

# **Cleaning and Mapping Robot Laboratory Journal**

**Beginning 5-27-2020**

**Brian Lauer**



## Contents

<b>Wednesday, May 27, 2020</b>	<b>7</b>
<b>Thursday, May 28, 2020</b>	<b>9</b>
<b>Monday, June 1, 2020</b>	<b>11</b>
<b>Tuesday, June 2, 2020</b>	<b>13</b>
<b>Wednesday, June 3, 2020</b>	<b>15</b>
<b>Thursday, June 4, 2020</b>	<b>17</b>
<b>Friday, June 5, 2020</b>	<b>19</b>
<b>Monday, June 8, 2020</b>	<b>21</b>
<b>Wednesday, June 10, 2020</b>	<b>23</b>
<b>Thursday, June 11, 2020</b>	<b>25</b>
<b>Tuesday, June 16, 2020</b>	<b>27</b>
<b>Wednesday, June 17, 2020</b>	<b>29</b>
<b>Thursday, June 18, 2020</b>	<b>31</b>
<b>Friday, June 19, 2020</b>	<b>33</b>
<b>Monday, June 22, 2020</b>	<b>35</b>
<b>Tuesday, June 23, 2020</b>	<b>37</b>
<b>Wednesday, June 24, 2020</b>	<b>39</b>
<b>Thursday, June 25, 2020</b>	<b>41</b>
<b>Friday, June 26, 2020</b>	<b>43</b>
<b>Monday, June 29, 2020</b>	<b>45</b>
<b>Tuesday, June 30, 2020</b>	<b>47</b>

## Contents

---

<b>Thursday, July 2, 2020</b>	<b>49</b>
<b>Monday, July 6, 2020</b>	<b>51</b>
<b>Tuesday, July 7, 2020</b>	<b>53</b>
<b>Wednesday, July 8, 2020</b>	<b>55</b>
<b>Thursday, July 9, 2020</b>	<b>57</b>
<b>Saturday, July 11, 2020</b>	<b>59</b>
<b>Monday, July 13, 2020</b>	<b>61</b>
<b>Tuesday, July 14, 2020</b>	<b>63</b>
<b>Wednesday, July 15, 2020</b>	<b>65</b>





## Wednesday, May 27, 2020

### Problem Statement:

Various cleaning and mapping (CAM) robots such as the Roomba by iRobot are commercially available today and are able to clean different areas of a home or business. Some of the top of the line robots by iRobot like the Roomba s9+, Roomba i7+, and Braava m6 are able to map rooms with a technology called Imprint Smart Mapping technology. In the app for each robot, a clean map report can be generated<sup>1</sup>. The generated 2D map will show areas that were cleaned by the robot and the white areas that contained obstacles like furniture or appliances. In these robots that support the mapping feature, a camera is mounted at an angle of 45 degrees that takes pictures of a room regularly. These images are then processed using Visual Simultaneous Localization and Mapping (VSLAM)<sup>2</sup>. Multiple alternative techniques for localization and mapping a robot including lidar and RFID technology. In the proposed research, RFID technology will be utilized for the purpose of possibly lowering costs and computational complexity.

### Research efforts:

In [?], a probabilistic model for localizing an RFID tag is given for the purpose of localizing a robot in its environment. Given a the position  $x$  of the tag and data  $z_{1:t}$ , the probability of obtaining the correct position of the robot given the observations is

$$p(x|z_{1:t}) = \alpha p(z_t|x)p(x|z_{1:t-1})$$

where  $\alpha$  is the normalization constant. The purpose of this study was to determine the locations of RFID tags in an office building using a laser range finder and an RFID reader mounted on the robot. With the RFID tag's locations the path of the robot could be estimated which is known as localization. However, it does not use RFID specifically for mapping an environment. Therefore, techniques must be found for determining how to map the robot's environment. This may only be possible with sensors like a laser range finder, LIDAR sensor, or other digital devices like a camera.

<sup>1</sup>[https://homesupport.irobot.com/app/answers/detail/a\\_id/1467](https://homesupport.irobot.com/app/answers/detail/a_id/1467)

<sup>2</sup><https://www.thezebra.com/resources/home/how-roomba-works/>





## Thursday, May 28, 2020

I received the resources for the iRobot Create 2 from Jaden today in his Google Drive. The iRobot Create 2 is a hackable device designed for education rather than commercial use<sup>1</sup>. On iRobot's site, many different projects have been done with this robot such as a Nerf-gun mounted turret modification, nerf-gun dart collector robot, camera bot, pokemon go egg hatching, zcm driver, and path finder. All of them are documented in the provided footnote.

<sup>1</sup><https://www.irobot.com/About-iRobot/STEM/Create-2/Projects.aspx>



## Monday, June 1, 2020

Today, I did some research on how to implement the mapping feature with a LIDAR sensor as Prof. Miah mentioned that this technology is available in some of the labs. I looked at a tutorial on how to interface the Neato LIDAR sensor with a Beaglebone <sup>1</sup>. By simply connecting the LIDAR sensor to the TX and RX pins of the board, data can be sent to the device or the encoded can be retrieved. The mentioned LIDAR comes with a motor which can be powered with a 3V power supply. To read from the device the device file in the /dev directory must be opened. Each packet consists of 22 bytes. The code below:

```
print("Hello")

with open("/dev/tty02", "rb") as f:
    byte = f.read(1)
    count = 1
    while byte != "" and count <= 22*100:
        print(byte.encode('hex') + ":")
        byte = f.read(1)
        count += 1

print("Goodbye")
```

where the LIDAR is located at /dev/tty02. However, the LIDAR sensor that Prof. Miah has in his robotics lab is likely different as there are a lot of different options out there.

