User Documentation

Team Android Optimizers

Jose Franco Baquera
Atiya Kailany
Jared Peterson

CS 448
December 7, 2019

Table of Contents

1. Introduction

In this document, we will describe in detail the developer and user manuals, as well as how to install and run the application on an Android Virtual Device. The instructions on this document were specifically written for the *Autonomous Robotic Vehicle (ARV) Application* project. We note that the developer manual will be used by those who wish to continue implementing and modifying the source code while the *Android Virtual Device* installation guide will be used by those who wish to test the application in an emulator rather than on a physical tablet. Lastly, the user manual will describe how to use the application and will assume that it was successfully installed and run on either an emulator or physical tablet.

2. Developer Manual

In this section, we discuss the developer manual that will be used by anyone who wishes to continue implementing and modifying the project. It is important to note that the following list of steps should be completed in a sequential manner.

2.1 Hardware and Memory Requirements Check

Before installing Android Studio on your computer, please make sure that your system meets the following list of memory requirements:

- The computer must have at least 8GB of free disk or solid-state drive memory. This memory is needed to store not only the integrated development environment itself, but also emulator files that will be created in the future.

- It is recommended that the computer has at least 4GB of RAM memory.

We note that the computer must also have a 64-bit operating system and 64-bit processors. More precise system requirements can be found in the *System Requirements* section of the following official website: https://developer.android.com/studio

2.2 Operating System Recommendation

It is recommended to download Android Studio on either a Windows or OS X machine. If a Linux machine is chosen, please be aware that you might need sudo or root privileges. Generally speaking, downloading and installing Android Studio on either a Windows or OS X machine is much easier and smoother than on a Linux machine.

2.3 Download and Installation of Android Studio

To download and install the newest version of Android Studio, please click on the following link: https://developer.android.com/studio

The website will automatically recommend the newest Android Studio version that best suits your computer's operating system and hardware. To download and install Android Studio, press the green and large *DOWNLOAD ANDROID STUDIO* button. While installing the development environment, please make sure to check the *Android Virtual Device* and *Do Not Import Settings* boxes. Furthermore, please choose the *Standard* install type and not the *Custom* one. All other configurations should be left to *recommended, automatic*, or anything that was already pre-selected. If done correctly, you should see the following dialog box after the installation is completed:

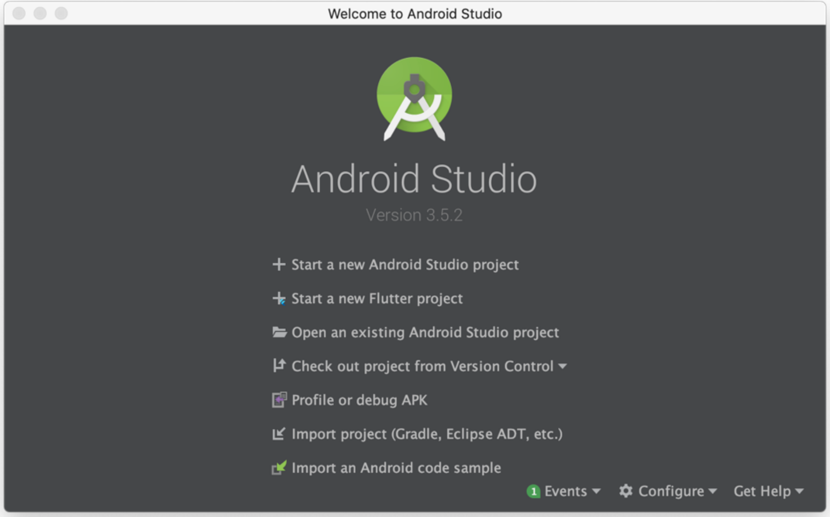

Welcome to Android Studio
Android Studio
Version 3.5.2
+ Start a new Android Studio project
Start a new Flutter project
Open an existing Android Studio project
Check out project from Version Control ▾
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
1 Events ▾
Configure ▾
Get Help ▾

Figure 1– Screenshot of dialog box that pops up after downloading, installing, and running Android Studio.

It is *critical* to note that our project is compatible only with Android Studio 3.5 or higher. That is, earlier versions of Android Studio will not compile our source code and Gradle files.

If you have trouble downloading and installing Android Studio, please refer to online resources or YouTube videos that provide more information about the installation process. Downloading and installing Android Studio should, however, be easy and quick.

2.4 Downloading GitHub Source Code and Opening Existing Project

Download the project's source code from GitHub by clicking on the following link: https://github.com/PIxLLab/MapInterface-AndroidApp

Please be aware that this is a private GitHub repository, meaning that you must be invited to it in order to access it. If you wish to request access, please contact Ahmed Khalaf (khalaf@nmsu.edu) or Dr. Toups (z@cs.nmsu.edu) since they are the only individuals who can grant such requests. We note that access to the GitHub repository is needed only if you do not have the *ARVWearable* folder with all the required source code.

