# Android phone controlled Beagle board based PSCR in a Dynamic Spectrum Access environment

Aravind Radhakrishnan

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Dr.Charles W.Bostian, Chair
Dr.Timothy Pratt
Dr.Yaling Yang

Sep 5, 2010
Blacksburg, Virginia

Keywords: DSA, DSA Broker, Cooperative Spectrum Sensing

# Android phone controlled Beagle board based PSCR in a Dynamic Spectrum Access environment

Aravind Radhakrishnan

(ABSTRACT)

Public Safety Cognitive Radio (PSCR) is a Software Defined Radio(SDR) developed by the Center for Wireless Telecommunications (CWT) at Virginia Tech. PSCR can configure itself to interoperate with any public safety waveform it finds during the scan procedure. It also offers users the capability to scan/classify both analog and digital waveforms.

The current PSCR architecture can only run on a general purpose processor and hence is not deployable to the public safety personnel. In the first part of this thesis an Android based control application for the PSCR on a Beagle Board(BB) and the GUI for the control application are developed. The Beagle Board is a low-cost, fan-less single board computer that unleashes laptop-like performance and expandability. The Android based Nexus One connected to the Beagle Board via USB is used to control the Beagle Board and enable operations like scan, classify, talk, gateway etc. In addition to the features that exist in the current PSCR a new feature that enables interoperation with P25 (CPFSK modulation) protocol based radios is added. In this effort of porting the PSCR to Beagle Board my contributions are the following (i) communication protocol between the Beagle Board and the Nexus One (ii) PSCR control application on the Android based Nexus One (iii) detection/classification of P25 protocol based radios.

In the second part of this thesis, a prototype testbed of a Dynamic Spectrum Access

(DSA) broker that uses the Beagle Board PSCR based sensor/classifier is developed. DSA in simple terms is a concept that lets the user without license (secondary user) to a particular frequency access that frequency, when the licensed user (primary user) is not using it. In the proposed testbed we have two Beagle Board based sensor/classifiers that cooperatively scan the spectrum and report the results to the central DSA broker. The DSA broker then identifies the frequency spectrum without primary users and informs the secondary users about the free spectrum. The secondary users can then communicate among each other using the frequency band allocated by the DSA broker. When the primary user enters the spectrum occupied by the secondary user, the DSA broker instructs the secondary user to use a different spectrum. Based on the experiments conducted on the testbed setup in the CWT lab environment, the average time taken by the DSA broker to detect the presence of primary user is 0.636 secs and the average time taken for the secondary user to leave the frequency band that interferes with the primary user is 0.653 secs.

This project is supported by Award No.2009-SQ-B9-K011 awarded by the National Institute of Justice,Office of Justice Programs,US Department of Justice. The opinions,findings and conclusions or recommendations expressed are those of the author and do not necessarily reflect the views of the Department of Justice

# Acknowledgements

I would like to express my sincere gratitude to my advisor Dr.Charles W.Bostian. I met Dr.Bostian in Fall 2008 and at that time I did not have any wireless communication background. Dr.Bostian was kind enough to trust me and he gave me the wonderful opportunity to work in the CWT Lab. It would not be an exaggeration to say that CWT lab laid the foundation, for the knowledge I gained in the field of wireless communications and computer networking.

I would like to thank my committee members Dr.Timothy Pratt and Dr.Yaling Yang for being very supportive during the course of my thesis work.

I would like to thank all the CWT students with whom I worked closely. They are Qinqin Chen, Feng Andrew Ge, Rohit Ranganekar, Al Fayez, Sujit Nair, Jeanette Nounagnon, Alex Young, Ying Wang and Mark Silvius.

I would also like to thank Judy Hood for being really kind and supportive. Thank you for all the hard work that you have put in, to make sure all our appointments, meetings and administrative tasks went through without any hassles.

Last but not least, I would like to thank my parents for always being there for me.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

The first part of this thesis involves porting the Public Safety Cognitive Radio(PSCR) (a Software Defined Radio that can configure itself to interoperate with any public safety waveform it finds during the scan procedure)from a general purpose processor to the Beagle Board. The motivation behind doing this is to make the PSCR architecture compatible with that used by the Land Mobile Radio(LMR) industry. There are several challenges involved in porting the PSCR to the Beagle Board and a couple of them are addressed in this thesis. The first is the performance of PSCR on Beagle Board.The Beagle Board is a low-cost, fan-less single-board computer based on TI's OMAP3 device family [1], with all of the expandability of today's desktop machines, but without the bulk, expense, or noise. All of these advantages come at a price and that is performance. The processing capability of Beagle Board is much less than a laptop. This lack of performance is taken into account while designing the architecture of PSCR for the Beagle Board. Second, the Beagle Board does not have a user interface,therefore in order to control the Beagle Board we need a device with a user

Interface. The Android based Google's Nexus One phone is used for this purpose. It is connected to the Beagle Board using a USB interface and uses a communication protocol designed as a part of this thesis to communicate with the PSCR.

The second part of this thesis uses the Beagle Board based PSCR to create a cooperative sensor network,that allows Dynamic Spectrum Access. The demand for the spectrum is ever increasing whereas the available spectrum is limited. This leads to scarcity of spectrum [2]. The scarcity of spectrum and the poor spectrum utilization efficiency [3] [4] [5] led to the invention of the Dynamic Spectrum Access(DSA) technology. The main goal of DSA is to increase spectrum utilization by allocating spectrum to the secondary users(users who do not have a license to use the spectrum) in the absence of the primary users(users having a license to use the spectrum). DSA as defined in draft standard IEEE 1900.1 "A technique by which a radio system dynamically adapts to select operating spectrum to use available(in local time-frequency space)spectrum holes with limited spectrum use rights" [6]. Going by this definition, a flexible radio system that can operate in different frequency bands and that can support different type of waveforms is needed to achieve DSA. Such flexibility is offered by cognitive radio technology and is defined in IEEE 1900.1 as: "Radio in which communication systems are aware of their environment and internal state and can make decisions about their radio operating behavior based on that information and predefined objectives". The importance of the DSA technology and the suitability of cognitive radio to achieve it was the main motivation for using the Beagle Board based PSCR to achieve DSA.

### 1.1.1 Background on the existing PSCR architecture and it's drawbacks

#### Cognitive Radio

Cognitive Radios can be formally defined as Software Defined Radios(SDR) that can change their transmitter/receiver parameters based on the knowledge they gain by observing the surrounding radio conditions [7] [8].

#### Basic PSCR architecture

Public Safety Cognitive Radio is an SDR that can scan the public safety frequency bands [9] [10] and configure itself to interoperate with different types of public safety waveforms. This capability of PSCR can be used to communicate with any public safety waveform. It can also be used as a gateway to bridge waveforms that are not compatible with each other. The PSCR Architecture [9]is shown in Figure 1.1.

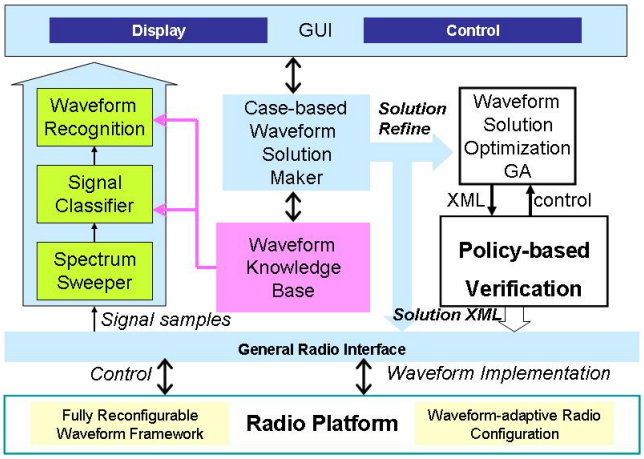


**Figure 1.1** CWT PSCR Node Block Diagram. 2007 Bin Le. Reprinted [10], with permission, from B. Le, "Building a Cognitive Radio: From Architecture Definition to Prototype Implementation," Ph.D. Dissertation, in Dept. of Electrical and Computer Engineering. 2007, Virginia Polytechnic Institute and State University: Blacksburg, VA. .

The PSCR is based on the cognition loop shown in Figure 1.2 that was developed in Center for Wireless Telecommunication Lab [8] [9].

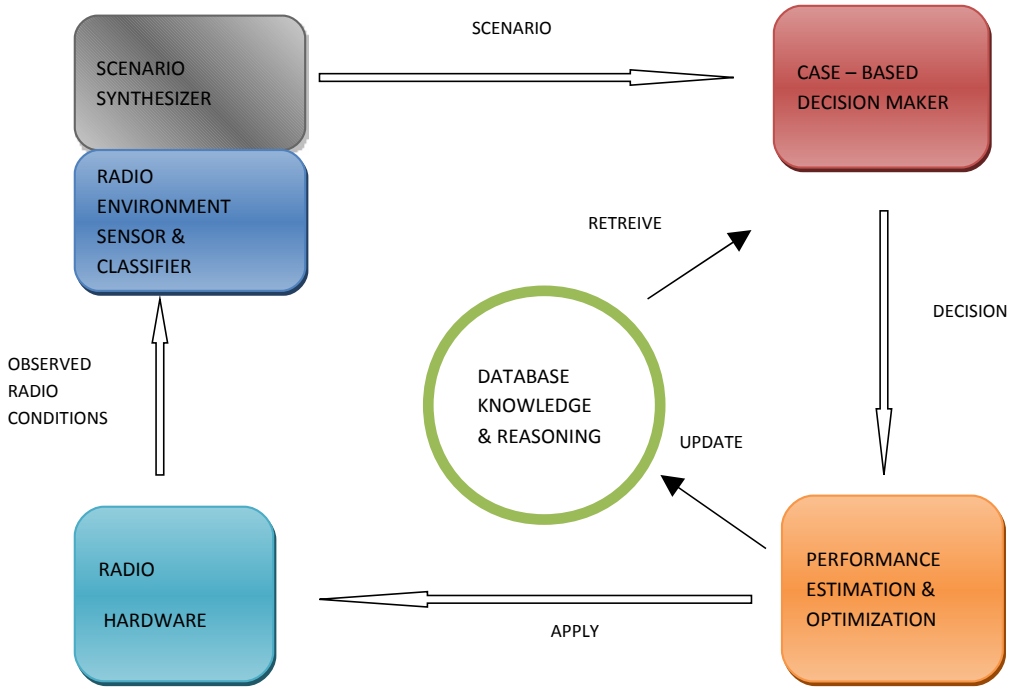


**Figure 1.2** Cognition Cycle.

The Cognition loop begins by collecting the observed radio environment conditions. The scenario synthesizer then creates a scenario representation for the observed radio conditions. The case-based decision maker compares this scenario with the results in the database and decides if the existing settings are satisfying or if new optimal settings are needed. If new optimal settings are needed, the optimizer identifies the optimal settings and updates the database. These new settings are then used to configure the radio hardware.

### PSCR - Modes of Operation.

The PSCR can operated in three different modes. They are (i) Scan: This mode is used to scan the frequency spectrum and identify the type of waveform. This mode makes use of the signal sensor, classifier and waveform recognizer. (ii) Talk: This mode is used to

communicate with any of the public safety waveforms in the waveform knowledge base or from the public safety waveforms that were detected in the Scan procedure. Once the type of public safety waveform is selected, the PSCR configures the USRP/GNURADIO to establish the link from the PSCR to the required public safety device. (iii) Gateway: This mode can be used to make PSCR act as a bridge to link two public safety waveforms that are not compatible with each other.

### Vehicular PSCR

The Vehicular PSCR(VPSCR) was an effort to make the PSCR portable, so that the public safety officials can make use of the PSCR. The VPSCR has a remote personal digital assistant(PDA) that controls the PSCR installed on a laptop (located in a public safety vehicle). This is shown in Figure 1.3. Using the PDA, the public safety officials can remotely operate the PSCR in scan, talk or gateway mode.

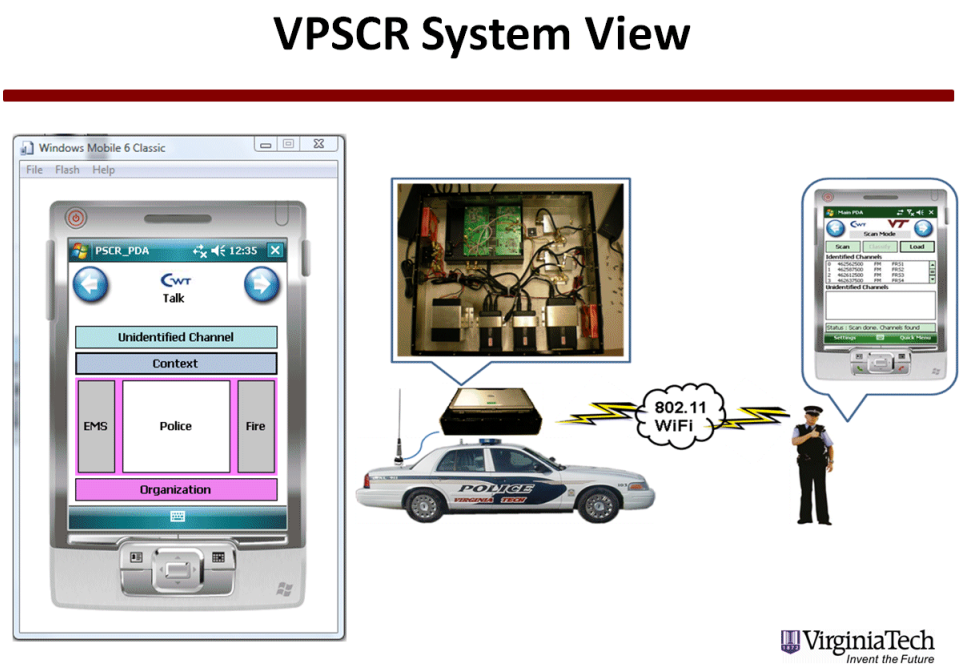


**Figure 1.3** Vehicular PSCR system view.

### Problem Statement

Although VPSCR is portable to some extent, it is not completely mobile. The user should be within a specified distance from the vehicle in order to use the PDA and remotely control the PSCR on the vehicle. This is a major drawback of the VPSCR. The goal is to make the PSCR into a hand-held radio. As a first step, the PSCR is installed on the Beagle Board. The Android based Google's Nexus One phone connected to Beagle Board via USB is used to control the PSCR on the Beagle Board. Using this Android phone the user can configure the PSCR framework to operate in either scan, talk or gateway mode.

Compared to the PSCR control application on the PDA, the control application on the Android phone is different in several ways: (i) The audio processing technique used is different

in the Android phone compared to the PDA. In the case of PDA, the audio was sent over the wifi to the remote PSCR and hence it needed several stages of digital audio processing before sending it to the GNURADIO transmit path flow graph. But in the case of Android phone, the speaker of the Android phone is connected to the MIC of the Beagle Board using a wired connection. Therefore, audio from the Android phone is directly fed to the MIC of the Beagle Board, which acts as audio source for the GNURADIO transmit path flow graph. This is explained in detail in Section 2.3. (ii) Wifi was used to control the PSCR from the PDA, whereas USB based tethered connection is used to control the PSCR from the Android phone. This is shown in Section 2.2.5.

## 1.2 Contributions

The contributions from the first part of this thesis are (i)A communication protocol between the PSCR on the Beagle Board and the Android based Nexus One. This protocol is used when the user pushes either scan/classify/talk on the phone to communicate with the PSCR and configure the Software Defined Radio to enable these functions (ii) Developing the Android phone GUI along with students from the Human Factors Engineering and Ergonomics Laboratory, Department of Industrial and Systems Engineering, Virginia Tech (iii) Android phone PSCR control application and finite state machine to control the Beagle Board (iv) Audio processing on the Android phone (v) In addition to the existing PSCR features an additional feature that enables detection of P25 protocol based radios is developed as a part of this thesis.

In the second part of this thesis, a prototype of the cooperative sensor network that uses the Beagle Board based PSCR to achieve Dynamic Spectrum Access is proposed and implemented. This testbed is based on the cooperative sensor network that was created for the

Dyspan 2008 challenge. This work was the combined effort of Feng Ge, Rohit Ranganekar and myself. As a part of this prototype testbed, a Beagle Board based sensor architecture, DSA broker architecture, secondary user architecture and a custom designed protocol to enable communication between the different nodes in the testbed is developed.

# Chapter 2

# Porting PSCR to Beagle Board

## 2.1 Software Defined Radios

Software Defined Radios(SDR) was first defined by Joseph Mitola in 1991 [11] as " A software radio is a radio whose channel modulation waveforms are defined in software. That is, waveforms are generated as sampled digital signals, converted from digital to analog via a wideband DAC and then possibly upconverted from IF to RF. The receiver, similarly, employs a wideband analog to digital converter (ADC) that captures all of the channels of the software radio node. The receiver then extracts, downconverts and demodulates the channel waveform using software on a general purpose processor." The SDR offers a flexible radio architecture that can be reconfigured in real-time [12]. By having such a reconfigurable architecture,the signal reception can be improved by just upgrading the software with improved interference handling,encryption and error recovery schemes. Thus most of the improvements or updates to the software defined radio design can be made through software. On the other hand, the conventional radio architecture is mainly dependent on the hardware and has very limited configurability using software. This limitation in reconfigurability is a

major disadvantage, if the radio needs to be upgraded. The only option in this case is to redesign the radio from scratch.

The architecture of a software defined radio [12] is shown in Figure 2.1. The antenna is used to receive the modulated RF signals and it can be controlled by software to tune to any frequency within its range. Traditional radios need different RF front ends for different frequency bands, but the software radio has an RF front end that can cover wide range of frequencies. The RF front end converts the received RF signals to an intermediate frequency, before sending it to the ADC. The ADC converts the analog signal into digital bits. The output from ADC is fed into the digital down converter(DDC). The digital down converter converts the signal from the intermediate frequency to baseband. The higher the sampling rate the more demanding and expensive the hardware gets. This is the main reason for converting the signal to baseband before processing it. The final stage is the processing of the digital baseband signal and this can be done using different types of hardware like Field Programmable Gate Arrays(FPGA), ASICS or DSP. All these hardware support some level of re-programmability [12]. ASICs use fixed silicon to implement the system and they are optimized for that particular system. FPGAs on the other hand provide a lot more flexibility from a hardware perspective than the ASIC, but they are not as optimized as the ASIC. DSP uses microprocessors and can be programmed using high level languages. DSP offers the most flexibility among the three, but there are trade offs when it comes to performance. The trade off between flexibility and optimization in terms of power consumption and speed needs to be considered before selecting one of these hardwares. Now that we have discussed the general architecture of the SDR, let's see the specific hardware and software components that make up the SDR used in this thesis.