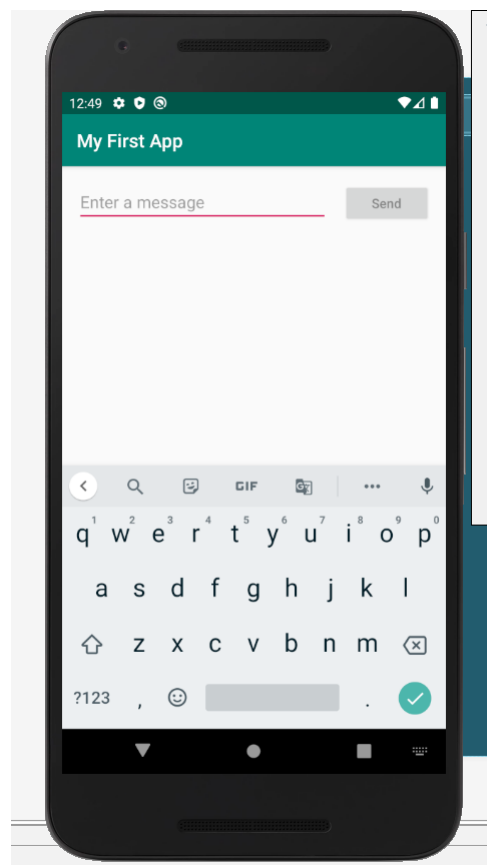
<sup>1</sup><https://www.alexschimp.com/neato-lidar-data-beaglebone-black/>



## Tuesday, June 2, 2020

In yesterday's meeting with Jaden, my partner, we agreed that he would focus mostly on the hardware side of things as he has the robot and beaglebone while I work on building an Android app for showing the map. In this way, I will be able to make more of a significant contribution toward the project with the resources I currently have. I have never built an app before so some research will be required before I start developing the app.

I started by following the Android tutorial<sup>1</sup>. The UI for each app is built using a system of layouts and widgets where the layouts are ViewGroup objects and the widgets are View objects. Objects in the layout like text objects and buttons can be constrained to each other. For each TextView object, a `textSize` and `textColor` variable can be set. The result of the tutorial is shown below:



<sup>1</sup><https://developer.android.com/training/basics/firstapp/creating-project>

Tuesday, June 2, 2020

---

where after text is entered and the button is pressed, a new activity is shown which displays the text under the top app bar.

## Wednesday, June 3, 2020

Today I took a pause on developing the app by looking at the link to the Lidar sensor that Dr. Miah sent us that is available in the robotics facility <sup>1</sup>. The model is a RPiLIDAR S1 360°. According to the datasheet it can perform 2D 360° scan of an area with a 40 meter range. The sampling rate is 9200 samples per second and the sensor can tilt vertically up 0.391°. While scanning, the data output of the sensor over UART includes both the distance between the rotating LIDAR sensor and the sampling point while the heading is the current angle of measurement in degrees. A flag in the data will signal the start of a new scan. Both the distance and heading values will be sent as pairs for data communication for each sample point. The pinout for the device is as follows:

Pin Name	Description
VCC	Power
GND	Power
RX	Serial Input
TX	Serial Output

where there are two pins for VCC and GND each as separate power is needed for the sensor and motor system that rotates the sensor. Because the connector used for the sensor is a SH1.0-6P, some special wires may be needed for connecting to the device. As far as the specification for the serial interface goes, the baud rate is 256000 bps and uses 3.3V TTL logic.

Since I am on my laptop and do not have android studio installed on it, I began installing it. I realized that the Create 2 does not have any sort of internet connectivity, so a web server will be needed on the Beaglebone to communicate with it wirelessly. In the app, some functionality will need to be developed to discover the Roomba on the network.

I created a new project called LidarMap using the bottom navigation bar template.

<sup>1</sup>[https://www.robotshop.com/en/rplidar-s1-360-laser-scanner-40-m.html?utm\\_source=google&utm\\_medium=surfaces&utm\\_campaign=surfaces\\_across\\_google\\_usen&gclid=EAIaIQobChMI9JyBto7m6QIVkIbACh3GMwhZEAYYASABEgIebPD\\_BwE](https://www.robotshop.com/en/rplidar-s1-360-laser-scanner-40-m.html?utm_source=google&utm_medium=surfaces&utm_campaign=surfaces_across_google_usen&gclid=EAIaIQobChMI9JyBto7m6QIVkIbACh3GMwhZEAYYASABEgIebPD_BwE)





## Thursday, June 4, 2020

I spent some time wireframing the UI for the app on paper today. The app will have a bottom navigation bar with the pages Discovery and Map. The Discovery page will contain the list of discovered robots while the Map page will contain the map similar to the map shown in the Roomba app. Similar to my building energy management project, the robots should be discovered automatically with an "autodiscovery" feature. While waiting for devices to be discovered I want to add an indefinite progress bar spinner. Since I realized that it is possibly going to be more difficult to change the current icons in the current app I decided that I will simply start making the app using a blank template.