Once you have the required files, click on the *Open an existing Android Studio project* option and open the **entire** *ARVWearable* folder. It is critical to note that if you downloaded the required files from GitHub, the *ARVWearable* folder will be inside the *MapInterface-AndroidApp* folder. **Do not open the *MapInterface-AndroidApp* folder as this is incorrect and will make the project not compile.**

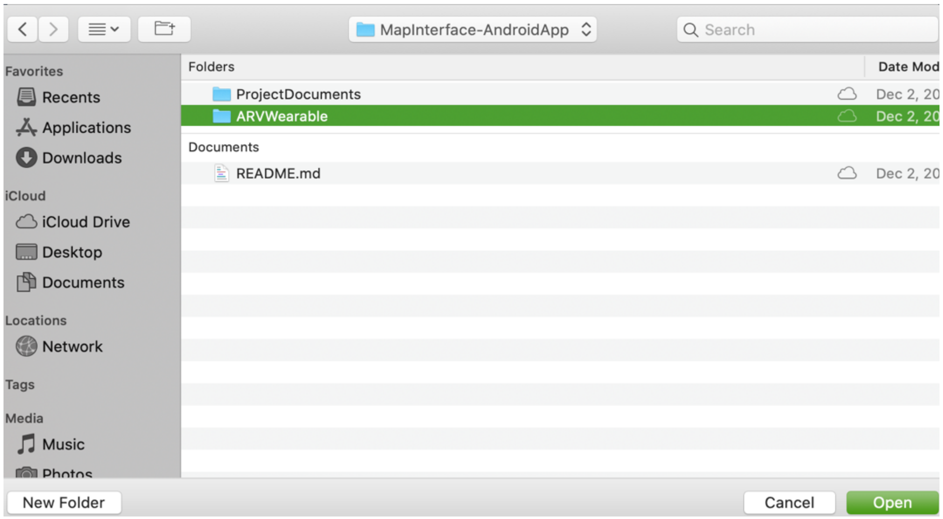
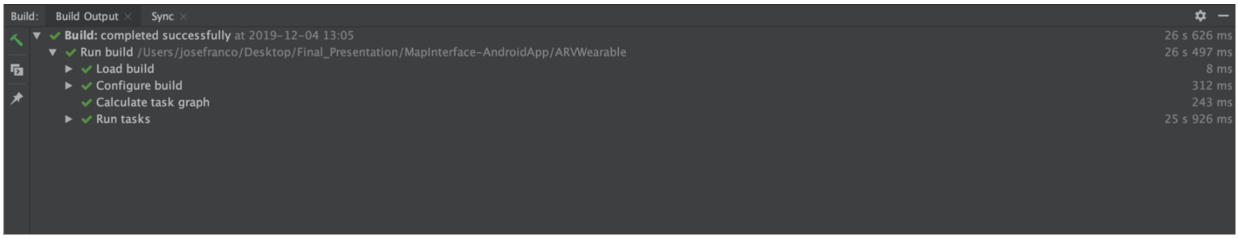


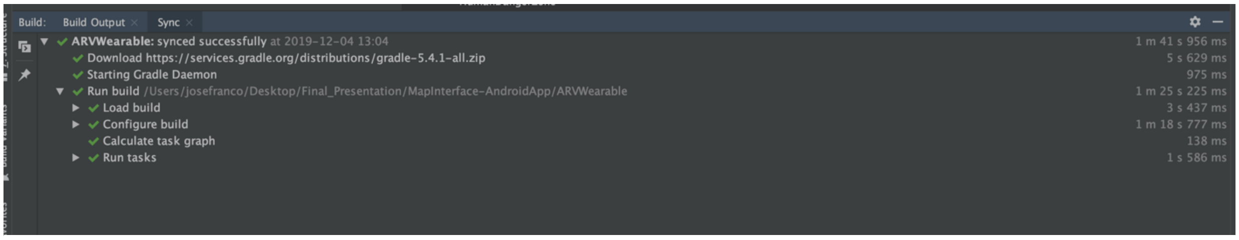
Figure 2 – Screenshot of *ARVWearable* folder needed to run the application. This folder will be inside the *MapInterface-AndroidApp* folder if the files were downloaded from GitHub.

Android Studio will automatically build and sync the project once it is opened. A successful project sync and build will have all "green checkmarks," as illustrated in Figures 3 and 4.



```
▼ ✓ Build: completed successfully at 2019-12-04 13:05                                                       26 s 626 ms
  ▼ ✓ Run build /Users/josefranco/Desktop/Final_Presentation/MapInterface-AndroidApp/ARVWearable            26 s 497 ms
    ▶ ✓ Load build                                                                                                 8 ms
    ▶ ✓ Configure build                                                                                          312 ms
      ✓ Calculate task graph                                                                                     243 ms
    ▶ ✓ Run tasks                                                                                           25 s 926 ms
```

