International Journal of Enhanced Research in Science, Technology & Engineering
ISSN: 2319-7463, Vol. 8 Issue 1, January-2019, Impact Factor: 4.059

GSM/PSTN SMS & DATA Gateway using Raspberry Pi B+

Tariq Ahmed[1], Mohammed Abdul Qadeer[2]

[1]University Women's Polytechnic Aligarh Muslim University, India
[2]Department of Computer Engineering, Aligarh Muslim University, India

ABSTRACT

The emergence of Voice over Internet Protocol (VoIP) has significantly reduced the cost of telephone communication, presenting a viable alternative to the traditional Plain Old Telephone System (POTS). This development is particularly beneficial for temporary or rural sites where affordable communication solutions are highly desired by organizations and individual users. Employees at temporary sites, such as construction sites, often require efficient and cost-effective systems to maintain communication during work hours. However, existing EPABX (Electronic Private Automatic Branch Exchange) systems are often expensive and involve substantial setup costs, making them less accessible for such contexts. This paper proposes the development of an affordable, single-box solution that integrates GSM/PSTN communication, SMS, and data gateway functionalities with minimal installation costs. The proposed device is specifically designed for temporary and rural environments, addressing the unique challenges of these settings. Additionally, the system emphasizes low power consumption to mitigate the limitations posed by inadequate power infrastructure in such areas. The working model is expected to provide an economically viable, energy-efficient, and reliable communication system that meets the needs of users in temporary or remote locations.

Keywords – VoIP, Data Gateway, EPABX, Public Switched Telephone Network SIP, RTP

INTRODUCTION

Traditional communication setups often involve the installation of a gateway at the customer's site while configuring the EPABX server at a separate location. These two components operate as independent entities, resulting in high setup and maintenance costs. This approach is particularly impractical in environments with insufficient communication infrastructure or unreliable power supplies, such as construction sites or rural areas. This research aims to develop an integrated, single-box solution that combines GSM/PSTN, SMS, and data gateway functionalities with the capabilities of an EPABX server. The system leverages the Raspberry Pi B+ platform to deliver a compact, reliable, and cost-effective communication solution. By merging these functionalities into a single device, the proposed system eliminates the need for separate installations, significantly reducing costs and enhancing portability. Additionally, the solution is designed to minimize power consumption, addressing the limitations of sites with inadequate power backup. The project aims to design and develop a GSM/PSTN SMS Data Gateway using Raspberry Pi B+ and a Huawei E303F modem. Key objectives include:

1. Establishing a Voice over IP (VoIP) network with a Raspberry Pi B+ IP-based EPABX server for inbound and outbound call routing.
2. Integrating the Huawei E303F GSM modem with Raspberry Pi B+ to enable VoIP users to connect to the GSM network.
3. Configuring the Huawei E303F GSM modem as an SMS Gateway for sending, receiving, and managing SMS messages.
4. Customizing the Huawei E303F GSM modem as a Data Gateway to provide internet connectivity to users.

The full featured gateway and EPABX server can easily be installed at temporary sites. This allows free SIP based calling across the network. Gateway provides low cost outgoing calls and data services. The Raspberry Pi B+ has been used as system design platform for this project. Additional functionality will comprise of Huawei E303 modem to make a GSM/PSTN, SMS and Data Gateway around Raspberry Pi B+ .The software component utilized is Asterisk, which serves as the interface programming layer, facilitating communication between hardware devices for making and receiving calls.Basically Asterisk is full-fledged EPABX framework compiled on Linux platform [1]. The GSM module for SMS gateway is controlled through AT commands. The 3G USB modem also enables data services through Sakis3g application.

LITERATURE REVIEW

This paper [18] explores developing a GSM gateway using a Bluetooth-enabled mobile phone for routing VoIP calls. While it enables free calls for registered users on an Asterisk-based PBX, it faces challenges such as high power consumption, disconnections, and poor voice quality. In contrast, the proposed GSM/PSTN SMS Data gateway offers better portability, lower power usage, and improved call quality.

The project [5] integrates PSTN and VoIP using an Asterisk server and SPA3000 gateway but faces high costs and lacks gateway integration with the Asterisk server, leading to resource inefficiency.

This approach [5] develops a PSTN-VoIP system using a motherboard, data modem, and flash memory. AstLinux, a lightweight combination of CentOS and Asterisk, is installed on a flash drive to reduce power consumption by avoiding a hard disk. However, the expensive Via motherboard increases overall costs. Since Asterisk is not natively compatible with data modems, ZAP channel files were modified for integration. The system connects the motherboard to PSTN and POTS networks but remains costly and complex. The availability of ethernet port on the motherboard allows the system to stay connected with Internet.

Multiple Providers of GSM/PSTN Gateway: The DBL GOIP-1 one-channel GSM gateway SIP IP Phone Adapter supports dual protocols: ITU-H.323 V4 and IETF SIP V2 [19]. It is compatible with G.711 A/μ law, G.729A/B, and G.723.1 codecs, and includes a router function [19]. The device offers one SIM card port for GSM extensions and supports both ITU-H.323 V4 and IETF SIP V2 protocols [19]. These products price range from $ 120.00 - $120.00.There are multiple providers e.g. Matrix, GoIP offer GSM/PSTN Gateway but their cost is very high. This project has implemented a system that is very cost effective.

In contrast, the proposed architecture uses a dedicated GSM/PSTN SMS Data gateway, offers a more cost-effective and energy-efficient solution integrated with the Asterisk PBX.

3. SYSTEM IMPLEMENTATION

3.1 Hardware Components

This project involves key components: Raspberry Pi B+ Microcontroller, a compact, low-cost single-board computer featuring ARM-based architecture; Huawei E303 GSM Modem, the Global System for Mobile Communication (GSM) is a globally adopted standard that provides digital cellular services, enabling mobile voice calls, text messaging (SMS), and data transmission. [15]; a Subscriber Identity Module (SIM) card inserts into mobile phone or GSM modem required to establish inbound call or outbound call [16] Micro SD Card, essential for storing and running the Raspbian image and RasPBX software; and an HDMI Cable for initial device setup.

