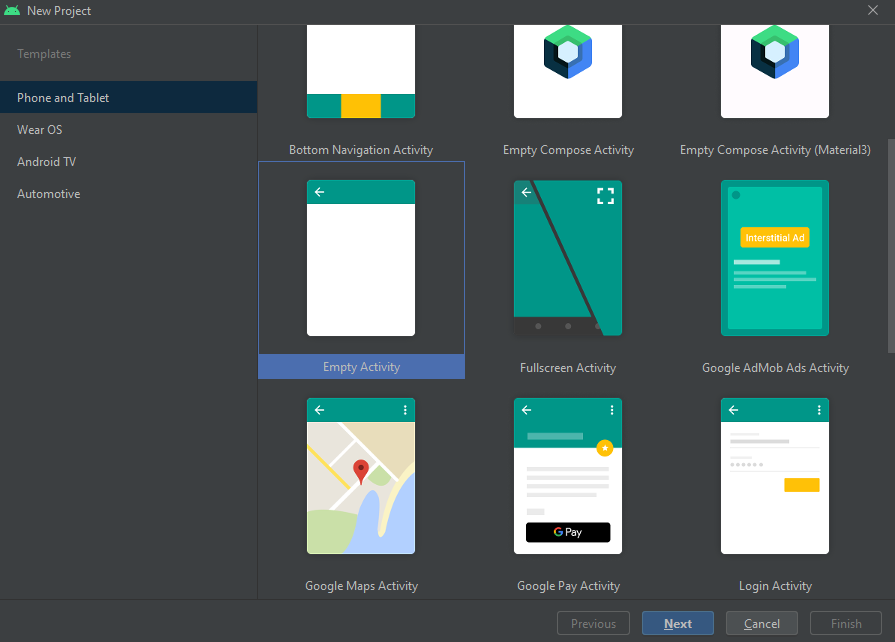
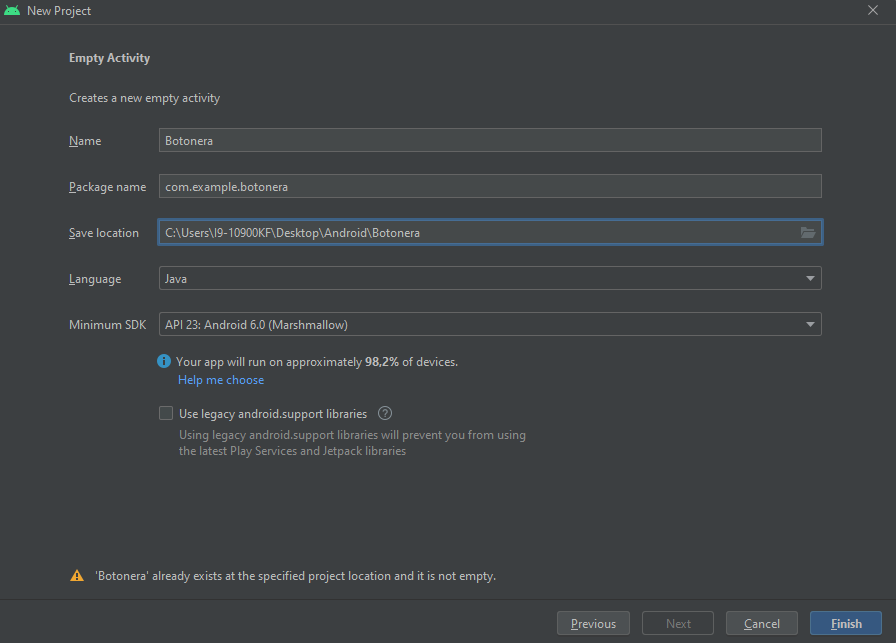
Manual de Desarrollo Aplicación Móvil y ESP32

**Desarrollo de Aplicación**

1. **Crear Nuevo Proyecto:**

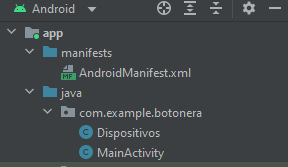
Para el desarrollo de la aplicación móvil se utilizó Android Studio, más específicamente la versión Dolphin 2021.3.1.17, la cual. Para crear un nuevo proyecto, primero se inicia la aplicación de Android Studio, una vez dentro se presiona el botón “New Proyect”, aquí aparecerá una serie de opciones para el inicio de la aplicación, para este caso se usará el “Empty Activity” que se encuentra en “Phone and Tablet”, esta una página en blanco, la cual se puede editar libremente, también posible usar alguna de las otras opciones, pero se debe tener en cuenta que estas tienen sus propias configuraciones.

Una vez seleccionada la actividad que se usará, se pasa a la siguiente página, en esta se puede cambiar el nombre y dirección de en qué se almacenará la ampliación. Para este proyecto se estará utilizando Java como lenguaje de programación, pero también puede realizarse con Kotlin, de todas formas, la aplicación de Android Studio tiene la opción de transportar el código de Java a Kotlin si se crea un proyecto en Java. Por última, en la opción del SDK mínimo se indica la versión de Android mínima requerida para ejecutar el programa, para este caso se utilizará la versión API 23 correspondiente a la versión de Android 6.0, debajo aparecerá un porcentaje aproximado de dispositivos que abarcará dependiendo de la versión mínima seleccionada.

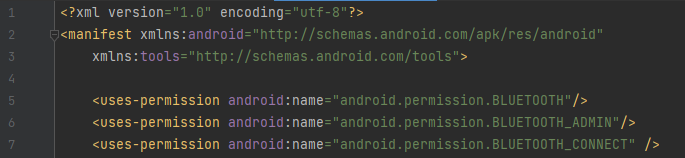


* 1. **Interfaces Graficas:**

Esta aplicación cuenta con dos pantallas (o actividades), la primera es la que muestra los dispositivos vinculados al celular, la segunda es la página principal que contiene la botonera, el código de ambas se puede encontrar en la carpeta java, con los nombres de “Dispositivos” y “MainActivity” respectivamente.

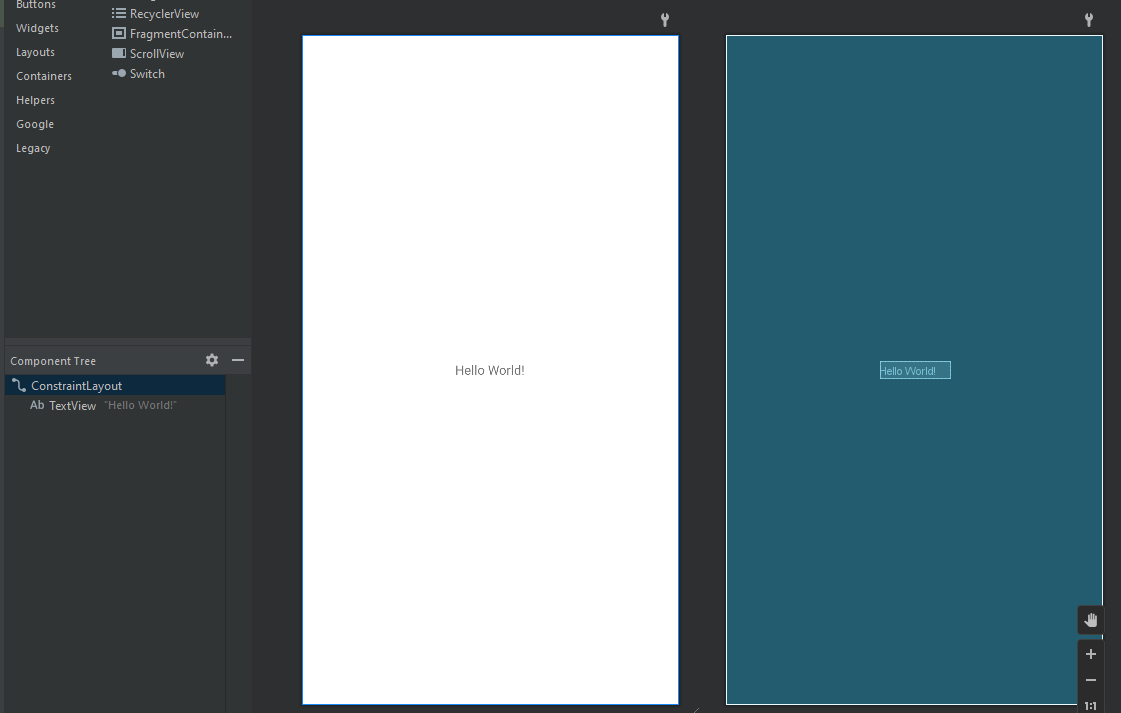


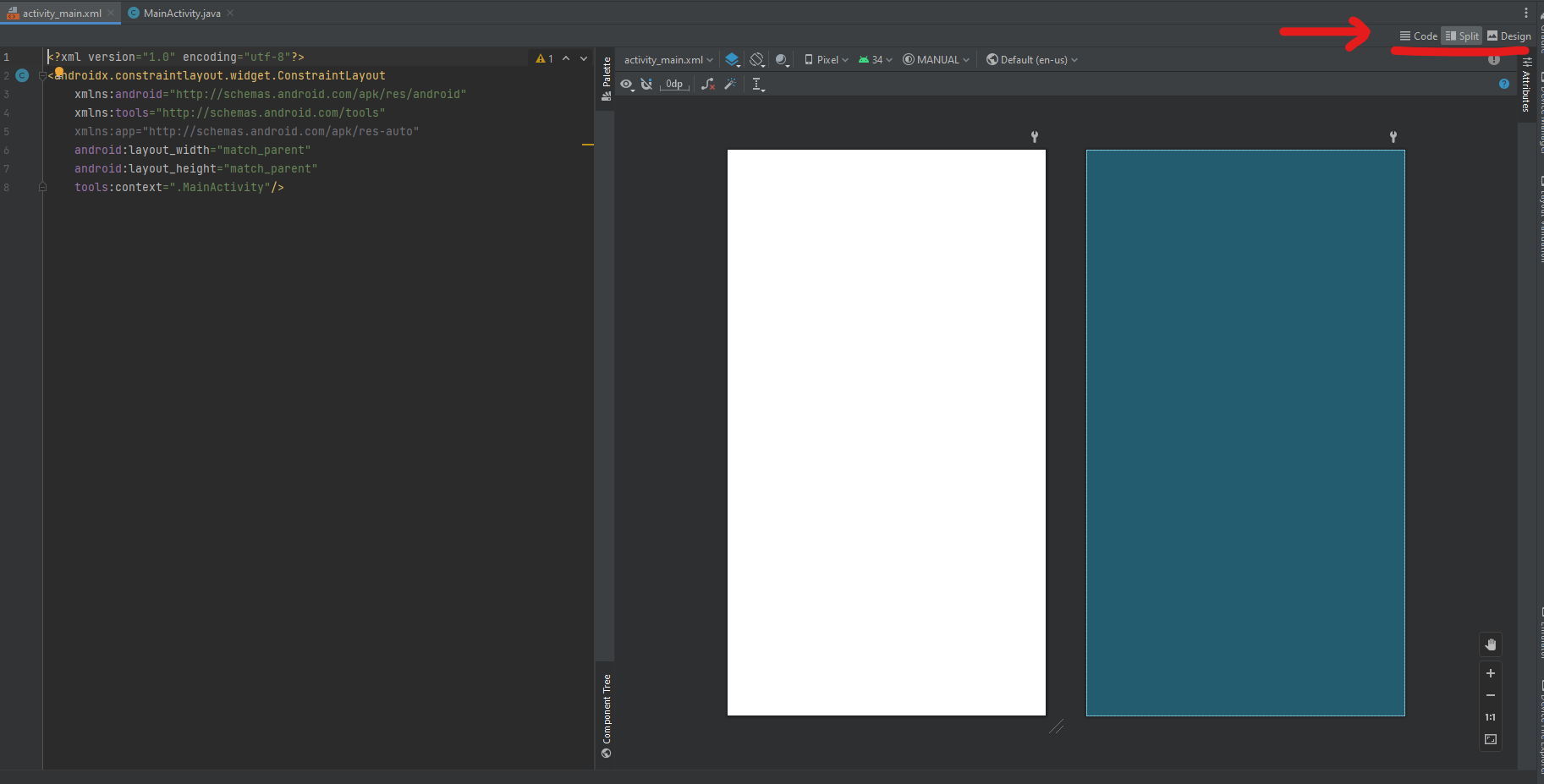
En un principio, al crear un nuevo proyecto, la única “activity” que aparecerá es MainActivity, la cual viene por defecto, las demás activites se pueden añadir dentro del proyecto, pero por el momento nos concentraremos en la página principal, la cual corresponde a la botonera.

