Communication System Design of Remote Areas using Openbts

Winarno Sugeng
Departement of Informatics Engineering
Institut Teknologi Nasional (Itenas)
Bandung – INDONESIA

Theta Dinnarwaty Putri
Departement of Informatics Engineering
Institut Teknologi Nasional (Itenas)
Bandung – INDONESIA

***Abstract*—OpenBTS is a software-based GSM BTS, which allows GSM cell phone users to make phone calls or send SMS (short messages), without using a commercial service provider network. OpenBTS is known as the first open source implementation of the GSM industry standard protocol. The OpenBTS network is a network that is easy to implement and also inexpensive in maintenance and installation. Communication using a cell phone is not only needed in urban areas but remote areas currently require. But the problem is that not all remote areas get services from commercial cellular operators. By implementing mobile phone communication using OpenBTS, remote communication is very likely to be implemented. In this research communication design was delivered using GSM mobile phones using OpenBTS with telephone and SMS services.**

***Keywords*—OpenBTS; GSM; communication; remote areas**

I. Introduction

Communication is a process in which a person or several people, groups, organizations, and communities create, and use the information to connect with the environment and others [1]. Even so, they can exchange information.

Voice Over Internet Protocol (VoIP) is the easiest way to make phone calls over the internet by sending packets through a packet switched networks [7]. This technology does not use the Public Telephone Network (PSTN), but calls are made through internet protocol data networks. VoIP has enormous benefits of increasing savings, high-quality voice and video streaming and some other value-added services. The current VoIP software applications that have been used are Skype, Google talks and Windows Live Messenger, and other similar applications.

OpenBTS (Open Source Base Transceiver Station) is an open source GSM-based GSM software, which allows GSM phones to communicate without a network of commercial cellular operators [4]. OpenBTS is known as the first open source implementation of the standard GSM industry protocol. For transmitting power of 100mW, the price is around 25 million rupiah per piece, far below the usual Cellular BTS which is usually in the order of a million to several billion rupiahs.

One of the criteria in remote areas contained in article 2 of Permendikbud Number 34 of 2012 are not available and/or very limited in public facilities, educational facilities, health facilities, electricity facilities, information and communication facilities, and clean water facilities.

To enable remote communication, in this case, communication in areas using a mobile phone, this condition includes communication at the same time. OpenBTS which is a GSM-paketbased software which is open source and allows communication free of cost from commercial cellular operators. Communication using OpenBTS can be built independently.

II. System Architecture

A. OpenBTS Architecture

Basically, VoIP-based communication utilizing the internet network [7] is sufficient, but in the field, the communication infrastructure that used to exist is GSM-based, the communication device that is widely used is GSM-based. For this reason, further research is needed to bridge this problem. OpenBTS is the solution for this, where OpenBTS is able to unite internet-based communication and GSM. So that communication technology still uses VoIP but the advanced network uses GSM.

OpenBTS is a BTS using a software-based GSM (Global System for Mobile Communications) network, which allows GSM mobile phone users to make phone calls or short messages in this case SMS (Short Message Service), without using a network of telephone operators commercial mobile. open source software used to revolutionize mobile networks using traditional telco, traditional, complex protocols and hardware systems with flexible internet protocols and network architecture [8]. In this study used OpenBTS V5.0, which features GPRS Support, A3-A8-A5 / 1 authentication and encryption, handover support, and more structured files.

The BTS function for sending and receiving GSM signals is replaced by USRP (Universal Software Radio Peripheral) which is controlled by transceiver applications. All activities on layer 1 (one) on the BTS side will be processed by USRP. Furthermore, the upper layer functions of BSC and MSC will be replaced by OpenBTS software. Then the VLR and HLR functions as registers will be replaced by asterisk software. All software on this network communicates using SIP (session initiation protocol). USRP is a high-speed Digital Signal Processing (DSP) based radio-based radio platform for the practical development and deployment of flexible RF systems from DC to 6 GHz. USRP is hardware and currently has several versions. The most recent version uses Gigabit Ethernet so it can be stored on top of the tower easily in order to cover a wider area [2]. Figure 1 is the default OpenBTS architecture

[8]. Figure 2 is an OpenBTS system of Range Network type RAD1 production, in this case, used in research. The price of the RAD1 OpenBTS module is quite expensive for the size of the study, currently, around the US $ 3,000 and must be purchased directly from the United States (https://rangenetworks.com/get-started/), entered in the USRP1 category. But when compared to the price of commercial BTS is not based on open-source software that is widely used for the same specification OpenBTS price is very cheap, because the price of commercial BTS can reach US$ 300,000. Commercial BTS is about 100 (one hundred) times more expensive than OpenBTS.

