

TZ

Setting up a VoIP communication between a Raspberry Pi and an IP phone

Technical report

Subject description: The goal of this project is to establish a telephone communication over IP between a raspberry PI and an IP telephone (Alcatel brand) via a local network. We will use the SIP protocol to establish this telephone communication between the two endpoints (IP phone + Raspberry) through an Asterisk PBX server.

Web article: <https://guillaume.nibert.fr/voip-asterisk-rpi-ipphone-project/>.

GitHub repository: <https://github.com/guillaumenibert/VoIP-Asterisk-WebRTC-SIP>.



This report is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/).

Guillaume Nibert

Supervisor: Dr. Ahmed Lounis

Change history

Date	Revision	Description
12/02/2021	1.0.0	First version of the report (only available in French)
03/02/2022	1.0.1-EN	English version, updated GitHub link and license

Table of contents

Table of contents	2
Introduction	4
1. SIP protocol and VoIP communication	6
2. Implementation of an Asterisk IP PBX server	8
Technology choices	9
Prerequisite	9
Installation of Asterisk	10
SIP configuration and user creation	20
SIP configuration - pjsip.conf	20
Dial plan - extensions.conf	22
3. Installation and configuration of a SIP client on the Raspberry Pi	24
Technology choices	24
Prerequisites	24
Installation and configuration of the Linphone SIP client	25
Client-side testing	25
Server-side testing	26
4. IP phone configuration	27
Configuration information	29
Configuring the phone in SIP mode - on the device	30
HTTP server installation	32
Transfer of configuration files and firmware to the Alcatel phone	33
Client-side testing	37
Server-side testing	37
5. Communication tests	39
Raspberry Pi to Alcatel IP Touch	39
Alcatel IP Touch to Raspberry Pi	41
6. Development of a JavaScript SIP client using WebRTC	43
WebRTC & WebSocket	43
Configuration du serveur Asterisk pour prendre en charge l'API WebRTC	45
Generating a self-signed SSL/TLS certificate	45
Enabling the Asterisk HTTP server	48
Editing pjsip.conf to support WebRTC	49
Editing extensions.conf to support WebRTC	52
Communication tests with the Web Browser Phone client	53
Development of a JavaScript SIP client	57

Conclusion	58
Table of illustrations (excluding appendixes)	59
Abbreviations (excluding appendixes)	60
Références	62
Attributions	64
Appendixes	65
Appendix A1 - Installation of a virtual machine under Debian 10 Buster	65
I - Preparation of the VM	65
II - Installation and configuration of Debian 10	69
Annexe A2 - Installation et configuration de Raspberry Pi OS Buster (64 bits) pour Raspberry Pi 3B+	79

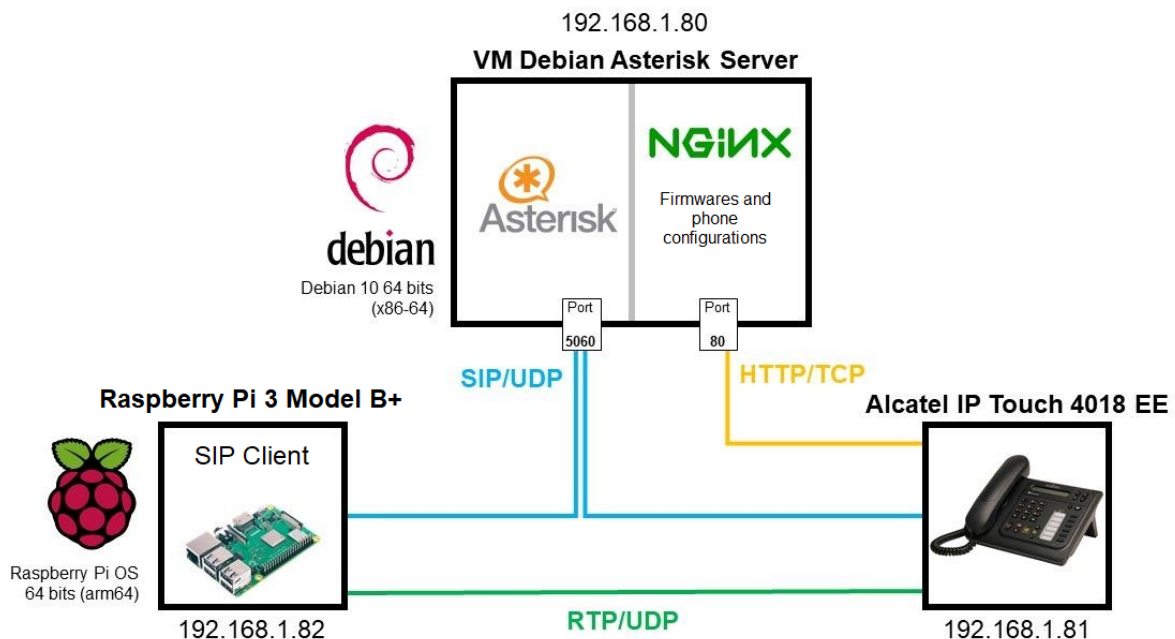
Introduction

Telegraph, analogue telephone, digital telephone, IP telephone with wired or wireless modes, telephony is currently a widely used and very convenient global link, indispensable for fast and real-time exchanges, for all purposes. It is quite reliable, not too expensive, and offers ever more extensive possibilities at an ever faster rate.

The aim of this project is to use the IP protocol used in the Internet, to make devices communicate with each other (commonly called endpoints). This project will therefore implement two endpoints with the following hardware characteristics:

- endpoint 1: Alcatel IP Touch 4018 EE phone;
- endpoint 2: Raspberry Pi 3 Model B+.

Below is a global diagram of the infrastructure architecture in the **192.168.1.0** network (*the router is not shown for readability reasons*):



(Figure 1 - Overall infrastructure diagram)

Simply, without going into detail, in this diagram there are two clients: the end points which are the *Alcatel IP Touch 4018 EE* and the *Raspberry Pi*. When there is a telephone communication between these two devices, the protocol used for the initiation of the communication is SIP (*Session Initiation Protocol*) for the application layer and UDP for the transport layer of the Internet protocol stack. This initiation goes through an intermediary: the

SIP initiation server (Asterisk¹) which is accessible on the Debian Asterisk Server virtual machine on port 5060 (*in blue*). Once the initiation is done, the two devices connect to each other directly to let the audio pass via the RTP protocol (*in green*).

An important point concerns the Alcatel phone part, which needs at each startup to check its firmware and configuration files allowing it to connect to the Asterisk server. These files are stored in the Debian virtual machine and are available to the Alcatel phone via the HTTP server on port 80 (*in orange*).

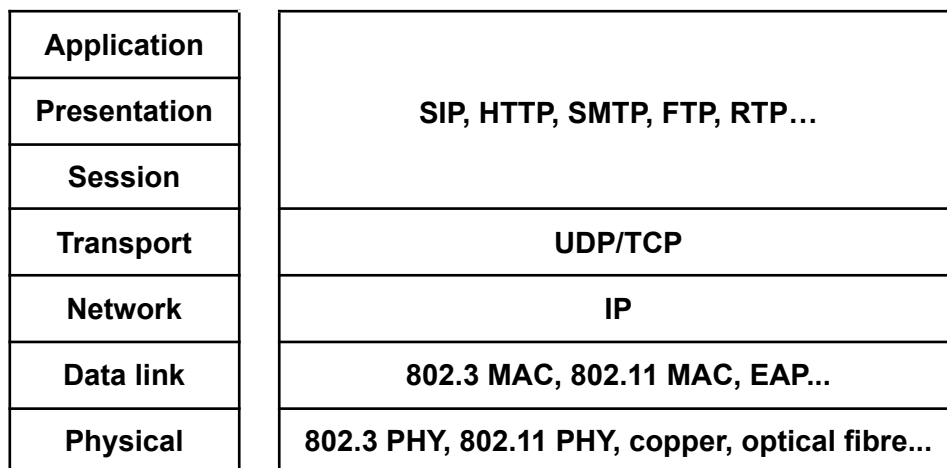
This report details the architecture of the system as well as the implementation of the entire IP telephony infrastructure allowing communication between these two devices. Firstly, some theoretical elements on the protocols used will be looked at, in particular on the role and functioning of SIP in VoIP and RTP. Secondly, the interest of using the Asterisk system in the design of this infrastructure and its implementation will be evaluated, then the two clients (Raspberry Pi and Alcatel phone) will be configured and tested. Finally a user-friendly SIP client in JavaScript for the Raspberry Pi will be studied.

¹ In reality *Asterisk* is a system that is also capable of handling other protocols. It has a SIP server. When we talk about the *Asterisk* server in this report, we must understand that we are talking about the SIP server that *Asterisk* offers.

1. SIP protocol and VoIP communication

The first questions to ask before starting the implementation of this infrastructure are: What is VoIP? What is SIP and how does it relate to VoIP? What is the relationship between SIP and RTP?

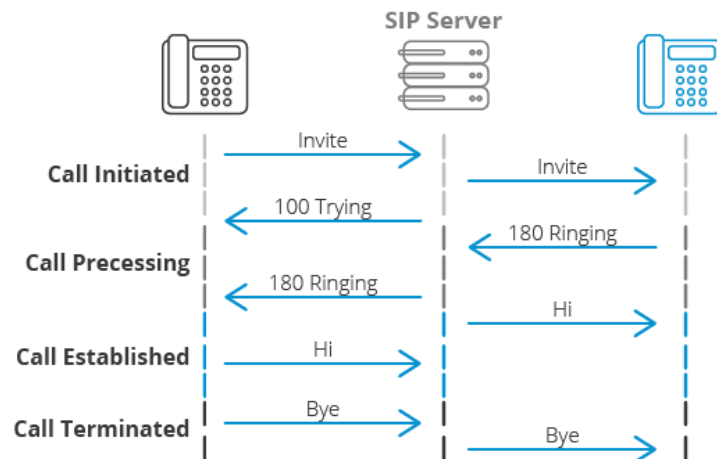
Answering the first question requires understanding the internet stack (Internet protocol suite).



(Figure 2 - Internet protocol stack)

VoIP stands for Voice over Internet Protocol, voice is specifically an analogue signal which can be acquired by means of a microphone and then digitally encoded using an analogue-to-digital converter. Once encoded, the data can be passed through the Internet protocol stack. In this diagram, it is therefore possible to transfer voice encoded using the HTTP protocol, transported in TCP, in an IP network via an Ethernet link to another device using this same protocol stack. Note that not all application protocols are necessarily suitable for transferring encoded voice. Some are suitable for real-time communication, e.g. transmission of encoded voice (RTP), others for establishing a communication (SIP)...

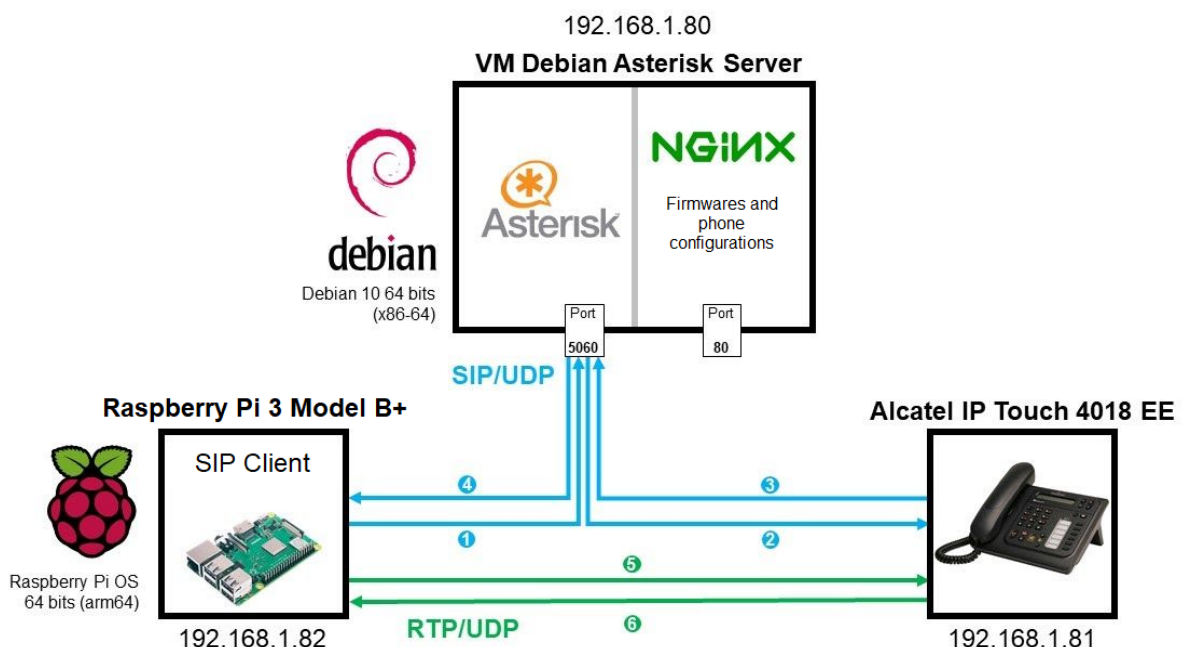
SIP (Session Initiation Protocol) is the protocol that allows establishing a communication between endpoints. Typically, an endpoint initially acts as a SIP client to contact another endpoint. It first queries a SIP server which provides it with information about the identity of this other endpoint and how to access it in the IP network. Once the information has been received, communication takes place directly between the two endpoints via the Real-time Transport Protocol (RTP). This protocol transmits the encoded voice. Once the communication is over, SIP takes over again to close the connection (session).



(Figure 3 - Establishment and termination of a VoIP communication using SIP - attribution: 3cx.com)

Going back to our very first diagram, for the establishment of a communication (e.g. Raspberry Pi calls Alcatel):

- a. RPi sends a contact request to Alcatel (*Invite*) through Asterisk. Alcatel receives the *Invite* response, rings and indicates RPi that it is ringing. Someone picks up the Alcatel, the contact is established, so...
- b. ...RTP takes over and transfers the voices between the two devices in direct connection (without going through Asterisk).



(Figure 4 - Establishing a telephone call between the Raspberry Pi and the Alcatel IP Touch 4018 EE telephone)

Once the call is over (one person hangs up), SIP takes care of closing the session between the two participants.

2. Implementation of an Asterisk IP PBX server

In large organisations (companies, public services, etc.), there are mainly two types of telephone communication: internal communication and external communication. The agents therefore have an internal fixed telephone capable of making calls to another internal fixed telephone or an external (public) telephone. For this to work, a PBX, also known as a Private Branch Exchange, is needed.



The PBX (on the left), has many advantages, including:

- financially, the bill is reduced, as internal calls do not go through the public network (Orange, formerly France Telecom in France);
- more internal numbers can be allocated without difficulty;
- it is possible to offer services such as conference calls, call forwarding, call transfer;
- and, of course, linking an internal line to an external line.

A comprehensive list of the functions of a traditional PBX can be found on Wikipedia **(1)**.

(Figure 5 - PBX Matra MC6500 serie)

Now that the overall operation of this system has been presented, how can communication using SIP protocol be achieved?

Historically, there have been three main phases in the evolution of telephone communications:

1. The public switched telephone network (PSTN) which is analogue. In this network, the SIP protocol cannot be used because SIP is a digital protocol.
2. The Integrated Services Digital Network (ISDN) is a digital network whose network stack is based on the OSI model. Starting at Layer 3 (transport), the following protocols are used: Q.931 **(2)**, X.25 layer 3 **(3)**. Here it is Layer 3 which is problematic: SIP relies on IP to work.
3. Voice over IP, over the Internet, is digital. SIP can work in this case. The associated PBX is an IP PBX (Internet Protocol private branch exchange).

There are many IP PBXs in the world. Asterisk has the distinction of being the world's number one in terms of usage. It has a free version and a proprietary version. And it offers interoperability with older networks (PSTN and ISDN) by means of hardware cards and software modules. This last point is probably a major factor in its success: companies that still use old equipment, and that, in a perspective of modernisation, migrate to recent

equipment, probably use several systems (PSTN, ISDN or VoIP), Asterisk will make it possible to manage these three systems simultaneously.

So let's proceed with the installation of Asterisk.

Technology choices

- OS: Debian 10, it is free and is mainly used as a server in the computer world. It is a system with regularly updated packages in terms of stability and security.
- Asterisk: 18 LTS² release compiled from source. The version in the Debian repositories is old (16 for Debian 10) and although it is also an LTS version, the fact that it is already compiled offers less flexibility in terms of adding modules. In addition, the module that handled SIP³ in the 16 release has been deprecated since the 17 release **(4)** in favour of a new module (**PJSIP**) that can handle SIP as well as NAT traversal functions with SIP⁴ **(5)**.

Prerequisite

Have an up-to-date Debian 10 machine (without GUI) connected to the local network and to the internet, also with SSH access (see [appendix A1](#) for the detailed implementation of this prerequisite).

Consider in this section the following information from this machine:

IP address	User	Password
192.168.1.80	asterisktz	voiputc
	root	voiputc

² LTS: Long Term Support, version maintained for a long period of time. These are the versions to be preferred for a production launch.

³ *chan_sip.so*, the SIP management module of Asterisk 16 and earlier is officially no longer maintained.

⁴ The issue of NAT traversal with SIP is not seen in this report.

Installation of Asterisk

Note: the libraries allowing the management of telephone interface cards from Digium⁵ (the company maintaining Asterisk) and the management of protocols used in ISDN⁶ networks will not be installed.

1. Connect via SSH to the `asterisktz` machine.

```
# ssh login@vm_ip_address -p 22
ssh asterisktz@192.168.1.80 -p 22
```

2. Install the packages allowing the compilation of Asterisk and the prerequisites.

```
asterisktz@asterisktz:~$                               Installing the build chain and prerequisites
sudo apt update && sudo apt install linux-headers-$(uname -r)
build-essential autoconf libglib2.0-dev libtool net-tools
```

3. Reboot the Debian machine and connect again via SSH.

```
asterisktz@asterisktz:~$                               Rebooting the Debian machine
sudo reboot
```

```
# SSH reconnection
ssh asterisktz@192.168.1.80 -p 22
```

4. Go to the directory `/usr/src` and download Asterisk 18 at this address:
<https://downloads.asterisk.org/pub/telephony/asterisk/asterisk-18-current.tar.gz>.

```
asterisktz@asterisktz:~$                               Changing directory to /usr/src
cd /usr/src
```

```
asterisktz@asterisktz:/usr/src$                       Downloading Asterisk 18 LTS
sudo wget https://downloads.asterisk.org/pub/telephony/asterisk/asterisk-18-current.tar.gz
```

5. Decompress the archive.

```
asterisktz@asterisktz:/usr/src$                       Decompressing the archive
sudo tar -zxvf asterisk-18-current.tar.gz
```

⁵ DAHDI module (Digium/Asterisk Hardware Device Interface):
<https://wiki.asterisk.org/wiki/pages/viewpage.action?pagelId=4817506>.

⁶ libpri library: <https://wiki.asterisk.org/wiki/pages/viewpage.action?pagelId=4817506>.

6. Run the `install_prereq` script installing the prerequisites.

Note: To find out the minor version number, type `ls`. In this report, it is Asterisk 18.2.

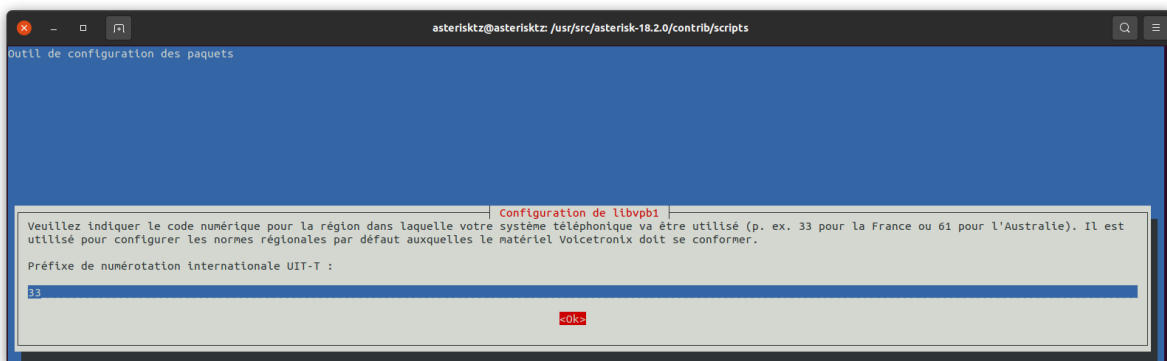
```
asterisktz@asterisktz: /usr/src$ Changing directory
```

```
cd asterisk-18.2.0/contrib/scripts/
```

```
asterisktz@asterisktz: /usr/src/asterisk-18.2.0/contrib/scripts$ Prerequisites
```

```
sudo ./install_prereq install
```

During the installation of the prerequisites a window appears asking to select the telephone number code. Select **33** for France and validate with **<Ok>**. You can find an international list of area codes on this site: <https://countrycode.org/>.



(Figure 6 - Area code setting)

7. Checking the required dependencies.

```
asterisktz@asterisktz: /usr/src/asterisk-18.2.0/contrib/scripts$ Changing directory
```

```
cd ../..
```

```
asterisktz@asterisktz: /usr/src/asterisk-18.2.0$ Running the configure script
```

```
sudo ./configure
```

If the operation is successful, you will get a message similar to this one with the Asterisk logo:

```

      .$$$$$$$$$$$$$$$$$=..
      .7$77..          .7$77:..
      .7$77..          .7$77:..
      .$$:..           ,,$7.7
      .77.            7$$$$$      .$$77
      ..$$            $$$$$$      .$$$7
      ..7$   .?.   $$$$$$   .?.   7$$$$.
      $.$.   .$$$7. $$$$77 .7$$$$.   .$$$$.
      .777.   .$$$$$$$77$$$$77$$$$$$$7.   $$$$.
      $$$~   .7$$$$$$$$$$$$$$$$$7.   .$$$$.
      .$$7    .7$$$$$$$$$7:   ?$$$$.
      $$$     ?7$$$$$$$$$$$$$I   .$$$7
      $$$     .7$$$$$$$$$$$$$$$$$   :$$$$.
      $$$     $$$$$$7$$$$$$$$$$$$$   .$$$$.
      $$$     $$$  7$$$7  .$$$   .$$$$.
      $$$$    $$$$$$   .$$$7   .$$$$.
      7$$$7    7$$$$$   7$$$   7$$$$
      $$$$$$   $$$$$$   $$$
      $$$7.    $$$   (TM)
      $$$$$$.   .7$$$$$  $$
      $$$$$$$$$$$$$$7$$$$$$$$$. $$$$$$
      $$$$$$$$$$$$$$.

