PUERTO PARALELO

EXPERIMENTOS

PARTE 1

Carlos Rey Marcos

Con la colaboración de Mueen Sajjad y Diego Rodríguez

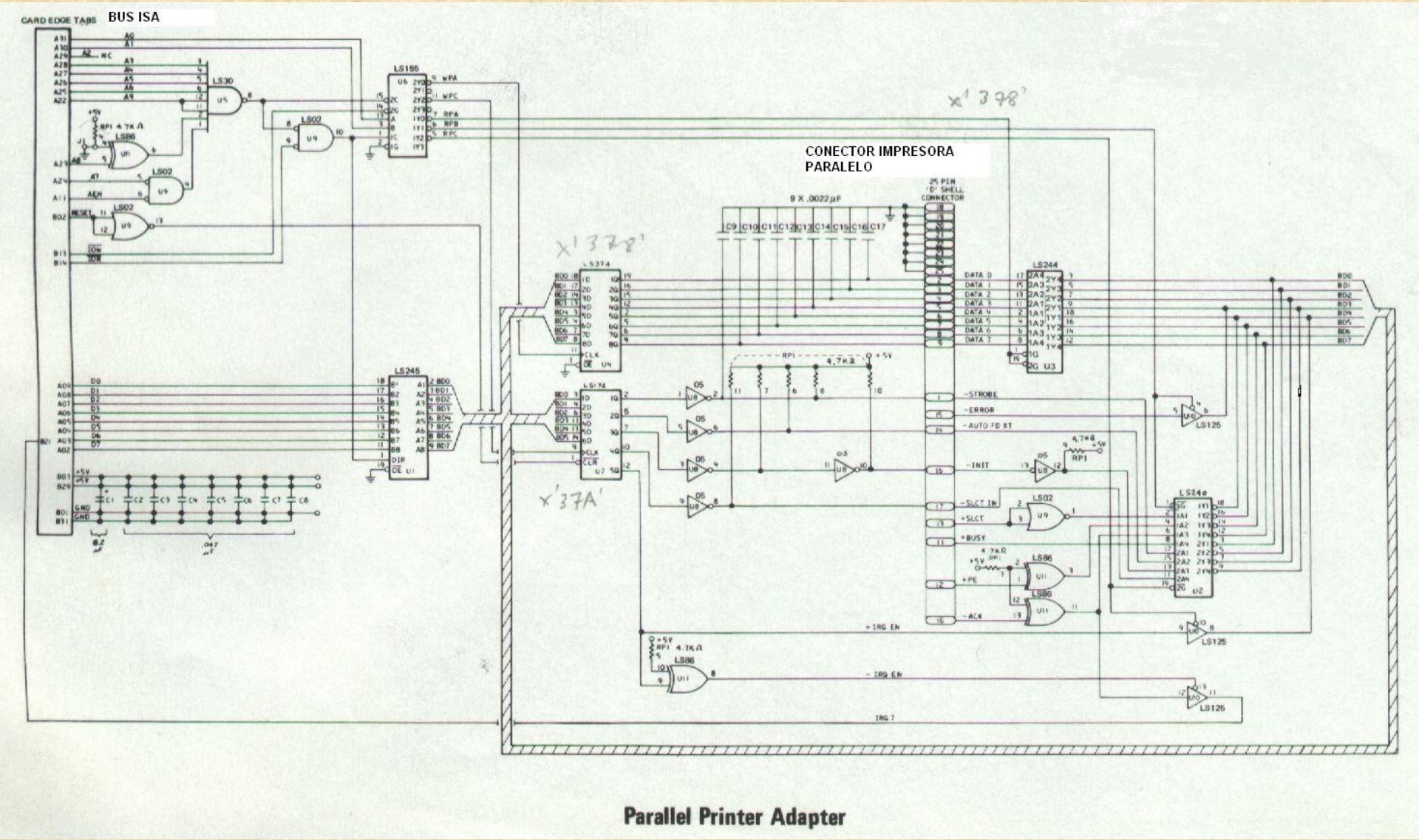
Motivación: compartir las posibilidades del puerto paralelo para aplicaciones creativas en el mundo real

Metodología: descripción de ejemplos reales y conceptos básicos, definición de las áreas de conocimiento implicadas, búsqueda de alternativas prácticas

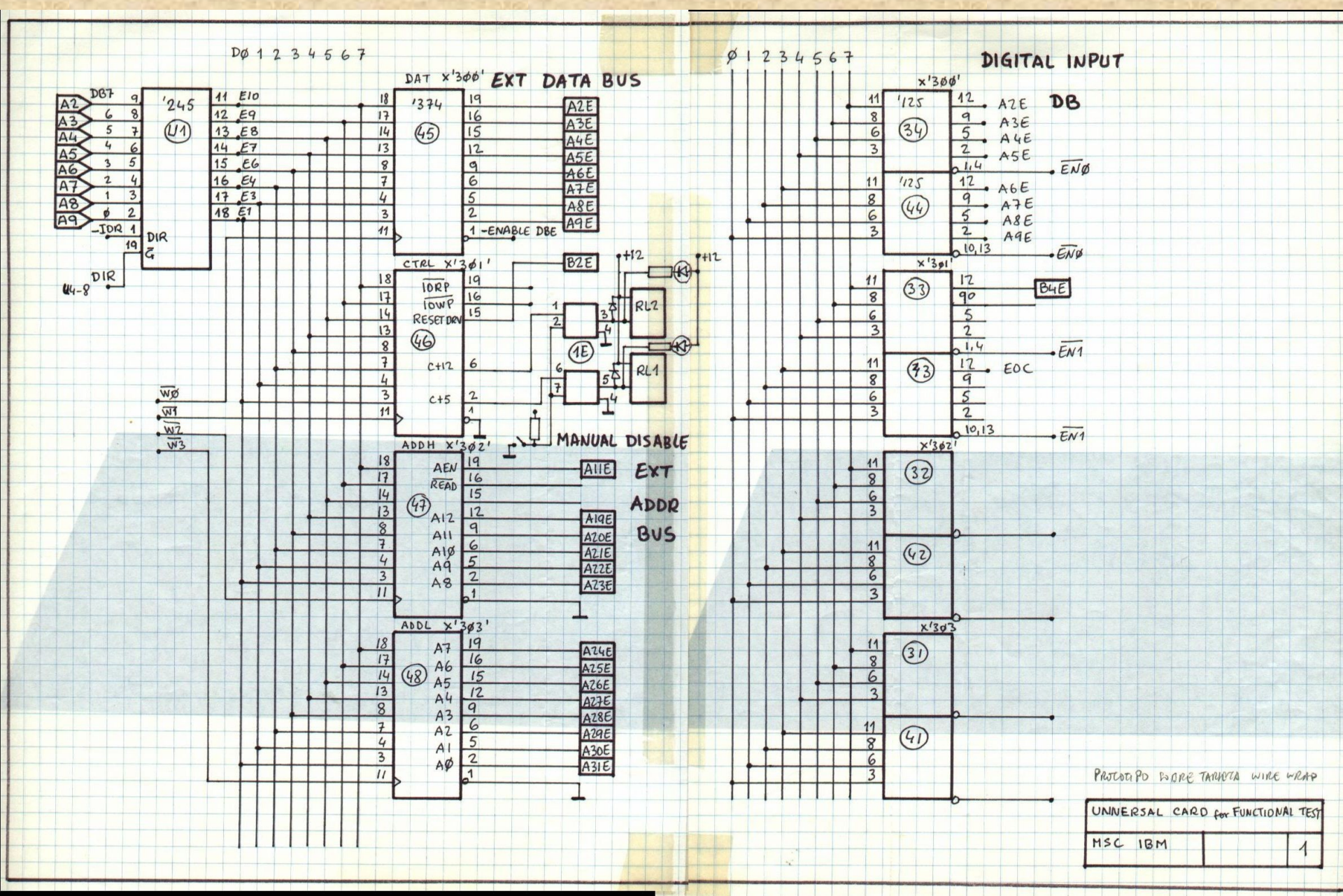
Recursos: PC con sistema operativo GNU/Linux, compilador de lenguaje C, tarjeta de desarrollo de prototipos, módulos funcionales preensamblados, ejemplos de código, componentes electrónicos, soldador.

Desarrollo: participación activa orientada a las aplicaciones de interés personal

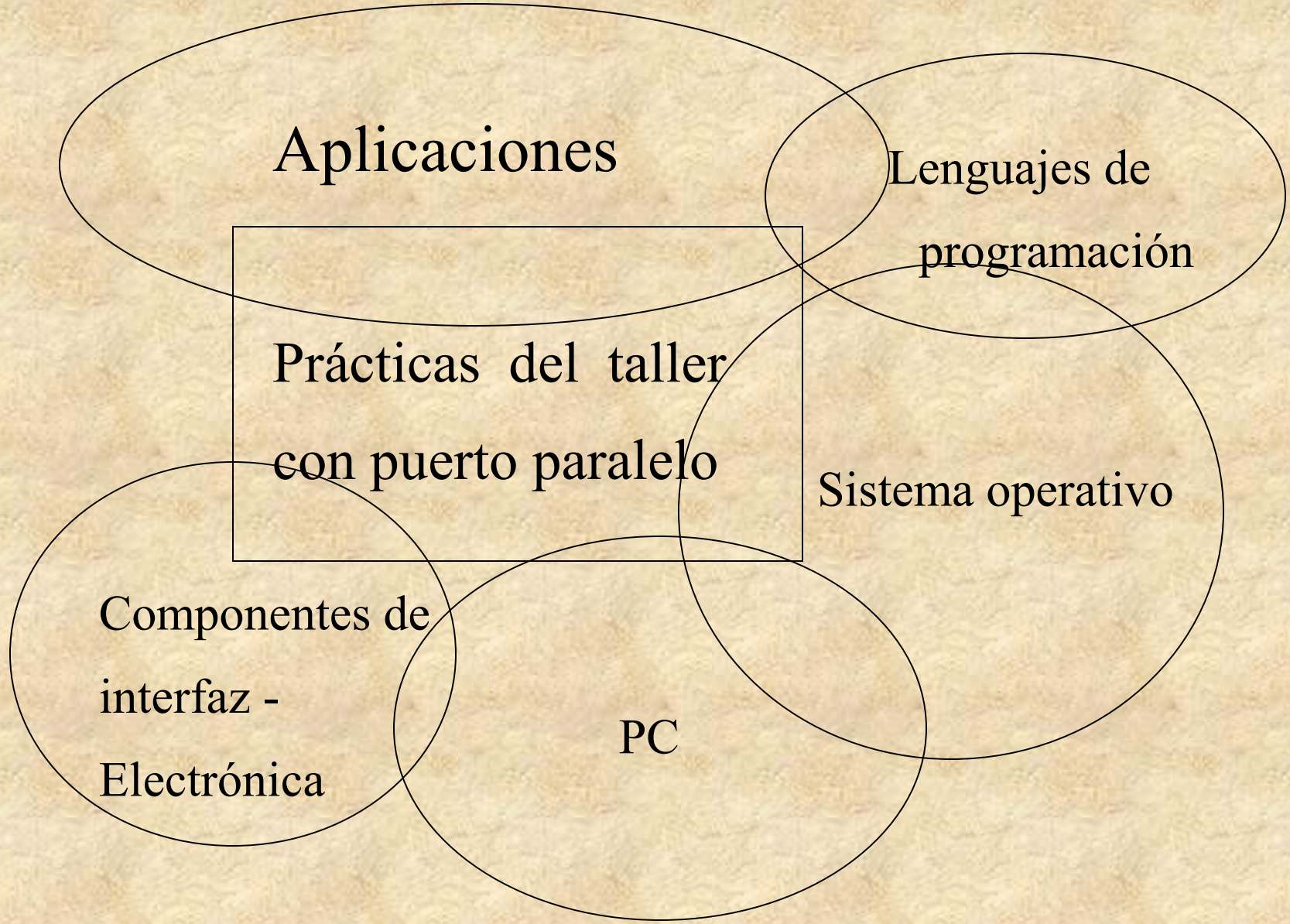
Esquema del controlador puerto paralelo



Esquema de la Tarjeta de Verificación de Circuitos



Entornos


Aplicaciones
Lenguajes de programación
Prácticas del taller con puerto paralelo
Sistema operativo
Componentes de interfaz - Electrónica
PC

Tipos de comunicación de datos

Paralelo

Requiere un número elevado de hilos para la transmisión. El ancho puede ser 4,8,16,32 o 64 bits

