La aplicación del cliente se inicia y lee el SSID de la red WIFI al que esta conectado el dispositivo, si esta coincide con la red en la cual se va a realizar el broadcasting la aplicación pasa a la siguiente pantalla, de lo contrario, envía las credenciales para que el usuario se conecte a la red correcta. Ver Figura 1 y Figura 2

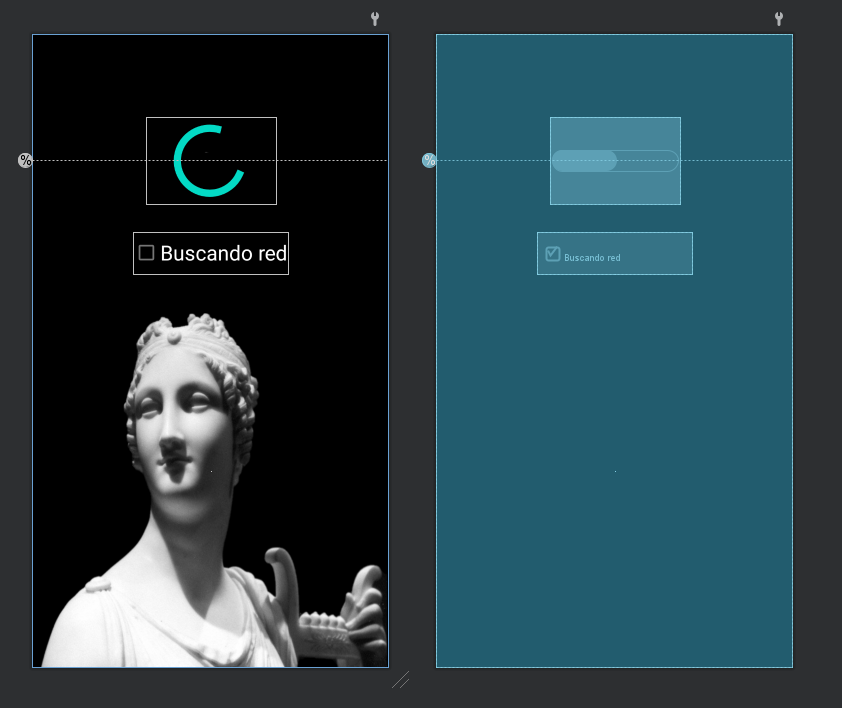


Figura 1: Pantalla inicial

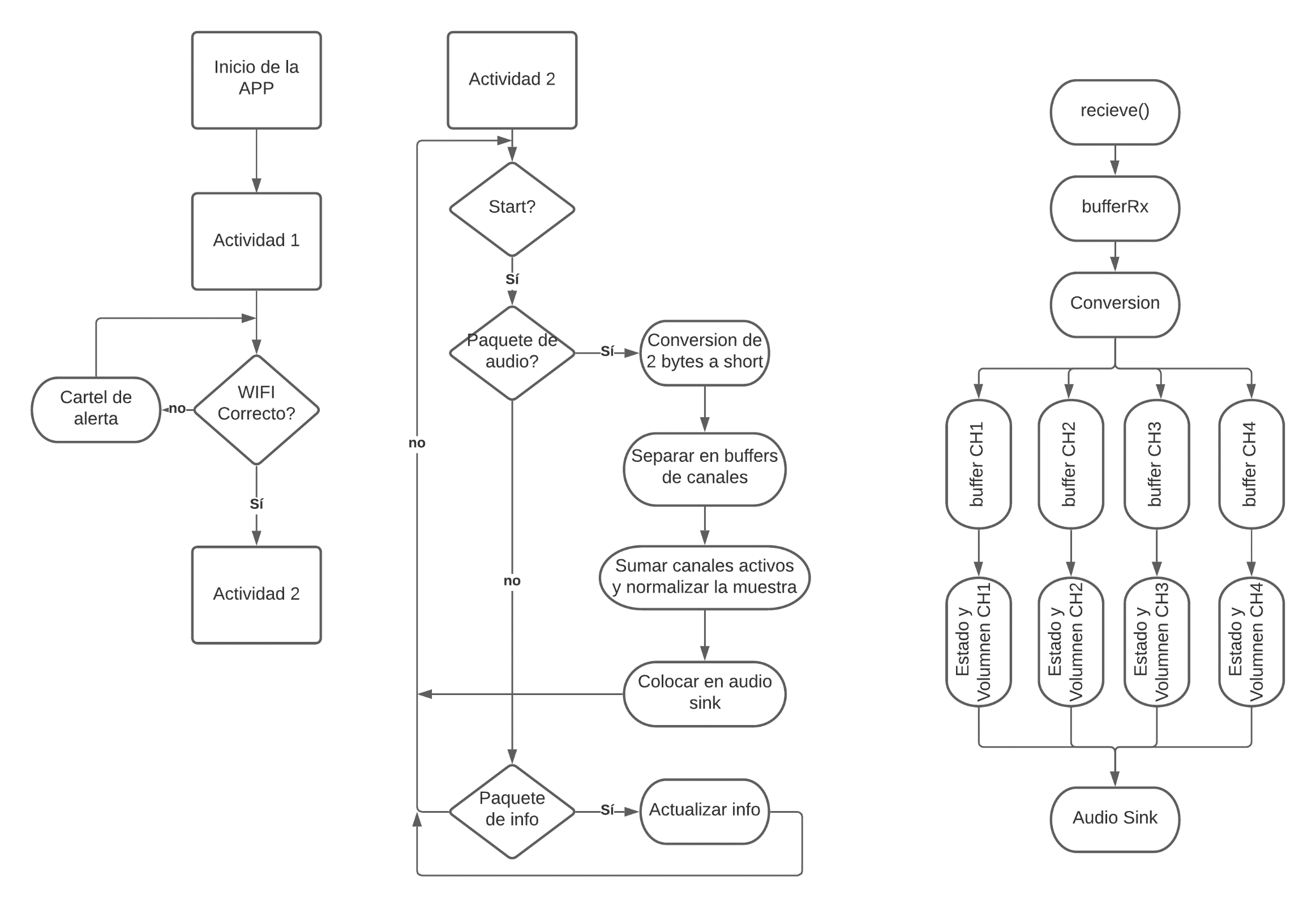


Figura 2: Flujo de la pantalla inicial

Una vez que se está en la red correcta y se accede a la segunda pantalla se disponen de los controles de los diferentes canales y un botón para detener o iniciar el streaming si el usuario lo desea. Ver Figura 3 y Figura 4.