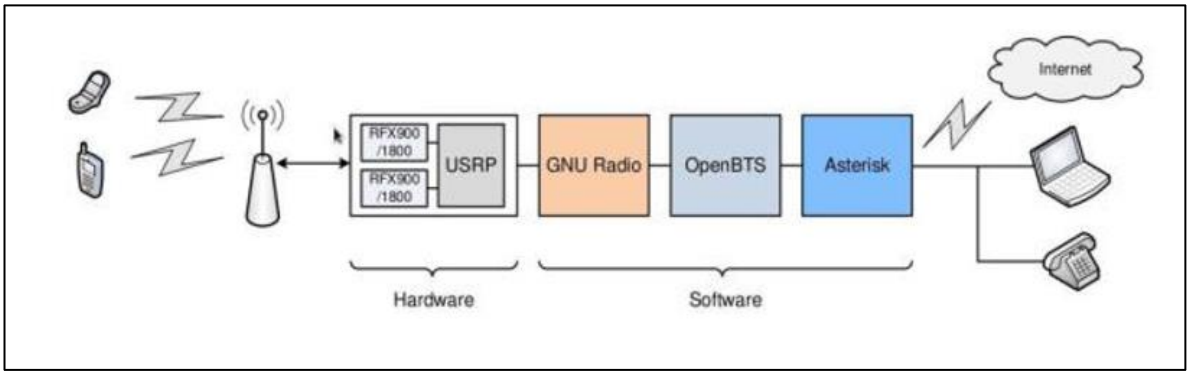


Fig. 1. OpenBTS standard architecture [2].

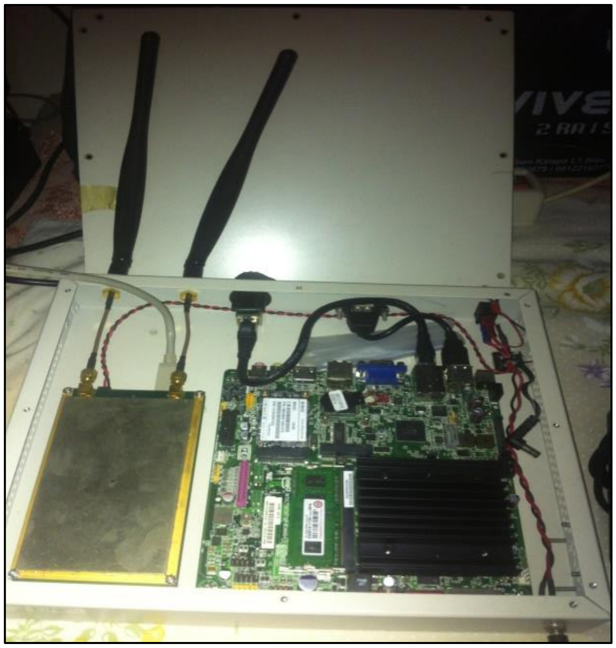


Fig. 2. OpenBTS module type USRP1 from Range Network

B. UHD

UHD (USRP Hardware Driver) is a device driver provided by Research Ettus for use by USRP products. This product supports Linux, MacOS, and Windows platforms. Some frameworks include GNU Radio, LabVIEW, Matlab and Simulink using UHD [6]. Functions provided by UHD can also be accessed directly with the UHD API, which provides native support for C ++. UHD also provides the portability of the entire family of USRP products. Applications developed for a particular USRP model will support other USRP models if proper consideration is given to sample rates and other parameters. PyBOMBS (Phyton Build Overlay Managed Bundle System) is a management system for resolving dependencies or a collection of multiple applications for radio software including UHD installations.

C. Asterisk

Asterisk is a free and open source framework for building communications applications that can transform ordinary computers into rich communications server or central phone features [5]. The Asterisk function used in OpenBTS is PBX Switching Core, which functions to handle incoming calls in the direction of Asterisk, calls can come from various interfaces and Asterisk can be used to create and deploy a wide range of telephony applications and services, including IP PBXs, VoIP gateways, calls center ACDs and IVR systems. Asterisk is released under the GNU General Public License (GPL) and is available for free download.

D. Smqueue

OpenBTS requires to store and forward facilities such as the SMS Center of a conventional GSM network to send SMS. This function is represented by a smqueue application [3]. Smqueue uses the RFC 3428 standard as a method for sending Instant messages between users on the OpenBTS network. This standard is an extension of the SIP, therefore the procedure for connecting also follows the protocol.

III. Design Process

The design process begins by installing USRP (Universal Peripheral Radio Software) which is equipped with USRP drivers using GnuRadio to introduce the software operating system used in Linux Ubuntu. After GnuRadio is installed and knows USRP, OpenBTS's open process is the main software that utilizes USRP. The main key of OpenBTS is an IP-based telephone center (Softswitch), in this study Asterisk was then a means of registering a GSM SIM card. The process is carried out with a GSM communication system like the services of commercial cellular operators in general.

The system used is a collection of several software and devices that communicate with each other. OpenBTS and all software that is on the list of needs for computers.

The entire system components designed and their interconnections are shown in the block diagram in Figure 3.

