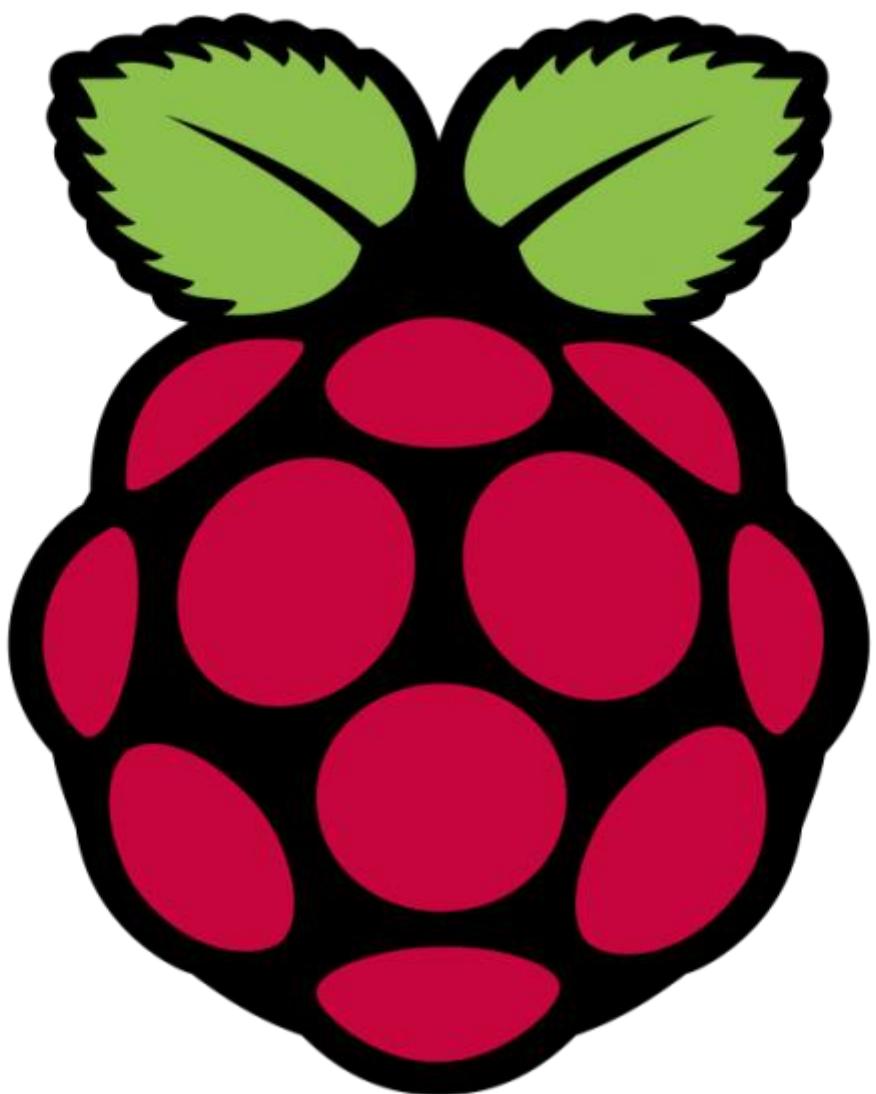


Raspberry Pi



- 1 CONFIGURACIÓN**
- 2 RED**
- 3 COMANDOS LINUX**
- 4 TRUCOS**
- 5 CÁMARA**
- 6 PUERTO SERIAL**
- 7 ARDUINO**
- 8 PROGRAMACIÓN PYTHON**
- 9 FIRMATA**
- 10 VPN**
- 11 SAMBA**

