WifiDirect Guide

David Bui
Fall 2021

Preliminaries

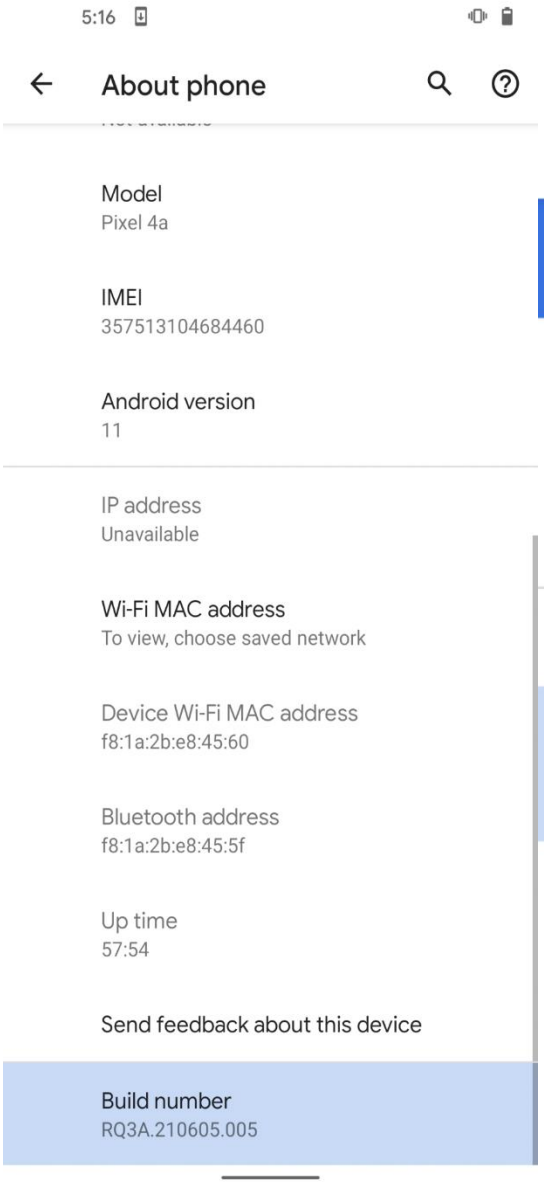
This is a Wi-Fi Direct user guide for the Disconnected Data Distribution project for the SJSU CS Systems group under Professor Ben Reed. Specifically, this guide will go over the SimpleWifiDirect repository [2]. The repository contains an example app to demonstrate the Wi-Fi Direct wrapper package to simplify calling Wi-Fi Direct operations. Links to the repository and other useful resources can be found at the end of this guide.

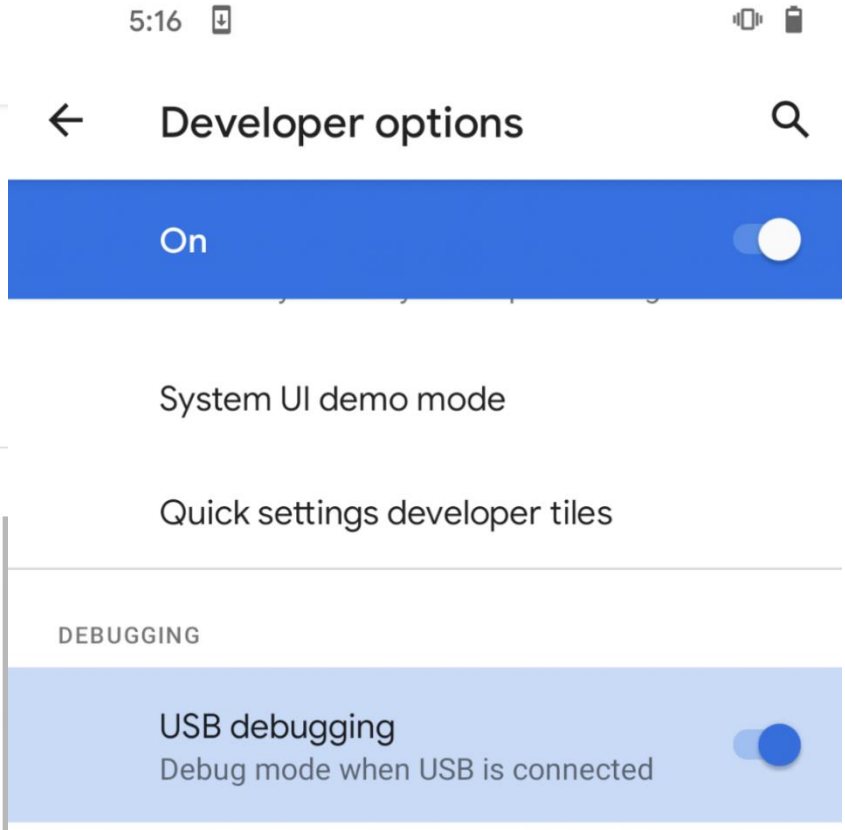
Wi-Fi Direct is a Wi-Fi standard for peer-to-peer connections that allows two devices to directly connect with one another. All Android devices have this feature natively, and it is usually already enabled automatically under Settings. IOS devices currently do not implement Wi-Fi Direct. Devices that are connected through Wi-Fi direct form a group with a group owner device.

