Implementing GSM Base Transceiver Station for Disaster Management

A Project Report

submitted by

Hari Venkat Kiran
EE13B026

under the guidance of
Dr.Devendra Jalihal

in partial fulfilment of the requirement for
the award of the degree of

Bachelor of Technology

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS

JUNE 2017

Contents

ABSTRACT

This project is aimed at implementing a small scale private GSM network using open source software and hardware enabled with WiFi and support for VoIP protocols.The project also aims to investigate features of such a GSM base station including robustness, the range of coverage and power consumption. The typical base transceiver station will require a host processor to run the telephony engine that takes care of the GSM protocols and connects to the VoIP Server. Raspberry Pi Model 3B, Model 2B and a HP laptop are the three different host processors used here to compare their relative features. The telephony engine used here is the open source project Yate and the interface with the GSM layer is through its extension YateBTS. The radio transceiver used to communicate at the various GSM bands is implemented using a Software Defined Radio. The radio in use is Nuand's BladeRF x40. The VoIP server is setup using the Raspberry Pi Distro called RasPBX on a Model 2B. The phone application CSipSimple is used for registering the SIP clients. The BTS setup using the above components is able to broadcast a network, place GSM calls within the network and connect to SIP clients through an outbound VoIP server successfully. USB power meters are used to test the power consumption for the BTS under different configurations and different processors. Range tests are conducted by measuring the RSSI of the network on different connected phones.

ABBREVIATIONS:

BTS	Base Transceiver Station
IAX	Inter Asterisk Exchange
NIB	Network in a Box
PBX	Private Branch Exchange
SDR	Software Defined Radio
SIP	Session Initiation Protocol
VoIP	Voice Over Internet Protocol

LIST OF TABLES

INTRODUCTION

Today GSM network has found widespread coverage all over India and even includes most of the rural regions of the country. However there are still many remote places such as along railway tracks in rural places containing little to almost zero population, where there is no network coverage. Under normal conditions the lack of coverage in such places do not raise any concern. However these locations which are cut-off from the rest of the world present a serious problem during times of emergency when lack of real time communications adversely affect relief efforts. One way to resolve this is by creating a fast deployable local network that will enable communication in the local area and also have a dedicated backhaul to the rest of the world. This local network should work without the need for specialised handsets so that it is easily adaptable with the existing mobile phones and hence using the GSM/GPRS framework alongside wireless VoIP protocols through WiFi will be the best fit solution.This private network can then be connected to the existing GSM/LTE networks through a dedicated backhaul. The components involved in setting up this Base Transceiver Station will be determined by the range of coverage required, the number of channels and also the power requirements of the system.

BTS SYSTEM OVERVIEW

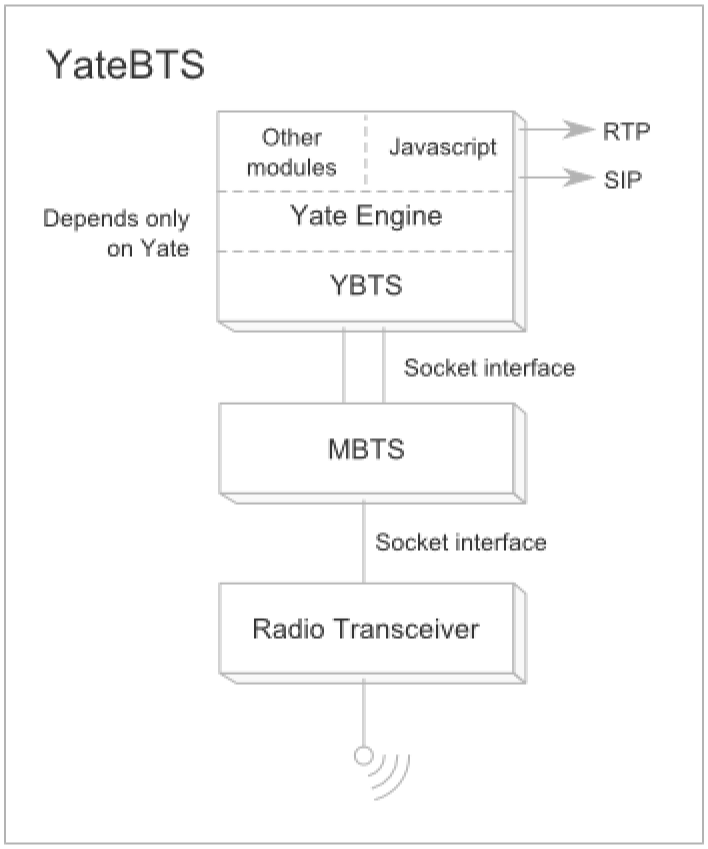
The Base Transceiver Station here consists of a Raspberry Pi acting as a linux server that runs a set of open source software components developed under the Yate and YateBTS project. It also consists of a Software Defined Radio which acts as the transceiver in the system. Additionally Asterisk is also included to form a PBX that allows gateway to the outside world through VoIP. Incidentally Yate alone is sufficient to connect through VoIP but Asterisk is used here to make the system modular.

Yate and YateBTS:

Yate is a telephony engine whose name stands for **Y**et **A**nother **T**elephony **E**ngine. It has a flexible routing engine that can easily be extended to voice, data, video and instant messaging. Yate is mainly written in C++ and supports various scripting languages. Yate is licensed under GNU General Public License.

YateBTS is a software implementation of the GSM/GPRS Radio network based on Yate. As per the official documentation of yateBTS, there are two main parts : *" the lower layer managed by the MBTS and the radio transceiver which handles the GSM part of the system and the Network layer handled by Yate and consists of Yate Module, YBTS and other yate application modules (Network in a Box, Javascript, accfile for outbound calls on SIP/IAX). The*

MBTS module connects to the Radio transceiver using a socket interface similar to the connection with Network layer. The Javascript module handles RTP and SIP."



<u>Fig. Block Diagram of YateBTS System from official yate wiki</u>

The MBTS is the radio resource manager handling the GSM aspects of the communication and is interfaced with the VoIP features of Yate such as IAX/SIP through yateBTS. When YateBTS is used as a standalone GSM/GPRS network connected by VoIP to the outside world then it is called **Network in a Box (NIB)**

<u>Software Defined Radios (SDR) :</u>

As digital signal processing started becoming more and more powerful, conventional radio hardwares like mixers, filters, amplifiers etc. could be replaced by software implementations of the same. These software implementation essentially create a radio system that could be programmed to transmit and receive a wide variety of protocols as per the requirement. Originally developed for military applications, the decreasing cost of computing resources has made the device an affordable and viable option replacing expensive hardware.

There are quite a few different SDRs available right now and for the purpose of this project, **Nuand's BladeRF x40** is chosen. The device costs around USD420 and is an open source hardware with the design files made available to the public. The manufacturer specifications of the BladeRF x40 are as follows:

Frequency Range:	300MHz to 3.8GHz
FPGA:	Altera Cyclone IV 40kLE
Power Options:	Fully Bus Powered over USB 3.0
	External Power Option via 5V DC barrel Jack

The BladeRF x40 contains a 40kLE FPGA that will run the software containing the signal processing algorithms and control the Transceiver for example: GNU Radio etc.
BladeRF has stable software support in Windows, Mac and Linux. The firmware version can be upgraded using a command line interface called **bladeRF-cli**

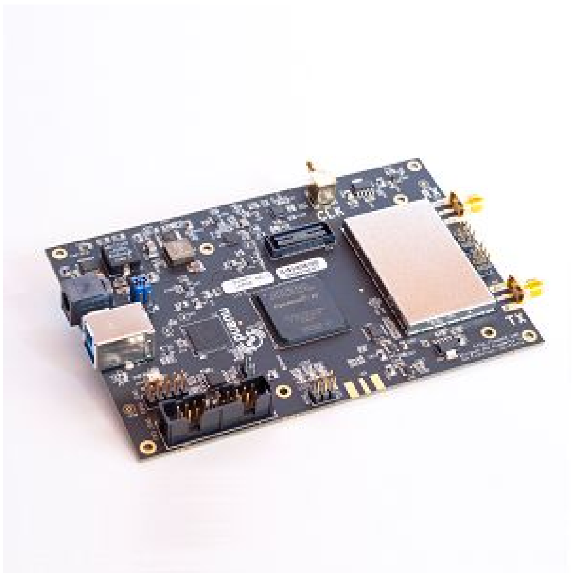


Fig. BladeRFx40

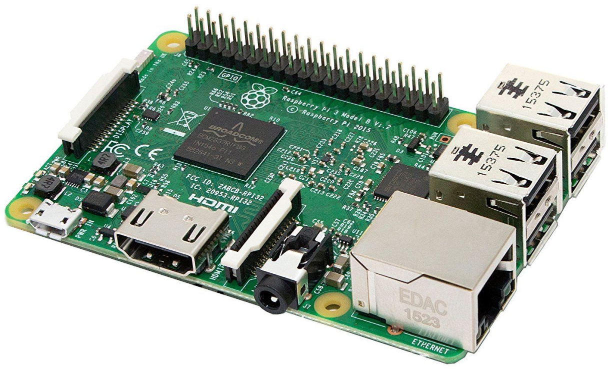


Fig. Raspberry Pi 2

Asterisk and FreePBX:

Asterisk is a software implementation of a Private Branch eXchange **(PBX)**. Asterisk provides features like voice call, Interactive Voice Response, conference calling etc. Asterisk can also act as a proxy and provide communication between different protocols including Voice over Internet Protocols **(VoIP)** like Session Initiation Protocol **(SIP)**, Inter Asterisk eXchange **(IAX)**, and also as a gateway to Public Switched Telephone Networks **(PSTN)** with the help of specialised hardware example: Sangoma cards for PSTN. Asterisk is released under dual licence model using GNU General Public License for free software distribution and also a proprietary software license to distribute commercially for unpublished components.

FreePBX is a web based open source graphical user interface for Asterisk. The graphical UI allows easier registration of users, configuration of extensions and other administrative tasks. It is licensed under the GNU General Public License version 3. It supports installation in Raspberry Pi.

Raspberry Pi:

Raspberry Pi is a credit card sized single board computer. They now form a series of computers with varying processing power, memory, wireless adapters and power consumption. The Pi hardware consists of a Broadcomm System on Chip which includes an ARM based CPU, USB 2.0 peripherals, one Ethernet Port, Wireless adapter-Wifi Bluetooth.

