

Administración de servicios y utilidades web con Raspberry Pi



Autor: Rubén Mellado Culebras

Curso: 20-21

Año: 2021

Ciclo Formativo: ASIR

Nombre del Centro: IES Serpis, Valencia

Tutor Individual: Juan Ignacio Febrero Senar

Propuesta detallada

El proyecto trata de poder tener el control de los diferentes servicios informáticos en Linux de manera remota. Controlar estos desde un dispositivo conectado a Internet, mediante una aplicación de mensajería instantánea como es el caso de Telegram, y usar Python de intérprete para ello.

Así conseguimos que no sea necesario tener que acceder desde un navegador web, para controlar los servicios que se consideren precisos, y aprovechando la versatilidad de uso que esto puede proporcionarnos.

En este caso, se va a hacer uso de un servidor en local, de bajo coste. Para ello, se va a usar el mini ordenador más vendido del mundo (cifras oficiales, 30 millones), el Raspberry Pi, en su versión 3B.

Justificación

El principal problema de la actualidad en el ámbito informático es el gasto que se puede conseguir con un buen sistema informático para administrar tus servicios en red. Si quieres hacerlo, tienes que ir a por un equipo medianamente decente y de las 3 o 4 cifras no baja.

Luego se tienen que administrar licencias, configuración y mantenimiento de software, infraestructura, refrigeración, costes de mantenimiento y limpieza, etc. Al final se acaba perdiendo mucho tiempo y dinero, aun siendo para algo doméstico o un pequeño negocio.

El principal motivo por el que he elegido este proyecto, es por el gran potencial que le veo que tiene este pequeño componente electrónico que está al alcance de cualquiera, y con características que aunque parezcan escasas, para muchos casos son más que suficientes.

En este proyecto vamos a ver alternativas reales a las que existen de pago, para un entorno profesional. Lo mejor de todo, usando Linux, con software libre, y/o gratuito. Aparte de ver los posibles usos que nos puede dar esta bestia, como es el caso de la domótica low-cost gracias a sus puertos de expansión GPIO. Que aun estando presentes y ocupando cierta porción, para muchos pasan desapercibidos o inexistentes, sin saber cuán útiles pueden ser.

Objetivos del proyecto

No depender de grandes conocimientos en desarrollo web para poder gestionar de manera remota lo que pueda suceder en tu servidor local o de empresa. Usando una alternativa real al uso de páginas web o similar para poder administrar tus servicios y red interna de este. Aunque en el proyecto presente se muestra una manera de visualizar servicios virtuales activos al momento, también se puede extrapolar a servicios físicos como pueda ser la domótica.

Administración de servicios y utilidades web con Raspberry Pi

Rubén Mellado Culebras

Índice

1 - ¿Qué es Raspberry Pi?	8
Historia de Raspberry Pi	8
Características de Raspberry Pi	8
Variedad y modelos en el mercado	8
2 - ¿Qué es Telegram?	10
¿Es solo para mensajes de texto?	10
Plataformas soportadas	10
3 - ¿Qué es Python?	11
Historia de Python	11
Plataformas soportadas	11
4 - ¿Qué es Bash?	12
Historia de Bash	12
Plataformas soportadas	12
Intérprete de órdenes	12
5 - Configuración inicial Raspberry Pi	13
Descargar Sistema Operativo	13
Descomprimir ISO	13
Grabar ISO en MicroSD	14
Primer inicio RaspiOS	16
Arrancar desde USB	16
Clonar MicroSD a HDD / SSD por USB	18
6 - Instalación y configuración de Samba	21
Instalar Samba	21
Acceder remotamente	23
7 - Instalación y configuración de SSH	24
Instalación de SSH	24
Conexión remota	24
Claves SSH	24
8 - Instalación y configuración de VoIP / SIP con Asterisk y ZoiPer	27
Introducción a VoIP	27
Asterisk	27
Crear usuarios para VoIP	29

Instalar cliente de VoIP	31
9 - Instalación y configuración de Apache2	34
Instalación de Apache2	34
10 - Instalación y configuración de PHP	35
Instalar PHP	35
Carpeta compartida de PHP	36
Página usando sesiones con PHP	37
11 - Placa PCB para indicar servicios activos	39
Interfaz de GPIO	39
Calcular resistencia	39
Diseño placa PCB	39
12 - Script crear y eliminar servicio LED	41
Contenido archivos de activación de LEDs	41
Servicios con systemd	42
Uso de los servicios de los LEDs	42
13 - Instalar y conectar Python a Telegram	44
Instalar paquetes de Python	44
Crear bot de Telegram	45
Conectar Python a Telegram	46
Activar el bot de Python a Telegram	47
Editar perfil del bot de Telegram	48
14 - Dificultades encontradas y aprendizajes alcanzados	50
Dificultades:	50
Aprendizajes:	50
15 - Experiencia personal	51
16 - Conclusiones finales	52
17 - Dedicatoria o agradecimiento	53
18 - Contenido soporte digital	54
Carpetas	54
Archivos sueltos	54
19 - Software utilizado, uso e instalación	56
Bibliografía y Webgrafía	57

1 - ¿Qué es Raspberry Pi?



Historia de Raspberry Pi

Este mini ordenador, tiene su origen en Reino Unido. Se creó bajo el lema de Open Source. Y su objetivo era que cualquiera pudiese permitirse un ordenador mínimamente modesto, para la casa, y a un muy bajo coste (50€ aproximadamente). Originalmente disponía de salida de vídeo DSI, RCA y HDMI, por lo que permite conectar a prácticamente cualquier televisor o monitor.

Características de Raspberry Pi

El concepto de Raspberry Pi, es un mini ordenador con gran versatilidad. Basado en arquitectura ARM, como la mayoría de los dispositivos móviles de bolsillo. Es decir, no usa un CPU como los que acostumbramos a ver del tipo x86, o x64, típicos en plataformas de PC.

Aparte de esto, tiene un factor de forma como una tarjeta de crédito. Se le conoce como SBC (Single-Board Computer) a este tipo de mini ordenadores que integran la mayoría de componentes. Otra característica, que al ser un equipo que funciona con ARM, no usa un OS de PC tradicional. Por ello se usan otros, como pueda ser: Raspbian (ahora llamado Raspios), Android, Ubuntu, RetroPie, LibreElec, OSMC, y Windows IoT, entre otros.

Variedad y modelos en el mercado

Se puede decir que la más conocida y popular debe de ser la Raspberry Pi. Pero esto no quiere decir que tan solo exista esta marca.

Raspberry Pi factor de forma clásico para escritorio (a fecha de 5-2021):

1A (2011) – BCM2835 (1c, 0'7GHz) – 256MB – SD - 26pin

1B (2012) – 2x USB – RJ-45 – 512MB

1B+ (2012) – 4x USB – MicroSD

2B (2014) – BCM2836 (4c, 0'9GHz) – 1GB – 40pin

3B (2015) – BCM2837 (4c, 1'2GHz) – WiFi + BT – 64bit

3B+ (2018) – BCM2837B0 (4c, 1'4GHz) – 2.4 y 5GHz

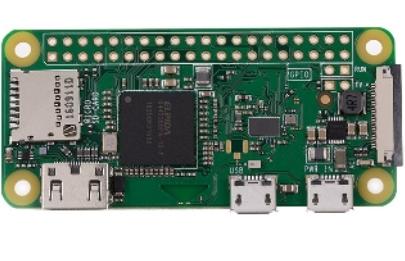
3A+ (2018) – 512MB – BCM2837 – Wi-Fi + BT (no RJ)

4B (2019) – 1 a 8GB BCM2711 (4c, 1'5GHz) – USB 3.0 – 2x microHDMI – máximo 4k 60Hz
– BT 5.0

Zero (2015) – BCM2835 (1c, 1GHz) – 512MB – miniHDMI – 1x Micro USB a USB (OTG) –
pines no soldados - 40pin

Zero W (2017) – BT + WiFi

Zero WH (2017) – BT + WiFi – pines soldados

Modelo A	Modelo B (el del proyecto)	Modelo Zero
		

Algunas otras marcas o modelos populares de SBC que han surgido tras esta, y que siguen de cerca a Raspberry Pi (versión PC):

- ASUS Tinker Board
- nVidia Jetson
- Udoos x86 Ultra
- Latte Panda
- Banana Pi

Algo a destacar muy a favor de la placa SBC elegida para este proyecto, es la gran comunidad que la respalda. La cual ha logrado fortalecer muy rápidamente a la marca y ecosistema de esta.

Ya sea que quieras usarlo como un PC de sobremesa, crear un servidor local, una estación meteorológica, o incluso un equipo de DJ a bajo coste. Se pueden encontrar en Internet muchos tutoriales y consejos de gente que seguro ya ha experimentado con este tipo de cosas antes.

2 - ¿Qué es Telegram?



¿Es solo para mensajes de texto?

Es un recurso software orientado a la mensajería instantánea y VOIP, que requiere del uso de Internet para funcionar. Otra de sus funciones es la de usarlo como almacenamiento de archivos, salas de chats interactivas, chats de difusión, uso de bots para automatizar tareas, pagos, juegos, Inteligencia Artificial, entre otros usos.

Aunque ofrece cierta privacidad y seguridad al usuario, se han conocido casos como el de Estado Islámico en 2014, que filtraron datos de sus acciones. Así que este comenzó a usar mayor seguridad entre emisor y receptor en la codificación de los chats.

A diferencia de su competencia, esta se considera de software libre (su servidor no), ya que permite que se pueda consultar, se pueda modificar y se pueda utilizar su código fuente.

Por otro lado, y en vista de lo anterior, aún hoy día muchos la consideran más insegura, vulnerable y poco fiable en criptografía en los datos enviados por el usuario, respecto a su competencia directa.

Plataformas soportadas

Actualmente dispone de aplicación web y de escritorio disponible para las plataformas de iOS, Android, Linux, MacOS, y Windows, entre otras. Y en enero de 2021, superó los 500 millones de usuarios activos.

3 - ¿Qué es Python?



Historia de Python

Es un lenguaje de programación pautado. Sus principios son la legibilidad y productividad. Con origen en 1991, a día de hoy, sigue recibiendo actualizaciones y gran respaldo por su comunidad de usuarios.

Está basado en licencia de código abierto, llamada Python Software Foundation License.

Plataformas soportadas

Da soporte a plataformas GNU/Linux, MacOS X, y Windows, entre otros.

Otra aplicación es el uso de Micro Python, que se usa en microcontroladores como el Raspberry Pi Pico, Pyboard, STM32, entre otras. Por su versatilidad y poco espacio requerido (mínimo de 256 KB de ROM, y 16KB de RAM) para su funcionamiento.

4 - ¿Qué es Bash?



Historia de Bash

Es un lenguaje secuencial de órdenes al sistema operativo. Está escrito y diseñado para el proyecto GNU para reemplazar el predecesor shell Bourne. Con origen en 1989, actualmente, sigue usándose en la mayoría de distribuciones de Linux, y en MacOS (anteriores a MacOS Catalina).

Plataformas soportadas

A parte de la posibilidad de usarse en sistemas operativos Unix (GNU/Linux y MacOS), también se encuentra disponible para Windows 10, y para Android.

Intérprete de órdenes

Como cualquier intérprete de programación, Bash también usa ciertos tipos de parámetros comunes. Como son las variables del sistema, variables de entorno, bucles for, bucles while, y condicionales if, entre otros.

Con lo anterior ya se puede usar la línea de comandos para consultas, etc. Pero, algo muy útil para automatizar tareas, será el uso de Bash Script, que consiste en lo anterior, guardarlo en archivos (generalmente con .sh) de manera secuencial.

Estos se podrán ejecutar manualmente de manera independiente, cada uno para una función específica. Y si se quiere automatizar más, se puede hacer uso de /etc/rc.local, o de /etc/systemd/system, en el cual asignar un proceso que se realice de manera automática con el sistema.

5 - Configuración inicial Raspberry Pi

Descargar Sistema Operativo

Descargar el OS para Raspberry Pi (ARM).

Raspberry Pi OS with desktop

Release date: January 11th 2021

Kernel version: 5.4

Size: 1,171MB

[Show SHA256 file integrity hash:](#)

[Download](#)

[Download torrent](#)

cb1efa778f3a4effda7bf6f62
2e8e8e779f5303ac77ac8c558
061aece9020fe6

[Release notes](#)

Comprobar que está íntegra la ISO de RaspiOS.

```
ruben@UbuntuPC-MS-7A34:~$ cd Descargas/iso-raspi/  
ruben@UbuntuPC-MS-7A34:~/Descargas/iso-raspi$ sha256sum 2021-01-11-raspios-buster-armhf.zip  
cb1efa778f3a4effda7bf6f622e8e779f5303ac77ac8c558061aece9020fe6 2021-01-11-raspios-buster-armhf.zip  
ruben@UbuntuPC-MS-7A34:~/Descargas/iso-raspi$ # el hash del comando  
ruben@UbuntuPC-MS-7A34:~/Descargas/iso-raspi$ echo "cb1efa778f3a4effda7bf6f622e8e779f5303ac77ac8c55  
8061aece9020fe6" > /tmp/a.txt  
ruben@UbuntuPC-MS-7A34:~/Descargas/iso-raspi$ # el hash de la pagina web  
ruben@UbuntuPC-MS-7A34:~/Descargas/iso-raspi$ echo "cb1efa778f3a4effda7bf6f622e8e779f5303ac77ac8c55  
8061aece9020fe6" > /tmp/b.txt  
ruben@UbuntuPC-MS-7A34:~/Descargas/iso-raspi$  
ruben@UbuntuPC-MS-7A34:~/Descargas/iso-raspi$ # el comparador diff  
ruben@UbuntuPC-MS-7A34:~/Descargas/iso-raspi$ diff /tmp/a.txt /tmp/b.txt  
ruben@UbuntuPC-MS-7A34:~/Descargas/iso-raspi$ ?  
0: orden no encontrada  
ruben@UbuntuPC-MS-7A34:~/Descargas/iso-raspi$
```

~ Código:

```
~$ sha256sum 2021-01-11-raspios-buster-armhf.zip  
~$ echo "el_hash_aqui" > /tmp/a.txt  
~$ echo "el_otro_hash_aqui" > /tmp/b.txt  
~$ diff /tmp/a.txt /tmp/b.txt  
# Si da algo, es distinto al de la web.  
~$ $? # Si da "0", el diff ha ido bien.
```

Descomprimir ISO

Descomprimir la ISO.zip.

```
ruben@UbuntuPC-MS-7A34:~/Descargas/iso-raspi$ ll | grep raspios
-rw-rw-r-- 1 ruben ruben 1227921990 mar 16 18:16 2021-01-11-raspios-buster-armhf.zip
ruben@UbuntuPC-MS-7A34:~/Descargas/iso-raspi$ unzip *.zip
Archive: 2021-01-11-raspios-buster-armhf.zip
  inflating: 2021-01-11-raspios-buster-armhf.img
ruben@UbuntuPC-MS-7A34:~/Descargas/iso-raspi$ ll | grep raspios
-rw-r--r-- 1 ruben ruben 3963617280 ene 11 14:15 2021-01-11-raspios-buster-armhf.img
-rw-rw-r-- 1 ruben ruben 1227921990 mar 16 18:16 2021-01-11-raspios-buster-armhf.zip
ruben@UbuntuPC-MS-7A34:~/Descargas/iso-raspi$ 
```

~ Código:

```
~$ ll | grep raspios
~$ unzip *.zip          # El * es por que solo hay 1 archivo
~$ ll | grep raspios
```

