MÓDULO 4: PROJECTO 14 "Retocar el logotipo de Arduino"

CURSO PROGRAMACIÓN DE PLACAS ROBÓTICAS

Introducción

- Ha llegado el momento de controlar nuestro pc mediante Arduino. Cuando programamos nuestro Arduino, estamos abriendo una conexión entre el pc y el microcontrolador. Podemos utilizar dicho conexión para enviar y recibir información a otras aplicaciones.

- Arduino tiene un chip que convierte la comunicación basada en usb que utiliza el pc, en la comunicación serie que utiliza Arduino. La comunicación en serie consiste en que dos computadoras, nuestro Arduino y el pc, están intercambiando bits de información en serie, o dicho de otro modo, un bit detrás de otro.

- Cuando se comunican en serie, las computadoras necesitan decirse la una a la otra a que velocidad se están hablando. Seguramente os habréis percatado que cuando usamos el monitor serie hay un número en la esquina inferior derecha de la ventana. Este número, 9600 bits por segundo, o baudios, es el mismo valor que declaramos al utilizar la función **Serial.begin()**. Esta es la velocidad a la que Arduino y el pc intercambian datos.

- Hemos usado el monitor serie para ver los valores provenientes de las entradas analógicas; nosotros utilizaremos un método similar para obtener valores en un programa que escribiremos en un entorno de programación llamado **Processing**. Processing está basado en Java y el entorno de desarrollo de Arduino está basado en Processing. Son muy parecidos, así que no nos será muy extraño trabajar con el.

Introducción

- Antes de empezar con este proyecto, descargaremos la última versión de Processing desde processing.org/download. Nos será muy útil echar un vistazo a la guías "Getting started" y "Overview". Ellas nos ayudarán a familiarizarnos con Processing antes de que empecemos a escribir software para comunicarnos con nuestro Arduino.

- La forma más eficiente de enviar datos entre Arduino y Processing es utilizando la función **Serial.write()** en Arduino. Es similar a la función **Serial.print()** utilizada con anterioridad en el sentido que también envía información al pc conectado, pero que en lugar de enviar información legible para los humanos como número y letras, envía valores comprendidos entre el 0 y 255 como un flujo puro de bytes. Esto limita los valores que pueden ser enviados por Arduino, pero permite una rápida transmisión de información.

- Tanto en nuestro pc como en Arduino, existe algo llamado buffer serie, el cual, retiene la información hasta que sea leída por el programa. Nosotros enviaremos bytes desde el Arduino al buffer serie de Processing. Entonces Processing sacará dichos bytes del buffer para poder leerlos. Una vez que el programa haya leído la información del buffer, dejará el espacio libre para almacenar más información.

