University of West Bohemia
Faculty of Applied Sciences
Department of Computer Science and Engineering

Master's thesis

Electronic Gatekeeper using ETHERNET

Plzeň 2019 Hamza Elghoul

Místo této strany bude zadání práce.

Declaration

I hereby declare that this master's thesis is completely my own work and that I used only the cited sources.

Plzeň, 2 May 2018

Hamza Elghoul

Abstract

The aim of this thesis is the analysis and the realization of a budget 2-way communication system using ETHERNET connection, more specifically a door intercom station (hereinafter ”Bouncer” or "Gatekeeper"), where it is possible to make voice calls between two CLIENTS, a CALLER and a CALLEE with an acceptable to minimal delay.

There are many alternative methods available for implementing such systems, this project will try to compare some of these solutions and choose the most convenient platform according to a number of factors, such as bandwidth, processing power needed and communication protocols in question.

Keywords: VoIP, SIP,SDP, server, client, LAN, ethernet, embedded, audio, raspberry pi

Contents

List of Figures

Acronyms

BBB Beagle Bone Black. 17

IP Internet Protocol. 11

LAN Local Area Network. 11

PBX Private Branch Exchange. 27

SIP Session Initiation Protocol. 21

UI User Interface. 12

VOIP Voice Over IP. 19

VPN Virtual Private Network. 13

1 Preface

Considering the high rate of crime and insecurity, there is an urgent need to design a home security system that takes proper measures to prevent intrusion, unwanted and unauthorized user(s). To solve these problems, knowing the person at one's doorstep is a key. This project aims to make this process smoother and more reliable and most importantly expandable.

The doorkeeper must be IP-compatible in order to take advantage of the addressing scheme provided by Internet protocol and network architecture available. Added to that, the solution shall be an IP network-compatible intercom system that employs packet audio technology. It shall be capable of connecting the main station, located at the gate, and the different Internet Protocol (IP) stations - clients, that are part of the local area network Local Area Network (LAN). The system shall feature an echo cancelling function that prevents acoustic feedback and echo in order to make possible full-duplex hands-free conversations between stations. Additionally, the use of IP stations shall allow the construction of systems that can operate without an exchange. Centralized control of the system shall be possible using a PC or similar dedicated server, running a manager software, which will control the communication session parameters and allow management and expansion.

2 Analysis

2.1 Objectives and Requirements

The real-time voice transmission over Ethernet medium is being exploited by various types of applications to abate the growing need of a secured, low-cost and compact embedded device. Real time Audio & Video conferencing is nothing but performing live chat with two or more hosts in real time over wired or wireless network. In this project a new fully-functional embedded system has been proposed according to a number of factors, in order to be able to transmit voice in real-time over Ethernet with an acceptable audio quality.

Developing an embedded product requires three main steps: selection of the proper hardware platform, operating system, and User Interface (UI) development, if applicable. All three of these steps are very closely tied together and have a significant impact on time-to-market, project costs, and the overall quality of the project. It is critical to select the proper hardware platform that also has excellent operating system support as well as the proper UI tools.

With a variety of different single board computers, it can be difficult to choose which is right for one's project. This section will look at the parameters of the most popular single board computers available on the market and choose the adequate one according to the requirement of the the gatekeeper.

2.2 Hardware

Embedded systems, or single board computers, are now manufactured in large quantities and are widely and commercially available. There is a huge range of such devices on the market, each with different parameters and primarily produced for a particular segment of use. Almost every embedded system now has support across many operating systems, even tailor-made platforms are being developed for them. Most of these created operating systems are based on the Linux kernel, bringing in a lot of capabilities such as web-server, Virtual Private Network (VPN), media center and even in home automation.

Embedded systems are primarily developed for specific applications and then deployed in difficult conditions such as high temperatures, humidity, vibration and more. Virtually in every home appliance, we can find a unique purpose-made embedded system. In this chapter, a couple of theses systems will be tested for a specific type of application, which is audio and voice communication. Among these systems are: RaspberryPi, BeagleBone Black, Cubieboard, UP, BananaPi, Arduino, and others.

2.2.1 Arduino

I'm sure everybody heard about the Arduino, Raspberry Pi and Beagle-Bone boards among other types of boards.

The Arduino is a microcontroller. A microcontroller is just one tiny part of a computer. The arduino can be programmed in C, but can't run an operating system. On the other hand, the Rasperry Pi and Orange Pi are computers, which means they can run an operating system by their own.

The Arduino is simply perfect for electronics projects and prototyping. It is really easy to connect some LED's, sensors, motors into the board directly. To program the Arduino, their official software is needed (can be download for free). Basically with that software it is possible to upload source code directly into the Arduino through USB, unplug the USB cable, attach a battery to the Board and It will run our program forever.

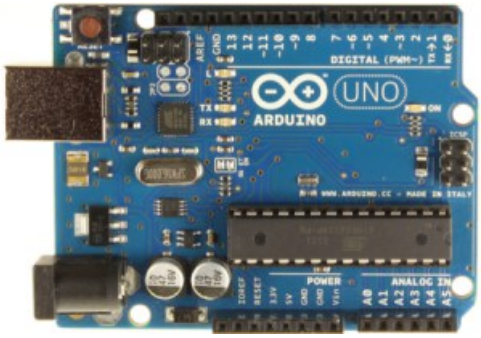


Figure 2.1: Arduino UNO board

Despite having an ETHERNET port, 14 Digital I/O Pins and being very easy to use, an Arduino is a micro-controller motherboard. A micro-controller is a simple computer that can run one program at a time, over and over again, which ,in our case, is not convenient. Added to that the small memory on-board (just 32 KB of which 0.5 KB used by bootloader) and the low clock Speed of just 16 MHz, makes Arduino a platform not suitable for the project.

2.2.2 RaspberryPi 3 Model B

Figure 2.2: Raspberry Pi3 Model B+

The Raspberry Pi 3 Model B is already the third generation of single board systems, compared to the first and second generations, the third brings an easier way to install any operating system supported, which can be handled by inexperienced users, as well as a more powerful 1.2 GHz ARM Cortex-A53 processor. It has 1GB of on-board RAM, includes a 100 Mbit/s Ethernet interface for network connectivity. HDMI port and four USB 2.0 ports for peripheral connection, populated 40 GPIO pins, audio output jack and a microSD card slot for data storage.

Power is supplied via microUSB connector. The device is powered by 5V with a recommended supply current of at least 1A. The indisputable advantage of RaspberryPi is the ability to power this device with a conventional mobile phone charging adapter.

2.2.3 Banana Pi

Banana Pi is called obviously after the Raspberry Pi board and is really similar to it to some extent.

It comes in different versions, all open source and freely available for use and modification. Its advantages over Raspberry Pi include built-in buttons on the motherboard (On / Off, reset), compatibility with Raspberry Pi accessories (displays, cameras modules). Banana Pi is released in several versions depending on equipment and performance.

•Banana Pi BPI-M1+

The improved version of the original Banana Pi is called BPI-M1 + and comes with a more powerful processor,the A20 ARM Cortex A7, a two core processor with 1 GHz clock. 1GB of DDR3 memory, Gigabit Ethernet, HDMI video output, Audio Jack, GPIO ports and 2x USB. It also has an integrated Wi-Fi wireless chip with antenna, system control buttons and a MicroSD card slot. The power supply is guaranteed in the same way, ie via a microUSB.

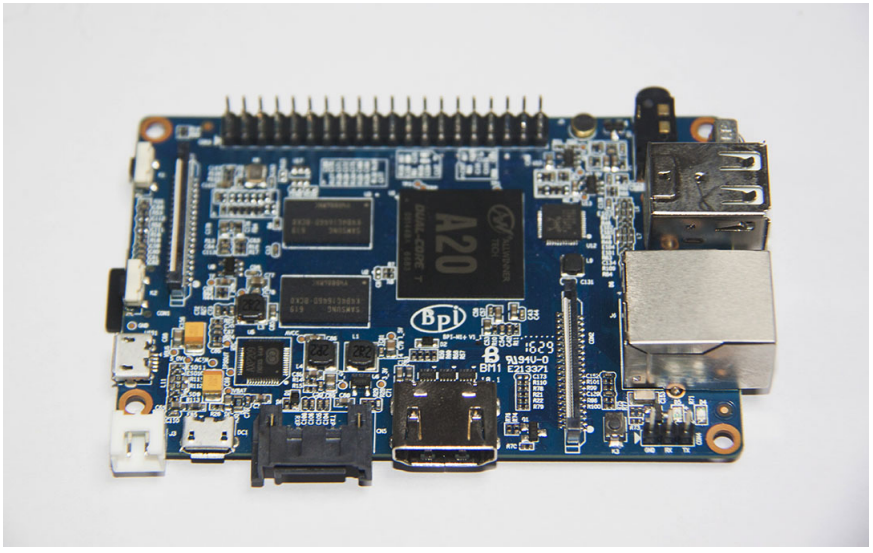


Figure 2.3: Banana Pi BPI-M1+ [8]

2.2.4 Orange Pi

Following the great success of Raspberry Pi,a number of other "fruit" embedded systems competitors are being developed. As these systems are created after the success of their predecessor, they eliminate some of the deficiencies of the original systems and come in improved variants with greater performance. Orange Pi system on board has ARM Series A7 processor from AllWinner. This platform is established and manufactured in China, specifically by Shenzhen Xunlong Software CO., Limited.

•Orange Pi PC

This board offers a 1.6 GHz quad-core H3 Cortex-A7 processor, 1 GB DDR3 RAM, 100 MBit / s Ethernet, a MicroSD card slot, an HDMI connector, and the same interface as Raspberry Pi. An indisputable advantage over Raspberry Pi is greater processor performance, an on / off button and, of course, a price that is around US $ 15. The entire device is powered by a 5V / 2A source.

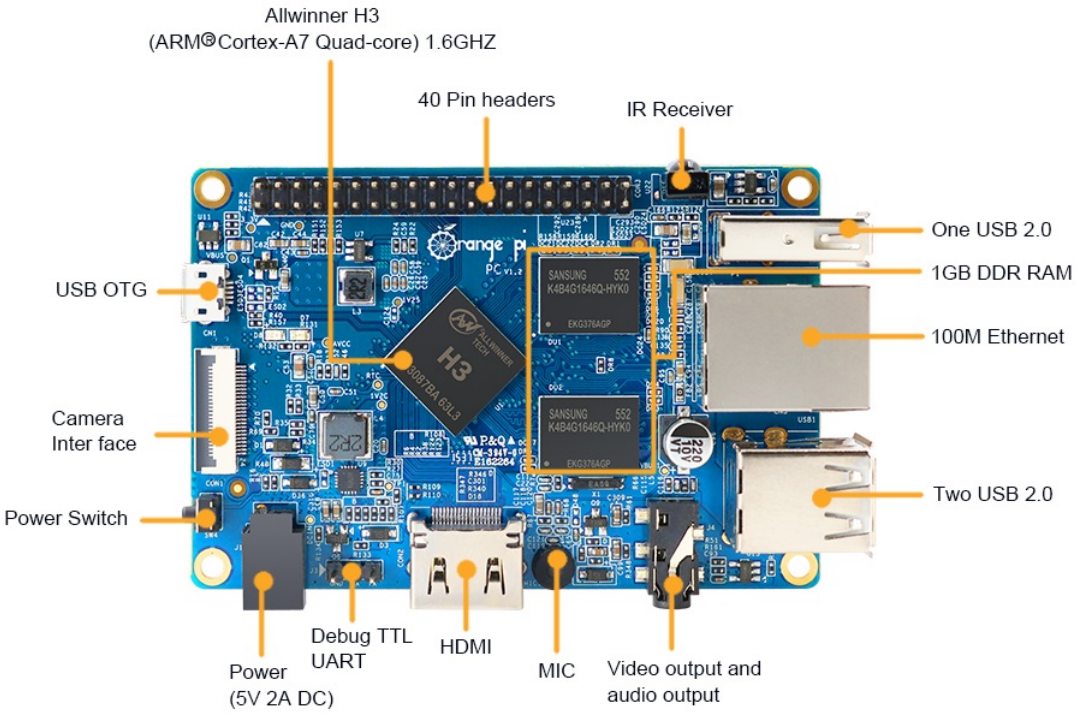


Figure 2.4: Orange Pi PC Board [9]

2.2.5 CubieBoard 2

Among the less-known single board computers is the open source platform CubieBoard. CubieBoard is one of the few development groups dedicated to the ability to concatenate multiple embedded systems to create a more powerful platform for more demanding applications such as Web Cloud, WebServer and many more. The device has a 912 MHz AllWinner A20 processor, 1 GB DDR3 RAM and has 3.4 GB of internal flash memory. For additional memory expansion, there, a microSD card slot and a SATA connector for hard drive connection. Network connectivity is provided by 100 MBit / s Ethernet, USB and Wi-Fi support. CubieBoard also has HDMI and can be powered from USB or use a standard 5V / 2A DC power interface.

