**Universidad Autónoma de Baja California**

Facultad de ciencias químicas e ingeniería

****

**Materia.**

Microprocesadores y Microcontroladores.

**Maestro.**

Garcia Lopez Jesus Adan.

**Alumno**

Gonzalez Cardiel Luis Enrique

**Matricula:**

1217258

**Grupo:**

561

**Trabajo:**

Practica No. 9

09/11/2018

**Práctica 9**

**Programación del uC del periférico de comunicación serie utilizando interrupciones.**

**Objetivo:** Mediante esta práctica el alumno aprenderá el uso básico para inicializar y operar, bajo un esquema de interrupciones, el puerto serie del microcontrolador.

**Equipo:**  - Computadora Personal

- Módulo T-Juino

**Teoría:** - Manejo del Periférico de Comunicación Serie 0 (UART0) del microcontrolador ATmega1280/2560

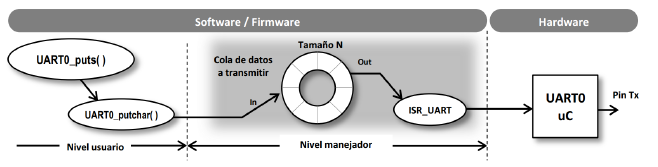
- Secuencias de escape ANSI.

**Actividades a realizar:**

**1. Bajo los esquemas de la Figura 1 y Figura 2, implemente el manejador para recibir y enviar datos vía puerto serie 0 (UART0) utilizando interrupciones.**

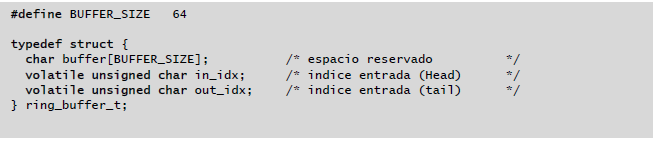
**Transmisión y Recepción de datos bajo el uso de interrupciones.** Para hacer uso eficiente de la transmisión y recepción de datos bajo el esquema de interrupciones es necesario una cola circular para transmitir, y otra para recibir.

En la cola de transmisión se introducen los datos a transmitir para que después la rutina de servicio de interrupción (ISR) correspondiente los tome al momento adecuado para su transmisión. En la Figura 1 se observa que la función UART0\_putchar( ) es la que introduce cada dato a la cola circular siempre y cuando exista lugar para hacerlo; de lo contrario tendrá que esperar a que se libere un lugar.



**Figura 1**. Esquema para el manejo de paquetes de transmisión mediante cola circular.

**Listado 1**. Tipo de dato ring\_buffer\_t.Y



La ISR correspondiente a la transmisión del UART0 es la que tiene la responsabilidad de tomar dato por dato de la cola circular e introducirlos al UART0, esto a cada momento que el registro UDR0 (de transmisión) esté libre para poder escribir en dicho registro. Si por algún motivo no existen datos en la cola circular la ISR debería de deshabilitar su interrupción correspondiente para evitar que continuamente se invoque la interrupción.

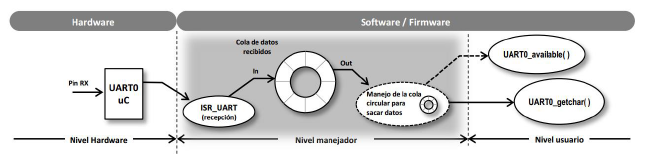
Considerando lo anterior, implementar la siguiente rutina:

a)  **ISR( SIG\_USART0\_DATA )** : Rutina de servicio de interrupción para el evento de registro de transmisión vació. Esta rutina extrae de la cola circular el dato que será transmitido por el UART0.

Y modificar:

b) void **UART0\_putchar**(char data): Función que coloca el dato en la cola circular.

En la cola de recepción, la ISR es quien introduce los datos recibidos. En la Figura 2 se observa que UART0\_getchar( ) es quien extrae los datos de la cola. implemente el manejador para recibir datos vía puerto serie 0 (UART0). Considera que se requiere la implementación de un mecanismo de recepción almacenar los datos en la cola circular por lo que se requieren las siguientes funciones.



