

UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS,
ELECTRÓNICA E INDUSTRIAL

CARRERA DE TELECOMUNICACIONES



PROYECTO FINAL

Octavo 'A'

INFORME

Tema:

ASTERISK

Integrante:

Josue Balseca Castro
Jean Carlos León

Fecha de Envío:

lunes 17 de junio

Fecha de Entrega:

lunes 24 de junio

Docente: Ing. Santiago Manzano.
marzo 2024 - agosto 2024

AMBATO-ECUADOR

2024

Índice

1. TEMA	4
2. OBJETIVOS	4
2.1. Objetivo General	4
2.2. Objetivos Específicos	4
3. RESUMEN	4
4. ABSTRACT	4
5. INTRODUCCIÓN	4
6. FUNDAMENTACIÓN TEÓRICA	4
6.1. Asterisk	4
6.2. Softphones: Concepto y Funcionalidad	5
6.2.1. Protocolos y Tecnologías Subyacentes	5
6.2.2. Zoiper	5
6.2.3. MicroSIP	5
6.3. Servidor Flask	6
7. EQUIPOS Y MATERIALES	6
7.1. Hardware	6
7.2. Software	6
8. DESARROLLO	6
8.1. Instalacion Asterisk	6
8.1.1. Actualice el sistema e instale las dependencias requeridas.	6
8.1.2. SELinux	6
8.1.3. Configuracion de dependencias	7
8.1.4. Make	8
8.1.5. inicialización	9
8.2. Comunicación Entre Centralitas	10
8.2.1. Configuración Iax.conf	10
8.2.2. Configuracion archivo sip.conf	10
8.2.3. Configuración archivo extensions.conf	11
8.2.4. Verificación de la comunicación entre centralitas	12
8.3. Comunicación Teléfono Ip	14
8.3.1. Configuracion Asterisk	14
8.3.2. Configuración Teléfono	15
8.3.3. Verificación con MicroSIP	17
8.4. Control De Dispositivo Físico ON/OFF con un foco utilizando Asterisk	19
8.4.1. Configuración de la ESP8266	19
8.4.2. Configuración en Asterisk	20
8.4.3. Verificación de encendido y apagado del foco	21
8.5. Consultar El Ultimo Dato	23
8.5.1. Configuración de Asterisk	23
8.5.2. Verificación de Resultados	24
8.6. Encuesta De Satisfacción	27
8.6.1. Script de php creado	27
8.6.2. Configuración de Asterisk	27
8.6.3. Verificación de Resultados	28
8.7. Sistema de Alarmas para Niveles De Temperatura, Humedad Y Luminosidad basado en Llamadas de alerta en Asterisk para prevención de incendios	32
8.7.1. Configuración de la ESP8266	32
8.7.2. Configuración en Asterisk	34
8.7.3. Configuración del script en php para las llamadas	35
8.7.4. Configuración servidor Flask	36
8.7.5. Verificación de Resultados	38

9. RECOMENDACIONES	44
10. CONCLUSIONES	44
11. BIBLIOGRAFÍA	44
12. ANEXOS	44

Índice de figuras

1. SELinux	7
2. status SELinux	7
3. inicio asterisk	8
4. menu select	8
5. make install	9
6. pantalla de inicio asterisk	9
7. configuracion archivo aix.conf	10
8. configuración archivo pjsip.conf	11
9. configuración archivo extensions.conf	12
10. Server A	12
11. Server B	13
12. configuracion archivo extensions.conf	14
13. configuracion archivo pjsip.conf	14
14. interfaz telefono Ip Fanvil X5s	15
15. interfaz configuracion lineas SIP	16
16. lineas SIP 1 y 2 escogidas para configurar	16
17. linea SIP 1 de Josue	17
18. linea SIP 2 de Jean	17
19. cuenta en MicroSIP	17
20. llamada realizada desde el teléfono a la Pc	18
21. Llamadas reallizadas a la extension 800	21
22. Pruebas realizadas con la ESP8266	22
23. Petición recibida en el servidor HTTP, /encender	22
24. Petición recibida en el servidor HTTP, /apagar	22
25. Entrada de la llamada en la aplicación móvil Zoiper	24
26. Entrada de la llamada en la aplicación de microSIP del ordenador	25
27. Mensajes por consola al seleccionar la humedad	25
28. Mensajes por consola al seleccionar la temperatura	25
29. Mensajes por consola al seleccionar la luz	26
30. Mensajes por consola al seleccionar la voltaje	26
31. Interfaz para muestra de la tabla de los últimos tres datos de la tabla sensor	26
32. Entrada de la llamada en la aplicación móvil Zoiper	28
33. Entrada de la llamada en la aplicación de microSIP del ordenador	29
34. Mensajes por consola al seleccionar la opción 1	29
35. Mensajes por consola al seleccionar la opción 2	30
36. Mensajes por consola al seleccionar la opción 2	30
37. Interfaz para muestra de la tabla de los datos en la tabla votación	31
38. Interfaz para conteo de datos de cada opción y un conteo total	31
41. Petición receptada de la ruta /check en el servidor Flask	39
39. Verificación a la red Wi-Fi al monitor serial	39
40. Ingreso de los datos, y verificación del envío de datos de la ESP8266 a los servidores flask y HTTP.	39
42. Peticiones receptoradas de la ruta /datos en el servidor Flask	39
44. Datos vistos en el monitor serial y el envío de la alerta de niveles bajos y ejecutándose la alarma	40
43. Datos vistos en el monitor serial y el envío de la alerta de niveles altos y ejecutándose la alarma	40
45. Visualización en el CLI de asterisk las alarmas realizadas para alerta de nivel bajo	40
46. Visualización en el CLI de asterisk las alarmas realizadas para alerta de nivel alto	40

47.	Entrada de alerta en la aplicación móvil Zoiper	41
48.	Entrada de alerta en la aplicación de MicroSIP	41
49.	Niveles muy bajo de luminosidad	42
50.	Niveles medio de luminosidad	42
51.	Nivel alto de luminosidad	42
52.	Dashboard de los sensores en la interfaz de la pagina web	43
53.	Mensaje de confirmación de alerta realizada visualizado en la interfaz de la pagina web . .	43

1. TEMA

Sistema de comunicaciones utilizando Asterisk para diversas aplicaciones de telecomunicaciones.

2. OBJETIVOS

2.1. Objetivo General

- Desarrollar un sistema de comunicaciones avanzadas basado en Asterisk para el uso en diversas aplicaciones en telecomunicaciones

2.2. Objetivos Específicos

- Configurar comunicaciones entre centralitas usando Asterisk.
- Implementar configuraciones necesarias para realizar llamadas a un teléfono IP.
- Controlar un dispositivo físico mediante comandos Asterisk.
- Consultar y recuperar el último dato almacenado en una base de datos desde Asterisk.
- Desarrollar un formulario para la realización de encuestas de satisfacción de clases.
- Integrar un sistema de alertas mediante llamadas

3. RESUMEN

El presente proyecto aborda la implementación de un sistema basado en Asterisk, con diversas funcionalidades que incluyen la interconexión de centralitas, llamadas VoIP, control de dispositivos, consultas en bases de datos, encuestas de satisfacción y alertas. Para la configurar de todos estos apartados se configura los archivos pjsip.conf y extensions.conf generando usuario y configuraciones de marcado de igual manera con la ayuda de scripts en *.php para la comunicación con la base de datos y los valores dados por el Esp8266 con Mqtt.

4. ABSTRACT

This advanced communications final project involves implementing an Asterisk-based system with various functionalities, including PBX interconnection, VoIP calls, device control, database queries, satisfaction surveys, and alerts.

5. INTRODUCCIÓN

En el ámbito de las telecomunicaciones, Asterisk se destaca como una herramienta poderosa y flexible para la creación de sistemas de comunicaciones. Este proyecto se centra en la utilización de Asterisk para diversas aplicaciones prácticas que reflejan escenarios comunes en la industria. La interconexión de centralitas, la posibilidad de realizar llamadas VoIP, el control de dispositivos mediante comandos, la consulta en bases de datos, la realización de encuestas y la implementación de alertas son actividades que demuestran el alcance y la versatilidad de Asterisk en un entorno de comunicaciones avanzadas.

6. FUNDAMENTACIÓN TEÓRICA

6.1. Asterisk

Asterisk es una plataforma de código abierto para la creación de sistemas de comunicación. Es utilizada ampliamente para implementar centralitas telefónicas IP (PBX), gateways VoIP, servidores de conferencias y otros sistemas de telecomunicaciones. Entre sus características más destacadas se encuentran la capacidad de manejar múltiples protocolos de señalización, como SIP y IAX, y la flexibilidad para integrarse con diversos servicios y aplicaciones mediante scripts y APIs.[1]

Comunicaciones entre centralitas: La interconexión de centralitas permite la extensión de redes telefónicas a través de diferentes ubicaciones geográficas.[1]

Configuraciones de llamadas VoIP: Habilitar llamadas desde una PC a un teléfono IP implica la correcta configuración de softphones y teléfonos IP en la red.[1]

Control de dispositivos: Utilizando Asterisk, se pueden enviar comandos a dispositivos conectados para realizar acciones como prender un foco.[1]

Consultas en bases de datos: Integrar bases de datos con Asterisk permite la recuperación y gestión de información en tiempo real.[1]

Encuestas de satisfacción: Formularios y encuestas son herramientas clave para medir la calidad del servicio educativo.[1]

6.2. Softphones: Concepto y Funcionalidad

Un softphone es una aplicación de software que permite realizar llamadas telefónicas a través de Internet usando un ordenador, smartphone u otro dispositivo habilitado. A diferencia de los teléfonos físicos tradicionales, los softphones utilizan protocolos de Voz sobre IP (VoIP) para la transmisión de voz, permitiendo así la comunicación sin necesidad de hardware adicional más allá del dispositivo en el que están instalados.[4]

6.2.1. Protocolos y Tecnologías Subyacentes

Los softphones funcionan sobre protocolos estándar de VoIP, como el Protocolo de Iniciación de Sesión (SIP) y el Protocolo de Transporte en Tiempo Real (RTP), que gestionan la señalización y la transmisión de datos de voz, respectivamente.[4]

SIP (Session Initiation Protocol): Es un protocolo de señalización utilizado para establecer, modificar y finalizar sesiones multimedia como llamadas de voz y video. SIP es un estándar abierto ampliamente utilizado por la mayoría de las soluciones VoIP, incluyendo softphones.[4]

RTP (Real-time Transport Protocol): Es utilizado para la entrega de datos en tiempo real, como audio y video, en aplicaciones de transmisión en tiempo real. RTP se encarga de la transmisión y el empaquetamiento de los datos de voz en la red.[4]

6.2.2. Zoiper

Zoiper es un softphone multiplataforma compatible con diversos sistemas operativos como Windows, macOS, Linux, Android y iOS. Entre sus características destacadas se incluyen:[2]

Compatibilidad con múltiples protocolos: Zoiper soporta SIP e IAX (Inter-Asterisk Exchange Protocol), lo que lo hace versátil para su uso con diferentes servidores y sistemas VoIP.[2]

Interfaz de usuario intuitiva: Zoiper ofrece una interfaz amigable y fácil de usar, lo que facilita su configuración y operación.[2]

Seguridad: Soporta cifrado de voz mediante SRTP y ZRTP, y cifrado de señalización con TLS para proteger las comunicaciones.[2]

Integración con directorios: Zoiper permite la integración con directorios de contactos y la sincronización con aplicaciones externas, facilitando la gestión de contactos y llamadas.[2]

6.2.3. MicroSIP

MicroSIP es otro softphone popular que se enfoca en la simplicidad y el bajo consumo de recursos. Es una aplicación ligera para Windows, conocida por su facilidad de uso y configuración rápida. Algunas de sus características clave son:[5]

Ligereza y eficiencia: MicroSIP es una aplicación de pequeño tamaño y baja demanda de recursos, ideal para sistemas con limitaciones de hardware.[5]

Alta calidad de audio: Ofrece soporte para códecs de alta calidad como Opus, G.711, G.722, y otros, asegurando una excelente calidad de voz.[5]

Compatibilidad con SIP: Totalmente compatible con el protocolo SIP, permite la integración con la mayoría de los servidores y servicios VoIP.[5]

Portabilidad: MicroSIP puede ejecutarse directamente desde una memoria USB sin necesidad de instalación, lo que facilita su uso en diferentes dispositivos.[5]

6.3. Servidor Flask

Flask Python adopta un enfoque diferente. El framework web lanzado por el desarrollador austriaco Armin Ronacher en 2010, es más minimalista. Flask solo incluye el motor de plantillas Jinja y una biblioteca llamada “tool”. Sin embargo, ofrece la posibilidad de integrar funciones de terceros. El framework Flask está bajo una licencia BSD. Es gratuito y de código abierto. Como contrapunto a Django y otros frameworks. [3]

7. EQUIPOS Y MATERIALES

7.1. Hardware

- Laptop Servidor para Asterisk)
- leds
- telefono ip
- Protoboard
- foco
- rele
- Esp8266
- cables ethernet
- Access Point

7.2. Software

- Asterisk
- Softphone (Zoiper, Microsip)
- Mysql
- Phpmyadmin
- Python

8. DESARROLLO

8.1. Instalacion Asterisk

En primer lugar, asegurese de ejecutar todos los comandos en modo super usuario

8.1.1. Actualice el sistema e instale las dependencias requeridas.

```
apt update
```

```
apt install nano wget tar ncurses-devel
```

8.1.2. SELinux

deshabilitar SELinux, ya que si no lo hacemos tendremos problemas más tarde, por ejemplo, con la reproducción de grabaciones.

Primero use sestatus. Le mostrará el estado actual de SELinux:

```
[root@centos ~]# sestatus
SELinux status:                                enabled
SELinuxfs mount:                             /sys/fs/selinux
SELinux root directory:                      /etc/selinux
Loaded policy name:                           targeted
Current mode:                                 enforcing
Mode from config file:                      enforcing
Policy MLS status:                           enabled
Policy deny_unknown status:                 allowed
Memory protection checking:                actual (secure)
Max kernel policy version:                  33
```

Figura 1: SELinux

con el comando: sed -i s/SELINUX=enforcing/SELINUX=disabled/g /etc/selinux/config desactivamos SELinux y posteriormente ingresar un setenforce 0 para entrar en modo permissivo

8.1.3. Configuracion de dependencias

ingresamos los siguientes comandos

```
cd /usr/src
wget http://downloads.asterisk.org/pub/telephony/asterisk/asterisk-18-current.tar.gz
tar zxvf asterisk-18-current.tar.gz
rm -rf asterisk-18-current.tar.gz
cd asterisk-18*/
```

Dado que chan-pjsip requiere algunas bibliotecas adicionales que quizás deseé agregar –with-jansson-bundled –with-pjproject-bundled a configure comando.

* A partir de Asterisk 15.0.0, está habilitado de forma predeterminada, pero se puede deshabilitar con el –without-pjproject-bundled opción a ./configure.

./configure –libdir=/usr/lib64

Después de que termine, verá la pantalla “all-is-done” con el logotipo de Asterisk.

```
[root@asteriskl6 ~]# sestatus
SELinux status:                                disabled
```

Figura 2: status SELinux

8.1.4. Make

Entonces podrías make y esto compilaría su software. Pero es mejor poder seleccionar algunas opciones adicionales, funciones, aplicaciones, códigos, etc.....

make menuselect

Figura 3: inicio asterisk

Durante el proceso, puede verificar, cambiar, seleccionar y anular la selección de todas las opciones, por ejemplo, usar ODBC en lugar de almacenamiento FILE para el correo de voz. Notará que no puede seleccionar todos los módulos. Pero también muestra qué dependencias se requieren.