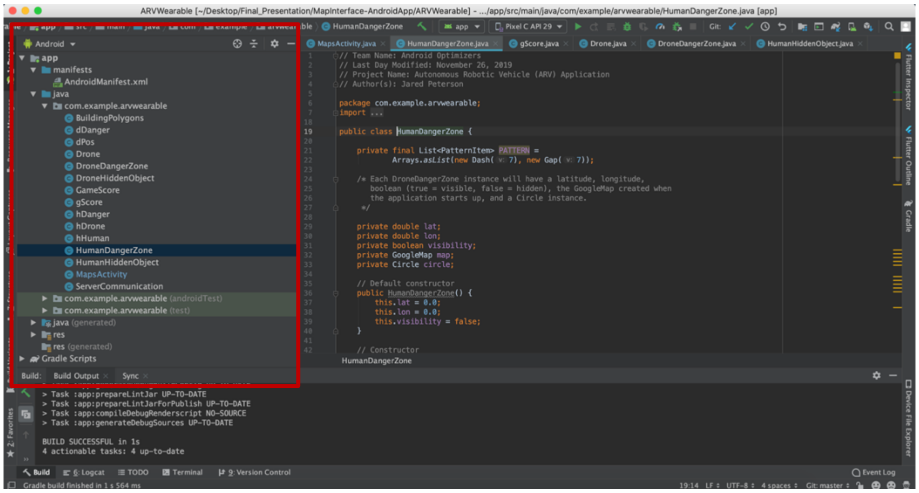
Figure 3 – Screenshot of a successful build.



```
▼ ✓ ARVWearable: synced successfully at 2019-12-04 13:04                                                 1 m 41 s 956 ms
    ✓ Download https://services.gradle.org/distributions/gradle-5.4.1-all.zip                                 5 s 629 ms
    ✓ Starting Gradle Daemon                                                                                     975 ms
  ▼ ✓ Run build /Users/josefranco/Desktop/Final_Presentation/MapInterface-AndroidApp/ARVWearable         1 m 25 s 225 ms
    ▶ ✓ Load build                                                                                            3 s 437 ms
    ▶ ✓ Configure build                                                                                  1 m 18 s 777 ms
      ✓ Calculate task graph                                                                                     138 ms
    ▶ ✓ Run tasks                                                                                             1 s 586 ms
```

Figure 4 – Screenshot of a successful sync.

If the project is opened successfully, you will be able to modify the source code and other aspects of our project. Figure 5 illustrates where our source code can be located.



```
// Team Name: Android Optimizers
// Last Day Modified: November 26, 2019
// Project Name: Autonomous Robotic Vehicle (ARV) Application
// Author(s): Jared Peterson

package com.example.arvwearable;
import ...

public class HumanDangerZone {

    private final List<PatternItem> PATTERN =
            Arrays.asList(new Dash( v: 7), new Gap( v: 7));

    /* Each DroneDangerZone instance will have a latitude, longitude,
       boolean (true = visible, false = hidden), the GoogleMap created when
       the application starts up, and a Circle instance.
     */

    private double lat;
    private double lon;
    private boolean visibility;
    private GoogleMap map;
    private Circle circle;

    // Default constructor
    public HumanDangerZone() {
        this.lat = 0.0;
        this.lon = 0.0;
        this.visibility = false;
    }

    // Constructor
```

```
> Task :app:prepareLintJar UP-TO-DATE
> Task :app:prepareLintJarForPublish UP-TO-DATE
> Task :app:compileDebugRenderscript NO-SOURCE
> Task :app:generateDebugSources UP-TO-DATE

BUILD SUCCESSFUL in 1s
4 actionable tasks: 4 up-to-date
```

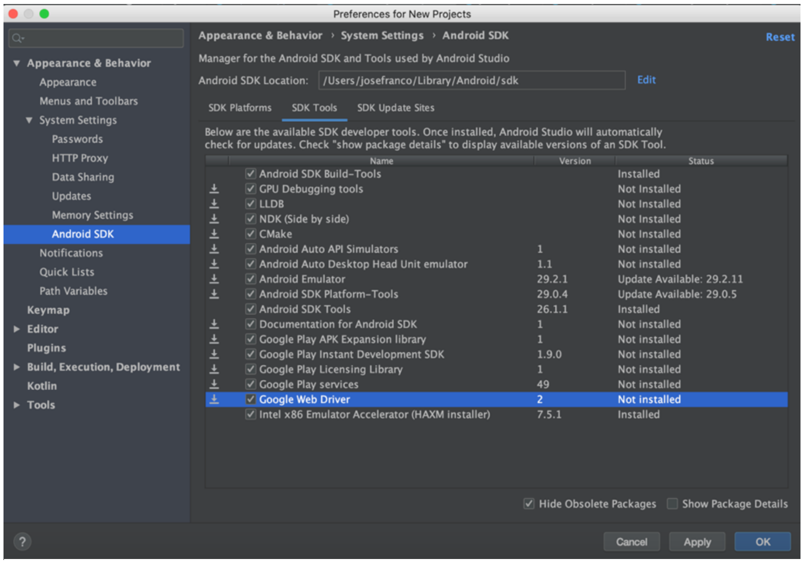
Gradle build finished in 1 s 564 ms

Figure 5 – Screenshot of what can be displayed if the project is opened successfully. Source files can be found in the *app* drop down menu on the left-hand side.

2.5 Installation of Android SDK Tools

Before compiling or running the program, please download and install all Android SDK tools. This can be accomplished by completing the following list of steps:

- Click *Tools.*
- Click on *SDK Manager.*
- Once the *Preferences for New Projects* dialog box is opened, select the *SDK Tools* submenu and select all available tools (see Figure 4).
- Click *OK* and wait for all tools to download.



Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: /Users/josefranco/Library/Android/sdk

Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
☑ Android SDK Build-Tools		Installed
☑ GPU Debugging tools		Not Installed
☑ LLDB		Not Installed
☑ NDK (Side by side)		Not Installed
☑ CMake		Not Installed
☑ Android Auto API Simulators	1	Not installed
☑ Android Auto Desktop Head Unit emulator	1.1	Not installed
☑ Android Emulator	29.2.1	Update Available: 29.2.11
☑ Android SDK Platform-Tools	29.0.4	Update Available: 29.0.5
☑ Android SDK Tools	26.1.1	Installed
☑ Documentation for Android SDK	1	Not installed
☑ Google Play APK Expansion library	1	Not installed
☑ Google Play Instant Development SDK	1.9.0	Not installed
☑ Google Play Licensing Library	1	Not installed
☑ Google Play services	49	Not installed
☑ Google Web Driver	2	Not installed
☑ Intel x86 Emulator Accelerator (HAXM installer)	7.5.1	Installed

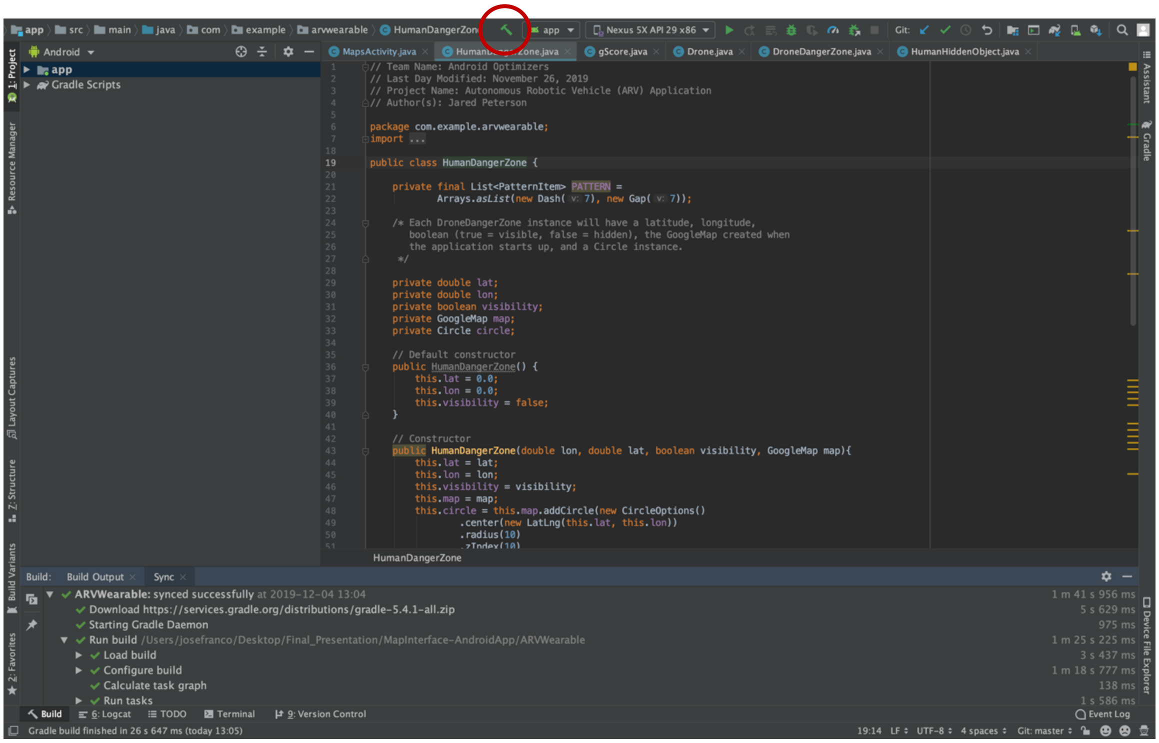
☑ Hide Obsolete Packages ☐ Show Package Details

Figure 6 – Screenshot of all potential tools that need to be downloaded before the project is compiled or run.

It is worth mentioning that Android Studio might need to be manually restarted after all selected tools are downloaded.

2.6 Building Project After Modifications

After all tools are downloaded, you can build the project (i.e. if it has not been automatically done by Android Studio) by simply clicking the *Make Project* button located on top of the window. The *Make Project* button should be pressed any time modifications are made since it checks for compiling, building, and syncing errors. Figure 7 illustrates the location of this button within the development environment.



```
// Team Name: Android Optimizers
// Last Day Modified: November 26, 2019
// Project Name: Autonomous Robotic Vehicle (ARV) Application
// Author(s): Jared Peterson

package com.example.arvwearable;
import ...

public class HumanDangerZone {

    private final List<PatternItem> PATTERN =
            Arrays.asList(new Dash( v: 7), new Gap( v: 7));

    /* Each DroneDangerZone instance will have a latitude, longitude,
       boolean (true = visible, false = hidden), the GoogleMap created when
       the application starts up, and a Circle instance.
     */

    private double lat;
    private double lon;
    private boolean visibility;
    private GoogleMap map;
    private Circle circle;

    // Default constructor
    public HumanDangerZone() {
        this.lat = 0.0;
        this.lon = 0.0;
        this.visibility = false;
    }

    // Constructor
    public HumanDangerZone(double lon, double lat, boolean visibility, GoogleMap map){
        this.lat = lat;
        this.lon = lon;
        this.visibility = visibility;
        this.map = map;
        this.circle = this.map.addCircle(new CircleOptions()
                .center(new LatLng(this.lat, this.lon))
                .radius(10)
                .zIndex(10)
```

```
▼ ✓ ARVWearable: synced successfully at 2019-12-04 13:04                                1 m 41 s 956 ms
    ✓ Download https://services.gradle.org/distributions/gradle-5.4.1-all.zip             5 s 629 ms
    ✓ Starting Gradle Daemon                                                                  975 ms
  ▼ ✓ Run build /Users/josefranco/Desktop/Final_Presentation/MapInterface-AndroidApp/ARVWearable   1 m 25 s 225 ms
    ▶ ✓ Load build                                                                           3 s 437 ms
    ▶ ✓ Configure build                                                                 1 m 18 s 777 ms
      ✓ Calculate task graph                                                                  138 ms
    ▶ ✓ Run tasks                                                                            1 s 586 ms
Gradle build finished in 26 s 647 ms (today 13:05)
```

