ABSTRACT
Does smartphone application development provide an opportunity to explore various aspects of embedded software? This question is the primary motivator behind the ideas explored in this paper. We cannot deny the ubiquitous nature of smartphones. Leveraging on this already available “platform” to convey embedded software concepts to Computer Science (CS) students seems an exciting opportunity. Traditionally CS have often shied away from the field of embedded systems owing to their perception of this area as “hardware” oriented, not without reason. We explore the Android platform as a means of advancing embedded software concepts to CS students.

Categories and Subject Descriptors
K.3.2 [Computer and Information Science Education]: computer science education, embedded software education.

General Terms
Computer Science Education, Curriculum.

Keywords
Embedded software, embedded systems education.

1. INTRODUCTION
The “right way” to teach embedded systems concepts, skills and techniques has been a contentious point, often with educators taking diametrically opposite viewpoints. Lee [1] contends that embedded system education should focus on the interplay of the embedded systems and the physical operating environment, appropriately labeled the cyber-physical approach to embedded system design. Most traditional embedded systems and software courses offered at various universities around the world focus on teaching the hardware and software components that comprise a working system, typically centered on a hardware “platform.” This is evident from several papers presented in the ongoing workshop series of the Workshop on Embedded Systems Education (WESE) [2]. Each approach is right in its own way, but we should not lose sight of the primary goal of embedded systems and software education, which is to introduce the students to the “unique” features of this area, contrasting it with traditional computing applications for information processing.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Conference’10, Month 1–2, 2010, City, State, Country. Copyright 2010 ACM 1-88113-000-0/00/0010...$10.00.

The perspective that is presented in this paper derives from our experience in teaching Computer Science (CS) students about this unique and exciting field. This field is often perceived by undergraduate CS students as “hardware” oriented, not without reason. This perception hinders CS students from considering this dynamic field as an option for their future career, despite the fact that embedded systems offers exciting new opportunities for them. We need to find creative ways of bringing the excitement of this field within the reach of CS students [3][4].

The rapid growth of the smartphone as one of the most powerful and widely deployed devices among users [5], especially among university students, provides us with a new opportunity to introduce embedded software to CS students. The smartphone represents a case of extreme integration, comprising a powerful computing engine, multi-protocol communication connection capabilities, several sensors for real world interaction, GUI and haptic feedback for user interaction. Given its ubiquity and ease of access, the smartphone provides an excellent springboard for learning embedded software concepts, techniques and practices.

In this paper, we elucidate our approach to exploit this powerful platform to teach embedded software concepts to CS students. We briefly describe the course including the list of topics, the hands-on laboratory exercises and some reflections on the experience with the course. We also give a brief account of the students’ perception and opinions on the course.

Earlier we taught a traditional embedded software course that we described in detail in [3][4]. As part of this course students proposed and implemented a course project on a topic of their own choice. Over the years, we noticed that applications for the smartphone platform became the main choice of the students for course projects. This prompted us to redesign the course to focus on the smartphone platform, and the result is the updated course described in this paper.

We were faced with the prospect of selecting one among the two popular smartphone platforms, iOS and Android. We chose to use Android for the following reasons: (1) Android application development is supported on multiple computing platforms, with minimal restrictions on application development and deployment, (2) Java is the primary programming language for Android application development, which most students are already familiar from their earlier programming courses, and (3) The Android platform provides a rich set of features that enables us to demonstrate several aspects of embedded software development. The salient features of smartphone application development are similar in both iOS and Android, and hence the choice of one platform does not preclude the use of the other for application development. Indeed, in our course some students chose to implement their course project using iOS instead of Android, and
did not face any significant challenge in translating their skills to the new platform, except for learning the specific features of the application development for the iOS platform.

The paper is organized as follows. Section 2 reviews some of the background for the course. Section 3 provides the details of the course. Section 4 reflects on the students’ background and comments on the course. Finally we give conclusions in Section 5.

