# CSE2MAD Lab 5 – BLUETOOTH SCANNER with ALERT DIALOG

AIMS

Today will be making a Bluetooth Scanner app we will be exploring;

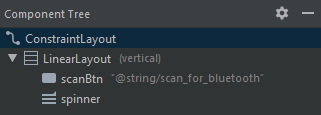
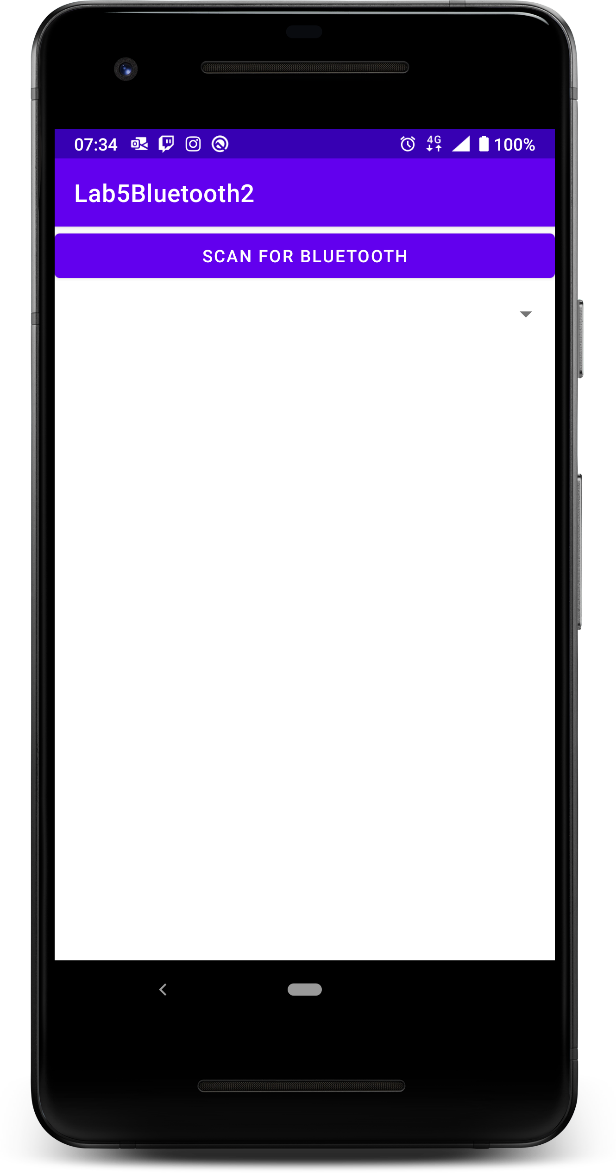
* The BluetoothAdapter class
* Dangerous permissions at Runtime
* Alert Dialogs

We will be revisiting various concepts learned through the semester including;

* Broadcast receivers
* ArrayAdapters

Step 1 – Making the Main ACTivity ui & REQUESTING PERMISSIONS

1. Create a new project & choose the empty activity template. (See week 1 lab)
2. Create the UI in the visual DESIGN view for the main activity as in Figure 1. If you need a hint see the component tree in Figure 2 and layout code from previous labs. You may get errors, just accept new minimum spinner height and ignore anything else.



It’s a little tricky to see, but there is a ‘spinner’ here (dropdown)

Fig 1 Fig 2

1. Now set up the member variables for your button & spinner and binding them to your view objects, and add the button onClick Listener

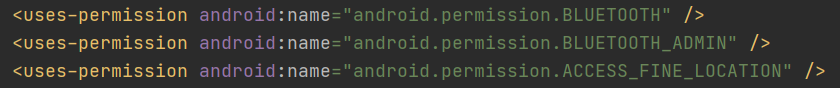
**private** Spinner **resultsSpinner**;  
**private** Button **scanBtn**;

**private** String **TAG** = **"MainAct"**;  
  
@Override  
**protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 setContentView(R.layout.***activity\_main***);  
   