Installation:

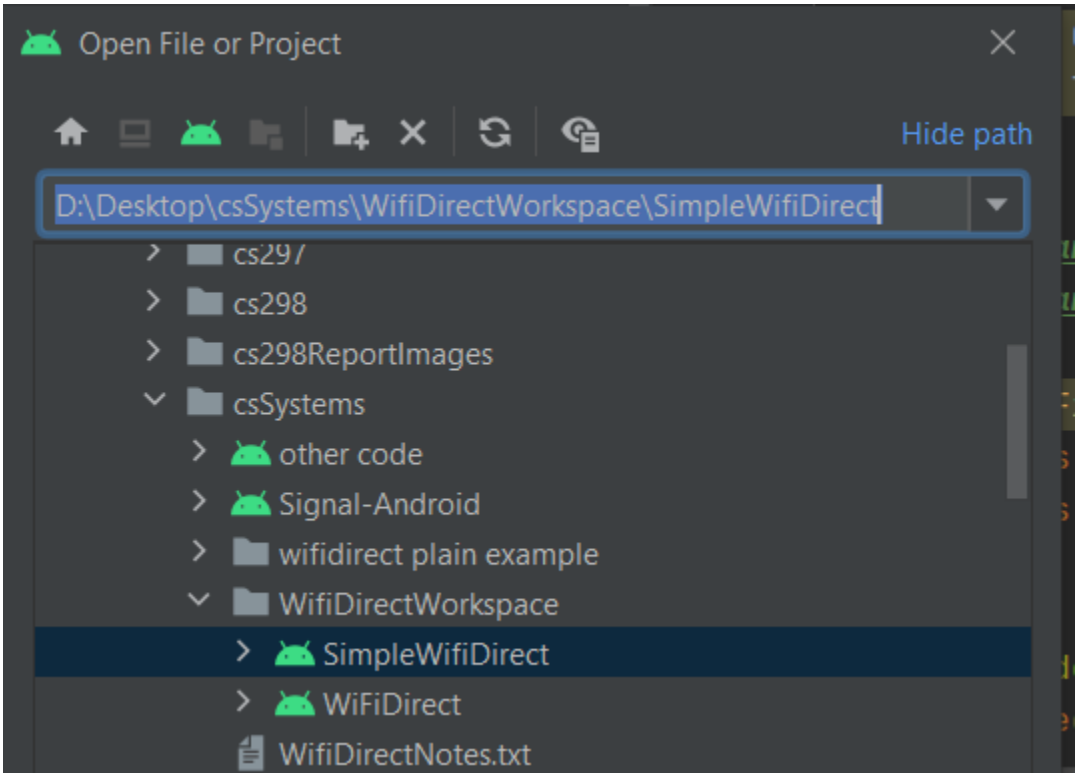
See the README in the SimpleWifiDirect Repo for more detailed installation information. Use this guide with the README for installation. Screenshots are taken on a Google Pixel 4a running Android OS version 12. Android studio is version 2020.3.1 for Windows.

On your android phone enable developer mode by repeatedly tapping on the build number located under Settings. A pop will appear displaying if developer was successfully enabled. After this enable USB debugging mode also located under settings.

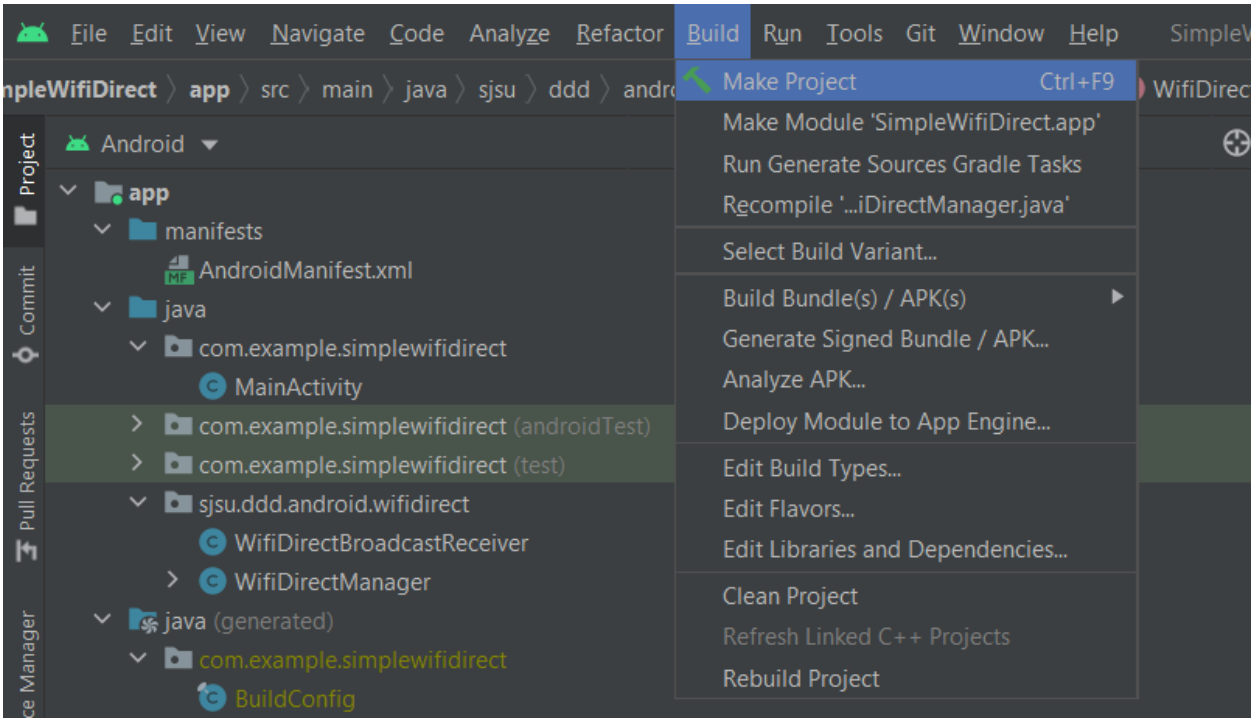

5:16
About phone
Model
Pixel 4a
IMEI
357513104684460
Android version
11
IP address
Unavailable
Wi-Fi MAC address
To view, choose saved network
Device Wi-Fi MAC address
f8:1a:2b:e8:45:60
Bluetooth address
f8:1a:2b:e8:45:5f
Up time
57:54
Send feedback about this device
Build number
RQ3A.210605.005


