Open Source Mobile Network

U. Sajeev, R. Muralitharan, M. M. M. Ramsan, M. N. M. Zamrath, Dileeka Dias

Department of Electronics and Telecommunications Engineering,

University of Moratuwa,

Moratuwa, Sri Lanka

ABSTRACT

The paper presents the development of a Global System for Mobile communication (GSM) network built on Open Source Software (OSS) and tested as a stand-alone mobile network. Establishing a mobile Base Transceiver Station (BTS) along with a switch was the main objective of this project.

Our project includes configuration of hardware components, selection and integration of the proper open source software and, testing of compatible versions of software, hardware-software integration and interconnecting two such systems via the Internet. The key contribution and novelty of the project is the development of a stand-alone mobile network BTS using the Universal Software Radio Peripheral (USRP) and a Banana-Pi device.

1.0 INTRODUCTION

Throughout the history of Linux, we see that open source projects have gathered the attention of the tech-savvy community. Open source projects have created a major impact and opened up new opportunities in industries such as computing and telecommunications. The tendency of creating one's own mobile network using Software Defined Radios (SDRs) has been increasing among the research community as well as in the commercial sector. From Figure 1 we can clearly see that OSS adaptation in the service sector that includes telecommunications is high. Companies such as Range Networks [1], who develop such Software Defined Radios (SDRs), are not trying to compete with Nokia, Alcatel-Lucent and Ericsson, but looking deeper into low-cost alternatives to huge mobile networks built for mobile carriers. Also these companies are looking into applications of SDR in academia, public safety and helping underserved geographies such as Africa.

The main objective of this project is to develop a stand-alone mobile network using OSS, the access to which can be controlled as required. Further the mobile network should be connected to commercial networks via the Internet. There are two main components in this architecture, the radio front-end hardware and a suite of open source signal processing, communication and control software. The USRP, a SDR device is used as the radio front-end that provides the air interface for the mobile stations. This hardware is controlled by a set of OSS which is implemented and integrated in a single board computer, namely a Banana-Pi board. A software telephone Private Branch Exchange (PBX) implementation allows the registered phones to communicate with each other and also allows to get connected to the Public Switched Telephone Network (PSTN) or Voice over IP (VoIP) service providers. Therefore, two different stand-alone systems could be interconnected through the Internet by establishing a Session Initiation Protocol (SIP) trunk. Figure 2 shows the hardware /software architecture of the base station.

The main communicator between the hardware and the software is the USRP Hardware Driver (UHD Drive). GNU Radio provides the runtime signal processing and communicating with external hardware. OpenBTS software is a Linux Operating System (OS) based application. Also it allows SDRs to present a GSM air interface for GSM-

Compatible mobile phones to be used as SIP clients in VoIP networks. OpenBTS has all the configuration parameters of a BTS. OsmoTRX is an SDR transceiver that implements the physical layer of a BTS. OsmoTRX is based on transceiver code from the OpenBTS project. Asterisk is a software telephone PBX implementation which allows the registered phones to communicate with each other and also allows to get connected to PSTN or VoIP service providers. OpenBTS uses Asterisk to register each connected mobile user overs it's Subscriber Identity Module (SIM) as a SIP Client.

After setting up two stand-alone networks, two public Internet Protocol (IP) addresses are assigned to the systems. These systems are then connected to the Internet through their Ethernet ports. By setting appropriate SIP configurations in Asterisk we establish an SIP trunk between two systems. Now calls can be made from a mobile station attached from one network to mobile station attached to another network. It is necessary to have the same database of users in both systems [14].

Section 2.0 presents a review of a similar project on creating a base station. In Section 3.0, we discuss the hardware / software selections, and the implementation of the test system. In Section 4.0, discuss how our system could be adapted for real life applications and the paper concludes with Section 5.0.

2.0 LITERATURE REVIEW

Though mobile telephony technologies have advanced in leaps and bounds, there are many areas in the world which do not have access to these. An example of an SDR-based solution for this is, Village Cell [5]. This is a low cost implementation of a stand-alone tower to accommodate GSM calls intra-cell as well as inter-cell. The SDR with radio front-end used in this implementation is USRP2 while others are based on open source software. The concern lies on the optimized usage of hardware and the immobility. OpenBTS is implemented on a Personal Computer (PC) machine while call server, Asterisk has been implemented on another PC for each router used in the network.