3.2 Software Components

The Raspberry Pi Foundation developed Raspbian OS, the proprietary operating system for the Raspberry Pi B+ [26]. It is a Linux-based OS optimized for the Raspberry Pi, providing a graphical user interface along with essential programs, utilities, and applications [26]. This project uses the Raspbian Debian version [26], which can be easily installed on an SD card to boot the Raspberry Pi. Remote access can be achieved via SSH (Secure Shell) using software like **PuTTY**. PuTTY is an open-source application developed by Simon Tatham [27] that enables remote connections to a Raspberry Pi from another computer. By entering the IP address of the Raspberry Pi and the default username and password ("pi" and "raspberry"), users can access the Pi's command line interface and control it remotely. Sakis3g is a script designed to access internet settings from a SIM card and establish an internet connection using a USB dongle [28]. It is developed to work with Linux distributions. The Sakis3g script can be operated through both the command line interface and a graphical user interface [28]. It enables automatic connection to a 3G network during boot without requiring user interaction. The script provides an interactive GUI for connecting to a specific USB dongle and is compatible with most Huawei dongles.

3.3 Asterik

Asterisk is a linux based environment used for communication. It is an open source platform. It can run on any virtual machine or operating system. It was started by Mark Spencer in 1999 [2]. The core function of asterisk is IP-EPABX. It is designed to work with various protocols such as SIP, SCCP, H.323 [2].Asterisk includes five major parts and loadable modules that support the development of complex telephony systems in an industry [5] [6] [7].

It offers various incredible features widely used for most VoIP applications like conferencing, voicemail, Interactive Voice Response ,call transfer, call hold, call waiting, call conference, voice messaging, phonebook, ring group etc. [3][4][6].The Session Initiation Protocol (SIP) can be used to link a VoIP phone or softphone to the EPABX. Moreover, it supports analog and digital connections to the, the standard telephone infrastructure i.e. Public Switched Telephone Network (PSTN). Additionally, it has features like speech recognition and sound file playback that may be

easily combined to develop customized voice applications. We have default locations to save the files on asterisk server. Key Directories for Asterisk Files are listed below [28]:

/etc/asterisk/: Houses all configuration files necessary for system operation.
/usr/lib/asterisk/modules/: Contains loadable modules, including those for codecs, channel drivers, and format handling.
/var/lib/asterisk/: Stores resources such as sound files, images, firmware, and security keys.
/var/spool/asterisk/: Temporary storage location for files, including voicemail data.
/var/run/: Maintains process ID (PID) files for active processes, including Asterisk.
/var/log/asterisk/: Dedicated directory for storing Asterisk log files.
/var/log/asterisk/cdr-csv/: Contains detailed call records (CDRs) in CSV format.

Asterisk offers analog and digital connections to traditional telephone networks (PSTN). Once a call enters the system, it can be processed with various features, such as voicemail, call answering, and digit reading.

The major components of asterisk are: channels and extensions.conf –the dial plan in asterisk. A channel is a call in Asterisk established between a phone and Asterisk system. A connection between two phones could exist, representing two channels for this call, as illustrated in Figure 3.1.The most commonly media stream passed among end points over the channels is audio stream. Channels can utilize various technologies, such as SIP, IAX, H323, and others, as they are established [28].

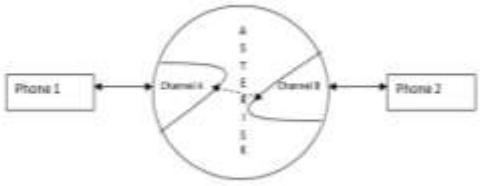

Figure 3.1 Connection between two phones with two channels

Dialplan defines the sequence of rules called extensions that are used by Asterisk administrator to route the call. The /etc/asterisk/extensions.conf file contains the Asterisk dialplan, which facilitates call management and control. When a call enters the system, the dialed number is used to identify the appropriate extension in the dialplan to process the call. Each extension includes a list of dialplan applications that will be executed on the channel. Asterisk dial plan allows applications to perform various functions such as play sound, answer call and interaction with database. The Asterisk dialplan allows multiple applications to be combined, enabling customized call handling through scripting interfaces in any programming language.

3.4 Dialplan Syntax & Dial Plan Pattern Matching

This dialplan triggers three applications when the extension 1342 is dialed. It first answers the call, then plays a previously recorded sound file, and finally hangs up the active channel. In the [calldemo] context, the exten keyword defines the extension, with the number "1" indicating the first step. The n denotes the next action, with "Answer" initiating the call.

```
[calldemo]
exten => 1342,1,Answer()
exten => s,n,Playback(hello-dear)
exten  =>s,n,Hangup()
```

The dialplan is a collection of contexts and context is a collection of extensions. Multiple contexts can have same extension if isolated with one another with its unique name. When there is a call it lands in the context declared by that channel.

Dial Plan Pattern Matching in Asterisk allows for compact and efficient configurations using patterns. For example, "exten => _06NXXXXXX,1,Answer()" matches numbers starting with 06 followed by any 9 digits, and "exten => _.,1,Playback()" matches any number of digits. Patterns like N (2-9), X (0-9), Z (1-9), and ranges such as [2-8] provide flexible matching options, such as "_03[1567]." for numbers starting with 031, 035, 036, or 037.

3.5 Soft phones

VOIP phones vary in price, features, and sound quality, using either SIP or IAX protocols [29]. For testing, free softphones like X-lite and Zoiper are effective [30]. These softphones run on a host computer, utilizing its microphone or headset. They are compatible with most operating systems and can register with an Asterisk server to place and

receive calls. The incoming calls are routed through Asterisk extensions. Both X-Lite and Zoiper are easy to set up and free to use, with sound quality depending on PC resources and network quality. The main advantage is their low cost.

