Air Guide

an Android Approach to Flight Information

A graduate project submitted in partial fulfillment of the requirements

for the degree of Master of Science in Electrical Engineering

By

Rolando Douglas

in collaboration with

Avetis Petrosyan

December 2015
The graduate project of Rolando Douglas is approved:

______________________________  ________________________________
Dr. Robert D. McIlhenny  Date

______________________________  ________________________________
Dr. Sharlene Katz  Date

______________________________  ________________________________
Dr. Ronald W. Mehler, Chair  Date

California State University, Northridge
DEDICATION

I dedicate this graduate project to my loving wife Kim Douglas. Thank you for being so patient and understanding during this time. I appreciate all the support and encouraging words you have given me. I could not have accomplished this without you.

I would also like to thank my parents, Rogelio and Anelaisa Douglas. Thank you for pushing me to be the very best. My brothers Alex and Marlon, I have always looked up to you guys and am grateful for everything you have taught me.
# TABLE OF CONTENTS

Signature ........................................................................................................................................... ii  
Dedication ....................................................................................................................................... iii  
List of Figures ...................................................................................................................................... vi  
List of Tables ....................................................................................................................................... viii  
Abstract ........................................................................................................................................... ix  
Chapter 1: Introduction ................................................................................................................... 1  
Chapter 2: History of Radio Navigation ......................................................................................... 3  
Chapter 3: RTL-SDR ....................................................................................................................... 8  
  Description and Approach ............................................................................................................. 8  
  USB Drivers ................................................................................................................................. 10  
Chapter 4: Digital Signal Processing ............................................................................................ 12  
  Decoding Radio Beacon Signals with DSP techniques ......................................................... 12  
  IQ Data ...................................................................................................................................... 18  
  Finite Impulse response ............................................................................................................ 23  
Chapter 5: OpenSL ES .................................................................................................................... 35  
Chapter 6: Android Specific .......................................................................................................... 39  
  Android Make File .................................................................................................................... 39  
  Java Native Interface (JNI) ...................................................................................................... 41  
  Android Manifest ...................................................................................................................... 43  
  AirGuideActivity Layout ......................................................................................................... 46  
Chapter 7: Map Software ............................................................................................................. 49  
  Map Software Types ................................................................................................................ 49  
  Map Tile Creation .................................................................................................................... 51  
  Radio Beacon Symbols and Locations .................................................................................... 53  
Chapter 8: Software Architecture .................................................................................................. 57  
Chapter 9: Test ............................................................................................................................... 63  
  Physical Car Setup .................................................................................................................... 63  
  Locations and Results .............................................................................................................. 66  
  User Interface / Map ................................................................................................................ 69  
  RTL-SDR Testing ...................................................................................................................... 72  
Chapter 10: Possible Improvements .............................................................................................. 73  
  Overlay bearing on map ............................................................................................................. 73
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Better RTL-SDR Hardware</td>
<td>74</td>
</tr>
<tr>
<td>Connect to two NavAids</td>
<td>75</td>
</tr>
<tr>
<td>Automatic adjustment to Magnetic Declination</td>
<td>76</td>
</tr>
<tr>
<td>References</td>
<td>77</td>
</tr>
<tr>
<td>Appendix A: Setup Development Environment</td>
<td>80</td>
</tr>
<tr>
<td>Appendix B: Android Code</td>
<td>106</td>
</tr>
<tr>
<td>Appendix C: Native C Code</td>
<td>174</td>
</tr>
<tr>
<td>Appendix D: NavAid database</td>
<td>240</td>
</tr>
<tr>
<td>Appendix E: Fileparser Input</td>
<td>244</td>
</tr>
<tr>
<td>Appendix F: Acronyms</td>
<td>259</td>
</tr>
<tr>
<td>Appendix G: Addendum</td>
<td>260</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Illustration of triangulation using two radio beacons</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Illustration of radio beacon Signals</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>Connection Bus Interface</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>Software and Hardware Architecture</td>
<td>11</td>
</tr>
<tr>
<td>5</td>
<td>Radio Beacon Signal Decoding Diagram</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>Filtering Instantaneous Bearing</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>Addition of Quadrature Signals</td>
<td>19</td>
</tr>
<tr>
<td>8</td>
<td>New Signal Created Through Summation of Quadrature Signals</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>Modulate an RF Signal</td>
<td>21</td>
</tr>
<tr>
<td>10</td>
<td>Demodulate an RF Signal</td>
<td>21</td>
</tr>
<tr>
<td>11</td>
<td>FIR Chebyshev 30Hz 190th Order Low-pass Filter</td>
<td>24</td>
</tr>
<tr>
<td>12</td>
<td>Magnitude Response of 30Hz Low-pass FIR Filter</td>
<td>25</td>
</tr>
<tr>
<td>13</td>
<td>FIR Chebyshev 9960Hz ± 480Hz Band-pass Filter</td>
<td>26</td>
</tr>
<tr>
<td>14</td>
<td>FIR Chebyshev 9960Hz ± 480Hz Band-pass Filter Magnitude Response</td>
<td>27</td>
</tr>
<tr>
<td>15</td>
<td>FIR Chebyshev Low-pass 30Hz Filter 95th Order</td>
<td>28</td>
</tr>
<tr>
<td>16</td>
<td>FIR Chebyshev Low-pass Filter Magnitude Response</td>
<td>29</td>
</tr>
<tr>
<td>17</td>
<td>First Attempt of Decoding Both the Reference and Variable Signals</td>
<td>31</td>
</tr>
<tr>
<td>18</td>
<td>Final Version of Decoding Both the Reference and Variable Signals</td>
<td>32</td>
</tr>
<tr>
<td>19</td>
<td>Close Up of First Attempt of Decoding the Signals</td>
<td>33</td>
</tr>
<tr>
<td>20</td>
<td>Close Up of Final Version of Decoding the Signals</td>
<td>34</td>
</tr>
<tr>
<td>21</td>
<td>Startup Sequence (Main Java Thread Executes this Call)</td>
<td>37</td>
</tr>
<tr>
<td>22</td>
<td>Threaded Function to Playback Morse Audio</td>
<td>37</td>
</tr>
<tr>
<td>23</td>
<td>Morse Code Buffer Written to be Enqueued</td>
<td>38</td>
</tr>
<tr>
<td>24</td>
<td>Example of Native C Code Compiled to a Library</td>
<td>40</td>
</tr>
<tr>
<td>25</td>
<td>Android Manifest - Permissions</td>
<td>43</td>
</tr>
<tr>
<td>26</td>
<td>Air Guide Layout with Visible Options</td>
<td>47</td>
</tr>
<tr>
<td>27</td>
<td>Air Guide Layout with Options Hidden</td>
<td>48</td>
</tr>
<tr>
<td>28</td>
<td>IFR Chart from skyvector.com</td>
<td>50</td>
</tr>
<tr>
<td>29</td>
<td>Mobile Atlas Creator v1.9.16</td>
<td>52</td>
</tr>
<tr>
<td>30</td>
<td>VOR Symbol</td>
<td>53</td>
</tr>
<tr>
<td>31</td>
<td>VOR-DME Symbol</td>
<td>53</td>
</tr>
<tr>
<td>32</td>
<td>VORTAC Symbol</td>
<td>53</td>
</tr>
<tr>
<td>33</td>
<td>TACAN Symbol</td>
<td>53</td>
</tr>
<tr>
<td>34</td>
<td>NDB Symbol</td>
<td>53</td>
</tr>
<tr>
<td>35</td>
<td>DB Browser for SQLite</td>
<td>54</td>
</tr>
<tr>
<td>36</td>
<td>DB Browser for SQLite - Import CSV File</td>
<td>55</td>
</tr>
<tr>
<td>37</td>
<td>Top level software architecture</td>
<td>58</td>
</tr>
<tr>
<td>38</td>
<td>AirGuideActivity software architecture</td>
<td>59</td>
</tr>
<tr>
<td>39</td>
<td>Radio Antenna</td>
<td>64</td>
</tr>
<tr>
<td>40</td>
<td>Physical Car Setup</td>
<td>64</td>
</tr>
<tr>
<td>41</td>
<td>Improvement - Overlay Bearing</td>
<td>73</td>
</tr>
<tr>
<td>42</td>
<td>Improvement - Overlay Two Bearings</td>
<td>75</td>
</tr>
<tr>
<td>43</td>
<td>NDK Downloads OS Choice Page</td>
<td>81</td>
</tr>
</tbody>
</table>
Figure 44 NDK Downloads Accept Page ................................................................. 81
Figure 45 NDK Extraction Command ................................................................. 82
Figure 46 NDK Extraction Verification .............................................................. 82
Figure 47 Android SDK Install ........................................................................ 85
Figure 48 Eclipse Launch Workspace Location Verification ........................... 86
Figure 49 Eclipse Right-Click to Import Project ............................................. 87
Figure 50 Eclipse Import Existing Android Project ...................................... 88
Figure 51 Eclipse Verify Android Project to Import ........................................ 89
Figure 52 SDK Version to Use ........................................................................... 90
Figure 53 Create a Project Directory in Home Folder ...................................... 91
Figure 54 Install Script Complete ................................................................. 92
Figure 55 104.3MHz FM Radio Playing Verification ...................................... 93
Figure 56 Java Link to Install JRE ................................................................. 94
Figure 57 Installed JRE Verification ............................................................. 95
Figure 58 Java Link to Install JDK ................................................................. 97
Figure 59 Eclipse to Import Android Project ............................................... 100
Figure 60 Verify Android Project to Import ............................................... 101
Figure 61 ADB Lists Android Device as Hexadecimal Values ..................... 103
Figure 62 ADB Connection Successful ....................................................... 104
Figure 63 LogCat symbol ............................................................................. 104
Figure 64 LogCat Log Through ADB ......................................................... 105
# LIST OF TABLES

Table 1 Format of NavAids_db ................................................................. 56  
Table 2 Layout of NavAid database input ............................................... 60  
Table 3 Sample of bearing calculation.................................................... 70  
Table 4 Sample of FileParser Input ....................................................... 71
ABSTRACT

Air Guide

an Android Approach to Flight Information

By

Rolando Douglas

Master of Science in Electrical Engineering

Radio navigation is one form of aeronautical navigation used by tuning into frequencies to determine a position. Pilots need to maintain a list of frequencies for all radio beacons and manually tune to them to receive a bearing. Once received, they plot this bearing on a map, usually physical. This can be quite inconvenient and is error prone while in flight.

This formal literature covers our proposal to solve this problem. Air Guide will automate the majority of these tasks and consolidate the tools required to accomplish this into a single Android based mobile device. Using this device, one can connect an over the counter RTL-SDR receiver and tune to any desired radio beacon. The application will
pick up the radio beacon’s signal and calculate a bearing. The bearing is displayed on a map based User Interface.

My contribution to this joint project includes designing the software architecture, creating the UI (User Interface) including all user interactions, integrating offline-mapping software, the creation of map tiles for offline use, and the overlay of NavAid data.

The result from our integrated prototype is successful.
CHAPTER 1: INTRODUCTION

This project consists of creating an Android application that decodes radio beacon signals to calculate a bearing and present it on a map with an RTL-SDR dongle. It contains an easy to use UI (user interface) that allows pilots to select from any radio beacon station within range with the click of a button to receive a calculated bearing. A user defined Magnetic Declination is required to compensate for Magnetic North to display a bearing relative to True North. Morse code is replayed once every ten seconds for verification purposes. A GPS calculated bearing is also present if GPS is available. DSP techniques have been incorporated in the software design. From the user’s perspective, it’s a simple and cost effective means of adding NavAid decoding technology to any aircraft. The user interface along with the offline map tiles makes it a very convenient and easy to use tool with minimal interaction from the pilot. This method is preferable so they concentrate on flying. The interface is also highly customizable and no internet connection is required during operation.

Rolando Douglas was responsible for creating the map/user interface. Avetis Petrosyan was responsible for decoding the analog signals of the radio beacon using DSP techniques. Our responsibilities overlapped during integration and test when refining the algorithm to convert the radio signals into a bearing in addition to user interface updates.

The results from this project were successful. Air Guide can decode radio signals to get an accurate bearing. The bearing has been verified against IFR charts as well as to the GPS computed bearing. Our project was tested in flight, however the results had a higher
standard deviation than the one experienced on the ground. See Physical Car Setup for more detail.
CHAPTER 2: HISTORY OF RADIO NAVIGATION

Radio navigation was invented and heavily exercised during World War I. This form of geolocation was known as Radio Direction Finding, RDF. Every radio beacon has a designated frequency and a known geolocation. These frequencies do repeat because there are more beacons than there are available frequencies but they are spread out enough so that the same frequencies do not overlap. Many low-range stations were created solely for this purpose. This could also be accomplished with AM radio stations that are higher powered. When a pilot tunes to one of these frequencies, a directional antenna would be used to determine the direction of the broadcasting signal. If a second measurement from a different radio beacon could be determined, triangulation would be used to geolocate a coordinate. Although using this method would be a lot less accurate than current GPS technologies, this form of radio navigation is still in use today. Figure 1 is an illustration of two radio beacons transmitting the bearing of the aircraft relative to magnetic north.
There are five types of radio beacons used for aeronautical navigation. They are NDB, VOR, VOR-DME, TACAN, and VORTAC.

NDB, Non-Directional Beacon, is an evolved form of the original RDF beacons used during World War I. These NDBs have a much greater range because the signals follow the curvature of the earth. The signal’s intensity is measured to determine distance.

VOR, VHF (Very High Frequency) Omni Directional Radio Range, is a radio beacon that transmits four encoded signals. This technology was deployed in the USA in 1946 and is in use worldwide. The first signal is an audio signal that plays the beacon’s unique station ID in Morse code. This serves two purposes. To make sure the right station is tuned in and to verify that the signal is clean, the less static from this audible wave, the stronger (more amplitude, smoother sine waves, more defined) the signal. The second signal that is encoded within the tuned frequency is the voice signal. The purpose of the voice signal
is to transmit in-flight advisories or live flight broadcasts. The last two signals, variable and reference, are sinusoidal and are differentiated from one another to determine the phase angle. This offset angle is the bearing of the aircraft relative to magnetic north. Figure 2 is an illustration of how the two angles would appear at different positions relative to the beacon. The black sine wave is the reference wave and always remains the same. It is an omnidirectional signal that operates at 30Hz which is encoded via a band-pass filter that operates at 9960Hz ± 480Hz. The red sine wave is the variable wave and is offset to the reference wave depending on where you are located when receiving the signals. The reference signal is a directional signal that operates at 30Hz. The radio beacon will be explained in detail in chapter Decoding Radio Beacon Signals with DSP techniques. For example, directly south of the radio beacon, the two signals are inline showing a 0° offset which is a bearing of 0°. Directly to the west of the radio beacon, the reference signal is offset by 90°, which is a bearing of 90°. The same logic applies to north and east.
VOR-DME is a VOR receiver with an additional transmitter, DME, Distance Measuring Equipment. The aircraft transmits this signal to the DME station and a signal is echoed back to the aircraft. The time it takes for the signal to return is used to calculate distance. With a clean enough signal, one can determine their location with a lock on only one VOR-DME using a combination of the VOR’s reported bearing and the DME’s distance.

TACAN is an NDB that transmits azimuth and distance. These are primarily used by the military and government aircrafts, like NASA space shuttles.

VORTAC is a highly precise VOR receiver combined with a TACAN used primarily by the military.

Until recent years, this technology was all done using analog hardware and a physical map (i.e. IFR Chart, VFR Chart, general map). Air Guide is an application intended to
make this experience more user friendly. Our project will use a Software Defined Radio to receive the radio beacon’s analog signals and convert them into digital values that can be used to calculate a bearing. This is all accomplished on an Android based device. Combine this information with the latest mapping software including all USA based radio beacons that the user can easily tap to connect and you have a device that gives small aircraft pilots a simplified navigation tool.
CHAPTER 3: RTL-SDR

Description and Approach

An open source community modified RTL-SDR, originally manufactured as a PAL TV Tuner, after they realized the potential of its RTL2832U chipset to create alternate device drivers to be used as a radio frequency receiver on Linux OS. Some of its new capabilities include Frequency Modulation (FM), Amplitude Modulation (AM), Upper/Lower Sideband (USB/LSB), Continuous Wave (CW), Ham radio, etc. The drivers reconfigured the PAL TV Tuner to a Software Defined Radio for many different purposes. Processing FM and AM signals are the intent of this project.

Various projects have been created using the open source community RTL-SDR drivers, one in particular that was useful to this project. Kyle Keen created an rtl_fm package, designed to demodulate various radio frequency signals. For example, his software can demodulate any FM station and prepare an audio buffer for playback. The audio buffer is played using the SOX driver; an independent audio driver on Linux OS separate from that of this RTL-SDR. This project builds on top to achieve radio beacon signal decoding abilities, which is described in Decoding Radio Beacon Signals with DSP techniques.

Ezcap EZTV668 DVB-T Digital TV USB 2.0 Dongle is the RTL-SDR USB dongle used for the purposes of this project. The device is equipped with a USB 2.0 and a female PAL connection. A miniature-sized antenna is included in the package as well; however, it was insufficient to receive a clean signal. It was replaced with an alternate twenty-four inch tall fiberglass mast magnetic scanner antenna with BNC connector. A PAL to BNC adapter was necessary for the connection for the antenna to the RTL-SDR dongle. To
connect the RTL-SDR to the Android device, a male micro USB to female USB 2.0 adapter was required. Figure 3 depicts the connection bus interface with all the hardware.

![Figure 3 Connection Bus Interface](image-url)
**USB Drivers**

The Android device and RTL-SDR communicate via USB, hence the need for USB drivers. A bidirectional connection, the Android device commands the RTL-SDR to tune and listen to a certain radio beacon frequency, which then relays all the incoming data to the Android device for processing.

Enabling the USB connection starts from the AndroidManifest.xml file. A `<uses-features>` element is required to notify the user to enable USB communication. Next, an `<intent-filter>` is used to allow the RTL-SDR dongle to notify the user there is an intent to use the USB connection. It acts as a high priority hardware interrupt, providing the user the ability to allow or deny the connection. Lastly, the `<meta-data>` attribute, allows the Android OS to get a hash map of all available connected components. One USB connection (RTL-SDR dongle) will be visible as it is the only external connected device.

The permissions above are required to open the USB connection.

UsbManager class is an Android SDK class, which provides access to any USB peripherals. This class returns a list of all USB connected devices in the form of a hash map. Hash maps are data structures, similar to an array, they hold numerous elements, however they are accessed by converted key strings instead of standard integer index values. The information is programmatically displayed in LogCat for the developer or user to identify a successful connection when the RTL-SDR is connected via USB.

The native C side of the USB portion, developed by an open source community, is used to initialize the RTL-SDR via USB to communicate with the RTL-SDR. Android OS natively runs on Java, however the open source community wrote all the RTL-SDR
drivers in C. The Android NDK (Native Development Kit) allows native C code to run in conjunction with Java on an Android OS via JNI (Java Native Interface). The main application starts with Java, which starts a thread in native C via JNI. Next, the native C thread calls a function back in Java via JNI to open the USB port. The Android device may now communicate with the RTL-SDR via USB allowing the user to command the RTL-SDR to tune to various radio beacon(s) via the UI (user interface). As its name implies, the user interface shown in Figure 25, is the only way the user can command the application. The user through the UI, written in Java, commands the RTL-SDR to tune to a certain radio beacon with the click of a connect button. Figure 4 depicts the architecture described.

Figure 4 Software and Hardware Architecture
CHAPTER 4: DIGITAL SIGNAL PROCESSING

Decoding Radio Beacon Signals with DSP techniques

VOR (Very High Frequency Omnidirectional Range) is a type of radio beacon used for aircraft travel. It emits four encoded signals; Morse code Identifier, voice, reference and variable signal. The Morse code Identifier emits the airport name in Morse code, for example, the Van Nuys airport transmits VNY in Morse code. This is used to verify it is connected to the specific VOR and is receiving valid data. The voice portion of the signal is to warn users if there are severe weather conditions that can potentially affect landing. The reference signal is an omnidirectional signal with a frequency of 30Hz. To decode the reference signal data, the user needs to filter the data with a 9960Hz ± 480Hz band-pass filter, frequency demodulate the result and filter once more with a 30Hz low-pass filter. The result is a sinusoidal wave, a clear line of sight increases smoothness and clarity of the sine wave. The variable signal is very similar to the reference signal in terms that it also is transmitted at 30Hz, decoded properly it becomes a sinusoidal wave however the variable signal is a directional signal unlike the omnidirectional reference signal, meaning the pilots location in reference to the radio beacon matters. An aircraft to the east of the radio beacon will receive the variable signal at a different time (180° phase shift between them) than an aircraft to the west of the radio beacon. To decode the variable signal, filter the data with a 30Hz low-pass filter. The reference and variable signals both contain a frequency of 30Hz, which is the reason the reference signal must be encoded via more filters. Fitting two signals on the same frequency would leave the receiving instrument with undeterminable data, hence the need to encode one of the two signals in additional filtering.
The valid frequency ranges for radio beacons are from 108MHz to 117.95MHz with 50KHz offsets. These frequencies are exclusively reserved for radio beacons. The 50KHz offsets ensure the frequencies between the radio beacons do not interfere with one another, even if the receiving device has low resolution during tuning. Radio beacons with the same frequency exist, as there are not enough frequencies to cover them all. However, beacons with the same frequencies are far enough apart to avoid any possibility of overlap.

To calculate bearing, one must find the phase difference between the reference and variable signals. This becomes a tedious process. Using DSP (Digital Signal Processing) techniques in software, an algorithm was created to calculate bearing, however, issues arose. Throughout testing trials, the algorithm was perfected to provide accurate results. Figure 5 represents the flow chart diagram on decoding a radio beacon signal to calculate bearing.

![Figure 5 Radio Beacon Signal Decoding Diagram](image)
An intermediate goal was to use Frequency Demodulation to decode an FM radio station, 104.3MHz as an example. A successful attempt ensures the correct use of the RTL-SDR.

Two choices exist to play the audio, using the built-in Android SDK Java class or with OpenSL ES (written in C). The first approach allows an easy to use interface with minimal coding effort, however, it contains performance issues since the source audio buffer is passed through JNI. The second approach is a more difficult programmatic approach although it contains performance benefits since there is no use of JNI as they are both written in C. More detail about audio drivers can be found in the drivers. As a general rule of thumb, minimizing JNI (Java Native Interface) use, which is C to Java communication and vice versa, is preferable. The use of JNI entails processor intensive tasks and complex logic. Hence, the OpenSL ES drivers were written.

Through the correct initialization and tuning of the RTL-SDR, the audio buffer was filtered with the use of IQ data and passed as a buffer to the OpenSL ES drivers to play the audio through the speakers. The intermediate goal was met ensuring successful control of the RTL-SDR dongle.

Decoding the radio beacon signal requires the use of DSP (Digital Signaling Processing), a complex topic in itself. Internet research revealed a thesis written by Thomas N Tye on how to decode a radio beacon signal using DSP techniques. A few modifications were required to the block diagram representation to decode it.

Through trial and error, it was determined to use a FIR (Finite Impulse Response) filter of 190th order to produce smooth sinusoidal waves for both reference and variable signals. Each signal, reference and variable, contain multiple data points that are stored in
independent arrays. The arrays are written to files and loaded in MATLAB to be plotted for visual perspective. This visual interpretation of both sinusoidal waves creates a pictorial representation of the phase difference (bearing) as can be seen in Figure 20.

An algorithm was designed to determine the average phase difference (bearing) between the two signals (non-ideal sine waves). To search for the top peaks of both sine waves, the array data is checked for an overall increase in numeric value, store the highest value before it is an overall descent. It works for non-ideal conditions as well assuming the values are ascending followed by a brief descent and the ascent continues, hence an overall increase rather than a strict ideal increase in numeric value. The pattern is continually repeated to store all top peaks of the sine waves. The algorithm is used on both reference and variable sine waves. Once the top peaks are discovered for both sine waves, the algorithm checks for the phase difference between them. Averages of five to six top peaks are stored for every iteration of live data. Instantaneous bearing (phase difference) is computed for each with the following subtraction formula:

\[
\text{phase difference} = \text{variable top peak index} - \text{reference top peak index}
\]

Each phase difference is passed to a moving average sum function, which filters the last thirty phase differences to produce a consistent average bearing. Special conditions are handled as well such as a bearing of 0° followed by a bearing of 359°. From a mathematical perspective, the average between 0° and 359° is roughly 180°. Special measurements were taken to provide an accurate bearing for rollover conditions of 359° to the pilot. If a similar pattern is detected, 360° is added to the lower of the two bearings.
being compared and the result is modulated by 360°. This pattern is represented in Figure 6.

![Diagram of Filtering Instantaneous Bearing]

**Figure 6 Filtering Instantaneous Bearing**

Following the logic from filtering instantaneous bearing, there is potential to contain bearings greater than 360°. The problem is solved when the moving average sum filter is applied. After the collection of thirty consecutive bearings, an average bearing is calculated with the following formula:

\[
Average\ Bearing = \left( \frac{\text{sum of the last 30 bearings}}{30} \right) \mod 360
\]

The current average bearing is compared to the previous average bearing for any special rollover conditions as mentioned earlier in Figure 6, if any exist, 360° is added to the
smaller of the two average bearings. A similar formula is used to calculate the final bearing, which is from a collection of the last 30 consecutive average bearings.

\[ Bearing = \left( \frac{\text{sum of the last 30 average bearings}}{30} \right) \mod 360 \]

The final bearing is requested from the Java portion of the project and outputted on the screen for the pilot.

The results were accurate but it was enhanced further by searching for the bottom peaks in addition to the top peaks. Searching for the bottom peaks applies the same algorithm as searching for the top peaks but with reverse logic. The average phase difference of both peaks generates a more consistent phase difference resulting in a more stable bearing.

In summary, the data is passed through the radio beacon signal decoding diagram (Figure 5) discussed above to separate the signals. The Morse code identifier is passed on to the audio drivers, the reference and variable signals are filtered through a complex moving average summation to produce a bearing that is displayed to the user. The implementation of Figure 5 can be found in rtl_vor_decoder.h and rtl_vor_decoder.c in Appendix C: Native C Code.
IQ Data

The open source community wrote software using IQ Data to demodulate signals. Their software was used to demodulate the signal from the radio beacon, specifically amplitude and frequency demodulation. The following chapter explaining IQ data is written in thanks to the explanation provided by Alan\(^1\) from Tektronix. Starting with the basics, the following is the equation of RF (Radio Frequency) Modulation.

\[
V(t) = A \sin(2\pi f t + \varphi)
\]

Where:

\[A \text{ is the peak amplitude}\]

\[f \text{ is the frequency}\]

\[t \text{ is time}\]

\[\varphi \text{ is the phase shift}\]

Changing any of the following variable(s) above will modulate the signal, either amplitude modulation, frequency modulation, phase modulation or a combination of the three. There are four internal signals packed into the VOR signal, two of the four are the same frequency. Modulation and filtering makes this possible.

I and Q are a way of describing the magnitude (amplitude) and the phase signal. They are Cartesian translations of the polar amplitude and phase waveforms. They have a phase

\(^1\)http://www.tek.com/blog/what%E2%80%99s-your-iq-%E2%80%93-about-quadrature-signals%E2%80%A6
difference of 90°, which by definition makes them quadrature signals. An example of a quadrature signal is a sine wave and a cosine wave; they have a phase difference of 90°. Both I (In-phase) and Q (Quadrature) represent the amplitudes of their respected signals.

To modulate signals, add the quadrature signals (I*cos() and Q*sin()) together. The summation will be the output as depicted in Figure 7.

![Figure 7 Addition of Quadrature Signals](image)

Based on Figure 7, I = 0 and Q = 1 produce a sine wave which is just a 90° phase shifted signal. I = 1 and Q = 0 produce a cosine wave (no phase shift). I = 0 and Q = 0 produce a flat line (output = 0). I = 1 and Q = 1 produce a new signal that is phase shifted by 45° with a higher amplitude depicted in Figure 8.
Figure 8 New Signal Created Through Summation of Quadrature Signals
One can vary the I and Q values vs. time to create modulated RF signals. I(t) mixes with the RF carrier sine wave, Q(t) mixes with the same RF carrier sine wave at a 90° phase offset, the results are added to create a modulated RF signal, see Figure 9.

Figure 9 Modulate an RF Signal

LO = Local Oscillator

Figure 10 Demodulate an RF Signal
For the purposes of this project, none of the data was modulated, as the modulated signal is received not transmitted. The radio beacon signal is modulated at the source, demodulation is used as one stage of decoding the signal. Demodulation is the same process as modulation in reverse. One can mix the RF signal with local oscillator signals in quadrature to create I and Q baseband signals. Figure 10 is a representation of demodulation.
Finite Impulse response

Two methods exist for generating filters; Finite Impulse Response (FIR) and Infinite Impulse Response (IIR). As the name suggests, IIR filters have an infinite impulse response. They have a feedback loop (recursive); the input of the filter is dependent on the output of the filter. A disadvantage is they may produce unstable results, once unstable, they will remain unstable due to the recursive property. The phase response is not linear and is usually less accurate than FIR filters. The advantages of an IIR filter are the use of less memory, faster computation times and usually produce better results with fewer data points compared to FIR filters. FIR filters have a finite impulse response. They are non-recursive, output only depends on input, therefore produce a stable signal. The phase response is linear and more accurate. The disadvantage with FIR filters consist of requiring more memory, slower computation times and typically require greater amount of input data. FIR was chosen and implemented for the current project due to the vast amount of input data available and the necessity of producing a stable, more accurate response with linear phase.

The Digital Signal Processing toolkit by MATLAB was used to generate the necessary FIR filters. The diagram within depicts the actual parameters used to generate the coefficients. FIR filter algorithms were developed with the use of the generated coefficients from the DSP toolkit to create low-pass and band-pass filters. The filters are one of the steps required to separate the encoded signals.
The DSP toolkit as seen in Figure 11 offered a few design method options to generate FIR filters. Chebyshev type filters were chosen due to their steep roll off characteristics in the unwanted range at the expense of a ripple in the pass band. A high order is required to produce a smooth sinusoidal wave and the order must be less than the number of elements. Given the maximum amount of incoming live data per iteration results in 196 elements, a 190th order filter was chosen to produce the highest possible signal. The Nyquist formula, to double the highest possible frequency, was used to calculate the sampling frequency. The highest possible frequency is 9960Hz + 480Hz = 10440Hz, hence 24KHz was chosen as it just exceeds doubling the frequency. The cutoff frequency
(Fc:) for the low-pass filter is set at 30Hz. All the components are in place to generate the coefficients. Once the coefficients are generated, they are multiplied in reverse order by the live data to filter out the variable signal.

MATLAB displays the magnitude response as well for the designed filter, see Figure 12. The filter contains a steep roll off from 30Hz producing great results.
The 190th order band-pass filter is generated much the same way as the low-pass filter with the exception that band-pass is specified in Response Type and there are two cutoff frequencies. Cutoff frequency 1 (Fc1) is set to 9960Hz ± 480Hz = 9480Hz and cutoff frequency 2 (Fc2) at 9960Hz + 480Hz = 10440Hz. The low-pass filter and band-pass filter use the same equation to generate the appropriate response with the exceptions of their generated coefficients.
The magnitude response for the band-pass filter is shown in Figure 14, the drop-off is evident from all frequencies outside of 9480Hz to 10440Hz. An accurate filter is required to generate clean smooth consistent responses.
Figure 15 FIR Chebyshev Low-pass 30Hz Filter 95th Order
Additionally a 95\textsuperscript{th} order low-pass 30Hz filter was required to decode the reference signal. Unlike the variable signal, the reference signal is encoded via a combination of band-pass filter, frequency modulation and low-pass filter. Decoding it requires it to be band-pass filtered, frequency demodulated, and then low-pass filtered. Frequency demodulating it cuts the data in half resulting in 98 elements from the original 196 elements. Therefore, the original 190\textsuperscript{th} order 30Hz low-pass filter becomes invalid in this case since it is a higher order than the number of elements.

A new issue arises as there are double the variable data elements vs the reference data elements hence a one to one comparison cannot be made. Two options exist, to take
every other piece of variable signal or to cutoff the last half of the variable signal. Trial and error determined option one yielded better results.

Initially a 20th order Chebyshev filter was used to generate the FIR filters. They were used to generate the reference and variable signals. The responses were printed to a text file and plotted in MATLAB. The sine waves became apparent although far from distortion free. The two main causes were the low order FIR filtering implemented (20th order) and an incorrect programmatic representation of the FIR filter. Both first and final version plots are shown below to show the drastic difference in filtering and the correct programmatic representation of FIR filtering. Phase difference between the two signals will be calculated, hence the precision of the sine waves matter. See Decoding Radio Beacon Signals with DSP techniques for more detail.
Figure 17 First Attempt of Decoding Both the Reference and Variable Signals
Figure 18 Final Version of Decoding Both the Reference and Variable Signals

The source data is a raw recording of the VNY VOR from near 16221 Mulholland Dr. Los Angeles, CA 90049. An algorithm was developed to parse out the reference and variable signals from this raw source. DSP (Digital Signal Processing) techniques were utilized to generate clean sine waves with the use of higher order (190th and 95th) FIR filters. The drastic difference is apparent from first attempt to the final version.
Figure 19 Close Up of First Attempt of Decoding the Signals
Figure 19 and Figure 20 are zoomed in versions of the first attempt and final version, respectively. The first attempt generates noisy sine waves and seems to contain a 180° phase shift between each reference sine wave possibly from the incorrect programmatic representation of the initial FIR filter. Research, trial and error yielded much better results as can be seen in the final version. Although the sine waves are not perfect, they contain much less distortion especially considering the data is received using a small piece of hardware (RTL-SDR) which was intended for an entirely other purpose. It is evident that phase difference between the reference and the variable signal is roughly 340° from the raw source. The bearing has been GPS verified as well as compared to an IFR chart. The next challenge was to take these data points and compute a 340° phase difference (bearing) programmatically from these signals.
CHAPTER 5: OPENSLS ES

OpenSL ES (Open Sound Library for Embedded Systems) is a high performance, cross platform, hardware accelerated C-language audio API. It is the audio version of OpenGL, which is used for graphics. OpenSL ES is used to transfer the Morse code from the radio beacon signal to the audio drivers of the Android device.

OpenSL ES drivers were initially used to play audio from a streaming radio station such as 104.3MHz. This was done as a proof of concept to verify its functionality as it is a lower level interpretation of audio drivers in comparison to using a built-in Java Android SDK interface. The latter was considered however due to performance constraints of piping audio buffers consistently through JNI (Java Native Interface), OpenSL ES became the clear choice. The audio buffer was sourced from the C portion of the project and now played through the speakers with a C function call utilizing the OpenSL ES drivers. Efficiency is always best practice in development especially in conjunction with the existing heavy computational tasks of decoding the radio signals.

The interpretation of OpenSL ES is as follows: A startup function called startListening() is called during initialization. The function startListening() creates a mutex called g_mutex. It reserves this g_mutex and creates a thread function called enqueueAudioForPlayback(). The following function contains an infinite while loop with a wait for g_mutex that waits forever until it is notified to continue. Once notified to continue, it enqueues the current audio buffer which translates to playing the audio buffer through the OpenSL ES drivers. The drivers dequeue the audio buffer by playing it through the device speakers. Every time the audio has been dequeued (played through the
audio drivers), a callback function is called which does an assert check to verify no errors exist. It notifies g.mutex is available which releases the mutex for it to be reserved again. In translation, the software plays the 32Kb audio buffer, then plays the next 32Kb audio buffer and the cycle continues. The mutexes create a safe scenario for everything to be played completely so the user will perceive it as one continuous audio whether it is Morse code or listening to the radio station. This is the framework of OpenSL ES.

After initialization, the audio drivers wait for g_mutex to be released since nothing has been enqueued. Two full 32 Kb audio buffers are filled then are both enqueued for playback. Two audio buffers are required at all times to create a seamless audio playback experience for the user. When one audio buffer is playing the next one is being filled as it needs to be ready for playback when the current one completes. Therefore, initially two audio buffers are filled and continued from there to always be one buffer ahead. In simplified terms, 32 Kb of data is roughly one second of audio based on a 24KHz sampling frequency, the audio that is playing is always roughly one second behind of the actual live data.

In summary, following the process ensures complete playback of every audio buffer and seamless transitions to the next audio buffer. Until the current audio buffer is completely played, the mutex is not released for the next audio buffer to continue guarantying complete playback. Two filled audio buffers are required at all times to create the seamless transitions. Everything is possible since the playback function is on a separate thread from the fill buffer thread. While the audio is playing, the next buffer is being filled. Figure 21, Figure 22 and Figure 23 explain the main architecture.
Figure 21 Startup Sequence (Main Java Thread Executes this Call)

Figure 22 Threaded Function to Playback Morse Audio
Figure 23 Morse Code Buffer Written to be Enqueued
CHAPTER 6: ANDROID SPECIFIC

Android Make File

Android NDK (Native Development Kit) is required to compile native C code on Android. All the RTL-SDR software needed to decode the NavAid radio signal is written in C. The make file, Android.mk, contains build instructions for the compiler for the native C software. It specifies the coding language, the dependencies, the type of output, the name of the output libraries, the include paths, etc.

Every new library is sectioned off with include $(CLEAR_VARS) and include $(BUILD_SHARED_LIBRARY). The include $(CLEAR_VARS) command clears all the local variables while the include $(BUILD_SHARED_LIBRARY) builds a shared library. There are other options to build as well such as BUILD_EXECUTABLE, BUILD_STATIC_LIBRARY and BUILD_PREBUILT. The BUILD_SHARED_LIBRARY is needed so that the Java side can communicate with the C side. The C side acts as an API library, which Java will load to make function calls.

LOCAL_MODULE translates to compiled library name. From the following example, it is set to native-audio-jni. This generated library will be called native-audio-jni to which Java must load to access its API.

LOCAL_SRC_FILES specifies the source file(s) that make up the particular module. In this instance, it is native-audio-jni.c in the “src” subfolder.

LOCAL_C_INCLUDES is the path to the include files required by the module. In the example below, two paths are specified.
LOCAL_CFLAGS specifies any additional flags that need to be passed to the compiler. The –DANDROID allows logging of data to LogCat for easier debugging.

LOCAL_LDLIBS specifies libraries to be linked to access to their API. –lOpenSLES allows use of OpenSL ES library drivers required to tap into the audio drivers for audio playback on the device speakers directly from native C. –landroid allows use of the android logging feature which was specified in the LOCAL_CFLAGS as well.

```
include $(CLEAR_VARS)
LOCAL_MODULE    := native-audio-jni
LOCAL_SRC_FILES := src/native-audio-jni.c
LOCAL_C_INCLUDES += $(LOCAL_PATH)/include
$(LOCAL_PATH)/android/include
LOCAL_CFLAGS := -DANDROID -DUSE_LIBLOG
# for native audio
LOCAL_LDLIBS  += -lOpenSLES
# for logging
LOCAL_LDLIBS  += -llog
# for native asset manager
LOCAL_LDLIBS  += -landroid
include $(BUILD_SHARED_LIBRARY)
```

Figure 24 Example of Native C Code Compiled to a Library

Using a combination of the flags specifies the libraries that are built and their dependencies, in addition to their own capabilities and providing access to other libraries.

Compiling this portion is covered in detail in Java Native Interface (JNI) section.
Java Native Interface (JNI)

Java Native Interface is a framework that allows Java to communicate with Native C/C++ code in either direction. A few rules exist for its use. First, the module needs to be compiled by the NDK as a shared library. The Java portion needs to load the library and needs function primitives, for example:

```java
// Load the library
System.loadLibrary("native-audio-jni");
// Function primitive
public static native void createEngine();
```

Setup is required as well on the C side. When an Android project is created, it prompts for a naming convention. The current project has it set to com.example.airguide. JNI calls are declared in AirGuideActivity.java, the main Java file that initializes everything. The naming convention always starts with the keyword Java. Refer to the following as an example:

```java
void Java_com_example_airguide_AirGuideActivity_createEngine(JNIEnv* env, jclass clazz)
```

Arguments require special typecasting to convert Java variable type(s) to C variable type(s). Normal C style code may be written inside the function. The JNI header is required in the C side as well with the following line #include <jni.h>. That concludes the setup for JNI. See Appendix C: Native C Code for software examples, specifically rtl_fm.c.

To call the function from Java, simply call it by the function primitive as follows:
JNI is essential for Air Guide to function as it is written in both Java and C. However due to complexity and performance issues, use should be kept to a minimum. To coincide with this theory, all radio beacon signal decoding capabilities were kept in C. Java would only request bearing and command new tuning frequencies aside from initialization.

Java would poll the bearing at a rate of 1Hz from the UI thread, which made the Android device unresponsive. To overcome this issue, a new thread was created in the Air Guide process, which would poll the data and save it to a member variable at a rate of 1Hz. The UI thread would display the member variable on the Android device screen. The UI thread and the get bearing thread live within the same process; hence, they share the same address space, enabling the UI thread access to the member variable to display on the screen. To protect the variable(s) from race conditions, mutexes are used which are acquired and released during access to the bearing member variable. This ensures that the correct bearing is displayed every time.
Android Manifest

The Android Manifest is an xml file that every Android application needs in order to build. It provides the Android system information about the required permissions, orientation of the application, the sequence of activities in order, etc. There is always one manifest per project and the file is called AndroidManifest.xml.

The first thing listed in the Android Manifest is the target SDK version for the intended application. There is a minimum and a target SDK. The minimum is set to version 15, which is Android 4.0.3, Ice Cream Sandwich. The intended target was the latest version at the time, which was version 21, Android 5.0 Lollipop. The application will still run past the intended target, but will become unreliable. It is best to rebuild the application with the new intended target if a newer one is available. The application will not run on older targets.

The next part of the Android Manifest is the list of permissions. These permissions are needed in order to grant the application access to hardware on the device, i.e. camera, internet, GPS, etc. or access to personal information. Figure 25 is a list of the permissions needed for this project.

```xml
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
```

Figure 25 Android Manifest - Permissions
ACCESS_COARSE_LOCATION and ACCESS_FINE_LOCATION are needed in order to access the location based information on the device. ACCESS_COARSE_LOCATION is a permission that will grant access to the network provider, i.e. the cellular tower the device is connected. ACCESS_FINE_LOCATION is a permission that will grant access to the GPS receiver. This project used this to orient the plane icon when GPS is available. It is also used it to calculate a bearing to a selected radio beacon as a comparator to the generated bearing from radio signals.

MODIFY_AUDIO_SETTINGS is used to change the audio levels when playing back the Morse code.

ACCESS_WIFI_STATE, ACCESS_NETWORK_STATE, and INTERNET are used to access the internet. This was only used during unit testing when the map software was connecting to Google Maps. The Map tiles were downloaded at runtime from the Google server. This API was abandoned for OSMDroid, which allows use of offline map tiles. These permissions are no longer required.

WRITE_EXTERNAL_STORAGE and RECORD_AUDIO were also used only during unit testing. This was needed to record audio files so they could be processed at a later time. Since this project requires a connection to a radio beacon with a direct line of sight, that had to be conducted on-site at specific places in the hills. By creating the audio samples of radio signals, integration and test could be stream lined. These permissions are not needed for the final application.

The next thing listed in the manifest is information about which activity will be launched first. The first activity to be launched is the AirGuideActivity and it will be launched in
landscape mode. It cannot rotate out of that orientation. This Activity is launched when one of two intents take place. When the application icon in the app drawer is clicked, the MAIN intent is activated. The other instance is when a new USB device is attached, the USB_DEVICE_ATTACHED action is activated.

The complete manifest can be found in Appendix B: Android Code on page 163.
AirGuideActivity Layout

The application layout was designed with ADT in Eclipse. It has a user-friendly user interface where one can drag and drop widgets onto a blank canvas to create a custom layout. The output is an xml file that Android uses to create the final application. The bottom layer of the layout is the OsmDroid MapView. On the top left corner is a compass that will always point to true north in relation to the map orientation. The top center of the map contains a formatted table that will display the name of the selected radio beacon selected and the computed bearing relative to it. There are two bearings displayed. The right one is the bearing computed from the radio signal. The left one is the bearing calculated from GPS.