3.0 SYSTEM IMPLEMENTATION

3.1 Hardware Platform

3.1.1 Software Defined Radio (SDR)

An SDR is a hardware communications platform that uses software implementation of digital communications algorithms [9]. SDR provides a flexible platform to study, and test communication systems practically. Normally in wireless and mobile communication systems we learn the theory behind the communications and get the basic understanding of different building blocks of a communications system. We mostly use simulation software such as MATLAB in order to design and analyze the characteristics of systems, where simulations are based on simplifying impractical assumptions. Therefore, SDR can be used in order to reinforce the theoretical aspects with hands-on learning.

The USRP is an SDR reconfigurable Radio Frequency (RF) hardware designed to build and test digital communication systems. In our project we have used the latest product of Ettus Research known as USRP N210.

3.1.2 SBX USRP Daughterboard (400 MHz – 4.4 GHz)[10]

The SBX is a wide bandwidth transceiver. It has local oscillators to operate transmit and receive chains separately allowing for dual-band operation. This board is necessary for the sampling of the signals.

3.1.3 Antennas

VERT900 antenna, a product of Ettus research has been used in our project along with the USRP for transmission and reception. These are usually rubber duck-style antennas similar in appearance to a normal Wi-Fi antenna with a Sub Miniature version A (SMA) connector [11].

3.1.4 Embedded Hardware Platform: Banana-Pi

Even though the initial implementation was done using a PC, our objective was to have a compact version of the system. So our choice was the Banana-Pi (B-Pi) board. [8].

3.2 Open Source Software (OSS)

3.2.1 USRP Hardware Drive (UHD Drive)

The UHD Drive software is the hardware driver for all USRP devices to establish communication between the system and the USRP. It works on platforms such as Linux, Windows, and Mac OS and also can be built with GNU Compiler Collection (GCC), Clang, and Microsoft Visual C++ (MSVC) compilers. The main role of the UHD is to supply the host driver and provide an Application Program Interface (API) to SDR devices. The UHD allows using its functionality alone or with third party software such as GNURadio, OpenBTS, LabVIEW [9]. There are some other dependencies also have to be installed to fulfill proper communication such as CMake to generate project built files, boost libraries, Python compilers, LibUSB to support USB hardware and Cheetah to generate source code.

3.2.2 GNU Radio

GNURadio is a free and open-source software development toolkit that provides the runtime signal processing and processing blocks to implement software radios. The software architecture of GNURadio is developed as a combination of Python and C++, where Python is the top layer of the application such as functioning rules, definitions, the graph and the setting options, with C++ handling performance and critical signal processing blocks development using processor floating point extensions [9]. Also it has graphical user interface and GNURadio companions similar to simulink and it allows dragging and dropping radio modules to design our own system.

3.2.3 OpenBTS

The OpenBTS project was about developing an open-source application for UNIX where presenting a GSM air interface for the USRP that is also uses the Asterisk VoIP PBX soft switch for calls routing. This provides a platform for integration of ubiquitous GSM air interface and VoIP back end with relatively lower cost than those already available technologies [12]. In this project the OpenBTS is described as a collection of parallel logical channels which are defined under GSM 04.03 and each channel roughly follows the layers of Open Systems Interconnection (OSI) Model.

3.2.4 Private Branch Exchange (PBX)

A PBX is a call exchange or switching system that serves a private organization and performs concentration of central office lines or trunks and provides intercommunication between a large numbers of telephones within an organization. The central office provides connections to the any other cellular or telephone network through SIP trunks or the public switched telephone network and the concentration aspect of a PBX facilitates the shared use of these links between all stations in the organization and to external networks. We have used Asterisk for the PBX requirement of our network. Documentation on exchange and switching functions carried by Asterisk are easily accessible compared to other PBX platforms. Further, Asterisk supports the Linux OS [13]. There is a large set of different PBX's are available today such as Asterisk, Yate, Free Switch, Free-PBX and so on. Each of these PBX has their own advantages and capabilities in terms of speed, capacity, protocol used and type of traffic carried.