I thus created a new Android Studio project named Create2LidarMap using Android 4.0.3 (Ice Cream Sandwich) using the empty activity template. I then followed <sup>1</sup> to create a bottom nav bar from scratch.

<sup>1</sup><https://androidwave.com/bottom-navigation-bar-android-example/>



## Friday, June 5, 2020

I worked on adding the bottom navigation menu from scratch into the app. The icons were successfully added where one is a magnifying glass and the other is a quilt for the map page. The NavHostFragment component in the MainActivity page will provide an area above the nav menu for self contained navigation to occur. Essentially two different fragments will be created for the discovery and map respectively. However after adding the component to the main activity and importing the Android Jetpack package `androidx.navigation.fragment`, I got a warning stating that to Replace the tag With `FragmentContainerView`. I made the change to this View but another issue came up as the `navGraph` attribute expects a resource of type `String` but currently has a navigation resource type (`@navigation/mobile_navigation`). Inside the `onCreate` method in the `MainActivity` class, I added the line

```
this.getSupportActionBar().hide();
```

To remove the top action bar with the name of the activity which in my opinion makes the app look a little cleaner.



## Monday, June 8, 2020

Because of the issues mentioned previously with adding the navbar from a blank project, I decided to simply take the original LidarMap project and modify it with the appropriate fragment and subtract and add the appropriate navbar elements. In the java directory I created a package `com.example.ui.lidarmap.map` where I created the java file `MapFragment.java` and `MapViewModel.java`. The source code for `MapFragment` was generated by default and contains the following:

```
package com.example.lidarmap.ui.map;

import androidx.lifecycle.ViewModelProviders;

import android.os.Bundle;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import com.example.lidarmap.R;

public class MapFragment extends Fragment {

    private MapViewModel mViewModel;

    public static MapFragment newInstance() {
        return new MapFragment();
    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
                             @Nullable ViewGroup container,
                             @Nullable Bundle savedInstanceState) {
        return inflater.inflate(R.layout.map_fragment, container, false);
    }
}
```

```
@Override
public void onActivityCreated(@Nullable Bundle savedInstanceState) {
    super.onActivityCreated(savedInstanceState);
    mViewModel = ViewModelProviders.of(this).get(MapViewModel.class);
    // TODO: Use the ViewModel
}

}
```

I created a second package named `com.example.lidarmap.ui.discovery` with the files `DiscoveryFragment` and `DiscoveryViewModel` which similar code to the map fragment and view model classes. After adding these files, I generated the layout files. However, the build failed with the exception: `java.lang.RuntimeException`. I thus decided to once again create a new project from scratch in hopes of fixing the previous issues from the `CAMRobotMap` project.

I decided to scrap all my previous ideas and simply follow a tutorial on Youtube at <sup>1</sup>. At the beginning of the tutorial, the speaker mentioned that if only two destinations are needed (map, discovery) then it is better to use a tab layout instead. I deleted the old project with the bottom nav and created a new project with tabbed activity template. Instead I decided the follow the tutorial <sup>2</sup>. In the "Discovery" tab, I added a `ListView` object

<sup>1</sup><https://www.youtube.com/watch?v=tPV8xA7m-iw>

<sup>2</sup>[https://www.youtube.com/watch?v=h4HwU\\_ENXYM](https://www.youtube.com/watch?v=h4HwU_ENXYM)

## Wednesday, June 10, 2020

What I plan on accomplishing today is creating a github repository for the android app and sharing with Jaden, so that we can both work on the software together as he cannot do much due to hardware limitations with the Roomba. Next I would like to start writing a method to create a TCP/UDP client and send out broadcast messages on the network for the Roomba. First, I asked Dr. Miah for permission before pushing the project to a public Github repository. However, I found that I recent change was made to github that allows for unlimited private repos with fewer than 3 or 4 collaborators. When attempting to the push the changes to the repository I got the error:

```
Push rejected: Push to origin/master was rejected
```

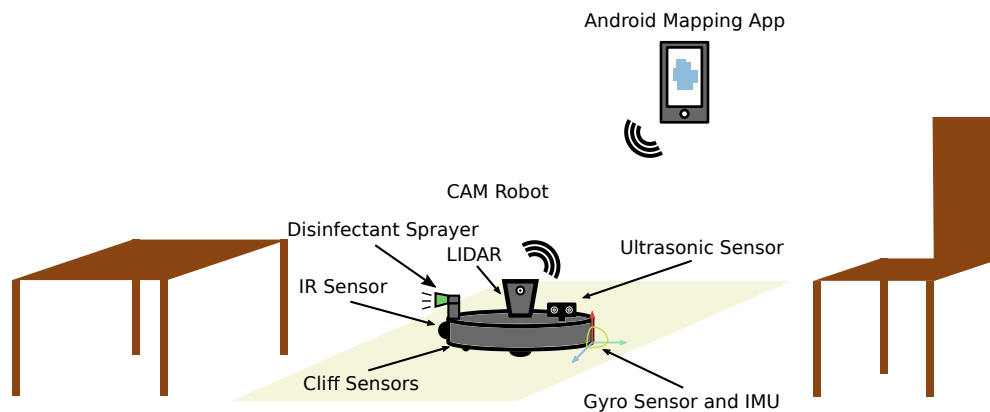
This was fixed after I rebased from master to refs/remotes/origin/master. Thus, I was able to succesfully push the project to the github repo LidarMappingRobot.