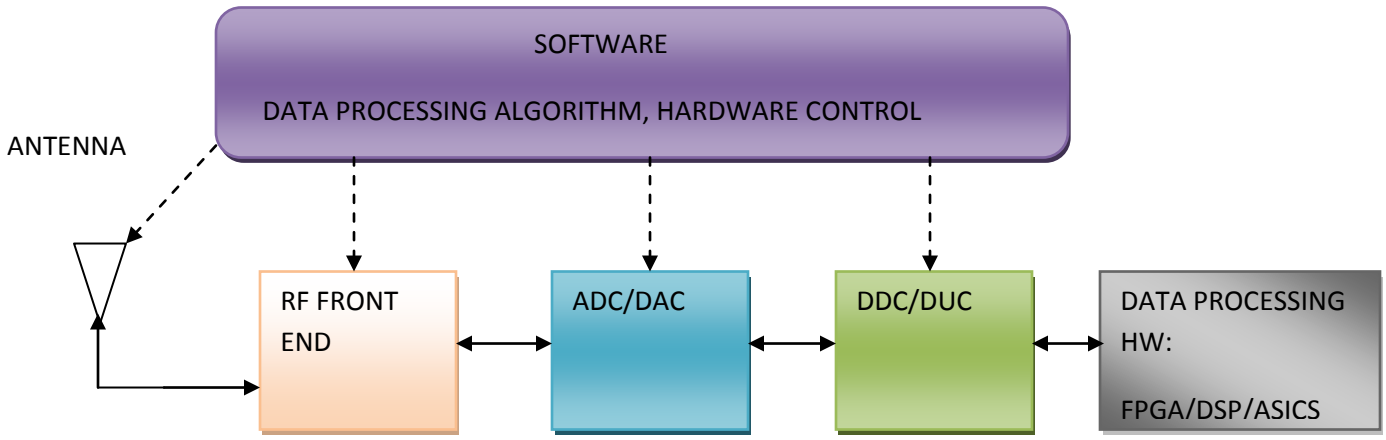


**Figure 2.1** High Level of SDR architecture.

### 2.1.1 SDR Software/Hardware

The Universal Software Radio Peripheral (USRP) [13]is the RF front end hardware used by the PSCR. The USRP version used as a part of this thesis contains a mother board that has four 12-bit 64 Msps analog to digital converters(ADC), four 14-bit 128 Msps digital to analog converters(DAC). The antenna is connected to daughter boards which provide RF amplification and are mounted on the mother board. Apart from these, the USRP hardware also contains an Alter FPGA, a Cypress FX2 USB 2.0 controller, a digital down converter(DDC) and a digital up converter(DUC).

GNURADIO [14] is the software that controls the USRP and helps the user build a radio by creating the signal flow graph connecting the various signal processing blocks. The signal processing blocks are implemented using C++ and the flow graph that connects the various signal processing blocks is implemented using Python.

## 2.2 Porting PSCR to Beagle Board

### 2.2.1 Beagle Board

The Beagle Board is shown in Figure. 2.2. This board is a low cost OMAP3530 processor based board [1] [15]. The Beagle Board comes with a Micron POP(Package on Package) memory. This means that the NAND and SDRAM are mounted on top of the OMAP3530 processor. This Micron POP memory [15] provides (i) 2Gb NAND * 16 (256MB) (ii)2Gb MDDR SDRAM * 32 (256MB). Additional memory elements can be added by using a SD or MMC in the SD/MMC slot, or by using a USB thumb drive/hard drive. In our setup we have used the SD card.

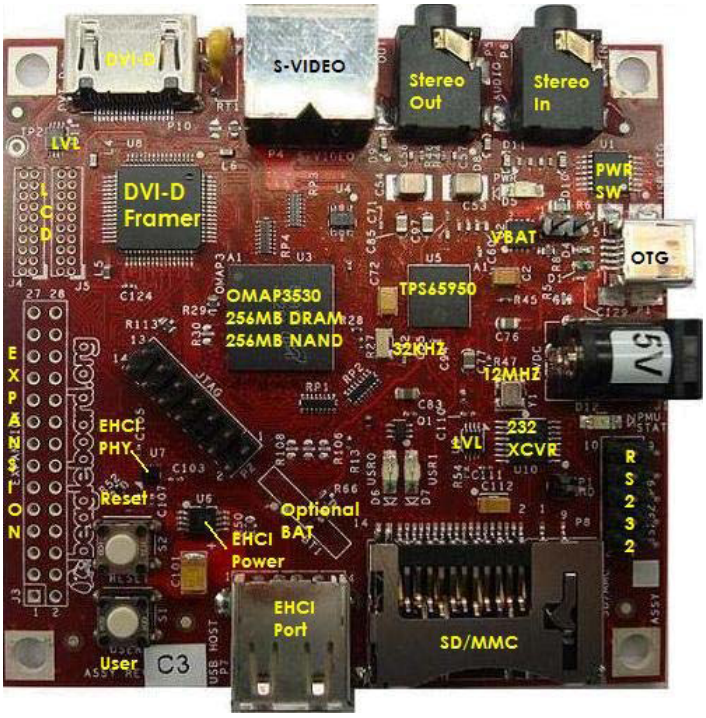


**Figure 2.2** Beagle Board.

The audio is fed into the Beagle Board using the stereo audio input connector. Likewise,

the audio from the Beagle Board can be sent out using the stereo audio output connector. Both audio in and audio out are supported by the onboard audio CODEC provided by TPS65950 [15]. As the Beagle Board does not contain a user interface we need to connect Beagle Board to a liquid-crystal display (LCD) screen for development purposes and this is done using the DVI-D interface. The different components connected to the Beagle Board, including the LCD screen are shown in Figure 2.3

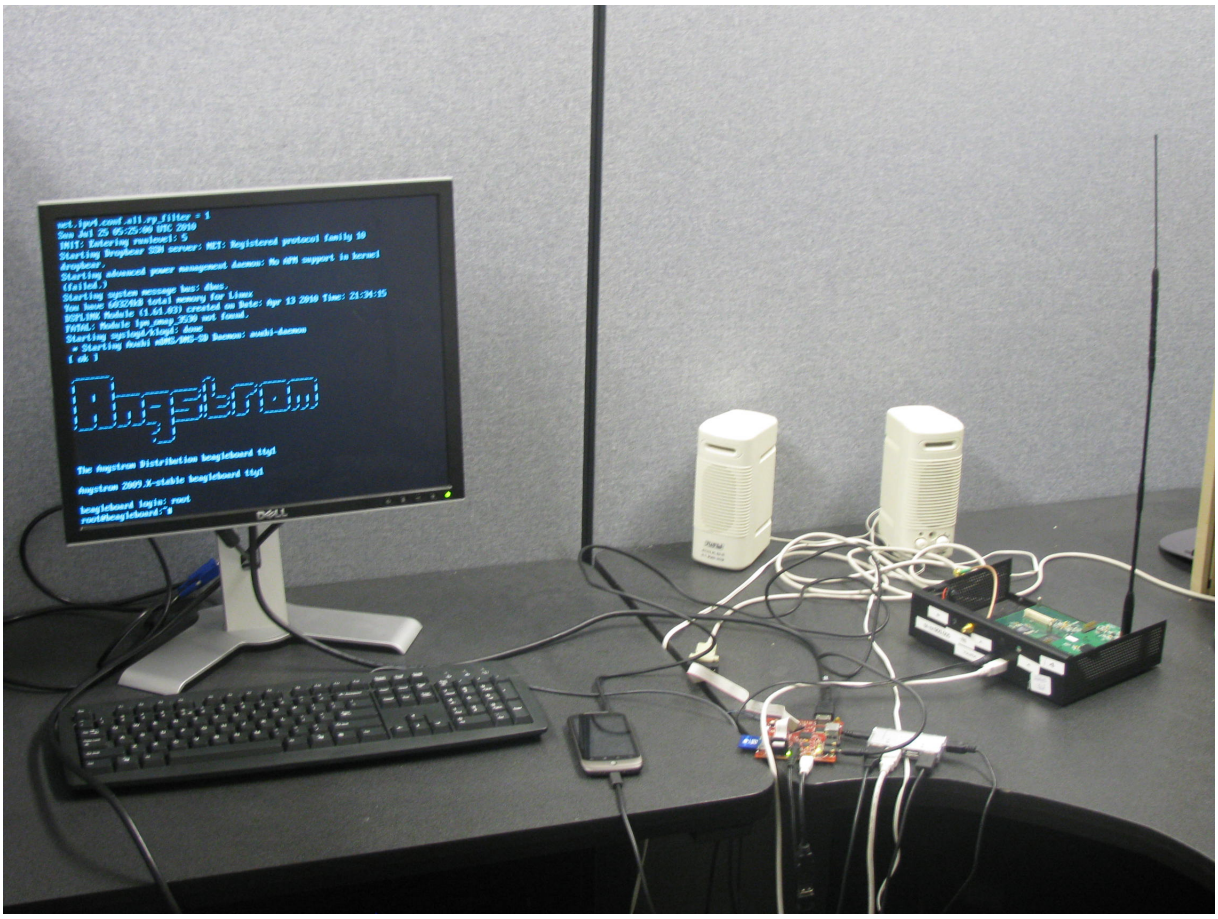

**Figure 2.3** Setup showing the LCD screen, Android phone, and Beagle Board.

### 2.2.2 Setting up GNURADIO/DSP on the Beagle Board

#### GNURADIO on the Beagle Board

Figure 2.4 shows the GNURADIO-USRP architecture that is used in this thesis. As shown in the figure, the GNURADIO has to be installed on the Beagle Board. In order to install GNURADIO, Open Embedded(OE), the build framework for embedded Linux, is set up [16] on the Beagle Board. OE offers an excellent cross-compile environment and has the capability to run on many hardware architectures and any Linux distribution.

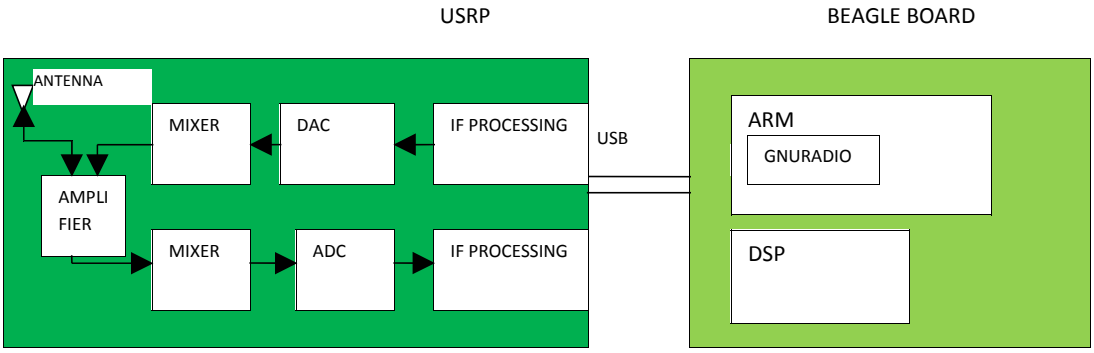


**Figure 2.4** Hardware and Software architecture of USRP-GNURADIO combination.

#### Libraries needed to use DSP on the Beagle Board

The DSP block on the Beagle Board is used for Finite Impulse Response(FIR)filtering [17]. Two types of FIR filters are implemented on the DSP: (i) FIR Filter with floating-point real coefficients and floating-point complex input/output and (ii) A FIR filter with floating-point real coefficients and floating-point real input/output data.

Several libraries must be set up in order to use the DSP. They are (i) DSP/BIOS Library [18] is used to develop embedded real time applications on the DSP chip. It provides a small

firmware and tools for real-time tracing and analysis, (ii) TI DSP/BIOS Link (DSPLink) Library is used for the inter-process communication(IPC) between the ARM and DSP on the Beagle Board. DSP/BIOS Link achieves IPC using techniques like shared memory, message passing etc. (iii) Code Composer Studio(CCStudio) Integrated development environment(IDE) for TI's DSPs. CCstudio provides a suite of tools to develop embedded applications and to debug them. It has compilers for every type of TI's device. In our case we have used C6x Compiler [19]. Some of the other features that it provides are project build environment, debugger, simulator.

### 2.2.3 Calling a DSP block from the ARM of the Beagle Board

Once all of these different libraries are set up as shown in Section 2.2.2, the DSP block can be called from the GNURADIO flow graph, that is residing on the ARM of the Beagle Board, as shown in Figure 2.5. Python is the programming language that is used to write the flow graph and connect the C++ signaling blocks. The call from python to C++ is made possible by SWIG module. The C++ signalling blocks are either on the ARM or on the DSP. If it were on the ARM then it is a simple local call within the ARM. But if the C++ signalling block resides on the DSP then in that case the DSP/BIOS Link interface is used to call the DSP signalling block.

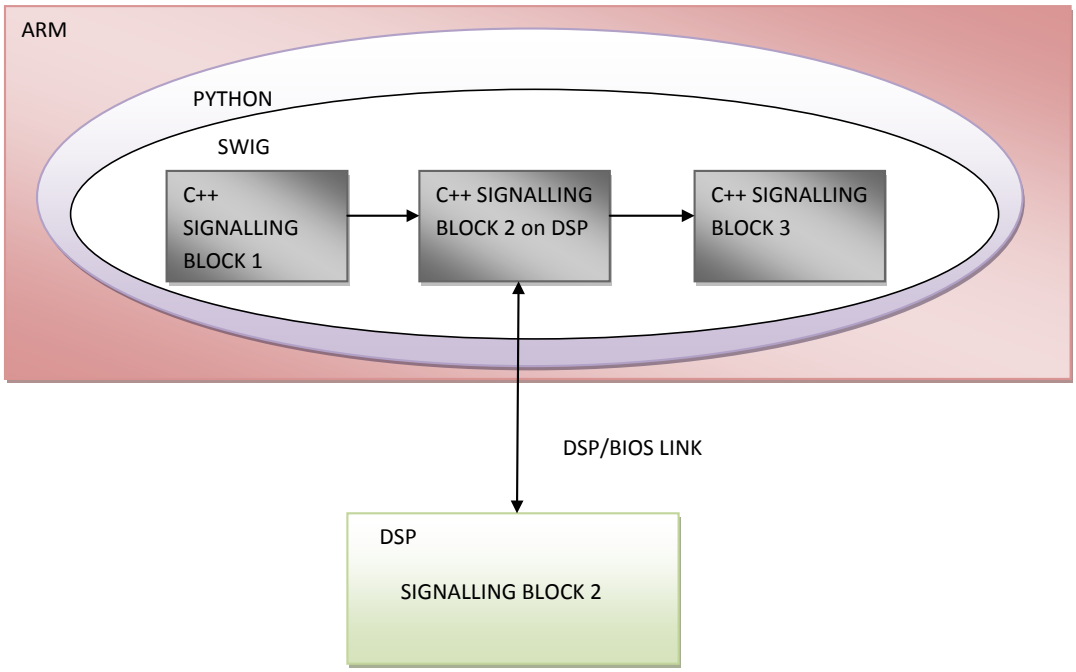


**Figure 2.5** Signal flow from the ARM to DSP of the Beagle Board.

Three components of the DSP/BIOS link library are used to talk to the signalling block residing on the DSP. They are shown in Figure 2.6. The PROC component [20] provides the basic functionalities like (i)loading a DSP executable on to the DSP processor (ii)starting and stopping execution of the DSP executable on the DSP processor and (iii)writing to and reading from the DSP memory. The POOL component has APIs that can be used to establish shared memory between the ARM and the DSP. CHNL, the channel component, is used to configure buffers for data transfer between the ARM and the DSP. Thus the CHNL and POOL components together enable inter-process communication between ARM and DSP. Using this IPC mechanism, the real or complex data collected from the USRP by the ARM is sent to the DSP, after converting the data from floating-point format to fixed-point format. This conversion is necessary because the DSP here uses fixed-point format.

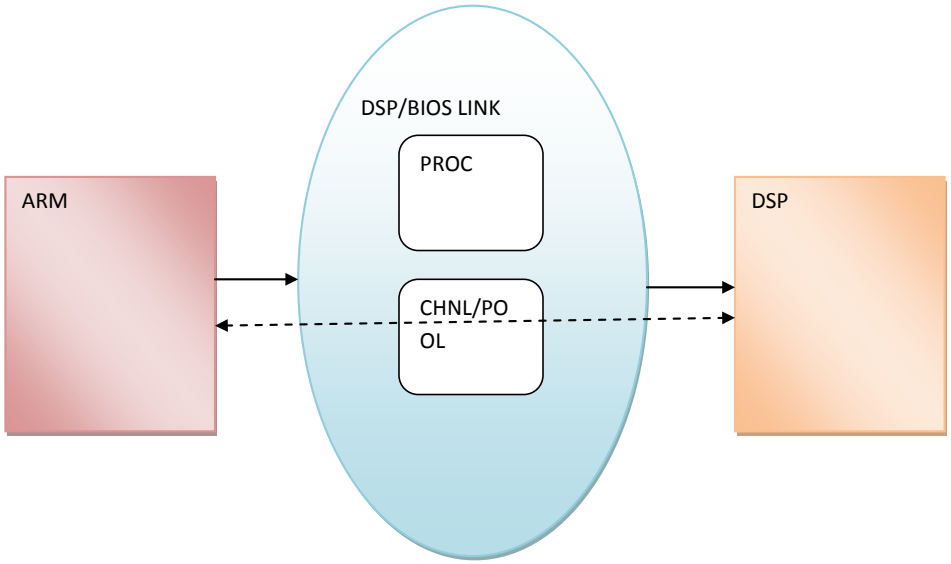


**Figure 2.6** Components of DSP/BIOS link library.

### 2.2.4 Google's Nexus One phone

Google's Nexus One phone shown in Figure 2.7 is the mobile phone used in this thesis and is based on the Android Operating System. The PSCR control application developed as a part of this thesis is installed on this phone and is used to control the PSCR on the Beagle Board. Using this PSCR control application, the user will be able to configure the PSCR on the Beagle Board to do the following: (i) scan the radio environment and identify other public safety personnel in the vicinity (ii) communicate with other public safety personnel/department in push-to-talk mode (iii) act as a gateway between different public safety departments.


**Figure 2.7** Google's Nexus One mobile phone.

In order to develop a PSCR control application on Android phone, a development environment is needed. Android SDK [21]along with Eclipse development environment is installed on a Windows based computer, which is used for development. The instructions provided in the reference [21] are used to setup the development machine. Java is used as the programming language to develop the PSCR control application. To debug the Android application using the development machine a tool called Android Debug Bridge(adb) is used. The Android Debug Bridge has 3 components that lets us debug the Android application: (i)The Android Debug Bridge client that can be invoked on the development machine using a shell command (ii)The Android Debug Bridge server that runs as a background process on the development machine and connects the client on the development machine and the daemon on the Android phone. (iii)A daemon that runs as a background process on the Android phone. 'adb -d logcat' is the command that is used to invoke the debugger on the development machine; the -d option here directs the adb tool to the Android device connected to

the USB.

