The practical experience of implementing a GSM BTS through Open Software/Hardware

(Invited Paper)

E. Natalizio, V. Loscrí, G. Aloi
DEIS
University of Calabria
Italy
Email: {enatalizio,vloscri,aloi}@deis.unical.it

N. Paolí, N. Barbaro
Orangee srl
Via Pedro Alvares Cabrai
87036 Rende - Italy
Email: {n.paoli,n.barbaro}@orangee.com

***Abstract*—The objective of this paper is to show the implementation experience of a GSM Base Transceiver System (BTS) by using Universal Software Radio Peripheral, which is a multi-purpose motherboard for Software Defined Radio (SDR), and a Personal Computer (PC). The OpenBTS project provides a software suite, created for the GNU Radio environment, and able to mimic the behavior of a GSM BTS. In order to allow GSM users to make calls and send text messages, it is necessary to operate some hardware modifications on the SDR motherboard, to set up a Session Initiation Protocol (SIP) server, to configure the server for providing connectivity between the mobile devices and the VoIP backhaul. In this paper, we will illustrate the needed software and hardware, and the steps necessary to create a small, cheap and autonomous GSM BTS.**

I. Introduction

[1] In recent years, the research and industry worlds have focused a lot of attention on Cognitive Networks (CN). Cognitive Networks are a very interesting concept for both the theoretical and practical aspects. Mainly due to the “cognitive” capabilities, from the theoretical point of view, CN represent the connecting link between telecommunications systems and artificial intelligence. At the same time, a telecommunication device able to sense the radio environment, discover under utilized frequency range and allocate its transmission in the most convenient frequency spot represents a great commercial innovation for all telecommunications providers. Historically, in the telecommunications field, CN are the evolution and the extension of Software Defined Radio (SDR). The main objective of SDR is to convert all of the analog blocks of a telecommunications system into equivalent digital blocks, in order to push the analog components of the architecture towards its extremities (the antenna) and make the whole system programmable. The advantages of such a system are well-known: low-cost implementation, easy to use, customization, interoperability, enhanced security, etc. Especially the reduced costs related to the hardware implementation of SDR systems represent an economically viable solution for telecommunications providers. The possibility to purchase multi-purpose hardware able to be programmed in order to change its behavior also created a lot of interest from research laboratories all around the world. In fact, hardware devices such as the Ettus Research USRP [1] used in combination with GNU Radio software [2] allow researchers to design, develop and test their ideas. In [3] it is possible to find several projects developed with the cited hardware/software, from Dynamic Spectrum Schemes to the whole IEEE 802.15.4 standard. Among these projects, the OpenBTS [4] one deserves a special mention, because it implements the whole GSM stack through GNU Radio (more details about OpenBTS will follow in the rest of the paper). The choice to implement a second generation BTS is motivated by several reasons: GSM is currently the most used cellular system in the world, its technology is consolidated and stable, the specifications of the standard are publicly available, its behavior is easier to emulate in respect of CDMA or WCDMA systems. In respect of the GSM architecture, reported in Fig. 1, the practical experience illustrated in this work focuses solely on the development of GSM BTS, however through the usage of some additional software it will show how it is possible to connect GSM mobiles and VoIP terminals.

The paper is organized as follows. Section II presents some literature works in the field of Cognitive Networks and Cognitive Radio Networks. Section III and IV introduces the

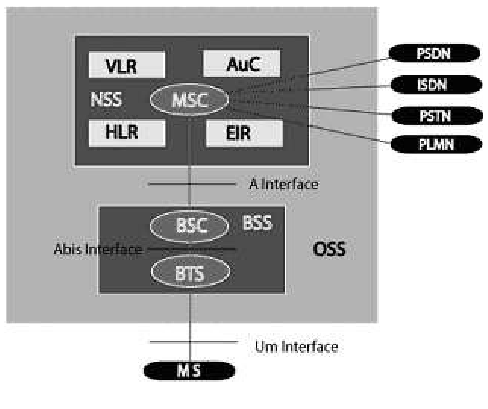

Fig. 1. Architecture of the GSM.