Specifications of Raspberry Pi 2 Model B and Raspberry Pi 3 Model B :

Specs	**Pi 3 Model B**	**Pi 2 Model B**
SoC	BCM 2837	BCM 2836
CPU	Quad COrtex A53 @ 1.2GHz	Quad COrtex A7 @ 900 MHz
RAM	1GB SDRAM	1GB SDRAM
Wireless	802.11n / Bluetooth 4.0	None

The Raspberry Pi is recommended to be powered with a 5V 2A power supply for Pi 3 model B according to the official documentation from the Raspberry Pi Foundation.

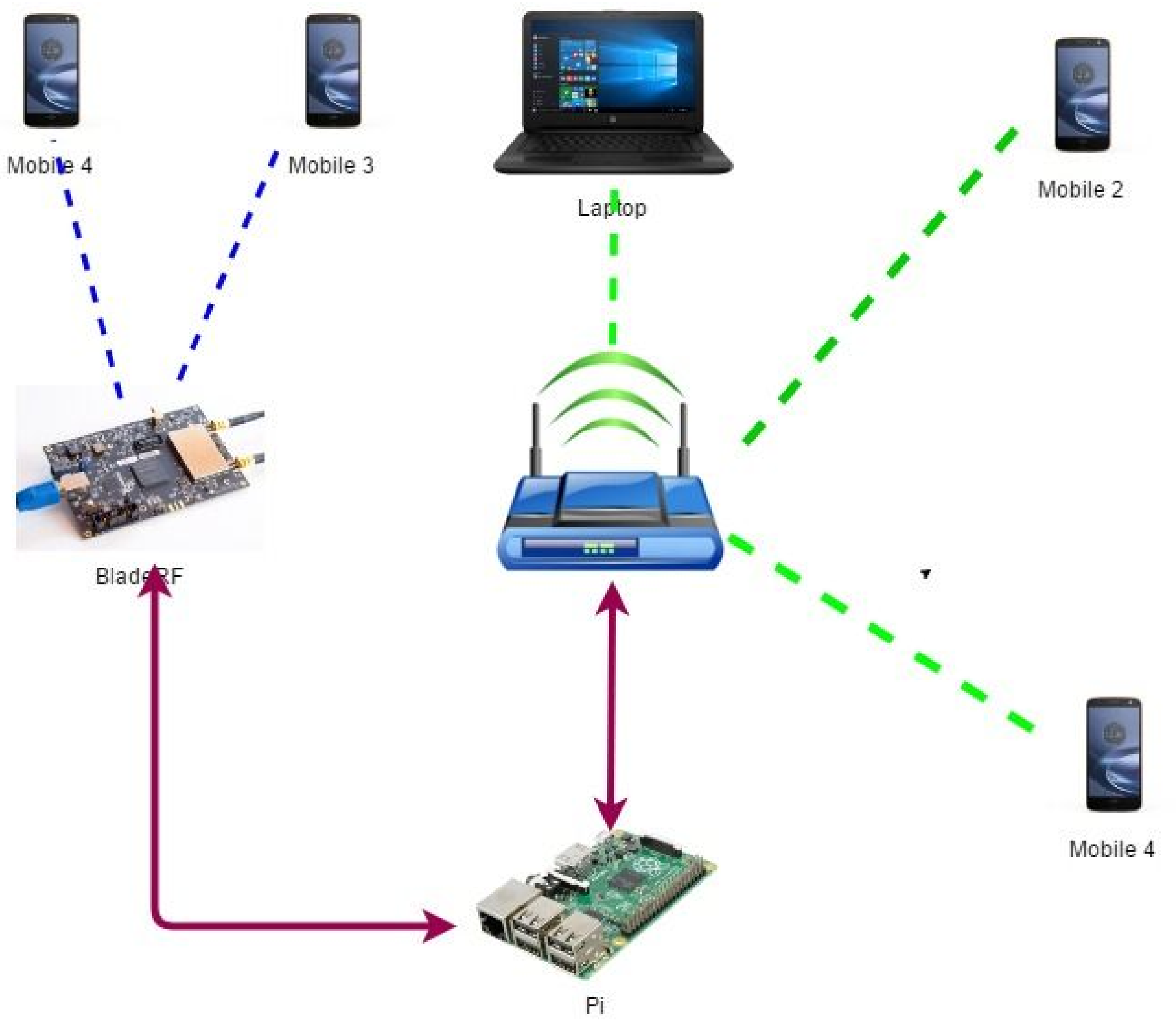


Fig. Block Diagram of the Yate System (Green-WiFi, Blue GSM 2.5G)

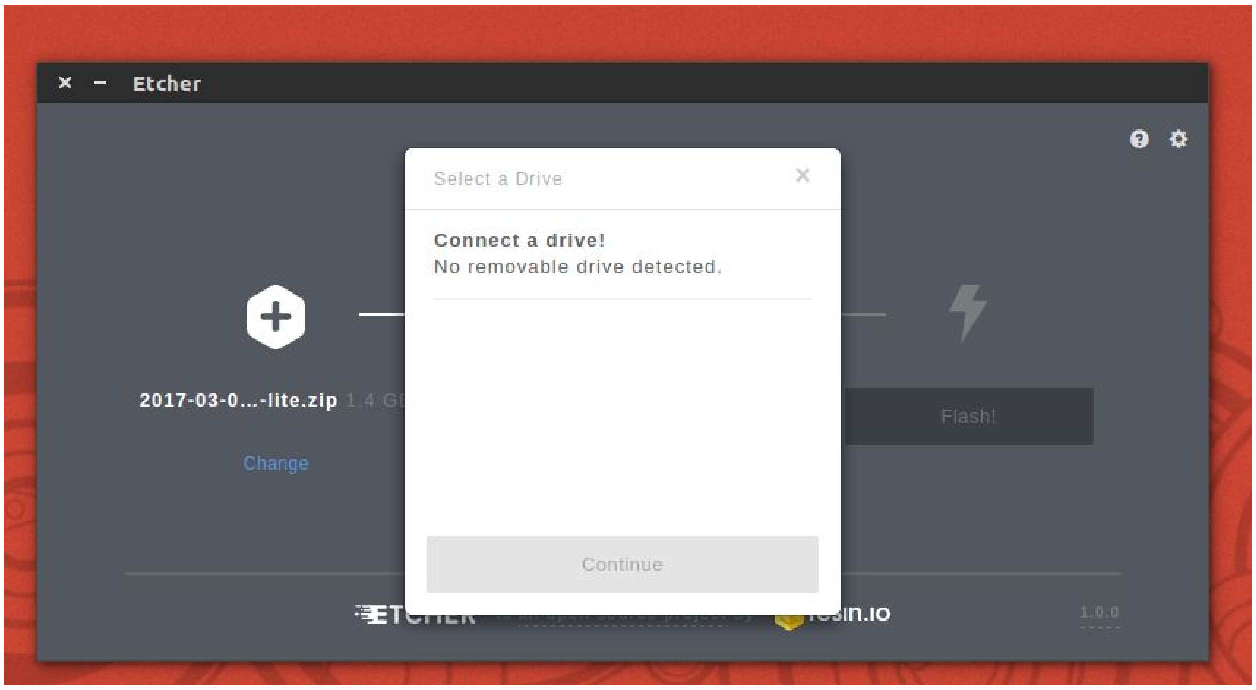
INSTALLATION AND TROUBLESHOOTING -YATEBTS

Downloading the Raspbian Image:

The operating system can be downloaded at https://www.raspberrypi.org/downloads/raspbian/
The operating system used in this guide is Raspbian Jessie lite.
The image has to be installed in a memory card and the easier way to do it is by using an image writing tool. The official page of Raspberry Pi suggest using "**Etcher**" available at https://etcher.io/ .
Ensure to download the appropriate version of the software based on your operating system, and extract it.

- Connect the removable device onto which the operating system is to be loaded. A memory card reader can be used, if the memory card slot is unavailable.
- Open Etcher and Select the Raspbian operating system zip file in the first step.
- Select the correct device to create the live image.

Please make sure that the device selected is the "correct" one or you risk erasing any other drive accidentally selected.



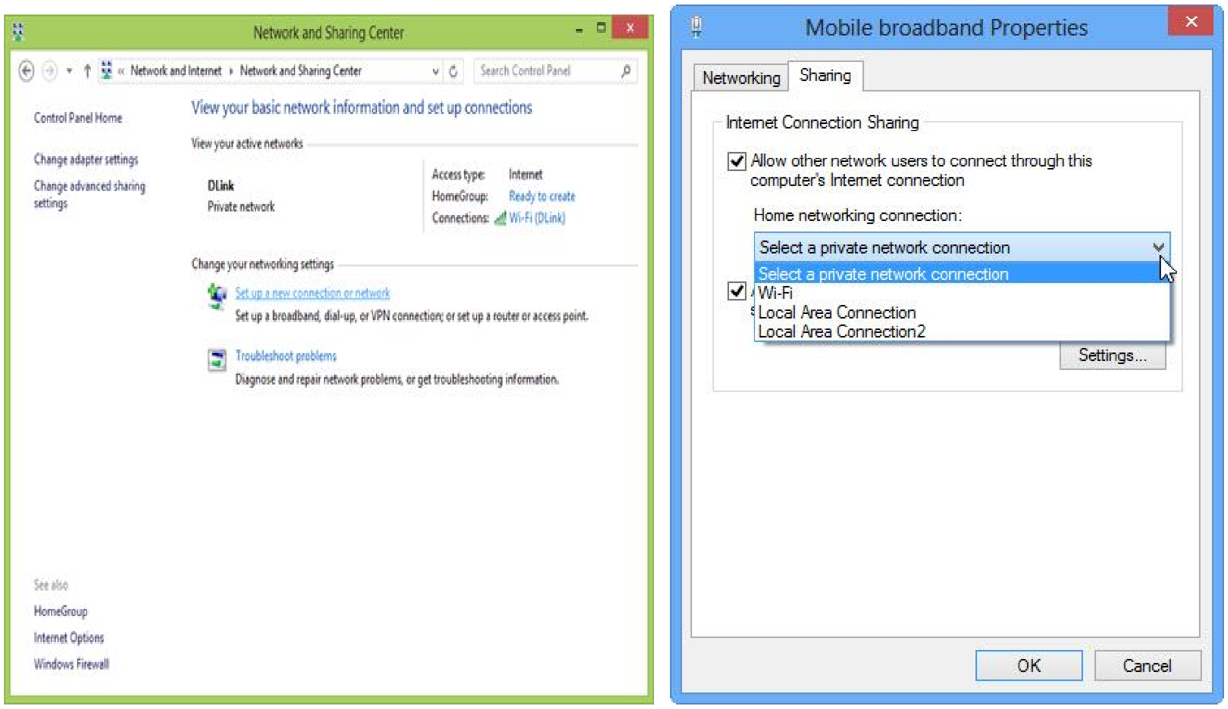
<u>**Remote Access of Raspberry Pi:**</u>

This following step is not required if an HDMI monitor is available during first boot. For security reasons, **SSH** is disabled by default in the newer versions of Raspbian. If a monitor is available then after a normal booting up of the Pi. It can be enabled by accessing the Raspberry Pi configuration page by typing "**sudo raspi-config**" in the raspberry pi terminal. Otherwise it can be enabled by modifying the Raspian Live image. After setting up the memory card with the Raspbian image, mount it in your laptop and open the boot partition.In the **boot partition** of the image, place an empty file called "**ssh**" without any extension. This will enable **SSH** in the Pi for remote login.

