Situación de aprendizaje 9: Semáforo adaptado

Programación con Arduino y posterior prototipado de un semáforo que incluya una señal auditiva cuando el semáforo esté en rojo (verde para peatones), para adaptarlo a personas con deficiencia visual.

semaforo.jpg

COMPONENTES	**Arduino Uno** **ArduinoUno.png**
	Tira leds NeoPixel **TIraNeopixel.jpg**
	Zumbador **rZumbadorrecortado.png**
	Conectores 5v y Gnd **Recibidos (442) - raguado@juandelanuza.org - C**

Trabajo de indagación: En cualquier proyecto debemos investigar sobre el funcionamiento y comportamiento esperado para nuestro prototipo. En este caso, sabemos que los semáforos adaptados emiten un tono discontinuo cuando los peatones pueden cruzar, posibilitando de este modo a personas con dificultades de visión el paso de un lado a otro de forma segura. Los semáforos mediante código de colores (rojo, ámbar y verde), indican a los conductores/peatones si es momento de esperar o avanzar. La secuencia que siguen los semáforos de conductores es: **rojo**, **verde** y **ámbar**. Cuando esta rojo para

coches, el de peatones esta en verde y al contrario. En el caso que nos toca, tendremos entonces que contemplar emitir los tonos, cuando el semáforo de coches este en **rojo**.

Para todos nuestros prototipos, vamos a utilizar una **herramienta** de **diseño 2D** con la que representaremos nuestro esquema de conexiones.

Diseño 2D

Es conveniente utilizar alguna herramienta para diseñar nuestro circuito antes de proceder al montaje. De este modo evitaremos cometer errores con el cableado. Existen muchas herramientas online para este propósito y algunas de ellas como Tinkercad incluso permiten su simulación, pero tienen mayor limitación en las librerías de componentes, por lo que en este caso utilizaremos EasyEDA.

> Puedes encontrar un tutorial para crear tus diseños con EasyEDA pinchando aquí.

EasyEDA(Standard) - A Simple and Powerful Electronic Circuit Design Tool.png

Como podemos observar en nuestro esquemático, vamos a conectar dos actuadores a nuestro Arduino. Ambos tienen un pin a **GND** y otro a **5V**, y cada uno de ellos tiene un pin de datos que asignaremos a los **pines 2 y 3**.

La placa de Arduino tiene 3 pines conectados a GND y uno a 5v, por lo que haremos uso del adaptador incluido en el kit para la conexión a 5v o en su defecto se podría utilizar una placa de prototipado.

Para abordar la programación de este proyecto, en primer lugar **inicializaremos** los componentes y **crearemos 4 funciones**: "*encender verde*"," *encender ámbar*"," *encender rojo*" *y* "*sonido*". De este modo, será tan sencillo como llamar a cada una de ellas en **el bucle principal**, las cuales se repetirán indefinidamente.

Construcción

Arduino tiene únicamente 1 pin a 5v y 3 a Gnd.

En esta práctica, utilizamos el Zumbador que tiene 3 pines (5v, Gnd y Señal) y la tira led Neopixel que igualmente tiene otros 3 pines (5v, Gnd y datos).

Para alimentar a 5v ambos actuadores, necesitaremos utilizar el conector multiplicador (1 a 4). El pin de datos de la tira led irá según nuestra programación al Pin2 de Arduino y el del zumbador al Pin3 de Arduino. El Gnd del zumbador y el de la tira led, los conectaremos a cualquiera de los 3 pines habilitados para ello en Arduino sin necesidad de utilizar el multiplicador.

20221119_074429.jpg

Programación - Inicialización de componentes

Representamos lo primero de todo un diagrama de flujo que nos ayude a comprender el comportamiento de nuestro proyecto:

Bitbloq Robotics - Documento sin título.png

El siguiente paso, una vez hemos diseñado nuestro circuito es programarlo. Para ello abrimos el editor web de ArduinoBlocks y nos conectamos con nuestro usuario. Partiendo de un proyecto en blanco eligiendo como placa **Arduino Uno**, lo primero que haremos será sacar en pantalla nuestros componentes.

En primer lugar arrastraremos al área de trabajo el Zumbador, que lo encontraremos dentro del bloque de actuadores.

ArduinoBlocks.png

y haremos lo mismo con la tira led de NeoPixel, que la encontraremos en Visualización, NeoPixel.

ArduinoBlocks (1).png

Con esto ya tenemos todos los componentes que intervienen en nuestro circuito.

Ahora debemos asignarle a cada uno de ellos el Pin al que, en el paso anterior, hemos decidido que debe ir conectado. Para el **Zumbador** tendremos que modificarlo al **Pin3** y deberemos asegurarnos que la Inicialización de la tira **Led** la hacemos sobre el **Pin2**. Además, podemos observar que la tira Led tiene otros 3 parámetros a los que debemos prestar atención. El primero de ellos es la codificación de colores que tiene nuestro componente. En nuestro caso "GRB" (green, red, blue). El segundo de ellos es la frecuencia, que fijaremos en 800Khz, y el

tercero, indica el número de leds que compone la tira. En nuestro caso 8, pero debemos tener en cuenta que la posición del primer led se cuenta como 0, por tanto deberemos indicar 7.