Figure 2.5: CubieBoard 2 system-on-chip [10]

2.2.6 Beagle Bone Black

Another alternative to consider is Beagle Bone Black (BBB), a low-cost, community-supported development platform for developers and hobbyists that can Boot Linux in under 10 seconds and get started on development in less than 5 minutes with just a single USB cable.

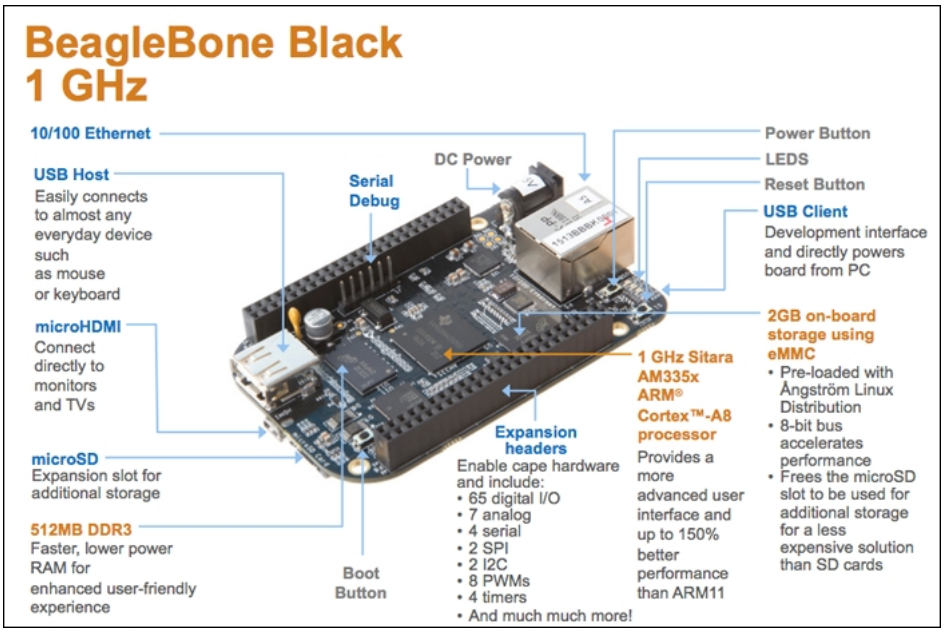


Figure 2.6: Beagle Bone Black board [11]

BBB is well equipped, with 512MB DDR3 RAM, 4GB 8-bit eMMC on-board flash storage, 3D graphics accelerator, 1 USB client for power & communications and 1 USB host and of course Ethernet port. But the lack of any AUDIO/VIDEO input device is a weakness, which can be overcame by using extension boards called "Hats" or even by USB compatible devices, but the BBB has only 1 USB port, and usb hubs are known to cause issues as they overload the one port they are connected to and may damage the devices connected to them.

Added to that, as the previously stated boards, the BeagleBone Black is a newcomer, which means it lacks in experience and support.

In a nuttshell, this will leave us with the first alternative, the Raspberry Pi 3 board, the ultimate single-board, low-cost, high-performance computer.

The Raspberry Pi Foundation even builds its own Distro of Linux – Raspbian – that comes with all the tools we could ever need on the Raspberry Pi preinstalled, and is also able to run software packages a lot better than the Beaglebone.

With the incredible amount of documentation and support available online, it can be quite easy to do whatever needed with a Raspberry Pi.

The multiple USB ports available in the RPi, the audio output, the Linux based OS and the enough good computing power to do audio and network processing makes it the most convincing platform for implementing this project, while most of the Chinese clones (Banana and Orange Pi) use

different System on chip (from ALLWINNER) which can contribute to a number of compatibility problems, the Raspberry uses a reliable high quality SoC from Broadcom, added to the huge lack of support and documentation available.

2.3 Application

In this section, the Raspberry Pi is intended to have capability of gathering audio from a USB microphone. As the system is IP based, and in order to make implementation more convenient and portable, a Voice Over IP (VOIP) protocol is to be used to assure the transmission of audio packets between the different clients in the system.

VOIP is the method for delivering phone calls "over IP." VOIP uses the Internet Protocol (IP) to transmit voice as packets over an IP network. So VOIP can be achieved on any data network that uses IP, like Internet, Intranets and Local Area Networks (LAN). Here the voice signal is digitized, compressed and converted to packets and then transmitted over the IP network. In order to to set up and tear down calls, carry information required to locate users and negotiate capabilities, a signalling protocol is needed.

In general, this is how a typical VOIP setup looks like:

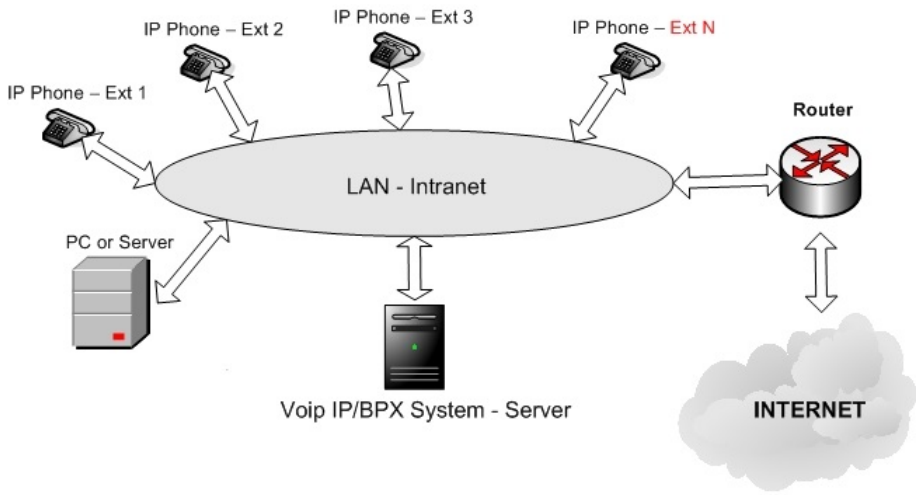


Figure 2.7: VOIP LAN/NETWORK diagram

The idea, is to have a server who would manage the backend/structure of the whole system, and a number of clients, all interconnected by the means of IP capable network, for example a LAN. Since our usage is limited to a simple voice call between the clients in the system, so there no need of any

internet access for this type of gatekeeper project, where the communication between different elements is purely internal.

The next challenge is the design of such system using the capabilities offered by the system on chip chosen for this purpose.

Added to that, the voice communication needs to be somehow processed in a timely manner, in order to make the whole system suitable for use in production-like environment with little to no support, and overall reliable and ready for future upgrades or extension, for example, adding a client or otherwise deleting/modifying him.

2.4 Audio capture and digitalisation

Audio transmission and codecs

Voice Over internet Protocol uses computer algorithms called "codecs" to convert the voice signal from your telephone into a digital signal that is then transmitted over the internet. If you are calling a regular telephone number, the signal is then converted back on the other end by switches that are near your terminating phone. This is why every call is a local call with VoIP.

G.711

This is the basic PSTN codec (public telephone network). If someone talks about PCM (Pulse Code Modulation) along with a telephone network, he certainly means G.711. It uses two encoding methods. ulaw in North America and **alaw** in the rest of the world . It transmits 8 bits 8,000 times per second, or 64,000 bits per second . Compared to a CD quality that encodes 16 bits with a sampling rate of 44,100 Hz, it is indeed a great compromise in quality requirements.

Many people say that G.711 is a non-compression codec. It's not entirely true if we consider coding to be a form of compression. This codec has become the basis for further codecs developed by it. It has minimal CPU requirements.

G.726

This codec has been in the world for some time (we don't see it much anymore) and it belongs to the original compression codecs. It is known as Adaptive Differential Pulse Width Modulation. (ADPCM). It works at several sampling rates from 16Kbps to 32 Kbps.

G.726 provides almost identical quality to G.711, but only with half the bandwidth. This is because instead of sending the result of the quantization measurement, it only sends information about the description of the differ-

ence between the previous and the current sample. G.726 was unsuccessful in not supporting modem and fax signals in the 1990s, but is experiencing a comeback due to low CPU usage.

G.729A

G.729A delivers surprisingly high-quality sound , given how much bandwidth it consumes . This is due to the use of Conjugate-Structure Algebraic-Code-Excited Linear Predictions (CS-ACELP).

CELP is a popular voice compression method. A codebook of sounds is built by mathematically modeling various human voices. Instead of sending the current voice sample, it sends the code of the corresponding voice sample. However, due to patents, we cannot freely use the codec, but we have to pay for its license. Yet it is very popular and very well supported by various phones and systems. To achieve an impressive compression ratio, the codec uses relatively much CPU processing power. Uses 8 Kbps bandwidth.

GSM

It provides similar parameters to G.729A except that it is available and usable from darma. Operates at 13 Kbps.

iLBC

The Internet Low Bitrate Codec provides an attractive mix of quality and bandwidth usage. Very well suited to use on lossy network lines. Since iLBC uses a complex of algorithms to achieve a high degree of compression, it puts a lot of strain on the CPU . It is a patent of Global IP Sound, it is enough to register.

Speex

Speex is a codec that can change the sampling rate according to network conditions. This is a GNU-licensed product . Speex operates somewhere at a transfer rate of 2.15 to 22.4 Kbps .

MP3

Moving Picture Experts Group Audio Layer 3 Encoding Standard. This codec is typically used for Music on Hold . Not directly a phone codec, it is optimized for music.[16]

2.4.1 Signaling Protocols

There are different signaling protocols used in the VOIP environment, such as IAX (The “Inter-Asterisk eXchange” Protocol), Session Initiation Protocol (SIP), H.323 and MGCP. The most popular and supported protocols are SIP and H.323.

The H.323 protocol SIP, both support voice over IP and multimedia communications, but the standards were developed by different standards-

setting bodies and have developed in different ways. As a result, both target multimedia transmitted over IP networks, but have different capabilities, which can be a strength or a weakness, depending on network operator or enterprise needs.

While both protocols were developed starting in 1996, SIP has become the VoIP and multimedia standard of choice over time and is being used by large hardware and software vendors, including Microsoft and Cisco. While few manufacturers are working on new H.323 implementations, it is still in use in many legacy systems, and some standards work continues.

H.323 protocol overview

H.323 is a binary-based standard developed by the International Telecommunication Union to support rich-media communications over IP networks that was initially focused on video conferencing, but now includes audio and video conferencing. H.323 is a well-defined and well-structured protocol, with specific definitions for establishing sessions, which can be loosely compared to calls, services and session components. Because of its rigid services definition that comes out of the telecom-based standards body, all H.323 implementations are generally interoperable.