<u>**Network Sharing and IP address**</u>

The Pi can be accessed remotely if it is part of the same network as your laptop. This can be done in multiple ways. In Pi2 and older versions the Pi does not have onboard wifi hardware. However it can be added by using USB WiFi adapters. If connected to a WiFi network then the IP address can be looked up in the Router home page or directly from the Pi. In our setup we use an ethernet cable directly connecting the Pi to the laptop and sharing the WiFi of the laptop to the ethernet port.

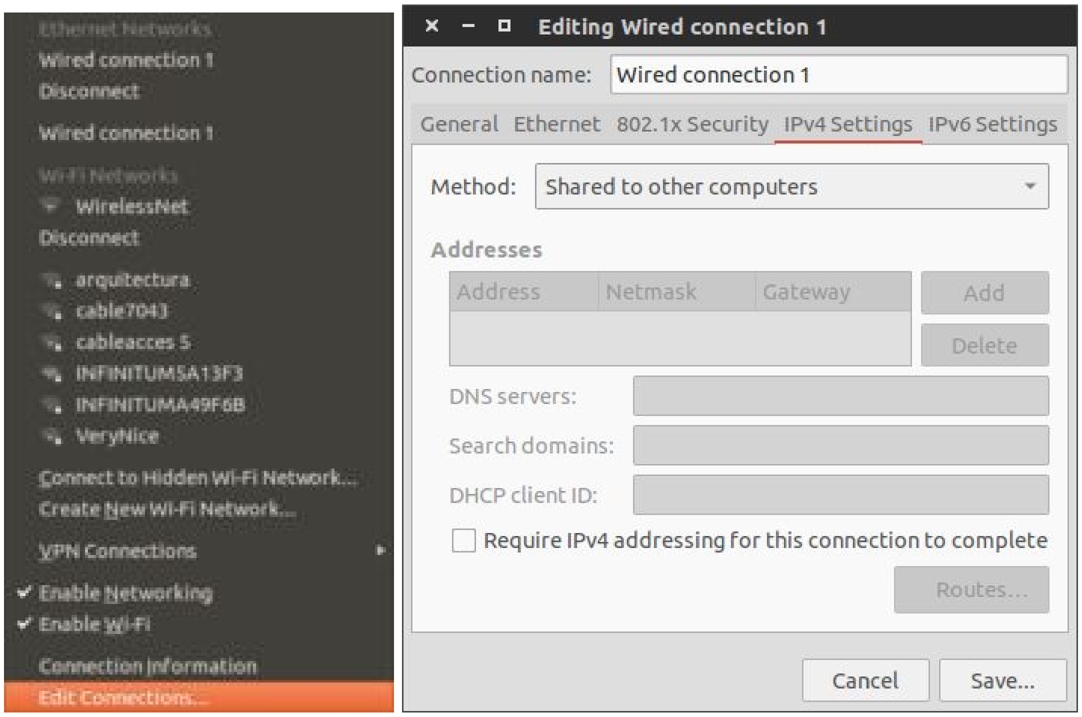
- **In windows** this can be done by opening Network and Sharing Centre after connecting the laptop to a WiFi router. Right clicking the WiFi details will open a dropdown menu. Clicking on **Properties option** will open a new window where on the top, a tab called "Sharing" will be available. In the tab, we can check the option to share the WiFi to the ethernet port. The laptop will then assign a valid IP address to the Pi through the ethernet port. The IP address can be found out by opening the command prompt and typing the following command "**arp -a**". The command will show two sets of IP addresses and the Pi can be recognised through the MAC address. All Raspberry Pi have a recognisable MAC address with a fixed format. It can be looked up on the internet.


Network and Sharing Center
Network and Internet › Network and Sharing Center
Search Control Panel
Control Panel Home
Change adapter settings
Change advanced sharing settings
View your basic network information and set up connections
View your active networks
DLink
Private network
Access type: Internet
HomeGroup: Ready to create
Connections: Wi-Fi (DLink)
Change your networking settings
Set up a new connection or network
Set up a broadband, dial-up, or VPN connection; or set up a router or access point.
Troubleshoot problems
Diagnose and repair network problems, or get troubleshooting information.
See also
HomeGroup
Internet Options
Windows Firewall
Mobile broadband Properties
Networking
Sharing
Internet Connection Sharing
Home networking connection:
Select a private network connection
Select a private network connection
Wi-Fi
Local Area Connection
Local Area Connection2
Settings...
OK
Cancel

[✓] Allow other network users to connect through this computer's Internet connection

Fig. WiFi Sharing Settings in Windows

- **In Ubuntu** the network can be shared by accessing the network drop down menu from the toolbar. When the Pi is connected through the ethernet port, select the **edit connections** and further select the **Wired Connection**. In the newly opened **Edit window** under the **IPv4 settings** tab, Select the **Method** as **Shared to other computers**.This will share the WiFi connection to the ethernet port. The IP address of the device connected to the ethernet port can be figured out by opening and the **Terminal** and typing the following command **"arp -a".** As in the previous case the command will show two sets of IP addresses and the Pi can be recognised through the MAC address. All Raspberry Pi have a recognisable MAC address with a fixed format. It can be looked up on the internet.



Editing Wired connection 1

Connection name: Wired connection 1

Method: Shared to other computers

DNS servers:

Search domains:

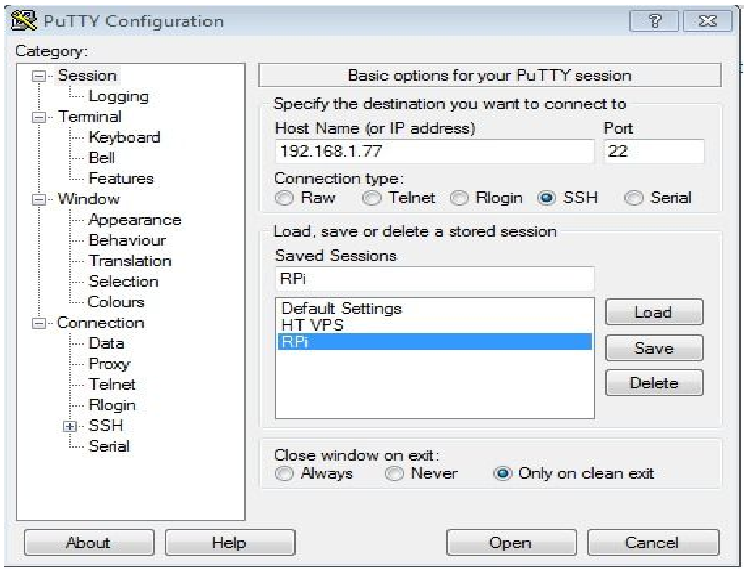
DHCP client ID:

☐ Require IPv4 addressing for this connection to complete

Fig. WiFi Sharing Settings in Ubuntu

SSH into the Pi:

In Windows a software called PuTTy can be downloaded at http://www.putty.org/ This is an executable that will allow us to SSH into any device from Windows.



PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 192.168.1.77 Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions: RPi

Default Settings
HT VPS
RPi

Close window on exit: Always Never [✓ Only on clean exit]

Fig. Putty on Windows

The IP address of the Pi should be entered into the field and then the clicking on **Open** will create a terminal running on the Pi. The default username is **“Pi”** and the default password is “**raspberry**”.

In Ubuntu the Terminal has a command to SSH into the Pi. The format of the command is: **ssh username@ipaddress**. So for example: **ssh pi@10.42.0.75** And then type in the default password if it has not been changed by anyone.

Note: The PuTTy terminal will close the connection if it remains idle for long. This will create issues during the installation of software packages in the Pi where the terminal will be busy for long durations. If the connection is terminated then all the processes will also be halted resulting in incomplete installations and other issues.

To work around the above mentioned issues it is possible by pushing the processes to the background or by dissociating them from the terminal. An easy way to do is by using an application called **“screen”**. A new window from the putty terminal can be opened by typing in **“screen bash”** . And when the terminal is held busy for a long time it can be disconnected by pressing **“Ctrl +A +D”**. If the PuTTy connection closes then the process will not be affected.
A list of such processes can be found by typing **screen -ls.** A “screen” can be reconnected to work further by typing **screen -r <name>**. No such issues occur in Ubuntu. But the same is still recommended.

Installing BladeRF, yate , yateBTS

The following steps are after the successful booting into a Raspberry Pi either using a monitor or by SSH-ing into the system. Reminder to share WiFi to the RPi if ethernet is used for SSH.

Installing BladeRF related libraries and tools:
First the bladeRF related libraries are to be installed in Rpi. It can be installed using the following commands in the RPi Terminal

```
sudo apt-get update
sudo apt-get install git apache2 php5 bladerf libbladerf-dev libbladerf0 automake
```

Then we can test if the required libraries to recognize and run commands related to bladeRF are present. Connect the bladeRF to the USB port (designed for 3.0 but RPi does not have a USB3.0 port) and open the bladeRF command line interface.
(Note: In later versions of the repository for installation in ubuntu etc. the bladeRF CLI is not included in the distribution)

```
sudo bladeRF-cli -i
```

In the CLI you can type the command “version” to find out details about the Firmware and FPGA version after type quit to exit bladeRF cli.

Prerequisite software required for the installation:

Now before we proceed further in the installation of Yate and yateBTS we need to install additional applications that will be needed to download and build the source code for yate and yateBTS

This is needed to checkout the yate and yateBTS source

```
sudo apt-get install subversion
```

This configuration script builder. After fetching Yate you will have to run autogen.sh to generate the configure file

```
sudo apt-get install autoconf
```

The following two commands are very important to establish proper connection to baldeRF while running yate and yateBTS.

```
sudo apt-get install libusb-1.0-0-dev
sudo apt-get install libgsm1-dev
```

Fetching yate and yateBTS using svn:

Yate is maintained by an active set of developers who updated the repository regularly to fix bugs and other issues. Earlier versions had issues with compatibility with bladeRF and there was a strong dependence on the version number of yate and the FPGA version on bladeRF. This created difficulties and allowed only certain versions to work properly. However recently yate provides tested bladeRF FPGA images and hence the version numbers should not be of issue.