**Figura 2.** Esquema para el manejo de paquetes de recepción mediante cola circular.

Bajo el esquema de la Figura 2 implemente el manejador para recibir datos vía puerto serie 0 (UART0). Considera que se requiere la implementación de un mecanismo de recepción almacenar los datos en la cola circular por lo que se requieren las siguientes funciones:

c)  **ISR( SIG\_USART0\_RECV )**: Rutina de servicio de interrupción para el evento de recepción completa. Esta rutina inserta a la cola circular el dato que fue recibido por el UART0.

d) uint8\_t **UART0\_available**( void ): Función que retorna 1 si existe(n) dato(s) en la cola circular.

e) char **UART0\_getchar**(void): Función toma el dato correspondiente a salir de la cola circular. Esto si existe alguno dato en la cola circular, de lo contrario espera a que exista uno para tormarlo y retornarlo.

**2. Implementar la siguiente función:**

UART\_Ini(uint8\_t com, uint16\_t baudrate, uint8\_t size,

uint8\_t parity, uint8\_t stop)

Función que inicializa el periférico del UART en un esquema de interrupciones. Y la configuración es dada por los parámetros, donde:

* **com**: representa el número de UART a configurar. Considerar 0 y 1.
* **baudrate**: representa la velocidad en Baud de configuración, puede ser no estándar.
* **size**: representa el número de bits de los datos con los que operará el UARTx. Considerar de 5 a 8 bits.
* **parity**: representa el tipo de paridad con los que operará el UARTx. Considerar 0: No paridad, 1: impar, 2: par.
* **stop**: representa el número de bits de paro con los que operará el UARTx.

Considerar 1 ó 2.

**3. Reutilizar las funciones de la Práctica 2:**

a) void gets(char \*str)→ void UART0\_gets(char \*str)

Función que retorna una cadena haciendo uso de UART0\_getchar(), la cadena se retorna en el apuntador str.

b) void puts(char \*str) → void UART0\_puts(char \*str)

Función que imprime una cadena mediante UART0\_putchar().

c) void itoa(char\* str, uint16\_t number, uint8\_t base)

d) uint16\_t atoi(char \*str)

4. En base a las secuencias de escape, implementar las siguientes funciones:

1. void clrscr( void )

Funcion que limpia la terminal mediante la secuencia de escape.

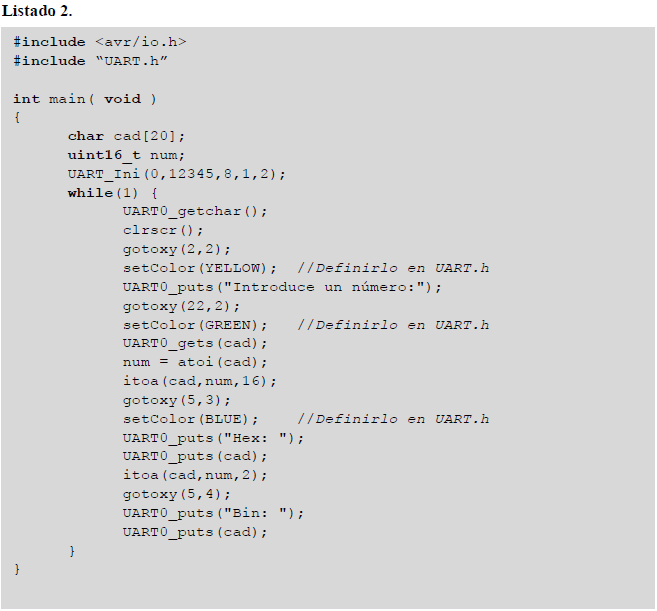
1. void setColor(uint8\_t color)

Función que envía la secuencia de escape para configurar el color del texto que se desplegará en la terminal.