3.6 RasPBX

Asterisk is a free open source framework and provides the facility to develop communication applications. FreePBX is a front end GUI based application to manage, configure and administrate asterisk features. RasPBX is a program combined with the features of Asterisk and FreePBX for Raspberry Pi. This implementation is available free and requires SIP phones for communication. This is installed on Raspberry Pi B+ as EPABX server. The latest version of RasPBX can be downloaded from website-raspberry-asterisk.org/downloads/.

3.7 Win 32Disk Imager

Win32 Disk Imager is an open source application to write image files on to the USB drives or SD cards [31]. This project used this utility to create the image file of Raspbian OS on the mirco SD card. Win 32 Disk Imager is available for free and zip file can easily be downloaded.

3.8 AT Commands

Table 3.2 AT commands for GSM modem [32]

AT Command	Explanation	AT Command	Explanation	AT Command	Explanation
AT^U2DIAG=0	Sets the dongle in modem mode	AT+CMGD=index [,flag]	Deletes messages from the specified storage location	AT?	Provides help for available commands
AT^CVOICE=?	Displays the status of the modem's voice functionality	AT+CMGF=1 AT+CMGL	Switches to SMS text mode Lists stored messages	ATI	Retrieves relevant information from the modem
AT^CVOICE=0	Enables voice functionality	AT+CMGS="MOBILE NO."	Sends a new SMS (message followed by {CTRL+Z})	AT+CSQ	Indicates signal quality
AT^CVOICE=1	Disables voice functionality	AT+CSCA?	Checks the SMSC (Short Message Service Center) number	AT	Checks the status of the modem
AT+CPBS	Selects SIM storage location	AT+CSCA="+SMSC"	Sets the SMSC number	AT+CPBW	Writes a contact to the SIM card

AT is abbreviation for **AT**tention. These commands control the modems. Mobile phones and GSM modem are compatible with AT commands that are specific to the GSM technology. It contains two categories of AT commands: Extended Commands, which begin with the "+" symbol, and Basic Commands, which do not. To control the GSM module AT commands given in Table 3.2 are used.

AT commands can be issued to control the GSM modem to access configuration information related to SIM card and modem [33] .It can also access information pertaining to data and voice, fax services, MMS services, SMS services [33]. These commands can be used to redial, receive, hangup the call and can also be used to send, receive, read, and delete the SMS. This project used AT commands to set up SMS gateway and dongle.

3.9 Session Initiation Protocol (SIP)

Table 3.3 Details of various SIP Response Messages [12]

SIP Messages	Type	Description	Function
REGISTER	Request	Registers the address or location of the IP phone.	Registers the client with the registrar server.
INVITE	Request	Initiates a media session between participants.	Specifies media details and initiates a call.
ACK	Request	Confirms the successful establishment of a media session.	Confirms the call setup after a successful 2xx response.
CANCEL	Request	Cancels a pending call attempt or unestablished request.	Terminates an unestablished call attempt.
BYE	Request	Ends an established call.	Terminates an active call, initiated by either party.
1xx (Informational)	Response	Provisional messages (e.g., phone is ringing).	Indicates the request is being processed.
2xx (Success)	Response	Request has succeeded.	The request has been successfully processed.
3xx (Redirection)	Response	Redirects the user to another location.	Guides the client to a different resource location.
4xx (Client Error)	Response	Represents errors caused by the client's request.	Example: Resource not found (404).
5xx (Server Error)	Response	Indicates the server's inability to process a request.	Internal server error
6xx (Global Failure)	Response	Global failure, no information on the user.	Indicates no information about the user in the system.

The Session Initiation Protocol (SIP) is a protocol used for signalling [9]. It operates at the application layer. SIP is used to manage multimedia communications, including voice, video, instant messaging, and file transfer [10]. It is defined by the IETF in RFC 3261 [10].

SIP Messages
SIP messages are divided into request and response messages as given in table, which are used to establish communication [12].

SIP supports two types of sessions: unicast, involving two parties, and multicast, involving multiple parties [9]. It consists of user agent clients (e.g., VoIP phones or smartphones) that send communication requests to user agent servers [9]. SIP servers include registrar servers, which store client names; proxy servers, which forward call requests to IP phones or other servers; and redirect servers, which locate endpoints, especially useful for mobile users with changing locations [9][11]. During the SIP communication process, when a client sends a request, the server responds with codes indicating whether the request was successful or failed.

3.10 RTP
Real-Time Protocol (RTP) ensures the end-to-end delivery of real-time audio and video data over IP networks, supporting both unicast and multicast services. While RTP provides best-effort delivery, it does not guarantee quality of service (QoS) or resource allocation. It works in conjunction with RTCP, which monitors QoS and synchronizes media streams by providing call quality metrics such as jitter, packet loss, and delay. RTP is essential in VoIP communications and can also be used for media on demand and multi-participant multimedia conferences [10] [13] [14].

4.System Description
The project integrates a Raspberry Pi B+ configured with RasPBX to function as an EPABX server. It is connected to a local network through Wi-Fi, allowing registered mobile or PC users, using softphones like X-lite or Zoiper, to make outbound calls via a cellular trunk to GSM and PSTN numbers. The Huawei E303F dongle, acting as a GSM/PSTN gateway, facilitates these outbound calls using an Airtel SIM card. The system also supports inbound cellular calls, redirecting them to registered extensions on the EPABX [35]. In addition to voice calls, the system handles SMS, enabling users to send, receive, and delete messages. Internet access is provided through the dongle using Sakis3g scripts, and the Lynx browser is used for web browsing.