There are also two SlidingDrawers layouts that overlay the map. The first layout on the right side contains several options for the user to change. Among them is the ability for the user to select whether the map can be viewed “North Up” or “Heading Up”. “North Up” means the map will always orient with North pointed directly to the top of the screen. The plane icon will rotate freely. “Heading Up” is where the plane will remain fixed pointing to the top of the screen and the map will rotate freely. Next is the ability to center the map around the plane called “Center on Plane”. Next is a list of NavAid types that the user can chose to show/hide. Lastly is the magnetic declination that is needed to compensate for earth’s magnetic field.

The second layout on the bottom displays detailed information about the NavAid the user has selected. For example in Figure 26, Ventura has been selected and is highlighted blue. The bottom layout displays the selected radio beacon's name, type, channel, frequency and associated airport. When a different NavAid is selected, that layout will be
filled with new information and when the user clicks “connect”, it will highlight blue and tune to the new frequency.

Figure 26 Air Guide Layout with Visible Options
Figure 27 Air Guide Layout with Options Hidden

Figure 26 and Figure 27 show the application's layout of the user interface, the first shows the options to the user, the second shows a full screen mode. The user has the ability to zoom in and out of the map since different zoom levels of map tiles are stored on the device for offline usage in flight.

The resulting xml file can be found in Appendix B: Android Code on page 165.
Map Software Types

The initial map software chosen was Google Maps. Since this application was designed for an Android operating system, which is owned by Google, using its native mapping software was the first choice. Google provides this service free of charge for up to 25,000 map tile downloads per day. After that, there is a usage charge. This was plenty for the purpose of this project. The intention was to have users download the map tiles beforehand of the regions they were going to travel, and use the application offline in the plane, since it is difficult to gain internet access while in flight. During development, it was discovered that Google Maps does not allow users to download and store map tiles for offline usage. This API only works with an active internet connection. Hence, this approach was abandoned. Additional research led us to another API similar to Google Maps called OpenStreetMaps.

OpenStreetMaps, also known as OSMDroid is a free API that allows use of online and offline tile mapping. It provides the ability to use custom tiles. Therefore, users could have the option to switch between map tiles designed for the road and with IFR charts designed for air travel. Figure 28 is an example of an IFR chart from skyvector.com.
The API is available for download at [https://github.com/osmdroid/osmdroid](https://github.com/osmdroid/osmdroid).
Map Tile Creation

Before using OpenStreetMap, the map tiles will need to be downloaded for the region and zoom level of choice. A tool called Mobile Atlas Creator was used for this purpose, http://mobac.sourceforge.net/. This is a free tool that extracts map tiles of your choice and packages them into a zip file. It allows the user to select any region in combination with numerous zoom levels. The size of the download grows exponentially with the closer zoom levels, as more tiles exist to display areas that are more detailed. Download the application from the website, extract the zip file and store the contents on the C drive. Open the extracted folder and run “Mobile Atlas Creator.exe”. Figure 29 shows an example of the application. Version 1.9.16 of Mobile Atlas Creator was used to package the offline map tiles. Use the following options:

<table>
<thead>
<tr>
<th>Map Source: OpenStreetMap MapQuest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zoom Levels: 5 - 11</td>
</tr>
<tr>
<td>Atlas Content:</td>
</tr>
<tr>
<td>- rename Layer</td>
</tr>
<tr>
<td>- Click “Add Selection”</td>
</tr>
</tbody>
</table>

Select the top most left corner of the desired region by clicking the left mouse button and dragging it to the bottom most right corner, then release. This should create a red square of the region you want to download. For the purpose of this project, all of California was highlighted. Parts of the neighboring states and the ocean were selected since California is not square.

The mouse scroll wheel is used to adjust the zoom levels. To pan left/right or up/down, hold the right mouse button down and move the mouse.
Figure 29 Mobile Atlas Creator v1.9.16

Note that the higher the zoom levels require larger selection areas, which in turn requires larger output file sizes. A maximum zoom level of 11 was chosen as this is for aircraft travel. Under “Saved profiles”, click “Create Atlas”. This will start the download process of all the map tiles for your custom region. The created file was roughly 2 GBs in size as a zip file. The zip file contains all the necessary map tiles with the various zoom levels of the selected regions. Rename the file to “osmdroid.zip”. Place this file on the local memory of the Android device in the following path:

```
/sdcard/osmdroid/osmdroid.zip
```

Although the path begins with /sdcard, the directory belongs to local memory.
Radio Beacon Symbols and Locations

Symbols for each type of Radio beacon have to be created. The symbols were obtained from Wikipedia, [http://wiki.flightgear.org/Radio_beacons](http://wiki.flightgear.org/Radio_beacons). An alternate version was created highlighted in blue to depict the selected beacon. The images must be saved as .png files with the background transparent.

![VOR Symbol](image1)
![VORTAC Symbol](image2)
![NDB Symbol](image3)

The next step is to create a lookup table with all the locations of the radio beacons.

Ourairports.com, [http://ourairports.com/data/](http://ourairports.com/data/), provides a downloadable csv file with all the beacons, names, IDs, latitude, longitude, and radio frequencies for the entire world. It will also include other fields that aren’t needed for our application but can be left in because the “GET” commands for the SQL database will skip over them. The file can be filtered with Excel to only contain NavAids for the USA. The table was filtered further to include beacons in California manually using the list from [http://www.worldaerodata.com/nav/US/CALIFORNIA.php](http://www.worldaerodata.com/nav/US/CALIFORNIA.php).

A SQL lite database is created with the CSV file. An open-source database browser was used for SQLite to create the prefilled database. The executable is available for download at [https://github.com/sqlitebrowser/sqlitebrowser/releases](https://github.com/sqlitebrowser/sqlitebrowser/releases).
Use this tool to create a new Database. Name the database as “NavAids.sqlite”

1. Navigate to “File”, then to “Import”
   
   a. Navigate to the .csv file
   
   b. Name the table name as “NavAids_db”
   
   c. Check the “Column names in first line”
   
   d. Field Separator is “Tab”
   
   e. Figure 36 demonstrates how the output should appear.
f. After the csv file has been imported, click on Modify table.

g. Change the format for each of the fields to match Table 1
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Not</th>
<th>PK</th>
</tr>
</thead>
<tbody>
<tr>
<td>_id</td>
<td>INTEGER</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>State</td>
<td>CHAR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>TEXT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>TEXT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>TEXT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Channel</td>
<td>TEXT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Freq</td>
<td>DOUBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latitude</td>
<td>DOUBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latitude_Degree</td>
<td>INTEGER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latitude_Minutes</td>
<td>INTEGER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latitude_Seconds</td>
<td>DOUBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latitude_Direction</td>
<td>TEXT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Longitude</td>
<td>DOUBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Longitude_Degree</td>
<td>INTEGER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Longitude_Minutes</td>
<td>INTEGER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Longitude_Seconds</td>
<td>DOUBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Longitude_Direction</td>
<td>TEXT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Altitude</td>
<td>DOUBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Airport</td>
<td>TEXT</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1 Format of NavAids_db

2. Create a second table called “android_metadata”
   a. Add one field of text data. Name the field “locale”
   b. Insert a row with the value “en_US”

More detail on creating SQLite databases can be found here:


Save the database as “NavAids.sqlite”. Store this file in the assets directory of the Android project. Android’s native Structured Query Language, also known as SQL server, will be used to access the table of information with a simple SQL command at runtime.
CHAPTER 8: SOFTWARE ARCHITECTURE

The Software Architecture is composed of three main threads, the Update Bearing Thread (UBT), the NavAid Decoding Thread (NADT) and the User Interface Thread (UIT) that all run inside the AirGuideActivity Process (AGAP). The AGAP is what is executed when the application first starts. It spawns the UBT, UIT and the NADT, manages the map software and all user interactions. The AGAP contains all USA based radio beacons overlayed on a map based on their geolocations. When one of the beacons is selected, AGAP will transmit the associated frequency to the NADT and poll UBT for an updated bearing. The NADT is in control of accessing the RTL-SDR hardware, retrieving audio samples, and converting those samples into a bearing. See Figure 37 for an illustration. That will be covered in Decoding Radio Beacon Signals with DSP techniques section. The AGAP will take that bearing and display it on the screen.
The AGAP is composed of three main classes, NavAids, LocationOverlay, and MyHeading. Figure 38 illustrates this architecture. There are two helper classes that each of the classes above use known as Type and MyLog. Type contains a few lists of enumerations and a structure used to hold information on the selected radio beacon. MyLog contains functionality used to log operational and debug messages during development and at runtime.
The AGAP first initializes the widgets. The widgets are created with EventListeners that are apart of Android’s background thread. It then loads the OsmDroid map on the screen. The map layout is explained in User Interface / Map section. It then creates a SQL Server and loads the database with the detailed radio beacon data formatted with information as outlined in Table 2. This information is stored in the NavAid Class.
Using the new SQL database, icons for every entry are created and added to the map depending on the radio beacon type and geolocation. With the completion of the map initialization, the UBT is spawned. The operations for UBT are explained in Decoding Radio Beacon Signals with DSP techniques. The GPS manager is called to begin requesting geolocations. This location will update every time the position is changed by one square block. The new location is added to the MyHeading class where the last ten are maintained and a new heading is computed.

AGAP spawns the UIT. This thread will execute at a rate relative to the GPS updates. The purpose of this thread is to update the position of the plane icon within LocationOverlay and orientation of the map. This operation will extract the last location reported by the GPS and create a geolocation. Depending on if the user has chosen “North Up” or “Heading Up”, the map/plane icon is oriented to the new heading and the
plane icon is repositioned. Afterwards the stored location is updated with a new requested GPS.

AGAP spawns the NADT. This process is explained in Decoding Radio Beacon Signals with DSP techniques.

Now that all the initialization for the AGAP is complete, the Android OS is waiting for one of the listeners to be activated.

The first listener to cover is the onClick Listener. This listener is activated when a button is clicked. The two buttons available are the Set Magnetic Declination button and the Connect button.

The Set Magnetic Declination button will take a value from an available text box and add it to the computed bearing from radio signals. This computed bearing is relative to magnetic north although true north is what is required to be displayed. It then updates a text display of the last valid entered value.

The Connect button is displayed when a radio beacon is selected. This will first access the SQL database and extract the frequency of the selected radio beacon. If the frequency is valid, it will also extract the geolocation of the radio beacon and swap the icon with a blue highlighted version. Lastly, it transmits the new frequency to the NADT to process the bearing from the new radio beacon.

The next Listener is the onCheckedChange Listener. This Listener is activated when a check box is (un)checked. The check boxes available are the five different radio beacon types in the right layout window; VOR, VORTAC, VORDME, TACAN, and NDB. This
will hide/show all overlay widgets of that type. It then refreshes the plane icon so it appears on top of all other icons.

The last listener is the onTouch Listener. This Listener is activated when the radio buttons are touched or the compass. The Radio buttons available are “Heading Up” and “North Up”. These will toggle the orientation the user wants the map/plane icon oriented. The compass icon acts like the toggle between the “North Up” and “Heading Up” radio buttons.
CHAPTER 9: TEST

Physical Car Setup

The physical setup in the car is fundamental for this project as high altitude and clear line of sight are required for a clear signal. The car was driven to numerous locations to find the best possible location that happened to be near the intersection of Mulholland Dr. and Corda Dr. This location provided a clear line of sight to KVNY (Van Nuys) VOR. Setup was required to develop, test and integrate the project at the destination.

The antenna is a fundamental piece in receiving the signal. The COTS RTL-SDR provided a 6” antenna but was not capable of receiving the necessary frequency range. Thus, two larger antennas were purchased to replace it. They both were attached to a magnetic base that was mounted on top of the vehicle. The magnetic base acted as an inductor that properly grounded the antenna with the use of the steel body of the entire car resulting in much better receiving capability. Two types of antennas were used: the first had three antennas attached to a large magnetic base and the other was a standard antenna with an average size magnetic base. All antennas are 24” in length. The resulting performance revealed similar performance therefore the smaller antenna was used.
The physical setup requires the following parts: computer (notebook) containing the project, mini wireless N router, Android device (Samsung Galaxy Tab S), RTL-SDR USB dongle, 24” magnetic antenna, male USB to male Micro USB cable for the initial ADB setup, female USB to male micro USB, PAL to BNC adapter. See Figure 40.

The wireless router is required to create a LAN (Local Area Network) for ADB (Android Debug Bridge) communication. The computer and android device are connected to the
same LAN and a wireless ADB connection is established. ADB allows wireless installs of the .apk (executable project for Android) on the Android device and allows use of LogCat for debugging purposes. The Android device is connected to the RTL-SDR. The RTL-SDR is connected to the antenna which is mounted to the steel roof of the car for optimized receiving capability.
Locations and Results

The first location began at the Van Nuys airport right next to the radio beacon. This is located just North of the runway. The testing site began on Chase Ave between Hayvenhurst Ave and Gothic Ave \{34.224822, -118.491597\}. The readings at this location were unreliable due to the short distance to the radio beacon. We were unaware of the minimum distance to the radio beacon for an accurate reading. The ideal minimum distance is one thousand feet above ground level (agl)\(^2\) and at least half a mile away. Using a vehicle to test creates an elevation limitation however elevated locations with clear line of sight were sought out to achieve the best possible location.

The next location was the top level of the CSUN parking structure B5 due to the greater distance and higher elevation. Reception at this location was better but very spotty. The cause was no line-of-sight with the radio beacon and the low elevation. In addition, the RTL-SDR receiver is a weak version, it required a strong signal to properly operate.

The hills in the San Fernando Valley seemed to be the best possible alternative. There are very few spots within the valley that has a direct line-of-sight with the Van Nuys airport that one can reach in a car. Two locations with fairly good coverage were discovered after testing numerous locations. The first was at the southernmost point of Reseda Blvd. \{34.135424, -118.549963\}. This had a decent line-of-sight with the airport and had an elevation of 500 feet. It served as a great test location. The other location was in the parking lot of the Bel Air Presbyterian Church \{34.130626, -118.484659\}, this was the

\(^2\)http://flighttraining.aopa.org/magazine/2000/December/200012_Features_The_ABCs_Of_VORs.html
best location discovered. It includes a full view of the entire valley with clear line of sight to the VNY VOR and an elevation of 500 feet. Air Guide performed best at the church, reporting a consistent bearing of 335°.

After successful verification of decoded bearings from numerous NavAids, the next step was a flight test. The first flight test yielded inconsistent results; GPS updates were faulty, the bearings were oscillating heavily. Further ground tests revealed antenna placement inside the aircraft was the cause. Nonetheless, a few software modifications were made to resolve GPS updates as well as to the decoding bearing algorithm.

A new ground location was discovered off Highland Dr {34.247852, -119.038798} in Camarillo to test transitions between two radio beacons. It has direct line of sight to CMA {34.212519, -119.094353} and VTU {34.115059, -119.049493}. Further ground tests were conducted at this location to simulate the flight test by placing the antenna inside the vehicle. The results coincided with the flight test yielding inconsistent results as well. Outside placement of the antenna directly on the ground yielded inconsistent data as well however with fewer oscillations compared to the inside vehicle antenna placement.

Placing the antenna anywhere other than the roof of the car yields less consistent results especially when not mounted to the cars steel body. Through experimentation, our findings conclude that optimal antenna placement requires a two foot distance from the cars antenna. These finding coincide with the poor results from the flight test. Both radio beacons produced valid bearings confirming our results; CMA with a bearing of 220° and the transition to VTU reported a bearing of 171°.
A second flight test was conducted with an aircraft equipped with an outside mounted antenna with a BNC connector. The results were much more stable although there was a consistent 20° offset from what the bearing should have been. A few possible reasons exist. The first being incompatibility with the outside antenna. The second being that the RTL-SDR is an inexpensive PAL TV Tuner that does not have the hardware to receive signals from great distances. The results from our findings inside our possible realm (ground testing from a car) of testing consistently yielded good results.
User Interface / Map

The original intent for the application was to connect to two different radio beacons at once. Each radio beacon was to provide a bearing relative to its location. Knowing the location and the bearing to each radio beacon, one can use triangulation to find the intersection, relying solely on radio signals rather than GPS for position. This approach was abandoned due to hardware limitations. The RTL-SDR receiver was not powerful enough to switch between radio beacon frequencies fast enough in conjunction with the algorithm to compute the new bearing. Prior to that knowledge, an algorithm was designed for the User Interface Thread (UIT) to verify its operation. Only after, would it be integrated with the Update Bearing Thread (UBT). Hence, a driving class was created to simulate output from the UBT while the decoding algorithm was simultaneously being developed. FileParser and MyTriangulation were created for that purpose.

FileParser simulated outputs that UBT would supply. It first needed an input data file that trends our position traveling in a straight line north at 100 mph. Using the following calculations, an assumption was made that each 0.012 degree change in Latitude or Longitude took one second.

\[
0.1 \text{ Degree} = 7 \text{ mile} \\
0.14 \text{ Degree} = 1 \text{ mile} \\
100 \frac{\text{mi}}{\text{hour}} = 0.85 \frac{\text{mi}}{\text{sec}} \\
(0.85 \frac{\text{mi}}{\text{sec}}) \times (0.014 \frac{\text{Degree}}{\text{mi}}) = 0.012 \frac{\text{Degree}}{\text{sec}}
\]
Using the following calculations, the change in bearing between a radio beacon was simulated; VNY {34.223444, -118.491647}, against the current location, starting at {34.094987, -118.666071}. Note all variables below are in radians.

\[
y = \sin(lon2 - lon1) \times \cos(lat2)
\]

\[
x = \cos(lat1) \times \sin(lat2) - \sin(lat1) \times \cos(lat2) \times \cos(lon2 - lon1)
\]

\[
bearing = \text{atan2}(y, x)
\]

Table 3 shows a sample of output. The Latitude of the current position is incremented by \(0.012 \frac{\text{Degree}}{\text{sec}}\).

<table>
<thead>
<tr>
<th>Us(Degree)</th>
<th>RB(Degree)</th>
<th>Bearing (radians)</th>
<th>Bearing (Degree)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lat1</td>
<td>Lon1</td>
<td>Lat2</td>
<td>Lon2</td>
</tr>
<tr>
<td>34.09</td>
<td>-118.67</td>
<td>34.22</td>
<td>-118.49</td>
</tr>
<tr>
<td>34.11</td>
<td>-118.67</td>
<td>34.22</td>
<td>-118.49</td>
</tr>
<tr>
<td>34.12</td>
<td>-118.67</td>
<td>34.22</td>
<td>-118.49</td>
</tr>
<tr>
<td>34.13</td>
<td>-118.67</td>
<td>34.22</td>
<td>-118.49</td>
</tr>
</tbody>
</table>

Table 3 Sample of bearing calculation

The process is repeated for a second radio beacon, CMA at {34.212519, -119.094353}. Table 4 shows a sample of the final output (Repeated for 10 minutes). Refer to Appendix E: Fileparser Input for the complete input data sheet.
<table>
<thead>
<tr>
<th>Time</th>
<th>UID_1</th>
<th>Bearing_1</th>
<th>UID_2</th>
<th>Bearing_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:00:00</td>
<td>VNY</td>
<td>48.28</td>
<td>CMA</td>
<td>288.47</td>
</tr>
<tr>
<td>1:00:01</td>
<td>VNY</td>
<td>51.05</td>
<td>CMA</td>
<td>286.70</td>
</tr>
<tr>
<td>1:00:02</td>
<td>VNY</td>
<td>54.05</td>
<td>CMA</td>
<td>284.91</td>
</tr>
<tr>
<td>1:00:03</td>
<td>VNY</td>
<td>57.30</td>
<td>CMA</td>
<td>283.08</td>
</tr>
<tr>
<td>1:00:04</td>
<td>VNY</td>
<td>60.81</td>
<td>CMA</td>
<td>281.22</td>
</tr>
<tr>
<td>1:00:05</td>
<td>VNY</td>
<td>64.57</td>
<td>CMA</td>
<td>279.34</td>
</tr>
</tbody>
</table>

Table 4 Sample of FileParser Input

With the simulated UBT output file, FileParser can then parse through the entries and play them back at the rate of the timestamps. An additional class, MyTriangulation, was created to compute the intersection from the radio beacon's reported bearings. Some additional adjustments to the AirGuideActivity class were needed to rely on the output from MyTriangulation for the plane's final location rather than GPS.

The result would depict the plane icon traveling North at a predefined speed, the display at the top center with the bearing values would update, and the map would follow the icon as it travels.
RTL-SDR Testing

Countless amounts of on-site testing became tedious and counter-productive. No internet connection available for research became problematic as well. For this reason, a few modifications were made to the software to log the raw data from the radio beacon signal. A playback feature was added to playback the raw data for decoding purposes in a home environment. The goal was to decode the signal to a certain bearing due to the location of the recorded raw source data.

Once the decoding algorithm was perfected to separate the signals, the reference and variable signals were plotted on MATLAB to verify the results. An algorithm was written to calculate the bearing based on the phase difference of the signals. After a successful attempt of retrieving bearing from the raw source, car testing began again on live data. Modifications were continuously made to produce more stable results and handle special types of conditions. Driving tests were performed within the vicinity to verify bearing was actively changing.
CHAPTER 10: POSSIBLE IMPROVEMENTS

Overlay bearing on map

One area of improvement is to draw the bearing indicator on the map. With this, the operator can visualize what the bearing is to their chosen NavAid. Figure 41 illustrates this idea. The bearing indicator would be shaped like a cone that represents the error associated with the calculated bearing. The farther out from the bearing, the less accurate, within that region, your position. A ghosting effect would be created for the previous NavAid, having it fade away over a 15 minute period. Hence, an intersection can be visualized and the position can be approximated.

Figure 41 Improvement - Overlay Bearing
Better RTL-SDR Hardware

An area of improvement is to try other RTL-SDR USB dongles for performance comparisons. One other RTL-SDR USB dongle called the Newsky TV28T RTL-SDR receiver was used which delivered similar performance to that of the EZCAP EZTV668 RTL-SDR. The EZTV668 RTL-SDR was the popular choice and was used throughout the development of Air Guide due to the performance similarities. No other RTL-SDR was used with the assumption that the performance would be similar, however, this may not have been the case if Air Guide tried all available RTL-SDR dongles. In addition to RTL-SDR’s that used chipsets different from the RTL2832U.
Connect to two NavAids

The improved hardware would enable us to add the capability to the software to connect to two NavAids at once. By connecting to two NavAids, the two bearing cones overlays can be displayed overlapping the region, centering on the current location. See Figure 42 for an illustration.

![Figure 42 Improvement - Overlay Two Bearings](image)

The ability to automatically connect to a closer NavAid if available can also be incorporated. With knowledge of all NavAid locations and the current approximate location, one can determine other NavAids with closer range. This would make the tool completely autonomous and allow the pilot to concentrate on tasks that are more important.
Automatic adjustment to Magnetic Declination

A useful improvement would be the ability to download a table of current Magnetic Declinations based on regions. At the start of the application, with an active internet connection, it would connect to our server and download the current table. So as the aircraft travels, the Magnetic Declination would update based on your new location within each region preventing the bearing from using stale data.
REFERENCES

OsmoSDR, RTL-SDR description, Retrieved August 2014, from

   http://sdr.osmocom.org/trac/wiki/rtl-sdr


   https://etd.ohiolink.edu/!etd.send_file?accession=ohiou1177702951&disposition=inline

Rtl_fm Guide, Description of rtl_fm by Kyle Keen, Retrieved September 2014, from

   http://kmkeen.com/rtl-demod-guide/

Android SDK Developers Site, ADT Eclipse Package, Retrieved July 2014, from


Android NDK Developers Site, Android NDK Bundle, Retrieved May 2015, from

Wikipedia, VOR (VHF Omnidirectional Range), Retrieved July 2015, from

https://en.wikipedia.org/wiki/VHF_omnidirectional_range

Wikipedia, (FIR) Finite Impulse Response, Retrieved October 2014, from


BRK Electronics, radio beacon icon image, Retrieved October 2015, from

http://www.brkelectronics.com/BRK%20images/Onelink%20RF%20Icon.jpg

GIT, Installing Git, Retrieved August 2014, from


GNU Radio, Installing GNU Radio, Retrieved July 2014, from


GNU Radio Build Script, Script by Marcus Leech, Retrieved July 2014, from

http://www.sbrac.org/files/build-gnuradio

Mobile Atlas Creator, Installation and Usage, Retrieved January 2015, from

http://mobac.sourceforge.net/
OpenStreetMap, Various Map Types and Usage, Retrieved August 2014, from https://www.openstreetmap.org/


APPENDIX A: SETUP DEVELOPMENT ENVIRONMENT

The following are step-by-step instructions on the configuration of the actual development environment. The application was developed using the bundled package of ADT (Android Development Tool) which includes a copy of Eclipse IDE. This is all provided by Google from their site http://developer.android.com/tools/help/adt.html. The tools are necessary to generate the .APK file, which is the installation file for Android OS. It is installed as an application, which is then executed by the user.

Setup ADT with Eclipse for Mac OS X

1. Navigate to home folder in Finder, create a folder called “project”
   a. The project along with Eclipse (ADT), NDK will be stored locally in this folder

2. Download adt-bundle-mac-x86_64-20140702.zip from the following link:
   http://dl.google.com/android/adt/adt-bundle-mac-x86_64-20140702.zip

3. Download android-ndk-r10e-darwin-x86_64.bin from the following link:
4. Google will prompt to accept the terms and conditions, check the checkbox after reading and agreeing to the conditions, then download the file.
5. Navigate to downloaded files (typically in the Downloads folder)

6. Move downloaded files to the “project” folder.
   a. `<home>/project/adt-bundle-mac-x86_64-20140702.zip`
   b. `<home>/project/android-ndk-r10e-darwin-x86_64.bin`

7. Type following commands in terminal:
   a. `cd ~/project/folder`
      i. Change directory to `<home>/project/` folder
   b. `ls -l`
      i. Verify files from step 6
   c. `chmod a+x android-ndk-r10e-darwin-x86_64.bin`
   d. `./android-ndk-r10e-darwin-x86_64.bin`

8. Verify “Everything is Ok” at the end of the extraction shown in Figure 46.
9. Double click “adt-bundle-mac-x86_64-20140702.zip” to unzip contents to current directory. Verify two folders are present in <home>/project/ folder

10. Type following commands in terminal to see hidden files:
   a. defaults write com.apple.finder AppleShowAllFiles YES
   b. killall Finder

11. Navigate to <home> directory, create a hidden file labeled “.bash_profile” if one does not exist. Populate file with the following contents:

   #Android tools
   export PATH=${PATH}:~/project/adt-bundle-mac-x86_64-20140702/sdk/tools
   export PATH=${PATH}:~/project/adt-bundle-mac-x86_64-20140702/sdk/platform-tools

   #Android NDK (lets you use idk-build from any directory)
   export PATH=${PATH}:~/project/android-ndk-r10e
   alias nb=ndk-build
   alias nc=ndk-build\ clean
   alias nu='android update project -p . -s --target android-21'
   alias na=cd\ ~/project/AirGuide-Dev\;nc\nb\;nu\;

12. The options from step 11 simplify the process of developing with the following benefits:
   a. The “export” commands are the equivalent of setting environment variables in Windows. They add the paths so it may be visible from any folder on the operating system. Therefore, allowing the user to access functionalities such
as executing “android” (covered below) from any folder without the need to change directory to that specific folder.

b. The “alias” is another useful feature. It creates shortcuts to typing in the entire command lines, equivalent to having variables in code. Essentially typing “nb” in the terminal executes “ndk-build”, etc.

13. Restart the computer to load bash file

14. Optional (not recommended): To hide all hidden files type the following commands in terminal:

   a. defaults write com.apple.finder AppleShowAllFiles NO

   b. killall Finder

15. Type the following in terminal:

   a. android

16. The Android SDK Manager appears. It contains a list of Android SDK libraries available for different versions of Android. Download the necessary version(s). The current Android device (Samsung Galaxy Tab S) requires Android 5.0.1. This version must match Android device OS. Select all the “Tools” as well and Android Support Repository and Android Support Library in the Extras section. Click “Install 49 packages…” which will pop-up a new prompt.

   a. Note: Multiple versions may be downloaded
17. Click the following in order
   a. “Android SDK License”
   b. “Accept License”
   c. “Install”
      i. This process is time consuming
      ii. Verify all contents of an API level are installed
18. MAC OS X comes pre-installed with JDK and JRE, no install required. Do not overwrite preinstalled versions of JDK and JRE from Java site
   a. OS X will prompt for update(s) if one becomes available
19. Navigate to <home>/project/adt-bundle-mac-x86_64-20140702/eclipse from Finder
20. Launch Eclipse
a. For initial launch, user must right-click and select open. Operating system
will not allow “Unauthorized Developer” apps to launch without explicit
permission.

Figure 48 Eclipse Launch Workspace Location Verification

21. Specify workspace path as shown in Figure 48

   a. Set a specific location and leave it unchanged throughout the project life.

   b. Click OK

22. Right click in the area of “Package Explorer” and select “Import”
23. Follow the wizard
   a. Expand Android folder
   b. Double click “Existing Android Code Into Workspace”
24. Click Browse and navigate to project
   a. `<home>/project/AirGuide-Dev`

25. Select AirGuide-Dev and click Open

26. Eclipse automatically populates the table below and checks the checkmark under “Projects to Import” as shown in Figure 51
   a. Verify checkmark is checked
   b. Click “Finish”
Figure 51 Eclipse Verify Android Project to Import

27. Expand AirGuide-Dev project and verify current android version is set to 5.0.1 in this case. To make changes to version:

a. Right click on AirGuide

b. Select Properties

c. Select Android

d. Select the correct version

e. Click Apply

f. Click OK
28. Code is ready to be compiled
   a. To compile native C portion of the project (RTL-SDR portion with math and processor intensive functionality to calculate the bearing, etc.) the user must use the NDK library. The environments were setup in step 11 to simplify this process. Type the following in terminal:
      i. `na`
   b. To compile the Java portion of the project (user interface portion (map, icons, ability to command, display icons, etc.)
      i. Select Eclipse, click on “Project” from the menu bar, click on “Clean…”
      ii. Verify “Build Automatically” is checked
29. Compiling the project creates the necessary .APK file (install file for Android OS) that is ready to be installed on the Android device.
Setup ADT with Eclipse for Ubuntu

How to install RTL-SDR on Ubuntu 13.04:

1. Navigate to http://sdr.osmocom.org/trac/wiki/rtl-sdr

2. Create folder in users home directory:
   a. home/<user>/project

   ![Figure 53 Create a Project Directory in Home Folder](image)

3. GIT is required. This application extracts data from repositories very similar to subversion. Its a very popular source code control tool.

4. Type the following in terminal:
   a. sudo apt-get install git
   b. Type in <password> when prompted.
   c. Reply with “Y” when prompted “Do you want to continue?”
   d. GIT will install

5. GNU Radio is required. Install site:


6. Two choices to install GNU Radio on Ubuntu, Type the following in terminal:
   a. (Not Recommended) sudo apt-get install gnuradio
   b. (Recommended) Script written by Marcus Leech which handles all prerequisites.
      i. wget http://www.sbrac.org/files/build-gnuradio & & chmod a+x ./build-
gnuradio && ./build-gnuradio

ii. Answer “Y” without the quotes to the following prompts:

(1) Proceed? y

(2) Do you have SUDO privileges? Y

iii. This step is very time consuming (up to a few hours).

iv. Verify “All Done” shown in Figure 54

![Install Script Complete](image)

7. Install RTL-SDR, type the following in terminal:

a. cd ~/project

b. git clone git://git.osmocom.org/rtl-sdr.git

i. Grabs latest version of RTL-SDR software and stores in /home/<user>/project folder

c. cd rtl-sdr

d. mkdir build
e. cd build
f. cmake ./
g. make
h. sudo make install
i. Type user password if/when prompted
j. sudo ldconfig

8. Verify RTL-SDR software is installed
   a. Connect RTL-SDR dongle to the laptop via USB
   b. Type in terminal: rtl_fm -f 104.3e6 -W -s 200000 -r 48000 - | aplay -r 48k -f S16_LE
   c. Figure 55 depicts the output. Verify audio playing through computer speakers.

9. Install Java
b. If link above is unreachable, try alternative link


c. Download “Linux x64 *” version from list shown in Figure 56

   i. Current Ubuntu OS is installed as 64-bit, hence Java x64 install required

   ![Java Link to Install JRE](image)

   **Figure 56 Java Link to Install JRE**

   d. Click on “Instructions” link that correlate to the downloaded file for install

   instructions. Steps listed below for simplicity. Type the following in terminal:

   i. cd /usr

   ii. ls

   (1) Displays all files/folders that exist in current folder. If “java” is not among them, then create it with by typing the following in terminal:

   (a) sudo mkdir java
(2) Type password if/when prompted

iii. cd java

e. Move downloaded java zip file to /usr/java with the following commands in terminal:

i. cd /home/<user>/Downloads/

(1) Assuming java zip file exists in /home/<user>/Downloads

ii. sudo mv jre-7u21-linux-x64.tar.gz /usr/java/

iii. cd /usr/java/

iv. sudo tar xzvf jre-7u21-linux-x64.tar.gz

v. sudo rm jre-7u21-linux-x64.tar.gz

(1) Deletes *.tar.gz file after extraction

f. Verify jre1.7.0_21 exists as a folder as depicted in Figure 57

![Figure 57 Installed JRE Verification](image)

10. Install JDK
a. Navigate to


b. Download latest version JDK version (at this time current version is 7u21)

c. Read and click Accept License Agreement to download file

d. Click jdk-7u21-linux-x64.tar.gz

e. Install instructions can be found at

   http://docs.oracle.com/javase/7/docs/webnotes/install/linux/linux-jdk.html

   i. Move the java zip file you just downloaded to /usr/java/

f. Summary of install instructions. Type the following commands in terminal:

   i. cd /home/<user>/Downloads

   ii. sudo mv jdk-7u21-linux-x64.tar.gz /usr/java/.

       (1) Enter password if prompted

       (2) Moving the java zip file to /usr/java

   iii. cd /usr/java/

   iv. sudo tar zxf jdk-7u21-linux-x64.tar.gz

   v. sudo rm jdk-7u21-linux-x64.tar.gz

       (1) Removes the *.tar.gz file after extraction

11. Install Android SDK with Eclipse


   b. Select Linux 64-bit install from the ADT Bundle table since machine is running

       64-bit Ubuntu
c. Click on “adt-bundle-linux-x86_64-20130514.zip”

d. Accept the user agreements to download file.

e. Install instruction site for reference:


f. Extract zip file to /home/<user>/project folder

g. Unzipped folder contains folder labeled eclipse. Open eclipse folder, right click on eclipse, click on Make Link to create a shortcut. Drag and drop shortcut to desktop for simplicity

h. Double-click “link to eclipse” shortcut

i. Chose a workspace throughout life of project. /home/<user>/project/workspace was the default

12. Install Android NDK

b. Click on android-ndk-r8e-linux-x86_64.tar.bz2 to download 64-bit version of Android NDK.

c. Select “Save File” to download file into “Downloads” directory

d. A prompt will appear on the site, select “I have read and agree with the above terms and conditions” and click on Download android-ndk-r8e-linux-x86_64.tar.bz2

e. Navigate to “Downloads” directory in home folder. Extract android-ndk-r8e-linux-x86_64.tar.bz2 to /home/<user>/project/

13. Java, Android SDK and NDK are now installed. Add them to our environment variables

   a. Navigate to the home folder

   b. Press Ctrl+H to see the hidden files and folders

   c. Right-Click on .profile

   d. Click on “Open with text editor”

   e. Add the following to the end of the file (press enter a few times to start a few lines down):

   # AirGuide Start

   # Java (JDK) setup

   export JAVA_HOME="/usr/java/jdk1.7.0_21"

   export PATH=${PATH}:${JAVA_HOME}/bin

   # Android tools

   export PATH=${PATH}:~/project/adt-bundle-linux-x86_64-
20130514/sdk/tools

    export PATH=${PATH}:~/project/adt-bundle-linux-x86_64-

20130514/sdk/platform-tools

    #Android NDK (let's you use ndk-build from any directory)
    export PATH=${PATH}:~/project/android-ndk-r8e
    alias nb=ndk-build
    alias nc=ndk-build\ clean
    alias nu='android update project -p . -s --target android-21’
    alias na=cd\ ~/project/AirGuide-Dev\;nc\;nb\;nu\;

    ########################### AirGuide End ###########################

f. Verify all paths are correct for setup

g. Save and close file

h. Restart or just log out and log in

i. Causes .profile to be reloaded

14. Type command in terminal:

a. sudo apt-get install ia32-libs

i. For 64-bit versions of Ubuntu, this step is required for “adb” command to
   function

15. To import existing project, copy the AirGuide-Dev project to /home/<user>/project/

16. In eclipse, right click and select “Import” in the “Package Explorer” section of
    Eclipse, the empty area (either white or pink)

17. Select Android->Existing Android Code Into Workspace->Next > or double-click
Existing Android Code Into Workspace

18. Select Browse and locate the AirGuide Dev project, select OK and click Finish
19. To compile the native C portion of the project, type the following command in terminal:
   a. Na

20. To compile the Java portion of the project
   a. Click on “Project” from the menu bar
   b. Click on “Clean…”
   c. Verify “Build Automatically” is checked

21. Compiling the project creates the necessary .APK file (install file for Android OS) that is ready to be installed on the Android device.

Figure 60 Verify Android Project to Import
**Setup Android Debug Bridge**

Android Debug Bridge (ADB) is a command line tool which allows wireless communication between the development environment (Eclipse) and the Android device (Samsung Galaxy Tab S). It allows wireless installs of the .APK (install file for android) directly from the IDE (Eclipse) to the Android device. In addition, it allows use of LogCat, a logging device, for debugging purposes. More information can be found on the android developers site: [http://developer.android.com/tools/help/adb.html](http://developer.android.com/tools/help/adb.html)

The use of ADB is critical for our project since the Android tablet only has one micro USB port reserved for the RTL-SDR dongle. Under normal circumstances, development of Android entails emulation of an Android device or a direct connection via USB to the Android device for the .APK installs. ADB enables efficient installs of the compiled .APK file and live logs via LogCat.

ADB resides in `<sdk>/platform-tools/`, if the environment variables were set in the bash script as explained in the Setup ADT with Eclipse for Mac OS X, then access to ADB is available from any path.

Follow the steps below to connect the Android device to the development computer (OS X or Linux):

1. Connect both the Android device and the development computer to the same network
2. Connect the Android device via USB to the development computer
3. Open the terminal, type the following commands:
   a. `adb kill-server`
b. `adb start-server`

c. `adb devices`

![ADB Terminal Output](image)

**Figure 61** ADB Lists Android Device as Hexadecimal Values

4. The Android device will show up in the form of a long list of hexadecimal values followed by device. See Figure 61

5. Type the following in the terminal:

   a. `adb tcpip 5555`
      
      i. Translates to communication via tcp/ip on port 5555

   b. `adb connect <Android IP Address>`
      
      i. In the following case, the `<Android IP address>` = 192.168.0.101, hence “adb connect 192.168.0.101”

6. ADB communication between devices is complete
Open the AirGuide-Dev project using Eclipse and click on the LogCat symbol on the right side of Eclipse. LogCat displays a log of current activity, see Figure 64 for a screenshot example.
The colors vary by priority, listed from lowest to highest priority.