1. void gotoxy(uint8\_t x, uint8\_t y)

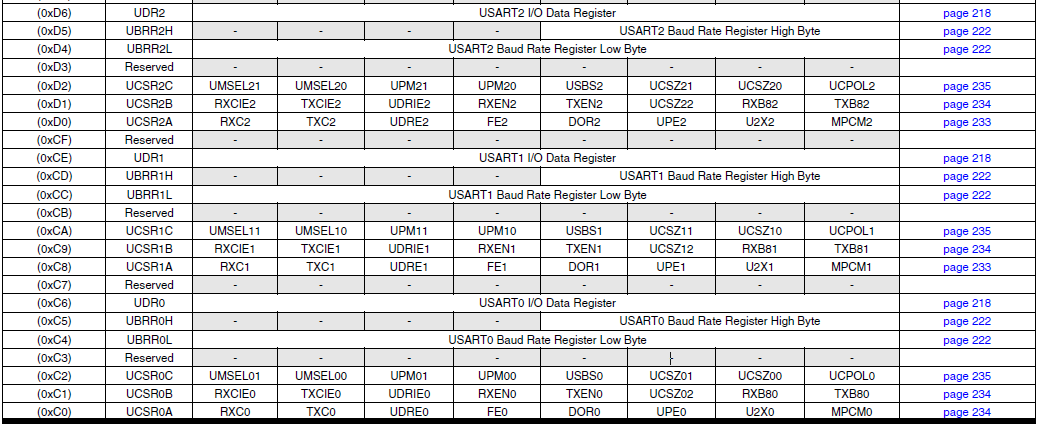
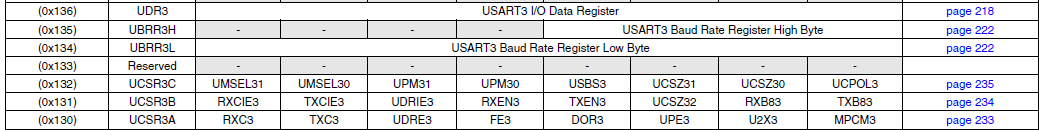
Función que posiciona el cursor en la terminal en la coordenada x,y que lleguen como parámetro, utilizando la secuencia de escape.

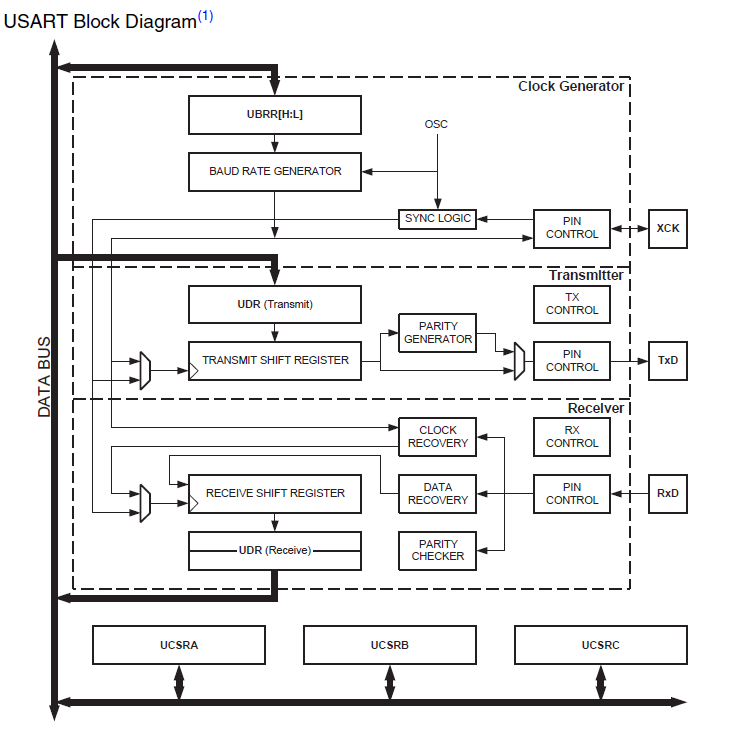
Utilizar el Listado 2 para comprobar el funcionamiento de las funciones.

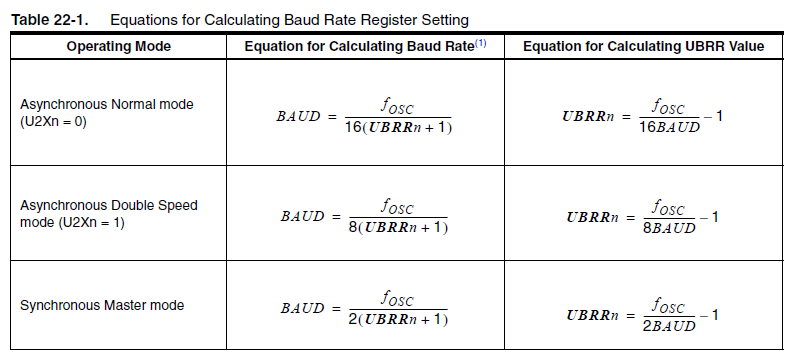


**Teoria:**

**Manejo del Periférico de Comunicación Serie 0 (UART0) del microcontrolador ATmega1280/2560**







Nota: 1. La velocidad en baudios se define como la velocidad de transferencia en bits por segundo (bps).

