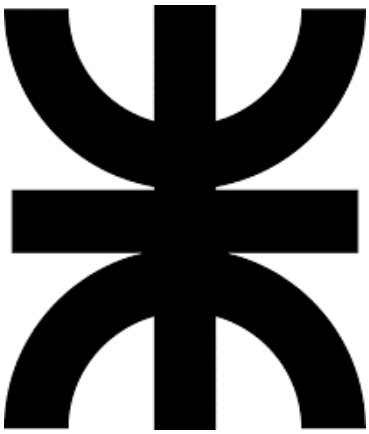
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL CÓRDOBA



G.IN.T.E.A.

Aplicación de Escáner en SDR++

LANZAMIENTO DE LA APLICACIÓN EN BASH

Autores:
Carlos Zerbini, Guillermo Riva, Luciano Martinez

CÓRDOBA, ARGENTINA
Marzo de 2024

CONTENIDO

1. Introducción

En el presente documento se explorará el uso de la aplicación SDR++ como escáner de radiofrecuencia (RF). Además, se detallará el funcionamiento del ejecutable sdrppEXE, diseñado para simplificar la experiencia de uso de SDR++ en aplicaciones que involucran tres receptores SDR.

2. Uso del ejecutable

Para garantizar el correcto funcionamiento del programa, es necesario ubicar la carpeta del ejecutable (sdrppEXE) en el escritorio (Desktop). Dentro de esta carpeta se encuentra el archivo **Scanner.sh**. Se debe otorgar permisos de ejecución al archivo utilizando el comando **chmod +x Scanner.sh**.

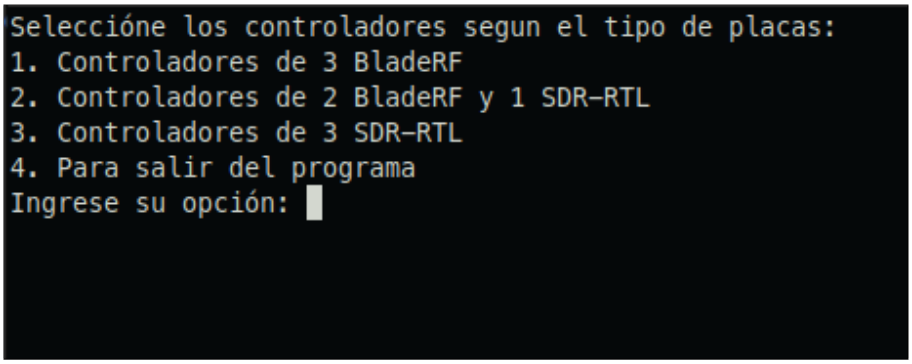


Figura 1: Menu de inicio del programa.

En la Figura 1 se muestra el menú de inicio del programa. Este menú permite seleccionar entre tres combinaciones de controladores de los receptores de SDR. En caso de no conocer el controlador de cada placa, se debe consultar la documentación correspondiente.

Figura 2: Tres instancias de SDR++.

Una vez seleccionada la combinación, se abrirán tres ventanas de la aplicación SDR++, con cada receptor seleccionado en su ventana correspondiente, como se muestra en la Figura 2.

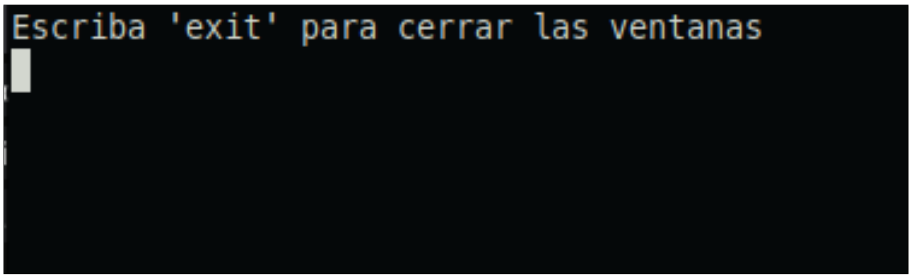


Figura 3: Comando de cierre del programa.

