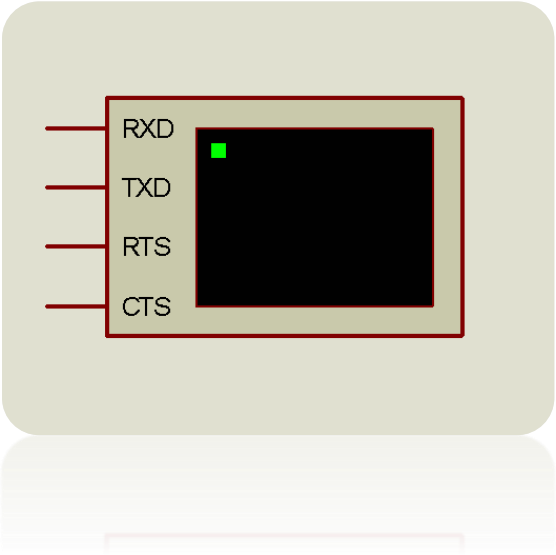
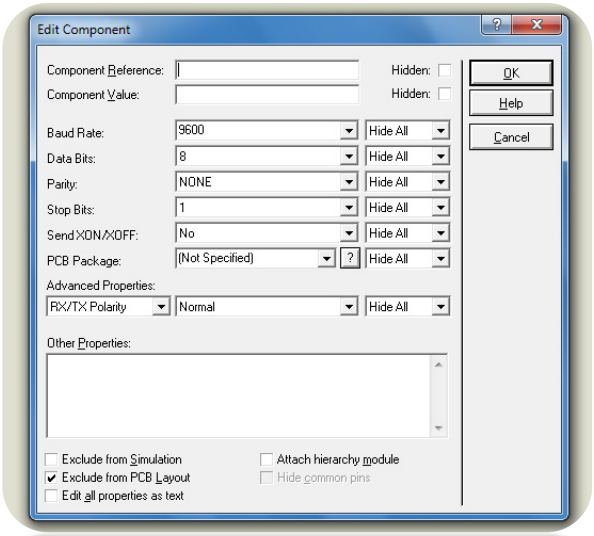
Terminal Virtual en Proteus.

Una terminal virtual sirve para transmitir o recibir datos de forma serial y puede usarse para verificar las transmisiones seriales en nuestros circuitos, ya sea recibiendo datos de la terminal virtual o enviando datos hacia ella para verificar que los reciba. En la lista de instrumentos virtuales aparece como VIRTUAL TERMINAL y usa el protocolo RS232 para enviar o recibir datos.



Las terminales de este instrumento son: RXD para recibir datos; TXD para enviar datos en formato ASCII desde el teclado de la PC, es decir, la terminal virtual enviará los datos ASCII que ingresemos hacia donde conectemos esta terminal; RTS (Ready to send) y CTS (Clear to send). En las propiedades de la terminal virtual podemos configurar los parámetros de la transmisión, incluyendo su velocidad.

Entre las configuraciones podemos definir:



- Baud Rate: este campo contiene la velocidad en baudios de la transmisión serial, puede ir de 110 a 57600 baudios.
- Data Bits (bits de datos): para indicar cuántos bits por dato se enviarán, las opciones son 7 u 8.
- Parity (paridad): aquí se define el bit de paridad; las opciones son NONE (ninguno), EVEN (par) u ODD (impar).
- Stop Bits (bits de detención): permite elegir los bits para la detención.
- Send XON/XOFF: en este campo debemos definir si se enviarán los comandos XON y XOFF o no.

La forma más fácil de ejemplificar el uso de las terminales virtuales es generando una comunicación entre dos de ellas, para lo cual, simplemente, conectamos la terminal TXD de una a la terminal RXD de la otra. Esto podemos verlo en el archivo TerminalVirtual.dsn. Al iniciar la simulación, se mostrarán dos ventanas llamadas

Virtual Terminal –TRANSMISOR y Virtual Terminal – RECEPTOR. Si hacemos un clic en la ventana del transmisor para resaltarla, podremos escribir un mensaje mediante el teclado de nuestra computadora. Los datos se transmitirán hacia el receptor y se mostrarán en su ventana. De esta manera hemos realizado una comunicación serial entre las dos terminales virtuales.

