INTERFACING A SENSORY MICROCONTROLLER WITH AN ANDROID SMARTPHONE

AN INDUSTRIAL INTERNSHIP REPORT

submitted by

SHUBHAM SAINI

(10BCE1097)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE ENGINEERING

Based on the work done at the

Robert Bosch Centre for Cyber-Physic Systems,

Indian Institute of Science,

Bangalore

MAY 27 - JULY 12, 2013

DECLARATION BY THE CANDIDATE

I hereby declare that the in-plant training report entitled **"INTERFACING A SENSORY MICROCONTROLLER WITH AN ANDROID SMARTPHONE"** submitted by me to Vellore Institute of Technology, Chennai in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in **Computer Science Engineering** is a record of bonafide industrial training undertaken by me under the supervision of **Dr. G. K. Ananthasuresh** at the Robert Bosch Centre for Cyber-Physical Systems, Indian Institute of Science**.** I further declare that the work reported in this report has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Chennai: Signature

Date:

BONAFIDE CERTIFICATE

This is to certify that the in-plant training report entitled "**INTERFACING A SENSORY MICROCONTROLLER WITH AN ANDROID SMARTPHONE"** submitted by **SHUBHAM SAINI (10BCE1097)** is a record of the work done by him under my supervision at the Robert Bosch Centre for Cyber-Physical Systems, Indian Institute of Science, Bangalore. He has successfully completed the project assigned to him. His work was to link up an accelerometer-microcontroller combination to a smart phone and writing code to perform a few functions. He has demonstrated working of this and in my opinion meets the necessary standards for submission.

Dr. Jeganathan L.
Program Manager (B.Tech CSE)

Date:

Dr. G. K. Ananthasuresh
SUPERVISOR at IISc

Date:

Examiner (s) Signature
1. **2.**

Date: 12-7-2013

CERTIFICATE BY THE TRAINING OFFICER

This is to certify that the project report entitled "**INTERFACING A SENSORY MICROCONTROLLER WITH AN ANDROID SMARTPHONE"** submitted by **SHUBHAM SAINI (10BCE1097** is a record of bonafide in-plant training undertaken by him under my supervision. He has successfully completed the project assigned to him. His work was to link up an accelerometer-microcontroller combination to a smart phone and writing code to perform a few functions. He has demonstrated working of this and in my opinion meets the necessary standards for submission.

Dr. G. K. Ananthasuresh

Professor, Mechanical Engineering
Indian Institute of Science,
Bangalore.
suresh@mecheng.iisc.ernet.in
Tel: +91 (80) 2293 2334

ACKNOWLEDGEMENT

I take this opportunity to express my profound gratitude and deep regards to my guide Dr. G. K. Ananthasuresh for his exemplary guidance, monitoring and constant encouragement throughout my internship at the Robert Bosch Centre for Cyber-Physical Systems, Indian Institute of Science, Bangalore.

I also take this opportunity to express a deep sense of gratitude to Mr. Dhruv Saxena, for his cordial support, guidance as well as for providing necessary information regarding the project which helped me in completing this task through various stages.

I am obliged to staff members of the centre, for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

Place : Chennai

Date :

TABLE OF CONTENTS

ABSTRACT

It is intended that a micro-controller, an accelerometer and a Bluetooth chip embedded into a ring be interfaced with an Android based smart phone. The smart phone app currently has an image gallery and a music player in it. The ring is able to control the app wirelessly allowing the user to browse through the images, skip songs, control the music volume, control the mobile phone profile and make an SOS call in case of any emergency through simple hand gestures. All these options can also be accessed via the touch screen well should the user decide not to use the ring.

The app was developed using the Android Development Tools (ADT) version 22 built within the Eclipse IDE. The app will initiate connection to the ring via Bluetooth and start taking commands from it upon successful connection. The commands will be received in the form of strings, and if the command matches with the commands hardcoded into the app, appropriate action will be performed.

LIST OF FIGURES:

INTRODUCTION

Background

The main idea behind the project is to interface a networked ring that not only serves the purpose of aesthetics but also have certain sensing and cognitive abilities to an Android based smart phone. The ring developed at the centre communicates to a smart phone via Bluetooth and control certain functionality. The necessary operations are carried out through user's hand gestures.

Problem Statement

The aim is to develop an android app consisting of an image gallery and a simple music player. The app will initiate connection to the ring via Bluetooth and start taking commands from it upon successful connection. The commands will be received in the form of strings, and if the command matches with the commands hardcoded into the app, appropriate action will be performed.

Motivation

The motivation for our project was to create an intuitive input system that would be easy and fun to use for mobile and computer applications. The ring enables a person to use a computer or mobile phone by performing intuitive hand gestures. Our gesture input system can be conveniently used by anyone who wishes not hold the mobile phone in their hand or controlling it while it is kept in pocket or on a desk. It can also be used by someone who wishes not to be tied down to a desk when using a computer, making it perfect for giving presentations or web surfing from the couch.

Related work

A recent Master of Design (MDes) project in the Centre for Product Design and Manufacturing (CPDM) demonstrated the feasibility of a ring and a locket that have the following functionalities: (i) ability to put a mobile phone into the silent mode and turn it back to the ringing mode at the push of a button on the ring or the locket; and ability to play music on the phone by using the ring or the locket so that a misplaced phone can be found; (ii) measure the pulse rate of the person wearing the ring; and (iii) exchanging contact information by shaking hands with another person owning the same type of jewellery.