Serie_

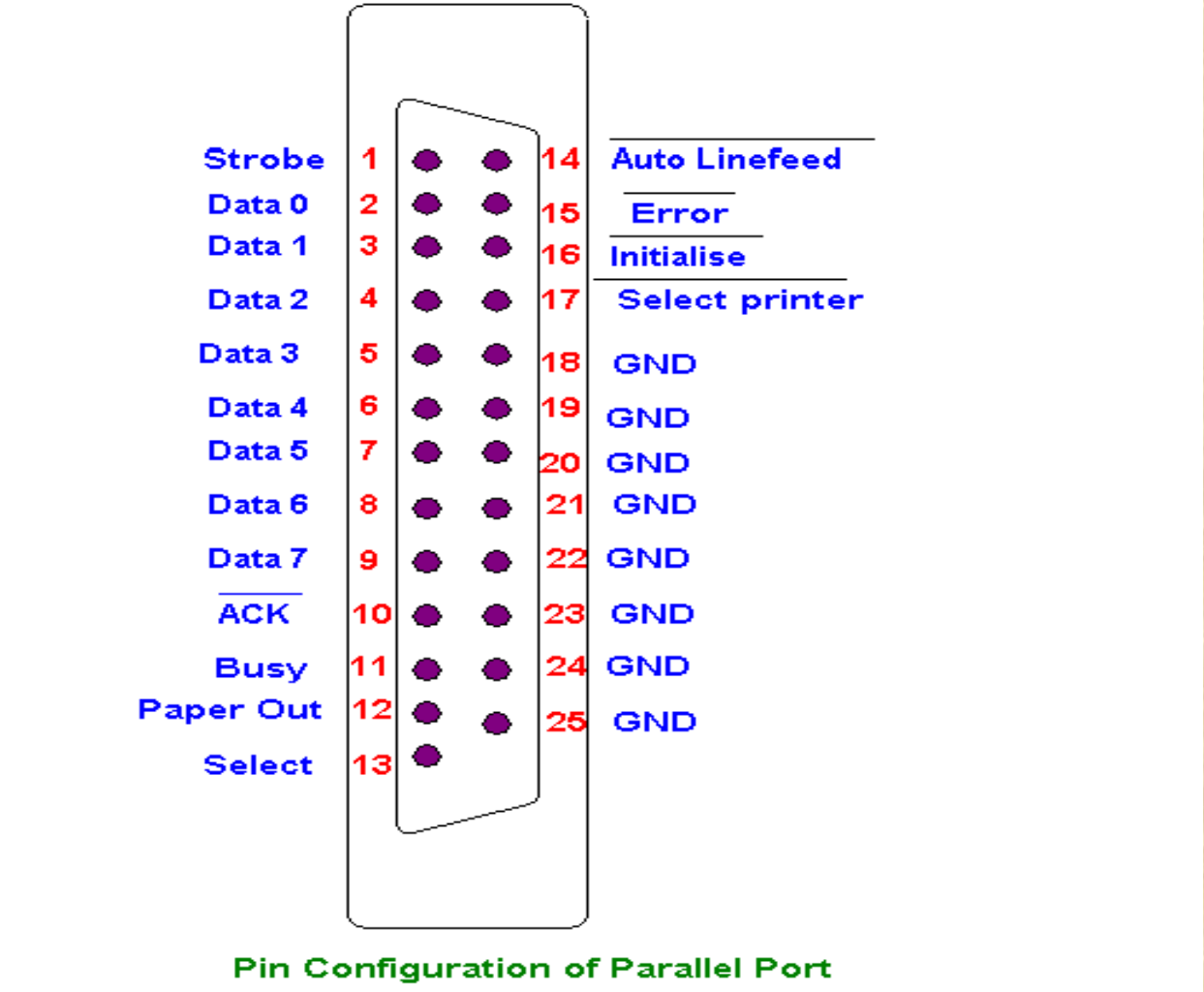
Normalmente requiere sólo 2 hilos para transmisión en una dirección. Uno hilo es para datos y otro para tierra.

Cómo identificar el puerto paralelo en el PC IBM

- Normalmente el conector está situado en la parte trasera del ordenador

Es un conector de 25 pines hembra

Configuración de los pines del puerto paralelo



Pines hembra

Grupos de pines del puerto paralelo

1.- Datos
2.- Estado
3.- Control

1.-Datos:

- Incluye del pin 2 al pin 9 con identificador Data-0 a Data-7
- En las primeras versiones eran pines de salida.

Dirección	Bit no.	Nombre	Número
Base +0 (0x378)	7	Data 7	Pin 9
	6	Data 6	Pin 8
	5	Data 5	Pin 7
	4	Data 4	Pin 6
	3	Data 3	Pin 5
	2	Data 2	Pin 4
	1	Data 1	Pin 3
	0	Data 0	Pin 2

2.- Estado

Los bits de estado son de entrada, sólo se pueden leer

Dirección	Num bit	Nombre	Número Pin
Base +1 (0x379)	7	Busy	Pin 11 **(invertido)**
	6	ACK	Pin 10
	5	Paper Out	Pin 12
	4	Select	Pin 13
	3	Error	Pin 15
	2	IRQ	Pin 21
	1	Reservado	-
	0	Reservado	-

3.- Control

Los bits de control son de lectura/escritura. Para impresora son sólo de escritura.

Dirección	Bit	Nombre	Número
Base +2 (0x37A)	7	No usado	-
	6	No usado	-
	5	Enable **bi-direccional**	-
	4	Enable IRQ **Via linea ACK**	-
	3	Selecciona impresora	Pin 17 **(Invertido)**
	2	Inicializa impresora	Pin 16
	1	Auto alimenta línea	Pin 14 **(Invertido)**
	0	Strobe	Pin 1 **(Invertido)**

Como acceder al puerto paralelo desde programa.

El puerto paralelo tiene asignada una dirección de E/S:

1.- 0x378 (es la dirección más frecuente)
2.- 0x278 (dirección para una segunda impresora)
3.- 0x3BC (para una tercera impresora)

Cómo conocer la dirección del puerto en un PC

- Arrancar y entrar en BIOS. Visualizar la dirección del puerto registrada.
- Si está configurada como auto puede ser:
  1.- Asignación explícita en el BIOS.
  2.- Verificar la dirección en Windows a través de:

  Control panel →System →Device Manager → Ports→Printer port→Properties →Resources →Input/Output Range.

Diferencias entre el espacio de direcciones de memoria y el de direcciones de E/S

En la familia x86 de Intel hay dos espacios de direcciones separados

- **Direcciones de memoria**

  Depende de la cantidad de memoria RAM/ROM instalada en el PC y de las líneas de direcciones del bus.

- **E/S**

  Este espacio de direcciones depende del número de líneas de dirección disponible en el bus para dispositivos de E/S.

Como direccionar estos tipos de direcciones desde C

→ Las direcciones de memoria no están accesible para el programador pero pueden direccionarse mediante notación de punteros

→ Las direcciones de E/S pueden accederse en Turbo C/C++ o C utilizando las siguientes instrucciones:

En Windows Turbo C/C++: inport(), inportb(), outport(), outportb()

En Linux C: ioperm(), inb(), outb()

Relación entre los bits del puerto y los niveles de tensión en los pines.

- Cada bit presente en una dirección tiene relación directa con la tensión que aparece en el pin correspondiente.
  por ejemplo un **'1'** en el registro de una dirección de E/S corresponde a un **nivel lógico alto** en el pin.

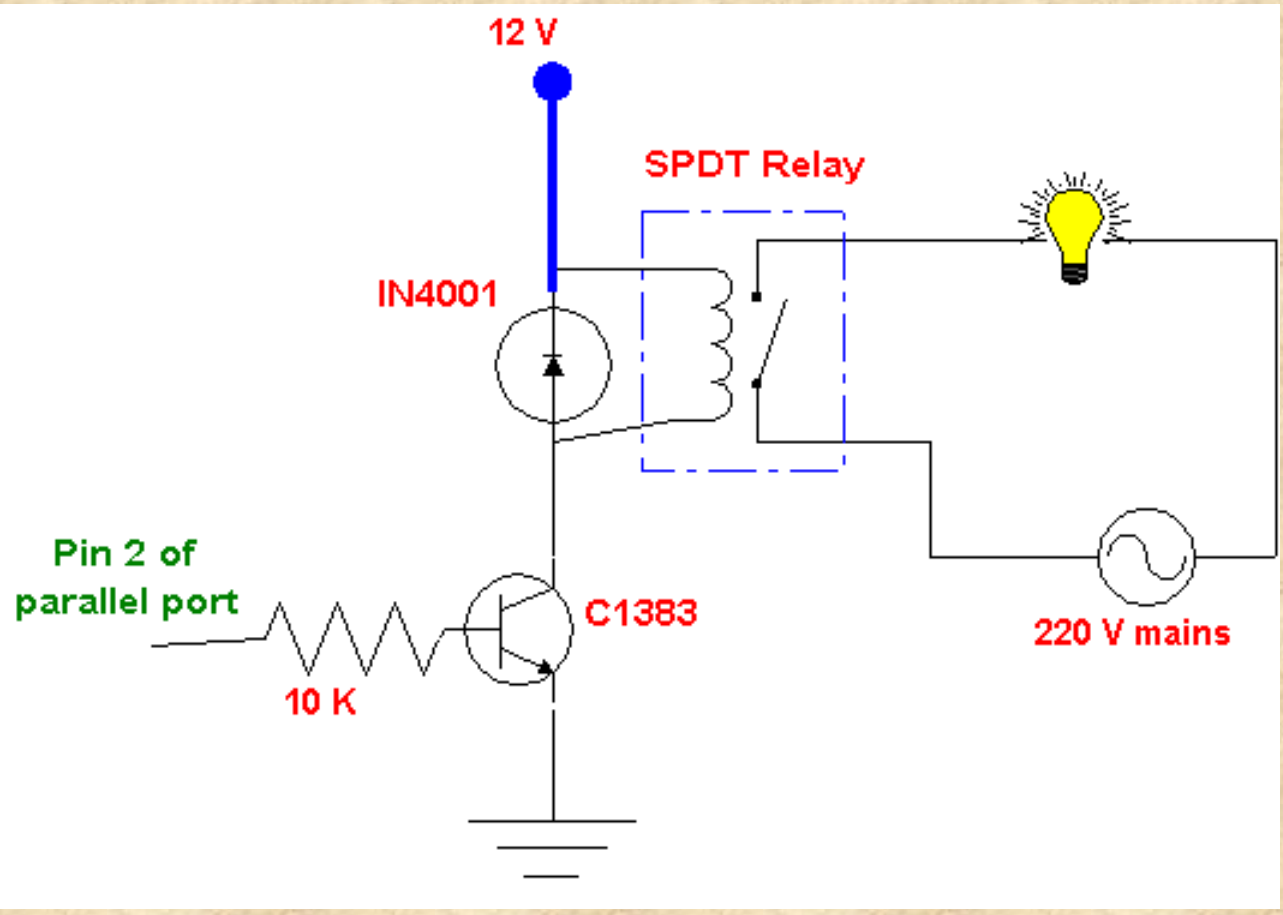
Niveles lógicos usuales en el puerto paralelo.

- Nivel alto ("1") significa tensión de +5V respecto a tierra.
- Nivel bajo ("0") significa tensión 0V respecto a tierra.

Casos especiales en los niveles lógicos

Hay casos en los que se invierten los niveles físicos con relación a su nivel lógico. En estos casos un nivel físico correspondiente a nivel lógico **alto** es de 0V y un nivel físico **bajo** es de 5V. Unos ejemplos son los bits de **Busy y Strobe**.

Demostración del control de luz de una bombilla a través del puerto paralelo mediante un transistor y un relé.



Esquema del circuito

Programa en C para el ejemplo de control

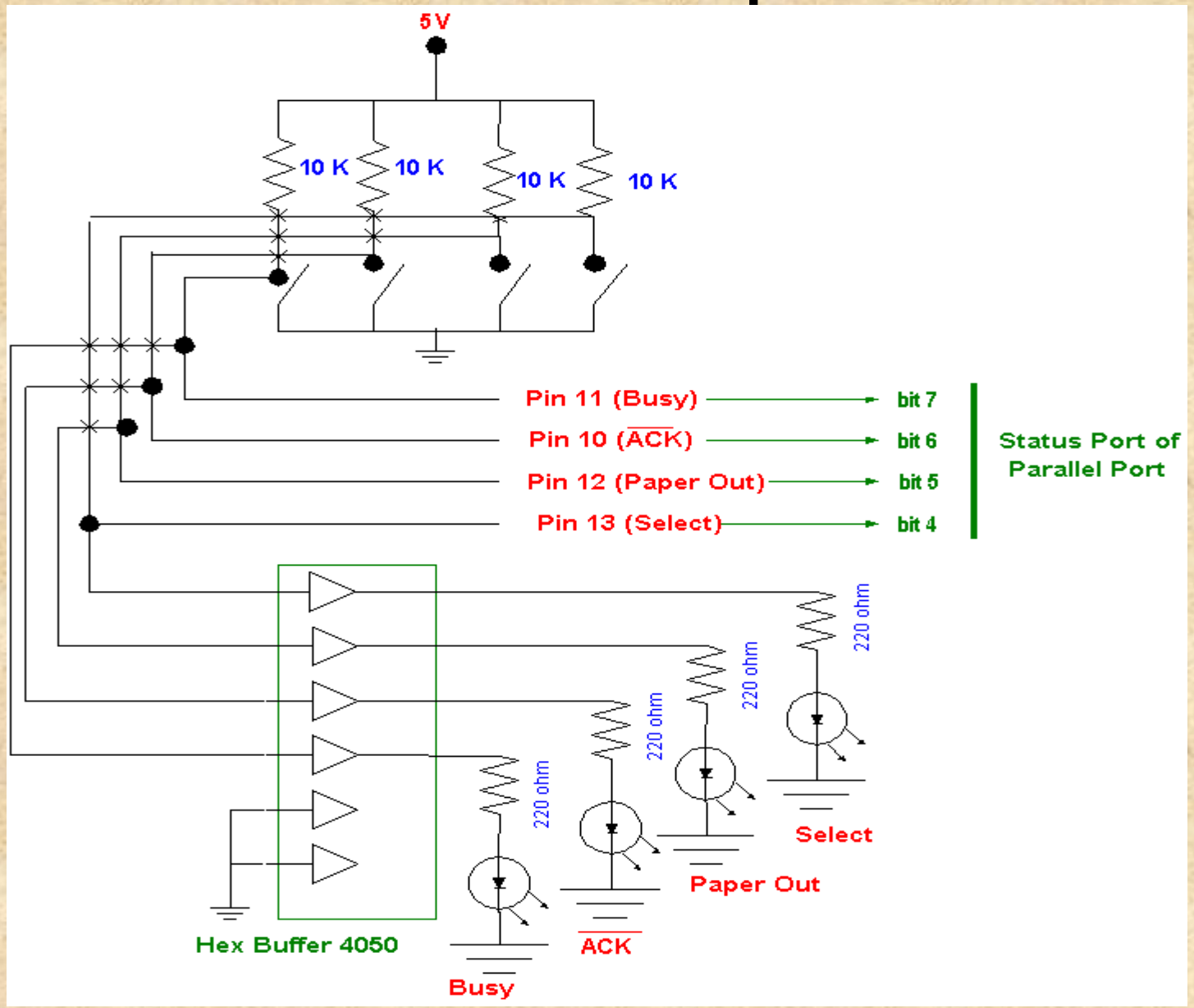
Programa en C para control ON/OFF a través del teclado

```
#include<stdio.h>
void main(void)
{
        outportb(0x378,0x01);     // enciende la bombilla
        getch();                  // espera a que se pulse una tecla
        outportb(0x378,0x00);     // apaga la bombilla
        getch();                  // espera a que se pulse una tecla
}
```

