PROGRAMACIÓN MULTIMEDIA
Y
DISPOSITIVOS MÓVILES

ACTIVIDAD EVALUABLE

MEMORIA

Pablo Giner Barrios

ÍNDICE

DESCRIPCIÓN DE LA APP

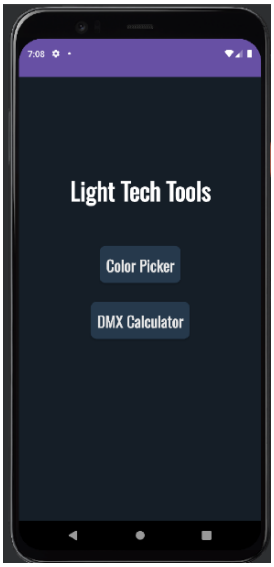
La aplicación es un compendio de herramientas que un técnico de iluminación necesita en su día a día laboral. La primera se trata de un selector de color para poder ver el código hexadecimal de un color concreto seleccionado y poder aplicarlo en algunos softwares de iluminación. La segunda es un selector de canales que permite ver la configuración de un DIP Switch para configurar el canal de algún aparato DMX o saber en qué canal DMX está configurado dicho aparato.

Consta de tres pantallas. La primera es un menú en el que puedes seleccionar la herramienta deseada (el Color Picker o el Selector de canales para de DIP Switch). Las otras dos corresponden a las mencionadas herramientas.

GUÍA DE USUARIO

MENÚ

Al iniciar la aplicación aparecerá la vista de un menú con dos opciones: Color Picker o DMX Calculator. Se habrá de escoger el que se quiera usar.



COLOR PICKER

En la parte superior de la pantalla se verá una imagen circular con toda la posible selección de tonos. El centro corresponde con el blanco y, al acercarse al borde, se incrementa la saturación de los diferentes tonos de color.

Bajo esta imagen, se muestras dos barras horizontales:

La primera (canal alpha) que controla la opacidad del color. Si se desplaza el selector hacia la izquierda, incrementará la transparencia del color hasta llegar a un tono completamente transparente. Si desplazamos a la derecha, incrementaremos su opacidad hasta llegar a tener un tono completamente opaco.

La segunda (brillo) controla el brillo del tono seleccionado. Si se desplaza a la izquierda, el brillo disminuye hasta llegar al negro. Si se desplaza hacia la derecha, el brillo del color aumenta hasta llegar a una saturación máxima (255).

La combinación de la selección de tono, opacidad y brillo dan como resultado el color que se muestra en el cuadrado inferior, cuyo código hexadecimal se muestra debajo precedido por el símbolo #.

DMX CALCULATOR

Esta herramienta está dividida en dos partes. En la zona superior podemos ver un selector de canales (Channel Selector) en el que podemos ver el canal seleccionado en base decimal. Consta de 3 cifras (unidades, decenas y centenas de derecha a izquierda). Cada cifra consta de dos botones: uno superior para incrementar en 1 la cifra y otro inferior para disminuir en 1 la cifra.

El canal máximo seleccionable es 511 y el mínimo es 000, con lo que conseguimos los 512 canales disponibles en el protocolo DMX. Al modificar el canal cambiará la posición de los pines del DIP Switch de la zona inferior.

En la parte inferior podemos ver la conversión del canal seleccionado al modelo DIP Switch, modo de selección de canal inicial para muchos aparatos de espectáculo tales como máquinas de humo o focos led. Si se pulsan los pines del DIP Switch, podremos ver cómo el canal del

Channel Selector varía, de modo que podemos ver en qué canal está configurado un aparato concreto a partir de la posición de los pines en su DIP Switch.

JUSTIFICACIÓN DEL CÓDIGO

La aplicación consiste en 3 *activities* con sus correspondientes *layouts*:

MAIN ACTIVITY

La llamada **MainActivity** corresponde con la pantalla del menú. Al inicio he declarado dos variables que se inicializarán más tarde correspondientes a las **CardView** declaradas en el *layout* y que se corresponden con los botones que nos llevarán a las otras dos pantallas.

Dentro de la función **onCreate**, cargo el archivo de *layout activity_main* e invoco a las funciones **initComponents()** e **initListeners()**. Estas dos funciones serán una constante en todas las pantallas, puesto que, de este modo, queda mucho más ordenado el código.

La primera, enlaza las variables *lateinit* declaradas al principio con sus respectivos elementos visuales del código xml del *layout*. La segunda, inicia las funciones *listener*, que, en este caso, son **setOnClickListener()**, y lo que hacen es iniciar la vista correspondiente al botón pulsado.