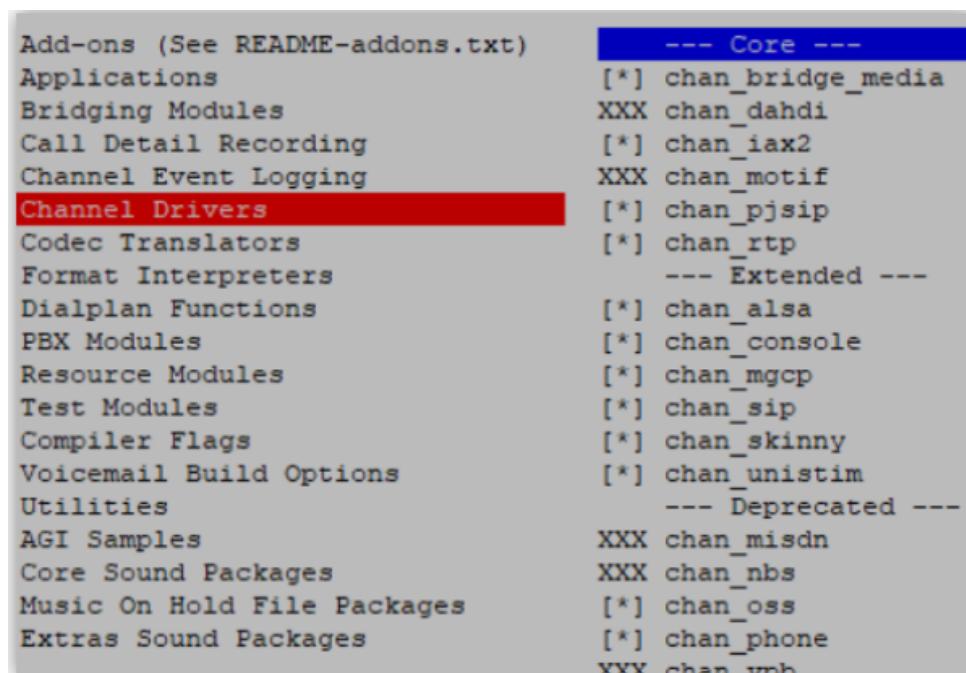


Figura 4: menu select

Es hora de instalar todo. Este y el siguiente proceso pueden tomar unos minutos.

se hace un make install y un make samples

Luego muévalos a una nueva carpeta (por ejemplo. /etc/asterisk/samples/) y crear configuración básica con make basic-pbx.

```
mkdir /etc/asterisk/samples
```

```
mv /etc/asterisk/*.* /etc/asterisk/samples/
```

```
make basic-pbx
```

Aunque Asterisk ya está listo ahora debe hacerlo make config para crear archivos de inicio y facilitar el uso de comandos básicos.

```
make config
```

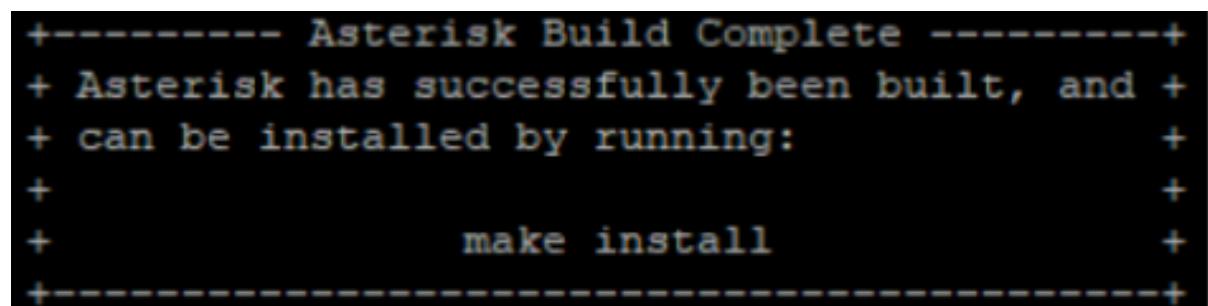
8.1.5. inicialización

ingresar los siguientes comandos para ver el estado de asterisk

```
systemctl start asterisk
```

```
systemctl status asterisk
```

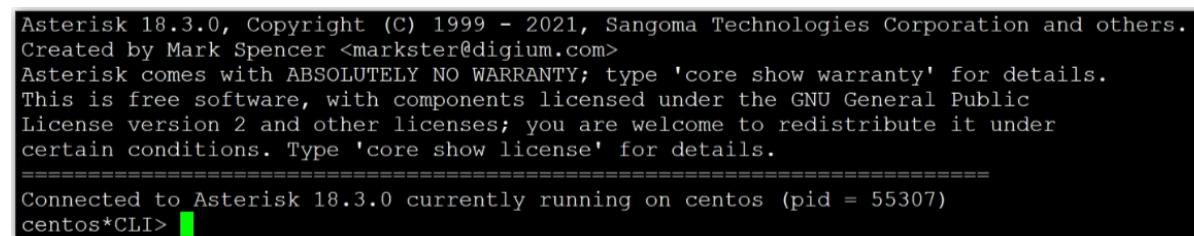
También puede ingresar /etc/asterisk directorio y ver todos los archivos básicos y sample/ carpeta



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

Figura 5: make install

despues ingresa asterisk -rvvvvv



```
Asterisk 18.3.0, Copyright (C) 1999 - 2021, Sangoma Technologies Corporation and others.
Created by Mark Spencer <markster@digium.com>
Asterisk comes with ABSOLUTELY NO WARRANTY; type 'core show warranty' for details.
This is free software, with components licensed under the GNU General Public
License version 2 and other licenses; you are welcome to redistribute it under
certain conditions. Type 'core show license' for details.
=====
Connected to Asterisk 18.3.0 currently running on centos (pid = 55307)
centos*CLI> [REDACTED]
```

Figura 6: pantalla de inicio asterisk

8.2. Comunicación Entre Centralitas

La siguiente configuración permitirá la comunicación entre dos centralitas del server A y el server B

8.2.1. Configuración Iax.conf

Ubicados en el server B (Josue), se abre el archivo iax.conf, para editarlo de la siguiente manera:

```
[serverA]
type=friend
username=servidorB
auth=plaintext
host=192.168.7.2
context=entrantes
trunk=yes
requirecalltoken=no
```

Figura 7: configuracion archivo aix.conf

Donde:

type: corresponde al tipo de usuario, en este caso friend lo que le permite hacer y recibir llamadas.

username: nombre del usuario.

auth: Tipo de autenticación realizada, en la configuración realizada la contraseña se guarda en texto plano, existen otras opciones como MD5 y RSA en las que el nivel de seguridad incrementa progresivamente.

host: dirección IP del servidor A.

trunk: yes, permite activar el modo troncal, cuando varias llamadas van al mismo destino en lugar de mandar una trama por cliente, se manda una trama para los 3 con el audio de cada uno, esto permite un mejor aprovechamiento de los recursos de la red.

Requirecalltoquen: no, disminuye la seguridad de IAX , puesto que permite el acceso a invitados desactivando la validación para llamadas no autenticadas.

8.2.2. Configuracion archivo sip.conf

Se configura el archivo sip.conf, donde se crean los usuarios propios del servidor B, los usuarios serán 80011 y 80012, tal como se muestra a continuación:

```

[80011]
type=friend
username=80011
secret=80011
callerid=80011
host=dynamic
directmedia=no
insecure=invite
context=entrantes

[80012]
type=friend
username=80012
secret=80012
callerid=80012
host=dynamic
directmedia=no
insecure=invite
context=entrantes

```

Figura 8: configuración archivo pjsip.conf

Los parámetros de cada usuarios significa lo siguiente:

type: tipo de usuario.(idéntico al especificado para el servidor B debido a que se podrán recibir y hacer llamadas desde cada teléfono).

username: nombre del usuario.

callerid: número telefónico de identificación.

secret: contraseña.

host: dirección IP, en este caso dynamic lo que le permite conectarse con cualquier dirección IP.

context: contexto por defecto donde entraran las llamadas.

8.2.3. Configuración archivo extensions.conf

Luego se edita el archivo extensions.conf, que tendra la información de los usuarios creados anteriormente en sip.con (80011 y 80012) si no también la del servidor con el que se va a comunicar

```

[entrantes]
exten => _9xxxx,1,Dial(IAX2/servidor/${EXTEN},90,tr)
exten => _9xxxx,2,Hangup()

exten => 80011,1,Dial(SIP/80011,10,tTrwW)
exten => 80011,2,Answer()
exten => 80011,n,Hangup()

exten => 80012,1,Dial(SIP/80012,10,tTrwW)
exten => 80012,2,Answer()
exten => 80012,n,Hangup()

```

Figura 9: configuración archivo extensions.conf

Donde:

Dial, answer y hangup indican el intento, respuesta y finalización de la llamada respectivamente.

8xxxx indica que las extensiones que maneja el servidor B inician por 8 y tienen 4 dígitos, por lo tanto estas pueden ser cualquiera entre 80000-89999.

Finalmente, (IAX2/servidora/EXTEND) responde al formato (IAX2/peer name/extend) que en este caso indica que el servidor B se comunicara con el servidora mediante el protocolo IAX2, EXTEND devuelve la extensión marcada.

Nota: las configuraciones del servidor A seran exactamente igual que la anteriores modificando las ip y los usuarios por 90011 y 90012

8.2.4. Verificación de la comunicación entre centralitas

La comunicación entre dos teléfonos de centrales diferentes, se puede observar en el CLI de cada una de las centrales, a continuación se muestra una llamada desde el usuario 90011(Servidor A) al usuario 80012(Servidor B):

```

-- Executing [80012@entrantes:1] Dial("SIP/90011-00000002", "IAX2/servidorb/
80012,90,tr") in new stack
-- Called IAX2/servidorb/80012
-- Call accepted by 192.168.210.53 (format gsm)
-- Format for call is gsm
-- IAX2/servidorb-17038 is ringing
-- IAX2/servidorb-17038 is ringing
-- IAX2/servidorb-17038 answered SIP/90011-00000002
-- Hungup 'IAX2/servidorb-17038'
== Spawn extension (entrantes, 80012, 1) exited non-zero on 'SIP/90011-0000000
2'

```

Figura 10: Server A

Donde se muestra cuando un usuario ubicado en el servidor B es llamado, acepta la llamada y por ultimo se establece la comunicación entre los dos usuarios.

```
License version 2 and other licenses; you are welcome to redistribute it under  
certain conditions. Type 'core show license' for details.  
=====  
== Parsing '/etc/asterisk/asterisk.conf': == Found  
== Parsing '/etc/asterisk/extconfig.conf': == Found  
Connected to Asterisk 1.8.13.1~dfsg-1ubuntu2 currently running on telematica (pi  
d = 2174)  
Verbosity is at least 17  
-- Accepting UNAUTHENTICATED call from 192.168.210.108:  
  > requested format = gsm,  
  > requested prefs = (),  
  > actual format = gsm,  
  > host prefs = (),  
  > priority = mine  
-- Executing [80012@entrantes:1] Dial("IAX2/servidora-3080", "SIP/80012,10,t  
TrWW") in new stack  
== Using SIP RTP CoS mark 5  
  -- Called SIP/80012  
  -- SIP/80012-00000005 is ringing  
  -- SIP/80012-00000005 answered IAX2/servidora-3080  
== Spawn extension (entrantes, 80012, 1) exited non-zero on 'IAX2/servidora-30  
80'  
  -- Hungup 'IAX2/servidora-3080'  
telematica*CLI> █
```

Figura 11: Server B

8.3. Comunicación Teléfono Ip

8.3.1. Configuracion Asterisk

Para la comunicacion entre el sistema Asterisk y un telefono Ip primero realizamos la configuracion del archivo **pjsip.conf** y **extensions.conf**

En el archivo extensions agregamos el sistema de marcado de las extensiones agregadas en el pjsip

```
[internal]
exten => 6001,1,Dial(PJSIP/6001,20)
exten => 6001,n,Hangup()

exten => 6002,1,Dial(PJSIP/6002,20)
exten => 6002,n,Hangup()
```

Figura 12: configuracion archivo extensions.conf

En el archivo pjsip agregamos el numero de las extensiones o usuarios que vamos a usar

```
[6001]
type = endpoint
context = internal
disallow = all
allow = ulaw
auth = 6001
aors = 6001

[6001]
type = auth
auth_type = userpass
password = 1234
username = 6001

[6001]
type = aor
max_contacts = 1

[6002]
type = endpoint
context = internal
disallow = all
allow = ulaw
auth = 6002
aors = 6002

[6002]
type = auth
auth_type = userpass
password = 1234
username = 6002

[6002]
type = aor
max_contacts = 2
```

Figura 13: configuracion archivo pjsip.conf

8.3.2. Configuración Teléfono

Posteriormente conectamos el teléfono ip a corriente y el lan al Ap, con la Ip que nos proporciona el teléfono ingresamos a un navegador para configurar las líneas SIP en la interfaz.

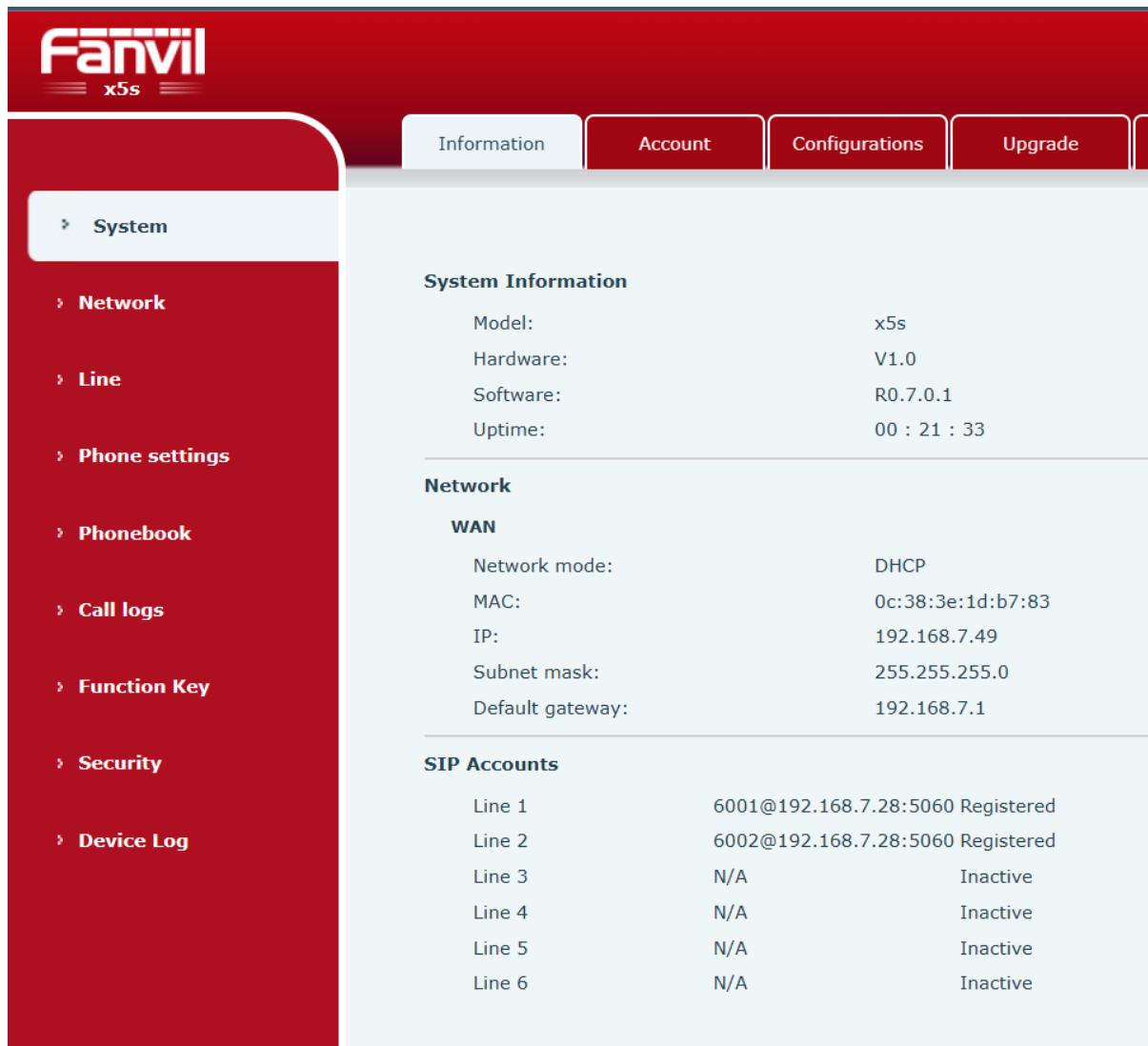


Figura 14: interfaz telefono Ip Fanvil X5s

Seleccionar el apartado Line para configurar las líneas SIP y escogemos la linea que deseamos configurar

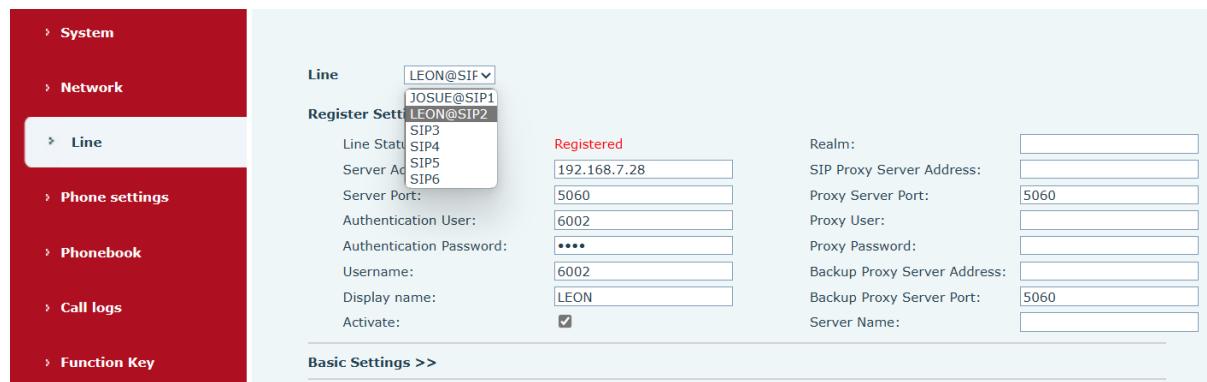


Figura 15: interfaz configuracion lineas SIP

Escoger las líneas que se desea configurar como se muestra en la siguiente figura

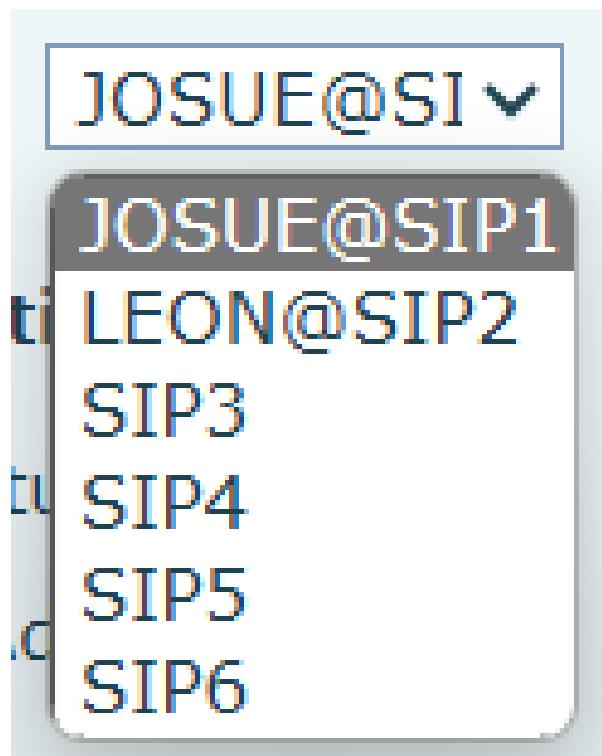


Figura 16: lineas SIP 1 y 2 escogidas para configurar

Configurar el apartado de Server Address, Server port, Auth-user, Passwd, User name y Display name en ambas líneas con las credenciales configuradas en Asterisk.