```

```

configure: Package configured for:
configure: OS type : linux-gnu
configure: Host CPU : x86_64
configure: build-cpu:vendor:os: x86_64 : pc : linux-gnu :
configure: host-cpu:vendor:os: x86_64 : pc : linux-gnu :

```

If it went wrong, here is a link to the documentation:
<https://wiki.asterisk.org/wiki/display/AST/Checking+Asterisk+Requirements>

8. Selecting the options to be installed for Asterisk. Ensure that the terminal is at least **80 x 27** in size.

```
asterisktz@asterisktz: /usr/src/asterisk-18.2.0$
```

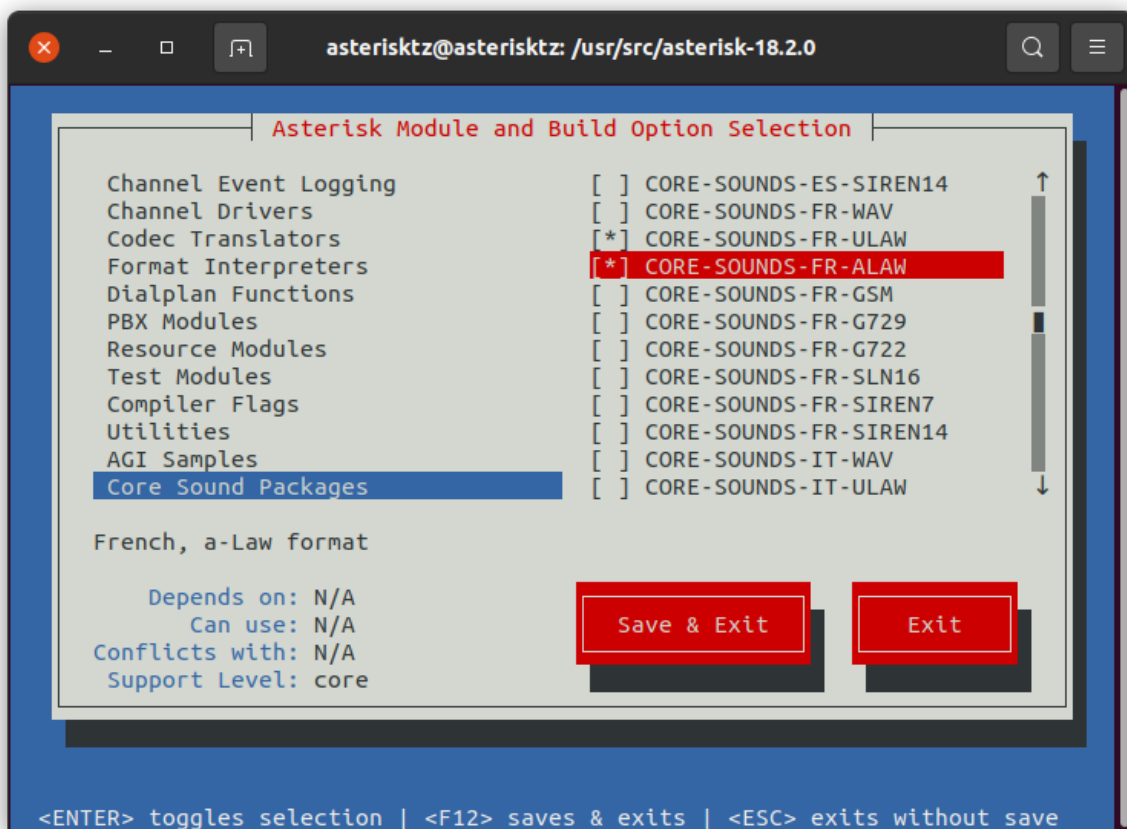
Choosing options for Asterisk

```
sudo make menuselect
```

A window appears allowing you to choose the options.

Go to the Core Sound Package section to install the French sounds (or another language if you prefer):

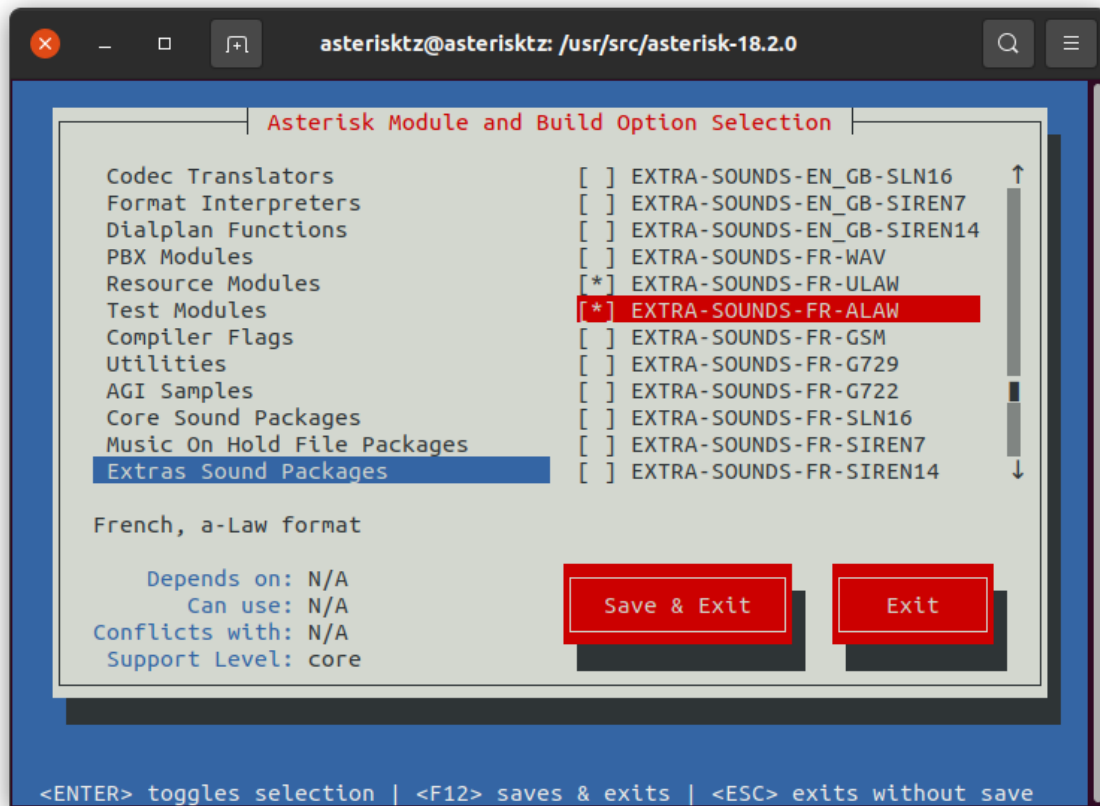
- Deselect the **CORE-SOUNDS-EN-GSM** package. These are the English sounds with the GSM audio codec.
- Select the **CORE-SOUNDS-FR-ULAW** and **CORE-SOUNDS-FR-ALAW**. These are the French sounds with ULAW and ALAW⁷ codecs (supported by the Alcatel IP Touch 4018EE (6)).



(Figure 7 - Selection of Asterisk sound modules)

⁷ These codecs are defined by the ITU G.711 standard: <https://www.itu.int/rec/T-REC-G.711-198811-I/en>. The ALAW codec is used in Europe and Africa. The ULAW codec in North America and Japan (7).

Then in the Extra Sound Packages section, select the **EXTRA-SOUNDS-FR-ULAW** and **EXTRA-SOUNDS-FR-ALAW** packages.



(Figure 8 - Selection of Asterisk extra sound modules)

Confirm by pressing the **F12** key.

9. Compile the program with the previously chosen options and install Asterisk.

```
asterisktz@asterisktz: /usr/src/asterisk-18.2.0$
```

Compiling Asterisk

```
sudo make
```

The compilation takes time depending on the power of the machine. Once successful...

```
+----- Asterisk Build Complete -----+
+ Asterisk has successfully been built, and +
+ can be installed by running:             +
+                                           +
+           make install                   +
+----- Asterisk Build Complete -----+
```

...the installation of Asterisk can begin.

```
asterisktz@asterisktz: /usr/src/asterisk-18.2.0$
```

Installing Asterisk

```
sudo make install
```

The installation ends with this message:

```
+----- Asterisk Installation Complete -----+
+
+   YOU MUST READ THE SECURITY DOCUMENT   +
+
+ Asterisk has successfully been installed. +
+ If you would like to install the sample +
+ configuration files (overwriting any    +
+ existing config files), run:           +
+
+ For generic reference documentation:    +
+   make samples                          +
+
+ For a sample basic PBX:                 +
+   make basic-pbx                        +
+
+
+----- or -----+
+
+ You can go ahead and install the asterisk +
+ program documentation now or later run:  +
+
+           make progdocs                  +
+
+ **Note** This requires that you have   +
+ doxygen installed on your local system  +
+-----+

```

10. Create the sample configuration files in the `/etc/asterisk` folder.

```
asterisktz@asterisktz: /usr/src/asterisk-18.2.0$
```

Creating sample configuration files

```
sudo make samples
```

11. Install the start-up scripts.

```
asterisktz@asterisktz: /usr/src/asterisk-18.2.0$
```

Installing start-up scripts

```
sudo make config
sudo ldconfig
```

12. Setting up the automatic start of the Asterisk service when the machine is launched.

```
asterisktz@asterisktz: /usr/src/asterisk-18.2.0$
```

Creating an asterisk user

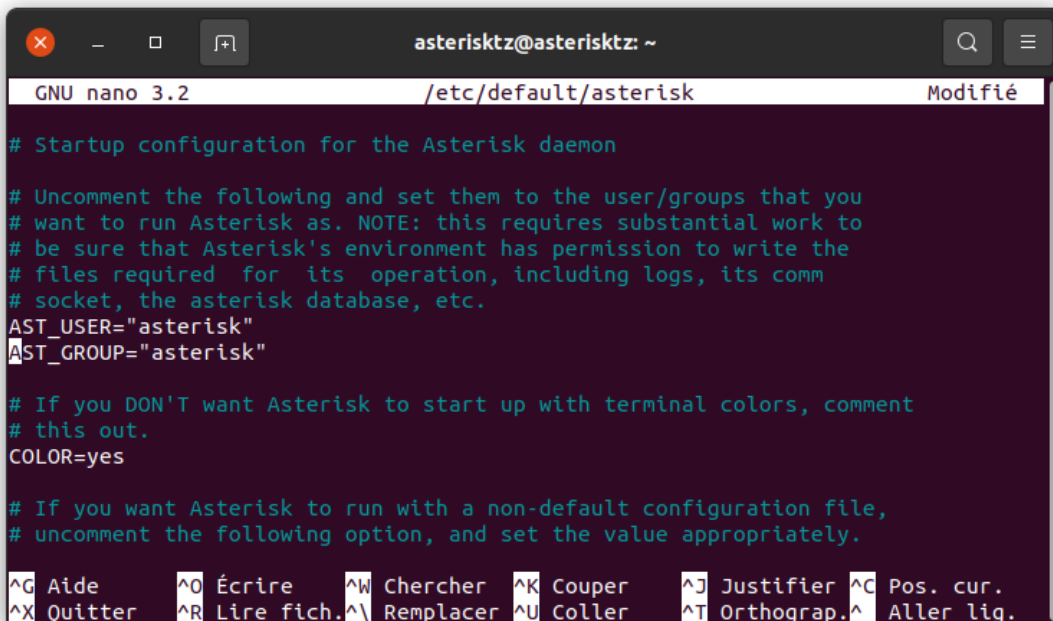
```
cd
sudo groupadd asterisk
sudo useradd -r -d /var/lib/asterisk -g asterisk asterisk
sudo usermod -aG audio,dialout asterisk
sudo chown -R asterisk.asterisk /etc/asterisk
sudo chown -R asterisk.asterisk /var/{lib,log,spool}/asterisk
sudo chown -R asterisk.asterisk /usr/lib/asterisk
```

```
asterisktz@asterisktz: ~$
```

Adding asterisk as the default user of the **Asterisk** service - 1

```
sudo nano /etc/default/asterisk
```

Uncomment the lines `AST_USER="asterisk"` and `AST_GROUP="asterisk"` (remove the `#` before each line). Save with **Ctrl + O**. Exit with **Ctrl + X**.



```
asterisktz@asterisktz: ~
GNU nano 3.2 /etc/default/asterisk Modifié
# Startup configuration for the Asterisk daemon
# Uncomment the following and set them to the user/groups that you
# want to run Asterisk as. NOTE: this requires substantial work to
# be sure that Asterisk's environment has permission to write the
# files required for its operation, including logs, its comm
# socket, the asterisk database, etc.
AST_USER="asterisk"
AST_GROUP="asterisk"
# If you DON'T want Asterisk to start up with terminal colors, comment
# this out.
COLOR=yes
# If you want Asterisk to run with a non-default configuration file,
# uncomment the following option, and set the value appropriately.
^G Aide      ^O Écrire    ^W Chercher  ^K Couper    ^J Justifier ^C Pos. cur.
^X Quitter   ^R Lire fich.^_ Remplacer  ^U Coller   ^T Orthograp.^_ Aller lig.
```

(Figure 9 - Setting asterisk as the default user of the Asterisk service)

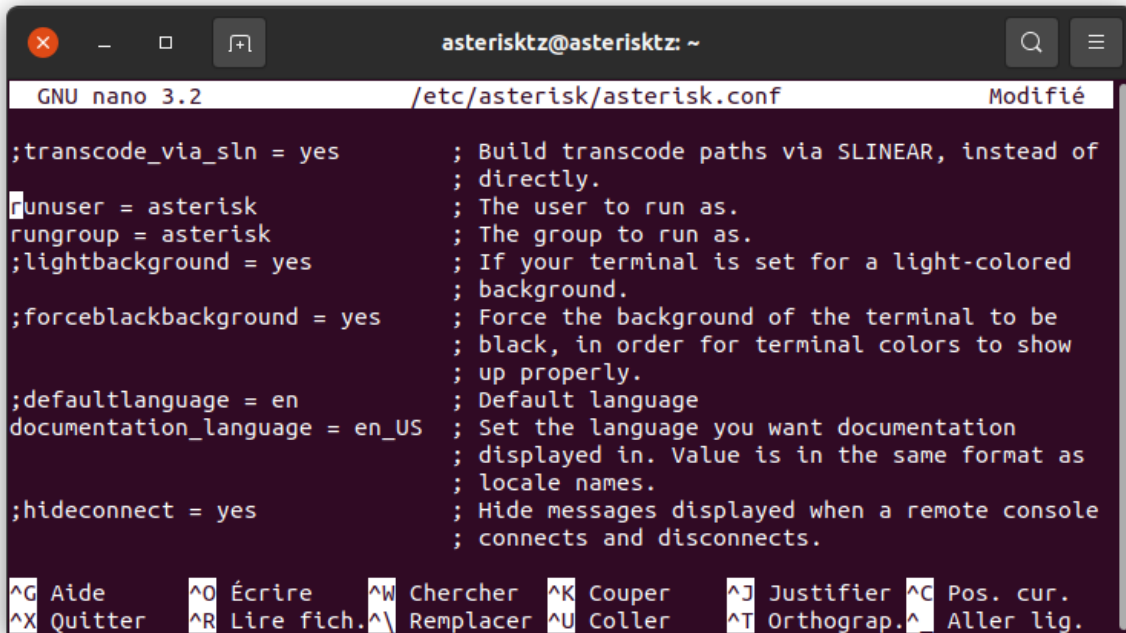
```
asterisktz@asterisktz: ~$
```

Adding asterisk as the default user of the **Asterisk** service - 2

```
sudo nano /etc/asterisk/asterisk.conf
```

Uncomment the lines (remove the semicolon ; before each line). Save with **Ctrl + O**. Exit with **Ctrl + X**.

```
runuser = asterisk ; The user to run as.
rungroup = asterisk ; The group to run as.
```



```
GNU nano 3.2 /etc/asterisk/asterisk.conf Modifié
;transcode_via_sln = yes           ; Build transcode paths via SLINEAR, instead of
;                                  ; directly.
runuser = asterisk                 ; The user to run as.
rungroup = asterisk                ; The group to run as.
;lightbackground = yes            ; If your terminal is set for a light-colored
;                                  ; background.
;forceblackbackground = yes       ; Force the background of the terminal to be
;                                  ; black, in order for terminal colors to show
;                                  ; up properly.
;defaultlanguage = en             ; Default language
documentation_language = en_US    ; Set the language you want documentation
;                                  ; displayed in. Value is in the same format as
;                                  ; locale names.
;hideconnect = yes                ; Hide messages displayed when a remote console
;                                  ; connects and disconnects.

^G Aide      ^O Écrire   ^W Chercher ^K Couper   ^J Justifier ^C Pos. cur.
^X Quitter   ^R Lire fich.^_ Remplacer ^U Coller   ^T Orthograp.^_ Aller lig.
```

(Figure 10 - Setting *asterisk* as the default user of the Asterisk service)

13. Start the Asterisk service

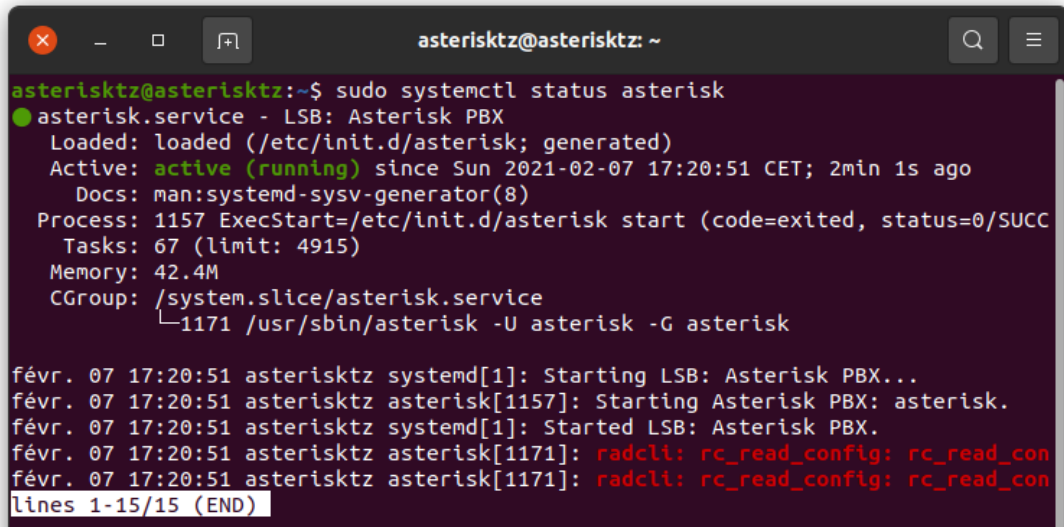
```
asterisktz@asterisktz:~$ Starting Asterisk
```

```
sudo systemctl start asterisk
```

You can now check its current status with the following command:

```
asterisktz@asterisktz:~$ Starting Asterisk
```

```
sudo systemctl status asterisk
```



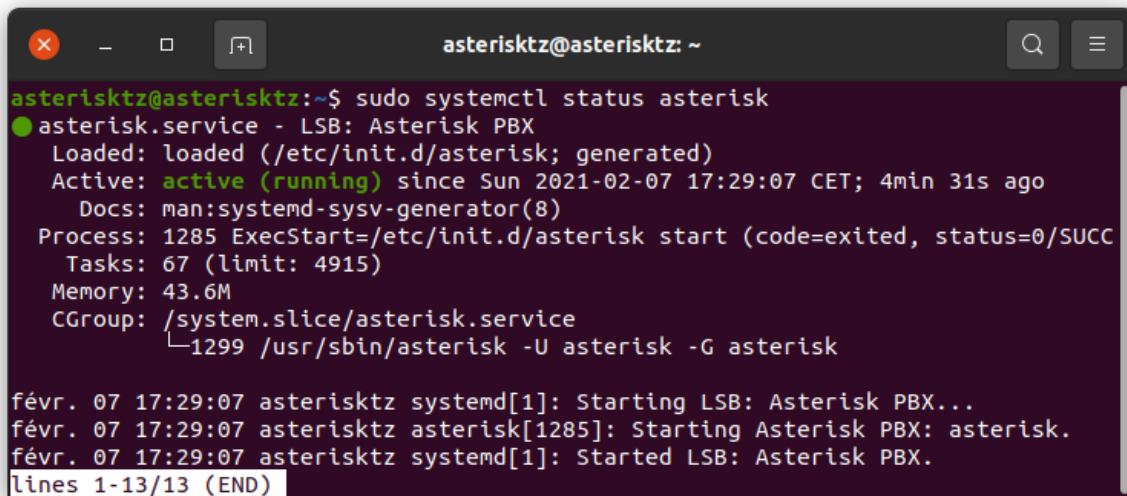
```
asterisktz@asterisktz: ~  
asterisktz@asterisktz:~$ sudo systemctl status asterisk  
● asterisk.service - LSB: Asterisk PBX  
  Loaded: loaded (/etc/init.d/asterisk; generated)  
  Active: active (running) since Sun 2021-02-07 17:20:51 CET; 2min 1s ago  
    Docs: man:systemd-sysv-generator(8)  
 Process: 1157 ExecStart=/etc/init.d/asterisk start (code=exited, status=0/SUCCESS)  
   Tasks: 67 (limit: 4915)  
  Memory: 42.4M  
   CGroup: /system.slice/asterisk.service  
           └─1171 /usr/sbin/asterisk -U asterisk -G asterisk  
  
févr. 07 17:20:51 asterisktz systemd[1]: Starting LSB: Asterisk PBX...  
févr. 07 17:20:51 asterisktz asterisk[1157]: Starting Asterisk PBX: asterisk.  
févr. 07 17:20:51 asterisktz systemd[1]: Started LSB: Asterisk PBX.  
févr. 07 17:20:51 asterisktz asterisk[1171]: radcli: rc_read_config: rc_read_con  
févr. 07 17:20:51 asterisktz asterisk[1171]: radcli: rc_read_config: rc_read_con  
lines 1-15/15 (END)
```

(Figure 11 - Launch of the Asterisk service with errors)

At this stage, if there are these two red lines, they can be corrected in this way⁸:

```
asterisktz@asterisktz:~$  
  
sudo systemctl stop asterisk  
  
sudo sed -i 's";\[radius\]"\[radius\]"g' /etc/asterisk/cdr.conf  
  
sudo sed -i 's";radiuscfg =>  
/usr/local/etc/radiusclient-ng/radiusclient.conf"radiuscfg =>  
/etc/radcli/radiusclient.conf"g' /etc/asterisk/cdr.conf  
  
sudo sed -i 's";radiuscfg =>  
/usr/local/etc/radiusclient-ng/radiusclient.conf"radiuscfg =>  
/etc/radcli/radiusclient.conf"g' /etc/asterisk/cel.conf  
  
sudo systemctl start asterisk
```

⁸ Fix: <https://www.clearhat.org/blog/post/a-fix-for-apt-install-asterisk-on-ubuntu-18-04>.