Challenges

Maintaining a stable Bluetooth connection between the ring and the mobile phone posed a serious problem. The android API provides no methods that trigger an event in case of a disruption in Bluetooth connection. The solution is to determine the time a communication between the ring and mobile phone took place and attempting to re-connect if it exceeds a certain time.

Another challenge was to display huge images in the gallery. The images to be displayed are first decoded into Bitmap format, and then displayed. So if an image is of 5MB, the converted BMP format can be well over 20MB. This leads to OutOfMemoryException. This was solved by scaling all the images to ½ the original size. Although it caused some loss in image quality, it is the only known solution.

One of the features of the app is to dial an emergency number when the ring is tapped twice. Since a single tap gesture is associated with change in mobile phone profile, in case of a double tap the mobile switched the profile and then dialed the emergency number. This was solved by waiting for a second tap for 1.5 seconds. If no second tap gesture is performed within this time, mobile phone profile was toggled.

Essence of the Project

The backbone of this entire project is the Bluetooth functionality. In the initial phase of the project, only a very basic system that initiated Bluetooth connection and sent/received data was implemented. Only when this communication system was established did we move onto more complex system.

Two separate modules: Image Gallery and Music Player were built independently. All the features needed in the final product were put into them (except the Bluetooth control). These modules were then put on top of the Bluetooth communication system and the modules were integrated.

Assumptions

- The images stored on the mobile device that are to be displayed in the gallery are of JPG,PNG or BMP format only.
- The songs stored on the mobile device to be played in the music player are of MP3 format only.
- The ring is already paired with the mobile phone.

OVERVIEW OF THE PROPOSED SYSTEM

The ring, developed at the centre communicates to a smart phone via Bluetooth. . The app will initiate connection to the ring via Bluetooth and start taking commands from it upon successful connection. The commands will be received in the form of strings, and if the command matches with the commands hardcoded into the app, appropriate action will be performed. The smart phone app consists of two modules: an image gallery and a music player. The ring is able to control the app wirelessly allowing the user to browse through the images, skip songs, control the music volume, control the mobile phone profile and make a SOS call in case of any emergency. All these options can be accessed via the touch screen as well should the user decides not to use the ring. Unlike the previous version, this ring doesn't have any button. Instead the user's hand gestures perform the necessary operations.

User Interfaces:

- Bluetooth Mode/Manual Mode
- Image Gallery
- Music Player

Hardware Interfaces:

- Rovio Networks -RN42 Bluetooth SPP chip
- Android based smart-phone

Software Interfaces:

- Windows 7
- Android Development Tools (ADT) version 22
- Android SDK (Eclipse + ADT)

Functional Requirements

- Buttons to view next/previous image
- Button to switch between image gallery/music player
- Option to choose between silent profile/vibrate profile
- Option to work without Bluetooth support
- Buttons to play next/previous song
- Volume control seek bar
- Mute button

The project activities Gantt Chart was constructed as follows:

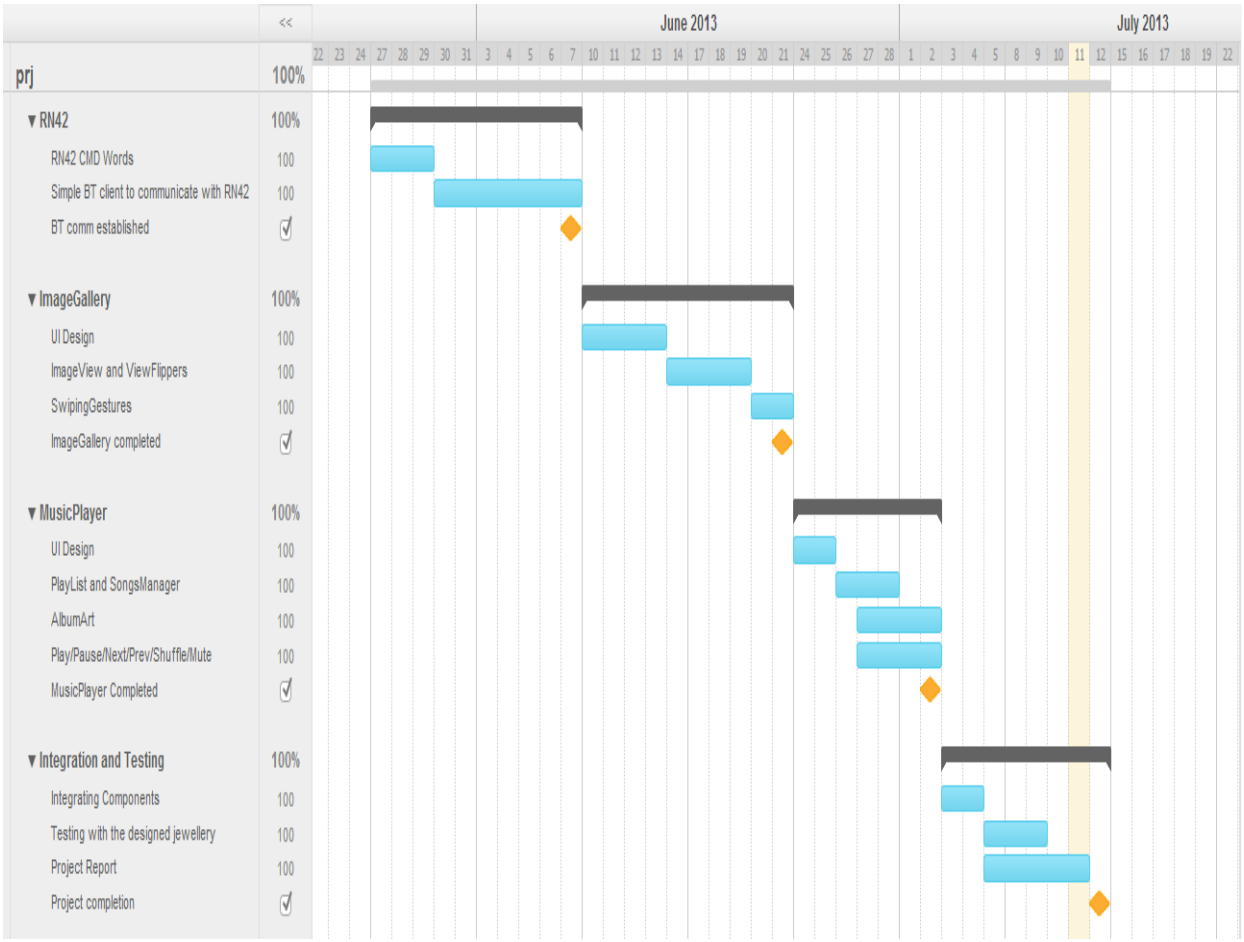

Fig 1: Gantt Chart

DESIGN OF THE SYSTEM

The system is broadly divided into 3 modules: The Image Gallery, The Music Player and a Bluetooth control module. The image gallery module and music player module are designed and implemented independent of each other, and are then put together above the Bluetooth module.

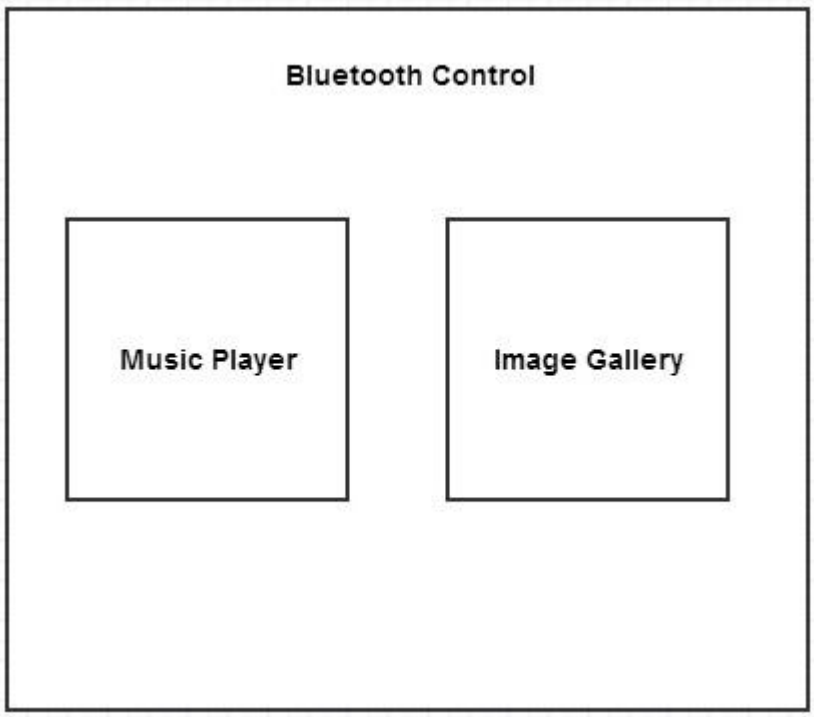

Fig 2: Overview of the system

The image gallery consists of various UI elements like buttons to browse through images, swiping gestures for easy navigation of the images and the main image container. The separate swiping gesture detection class has been implemented local to the image gallery class in order to detect finger swipes and browse the images.

The music player class consists of UI elements like play/play button, next/previous buttons, volume control seek bar, mute button and a playlist. The music player class borrows components from the SongsManager class and PlaylistActivity class. The SongsManager class is responsible for looking up songs in the phones memory and save it in an appropriate data structure. The PlaylistActivity class uses the songsList to generate a playlist. The elements of the playlist are individual songs, when clicked plays the corresponding song.