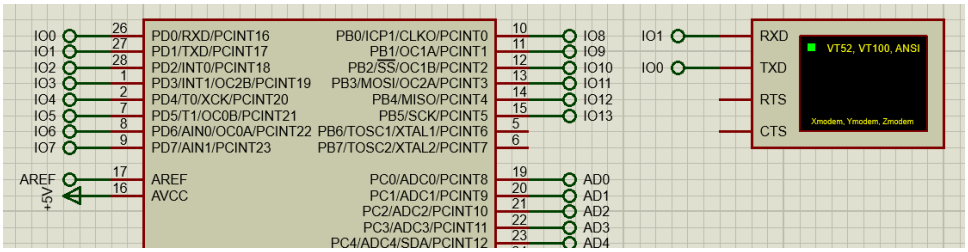
Las dos terminales deben estar configuradas de forma idéntica para que la transmisión se lleve a cabo sin fallas; el nombre que dimos a las terminales nos permite identificarlas en la simulación. Si hacemos un clic con el botón derecho sobre la ventana de una terminal, se abrirá un menú contextual que contiene algunas opciones de configuración. Encontramos:

- Borrar la pantalla (Clear Screen)
- pausar la transmisión (Pause)
- copiar o pegar (Copy/Paste)
- hacer que se reflejen en la pantalla los caracteres que escribimos (Echo Typed Characters)
- cambiar a modo hexadecimal (Hex Display Mode)
- modificar la fuente que se mostrará en la ventana de la terminal (Set Font).

¿Cómo conectar la placa de desarrollo de la simulación (Arduino UNO en esta ocasión) a la terminal virtual?

Para ello debemos identificar los pines de la placa de desarrollo que permiten el manejo de la conexión serial. En el caso del Arduino UNO los pines son los pines 0 y 1, que corresponden a los pines 0 y 1 del puerto D. En otras placas de desarrollo podemos encontrar hasta más de un par de pines que pueden utilizarse para este fin.
Se debe tener especial cuidado en conectar el pin TX de la placa al RX de la terminal y el RX de la placa al TX de la terminal, o de otro modo la comunicación no podrá realizarse.

BOARD	USB CDC NAME	SERIAL PINS	SERIAL1 PINS	SERIAL2 PINS	SERIAL3 PINS
Uno, Nano, Mini		0(RX), 1(TX)			
Mega		0(RX), 1(TX)	19(RX), 18(TX)	17(RX), 16(TX)	15(RX), 14(TX)
Leonardo, Micro, Yún	Serial		0(RX), 1(TX)		
Uno WiFi Rev.2		Connected to USB	0(RX), 1(TX)	Connected to NINA	
MKR boards	Serial		13(RX), 14(TX)		
Zero	SerialUSB (Native USB Port only)	Connected to Programming Port	0(RX), 1(TX)		
Due	SerialUSB (Native USB Port only)	0(RX), 1(TX)	19(RX), 18(TX)	17(RX), 16(TX)	15(RX), 14(TX)
101	Serial		0(RX), 1(TX)		



Una vez que los pines han sido identificados y se hizo la conexión, debemos definir en la función void setup() que dichos pines serán utilizados para tal fin. Para ello, el lenguaje de programación de los Arduino nos provee la función Serial.begin(), donde deberemos además indicar la velocidad de la conexión (baudrate), normalmente 9600.

Lo anterior serían los pasos mínimos para establecer la comunicación entre la terminal virtual de Proteus y la placa de desarrollo del simulador.

Luego, en las distintas funciones que necesitemos definir, podremos enviar datos a la terminal y recibir datos ingresados por teclado con las funciones Serial.print() (o Serial.println()) y Serial.read() (o alguna de sus variantes, según nuestra necesidad) respectivamente.

```
void loop() {
  if (seguir){
    Serial.println("Seleccione el ejercicio a testear: ");
    Serial.println("1-: Ingrese dos numero y muestre la suma.");
    Serial.println("2-: Preguntar Nombre, Apellido, Direccion y Celular y completar las respuestas.");
    Serial.println("3-: Sumar n numeros ingresados por terminal y calcular su media.");
    Serial.println("4-: Pedir un valor entre 0 y 255 para controlar un diodo led.");
    Serial.println("5-: Ingresar una secuencia de 8 bits que activaran los leds del puerto D.");
    Serial.println("6-: relizar una funcion bool cerradura(tipo) clave");
    seguir = false;
    int dato;
    int control;
    while (Serial.available() == 0){};
        dato = Serial.parseInt();
        int suma;
        //Serial.println(dato);
        switch (dato) {
          case 1:
```