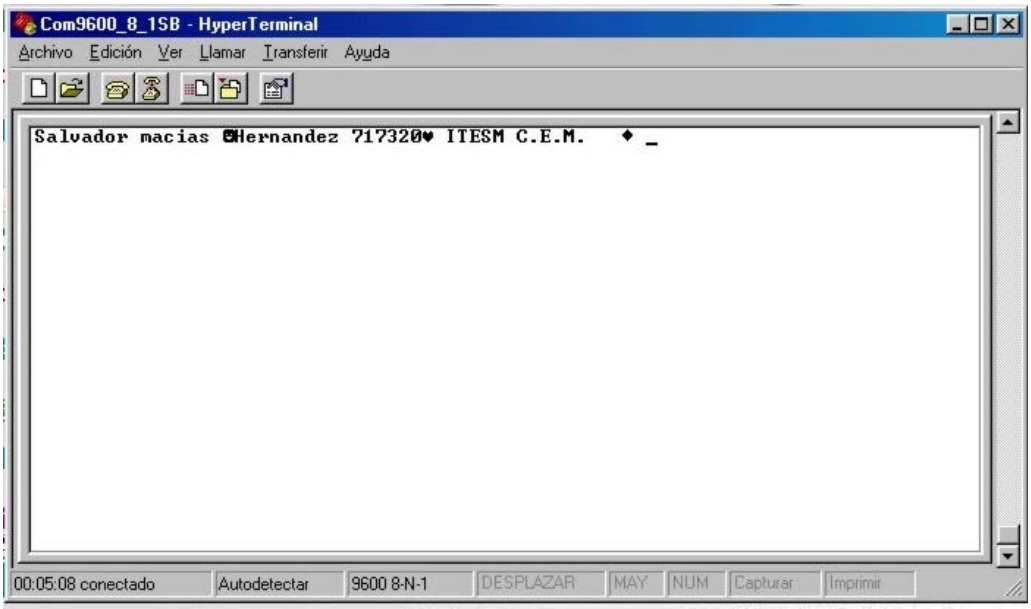

Figura 8.46 - Impresión de pantalla que muestra los datos que estuvieron presentes en el canal de comunicación. Vemos la información que se desplegó en los diferentes displays. También podemos observar los códigos de control que se emitieron para coordinar el trabajo de los microcontroladores.

REFERENCIAS

[1] Salvador Macías Hernández; **DISEÑO Y ANÁLISIS DE UN PUENTE H**; http://www.famaher.com/SubDominios/SMH/Proyectos/PuenteH/PuenteH.htm

[2] F.R.Bahía Blanca; **AMPLIFICADORES EN CONFIGURACIÓN PUENTE O CONFIGURACIÓN**; Universidad Tecnológica Nacional.

[3] Martin D. Seyer; **RS-232 MADE EASY CONECTING COMPUTERS, PRINTERS, TERMINALS, AND MODEMS**; Prentice Hall, Segunda Edición

[4] Salvador Macías Hernández; **PROGRAMACIÓN DE UN DISPLAY DE LCD**; http://www.famaher.com/SubDominios/SMH/Escolares/DisplayLCD/DisplayLCD.htm

[5] Hantronix; **CHARACTER MODULE INITIALIZATION**

[6] Salvador Macías Hernández; **COMUNICACIÓN DE LA PC A UN DISPLAY DE LCD;** http://www.famaher.com/SubDominios/SMH/Proyectos/TesisLTI/Microprocesador/ComPCaLCD/ComPCaLCD.htm

[7] Salvador Macías Hernández; **COMUNICACIÓN DE TRES MICROCONTRADORES**; http://www.famaher.com/SubDominios/SMH/Proyectos/TesisLTI/Microprocesador/Com3Micros/Com3Micros.htm

[8] Salvador Macías Hernández; **PRE - PRESENTACIÓN DE LA TESIS DISEÑO DE UNA LÍNEA TRANSPORTADORA INTELIGENTE**; http://www.famaher.com/SubDominios/SMH/Proyectos/TesisLTI/PrePresentacion/PrePresentacion.htm