The design process involved assembling the integrated system with modules like the Raspberry Pi B+, Asterisk 11.15.0, and FreePBX 2.11. This setup allows flexible management of client extensions and supports free VoIP calls within the LAN. The Huawei E303F modem was customized to handle GSM/PSTN calls, SMS, and internet connectivity, using AT commands. After ensuring all components met specifications, the system was tested successfully to meet the project's objectives.

Setting up Raspberry Pi B+ as Asterisk PBX server

To configure the Raspberry Pi B+ EPBX server, several components are required. These include a Raspberry Pi B+ microcontroller, a micro Secure Digital card, LAN cable (i.e. Ethernet cable), a micro USB charging cable, and an HDMI-compatible screen with a cable. Additionally, the Raspbian-Wheezy image, Asterisk and FreePBX software, and the Win32Disk Manager program are also required.

4.1 Install SD card to boot Raspberry Pi B+

To boot a Raspberry Pi, an SD card formatted with a bootloader and an operating system is required. The process involves downloading and installing Win32 Disk Imager on a Windows PC, selecting the raspbian-wheezy.img file, and writing it to the SD card [31]. After completing the write process, the SD card is ejected and inserted into the Raspberry Pi B+, which is then powered on. The system boots and prompts for a login; the default credentials for Raspbian are pi and raspberry. Initially, an HDMI screen and cable were used, but future access will be via network for remote operation.

4.2 Enabling SSH

Secure Shell (SSH) allows remote access to the Raspberry Pi through a command-line interface. Raspbian includes SSH functionality by default. To enable SSH, open the LX Terminal on the Raspberry Pi and execute the command sudo raspi-config. Navigate to the "ssh" option, press Enter, and select "Enable." Once the program runs, a confirmation message appears, indicating that SSH is successfully activated as shown in Fig.4.2.

Figure 4.2 SSH server Enabled

4.3 Network Configuration

The network configuration of a Raspberry Pi can be adjusted to utilize either Dynamic Host Configuration Protocol (DHCP), which is the default setting, or static IP addressing. The existing configuration can be verified by executing the command cat /etc/network/interfaces in the terminal. When the file contains the line iface eth0 inet dhcp, it indicates that DHCP is in use. Conversely, the presence of iface eth0 inet static denotes a static IP configuration. This particular project employs static IP settings because the Raspberry Pi is connected to an isolated network via Ethernet, which lacks DHCP capabilities. To set up a static IP, the file /etc/network/interfaces can be edited using the command sudo nano /etc/network/interfaces. The configuration should then be updated to include iface eth0 inet static along with the appropriate values for the IP address, subnet mask, and gateway.

For static IP settings, add lines for address, netmask, and gateway. This project is configured with the settings as highlighted in Fig. 4.3.

iface eth0 inet static
address 10.0.34.20
netmask 255.255.255.0
gateway 10.0.34.1
dns-nameservers 10.0.6.6 4.2.2.2

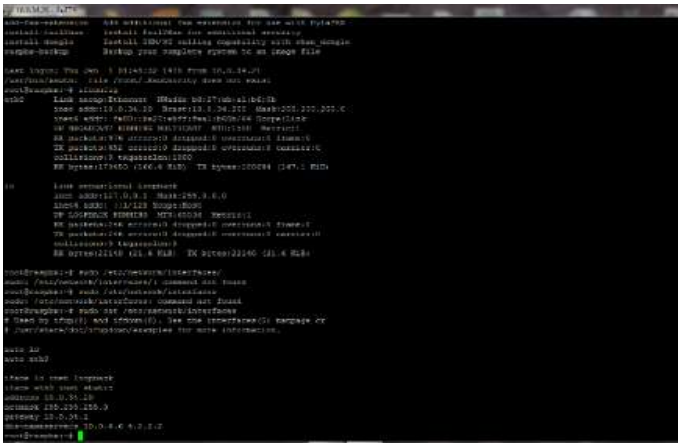

Figure 4.3 Network settings on Raspberry Pi B+

Enter Y to save the changes to the buffer and Exit nano editor by pressing combination of keys Ctrl+X. Then press Enter and return to command line prompt. Reboot the raspberry pi and tested the IP setting by logging over a SSH session using PuTTY program

On a Windows PC, this project uses the PuTTY program to remotely access the Raspberry Pi, eliminating the need for an HDMI screen, HDMI cable, and USB keyboard directly connected to the device. PuTTY program is a single file can be easily downloaded [27] and installed on PC. This project passes the following configurations as shown in Fig. 4.4 to access raspberry pi remotely. Enter the IP address and click "Open". With a warning message it will prompt you for the user ("root") and password ("raspberry"). Now user is ready to access raspberry pi remotely using command line interface as displayed in Fig. 4.5.

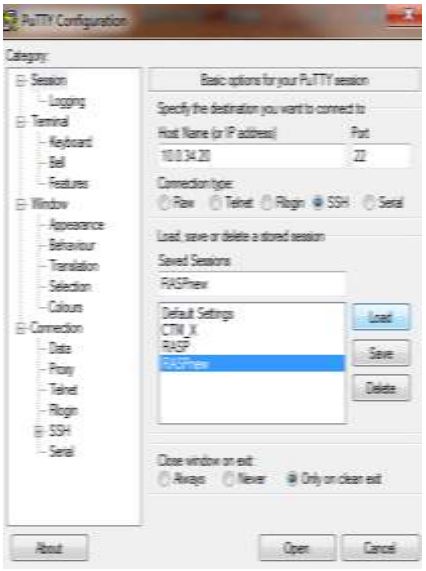

Figure 4.4 PuTTy Configuration

Figure 4.5 Remotely Accessing Raspberry Pi B+ using PuTTY Utility

4.4 Setup and Configuration- Huawei E303F USB Modem

Before configuring the dongle we require to change usb port to usb modem. Then connect the modem to Raspberry Pi. Configuring a 3G USB dongle as a modem for Asterisk faced challenges, particularly with outgoing and incoming call recognition, as Asterisk showed an error: "Error checking subscriber phone number." The issue arose because Asterisk could not verify the mobile number in the dongle.conf file as it was not stored on the SIM. The solution involved saving the mobile number to the SIM using AT commands: AT+CPBS="ON" (select SIM storage) and AT+CPBW=1,"9760******",145 (store the number). After restarting Asterisk, the dongle was recognized and functional, verified through the Asterisk CLI command dongle show devices.