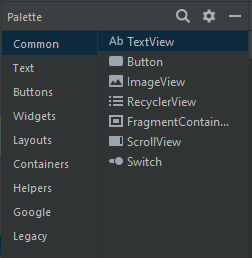
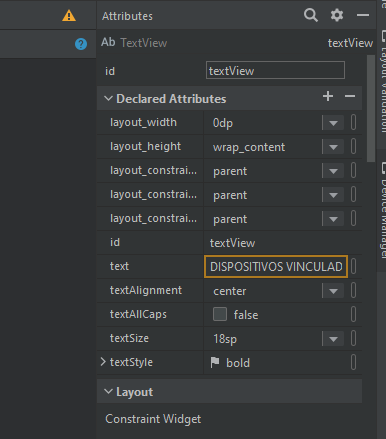
Otra cosa que es importante mencionar es la carpeta manifests, en la que se encuentra un archivo .xml que contiene la información esencial de la aplicación, como el nombre de esta, la paginas que cargará el programa, etc. Aquí se añaden los permisos de Bluetooth que se utilizaran, los cuales son necesarios para que la aplicación acceda al Bluetooth del dispositivo, por el momento no se usaran en la página principal, pero serán importantes para más adelante.

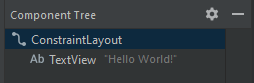
Al iniciar un nuevo proyecto se cargarán dos archivos, “MainActivity.java”, que es donde ira todo el código del activity, y “activity\_main.xml”, que es donde está todo lo referente a la interfaz gráfica del activity

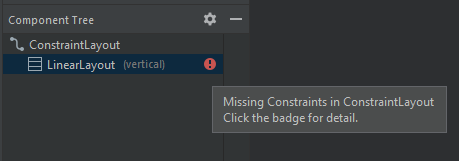


 Por hora dejaremos el código de lado y pasaremos a la interfaz gráfica, en esta solo aparecerá un “TextView” que dice “Hello World”, este no es necesario un puede eliminarse presionando sobre él y dándole al botón suprimir.

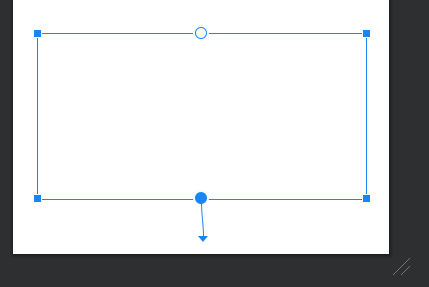
La interfaz se puede visualizar como código xml, diseño o ambas seleccionando en el panel que se encuentra en la esquina superior derecha

Para editar cualquier componente de la interfaz gráfica basta con presionar sobre él, esto desplegara el panel de atributos, en donde se encuentran todas las características de dicho componente, aquí se pueden editar cosas como el id, tamaño, color, etc. Si no hay ningún componente seleccionado el panel mostrara los atributos de la página completa (Correspondiente al componente principal “ConstraintLayout”, el cual contiene al resto de componentes). También es posible agregar más componentes a la pantalla arrastrándolos desde el panel “Palette”, y pueden visualizarse los componentes que ya se encuentran en la pantalla en el panel “Component Tree” que se encuentra un poco más abajo.

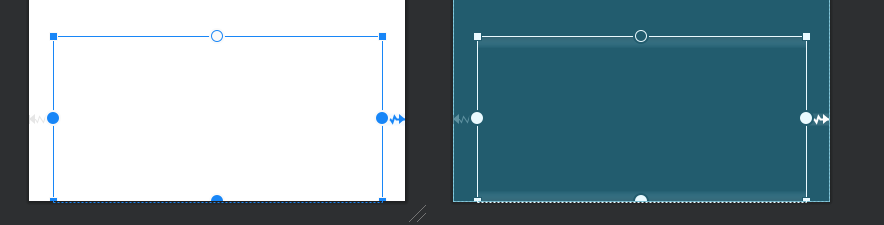
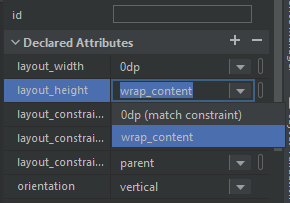
****

Continuando con la botonera, en la sección de Layouts de la paleta se debe buscar el componente “LinearLayout (Horizontal)”, también se puede buscar utilizando la lupa en la parte superior. Una vez encontrado, se arrastra hacia la pantalla, este aparecerá automáticamente en el árbol de componentes. LinearLayout es otro Layout que contendrá sus propios componentes, se utilizará para mantener una separación entre estos y que la botonera se vea más ordenada.

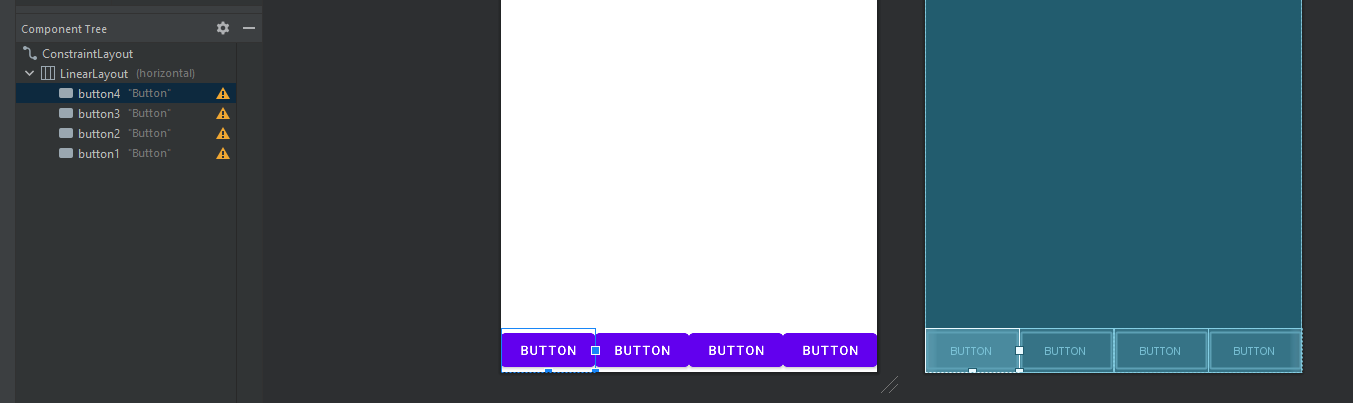
Aparecerá un error en el LinearLayout, este error indica que los límites del componente no están definidos, para corregir esto, primero se ajusta el tamaño para que no cubra toda la pantalla, luego sujetando uno de los círculos de los extremos, se arrastra hacia el borde correspondiente. Se debe hacer esto con todos los extremos exceptuando el superior.



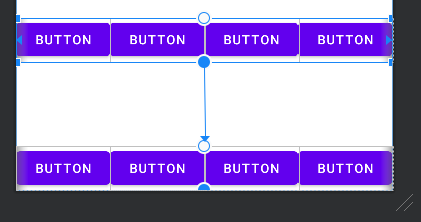
El resultad debe verse algo así.

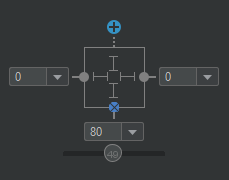
 En los atributos se el tamaño, dejando el layout\_width en 0dp, para que se ajuste al tamaño de la pantalla, y el layout\_height en wrap\_content, para que su tamaño se ajuste dependiendo de lo que haya en su interior, como por el momento está vacío, se verá plano.

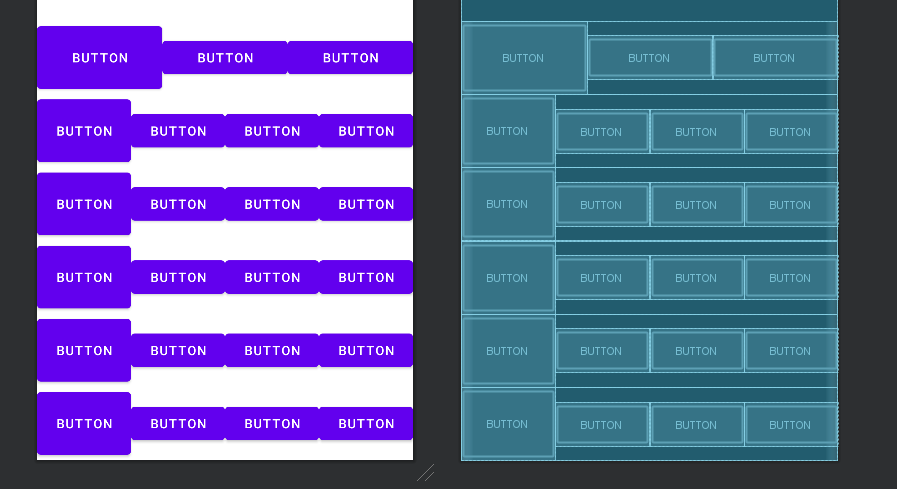
Cabe decir que los tamaños de los componentes, textos y demás elementos que se mostraran a partir de este punto son solamente referenciales y pueden ser ajustados libremente al gusto del desarrollador. Así como los tamaños, los nombres de ID asignados para este programa pueden ser modificados o se pueden mantener el que viene por defecto en Android Studio, sin embargo, todos los componentes deben tener su propio ID y el nombre de este no puede repetirse.

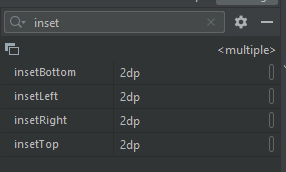
A continuación, se arrastra un componente “Button” dentro del Layout en el árbol de componente, en total deben hacer 4 botones, estos ajustaran su tamaño automáticamente al ingresar gracias al wrap\_content.

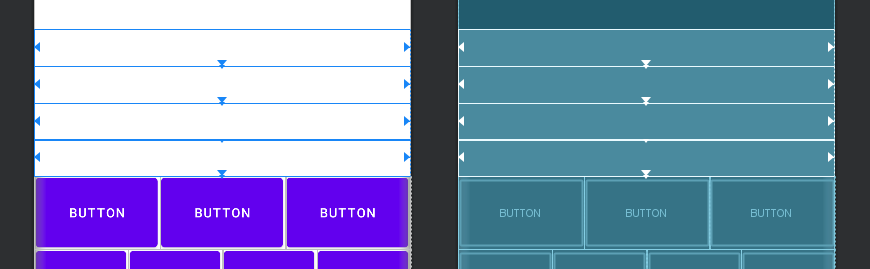
En total deben haber 6 Layout por lo que se debe repetir este proceso, con la diferencia de que en borde inferior del nuevo Layout debe conectarse al borde superior del anterior. El ultimo Layout de arriba solo debe contener 3 botones en lugar de 4.



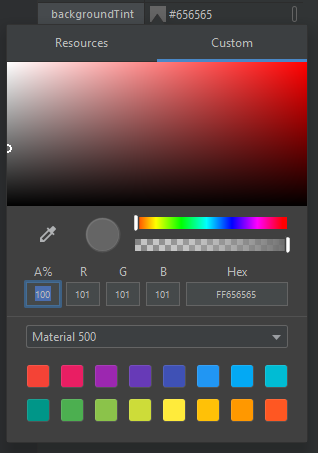
En caso de cometer algún error algún error con los límites, estos se pueden eliminar en los atributos dándole al botón del respectivo limite, luego desde la interfaz se podrá arrastrar nuevamente hacia el lugar correcto.