The PSCR control application is bundled into an Android package and is marked by an archive file [21] with an .apk suffix. This file can then be used to install the PSCR control application on any Android based phone. An important characteristics of any application running on an Android device is that each application runs as a separate Linux process and has its own Virtual Machine(VM). This way each application is isolated from other applications on the Android phone.

The Android GUI was designed in collaboration with Gyu Hyun Kwon, graduate student, Human Factors Engineering and Ergonomics Laboratory, Department of Industrial and Systems Engineering, Virginia Tech. Software verification and testing of the code is done by Nikhil Rahagude, graduate student, Department of Electrical and Computer Engineering, Virginia Tech.

### 2.2.5 Tethering

The Android based phone and Beagle Board are connected using USB and tethering is used to make them communicate with each other. Tethering, in general, is a technique that gives internet access to computers by using mobile phones as a modem. In order to enable Tethering on Android bases Nexus One, we need to use an option called *Internet tethering* on the *Settings* menu. Once this option is enabled, the Linux operating system on the Beagle Board detects the Android phone as a network adapter and it is usually referred to as usb(x); x is a number that depends on which USB port the Android phone is connected to.

Once the Android phone is detected as a network adapter, an IP address is allocated both to the Android phone and to the usb(x) interface that was detected on the Beagle Board. The following commands were used to give IP address to the Android phone and the usb(x)

interface on the Beagle Board. On the Beagle Board, sudo ifconfig usb0 192.168.1.1 On the Android phone, sudo ifconfig usb0 192.168.1.2 Once this is done the Android phone and the Beagle Board can ping each other and the basic connectivity is established between them.

### 2.2.6 Interaction between Android phone and the Beagle Board

Figure 2.8 shows the architecture used for communication, between the Android phone and the Beagle Board. The Android phone contains the PSCR control application and sends commands to the Beagle Board, to achieve the required functionality based on the user's request. The PSCR control application has two components to enable this functionality, the Android GUI and the Android client. The Android client receives commands from the Android GUI and uses UDP sockets to talk to the Beagle Board over the tethered USB connection.

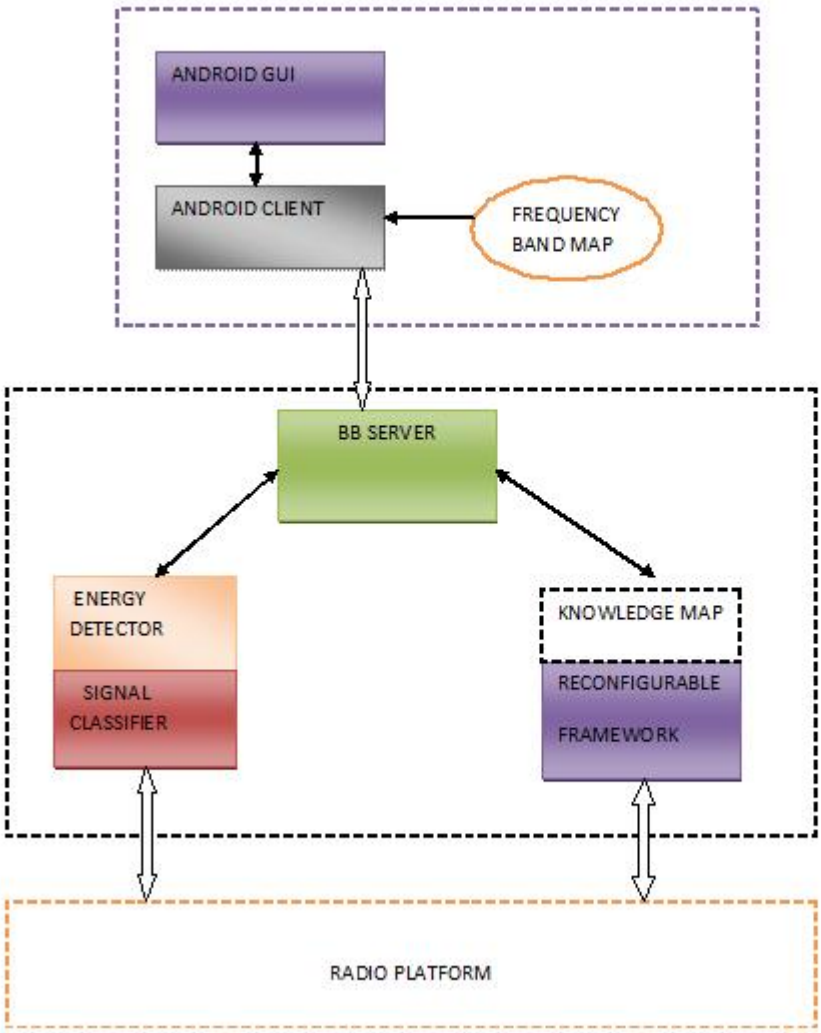


**Figure 2.8** Architecture used for communication between the Android phone and the Beagle Board.

The Beagle Board has a server that waits continuously to receive commands from the Android client. Based on the command received, the server calls the appropriate module and returns the results back to the Android client. For example, if the user wants to scan the frequency spectrum, then the energy detector module is called and the scanned results from the energy detector module are sent back to the Android client. The Android client then uses the

frequency band map to identify the public safety department to which the scan results belong and sends the public safety department names to the Android GUI.

The signal classifier module, is used to classify the detected signals. This module originally had the capability to classify signals having FM/AM as modulation. P25 based radios use CPFSK as the modulation, and capability to classify signals based on this modulation was added to this module as a part of this thesis.

P25 is a user driven set of system standards that define the radio communications systems [22] that can serve the public safety departments. The radios that conform with these standards are called P25 radios. The P25 radios have both analog and digital communication capabilities. They can communicate with legacy radios using analog communication and with other P25 radios using digital communication. The P25 radio for which the detection capabilities are included in the PSCR is based on CPFSK modulation.

The reconfigurable framework is a module that can configure itself based on the command sent to it by the Beagle Board server. For instance, if a command to enable push-to-talk is sent by the Beagle Board server, then the framework module configures its MAC and PHY layers to achieve this functionality. This module makes use of a knowledge map shown in Table 2.1 to identify the configuration settings of USRP for a particular mode (talk mode in this case). In the older version of PSCR on the GPP, this table was in a database, and if we were to use a database on the Beagle Board it would be an overhead. Therefore, this knowledge map is implemented using C++ standard template library hash maps. Retrieving information from these C++ hash maps is simple and has no overhead. The radio-config-talk xml file contains information that is used to configure the USRP(for example it may contain the daughter board that is used, number of sample per symbol etc). The waveform-talk xml file contains information like transmitter, receiver frequency and type of MAC layer, this information is needed to configure the PHY and MAC layers.

### 2.2.7 Object Oriented PSCR control application on the Android phone

The PSCR control application is built on the Android phone using object oriented approach. There are different classes, both custom and inbuilt, that make up the PSCR control application. Some of the important classes, data structures and xml files used to built the application are shown in Figure 2.9.

**Table 2.1** Knowledge map,showing the configuration files to be used, for a given channel group and modulation.

| **(ChannelGroup,Modulation)** | **(radioconfig,waveform)** |
|---|---|
| (frs,fm) | (radio-config-talk-fm-frs.xml,waveform-talk-fm-frs.xml) |
| (frs,bpsk) | (radio-config-talk-bpsk-frs.xml,waveform-talk-bpsk-frs.xml) |
| (efj,fm) | (radio-config-talk-fm-efj.xml,waveform-talk-fm-efj.xml) |

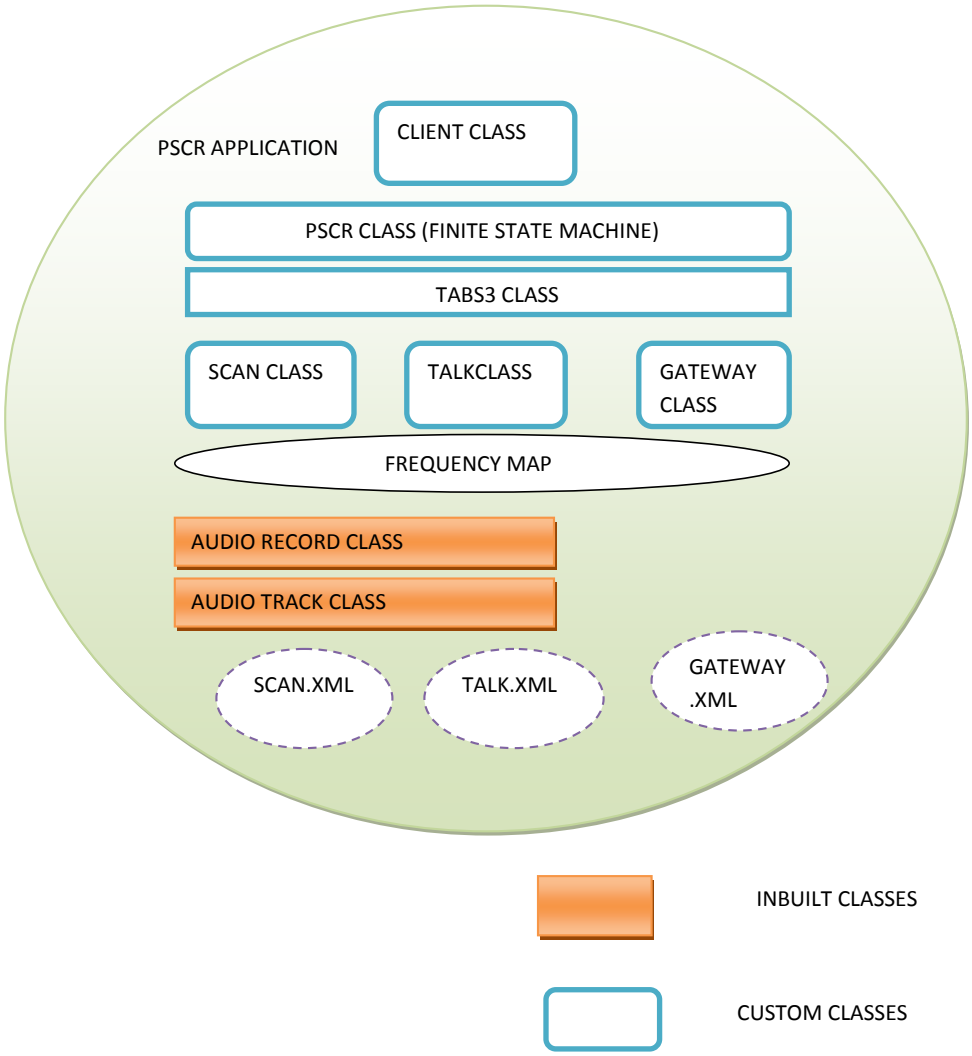


**Figure 2.9** PSCR-Android object oriented architecture.

The Scan.xml, Talk.xml and Gateway.xml defines the layout structure of the GUI for the tabs Scan, Talk and Gateway respectively. The PSCR Class stores the current state of the PSCR on the Beagle Board. Also, this class is responsible for bringing up the main GUI of the PSCR control application, when the application is selected on the Android phone. The PSCR control application icon as it appears on the Android phone is shown in Figure . 2.10


**Figure 2.10** PSCR control application icon.

Tabs3 class is called by the PSCR class to display the 3 tabs *Scan, Talk* and *Gateway*. Depending on which of the tabs is selected, the corresponding class is invoked. The Scan class is responsible for handling all the responses from the user under the Scan tab. The Scan Class is responsible for the following (i)Sending the scan command to the Beagle Board and displaying the results (ii) Sending commands to configure and control the Beagle Board PSCR for push-to-talk mode, based on the user's selection from the scan results (iii) Invoking the inbuilt audio record and audio track classes, to let the user talk in push-to-talk mode. These inbuilt classes are used for audio processing as shown in Section 2.3

The talk class is responsible for handling the user responses under the Talk tab. The talk class like scan class is responsible for configuring and controlling the Beagle Board PSCR for push-to-talk. The difference is that the talk class has a fixed set of public safety departments that it displays to the user for selection. The talk class also make use of the inbuilt audio record and audio track classes to enable push-to-talk. The gateway class is responsible for all the user actions under the Gateway tab. The Gateway class configures the Beagle Board

PSCR framework to the Gateway mode based on the users selection from the GUI. This class does not use the audio record and audio track classes. The client class is invoked by scan, talk and gateway classes every time the Android phone needs to communicate with the Beagle Board. UDP sockets are used for this purpose over a tethered USB connection. The Frequency Map is used by all the three classes to map the selections made on the GUI to a particular frequency, before sending it to the Beagle Board.

## 2.3 Audio processing

The different stages of audio processing from the Android phone to the Beagle Board and in turn to the USRP are shown in Figure 2.11. The voice from MIC of Android phone is first sampled at 11025 Hz and then the samples are encoded using 16 bit pulse coded modulation (i.e.) 16 bits per sample. After this step the samples are written into a buffer. These functionalities are achieved using the audio recorder module that is supported by the Android phone. Once this is written into the buffer, it is read from the buffer using the audio track module of the Android phone. Data is read from the buffer by the audio track module, as and when it is written into the buffer by the audio recorder module. Therefore, it behaves as a continuous stream. The encoded samples read from the buffer are sent to the Android phone speakers using the audio track module. These samples are then routed to the MIC of the Beagle Board using a cable that connects Android phones speaker with MIC of Beagle Board as shown in Figure 2.12.

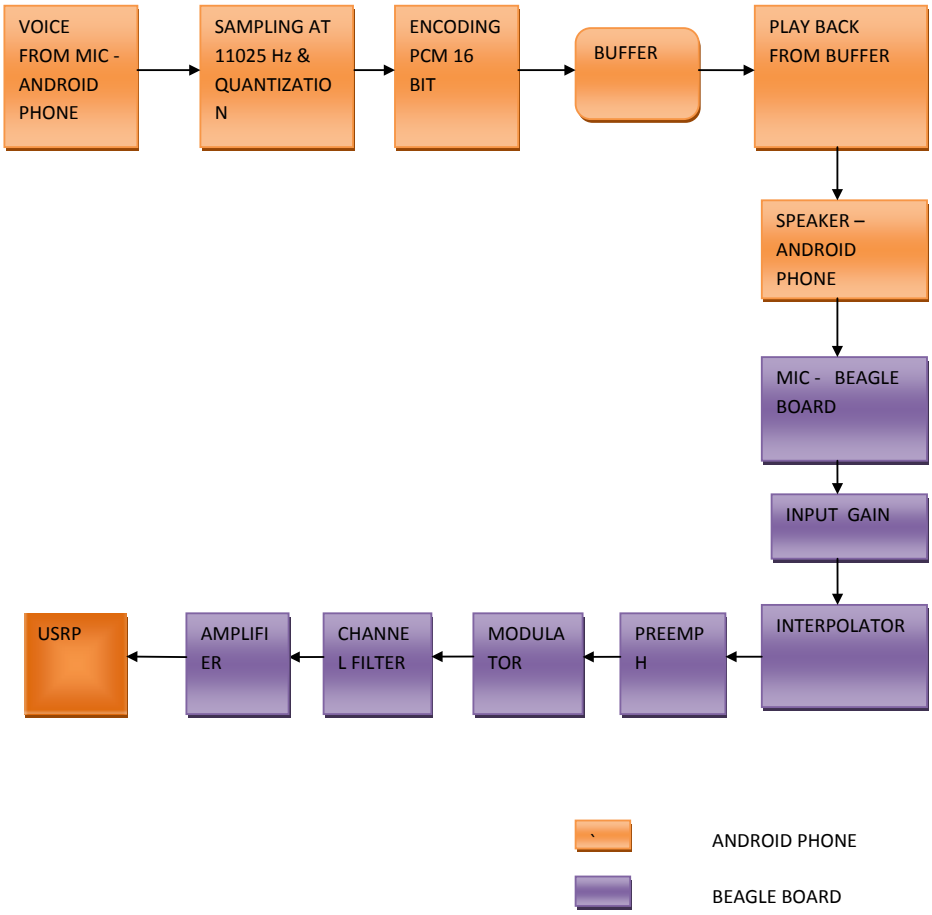


**Figure 2.11** Audio processing stages from the Android phone to the Beagle Board.

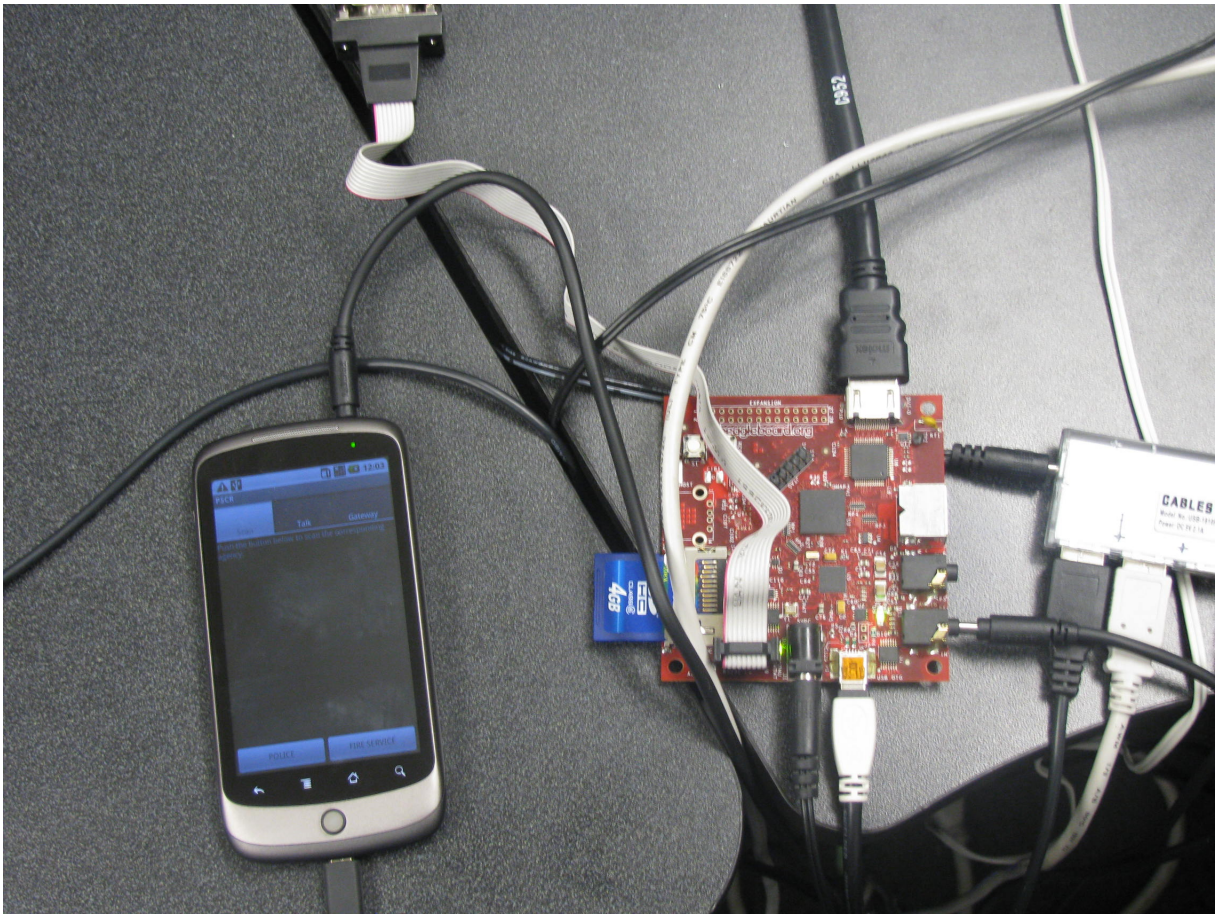


**Figure 2.12** Beagle Board - Android Phone.

Now these samples received by the MIC of the Beagle Board act as the audio source for the GNURADIO signal flow graph that was created by the framework module based on the commands from the Android phone. The first stage of this flow graph increases the gain of the audio signal using the gr.multiply_const_ff gnuradio block. Second, the gr.interp_fir_filter_fff interpolator block is used to increase the sampling rate of the incoming signal to match the sampling rate of the ADC. Third, pre-emphasis is done to improve the signal to noise ratio (SNR) of the audio signal using the fm_preemph block. Fourth, the gr.frequency_modulator_fc block is used to modulate the data to FM (frequency modulation) by giving the center frequency

(used for transmission) as input to this block. Fifth, gr.fir_filter_ccf gnuradio block is used as channel filter to limit the bandwidth of transmission. This channel filter is implemented in the DSP and hence a call to this block will be routed to the corresponding DSP block as shown in Section 2.2.2. Finally, the signal is amplified using the gr.multiply_const_cc block and sent to the USRP for transmission.