Programa en C para hacer parpadear la luz de la bombilla 5 veces

```
        for (int i=1; i<=5,i++);
{
        outport b (0x378,0x01);   // enciende la bombilla
        delay (1000);             // espera 1000msec = 1sec
        outportb(0x378,0x00);     // apaga la bombilla
}
```

Entrada de datos a través de pulsadores



Rangos de tensión para niveles lógicos alto y bajo

Conexiones del circuito para la prueba

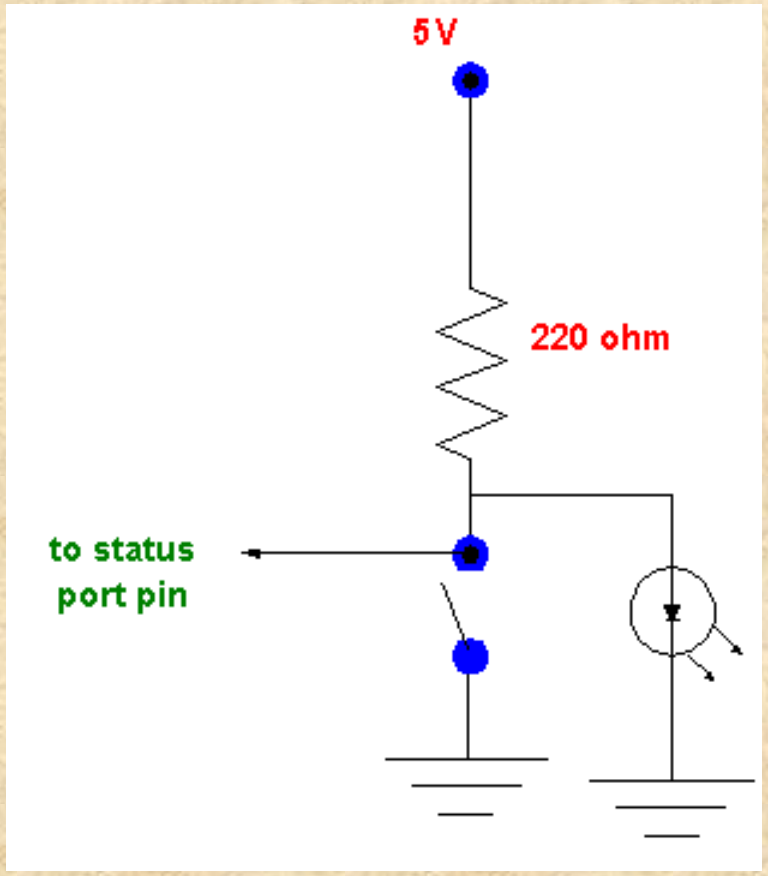
Aunque las conexiones sean correctas los resultados pueden parecer incorrectos. La razón es el rango de tensiones considerado válido en la salida de los puertos.

- El puerto paralelo es compatible con **TTL** (Transistor Transistor Logic) en el cual:

Nivel bajo: es una tensión entre **0 a 0.8 V**

Nivel alto: es una tensión entre **2.0 a 5V**

- Cualquier nivel intermedio es considerado como indefinido.
- Tensiones superiores o inferiores a las definidas pueden dañar los circuitos del puerto u otros circuitos TTL.

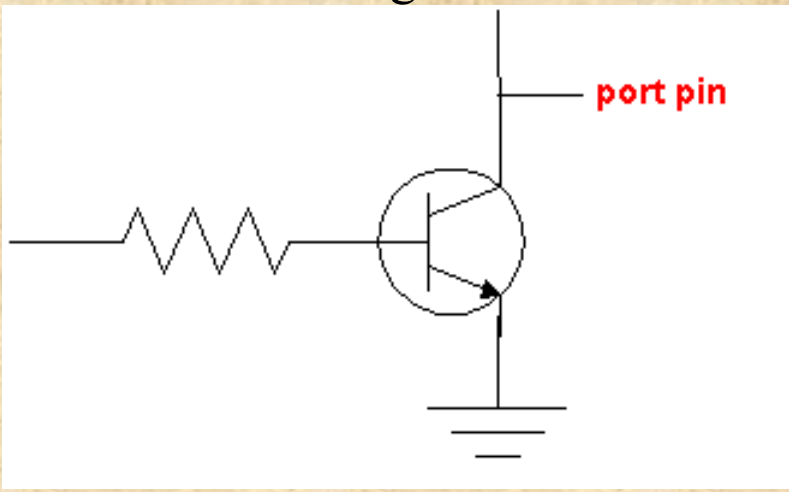


Consideraciones del puerto de control

El puerto de control puede utilizarse para entrada o salida pero hay que tener en cuenta algunas desviaciones

Salida Colector Abierto

En algunas implementaciones los puertos de control utilizan circuitos de colector abierto, que presenta una configuración como la de la figura:



Niveles lógicos de pin con Colector Abierto

1.- Nivel bajo (0V).

Es realmente un cortocircuito a tierra pues el transistor conduce.

2.- Nivel alto (Circuito Abierto)

Presentará la tensión +Vcc a través a una

resistencia (Pull-up)

Como poner un nivel lógico Alto en circuito con Colector Abierto

- **Debería conectarse una resistencia externa (pull-up) a la fuente de alimentación, que podría ser diferente de la del sistema lógico del PC.**
- **En el caso del puerto paralelo no conviene utilizar tensiones superiores a la de Vcc, de +5 V.**

Introducción a la amplificación para conexiones al puerto paralelo

- Se utilizan normalmente amplificadores (buffers) con el propósito de aislar el puerto de la carga.
- Para interfaz digital se suelen utilizar amplificadores o inversores del tipo

CD4050 (buffer), CD4049 (inversor) CMOS

o

- **74 LS245, 74 LS244 TTL**

¿Por qué necesitamos amplificadores?
Introducción a las corrientes de entrada y salida

Corriente de salida (Source current)

En nivel alto el puerto paralelo o cualquier circuito TTL puede proporcionar una coriente máxima de salida manteniendo el nivel lógico. Esta corriente se denomina corriente de salida en nivel alto (source).

- Para el puerto paralelo puede ser de 1mA aproximadamente.
- Por encima de esa corriente la tensión baja de los niveles aceptables. Debe evitarse

Corriente de entrada (Sink current)

En el caso del pin en nivel lógico bajo la corriente que puede admitirse se denomina corriente de entrada (sink current).

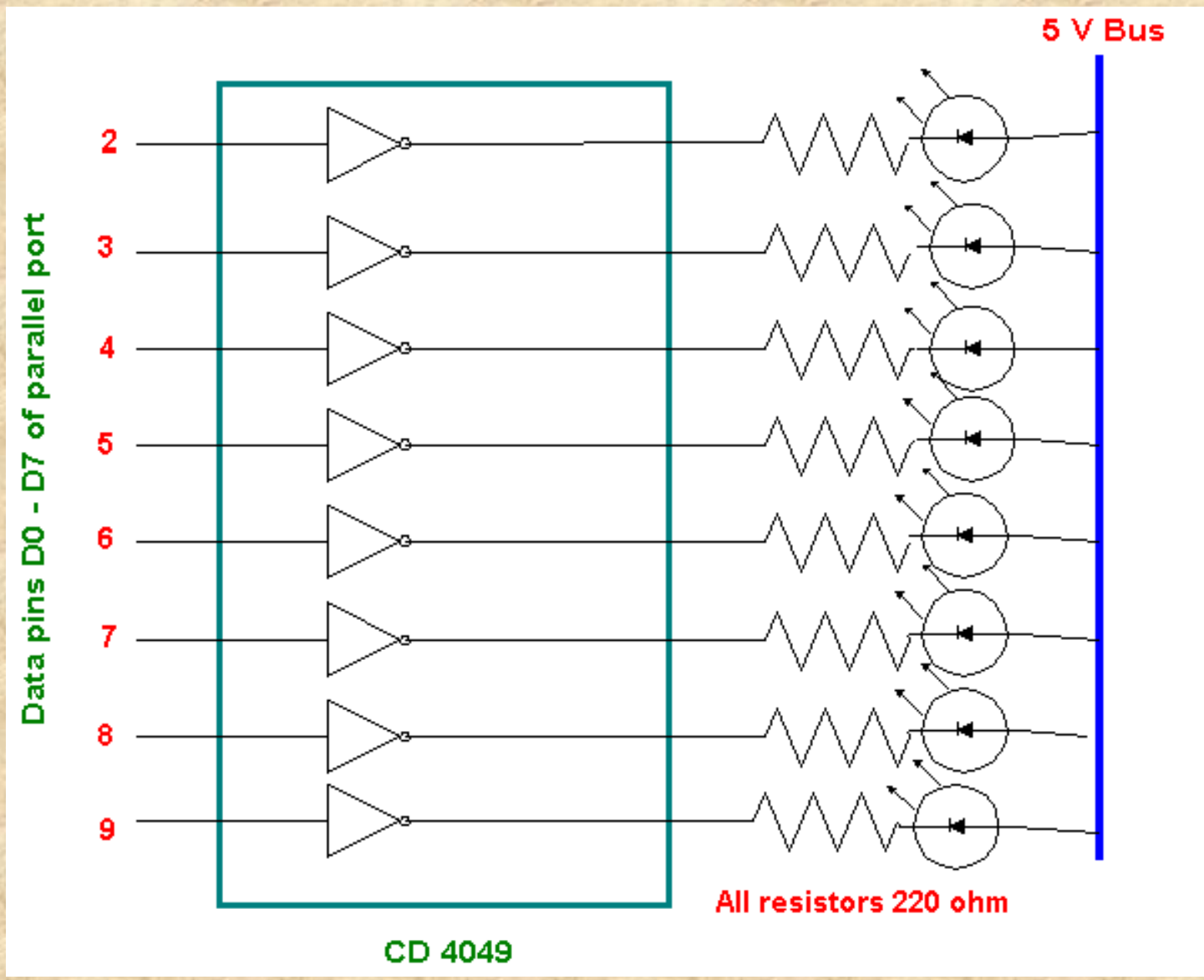
- **Una corriente superior al límite puede dañar el circuito.**
- **La corriente de entrada es normalmente superior a la salida en nivel alto para el mismo tipo de dispositivo.**
- **El rango usual es de unos 10mA para el puerto paralelo.**

Ventaja de utilizar amplificadores

- **Tienen límites más altos de entrada y salida que los de los circuitos normales**.
- **Proporcionan protección a los circuits internos que son más costosos.**

Es una buena práctica utilizar amplificadores (buffers) tanto para entrada como para salida.

Ejemplo de uso de buffers en salida de datos



Progama en C para salida de números

```
/* programa para la salida de números binarios de 0 a 255)
void main (void)
{
 for (int i=0;i<256;i++)
  {
    outportb(0x378,i)
    delay(100);
  }
}
```

Programa para mostrar una luz móvil

```
char k=1;
int i;
for(i=0; i<8; i++) {
        outportb(0x378, k<<i);
        delay(500);
}
k=128;
for(i=0; i<8; i++){
        outportb(0x378, k>>i);
        delay(500);
}
```

PUERTO PARALELO

EXPERIMENTOS

PARTE 2

PRACTICAS

Experimentos de instrumentación

1.-Observación de señales binarias (osciloscopio, analizador).
2.-Detección de cambios, contadores, medida de tiempos.
3.-Ensayos de estímulo-respuesta, medida de tiempos de reacción.
4.-Observación de señales analógicas mediante la conexión de un circuito Conversor Analógico/Digital.
5.-Multiplexión de entradas/salidas para ampliar el número de señales
6.-Tarjeta de verificación de circuitos (TVC)
7.-Interconexión entre PCs (TWC)

Control externo - Domótica

1.- Control de señales binarias.
2.- Conversión Digital/Analógica, amplificador.
3.- Aislamiento, fotoacopladores, relés.
4.- Control de potencia mediante fotoacoplador + Triac.
5.- El PC como controlador conectado a internet:
Servidor, cliente, controladores

Ejercicios del Taller

1.- Cable + extensión de conexiones.
2.- Placa de desarrollo con LEDS.
3.- Entradas digitales: Interruptores, pulsadores.
4.- Salidas digitales: servos, señales binarias variables.
5.- Entradas analógicas: sensores, temperatura, C A/D
6.- Salida de potencia: fotoacopladores + Triac
7.- Salidas: control señal analógica, comparador, C D/A

Bloques del Taller

1. Equipo físico:
   - PC
   - Cable de conexión puerto paralelo a tarjeta de desarrollo
   - Placa de desarrollo y tarjetas de tiras
   - Módulos: alimentación, señales, pulsadores
   - Componentes: resistencias, transistores, triacs, LEDs
   - Soldador y estaño
2. Programas:
   - Sistema operativo Linux Xubuntu
   - Compilador de C para Linux: gcc
   - Funciones de utilidad para los ejemplos
   - Código fuente de los ejemplos
3. Hoja de instrucciones

Precauciones con el hardware

1.-Dibujar el circuito que se va a montar con todos sus componentes antes de empezar a montar.
2.-Montar los componentes con la fuente de alimentación desconectada.
3.-En la primera prueba observar las señales mediante LED o voltímetro.
4.-En los montajes con tensión de red de 220 V realizar las conexiones “con una mano”.

