COMUNICACIÓN CON EL EXTERIOR

El puerto serie en Arduino y los tipos String

OBJETIVOS

.

- Comprender la comunicación vía puerto serie.
- Utilizar la librería Serial.
- Operaciones con enteros.
- Los tipos String y char.
- Operando con Strings.
- La instrucción while.

MATERIAL REQUERIDO.

	**Arduino Uno o similar**. Un PC con el entorno de Arduino correctamente instalado y configurado.

COMUNICACIÓN SERIE CON EL MUNDO EXTERIOR

Más antes que después, vamos a necesitar comunicar nuestro Arduino con nuestro PC. Las razones son varias, enviarle órdenes o recibir información o señales por ejemplo.

Los PCs disponen de teclados, pantallas y adaptadores de red, pero con Arduino tenemos que usar el puerto USB que establecerá una **conexión en serie** con nuestro PC.

La comunicación en serie es muy sencilla, bastan dos hilos para enviar una diferencia de tensión entre ellos y poder marcar niveles alto (5V) y bajo(0V) y con esto podemos transmitir información digital. Ahora solo nos falta pactar dos cosas entre quien envía y quien recibe:

- Un **código común** para codificar los caracteres que enviamos.
- Un **acuerdo de velocidad** para saber a qué ritmo hay que leer los datos.

El código común que vamos a usar con Arduino se llama **código ASCII** y es estándar en todos los PCs. Es una forma de codificar las letras mediantes números que representas estos caracteres. Recordad que solo podemos transmitir unos y ceros.

Así por ejemplo la letra A se representa por el numero 65, la B el 66, C el 67… Prácticamente todos los PCs actuales utilizan este código y eso incluye a Windows, Mac y Linux (y por eso podemos leer emails enviados desde distintas plataformas), pero es importante comprender que esté es uno más entre varios códigos de caracteres posibles (EBCDIC por ejemplo).

- *Actualmente, en realidad, se suele usar una extensión del **código ASCII** (llamada **Unicode**) que permita el uso de caracteres no incluidos en la tabla original, y que permita representar caracteres como las Ñ, o acentos para el español, pero también alfabetos distintos como el Kanji chino o el alfabeto cirílico. Y este es el motivo por el que podéis leer las letras chinas o rusas en las páginas de internet de estos países..*

El otro factor a pactar para realizar una comunicación serie es la velocidad. Dado que solo disponemos de dos hilos para transmitir, necesitamos saber cuándo hay que leer la línea y esto se hace estableciendo un acuerdo de velocidad. Si la velocidad de envío es distinta de la velocidad de lectura, el mensaje final será irreconocible.

Buena parte de los errores de comunicación serie programando con Arduino se suelen deber a una diferencia de velocidad entre el emisor y el receptor

ESTABLECIENDO LA COMUNICACIÓN SERIE

.

Arduino dispone de una librería serie incluida llamada Serial, que nos permite envía información al PC y para usarla simplemente tenemos que pedirle en nuestro setup() que la incluya. La instrucción que se encarga es:

```
Serial.begin( velocidad )  ;
```

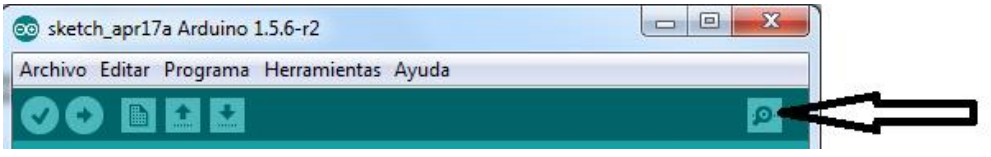
- *Nótese que Serial tiene la S mayúsculas y que C++ diferencia entre mayúsculas y minúsculas*

La velocidad es una valor entre 300 y 115.200 bits por segundo. Y suele ser costumbre establecerla en 9600 (el valor por defecto) pero no hay ninguna razón para ello y esta no es una velocidad especialmente alta.