5:16
Developer options
On
System UI demo mode
Quick settings developer tiles
DEBUGGING
USB debugging
Debug mode when USB is connected

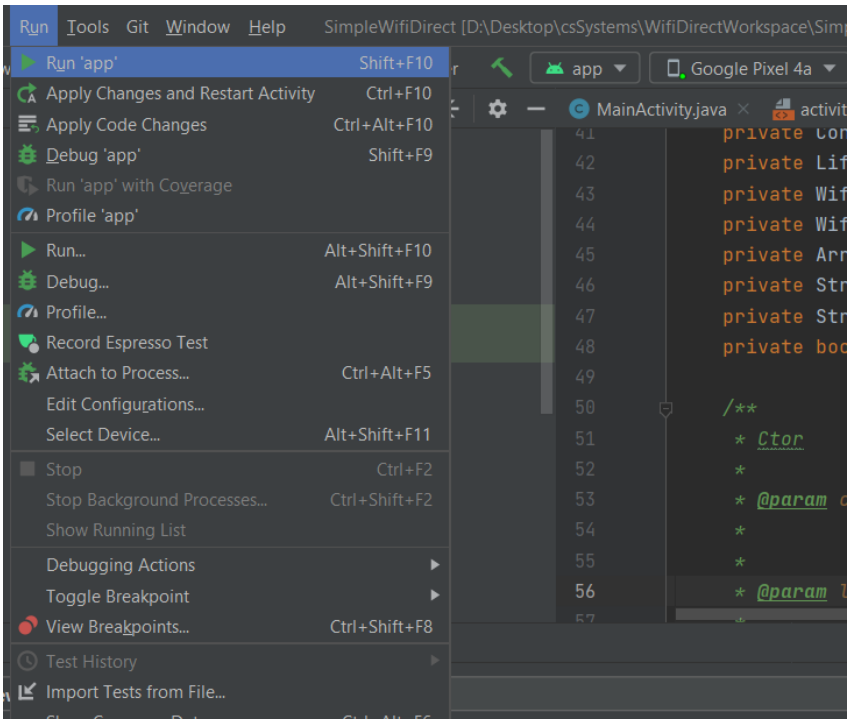
Once Android studio is installed and the SimpleWifiDirect repo is cloned, open Android studio, click File/Open, and select the SimpleWifiDirect repository. Note this is how you open a new project when you already have one open. The initial screen will look slightly different.



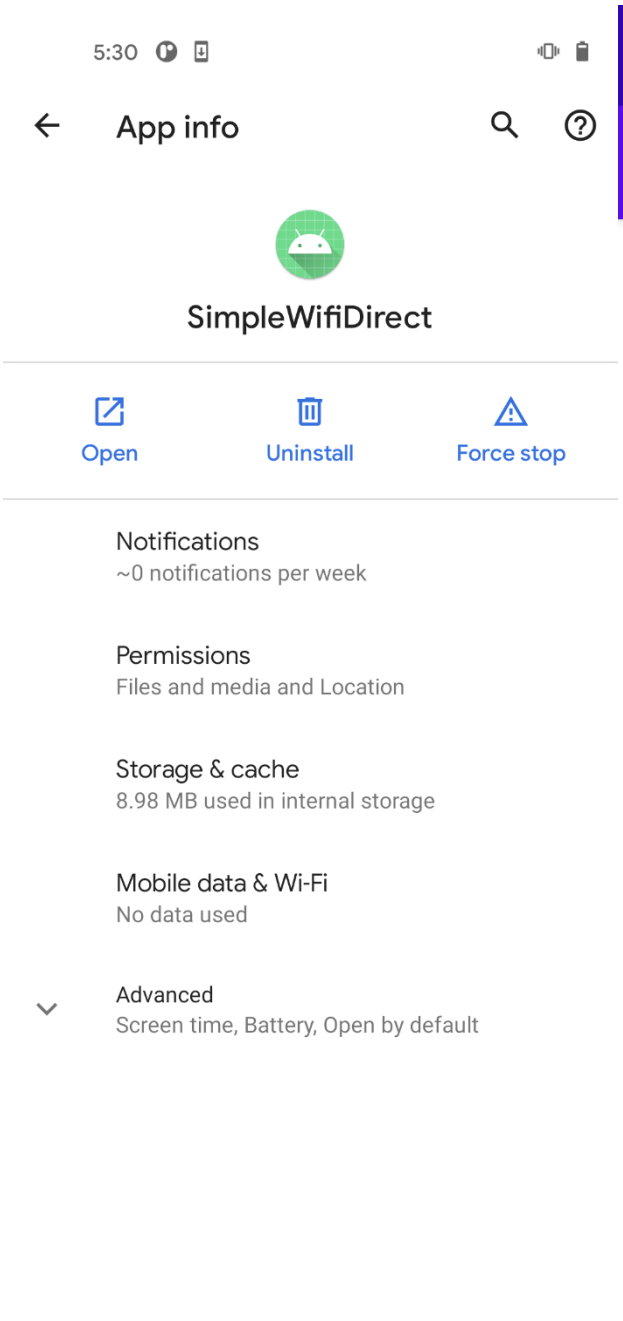
The next step is to build the project. The SJSU network is currently blocking whatever initial script runs when any Android Project. So either hotspot your phone or run this step on your home network. Click Build/Make Project to start the process. On a SJSU network, this process will just run forever and never complete.