[1]This work has been carried out under the framework of OPEN-KNOWTECH, National Italian Project #DM21301, financed by the Italian Ministry of University and Research

hardware and software used for the GSM BTS implementation, respectively. Section V highlights some of the issues to be solved and some initial results of this experience, and finally Section VI draws the conclusion of the work.

II. Related Works

The concept of Cognitive Networks has existed in the research world for a number of years already. The first formal research on CN is in [5]. More recent research tends to separate the CN concept from the Cognitive Radio Networks (CRN). The pioneeristic work of Mitola [6] can be classified in the latter branch. In fact in [7], the author explains how CRs are able to interact amongst themselves in the general context of CN. The same approach is also used by Haykin [8] and Neel [9]: the former analyzes multi-user CR networks, the latter models the network as a multi-player game to determine the convergence criteria. The research on the CRNs is focused mainly on the PHY and MAC layer issues, usually by considering some end-to-end objective. In a CRN, still the radios take individual decisions, even though they interact in a cooperative fashion. Currently the CRNs applications include cooperative spectrum sensing [10], [11] and emergency networks [12]. The first citation of CN is from Clark [13], he proposes a network able to self-assemble and re-assemble through high level instructions, capable of discovering and fixing failures and malfunctions. In order to do this, it is necessary to have a Knowledge Plane (KP) that is transversal to the protocol stack, and that allows the devices to take cogntive decisions. The KP increases the intelligent capabilities of the terminals. Saracco in [14] noticed how in recent years the focus of research and industry moved from the resources control to user satisfaction, this would eventually lead to move intelligence from the network core to the terminals. While CRNs are focused on PHY and MAC layer, CNs work by taking into account the whole protocol stack. Furthermore, CNs devices are less autonomous than CRNs devices, in fact they have to cooperate in order to pursue a common objective, and the cognitive processes can be parallelized onto several network devices. Mahonen [15] states that CRs can be considered as a logical subset of CNs. Recently, several research groups have proposed architectures for CNs. These architectures can be classified according to their objective: the first category aims to solve complex problems by using the cognitive capabilities of the devices. E2R II [16] is a project that falls into this typology, it offers connectivity for all the devices through the usage of a connection-less reconfiguration. The main purpose of this ambitious project is to offer and preserve connectivity for all users. The same purpose is pursued by the m@Angel platform [17], which proposes a CN architecture for managing mobility in heterogeneous networks. Both these projects focus on interoperability and maintenance of 4G wireless and cellular networks. Instead, a second typology of CNs is presented by the Centre for Telecommunications Value-Chain Research (CTVR) of Trinity College [18]. In this case, the CN platform consists of reconfigurable wireless nodes. Nodes in [18] are able to solve several problems by modifying their protocol stack depending on the network operations conditions. The objectives pursuable from this typology of CNs extend beyond connectivity and mobility management. This work falls in the second category, since one of the main objectives of the OPENKNOWTECH project is to create devices able to use their cognitive capabilities in order to solve complex problems and guarantee network reactivity under different environmental conditions. Specifically, this work, by using the experience of the OpenBTS project [30], illustrates the steps to follow in order to transform commercial hardware in a GSM base station.

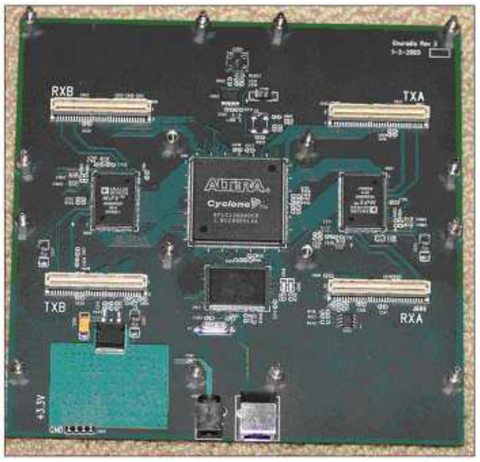

Fig. 2. Ettus Research Motherboard

III. Hardware

As we introduced in I, the choice of implementing a GSM BTS is motivated by two main reasons: the ease with which an FDMA/TDMA system can be implemented, and the inexpensive cost of such implementation. In fact, in order to have a fully operating GSM BTS, only a TX/RX daughterboard (working in the GSM bands) installed in a general purpose motherboard, connected to a personal computer is needed. Specifically, we used the following hardware:

- 1 Motherboard Ettus Research Universal Software Radio Peripheral (Fig. 2);
- 2 Daugtherboards RFX900, each of them equipped with a VERT900 antenna;
- several GSM terminals equipped with SIM cards.