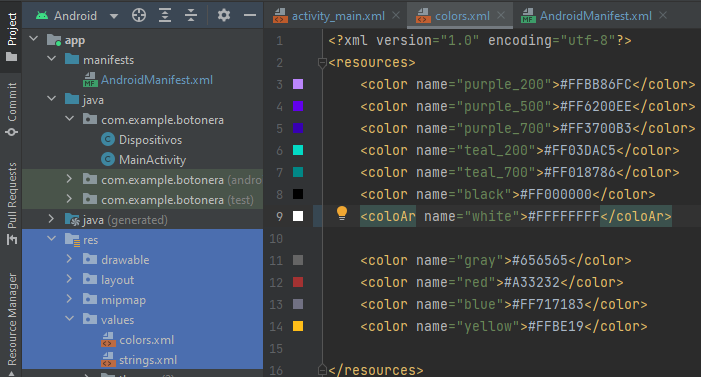
Para darle mejor forma, se seleccionan todos los botones que estén al fondo del lado izquierdo y se cambia su atributo layout\_height de wrap\_content a 80dp.

Al resto de botones se les ajusta el layout\_height como match\_parent, de esta forma se ajustarán al tamaño del Layout automáticamente. Para evitar que los botones estén demasiado juntos, en la lupa de los atributos hay que buscar “inset”, y configurar los resultados como “2dp”, esto ajustara un poco el tamaño para que los botones no se vean tan justos. El “layout\_height” de todos los botones debería ajustarse a 1 automáticamente, pero de no ser el caso se debe ajustar manualmente y dejarse 1. El resultado debe verse similar a esto.

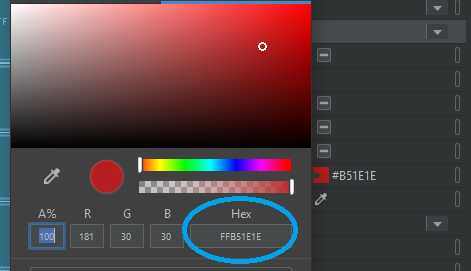
 Lo siguiente es añadir a la pantalla los 4 TextView que servirán para mostrar la información. Para eso se arrastran a la pantalla, se adjuntan los limites laterales al borde de la pantalla y los inferiores al límite superior del componente de abajo, de la misma forma en que se hizo con los Layout. Para una mejor visión de estos se ajusta el layout\_width en 0dp y el layout\_height en 40dp. Se elimina el texto de atributo “Text” para que estos se encuentren en blanco al momento de iniciar el programa, y se busca el atributo “TextSize”, al cual se le da un valor de 25sp para visualizar los mensajes que aparezcan en estos. Cabe mencionar que se usa “dp” para el tamaño de los objetos, y “sp” para el tamaño del texto.

Si se desea cambiar el color de los botones existe dos opciones: La primera es buscar el atributo “backgroundTint” y ajustar el color desde ahí.



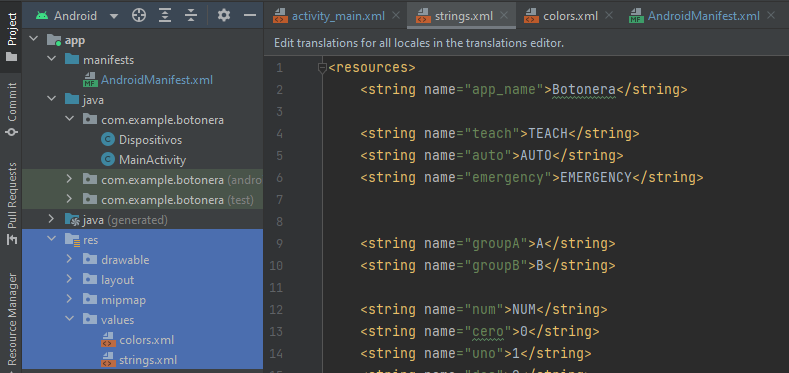
La segunda opción es crear un nuevo color, para esto se debe ir a archivo “colors.xml”, el cual se encuentra en la carpeta “values” dentro de “res”. Una vez aquí, se crea un nuevo color con la línea de código “<color name="(*NOMBRE DEL COLOR*)"> (*CODIGO HEX*) </color> “, de la forma que se muestra en la imagen.

A modo de sugerencia, este código puede obtenerse de cualquier atributo que tenga que ver con color.

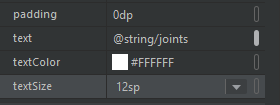


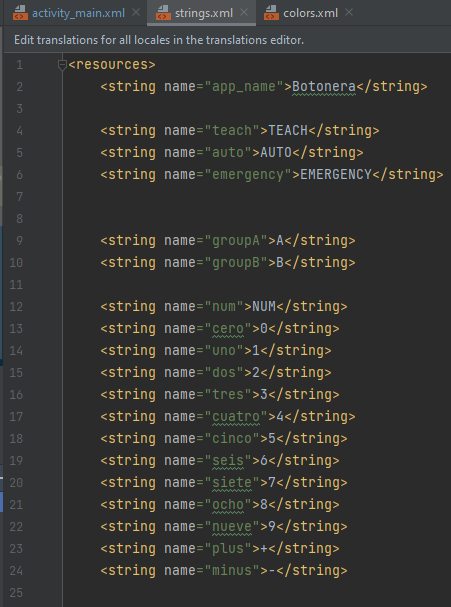
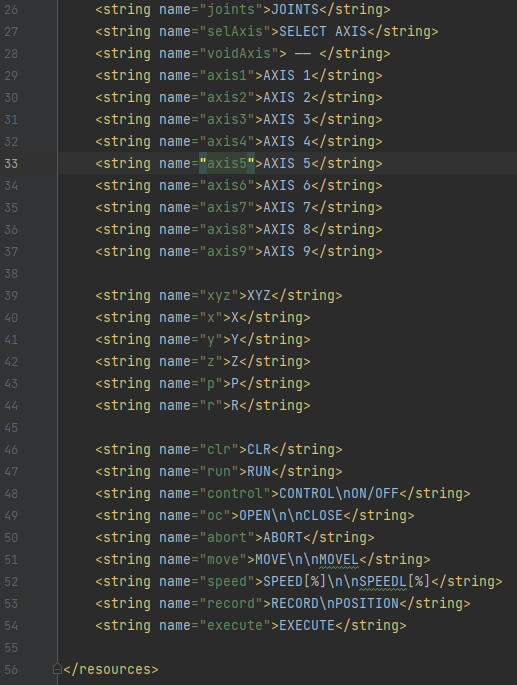
Por último, en el atributo “backgroundTint” se busca el color escribiendo “@color/ (*NOMBRE DEL COLOR*), para este caso se le dio un color gris a todos los botones, exceptuando el botón de emergencia, el cual es de color rojo. Crear un nuevo color es útil cuando este mismo se va a usar en repetidas ocasiones, o en varios componentes, de este modo no se ajusta el color de los componentes uno por uno buscando entre la paleta, si no que se añade el color ya creado directamente.

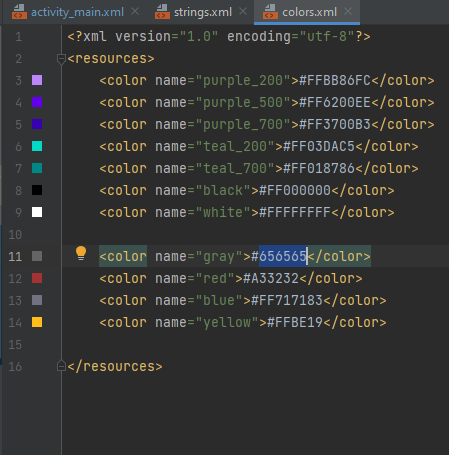


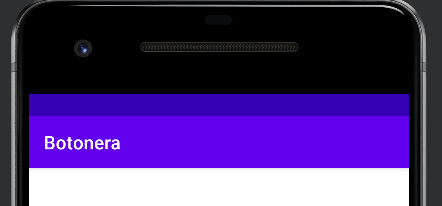
Para cambiarle el nombre a los botones, también existen dos formas: La primera es escribir el nombre del botón directamente en el atributo “Text”, esto es posible, sin embargo, la aplicación de Android Studio sugiere no hacerlo, en su lugar, la otra opción es ir al archivo “string.xml”, el cual se encuentra en el mismo lugar que “color.xml”, y aquí crear una String nuevo, que contenga el mensaje, esto logra añadiendo el comando “<string name="(*NOMBRE DEL STRING*)">(*MENSAJE*)</string> “.

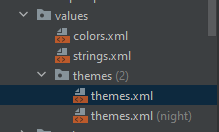
Y finalmente agregar en el atributo Text el String creado, escribiendo “@string/ (*NOMBRE DEL STRING*). Se le da un tamaño de letra de 12sp, y un color blanco para que contraste con el fondo gris del botón, es importante definir un color de letra, dado que, si no se hace, la letra tomara automáticamente el color del fondo general, el variará dependiendo si el dispositivo se encuentra en modo claro u oscuro. También se añadió un padding de 0dp, como su valor es de 0 no afecta demasiado y puede omitirse, pero añadir este atributo ayuda a que se ajuste correctamente el tamaño de los botones, los cuales crecen según el tamaño del mensaje que estos muestran, pero vuelven a tener el mismo tamaño con un padding.



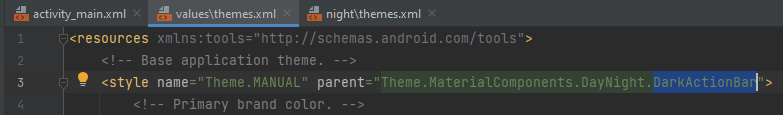
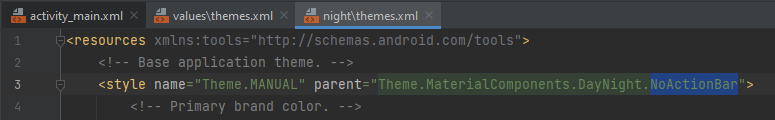
 Al igual que para el caso de los colores, es útil crear un String por si en otro momento debe usarse. A continuación, se presentarán todos los String y colores que se están utilizando para este proyecto.



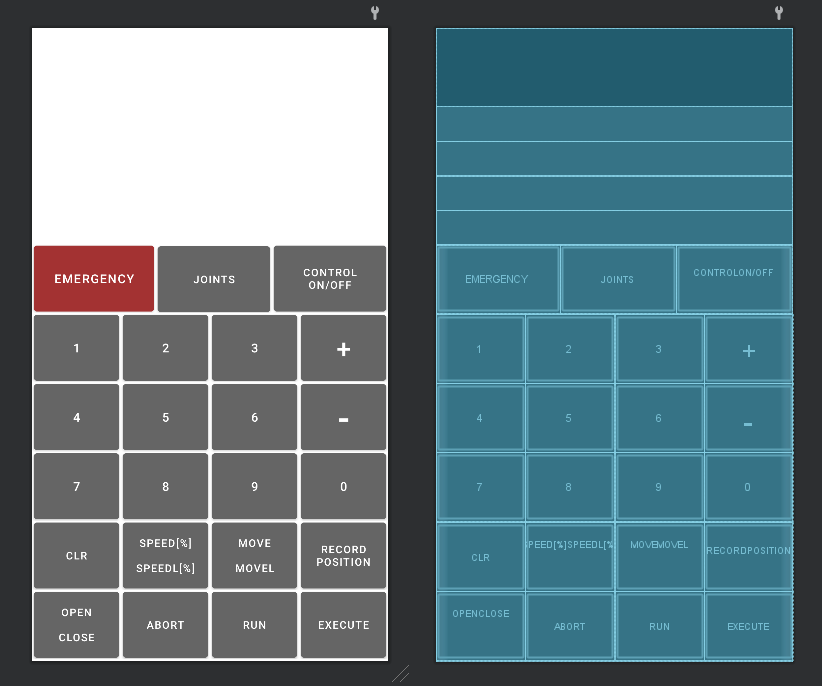
 Lo último referente a la interfaz gráfica de la página principal es la “ActionBar”, la cual es una pequeña barra con el nombre de la aplicación, esta viene por defecto al crear un programa.

Para quitarla debe ir al archivo “themes.xml”, que se encuentra en la carpeta “themes”, dentro de “value”.