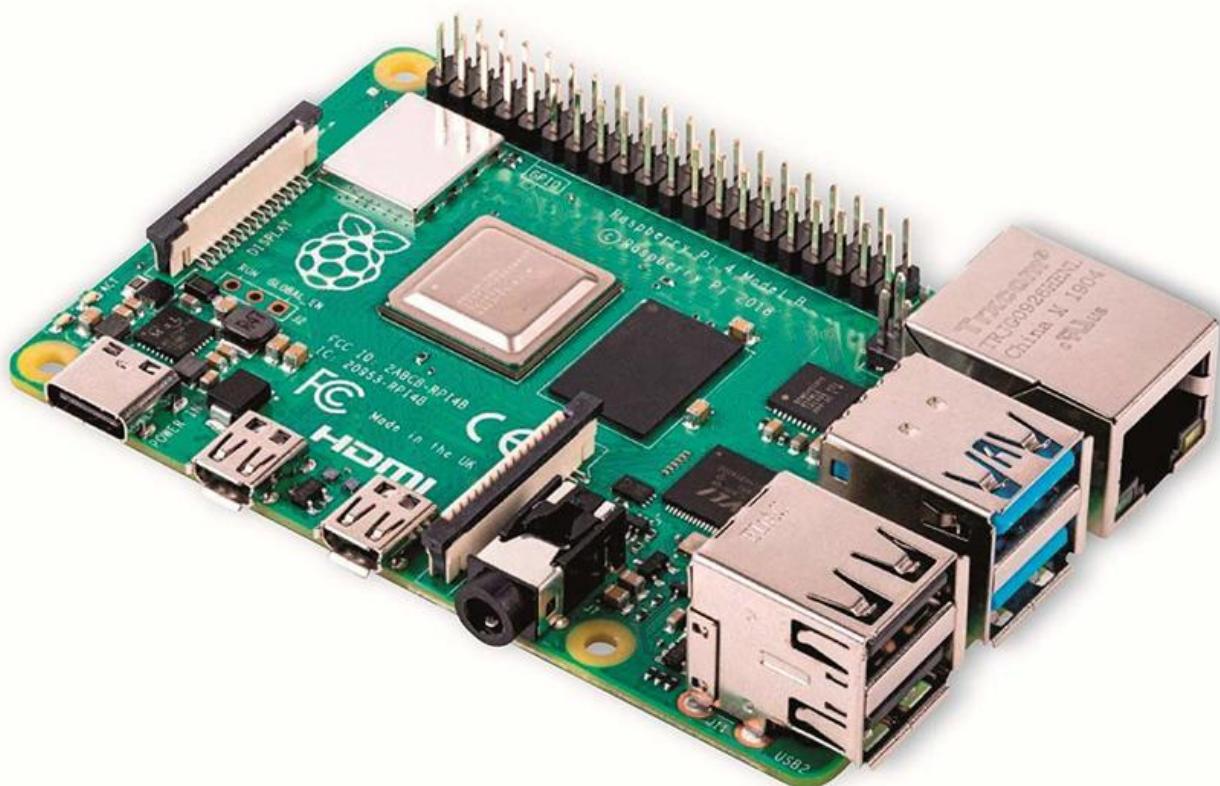
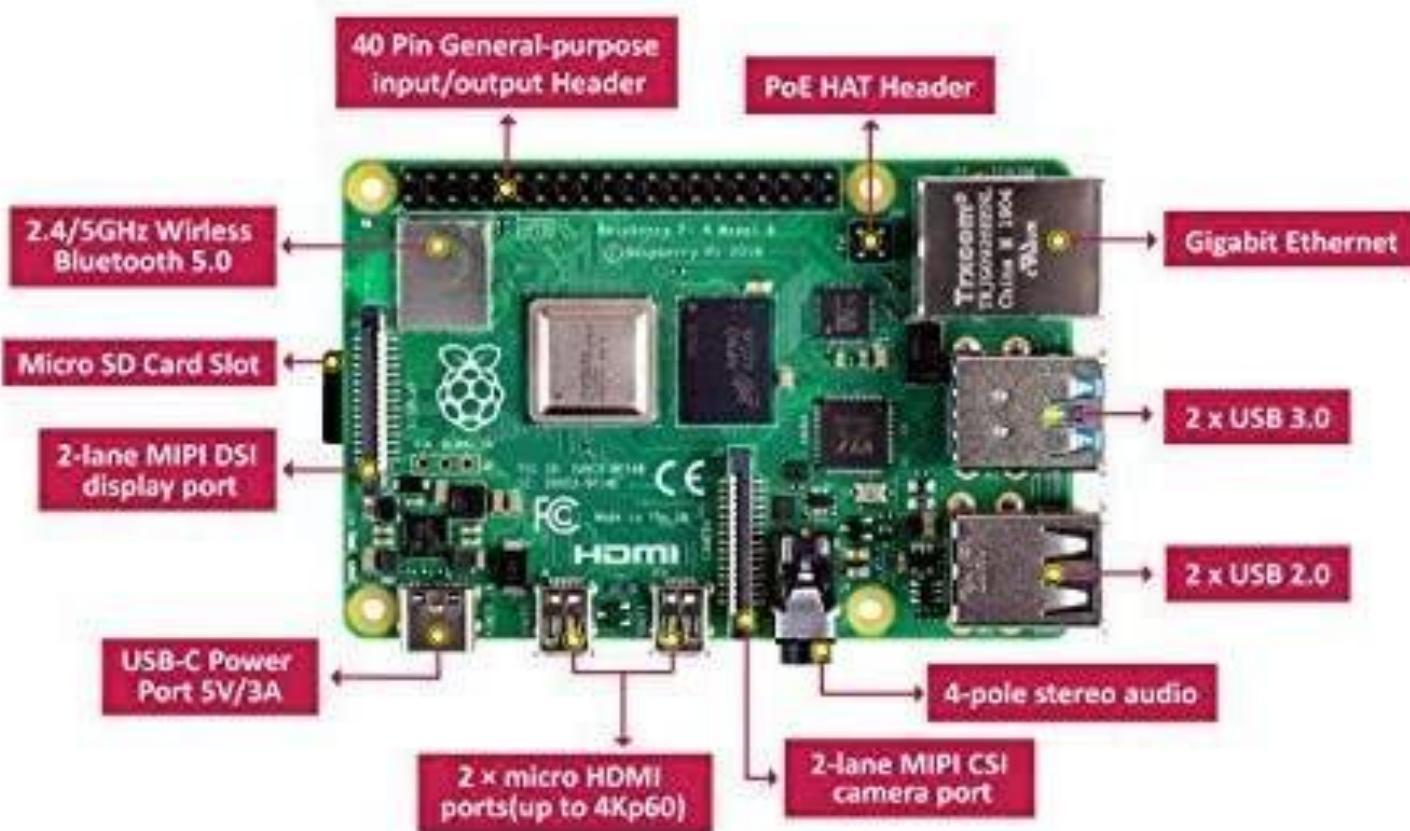
1.- CONFIGURACIÓN