The GSM modem was tested using the Linux-based Picocom program, installed on the Raspberry Pi B+ with sudo apt-get install picocom [36]. The modem was accessed via the serial port using the command picocom /dev/ttyUSB2 -b 115200 as shown in Fig. 4.6, enabling communication and interaction through the serial interface [36].

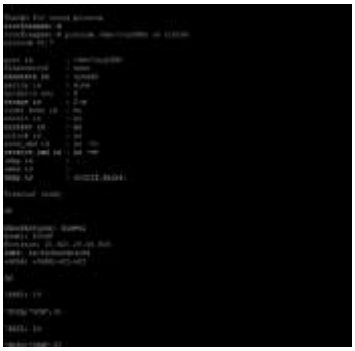

Figure 4.6 USB dongle interaction using AT commands

4.5 Enable voice command support of GSM dongle using AT commands.

To enable voice calls, the command AT^U2DIAG=0 sets the USB dongle to modem mode. The voice feature of the modem was verified using the Picocom terminal by entering AT^CVOICE=?, which returned ^CVOICE:(1) indicating voice was enabled. To activate the voice feature, the command AT^CVOICE=0 was successfully issued. Exiting the Picocom terminal required pressing CTRL-A followed by CTRL-X.

During setup, a 1A adaptor powering the Raspberry Pi B+ and the USB dongle caused connection issues despite the dongle being listed with the lsusb command. Switching to a 2A adaptor resolved the issue by supplying sufficient power for both devices to operate simultaneously.

4.6 Setting and Configuration – Raspberry Pi B+ as EPBX Server

Download the Raspbx image combination of Asterisk and FreePBX environment by visiting the website **www.raspberry-asterisk.org** and save it on desktop. Next write the raspbx image on to the SD card using Win32 Disk Imager utility. Insert the SD card into the raspberry pi and launch the RasPBX. Once booted successfully it shows the prompt as shown in Fig.4.7.

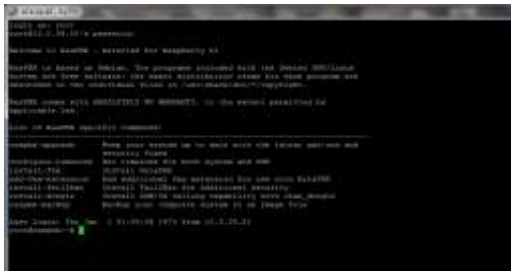

Figure 4.7 Raspberry Pi B+ EPBX server's Command Prompt

4.7 Create Extensions on Raspberry Pi B+ EPBX server

To create extensions on the Raspberry Pi server, access the FreePBX GUI by entering the PBX server address (e.g., 10.0.34.20) in a browser. Log in with the credentials admin/admin, navigate to Applications > Extensions, and select Generic SIP Device. Fill in required fields such as User Extension (e.g., 900), Display Name (e.g., saqib), Secret, and enable Voicemail. Leave other settings as default and submit. Repeat the process to create additional extensions for all softphones to register with the Raspberry Pi PBX server as shown in Fig. 4.8.

Figure 4.8 List of Extensions created on Raspberry Pi B+ PBX server

4.8 Configure Softphone

This project avoids using hard phones, as they require expensive interfaces with the EPBX. Instead, softphone applications like X-lite as shown in Fig. 4.9 or Zoiper, which are free, are used to connect to the Raspberry Pi PBX. To set up a SIP client, the IP address of the PBX server (e.g., 10.0.34.20) is needed. SIP clients are configured on mobile

phones using the built-in SIP settings, where the username corresponds to the extension number, the password is the secret value, and the server address is the PBX's IP. Softphone applications such as X-lite for PC are also used. The X-lite client, for example, registers with the Raspberry Pi B+ Asterisk-based PBX server at 10.0.34.21 with extension 900. Upon registration, Asterisk responds with an acknowledgment. If the client is authorized, an "OK" response is generated; otherwise, a "401 Unauthorized" message is returned, prompting the client to try again.

To verify registration status on the Raspberry Pi B+ PBX server, run sip show peers as shown in Fig. 4.10 after starting Asterisk with the command asterisk -vvvvvc.

Figure 4.9 Xlite- SIP Account Details

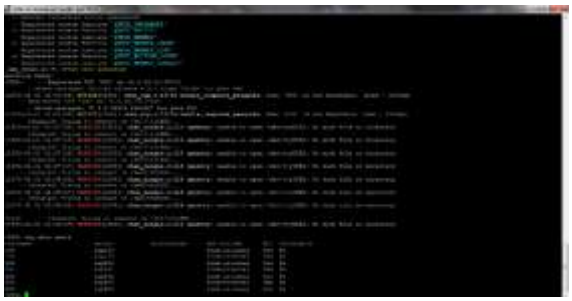

Figure 4.10 Registered SIP peers configuration

All VOIP calls within a local network are routed through Raspberry Pi B+ PBX server to registered extension.

As asterisk server holds IP address of all registered systems an INVITE request is generated and forwarded by the client. It establishes SIP session to the callee phone by the Raspberry Pi B+ EPBX server. The call receiver - X-lite softphone with extension number 800 starts ringing and responds with the “180 Ringing” code to the Raspberry Pi B+ EPBX server. It further carries the response to the caller X-lite softphone with extension number 900. The receiver picks up the call and send the “200 OK” status. This message carries SDP information to declare the destination for RTP streams. The 900 extension confirms the response and EPBX server forwards the ACK status to the extension 800 as demonstrated in Fig.4.11. Now call is established and end points are ready to send RTP streams. These softphones are tested using asterisk based server on Raspberry Pi B+ and delivers the good results. The voice quality was up to the mark.