A terminal window titled 'asterisktz@asterisktz: ~' showing the output of the command 'sudo systemctl status asterisk'. The output indicates that the 'asterisk.service' is active and running. It provides details such as the loaded path, active status since Sun 2021-02-07 17:29:07 CET, process ID 1285, and memory usage of 43.6M. Log messages at the bottom show the service starting successfully on Feb 07 at 17:29:07.

```
asterisktz@asterisktz:~$ sudo systemctl status asterisk
● asterisk.service - LSB: Asterisk PBX
   Loaded: loaded (/etc/init.d/asterisk; generated)
   Active: active (running) since Sun 2021-02-07 17:29:07 CET; 4min 31s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 1285 ExecStart=/etc/init.d/asterisk start (code=exited, status=0/SUCCESS)
    Tasks: 67 (limit: 4915)
   Memory: 43.6M
    CGroup: /system.slice/asterisk.service
            └─1299 /usr/sbin/asterisk -U asterisk -G asterisk

févr. 07 17:29:07 asterisktz systemd[1]: Starting LSB: Asterisk PBX...
févr. 07 17:29:07 asterisktz asterisk[1285]: Starting Asterisk PBX: asterisk.
févr. 07 17:29:07 asterisktz systemd[1]: Started LSB: Asterisk PBX.
lines 1-13/13 (END)
```

(Figure 12 - Launch of the Asterisk service without errors)

14. Automatic start-up of the Asterisk service.

```
asterisktz@asterisktz:~$ Automatic start-up of Asterisk
sudo /lib/systemd/systemd-sysv-install enable asterisk
```

Asterisk is operational. Let's move on to user creation and SIP configuration!

SIP configuration and user creation

So we have an operational server, we must now configure SIP and create users. We will create 3 users with the following characteristics:

	Alcatel phone	Raspberry Pi	Test
Purpose	Dedicated account for the Alcatel IP Touch 4018 EE phone	Dedicated account for the Raspberry Pi	Dedicated account for testing
Display name	Alcatel IP Touch	Raspberry Pi	Guillaume Nibert
Phone number	5001	5002	5003
Login	alcatel	rpi	guillaume
Password	11111111	22222222	33333333

There are two configuration files to edit: *pjsip.conf* and *extensions.conf*. The first one is used to create accounts, configure the operation of SIP (UDP/TCP), the authentication systems... and the second to define the behaviour of the system, more precisely the dialling plan (similar to routing if we were talking about IP packets). This "routing" is done according to telephone numbers (identifiers).

These files are present in */etc/asterisk* and in the *asterisk_sip* directory of the GitHub repository.

SIP configuration - pjsip.conf

1. Rename the *pjsip.conf* configuration file to *pjsip_original.conf*.

```
asterisktz@asterisktz:~$
```

Saving the initial *pjsip.conf* configuration file

```
sudo mv /etc/asterisk/pjsip.conf /etc/asterisk/pjsip_original.conf
```

2. Create a *pjsip.conf* file...

```
asterisktz@asterisktz:~$
```

Setting up accounts and SIP

```
sudo nano /etc/asterisk/pjsip.conf
```


...and write the following content:

/etc/asterisk/pjsip.conf

```
[transport-udp]
type=transport
protocol=udp
bind=0.0.0.0

; Basic templates, they will be copied for each user

[endpoint_basic](!)
type=endpoint          ; endpoint (phone/rpi/pc...)
context=plan-num      ; uses the dial plan defined in extensions.conf
disallow=all          ; disabling all audio codecs
allow=ulaw             ; except the ULAW codec
allow=alaw             ; and the ALAW codec
language=fr

[authentication](!)
type=auth              ; type of section: authentication
auth_type=userpass    ; password authentication

[aor_template](!)
type=aor               ; find out where the endpoint can be contacted
max_contacts=1

; Definitions of user accounts associated with equipment

[alcatel](endpoint_basic)
auth=alcatel
aors=alcatel
callerid="Alcatel IP Touch" <5001> ; to have the name of the caller displayed
[alcatel](authentication)
password=11111111
username=alcatel
[alcatel](aor_template)

[rpi](endpoint_basic)
auth=rpi
aors=rpi
callerid="Raspberry Pi" <5002>
[rpi](authentication)
password=22222222
username=rpi
[rpi](aor_template)

[guillaume](endpoint_basic)
auth=guillaume
aors=guillaume
callerid="Guillaume Nibert" <5003>
[guillaume](authentication)
password=33333333
username=guillaume
[guillaume](aor_template)
```

The creation of the accounts and the configuration of SIP is complete. Let's move on to the dial plan.

Dial plan - extensions.conf

1. Rename the `extensions.conf` configuration file to `extensions_original.conf`.

```
asterisktz@asterisktz:~$
```

Saving the initial `extensions.conf` configuration file

```
sudo mv /etc/asterisk/extensions.conf  
/etc/asterisk/extensions_original.conf
```

2. Create an `extensions.conf` file...

```
asterisktz@asterisktz:~$
```

Defining a dial plan

```
sudo nano /etc/asterisk/extensions.conf
```

...and write the following content:

`/etc/asterisk/extensions.conf`

```
[plan-num]
exten => 5001,1,Answer(500)
exten => 5001,2,Dial(PJSIP/alcatel,25)
exten => 5001,3,Hangup()

exten => 5002,1,Answer(500)
exten => 5002,2,Dial(PJSIP/rpi,25)
exten => 5002,3,Hangup()

exten => 5003,1,Answer(500)
exten => 5003,2,Dial(PJSIP/guillaume,25)
exten => 5003,3,Hangup()

; 1, 2 and 3 correspond to the priorities of the Answer(),
; Dial() and Hangup() application calls. 1 being the highest
priority.
; We can also write 1,n,n where the first n corresponds to 2 and the
; second to 3.
```

“The **Answer()** application takes a delay (in milliseconds) as its first parameter. Adding a short delay is often useful to ensure that the endpoint has time to start processing the audio before starting the communication via the **Dial()** application. Otherwise, you may not hear the very beginning” (8). **Hangup()** as the name suggests hangs up the current call.

The dial plan is complete. The SIP accounts have been created. We can now proceed to the configuration of a SIP client on the Raspberry Pi.

3. Installation and configuration of a SIP client on the Raspberry Pi

The installation and configuration of a SIP client on the Raspberry Pi is necessary to communicate with VoIP. This client will connect to the Asterisk server and depending on the number the client is calling, the server will use the dial plan defined in `extensions.conf` to contact the right endpoint (Alcatel phone for example).

Technology choices

- OS: Raspberry Pi OS Buster (arm64), the 64-bit version is only very recent but seems to be promising. Indeed, it is more powerful than the 32 bits version (armhf) **(9)**. This is a significant advantage, especially on a Raspberry Pi 3B+ which will be used in desktop mode. Any improvement in performance is worthwhile. Furthermore, Raspberry Pi OS is a system officially maintained by the Raspberry Pi Foundation.
- SIP client: *linphonec*, the command line version of *Linphone*. It is stable and works perfectly on Raspberry Pi OS. It is open source. Unfortunately, the popular client *Jami* (formerly *GNU Ring*), developed by *Savoir-faire Linux* does not seem to work well with Raspberry Pi OS.

Prerequisites

Having an *up-to-date Raspberry Pi 3B+* running *Raspberry Pi OS Buster (64-bit)*⁹ connected to the local network and the internet, also with SSH access (see [appendix A2](#) of the PDF report for the detailed implementation of this requirement). The Asterisk server must be running.

Consider in this section the following information from this machine:

IP address	User	Password
Eth: 192.168.1.82	pi	voippiutc
Wi-Fi: 192.168.1.92		

⁹ `sudo apt update && sudo apt upgrade -y && sudo apt dist-upgrade -y && sudo apt autoremove -y && sudo reboot`

Installation and configuration of the Linphone SIP client

1. Start the Raspberry Pi then launch a terminal and install Linphone.

```
pi@raspberrypi:~$
```

Installing Linphone

```
sudo apt install linphone -y
```

2. Once the installation is complete, register the associated *rpi* user on the server.

```
pi@raspberrypi:~$
```

Executing *linphonec*

```
linphonec
```

```
linphonec>
```

Enregistrement du client *rpi* sur le serveur Asterisk

```
#register sip:login@domain domain <password>
```

```
register sip:rpi@192.168.1.80 192.168.1.80 22222222
```

The *rpi* client is connected to the Asterisk server. It can therefore contact the Alcatel telephone and vice versa.

Client-side testing

If the registration on the server was successful, the previously executed command returns this:

```
linphonec> Refreshing on sip:rpi@192.168.1.80...  
linphonec> Registration on sip:192.168.1.80 successful.  
linphonec>
```

Server-side testing

On a server console, type the command `sudo asterisk -rvvv`

Once entered, type the command `pjsip show endpoints`. If the Raspberry Pi's SIP client is connected then the console will return this:

Output

```
asterisktz*CLI> pjsip show endpoints

Endpoint: <Endpoint/CID.....> <State.....>
<Channels.>
  I/OAuth:
<AuthId/UserName.....>
  Aor: <Aor.....> <MaxContact>
  Contact: <Aor/ContactUri.....> <Hash.....> <Status>
<RTT(ms)..>
  Transport: <TransportId.....> <Type> <cos> <tos>
<BindAddress.....>
  Identify:
<Identify/Endpoint.....>
  Match: <criteria.....>
  Channel: <ChannelId.....> <State.....>
<Time.....>
  Exten: <DialedExten.....> CLCID: <ConnectedLineCID.....>
=====
==

Endpoint:   alcatel/5001                               Unavailable   0 of inf
  InAuth:   alcatel/alcatel
  Aor:      alcatel                                   1

Endpoint:   guillaume/5003                             Unavailable   0 of inf
  InAuth:   guillaume/guillaume
  Aor:      guillaume                               1

Endpoint:   rpi/5002                                   Not in use    0 of inf
  InAuth:   rpi/rpi
  Aor:      rpi                                     1
  Contact:  rpi/sip:rpi@192.168.1.82;transport=udp  cec2f9dd2f NonQual  nan

Objects found: 3
```

It is clear that the client has an IP address of **192.168.1.82** and is connected. The information **"Not in use"** indicates that there is no call in progress.

The Alcatel phone is still not connected, it is time to integrate it into the system.

4. IP phone configuration

The Alcatel IP Touch 4018 EE is a telephone with two operating modes:

1. NOE (New Office Environment) mode: proprietary communication protocol developed by Alcatel/Lucent;
2. SIP mode.

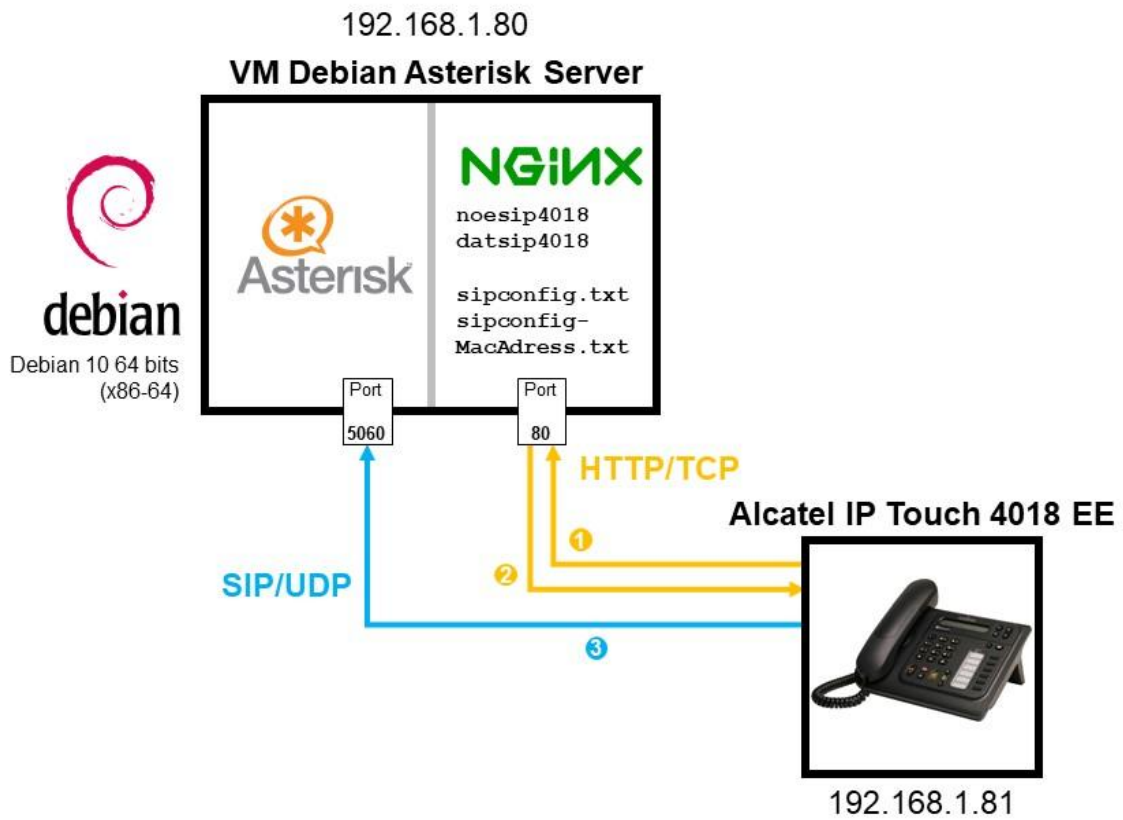
First, it must be configured in SIP mode. Then, it has the particularity to configure itself automatically by downloading its configuration files and its firmware files on a *TFTP*, *HTTP* or *HTTPS* server at startup.

Among the three protocols, the most secure is *HTTPS*. We tried to implement it with 2048 bit RSA certificates and *cipher suites* compatible with older equipment. Unfortunately, being in a local network, the Alcatel phone seems not to accept self-signed certificates¹⁰. Given this problem, there are only two choices: *TFTP* or *HTTP*. These application protocols are not very secure: no authentication, transfer of data in clear text over the network... The choice is therefore made at the level of the transport protocols: *TFTP* necessarily uses *UDP*, a non-connected mode, whereas *HTTP* can be configured to use *TCP*, a connected mode, which includes error detection and correction. The programs transmitted are firmware, if data is corrupted due to an error, it could brick the phone. So let's use the *HTTP* protocol associated with the *TCP* protocol. We will therefore set up a lightweight *NGINX*¹¹ *HTTP* server, whose sole purpose is to provide firmware and configuration files to the Alcatel IP Touch phone. It will be accessible at the address **192.168.1.80** on port **80**.

Of course, if possible, for production use, we recommend using *HTTPS*.

¹⁰ According to this forum, it seems that Alcatel IP Touch phones only trust the certification authority issued by the Alcatel-Lucent company. The root certificate from Alcatel-Lucent seems to be inside the phone: <https://www.alcatelunleashed.com/viewtopic.php?t=28822>. We do not have the means to get a certificate from the Alcatel-Lucent certification authority.

¹¹ NGINX open-source HTTP server: <https://www.nginx.com/>.



(Figure 13 - Retrieving Alcatel phone configuration files via HTTP)

Configuration information

IP address of the Alcatel phone. This must be configured on the router (see appendix A1.15).	192.168.1.81
IP address and port of the HTTP server	192.168.1.80:80
PoE Injector connected to the router	TP-Link TL-POE150S ¹²

¹² TP-Link TL-POE150S: <https://www.tp-link.com/en/business-networking/accessory/tl-poe150s/>.

Configuring the phone in SIP mode - on the device

Connect the phone to the PoE injector and follow the instructions in the installation manual below (screenshots of the procedure).

3. SIP stand-alone mode

3.1. Firmware version identification

In addition to existing call processing support for the Alcatel-Lucent OmniPCX™ Communication Servers (NOE signaling protocol), the Alcatel-Lucent IP Touch 40x8EE phone can also be configured as Session Initiation Protocol (SIP) endpoints and, therefore, operate in a standard SIP environment. The Alcatel-Lucent IP Touch 40x8 EE phone may also switch to SIP survival mode in SIP standalone mode.

Firmware version identification - phone out of the box.

In this context, the phone is natively configured in NOE mode

- 1- Power on the phone.
- 2- At the phase 2/5 network setup, press the "i" key and just after press the "#" key.
- 3- '0 - MAC address' menu appears.
- 4- Scroll down to '2 - Soft infos'.
- 5- Press the OK key.
- 6- The version is displayed : 'Version NOE 4.xx.xx'. If the version is at least 4.xx.xx, SIP boot mode is possible.

Firmware version identification - phone already installed.

- 1- Power on the phone.
- 2- At the phase 2/5 network setup, press the "i" key and just after the "#" key.
- 3- '0 - MAC address' menu appears.
- 4- Scroll down to '2 - Soft infos'.
- 5- Press the OK key.
- 6- Scroll down to 'Run mode':
 - if NOE is displayed, it means that the NOE signaling protocol is running,
 - if SIP is displayed, it means that the SIP signaling protocol mode is running.

3.2. Switching From NOE To SIP Mode

There are two methods.

Method 1 - Manual

- 1- Power on the phone.
- 2- At the phase 2/5 network setup, press the "i" key and just after the "#" key.
- 3- '0 - MAC address' menu appears.
- 4- Scroll down to '2 - Soft infos'.
- 5- Press OK.
- 6- 'Version NOE x.xx.xx' is displayed.
- 7- Scroll down to 'Run mode', 'Set Mode: NOE' is displayed.
- 8- Press OK to switch from NOE to SIP mode.
- 9- Press '#' to save the new configuration.
- 10- Press the Hang Up key to reset the device to take into account the new configuration.

For checking SIP status, follow 'Firmware version identification already installed' section.

(Figure 14 - Alcatel-Lucent, 3. SIP stand-alone mode In: IP Touch 4008/4018 Extended Edition - SIP Phone Installation Guide - 8AL90824AAAA ed02, p.4-5, August 2010, available at:

<https://www.cluster2.hostgator.co.in/files/writeable/uploads/hostgator136107/file/iptouchsipphoneinstallationguide-ed02.pdf>)

Once in SIP mode, the phone's IP address must be configured to match the one defined in the router (**192.168.1.81**), and it must also be provided with the IP address of the *HTTP* server and its port.

1. Connect the phone to the PoE injector.
2. At **phase 2/5 network setup**, press “**i**” then “**#**” until the “**MAC address**” menu appears.
3. If there is already a password configured, enter **000000**.
4. Scroll down to **IP Parameters**, then click **OK**.
5. Scroll down, then select **IP mode: Static**.
6. Scroll down, then enter the IP address: **IP@: 192.168.1.81**
7. Scroll down, then enter the subnet mask: **Subnet: 255.255.255.0**
8. Scroll down, then enter the router IP address: **Router: 192.168.1.254**
9. Scroll down, then select from DL Scheme: **HTTP**.
10. Scroll down, then select Use **Defaultport**.
11. Scroll down, then select the *HTTP* server address: **DL Addr: 192.168.1.80**
12. Scroll down, then select the *HTTP* server port: **DL Port: 80**
13. Scroll down, do not select a *VLAN*.
14. Scroll down to **save**, then click **OK**.

The configuration on the device is complete, however for now it will keep restarting in a loop as the *HTTP* server does not yet exist, nor do the associated configuration files and phone firmwares.

HTTP server installation

So let's go back to our Debian virtual machine and install an HTTP server.

1. Connect via SSH to the `asterisktz` machine.

```
ssh asterisktz@192.168.1.80 -p 22
```

2. Install *nginx* HTTP server.

```
asterisktz@asterisktz:~$
```

Installing *nginx* server

```
sudo apt install nginx
```

```
asterisktz@asterisktz:~$
```

Checking its status

```
sudo systemctl status nginx
```

If the output shows **active** then it is working, otherwise you need to run it (`sudo systemctl start nginx`). Normally it is configured to start automatically at startup, if this is not the case then run the following command:

```
sudo /lib/systemd/systemd-sysv-install enable nginx
```

The *HTTP* server is ready, all that remains is to transfer the firmware, the information on the SIP account dedicated to the Alcatel phone and the IP address of the Asterisk server.

Transfer of configuration files and firmware to the Alcatel phone

In practise, there are 4 files to transfer to the HTTP server (10):

- ***sipconfig.txt***: global configuration file, contains the settings to be applied to all Alcatel IP Touch 4018EE phones connected to the network.
- ***sipconfig-MacAddress.txt***: configuration file specific to a single Alcatel IP Touch 4018EE phone. This is the MAC address of the phone in question which must be written in lowercase.
- ***noesip4018***: proprietary firmware containing the SIP protocol application for the Alcatel IP Touch 4018EE.
- ***datsip4018***: resources containing ringtones and different melodies.