FABRICANTE: Raspberry Pi Foundation



	RASPBERRY PI 4 MODEL B	RASPBERRY PI 3 MODEL B+	RASPBERRY PI 3 MODEL B
PROCESADOR	Broadcom BCM2711B0, quad-core Cortex-A72	Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC	Broadcom BCM2837, Cortex-A53 (ARMv8) 64-bit SoC
FRECUENCIA DE RELOJ	1,5 GHz	1,4 GHz	1,2 GHz
GPU	VideoCore VI 500 MHz	VideoCore IV 400 MHz	VideoCore IV 400 MHz
MEMORIA	1/2/4 GB LPDDR4-3200	1GB LPDDR2 SDRAM	1GB LPDDR2 SDRAM
CONECTIVIDAD INALÁMBRICA	Wi-Fi 2,4GHz / 5GHz IEEE 802.11.b/g/n/ac Bluetooth 5.0, BLE	Wi-Fi 2,4GHz / 5GHz IEEE 802.11.b/g/n/ac Bluetooth 4.2, BLE	Wi-Fi 2,4GHz IEEE 802.11.b/g/n Bluetooth 4.1
CONECTIVIDAD DE RED	Gigabit Ethernet	Gigabit Ethernet over USB 2.0 (300 Mbps de máximo teórico)	Fast Ethernet 10/100 Gbps
PUERTOS	GPIO 40 pines 2 x Micro HDMI 2 x USB 2.0 2 x USB 3.0 CSI (cámara Raspberry Pi) DSI (pantalla táctil) Toma auriculares / video compuesto Micro SD USB-C (alimentación) Power-over-Ethernet (PoE)	GPIO 40 pines HDMI 4 x USB 2.0 CSI (cámara Raspberry Pi) DSI (pantalla táctil) Toma auriculares / video compuesto Micro SD Micro USB (alimentación) Power-over-Ethernet (PoE)	GPIO 40 pines HDMI 4 x USB 2.0 CSI (cámara Raspberry Pi) DSI (pantalla táctil) Toma auriculares / video compuesto Micro SD Micro USB (alimentación)
FECHA DE LANZAMIENTO	24/06/2019	14/3/2018	29/2/2016

La Raspberry Pi es una computadora del tamaño de una tarjeta de crédito que puede correr varias distribuciones de Linux y otros sistemas operativos como RISC OS y Windows 10, pudiendo utilizarse en proyectos electrónicos ya que suministra acceso a sus periféricos de bajo nivel y además es capaz de hacer la mayoría de cosas que hace un PC de escritorio, como correr programas de hoja de cálculo, procesadores de palabras o juegos, por ejemplo. Puede conectarse a un monitor o TV con conector HDMI. Pueden conectársele periféricos USB como teclado, mouse, cámaras web y muchos otros. Incorpora 4 puertos USB 2.0, y también pueden expandirse por medio de un hub con alimentación propia.



CARACTERÍSTICAS Raspberry Pi 3:

- SoC (System on a chip) Broadcom BCM2837 (CPU + GPU + DSP + SDRAM + puerto USB)
- CPU ARM Cortex A53 de 1.2 GHz, con aritmética de punto flotante, arquitectura de 64 bits
- GPU VideoCore IV. Soporta OpenGL ES 2.0, MPEG-2 y VC-1 (Comprando la licencia), 1080p30 H.264/MPEG-4 AVC
- Salidas de video HDMI, video compuesto por jack de 3.5 mm y DSI (Para paneles LCD). Puede conectarse a un puerto DVI con un conversor HDMI a DVI, o a VGA con un adaptador activo (Los pasivos usualmente no funcionan)
- RAM 1 GB compartida con la memoria de video de la GPU
- Salida de audio estéreo por conector estándar de 3.5 mm y por HDMI. Interface I²S
- Ranura para microSD para almacenar el sistema operativo y programas (No incluida). Se recomienda una tarjeta mínima clase 4 de al menos 4 GB
- WiFi 802.11.b/g/n con chip BCM43143
- Bluetooth 4.1 Low Energy (BLE)
- Ethernet 10/100 con conector RJ45
- Entrada de video con conector MIPI CSI
- 4 puertos USB 2.0 (Conectores tipo A hembra). Pueden ser expandidos mediante hub con alimentación propia
- 27 pines entrada/salida de uso general (GPIO) + UART + Bus I²C + Bus SPI
- **Conecotor USB micro B hembra para alimentación únicamente, mediante adaptadores USB**
- **Fuente recomendada: 5 V, 1.8 A mínimo**
- Alimentación por puerto USB micro B o conector de pines GPIO
- **Voltaje máximo en cualquier entrada / salida: 3.3 V DC**
- **Corriente máxima por pin GPIO: 16 mA**
- **Corriente máxima total en los pines GPIO: 100 mA** (La suma de la corriente de todas las salidas en cualquier momento no debe superar este valor)
- La tarjeta puede utilizar un máximo de 2.4 A (Entre consumo propio y periféricos). Si un periférico requiere que la corriente exceda este valor, se debe utilizar con un hub USB que posea alimentación propia (Hub activo)
- Sistemas operativos soportados: Debian (Raspbian), Fedora (Pidora), Arch Linux (Arch Linux ARM), Slackware Linux, Xbian, QtonPi, RISC OS, Windows 10 y algunos otros
- Lenguajes de programación: Python, C, Java, Perl, BBC BASIC, entre otros
- Dimensiones: 8.56 cm x 5.6 cm x 1.7 cm aprox.