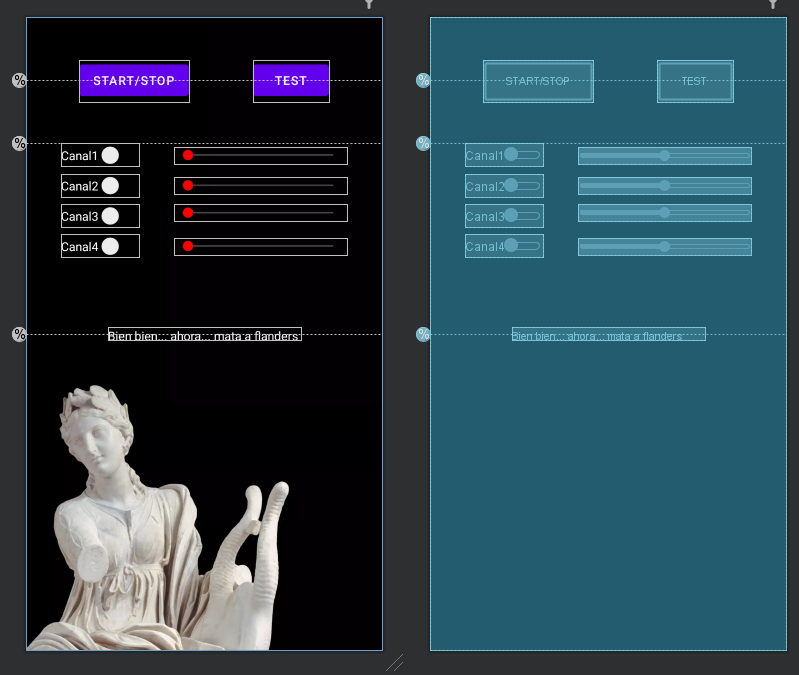


Figura : Pantalla controles

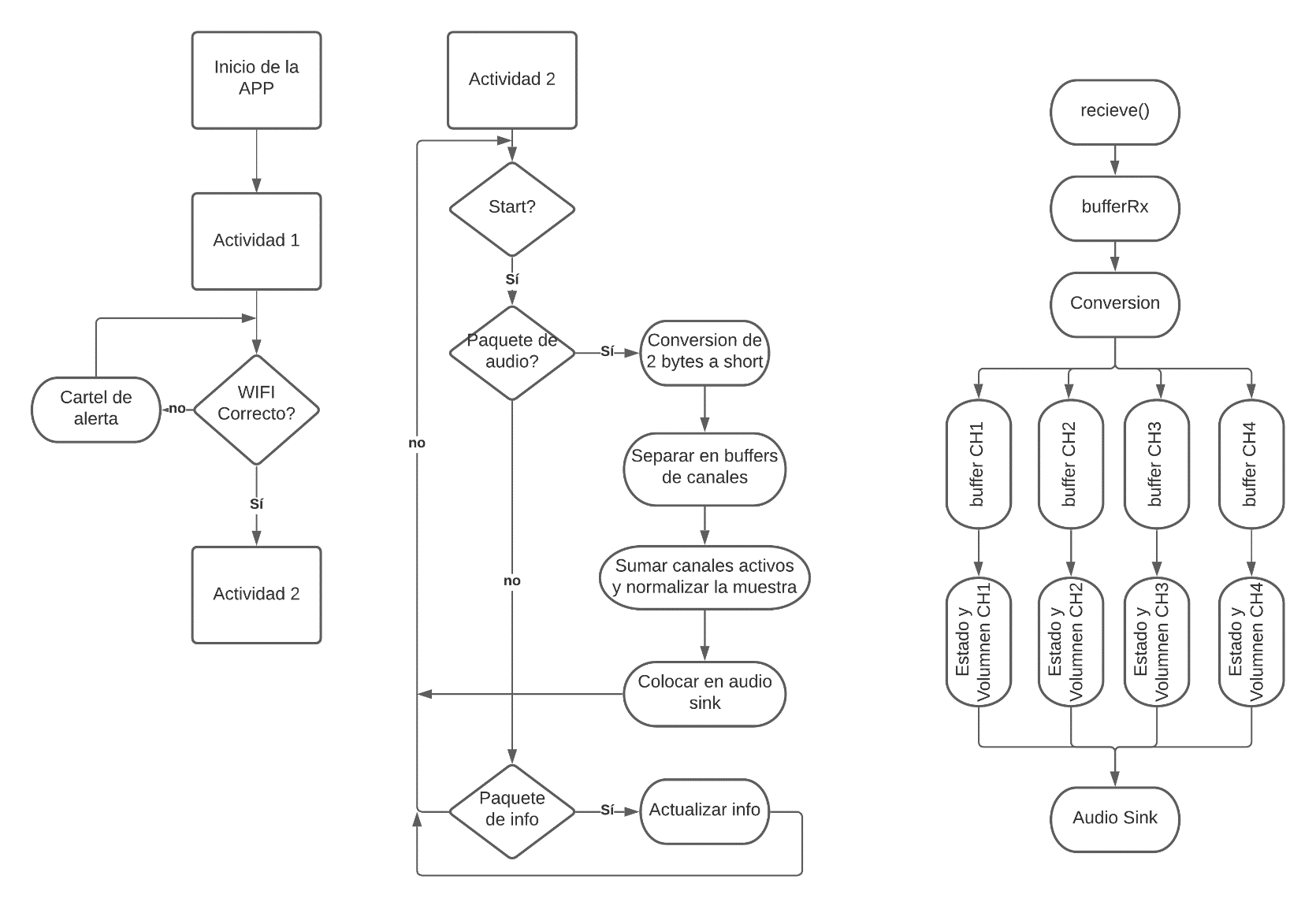


Figura : Flujo de la pantalla controles

La aplicación en esta segunda pantalla se encuentra esperando la recepción de paquetes en un socket creado para la transmisión multicast. El socket se crea apenas se inicia la segunda pantalla y una tarea asincrónica es encargada de recibir los paquetes. Los mismos se transmiten por UDP y el servidor los envia en cuanto los tiene listos, esto fuerza a tener una tarea dedicada a la recepción de paquetes. En cuanto esta tarea recibe un paquete realiza una discriminación primaria para identificar que tipo de paquete es, información o audio, y así colocar el paquete en el buffer correcto para su procesamiento.

Si el paquete es de audio se lo coloca en el bufferRx. Por la naturaleza de la comunicación cada muestra individual de audio se envia en 2 bytes separados, los mismos deben reagruparse en el receptor y por la naturaleza del receptor los datos deben convertirse al tipo de variable short para que sean aceptados por el reproductor.