The screenshot shows the 'Register Settings' configuration page. The 'Line Status' is set to 'Registered'. The 'Server Address' is '192.168.7.28', 'Server Port' is '5060', 'Authentication User' is '6001', 'Authentication Password' is '****', 'Username' is '6001', and 'Display name' is 'JOSUE'. The 'Activate' checkbox is checked. On the right side, the 'Realm' field is empty. The 'SIP Proxy Server Address' is '192.168.7.28', 'Proxy Server Port' is '5060', 'Proxy User' is empty, 'Proxy Password' is empty, 'Backup Proxy Server Address' is empty, 'Backup Proxy Server Port' is '5060', and 'Server Name' is empty.

Figura 17: linea SIP 1 de Josue

The screenshot shows the 'Register Settings' configuration page. The 'Line Status' is set to 'Registered'. The 'Server Address' is '192.168.7.28', 'Server Port' is '5060', 'Authentication User' is '6002', 'Authentication Password' is '****', 'Username' is '6002', and 'Display name' is 'LEON'. The 'Activate' checkbox is checked. On the right side, the 'Realm' field is empty. The 'SIP Proxy Server Address' is '192.168.7.28', 'Proxy Server Port' is '5060', 'Proxy User' is empty, 'Proxy Password' is empty, 'Backup Proxy Server Address' is empty, 'Backup Proxy Server Port' is '5060', and 'Server Name' is empty.

Figura 18: linea SIP 2 de Jean

8.3.3. Verificación con MicroSIP

Configurar una cuenta en el MicroSIP para realizar la llamada desde nuestra pc al teléfono Ip.

The screenshot shows the 'Cuenta' configuration window. The fields are as follows: 'Nombre de cuenta' is '6002', 'Servidor SIP' is '192.168.7.28', 'Proxy SIP' is '192.168.7.28', 'Nombre de usuario *' is '6002', 'Dominio *' is '192.168.7.28', 'Acceso' is '6002', and 'Contraseña' is '****'.

Figura 19: cuenta en MicroSIP

Realizar una llamada desde la Pc a la extensión 6001 para el teléfono o desde el teléfono a la extensión 6002 para verificar la conexión.

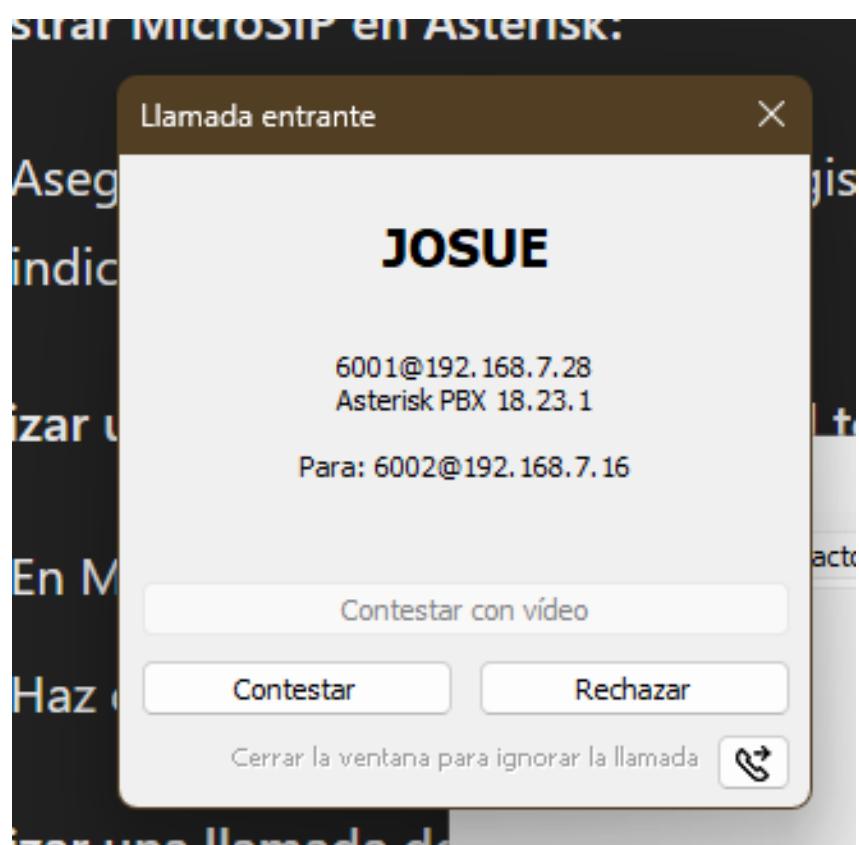


Figura 20: llamada realizada desde el teléfono a la Pc

8.4. Control De Dispositivo Físico ON/OFF con un foco utilizando Asterisk

Para el encendido y apagado del foco, se realizara por medio de peticiones HTTP por el método GET, levantando un servidor HTTP en la ESP8266, se instalan las librerías `curl`.^{en} el servidor para poder ejecutar sentencias por medio de consola.

8.4.1. Configuración de la ESP8266

Para la parte de la programación se utilizan las librerías de WebServer y la de Wifi para la comunicaciones por el protocolo TCP/IP:

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
```

Se ingresan las credenciales de la red, tanto para el usuario y la contraseña, además de abrir para el servidor HTTP el puerto 80 donde se enviaran los datos, y declaro el pin digital para controlar el encendido y apagado del foco.

```
const char* ssid = "LEON-RED";
const char* password = "76697978";
ESP8266WebServer server(80);
const int focoPin = 4; // Pin digital donde está conectado el LED
```

Para la parte del void `setup()`, se empieza con la comunicación serial y la conexión de Wifim para poder mostrar la dirección IP, asignada por la red, en este caso de utiliza un AP con un router PIX-LINK.

```
Serial.begin(115200);
WiFi.begin(ssid, password);
Serial.print("Connecting to WiFi");
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
}
Serial.println("\nWiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
```

Se declara los pines para el foco como salida, y se ubica en un estado bajo para el inicio del programa.

```
pinMode(focoPin, OUTPUT);
digitalWrite(focoPin, LOW); // Apaga el FOCO al inicio
```

Se crea las extensiones para el servidor con la dirección IP asignada por la red, esto para poder acceder a las URL: `http://192.168.7.3/encender` y `http://192.168.7.3/apagar`, entonces se visualizara en texto plano la pagina al acceder y se enviara un estado de alto para encender el foco, para el caso contrario será un estado en bajo para apagar el foco.

```
server.on("/encender", []() {
    Serial.println("Received request to /encender");
    digitalWrite(focoPin, HIGH); // HIGH enciende el FOCO
    server.send(200, "text/plain", "FOCO Encendido");
});

server.on("/apagar", []() {
    Serial.println("Received request to /apagar");
    digitalWrite(focoPin, LOW); // LOW apaga el FOCO
```

```

server.send(200, "text/plain", "FOCO Apagado");
});

server.onNotFound([]() {
    Serial.println("Received request to unknown path");
    server.send(404, "text/plain", "Not Found");
});

```

Al final del void setup(), se inicializa el servidor HTTP.

```

server.begin();
Serial.println("HTTP server started");

```

Para poder recibir las solicitudes de un cliente con un método GET, se detalla en el *void setup()* para poder estar en bucle cada que se envia una petición.

```
server.handleClient();
```

8.4.2. Configuración en Asterisk

Para la configuración se crea en este caso el archivo de configuración extensions.conf en la ubicación de /etc/asterisk; agregaremos en el contexto default la extension 800; se empezara grabando los audios en este caso para la bienvenida llamado audio-esp, otro para la elección entre 1. encender y 2. apagar; esc-op además se graba los audios para foco encendido led-on y para el foco apagado led-off se utiliza la función NoOp para poder corregir errores para visualizar por el CLI, los datos que ingresan o se manda, para después crear una condición que al elegir una elección se redirigía hacia la parte del dial de encender, apagar y se no se elegí uno de los valores me enviara a un audio de opción incorrecta.

```

root@SERVER:/etc/asterisk
GNU nano 5.6.1           extensions.conf

:ENCENDER UN FOCO - EXTENSION
exten => 800,1,Answer()
    same => 2,Playback(audio_esp) ; Un mensaje de bienvenida grabado
    same => 3,Read(choice,esc_op,1) ; Lee la elección del usuario (1 o 2)
    same => 4,NoOp(Opción elegida: ${choice})
    same => 5,GotoIf(${${choice} = 1}?encender:check_2)

exten => 800,6(check_2),GotoIf(${${choice} = 2}?apagar:incorrecto)

```

Para grabar los audios se utiliza la extensión 808, con la función record con formato .gsm del dial y para poder escuchar el audio se utiliza la extensión 809; para poder escuchar el audio.

```

;GRABACIONES DE AUDIO

exten => 808,1,record(encuesta_f.gsm,3)

exten => 809,1,answer()
    same => n,playback(encuesta_f)
    same => n,hangup()

```

Así, después de escoger algunas de las opciones se ingresa y se envía un mensaje a la consola de CLI que se envió la solicitud; se utiliza la función System para ejecutar por consola en el servidor de Alma Linux, con la librería curl de forma que se pueda mandar la petición HTTP con el método GET para guardar el dato de encender o apagar y se reproducira un audio donde se indique que el foco se ha encendido o

se ha apagado.



```
root@SERV01:/etc/asterisk
GNU nano 5.6.1           extensions.conf

; Encender el LED
exten => 800,n(encender),NoOp(Enviando solicitud para encender el LED)
same => n,System(curl -X GET http://192.168.7.5/encender)
same => n,Playback(led-on) ; Un mensaje grabado que diga "LED encendido"
same => n,Hangup()

; Apagar el LED
exten => 800,n(apagar),NoOp(Enviando solicitud para apagar el LED)
same => n,System(curl -X GET http://192.168.7.5/apagar)
same => n,Playback(led-off) ; Un mensaje grabado que diga "LED apagado"
same => n,Hangup()

; Opción incorrecta
exten => 800,n(incorrecto),NoOp(Opción incorrecta seleccionada)
same => n,Playback(invalido_cs) ; Un mensaje grabado que diga "Opción incorrecta"
same => n,Hangup()

^G Help      ^C Write Out   ^W Where Is    ^K Cut        ^T Execute     ^C Location
^X Exit      ^R Read File   ^\ Replace     ^U Paste      ^J Justify     ^L Go To Line
```

En el caso de que no se tenga las librerías de curl”, se instalará con los siguientes comando:

```
sudo dnf install curl
```

8.4.3. Verificación de encendido y apagado del foco

Se llama a la extensión 800; se verifica tanto por la aplicación de MicroSIP, como la aplicación del móvil utilizado como el Zoiper así se obtienen los siguientes resultados:

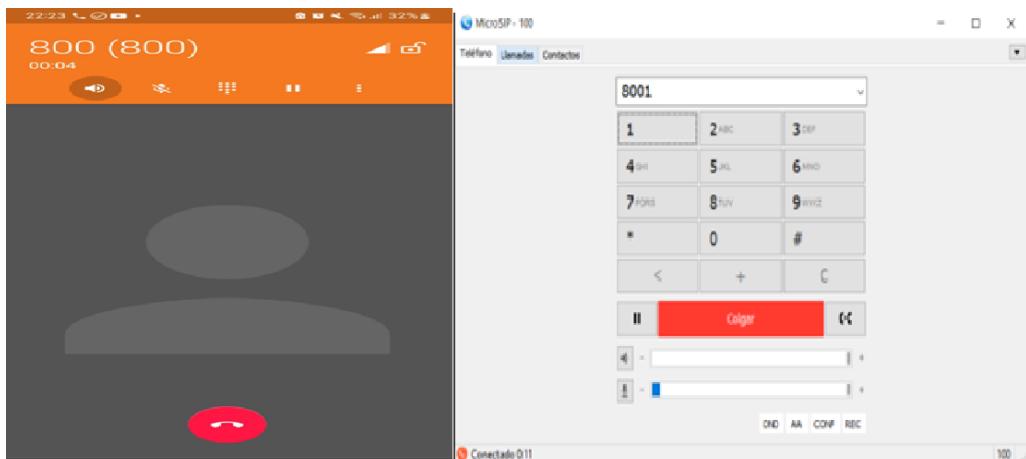


Figura 21: Llamadas realizadas a la extensión 800



Figura 22: Pruebas realizadas con la ESP8266

Se verifica que entra la llamada a la extensión, y se envía la sentencia al servidor, además de enviar la solicitud y visualizar que se tiene un correcto funcionamiento en cuanto a la reproducción de los audios grabados, y se obtiene una respuesta correcta del servidor HTTP en la Node MCU ESP8266.

```
== Contact 101@sip:101@192.168.7.4:46948;transport=UDP;rinstance=77d848796c80db11 has been
deleted
-- Added contact 'sip:101@192.168.7.4:36171;transport=UDP;rinstance=73ea578cc252a5c6' to
AOR '101' with expiration of 60 seconds
-- Executing [800@default:1] Answer("PJSIP/100-00000002", "") in new stack
-- Executing [800@default:2] Playback("PJSIP/100-00000002", "audio_esp") in new stack
-- <PJSIP/100-00000002> Playing 'audio_esp.gsm' (language 'es')
-- Executing [800@default:3] Read("PJSIP/100-00000002", "choice,esc_op,1") in new stack
-- Accepting a maximum of 1 digits.
-- <PJSIP/100-00000002> Playing 'esc_op.gsm' (language 'es')
SERVER*CLI> 
```

```
-- User entered '2'
-- Executing [800@default:4] NoOp("PJSIP/100-00000002", "Opción elegida: 2") in new stack
-- Executing [800@default:5] GotoIf("PJSIP/100-00000002", "0?encender:check_2") in new st
ack
-- Goto (default,800,6)
-- Executing [800@default:6] GotoIf("PJSIP/100-00000002", "1?apagar:incorrecto") in new s
tack
-- Goto (default,800,11)
-- Executing [800@default:11] NoOp("PJSIP/100-00000002", "Enviando solicitud para apagar
el LED") in new stack
-- Executing [800@default:12] System("PJSIP/100-00000002", "curl -X GET http://192.168.7.
5/apagar") in new stack
-- Executing [800@default:13] Playback("PJSIP/100-00000002", "led-off") in new stack
-- <PJSIP/100-00000002> Playing 'led-off.gsm' (language 'es')
-- Executing [800@default:14] Hangup("PJSIP/100-00000002", "") in new stack
== Spawn extension (default, 800, 14) exited non-zero on 'PJSIP/100-00000002'
SERVER*CLI> 
```

Datos recibidos correctamente.
Received request to /encender

Figura 23: Petición recibida en el servidor HTTP, /encender

Datos recibidos correctamente.
Received request to /apagar

Figura 24: Petición recibida en el servidor HTTP, /apagar

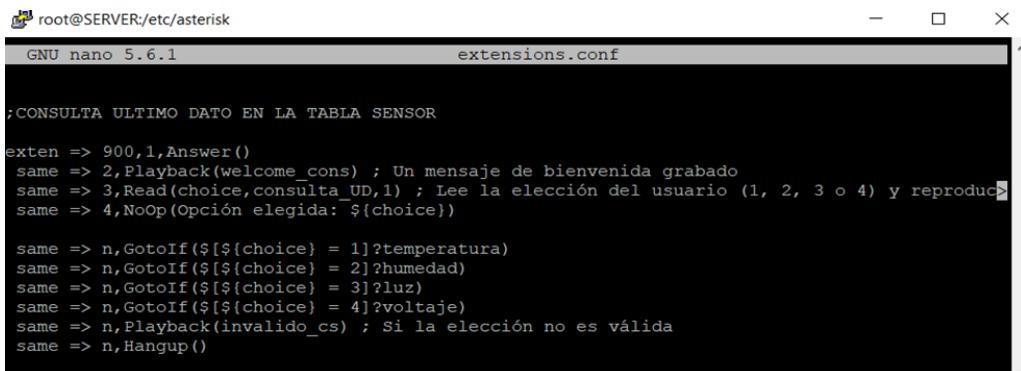
8.5. Consultar El Ultimo Dato

8.5.1. Configuración de Asterisk

Para la configuración de asterisk se tendrá en cuenta que deberemos tener una conexión estable a la base de datos, teniendo cargados los modulos odbc en el servidor, una vez tenemos aquello nos pasamos hacia la ubicación /etc/asterisk/ y configuraremos el archivo func_odbc.conf, al final del archivo añadimos un contexto [UltimoDato], donde agregamos la sentencia SQL, SELECT para poder leer el ultimo dato de la base de datos "sensores" de la tabla sensor.

```
[UltimoDato]
dsn=asterisk
readsql=SELECT ${ARG1} FROM sensor ORDER BY id DESC LIMIT 1
```

Ahora configuraremos el archivo de extensions.conf para agregar la extensión de la lectura del ultimo dato que sera la extensión 900, la cuál añadiremos primeramente un mensaje de saludo, teniendo en cuenta que se debe pregrabar los audios, empezando por el welcome, despues un audio de las opciones Audio_UD, además de un audio para cada uno de los datos como temperatura, humedad, luz y voltaje, la prioridad agregada siempre será la número 1 para contestar las solicitudes entrantes.