**BAUD** Baud rate (en bits por segundo, bps).

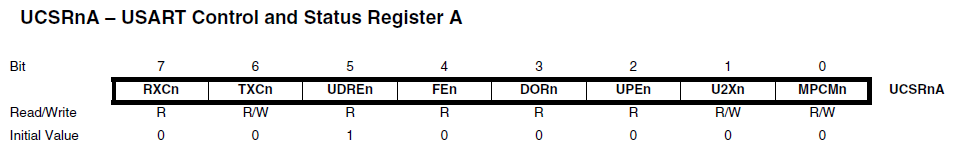
**fOSC** Sistema de frecuencia de reloj del oscilador.

**UBRRn** Contenido de los registros UBRRHn y UBRRLn, (0-4095).

**Operación de doble velocidad (U2Xn)**

La velocidad de transferencia se puede duplicar estableciendo el bit U2Xn en UCSRnA. La configuración de este bit solo tiene efecto para la operación asíncrona. Establezca este bit a cero cuando use la operación síncrona.

La configuración de este bit reducirá el divisor del divisor de la velocidad de transmisión de 16 a 8, duplicando efectivamente la velocidad de transferencia para la comunicación asíncrona. Sin embargo, tenga en cuenta que, en este caso, el Receptor solo usará la mitad del número de muestras (reducido de 16 a 8) para el muestreo de datos y la recuperación del reloj, y por lo tanto, se requiere una configuración de la velocidad en baudios y un reloj del sistema más precisos cuando se usa este modo. El transmisor, no hay desventajas.



**• Bit 7 - RXCn: USART Receive Complete**

Este bit de indicador se establece cuando hay datos no leídos en el búfer de recepción y se borra cuando el búfer de recepción está vacío (es decir, no contiene ningún dato no leído). Si el receptor está deshabilitado, el búfer de recepción se vaciará y, en consecuencia, el bit RXCn se pondrá a cero. El indicador RXCn se puede utilizar para generar una interrupción de recepción completa (consulte la descripción del bit RXCIEn).

**• Bit 6 - TXCn: USART Transmit Complete**

Este bit de indicador se establece cuando todo el cuadro en el Registro de cambio de transmisión se ha desplazado y no hay datos nuevos presentes en el búfer de transmisión (UDRn). El bit del indicador TXCn se borra automáticamente cuando se ejecuta una interrupción de transmisión completa, o se puede borrar escribiendo un uno en su ubicación de bit. El indicador TXCn puede generar una interrupción de transmisión completa (consulte la descripción del bit TXCIEn).

**• Bit 5 - UDREn: Registro de datos USART vacío**

El indicador UDREn indica si el búfer de transmisión (UDRn) está listo para recibir nuevos datos. Si UDREn es uno, el búfer está vacío y, por lo tanto, está listo para ser escrito. El indicador UDREn puede generar una interrupción de registro de datos vacío (consulte la descripción del bit UDRIEn).

UDREn se configura después de un reinicio para indicar que el transmisor está listo.

**• Bit 4 - FEn: Error de trama**

Este bit se establece si el siguiente carácter en el búfer de recepción tuvo un error de trama cuando se recibió, es decir, cuando el primer bit de parada del siguiente carácter en el búfer de recepción es cero. Este bit es válido hasta que se lee el búfer de recepción (UDRn). El bit FEn es cero cuando el bit de parada de los datos recibidos es uno. Siempre establezca este bit a cero al escribir en UCSRnA.

**• Bit 3 - DORn: Data OverRun**

Este bit se establece si se detecta una condición de OverRun de datos. Un Data OverRun se produce cuando el búfer de recepción está lleno (dos caracteres), es un nuevo carácter en espera en el registro de cambio de recepción y se detecta un nuevo bit de inicio. Este bit es válido hasta que se lee el búfer de recepción (UDRn). Siempre establezca este bit a cero al escribir en UCSRnA.