1. verbose (lowest priority),
2. debug
3. info
4. error
5. warn
6. assert (highest priority)

Selecting “debug” prints all priorities excluding verbose.
APPENDIX B: ANDROID CODE

AirGuideActivity.java

package com.example.airguide;
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.concurrent.Semaphore;
import android.app.Activity;
import android.content.ContentResolver;
import android.content.Context;
import android.content.Intent;
import android.hardware.usb.UsbDevice;
import android.hardware.usb.UsbDeviceConnection;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.TextView;
import android.widget.ImageView.ScaleType;
import android.widget.RadioButton;
import android.widget.Toast;
import com.example.airguide.MyLog;
import com.example.airguide.MyLog.LOG_TYPE;
import com.example.airguide.Type.FLIGHT_MODE;
import com.example.airguide.Type.MARKER_COLOR;
import org.osmdroid.ResourceProxy;
import org.osmdroid.tileprovider.tilesource.TileSourceFactory;
import org.osmdroid.tileprovider.tilesource.XYTileSource;
import org.osmdroid.util.GeoPoint;
import org.osmdroid.views.MapView;
import org.osmdroid.views.overlay.ItemizedIconOverlay;
import org.osmdroid.views.overlay.ItemizedIconOverlay.OnItemGestureListener;
import org.osmdroid.views.overlay.ItemizedOverlay;
import org.osmdroid.views.overlay.Overlay;
import org.osmdroid.views.overlay.OverlayItem;
import org.osmdroid.views.overlay.OverlayItem.HotspotPlace;

// Main application starts here.
public class AirGuideActivity extends Activity implements OnTouchListener,
        CheckBox.OnCheckedChangeListener, Button.OnClickListener {

    private static final String CLASS = "AirGuideActivity";
    static {
        try {
            System.loadLibrary("usb");
            System.loadLibrary("rtlsdr");
            System.loadLibrary("native-audio-jni");
            System.loadLibrary("rtlvordecoder");
            System.loadLibrary("rtlfm");

        } catch (Throwable t) {
            Log.w(CLASS, "Failed to load native library:" + t.getMessage(),
            t);
        }
    }

    final Handler mHandler = new Handler(); // Need handler for callbacks to the UI thread
    private final Semaphore mutex = new Semaphore(2, true);

    private EditText etInput;
    private Button btnUpdate;
    private int g_fChannel;
    private Thread myNativeThread = null;
    private SoundPool soundPool;
    private int soundID;
    private final Semaphore mutex = new Semaphore(2, true);
    private final Semaphore mutex = new Semaphore(2, true);
    private final Semaphore mutex = new Semaphore(2, true);

    // Main application starts here.
    public class AirGuideActivity extends Activity implements OnTouchListener,
            CheckBox.OnCheckedChangeListener, Button.OnClickListener {

        private static final String CLASS = "AirGuideActivity";

        static {
            try {
                System.loadLibrary("usb");
                System.loadLibrary("rtlsdr");
                System.loadLibrary("native-audio-jni");
                System.loadLibrary("rtlvordecoder");
                System.loadLibrary("rtlfm");

            } catch (Throwable t) {
                Log.w(CLASS, "Failed to load native library:" + t.getMessage(),
                t);
            }
        }

        final Handler mHandler = new Handler(); // Need handler for callbacks to the UI thread
        private final Semaphore mutex = new Semaphore(2, true);

        private EditText etInput;
        private Button btnUpdate;
        private int g_fChannel;
        private Thread myNativeThread = null;
        private SoundPool soundPool;
        private int soundID;
        private final Semaphore mutex = new Semaphore(2, true);
        private final Semaphore mutex = new Semaphore(2, true);
        private final Semaphore mutex = new Semaphore(2, true);

        thread;
float m_fHeading;
double m_dMagDec = 0;
double[] myLocation = new double[2];
boolean bUpdated = false;
int currentIndex = 1;

private long myProcessingTimeMillis = 0;
public int S_TO_MS = 1000; // ms

Button m_btSetWaypoint, m_btSetMagDec;
CheckBox m_cbVor, m_cbVorDme, m_cbVorTac, m_cbTacan, m_cbNdb,
m_cbCtrOnPlane;
EditText m_etMagDec;
FLIGHT_MODE m_Mode;
ImageView m_ivCompass;
Location m_loc = new Location("dummyprovider");
LocationOverlay myLocationOverlay;
MapView mapView;
MyHeading mh = new MyHeading(10);
MyLog mLog = new MyLog(LOG_TYPE.WARN);
NavAids myDbHelper;
RadioButton m_rbNorthUp, m_rbHeadingUp;
SlidingDrawer sd;
TextView tvName, tvType, tvChannel, tvFreq, tvAirport, tvState, tvItemIndex, tvID, tvMagDec;
Type.Waypoints[] savedPoints = new Type.Waypoints[2];

private LocationManager locMgr = null;

ArrayList<OverlayItem> overlay_List = new ArrayList<OverlayItem>();
ArrayList<OverlayItem> overlay_Vor_List = new ArrayList<OverlayItem>();
ArrayList<OverlayItem> overlay_VorDme_List = new
ArrayList<OverlayItem>();
ArrayList<OverlayItem> overlay_VorTac_List = new ArrayList<OverlayItem>();
ArrayList<OverlayItem> overlay_Tacan_List = new ArrayList<OverlayItem>();
ArrayList<OverlayItem> overlay_Ndb_List = new ArrayList<OverlayItem>();
ArrayList<OverlayItem> overlay_Question_List = new
ArrayList<OverlayItem>();
ArrayList<OverlayItem> overlay_Selected_List = new
ArrayList<OverlayItem>();
ItemizedIconOverlay<OverlayItem> overlayVor,
overlayVorDme, overlayVorTac, overlayTacan, overlayNdb, overlayQuestion, overlaySelected;

// Calls both initialize routines and runThread.
@Override
protected void onCreate(Bundle savedInstanceState) {
    final String TAG = CLASS + "::onCreate";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_air_guide);
    initialize();
    initializeRTL();
    runThread();

    mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
}

// This initializes the map and all the icons that appear on it. It populates the SQL server
// with the radio beacon information.
private void initialize() {
    final String TAG = CLASS + "::initialize";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

    // Initialize heading
    m_fHeading = (float) 0.0;
    m_Mode = FLIGHT_MODE.HEADING_UP;
    myLocation[0] = 34.141930;
    myLocation[1] = -118.538361;
    locMgr = (LocationManager)
        getSystemService(Context.LOCATION_SERVICE);

    // Find Images on display and Initialize Listeners
    m_ivCompass = (ImageView) findViewById(R.id.compass);
    m_rbNorthUp = (RadioButton) findViewById(R.id.rbNorthUp);
    m_rbHeadingUp = (RadioButton) findViewById(R.id.rbHeadingUp);
    m_btSetWaypoint = (Button) findViewById(R.id.btSetWaypoint1);
    m_btSetMagDec = (Button) findViewById(R.id.btSetMagDec);
    m_cbVor = (CheckBox) findViewById(R.id.cbVor);
    m_cbVorDme = (CheckBox) findViewById(R.id.cbVorDme);
    m_cbVorTac = (CheckBox) findViewById(R.id.cbVorTac);
    m_cbTacan = (CheckBox) findViewById(R.id.cbTacan);
    m_cbNdb = (CheckBox) findViewById(R.id.cbNdb);
    m_cbCtrOnPlane = (CheckBox) findViewById(R.id.cbCenterOnMe);
    m_etMagDec = (EditText) findViewById(R.id.etMagDec);
    m_cbVor.setChecked(true);
    m_cbVorDme.setChecked(true);
m_cbVorTac.setChecked(true);
m_cbTacan.setChecked(true);
m_cbNdb.setchecked(true);
m_rbNorthUp.setchecked(true);

m_ivCompass.setOnTouchListener(this);
m_rbNorthUp.setOnTouchListener(this);
m_rbHeadingUp.setOnTouchListener(this);
m_cbVor.setOnCheckedChangeListener(this);
m_cbVorDme.setOnCheckedChangeListener(this);
m_cbVorTac.setOnCheckedChangeListener(this);
m_cbTacan.setOnCheckedChangeListener(this);
m_cbNdb.setOnCheckedChangeListener(this);
m_btSetWaypoint.setOnClickListener(this);
m_btSetMagDec.setOnClickListener(this);

    // Load Map
    mapView = (MapView) findViewById(R.id.mapview);
    mapView.setBuiltInZoomControls(true);
    mapView.setMultiTouchControls(true);
    mapView.setUseDataConnection(false);

    String[] urls = {"http://127.0.0.1"};
    mapView.setTileSource(new XYTileSource("MapQuest", // name of the
main file that has been zipped
            ResourceProxy.string.mapquest_osm,
            2,       // min map zoom level
            12,      // max map zoom level
            256,     // tile size pixels
            ".jpg",  // extension of the tiles (can be ".png" or ".jpg" depending on the
files)
            urls));

    mapView.getController().setZoom(10);
    mapView.getController().setCenter(new GeoPoint(34.19363, -
118.452058));

    // Access SqlSever
    myDbHelper = new NavAids(this);
    try {
        myDbHelper.createDataBase();
    } catch (IOException ioe) {
        throw new Error("Unable to create database");
    }
    try {
        myDbHelper.openDataBase();

110
} catch (SQLException sqle) {
    throw sqle;
}

tvName = (TextView) findViewById(R.id.tvNavAid_Name);
tvType = (TextView) findViewById(R.id.tvNavAid_Type);
tvChannel = (TextView) findViewById(R.id.tvNavAid_Channel);
tvFreq = (TextView) findViewById(R.id.tvNavAid_Freq);
tvAirport = (TextView) findViewById(R.id.tvNavAid_Airport);
tvState = (TextView) findViewById(R.id.tvNavAid_State);
tvItemIndex = (TextView) findViewById(R.id.tvItemIndex);
tvID = (TextView) findViewById(R.id.tvID);
tvMagDec = (TextView) findViewById(R.id.tvCurrentMagDec);
sd = (SlidingDrawer) findViewById(R.id.drawer3);

addNavAidOverlays();

overlayVor = new ItemizedIconOverlay<OverlayItem>(this, overlay_Vor_List, myOnItemGestureListener);
overlayVorDme = new ItemizedIconOverlay<OverlayItem>(this, overlay_VorDme_List, myOnItemGestureListener);
overlayVorTac = new ItemizedIconOverlay<OverlayItem>(this, overlay_VorTac_List, myOnItemGestureListener);
overlayTacan = new ItemizedIconOverlay<OverlayItem>(this, overlay_Tacan_List, myOnItemGestureListener);
overlayNdb = new ItemizedIconOverlay<OverlayItem>(this, overlay_Ndb_List, myOnItemGestureListener);
overlayQuestion = new ItemizedIconOverlay<OverlayItem>(this, overlay_Question_List, myOnItemGestureListener);
overlaySelected = new ItemizedIconOverlay<OverlayItem>(this, overlay_Selected_List, myOnItemGestureListener);

mapView.getOverlays().add(overlayVor);
mapView.getOverlays().add(overlayVorDme);
mapView.getOverlays().add(overlayVorTac);
mapView.getOverlays().add(overlayTacan);
mapView.getOverlays().add(overlayNdb);
mapView.getOverlays().add(overlayQuestion);
mapView.getOverlays().add(overlaySelected);

// Map Refresh
mapView.invalidate();

myLocationOverlay = new LocationOverlay(this, mapView, (CheckBox) findViewById(R.id.cbCenterOnMe));
mapView.getOverlays().add(myLocationOverlay);
myLocationOverlay.showMyIcon();
myLocationOverlay.enableMyLocation();

Type temp = new Type();
savedPoints[0] = temp.new Waypoints();
savedPoints[1] = temp.new Waypoints();

bringPlaneToFront();

m_Mode = FLIGHT_MODE.NORTH_UP;
m_rbNorthUp.setChecked(true);
m_cbCtrOnPlane.setChecked(true);

m_dMagDec = Double.parseDouble(m_etMagDec.getText().toString());

mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");

private void initializeRTL()
{
    mUsbManager = (UsbManager)
    getSystemService(Context.USB_SERVICE);

    // initialize native audio system
    createEngine();
    createBufferQueueAudioPlayer();

    new Thread(new Runnable() {
        public void run() {
            while (true) {
            try {

                // Get a lock
                mutex.acquire();
                // Update the bearing
                g_nBrg = nativeRtlSdrGetBrg();
                // Release the lock
                mutex.release();
                // Run at 1Hz
                Thread.sleep(1000);

            } catch (InterruptedException e) {
                Log.e("d", "Exception Thrown");
                e.printStackTrace();
            }
        }
    });
}
// This extracts all USB devices that are available so we can use the RTL-SDR receiver.
void getDevice()
{
    Intent intent = getIntent();
    Log.d(CLASS, "intent: " + intent);
    String action = intent.getAction();

    UsbDevice device = (UsbDevice) intent.getParcelableExtra(UsbManager.EXTRA_DEVICE);

    if (UsbManager.ACTION_USB_DEVICE_ATTACHED.equals(action))
        Log.d(CLASS, "ACTION_USB_DEVICE_ATTACHED");
    else if (UsbManager.ACTION_USB_DEVICE_DETACHED.equals(action))
        Log.d(CLASS, "ACTION_USB_DEVICE_DETACHED");
    else
        Log.d(CLASS, "NO ACTION TAKEN");

    HashMap<String, UsbDevice> deviceList = mUsbManager.getDeviceList();
    Log.d(CLASS, "Listing keys" + deviceList.keySet());
}

// This starts the processing of the native audio system.
void startNativeThread()
{
    if (myNativeThread == null) {
        new Thread(new Runnable()
        {
            public void run()
            {
                Log.d(CLASS, "Starting nativeMain");

                int retval = nativeRtlSdrInitialize("113.1e6");
                Log.d(CLASS, "Native main returned " + retval);
            }
        }).start();
    }
}
// This opens the RTL-SDR device connected through USB
public UsbDeviceConnection open(String device_name) {
    Log.d(CLASS, "Open called " + device_name);
    UsbDevice usbDevice;
    usbDevice = mUsbManager.getDeviceList().get(device_name);
    if (usbDevice != null) {
        if (mUsbManager.hasPermission(usbDevice)) {
            Log.d(CLASS, "Calling
mUsbManager.openDevice(usbDevice)\n");
            return mUsbManager.openDevice(usbDevice);
        } else {
            Log.d(CLASS, "Missing permissions to open device
");
        }
    }
    return null;
}

OnItemGestureListener<OverlayItem> myOnItemGestureListener = new
ItemizedIconOverlay.OnItemGestureListener<OverlayItem>() {
    // This is called when a radio beacon icon on the map is tapped. It updates
    a local variable with the selected NavAids information.
    @Override
    public boolean onItemSingleTapUp(final int index, final OverlayItem
item) {

        int ii = Integer.parseInt(item.getTitle());
        String currentName = tvName.getText().toString();
        String newName = myDbHelper.getName(ii) + " (" +
        myDbHelper.getID(ii) + "")";

        tvName.setText(myDbHelper.getName(ii) + " (" +
        myDbHelper.getID(ii) + ")");
        tvType.setText("Type: " + myDbHelper.getType(ii));
        tvChannel.setText("Channel: " + myDbHelper.getChannel(ii));
        tvFreq.setText("Frequency: " + myDbHelper.getFreq(ii));
        tvAirport.setText("Airport: " + myDbHelper.getAirport(ii));
        tvState.setText("State: " + myDbHelper.getState(ii));
        tvItemIndex.setText(index + ");
        tvID.setText(ii + ");

        if (sd.isOpened()) {
            if (currentName.compareTo(newName) == 0) {
                sd.close();
            }
        } else {
            sd.open();
}
return true; // We 'handled' this event.

// This is called when a radio beacon icon on the map is long tapped. It does nothing

@Override
public boolean onItemLongPress(final int index, final OverlayItem item) {
    Toast.makeText(AirGuideActivity.this, "Item " + item.getTitle() + ", " + Toast.LENGTH_LONG).show();
    return false;
}

// This starts a thread that updates the Overlays on the map. This needs to be separate so
// the user can still interact with the map.
private void runThread() {
    final String TAG = CLASS + ":runThread";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

    new Thread() {
        public void run() {
            boolean exit = false;
            while (!exit) {
                try {
                    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            updateOverlays();
                        }
                    });
                    Thread.sleep(myLocationOverlay.GPS_UPDATE_INTERVAL);
                } catch (InterruptedException e) {
                    mLog.log(LOG_TYPE.ERROR, TAG, "Exception Thrown");
                    e.printStackTrace();
                    exit = true;
                }
            }
        }
    }.start();
}

mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
// This is called whenever the application is touched. If the compass, heading up, or north up icons are touched, appropriate actions are taken. If not, they are disregarded.
@Override
public boolean onTouch(View v, MotionEvent event) {
    if (event.getAction() == MotionEvent.ACTION_UP) {
        switch (v.getId()) {
        case R.id.compass:
            if (m_Mode == FLIGHT_MODE.NORTH_UP) {
                m_Mode = FLIGHT_MODE.HEADING_UP;
                m_rbHeadingUp.setChecked(true);
            } else {
                m_Mode = FLIGHT_MODE.NORTH_UP;
                m_rbNorthUp.setChecked(true);
            }
            break;
        case R.id.rbHeadingUp:
            m_Mode = FLIGHT_MODE.HEADING_UP;
            m_rbHeadingUp.setChecked(true);
            break;
        case R.id.rbNorthUp:
            m_Mode = FLIGHT_MODE.NORTH_UP;
            m_rbNorthUp.setChecked(true);
            break;
        default:
            break;
        }
    }
    return true;
}

// If one of these check boxes are changed, the overlay icons of that radio beacon type are hidden/shown. The airplane icon is the brought to the front and the map is refreshed.
@Override
public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
    final String TAG = CLASS + '::onCheckedChanged';
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");
    switch (buttonView.getId()) {
    case R.id.cbVor:
        mLog.log(LOG_TYPE.DEBUG, TAG, "VOR checkbox checked");
        if (isChecked)
            mapView.getOverlays().add(overlayVor);
        break;
    case R.id.rbHeadingUp:
        m_Mode = FLIGHT_MODE.HEADING_UP;
        m_rbHeadingUp.setChecked(true);
        break;
    case R.id.rbNorthUp:
        m_Mode = FLIGHT_MODE.NORTH_UP;
        m_rbNorthUp.setChecked(true);
        break;
    default:
        break;
    }
    return true;
}
else
    mapView.getOverlays().remove(overlayVor);
break;
case R.id.cbVorDme:
    mLog.log(LOG_TYPE.DEBUG, TAG, "VOR-DME checkbox checked");
    if (isChecked)
        mapView.getOverlays().add(overlayVorDme);
    else
        mapView.getOverlays().remove(overlayVorDme);
    break;
case R.id.cbVorTac:
    mLog.log(LOG_TYPE.DEBUG, TAG, "VORTAC checkbox checked");
    if (isChecked)
        mapView.getOverlays().add(overlayVorTac);
    else
        mapView.getOverlays().remove(overlayVorTac);
    break;
case R.id.cbTacan:
    mLog.log(LOG_TYPE.DEBUG, TAG, "TACAN checkbox checked");
    if (isChecked)
        mapView.getOverlays().add(overlayTacan);
    else
        mapView.getOverlays().remove(overlayTacan);
    break;
case R.id.cbNdb:
    mLog.log(LOG_TYPE.DEBUG, TAG, "NDB checkbox checked");
    if (isChecked)
        mapView.getOverlays().add(overlayNdb);
    else
        mapView.getOverlays().remove(overlayNdb);
    break;
default:
    break;
}
bringPlaneToFront();

// Refresh the map
mapView.invalidate();

mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
/ This returns the map containing the radio beacon type.  
private ItemizedIconOverlay<OverlayItem> findMapOverlayItem(String _name)  
  {  
    ItemizedIconOverlay<OverlayItem> map = null;  
    if (_name.compareToIgnoreCase("VOR") == 0) {  
      map = overlayVor;  
    } else if (_name.compareToIgnoreCase("VOR-DME") == 0) {  
      map = overlayVorDme;  
    } else if (_name.compareToIgnoreCase("VORTAC") == 0) {  
      map = overlayVorTac;  
    } else if (_name.compareToIgnoreCase("TACAN") == 0) {  
      map = overlayTacan;  
    } else if (_name.compareToIgnoreCase("NDB") == 0) {  
      map = overlayNdb;  
    } else {  
      map = overlayQuestion;  
    }  
    return map;  
  }

  // This is called whenever a radio beacon or the Set Magnitude button is clicked. When  
a radio beacon is clicked, information about is extracted from the SQL server and  
populated in the detail display as well as changing the frequency in the audio system.  
  @Override  
  public void onClick(View arg0) {  
    final String TAG = CLASS + "::onClick";  
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");  
    switch (arg0.getId()) {  
      case R.id.btSetWaypoint1:  
        // Set the frequency based on the user selected overlay  
        double freq =  
          myDbHelper.getFreq(Integer.parseInt(tvID.getText().toString()));  
        if(freq > 0)  
        {  
          savedPoints[0].setID(Integer.parseInt(tvID.getText().toString()));  
          savedPoints[0].setLat(myDbHelper.getLat(savedPoints[0].getID()));  
          swapNavAids(0, Type.MARKER_COLOR.BLUE);  
}

118
myProcessingTimeMillis = SystemClock.elapsedRealtime();
nativeRtlSdrChannel((int) (1000000 * freq));
Toast.makeText(getBaseContext(), "Setting Frequency to " + freq + "MHz",
Toast.LENGTH_SHORT).show();
}
break;
case R.id.btSetMagDec:
m_dMagDec = Double.parseDouble(m_etMagDec.getText().toString());
tvMagDec.setText("Current setting: " + m_dMagDec);
break;
default:
break;
}

mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");

// This will highlight a radio beacon if none are already selected. If one is selected, it will un-highlight the old beacon and highlight the new one.
private void swapNavAids(int _index, Type.MARKER_COLOR _color)
{
    final String TAG = CLASS + "::swapNavAids";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

    String mapOverlayName;
    OverlayItem temp;
    int itemIndex;

    // 1ST NAVAID TO CHECK
    if(overlaySelected.size() == 1)
    {
        int firstSelected = Integer.parseInt(overlaySelected.getItem(0).getTitle());
        // ID MATCH
        if( firstSelected == Integer.parseInt(tvID.getText().toString()) )
        {
            // Do nothing if marker is already in overlay_Selected
            return;
        }
        // COLOR MATCH
        else if(_color == getMarkerColor(overlaySelected, 0))
        {
            // Do nothing if marker is already in overlay_Selected
            return;
        }
    }
temp = overlaySelected.removeItem(0);
mapOverlayName =
myDbHelper.getType(Integer.parseInt(temp.getTitle()));
findMapOverlayItem(mapOverlayName).addItem(temp);
setMarker(temp, MARKER_COLOR.UNKNOWN);

}  
// 2ND NAVAID TO CHECK
else if(overlaySelected.size() == 2)  
{
    mLog.log(LOG_TYPE.VERBOSE, TAG, "HERE 4");
    int firstSelected =
    Integer.parseInt(overlaySelected.getItem(0).getTitle());
    int secondSelected =
    Integer.parseInt(overlaySelected.getItem(1).getTitle());
    // ID MATCH
    if( firstSelected == Integer.parseInt(tvID.getText().toString()) )
    {
        // Do nothing if marker is already in overlay_Selected
        return;
    }
    else if( secondSelected ==
    Integer.parseInt(tvID.getText().toString()) )
    {
        // Do nothing if marker is already in overlay_Selected
        return;
    }
    // COLOR MATCH
    else if(_color == getMarkerColor(overlaySelected, 0))
    {
        temp = overlaySelected.removeItem(0);
        mapOverlayName =
        myDbHelper.getType(Integer.parseInt(temp.getTitle()));
        findMapOverlayItem(mapOverlayName).addItem(temp);
        setMarker(temp, MARKER_COLOR.UNKNOWN);
    }
    // COLOR MATCH
    else if(_color == getMarkerColor(overlaySelected, 1))
    {
        temp = overlaySelected.removeItem(1);
        mapOverlayName =
        myDbHelper.getType(Integer.parseInt(temp.getTitle()));
        findMapOverlayItem(mapOverlayName).addItem(temp);
        setMarker(temp, MARKER_COLOR.UNKNOWN);
    }  
}  


else if(overlaySelected.size() > 2)
{
    // Something went wrong
    mLog.log(LOG_TYPE.ERROR, TAG, "More than 2 OverlayItems in overlay_Selected");
}

// Remove selected overlay
mapOverlayName = dbHelper.getType(savedPoints[_index].getID());
itemIndex = Integer.parseInt(tvItemIndex.getText().toString());
temp = findMapOverlayItem(mapOverlayName).removeItem(itemIndex);

// add overlay to new one
overlaySelected.addItem(temp);
setMarker(temp, _color);
updateSelectedOverlay();

mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");

// This adds selected radio beacons to a new map list that is always displayed. It also can remove them as well.
boolean added = false;
private void updateSelectedOverlay()
{
    if(overlaySelected.size() > 0)
    {
        if(!added)
        {
            mapView.getOverlays().add(overlaySelected);
            added = true;
        }
    }
    else
    {
        if(added)
        {
            mapView.getOverlays().remove(overlaySelected);
            added = false;
        }
    }

    mapView.invalidate();
}
private void bringPlaneToFront() {
    mapView.getOverlays().remove(myLocationOverlay);
    mapView.getOverlays().add(myLocationOverlay);
}

public void setMarker(OverlayItem item, Type.MARKER_COLOR _color) {
    final String TAG = CLASS + "::setMarker";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

    int ii = Integer.parseInt(item.getTitle());
    Drawable marker = null;
    Context mContext = getApplicationContext();

    if (myDbHelper.getType(ii).compareToIgnoreCase("VOR") == 0) {
        switch (_color) {
            case RED:
                marker = mContext.getResources().getDrawable(R.drawable.navaid_vor_red);
                break;
            case BLUE:
                marker = mContext.getResources().getDrawable(R.drawable.navaid_vor_blue);
                break;
            default:
                marker = mContext.getResources().getDrawable(R.drawable.navaid_vor);
                break;
        }
    } else if (myDbHelper.getType(ii).compareToIgnoreCase("VOR-DME") == 0) {
        switch (_color) {
            case RED:
                marker = mContext.getResources().getDrawable(R.drawable.navaid_vor_dme_red);
                break;
            case BLUE:
                marker = mContext.getResources().getDrawable(R.drawable.navaid_vor_dme_blue);
                break;
            default:
                marker = mContext.getResources().getDrawable(R.drawable.navaid_vor_dme);
        }
    }
else if (myDbHelper.getType(ii).compareToIgnoreCase("VORTAC") == 0) {
    switch (_color) {
    case RED:
        marker = mContext.getResources().getDrawable(R.drawable.navaid_vortac_red);
        break;
    case BLUE:
        marker = mContext.getResources().getDrawable(R.drawable.navaid_vortac_blue);
        break;
    default:
        marker = mContext.getResources().getDrawable(R.drawable.navaid_vortac);
        break;
    }
} else if (myDbHelper.getType(ii).compareToIgnoreCase("TACAN") == 0) {
    switch (_color) {
    case RED:
        marker = mContext.getResources().getDrawable(R.drawable.navaid_tacan_red);
        break;
    case BLUE:
        marker = mContext.getResources().getDrawable(R.drawable.navaid_tacan_blue);
        break;
    default:
        marker = mContext.getResources().getDrawable(R.drawable.navaid_tacan);
        break;
    }
} else if (myDbHelper.getType(ii).compareToIgnoreCase("NDB") == 0) {
    switch (_color) {
    case RED:
        marker = mContext.getResources().getDrawable(R.drawable.navaid_ndb_red);
        break;
    case BLUE:
        marker = mContext.getResources().getDrawable(R.drawable.navaid_ndb_blue);
        break;
    default:
        marker = mContext.getResources().getDrawable(R.drawable.navaid_ndb);
        break;
    }
marker =
mContext.getResources().getDrawable(R.drawable.navaid_ndb); 
break;
}
} else {
    marker =
mContext.getResources().getDrawable(R.drawable.orange_pin_question);
}

item.setMarker(marker);
item.setMarkerHotspot(HotspotPlace.CENTER);

mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");

// This will extract the current color of the selected icon.
public Type.MARKER_COLOR
getMarkerColor(ItemizedOverlay<OverlayItem> ol, int _position)
{
    Type.MARKER_COLOR color = Type.MARKER_COLOR.UNKNOWN;
    Context mContext = getApplicationContext();
    Drawable marker = ol.getItem(_position).getMarker(_position);
    if(
        marker.getConstantState().equals(mContext.getResources().getDrawable(R.drawable.navaid_vor_red).getConstantState()) )
            color = Type.MARKER_COLOR.RED;
        else if(marker.getConstantState().equals(mContext.getResources().getDrawable(R.drawable.navaid_vor_blue).getConstantState()) )
            color = Type.MARKER_COLOR.BLUE;
        else if(marker.getConstantState().equals(mContext.getResources().getDrawable(R.drawable.navaid_vor_dme_red).getConstantState()) )
            color = Type.MARKER_COLOR.RED;
        else if(marker.getConstantState().equals(mContext.getResources().getDrawable(R.drawable.navaid_vor_dme_blue).getConstantState()) )
            color = Type.MARKER_COLOR.BLUE;
        else if(marker.getConstantState().equals(mContext.getResources().getDrawable(R.drawable.navaid_vortac_red).getConstantState()) )
            color = Type.MARKER_COLOR.RED;
        else if(marker.getConstantState().equals(mContext.getResources().getDrawable(R.drawable.navaid_vortac_blue).getConstantState()) )
            color = Type.MARKER_COLOR.BLUE;
        else if(marker.getConstantState().equals(mContext.getResources().getDrawable(R.drawable.navaid_tacan_red).getConstantState()) )
            color = Type.MARKER_COLOR.RED;
        else if(marker.getConstantState().equals(mContext.getResources().getDrawable(R.drawable.navaid_tacan_blue).getConstantState()) )
            color = Type.MARKER_COLOR.BLUE;
        else if(marker.getConstantState().equals(mContext.getResources().getDrawable(R.drawable.navaid_compass_red).getConstantState()) )
            color = Type.MARKER_COLOR.RED;
        else if(marker.getConstantState().equals(mContext.getResources().getDrawable(R.drawable.navaid_compass_blue).getConstantState()) )
            color = Type.MARKER_COLOR.BLUE;
        else if(marker.getConstantState().equals(mContext.getResources().getDrawable(R.drawable.navaid_wpt_red).getConstantState()) )
            color = Type.MARKER_COLOR.RED;
        else if(marker.getConstantState().equals(mContext.getResources().getDrawable(R.drawable.navaid_wpt_blue).getConstantState()) )
            color = Type.MARKER_COLOR.BLUE;
        else if(marker.getConstantState().equals(mContext.getResources().getDrawable(R.drawable.navaid_wpt_dme_red).getConstantState()) )
            color = Type.MARKER_COLOR.RED;
        else if(marker.getConstantState().equals(mContext.getResources().getDrawable(R.drawable.navaid_wpt_dme_blue).getConstantState()) )
            color = Type.MARKER_COLOR.BLUE;
        else if(marker.getConstantState().equals(mContext.getResources().getDrawable(R.drawable.navaid_stc_black).getConstantState()) )
            color = Type.MARKER_COLOR.UNKNOWN;
    }
    return color;
}
else if(
    marker.getConstantState().equals(mContext.getResources().getDrawable(R.drawable.navaid_tacan_blue).getConstantState()) ) { color = Type.MARKER_COLOR.BLUE; }
else if(
    marker.getConstantState().equals(mContext.getResources().getDrawable(R.drawable.navaid_ndb_red).getConstantState()) ) { color = Type.MARKER_COLOR.RED; }
else if(
    marker.getConstantState().equals(mContext.getResources().getDrawable(R.drawable.navaid_ndb_blue).getConstantState()) ) { color = Type.MARKER_COLOR.BLUE; }

    return color;
}

// This adds items to the menu that pops up. We don’t use one so this is empty.
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.activity_air_guide, menu);
    MenuInflater blowUp = getMenuInflater();
    blowUp.inflate(R.menu.menu, menu);
    return true;
}

// This updates the orientation of the compass, map, and airplane icon depending on if
// we are north up or heading up.
@SuppressWarnings("unused")
private void updateOverlays()
{
    final String TAG = CLASS + "::updateOverlays";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

    // If GPS is turned on and there is a fix
    if(getGpsStatus())
    {
        int nLat = (int) (myLocation[0] * 1E6);
        int nLon = (int) (myLocation[1] * 1E6);
        GeoPoint gpt = new GeoPoint(nLat, nLon);

        switch (m_Mode) {
            case NORTH_UP:
                mapView.setMapOrientation(0);
                myLocationOverlay.updatePosition(gpt, m_fHeading);
                rotateImage(m_ivCompass, 0, m_ivCompass.getDrawable().getBounds().width() / 2,
m_ivCompass.getDrawable().getBounds().height() / 2);
    break;
    case HEADING_UP:
        mapView.setMapOrientation(-m_fHeading);
        myLocationOverlay.updatePosition(gpt, m_fHeading);
        rotateImage(m_ivCompass, -m_fHeading,
                    m_ivCompass.getDrawable().getBounds().width() / 2,
                    m_ivCompass.getDrawable().getBounds().height() / 2);
        break;
    case NO_UP:
        break;
    default:
        break;
}

Location loc =
    locMgr.getLastKnownLocation(LocationManager.GPS_PROVIDER);
if( loc != null && myLocationOverlay.hasChanged() )
{
    myLocation[0] = loc.getLatitude();
    myLocation[1] = loc.getLongitude();
    mh.addNode(myLocation[0], myLocation[1]);
    m_fHeading = mh.getAverage();
}

if(myLocationOverlay.checkGpsFix())
    myLocationOverlay.showMyIcon();
else
    myLocationOverlay.hideMyIcon();

updateWaypointData();

// Refresh the map
if(getGpsStatus())
{
    if(m_cbCtrOnPlane.isChecked())
    {
        mapView.getController().setCenter(new
        GeoPoint(myLocation[0], myLocation[1]));
        mapView.getController().animateTo(new
        GeoPoint(myLocation[0], myLocation[1]));
    }
mapView.invalidate();

mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");

@Override
protected void onPause() {
    myLocationOverlay.disableMyLocation();
    super.onPause();
}

private void updateWaypointData() {
    final String TAG = CLASS + ":::updateWaypointData";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");
    if(savedPoints[0].getID() > 0) {
        TextView wp1 = (TextView) findViewById(R.id.tvWaypoint1);
        TextView wp2 = (TextView) findViewById(R.id.tvWaypoint2);
        TextView wp1Name = (TextView) findViewById(R.id.tvWaypointTitle1);
        double bearing = mh.getAngleDegree(
            myLocation[0],
            myLocation[1],
            savedPoints[0].getLat(),
            savedPoints[0].getLon());
        bearing = (bearing - m_dMagDec + 360) % 360;
        // Round to 1 digit
        if(myLocationOverlay.checkGpsFix())
            wp1.setText((int)(bearing * 10)) / 10.0 + "";
        else
            wp1.setText("");
        updateWP2((int)(g_nBrg * 10)) / 10.0 + "";
        DecimalFormat df = new DecimalFormat("#.00000");
        final int id = -1;
if (id != savedPoints[0].getID())

wp1Name.setText(myDbHelper.getName(savedPoints[0].getID()));
}

mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
}

// This retrieves the GPS location.
protected void getMyLoc(){

if(getGpsStatus())
{
    final int updateDistance = 10; // meters
    //locListener = new MyLocationListener(getBaseContext());

    locMgr.requestLocationUpdates(LocationManager.GPS_PROVIDER,
    myLocationOverlay.GPS_UPDATE_INTERVAL,
    updateDistance,
    myLocationOverlay);
}
else
{
    Toast.makeText(getBaseContext(),
    "Your GPS is off", Toast.LENGTH_LONG).show();
}

// This checks the status of the GPS; if its enabled or disabled.
private Boolean getGpsStatus() {
    final String TAG = CLASS + ":getGpsStatus";
    ContentResolver contentResolver =
    getBaseContext().getContentResolver();
    boolean gpsStatus =
    Settings.Secure.isLocationProviderEnabled(contentResolver, 
    LocationManager.GPS_PROVIDER);

    return gpsStatus;
}

// This is called when the application starts. It calls getMyLoc, getDevice and
startNativeThread
@Override
protected void onResume() {
    super.onResume();
    myLocationOverlay.enableMyLocation();
    getMyLoc();
    getDevice();
    startNativeThread();
}

// This is called when the application is terminated. Its empty.
@Override
protected void onDestroy() {
    shutdown();
    System.exit(0);
}

// Saves the selected NavAids information to a global variable.
private void setWayPoint(int _id1, int _id2) {
    final String TAG = CLASS + "::setWayPoint";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");
    if(_id1 > 0) {
        savedPoints[0].setId(_id1);
        savedPoints[0].setLat(myDbHelper.getLat(_id1));
        savedPoints[0].setLon(myDbHelper.getLon(_id1));
        mLog.log(LOG_TYPE.VERBOSE, TAG, "savedPoints[0].lat" + myDbHelper.getLat(_id1));
        mLog.log(LOG_TYPE.VERBOSE, TAG, "savedPoints[0].lon" + myDbHelper.getLon(_id1));
    }
    if(_id2 > 0) {
        savedPoints[1].setId(_id2);
        savedPoints[1].setLat(myDbHelper.getLat(_id2));
        savedPoints[1].setLon(myDbHelper.getLon(_id2));
        mLog.log(LOG_TYPE.VERBOSE, TAG, "savedPoints[1].lat" + myDbHelper.getLat(_id2));
        mLog.log(LOG_TYPE.VERBOSE, TAG, "savedPoints[1].lon" + myDbHelper.getLon(_id2));
    }
}
mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
}

// This cycles through the SQL database and adds the radio beacons to the map.
private void addNavAidOverlays() {
    final String TAG = CLASS + "::addNavAidOverlays";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

    GeoPoint point;
    OverlayItem overlayitem;
    String typeRead;
    int count = myDbHelper.getNumRows();
    Drawable Vor =
        this.getResources().getDrawable(R.drawable.navaid_vor);
    Drawable VorDme =
        this.getResources().getDrawable(R.drawable.navaid_vor_dme);
    Drawable VorTac =
        this.getResources().getDrawable(R.drawable.navaid_vortac);
    Drawable Tacan =
        this.getResources().getDrawable(R.drawable.navaid_tacan);
    Drawable Ndb =
        this.getResources().getDrawable(R.drawable.navaid_ndb);
    Drawable Question =
        this.getResources().getDrawable(R.drawable.navaid_ndb);

    for (int i = 1; i <= count; i++) {
        point = new GeoPoint((int)(myDbHelper.getLat(i)*1000000),
            (int)(myDbHelper.getLon(i)*1000000));
        overlayitem = new OverlayItem(String.valueOf(i),
            myDbHelper.getID(i) + " (" + myDbHelper.getName(i) + ")", point);
        overlayitem.setMarkerHotspot(HotspotPlace.CENTER);
        typeRead = myDbHelper.getType(i);
        if (typeRead.compareToIgnoreCase("VOR") == 0) {
            mLog.log(LOG_TYPE.DEBUG, TAG, "VOR added");
            overlayitem.setMarker(Vor);
            overlay_Vor_List.add(overlayitem);
        } else if (typeRead.compareToIgnoreCase("VOR-DME") == 0) {
            mLog.log(LOG_TYPE.DEBUG, TAG, "VOR-DME added");
            overlayitem.setMarker(VorDme);
            overlay_VorDme_List.add(overlayitem);
        } else if (typeRead.compareToIgnoreCase("VORTAC") == 0) {
            mLog.log(LOG_TYPE.DEBUG, TAG, "VORTAC added");
            overlayitem.setMarker(VorTac);
        }
    }
}
overlay_VorTac_List.add(overlayitem);

} else if (typeRead.compareToIgnoreCase("TACAN") == 0) {
    mLog.log(LOG_TYPE.DEBUG, TAG, "TACAN added");
    overlayitem.setMarker(Tacan);
    overlay_Tacan_List.add(overlayitem);
}

else if (typeRead.compareToIgnoreCase("NDB") == 0) {
    mLog.log(LOG_TYPE.DEBUG, TAG, "NDB added");
    overlayitem.setMarker(Ndb);
    overlay_Ndb_List.add(overlayitem);
}

else {
    mLog.log(LOG_TYPE.DEBUG, TAG, "UNKNOWN added");
    overlayitem.setMarker(Question);
    overlay_Question_List.add(overlayitem);
}

overlay_List.add(overlayitem);

mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");

// This rotates an image to the desired angle.
private void rotateImage(ImageView iv, float fAngle, float fPivX, float fPivY) {
    final String TAG = CLASS + ":rotateImage";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

    Matrix matrix = new Matrix();
    iv.setScaleType(ScaleType.MATRIX); // required
    matrix.postRotate((float) fAngle, fPivX, fPivY);
    iv.setImageMatrix(matrix);

    mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
}

// This updates the text display for the bearing value calculated from the RTL-SDR.
private void updateWP2(String angle)
{
    TextView wp2 = (TextView) findViewById(R.id.tvWaypoint2);
    int swVal;
    int numOfSecs = 12;

    if((SystemClock.elapsedRealtime() - myProcessingTimeMillis) <
      (S_TO_MS * numOfSecs))
    {

swVal = (int)((S_TO_MS * numOfSecs) - 
(SystemClock.elapsedRealtime() - myProcessingTimeMillis))/S_TO_MS;
    swVal %= 6;
} else
    swVal = 7;

switch(swVal)
{
    case 0:
        wp2.setText("###.#-#");
        break;
    case 1:
        wp2.setText("###.-#");
        break;
    case 2:
        wp2.setText("##.-##");
        break;
    case 3:
        wp2.setText("#-##.#");
        break;
    case 4:
        wp2.setText("-##.#");
        break;
    case 5:
        wp2.setText("###.##");
        break;
    default:
        wp2.setText(angle);
        break;
}

// native function prototypes
private native int nativeRtlSdrInitialize(String channel);
private native int nativeRtlSdrChannel(int channel);
private native int nativeRtlSdrGetBrg();

// native audio
public static native void createEngine();
public static native void createBufferQueueAudioPlayer();
public static native void shutdown();
}
package com.example.airguide;

import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Matrix;
import android.graphics.Point;
import android.location.GpsStatus;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.location.LocationProvider;
import android.os.Bundle;
import android.view.MotionEvent;
import android.widget.CheckBox;
import android.widget.Toast;
import android.os.SystemClock;
import android.util.Log;
import com.example.airguide.MyLog.LOG_TYPE;
import org.osmdroid.api.IMapController;
import org.osmdroid.util.GeoPoint;
import org.osmdroid.views.MapView;
import org.osmdroid.views.Projection;
import org.osmdroid.views.overlay.MyLocationOverlay;

// This class controls updates to the airplane icon and controls updates from GPS data.
public class LocationOverlay extends MyLocationOverlay implements LocationListener{
    private static final String CLASS = "LocationOverlay";
    private final static int PADDING_ACTIVE_ZOOM = 50;
    private int height;
    private int width;
    private float mRotation = 0;
    private IMapController mc;
    private Bitmap marker;
    private Point currentPoint = new Point();
    private Location lastFix = null;
    private GeoPoint currentGeo = null;
    private org.osmdroid.views.MapView mapView;
    private Context mContext;
    private CheckBox m_cbCenterOnPlane;
    private MyLog mLog = new MyLog();
private LocationManager locationMgr;
private Location lastKnownLocation = null;
private long lastKnownLocationTimeMillis = 0;
public int GPS_UPDATE_INTERVAL = 500; // ms
private Location mLoc = null;
static int count = 0;
static int oldCount = 0;
private boolean showIcon = false;
private double prevLat = 0.0;
private double prevLon = 0.0;

// Constructor that copies some map information from AirGuideActivity and begins requesting updates for GPS locations from the LocationManager.
public LocationOverlay(Context _context, org.osmdroid.views.MapView _mapView, CheckBox _cb) {
    super(_context, _mapView);
    mapView = _mapView;
    mc = _mapView.getController();
    mContext = _context;
    this.marker = BitmapFactory.decodeResource(mContext.getResources(),
        R.drawable.plane_icon);
    this.m_cbCenterOnPlane = _cb;

    locationMgr = (LocationManager) mContext.getSystemService(Context.LOCATION_SERVICE);

    locationMgr.requestLocationUpdates(LocationManager.PASSIVE_PROVIDER, 1, 1, this);
}

// This overrides a function in the class MyLocationOverlay. If we have a GPS fix, the airplane icon is redrawn to the updated location.
@Override
public synchronized void draw(Canvas canvas, MapView mapView, boolean shadow) {
    final String TAG = CLASS + "::draw";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

    if (!shadow) {
        if (isMyLocationEnabled() && currentGeo != null) {
            drawMyLocation(canvas, mapView, lastFix, currentGeo);
        }
    }

    mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
}
protected void drawMyLocation(Canvas canvas, MapView mapView, Location last FIX, GeoPoint myLocation) {
    final String TAG = CLASS + "::drawMyLocation";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

    if (!showIcon)
        return;

    if (this.height == 0) {
        this.height = mapView.getHeight();
        this.width = mapView.getWidth();
    }
    Matrix matrix = new Matrix();
    matrix.setRotate(mRotation, marker.getWidth() / 2, marker.getHeight() / 2);
    Bitmap planeBitmap = Bitmap.createBitmap(marker, 0, 0,
    marker.getWidth(), marker.getHeight(), matrix, true);
    mapView.getProjection().toPixels(myLocation, currentPoint);
    float left = currentPoint.x
    - planeBitmap.getWidth() / 2;
    float top = currentPoint.y
    - planeBitmap.getHeight() / 2;
    canvas.drawBitmap(planeBitmap, left, top, null);

    if (m_cbCenterOnPlane.isChecked())
        inZoomActiveArea(currentPoint);

    mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
}

// This overrides a function is called every time the GPS is updated. It then checks to see if the change in position is large enough to update the airplane icons new position. This prevents any glitching from occurring when we are stationary.
@Override
public synchronized void onLocationChanged(Location loc) {
    super.onLocationChanged(loc);
    final String TAG = CLASS + "::onLocationChanged";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");
    double locDiff = 0.00005;

    handlePositionResults(loc);

    mLoc = loc;
}
// Check if it changed more than a few meters for it to be a valid change
if((Math.abs(prevLat - loc.getLatitude()) < locDiff) &&
   (Math.abs(prevLon - loc.getLongitude()) < locDiff))
   return;

// Store the previous coordinates
prevLat = loc.getLatitude();
prevLon = loc.getLongitude();

count++;
count %= 10000;
if (currentGeo != null && m_cbCenterOnPlane.isChecked()) {
   mc.animateTo(currentGeo);
   mapView.postInvalidate();
}

mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");

// This checks whether we need to recenter the map around the plane icon.
private boolean inZoomActiveArea(Point currentPoint) {
   if ((currentPoint.x > PADDING_ACTIVE_ZOOM && currentPoint.x <
        width - PADDING_ACTIVE_ZOOM) &&
        (currentPoint.y > PADDING_ACTIVE_ZOOM && currentPoint.y <
        height - PADDING_ACTIVE_ZOOM)) {
      return false;
   }
   return true;
}

// This is called every time the map is touched. When the plane is tapped, we (un)check
// the center on plane option and refresh the map
public boolean onTap(GeoPoint p, MapView map) {
   Projection projection = map.getProjection();
   Point tapPoint = projection.toPixels(p, null);
   Point myPoint = projection.toPixels(currentGeo, null);

   // Is it within 20 pixels?
   if (Math.pow(tapPoint.x - myPoint.x, 2.0) + Math.pow(tapPoint.y -
                   myPoint.y, 2.0) < Math.pow(20.0, 2)) {
      m_cbCenterOnPlane.setChecked(true);
      map.invalidate();
      //return dispatchTap();
      return true;
   }
} else {
    return false;
}
}

// This is called when the map is panned. The center on plane is unchecked during this operation.
@override
public boolean onTouchEvent(MotionEvent e, MapView mapView) {
    if(e.getAction() == MotionEvent.ACTION_MOVE) {
        m_cbCenterOnPlane.setChecked(false);
    }
    return super.onTouchEvent(e, mapView);
}

// This sets the member variables heading which is used to orient the angle of the plane icon.
public void updateHeading(float _nAngle) {
    mRotation = _nAngle;
}

// This sets the member variables heading and geolocation.
public void updatePosition(GeoPoint _pt, float _nAngle) {
    currentGeo = _pt;
    mRotation = _nAngle;
}

// This is called when the GPS is turned off and it hides the plane icon.
@Override
public void onProviderDisabled(String provider) {
    Toast.makeText(mContext, "Turned off GPS", Toast.LENGTH_SHORT).show();
    hideMyIcon();
}

// This is called when the GPS is turned on and it shows the plane icon.
@Override
public void onProviderEnabled(String provider) {
    Toast.makeText(mContext, "Turned on GPS", Toast.LENGTH_SHORT).show();
    showMyIcon();
}

// This function overrides a native android function that gets called when the status of the GPS changes between the following: AVAILABLE,
TEMPORARILY_UNAVAILABLE, and OUT_OF_SERVICE. This hides/shows the icon depending on the status.

```java
@override
public void onStatusChanged(String provider, int status, Bundle extras) {
    String statusString = null;
    switch (status) {
        case LocationProvider.OUT_OF_SERVICE:
            if (mLoc == null || mLoc.getProvider().equals(provider)) {
                if(checkGpsFix() == false) {
                    hideMyIcon();
                    mLoc = null;
                }
            }
            break;
        case LocationProvider.TEMPORARILY_UNAVAILABLE:
            if (mLoc == null || mLoc.getProvider().equals(provider)) {
                if(checkGpsFix() == false) {
                    hideMyIcon();
                }
            }
            break;
        case LocationProvider.AVAILABLE:
            showMyIcon();
            break;
    }
}
```