```
root@SERVER:/etc/asterisk
GNU nano 5.6.1 extensions.conf

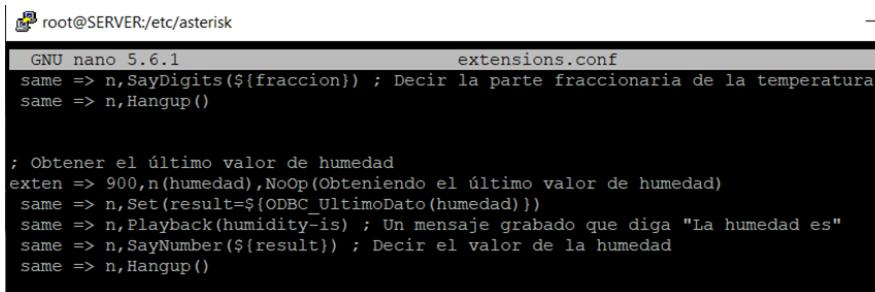
;CONSULTA ULTIMO DATO EN LA TABLA SENSOR

exten => 900,1,Answer()
same => 2,Playback(welcome_cons) ; Un mensaje de bienvenida grabado
same => 3,Read(choice,consulta_UD,1) ; Lee la elección del usuario (1, 2, 3 o 4) y reproduce
same => 4,NoOp(Opción elegida: ${choice})

same => n,GotoIf(${${choice} = 1}?temperatura)
same => n,GotoIf(${${choice} = 2}?humedad)
same => n,GotoIf(${${choice} = 3}?luz)
same => n,GotoIf(${${choice} = 4}?voltaje)
same => n,Playback(invalido_cs) ; Si la elección no es válida
same => n,Hangup()
```

Las funciones utilizadas de asterisk que llamamos serán NoOp para poder visualizar datos por la consola CLI, se añade una función Get para poder ingresar por teclado los números necesarios para las opciones, y se realiza condiciones con Gotoif para redirijir a una opción de acuerdo a la elección.

Una vez elegida la opción, se envía hacia el apartado ya sea de temperatura, humedad, luz y voltaje, se ubica igual una función NoOp para visualizar que se ha redirijido hacia esa opción, se reproduce un audio en donde se escuchara según el dato, si es temperatura, humedad, luz o voltaje, para llamar con una variable al contexto [UltimoDato] para utilizar la sentencia SQL, para despues leer el dato por un audio pregrabado por el bot, tener en cuenta que los audios son en formato wav o gsm.



```
root@SERVER:/etc/asterisk
GNU nano 5.6.1 extensions.conf

same => n,SayDigits(${fraccion}) ; Decir la parte fraccionaria de la temperatura
same => n,Hangup()

; Obtener el último valor de humedad
exten => 900,n(humedad),NoOp(Obteniendo el último valor de humedad)
same => n,Set(result=${ODBC_UltimoDato(humedad)})
same => n,Playback(humidity-is) ; Un mensaje grabado que diga "La humedad es"
same => n,SayNumber(${result}) ; Decir el valor de la humedad
same => n,Hangup()
```

```

; Obtener el último valor de luz
exten => 900,n(luz),NoOp(Obteniendo el último valor de luz)
same => n,Set(result=${ODBC_UltimoDato(luz)})
same => n,Playback(light-is) ; Un mensaje grabado que diga "La luz es"
same => n,SayNumber(${result}) ; Decir el valor de la luz
same => n,Hangup()

```

En el caso de tener números decimales se ha dividido la lectura de los datos ya sea un valor entero o fraccionario, para que se pueda reproducir la extensión y tener una lectura correcta.

```

; Obtener el último valor de temperatura y manejar el valor decimal
exten => 900,n(temperatura),NoOp(Obteniendo el último valor de temperatura)
same => n,Set(result=${ODBC_UltimoDato(temperatura)})
same => n,Set(entero=${CUT(result,,1)}) ; Obtener la parte entera
same => n,Set(fraccion=${CUT(result,,2)}) ; Obtener la parte fraccionaria
same => n,Playback(temperature-is) ; Un mensaje grabado que diga "La temperatura en °C es"
same => n,SayNumber(${entero}) ; Decir la parte entera de la temperatura
same => n,SayDigits(${fraccion}) ; Decir la parte fraccionaria de la temperatura
same => n,Hangup()

```

8.5.2. Verificación de Resultados

Se observa que la extensión creada es respondida por parte de los usuarios, ya sea con la aplicación de MicroSip o Zoiper.

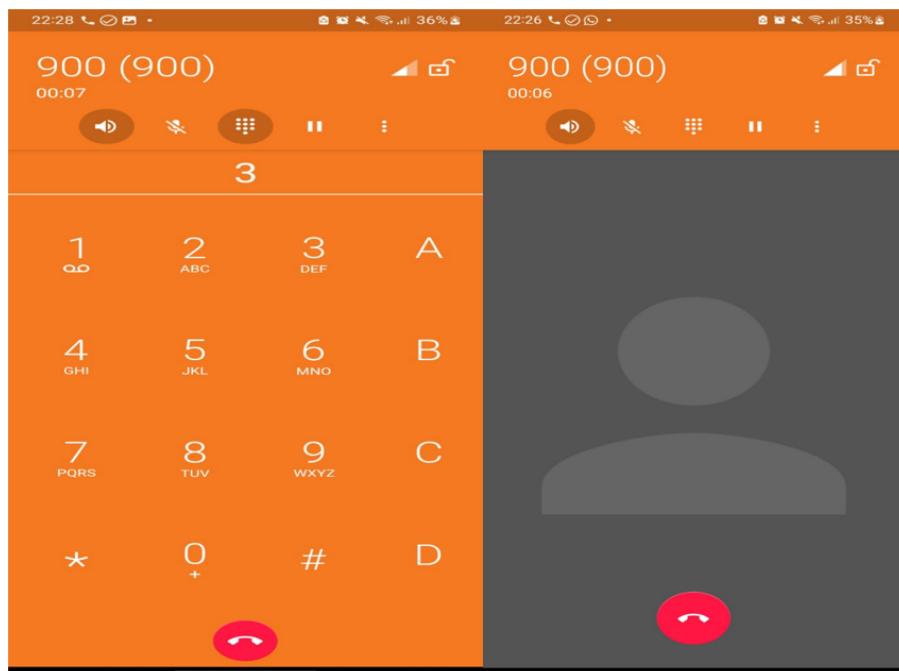


Figura 25: Entrada de la llamada en la aplicación móvil Zoiper

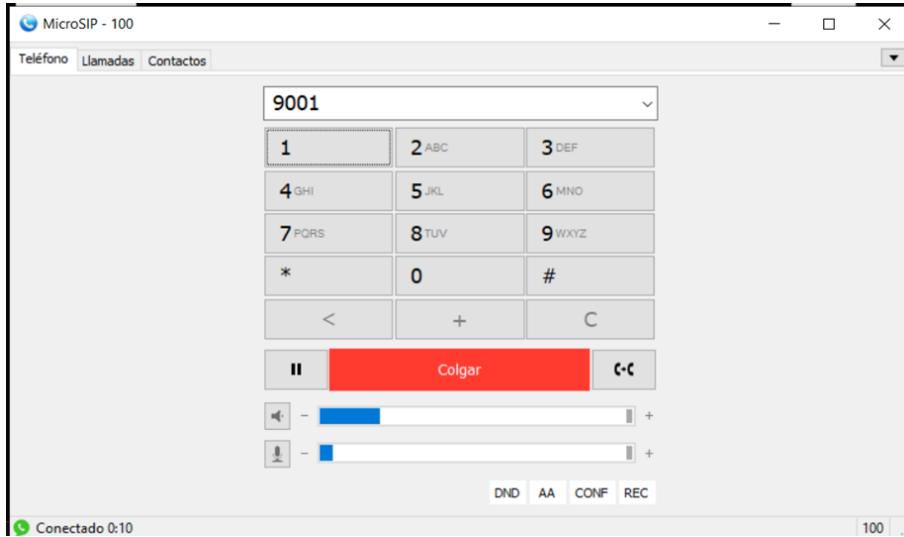


Figura 26: Entrada de la llamada en la aplicación de microSIP del ordenador

Se observa una lectura correcta de los datos en la consola de CLI, de forma que el funcionamiento es correcto.

```
-- Executing [9000default:1] Answer("PJSIP/101-00000003", "") in new stack
-- Executing [9000default:2] Playback("PJSIP/101-00000003", "welcome_cons") in new stack
-- <PJSIP/101-00000003> Playing 'welcome_cons.gsm' (language 'es')
-- Executing [9000default:3] Read("PJSIP/101-00000003", "choice,consulta_UD,1") in new stack
ack
-- Accepting a maximum of 1 digits.
-- <PJSIP/101-00000003> Playing 'consulta_UD.gsm' (language 'es')
-- User entered '2'
-- Executing [9000default:4] Noop("PJSIP/101-00000003", "Opción elegida: 2") in new stack
-- Executing [9000default:5] GotoIf("PJSIP/101-00000003", "0?temperatura") in new stack
-- Executing [9000default:6] GotoIf("PJSIP/101-00000003", "1?humedad") in new stack
-- Goto (default,900,19)
-- Executing [9000default:19] Noop("PJSIP/101-00000003", "Obteniendo el último valor de h
umedad") in new stack
-- Executing [9000default:20] Set("PJSIP/101-00000003", "result=53") in new stack
-- Executing [9000default:21] Playback("PJSIP/101-00000003", "humidity-is") in new stack
-- <PJSIP/101-00000003> Playing 'humidity-is.gsm' (language 'es')
-- Executing [9000default:22] SayNumber("PJSIP/101-00000003", "53") in new stack
-- <PJSIP/101-00000003> Playing 'digits/50.slin' (language 'es')
-- <PJSIP/101-00000003> Playing 'digits/and.slin' (language 'es')
-- <PJSIP/101-00000003> Playing 'digits/3.slin' (language 'es')
-- Executing [9000default:23] Hangup("PJSIP/101-00000003", "") in new stack
== Spawn extension (default, 900, 23) exited non-zero on 'PJSIP/101-00000003'
SERVER*CLI>
```

Figura 27: Mensajes por consola al seleccionar la humedad

```
-- Executing [9000default:1] Answer("PJSIP/101-00000003", "") in new stack
-- Executing [9000default:2] Playback("PJSIP/101-00000003", "welcome_cons") in new stack
-- <PJSIP/101-00000003> Playing 'welcome_cons.gsm' (language 'es')
-- Executing [9000default:3] Read("PJSIP/101-00000003", "choice,consulta_UD,1") in new stack
ack
-- Accepting a maximum of 1 digits.
-- <PJSIP/101-00000003> Playing 'consulta_UD.gsm' (language 'es')
-- User entered '2'
-- Executing [9000default:4] Noop("PJSIP/101-00000003", "Opción elegida: 2") in new stack
-- Executing [9000default:5] GotoIf("PJSIP/101-00000003", "0?temperatura") in new stack
-- Executing [9000default:6] GotoIf("PJSIP/101-00000003", "1?humedad") in new stack
-- Goto (default,900,19)
-- Executing [9000default:19] Noop("PJSIP/101-00000003", "Obteniendo el último valor de h
umedad") in new stack
-- Executing [9000default:20] Set("PJSIP/101-00000003", "result=53") in new stack
-- Executing [9000default:21] Playback("PJSIP/101-00000003", "humidity-is") in new stack
-- <PJSIP/101-00000003> Playing 'humidity-is.gsm' (language 'es')
-- Executing [9000default:22] SayNumber("PJSIP/101-00000003", "53") in new stack
-- <PJSIP/101-00000003> Playing 'digits/50.slin' (language 'es')
-- <PJSIP/101-00000003> Playing 'digits/and.slin' (language 'es')
-- <PJSIP/101-00000003> Playing 'digits/3.slin' (language 'es')
-- Executing [9000default:23] Hangup("PJSIP/101-00000003", "") in new stack
== Spawn extension (default, 900, 23) exited non-zero on 'PJSIP/101-00000003'
SERVER*CLI>
```

Figura 28: Mensajes por consola al seleccionar la temperatura

```

-- Executing [900@default:1] Answer("PJSIP/101-00000005", "") in new stack
-- Executing [900@default:2] Playback("PJSIP/101-00000005", "welcome_cons") in new stack
-- <PJSIP/101-00000005> Playing 'welcome_cons.gsm' (language 'es')
-- Executing [900@default:3] Read("PJSIP/101-00000005", "choice,consulta_UD,1") in new stack
ck
-- Accepting a maximum of 1 digits.
-- <PJSIP/101-00000005> Playing 'consulta_UD.gsm' (language 'es')
-- User entered '3'
-- Executing [900@default:4] NoOp("PJSIP/101-00000005", "Opción elegida: 3") in new stack
-- Executing [900@default:5] GotoIf("PJSIP/101-00000005", "0?temperatura") in new stack
-- Executing [900@default:6] GotoIf("PJSIP/101-00000005", "0?humedad") in new stack
-- Executing [900@default:7] GotoIf("PJSIP/101-00000005", "1?luz") in new stack
-- Goto (default,900,24)
-- Executing [900@default:24] Noop("PJSIP/101-00000005", "Obteniendo el último valor de luz")
z") in new stack
-- Executing [900@default:25] Set("PJSIP/101-00000005", "result=50") in new stack
-- Executing [900@default:26] Playback("PJSIP/101-00000005", "light-is")
<PJSIP/101-00000005> Playing 'light-is.gsm' (language 'es')
-- Executing [900@default:27] SayNumber("PJSIP/101-00000005", "50") in new stack
<PJSIP/101-00000005> Playing 'digits/50.slin' (language 'es')
-- Executing [900@default:28] Hangup("PJSIP/101-00000005", "") in new stack
== Spawn extension (default, 900, 28) exited non-zero on 'PJSIP/101-00000005'

```

Figura 29: Mensajes por consola al seleccionar la luz

```

-- Executing [900@default:1] Answer("PJSIP/100-00000006", "") in new stack
-- Executing [900@default:2] Playback("PJSIP/100-00000006", "welcome_cons") in new stack
-- <PJSIP/100-00000006> Playing 'welcome_cons.gsm' (language 'es')
-- Executing [900@default:3] Read("PJSIP/100-00000006", "choice,consulta_UD,1") in new stack
ck
-- Accepting a maximum of 1 digits.
-- <PJSIP/100-00000006> Playing 'consulta_UD.gsm' (language 'es')
-- User entered '4'
-- Executing [900@default:4] NoOp("PJSIP/100-00000006", "Opción elegida: 4") in new stack
-- Executing [900@default:5] GotoIf("PJSIP/100-00000006", "0?temperatura") in new stack
-- Executing [900@default:6] GotoIf("PJSIP/100-00000006", "0?humedad") in new stack
-- Executing [900@default:7] GotoIf("PJSIP/100-00000006", "0?luz") in new stack
-- Executing [900@default:8] GotoIf("PJSIP/100-00000006", "1?voltaje") in new stack
-- Goto (default,900,29)
-- Executing [900@default:29] Noop("PJSIP/100-00000006", "Obteniendo el último valor de voltaje") in new stack
-- Executing [900@default:30] Set("PJSIP/100-00000006", "result=0") in new stack
-- Executing [900@default:31] Set("PJSIP/100-00000006", "entero=0") in new stack
-- Executing [900@default:32] Set("PJSIP/100-00000006", "fraccion=") in new stack
-- Executing [900@default:33] Playback("PJSIP/100-00000006", "voltage-is")
<PJSIP/100-00000006> Playing 'voltage-is.gsm' (language 'es')
-- Executing [900@default:34] SayNumber("PJSIP/100-00000006", "0") in new stack
<PJSIP/100-00000006> Playing 'digits/0.slin' (language 'es')
-- Executing [900@default:35] SayDigits("PJSIP/100-00000006", "") in new stack
-- Executing [900@default:36] Hangup("PJSIP/100-00000006", "") in new stack
== Spawn extension (default, 900, 36) exited non-zero on 'PJSIP/100-00000006'

```

Figura 30: Mensajes por consola al seleccionar la voltaje

Además, se podrá verificar con una interfaz creada en php, para visualizar los datos de la tabla sensor de la base de datos sensores, solo de los últimos tres datos.

Verificar Mensaje				
Últimos Tres datos ingresados en la tabla sensor:				
ID	Temperatura	Humedad	Luz	Voltaje
2576	20.7	54	53	0
2575	20.7	54	53	0
2574	20.7	54	53	0

Figura 31: Interfaz para muestra de la tabla de los últimos tres datos de la tabla sensor

8.6. Encuesta De Satisfacción

8.6.1. Script de php creado

En el inicio del script se empieza declarando las variables para la conexión a la base de datos, empezando por establecer por localhost, se añade un usuario y contraseña, y por ultimo el nombre de la base de datos que sera "sensores".

```
// Configuración de la base de datos
$servername = "localhost";
$username = "comunicaciones";
$password = "1234";
$dbname = "sensores";
```

Despues se utiliza otra variable para poder utilizar la función "mysqli", para establecer la conexión.

```
$conn = new mysqli($servername, $username, $password, $dbname);
```

Se declara la sentencia SQL, INSERT para insertar los datos a la tabla votacion, con el número de columnas Ncedula y Op.