2. BACKGROUND
In our earlier papers [3][4] we presented a detailed review of the published literature describing embedded systems and software courses offered at various universities. Thus we do not wish to reproduce this material here, except for summarizing the important issues highlighted in the literature. One of the often repeated aspects of teaching embedded systems is the multi-disciplinary nature of the field, requiring students to possess knowledge of several different areas ranging from architecture, operating systems, programming languages, and domain-specific knowledge of at least one application area. Most courses are designed around a specific platform. The platform is used as a vehicle for illustrating various embedded systems and software concepts.

Android-based courses are being offered at several universities around the world. We do not specifically review the Android platform here, as there are ample resources available online dedicated to this purpose, including the Android’s main support website http://developer.android.com, where readers can find the most comprehensive documentation of the Android platform and its features. We review here the published literature on Android-based courses. Most of these courses are specifically focused on teaching Android application programming and specific skills-set for developing Apps for Android. There is no direct interest in teaching topics related to embedded systems or software in these courses. Most of these courses introduce the features of Android through a set of laboratory exercises.

Matos et al. [6] present an overview of the Android platform and discuss their experience with teaching Android-based course. They describe a simple Maps-based Android application that was used to illustrate the various features of Android platform.

Hu et al. [7] describe a smartphone software development course for the Android platform. The primary focus of this course is also on introducing various features of the Android platform through a series of lectures and a set of hands-on laboratory exercises.

Reed and Janzen [8] describe a hands-on laboratory course based on Android. They use a progressively developed application with new features added in each exercise to demonstrate the use of various features of Android. Along the way, they illustrate various software engineering aspects and entrepreneurial thinking in the course.

McCown [9] offered an Android-based tutorial at a conference. His expertise is based on teaching and Android and iPhone-based application programming course at Harding University.

Goadrich et al. [10] compare the iOS and Android platform in great detail, reviewing several aspects including IDE support, device support, and application development. They conclude that each platform has its advantages and disadvantages, and thus the choice is dictated by individual instructor’s preferences. They conclude that immaterial of the choice of the platform, there is no denying the fact that learners will definitely benefit from the experience of developing application for the mobile platform.

Similarly Goadrich et al. [11] explored the use of Android in a CS course on data structures and algorithms. They use the Android environment to enable students to implement moderate sized projects, with focus on conveying data structures and algorithmic aspects through the use of traditional card and dice games.

No paper on smartphone application development would be complete without mentioning the most successful of smartphone application courses, CS 193P, at Stanford University [12]. This course which has gone through several iterations is wholly dedicated to cover all aspects of iPhone/iOS application development. The wide availability of the course material and course lecture videos online contributes significantly to its wide popularity among smartphone application developers worldwide.

Other authors have explored the use of the Android AppInventor (http://appinventor.googlelabs.com/about) that provides a convenient visual programming platform for developing Android applications without having to do Java programming.

Spertino et al. [13] describe the first multi-site study of the use of AppInventor in introductory Computer Science courses. This was among the first studies on the use of AppInventor.

Wolber [14] describes the use of AppInventor in a course to introduce CS to non-computing majors. He describes various features of AppInventor, and thereafter describes several student projects developed in the course using this tool.

Morelli et al. [15] explored the issues of using AppInventor to reach out to high school students through creative projects. They explored the features of AppInventor together with groups of high school teachers and undergraduate students to evaluate how the tool could be used to reach out to high school students.

Wang et al. [17] were among the few papers with the primary focus of leveraging the Android platform for use in embedded system curriculum. They developed a set of laboratory modules in support of several areas including computer vision, 3D and 2D graphics, compilers, HW/SW co-design, and real-time systems. Their usage of Android is at a fairly advanced level, with the primary focus on Android playing a supporting role in exploring several CS topics. Thus the laboratory exercises and materials are more suited for graduate level courses. Indeed, we find their approach to be a creative way of leveraging the Android platform in several different areas. In contrast to their work, our usage of Android is at a more basic level, since our course is targeted at Junior/Senior level undergraduate students.