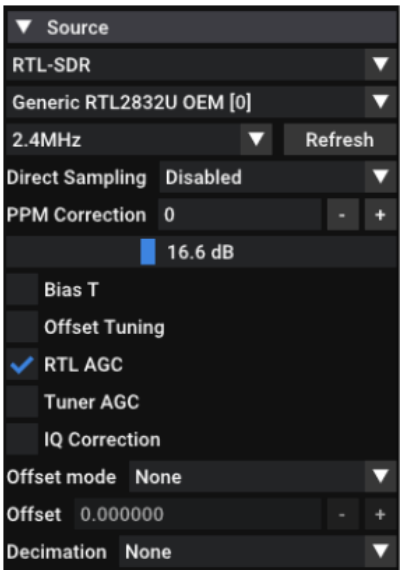
Después de abrir las tres ventanas de SDR++, el programa permite cerrarlas simultáneamente ingresando la palabra **exit** en la terminal, como se muestra en la Figura 3.

3. Uso de SDR++

Dentro de la aplicación, el usuario puede realizar diversas modificaciones para adaptarla a su aplicación específica como escáner de radiofrecuencia.

3.1. Selección del dispositivo

En el módulo de la izquierda se encuentra el menú de origen o *source*, como se muestra en la Figura 4a. Dentro de este menú se pueden realizar diversos cambios, como cambiar el tipo de receptor, tal como se observa en la Figura 4b.



(a) Menú fuente o *source*

(b) Selección de tipo de dispositivo

A su vez, si se tienen más de un dispositivo del mismo tipo conectado, se puede seleccionar cada uno individualmente en el segundo menú que se muestra en la Figura 4a con el nombre *Generic RTL2832U OEM [0]*. Este nombre corresponde a un

dispositivo específico, cuyo nombre puede cambiar según la configuración de la máquina.

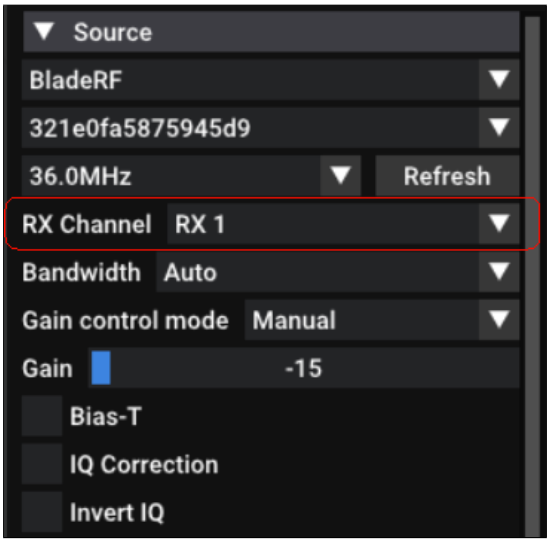


Figura 5: Selección de antena receptora.

Además de lo anterior, en caso de contar con un receptor que disponga de más de una antena receptora, como es el caso de las placas *nuand bladeRF*, el programa permite seleccionar cuál terminal de la placa utilizar. Esta configuración se puede modificar en la sección resaltada que se muestra en la Figura 5.

3.2. Control de ganancia y tasa de muestreo

Como se ve en la Figura 6, dentro del menú *source*, también se puede seleccionar la tasa de muestreo y ganancia, las cuales variarán según el receptor SDR que se utilice.

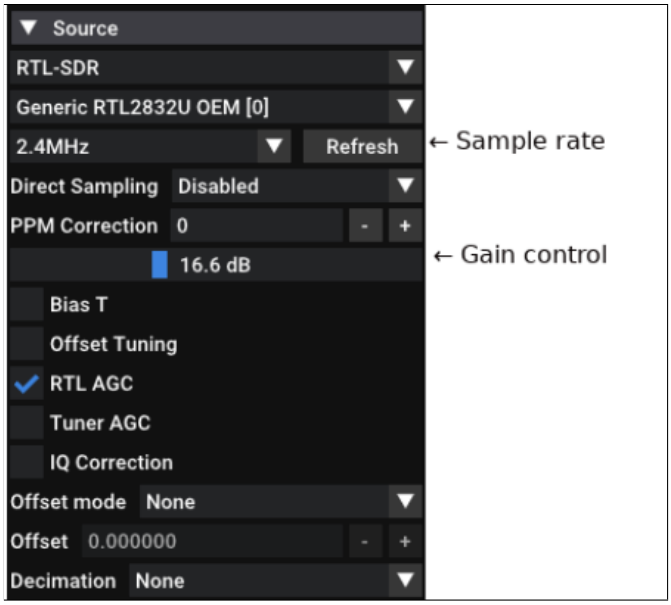


Figura 6: Selección de tasa de muestreo y ganancia.