```
$sql = "INSERT INTO votacion (Ncedula, Op) VALUES ('$ncedula', '$op')";
```

Se recoje los datos con dos variables, donde se utiliza argument para recojer los datos de la base de datos.

```
$ncedula = $argv[1];
$op = $argv[2];
```

Se realiza un condicional para poder verificar si el dato se ingreso correctamente, con una variable stm que verifica la conexión y la subida de los datos.

```
// Validar datos
if (strlen($ncedula) == 10 && in_array($op, ['1', '2', '3'])) {
    insertarVotacion($conn, $ncedula, $op);
} else {
    echo "Datos inválidos. Asegúrate de que la cédula tenga 10 números y la opción sea 1, 2 o 3.\n";
}

if ($stmt->execute()) {
    echo "Nuevo registro creado exitosamente\n";
} else {
    echo "Error: " . $stmt->error . "\n";
}
```

Una vez hecho aquello se desconecta de la base de datos y finaliza el script.

```
$conn->close();
```

8.6.2. Configuración de Asterisk

Para la configuración de asterisk se tendra en cuenta que deberemos tener una conexión estable a la base de datos, teniendo cargados los modulos odbc en el servidor, se verifica en el CLI de asterisk con el comando ".dbc show all".

Una vez verificado la conexión, se crea un script en php para insertar los datos desde la extensión. Nos redirigimos a la ubicación /etc/asterisk para configurar el archivo de extensions.conf para agregar la extensión de votación que sera la 850, la cuál añadiremos primeramente un mensaje de saludo, teniendo en cuenta que se debe pregrabar los audios, empezando por el welcome que sera audio-encuesta, después un audio de las opciones pt-encuesta, además de un audio para cuando se finaliza la encuesta, la prioridad utilizada será 1, para poder empezar la llamada con lo antes mencionado

```

GNU nano 5.6.1                               extensions.conf
;5.-Formulario - ENCUESTA SATISFACCIÓN DE CLASE;

exten => 850,1,answer()
    same => 2,playback(audio_encuesta)
    same => 3,read(cedula,pet_ced,10)
    same => n,GotoIf(${LEN(${cedula})} != 10)?invalid_cedula,1)
    same => n,read(op,audio_op,1)
    same => n,GotoIf(${op} < 1 | ${op} > 3)?invalid_option,1)
    same => n,NoOp(Número de cédula: ${cedula}, Opción: ${op})
    same => n,System(/home/jean/scripts2php/ingreso_bd.php ${cedula} ${op})
    same => n,NoOp(Resultado de ODBC_Encuesta: ${result})
    same => n,Playback(encuesta_f)
    same => n,hangup()

exten => 850,n(invalid_cedula),playback(invalid_cedula)
    same => n,hangup()

exten => 850,n(invalid_option),playback(invalido_cs)
    same => n,hangup()

^G Help      ^O Write Out   ^W Where Is     ^K Cut        ^T Execute   ^C Location
^X Exit      ^R Read File   ^\ Replace      ^U Paste      ^J Justify    ^

```

Las funciones utilizadas de asterisk que llamamos serán NoOp para poder visualizar datos por la consola CLI, se añade una función Get para poder ingresar por teclado el número de cédula y las opciones ha escoger, y se realiza condiciones con Gotoif para redirigir a una opción de acuerdo a la elección, se utiliza siempre para entrar a la lectura con prioridad n, despues de mandar lo anterior.

Se utiliza la función System para poder correr el script de php, se utiliza la línea de código siguiente:

```
same => n,System(/home/jean/scripts2php/ingreso_bd.php ${cedula} ${op})
```

8.6.3. Verificación de Resultados

Se observa que la extensión creada es respondida por parte de los usuarios, ya sea con la aplicación de MicroSip o Zoiper.

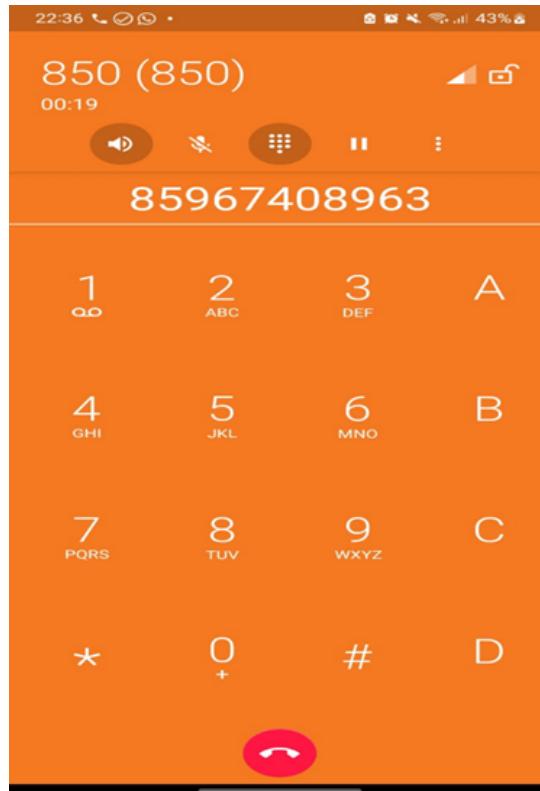


Figura 32: Entrada de la llamada en la aplicación móvil Zoiper

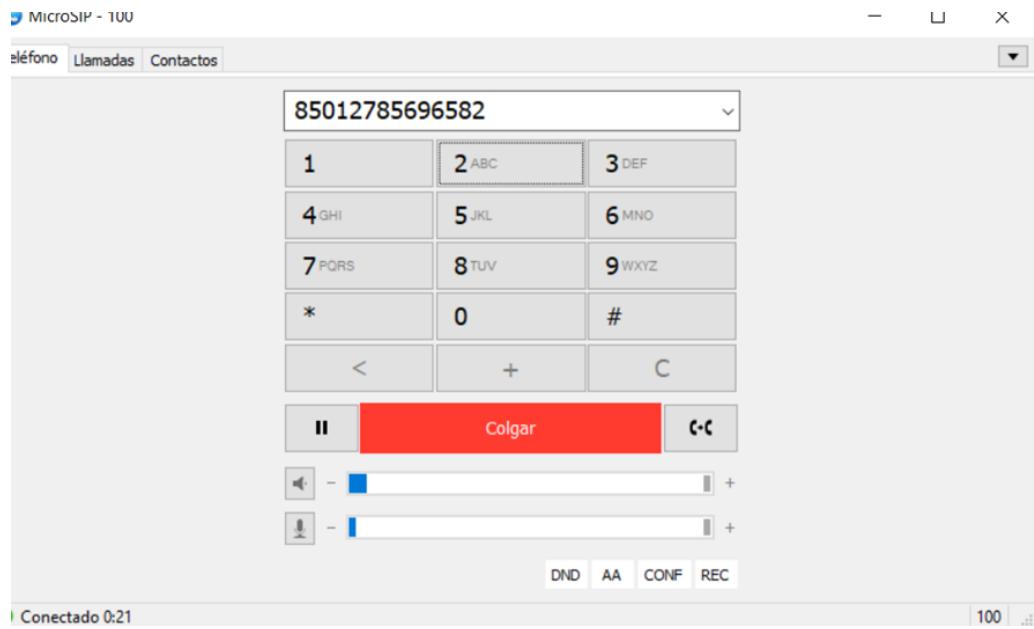


Figura 33: Entrada de la llamada en la aplicación de microSIP del ordenador

Se observa una lectura correcta de los datos en la consola de CLI, de forma que el funcionamiento es correcto.

```

== Spawn extension (default, 900, 36) exited non-zero on 'PJSIP/100-00000006'
-- Executing [850@default:1] Answer("PJSIP/100-00000007", "") in new stack
-- Executing [850@default:2] Playback("PJSIP/100-00000007", "audio_encuesta") in new stack
-- <PJSIP/100-00000007> Playing 'audio_encuesta.gsm' (language 'es')
-- Executing [850@default:3] Read("PJSIP/100-00000007", "cedula,pet_ced,10") in new stack
-- Accepting a maximum of 10 digits.
-- <PJSIP/100-00000007> Playing 'pet_ced.gsm' (language 'es')
-- User entered '1719564780'
-- Executing [850@default:4] GotoIf("PJSIP/100-00000007", "0?invalid_cedula,1") in new stack
-- Executing [850@default:5] Read("PJSIP/100-00000007", "op,audio_op,1") in new stack
-- Accepting a maximum of 1 digits.
-- <PJSIP/100-00000007> Playing 'audio_op.gsm' (language 'es')
-- User entered '1'
-- Executing [850@default:6] GotoIf("PJSIP/100-00000007", "0?invalid_option,1") in new stack
-- Executing [850@default:7] NoOp("PJSIP/100-00000007", "Número de cédula: 1719564780, Opción: 1") in new stack
-- Executing [850@default:8] System("PJSIP/100-00000007", "php /home/jean/scripts2php/ingreso_bd.php 1719564780 1") in new stack
-- Executing [850@default:9] NoOp("PJSIP/100-00000007", "Resultado de ODBC_Encuesta: ") in new stack
-- Executing [850@default:10] Playback("PJSIP/100-00000007", "encuesta_f") in new stack
-- <PJSIP/100-00000007> Playing 'encuesta_f.gsm' (language 'es')
-- Executing [850@default:11] Hangup("PJSIP/100-00000007", "") in new stack
== Spawn extension (default, 850, 11) exited non-zero on 'PJSIP/100-00000007'
[SERVER*CLI]> █

```

Figura 34: Mensajes por consola al seleccionar la opción 1

```

-- Executing [850@default:1] Answer("PJSIP/100-00000009", "") in new stack
-- Executing [850@default:2] Playback("PJSIP/100-00000009", "audio_encuesta") in new stack
-- <PJSIP/100-00000009> Playing 'audio_encuesta.gsm' (language 'es')
-- Executing [850@default:3] Read("PJSIP/100-00000009", "cedula,pet_ced,10") in new stack
-- Accepting a maximum of 10 digits.
-- <PJSIP/100-00000009> Playing 'pet_ced.gsm' (language 'es')
-- Executing [850@default:1] Answer("PJSIP/101-0000000a", "") in new stack
-- Executing [850@default:2] Playback("PJSIP/101-0000000a", "audio_encuesta") in new stack
-- <PJSIP/101-0000000a> Playing 'audio_encuesta.gsm' (language 'es')
-- Executing [850@default:3] Read("PJSIP/101-0000000a", "cedula,pet_ced,10") in new stack
-- Accepting a maximum of 10 digits.
-- <PJSIP/101-0000000a> Playing 'pet_ced.gsm' (language 'es')
-- User entered '5807962380'
-- Executing [850@default:4] GotoIf("PJSIP/100-00000009", "0?invalid_cedula,1") in new stack
k
-- Executing [850@default:5] Read("PJSIP/100-00000009", "op,audio_op,1") in new stack
-- Accepting a maximum of 1 digits.
-- <PJSIP/100-00000009> Playing 'audio_op.gsm' (language 'es')
-- User entered '2'
-- Executing [850@default:6] GotoIf("PJSIP/100-00000009", "0?invalid_option,1") in new stack
k
-- Executing [850@default:7] NoOp("PJSIP/100-00000009", "Número de cédula: 5807962380, Opción: 2") in new stack
-- Executing [850@default:8] System("PJSIP/100-00000009", "php /home/jean/scripts2php/ingreso_bd.php 5807962380 2") in new stack
-- Executing [850@default:9] NoOp("PJSIP/100-00000009", "Resultado de ODBC_Encuesta: ") in new stack

```

Figura 35: Mensajes por consola al seleccionar la opción 2

```

-- Executing [850@default:10] Playback("PJSIP/100-00000009", "encuesta_f") in new stack
-- <PJSIP/100-00000009> Playing 'encuesta_f.gsm' (language 'es')
-- Executing [850@default:11] Hangup("PJSIP/100-00000009", "") in new stack
== Spawn extension (default, 850, 11) exited non-zero on 'PJSIP/100-00000009'
-- User entered '8596740896'
-- Executing [850@default:4] GotoIf("PJSIP/101-0000000a", "0?invalid_cedula,1") in new stack
k
-- Executing [850@default:5] Read("PJSIP/101-0000000a", "op,audio_op,1") in new stack
-- Accepting a maximum of 1 digits.
-- <PJSIP/101-0000000a> Playing 'audio_op.gsm' (language 'es')
-- User entered '3'
-- Executing [850@default:6] GotoIf("PJSIP/101-0000000a", "0?invalid_option,1") in new stack
k
-- Executing [850@default:7] NoOp("PJSIP/101-0000000a", "Número de cédula: 8596740896, Opción: 3") in new stack
-- Executing [850@default:8] System("PJSIP/101-0000000a", "php /home/jean/scripts2php/ingreso_bd.php 8596740896 3") in new stack
-- Executing [850@default:9] NoOp("PJSIP/101-0000000a", "Resultado de ODBC_Encuesta: ") in new stack
-- Executing [850@default:10] Playback("PJSIP/101-0000000a", "encuesta_f") in new stack
-- <PJSIP/101-0000000a> Playing 'encuesta_f.gsm' (language 'es')
-- Executing [850@default:11] Hangup("PJSIP/101-0000000a", "") in new stack
== Spawn extension (default, 850, 11) exited non-zero on 'PJSIP/101-0000000a'
-- Removed contact 'sip:101@192.168.7.4:44160;transport=UDP;rinstance=ale29bcald413a4c' from AOR '101' due to request
-- Contact 101@sip:101@192.168.7.4:44160;transport=UDP;rinstance=ale29bcald413a4c has been deleted
SERVER*CLI> █

```

Figura 36: Mensajes por consola al seleccionar la opción 2

Además, se podrá verificar con una interfaz creada en php, para visualizar los datos de la tabla votacion de la base de datos sensores, solo de los últimos tres datos.

Formulario Comunicaciones Avanzadas

TABLA VOTACIONES

Ncuela	Op	Significado
1234123412	3	Malo
1234128902	3	Malo
1278569658	2	Regular
1719564780	1	Bueno
1719824257	3	Malo
8585850889	3	Malo
8596740896	3	Malo

Actualizar Tabla

Figura 37: Interfaz para muestra de la tabla de los datos en la tabla votación

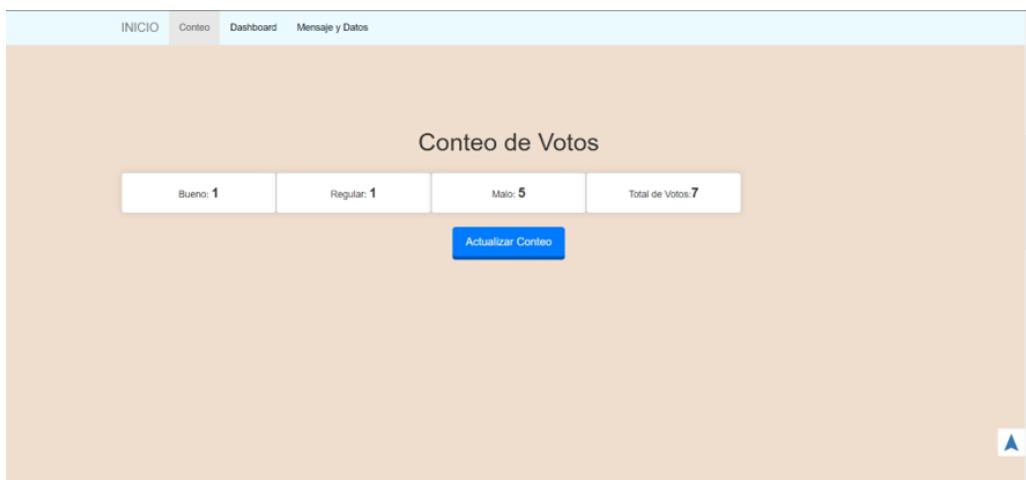


Figura 38: Interfaz para conteo de datos de cada opción y un conteo total

8.7. Sistema de Alarmas para Niveles De Temperatura, Humedad Y Luminosidad basado en Llamadas de alerta en Asterisk para prevención de incendios

8.7.1. Configuración de la ESP8266

Para la configuración del Node MCU, se utilizan las librerías de Wifi, HTTP Client, Wifi Client y Web-Server. Además, en los sensores se añade las librerías de DHT, para el sensor de temperatura y humedad DHT11.

```
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <DHT.h>
#define DHTTYPE DHT11
#define dht_dpin 5 // PIN D1 PARA DHT11
```

Se declaran las constantes y se definen los puertos para la foto resistencia, para el sensor DHT11, los pines de los led's para los indicadores, las credenciales de la red tanto el SSID y la contraseña, la URL de la dirección del script en php para poder ejecutarlo, además para las alertas se crea otro URL para enviar datos al servidor Flask.

```
#define dht_dpin 5 // PIN D1 PARA DHT11
DHT dht(dht_dpin, DHTTYPE);

const char* ssid = "LEON-RED";
const char* password = "76697978";

const char* serverNameBD = "http://192.168.7.2/Alarma_asterisk/insertar_datos.php";
const char* serverNameFlask = "http://192.168.7.4:80/datos";

const int fotoresistorPin = A0;
```

```
const int ledLuzBajo = 14; // PIN D4 LED para indicar luz baja
const int ledLuzMedio = 12; // PIN D6 LED para indicar luz media
const int ledLuzAlto = 13; // PIN D7 LED para indicar luz alta
```

Para la parte del void setup(), se empieza con la comunicación serial y la conexión de Wifi para poder mostrar la dirección IP, asignada por la red, en este caso de utiliza un AP con un router PIX-LINK.