**• Bit 2 - UPEn: Error de paridad USART**

Este bit se establece si el siguiente carácter en el búfer de recepción tuvo un Error de paridad cuando se recibió y la Comprobación de paridad se habilitó en ese punto (UPMn1 = 1). Este bit es válido hasta que se lee el búfer de recepción (UDRn). Siempre establezca este bit a cero al escribir en UCSRnA.

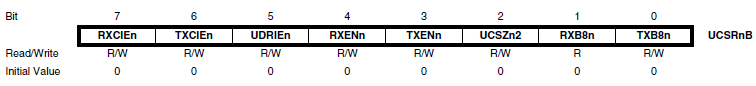
**• Bit 1 - U2Xn: duplica la velocidad de transmisión de USART**

Este bit solo tiene efecto para la operación asíncrona. Escriba este bit a cero cuando use la operación síncrona.

Escribir este bit en uno reducirá el divisor del divisor de velocidad en baudios de 16 a 8, duplicando efectivamente la velocidad de transferencia para la comunicación asíncrona.

**• Bit 0 - MPCMn: Modo de comunicación multiprocesador**

Este bit habilita el modo de comunicación multiprocesador. Cuando el bit MPCMn se escribe en uno, se ignorarán todas las tramas entrantes recibidas por el receptor USART que no contienen información de dirección. El transmisor no se ve afectado por la configuración de MPCMn.



**• Bit 7 - RXCIEn: RX Complete Interrupt Enable n**

Escribir este bit en uno habilita la interrupción en el indicador RXCn. Una interrupción de recepción completa de USART se generará solo si el bit RXCIEn se escribe en uno, el Indicador de interrupción global en SREG se escribe en uno y el bit RXCn en UCSRnA se establece.

**• Bit 6 - TXCIEn: habilitación de interrupción completa de TX n**

Escribir este bit en uno habilita la interrupción en el indicador TXCn. Se generará una interrupción de transmisión completa de USART solo si el bit TXCIEn se escribe en uno, el Indicador de interrupción global en SREG se escribe en uno y el bit TXCn en UCSRnA se establece.

**• Bit 5 - UDRIEn: Registro de datos USART Vacío Interrupción Habilitar n**

Escribir este bit en uno habilita la interrupción en el indicador UDREn. Se generará una interrupción de Registro de Datos Vacíos solo si el bit UDRIEn se escribe en uno, el Indicador de Interrupción Global en SREG se escribe en uno y el bit UDREn en UCSRnA se establece.

**• Bit 4 - RXENn: receptor habilitado n**

Escribir este bit en uno habilita el receptor USART. El receptor anulará la operación normal del puerto para el pin RxDn cuando esté habilitado. Al deshabilitar el receptor, se vaciará el búfer de recepción y se invalidarán los indicadores FEn, DORn y UPEn.

**• Bit 3 - TXENn: Transmisor habilitado n**

Escribir este bit en uno habilita el transmisor USART. El transmisor anulará la operación normal del puerto para el pin TxDn cuando esté habilitado. La desactivación del transmisor (escribir TXENn en cero) no entrará en vigencia hasta que se completen las transmisiones en curso y pendientes, es decir, cuando el Registro de cambios de transmisión y el Registro de almacenamiento intermedio de transmisión no contengan datos para transmitir. Cuando está deshabilitado, el transmisor ya no anulará el puerto TxDn.

**• Bit 2 - UCSZn2: Tamaño de carácter n**

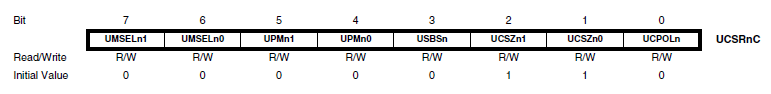
Los bits UCSZn2 combinados con el bit UCSZn1: 0 en UCSRnC establecen el número de bits de datos (tamaño de caracteres) en una trama que utilizan el receptor y el transmisor.

**• Bit 1 - RXB8n: Bit de recepción de datos 8 n**

RXB8n es el noveno bit de datos del carácter recibido cuando se opera con tramas en serie con nueve bits de datos. Debe leerse antes de leer los bits bajos de UDRn.