The device used for the GSM BTS implementation is a general purpose motherboard able to host several daughterboards. It can be connected to a personal computer through a USB 2.0 interface and is able to use a Radio Frequency bandwidth of 16 MHs in both directions. The motherboard is equipped with 1 Field programmable gate array, 4 AD (sampling rate: 64 Msps and resolution: 12 bit) and 4 DA (sampling rate: 128 Msps and resolution: 14 bit) converters, 4 downconverters and 2 upconverters (both with programmable conversion rates). As we can see in Fig. 2, the motherboard can accomodate 4 daugtherboards: 2 slots are used for the RX daughterboards and 2 for the TX daughterboards. Both the RX daughterboards

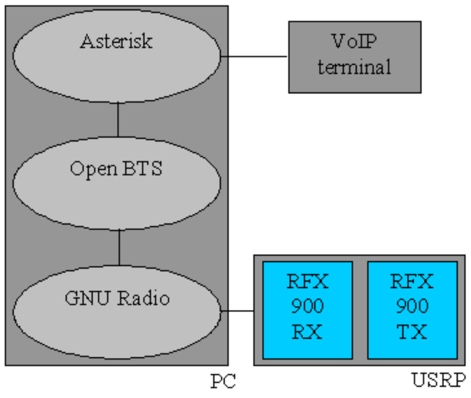

Fig. 3. Software/Hardware Architecture.

and the TX daughterboards need the RF stage, a tuner and a transmitter, respectively. The 4-slot separation permits two distinct and independent sections in RF for RX and TX, when we use non-IQ sampling. On the contrary, when the sampling is IQ, each section is used for only receiving or transmitting the two branches of the complex signal. The daughterboards used for this work are RFX900, which receive/transmit on the 750-1050 MHz band, with a transmit power of 200 mW (23dBm) through an ISM VERT900 antenna, 9 inches long and with a 3dBi gain.

IV. Software

In Fig. 3, we show the dependencies among the software architecture components. The OpenBTS [4] component is the core of the work, it runs over GNU Radio [2]. Specifically, the software installed on the personal computer for our practical implementation is:

- GNU Radio v3.3 (released on June 3rd, 2010)
- OpenBTS v2.6Mamou (released on August 1st, 2010)
- Asterisk v1.6.2.0

Even if this implementation work started using the OpenBTS v2.5.4Lacassine, we decided to switch to the more recent version because many bugs had been fixed. The configuration of the BTS and the smsqueue (the buffer used to forward text messages) have been simplified, but, above all, the power control procedure and the monitor of the physical channel have been remarkably enriched and the detection and tracking of the International Mobile Equipment Identity (IMEI) have been introduced.

In the following two subsections we provide an introduction to GNU Radio and Asterisk.

A. GNU Radio

GNU Radio is an Open Source project for Software Defined Radio started in 2000 by Eric Blossom. The main idea of this project is to convert all the hardware problems in software problems, or, in other terms, to transfer the complexity of radio systems design from the hardware to the software, trying to move the software as close to the antenna as possible. GNU Radio is a software suite constituted by several modules that can be combined with minimal hardware to implement radio systems based on a personal computer. It offers a graphical interface in order to facilitate the design of the SDR system through the creation of a graph, whose blocks represent the signal processing stages and the lines connecting the blocks highlight the data flow among the blocks. The signal processing blocks operate on infinite length data flow and have a finite number of input/output ports. The construction of the graph is accomplished by the language Python, which is defined as a scripting object-oriented language. In fact, it shows to be both easy and flexible as a scripting language and powerful and rich as a programming language. Furthermore, Python is:

- open source;
- portable. It is written in ANSI C, so it is possible to implement an interpreter for the most important software platforms;
- rich of libraries and mechanisms as a popular programming language.