Figure 7 – Screenshot demonstrating where the *Make Project* button is located within the development environment.

2.7 Testing Project and Application on Emulator

Please refer to section 3 to learn how to download and run an application on an Android Virtual Device.

2.8 Troubleshooting

At times, some building and syncing errors can be fixed by simply updating Android Studio and other SDK tools. Because of this, please check for updates regularly and download any new releases since this may help prevent such errors.

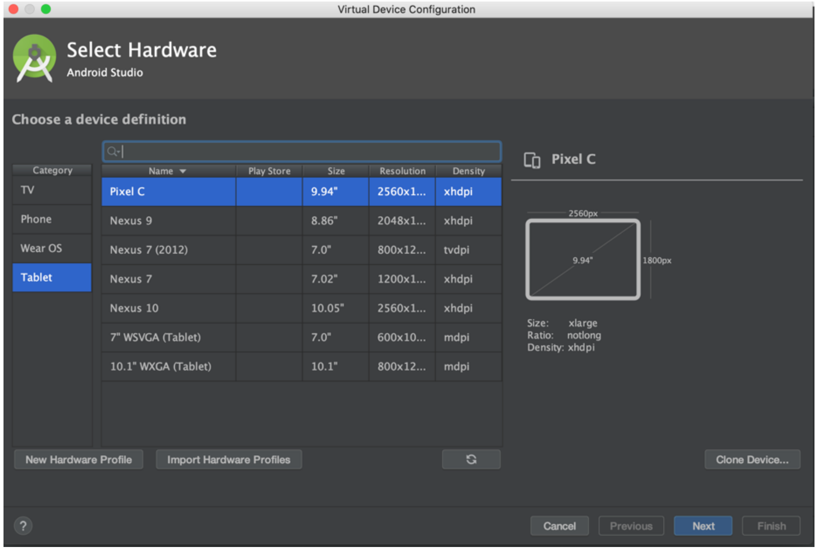
3. Android Virtual Device Installation and Running Application Guide

In this section, we discuss the *Android Virtual Device* installation and running application guide that will be used by anyone who wishes to test the application in an emulator rather than on a physical tablet. We assume that users completed section 2 successfully before attempting to run the application on an emulator. It is important to note that the following list of steps should be completed in a sequential manner.

3.1 Create a New Android Virtual Device

In order to run the application on an emulator, you first have to create a new virtual device. This can be accomplished by completing the following list of steps:

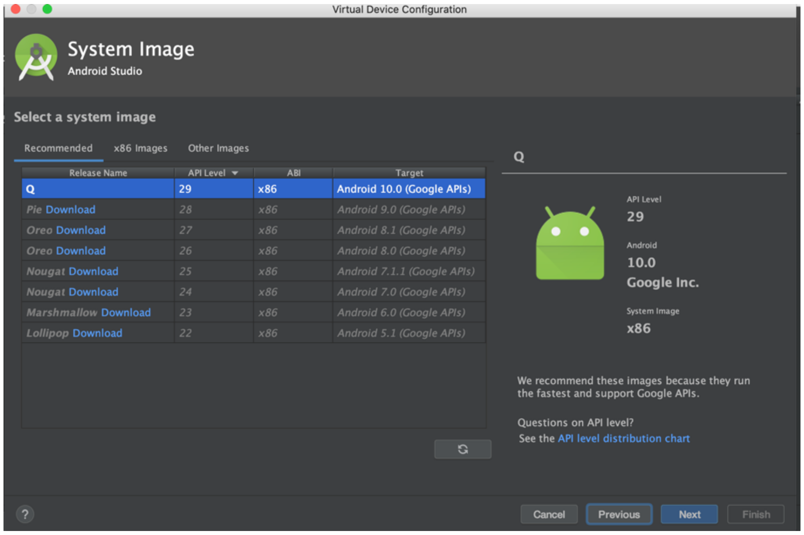
- Click *Tools.*
- Click on *AVD Manager.*
- Once the *Android Virtual Device Manager* dialog box is opened, click on the *+Create Virtual Device…* button.
- In the category section, press the *Tablet* option, then select any tablet configuration. We do, however, suggest using Pixel C (see Figure 8).
- Click *Next.*
- Though various different system images work with this application, we do suggest choosing *Q,* which is the system image that maps to an API level of 29 (see Figure 9). We note that other API levels might run the application much slower and that you might need to download the system image before it can be selected.
- Click *Next.*
- Name the *Android Virtual Device* as you please. Make sure that the startup orientation is setup to *Landscape* mode (see Figure 10).
- Click *Finish.* Your new emulated tablet should now be ready to run.



