PRÀCTICA 7

Treballant amb la USART. Aplicació amb el detector d'obstacles lateral.

Aquesta practica introdueix la comunicació sèrie (emprant USART) entre dos terminals, en el nostre cas el microcontrolador i un ordinador. Per simplicitat, restringirem la comunicació a la transmissió del microcontrolador cap a l'ordinador. Tot i que no es tracti, la recepció del microcontrolador provinent de l'ordinador té un tractament similar.

L'objectiu final és tenir un mitjà de depuració útil en projectes de més envergadura. En aquest sentit, en el darrer apartat de la pràctica s'aprofitarà aquest coneixement de la comunicació per enviar cap a l'hyperterminal l'estat del compàs.

En aquesta pràctica convé fixar-se, sobretot, en:

- *el protocol que s'empra en la comunicació sèrie*
- *la configuració de host i servidor en la comunicació sèrie*
- *l'adaptació de senyals en la comunicació sèrie*
- *i, finalment, en la configuració dels registres per a l'establiment de la comunicació*

Resum del material que cal per a realitzar aquesta pràctica.

- *Plataforma AVRstudio i programador PonyProg*
- *Placa ET-AVR*
- *Placa de prototipatge*
- *Detector lateral d'infraroigs*

7.0 L'ATmega i la comunicació sèrie

Si es vol veure, per exemple, la mesura de l'infrarroig lateral per la pantalla de l'ordinador enlloc de pels leds, cal establir una comunicació entre el microcontrolador i el PC.

La forma més simple d'establir aquesta comunicació és emprant la comunicació sèrie emprant el port sèrie del microcontrolador i el de l'ordinador mitjançant l'*hyperterminal*.

El protocol sèrie és la primera forma de comunicació que es va establir per transmetre dades entre dos terminals. És, per tant, un mètode molt simple. També és per aquesta simplicitat que avui en dia encara es fa servir en múltiples aplicacions. Tot i que a efectes de la pràctica no és del tot indispensable conèixer perfectament el funcionament del protocol sèrie, es recomana llegir la mecànica del funcionament i entendre el connexionat i adaptadors de nivell que calen en el seu ús. En aquest sentit és altament recomanable la lectura de l'apèndix A.

La comunicació entre el microcontrolador i el computador sol emprar un circuit de comunicació *full-duplex* anomenat USART (*Universal Synchronous and Asynchronous Serial Receiver and Transmitter*). Aquí, per a efectes de simplicitat només s'anomena i, tot seguit, expliquem el funcionament de la comunicació de forma força transparent a aquest circuit. Emprarem la comunicació asíncrona del mateix.

Connexionat i adaptació de nivells

La comunicació RS232 (bàsica) s'estableix amb 3 pins: RxD, que significa recepció, TxD que significa transmissió i GND, que és el terra que s'ha de connectar entre els elements que es comuniquen per unificar terres.

La placa ET-AVR (és a dir, el microcontrolador ATmega128) té dos ports per fer la comunicació sèrie. Aquí s'emprarà el port serie 0, conformat pels pins TxD0 i RxD0. El port RxD0, que rep la transmissió del PC cap al microcontrolador, es troba en el pin PortE.0; el port TxD0, que envia les dades cap al PC, es troba en el pin PortE.1.

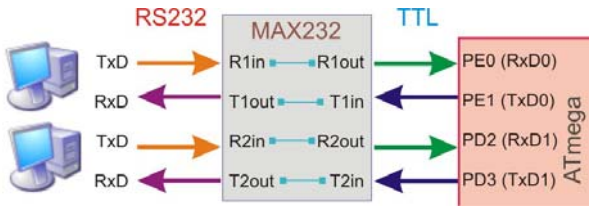

Figura 1

Una segona qüestió a tenir present és l'adaptació de nivells (voltatge) que s'ha de realitzar en la comunicació. Els nivells lògics o i 1 estàndard del microcontrolador s'identifiquen amb els valors 0V i 5V, respectivament. Per a l'ordinador, el 0 i 1 lògic solen entendre's com a valors (mitjos) de 12V i -12V, respectivament. Per tant, quan es realitza una comunicació entre ambdós sistemes cal realitzar una conversió de nivells. La forma més fàcil és emprant un circuit com el MAX232 (implementat per diferents fabricants de circuits integrats).

