MÓDULO 4: PROJECTO 6 " Theremin Optico"

CURSO PROGRAMACIÓN DE PLACAS ROBÓTICAS

Introducción

- El Theremin detecta la posición de las manos del músico en relación a dos antenas midiendo las variaciones en la capacidad eléctrica de las mismas. Las antenas están conectadas a un circuito analógico el cual, se encarga de generar el sonido. Una antena controla la frecuencia del sonido y otra el volumen del mismo.

- Aunque Arduino no puede replicar exactamente el misterioso sonido de este instrumento, si que le es posible emularlo mediante el uso de la función **tone()**. La figura siguiente nos muestra la diferencia entre los pulsos emitidos por la función **analogWrite()** y la función **tone()**.

Introducción

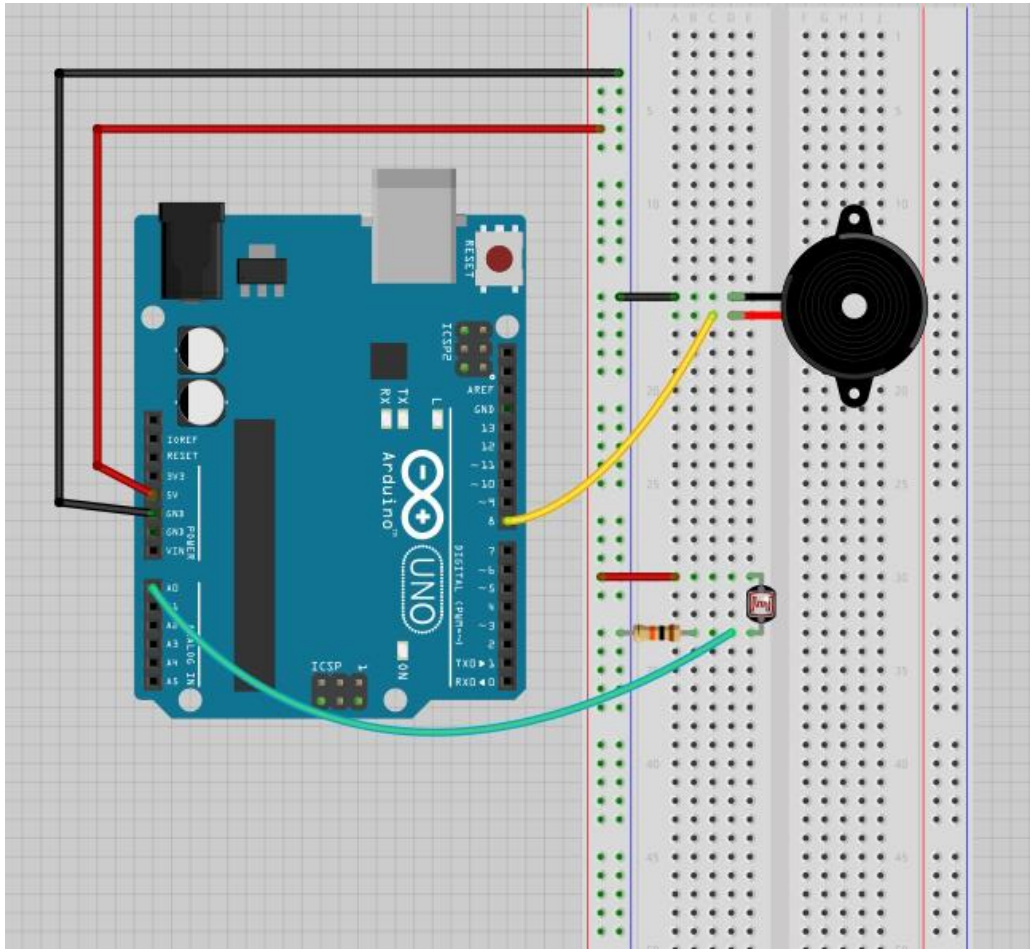
- Moviendo nuestras manos sobre el sensor, lograremos alterar la cantidad de luz que incide sobre la fotorresistencia. El cambio en la tensión del pin analógico, determinará que frecuencia de nota musical deberá reproducir.
- Obtenemos datos utilizando la función **analogRead()**, las lecturas no se encuentran siempre comprendidas entre 0 y 1023.
- La resistencia conectada a tierra limita el valor mínimo del rango y el brillo de la luz ambiental establece el valor máximo.
- En lugar de fijar un rango limitado, calibraremos los datos **High** y **Low** recogidos por el sensor, transformándolos en sonido mediante la función **map()** para de esta manera, obtener un rango de salida tan amplio como sea posible.
  - Esto tiene la ventaja añadida de poder ajustar las lecturas del sensor cada vez que movamos el circuito a un nuevo entorno, como por ejemplo, una habitación con diferentes condiciones lumínicas.