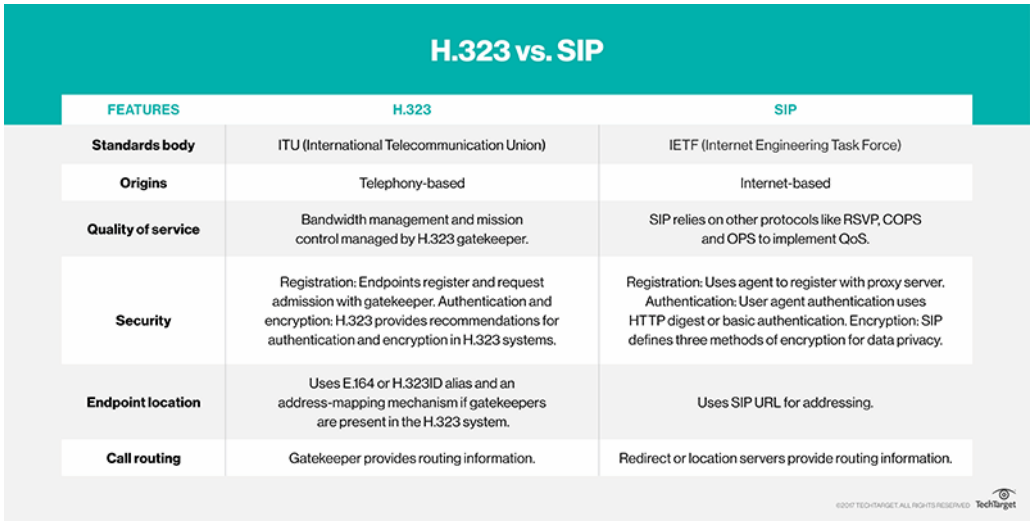
While H.323's rigid definition is an advantage in terms of interoperability, that rigidity can be its greatest challenge, because vendors are limited in their ability to layer additional features or services not supported by the protocol.

SIP overview

SIP was developed by the Internet Engineering Task Force. It was designed to set up a session between two points and to be a flexible component of the internet architecture. Unlike H.323, SIP's initial focus was voice communications, rather than video, but it was expanded to include video, application sharing, presence, instant messaging and other common communications applications.

SIP is an ASCII text-based standard leveraging much of the existing design of HTTP. SIP's text format, however, can result in large messages that aren't as suitable for networks that may have bandwidth, delay and processing issues. SIP is highly extensible, which allows developers to expand or add to its capabilities, and it supports rich-media communications,

as well as data transfer. SIP uses the Session Description Protocol (SDP) to define the characteristics of a session, which enables the use of encryption, transport protocol, the selection of voice and video codecs, and compression.



H.323 vs. SIP

FEATURES	H.323	SIP
Standards body	ITU (International Telecommunication Union)	IETF (Internet Engineering Task Force)
Origins	Telephony-based	Internet-based
Quality of service	Bandwidth management and mission control managed by H.323 gatekeeper.	SIP relies on other protocols like RSVP, COPS and OPS to implement QoS.
Security	Registration: Endpoints register and request admission with gatekeeper. Authentication and encryption: H.323 provides recommendations for authentication and encryption in H.323 systems.	Registration: Uses agent to register with proxy server. Authentication: User agent authentication uses HTTP digest or basic authentication. Encryption: SIP defines three methods of encryption for data privacy.
Endpoint location	Uses E.164 or H.323ID alias and an address-mapping mechanism if gatekeepers are present in the H.323 system.	Uses SIP URL for addressing.
Call routing	Gatekeeper provides routing information.	Redirect or location servers provide routing information.

Figure 2.8: SIP vs h.323 [1]

H.323 vs SIP

Unlike H.323, SIP leaves the specifics of feature implementations to developers, which gives them a great deal of flexibility when designing or using the protocol. SIP is also a simpler protocol than H.323, and it requires fewer messages to establish a session. H.323 requires a relatively large number of message exchanges to establish and manage sessions, but it is also highly reliable. Because of SIP's flexibility and extensibility, it quickly gained momentum among early vendors of IP telephony systems – particularly those offering platforms for hosted telephony services.

2.5 Communication Protocol

VoIP technology

This part introduces the basics of VoIP, a rundown on essential terms and the general workings of the technology. This part describes the basics of IP telephony and how VoIP calls get packetized and carried over external networks.

VoIP Voice over IP (Internet Protocols) allows to transfer multimedia data over IP networks using a set of protocols. These protocols include signaling protocols and multimedia data transfer protocols. Signaling protocols ensure registration, creation and termination of calls. The most widely used signaling protocols are SIP (Session Initiation Protocol), H.323, IAX2 (Inter-Asterisk eXchange). For multimedia data transfer, the RTP (Real Time Protocol) or its secure version of SRTP are the way to go. Usually UDP protocol is used on the transport layer. For connection to the PSTN legacy network, a so-called Getaway is used, which provides signaling and data conversion [2].

2.5.1 Session Initiation Protocol

Session Initiation Protocol (SIP) is a signaling protocol designed in 2002 by RFC 3261 [3]. This signaling protocol allows to build, modify, and end a session with one or more participants.

Features and characteristics

Nowadays, there are hundreds of other RFCs that either directly address or follow up SIP. As a rule, the Real-Time Protocol (RTP) is used for the transmission of multimedia data; Two basic SIP entities are defined for SIP functionality:

SIP is UTF-8 encoded client/server oriented text protocol. Nowadays, there are hundreds of other RFCs that either directly address or follow up SIP. As a rule, the Real-Time Protocol (RTP) is used for the transmission of multimedia data; Two basic SIP entities are defined for SIP functionality:

- **SIP User agent (UA)** - represents signaling end point.
- **SIP server** - contains SIP proxy, registrar, redirect server.

(A) SIP entity

(i) SIP User agent (UA)

A user agent is a signaling endpoint in SIP networks. The UA can be a hardware phone, a software client, a B2BUA, a PSTN gate, etc. The task of UA is to create a multimedia stream. User agent client (UAC) generates requests and receives responses, user agent server (UAS) receives requests and generates responses. UAS and UAC are logical entities only, and each terminal has both parts

(ii) SIP Proxy Server.
The task of the SIP proxy server is to redirect requests from the UA closer to the called party, usually to another proxy server. Proxy Server never generates SIP requests, only forwarding them to the called party.

(iii) Registrar server
The server receives SIP REGISTER messages from the UA, thus registering and updating the localization database.

(iv) Redirect server
Used to redirect. Receives search requests in the localization database and sends the user localization response back to the inquirer. [4]

(B) SIP messages

The communication between UA and SIP Servers is provided by SIP messages. They are divided into SIP REQUESTS, sometimes referred to as SIP METHODS, and SIP ANSWERS. Sip messages contain header and a payload. The first line of the header shows the message type, the header is separated by a free line from the message body. Below is the INVITE SIP request header:

Request-Line:INVITE sip:200@192.168.20.23;transport=UDP SIP/2.0

Via:SIP/2.0/UDP192.168.20.120:5060;branch=z9hG4bK-524287-1—3abf760e20a8cc74

Max-Forwards: 70

Contact: <sip:100@192.168.20.120:5060;transport=UDP>

To: <sip:200@192.168.20.23;transport=UDP>

From: <sip:100@192.168.20.23;transport=UDP>;tag=817a2873

Call-ID: ZA1hDnv44N7u6vpTR1QwDg..

CSeq: 2 INVITE

Allow: INVITE, ACK, CANCEL, BYE, NOTIFY, REFER, MESSAGE, OPTIONS, INFO, SUBSCRIBE

Content-Type: application/sdp

Supported: replaces, norefersub, extended-refer, timer, outbound, path, X-cisco-serviceuri

User-Agent: Z 3.9.32144 r32121

Allow-Events: presence, kpml

Content-Length:243

SIP methods

UAC generates SIP methods - requests, they ask for UAS a certain action. Six basic applications were defined in the original RFC 3261.

- INVITE - request to join or modify an existing connection
- ACK - final confirmation of the last response to the INVITE request
- BYE - request to terminate the connection
- REGISTER - register or unregister request
- CANCEL - request for termination during connection setup
- OPTIONS - Determine the counterparty properties

With the gradual development of VoIP, other SIP methods that bring new features have been introduced:

- PRACK - performs the same function as the ACK, except that it confirms temporary responses. This ensures reliable delivery.
- UPDATE - Allows you to change session parameters (stream codecs) without having to wait for the INVITE and re-INVITE methods to complete.
- REFER - call transfer request. The REFER method assures that the recipient will contact the next UA with the request to establish a connection.
- MESSAGE - sending short messages
- SUBSCRIBE, NOTIFY - logging and reporting node change statuses
- INFO - transmit signaling information during a call. [5]

SIP response codes

Each method must be answered. The only exception is the ACK method, which confirms the delivery of the last response to the INVITE request. The answers are divided into 6 groups and are marked with a three-digit code[6]:

• 1xx - temporary informative answers (for requests that have been received but not yet known) eg 100 (Trying) • 2xx - positive final answer (200 OK)
• 3xx - Redirect
• 4xx - negative response 401 (Unauthorized)
• 5xx - Server Side Error 503 (Service Unavaiable)
• 6xx - global error

(C) SIP communication

In the simplest configuration, it is possible to use two SIP entities, they communicate with each other by means of SIP messages, but usually the Private Branch Exchange (PBX) is also present in the SIP network. Usually, the control panel works in a proxy mode, forwarding the sip request closer to the client and passing the multimedia data directly between the end entities. A special case of communication is B2BUA. When two connections are always established between the client and the PBX and the multimedia data passes through the PBX.

SIP Proxy Server

An example of proxy server communication is shown in Figure below.

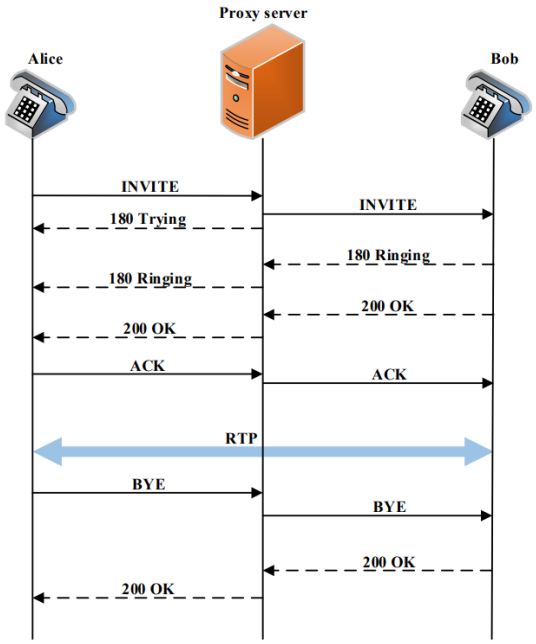


Figure 2.9: SIP message exchange - PROXY

In the domain managed by one SIP Proxy server, there are 2 clients, Alice and Bob. If Alice wants to contact Bob, she must send an INVITE request to the proxy server with a URI of bob@domain.com. The Sip server responds with a 100 Trying message and searches the localization database for Boba's IP address and forwards the INVITE request to it. Bob confirms the message by sending a 180 Ringing reply. If the call is accepted, a 200 OK Alert message is sent. Alice also responds with a 200 OK message to establish a connection and send RTP data. If Alice wants to end the call, she sends an BYE message, Bob confirms the call with 200 OK [7] .