Although the priority is to write the TCP/IP client code for the app, I would like to add subitems into the list view for the Discovery fragment.





## Thursday, June 11, 2020



**Figure 1:** Diagram representing the working principal of the cleaning and mapping robot

### Description:

The proposed cleaning and mapping robot based off the Roomba vacuum by iRobot will have the ability to collect sensor data from multiple onboard sensors connected to a single board computer (Beaglebone Blue) for the purpose of localization and mapping. IR sensors, ultrasonic sensors, cliff sensors, and LIDAR will help the robot to understand its environment better. The IMU and gyroscope will help to track orientation, velocity, and rotation. With the aid of the computer, the robot will be able to successfully map its environment like a floor of a house or building. The LIDAR sensor will perform a continuous 360° sweep of the environment to generate a 2D map of the environment. For visualizing the fully completed map, an Android app will be developed to display the outcome of the mapping graphically. Built-in WiFi support of the Beaglebone will act as the communication medium between the robot and client. iRobot has developed a technology of their own that can perform mapping of an area and identify areas of high dirt or dust concentration. However, the robot to be developed will seek to use cost effective sensors to clean the floor and build a map of the environment. The robot will exhibit human like reasoning and intelligence including perception and memory through storage of maps on the single board computer. Techniques like artificial intelligence will be deployed for improving the human like perception abilities of the device even further.



## Tuesday, June 16, 2020

Brian: I found a blog post on the Internet detailing how to implement a network scanning feature in Android<sup>1</sup>. At the end of section 1, it states that networking code should not run in the UI. In other words, a separate thread of execution must be created for network operations. In this case, scanning all IP addresses on a subnet is the task at hand. I worked on understanding the method `getIpv4Netmask()` which should return the network prefix length or in other words the number of bits in the network portion of the IP address. However I tried the code out in Eclipse and received a `NullPointerException`. This may have to do with passing the wrong IP address as a parameter into the method call. To try and understand what IP address should be passed in an argument, I used the lines

```
WifiManager wifiManger = (WifiManager) getSystemService(Context.WIFI_SERVICE);  
DhcpInfo dhcpInfo = wifiManger.getDhcpInfo();
```

in the `onCreate` method of the `MainActivity` class. However, the `getDhcpInfo` call was returning an error as the `WifiManager` is missing permissions. What I found is that the permission `ACCESS_WIFI_STATE` is required in the Android Manifest file which is accomplished by adding the line.

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

to the `AndroidManifest.xml` file. Also, I changed the lines to

```
WifiManager wifiManager = (WifiManager) getContext().getApplicationContext()  
.getSystemService(Context.WIFI_SERVICE);  
DhcpInfo dhcpInfo = wifiManager.getDhcpInfo();
```

As a test, since the wifi is not enabled on the emulator, I downloaded the app to a test Moto E phone. An added `dhcpInfo.ipAddress` to the the list of items. It displays the integer 318810304 which by using a decimal to IP converter I found the this represents the IP address 192.168.0.19.

<sup>1</sup><https://cheesecake labs.com/blog/developing-iot-apps-connecting-smart-devices/>



## Wednesday, June 17, 2020

Since I was having problems with finding the prefix length with the `getIpv4Netmask` provided in the footnote above, I decided to rather get the localhost of the device and get the network interfaces attached. In this way, hopefully a null reference exception will be avoided. Unfortunately, when I call `Inet4Address.getLocalHost()` the app crashes. I found that this is caused by a `NetworkOnMainThreadException` being thrown by the application as it was trying to perform a network operation on the main thread. I did not realize that it is absolutely required to add in a second thread to perform network operations.

I need to create another class in the `DiscoveryFragment` class that extends from the `AsyncTask` class. However, I discovered that the `AsyncTask` class was deprecated in API level 30. Instead I will use the more up to date `Executor` interface which contains one method: `execute`. In the implementation of the `execute` method, it creates a new thread and starts it as follows:

```
class TaskExecutor implements Executor {  
    public void execute(Runnable r) {  
        new Thread(r).start();  
    }  
}
```

Where `Runnable` is a second interface that adds the `run` method which will be executed when the thread is started. In the `run` method, I would like to get the network prefix of the subnet and add to the `ArrayList` to be added to the `ListView` in the android application.

This second issue to be solved is the issue of the `getLocalHost` method returning a `InetAddress` object bound to the IP address `127.0.0.1` rather than the actual address of the device. The network mask prefix of `127.0.0.1` is 8 rather than the expected 24 on my home network. After implementing some more of the code by using an instance of the `DhcpInfo` object, I was able to successfully display the prefix (24) in the app.



## Thursday, June 18, 2020

Today will be detected to searching for all devices under my networks subnet (255.255.255.0). I created a second task titled `ScanNetworkTask`. However, I would like to write a method that will be called in the `ScanNetworkTask` run method for obtaining the network prefix rather than using a separate thread to find the prefix. The purpose for doing this is to prevent a race condition with the class level member variable `mNetworkPrefix`. The problem I am having is that private member variables cannot be accessed from static methods in java. The variable I would like to reference is the instance of `DhcpInfo` which is initialized in the `onCreate` method of the `DiscoveryFragment` class. However, in the end, I just decided to add all the code including the code to determine the network prefix in a single task class titled `ScanSubnetTask`. The code after the network prefix code will loop from 192.168.0.0 up to 192.168.0.254 and ping each of those addresses by calling the `InetAddress.isReachable` method with a timeout of 80 ms. In the task, I decided to update the `ArrayList` with the list of reachable IP addresses. This causes an `IllegalStateException` because the background thread should not be sending notifications to the UI to update it. Correctly communicating with the UI from backgrounds will be necessary here <sup>1</sup>.