Junto con esto, se pueden realizar otras selecciones, las cuales se pueden ver en el manual de la aplicación disponible junto con el presente documento.

3.3. Controles de visualización de frecuencia y amplitud

El software SDR++ ofrece la capacidad de ajustar la frecuencia y amplitud visualizadas en pantalla en tiempo real. En la Figura 7, se pueden apreciar tres controles en el módulo de la derecha, junto con un control para la frecuencia central en el módulo superior.

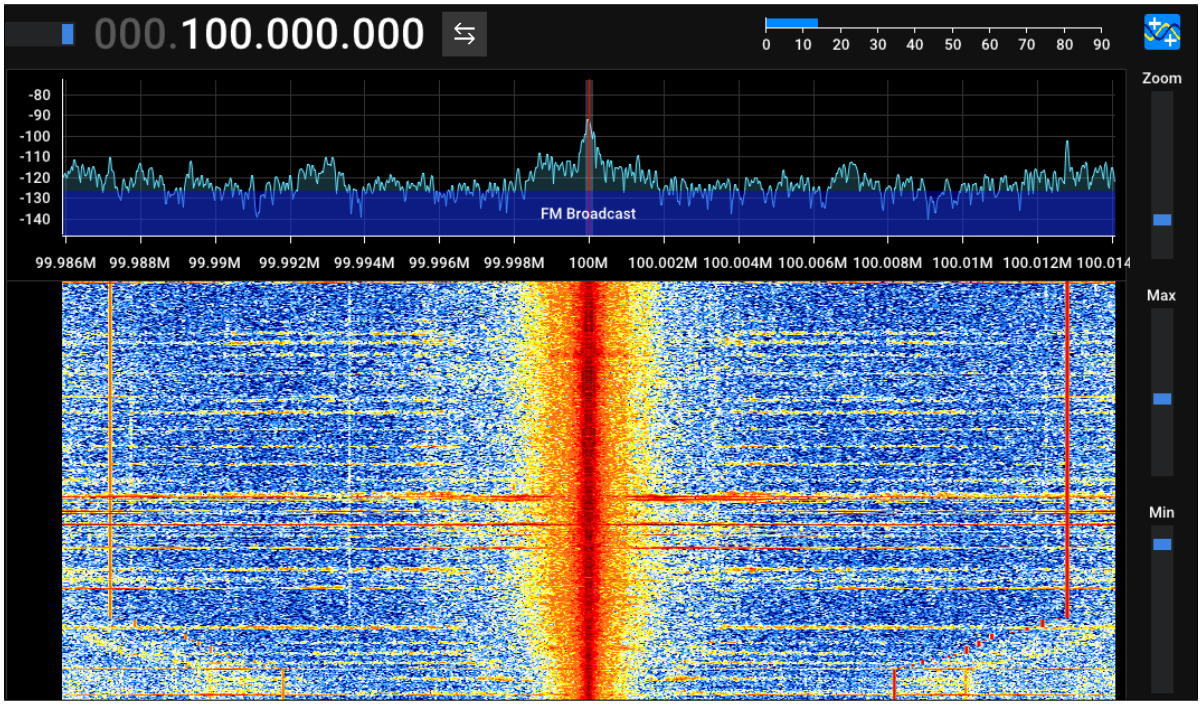


Figura 7: Controles de visualización de frecuencia y amplitud.

El control superior permite ingresar por teclado el valor de frecuencia central deseado. Además, esta puede ser ajustada utilizando el mouse, arrastrando la barra de frecuencia visualizada debajo de la gráfica dinámica.

Junto con lo anterior, se visualizan tres controles, los cuales consisten en lo siguiente:

- Zoom: permite controlar el ancho de banda visualizado en pantalla, realizando el zoom desde la frecuencia central.
- Max: controla el valor máximo de amplitud de las señales visualizadas en el espectro.
- Min: controla el valor mínimo de amplitud de las señales visualizadas en el espectro.