As per the functional requirements, the following use-case diagram was generated:

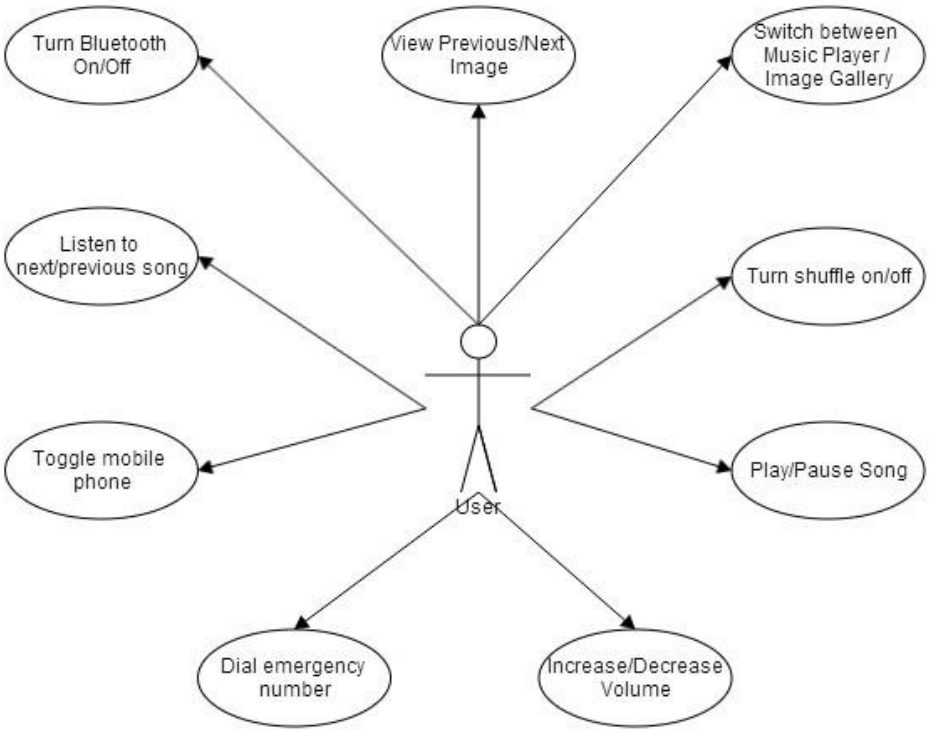

Fig 3: Use-Case Diagram

Five classes were identified for the system to be implemented:

1. BTActivity

Responsible for initiating the Bluetooth connection and communicating with the ring.

2. AndroidCustomGalleryActivity

The image gallery class. The images are displayed in ImageView within a ViewFlipper for containing multiple images.

3. MusicActivity

The music player class. It borrows components from the SongsManager class and PlaylistActivity class.

4. SongsManager

SongsManager class is responsible for looking up songs in the phones memory and save it in an appropriate data structure.

5. PlaylistActivity

The PlaylistActivity class uses the songsList to generate a playlist. The elements of the playlist are individual songs, when clicked plays the corresponding song.

The UML diagram was then constructed as follows:

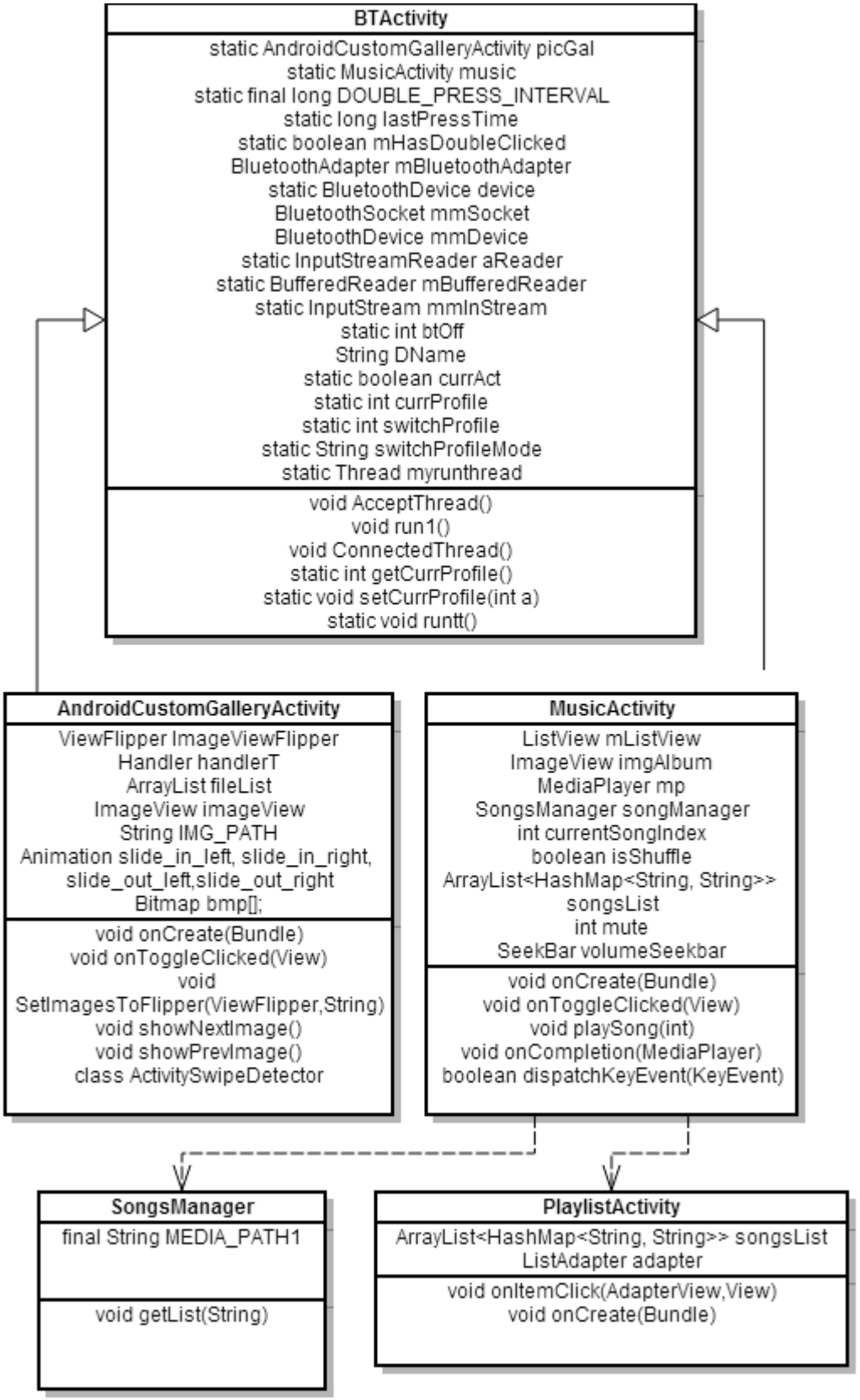

Fig 4: UML Diagram

IMPLEMENTATION OF THE SYSTEM

The system was developed using the Android Development Tools (ADT) version 22 built over the Eclipse IDE. ADT (Android Developer Tools) is a plugin for Eclipse that provides a suite of tools that are integrated with the Eclipse IDE. It offers you access to many features that help you develop Android applications quickly. ADT provides GUI access to many of the command line SDK tools as well as a UI design tool for rapid prototyping, designing, and building of your application's user interface.

Because ADT is a plug-in for Eclipse, you get the functionality of a well-established IDE, along with Android-specific features that are bundled with ADT. The following describes important features of Eclipse and ADT:

- Integrated Android project creation, building, packaging, installation, and debugging.
- SDK Tools integration
- Java programming language and XML editors
- Integrated documentation for Android framework APIs

The reasons for choosing Android development platform over other platforms like iOS or Windows:

- Low Barrier to Entry

There are no costly licensing fees or development tools. In fact, it is possible to develop applications without spending a dime.

- An Ideal Platform for Companies New To Mobile

Android applications are written in Java, with a rich set of libraries. Anyone with a working knowledge of Java can get Android applications up and running with relative ease.

- A Variety of Distribution Mechanisms

Unlike other mobile platforms, Android applications can be distributed in a variety of ways. You can use any number of third-party application stores (most notably Google’s Android Market) but you can also create your own distribution channels: for vertical market application purposes, to develop new application stores, and even put it behind corporate walls. If you build it, you can publish it.

- Open and Free Platform

License free, royalty free, and open source. That’s Android. No costly licensing fees. Also, since the underlying architecture of the Android SDK is open-source, you can

watch upcoming releases and provide feedback to the Android development team. This makes the Android platform very compelling for handset manufacturers and wireless operators.

UI Design and Layout

Original plan was to implement just the Bluetooth control system, with no UI. The system would be used for controlling entire android OS. However, due to security measures implemented within the android OS, it is impossible to perform any function outside of the application context unless your app is signed by the manufacturer.

So, it was decided to implement an Image Gallery and a Music Player to demonstrate the abilities of the ring.