Introducción

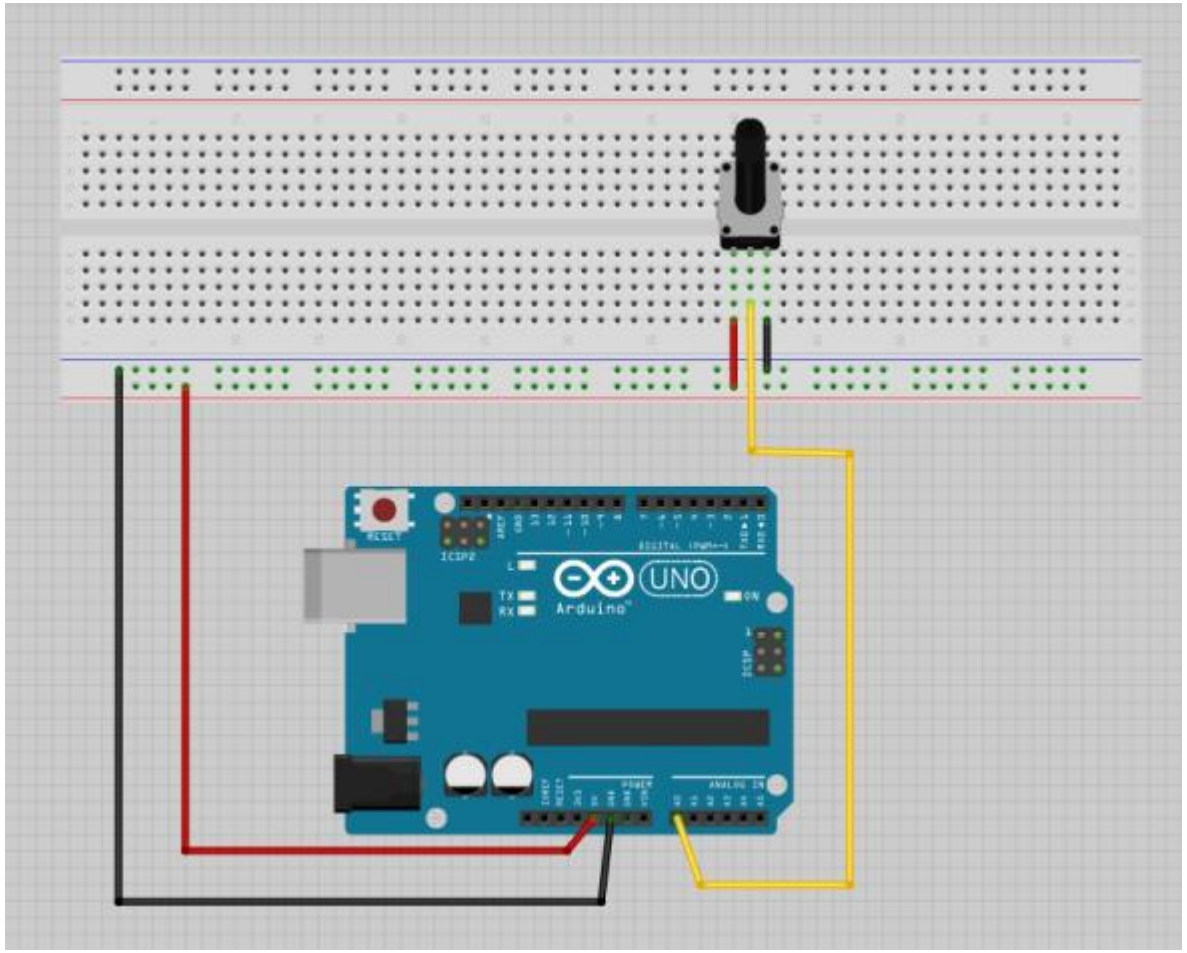
- Cuando utilizamos la comunicación en serie entre dispositivos y programas, no solo es importante que ambas partes conozcan la velocidad de la comunicación, si no que también deben conocer que tipo de información están esperando. Cuando conocemos a alguien, probablemente esperemos un “¡Hola!”, si en lugar de eso la otra persona dice algo como “El gato está bufado”, seguramente pensaremos que está un poco loco. Con el software, tenemos que saber de ambas partes que tipo de información se ha enviado y cual se ha recibido, o lo que es lo mismo, asegurarnos de que el tipo de información que hemos enviado es del mismo tipo que la que hemos recibido.

Montando el circuito

1. Conectamos la alimentación y la tierra a la protoboard.
2. Unimos la patilla izquierda del potenciómetro a la alimentación, la derecha a tierra y la central al pin analógico 0.

Montando el circuito

Componentes	
Potenciómetro	1



Comentario del código

- En este proyecto vamos a tener dos tipos de código fuente, el de Arduino y el de Processing. Primeramente veremos como es el del Arduino ya que es más corto y estamos más familiarizados con el. Después veremos la parte correspondiente a Processing.

- Primeramente, en la función **setup()**, iniciaremos la comunicación serie. El programa que hagamos posteriormente con Processing, deberá tener el mismo ratio de comunicación serie que nuestro Arduino.

- En la función **loop()**, usaremos el comando **Serial.write()** para enviar información a través de la conexión serie. **Serial.write()** solo puede enviar valores comprendidos entre 0 y 255. Para asegurarnos que los datos que enviamos se encuentran dentro de dicho rango, dividiremos el valor analógico leído entre 4.

Comentario del codigo

- Tras haber enviado un byte, esperaremos un milisegundo para que el ADC (Analog to Digital Conversion) se haya establecido. Finalmente, subiremos nuestro programa al Arduino y lo dejaremos a un lado para empezar a escribir el programa con Processing.

El código de Processing

- El lenguaje Processing es parecido a Arduino, pero tiene suficientes diferencias como para tener que echarle un vistazo a los tutoriales de iniciación mencionados anteriormente y poder así familiarizarnos con él.
- Abrimos un nuevo boceto de Processing. A diferencia de Arduino, Processing no sabe nada acerca de puertos serie a no ser que incluyamos una librería externa. El siguiente paso será importar la librería serie.