```
Serial.begin(115200);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.println("Connecting to WiFi...");
}

Serial.println("Connected to WiFi");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
```

Además se necesita para declarar los pines digitales de salida para los Led's, y se añade para que empiece con un estado en bajo.

```
pinMode(ledLuzBajo, OUTPUT);
pinMode(ledLuzMedio, OUTPUT);
pinMode(ledLuzAlto, OUTPUT);
```

```

digitalWrite(ledLuzBajo, LOW);
digitalWrite(ledLuzMedio, LOW);
digitalWrite(ledLuzAlto, LOW);

```

Para el void loop(), se agrega un condicional para verificar el estado de la red, y se permanece conectado se corre todo dentro del condicional.

```

if (WiFi.status() == WL_CONNECTED) {
}

```

Se declaran algunas variables para la parte del sensor para los datos de temperatura y humedad, además para leer los datos análogos del fotoresistor y poder mapear la luminosidad en un rango específico.

```

// Leer la temperatura y la humedad
float humidity = dht.readHumidity();
float temperature = dht.readTemperature();

// Leer el valor de la fotoresistencia
int fotoresistorValue = analogRead(fotoresistorPin);
float luminosity = map(fotoresistorValue, 0, 1023, 0, 100);

```

Se imprime los datos por comunicación serial, de todas las variables de temperatura, humedad y luminosidad.

```

// Mostrar los valores leídos en el monitor serial
Serial.print("Temperatura: ");
Serial.print(temperature);
Serial.println(" °C");
Serial.print("Humedad: ");
Serial.print(humidity);
Serial.println(" %");
Serial.print("Luminosidad: ");
Serial.print(luminosity);
Serial.println(" %");

```

Se realiza un condicional para visualizar el indicador de niveles de luminosidad.

```

if (luminosity < 20) {
    digitalWrite(ledLuzBajo, HIGH);
    digitalWrite(ledLuzMedio, LOW);
    digitalWrite(ledLuzAlto, LOW);
} else if (luminosity >= 20 && luminosity < 40) {
    digitalWrite(ledLuzBajo, LOW);
    digitalWrite(ledLuzMedio, HIGH);
    digitalWrite(ledLuzAlto, LOW);
} else if (luminosity >= 50 && luminosity <= 100) {
    digitalWrite(ledLuzBajo, LOW);
    digitalWrite(ledLuzMedio, LOW);
    digitalWrite(ledLuzAlto, HIGH);
}

```

para después enviar los datos como cliente al servidor HTTP, esto para poder visualizarlos y guardarlos en la interfaz y en la base de datos.

```

// Enviar datos al servidor de la base de datos
HTTPClient http;
String url = String(serverNameBD) + "?temperatura=" + String(temperature) + "&humedad=" +
String(humidity) + "&luz=" + String(luminosity);
Serial.print("URL: ");

```

```

Serial.println(url);
http.begin(wifiClient, url);
int httpResponseCode = http.GET();

if (httpResponseCode > 0) {
    String response = http.getString();
    Serial.print("Código de respuesta HTTP BD: ");
    Serial.println(httpResponseCode);
    Serial.println(response);
} else {
    Serial.print("Error al enviar GET BD: ");
    Serial.println(httpResponseCode);
}

```

Ademas se envian los datos para el servidor flask, el condicional nos sirve para verificar la respuesta del servidor HTTP y del servidor Flask, y finalizar el envio, esto para poder correr el script de llamadas de alertas desde el servidor de Asterisk.

```

// Enviar datos al servidor Flask
String urlFlask = String(serverNameFlask) + "?temperatura=" + String(temperature) + "&humedad=" +
String(humidity) + "&luz=" + String(luminosity);
Serial.print("URL Flask: ");
Serial.println(urlFlask);
http.begin(wifiClient, urlFlask);
int httpResponseCodeFlask = http.GET();

if (httpResponseCodeFlask > 0) {
    String response = http.getString();
    Serial.print("Código de respuesta HTTP Flask: ");
    Serial.println(httpResponseCodeFlask);
    Serial.println(response);
} else {
    Serial.print("Error al enviar GET Flask: ");
    Serial.println(httpResponseCodeFlask);
}

http.end();
}

```

Se agrega un delay para tener un rango en el envío de datos y no colapsar el servidor tanto con llamadas, como con datos recolectados.

```
delay(10000);
```

8.7.2. Configuración en Asterisk

Para la configuración dentro del servidor, se crea una extensión dentro de la ruta /etc/asterisk en el archivo de configuración extensions.conf lo que se añade es un contexto (arma-esp) en este contexto se utiliza la función DIAL para poder contestar y enviar las alertas a los usuarios ya agregados dentro de mi servidor mediante llamadas; entonces para aquello se añade -1XX para que llame a la extensión 1 y los demás números sean del 0-9 con el protocolo de sincronización PJSIP y recoja las extensiones de allí de los usuarios registrados, se toma un tiempo de espera y se podrá ingresar número para cancelar la llamada, en el caso de contestar la llamada se manda un audio con la función playback en el caso de contestar la llamada y colgar.

```
[alarma-esp]
;ALARMAS-ESP
exten => _1XX,1,Dial(PJSIP/${EXTEN},30,tT)
    same => n,Playback(vm-goodbye)
    same => n,Hangup()
```

En la extensión default se añade el número 300 o 301 para las llamadas en el caso de no llegar al servidor, se configura lo anterior mencionado de igual manera.

```
;ALARMAS-ESP
exten => _3XX,1,Dial(PJSIP/${EXTEN},30,tT)
    same => n,Playback(vm-goodbye)
    same => n,Hangup()
```

8.7.3. Configuración del script en php para las llamadas

Para el script de llamadas se tiene en cuenta que la ruta en la cuál se crea el archivo temporal donde se registran las llamadas es la ruta /var/spool/asterisk/outgoing/, teniendo en cuenta lo mencionado anteriormente se empieza declarando los parámetros de la llamada que será la alarma; con el cuál va a salir el nombre del usuario, el contexto al cuál se va enviar, con prioridad de llamada 1 para poder empezar la llamada.

```
$callerID = "Alarm System <301>";
$context = "alarma-esp";
$priority = 1;
$timeout = 30;
```

Se escoge el usuario al cuál se va a llamar, el audio el cuál se va reproducir este debe ser en formato gsm o wav; y de lo antes mencionado que el archivo se guardara temporalmente en la ruta TMP.

```
$extension1 = "100";
$audioFile1 = "/var/lib/asterisk/sounds/nivel-alto";
$callFile1 = "/tmp/alarm_bajo.call"; // Archivo temporal para la primera llamada
```

Para el contenido de la llamada se puede añadir aplicaciones, funciones, parámetros necesarios como la prioridad, el caller-id, la extensión, el protocolo de sincronización, el timeout, la data para reproducir el audio, que antes ya habíamos declarado.

```
$content1 = "Channel: PJSIP/$extension1\n";
$content1 .= "Callerid: $callerID\n";
$content1 .= "Context: $context\n";
$content1 .= "Extension: $extension1\n";
$content1 .= "Priority: $priority\n";
$content1 .= "Timeout: $timeout\n";
$content1 .= "Application: Playback\n";
$content1 .= "Data: $audioFile1\n"; // Ruta completa del archivo de audio para niveles bajos
```

Crear el archivo de llamada temporalmente para la primera llamada.

```
file_put_contents($callFile1, $content1);
```

Cambiar permisos del archivo temporal para la primera llamada.

```

chmod($callFile1, 0777);

Entonces se declara los mismos parámetros para realizar otra llamada pero en este caso para llamar al dispositivo móvil, cambiando la extensión a 101, y se utiliza el audio necesario; los audios cambiarán de acuerdo al nivel alto o bajo de los parametros de temperatura, humedad y luminosidad.

// Información para la segunda llamada (Niveles altos)
$extension2 = "101";
$audioFile2 = "/var/lib/asterisk/sounds/nivel-alto";
$callFile2 = "/tmp/alarm_alto.call"; // Archivo temporal para la segunda llamada

$content2 = "Channel: PJSIP/$extension2\n";
$content2 .= "Callerid: $callerID\n";
$content2 .= "Context: $context\n";
$content2 .= "Extension: $extension2\n";
$content2 .= "Priority: $priority\n";
$content2 .= "Timeout: $timeout\n";
$content2 .= "Application: Playback\n";
$content2 .= "Data: $audioFile2\n"; // Ruta completa del archivo de audio para niveles altos

// Crear el archivo de llamada temporalmente para la segunda llamada
file_put_contents($callFile2, $content2);

// Cambiar permisos del archivo temporal para la segunda llamada
chmod($callFile2, 0777);

```

Después los archivos se mueven al directorio de Asterisk para iniciar las llamadas y se los renombra.

```

$finalCallFile1 = "/var/spool/asterisk/outgoing/alarm_bajo.call";
$finalCallFile2 = "/var/spool/asterisk/outgoing/alarm_alto.call";

rename($callFile1, $finalCallFile1);
rename($callFile2, $finalCallFile2);

```

Para enviar el mensaje el cuál se ha realizado la llamada a las extensiones, se utiliza un formato .JSON para que se pueda leer en la interfaz.

```

$messageData = [
    'message' => 'Llamadas a Asterisk por niveles altos de sensores realizadas a las extensiones ' . $extension1 . '
    ' . $extension2 . ' con reproducción de audio'
];

```

Se crea un archivo con formato .JSON del mensaje para poder leerlo en la interfaz, en una ruta específica para además guardarlo.

```

$jsonFilePath = '/var/www/html/message.json';
file_put_contents($jsonFilePath, json_encode($messageData));

```

Para visualizar que esta corriendo el script y se realizaron las llamadas se imprime un mensaje.

```

echo "Llamadas a Asterisk por niveles altos de sensores realizadas a las extensiones $extension1 y
$extension2 con reproducción de audio\n";

```

8.7.4. Configuración servidor Flask

Se utiliza el servidor flask, específicamente por ser un servidor para HTTP, el servidor flask es un “micro” Framework escrito en Python y concebido para facilitar el desarrollo de Aplicaciones Web. Se utilizan las librerías “flask”, REQUEST”, además de paramiko “paramiko”.

```
from flask import Flask, request
import paramiko
```

Se tiene en cuenta, que flask escucha por el puerto 5000 pero asignamos el puerto 80 para las peticiones HTTP que enviará la ESP8266, para recolectar los datos.

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=80)
```

Entonces se empieza creando una app con la extensión /check" para verificar que el servidor esta funcionando correctamente.

```
@app.route('/check')
def check():
    return "Esta funcionando la pagina", 200
```

Para la otra asignación de la app se define algunas variables, en este caso se va a ingresar al servidor mediante el protocolo de SSH para ejecutar algunos comandos para correr el script de php de llamada-bajo.php o llamada-alto.php.

```
VM_HOST = '192.168.220.129'
VM_USERNAME = 'jean'
VM_PASSWORD = '1234'
PHP_SCRIPT_PATH1 = '/home/jean/scripts2php/llamada_alto.php'
PHP_SCRIPT_PATH2 = '/home/jean/scripts2php/llamada_bajo.php'
```

Se definen algunas funciones, estas para acceder por SSH como antes se menciono, con las variables antes declaradas, aquí se utilizaran las librerías PARAMIKO.

```
def ejecutar_script_en_vm_alto():
    try:
        ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        ssh.connect(VM_HOST, username=VM_USERNAME, password=VM_PASSWORD)

        stdin, stdout, stderr = ssh.exec_command(f'php {PHP_SCRIPT_PATH1}')
        salida = stdout.read().decode()
        error = stderr.read().decode()

        ssh.close()
```

Se declara un condicional para verificar si se ejecuto el script, en el caso de no ejecutarse se envía un mensaje de error.

```
if error:
    return f"Error ejecutando el script: {error}"
return salida
except Exception as e:
    return f"Error de conexión SSH: {str(e)}"
```

Igualmente otra función pero para llamar un script diferente ya que se tiene uno para el nivel bajo y otro para el nivel alto de las alertas.

```
def ejecutar_script_en_vm_bajo():
    try:
        ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        ssh.connect(VM_HOST, username=VM_USERNAME, password=VM_PASSWORD)

        stdin, stdout, stderr = ssh.exec_command(f'php {PHP_SCRIPT_PATH2}'
```

```

salida = stdout.read().decode()
error = stderr.read().decode()

ssh.close()

if error:
    return f"Error ejecutando el script: {error}"
return salida
except Exception as e:
    return f"Error de conexión SSH: {str(e)}"
```

Se crea otra app donde llegaran peticiones con el método "GET"de la ESP8266, esto con la extensión /datos definiendo una función dentro de la app recibir -datos.

```
@app.route('/datos', methods=['GET'])
def recibir_datos():
```

Se declara algunas variables de las cuáles se define como argumentos para recoger los datos con método get de la tempertura, humedad y luz, además de añadir otras variables para verificar los niveles altos o bajos de las variables.

```
umbral_temperatura_alto = 38.0
umbral_humedad_alto = 60.0
umbral_luz_alto = 70.0
```

```
umbral_temperatura_bajo = 19.0
umbral_humedad_bajo = 30.0
umbral_luz_bajo = 15.0
```

Se realiza un condicional para llamar a las funciones de acuerdo al nivel alto o bajo de los parámetros de tempertaura, humedad y luminosidad, de forma que se pueda ejecutar el script y se envie un mensaje que se esta ejecutando el script.

```
if temperatura >= umbral_temperatura_alto or humedad >= umbral_humedad_alto
or luz >= umbral_luz_alto:
```

```
    #alarma_alto = True
    resultado = ejecutar_script_en_vm_alto()
    return f"Script de alarma alta ejecutado. Resultado: {resultado}"
```

```
if temperatura <= umbral_temperatura_bajo or humedad <= umbral_humedad_bajo
or luz <= umbral_luz_bajo:
```

```
    #alarma_bajo = True
    resultado = ejecutar_script_en_vm_bajo()
    return f"Script de alarma baja ejecutado. Resultado: {resultado}"
```

8.7.5. Verificación de Resultados

Primero se verifica la conexión de la ESP8266 a la red del AP PIX-LINK, para que nos asigne una dirección IP para conectarnos a la red Wi-Fi; y se pueda empezar a enviar las peticiones.

```
192.168.7.2 - - [24/Jun/2024 22:16:39] "GET /check HTTP/  
1.1" 200 -
```

Figura 41: Petición receptada de la ruta /check en el servidor Flask

```
Connecting to WiFi...  
Connecting to WiFi...  
Connecting to WiFi...  
Connecting to WiFi...  
Connected to WiFi  
IP address:  
192.168.7.5
```

Figura 39: Verificación a la red Wi-Fi al monitor serial

Para despues empezar el envio de las peticiones HTTP al servidor dentro de la MV, para tambien enviar al servidor Flask y para que realice las alertas necesarias de acuerdo a los datos que recibe, el ingreso de datos a la base de datos.

```
Temperatura: 20.70 °C  
Humedad: 54.00 %  
Luminosidad: 47.00 %  
URL: http://192.168.7.3/Alarma_asterisk/insertar_datos.php?temperatura=20.70&humedad=54.00&luz=47.00  
Código de respuesta HTTP BD: 200  
Datos insertados correctamente  
URL Flask: http://192.168.7.2:80/datos?temperatura=20.70&humedad=54.00&luz=47.00  
Código de respuesta HTTP Flask: 200  
Datos recibidos correctamente.
```

Figura 40: Ingreso de los datos, y verificación del envio de datos de la ESP8266 a los servidores flask y HTTP.