Montando el circuito

1. Conectamos los cables de alimentación a nuestra protoboard.
2. Cogemos nuestro pequeño altavoz y conectamos una de sus patillas a tierra y la otra al pin digital 8 de nuestro Arduino.
3. Colocamos la fotorresistencia en la protoboard con una de sus patillas conectada a la alimentación de 5V. Seguidamente, conectamos la otra patilla de la fotorresistencia al pin analógico 0 y a tierra a través de la resistencia de 10KΩ.

Montando el circuito

Componentes	
Fotorresisténcias LDR	1
Altavoz piezoeléctrico	1
Resistencia 10 K Ω	1



Comentario del código

- Crearemos una variable para almacenar el valor que **analogRead()** ha recibido de la fotorresistencia (**sensorValue**).
- Crearemos dos variables más, una para el valor High (**sensorHigh**) y otra para el Low (**sensorLow**).
- Inicializaremos el valor de la variable **sensorLow** a 1023 y a 0 el de la **sensorHigh**, de esta manera, sabremos a ciencia cierta cuales son los valores máximos y mínimos reales.
- A continuación, creamos una constante llamada **ledPin**. Esta constante la utilizaremos como indicador de que el sensor ha acabado de calibrarse. En este proyecto utilizaremos el led de la placa que esta conectado al pin 13.

Comentario del código

- En la función **setup()**, estableceremos el pin 13 como salida mediante la función **pinMode()** y lo encenderemos.

- Calibraremos los valores máximos y mínimos del sensor. Utilizaremos la sentencia **while()** para ejecutar un bucle durante cinco segundos. Dicho bucle se ejecutará hasta que se cumple una determinada condición. En nuestro caso, haremos uso de la función **millis()** para comprobar el tiempo actual. Esta función nos dirá cuanto tiempo ha transcurrido desde que el Arduino ha sido encendido o reseteado por última vez.

- Dentro del bucle, vamos a leer el valor que nos da el sensor. Si dicho valor es menor que el contenido en la variable **sensorLow** (inicialmente era 1023), entonces actualizaremos el contenido de dicha variable con el valor obtenido.

Comentario del código

- Si por el contrario, el valor obtenido es mayor que el contenido en la variable **sensorHigh** (inicialmente era 0), entonces actualizaremos el contenido de dicha variable con el nuevo valor.
- Transcurridos cinco segundos, el bucle while finalizará, por lo que apagaremos el led asociado al pin 13. Utilizaremos los valores **High** y **Low** obtenidos del sensor para escalar la frecuencia que será utilizada en la parte principal del programa.
- A continuación, entraremos en la función **loop()**, para leer el valor del pin A0 y almacenarlo en la variable **sensorValue**.

Comentario del código

- Creamos una variable llamada **pitch**, cuyo valor es el resultado de mapear el contenido de la variable **sensorValue**. Usaremos los valores de **sensorLow** y **sensorHigh** como límites de los nuevos datos recogidos. Como valores de salida iniciales, probaremos con un rango de entre 50 y 4000.

- Llamaremos a la función **tone()** para que reproduzca el sonido. Contiene tres argumentos:
  - El pin en el que reproducirá el sonido (en nuestro caso será el pin 8).
  - La frecuencia a reproducir.
  - El tiempo de reproducción de la nota

Finalmente, utilizaremos la función **delay()** para obtener un retardo de 10 milisegundos y que el sonido tenga tiempo suficiente para ser reproducido..