Name	Play Store	Size	Resolution	Density
Pixel C		9.94"	2560x1...	xhdpi
Nexus 9		8.86"	2048x1...	xhdpi
Nexus 7 (2012)		7.0"	800x12...	tvdpi
Nexus 7		7.02"	1200x1...	xhdpi
Nexus 10		10.05"	2560x1...	xhdpi
7" WSVGA (Tablet)		7.0"	600x10...	mdpi
10.1" WXGA (Tablet)		10.1"	800x12...	mdpi

Size: xlarge
Ratio: notlong
Density: xhdpi

Figure 8 – Screenshot of what will be seen while choosing an Android tablet. We suggest choosing Pixel C, though any tablet will suffice.



Select a system image

Release Name	API Level	ABI	Target
Q	29	x86	Android 10.0 (Google APIs)
Pie Download	28	x86	Android 9.0 (Google APIs)
Oreo Download	27	x86	Android 8.1 (Google APIs)
Oreo Download	26	x86	Android 8.0 (Google APIs)
Nougat Download	25	x86	Android 7.1.1 (Google APIs)
Nougat Download	24	x86	Android 7.0 (Google APIs)
Marshmallow Download	23	x86	Android 6.0 (Google APIs)
Lollipop Download	22	x86	Android 5.1 (Google APIs)

Q

API Level: 29

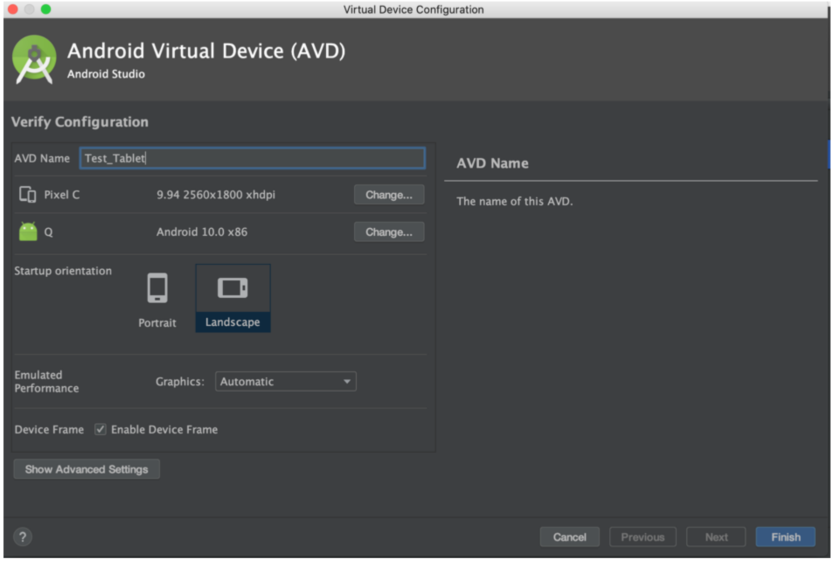
Android: 10.0 Google Inc.

System Image: x86

We recommend these images because they run the fastest and support Google APIs.

Questions on API level?
See the API level distribution chart

Figure 9 – Screenshot of what will be seen while selecting a system image. We recommend choosing *Q* (i.e. API level 29) since other API levels might make the application run slower.



Verify Configuration

AVD Name: Test_Tablet

Pixel C 9.94 2560x1800 xhdpi

Q Android 10.0 x86

Startup orientation: Landscape

Emulated Performance Graphics: Automatic

Device Frame ☑ Enable Device Frame

AVD Name

The name of this AVD.

Figure 10 – Screenshot of final step before creating the *Android Virtual Device*. We note that the startup orientation should be selected to *Landscape*.

3.2 Running the Android Virtual Device on an Emulator

After creating the new *Android Virtual Device,* it can now be run on an Android emulator. This can be accomplished by completing the following list of steps:

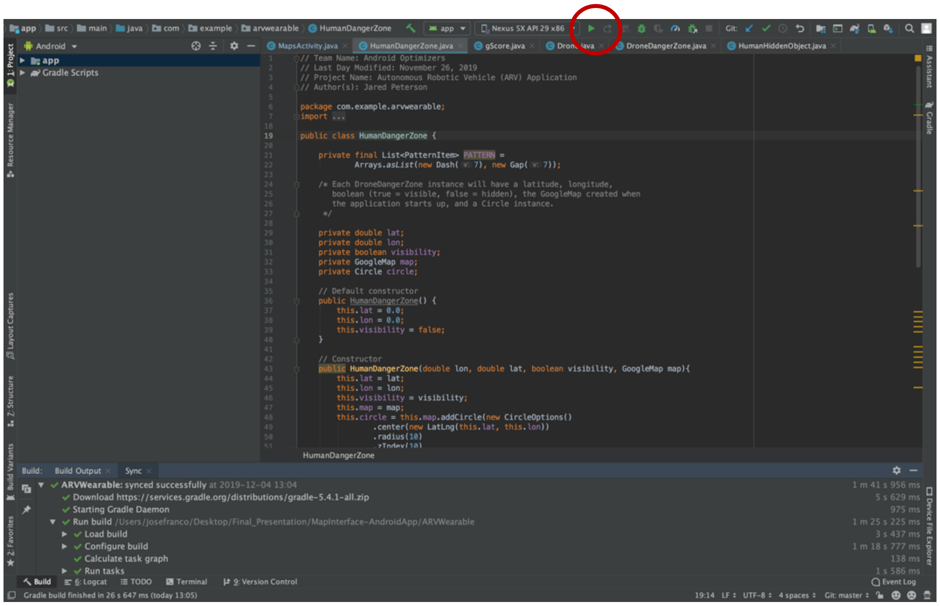
- Click *Tools.*
- Click on *AVD Manager.*
- Click on the “green arrow” next your newly created *Android Virtual Device.* This will launch your *Android Virtual Device* on the emulator.