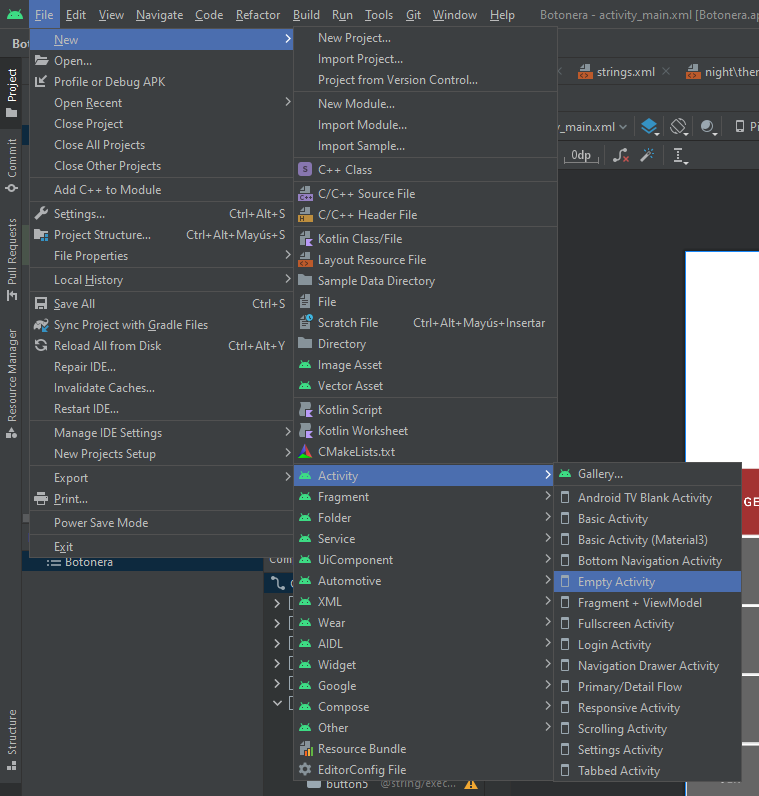
Aquí se debe cambiar la parte que dice “DarkActionBar” por “NoActionBar”. Esto se debe hacer en los dos archivos “themes”. El archivo “night” corresponde al modo oscuro del teléfono.

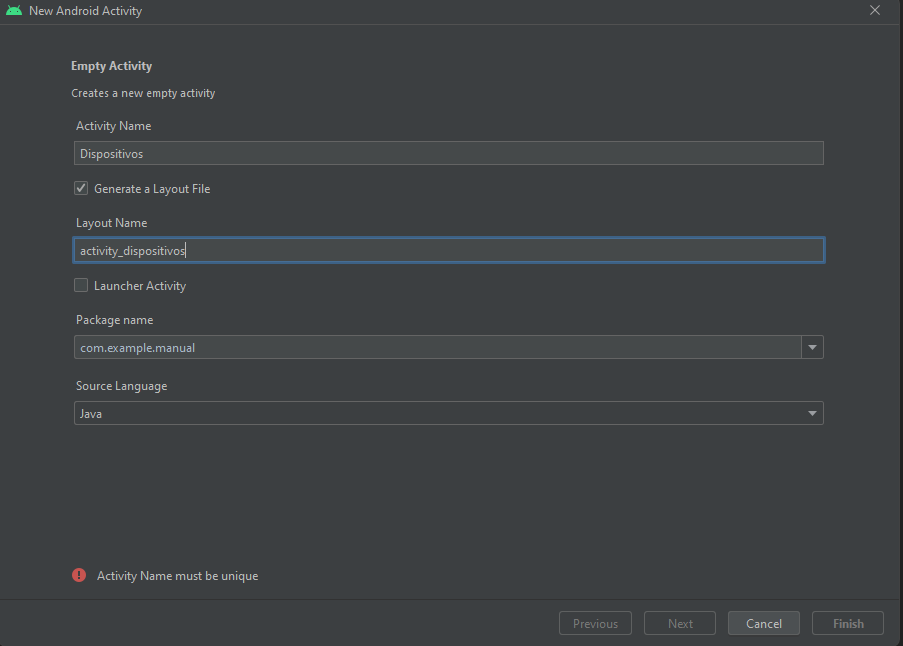
 

Y con todo esto, el resultado final debería verse así.



Para crear la otra pagina hay que crear una nueva “activity”, para esto se ingresa en “File”–> “New”–> “Activity”–> “Empty Activity”.

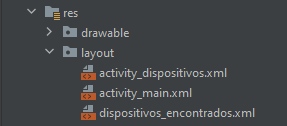


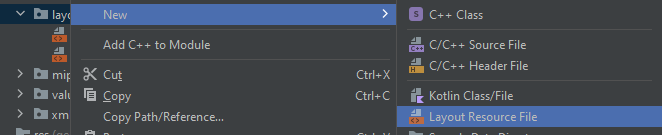
Se cambia el nombre a Dispositivos y se presiona en “Finish”.

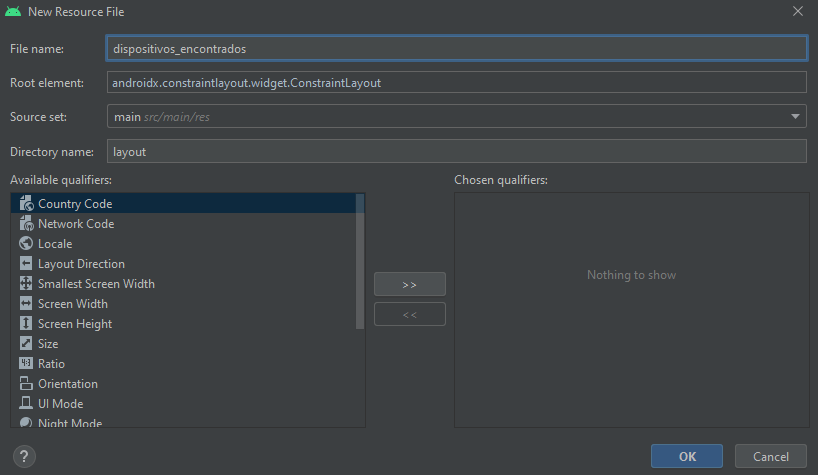
Esto creara un nuevo archivo .java para el código y uno .xml para la interfaz gráfica.

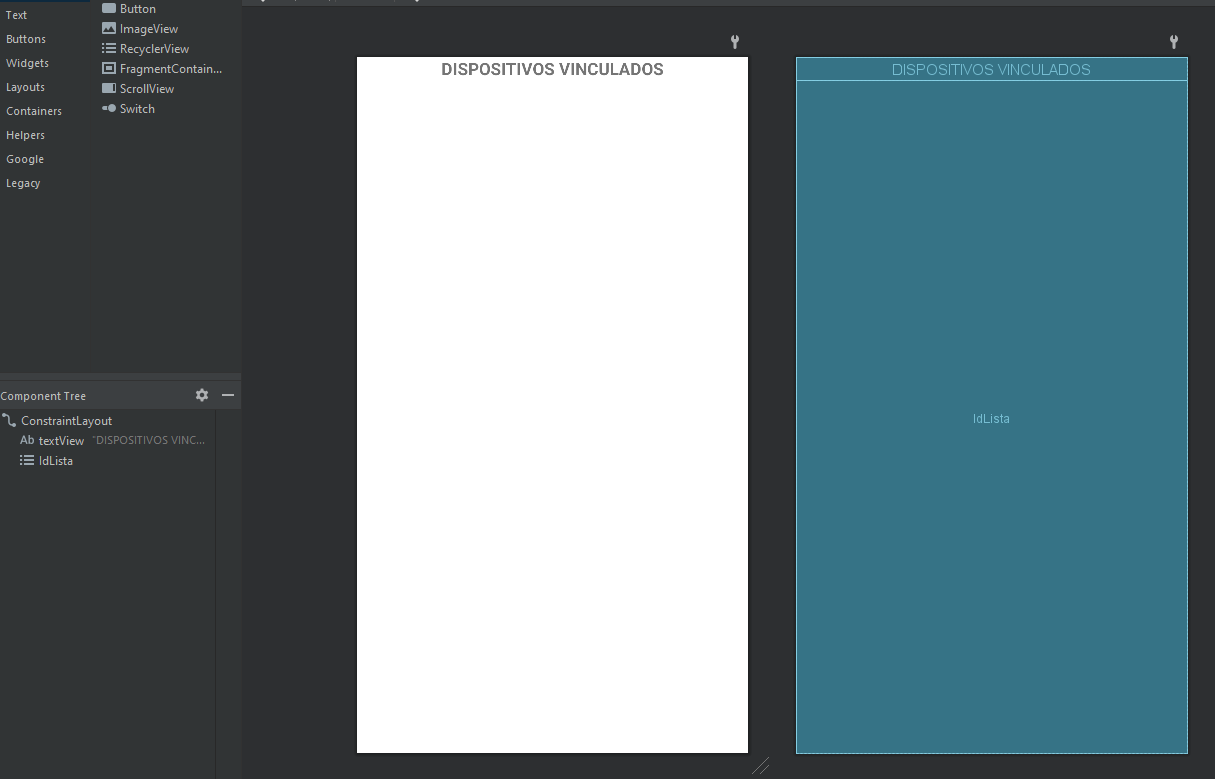


Para acceder a las interfaces graficas de cada página hay que dirigirse a la carpeta “res” y luego en “layout”, en esta se encontraran tres archivos .xml, los cuales corresponden a la página principal, y otras dos: “activity\_dispositivos” y “dispositivos\_encontrados”, siendo estos dos últimos partes de la página de “dispositivos”.

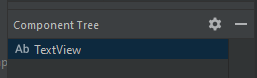
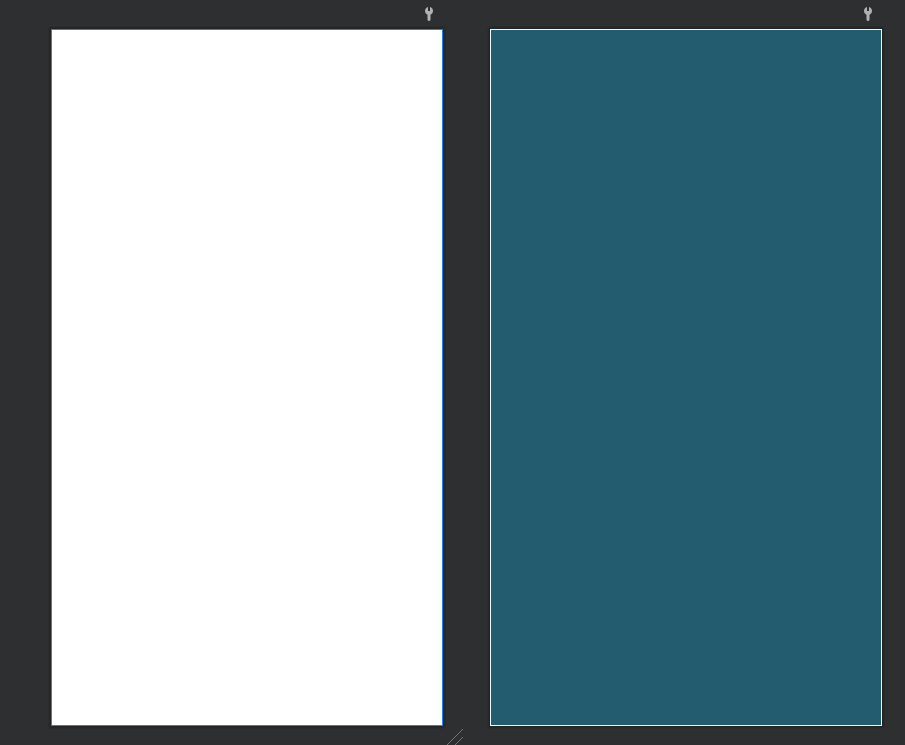


 Para crear el archivo .xml de “dispositivos\_encontrados” hay que dirigirse a la carpeta layout, y aquí crear un nuevo archivo “Layout Resource File”.