<sup>1</sup><https://developer.android.com/training/multiple-threads/communicate-ui>





## Friday, June 19, 2020

Today, I read through some of the android developer resources to determine how to update the UI from a background thread. A line that I found that should be added to the implementation of the run method is

```
android.os.Process.setThreadPriority(android.os.Process.THREAD_PRIORITY_BACKGROUND);
```

which moves the thread to the background. Inside the constructor of the `DiscoveryFragment` class I added an instance of an anonymous class which extends the `Handler` class will handle new message added to the task:

```
private DiscoveryFragment() {  
    // This is an anonymous class which extends the Handler class  
    handler = new Handler(Looper.getMainLooper()) {  
        @Override  
        public void handleMessage(Message inputMessage) {  
  
        }  
  
    };  
}
```

The problem is I don't know how to invoke the handler and what type of message will be passed into the `handleMessage` method. It turns out the `sendMessage` method must be used. From the android developer docs <sup>1</sup>, the `Message` class is used to define a message containing a description and data object to be sent to a handler. The `setData` must be called on an instance of the `Message` object in order to add data to the `Message` instance. When attempting to call the `sendMessage` method on the handler instance, I received the following exception:

```
java.lang.IllegalStateException:  
{ when=-14h27m43s520ms what=0  
target=com.example.lidarmappingrobot.ui.main.DiscoveryFragment$1 }  
This message is already in use.
```

<sup>1</sup><https://developer.android.com/reference/android/os/Message>



## Monday, June 22, 2020

One of the problems I discovered in my code, is that the `Message.sendMessage(Message)` method was being called inside a for loop. Thus, the same `Message` object was likely being used multiple times causing some issues. After adding the `sendMessage` call after the for loop the exception went away. However, no IP addresses were being listed in the Discovery page. Instead of assigning the `obj` field of the `Message` object to the `ArrayList`, I decided to use `Bundle` and the `putStringArrayList` method to set the data on the message. Now the problem appears to be that `handleMessage` is not being called which I added the single line to

```
mArrayList.add("Null pointer exception here!");
```

I realized I forgot to assign the `Bundle` object which is used for storing the `String ArrayList` and sending to the `Handler` over a `Message` to the output of `Message.getData()`.

A large problem I found in the current implementation of the subnet scanner I have adapted and built is that it takes at least 5-6 seconds in order to finish the call to `getByAddress` on the `InetAddress` object. This leads to the app taking over 10 minutes to finishing scanning all the devices. It turns out the call to `InetAddress.getHostName` takes a long time to call as it performs a DNS lookup under the hood. Therefore, a different approach may need to be taken instead. A solution was quickly found from the Stack Overflow post <sup>1</sup>, the `InetAddress.toString` method will simply print a slash and the name of the address instantly without any system call. A problem I am having is the inability to determine if addresses are reachable with the `InetAddress.isReachable` method.

<sup>1</sup><https://stackoverflow.com/questions/10420317/java-inetaddress-gethostname-taking-a-very-long-time-to-execute>



## Tuesday, June 23, 2020

Since I realized that scanning every single device on the network is not a good idea as it will take a long time, I decided to try to setup the SBC as a wireless access point. Because I don't have a Beaglebone, I will use a RPi instead given directions from stack exchange<sup>1</sup>. For this, I will follow the tutorial at <sup>2</sup> as the link provided in the previous url on stack exchange was returning a 404 error. However, I discovered that the RPi must be connected to the rest of the network via Ethernet, so this may not be a viable option either. A better alternative I found is using multicast sockets over UDP which would be more ideal in terms of speed <sup>3</sup>. The following code should be adapted for android to send data to the receiver (the Beaglebone).

```
import socket
import struct
import sys

message = 'very important data'.encode('utf-8')
multicast_group = ('224.3.29.71', 10000)

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

sock.settimeout(0.2)

ttl = struct.pack('b', 1)
sock.setsockopt(socket.IPPROTO_IP, socket.IP_MULTICAST_TTL, ttl)

sent = sock.sendto(message, multicast_group)
print('sent message')
while True:
    print('waiting to receive')
    try:
        data, server = sock.recvfrom(16)
    except socket.timeout:
        print('timed out')
        break
```

<sup>1</sup><https://raspberrypi.stackexchange.com/questions/66700/tcp-communication-with-android-via-wifi>

<sup>2</sup><https://www.raspberrypi.org/documentation/configuration/wireless/access-point-routed.md>

<sup>3</sup><https://pymotw.com/2/socket/multicast.html>

```
        else:
            print(data)
```

```
sock.close()
```