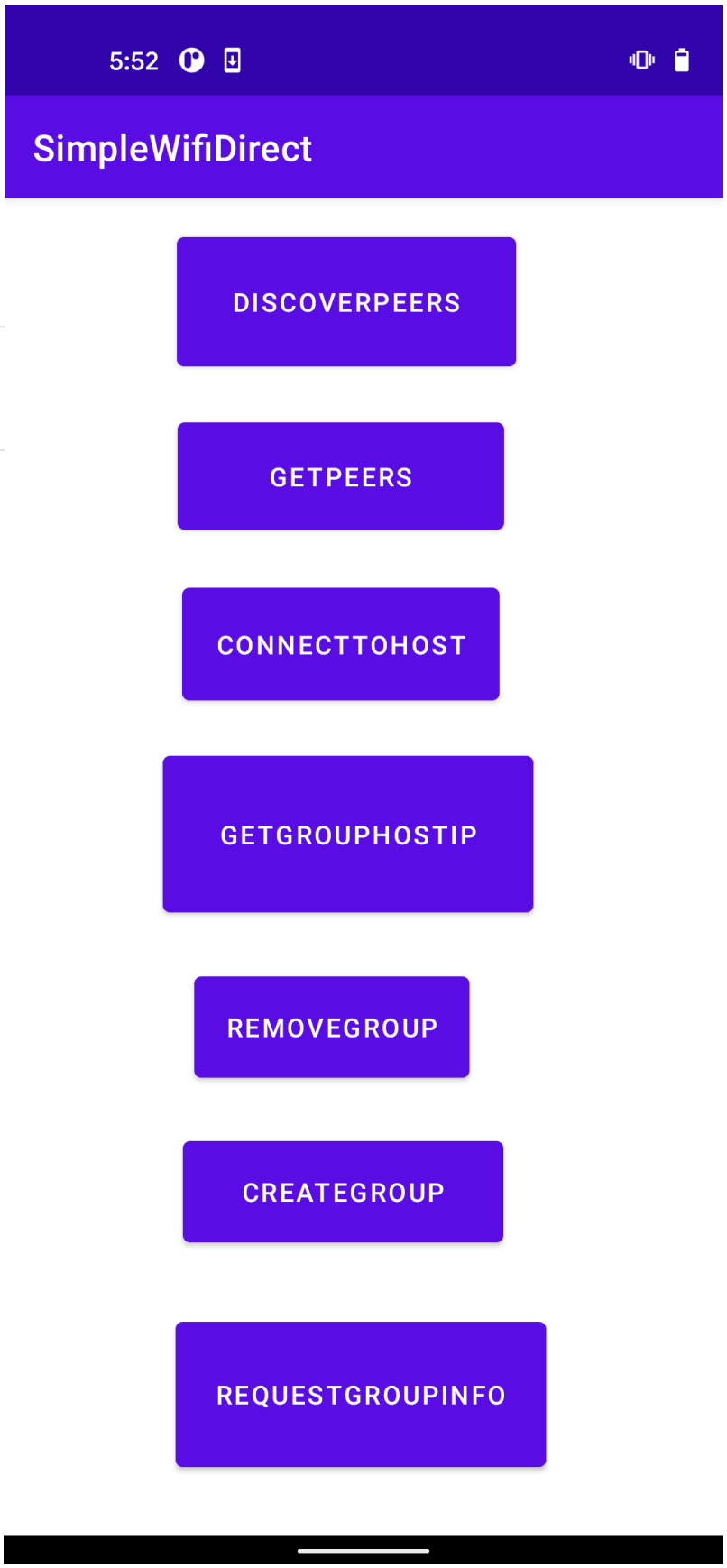


Connect your android phone to your computer. If everything is set up correctly the phone will show up in android studio under a drop down. Click Run/Run App to run the app on you phone. The app should automatically open on the phone if everything is correct.

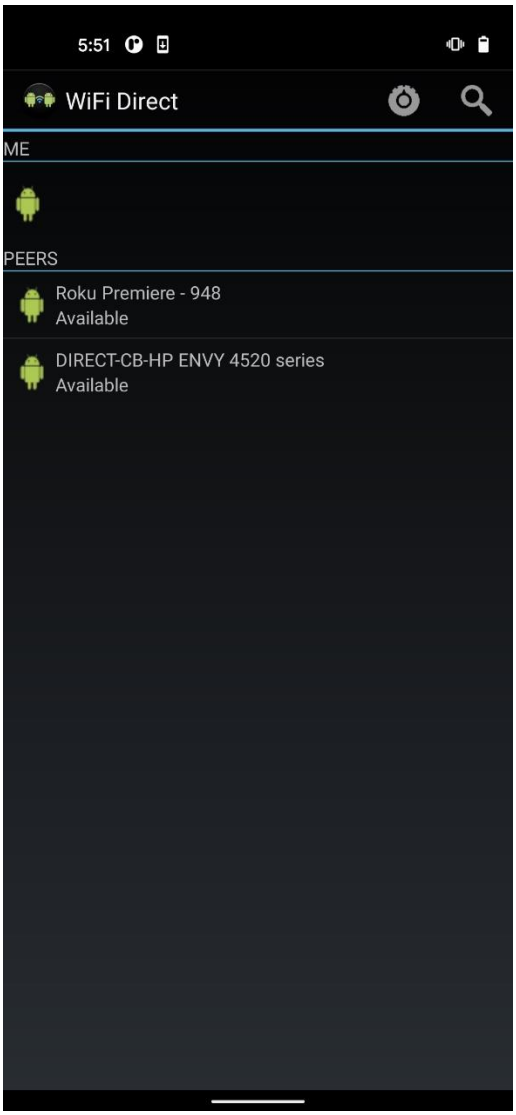


Swipe out of the app and hold click on the app itself. Click on App Info/Permissions. Give the app Location and File Permissions. The app itself should look like the picture on the right.