Grabar ISO en MicroSD

Conectar MicroSD al PC y comprobar su ruta “/dev”.

```
ruben@UbuntuPC-MS-7A34:~$ ls /dev/sd
sda  sda1 sda2 sda3 sda4 sda5 sda6 sdb   sdb1  sdb2  sdc
ruben@UbuntuPC-MS-7A34:~$ lsblk | grep sdc
sdc      8:32    1  14,5G  0 disk
ruben@UbuntuPC-MS-7A34:~$ 
```

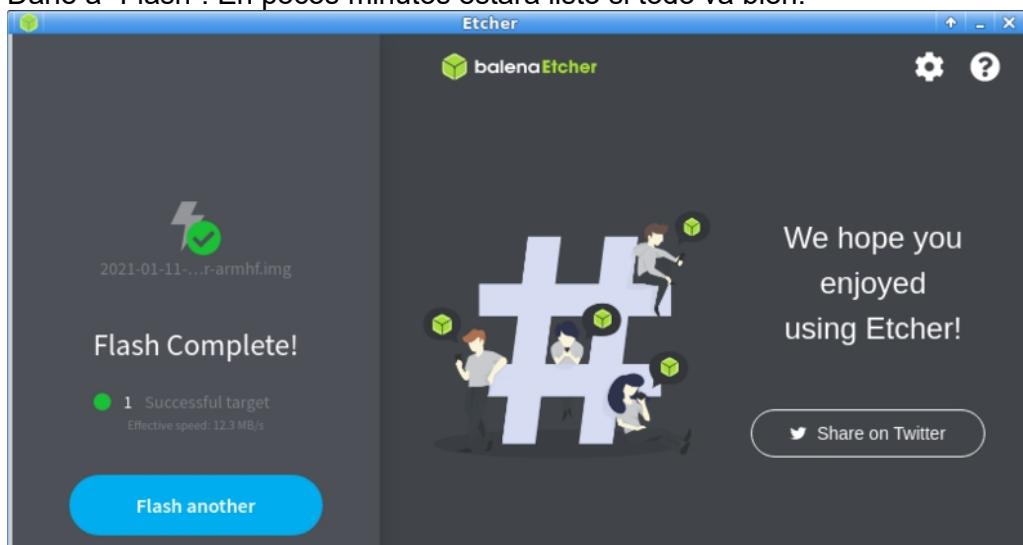
~ Código:

```
~$ ls /dev/sd          # Debería ser el último listado
~$ lsblk | grep sdc    # Con ver los GB se sabe que es la MicroSD
```

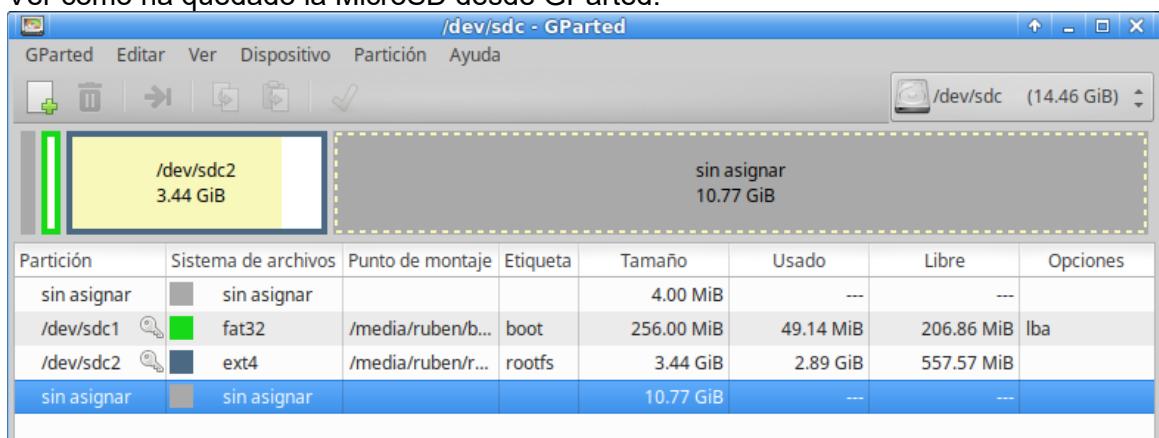
Iniciar el programa para grabar ISOs. Usaremos Balena Etcher.



Darle a “Flash”. En pocos minutos estará listo si todo va bien.



Ver como ha quedado la MicroSD desde GParted.



Ver como ha quedado la MicroSD desde terminal.

```
ruben@UbuntuPC-MS-7A34:~$ lsblk -f | grep sdc
sdc
└─sdc1 vfat      boot    F4F1-BC2C          /media/ruben/boot
└─sdc2 ext4      rootfs  163660a6-ad17-44fc-99c5-5c75e78ad815 /media/ruben/rootfs
ruben@UbuntuPC-MS-7A34:~$
```

~ Código:

```
~$ lsblk -f | grep sdc
```

Primer inicio RaspiOS

Conectar MicroSD y cables a RPi. Seguir los pasos que indica.



Cambiar contraseña que viene por defecto a otra distinta. Esto se hace debido a un ataque informático hace unos años que se aprovechaba de que los usuarios usaban la de por defecto para todo.



Arrancar desde USB

El mini ordenador Raspberry Pi, no tiene BIOS como la de un PC habitual. Para configurar las opciones de esta, es necesario hacerlo mediante un archivo ubicado en la unidad de arranque (MicroSD, HDD, etc). Este paso, es OTP (One Time Programmable). Así que se quedará permanente tras aplicarlo.

Editar archivo de arranque de sistema.

```
pi@raspberrypi:~ $ sudo nano /boot/config.txt  
pi@raspberrypi:~ $ █
```

~ Código:

```
~$ sudo nano /boot/config.txt
```

Añadir línea para arranque por USB.

```
GNU nano 3.2                                /boot/config.txt  
  
# --- para que arranque desde USB ---  
program_usb_boot_mode=1  
  
program_usb_boot_mode=1
```

Para aplicar la nueva configuración. Necesita un reinicio del sistema.

Tras esto, se puede comprobar si se ha aplicado el cambio. Tiene que dar este resultado, si es otro, es posible que no arranque desde USB, o tenga algún problema interno.

```
pi@raspberrypi:~ $ vcgencmd otp_dump | grep 17:  
17:3020000a  
pi@raspberrypi:~ $ █
```

~ Código:

```
~$ vcgencmd otp_dump | grep 17:
```

Tras esto, se puede iniciar desde el SSD por USB.

```
pi@raspberrypi:~ $ lsblk -f  
NAME   FSTYPE LABEL UUID                                     FSavail FSuse% MOUNTPOINT  
sda  
└─sda1  vfat   boot  F4F1-BC2C                           204,4M   19% /boot  
└─sda2  ext4   rootfs 163660a6-ad17-44fc-99c5-5c75e78ad815  99,1G    4% /  
pi@raspberrypi:~ $ ls /dev/sd*  
/dev/sda  /dev/sda1  /dev/sda2  
pi@raspberrypi:~ $ █
```

~ Código:

```
~$ lsblk -f  
~$ ls /dev/sd*
```

Clonar MicroSD a HDD / SSD por USB

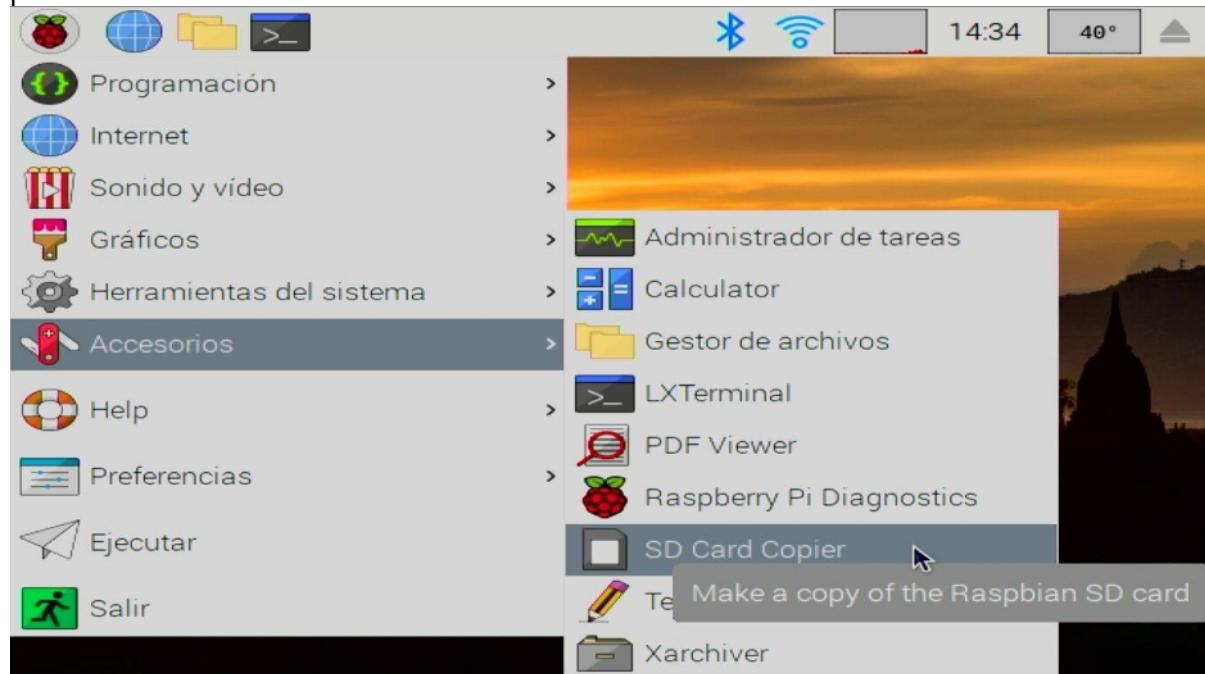
El motivo para hacer esto, aparte de así tener una copia limpia. Es por la velocidad de lectura y escritura de la MicroSD. Actualizar el sistema antes de la clonación.

```
pi@raspberrypi:~ $ sudo apt-get update && sudo apt-get upgrade  
Des:1 http://archive.raspberrypi.org/debian buster InRelease [32,9  
Des:2 http://archive.raspberrypi.org/debian buster/main armhf Packa  
Des:3 http://raspbian.raspberrypi.org/raspbian buster InRelease [15
```

~ Código:

```
~$ sudo apt-get update  
~$ sudo apt-get upgrade
```

Conectar HDD / SSD por USB. Iniciar el programa de SD Card Copier que viene preinstalado en el sistema.



Asegurarse de seleccionar el dispositivo correcto, e iniciar.



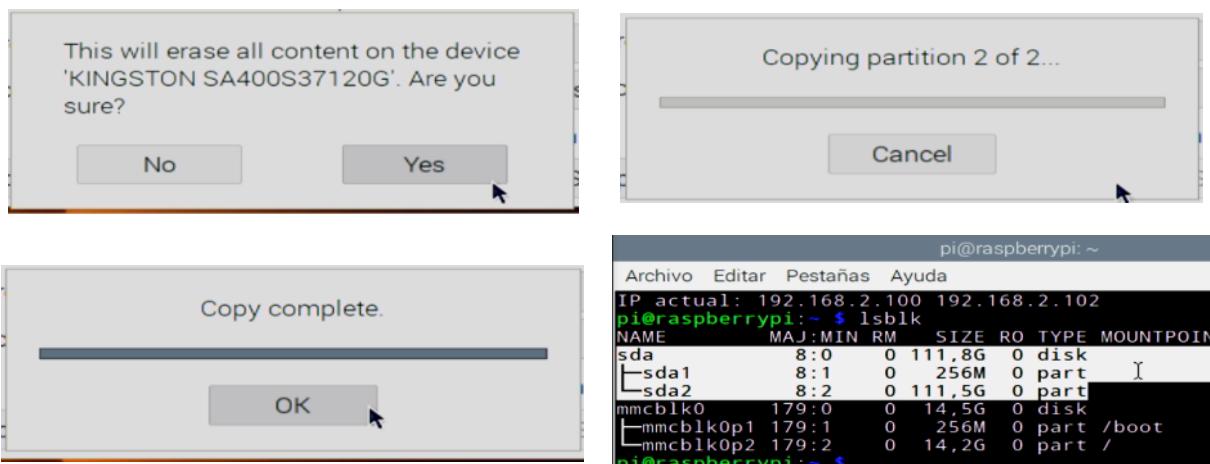
Comprobar que la unidad seleccionada en la ventana anterior, es la que queremos.

```
IP actual: 192.168.2.100 192.168.2.102
pi@raspberrypi:~ $ lsblk
NAME      MAJ:MIN RM    SIZE RO TYPE MOUNTPOINT
sda        8:0    0 111,8G  0 disk 
└─sda1     8:1    0 111,8G  0 part /media/pi/ssd 120gb
mmcblk0   179:0   0 14,5G  0 disk 
└─mmcblk0p1 179:1   0  256M  0 part /boot
└─mmcblk0p2 179:2   0 14,2G  0 part /
pi@raspberrypi:~ $ █
```

~ Código:

~\$ lsblk #Listar las unidades de almacenamiento conectadas a la Raspberry

Seguimos los pasos del asistente de SD Card Copier.



Tras esto, ya está creada la clonación con un nuevo UUID. Falta probar que arranque desde USB, que veremos en el siguiente apartado. Para esto, puede que se necesite actualizar el firmware de la Raspberry Pi.

