Android App Development

Importance in the LFEV project and basics of writing apps

Rameel Sethi
Lafayette College
Department of Electrical and Computer Engineering
Easton, Pennsylvania
sethir@lafayette.edu

***Abstract*— This document provides a discussion of the need for an Android version of a vehicle SCADA (Supervisory Control and Data Acquisition) application for the LFEV (Lafayette Formula Electric Vehicle) project and justifies development for Android over iOS. A brief overview of Android development is then provided (assuming the reader has basic knowledge of the programming language Java) and provides a step-by-step of a rudimentary 'Hello, world' Android application.**

Keywords*— *Android, LFEV, SCADA, Java, XML, activity, layout

I. INTRODUCTION

The LFEV (Lafayette Formula Electric Vehicle) project is a senior design project for Lafayette College's Electrical and Computer Engineering Department that has been running since 2013. The objective is to design an electric vehicle to compete in the Electric Vehicle category of the Formula Hybrid student competition. Now in the project's third year, many of the on-vehicle systems are expected to be competition-ready within a year or two. The vehicle software must perform two critical functions: data acquisition from hardware of all other subsystems, and control of the vehicle system state. There is a requirement for several different physical user interfaces to be present through which the SCADA system can be accessed, one of them being an Android cell phone application. This paper is intended to serve as a starting point for future project team members to consider the importance of developing an Android SCADA app and provide them a short introduction to Android development.

II. IMPORTANCE OF AN ANDROID APP FOR THE LFEV PROJECT

It may seem that an Android SCADA app is redundant since the necessary functions of data acquisition and system control can be performed on a desktop or laptop computer with much more processing power and screen space than an Android device. However, there are two main reasons why it is desirable to include an Android app as part of the vehicle SCADA system.

The first and most important reason is that in the actual Formula EV competition, the racecar technicians from the Lafayette Formula EV team need to be monitoring data and system state from the SCADA app while being as close to the racetrack as possible to tend to the car should some unsafe condition or other maintenance issue occur. Accomplishing both these tasks would be much easier with a cell phone on hand rather than carrying a desktop or laptop machine.

Another reason to warrant the development of an Android app is to serve as a demo app to showcase the functionality of the SCADA system to ECE department visitors and prospective students. The ability to download the app to their cell phones and see the SCADA functionality for themselves would spark their interest in the LFEV project as well as the ECE department, which would potentially result in stronger, more dedicated future LFEV teams. Fig. 1 shows an example of an Android vehicle SCADA app (called Prowl Torque)[1].

Fig. 1 Prowl Torque Android vehicle SCADA app

II. CHOOSING BETWEEN ANDROID AND iOS DEVELOPMENT

The Android operating system faces iOS as its major competitor. Mobile app developers are more often than not faced with a tough choice: whether to develop their app for Android or iOS (if not both). Table 1 shows a list of major differences developers must consider when selecting a mobile development platform. For LFEV purposes, ease of app store publication is the deciding factor in choosing Android.

TABLE I
DIFFERENCES BETWEEN ANDROID AND iOS DEVELOPMENT

Android	iOS
Java; less verbose (compared to Objective-C)	Objective-C; extremely verbose
Any dev OS fine	Need Mac OS X to dev
Publishing to Play Store easy	Difficult process of publishing to App Store
Free to develop	$99/yr to join iOS Developer Program
Many different devices and screen sizes need to be catered to	Max of 3 devices (iPod, iPhone, iPad) with known screen sizes
Eclipse IDE hard-to-use (Android Studio now official IDE though buggy)	Xcode mature IDE
Clunky drag-and-drop UI editor; need to write tons of XML	Easy-to-use Interface Builder

It is equally important for developers to consider their user platform demographic, which varies by location. In North America, iOS may be the platform of choice. However, in other areas of the world, Android usage is either equal to or far greater than that of iOS. Fig. 2 shows priority platform usage across the world[2].

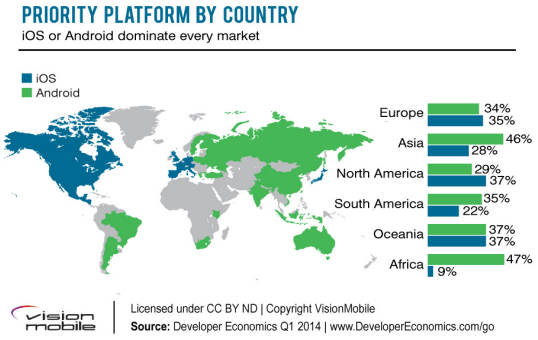

Fig. 2 Prowl Torque Android vehicle SCADA app

III. THE ANDROID OPERATING SYSTEM

The Android operating system was first released in 2007 beginning with version 1.5 'Cupcake'; the current version is version 5.1 'Lollipop'. It is based on the Linux kernel and previously used ANT for its build system, but has recently switched to Gradle.

Android apps are primarily written in Java, and since the purpose of the rest of this paper is to give future teams a short introduction to Android app development, we will assume that the reader has a working knowledge of Java and understands the terms *class, subclass, instance, extending, method and overriding* in the context of the Java programming language. User interface layouts are done in XML (Extensible Markup Language); however, XML is easy enough to understand without prior usage since it is a markup language rather than a programming language. The Android Studio IDE and SDK Tools are available as a free download for Windows, Mac OS X and Linux at:

http://developer.android.com/sdk/index.html

A. Android - Activity

An Activity is an instance of the *Activity* class in the Android SDK. The Activity manages user interaction with the screen and all UI widgets present. It is the developer's task to provide the app's functionality by extending the Activity class. The resulting Activity subclass often has methods overriding implementations of standard Activity class methods managing user interaction, as well as user-defined methods.