3.3 Build Precondition

Before trying to run the application on the running emulator, you must build and sync the project. Please refer to section 2.6 on how to build and sync the imported project.

3.4 Downloading and Running the Application on the Emulator

You can download and run the application on the emulator by simply clicking the *Run 'app'* button located on top of the window. Figure 11 illustrates the location of this button within the development environment.



```
// Team Name: Android Optimizers
// Last Day Modified: November 26, 2019
// Project Name: Autonomous Robotic Vehicle (ARV) Application
// Author(s): Jared Peterson

package com.example.arvwearable;
import ...

public class HumanDangerZone {

    private final List<PatternItem> PATTERN =
            Arrays.asList(new Dash( v: 7), new Gap( v: 7));

    /* Each DroneDangerZone instance will have a latitude, longitude,
       boolean (true = visible, false = hidden), the GoogleMap created when
       the application starts up, and a Circle instance.
     */

    private double lat;
    private double lon;
    private boolean visibility;
    private GoogleMap map;
    private Circle circle;

    // Default constructor
    public HumanDangerZone() {
        this.lat = 0.0;
        this.lon = 0.0;
        this.visibility = false;
    }

    // Constructor
    public HumanDangerZone(double lon, double lat, boolean visibility, GoogleMap map){
        this.lat = lat;
        this.lon = lon;
        this.visibility = visibility;
        this.map = map;
        this.circle = this.map.addCircle(new CircleOptions()
                .center(new LatLng(this.lat, this.lon))
                .radius(10)
                .zIndex(10)
```

```
▼ ✓ ARVWearable: synced successfully at 2019-12-04 13:04                                   1 m 41 s 956 ms
    ✓ Download https://services.gradle.org/distributions/gradle-5.4.1-all.zip                    5 s 629 ms
    ✓ Starting Gradle Daemon                                                                         975 ms
  ▼ ✓ Run build /Users/josefranco/Desktop/Final_Presentation/MapInterface-AndroidApp/ARVWearable 1 m 25 s 225 ms
    ▶ ✓ Load build                                                                               3 s 437 ms
    ▶ ✓ Configure build                                                                     1 m 18 s 777 ms
      ✓ Calculate task graph                                                                         138 ms
    ▶ ✓ Run tasks                                                                                1 s 586 ms
Gradle build finished in 26 s 647 ms (today 13:05)
```

Figure 11 – Screenshot demonstrating where the *Run 'app'* button is located within the development environment.

3.5 Internet Connection

You must be connected to the internet in order to use the application.

4. User Manual

In this section, we discuss how to use the application with the assumption that it was successfully installed and run in either an emulator or physical tablet. We note that Figure 12 illustrates a screenshot of the user interface and maps the functionality of each button to all numbers outlined in sections 4.1 through 4.5.

4.1 Selecting a Drone

To select a drone, simply scroll through the *Drone Selector* carousel located at the bottom left-hand corner **(1)**. It is important to note that *all* drone commands chosen will be applied to any drone currently selected. That is, two drones cannot be modified at the same time unless different drones are selected from the *Drone Selector* carousel.

4.2 Sending a Drone to a Location

To send a drone to a location, first select a coordinate on the map. Then, press the *Send* button located at the bottom right-hand corner **(2)**.

4.3 Sending a Drone to Search

To make a drone search an area, first select a coordinate on the map. Then, press the *Search* button located at the bottom right-hand corner **(3)**. Lastly, select another coordinate on the map. It is worth noting that the *Search* command requires users to choose two coordinate points in total.

4.4 Landing a Drone

To land a drone, simply press the *Land* button located at the bottom right-hand corner **(4)**.

4.5 Modifying a Drone's Altitude

To change the altitude of a drone to high, simply press the *High-Altitude* button located at the top left-hand corner **(5)**. Similarly, to change the altitude of a drone to low, press the *Low-Altitude* button located at the top left-hand corner **(6)**.