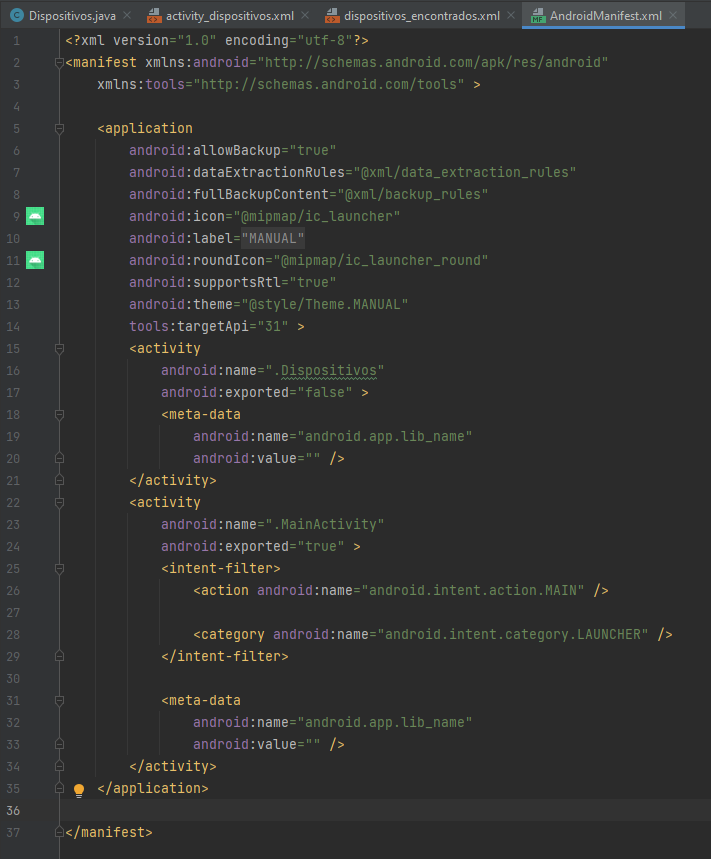
 Se le cambia el nombre por el antes mencionado y se presiona en OK.

 La página de “activity\_dispositivos”, esta consta esencialmente de dos cosas, un TextView con el nombre de la página, y un ListView, el cual es una lista de elementos, los cuales provienen de “dispositivos\_encontrados”.

Siguiendo con la página de “dispositivos\_encontrados”, en esta simplemente se reemplaza el ConstraintLayout por un TextView



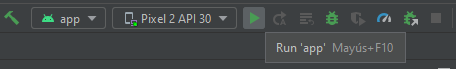
Las pagina de Dispositivos se añadira automaticamente en el Manifest.



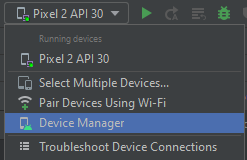
* 1. **Despliegue de Programa**

Para cargar el programa, se puede hacer desde un dispostivo movil real o desde un dispositivo virtual proporcionado por el progama de Android Studio.

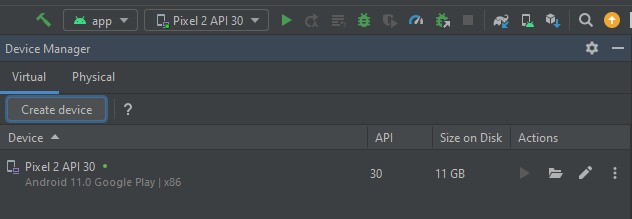
En el caso de usar un telefono movil, basta con conectarlo al computador con un cable USB, una vez aparesca conectado, se le da al boton “Run”



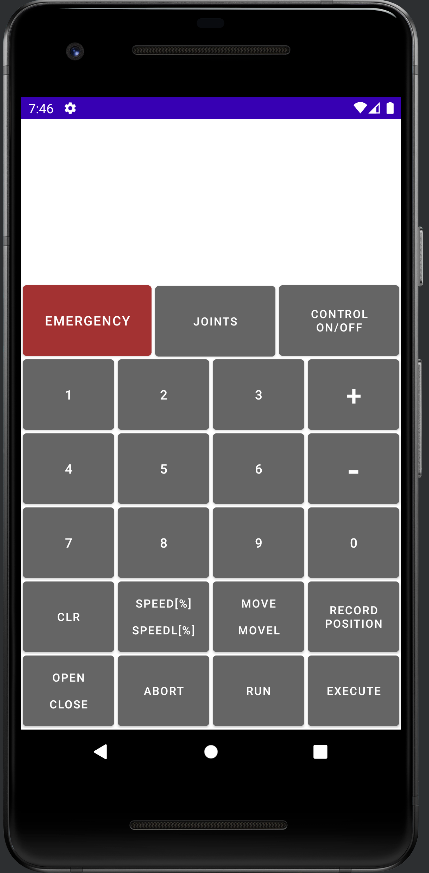
En cambio, si se desea usar un dispositivo virtual, primer se debe comprobar si ya existe uno instalado, en este caso es el dispositivo “Pixel 2 API 30”. Si no existen dispositivos, se ingresa en “Device Manager”.



Dentro de este apartado se preciona el boton de “Create device” y se continuan con los pasos que indica Android Studio para crear un dispositivo nuevo.

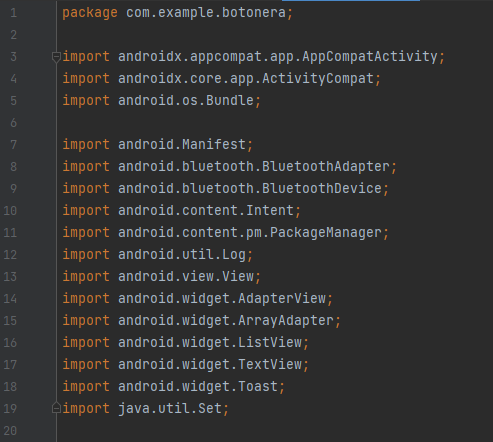


Ya con el dispositivo creado, se selecciona y se preciona en Run, el programa se cargara en el dispositivo y en caso de haber algun error, el mismo programa de Android Stuido indicara el poblema

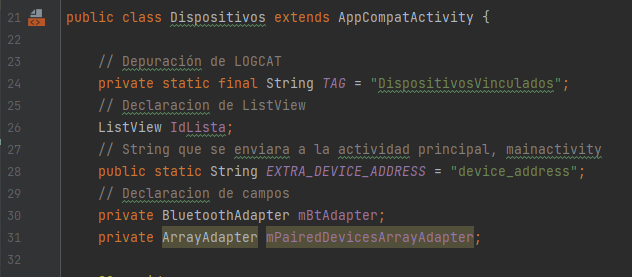


1. **Código del programa:**
   1. **Dispositivos:**

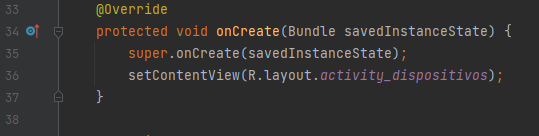
Para comenzar con el codigo del programa, lo primero es dirigirse al archivo “Dispositivos.java”, en este solo se encontrara una funcion “onCreate”, la cual viene por defecto al crear una nueva activity, y que se ejecuta al iniciar el programa. Para que el programa funcione correctamente, se deben importar las declaraciones que permitiran utilizar ciertas clases y funciones, a continuación se encuentran todas las importaciones que necesitara esta parte del programa, estas podria añadirse automaticamente al desarrollar el resto del programa, pero si alguna no se incluyo es necesario añadirla manualmente.



Dentro de la función principal se declaran las variables y datos que se utilizaran. BluetoothAdapter es una clase proporcionada por Android que permite realizar operaciones realacionadas con Bluetooth, y ArrayAdapter se utiliza para adaptar matrices o listas de datos a vistas.



La funció “onCreate” no es necesario modificarla. La anotación “Override” sirve para sobreescribir un metodo, lo cual ayuda a evitar errores tipografivos.



En la función “onResume” se comprueba el estado del Bluetooth llamando a la funcion “VerificarEstadoBT()”. “mPairedDevicesArrayAdapter” Se usa para listar cada dispositivos vinculados al movil dentro del layout “dispositivos\_encontrados”, creado anteriormente, por cual, en esta lista apareceran todos los elementos visuales que se hayan añadido “dispositivos\_encontrados”.

“IdLista” se asigna al objeto con el id del mismo nombre, el cual corresponde al elemento ListView de la interfaz grafica, luego se establece el adaptador “mPairedDevicesArrayAdapter” a la lista y un “listener” (“setOnItemClickListener”) para manejar los click de los elementos en la lista. La logica de ejecución al momento de hacer un click en uno de elementos de la lista se encuentra en el listener “mDeviceClickListener”.

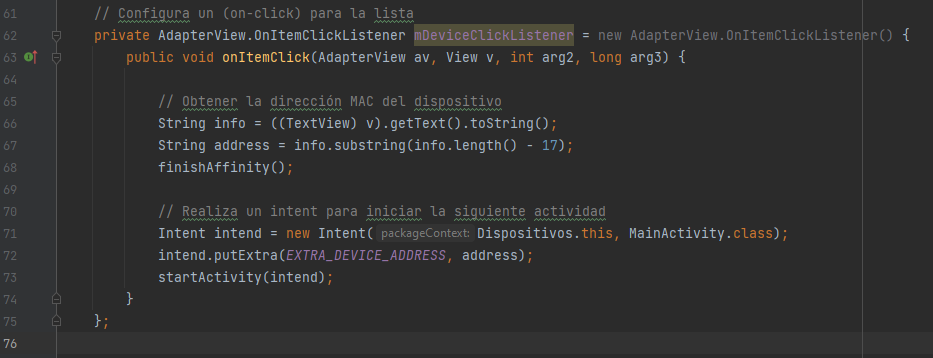
El adaptador “mBtAdapter” se establece con su configuracion predetermianda, luego se comprueba si el programa tiene permisos para conectarse a dispositivos Bluetooth, si los posee se obtiene la lista de dispositivos vinculados, y encaso de haber alguno, se recorre la lista de estos dispositivos y se agregan sus respectivos nombres y direcciones dentro del adaptador “mPairedDevicesArrayAdapter” para mostrarse en el ListView.



Para el listener, primero se crea un objeto de tipo “AdapterView.OnItemClickListener”, dentro de este se añade un metodo que es llamado cuando se da click en uno de los elementos de la lista. Este metodo obtiene el AdapterView que contiene la lista de “IdLista” (“av”), la vista del elemento que fue clickeado (“v”), la posicion del elemento dentro de la lista (“arg2”) y el ID del elemento (“arg3”).

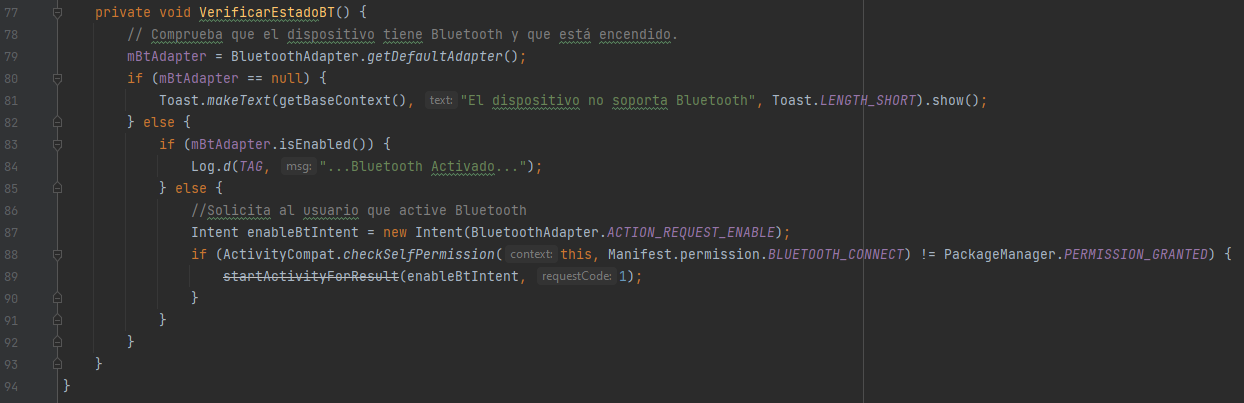
Con un String (“info”), se obtiene el texto del elemento, el cual contiene texto del elemento, el cual contiene el nombre del dispositivo y su direccion MAC, esta ultima se guarda en un nuevo String (“address2), recortando los ultimos 17 caracteres de todo el texto, correspondiente a la direccion MAC.

“finishAffinity()” es un metodo de la clase “Activity”, con este se cierran todas las actividades de la aplicación, luego se crea un nuevo objeto “Intent” para iniciar con este el activity “MainActivity”. A este Intent se le añades los String “EXTRA\_DEVICE\_ADDRESS” y la direccion MAC contenida en “address” como un extra, por ultimo, con “startActivity(intend)” se inicia el MainActivity.