La figura 2 mostra el connexionat que cal fer en el cas de l'ATmega128. En la pràctica que es realitza, però, només cal connectar els pins PortE.0 i PortE.1 d'acord amb l'esquema de la figura 2.1b. El connector cap al PC connectarà els pins RxD i TxD i establirà un terra comú.

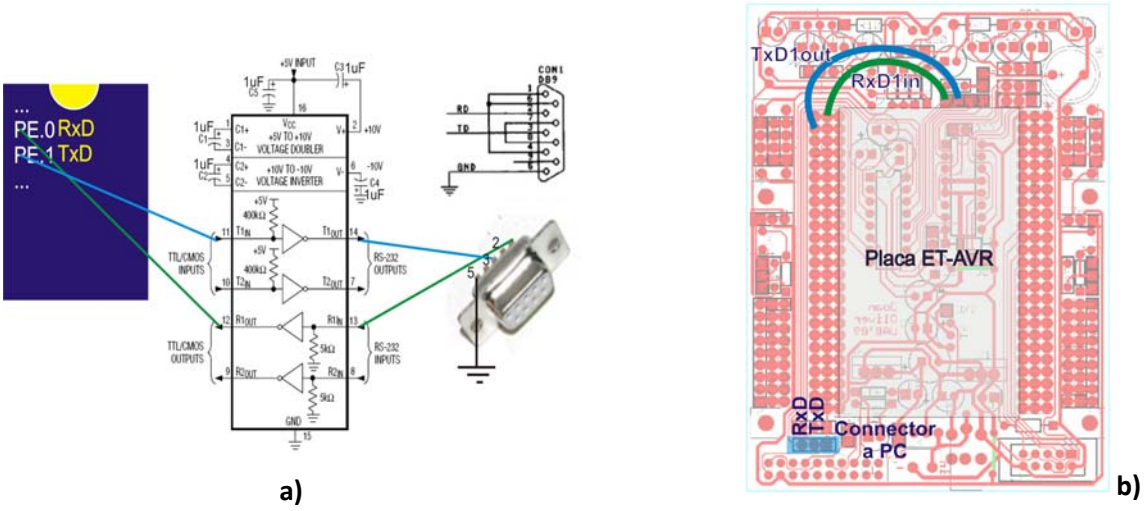

Figura 2. a) Esquema del connexionat que s'estableix amb el circuit MAX232 (interfísie entre el microcontrolador i el PC). b) Connexió a realitzar en la pràctica. Emprant el connector especificat es simplifica el cable de comunicació (no cal el connector DB9).

Configuració de registres

Un cop realitzada la connexió física entre terminals s'ha d'inicialitzar el protocol de comunicació en el microcontrolador. La configuració és una mica complexa respecte al que és indispensable per a iniciar-se en la programació del microcontrolador. Però entendre com s'estableix i com es realitza una comunicació sèrie pot retornar beneficis importants.

Per no donar de bones a primeres un excés d'informació, aquí només es comenten els bits de configuració que són indispensables per entendre una transmissió simple de 8 bits, sense paritat, amb 1 bit de stop i a una velocitat de transmissió de 4800 bauds. Es fa sobre el canal de comunicació 0 (pel 1 és el mateix canviant aquest nombre).

Resum de configuració de comunicació RS232 en el microcontrolador ATmega128

Els diferents registres sobre els que cal actuar són (només s'indiquen els indispensables per a entendre una transmissió simple):

