UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE
INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

MICROCONTROLADORES

USART ASINCRÓNICO

Andrés Acurio – Fabricio Borja
andyacurys@hotmail.com
fabricio.b.reinoso@hotmail.com

RESUMEN: *UART son las siglas de "Universal Asynchronous Receiver-Transmitter". Éste controla los puertos y dispositivos serie. Se encuentra integrado en la placa base o en la tarjeta adaptadora del dispositivo. Un UART dual, o DUART, combina dos UARTs en un solo chip. Existe un dispositivo electrónico encargado de generar la UART en cada puerto serie. Las funciones principales de chip UART son de manejar las interrupciones de los dispositivos conectados al puerto serie y de convertir los datos en formato paralelo, transmitidos al bus de sistema, a datos en formato serie, para que puedan ser transmitidos a través de los puertos y viceversa.*

ABSTRACT: *UART stands for "Universal Asynchronous Receiver-Transmitter". This controls the ports and serial devices .It is integrated into the motherboard or adapter card in the device. A dual UART, or DUART, combines two UART s into a single chip. There is an electronic device which generates the serial port UART in each. The main functions are to handle chip UART interrupts of the devices connected to the serial port and convert the data in parallel format, transmitted to the system bus, a serial data format, so they can be transmitted through ports and vice versa.*

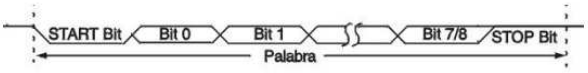
1 MARCO TEORICO

El USART (universal synchronous asynchronous receiver transmitter) es uno de los dos puertos series de los que dispone los PIC16F87X.

Puede funcionar de forma síncrona (half duplex) o asíncrona (full duplex).

Modo asíncrono:

1. Modo full-duplex (bidireccional).
2. Utiliza los pines:
• RC6/TX/CK: transmisión (salida).
• RC7/RX/CK: recepción (entrada).
3. Los datos enviados tienen tamaño de byte.
4. En el formato de la trama se añade un bit de Start=0 y un bit de Stop=1, y puede añadirse un noveno bit de datos (ejemplo bit de paridad) a los 8 bits del dato:

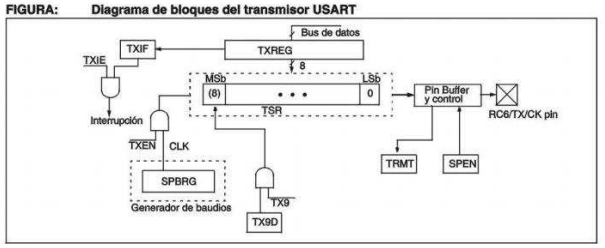


5. Esta forma de comunicar serie usa la norma RS-232 / RS-485.
6. Los bits se transmiten a una frecuencia fija y normalizada.
7. Los bloques que configuran la USART en modo asíncrono son:
• Circuito de muestreo.
• Generador de baudios.
• Transmisor asíncrono.
• Receptor asíncrono.
8. La USART no soporta la generación de paridad por hardware.
9. En modo asíncrono la USART se para al entrar el micro en modo SLEEP.

Transmisor Asincronico

_La transmisión se habilita mediante el bit TXEN, TXSTA El registro de transmisión es el TXREG. Para transmitir un dato el software lo escribe en este registro.
_Después de haber escrito el TXREG el dato pasa al registro de desplazamiento TSR, este registro no se carga hasta que el bit de STOP del dato anterior no se ha transmitido.
_Al quedar vacío el TXREG se activa el bit de interrupción TXIF (PIR1), habilitado por el bit TXIE (PIE). (TXIF no se desactiva por software, se desactiva sólo cuando se cargan nuevos datos).
_Hay otro bit el TRMT, TXSTA que muestra el estado del TSR, no produce ninguna interrupción. (Cuando activa TRMT está vacío).
_ Para enviar un dato con 9 bits hay habilitar el bit TX9, (TXSTA) y poner el que se quiere enviar en TX9D (TXSTA).