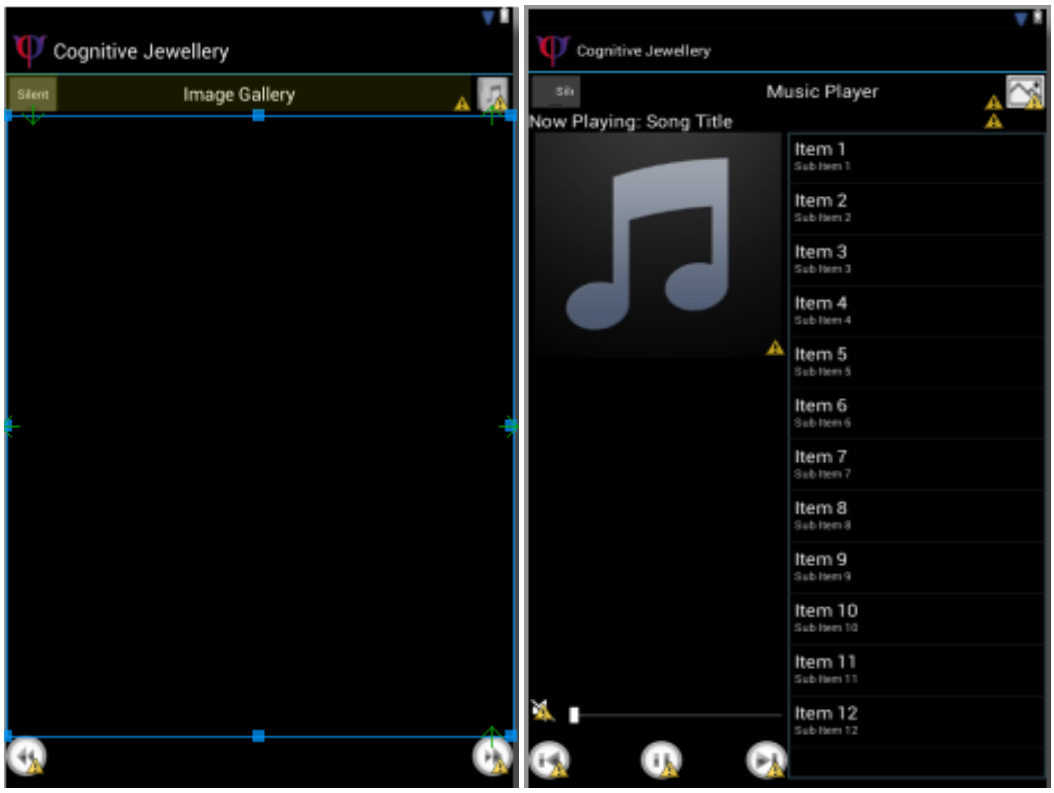

Fig 5: (a) Image Gallery Layout (b) Music Player Layout

Immediately after launching the app, the user would be presented with the choice to use Bluetooth profile or manually control the app. Should the user decide to use Bluetooth functionality, Bluetooth adapter will be turned on and connection to the ring will be initialized.

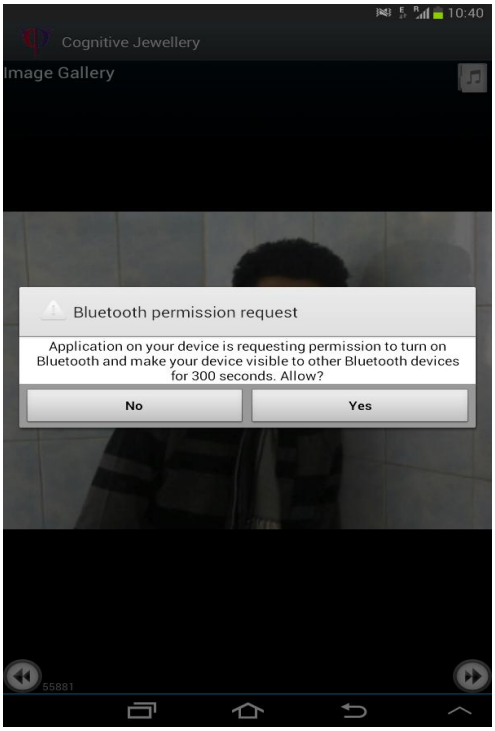

Fig 6: Asking the user to use Bluetooth profile

Next, the user will be presented with an Image Gallery. The user can browse through the images or move onto the music player. In the music player, the user has to options to skip to next song/previous song, increase/decrease volume or mute the song.

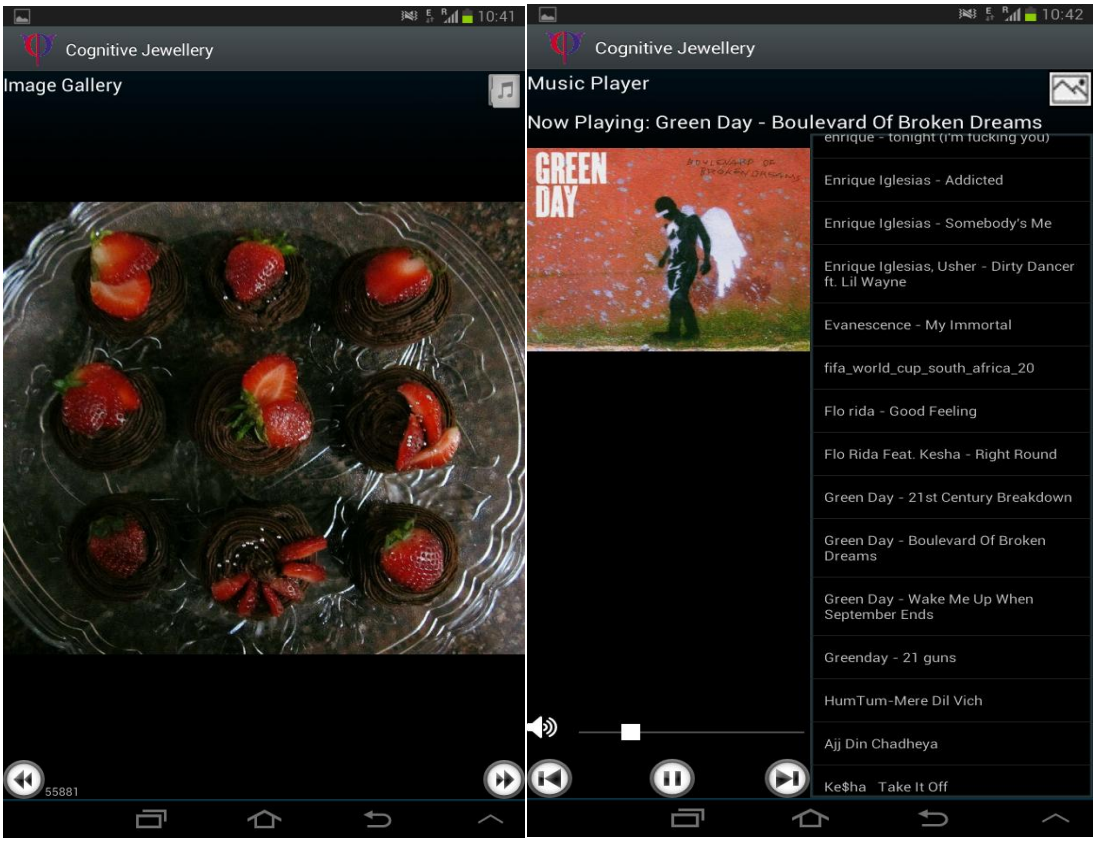

Fig 7: (a) The Image Gallery (b) The Music Player

At anytime, if the user double taps the ring once, the mobile phone profile toggles between Loud – Vibrate/Silent (depending on the toggle button provided at the top left corner).