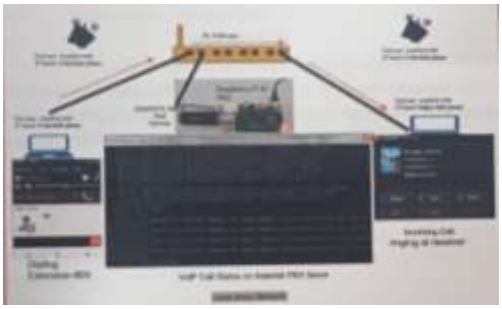

Figure 4.11 Free SIP based calls within local Network using Raspberry PI B+ PBX server

This set up is not only restricted with wired communication but also facilitate us with wireless Wi-Fi based mobile phone or laptop settings. Mobile phones or laptop connected to Wi-Fi router that establishes a wireless network. To do outbound call, this asterisk IP-EPBX server does not allow user to use normal phone keypad. For this softphone application – zoiper is installed on mobile devices. This small program is used to make VoIP calls. Another approach is to use built in facility of Android operating system to generate SIP client on mobile phones which in turn used to make VoIP calls.

4.9 Huawei E303 USB Dongle as GSM/PSTN Gateway

Once the USB dongle is enabled to support voice functionality and configured as a modem, as described in section 4.4, the next step is to customize it as a USB dongle GSM/PSTN gateway. This is achieved by configuring a SIP trunk to route cellular calls dialed from registered extensions on the Raspberry Pi B+ PBX server. The calls are then forwarded to external cellular or PSTN numbers via the USB dongle, based on the pattern matching defined in the SIP trunk parameters. A SIP trunk essentially serves as a connection to the VOIP server, facilitating the routing of calls. It allows multiple calls to be transmitted, with the only limitations being bandwidth availability and the resources of the computer running Asterisk, unless the VOIP server itself imposes a limit on the number of concurrent calls.

Launch the FreePBX application and navigate to Connectivity, then to Trunk. Add a new SIP trunk to enable calls through the USB modem. Set the trunk name as call2gsm, configure the Outbound Caller ID with the mobile number, and set the Maximum Channels to 1. Define the Dial Pattern and set the custom dial string as dongle/dongle0/$OUTNUM$. Additionally, the project is programmed to handle incoming calls received from any GSM or PSTN number via the USB dongle. These calls are routed to a specific registered extension through the Raspberry Pi B+ Asterisk-based server.

Once the GSM modem configuration file is specified and the SIP trunk is configured, the next step involves configuring outbound routes in the FreePBX environment. To do this, navigate to Connectivity and then Outbound Routes. Provide the route name, define the dial patterns that will be allowed for use with this route, and select the SIP trunk call2gsm under the Trunk Sequence for Matched Routes option. As defined in the extensions.conf file, the Asterisk server is now set up to manage all outgoing calls in accordance with the Asterisk dialplan.

The setup uses the context to map the dialed number to the corresponding dial patterns. In this case, dial patterns for GSM (i.e., ZXXXXXXXXX) and PSTN (i.e., XXXXXXXXXXX) are defined. When a SIP client dials any GSM or PSTN number, the call is routed to the corresponding number through the Asterisk server, according to the pattern match in the extensions.conf file generated by the FreePBX environment. Testing of the GSM gateway for outbound GSM or PSTN calls was successful as shown in Fig 4.12 & Fig.4.13.

To handle incoming cellular calls, the following configuration is applied to redirect the incoming call to a specified extension. In the FreePBX GUI, go to Connectivity and select the Inbound Routes tab to forward incoming calls from GSM or PSTN numbers to a configured extension. Enter the following details in the route window and leave other settings as default:

- **Route name**: route2gsm
- **Dial pattern**: ZXXXXXXXXX and XXXXXXXXXXX
- **Set Destination**: Extensions – 800

Once these configurations are applied to the inbound route, any incoming call to the GSM/PSTN gateway is dynamically redirected by the Asterisk PBX server to extension 800 as shown in Fig. 4.14, based on the dial pattern match.

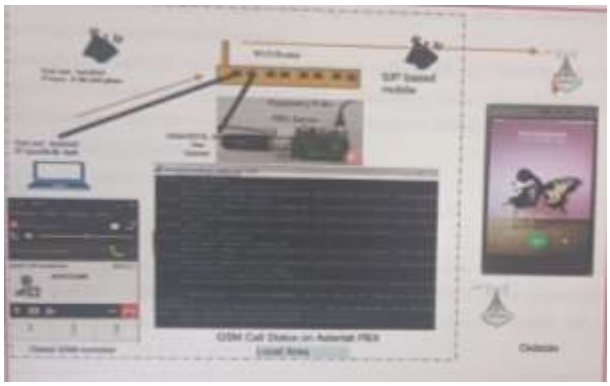

Figure 4.12 GSM Outbound Call using GSM/PSTN gateway

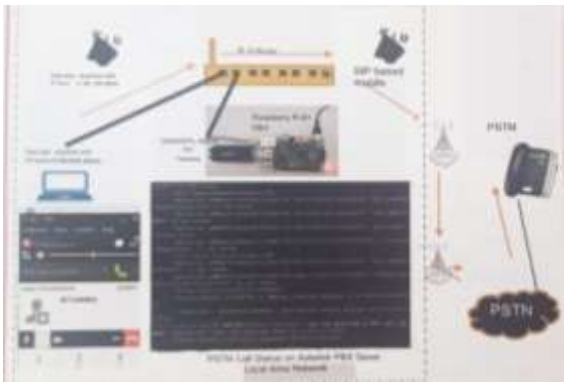

Figure 4.13 PSTN Outbound Call using GSM/PSTN gateway

Figure 4.14 Incoming call forwarded to etension using GSM/PSTN gateway

4.10 3G USB Modem as Data Gateway

This project facilitates internet access using a 3G USB modem on a Raspberry Pi B+ PBX server by employing the Sakis3G shell script, which enables the modem to connect to its GSM carrier. The Sakis3G script can be downloaded from http://www.sakis3g.com as a compressed file [38]. After downloading, the file should be decompressed using the command sudo tar -xzvf sakis3g.tar.gz. The command chmod +x sakis3g is used to make the script executable. The application can then be launched in an interactive GUI environment using sudo sakis3g --interactive. Once launched, the user selects the "Connect with 3G" option, followed by the appropriate modem category—in this case, the USB device category—and clicks OK to proceed.