COLORPICKER

Para esta *activity* importé una librería externa. Para ello, en el archivo **build.gradle.kts(Module:app)**, dentro de las *dependencies*, añadí esta línea:

***implementation* ("com.github.skydoves:colorpickerview:2.3.0")**

También añadí, dentro del archivo **settings.gradle.kts (Project Settings)**, dentro de *repositories*, las siguientes líneas:

```
maven {
    url = uri("https://oss.sonatype.org/content/repositories/snapshots/")
}
```

Una vez hecho esto, en la propia *activity* llamada ColorPicker, como en la anterior, declaro las variables *lateinit* correspondientes a los elementos del *layout* y los inicializo en la función **initComponents()** que invoco desde la función **onCreate**.

En el método **initListener()**, invocado también desde el **onCreate**, utiliza la variable **colorpickerView** para usar el método **setColorListener()**, proporcionado por la librería importada y crear un **ColorEnvelopelistener** y permite recoger el color seleccionado dentro de la rueda de color y colocarlo como fondo en el cuadro inferior y "setear" el código del color en el elemento de texto de la vista.

Desde **onCreate** declaro e inicializo dos variables inalterables (val) que corresponden con las barras horizontales de brillo y canal Alpha. Estos elementos me los proporciona la librería importada.

A continuación, uso los métodos **attachBrightnessSlider()** y **attachAlphaSlider()** con sus correspondientes variables para que, el color seleccionado en el *color picker* sea recogido por las barras y modificado (la primera permite modifica el brillo del tono seleccionado y que viene desde el método **setColorListener()**y la segunda, modificar el canal Alpha de este).

DMX CALCULATOR

Como en las anteriores, inicio las variables *lateinit* en el declaro las variables al inicio de la clase además de lista vacía **pinButtons** de elementos del tipo **LinearLayoutCompat** y las enlazo con sus respectivos elementos del *layout* en la función **initComponents()** (donde también introduzco en la lista **pinButtons** todos los **LinearLayoutCompat** correspondientes a los pines de la zona inferior de la vista. Esta función la invoco desde la función **onCreate**.

Desde el mismo **onCreate** invoco también la función **intListeners()**, que compendia el comportamiento de todos los elementos de la vista. Para los botones de la parte superior de la vista son métodos **setOnTouchListeners()**, para poder tener control sobre lo que hacen al ser pulsados y también el ser soltados. De este modo puedo hacer el efecto de iluminación cuando un botón es tocado por el usuario. Además, en estos métodos está implementado el código necesario para que, al pulsar un botón superior la cifra incremente y al pulsar uno inferior, la cifra disminuya. En caso de que la cifra de las centenas sea 5, las otras dos no podrán ser mayores de 1 y, si lo son en el momento de convertirse la de centenas en 5 las otras dos se convierten en 1.

Además, hay dos funciones para incrementar o decrementar la opacidad de la imagen de las teclas para que, cuando el botón sea usable la opacidad sea máxima y cuando no lo sea disminuya hasta un valor de 40 de 255.

Para el traspaso del número decimal representado en la parte superior de la vista a una versión decimal representada por los pines de la parte inferior está el método **numberCalc()**, donde recojo las cifras en forma de texto y las convierto en enteros. Una vez hecho esto las multiplico las centenas por 100, las decenas por 10 y las sumo a las unidades. El método **numberToBinary()** lo convierte en una *String* que representa el número en binario.

Con estas dos líneas me aseguro de que el número binario tiene 9 cifras para que coincida con el número de pines que tiene la aplicación:

val paddingZeros = 9 - binary.length

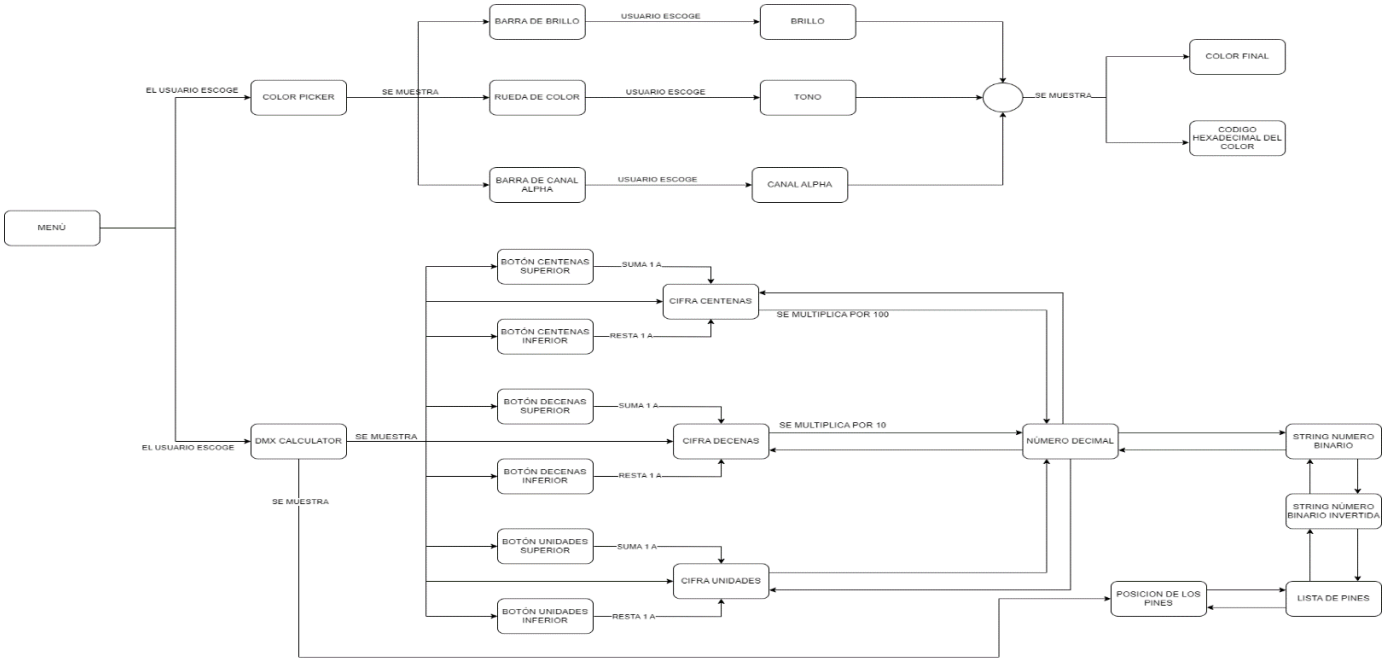
var paddedBinary = "0".repeat(paddingZeros) + binary