3.2.5 OsmoTRX

OsmoTRX is a software-defined radio transceiver that implements the Layer 1 physical layer of a BTS comprising some of the 3rd Generation Partnership Project (3GPP) specifications.

It is also a separate transceiver unit which is important in OpenBTS implementation on embedded systems.

3.3 System Components and Database Interconnection

The system components need to be installed and interconnected properly so that they can easily

communicate with each other. Each internal connection should be created using allocated port and IP's. As this entire project is implemented on a common platform, we only use local IP (127.0.01) and set of ports such 5060, 5062, 5063 and 5064 to create the connection between System components. If we installed each of them in different platforms which may be directly connected by any physical mediums or can be remotely connected through internet, then we need to define the Ethernet interfaces and gateways properly with their relevant IP addresses. Fig. 4 shows the entire system which consists of four sub systems and four databases and how each component will communicate.

3.4 Call and Short Message Service (SMS) Handling

3.4.1 Call handling process

When a call arrives to this system, it first reaches the OpenBTS and the OpenBTS checks the service-ownership of the arrived call by communicating with SIP Authentication server (Sipauthserve). The Sipauthserve makes a request to sqlite3 database (subscriber registry) to determine whether both the initiator and destination host of the call are registered in the database. The Sipauthserve allows creating the local SIP connection or external SIP trunk to pass the call only if both subscribers are registered in subscriber registry and if not it simply gives an error as "unable to create SIP (cause 20 known)."

If it is an authorized call then OpenBTS routes it to Asterisk through local host IP 127.0.0.1 and port 5062 (where 5060 is the SIP proxy port between them and for voice transmission they use port 5062). The OpenBTS and Asterisk communicate between each other through already established two way SIP connection which was created while installing them in the platform. Asterisk will contact with sqlite3 database through Unix Open Database Connectivity (ODBC) and check the destination International Mobile Subscriber Identity (IMSI) of dialed number under extension specification and retrieve the details of context and destination IP assigned to that subscribers. According to these details the call would be routed to its destination. Asterisk provides a facility to keep the entry of the call arrivals and its duration in log. So one can check this log to get the usage and use it for call billing.

3.4.2 SMS handling process

SMS queue also uses the similar way to forward messages to its destination as the format of SIP messages. When a SMS arrives to OpenBTS, it directly forwards the message to smqueue via the local host IP 127.0.0.1 and port 5063 to do the authentication and queuing. The authentication part of smqueue is done through the SIP Authentication server as mentioned in OpenBTS but it does not contact with Asterisk to get the destination IMSI given by the sender for queuing purpose. Instead of that it directly communicates with subscriber registry to retrieve the destination IMSI and send it via OpenBTS. The configuration parameters required to set up smqueue are stored in the database named smqueue.db and the parameters can be changed by accessing this database according our requirements.

3.5 Implementation in the Lab

Fig. 5 shows how we initially set up the GSM network using the lab resources as a test GSM station. We installed UHD drive, along with the necessary dependencies, GNURadio, OpenBTS and Asterisk in a PC which is installed with Ubuntu 12.04 LTS OS, in the order mentioned before. The USRP device is connected to the system through a gigabit Ethernet cable and the USRP device serves the mobile phones through two TRX VERT 900 Antennas. We used a few SIMs from foreign countries in the test phones to establish a stand-alone GSM network. The SIMs had different Mobile Country Code (MCC) and Mobile Network Code (MNC) values which are not used in Sri Lanka. So we will not interfere with local mobile service providers. A stand-alone system was set up in a laboratory environment using five mobile phones that was capable of handling text messages and calls within the network.

3.6 Implementation on Banana-Pi

We selected Banana-Pi (B-Pi) board for the implementation of the software tools so we can have a compact version of the system. Reasons why B-pi is specifically chosen over Raspberry-Pi for this project are to communicate with the Gigabit (Gb) Ethernet interface of USRP N210, its higher RAM capacity as installation of GNURadio and UHDrive needs more RAM space, B-Pi has ARM Processors with NEON which makes its operations faster than Raspberry-Pi [17] and OsmoTRX is compatible with the ARM Processors with NEON

Following are the steps related to the installation of software tools in the embedded system.