B2BUA communication

Another way to communicate is by using B2BUA, a special case of UA that is inserted between AUs. This special UA is the signaling endpoint. Where one connection is created between Alice and B2BUA and the other between Bob and B2BUA, unlike the Proxy server, which only forwards the communication. Usually, RTP data passes through B2BUA (PBX) and thus has full control over the ongoing call. However, the performance of the PBX using this mode is worse than the proxy mode, due to the need of forwarding RTP packets. This solution is used by Asterisk. However, this PBX offers the ability to send RTP data directly between end points.

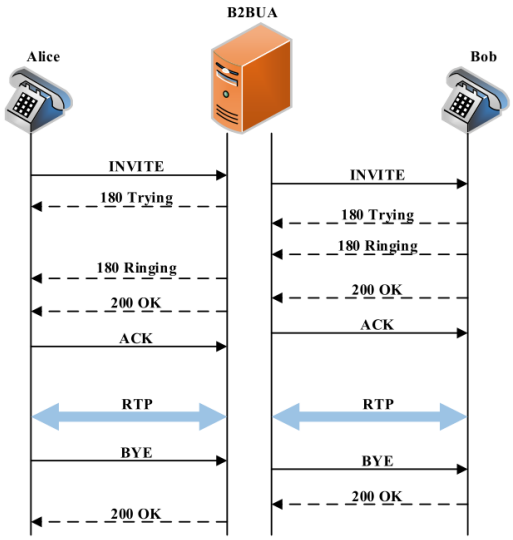


Figure 2.10: SIP message exchange - B2BUA

2.5.2 Real Time Protocol

In packet networks, Real-Time Transport Protocol (RTP) is most commonly used for voice transmission, allowing media to be transmitted through packet networks. The protocol is designed to minimize the delays and downtime of the transmitted audio or video signal. This protocol was issued in 1996 as RFC 1889 and was later replaced by RFC 3550.

The transmission of media over packet networks is very problematic and must take into account any possible packet delay, transmission delay over the IP network, and a delay in the buffer used by RTP.

By definition, RTP uses a so-called jitter buffer, which collects packets with transmitted media and plays them back in time to compensate for possible delays due to different transmission paths across the IP network. The total transmission delay includes network propagation time, buffer delay, and packet packet time. According to the ITU (International Telecommunication Union), the delay time should not exceed 150 ms, otherwise the conversation between the users won't be understandable, due to an important delay.

The RTP stream packet is supplemented with the RTP header and RTP data. The header includes information about the data type, the sequence number sent, the data timeline, and more. RTP is always transmitted on an even port and is transmitted by UDP. It is recommended to use Real-time Transport Control Protocol(RTCP), which broadcasts on the next port that RTP uses. During transmission, stream quality information is transmitted every 5 to 10 seconds via RTCP.

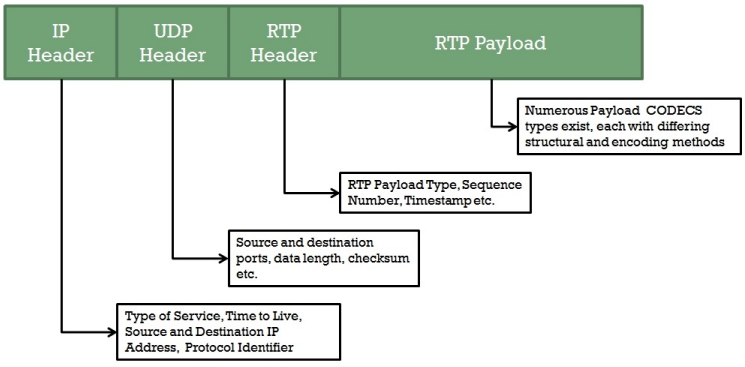


Figure 2.11: RTP Packet Structure[12]

3 Implementation

3.1 Available tools and solutions

As explained before, there is a number of requirements to satisfy in order to be a as reliable as it could be. One interesting approach is to use a soft Private Branch Exchange, that will act as a SIP server for the clients connected to it, and will act as the brains that will contain the necessary information for the deployet of the whole system, such as the number of clients available, the protocol used as so on.

There is a number of such solution available for use , such as 3CX, which is popular among small businesses, but it is not free, and obliges the user to choose between a fully featured version, or a free licence with a lot of limitations.

Another possible solution worth mentioning is Mumble, a private software turned public and free as part of a new community driven project. This project is implemented according to a server/client architecture but it is more of a push-to-talk solution. Even though during testing, it has been configured to auto-start and , it still pops up a dialogue box asking for confirmation before connecting to server, which is inconvenient and and complicated for average user.

A software package called OpenOB 3.0 is also available for testing purposes, as it is still in alpha2 stage, which means that it is still full of bugs and is not 100% compatible with all of the platforms, mainly issues around bus speeds for embedded and single-board computers. This library offers a number of codecs such as Opus and relies on GStreamer media framework for the underlying audio transport.

I would opt for FreePBX, because of the cost (Free) and it can be customised to suit any needs that might crop up.

3.2 Comunication Server

In this project, Asterisk has been selected among a couple of available IP/PBX solutions, Asterisk is an open source framework for building communications applications. Asterisk turns an ordinary computer into a communications server. Asterisk powers IP PBX systems, VoIP gateways, conference servers and other custom solutions. It is used by small businesses,

large businesses, call centers, carriers and government agencies, worldwide. Asterisk is free and open source.

Asterisk supports a variety of modes such as :

- VoIP getaway
- PBX
- IVR interactive voice guide
- Conference server
- Encryption medium
- Number translator
- And more..

3.2.1 Structure of Asterisk

The Asterisk system is designed to create an interface between a telephone application and a telephone interface (VoIP, TDM). With the SIP, IAX, H323 protocols, Asterisk works like channels associated with the kernel. A powerful tool for controlling Asterisk is the CLI command line as well as the Interface Manager. The cornerstone is Dialplan, which takes care of transferring incoming and outgoing calls or service call requests,see fig. 3.1 [6].

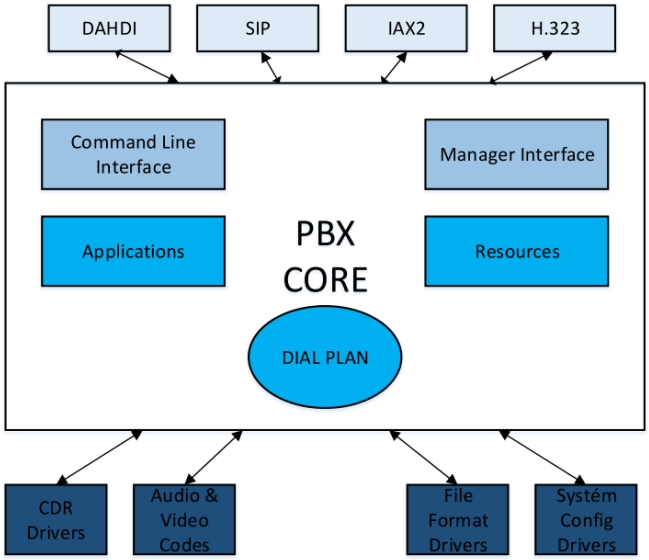


Figure 3.1: Asterisk architecture

The correct function of each protocol is assured by the so-called channel driver (STACK driver). There are several stacks in Asterisk, the name of the driver is associated with a protocol name such as chan_sip for SIP or chan_iax2 for IAX2. From Asterisk version 12, it is possible to use the two channel drivers for the SIP protocol, the original chan_sip and the new res_pjsip using the PJSIP library. In the following chapters both stacks are going to be described.

3.2.2 Asterisk on Raspberry Pi

RaspPBX is a project which brings the free and open source Asterisk and FreePBX into Raspberry Pi board. RaspPBX turns Pi into a communications server which can be used by small businesses with up to 12 extensions. FreePBX is a web-based open source GUI that controls and manages Asterisk. Our goal is to show installation of the latest RaspPBX into Raspberry Pi 3 Model B Rev 1.2.

The latest image available for download includes Asterisk 13.20.0 and FreePBX 14.0.2.10.

3.2.3 Configuration and installation of RasPBX server

The installation is easy and doesn't require an advanced user, most important steps are the following:

1. Flashing the SD Card with the downloaded ISO image
2. Boot the system and configure hostname / IP address
3. Choose your timezone,configure locale settings
4. Changing passwords using the **passwd** command

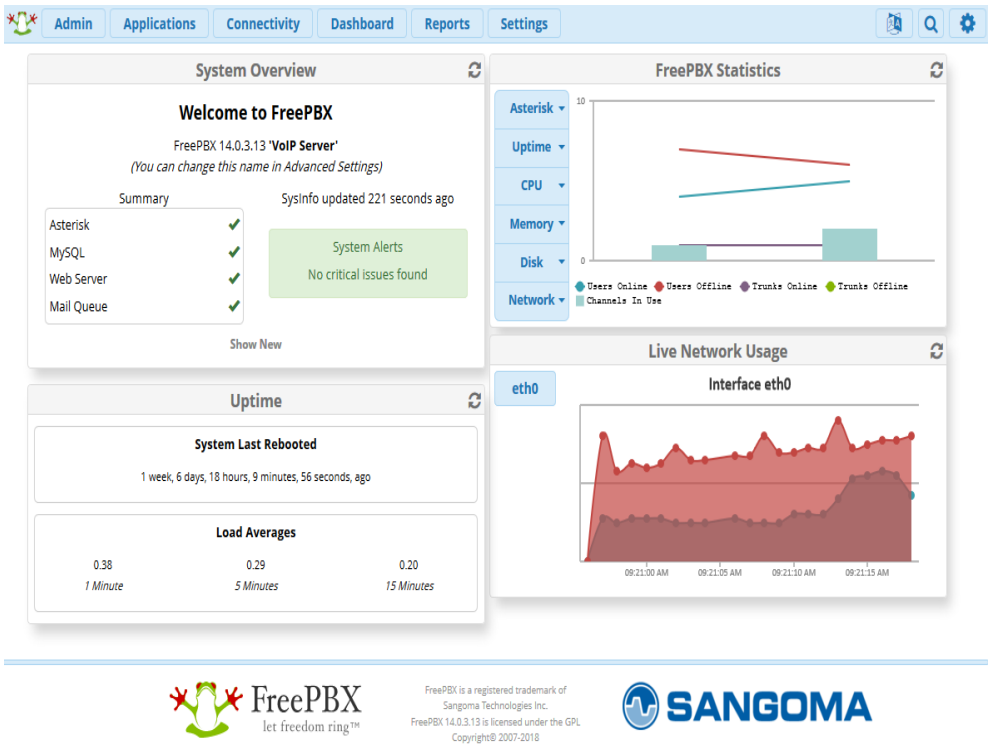


Figure 3.2: RasPBX user web interface

The most important is probably to go to Module Admin, checking online for module updates and apply them all by marking the necessary changes and hitting Process. It will take a few goes to fulfill all dependencies.

3.2.4 Native implementation of SIP protocol

The original sip channel driver was created in 2002. At that time, the SIP protocol was a relatively young standard, RFC 3261 was released and it was not clear what position the new protocol would build in the VoIP field. The original SIP stack was designed as a single channel and monolithic named chan_sip.

The chan_sip configuration is done in the sip.conf text file, which contains the global SIP settings section. This section is preceded by [general]. Furthermore, the name or account number in square brackets begins with the setting section for individual accounts, for example: [100]. Below is a basic example. All parameters that can be set for the sip protocol or individual accounts are listed in the literature [13].