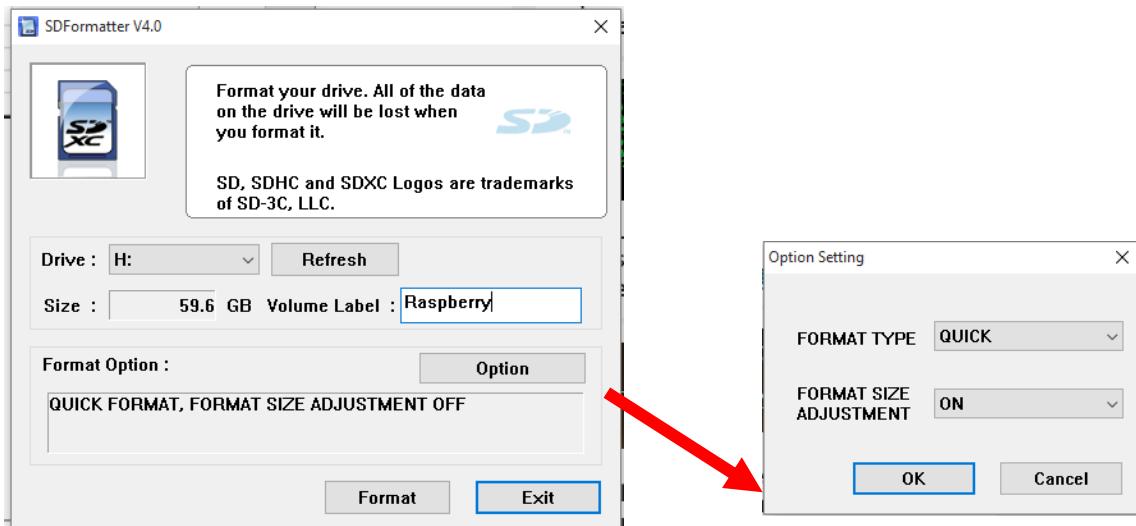
Características de la Raspberry Pi 4

- Broadcom BCM2711, Cortex núcleo cuádruple-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- SDRAM LPDDR4-2400 de 1 GB, 2 GB, 4 GB y 8 GB (según el modelo)
- 2,4 GHz y 5,0 GHz IEEE 802.11ac inalámbrico, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 puertos USB 3.0; 2 puertos USB 2.0.
- Cabezal GPIO estándar de 40 pines de Raspberry Pi (totalmente compatible con las placas anteriores)
- 2 x puertos micro-HDMI (soportados hasta 4kp60)
- Puerto de pantalla MIPI DSI de 2 vías
- Puerto de cámara MIPI CSI de 2 carriles
- Puerto de audio estéreo de 4 polos y de vídeo compuesto
- H.265 (decodificación 4kp60), H264 (decodificación 1080p60, decodificación 1080p30)
- Gráficos OpenGL ES 3.0
- Ranura para tarjetas Micro-SD para cargar el sistema operativo y el almacenamiento de datos
- 5V DC a través de conector USB-C (mínimo 3A*)
- 5V DC vía cabezal GPIO (mínimo 3A*)
- Alimentación a través de Ethernet (PoE) habilitada (requiere PoE HAT separado)
- Temperatura de funcionamiento: 0 – 50 grados C ambiente

PREPARACIÓN

OBSOLETO

1. Formatear SD (8/16/32 Gb) con “SDFormatter”



SD vs SDHC

La gran diferencia entre ambos tipos de tarjeta es que las SD miden su velocidad de escritura de datos en base a la velocidad máxima. Las de 40x escriben a 6 MB por segundo, las de 80x a 12 MB por segundo, las de 100x a 15 MB por segundo, etc. No obstante, hablamos de velocidad máxima. Si queremos asegurarnos de una calidad mínima, nos interesa pasar a **SDHC**, cuya clase mide la **velocidad mínima de escritura**. Así, una clase 2 escribe a 2 MB por segundo como mínimo, una clase 6 a 6 MB por segundo, y una clase 10 a 10 MB por segundo.