3. The Smartphone Platform
In this section, we review the unique features of the smartphone platform and comment on how these can be leveraged to explain the embedded software development process to the students. A typical smartphone platform is characterized by the following features:

• Powerful Application CPUs
  • Currently single-core CPUs integrated in a System on a chip (SOC)
  • Future trend is towards multi-core CPUs within a SOC
• On-board Storage/Memory for applications and data, typically containing many different kinds of memory.
• User interaction provided through display, typically a touchscreen sometimes augmented with physical buttons.
• Communications/Connectivity of several kinds including Telephone, 3G, WiFi, Bluetooth
Dedicated chips for display driver, touchscreen control, GPS, bluetooth, WiFi and more.

Powerful graphics capabilities, both 2D and 3D, typically with hardware acceleration support through on-board GPU.

Sensing the real-world environment through various sensors
- Accelerometer, gyroscope, compass, proximity, light, temperature, etc.
- One or more Camera(s)

Power supply through a battery leading to power constraint issues to be addressed within the applications.

Given such a versatile, easily accessible platform, it is definitely tempting to leverage its availability to teach embedded software development.

4. Course Structure

In this section, we first give our thoughts on the topics and issues to be covered in a typical course on embedded software. We then explain the structure of our course, including the list of topics covered in the course. Thereafter, we give a brief overview of the laboratory exercises that were included in the course. We then present a selection of the projects that were developed by the students in this course.

4.1 What Should be Covered?

One of the first questions to be addressed in any course structure is the set of topics to be covered. Some of these issues were delineated in our earlier papers [3][4]. As we set out to redesign our course around the Android platform we pondered over the specific issues that we would like to bring to the attention of the students about the unique aspects and issues in software development for embedded systems. We came up with the list of issues that we wished to highlight in our course:

- Embedded software development: Specifically emphasizing on the cross-platform development that is characteristic of developing software for embedded systems. In particular with smartphone applications, the “platform” is often hidden from the developers by the software development kit (SDK). Thus it is important to remind the students about exactly how the whole process of smartphone application development and deployment takes place.
- RTOS and Timeliness Issues: Coverage including processes, threads, inter-process synchronization and communication. While it may seem extraneous to cover these basic issues, typical smartphone application implementations need to make extensive use of these features in order to provide responsiveness in applications. Background threads are often leveraged to achieve the responsiveness.
- Interaction with the real world: Embedded systems are tightly integrated with the environment/domain within which they are deployed, often receiving input from the real world and responding to these inputs. This aspect should be clearly conveyed to the CS students, to contrast it with traditional software development which is primarily focused on information processing.
- Memory Management: One of the key issues in embedded software development is to be aware of resource constraints, in particular the memory constraints imposed by the platform. When writing applications, it is imperative to keep in mind the memory usage and take steps to limit the memory consumption. In addition, it is important to pay close attention to avoid memory leaks within applications.

- Testing and Debugging: Testing and debugging on embedded systems come with their own unique challenges. Cross-platform development and deployment implies the need to carry out extensive testing both using host-based testing and target testing. Testing frameworks and features supported within the platform should be explored.

4.2 Topics Covered in the Course

The list of topics covered in the course is given in Table 1.