- El UDR0, registre de dades. És un doble registre, un per transmissió i l'altre per recepció. Només s'emprarà en TXB (d'escriptura), que és el de transmissió. Per nosaltres serà transparent (no haurem d'actuar sobre ell).
- El registre A de control i de status, UCSR0A.
  - El bit 6, TXC0 (transmissió completada) indica que el buffer de transmissió és buit.
  - El bit 5, UDRE0 (USART Data Register Empty), indica que el buffer de transmissió pot rebre una nova dada.
- El registre B de control i de status, UCSR0B.
  - El bit 6, TXCIE0 (TX Complete Interrupt Enable), habilita que es generi una interrupció quan s'ha de transmès la dada (quan TXC0 és 1).
  - El bit 5, UDRIE0 (USART Data Register Empty Interrupt Enable), habilita la generació d'interrupció quan el bit UDRE0 és 1.
  - El bit 3, TXEN0 (Transmitter Enable), habilita el transmissor de la USART.
- Setup del nombre de bits que s'emeten en cada transmissió. S'estableix amb els bits
  - UCSZ02, que és el bit 2 del registre UCSR0B.
  - UCSZ01, UCSZ00, que són els bits 2 i 1, respectivament, del registre UCSR0C.
- Establiment del baud rate. Es fa amb el doble registre UBRR0 (UBRR0H, UBRR0L). Cal anar amb compta ja que UBRR0H comparteix espai de memòria amb UCSR0C. El bit7 (URSEL0) de UCSR0C controla quan s'accedeix a UBRR0H o a UCSR0C .

  Així, per posar el baud rate s'empren els bits:
  - UCSZ02, que és el bit 2 del registre UCSR0B.
- La inicialització de la comunicació és la mateixa per la transmissió que per la recepció. El control de la recepció segueix els mateixos passos que per la transmissió.

Per a la pràctica establirem la següent transmissió: 8 bits, sense paritat, amb 1 bit de stop i velocitat de transmissió de 38400 bauds. Això (d'acord amb el datasheet) vol dir que UCSZ0(2..0)=(011) i que UBRR0=207.

7.1 Creació d'un programa simple de transmissió de dades

El següent programa és un exemple de comunicació simple entre el microcontrolador i el computador. Senzillament s'envia una dada contínuament pel port sèrie, tot esperant que el PC la rebi. El programa principal no és res més que un buble. Aquesta simplicitat permet que es pugui comprovar fàcilment el seu funcionament connectant la placa (l'ATmega128) i l'entrada sèrie de l'ordinador mitjançant un cable sèrie.

Donat que només es realitza la transmissió cap al host, la rutina implementada és molt simple. El programa, en sí, té el següent funcionament:

- Es calcula una dada a enviar.
- Es configura el pin PORTE.1 com a sortida. Correspon a la sortida TxD0.
- S'envia la dada cap al port i s'espera que el buffer estigui buit (fent polling sobre el bit UDRE (Usart Data Register Empty)) per enviar una nova dada.
- Si es vol, és fàcil enviar la dada que es transmet cap a un port per ser visualitzada.

```
; Programa 7.1: Primer exemple de transmissió
; Joan Oliver. Gener 2010
; es genera un bucle...
; ...que va enviant a l'hyperterminal la dada del portA.
.INCLUDE "m128def.inc"
.DEF  srri    = R10                     ;registre guarda SREG en interrupcions
.DEF  temp    = R16                     ;registre temp
.DEF  dada    = R17                     ;registre temp per dada de transmissió
.EQU BASE = 0x30                        ;0 en codi ASCII
.CSEG
.ORG 0x00 rjmp    inici                 ;Servei interrupció inici
;********************INICI PROGRAMA*********************************
; Setup de ports d'E/S, UART i interrupcions externes
inici:  ldi     temp,LOW(RAMEND)        ;Setup del stack
        out     SPL,temp
        ldi     temp,HIGH(RAMEND)
        out     SPH,temp
        sbi     DDRE, 1                 ; setup ports -> PE.1 sortida TxD
        rcall   setup_UART              ; setup uart
;-----------------------------------------------------------------
;-------PROGRAMA PRINCIPAL-----------------------------------------
main: ldi     dada, BASE                ; comparar si ha canviat dada
altre:  rcall   transmit                ; si ha canviat nova transmissió
        inc     dada
        cpi     dada, 0x3A
        brlo    altre
        rjmp    main
;-------RUTINA DE SETUP DE LA UART---------------------------------
; a 4800 bauds(UBBR0=207=0x00CF) i frame format: 8data,1stop bit
setup_UART:
        LDI   temp,0x00                 ; baud rate
        sts     UBRR0H,temp
        LDI   temp,0xCF
        out     UBRR0L,temp
        LDI   temp,(1<<TXEN0)           ; enable transmissió
        out     UCSR0B,temp
        LDI   temp,(1<<UCSZ01)|(1<<UCSZ00) ; frame format
        sts     UCSR0C,temp
        ret
;-------RUTINA DE TRANSMISSIO--------------------------------------
;Molt senzillament, aquesta rutina envia un byte cada cop que es crida
;No es pot enviar fins que el registre de trasmissió queda buit
transmit: sbis UCSR0A, UDRE0            ; comprovar si buffer d'enviament buit
        rjmp    transmit                ; ...si no buit, esperar
        out     udr0, dada              ; si no es pot posar dada en buffer per ser enviada
        ret
```