En primer lugar, lo que has de mirar es el dispositivo en el que irá la tarjeta SD. No se parte de la misma base con una cámara compacta que dispara únicamente en JPG que de una DSLR de 18 megapíxeles que dispara en RAW y que graba vídeo Full HD. Si hablamos de una **cámara de entrada de gama, con una velocidad de escritura baja o media debería ser más que suficiente**. Conforme aumente la calidad del sensor de nuestra cámara, aumentará la necesidad de una SDHC. Si pensamos grabar vídeo **Full HD**, definitivamente necesitaremos una SDHC clase 10 para asegurarnos de que el vídeo se grabará de forma fluida y sin micro cortes. Algo similar si

usaremos nuestra tarjeta SD o SDHC como herramienta para transportar información entre dispositivos: si vamos a mover grandes volúmenes de archivos, mejor ahorrar tiempo a lo largo de muchos usos que no algo de dinero y acabar arrepintiéndonos por la lentitud.

No obstante, **no todos los dispositivos aceptan tarjetas SDHC**, deberemos fijarnos en que tenga el logotipo para asegurarnos de su compatibilidad. O incluso en el caso de que lo lleven, es posible que acepten este formato, pero con un máximo de 4 GB de almacenamiento.

2. Descargar **NOOBS v*.*** y, tras descomprimirlo, copiarlo en la SD
<https://www.raspberrypi.org/downloads/noobs/>



3. Conectar un monitor/TV, un ratón, un teclado. Iniciar la RPi, seleccionar **RASPBIAN** e instalarlo.
 4. Configurar el sistema. Tras estos pasos debiera de abrirse automáticamente, de no ser así se teclea “**sudo raspi-config**”

Usuario: **pi**

Contraseña: **raspberry**

- [La siguiente versión de Debian se llama «buster»](#) — no se ha decidido fecha de publicación aún.
- [Debian 9 \(«stretch»\)](#) — versión estable actual.
- [Debian 8 \(«jessie»\)](#) — antigua versión estable.
- [Debian 7 \(«wheezy»\)](#) — antigua versión estable.
- [Debian 6.0 \(«squeeze»\)](#) — antigua versión estable.
- [Debian GNU/Linux 5.0 \(«lenny»\)](#) — antigua versión estable.
- [Debian GNU/Linux 4.0 \(«etch»\)](#) — antigua versión estable.
- [Debian GNU/Linux 3.1 \(«sarge»\)](#) — antigua versión estable.
- [Debian GNU/Linux 3.0 \(«woody»\)](#) — antigua versión estable.
- [Debian GNU/Linux 2.2 \(«potato»\)](#) — antigua versión estable.
- [Debian GNU/Linux 2.1 \(«slink»\)](#) — antigua versión estable.
- [Debian GNU/Linux 2.0 \(«hamm»\)](#) — antigua versión estable.

TARJETAS > 32GB (desde WINDOWS)

OBSOLETO

De acuerdo con las especificaciones **SD**, cualquier tarjeta **SD** de más de **32 GB** es una tarjeta **SDXC** y tiene que formatearse con el sistema de archivos **exFAT**. Esto significa que la herramienta **SD Formatter** oficial, siempre formateará tarjetas que tengan **64 GB** o más como **exFAT**.

El cargador de arranque de Raspberry Pi, integrado en la GPU y no actualizable, sólo tiene soporte para la lectura de sistemas de archivos **FAT (FAT16 y FAT32)** y no puede arrancar desde un sistema de archivos **exFAT**. Por lo tanto, si desea utilizar **NOOBS** en una tarjeta de **64 GB** o más, debe reformatearla como **FAT32** primero antes de copiar los archivos NOOBS.

Las herramientas de formateo estándar incorporadas en **Windows** son limitadas, ya que sólo permiten que las particiones de hasta **32GB** sean formateadas como **FAT32**, por lo que para formatear una partición de **64GB** como **FAT32** necesitarás utilizar una herramienta de formateo de terceros.

Una simple herramienta para hacer esto es “**FAT32 Format**”, que se descarga como un solo archivo denominado “**fat32format.exe**” (no es necesario instalarlo).

Primero ejecutamos la herramienta “**SD Formatter**” con “**FORMAT SIZE ADJUSTMENT**” ajustado a “**ON**”, para asegurarse de que cualquier otra partición en la tarjeta **SD** sea borrada.