<table>
<thead>
<tr>
<th>Table 1: List of Course Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
</tr>
<tr>
<td>2. The Smartphone Platform</td>
</tr>
<tr>
<td>- Typical components in a smartphone</td>
</tr>
<tr>
<td>- Embedded Systems Structure</td>
</tr>
<tr>
<td>3. Embedded Software Development</td>
</tr>
<tr>
<td>- Hosts and Targets</td>
</tr>
<tr>
<td>- Mobile Application Development</td>
</tr>
<tr>
<td>- The Android SDK</td>
</tr>
<tr>
<td>4. Android Overview</td>
</tr>
<tr>
<td>- Android Architecture</td>
</tr>
<tr>
<td>- Components of an Android Application: Activities, Services, Broadcast Receivers and Content Providers</td>
</tr>
<tr>
<td>- Component Lifecycles</td>
</tr>
<tr>
<td>5. Android User Interface</td>
</tr>
<tr>
<td>- Event-driven Programming</td>
</tr>
<tr>
<td>- Handling UI Events in Android</td>
</tr>
<tr>
<td>- The Model-View-Controller Framework</td>
</tr>
<tr>
<td>6. Real-Time Operating Systems (RTOS)</td>
</tr>
<tr>
<td>- Review of Operating Systems Basics</td>
</tr>
<tr>
<td>- Processes and Threads</td>
</tr>
<tr>
<td>- Process Synchronization and Coordination</td>
</tr>
<tr>
<td>- Inter-process Communication</td>
</tr>
<tr>
<td>- Memory Management</td>
</tr>
<tr>
<td>- Processes and Thread Support in Android</td>
</tr>
<tr>
<td>- Concurrency, synchronization and Inter-process Communication Support in Android</td>
</tr>
<tr>
<td>- Memory Management Issues in Android</td>
</tr>
<tr>
<td>7. Multimedia and Graphics Support in Android</td>
</tr>
<tr>
<td>- Audio and Video Support</td>
</tr>
<tr>
<td>- 2D and 3D graphics</td>
</tr>
<tr>
<td>8. Data Storage Support in Android</td>
</tr>
<tr>
<td>9. Interaction with the Real World:</td>
</tr>
<tr>
<td>- Location Services and Maps in Android</td>
</tr>
<tr>
<td>- Sensors and Sensor support in Android</td>
</tr>
<tr>
<td>- Networking Support in Android</td>
</tr>
<tr>
<td>10. Testing and Debugging Support in Android</td>
</tr>
</tbody>
</table>
4.3 Using Android Features to Illustrate Embedded System/Software Features

After reviewing the list of topics covered in the course, we now shift our attention to explaining how we leveraged the smartphone platform to illustrate several embedded software development concepts and issues.

The “platform” of choice in our course was clear from the outset, since it was going to be based around the smartphone. Indeed, as reviewed in Section 3, the smartphone is a versatile and integrated unit that embodies several embedded systems features.

Software development for the platform requires the use of cross-platform development techniques, which are unique characteristics of embedded system development. Indeed, the discussion of the Android SDK in the course provided us with an excellent opportunity to introduce several cross-platform development concepts in context.

The detailed coverage of the Android architecture helped us to illustrate how a typical embedded systems architecture and platform is designed. The layered architecture of Android, with the lower Linux Kernel dealing with hardware interaction, and providing a standardized platform for the upper layers illustrates the abstraction issues in embedded systems. The middleware layers providing the libraries, application framework and the Dalvik virtual machine, frees application developers from the low-level concerns.

The fact that an Android application is composed of a set of interacting components (Activities, Services, Broadcast Receivers and Content Providers), with each component implementing one particular behavior makes it an interesting way to illustrate the interactions among them. Interestingly, the absence of a “main” program in an Android application is an eye-opener to CS students who are very much used to developing standard computer applications. The process of designing the application as a group of interacting components makes it a powerful means of teaching typical embedded systems behavior to the students.

User-Interfaces in Android are supported through the Activity class. The design and implementation of the user-interface requires the students to deal with user interaction and implementing the event listeners and event handlers to deal with the user’s interaction with the touch screen and hardware buttons. This provided a means for us to explain the concepts of event-driven programming and the model-view-controller (MVC) framework. The MVC framework enables us to explain to the students the principle of separation of concerns, whereby the model deals with the data, the View deals with the UI and the controller provides the interaction between the model and the view. This powerful framework enables implementation of modular applications.

The process lifecycle in Android whereby a process is classified as being of one kind: foreground, visible, service, background and empty, in that order of decreasing importance to the framework illustrates the concept of priorities in Android. This is especially important when the Android framework needs to reclaim resources in case the system is running short of resources. The framework picks the processes at the lowest priority to destroy to reclaim the resources.