// This sets a member variable and the status of the airplane icon.
```java
public void setShowIcon(boolean si) {
    showIcon = si;
}
```

// This checks whether the position stored has Changed since the last time it was checked.
```java
public boolean hasChanged() {
    if( oldCount == count )
        return false;

    oldCount = count;
    return true;
}
```
return true;
}

// This updates the member variable with a geolocation from the GPS
private void handlePositionResults(Location location) {
    if (location == null)
        return;

    lastKnownLocation = location;
    lastKnownLocationTimeMillis = SystemClock.elapsedRealtime();
}

// This checks if the location provided by GPS is current.
public boolean checkGpsFix() {
    boolean isGpsFix = false;

    if (SystemClock.elapsedRealtime() - lastKnownLocationTimeMillis <
        (GPS_UPDATE_INTERVAL * 6))
        isGpsFix = true;
    else
    {
        // Invalidate last known location in case there is no fix
        lastKnownLocation = null;
    }
    return isGpsFix;
}

// This displays the airplane icon
    boolean isMyIconVisible = false;
    public void showMyIcon(){
            final String TAG = CLASS + "::showMyIcon";
            if( !isMyIconVisible ){
                mLog.log(LOG_TYPE.ERROR, TAG, "myLocationOverlay ADDED");
                setShowIcon(true);
                isMyIconVisible=true;
            }
    }

// This hides the airplane icon
    public void hideMyIcon(){
            final String TAG = CLASS + "::hideMyIcon";
            if( !isMyIconVisible ){
                mLog.log(LOG_TYPE.ERROR, TAG, "myLocationOverlay ADDED");
                setShowIcon(true);
                isMyIconVisible=true;
            }
    }
if (isMyIconVisible)
{
    mLog.log(LOG_TYPE.ERROR, TAG, "myLocationOverlay REMOVED");
    setShowIcon(false);
    isMyIconVisible = false;
}
}
MyHeading.java

package com.example.airguide;

// This class maintains the last N number or locations the GPS reported and computes the average heading. It can also compute an angle between two geolocations
public class MyHeading {
    private int MAXELEMENTS;
    private int nodeCnt = 0;
    private Node root, curr;

    // Class that maintains a linked list of the aircraft’s last 10 positions.
    private class Node {
        Node next = null;
        double lat;
        double lon;

        Node(double _lat, double _lon)
        {
            lat = _lat;
            lon = _lon;
        }
    }

    // Constructor that accepts the maximum number of elements.
    public MyHeading(int maxElements){
        this.MAXELEMENTS = maxElements;
    }

    // This appends a new node with Latitude and Longitude to the end of the linked list. If the list is full, it will remove the first element.
    public void addNode(double lat, double lon)
    {
        // Check if the first root is null
        if(root == null)
        {
            // Create a new node and set as root
            root = new Node(lat, lon);
            curr = root;
            nodeCnt++;
        }
        else
        {
            // Create a new node
            curr.next = new Node(lat, lon);
            // Go to the next node
            // ...
curr = curr.next;
// Increment count
nodeCnt++;
// Move the root node if there are this.MAXELEMENTS elements
if(nodeCnt > this.MAXELEMENTS)
{
    root = root.next;
    nodeCnt--;
}
}

// This loops through a history of positions maintained by the Node Class, computes the
gle and returns the running average.
public int getAverage()
{
    // Make sure there are at least 2 elements in the nodes
    if(nodeCnt < MAXELEMENTS)
        return 0;

double x=0, y=0;
double angle=0;
Node n = root;

while(n.next != null)
{
    angle = getAngle(n.lat, n.lon, n.next.lat, n.next.lon);
    x += Math.cos(angle);
    y += Math.sin(angle);

    n = n.next;
}

x /= (nodeCnt - 1);
y /= (nodeCnt - 1);

angle = Math.atan2(y, x)*180/Math.PI;
angle += 360;
angle %= 360;
return (int)Math.round(angle);
}

// This takes the Latitude and Longitude of two positions and calls getAngle. Then
converts the radians to degrees.
public double getAngleDegree(double lat1, double lon1, double lat2, double lon2)
{  
    return (getAngle(lat1, lon1, lat2, lon2)*180/Math.PI + 360) % 360;
}

// Takes the Latitude and Longitude of two positions and returns the of the two relative to true north in radians.
public double getAngle(double lat1, double lon1, double lat2, double lon2)
{
    double lat1R = Math.PI/180 * lat1;
    double lon1R = Math.PI/180 * lon1;
    double lat2R = Math.PI/180 * lat2;
    double lon2R = Math.PI/180 * lon2;

    double y = Math.sin(lon2R-lon1R) * Math.cos(lat2R);
    double x = Math.cos(lat1R)*Math.sin(lat2R) - Math.sin(lat1R)*Math.cos(lat2R)*Math.cos(lon2R-lon1R);
    double brng = Math.atan2(y, x);

    return brng;
}

// Since this is always running, we assume its never empty so it simply returns false.
public boolean isEmpty() {return false;}
}
NavAids.java

package com.example.airguide;

import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import com.example.airguide.MyLog.LOG_TYPE;

import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import android.database.sqlite.SQLiteOpenHelper;
// This maintains the SQL database of NavAid information
public class NavAids extends SQLiteOpenHelper {
    private static final String CLASS = "NavAids";
    // The Android's default system path of your application database.
    private static String DB_PATH = "/data/data/com.example.airguide/databases/";
    private static String DB_NAME = "NavAid.sqlite";
    private static final String DATABASE_TABLE = "NavAids_db";
    private SQLiteDatabase myDataBase;
    private final Context myContext;
    public static final String KEY_ROWID = "_id";
    public static final String KEY_LAT = "Latitude";
    public static final String KEY_LONG = "Longitude";
    public static final String KEY_TYPE = "Type";
    public static final String KEY_NAME = "Name";
    public static final String KEY_ID = "ID";
    public static final String KEY_FREQ = "Freq";
    public static final String KEY_CHANNEL = "Channel";
    public static final String KEY_STATE = "State";
    public static final String KEY_ALTITUDE = "Altitude";
    public static final String KEY_AIRPORT = "Airport";
    private static boolean copied = false;
    MyLog mLog = new MyLog(MyLog.LOG_TYPE.DEBUG);

    // Constructor
    public NavAids(Context context) {
        super(context, DB_NAME, null, 1);
        myContext = context;
    }
}

// Constructor
public NavAids(Context context) {
    super(context, DB_NAME, null, 1);
    myContext = context;
}

144
// Creates an empty SQL database that will be overwritten with all the radio beacon’s information.

public void createDataBase() throws IOException {
    final String TAG = CLASS + "::createDataBase";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

    boolean dbExist = checkDataBase();
    if (dbExist && copied) {
        mLog.log(LOG_TYPE.DEBUG, TAG, "database already exist");
    } else {
        copied = true;
        mLog.log(LOG_TYPE.DEBUG, TAG, "creating new database");
        // By calling this method an empty database will be created into the default system path of your application so we are gonna be able to overwrite that database with our database.
        this.getReadableDatabase();

        try {
            copyDataBase();
        } catch (IOException e) {
            mLog.log(LOG_TYPE.DEBUG, TAG, "Error copying database");
            throw new Error("Error copying database");
        }

        mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
    }
}

// Check if the database already exist to avoid re-copying the file each time you open the application.
private boolean checkDataBase() {
    SQLiteDatabase checkDB = null;
    final String TAG = CLASS + "::checkDataBase";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

    try {
        String myPath = DB_PATH + DB_NAME;
        checkDB = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.OPEN_READONLY);
    } catch (SQLiteException e) {
        mLog.log(LOG_TYPE.ERROR, TAG, "database does't exist yet.");
    }

    if (checkDB != null) {

mLog.log(LOG_TYPE.INFO, TAG, "database exists");
checkDB.close();
}

mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
return checkDB != null ? true : false;
}

// Copies your database from your local assets-folder to the just created empty
database in the system folder, from where it can be accessed and handled. This is done by
transferring bytestream.
private void copyDataBase() throws IOException {
    final String TAG = CLASS + '::copyDataBase';
mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

    // Open your local db as the input stream
    InputStream myInput = myContext.getAssets().open(DB_NAME);

    // Path to the just created empty db
    String outFileName = DB_PATH + DB_NAME;

    // Open the empty db as the output stream
    OutputStream myOutput = new FileOutputStream(outFileName);

    // transfer bytes from the input file to the output file
    byte[] buffer = new byte[1024];
    int length;
    while ((length = myInput.read(buffer)) > 0) {
        myOutput.write(buffer, 0, length);
    }

    // Close the streams
    myOutput.flush();
    myOutput.close();
    myInput.close();

    mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
}

// Initializes and opens the SQL database
public void openDataBase() throws SQLException {
    final String TAG = CLASS + '::openDataBase';
mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

    String myPath = DB_PATH + DB_NAME;
myDataBase = SQLiteDatabase.openDatabase(myPath, null, SQLiteDatabase.OPEN_READONLY);

    mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");

    // Closes the SQL database safely
    @Override
    public synchronized void close() {
        if (myDataBase != null)
            myDataBase.close();

        super.close();
    }

    // Returns the number of entries in the SQL database
    public int getNumRows() {
        final String TAG = CLASS + '::getNumRows';
        mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

        Cursor c = myDataBase.rawQuery("SELECT COUNT(_id) FROM " + DATABASE_TABLE, null);
        c.moveToFirst();
        int row = c.getInt(0);
        c.close();

        mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
        return row;
    }

    // Accepts a Radio Beacon’s unique ID and returns the Latitude
    public double getLat(int l) {
        final String TAG = CLASS + '::getLat';
        mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

        double temp = getDouble(l, KEY_LAT);

        mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
        return temp;
    }

    // Accepts a Radio Beacon’s unique ID and returns the Longitude
    public double getLon(int l) {
        final String TAG = CLASS + '::getLong';
        mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

        double temp = getDouble(l, KEY_LONG);

        mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
        return temp;
    }
`double temp = getDouble(l, KEY_LONG);

mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
return temp;
}

// Accepts a Radio Beacon’s unique ID and returns the Type
public String getType(int l) {
final String TAG = CLASS + "::getType";
mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

String temp = getString(l, KEY_TYPE);

mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
return temp;
}

// Accepts a Radio Beacon’s unique ID and returns the Name
public String getName(int l) {
final String TAG = CLASS + "::getName";
mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

String temp = getString(l, KEY_NAME);

mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
return temp;
}

// Accepts a Radio Beacon’s unique ID and returns the ID
public String getID(int l) {
final String TAG = CLASS + "::getID";
mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

String temp = getString(l, KEY_ID);

mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
return temp;
}

// Accepts a Radio Beacon’s unique ID and returns the Frequency
public double getFreq(int l) {
final String TAG = CLASS + "::getFreq";
mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

double temp = getDouble(l, KEY_FREQ);
mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
return temp;
}

// Accepts a Radio Beacon’s unique ID and returns the Channel
public String getChannel(int l) {
    final String TAG = CLASS + "::getChannel";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

    String temp = getString(l, KEY_CHANNEL);

    mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
    return temp;
}

// Accepts a Radio Beacon’s unique ID and returns the Airport
public String getAirport(int l) {
    final String TAG = CLASS + "::getAirport";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

    String temp = getString(l, KEY_AIRPORT);

    mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
    return temp;
}

// Accepts a Radio Beacon’s unique ID and returns the State
public String getState(int l) {
    final String TAG = CLASS + "::getState";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

    String temp = getString(l, KEY_STATE);

    mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
    return temp;
}

// Accepts a Radio Beacon’s unique ID and an enumeration of the data to be returned in Integer format.
public int getInteger(int l, String _keyName) {
    final String TAG = CLASS + "::getLong";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

    String[] columns = new String[] { KEY_ROWID, _keyName };
    Cursor c = myDataBase.query(DATABASE_TABLE, columns,
    KEY_ROWID + "=" + l, null, null, null, null);

    // Process the Cursor...
}
int result = -1;

if (c != null) {
    c.moveToFirst();
    result = c.getInt(c.getColumnIndex(_keyName));
}
c.close();

mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
return result;
}

// Accepts a Radio Beacon’s unique ID and an enumeration of the data to be returned in Double format.
public double getDouble(int l, String _keyName) {
    final String TAG = CLASS + "::getDouble";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");
    String[] columns = new String[] { KEY_ROWID, _keyName };
    Cursor c = myDataBase.query(DATABASE_TABLE, columns, KEY_ROWID + "=" + l, null, null, null, null);
    double result = -1;

    if (c != null) {
        c.moveToFirst();
        result = c.getDouble(c.getColumnIndex(_keyName));
    }
c.close();

    mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
    return result;
}

// Accepts a Radio Beacon’s unique ID and an enumeration of the data to be returned in String format.
public String getString(int l, String _keyName) {
    final String TAG = CLASS + "::getString";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");
    String[] columns = new String[] { KEY_ROWID, _keyName };
    Cursor c = myDataBase.query(DATABASE_TABLE, columns, KEY_ROWID + "=" + l, null, null, null, null);
    String result = null;

    if (c != null) {
        c.moveToFirst();
        result = c.getString(c.getColumnIndex(_keyName));
    }
}
c.close();

mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
return result;

// This overrides a function in SQLiteOpenHelper and doesn’t do anything.
@Override
public void onCreate(SQLiteDatabase db) {
}

// This overrides a function in SQLiteOpenHelper and doesn’t do anything.
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
}
MyLog.java

package com.example.airguide;

import android.util.Log;

// This logs all debug commands with androids native logging system
public class MyLog {

    // An enumeration of the log levels allowed.
    public enum LOG_TYPE {
        VERBOSE, DEBUG, INFO, WARN, ERROR, ASSERT, NONE,
        _NULL_
    }

    // A constructor that sets the minimum log level. Defaults to VERBOSE.
    public MyLog() {
        LogLevel = LOG_TYPE.VERBOSE;
    }

    // A constructor that sets the minimum log level to the input.
    public MyLog(LOG_TYPE _ll) {
        if(LogLevel.compareTo(LOG_TYPE._NULL_) == 0)
            LogLevel = _ll;
    }

    // Accepts the log level, a tag, and the string to record. This is command then logs the
    // message in Android’s native logging utility.
    public void log(LOG_TYPE level, String Tag, String Message) {
        if (level.compareTo(LogLevel) < 0)
            return;
    
    switch (level) {
    case VERBOSE:
        Log.v(Tag, Message);
        break;
    case DEBUG:
        Log.d(Tag, Message);
        break;
    case INFO:
        Log.i(Tag, Message);
        break;
    case WARN:
        Log.w(Tag, Message);
        break;
    case ERROR:
Log.e(Tag, Message);
break;
case ASSERT:
    Log.wtf(Tag, Message);
    break;
default:
    break;
}

private static LOG_TYPE LogLevel = LOG_TYPE.NULL;
package com.example.airguide;

// This is a generic class that houses the definitions used in the class AirGuideActivity.
public class Type {

    // An enumeration of the map orientations.
    public enum FLIGHT_MODE {
        UNKNOWN, NORTH_UP, HEADING_UP, NO_UP;
    }

    // An enumeration of the different Radio Beacon Types
    public enum NAVAID_TYPE {
        UNKNOWN, VOR, VOR_DME, VORTAC, TACAN, NDB;
    }

    // An enumeration of Colors used to marker beacons on the map.
    public enum MARKER_COLOR {
        UNKNOWN, BLUE, RED
    }

    // Class is used to maintain the selected NavAids information.
    public class Waypoints {
        // Constructor that sets the ID and SavedOverlayIndex to -1
        Waypoints() {
            m_ID = -1;
            m_nSavedOverlayIndex = -1;
        }

        public void setID(int _ID) {
            m_ID = _ID;
        }

        public void setLat(double _Lat) {
            m_Lat = _Lat;
        }

        public void setLon(double _Lon) {
            m_Lon = _Lon;
        }

        public void setBearing(double _Bearing) {
            m_Bearing = _Bearing;
        }
    }
}
public void setSavedOverlayIndex(int _i) {
    m_nSavedOverlayIndex = _i;
}

public int getID() {
    return m_ID;
};

public double getLat() {
    return m_Lat;
};

public double getLon() {
    return m_Lon;
};

public double getBearing() {
    return m_Bearing;
};

public int getSavedOverlayIndex() {
    return m_nSavedOverlayIndex;
}

private int m_ID;
private double m_Lat;
private double m_Lon;
private double m_Bearing;
private int m_nSavedOverlayIndex;
package com.example.airguide;

import com.example.airguide.MyLog.LOG_TYPE;

// Class was created to unit test the user interface and map software. This simulates the output of the JNI code
public class MyTriangulation {

    private static final String CLASS = "MyTriangulation";
    private static MyLog mLog = new MyLog();

    // Calculates the bearing between two geolocations
    static public double calculateBearing(double _dLat1, double _dLon1, double _dLat2, double _dLon2) {
        final String TAG = CLASS + ":calculateBearing";
        mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

        double bearing = 0.0;
        double dLon = Math.toRadians(_dLon2 - _dLon1);
        double dLat1 = Math.toRadians(_dLat1);
        double dLat2 = Math.toRadians(_dLat2);

        double y = Math.sin(dLon) * Math.cos(dLat2);
        double x = Math.cos(dLat1) * Math.sin(dLat2) - Math.sin(dLat1) * Math.cos(dLat2) * Math.cos(dLon);

        bearing = Math.atan2(y, x);

        mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
        return Math.toDegrees(bearing);
    }

    // Calculates the intersection from two NavAids and their reported bearings.
    static public double[] calculateIntersection(Type.Waypoints _wp1,
                                                Type.Waypoints _wp2) {
        final String TAG = CLASS + ":calculateIntersection";
        mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

        double[] ret = {0.0, 0.0};
        double dLat_1 = Math.toRadians(_wp1.getLat());
        double dLat_2 = Math.toRadians(_wp2.getLat());
        double dLon_1 = Math.toRadians(_wp1.getLon());
        double dLon_2 = Math.toRadians(_wp2.getLon());
        double dLon_d = dLon_2 - dLon_1;

        double dLon_1 = Math.toRadians(_wp1.getLon());
        double dLon_2 = Math.toRadians(_wp2.getLon());
        double dLon_d = dLon_2 - dLon_1;
double dLat_d = dLat_2 - dLat_1;
double brng13 = Math.toRadians(_wp1.getBearing());
double brng23 = Math.toRadians(_wp2.getBearing());

double dist_12 = 2 * Math.asin(Math.sqrt(
    Math.pow(Math.sin(dLat_d/2), 2) +
    Math.cos(dLat_1) * Math.cos(dLat_2) *
    Math.pow(Math.sin(dLon_d/2), 2)
));

double brngA = Math.acos(
    (Math.sin(dLat_2) - Math.sin(dLat_1) * Math.cos(dist_12))
    / (Math.sin(dist_12) * Math.cos(dLat_1))
);  
if (Double.isNaN(brngA)) brngA = 0;  // protect against rounding

double brngB = Math.acos(
    (Math.sin(dLat_1) - Math.sin(dLat_2) * Math.cos(dist_12))
    / (Math.sin(dist_12) * Math.cos(dLat_2))
);  
double brng12, brng21;
if(Math.sin(dLon_2 - dLon_1) > 0){
    brng12 = brngA;
    brng21 = 2 * Math.PI - brngB;
} else{
    brng12 = 2 * Math.PI - brngA;
    brng21 = brngB;
}

double dAlpha_1 = (brng13 - brng12 + Math.PI) % (2*Math.PI) - Math.PI;
double dAlpha_2 = (brng21 - brng23 + Math.PI) % (2*Math.PI) - Math.PI;

double dAlpha_3 = Math.acos(-Math.cos(dAlpha_1) * Math.cos(dAlpha_2) +
    Math.sin(dAlpha_1) * Math.sin(dAlpha_2) * Math.cos(dist_12));
double d_13 = Math.atan2(Math.sin(dist_12) * Math.sin(dAlpha_2) * Math.cos(dAlpha_3),
    Math.cos(dAlpha_2) + Math.cos(dAlpha_1) * Math.cos(dAlpha_3));
ret[0] = Math.asin(Math.sin(dLat_1) * Math.cos(d_13) +
    Math.cos(dLat_1) * Math.sin(d_13) * Math.cos(brng13));
double dDeltaLon_13 = Math.atan2(Math.sin(brng13) * Math.sin(d_13) *
    Math.cos(dLat_1),
    Math.cos(d_13) - Math.sin(dLat_1) * Math.sin(ret[0]));
ret[1] = (dLon_1 + dDeltaLon_13 + 3 * Math.PI) % (2*Math.PI) -
    Math.PI;
ret[0] = Math.toDegrees(ret[0]);
ret[1] = Math.toDegrees(ret[1]);

mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
return ret;
package com.example.airguide;

import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStreamReader;
import java.util.StringTokenizer;
import com.example.airguide.MyLog.LOG_TYPE;
import android.util.Log;

// This class was used for debugging the application. A .csv file was created containing
time stamped radio beacon signals with UID and bearing readings. It would parse all the
information into a local array of ParsedPts.
public class FileParser {
    private static final String CLASS = "FileParser";
    MyLog mLog = new MyLog();

    int ptsRead;
    ParsedPts[] data;
    File f;

    public enum STATE {
        READ, WRITE
    };

    // This class maintains one timestamp of simulated radio signals, housing the Time, and
    2 UIDs and Bearings.
    public class ParsedPts {
        // Constructor that initializes the ID and Bearing to -1.
        public ParsedPts() {
            ID = new int[2];
            Bearing = new double[2];

            for (int ii = 0; ii < 2; ii++) {
                ID[ii] = -1;
                Bearing[ii] = -1;
            }
        }

        public double getTime(){return Time;};
        public long getTimeInMillis(){return (long) (Time * 1000);};
}
public intgetID(int i){return ID[i];};
public double getBearing(int i){return Bearing[i];};

protected void setTime(double _time){Time = _time;};
protected void setID(int i, int _id){ID[i] = _id;};
protected void setBearing(int i, double _Bearing){Bearing[i] = _Bearing;};

private double Time;
private int[] ID;
private double[] Bearing;
}

// Constructor that opens the file to be read with the specified input filename. Calls load() and parseAll()
public FileParser(String _pathName, STATE _state) {
    f = new File(_pathName);
    switch (_state) {
    case READ:
        ptsRead = 0;
        String[] demoPts = load();
        parseAll(demoPts);
        break;
    case WRITE:
        break;
    default:
        break;
    }
}

// This takes an input string and writes it to a file.
public void save(String buf) {
    final String TAG = CLASS + "::save";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");
    try {
        FileOutputStream fos = new FileOutputStream(f);
        fos.write(buf.getBytes());
        fos.close();
        Log.i(TAG, "File successfully saved.");
    } catch (Exception ex) {
        Log.e(TAG, "Error saving file: " + ex.getLocalizedMessage());
    }
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
}
public String[] load() {
    final String TAG = CLASS + "::load";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

    String[] stringArray = new String[4000];
    try {
        if (f.exists() && f.canRead()) {
            FileInputStream fis = new FileInputStream(f);
            DataInputStream in = new DataInputStream(fis);
            BufferedReader br = new BufferedReader(new InputStreamReader(in));
            String line = "";
            while ((line = br.readLine()) != null) {
                stringArray[ptsRead++] = line;
            }
            fis.close();
        } else {
            Log.e(TAG, "File does not exist!");
        }
    } catch (Exception ex) {
        Log.e(TAG, "Error loading file: " + ex.getLocalizedMessage());
    }
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
    return stringArray;
}

private void parseAll(String[] demoPts) {
    final String TAG = CLASS + "::parseAll";
    mLog.log(LOG_TYPE.VERBOSE, TAG, "Enter");

    data = new ParsedPts[ptsRead];
    // skip title row
    for (int i = 1; i < ptsRead; i++) {
        data[i] = new ParsedPts();
        StringTokenizer st = new StringTokenizer(demoPts[i], "t");
    }
}
data[i].Time = Double.parseDouble(st.nextToken());
data[i].ID[0] = Integer.parseInt(st.nextToken());
data[i].Bearing[0] = Double.parseDouble(st.nextToken());
data[i].ID[1] = Integer.parseInt(st.nextToken());
data[i].Bearing[1] = Double.parseDouble(st.nextToken());

mLog.log(LOG_TYPE.VERBOSE, TAG, "Time(" + data[i].Time
+ ") " +
"ID1(" + data[i].ID[0] + ") " +
"Bearing1(" + data[i].Bearing[0] + ") " +
"ID2(" + data[i].ID[1] + ") " +
"Bearing2(" + data[i].Bearing[1] + ")");

mLog.log(LOG_TYPE.VERBOSE, TAG, "Exit");
}
AndroidManifest.xml

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.airguide"
  android:versionCode="1"
  android:versionName="1.0">
  <uses-feature android:name="android.hardware.usb.host" />
  <uses-sdk
    android:minSdkVersion="15"
    android:targetSdkVersion="21"/>
  <uses-permission
    android:name="android.permission.ACCESS_COARSE_LOCATION" />
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
  <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
  <uses-permission android:name="android.permission.INTERNET" />
  <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
  <uses-permission android:name="android.permission.RECORD_AUDIO"/>
  <application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name">
    <activity
      android:name="com.example.airguide.AirGuideActivity"
      android:screenOrientation="landscape"
      android:label="@string/app_name"
      android:theme="@android:style/Theme.NoTitleBar">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
      <intent-filter>
        <action android:name="android.hardware.usb.action.USBDEVICEATTACHED"/>
      </intent-filter>
    </activity>
  </application>
</manifest>
<meta-data
    android:name="android.hardware.usb.action.USBDEVICEATTACHED"
    android:resource="@xml/device_filter" />
</activity>
</application>

</manifest>
Activity_air_guide.xml

<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_gravity="center">
        <org.osmdroid.views.MapView
            android:id="@+id/mapview"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"/>
        <ImageView
            android:id="@+id/compass"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
            android:src="@drawable/compassneedle"/>
    </RelativeLayout>
</FrameLayout>

<TableLayout
    android:layout_width="160dip"
    android:layout_height="75dip"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:background="#404040">
    <TableRow
        android:id="@+id/tableRow1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="5dip">
        <TextView
            android:id="@+id/tvWaypointTitle1"
            android:layout_width="0dip"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:text="Bearing"
            android:textColor="#ffffff"
            android:background="#ff0000"/>
    </TableRow>
</TableLayout>
<table>
<thead>
<tr>
<th>GPS</th>
<th>VOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>###.###</td>
<td>###.###</td>
</tr>
</tbody>
</table>

166
android:layout_height="wrap_content"
android:text="North Up"
android:textColor="#ffffff" />

<RadioButton
    android:id="@+id/rbHeadingUp"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Heading Up"
    android:textColor="#ffffff" />
</RadioGroup>

<CheckBox
    android:id="@+id/cbCenterOnMe"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Center on Plane"
    android:textColor="#ffffff" />

<TextView
    android:id="@+id/textView2"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="#003300"
    android:paddingLeft="10dp"
    android:text="Show NavAids"
    android:textColor="#ffffff" />

<CheckBox
    android:id="@+id/cbVor"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="VOR"
    android:textColor="#ffffff" />

<CheckBox
    android:id="@+id/cbVorDme"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="VOR-DME"
    android:textColor="#ffffff" />

<CheckBox
    android:id="@+id/cbVorTac"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
<CheckBox
    android:id="@+id/cbTacan"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="TACAN"
    android:textColor="#ffffff"/>

<CheckBox
    android:id="@+id/cbNdb"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="NDB"
    android:textColor="#ffffff"/>

<TextView
    android:id="@+id/textView3"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="#003300"
    android:paddingLeft="10dp"
    android:text="Magnetic Declination (deg)"
    android:textColor="#ffffff"/>

<TextView
    android:id="@+id/tvCurrentMagDec"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingLeft="10dp"
    android:text="Current setting: 12.3"
    android:textColor="#ffffff"/>

<TableLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:background="#404040">
    <TableRow
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="5dip">
        <EditText
            android:id="@+id/etMagDec"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingTop="5dip" />
    </TableRow>
</TableLayout>
<TableRow>
   <Button
      android:id="@+id/btSetMagDec"
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:layout_weight=".30"
      android:text="Set" />
</TableRow>

<TableLayout>

<TextView
   android:id="@+id/textView4"
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
   android:background="#003300"
   android:paddingLeft="10dp"
   android:text="About Us"
   android:textColor="#ffffff" />

<TextView
   android:id="@+id/textView5"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:paddingLeft="10dp"
   android:text="AirGuide is designed to assist small aircrafts navigate using only VOR signals. No GPS.\
   Written by Rolando Douglas and Avetis Petrosyan"
   android:textColor="#ffffff" />

</LinearLayout>
</SlidingDrawer>

<SlidingDrawer
   android:id="@+id/drawer3"
   android:layout_width="match_parent"
   android:layout_height="90dp"
   android:layout_alignParentBottom="true"
   android:content="@+id/content"
   android:handle="@+id/handle" >

<View
   android:id="@id/handle"
   android:layout_width="0dp"
   android:layout_height="wrap_content"
   android:layout_weight=".70"
   android:ems="10"
   android:inputType="numberDecimal"
   android:text="12.3" />
</TableLayout>
android:layout_width="match_parent"
android:layout_height="5dip" />

<LinearLayout
android:id="@+id/content"
android:layout_width="wrap_content"
android:layout_height="wrap_content">

<TextView
android:id="@+id/tvID"
android:layout_width="0dip"
android:layout_height="0dip"
android:text="ID" />

<TextView
android:id="@+id/tvItemIndex"
android:layout_width="0dip"
android:layout_height="0dip"
android:text="ItemIndex" />

<TableLayout
android:layout_width="600dip"
android:layout_height="match_parent"
android:background="#404040">

<TableRow
android:id="@+id/tableRow4"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:paddingLeft="10dip"
android:paddingRight="10dip"
android:paddingTop="5dip">

<TextView
android:id="@+id/tvNavAid_Name"
android:layout_width="0dip"
android:layout_weight="40"
android:text="Name (ID)"
android:textColor="#ffffff" />

<TextView
android:id="@+id/tvNavAid_Channel"
android:layout_width="0dip"
android:layout_weight="30"
android:text="Channel:"
android:textColor="#ffffff" />

</TableRow>

</TableLayout>

</LinearLayout>
<table>
<thead>
<tr>
<th>Airport:</th>
<th>Type:</th>
<th>Frequency:</th>
<th>State:</th>
</tr>
</thead>
</table>


<TextView
    android:id="@+id/tvSetWaypointTitle"
    android:layout_width="0dp"
    android:layout_weight="30"
    android:text="Set Waypoint:"
    android:textColor="#ffffff" />

<Button
    android:id="@+id/btSetWaypoint1"
    android:layout_width="40dp"
    android:layout_weight="30"
    android:layout_height="38dp"
    android:text="Connect" />
</TableRow>
</TableLayout>
</LinearLayout>
</SlidingDrawer>
</RelativeLayout>
</FrameLayout>
APPENDIX C: NATIVE C CODE

rtl_vor_decoder.h

#ifndef __RTL_VOR_DECODER_H
#define __RTL_VOR_DECODER_H

#include <stdio.h>
#include <stdlib.h>
#include <android/log.h>

#define DSP_BUF_LENGTH 1024
#define FILTER_LEN 191
#define FILTER_LEN_DIV_2 96
// maximum number of inputs that can be handled
// in one function call
#define MAX_INPUT_LEN 196
#define MAX_INPUT_LEN_FM 98
// maximum length of filter than can be handled
#define MAX_FLT_LEN 192
#define MAX_FLT_LEN_FM 96
// buffer to hold all of the input samples
#define BUFFER_LEN (MAX_FLT_LEN - 1 + MAX_INPUT_LEN)
#define BUFFER_LEN_FM (MAX_FLT_LEN_FM - 1 + MAX_INPUT_LEN_FM)

// Variable signal
struct var_signal
{
    int16_t sig16[DSP_BUF_LENGTH];
    int len;
};

// Reference signal
struct ref_signal
{
    int16_t sig16_fm[DSP_BUF_LENGTH];
    int16_t sig16[DSP_BUF_LENGTH];
    int sig16_len;
    int sig16_fm_len;

    int pre_r;
    int pre_j;
// Morse signal
struct morse
{
    int16_t sig16[DSP_BUF_LENGTH];
    int len;
};

// Chebyshev BP 9480 to 10440Hz (190th Order) Fs = 24K
double coeff_bp9960[FILTER_LEN] =
{
    0.0000018287032693954848, -0.0000018853390672521324,
    0.0000024615659702894056, -0.0000018880726831759971437,
    0.0000059433310258186736, 0.0000050896022568623101,
    -0.000010540826063220009, 0.00001467164941732732,
    0.000014637582644732556, 0.0000083226831759971437,
    0.0000042076884759952232, -0.000019880656908797476,
    0.00003312859384515705, -0.000037928416751089625,
    0.000030730603160209324, -0.000012907810145936849,
    -0.00000882104929365994517, 0.00002486012871388758,
    -0.000027561831340958458, 0.000016319319574277933,
    0.00000000000000000043024912912870240226,
    0.0000056169104808546997, -0.000014395678401305418,
    0.0006383893106801735,
    -0.00012799660848956349, 0.00017349811400341783,
    0.00015969364510112554, 0.00005772308949076063,
    0.00012923907942890783, -0.00035514076200525392,
    0.00053805124320847636, -0.00058682272397870443,
    0.00043985513196181697, -0.00010043684117934513,
    -0.00034902128964304119, 0.00076542118962154217,
    -0.0009443612950967449, 0.00093408877541015765,
    0.00058539888070843293, 0.000064113182604248279,
    0.00043847056332983601, -0.00073503556306724765,
    0.00073004038030534282, -0.0004740196309583211,
    0.00015190146511874725, 0.000000000000000017728055654942677,
    0.00018122119835603772, -0.00067473760412179687,
    0.0012401271994346009, -0.0014905368992705534,
    0.0010618660028594807, 0.00018552297601207035,
    -0.0020253347346688988, 0.0038667803972476551,
    -0.0049297663053359036, 0.0045484249572978405,
    -0.0024888565510157415, -0.0008605181967506984,
    0.0045340391309528693, -0.00728855243666264392,
0.00015190146511874725, -0.00047401963095883211, 0.00073004038030534282, -0.00073503556306724765, -0.00043847056332983601, -0.00058539888070843293, 0.00093408877541015765, -0.00099443612950967449, 0.00076542118962154217, -0.0004902128964304119, -0.00010043684117934513, 0.00043985513196181697, -0.00058682272397870443, 0.00053805124320847636, -0.00635514076200525392, 0.001292390794218907603, -0.00015969364510112554, 0.00017349811400341783, -0.00012799660848956349, 0.000063883931068081735, -0.000014395678401305418, -0.000056169104808546997, 0.0000000000000000004302491286012871388758, -0.0000088210492936994517, -0.000012907810145936849, 0.000030730603160209324, -0.000037928416751089625, 0.000033128593854515705, -0.000019880656908797476, -0.0000042076884759595223, 0.000000083226831759971437, 0.000010540826065320009, 0.00000050896022568623101, 0.0000000059433310258186736, -0.0000018880705242570137, 0.00000024615659702894056, -0.000001885390672521324, 0.0000018287032693954848 ];

double coeff_lp30_95[ FILTER_LEN_DIV_2 ] =
{
0.0000106978549240489, 0.000016610672754147585, 0.000029114814541279318, 0.000047426760666708667, 0.000073291679189699521, 0.00010876359198490538, 0.00015621861932406977, 0.00021836083967564327, 0.00029821939191446704, 0.0003991355686111107, 0.0005247388306302141, 0.00067891091502768959, 0.00086573750717169645, 0.00108447297383222, 0.0013543386352120521, 0.0016646944187954807, 0.0020246863010320248, 0.0024382697427245352, 0.0029090718809280471, 0.0034402745908474488, 0.0040344954856508814, 0.0046936699040979867, 0.0054189371658889705, 0.0062105345159254605, 0.0070677022204162429, 0.0079880632121461777, 0.0089702605048643975, 0.010008515307024717, 0.011098008366397528, 0.01223218657678227, 0.013403336287547524, 0.014602644091072278, 0.015820285140709402, 0.017045538293735913, 0.018266926603057631, 0.019472380922661455,
0.02064942366986056, 0.021785369126723284,
0.02286753686835547, 0.023883468182630169,
0.02482115688081377, 0.02566926192308936, 
0.026417323926248244, 0.027055963944871139,
0.027577065041395791, 0.027973931297141038,
0.028241420223987165, 0.028376045185067773,
0.028376045185067773, 0.028241420223987165,
0.027973931297141038, 0.027577065041395791,
0.026417323926248244, 0.02566926192308936,
0.023883468182630169, 0.02286753686835547,
0.021785369126723284, 0.02064942366986056,
0.019472380922661455, 0.018266926603057631,
0.017045538293735913, 0.015820285140709402,
0.014602644091072278, 0.013403336287547524,
0.01223218657678227, 0.011098008366397528,
0.01008515307024717, 0.0089702605048643975,
0.0079886032121461777, 0.007067702204162429,
0.0062105345159254605, 0.0054189371658889705,
0.0046936699040979867, 0.004034945954856058814,
0.0034402745908474488, 0.0029090718809280471,
0.0024382697427243532, 0.002024686301302048,
0.001664944187954807, 0.0013543386352105201,
0.0010894472973883222, 0.00086573750717169645,
0.00067891091502768959, 0.0005247388306302141,
0.0003991355686161107, 0.0002982193911446704,
0.00021836083967564327, 0.00015621861932406977,
0.00010876359198490538, 0.000073291679198699521,
0.000047426760666708667, 0.000029114814514279318,
0.000016610672754147585, 0.0000106978549240489

};

// Chebyshev LP 30Hz (190th Order) Fs = 24K
double coeff_lp30[ FILTER_LEN ] = 
{
0.0000067819585982025497, 0.00000531417998293001871,
0.0000073739047454319449, 0.0000099379770689245286,
0.000013089168405959946, 0.000016918910722709,
0.00002152769207330409, 0.000027025419207356121,
0.000033531744449653508, 0.000041176346735906871,
0.000050099169055559024, 0.000060450586758049834,
0.000072391532391878418, 0.000086093539781514547,
0.00010173872063827424, 0.0001195196620511545,
0.00013963924076327414, 0.00016231034948426047,
<table>
<thead>
<tr>
<th>Value</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0001877555312184527</td>
<td>0.00021620651775195734</td>
</tr>
<tr>
<td>0.00024790366929693965</td>
<td>0.00028309531278648147</td>
</tr>
<tr>
<td>0.00032203697701489528</td>
<td>0.00036499057239772512</td>
</tr>
<tr>
<td>0.000412223174032457332</td>
<td>0.000464006432025350957</td>
</tr>
<tr>
<td>0.00052061490332457716</td>
<td>0.0005823250177027391</td>
</tr>
<tr>
<td>0.00064941366929693965</td>
<td>0.00072215666028169266</td>
</tr>
<tr>
<td>0.00072215666028169266</td>
<td>0.000885696656205821</td>
</tr>
<tr>
<td>0.00080082732088401895</td>
<td>0.0010750644409914516</td>
</tr>
<tr>
<td>0.00084556469692672925</td>
<td>0.00112922675151529103</td>
</tr>
<tr>
<td>0.000917272462396211</td>
<td>0.0011800681439936666</td>
</tr>
<tr>
<td>0.001029554689692627925</td>
<td>0.0012177779689278927</td>
</tr>
<tr>
<td>0.0010826541440237975</td>
<td>0.00124717440197480497</td>
</tr>
<tr>
<td>0.001130512072629449245</td>
<td>0.00132612533873831745</td>
</tr>
<tr>
<td>0.001191418837003129386</td>
<td>0.00137062456896058602</td>
</tr>
<tr>
<td>0.00126741727462396211</td>
<td>0.00141834372152156976</td>
</tr>
<tr>
<td>0.001358386636699833352</td>
<td>0.001469116646347949</td>
</tr>
<tr>
<td>0.0014229554689692672925</td>
<td>0.0015226958213907892</td>
</tr>
<tr>
<td>0.001483797492465565951879</td>
<td>0.0015789888739987857</td>
</tr>
<tr>
<td>0.001549897492465565951879</td>
<td>0.001663706497219785168</td>
</tr>
<tr>
<td>0.0016055045299934591086</td>
<td>0.0017069707637234203111</td>
</tr>
<tr>
<td>0.00166688082121270737</td>
<td>0.00175835991316208851</td>
</tr>
<tr>
<td>0.0017259075691129519</td>
<td>0.00180239150744250625</td>
</tr>
<tr>
<td>0.00178931677920300487</td>
<td>0.00184826026204418132</td>
</tr>
<tr>
<td>0.001858511669970173312</td>
<td>0.0018943883032920901</td>
</tr>
<tr>
<td>0.00191358767820223195</td>
<td>0.00196051157675278342</td>
</tr>
<tr>
<td>0.0019683636699833352</td>
<td>0.002010641348477519103</td>
</tr>
<tr>
<td>0.002029554689692627925</td>
<td>0.00206127028886186861</td>
</tr>
<tr>
<td>0.0020826541440237975</td>
<td>0.00211710196987020407</td>
</tr>
<tr>
<td>0.002130512072629449245</td>
<td>0.00213465637236904467</td>
</tr>
<tr>
<td>0.0021877555312184527</td>
<td>0.00213763033619265669</td>
</tr>
<tr>
<td>0.002247836997998595</td>
<td>0.00213998513938008727</td>
</tr>
<tr>
<td>0.0022912113681221048</td>
<td>0.00214168981663819753</td>
</tr>
<tr>
<td>0.0023621850260754208</td>
<td>0.0021427217868271105</td>
</tr>
<tr>
<td>0.002388872581265563</td>
<td>0.00214306731372486279</td>
</tr>
<tr>
<td>0.0024092035782056617</td>
<td>0.0021427217868271105</td>
</tr>
<tr>
<td>0.0024229096014945668</td>
<td>0.00214168981663819753</td>
</tr>
<tr>
<td>0.0024298085979002772</td>
<td>0.00213998513938008727</td>
</tr>
<tr>
<td>0.0024298085979002772</td>
<td>0.00213763033619265669</td>
</tr>
<tr>
<td>0.0024298085979002772</td>
<td>0.00213465637236904467</td>
</tr>
<tr>
<td>0.0024298085979002772</td>
<td>0.00213110196987020407</td>
</tr>
</tbody>
</table>
0.012912113681221048, 0.012701282886188681, 0.012478369977998595, 0.012244071995683189, 0.011999114251920427, 0.011744246844703923, 0.011480241085617284, 0.011207885866956925, 0.0097942465565951879, 0.0088260262604418132, 0.0078931677920300487, 0.0069707637234203111, 0.0066688082121270737, 0.0060768633005747709, 0.0044335969855643599, 0.0039409091242295149, 0.0036215258374173145, 0.0030512027629449245, 0.00265641440237975, 0.0022955468962672925, 0.001963636699833352, 0.0016741727462396211, 0.0014118837003129386, 0.00097702192480974962, 0.00080082732088401895, 0.00064941365170771616, 0.00052061490332457532, 0.00041223174278300991, 0.00032203679704485928, 0.00024790366929693466, 0.000187555312184527, 0.00013963924076327414, 0.00010173872063827424, 0.000072391532391878418, 0.00005009916905559024, 0.000033531744449653508, 0.00002152769207330409, 0.000013089168405959946, 0.0000073739047454319449, 0.0000067819585982025497

};

// Chebyshev BP 990 to 1050Hz (190th order) Fs = 24K
double coeff_morse[ FILTER_LEN ] =