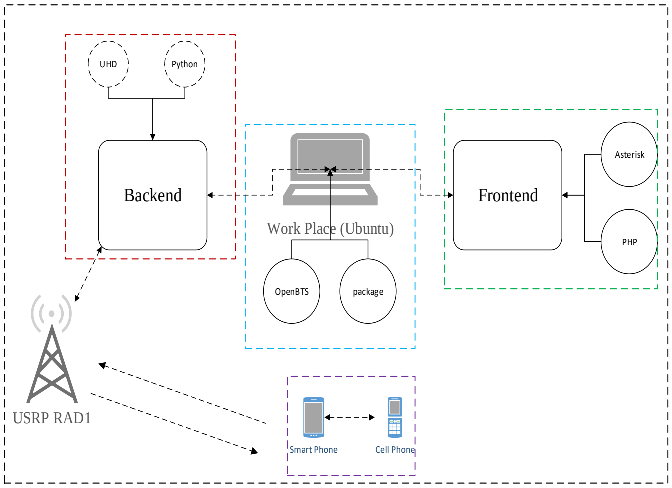


Fig. 3. System Block Diagram

From Figure 3 it can be seen that there are 3 main systems namely backend, Works Place, and frontend. The backend itself is a link between USRP RAD1 and workplace, using the UHD-hosted package. The workplace itself is a place where the operator/admin organizes the network through the OpenBTS interface and other supporting packages. While frontend is a system that regulates the relationship between users (Smartphones, cell phones) and other users.

The hardware component used is the RangeNetwork USRP type RAD1 (USRP1), while the software used is the Linux Operating System Ubuntu 14.04 32bit LTS, GNURadio, OpenBTS 5.0 (asterisk, sipauthserve, smqueue, openbts).

Before the installation process, some initial configuration is required. The first major configuration is USRP Handling, Ubuntu uses udev to handle hotplug devices, and in default does not provide non-root access to USRP. For this reason, several scripts are done to make USRP accessible to non-root, in this case, to handle USRP connect via USB both live and hotplug. If the USRP handling process is successful, the USRP will be recognized by the operating system.

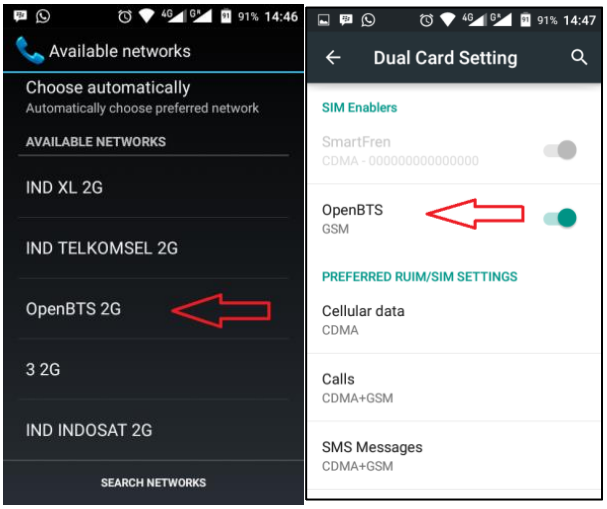
The installation process starts with the installation of development packages so that the installation supports UHD, namely: libboost-all-dev libusb-1.0-0-dev python-cheetah doxygen python-docutils autoconf libtool libosip2-dev libortp-dev g ++ sqlite3 libsqlite3-dev erlang libreadline6 -dev libncurses5-dev libuhd-dev libuhd003 libboost-dev ntp. The installation process continues with the GNURadio, OpenBTS installation. After the installation process, the basic configuration process on SQLite is done to create the OpenBTS database. Before being activated, the system reboots so that all installations work. The final stage is the activation of the following applications:

1) Asterisk, as a telephone operator machine
2) Sipauthserve, as a registration machine for the SIP protocol
3) Smqueue, as an SMS engine
4) Openbts, as a BTS engine

IV. Experiments And Results

A. *Telephone Card Registration*

The use of the system starts with new users who have not registered in the OpenBTS system to register, by configuring the BTS option on the mobile phone in the BTS name option selected as the name of the built BTS, in this case, the name configured is OpenBTS, as shown in Figure 4.


91% 14:46

Available networks

Choose automatically
Automatically choose preferred network

AVAILABLE NETWORKS

IND XL 2G

IND TELKOMSEL 2G

OpenBTS 2G

3 2G

IND INDOSAT 2G

SEARCH NETWORKS

91% 14:47

Dual Card Setting

SIM Enablers

SmartFren
CDMA - 000000000000000

OpenBTS
GSM

PREFERRED RUIM/SIM SETTINGS

Cellular data
CDMA

Calls
CDMA+GSM

SMS Messages
CDMA+GSM

Fig. 4. Registration Process

The operator's technical and administrative process before registering is registering the IMSI and giving a telephone number, in this case mapping the selected telephone number to IMSI into the OpenBTS database.The process is as follows:

IMSI 1st mobile 510113157642463, this will be numbered 0201063001, the order is:

```
sudo asterisk -rx "database put IMSI IMSI510113157642463 0201063001"
sudo asterisk -rx "database put PHONENUMBER 0201063001 IMSI510113157642463"
```