TREBALL PREVI (en negreta el què s'ha d'entregar en l'informe previ)

- Entendre l'algorisme. **Què envia aquest algorisme i què reb l'hyperterminal?**
- **Realitzar el corresponent diagrama de flux (a entregar previ a la pràctica).**
- Simular i comprovar el funcionament. Verificar el comportament dels registres de la USART en el funcionament de l'algorisme

EN EL LABORATORI

Configuració de l'hyperterminal

L'hyperterminal és una aplicació de comunicació simple que podem trobar en el menú de programes del Windows sota "Accessoris". L'hyperterminal s'ha de configurar amb els mateixos paràmetres de comunicació que s'utilitzen en el microcontrolador.

Comprovar com, un cop configurat l'hyperterminal i connectat l'ordinador amb el microcontrolador, s'envien les dades corresponents a l'ordinador.

7.2 El detector d'obstacles en l'hyperterminal

En aquesta part de la pràctica es demana visualitzar la mesura de l'IR lateral en l'hyperterminal. En la pràctica 6 es va controlar un servomotor amb la dada que provenia del detector d'obstacles. Ara es mesurarà la distància i s'enviarà l'hyperterminal. En concret es demana:

- Respecte al detector d'obstacles.
  - Es farà servir el Timer0 en mode *Fast PWM* per generar un pols de duració aproximada d'1 ms amb un període de 61 Hz (posar el prescaler a 1024). S'activaran les interrupcions per overflow i per comparació de fi de pols d'1 ms (semblant a la pràctica 6).
    - En interrupció per overflow, s'activarà l'emissor IR
    - En interrupció per comparació primer s'activarà la conversió ADC i després es deshabilitarà l'emissor IR.
  - Es farà servir el conversor ADC amb habilitació d'interrupció de fi de conversió.
    - Quan es produeixi la interrupció la dada convertida estarà disponible en el registre ADC (recordar treballar amb alineació a l'esquerra per tenir un sol byte) i s'activarà un senyal de DadaNova.
  - En el programa principal
    - En sortir de la interrupció de l'ADC el programa principal detectarà el senyal DadaNova i realitzarà una única transmissió de la dada convertida.
- Respecte a la USART
  - S'inicialitzarà a: 38400 bauds, 8 bits, no paritat, 1 bit de stop.
  - Com s'ha fet en l'apartat 7.1 recordar fer la rutina d'enviament de manera que el programa s'esperi mentre el buffer de sortida de la USART estigui ple.

La figura següent mostra una cronologia dels events esperats en el programa:

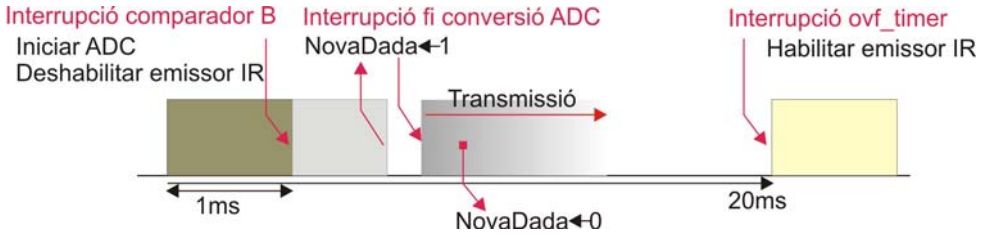

Figura 7.3

A continuació es dóna un esquema del programa que cal completar:

```
; Programa 7.2: Transmetent distància de xoc
; Joan Oliver. Gener 2010
.INCLUDE "m128def.inc"
.DEF  srri          = R0        ;registre guarda SREG en interrupcions
.DEF  temp          = R16       ;registre temporal
.DEF  dada          = R17       ;dada de transmissió
```

```
.CSEG
.ORG 0x00 RJMP  inici                   ;Servei interrupció inici
----POSAR ELS SALTS A RUTINES DE SERVEI D'INTERRUPCIÓ QUE FACIN FALTA----
RETI

;*******************INICI PROGRAMA*********************************
; Setup de ports d'E/S, UART i interrupcions externes
inici:  LDI     temp,LOW(RAMEND)    ;Setup del stack
        OUT     SPL,temp
        LDI     temp,HIGH(RAMEND)
        OUT     SPH,temp
        RCALL   setup_ports         ; setup ports
        RCALL   setup_timer         ; setup timer
        RCALL   setup_ADC           ; setup adc
        RCALL   setup_UART          ; setup uart
        SEI
; El bit T controla si hi ha una nova dada per enviar
main: BRTS      enviar
        RJMP    main
enviar: IN      dada, ADCH          ; Adquirir valor de conversió
        RCALL   transmit            ; Anar a rutina transmissió USART
        CLT
        RJMP    main
;*******************RUTINES*********************************
;------------------------ PORTS-------------------------------------
;Rutina de setup
setup_ports: SBI    DDRB, 0         ;pols emissor IR
        SBI     DDRE, 1             ;sortida transmissió
        LDI     temp, 0
        STS     DDRF, temp          ; PortF.0 entrada de fotodiode (pin 0)
        RET
;-------ADC-------------------------------------------------------------
; Setup ADC: Interrupció per fi conversió, ajust esquerra i prescaler=8=>t_conv=8*13/16000000s
setup_ADC: LDI  temp,(1<<ADEN)|(1<<ADIE)|(1<<ADPS1)|(1<<ADPS0)
        OUT     ADCSRA,temp
        LDI     temp,(1<<REFS0)|(1<<ADLAR)
        OUT     ADMUX,temp
        RET
; Interrupció: Habilita bit per iniciar comunicació
i_ADC: SET                          ;l'activació del bit T indica que s'ha convertit una nova dada
        RETI
;-------TIMER0-----------------------------------------------------------
;Setup: Interrupció per overflow i comparació, comparador a 1 ms, presc=1024, no cal activar sortida
setup_timer:
        ----SETUP DEL TIMER0----
        RET
;Interrupció per ovf: Habilitar emissor IR
i_tim0Ovf:
        ----RUTINA INT PER OVERFLOW DEL TIMER0----  ;→Activació emissor IR
        RETI
;Interrupció per comparació: Nova conversió ADC i deshabilitar emissor IR
i_tim0Cmp:
        ----RUTINA INT PER COMPARACIÓ DEL TIMER0----;→ Iniciar nova conversió i desactivar emissor I
        RETI
;------- UART-------------------------------------------------------
```

```
;Setup: a 2400 bauds i frame format: 8data,1stop bit
setup_UART:
        ----SETUP DE LA UART----
        RET
;Rutina transmissió: envia un byte cada cop que es crida i el buffer està buit
transmit:
        ----RUTINA DE TRANSMISSIÓ DE LA UART----
        RET
```

EXERCICI

TREBALL PREVI (en negreta el que s'ha d'entregar en l'informe previ)

- **Realitzar un esquema de blocs del funcionament del programa.**
- Simular i comprovar el correcte funcionament.

EN EL LABORATORI

- Programar la placa i verificar el seu comportament.
- Feu el diagrama de temps exacte de la temportització dels diferents senyals. Podeu agafar com a 0 d'escala la interrupció del timer0 per overflow. Justifiqueu numèricament la temportització dels senyals.

Arribats a aquest punt l'hyperterminal mostra per pantalla un conjunt de caràcters, alguns d'ells rars. El motiu és que s'envien bytes cap a l'hyperterminal mentre que aquest espera rebre caràcters ASCII. Per solucionar aquest problema una solució és convertir el byte (3 dígits decimals) als corresponents codis ASCII i enviar aquests.