Recomendaciones para los programas

1.-Instalar los paquetes de Desarrollo para que se pueda usar el compilador gcc.
2.-Editar los programas mediante cualquiera de los editores de pantalla completa: nano, emacs,…
3.-Guardar los códigos fuente en la misma carpeta de pruebas.
4.-Para ejecutar se debe acceder como **root** para poder tener acceso a los componentes físicos de la máquina.
5.-Utilizar dentro del programa alguna función que muestre el avance de la ejecución de cada paso (trace)
6.-Utilizar los bloques y funciones propuestos sin modificar, hasta entender su funcionamiento.

Ejercicios segundo día

1.-Lectura de señales variables, presentación gráfica

2.-Salida amplificada por transistor
3.-Conversión digital/analógico mediante resistencias y transistor
4.-Comparación de señales analógicas mediante transistores
5.-Conversión analógico/digital mediante comparación
6.-Presentación de resultados en LCD

Primer programa

```
#include <stdio.h>
#include <unistd.h>
#include <asm/io.h>      // asm/io.h es donde están definidas las funciones ioperm() y outb()
#define DATAPORT 0x378    /* Esta es la direccion más frecuente para el puerto paralelo*/
int main()   {      //Obtenemos permiso de acceso para la direccion de DATAPORT y las 2 siguientes
  if   (ioperm(DATAPORT, 3, 1))   { perror("ioperm"); return 1; }
     while(1)  {                        // Entramos en un bucle infinito
        int input;                      //  Le pedimos al usuario que introduzca un número
        printf ("Introduce un número entre 0 y 255 (-1 para salir)\n"); //Leemos valor. Guarda "input"
        scanf ("%d", &amp;input);              //Si "input" vale "-1" salimos del bucle
        if (input==-1) break;
                    // Si "input" no es "0" -"255" (y no era "-1“) no interesa, volvemos al bucle
        if (input < 0 || input > 255) continue;
        // Si llega aqui es que "input" vale entre "0" y "255" y podemos sacarlo por el puerto paralelo
        outb(input, DATAPORT);
     }      //Antes de terminar el programa dejamos los permisos de acceso a los puertos como estaban
   if (ioperm(DATAPORT, 3, 0)) { perror("ioperm"); return 1;  } //  Termina sin errores
  return 0;
}
```

Segundo programa

```
#include <stdio.h>    #include <unistd.h>   #include <asm/io.h>
#define DATAPORT 0x378      /* lpt1 */ // Esta es la direccion más frecuente para el puerto paralelo
#define DELAY 150000 // Duración de cada paso en microsegundos
#define REPEAT 4      // Numero de veces que se repite cada secuencia
void Blink(); void Alternate();  void ZigZag();  // Cabeceras de las Funciones utilizadas en el programa
int main()   {
  if (ioperm(DATAPORT, 3, 1)) {perror("ioperm"); return 1;}  //Obtenemos permiso acceso direccion DATAPORT  y 2 más
  int mode=0;     //  mode indica que secuencia es la que se esta ejecutando
  while(1)  {
    int i;
    for  (i = 0 ; i < REPEAT; i++)   {
      int reset=0;      //Comprueba el valor de mode
      switch (mode)     {       //Si vale 0 toca parpadeo
        case 0:      Blink();       break;       //Si vale 1 toca alternar pares e impares
        case 1:      Alternate();  break;       //Si vale 2 toca hacer un ZigZag
        case 2:      ZigZag();     break;       //Si vale otra cosa, vovemos a empezar. Para ello       // ponemos mode a -1
        default:     mode=-1;
      }
    }   //Despues de repetir REPEAT veces una secuencia pasamos a la siguiente sumandole 1 a mode
   mode++;
  }
  if (ioperm(DATAPORT, 3, 0)) {perror("ioperm"); return 1; }      //Volvemos a cerrar los puertos
  return 0;
}
```

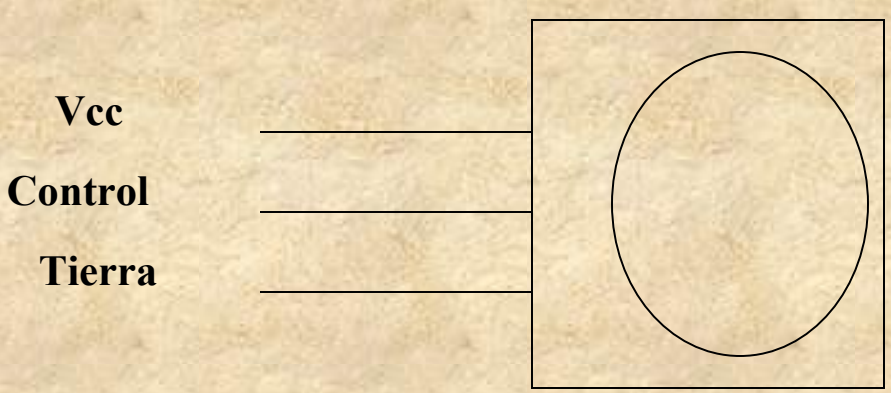
Funciones 1

```
void Blink()   {              // Esta función hace parpadear los LEDs.
   outb(255,DATAPORT);   // Ponemos todos los LEDs a 1

   usleep(2*DELAY);          // Y esperamos 2*DELAY microsegundos

   outb(0,DATAPORT);       // Luego ponemos todos los LEDs a 0

   usleep(2*DELAY);         // Y volvemos a esperar
 }
void Alternate() {    // Esta función enciende alternativamente los LEDs pares y los impares
  int i;
  for(i=0;i<4;i++)  {          // repetimos cuatro veces
    int output = 85;           // Ponemos 85 (01010101 en binario),iluminan los LEDs impares
    outb(output,DATAPORT);
    usleep(2*DELAY);     // Esperamos 2*DELAY
    output = output << 1;  // Desplazamos bits de output a la izquierda, nos queda "10101010"
    outb(output,DATAPORT); //Al pasarlo al puerto paralelo se iluminan los LEDs pares
    usleep(2*DELAY);
   }
}
```

Funciones 2

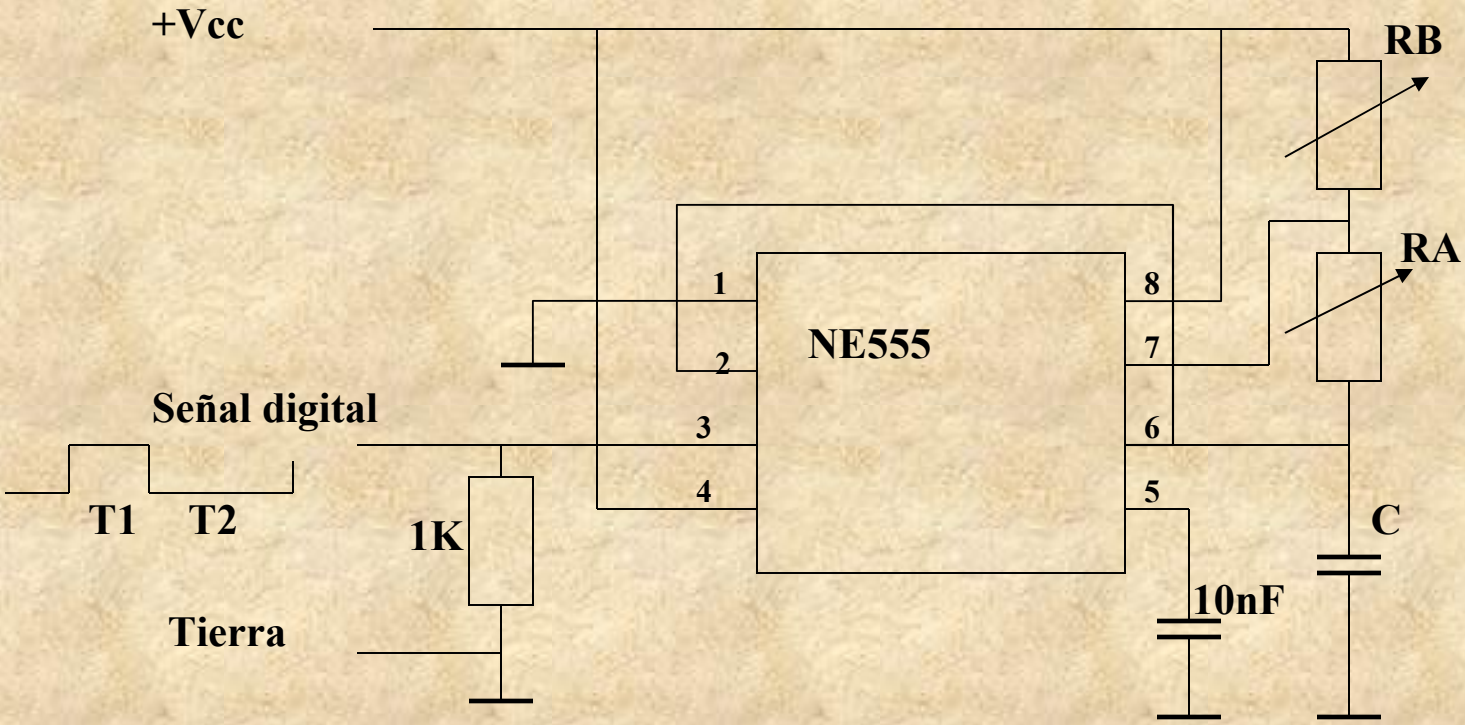
```
void ZigZag()  {  //Esta función enciende las luces al estilo del coche fantástico

int output = 1;     //output solo tiene el último bit distinto de 0
int direction = 1;  // direction indica el sentido de movimiento de la luz: 1 izquierda, 0 derecha.

while (output !=0)  {   // Empezamos hacia la derecha

   outb(output, DATAPORT);   // Sacamos output por el puerto
   usleep(DELAY);   // Si dirección es distinto de cero movemos bits output a la izquierda
   if (direction)
      output = output << 1;    //Si no, los movemos hacia la derecha
   else
      output = output >> 1; //Si output vale 256: ha salido luz por izq ->cambiar dirección
   if(output==256)    {      output = output >> 1;      direction = !direction;    }
  }
}
```