Se verifica que el servidor esta funcionando y que se puede acceder a la ruta de la app /check.
Se verifica que llegan los datos al servidor flask en la ruta de la app /datos, viendo las peticiones con el método "GET" que se reciben en el servidor.

```
192.168.7.5 - - [24/Jun/2024 23:00:27] "GET /datos?temperatura=20.90&humedad=54.00&luz=42.00 HTTP/1.1" 200 -  
192.168.7.5 - - [24/Jun/2024 23:00:36] "GET /datos?temperatura=20.90&humedad=54.00&luz=45.00 HTTP/1.1" 200 -  
192.168.7.5 - - [24/Jun/2024 23:00:44] "GET /datos?temperatura=20.90&humedad=54.00&luz=42.00 HTTP/1.1" 200 -  
192.168.7.5 - - [24/Jun/2024 23:00:52] "GET /datos?temperatura=20.90&humedad=54.00&luz=42.00 HTTP/1.1" 200 -  
192.168.7.5 - - [24/Jun/2024 23:01:00] "GET /datos?temperatura=20.90&humedad=54.00&luz=42.00 HTTP/1.1" 200 -
```

Figura 42: Peticiones receptadas de la ruta /datos en el servidor Flask

Así, comienza el envío de alertas cuando se registra niveles altos o bajos de temperatura, humedad o luminosidad, de acuerdo a los umbrales que se ubicaron el servidor flask, de forma que se observara de esta manera.

```

Temperatura: 20.80 °C
Humedad: 54.00 %
Luminosidad: 10.00 %
URL: http://192.168.7.3/Alarma_asterisk/insertar_datos.php?temperatura=20.80&humedad=54.00&luz=10.00
Código de respuesta HTTP BD: 200
Datos insertados correctamente
URL Flask: http://192.168.7.2:80/datos?temperatura=20.80&humedad=54.00&luz=10.00
Código de respuesta HTTP Flask: 200
Script de alarma baja ejecutado. Resultado: Llamadas a Asterisk por niveles bajos de sensores realizac

```

Figura 44: Datos vistos en el monitor serial y el envío de la alerta de niveles bajos y ejecutándose la alarma

```

Temperatura: 20.70 °C
Humedad: 55.00 %
Luminosidad: 97.00 %
URL: http://192.168.7.3/Alarma_asterisk/insertar_datos.php?temperatura=20.70&humedad=55.00&luz=97.00
Código de respuesta HTTP BD: 200
Datos insertados correctamente
URL Flask: http://192.168.7.2:80/datos?temperatura=20.70&humedad=55.00&luz=97.00
Código de respuesta HTTP Flask: 200
Script de alarma alta ejecutado. Resultado: Llamadas a Asterisk por niveles altos de sensores realizac

```

Figura 43: Datos vistos en el monitor serial y el envío de la alerta de niveles altos y ejecutándose la alarma

Se visualizara en el CLI del servidor que se realiza la llamada a los usuarios 101 y 100, esto de acuerdo al nivel alto o bajo

```

*ding/alarm_nivel_bajo.cdr
-- Attempting call on PJSIP/100 for application Playback(/var/lib/asterisk/sounds/Atencion-Niveles-bajo) (Retry 1)
-- Attempting call on PJSIP/101 for application Playback(/var/lib/asterisk/sounds/Atencion-Niveles-bajo) (Retry 1)
-- Called 100
-- Called 101
-- PJSIP/100-0000001c is ringing
-- PJSIP/100-0000001c answered
-- <PJSIP/100-0000001c> Playing '/var/lib/asterisk/sounds/Atencion-Niveles-bajo.gsm' (language 'es')

```

Figura 45: Visualización en el CLI de asterisk las alarmas realizadas para alerta de nivel bajo

```

*ding /var/spool/asterisk/outgoing/alarm_nivel_alto.cdr
-- Attempting call on PJSIP/100 for application Playback(/var/lib/asterisk/sounds/nivel-alto
o) (Retry 1)
-- Attempting call on PJSIP/101 for application Playback(/var/lib/asterisk/sounds/nivel-alto
o) (Retry 1)
-- Called 100
-- Called 101
-- PJSIP/100-00000014 is ringing
-- PJSIP/100-00000014 answered
-- <PJSIP/100-00000014> Playing '/var/lib/asterisk/sounds/nivel-alto.gsm' (language 'es')
-- PJSIP/101-00000015 is ringing
[Jun 24 22:52:55] NOTICE[2629]: pbx_spool.c:463 attempt_thread: Call completed to PJSIP/100
-- PJSIP/101-00000015 answered
-- <PJSIP/101-00000015> Playing '/var/lib/asterisk/sounds/nivel-alto.gsm' (language 'es')
[Jun 24 22:53:16] NOTICE[2630]: pbx_spool.c:463 attempt_thread: Call completed to PJSIP/101
    Resealed contact <101@100-100-7-4-105051> at 2019-06-24 22:53:16Z

```

Figura 46: Visualización en el CLI de asterisk las alarmas realizadas para alerta de nivel alto

Entonces se podrá visualizar que se recepta las llamadas tanto en la aplicación del ordenador MicroSIP, y en la aplicación del móvil Zoiper.

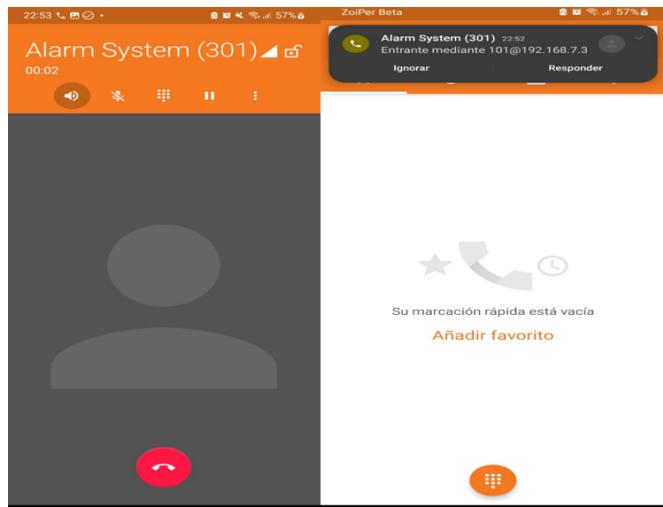


Figura 47: Entrada de alerta en la aplicación móvil Zoiper



Figura 48: Entrada de alerta en la aplicación de MicroSIP

Además se podra verificar con los indicadores Led, los niveles de luminosidad en donde se ubiquen los sensores para gestionar las alarmas contra incendios en el caso se implementar el prototipo.

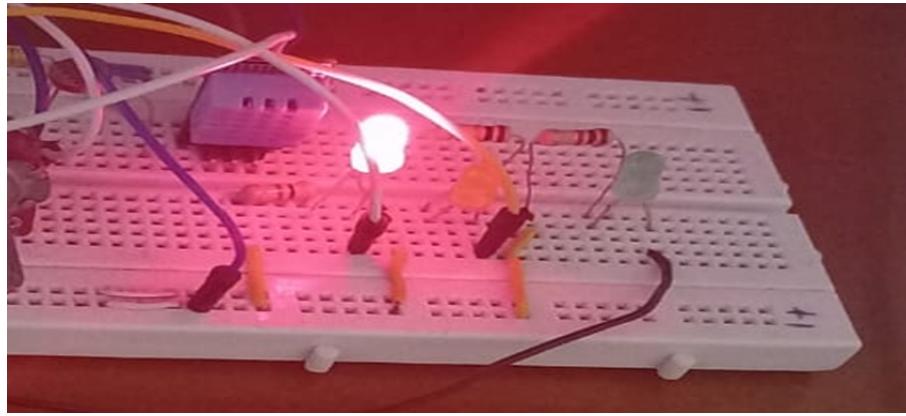


Figura 49: Niveles muy bajo de luminosidad

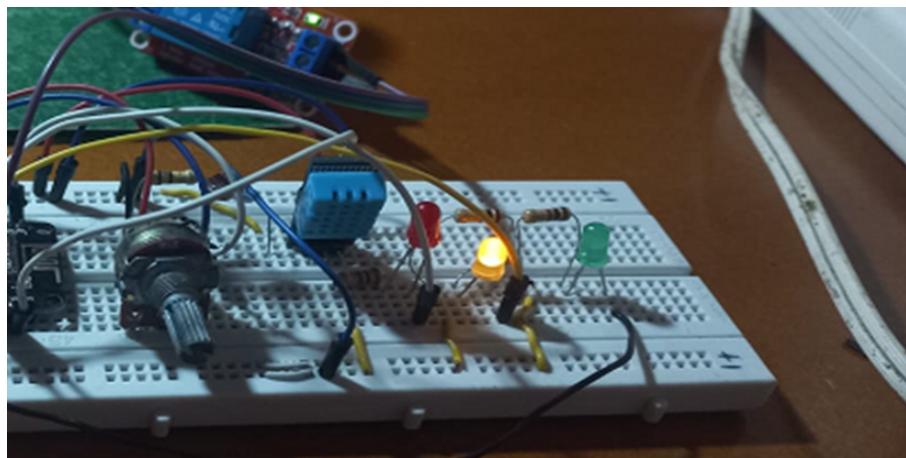


Figura 50: Niveles medio de luminosidad

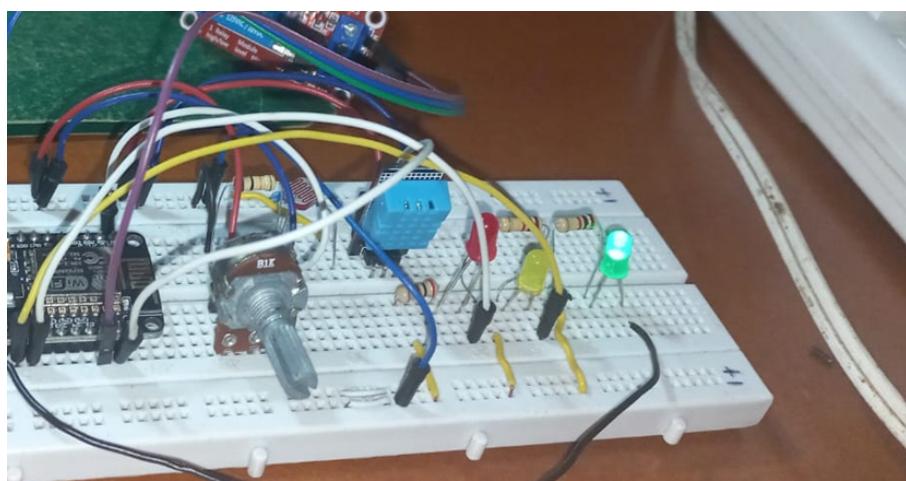


Figura 51: Nivel alto de luminosidad

Para monitorear mediante una interfaz se implemento un dashboard para visualizar los datos, y de acuerdo a indicadores de cada una de las variables de temperatura, humedad y luminosidad, además se verifica si se realizo la llamada al servidor Asterisk, con el mensaje recolectado del script de PHP.

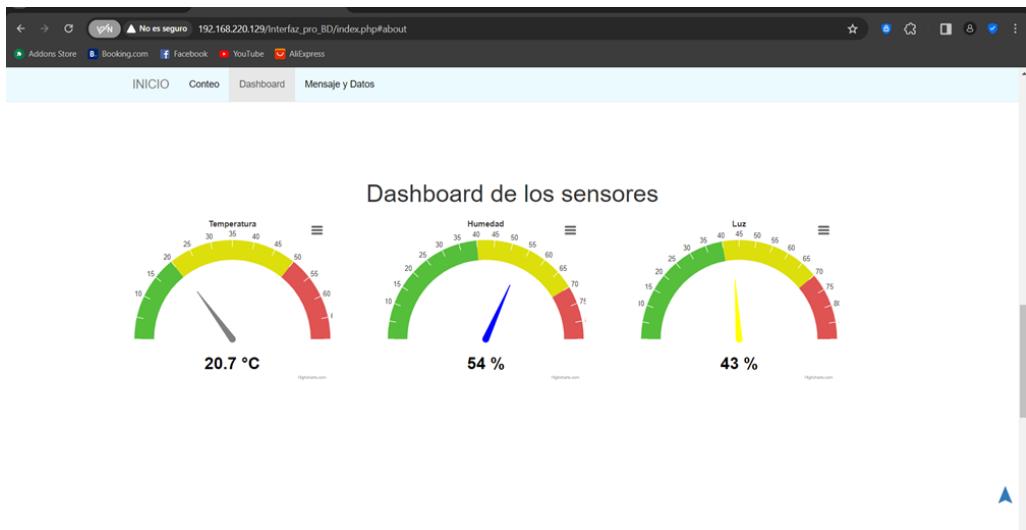


Figura 52: Dashboard de los sensores en la interfaz de la pagina web



Figura 53: Mensaje de confirmación de alerta realizada visualizado en la interfaz de la pagina web

9. RECOMENDACIONES

- Asegurar la correcta instalación y configuración de Asterisk para evitar problemas de conectividad.
- Verificar la compatibilidad de los dispositivos y el software utilizado.
- Realizar pruebas exhaustivas de cada funcionalidad antes de su implementación final.
- Mantener actualizados todos los componentes del sistema para asegurar su seguridad y desempeño óptimo.

10. CONCLUSIONES

El proyecto ha demostrado la versatilidad y eficacia de Asterisk en la implementación de diversas funcionalidades de comunicaciones avanzadas. Desde la interconexión de centralitas hasta el control de dispositivos y la integración de alertas, Asterisk se presenta como una solución robusta y adaptable a múltiples escenarios. La correcta ejecución de cada objetivo específico ha permitido desarrollar un sistema integral que puede ser utilizado en diferentes entornos de telecomunicaciones.

11. BIBLIOGRAFÍA

Referencias

- [1] L. Bryant, J. Madsen. *Asterisk: The Future of Telephony*. O'Reilly Media. McGraw-Hill, 2005.
- [2] R. Doyle. *VoIP System Design*. Artech House. Pearson, 2011.
- [3] Julián Dunayevich. *Asterisk*. Creative Commons, 2005.
- [4] Smith L. Madsen J. Meggelen, J. Van. *Asterisk: The Definitive Guide*. O'Reilly Media. Pearson, 2007.
- [5] Asterisk Project.l. *Asterisk Documentation*. <https://www.asterisk.org/documentation/>, 2002.

12. ANEXOS

```
EXTENSIONES PARA EL PROYECTO - PLAN DE MARCADO:  
1.- Comunicaciones por troncales ---- PC JEAN --- 101,100 ---- PC JOSUE --- 201,200  
2.- telefono ip --- 102 , 202  
3.-Prender el foco ---- 800 OPCIONES: 1.ENCENDIDO , 2.APAGADO  
4.-Consulta del ultimo dato en la base de datos -----900  
OPCIONES: 1.temperatura 2.humedad 3.luz 4.voltaje  
5.-Formulario - ENCUESTA SATISFACCIÓN DE CLASE ----- 850  
OPCIONES: 1.Bueno 2.Regular 3. Malo  
6.- CHAT BOT CON MQTT
```

NOTAS:
COMANDOS DENTRO DE ASTERISK:

```
pjsip show endpoint  
dialplan show  
core set verbose 3  
  
-SIEMPRE PARAR EL FIREWALLD DEL SISTEMA;  
  
systemctl stop firewalld  
  
-PARA CARGAR LOS ARCHIVOS DENTRO DEL CLI SIEMPRE:  
SERVER*CLI> reload  
SERVER*CLI> pjsip reload  
SERVER*CLI> extensions reload  
SERVER*CLI> func_odbc reload
```

```
nano extensions.conf

-----
[from-iax]

exten => _7XXX,1,Dial(IAX2/`serverB_out`/${EXTEN:1},30)
exten => _7XXX,2,Congestion
```

Esta configuración permite que al discar 7XXX en el servidor A, se haga una llamada al servidor B con ese número pero quitándole el prefijo 7.

```
SERVIDOR B:
-.-.-.-.-.-.-.-.-.-.-.-

cd /etc/asterisk
nano iax.conf
```

```
-----
[serverA_in]
type=user
auth=md5
secret=1234
context=from-iax

[serverA_out]
type=peer
host=192.168.7.XX
auth=md5
secret=1234
username=serverB_in
```

```

-----3-PRENDER EL FOCO:
-----CONEXION POR PETICIONES HTTP, MEDIANTE TCP/IP WIFI;
-----CÓDIGO DE ESP8266:
-----#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>

const char* ssid = "PIX";
const char* password = "76697978";

ESP8266WebServer server(80);

const int focoPin = 4; // Pin digital donde está conectado el LED

void setup() {
    Serial.begin(115200);
    WiFi.begin(ssid, password);

    Serial.print("Connecting to WiFi");
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("\nWiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());

    pinMode(focoPin, OUTPUT);
    digitalWrite(focoPin, LOW); // Apaga el FOCO al inicio

    server.on("/encender", []() {
        Serial.println("Received request to /encender");
        digitalWrite(focoPin, HIGH); // HIGH enciende el FOCO
        server.send(200, "text/plain", "FOCO Encendido");
    });

    server.on("/apagar", []() {
        Serial.println("Received request to /apagar");
        digitalWrite(focoPin, LOW); // LOW apaga el FOCO
    });
}

```

```

-----GRABAR UN AUDIO, Y DESPUES ESCUCHARLO LLAMANDO A UNA EXTENSION
-----exten => 808,1,record(audio_esp,gsm,6)

-----exten => 809,1,answer()
-----same => n.playback(audio_esp)
-----same => n.hangup()

-----exten => 808,1,record(esc_op,gsm,3)

-----exten => 809,1,answer()
-----same => n.playback(esc_op)
-----same => n.hangup()

-----exten => 808,1,record(led-on,gsm,3)

-----exten => 809,1,answer()
-----same => n.playback(led-on)
-----same => n.hangup()
-----
```

```

NOTA:
GRABAR AUDIO PARA "OPCION INCORRECTA" EN CASO DE NO TENER, nombre para el archivo: invalido.cs

-----
sudo yum install curl
curl --version

cd /etc/asterisk
nano extensiones.conf

--AGREGAR LA DIRECCIÓN IP DE LA ESP8266, QUE SALE AL CONECTARSE LA RED:
.....
WiFi connected
IP address:
192.168.7.3

ESP-MENU
-----
```

```

;ENCENDER UN FOCO - EXTENSION
exten => 800,1,Answer()
    same => 2.Playback(audio_esp) ; Un mensaje de bienvenida grabado
    same => 3.Read(choice,esc_op,1) ; Lee la elección del usuario (1 o 2)
    same => 4.NoOp(Opción elegida: ${choice})
    same => 5.Gotolf(${choice} = 1?encender:check_2)

exten => 800,6(check_2),Gotolf(${choice} = 2)?apagar:incorrecto

; Encender el LED
exten => 800,n(encender),NoOp(Enviando solicitud para encender el LED)
    same => n.System(curl -X GET http://192.168.7.3/encender)
    same => n.Playback(led-on) ; Un mensaje grabado que diga "LED encendido"
    same => n.Hangup()

; Apagar el LED
exten => 800,n(apagar),NoOp(Enviando solicitud para apagar el LED)
    same => n.System(curl -X GET http://192.168.7.3/apagar)
    same => n.Playback(led-off) ; Un mensaje grabado que diga "LED apagado"
    same => n.Hangup()

; Opción incorrecta
exten => 800,n(incorrecto),NoOp(Opción incorrecta seleccionada)
    same => n.Playback(invalido.cs) ; Un mensaje grabado que diga "Opción incorrecta"
    same => n.Hangup()
```

```

-----4.-CONSULTA DEL ULTIMO DATO DE LA BASE DE DATOS;
-----
VERIFICAR CONEXION CON LA BASE DE DATOS:
isql -v MariaDBcliente

DENTRO DE ASTERISK:
asterisk -v
asterisk -rvvvvv

SERVER*CLI> odbc show all

En el servidor de ALMA Linux:
cd /etc/asterisk

nano func_odbc.conf

-----
[UltimoDato]
dsn=asterisk
readsql=SELECT ${ARG1} FROM sensor ORDER BY id DESC LIMIT 1
-----

nano extensions.conf
```

```

GRABACIONES DE AUDIO:

GRABAR UN AUDIO, Y DESPUES ESCUCHARLO LLAMANDO A UNA EXTENSION:
-----
exten => 808,1,record(welcome_cons.gsm,3)

exten => 809,1,answer()
    same => n.playback(welcome_cons)
    same => n.hangup()

-----
exten => 808,1,record(consulta_UD.gsm,6)

exten => 809,1,answer()
    same => n.playback(consulta_UD)
    same => n.hangup()

-----
exten => 808,1,record(in valido_cs.gsm,3)

exten => 809,1,answer()
    same => n.playback(in valido_cs)
    same => n.hangup()