1) Lubuntu Installation
2) Setup the Environment
3) Dependency libraries installation.
4) GNURadio Installation.
5) OpenBTS Installation [20].
6) OsmoTRX Installation [21].
7) Asterisk Installation [22].

4.0 POTENTIAL APPLICATIONS

In this section we are giving some of the application scenarios where we can use our OSS based base station.

4.1 Managed Access System

This application is about allowing access to the network only for authorized users while blocking others. For example in a prison we may give access to the system only for authorized administrative officers. We need to have better power control of the transmission as people near the periphery of the prison premises can get attached to the network and may be blocked.

4.2 Low cost cellular PABX

In a small organization, for internal communication they may use our system instead of having a separate wired intercom where you need phones for each person and need physical connection with the PBX in a fixed architecture. The officers can make toll-free calls using their mobile phones within the office premises.

4.3 Emergency Communications

Consider a situation after a natural disaster such as a Tsunami or an earthquake, where much of the public communication systems are destroyed or congested. In such situation, to establish a temporary communication network in a local area, a system such as ours would be helpful.

4.4 Aid disaster relief operations

As the system is capable of detecting and tracking IMSI numbers of subscribers, we can use the system to detect whether people are trapped inside the collapsed building and identify those people using IMSI numbers. In addition by making calls using those numbers may be possible to locate the victims.

4.5 IoT based Applications

As an example, a scenario inside a small factory or warehouse, where RF signal strengths are low or not available, an internal communication network can be established. In addition if there are GSM-enabled sensor nodes, they can send alert signals or data through this system, and our network can communicate with public networks via a SIP trunk.

4.6 Commercial Applications

In addition to allowing multimode, multimedia terminals, since SDR and open source software based terminals can be configured through software they could facilitate roaming, reduced terminal costs, comfort in dynamic spectrum management, and allow for service personalization features and more. The issues created by proliferation of the new air interfaces could be solved by adapting SDR based systems [23].

5.0 CONCLUSION

The main objective of this project is to demonstrate the implementation of an Open Source Mobile Network using SDRs and open source software tools as well as to interconnect such systems through Internet. One of the greatest challenges that were faced was the lack of proper documentation of available software and hardware

tools to achieve this objective. However, a successful implementation was done, giving the authors valuable exposure to open source system development.

6.0 REFERENCES

[1] Range Networks Website [Online]. Available: http://www.rangenetworks.com/

[2] OpenBTS Website. [Online].

Available: http://openbts.org/

[3] Steven Max Patterson. 2014. Open source project builds mobile networks without big carriers, [Online]. Available: http://www.networkworld.com/article/2226543/wireless/open-source-project-builds-mobile-networks-without-big-carriers.html

[4] Banana Pi Website. [Online]. Available: http://www.bananapi.org/

[5] Abhinav Anand, Veljko Pejovic, David L. Johnson and Elizabeth M. Belding, "VillageCell: Cost Effective Cellular Connectivity in Rural Areas," ICTD2012 2012 Atlanta, GA, USA, ACM 978-1-4503-1045, 1/12/03.

[6] Taha Hajar, Wajeb Saab, and Tarek Sakakini, "NI USRP GSM Base Station," The Thirteenth FEA Student and Alumni Conference, American University of Beirut, Beirut, Lebanon, 2014.

[7] Karthik Vasudeva, Bekir Sait Ciftler, Armando Altamar and Ismail Guvenc, "An Experimental Study on RSS-Based Wireless Localization with Software Defined Radio," U.S. National Science Foundation, 978-1-4799-4608-2, 2014.

[8] Banana Pi Resources. [Online]. Available: http://www.lemaker.org/resources/939/banana_pi_quick_startguide.html

[9] Matthias Fähnle, "Software-Defined Radio with GNU Radio and USRP/2 Hardware Frontend: Setup and FM/GSM Applications," Bachelor thesis, University of Applied Sciences, Institute of Communication Technology , Germany, 2010.