Control de servomotor



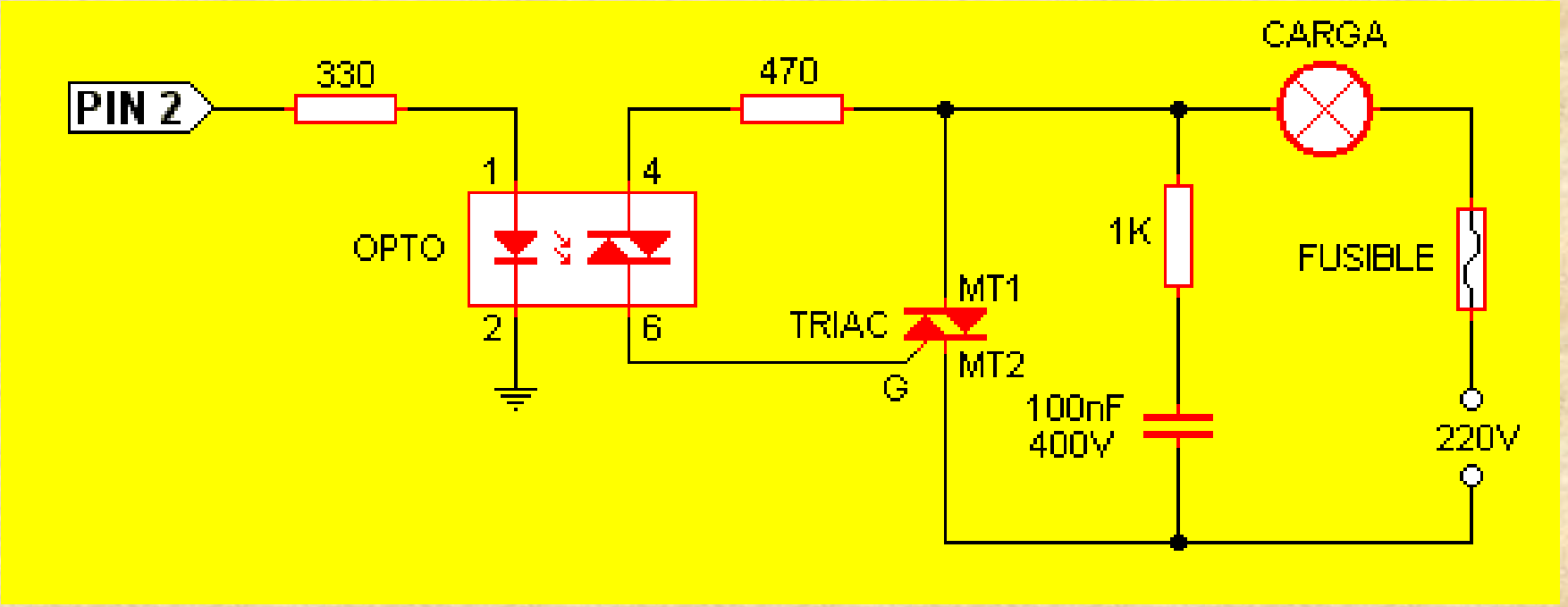
Lectura de señal digital

Oscilador de señal basado en NE555



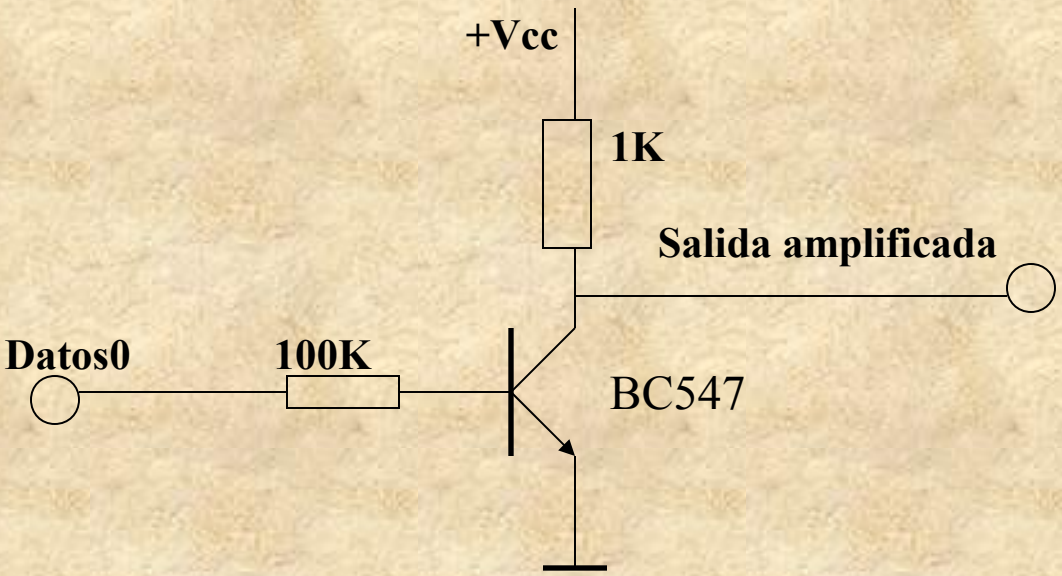
T1 = 0,7 x (RA+RB) x C T2 = 0,7 x RB x C

