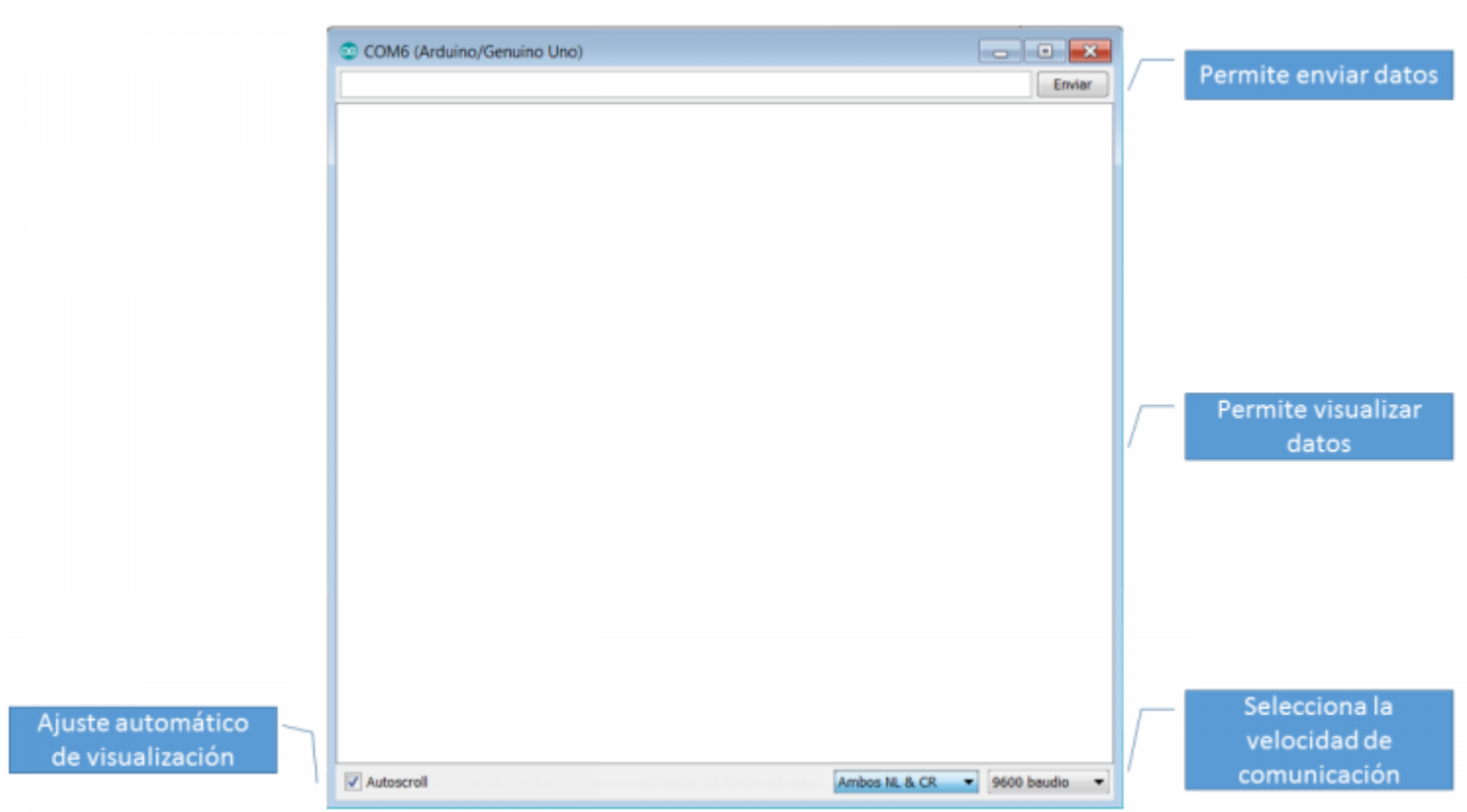
MÓDULO 4: PROJECTO 0.3 " Monitor serie y el serial plotter"