The audio signal received by the USRP connected to the Beagle Board is played back using the Beagle Board speakers and is not sent back to the Android phone. The gnuradio flow graph that is used to receive the audio signal and play back is shown in Figure 2.13. The channel filter (gr.fir_filter_ccf gnuradio block) in the flow graph filters out the signals that are out of the bandwidth of interest. The low noise amplifier ( gr.multiply_const_cc block) amplifies the signal. The amplified signal is then demodulated using the fm_demod gnuradio block. The demodulated signal is then filtered using the audio filter (gr.fir_filter_fff block) and sent to the de-emphasis (fm_deemph gnuradio block) module to improve the SNR. The audio signal is then sent to the speakers (connected to the audio out of the Beagle Board) for playback via a volume control module (gr.multiply_const_ff gnuradio block).

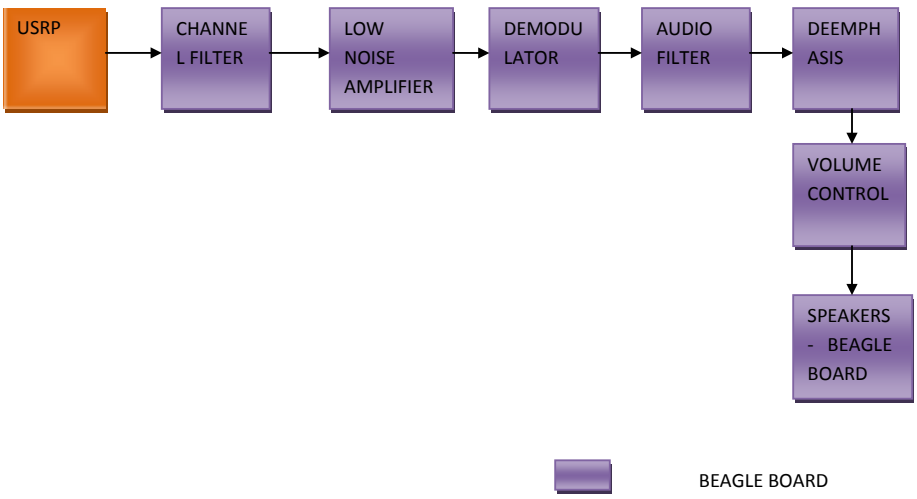


**Figure 2.13** Audio processing stages from the USRP to the Beagle Board speakers.

## 2.4 Beagle Board - Android phone interface communication protocol

The communication protocol is used to exchange information across the Beagle board - Android phone interface. The function calls and the socket communication that exists between the Beagle board - Android phone interface and also within the Beagle Board and Android phone are shown in Figure 2.14.

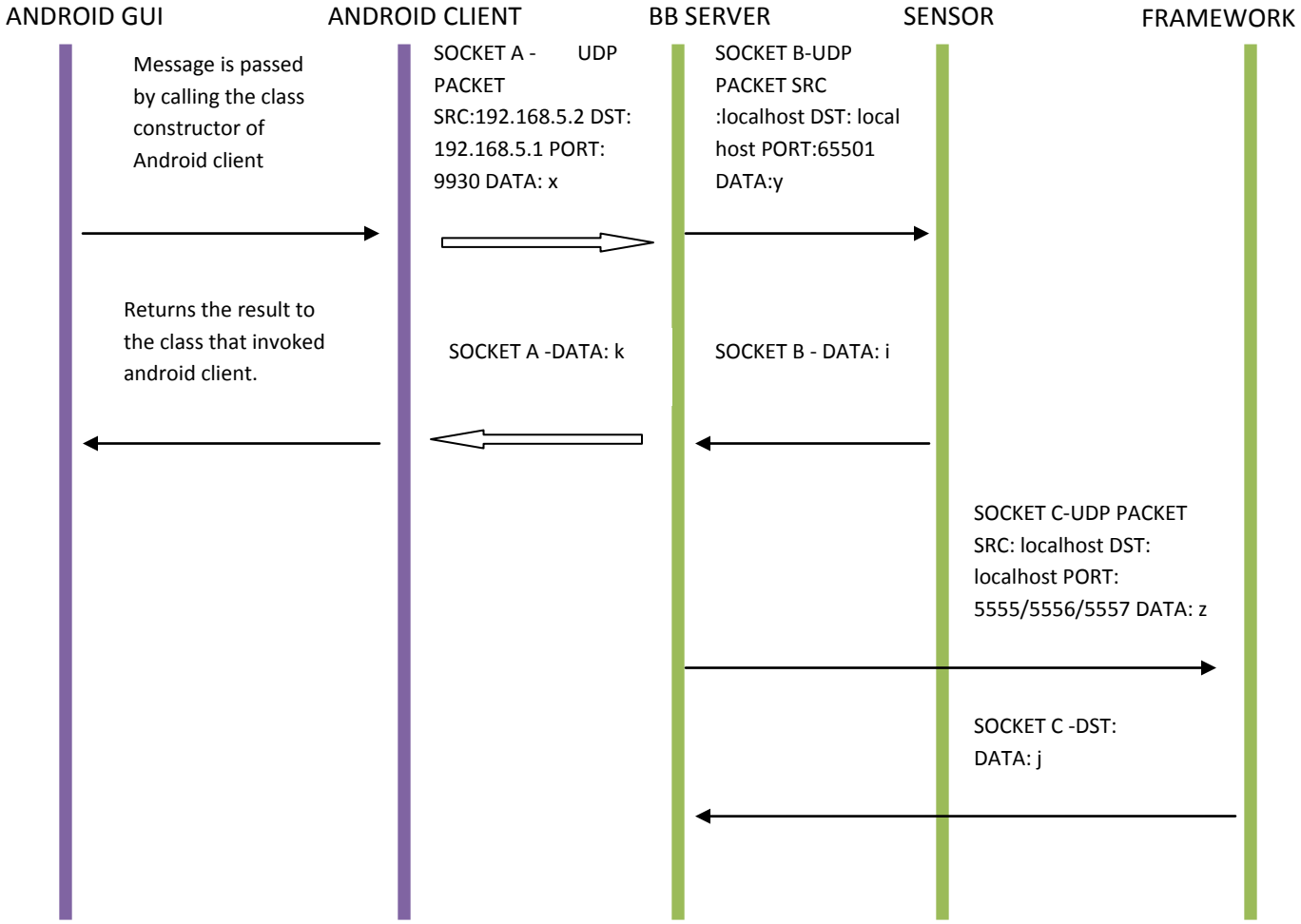


**Figure 2.14** Beagle Board - Android phone communication interface.

Every command given by the user from the Android GUI, instantiates the Android client class. This class is used to send the command to the Beagle Board server, using the tethered USB connection. Based on the command sent from the Android phone, the Beagle Board server constructs the appropriate command and sends it to the respective port. When the

command from the Android phone is to scan a particular frequency band, then the commands are sent by the Beagle Board server to port 65501 using UDP sockets. The Beagle Board sensor that keeps listening on port 65501 receives this command and starts scanning. If the command from the Android phone is to establish push-to-talk mode or to enable a gateway between two departments, then the command is sent by the Beagle Board server to port 5555. The framework server that is listening on port 5555 receives this command and configures the framework.Configuring the framework involves (i)setting up the parameters of the USRP like frequency, modulation, etc. (ii)forming the flow graph necessary to enable talk/gateway. Once the framework is configured, the framework server opens the port 5556 and starts listening on this port. To start talking, the command is sent by the Beagle Board server to the port 5556, the framework server upon receiving this command starts the flow graph. Now the framework server opens port 5557. When the command from the Android phone is to toggle between the talk and listen (in the push-to-talk mode) , the command is sent by the Beagle Board server to this port. And, when the command from Android phone is to stop the gateway mode or push-to-talk mode the command is sent by the Beagle Board server to the port 5556.

There are three main tabs in the Android PSCR GUI from which commands can be issued. They are Scan, Talk and Gateway. A brief explanation for each of these tabs is given in the following sections.

### 2.4.1 Scan tab

. The main purpose of this tab is to help the user in identifying other public safety personnel in the vicinity and to communicate with them. As shown in Figure 2.15, the scan tab has two options: Police and Fire service. When the Police button is pressed, the frequency

band corresponding to Police department is scanned and reported. Similarly, when the Fire service button is pressed, the frequency band corresponding to the Fire service department is scanned and reported. In our setup for illustration purposes, the frequency band from 462MHz to 463MHz is treated as Police frequency band and 467MHz to 468MHz is treated as Fire service frequency band.


**Figure 2.15** Scan tab.

Now let's see the sequence of operations that take place when one of these buttons is pressed. Once a button is pressed, the command is sent to the Android client, which in turn sends the command to the Beagle Board server using the tethered USB connection. The Beagle Board server then constructs the command to scan the frequencies and sends it to the Beagle Board sensor. The Beagle Board sensor scans the required frequency bands using the USRP RF front end and then writes the results into an xml file. After writing the results to the xml file the Beagle Board sensor sends an acknowledgement to the Beagle Board server. Now, the Beagle Board server parses the xml file and wraps the results in the format required to

communicate with the Android phone. After wrapping up results in the required format the results are sent to the Android phone using the tethered USB connection. This sequence of operations is shown in Figure 2.16

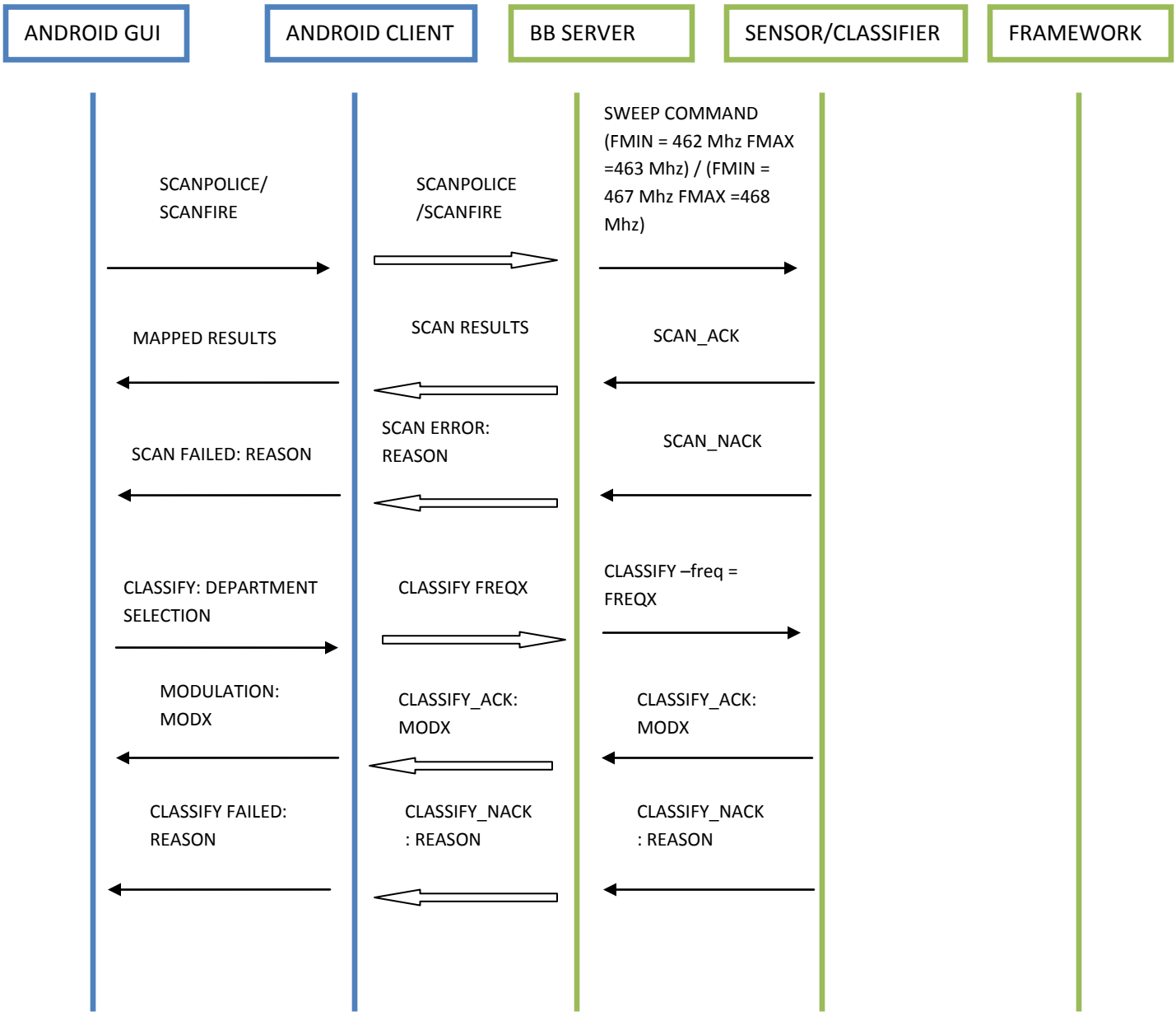


**Figure 2.16** Scan mode: sequence of commands used to scan/classify.

Once the Android phone receives the scan results it parses the results and looks them up in the internal map to identify the department to which the band belongs. An example of the

internal map in the Android phone is shown in Table 2.2.

**Table 2.2** Example mapping between departments and frequencies

| DEPARTMENT | FMIN | FMAX |
|---|---|---|
| VT POLICE | 462500000 | 462650000 |
| BB POLICE | 462650000 | 462750000 |
| VT FIRESERVICE | 467500000 | 467650000 |
| BB FIRESERVICE | 467650000 | 467750000 |

After identifying the departments to which the scan results belong, it displays this information to the user as shown in Figure 2.17.


**Figure 2.17** Scan tab: When Police frequency band is scanned,When Fireservice frequency band is scanned

Depending on which department the user wants to communicate with, he/she can select the

department. This brings up a popup menu that has 3 buttons: namely talk/over, stop, and classify as shown in Figure 2.18.


**Figure 2.18** Scan tab: popup menu.

When talk button is pushed, the command is sent to the Android client. The Android client needs to issue two commands in order to setup the Beagle Board PSCR framework. First, the Android client sends the TALK_CONNECT command along with the waveform type, channel group, modulation and frequency to the Beagle Board server. The Beagle Board server then, using this information, constructs a command and sends it to the Beagle Board PSCR framework. An example for this command is waveform fm frs 462500000. Here fm is the modulation type, frs is the channel group and 4625000000 is the frequency used to communicate. The PSCR framework now constructs the flow graph using GNURADIO signal processing blocks. If the framework configuring procedure is successful, acknowledgement is sent to the Android client.

The Android client now sends the second command TALK_PTT_PRESSED to begin talking. The Beagle Board server in response to this sends a command "start" to the framework. If successful, the acknowledgement is sent to the Android client and the message "Start Talking" is displayed in the Android GUI. Now, if the user intends to listen, then he/she has to press the over button. This command is sent to the Android client, which then sends TALK_PTT_RELEASED to Beagle Board server. The Beagle Board server in response to this sends command "over" to the framework and if successful, "Listening" is displayed on the Android GUI. If the user intends to stop the connection, the user has to press the "stop" button and if successful "Connection Released" is displayed on the GUI. If configure/start/stop/over commands fail for some reason then the corresponding error message is displayed on the Android GUI. This procedure is shown in Figure 2.19 and Figure 2.20.

