

Comunicación Serie: Conceptos Generales

***Sistemas de Procesamiento de Datos
Tecnatura Superior en Programación.
UTN-FRA***

Autores: *Ing. Darío Cuda*

Revisores: *Lic. Mauricio Dávila*

Versión : 1



Esta obra está bajo una [Licencia Creative Commons Atribución-CompartirIgual 4.0 Internacional](https://creativecommons.org/licenses/by-sa/4.0/).

¿Qué es la comunicación serie?

El puerto serie envía y recibe bytes de información un bit a la vez. Aun y cuando esto es más lento que la comunicación en paralelo, que permite la transmisión de un byte completo por vez, este método de comunicación es más sencillo y puede alcanzar mayores distancias. Por ejemplo, la especificación IEEE 488 para la comunicación en paralelo determina que el largo del cable para el equipo no puede ser mayor a 20 metros, con no más de 2 metros entre cualesquier dos dispositivos; por el otro lado, utilizando comunicación serie el largo del cable puede llegar a los 1200 metros.

Típicamente, la comunicación serie se utiliza para transmitir datos en formato ASCII. Para realizar la comunicación se utilizan 3 líneas de transmisión: (1) Tierra (o referencia), (2) Transmitir, (3) Recibir. Debido a que la transmisión es asincrónica, es posible enviar datos por un línea mientras se reciben datos por otra. Existen otras líneas disponibles para realizar handshaking, o intercambio de pulsos de sincronización, pero no son requeridas. Las características más importantes de la comunicación serie son la velocidad de transmisión, los bits de datos, los bits de parada, y la paridad. Para que dos puertos se puedan comunicar, es necesario que las características sean iguales.

-Velocidad de transmisión (baud rate): Indica el número de bits por segundo que se transfieren, y se mide en baudios (bauds). Por ejemplo, 300 baudios representa 300 bits por segundo. Cuando se hace referencia a los ciclos de reloj se está hablando de la velocidad de transmisión. Por ejemplo, si el protocolo hace una llamada a 4800 ciclos de reloj, entonces el reloj está corriendo a 4800 Hz, lo que significa que el puerto serie está muestreando las líneas de transmisión a 4800 Hz. Las velocidades de transmisión más comunes para las líneas telefónicas son de 14400, 28800, y 33600. Es posible tener velocidades más altas, pero se reduciría la distancia máxima posible entre los dispositivos. Las altas velocidades se utilizan cuando los dispositivos se encuentran uno junto al otro, como es el caso de dispositivos GPIB.

-Bits de datos: Se refiere a la cantidad de bits en la transmisión. Cuando la computadora envía un paquete de información, el tamaño de ese paquete no necesariamente será de 8 bits. Las cantidades más comunes de bits por paquete son 5, 7 y 8 bits. El número de bits que se envía depende en el tipo de información que se transfiere. Por ejemplo, el ASCII estándar tiene un rango de 0 a 127, es decir, utiliza 7 bits; para ASCII extendido es de 0 a 255, lo que utiliza 8 bits. Si el tipo de datos que se está transfiriendo es texto simple (ASCII estándar), entonces es suficiente con utilizar 7 bits por paquete para la comunicación. Un paquete se refiere a una transferencia de byte, incluyendo los bits de inicio/parada, bits de datos, y paridad. Debido a que el número actual de bits depende en el protocolo que se seleccione, el término paquete se usa para referirse a todos los casos.

-Bits de parada: Usado para indicar el fin de la comunicación de un solo paquete. Los valores típicos son 1, 1.5 o 2 bits. Debido a la manera como se transfiere la información a través de las líneas de comunicación y que cada dispositivo tiene su propio reloj, es posible que los dos dispositivos no estén sincronizados. Por lo tanto, los bits de parada no sólo indican el fin de la transmisión sino además dan un margen de tolerancia para esa diferencia de los relojes. Mientras más bits de parada se usen, mayor será la tolerancia a la sincronía de los relojes, sin embargo la transmisión será más lenta.