[10] Ettus Research Website. [Online]. http://www.ettus.com/product/details/SBX120

[11] Ettus Research Website. [Online]. http://www.ettus.com/product/details/VERT900

[12] David A. Burgess, Harvind S. Samra, "The Open BTS Project", Kestrel Signal Processing, Inc ,Fairfield, California , 2008

[13] Range Networks Website[Online]. http://wush.net/trac/rangepublic/wiki/asteriskConf

[14] AsteriskTM Website [Online] . http://www.asteriskdocs.org/en/2nd_Edition/asterisk-book-html-chunk/connecting_two_asterisk.htm

[15] OpenBTS. "Installing and Configuring Asterisk." Internet: http://wush.net/trac/rangepublic/wiki/asteriskConfig [Apr. 30, 2015].

[16] Asterisk. "Connecting Two Asterisk Boxes Together via SIP." Internet: http://www.asteriskdocs.org/en/2nd_Edition

[17] T Tsou. "OsmoTRX." Internet: http://openbsc.osmocom.org/trac/wiki/OsmocomOverview?version=10

Feb. 10, 2015 [Apr. 30, 2015]

[18] T Zhang. "Quick Start Guide." Internet: http://www.lemaker.org/resources/939/banana_pi_quick_start_guide.html, Apr. 11, 2014 [Apr. 30, 2015].

[19] O W Purbo. "OpenBTS 2.8." Internet: http://opensource.telkomspeedy.com/wiki/index.php/OpenBTS:_N210_GNURadio_3.7.0#OpenBTS_2.8 Jul. 24, 2014 [Apr. 30, 2015].

[20] T Tsou. "BuildInstallRun." Internet: http://openbts.org/w/index.php/BuildInstallRun Nov. 20, 2014 [Apr. 30, 2015].

[21] University of Helsinki - Department of Computer Science. "Package gnuradio." Internet: https://packages.debian.org/search?arch=armhf&keywords=gnuradio, 2015 [Apr. 30, 2015].

[22] Hewlett-Packard. "Download Page for uhd-host_3.4.2-1_armhf.deb on ARM Hard Float machines." Internet: https://packages.debian.org/wheezy/armhf/uhd-host/download, 2015 [Apr. 30, 2015].

[23] C Caicedo. "Software Defined Radio and Software Radio Technology: Concepts and Applications." Internet: https://www.academia.edu/1319161/Software_Defined_Radio_and_Software_Radio_Technology_Concepts_and_Applications Mar. 2007 [Apr. 30, 2015].

[24] Sysmocom. "900 MHz E-GSM duplexer 30W." Internet: http://shop.sysmocom.de/products/dx900-kt30 [Apr. 30, 2015].

[25] A Steil. "OpenBTS." Internet: http://www.fh-kl.de/~andreas.steil/Projekte/OpenBTS/ [Apr. 30, 2015].

[26] D Spinellis and V Giannikas. "Organizational Adoption of Open Source Software." Internet: http://www.dmst.aueb.gr/dds/pubs/jrnl/2012-JSS-OSS-Industry-Use/html/SG11.html, Mar. 2012 [Apr. 30, 2015].

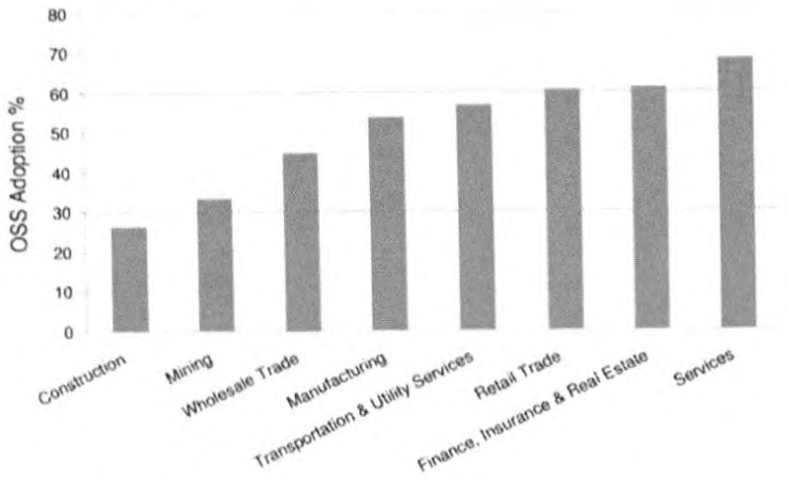

Fig 1 : Evidence of Open Source Software (OSS) Adoption across Industries. [26]