To continue with the installation fetch the yate source from the link http://voip.null.ro/svn/yate/trunk

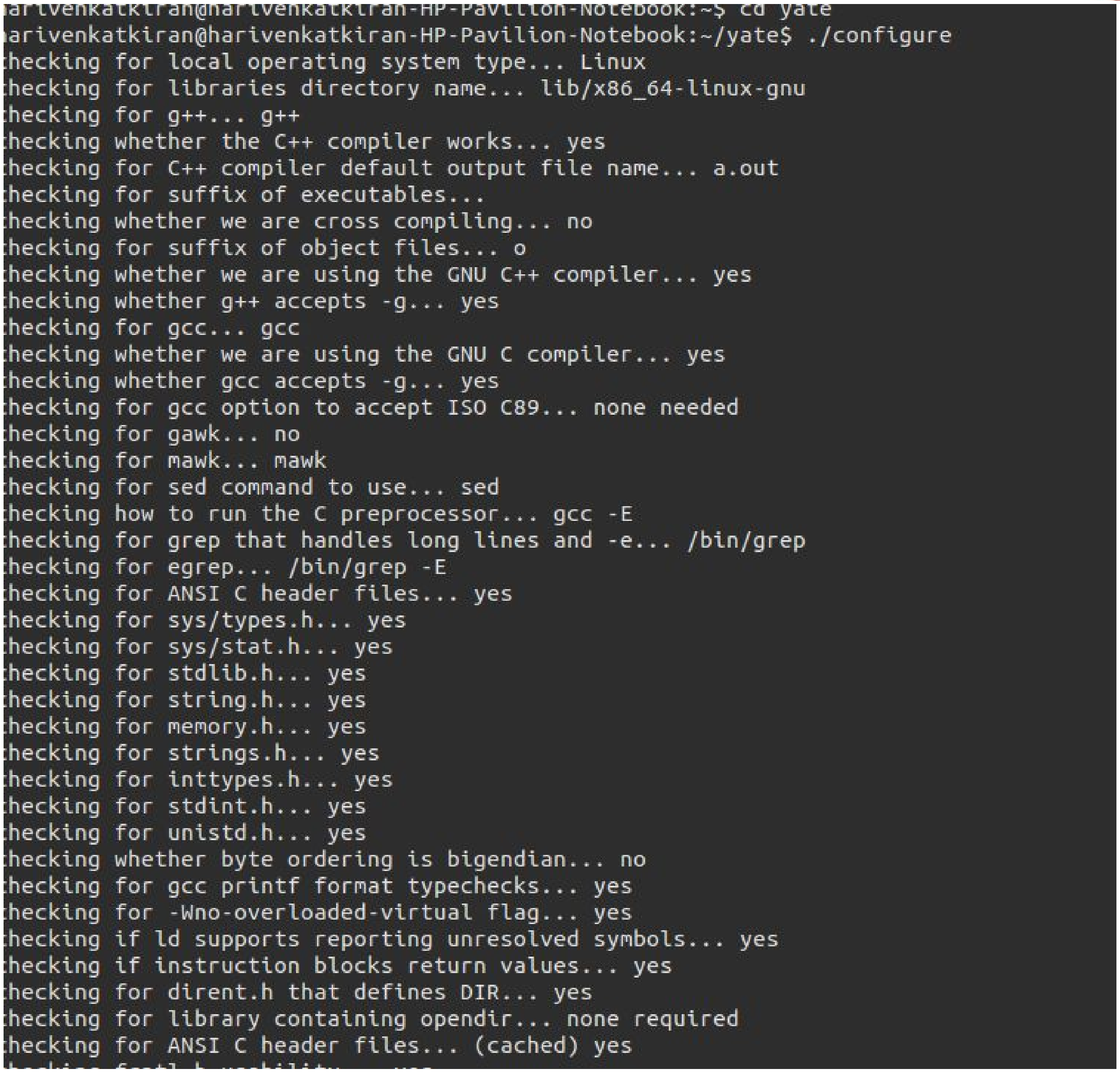
```
svn checkout http://voip.null.ro/svn/yate/trunk yate-SVN

cd yate-SVN
svn up
./autogen.sh
```

```
./configure --prefix=/usr/local
make -j4
sudo make install-noapi
sudo ldconfig
 cd ..
```

The above step will take 10-15 minutes depending on the RPi version. In the yate-SVN folder we can check the version of the repository and a lot more details by typing **“svn info”**. The repository version can be controlled by using standard svn commands like
svn up -r <version number>

The command above work by doing the following: **autogen.sh** will generate the **configure script** which in turn checks the dependencies. This is very useful in debugging the installation process in case the setup doesn’t work. The most important two are **libgsm1 and libusb** which are included as part of the prerequisites.

Note: *You should look at the configure output and check that all features you need are detected. If they are not you should first check if you are not missing the corresponding development packages. If there are errors in ./configure step then you may need to install additional packages indicated from the error messages in the terminal. This is the best way to troubleshoot and also the most common cause of issues because necessary dependencies may not have been present.*
The image below captures the ./configure command and next to the dependencies we can check if the necessary ones are present. If it doesn’t throw up any errors then checking for the two main ones mentioned above should be sufficient.



```
arivenkatkiran@harivenkatkiran-HP-Pavilion-Notebook:~$ cd yate
arivenkatkiran@harivenkatkiran-HP-Pavilion-Notebook:~/yate$ ./configure
hecking for local operating system type... Linux
hecking for libraries directory name... lib/x86_64-linux-gnu
hecking for g++... g++
hecking whether the C++ compiler works... yes
hecking for C++ compiler default output file name... a.out
hecking for suffix of executables...
hecking whether we are cross compiling... no
hecking for suffix of object files... o
hecking whether we are using the GNU C++ compiler... yes
hecking whether g++ accepts -g... yes
hecking for gcc... gcc
hecking whether we are using the GNU C compiler... yes
hecking whether gcc accepts -g... yes
hecking for gcc option to accept ISO C89... none needed
hecking for gawk... no
hecking for mawk... mawk
hecking for sed command to use... sed
hecking how to run the C preprocessor... gcc -E
hecking for grep that handles long lines and -e... /bin/grep
hecking for egrep... /bin/grep -E
hecking for ANSI C header files... yes
hecking for sys/types.h... yes
hecking for sys/stat.h... yes
hecking for stdlib.h... yes
hecking for string.h... yes
hecking for memory.h... yes
hecking for strings.h... yes
hecking for inttypes.h... yes
hecking for stdint.h... yes
hecking for unistd.h... yes
hecking whether byte ordering is bigendian... no
hecking for gcc printf format typechecks... yes
hecking for -Wno-overloaded-virtual flag... yes
hecking if ld supports reporting unresolved symbols... yes
hecking if instruction blocks return values... yes
hecking for dirent.h that defines DIR... yes
hecking for library containing opendir... none required
hecking for ANSI C header files... (cached) yes
```

Fig. Terminal while running the ./configure command

The command make install-noapi is required when Doxygen or Kdoc are not available. They are useful for documentation purposes and are not necessary for installation

Now fetch the yateBTS source from the link http://voip.null.ro/svn/yatebts/trunk

svn checkout http://voip.null.ro/svn/yatebts/trunk yatebts-SVN

cd yatebts-SVN
svn up (***required for downloading latest version***)
./autogen.sh

```
./configure --prefix=/usr/local
make -j4
sudo make install
sudo ldconfig
cd ..
```

The above step will also take 10-15 minutes depending on the RPi version.

Next, we will symlink the NIB web UI into our apache www folder:

```
cd /var/www/html/
sudo ln -s /usr/local/share/yate/nib_web nib
```

And provide “write” permission to the configuration files:

```
 sudo chmod -R a+w /usr/local/etc/yate
```

You can now access your BTS web User Interface from the browser in your laptop by typing the following URL in the search bar. This is a webpage hosted in the Pi and was why apache was downloaded as one of the prerequisite software.

```
http://ip-of-your-rpi/nib
```

Configuration

- Set the following values in the NIB web interface. For testing purposes. Note : We are using the GSM-R (Railways band) to reduce interference with existing commercial carriers. The GSM bands are all licensed and hence the experiment should ideally be conducted in a Faraday cage.
- Open the **/usr/local/etc/yate/ybts.conf** file either with nano or vi

```
sudo nano  /usr/local/etc/yate/ybts.conf
```

- And update the following values:(See Screenshot below for values)

```
Radio.Band=900
```

Radio.C0=91

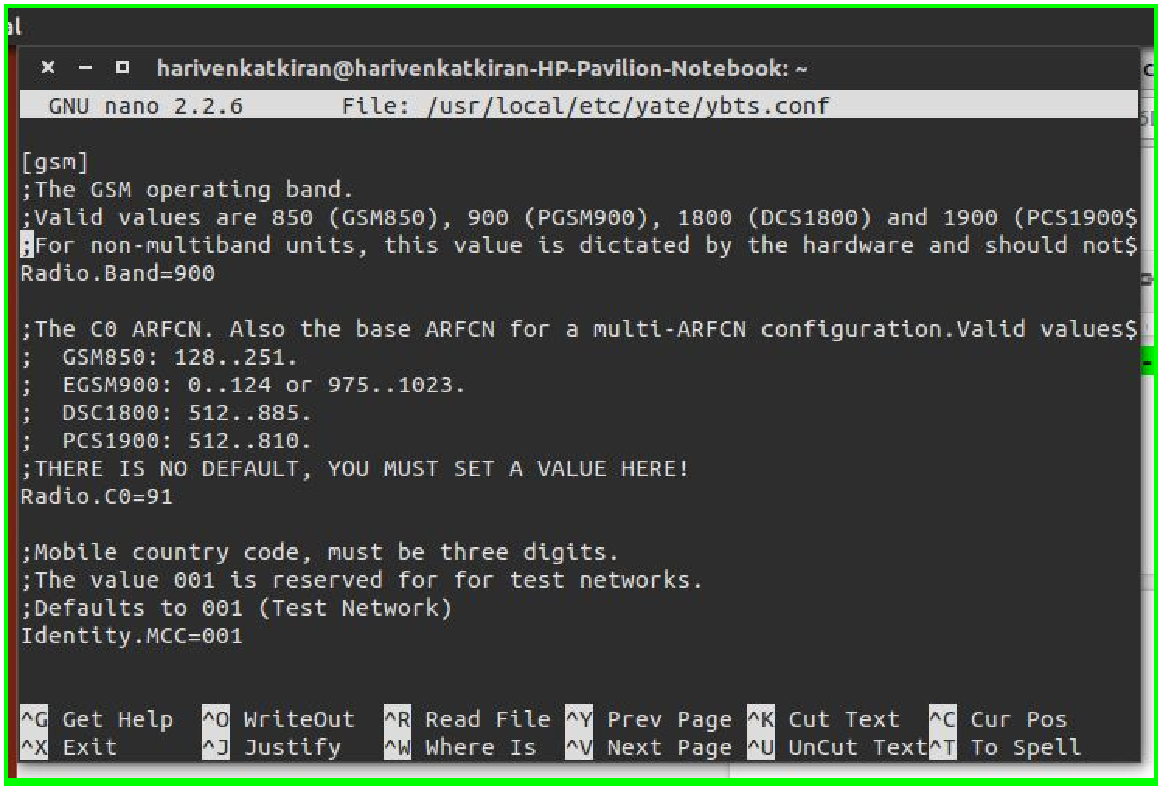
Identity.MCC=YOUR_COUNTRY_MCC (001)