Prueba de rendimiento lectura / escritura de MicroSD.

```
pi@raspberrypi:~ $ # prueba escritura microsd
pi@raspberrypi:~ $ dd if=/dev/zero of=/tmp/temp.txt bs=1M count=1024 oflag=direct status=progress
1064304640 bytes (1,1 GB, 1015 MiB) copied, 113 s, 9,4 MB/s
1024+0 registros leidos
1024+0 registros escritos
1073741824 bytes (1,1 GB, 1,0 GiB) copied, 113,942 s, 9,4 MB/s
pi@raspberrypi:~ $ dd if=/tmp/temp.txt of=/dev/null bs=1M count=1024 iflag=direct status=progress
1070596096 bytes (1,1 GB, 1021 MiB) copied, 49 s, 21,8 MB/s
1024+0 registros leidos
1024+0 registros escritos
1073741824 bytes (1,1 GB, 1,0 GiB) copied, 49,1438 s, 21,8 MB/s
pi@raspberrypi:~ $ █
```

Prueba de rendimiento lectura / escritura de SSD en USB.

```
pi@raspberrypi:~ $ # prueba escritura SSD USB
pi@raspberrypi:~ $ dd if=/dev/zero of=/tmp/temp.txt bs=1M count=1024 oflag=direct
  status=progress
1048576000 bytes (1,0 GB, 1000 MiB) copied, 32 s, 32,7 MB/s
1024+0 registros leídos
1024+0 registros escritos
1073741824 bytes (1,1 GB, 1,0 GiB) copied, 32,7601 s, 32,8 MB/s
pi@raspberrypi:~ $ # prueba lectura SSD USB
pi@raspberrypi:~ $ dd if=/tmp/temp.txt of=/dev/null bs=1M count=1024 iflag=direct
  status=progress
1042284544 bytes (1,0 GB, 994 MiB) copied, 30 s, 34,7 MB/s
1024+0 registros leídos
1024+0 registros escritos
1073741824 bytes (1,1 GB, 1,0 GiB) copied, 30,8743 s, 34,8 MB/s
pi@raspberrypi:~ $ █
```

Tabla comparativa rendimiento:

	Lectura	Escritura
Micro SD	21,8 MB/s	9,4 MB/s
USB SSD / HDD externo	34,8 MB/s	32,8 MB/s

~ Código:

```
# Escritura
~$ dd if=/dev/zero of=/tmp/temp.txt bs=1M count=1024 oflag=direct status=progress

# Lectura
~$ dd if=/tmp/temp.txt of=/dev/null bs=1M count=1024 iflag=direct status=progress
```

6 - Instalación y configuración de Samba

Instalar Samba

Actualizar la base de datos de actualizaciones, e instalar Samba.

```
IP actual: 192.168.2.101
pi@raspberrypi:~ $ sudo apt-get update > /dev/null
pi@raspberrypi:~ $ sudo apt-get install samba -y
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Paquetes sugeridos:
  bind9 bind9utils ctdb ldb-tools ntp | chrony smbdap-tools ufw winbind
Se instalarán los siguientes paquetes NUEVOS:
  samba
```

~ Código:

```
~$ sudo apt-get update > /dev/null    # /dev/null para que no salga por pantalla
~$ sudo apt-get install samba -y      # -y para que no pida confirmación
```

Comprobar que funciona.

```
pi@raspberrypi:~ $ systemctl status smbd | head -3
● smbd.service - Samba SMB Daemon
  Loaded: loaded (/lib/systemd/system/smbd.service; enabled; vendor preset: enabled)
  Active: active (running) since Sat 2021-05-22 15:11:55 CEST; 38s ago
    Main PID: 1144 (smbd)
   CGroup: /system.slice/smbd.service
           └─1144 /usr/sbin/smbd --foreground
```

~ Código:

```
~$ systemctl status smbd | head -3
```

Crear la carpeta que queremos compartir. Opcional, podemos meter algún archivo para luego comprobar que sí se puede leer desde el dispositivo remoto.

```
pi@raspberrypi:~ $ sudo mkdir compartida_RPi
pi@raspberrypi:~ $ sudo chmod 777 /home/pi/compartida_RPi
pi@raspberrypi:~ $ ls -l | grep compartida_RPi
drwxrwxrwx 2 root root 4096 may 22 14:14 compartida_RPi
pi@raspberrypi:~ $
```

~ Código:

```
~$ sudo mkdir compartida_RPi
~$ sudo chmod 777 compartida_RPi
```

```
~$ ls -l | grep compartida_RPi
```

Cambiar configuración de archivo de Samba para añadir la carpeta que hemos creado.

```
IP actual: 192.168.2.101  
pi@raspberrypi:~ $ ls /etc/samba/  
gdbcommands smb.conf tls  
pi@raspberrypi:~ $ sudo nano /etc/samba/smb.conf #Configurar Samba  
pi@raspberrypi:~ $ cat /etc/samba/smb.conf | tail -8  
[compartida_RPi]  
    comment = Carpeta compartida Raspberry Pi  
    path = /home/pi/compartida_RPi  
    browseable = yes  
    writable = yes  
    guest ok = yes  
    create mask = 0666  
    directory mask = 0766  
pi@raspberrypi:~ $ █
```

~ Código:

```
~$ ls /etc/samba/  
~$ sudo nano /etc/samba/smb.conf # Configurar Samba  
~$ cat /etc/samba/smb.conf | tail -8
```

Añadir un usuario para acceder con él remotamente.

```
pi@raspberrypi:~ $ sudo useradd pi-samba  
pi@raspberrypi:~ $ sudo passwd pi-samba  
Nueva contraseña:  
Vuelva a escribir la nueva contraseña:  
passwd: contraseña actualizada correctamente  
pi@raspberrypi:~ $ sudo smbpasswd -a pi-samba  
New SMB password:  
Retype new SMB password:  
pi@raspberrypi:~ $ █
```

~ Código:

```
~$ sudo useradd pi-samba  
~$ sudo passwd pi-samba # En este caso, usr = pwd  
~$ sudo smbpasswd -a pi-samba # Esto añade un usuario a samba
```

Recargar el servicio de Samba.

```
IP actual: 192.168.2.101
pi@raspberrypi:~ $ sudo systemctl reload smbd
pi@raspberrypi:~ $ smbclient -L 192.168.2.101
Unable to initialize messaging context
Enter WORKGROUP\pi's password:
      Sharename      Type      Comment
      -----      ----      -----
      print$        Disk      Printer Drivers
      compartida_RPi  Disk      Carpeta compartida Raspberry Pi
      IPC$          IPC       IPC Service (Samba 4.9.5-Debian)
Reconnecting with SMB1 for workgroup listing.

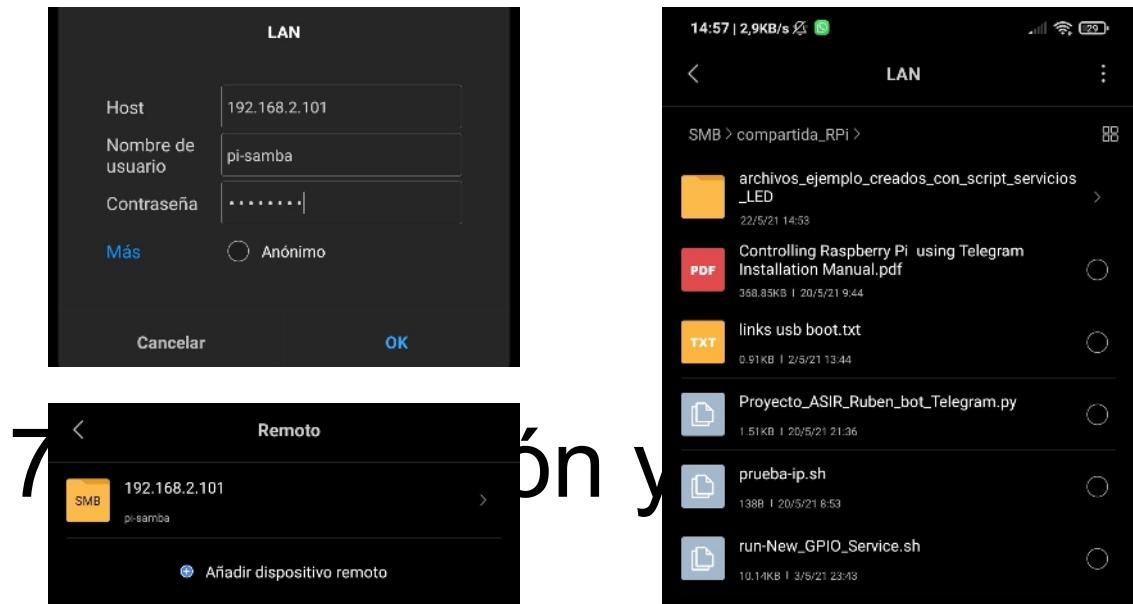
      Server          Comment
      -----
      Workgroup        Master
      -----
      WORKGROUP
pi@raspberrypi:~ $
```

~ Código:

```
~$ sudo systemctl reload smbs
~$ smbclient -L 192.168.2.101      # Pide contraseña, pero no es necesaria
```

Acceder remotamente

Acceder desde otro dispositivo en la misma red.



7. Conexión y configuración de SSH

Instalación de SSH

Ya viene instalado el SSH en la Raspberry Pi. De esta manera podemos instalar el SSH.

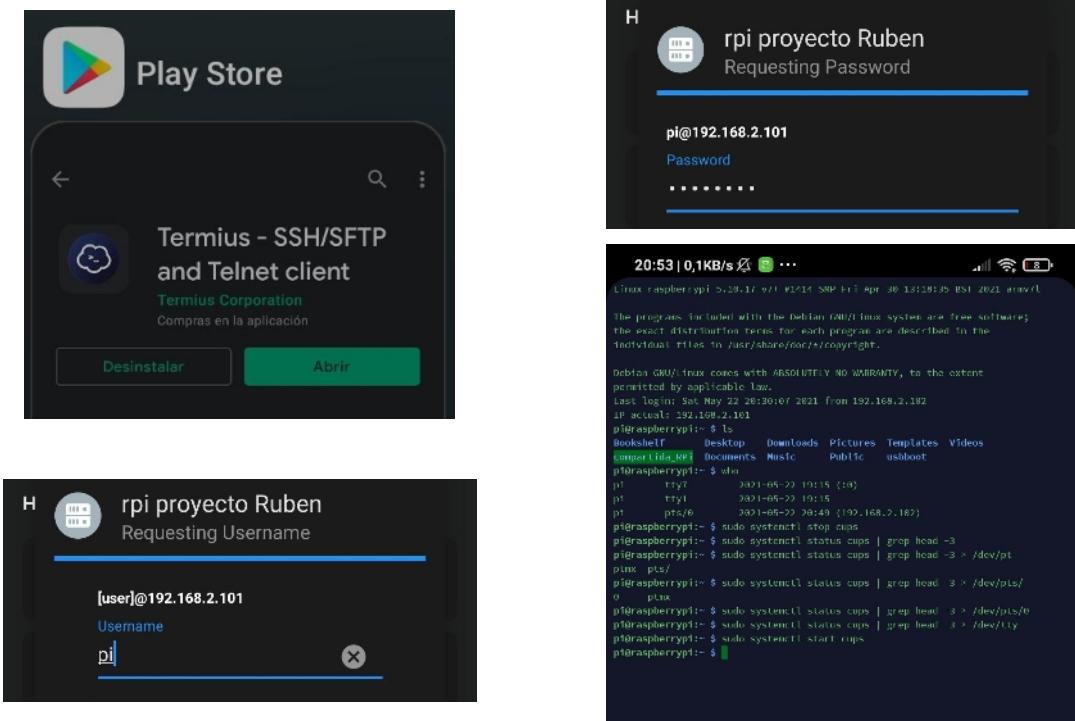
```
pi@raspberrypi: ~
Archivo Editar Pestañas Ayuda
IP actual: 192.168.2.101
pi@raspberrypi: ~ $ sudo apt-get update > /dev/null
pi@raspberrypi: ~ $ sudo apt-get install openssh-server
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
openssh-server ya está en su versión más reciente (1:7.9p1-10+deb10u2+rpt1).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 2 no actualizados.
pi@raspberrypi: ~ $
```

~ Código:

```
~$ sudo apt-get update > /dev/null
~$ sudo apt-get install openssh-server
```

Conexión remota

Probamos a entrar desde otro dispositivo de la misma red.



Claves SSH

Hay varias opciones a elegir: dsa, ecdsa, ed25519, rsa, etc.

```
ruben@UbuntuPC-MS-7A34:~$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/ruben/.ssh/id_dsa):
Created directory '/home/ruben/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ruben/.ssh/id_dsa.
Your public key has been saved in /home/ruben/.ssh/id_dsa.pub.
The key fingerprint is:
```

~ Código:

~\$ ssh-keygen -t dsa

Copiar la clave generada del cliente al servidor.

```
ruben@UbuntuPC-MS-7A34:~$ ssh-copy-id -i .ssh/id_dsa.pub pi@192.168.2.101
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: ".ssh/id_dsa.pub"
The authenticity of host '192.168.2.101 (192.168.2.101)' can't be established.
ECDSA key fingerprint is SHA256:MHAxSxfFCiuf2qQXI6yQVlnjLy3nGTMxtKBJAVUYm9E.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
pi@192.168.2.101's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'pi@192.168.2.101'"
and check to make sure that only the key(s) you wanted were added.

ruben@UbuntuPC-MS-7A34:~$
```

~ Código:

```
~$ ssh-copy-id .ssh/id_dsa.pub pi@192.168.2.101
```

Acceder con claves públicas.

```
ruben@UbuntuPC-MS-7A34:~$ ssh -i .ssh/id_dsa.pub pi@192.168.2.101
pi@192.168.2.101's password:
Linux raspberrypi 5.10.17-v7+ #1414 SMP Fri Apr 30 13:18:35 BST 2021 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat May 22 23:17:04 2021 from 192.168.2.100
IP actual: 192.168.2.101
pi@raspberrypi:~ $
```

~ Código:

```
~$ ssh -i .ssh/id_dsa.pub pi@192.168.2.101
```

Poder lanzar aplicaciones del servidor, desde el cliente. Este paso realizarlo en los dos.

```
IP actual: 192.168.2.101
pi@raspberrypi:~ $ sudo nano /etc/ssh/sshd_config
pi@raspberrypi:~ $ sudo systemctl restart ssh
pi@raspberrypi:~ $ sudo systemctl status ssh.service | head -3
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2021-05-22 23:07:30 CEST; 24s ago
pi@raspberrypi:~ $
```

~ Código:

```
~$ sudo nano /etc/ssh/sshd_config    # En este modificar lo de la siguiente imagen.
~$ sudo systemctl restart ssh
~$ sudo systemctl status ssh.service | head -3
```

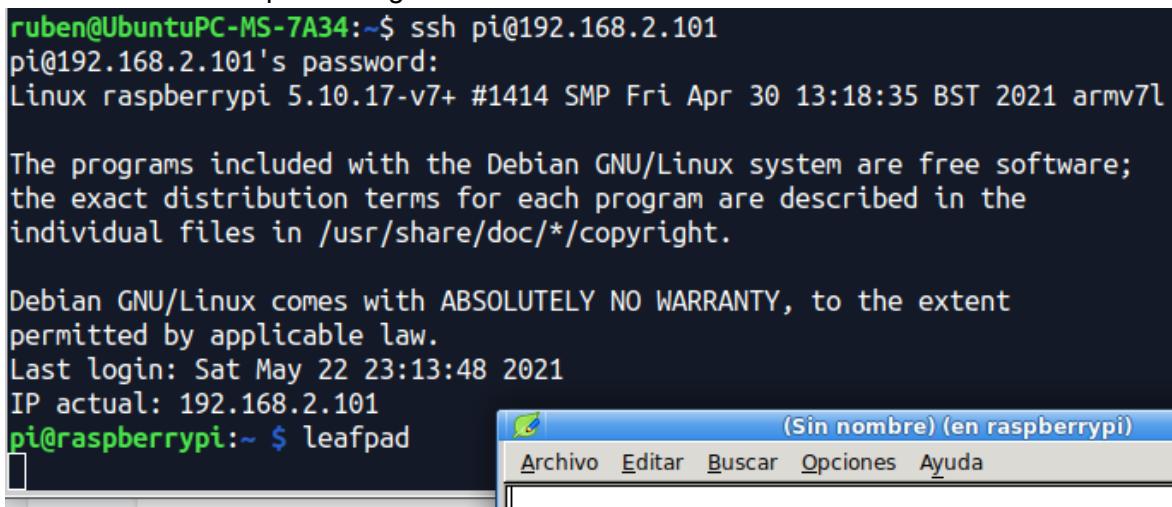
Cambiar estas 2 opciones de “no” a “yes”



```
GNU nano 3.2          /etc/ssh/sshd_config          Modificado
# ForwardAgent no
# ForwardX11 no
ForwardAgent yes
ForwardX11 yes
# ForwardX11Trusted yes
# PasswordAuthentication yes
# HostbasedAuthentication no

^G Ver ayuda ^O Guardar ^W Buscar ^K Cortar txt^J Justificar^C Posición
^X Salir ^R Leer fich.^N Reemplazar^U Pegar txt ^I Ortografía^L Ir a línea
```

Probar a abrir una aplicación gráfica remotamente.



```
ruben@UbuntuPC-MS-7A34:~$ ssh pi@192.168.2.101
pi@192.168.2.101's password:
Linux raspberrypi 5.10.17-v7+ #1414 SMP Fri Apr 30 13:18:35 BST 2021 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat May 22 23:13:48 2021
IP actual: 192.168.2.101
pi@raspberrypi:~ $ leafpad
```

~ Código:

```
~$ ssh pi@192.168.2.101
~$ leafpad
```

8 - Instalación y configuración de VoIP / SIP con Asterisk y ZoiPer

Introducción a VoIP

El objetivo de este software es poder tener conversaciones de telefonía mediante la red. Actualmente en muchas empresas, y cada vez más en particulares, se está implementando este tipo de sistemas, ya que reducen costes, se pueden configurar, son más eficientes, facilita portabilidad, y son fáciles de instalar, monitorear y actualizar, entre otras ventajas.