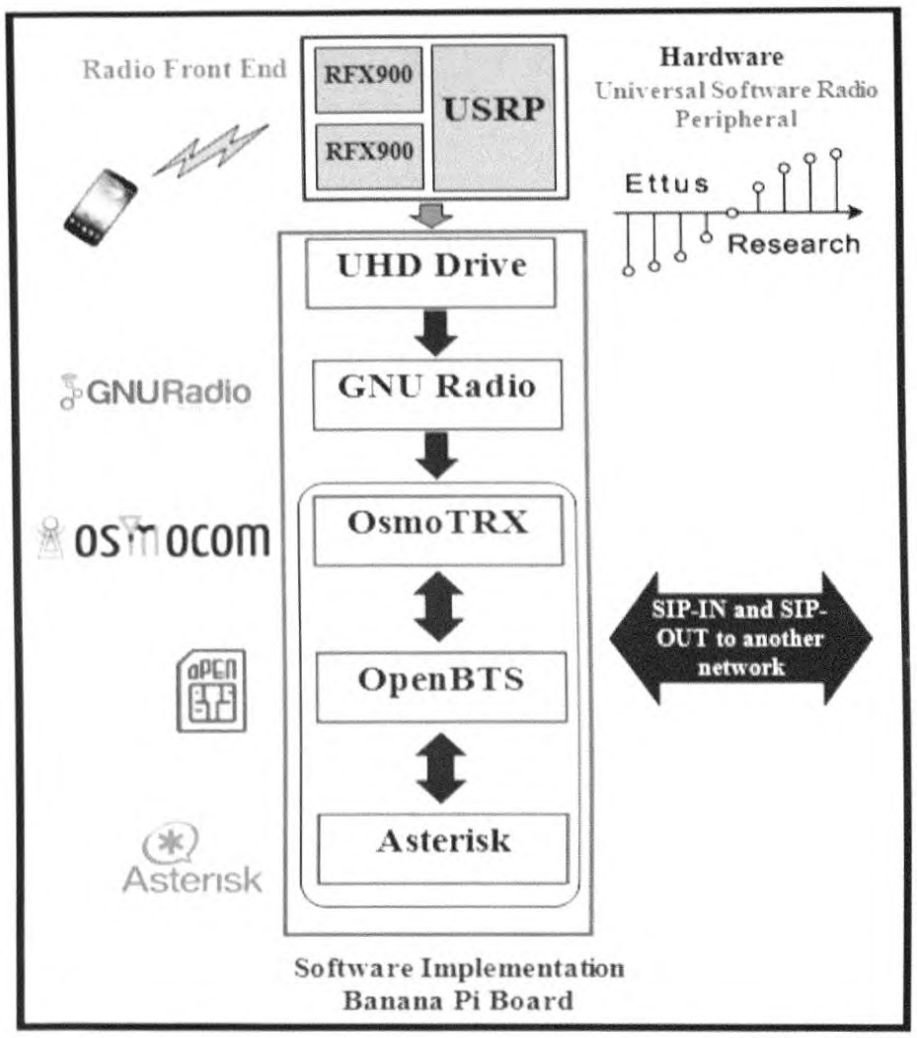

Fig. 2 : Overall Network Architecture describing components of Hardware Platform and Software