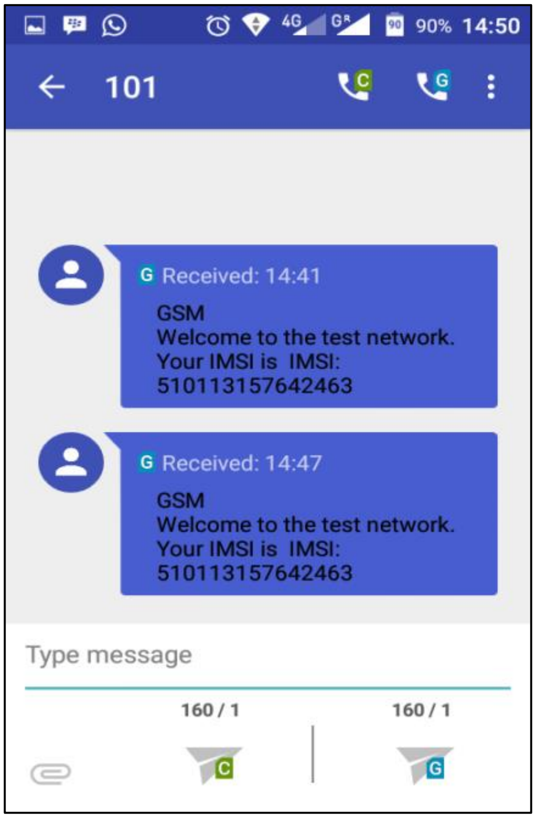
IMSI 2nd mobile 510101421302212, this will be numbered 0201063002, the commands are:

```
sudo asterisk -rx "database put IMSI IMSI510101421302212 0201063002"
sudo asterisk -rx "database put PHONENUMBER 0201063002 IMSI510101421302212"
```

The test results after number mapping can be seen in the asterisk database, via the asterisk -rx command "database show" will read as follows:

```
/IMSI/IMSI510101421302212          : 0201063002
/IMSI/IMSI510113157642463          : 0201063001
/PHONENUMBER/0201063001            : IMSI510113157642463
/PHONENUMBER/0201063002            : IMSI510101421302212
```

After the registration process the operator then the telephone operator through the registration process, in this case, activates the IMEI sim card used which has been registered in the database server and the user will get a new alias or phone number that can be used in the OpenBTS network. The activation process by sending an SMS to number 101 with the contents of the telephone number message that has been mapped, if the number activation process is successful, the system will send an SMS sign of successful registration activation as shown in Figure 5.



G Received: 14:41

GSM
Welcome to the test network.
Your IMSI is IMSI:
510113157642463

G Received: 14:47

GSM
Welcome to the test network.
Your IMSI is IMSI:
510113157642463

Fig. 5. Successful Registration

B. Telephone Service

Telephone services starting with the system will check the IMSI data and the destination number to be registered or not. If it is registered, it will be connected to the destination number and the phone record data will be stored in the system database. If the destination number is not registered, the system will send a warning and if the destination number is busy, the system will send a busy tone, as in Figure 6.

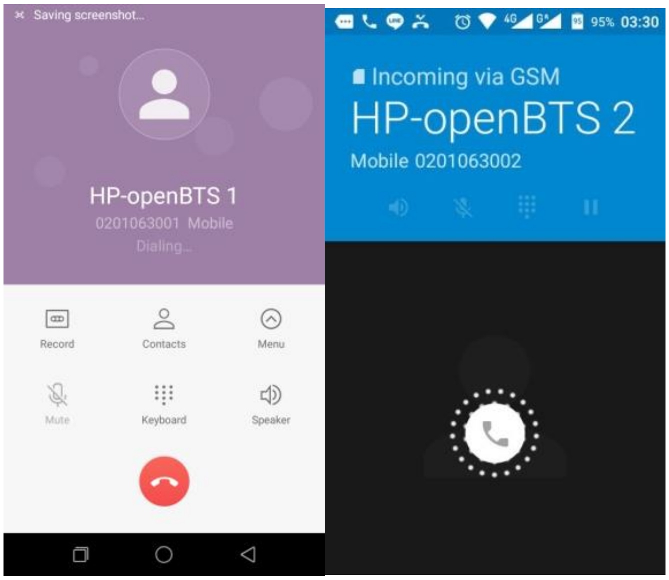
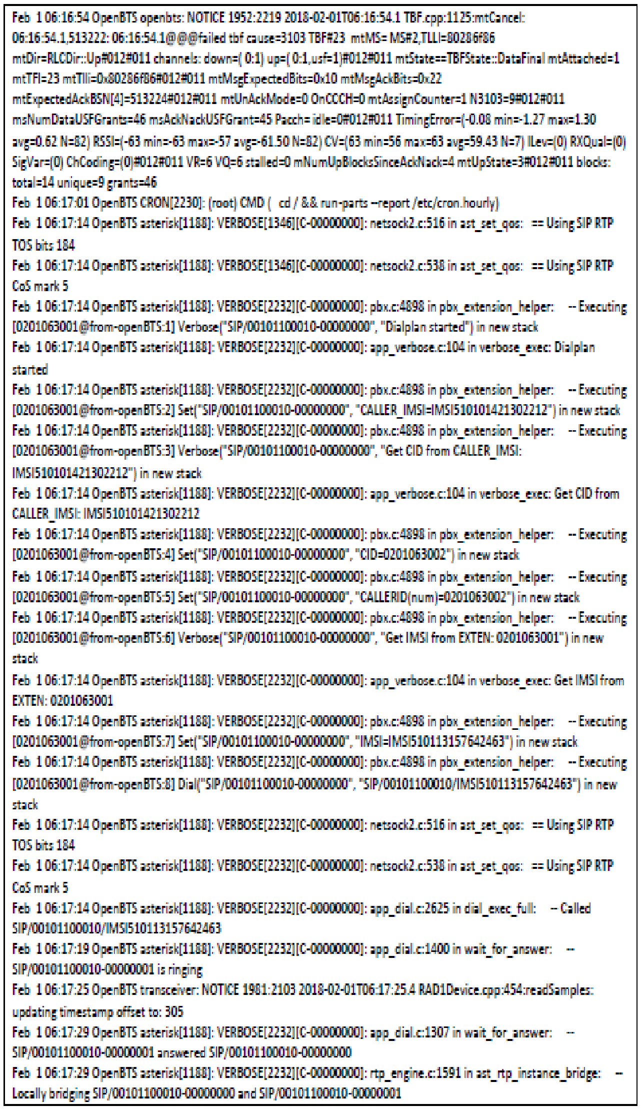