Asterisk

Instalar Asterisk y comprobar la versión actual.

```
pi@raspberrypi:~ $ sudo apt-get update > /dev/null
pi@raspberrypi:~ $ sudo apt-get install asterisk
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  asterisk-config asterisk-core-sounds-en asterisk-core-sounds-en-gsm
pi@raspberrypi:~ $ asterisk -V
Asterisk 16.2.1~dfsg-1+deb10u2
pi@raspberrypi:~ $
```

~ Código:

```
~$ sudo apt-get update > /dev/null
~$ sudo apt-get install asterisk
~$ asterisk -v
```

Por defecto viene en inglés. Vamos a descargar y cambiar el idioma a español.

```
pi@raspberrypi:~ $ apt-cache search asterisk | grep "es-\\|Spanish"
asterisk-core-sounds-es - asterisk PBX sound files - Spanish
asterisk-core-sounds-es-g722 - asterisk PBX sound files - es-mx/g722
asterisk-core-sounds-es-gsm - asterisk PBX sound files - es-mx/gsm
asterisk-core-sounds-es-wav - asterisk PBX sound files - es-mx/wav
asterisk-prompt-es-co - Colombian Spanish voice prompts for Asterisk
asterisk-prompt-es - Spanish prompts for the Asterisk PBX
pi@raspberrypi:~ $
```



```
pi@raspberrypi:~ $ sudo apt-get install asterisk-core-sounds-es asterisk-core-sounds-es-g722 asterisk-core-sounds-es-gsm asterisk-core-sounds-es-wav asterisk-prompt-es
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes NUEVOS:
```

~ Código:

```
~$ apt-cache search asterisk | grep "es-\\|Spanish" # Para que liste solo los de español.
~$ sudo apt-get install asterisk-core-sounds-es asterisk-core-sounds-es-g722 asterisk-core-sounds-es-gsm asterisk-core-sounds-es-wav asterisk-prompt-es
```

Comprobar los paquetes instalados, en este caso, los que estén en español.

```
pi@raspberrypi:~ $ sudo dpkg -l asterisk* | grep '\\-es'
ii  asterisk-core-sounds-es                      1.6.1-1
ii  asterisk-core-sounds-es-g722                  1.6.1-1
ii  asterisk-core-sounds-es-gsm                  1.6.1-1
ii  asterisk-core-sounds-es-wav                  1.6.1-1
ii  asterisk-prompt-es                         1.4-1
un  asterisk-prompt-es-mx                     <ninguna>
pi@raspberrypi:~ $
```

~ Código:

```
~$ sudo dpkg -l asterisk* | grep "\-es"
```

Ver audios que tiene por defecto para el uso de "Playback()". Y descripción de los audios.

```
pi@raspberrypi:~ $ ls /usr/share/asterisk/sounds/es/
agent-alreadyon.gsm    privacy-unident.gsm      vm-mailboxfull.gsm
agent-incorrect.gsm    priv-callee-options.gsm   vm-message.gsm
agent-loggedoff.gsm    priv-callpending.gsm    vm-messages.gsm
agent-loginok.gsm      priv-introsaved.gsm    vm-minutes.gsm
agent-newlocation.gsm  priv-recordintro.gsm   vm-mismatch.gsm
```

~ Código:

```
~$ ls /usr/share/asterisk/sounds/es/
```

```
pi@raspberrypi:~ $ cat /usr/share/asterisk/sounds/es/core-sounds-es.txt
; Core Asterisk Sounds in Spanish

agent-alreadyon: Ese agente ya ha sido autenticado. Por favor ingrese su numero de agente seguido por la tecla de numero.
agent-incorrect: Clave incorrecta. Por favor, ingrese su numero de agente seguido por la tecla de numero
agent-loggedoff: Agente desconectado
agent-loginok: Agente conectado
agent-newlocation: Por favor ingrese una nueva extension seguida por la tecla de numero
agent-pass: Por favor ingrese su contraseña seguida por la tecla de numero
agent-user: Autenticacion de agente. Por favor ingrese su numero de agente seguido por la tecla de numero
auth-incorrect: Contraseña incorrecta. Por favor ingrese su contraseña seguida por la tecla de numero
auth-thankyou: Gracias
beep: [tono simple]
beeperr: [ tono de error ]
conf-adminmenu: Por favor marque 1 para habilitar o deshabilitar el modo mudo, 2 para bloquear o desbloquear la conferencia, 3 para expulsar al ultimo usuario, 4 o 6 para aumentar o disminuir el volumen de la conferencia, 7 o 9 para el volumen suyo, u 8 para salir.
```

~ Código:

```
~$ cat /usr/share/asterisk/sounds/es/core-sounds-es.txt
```

Archivos de configuración. Antes de modificar, hacer una copia de respaldo.

```
ls /etc/asterisk/
```

```
/etc/asterisk/cel.conf
```

```
/etc/asterisk/cdr.conf
```

Crear usuarios para VoIP

Hacemos una copia del archivo.

```
pi@raspberrypi:~ $ sudo cp /etc/asterisk/sip.conf /etc/asterisk/sip.conf.bu
```

~ Código:

```
~$ sudo cp /etc/asterisk/sip.conf /etc/asterisk/sip.conf.bu
```

Eliminamos líneas vacías y comentadas, para que visualmente sea más agradable.

```
pi@raspberrypi:~ $ sudo vi /etc/asterisk/sip.conf
pi@raspberrypi:~ $ ls -l /etc/asterisk/sip*
-rw-r----- 1 asterisk asterisk 1324 may 23 20:47 /etc/asterisk/sip.conf
-rw-r----- 1 root      root     95096 may 23 20:32 /etc/asterisk/sip.conf.bu
-rw-r----- 1 asterisk asterisk   790 ago 27 2020 /etc/asterisk/sip_notify.conf
pi@raspberrypi:~ $
```

~ Código:

```
~$ sudo vi /etc/asterisk/sip.conf
:g/^s*/d      # Esto borra las líneas comentadas, que empiezan por ";".
:g/^$/d      # Esto borra las líneas en blanco o vacías.
:wq!          # Esto sobreescribe o guarda y sale del archivo.
~$ ls -l /etc/asterisk/sip.conf # Podemos ver la diferencia de tamaño del archivo.
```

Editamos el archivo sip.conf.

```
pi@raspberrypi:~ $ sudo nano /etc/asterisk/sip.conf
pi@raspberrypi:~ $
```

~ Código:

```
~$ sudo nano /etc/asterisk/sip.conf
```

Le añadimos estas líneas en el apartado general.

```
[general]
context=public           ; Default context for incoming calls. D
allowoverlap=no          ; Disable overlap dialing support. (Def
udpbindaddr=0.0.0.0      ; IP address to bind UDP listen socket
tcpenable=no              ; Enable server for incoming TCP connec
tcpbindaddr=0.0.0.0      ; IP address for TCP server to bind to
transport=udp            ; Set the default transports. The orde
srvlookup=yes            ; Enable DNS SRV lookups on outbound ca

qualify=yes    ; monitoreas conexion de VoIP
language=es     ; idioma por defecto
disallow=all    ; desactivar todos los codificadores
allow=ulaw      ; permitir codificadores segun preferencia
```

Y creamos las secciones de los usuarios.

```
GNU nano 3.2                               /etc/asterisk/sip.conf

[usuario](!)          ; creamos una plantilla
    type=friend      ; user -> entrante
    ;peer -> saliente
    ;friend -> ambas
    host=dynamic
    context=ProyectoASIR ; definir tambien en extensions.conf

;Extension jefe
[ext101](usuario)           ; el parentesis es para usar plantilla de arriba
username=Ruben               ; usuario en la extension jefe
secret=1234                  ; contraseña del usuario
;port=5061                   ; servidor usa el 5060, usar distinto para pruebas desde este

;Extension becario
[ext102](usuario)
username=chicoPracticas
secret=1234
;port=5061

;Extension recepcionista
[ext103](usuario)
username=Gertrudis
secret=1234
;port=5061
```

~ Código:

Ejemplo configurar un usuario (sin la plantilla “usuario”):

```
;Extension jefe
[ext101]
type=friend
host=dynamic
context=ProyectoASIR
username=Ruben
secret=1234
;port=5061
```

Editar archivo de extensiones, es decir, la redirección de las llamadas.

```
GNU nano 3.2                               /etc/asterisk/extensions.conf

[ProyectoASIR]
;jefe
exten => 101,1,Dial(SIP/ext101)

;becario
exten => 102,1,Dial(SIP/ext102)

;repcionista
exten => 103,1,Playback(demo-congrats) ; no necesario poner ruta ni extension
exten => 103,2,Dial(SIP/ext103)
exten => 103,3,Hangup()

;exten -> nº extension, prioridad, aplicacion que ejecuta
;           num emisor   orden   la que conecta dispositivos
;Dial -> Hacer llamada
;Playback -> Usa la ruta del idioma (es) con audios
;Hangup -> Cuelga la llamada
```

~ Código:

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

Podemos ver en el ejemplo de arriba, que con tan solo añadir las líneas:

[ProyectoASIR]

exten => 101,1,Dial(SIP/ext101)

Es suficiente para poder realizar llamadas.

En resumen, los archivos usados y para qué:

/etc/asterisk/sip.conf # añadir usuarios (entrada/salida).

/etc/asterisk/extensions.conf # decir donde redirige la llamada, Dialplan, y playback.

Vamos a ver el servicio de asterisk, desde la consola interna que tiene.

```
pi@raspberrypi:~ $ sudo asterisk -rvvv
Asterisk 16.2.1~dfsg-1+deb10u2, Copyright (C) 1999 - 2018, Digium, Inc. 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 16.2.1~dfsg-1+deb10u2 currently running on raspberrypi (pid = 475)
raspberrypi*CLI> sip reload
Reloading SIP
raspberrypi*CLI>
```

~ Código:

```
~$ sudo asterisk -rvvv # Cuantas más "v", más información suele dar
```

Otros comandos útiles dentro de la consola de asterisk:

sip show users # lista los usuarios, nombre usuario, contraseña, grupo, etc.

sip show peers # lista los usuarios conectados

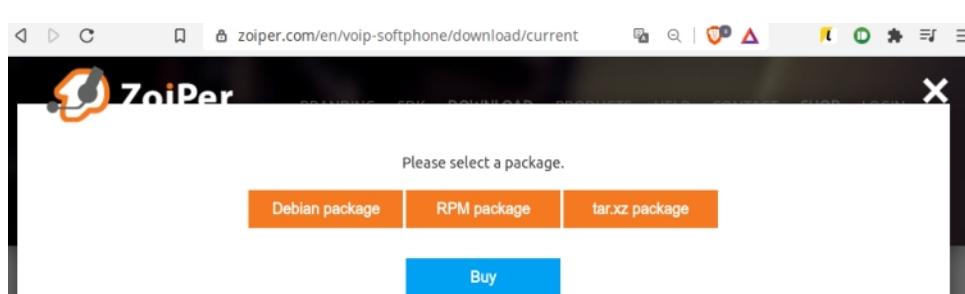
sip reload # cada usuario añadido, se puede actualizar desde consola asterisk

dialplan reload # actualiza si añade extensiones etc

core show channels # ver canales de conexión

sip show channels # ver canales de conexión

Instalar cliente de VoIP



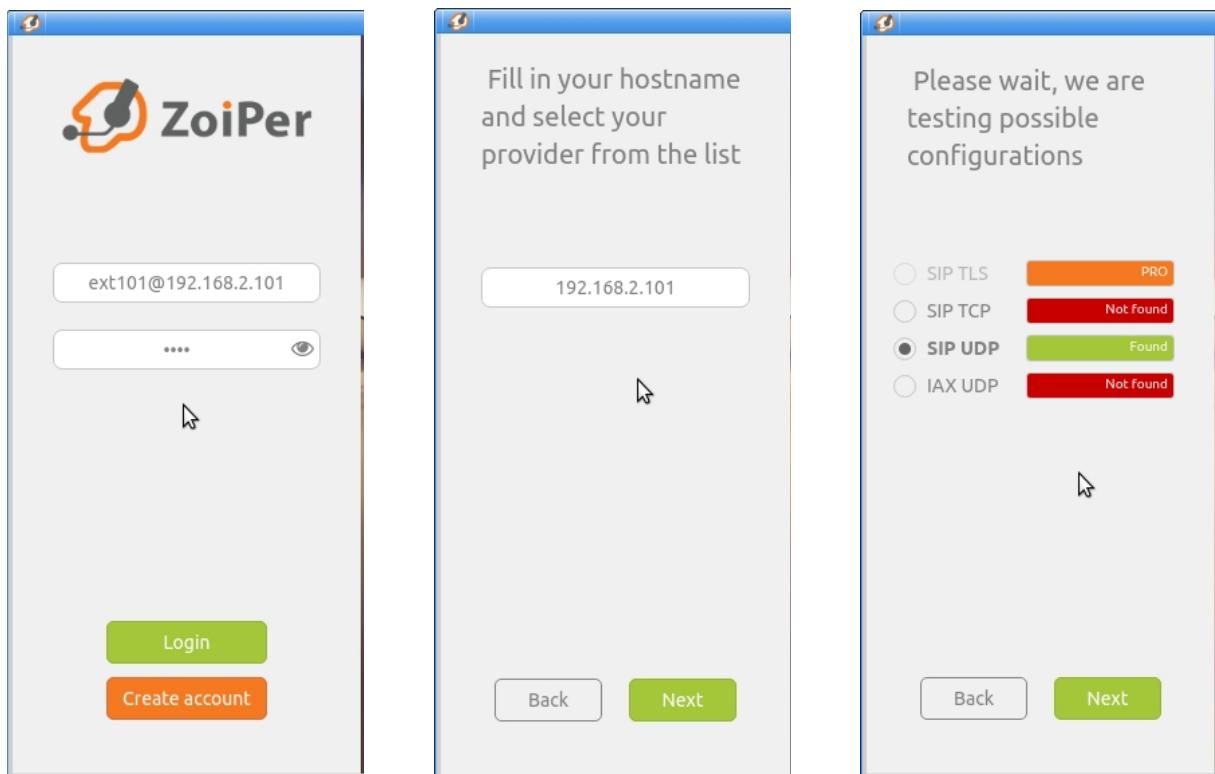
Descargar e instalar.

```
ruben@UbuntuPC-MS-7A34:~$ cd Descargas/  
ruben@UbuntuPC-MS-7A34:~/Descargas$ sudo dpkg -i Zoiper5_5.4.12_x86_64.deb [
```

~ Código:

```
~$ cd Descargas  
~$ sudo dpkg -i Zoiper5_5.4.12_x86_64.deb
```

Iniciar el programa e iniciar sesión con el usuario y la IP de la Raspberry.