-Paridad: Es una forma sencilla de verificar si hay errores en la transmisión serie. Para paridad par e impar, el puerto serie fijará el bit de paridad (el último bit después de los bits de datos) a un valor para asegurarse que la transmisión tenga un número par o impar de bits en estado alto lógico. Por ejemplo, si la información a transmitir es 011 y la paridad es par, el bit de paridad sería 0 para mantener el número de bits en estado alto lógico como par. Si la paridad seleccionada fuera impar, entonces el bit de paridad sería 1, para tener 3 bits en estado alto lógico.

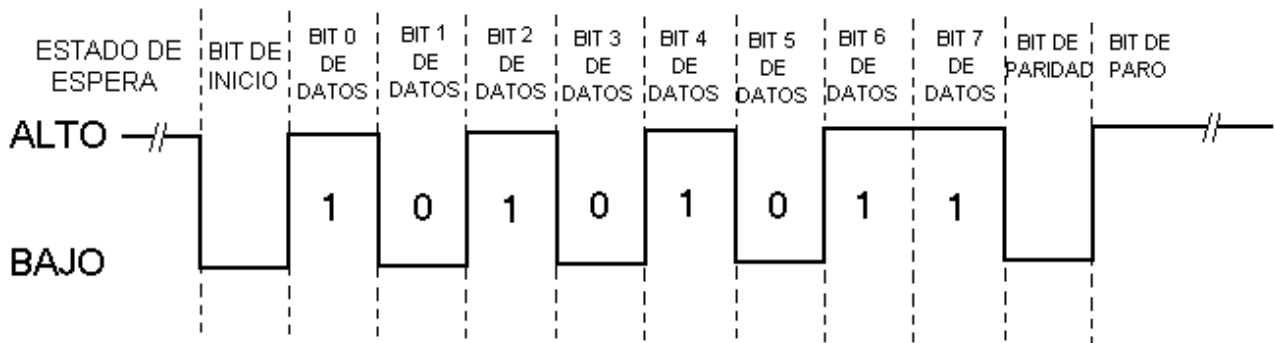
Comunicación serie en Arduino

Prácticamente todas las placas Arduino disponen al menos de una unidad UART. Las placas Arduino UNO y Mini Pro disponen de una unidad UART que operan a nivel TTL 0V / 5V, por lo que son directamente compatibles con la conexión USB. Por su parte, Arduino Mega y Arduino Due disponen de 4 unidades UART TTL 0V / 5V.

Los puertos serie están físicamente unidos a distintos pines de la placa Arduino. Lógicamente, mientras usamos los puertos de serie no podemos usar como entradas o salidas digitales los pines asociados con el puerto serie en uso.

En Arduino UNO y Mini Pro los pines empleados son 0 (RX) y 1 (TX). En el caso de Arduino Mega y Arduino Due, que tienen cuatro puertos de serie, el puerto serie 0 está conectado a los pines 0 (RX) y 1 (TX), el puerto serie 1 a los pines 19 (RX) y 18 (TX) el puerto serie 2 a los pines 17 (RX) y 16 (TX), y el puerto serie 3 a los pines 15 (RX) y 14 (TX).

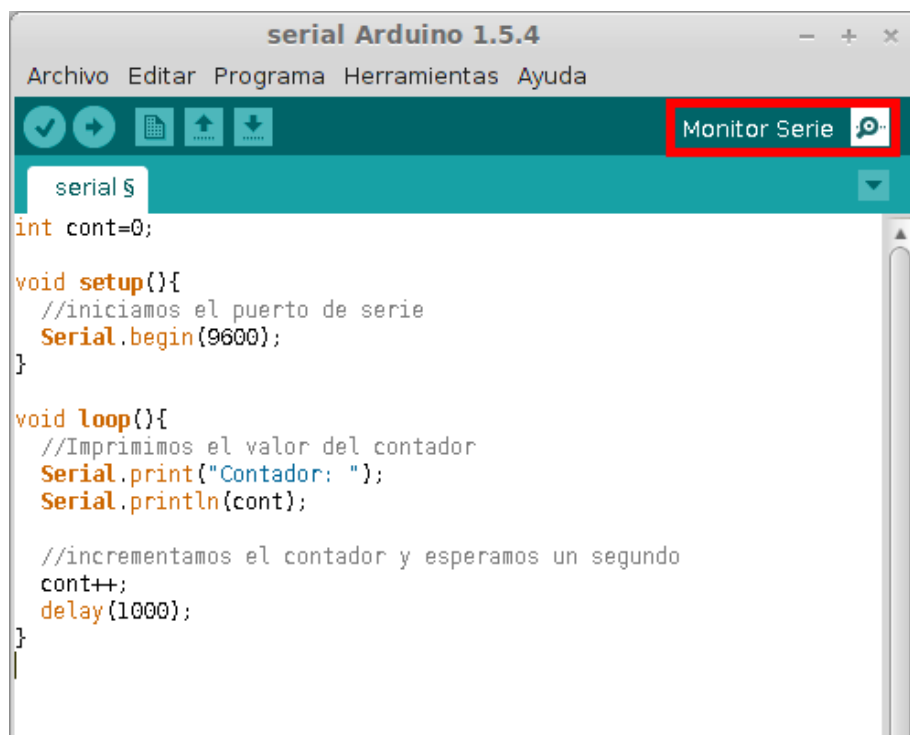
Muchos modelos de placas Arduino disponen de un conector USB o Micro USB conectado a uno de los puertos de serie, lo que simplifica el proceso de conexión con un ordenador, para lo cual poseen un chip que varía según el fabricante y que hace la función de conversión del protocolo USB al Serial asincrónico como por ejemplo el FTDI232 o el CH340G.