FIGURA: Diagrama de bloques del transmisor USART



R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7 **CSRC:** Selección de la fuente de la señal de reloj
Modo asíncrono:
No importa
Modo síncrono:
1 = Modo maestro (reloj generado internamente por BRG)
0 = Modo esclavo (reloj de fuente externa)

bit 6 **TX9**: Habilitación transmisión del 9-bit
1 = Selecciona transmisión 9-bit
0 = Selecciona transmisión 8-bit

bit 5 **TXEN**: Habilita transmsión
1 = Transmit enabled
0 = Transmit disabled

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

bit 4 **SYNC**: Selección del modo USART.
1 = Modo síncrono
0 = Modo asíncrono

bit 3 **No implementado:** Se lee como '0'

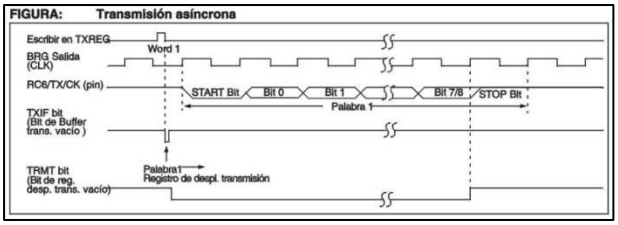
bit 2 **BRGH**: Selección de velocidad.
Modo asíncrono:
1 = Velocidad alta
0 = Velocidad baja
Modo síncrono:
No se utiliza en este modo.

bit 1 **TRMT**: Bit de estado del registro de desplazamiento de transmisión.
1 = TSR vacío
0 = TSR lleno

bit 0 **TX9D:** 9th bit de transmisión, puede ser de paridad

Pasos a seguir para implementar la transmisión:
1. Configurar RC6/TX/CK como salida y RC7/RX/DT como entrada.
2. Poner SYNC=0 y SPEN=1, USART en modo asíncrono
3. SI se desea activar interrupciones activar TXIE=1.
4. Si el dato es de 9 bits TX9=1 y cargar TX9D
5. Cargar X en SPBRG, y elegir BRGH para controlar la frecuencia de trabajo.
6. Activar la transmisión TXEN=1,
7. Cargar en TXREG el dato a transmitir.

FIGURA: Transmisión asíncrona



Receptor Asincrónico

La recepción se habilita mediante el bit CREN, (RCTA).

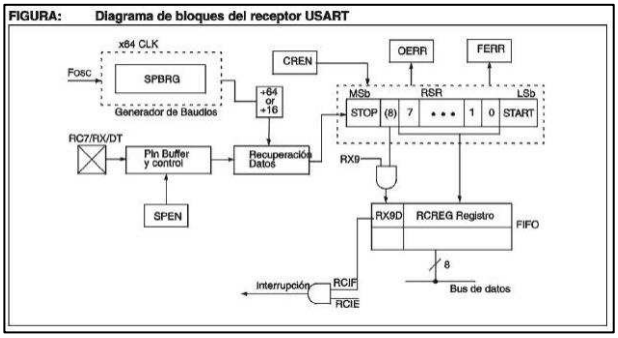
_Los datos entran por el pin RC7/RX/DT, llegan hasta el muestreador y se cargan en el registro de desplazamiento RSR de forma serie.

_Al recibir el bit de STOP, el dato contenido en RSR pasa al registro RCREG si está vacío, y se activa el bit de interrupción RCIF, (PIR1). Habilitada mediante el bit RCIE (PIE1). (RCIF es de sólo lectura y se desactiva por hardware al leer RCREG).

_El registro RCREG admite dos datos a la espera de ser leídos. Formando un FIFO de dos niveles. Si se reciben tres datos sin que RCREG se lea, el último se pierde. Se produce un error de sobreescritura y hay que reiniciar el receptor. El bit de sobre escritura OERR (RCSTA), se desactiva reseteando el receptor. (CREN=0).