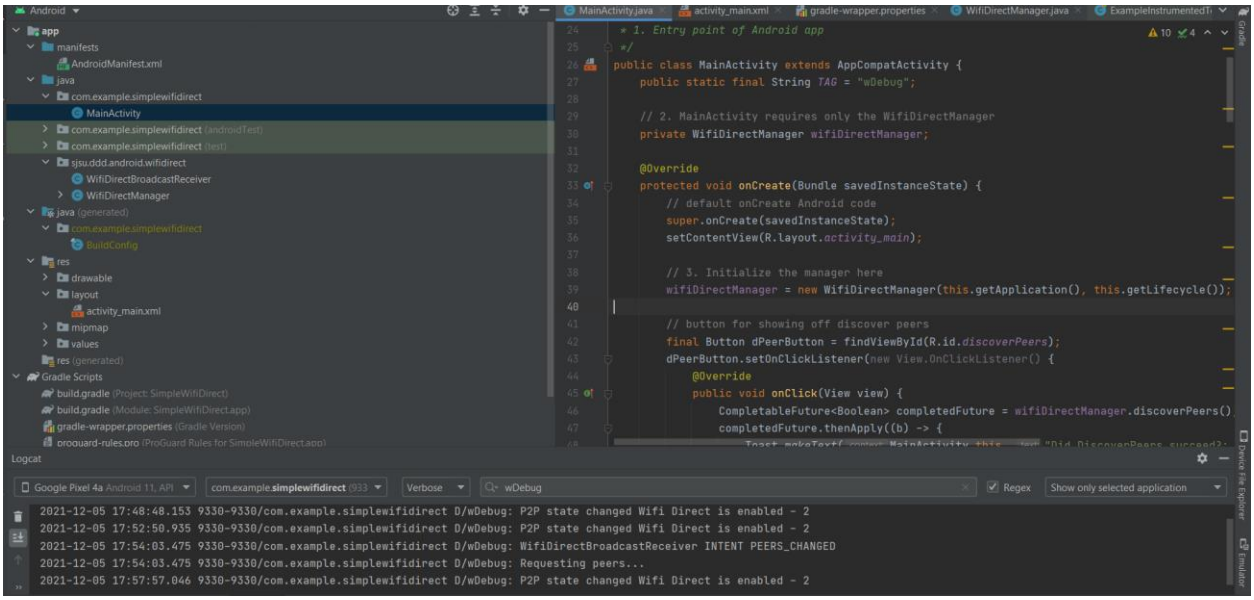


The SimpleWifiDirect app contains a simple button GUI to test our WifiDirect package. The original Google example shown below is feature complete WifiDriect example and allows you to transfer pictures between Android phones. See our updated Google example repository for running the app [3]. The Google code itself is difficult to trace but can serve as a good reference for performing file transfer operations.



WifiDirect Code

This section is a small walkthrough of the SimpleWifiDirect code. The code is commented for tracing purposes. Try to follow the comments while following this guide. Our WifiDirectPackage is located under sjsu/ddd/android/wifidirect. The package contains only two classes WifiDirectManager and WifiDirectBroadcastManager. The main entry point of the app is the MainActivity.java class. Logging for errors and callbacks is using the the wDebug tag using Android Studio's logcat. For those not familiar with Android this will be your System.print().



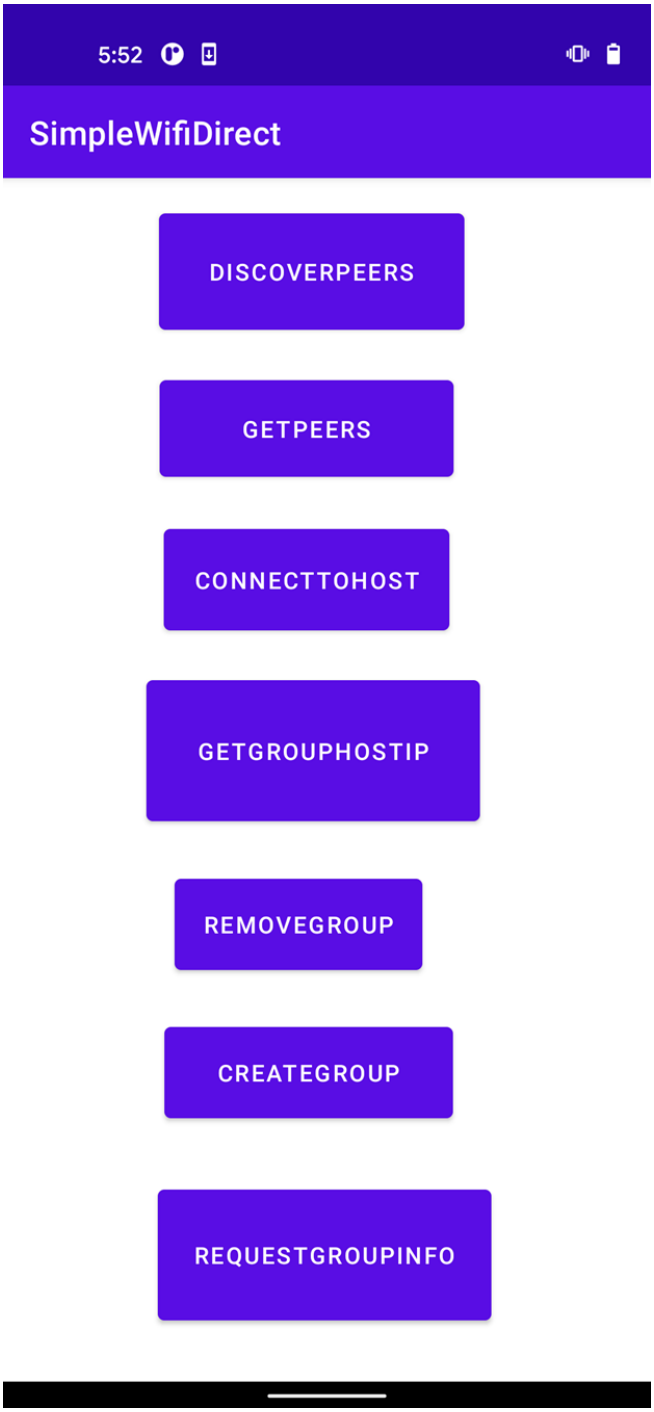
The seven buttons inside the onCreate method are tests for testing out The WifiDirectManager features.