Fig. 6. Telephone Call



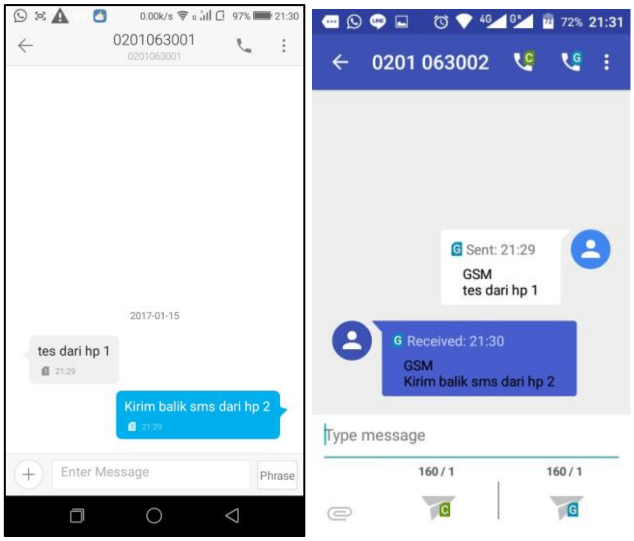
```
Feb  1 06:16:54 OpenBTS openbts: NOTICE 1952:2219 2018-02-01T06:16:54.1 TBF.cpp:1125:mtCancel:
06:16:54.1,513222: 06:16:54.1@@@failed tbf cause=3103 TBF#23  mtMS= MS#2,TLLI=80286f86
mtDir=RLCDir::Up#012#011 channels: down=( 0:1) up=( 0:1,usf=1)#012#011 mtState==TBFState::DataFinal mtAttached=1
mtTFI=23 mtTlli=0x80286f86#012#011 mtMsgExpectedBits=0x10 mtMsgAckBits=0x22
mtExpectedAckBSN[4]=513224#012#011 mtUnAckMode=0 OnCCCH=0 mtAssignCounter=1 N3103=9#012#011
msNumDataUSFGrants=46 msAckNackUSFGrant=45 Pacch= idle=0#012#011 TimingError=(-0.08 min=-1.27 max=1.30
avg=0.62 N=82) RSSI=(-63 min=-63 max=-57 avg=-61.50 N=82) CV=(63 min=56 max=63 avg=59.43 N=7) ILev=(0) RXQual=(0)
SigVar=(0) ChCoding=(0)#012#011 VR=6 VQ=6 stalled=0 mNumUpBlocksSinceAckNack=4 mtUpState=3#012#011 blocks:
total=14 unique=9 grants=46
Feb  1 06:17:01 OpenBTS CRON[2230]: (root) CMD (   cd / && run-parts --report /etc/cron.hourly)
Feb  1 06:17:14 OpenBTS asterisk[1188]: VERBOSE[1346][C-00000000]: netsock2.c:516 in ast_set_qos:   == Using SIP RTP
TOS bits 184
Feb  1 06:17:14 OpenBTS asterisk[1188]: VERBOSE[1346][C-00000000]: netsock2.c:538 in ast_set_qos:   == Using SIP RTP
CoS mark 5
Feb  1 06:17:14 OpenBTS asterisk[1188]: VERBOSE[2232][C-00000000]: pbx.c:4898 in pbx_extension_helper:    -- Executing
[0201063001@from-openBTS:1] Verbose("SIP/00101100010-00000000", "Dialplan started") in new stack
Feb  1 06:17:14 OpenBTS asterisk[1188]: VERBOSE[2232][C-00000000]: app_verbose.c:104 in verbose_exec: Dialplan
started
Feb  1 06:17:14 OpenBTS asterisk[1188]: VERBOSE[2232][C-00000000]: pbx.c:4898 in pbx_extension_helper:    -- Executing
[0201063001@from-openBTS:2] Set("SIP/00101100010-00000000", "CALLER_IMSI=IMSI510101421302212") in new stack
Feb  1 06:17:14 OpenBTS asterisk[1188]: VERBOSE[2232][C-00000000]: pbx.c:4898 in pbx_extension_helper:    -- Executing
[0201063001@from-openBTS:3] Verbose("SIP/00101100010-00000000", "Get CID from CALLER_IMSI:
IMSI510101421302212") in new stack
Feb  1 06:17:14 OpenBTS asterisk[1188]: VERBOSE[2232][C-00000000]: app_verbose.c:104 in verbose_exec: Get CID from
CALLER_IMSI: IMSI510101421302212
Feb  1 06:17:14 OpenBTS asterisk[1188]: VERBOSE[2232][C-00000000]: pbx.c:4898 in pbx_extension_helper:    -- Executing
[0201063001@from-openBTS:4] Set("SIP/00101100010-00000000", "CID=0201063002") in new stack
Feb  1 06:17:14 OpenBTS asterisk[1188]: VERBOSE[2232][C-00000000]: pbx.c:4898 in pbx_extension_helper:    -- Executing
[0201063001@from-openBTS:5] Set("SIP/00101100010-00000000", "CALLERID(num)=0201063002") in new stack
Feb  1 06:17:14 OpenBTS asterisk[1188]: VERBOSE[2232][C-00000000]: pbx.c:4898 in pbx_extension_helper:    -- Executing
[0201063001@from-openBTS:6] Verbose("SIP/00101100010-00000000", "Get IMSI from EXTEN: 0201063001") in new
stack
Feb  1 06:17:14 OpenBTS asterisk[1188]: VERBOSE[2232][C-00000000]: app_verbose.c:104 in verbose_exec: Get IMSI from
EXTEN: 0201063001
Feb  1 06:17:14 OpenBTS asterisk[1188]: VERBOSE[2232][C-00000000]: pbx.c:4898 in pbx_extension_helper:    -- Executing
[0201063001@from-openBTS:7] Set("SIP/00101100010-00000000", "IMSI=IMSI510113157642463") in new stack
Feb  1 06:17:14 OpenBTS asterisk[1188]: VERBOSE[2232][C-00000000]: pbx.c:4898 in pbx_extension_helper:    -- Executing
[0201063001@from-openBTS:8] Dial("SIP/00101100010-00000000", "SIP/00101100010/IMSI510113157642463") in new
stack
Feb  1 06:17:14 OpenBTS asterisk[1188]: VERBOSE[2232][C-00000000]: netsock2.c:516 in ast_set_qos:   == Using SIP RTP
TOS bits 184
Feb  1 06:17:14 OpenBTS asterisk[1188]: VERBOSE[2232][C-00000000]: netsock2.c:538 in ast_set_qos:   == Using SIP RTP
CoS mark 5
Feb  1 06:17:14 OpenBTS asterisk[1188]: VERBOSE[2232][C-00000000]: app_dial.c:2625 in dial_exec_full:    -- Called
SIP/00101100010/IMSI510113157642463
Feb  1 06:17:19 OpenBTS asterisk[1188]: VERBOSE[2232][C-00000000]: app_dial.c:1400 in wait_for_answer:    --
SIP/00101100010-00000001 is ringing
Feb  1 06:17:25 OpenBTS transceiver: NOTICE 1981:2103 2018-02-01T06:17:25.4 RAD1Device.cpp:454:readSamples:
updating timestamp offset to: 305
Feb  1 06:17:29 OpenBTS asterisk[1188]: VERBOSE[2232][C-00000000]: app_dial.c:1307 in wait_for_answer:    --
SIP/00101100010-00000001 answered SIP/00101100010-00000000
Feb  1 06:17:29 OpenBTS asterisk[1188]: VERBOSE[2232][C-00000000]: rtp_engine.c:1591 in ast_rtp_instance_bridge:    --
Locally bridging SIP/00101100010-00000000 and SIP/00101100010-00000001
```