A continuación, desde línea de comandos, ejecutamos la herramienta “**FAT32 Format**” (fat32format.exe):
C:\FAT32FORMAT> fat32format e:
 (asegúrese de elegir la letra de unidad correcta)

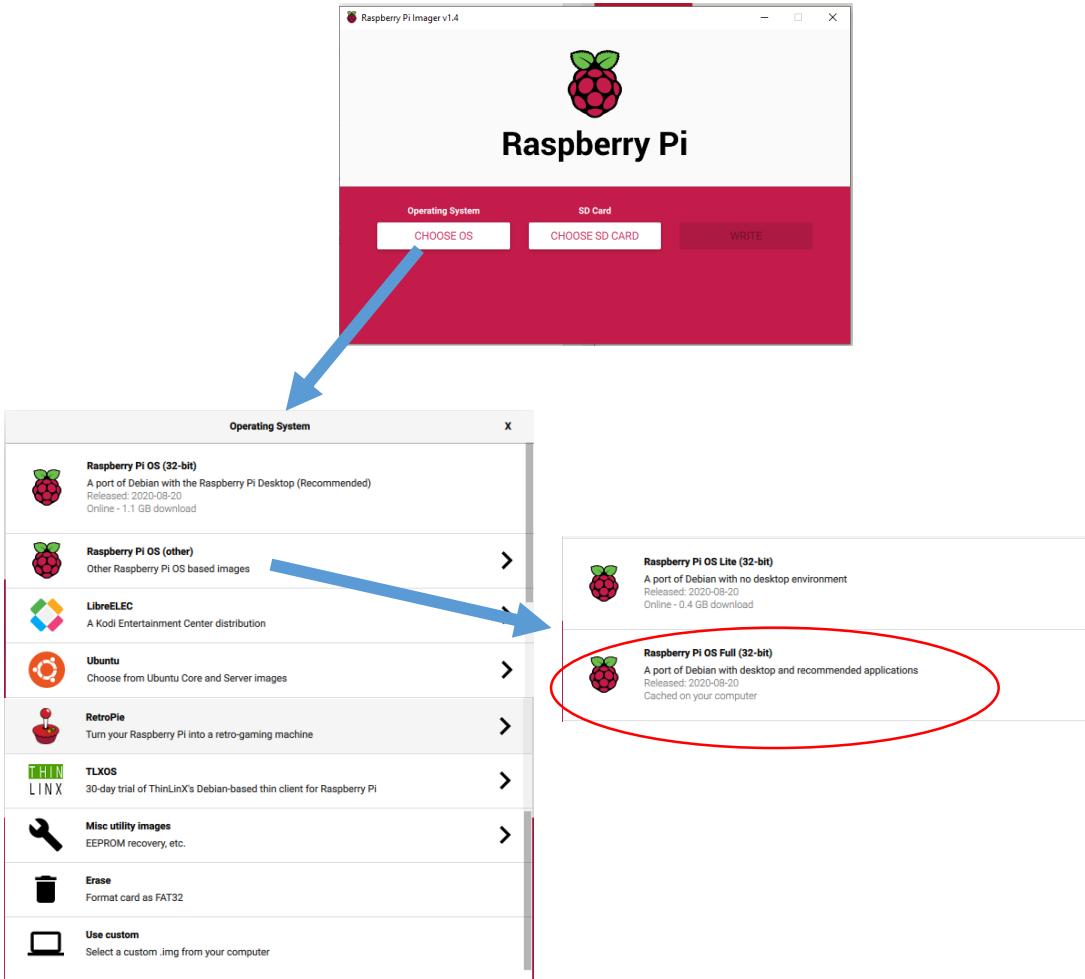
Después de que haya terminado, puede continuar con el resto de las instrucciones de **NOOBS**, explicadas anteriormente.

PREPARACIÓN

13/10/2020

A 13/10/2020, tal y como se explica en la WEB de NOOBS, la instalación del S.O. en la Raspberry se hará del siguiente modo:

1. Descargamos el programa **Raspberry Pi Image** <https://www.raspberrypi.org/downloads/>
2. Tras instalarlo en Windows, lo ejecutamos y seleccionamos el software a instalar, así como la SD donde instalarlo.



3. Tras la grabación, insertamos la SD en la raspi y seguimos los pasos que se nos indique

El nuevo S.O. de Raspbian, pasa a llamarse → **Raspberry Pi OS**

EL SISTEMA GNU/LINUX

La Raspberry Pi cuenta con un completo sistema operativo, con entorno gráfico y herramientas de programación de diverso tipo. Vamos a utilizar este entorno para realizar la mayor parte del taller. Sin embargo, debemos precisar que la forma de trabajo habitual en desarrollo de sistemas empotrados es que se utilice un PC y programemos la Raspberry Pi remotamente.

GNU/Linux es el nombre habitual del sistema operativo que lleva la Raspberry. **Raspbian** y **Debian** no son más que distribuciones de este sistema operativo. Es decir, **Raspbian** es una selección de paquetes de **GNU/Linux**, compilados para una arquitectura concreta, y empaquetados con ayuda de herramientas específicas para conseguir una experiencia de usuario agradable. En lugar de ir aquí y allá en busca de instaladores y drivers como hacemos en Microsoft Windows, en GNU se especializan en conjuntos de paquetes con fines específicos.