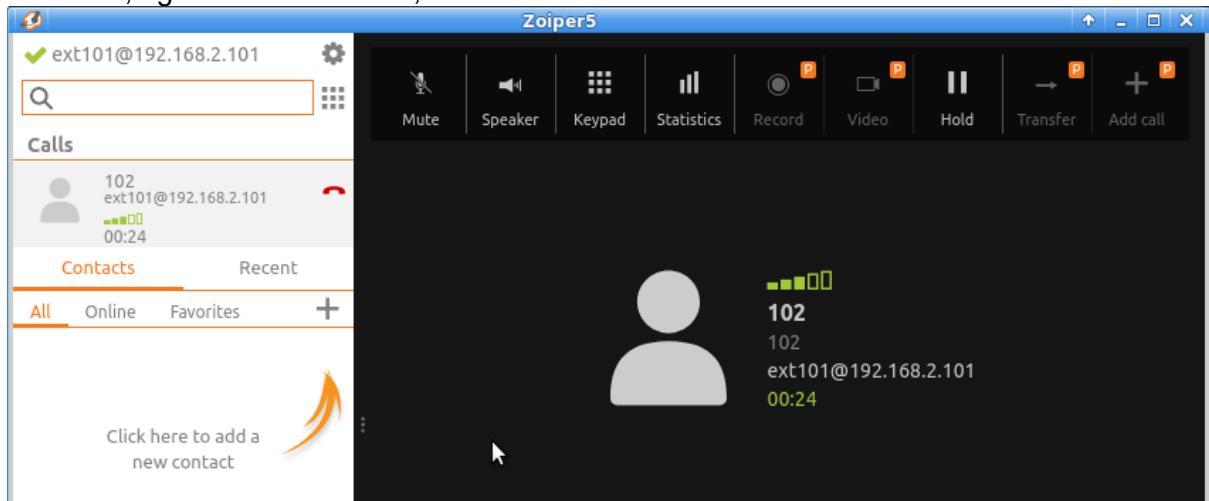
Hacer lo mismo desde otro dispositivo en la misma red, y ver la consola de Asterisk.

```
3 sip peers [Monitored: 1 online, 2 offline Unmonitored: 0 online, 0 offline]  
-- Registered SIP 'ext102' at 192.168.2.102:57985  
[May 24 00:10:27] NOTICE[711]: chan_sip.c:24888 handle_response_pepoke: Peer 'ext102' is now Reachable. (3ms / 20  
raspberrypi*CLI> sip show peers  
Name/username          Host                               Dyn Forcerport Comedia     ACL Port      Status  
ext101/Ruben           192.168.2.100                         D  Auto (No)  No          58062        OK (1 ms)  
ext102/chicoPracticas  192.168.2.102                         D  Auto (No)  No          57985        OK (3 ms)  
ext103/Gertrudis       (Unspecified)                      D  Auto (No)  No          0            UNKNOWN  
3 sip peers [Monitored: 2 online, 1 offline Unmonitored: 0 online, 0 offline]  
raspberrypi*CLI> [
```

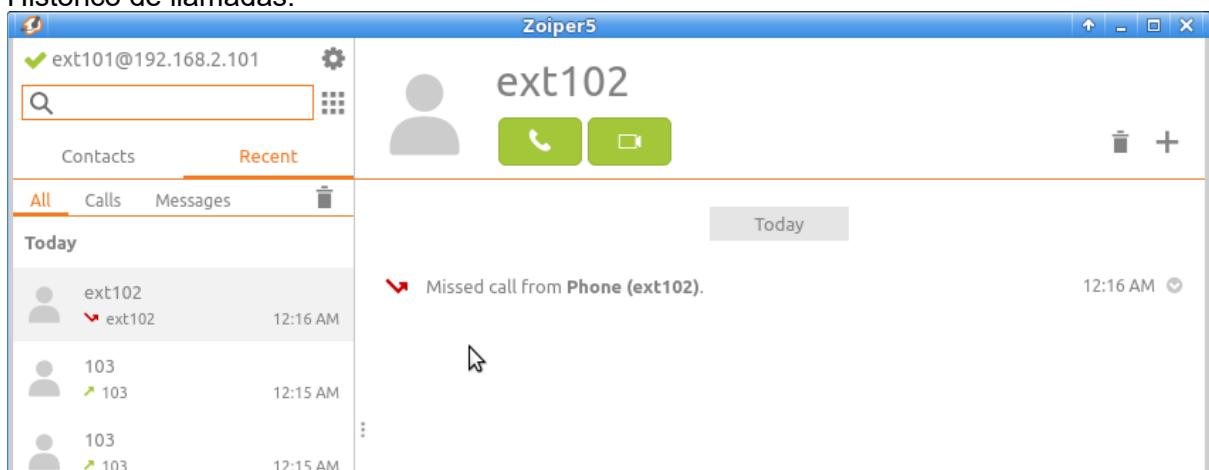
~ Código:

```
~$ sudo asterisk -rvvv  
    sip show peers
```

Como en cualquier teléfono tradicional, permite hacer llamadas. Dispone de un histórico de llamadas, agenda de contactos, etc.



Histórico de llamadas.



Mientras, podemos ver en el terminal de asterisk el cómo se han ido creando logs de qué iba sucediendo.

```
==> Using SIP RTP CoS mark 5
-- Executing [101@ProyectoASIR:1] Dial("SIP/ext102-00000003", "SIP/ext101") in new stack
-- Using SIP RTP CoS mark 5
-- Called SIP/ext101
-- SIP/ext101-00000004 is ringing
-- Spawn extension (ProyectoASIR, 101, 1) exited non-zero on 'SIP/ext102-00000003'
-- Using SIP RTP CoS mark 5
[May 24 00:17:30] NOTICE[711][C-00000005]: chan_sip.c:26699 handle_request_invite: Call from 'Ruben' (192.168.2.100:58062) to extension 'ext102' rejected because extension not found in context 'ProyectoASIR'.
-- Using SIP RTP CoS mark 5
[May 24 00:17:37] NOTICE[711][C-00000006]: chan_sip.c:26699 handle_request_invite: Call from 'Ruben' (192.168.2.100:58062) to extension '1' rejected because extension not found in context 'ProyectoASIR'.
-- Using SIP RTP CoS mark 5
-- Executing [102@ProyectoASIR:1] Dial("SIP/ext101-00000005", "SIP/ext102") in new stack
-- Using SIP RTP CoS mark 5
-- Called SIP/ext102
-- SIP/ext102-00000006 is ringing
-- SIP/ext102-00000006 is ringing
-- Spawn extension (ProyectoASIR, 102, 1) exited non-zero on 'SIP/ext101-00000005'
-- Using SIP RTP CoS mark 5
-- Executing [101@ProyectoASIR:1] Dial("SIP/ext102-00000007", "SIP/ext101") in new stack
-- Using SIP RTP CoS mark 5
-- Called SIP/ext101
-- SIP/ext101-00000008 is ringing
-- Spawn extension (ProyectoASIR, 101, 1) exited non-zero on 'SIP/ext102-00000007'
raspberrypi*CLI>
```

9 - Instalación y configuración de Apache2

Instalación de Apache2

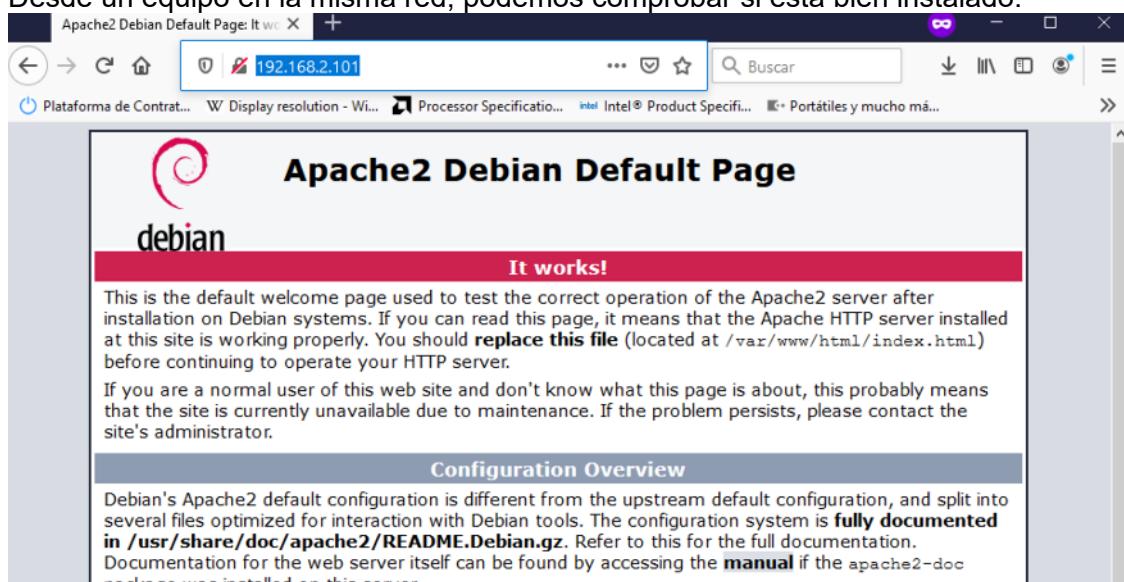
Instalar Apache2.

```
IP actual: 192.168.2.101
pi@raspberrypi:~ $ sudo apt-get update > /dev/null
pi@raspberrypi:~ $ sudo apt-get install apache2
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap
Paquetes sugeridos:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom
Se instalarán los siguientes paquetes NUEVOS:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap
0 actualizados, 8 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 1.969 kB de archivos.
Se utilizarán 6.164 kB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] ■
```

~ Código:

```
~$ sudo apt-get update > /dev/null
~$ sudo apt-get install apache2
```

Desde un equipo en la misma red, podemos comprobar si está bien instalado.



10 - Instalación y configuración de PHP

Instalar PHP

Instalar PHP en la Raspberry.

```
IP actual: 192.168.2.101
pi@raspberrypi:~ $ sudo apt-get update > /dev/null
pi@raspberrypi:~ $ sudo apt-get install php
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
  libapache2-mod-php7.3 php-common php7.3 php7.3-cli php7.3-common php7.3-json
  php7.3-opcache php7.3-readline
Paquetes sugeridos:
  php-pear
Se instalarán los siguientes paquetes NUEVOS:
  libapache2-mod-php7.3 php php-common php7.3 php7.3-cli php7.3-common
  php7.3-json php7.3-opcache php7.3-readline
0 actualizados, 9 nuevos se instalarán, 0 para eliminar y 0 no actualizados.
Se necesita descargar 2.981 kB de archivos.
Se utilizarán 14,1 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n] ■
```

~ Código:

```
~$ sudo apt-get update > /dev/null
~$ sudo apt-get install php
```

Crear un index.php para ver si arranca el módulo de PHP en Apache2

```
pi@raspberrypi:~ $ cd /var/www/html/ && ls -l
total 12
-rw-r--r-- 1 root root 10701 may 24 12:57 index.html
pi@raspberrypi:/var/www/html $ sudo mv index.html index.html.bu
pi@raspberrypi:/var/www/html $ sudo nano index.php && cat index.php
<?php
print "<p>Página información PHP </php>\n";
phpinfo();
pi@raspberrypi:/var/www/html $ ■
```

~ Código:

```
~$ cd /var/www/html/
~$ sudo mv index.html index.html.bu
~$ sudo nano index.php
```

Comprobar que funciona el PHP

Página información PHP

PHP Version 7.3.27-1~deb10u1

System	Linux raspberrypi 5.10.17-v7+ #1414 SMP Fri Apr 30 13:18:35 BST 2021 armv7l
Build Date	Feb 13 2021 16:31:40
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini Path)	/etc/php/7.3/apache2

Carpeta compartida de PHP

Vincular Samba con php para las publicaciones en la página web.

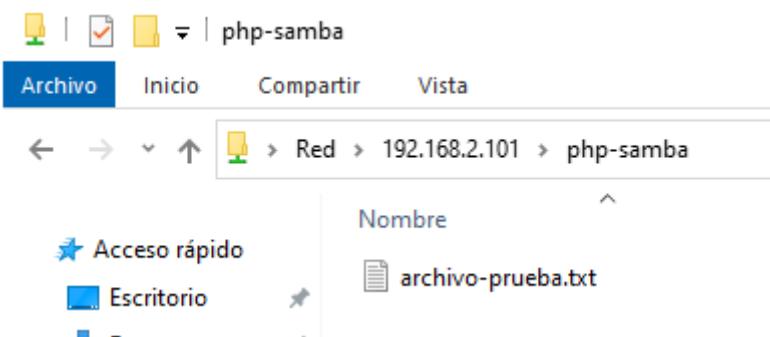
```
pi@raspberrypi:/var/www/html $ sudo mkdir php-samba && sudo chmod 777 php-samba
pi@raspberrypi:/var/www/html $ ls -l
total 20
-rw-r--r-- 1 root root 10701 may 24 12:57 index.html.bu
-rw-r--r-- 1 root root    63 may 24 13:33 index.php
drwxrwxrwx 2 root root  4096 may 24 14:17 php-samba
pi@raspberrypi:/var/www/html $
```

Se va a usar el mismo usuario “pi-samba” que se había creado en el apartado de instalación de Samba.

Añadir la carpeta de PHP al fichero configuración de Samba.

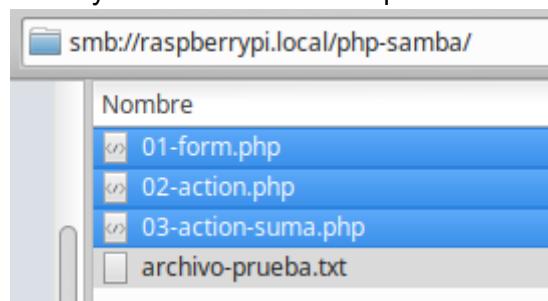
```
pi@raspberrypi:/var/www/html $ cat /etc/samba/smb.conf | tail -8
[php-samba]
    comment = Carpeta PHP para publicaciones web
    path = /var/www/html/php-samba
    writable = yes
    browseable = yes
    guest ok = yes
    create mask = 0666
    directory mask = 0766
pi@raspberrypi:/var/www/html $
```

Podemos ver que ya se está compartiendo.



Página usando sesiones con PHP

Crear y subir los 3 archivos que vamos a usar.



Desde la Raspberry, comprobar que se han pasado a la carpeta de php-samba.

```
pi@raspberrypi:~ $ ls /var/www/html/
index.html.bu index.php  php-samba
pi@raspberrypi:~ $ ls /var/www/html/php-samba/
01-form.php 02-action.php 03-action-suma.php  archivo-prueba.txt
pi@raspberrypi:~ $ ls -l /var/www/html/php-samba/
total 16
-rw-r--r-- 1 root root 1177 may 25 00:59 01-form.php
-rw-r--r-- 1 root root 2581 may 25 01:00 02-action.php
-rw-r--r-- 1 root root  956 may 25 01:00 03-action-suma.php
-rw-r--r-- 1 root root    19 may 24 14:31 archivo-prueba.txt
pi@raspberrypi:~ $
```

Acceder desde el navegador web, en este caso, poner: IP/php-samba



Index of /php-samba

Name	Last modified	Size	Description
------	---------------	------	-------------

Parent Directory		-	
01-form.php	2021-05-25 00:59	1.1K	
02-action.php	2021-05-25 01:00	2.5K	
03-action-suma.php	2021-05-25 01:00	956	
archivo-prueba.txt	2021-05-24 14:31	19	

Apache/2.4.38 (Raspbian) Server at 192.168.2.101 Port 80

Ahora podemos clicar el el archivo primero, el de “01-form.php”.