Also, if the user wishes to dial an emergency SOS call, the user needs to double tap the ring twice.

Fig 8: Dialing emergency SOS call

The overview of the entire system can be represented by a flow chart:

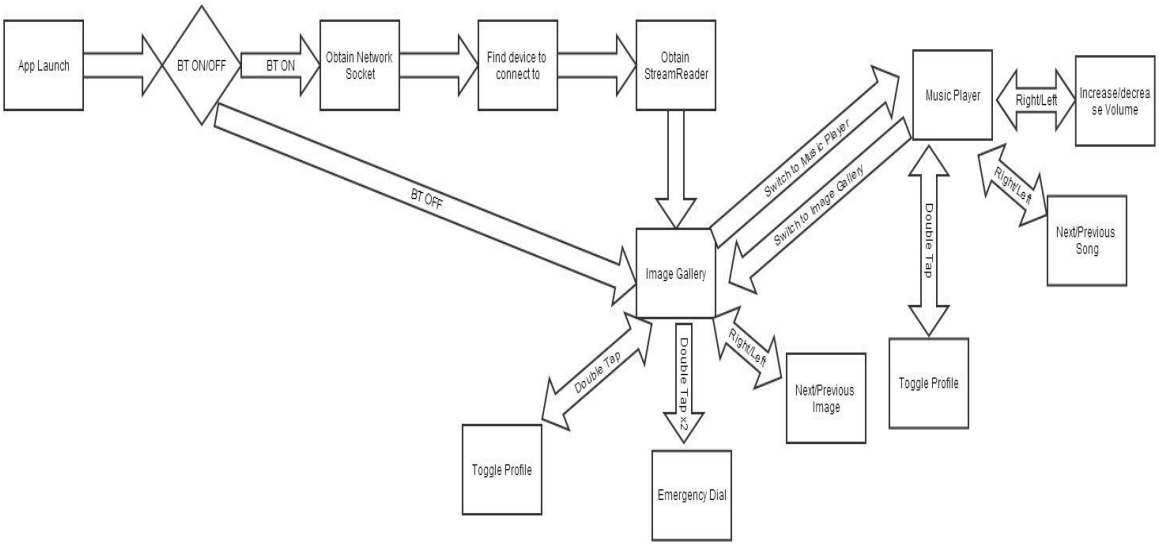

Fig 9: Flowchart of the system

Bluetooth is the building blocks of this system. Both the image gallery and music player are integrated over this Bluetooth module. A typical Bluetooth process can be described in five steps:

1. Scan for other Bluetooth devices
2. Query the local Bluetooth adapter for paired Bluetooth devices
3. Establish RFCOMM channels
4. Connect to other devices through service discovery
5. Transfer data to and from other devices

Enable Bluetooth and enabling discoverability:

```
Intent discoverableIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION, 300);
startActivity(discoverableIntent);
```

Querying paired devices:

```
Set<BluetoothDevice> pairedDevices = mBluetoothAdapter.getBondedDevices();
// If there are paired devices
if (pairedDevices.size() > 0) {
    // Loop through paired devices
    for (BluetoothDevice device : pairedDevices) {
        // Add the name and address to an array adapter to show in a ListView
```

```
        mArrayAdapter.add(device.getName() + "\n" + device.getAddress());
    }
}
```

Initiating a connection:

```
public ConnectThread(BluetoothDevice device) {
        // Use a temporary object that is later assigned to mmSocket,
        // because mmSocket is final
        BluetoothSocket tmp = null;
        mmDevice = device;

        // Get a BluetoothSocket to connect with the given BluetoothDevice
        try {
            // MY_UUID is the app's UUID string, also used by the server code
            tmp = device.createRfcommSocketToServiceRecord(MY_UUID);
        } catch (IOException e) { }
        mmSocket = tmp;
    }

    public void run() {
        // Cancel discovery because it will slow down the connection
        mBluetoothAdapter.cancelDiscovery();

        try {
            // Connect the device through the socket. This will block
            // until it succeeds or throws an exception
            mmSocket.connect();
        } catch (IOException connectException) {
            // Unable to connect; close the socket and get out
            try {
                mmSocket.close();
            } catch (IOException closeException) { }
            return;
        }

        // Do work to manage the connection (in a separate thread)
        manageConnectedSocket(mmSocket);
    }

    /** Will cancel an in-progress connection, and close the socket */
    public void cancel() {
        try {
            mmSocket.close();
        } catch (IOException e) { }
    }
```

TESTING:

For its android platform, Google provides a number of useful tools like the Android Virtual Device, Dalvik Debug Monitor service and LogCat.