CURSO PROGRAMACIÓN DE PLACAS ROBÓTICAS

Monitor serie

- Mediante este monitor podemos ver los valores de nuestras variables, así como mensajes de texto que nos permitirán conocer como esta funcionando nuestro programa, o simplemente interactuar con el.
- El monitor serie consiste en una consola de entrada y salida, esto quiere decir que podremos mostrar datos enviados por nuestra placa arduino y también podremos enviar datos a nuestra placa.
- Podremos acceder a el a través del menú Herramientas/Monitor serie o haciendo click en el icono que parece una lupa ubicado a la derecha en el IDE

Monitor serie



Monitor serie

- La comunicación serie se lleva a cabo por los pines TX / RX y usa niveles lógicos TTL (5V o 3.3V dependiendo de la placa).
- Esta se utiliza para la comunicación entre la placa Arduino y una computadora u otros dispositivos.
- Todas las placas Arduino tienen al menos un puerto serie (también conocido como UART o USART). Por lo tanto, si utilizamos esta función, no se pueden utilizar los pines 0 y 1 para la entrada o salida digital.

Funciones básicas del Monitor serie

- **begin()**
  - Establece la velocidad de datos en bits por segundo (baudios) para la transmisión de datos en serie. Para comunicarse con el ordenador, utilice una de estas velocidades: 300, 600, 1200, 2400, 4800, **9600**, 14400, 19200, 28800, 38400, 57600 o **115200**.Por tanto, su ejecución es imprescindible antes de realizar cualquier transmisión por dicho canal. Se debe indicar la velocidad del canal como argumento.
  - **Sintaxis**
    - Serial.begin(velocidad)
    - Serial.begin(velocidad, configuracion)
  - **Parametros**
    - **velocidad**: es la velocidad de comunicación del puerto serie en bits por segundo (baudios)
    - **Configuracion**: selecciona el numero de datos, paridad y bits de parada. Los valores validos son: SERIAL_5N1, SERIAL_6N1, SERIAL_7N1, **SERIAL_8N1** (por defecto), SERIAL_5N2, SERIAL_6N2, SERIAL_7N2, SERIAL_8N2, SERIAL_5E1, SERIAL_6E1, SERIAL_7E1, SERIAL_8E1, SERIAL_5E2, SERIAL_6E2, SERIAL_7E2, SERIAL_8E2, SERIAL_5O1, SERIAL_6O1, SERIAL_7O1, SERIAL_8O1, SERIAL_5O2, SERIAL_6O2, SERIAL_7O2 y SERIAL_8O2.

Funciones básicas del Monitor serie

- **end()**
  - No tiene ningún argumento ni devuelve nada, y se encarga de cerrar el canal serie. Permitiendo que los pines RX y TX sean usados para entradas y salidas generales.
  - **Sintaxis**
    - Serial.end()

- **print()**
  - Envía a través del canal serie un dato (especificado como parámetro) desde la placa arduino hacia el exterior. Ese dato puede ser de cualquier tipo: carácter, cadena, número entero, número decimal (por defecto de dos decimales), etc. Si el dato se especifica explícitamente (en vez de a través de una variable), hay que recordar que los caracteres se han de escribir entre comillas simples y las cadenas entre comillas dobles.
  - **Sintaxis**
    - Serial.print(valor, formato)
  - **Parametros**
    - valor: el valor a imprimir (cualquier tipo de dato)
    - formato: Especifica la base numérica (BIN (binario o base 2), OCT (octal o base 8), DEC (decimal o base 10), HEX (hexadecimal o base 16)) o el número de decimales (para los datos float).