No es seguro | 192.168.2.101/php-samba/01-form.php

Inserta las credenciales

Usuario:

Contraseña:

Al enviar, abrirá el siguiente archivo, el action. En este comprobará si la contraseña introducida coincide con “root”.

The screenshot shows a web page with a header indicating it's not secure (No es seguro) and the URL 192.168.2.101/php-samba/02-action.php?usuari=ru... . The main content displays user input fields and a form submission button.

El usuario introducido es: ruben

La contraseña introducida es: root (4813494d137e1631bba301d5acab6e7bb7aa74ce1185d456565ef51d737677b2)

Las contraseña introducida coincide con 4813494d137e1631bba301d5acab6e7bb7aa74ce1185d456565ef51d737677b2

— Lista de artículos que desea comprar —

Bolígrafo rojo 0,45€

Bolígrafo negro 0,35€

Bolígrafo azul 0,25€

[⬅ ATRÁS](#)

Al darle al botón “comprar”, veremos que hace la suma de los artículos elegidos en el action.

The screenshot shows a web page with a header indicating it's not secure (No es seguro) and the URL 192.168.2.101/php-samba/03-action-suma.php?arti... . The main content displays calculated values and a back-link.

El usuario está autentificado correctamente

'Boli_rojo' = 0.45 €

'Boli azul' = 0.25 €

Suma total de la compra: 0.7 €

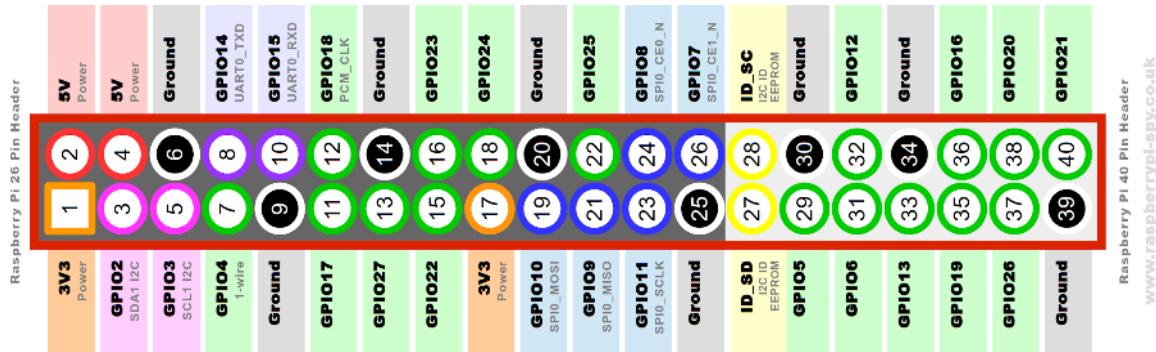
[⬅ ATRÁS](#)

11 - Placa PCB para indicar servicios activos

Al querer controlar de manera visual qué servicios están o no activos, se va a hacer uso de diodos LED conectados al puerto de GPIO.

Interfaz de GPIO

Aquí podemos ver qué utilidad tiene cada uno de los 40 pines que tiene la interfaz.



Con esto se puede ver que hay un total de 17 pines GPIO digitales, que son los que se van a usar, siguiendo este esquema.

40 (V)	38 (B)	33 (B)	22 (B)	13 (B)
	37 (B)	32 (B)	18 (B)	12 (B)
	36 (B)	31 (B)	16 (B)	11 (B)
	35 (B)	29 (B)	15 (B)	7 (B)

Color del LED usado: B = Blanco V = Verde

Calcular resistencia

Según el color del LED, necesita un voltaje u otro para funcionar. Usaremos LEDs color blanco, y verde. Estos funcionan entre 3v y 3v4. Lo calcularemos para 3v, aunque se puede dejar sin. Así que vamos a poner una resistencia de protección.

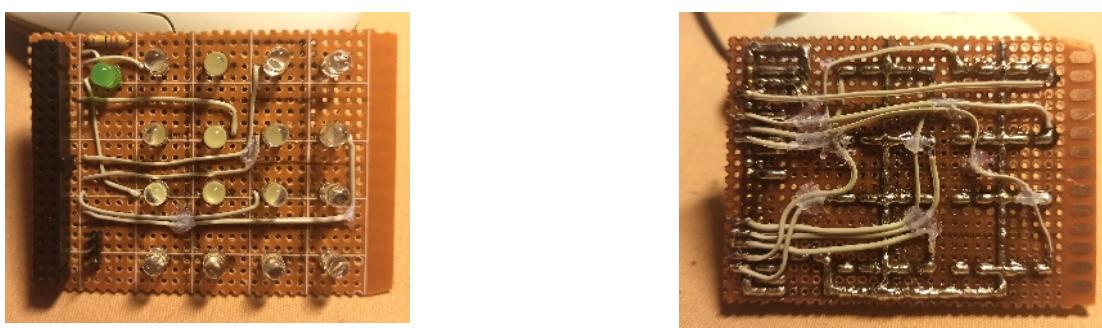
$$R = \frac{V}{I} = \frac{V_f - V_{Led}}{I_{Led}} = \frac{3'3v - 3v}{20mA} = \frac{0'3v}{0'02A} = 15 \Omega$$

Como no tengo una de 15Ω, pondré una de 22Ω.

Diseño placa PCB

Se va a usar 1x placa PCB 7cm x 9cm, 1x conector macho de PATA, 1x LED verde, 16x LEDs blancos, 1x resistencia de 22Ω, 3x pines de metal para salida de 1x GND 1x 3v3 y 1x 5v, cables, y estaño para soldar.

Tras montar todos los componentes en la placa PCB, ha quedado así.



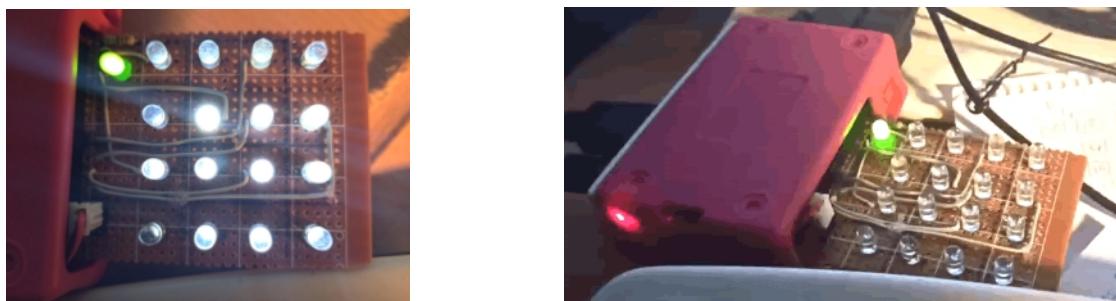
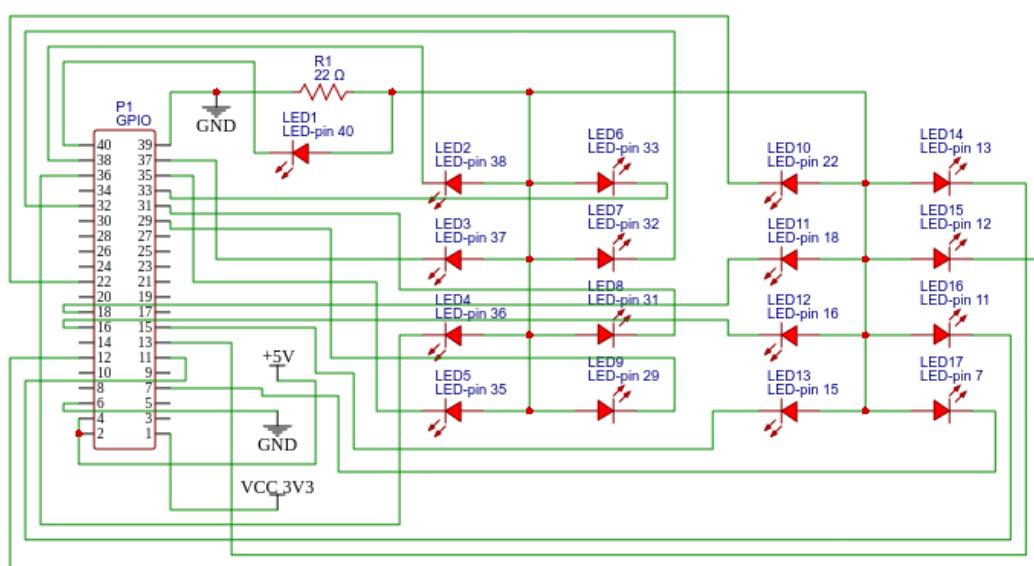


Diagrama esquemático de las conexiones en el PCB. No está hecho a escala, ni para uso real tal como está, solo a modo ejemplo de cómo se ha hecho en el PCB de las imágenes.



12 - Script crear y eliminar servicio LED

Contenido archivos de activación de LEDs

Los 4 archivos creados con el script, contienen un código similar a esto:

Script control de servicio activo .sh	Archivo del servicio LED .service
<pre>#!/bin/bash # /etc/GPIO_LED_SH_PY/LED-40-sshd.sh while true; do systemctl status sshd.service head -3 tail -1 grep "(running)" > /dev/null if [\$? = 0]; then python /etc/GPIO_LED_SH_PY/LED- 40-sshd-on.py > /dev/null echo "LED 40 sshd on - .sh" else python /etc/GPIO_LED_SH_PY/LED- 40-sshd-off.py > /dev/null echo "LED 40 sshd off - .sh" fi sleep 0.1 done</pre>	<pre>[Unit] Description= Servicio de sshd para LED en GPIO 40 Documentation= https://github.com/rubentey/scripts After=systemd-user-sessions.service Wants=network.target [Service] Type=simple ExecStart=/etc/GPIO_LED_SH_PY/LED-40- sshd.sh [Install] WantedBy=multi-user.target</pre>
Archivo enciende LED .py	Archivo apaga LED .py
<pre>#!/bin/python # Pin: 40 Service: sshd import RPi.GPIO as GPIO import time GPIO.setmode(GPIO.BOARD) GPIO.setup(40,GPIO.OUT) GPIO.output(40,1) print ("LED sshd on - .py") time.sleep(2) GPIO.cleanup()</pre>	<pre>#!/bin/python # Pin: 40 Service: sshd import RPi.GPIO as GPIO import time GPIO.setmode(GPIO.BOARD) GPIO.setup(40,GPIO.OUT) GPIO.output(40,0) print ("LED sshd off - .py") time.sleep(2) GPIO.cleanup()</pre>

El script mencionado en el punto anterior, consta de más de 330 líneas, por lo que estará adjuntado en el soporte digital, y el enlace a [GitHub en la Webgrafía](#).

Consiste en 4 menús.

- 1º- Hacer que el script se use con “sudo”.
- 2º- Consultar que LED y servicio están usados.
- 3º- Asignar un servicio a un LED.
- 4º- Eliminar un servicio a un LED.

En el apartado 3º, que es el más relevante, los archivos que se crean son 4.

- 1º- Archivo .sh para comprobar si X servicio está activo. Si lo está, ejecuta el 2º archivo, si no lo está, ejecuta el 3º.
- 2º- Archivo .py para activar LED asignado a X servicio.
- 3º- Archivo .py para desactivar LED asignado a X servicio.
- 4º- Servicio de Systemd para controlar el 1º archivo.

Servicios con systemd

Algunos comandos útiles para el uso y control de servicios, como los creados en este apartado del script .sh:

Con systemctl:

```
sudo systemctl start led-40-ssh.service  
sudo systemctl status led-40-ssh.service  
sudo systemctl stop led-40-ssh.service  
sudo systemctl enable led-40-ssh.service  
sudo systemctl disable led-40-ssh.service  
sudo systemctl daemon-reload  
sudo systemctl restart led-40-ssh.service  
sudo systemctl edit led-40-ssh.service --full
```

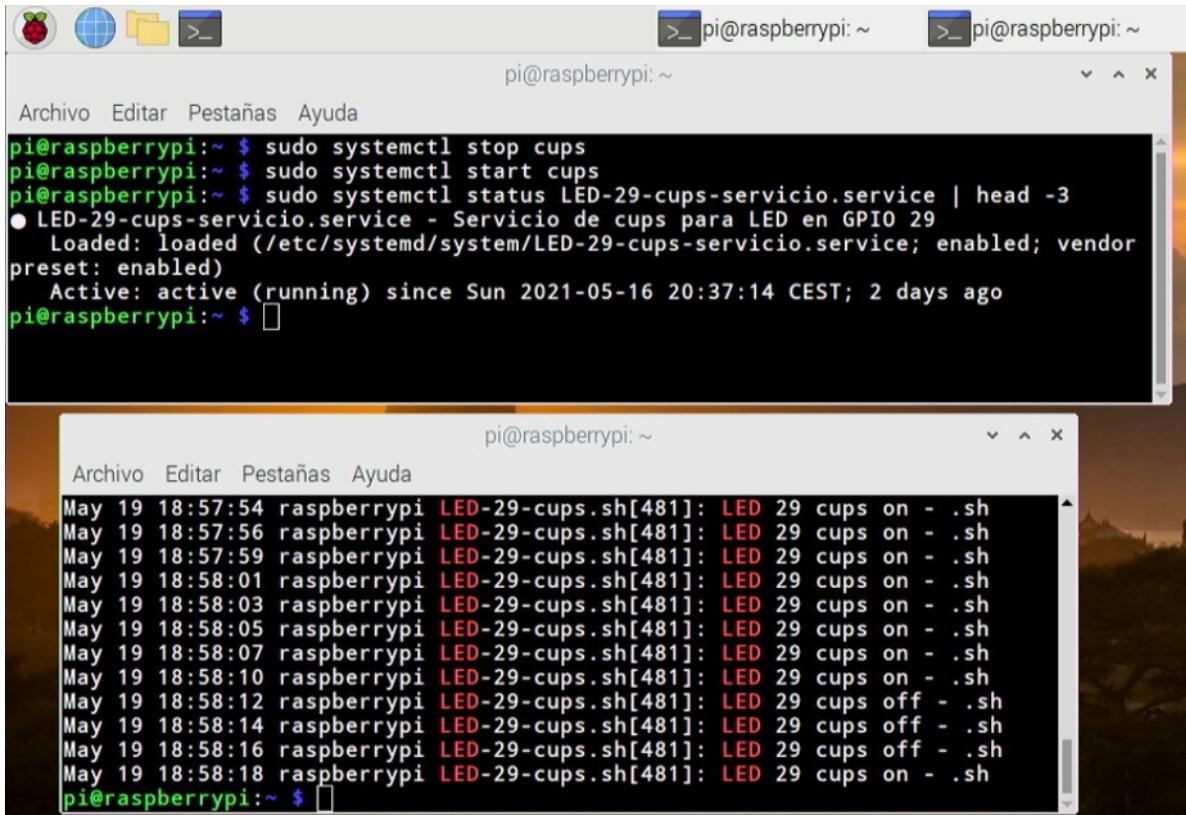
Con otros:

```
sudo journalctl -u led-40-ssh.service -f  
cat /var/log/syslog  
cat /var/log/syslog | tail -10
```

Uso de los servicios de los LEDs

Tras hacer pruebas con el script para crear nuevos archivos de servicios, podemos probar a ver qué tal va, y se vería tal que así.