 *//Spinner* **resultsSpinner** = (Spinner) findViewById(R.id.***spinner***);  
  
 *//Button* **scanBtn** = (Button) findViewById(R.id.***scanBtn***);  
 **scanBtn**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {

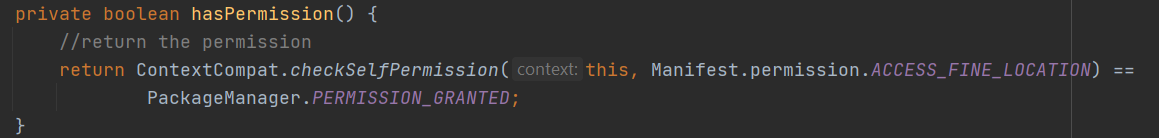
//Do something on button click

}  
 });  
}

1. Add our permissions required to use Bluetooth to the normal place, the AndroidManifest.xml just above the application tag.



1. Now we need to check permissions. As per this weeks lecture, ACCESS\_FINE\_LOCATION is considered a ‘Dangerous permission” and a request must be made to the user DURING RUNTIME. This is the same process we used in Lab 2 except this time we are just requesting one permission instead of an array of permissions. Make a new class under the onCreate called hasPermission();



1. So now we know if the app already has required permissions but what if it doesn’t? Well then we wil need to ask permissions 😊 Create the ActivityResultLauncher outside your onCreate(), this is a callback method that handles the results of the permissions query. It is the new recommended way to request permissions. For more info view <https://developer.android.com/training/permissions/requesting> If you have troubles reading the image below please use [this gist](https://gist.github.com/latrobe-cs-educator/114dd0cc0285864d07d197ec5ec6ec7a).

Text

Description automatically generated

1. But how do we launch the launcher? Well we want it to run ONLY if permission have not already been granted so we can make an if statement in the onCreate() as we definitely want it to run at least once. Here we are using the hasPermission() function we created earlier to check permission has been granted, if not we make the button inactive as it wouldn’t work anyway and launch the requestpermissionlauncher passing in the permission we are asking for.

Text

Description automatically generated

Now run your app, you should be asked permission on your first run like below;

Graphical user interface, application, Teams

Description automatically generated

Now we have finally finished the setup we can start working on the scan functionality. Yay!

Step 2 – SCAN FOR BLUETOOTH DEVICES

1. As we are dealing with Bluetooth add a Bluetooth member variable



1. And initalise it inside the onCreate()

A screenshot of a computer

Description automatically generated with low confidence

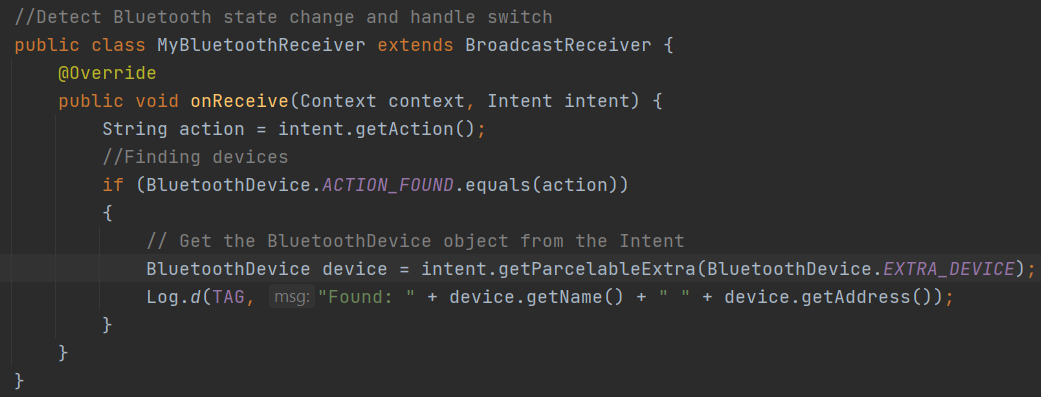
1. Now you can put the call to BA.startDiscovery() inside your button onClick() to start looking for bluetooth discoverable devices.