Identity.MNC=YOUR_OPERATOR_MNC (01)

Identity.ShortName=yateBTS

Radio.PowerManager.MaxAttenDB=10

Radio.PowerManager.MinAttenDB=10



harivenkatkiran@harivenkatkiran-HP-Pavilion-Notebook: ~

```
  GNU nano 2.2.6          File: /usr/local/etc/yate/ybts.conf

[gsm]
;The GSM operating band.
;Valid values are 850 (GSM850), 900 (PGSM900), 1800 (DCS1800) and 1900 (PCS1900$
;For non-multiband units, this value is dictated by the hardware and should not$
Radio.Band=900

;The C0 ARFCN. Also the base ARFCN for a multi-ARFCN configuration.Valid values$
;   GSM850: 128..251.
;   EGSM900: 0..124 or 975..1023.
;   DSC1800: 512..885.
;   PCS1900: 512..810.
;THERE IS NO DEFAULT, YOU MUST SET A VALUE HERE!
Radio.C0=91

;Mobile country code, must be three digits.
;The value 001 is reserved for for test networks.
;Defaults to 001 (Test Network)
Identity.MCC=001

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

- Now, edit the **/usr/local/etc/yate/subscribers.conf**: To do so the terminal based text editor "nano" can be used. The following command can be used to access the file.

  sudo nano /usr/local/etc/yate/subscribers.conf

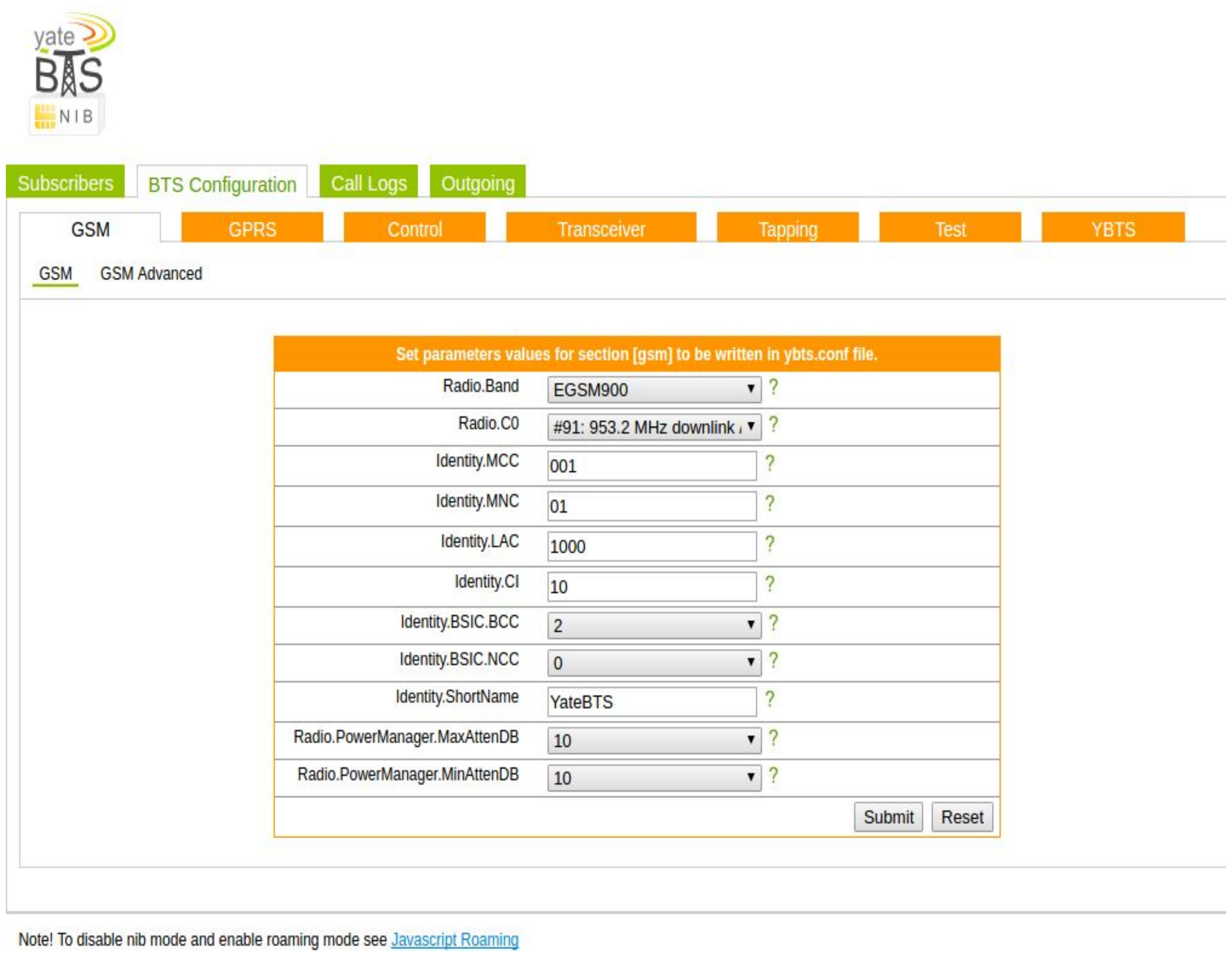
- Now change to following entries in the file to the ones specified below

  country_code=YOUR_CONTRY_CODE

  regexp=.*

- The commands to save the file will be displayed in the bottom in the terminal.

WARNING Using the **.*** regular expression will make() **EVERY** GSM phone in your area connect to your BTS.Please set the final values as seen here



Set parameters values for section [gsm] to be written in ybts.conf file.

Radio.Band	EGSM900
Radio.C0	#91: 953.2 MHz downlink
Identity.MCC	001
Identity.MNC	01
Identity.LAC	1000
Identity.CI	10
Identity.BSIC.BCC	2
Identity.BSIC.NCC	0
Identity.ShortName	YateBTS
Radio.PowerManager.MaxAttenDB	10
Radio.PowerManager.MinAttenDB	10

Note! To disable nib mode and enable roaming mode see Javascript Roaming

Fig. Network in a Box (NIB) Settings

The Options for the MCC and MNC are the generally used ones for testing purposes.
The short name only works for a paid version of the software and will generally default to the name “Test”.
The Radio Power manager Manager sets the power output by the transceiver and is actually the attenuated range of operation . It means that the actual transmit power is from -Min dB to -Max dB attenuated with respect to the maximum possible transmit power.
The call logs option can be switched on to help examine the durations of the calls and the frequency of successful ones over dropped ones. Yate has a non queueing system for non-emergency calls meaning calls will be dropped if the network is busy. This can be used to

figure out problems with the robustness of the system.

The option LAC is “**local Area Code**” and is assigned arbitrarily. Incase of multiple BTS units then the LAC has to be unique to each of them. The remaining parameters are also useful in case of multiple cells, and multiple BTS units.

- Finally, we can start the new BTS by executing the command (with the BladeRF plugged in)

  ```
  sudo yate -s
  ```

- If everything was configured correctly, the following messages will be displayed,