Funciones básicas del Monitor serie

- **println()**
  - Hace exactamente lo mismo que Serial.print(), pero además, añade automáticamente al final de los datos enviados dos caracteres extra: el de retorno de carro (código ASCII nº 13)(\r) y el de nueva línea (código ASCII nº 10) (\n). La consecuencia es que al final de la ejecución de Serial.println() se efectúa un salto de línea.
  - **Sintaxis**
    - Serial.println(valor, formato)
  - **Parámetros**
    - valor: el valor a imprimir (cualquier tipo de dato)
    - formato: Especifica la base numérica (BIN (binario o base 2), OCT (octal o base 8), DEC (decimal o base 10), HEX (hexadecimal o base 16)) o el número de decimales (para los datos float).

Ejemplo 01

```
void setup() {
  Serial.begin(9600);//Inicia la comunicacion serie a 9600 baudios
  Serial.println("Hola monitor serie");//imprime una secuencia de texto
  Serial.println("Asi se imprime texto");  //imprime una secuencia de texto
}
void loop() {}
```

Ejemplo 2 – Imprimir numero en diferentes formatos

```
int analogValue = 0;  // variable para guardar el valor analogico

void setup() {

    Serial.begin(9600);  //Inicia la comunicacion serie a 9600 baudios

}

void loop() {

   analogValue = analogRead(0);  //lee el valor analogico en el pin A0

   Serial.println(analogValue); // Imprime el valor en varios formatos

   Serial.println(analogValue, DEC);  // Imprime el valor en decimal

   Serial.println(analogValue, HEX);  // Imprime el valor en hexadecimal

   Serial.println(analogValue, OCT);  // Imprime el valor en octal

   delay(100);  //espera 100 millisegundos para la siguiente lectura

}
```

Funciones básicas del Monitor serie

- **available()**
  - Devuelve el número de bytes –caracteres– disponibles para ser leídos y que provienen del exterior a través del canal serie. Estos bytes ya han llegado al microcontrolador y permanecen almacenados temporalmente en una pequeña memoria de 64 bytes que tiene la placa arduino –llamada "buffer"– hasta que sean procesados mediante la instrucción *Serial.read().*
  - Esta función devuelve el numero de bytes para leer. Si no hay bytes para leer, esta instrucción devolverá un 0. No tiene parámetros.
  - **Sintaxis**
    - Serial.available()

Ejemplo 3 – Leer el puerto serie con la función read()

```
int byteEntrante = 0;   // guarda los datos de entrada serial

void setup() {

      Serial.begin(9600);// inicia la comunicación serie a 9600 baudios

 }

void loop() {

   // lee e imprime en el monitor serie cuando se reciben datos

  if (Serial.available() > 0) {

     byteEntrante = Serial.read(); // lee el byte entrante

     Serial.print("He recibido: "); // imprime el byte entrante

     Serial.println(byteEntrante, DEC);

       }
}
```

Funciones básicas del Monitor serie

- **read()**
  - Devuelve el primer byte aún no leído de los que estén almacenados en el buffer de entrada del chip TTL-UART. Al hacerlo, lo elimina de ese buffer. Para leer el siguiente byte, se ha de volver a ejecutar *Serial.read().* Y hacer así hasta que se hayan leído todos. Cuando no haya más bytes disponibles, *Serial.read()* devolverá -1. Esta función no tiene parámetros.
  - **Sintaxis**
    - Serial.read()

Funciones básicas del Monitor serie

- **parseFloat()**
  - Lee del buffer de entrada (eliminándolos de allí) todos los datos hasta que se encuentre con un número decimal. Su valor de retorno – de tipo “float”– será entonces ese número decimal encontrado. Cuando detecte el primer carácter posterior no válido, dejará de leer (y por tanto, no seguirá eliminando datos del buffer). Esta instrucción no tiene parámetros.
  - **Sintaxis**
    - Serial.parseFloat()
- **parseInt()**
  - Lee del buffer de entrada (eliminándolos de allí) todos los datos hasta que se encuentre con un número entero. Su valor de retorno –de tipo “long”– será entonces ese número entero encontrado. Cuando detecte el primer carácter posterior no válido, dejará de leer (y por tanto, no seguirá eliminando datos del buffer). Esta instrucción no tiene parámetros.
  - **Sintaxis**
    - Serial.parseInt()