As the Alcatel-Lucent website does not necessarily provide the appropriate documentation for the structure of the ***sipconfig.txt*** files, we have based ourselves on an example configuration file created by Florian Duraffourg, a graduate of Télécom SudParis:

<https://github.com/fduraffourg/utis/blob/master/iptouch/sipconfig-reynoud.txt>

Concerning the firmwares, also difficult to find on the official Alcatel-Lucent website, we found them on the Alcatel Unleashed forum through fbird's post on March 21st 2016:

<https://www.alcatelunleashed.com/viewtopic.php?p=95015#p95015>

The most important file is ***sipconfig-MacAddress.txt*** whose important content in bold and green is as follows (we have voluntarily removed the non-essential parts, you will find the complete file in the project repository available in the following paragraph):

Note: we added comments in this report, to avoid mistakes, you should take the one located in the repository.

sipconfig-MacAddress.txt (green fields are to be taken into account)

```
[...]  
  
[dns]  
  
#####  
## The primary DNS IP address HAS TO BE FILLED  
## If no DNS, use the SIP proxy address instead  
#####  
  
  dns_addr=192.168.1.254 # DNS of the Freebox (or another router)  
  dns2_addr=  
  hostname=192.168.1.254  
  
[sip]  
  
#####  
## Domain name : IP address, FQDN or domain name (see the SIP proxy config)  
#####  
  
  domain_name=192.168.1.80 # Asterisk server's domain  
  
#####  
## Primary SIP proxy and SIP registrar settings
```

```
##
## Proxy address : IP address, FQDN or domain name
## Registrar address : IP address, FQDN or domain name (usually, the proxy)
## SIP proxy UDP port : usually 5060
## SIP registrar UDP port : by default 5060
#####

proxy_addr=192.168.1.80
proxy_port=5060
registrar_addr=192.168.1.80
registrar_port=5060
outbound_proxy_addr=
outbound_proxy_port=

#####
## Redundancy settings
##
## Proxy address : IP address, FQDN or domain name
## Registrar address : IP address, FQDN or domain name (usually, the proxy)
## SIP proxy UDP port : usually 5060
## SIP registrar UDP port : by default 5060
## sip_transport_mode_survi : Transport mode in PCS mode
##      0 = UDP or TCP
##      1 = UDP
##      2 = TCP
#####

proxy2_addr=192.168.1.80
proxy2_port=5060
registrar2_addr=192.168.1.80
registrar2_port=5060
outbound_proxy2_addr=
outbound_proxy2_port=
pcs_addr=192.168.1.80
pcs_port=5060
sip_transport_mode_survi=0 # in our case, it will be UDP
option_timer=120

#####
## Global SIP parameters
## Transport mode : 0 = UDP or TCP
##      1 = UDP
##      2 = TCP
## local_rtp_port : RFC3605 is not supported in this release, so
##      only default value can be used
## PRACK type : 0 = PRACK supported
##      1 = PRACK required
##      2 = PRACK disabled
## Codec settings : 0 = G711 (PCMU)
##      4 = G723.1
##      8 = G711 (PCMA)
##      18 = G729A
#####

register_expire=3600
register_retry=300
local_sip_port=
sip_transport_mode=0 # dans notre cas, ce sera de l'UDP
local_rtp_port=42000
local_rtcp_port=42001
prack_type=0
preferred_vocoder=8,0,4,18
```

```
#####  
## SIP authentication.  
##  
## Realm : If no authentication, leave empty  
## Authentication name : HAS TO BE FILLED  
##           If no authentication, PUT A VALUE LIKE none  
## Authentication password : If no authentication, leave empty  
#####  
  
authentication_realm=192.168.1.80 # authentication is done on Asterisk  
authentication_name=alcatel      # username and password  
authentication_password=11111111  
user_name=alcatel  
display_name=Alcatel IP Touch    # display name when calling  
  
[...]  
  
[sntp]  
  
#####  
## SNTP server settings (can be OXE or an external server)  
##  
## Timezone construction : UT::60:032802:103103 (Paris - 2021)  
##           GMT delta : 60 = + sixty minutes from GMT time  
##           Daylight saving start (mmdh) : 032902 = 28 March 2am  
##           Daylight saving end (mmdh) : 103103 = 31 October 3am  
## The daylight saving settings HAVE to be changed each year.  
#####  
  
sntp_addr=192.168.1.254          # To synchronise the phone's time  
timezone=UT::60:032802:103103   # with the Freebox's built-in NTP server  
                                # The time zone is to be changed every year,  
                                # it is set according to the GMT zone and  
                                # manages summer and winter time.  
  
[...]  
  
[init]  
  
#####  
## For IP Touch with SIP binary in 1.xx, 2.00.10 and 2.00.20, equal or greater  
## than 2.00.81  
##     mode 0 = SIP  
##     mode 1 = NOE  
##  
## For IP Touch with SIP binary 2.00.30 to 2.00.80  
##     mode 0 = NOE  
##     mode 1 = SIP  
#####  
  
application_mode=0              # SIP mode (depending on firmware version)  
  
[audio]  
  
#####  
## Tone country : 0 = English  
##           1 = French  
##           2 = German  
##           3 = Italian  
##           4 = Spanish  
##           5 = Dutch  
##           6 = Portuguese  
## DTMF type : 0 = RFC2833
```

```
##          1 = In-band
##          2 = SIP INFO
## DTMF level / RLR handset / SLR handset / Sidetone handset :
## 0 = 0db, 1 = +3db, 2 = +6db, 3 = -3db, 4 = -6db
## VAD / DTMF feedback / Hearing Aid :
## 0 = VAD not used
## 1 = VAD used
#####

tone_country=1    # to be set according to the standards used in France
dtmf_type=1
dtmf_level=0
dtmf_avt_payload_type=96
vad=0
dtmf_feedback_enable=0
rlr_handset=0
slr_handset=0
sidetone_handset=2
hearing_aid_enable=0

[appl]

#####
## Password to access the administrator menu on the phone (digits only)
## Power priority : 1 = critical
##                 2 = high
##                 3 = low
## Time format : 0 = 24 hours format
##               1 = AM / PM
## Speed dial numbers (first and last name, URI)
#####

admin_password=000000    # admin password when pressing i then #.
bluetooth_parameters=blue
supported_language=0
remote_forward_code=
remote_forward_deactive_code=
power_priority=
asset_id=
time_format=0
speed_dial_1_first_name=
speed_dial_1_last_name=
speed_dial_1_uri=
speed_dial_2_first_name=
speed_dial_2_last_name=
speed_dial_2_uri=
speed_dial_3_first_name=
speed_dial_3_last_name=
speed_dial_3_uri=
speed_dial_4_first_name=
speed_dial_4_last_name=
speed_dial_4_uri=
[...]
```

1. Download all these 2 firmware files (*noesip4018* and *datsip4018*, found on the Alcatel Unleashed forum) and the 2 configuration files (*sipconfig.txt* and *sipconfig-MacAddress.txt*), available in the project repository: <https://github.com/guillaumenibert/VoIP-Asterisk-WebRTC-SIP/tree/main/iptouch4018> [ee](#).

2. Place them in the `/var/www/html` of the Debian virtual machine.
3. Connect the phone to the PoE injector to turn it on. The Alcatel IP Touch will fetch the latest firmware from the `HTTP NGINX` server, the configurations and will be connected to the Asterisk server.

Client-side testing

If the phone displays the date and time and does not restart in a loop, it is connected to the Asterisk server.

Server-side testing

On a server console, type the command `sudo asterisk -rvvv`

Once entered, type the command `pjsip show endpoints`. If the Raspberry Pi's SIP client is connected then the console will return this:

Output

```
asterisktz*CLI> pjsip show endpoints

Endpoint: <Endpoint/CID.....> <State.....>
<Channels.>
  I/OAuth:
<AuthId/UserName.....>
  Aor: <Aor.....> <MaxContact>
  Contact: <Aor/ContactUri.....> <Hash....> <Status>
<RTT(ms)..>
  Transport: <TransportId.....> <Type> <cos> <tos>
<BindAddress.....>
  Identify:
<Identify/Endpoint.....>
  Match: <criteria.....>
  Channel: <ChannelId.....> <State.....>
<Time.....>
  Exten: <DialedExten.....> CLCID: <ConnectedLineCID.....>
=====
==

Endpoint:  alcatel/5001                                Not in use      0 of inf
  InAuth:  alcatel/alcatel
  Aor:     alcatel                                     1
  Contact: alcatel/sip:alcatel@192.168.1.81          8c02c0558c NonQual  nan

Endpoint:  guillaume/5003                               Unavailable     0 of inf
  InAuth:  guillaume/guillaume
  Aor:     guillaume                                  1

Endpoint:  rpi/5002                                    Not in use      0 of inf
  InAuth:  rpi/rpi
  Aor:     rpi                                        1
  Contact: rpi/sip:rpi@192.168.1.82;transport=udp    cec2f9dd2f NonQual  nan

Objects found: 3
```

The client has an IP address of **192.168.1.81** and is connected. The information **“Not in use”** indicates that there is no call in progress.

5. Communication tests

All endpoints are connected to the Asterisk server. It is therefore possible to call from the Raspberry Pi to the Alcatel or from the Alcatel to the Raspberry Pi.

In order to carry out the communication, you must first have launched the Asterisk server, the TFTP server, turned on all the peripherals and connected an audio output on the Raspberry Pi (audio jack, Bluetooth or HDMI) to be able to listen to the audio stream.

Raspberry Pi to Alcatel IP Touch

1. Launch a terminal and run *linphonec*.

```
pi@raspberrypi:~$
```

Executing *linphonec*

```
linphonec
```

2. Call the Alcatel IP Touch phone, it has the number 5001 (see [part 2 - configuration and user creation](#)).

```
linphonec> call 5001
```

Output (with comments)

```
# Error message, not important, video is disabled, we are only doing VoIP.
2021-02-09 13:30:36:367 ortp-error-LinphoneCore has video disabled for both
capture and display, but video policy is to start the call with video. This is a
possible mis-use of the API. In this case, video is disabled in default
LinphoneCallParams

# Linking to the Alcatel phone
Establishing call id to sip:5001@192.168.1.80, assigned id 1
# The Alcatel phone has been found, it is contacted, it rings on the Alcatel
side.
Contacting sip:5001@192.168.1.80
linphonec> Call 1 to sip:5001@192.168.1.80 in progress.
linphonec> Call 1 with sip:5001@192.168.1.80 connected.
# We picked up the Alcatel phone.
Call answered by sip:5001@192.168.1.80
# Communication is in progress, audio is playing, settings are adjusted.
linphonec> 2021-02-09 13:30:36:563 ortp-error-no such method on filter
MSPulseWrite, fid=16394 method index=2
Media streams established with sip:5001@192.168.1.80 for call 1 (audio).
Call is updated by remote.
linphonec> 2021-02-09 13:30:40:761 ortp-error-no such method on filter
MSPulseWrite, fid=16394 method index=2
Call parameters were successfully modified.
linphonec> Media streams established with sip:5001@192.168.1.80 for call 1
(audio).
Call is updated by remote.
linphonec> Call parameters were successfully modified.
```

```
linphone> Media streams established with sip:5001@192.168.1.80 for call 1  
(audio).  
# The call has just ended, someone has hung up one of the devices.  
Call terminated.  
linphone> Call 1 with sip:5001@192.168.1.80 ended (No error).
```

When the call is initiated, the Alcatel phone screen displays the following:



(Figure 15 - Call from Raspberry Pi to Alcatel IP Touch 4018 EE)

The communication therefore works in one direction. Let's see what happens if the Alcatel phone calls the Raspberry Pi.

Alcatel IP Touch to Raspberry Pi

1. From the phone, call the Raspberry Pi's number 5002 (see **part 2 - configuration and user creation**).



(Figure 16 - Call from Alcatel IP Touch 4018 EE to Raspberry Pi)

2. From the Raspberry Pi terminal, make sure that **linphonec** is active. When the Alcatel launches its call, it is received in the terminal:

Output

```
linphonec> Receiving new incoming call from "Alcatel IP Touch"  
<sip:5001@192.168.1.80>, assigned id 3
```

To answer it, just type **answer** and the call **ID**:

```
answer 3
```

The communication is launched and works in the same way.

Output

```
linphonec> Receiving new incoming call from "Alcaltel IP Touch"  
<sip:5001@192.168.1.80>, assigned id 3  
answer 3  
Connected.  
linphonec> Call 3 with "Alcaltel IP Touch" <sip:5001@192.168.1.80> connected.  
2021-02-09 13:46:28:345 ortp-error-no such method on filter MSPulseWrite,  
fid=16394 method index=2  
Media streams established with "Alcaltel IP Touch" <sip:5001@192.168.1.80> for  
call 3 (audio).  
linphonec> Call is updated by remote.  
linphonec> 2021-02-09 13:46:28:424 ortp-error-no such method on filter  
MSPulseWrite, fid=16394 method index=2  
Call parameters were successfully modified.  
linphonec> Media streams established with "Alcaltel IP Touch"  
<sip:5001@192.168.1.80> for call 3 (audio).  
Call is updated by remote.  
linphonec> Call parameters were successfully modified.  
linphonec> Media streams established with "Alcaltel IP Touch"  
<sip:5001@192.168.1.80> for call 3 (audio).  
Call terminated.  
linphonec> Call 3 with "Alcaltel IP Touch" <sip:5001@192.168.1.80> ended (No  
error).
```

The communications therefore work in both directions. The objective of the next part is to create a graphical interface in JavaScript on the Raspberry Pi side, more user-friendly than the *linphonec* command line client.

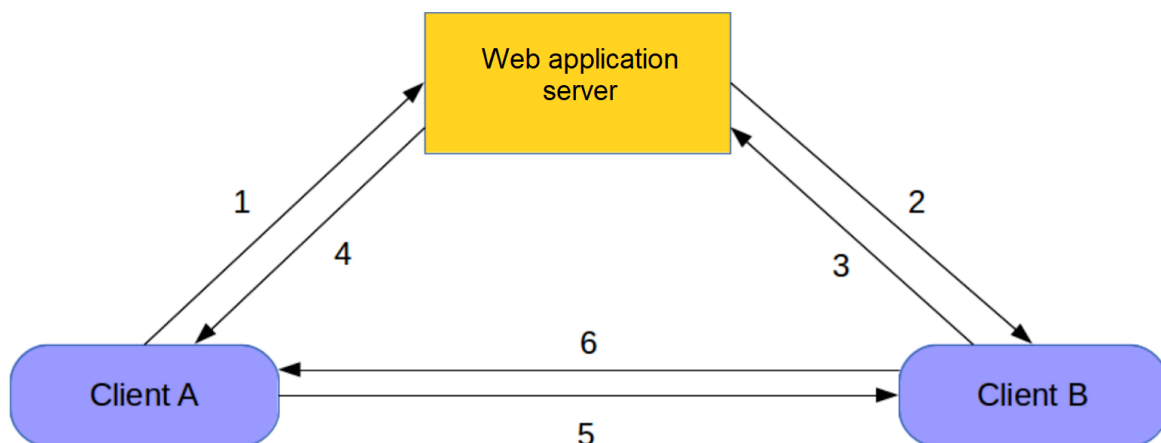
6. Development of a JavaScript SIP client using WebRTC

The establishment of the communication (SIP) as well as the communication itself (RTP) works correctly. The development of a SIP client program in JavaScript will require modifications to our environment. Indeed, we are going to make a call from a web browser supporting JavaScript to another device (having a JavaScript SIP client or not). Before starting the implementation. It is necessary to understand what the WebRTC, WebSocket APIs are.

WebRTC & WebSocket

The WebRTC (*Web Real-Time Communication*) API is a software interface whose purpose is to link two devices so that they can communicate directly. This connection requires opening a communication channel between a client and a server: the technology that allows this is the WebSocket API.

In concrete terms, establishing a connection works in a similar way to SIP. Below is an adapted explanatory diagram from Wikipedia.



(Figure 17 - Establishing a connection between two clients)

- “1: A asks the server for a connection with B.
- 2: The server relays the request from A to B.
- 3: If B accepts, it sends a connection request to A.
- 4: The server relays the request to A.
- 5 and 6: Bidirectional PeerConnection is established.” - [Wikipédia](#).

The PeerConnection corresponds in our case to the RTP flow between the two clients. Once the communication is established, as with SIP, the communication between the two clients is direct and the media flows do not pass through the application web server.

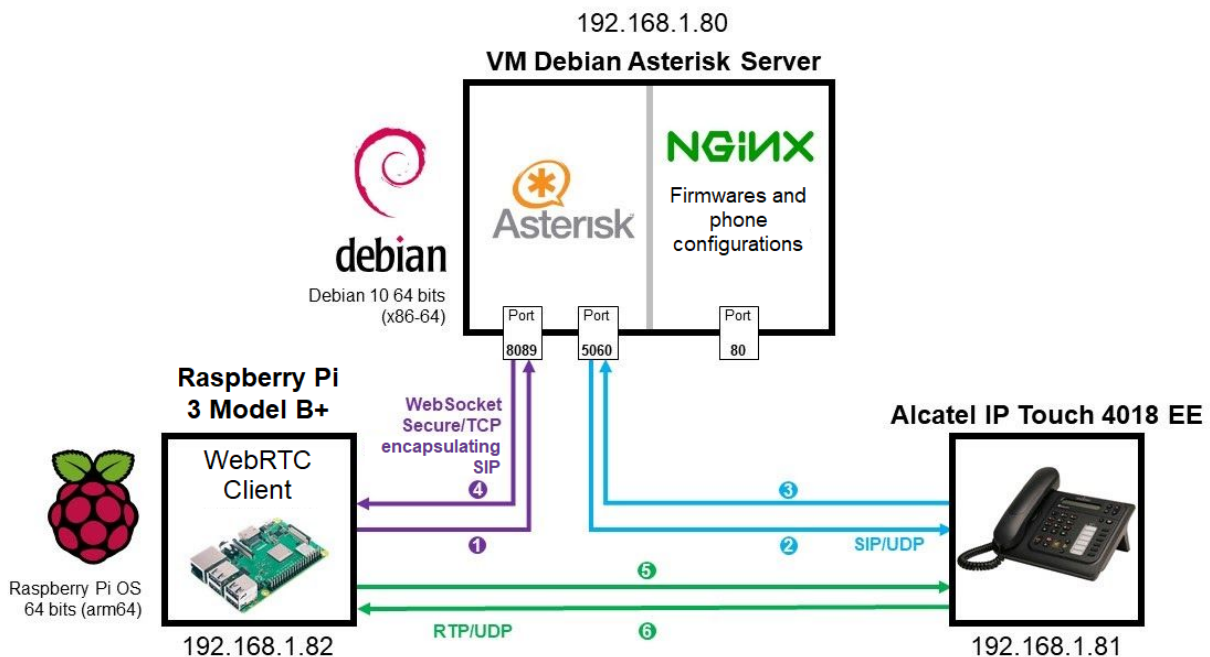
Let's imagine now:

- Client A: Raspberry Pi, with Mozilla Firefox web browser supporting WebRTC.
- Client B: Alcatel IP Touch 4018 EE, not supporting WebRTC.

How can the two endpoints communicate with each other?

The Alcatel phone cannot support WebRTC, it is proprietary hardware, the source code is closed.

On the Raspberry Pi side, however, it is possible to use WebRTC and SIP by encapsulating the SIP protocol in a WebSocket. This is defined in RFC 7118¹³ and requires a server that can handle WebRTC/SIP for client A and only SIP for client B. The Asterisk server supports WebRTC with SIP. Therefore, modifications are required to make the server capable of supporting WebRTC.



(Figure 18 - Raspberry Pi calls Alcatel IP Touch from a WebRTC client)

At the Raspberry Pi client level, web browsers such as Mozilla Firefox, Safari or those based on Chromium natively implement the WebRTC API. In this project, Mozilla Firefox will be used as a client using WebRTC.

¹³ RFC 7118: *The WebSocket Protocol as a Transport for the Session Initiation Protocol (SIP)*, available at: <https://tools.ietf.org/html/rfc7118>.

Configuring the Asterisk server to support the WebRTC API

In order to improve the security between the WebRTC client and the Asterisk server, a secure web socket (WSS) via TLS will be set up. We will therefore first generate a self-signed certificate.

Generating a self-signed SSL/TLS certificate

In order to improve security and modernise, we adopt a certificate generated with ECDSA algorithms, which are much more efficient than the classic RSA algorithms **(11)**. We will use the ECDSA P-521 algorithm, recommended by the ANSSI (French National Agency for the Security of Information Systems) **(12)** and compatible with Mozilla Firefox¹⁴.

To perform this operation, you must first have started the Debian virtual machine and have the OpenSSL tool.

¹⁴ Mozilla Firefox uses NSS (Network Security Services), a library that supports this algorithm.

1. Connect via SSH to the `asterisktz` machine.

```
# ssh login@vm_ip_address -p 22  
ssh asterisktz@192.168.1.80 -p 22
```

2. Create the folders in which the Certificate Authority's certificate, called the root certificate (`ca`), the certificate associated with the IP `192.168.1.80` (`certs`) and the certificate signing request file to the authority (`csr`) are stored.

```
asterisktz@asterisktz:~$ Creating folder
```

```
mkdir ca && mkdir certs && mkdir csr
```

3. Creation of certificates.

```
asterisktz@asterisktz:~$ Create the private key of the root certificate (certification authority)
```

```
openssl ecparam -genkey -name secp521r1 -out ca/TZVoIP-Root-CA.key
```

```
asterisktz@asterisktz:~$ Generate the root certificate from its private key
```

```
openssl req -x509 -new -nodes -key ca/TZVoIP-Root-CA.key -sha384 -days  
3650 -utf8 -out ca/TZVoIP-Root-CA.crt
```

Information to be filled in:

```
You are about to be asked to enter information that will be incorporated  
into your certificate request.  
What you are about to enter is what is called a Distinguished Name or a DN.  
There are quite a few fields but you can leave some blank  
For some fields there will be a default value,  
If you enter '.', the field will be left blank.  
-----
```

```
Country Name (2 letter code) [AU]:FR  
State or Province Name (full name) [Some-State]:Hauts-de-France  
Locality Name (eg, city) []:Compiègne  
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Université de  
Technologie de Compiègne  
Organizational Unit Name (eg, section) []:TZ VoIP  
Common Name (e.g. server FQDN or YOUR name) []:TZ VoIP Root  
Email Address []:guillaume.nibert@etu.utc.fr
```

```
asterisktz@asterisktz:~$ Generate the private key of the IP address certificate and its signature  
request file.
```

```
openssl req -new -sha384 -nodes -utf8 -out csr/asterisktz.csr -newkey  
ec:<(openssl ecparam -name secp521r1) -keyout certs/asterisktz.key
```

Information to be filled in:

```
Generating an EC private key
writing new private key to 'certs/asterisktz.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:Hauts-de-France
Locality Name (eg, city) []:Compiègne
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Université de
Technologie de Compiègne
Organizational Unit Name (eg, section) []:TZ VoIP
Common Name (e.g. server FQDN or YOUR name) []:192.168.1.80
Email Address []:guillaume.nibert@etu.utc.fr

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

```
asterisktz@asterisktz:~$ Create the file containing the parameters of the certificate to be created.
```

```
nano csr/openssl-v3.cnf
```

csr/openssl-v3.cnf

```
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment,
dataEncipherment
subjectAltName = @alt_names

[alt_names]
IP.1 = 192.168.1.80
```

```
asterisktz@asterisktz:~$ Generate the certificate and signature with the certification authority.
```

```
openssl x509 -req -in csr/asterisktz.csr -CA ca/TZVoIP-Root-CA.crt -CAkey
ca/TZVoIP-Root-CA.key -CAcreateserial -out certs/asterisktz.crt -days 365
-sha384 -extfile csr/openssl-v3.cnf
```

```
asterisktz@asterisktz:~$ Produce the full-chain certificate.
```

```
cat certs/asterisktz.crt certs/asterisktz.key > certs/asterisktz.pem
```

```
asterisktz@asterisktz:~$ Restriction to read-only rights on this certificate to prevent modification
```

```
chmod a+r certs/asterisktz.pem
```

The self-signed certificate has been created. For more details about the certification process for a non-self-signed certificate, see: <https://letsencrypt.org/how-it-works/>. Let's move on to enabling the HTTP server built into Asterisk.

Enabling the Asterisk HTTP server

On the same machine:

1. Edit the configuration file for Asterisk's built-in HTTP server.