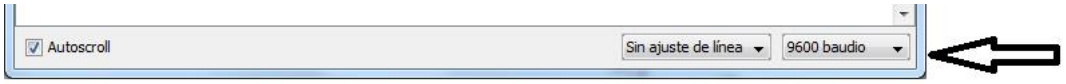
Para enviar un mensaje desde Arduino a nuestro PC podemos usar las funciones Serial.print() y Serial.println().Veamos un ejemplo:

```
int LED = 10 ; int boton = 6 ;
bool estado = false ;
void setup()
     {
         Serial.begin(9600) ;                // Inicializa el Puerto seria 9600 bits por segundo
     }
void loop()
    {
         int i = 54 ;
         Serial.println( i );
    }
```

El println() enviara el valor de i al puerto serie de Arduino (*repetidamente*). Para leerlo en nuestro PC necesitamos un monitor de puerto serie. El IDE de Arduino incluye uno muy sencillo, pero suficiente que se invoca con el botón del monitor:



Necesitamos además asegurarnos de que la velocidad de conexión es la misma en ambos extremos. Fíjate en la parte inferior derecha del monitor serie:



Normalmente la velocidad por defecto son los 9600 bits por segundo o **baudios** en los que hemos programado nuestra puerta serie, y si lo desplegáis, veréis las diferentes velocidades aceptables para Arduino.

- *Estrictamente hablando, bits por segundo y baudios no son exactamente lo mismo salvo bajo ciertas condiciones particulares que en Arduino se cumplen, por lo que aquí podemos usarlos como sinónimos.*
- *En el mundo Arduino parece haber un acuerdo de usar velocidades bajas como 9600 en lugar de más altas como 115200, para evitar problemas. Esto es algo que hace años estaba justificado por problemas de transmisión, pero con la tecnología actual no hay motivo para ello. Es más, en cuanto necesitemos utilizar dispositivos de comunicaciones como adaptadores Ethernet o BlueTooth para comunicarnos, la velocidad tendrá que subir necesariamente.*

Ahora que sabemos enviar información y resultados al PC, vamos a ver cómo podemos operar con enteros y mostrar el resultado en la puerta serie. En C++ los operadores numéricos son los normales en cálculo (y algunos menos frecuentes):

- Adición: +
- Resta: –
- Multiplicación: *
- División entera: / Cociente sin decimales (puesto que operamos con enteros
- Resto: % Devuelve el resto de una división.

En C++ tenemos que expresar las operaciones matemáticas en una sola línea y utilizar paréntesis para garantizar que se opera como necesitamos. Vamos con algunos ejemplos:

OPERACIÓN	RESULTADO	COMENTARIO
int i = 4 * 2	resultado = 8	
int i = 4 * 2 / 3	resultado = 2	Porque desprecia los decimales al ser entero
int i = 14 % 3	resultado = 2	El resto de 14 entre 3
Int i = 2 + 8 / 2	resultado = 6	Calcula primero la división.
Int i = (2+8) / 2	resultado = 5	El paréntesis fuerza a que se realice primero la suma

Dada una expresión, la precedencia de operadores indica que operaciones se realizaran antes y cuales después en función de su rango. Para los que se inician en C++ no es fácil saber que operadores tienen preferencia, por lo que es más seguro que ante la duda uséis paréntesis.

Los paréntesis fuerzan las operaciones de una forma clara y conviene utilizarlos ante la duda porque de otro modo, detectar los errores de operación puede volverse muy difícil especialmente cuando uno empieza a programar.

El operador resto es más útil de lo que parece a primera vista porque nos permite saber si un numero es múltiplo de otro. Supongamos que queremos saber si un número dado es par.

Podríamos escribir un programa como este:

```
void setup()
     {
        Serial.begin(9600) ; // Inicializa el Puerto serie
     }
void loop()
    {
       int i = 27 ; //El número en cuestión
       if ( i % 2 == 0)
            Serial.println("Es par.") ;
      else
            Serial.println("Es impar");
    }
```

Dando a i distintos valores podemos comprobar cómo funciona el operador resto %. Volveremos sobre esto cuando veamos algunos ejemplos de cómo calcular números primos.