General Settings:

```
[general]
context=all           ;context to which users belong
bindaddr=0.0.0.0:5060 ;The IP address and port on which Asterisk listens
srvlookup=yes         ;DNS permission
transport=udp         ;enable transmission protocol
disallow=all          ;disables all codecs
allow=alaw            ;allow only ALAW codec
language=en           ;default language
```

Client Settings:

```
[100]                 ;account name
type=friend           :account type
context=kontext1      ;to which context the client belongs
secret=password       ;password
host=dynamic          ;enable registration on all IP addresses
disallow=all          ;disabling all codecs
allow=alaw            ;allow only ALAW codec
```

3.2.5 PJSIP STACK

With the development of VoIP over time, SIP has also evolved , and a number of new enhancements have been made in the form of new RFCs. Thus, modifying the original chan_sip driver was increasingly challenging, and for this reason, the AstriDevCon 2012 development community decided to implement a new stack. The challenge was not only to design a new set of modules for the SIP protocol for today's needs, but there was also an effort to make it flexible for future adjustments. After analyzing various sip libraries, it was decided to implement a new stack based on the PJSIP libraries sponsored by Teluu. The PJSIP stack is implemented in Asterisk since version 12 in 2014 [14]

Reasons for selecting PJSIP:

- Written in C ++ which made it easier to implement it into Asterisk
- It is widely used and is used by many applications, test proved
- It is still maintained
- Developers previous experience with PJSIP from the Asterisk SCF project

The development of PJSIP is mainly focused on having small footprint, modular, and very portable SIP stack for embedded development purpose (although it's perfectly good for Win32/Linux/MacOS as well). Some of the characteristics of PJSIP:

it is built on top of PJLIB, and since PJLIB is a very very portable library, basically PJSIP can run on any platforms where PJLIB are ported (including platforms where normally it would be hard to port existing programs to, such as Symbian and some custom OSes). it has quite small footprint, although probably it's not the smallest SIP stack on the planet, it is quite customizable and modular, meaning that features that are not needed won't get linked into the executable, it has pretty good performance (thousands of calls per second), and it has quite a lot of SIP features. A high level SIP multimedia user agent API is available for both C and Python language.

3.2.6 PJSIP Architecture

A PJSIP project consists of several separate libraries which are responsible for different features. Basic PJSIP architecture for the client application can be seen on the figure below.

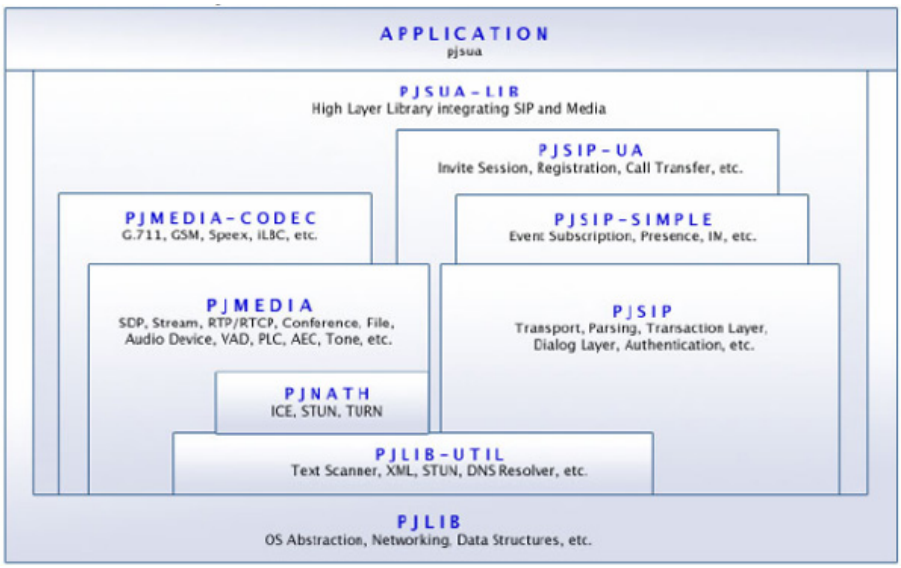


Figure 3.3: PJSIP architecture [15]

3.3 Client

Operating system Installation

There are several operating systems optimized for RPi. Raspbian - Based on Debian distribution, is the recommended option as this is the most popular OS for RPi. An operating system is going to be installed on the clients boards, however, as explained before, the server unit will be running RasPBX operating system instead, which is a complete OS package, based on Raspian, and contains the needed libraries and packages needed for the server to work as expected, added to that, it comes pre-configured and running a web-server, that acts as the configuration tool for the server, and makes the clients management easier to full fill.

Client interface

The client will be implemented in Python language, using SIP SIMPLE client SDK [17] , by AG-PROJECTS based on the PJSIP project version 2.8. PJSIP is a free and open source multimedia communication library written in C language implementing standard based protocols such as SIP, SDP, RTP, STUN, TURN, and ICE. It combines signaling protocol (SIP) with rich multimedia framework and NAT traversal functionality into high level API that is portable and suitable for almost any type of systems ranging from desktops, embedded systems, to mobile handsets.

PJSIP is both compact and feature rich. It supports audio, video, presence, and instant messaging, and has extensive documentation. PJSIP is very portable. On mobile devices, it abstracts system dependent features and in many cases is able to utilize the native multimedia capabilities of the device.

The idea is to implement a system made of a number of clients, that are raspberry pi based, and a server RasPBX that will have the role of the central point where clients must connect to in order to communicate to each other throught a switch or a router.

Every client in the system will be equipped with a usb microphone as an input and as an output a headset or small speaker, connected via audio JACK slot. This will act as ring notifier too.

The client is implemented in PYTHON programming language using the use of the SIP SIMPLE library from AG-PROJECTS.

The client is a script in python that will be running in the background waiting for call request from other client registered against the RasPBX registrar.

SIP proocol is the signaling protocol that is going to be used to exchange the request and reply messages explained above, and RTP protocol is the protocol used for voice packet transfert between the users.

The user interface are implemented using the tkinter module in Python.

Here is a code snippets of the client in both states, incoming and outgoing session:

Incoming session

```
adigeo@ag-blink:~$sip-audio-session
Using account 31208005169@ag-projects.com
Logging SIP trace to file "/Users/adigeo/Library/Application
Support/Blink/logs/sip_trace.txt"
Logging PJSIP trace to file "/Users/adigeo/Library/Application
Support/Blink/logs/pjsip_trace.txt"
Available audio input devices: None,
 system_default,
 Built-in Input,
Built-in Microphone
Available audio output devices: None,
system_default, Built-in Output
Using audio input device: Built-in Microphone
Using audio output device: Built-in Output
Using audio alert device: Built-in Output
Available control keys:
 s: toggle SIP trace on the console
 j: toggle PJSIP trace on the console
 n: toggle notifications trace on the console
 p: toggle printing RTP statistics on the console
 h: hang-up the active session
 r: toggle audio recording
 m: mute the microphone
 i: change audio input device
 o: change audio output device
 a: change audio alert device
 <>: adjust echo cancellation
 SPACE: hold/unhold
 Ctrl-d: quit the program
 ?: display this help message
2009-08-25 16:37:12 Registered contact
"sip:hxsyungk@192.168.1.124:59164" for
```

```
sip:31208005169@ag-projects.com
at 81.23.228.150:5060;transport=udp (expires in 600 seconds).
Other registered contacts:
 sip:31208005169@192.168.1.123:5060 (expires in 274 seconds)
 sip:kwbfxyvl@192.168.1.124:59116 (expires in 522 seconds)
 sip:ilmegvkp@192.168.1.124:59003 (expires in 339 seconds)
 sip:31208005169@192.168.1.1;uniq=5B2860C44383A3D6705629A7E1FB8
(expires in 1162 seconds)
Detected NAT type: Port Restricted
Incoming audio session from 'sip:adi@umts.ro',
 do you want to accept?
(y/n)
Audio session established using "speex" codec at 16000Hz
Audio RTP endpoints 192.168.1.124:50378 <-> 85.17.186.6:58868
RTP audio stream is encrypted
Remote SIP User Agent is "Blink-0.9.0"
Remote party has put the audio session on hold
Audio session is put on hold
Audio session ended by remote party
Session duration was 6 seconds
2009-08-25 16:37:44 Registration ended.
```

Outgoing session

```
adigeo@ag-blink:~$sip-audio-session -a umts ag@ag-projects.com
Using account adi@umts.ro
Logging SIP trace to file "/Users/adigeo/Library/Application
Support/Blink/logs/sip_trace.txt"
Logging PJSIP trace to file "/Users/adigeo/Library/Application
Support/Blink/logs/pjsip_trace.txt"
Available audio input devices: None, system_default,
 Built-in Input,
Built-in Microphone
Available audio output devices: None, system_default,
 Built-in Output
Using audio input device: Built-in Microphone
Using audio output device: Built-in Output
Using audio alert device: Built-in Output
Available control keys:
 s: toggle SIP trace on the console
 j: toggle PJSIP trace on the console
```

```
 n: toggle notifications trace on the console
 p: toggle printing RTP statistics on the console
 h: hang-up the active session
 r: toggle audio recording
 m: mute the microphone
 i: change audio input device
 o: change audio output device
 a: change audio alert device
 <>: adjust echo cancellation
 SPACE: hold/unhold
 Ctrl-d: quit the program
 ?: display this help message
Initiating SIP audio session from 'sip:adi@umts.ro'
to 'sip:ag@agprojects.com' via sip:85.17.186.7:5060;transport=udp...
Audio session established using "speex" codec at 16000Hz
ICE negotiation succeeded in 1s:412
Audio RTP endpoints 192.168.1.124:50852 (ICE type host) <->
192.168.1.124:50871 (ICE type host)
RTP audio stream is encrypted
Audio session is put on hold
Remote party has put the audio session on hold
Detected NAT type: Port Restricted
Ending audio session...
Audio session ended by local party
44
Session duration was 7 seconds
```

Configuration

Sip Simple middleware uses command line tools for configurating the sip clients and managing functions such as Register againt the sip server, but in order to work as intented, it is critical to make sure all dependencies are installed correctly as well as the needed SDK:

SIP SIMPLE Client SDK

darcs get –set-scripts-executable http://devel.ag-projects.com/repositories/python-sipsimple

Dependencies

Several dependencies provided by AG Projects can be accessed in the

same way:

darcs get http://devel.ag-projects.com/repositories/python-xcaplib darcs get http://devel.ag-projects.com/repositories/python-msrplib
darcs get http://devel.ag-projects.com/repositories/python-application
darcs get http://devel.ag-projects.com/repositories/python-backports
darcs get http://devel.ag-projects.com/repositories/python-gnutls
darcs get http://devel.ag-projects.com/repositories/python-cjson
darcs get http://devel.ag-projects.com/repositories/python-greenlet
darcs get http://devel.ag-projects.com/repositories/python-eventlib
darcs get http://devel.ag-projects.com/repositories/python-otr

Middleware API

This section describes the Middleware API for SIP SIMPLE client SDK that can be used for developing a user interface (e.g. Graphical User Interface). The Middleware provides a non-blocking API that communicates with the user interface asynchronously by using Notifications.

For its configuration, the Middleware uses the Configuration API.