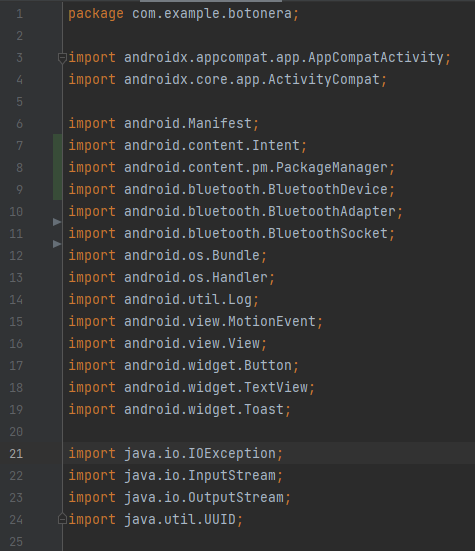
Por ultimo la función VerificarEstadoBT, para esta, en primer lugar se obtiene el adaptador Bluetooth predeterminado, el cual es esencial para interactuar con las funcionalidades Bluetooth del dispositivo. Luego se verifica si el dispositivo soporta Bluetooth, en caso de que “mBtAdapter” sea nulo, significa que el dispositivo no lo hace y se indica esto al usuario con un mensaje.

Por el contrario, si el dispositivo tiene soporte para Bluetooth se verifica si este se encuentra encendido, si no lo está, por medio de un Intent se solicita al usuario que lo active. Antes de lanzar este Intent, se verifica si los permisos de conexión Bluetooth se encuentran activados, si no lo están se solicita, pero si se encuentran activos entonces el programa lanza en Intent por medio del metodo “startActivityForResult”

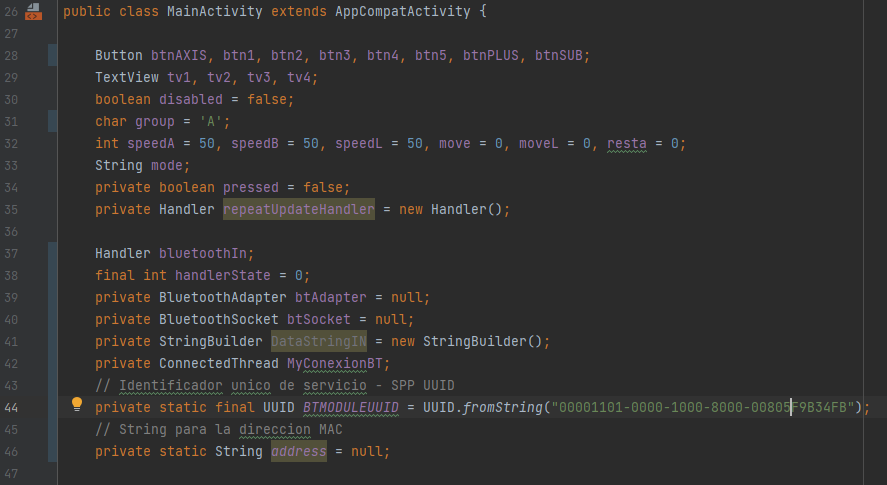


* 1. **Main Activity**

Para el desarrollo de la página principal lo primero es importar las clases y paquetes necesarios para que esta funcione correctamente, esta pueden importarse automaticamente a medida que se avance con el programa, pero si alguna no aparece es necesario escribirla manualmente.



Dentro de la funcion principal se declaran y defininen las variables y objetos que se utilizaran, algunos de estos corresponden a los componentes de la interfaz grafica, como los botones o bloques de texto, pueden ser usados dentro del programa haciendo una referencia a su Id. Para este caso, el Id “btnAXIS” corresponde al boton “JOINTS”, el Id “btnPLUS” al boton “+”, el Id “btnSUB” al boton “-“ y el resto de Ids a los botones con su respectivo número, lo mismo pasa con los ID “tv”, estos corresponden a su respectivo TextView, empezando desde el que se encuentra mas arriba como tv1, hasta el de mas abajo como tv4.



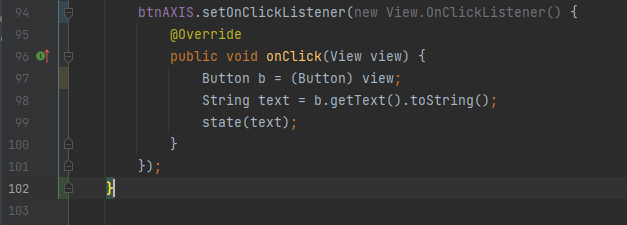
A continuación se trabajara dentro de la funcion “onCreate”, la cual viene ya incluida en el codigo al momento de crear una activity nueva. Aquí se encontrá un handler que ayudará al envio de datos al ESP32 via Bluetooth, pero de esta se hablará mas adelante.



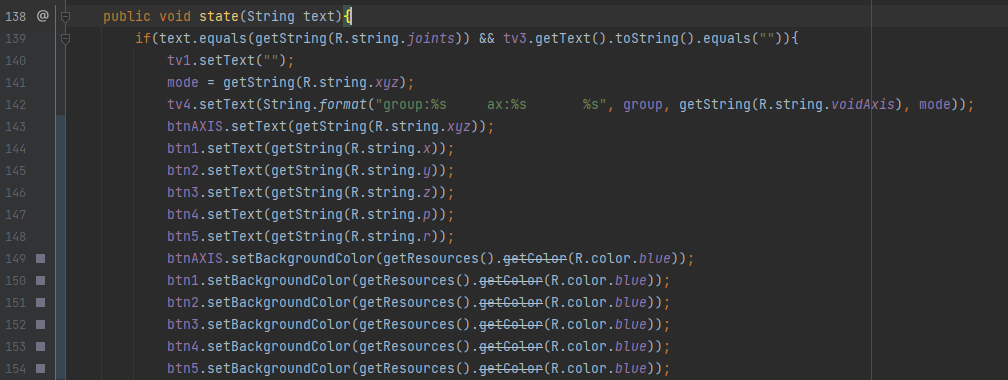
Un poco mas abajo del handler se obtiene el adaptador Bluetooth predeterminado del dispositivo y se llama al metodo “VerificarEstadoBT” para comprobar si el Bluetooth del dispsitivo se encuntra encendido o no. Este metodo tambien se mencionará mas adelante. Luego de esto se configuran los botones y TextViews mediante “findViewById”, para hacer referencia al Id de los respectivos objetos dentro de la interfaz grafica. Al String “mode” se le asigna el texto “JOINTS”, este texto es el que se encuentra en el archivo “strings.xml” de la carpeta “values” vista anteriormente.

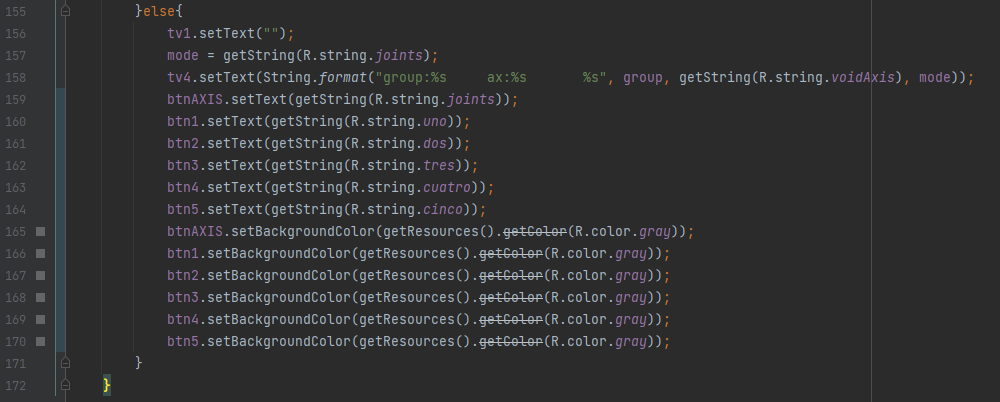


Por ultimo, se crea un metodo, el cual se ejecuta al presionar el boton “JOINTS”, de Id “btnAXIS”, y que extrae lo que hay escrito en el boton y lo envia a una funcion llamada “state”. Con esto se cierra la función “onCreate”.



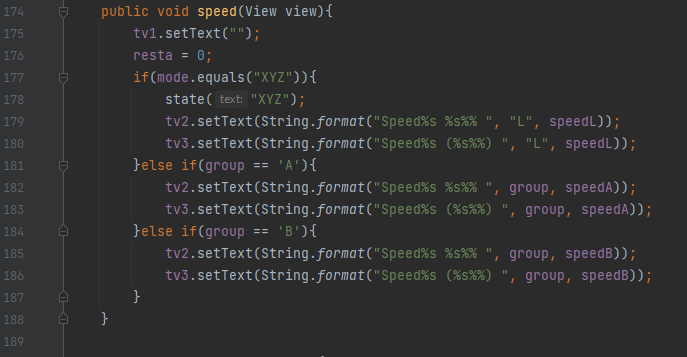
Continuando con el programa, aun dentro de la función principal, se crea la funcion state, la función de esta es cambiar el modo de la botonera, de “Joints” a “XYZ”, si detecta que el modo actual es “Joints” lo cambia a “XYZ” y viceversa. Ademas de cambiar el modo, tambien cambian los botones del 1 al 5 por los botones X, Y, Z, P y R respectivamente, al igual que el boton “JOINTS” por el boton “XYZ”, el cual cumple la misma funcion si es presionado. Tambien se cambia ligeramente el color de estos en modo XYZ para diferenciar cuales fueron los botones cambiados. Todo esto es meramente una ayuda visual y puede ser modificado si se requiere.



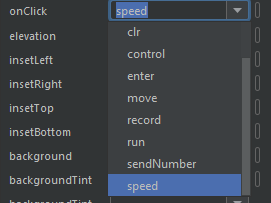


Lo anterior se muesta a traves del tv4. El TextView tv4 muestra unicamente el grupo y modo actual, asi como el axis que esta activo, el grupo fue definido en la declaracion de variables como “A”, por lo cual al inicial la aplicación empezara en este grupo, mientras que el modo inicial el Joints, declarado anteriormente en la funcion onCreate. Como para este punto aun no se ha presionado ningun boton numerico, el axis muestra un guion como señal de vacio.

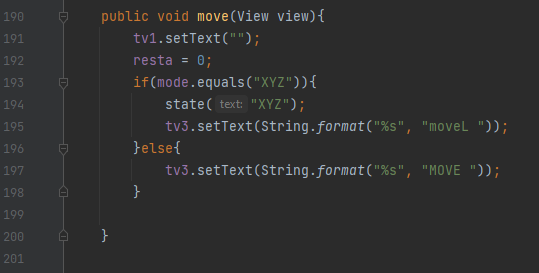
La funcion “speed” ajusta las velocidades de Scorbot. Existen 3 tipos de velicidades, “SpeedA” correspondiente al grupo A, “SpeedB” correspondiente al grupo B, y “SpeedL” que es la velocidad para los Axis de XYZ. Al precionar el boton Speed se escribira en el tv3 el tipo correspondiente de speed que se modificara según se cumpla alguna de las condiciones antes menconadas. En este punto, los botones numericos dejaran de funcionar para seleccionar los axis y pasaran a modo de escritura para poder ingresar el porcentaje de velocidad que se desea usar. Este modo de escritura se vera en la funcion correspondiente a los botones numeros.

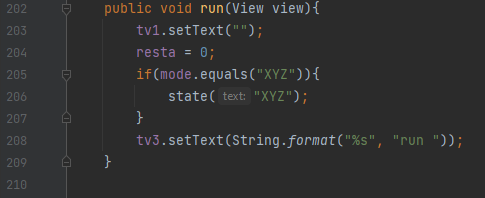


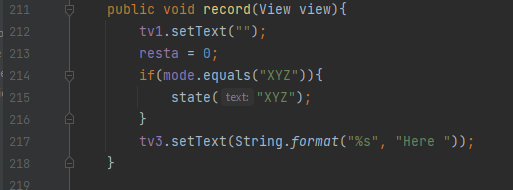
Finalmente, se asigna esta funcion al boton de speed. Para asignar una función a un boton se debe ir a la interfaz grafica y presionar el boton al cual se planea asignar dicha función, luego se busca el atributo “onClick” y se seleciona la función correspondiente, es importante mencionar que esta debe ser publica o no aparecerá aquí. El atributo “onClick”, como su nombre indica, ejecuta la función asignada al boton con dicho atributo cada vez que este es presionado.