En este programa hemos usado de un modo diferente el Serial.println() pasándole una String de texto entrecomillada. Serial.print() envía el texto (entrecomillado) que le pongamos pero no da salto de línea cuando termina. En cambio Serial.println() hace lo mismo e incluye al final ese salto de línea.

```
void setup()
     {
        Serial.begin(9600) ; // Inicializa el Puerto serie
     }
void loop()
     {
          Serial.print("Buenos ") ;
          Serial.print("Dias ") ;
          Serial.println("a todos.") ;
     }
```

C++ dispone de un tipo de variables llamadas Strings, capaces de contener textos. Podemos operar con ellas simplemente definiéndolas como cualquier otro tipo de C++:

```
 void loop()
      {
          int resultado = 25 ;
          String s = “ El resultado es: ” ; // Nótese que la S de string es mayúscula.
          Serial.print( s) ;
          Serial.println( resultado);
       }
```

Un tipo String se define simplemente poniendo entre comillas dobles un texto, y se puede operar con ellas de una forma similar a como operamos con enteros. Prueba:

```
void loop()
    {
        String a = "hola " ;
        String b = "a todos." ;
        Serial.println( a + b);
    }
```

Y también podemos construir un String sobre la marcha así:

```
void loop()
    {
        int resultado = 25 ;
        String s = "El resultado es: " ;
        Serial.println( s + String( resultado ));
    }
```

Donde imprimimos el resultado de concatenar s String, y la conversión de un int a String (El operador + añade un String al final de otro).

RECIBIENDO MENSAJES A TRAVÉS DEL PUERTO SERIE

.

Hasta ahora solo hemos enviado mensajes desde Arduino hacia el PC, ¿Pero como recibimos mensajes en Arduino?

En primer lugar disponemos de una función llamada Serial.parseInt() que nos entrega lo que se escribe en el monitor serie convertido a entero:

```
void loop()
     {
         if (Serial.available() > 0)
             {
                 int x = Serial.parseInt();
                 Serial.println ( x) ;
             }
     }
```

Este programa simplemente recibe en x los números que nos tecleen en la consola (cuando pulsemos intro) y si es un texto, lo interpreta como cero.

Hemos utilizado otra función de Serial : Available() que es un booleano. Conviene por costumbre comprobar que antes de leer el puerto serie hay

algo que nos han enviado. Si lo hay Available() es True y en caso contrario es False.

Para leer un String del puerto serie tenemos que complicarnos un poco más y hablar del tipo char.

Uno de de las mayores quebradero de cabeza al iniciarse en C++ es comprender la diferencia, anti-intuitiva, entre char y String. char es un tipo que representa un único carácter y se define con comillas simples, a diferencia de String que necesita comillas dobles:

```
char  c = 'a'  ;
String s ="a"  ;
```

Aunque parezca lo mismo para C++ son muy distintos.

Para leer una cadena desde el puerto serie necesitamos leer un carácter cada vez y después montar un String a partir de ellos, pero antes, asegúrate de seleccionar ambos NL & CR en la parte inferior del monitor serie, para garantizar que se envía el carácter de fin de línea:

Un programa para leer la consola sería algo así:

```
void setup()
       { Serial.begin(9600); }

void loop ()
     {
         char c = ' ' ;
         String mensaje ="" ;
         If (Serial.available()) //Comprobamos si hay algo esperando
              {
                 while( c != '\n') //Si lo hay, lo leemos hasta el intro
                     {
                       mensaje = mensaje + c ; // Añadimos lo leído al mensaje
                       c = Serial.read(); //Leer 1 carácter
                       delay(25);
                     }
                 Serial.println( mensaje); //Al salir imprimir el mensaje
                 mensaje = "" ; //Bórralo para la próxima vez
              }
      }
```

Aquí usamos otra instrucción de C++ llamada while. Es similar a if, Ejecuta repetidamente el bloque que le sigue mientras se cumpla la condición que le pasamos entre paréntesis:

```
while ( condición)
            {   …… }
```

Cuando lee el intro final de lo que escribimos, La condición c != '\n' se torna falso y sale del while.