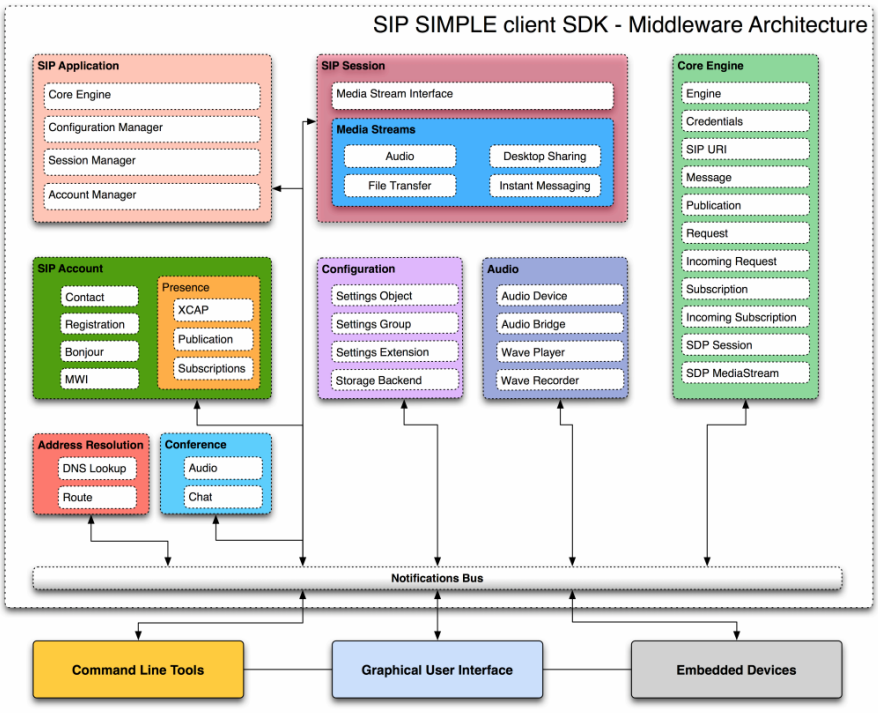


Figure 3.4: SIP SIMPLE client SDK[18]

Middleware Methods

sip-register

Implemented in sipclients/sip-register
This script is used to Register a SIP end-point with a SIP Registrar or broadcast the local SIP URI using Bonjour mDNS.

Description

SIP protocol offers a discovery capability. If a user wants to initiate a session with another user, he must discover the current host(s) at which the destination user is reachable. To do this, SIP network elements consult an abstract service known as a location service, which provides address bindings for a particular domain. Registration entails sending a REGISTER request to a special type of UAS known as a registrar. A registrar acts as the front end to the location service for a domain, reading and writing mappings based on the contents of REGISTER requests. This location service is then typically consulted by a proxy server that is responsible for routing requests for that domain. This script implements REGISTER method, which registers the contact (ip, port and transport) for a given address of record (SIP address).

sip-audio-session

Implemented in sipclients/sip-audio-session Setup a single SIP audio session using RTP/sRTP media. Description This script can be used for interactive audio session or for scripting alarms. The script returns appropriate shell response codes for failed or successful sessions. The script can be setup to auto answer and auto hangup after predefined number of seconds, detects SIP negative response codes, missing ACK and the lack of RTP media after a session has been established. Once the media stream is connected, the outcome of the ICE negotiation and the selected RTP candidates are displayed.

sip-session

Implemented in sipclients/sip-session
Setup one or more SIP sessions with Audio (RTP/sRTP

Description

sip-session command line script is a show-case for the powerful features of

SIP SIMPLE development kit related to establishing, modifying and terminating SIP sessions with multiple media types like VoIP, Instant Messaging and File Transfer. The script has the following features:

1. Registers with a SIP registrar and is available for incoming sessions
2. Switches between multiple sessions and provides in-call controls like Hold and Mute
3. Handles outgoing SIP sessions with combinations of media types based on RTP and MSRP protocols
4. Performs NAT traversal using ICE and MSRP relay extension
5. Provides control for the input, output and alert audio devices
6. Records the RTP audio streams (input, output or combined)
7. Enable text input and output for Instant Messaging sessions
8. Provides File Transfer capability with progress indicator
9. Gives access to real-time traces of involved protocols (DNS, SIP and MSRP)

GUI

Based on the previously explained methods and function, the GUI application is to be implemented in Python language, using the tKinter package.

Tkinter is Python's de-facto standard GUI package. It is a thin object-oriented layer on top of Tcl/Tk. Tkinter is not the only GuiProgramming tool-kit for Python. It is however the most commonly used one.

The GUI is just a simple user interface designed to dial an extension number using numbered button from 0 to 9, initiate a call, hang-up, finish a call and cancel the dialled number in case of a mistake.

4 SIP Intercom User Quickstart Guide

4.0.1 Installing RasPBX

This guide provide a simple setup sample for getting started using RasPBX SIP Server

1. **Download, Extract and Copy RaspPBX Image to SD Card**

   $ wget http://download.raspberry-asterisk.org/raspbx-04-04-2018.zip
   $ sudo dd bs=4M if=raspbx-04-04-2018.img of=/dev/mmcblk0 status=progressconv=fsync
   $ unzip raspbx-04-04-2018.zip

2. **Expanding filesystem on SD card**

   By default the image utilizes only 4GB of your SD card space. Login to the console with username root and password raspberry and issue the command below.

   Navigate to Advanced Options-> A1 Expand Filesystem Ensures that all of the SD card storage is available to the OS. The filesystem will be enlarged upon the next reboot.

3. **Configure Static IP Address** Set static IP address for interface eth0.

   # echo "interface eth0" » /etc/dhcpcd.conf
   # echo "static ip_address=172.17.100.50/16" » /etc/dhcpcd.conf
   # echo "static routers=172.17.100.1" » /etc/dhcpcd.conf
   # echo "static domain_name_servers=172.17.100.1 8.8.8.8" » /etc/dhcpcd.conf

4. **Set timezone** We need to configure the correct timezone # dpkg-reconfigure tzdata

Figure 4.1: Setting Timezone for RasPBX

5. **Upgrade RasPBX**

   Asterisk is supplied by RasPBX repositories, use raspbx-upgrade to get updates. The upgrade process takes more than 1 hour. You can use ssh to login to RasPBX with the default username root and password raspberry.

   # **raspbx-upgrade**

   After the upgrade, check the version of Asterisk with the command below.

   # **asterisk -rx 'core show version´**

```
root@raspbx:~#
root@raspbx:~# asterisk -rx 'core show version'
Asterisk 13.22.0 built by root @ raspbx on a armv6l running Linux on 2018-07-15
19:47:36 UTC
root@raspbx:~#
```

Figure 4.2: Asterisk Version is 13.22.0 After Upgrade

6. **FreePBX Initial Setup**

   Create Account for FreePBX Administration

   Enter address http://172.17.100.50 into your web browser. Fill the password and click Create Account.

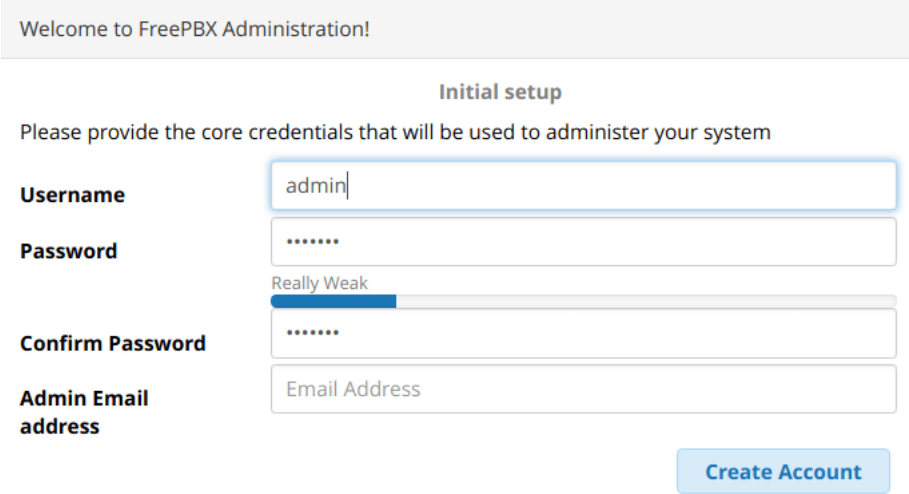


Figure 4.3: Creating New Account for Web Administration

Once account is created, click FreePBX Administration. Login window will appear.

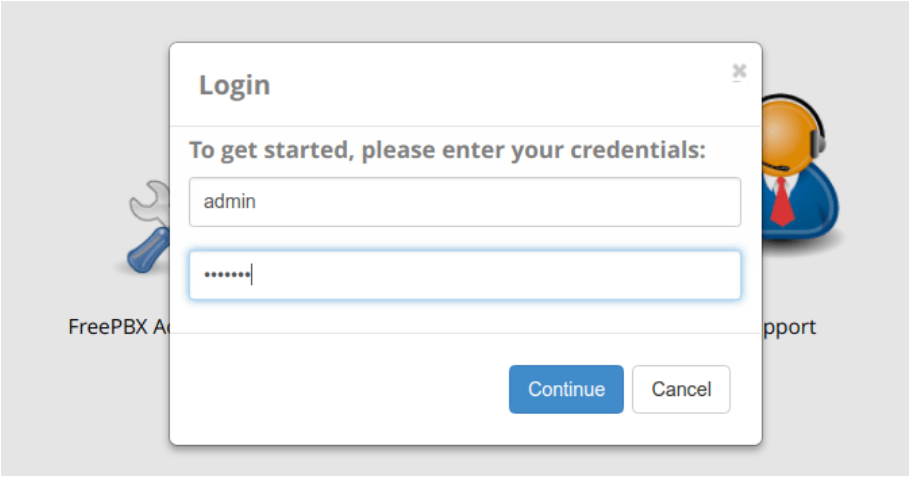


Figure 4.4: Login to Web Interface PBX with New Password

Select Default Locales

After login, initial configuration wizard starts. We need to configure system language and timezone for FreePBX.

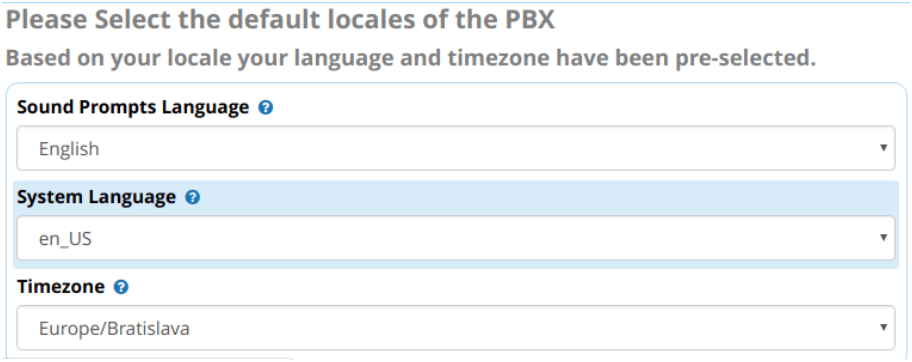


Figure 4.5: Selecting System Language and Timezone

Adding SIP Extensions to FreePBX We are going to create two chan_sip extensions 1010 and 1020 in order to test local call between phones registered to RasPBX. Our dial plan consists of the pattern 1XXX that we will assign to the extensions registered to our RaspPBX. Navigate to Applications-> Extensions. Click Add Extension and select Add New Chan_SIP Extension