Control de Triac

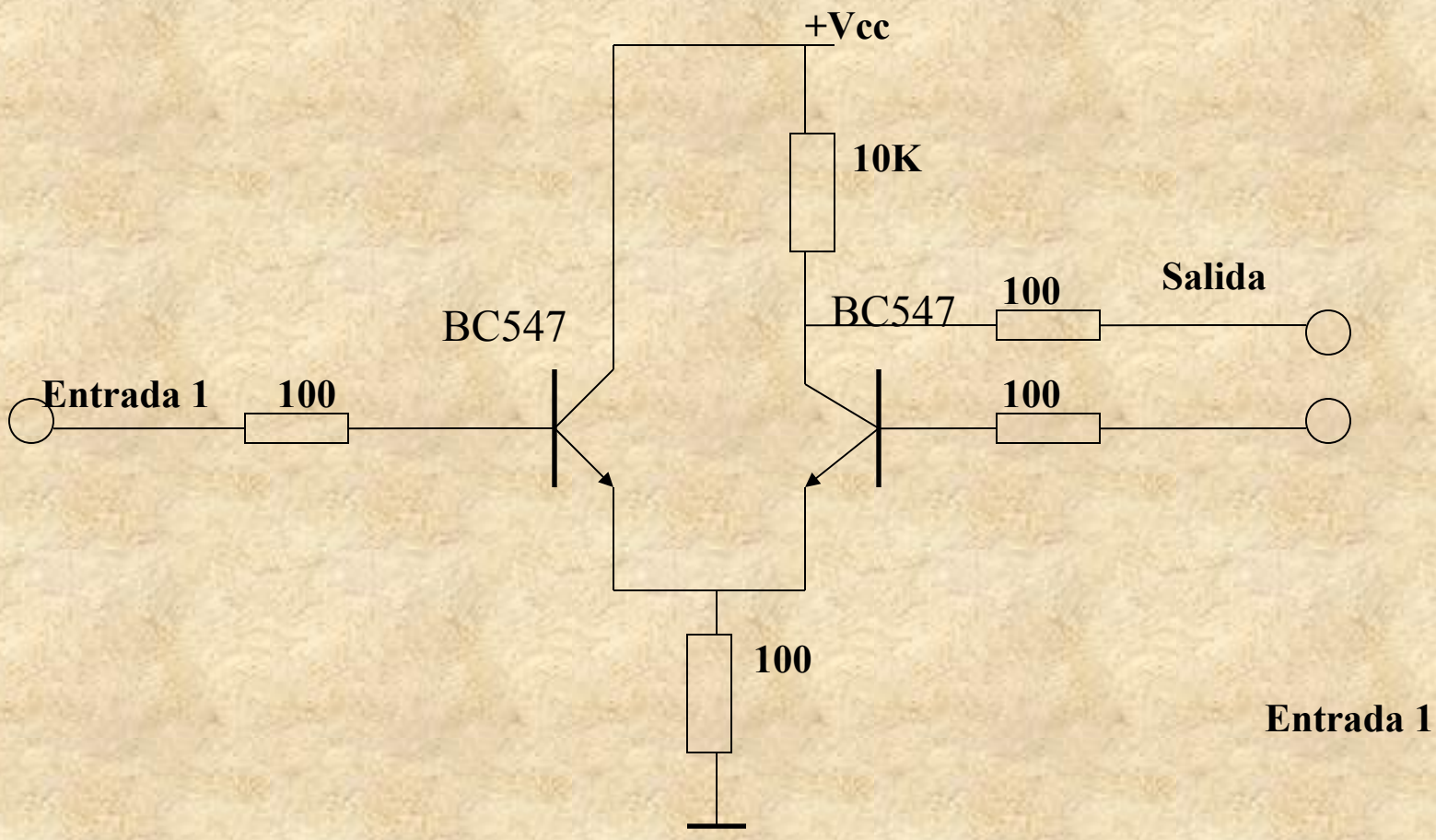


OPTO = MOC3021, TRIAC = SC146M

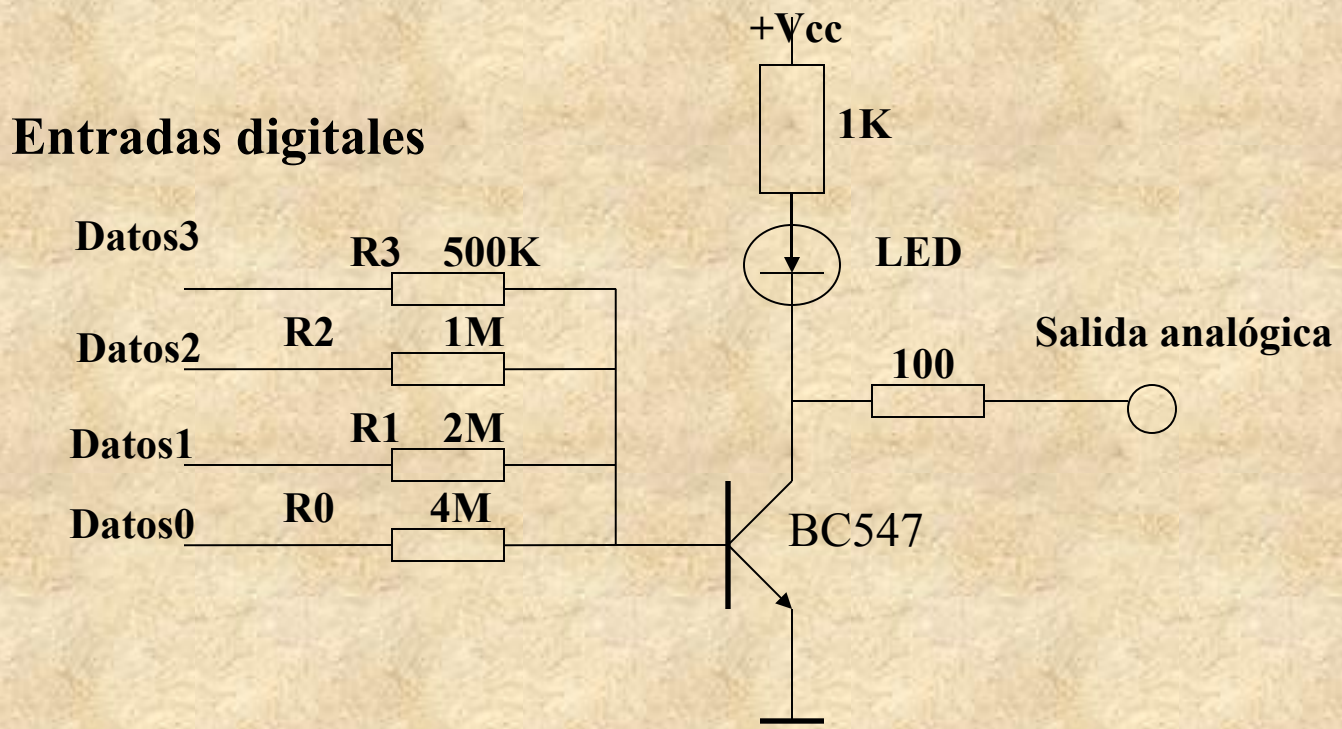
Salida amplificada con transistor



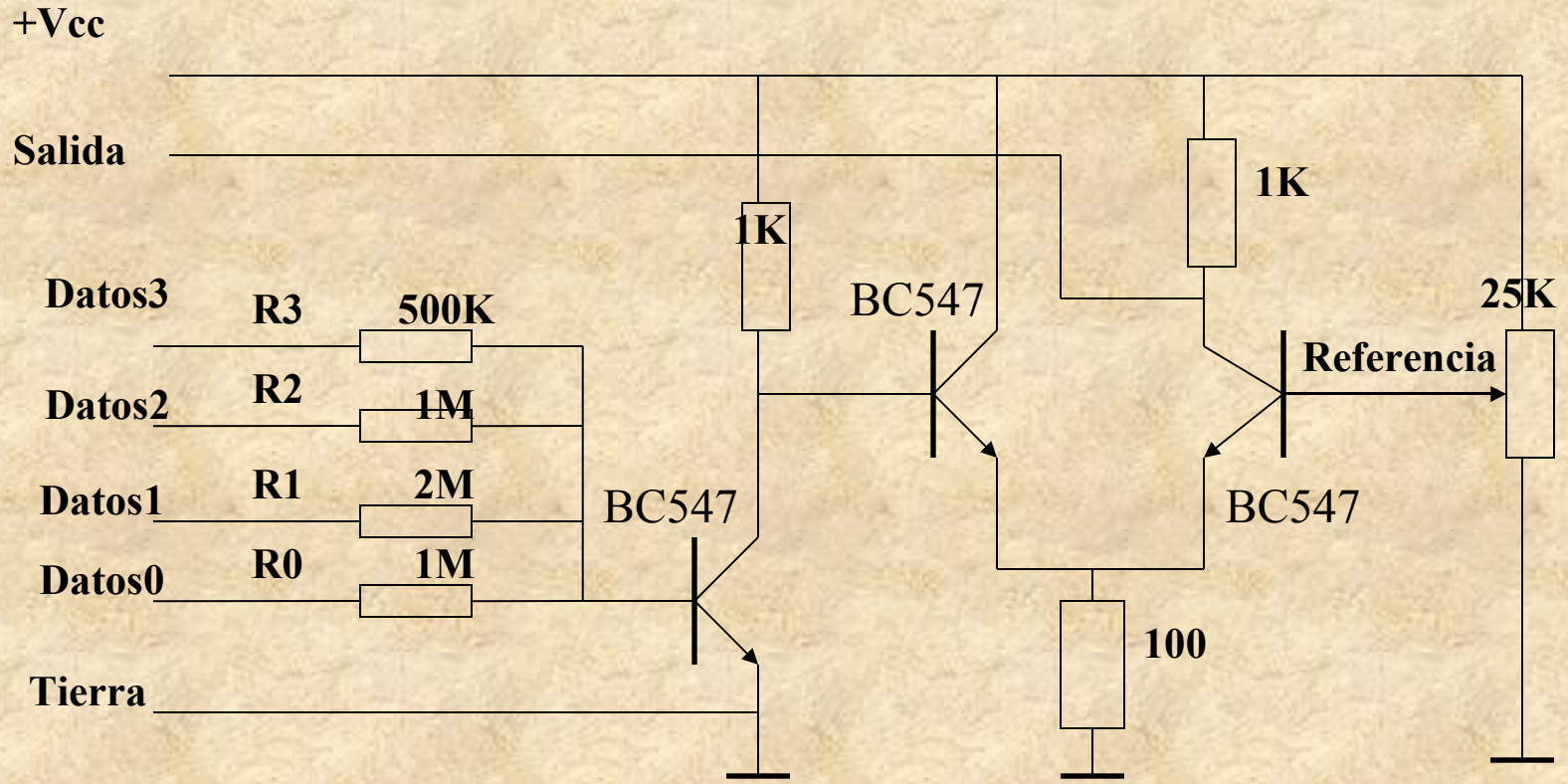
Comparador con transistores



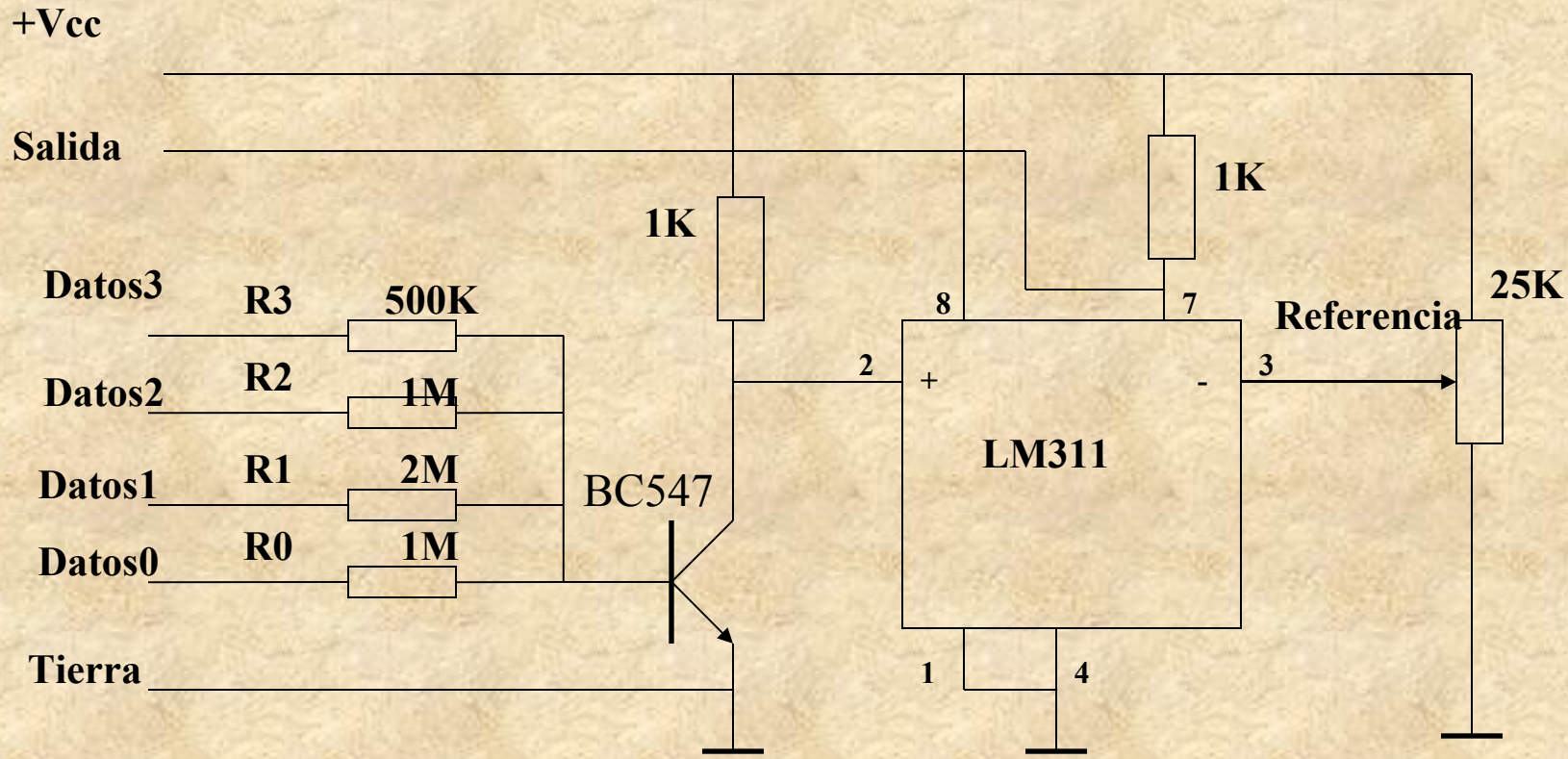
Conversor D/A elemental



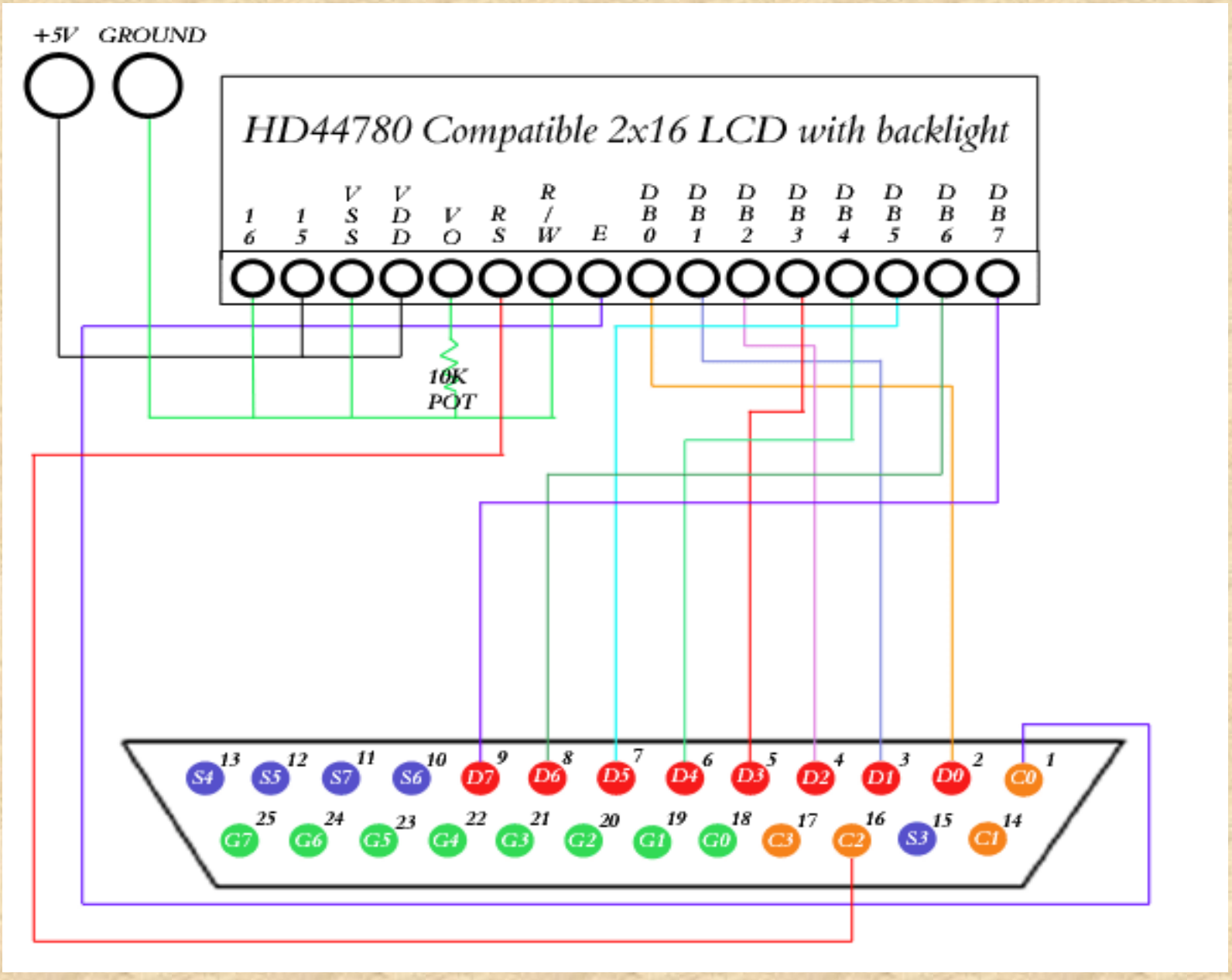
Conversor A/D elemental



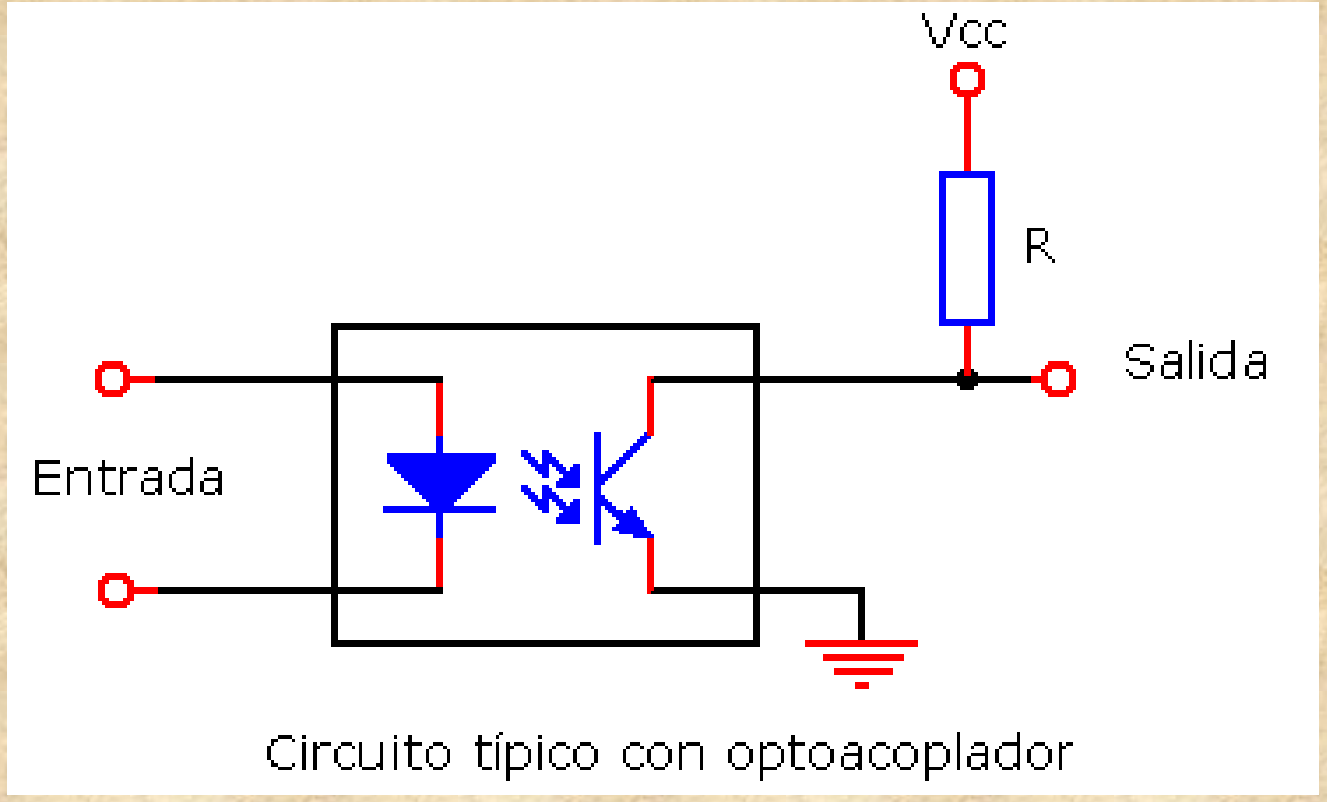
Conversor A/D con LM311



Salida display LCD

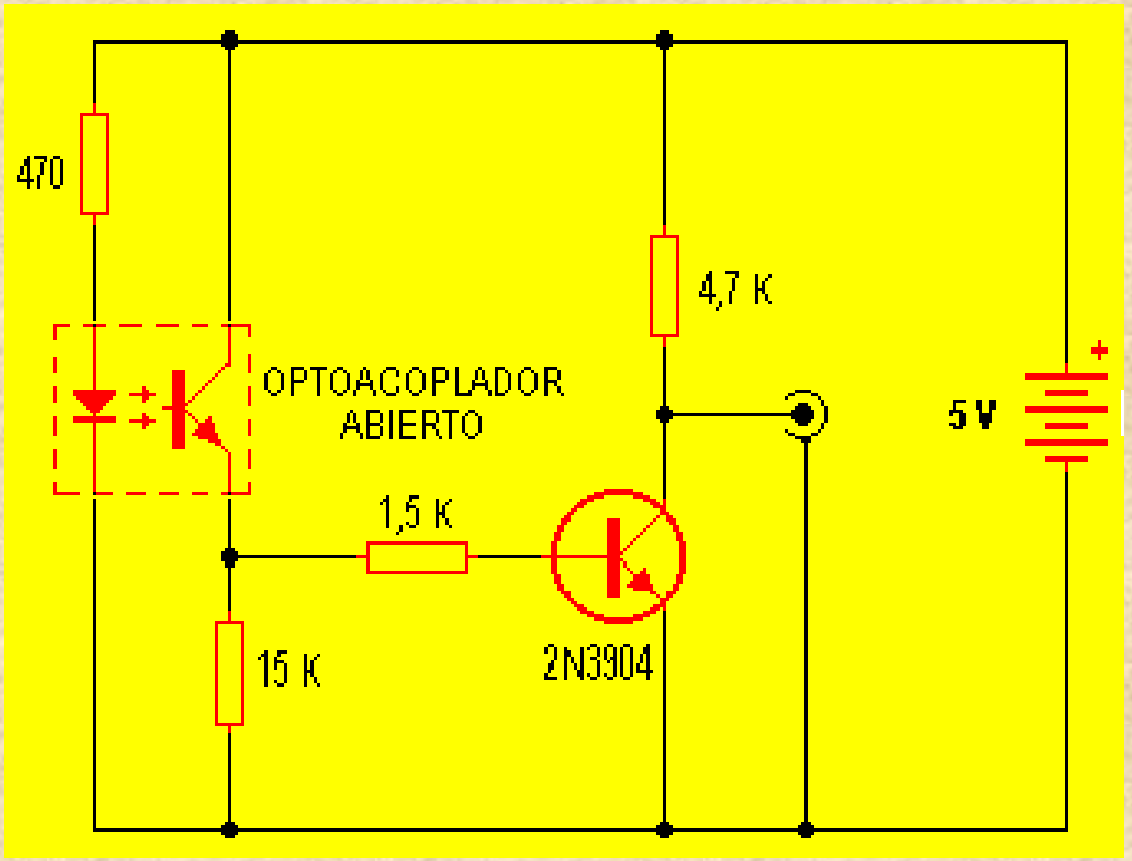


Entrada aislada

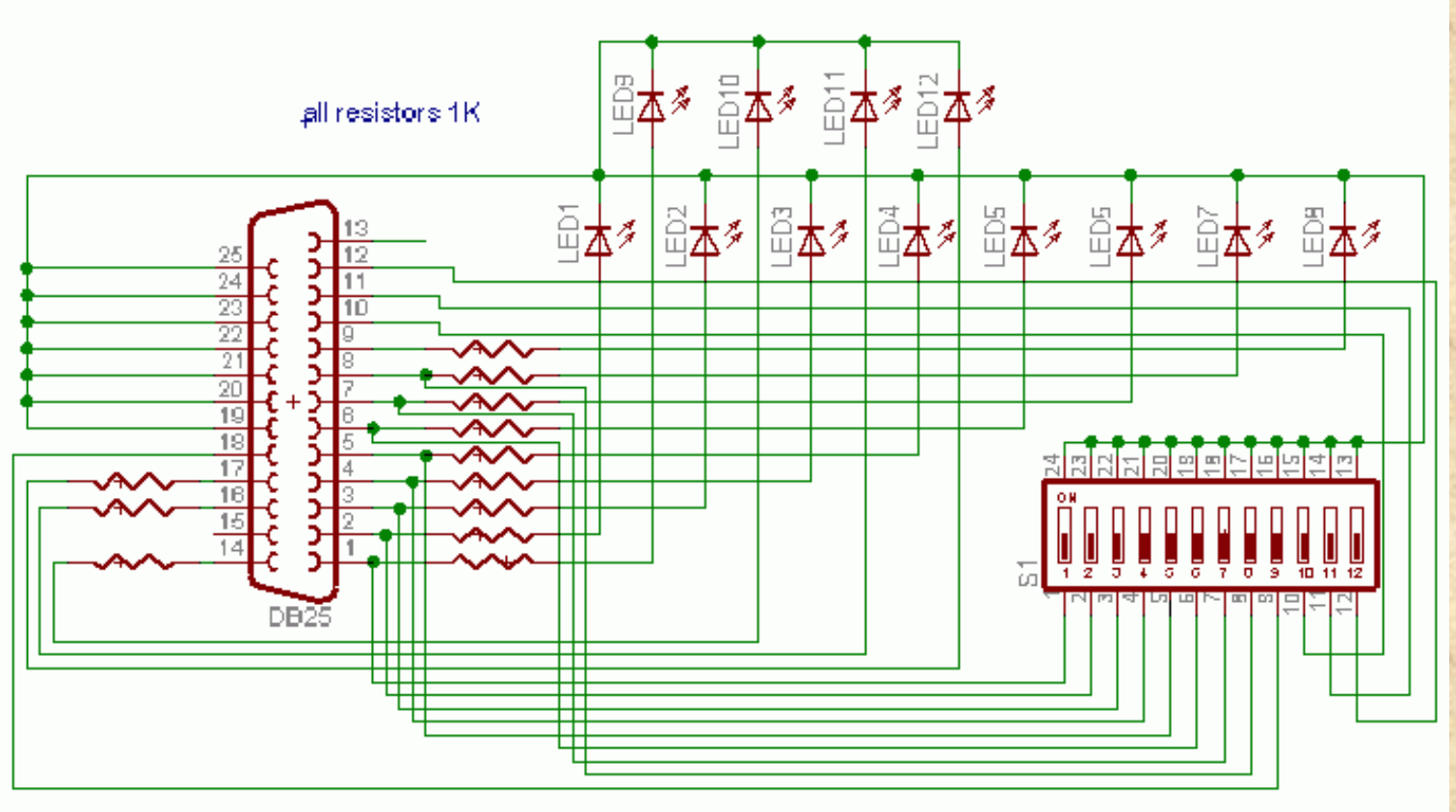


Circuito típico con optoacoplador

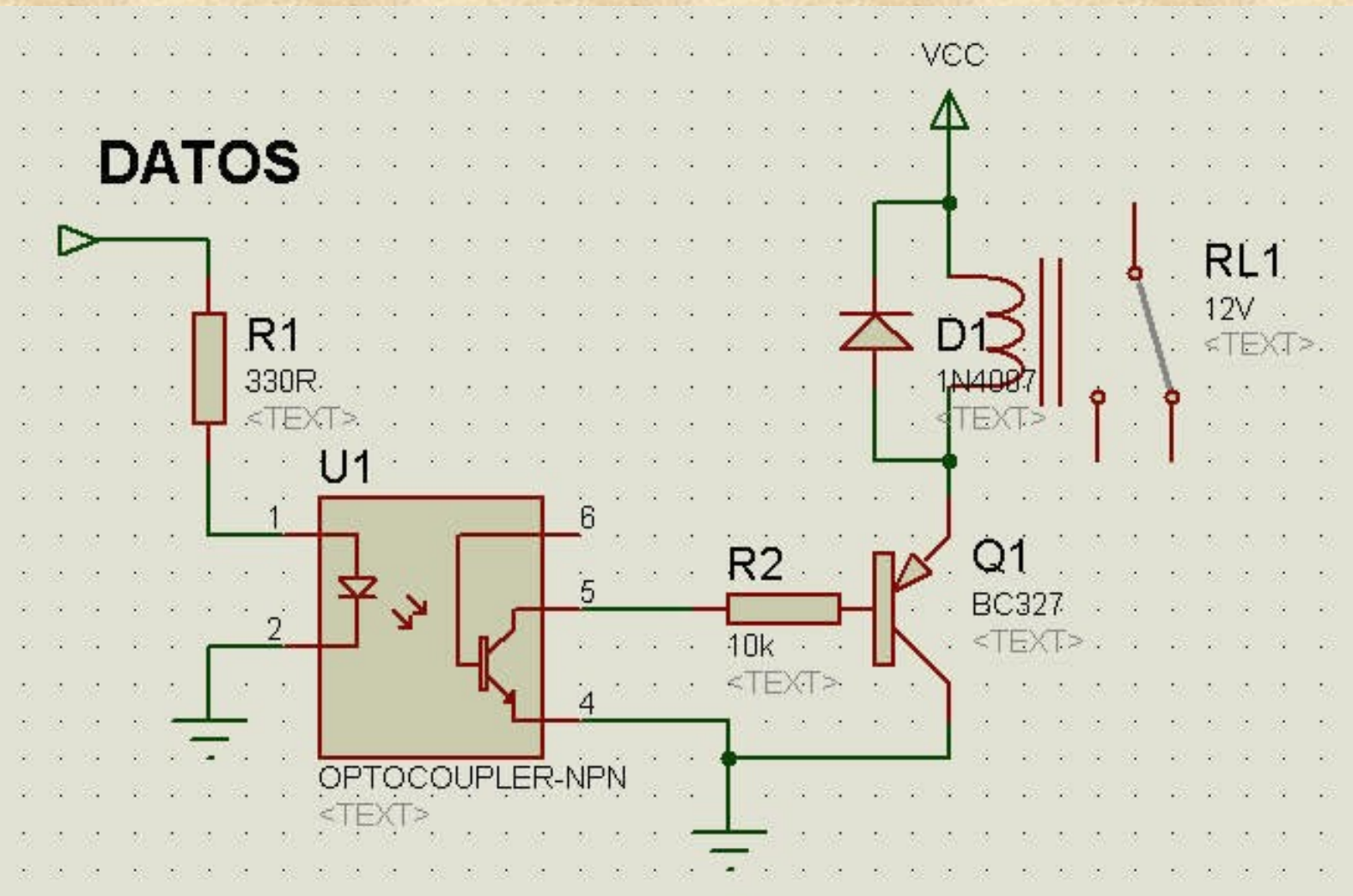
Salida aislada



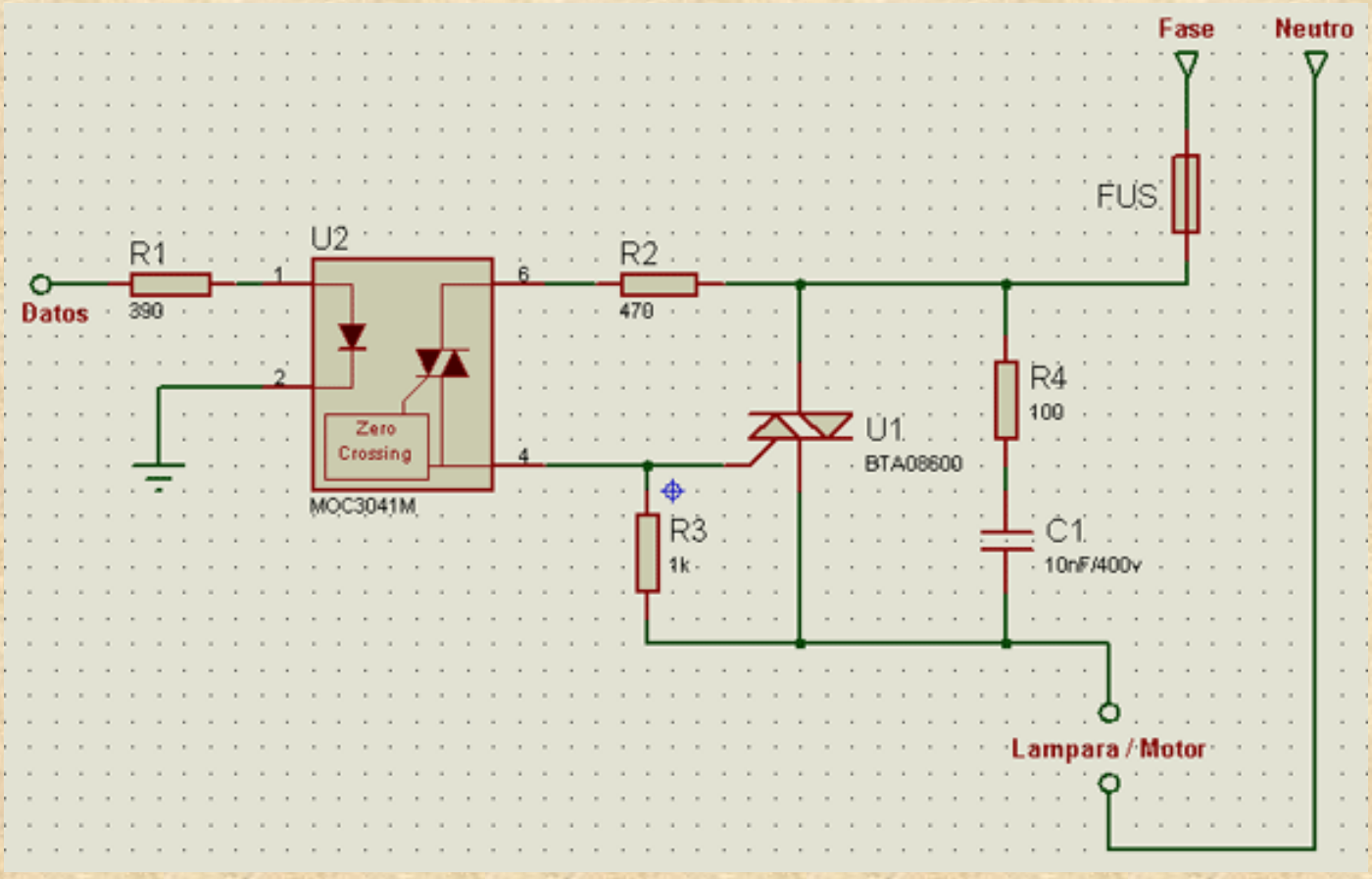
Placa de interruptores y LEDS



Control de Relé



Control de Triac



PUERTO PARALELO

EXPERIMENTOS

PARTE 3

TECNICAS AVANZADAS DE CONEXION DEL PUERTO PARALELO

Introducción al Puerto Paralelo Bi-direccional

- En la actualidad los puertos paralelos del PC son bidireccionales.
- Bi-direccional significa que se puede leer y escribir 8 bits.
- Se puede verificar el tipo de puerto de un PC entrando en la configuración del BIOS. Se verificará la dirección y el tipo de puerto:

SPP	Puerto paralelo estándar. No bidireccional
Bi-directional	Puerto paralelo bidireccional estándar.
EPP	Enhanced Parallel Port
ECP	Extended Capabilities Parallel Port

¿Cómo configurar la bidireccionalidad?