```

```

AUDIOS PARA TEMPERATURA, HUMEDAD, LUZ Y VOLTAJE:

-----
exten => 808,1,record(temperature-is.gsm,3)

exten => 809,1,answer()
    same => n.playback(temperature-is)
    same => n.hangup()

-----
exten => 808,1,record(.gsm,3)

exten => 809,1,answer()
    same => n.playback(humidity-is)
    same => n.hangup()

-----
exten => 808,1,record(light-is.gsm,3)

exten => 809,1,answer()
    same => n.playback(light-is)
    same => n.hangup()

-----
exten => 808,1,record(voltage-is.gsm,3)

exten => 809,1,answer()
    same => n.playback(voltage-is)
    same => n.hangup()

```

```

EXTENSIONES
PARA CONSULTA ULTIMO DATO

[consultas]

exten => 900.1.Answer()
same => 2.Playback(welcome_cons) ; Un mensaje de bienvenida grabado
same => 4.Read(choice,consulta_UD,1); Lee la elección del usuario (1, 2, 3 o 4) y reproduce un audio
same => 5.NoOp(Opción elegida: ${choice})

same => n.GotoIf(${choice} = 1]?temperatura)
same => n.GotoIf(${choice} = 2]?humedad)
same => n.GotoIf(${choice} = 3]?luz)
same => n.GotoIf(${choice} = 4]?voltaje)
same => n.Playback(valido.cs) ; Si la elección no es válida
same => n.Hangup()

; Obtener el último valor de temperatura y manejar el valor decimal
exten => 900.n(temperatura).NoOp(Obteniendo el último valor de temperatura)
same => n.Set(result=${ODBC_UltimoDato(temperatura)})
same => n.Set(entero=${CUT(result,,1)}) ; Obtener la parte entera
same => n.Set(fraccion=${CUT(result,,2)}) ; Obtener la parte fraccionaria
same => n.Playback(temperature-is) ; Un mensaje grabado que diga "La temperatura en °C es"
same => n.SayNumber(${entero}) ; Decir la parte entera de la temperatura
same => n.SayDigits(${fraccion}) ; Decir la parte fraccionaria de la temperatura
same => n.Hangup()

; Obtener el último valor de humedad
exten => 900.n(humedad).NoOp(Obteniendo el último valor de humedad)
same => n.Set(result=${ODBC_UltimoDato(humedad)})
same => n.Playback(humidity-is) ; Un mensaje grabado que diga "La humedad es"
same => n.SayNumber(${result}) ; Decir el valor de la humedad
same => n.Hangup()

; Obtener el último valor de luz
exten => 900.n(luz).NoOp(Obteniendo el último valor de luz)
same => n.Set(result=${ODBC_UltimoDato(luz)})
same => n.Playback(light-is) ; Un mensaje grabado que diga "La luz es"
same => n.SayNumber(${result}) ; Decir el valor de la luz
same => n.Hangup()

```

5.-Formulario – ENCUESTA SATISFACCIÓN DE CLASE;

- 1.bueno
- 2.regular
- 3.malo

EN LA BASE DE DATOS:

ACCEDER CON LA DIRECCIÓN IP A PHPMYADMIN:
<http://192.168.220.129/phpmyadmin>

Creación en la base de datos, de una nueva tabla:
 crear una tabla de votación:
 dos columnas para el # de cedula y la opción:

- Ncedula
- Op

```

EN EL SERVIDOR ALMA LINUX:

FUNCION Nueva en:
cd /etc/asterisk
nano func_odbc.conf

[Encuesta2]
dsn=asterisk
writesql=UPDATE votacion SET Op='${SQL_ESC(${VAL1})}' WHERE Ncedula='${SQL_ESC(${ARG1})}'

writesql=INSERT INTO `votacion` ('Ncedula', 'Op') VALUES ('1719024256', '1');

-----cd etc/asterisk
-----nano extensions.conf
-----GRABO OTRO AUDIO PARA LA ENCUESTA:
-----exten => 808,1,record(audio_op.gsm,3)
-----exten => 809,1,answer()
-----same => n.playback(audio_op)
-----same => n.hangup()

-----exten => 808,1,record(audio_encuesta.gsm,3)
-----exten => 809,1,answer()
-----same => n.playback(audio_encuesta)
-----same => n.hangup()

-----exten => 808,1,record(pet_ced.gsm,3)
-----exten => 809,1,answer()
-----same => n.playback(pet_ced)
-----same => n.hangup()
-----
```

AÑADIMOS OTRA EXTENSION PARA LA ENCUESTA Y GUARDAR EL DATO:

```

-----exten => 850,1,answer()
-----same => 2,playback(audio_encuesta)
-----same => 3,read(cedula,pet_ced,10)
-----same => n.GotoIf(${LEN(${cedula}) != 10}?invalid_cedula,1)
-----same => n.read(op.audio_op,1)
-----same => n.GotoIf(${op} < 1 | ${op} > 3)?invalid_option,1)
-----same => n.NoOp(Número de cédula: ${cedula}, Opción: ${op})
-----same => n.Set(result=${ODBC_Encuestacurso(${cedula},${op}))}
-----same => n.NoOp(Resultado de ODBC_Encuesta: ${result})
-----same => n.Playback(encuesta_f)
-----same => n.hangup()

-----exten => 850,n(invalid_cedula),playback(invalid_cedula)
-----same => n.hangup()

-----exten => 850,n(invalid_option),playback(invalido_cs)
-----same => n.hangup()
```

```

<!doctype html>
<html lang="en" class="no-js">
<head>
    <meta charset="utf-8">
    <meta http-equiv="x-ua-compatible" content="ie-edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="canonical" href="https://html5-templates.com/" />
    <title>Interfaz-Formulario</title>
    <meta name="description" content="Simplified Bootstrap template with sticky menu">
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <link href="css/sticky-menu.css" rel="stylesheet">
    <style>

        table {
            width: 60%;
            margin-top: 20px;
            border: 1px solid #ccc;
            border-collapse: collapse;
            background-color: #fff;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }

        th, td {
            border: 1px solid #ddd;
            padding: 10px;
            text-align: center;
        }

        th {
            background-color: #f2f2f2;
            color: #333;
        }

        tr:nth-child(even) {
            background-color: #f9f9f9;
        }

    </style>

```

```

.vote-counts {
    display: flex;
    justify-content: space-around;
    width: 80%;
    margin-top: 25px;
}

.vote-counts div {
    background-color: #fff;
    padding: 15px;
    border: 1px solid #ccc;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    text-align: center;
    width: 30%;
}

.vote-counts div span {
    font-size: 20px;
    font-weight: bold;
    color: #333;
}

button {
    padding: 10px 20px;
    font-size: 16px;
    color: #fff;
    background-color: #007bff;
}

```

```

        color: #333;
        line-height: 1.6;
    }

```

```

</style>
</head>
<body id="page-top" data-spy="scroll" data-target=".navbar-fixed-top">
    <nav class="navbar navbar-default navbar-fixed-top" role="navigation">
        <div class="container">
            <div class="navbar-header page-scroll">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-ex1-collapse">
                    <span class="sr-only">Formulario Avanzadas</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand page-scroll" href="#page-top">INICIO</a>
            </div>

            <div class="collapse navbar-collapse navbar-ex1-collapse">
                <ul class="nav navbar-nav">
                    <li class="hidden">
                        <a class="page-scroll" href="#page-top">Welcome</a>
                    </li>
                    <li>
                        <a class="page-scroll" href="#about">Conteo</a>
                    </li>
                    <li>
                        <a class="page-scroll" href="#whatwedo">Dashboard</a>
                    </li>
                    <li>
                        <a class="page-scroll" href="#contact">Mensaje y Datos</a>
                    </li>
                </ul>
            </div> <!-- .navbar-collapse -->
        </div>
    </nav>

```

[View this project on GitHub →](#)

```

function fetchSensorData() {
    fetch('fetch_sensor_data.php')
        .then(response => response.json())
        .then(data => {
            if (data && data.length > 0) {
                const temperatureChart = Highcharts.charts[0];
                const humidityChart = Highcharts.charts[1];
                const lightChart = Highcharts.charts[2];
                const sensorData = data[0]; // Usar el primer conjunto de datos
                const temp = parseFloat(sensorData.temperatura);
                const humidity = parseFloat(sensorData.humedad);
                const light = parseFloat(sensorData.luz);

                if (temperatureChart) {
                    temperatureChart.series[0].points[0].update(temp);
                }
                if (humidityChart) {
                    humidityChart.series[0].points[0].update(humidity);
                }
                if (lightChart) {
                    lightChart.series[0].points[0].update(light);
                }
            } else {
                console.error('No se recibieron datos de fetch_sensor_data.php');
            }
        })
        .catch(error => console.error('Error al obtener datos del sensor:', error));
}

// Gráfico de Temperatura
Highcharts.chart('temperatureContainer', {
    ...
})

```

```

}
// Gráfico de Temperatura
Highcharts.chart('temperatureContainer', {
    chart: {
        type: 'gauge',
        plotBackgroundColor: null,
        plotBackgroundImage: null,
        plotBorderWidth: 0,
        plotShadow: false,
        height: '250px' // Altura del gráfico de temperatura
    },
    title: {
        text: 'Temperatura'
    },
    pane: {
        startAngle: -90,
        endAngle: 90,
        background: null,
        center: ['50%', '75%'],
        size: '150%'
    },
    yAxis: {
        min: 0,
        max: 70,
        tickPixelInterval: 30,
        tickPosition: 'inside',
        tickColor: Highcharts.defaultOptions.chart.backgroundColor || '#FFFFFF',
        tickLength: 10,
        tickWidth: 1,
        minorTickInterval: null,
        labels: {
            distance: 10,
            style: {
                fontSize: '10px'
            }
        }
    }
})

```

```

1  <?php
2  $servername = "localhost";
3  $username = "comunicaciones";
4  $password = "1234";
5  $dbname = "sensores";
6
7  // Crear la conexión
8  $conn = new mysqli($servername, $username, $password, $dbname);
9
10 // Verificar la conexión
11 if ($conn->connect_error) {
12     die("connection failed: " . $conn->connect_error);
13 }
14
15 $sql = "SELECT temperatura, humedad, luz, voltaje timestamp FROM sensor ORDER BY timestamp DESC LIMIT 1";
16 $result = $conn->query($sql);
17
18 $data = array();
19
20 if ($result->num_rows > 0) {
21     // Salida de datos de cada fila
22     while($row = $result->fetch_assoc()) {
23         $data[] = $row;
24     }
25 } else {
26     echo "0 results";
27 }
28 $conn->close();
29
30 echo json_encode($data);
31 ?>
32 |

```

```

<?php
// Establecer los parámetros de conexión
$host = 'localhost';
$usuario = 'comunicaciones';
$contrasena = '1234';
$base_datos = 'sensores';

// Crear una conexión a la base de datos
$bd = new mysqli($host, $usuario, $contrasena, $base_datos);

// Verificar la conexión
if ($bd->connect_error) {
    die("Error en la conexión a la base de datos: " . $bd->connect_error);
}

// Consulta SQL para obtener los últimos cuatro datos de la tabla sensor
$sql = "SELECT id, temperatura, humedad, luz, voltaje FROM sensor ORDER BY id DESC LIMIT 3";

// Ejecutar la consulta
$resultado = $bd->query($sql);

// Verificar si se obtuvieron resultados
if ($resultado && $resultado->num_rows > 0) {
    echo "<div class='last-data-container'>";
    echo "<strong> Últimos Tres datos ingresados en la tabla sensor:</strong>";
    echo "<br>";
    echo "<table style='background-color: white; text-align: center; border-collapse: collapse; width: 50%; margin: 20px auto;'>";
    echo "<thead style='background-color: #246355; border-bottom: solid 5px #0F362D; color: white;'>";
    echo "<tr>";
    echo "<th>ID</th>";
    echo "<th>Temperatura</th>";
    echo "<th>Humedad</th>";
    echo "<th>Luz</th>";
    echo "<th>Voltaje</th>";
    echo "</tr>";
    echo "</thead>";
    echo "<tbody>";
    echo "<tr>";
    echo "<td>" . $fila['id'] . "</td>";
    echo "<td>" . $fila['temperatura'] . "</td>";
    echo "<td>" . $fila['humedad'] . "</td>";
    echo "<td>" . $fila['luz'] . "</td>";
    echo "<td>" . $fila['voltaje'] . "</td>";
    echo "</tr>";
    echo "</tbody>";
    echo "</div>";
    echo "<br>";
    // Botón para actualizar la tabla de manera dinámica
    echo "<button onclick='updateTable()>Actualizar Tabla</button>";
    echo "<br>";
} else {
    echo "No se encontraron datos en la tabla sensor.";
}
// Cerrar la conexión a la base de datos

```

```

m_dvz_02> main.html.php
if ($resultado && $resultado->num_rows > 0) {
    echo "<br>";
    echo "<table style='background-color: white; text-align: center; border-collapse: collapse; width: 50%; margin: 20px auto;'>";
    echo "<thead style='background-color: #246355; border-bottom: solid 5px #0F362D; color: white;'>";
    echo "<tr>";
    echo "<th>ID</th>";
    echo "<th>Temperatura</th>";
    echo "<th>Humedad</th>";
    echo "<th>Luz</th>";
    echo "<th>Voltaje</th>";
    echo "</tr>";
    echo "</thead>";
    echo "<tbody>";
    echo "<tr>";
    echo "<td>" . $fila['id'] . "</td>";
    echo "<td>" . $fila['temperatura'] . "</td>";
    echo "<td>" . $fila['humedad'] . "</td>";
    echo "<td>" . $fila['luz'] . "</td>";
    echo "<td>" . $fila['voltaje'] . "</td>";
    echo "</tr>";
    echo "</tbody>";
    echo "</table>";
    echo "</div>";
    echo "<br>";
    // Botón para actualizar la tabla de manera dinámica
    echo "<button onclick='updateTable()>Actualizar Tabla</button>";
    echo "<br>";
} else {
    echo "No se encontraron datos en la tabla sensor.";
}
// Cerrar la conexión a la base de datos

```

Activar Win
Ve a Configura

```

rma.asterisk > ./servidor_1.py > recibir_datos
1  from flask import Flask, request
2  import paramiko
3
4  app = Flask(__name__)
5
6  # Configura los detalles de la máquina virtual
7  VM_HOST = '192.168.220.129'
8  VM_USERNAME = 'jean'
9  VM_PASSWORD = '1234'
10 PHP_SCRIPT_PATH1 = '/home/jean/scripts2php/llamada_alto.php'
11 PHP_SCRIPT_PATH2 = '/home/jean/scripts2php/llamada_bajo.php'
12
13
14 def ejecutar_script_en_vm_alto():
15     try:
16         ssh = paramiko.SSHClient()
17         ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
18         ssh.connect(VM_HOST, username=VM_USERNAME, password=VM_PASSWORD)
19
20         stdin, stdout, stderr = ssh.exec_command(f'php {PHP_SCRIPT_PATH1}')
21         salida = stdout.read().decode()
22         error = stderr.read().decode()
23
24         ssh.close()
25
26         if error:
27             return f"Error ejecutando el script: {error}"
28         return salida
29     except Exception as e:
30         return f"Error de conexión SSH: {str(e)}"
31
32 def ejecutar_script_en_vm_bajo():
33     try:
34         ssh = paramiko.SSHClient()
35         ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
36         ssh.connect(VM_HOST, username=VM_USERNAME, password=VM_PASSWORD)
37
38         stdin, stdout, stderr = ssh.exec_command(f'php {PHP_SCRIPT_PATH2}')
39         salida = stdout.read().decode()
40         error = stderr.read().decode()
41
42         ssh.close()
43
44         if error:
45             return f"Error ejecutando el script: {error}"
46         return salida
47     except Exception as e:
48         return f"Error de conexión SSH: {str(e)}"
49

```

```

<?php
$servername = "localhost";
$username = "comunicaciones";
$password = "1234";
$dbname = "sensores";

$temperatura = $_GET['temperatura'];
$humedad = $_GET['humedad'];
$luz = $_GET['luz'];

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "INSERT INTO sensor (temperatura, humedad, luz, voltaje) VALUES ('$temperatura', '$humedad', '$luz', '0')";

if ($conn->query($sql) === TRUE) {
    echo "Datos insertados correctamente";
} else {
    echo "Error al insertar datos: " . $conn->error;
}
$conn->close();
?>

```