Todas las opciones que podemos realizar con la tira de leds, las podemos encontrar al seleccionar en "Visualización" el apartado "NeoPixel": Las que vamos a utilizar son: *iniciar, establecer brillo, limpiar, mostrar y establecer pixel # a un determinado color.*

- ***Establecer brillo*** **-** Como su nombre indica, haciendo uso de este parámetro vamos a poder determinar la intensidad del brillo de los leds. Su valor puede ir de 0 a 255, siendo 0 la intensidad mínima (apagado) y 255 la máxima. A mayor intensidad de brillo, mayor demanda de corriente tendrá nuestro circuito, por lo que este valor es muy importante a la hora de elegir el método de alimentación de la tira NeoPixel. Cada Led consume unos 60mA (0,3W), dando color blanco intenso (20mA por cada componente de color) Esto supone un consumo de 9W para 30 LED, y 18W para 60 LED, lo que es mucha potencia en una fuente de 5V. Si nuestra tira demanda más potencia que la que puede obtener del propio Arduino, estaremos obligados a alimentar los leds con una fuente externa. Para no tener que recurrir a una fuente externa, en nuestro ejercicio, fijaremos la intensidad del brillo en 10.
- ***Establecer pixel # " " color*** **-** Gracias a este componente podemos determinar, teniendo en cuenta que el primer led ocupa la posición 0, de que color queremos que se ilumine un determinado Led. Si quiero que el led 2 de mi tira de NeoPixel se encienda de color rojo, bastará con configurarlo como: *Establecer pixel # 1 color rojo.*
- ***Mostrar*** - Cualquier configuarción que haga, no será visible hasta que la muestre. Este componente lo utilizaremos cada vez que queramos mostrar un cambio. Para que el Pin 2 del apartado anterior se muestre en color rojo, tendremos que utilizar el bloque "mostrar".
- ***Limpiar*** - Utilizaremos este componente cada vez que queramos eliminar cualquier configuración anterior. Sí quiero que el led 2 de mi tira que anteriormente lo hemos fijado a color rojo se apague, utilizaré este bloque.

Todos estos valores, los actualizaremos y pondremos en el bloque de "Inicializar", para que de este modo, **Arduino** sepa que en el **Pin2** va a tener conectada una tira **NeoPixel** de 8 leds.

ArduinoBlocks (5).png

Para crear funciones, debemos abrir el menú de funciones y arrastrar el bloque al área de trabajo:

ArduinoBlocks (5).png

Programación - Funciones

Vemos el código para las funciones. Todas se van a comportar igual salvo la diferencia de los colores, los leds que encendemos y el tiempo de espera para cada color. Definimos los **pines 1 y 2 para** encender en color **verde**. Los **pines 3 y 4 para** el color **ámbar** y por último, los **pines 5 y 6 para** el color **rojo**. El tiempo que permanece la luz de color roja y verde lo definimos en 10 segundos, (10.000 milisegundos) y el de la luz en ámbar lo fijamos en 3 segundos (3.000 milisegundos).

Hay que tener en cuenta que la posición del vector de Leds para BitBlock comienza en 0. Por tanto, el primer led ocupa la posición 0, el segundo la posición 1, y así sucesivamente.

Como nuestra tira tiene 8 pines, en nuestro ejercicio no utilizaremos ni el primero ni el último.

encender verde: *Esablecemos pixeles de las posiciones 2 y 3 a color verde, los mostramos y lo mantenemos así durante 10 segundos.*

ArduinoBlocks (6).png

encender ámbar : *Esablecemos pixeles de las posiciones 3 y 4 a color ámbar, los mostramos y lo mantenemos así durante 3 segundos.*

ArduinoBlocks (7).png

encender rojo: *Esablecemos pixeles de las posiciones 5 y 6 a color rojo, los mostramos y lo mantenemos así durante 10 segundos.*

ArduinoBlocks (8).png

sonar zumbador: *Teniendo el zumbador conectado al Pin3 de Arduino, emitimos un sonido con frecuencia de 500Hz durante 1 segundo y al finalizar, esperamos 1 segundo.*

ArduinoBlocks (10).png

Teniendo ya definidas nuestras funciones, tan solo nos quedaría incluirlas en el Bucle principal para que se repitan indefinidamente, pero para cumplir con las especificaciones del programa, debemos realizar algunas modificaciones para que mientras el semáforo esta en rojo se escuche el sonido del zumbador. Por tanto, tenemos que hacer un llamamiento a la función "*sonar zumbador*" desde

la función "*encender rojo*". Si observamos bien, para mantener las luces rojas, hemos tenido que utilizar una espera de 10 segundos, que ahora podremos sustituir por la espera que se produce cada vez que se produce un sonido por el zumbador (suena 1 segundo y espera en silencio otro segundo). En concreto, cada vez que llamamos a la función del zumbador, transcurren 2 segundos, por lo que si la repetimos 5 veces, estaremos empleando 10 segundos que son los que queremos que la luz roja permanezca encendida mientras escuchamos un "beep" de manera intermitente. La función modificada quedaría de la siguiente manera:

encender rojo actualizada

ArduinoBlocks (11).png

Con esta actualización ya estariamos cumpliendo los requisitos del enunciado, y nuestro programa finalmente quedaría del siguiente modo:

ArduinoBlocks (6).png

https://www.youtube.com/embed/hXE8xaI2WKQ

Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

logo.png

Revision #24
Created 2022-10-15 09:13:34 CEST by Ricardo Aguado Vallejo
Updated 2023-01-17 15:53:19 CET by Equipo CATEDU