B. Android - Layout

A layout defines the set of user interface objects and their position on the screen. It is composed of XML definitions which create widgets on the screen, such as text fields, buttons and sliders. The resulting widgets exist in a hierarchy of View objects called the view hierarchy. XML tags are similar to HTML tags which make up web pages, except that whereas HTML has a fixed set of tags, XML allows user-defined tags. At runtime, the XML tags are parsed and widgets corresponding to each tag are populated on the screen.

IV. A SIMPLE ANDROID APP - GEOQUIZ

We shall now dive headfirst into developing a simple 'Hello, world' level Android app called GeoQuiz[3], which is shown in Fig. 3. This app poses a statement 'Constantinople is the captial of Turkey', with True and False buttons. Pressing True leads to an 'Incorrect' message being displayed, while pressing 'False' leads to a 'Correct' message begin displayed (of course, the above is debatable since Constantinople was the old name of the capital of Turkey until it was renamed to Istanbul).

Fig. 3 GeoQuiz app

A. Layout Design

The first step is to write the XML corresponding to the above layout. Fig. 4 shows the screen in Fig. 3 with the individual UI widgets highlighted.

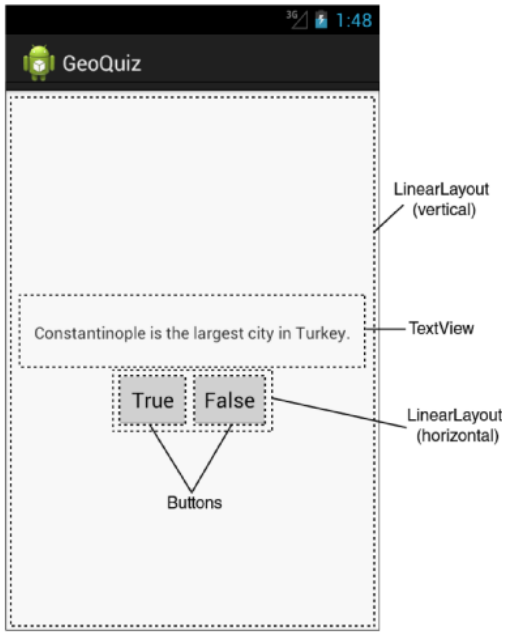

Fig. 4 GeoQuiz layout with widgets highlighted

The layout in Fig. 4 translates to the XML below in layout/activity_quiz.xml. Note that the root LinearLayout tag is vertical in orientation (to make subsequent widgets be placed below each other) but the LinearLayout tag enclosing the two Button tags is horizontal (making the True/False buttons side-by-side). Also note there is no tag corresponding to the Correct/Incorrect message (called a toast) since this is displayed by the methods for the True/False buttons by accessing a separate Toast class.

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:gravity="center"
android:orientation="vertical" >
        <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="24dp"
        android:text="@string/question_text" />
        <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal" >
                <Button
                android:id="@+id/true_button"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/true_button" />
                <Button
                android:id="@+id/false_button"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/false_button" />
        </LinearLayout>
</LinearLayout>
```

It is good practice to declare strings in a string resource file in res/strings.xml and refer to them in the XML layout or Java Activity subclass.

```
<?xml version="1.0" encoding="utf-8"?>

<resources>

        <string name="app_name">GeoQuiz</string>

        <string name="question_text">Constantinople is
the largest city in Turkey.</string>

        <string name="true_button">True</string>

        <string name="false_button">False</string>

        <string name="correct_toast">Correct!</string>

        <string name="incorrect_toast">Incorrect!</string>

        <string name="menu_settings">Settings</string>

</resources>
```

A. The Quiz Activity - QuizActivity.java

Let us now extend the Activity class to provide the functionality for the GeoQuiz app. We begin by creating two Button objects to represent the two buttons. The onCreate() method is present for all activities and serves to initialize the activity state, possibly with data passed from other activities if there are multiple in an app. It also inflates the XML into UI widgets.

```
public class QuizActivity extends Activity {
   private Button mTrueButton;
   private Button mFalseButton;


   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_quiz);
   }
```

We next fetch the True button widget from the layout and set a listener to perform an action when the button is pressed. We do so by overriding the listener's onClick() method to make a toast within the app saying 'Incorrect'. This is done by accessing the Toast object, changing its text to 'Incorrect' and displaying the toast. A similar method override is done for the False button, but with the toast displaying 'Correct' instead.

```
   mTrueButton = (Button)findViewById(R.id.true_button);
   mTrueButton.setOnClickListener(new
View.OnClickListener() {
      @Override
      public void onClick(View v) {
         Toast.makeText(QuizActivity.this,
                        R.string.incorrect_toast,
                        Toast.LENGTH_SHORT).show();
      }
   });
}
```

V.CONCLUSIONS

In this paper we have looked at the reasons for developing a mobile version of a SCADA application for the Lafayette Formula EV project, the reasons for choosing Android over iOS for the application platform, and a quick introduction to Android app development. It is hoped that the 2016 and beyond LFEV teams will read this paper and gain inspiration to develop the Android SCADA app.

REFERENCES

[1] http://play.google.com/store/apps/details?id=org.prowl.torque

[2] http://developereconomics.com/report/q1-2014-regional-outlook/

[3] B. Hardy and B. Phillips, Android Programming: The Big Nerd Ranch Guide (2013)