Para permitir la entrada de datos se debe definir:

1. El puerto como bi-direccional, EPP o ECP.
2. Escribir un 1 en el bit 5 del puerto de control

Ejemplo de código:

```
x = inportb(0x37A);          //lee el puerto
y = x | 0x20;                //activa el bit mediante OR lógico
// 1 en la posición 5
// y guardar el resultado en y como 0x20 = 00100000
outportb(0x37A,y);           //envia al puerto de control
```

Ahora puede leerse el puerto (bits 2 a 9) mediante la sentencia:

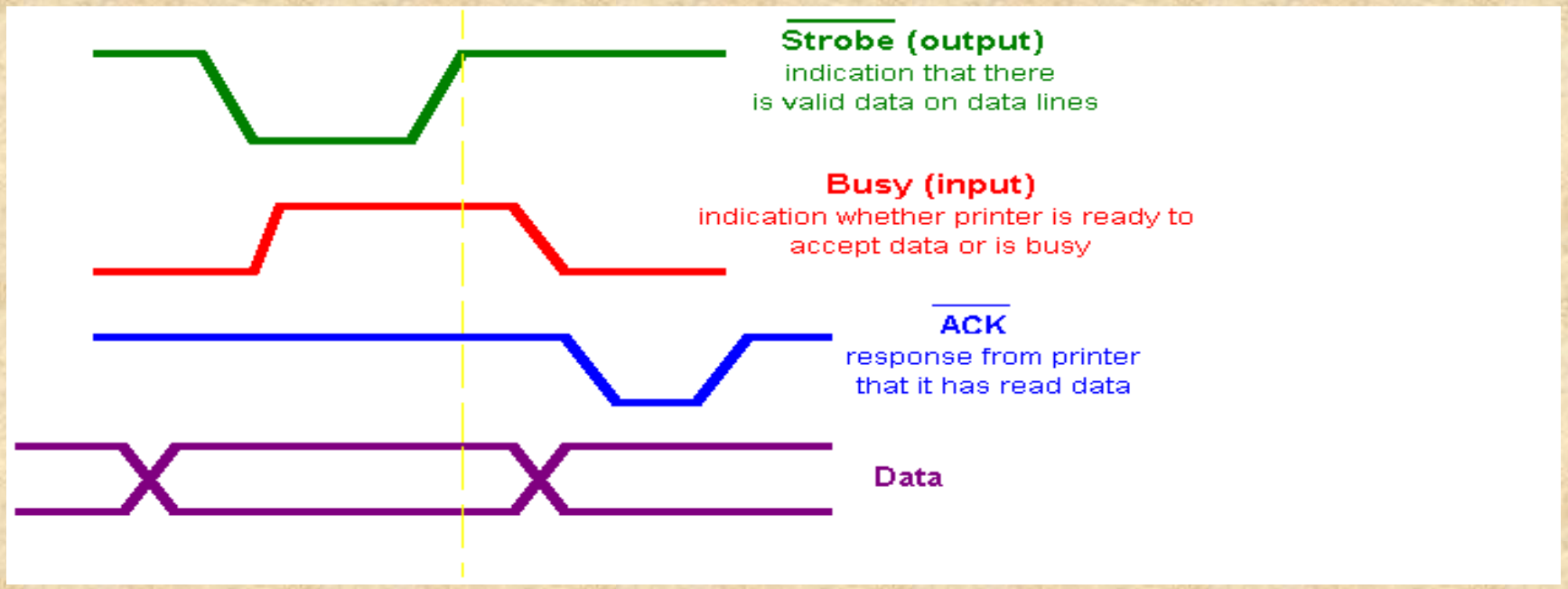
```
z=inportb(0x378);
```

Introducción a EPP y ECP

Antes de entrar en detalles de EPP & ECP veamos por qué se diseñó así

Protocolo utilizado por la impresora (Centronics Standard)

Según este estándard las señales deben ser así:

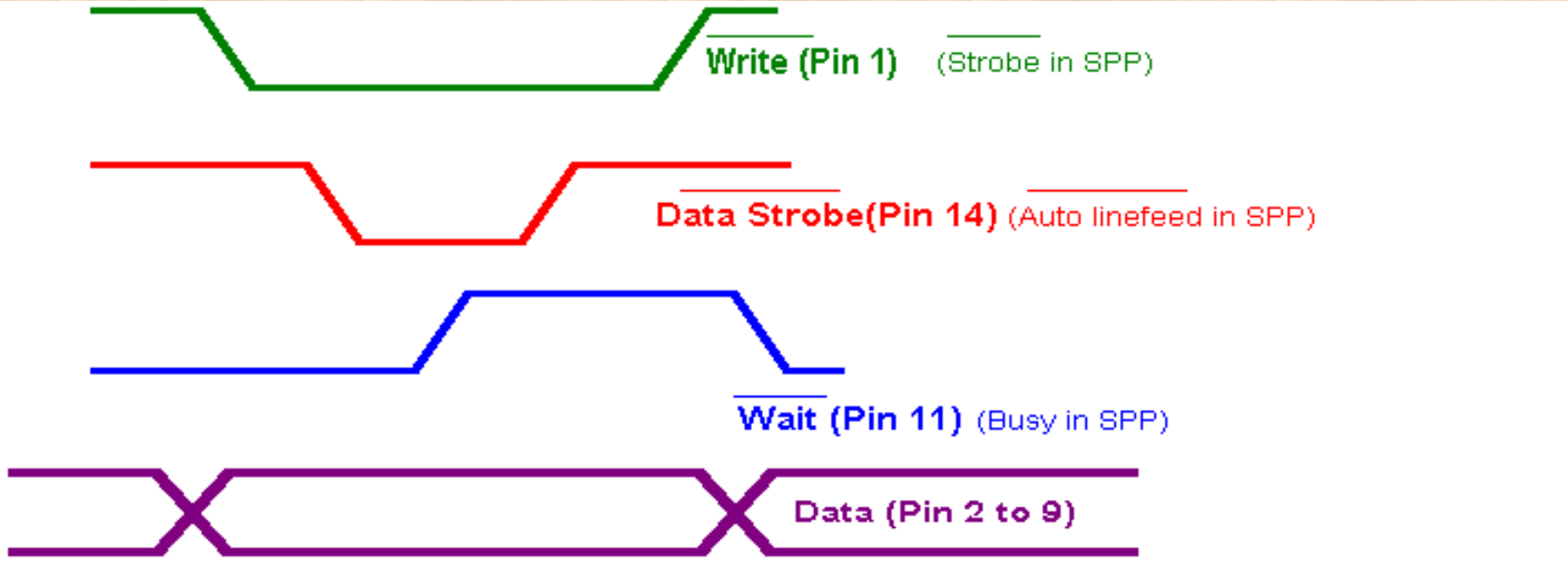


Las señales se pueden generar por programa en el PC, debe esperar la respuesta de la impresora antes de sacar la señal siguiente. Es un proceso lento con velocidades de 50KB/s a 150KB/s.

Introducción al protocolo EPP (Enhanced Parallel Port)_

EPP se diseñó para realizar el protocolo mediante un chip dedicado para liberar al procesador principal. El diagrama de señales es:

EPP Ciclo de escritura:



La única parte a realizar por programa es escribir el byte a enviar en el puerto EPP, en la dirección (Base Address+4) {eg.0x37C}

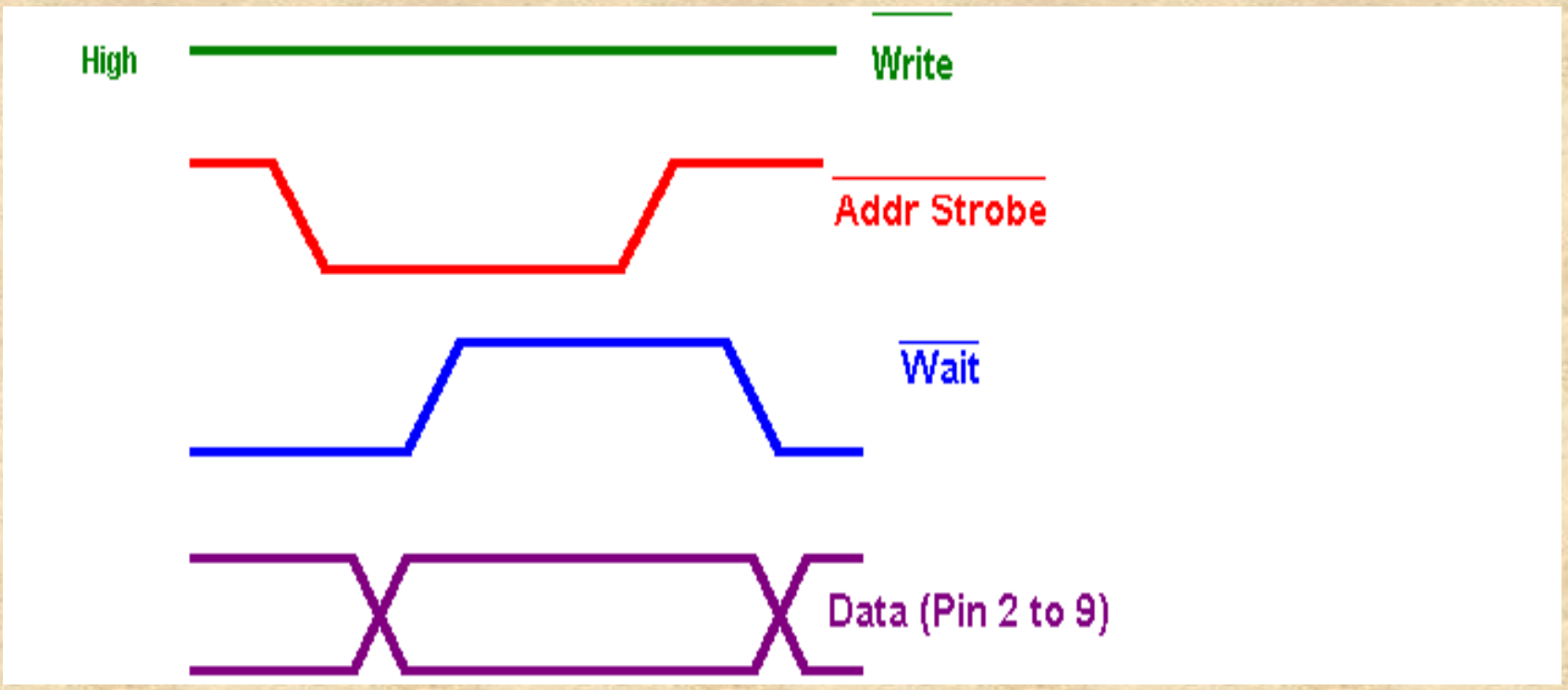
***Nota**: este puerto es diferente del SPP. Sin embargo puede usarse el puerto 0x378 de la forma usual.*

Las señales mostradas se generan automáticamente por el chip.

Descripción del protocolo EPP con chip

1.- El programa escribe el byte a enviar en el registro de datos del EPP (Base +4)
2.- La señal Write se pulsa a nivel bajo por el PC para indicar a la impresora que se está enviando un dato.
3.- Después se escribe el byte de datos (pin 2 a pin 9)
4.- Se lee el pin Wait. Es una entrada desde la impresora. Si está a nivel bajo indica que la impresora está esperando a recibir datos del PC. Después de escribir el byte de datos en el puerto EPP el PC baja la señal DATA STROBE para indicar que la impresora puede leer el dato desde las líneas de datos del puerto.
5.- La impresora lee los datos y pone la línea WAIT en alto durante un periodo breve. Esto indica al PC que la impresora ha leido los datos, con lo que se puede dar por terminado el ciclo de escritura del EPP.
6.- La señal DATA STROBE se pone en alto. La señal WRITE se pone también en alto.
7.- Se termina el ciclo de escritura del puerto EPP.

EPP Ciclo de Lectura de Dirección



Descripción del protocolo usado por el Ciclo de Lectura de Dirección del EPP

1.- La impresora informa al PC que tiene que enviar datos al PC activando la señal de interrupción (pin 10, ACK en SPP). La implementación depende del software. Se puede utilizar cualquiera otra señal para señalización.
2.- Cuando el PC está libre responde a esta señal y lee el registro de dirección del EPP (Base+3).
3.- La señal ADDR STROBE se pulsa a nivel bajo si la señal Wait está baja. Esto se hace automáticamente cuando el procesador lee el Registro de Dirección. Esta señal indica a la impresora que el PC está preparado par iniciar un ciclo de lectura EPP.
4.- El PC espera la señal de reconocimiento desde la impresora hasta que la señal WAIT suba.
5.- Cuando la señal Wait sube se leen los bits de datos (pines 2 a 9)
6.- La señal Address Strobe se sube a nivel alto. Esto acaba el ciclo de lectura EPP.

Indicación de Time-out en EPP

Es posible que el PC inicie un ciclo de escritura pero no reciba la señal de reconocimiento de la impresora, por ejemplo cuando no hay conectada impresora. Para evitar que el PC se cuelgue por esta espera se establece una espera de tipo Watchdog.

1.- Cuando el PC escribe datos al puerto de dirección o datos inicia un temporizador.

2.- El PC espera la subida de la señal de WAIT. Si no ocurre en unos 10μsec, (depende del tipo de puerto), se indica que el tiempo ha sobrepasado poniendo el Bit0 del Registro de Estado (Base+1).

3.- El PC termina su espera e informa al usuario del problema.

Como preparar el puerto para utilizar las características del EPP

Antes de iniciar los ciclos de lectura/escritura de datos o direcciones del EPP, el puerto se debe configurar correctamente.

En reposo, un puerto EPP debería tener las señales Address Strobe, Data Strobe, Wait y Reset en estado alto. Conviene inicializar escribiendo xxxx0100 en el puerto de control (Base+2).

Para saber más detalles puede consultarse el documento. “Interfacing The Enhanced Parallel Port” en “www.beyondlogic.org”.

Incluye ejemplos de programas.

Breve introducción al ECP (Extended Capability Port)

Mejoras del ECP

ECP soporta las características de SPP y EPP. Además incluye las siguientes mejoras:

1.- Puede usar DMA (Direct Memory Access) para transferir los datos sin intervención de la CPU.

2.- Puede usar FIFO (first in first out) para transmitir más de un byte automáticamente antes de necesitar la atención dela CPU.

3.- Soporta compresión de datos utilizando el esquema RLE (Run Length Encoding). Este esquema permite transmitir un sólo byte si ese mismo byte se repite en la secuencia a transmitir.

4.- Soporta un método de direccionamiento del canal mediante el cual se puede comunicar con diferentes módulos del equipo externo. Por ejemplo con los Faxes, copiadora e impresora integradas en una sola máquina.

Para más detalles del ECP y su programación consultar el documento indicado en la transparencia anterior.