EXERCICI PYTHON (opcional)

- En Python implementar el programa que converteixi a enter el byte rebut pel port de comunicació sèrie.

7.3 Controlant el servomotor des de l'hyperterminal

Finalment en aquest apartat es tractarà la recepció d'informació per la USART. Es farà controlant el servomotor des de l'hypeterminal.

El funcionament és molt simple. Amb les tecles '+' i '–' del teclat es donarà velocitat o es treurà velocitat als servomotors. Per fer-ho es faran servir els següents recursos:

- Generació del pols de control del servomotor. Es farà servir el timer1 configurat en la forma de Fast PWM, amb TOP a 20ms i interrupcions per TOP i per comparador B (com en la pràctica 6). Donat que no cal controlar res més amb el timer es pot aprofitar la sortida OC1B per enviar el pols cap al servomotor.
- Recepció del control de velocitat: es farà servir l'entrada RxD de la USART configurada en mode de recepció per interrupció, 2400 bauds, 8 bits,sense paritat i un bit de stop.
- Control de velocitat: es faran servir les tecles '+' i '-'.
    - En el setup del timer s'inicialitzarà a un ampla de pols d'1.5ms
    - Quan s'apreti '+' s'incrementarà el pols, fins a un màxim de 2ms
    - Quan s'apreti '-' es decrementarà el pols fins a un mínim d'1ms

EXERCICI

TREBALL PREVI (en negreta el que s'ha d'entregar en l'informe previ)

- **Realitzar un esquema de blocs del funcionament del programa.**
- **Fer el programa de control del servomotor per recepció d'hyperterminal.**

- Simular i comprovar el correcte funcionament.

EN EL LABORATORI

- Programar la placa i verificar el seu comportament.
- Feu el diagrama de temps exacte de la temportització dels diferents senyals. Podeu agafar com a 0 d'escala la interrupció del timer0 per overflow. Justifiqueu numèricament la temportització dels senyals.

Millores que es podrien introduir

- Controlar que el pols del motor mai sigui més petit d'1 ms o més gran de 2 ms.
- Aprofitar el setup de la USART per enviar un eco del motor cap al PC indicant l'ampla de pols.

APÈNDIX A

La comunicación serie RS232

Para una descripción más formal del protocolo de comunicación serie se recomienda consultar el estándar EIA RS232-c

El protocolo de comunicación serie full-dúplex RS232 asíncrona establece una comunicación fácil entre computador y periférico.

El protocolo de comunicación RS232 establece el formato de la figura XX de transmisión de datos. Está formado por:

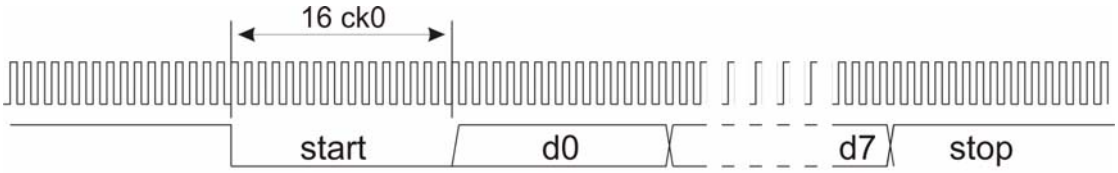

Figura 1

- *El estado de reposo implica un 1 lógico.*
- *Un bit de start a 0 lógico.*
- *7 u 8 bits de datos.*
- *Posibilidad de bit de paridad.*
- *Uno, uno y medio o dos bits de stop, a 1 lógico.*

La velocidad de comunicación se mide en baudios, que es el número de bits que se envían por segundo. Velocidades típicas de transmisión, en baudios, son 300, 600, 1200, 2400, 4800, 9600, 19200, 57600 y 115200.

Durante la transmisión (recepción) la duración de cada bit es de 16 ciclos de reloj base. Por consiguiente, en la realización de los drivers (transmisor y receptor) se tendrá que contemplar la sincronización de la transmisión / recepción de datos con el reloj base.