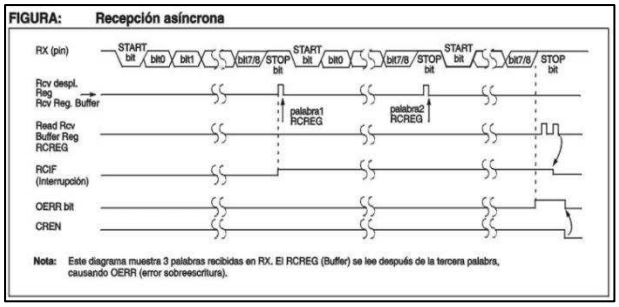
_El error de encuadre FERR, (RCSTA) se produce si el bit de STOP es un cero. El 9th bit y FERR se cargan a la vez que RCREG, al leer el último dato de RCREG por lo tanto siempre hay que leer el 9th bit y FERR antes de leer RCREG.

FIGURA: Diagrama de bloques del receptor USART



Pasos a seguir para programar la recepción:

1. Configurar RC6/TX/CK como salida y RC7/RX/DT como entrada.
2. Cargar X en SPBRG, y elegir BRGH para controlar la frecuencia de trabajo.
3. Poner SYNC=0 y SPEN=1, USART en modo asíncrono
4. SI se desea activar interrupciones activar RCIE=1.
5. Si el dato es de 9 bits RX9=1.
6. Habilitar la recepción con CREN=1.
7. Al completarse la recepción RCIF=1 y produce interrupción si se ha habilitado.
8. Se lee el registro RCSTA y se averigua si se ha producido algún error.
9. Leer el dato de RDREG.



R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 **SPEN:** Habilita el puerto serie
1 = Habilita el puerto serie
0 = Deshabilitado

bit 6 **RX9**: Habilita la recepción del 9-bit
1 = Selecciona recepción con 9-bit
0 = Selecciona recepción con 8-bit

bit 5 **SREN**: Habilita la recepción sencilla
Modo asíncrono:
No influye
Modo síncrono-maestro:
1 = Habilita la recepción sencilla
0 = Deshabilita la recepción sencilla
Este bit se desactiva después de la recepción
Modo síncrono-esclavo:
No influye

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 4 **CREN**: Habilita la recepción continua
Modo asíncrono:
1 = Habilita la recepción continua
0 = Deshabilita la recepción continua
Modo síncrono:
1 = Habilita la recepción continua.
0 = Deshabilita la recepción continua.

bit 3 **ADDEN:** Habilita la detección de la dirección
Modo asíncrono con 9-bit (RX9 = 1):
1 = Habilita la detección de la dirección, sólo recibe el dato y produce interrup. de recepción cuando RSR<8> está activo.
0 = Deshabilita la detección de la dirección, se reciben todos los bytes, y 9th bit puede usarse para paridad

bit 2 **FERR**: Error de encuadre
1 = Error de encuadre (puede actualizarse leyendo RCREG y recibiendo el próximo byte válido)
0 = No hay error de encuadre

bit 1 **OERR**: Error de sobreescritura
1 = Error de sobreescritura (puede ser borrado escribiendo un cero en CREN)
0 = No hay Error de sobreescritura

bit 0 **RX9D:** 9th bit del dato recibido (la paridad debe ser calculada por el software de usuario)

Comunicación Serial RS232

El microcontrolador PIC 16F877A dispone de varios módulos de comunicación serie independientes, además cada uno se puede configurar a funcionar en modos diferentes. El USART es uno de los primeros sistemas de comunicación serie. Las versiones nuevas de este sistema están actualizadas y se les denomina un poco diferente - EUSART.

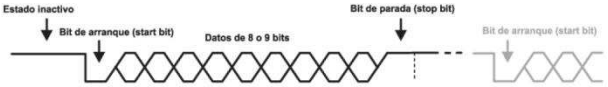
El módulo Transmisor/Receptor Universal Síncrono/Asíncrono USART es un periférico de comunicación serie de entrada/salida. Contiene todos los generadores de señales de reloj, registros de desplazamiento y búfers de datos necesarios para realizar transmisión de datos serie de entrada/salida.

El USART integrado en el PIC16F877A posee las siguientes características:

- Transmisión y recepción asíncrona en modo Full-duplex;
- Caracteres de anchura de 8 – 9 bits programables;
- Detección de dirección en modo de 9 bits;
- Detección de errores por saturación del búfer de entrada; y
- Comunicación Half Duplex en modo síncrono.