The buttons right now are divided into two groups

The client buttons: DISCOVERPEERS, GETPEERS, CONNECTTOHOST, GETGROUPHOSTIP

Host buttons: CREATE GROUP, REQUESTGROUPINFO

Then there is the REMOVGROUP button which can be used for both.



Client Buttons

When DISCOVERPEERS button is pressed if will go through this function path

discoverPeers() in WifiDirectManager

onReceive() in WifiDirectBroadcastReceiver

under the if in WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION

requestPeers();

onPeersAvailable() back in WifiDirectManager .

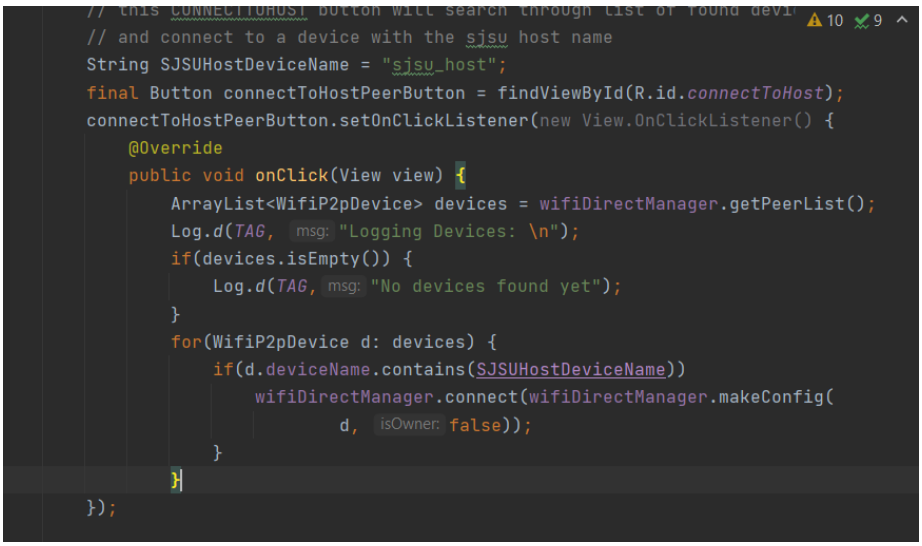
An ArrayList<WifiP2PDevice> is then stored in WifiDirectManager.

GETPEERS returns this Arraylist of devices.

The list will be logged in Android studio using Logcat with the tag wDebug.

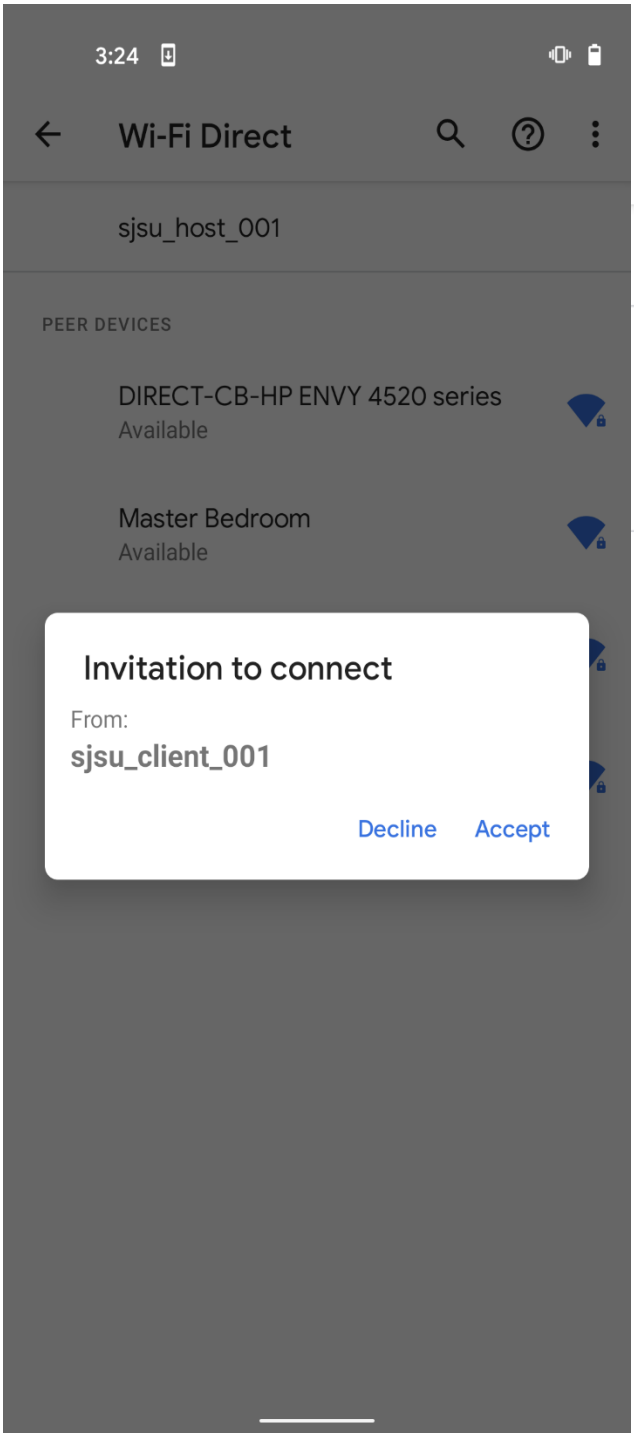
CONNECTTOHOST is a hard coded button that looks through this device list and connects to any Wi-Fi Direct device that has the name sjsu_host. The connection first requires converting the WifiP2PDevice into a WifiP2PConfig.