Por lo demás, comprobamos si hay algo disponible en la puerta serie y de ser así montamos el mensaje leyendo un char cada vez y sumándoselo a mensaje para construir un String que podamos imprimir al salir.

- *El motivo del delay(25) es que a una velocidad tan lenta, enviar un char de 8 bits por la puerta serie, tarda mucho más de lo que tarda Arduino en ejecutar las instrucciones del while y volver a empezar. Por eso si se suprime el delay (y os recomiendo la prueba) leerá un carácter bueno (de la palabra escrita y como 10 caracteres basura para un Arduino UNO o Mega).*
- *Si subimos la velocidad de comunicación a 115200 bits por segundo, comprobareis que no hay este problema ya que al multiplicar la velocidad de envío por más de 10 Arduino ya no tiene tiempo de volver a por más caracteres antes de que lleguen.*

RESUMEN DE LA SESIÓN

.

- Hemos visto como establecer la comunicación con el PC externo, tanto para enviar como para recibir mensajes enteros y de texto.
- Hemos presentado los tipos String y char.
- Hemos visto las reglas básicas para operar con enteros y con Strings.
- Presentamos una nueva instrucción: while.

ANTERIOR **SIGUIENTE**

(34) COMMENTS

Alberto Palomero *15 Dic 2014*

Increible vuestro trabajo. Enorabuena!

Reply

Me gustaria saber si existe la posibilidad de obtener este tutorial de manera offline, es decir si puedo descargarlo de algun lugar

Gracias!

admin *15 Dic 2014*

Hola Alberto,
Gracias por tus palabras, siempre vienen bien los ánimos.
Me temo que no hemos planteado la descraga de los tutoriales, porque ni se nos había ocurrido, pero me imagino que puedes descrgar cada una de las sesiones, mediante tu navegador a un fichero o pagina HTML. Después deberías poder abrirla en local sin más problemas
Un saludo y hasta pronto

Reply

Miguel Flores *09 May 2015*

Excelente tutorial . He revisado y leído muchos relacionados con Arduino , pero ninguno con la simpleza y calidad de este. Te felicito.
iiiiiii

Reply

STARKMANN *24 Ago 2015*

Es aquí donde ya no se si sigo o me he perdido.
De todo lo que he ahora, entiendo que podemos comunicarnos entre pc y arduino en un sentido y en otro, aunque las funciones ya se me escapan.

Reply

admin *24 Ago 2015*

Tranqui Starkmann, es mas facil de lo que parece pero a veces hay que ir digiriendo las cosas para que se asientem. Ya veras como no tienes problema con la spuertas series

Reply

Jesús Sánchez *24 Ago 2015*

Hola Starkmann. Piensa en el monitor Serial como una ventana de comunicación con Arduino. La podemos usar para que nos “hable” o para hablar con él. Podemos hacer un sketch que lea temperatura, voltajes, o lo que sea, datos de otro Arduino y decirle que nos los muestre por el monitor Serial. Por contra, podemos hacer otro sketch con el que Arduino espera una orden nuestra dada a través de Serial, ya sea un carácter o un texto y ejecuta una instrucción, como enceder un LED o lanzar un cohete.

Reply

Saludos.

Victor *07 Nov 2015*

genial tus tutoriales, me etan sirviendo muchisimo. muchismas gracias¡¡

Reply

Reply

Manuel *10 Dic 2015*

Tengo un problema desde el primer ejemplo sesión, se compila bien, pero cuando la subo al arduino me da el siguiente error:
“avrdude: ser_open(): can’t set com-state for “\\.\COM1”
Error while setting port parameters: 9600 N 8 1

He comprobado que la velocidad me coincide en arduino y en el COM1 (9600)

Me podeis ayudar?
Garcias!