Therefore, the GNU Radio architecture is based on a hybrid C++/Python where the signal processing blocks are implemented in C++, while the graph, the functioning rules definition and the setting options are implemented in Python. The integration among the two programming languages is realized through SWIG, which allows Python to connect the blocks and run them without any interpretation. GNU Radio offers the SDR designer more than 100 sinks, sources and primitives of I/O, TCP, high-speed AD and DA, audio cards, filters, NCO, VCO, modulators, demodulators, FEC, etc. Furthermore, it offers on-the-fly reconfigurability of the designed system through the Graphical User Interface (GUI), built with WxPython and blocks visual editing and connection through GNU Radio Companion (GRC), which translate the blocks graph in a Python script.

B. Asterisk

In OpenBTS, all the functions, which in the GSM are provided by the Base Station Controllers (BSCs), the Mobile Switching Centers (MSCs), the Home and Visitor Location Registers (HLR, VLR), are not implemented, because the USRP is used to provide only the GSM “Um” (Fig. 1) air interface to the GSM devices that are considered as Session Initiation Protocol (SIP) endpoints. In order to allow the BTS to correctly route calls and text messages, as well as for the registration and authentication of the GSM devices with the network (that happen in the HLR), it is necessary to use a communication server, such as Asterisk. By installing Asterisk and connecting it with the OpenBTS, Subscriber Identity Modules (SIM) will be considered as SIP users, and their International Mobile Subscriber Identity (IMSI) will be used as an SIP username. Thus, the IMSI of the GSM device has to be already set in the setting file of Asterisk. In order to substantiate the last consideration, the OpenBTS is logically connected in Fig. 3 to the Asterisk PBX, which manages calls and text messages among the GSM devices and the VoIP terminals.

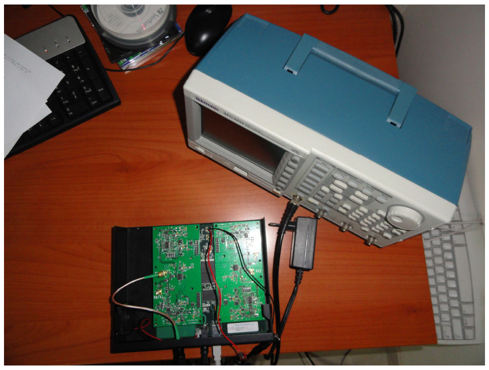
Fig. 4. The hardware necessary to implement a GSM BTS.

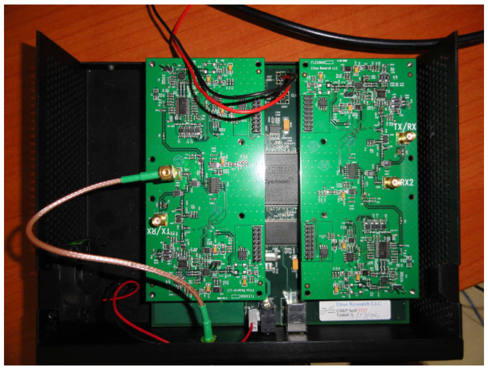

Fig. 5. Detail of the motherboard modified.

V. Putting it all together

In this last section, we will introduce and explain some issues to face when implementing a GSM BTS through USRP and GNU Radio.