```
asterisktz@asterisktz:~$ Editing the configuration file for Asterisk's built-in HTTP server.
```

```
sudo nano /etc/asterisk/http.conf
```

2. Replace the contents with:

/etc/asterisk/http.conf

```
[general]
enabled=no
tlsenable=yes
tlsbindaddr=0.0.0.0:8089
tlscertfile=/home/asterisktz/certs/asterisktz.crt
tlsprivatekey=/home/asterisktz/certs/asterisktz.key
enablestatic=no
sessionlimit=1000
```

Here only encrypted connections (***tlsenable=yes***) are allowed on port 8089. Unencrypted connections would be ***enabled*** to ***yes*** on port 8088.

3. Restart the Asterisk service to enable the built-in HTTP server.

```
asterisktz@asterisktz:~$ Application des changements
```

```
sudo systemctl restart asterisk
```

To check that the HTTP server is enabled, simply type the command `sudo asterisk -rvvv`, then the command `http show status`. It should return this:

```
asterisktz*CLI> http show status
HTTP Server Status:
Prefix:
Server: Asterisk/18.2.0
Server Disabled

Enabled URI's:
/httpstatus => Asterisk HTTP General Status
/phoneprov/... => Asterisk HTTP Phone Provisioning Tool
/metrics/... => Prometheus Metrics URI
/ari/... => Asterisk RESTful API
/ws => Asterisk HTTP WebSocket

asterisktz*CLI>
```

The element we are interested in: the use of WebSocket for SIP (*/ws*).
Let's move on to the configuration of *pjsip.conf* and *extensions.conf* to take into account both WebRTC and SIP. We have based ourselves on the *BrowserPhone*¹⁵ project and have adapted the configuration files in question. The files are available in the *asterisk_webrtc* directory of the GitHub repository.

Editing *pjsip.conf* to support WebRTC

1. Edit the *pjsip.conf* configuration file.

```
asterisktz@asterisktz:~$
```

Editing the configuration file *pjsip.conf*

```
sudo nano /etc/asterisk/pjsip.conf
```

2. Replace content with:

/etc/asterisk/pjsip.conf

```
[global]
max_forwards=70
user_agent=AsteriskTZ
default_realm=192.168.1.80
keep_alive_interval=300

; == Transport

[udp_transport]
type=transport
protocol=udp
bind=0.0.0.0
tos=af42
cos=3
```

¹⁵ BrowserPhone: <https://github.com/InnovateAsterisk/Browser-Phone>.

```
[wss_transport]
type=transport
protocol=wss
bind=0.0.0.0

[tcp_transport]
type=transport
protocol=tcp
bind=0.0.0.0

[tls_transport]
type=transport
protocol=tls
bind=0.0.0.0
cert_file=/home/asterisk/certs/asterisktz.crt
priv_key_file=/home/asterisk/certs/asterisktz.key
cipher=ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA
384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:D
HE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384;
method=tlsv1_2

; == ACL

[ac1] ; Communications are only allowed in Class A, B and C local networks.
type=acl
deny=0.0.0.0/0.0.0.0
permit=10.0.0.0/255.0.0.0
permit=172.16.0.0/255.240.0.0
permit=192.168.0.0/255.255.0.0

; Templates

[single_aor](!)
max_contacts=1
qualify_frequency=120
remove_existing=yes

[userpass_auth](!)
auth_type=userpass

[basic_endpoint](!)
moh_suggest=default
context=from-extensions
inband_progress=no
rtp_timeout=120
message_context=textmessages
allow_subscribe=yes
subscribe_context=subscriptions
direct_media=yes
dtmf_mode=rfc4733
```

```
device_state_busy_at=1
disallow=all

[phone_endpoint] (!)
allow=ulaw,alaw

[webrtc_endpoint] (!)
transport=wss_transport
allow=ulaw,alaw
dtls_auto_generate_cert=yes
webrtc=yes

; Users

[alcatel] (basic_endpoint,phone_endpoint)
type=endpoint
callerid="Alcatel IP Touch" <5001>
auth=alcatel
aors=alcatel
[alcatel] (single_aor)
type=aor
[alcatel] (userpass_auth)
type=auth
username=alcatel
password=11111111

[rpi] (basic_endpoint,webrtc_endpoint)
type=endpoint
callerid="Raspberry Pi" <5002>
auth=rpi
aors=rpi
[rpi] (single_aor)
type=aor
[rpi] (userpass_auth)
type=auth
username=rpi
password=22222222

[guillaume] (basic_endpoint,webrtc_endpoint)
type=endpoint
callerid="Guillaume Nibert" <5003>
auth=guillaume
aors=guillaume
[guillaume] (single_aor)
type=aor
mailboxes=guillaume@default
[guillaume] (userpass_auth)
type=auth
username=guillaume
password=33333333
```

For more details, please refer to the PJSIP documentation:

<https://wiki.asterisk.org/wiki/display/AST/PJSIP+Configuration+Sections+and+Relationships>.

Editing *extensions.conf* to support WebRTC

3. Edit the configuration file *extensions.conf*.

```
asterisktz@asterisktz:~$
```

Editing the configuration file *extensions.conf*

```
sudo nano /etc/asterisk/extensions.conf
```

4. Replace the contents with:

/etc/asterisk/extensions.conf

```
[general]
static=yes
writeprotect=yes
priorityjumping=no
autofallthrough=no

[globals]
ATTENDED_TRANSFER_COMPLETE_SOUND=beep

[textmessages] ; Allows you to send text for WebRTC clients
exten => 5002,1,Gosub(send-text,s,1(rpi))
exten => 5003,1,Gosub(send-text,s,1(guillaume))

[subscriptions] ; Allows to know the status of an endpoint (in call or available)
exten => 5001, hint, PJSIP/alcatel
exten => 5002, hint, PJSIP/rpi
exten => 5003, hint, PJSIP/guillaume

[from-extensions]
; When you call 5000, you get music.
exten => 5000,1,Gosub(moh,s,1)
; Extensions
exten => 5001,1,Gosub(dial-extension,s,1,(alcatel))
exten => 5002,1,Gosub(dial-extension,s,1,(rpi))
exten => 5003,1,Gosub(dial-extension,s,1,(guillaume))
; If you have anything other than 5000, 5001, 5002 or 5003 then it is a wrong
; number, so hang up.
exten => _[+*0-9].,1,NoOp(You called: ${EXTEN})
exten => _[+*0-9].,n,Hangup(1)

exten => e,1,Hangup()

[moh] ; "function" for music (see 5000). Note: "function" is an abusive term, it
; is actually called "context".
```



```
exten => s,1,NoOp(Music On Hold)
exten => s,n,Ringing()
exten => s,n,Wait(2)
exten => s,n,Answer()
exten => s,n,Wait(1)
exten => s,n,MusicOnHold()

[dial-extension] ; "function" to call an endpoint.
exten => s,1,NoOp(Calling: ${ARG1})
exten => s,n,Set(JITTERBUFFER(adaptive)=default)
exten => s,n,Dial(PJSIP/${ARG1},30)
exten => s,n,Hangup()

exten => e,1,Hangup()

[send-text] ; "function" to send text.
exten => s,1,NoOp(Sending Text To: ${ARG1} From: ${MESSAGE(from)})
exten => s,n,Set(PEER=${CUT(CUT(CUT(MESSAGE(from),@,1),<,2),:,)})
exten => s,n,Set(FROM=${SHELL(asterisk -rx 'pjsip show endpoint ${PEER}' | grep 'callerid '
| cut -d':' -f2- | sed 's/^\ *//' | tr -d '\n'))
exten => s,n,Set(CALLERID_NUM=${CUT(CUT(FROM,<,1),<2)})
exten => s,n,Set(FROM_SIP=${STRREPLACE(MESSAGE(from),<sip:${PEER}@,<sip:${CALLERID_NUM}@)})
exten => s,n,MessageSend(pjsip:${ARG1},${FROM_SIP})
exten => s,n,Hangup()
```

For more details, please refer to the documentation on the configuration of the dial plan:
<https://wiki.asterisk.org/wiki/display/AST/Contexts%2C+Extensions%2C+and+Priorities>.

5. Restart the *asterisk* service.

```
asterisktz@asterisktz:~$
```

Restarting the asterisk service.