Fig. 7. Telephone Call Debugging

The OpenBTS system debugging results in /var/log/syslog observations when used as a telephone service can be seen in Figure 7. It can be seen that the system runs the SIP protocol by calling IMSI, so it is not a call based on the telephone number, even though it is calling based on the telephone number, but actually the number is the card number, in this case, IMSI, while the telephone number is formed by the operator.

C. SMS Service

The system starts by entering the message via the 1st cell phone then the message is sent to the 2nd cell phone. The OpenBTS system will check IMSI data, fill in the message and destination number before sending it to the destination number. If the destination number is registered, the SMS data will enter the SMS database and the SMS queue system will then be sent to the destination number registered on the system. The shipping process can be seen in Figure 8.



2017-01-15

tes dari hp 1
21:29

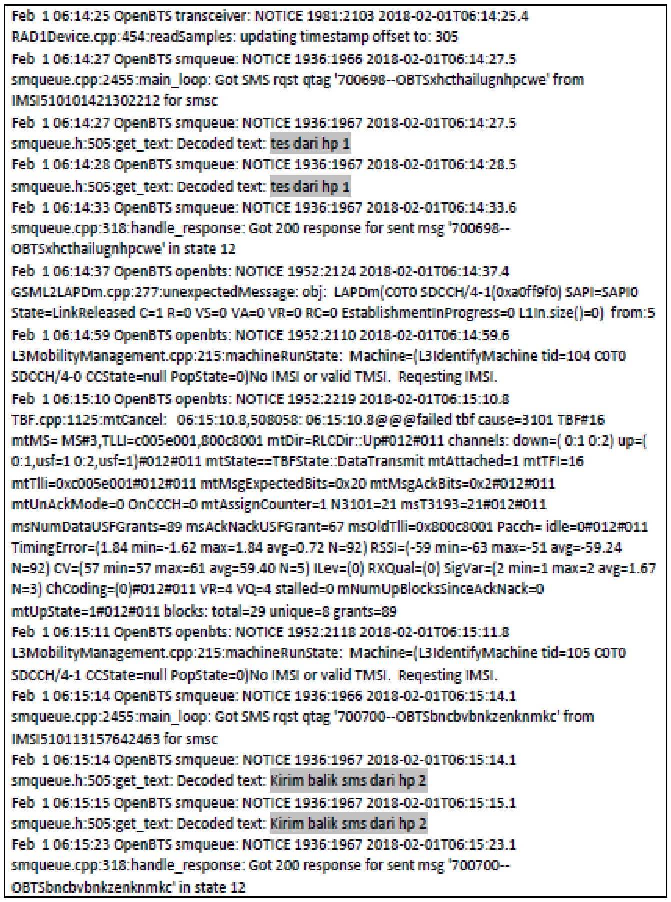
Kirim balik sms dari hp 2
21:29

Sent: 21:29
GSM
tes dari hp 1

Received: 21:30
GSM
Kirim balik sms dari hp 2

Fig. 8. SMS Between Cellphones

The OpenBTS system debugging results in /var/log/syslog observations when used as an SMS service can be seen in Figure 9. It appears that Smqueue happened to call -1st to the 2nd phone with the short message "test dari hp 1" and vice versa from the phone to-2 to the phone-1 with the contents of the short message "kirim balik sms dari hp 2".



```
Feb  1 06:14:25 OpenBTS transceiver: NOTICE 1981:2103 2018-02-01T06:14:25.4
RAD1Device.cpp:454:readSamples: updating timestamp offset to: 305
Feb  1 06:14:27 OpenBTS smqueue: NOTICE 1936:1966 2018-02-01T06:14:27.5
smqueue.cpp:2455:main_loop: Got SMS rqst qtag '700698--OBTSxhcthailugnhpcwe' from
IMSI510101421302212 for smsc
Feb  1 06:14:27 OpenBTS smqueue: NOTICE 1936:1967 2018-02-01T06:14:27.5
smqueue.h:505:get_text: Decoded text: tes dari hp 1
Feb  1 06:14:28 OpenBTS smqueue: NOTICE 1936:1967 2018-02-01T06:14:28.5
smqueue.h:505:get_text: Decoded text: tes dari hp 1
Feb  1 06:14:33 OpenBTS smqueue: NOTICE 1936:1967 2018-02-01T06:14:33.6
smqueue.cpp:318:handle_response: Got 200 response for sent msg '700698--
OBTSxhcthailugnhpcwe' in state 12