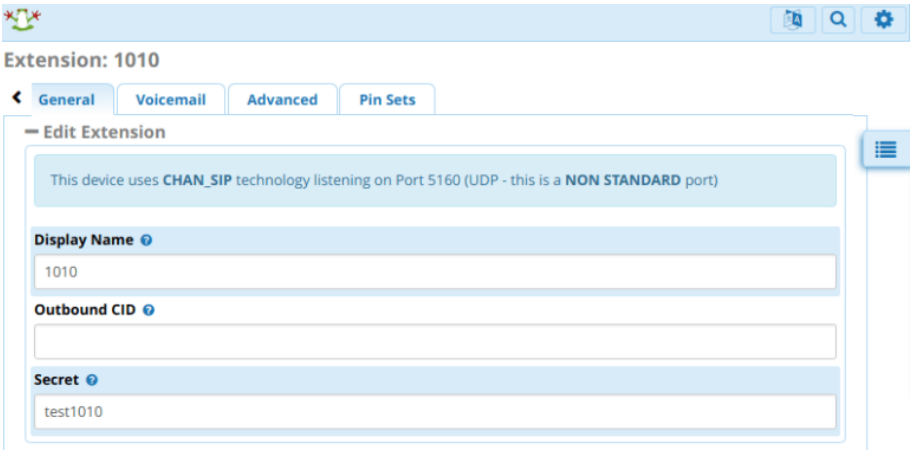


Figure 4.6: Adding Chan_Sip Extension 1010

Create the second extension 1020

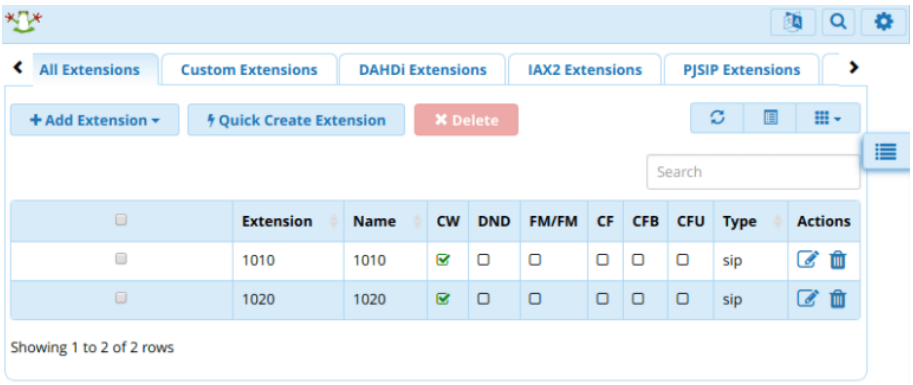


Figure 4.7: Chan_SIP Extensions 1010 and 1020

Configure Client/Softphone

Using command line interface, register the accounts on the different clients platforms (Raspberry Pi compute units)

pi@raspberrypi:~$ sip-register -a -j 1010@172.17.100.50:5160

```
pi@raspberrypi:~ $ sip-register -h
Usage: sip-register [options]

This script registers the contact address of the given SIP account to the SIP
registrar and refresh it while the program is running. When Ctrl+D is pressed
it will unregister.

Options:
  -h, --help            show this help message and exit
  -a NAME, --account=NAME
                        The name of the account to use. If not supplied, the
                        default account will be used.
  -c CONFIG_DIRECTORY, --config-directory=CONFIG_DIRECTORY
                        The configuration directory to use. This overrides the
                        default location.
  -s, --trace-sip       Dump the raw contents of incoming and outgoing SIP
                        messages (disabled by default).
  -j, --trace-pjsip     Print PJSIP logging output (disabled by default).
  -n, --trace-notifications
                        Print all notifications (disabled by default).
  -r MAX_REGISTERS, --max-registers=MAX_REGISTERS
                        Max number of REGISTERs sent (default 1, set to 0 for
                        infinite).
  -b, --batch           Run the program in batch mode: reading input from the
                        console is disabled. This is particularly useful when
                        running this script in a non-interactive environment.
pi@raspberrypi:~ $
```

Figure 4.8: Registering a new account against RasPBX

If you fail to register the client , you can troubleshoot registration by connecting to Asterisk console with the command.

asterisk -r

The Picture 4.9 shows the unsuccessful attempt to register SIP client configured as the extension 1010 when wrong password is entered.

```
raspbx*CLI>
[2018-09-10 09:20:10] NOTICE[1315]: chan_sip.c:28691 handle_request_register:
 Registration from '<sip:1010@172.17.100.50;transport=UDP>' failed for '172.1
7.100.2:53749' - Wrong password
[2018-09-10 09:20:12] NOTICE[1315]: chan_sip.c:28691 handle_request_register:
 Registration from '<sip:1010@172.17.100.50;transport=UDP>' failed for '172.1
7.100.2:53749' - Wrong password
[2018-09-10 09:20:16] NOTICE[1315]: chan_sip.c:28691 handle_request_register:
 Registration from '<sip:1010@172.17.100.50;transport=UDP>' failed for '172.1
7.100.2:53749' - Wrong password
[2018-09-10 09:20:24] NOTICE[1315]: chan_sip.c:28691 handle_request_register:
 Registration from '<sip:1010@172.17.100.50;transport=UDP>' failed for '172.1
7.100.2:53749' - Wrong password
raspbx*CLI>
```

Figure 4.9: Failed Attempt to Register Extension 1010 When Wrong Password is Provided

Testing Local Calls between Registered clients

Once we finished configuration of the both clients , we will try to establish call between extensions 1010 and 1020.

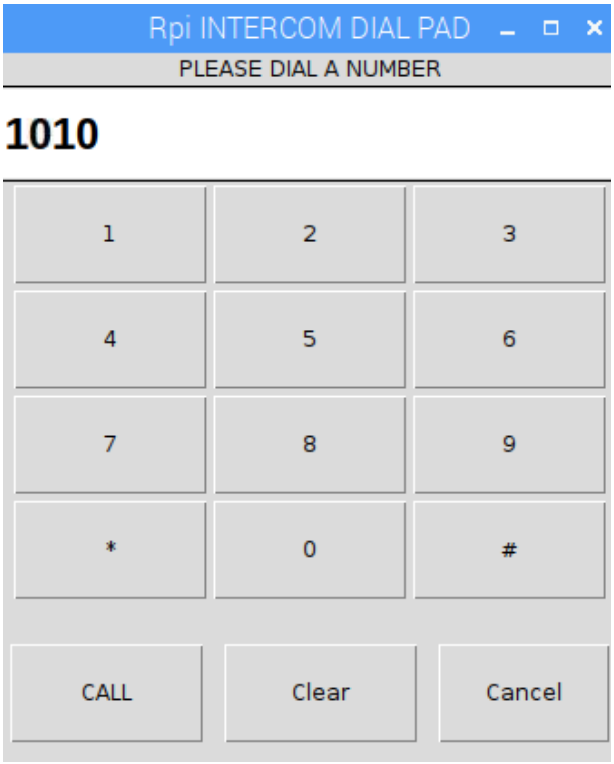


Figure 4.10: Dialing extension 1010 from the python GUI

Active calls can be found with the command below (Picture 4.10).

/usr/sbin/asterisk -rx 'core show channels'

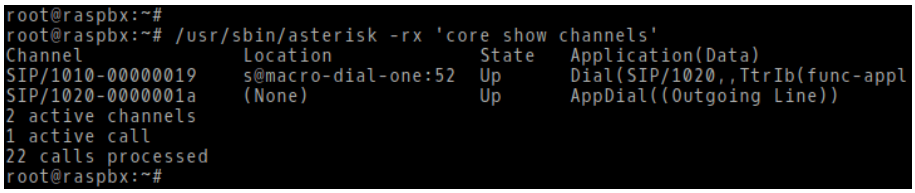


Figure 4.11: Checking Active Calls Using Asterisk Console

They are 22 calls placed at all with one active call from extension 1010 to 1020.

5 System optimisations and possible upgrades

Force RasPBX to Use Https for Secure FreePBX Administration

We will use the self-signed certificate. First, enable ssl mode.

a2enmod ssl

The file /etc/apache2/sites-available/default-ssl has the configuration for an http server. To enable ssl site default-ssl issue the command below.

a2ensite default-ssl

Restart apache server.

systemctl restart apache2

Secure login to FreePBX using https.

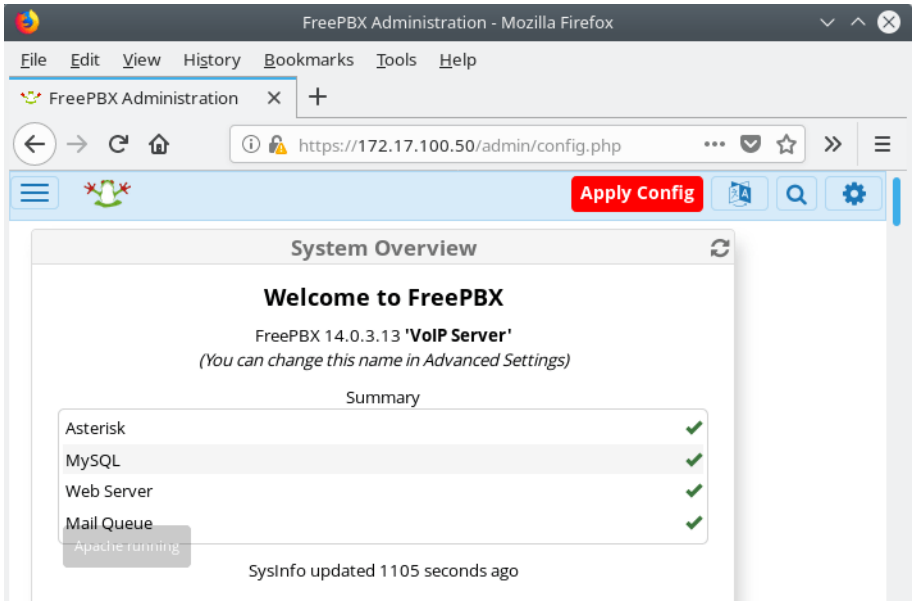


Figure 5.1: Using https for FreePBX Administration

Real time Video monitoring with motion detection

The raspberry pi is capable of video streaming to a web browser, and that is possible thanks to the camera module unit and a Real-time HTTP/HTTPS Streaming Server with the native uv4l-server module streaming server.

Among the other things, it offers a Web interface from which it's possible to see the video stream in various ways and a Control Page allowing to fully control the camera settings while streaming with any Video4Linux application. Other than secure HTTPS protocol, basic authentication for both the normal and admin users is also supported.

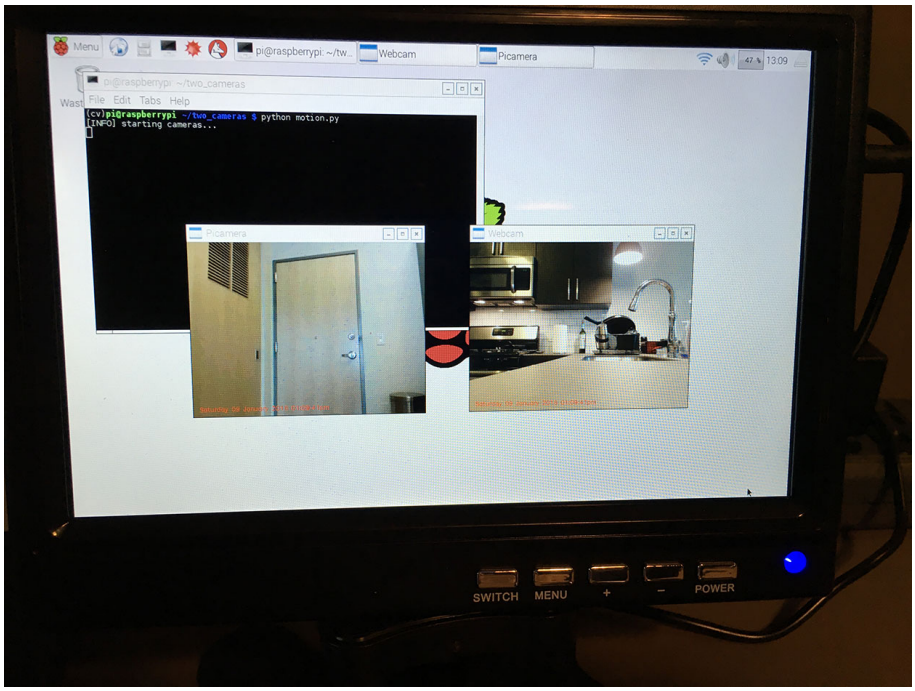


Figure 5.2: Two-way Video streaming using Raspberry Pi

Figure 5.3: Camera module connected to a Raspberry Pi

Another interesting application is the motion detection capability of the openCV library, which can turn the system to a home security device capable of image processing and alerting the owner of any suspicious behavior in the field of view of the camera while he is outside.