EUSART EN MODO ASÍNCRONO

El USART transmite y recibe los datos utilizando la codificación de no retorno a cero - NRZ (non-returnto-zero). Como se muestra en la siguiente figura, no se utiliza una señal de reloj y los datos se transmiten de forma muy simple.

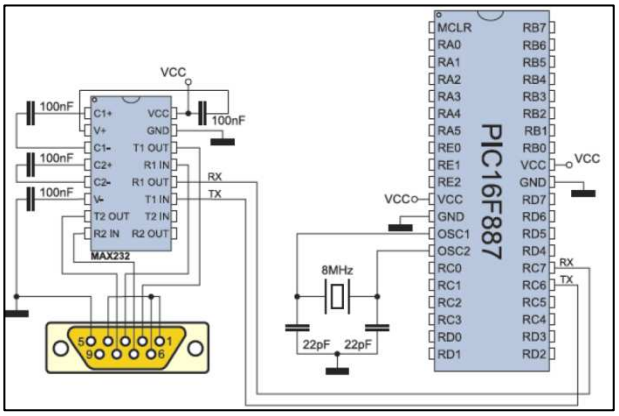


Cada dato se transmite de la siguiente forma:

• En estado inactivo la línea de datos permanece en estado alto (1);
• Cada transmisión de datos comienza con un bit de arranque (START), el cual, siempre es cero (0);
• Cada dato tiene un ancho de 8 o 9 bits (primero se transmite el bit menos significativo- LSB); y
• Cada transmisión de datos termina con un bit de parada (STOP), el cual, siempre es uno (1).

La siguiente figura muestra un ejemplo de cómo se conecta de manera habitual un microcontrolador PIC que utiliza el módulo USART.

El circuito RS-232 se utiliza como un convertidor de nivel de voltaje, para adaptar los niveles de voltaje de la PC y el microcontrolador al protocolo RS232.



Librería UART

MikroBasic posee la librería **"UART Library"** que nos permite de manera simple utilizar el USART del PIC16F877A, para utilizar esta librería lo primero que se tiene que hacer es configurar el USART de la siguiente manera:

UART1_Init(baud_rate**)** ' baud rate = velocidad de transmisión Ej. 9600 bps

Esta instrucción Configura e inicializa el modulo UART de la siguiente manera:

- Recepción habilitada
- Transmisión habilitada
- Trama de datos de 8 bits
- 1 bit de parade "STOP"
- Paridad de datos deshabilitado

 Operación asíncrona.
UART1_Data_Ready() ‘ Esta instrucción testea si el dato recibido en el buffer está listo para ser leído retorna 1 si el dato está listo y 0 si no existe dato.

Una vez que el dato esté listo para ser leído se utiliza:

Rx = **UART1_Read()** ‘ Este instrucción retorna el valor listo y almacenado en el buffer, ojo, primero debe utilizarse UART1_Data_Ready para saber si el buffer esta lleno y listo para leer.

UART1_Write(dato) ‘ Esta función transmite un **dato**

4. CONCLUSIONES

- El controlador del UART es el componente clave del subsistema de comunicaciones series de una computadora.
- Cada UART contiene un registro de desplazamiento que es el método fundamental de conversión entre las forma serie y paralelo.

5. BIBLIOGRAFÍA

- http://informatica.uv.es/~rmtnez/sbm/TEMA25-b&w.pdf
- http://www.aquihayapuntes.com/indice-practicas-pic-en-c/comuicacion-serie-asincrona-entre-dos-pics-con-la-usart.html
- http://es.scribd.com/doc/174943675/Modulo-Usart