When a GSM connection is established over the Um interface, the GSM (which a FDMA/TDMA system) assigns an Absolute Radio Frequency Channel Number (ARFCN) and a definite time slot. The ARFCN consists of two physical radio carriers. The GSM is synchronized with the BTS clock through the Synchronization Channel (SCH) and Frequency Correction Channel (FCCH). The SCH transmits the current TDMA time and the FCCH generates a tone used by the GSM device to command its local oscillator. The main hardware issue is related to this timing process. In fact, the internal clock of the USRP is set to work on 64 MHz, whereas the GSM system has to work on a clock frequency that is a multiple of 13 MHz, in order to make the synchronization between BTS and mobile devices possible. If we consider that the maximum error allowed by GSM specifications is 45 Hz, and that the simple clock oscillator produces a KHz error on the RF carrier, it is understandable why the current USRP needs to be modified to allow RX/TX GSM signalling. Specifically, the effect of the lack of synchronization produces a beacon signal from the BTS that is outside the frequency range scanned by the mobile devices during the roaming phase. The solution to this problem is given by the installation of an external clock on the motherboard. The external clock has to be set according to the GSM specifications. In our implementation we used a signal source that produces a 52 MHz square wave, with an amplitude of 1.5 V (0-1.5 V) and an offset of 750 mV. In order to install the external clock, we had to modify the USRP motherboard by removing the physical connection of the board to the internal clock and enabling the input of a different clock signal. The visual result of the modification accomplished on the motherboard are shown in Fig. 4 and in detail in Fig. 5, where the red and black wires at the top of the figure are connected to the external clock section of the motherboard and transport the signal of a signal generator.

With the new configuration, all the cellular phones used for the experiment were able to detect the presence of a BTS and register with it. In Fig. 6 we show the display of one of the devices connected to the OpenBTS.

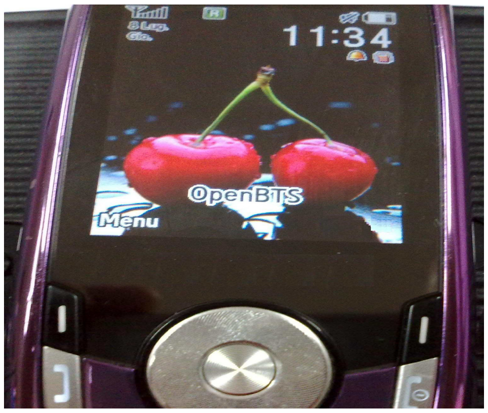

Fig. 6. Example of GSM device connected to the OpenBTS.

As we said in Subsection IV-B, in order to allow the devices to make calls and send text messages, we must first associate their IMSI with a SIP username. This is done in the file `/etc/asterisk/sip.conf`,where we specify the following instructions:

```
[IMSI22288141xxxxxxx]
callerid=2106
canreinvite=no
type=friend
context=sip-local
allow=gsm
host=dynamic
dtmfmode=info
```

Therefore, it is also necessary to discover the IMSI of each SIM. This can be achieved by connecting the cellular phones to the PC through a USB connection and running a Python script. The script interrogates the device about the IMSI contained in the SIM card, and allows it to store and use it. Another Asterisk file is in charge for routing the calls: `/etc/asterisk/extensions.conf` . It indicates how to route the calls incoming towards the known IMSI, for example: `exten=>2106,1,Macro(dialSIP,IMSI22288141xxxxxxx)` , where `dialSIP` specifies the behavior for the various cases (busy, congestion, no answer, etc): `[macro-dialSIP]`

```
exten=>s,1,Dial(SIP/$ARG1)
exten=>s,2,Goto(s-$DIALSTATUS,1)
exten=>s-NOANSWER,1,Hangup
exten=>s-BUSY,1,Busy(30)
```

```
exten=>s-CONGESTION,1,Congestion(30)
exten=>s-CHANUNAVAIL,1,playback(ss-noservice)
exten=>s-CANCEL,1,Hangup
```

Upon the completion of the Asterisk settings, the mobile terminals are ready to make/receive calls (in order to send text messages, a store-and-forward server also has to be set up in Asterisk) to/from other GSM devices as well as VoIP terminals. Since the environment where we tested the BTS functioning is crowded by electronic equipment, and the power available was very limited in order not to overlap with the existing providers, further issues were mostly related to the interference and the coverage of the BTS. Because of other systems' interference, the ARFCN initially chosen (#20, uplink 894 MHz-downlink 939 MHz) showed not to be strong enough to permit a standard phone call. Hence, mobile devices had to frequently re-search for available networks, wasting their batteries. In the following we summarized some of the results obtained by this implementation.