4. Código Bash del ejecutable

A continuación se explica el funcionamiento del código del ejecutable *Scanner.sh*, el cual está escrito en Bash y está diseñado para ser ejecutado desde la terminal de Linux.

```
# Bucle para ingresar una opción del menu
while [[ $OPTION -lt 1 || $OPTION -gt 4 ]]; do
    clear

    echo "Selecciónе los controladores segun el tipo de placas:"
    echo "1. Controladores de 3 BladeRF"
    echo "2. Controladores de 2 BladeRF y 1 SDR-RTL"
    echo "3. Controladores de 3 SDR-RTL"
    echo "4. Para salir del programa"

    read -p "Ingrese su opción: " OPTION

    if [[ $OPTION -lt 1 || $OPTION -gt 4 ]]; then
        echo "Opción no válida. Por favor, seleccione una opción válida."
    fi
done
clear
```

Figura 8: Código del menú principal.

En la Figura 8 se muestra el módulo principal del programa. A través de un bucle, imprime en pantalla el menú principal, lee la entrada del usuario y evalúa si es correcta. En caso de ingreso incorrecto, se repiten las sentencias. Si no hay errores, se sale del bucle y se pasa al siguiente módulo.

```
# Selección
case $OPTION in

    1)  echo "1. Controladores de 3 BladeRF"
        option_1_funtion
        exit_funtion
        ;;

    2)  echo "2. Controladores de 2 BladeRF y 1 SDR-RTL"
        option_2_funtion
        exit_funtion
        ;;

    3)  echo "3. Controladores de 3 SDR-RTL"
        option_3_funtion
        exit_funtion
        ;;

    4)
        echo "Saliendo del programa"
        exit 0
        ;;

esac
```

Figura 9: Código de la estructura de control.

En la Figura 9 se muestra la estructura de control que evalúa la entrada del usuario, ya validada. Basándose en esa selección, llama a las funciones correspondientes, excepto en el caso número cuatro, donde el programa se cierra.

El código de la Figura 10 corresponde a una estructura de validación de la existencia de los archivos de configuración. Esta función será llamada dentro de las demás funciones para verificar la existencia de los archivos *JSON*.

```
# Función para verificar si un archivo existe
check_file() {
    if [ ! -f "$1" ]; then
        echo "El archivo $1 no existe o no es accesible."
        exit 1
    fi
}
```

Figura 10: Código de la verificación de existencia de los archivos de configuración.

El siguiente módulo, que se muestra en la Figura 11, corresponde a la primera parte de la función *option_1_function*. En este módulo, para simplificar el uso de las direcciones de los archivos, se establece la dirección base de las carpetas de los archivos de configuración de SDR++ en *config_dir* y del ejecutable en *set_dir*. Luego, se llama a la función *check_file* para verificar la existencia de los archivos. Una vez validados, estos se copian al directorio de configuración del programa SDR++.

```
# Funciónes de escritura de archivos
option_1_function(){

    echo "Abriendo SDR++"
    # Dirección del directorio de los archivos de configuración
    config_dir="$HOME/.config/sdrpp"          # Dirección de la carpeta de configuración de SDR++
    set_dir="$HOME/Desktop/sdrppEXE/Set_1"    # Dirección de los archivos de configuración del ejecutable

    # Verifica si existen los archivos en la carpeta "sdrppEXE"
    check_file "$set_dir/Device_1/config.json"
    check_file "$set_dir/Device_1/bladerf_config.json"
    mkdir -p "$config_dir"

    # Copia los archivos a la carpeta de configuración
    cp "$set_dir/Device_1/config.json" "$config_dir"
    cp "$set_dir/Device_1/bladerf_config.json" "$config_dir"

    # Ejecutar SDR++
    /usr/bin/sdrpp &

    sleep 7s
```

Figura 11: Función del menú 1 para la apertura de ventanas.

Dentro de este módulo se deberá configurar la dirección de la carpeta de los archivos de configuración según dónde se posicione dentro del sistema. En este caso, la misma se encuentra en la carpeta *Desktop*.

Lo anterior corresponde a la configuración de una única ventana de SDR++. Es necesario realizar el mismo proceso nuevamente para los demás archivos correspondientes a las otras dos ventanas del programa, como se muestra en la Figura 12.