180
0.0065794075927196674, 0.013651848961132425,
0.019804319921949538, 0.024569203867615564,
0.02758251041559753, 0.026168521762923096,
0.02758251041559753, 0.028613318149877054,
0.019804319921949538, 0.013651848961132425,
0.0065794075927196674, -0.00087940345352275316,
-0.0081676496354724307, -0.014749129853260505,
-0.020150186220651758, -0.023995819999930515,
0.026037211647164635, -0.026168521762923096,
0.016208024151163552, -0.014749129853260505,
0.0041259729846933506, -0.010426491910951552,
0.0081159118354676846, 0.013175593462150213,
0.017056975797512849, 0.019943701805858489,
0.020491007911653224, 0.0064981213655342209,
0.018014167668945535, 0.0025667371610071307,
0.010970674952601709, -0.0096678871743769157,
0.0018687915152673346, -0.01311724877621136,
-0.0064962709342183175, 0.0022104513531651048,
0.0081159118354676846, -0.013175593462150213,
0.017056975797512849, 0.019943701805858489,
0.020491007911653224, 0.0064981213655342209,
0.018014167668945535, -0.0064962709342183175,
0.010970674952601709, 0.0022104513531651048,
0.0018687915152673346, -0.0064962709342183175,
-0.0064962709342183175, -0.0086832251705908203,
-0.0006441945936784347, -0.0032304181709916966,
0.00046688713067770355, 0.0020466604349128456,
0.004149203460257711, 0.0057274907336080866,
0.0067203208461758188, 0.0071718326956823313,
0.0069583894433408525, 0.006317559176835197,
0.0053007149669341324, 0.004029884325845912,
0.0026322737832265063, 0.0012293892701235813,
0.000072113402588400488, -0.0011872562582104491,
-0.002056923990311143, -0.002649355553760654,
-0.0029591377101518082, -0.0030040958419678405,
0.0028206184119652206, -0.0024580264607269851,
-0.0019726093149101259, -0.0014218985561807065,
-0.00085965608846096958, -0.00033192347929174,
0.00012566373982985174, 0.00048924121804917676,
0.00074682874692224676, 0.00089737340673703,
0.00094897546193956788, 0.0009165828538640193,
0.0008194633924096625, 0.00067871578336285417,
0.00051050345723562167, 0.00034702047448588187,
0.00018973629530784122, 0.000054195450102726804,
0.000052909703928629177, -0.00012892177435594725,
-0.00017461593723950655, -0.00019338589827072193,
-0.00019034079597882227, -0.00017142204094157867,
0.00012566373982985174, 0.00048924121804917676,
};

void decodeBrg(int _preJ, int _preR, int16_t *sig16, int _sig16_len, int* _brg, int* _count);
void deleteBrgNodes();

#endif /* __RTL_VOR_DECODER_H */
#include "rtl_vor_decoder.h"
#include <math.h>

// This value is important - consider moving it to the top of the file
#define NUM_OF_SAMPLES 5000
#define FILTER_AMOUNT 135
#define MAX_PEAKS 18 // was 20
#define BRG_FLTR_SIZE 30
#define MAX_BRG_NODES 30

#define DEBUG_PRINT 0

// array to hold morse samples
double insamp_morse[ BUFFER_LEN ];
// array to hold input samples
double insamp_var[ BUFFER_LEN ];
double insamp_ref[ BUFFER_LEN ];
// array to hold reference fm input samples
double insamp_fm[ BUFFER_LEN_FM ];

// variables for bearing and update counts
static int g_nBrg = 0;
static int g_nCount = 0;

struct brgList{
    int nBrg;
    struct brgList *next;
};

static struct brgList * g_Head = NULL;
static struct brgList * g_Curr = NULL;
static struct brgList * g_Node = NULL;

// Initialize the variables
void initVars(int _nTopPks[][MAX_PEAKS], int _nBtmPks[][MAX_PEAKS], int _nRefIdx, int _nVarIdx)
{
    int i = 0,
    nInitVal = -1;
    double dInitVal = -1.0;

    for(i = 0; i < MAX_PEAKS; i++)
// Initialize the bearing linked list
void initBrgNodes()
{
    int i = 0;

    for (; i < MAX_BRG_NODES; i++)
    {
        g_Node = (struct brgList *)malloc(sizeof(struct brgList));
        g_Node->nBrg = 0;
        g_Node->next = g_Head;
        g_Head = g_Node;
        g_Curr = g_Node;
    }

    g_Node = g_Head;
}

// Delete the bearing linked list
void deleteBrgNodes()
{
    int i = 0;

    g_Node = g_Head;

    for (; i < MAX_BRG_NODES; i++)
    {
        free(g_Node);
    }
}

// Figure out the direction of travel (up or down) based on a set of data points
// Store the top and bottom peaks based on the conditions above
void checkDirection(double *dData, int _nDataLen, int *nTopPks, int *nBtmPks)
{
    const int UP = 1, DOWN = 0;

    int bDirection = DOWN;
    int nUpCnt = 0,
        nDnCnt = 0,
        nTopIdx = 0,
nBtmIdx = 0;
int nInitTurnId[2] = {0, 0};
int i = 0;

for (i = 0; i < _nDataLen - 1; i++)
{
    // If the next point is lower than the current one (direction is down)
    if(_dData[i+1] < _dData[i])
    {
        // Count the number of downs
        nDnCnt++;
        // Store the first down count
        if(nDnCnt <= 1)
        {
            nInitTurnId[DOWN] = i+1;
        }
    }

    // If the sequential number of downs are more than the FILTER_AMOUNT
    if(nDnCnt > FILTER_AMOUNT && bDirection == UP)
    {
        // Print a message to the user
        __android_log_print(ANDROID_LOG_INFO, "rtl_fm",
"DOWN: Index: %d, Value %lf, UP Direction Points: %d\n", nInitTurnId[DOWN],
dInitTurnVal[DOWN], nUpCnt);

        // Set the direction as Down (indicating the sine wave switched direction)
        bDirection = DOWN;
        _nTopPks[nTopIdx] = nInitTurnId[DOWN];

#if DEBUG_PRINT
        __android_log_print(ANDROID_LOG_INFO, "rtl_fm",
"Top Peak: %d\n", _nTopPks[nTopIdx]);
#endif

        nTopIdx++;

        // Reset the up count to 0
        nUpCnt = 0;
    }
    else
    {
        // Count the number of ups
        nUpCnt++;
    }
}
if(nUpCnt <= 1)
{
    nInitTurnId[UP] = i+1;
}

// If the sequential number of ups are more than the
FILTER_AMOUNT
if(nUpCnt > FILTER_AMOUNT && bDirection == DOWN)
{
    // Print a message to the user
    //__android_log_print(ANDROID_LOG_INFO, "rtl_fm",
    "UP: Index: %d, Value %lf, Down Direction Points: %d\n", nInitTurnId[UP],
    dInitTurnVal[UP], nDnCnt);
    // Set the direction as Up (indicating the sine wave switched
direction)
    bDirection = UP;
    _nBtmPks[nBtmIdx] = nInitTurnId[UP];

    #if DEBUG_PRINT
    __android_log_print(ANDROID_LOG_INFO, "rtl_fm",
    "Bottom Peak: %d\n", _nBtmPks[nBtmIdx]);
    #endif

    nBtmIdx++;
    // Reset the down count to 0
    nDnCnt = 0;
}

#if DEBUG_PRINT
__android_log_print(ANDROID_LOG_INFO, "rtl_fm", "Up Count: %d, Down
Count: %d\n", nUpCnt, nDnCnt);
#endif

// Computes the bearing by considering all cases (cases that go from 359 to 0 degrees as
well)
// Uses a moving average sum of the last 30 points to calculate a new bearing
int computeBrg(double _dTopBrg, double _dBtmBrg)
{
    static unsigned int unCnt = 1; // Start from 1, don't want a divide by zero
    condition
    static int nBrg[BRG_FLTR_SIZE];

    int i = 0, nBrgAvg = 0, nSum = 0;
static int nDoOnce = 0;

// Clear the array the first time
if (unCnt <= 1)
{
    for (i = 0; i < BRG_FLTR_SIZE; i++)
        nBrg[i] = 0;
}

// Find out if there is a difference greater than 180
// Example: Looking for the 350 deg and 10 deg
if ((_dTopBrg - _dBtmBrg) > 180.0)
{
    _dBtmBrg += 360;
}
else if ((_dBtmBrg - _dTopBrg) > 180.0)
{
    _dTopBrg += 360;
}

unCnt = unCnt % BRG_FLTR_SIZE;

// Get an average of the bottom and top bearing and modulate the value by 360
nBrg[unCnt] = (unsigned int)((_dTopBrg + _dBtmBrg) / 2) % 360;

// Checking for conditions of 350+ jumping to 0+
if(unCnt > 2)
{
    if ((nBrg[unCnt] - nBrg[unCnt - 1]) > 180 && nBrg[unCnt - 1] < 180)
    {
        nBrg[unCnt - 1] += 360;
    }
    else if ((nBrg[unCnt - 1] - nBrg[unCnt]) > 180 && nBrg[unCnt] < 180)
    {
        nBrg[unCnt] += 360;
    }
}

// This is the part that makes it take the average once we have all the values up to
BRG_FLTR_SIZE
if (unCnt == 0)
{
    for (i = 0; i < BRG_FLTR_SIZE; i++)
    {
        nBrgAvg += nBrg[i];
    }
nBrgAvg = (nBrgAvg / BRG_FLTR_SIZE) % 360;

if (nDoOnce == 0)
{
    // Use the first bearing to fill the global bearing linked list
    for (i = 0; i < MAX_BRG_NODES; i++)
    {
        g_Node->nBrg = nBrgAvg;
        g_Node = g_Node->next;
    }
    g_Node = g_Head;

    nDoOnce = 1;
}
else
{
    // Go to the next element in the linked list
    if (g_Node->next != NULL)
    {
        g_Node = g_Node->next;
    }
    else // Otherwise go back to the head node
    {
        g_Node = g_Head;
    }

    // Store the average bearing from this calculation
    if ((g_Node->nBrg - nBrgAvg) > 180)
    {
        // If the difference between these values are greater than
        180, make an adjustment
        g_Node->nBrg = nBrgAvg + 360;
    }
    else
    {
        // Otherwise just store the value
        g_Node->nBrg = nBrgAvg;
    }
}

    // Average out the bearing
    for (i = 0; i < MAX_BRG_NODES; i++)
    {
        nSum += g_Curr->nBrg;
        if(g_Curr->next != NULL)
            g_Curr = g_Curr->next;
    }
// Point it back to the first node
g_Curr = g_Head;

    g_nBrg = (nSum / MAX_BRG_NODES) % 360;
}

__android_log_print(ANDROID_LOG_INFO, "rtl_fm", "Brg (Inst): %d\n", nBrgAvg, g_nBrg);
    g_nCount++;
}

unCnt++;

    return nBrgAvg;
}

// Get the bearing by subtracting the top and bottom peaks of the variable and reference signals
// Then call the compute bearing function to take care of all scenarios to give a smoothed average bearing
void getBearing(int _nTopPks[][MAX_PEAKS], int _nBtmPks[][MAX_PEAKS], int _nRefIdx, int _nVarIdx)
{
    int i = 1;
        double dTopBearing = 0.0,
    dBtmBearing = 0.0;

for(i = 1; i < MAX_PEAKS; i++)
{
    // Skip the first iteration, since we need to have a peak difference between the two signals

    // If any one of the values are set to -1, ignore the rest
    if(_nTopPks[_nVarIdx][i] < 0 || _nTopPks[_nRefIdx][i] < 0
    || _nBtmPks[_nVarIdx][i] < 0 || _nBtmPks[_nRefIdx][i] < 0 )
    {
        break;
    }

    // If the variable top peak is ahead of the reference top peak
    else if(_nTopPks[_nVarIdx][i] > _nTopPks[_nRefIdx][i] &&
    _nBtmPks[_nVarIdx][i] > _nBtmPks[_nRefIdx][i])
    {
        dTopBearing = _nTopPks[_nVarIdx][i] - _nTopPks[_nRefIdx][i];
    dBtmBearing = _nBtmPks[_nVarIdx][i] - _nBtmPks[_nRefIdx][i];
}
// Check the next variable
else if (_nTopPks[_nVarIdx][i + 1] > _nTopPks[_nRefIdx][i] &&
    _nBtmPks[_nVarIdx][i + 1] > _nBtmPks[_nRefIdx][i])
{
    dTopBearing = _nTopPks[_nVarIdx][i+1] -
    _nTopPks[_nRefIdx][i];
    dBtmBearing = _nBtmPks[_nVarIdx][i+1] -
    _nBtmPks[_nRefIdx][i];
}
else
{
    if (_nTopPks[_nVarIdx][i + 1] < 0 || _nBtmPks[_nVarIdx][i + 1] <
0)
    {
        break;
    }

    // Assume a 359 to 0 degrees angle
    dTopBearing = dBtmBearing = 359.0;
}

computeBrg(dTopBearing, dBtmBearing);

// Initialize all variables
initVars(_nTopPks, _nBtmPks, 0, 1);

//---------------------------INTERPRET SIGNAL END--------------------------------------

// the FIR filter function, used for filtering Morse Code
void firFloatMorse( double *coeffs, double *input, double *output,
    int length, int filterLength )
{
    double acc;     // accumulator for MACs
    double *coeffp; // pointer to coefficients
    double *inputp; // pointer to input samples
    int n;
    int k;

    // put the new samples at the high end of the buffer
    memcpy( &insamp_morse[filterLength - 1], input,
        length * sizeof(double) );

    // apply the filter to each input sample
    for ( n = 0; n < length; n++ )
    {
// calculate output n
coefp = coeffs;
inputp = &insamp_morse[filterLength - 1 + n];
acc = 0;
for ( k = 0; k < filterLength; k++ ) {
    acc += (*coefp++) * (*inputp--);
}
output[n] = acc;

// shift input samples back in time for next time
memmove( &insamp_morse[0], &insamp_morse[length],
    (filterLength - 1) * sizeof(double) );

// the FIR filter function, used for filtering the reference signal
void firFloatREF( double *coeffs, double *input, double *output,
    int length, int filterLength )
{
    double acc;     // accumulator for MACs
    double *coefp; // pointer to coefficients
    double *inputp; // pointer to input samples
    int n;
    int k;

    // put the new samples at the high end of the buffer
    memcpy( &insamp_ref[filterLength - 1], input,
        length * sizeof(double) );

    // apply the filter to each input sample
    for ( n = 0; n < length; n++ ) {
        // calculate output n
        coefp = coeffs;
        inputp = &insamp_ref[filterLength - 1 + n];
        acc = 0;
        for ( k = 0; k < filterLength; k++ ) {
            acc += (*coefp++) * (*inputp--);
        }
        output[n] = acc;
    }

    // shift input samples back in time for next time
    memmove( &insamp_ref[0], &insamp_ref[length],
        (filterLength - 1) * sizeof(double) );
// the FIR filter function, used for filtering the variable signal
void firFloatVAR( double *coeffs, double *input, double *output,
   int length, int filterLength )
{
    double acc;   // accumulator for MACs
    double *coeffp;   // pointer to coefficients
    double *inputp;   // pointer to input samples
    int n;
    int k;

    // put the new samples at the high end of the buffer
    memcpy( &insamp_var[filterLength - 1], input,
        length * sizeof(double) );

    // apply the filter to each input sample
    for ( n = 0; n < length; n++ ) {
        // calculate output n
        coeffp = coeffs;
        inputp = &insamp_var[filterLength - 1 + n];
        acc = 0;
        for ( k = 0; k < filterLength; k++ ) {
            acc += (*coeffp++) * (*inputp--);
        }
        output[n] = acc;
    }

    // shift input samples back in time for next time
    memmove( &insamp_var[0], &insamp_var[length],
        (filterLength - 1) * sizeof(double) );
}

// the FIR filter function, used for filtering the reference signal during FM
void firFloatFM( double *coeffs, double *input, double *output,
   int length, int filterLength )
{
    double acc;   // accumulator for MACs
    double *coeffp;   // pointer to coefficients
    double *inputp;   // pointer to input samples
    int n;
    int k;

    // put the new samples at the high end of the buffer
    memcpy( &insamp_fm[filterLength - 1], input,
length * sizeof(double));

// apply the filter to each input sample
for ( n = 0; n < length; n++ ) {
    // calculate output n
    coeffp = coeffs;
    inputp = &insamp_fm[filterLength - 1 + n];
    acc = 0;
    for ( k = 0; k < filterLength; k++ ) {
        acc += (*coeffp++) * (*inputp--);
    }
    output[n] = acc;
}

// shift input samples back in time for next time
memmove( &insamp_fm[0], &insamp_fm[length], (filterLength - 1) * sizeof(double));

// Convert int16's to doubles
void int16ToDouble( int16_t *input, double *output, int length )
{
    int i;
    for ( i = 0; i < length; i++ ) {
        output[i] = (double)input[i];
    }
}

// Convert double's to int16's
void doubleToInt16( double *input, int16_t *output, int length )
{
    int i;
    for ( i = 0; i < length; i++ ) {
        if ( input[i] > 32767.0 ) {
            input[i] = 32767.0;
        } else if ( input[i] < -32768.0 ) {
            input[i] = -32768.0;
        }
        // convert
        output[i] = (int16_t)input[i];
    }
}
// Multiply the variables
void multiply(int ar, int aj, int br, int bj, int *cr, int *cj)
{
    *cr = ar*br - aj*bj;
    *cj = aj*br + ar*bj;
}

// Get the polar discriminant
int polar_discriminant(int ar, int aj, int br, int bj)
{
    int cr, cj;
    double angle;
    multiply(ar, aj, br, -bj, &cr, &cj);
    angle = atan2((double)cj, (double)cr);
    return (int)(angle / 3.14159 * (1<<14));
}

// Version of FM Demodulation
void my_fm_demod(struct ref_signal *sRef)
{
    int i;
    int16_t *lp = sRef->sig16;

    sRef->sig16_fm[0] = (int16_t)polar_discriminant(lp[0],
        lp[1],
        sRef->pre_r,
        sRef->pre_j);

    for (i = 2; i < (sRef->sig16_len); i += 2)
    {
        sRef->sig16_fm[i/2] = (int16_t)polar_discriminant(lp[i],
            lp[i+1],
            lp[i-2],
            lp[i-1]);
    }

    sRef->pre_r = lp[sRef->sig16_len - 2];
    sRef->pre_j = lp[sRef->sig16_len - 1];
    sRef->sig16_fm_len = sRef->sig16_len/2;
void getMorse(int16_t *_sig16, int _sig16_len)
{
    static int nDoOnce = 0;
    int16_t output[MAX_INPUT_LEN];
    double dIn[MAX_INPUT_LEN];
    double dOut[MAX_INPUT_LEN];
    int i = 0;

    if(nDoOnce == 0)
    {
        // Clear the array
        memset( insamp_morse, 0, sizeof( insamp_morse ) );
        // Only do this once
        nDoOnce = 1;
    }
    else
    {
        // convert to doubles
        int16ToDouble( _sig16, dIn, _sig16_len );

        for(i = 0; i < _sig16_len; i++)
        {
            dIn[i] *= 20;
        }

        // perform the filtering
        firFloatMorse( coeff_morse, dIn, dOut, _sig16_len, FILTER_LEN );
        // convert to ints
        doubleToInt16( dOut, output, _sig16_len );

        // play audio
        selectClipBuf(output, _sig16_len);
    }
}

// Main function that initializes everything and calls the functions in steps
// to correctly decode the VOR signals and get an actual bearing, pass the bearing using
// pass by reference
void decodeBrg(int _preJ, int _preR, int16_t *sig16, int _sig16_len, int* _brg, int* _count)
{
    static int nDoOnce = 0;
}
int i = 0;
int16_t output[MAX_INPUT_LEN];
double dIn[MAX_INPUT_LEN],
  dRefOut[MAX_INPUT_LEN],
  dFmIn[MAX_INPUT_LEN/2],
  dFmOut[MAX_INPUT_LEN/2],
  dVarOut[MAX_INPUT_LEN];
static struct ref_signal sRef;

// Interpreting Signal: Start

static double dBufVar[NUM_OF_SAMPLES+MAX_INPUT_LEN],
  dBufRef[NUM_OF_SAMPLES+MAX_INPUT_LEN];
static int nBufIdx = 0;
const int REF = 0, VAR = 1;

int nTopPks[2][MAX_PEAKS],
  nBtmPks[2][MAX_PEAKS];

// Interpreting Signal: End

if(nDoOnce == 0)
{
  // Interpreting Signal: Start
  initBrgNodes();
  initVars(nTopPks, nBtmPks, REF, VAR);
  // Interpreting Signal: End

  // Clear the array
  memset( insamp_var, 0, sizeof( insamp_var ) );
  memset( insamp_ref, 0, sizeof( insamp_ref ) );
  memset( insamp_fm, 0, sizeof( insamp_fm ) );
  // Only do this once
  nDoOnce = 1;

  sRef.pre_j = _preJ;
  sRef.pre_r = _preR;
}
else
{
  // convert int16 to doubles
  int16ToDouble( _sig16, dIn, _sig16_len );

  // Filter the variable signal
  firFloatVAR( coeff_lp30, dIn, dVarOut, _sig16_len, FILTER_LEN );
// Filter the reference signal
firFloatREF( coeff_bp9960, dIn, dRefOut, _sig16_len, FILTER_LEN );

// Convert doubles to int16's
doubleToInt16( dRefOut, sRef.sig16, _sig16_len );
sRef.sig16_len = _sig16_len;

// FM demodulate the reference portion
my_fm_demod(&sRef);

// Convert int16 to double
int16ToDouble(sRef.sig16_fm, dFmIn, sRef.sig16_fm_len);

firFloatFM( coeff_lp30_95, dFmIn, dFmOut, sRef.sig16_fm_len, FILTER_LEN_DIV_2 );

for(i = 0; i < sRef.sig16_fm_len; i++)
{
    // Buffer the Reference and Variable Data
dBufRef[nBufIdx] = dFmOut[i];
dBufVar[nBufIdx] = dVarOut[i*2]; // Use every other approach (forms smoother variable signal)
nBufIdx++;
}

// Interpret Signal: Start

if(nBufIdx >= NUM_OF_SAMPLES)
{
    // Stores the top and bottom peaks and their associated differences
    checkDirection(dBufVar, nBufIdx, nTopPks[VAR],
    nBtmPks[VAR]);
    checkDirection(dBufRef, nBufIdx, nTopPks[REF],
    nBtmPks[REF]);

    // Gets the bearing
    getBearing(nTopPks, nBtmPks, REF, VAR);

    *_count = g_nCount;
    *_brg = g_nBrg;

    // Clear the buffer and index
    nBufIdx = 0;
}

// Interpret Signal: End
rtl_fm.c

// This file has been written by Kyle Keen, it has been slightly modified to inject the
// necessary portions of the code to tap in to the RTL-SDR. It has been pasted here for
// reference.

/*
 * rtl-sdr, turns your Realtek RTL2832 based DVB dongle into a SDR receiver
 * Copyright (C) 2012 by Steve Markgraf <steve@steve-m.de>
 * Copyright (C) 2012 by Hoernchen <la@tfc-server.de>
 * Copyright (C) 2012 by Kyle Keen <keenerd@gmail.com>
 * Copyright (C) 2013 by Elias Oenal <EliasOenal@gmail.com>
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 2 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 */

/*
 * written because people could not do real time
 * FM demod on Atom hardware with GNU radio
 * based on rtl_sdr.c and rtl_tcp.c
 * todo: realtime ARMv5
 *     remove float math (disqualifies complex.h)
 *     in-place array operations
 *     sanity checks
 *     nicer FIR than square
 *     scale squelch to other input parameters
 *     test all the demodulations
 *     pad output on hop
 *     nearest gain approx
 *     frequency ranges could be stored better
 */

#include <errno.h>
#include <signal.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
// for __android_log_print(ANDROID_LOG_INFO, "YourApp", "formatted message");
#include <android/log.h>

#ifndef _WIN32
#include <unistd.h>
#else
#include <Windows.h>
#include <fcntl.h>
#include <io.h>
#include "getopt/getopt.h"
#define usleep(x) Sleep(x/1000)
#define round(x) (x > 0.0 ? floor(x + 0.5): ceil(x - 0.5))
#endif

#include <pthread.h>
#include <libusb.h>

#include "rtl-sdr.h"
#include "native-audio-jni.h"
#include "rtl_vor_decoder.h"

#define DEFAULT_SAMPLE_RATE 24000
#define DEFAULT_ASYNC_BUF_NUMBER 32
#define DEFAULT_BUF_LENGTH (1 * 16384)
#define MAXIMUM_OVERSAMPLE 16
#define MAXIMUM_BUF_LENGTH (MAXIMUM_OVERSAMPLE * DEFAULT_BUF_LENGTH)
#define AUTO_GAIN -100

static pthread_t demod_thread;
static pthread_mutex_t data_ready; /* locked when no fresh data available */
static pthread_mutex_t data_write; /* locked when r/w buffer */
static int do_exit = 0;
static rtlsdr_dev_t *dev = NULL;
static int lcm_post[17] = {1,1,1,3,1,5,3,7,1,9,5,11,3,13,7,15,1};

static int *atan_lut = NULL;
static int atan_lut_size = 131072; /* 512 KB */
static int atan_lut_coef = 8;

static int g_nCount;
static int g_nBrg;

struct fm_state
{
    int now_r, now_j;
    int pre_r, pre_j;
    int prev_index;
    int downsample; /* min 1, max 256 */
    int post_downsample;
    int output_scale;
    int squelch_level, conseq_squelch, squelch_hits, terminate_on_squelch;
    int exit_flag;
    uint8_t buf[MAXIMUM_BUF_LENGTH];
    uint32_t buf_len;
    int signal[MAXIMUM_BUF_LENGTH]; /* 16 bit signed i/q pairs */
    int16_t signal2[MAXIMUM_BUF_LENGTH]; /* signal has low-pass, signal2 has demod */
    int signal_len;
    int signal2_len;
    FILE *file;
    int edge;
    uint32_t freqs[1000];
    int freq_len;
    int freq_now;
    uint32_t sample_rate;
    int output_rate;
    int fir_enable;
    int fir[256]; /* fir_len == downsample */
    int fir_sum;
    int custom_atan;
    int deemph, deemph_a;
    int now_lpr;
    int prev_lpr_index;
    int dc_block, dc_avg;
    void (*mode_demod)(struct fm_state*);
};

void usage(void)
{
    __android_log_print(ANDROID_LOG_INFO, "rtl_fm",
            "rtl_fm, a simple narrow band FM demodulator for RTL2832 based DVB-
T receivers\n\n" "Use:\rtl_fm -f freq [-options] [filename]\n" "t-f frequency_to_tune_to [Hz]\n" "t (use multiple -f for scanning, requires squelch)\n" "t (ranges supported, -f 118M:137M:25k)\n"
"\t[-s sample_rate (default: 24k)]\n" "\t[-d device_index (default: 0)]\n" "\t[-g tuner_gain (default: automatic)]\n" "\t[-l squelch_level (default: 0/off)]\n" "\t[-o oversampling (default: 1, 4 recommended)]\n" "\t[-p ppm_error (default: 0)]\n" "\t[-E sets lower edge tuning (default: center)]\n" "\t[-N enables NBFM mode (default: on)]\n" "\t[-W enables WBFM mode (default: off)]\n" "\t [-N -s 170k -o 4 -A fast -r 32k -l 0 -D]\n" "\t(filename (a -' dumps samples to stdout)\n" "\t (omitting the filename also uses stdout)\n"

"Experimental options:\n" "\t[-r output_rate (default: same as -s)]\n" "\t[-t squelch_delay (default: 20)]\n" "\t (+values will mute(scan, -values will exit)]\n" "\t[-M enables AM mode (default: off)]\n" "\t[-L enables LSB mode (default: off)]\n" "\t[-U enables USB mode (default: off)]\n" "\t[-D enables DSB mode (default: off)]\n" "\t [-R enables raw mode (default: off, 2x16 bit output)]\n" "\t[-F enables high quality FIR (default: off/square)]\n" "\t[-D enables de-emphasis (default: off)]\n" "\t[-C enables DC blocking of output (default: off)]\n" "\t[-A std/fast/lut choose atan math (default: std)]\n\n"

"Produces signed 16 bit ints, use Sox or aplay to hear them.\n" "\t trtl_fm ... - | play -t raw -r 24k -e signed-integer -b 16 -c 1 -V1 -\n" "\t | aplay -r 24k -f S16_LE -t raw -c 1\n" "\t -s 22.5k - | multimon -t raw /dev/stdin\n"

exit(1);
"

#endif

BOOL WINAPI sighandler(int signum)
{
    if (CTRL_C_EVENT == signum) {
        __android_log_print(ANDROID_LOG_WARN, "rtl_fm", "Signal caught,
exitin!\n");
        do_exit = 1;
        rtlsdr_cancel_async(dev);
        return TRUE;
    }
    return FALSE;
}

#ifdef _WIN32

#endif

203
static void sighandler(int signum)
{
    __android_log_print(ANDROID_LOG_INFO, "rtl_fm", "Signal caught, exiting!
    do_exit = 1;
    rtlsdr_cancel_async(dev);
}
#endif

void rotate_90(unsigned char *buf, uint32_t len)
/* 90 rotation is 1+0j, 0+1j, -1+0j, 0-1j
   or [0, 1, -3, 2, -4, -5, 7, -6] */
{
    uint32_t i;
    unsigned char tmp;
    for (i=0; i<len; i+=8) {
        /* uint8_t negation = 255 - x */
        tmp = 255 - buf[i+3];
        buf[i+3] = buf[i+2];
        buf[i+2] = tmp;
        buf[i+4] = 255 - buf[i+4];
        buf[i+5] = 255 - buf[i+5];
        tmp = 255 - buf[i+6];
        buf[i+6] = buf[i+7];
        buf[i+7] = tmp;
    }
}

void low_pass(struct fm_state *fm, unsigned char *buf, uint32_t len)
/* simple square window FIR */
{
    int i=0, i2=0;
    while (i < (int)len) {
        fm->now_r += ((int)buf[i] - 128);
        fm->now_j += ((int)buf[i+1] - 128);
        i += 2;
        fm->prev_index++;
        if (fm->prev_index < fm->downsample) {
            continue;
        }
        fm->signal[i2] = fm->now_r * fm->output_scale;
        fm->signal[i2+1] = fm->now_j * fm->output_scale;
        fm->prev_index = 0;
        fm->now_r = 0;
    
    }
}
void build_fir(struct fm_state *fm)
/* for now, a simple triangle */
/* fancy FIRs are equally expensive, so use one */
/* point = sum(sample[i] * fir[i] * fir_len / fir_sum) */
{
    int i, len;
    len = fm->downsample;
    for(i = 0; i < (len/2); i++) {
        fm->fir[i] = i;
    }
    for(i = len-1; i >= (len/2); i--) {
        fm->fir[i] = len - i;
    }
    fm->fir_sum = 0;
    for(i = 0; i < len; i++) {
        fm->fir_sum += fm->fir[i];
    }
}

void low_pass_fir(struct fm_state *fm, unsigned char *buf, uint32_t len)
/* perform an arbitrary FIR, doubles CPU use */
/* possibly bugged, or overflowing */
{
    int i=0, i2=0, i3=0;
    while (i < (int)len) {
        i3 = fm->prev_index;
        fm->now_r += ((int)buf[i] - 128) * fm->fir[i3] * fm->downsample / fm->fir_sum;
        fm->now_j += ((int)buf[i+1] - 128) * fm->fir[i3] * fm->downsample / fm->fir_sum;
        i += 2;
        fm->prev_index++;
        if (fm->prev_index < fm->downsample) {
            continue;
        }
        fm->signal[i2] = fm->now_r * fm->output_scale;
        fm->signal[i2+1] = fm->now_j * fm->output_scale;
        fm->prev_index = 0;
        fm->now_r = 0;
        fm->now_j = 0;
    }
}
i2 += 2;
}
fm->signal_len = i2;
}

int low_pass_simple(int16_t *signal2, int len, int step)
// no wrap around, length must be multiple of step
{
    int i, i2, sum;
    for(i=0; i < len; i+=step) {
        sum = 0;
        for(i2=0; i2<step; i2++) {
            sum += (int)signal2[i + i2];
        }
        //signal2[i/step] = (int16_t)(sum / step);
        signal2[i/step] = (int16_t)(sum);
    }
    signal2[i/step + 1] = signal2[i/step];
    return len / step;
}

void low_pass_real(struct fm_state *fm)
/* simple square window FIR */
// add support for upsampling?
{
    int i=0, i2=0;
    int fast = (int)fm->sample_rate / fm->post_downsample;
    int slow = fm->output_rate;
    while (i < fm->signal2_len) {
        fm->now_lpr += fm->signal2[i];
        i++;
        fm->prev_lpr_index += slow;
        if (fm->prev_lpr_index < fast) {
            continue;
        }
        fm->signal2[i2] = (int16_t)(fm->now_lpr / (fast/slow));
        fm->prev_lpr_index -= fast;
        fm->now_lpr = 0;
        i2 += 1;
    }
    fm->signal2_len = i2;
}

/* define our own complex math ops
   because ARMv5 has no hardware float */
void multiply(int ar, int aj, int br, int bj, int *cr, int *cj)
{
    *cr = ar*br - aj*bj;
    *cj = aj*br + ar*bj;
}

int polar_discriminant(int ar, int aj, int br, int bj)
{
    int cr, cj;
    double angle;
    multiply(ar, aj, br, -bj, &cr, &cj);
    angle = atan2((double)cj, (double)cr);
    return (int)(angle / 3.14159 * (1<<14));
}

int fast_atan2(int y, int x)
/* pre scaled for int16 */
{
    int yabs, angle;
    int pi4=(1<<12), pi34=3*(1<<12);  // note pi = 1<<14
    if (x==0 && y==0) {
        return 0;
    }
    yabs = y;
    if (yabs < 0) {
        yabs = -yabs;
    }
    if (x >= 0) {
        angle = pi4 - pi4 * (x-yabs) / (x+yabs);
    } else {
        angle = pi34 - pi4 * (x+yabs) / (yabs-x);
    }
    if (y < 0) {
        return -angle;
    }
    return angle;
}

int polar_disc_fast(int ar, int aj, int br, int bj)
{
    int cr, cj;
    multiply(ar, aj, br, -bj, &cr, &cj);
    return fast_atan2(cj, cr);
}

int atan_lut_init()
{
  int i = 0;

  atan_lut = malloc(atan_lut_size * sizeof(int));

  for (i = 0; i < atan_lut_size; i++) {
    atan_lut[i] = (int) (atan((double) i / (1<<atan_lut_coef)) / 3.14159 * (1<<14));
  }

  return 0;
}

int polar_disc_lut(int ar, int aj, int br, int bj)
{
  int cr, cj, x, x_abs;

  multiply(ar, aj, br, -bj, &cr, &cj);

  /* special cases */
  if (cr == 0 || cj == 0) {
    if (cr == 0 && cj == 0) {return 0;}
    if (cr == 0 && cj > 0) {return 1<<13;}
    if (cr == 0 && cj < 0) {return -(1<<13);}
    if (cj == 0 && cr > 0) {return 0;}
    if (cj == 0 && cr < 0) {return 1<<14;}
  }

  /* real range -32768 - 32768 use 64x range -> absolute maximum: 2097152 */
  x = (cj << atan_lut_coef) / cr;
  x_abs = abs(x);

  if (x_abs >= atan_lut_size) {
    /* we can use linear range, but it is not necessary */
    return (cj > 0) ? 1<<13 : -1<<13;
  }

  if (x > 0) {
    return (cj > 0) ? atan_lut[x] : atan_lut[x] - (1<<14);
  } else {
    return (cj > 0) ? (1<<14) - atan_lut[-x] : -atan_lut[-x];
}
return 0;
}

void fm_demod(struct fm_state *fm)
{
    int i, pcm;
    pcm = polar_discriminant(fm->signal[0], fm->signal[1],
                              fm->pre_r, fm->pre_j);
    fm->signal2[0] = (int16_t)pcm;
    for (i = 2; i < (fm->signal_len); i += 2) {
        switch (fm->custom_atan) {
            case 0:
                pcm = polar_discriminant(fm->signal[i], fm->signal[i+1],
                                          fm->signal[i-2], fm->signal[i-1]);
                break;
            case 1:
                pcm = polar_disc_fast(fm->signal[i], fm->signal[i+1],
                                       fm->signal[i-2], fm->signal[i-1]);
                break;
            case 2:
                pcm = polar_disc_lut(fm->signal[i], fm->signal[i+1],
                                      fm->signal[i-2], fm->signal[i-1]);
                break;
        }
        fm->signal2[i/2] = (int16_t)pcm;
    }
    fm->pre_r = fm->signal[fm->signal_len - 2];
    fm->pre_j = fm->signal[fm->signal_len - 1];
    fm->signal2_len = fm->signal_len/2;
}

void am_demod(struct fm_state *fm)
/* todo, fix this extreme laziness */
{
    int i, pcm;
    for (i = 0; i < (fm->signal_len); i += 2) {
        // hypot uses floats but won't overflow
        //fm->signal2[i/2] = (int16_t)hypot(fm->signal[i], fm->signal[i+1]);
        pcm = fm->signal[i] * fm->signal[i];
        pcm += fm->signal[i+1] * fm->signal[i+1];
        fm->signal2[i/2] = (int16_t)sqrt(pcm);// * fm->output_scale;
    }
    fm->signal2_len = fm->signal_len/2;
    // low-pass? (3khz) highpass? (dc)
void usb_demod(struct fm_state *fm)
{
    int i, pcm;
    for (i = 0; i < (fm->signal_len); i += 2) {
        pcm = fm->signal[i] + fm->signal[i+1];
        fm->signal2[i/2] = (int16_t)pcm; // * fm->output_scale;
    }
    fm->signal2_len = fm->signal_len/2;
}

void lsb_demod(struct fm_state *fm)
{
    int i, pcm;
    for (i = 0; i < (fm->signal_len); i += 2) {
        pcm = fm->signal[i] - fm->signal[i+1];
        fm->signal2[i/2] = (int16_t)pcm; // * fm->output_scale;
    }
    fm->signal2_len = fm->signal_len/2;
}

void raw_demod(struct fm_state *fm)
{
    /* hacky and pointless code */
    int i;
    for (i = 0; i < (fm->signal_len); i++) {
        fm->signal2[i] = (int16_t)fm->signal[i];
    }
    fm->signal2_len = fm->signal_len;
}

void deemph_filter(struct fm_state *fm)
{
    static int avg; // cheating...
    int i, d;
    // de-emph IIR
    // avg = avg * (1 - alpha) + sample * alpha;
    for (i = 0; i < fm->signal2_len; i++) {
        d = fm->signal2[i] - avg;
        if (d > 0) {
            avg += (d + fm->deemph_a/2) / fm->deemph_a;
        } else {
            avg += (d - fm->deemph_a/2) / fm->deemph_a;
        }
    fm->signal2[i] = (int16_t)avg;
}
void dc_block_filter(struct fm_state *fm) {
    int i, avg;
    int64_t sum = 0;
    for (i = 0; i < fm->signal2_len; i++) {
        sum += fm->signal2[i];
    }
    avg = sum / fm->signal2_len;
    avg = (avg + fm->dc_avg * 9) / 10;
    for (i = 0; i < fm->signal2_len; i++) {
        fm->signal2[i] = avg;
    }
    fm->dc_avg = avg;
}

int mad(int *samples, int len, int step) /* mean average deviation */
{
    int i = 0, sum = 0, ave = 0;
    if (len == 0)
        {return 0;}
    for (i = 0; i < len; i += step) {
        sum += samples[i];
    }
    ave = sum / (len * step);
    sum = 0;
    for (i = 0; i < len; i += step) {
        sum += abs(samples[i] - ave);
    }
    return sum / (len / step);
}

int post_squelch(struct fm_state *fm) /* returns 1 for active signal, 0 for no signal */
{
    int dev_r, dev_j, len, sq_l;
    /* only for small samples, big samples need chunk processing */
    len = fm->signal_len;
    sq_l = fm->squelch_level;
    dev_r = mad(&(fm->signal[0]), len, 2);
    dev_j = mad(&(fm->signal[1]), len, 2);
    if ((dev_r > sq_l) || (dev_j > sq_l)) {
        fm->squelch_hits = 0;
    }
static void optimal_settings(struct fm_state *fm, int freq, int hopping)
{
    int r, capture_freq, capture_rate;
    fm->downsample = (1000000 / fm->sample_rate) + 1;
    fm->freq_now = freq;
    capture_rate = fm->downsample * fm->sample_rate;
    capture_freq = fm->freqs[freq] + capture_rate/4;
    capture_freq += fm->edge * fm->sample_rate / 2;
    fm->output_scale = (1<<15) / (128 * fm->downsample);
    if (fm->output_scale < 1) {
        fm->output_scale = 1;}
    if(fm->mode_demod == &fm_demod){
        fm->output_scale = 1;}
    __android_log_print(ANDROID_LOG_INFO, "rtl_fm", "Output Scale: %i.\n", fm->output_scale);
    /* Set the frequency */
    r = rtlsdr_set_center_freq(dev, (uint32_t)capture_freq);
    if (hopping) {
        return;
    }
    __android_log_print(ANDROID_LOG_INFO, "rtl_fm", "Oversampling input by: %ix.\n", fm->downsample);
    __android_log_print(ANDROID_LOG_INFO, "rtl_fm", "Oversampling output by: %ix.\n", fm->post_downsample);
    __android_log_print(ANDROID_LOG_INFO, "rtl_fm", "Buffer size: 1000 * 0.5 * 1cm_post[%f] * (float)DEFAULT_BUF_LENGTH / (float)capture_rate);
    if (r < 0) {
        __android_log_print(ANDROID_LOG_WARN, "rtl_fm", "WARNING: Failed to set center freq.\n");}
    else {
        __android_log_print(ANDROID_LOG_INFO, "rtl_fm", "Tuned to %u Hz.\n", capture_freq);
    }
    /* Set the sample rate */
    __android_log_print(ANDROID_LOG_INFO, "rtl_fm", "Sampling at %u Hz.\n", capture_rate);
    if (fm->output_rate > 0) {
        __android_log_print(ANDROID_LOG_INFO, "rtl_fm", "Output at %u Hz.\n", fm->output_rate);
    }
} else {
    __android_log_print(ANDROID_LOG_INFO, "rtl_fm", "Output at %u
Hz.\n", fm->sample_rate/fm->post_downsample);
    r = rtlsdr_set_sample_rate(dev, (uint32_t)capture_rate);
    if (r < 0) {
        __android_log_print(ANDROID_LOG_WARN, "rtl_fm", "WARNING:
Failed to set sample rate.\n");
    }
}

void full_demod(struct fm_state *fm)
{
    int i, sr, freq_next, hop = 0;

    if(fm->buf_len == -1)
        return;
    rotate_90(fm->buf, fm->buf_len);
    if (fm->fir_enable) {
        low_pass_fir(fm, fm->buf, fm->buf_len);
    } else {
        low_pass(fm, fm->buf, fm->buf_len);
    }
    pthread_mutex_unlock(&data_write);

    fm->mode_demod(fm);

    if (fm->mode_demod == &raw_demod) {
        fwrite(fm->signal, 4, fm->signal_len, fm->file);
        return;
    }

    // Decode the bearing from the data
    decodeBrg(fm->pre_j, fm->pre_r, fm->signal2, fm->signal2_len, &g_nBrg,
              &g_nCount);

    sr = post_squelch(fm);
    if (!sr && fm->squelch_hits > fm->conseq_squelch) {
        if (fm->terminate_on_squelch) {
            fm->exit_flag = 1;
        }
        if (fm->freq_len == 1) { /* mute */
            for (i=0; i<fm->signal_len; i++) {
                fm->signal2[i] = 0;
            }
        } else {
            hop = 1;
        }
    }
}
if (fm->post_downsample > 1) {
    fm->signal2_len = low_pass_simple(fm->signal2, fm->signal2_len, fm->post_downsample);
} else if (fm->output_rate > 0) {
    low_pass_real(fm);
} else if (fm->deemph) {
    deemph_filter(fm);
} else if (fm->dc_block) {
    dc_block_filter(fm);

// Play Morse Code
selectClipBuf(fm->signal2, fm->signal2_len);

/* ignore under runs for now */
if (hop) {
    freq_next = (fm->freq_now + 1) % fm->freq_len;
    optimal_settings(fm, freq_next, 1);
    fm->squelch_hits = fm->conseq_squelch + 1; /* hair trigger */
    /* wait for settling and flush buffer */
    usleep(5000);
    rtlsdr_read_sync(dev, NULL, 4096, NULL);
}

static void rtlsdr_callback(unsigned char *buf, uint32_t len, void *ctx)
{
    struct fm_state *fm2 = ctx;
    int nRead = 0;

    if (do_exit) {
        return;
    } else if (!ctx) {
        return;
    }
    pthread_mutex_lock(&data_write);

    memcpy(fm2->buf, buf, len);
    fm2->buf_len = len;
    pthread_mutex_unlock(&data_write);