The following receiver code should be added onto for pulling lidar data from the robot that the bone is connected to:

```
import socket
import struct
import sys

multicast_group = '224.3.29.71'
server_address = ('', 10000)

# Create the socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Bind to the server address
sock.bind(server_address)

# Tell the operating system to add the socket to the multicast group
# on all interfaces.
group = socket.inet_aton(multicast_group)
mreq = struct.pack('4sL', group, socket.INADDR_ANY)
sock.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP, mreq)

# Receive/respond loop
while True:
    print >>sys.stderr, '\nwaiting to receive message'
    data, address = sock.recvfrom(1024)

    print >>sys.stderr, 'received %s bytes from %s' % (len(data), address)
    print >>sys.stderr, data

    print >>sys.stderr, 'sending acknowledgement to', address
    sock.sendto('ack', address)
```

Luckily, there is a class in the Android SDK specifically for multicast communication, `MulticastSocket`. One of the problems I am running into is how to properly obtain the IP address from the received datagram packet from the remote listener on the RPi. In other words after the call to `DatagramSocket.receive`, the argument passed to the method should receive the message received and have the ip and port of the remote host encoded in it. What I discovered is that a message must be received multiple times in order to properly receive the message from the server. However, when I restart the app, the packet sometimes gets lost and my code gets stuck in an infinite loop. A potential solution is to add a refresh button to look for devices in case the app did not get a response.

## Wednesday, June 24, 2020

I altered the code inside the run method of the FindRobotTask class to only iterate up to 10 times to receive a message from the SBC. This will prevent the infinite loop issue. Next what I would like to do is add in a floating action button to refresh whenever devices don't show up in the list view. The code is fairly simple as all I have to do is call `setOnClickListener` on the `FloatingActionButton` object:

```
FloatingActionButton fab = (FloatingActionButton) discoveryView.findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mArrayList.add("Refreshed");
        new Thread(new FindRobotTask()).start();
    }
});
```

However, the problem I have is the Refreshed string is not being properly added to the array list every time as it is not showing in the ListView. I added the line

```
Snackbar.make(discoveryView, "Replace with your own action", Snackbar.LENGTH_LONG)
    .setAction("Action", null).show();
```

in the definition of the `onClick` method which creates a pop up at the bottom of the screen whenever the button is clicked. This appears to work but items are still not being added to the listview. I was able to fix the problem of the refresh button stretching across the entire bottom of the screen by switch from using a `RelativeLayout` to a `CoordinatorLayout` in the `fragment_discovery` layout. Because it is in need of an update, I would like to work on the `ListView` adapter which will show the name of the device in large text and the ip address below in smaller text with the format, ip: [ip address]. I will use the post <sup>1</sup> for the task.

Something I discovered is that the application crashes when I select one of the items in the list view after pressing the refresh button. This is because the attempt to update the array list inside the `onClick` method for the fab is illegal as it is running on a separate thread. Instead a custom adapter class must be written with a definition of `notifyDataSetChanged` which will be called if data has changed or new data is available in order to update the list view.

The problem with the `ListView` not updating was resolved by calling `notifyDataSetChanged` on the `ArrayAdapter` object in the `handleMessage` method.

<sup>1</sup><https://www.vogella.com/tutorials/AndroidListView/article.html>

Wednesday, June 24, 2020

---

At the end of the day, all issues were resolved and I am able to successfully discover a RPi on the network with refresh implemented. Changes were pushed to the github.



## Thursday, June 25, 2020

In the biweekly meeting, it was brought up that a disinfection device will be added to the robot consisting of a stepper motor and a pump attached to a tank filled with disinfectant.



## Friday, June 26, 2020

To improve the flow of the app, I worked on adding a local database to the app using Room which provides an abstraction layer over SQLite<sup>1</sup>. Data access objects which must be coded in a Java interface to provide access to the database through methods. To add library support to the project, I included the following in the build.gradle file:

```
def room_version = "2.2.5"

implementation "androidx.room:room-runtime:$room_version"
annotationProcessor "androidx.room:room-compiler:$room_version"
// For Kotlin use kapt instead of annotationProcessor

// optional - Kotlin Extensions and Coroutines support for Room
implementation "androidx.room:room-ktx:$room_version"

// optional - RxJava support for Room
implementation "androidx.room:room-rxjava2:$room_version"

// optional - Guava support for Room, including Optional and ListenableFuture
implementation "androidx.room:room-guava:$room_version"

// Test helpers
testImplementation "androidx.room:room-testing:$room_version"
```

Next, I created a class called Robot that represents a table called Robot with the following columns

- id
- name
- ip

Secondly, I created a database access object titled RobotDao which contains the following methods:

- getAll() which selects rows from the Robot table
- loadByIds() selects rows from the Robot table according to ids

<sup>1</sup><https://developer.android.com/training/data-storage/room>

- `insert()` which will insert its parameters into the database
- `delete()` will delete its parameters from the database

## Monday, June 29, 2020

I was able to determine how to instantiate a new database and perform queries on it using the RobotDao interface. What I need to figure out is how to only insert into the database if the entry to be inserted is unique. For this I wrote the Query method in the RobotDao class titled loadByIp that will select all entries in the database that have ip equal to the same ip that was recently discovered by the app using multicasting:

```
@Query("SELECT * FROM Robot WHERE ip = :ip")
List<Robot> loadByIp(String ip);
```

The problem is I don't know if the query will return null if it fails. For this I will simply try to use the logcat facility which allows printing log messages with a given TAG.

However, when compiling the app, I ran into the compile time error:

```
error: annotation value must be a class literal
```

which was resolved by changing the annotation

```
@Database(entities = (Robot.class), version = 1)
```

to

```
@Database(entities = {Robot.class}, version = 1)
```

However, no log message were printing to the logcat with the line

```
Log.d(TAG, "Size of List: " + Integer.toString(robotsWithIp.size()));
```

which will need to be debugged.