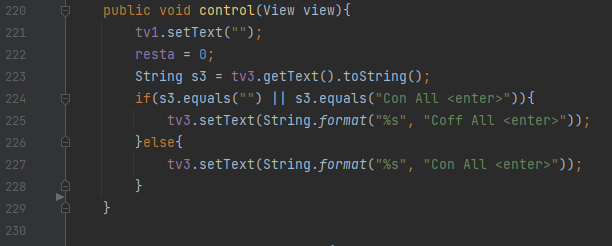


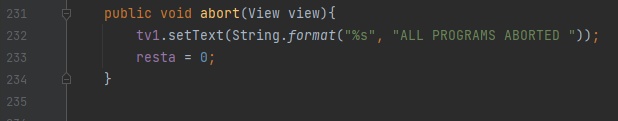
Al igual que speed, move tiene dos modos, siendo “MOVE” el estado normal para el modo joints, y “moveL” el estado del modo “XYZ”. Las siguientes funciones actuan de forma similar y deben ser asignadas a su respectivo boton.





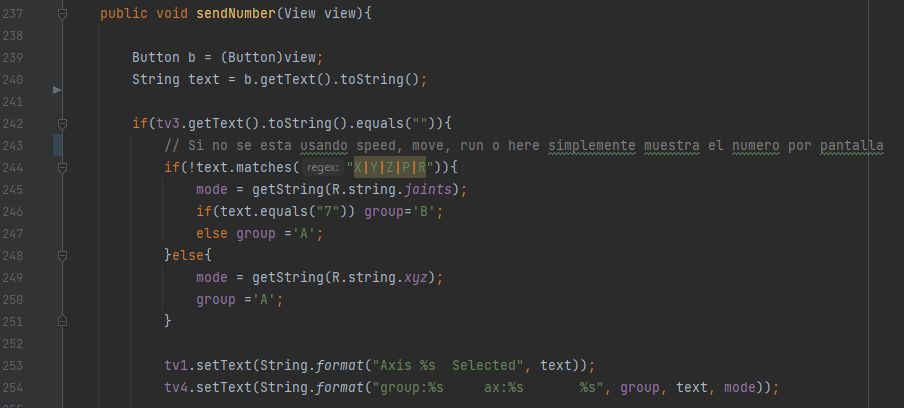




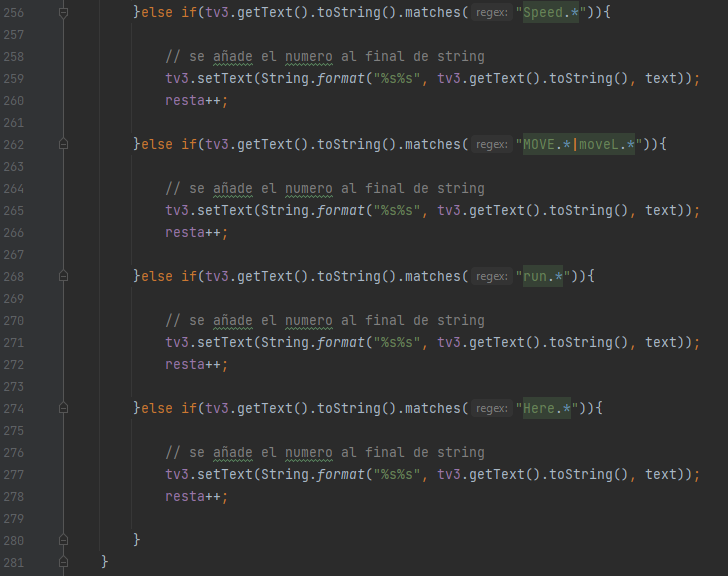


Para las fuciones “speed”, “move”, “run”, “record”, se usa la funcion “state” para cambiar a modo “joinst” si es que se el modo se encuentra en “xyz” y asi evitar errores, los botones numericos dejan de funcionar los axis y pasan a ser botones de escritura, y el mensaje escrito se envia por medio de otra función, la cual junto a la relacionada a los botones se veran a continuación.

La función “sendNumber” es la encargada del funcionamiento de los botones. Primeramente extrae el texto correspondiente al boton que se presiono y lo aguarda en un String, luego verifica si tv3 esta vacio o no, en caso de estarlo, se cambia al modo correspondiente al boton (joint si es numero, XYZ si es letra) y el grupo, el numero 7 corresponde al grupo B, el resto al grupo A. El resultado se muestra en pantalla, en tv1 el axis selecionado, y en tv4 la información general respecto al grupo, axis y modo.

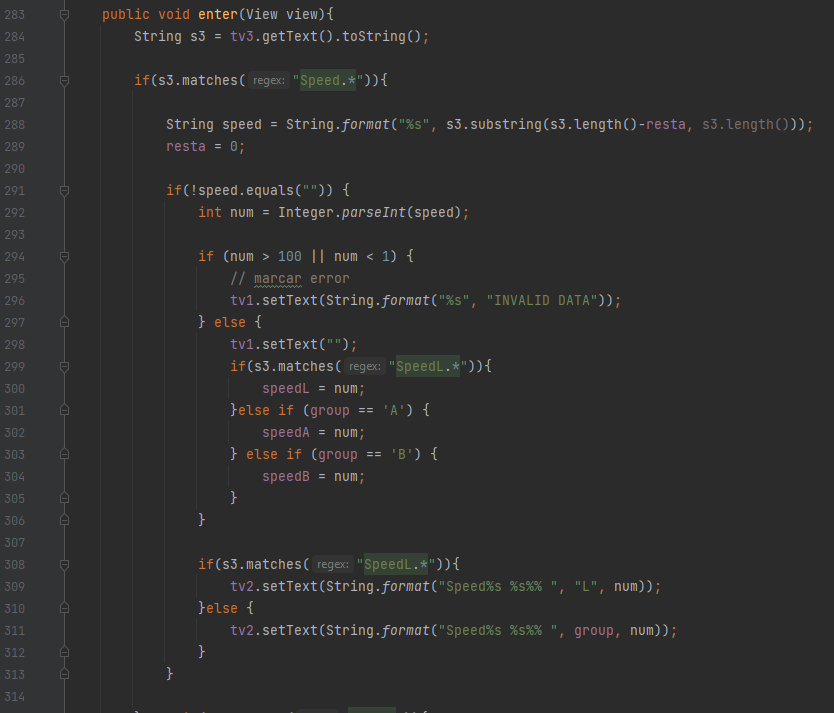


En caso de que tv3 no se encuentra vacio como resultado de haber apretado los botones speed, move, run o record, el numero guardado en el String text se añade al final del String en tv3 y se aumenta el contador “resta”, el cual comienza en 0 al momento de ingresar el primer número. Este contador es importante para la parte de envio. Cabe decir que para estos 4 casos no puede ingresar una letra, solo un número, esto gracias a que en cada una de estas funciones se cambio el modo a “joints” antes de continuar. Lo ultimo por hacer es asiganar esta función a todos los botones numericos.

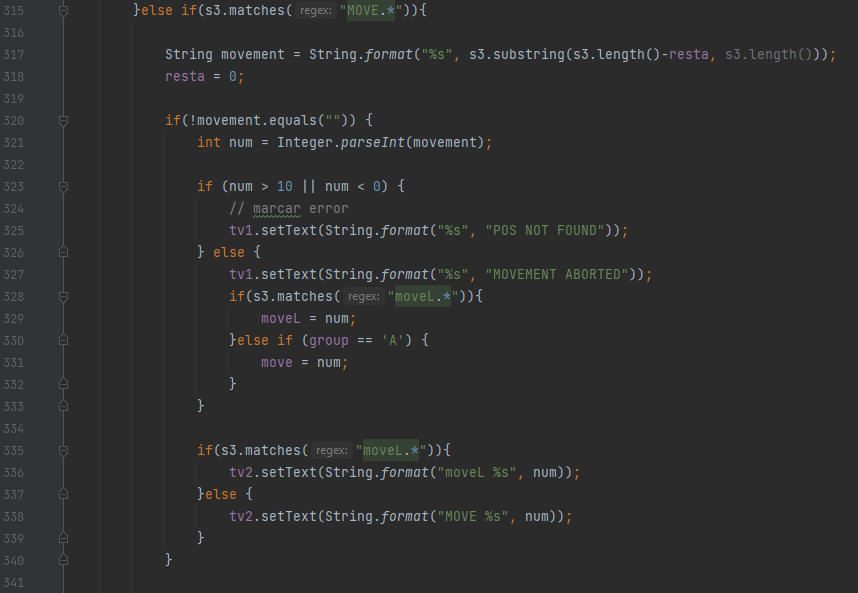


Pasando a la funcion “enter” correspondiente a la funcion de envio, esta guarda en un String “s3” lo que encuentre en tv3. Lo primero que verifica es si en este cuadro de texto se encontraba la palabra “Speed”, si encuentra esta palabra, recorta solo los números ingresados al String s3, ayudandose del contador “resta” para saber la cantidad de números que fueron ingresados y rescortando esa cantidad al final de “s3”.

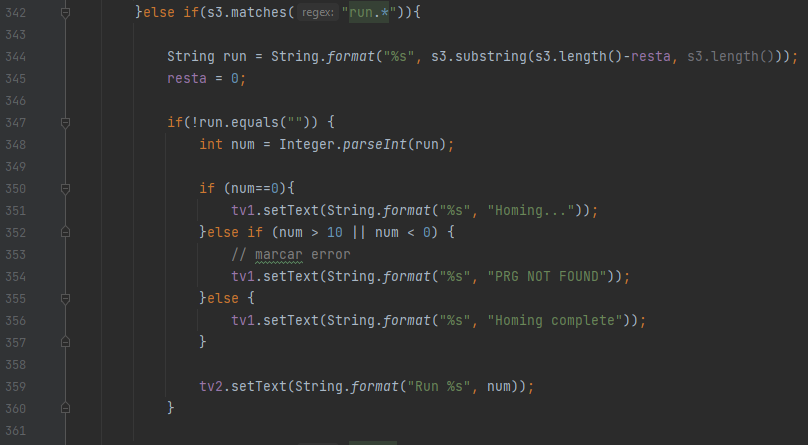
Si el String en el cual se guardaron los números recortados no se encuentra vacio (es decir, hay uno o mas numeros en él), transforma el String a un valor numerico entero y verifica si este numero es mayor que 100 o menor que 1, en caso de serlo, indica el error a traves de tv1 (recordar que la velocidad no puede ser inferior al 1% ni exceder el 100%). Si el número se encuntra dentro de este rango asigna el numero a su respectivo tipo de velocidad, dependiendo la condiciendo que se cumpla, y muestra por tv2 el correspondiente resultado.



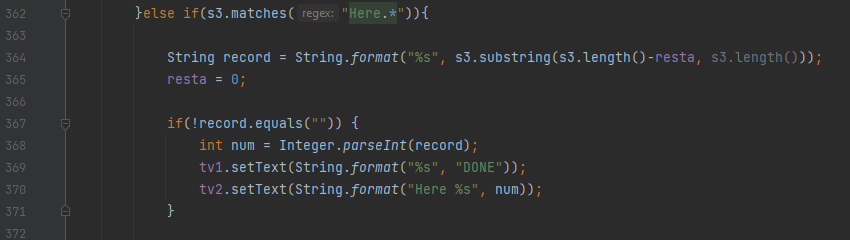
Para el caso de “MOVE” actua de forma similar, asignando el numero a su respectiva variable “move”, con la diferencia de que en este caso el número no puede ser mayor que 10 no menor que 1. El String “MOVEMENT ABORTED” se usa momentaneamente como apoyo visual.



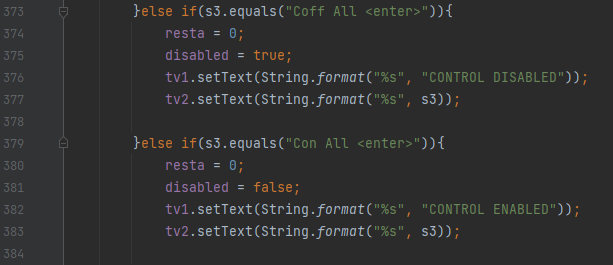
Para el cado de “run”, el numero debe estar en un rango de 0 a 10, si se pasa de este rango muestra un error por tv1, en caso contrario muestra la instrucción que esta realizando (solo esta implementado el caso de 0).