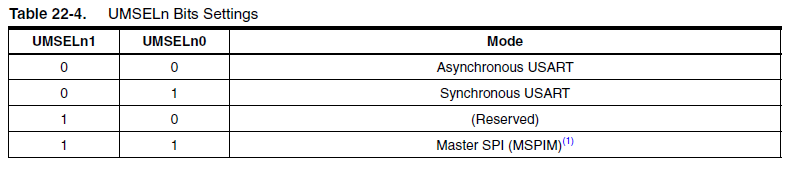
**• Bit 0 - TXB8n: Bit de transmisión de datos 8 n**

TXB8n es el noveno bit de datos en el carácter que se transmitirá cuando se opera con tramas en serie con nueve bits de datos. Debe escribirse antes de escribir los bits bajos en UDRn.



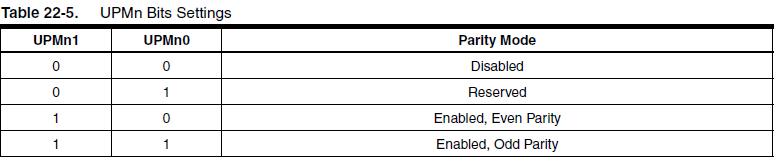
**• Bits 7: 6 - UMSELn1: 0 USART Mode Select**

Estos bits seleccionan el modo de operación del USARTn



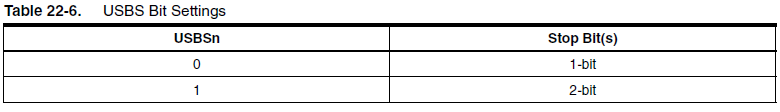
**• Bits 5: 4 - UPMn1: 0: Modo de paridad**

Estos bits habilitan y establecen el tipo de generación de paridad y verificación. Si está habilitado, el Transmisor generará y enviará automáticamente la paridad de los bits de datos transmitidos dentro de cada trama. El receptor generará un valor de paridad para los datos entrantes y lo comparará con la configuración de UPMn. Si se detecta una falta de coincidencia, se establecerá el indicador UPEn en UCSRnA.



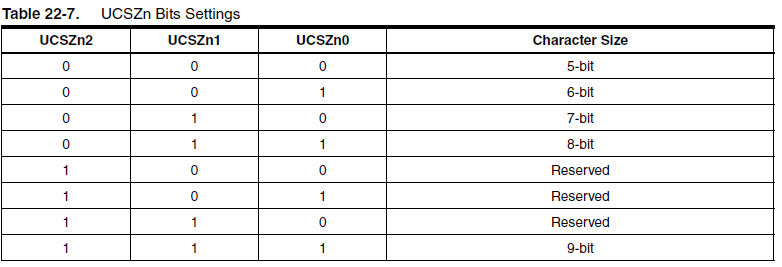
**• Bit 3 - USBSn: Stop Bit Select**

Este bit selecciona el número de bits de parada a ser insertados por el transmisor. El receptor ignora esta configuración.



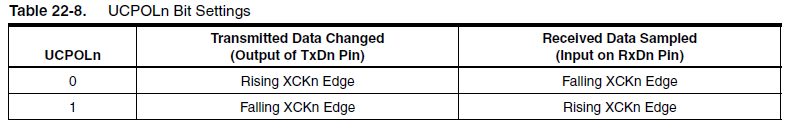
**• Bit 2: 1 - UCSZn1: 0: Tamaño del carácter**

Los bits UCSZn1: 0 combinados con el bit UCSZn2 en UCSRnB establecen el número de bits de datos (Tamaño de Caracteres) en una trama que utilizan el Receptor y el Transmisor.



**• Bit 0 - UCPOLn: polaridad del reloj**

Este bit se usa solo para el modo síncrono. Escriba este bit a cero cuando se utilice el modo asíncrono. El bit UCPOLn establece la relación entre el cambio de salida de datos y la muestra de entrada de datos, y el reloj síncrono (XCKn).



**Secuencias de escape ANSI.**

Las secuencias de escape ANSI permiten enviar información de control a la consola para cambiar los atributos del texto representado.