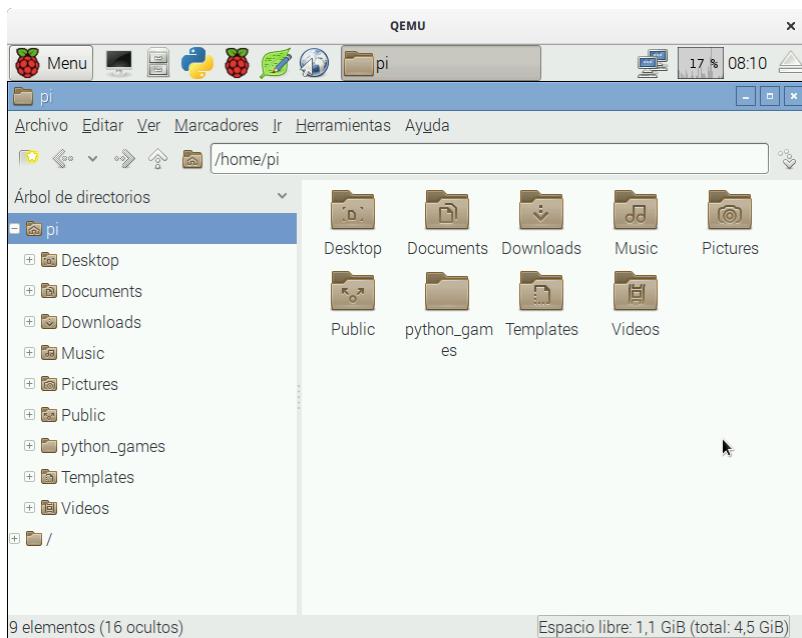
GNU es el nombre correcto del sistema operativo. Quiere decir **GNU's Not Unix**, es decir, **GNU no es Unix**. Es un acrónimo recursivo. Hace referencia a que no contiene ni una sola línea de Unix, el sistema operativo privativo de AT&T, que luego vendió a SCO y licenció a IBM, Sun, HP, Silicon Graphics, Fujitsu, Microsoft, etc. El sufijo **Linux** se refiere al **kernel** (núcleo) del sistema operativo. **GNU** tiene su propio **kernel**, el **HURD**, pero todavía no está listo para su uso general. Por eso la mayoría de las distribuciones añaden a GNU alguno de los kernels libres que hay por ahí (**Linux**, **FreeBSD**, **NetBSD**, etc.)

Nada más conectar la Raspberry Pi a la alimentación arrancará en un entorno gráfico como el que se muestra en la figura al comienzo del capítulo. En la parte superior aparecen los siguientes elementos.

-  **Menu** Menú de aplicaciones.
-  Terminal de línea de órdenes.
-  Herramienta de configuración de Raspberry Pi.
-  Herramienta de administración de archivos.
-  Entorno integrado de desarrollo en Python (IDLE).
-  Editor de textos.
-  Navegador básico de web.

Desde el menú es posible ejecutar la mayoría de las aplicaciones instaladas. No obstante, con los botones de lanzamiento rápido de aplicaciones tendremos suficiente para la mayoría de las actividades del taller.

El sistema de archivos



La caja de texto en la parte superior indica `/home/pi` que es la carpeta actual. Las rutas de los archivos y las carpetas utilizan el carácter `/` como separador.

La carpeta `/` sin más es la carpeta **raíz**, de donde cuelga todo.

Aquí **no hay nombres de unidades**, todas las unidades se ven en algún punto del árbol de carpetas que nace en la carpeta raíz.

La ruta `/home/pi` hace referencia a que se encuentra en la carpeta `pi` de la carpeta `home`. Como puedes imaginar se trata de la carpeta personal. El nombre `home` se refiere a que contiene todas las carpetas personales (casa en inglés). Y dentro de esa carpeta, la carpeta `pi` es la del usuario `pi`.