Este mismo proceso se realiza para las demás opciones del menú, el cual llama a las funciones *option_2_function* y *option_3_function*, las cuales poseen la misma lógica en el código, cambiando únicamente la carpeta del dispositivo seleccionado.

```
    # Segunda ventana
    echo "2da Ventana"
    check_file "$set_dir/Device_2/config.json"
    check_file "$set_dir/Device_2/bladerf_config.json"

    cp "$set_dir/Device_2/config.json" "$config_dir"
    cp "$set_dir/Device_2/bladerf_config.json" "$config_dir"

    /usr/bin/sdrpp &

    sleep 7s

    # Tercera ventana
    echo "3ra Ventana"
    check_file "$set_dir/Device_3/config.json"
    check_file "$set_dir/Device_3/rtl_sdr_config.json"

    cp "$set_dir/Device_3/config.json" "$config_dir"
    cp "$set_dir/Device_3/bladerf_config.json" "$config_dir"

    /usr/bin/sdrpp &

    sleep 7s
}
```

Figura 12: Función del menú 1 para la apertura de las otras dos ventanas.

Finalmente, en la Figura 13 se muestra una función que se llama dentro del *case* para facilitar el cierre de las tres ventanas simultáneamente ingresando la palabra **exit**. Esta función consiste en un bucle que imprime en pantalla un mensaje solicitando al usuario que ingrese la palabra para salir del programa. Luego, valida el ingreso en el mismo bucle y, en caso de que el usuario lo indique, cierra todas las ventanas del SDR++ con el comando *killall sdrpp*.

```
exit_function(){
    # Bucle de salida
    clear
    exit_word="exit"
    user_input=""

    while [ "$user_input" != "$exit_word" ]; do
        echo "Escriba 'exit' para cerrar las ventanas"
        read  user_input
    done

    # Cerrar todas las instancias de SDR++
    killall sdrpp

    echo "SDR++ cerrado"
    sleep 2s
    clear
}
```

Figura 13: Función para el cierre de las ventanas.

4.1. Archivos de configuración de SDR++

El código anterior busca gestionar los archivos de configuración de SDR++, ajustando el tamaño de la ventana de la aplicación y preconfigurando los archivos de la carpeta **/.config/sdrpp/** para cada placa receptora de SDR. En cada ventana, se deben configurar dos archivos de dicha carpeta: **config.json** y **<nombre de la placa>_config.json**, donde en <*nombre de la placa*> se debe sustituir por el nombre del controlador de la placa SDR correspondiente.

```
65      "moduleInstances": {
66          "Airspy Source": {
67              "enabled": true,
68              "module": "airspy_source"
69          },
70          "AirspyHF+ Source": {
71              "enabled": true,
72              "module": "airspyhf_source"
73          },
74          "Audio Sink": {
75              "enabled": true,
76              "module": "audio_sink"
77          },
78          "Audio Source": {
79              "enabled": true,
80              "module": "audio_source"
81          },
82          "BladeRF Source": {
83              "enabled": true,
84              "module": "bladerf_source"
```

Figura 14: Archivo de configuración de SDR++, sección de dispositivos.

El archivo *config.json* posee una sección de instancias de los distintos módulos soportados por la aplicación, algunos de los cuales se presentan en la Figura 14. Los nombres de estos dispositivos son importantes, ya que en la siguiente sección, mostrada en la Figura 15, en la línea 168 (que varía en cada terminal), se selecciona el dispositivo que se quiere abrir en la ventana actual.

```
159     "modules": [],
160     "modulesDirectory": "/usr/lib/sdrpp/plugins",
161     "offset": 0.0,
162     "offsetMode": 0,
163     "resourcesDirectory": "/usr/share/sdrpp",
164     "showMenu": true,
165     "showWaterfall": true,
166     "snrSmoothing": false,
167     "snrSmoothingSpeed": 20,
168     "source": "BladeRF",
169     "streams": {
170         "Radio": {
171             "muted": false,
172             "sink": "Audio",
173             "volume": 1.0
174         }
175     },
```

Figura 15: Archivo de configuración de SDR++, selección de dispositivos.

Al final del código del archivo, se encuentra la sección de configuración del tamaño de la ventana que se desea abrir, como se puede observar en la Figura 16.