Invierto esta *String* y recorro la lista que llené con los **LinearLayoutCompat** y recorro el *String*. Si el bit es 1 el pin correspondiente toma una *gravity* TOP. Si el bit es 0, el pin correspondiente toma un *gravity* BOTTOM.

De este modo tengo la representación del número decimal en los pines de la parte inferior.

Para los pines sí que usé método **onClickListener()**, porque no necesitaba ningún efecto adicional. De este modo, cada vez que pulso un pin se inicia el método **changeGravityPin()** que lo que hace es invertir la gravedad de este, es decir, si tiene *gravity* BOTTOM la convierte en UP y viceversa. También se inicia el método **calculateNumber()** que recorre la lista **pinButtons** comprobando la *gravity* y reconstruyendo el número binario como *String*. Cuando termina de recorrerla invierte la *String* y llama al método **binaryToDecimal()** que convierte en número binario en decimal y lo convierte en las cifras que se verán modificadas en la parte superior de la vista.

DIAGRAMA DE LA APLICACIÓN



ASPECTOS QUE SE QUIERAN DESTACAR

Sé por experiencia propia que este tipo de aplicaciones son muy útiles en el día a día de un técnico de iluminación. En este caso solo he implementado dos herramientas, pero hay muchas otras que se podrían implementar añadiéndolas a la aplicación para tener centralizadas, en una misma app, todas las herramientas necesarias para el trabajo ágil en el momento de un montaje teatral o de una orquesta o de cualquier otro tipo de espectáculo en vivo.

Una de las ampliaciones que se me ocurren es conectar el colorPicker con las páginas web de Rosco y de Lee (las principales marcas de filtros en España) para que mostrase la numeración tanto de una casa como de la otra de los filtros con los colores más parecidos a los seleccionados sin necesidad de ir a la página y buscarlo tú mismo por el código hexadecimal que la aplicación te ha dado. El usuario tendría que dar un paso menos que la app le estaría ahorrando.

En cuanto al DMX Calculator, creo que es interesante explicar al lector de este documento (dado que creo que es algo poco conocido fuera del gremio de la técnica del espectáculo) lo que es un DIP Switch.

En estas tres imágenes se pueden ver dos dispositivos (a la derecha un dimmer, abajo a la izquierda una máquina de humor, abajo a la derecha un estrobo) con selección de canal por medio de un DIP Switch (ese rectángulo de color rojo o azul con pines blancos numerados.)

Ese canal, que se selecciona mediante los pines, es el que identifica el dispositivo dentro de toda la red DMX (protocolo de comunicación usado en los espectáculos) para que pueda comunicarse con la controladora que le dará las órdenes para actuar o no actuar.

En el caso de la máquina de humo, permitirá que el iluminador, durante el espectáculo, decida si debe o no tirar humo. En el caso del estrobo, ese canal será el que permitirá, desde la mesa de control de luces, decidir si el aparato se enciende o no. En el caso del dimmer, sirve de comunicación con otros aparatos que funcionen por tensión de corriente eléctrica (focos de incandescencia, sopladores de aire, pirotécnia, etc…) y se selecciona solo el primero de los canales. El de la imagen consta de 8 canales, con lo que en el DIP Switch se seleccionaría el primero y, a partir de ahí, cada uno de los canales tendría uno más que el anterior.

Algunos DIP Switch tienen 10 pines, es decir, uno más que la aplicación que presento. Este décimo pin tan solo es un interruptor (un On/OFF) y no tiene valor para la determinación del canal DMX.