Next, the USB modem is identified (e.g., Huawei Mobile), and its interface is configured, with Interface #0 being selected for this project. The software then prompts for the operator's Access Point Name (APN), which is automatically recommended by Sakis3G, and the user provides the required username and password for authentication. After the setup process completes successfully as shown in Fig.4.15, a confirmation window is displayed, the modem is ready for internet access.

To utilize the internet through this data gateway, the lynx web browser must be installed using the command sudo apt-get install lynx. Internet connectivity can then be tested by launching the browser with lynx http://www.google.com, as demonstrated in Fig. 4.16.

Figure 4.15 E303F Dongle connected to Airtel Service Provider

Figure 4.16 Internet access using data gateway installed on Raspberyy Pi B+ server

4.11 E303F USB Modem as SMS Gateway

4.12

This project used Huwaei E303F USB dongle to connect the Raspberry Pi B+ to the GSM network. Generally 3G USB dongle operates in two modes- storage mode and modem mode. It becomes a troublesome job to configure USB dongle in modem mode under Linux. This challenge is already accomplished in Section 6.2.4. Huwaei is already switched in 3G modem mode.

Before you send SMS using SMS gateway, it is necessary to check and set the correct Short Message Service Centre (SMSC) number. When you send SMS using +CMGS AT command, GSM modem will forward the SMS to particular destination perform using service center address +91897051914.
To set and check service centre address enter the following commands:
AT+CSCA? // Check service center address AT+CSCA=”+91897051914” // Sets service center address

4.13 Sending SMS

To use a GSM modem as an SMS gateway, the 3G dongle must first be connected as a modem. Communication with the modem is then established using AT commands, which are sent via the picocom application. These commands facilitate sending and reading SMS messages over a serial link. To begin, the modem on the serial port is accessed by typing the command picocom /dev/ttyUSB2 -b 115200, which opens a direct communication interface.

Upon entering the command AT and pressing ENTER, the modem should respond with OK, confirming the connection. SMS messages can be sent either through text mode or Protocol Data Unit mode. PDU mode involves binary encoding, adding complexity to the system design whereas Text mode is basic and easy to use. In this project, SMS transmission is demonstrated in text mode. The text mode is activated with the command AT+CMGF=1, which returns OK to indicate successful configuration.

To send an SMS, the CMGS command is used as follows: AT+CMGS="phonenumber"<CR>the message to send<Ctrl+Z>. The phone number can be entered with or without the country code (e.g., 0XXXXXXXXX or +91XXXXXXXXXX). Here, <CR> represents the Enter key, while <Ctrl+Z> signals the end of the message. Upon successful transmission, the modem responds with +CMGS: 152 OK as demonstrated in the Fig.4.17.

Figure 4.17 Sending SMS using SMS Gateway

Handling Received SMS and Delete SMS

To read incoming messages, the modem must first be set to text mode. To retrieve all received SMS messages, the command AT+CMGL="ALL" is used as displayed in Fig.4.18. The parameter "ALL" can be replaced with other values depending on the desired message type. The available options include: "ALL" to list all messages, "REC READ" to list all received and already read messages, "REC UNREAD" to list all received but unread messages, "STO SENT" for stored and sent messages, and "STO UNSENT" for messages that are stored but not sent.

Figure 4.18 Read ALL SMS using SMS Gateway

To manage storage and free up space when memory is full, the +CMGD command is used to delete messages. This command follows the syntax +CMGD=index[,flag], where index specifies the integer value of the location of the SMS to be deleted, and flag determines the scope of deletion based on SMS status. The flag value can be one of the following: if '0' it deletes message defined at the particular index, if '1' deletes only all "received read" messages, if '2' deletes all "received read" and "stored sent" data, if '3' deletes all messages whether "received, sent, and unsent" messages, and 4 to delete all messages. In this project, the flag value 0 was used to delete specific messages at a given index, while the value 2 was tested to delete all read messages from storage. For example, the command AT+CMGD=1,2 deletes all read messages, and the response OK confirms successful execution.

This project used a terminal program to issue AT commands to send, receive and delete SMS. But this project may further be improved by writing a program to operate GSM modem through AT commands. This program can be implemented using programming languages such as C, C++, Java, Visual Basic, or others [39].

CONCLUSION

The design and development process of the project was successfully achieved. The embedded system performs as expected. The end product is a compact, all-in-one system that serves both as a gateway and a Private Branch Exchange server.The implemented system is very portable and light-weight. The system is user friendly and could be configured and installed with minimal efforts. The integrated system is hassle free a cost efficient solution and significantly cuts down the call rates for home users and small business organizations. This project has been tested up to 6 parallel VoIP voice calls. The system delivers the good voice call quality but sometimes it gets hang up and delivers echo problem. This is easy to maintain as it needs no add on wiring set up or hardware. Wireless devices such as mobile phones or laptops can communicate free of charge within the range of Wi-Fi network.