Dalvik Debug Monitor service served as the primary testing tool because it includes LogCat, a logging utility, shows information on currently running threads, and emulates certain Android features used extensively in the project, such as Bluetooth

data transfer. The testing strategy used throughout the course of the project was basic yet informative and effective; the majority of testing, debugging and data checking was done through LogCat, the logging utility. The utility divides all log messages by tags: Verbose, debug, Info, Warning and Error; additionally, there's a filtering mechanism that can divide messages by class names in which they were generated. In android, logging is used to print stack traces, notify the developer of the system status changes, and print user-defined information.

This methodology was chosen in favor of unit testing for several reasons:

- Logging is somewhat more versatile since its possible to check loop control, Activity and Service status, user preferences etc.
- There's an additional time overhead involved in writing testing classes, which is not necessary as the output values can be checked by logging.
- Test-driven development wasn't adopted from the start due to the novelty of the platform.

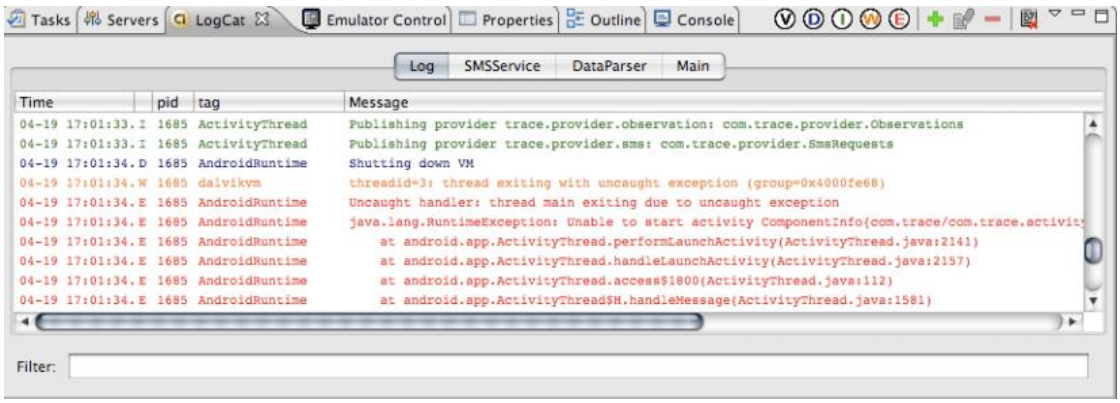

Fig 10: LogCat view

CONCLUSIONS AND FUTURE WORK

The project has met all of its main requirements such as taking commands through hand gestures over Bluetooth, implementing a basic image gallery and music player. The purpose of the project was to demonstrate the cognitive and sensing abilities of the ring, and not just developing an android app. In that respect, the project was a success; the Android platform was practiced at an advanced level.

The choice of Android platform was a strongly positive one; no other platform has so far achieved the same flexibility, openness, and number of features. Apples iOS, as

stated before, provides a similar set of features, but it is a proprietary platform that cannot be used to build a budget devices aimed at the developing markets.

In terms of adaptability, it should be noted that the development followed standard Android practices in design and configuration, and will not, therefore prevent anyone familiar with the platform from extending it. The application used standard Android paradigms such as XML-based layout, and values resources, which are used to store string and array constants in XML files that are separate from the main java code. Every effort has been made to keep the programs functions concise, versatile and re-usable. The code has been properly formatted and documented to improve readability.

The following technologies are learned in depth:

- Eclipse IDE
- Bluetooth
- Java OOP
- XML-based layout
- Android Views
- Android OS events
- Swiping Gestures
- Project Management

Future Enhancements

In future, the system can be extended to accommodate more number of gestures. The android app can be evolved into a more complex, sophisticated image gallery and music player.

Also, various android device manufacturers can be contacted to seek permission to write apps with System Permissions. This will allow OS-wide control through the ring.

APPENDIX 1 - GLOSSARY

.apk file - Android application package file. Each Android application is compiled and packaged in a single file that includes all of the application's code (.dex files), resources, assets, and manifest file.

.dex file - Compiled Android application code file.

Activity - A single screen in an application, with supporting Java code, derived from the Activity class.

adb - Android Debug Bridge, a command-line debugging application included with the SDK. It provides tools to browse the device, copy tools on the device, and forward ports for debugging.

DDMS - Dalvik Debug Monitor Service, a GUI debugging application included with the SDK. It provides screen capture, log dump, and process examination capabilities.

Intent - A message object that you can use to launch or communicate with other applications/activities asynchronously. An Intent object is an instance of Intent.

Layout - An XML file that describes the layout of an Activity screen.

Manifest File - An XML file that each application must define, to describe the application's package name, version, components (activities, intent filters, services), imported libraries, and describes the various activities, and so on.

Resources – Non-programmatic application components that are external to the compiled application code, but which can be loaded from application code using a well-known reference format.

View - An object that draws to a rectangular area on the screen and handles click, keystroke, and other interaction events. A View is a base class for most layout components of an Activity or Dialog screen (text boxes, windows, and so on).

Widget - One of a set of fully implemented View subclasses that render form elements and other UI components, such as a text box or popup menu.