```
sudo systemctl restart asterisk
```

Communication tests with the Web *Browser Phone* client

After restarting the asterisk service. We will go to the Raspberry Pi and install Mozilla Firefox via the command `sudo apt install firefox-esr`.

Let's open Mozilla Firefox and enter the following URL in the address bar:
<https://www.innovateasterisk.com/phone/>.

You must configure the fields as follows and then click on **Save**:

Asterisk Server Address:
192.168.1.80

WebSocket Port:
8089

WebSocket Path:
/ws

Subscribe Extension (Internal):
5002

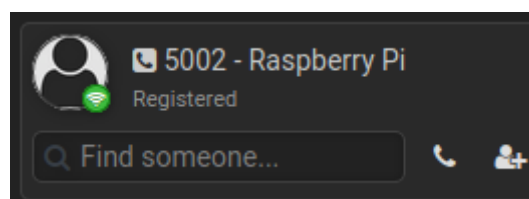
Full Name:
Raspberry Pi

SIP Username:
rpi

SIP Password:
.....

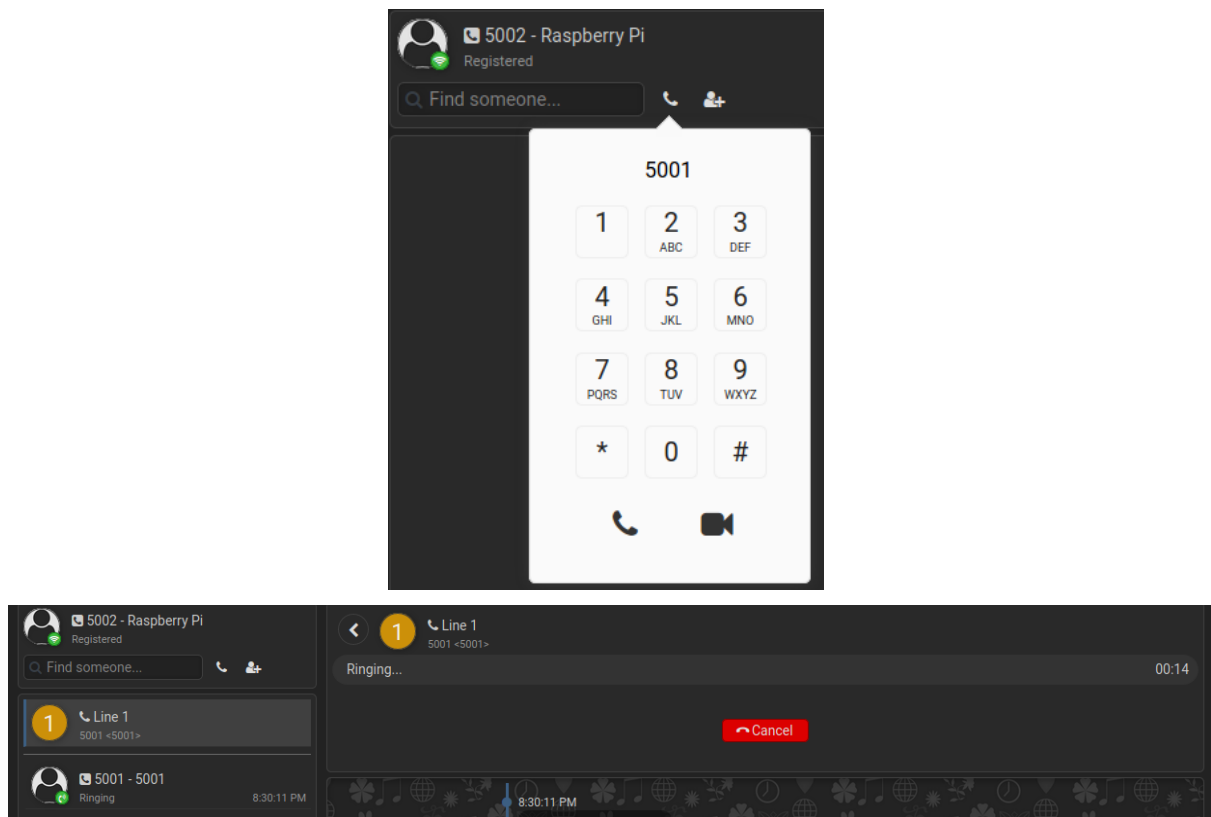
(Figure 19 - Configuring the WebRTC Browser Phone client)

The *Registered* indication indicates that the client is indeed connected to the Asterisk server.



(Figure 20 - Registering the *rpi* client on the Asterisk server)

So we can call the Alcatel IP Touch.



(Figure 21 - Raspberry Pi to Alcatel IP Touch 4018 EE call)

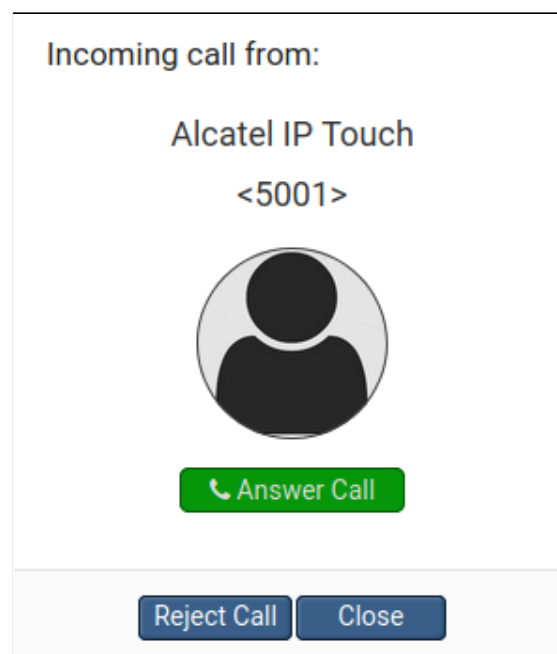


(Figure 22 - Receiving the call from the Raspberry Pi)

Or from the Alcatel phone, call the Raspberry Pi.



(Figure 23 - Alcatel IP Touch 4018 EE to Raspberry Pi call)



(Figure 24 - Receiving the call from the Alcatel IP Touch)

Development of a JavaScript SIP client

Given the project and time constraints, we only had time to test existing solutions (*Browser Phone*). However, the development of a SIP client can be done via already existing libraries such as: [SIP.js](#), [JsSip](#), [sipML5](#)... For information *Browser Phone* uses SIP.js.

Conclusion

This project allowed the development of a VoIP communication between a Raspberry Pi and an Alcatel IP Touch 4018 EE phone. We discovered new protocols such as SIP or RTP and the WebRTC API, which allows us to encapsulate SIP to establish a communication between several endpoints, using a web browser or not. It is easy to see the potential of IP for communications, and the need to use this type of technology in companies or public services.

This project could be improved by the implementation of encryption in particular with the SIPS protocol (SIP over SSL/TLS) at the level of the establishment of the communication between two end points (done in part when encapsulated in a WebSocket over TLS), but this could be integral, since the Alcatel supports SIP over TLS very well. Finally, the RTP protocol can also be encrypted, this is the SRTP (*Secure Real-time Transport Protocol*) when there is a call in progress.

I would like to thank Mr Lounis for having proposed this experimental work, which was particularly enriching, as I was unaware of a large part of the functioning of IP telephony. This subject allowed me to manipulate both the back end and front end of the system and to measure the power of this architecture when it is integrated into the Internet network.

Table of illustrations (excluding appendixes)

Figure 1 - Overall infrastructure diagram.....	4
Figure 2 - Internet protocol stack.....	6
Figure 3 - Establishment and termination of a VoIP communication using SIP - attribution: 3cx.com.....	7
Figure 4 - Establishing a telephone call between the Raspberry Pi and the Alcatel IP Touch 4018 EE telephone.....	7
Figure 5 - PBX Matra MC6500 serie.....	8
Figure 6 - Area code setting.....	11
Figure 7 - Selection of Asterisk sound modules.....	13
Figure 8 - Selection of Asterisk extra sound modules.....	14
Figure 9 - Setting asterisk as the default user of the Asterisk service.....	16
Figure 10 - Setting asterisk as the default user of the Asterisk service.....	17
Figure 11 - Launch of the Asterisk service with errors.....	18
Figure 12 - Launch of the Asterisk service without errors.....	19
Figure 13 - Retrieving Alcatel phone configuration files via HTTP.....	28
Figure 14 - Alcatel-Lucent, 3. SIP stand-alone mode In: IP Touch 4008/4018 Extended Edition - SIP Phone Installation Guide - 8AL90824AAAA ed02, p.4-5, August 2010.....	30
Figure 15 - Call from Raspberry Pi to Alcatel IP Touch 4018 EE.....	40
Figure 16 - Call from Alcatel IP Touch 4018 EE to Raspberry Pi.....	41
Figure 17 - Establishing a connection between two clients.....	43
Figure 18 - Raspberry Pi calls Alcatel IP Touch from a WebRTC client....	44
Figure 19 - Configuring the WebRTC Browser Phone client.....	54
Figure 20 - Registering the rpi client on the Asterisk server.....	54
Figure 21 - Raspberry Pi to Alcatel IP Touch 4018 EE call.....	55
Figure 22 - Receiving the call from the Raspberry Pi.....	55
Figure 23 - Alcatel IP Touch 4018 EE to Raspberry Pi call.....	56
Figure 24 - Receiving the call from the Alcatel IP Touch.....	56

Abbreviations (excluding appendixes)

Abbreviation	Description
802.11	IEEE 802.11 (Wi-Fi) standard.
802.3	IEEE 802.3 (Ethernet) standard.
ANSSI	Agence nationale de la sécurité des systèmes d'information (French National Agency for the Security of Information Systems).
ECDSA	Elliptic Curve Digital Signature Algorithm, asymmetric digital signature algorithm using elliptic curve cryptography.
FTP	File Transfer Protocol.
HDMI	High-Definition Multimedia Interface.
HTTP	Hypertext Transfer Protocol.
HTTPS	HyperText Transfer Protocol Secure.
IEEE	Institute of Electrical and Electronics Engineers.
IP	Internet Protocol, network layer of the TCP/IP model.
IP PBX	Internet Protocol Private Vbranch eXchange, this is PBX operating on the internet stack.
ISDN	Integrated Services Digital Network, digital telephone network with speeds of up to 2 Mbit/s (Wikipedia).
LTS	Long Term Support, long term supported software/system version.
MAC	Media Access Control, data link layer protocol of the OSI model.
NAT	Network Address Translation.
P-521	Encryption algorithm using elliptic curves developed by the National Institute of Standards and Technology.
PBX	Private Branch eXchange, used to link telephone endpoints.
PHY	Physical layer of the OSI model.
PoE	Power over Ethernet, IEEE 802.3af standard, a PoE device allows a device to be powered over Ethernet while retaining the ability to transfer data.
PSTN	Public Switched Telephone Network, analog telephone network.
RFC	Request for comments, official document specifying Internet technologies.

RPi	Raspberry Pi.
RSA	"RSA (Rivest–Shamir–Adleman) is a public-key cryptosystem that is widely used for secure data transmission. It is also one of the oldest. The acronym "RSA" comes from the surnames of Ron Rivest, Adi Shamir and Leonard Adleman" - Wikipedia .
RTP	application protocol allowing, among other things, the transfer of audio or video streams.
SIP	Session Initiation Protocol, protocol establishing VoIP communication between two endpoints.
SIPS	Session Initiation Protocol over SSL/TLS.
SMTP	Simple Mail Transfer Protocol.
SRTP	Secure Real-time Transport Protocol.
SSH	Secure Shell, application communication protocol.
SSL	Secure Socket Layer, protocol for securing exchanges.
TCP	Transmission Control Protocol, transport layer protocol of the OSI model (connected mode).
TFTP	Trivial File Transfer Protocol, application protocol allowing the transfer of files by UDP.
TLS	Transport Layer Security, successor of SSL.
TZ	Teaching unit specific to the University of Technology of Compiègne, it consists of an experimental project carried out by a student, supervised by a teacher.
UDP	User Datagram Protocol, transport layer protocol of the OSI model (unconnected mode).
VM	Virtual Machine.
VoIP	Voice over Internet Protocol.
WebRTC	"WebRTC (Web Real-Time Communication) is a free and open-source project providing web browsers and mobile applications with real-time communication (RTC) via application programming interfaces (APIs)" - Wikipedia .
WS	"WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection" - Wikipedia .

References

- (1) Wikipedia, *Business telephone system*, 21st january 2022, available at: https://en.wikipedia.org/wiki/Business_telephone_system.
- (2) ITU Telecommunication Standardization Sector, *ISDN user-network interface layer 3 specification for basic call control*, ITU, May 1998, available at: <https://www.itu.int/rec/T-REC-Q.931-199805-I/en>.
- (3) ITU Telecommunication Standardization Sector, *Interface between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE) for terminals operating in the packet mode and connected to public data networks by dedicated circuit*, ITU, October 1996, available at: <https://www.itu.int/rec/T-REC-X.25-199610-I/en>.