Another interesting concept that can be illustrated is the need for implementing responsive applications. The Android framework will cause the Application Not Responding (ANR) dialog if the long running computation is carried out on the UI thread and blocks the UI thread for over a few seconds (currently 5 sec). This requires the application designers to make use of background threads or AsyncTask (a special construct in Android to perform long-running processing in the background) to offload computation from the UI thread. This requires the interactions with the UI to be implemented through Handlers and communication from the background thread to the UI thread. This concept is very useful in illustrating inter-thread interaction to the students.

Inter-process communication is inherent in Android framework through the use of the Intent message passing mechanism. Since Android applications are implemented as a collection of components, there is a need to provide mechanisms for these components to interact with each other. The Intent mechanism provides the glue to make this interaction possible. This feature enables us to illustrate inter-process communication in detail. In addition, the use of Pending Intents and Broadcast Receivers enables us to illustrate asynchronous and event-triggered computation to be performed by the BroadcastReceiver provides an interesting case study of interactions among components.

Coupled with the above, the use of the Service component to offload non-UI work to be done in the background and interaction with the service requiring inter-process communication mechanisms provides an excellent opportunity to illustrate several RTOS concepts.

Another important issue to be addressed is memory management and the requirement of tracking memory usage and finding and addressing memory leak problems. As Android applications are developed in Java, some of the traditional pointer related memory problems can be avoided. The use of garbage collection in Java provides an automatic means of addressing common memory leak problems. However, this does not preclude the memory leak problems completely in Android. The need to pass context between components could inadvertently introduce memory leaks. Furthermore, the need to optimize memory usage can be vociferously emphasized. This is to avoid the Android framework reaching low resource availability situation, thus prompting the framework to destroy some of the running applications.

Another important aspect of the smartphone is the availability of multiple types of sensors: GPS for location, accelerometer, gyroscope and compass for orientation and movement, proximity and light sensors. All these sensors provide a continuous feedback about the environment to the device and enable implementation of interesting applications. Android provides a versatile and uniform framework for accessing these sensors using Location Manager and Sensor Manager API that provide a friendly interface to obtain the sensor inputs. The design and implementation of applications that take in the real-time sensor data and use them to alter the application behavior provides an excellent mechanism to illustrate to the students the interaction of the embedded system with the real world and being able to modify the behavior based on input from the real world. Indeed, location based services are now the hottest trend in mobile applications.

The Android framework provides excellent support for implementing testing and debugging of applications. The use of testing based on the JUnit testing framework provides an important means of illustrating the need for designing tests during the course of application development. Furthermore, Android provides several test classes customized to the Android environment that application developers can leverage to
implement tests. We used this to illustrate testing concepts to the students. Furthermore, Android provides a number of tools to carry out extensive tests on applications, like the Monkey runner and the UI/Application Exerciser Monkey. Debugging support is integrated into the Android framework, and enables real-time debugging both using the Eclipse IDE and command-line tools.

### 4.4 Hands-On Laboratory Exercises

Any embedded software course will be incomplete without a set of hands-on laboratory exercises to complement the theory taught in the lectures. Since the course is based around the Android SDK, it was important that we include several hands-on exercises to illustrate the features of the Android framework. The hands-on exercises were designed to be a progressive introduction of various features of the Android framework through a sequence of exercises, with each exercise building upon the result of the previous exercise. The goal was to always have a working application that the students could use in practice.

The primary motivation in designing these exercises was to introduce various features of the Android framework in context by showing their usage in a real application, rather than a sterile introduction to all the features. We believe that the usage context for the features promotes better understanding of the features among students.

The exercises were organized into four simple applications: (a) An application to access to the course materials that were available on the course website, (2) an Audio player application to illustrate multimedia features, (3) a simple ball and paddle game application to illustrate graphics and sensor input features, and (4) a proximity alert application that leveraged on location information and Maps support in Android.

**Course Materials Application:** This application was designed to provide students with access to the materials on the course website like the course information, schedule and course notes. The application was progressively designed in the following steps:

1. Creating a simple User Interface
2. Using ListView in Android to implement a clickable menu and using Intents to start an Activity.
3. Downloading and parsing XML content from a Website.
4. Using AsyncTask to do processing in the background.
5. Using WebView to display HTML content.