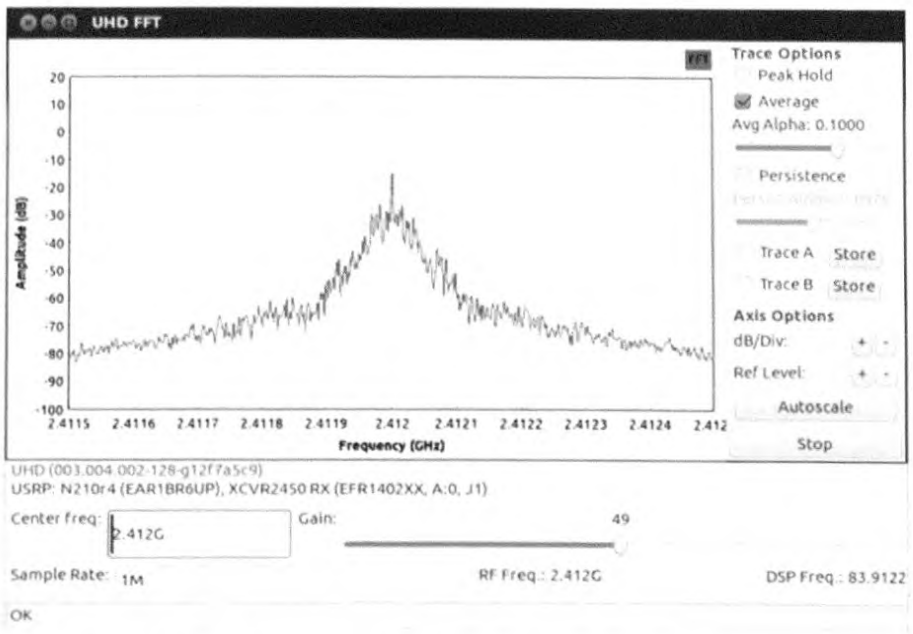

Fig. 3 : GNU Radio GUI Interface

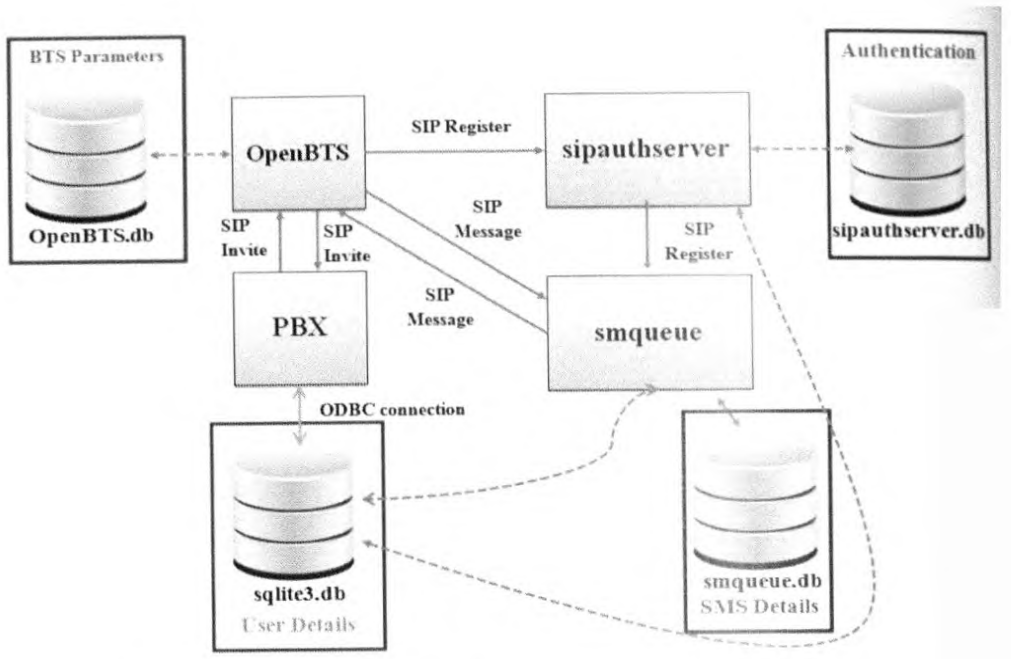

Fig. 1 : Database Interconnections