    /* single threaded uses 25% less CPU? */
    /* full_demod(fm2); */
}
static void *demod_thread_fn(void *arg)
{
    struct fm_state *fm2 = arg;
    while (!do_exit) {
        pthread_mutex_lock(&data_ready);
        full_demod(fm2);
        if (fm2->exit_flag) {
            do_exit = 1;
            rtlsdr_cancel_async(dev);
        }
    }
    return 0;
}

double atofs(char* f)
/* standard suffixes */
{
    char* chop;
    double suff = 1.0;
    chop = malloc((strlen(f)+1)*sizeof(char));
    strncpy(chop, f, strlen(f)-1);
    switch (f[strlen(f)-1]) {
        case 'G':
            suff *= 1e3;
            break;
        case 'M':
            suff *= 1e3;
            break;
        case 'k':
            suff *= 1e3;
            break;
        default:
            suff *= atof(chop);
            break;
    }
    free(chop);
    if (suff != 1.0) {
        return suff;
    }
    return atof(f);
}

void frequency_range(struct fm_state *fm, char *arg)
{
    char *start, *stop, *step;
    int i;
    start = arg;
    stop = strchr(start, ':') + 1;
    stop[-1] = '\0';
    step = strchr(stop, ':') + 1;
    step[-1] = '\0';
    for(i=(int)atofs(start); i<=(int)atofs(stop); i+=(int)atofs(step))
    {
    }
struct fm_state fm;
int main(int argc, char **argv, char * frq)
{
    #ifdef _WIN32
        struct sigaction sigact;
    #endif

    char *filename = NULL;
    int n_read, r, opt, wb_mode = 0;
    int i, gain = AUTO_GAIN; // tenths of a dB
    uint8_t *buffer;
    uint32_t dev_index = 0;
    int device_count;
    int ppm_error = 0;

    fm->freqs[fm->freq_len] = (uint32_t)i;
    fm->freq_len++;
}
stop[-1] = ':';
step[-1] = ':';
}

void fm_init(struct fm_state *fm)
{
    fm->freqs[0] = 100000000;
    fm->sample_rate = DEFAULT_SAMPLE_RATE;
    fm->squelch_level = 0;
    fm->conseq_squelch = 20;
    fm->terminate_on_squelch = 0;
    fm->squelch_hits = 0;
    fm->freq_len = 0;
    fm->edge = 0;
    fm->fir_enable = 0;
    fm->prev_index = 0;
    fm->post_downsample = 1; // once this works, default = 4
    fm->custom_atan = 0;
    fm->deemph = 0;
    fm->output_rate = -1; // flag for disabled
    fm->mode_demod = &fm_demod;
    fm->pre_j = fm->pre_r = fm->now_r = fm->now_j = 0;
    fm->prev_lpr_index = 0;
    fm->deemph_a = 0;
    fm->now_lpr = 0;
    fm->dc_block = 0;
    fm->dc_avg = 0;
}

    struct sigaction sigact;
char vendor[256], product[256], serial[256];
fм_init(&fm);
pthread_mutex_init(&data_ready, NULL);
pthread_mutex_init(&data_write, NULL);

g_nBrg=0;
g_nCount=0;

while ((opt = getopt(argc, argv, "d:f:g:s:b:l:o:t:p:EFA:NWMULRDC")) != -1) {
    switch (opt) {
        case 'd':
            dev_index = atoi(optarg);
            break;
        case 'f':
            if (strchr(optarg, ':'))
                {frequency_range(&fm, optarg);}
            else
                {
                    fm.freqs[fm.freq_len] = (uint32_t)atofs(frq);
                    fm.freq_len++;
                }
            break;
        case 'g':
            gain = (int)(atof(optarg) * 10);
            break;
        case 'l':
            fm.squelch_level = (int)atof(optarg);
            break;
        case 's':
            fm.sample_rate = (uint32_t)atofs(optarg);
            break;
        case 'r':
            fm.output_rate = (int)atofs(optarg);
            break;
        case 'o':
            fm.post_downsample = (int)atof(optarg);
            if (fm.post_downsample < 1 || fm.post_downsample > MAXIMUM_OVERSAMPLE) {
                __android_log_print(ANDROID_LOG_INFO, "rtl_fm", "Oversample must be between 1 and %in", MAXIMUM_OVERSAMPLE);
            } break;
        case 't':
            fm.conseq_squelch = (int)atof(optarg);
            if (fm.conseq_squelch < 0) {
                fm.conseq_squelch = -fm.conseq_squelch;
                fm.terminate_on_squelch = 1;
            }
    }
}
} 
break;
case 'p':
    ppm_error = atoi(optarg);
    break;
case 'E':
    fm.edge = 1;
    break;
case 'F':
    fm.fir_enable = 1;
    break;
case 'A':
    if (strcmp("std", optarg) == 0) {
        fm.custom_atan = 0;
    }
    if (strcmp("fast", optarg) == 0) {
        fm.custom_atan = 1;
    }
    if (strcmp("lut", optarg) == 0) {
        atan_lut_init();
        fm.custom_atan = 2;
    }
    break;
case 'D':
    fm.deemph = 1;
    break;
case 'C':
    fm.dc_block = 1;
    break;
case 'N':
    fm.mode_demod = &fm_demod;
    break;
case 'W':
    wb_mode = 1;
    fm.mode_demod = &fm_demod;
    fm.sample_rate = 170000;
    fm.output_rate = 32000;
    fm.custom_atan = 1;
    fm.post_downsample = 4;
    fm.deemph = 1;
    fm.squelch_level = 0;
    break;
case 'M':
    fm.mode_demod = &am_demod;
    break;
case 'U':
    fm.mode_demod = &usb_demod;
    break;
case 'L':

        fm.mode_demod = &lsb_demod;
        break;
      case 'R':
        fm.mode_demod = &raw_demod;
        break;
      default:
        usage();
        break;
    }
*/
/* quadruple sample_rate to limit to \&\&\&\& to \&\&\&\&/2 */
fm.sample_rate *= fm.post_downsample;

    if (fm.freq_len == 0) {
        __android_log_print(ANDROID_LOG_WARN, "rtl_fm", "Please specify a frequency.\n")
        exit(1);
    }

    if (fm.freq_len > 1) {
        fm.terminate_on_squelch = 0;
    }

    if (argc <= optind) {
        //usage();
        filename = ".-";
    } else {
        filename = argv[optind];
    }

    buffer = malloc(lcm_post[fm.post_downsample] * DEFAULT_BUF_LENGTH * sizeof(uint8_t));

    device_count = rtlsdr_get_device_count();
    if (!device_count) {
        __android_log_print(ANDROID_LOG_WARN, "rtl_fm", "No supported devices found.\n")
        exit(1);
    }

    __android_log_print(ANDROID_LOG_INFO, "rtl_fm", "Found %d device(s):\n", device_count);
    for (i = 0; i < device_count; i++) {
        rtlsdr_get_device_usb_strings(i, vendor, product, serial);
        __android_log_print(ANDROID_LOG_INFO, "rtl_fm", " %d: %s, %s, SN: %s\n", i, vendor, product, serial);

    219
__android_log_print(ANDROID_LOG_INFO, "rtl_fm", "\n");

__android_log_print(ANDROID_LOG_INFO, "rtl_fm", "Using device %d:
" ,
    dev_index, rtlsdr_get_device_name(dev_index));

r = rtlsdr_open(&dev, dev_index);
if (r < 0) {
    __android_log_print(ANDROID_LOG_DEBUG, "rtl_fm", "Failed to
open rtlsdr device #%d.\n", dev_index);
    exit(1);
}
#endif
sigact.sa_handler = sighandler;
sigemptyset(&sigact.sa_mask);
sigact.sa_flags = 0;
sigaction(SIGINT, &sigact, NULL);
sigaction(SIGTERM, &sigact, NULL);
sigaction(SIGQUIT, &sigact, NULL);
sigaction(SIGPIPE, &sigact, NULL);
#endif
SetConsoleCtrlHandler( (PHANDLER_ROUTINE) sighandler, TRUE );
#endif

rgba;  
/* WBFM is special */
if (wb_mode) {
    fm.freqs[0] += 16000;
}

if (fm.deemph) {
    fm.deemph_a = (int)round(1.0/((1.0-exp(-1.0/(fm.output_rate * 75e-6)))));
}

optimal_settings(&fm, 0, 0);
build_fir(&fm);

/* Set the tuner gain */
if (gain == AUTO_GAIN) {
    r = rtlsdr_set_tuner_gain_mode(dev, 0);
} else {
    r = rtlsdr_set_tuner_gain_mode(dev, 1);
    r = rtlsdr_set_tuner_gain(dev, gain);
}
if (r != 0) {
__android_log_print(ANDROID_LOG_WARN, "rtl_fm", "WARNING: Failed to set tuner gain.\n");
} else if (gain == AUTO_GAIN) {
    __android_log_print(ANDROID_LOG_INFO, "rtl_fm", "Tuner gain set to automatic.\n");
} else {
    __android_log_print(ANDROID_LOG_INFO, "rtl_fm", "Tuner gain set to %0.2f dB.\n", gain/10.0);
}

r = rtlsdr_set_freq_correction(dev, ppm_error);

if (strcmp(filename, "-") == 0) { /* Write samples to stdout */
    fm.file = stdout;
#ifdef _WIN32
    _setmode(_fileno(fm.file), _O_BINARY);
#endif
    else {
        fm.file = fopen(filename, "wb");
        if (!fm.file) {
            fprintf(stderr, "Failed to open %s\n", filename);
            exit(1);
        }
    }
/* Reset endpoint before we start reading from it (mandatory) */
r = rtlsdr_reset_buffer(dev);

if (r < 0) {
    __android_log_print(ANDROID_LOG_WARN, "rtl_fm", "WARNING: Failed to reset buffers.\n");
}

pthread_create(&demod_thread, NULL, demod_thread_fn, (void *)(&fm));
rtlsdr_read_async(dev, rtlsdr_callback, (void *)(&fm),
    DEFAULT_ASYNC_BUF_NUMBER,
    lcm_post[fm.post_downsample] * DEFAULT_BUF_LENGTH);

if (do_exit) {
    __android_log_print(ANDROID_LOG_INFO, "rtl_fm", "\nUser cancel, exiting...\n");
} else {
    __android_log_print(ANDROID_LOG_DEBUG, "rtl_fm", "\nLibrary error %d, exiting...\n", r);
    rtlsdr_cancel_async(dev);
    pthread_mutex_destroy(&data_ready);

    221
pthread_mutex_destroy(&data_write);

if (fm.file != stdout) {
    fclose(fm.file);
}

rtltdr_close(dev);
free (buffer);
deleteBrgNodes();
return r >= 0 ? r : -r;
}

#ifdef ANDROID
#include <jni.h>
#include "libusbhelper.h"

JNIEXPORT jint JNICALL Java_com_example_airguide_AirGuideActivity_nativeRtlSdrInitialize(JNIEnv *envp,
jobject objp, jstring channel)
{
    init_libusbhelper(envp,objp);

    // Amplitude modulation at 113.1MHz
    char * args[] = {"rtl_fm", "-M", "-f", "113.1e6"};
    char *name;

    name = (char *)(*envp)->GetStringUTFChars(envp, channel, NULL);

    return main(4,args, name);
}

JNIEXPORT void JNICALL Java_com_example_airguide_AirGuideActivity_nativeRtlSdrChannel(JNIEnv *envp,
jobject objp, jint freq)
{
    __android_log_print(ANDROID_LOG_INFO, "rtl_fm", "Set Freq to : %d Hz\n", freq);

    fm.freqs[0] = (uint32_t)freq;

    // Set the frequency to connect to the new station
    optimal_settings(&fm, 0, 1);
}
JNIEXPORT int JNICALL
Java_com_example_airguide_AirGuideActivity_nativeRtlSdrGetBrg(JNIEnv *envp,
jobject objp)
{
    //__android_log_print(ANDROID_LOG_INFO, "rtl_fm", "Brg Req: %d",
g_nBrg);

    return g_nBrg;
}

#undef
#ifndef __NATIVE_AUDIO_JNI_H__
#define __NATIVE_AUDIO_JNI_H__

#include <pthread.h>

typedef struct sMutex_{
  pthread_mutex_t m;
  pthread_cond_t c;
  unsigned char s;
} sMutex;

int selectClipBuf(short *, int);

#endif
// This file was taken from the NDK example project but has been heavily modified for
// the purposes of this project, it’s main purpose is to play the Morse code audio on the
// native C side.

/*
 * Copyright (C) 2010 The Android Open Source Project
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *   http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
 * implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

/* This is a JNI example where we use native methods to play sounds
 * using OpenSL ES. See the corresponding Java source file located at:
 *
 *   src/com/example/nativeaudio/NativeAudio/NativeAudio.java
 */

#include <assert.h>
#include <jni.h>
#include <string.h>
#include <pthread.h>

// for __android_log_print(ANDROID_LOG_INFO, "YourApp", "formatted message");
#include <android/log.h>

// for native audio
#include <SLES/OpenSLES.h>
#include <SLES/OpenSLES_Android.h>

// for native asset manager
#include <sys/types.h>
#include <android/asset_manager.h>
#include <android/asset_manager_jni.h>


#include "native-audio-jni.h"

#include <pthread.h>
static pthread_t tid;
static int g_on = 1;

#define DEBUG_PRINT 0

// engine interfaces
static SLObjectItf engineObject = NULL;
static SLEngineItf engineEngine;

// output mix interfaces
static SLObjectItf outputMixObject = NULL;
static SLEnvironmentalReverbItf outputMixEnvironmentalReverb = NULL;

// buffer queue player interfaces
static SLObjectItf bqPlayerObject = NULL;
static SLPlayItf bqPlayerPlay;
static SLAndroidSimpleBufferQueueItf bqPlayerBufferQueue;
static SLEffectSendItf bqPlayerEffectSend;
static SLVolumeItf bqPlayerVolume;

// aux effect on the output mix, used by the buffer queue player
static const SLEnvironmentalReverbSettings reverbSettings =
SL_I3DL2_ENVIRONMENT_PRESET_STONECORRIDOR;

#define NUM_BUFS 2
#define RADIO_FRAMES (32000 * 10)
static short radioBuffer[NUM_BUFS][RADIO_FRAMES];
static int radioSize[NUM_BUFS];
static int readIndex = 0;
static int writeIndex = 0;
static short *nowBuffer[NUM_BUFS];
static unsigned int nowSize[NUM_BUFS];
static short nowIndex = 0;
static unsigned int nowCount = 0;

static void * g_mutex;

// Mutuexes ensure synchronization between callbacks and processing code
void* createMutex(void)
{
    sMutex  *p;
    p = (sMutex*)malloc(sizeof(sMutex));
if (p == NULL)  
    return p;
memset(p, 0, sizeof(sMutex));
if (pthread_mutex_init(&(p->m), (pthread_mutexattr_t*)NULL) != 0) {
    free((void*)p);
    return NULL;
}
if (pthread_cond_init(&(p->c), (pthread_condattr_t*)NULL) != 0) {
    pthread_mutex_destroy(&(p->m));
    free((void*)p);
    return NULL;
}
p->s = (unsigned char)1;
return p;
}

int waitForMutex(void *lock)  
{
    sMutex *p;
    int   retval = 0;
    p = (sMutex*)lock;
    pthread_mutex_lock(&(p->m));
    while (!p->s) {
        pthread_cond_wait(&(p->c), &(p->m));
    }
    p->s = (unsigned char)0;
    pthread_mutex_unlock(&(p->m));
}

void notifyMutex(void *lock)  
{
    sMutex *p;
    p = (sMutex*)lock;
    pthread_mutex_lock(&(p->m));
    p->s = (unsigned char)1;
    pthread_cond_signal(&(p->c));
    pthread_mutex_unlock(&(p->m));
}

void destroyMutex(void *lock)  
{
    sMutex *p;
    p = (sMutex*)lock;
    if (p == NULL)  
        return;

227
notifyMutex(p);
pthread_cond_destroy(&(p->c));
pthread_mutex_destroy(&(p->m));
free(p);
}

void* enqueueAudioForPlayback(void *arg)
{
    unsigned long i = 0;
pthread_t id = pthread_self();
SLresult result;

    while (g_on)
    {
        waitForMutex(g_mutex);

        if (nowCount-- > 0)
        {
            usleep(100);
#if DEBUG_PRINT
            __android_log_print(ANDROID_LOG_INFO, "Sleep Activated",
            "Sleep");
#endif
            #endif

        }

        result = (*bqPlayerBufferQueue)->Enqueue(bqPlayerBufferQueue,
nowBuffer[nowIndex], nowSize[nowIndex] * sizeof(short));
#if DEBUG_PRINT
            __android_log_print(ANDROID_LOG_INFO, "Enqueued", "Index: %d ,
Size: %d", nowIndex, nowSize[nowIndex] * sizeof(short));
#endif
        nowIndex = (nowIndex ? 0 : 1);

        if (!g_on)
            return NULL;
    }

    return NULL;
}

void startListening(void)
{
    int err;
g_mutex = createMutex();
    waitForMutex(g_mutex);
}
err = pthread_create(&tid, NULL, &enqueueAudioForPlayback, NULL);
if (err != 0)
    __android_log_print(ANDROID_LOG_INFO, "Native Audio", "Can't create thread : [%s]", strerror(err));
else
    __android_log_print(ANDROID_LOG_INFO, "Native Audio", "Thread created successfully");

int selectClipBuf(short * _buf, int _size)
{
    static int next = 0;
    static unsigned int waitOnce = 0, dontPlay = 0, doOnce = 0;
    static int currIndex = 0;
    static int i = 0;
    int j = 0, k = 0;

    for (i = currIndex; i < (currIndex + (_size)); i++)
    {
        radioBuffer[writeIndex][i] = _buf[j];
        j++;
    }
    currIndex = i;

    if (!doOnce)
    {
        nowBuffer[0] = radioBuffer[0];
        nowBuffer[1] = radioBuffer[1];
        doOnce = 1;
    }

    if (currIndex >= 32000 * 1)
    {
        nowSize[writeIndex] = currIndex;
        nowCount++;

        if (waitOnce > 0 && dontPlay == 0)
        {
            // For long clips, we would enqueue only once
            // but since we are streaming playback we enqueue at least 2
            buffers to start
            SLresult result;

            result = (*bqPlayerBufferQueue)-&Enqueue(bqPlayerBufferQueue, nowBuffer[0], nowSize[0]);
        }
    }
result = (*bqPlayerBufferQueue)->Enqueue(bqPlayerBufferQueue, nowBuffer[1], nowSize[1]);

if (SL_RESULT_SUCCESS != result) {
    return 0;
}

dontPlay = 1;

#if DEBUG_PRINT
    __android_log_print(ANDROID_LOG_INFO, "NA - Buffer Ready", "Size: %d", nowSize[writeIndex]);
#endif

// reset index to 0
currIndex = 0;

// goto next buffer
writeIndex = (writeIndex ? 0 : 1);

// Modified waitOnce from waitOnce++ to waitOnce = 1
waitOnce = 1;

return 1;

// this callback handler is called every time a buffer finishes playing
void bqPlayerCallback(SLAndroidSimpleBufferQueueItf bq, void *context) {
    static int i = 0;

    assert(bq == bqPlayerBufferQueue);
    assert(NULL == context);

    notifyMutex(g_mutex);

#if DEBUG_PRINT
    __android_log_print(ANDROID_LOG_INFO, "Native Audio", "*****Callback*****");
#endif
}

// create the engine and output mix objects
void Java_com_example_airguide_AirGuideActivity_createEngine(JNIEnv* env, jclass clazz)
{
    SLresult result;

    // create engine
    result = slCreateEngine(&engineObject, 0, NULL, 0, NULL, NULL);
    assert(SL_RESULT_SUCCESS == result);

    // realize the engine
    result = (*engineObject)->Realize(engineObject, SL_BOOLEAN_FALSE);
    assert(SL_RESULT_SUCCESS == result);

    // get the engine interface, which is needed in order to create other objects
    result = (*engineObject)->GetInterface(engineObject, SL_IID_ENGINE, &engineEngine);
    assert(SL_RESULT_SUCCESS == result);

    // create output mix, with environmental reverb specified as a non-required interface
    const SLInterfaceID ids[1] = {SL_IID_ENVIRONMENTALREVERB};
    const SLboolean req[1] = { SL_BOOLEAN_FALSE);
    result = (*engineEngine)->CreateOutputMix(engineEngine, &outputMixObject, 1, ids, req);
    assert(SL_RESULT_SUCCESS == result);

    // realize the output mix
    result = (*outputMixObject)->Realize(outputMixObject, SL_BOOLEAN_FALSE);
    assert(SL_RESULT_SUCCESS == result);

    // get the environmental reverb interface
    // this could fail if the environmental reverb effect is not available,
    // either because the feature is not present, excessive CPU load, or
    // the required MODIFY_AUDIO_SETTINGS permission was not requested and granted
    result = (*outputMixObject)->GetInterface(outputMixObject, SL_IID_ENVIRONMENTALREVERB, &outputMixEnvironmentalReverb);
    if (SL_RESULT_SUCCESS == result) {
        result = (*outputMixEnvironmentalReverb)->SetEnvironmentalReverbProperties(
                outputMixEnvironmentalReverb, &reverbSettings);
    }
    // ignore unsuccessful result codes for environmental reverb, as it is optional for this example

}
startListening();

// create buffer queue audio player
void
Java_com_example_airguide_AirGuideActivity_createBufferQueueAudioPlayer(JNIEnv *
env,
   jclass clazz)
{
   SLresult result;

   // configure audio source
   SLDataLocator_AndroidSimpleBufferQueue loc_bufq = {
      SL_DATALOCATOR_ANDROIDSIMPLEBUFFERQUEUE, 2);
   SLDataFormat_PCM format_pcm = { SL_DATAFORMAT_PCM, 1,
      SL_SAMPLINGRATE_32,
      SL_PCMSAMPLEFORMAT_FIXED_16,
      SL_PCMSAMPLEFORMAT_FIXED_16,
      SL_SPEAKER_FRONT_CENTER,
      SL_BYTEORDER_LITTLEENDIAN };
   SLDataSource audioSrc = { &loc_bufq, &format_pcm };  

   // configure audio sink
   SLDataLocator_OutputMix loc_outmix = {
      SL_DATALOCATOR_OUTPUTMIX, outputMixObject};
   SLDataSink audioSnk = { &loc_outmix, NULL };  

   // create audio player
   const SLInterfaceID ids[5] = { SL_IID_PLAY, SL_IID_BUFFERQUEUE,
      SL_IID_EFFECTSEND, SL_IID_ANDROIDCONFIGURATION,
      /*SL_IID_MUTESOLO*/,/* SL_IID_VOLUME */};
   const SLboolean req[5] = { SL_BOOLEAN_TRUE, SL_BOOLEAN_TRUE,
      SL_BOOLEAN_TRUE, SL_BOOLEAN_TRUE,
      /*SL_BOOLEAN_TRUE*/,/* SL_BOOLEAN_TRUE */};
   result = (*engineEngine)->CreateAudioPlayer(engineEngine, &bqPlayerObject,
      &audioSrc, &audioSnk,
      5, ids, req);
   assert(SL_RESULT_SUCCESS == result);

   // CreateAudioPlayer and specify SL_IID_ANDROIDCONFIGURATION
   // in the required interface ID array. Do not realize player yet.
   // ...
   SLAndroidConfigurationItf playerConfig;
   result = (*bqPlayerObject)->GetInterface(bqPlayerObject,
      SL_IID_ANDROIDCONFIGURATION, &playerConfig);
assert(SL_RESULT_SUCCESS == result);  
SLint32 streamType = SL_ANDROID_STREAM_MEDIA;  
result = (*playerConfig)->SetConfiguration(playerConfig,  
        SL_ANDROID_KEY_STREAM_TYPE, &streamType, sizeof(SLint32));  
assert(SL_RESULT_SUCCESS == result);  
// ...  
// Now realize the player  

// realize the player  
result = (*bqPlayerObject)->Realize(bqPlayerObject, SL_BOOLEAN_FALSE);  
assert(SL_RESULT_SUCCESS == result);  

// get the play interface  
result = (*bqPlayerObject)->GetInterface(bqPlayerObject, SL_IID_PLAY,  
        &bqPlayerPlay);  
assert(SL_RESULT_SUCCESS == result);  

// get the buffer queue interface  
result = (*bqPlayerObject)->GetInterface(bqPlayerObject,  
        SL_IID_BUFFERQUEUE,  
        &bqPlayerBufferQueue);  
assert(SL_RESULT_SUCCESS == result);  

// register callback on the buffer queue  
result = (*bqPlayerBufferQueue)->RegisterCallback(bqPlayerBufferQueue,  
        bqPlayerCallback, NULL);  
assert(SL_RESULT_SUCCESS == result);  

// get the effect send interface  
result = (*bqPlayerObject)->GetInterface(bqPlayerObject,  
        SL_IID_EFFECTSEND,  
        &bqPlayerEffectSend);  
assert(SL_RESULT_SUCCESS == result);  

// get the volume interface  
result = (*bqPlayerObject)->GetInterface(bqPlayerObject, SL_IID_VOLUME,  
        &bqPlayerVolume);  
assert(SL_RESULT_SUCCESS == result);  

// set the player's state to playing  
result = (*bqPlayerPlay)->SetPlayState(bqPlayerPlay,  
        SL_PLAYSTATE_PLAYING);  
assert(SL_RESULT_SUCCESS == result);  

}
// shut down the native audio system
void Java_com_example_airguide_AirGuideActivity_shutdown(JNIEnv* env, jclass clazz)
{
    // destroy buffer queue audio player object, and invalidate all associated interfaces
    if (bqPlayerObject != NULL) {
        (*bqPlayerObject)->Destroy(bqPlayerObject);
        bqPlayerObject = NULL;
        bqPlayerPlay = NULL;
        bqPlayerBufferQueue = NULL;
        bqPlayerEffectSend = NULL;
        bqPlayerVolume = NULL;
    }

    // destroy output mix object, and invalidate all associated interfaces
    if (outputMixObject != NULL) {
        (*outputMixObject)->Destroy(outputMixObject);
        outputMixObject = NULL;
        outputMixEnvironmentalReverb = NULL;
    }

    // destroy engine object, and invalidate all associated interfaces
    if (engineObject != NULL) {
        (*engineObject)->Destroy(engineObject);
        engineObject = NULL;
        engineEngine = NULL;
    }
}
Android.mk

# This file was taken as an example from RTL-SDR open source community but has been
# heavily modified for the purposes of compiling the necessary libraries for access from
# the Java side

# Copyright 2012 OSMOCOM Project
#
# This file is part of rtl-sdr
#
# GNU Radio is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 3, or (at your option)
# any later version.
#
# GNU Radio is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with GNU Radio; see the file COPYING. If not, write to
# the Free Software Foundation, Inc., 51 Franklin Street,
# Boston, MA 02110-1301, USA.
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)
LOCAL_MODULE := usb
LOCAL_SRC_FILES := lib/libusb.so
LOCAL_C_INCLUDES := $(LOCAL_PATH)/../jni/include
include $(PREBUILT_SHARED_LIBRARY)

# The librtlsdr shared library
#
include $(CLEAR_VARS)
LOCAL_MODULE := rtlsdr
LOCAL_SRC_FILES := \  
src/librtlsdr.c \  
src/tuner_e4k.c \  
src/tuner_fc0012.c \  
src/tuner_fc0013.c \  
src/tuner_r82xx.c \  
src/tuner_fc2580.c \  
LOCAL_LDLIBS := -llog
LOCAL_CFLAGS := -DANDROID -DUSE_LIBLOG
LOCAL_SHARED_LIBRARIES := usb
LOCAL_C_INCLUDES += $(LOCAL_PATH)/include
include $(BUILD_SHARED_LIBRARY)

include $(CLEAR_VARS)
LOCAL_MODULE := native-
audio-jni
LOCAL_SRC_FILES := src/native-
audio-jni.c
LOCAL_C_INCLUDES += $(LOCAL_PATH)/include
$(LOCAL_PATH)/android/include
LOCAL_CFLAGS := -DANDROID -DUSE_LIBLOG
# for native audio
LOCAL_LDLIBS += -lOpenSLES
# for logging
LOCAL_LDLIBS += -llog
# for native asset manager
LOCAL_LDLIBS += -landroid
include $(BUILD_SHARED_LIBRARY)

include $(CLEAR_VARS)
LOCAL_MODULE := rtlvordecoder
LOCAL_SRC_FILES := src/rtl_vor_decoder.c
LOCAL_C_INCLUDES += $(LOCAL_PATH)/include
$(LOCAL_PATH)/android/include
LOCAL_CFLAGS := -DANDROID -DUSE_LIBLOG
LOCAL_SHARED_LIBRARIES += usb
LOCAL_SHARED_LIBRARIES += rtlsdr
LOCAL_SHARED_LIBRARIES += native-
audio-jni
LOCAL_LDLIBS := -llog
LOCAL_LDLIBS += -landroid
include $(BUILD_SHARED_LIBRARY)

include $(CLEAR_VARS)
LOCAL_MODULE := rtlfm
LOCAL_SRC_FILES := src/rtl_fm.c src/libusbhelper.c
LOCAL_C_INCLUDES += $(LOCAL_PATH)/include \
    $(LOCAL_PATH)/android/include \
    /usr/local/include
LOCAL_CFLAGS := -DANDROID -DUSE_LIBLOG
LOCAL_SHARED_LIBRARIES += usb
LOCAL_SHARED_LIBRARIES += rtlsdr
LOCAL_SHARED_LIBRARIES += native-
audio-jni
LOCAL_SHARED_LIBRARIES += rtlvordecoder
LOCAL_LDLIBS := -llog
LOCAL_LDLIBS += -landroid
include $(BUILD_SHARED_LIBRARY)
Application.mk

APP_STL := stlport_shared
Open Source RTL-SDR Software

The Native portion of the code works in conjunction with the open source RTL-SDR software, which is available on the following GIT repository:

```plaintext
git://git.osmocom.org/rtl-sdr.git
```

The user just needs to type the following in a terminal and all the source software will be downloaded to the current directory.

```plaintext
git clone git://git.osmocom.org/rtl-sdr.git
```

If the user does not have GIT installed, please reference https://git-scm.com/book/en/v2/Getting-Started-Installing-Git for all necessary installation instructions.
### APPENDIX D: NAVAID DATABASE