En la ventana de arriba, vemos como al activar / desactivar un servicio como puede ser el “CUPS”. Y en la ventana de abajo, destinada a ver el log del sistema, se ve claramente como el servicio de LEDs detecta correctamente la interacción hecha, y manda la señal con el “echo “LED 29 cups on / off .sh””.



The image shows two terminal windows running on a Raspberry Pi desktop. The top window has a title bar 'pi@raspberrypi: ~' and displays the following command-line session:

```
pi@raspberrypi:~ $ sudo systemctl stop cups
pi@raspberrypi:~ $ sudo systemctl start cups
pi@raspberrypi:~ $ sudo systemctl status LED-29-cups-servicio.service | head -3
● LED-29-cups-servicio.service - Servicio de cups para LED en GPIO 29
  Loaded: loaded (/etc/systemd/system/LED-29-cups-servicio.service; enabled; vendor
preset: enabled)
    Active: active (running) since Sun 2021-05-16 20:37:14 CEST; 2 days ago
pi@raspberrypi:~ $
```

The bottom window also has a title bar 'pi@raspberrypi: ~' and displays the following command-line session:

```
pi@raspberrypi:~ $ cat /var/log/syslog | grep "LED"
May 19 18:57:54 raspberrypi LED-29-cups.sh[481]: LED 29 cups on - .sh
May 19 18:57:56 raspberrypi LED-29-cups.sh[481]: LED 29 cups on - .sh
May 19 18:57:59 raspberrypi LED-29-cups.sh[481]: LED 29 cups on - .sh
May 19 18:58:01 raspberrypi LED-29-cups.sh[481]: LED 29 cups on - .sh
May 19 18:58:03 raspberrypi LED-29-cups.sh[481]: LED 29 cups on - .sh
May 19 18:58:05 raspberrypi LED-29-cups.sh[481]: LED 29 cups on - .sh
May 19 18:58:07 raspberrypi LED-29-cups.sh[481]: LED 29 cups on - .sh
May 19 18:58:10 raspberrypi LED-29-cups.sh[481]: LED 29 cups on - .sh
May 19 18:58:12 raspberrypi LED-29-cups.sh[481]: LED 29 cups off - .sh
May 19 18:58:14 raspberrypi LED-29-cups.sh[481]: LED 29 cups off - .sh
May 19 18:58:16 raspberrypi LED-29-cups.sh[481]: LED 29 cups off - .sh
May 19 18:58:18 raspberrypi LED-29-cups.sh[481]: LED 29 cups on - .sh
pi@raspberrypi:~ $
```

~ Código:

```
# Ventana de arriba
~$ sudo systemctl stop cups
~$ sudo systemctl start cups
~$ sudo systemctl status LED-29-cups-servicio.service | head -3

# Ventana de abajo, ejecutada tras unos segundos
~$ cat /var/log/syslog | grep "LED"
```

13 - Instalar y conectar Python a Telegram

De serie en la mayoría de distribuciones de los sistemas operativos GNU/Linux actuales, suele venir ya instalado Python. Debido a que ciertas partes del sistema operativo o de programas, lo requieren para su funcionamiento. Por ello no hará falta instalarlo en la mayoría de los casos. Así que si en la distro que se está usando, viene Python integrado, no se recomienda bajo ningún concepto, ejecutar un “purge” o un “remove”, de Python.

En el caso de Raspberry, en la distribución que se ha elegido, ya viene por defecto Python instalado. Así que tan solo hará falta instalar las extensiones que vamos a requerir para poder usarlo para el fin que buscamos. En este caso, que se pueda interconectar Python con Telegram, y que Python pueda interactuar con los pines del GPIO, y así poder usar correctamente el script del apartado anterior.

Instalar paquetes de Python

Instalar paquete para usar los pines de GPIO. Con este podremos usar los LEDs.

```
pi@raspberrypi:~ $ sudo apt-get update && sudo apt-get upgrade > /dev/null
Obj:1 http://archive.raspberrypi.org/debian buster InRelease
Obj:2 http://raspbian.raspberrypi.org/raspbian buster InRelease
Leyendo lista de paquetes... Hecho
pi@raspberrypi:~ $ sudo apt-get install -y python-pip > /dev/null
pi@raspberrypi:~ $ sudo pip install -U RPi.GPIO
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already up-to-date: RPi.GPIO in /usr/lib/python2.7/dist-packages (0.7.0)
pi@raspberrypi:~ $
```

~ Código:

```
#Actualizar el base de datos de versiones de aplicaciones, y actualizar el sistema
~$ sudo apt-get update && sudo apt-get upgrade > /dev/null
~$ sudo apt-get install -y python-pip > /dev/null      # Instalar gestor de paquetes
~$ sudo pip install -U RPi.GPIO      # Controla los pines de GPIO
```

Instalar paquete de “telepot”. El cual nos permitirá conectar Python a Telegram.

```
IP actual: 192.168.2.102
pi@raspberrypi:~ $ sudo pip install -U telepot
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting telepot
Requirement already satisfied, skipping upgrade: urllib3>=1.9.1 in /usr/lib/python2.7/dist-packages (from telepot) (1.24.1)
Installing collected packages: telepot
Successfully installed telepot-12.7
pi@raspberrypi:~ $
```

~ Código:

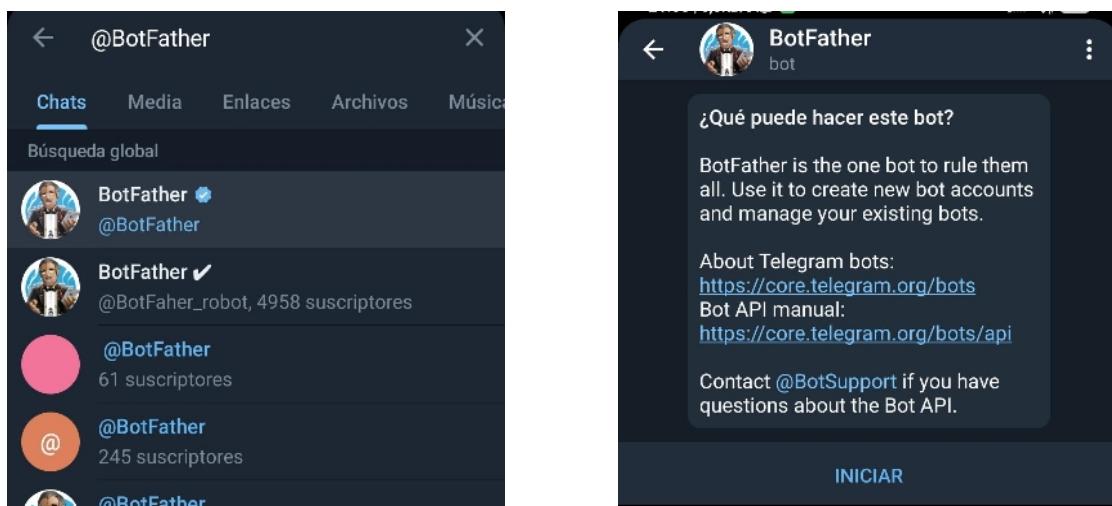
```
~$ sudo pip install -U telepot
```

Crear bot de Telegram

Instalar aplicación de Telegram en el teléfono móvil o en el ordenador.



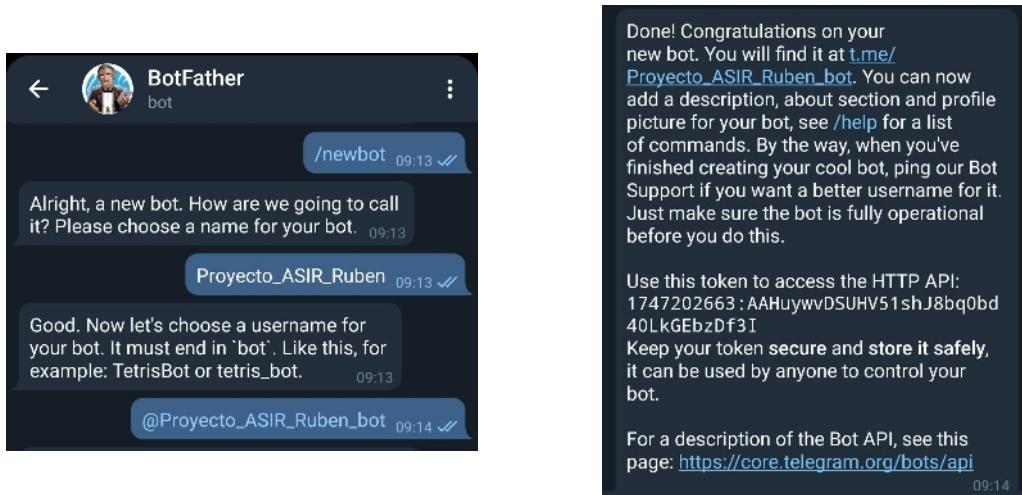
Crear una cuenta o iniciar sesión, y abrir chat con la cuenta de @BotFather.



Vamos a darle a "/start", esto nos desplegará un menú con las opciones disponibles.



En este caso, vamos a usar la opción de "/newbot". Asignaremos un nombre al bot, y un nombre de usuario (acabado en bot). Al acabar con el asistente, nos dará el mensaje del resultado. Este mensaje es muy importante, ya que es el que contiene el token de identificador único para nuestro bot con Python.



Tras esto, ya tenemos creado el bot de Telegram, resta enlazarlo a Python en Raspberry Pi.

Conectar Python a Telegram

En este caso, pasa como con el script del apartado 7, al ser tan largo, estará adjuntado con los demás archivos del proyecto en la unidad de almacenamiento externo, y en el [Github de la webgrafía](#).

Un pequeño ejemplo de cómo está hecho, pero a pequeña escala:

```
#!/bin/python
# Bot telegram @Proyecto_ASIR_Ruben_bot

import time
import os
import RPi.GPIO as GPIO
import telepot
from telepot.loop import MessageLoop

def action(msg):
    message = "Bienvenid@"

    chat_id = msg['chat']['id']
    cmd = msg['text']
    print ('Recibido: %s' % cmd)

    if 'on' in cmd:
        message = "on "
    if 'cups' in cmd:
        message = message + "cups "
        os.system('sudo systemctl start cups')
```

```
message = message + "service"
telegram_bot.sendMessage (chat_id, message)

if 'off' in cmd:
    message = "off "
if 'cups' in cmd:
    message = message + "cups "
    os.system('sudo systemctl stop cups')

message = message + "service"
telegram_bot.sendMessage (chat_id, message)

telegram_bot = telepot.Bot('1747202663:AAHuywvDSUHV51shJ8bq0bd40LkGEbzDf3I')
print (telegram_bot.getMe())

MessageLoop(telegram_bot, action).run_as_thread()
print ('Tamo ready....')

while 1:
    time.sleep(1)
```

En este caso no ha funcionado correctamente para hacer el mensaje de bienvenida al darle a "/start". Pero puede ser útil para ciertas versiones.

```
pi@raspberrypi:~ $ python -m pip install --upgrade pip
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting pip
  Downloading https://files.pythonhosted.org/packages/27/79/8a850fe3496446ff0d58
4327ae44e7500daf6764ca1a382d2d02789accf7/pip-20.3.4-py2.py3-none-any.whl (1.5MB)
  100% |██████████| 1.5MB 156kB/s
Installing collected packages: pip
Successfully installed pip-20.3.4
pi@raspberrypi:~ $
```

~ Código:

```
~$ python -m pip install --upgrade pip
```

Este puede ayudar a solucionar el error de “[ERROR] No module named telegram.ext”. En este caso no lo ha solucionado.

```
pi@raspberrypi:~ $ sudo pip install python-telegram-bot
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting python-telegram-bot
  Downloading https://files.pythonhosted.org/packages/40/81/6d361e13a3eb02c3e1d4
886445dcc464d664a32cb0e60d506f07e2b6ac77/python-telegram-bot-13.1.tar.gz (284kB)
  100% |██████████| 286kB 700kB/s
```

~ Código:

```
~$ sudo pip install python-telegram-bot
```

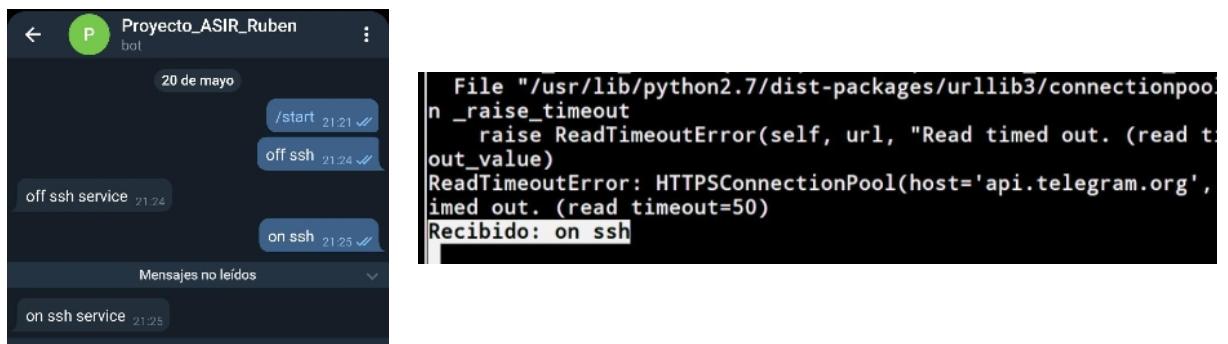
Activar el bot de Python a Telegram

Poner en marcha el bot, ejecutando por terminal el archivo de .py de Python

~ Código:

```
~$ python Proyecto_ASIR_Ruben_bot_Telegram.py
```

Ver que hace petición-respuesta. Y lo que llega al terminal de la Raspberry Pi.



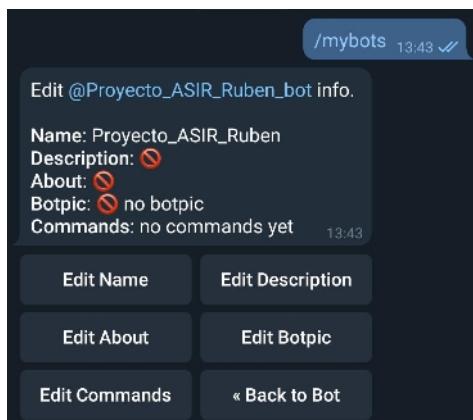
Si el servicio elegido está asignado a un LED del GPIO, éste se encenderá / apagará.



Editar perfil del bot de Telegram

Para que parezca más profesional, vamos a editar el perfil del bot. Para ello, contactamos con @BotFatherBot. Le añadimos una imagen, una descripción, etc.





Le añadimos una descripción. Y comprobamos el cómo queda el perfil del bot que estamos usando.



14 - Dificultades encontradas y aprendizajes alcanzados

Dificultades:

La primera el límite de hojas que se pueden presentar, me habría gustado poder añadir más cosas, pero he tenido que reducir y quitar lo máximo posible el contenido listado en el documento.

En el apartado del script run_New_GPIO_Service.sh, ha habido algunas dificultades como la de el filtrado de números que tras buscar varios días en foros (stackoverflow incluido), fue cambiar un “==”, por un “-eq” y ya perfecto. Aunque no pude implementar variables para ahorrar líneas de archivo.

En el mismo script run_New_GPIO_Service.sh, el apartado de crear servicio, he estado durante tiempo buscando tutoriales y páginas web con información para ver como poder usar los servicios, y lo único que faltaba por hacerle fue el “enable” al servicio del archivo creado.