## Tuesday, June 30, 2020

When using the `loadByIp` method, it will return a List of size 0 if a robot with the provided ip is not found in the database. This is the condition that will be checked before adding a new entry to the database when refreshing or on creation of the discovery fragment. In order to avoid adding multiple copies of the same device to the ListView I used a counter variable that gets incremented once when a device is added whether it be loaded from the database or directly from a multicast request:

```
if(mTaskCounterVar == 0) {
    if(robotsWithIp.size() == 0) {
        ipArrayList.add(recvMsg + ": " + ip);
        Robot robot = new Robot(recvMsg, ip);
        db.robotDao().insert(robot);
        mTaskCounterVar++;
        // Else load the robot from the database
    } else if (robotsWithIp.size() > 0) {
        for(Robot robot : robotsWithIp) {
            if(robot != null) {
                ipArrayList.add(robot.name + ": " + robot.ip);
            }
        }
        mTaskCounterVar++;
    }
}
```

Next, what I would like to do is add a feature to be able to switch to the map view when clicking on a device in the list view. Something that Jaden brought up in a meeting yesterday is it would be nice to require some sort of passcode in order to control the robot in case multiple people are connected to the robot at the same time. However, this might be a feature we could add later.

From the post <sup>1</sup>, I was able to research some of the classes need to switch fragment views.

<sup>1</sup><https://stackoverflow.com/questions/23212162/how-to-move-from-one-fragment-to-another-fragment-on-click-of-an-imageview-in-an>





## Thursday, July 2, 2020

The code from user GvSharma was pasted into the onItemClick method for each of the listView elements:

```
Fragment fragment = new MapFragment();
FragmentManager fragmentManager = getActivity().getSupportFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
fragmentTransaction.replace(R.id.content_map, fragment);
fragmentTransaction.addToBackStack(null);
fragmentTransaction.commit();
```

Different code may likely be needed for the listview elements as the code above is meant for buttons objects. Just as a test I placed the above code in the listener for the floating action button; however, to no avail. I posted a question on Stack Overflow to see if anyone had any solutions.



## Monday, July 6, 2020

One of the comments in response to the Stack Overflow post was that the `ListView` is out of date and should be replaced by a `RecyclerView` view object. The first goal will be to switch over to this more up to date and flexible object. The first step is to make the replacement in the layout file for the discovery view. What needs to be done fragment is the follow:

- Get a handle to the `RecyclerView` using `findViewById`
- Call `setHasFixedSize` if the recycler content does not change the size of each entry
- Instantiate a `LayoutManager` object and set it on the `recyclerview` object using `setLayoutManager`
- Specify an adapter on the recycler view using an adapter class that extends `RecyclerView.Adapter`

Secondly, I rewrote the `DiscoveryArrayAdapter` to extend from `RecyclerView.Adapter`<sup>1</sup>. The class contains an inner class titled `ViewHolder` which inherits from `RecyclerView.ViewHolder`. The following method were also added:

- Constructor assigning the string array list passed in to the class level array list
- `onCreateViewHolder` override invoked by the layout manager which will get an instance of the `MyViewHolder` inner class
- `onBindViewHolder` invoked by the layout manager which will take the data passed and assign in to the two `TextView` objects inflated by the adapter
- `getItemCount` override which will return the size of the internal array list

<sup>1</sup><https://developer.android.com/guide/topics/ui/layout/recyclerview>



## Tuesday, July 7, 2020

I received another comment on my post regarding switching between tabs in the tab layout. The user suggested that I do not need `TabLayout` if I simply want to switch between fragments. An even better idea would be to simply use activities instead of fragments for switching between discovering devices and the map view. I will create a new app to test this out in. Also, I would like to use inspiration from the first app tutorial on the android developer site<sup>1</sup>. However, the code I wrote yesterday to migrate the `ListView` over to a `RecyclerView` worked perfectly as intended. As a small step away from the current task, I want to get the onclick listener for the recycler view working. To do this, the `setOnClickListener` method of the `View` object that starts the fragment is defined as follows.

```
holder.mView.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        Toast.makeText(mContext, "Hello at " +  
            Integer.toString(position),  
            Toast.LENGTH_SHORT).show();  
    }  
});
```

Instead of having a toast, I will simply create a new `Intent` object to move to another activity once the app is migrated over from using the `TabLayout`.

After the new Android studio project was started, I worked on figuring how to customize the tool bar. This was done by first changing the project settings. To not have an action bar in the `styles.xml` file. Next a `Toolbar` was added in the `main_activity.xml` file. However, the once I copied all the code from the project using two tabs into the project using only activities, a problem was encountered as no items were being added to the recycler view.

I discovered the source of the problem to be that the first entry in the `RecyclerView` is being hidden by the toolbar in the main activity layout. Therefore, I need to determine how to constrain the top of the `RecyclerView` to the bottom of the toolbar. A more complicated approach was taken by using the `AppBarLayout` which also creates a nested `CollapsingBarLayout`. In all, this allows the top activity bar to be expanded and collapsed. Soon I will get the map activity hooked up to the on click listener for the recycler view items.