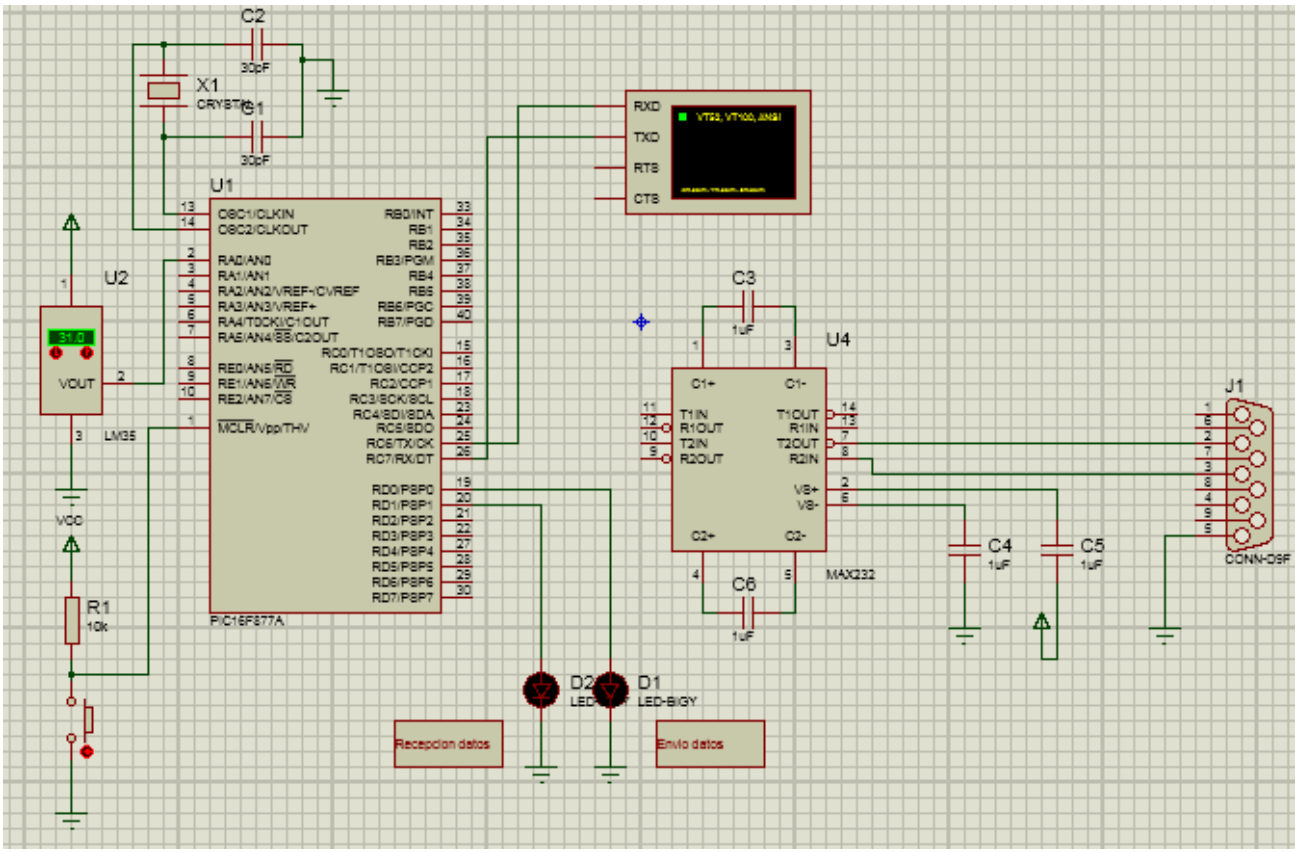
ANEXOS

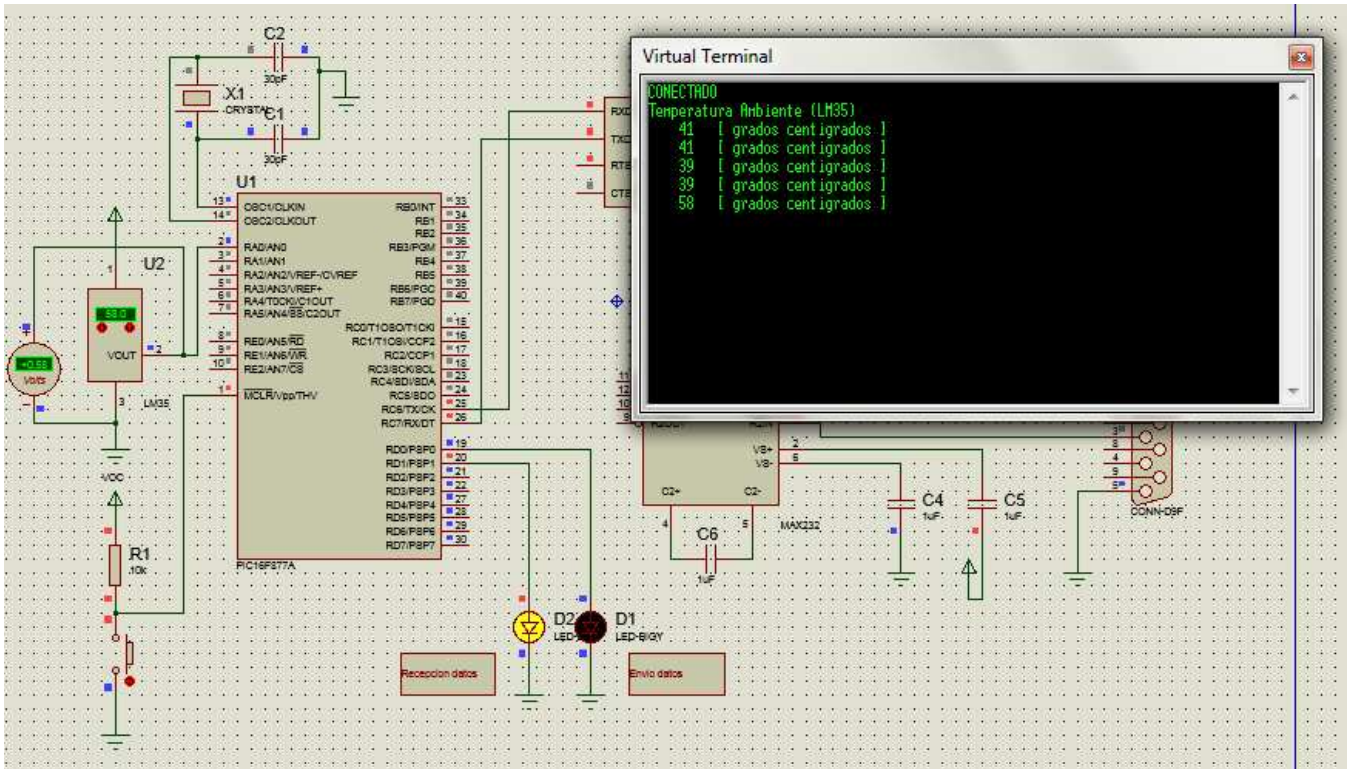
SENSOR DE TEMPERATURA Y EL PIC16F877 CON COMUNICACIÓN USART ASINCRONICO CON LA PC.