Alerting the user can be by sending email alerts, if the raspberry pi is connected to the internet and configured to send emails, or using a GSM module expansion hat, that enables the intercom to send sms notification or even call the owner and establish a live streaming session between both parties.

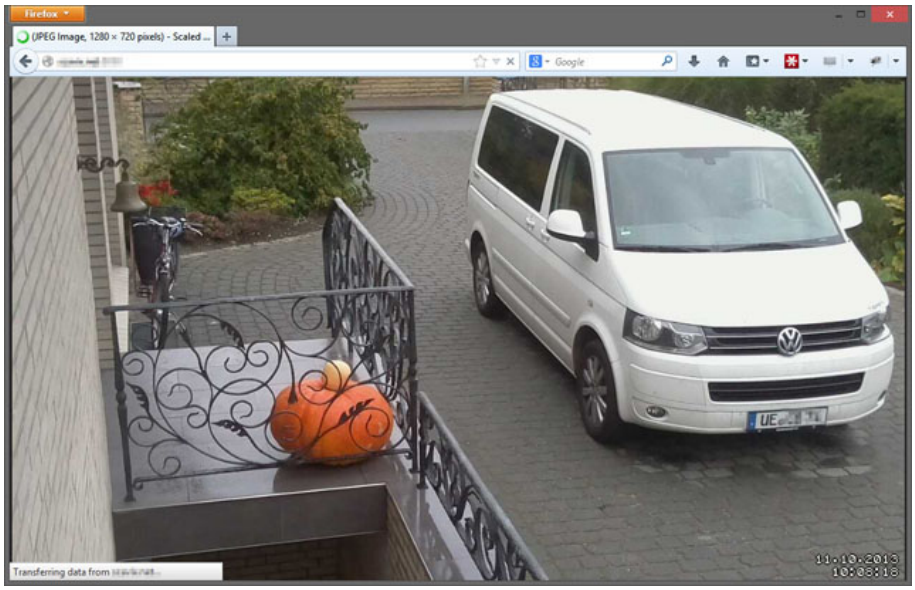


Figure 5.4: Raspberry Pi as a surveillance camera

Button/Keypad upgrade

The touchscreen used in this project can fail from time to time due to vandalism or severe outdoor climate, or even for hackers who can try to gain access to the brains of the system, so in order to make it more secure, a metal keypad can be a reasonable solution to secure the whole intercom system, which starts at the door step outside the residance od the user.

Figure 5.5: Antivandal outdoor keypad

SRTP instead of RTP

SRTP is not a transport, it is simply the encryption of the RTP to secure it, hence the S before RTP. The RTP is still transported in UDP but both parties to the call have exchanged keys in the SIP to enable encryption. You can use SRTP regardless of the transport used for the SIP as they are unrelated

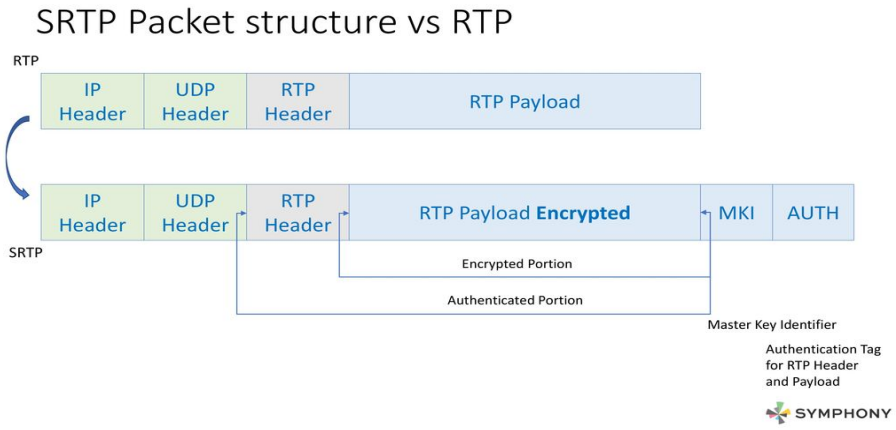


Figure 5.6: SRTP Packet structure vs RTP

RFID capability and access control

In order to implement this feature, a pretty simple setup composed of an RFID reader,the Rasperry Pi and a RFID tag is more than enought. When someone places an RFID card against the reader hidden behind a poster by their front door, the reader grabs the code and the Pi compares it with a list of authorized users. If the card is on the list, the Pi triggers the door lock using a signal line originally designed to work with an intercom system. If the user isn't on the list, access won't be granted.

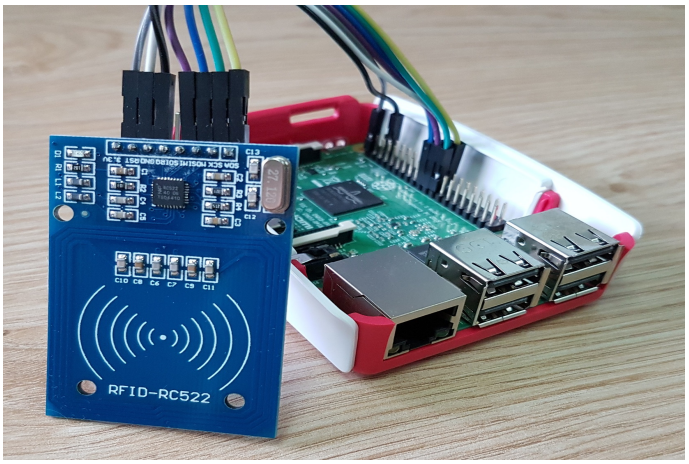


Figure 5.7: RC522 RFID Tag Reading with the Raspberry Pi [19]

Expand the boards with temperature sensor

An intersting module made by the company BigClown called CORE MODULE - NR, is a small board with a 32-bit ARM microcontroller, 192 kB of flash memor and 20 kB of RAM. Also, it has an integrated digital temperature sensor, and 3D accelerometer that can be a tool to know the outdoor temperature. This module is 100% comatible with the raspberry pi microcomputer, and can be directly connected via usb or GPIO pins, which means that it can get ower from the raspberry pi itself without the need for a separate energy source.

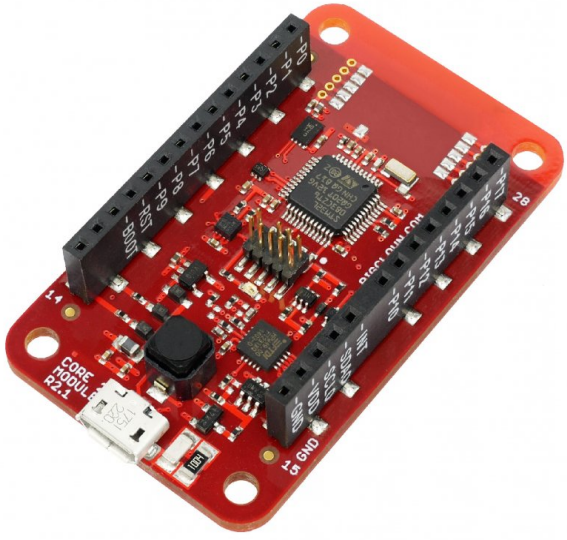


Figure 5.8: Core Module Board by Big Clown

6 Conclusion

PJSIP libraries is an ideal solution for the development of SIP client applications and don't bother about the SIP Background implementation. It doesn't contain full SIP server realization, but Server Application could be also built based on the PJSIP library API and all low layer possibilities it references. Such a way of development will avoid implementation of basic SIP features, and will require only creation of server specific functionality. That will save a lot of time and money. As a result, it will give an opportunity to develop a fully functioning, stable and secure application that will support the majority of usable platforms with relatively low efforts.

Bibliography

[1] I. Lazar . H.323 vs. SIP: What's the difference? [online] [cit. 2019-04-05] Available from: https://searchunifiedcommunications.techtarget.com/answer/Differences-between-H323-and-SIP

[2] P. HAGENDORF. XML Schema for Media Control: RFC 5168. 2008. [online] [cit. 2019-04-05] Available from: https://tools.ietf.org/html/rfc5168

[3] SIP: Session Initiation Protocol.[online][cit. 2019-04-05] Available online at: https://www.ietf.org/rfc/rfc3261.txt

[4] VOZŇÁK, Miroslav. Technologie a protokoly multimediálních komunikací pro integrovanou výuku VUT a VŠB-TUO. Ostrava: Vysoká škola báňská - Technická univerzita Ostrava, 2014.[cit. 2019-04-05] ISBN 978-80-248-3326-2.

[5] ČERVENKA, Marek. Asterisk 12 – New Age ROOT.cz [online]. 2014, , 1 . Available from: https://www.root.cz/clanky/asterisk-12-new-age/

[6] ŠILHAVÝ, PH.D, Ing. Pavel. Telekomunikační a informační systémy. BRNO, 2014. Vysoké učení technické v Brně Fakulta elektrotechniky a komunikačních technologií Ústav telekomunikací Technická 12, 616 00 Brno.

[7] The Session Initiation Protocol - The Internet Protocol Journal - Volume 6, Number 1. CISCO [online]. [cit. 2019-04-05]. Available here: http://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-23/sip.html

[8] Banana Pi official website [cit. 2019-04-05] [online] http://www.banana-pi.org/m1plus.html

[9] Orange Pi official website [cit. 2019-04-05] [online] http://www.orangepi.org/orangepipc/

[10] CubieBoardA series of open source hardware [cit. 2019-04-05] [online] http://cubieboard.org/2013/06/19/cubieboard2-is-here/

[11] MakeZine.com - One BeagleBone Black for All Kinds of Weekend Projects! [cit. 2019-04-05] [online] https://makezine.com/2014/12/06/one-beaglebone-black-for-all-kinds-of-weekend-projects/

[12] Chet Hosmer, Protocol Data Hiding ,March 06, 2012 [online]. [cit. 2019-04-05]. Available here: https://www.allencorporation.com/pdf/nm030612.pdf

[13] STELIOS ANTONIOU. VoIP Signaling Protocols: RFC 5168 [online]. 2010 [cit. 2019-04-05][online]. available here: https://www.pluralsight.com/blog/it-ops/voip-signaling-protocols

[14] Asterisk Versions. Asterisk.org [online]. Russell Bryant, 2016 [cit. 2019-04-05][online].available from: https://wiki.asterisk.org/wiki/display/AST/Asterisk+Versions

[15] ASTAPCHIK MARINA - USING PJSIP LIBRARY IN SERVER AND CLIENT APPLICATIONS 27-12-2012[online] [cit. 2019-04-05].available from: https://www.elinext.com/blog/using-pjsip-library-in-server-and-client-applications/

[16] Asterisk : The Future of Telephony by Jared Smith ; Jim Van Meggelen ; Leif Madsen, September 26, 2005 [cit. 2019-04-05] ISBN 978-05-960-0962-5

[17] SIP SIMPLE Client SDK by AG- PROJECTS 2018 [cit. 2019-04-05] [online] https://sipsimpleclient.org/developer-guide/

[18] SIP SIMPLE Client SDK by AG- PROJECTS 2018 [cit. 2019-04-05][online] available here: http://download.ag-projects.com/SipClient/SIPSIMPLE-Manual.pdf

[19] Matt, 25 February 25. RC522 RFID Tag Reading with the Raspberry Pi[cit. 2019-04-05][online] available here: https://www.raspberrypi-spy.co.uk/2018/02/rc522-rfid-tag-read-raspberry-pi/