Reply

admin *11 Dic 2015*

Hola Manuel,
Aparentemente no puedes abrir el com1 para programar tu arduino y es bastante probable de que no tengas bien seleccionado el modelo o el puerto, porque normalmente el primer puerto que selecciona el IDE es el COM4 y no el COM1.
Revisa la configuracion del IDE: Modelo ed placa y puerto serie a ver que pasa y me dices

Reply

Manuel *11 Dic 2015*

He revisado el IDE de arduino y en el menú herramientas la placa es “Arduino/Genuino uno” y el puerto es COM1
Por otra parte entiendo que si puedo subir los programas desde la tarjeta al PC es porque el puerto serie me funciona. He hecho todas las sesiones anteriores a esta (establecer comunicaciones serie) y todo ha funcionado

Reply

admin *11 Dic 2015*

Tienes toda la razon Manuel, si subes el programa es que funciona la puerta serie correctamente.
¿Puede ser que tengas abierta la consola o mas de una instancia de arduino y por eso se bloquea la puerta com asignada?

Reply

Manuel *11 Dic 2015*

He cambiado manualmente en administrador de dispositivos en puertos a arduino a COM6 e reiniciado todo y ahora funciona.
No obstante, muchas gracias por tu tiempo.

Reply

Miguel *14 Ene 2016*

Buenas

Leyendo este tutorial se me ocurrió una escribir un programa, pero debo estar haciendo algo mal porque no funciona. Os cuento. Quiero hacer un programa que lea por teclado unos números, y que los guarde en diferente variables para luego trabajar con ellos. Sería algo como:

```
if(Serial.available())
{
int num1 = Serial.parseInt(); //Guardar el primero número, de varios dígitos
int num2 = Serial.parseInt(); //Guardar el segundo número, de varios dígitos
.....
}
```

Debo hacer algo mal porque el primero lo guarda bien, pero el segundo me lo pone a 0. He probado con delay() para que me dé tiempo a escribir pero no consigo nada. ¿Se os ocurre algo?

Saludos

Reply

admin *14 Ene 2016*

Hola Miguel, si tu programa es como en el esquema, nunca asignará el valor de la segunda variable, porque cunado hay algo disponible entra en el if y se lo asigan a num1, ya no queda nada y por tanto no creo que espere a que le teclees algo (Aunque no lo he probado) sospecho que se larga sin mas y la proxima vez que haya algo de nuevo va a num1

Reply

Miguel *15 Ene 2016*

He intentado cambiarlo un poco, y poner algo como

```
if(Serial.available())
int num1 = Serial.parseInt(); //Guardar el primero número, de varios dígitos

if(Serial.available())
int num2 = Serial.parseInt(); //Guardar el primero número, de varios dígitos
```

pero tampoco lo lee bien. Va alternando, y dependiendo de en qué parte del bucle esté el programa cuando pulso intro, lo guarda en num1 o en num2. Seguiré investigando por internet a ver qué encuentro.

admin *15 Ene 2016*

Reply

Hola Miguel prueba algo asi:
int num1 = 0 ; int num2 =0 ;
if(Serial.available())
num1 = Serial.parseInt(); //Guardar el primero número, de varios dígitos

if(Serial.available())
num2 = Serial.parseInt(); //Guardar el primero número, de varios dígitos

Serial.print(String(num1)+ "\t" + String (num2)+ "\n");
delay (500);

Si ahora introduces un par de numeros separados por comas, como 123,234 te asignará cada valor a una de las variables num1 y num2