Aunque computador y periférico se programen para transmitir datos a una frecuencia determinada, es difícil asegurar que las frecuencias base de los relojes estén perfectamente sincronizadas. Por ello la emisión y recepción de datos se realiza tomando como base una frecuencia 16 veces superior a la frecuencia de transmisión de datos. Por otra parte, durante la recepción de datos pueden producirse en la transmisión glitches que, debido a interferencias, pueden introducir errores de recepción. Para minimizar este efecto suelen tomarse distintas muestras de la recepción del bit, tomando como resultado bueno, el valor que más se sucede dentro de la recepción del bit. Es por ello, que durante la recepción se deberá considerar circuitería adicional para asegurar un comportamiento más fiable de la transmisión.

En nuestro caso se va a utilizar un formato simple de comunicación asíncrona: se elige una comunicación full-dúplex basada en un bit de start, 8 bits de datos sin paridad, y un bit de stop. Aunque la implementación del protocolo en todas sus opciones tan sólo exige un registro de control de operación adicional, para simplicidad, se ha optado por fijar las características de transmisión.

En una comunicación simple mediante protocolo RS232 (sin control de recepción) tan sólo son necesarias tres líneas de conexión entre el PC y el periférico: RxD o recepción, TxD o transmisión, y tierra.

Puerto serie.

Aunque el puerto serie puede tener un tamaño de 25 pines, el formato más utilizado es el de 9 pines. La tabla del pinout para ambos conectores es:

Señal		Función	Pinout 9 pines	Pinout 25 pines
Transmit Data	*TxD*	*Salida serie*	*3*	*2*
Receive Data	*RxD*	*Entrada serie*	*2*	*3*
Request to Send	*RTS*	*Informa al módem que la UART está preparada para intercambiar datos*	*7*	*4*
Clear to Send	*CTS*	*Módem preparado par intercambiar datos*	*8*	*5*
Data Set Ready	*DSR*	*Inidica a la UART que el módem está preparado para establecer conexión*	*6*	*6*
Signal Ground	*GND*		*5*	*7*
Carrier Detection	*CD*	*Se activa cuando el módem recibe una portadora del otro módem*	*1*	*8*
Data Terminal Ready	*DTR*	*Informa al módem que la UART está preparada para establecer conexión*	*4*	*20*
Ring Indicator	*RI*	*Se activa cuando el módem detecta una señal indicadora de sonido.*	*9*	*22*

Tabla 1

En sistemas de prototipado con el PC o en comunicaciones serie dedicadas con microcontroladores en los que los errores de comunicación son prácticamente nulos existen dos conexionados típicos simples que eliminan el protocolo de conexión: el null modem y la conexión loopback.

Null modem

Se establece el conexionado de la figura YYY, en el que sólo se requieren tres conexiones: TxD, RxD y GND. Las conexiones dan a entender al computador que está conectado con un módem. La transmisión del PC se recibe en la UART por la señal de entrada. Y viceversa, la transmisión de la UART se envía al bit de recepción del PC.

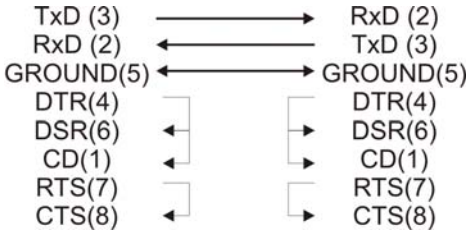

Figura 2. Comunicación RS232 en modo Null Modem.

La conexión Null modem es la que se utiliza en la práctica. Es una conexión simple que sólo requiere las tres conexiones mostradas en la figura.

Conexión loopback

La conexión loopback es útil cuando se realiza un programa de debugging en el PC. La señal de datos de salida del PC se conecta a la de entrada. De esta forma, toda señal enviada por el PC es recibida en la línea de entrada como si fuera un eco del módem.

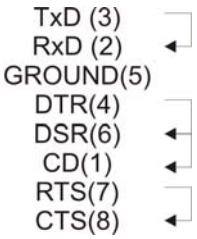

Figura 3. Comunicación RS232 en modo LoopBack.