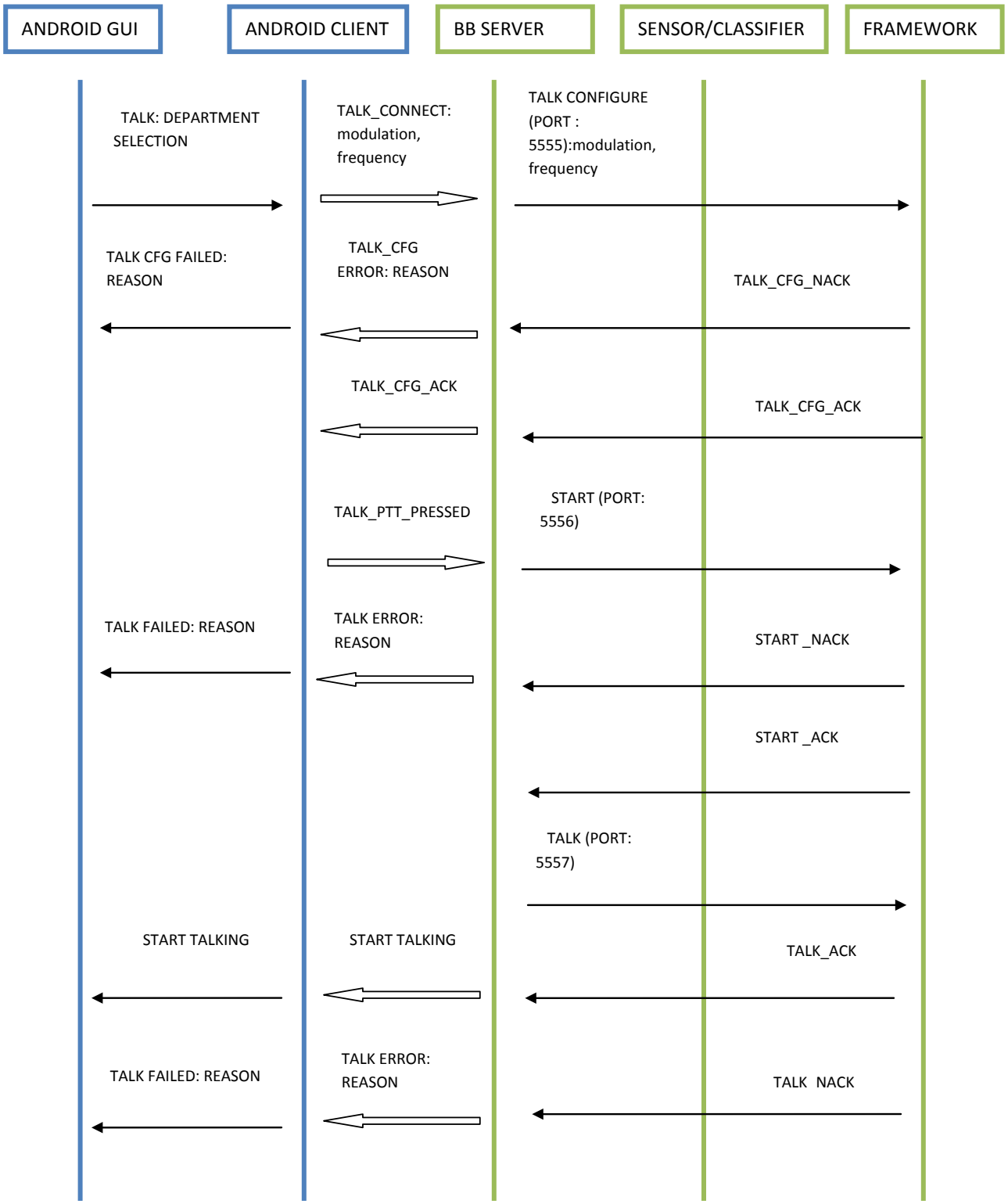


**Figure 2.19** Talk mode: connection establishment procedure.

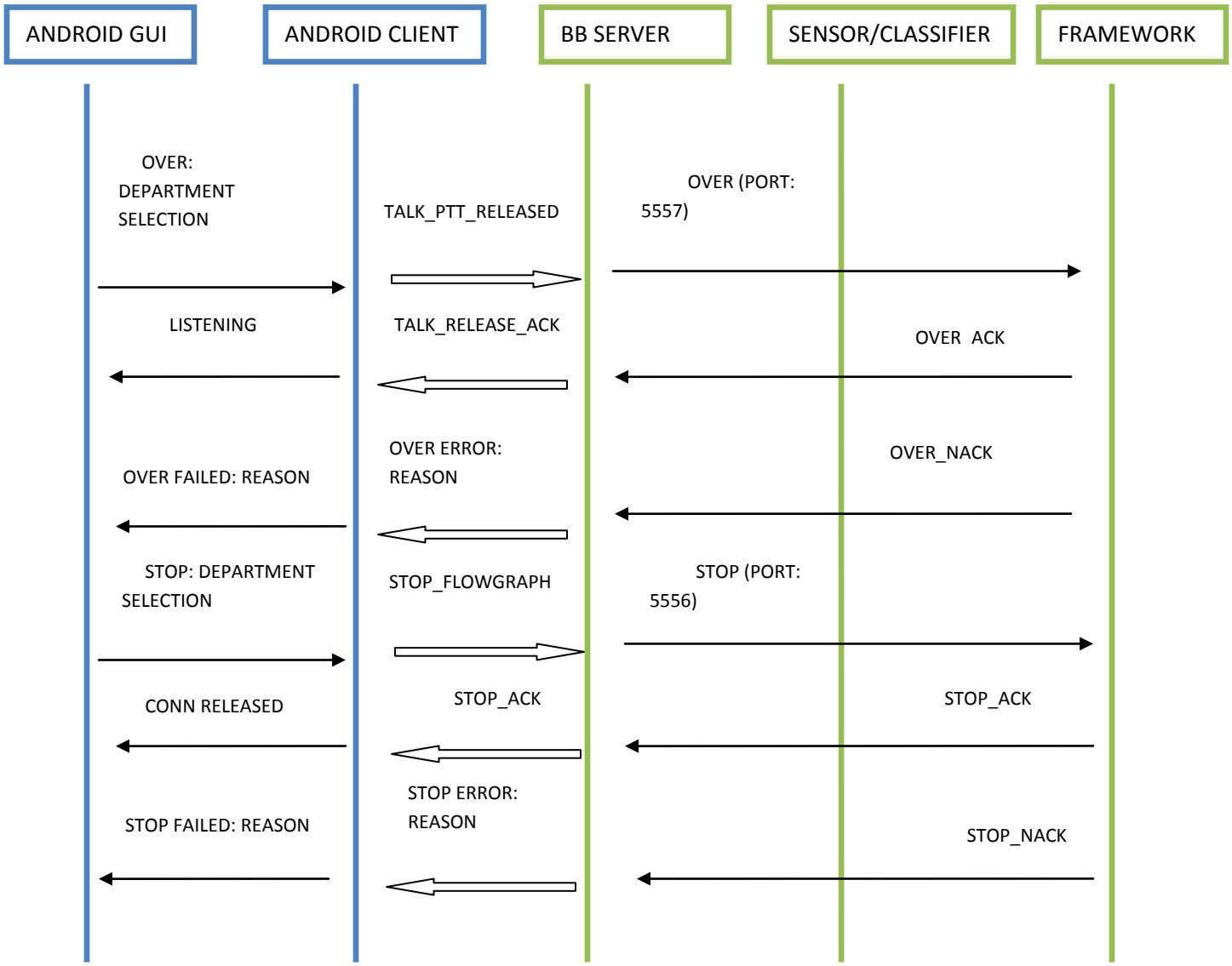


**Figure 2.20** Talk mode: stop/over commands

### 2.4.2 Talk tab

The talk tab is used to communicate with one of the preloaded set of public safety personnel/departments displayed on the Android GUI. A screen shot of the Talk tab is shown in Figure 2.21. The GUI has one icon for each department; namely police, fire, etc. Depending on the icon selected the talk mode is established by sending a command to the Beagle Board as explained before. The sequence of commands is the same as the establishment of push-to-talk mode under the scan tab.

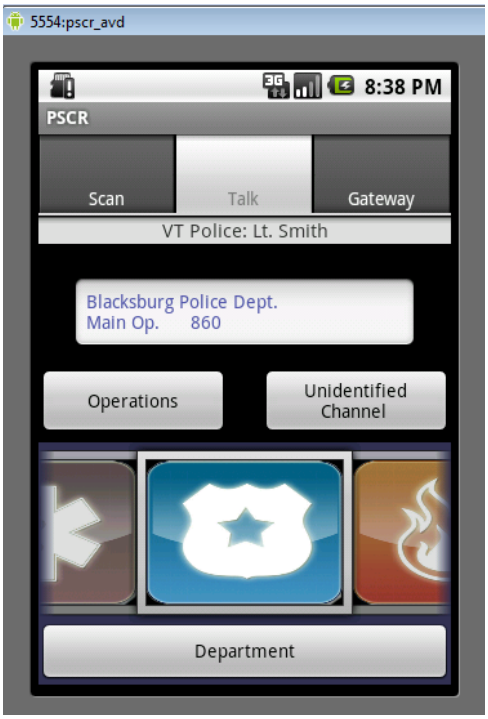


**Figure 2.21** Talk tab.

### 2.4.3 Gateway tab

The Gateway tab shown in Figure 2.22 is used to configure the framework, to act as the gateway between different public safety departments. For instance, the framework can act

as the gateway between two waveforms having different centre frequency/modulation etc.

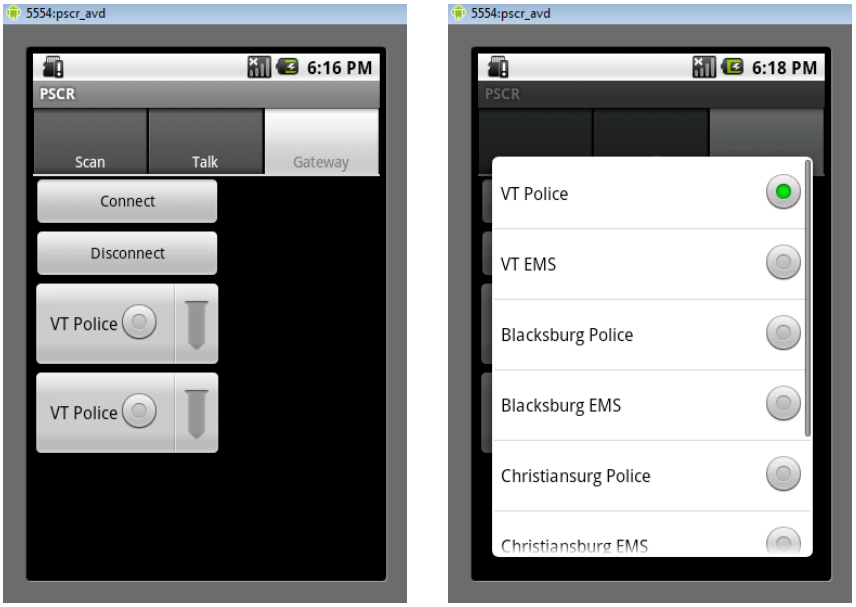


**Figure 2.22** Gateway tab, Spinner menu on the gateway tab

The two departments for which the framework is configured are selected from the Android GUI. Once the two departments are selected, the connect button is pressed to send the command to the Beagle Board and configure the framework. The Android client converts the departments selected in the Android GUI into appropriate waveform type, frequency, etc and sends it to the Beagle Board server. The Beagle Board server then constructs the command and sends it to the framework and configures it. Upon successful completion of this sequence of operations "Connection established between department 1 and department 2" is displayed on the Android GUI. If there is any failure during this sequence of operations, the appropriate failure message is displayed to the user. This sequence of operations is shown in Figure 2.23

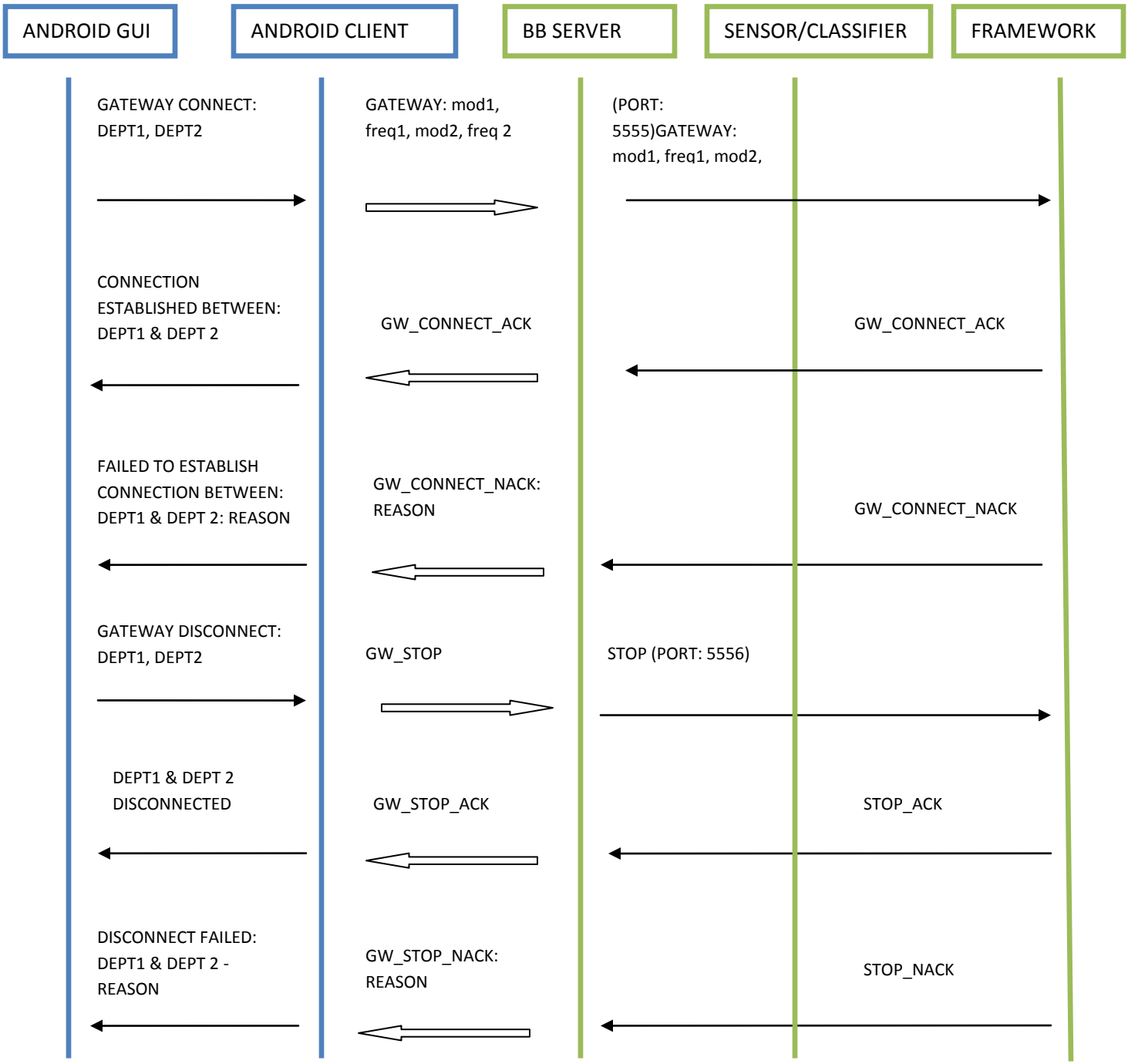


**Figure 2.23** Gateway mode: sequence of commands used to connect/disconnect gateway.

### 2.4.4 PSCR state diagram

There are 5 possible states for the PSCR on the Beagle Board, and the current state of the PSCR is maintained in the Android phone. The current state and state transitions are tracked in the Android phone to avoid the redundant communication between the Android phone and the Beagle Board. The state diagram is shown in Figure 2.24.

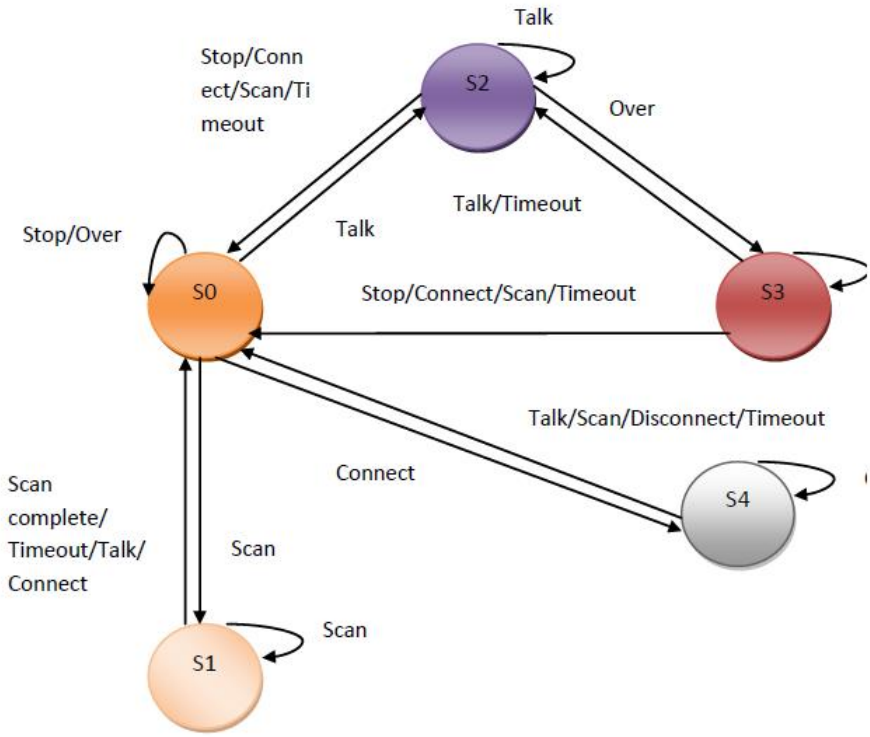


**Figure 2.24** Framework state diagram.

In state S0, the PSCR is not configured yet and waits to receive command from the Android phone. When in state S0, if the user issues stop or over command from Scan/Talk tab, "No Connection Exists" is displayed to the user and the PSCR state is maintained in state S0. If a talk command is issued by the user, then the command to configure the framework is sent

to the Beagle Board server listening on port number 9930. The Beagle Board server then constructs the command to configure the framework and sends it to port 5555, where the PSCR framework is listening to receive commands. If the PSCR framework gets configured successfully, then the acknowledgement is sent to the Android phone by the Beagle Board server. Once the acknowledgement is received by the Android phone the PSCR framework state is updated to S2 and the user can begin talking.

If the user issues scan command in state S0, then the command to begin scanning is sent to the Beagle Board server and the PSCR state is updated to S1. Similarly, if the user issues connect command in state S0, then the command to configure the PSCR framework in gateway mode is sent to the Beagle Board. After receiving acknowledgement for the connect command the PSCR state is updated to S4.

When in state S2, if the user issues command "talk" to connect to the already connected public safety personnel, then "Already in talk mode with this personnel" is displayed to the user. If command "stop" is issued for the connected public safety personnel, the command is sent to the Beagle Board server and in turn to the PSCR framework waiting on port 5556. Now the PSCR framework is stopped and it starts listening on port 5555 to receive configuration commands. The PSCR framework state is updated to state S0. If "over" command is issued for the connected public safety personnel, then the command is sent to port 5557 in Beagle Board and the PSCR framework is configured to listen mode. The PSCR framework state in Android phone is updated to state S3.

When in state S3, if the user issues over command for the connected public safety personnel, "Already in listen mode with this personnel" is displayed to the user and the state is retained. If a stop command is issued for the connected public safety personnel, then the PSCR framework is stopped and the PSCR framework state is updated to S0. When Talk command is issued for the connected public safety personnel, then the PSCR framework is

configured and moved to talk mode and PSCR framework state in Android phone is updated to S2. Whether in state S2/S3, if the user issues talk command for any public safety personnel/department other than the one that is currently connected, stop command is issued for the existing connection. After stopping the existing connection, the PSCR state is updated to state S0. Once the previous connection is stopped, the talk command is sent to configure the PSCR framework with new settings. Once the acknowledgement is received from the Beagle Board server, the PSCR state in the Android phone is updated to S2. Similarly, when in state S2/S3 if the user issues over/stop command for any other public safety personnel other than the one that is currently connected, then "No connection exists" is displayed to the user and the PSCR framework state is maintained.