  ```
  Starting MBTS...
  Yate engine is initialized and starting up on raspberrypi
  RTNETLINK answers: File exists
  MBTS ready
  ```

At this point, the middle LED for the bladeRF should start blinking. This indicates that the FPGA is properly loaded and that the system is up and running. In case there are issues present, then running the command **sudo yate -vvv CDa -s** will provide verbose output messages that are color coded.

Connecting to the Network:

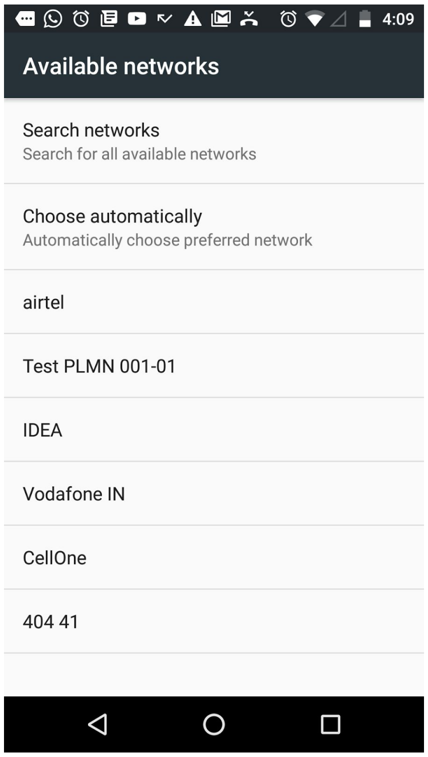
Now if the above settings displayed in the NIB web UI are used then the network setup will have the name **TEST -001 - 01**. Any phone can be connected to the network by accessing “**Settings**”. Under that select the option related to **Cellular networks** or other similarly named options. And under that search for network operators. Don’t forget to disable the “automatic” option. Also it would help to set the preference to **2G networks**. The network setup from our BTS will be a 2.5G network. The system works with **dual SIM phones** too.

Note: It has been claimed in another study that the phone softwares are designed to connect to the faster networks in case of resets, which are the 3G and 4G networks .This may affect the phone’s connection to our BTS but it hasn’t been observed conclusively. This plays a more important role in network spoofing and not related to the goal of our setup.

Under the available networks select the Test Network broadcast from our BTS. Since it is not the home network of the SIM card being used the phone will consider it to be under **roaming conditions**.

The signal strength, RSSI and other related parameters can be measured using an application called Network Cell Info lite available in Android play store. This will give us an approximate indication of the strength of the signal (doesn’t update fast enough to test the range. Needs to be reopened for more accuracy) and will help in rough estimation of the range.

The Phone number is also assigned by yate through a message sent while connecting to the network for the first time. It is a 9 digit number where the first 2 digits are the country code specified in the settings. While the number hasn’t changed generally, it has been observed that the number assigned had changed abruptly once without any indication.


4:09

Available networks

Search networks
Search for all available networks

Choose automatically
Automatically choose preferred network

airtel

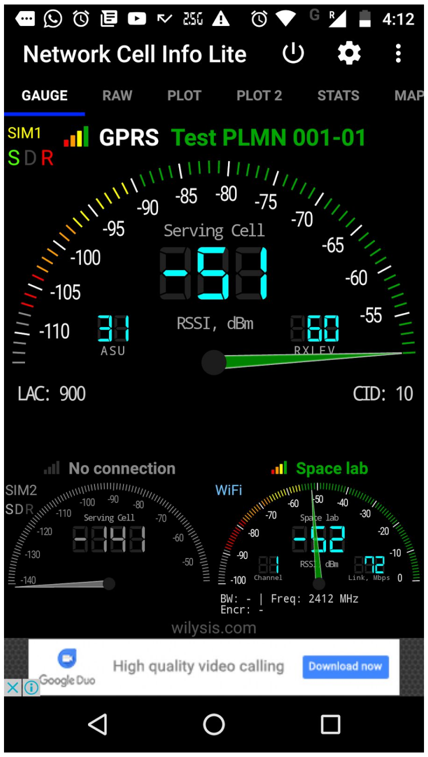
Test PLMN 001-01

IDEA

Vodafone IN

CellOne

404 41


4:12

Network Cell Info Lite

GAUGE RAW PLOT PLOT 2 STATS MAP

SIM1 GPRS Test PLMN 001-01

Serving Cell -51 RSSI, dBm

31 ASU

60 RXLEV

LAC: 900 CID: 10

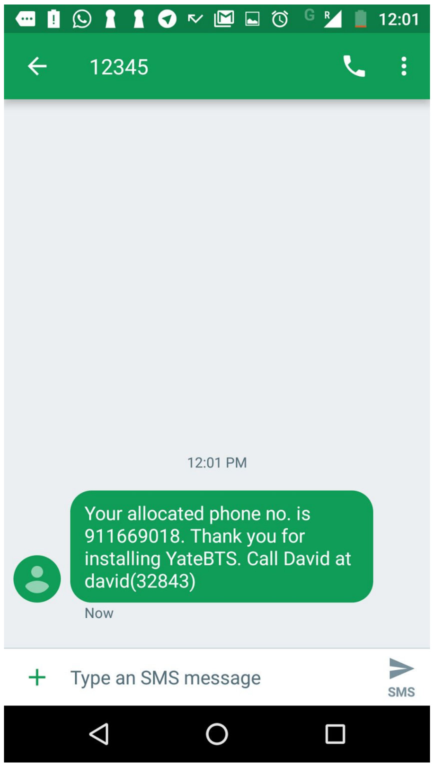
SIM2 No connection

WiFi Space lab -62 RSSI, dBm

BW: - | Freq: 2412 MHz
Encr: -

wilysis.com

High quality video calling Download now


12:01

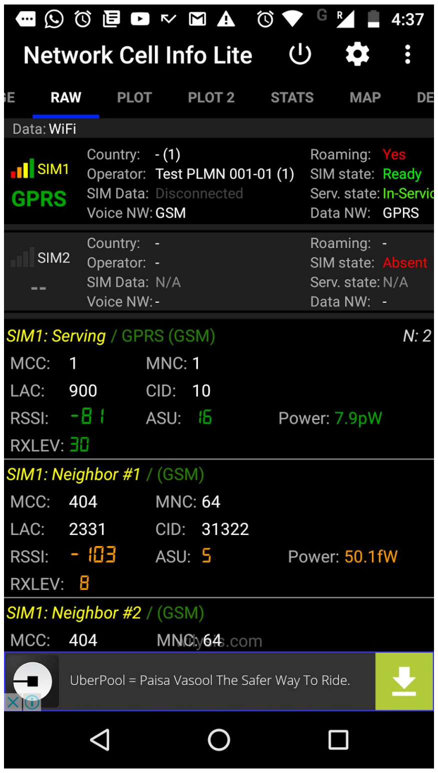
12345

12:01 PM

Your allocated phone no. is 911669018. Thank you for installing YateBTS. Call David at david(32843)

Now

Type an SMS message


4:37

Network Cell Info Lite

RAW PLOT PLOT 2 STATS MAP

Data: WiFi

SIM1 GPRS
Country: - (1)
Operator: Test PLMN 001-01 (1)
SIM Data: Disconnected
Voice NW: GSM
Roaming: Yes
SIM state: Ready
Serv. state: In-Servic
Data NW: GPRS

SIM2 --
Country: -
Operator: -
SIM Data: N/A
Voice NW: -
Roaming: -
SIM state: Absent
Serv. state: N/A
Data NW: -

SIM1: Serving / GPRS (GSM) N: 2
MCC: 1 MNC: 1
LAC: 900 CID: 10
RSSI: -81 ASU: 16 Power: 7.9pW
RXLEV: 30

SIM1: Neighbor #1 / (GSM)
MCC: 404 MNC: 64
LAC: 2331 CID: 31322
RSSI: -103 ASU: 5 Power: 50.1fW
RXLEV: 8

SIM1: Neighbor #2 / (GSM)
MCC: 404 MNC: 64

UberPool = Paisa Vasool The Safer Way To Ride.

Fig.Settings to connect to Base Station. And monitoring on Cell Info App

The above images show the setup on the phone for connecting to the BTS and also checking its signal strength and other related parameters. It is to be noted that SMS functionality is also available.

<u>INSTALLING ASTERISK SERVER FOR VoIP :</u>

Note: Will require a router to allow for SIP clients .

Yate also allows for outbound VoIP setup. To test the same an asterisk server can be setup on another system including another Pi or even on the same one. Presently the more stable of installations of asterix is through a pre-installed image of Raspbian called RasPBX.

The RasPBX OS can be downloaded from the following page: http://www.raspberry-asterisk.org/downloads/ And installed following the process mentioned earlier using etcher directly without any need for modifying or addition of any file after mounting it. The login credentials have been updated during their pre-installation into:
User name: root
Password: raspberry

Upon sshing into the RasPBX system type the following commands to update the system

raspbx-upgrade

Accept the upgrade prompt and the step may take a few minutes to complete. Follow this by configuring the timezone for the Pi.

configure-timezone

Note: the above step will work in ubuntu but may have issues in PuTTy depending on opening the graphical window for configuring.

Now to configure the timezone for Asterisk/FreePBX by manually editing through the console. First view the contents of the directory ***/usr/share/zoneinfo*** and find your timezone.

ls /usr/share/zoneinfo

ls /usr/share/zoneinfo/Asia

And then copy your timezone over to /etc/localtime .This has to be done by properly selecting the appropriate timezone and replace the command with the appropriate country/timezone.