- $\approx 50m^2$ of coverage;
- Able to mantain connectivity with mobile users;
- ≈ 150 GSM to GSM calls established and successfully terminated;
- $\approx$ 100 VoIP connection established and successfully terminated;
- ≈ 50 text message delivered.

As future works we intend to test handover and resource allocation strategies by deploying more BTS and foreseeing the usage of a central controller unit.

VI. Conclusion

In this paper we reported the practical experience of implementing a lab-made version of a GSM BTS. In order to achieve this objective we used a PC, a multi-purpose motherboard for SDR on which we installed two RX/TX daughterboards (working on GSM900 frequencies) and a software project based on GNU Radio. The experience was particularly fruitful due to all the issues that arose and were solved by modifying both the hardware and the software. The new installed version of OpenBTS (released in August 2010) also allows the developer to investigate RSSI and power control issues. Currently, we are running more experiments with the new version in order to determine the exact BTS coverage and the relevance of the noise in the test environment.

References

[1] www.ettus.com.

[2] GNURadio: http://www.gnu.org/software/gnuradio.

[3] www.cgran.org.

[4] Project OpenBTS: http://openbts.sourceforge.net.

[5] R. W. Thomas, L. A. DaSilva, and A. B. Mackenzie, "Cognitive networks," in Proc. of IEEE DySPAN 2005, pp. 352-360, Nov. 2005.

[6] J. Mitola and G. Q. Maguire., "Cognitive radio: Making software radios more personal," IEEE Personal Communications, vol. 6, no. 4, pp. 13-18, 1999.

[7] J. Mitola, Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio. PhD thesis, Royal Institute of Technology (KTH), 2000.

[8] S. Haykin, "Cognitive radio: Brain-empowered wireless communication," IEEE Journal on Selected Areas in Communication, vol. 23, pp. 201-220, February 2005.

[9] J. O. Neel, J. H. Reed, and R. P. Gilles, "Convergence of cognitive radio networks," in Proc. of IEEE WCNC 2004, vol. 4, pp. 2250-2255, 2004.

[10] G. Ganesan and Y. Li, "Cooperative spectrum sensing in cognitive radio networks," in Proc. of IEEE DySPAN 2005, pp. 137-143, 2005.

[11] S. M. Mishra, A. Sahai, and R. W. Brodersen, "Cooperative sensing among cognitive radios," in Proc. of IEEE ICC 2006, vol. 4, pp. 1658-1663, 2006.

[12] P. Pawelczak, R. V. Prasad, L. Xia, and I. G. M. M. Niemegeers, "Cognitive radio emergency networks - requirements and design," in Proc. of IEEE DySPAN 2005, pp. 601-606, 2005.

[13] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, "A knowledge plane for the Internet," in Proc. of SIGCOMM '03, pp. 3-10, ACM Press, 2003.

[14] R. Saracco, "Forecasting the future of information technology: How to make research investment more cost-effective," IEEE Communications Magazine, vol. 41, pp. 38-45, December 2003.

[15] P. Mahonen, J. Riihijarvi, M. Petrova, and Z. Shelby, "Hop-by-hop toward future mobile broadband IP," IEEE Communications Magazine, vol. 42, no. 3, pp. 138-146, 2004.

[16] D. Bourse, M. Muck, O. Simon, N. Alonistioti, K. Moessner, E. Nicollet, D. Bateman, E. Buracchini, G. Chengeleroyen, and P. Demestichas, "End-to-end reconfigurability (E2R II): Management and control of adaptive communication systems." Presented at IST Mobile Summit 2006, June 2006.

[17] P. Demestichas, V. Stavroulaki, D. Boscovic, A. Lee, and J. Strassner, "m@ANGEL: Autonomic management platform for seamless cognitive connectivity to the mobile internet," IEEE Communications Magazine, vol. 44, no. 6, pp. 118-127, 2006.

[18] P. Sutton, L. E. Doyle, and K. E. Nolan, "A reconfigurable platform for cognitive networks," in Proc. of CROWNCOM 2006, 2006.