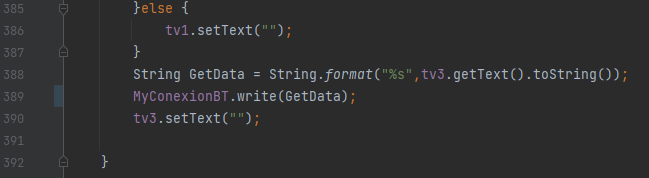
Para el caso de “here” solo muestra en tv1 que se realizó, y en tv2 el comando ingresado.



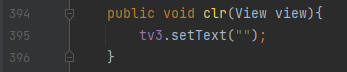
En los casos de “Coff All” y “Con All” se utiliza la variable booleana “disabled” para activar o discartivar los botones, y se indica en tv1 la acción realiada.



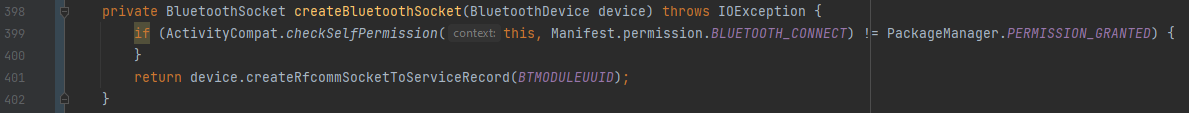
Si no se cumple ninguna de estas condiciones, se limpia el contenido de tv1. Por ultimo se guarda el contenido de tv3 en un String, este se envia al ESP32 via Bluetooth y se limpia el contenido de tv3. Esta función se debe asignar el boton “Execute”.



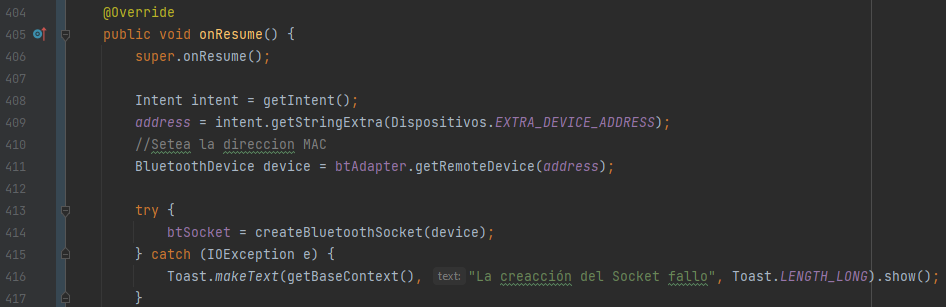
La funcion “clr” simplemente limpia el contenid de tv3.



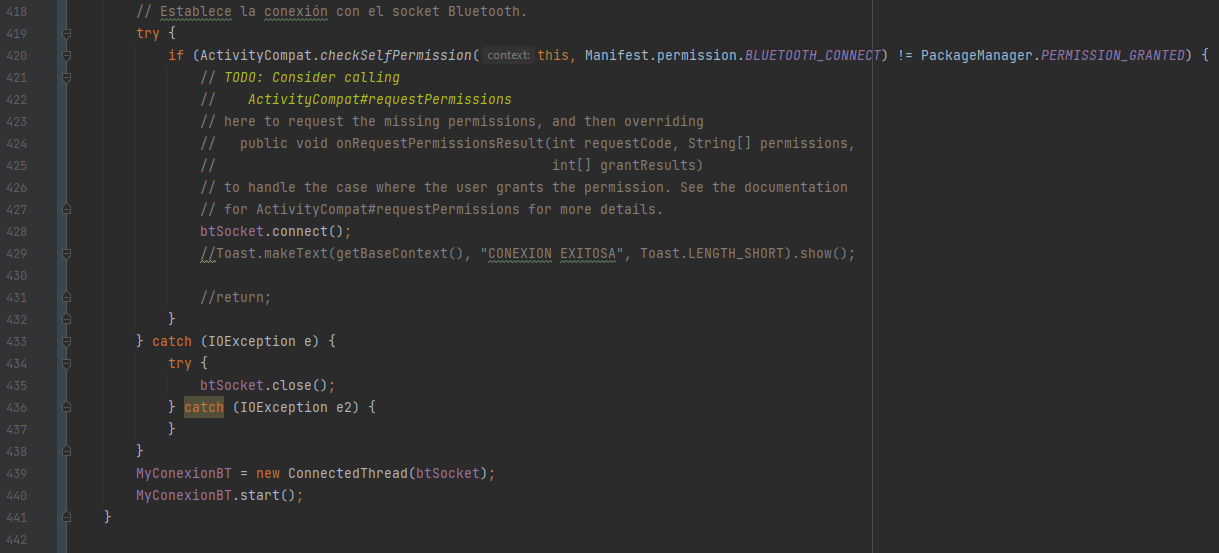
createBluetoothSocket toma un dispositivo Bluetooth (BluetoothDevice) y devuelve un socket Bluetooth (BluetoothSocket). Dentro de este se verifica si al programa se le asignaron los permisos Bluetooth correspondietes y luego crea y retorna un socket Bluetooth con el uuid BTMODULEUUID.



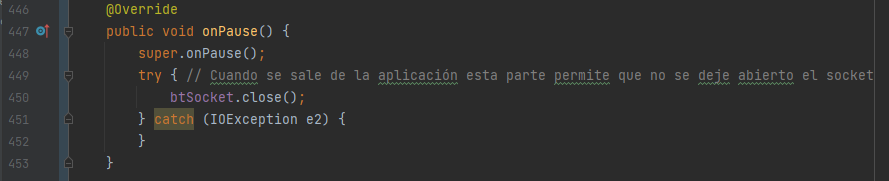
onResume se activa cuando se continua la ejecucion del programa una vez este haya sido pausado. En este, se recupera el Intent que inicio la activity, se guarda la direccion MAC del dispositivo Bluetooth anteriormente conectado y se crea un objeto de la clase BluetoothDevice a partir de la direccion MAC. Luego se intenta crear un socket llamando al metodo createBluetoothDevice, si no se logra, se envia un mensaje indicando esto.



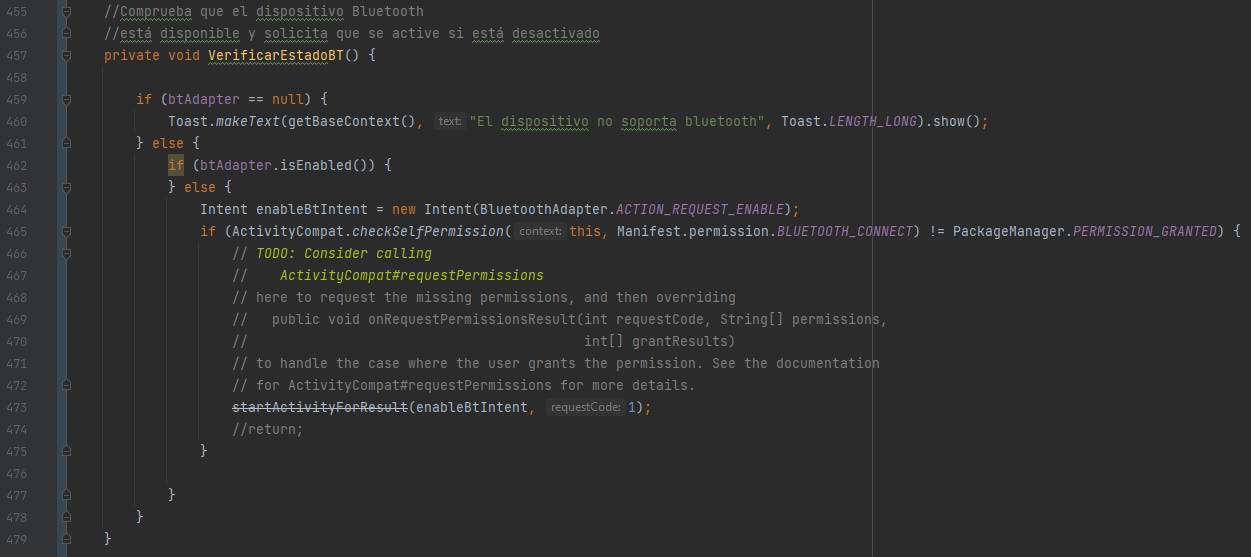
Se verifica si se poseen los permisos de Bluetooth, y se intenta establecer la conexión Bluetooth llamando al metodo “btSocket.connect()”, si no se consigue se cierra el socket. Una vez establecida la comunicación Bluetooth exitosamente, con “ConnectedThread” se crea un hilo dedicado a la comunicación continua entre los dispostivos, y se inicia mediante el metodo “MyConexionBT.start()”.



Cuando la activity esta a punto de ser pausada o se vuelve invisible se activa el metodo “onPause”, el cual intenta cerrar el socket para liberar los recursos relacionado con la conexión Bluetooth antes de que la activity se pause.



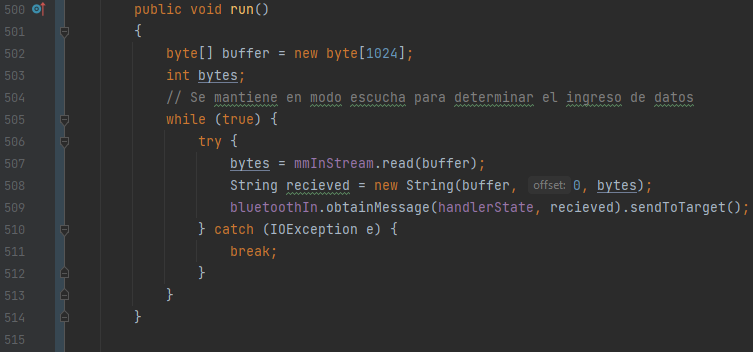
El metodo “VerificarEstadoBT” comprueba primeramente si el adaptador Bluetooth es nulo, de ser el caso, muestra un mensaje indicando que el dispositivo no sopota Bluetooth. Si no es nulo, verifica si el adapatador esta habilitado, si no lo esta, mediando un Intent se solicita al usuario que lo active.



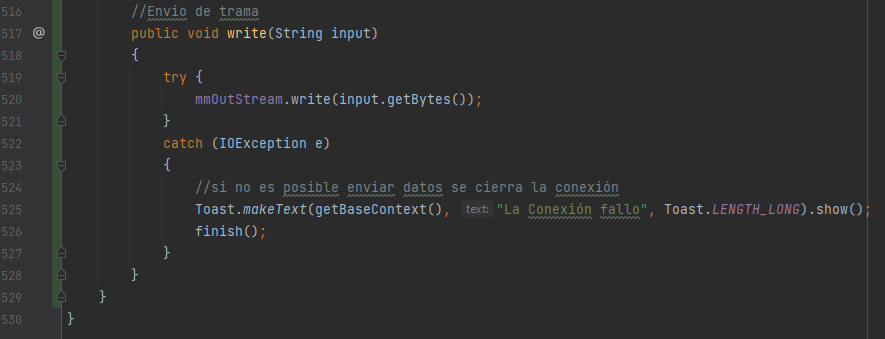
ConnectedThread es una clase interna del MainActivity que maneja la comunicación Bluetooth asincrona. “mmInStrream” y “mmOutStream” son objetos usados para la transmision de entrada y salida de datos. El constructor con el mismo nombre de la clase, toma un socket Bluetooth y obtiene sus flujo de entrada y salida.



El metodo “run” se ejecuta cuando se inicia ConnectedThread, este consta de un ciclo infinita que lee y almacena datos desde el flujo de entrada para enviarlos al handler a modo de cadena.



Por ultimo, el metodo “write” permite enviar datos a traves del socket Bluetooth, tomando un String e intentando covertirlo en una matriz de bytes, luego enviarla al ESP32



Retomando el Handler de la funcion “onCreate”, este se utiliza para manejar los mensajes provenientes del hilo “ConnectedThread”, los cuales vienen del ESP32, se extrae el contenido de estos y se muestran en el tv1.



* 1. **ESP32:**

A