<sup>1</sup><https://developer.android.com/training/basics/firstapp/starting-activity>



## Wednesday, July 8, 2020

In the `DiscoveryArrayAdapter` class, I added code to launch another activity from an Intent in the `onClick` listener for each `RecyclerView` item. The following code was added:

```
Intent intent = new Intent(mContext, MapActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
mContext.startActivity(intent);
```

The flag `FLAG_ACTIVITY_NEW_TASK` is needed as the activity is being started from outside an activity context. After the toolbar was successfully added in the `MapActivity` class, a back arrow was added in the upper left hand corner using the following code:

```
getSupportActionBar().setDisplayHomeAsUpEnabled(true);
getSupportActionBar().setDisplayShowHomeEnabled(true);
toolbar.setNavigationOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        finish();
    }
});
```

What I would like to determine is how to make the toolbar not expandable or collapsible. This causes a problem with the back arrow not being visible until the toolbar is fully expanded. I simply needed to remove the inner view from the `AppBarLayout` view that the `TabLayout` is nested under.

To change the text color of the `AppBar` I added a new style to the `styles.xml` which gives the text and back button a color of white. Lastly, I would like to add multiple threads to the python application to have one receiver thread and multiple direct app communication threads. A possible demo could be toggling an LED of the RPi.





## Thursday, July 9, 2020

I worked on copying over the code and project structure from the second project to the `LidarMappingRobot` project we use for the Github repository for the first part of today. After this was finished, I began working on how to remove a robot from the recycler view when the receiver program is closed. What I will need to determine is how to remove from the string array list `mArrayList` using the `remove` method once it is found that the robot is no longer available. I added a new query to the `RobotDao` to load all the ips from the Robot database.



## Saturday, July 11, 2020

The algorithm is documented below for the FindRobotTask class that implements Runnable:

- "Nuke" the table (i.e. delete everything from the Robot table)
- Load all ips from the database
- Search for devices receiving on (224.3.29.71, 10000).
- If the IP of the host (robot) matches an IP already in the database, load the device from the database.
- Else if the IP of the host (robot) is not already in the database, add the device including the name and IP to the database.
- Loop through all the IP addresses in the database and test whether a match is found each time a multicast request is sent out. If a given IP is not found, look up its corresponding ID and remove it from the RecyclerView. Else do nothing. Somehow the ID will need to correspond to the position of the item in the RecyclerView.

One additional algorithm/method that will need to be written is how to check whether any elements in a List type don't exist in a second List type (ArrayList). Both of these Lists will contain IP address, the dbIps List may contain robots that are disconnected. A useful method is the List.contains method which checks if a List contains an object. However, at the end of the day, after adding the necessary code, no robots were being removed from the RecyclerView and none were being added properly in the background threads.



## Monday, July 13, 2020

Because one of the main tasks now is to design a robot in OpenSCAD with multiple different sensors and a disinfectant sprayer, I worked on prototyping the model in OpenSCAD. For a simplicity and a tight turning radius, I would like the robot to have two powered side wheels with a single caster wheel in the front. Both wheels will be powered with an h-bridge motor driver for rotation in two directions. I high level design I am considering is based off <sup>1</sup>.

<sup>1</sup><https://store.arduino.cc/usa/arduino-robot>



## Tuesday, July 14, 2020

Half of the today was spent working on the app and fixing the bug of devices not showing up in the discovery activity. For this purpose I worked on writing log message to logcat with the statement

```
Log.d(TAG,message);
```

Debug log messages cannot be sent to the logcat as they are not loggable. Instead, I use info messages. A mistake I made was including a filter in the logcat. Once no filter was applied. Everything appeared to be fixed and log messages were visible. The problem appears to be in the FindRobotTask where I continue looping with no stopping condition:

```
for(int i = 0; i < 10; i++) {
    boolean uniqueIp = false;
    DatagramPacket recv = new DatagramPacket(buf, buf.length);
    sock.receive(recv);
    ip = recv.getAddress().getHostAddress();
    recvMsg = new String(recv.getData(), 0, recv.getLength());

    // If the IP is not in the database and not the localhost address,
    // add it.
    if(!dbIps.contains(ip) && !ip.equals(localIpAddress)) {
        foundIps.add(ip);
        Robot robot = new Robot(recvMsg, ip);
        db.robotDao().insert(robot);
    }
}
```

A simple solution to this problem is to simply add a break statement after `db.robotDao().insert(robot);` which will break out of the loop once a robot has been found. This fixes the problem if the receiver program is running. However, if not, then the app will timeout and problems will occur: causing the app to still display previous robot running. A potential but kind of bad solution is to use

```
MulticastSocket.setSoTimeout(int timeout);
```

which will set the timeout in milliseconds.

Another problem I am facing is how the device will not show up after shutting down the receiver program and restarting it. The device however will disappear from the view if the receiver program is stopped (using CTRL + C). Another problem arose when attempting to refresh when the device is already listed on the display as a `SocketException` is thrown when the call to receive times out.





## Wednesday, July 15, 2020

Prototyping on the new robot was worked on today. What will likely need to be changed is the wheel positions as they collide with the bottom base. Potential solutions include moving the wheels partially into the base with two horizontal slots similar to the configuration by Arduino. Axles will be connected to the wheels through DC motors mounted to the bottom side of the base. Power wires will be connected to the motors from the Beaglebone. Below is the current model of the robot in OpenSCAD.