- (4) Matt Fredrickson, *PSA: chan_sip status changed to “deprecated” & Asterisk 17.0.0-rc2 Release*, Asterisk.org, 25th september 2019, available at: https://www.asterisk.org/deprecating-chan_sip-asterisk-17-0-0-rc2-release/.
- (5) Y. Yeryomin, F. Evers and J. Seitz, *II. The NAT and firewall problem In: Solving the firewall and NAT traversal issues for SIP-based VoIP*, International Conference on Telecommunications, p.1-2, July 2008, DOI: 10.1109/ICTEL.2008.4652645, available at: https://www.researchgate.net/publication/224341038_Solving_the_firewall_and_NAT_traversal_issues_for_SIP-based_VoIP.
- (6) Alcatel-Lucent, *Audio characteristics In: Alcatel-Lucent IP Touch 4008/4018 Extended Edition Phones*, p.2, 2013, available at: https://assets.bmdstatic.com/assets/Data/brochure/SKU01413265_2.pdf.
- (7) Wikipédia, *G.711*, 20th august 2021, available at: <https://en.wikipedia.org/wiki/G.711>.
- (8) Malcolm Davenport, *Answer, Playback, and Hangup Applications In: Asterisk Documentation*, Asterisk.org, 19th december 2013, available at: <https://wiki.asterisk.org/wiki/display/AST/Answer%2C+Playback%2C+and+Hangup+Applications>.
- (9) bluesman, *64 bit Raspberry Pi OS is here!*, Audiophile Style, 4th june 2020, available at: <https://audiophilestyle.com/forums/topic/59499-64-bit-raspberry-pi-os-is-here/>.
- (10) Alcatel-Lucent, *3.3. Initializing an IP Touch 40x8 EE phone In: IP Touch 4008/4018 Extended Edition - SIP Phone Installation Guide - 8AL90824AAAA ed02*, p.7-8, August 2010, available at: <https://www.cluster2.hostgator.co.in/files/writeable/uploads/hostgator136107/file/iptouchsipphoneinstallationguide-ed02.pdf>.
- (11) SECTIGO Store, *ECDSA vs RSA: Everything You Need to Know*, 9th june 2020, available at: <https://sectigostore.com/blog/ecdsa-vs-rsa-everything-you-need-to-know/>.
- (12) Agence nationale de la sécurité des systèmes d'information, *Logarithme discret dans les courbes elliptiques définies sur GF(p) In: Référentiel Général de Sécurité version 2.0 - Annexe B1*, p.19-20, 21st february 2014, available at: https://www.ssi.gouv.fr/uploads/2014/11/RGS_v-2-0_B1.pdf.

(13) Raspberry Pi Foundation, *Static IP address In: TCP/IP networking*, raspberrypi.org, 2021,
available at:
<https://www.raspberrypi.com/documentation/computers/configuration.html#static-ip-addresses>.

All references were consulted on 1st February 2022.

Attributions

Figure 5 - PBX Matra MC6500 serie: the original uploader was After310 at French Wikipedia, *PABX Matra série MC6500*, [CC BY-SA 3.0](#), via Wikimedia Commons, available at: https://commons.wikimedia.org/wiki/File:PABX_Matra6500.JPG.

Figure 17 - Establishing a connection between two clients: adapted from the original work of Feyd-Aran, *Etablissement d'une connexion par WebRTC*, [CC BY-SA 3.0](#), via Wikimedia Commons, available at: https://commons.wikimedia.org/wiki/File:Etablissement_d'une_connexion_par_WebRTC.svg

Appendixes

Appendix A1 - Installation of a virtual machine under Debian 10 Buster

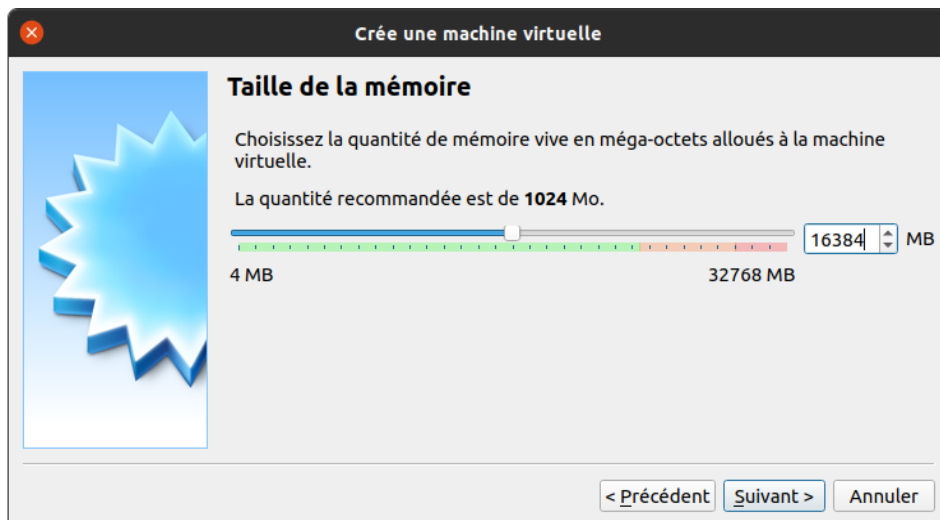
I - Preparation of the VM

Prerequisites: VirtualBox installed (<https://www.virtualbox.org/wiki/Downloads>) and an Internet connection.

1. Download Debian 10 Buster **AMD64 netinst image**:
<https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/>
2. Create a VM: **Launch VirtualBox** > **New** then click **Next**.



3. Allocate at least **8 GiB** of RAM and click **Next**.



4. Select **Create a virtual disk now** then click **Create**.
5. Choose the **VDI (VirtualBox Disk Image)** type and click **Next**.
6. Select **Dynamically allocated** and click **Next**.
7. Select a minimum size of **25 GiB** for the project and click **Create**.

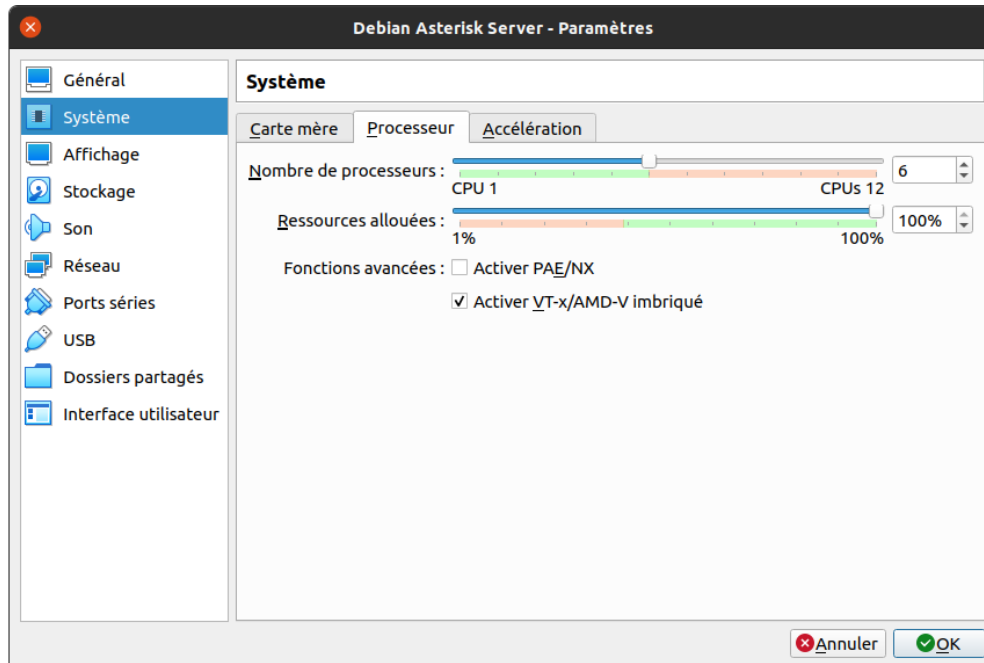


8. (Optional) In order to improve the performance of the VM it is interesting to use virtualisation. To do so, you must first enable Intel VT-x or AMD-V in the BIOS/UEFI.

For Ubuntu users only, you will need to perform an additional manipulation by opening a terminal and typing the following command:

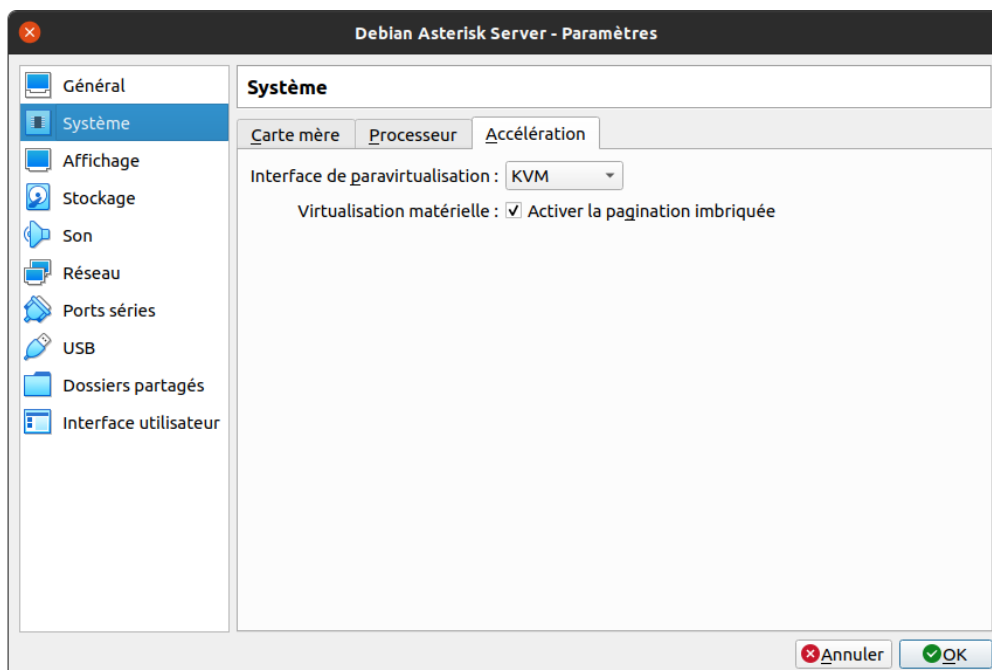
```
VBoxManage modifyvm "Debian Asterisk Server" --nested-hw-virt on
```

9. Launch VirtualBox, select the **Debian Asterisk Server** machine, click on **Configuration** > **System** > **Processor** tab. Set the half of the CPU available on your PC and check **Enable nested VT-x/AMD-V**.

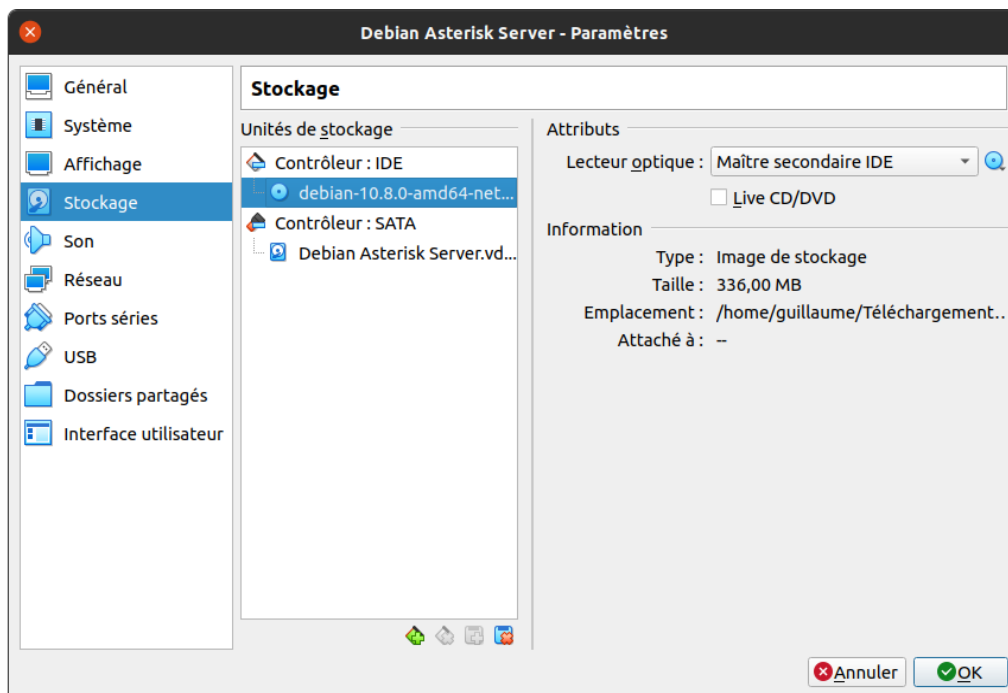


10. In the Acceleration tab, select the paravirtualisation interface:

- **KVM** if the host machine is a Linux system;
- **Hyper-V** if the host machine is a Windows system;
- **Minimal** if the host machine is a macOS system;

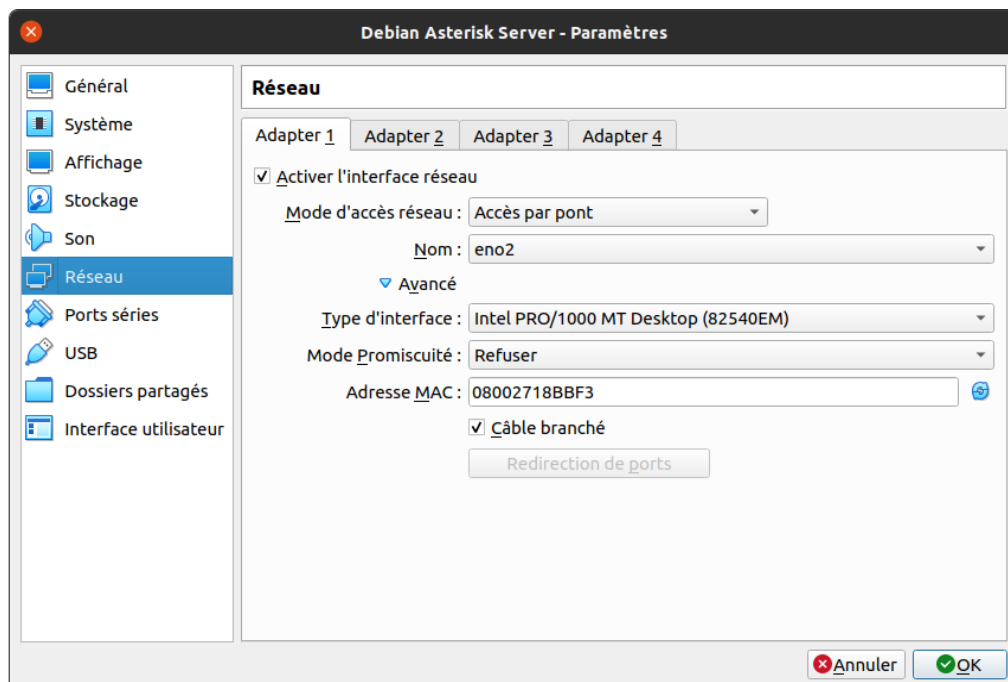


11. In **Storage**, select the disk image containing Debian 10 Netinst.



12. In **Network**, **Adapter 1** tab, select **Bridge Access**, and click **OK**.

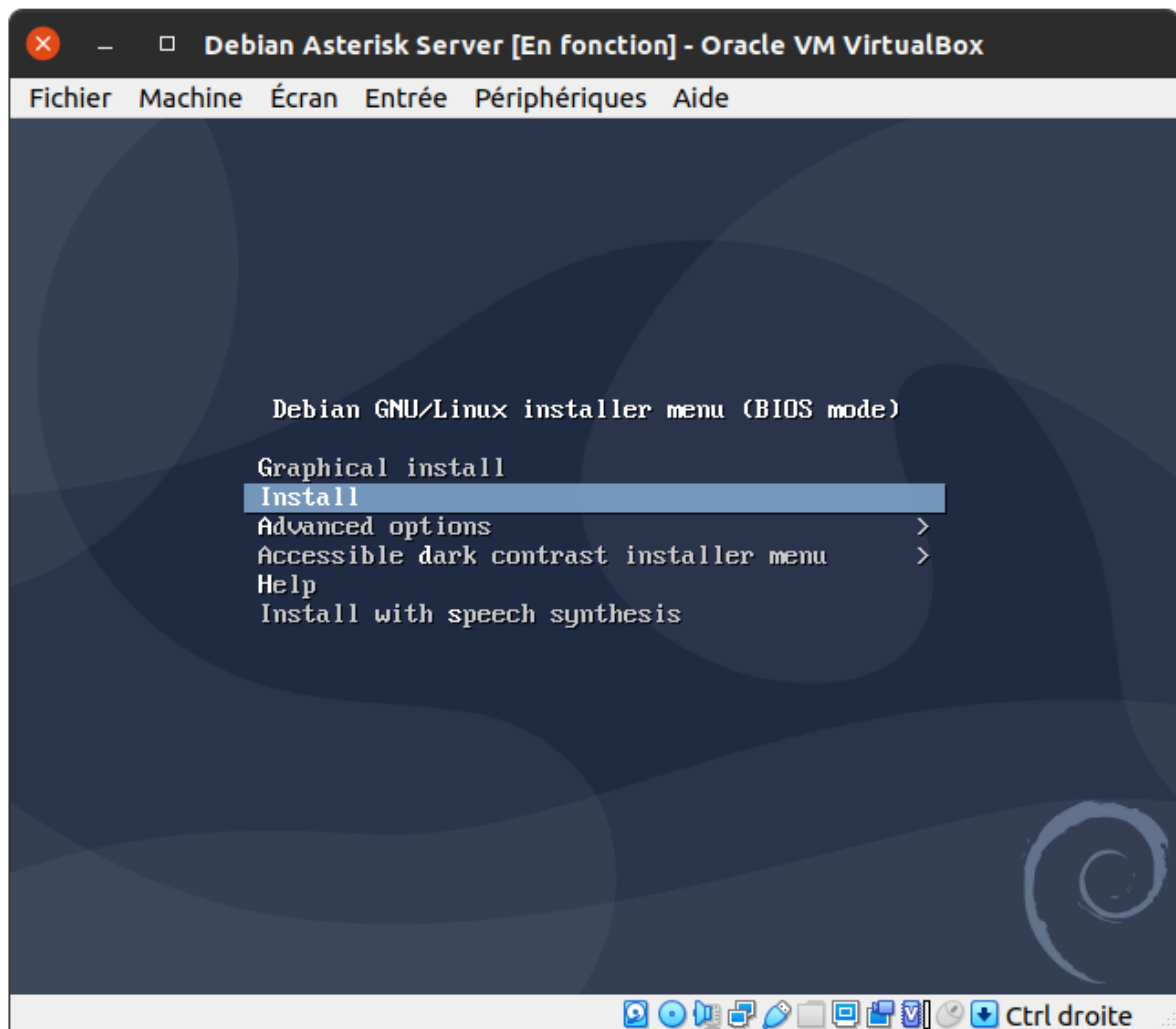
Note: This is an arbitrary choice, we made this choice to make it easier to access the platform from the local network. It is perfectly possible to use NAT mode and to do port forwarding, however it will be necessary to perform additional operations on Asterisk.



The VM configuration is finished, let's go to the Debian installation!

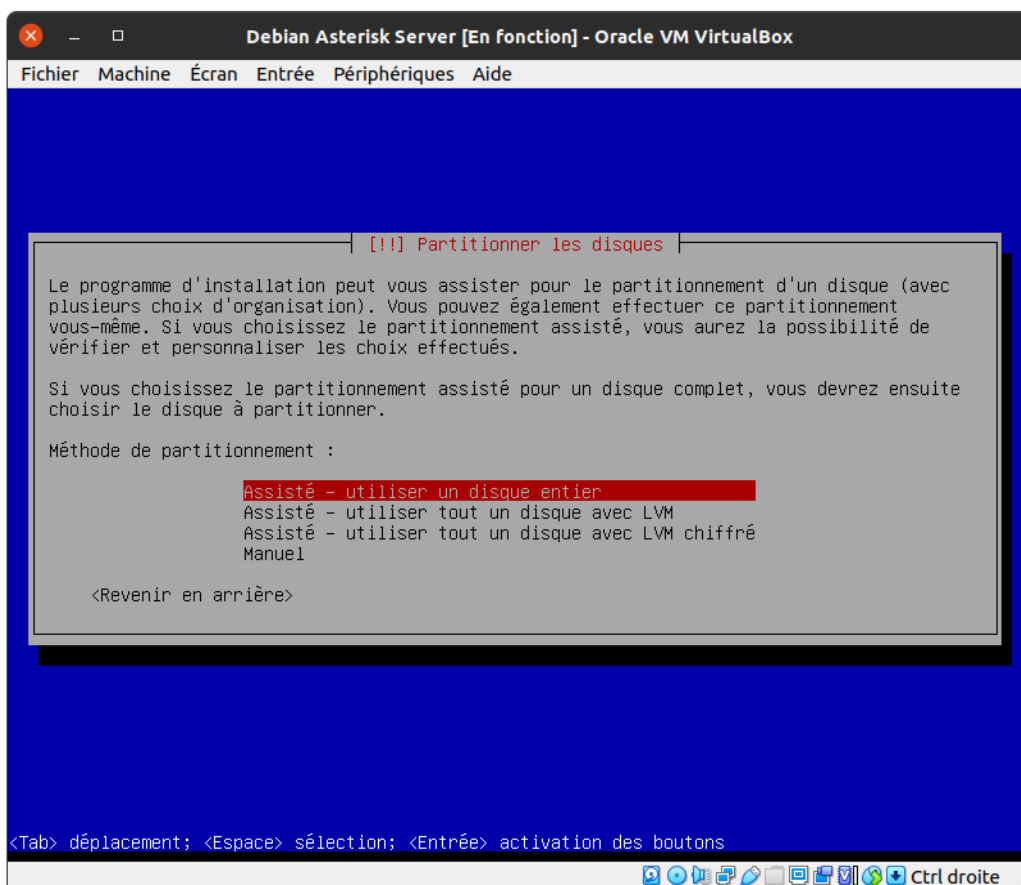
II - Installation and configuration of Debian 10

1. Launch the VM by clicking on **Start**, and choose the Debian Netinst ISO as boot disk.
2. Select **Install**.

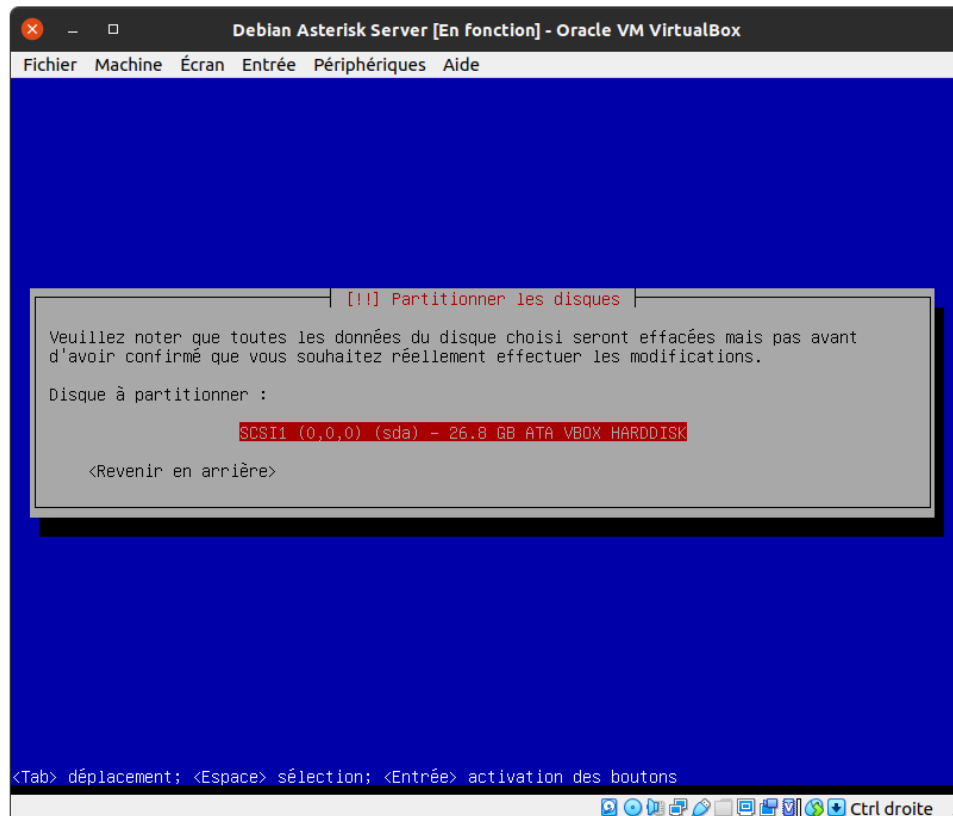


3. **[[!]] Select a language**
Select a language, then press **Enter**.
4. **[[!]] Choose your geographical location**
Select a country, in this tutorial it will be **France**.
5. **[[!]] Configure the keyboard layout**
Select the keyboard layout corresponding to your keyboard.

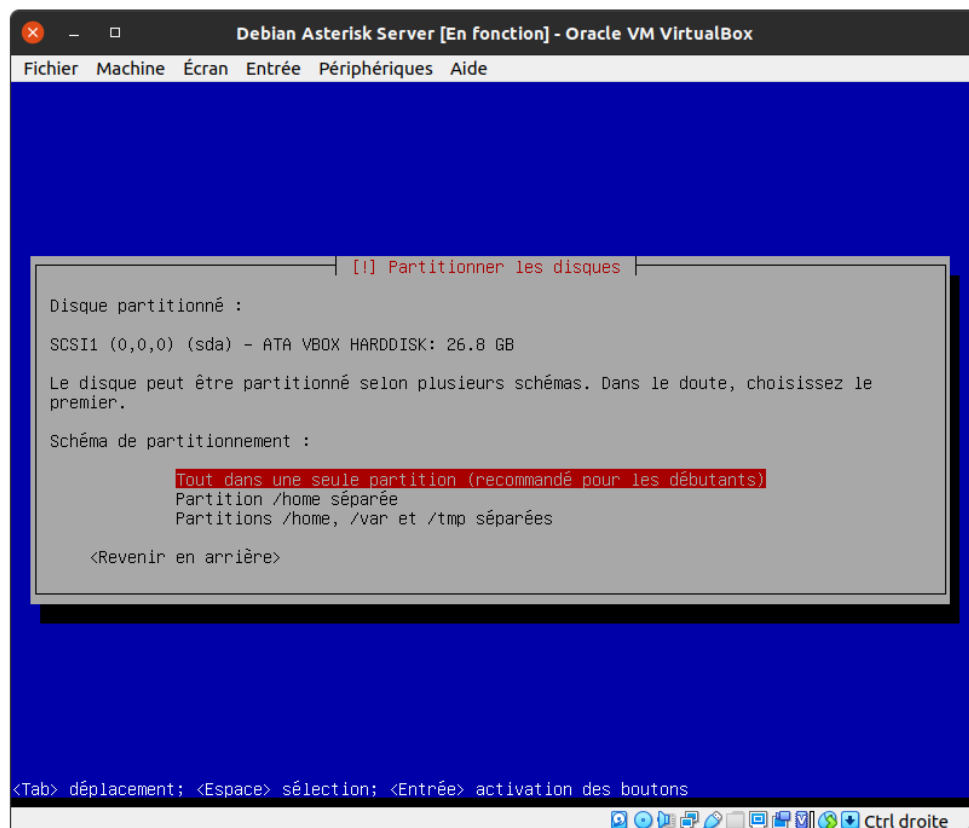
6. **[!!] Configuring the network - machine name**
Choose a name for the machine, here it will be: **asterisktz**, then select **<Continue>**.
7. **[!!] Configuring the network - domain**
Leave blank, then select **<Continue>**.
8. **[!!] Create users and choose passwords - root**
Choose a password for the superuser (root), here it will be **voiputc**.
Confirm the password and then **<Continuer>**.
9. **[!!] Create users and choose passwords - non-root user**
Choose a name and a login for the new non-root user, here it will be **asterisktz** as well as a password, in our case it will also be **voiputc**. Of course, in a production environment the root password and the password of the classic user must not be the same. Here the goal is educational.
10. **[!!] Partitioning disks**
Choose **Assisted - use an entire disk**.



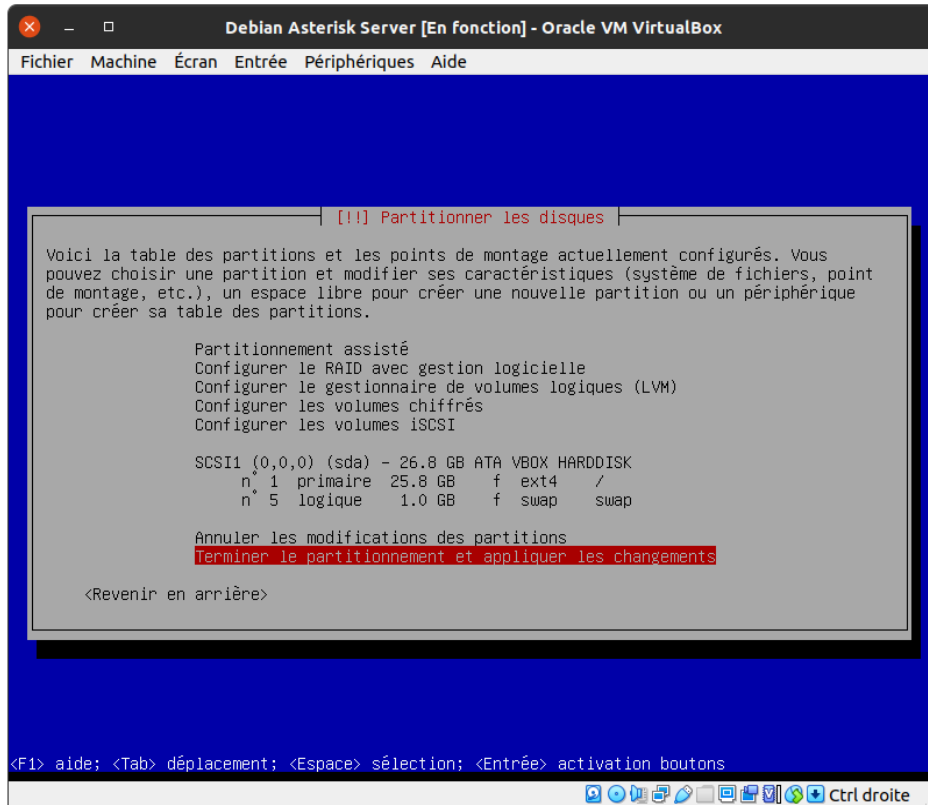
Select the only disk to be partitioned.



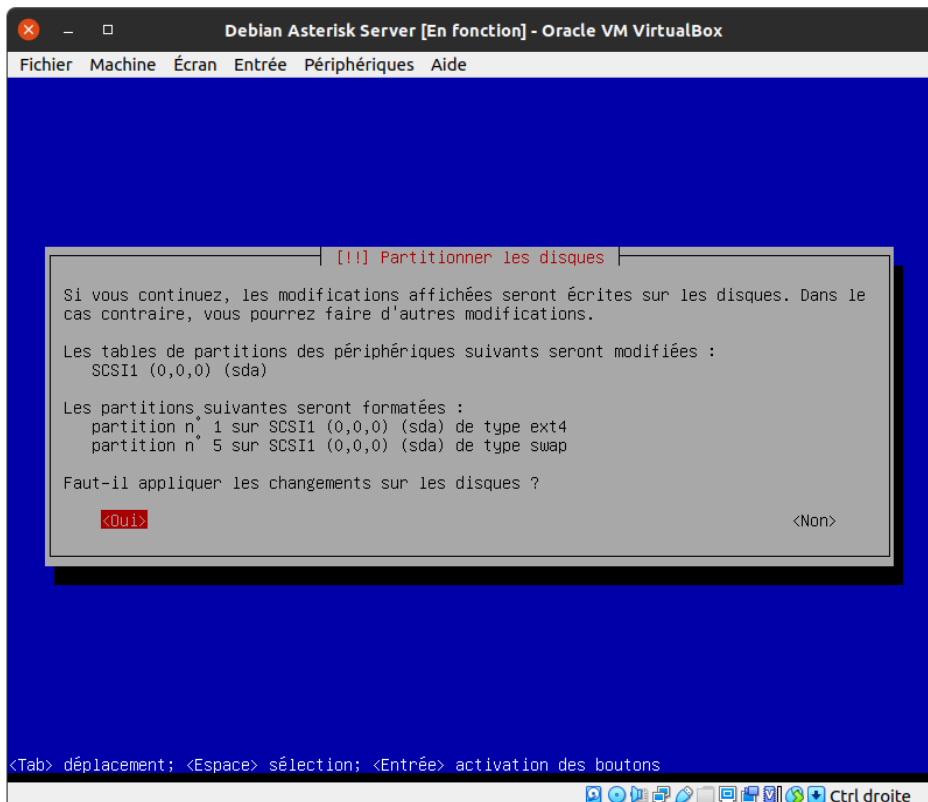
Choose the partitioning scheme: **Everything in one partition.**



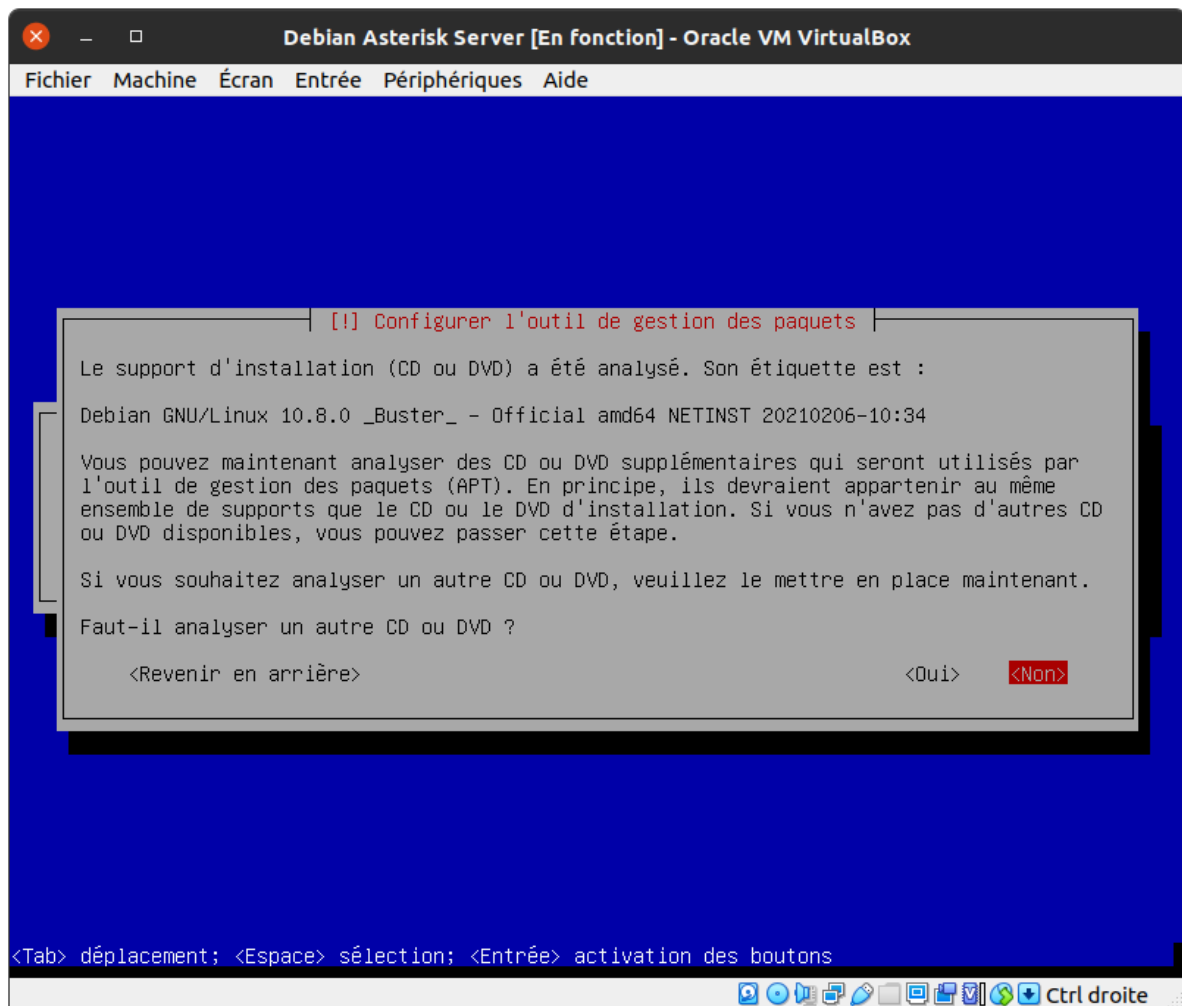
Then select **Finish partitioning and apply the changes**.



Confirm the changes by selecting **<Yes>**.

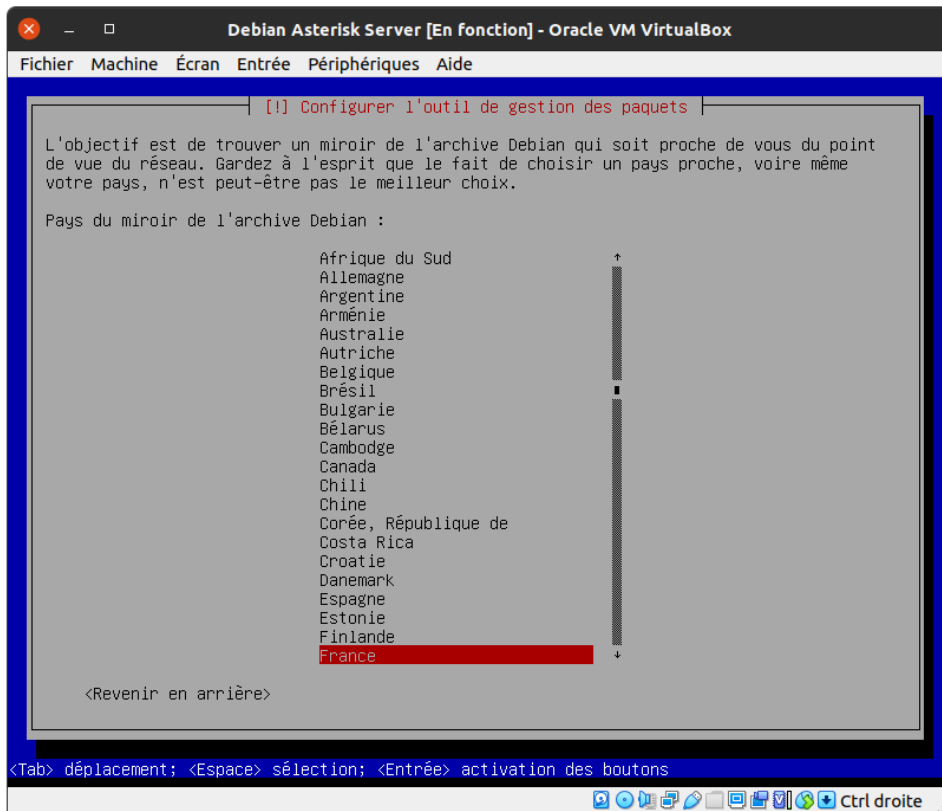


The system is installed. At the end of the installation, the installer proposes to scan another CD or DVD, you must choose **<No>**.

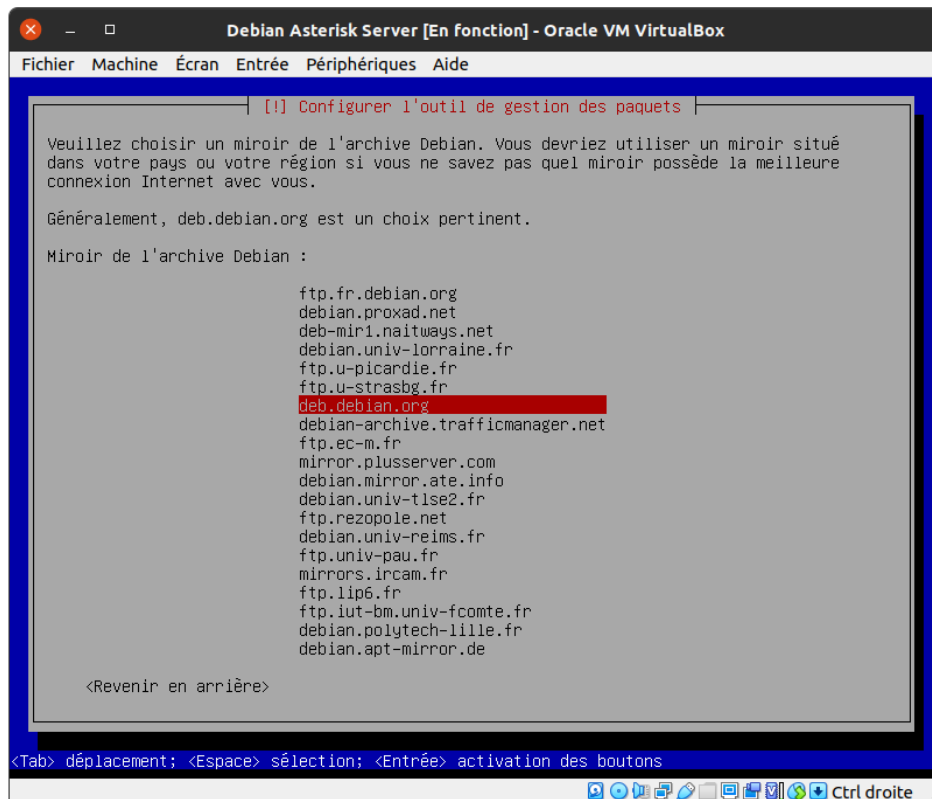


11. [!] Configuring the package management tool

Here it will be **France**...



...then the **deb.debian.org** mirror.

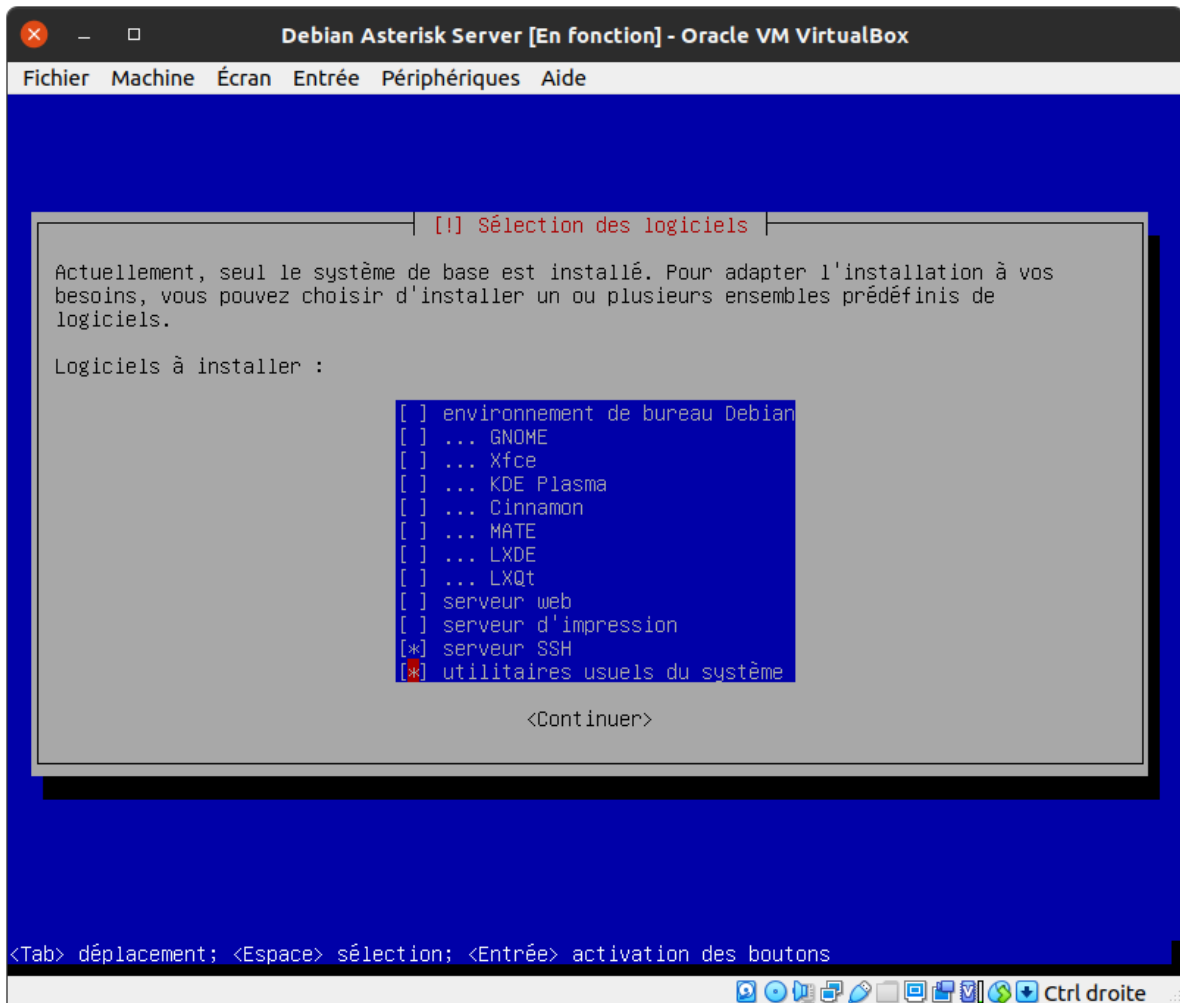


After choosing the mirror, a window appears to configure the HTTP proxy, configure this part if necessary, then select **<Continue>**.

12. [!!] Selection of software

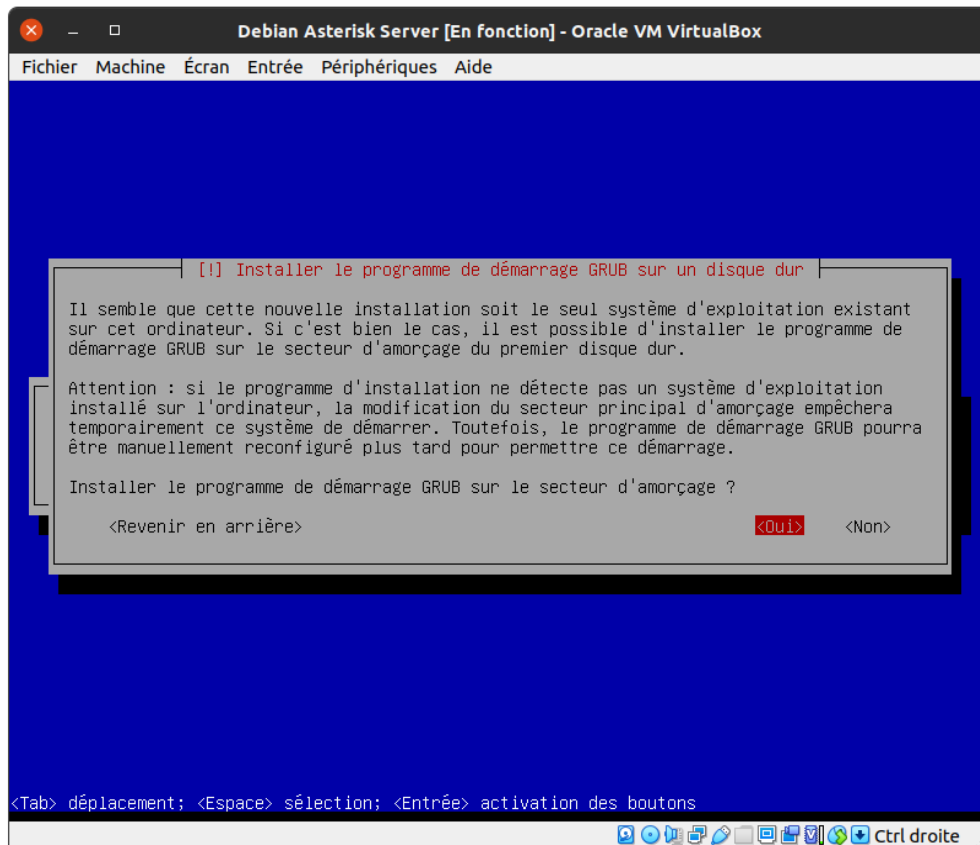
Select only the SSH server and the usual system utilities as below, then click **<Continue>**.

Note: the selection is made with the space bar.

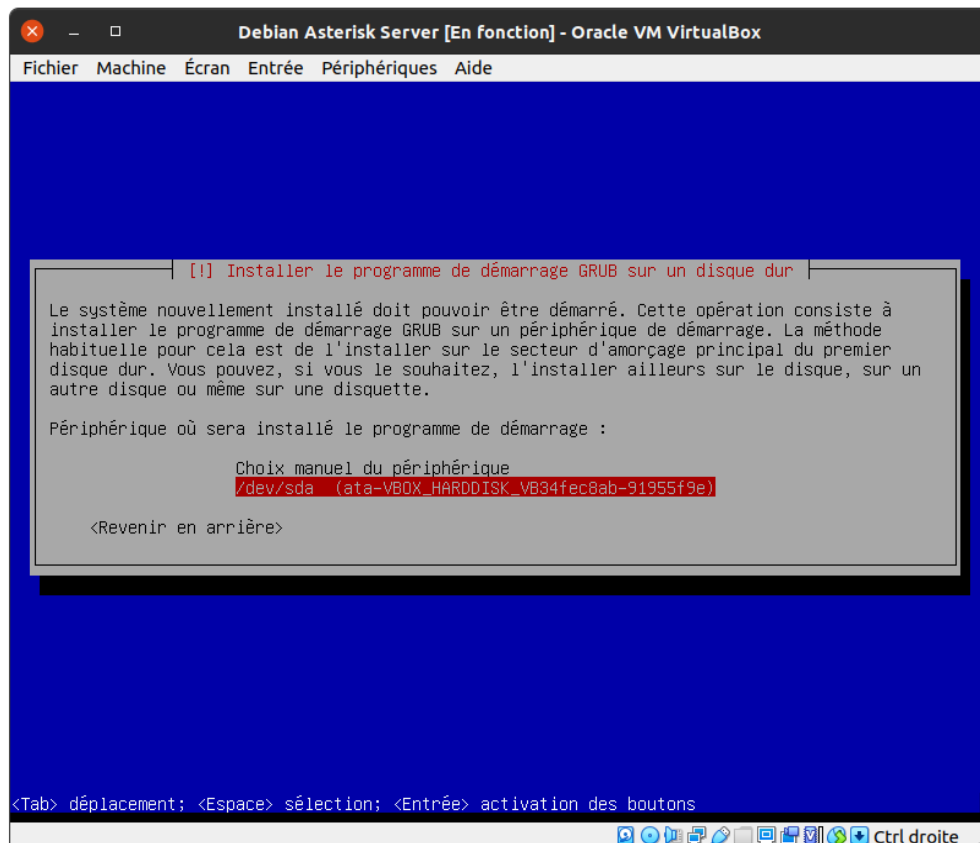


13. [!!] Installing the GRUB boot program on a hard disk

Select **<Yes>**.

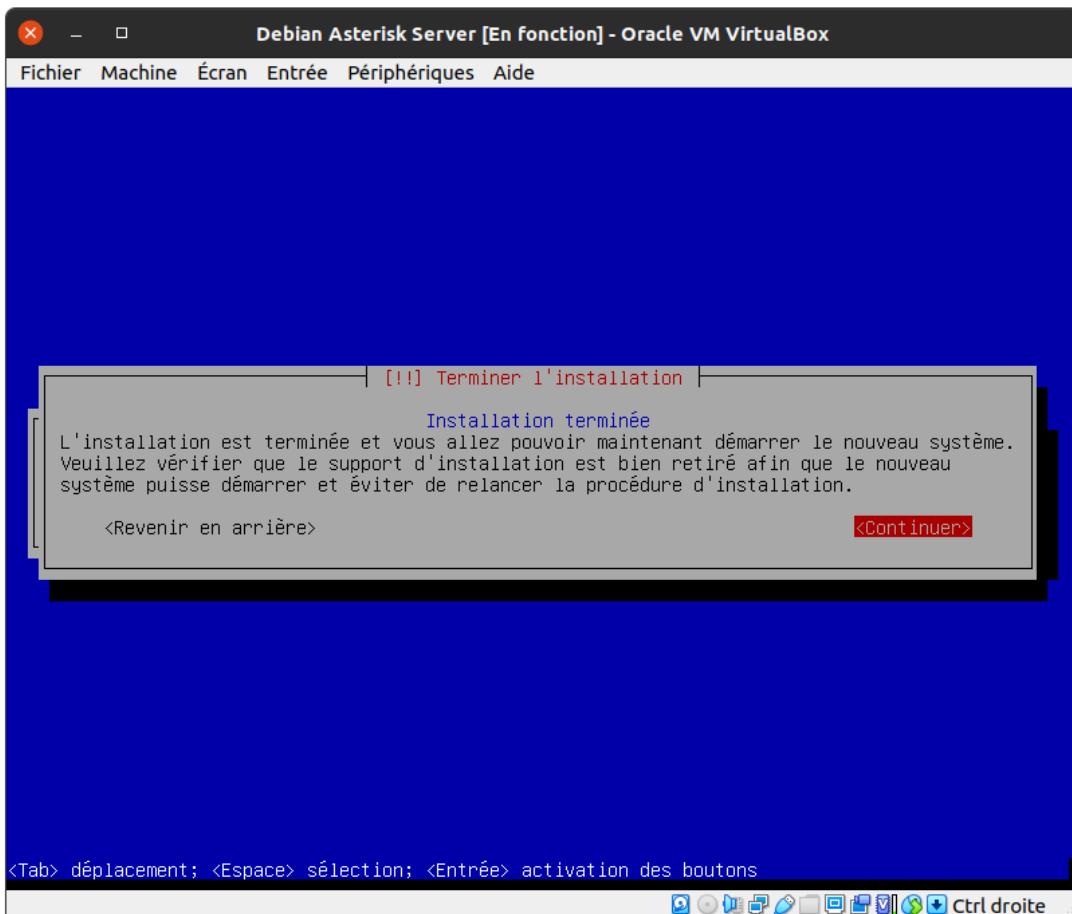


Then select the hard drive `/dev/sda`.



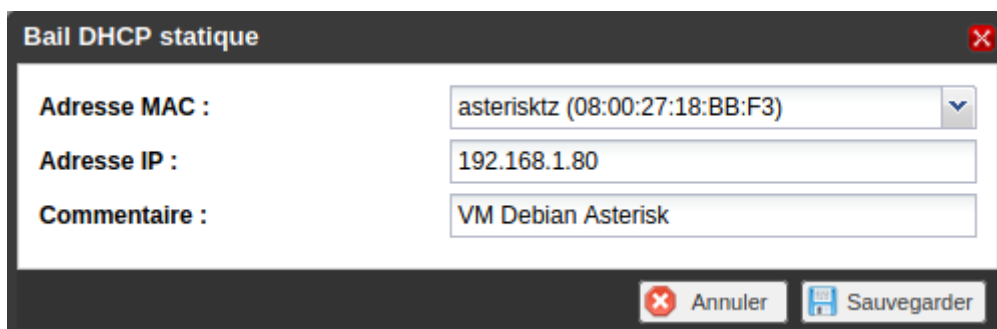
14. **[!!] Finish the installation**

Remove the installation media (VirtualBox usually does this automatically) then select **<Continue>**.



15. After the reboot, configure a static IP address from the router (here, it is a Freebox). This is possible because we have configured a bridge access from the VirtualBox interface. More information on the procedure for assigning a static DHCP lease on Freebox routers:

<https://wxfrantzconcept.wordpress.com/2016/10/18/assigner-une-adresse-ip-fixe-avec-sa-freebox/>.



16. Restart the virtual machine and connect via SSH.