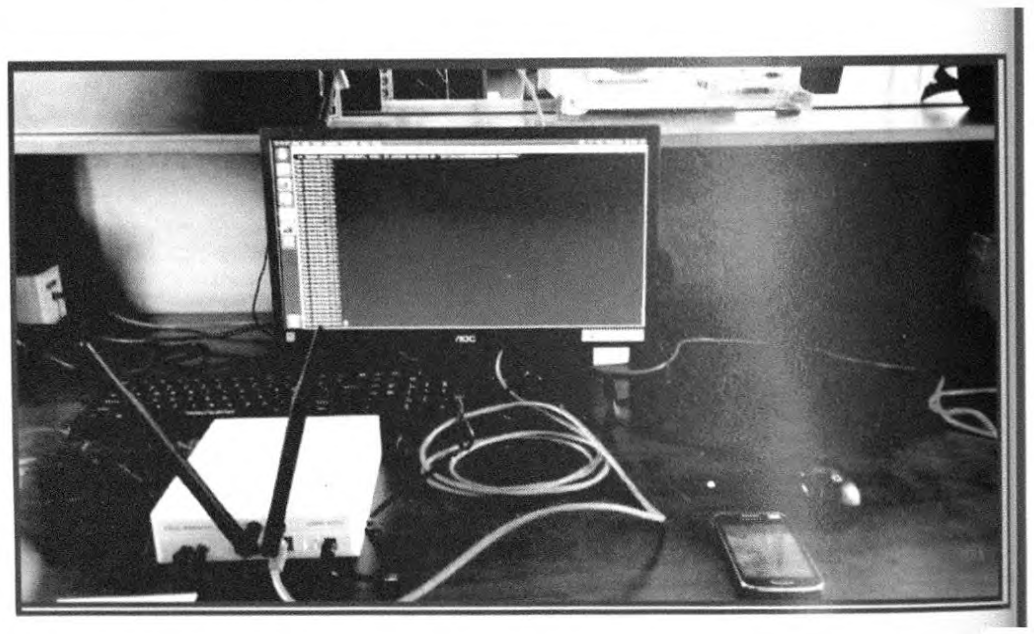

Fig . 5 : Lab Implementation

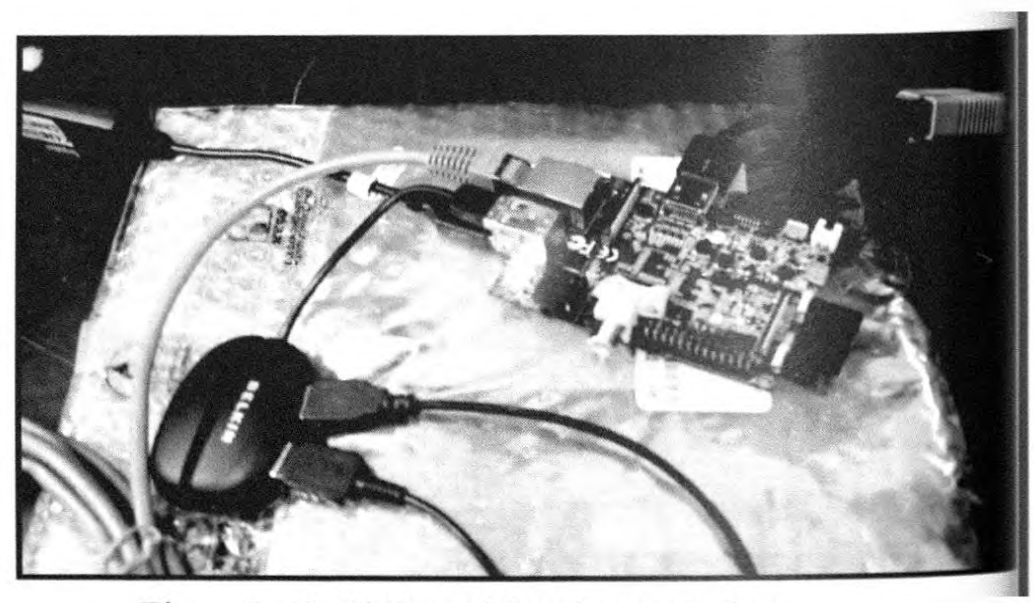

Fig . 6 : B-Pi Board Implementation