<table>
<thead>
<tr>
<th>KeyID</th>
<th>State</th>
<th>ID</th>
<th>Name</th>
<th>Type</th>
<th>Channel</th>
<th>Freq</th>
<th>Latitude</th>
<th>Longitude</th>
<th>Airport</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CA</td>
<td>CV</td>
<td>ABETA</td>
<td>NDB</td>
<td>233.0</td>
<td>40.9645</td>
<td>-124.1</td>
<td>KACV</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>CA</td>
<td>AHC</td>
<td>AMEDEE</td>
<td>VOR-DME</td>
<td>027X</td>
<td>109.0</td>
<td>40.2679</td>
<td>-120.15</td>
<td>AHC</td>
</tr>
<tr>
<td>3</td>
<td>CA</td>
<td>ACV</td>
<td>ARCATA</td>
<td>VOR-DME</td>
<td>039X</td>
<td>110.2</td>
<td>40.9814</td>
<td>-124.11</td>
<td>KACV</td>
</tr>
<tr>
<td>4</td>
<td>CA</td>
<td>AVE</td>
<td>AVENAL</td>
<td>VORTAC</td>
<td>118X</td>
<td>117.1</td>
<td>35.647</td>
<td>-119.98</td>
<td>L84</td>
</tr>
<tr>
<td>5</td>
<td>CA</td>
<td>BAB</td>
<td>BEALE</td>
<td>TACAN</td>
<td>023X</td>
<td>-1.0</td>
<td>39.1348</td>
<td>-121.44</td>
<td>KBAB</td>
</tr>
<tr>
<td>6</td>
<td>CA</td>
<td>NSD</td>
<td>BEAVER</td>
<td>TACAN</td>
<td>086X</td>
<td>-1.0</td>
<td>32.88</td>
<td>-118.44</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>CA</td>
<td>LG</td>
<td>BECCA</td>
<td>NDB</td>
<td>233.0</td>
<td>33.7567</td>
<td>-118.08</td>
<td>KSLI</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>CA</td>
<td>BSR</td>
<td>BIG SUR</td>
<td>VORTAC</td>
<td>087X</td>
<td>114.0</td>
<td>36.1813</td>
<td>-121.64</td>
<td>HGT</td>
</tr>
<tr>
<td>9</td>
<td>CA</td>
<td>BIH</td>
<td>BISHOP</td>
<td>VOR-DME</td>
<td>033X</td>
<td>109.6</td>
<td>37.377</td>
<td>-118.37</td>
<td>BIH</td>
</tr>
<tr>
<td>10</td>
<td>CA</td>
<td>BLH</td>
<td>BLYTHE</td>
<td>VORTAC</td>
<td>121X</td>
<td>117.4</td>
<td>33.5961</td>
<td>-114.76</td>
<td>15AZ</td>
</tr>
<tr>
<td>11</td>
<td>CA</td>
<td>AN</td>
<td>BOING</td>
<td>NDB</td>
<td>245.0</td>
<td>32.7399</td>
<td>-117.22</td>
<td>KSAN</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>CA</td>
<td>SF</td>
<td>BRIJJ</td>
<td>NDB</td>
<td>379.0</td>
<td>37.5722</td>
<td>-122.26</td>
<td>KSFO</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>CA</td>
<td>CMA</td>
<td>CAMARILLO</td>
<td>VOR-DME</td>
<td>105X</td>
<td>115.8</td>
<td>34.2125</td>
<td>-119.09</td>
<td>CMA</td>
</tr>
<tr>
<td>14</td>
<td>CA</td>
<td>NFG</td>
<td>CAMP PENDLETON</td>
<td>TACAN</td>
<td>055X</td>
<td>-1.0</td>
<td>33.2744</td>
<td>-117.39</td>
<td>KNFG</td>
</tr>
<tr>
<td>15</td>
<td>CA</td>
<td>FCH</td>
<td>CHANDLER</td>
<td>NDB</td>
<td>344.0</td>
<td>36.7239</td>
<td>-119.83</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>CA</td>
<td>CIC</td>
<td>CHICO</td>
<td>VOR-DME</td>
<td>035X</td>
<td>109.8</td>
<td>39.7898</td>
<td>-121.85</td>
<td>CIC</td>
</tr>
<tr>
<td>17</td>
<td>CA</td>
<td>NID</td>
<td>CHINA LAKE</td>
<td>TACAN</td>
<td>053X</td>
<td>-1.0</td>
<td>35.6879</td>
<td>-117.69</td>
<td>NID</td>
</tr>
<tr>
<td>18</td>
<td>CA</td>
<td>UAD</td>
<td>CHUALAR</td>
<td>NDB</td>
<td>263.0</td>
<td>36.4909</td>
<td>-121.47</td>
<td>KSNS</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>CA</td>
<td>CZQ</td>
<td>CLOVIS</td>
<td>VORTAC</td>
<td>076X</td>
<td>112.9</td>
<td>36.8843</td>
<td>-119.82</td>
<td>KFAT</td>
</tr>
<tr>
<td>20</td>
<td>CA</td>
<td>CPM</td>
<td>COMPTON</td>
<td>NDB</td>
<td>378.0</td>
<td>33.8889</td>
<td>-118.25</td>
<td>CPM</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>CA</td>
<td>CCR</td>
<td>CONCORD</td>
<td>VOR-DME</td>
<td>117X</td>
<td>117.0</td>
<td>38.0449</td>
<td>-122.05</td>
<td>CCR</td>
</tr>
<tr>
<td>22</td>
<td>CA</td>
<td>NXP</td>
<td>COYOTE</td>
<td>TACAN</td>
<td>063X</td>
<td>-1.0</td>
<td>34.2967</td>
<td>-116.16</td>
<td>NXP</td>
</tr>
<tr>
<td>23</td>
<td>CA</td>
<td>CEC</td>
<td>CRESCENT CITY</td>
<td>VORTAC</td>
<td>027X</td>
<td>109.0</td>
<td>41.7796</td>
<td>-124.24</td>
<td>KCEC</td>
</tr>
<tr>
<td>24</td>
<td>CA</td>
<td>DAG</td>
<td>DAGGETT</td>
<td>VORTAC</td>
<td>079X</td>
<td>113.2</td>
<td>34.9625</td>
<td>-116.58</td>
<td>KBYS</td>
</tr>
<tr>
<td>25</td>
<td>CA</td>
<td>MY</td>
<td>DEORO</td>
<td>NDB</td>
<td>210.0</td>
<td>32.7799</td>
<td>-117.04</td>
<td>MYF</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>CA</td>
<td>EDW</td>
<td>EDWARDS</td>
<td>VORTAC</td>
<td>111X</td>
<td>116.4</td>
<td>34.9824</td>
<td>-117.73</td>
<td>KEDW</td>
</tr>
<tr>
<td>27</td>
<td>CA</td>
<td>NJK</td>
<td>EL CENTRO</td>
<td>TACAN</td>
<td>047X</td>
<td>-1.0</td>
<td>32.8318</td>
<td>-115.68</td>
<td>KNJK</td>
</tr>
<tr>
<td>28</td>
<td>CA</td>
<td>EMT</td>
<td>EL MONTE</td>
<td>NDB</td>
<td>359.0</td>
<td>34.0835</td>
<td>-118.03</td>
<td>EMT</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>CA</td>
<td>HYP</td>
<td>EL NIDO</td>
<td>VOR-DME</td>
<td>089X</td>
<td>114.2</td>
<td>37.2194</td>
<td>-120.4</td>
<td>DQ8</td>
</tr>
<tr>
<td>30</td>
<td>CA</td>
<td>ELB</td>
<td>EL TORO</td>
<td>VOR-DME</td>
<td>119X</td>
<td>117.2</td>
<td>33.676</td>
<td>-117.73</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>CA</td>
<td>SA</td>
<td>EXECC</td>
<td>NDB</td>
<td>356.0</td>
<td>38.4497</td>
<td>-121.55</td>
<td>KSAC</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>CA</td>
<td>FLW</td>
<td>FELLOWS</td>
<td>VORTAC</td>
<td>122X</td>
<td>117.5</td>
<td>35.0931</td>
<td>-119.87</td>
<td>L17</td>
</tr>
<tr>
<td>33</td>
<td>CA</td>
<td>FIM</td>
<td>FILLMORE</td>
<td>VORTAC</td>
<td>072X</td>
<td>112.5</td>
<td>34.3567</td>
<td>-118.88</td>
<td>KNTD</td>
</tr>
<tr>
<td>34</td>
<td>CA</td>
<td>FJS</td>
<td>FORT JONES</td>
<td>VOR-DME</td>
<td>033X</td>
<td>109.6</td>
<td>41.4497</td>
<td>-122.81</td>
<td>106</td>
</tr>
<tr>
<td>35</td>
<td>CA</td>
<td>FOT</td>
<td>FORTUNA</td>
<td>VORTAC</td>
<td>087X</td>
<td>114.0</td>
<td>40.6713</td>
<td>-124.23</td>
<td>FOT</td>
</tr>
<tr>
<td>36</td>
<td>CA</td>
<td>FRA</td>
<td>FRIANT</td>
<td>VORTAC</td>
<td>103X</td>
<td>115.6</td>
<td>37.1044</td>
<td>-119.6</td>
<td>SCL0</td>
</tr>
<tr>
<td>37</td>
<td>CA</td>
<td>GVO</td>
<td>GAVIOTA</td>
<td>VORTAC</td>
<td>085X</td>
<td>113.8</td>
<td>34.5313</td>
<td>-120.09</td>
<td>KSBA</td>
</tr>
<tr>
<td>38</td>
<td>CA</td>
<td>GWF</td>
<td>GEN WILLIAM J FOX</td>
<td>NDB</td>
<td>282.0</td>
<td>34.7387</td>
<td>-118.22</td>
<td>WJF</td>
<td></td>
</tr>
<tr>
<td>KeyID</td>
<td>State</td>
<td>ID</td>
<td>Name</td>
<td>Type</td>
<td>Channel</td>
<td>Freq</td>
<td>Latitude</td>
<td>Longitude</td>
<td>Airport</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-----</td>
<td>------------</td>
<td>--------</td>
<td>---------</td>
<td>-------</td>
<td>-----------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>39</td>
<td>CA</td>
<td>GFS</td>
<td>GOFFS</td>
<td>VORTAC</td>
<td>091X</td>
<td>114.4</td>
<td>35.1311</td>
<td>-115.18</td>
<td>KOAK</td>
</tr>
<tr>
<td>40</td>
<td>CA</td>
<td>GMN</td>
<td>GORMAN</td>
<td>VORTAC</td>
<td>108X</td>
<td>116.1</td>
<td>34.804</td>
<td>-118.86</td>
<td>L94</td>
</tr>
<tr>
<td>41</td>
<td>CA</td>
<td>GLJ</td>
<td>GUADALUPE</td>
<td>VOR</td>
<td></td>
<td>111.0</td>
<td>34.9524</td>
<td>-120.52</td>
<td>KSMX</td>
</tr>
<tr>
<td>42</td>
<td>CA</td>
<td>MY</td>
<td>HALOW</td>
<td>NDB</td>
<td></td>
<td>222.0</td>
<td>39.1694</td>
<td>-121.61</td>
<td>KMYV</td>
</tr>
<tr>
<td>43</td>
<td>CA</td>
<td>HNW</td>
<td>HANGTOWN</td>
<td>VOR-DME</td>
<td>102X</td>
<td>115.5</td>
<td>38.7247</td>
<td>-120.75</td>
<td>PVF</td>
</tr>
<tr>
<td>44</td>
<td>CA</td>
<td>HEC</td>
<td>HECTOR</td>
<td>VORTAC</td>
<td>074X</td>
<td>112.7</td>
<td>34.797</td>
<td>-116.46</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>CA</td>
<td>HDF</td>
<td>HOMELAND</td>
<td>VOR</td>
<td></td>
<td>113.4</td>
<td>33.7763</td>
<td>-117.19</td>
<td>KRIV</td>
</tr>
<tr>
<td>46</td>
<td>CA</td>
<td>HGT</td>
<td>HUNTER LIGGETT</td>
<td>NDB</td>
<td></td>
<td>209.0</td>
<td>35.9611</td>
<td>-121.19</td>
<td>HGT</td>
</tr>
<tr>
<td>47</td>
<td>CA</td>
<td>IPL</td>
<td>IMPERIAL</td>
<td>VORTAC</td>
<td>106X</td>
<td>115.9</td>
<td>32.7489</td>
<td>-115.51</td>
<td>KNJK</td>
</tr>
<tr>
<td>48</td>
<td>CA</td>
<td>NRS</td>
<td>IMPERIAL BEACH</td>
<td>TACAN</td>
<td>029X</td>
<td>-1.0</td>
<td>32.5642</td>
<td>-117.11</td>
<td>NRS</td>
</tr>
<tr>
<td>49</td>
<td>CA</td>
<td>SC</td>
<td>JOTLY</td>
<td>NDB</td>
<td></td>
<td>271.0</td>
<td>37.8317</td>
<td>-121.14</td>
<td>KSCK</td>
</tr>
<tr>
<td>50</td>
<td>CA</td>
<td>JLI</td>
<td>JULIAN</td>
<td>VORTAC</td>
<td>087X</td>
<td>114.0</td>
<td>33.1405</td>
<td>-116.59</td>
<td>RNM</td>
</tr>
<tr>
<td>51</td>
<td>CA</td>
<td>CC</td>
<td>KANAN</td>
<td>NDB</td>
<td></td>
<td>335.0</td>
<td>38.0464</td>
<td>-122.03</td>
<td>CCR</td>
</tr>
<tr>
<td>52</td>
<td>CA</td>
<td>UK</td>
<td>KEARN</td>
<td>NDB</td>
<td></td>
<td>371.0</td>
<td>39.2747</td>
<td>-123.25</td>
<td>KUKI</td>
</tr>
<tr>
<td>53</td>
<td>CA</td>
<td>LHS</td>
<td>LAKE HUGHS</td>
<td>VORTAC</td>
<td>021X</td>
<td>108.4</td>
<td>34.683</td>
<td>-118.58</td>
<td>TSP</td>
</tr>
<tr>
<td>54</td>
<td>CA</td>
<td>LOP</td>
<td>LAMPSON</td>
<td>NDB</td>
<td></td>
<td>217.0</td>
<td>38.9952</td>
<td>-122.88</td>
<td>1O2</td>
</tr>
<tr>
<td>55</td>
<td>CA</td>
<td>HU</td>
<td>LANEY</td>
<td>NDB</td>
<td></td>
<td>400.0</td>
<td>38.617</td>
<td>-121.6</td>
<td>KSMF</td>
</tr>
<tr>
<td>56</td>
<td>CA</td>
<td>RD</td>
<td>LASSN</td>
<td>NDB</td>
<td></td>
<td>367.0</td>
<td>40.3928</td>
<td>-122.29</td>
<td>KRDD</td>
</tr>
<tr>
<td>57</td>
<td>CA</td>
<td>NLC</td>
<td>LEMOORE</td>
<td>TACAN</td>
<td>080X</td>
<td>-1.0</td>
<td>36.3441</td>
<td>-119.97</td>
<td>NLC</td>
</tr>
<tr>
<td>58</td>
<td>CA</td>
<td>LIN</td>
<td>LINDEN</td>
<td>VORTAC</td>
<td>095X</td>
<td>114.8</td>
<td>38.0746</td>
<td>-121</td>
<td>KRIU</td>
</tr>
<tr>
<td>59</td>
<td>CA</td>
<td>LAX</td>
<td>LOS ANGELES</td>
<td>VORTAC</td>
<td>083X</td>
<td>113.6</td>
<td>33.9331</td>
<td>-118.43</td>
<td>KLAX</td>
</tr>
<tr>
<td>60</td>
<td>CA</td>
<td>NA</td>
<td>MAAGG</td>
<td>NDB</td>
<td></td>
<td>337.0</td>
<td>33.6888</td>
<td>-117.86</td>
<td>KSNA</td>
</tr>
<tr>
<td>61</td>
<td>CA</td>
<td>ECA</td>
<td>MANTECA</td>
<td>VORTAC</td>
<td>107X</td>
<td>116.0</td>
<td>37.8336</td>
<td>-121.17</td>
<td>KSCK</td>
</tr>
<tr>
<td>62</td>
<td>CA</td>
<td>RIV</td>
<td>MARCH</td>
<td>TACAN</td>
<td>077X</td>
<td>-1.0</td>
<td>33.9065</td>
<td>-117.28</td>
<td>KRIV</td>
</tr>
<tr>
<td>63</td>
<td>CA</td>
<td>MYV</td>
<td>MARYSVILLE</td>
<td>VOR-DME</td>
<td>045X</td>
<td>110.8</td>
<td>39.0986</td>
<td>-121.57</td>
<td>KMYV</td>
</tr>
<tr>
<td>64</td>
<td>CA</td>
<td>MXW</td>
<td>MAXWELL</td>
<td>VORTAC</td>
<td>037X</td>
<td>110.0</td>
<td>39.3176</td>
<td>-122.22</td>
<td>WLW</td>
</tr>
<tr>
<td>65</td>
<td>CA</td>
<td>MCC</td>
<td>MC CLELLAN</td>
<td>VOR-DME</td>
<td>029X</td>
<td>109.2</td>
<td>38.6674</td>
<td>-121.4</td>
<td>KMCC</td>
</tr>
<tr>
<td>66</td>
<td>CA</td>
<td>ENI</td>
<td>MENOCINO</td>
<td>VORTAC</td>
<td>070X</td>
<td>112.3</td>
<td>39.0532</td>
<td>-123.27</td>
<td>KUKI</td>
</tr>
<tr>
<td>67</td>
<td>CA</td>
<td>SM</td>
<td>METRE</td>
<td>NDB</td>
<td></td>
<td>230.0</td>
<td>38.795</td>
<td>-121.6</td>
<td>KSMF</td>
</tr>
<tr>
<td>68</td>
<td>CA</td>
<td>NMX</td>
<td>MIRAMAR</td>
<td>TACAN</td>
<td>033X</td>
<td>-1.0</td>
<td>32.8697</td>
<td>-117.15</td>
<td>KNMX</td>
</tr>
<tr>
<td>69</td>
<td>CA</td>
<td>MZB</td>
<td>MISSION BAY</td>
<td>VORTAC</td>
<td>125X</td>
<td>117.8</td>
<td>32.7822</td>
<td>-117.23</td>
<td>SEE</td>
</tr>
<tr>
<td>70</td>
<td>CA</td>
<td>MOD</td>
<td>MODESTO</td>
<td>VOR-DME</td>
<td>093X</td>
<td>114.6</td>
<td>37.6273</td>
<td>-120.96</td>
<td>KMOD</td>
</tr>
<tr>
<td>71</td>
<td>CA</td>
<td>NUQ</td>
<td>MOFFETT</td>
<td>TACAN</td>
<td>123X</td>
<td>-1.0</td>
<td>37.4324</td>
<td>-122.06</td>
<td>KNUQ</td>
</tr>
<tr>
<td>72</td>
<td>CA</td>
<td>MOG</td>
<td>MONTAGUE</td>
<td>NDB</td>
<td></td>
<td>404.0</td>
<td>41.7272</td>
<td>-122.48</td>
<td>SIY</td>
</tr>
<tr>
<td>73</td>
<td>CA</td>
<td>MQO</td>
<td>MORRO BAY</td>
<td>VORTAC</td>
<td>071X</td>
<td>112.4</td>
<td>35.2523</td>
<td>-120.76</td>
<td>KSMX</td>
</tr>
<tr>
<td>74</td>
<td>CA</td>
<td>MR</td>
<td>MUNSO</td>
<td>NDB</td>
<td></td>
<td>385.0</td>
<td>36.6208</td>
<td>-121.94</td>
<td>KMRY</td>
</tr>
<tr>
<td>75</td>
<td>CA</td>
<td>EED</td>
<td>NEEDLES</td>
<td>VORTAC</td>
<td>099X</td>
<td>115.2</td>
<td>34.766</td>
<td>-114.47</td>
<td>IFP</td>
</tr>
<tr>
<td>76</td>
<td>CA</td>
<td>BF</td>
<td>NILEY</td>
<td>NDB</td>
<td></td>
<td>385.0</td>
<td>35.3724</td>
<td>-118.98</td>
<td>KBFL</td>
</tr>
<tr>
<td>77</td>
<td>CA</td>
<td>NZY</td>
<td>NORTH ISLAND</td>
<td>TACAN</td>
<td>117X</td>
<td>-1.0</td>
<td>32.7026</td>
<td>-117.22</td>
<td>KNZY</td>
</tr>
<tr>
<td>78</td>
<td>CA</td>
<td>OAK</td>
<td>OAKLAND</td>
<td>VORTAC</td>
<td>115X</td>
<td>116.8</td>
<td>37.7259</td>
<td>-122.22</td>
<td>KOAK</td>
</tr>
<tr>
<td>KeyID</td>
<td>State</td>
<td>ID</td>
<td>Name</td>
<td>Type</td>
<td>Channel</td>
<td>Freq</td>
<td>Latitude</td>
<td>Longitude</td>
<td>Airport</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-----</td>
<td>----------------</td>
<td>---------</td>
<td>---------</td>
<td>-------</td>
<td>----------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>79</td>
<td>CA</td>
<td>OCN</td>
<td>OCEANSIDE</td>
<td>VORTAC</td>
<td>100X</td>
<td>115.3</td>
<td>33.2406</td>
<td>-117.42</td>
<td>KNFG</td>
</tr>
<tr>
<td>80</td>
<td>CA</td>
<td>OVE</td>
<td>OROVILLE</td>
<td>VORTAC</td>
<td>092X</td>
<td>114.5</td>
<td>34.6314</td>
<td>-118.06</td>
<td>KPMO</td>
</tr>
<tr>
<td>81</td>
<td>CA</td>
<td>PAI</td>
<td>PACOIMA</td>
<td>VORTAC</td>
<td>103X</td>
<td>115.5</td>
<td>33.37</td>
<td>-116.43</td>
<td>KPSP</td>
</tr>
<tr>
<td>82</td>
<td>CA</td>
<td>PDG</td>
<td>PAJAR</td>
<td>VORTAC</td>
<td>059X</td>
<td>112.2</td>
<td>33.9183</td>
<td>-117.53</td>
<td>RIR</td>
</tr>
<tr>
<td>83</td>
<td>CA</td>
<td>PSP</td>
<td>PALM SPRINGS</td>
<td>VORTAC</td>
<td>073X</td>
<td>117.9</td>
<td>34.102</td>
<td>-114.68</td>
<td>P20</td>
</tr>
<tr>
<td>84</td>
<td>CA</td>
<td>PMD</td>
<td>PALMDALE</td>
<td>VORTAC</td>
<td>010X</td>
<td>114.3</td>
<td>35.6724</td>
<td>-120.63</td>
<td>KPRB</td>
</tr>
<tr>
<td>85</td>
<td>CA</td>
<td>PXN</td>
<td>PANOCHE</td>
<td>VORTAC</td>
<td>035X</td>
<td>109.8</td>
<td>32.6103</td>
<td>-116.98</td>
<td>KSDM</td>
</tr>
<tr>
<td>86</td>
<td>CA</td>
<td>PDZ</td>
<td>PARADISE</td>
<td>VORTAC</td>
<td>012X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>POC</td>
</tr>
<tr>
<td>87</td>
<td>CA</td>
<td>PKE</td>
<td>PARKER</td>
<td>VORTAC</td>
<td>021X</td>
<td>108.4</td>
<td>40.5046</td>
<td>-122.29</td>
<td>KRDD</td>
</tr>
<tr>
<td>88</td>
<td>CA</td>
<td>PRB</td>
<td>PASO ROBLES</td>
<td>VORTAC</td>
<td>010X</td>
<td>104.0</td>
<td>40.0989</td>
<td>-122.24</td>
<td>KRBL</td>
</tr>
<tr>
<td>89</td>
<td>CA</td>
<td>SB</td>
<td>PETIS</td>
<td>VORTAC</td>
<td>009X</td>
<td>109.8</td>
<td>32.6103</td>
<td>-116.98</td>
<td>KSDM</td>
</tr>
<tr>
<td>90</td>
<td>CA</td>
<td>PNY</td>
<td>POGGI</td>
<td>VORTAC</td>
<td>021X</td>
<td>108.4</td>
<td>40.5046</td>
<td>-122.29</td>
<td>KRDD</td>
</tr>
<tr>
<td>91</td>
<td>CA</td>
<td>NTD</td>
<td>POINT MUGU</td>
<td>TACAN</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>92</td>
<td>CA</td>
<td>PYE</td>
<td>POINT REYES</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>93</td>
<td>CA</td>
<td>POM</td>
<td>POMONA</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>94</td>
<td>CA</td>
<td>ROM</td>
<td>PRIEST</td>
<td>VOR</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>95</td>
<td>CA</td>
<td>PBT</td>
<td>PROBERTA</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>96</td>
<td>CA</td>
<td>BBL</td>
<td>RED BLUFF</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>97</td>
<td>CA</td>
<td>RED</td>
<td>REDDING</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>98</td>
<td>CA</td>
<td>LV</td>
<td>REIGA</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>99</td>
<td>CA</td>
<td>RAL</td>
<td>RIVERSIDE</td>
<td>VOR</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>100</td>
<td>CA</td>
<td>OS</td>
<td>ROMEN</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>101</td>
<td>CA</td>
<td>SAC</td>
<td>SACRAMENTO</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>102</td>
<td>CA</td>
<td>SNS</td>
<td>SALINAS</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>103</td>
<td>CA</td>
<td>NUC</td>
<td>SAN CLEMENTE</td>
<td>TACAN</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>104</td>
<td>CA</td>
<td>SFO</td>
<td>SAN FRANCISCO</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>105</td>
<td>CA</td>
<td>SJY</td>
<td>SAN JACINTO</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>106</td>
<td>CA</td>
<td>SJC</td>
<td>SAN JOSE</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>107</td>
<td>CA</td>
<td>RZS</td>
<td>SAN MARCUS</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>108</td>
<td>CA</td>
<td>NSI</td>
<td>SAN NICOLAS</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>109</td>
<td>CA</td>
<td>NSI</td>
<td>SAN NICOLAS IS</td>
<td>TACAN</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>110</td>
<td>CA</td>
<td>SXC</td>
<td>SANTA CATALINA</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>111</td>
<td>CA</td>
<td>SMO</td>
<td>SANTA MONICA</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>112</td>
<td>CA</td>
<td>STS</td>
<td>SANTA ROSA</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>113</td>
<td>CA</td>
<td>SAU</td>
<td>SAUSALITO</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>114</td>
<td>CA</td>
<td>SGD</td>
<td>SCAGGS ISLAND</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>115</td>
<td>CA</td>
<td>SLI</td>
<td>SEAL BEACH</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>116</td>
<td>CA</td>
<td>EHF</td>
<td>SHAFTER</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>117</td>
<td>CA</td>
<td>SWR</td>
<td>SQUAW VALLEY</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>118</td>
<td>CA</td>
<td>TRM</td>
<td>THERMAL</td>
<td>VORTAC</td>
<td>004X</td>
<td>110.0</td>
<td>34.0784</td>
<td>-117.79</td>
<td>PVG</td>
</tr>
<tr>
<td>KeyID</td>
<td>State</td>
<td>ID</td>
<td>Name</td>
<td>Type</td>
<td>Channel</td>
<td>Freq</td>
<td>Latitude</td>
<td>Longitude</td>
<td>Airport</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-----</td>
<td>------------</td>
<td>--------</td>
<td>---------</td>
<td>-------</td>
<td>-----------</td>
<td>-----------</td>
<td>---------</td>
</tr>
<tr>
<td>119</td>
<td>CA</td>
<td>TCY</td>
<td>TRACY</td>
<td>NDB</td>
<td></td>
<td>203.0</td>
<td>37.6924</td>
<td>-121.44</td>
<td>TCY</td>
</tr>
<tr>
<td>120</td>
<td>CA</td>
<td>SUU</td>
<td>TRAVIS</td>
<td>TACAN</td>
<td>113X</td>
<td>-1.0</td>
<td>38.2456</td>
<td>-121.95</td>
<td>KSUU</td>
</tr>
<tr>
<td>121</td>
<td>CA</td>
<td>TZZ</td>
<td>TRAVIS</td>
<td>VOR</td>
<td>116.4</td>
<td>38.3442</td>
<td>-121.81</td>
<td></td>
<td></td>
</tr>
<tr>
<td>122</td>
<td>CA</td>
<td>TTE</td>
<td>TULE</td>
<td>VOR-DME</td>
<td>029X</td>
<td>109.2</td>
<td>35.9131</td>
<td>-119.02</td>
<td>PTV</td>
</tr>
<tr>
<td>123</td>
<td>CA</td>
<td>TNP</td>
<td>TWENTYNINE PALMS</td>
<td>VORTAC</td>
<td>089X</td>
<td>114.2</td>
<td>34.1122</td>
<td>-115.77</td>
<td>NXP</td>
</tr>
<tr>
<td>124</td>
<td>CA</td>
<td>VNY</td>
<td>VAN NUYS</td>
<td>VOR-DME</td>
<td>078X</td>
<td>113.1</td>
<td>34.2234</td>
<td>-118.49</td>
<td>VNY</td>
</tr>
<tr>
<td>125</td>
<td>CA</td>
<td>VBG</td>
<td>VANDENBERG</td>
<td>TACAN</td>
<td>059Y</td>
<td>-1.0</td>
<td>34.7326</td>
<td>-120.58</td>
<td>KVBG</td>
</tr>
<tr>
<td>126</td>
<td>CA</td>
<td>VTU</td>
<td>VENTURA</td>
<td>VOR-DME</td>
<td>019X</td>
<td>108.2</td>
<td>34.1151</td>
<td>-119.05</td>
<td></td>
</tr>
<tr>
<td>127</td>
<td>CA</td>
<td>VCV</td>
<td>VICTORVILLE</td>
<td>VOR-DME</td>
<td>031X</td>
<td>109.4</td>
<td>34.5942</td>
<td>-117.39</td>
<td>VCV</td>
</tr>
<tr>
<td>128</td>
<td>CA</td>
<td>VI</td>
<td>VILIA</td>
<td>NDB</td>
<td></td>
<td>220.0</td>
<td>36.2548</td>
<td>-119.31</td>
<td>VIS</td>
</tr>
<tr>
<td>129</td>
<td>CA</td>
<td>VIS</td>
<td>VISALIA</td>
<td>VOR-DME</td>
<td>031X</td>
<td>109.4</td>
<td>36.3673</td>
<td>-119.48</td>
<td>HJO</td>
</tr>
<tr>
<td>130</td>
<td>CA</td>
<td>ILA</td>
<td>WILLIAMS</td>
<td>VORTAC</td>
<td>091X</td>
<td>114.4</td>
<td>39.0711</td>
<td>-122.03</td>
<td>O52</td>
</tr>
<tr>
<td>131</td>
<td>CA</td>
<td>OSI</td>
<td>WOODSIDE</td>
<td>VORTAC</td>
<td>086X</td>
<td>113.9</td>
<td>37.3925</td>
<td>-122.28</td>
<td>KNUQ</td>
</tr>
<tr>
<td>132</td>
<td>CA</td>
<td>MO</td>
<td>WOWAR</td>
<td>NDB</td>
<td></td>
<td>367.0</td>
<td>37.5731</td>
<td>-120.86</td>
<td>KMOD</td>
</tr>
</tbody>
</table>
## APPENDIX E: FILEPARSER INPUT

<table>
<thead>
<tr>
<th>Time</th>
<th>Symbol</th>
<th>Price</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:00:00</td>
<td>VNY</td>
<td>48.2813</td>
<td>CMA</td>
<td>288.4667</td>
</tr>
<tr>
<td>12:00:01</td>
<td>VNY</td>
<td>51.0501</td>
<td>CMA</td>
<td>286.7028</td>
</tr>
<tr>
<td>12:00:02</td>
<td>VNY</td>
<td>54.0529</td>
<td>CMA</td>
<td>284.9057</td>
</tr>
<tr>
<td>12:00:03</td>
<td>VNY</td>
<td>57.3025</td>
<td>CMA</td>
<td>283.0781</td>
</tr>
<tr>
<td>12:00:04</td>
<td>VNY</td>
<td>60.8069</td>
<td>CMA</td>
<td>281.223</td>
</tr>
<tr>
<td>12:00:05</td>
<td>VNY</td>
<td>64.5681</td>
<td>CMA</td>
<td>279.3438</td>
</tr>
<tr>
<td>12:00:06</td>
<td>VNY</td>
<td>68.5793</td>
<td>CMA</td>
<td>277.4441</td>
</tr>
<tr>
<td>12:00:07</td>
<td>VNY</td>
<td>72.8231</td>
<td>CMA</td>
<td>275.5278</td>
</tr>
<tr>
<td>12:00:08</td>
<td>VNY</td>
<td>77.2703</td>
<td>CMA</td>
<td>273.599</td>
</tr>
<tr>
<td>12:00:09</td>
<td>VNY</td>
<td>81.8778</td>
<td>CMA</td>
<td>271.662</td>
</tr>
<tr>
<td>12:00:10</td>
<td>VNY</td>
<td>86.5952</td>
<td>CMA</td>
<td>269.7212</td>
</tr>
<tr>
<td>12:00:11</td>
<td>VNY</td>
<td>91.3582</td>
<td>CMA</td>
<td>267.781</td>
</tr>
<tr>
<td>12:00:12</td>
<td>VNY</td>
<td>96.1026</td>
<td>CMA</td>
<td>265.8459</td>
</tr>
<tr>
<td>12:00:13</td>
<td>VNY</td>
<td>100.7646</td>
<td>CMA</td>
<td>263.9203</td>
</tr>
<tr>
<td>12:00:14</td>
<td>VNY</td>
<td>105.2869</td>
<td>CMA</td>
<td>262.0083</td>
</tr>
<tr>
<td>12:00:15</td>
<td>VNY</td>
<td>109.6223</td>
<td>CMA</td>
<td>260.1141</td>
</tr>
<tr>
<td>12:00:16</td>
<td>VNY</td>
<td>113.7367</td>
<td>CMA</td>
<td>258.2414</td>
</tr>
<tr>
<td>12:00:17</td>
<td>VNY</td>
<td>117.6068</td>
<td>CMA</td>
<td>256.3939</td>
</tr>
<tr>
<td>12:00:18</td>
<td>VNY</td>
<td>121.2218</td>
<td>CMA</td>
<td>254.5748</td>
</tr>
<tr>
<td>12:00:19</td>
<td>VNY</td>
<td>124.5803</td>
<td>CMA</td>
<td>252.787</td>
</tr>
<tr>
<td>12:00:20</td>
<td>VNY</td>
<td>127.6879</td>
<td>CMA</td>
<td>251.033</td>
</tr>
<tr>
<td>12:00:21</td>
<td>VNY</td>
<td>130.5554</td>
<td>CMA</td>
<td>249.3153</td>
</tr>
<tr>
<td>12:00:22</td>
<td>VNY</td>
<td>133.1969</td>
<td>CMA</td>
<td>247.6355</td>
</tr>
<tr>
<td>12:00:23</td>
<td>VNY</td>
<td>135.628</td>
<td>CMA</td>
<td>245.9953</td>
</tr>
<tr>
<td>12:00:24</td>
<td>VNY</td>
<td>137.8651</td>
<td>CMA</td>
<td>244.3959</td>
</tr>
<tr>
<td>12:00:25</td>
<td>VNY</td>
<td>139.9246</td>
<td>CMA</td>
<td>242.8381</td>
</tr>
<tr>
<td>12:00:26</td>
<td>VNY</td>
<td>141.822</td>
<td>CMA</td>
<td>241.3226</td>
</tr>
<tr>
<td>12:00:27</td>
<td>VNY</td>
<td>143.572</td>
<td>CMA</td>
<td>239.8497</td>
</tr>
<tr>
<td>12:00:28</td>
<td>VNY</td>
<td>145.1883</td>
<td>CMA</td>
<td>238.4195</td>
</tr>
<tr>
<td>12:00:29</td>
<td>VNY</td>
<td>146.6833</td>
<td>CMA</td>
<td>237.0319</td>
</tr>
<tr>
<td>12:00:30</td>
<td>VNY</td>
<td>148.0684</td>
<td>CMA</td>
<td>235.6866</td>
</tr>
<tr>
<td>12:00:31</td>
<td>VNY</td>
<td>149.3538</td>
<td>CMA</td>
<td>234.383</td>
</tr>
<tr>
<td>12:00:32</td>
<td>VNY</td>
<td>150.5486</td>
<td>CMA</td>
<td>233.1206</td>
</tr>
<tr>
<td>12:00:33</td>
<td>VNY</td>
<td>151.6614</td>
<td>CMA</td>
<td>231.8986</td>
</tr>
<tr>
<td>12:00:34</td>
<td>VNY</td>
<td>152.6993</td>
<td>CMA</td>
<td>230.7161</td>
</tr>
<tr>
<td>12:00:35</td>
<td>VNY</td>
<td>153.6692</td>
<td>CMA</td>
<td>229.5722</td>
</tr>
<tr>
<td>12:00:36</td>
<td>VNY</td>
<td>154.577</td>
<td>CMA</td>
<td>228.466</td>
</tr>
<tr>
<td>12:00:37</td>
<td>VNY</td>
<td>155.4281</td>
<td>CMA</td>
<td>227.3963</td>
</tr>
<tr>
<td>12:00:38</td>
<td>VNY</td>
<td>156.2272</td>
<td>CMA</td>
<td>226.3622</td>
</tr>
<tr>
<td>Time</td>
<td>VNY</td>
<td>CMA</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>---------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>12:00:39</td>
<td>VNY</td>
<td>CMA</td>
<td>225.3625</td>
<td></td>
</tr>
<tr>
<td>12:00:40</td>
<td>VNY</td>
<td>CMA</td>
<td>224.396</td>
<td></td>
</tr>
<tr>
<td>12:00:41</td>
<td>VNY</td>
<td>CMA</td>
<td>223.4618</td>
<td></td>
</tr>
<tr>
<td>12:00:42</td>
<td>VNY</td>
<td>CMA</td>
<td>222.5586</td>
<td></td>
</tr>
<tr>
<td>12:00:43</td>
<td>VNY</td>
<td>CMA</td>
<td>221.6854</td>
<td></td>
</tr>
<tr>
<td>12:00:44</td>
<td>VNY</td>
<td>CMA</td>
<td>220.8411</td>
<td></td>
</tr>
<tr>
<td>12:00:45</td>
<td>VNY</td>
<td>CMA</td>
<td>220.0247</td>
<td></td>
</tr>
<tr>
<td>12:00:46</td>
<td>VNY</td>
<td>CMA</td>
<td>219.235</td>
<td></td>
</tr>
<tr>
<td>12:00:47</td>
<td>VNY</td>
<td>CMA</td>
<td>218.4711</td>
<td></td>
</tr>
<tr>
<td>12:00:48</td>
<td>VNY</td>
<td>CMA</td>
<td>217.7321</td>
<td></td>
</tr>
<tr>
<td>12:00:49</td>
<td>VNY</td>
<td>CMA</td>
<td>217.0168</td>
<td></td>
</tr>
<tr>
<td>12:00:50</td>
<td>VNY</td>
<td>CMA</td>
<td>216.3245</td>
<td></td>
</tr>
<tr>
<td>12:00:51</td>
<td>VNY</td>
<td>CMA</td>
<td>215.6542</td>
<td></td>
</tr>
<tr>
<td>12:00:52</td>
<td>VNY</td>
<td>CMA</td>
<td>215.0051</td>
<td></td>
</tr>
<tr>
<td>12:00:53</td>
<td>VNY</td>
<td>CMA</td>
<td>214.3764</td>
<td></td>
</tr>
<tr>
<td>12:00:54</td>
<td>VNY</td>
<td>CMA</td>
<td>213.7672</td>
<td></td>
</tr>
<tr>
<td>12:00:55</td>
<td>VNY</td>
<td>CMA</td>
<td>213.1767</td>
<td></td>
</tr>
<tr>
<td>12:00:56</td>
<td>VNY</td>
<td>CMA</td>
<td>212.6043</td>
<td></td>
</tr>
<tr>
<td>12:00:57</td>
<td>VNY</td>
<td>CMA</td>
<td>212.0493</td>
<td></td>
</tr>
<tr>
<td>12:00:58</td>
<td>VNY</td>
<td>CMA</td>
<td>211.5109</td>
<td></td>
</tr>
<tr>
<td>12:00:59</td>
<td>VNY</td>
<td>CMA</td>
<td>210.9885</td>
<td></td>
</tr>
<tr>
<td>12:01:00</td>
<td>VNY</td>
<td>CMA</td>
<td>210.4815</td>
<td></td>
</tr>
<tr>
<td>12:01:01</td>
<td>VNY</td>
<td>CMA</td>
<td>209.9893</td>
<td></td>
</tr>
<tr>
<td>12:01:02</td>
<td>VNY</td>
<td>CMA</td>
<td>209.5114</td>
<td></td>
</tr>
<tr>
<td>12:01:03</td>
<td>VNY</td>
<td>CMA</td>
<td>209.0471</td>
<td></td>
</tr>
<tr>
<td>12:01:04</td>
<td>VNY</td>
<td>CMA</td>
<td>208.596</td>
<td></td>
</tr>
<tr>
<td>12:01:05</td>
<td>VNY</td>
<td>CMA</td>
<td>208.1576</td>
<td></td>
</tr>
<tr>
<td>12:01:06</td>
<td>VNY</td>
<td>CMA</td>
<td>207.7313</td>
<td></td>
</tr>
<tr>
<td>12:01:07</td>
<td>VNY</td>
<td>CMA</td>
<td>207.3168</td>
<td></td>
</tr>
<tr>
<td>12:01:08</td>
<td>VNY</td>
<td>CMA</td>
<td>206.9136</td>
<td></td>
</tr>
<tr>
<td>12:01:09</td>
<td>VNY</td>
<td>CMA</td>
<td>206.5212</td>
<td></td>
</tr>
<tr>
<td>12:01:10</td>
<td>VNY</td>
<td>CMA</td>
<td>206.1394</td>
<td></td>
</tr>
<tr>
<td>12:01:11</td>
<td>VNY</td>
<td>CMA</td>
<td>205.7676</td>
<td></td>
</tr>
<tr>
<td>12:01:12</td>
<td>VNY</td>
<td>CMA</td>
<td>205.4056</td>
<td></td>
</tr>
<tr>
<td>12:01:13</td>
<td>VNY</td>
<td>CMA</td>
<td>205.0529</td>
<td></td>
</tr>
<tr>
<td>12:01:14</td>
<td>VNY</td>
<td>CMA</td>
<td>204.7093</td>
<td></td>
</tr>
<tr>
<td>12:01:15</td>
<td>VNY</td>
<td>CMA</td>
<td>204.3745</td>
<td></td>
</tr>
<tr>
<td>12:01:16</td>
<td>VNY</td>
<td>CMA</td>
<td>204.048</td>
<td></td>
</tr>
<tr>
<td>12:01:17</td>
<td>VNY</td>
<td>CMA</td>
<td>203.7297</td>
<td></td>
</tr>
<tr>
<td>12:01:18</td>
<td>VNY</td>
<td>CMA</td>
<td>203.4192</td>
<td></td>
</tr>
<tr>
<td>12:01:19</td>
<td>VNY</td>
<td>CMA</td>
<td>203.1163</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>VNY</td>
<td>Price</td>
<td>CMA</td>
<td>Price</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>--------</td>
<td>------</td>
<td>--------</td>
</tr>
<tr>
<td>12:01:20</td>
<td>VNY</td>
<td>170.1587</td>
<td>CMA</td>
<td>202.8208</td>
</tr>
<tr>
<td>12:01:21</td>
<td>VNY</td>
<td>170.2961</td>
<td>CMA</td>
<td>202.5323</td>
</tr>
<tr>
<td>12:01:22</td>
<td>VNY</td>
<td>170.4296</td>
<td>CMA</td>
<td>202.2506</td>
</tr>
<tr>
<td>12:01:23</td>
<td>VNY</td>
<td>170.5596</td>
<td>CMA</td>
<td>201.9756</td>
</tr>
<tr>
<td>12:01:24</td>
<td>VNY</td>
<td>170.6862</td>
<td>CMA</td>
<td>201.7069</td>
</tr>
<tr>
<td>12:01:25</td>
<td>VNY</td>
<td>170.8094</td>
<td>CMA</td>
<td>201.4445</td>
</tr>
<tr>
<td>12:01:26</td>
<td>VNY</td>
<td>170.9294</td>
<td>CMA</td>
<td>201.188</td>
</tr>
<tr>
<td>12:01:27</td>
<td>VNY</td>
<td>171.1604</td>
<td>CMA</td>
<td>199.9888</td>
</tr>
<tr>
<td>12:01:28</td>
<td>VNY</td>
<td>171.2715</td>
<td>CMA</td>
<td>200.6922</td>
</tr>
<tr>
<td>12:01:29</td>
<td>VNY</td>
<td>171.3799</td>
<td>CMA</td>
<td>200.4525</td>
</tr>
<tr>
<td>12:01:30</td>
<td>VNY</td>
<td>171.4857</td>
<td>CMA</td>
<td>199.2181</td>
</tr>
<tr>
<td>12:01:31</td>
<td>VNY</td>
<td>171.5889</td>
<td>CMA</td>
<td>198.9888</td>
</tr>
<tr>
<td>12:01:32</td>
<td>VNY</td>
<td>171.6897</td>
<td>CMA</td>
<td>198.7645</td>
</tr>
<tr>
<td>12:01:33</td>
<td>VNY</td>
<td>171.7881</td>
<td>CMA</td>
<td>198.5449</td>
</tr>
<tr>
<td>12:01:34</td>
<td>VNY</td>
<td>171.8842</td>
<td>CMA</td>
<td>198.33</td>
</tr>
<tr>
<td>12:01:35</td>
<td>VNY</td>
<td>171.9781</td>
<td>CMA</td>
<td>198.1195</td>
</tr>
<tr>
<td>12:01:36</td>
<td>VNY</td>
<td>172.0699</td>
<td>CMA</td>
<td>197.9134</td>
</tr>
<tr>
<td>12:01:37</td>
<td>VNY</td>
<td>172.1596</td>
<td>CMA</td>
<td>197.7116</td>
</tr>
<tr>
<td>12:01:38</td>
<td>VNY</td>
<td>172.2473</td>
<td>CMA</td>
<td>197.5139</td>
</tr>
<tr>
<td>12:01:39</td>
<td>VNY</td>
<td>172.3331</td>
<td>CMA</td>
<td>197.3202</td>
</tr>
<tr>
<td>12:01:40</td>
<td>VNY</td>
<td>172.417</td>
<td>CMA</td>
<td>197.1303</td>
</tr>
<tr>
<td>12:01:41</td>
<td>VNY</td>
<td>172.4991</td>
<td>CMA</td>
<td>197.9442</td>
</tr>
<tr>
<td>12:01:42</td>
<td>VNY</td>
<td>172.5795</td>
<td>CMA</td>
<td>197.7618</td>
</tr>
<tr>
<td>12:01:43</td>
<td>VNY</td>
<td>172.6582</td>
<td>CMA</td>
<td>197.583</td>
</tr>
<tr>
<td>12:01:44</td>
<td>VNY</td>
<td>172.7352</td>
<td>CMA</td>
<td>197.4076</td>
</tr>
<tr>
<td>12:01:45</td>
<td>VNY</td>
<td>172.8106</td>
<td>CMA</td>
<td>197.2355</td>
</tr>
<tr>
<td>12:01:46</td>
<td>VNY</td>
<td>172.8845</td>
<td>CMA</td>
<td>197.0667</td>
</tr>
<tr>
<td>12:01:47</td>
<td>VNY</td>
<td>172.9569</td>
<td>CMA</td>
<td>196.9011</td>
</tr>
<tr>
<td>12:01:48</td>
<td>VNY</td>
<td>173.0278</td>
<td>CMA</td>
<td>196.7386</td>
</tr>
<tr>
<td>12:01:49</td>
<td>VNY</td>
<td>173.0974</td>
<td>CMA</td>
<td>196.5791</td>
</tr>
<tr>
<td>12:01:50</td>
<td>VNY</td>
<td>173.1655</td>
<td>CMA</td>
<td>196.4226</td>
</tr>
<tr>
<td>12:01:51</td>
<td>VNY</td>
<td>173.2324</td>
<td>CMA</td>
<td>196.2688</td>
</tr>
<tr>
<td>12:01:52</td>
<td>VNY</td>
<td>173.2979</td>
<td>CMA</td>
<td>196.1179</td>
</tr>
<tr>
<td>12:01:53</td>
<td>VNY</td>
<td>173.3622</td>
<td>CMA</td>
<td>195.9696</td>
</tr>
<tr>
<td>12:01:54</td>
<td>VNY</td>
<td>173.4253</td>
<td>CMA</td>
<td>195.824</td>
</tr>
<tr>
<td>12:01:55</td>
<td>VNY</td>
<td>173.4872</td>
<td>CMA</td>
<td>195.681</td>
</tr>
<tr>
<td>12:01:56</td>
<td>VNY</td>
<td>173.5479</td>
<td>CMA</td>
<td>195.5405</td>
</tr>
<tr>
<td>12:01:57</td>
<td>VNY</td>
<td>173.6076</td>
<td>CMA</td>
<td>195.4024</td>
</tr>
<tr>
<td>12:01:58</td>
<td>VNY</td>
<td>173.6661</td>
<td>CMA</td>
<td>195.2666</td>
</tr>
<tr>
<td>12:01:59</td>
<td>VNY</td>
<td>173.7236</td>
<td>CMA</td>
<td>195.1332</td>
</tr>
<tr>
<td>12:02:00</td>
<td>VNY</td>
<td>173.7236</td>
<td>CMA</td>
<td>195.0021</td>
</tr>
<tr>
<td>Time</td>
<td>Symbol</td>
<td>Price</td>
<td>Symbol</td>
<td>Price</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>12:02:01</td>
<td>VNY</td>
<td>173.78</td>
<td>CMA</td>
<td>194.8731</td>
</tr>
<tr>
<td>12:02:02</td>
<td>VNY</td>
<td>173.8355</td>
<td>CMA</td>
<td>194.7463</td>
</tr>
<tr>
<td>12:02:03</td>
<td>VNY</td>
<td>173.89</td>
<td>CMA</td>
<td>194.6216</td>
</tr>
<tr>
<td>12:02:04</td>
<td>VNY</td>
<td>173.9435</td>
<td>CMA</td>
<td>194.499</td>
</tr>
<tr>
<td>12:02:05</td>
<td>VNY</td>
<td>173.9961</td>
<td>CMA</td>
<td>194.3783</td>
</tr>
<tr>
<td>12:02:06</td>
<td>VNY</td>
<td>174.0478</td>
<td>CMA</td>
<td>194.2596</td>
</tr>
<tr>
<td>12:02:07</td>
<td>VNY</td>
<td>174.0986</td>
<td>CMA</td>
<td>194.1428</td>
</tr>
<tr>
<td>12:02:08</td>
<td>VNY</td>
<td>174.1486</td>
<td>CMA</td>
<td>194.0279</td>
</tr>
<tr>
<td>12:02:09</td>
<td>VNY</td>
<td>174.1977</td>
<td>CMA</td>
<td>193.9147</td>
</tr>
<tr>
<td>12:02:10</td>
<td>VNY</td>
<td>174.246</td>
<td>CMA</td>
<td>193.8034</td>
</tr>
<tr>
<td>12:02:11</td>
<td>VNY</td>
<td>174.2935</td>
<td>CMA</td>
<td>193.6938</td>
</tr>
<tr>
<td>12:02:12</td>
<td>VNY</td>
<td>174.3402</td>
<td>CMA</td>
<td>193.5859</td>
</tr>
<tr>
<td>12:02:13</td>
<td>VNY</td>
<td>174.3862</td>
<td>CMA</td>
<td>193.4796</td>
</tr>
<tr>
<td>12:02:14</td>
<td>VNY</td>
<td>174.4314</td>
<td>CMA</td>
<td>193.375</td>
</tr>
<tr>
<td>12:02:15</td>
<td>VNY</td>
<td>174.476</td>
<td>CMA</td>
<td>193.2719</td>
</tr>
<tr>
<td>12:02:16</td>
<td>VNY</td>
<td>174.5198</td>
<td>CMA</td>
<td>193.1704</td>
</tr>
<tr>
<td>12:02:17</td>
<td>VNY</td>
<td>174.5629</td>
<td>CMA</td>
<td>193.0704</td>
</tr>
<tr>
<td>12:02:18</td>
<td>VNY</td>
<td>174.6053</td>
<td>CMA</td>
<td>192.9719</td>
</tr>
<tr>
<td>12:02:19</td>
<td>VNY</td>
<td>174.6471</td>
<td>CMA</td>
<td>192.8748</td>
</tr>
<tr>
<td>12:02:20</td>
<td>VNY</td>
<td>174.6883</td>
<td>CMA</td>
<td>192.7792</td>
</tr>
<tr>
<td>12:02:21</td>
<td>VNY</td>
<td>174.7288</td>
<td>CMA</td>
<td>192.6849</td>
</tr>
<tr>
<td>12:02:22</td>
<td>VNY</td>
<td>174.7687</td>
<td>CMA</td>
<td>192.592</td>
</tr>
<tr>
<td>12:02:23</td>
<td>VNY</td>
<td>174.8081</td>
<td>CMA</td>
<td>192.5005</td>
</tr>
<tr>
<td>12:02:24</td>
<td>VNY</td>
<td>174.8468</td>
<td>CMA</td>
<td>192.4102</td>
</tr>
<tr>
<td>12:02:25</td>
<td>VNY</td>
<td>174.885</td>
<td>CMA</td>
<td>192.3212</td>
</tr>
<tr>
<td>12:02:26</td>
<td>VNY</td>
<td>174.9226</td>
<td>CMA</td>
<td>192.2335</td>
</tr>
<tr>
<td>12:02:27</td>
<td>VNY</td>
<td>174.9596</td>
<td>CMA</td>
<td>192.1469</td>
</tr>
<tr>
<td>12:02:28</td>
<td>VNY</td>
<td>174.9961</td>
<td>CMA</td>
<td>192.0616</td>
</tr>
<tr>
<td>12:02:29</td>
<td>VNY</td>
<td>175.0321</td>
<td>CMA</td>
<td>191.9775</td>
</tr>
<tr>
<td>12:02:30</td>
<td>VNY</td>
<td>175.0676</td>
<td>CMA</td>
<td>191.8945</td>
</tr>
<tr>
<td>12:02:31</td>
<td>VNY</td>
<td>175.1026</td>
<td>CMA</td>
<td>191.8126</td>
</tr>
<tr>
<td>12:02:32</td>
<td>VNY</td>
<td>175.1371</td>
<td>CMA</td>
<td>191.7318</td>
</tr>
<tr>
<td>12:02:33</td>
<td>VNY</td>
<td>175.1711</td>
<td>CMA</td>
<td>191.6521</td>
</tr>
<tr>
<td>12:02:34</td>
<td>VNY</td>
<td>175.2046</td>
<td>CMA</td>
<td>191.5735</td>
</tr>
<tr>
<td>12:02:35</td>
<td>VNY</td>
<td>175.2377</td>
<td>CMA</td>
<td>191.4959</td>
</tr>
<tr>
<td>12:02:36</td>
<td>VNY</td>
<td>175.2703</td>
<td>CMA</td>
<td>191.4194</td>
</tr>
<tr>
<td>12:02:37</td>
<td>VNY</td>
<td>175.3025</td>
<td>CMA</td>
<td>191.3438</td>
</tr>
<tr>
<td>12:02:38</td>
<td>VNY</td>
<td>175.3342</td>
<td>CMA</td>
<td>191.2692</td>
</tr>
<tr>
<td>12:02:39</td>
<td>VNY</td>
<td>175.3655</td>
<td>CMA</td>
<td>191.1956</td>
</tr>
<tr>
<td>12:02:40</td>
<td>VNY</td>
<td>175.3964</td>
<td>CMA</td>
<td>191.1229</td>
</tr>
<tr>
<td>12:02:41</td>
<td>VNY</td>
<td>175.4269</td>
<td>CMA</td>
<td>191.0512</td>
</tr>
<tr>
<td>Time</td>
<td>Currency</td>
<td>Bid</td>
<td>Ask</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>12:03:23</td>
<td>VNY</td>
<td>176.4225</td>
<td>188.6906</td>
<td></td>
</tr>
<tr>
<td>12:03:24</td>
<td>VNY</td>
<td>176.4409</td>
<td>188.6465</td>
<td></td>
</tr>
<tr>
<td>12:03:25</td>
<td>VNY</td>
<td>176.4592</td>
<td>188.6029</td>
<td></td>
</tr>
<tr>
<td>12:03:26</td>
<td>VNY</td>
<td>176.4773</td>
<td>188.5597</td>
<td></td>
</tr>
<tr>
<td>12:03:27</td>
<td>VNY</td>
<td>176.4952</td>
<td>188.5169</td>
<td></td>
</tr>
<tr>
<td>12:03:28</td>
<td>VNY</td>
<td>176.5129</td>
<td>188.4746</td>
<td></td>
</tr>
<tr>
<td>12:03:29</td>
<td>VNY</td>
<td>176.5304</td>
<td>188.4327</td>
<td></td>
</tr>
<tr>
<td>12:03:30</td>
<td>VNY</td>
<td>176.5478</td>
<td>188.3912</td>
<td></td>
</tr>
<tr>
<td>12:03:31</td>
<td>VNY</td>
<td>176.5649</td>
<td>188.3501</td>
<td></td>
</tr>
<tr>
<td>12:03:32</td>
<td>VNY</td>
<td>176.582</td>
<td>188.3094</td>
<td></td>
</tr>
<tr>
<td>12:03:33</td>
<td>VNY</td>
<td>176.5988</td>
<td>188.2691</td>
<td></td>
</tr>
<tr>
<td>12:03:34</td>
<td>VNY</td>
<td>176.6155</td>
<td>188.2291</td>
<td></td>
</tr>
<tr>
<td>12:03:35</td>
<td>VNY</td>
<td>176.632</td>
<td>188.1896</td>
<td></td>
</tr>
<tr>
<td>12:03:36</td>
<td>VNY</td>
<td>176.6484</td>
<td>188.1504</td>
<td></td>
</tr>
<tr>
<td>12:03:37</td>
<td>VNY</td>
<td>176.6646</td>
<td>188.1116</td>
<td></td>
</tr>
<tr>
<td>12:03:38</td>
<td>VNY</td>
<td>176.6806</td>
<td>188.0732</td>
<td></td>
</tr>
<tr>
<td>12:03:39</td>
<td>VNY</td>
<td>176.6965</td>
<td>188.0351</td>
<td></td>
</tr>
<tr>
<td>12:03:40</td>
<td>VNY</td>
<td>176.7122</td>
<td>187.9974</td>
<td></td>
</tr>
<tr>
<td>12:03:41</td>
<td>VNY</td>
<td>176.7278</td>
<td>187.96</td>
<td></td>
</tr>
<tr>
<td>12:03:42</td>
<td>VNY</td>
<td>176.7433</td>
<td>187.923</td>
<td></td>
</tr>
<tr>
<td>12:03:43</td>
<td>VNY</td>
<td>176.7586</td>
<td>187.8863</td>
<td></td>
</tr>
<tr>
<td>12:03:44</td>
<td>VNY</td>
<td>176.7737</td>
<td>187.85</td>
<td></td>
</tr>
<tr>
<td>12:03:45</td>
<td>VNY</td>
<td>176.7887</td>
<td>187.814</td>
<td></td>
</tr>
<tr>
<td>12:03:46</td>
<td>VNY</td>
<td>176.8036</td>
<td>187.7783</td>
<td></td>
</tr>
<tr>
<td>12:03:47</td>
<td>VNY</td>
<td>176.8184</td>
<td>187.7429</td>
<td></td>
</tr>
<tr>
<td>12:03:48</td>
<td>VNY</td>
<td>176.833</td>
<td>187.7079</td>
<td></td>
</tr>
<tr>
<td>12:03:49</td>
<td>VNY</td>
<td>176.8474</td>
<td>187.6732</td>
<td></td>
</tr>
<tr>
<td>12:03:50</td>
<td>VNY</td>
<td>176.8618</td>
<td>187.6388</td>
<td></td>
</tr>
<tr>
<td>12:03:51</td>
<td>VNY</td>
<td>176.876</td>
<td>187.6046</td>
<td></td>
</tr>
<tr>
<td>12:03:52</td>
<td>VNY</td>
<td>176.8901</td>
<td>187.5708</td>
<td></td>
</tr>
<tr>
<td>12:03:53</td>
<td>VNY</td>
<td>176.904</td>
<td>187.5373</td>
<td></td>
</tr>
<tr>
<td>12:03:54</td>
<td>VNY</td>
<td>176.9178</td>
<td>187.5041</td>
<td></td>
</tr>
<tr>
<td>12:03:55</td>
<td>VNY</td>
<td>176.9315</td>
<td>187.4712</td>
<td></td>
</tr>
<tr>
<td>12:03:56</td>
<td>VNY</td>
<td>176.9451</td>
<td>187.4385</td>
<td></td>
</tr>
<tr>
<td>12:03:57</td>
<td>VNY</td>
<td>176.9586</td>
<td>187.4062</td>
<td></td>
</tr>
<tr>
<td>12:03:58</td>
<td>VNY</td>
<td>176.9719</td>
<td>187.3741</td>
<td></td>
</tr>
<tr>
<td>12:03:59</td>
<td>VNY</td>
<td>176.9852</td>
<td>187.3423</td>
<td></td>
</tr>
<tr>
<td>12:04:00</td>
<td>VNY</td>
<td>176.9983</td>
<td>187.3108</td>
<td></td>
</tr>
<tr>
<td>12:04:01</td>
<td>VNY</td>
<td>177.0113</td>
<td>187.2795</td>
<td></td>
</tr>
<tr>
<td>12:04:02</td>
<td>VNY</td>
<td>177.0242</td>
<td>187.2485</td>
<td></td>
</tr>
<tr>
<td>12:04:03</td>
<td>VNY</td>
<td>177.037</td>
<td>187.2178</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>Symbol</td>
<td>Mid-Price</td>
<td>Type</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>-----------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>12:04:04</td>
<td>VNY</td>
<td>177.0496</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:05</td>
<td>VNY</td>
<td>177.0622</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:06</td>
<td>VNY</td>
<td>177.0746</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:07</td>
<td>VNY</td>
<td>177.087</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:08</td>
<td>VNY</td>
<td>177.0992</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:09</td>
<td>VNY</td>
<td>177.1114</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:10</td>
<td>VNY</td>
<td>177.1234</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:11</td>
<td>VNY</td>
<td>177.1354</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:12</td>
<td>VNY</td>
<td>177.1472</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:13</td>
<td>VNY</td>
<td>177.1589</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:14</td>
<td>VNY</td>
<td>177.1706</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:15</td>
<td>VNY</td>
<td>177.1822</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:16</td>
<td>VNY</td>
<td>177.1936</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:17</td>
<td>VNY</td>
<td>177.205</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:18</td>
<td>VNY</td>
<td>177.2163</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:19</td>
<td>VNY</td>
<td>177.2274</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:20</td>
<td>VNY</td>
<td>177.2385</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:21</td>
<td>VNY</td>
<td>177.2495</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:22</td>
<td>VNY</td>
<td>177.2605</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:23</td>
<td>VNY</td>
<td>177.2713</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:24</td>
<td>VNY</td>
<td>177.282</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:25</td>
<td>VNY</td>
<td>177.2927</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:26</td>
<td>VNY</td>
<td>177.3033</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:27</td>
<td>VNY</td>
<td>177.3138</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:28</td>
<td>VNY</td>
<td>177.3242</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:29</td>
<td>VNY</td>
<td>177.3345</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:30</td>
<td>VNY</td>
<td>177.3448</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:31</td>
<td>VNY</td>
<td>177.355</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:32</td>
<td>VNY</td>
<td>177.3651</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:33</td>
<td>VNY</td>
<td>177.3751</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:34</td>
<td>VNY</td>
<td>177.385</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:35</td>
<td>VNY</td>
<td>177.3949</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:36</td>
<td>VNY</td>
<td>177.4047</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:37</td>
<td>VNY</td>
<td>177.4144</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:38</td>
<td>VNY</td>
<td>177.4241</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:39</td>
<td>VNY</td>
<td>177.4336</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:40</td>
<td>VNY</td>
<td>177.4432</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:41</td>
<td>VNY</td>
<td>177.4526</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:42</td>
<td>VNY</td>
<td>177.462</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:43</td>
<td>VNY</td>
<td>177.4713</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:04:44</td>
<td>VNY</td>
<td>177.4805</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>VNY</td>
<td>CNY</td>
<td>Rate</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
<td>------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>12:04:45</td>
<td>VNY</td>
<td>177.4897</td>
<td>CMA</td>
<td>186.126</td>
</tr>
<tr>
<td>12:04:46</td>
<td>VNY</td>
<td>177.4987</td>
<td>CMA</td>
<td>186.1041</td>
</tr>
<tr>
<td>12:04:47</td>
<td>VNY</td>
<td>177.5078</td>
<td>CMA</td>
<td>186.0822</td>
</tr>
<tr>
<td>12:04:48</td>
<td>VNY</td>
<td>177.5167</td>
<td>CMA</td>
<td>186.0605</td>
</tr>
<tr>
<td>12:04:49</td>
<td>VNY</td>
<td>177.5256</td>
<td>CMA</td>
<td>186.039</td>
</tr>
<tr>
<td>12:04:50</td>
<td>VNY</td>
<td>177.5345</td>
<td>CMA</td>
<td>186.0176</td>
</tr>
<tr>
<td>12:04:51</td>
<td>VNY</td>
<td>177.5433</td>
<td>CMA</td>
<td>185.9964</td>
</tr>
<tr>
<td>12:04:52</td>
<td>VNY</td>
<td>177.552</td>
<td>CMA</td>
<td>185.9753</td>
</tr>
<tr>
<td>12:04:53</td>
<td>VNY</td>
<td>177.5606</td>
<td>CMA</td>
<td>185.9544</td>
</tr>
<tr>
<td>12:04:54</td>
<td>VNY</td>
<td>177.5692</td>
<td>CMA</td>
<td>185.9336</td>
</tr>
<tr>
<td>12:04:55</td>
<td>VNY</td>
<td>177.5777</td>
<td>CMA</td>
<td>185.913</td>
</tr>
<tr>
<td>12:04:56</td>
<td>VNY</td>
<td>177.5862</td>
<td>CMA</td>
<td>185.8925</td>
</tr>
<tr>
<td>12:04:57</td>
<td>VNY</td>
<td>177.5946</td>
<td>CMA</td>
<td>185.8721</td>
</tr>
<tr>
<td>12:04:58</td>
<td>VNY</td>
<td>177.603</td>
<td>CMA</td>
<td>185.8519</td>
</tr>
<tr>
<td>12:04:59</td>
<td>VNY</td>
<td>177.6113</td>
<td>CMA</td>
<td>185.8319</td>
</tr>
<tr>
<td>12:05:00</td>
<td>VNY</td>
<td>177.6195</td>
<td>CMA</td>
<td>185.8119</td>
</tr>
<tr>
<td>12:05:01</td>
<td>VNY</td>
<td>177.6277</td>
<td>CMA</td>
<td>185.7921</td>
</tr>
<tr>
<td>12:05:02</td>
<td>VNY</td>
<td>177.6358</td>
<td>CMA</td>
<td>185.7725</td>
</tr>
<tr>
<td>12:05:03</td>
<td>VNY</td>
<td>177.6439</td>
<td>CMA</td>
<td>185.7529</td>
</tr>
<tr>
<td>12:05:04</td>
<td>VNY</td>
<td>177.6519</td>
<td>CMA</td>
<td>185.7335</td>
</tr>
<tr>
<td>12:05:05</td>
<td>VNY</td>
<td>177.6599</td>
<td>CMA</td>
<td>185.7142</td>
</tr>
<tr>
<td>12:05:06</td>
<td>VNY</td>
<td>177.6678</td>
<td>CMA</td>
<td>185.6951</td>
</tr>
<tr>
<td>12:05:07</td>
<td>VNY</td>
<td>177.6756</td>
<td>CMA</td>
<td>185.6761</td>
</tr>
<tr>
<td>12:05:08</td>
<td>VNY</td>
<td>177.6834</td>
<td>CMA</td>
<td>185.6572</td>
</tr>
<tr>
<td>12:05:09</td>
<td>VNY</td>
<td>177.6912</td>
<td>CMA</td>
<td>185.6384</td>
</tr>
<tr>
<td>12:05:10</td>
<td>VNY</td>
<td>177.6989</td>
<td>CMA</td>
<td>185.6198</td>
</tr>
<tr>
<td>12:05:11</td>
<td>VNY</td>
<td>177.7065</td>
<td>CMA</td>
<td>185.6013</td>
</tr>
<tr>
<td>12:05:12</td>
<td>VNY</td>
<td>177.7141</td>
<td>CMA</td>
<td>185.5829</td>
</tr>
<tr>
<td>12:05:13</td>
<td>VNY</td>
<td>177.7217</td>
<td>CMA</td>
<td>185.5646</td>
</tr>
<tr>
<td>12:05:14</td>
<td>VNY</td>
<td>177.7291</td>
<td>CMA</td>
<td>185.5464</td>
</tr>
<tr>
<td>12:05:15</td>
<td>VNY</td>
<td>177.7366</td>
<td>CMA</td>
<td>185.5284</td>
</tr>
<tr>
<td>12:05:16</td>
<td>VNY</td>
<td>177.744</td>
<td>CMA</td>
<td>185.5105</td>
</tr>
<tr>
<td>12:05:17</td>
<td>VNY</td>
<td>177.7513</td>
<td>CMA</td>
<td>185.4927</td>
</tr>
<tr>
<td>12:05:18</td>
<td>VNY</td>
<td>177.7586</td>
<td>CMA</td>
<td>185.475</td>
</tr>
<tr>
<td>12:05:19</td>
<td>VNY</td>
<td>177.7659</td>
<td>CMA</td>
<td>185.4574</td>
</tr>
<tr>
<td>12:05:20</td>
<td>VNY</td>
<td>177.7731</td>
<td>CMA</td>
<td>185.4399</td>
</tr>
<tr>
<td>12:05:21</td>
<td>VNY</td>
<td>177.7803</td>
<td>CMA</td>
<td>185.4226</td>
</tr>
<tr>
<td>12:05:22</td>
<td>VNY</td>
<td>177.7874</td>
<td>CMA</td>
<td>185.4053</td>
</tr>
<tr>
<td>12:05:23</td>
<td>VNY</td>
<td>177.7944</td>
<td>CMA</td>
<td>185.3882</td>
</tr>
<tr>
<td>12:05:24</td>
<td>VNY</td>
<td>177.8015</td>
<td>CMA</td>
<td>185.3712</td>
</tr>
<tr>
<td>12:05:25</td>
<td>VNY</td>
<td>177.8084</td>
<td>CMA</td>
<td>185.3543</td>
</tr>
<tr>
<td>Time</td>
<td>VNY</td>
<td>CNY</td>
<td>Time</td>
<td>VNY</td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>-----</td>
</tr>
<tr>
<td>12:05:26</td>
<td>VNY</td>
<td>177.8154</td>
<td>CMA</td>
<td>185.3374</td>
</tr>
<tr>
<td>12:05:27</td>
<td>VNY</td>
<td>177.8223</td>
<td>CMA</td>
<td>185.3207</td>
</tr>
<tr>
<td>12:05:28</td>
<td>VNY</td>
<td>177.8291</td>
<td>CMA</td>
<td>185.3041</td>
</tr>
<tr>
<td>12:05:29</td>
<td>VNY</td>
<td>177.8359</td>
<td>CMA</td>
<td>185.2876</td>
</tr>
<tr>
<td>12:05:30</td>
<td>VNY</td>
<td>177.8427</td>
<td>CMA</td>
<td>185.2712</td>
</tr>
<tr>
<td>12:05:31</td>
<td>VNY</td>
<td>177.8494</td>
<td>CMA</td>
<td>185.2549</td>
</tr>
<tr>
<td>12:05:32</td>
<td>VNY</td>
<td>177.8561</td>
<td>CMA</td>
<td>185.2387</td>
</tr>
<tr>
<td>12:05:33</td>
<td>VNY</td>
<td>177.8627</td>
<td>CMA</td>
<td>185.2226</td>
</tr>
<tr>
<td>12:05:34</td>
<td>VNY</td>
<td>177.8693</td>
<td>CMA</td>
<td>185.2066</td>
</tr>
<tr>
<td>12:05:35</td>
<td>VNY</td>
<td>177.8759</td>
<td>CMA</td>
<td>185.1907</td>
</tr>
<tr>
<td>12:05:36</td>
<td>VNY</td>
<td>177.8824</td>
<td>CMA</td>
<td>185.1749</td>
</tr>
<tr>
<td>12:05:37</td>
<td>VNY</td>
<td>177.8888</td>
<td>CMA</td>
<td>185.1592</td>
</tr>
<tr>
<td>12:05:38</td>
<td>VNY</td>
<td>177.8953</td>
<td>CMA</td>
<td>185.1436</td>
</tr>
<tr>
<td>12:05:39</td>
<td>VNY</td>
<td>177.9017</td>
<td>CMA</td>
<td>185.1281</td>
</tr>
<tr>
<td>12:05:40</td>
<td>VNY</td>
<td>177.9080</td>
<td>CMA</td>
<td>185.1127</td>
</tr>
<tr>
<td>12:05:41</td>
<td>VNY</td>
<td>177.9144</td>
<td>CMA</td>
<td>185.0973</td>
</tr>
<tr>
<td>12:05:42</td>
<td>VNY</td>
<td>177.9206</td>
<td>CMA</td>
<td>185.0821</td>
</tr>
<tr>
<td>12:05:43</td>
<td>VNY</td>
<td>177.9269</td>
<td>CMA</td>
<td>185.0669</td>
</tr>
<tr>
<td>12:05:44</td>
<td>VNY</td>
<td>177.9331</td>
<td>CMA</td>
<td>185.0519</td>
</tr>
<tr>
<td>12:05:45</td>
<td>VNY</td>
<td>177.9392</td>
<td>CMA</td>
<td>185.0369</td>
</tr>
<tr>
<td>12:05:46</td>
<td>VNY</td>
<td>177.9454</td>
<td>CMA</td>
<td>185.0220</td>
</tr>
<tr>
<td>12:05:47</td>
<td>VNY</td>
<td>177.9515</td>
<td>CMA</td>
<td>185.0072</td>
</tr>
<tr>
<td>12:05:48</td>
<td>VNY</td>
<td>177.9575</td>
<td>CMA</td>
<td>184.9925</td>
</tr>
<tr>
<td>12:05:49</td>
<td>VNY</td>
<td>177.9636</td>
<td>CMA</td>
<td>184.9779</td>
</tr>
<tr>
<td>12:05:50</td>
<td>VNY</td>
<td>177.9695</td>
<td>CMA</td>
<td>184.9634</td>
</tr>
<tr>
<td>12:05:51</td>
<td>VNY</td>
<td>177.9755</td>
<td>CMA</td>
<td>184.9489</td>
</tr>
<tr>
<td>12:05:52</td>
<td>VNY</td>
<td>177.9814</td>
<td>CMA</td>
<td>184.9345</td>
</tr>
<tr>
<td>12:05:53</td>
<td>VNY</td>
<td>177.9873</td>
<td>CMA</td>
<td>184.9203</td>
</tr>
<tr>
<td>12:05:54</td>
<td>VNY</td>
<td>177.9931</td>
<td>CMA</td>
<td>184.9063</td>
</tr>
<tr>
<td>12:05:55</td>
<td>VNY</td>
<td>177.9999</td>
<td>CMA</td>
<td>184.8919</td>
</tr>
<tr>
<td>12:05:56</td>
<td>VNY</td>
<td>178.0047</td>
<td>CMA</td>
<td>184.8779</td>
</tr>
<tr>
<td>12:05:57</td>
<td>VNY</td>
<td>178.0105</td>
<td>CMA</td>
<td>184.8639</td>
</tr>
<tr>
<td>12:05:58</td>
<td>VNY</td>
<td>178.0162</td>
<td>CMA</td>
<td>184.8500</td>
</tr>
<tr>
<td>12:05:59</td>
<td>VNY</td>
<td>178.0219</td>
<td>CMA</td>
<td>184.8362</td>
</tr>
<tr>
<td>12:06:00</td>
<td>VNY</td>
<td>178.0275</td>
<td>CMA</td>
<td>184.8225</td>
</tr>
<tr>
<td>12:06:01</td>
<td>VNY</td>
<td>178.0331</td>
<td>CMA</td>
<td>184.8089</td>
</tr>
<tr>
<td>12:06:02</td>
<td>VNY</td>
<td>178.0387</td>
<td>CMA</td>
<td>184.7953</td>
</tr>
<tr>
<td>12:06:03</td>
<td>VNY</td>
<td>178.0443</td>
<td>CMA</td>
<td>184.7818</td>
</tr>
<tr>
<td>12:06:04</td>
<td>VNY</td>
<td>178.0498</td>
<td>CMA</td>
<td>184.7684</td>
</tr>
<tr>
<td>12:06:05</td>
<td>VNY</td>
<td>178.0553</td>
<td>CMA</td>
<td>184.7551</td>
</tr>
<tr>
<td>12:06:06</td>
<td>VNY</td>
<td>178.0608</td>
<td>CMA</td>
<td>184.7418</td>
</tr>
<tr>
<td>Time</td>
<td>VNY</td>
<td>CMA</td>
<td>Time</td>
<td>VNY</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>--------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>12:06:07</td>
<td>178.0662</td>
<td>184.7286</td>
<td>12:06:08</td>
<td>178.0716</td>
</tr>
<tr>
<td>12:06:09</td>
<td>178.0769</td>
<td>184.7024</td>
<td>12:06:10</td>
<td>178.0823</td>
</tr>
<tr>
<td>12:06:11</td>
<td>178.0876</td>
<td>184.6766</td>
<td>12:06:12</td>
<td>178.0929</td>
</tr>
<tr>
<td>12:06:17</td>
<td>178.1188</td>
<td>184.5636</td>
<td>12:06:18</td>
<td>178.1239</td>
</tr>
<tr>
<td>12:06:19</td>
<td>178.129</td>
<td>184.4912</td>
<td>12:06:20</td>
<td>178.1341</td>
</tr>
<tr>
<td>12:06:23</td>
<td>178.1491</td>
<td>184.4442</td>
<td>12:06:24</td>
<td>178.154</td>
</tr>
<tr>
<td>12:06:27</td>
<td>178.1687</td>
<td>184.3982</td>
<td>12:06:28</td>
<td>178.1735</td>
</tr>
<tr>
<td>12:06:29</td>
<td>178.1784</td>
<td>184.3756</td>
<td>12:06:30</td>
<td>178.2114</td>
</tr>
<tr>
<td>12:06:33</td>
<td>178.2251</td>
<td>184.331</td>
<td>12:06:34</td>
<td>178.2297</td>
</tr>
<tr>
<td>12:06:35</td>
<td>178.2342</td>
<td>184.3091</td>
<td>12:06:36</td>
<td>178.2387</td>
</tr>
<tr>
<td>12:06:37</td>
<td>178.2432</td>
<td>184.2873</td>
<td>12:06:38</td>
<td>178.2476</td>
</tr>
<tr>
<td>12:06:39</td>
<td>178.2521</td>
<td>184.2658</td>
<td>12:06:40</td>
<td>178.2565</td>
</tr>
<tr>
<td>12:06:41</td>
<td>178.2609</td>
<td>184.2551</td>
<td>12:06:42</td>
<td>178.2665</td>
</tr>
</tbody>
</table>