```
cp    /usr/share/zoneinfo/Asia/Kolkata   /etc/localtime
```

Then reboot the system by typing **reboot** in the terminal

Expanding the file system. This needs to be done to allow the OS to make use of the full size of the memory card. It is done by first viewing the size of the partition called /dev/root by using the command **df -h** If the root partition shown is around 4GB and the memory card is for a much larger size then it can expanded.

Then by running the **raspi-config** command and choosing the **Advanced options** to expand the filesystem. Then upon rebooting the root partition will fill to the size of the memory card. Note: Again the raspi-config command will open a graphical window and that may cause issues in PuTTy.

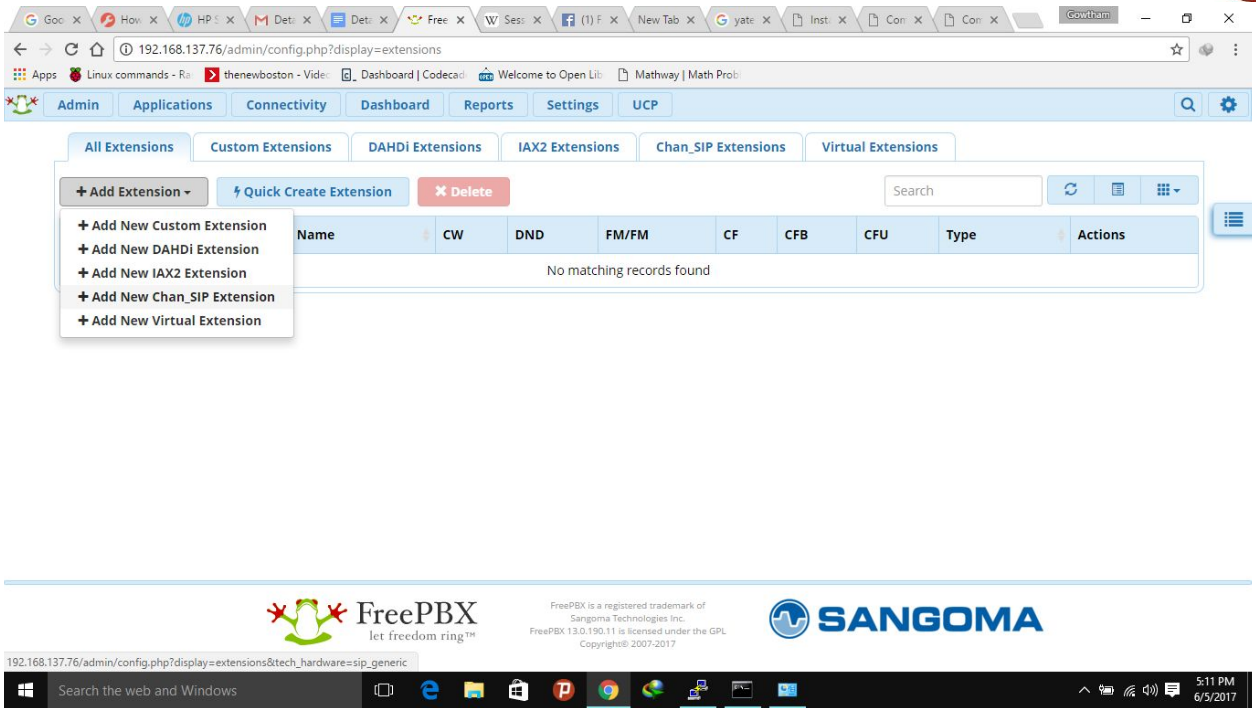
Asterix/ FreePBX Configuration:

Similar to the NIB interface , FreePBX is a graphical UI that can be accessed through a browser since it is hosted in the RasPBX system. This will be the home page of the server and hence can be accessed by just typing in the IP address of the RasPBX system in the browser of your laptop.

Click on the icon titled "**FreePBX administration**" and the login and password for it is "**admin**" both times. Upon logging into the FreePBX we can add new SIP extensions. Under the menu "**Applications**" click on **Extensions**. And then click on **Add Extension > Add New Chan_SIP Extension**.

In order to register a SIP Client, we need to create a SIP extension. Edit the fields under New Chan_SIP Extension appropriately: Given below is an example:

User Extension: This will be the number assigned to the SIP Client and can be any number. For our example we can start with 2001.
Display name: This field is for the name to be assigned to the SIP extension.
Secret: This is the password to be used to authenticate/register a new SIP client to this extension.

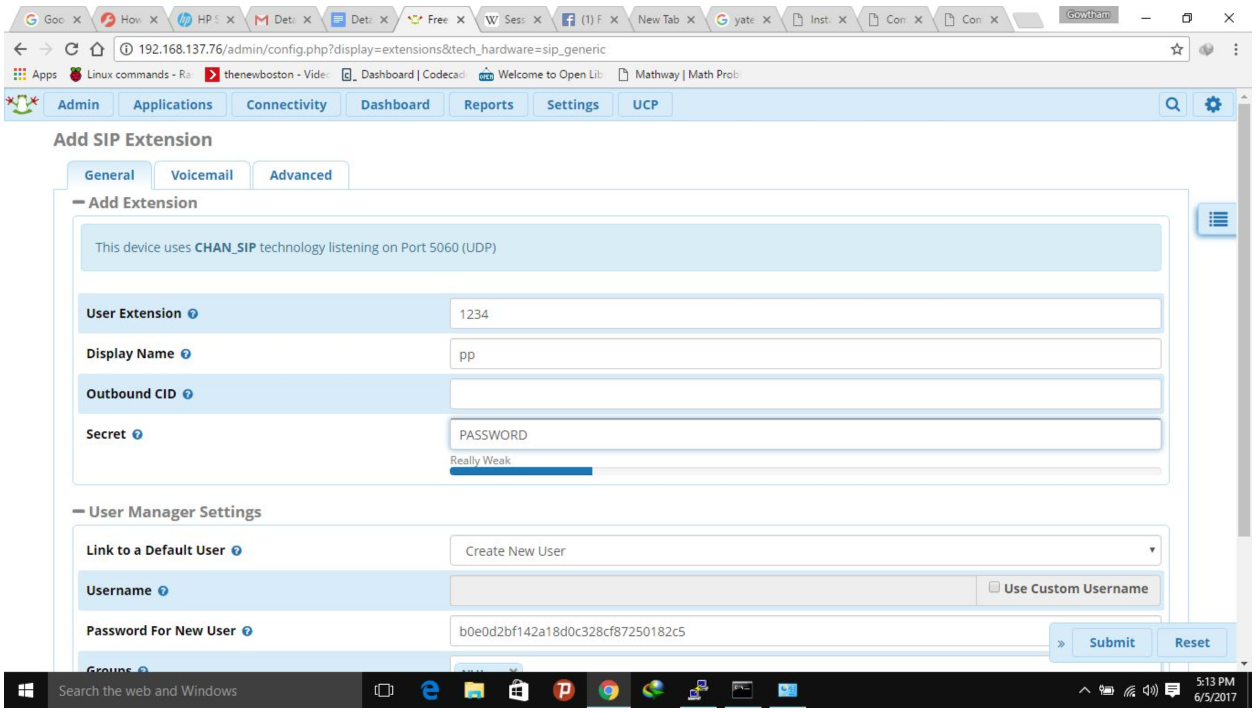


192.168.137.76/admin/config.php?display=extensions

Name	CW	DND	FM/FM	CF	CFB	CFU	Type	Actions
No matching records found								

FreePBX is a registered trademark of Sangoma Technologies Inc. FreePBX 13.0.190.11 is licensed under the GPL Copyright© 2007-2017

192.168.137.76/admin/config.php?display=extensions&tech_hardware=sip_generic



192.168.137.76/admin/config.php?display=extensions&tech_hardware=sip_generic

Add SIP Extension

– Add Extension

This device uses CHAN_SIP technology listening on Port 5060 (UDP)

User Extension	1234
Display Name	pp
Outbound CID	
Secret	PASSWORD (Really Weak)

– User Manager Settings

Link to a Default User	Create New User
Username	☐ Use Custom Username
Password For New User	b0e0d2bf142a18d0c328cf87250182c5
Groups	

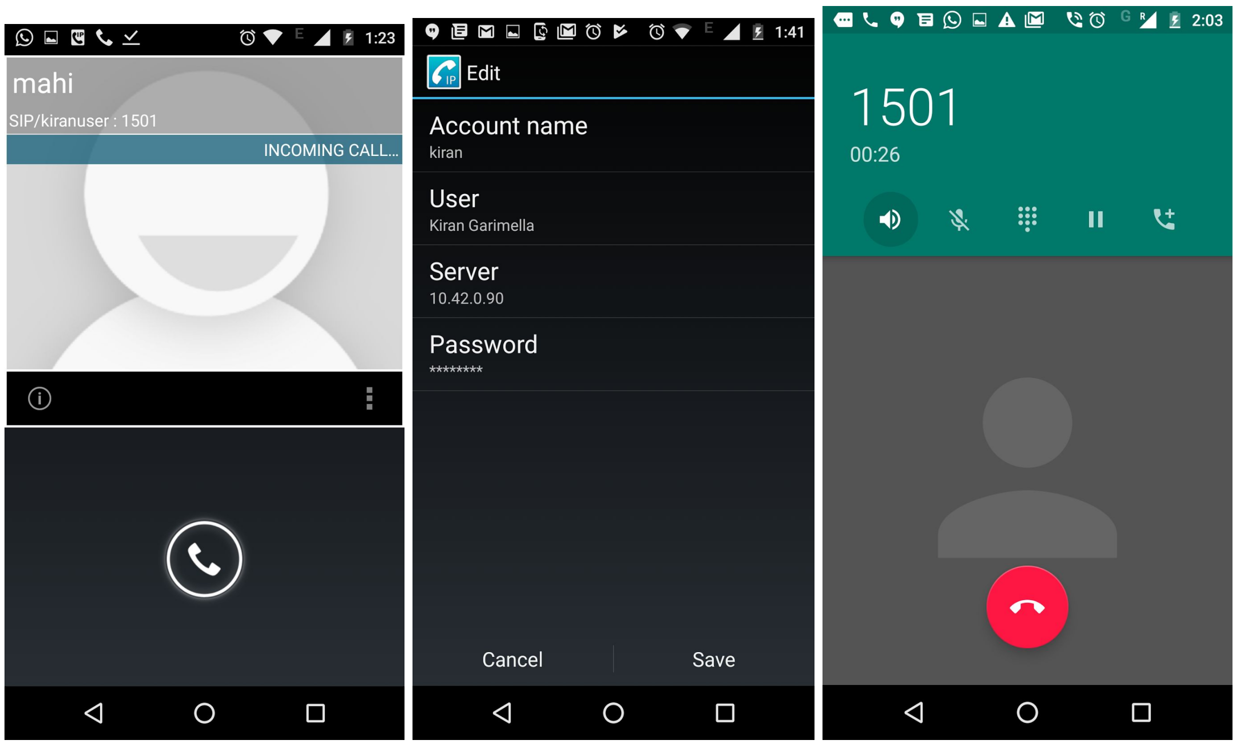
Fig. FreePBX Settings to Create New Extension

Note: When all the necessary fields have been entered, Click on **Submit** and also follow it by clicking on **Apply-Config** located on the top right of the screen.

The process can be repeated to set up another extension to allow two users to connect and test the system.

Registering a SIP Client:

The SIP client has to be part of the same network as the Pi. The SIP Client can be registered on a computer or a phone. In android systems there are quite a few popular apps for SIP clients and we use one called **CSipSimple**.


mahi
SIP/kiranuser : 1501
INCOMING CALL...

Account name
kiran
User
Kiran Garimella
Server
10.42.0.90
Password
****

1501
00:26

Fig. CSipSimple Application Settings and GSM to VoIP Inbound Call

While registering the SIP client the following information will be needed
IP Address of the Pi, UserName/AccountName, Password to authenticate the SIP Client to use the particular extension.

In the CSipSimple app, click on the **Add Account > Basic**

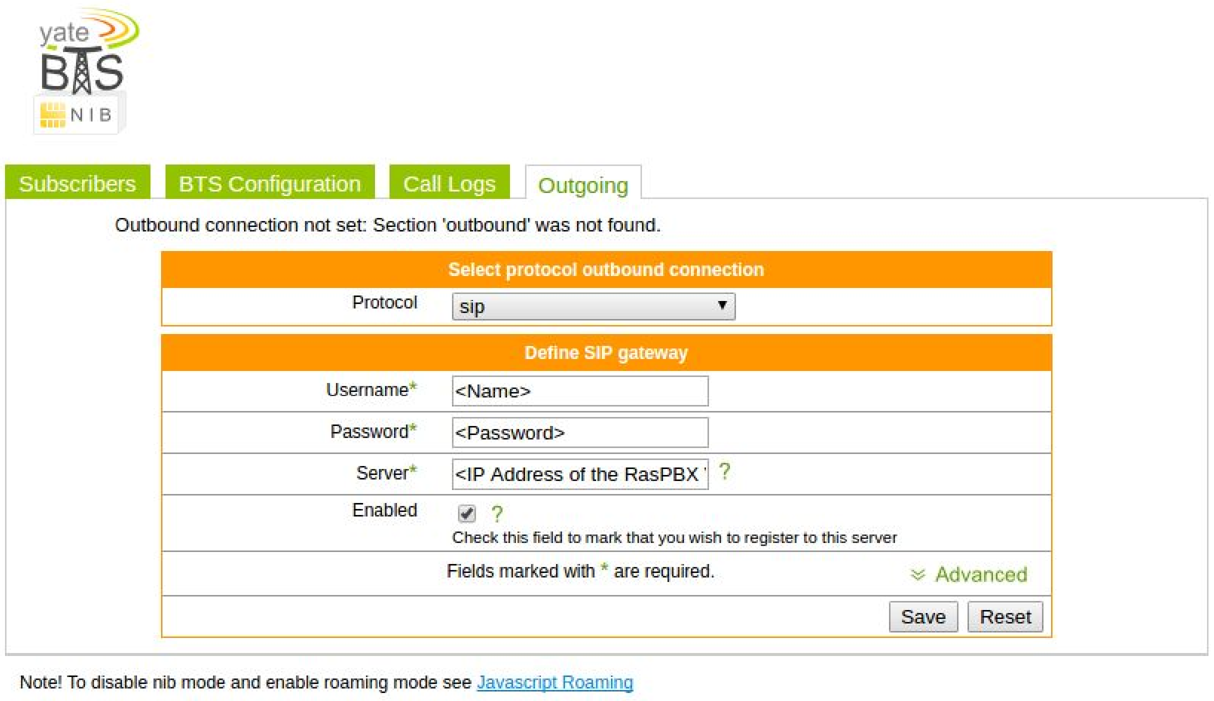
Account Name : <Name to be given>
User: <any extension number created in FreePBX> example: 2001
Server: <IP Address of the Pi> example: 10.42.0.90

Password: <Password set in the FreePBX for that particular extension>

Now the Client will be registered to RaspBX VoIP Server.

VoIP Outbound Calls from YateBTS to SIP Server:

YateBTS can also be registered as a SIP client and the GSM handsets on the Yate side can call other SIP clients outside the GSM network through this outbound option. The option to register YateBTS as a SIP client can be found in NIB of the yateBTS Raspberry Pi.The user name will be the extension number to be associated with the BTS and the password will be the one used for the extension setup in FreePBX. The server address will be the IP Address of the RasPBX VoIP Server. If yateBTS is installed in the same Pi as Asterisk then the loopback address 127.0.0.1 can be used instead.



Outbound connection not set: Section 'outbound' was not found.

Select protocol outbound connection	
Protocol	sip

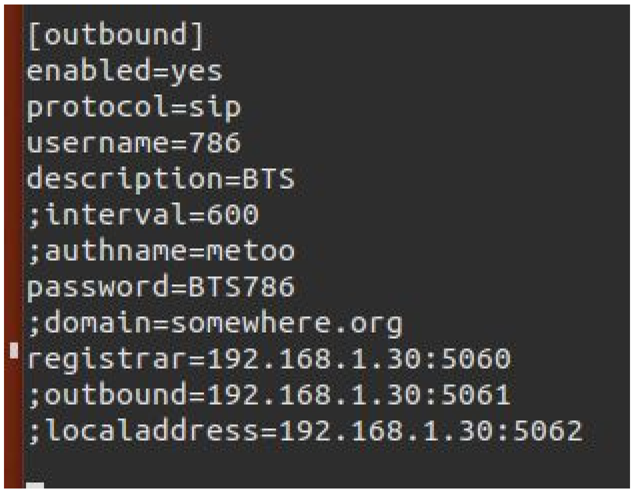
Define SIP gateway	
Username*	<Name>
Password*	<Password>
Server*	<IP Address of the RasPBX ' ?
Enabled	✓ ? Check this field to mark that you wish to register to this server

Fields marked with * are required.

Note! To disable nib mode and enable roaming mode see Javascript Roaming

Fig. NIB Settings for Outgoing VoIP

If NIB is not accessible then the outbound settings can be configured in the file **/usr/local/etc/yate/accfile.conf** .The file entry should be according to the image below.