When in state S1, if the user again issues scan command, then the state is retained and the scan command is sent to the Beagle Board. In this state if the user issues connect/talk command then the scanning is stopped and the PSCR state is updated to state S0. After stopping the scanning, the appropriate command is sent to configure the PSCR framework in gateway/talk mode and the PSCR state is updated as explained above.

In any of these states if a timeout occurs due to the delay in response from the Beagle Board, then the PSCR state is moved to state S0 as shown in Figure 2.24.

## 2.5 FM/AM/CPFSK classification

In order to classify a particular signal, the classify command is sent to the sensor/classifier waiting on port 65501 on the Beagle Board. The sensor/classifier scans the given frequency and then classifies it using the algorithm shown in Figures 2.25 and 2.26.

As a first step, the phase difference between the adjacent points in the results from the scan

operation is calculated.Then the count of the number of phase difference that are greater than threshold1 is calculated. If this count is greater than threshold2 then the signal is phase discontinuous and if less than threshold2 it is phase continuous. If its phase is continuous, the modulation of the signal is AM, FM or CPFSK. To check if the modulation type of the signal is AM, the standard deviation of the amplitude of the signal is compared to threshold3 and if it is greater, then the modulation is AM. If the standard deviation is less than threshold3 then the modulation is FM or CPFSK. Now, to check if the modulation of the signal is FM, we use the formula shown in [23] Figure 2.27.

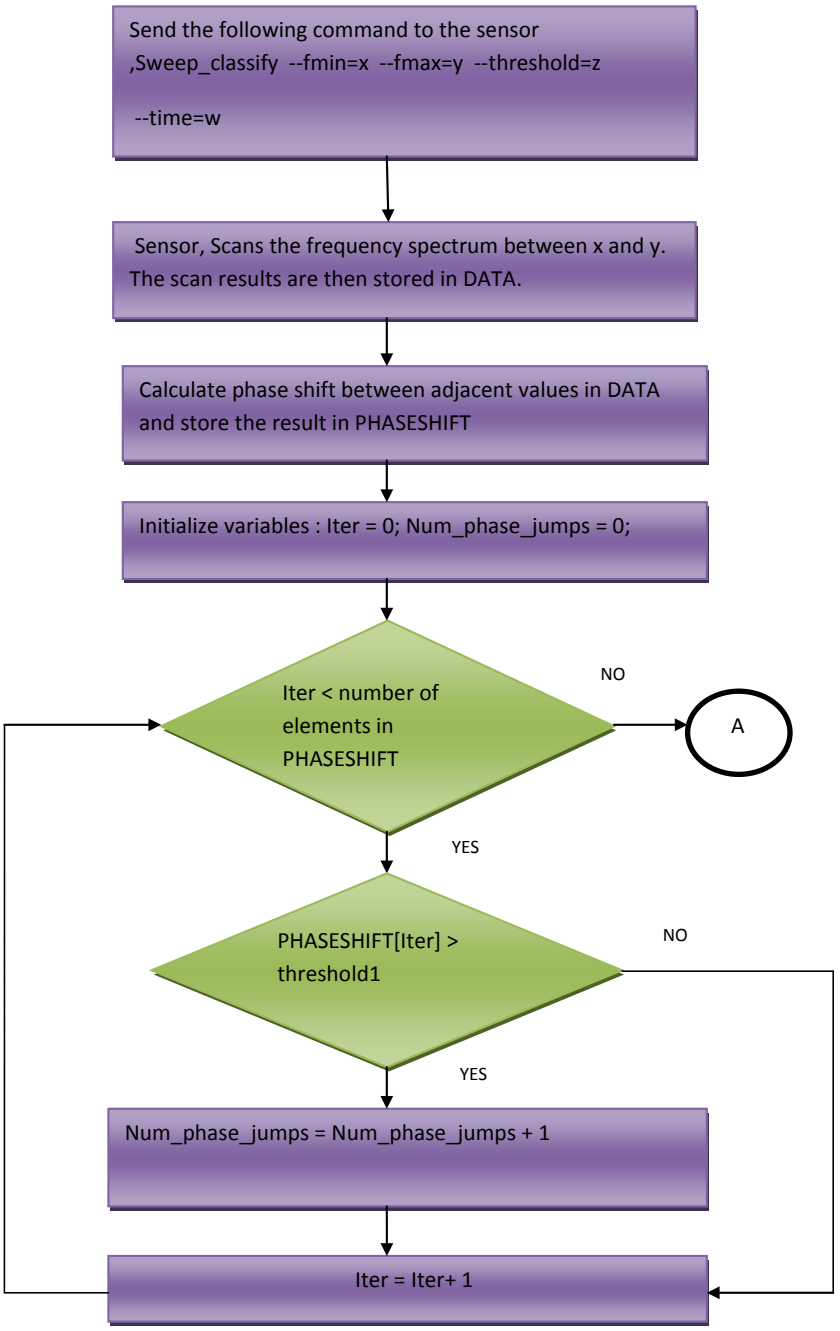


**Figure 2.25** Classify algorithm, part 1.

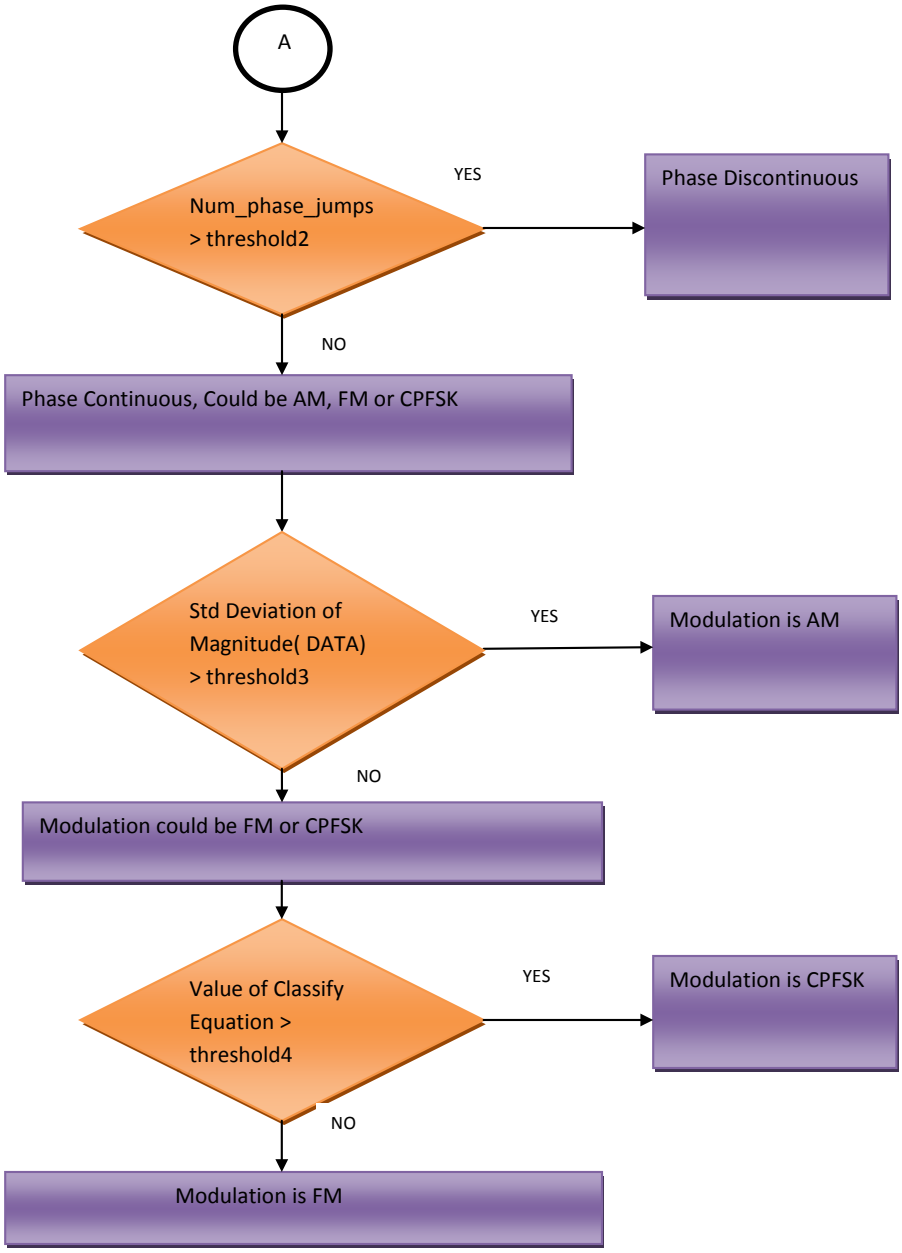


**Figure 2.26** Classify algorithm, part 2.

$$\frac{\sigma_{ti}}{\mu_{ti}} = \frac{\sqrt{E[(ti(t) - E[ti(t)])^2]}}{E[ti(t)]}$$

**Figure 2.27** Formula used for calculating the ratio that determines if a signal is FM or CPFSK.

In this formula, ti(t) = the time interval between the adjacent zero-crossing points in the signal. The numerator is the variance of ti(t) and the denominator is the mean of ti(t). If the value of this ratio is less than threshold4, then the modulation of the signal is FM and if value of ratio is greater than threshold4, then the modulation is CPFSK. The value of threshold1, threshold2, threshold3 and threshold4 are calculated based on experiments. These values depend on the radio environment and the reception quality of USRP. Based on the experiments conducted in the CWT lab, the values of threshold1, threshold2, threshold3 and threshold4 are 30, 1400, 0.1, 0.03 respectively.

# Chapter 3

# Cooperative Spectrum Sensing to achieve Dynamic Spectrum Access

## 3.1 Dynamic Spectrum Access

Cognitive Radios (CR) have gained increasing popularity in improving spectrum utilization. CR can configure themselves to transmit/receive in any frequency as per need. Traditionally, the frequency bands are allocated to users by the Federal Communications Commission. These users who are licensed to use the band are called primary users. Although the primary users have the license to use the band, they don't use the frequency band all the time. This results in under-utilization of the frequency spectrum. This issue can be addressed using the DSA techniques based on Cognitive Radios. DSA suggests the allocation of the licensed band to other users (generally called secondary users) during the absence of the primary user. The secondary users must vacate the frequency band on primary users return. In order to achieve this, there are several different functions that are performed [28], (i)Identifying the frequency bands that are not used by the primary users. (ii)Once the unused frequency

bands are identified, selecting the best available frequency band based on the secondary user's need. (iii)Distributing the unused frequency band information and coordinating the spectrum sharing among the several different users. (iv) Constantly monitoring the frequency bands to identify the return of primary users and to alert secondary users to vacate the frequency band.

To achieve these functionalities we make use of a cooperative spectrum sensing technique along with a DSA broker [29]. Cooperative spectrum sensing is used to overcome the failures of individual signal sensors under conditions like shadowing, multipath and interference [30]. Also, the availability of channels varies geographically and the users must be given the right set of vacant channels to communicate among each other. For this reason the sensors are geographically distributed to give the complete picture of the radio environment in the given region. The data from all the sensors must be combined in a centralized system which can then analyse the data and identify the vacant channel. Once the vacant channel is identified, the information is sent to the secondary users to enable communication among them. As all these sensors keep sending the RF information to the DSA broker regularly, the DSA broker is aware of the presence/absence of the primary user at any point in time. This knowledge allows the DSA broker to alert the secondary user on return of the primary user.

## 3.2 Sensor Architecture

In a cooperative spectrum sensing environment low data latency is critical and this is taken into consideration while designing the architecture of the sensor. The sensor architecture is shown in Figure 3.1.

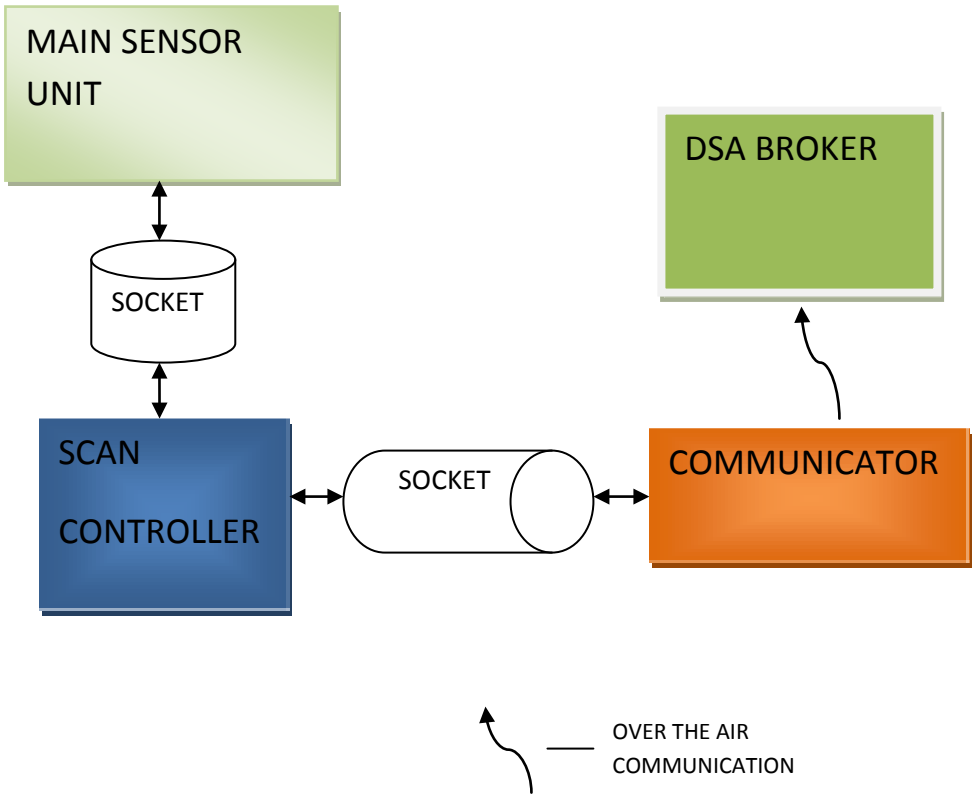


**Figure 3.1** Sensor architecture.

The main sensor unit opens a socket and waits for the command from the scan controller, which tells the sensor the frequency range to scan and the threshold used to detect the signals. Once the main sensor unit has scanned the frequency bands as per the request of the scan controller, the results are written to an xml file and an acknowledgement is sent to the scan controller. Now the scan controller parses the information in the channels.xml file and sends it to the communicator module. The communicator module then embeds this information into a packet structure that is used in the cooperative spectrum sensing network. The packet structure is shown in Figure 3.2.

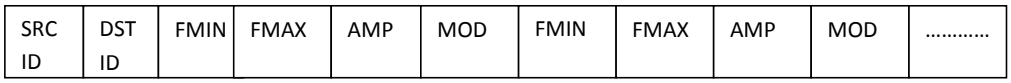


**Figure 3.2** Sensor packet format.

The SRC ID is the identifier of the sensor itself. The range for the SRC ID is between 1 and the number of sensors present in the network. The DST ID is the identifier of the intended destination. Typically this field has a value 0 which is the identifier for the DSA broker. The FMIN, FMAX are the minimum and maximum frequencies of the frequency band, AMP is the amplitude of the signal in the frequency band, MOD is the modulation of the signal. Once the packet structure is formed, the next step is to send it to the DSA broker. For this the USRP RF front end based radio channel / wifi can be used as the control channel. If USRP front end is used as the control channel, then a particular frequency is fixed as the control channel. Since all of the different sensors may try to send the data to the DSA broker at the same time, Carrier Sense Multiple Access technique is used to resolve the conflict. Once it is seen that there is no conflict with other sensors, the data is sent to the DSA broker. If there is conflict for a duration longer than a specific threshold, the data is ignored and the scan procedure is repeated. If the data is sent ignoring the conflict duration, then the information sent is stale and this impacts the judgement of the DSA broker in identifying the unused spectrum. To ensure low data latency, the communicator module has an inbuilt buffer that stores the data previously sent to the DSA broker. Every time the communicator module receives data from the scan controller, the new information is verified against the information in the buffer. If and only if there is a difference in information above a certain threshold, this new data is sent to the DSA broker. And after sending this new data to the DSA broker, the buffer is updated with this new data. The flow chart shown in Figure 3.3 describes this algorithm.

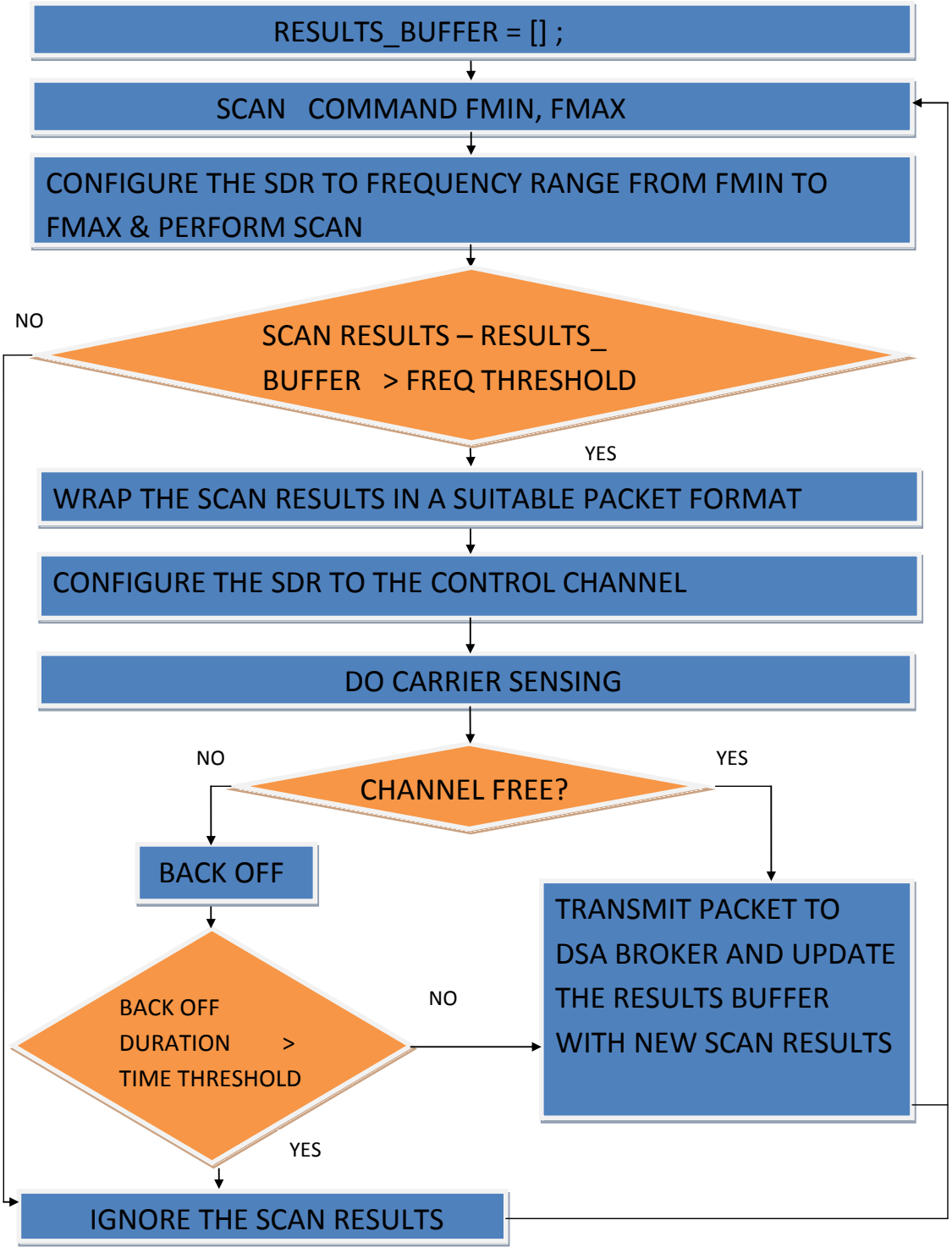


**Figure 3.3** Sensor flowchart.

Now let's see the significance behind doing this. Say that the communicator module does not have a buffer and that it tries to transmit every time the scan controller sends data to it. This results in constant demand for the control channel by all the sensors in the Network. The consequence is longer back off duration that exceeds the threshold and no information is sent to the DSA broker.