En el Python, me habría gustado hacer una interfaz más intuitiva con botones en lugar de escribir todo. Pero las librerías y configuraciones que se sugerían tanto en en internet como en la página oficial, no se acababan de instalar bien o saltaban errores.

Me habría gustado poder implementar una base de datos para hacer consultas al Python y por tanto que este haga una comprobación de login con el telegram a ver si el usuario que envía el mensaje está habilitado en el sistema.

Aprendizajes:

He podido aprender cosas que no pensaba que fuesen tan útiles y emocionantes de hacer. El poder usar los pines de GPIO de la Raspberry de forma que se puedan controlar visualmente el estado de los servicios. El crear servicios, que tienen su truco pero muy útiles e interesantes. El crear un bot de Telegram y usarlo con Python, que como ampliación a un futuro proyecto, reemplazar un LED por una salida a algo de domótica.

El servicio de Asterisk tampoco lo había usado ni lo conocía. Con la gran cantidad de opciones que dispone, chats, grupos, videollamada, etc. Es interesante ver el cómo software libre puede ser tan bueno o a veces incluso mejor que uno privativo

15 - Experiencia personal

Ha sido un placer poder haber tenido la oportunidad de conocer esta rama del estudio y aprendizaje. Gracias a esta he podido llegar a conocer y poner cierto interés a este tipo de mini ordenadores.

El uso de los servicios gracias al uso de Systemd, es algo que ni conocía ni había llegado a probar. Pero eso sí, sé que ahora me parece algo que ha venido a mi vida para quedarse y no para poco tiempo.

Tras haber realizado esta práctica, aunque haya sido para un propósito educativo, sé que en mayor o menor medida la implantaré para un uso particular. Como pueda ser un servidor en casa con el que controlar alguna pequeña cosa de domótica a través del teléfono móvil con el Telegram.

El uso de recursos compartidos y que puedas habilitarlo o denegar el paso con un simple click en el Telegram, es algo que aunque insignificante, puede ser realmente útil para este y otros muchos propósitos de la vida cotidiana que requieran de la informática.

Solo sé que me llevo mucho más que una gran e inolvidable experiencia con todo lo que ha englobado este proyecto.

Personalmente se puede decir que se puede hacer esto mismo en una máquina virtual, pero poder experimentar casi en tus manos lo que se ha logrado, esto no tiene precio.

Así que no tengo más que decir que me ha encantado poder haber realizado este proyecto por mí mismo, con la ayuda de algún tutorial, ya que no nacemos sabiendo, y poder llevarme conmigo toda la buena experiencia obtenida con este.

No me queda nada más que decir que Gracias por todo, y agradecer al profesorado todo lo que me ha podido aportar en estos años.

16 - Conclusiones finales

A pesar de que este proyecto haya sido un requisito indispensable para la finalización de todo este curso. Desde el primer momento quise dedicarme a enfocarlo a lo que a grandes rasgos se considera el ciclo, a los servicios y la administración de estos, con uso de redes.

A parte, que gracias a sistemas como el Raspberry Pi (el elegido para este ya tiene unos años), que aún siendo pequeños y de escasos recursos, permiten hacer grandes proyectos. Y partiendo de la base del software libre.

Aunque es cierto que no conocía del todo las características que en este se podía llegar a exprimir, quise darlas a conocer un poco más tras haberlas descubierto para mí mismo. Como es el uso de la domótica (en este caso lucecitas LED), o el uso de tener un Bot de Telegram con el que hablarle a tu Raspberry Pi.

Como es normal y a cualquiera le puede pasar por la cabeza al ver a este tipo de mini ordenadores con los limitantes que tienen como es el 1GB de RAM, es que si será suficiente para hacer funcionar todo lo que se tiene pensado incorporarle.

Cosa que la duda a mí mismo también me surgía, pero tras ver que al implementar un servicio funcionaba bien, al implementar el siguiente también, y así sucesivamente... Ha dejado clara una cosa, que este dispositivo es una obra de la ingeniería en cuanto a optimización y gestión de recursos usados en su interior. Lo cual ha permitido que se pueda haber llevado a cabo la práctica.

Es cierto que el proyecto está orientado unas 40 horas aproximadamente, y que puede ser unas pocas más o unas pocas menos. En mi caso la realización completa de todo el proyecto, con base de prueba y error sobretodo en los scripts de bash y de python, ha llegado a las más de 140 horas de dedicación.

Decir que muchos de los tutoriales carecían de ciertos pasos realmente necesarios para alguien inexperto en los temas a tocar, o que algunas librerías necesarias, ya estaban obsoletas o no disponían de instalador.

Aún así ha sido un gran proyecto y me ha hecho sentirme realizado como persona, ante los usos que le he podido dar y llevar a cabo en este mini dispositivo.

Sí que es cierto que el proyecto tiene cierta parte de programación, y que igual es un poco raro ver un proyecto de programación en un grado de hardware. Este es otro de esos toques con los que llevaba en mente de realizar cuando llegase el momento de hacer el proyecto presente.

Un límite que me ha limitado bastante a la hora de realizar este proyecto son las páginas máximas, ya que sin este límite ni el del tiempo a usar, me habría gustado implementar más y mejores cosas. Como el uso de MySQL en el programa de Python para el Bot, y así hacer que los usuarios se autentiquen y se haga una consulta a la base de datos a ver si este usuario existiese en la tabla.

Para concluir, muy gratamente sorprendido por este mini ordenador y todo su potencial.

17 - Dedicatoria o agradecimiento

No puedo acabar este proyecto y por tanto este curso, sin agradecer al profesorado del IES Serpis, tanto su profesionalidad como su implicación en el progreso y aprendizaje del alumnado.

En especial al profesorado del departamento de redes, el departamento de hardware y software, y al departamento de seguridad informática. Por los cuales realmente se ha podido sentir uno satisfecho y agradecido por su trabajo y por ende, de sus frutos que ha llevado este.

A pesar del periodo C19 sufrido a nivel internacional, se ha experimentado la buena profesionalidad de los representantes de estos departamentos en las clases presenciales y online.

En especial quiero agradecer a Juan Ignacio Febrer Senar, y a Jose Ramón por su imparable continuidad en insistencia en que todo el alumnado logre aprender lo máximo y mejor posible para que en un futuro se puedan valer por sí mismos.

Me he llevado una buena impresión del IES Serpis gracias a profesionales como estos mencionados anteriormente. Los cuales espero no pierdan la ilusión y esperanza que nos hacen percibir en todo momento.

18 - Contenido soporte digital

Carpetas

Apache2

index.html.bu
index.php

Asterisk

cdr.conf.bu
cel.conf.bu
core-sounds-es.txt
extensions.conf
extensions.conf.bu
sip.conf
sip.conf.bu

Ejemplos_LEDs

LED-29-cups-off.py
LED-29-cups-on.py
LED-29-cups-servicio.service
LED-29-cups.sh

PCB

PCB LEDs GPIO servicios_2021-05-17.json
Schematic_PCB LEDs GPIO servicios_2021-05-17.pdf
Schematic_PCB LEDs GPIO servicios_2021-05-17.png

PHP

01-form.php
02-action.php
03-action-suma.php

Samba

smb.conf
smb.conf.bu

Archivos sueltos

IESSerpis_2021_21_Proyecto_ASIR_Ruben_Mellado_Culebras-Memoria_Proyecto.pdf

Proyecto_ASIR_Ruben_bot_Telegram.py

run-New_GPIO_Service.sh

servicios-con-LED-actualmente.sh

2021-01-11-raspios-buster-armhf.zip

balena-etcher-electron-1.5.116-linux-x64.zip

Zoiper5_5.4.12_x86_64.deb

19 - Software utilizado, uso e instalación

Instalación ISO de Raspberry

Balena Etcher - La instalación se indica en el apartado correspondiente
GParted - Viene instalado de serie

Carpetas compartidas en red

Samba - La instalación se indica en el apartado correspondiente

Conexión a terminal remoto

SSH - La instalación se indica en el apartado correspondiente

Llamadas por internet VoIP (protocolo SIP)

Asterisk - La instalación se indica en el apartado correspondiente
ZoiPer - La instalación se indica en el apartado correspondiente

Servicio de páginas web

Apache2 - La instalación se indica en el apartado correspondiente
PHP - La instalación se indica en el apartado correspondiente

Placa de PCB para los LEDs que indican servicio activo

EasyEDA - Es una página web con servicio en la nube

Scripts (servicios para LEDs y bot de Telegram)

Bash - Viene integrado en el sistema
Python - En las distros más recientes, viene integrado
Los paquetes de Python se instalan en el apartado correspondiente
Telegram - La instalación se indica en el apartado correspondiente

20 - Bibliografía y Webgrafía

1 - ¿Qué es Raspberry Pi?

- ⌚ Variedad y modelos en el mercado (https://es.wikipedia.org/wiki/Raspberry_Pi)

2 - ¿Qué es Telegram?

- ¿Es solo mensajes de texto? (<https://es.wikipedia.org/wiki/Telegram>)

3 - ¿Qué es Python?

- Historia de Python (<https://es.wikipedia.org/wiki/Python>)
- Plataformas soportadas (<https://www.python.org/downloads/>)
- Plataformas soportadas (<https://micropython.org/>)

4 - ¿Qué es Bash?

- Historia de Bash (<https://es.wikipedia.org/wiki/Bash>)
- Intérprete de órdenes (<https://www.gnu.org/savannah-checkouts/gnu/bash/manual/bash.html>)

5 - Configuración inicial Raspberry Pi:

- ⌚ Descargar Sistema Operativo (<http://raspberrypi.org/software/operating-systems/>)
- ⌚ Grabar ISO en MicroSD (<https://www.balena.io/etcher/>)
- ⌚ Grabar ISO en MicroSD (sudo apt-get install gparted)
- ⌚ Arrancar desde USB (<https://www.kernel.org/doc/html/v5.0/admin-guide/kernel-parameters.html>)
- ⌚ Arrancar desde USB (<https://www.raspberrypi.org/documentation/installation/installing-images/>)
- ⌚ Arrancar desde USB (<https://github.com/raspberrypi/usbboot>)
- ⌚ Arrancar desde USB (<https://www.raspberrypi.org/documentation/hardware/raspberrypi/bootmodes/msd.md>)
- ⌚ Arrancar desde USB (<https://www.raspberrypi.org/documentation/hardware/raspberrypi/bootmodes/device.md>)
- ⌚ Arrancar desde USB (<https://www.raspberrypi.org/documentation/hardware/raspberrypi/bootmodes/net.md>)
- ⌚ Arrancar desde USB (<https://www.raspberrypi.org/documentation/hardware/raspberrypi/bootmodes/host.md>)
- ⌚ Arrancar desde USB (https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2711_bootloader_config.md)
- ⌚ Arrancar desde USB (<https://raspberrypi.stackexchange.com/questions/100720/rpi4-wont-boot-with-powered-usb-hub>)
- ⌚ Arrancar desde USB (<https://www.raspberrypi.org/documentation/configuration/config-txt/boot.md>)

- ⌚ Arrancar desde USB (<https://www.raspberrypi.org/documentation/configuration/config-txt/README.md>)
- ⌚ Arrancar desde USB (<https://www.raspberrypi.org/forums/viewtopic.php?p=462172>)

8 - Instalación y configuración de VoIP / SIP con Asterisk y ZoiPer:

- Asterisk (<https://linuxize.com/post/how-to-install-asterisk-on-ubuntu-18-04/>)
- Asterisk (<https://www.youtube.com/playlist?list=PLXXznRYETLfnWuAQHrMayGDPnBhSBICb>)

11 - Placa PCB para servicios activos:

- ⌚ Interfaz de GPIO (<https://duckduckgo.com/?q=gpio+raspberry&ia=web>)
- ⌚ Calcular resistencia (<https://www.digikey.es/es/resources/conversion-calculators/conversion-calculator-resistor-color-code>)
- ⌚ Diseño placa PCB (<https://easyeda.com/editor>)

12 - Script crear y eliminar servicio LED:

- ⌚ Github con el script (<https://github.com/rubentey/scripts/tree/main/script%20prueba%20LED%20GPIO>)
- ⌚ Contenido archivos de activación de LEDs (<https://stackoverflow.net.xyz/es/q/4965722>)
- ⌚ Contenido archivos de activación de LEDs (<https://stackoverflow.com/questions/1921894/grep-and-python>)
- ⌚ Contenido archivos de activación de LEDs (<https://osxdaily.com/2018/04/05/how-exclude-word-grep-command-line/>)
- ⌚ Servicios con systemd (<https://www.youtube.com/watch?v=fYQBvjYQ63U>)
- ⌚ Servicios con systemd (<https://www.devdungeon.com/content/creating-systemd-service-files>)
- ⌚ Servicios con systemd (<https://atareao.es/tutorial/trabajando-con-systemd/vigila-una-web-con-un-servicio-de-systemd/>)
- ⌚ Servicios con systemd (<https://atareao.es/tutorial/trabajando-con-systemd/como-crear-un-servicio-con-systemd/>)
- ⌚ Servicios con systemd (<https://atareao.es/tutorial/trabajando-con-systemd/>)
- ⌚ Servicios con systemd (<https://www.youtube.com/playlist?list=PL6IQ3nFZzWfpKKWfZMRxiuEBwqQBwjzS1>)

13 - Conectar Python a Telegram:

- ⌚ Github con el script (https://github.com/rubentey/python-bot/blob/main/Proyecto_ASIR_Ruben_bot_Telegram.py)
Instalar paquetes de Python (<https://askubuntu.com/questions/865554/how-do-i-install-python-3-6-using-apt-get>)
Conectar Python a Telegram (<https://maker.pro/raspberry-pi/projects/how-to-create-a-telegram-bot-with-a-raspberry-pi>)
- ⌚ Conectar Python a Telegram (<https://codereview.stackexchange.com/questions/257167/how-to-optimize-my-telegram-bot-written-in-python>)
- ⌚ Conectar Python a Telegram (<https://atareao.es/tutorial/crea-tu-propio-bot-para-telegram/bot-en-python-para-telegram/>)

- ⌚ Conectar Python a Telegram (<https://atareao.es/podcast/controlar-tu-raspberry-desde-telegram/>)
- ⌚ Conectar Python a Telegram (<https://www.youtube.com/watch?v=jhFsFZXZbu4>)
- ⌚ Conectar Python a Telegram (https://www.youtube.com/watch?v=_rjpvNLghe)
- ⌚ Conectar Python a Telegram (<https://www.youtube.com/watch?v=rRJ6H5gxaNA&list=PLiSFWeqxflp1MdPhpNVL7ldlh25Lqfdvn>)
- ⌚ Conectar Python a Telegram (<https://www.youtube.com/watch?v=DBoWGTO4TaU>)
- ⌚ Conectar Python a Telegram (<https://www.youtube.com/watch?v=73coNZ-5hWM>)
- ⌚ Conectar Python a Telegram (<https://github.com/python-telegram-bot/python-telegram-bot/issues/1691>)

Otros enlaces de interés:

- ⌚ Canal de YT de Luke Smith, muy útil para consultar y aprender utilidades de Linux:
<https://www.youtube.com/channel/UC2eYFnH61tmylmy1mTYvhA>
 - ⌚ Blog de Atareao consultado para algunos puntos del proyecto:
<https://atareao.es/tutoriales/>
- MP4 a GIF - imágenes de LEDs en PCB encendidos (~\$ ffmpeg -i in.mp4 out.gif)