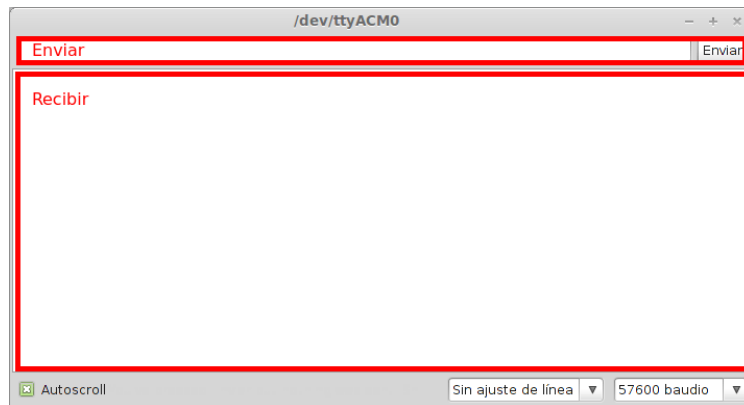
La figura muestra una comunicación Serial asincrónica Unipolar (un solo nivel de tensión), los 8 bits de datos (1 Byte) se delimitan con un bit de Start que es un flanco de bajada de la línea de 1 a 0 y un bit de paro o Stop que es la línea en estado alto. El noveno bit ya no se usa y es el de paridad. Es común en las configuraciones de puerto serial o convertidores de USB a Serial utilizar la palabra 8N1 que significa 8 bits de datos, No paridad y 1 Bit de Stop.

Conexión del Arduino y la PC

Para realizar la conexión mediante puerto serie únicamente es necesario conectar nuestra placa Arduino empleando el mismo puerto que empleamos para programarlo. A continuación abrimos el IDE Standard de Arduino y hacemos click en el "Monitor Serial" como se indica en la imagen.



El monitor de puerto serie es una pequeña utilidad integrada dentro de IDE Standard que nos permite enviar y recibir fácilmente información a través del puerto serie. Su uso es muy sencillo, y dispone de dos zonas, una que muestra los datos recibidos, y otra para enviarlos. Estas zonas se muestran en la siguiente imagen.



Ejemplo

En este ejemplo empleamos el puerto serie para encender o apagar el LED integrado en la placa Arduino. Para ello enviamos un carácter a la placa Arduino, empleando el monitor serial. En caso de enviar 'a' la placa Arduino apaga el LED, y en caso de enviar 'b' lo enciende.

```
int option;
int led = 13;

void setup(){
  Serial.begin(9600);
  pinMode(led, OUTPUT);
}

void loop(){
  //si existe datos disponibles los leemos
  if (Serial.available()>0){
    //leemos la opcion enviada
    option=Serial.read();
    if(option=='a') {
      digitalWrite(led, LOW);
      Serial.println("OFF");
    }
    if(option=='b') {
      digitalWrite(led, HIGH);
      Serial.println("ON");
    }
  }
}
```