### 3.2.1 Main sensor unit

#### Signal Detection

Signal detection is one of the important [24] concepts that help the cooperative spectrum sensing network to detect the primary user and make sure the secondary user does not interfere with the primary user. Therefore, the signal detection time determines the performance of the cooperative spectrum sensing network.

If the signal is known *apriori*, then the well known matched filter technique can be used to detect the signal. The matched filter is the optimal method for signal detection as it maximizes the SNR of the signal that was received. Though matched filter is the optimal signal detection method, we need to have *apriori* knowledge of the primary user signal [25]. This means that the signal detector needs to know about the modulation type, packet format and apart from these it needs to achieve time and carrier synchronization with the primary user signal. All of these may be possible, because most primary users have a fixed pilot signal, a preamble or other parameters that enable synchronization. But, in spite of the feasibility of such a matched filter based signal detector, the main drawback is that sensor unit needs to have a separate receiver for each type of primary user signal.

For this reason, we use the simpler non-coherent detection method [25] [24]. The basic non-

coherent energy detection method is shown in Figure 3.4. This method as seen does not require synchronization or *apriori* knowledge of the signal. The received analog signal is filtered using a band pass filter that only allows the signal in the band of interest. Then the analog signal is converted to digital signal using the ADC. The resulting signal is squared and averaged over T samples and this gives the energy of the signal. This value is tested against a threshold and if it is greater than the threshold, then we say that the signal is detected. This method is not so flexible, especially in the case of narrow band signals because of the filtering that is done at the initial stage. This drastically limits the bandwidth that can be detected at any point in time.

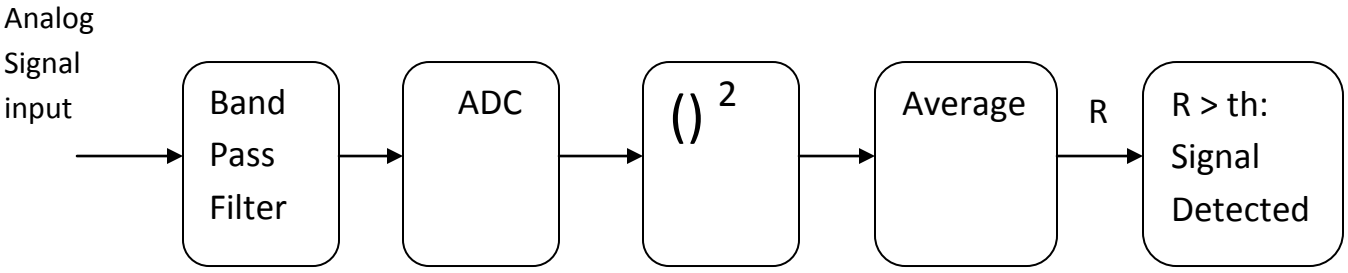


**Figure 3.4** Basic energy detection technique.

To overcome this bandwidth limitation, we use the following energy detection technique that is based on Welch periodogram [26] [24] as shown in Figure 3.5. The band pass filter that was used in the previous technique is removed; this takes care of the bandwidth limitation. Then the FFT is applied to the digital signal that comes out of ADC and magnitude square of FFT is taken. The result is then averaged and tested against the threshold.

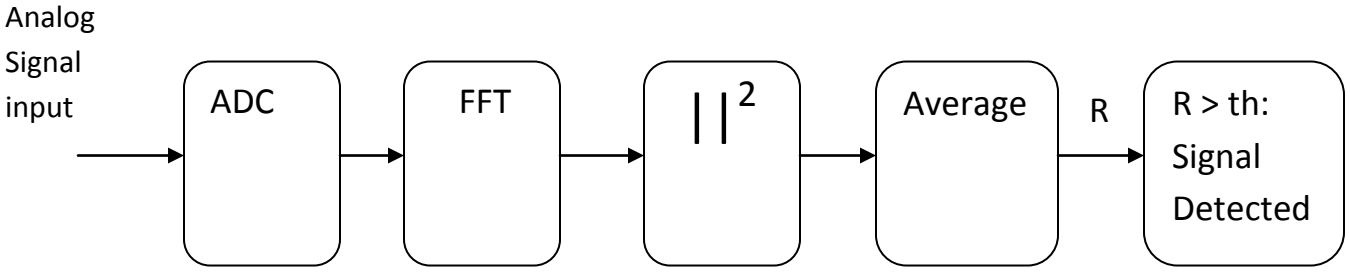


**Figure 3.5** Energy detection based on Welch periodogram.

# Chapter 4

# Using the Beagle Board based PSCR in a Dynamic Spectrum Access System

## 4.1 Introduction

DSA, as explained previously, is the technique used to improve spectrum utilization. This technique suggests the allocation of a licensed band to the secondary users in the absence of the primary user. The secondary users must vacate the frequency band on primary user's return. In order to achieve this, the Beagle Board based sensor/classifier is used to scan the radio environment and detect the presence of the primary user. This information is sent to the DSA broker which then allocates the free spectrum to the secondary users. The DSA broker is also responsible for notifying the secondary users to vacate the frequency band, on primary user's return.

## 4.2 DSA broker

The architecture of the DSA broker is shown in Figure 4.1.

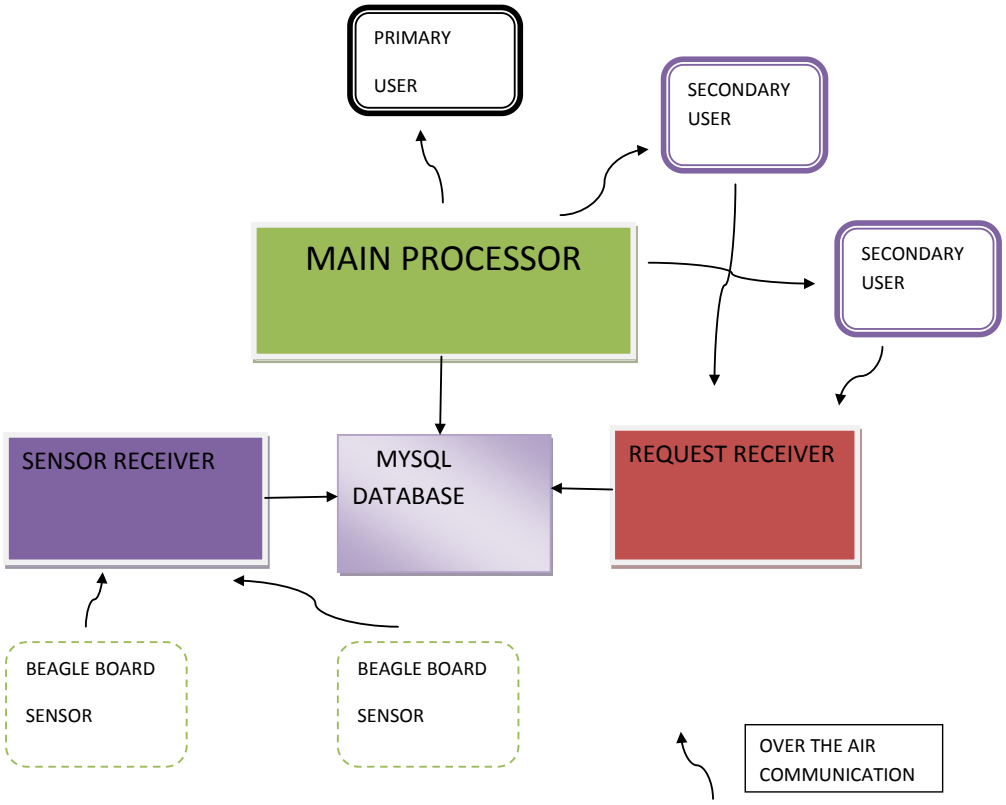


**Figure 4.1** DSA broker architecture.

The sensor receiver receives data from all the different sensors. Based on the sensor from which the packet was received it constructs the SQL Query to populate the corresponding MySQL table. For example: sensor1, sensor2 are the tables that are used to populate the information sent from sensor1, sensor2, respectively. Another important function of the sensor receiver is to populate the sensory summary table shown in Table 4.1 by combining the information from all the sensors.

Now let's see how the sensor receiver populates the sensory summary table: (i) Say that the sensor receiver received data from sensor1; now the old contents from sensor1 are removed from the sensor summary table by sensor receiver. (ii) Once the old contents of sensor1 are removed from the sensor summary table, the new contents of sensor1 are inserted into the summary table. (iii)This procedure is repeated for every sensor for which sensor receiver received data.

**Table 4.1** Example showing the summary of information received from all the sensors

| **FMIN** | **FMAX** | **FC** | **BANDWIDTH** | **POWER** | **MODULATION** | **SensorName** |
|---|---|---|---|---|---|---|
| 460000000 | 462000000 | 461000000 | 2000000 | 10 | FM | Sensor-1 |
| 464000000 | 465000000 | 464500000 | 1000000 | 10 | unknown | Sensor-2 |

The request receiver receives request for channel from the secondary users and stores the ID of the secondary users in the secondary user registration table shown in Table 4.2.

**Table 4.2** Secondary user registration table

| **Secondary user ID** | **Allocated center frequency** | **Bandwidth** |
|---|---|---|
| 1 | 462562500 | 2500 |
| 2 | 462712500 | 2500 |

The main processor is the core of the DSA broker which processes the information in the sensor summary table and populates the spectrum information table shown in Table 4.3. The main processor identifies the free frequency or frequencies based on the information in the sensor summary table. The main processor has a fixed frequency range within which it looks for the free spectrum. Let's assume that 460 MHz to 468 MHz is the range of frequencies within which the main processor looks for free spectrum. All frequency bands within this range that are not listed in sensor summary table are considered to be free spectrum. These

entries are inserted into the spectrum information table and have the occupancy column value of 0. Before any entry is moved into the spectrum information table from the sensor summary table, it is classified as a secondary user or a primary user. The parameters that decide whether a user is a primary user or a secondary user are configurable. In the example shown, users with frequency modulation are referred to as primary users and users having any other modulation type are referred to as secondary users. Once this classification is made, these entries from sensor summary table are inserted into the spectrum information table with the occupancy column value of 1 and user type column value of primary/secondary. From the contents of spectrum information table, those frequencies that have an occupancy value of 0 are distributed among the secondary users. Now that the secondary users have the unoccupied frequency band information, they can begin communication. Meanwhile, the main processor continues to process the summary table and populate the spectrum information table. At any point in time if the main processor sees that the primary user has returned to the frequency band that was allocated to the secondary user, it notifies the secondary user. Now the secondary user moves to a different frequency band, allocated by the DSA broker and continues communication or if no other frequency band is available, then terminates communication.

**Table 4.3** Spectrum information table - Table containing the information about the frequency band occupied by the primary user, secondary user and the unoccupied frequency band

| FMIN | FMAX | FC | BANDWIDTH | POWER | MODULATION | SensorName | Occupancy | User type |
|---|---|---|---|---|---|---|---|---|
| 460000000 | 462000000 | 461000000 | 2000000 | 10 | FM | Sensor-1 | 1 | primary |
| 462000000 | 463000000 | 462500000 | 1000000 | - | - | - | 0 | - |
| 464000000 | 465000000 | 464500000 | 1000000 | 10 | dbpsk | Sensor-2 | 1 | secondary |

## 4.3 Secondary user architecture

The secondary user architecture used in this testbed is shown below in Figure 4.2. The secondary user receives information from the DSA broker using USRP hardware shown in Figure 4.3 at 900MHz (control channel frequency). This information is sent to the DSA broker listener. The DSA broker listener then reconfigures the physical layer to this frequency. The physical layer has two components, a hardware component and a software component.

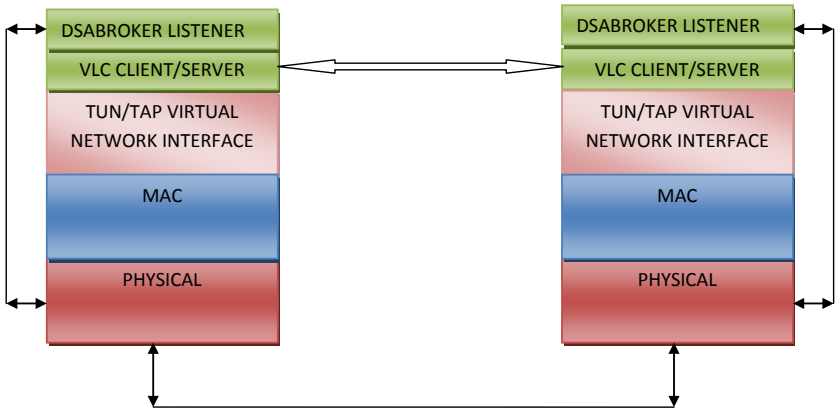


**Figure 4.2** Secondary user architecture.

GNURADIO is the software component and USRP is the hardware component.After configuring the physical layer, the DSA broker listener creates a virtual network device(TUN/TAP) and allocates an IP address. virtual network device [27]is a simple point-to-point device, which, instead of receiving packets from physical media, receives them from a user space program that imports this device. Similarly, instead of sending packets via physical media, the virtual network device sends them to a user space program that imported this device. TUN is a virtual point-to-point network device and provides 2 interfaces; /dev/tunX which

is a character device and tunX, a virtual point-to-point interface. The DSA broker listener opens a /dev/tunX interface and writes the application data to it. This data is then received by the kernel.

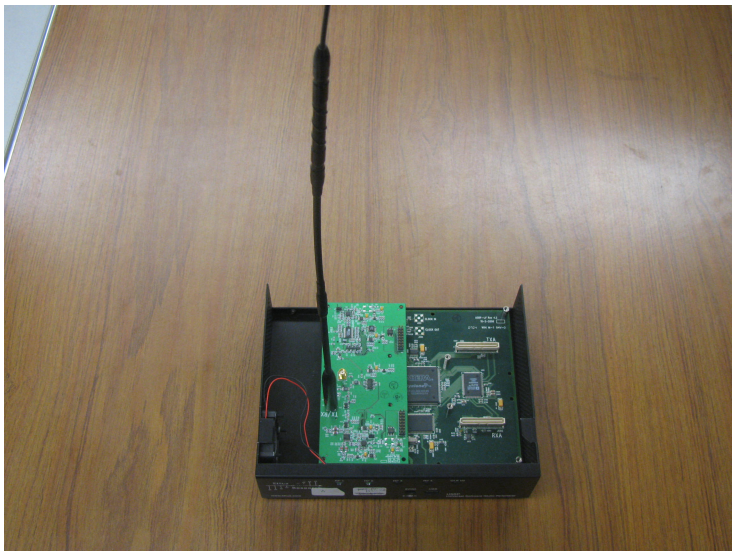

**Figure 4.3** USRP.

After assigning an IP address to the virtual network device, the DSA broker listener then configures the VLC client/server to either listen to an incoming audio stream or send an audio stream out. If the secondary user application is VLC client, then it streams out audio. The audio sent by the VLC client has the IP address of the secondary user (that is acting as VLC server), as the destination IP address. This audio data is written to the /dev/tunX interface, which is read by the kernel as explained above and sent to the physical layer. On the other hand, if the secondary user application is VLC server, then it should receive an audio stream from the client. In this case the audio stream from the client is received by the physical layer. This audio data is then written by the kernel to the tunX interface. The VLC server then reads this data from the /dev/tunX interface.

## 4.4 Testbed setup

The testbed is shown below in Figure 4.4. The Beagle Board based sensor and the laptop based sensor are the two sensors used in this testbed. The laptop based sensor used in this testbed , uses the same sensor architecture that was designed for the Beagle Board. The reason for using the laptop based sensor is due to limited availability of the Beagle Board. Both the sensors continuously scan the spectrum of interest(462MHz to 463MHz in this testbed) and keeps reporting the scanned results to the DSA broker using Wifi. The USB Wifi dongle connected to the Beagle Board provides Wifi capability to the Beagle Board. The DSA broker receives the results from both the sensors and populates the sensor summary table shown in Table 4.1. The DSA broker then processes these results in the sensor summary table and identifies the frequency bands occupied by the primary user and the secondary user. In general, primary users can be identified based on various parameters like modulation, bit rate, packet format etc. In this testbed, the primary user is identified by a two step process. The energy detection technique described in the previous chapters is used to detect the signal. From the detected signals, signals with an energy level above a certain threshold (configurable) are then classified as digital or analog signals. The analog signals above this threshold are classified as primary users. Signals whose energy level is below a threshold or digital signals are classified as secondary users. This information is then populated into the the spectrum information table shown in Table 4.3.