**Music Player Application:** This application progressively implemented a simple audio player. The player was implemented in the following steps:

1. A basic music player that can play a single audio file.
3. Music player using the MediaStore content provider to enable selecting and playing from all the music files stored on the device.

**Simple 2D Game:** This application implemented a simple ball and paddle game which was then enhanced using sensor input. The application evolved to illustrate the following features:

1. A simple 2D game implemented by extending the Drawable class in Android.
2. Augmenting the game to make use of device orientation perceived through accelerometer and compass sensors to control the game.

### 4.5 Course Projects

The course required students to implement a course project to demonstrate their understanding of the various topics covered in the course. Students worked either individually, or in groups of two to propose their own project idea, and then carry it to completion. Indeed we were pleasantly surprised at the innovative ideas that the students came up with for their course projects. Most of these projects were implemented using the Android SDK and Eclipse IDE to work on Android devices. A sample of the projects implemented in the course is given below:

- A photo album display application that connects to online service like Facebook, downloads and displays photos.
- A Bluetooth based remote controller for a robot vehicle. This project required the design of a custom protocol for communication between the phone and the robot.
- A multi-function application targeting the university students that enables the students to access information from the university’s student information system, and provide PIM features (implemented using iOS platform).
- A traveller’s organizer including checklist management, and destination information management.
- An innovative wake-up alarm that requires users to play a number game to completion in order to turn off the alarm.
- An augmented reality game that made use of the camera, user’s location and graphics.
- A musician’s assistant that includes a metronome, tuner and additional music exercises.
- Personal information and finance management application.
- A public transit information and route planner application.
- A 2D cannons and shooter game implemented for an Android Tablet using AndEngine game development engine.

#### 5. STUDENTS

##### 5.1 Student Background

Our course is offered as a Junior/Senior level undergraduate elective. Hong Kong follows a three-year undergraduate curriculum (soon to switch to a four-year curriculum). The course is positioned to be taken by students who already have taken several foundation courses in Computer Science and Engineering. These include courses on object-oriented programming, data structures, architecture and operating systems. Thus the students are knowledgeable of the background topics essential to take this course.

We have been offering the course on Embedded Software for over six years now (described earlier in [3][4]). Thus the accumulated experience with offering the course helped us significantly in redesigning the course with the focus on Android.
The redesigned course with Android focus increased the course enrollment by about 30%. Typically we attracted about 20-25 students in the course. However, this year the enrollment increased to 33 students. The course attracted a balanced mixture of Computer Science and Computer Engineering students, mostly senior year students, with some junior year students. Interestingly, there was a noticeable enrollment of students from overseas institutions at our university. These students were interested in taking this course because they do not have such an opportunity in their home institutions.

5.2 Student Evaluations

The course was well-received by the students and garnered excellent reviews garnering a score of 14 points over the departmental and school average evaluations. The overall evaluation of the course was comparable to the earlier version of the course which did not cover Android in detail. A sample of (unedited) comments from the students about the course is given below:

It is good to learn android platform in this course.

The instructor provides a lot of examples to demonstrate how the technology can be done on an android device, like proximity alert, accelerometer. This is easy to catch up what is going on. For the theory part, multiple fields also be included, many new knowledge can be learnt or the old one can be refreshed. It is interesting to have an overview of how this different issue of different aspects comes together. And the most important thing is good is that the instructor often give a general view of a topic first, then focus on the android/an embedded system.

Thorough introduction to basic concepts of embedded systems

Students suggested that some overview of iOS and comparison between the two smartphone platforms would definitely bring added benefits to the students. We will examine this issue further during the next offering of the course.

6. CONCLUSIONS

We presented our views on how to leverage the Smartphone platform to teach embedded software concepts to CS students. Our experience with offering the course using this approach has been overwhelmingly positive. It is our belief (perhaps biased) that this learning opportunity should be effectively utilized to promote the study of this exciting field to CS students.

7. ACKNOWLEDGMENTS

The author wishes to express his sincere thanks to Google Inc. for their support.

8. REFERENCES