Feb  1 06:14:37 OpenBTS openbts: NOTICE 1952:2124 2018-02-01T06:14:37.4
GSML2LAPDm.cpp:277:unexpectedMessage: obj:  LAPDm(C0T0 SDCCH/4-1(0xa0ff9f0) SAPI=SAPI0
State=LinkReleased C=1 R=0 VS=0 VA=0 VR=0 RC=0 EstablishmentInProgress=0 L1In.size()=0)  from:5
Feb  1 06:14:59 OpenBTS openbts: NOTICE 1952:2110 2018-02-01T06:14:59.6
L3MobilityManagement.cpp:215:machineRunState:  Machine=(L3IdentifyMachine tid=104 C0T0
SDCCH/4-0 CCState=null PopState=0)No IMSI or valid TMSI.  Reqesting IMSI.
Feb  1 06:15:10 OpenBTS openbts: NOTICE 1952:2219 2018-02-01T06:15:10.8
TBF.cpp:1125:mtCancel:   06:15:10.8,508058: 06:15:10.8@@@failed tbf cause=3101 TBF#16
mtMS= MS#3,TLLI=c005e001,800c8001 mtDir=RLCDir::Up#012#011 channels: down=( 0:1 0:2) up=(
0:1,usf=1 0:2,usf=1)#012#011 mtState==TBFState::DataTransmit mtAttached=1 mtTFI=16
mtTlli=0xc005e001#012#011 mtMsgExpectedBits=0x20 mtMsgAckBits=0x2#012#011
mtUnAckMode=0 OnCCCH=0 mtAssignCounter=1 N3101=21 msT3193=21#012#011
msNumDataUSFGrants=89 msAckNackUSFGrant=67 msOldTlli=0x800c8001 Pacch= idle=0#012#011
TimingError=(1.84 min=-1.62 max=1.84 avg=0.72 N=92) RSSI=(-59 min=-63 max=-51 avg=-59.24
N=92) CV=(57 min=57 max=61 avg=59.40 N=5) ILev=(0) RXQual=(0) SigVar=(2 min=1 max=2 avg=1.67
N=3) ChCoding=(0)#012#011 VR=4 VQ=4 stalled=0 mNumUpBlocksSinceAckNack=0
mtUpState=1#012#011 blocks: total=29 unique=8 grants=89
Feb  1 06:15:11 OpenBTS openbts: NOTICE 1952:2118 2018-02-01T06:15:11.8
L3MobilityManagement.cpp:215:machineRunState:  Machine=(L3IdentifyMachine tid=105 C0T0
SDCCH/4-1 CCState=null PopState=0)No IMSI or valid TMSI.  Reqesting IMSI.
Feb  1 06:15:14 OpenBTS smqueue: NOTICE 1936:1966 2018-02-01T06:15:14.1
smqueue.cpp:2455:main_loop: Got SMS rqst qtag '700700--OBTSbncbvbnkzenknmkc' from
IMSI510113157642463 for smsc
Feb  1 06:15:14 OpenBTS smqueue: NOTICE 1936:1967 2018-02-01T06:15:14.1
smqueue.h:505:get_text: Decoded text: Kirim balik sms dari hp 2
Feb  1 06:15:15 OpenBTS smqueue: NOTICE 1936:1967 2018-02-01T06:15:15.1
smqueue.h:505:get_text: Decoded text: Kirim balik sms dari hp 2
Feb  1 06:15:23 OpenBTS smqueue: NOTICE 1936:1967 2018-02-01T06:15:23.1
smqueue.cpp:318:handle_response: Got 200 response for sent msg '700700--
OBTSbncbvbnkzenknmkc' in state 12
```