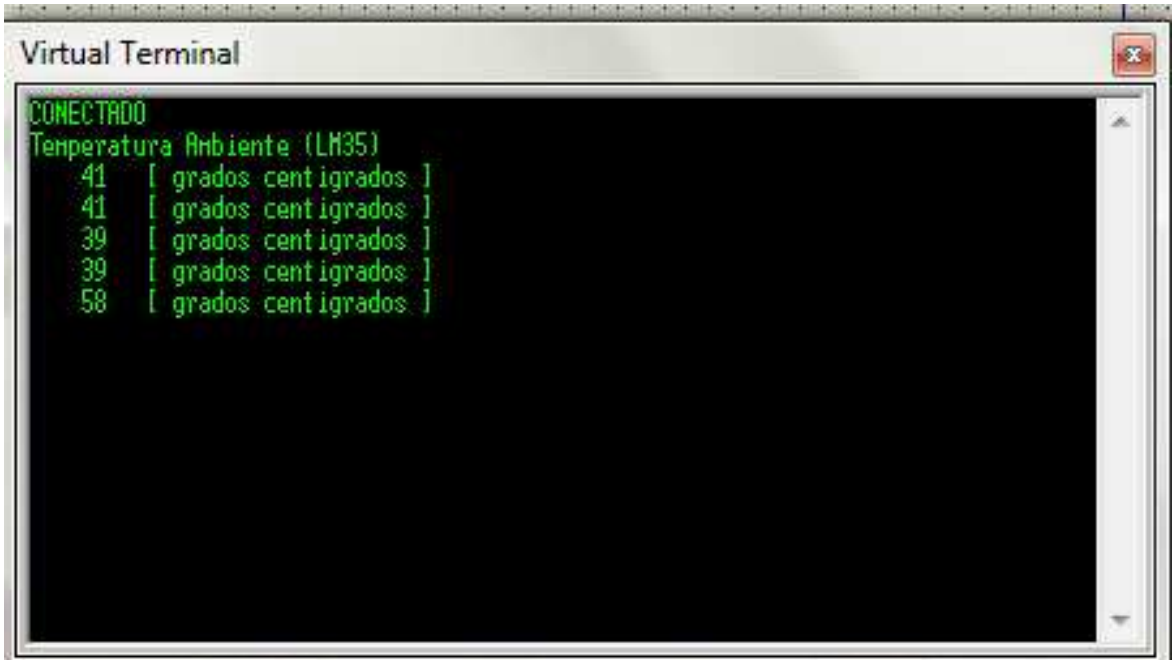
Código MikroBasic:

```
Program usart
' Definicion de variables globales
dim uart_rd as byte  'Guarda la letra enviada de PC en  (uart_rd)
dim temp as word      'Se guarda en (tem) el valor A/D leido del sensor
dim txt as string[5]

'Prgrama principal
main:
TRISD=0      'Salida para prender led
PORTD=0      'Apago el led estado inicial
ADC_init()   'Iniciaizo el ADC
UART1_Init(9600) 'Inicializo modulo UART con velocidad de trasmision de 9600bps
Delay_ms(300)    'Espera para que el UART se estabilize
UART1_WRITE(10)              'Nueva Linea
UART1_WRITE_Text("CONECTADO")  'Envia mensaje a la PC
UART1_WRITE(10)              'Nueva Linea
UART1_WRITE(13)              'Enter

'Bucle infinito
while (TRUE)
'Si existe un dato para leer entra al if (retorna 1 si el dato esta listo y 0 si no existe dato.)
     if(UART1_Data_Ready()<>0)then
          PORTD.1=1          '|
          DELAY_MS(500)        '|Led indicador de recepción de datos al PIC
          PORTD.1=0          '|
     'Lee el dato recibido de la PC y lo guardaen (uart_rd)
     uart_rd=UART1_Read()
       select case uart_rd
        'Si se preciona tecla("t") lee temperatura sensor.
        case "t"
          temp= ADC_Read(0)  'Lee del PORTA.0 la temperatura.
          temp=temp/2
          wordtostr(temp,txt)'Transforma el Word en String para enviarlo a la PC
          UART1_Write_Text(txt)'Esta funcion transmite un dato del PIC hacia la PC
          PORTD.0=1          '|
          DELAY_MS(500)        '|Led indicador de envio de datos a la PC
          PORTD.0=0          '|
       end select
     end if

wend
end.
```

Simulación:


C2
30pF
X1
CRYSTAL
C1
30pF
U1
PIC16F877A
U2
LM35
VOUT
VCC
R1
10k
RXD
TXD
RTS
CTS
C3
1uF
U4
MAX232
C4
1uF
C5
1uF
C6
1uF
J1
CONN-D9F
D2
LED
D1
LED-BIGY
Recepcion datos
Envio datos


Virtual Terminal
CONECTADO
Temperatura Ambiente (LM35)
41 [grados centigrados]
41 [grados centigrados]
39 [grados centigrados]
39 [grados centigrados]
58 [grados centigrados]
C2
30pF
X1
CRYSTAL
C1
30pF
U1
PIC16F877A
U2
LM35
VCC
R1
10k
C3
1uF
U4
MAX232
C4
1uF
C5
1uF
C6
1uF
CONN-D9F
D2
LED
D1
LED-BIGY
Recepcion datos
Envio datos


Virtual Terminal
CONECTADO
Temperatura Ambiente (LM35)
41 [grados centigrados]
41 [grados centigrados]
39 [grados centigrados]
39 [grados centigrados]
58 [grados centigrados]