Comentario del codigo

El código de Processing

- Necesitamos crear una instancia del objeto de la librería, tal y como hemos hecho en capítulos anteriores con la librería del servo motor. Este objeto de nombre único será el que utilicemos cada vez que queramos recurrir a la conexión serie.
- A continuación, si queremos usar imágenes en Processing, necesitamos crear un objeto que almacene dicha imagen y tenga un nombre.
- Seguidamente, crearemos una variable que almacene la tonalidad de fondo del logo de Arduino. El logo es un archivo con extensión .png y está hecho sobre una transparencia, por lo que podemos ver los cambios en el color de fondo.

Comentario del codigo

El código de Processing

- Al igual que Arduino, Processing también tiene una función **setup()**. Aquí será donde abriremos la conexión serie y le daremos al programa toda una colección de parámetros que usará mientras se ejecuta.
- Podemos cambiar la manera en que Processing trabaja con los colores. Normalmente trabaja con una paleta de colores **RGB** (Red, Green, Blue). En este programa sin embargo, vamos a utilizar una paleta de colores llamada **HSB** (Hue, Saturation, Brightness), en donde cambiaremos el fondo (Hue en inglés) al hacer girar el potenciómetro.

Comentario del codigo

El código de Processing

- La función **colorMode()** tiene dos argumentos: el tipo de modo y el valor máximo que puede esperar.
- Para cargar la imagen en el boceto, tenemos que leerla mediante el objeto llamado logo que hemos creado anteriormente. Cuando proporcionamos la url de una imagen, Processing la descargará cuando ejecutemos el programa.
- Con la función **size()**, le estamos como de grande ha de ser la ventana. Si utilizamos **logo.width** y **logo.height** como argumentos, la ventana del boceto se escalará automáticamente al tamaño de la imagen que estamos utilizando.

Comentario del codigo

El código de Processing

- Processing tiene la habilidad de mostrar mensajes de estado utilizando el comando **println()**. Si utilizamos esto en conjunto con la función **Serial.list()**, obtendremos una lista de todos los puertos serie que nuestro pc tiene disponibles al empezar el programa por vez primera.
- Utilizaremos esto una vez que hayamos terminado de programar para ver en que puerto se encuentra nuestro Arduino.
- A continuación, tenemos que darle a Processing la información necesaria sobre la conexión serie.
- Para poder pasar los datos necesarios a nuestro objeto serie **myPort**, el programa necesita saber que dicho objeto es una nueva instancia del objeto serie.

Comentario del codigo

El código de Processing

- Los parámetros que esperará son con que aplicación va a comunicarse, a través de que puerto va a realizar dicha comunicación y a que velocidad.
- El atributo **this** le dice a Processing que vamos a utilizar la conexión serie en esa aplicación específica. El argumento **Serial.list()[0]**, especifica que puerto serie estamos utilizando. **Serial.list()** contiene un array de todos los dispositivos serie conectados. El argumento **9600** nos tiene que resultar familiar, ya que define la velocidad a la que el programa va a comunicarse.
- La función **draw()** es la equivalente a la loop() de Arduino, en el sentido de que es un bucle que se ejecuta una y otra vez sin parar. Es en esta función donde se dibujan los objetos en la ventana del programa.

Comentario del codigo

El código de Processing

- Lo que vamos a hacer ahora es comprobar si hemos recibido información desde el Arduino. El comando **myPort.available()** nos dirá si hay algo en el buffer serie. Si hay bytes, los guardaremos en la variable **bgcolor** y los mostraremos por la pantalla de debug.
- La función **background()** establece el color de la ventana. Requiere tres argumentos. El primero es la tonalidad (hue en inglés), el segundo el brillo (brightness) y el último la saturación (saturation). Nosotros utilizaremos el contenido de la variable bgcolor para el parámetro de tonalidad, pero el brillo y la saturación la estableceremos a su valor máximo, 255.

Comentario del codigo

El código de Processing

- Dibujaremos el logo con el comando **image()**. Para ello, necesitamos decirle que dibujar y cuales son las coordenadas de inicio para empezar a dibujar en la ventana. Nosotros empezaremos con las coordenadas 0,0, que indican la esquina superior izquierda.