253
<table>
<thead>
<tr>
<th>Time</th>
<th>VNY</th>
<th>CMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:06:48</td>
<td>178.2652</td>
<td>184.2445</td>
</tr>
<tr>
<td>12:06:49</td>
<td>178.2696</td>
<td>184.2339</td>
</tr>
<tr>
<td>12:06:50</td>
<td>178.2739</td>
<td>184.2234</td>
</tr>
<tr>
<td>12:06:51</td>
<td>178.2782</td>
<td>184.2129</td>
</tr>
<tr>
<td>12:06:52</td>
<td>178.2825</td>
<td>184.2025</td>
</tr>
<tr>
<td>12:06:53</td>
<td>178.2867</td>
<td>184.1922</td>
</tr>
<tr>
<td>12:06:54</td>
<td>178.291</td>
<td>184.1819</td>
</tr>
<tr>
<td>12:06:55</td>
<td>178.2952</td>
<td>184.1716</td>
</tr>
<tr>
<td>12:06:56</td>
<td>178.2994</td>
<td>184.1614</td>
</tr>
<tr>
<td>12:06:57</td>
<td>178.3035</td>
<td>184.1512</td>
</tr>
<tr>
<td>12:06:58</td>
<td>178.3077</td>
<td>184.1411</td>
</tr>
<tr>
<td>12:06:59</td>
<td>178.3118</td>
<td>184.1311</td>
</tr>
<tr>
<td>12:07:00</td>
<td>178.3159</td>
<td>184.121</td>
</tr>
<tr>
<td>12:07:01</td>
<td>178.32</td>
<td>184.1111</td>
</tr>
<tr>
<td>12:07:02</td>
<td>178.3241</td>
<td>184.1012</td>
</tr>
<tr>
<td>12:07:03</td>
<td>178.3282</td>
<td>184.0913</td>
</tr>
<tr>
<td>12:07:04</td>
<td>178.3322</td>
<td>184.0815</td>
</tr>
<tr>
<td>12:07:05</td>
<td>178.3362</td>
<td>184.0717</td>
</tr>
<tr>
<td>12:07:06</td>
<td>178.3402</td>
<td>184.062</td>
</tr>
<tr>
<td>12:07:07</td>
<td>178.3442</td>
<td>184.0523</td>
</tr>
<tr>
<td>12:07:08</td>
<td>178.3481</td>
<td>184.0427</td>
</tr>
<tr>
<td>12:07:09</td>
<td>178.3521</td>
<td>184.0331</td>
</tr>
<tr>
<td>12:07:10</td>
<td>178.356</td>
<td>184.0235</td>
</tr>
<tr>
<td>12:07:11</td>
<td>178.3599</td>
<td>184.014</td>
</tr>
<tr>
<td>12:07:12</td>
<td>178.3638</td>
<td>184.0046</td>
</tr>
<tr>
<td>12:07:13</td>
<td>178.3676</td>
<td>183.9952</td>
</tr>
<tr>
<td>12:07:14</td>
<td>178.3715</td>
<td>183.9852</td>
</tr>
<tr>
<td>12:07:15</td>
<td>178.3753</td>
<td>183.9765</td>
</tr>
<tr>
<td>12:07:16</td>
<td>178.3791</td>
<td>183.9672</td>
</tr>
<tr>
<td>12:07:17</td>
<td>178.3829</td>
<td>183.958</td>
</tr>
<tr>
<td>12:07:18</td>
<td>178.3867</td>
<td>183.9488</td>
</tr>
<tr>
<td>12:07:19</td>
<td>178.3904</td>
<td>183.9396</td>
</tr>
<tr>
<td>12:07:20</td>
<td>178.3942</td>
<td>183.9305</td>
</tr>
<tr>
<td>12:07:21</td>
<td>178.3979</td>
<td>183.9215</td>
</tr>
<tr>
<td>12:07:22</td>
<td>178.4016</td>
<td>183.9124</td>
</tr>
<tr>
<td>12:07:23</td>
<td>178.4053</td>
<td>183.9035</td>
</tr>
<tr>
<td>12:07:24</td>
<td>178.4089</td>
<td>183.8945</td>
</tr>
<tr>
<td>12:07:25</td>
<td>178.4126</td>
<td>183.8856</td>
</tr>
<tr>
<td>12:07:26</td>
<td>178.4162</td>
<td>183.8768</td>
</tr>
<tr>
<td>12:07:27</td>
<td>178.4198</td>
<td>183.868</td>
</tr>
<tr>
<td>12:07:28</td>
<td>178.4234</td>
<td>183.8592</td>
</tr>
<tr>
<td>Time</td>
<td>VNY</td>
<td>CMA</td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>12:07:29</td>
<td>178.427</td>
<td>183.8504</td>
</tr>
<tr>
<td>12:07:30</td>
<td>178.4306</td>
<td>183.8417</td>
</tr>
<tr>
<td>12:07:31</td>
<td>178.4342</td>
<td>183.8331</td>
</tr>
<tr>
<td>12:07:32</td>
<td>178.4377</td>
<td>183.8245</td>
</tr>
<tr>
<td>12:07:33</td>
<td>178.4412</td>
<td>183.8159</td>
</tr>
<tr>
<td>12:07:34</td>
<td>178.4447</td>
<td>183.8073</td>
</tr>
<tr>
<td>12:07:35</td>
<td>178.4482</td>
<td>183.7988</td>
</tr>
<tr>
<td>12:07:36</td>
<td>178.4517</td>
<td>183.7904</td>
</tr>
<tr>
<td>12:07:37</td>
<td>178.4551</td>
<td>183.7819</td>
</tr>
<tr>
<td>12:07:38</td>
<td>178.4586</td>
<td>183.7736</td>
</tr>
<tr>
<td>12:07:39</td>
<td>178.462</td>
<td>183.7652</td>
</tr>
<tr>
<td>12:07:40</td>
<td>178.4654</td>
<td>183.7569</td>
</tr>
<tr>
<td>12:07:41</td>
<td>178.4688</td>
<td>183.7486</td>
</tr>
<tr>
<td>12:07:42</td>
<td>178.4722</td>
<td>183.7404</td>
</tr>
<tr>
<td>12:07:43</td>
<td>178.4756</td>
<td>183.7322</td>
</tr>
<tr>
<td>12:07:44</td>
<td>178.4789</td>
<td>183.724</td>
</tr>
<tr>
<td>12:07:45</td>
<td>178.4822</td>
<td>183.7159</td>
</tr>
<tr>
<td>12:07:46</td>
<td>178.4856</td>
<td>183.7078</td>
</tr>
<tr>
<td>12:07:47</td>
<td>178.4889</td>
<td>183.6997</td>
</tr>
<tr>
<td>12:07:48</td>
<td>178.4922</td>
<td>183.6917</td>
</tr>
<tr>
<td>12:07:49</td>
<td>178.4954</td>
<td>183.6837</td>
</tr>
<tr>
<td>12:07:50</td>
<td>178.4987</td>
<td>183.6757</td>
</tr>
<tr>
<td>12:07:51</td>
<td>178.502</td>
<td>183.6678</td>
</tr>
<tr>
<td>12:07:52</td>
<td>178.5052</td>
<td>183.6599</td>
</tr>
<tr>
<td>12:07:53</td>
<td>178.5084</td>
<td>183.652</td>
</tr>
<tr>
<td>12:07:54</td>
<td>178.5116</td>
<td>183.6442</td>
</tr>
<tr>
<td>12:07:55</td>
<td>178.5148</td>
<td>183.6364</td>
</tr>
<tr>
<td>12:07:56</td>
<td>178.518</td>
<td>183.6287</td>
</tr>
<tr>
<td>12:07:57</td>
<td>178.5212</td>
<td>183.621</td>
</tr>
<tr>
<td>12:07:58</td>
<td>178.5243</td>
<td>183.6133</td>
</tr>
<tr>
<td>12:07:59</td>
<td>178.5275</td>
<td>183.6056</td>
</tr>
<tr>
<td>12:08:00</td>
<td>178.5306</td>
<td>183.598</td>
</tr>
<tr>
<td>12:08:01</td>
<td>178.5337</td>
<td>183.5904</td>
</tr>
<tr>
<td>12:08:02</td>
<td>178.5368</td>
<td>183.5828</td>
</tr>
<tr>
<td>12:08:03</td>
<td>178.5399</td>
<td>183.5753</td>
</tr>
<tr>
<td>12:08:04</td>
<td>178.543</td>
<td>183.5678</td>
</tr>
<tr>
<td>12:08:05</td>
<td>178.546</td>
<td>183.5603</td>
</tr>
<tr>
<td>12:08:06</td>
<td>178.5491</td>
<td>183.5529</td>
</tr>
<tr>
<td>12:08:07</td>
<td>178.5521</td>
<td>183.5455</td>
</tr>
<tr>
<td>12:08:08</td>
<td>178.5551</td>
<td>183.5381</td>
</tr>
<tr>
<td>12:08:09</td>
<td>178.5581</td>
<td>183.5308</td>
</tr>
<tr>
<td>Time</td>
<td>Currency</td>
<td>Bid</td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td>-----------</td>
</tr>
<tr>
<td>12:08:10</td>
<td>VNY</td>
<td>178.5611</td>
</tr>
<tr>
<td>12:08:11</td>
<td>VNY</td>
<td>178.5641</td>
</tr>
<tr>
<td>12:08:12</td>
<td>VNY</td>
<td>178.5671</td>
</tr>
<tr>
<td>12:08:13</td>
<td>VNY</td>
<td>178.5701</td>
</tr>
<tr>
<td>12:08:14</td>
<td>VNY</td>
<td>178.573</td>
</tr>
<tr>
<td>12:08:15</td>
<td>VNY</td>
<td>178.5759</td>
</tr>
<tr>
<td>12:08:16</td>
<td>VNY</td>
<td>178.5789</td>
</tr>
<tr>
<td>12:08:17</td>
<td>VNY</td>
<td>178.5818</td>
</tr>
<tr>
<td>12:08:18</td>
<td>VNY</td>
<td>178.5847</td>
</tr>
<tr>
<td>12:08:19</td>
<td>VNY</td>
<td>178.5876</td>
</tr>
<tr>
<td>12:08:20</td>
<td>VNY</td>
<td>178.5904</td>
</tr>
<tr>
<td>12:08:21</td>
<td>VNY</td>
<td>178.5933</td>
</tr>
<tr>
<td>12:08:22</td>
<td>VNY</td>
<td>178.5961</td>
</tr>
<tr>
<td>12:08:23</td>
<td>VNY</td>
<td>178.599</td>
</tr>
<tr>
<td>12:08:24</td>
<td>VNY</td>
<td>178.6018</td>
</tr>
<tr>
<td>12:08:25</td>
<td>VNY</td>
<td>178.6046</td>
</tr>
<tr>
<td>12:08:26</td>
<td>VNY</td>
<td>178.6074</td>
</tr>
<tr>
<td>12:08:27</td>
<td>VNY</td>
<td>178.6102</td>
</tr>
<tr>
<td>12:08:28</td>
<td>VNY</td>
<td>178.613</td>
</tr>
<tr>
<td>12:08:29</td>
<td>VNY</td>
<td>178.6158</td>
</tr>
<tr>
<td>12:08:30</td>
<td>VNY</td>
<td>178.6186</td>
</tr>
<tr>
<td>12:08:31</td>
<td>VNY</td>
<td>178.6213</td>
</tr>
<tr>
<td>12:08:32</td>
<td>VNY</td>
<td>178.624</td>
</tr>
<tr>
<td>12:08:33</td>
<td>VNY</td>
<td>178.6268</td>
</tr>
<tr>
<td>12:08:34</td>
<td>VNY</td>
<td>178.6295</td>
</tr>
<tr>
<td>12:08:35</td>
<td>VNY</td>
<td>178.6322</td>
</tr>
<tr>
<td>12:08:36</td>
<td>VNY</td>
<td>178.6349</td>
</tr>
<tr>
<td>12:08:37</td>
<td>VNY</td>
<td>178.6376</td>
</tr>
<tr>
<td>12:08:38</td>
<td>VNY</td>
<td>178.6402</td>
</tr>
<tr>
<td>12:08:39</td>
<td>VNY</td>
<td>178.6429</td>
</tr>
<tr>
<td>12:08:40</td>
<td>VNY</td>
<td>178.6456</td>
</tr>
<tr>
<td>12:08:41</td>
<td>VNY</td>
<td>178.6482</td>
</tr>
<tr>
<td>12:08:42</td>
<td>VNY</td>
<td>178.6508</td>
</tr>
<tr>
<td>12:08:43</td>
<td>VNY</td>
<td>178.6535</td>
</tr>
<tr>
<td>12:08:44</td>
<td>VNY</td>
<td>178.6561</td>
</tr>
<tr>
<td>12:08:45</td>
<td>VNY</td>
<td>178.6587</td>
</tr>
<tr>
<td>12:08:46</td>
<td>VNY</td>
<td>178.6613</td>
</tr>
<tr>
<td>12:08:47</td>
<td>VNY</td>
<td>178.6639</td>
</tr>
<tr>
<td>12:08:48</td>
<td>VNY</td>
<td>178.6664</td>
</tr>
<tr>
<td>12:08:49</td>
<td>VNY</td>
<td>178.669</td>
</tr>
<tr>
<td>12:08:50</td>
<td>VNY</td>
<td>178.6715</td>
</tr>
<tr>
<td>Time</td>
<td>VNY</td>
<td>178.6741</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>12:08:51</td>
<td>VNY</td>
<td>178.6766</td>
</tr>
<tr>
<td>12:08:52</td>
<td>VNY</td>
<td>178.6791</td>
</tr>
<tr>
<td>12:08:53</td>
<td>VNY</td>
<td>178.6817</td>
</tr>
<tr>
<td>12:08:54</td>
<td>VNY</td>
<td>178.6842</td>
</tr>
<tr>
<td>12:08:55</td>
<td>VNY</td>
<td>178.6866</td>
</tr>
<tr>
<td>12:08:56</td>
<td>VNY</td>
<td>178.6891</td>
</tr>
<tr>
<td>12:08:57</td>
<td>VNY</td>
<td>178.6916</td>
</tr>
<tr>
<td>12:08:58</td>
<td>VNY</td>
<td>178.6941</td>
</tr>
<tr>
<td>12:08:59</td>
<td>VNY</td>
<td>178.6965</td>
</tr>
<tr>
<td>12:09:00</td>
<td>VNY</td>
<td>178.6990</td>
</tr>
<tr>
<td>12:09:01</td>
<td>VNY</td>
<td>178.7014</td>
</tr>
<tr>
<td>12:09:02</td>
<td>VNY</td>
<td>178.7038</td>
</tr>
<tr>
<td>12:09:03</td>
<td>VNY</td>
<td>178.7063</td>
</tr>
<tr>
<td>12:09:04</td>
<td>VNY</td>
<td>178.7087</td>
</tr>
<tr>
<td>12:09:05</td>
<td>VNY</td>
<td>178.7111</td>
</tr>
<tr>
<td>12:09:06</td>
<td>VNY</td>
<td>178.7135</td>
</tr>
<tr>
<td>12:09:07</td>
<td>VNY</td>
<td>178.7159</td>
</tr>
<tr>
<td>12:09:08</td>
<td>VNY</td>
<td>178.7182</td>
</tr>
<tr>
<td>12:09:09</td>
<td>VNY</td>
<td>178.7206</td>
</tr>
<tr>
<td>12:09:10</td>
<td>VNY</td>
<td>178.7230</td>
</tr>
<tr>
<td>12:09:11</td>
<td>VNY</td>
<td>178.7253</td>
</tr>
<tr>
<td>12:09:12</td>
<td>VNY</td>
<td>178.7276</td>
</tr>
<tr>
<td>12:09:13</td>
<td>VNY</td>
<td>178.7300</td>
</tr>
<tr>
<td>12:09:14</td>
<td>VNY</td>
<td>178.7323</td>
</tr>
<tr>
<td>12:09:15</td>
<td>VNY</td>
<td>178.7346</td>
</tr>
<tr>
<td>12:09:16</td>
<td>VNY</td>
<td>178.7369</td>
</tr>
<tr>
<td>12:09:17</td>
<td>VNY</td>
<td>178.7392</td>
</tr>
<tr>
<td>12:09:18</td>
<td>VNY</td>
<td>178.7415</td>
</tr>
<tr>
<td>12:09:19</td>
<td>VNY</td>
<td>178.7438</td>
</tr>
<tr>
<td>12:09:20</td>
<td>VNY</td>
<td>178.7461</td>
</tr>
<tr>
<td>12:09:21</td>
<td>VNY</td>
<td>178.7483</td>
</tr>
<tr>
<td>12:09:22</td>
<td>VNY</td>
<td>178.7506</td>
</tr>
<tr>
<td>12:09:23</td>
<td>VNY</td>
<td>178.7528</td>
</tr>
<tr>
<td>12:09:24</td>
<td>VNY</td>
<td>178.7551</td>
</tr>
<tr>
<td>12:09:25</td>
<td>VNY</td>
<td>178.7573</td>
</tr>
<tr>
<td>12:09:26</td>
<td>VNY</td>
<td>178.7595</td>
</tr>
<tr>
<td>12:09:27</td>
<td>VNY</td>
<td>178.7617</td>
</tr>
<tr>
<td>12:09:28</td>
<td>VNY</td>
<td>178.7639</td>
</tr>
<tr>
<td>12:09:29</td>
<td>VNY</td>
<td>178.7661</td>
</tr>
<tr>
<td>12:09:30</td>
<td>VNY</td>
<td>178.7683</td>
</tr>
<tr>
<td>Time</td>
<td>VNY</td>
<td>CMA</td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
<td>---------</td>
</tr>
<tr>
<td>12:09:32</td>
<td>VNY</td>
<td>178.7705</td>
</tr>
<tr>
<td>12:09:33</td>
<td>VNY</td>
<td>178.7727</td>
</tr>
<tr>
<td>12:09:34</td>
<td>VNY</td>
<td>178.7749</td>
</tr>
<tr>
<td>12:09:35</td>
<td>VNY</td>
<td>178.777</td>
</tr>
<tr>
<td>12:09:36</td>
<td>VNY</td>
<td>178.7792</td>
</tr>
<tr>
<td>12:09:37</td>
<td>VNY</td>
<td>178.7813</td>
</tr>
<tr>
<td>12:09:38</td>
<td>VNY</td>
<td>178.7834</td>
</tr>
<tr>
<td>12:09:39</td>
<td>VNY</td>
<td>178.7856</td>
</tr>
<tr>
<td>12:09:40</td>
<td>VNY</td>
<td>178.7877</td>
</tr>
<tr>
<td>12:09:41</td>
<td>VNY</td>
<td>178.7988</td>
</tr>
<tr>
<td>12:09:42</td>
<td>VNY</td>
<td>178.7919</td>
</tr>
<tr>
<td>12:09:43</td>
<td>VNY</td>
<td>178.794</td>
</tr>
<tr>
<td>12:09:44</td>
<td>VNY</td>
<td>178.7961</td>
</tr>
<tr>
<td>12:09:45</td>
<td>VNY</td>
<td>178.7982</td>
</tr>
<tr>
<td>12:09:46</td>
<td>VNY</td>
<td>178.8003</td>
</tr>
<tr>
<td>12:09:47</td>
<td>VNY</td>
<td>178.8024</td>
</tr>
<tr>
<td>12:09:48</td>
<td>VNY</td>
<td>178.8044</td>
</tr>
<tr>
<td>12:09:49</td>
<td>VNY</td>
<td>178.8065</td>
</tr>
<tr>
<td>12:09:50</td>
<td>VNY</td>
<td>178.8085</td>
</tr>
<tr>
<td>12:09:51</td>
<td>VNY</td>
<td>178.8106</td>
</tr>
<tr>
<td>12:09:52</td>
<td>VNY</td>
<td>178.8126</td>
</tr>
<tr>
<td>12:09:53</td>
<td>VNY</td>
<td>178.8146</td>
</tr>
<tr>
<td>12:09:54</td>
<td>VNY</td>
<td>178.8166</td>
</tr>
<tr>
<td>12:09:55</td>
<td>VNY</td>
<td>178.8187</td>
</tr>
<tr>
<td>12:09:56</td>
<td>VNY</td>
<td>178.8207</td>
</tr>
<tr>
<td>12:09:57</td>
<td>VNY</td>
<td>178.8227</td>
</tr>
<tr>
<td>12:09:58</td>
<td>VNY</td>
<td>178.8247</td>
</tr>
<tr>
<td>12:09:59</td>
<td>VNY</td>
<td>178.8267</td>
</tr>
<tr>
<td>12:10:00</td>
<td>VNY</td>
<td>178.8286</td>
</tr>
</tbody>
</table>
# APPENDIX F: ACRONYMS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADB</td>
<td>Android Debug Bridge</td>
</tr>
<tr>
<td>ADT</td>
<td>Android Development Tool</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>COTS</td>
<td>Commercial off-the-shelf</td>
</tr>
<tr>
<td>DME</td>
<td>Distance Measuring Equipment</td>
</tr>
<tr>
<td>FIR</td>
<td>Finite Impulse Response</td>
</tr>
<tr>
<td>IIR</td>
<td>Infinite Impulse Response</td>
</tr>
<tr>
<td>JNI</td>
<td>Java Native Interface</td>
</tr>
<tr>
<td>LO</td>
<td>Local Oscillator</td>
</tr>
<tr>
<td>NavAid</td>
<td>Navigational Aide. Also known as Radio Beacon</td>
</tr>
<tr>
<td>NDB</td>
<td>Non-Directional Beacon</td>
</tr>
<tr>
<td>NDK</td>
<td>Native Development Kit</td>
</tr>
<tr>
<td>PAL</td>
<td>Phase Alternating Line</td>
</tr>
<tr>
<td>RDF</td>
<td>Radio Direction Finding</td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
</tr>
<tr>
<td>UBT</td>
<td>Update Bearing Thread</td>
</tr>
<tr>
<td>UIT</td>
<td>User Interface Thread</td>
</tr>
<tr>
<td>VOR</td>
<td>VHF (Very High Frequency) Omni Directional Radio Range</td>
</tr>
</tbody>
</table>
APPENDIX G: ADDENDUM

Air Guide is a joint graduate project between Avetis Petrosyan and Rolando Douglas.

This document will explain the division of responsibilities between the two parties.

Avetis Petrosyan is responsible for all the following tasks:

- Setup ADT for Mac OS X
- Setup ADB
- Interface the RTL-SDR with the Android device
- Create Radio Beacon decoding software with the use of DSP techniques
- Create FIR filters to separate analog signals into reference, variable and Morse Code signals
- Create an algorithm to calculate the phase difference between reference and variable signals to determine bearing
- Create audio drivers using OpenSL ES to playback Morse Code
- Unit Test

Rolando Douglas is responsible for all the following tasks:

- Setup ADT for Ubuntu Linux
- Create the User Interface for an Android device
- Interface with offline mapping software
- Interface with GPS
- Create map tile database for select regions and zoom levels of choice and all icons
- Create database of NavAid information (i.e. geolocation, channel, frequency, ID…)
- Create algorithms to triangulate position, hide/show overlayed icons and sliding drawers, and calculate heading
- Unit Test

Both parties shared responsibilities for the following tasks:

- Integration
- Physical setup and testing the operation of the system in several locations
- Debug code when problems were found