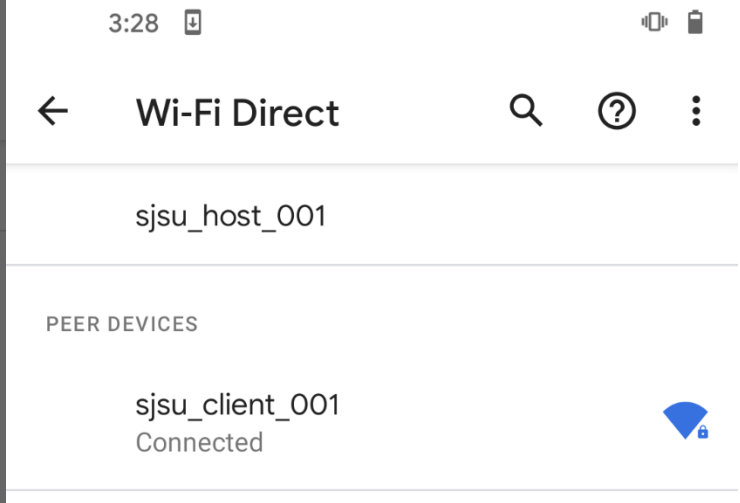
WifiDirectManager makeConfig() takes a WifiDirectDevice and a boolean indicating whether this the device is going to be the host of a WifiDirect group or the connecting device should be made the host.



```
// this CONNECTTOHOST button will search through list of found devi
// and connect to a device with the sjsu host name
String SJSUHostDeviceName = "sjsu_host";
final Button connectToHostPeerButton = findViewById(R.id.connectToHost);
connectToHostPeerButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        ArrayList<WifiP2pDevice> devices = wifiDirectManager.getPeerList();
        Log.d(TAG, msg: "Logging Devices: \n");
        if(devices.isEmpty()) {
            Log.d(TAG, msg: "No devices found yet");
        }
        for(WifiP2pDevice d: devices) {
            if(d.deviceName.contains(SJSUHostDeviceName))
                wifiDirectManager.connect(wifiDirectManager.makeConfig(
                        d, isOwner: false));
            }
        }
});
```

Note the device with the sjsu_host name should be discovered by DISCOVERPEERS before clicking this button. The ‘host’ device will get a pop displaying that this client device is asking for a connection. Press accept. A WiFi-Direct group should then be established between client and host.


3:24
Wi-Fi Direct
sjsu_host_001
PEER DEVICES
DIRECT-CB-HP ENVY 4520 series
Available
Master Bedroom
Available
Invitation to connect
From:
sjsu_client_001
Decline
Accept


3:28
Wi-Fi Direct
sjsu_host_001
PEER DEVICES
sjsu_client_001
Connected
REMEMBERED GROUPS
DIRECT-Na-sjsu_host_001

GetGroupHostIP returns the IP address of the owner of the Wi-Fi Direct Group. Use this IP to open a socket to communicate to the owner of the group. See the original Wi-Fi Direct code for an example. Below is picture from the FileTransferService Class.

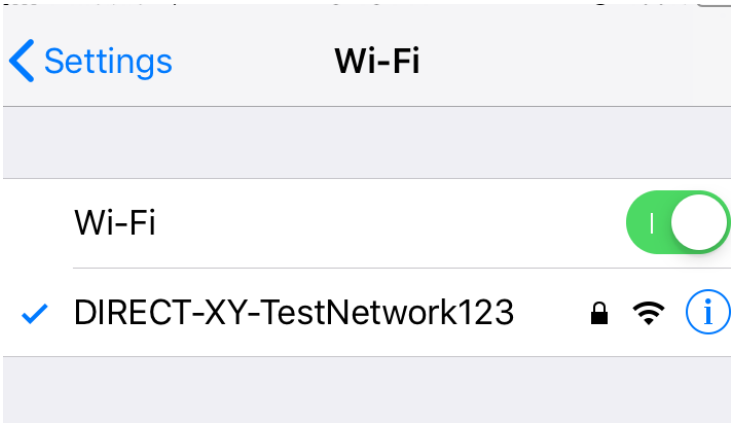
```
String fileUri = intent.getExtras().getString(EXTRAS_FILE_PATH);
String host = intent.getExtras().getString(EXTRAS_GROUP_OWNER_ADDRESS);
Socket socket = new Socket();
int port = intent.getExtras().getInt(EXTRAS_GROUP_OWNER_PORT);

try {
    Log.d(WiFiDirectActivity.TAG,  msg: "Opening client socket - ");
    socket.bind( bindpoint: null);
    socket.connect((new InetSocketAddress(host, port)), SOCKET_TIMEOUT);
```

The host is this group IP address. The port is just a hardcoded number.

Host Buttons

CREATEGROUP sets up a Wi-Fi Direct group with the current device as the group owner. This is a legacy feature and the created group acts like a wireless access point. Any device with Wi-Fi connectivity can discover this network and connect to it.



RequestGroupInfo returns a list of WIfiP2PDevices of the devices in the current Wi-Fi Direct group. Note this does not return the IP address of the devices in a group only the names and MAC addresses of each device. The device IP address of a client can only be retrieved after opening a socket. This stack overflow post details its implementation [8].

REMOVEGROUP disbands the current group this device is in. If the device is the the group owner, the group is disbanded. If the device is the client device the device leaves the group.

Background:

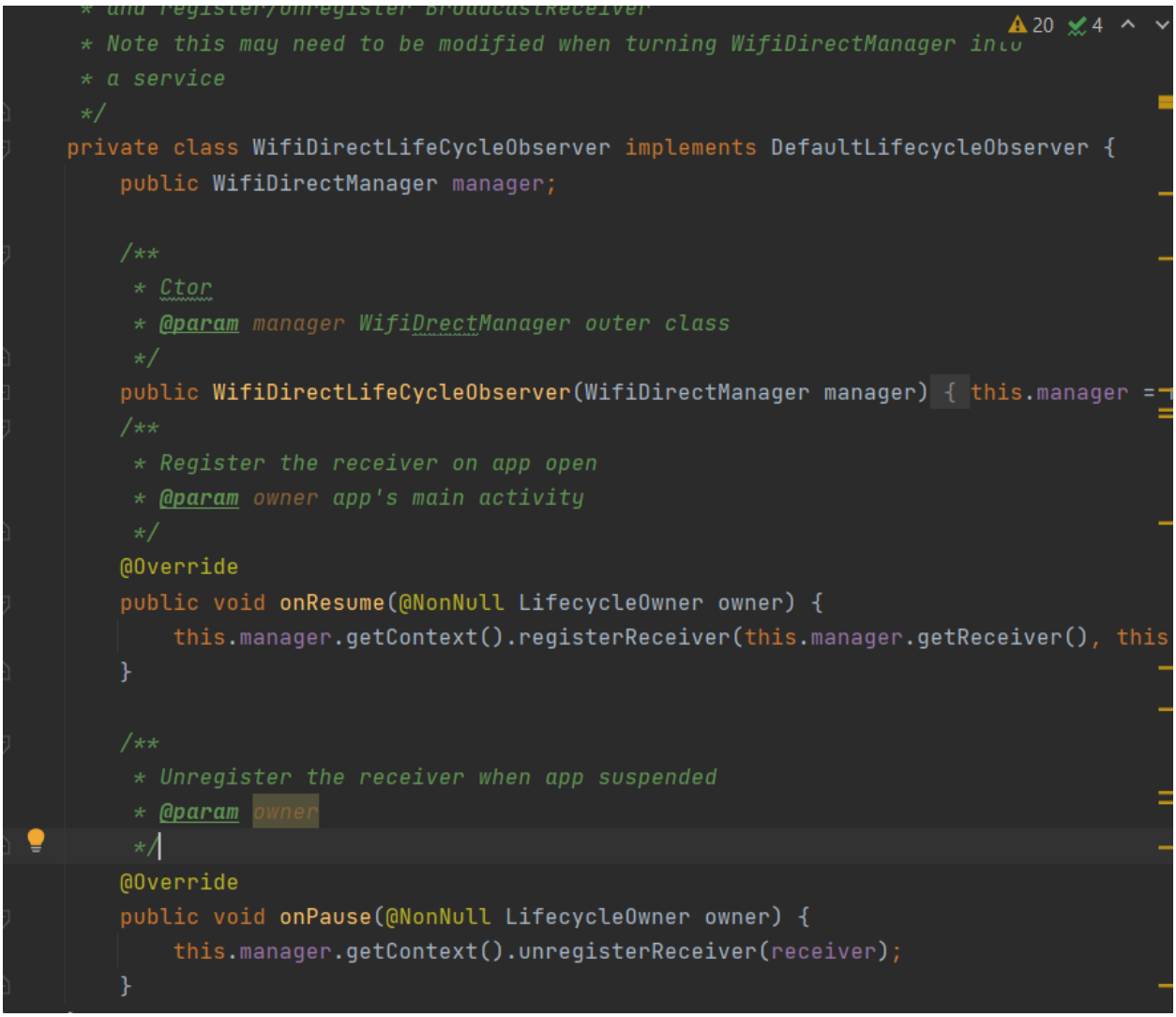
David Bui

Things to Look out for:

There are many more WifiP2PManager functions that we might need in the future. Many revolve around intents so make sure to check the WifiP2PManager library for a list of those unused intents [5]. Many Wi-Fi Direct functions go through this sequence where once a function is called using WifiP2PManager an Intent in WifiDirectBroadcastManager is triggered and where we call another function that has a callback function implemented with an Interface. Currently the intents are being registered using an inner class that hooks unto the App Lifecycle. This will need to be changed in the future if we are using Wi-Fi Direct as a service app.

Currently the app registers and unregisters the WifiDirectBroadcast receiver on app suspend and app resumption. If there are any questions, please message me at my email [7]. Good Luck!

Lifecycle Hook Class.



```
 * and register/unregister BroadcastReceiver
 * Note this may need to be modified when turning WifiDirectManager into
 * a service
 */
private class WifiDirectLifeCycleObserver implements DefaultLifecycleObserver {
    public WifiDirectManager manager;

    /**
     * Ctor
     * @param manager WifiDrectManager outer class
     */
    public WifiDirectLifeCycleObserver(WifiDirectManager manager) { this.manager =
    /**
     * Register the receiver on app open
     * @param owner app's main activity
     */
    @Override
    public void onResume(@NonNull LifecycleOwner owner) {
        this.manager.getContext().registerReceiver(this.manager.getReceiver(), this
    }

    /**
     * Unregister the receiver when app suspended
     * @param owner
     */
    @Override
    public void onPause(@NonNull LifecycleOwner owner) {
        this.manager.getContext().unregisterReceiver(receiver);
    }
```

Useful Links

1. The SJSU CS Systems GitHub Organization. Ask Professor Ben Reed for an invite before trying to access the other WiFiDirect repos.

   https://github.com/SJSU-CS-systems-group
2. The WifiDirect example repository using our created package

   https://github.com/SJSU-CS-systems-group/SimpleWifiDirect
3. The Updated Google WifiDirect example repository. We updated this to run on modern android phones. Working as of Dec 2021.

   https://github.com/SJSU-CS-systems-group/WiFiDirect
4. The original WifiDirect example repository. Once cloned the sample will be under the master branch in /samples/WiFiDirectDemo. Note: This currently does not work on modern android phones. Use as a reference.

   https://android.googlesource.com/platform/development/
5. Google Documentation on WifiDirect (known original as WifiP2P)

   https://developer.android.com/guide/topics/connectivity/wifip2p#discover-peers
6. The CS Systems Group Facebook. Contains communications and resources about the overall Disconnected Data Distribution Project. Ask Professor Ben Reed for access.

   https://www.facebook.com/groups/sjsu.cs.ddd
7. My personal email. If you have any issues or questions send me an email and I'll try to get back to you as soon as I can.

   bbdavidbb@gmail.com
8. Get Client device IP address

https://stackoverflow.com/questions/19008257/how-to-find-wifip2p-client-devices-address