4.6 Screenshot of User Interface and Mapping of Buttons

The following screenshot maps the corresponding buttons to their functionality:

(1): *Drone Selector* **(2)**: *Send* **(3)**: *Search* **(4)**: *Land* **(5)**: *High-Altitude* **(6)**: *Low-Altitude*

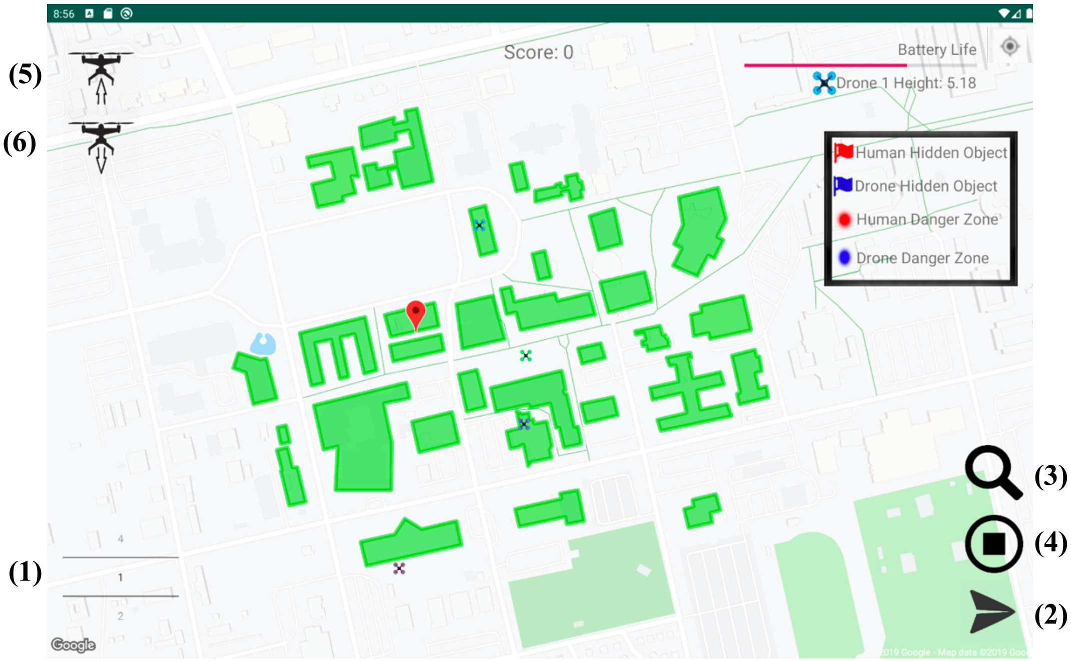


Figure 12 – Screenshot of user interface and the mappings between its buttons and their functionalities.

4.7 Delay Between Commands Warning

Because of the nature of the API used for this project, there might be a small delay when modifying a drone. Please allow a few seconds for the user interface to display any changes. If no changes are displayed after a long period of time, it is more than likely that the server froze (see section 4.9).

4.8 Drone One Bug

As of the writing of this document, it is sometimes difficult to modify drone one due to a bug found in *Spitfire*. We note that fixing this bug is outside our scope of work since it involves modifying and reprogramming *Spitfire*. We recommend, therefore, using drones two through four when running the application since there is no guarantee that drone one will be working at any given time.

4.9 Troubleshooting

The following troubleshooting tips will provide helpful information that was learned during the development of this project:

- *Server Down* – If the application displays a *Server Down* error message, then *Spitfire* has been turned off. Please contact Ahmed Khalaf or Dr. Toups as they are the only ones authorized to turn it back on.

- *Frozen Drones* – If *Spitfire* freezes, then drones will not change in altitude or location after a command is pressed, even after waiting for a long time. Please contact Ahmed Khalaf or Dr. Toups as they are the only ones authorized to restart *Spitfire*.

- *Game Engine* – Unlike *Spitfire,* the game engine is not constantly on, meaning that you will not be able to see hidden objects or dangers zones. Please contact Ahmed Khalaf or Dr. Toups as they are the only ones authorized to turn on the game engine.

- *Emulated Tablet Not Rotating* – The emulated tablet will not go into landscape mode if its *Auto-rotate* setting is not turned on (see Figure 13).

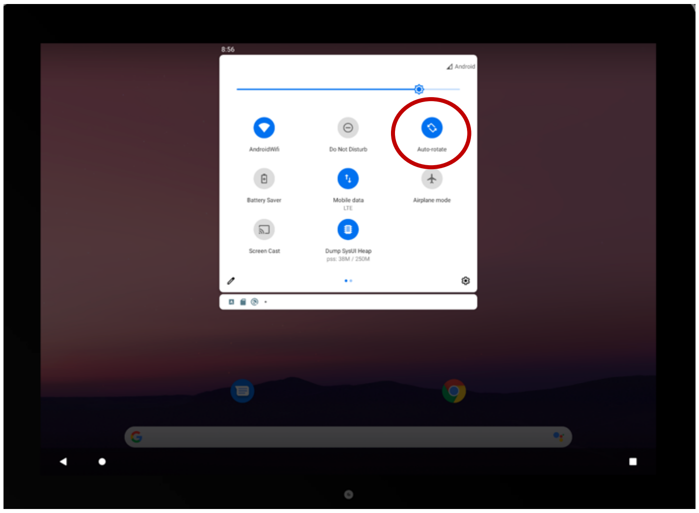


Figure 13 – Screenshot of where the Auto-rotate setting is found. This setting must be turned on in order for the tablet to successfully "rotate."