NOTE: Please view the startDiscovery() documentation here, as we are targeting Android 11 (API 30 Build code R ) we have the correct permissions, however for builds after Android 11 there we be some permission changes. Google is great at this, I cannot encourage you enough to read the docs for functions as you code as things change all the time!

Text

Description automatically generated

1. When as device has been located an "android.bluetooth.device.action.FOUND" broadcast will be generated, so we will need a broadcast receiver, and as we want to interact easily with the activity we will make it a subclass inside the MainActivity like we did last week.



1. And of course as we have a Broadcast Receiver we need an………..? Intent filter yes and to register our Broadcast receiver. First we must declare the Broadcast receiver with the rest of our member variables

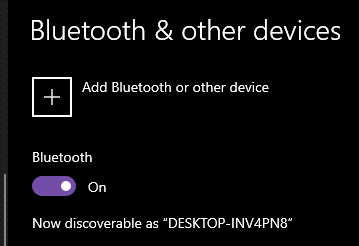


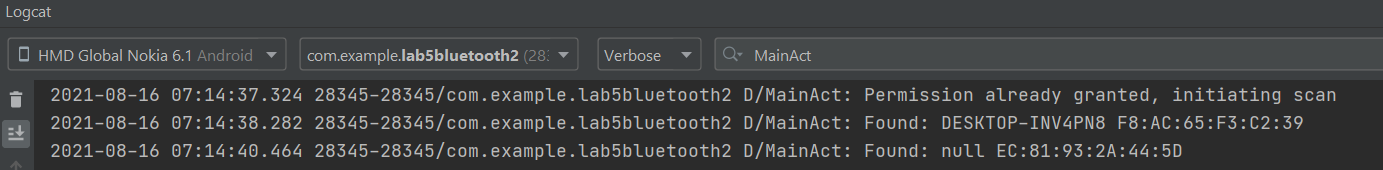
1. And add the filter and registerReceiver method into the onCreate() in the MainActivity.

Text

Description automatically generated

Now you should be able to detect discoverable Bluetooth devices BUT they will need to be set as discoverable. The devices will appear in your logcat. Not all devices will have a name, they may only return the address





Step 3 – SET UP SPINNER & ARRAY ADAPTER

1. Now that we are getting results it would be nice to display them somewhere.. a spinner perhaps.. Thus far we have linked the spinner object to the View in the onCreate() but some additional preparation is required. Add the following code into the onCreate() under the linked spinner object. Here we are making an ArrayList to store the information we have received. Next we create an arrayAdapter where we specify the xml of the spinner and the list we created earlier. Then we set the adapter to the results spinner.