FUTURE WORK

Some of the suggestions are provided for future work an algorithm could be designed for minimal cost routing which should select port dynamically. And also by installing the FXO interface with Raspberry Pi B+ a PSTN to VoIP gateway permitting calls from a PSTN phone line to reach a VoIP user.The Raspberry Pi can be integrated with cloud services to create a robust communication and monitoring system capable of managing PSTN, GSM, and data processing. Cloud systems clubbed with the Raspberry Pi could provide advance data analysis, data retention, and centralized management. Among the various cloud platform options, one can choose from AWS, Microsoft Azure, or Google Cloud. Cloud services would assist to manage call logs, SMS records, and IoT devices and offers an integrated centralised setup for monitoring and control.

REFERENCES

[1]. Mohammad Masusdur Rahaman, Nafish Sarwar Islam Using Asterisk PBX "VoIP Implementation Using Asterisk PBX", IOSR Journal of Business and Management (IOSR-JBM,Vol. 15, Issue 6 Jan 2014, e-ISSN 2278—487X pp 47- 53.
[2]. Ms.Harshada Jagtap, D.G.Gahane, " Asterisk based IP-Based IP-PBX Cost Efficient Server For Small Organization" International Journal on Recent and Innovation Trends in Computing and Communication" ,Vol.3,Issue 2 ISSN 2321-8169 pp.084-086
[3]. Ale Imran, M.A. Qadeer," Conferencing, Paging, Voice Mailing via Asterisk EPBX", International conference on Computer Engineering and Technology, 2009.
[4]. Fumikazu Iseki,Yuki Sato, Moo Wan Kim, " VoIP System Based on Asterisk for Enterprise Network" ,ICACT 2011.
[5]. Priyesh Wadhwa ,"Design of PSTN-VoIP Gateway with inbuilt PBX & SIP extensions for Wireless medium" M.Tech Thesis Department of Computer Science and Engineering, Indian Institute of Technology Bombay,2007
[6]. Ghita B.V., Furnell, S.M., Lines,B.M., Le-Foll D., Ifeachor, E.C (2001), "Network quality of service monitoring for IP telephony", Internet Research: Electronic Networking Applications and Policy, Vol. 11 No.1, pp.26-34
[7]. Thorne D.J (2001), "VoIP-the access dimension", British Telecom Technical Journal, Vol.19 No.2, pp.33-43
[8]. Bourrea, M., Dogan P.(2004), "Service –based vs. facility –based competition in local access networks", Information Economics and Policy, Vol.16 No.2 ,pp 287-306
[9]. Anshuman Srivastava Kumud Sharma, " New Generation Networks Architecture between H.323 and SIP Protocol", International Journal of Advanced Computer Science and Information Technology 2013, Vol. 2 Issue 1 ,pp. 34-41 ISSN 2320-0235
[10]. Sheetal Jalendry Shradha Verma, " A Detailed Review on Voice over Internet Protocol (VoIP)" International Journal of Engineering Trends and Technology May 2015, Vol. 23, pp. 161 ISSN: 2231-5381
[11]. M. Handley and V. Jacobson, SDP: Session Description Protocol. RFC 2327, April 1998. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2327.txt
[12]. http://www.tutorialspoint.com/session_initiation_protocol/session_initiation_protocol_messaging.htm
[13]. http://www.cse.wustl.edu/~jain/books/ftp/rtp.pdf
[14]. https://www.rfc-editor.org/pdfrfc/rfc1889.txt.pdf
[15]. global_system_for_mobile_communication_technology.pdf
[16]. http://www.transparencymarketresearch/subscriber-identity-module.htm
[17]. http://www.developers home .com/sms/GSMModemIntro.asp
[18]. Qadeer Mohammed A,Priyanka Gupta, Neha Agrawal , " GSM and PSTN Gateway for Asterisk EPBX."Wireless and Optical Communications Networks (WOCN),IEEE International Conference 2013
[19]. http://www.dbltek.com/products/goip-1.html
[20]. Manav Chandna Nishant Tyagi Rahul Singh Dr. Arvind Rehalia," Case Study : Raspberries Pi B+", International Journal of Advanced Research in Computer Science and Software Engineering, August 2015, Vol.5 Isuue 8,ISSN: 2277 128X
[21]. Soundhar Ganesh S,Venkatash S, Vidhyasagar P, Maragatharaj S," Raspberry Pi Based Interactive Home Automation System through Internet of Things", International Journal for Research in Applied Science & Engineering Technology (IJRASET) March 2015 Volume 3 Issue III, ISSN: 2321-9653
[22]. https://www.raspberrypi.org/blog/price-cut-raspberry-pi-model-b-now-only-25/
[23]. http://www.adafruit.com/pi-specs.pdf accessed
[24]. http://www.modem3g.com/huawei-e303-usb-modem.html
[25]. http://www.modem3g.com/huawei-e303-usb-modem.html
[26]. https://www.raspbian.org/
[27]. http://www.putty.org
[28]. http://www.apricot.net/Jonny_Martin_Asterisk.pdf
[29]. http://www.counterpath.com/xlite-comparison/
[30]. https://www.zoiper.com/en
[31]. https://www.sourceforge.net/proects/win32disImager/
[32]. http://www.engineersgarage.com/tutorials/at-commands?page=4
[33]. http://www.engineersgarage.com/tutorials/at-commands
[34]. http://www.lovisolo.com/GUIDE-HOWTO/chan_dongle_commands.pdf
[35]. Mohammad A Qadeer, M J R Khan Ale Imran," Asterisk VoIP Private Branch Exchange", IMPACT-2009, in IEEE, 2009.
[36]. https://developer.ridgerum.com/wiki/index.php/Setting_up_Picocom_-_Ubuntu
[37]. http://www.raspberryasterisk.org/downloads/
[38]. http://www.sakis3g.com/downloads/sakis3g.tar.gz
[39]. Bin Abdullah AT,Ismail IB,Ibrahim AB, Hakim Bin Noor MZ(2011),"Library shelf management system using RFID technology", IEEE trans. System Engineering and Technology(ICSET),IEEE International Conference, pp 215-218