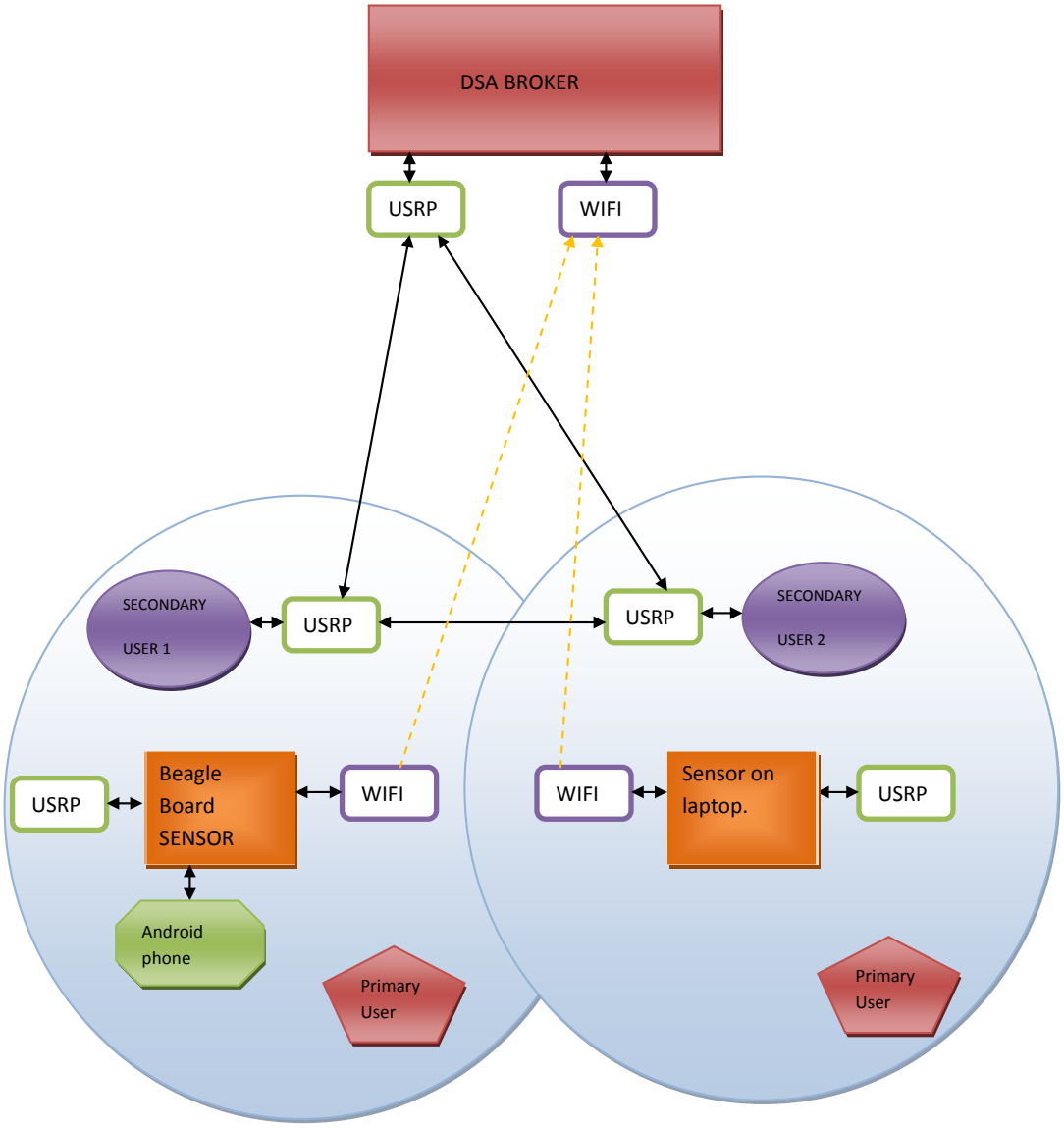


**Figure 4.4** Testbed.

The secondary users need to register with the DSA broker, if they need to use the spectrum. In this testbed, for illustration purposes, we use only 2 secondary users and they register with the DSA broker. Once the secondary users register, the DSA broker allocates the free spectrum to the secondary users. In this testbed, for illustration purposes, the unoccupied

Family Radio Service(FRS) frequencies in the frequency range 462MHz to 463MHz are allocated to the secondary user. A list of the FRS frequencies in the frequency range 462MHz to 463MHz is shown in Table 4.4.

**Table 4.4** FRS frequencies in the frequency range 462MHz to 463MHz

| Channel | Frequency (in MHz) |
|---|---|
| 1 | 462.5625 |
| 2 | 462.5875 |
| 3 | 462.6125 |
| 4 | 462.6375 |
| 5 | 462.6625 |
| 6 | 462.6875 |
| 7 | 462.7125 |

If the primary user returns to the FRS frequency allocated to the secondary user, then the DSA broker identifies another FRS frequency in this frequency range (462MHz to 463MHz) and allocates it to the secondary users. The secondary user then needs to reconfigure itself to the new frequency.

## 4.5 Experimental Results

Figure 4.5 shows the different time intervals that reflect the performance of the testbed and its response to primary user's arrival. T1 is the time taken by the DSA broker to identify the presence of primary user. T2 is the time taken to identify if there is an overlap between the primary user and the secondary user frequency bands. T3 is the time taken for the secondary user to move out of the overlapping primary user frequency band. T4 is the time taken for the secondary user to reconfigure itself to the frequency allocated by the DSA broker. All

of these time intervals are measured from the the start of primary user transmission to their respective end points.

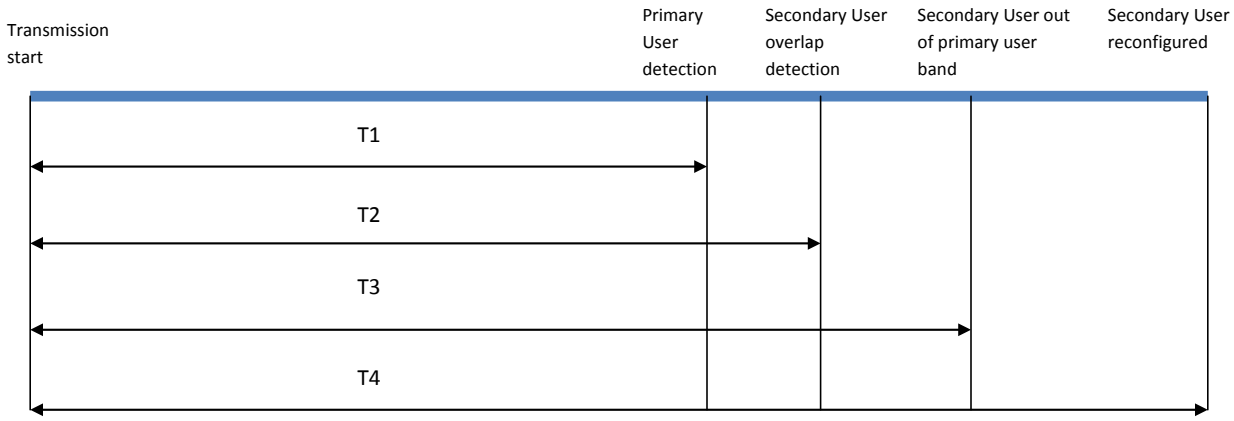


**Figure 4.5** Testbed timing diagram.

Since the primary user, DSA broker, sensor and the secondary user are different systems, measuring these timings is difficult as they need to be synchronized. But perfect synchronization may not be possible. Hence, to overcome the synchronization issue the DSA broker, the primary user and the secondary user were implemented on the same laptop. The python code benchmark_tx.py from the gnuradio examples was used to emulate the primary user, by transmitting an analog signal with energy level above a certain threshold. The start time of this transmission is noted. The sensors that are continuously scanning the radio environment keep sending the scan results to the DSA broker. The DSA broker will then process this information and identify the primary user's presence. The time at which the primary user was identified is noted. The difference between this time and the primary user transmission start time gives T1. After identifying the primary user's presence, the DSA broker will check to see if there is an overlap between the primary user and the secondary user frequency bands. In order to do this, the DSA broker queries the secondary user registration table and the

spectrum information table . From the secondary user registration table, the DSA broker gets the frequency range of the secondary users and from the spectrum information table the DSA broker gets the frequency range of the primary user. Then to check if there is an overlap, the frequency range of the secondary user is compared with the frequency range of the primary user. If the secondary user frequency range lies between the minimum and maximum frequency of the primary user, then there is an overlap. If overlap is identified, the time at which it was identified is noted. The difference between this time and the primary user transmission start time gives T2. Once the overlap is identified the DSA broker instructs the secondary user to a different frequency band using the control channel. After receiving this information from the DSA broker, the secondary user stops transmission on the previous frequency. The time at which the secondary user stops transmission is noted. The difference between this time and primary user transmission start time gives T3. The secondary user then reconfigures itself to the new frequency allocated by the DSA broker. Once the reconfiguration is complete the time is noted. The difference between this time and the primary user transmission start time gives T4.

The following table. 4.5 shows the values of the time intervals T1, T2, T3, T4 (in seconds) for 10 different trials.

**Table 4.5** Testbed response timings

| **Trial Number** | **DSA broker - primary user detection time(T1 secs)** | **overlapping secondary - primary detection(T2 secs)** | **secondary user out of primary user band(T3 secs)** | **secondary user reconfiguration time(T4 secs)** |
|---|---|---|---|---|
| **1** | 0.554 | 0.556 | 0.572 | 0.658 |
| **2** | 0.642 | 0.643 | 0.652 | 0.721 |
| **3** | 0.589 | 0.591 | 0.609 | 0.678 |
| **4** | 0.602 | 0.604 | 0.619 | 0.709 |
| **5** | 0.621 | 0.622 | 0.640 | 0.718 |
| **6** | 0.597 | 0.599 | 0.613 | 0.691 |
| **7** | 0.720 | 0.721 | 0.737 | 0.811 |
| **8** | 0.610 | 0.611 | 0.627 | 0.701 |
| **9** | 0.750 | 0.751 | 0.771 | 0.859 |
| **10** | 0.678 | 0.680 | 0.693 | 0.776 |
| **Average** | 0.636 | 0.637 | 0.653 | 0.732 |

The time taken by the sensors to scan the radio environment and report the results to the DSA broker is the main contributor to time T1. On an average the sensors take 500 - 600 ms to scan and report the results to the DSA broker. When the time taken to scan and report the results to DSA broker is subtracted from T1, the difference is the time taken by the DSA broker algorithm to detect the primary user. The difference between T2 and T1 is the time taken by the DSA broker algorithm to identify the overlap between the secondary user and primary user frequency bands. The difference between T3 and T2 is the time taken by the secondary user to stop its transmission on the previous frequency (the frequency that is overlapping with the primary user frequency), after being notified by the DSA broker. The difference between T4 and T3 is the time taken by the secondary user to reconfigure

the GNURADIO flow graph and begin the transmission using the new frequency allocated by the DSA broker.

These results are comparable to the results obtained from the testbed that was used in an IEEE Globecom paper [31], for which I was the second author. In this paper, the testbed did not have any Beagle Board based sensor and all the sensors were laptop based. Also, the signals with Differential Binary Phase Shift Keying(dbpsk) as modulation were classified as primary users. According to the results from this paper, the time taken for the secondary users to become aware of the primary user's presence and move to a different frequency band is around 993 ms. But in the testbed used in this thesis , it is around 732 ms. This difference of around 250 ms is because of the difference in the parameters used to define the primary user. As a specific modulation type was used to classify the primary user in the globecom testbed, it takes more time for classification and in turn identification of the primary user. But in the testbed used in this paper, all analog signals above a certain threshold are classified as a primary user and therefore takes lesser time compared to the globecom testbed.

# Chapter 5

# Conclusion

## 5.1 Summary

In Chapter 2, the challenges involved in porting the PSCR architecture to the Beagle Board are discussed. First, the advantages in porting the PSCR to the Beagle Board and the performance concerns are highlighted. Second, the communication architecture between the Beagle Board and the Android phone is proposed and explained. Third, the PSCR control application on the Android phone that controls the Beagle Board PSCR is discussed. Fourth, the different stages of audio processing between the Android phone and the Beagle Board is explained. Fifth, the communication protocol that is used for communication over the Android phone - Beagle Board interface is discussed. Last, the classification algorithm used to classify AM/FM/CPFSK modulations is explained.

In Chapter 3, The cooperative spectrum sensing technique to achieve DSA is discussed. First, the DSA broker architecture and the MySQL database table that is used to achieve DSA are explained. Second, the Beagle Board based Sensor Architecture is explained. Last,

the communication protocol and the packet format used for communication between the sensor and the DSA broker are discussed.

In Chapter 4, the testbed setup that achieves DSA using Beagle Board based sensor is examined. First, the testbed setup and the connection between the different nodes of the testbed is discussed. Second, the secondary user architecture is explained. Finally, the parameters used to analyse the performance of the testbed are discussed and the experimental results for these parameters are shown.

## 5.2 Contributions

Listed below are some of the contributions from this thesis, 1) Communication architecture between the Beagle Board and the Android phone. 2) The communication protocol for the Beagle Board - Android phone interface. 3) Development of PSCR control application on Android based Nexus One phone to operate the Beagle Board in scan, talk and gateway mode. 4) Finite state machine on the Android phone that reduces redundant communication with the Beagle Board. 5) Audio processing on the Android phone. 6) P25 radio signal detection capability that was added to the PSCR architecture. 7) The DSA broker and the Beagle Board based sensor architecture that was used in the testbed setup to achieve DSA.

## 5.3 Future work

1) Updating the PSCR framework to extend support for P25 Radios. 2) Adding new capabilities to the Beagle Board based sensor to detect wide range of primary users with different properties. 3) Expanding the scale of the current testbed setup with multiple DSA Brokers and sensors to cover a wider geographical area and analyze the performance.

# Bibliography


[1] [Online]:http://beagleboard.org/.

[2] G. Staple and K. Werbach, The end of spectrum scarcity [spectrum allocation and utilization], *IEEE Spectrum*, vol. 41, no. 3, pp. 4852, March. 2004.

[3] D. A. Roberson, C. S. Hood, J. L. LoCicero, and J. T. MacDonald, Spectral occupancy and interference studies in support of cognitive radio technology deployment, in *1st IEEE Workshop on Networking Technologies for Software Defined Radio Networks*, pp. 2635, Sept. 2006.

[4] R. Chiang, G. Rowe, and K. Sowerby, A quantitative analysis of spectral occupancy measurements for cognitive radio, in *IEEE 65th Vehicular Technology Conference*, pp. 30163020, April. 2007.

[5] Spectrum occupancy measurements, Tech. Rep. [Online]. Available:http://www.sharedspectrum.com/measurements/.

[6] J. Chapin and W. Lehr, Cognitive radios for dynamic spectrum access - the path to market success for dynamic spectrum access technology, in *In Proc. of the IEEE New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, 2007.

[7] S. Haykin, Cognitive radio: brain-empowered wireless communications, *IEEE Journal on Selected Areas in Communications* 23 (2) 201–220, 2005.

[8] J. Mitola, Cognitive radio: An integrated agent architecture for software defined radio, Ph.D. dissertation, Royal Institute of Technology (KTH), Sweden, 2000.

[9] Feng Ge, Qinqin Chen, Ying Wang, Charles W. Bostian, Thomas W. Rondeau, and Bin Le,"Cognitive Radio: From Spectrum Sharing to Adaptive Learning and Reconfiguration," in *Aerospace Conference, 2008 IEEE*, pp. 1–10, 2008.

[10] Bin Le,"Building a Cognitive Radio : From Architecture Definition to Prototype Implementation", Ph.D. Dissertation, in Dept. of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University: Blacksburg, VA, 2007.

[11] J Mitola,"The Software Radios",*IEEE National Telesystems Conference*, Digital Object Identifier 10.1109/NTC.1992.267870, 1992.

[12] Jeffrey H.Reed,"Software Radio: A Modern Approach to Radio Engineering", Prentice Hall PTR, New Jersey, 2002.

[13] M. Ettus, The universal software radio peripheral (USRP),. [Online].Available:http://www.ettus.com,EttusResearch,LLC, 2008.

[14] E. Blossom, Exploring GNU radio,. [Online]. Available:http://www.gnu.org/software/gnuradio/doc/exploring-gnuradio.html, November. 2004.

[15] Beagle Board,"System reference Manual,". [Online]. Available:http://beagleboard.org/static/BBSRM_latest.pdf

[16] [Online]:http://wiki.openembedded.net/index.php/Main_page.

[17] A.Fayez,Qinqin Chen,Jeannette Nounagnon and Charles W.Bostian,"Leveraging embedded heterogeneous processors for software defined radio applications".*SDR'10 Technical Conference*(yet to be published).

[18] Texas Instruments,"TMS320C6000 DSP/BIOS User's Guide,". [Online]. Available:http://focus.ti.com/lit/ug/spru303b/spru303b.pdf

[19] Texas Instruments,"C6000 Code Generation Tools,". [Online]. Available:http://focus.ti.com/docs/toolsw/folders/print/ccstudio.html

[20] Texas Instruments,"DSP/BIOS LINK Programmer's Guide,". [Online]. Available:http://pixhawk.ethz.ch/_media/omap/programmersguide.pdf

[21] [Online]:http://developer.android.com/sdk/index.html.

[22] Training Guide,"P25 Radio Systems" [Online]:http://www.danelec.com

[23] Qinqin Chen,"Cognitive Gateway to Promote Interoperability, Coverage and Throughput in Heterogeneous Communication Systems", Ph.D. Dissertation, in Dept. of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University: Blacksburg, VA, 2009.

[24] Danijela Cabric,Artem Tkachenko,Robert W.Brodersen, "Experimental Study of Spectrum Sensing based on Energy Detection and Network Cooperation", 2006.

[25] D.Cabric. S.M. Mishra. R.W. Brodersen, Implementation Issues in Spectrum Sensing, *In Asilomar Conference on Signal, Systems and Computers*, November. 2004.

[26] A. V. Oppenheim, R. W. Schafer and J. R. Buck, Discrete-Tme Signal Processing, Prentice Hall, 1999.

[27] Virtual Point-to-Point (TUN) and Ethernet (TAP) Devices [Online].Available from:URL:http://vtun.sourceforge.net/tun/.

[28] I. F. Akyildiz, W. Y. Lee, M. C. Vuran, and S. Mohanty, Next generation/dynamic spectrum access/cognitive radio wireless networks: a survey, *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 50, pp. 2127–2159, 2006.

[29] Feng Ge,"Software Radio-Based Decentralized Dynamic Spectrum Access Networks: A Prototype Design and Enabling Technologies ", Ph.D. Dissertation, in Dept. of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University: Blacksburg, VA, 2009.

[30] S. M. Mishra, A. Sahai, and R. W. Brodersen, Cooperative sensing among cognitive radios, in *IEEE International Conference on Communications*, pp. 1658–1663, 2006.

[31] Feng Ge, Aravind Radhakrishnan, Mustafa Y. ElNainay, Qinqin Chen, Charles W. Bostian, and Allen B. MacKenzie, "Software Radio-Based Decentralized Dynamic Spectrum Access Networks: A Prototype Design and Experimental Results", *IEEE Globecom Symposium on Selected Areas in Communications*, 2010.