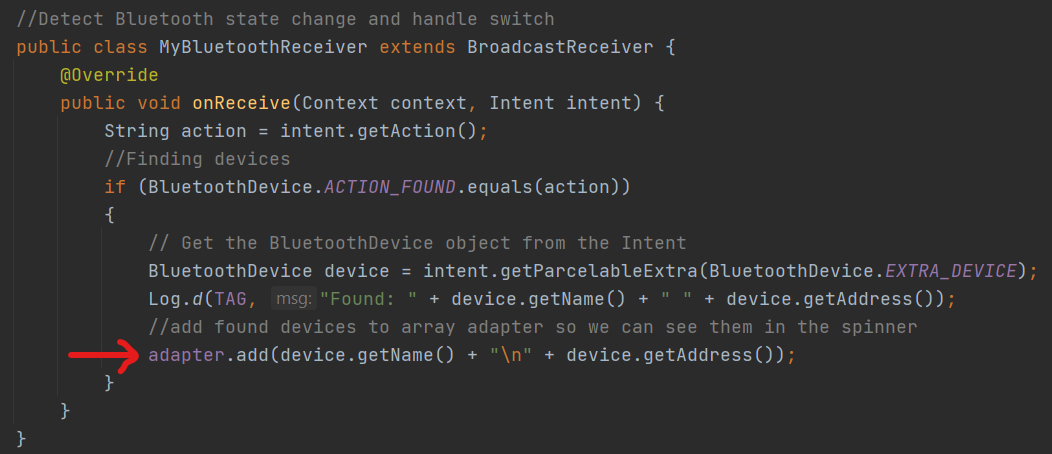
Text

Description automatically generated

1. We will also need to add the adapter as a member variable!



1. Now in the Broadcast receiver we have to add the devices to the arraylist adapter as they are found, this will propagate the spinner.

****

1. And lastly we must unregister the broadcast receiver in the onDestroy()

Text

Description automatically generated

1. Now you should have a functioning app ready to scan all your devices 😊

A picture containing text, monitor, electronics, screenshot

Description automatically generated

# CSE2MAD Lab 5 – CHECK FOR BLUETOOTH

Logically if an app is scanning for Bluetooth devices, it would be wise to have Bluetooth on so we can connect to them.

1. We should do this as soon as they press the button to scan so we should use a conditional statement in the onClick, as we don’t want to activate the Bluetooth when the app is just opened as to not waste the battery. Though its really up to you 😊 In this instance we will check the state of the Bluetooth using the isEnabled method of the Bluetooth Adapter class, read more here

<https://developer.android.com/reference/android/bluetooth/BluetoothAdapter#isEnabled()>

Text

Description automatically generated

1. Hmm so what if they don’t have it enabled? It’s a bit rude to just enable things without asking so we can make an Alert Dialog. Dialogs as very useful tools which are quite simple to build. Please read more about Dialogs here

<https://developer.android.com/guide/topics/ui/dialogs>

Now we should add and else the above is statement that will call the custom method to create the dialog.

Graphical user interface, text

Description automatically generated

1. See an error? Yes we need to create the BTAlertDialog function that we have called in the else statement. Please create it OUTSIDE the onCreateas it does not need to run as soon as the Activity is opened. There is no return required so it will be void.
2. First create an AlertDialog builder object, we will use this builder object to set the Dialog attributes to.



1. Now we can add the buttons to the builder, these are purely programmatic, so we set the button text here too.

Text

Description automatically generated

1. So at the moment the buttons just create logcat messages, in which onClick would you place the code below to enable Bluetooth? We are just calling enable() on the Bluetooth adapter object we already created.



1. Now we only have button but the user needs more information, we van use the builder object to add more details to the dialog



1. We then have to create the dialog from the builder object

A screenshot of a computer

Description automatically generated with low confidence

1. And now we have completed the dialog we can show it, completing out BTAlertDialog() function

A picture containing text, orange

Description automatically generated

1. Now you should have an Alert Dialog that turns on the Bluetooth if the user presses OK

A picture containing text, monitor, screenshot, electronics

Description automatically generated

## No Android Device? LET’S SIDELOAD!

What’s sideloading? “Sideloading” an app is the common term for installing it without downloading directly from the Google Play Store.

Bluetooth does not work on the Android Emulator so make sure you use real devices to test your code. However, if you have no device, please contact other students/friends/family as they will be able to run your app if you share it with them via Google Drive. Thus, sideloading your app. Obviously you should only sideload apps from known sources.

To do so you will need to;

1. build your app in Android Studio via the Build > Build Bundles / APKs > Build APK

Graphical user interface, text

Description automatically generated

1. You should get a notification in the bottom left of your screen if you are successful. Note the build will not work if you have errors. You can still use the emulator to check for crashes but it will not scan.

Graphical user interface, text

Description automatically generated

1. Click locate to find your app APK, then upload it to google drive
2. Click share and enter the persons gmail address (it all works better through gmail)
3. They will receive a share notification and should be able to click on the file and it will install. If it doesn’t work they may need to opt-in to install unknown apps

<https://developer.android.com/distribute/marketing-tools/alternative-distribution#unknown-sources>