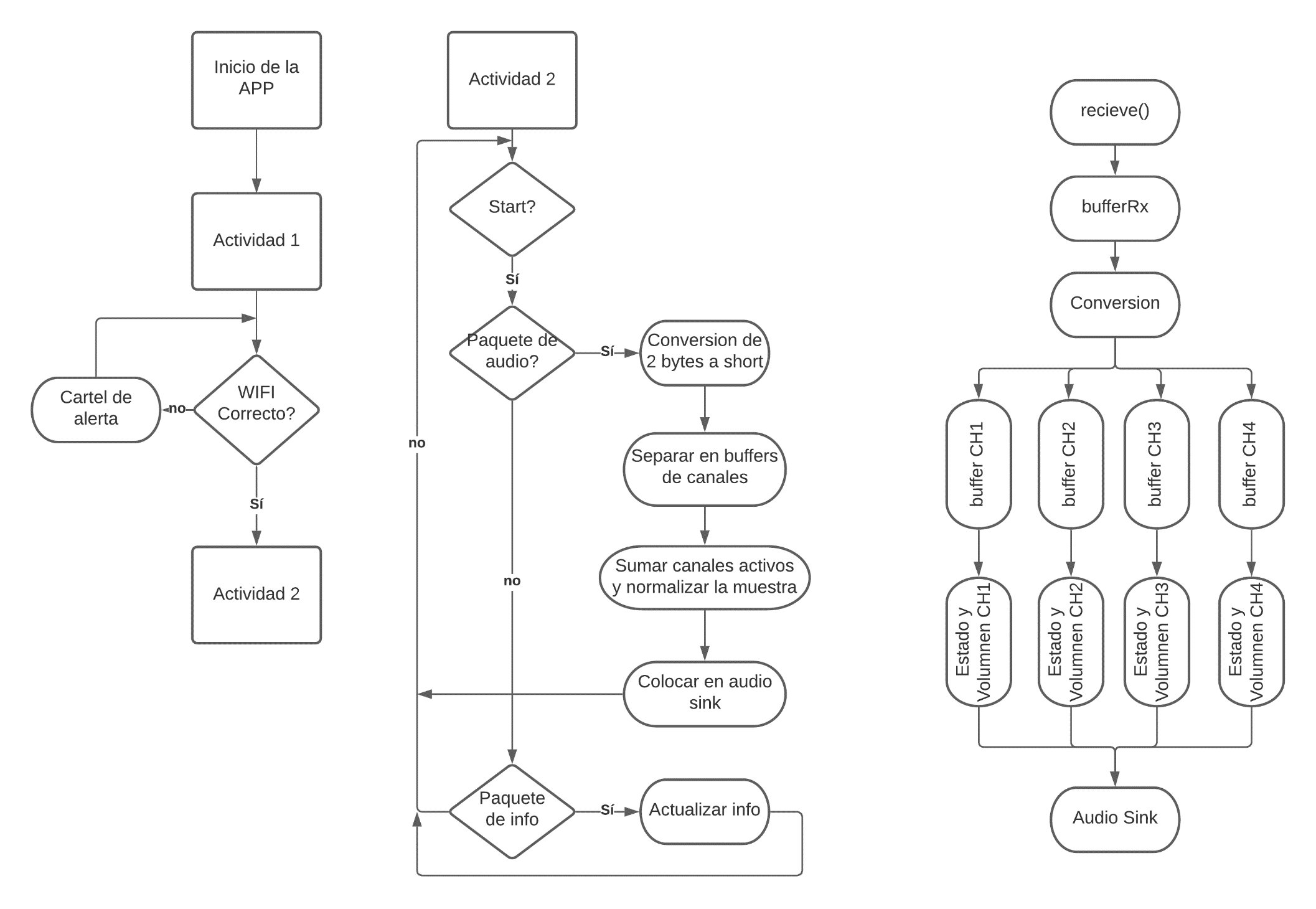


Figura 5: Flujo de datos

En un solo paquete se envían un gran número de muestras (actualmente 120 muestras) por cada canal (lo cual deja 480 muestras diferentes en el bufferRx por cada paquete recibido). Estas muestras deben separarse mientras se las convierte, las mismas se separan en 4 buffers diferentes para poder discriminar cada canal y aplicarle los controles deseados. Una vez separadas las muestras de los canales se les aplican los controles individuales (encendido y volumen) a cada muestra y se suman los 4 canales y se deja en un buffer auxiliar para que sea enviado al reproductor. Para toda esta tarea se creo un nuevo thread el cual espera la liberación de un semáforo que es liberado por la tarea de recepción, cuando este semáforo se libera, significa que hay datos disponibles en el buffer y pueden ser procesados.

Una vez que los datos del audio son puestos en el buffer auxiliar se libera otro semáforo el cual esta esperando ser tomado por el thread que toma los datos de este buffer y los envia al reproductor. Este thread también revisa el estado de los canales ya que si se encuentran todos apagados detiene el streaming para disminuir el consumo de energía y recursos en el dispositivo.

Estado de desarrollo de la app TherpsiCORE

Características escritas y probadas falta integrar al master

Características posibles a sumar

Características integradas a la app

Características con problemas

* Doble pantalla (Inicial + control de canales)
* Selección de canal individual
* Volumen individual de canal
* Log general de información
* Verificación de la correcta red wifi
* Recepción de mensajes de servicio desde servidor
* Separación de información en canales
* Incorporar un “vúmetro visual”
* Incorporar telemetria de uso
* Manejo dinámico de puertos e IP
* Modificar frecuencia de muestreo
* Mejoras visuales de la app (imágenes, colores, etc)
* Modificación de los textos en la app desde el servidor