Miguel *15 Ene 2016*

Gracias, lo probaré a ver qué tal

Guillermo *04 Feb 2016*

Reply

Estimado Admin.
Realmente percibo la excelente voluntad de enseñar la programación de Arduino, desde ya les agradezco todo lo que estoy aprendiendo (en realidad todavía estoy bajando los capítulos para luego con tiempo ir leyendolos) pero algo ya voy entendiendo (no mucho).
Y bueno recién empiezo con Arduino y me resulta bastante difícil cambiar la filosofía de las programaciones yo vengo de la industria de muchos años de programaciones de PLC y Visual Basic 5 y 6 pero aprieto los dientes y le meto para adelante.
En la comunicación de PC_ arduino _ visor LCD (16,2) tengo una rutina para mandar los datos desde VB6 y mostrarlos en el visor LCD, pero no logro que el mensaje cuando excede los primeros 16 caracteres pase al renglón inferior para completar el mismo.
Que estaré haciendo mal?
Un abrazo grande.Muchas gracias por la atención. Guillermo

admin *04 Feb 2016*

Reply

Se bienvenido Guillermo y ya verás como esto es mucha mas facil que los PLCs y sobre todo que Visual Basic, sobre todo porque el lenguaje de arduino es un sencillo C++ con librerias sin mas complicaciones.
Si nos mustras tu programa intentareos echarte una mano encantados

Un saludo

Diego Pedraza *20 Feb 2016*

Reply

Tengo una pregunta, el puerto serie de mi arduino me permite conectar varios módulos al mismo tiempo y obtener y enviar información de los mismos?, es decir, puedo conectar un modulo bluetooth y un modulo gps al mismo arduino y obtener información de los dos?

de antemano, muchas gracias por su respuesta.

admin *20 Feb 2016*

Reply

No es conveniente porque el puerto serie no tiene pines d econtrol que gestionen quien puede hablar en cada momento y por tanto acabarian chocando si mas de un dispositivo externo habla a la vez.
Pero por eso modelos Arduino con el Mega o el DUE tiene tres puertos serie independientes que puedes usar para controlar tres dispositivos diferentes serie

Jesús Sánchez *20 Feb 2016*

Reply

Hola a todos. ¿Y usando NewSoftSerial.h no sería posible? No la he probado, pero como puede "crear" otro puerto Serial…

Gracias y saludos.

admin *20 Feb 2016*

Reply

Hola Jesus, Yo lo he probado y va bastante bien siempre que no llegues a 115200 con un arduino UNO

Jesús Sánchez *20 Feb 2016*

Reply

Gracias Admin. Se me olvidó comentar que hay que tener cuidado al utilizarla y no enviar o recibir datos por los dos puertos al mismo tiempo, sino que se debe utilizar uno, y luego el otro, todo esto en la programación del sketch.

Gracias de nuevo.

Eugenio *27 Feb 2016*

Reply

Hola, y enhorabuena por tan buenas explicaciones de manera sencilla…y aquí ahora mi duda.
Siguiendo desde el primer tuto de encender un LED, ahora lo quiero hacer enviando u 0 o 1 desde la CONSOLA:

```
int LED = 13 ;
void setup()
{
pinMode( LED, OUTPUT) ;
Serial.begin(115200) ; // Inicializa el Puerto seria 115200 bits por segundo
}
void loop()
{
if (Serial.available())
{
digitalWrite(LED, Serial.parseInt()) ; // escribimos el valor leido por el puerto serie
}
}
```

Sencillo y funcional, pero si lo probáis veréis que entre que tecleas 1 y presionas ENTER, tarda alrededor de un segundo en encenderse el LED. o si tecleas 0, un segundo en apagarse.
Es como si tardara mucho el ARDUINO UNO en ejecutar la instrucción. En cambio cuando activo el RETORNO DE CARRO, o NUEVA LINEA o AMBOS NL & CR, y presiono 1 y ENTER, el cambio es instantáneo, pero claro, al momento me apaga otra vez el LED.
A qué se puede deber?
Gracias!

admin *29 Feb 2016*

Reply

No me atrevo a responderte sin probar el circuito, pero si que me ha pasado que a veces el parseint es un poco lento para recibir el valor sin que tenga claro porqué

Miguel Angel *06 Mar 2016*

Reply

Hola a todos;
Yo me he perdido ya en la sesion 7, no entiendo nada de lo que pones en el último programa de la sesión;

```
{
char c =' ';
String mensaje = "";
if (Serial.available ());//Comprobamos si hay algo esperado

{
while( c!= '\n') //si lo hay lo leemos hasta el intro
{
mensaje=mensaje+c;//añadimos lo leido al mensaje
c=Serial.read ();//lee un caracter
delay(25);

}

Serial.println(mensaje);//Al salir imprimir el meensaje
mensaje = "" ; //borrarlo para la proxima v
}

}
```