```
# ssh login@vm_ip_address -p 22  
ssh asterisktz@192.168.1.80 -p 22
```

17. Install updates and sudo.

```
asterisktz@asterisktz:~$ Login as root
```

```
su
```

```
root@asterisktz:/home/asterisktz# Installing updates and sudo
```

```
apt update && apt upgrade -y && apt dist-upgrade -y && apt autoremove -y  
&& apt install sudo -y
```

```
root@asterisktz:/home/asterisktz# Configuring sudo
```

```
sudo visudo -f /etc/sudoers.d/asterisktz
```

```
/etc/sudoers.d/asterisktz Add the user asterisktz as a superuser and save  
via Ctrl + O
```

```
asterisktz    ALL=(ALL:ALL) ALL
```

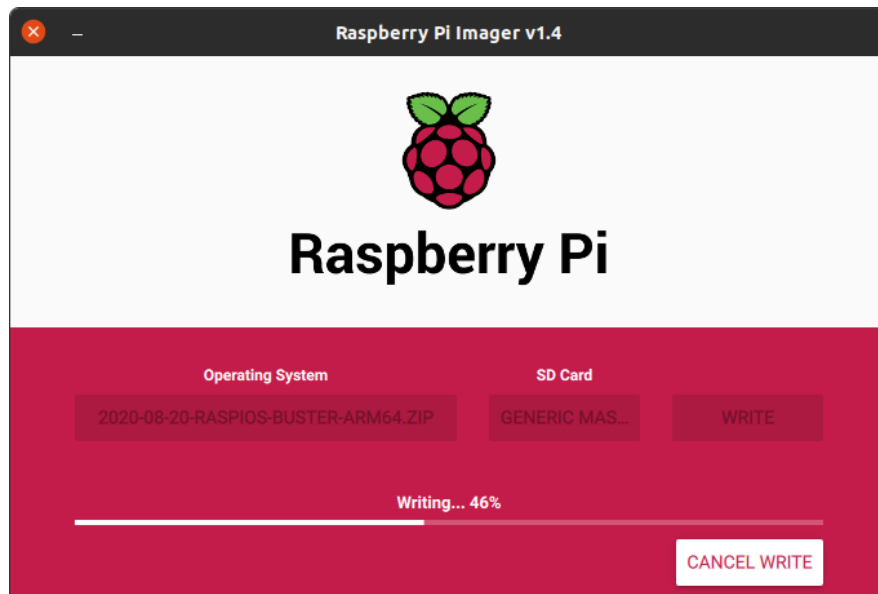
18. Restart the virtual machine and connect via SSH.

```
ssh asterisktz@192.168.1.80 -p 22
```

The Debian machine is ready, you can return to the [installation of the Asterisk server](#).

Appendix A2 - Installing and configuring Raspberry Pi OS Buster (64-bit) for Raspberry Pi 3B+

1. Download and flash¹⁶ the image `2020-08-20-raspbios-buster-arm64.zip` available here:
https://downloads.raspberrypi.org/raspbios_arm64/images/raspbios_arm64-2020-08-24/.



At the end of the process, the Raspberry Pi is functional, but it is worth enabling the SSH server and configuring a static IP on the Ethernet and/or Wi-Fi interfaces.

2. To set up SSH, simply create a file named `ssh` containing nothing at the root of the microSD card's `boot` drive.
Note: this step is optional.
3. To configure a static IP on both interfaces, simply go to the rootfs disk and add the following lines¹⁷ to the end of `/etc/dhcpd.conf` file **(13)**:
Note: This step is optional; you can also configure a static IP via the router (see [Appendix A1.15](#)).

¹⁶ It is recommended to use Raspberry Pi Imager to flash the microSD card:
<https://www.raspberrypi.org/software/>.

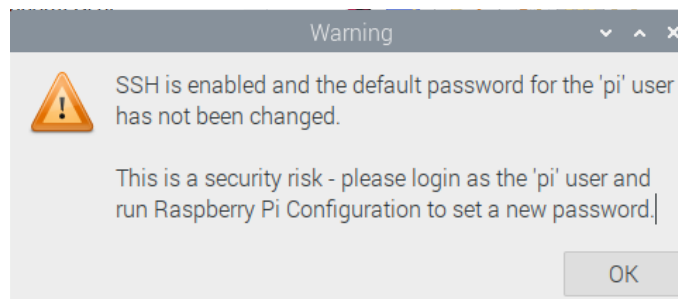
¹⁷ We arbitrarily chose these static IP addresses because they are outside the DHCP server allocation range of the router to which the Raspberry Pi is connected. You will need root rights to make this change. The first DNS server is that of the Freebox and the second of DNS.WATCH:
<https://dns.watch/>.

/rootfs/etc/dhcpd.conf

```
# Static IP address - Ethernet interface
interface eth0
static ip_address=192.168.1.82/24
static routers=192.168.1.254
static domain_name_servers=192.168.1.254 84.200.69.80

# Static IP address - Wi-Fi interface
interface wlan0
static ip_address=192.168.1.92/24
static routers=192.168.1.254
static domain_name_servers=192.168.1.254 84.200.69.80
```

4. Plug the microSD card into the Raspberry Pi, connect a keyboard, mouse, monitor and start it up.
5. Click **OK** on the SSH-related message. In practice, you would have to change the login and password. This is not the purpose of the project.



6. Configure the Raspberry Pi, click on **Next**.

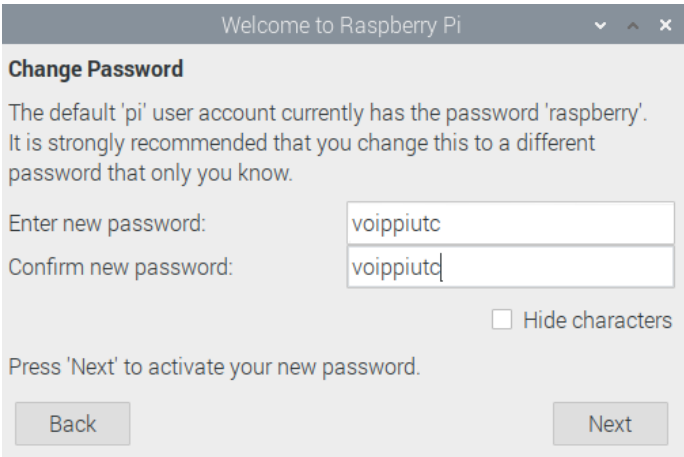


7. Set up languages.



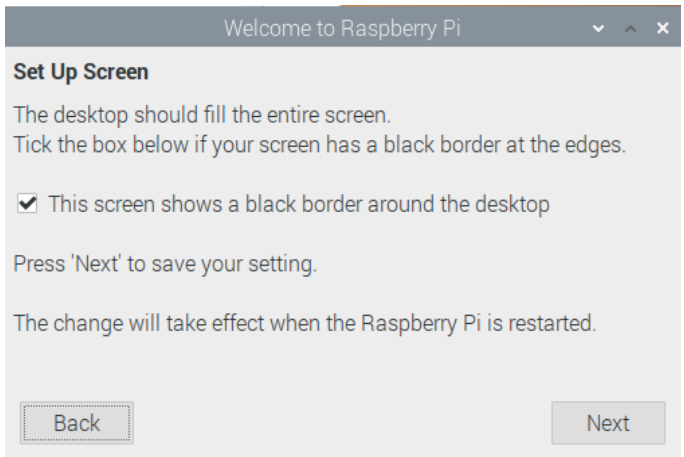
The screenshot shows a window titled "Welcome to Raspberry Pi" with a sub-header "Set Country". Below the sub-header is a paragraph: "Enter the details of your location. This is used to set the language, time zone, keyboard and other international settings." There are three dropdown menus: "Country:" set to "France", "Language:" set to "French", and "Timezone:" set to "Paris". Below these are two checkboxes: "Use English language" and "Use US keyboard", both of which are unchecked. At the bottom, there is a paragraph: "Press 'Next' when you have made your selection." and two buttons: "Back" and "Next".

8. Change the default password. Here it will be **voippiutc**.



The screenshot shows a window titled "Welcome to Raspberry Pi" with a sub-header "Change Password". Below the sub-header is a paragraph: "The default 'pi' user account currently has the password 'raspberrry'. It is strongly recommended that you change this to a different password that only you know." There are two text input fields: "Enter new password:" containing "voippiutc" and "Confirm new password:" containing "voippiutq". To the right of the second field is a checkbox labeled "Hide characters" which is unchecked. At the bottom, there is a paragraph: "Press 'Next' to activate your new password." and two buttons: "Back" and "Next".

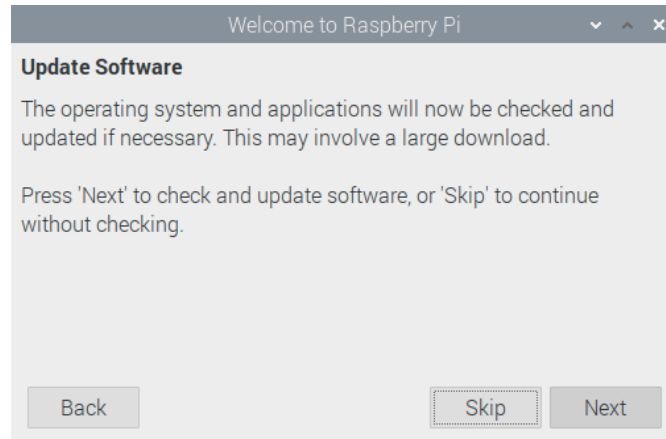
9. Set up the screen.



The screenshot shows a window titled "Welcome to Raspberry Pi" with a sub-header "Set Up Screen". Below the sub-header is a paragraph: "The desktop should fill the entire screen. Tick the box below if your screen has a black border at the edges." There is a checkbox labeled "This screen shows a black border around the desktop" which is checked. Below this is a paragraph: "Press 'Next' to save your setting." and another paragraph: "The change will take effect when the Raspberry Pi is restarted." At the bottom, there are two buttons: "Back" and "Next".

10. Select Wi-Fi network or not if the connection is in Ethernet.

11. Update the system and software by clicking **Next**. This may take some time.



12. Reboot the Raspberry Pi.



The system is operational, you can go to the [Linphone SIP client installation](#).