Así es posible seleccionar:

1. El estilo del texto: normal, claro, subrayado, parpadeante, inverso y oculto
2. El color del fondo
3. El color del texto

El funcionamiento es muy sencillo, entre la información que se envía de salida a la consola, se incluyen las secuencias de escape dando instrucciones de cómo se ha de representar el texto a continuación.

Por ejemplo, si con printf se vuelca "Palabra %sresaltada%s" insertando en el primer %s la cadena de control para hacer que el texto sea verde, el texto que se escribe a continuación ('resaltada') tendrá color verde. En el último %s se debería introducir la secuencia de escape para volver a la normalidad. De no ser así, toda salida posterior sería de color verde !!!

**Sintáxis:**

Las secuencias de escape se pueden utilizar desde cualquier script o programa que envíe información a la consola. Por ejemplo, en C podríamos definir las siguientes secuencias:

1. const char \*const normal = "\033[0m";
2. const char \*const verde = "\033[0;40;32m";
3. const char \*const subrayado\_fazul\_verde = "\033[4;44;32m";

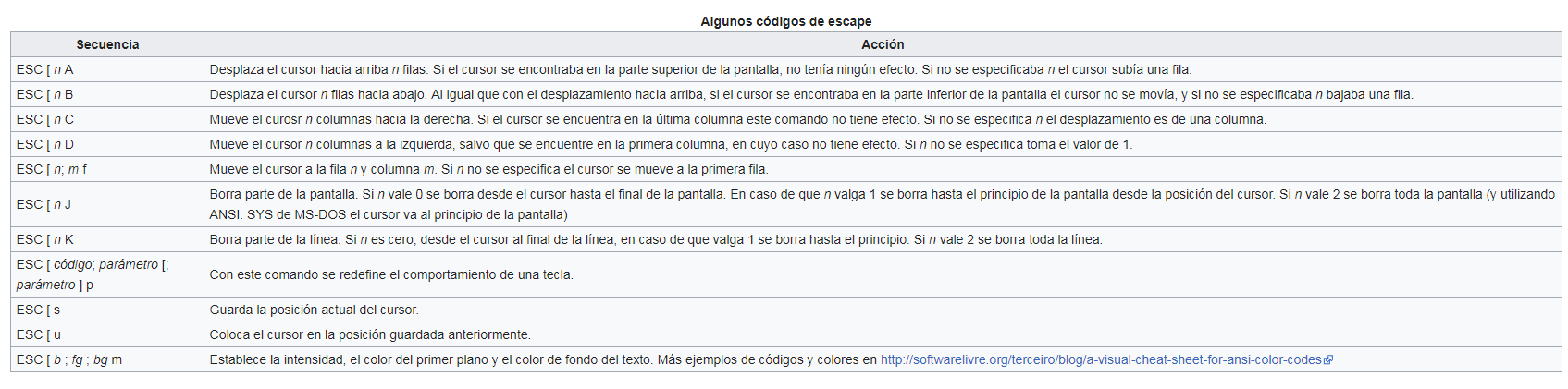
La primera selecciona el estilo, fondo y color de texto normal. Es la cadena que deberiamos utilizar para terminar los efectos anteriormente aplicados. La segunda cadena determina que el color de texto sea verde. La tercera cadena determina que el texto ha de estar subrayado, ser de color verde y estar sobre fondo azul.

La sintáxis sería: "\033[x;xx;xxm" dónde cada 'x' representa un dígito.

El primer dígito especifica el estilo:

* 0 -> Normal
* 1 -> Claro (el color se diluye, permite hacer dos tonos de cada color: azul/azulclaro, rojo/rojoclaro...)
* 4 -> Subrayado
* 5 -> Parpadeante
* 7 -> Inverso
* 8 -> Oculto (Pensado para pedir contraseñas al usuario)

**Siendo ESC = \033**



**Conclusión:**

El aprender a usar un microcontrolador con la configuración USART es muy importante ya que a través de este nos podemos comunicar con módulos wifi,bluethooth una computadora entre otros.Me parecio interesante como el micro se comunica por el UART0 así que me puse a investigar y mire que hay otro microcontrolador el Atmega16u2 el que es el que se encarga de hacer la comunicación de la computadora al Atmega2560/1280. Investigando tambien mire que existen los FTDI y con estos nos podríamos comunicar con los demás USART’s sin necesidad de usar el 0.