Fig. 9. SMS Debugging

D. Signal Quality

Voice quality and SMS reception are very dependent on the condition of the signal strength emitted by the antenna. As an illustration of the results of sound quality testing in this study using indoor standard antennas can be seen in table 1.

TABLE I. SOUND QUALITY RECEIVED BASED ON DISTANCE AND SIGNAL STRENGTH

Distance (m)	Signal Quality (dBm)	Signal Quality (mW)	Sound Quality
3	-53	0,0000050119	Very good
6	-55	0,0000031623	Very good
9	-57	0,0000019953	Good
12	-57	0,0000019953	Good
15	-59	0,0000012589	Enough
20	-64	0,0000003981	Not good
25	-67	0,0000001995	Not good
27	-71	0,0000000794	Bad
30	-81	0,0000000079	Bad

The next test is SMS, carried out on the message delay sent based on signal strength and distance from USRP, the results of the test can be seen in table 2.

TABLE II. SIGNAL QUALITY TESTING BASED ON DISTANCE AND QUALITY OF SMS

Distance (m)	Signal Quality (dBm)	Signal Quality (mW)	SMS sending
3	-51	0,0000079433	Sent
6	-53	0,0000050119	Sent
9	-56	0,0000025119	Sent
12	-56	0,0000025119	Sent
15	-60	0,0000010000	Sent
20	-64	0,0000003981	Sent
25	-69	0,0000001259	Sent
27	-71	0,0000000794	Not sent
30	-85	0,0000000032	Not sent

From tables 1 and 2 it can be concluded that the quality of service from OpenBTS will depend on the distance and also the signal quality obtained by the user. The mathematical calculations to get signal quality (mW) are as follows:

$$P_{(\mathrm{mW})} = 1\mathrm{mW} \cdot 10^{(P(\mathrm{dBm})/\ 10)}$$

Example : 13dBm = ? Mh

$$P_{(\mathrm{mW})} = 1\mathrm{mW} \cdot 10^{(13\mathrm{dBm}/\ 10)} = 19.95\mathrm{mW}$$

So 13dBm is 19.95mW.

V. CONCLUSION

From the results of research conducted, the design of mobile telephone communications without involving commercial cellular operators is very likely to be applied in remote areas, this is because without involving commercial cellular operators telephone and SMS services can be realized. In connection with sound quality and SMS reception, it can be improved by using an amplified outdoor antenna. The main problem in implementing the system is only on OpenBTS hardware that must be imported from outside because in Indonesia no one sells the equipment. But with the progress of the Online-based purchase system, the procurement of OpenBTS hardware is not an obstacle. From the results of the research achievements, further research that is currently being developed is GPRS data service to provide information and post-disaster data to the SAR team regarding the location of the victim and can be seen in real-time through a web browser on a smartphone. This research can be realized because communication can be realized independently with smartphone communication devices that are commonly sold in the market but in realizing a communication system without involving commercial operators.

REFERENCES

[1] Febrianty, O., “Definisi Komunikasi”, 2015, (Online), (http://slideplayer.info/slide/2838078/, accessed 20-12-2017)

[2] Iedema, “Getting Started with OpenBTS”, O'Reilly, Range Networks, 2015

[3] Juhana and Hatorangan, “Mobile phone auto registration to OpenBTS-based cellular network in disaster situation”, 8th International Conference on Telecommunication Systems Services and Applications (TSSA), 2014

[4] Purbo, O., W., “Bongkar Rahasia openBTS untuk Jaringan Operator Selular”, Andi Offset, Yogyakarta, 2013.

[5] Utama, “Perancangan Dan Instalasi SIP Sebagai Sarana Komunikasi Audio Visual Dengan Asterisk Sebagai IP PBX Server”, Jurusan Teknik Elektro Fakultas Teknik Universitas Diponegoro, Semarang, 2012

[6] Wibowo, Rizal, and Gunawan, “Implementasi Voice Service Pada Jaringan GSM Menggunakan OpenBTS V5.0”, e-Proceeding of Applied Science : Vol.1, No.2 Agustus 2015

[7] Jalendry, S. and Verma, S., “A Detail Review on Voice over Internet Protocol (VoIP)”, Dept. of CSE., Manav Rachna International University,Faridabad, Haryana, India, 2015

[8] _____, “OpenBTS Application Suite Release 4.0”, Range Networks, Inc., San Francisco, California 94107, United States of America, 2014