```
[outbound]
enabled=yes
protocol=sip
username=786
description=BTS
;interval=600
;authname=metoo
password=BTS786
;domain=somewhere.org
registrar=192.168.1.30:5060
;outbound=192.168.1.30:5061
;localaddress=192.168.1.30:5062
```

Fig. Contents of accfile.conf

The section should be renamed to **outbound**. And the line should be changed to **enabled=yes**. The field **registrar** should contain the IP Address of the VoIP Server and should be uncommented. The other fields need not be disturbed.

Essentially the YateBTS will search for the dialed number in the GSM registry and upon not finding it, Yate will then connect on the outbound line through the internet (of the yateBTS Raspberry Pi) to the RasPBX VoIP server and then the VoIP server will place the call.

Presently this allows for calls to go one way from the GSM network to the outside. However the reverse is not possible through the existing setup because the current setup has the BTS itself registered as a client which abstracts all the GSM phones on its side of the network from the VoIP server. Additionally multiple simultaneous phone calls can be placed to the VoIP network from behind the BTS and this was tested by placing two simultaneous calls between four phones.

VoIP Server can also be connected to the outside world through a trunk by purchasing a license/lease from commercial SIP service providers.

POWER REQUIREMENT OF THE BTS

The power drawn is measured by a couple of USB Power meters which are generally used to measure the strength of power sources. Their accuracy is not exactly known, however they do provide a reasonable estimate of the power consumed at the source and also by the BladeRF. The power is measured at the source for both the Raspberry Pi Model 2B and Model 3B as well as at the USB port powering the BladeRF. As for the laptop instance, the power is only measured at the BladeRF USB port. The power delivered to BladeRF's Transceivers can be controlled through the software and YateBTS allows the settings to attenuate the TRX power with respect to the D/A converters. The settings can be changed either using the NIB_WEB tool or by changing the values in the file **/usr/local/etc/yate/ybts.conf**.

Power Measurement:

Radio Power Attenuation (dBm)	Source (W)		BladeRF (W)		
	Pi Model 2B	Pi Model 3B	Pi Model 2B	Pi Model 3B	HP Laptop
0	**6.00**	**6.10**	**3.37**	**3.38**	**3.56**
10	**5.51**	**5.50**	**2.93**	**2.94**	**2.96**
20	**5.36**	**5.4**	**2.91**	**2.90**	**2.86**
35	**5.16**	**5.4**	**2.86**	**2.86**	**2.77**

Note: For higher values of attenuation the change is current drawn will be much smaller and hence the power meters may not be as accurate as desired at these higher attenuation values.

As can be inferred here there is no significant change in power consumed and supplied by the two Raspberry Pi models and as such provide the same effect. However while conducting the tests it had been noticed that when the attenuation is kept zero at the TRX, the BladeRF is reset multiples times by the software and this could be due to insufficient power supply from the Raspberry Pi. It can also be noticed from the table that the power supplied by the laptop is higher than the other two at zero TRX attenuation. Since the power supplied under the remaining attenuation conditions are very similar, this reinforces the inference about insufficient power supply at zero TRX attenuation.

RANGE OF THE BTS :

The range of any BTS is dependent on a multitude of factors which include transmit power, frequency used, height of the antenna over surrounding terrain, antenna characteristics and gains, density of the obstacles in the region eg: suburban, cities, etc. A few of these parameters are within control of the BTS system and can be tuned for optimum range. The range is tested here by measuring the Received Signal Strength Index (RSSI) on the phone. The app used is Network Cell Info Lite. For a phone to be able to communicate over GSM the RSSI should be above -110dBm below which the phone will drop the connection with the network. For our BTS system running on a HP laptop with normal antennas found in WiFi routers and configured at zero attenuation from the maximum for TRX power, the RSSI measurements were carried out on the roof of the Electrical Sciences Building(ESB) IITM. The BTS was placed on a table on top of the water tank which was the highest point on the terrace. The phones remained connected to the network all through the terrace of ESB and occasionally disconnected when behind the array of solar panels. However at the farthest boundaries of the ESB the phones were disconnected from the network.

There are a couple of ways to increase the coverage of the network.

Antenna :
The antenna plays an important role in the gain introduced to the system. Directional antennas increase the gain of a system along one direction, whereas omni directional antennas radiate uniform power in all the directions. The antennas can also be placed at a much higher location than the base station and can further increase the range of the system.

Power Amplifier :
A power amplifier can significantly increase the transmit power of the BTS and can increase the gain of the system which in turn will increase the distance too. Typically a 6dBm increase in transmit power should see a doubling of the range.

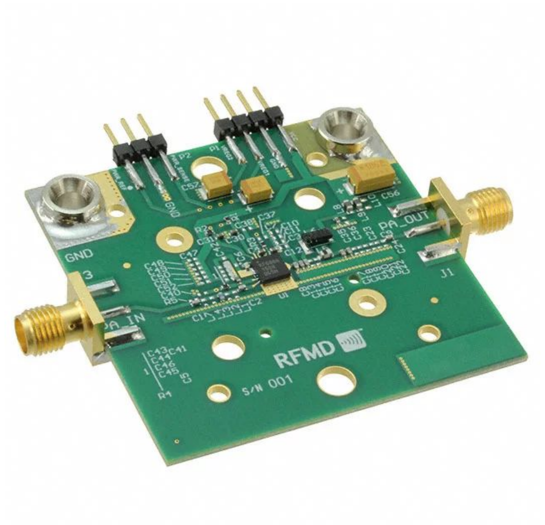


Fig. Power Amplifier RFMD6886

In our BTS, the power amplifier RFMD6886 had been used to increase the transmit power. It had been tuned to add a gain of 20dBm to the BTS. The power amplifier has an output saturation power of 36.5dBm. The BTS should be configured to input within the limit, taking into consideration its tuned gain, to ensure saturation does not occur in the power amplifier.

CONCLUSION AND FUTURE SCOPE

The project has been able successfully to set up a private GSM network enabled with wifi for VoIP communication. The system is portable and very easy to deploy within a short time. The current hardware supports one 200KHz channel and hence 7 calls can be placed simultaneously. At present the system supports two way GSM phone calls, SMS functionality and GSM to SIP calls. With an appropriately tuned Power amplifier the TRX of the bladeRF can be run at attenuation values that are stable with the Raspberry Pi models, thus ensuring the portability of the system. The success rates for the calls placed are not uniform over time and the reason behind the dropping of calls or network congestion have not been exactly established. As for possible improvements to the system, a much more powerful omnidirectional antenna can easily double the coverage of the system. Also if the linux server could supply more power then it can itself be configured to run as a wireless access point thereby removing the need for a router for the VoIP communication. The Raspberry Pi model 3B does have a wireless adapter that can be configured as an access point. However adding another antenna for the wifi will seriously strain the Pi when it is already powering the BladeRF. Other single board computers whose specifications are more powerful than the Pi model 3B can be tested. An aspect that had not been properly investigated is the LTE backhaul from the BTS to connect it to the rest of the world's GSM network. While the best way to introduce this is via dedicated hardware for the LTE backhaul, alternative plausible solutions are also present which include using voice enabled 4G USB dongles. The dongles communicate with VoIP servers and outbound calls can be made from the BTS. The effectiveness of this alternative has not been tested here. In conclusion the BTS established here has the potential to be a viable and field deployable solution provided the necessary avenues for increasing range and robustness are explored further.

REFERENCES:

1. Kenneth van Rijsbergen, The effectiveness of a homemade IMSI catcher build with YateBTS and a BladeRF, University of Amsterdam http://www.delaat.net/rp/2015-2016/p86/report.pdf