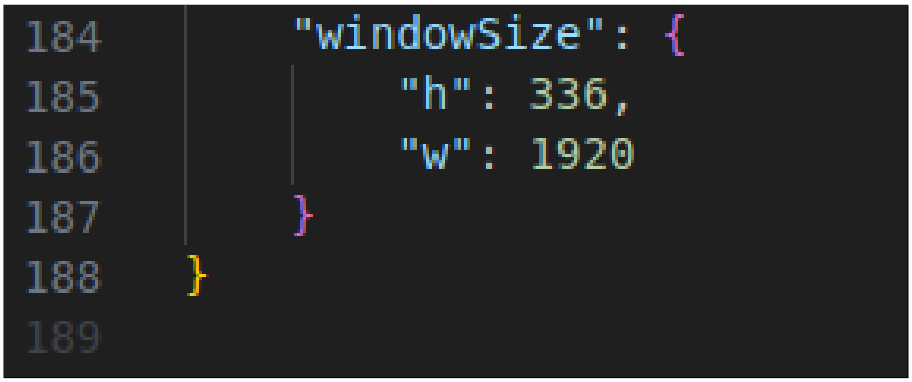


Figura 16: Archivo de configuración de SDR++, selección de tamaño de ventana.

El archivo **config.json** permite definir el tamaño de la ventana y el tipo de dispositivo que se desea preconfigurar en la ventana correspondiente. Para cada dispositivo específico, se dispone de un archivo **<nombre de la placa>_config.json**. Por ejemplo, en el caso de las placas nuand SDR bladeRF, este archivo se llamaría **bladerf_config.json** y contiene los parámetros específicos del dispositivo que se abrirá en cada ventana.

Figura 17: Archivo de configuración de BladeRF.

En la Figura 17, se muestra el código del archivo mencionado. En la línea 3, se encuentra el módulo *devices*, donde se definen dos submódulos correspondientes a dos placas del mismo tipo (nuand SDR bladeRF). En las líneas 4 y 11, se especifican los nombres de cada placa, los cuales son predefinidos por el controlador de cada tipo de placa en particular. Dentro del argumento de *device* en la línea 2, se debe establecer el nombre de la placa que se desea abrir en la ventana correspondiente.

Para cada una de las ventanas, se requieren los archivos de configuración específicos mencionados, los cuales deben ser configurados antes de utilizar el ejecutable.

Cada vez que se utiliza el programa SDR++, estos archivos guardan la configuración previa con la que se lo utilizó.

Para añadir un nuevo dispositivo, primero se debe abrir manualmente el programa y seleccionar el controlador y nombre del mismo, como se muestra en la Sección 3.1. Una vez que se cierra la aplicación, se deben ubicar los archivos en la carpeta **/.config/sdrpp/**, copiarlos dentro de la carpeta del ejecutable y agregarlos al código bash.

5. Solución de errores y conclusiones

Durante la ejecución del programa SDR++, pueden surgir ciertos inconvenientes. El principal problema suele ser la falta de instalación de los controladores correspondientes para una placa SDR específica. Por lo tanto, es necesario identificar primero el tipo de controlador que utiliza cada placa. Esta información puede encontrarse en la documentación proporcionada para cada dispositivo en particular. Los controladores que utiliza SDR++ están disponibles en la web. Por ejemplo, para la placa nuand SDR bladeRF, puede consultarse la guía de instalación de bladeRF en Ubuntu.

Si bien SDR++ permite seleccionar entre los distintos receptores de una placa, como se detalla en la Sección 3.1, no permite el uso simultáneo de más de una antena de una misma placa debido a limitaciones inherentes a la aplicación. Frente a esta restricción, se ha explorado la alternativa de emplear el software **GNURadio**, el cual permite el uso simultáneo de una o más antenas receptoras de una misma placa. En el siguiente foro se detalla cómo configurar **GNURadio** para esta aplicación.

Dentro de las aplicaciones en las que se busca hacer uso del software, se consideró la idea de utilizar placas de SDR-RTL económicas, con poco ancho de banda. Estas placas, mediante el módulo de **Scanner** disponible en la aplicación, permiten barrer un amplio rango de frecuencias. Sin embargo, un inconveniente con esta aplicación es que al guardar la información capturada por el programa, no se especifican las frecuencias, y además se solapan las muestras en los distintos saltos de frecuencia.

Ante este inconveniente, se exploró la posibilidad de utilizar el software Spektrum, el cual ofrece una interfaz similar a la de un analizador de espectro y permite visualizar amplitud y frecuencia en todo el ancho de banda del barrido, siendo superior a SDR++ en este aspecto. Sin embargo, esta solución no permite el almacenamiento de las lecturas tomadas por el receptor.

Se adjunta el enlace al manual de SDR++, junto con un enlace a las distintas versiones de SDR++ según el sistema operativo que se utilice para su instalación.