Cuando escribo esto en el programa y escribo algo en el monitor, me salen muchas "ÿ" seguidas de la palabra escrita, eso es normal o solo debe salir lo que escribo?
No se pero me estoy agobiando porque yo hasta ahora intento comprender todas las secuencias del programa para saber que estoy haciendo, pero aqui me pierdo.

admin *06 Mar 2016*

Reply

Tranqui Miguel Angel, a todos nos ha pasado al principio pero no hay nada dificil que no puedas entender. simplemente vamos sumando conceptos nuevos y al final siempre hay alguno que se atraviesa, pero tranqui, insiste y veras como no hay secretos.
Los chinitos que te aslen en el monitor suelen ser causados por una diferencia de velocidad entre la que has programado en tu arduino y la del monitor serie. comprueba que ambos estan a la misma velocidad y una vez que recibas datos, dinoslo y vemos que mas necesitas ¿Vale?

Miguel Angel *09 Mar 2016*

Reply

Hola ;
He comprobado la velocidad tanto del programa y el monitor y es el mismo, y me siguen saliendos los chinitos como dices ¿Que hago?

Jose Rodriguez *12 Mar 2016*

escribiendo esto me da este error:

Reply

```
void setup()
{
Serial.begin(9600) ; // Inicializa el Puerto serie
}
void loop()
{
int resultado = 25 ;
String s = " El resultado es: " ; // Nótese que la S de string es mayúscula.
Serial.print( s) ;
Serial.println( resultado);
}
```

Arduino:1.6.6 (Windows 7), Placa:"Arduino/Genuino Uno"

comunicacion_serie:9: error: stray '\342' in program

String s = â€œ El resultado es: â€? ; // NÃ³tese que la S de string es mayÃºscula.

^

comunicacion_serie:9: error: stray '\200' in program

comunicacion_serie:9: error: stray '\234' in program

comunicacion_serie:9: error: stray '\342' in program

comunicacion_serie:9: error: stray '\200' in program

comunicacion_serie:9: error: stray '\235' in program

C:\Users\PC CASA\Documents\Arduino\comunicacion_serie\comunicacion_serie.ino: In function 'void loop()':

comunicacion_serie:9: error: 'El' was not declared in this scope

String s = â€œ El resultado es: â€? ; // NÃ³tese que la S de string es mayÃºscula.

^

exit status 1
stray '\342' in program

Este informe podría tener más información con
"Mostrar salida detallada durante la compilación"
activala desde Archivo > Preferencias

admin *12 Mar 2016*

Hola Jose, este error es similar a otro que nos enviaron hace un par de dias. El problema no está en el codigo sino en una mala eleccion de las comillas que delimitan el string: No son las comillas de encima del 2 sino otras (que no se de donde sacais) prueba a cambiarlas

Reply

Jose Rodriguez *12 Mar 2016*

el texto es copiado directamente de la web, pero efectivamente si lo escribo yo directamente no me da ese problema

Reply

Miguel Angel *12 Mar 2016*

Ya he solucionado lo de los chinitos por un simple ; puesto de más me salían los chinitos y me iba a volver loco.
Bueno otra pregunta? que significa ('\n') no se interpretarlo, al igual cuando pones char c = ' ' eso como lo puedo interpretar para que lo comprenda.

Reply

admin *12 Mar 2016*

Hola Miguel Angel, EL caracter "\n" equivale a intro. Hay varios codigos de estos en la tabla de carcateres de C++. ("\t" es tabulador)

Reply

GIVE A REPLY

Conectado como Javier. ¿Quieres salir?

Message

Post comment

Sí, agrégame a tu lista de correos.

CATEGORIAS DE LOS PRODUCTOS

Selecciona una categoría