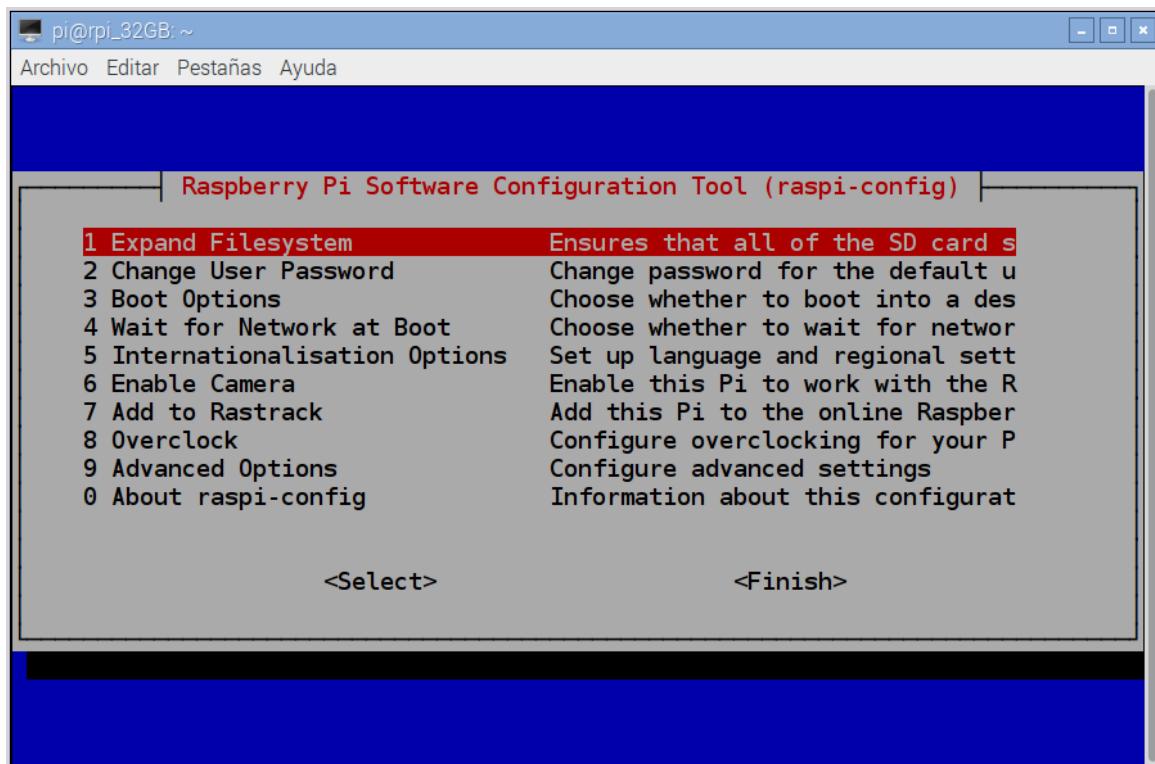
Efectivamente, `pi` es el nombre del usuario creado por defecto en el sistema cuando se instala.

- `/home/` Carpetas personales de los usuarios.
- `/root/` Carpeta personal del administrador (usuario `root`).
- `/etc/` Archivos de configuración del sistema.
- `/boot/` Archivos necesarios para el arranque del sistema.
- `/bin/` Órdenes básicas (ejecutables del sistema).
- `/usr/bin/` Resto de órdenes del sistema (ejecutables).
- `/lib/` Bibliotecas básicas del sistema (biblioteca en inglés es *library*).
- `/usr/lib/` Resto de bibliotecas del sistema.
- `/usr/local/` Software instalado de forma manual, no perteneciente al sistema.
- `/tmp/` Carpeta temporal.
- `/dev/` Dispositivos del sistema. En GNU todos los dispositivos se ven como archivos especiales.

RASPI-CONFIG

OBSOLETO

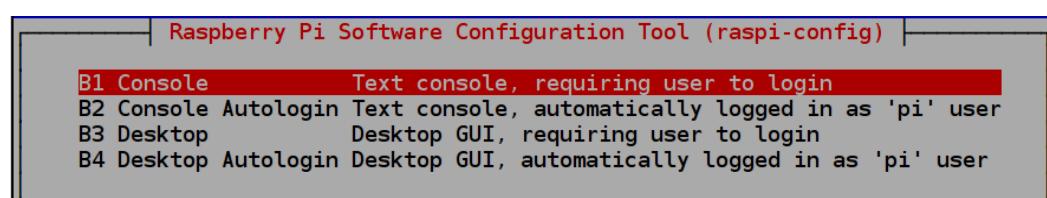
La primera vez que arrancamos la **Raspberry Pi** con **Raspbian**, debería iniciarse una pantalla azul tipo MSDOS, BIOS o Terminal como la siguiente.

**Expand Filesystem**

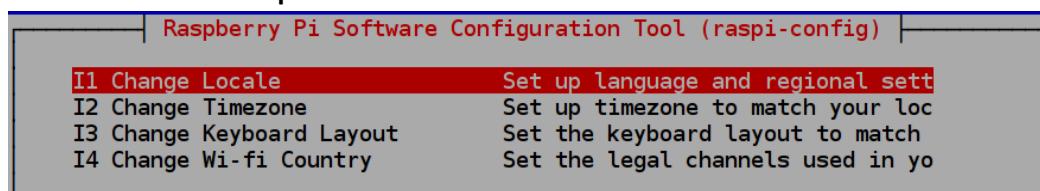
La tarjeta SD tendrá dos particiones: una de **boot** (arranque) que hace las veces de BIOS y que es visible poniendo la tarjeta en un PC con Windows o Mac y otra principal, donde está instalado Raspbian y que solo es visible en un PC con Linux. Como las imágenes por defecto que se usan como base para instalar Raspbian son de **2GB**, si hemos usado una SD más grande el resto del espacio estará sin utilizar. Al usar esta opción del menú de configuración haremos que Raspbian aproveche todo el espacio restante de la tarjeta.

Change User Password

Para cambiar la contraseña de fábrica. Por defecto es: user '**pi**' y pass '**raspberry**'.

Boot Options**Wait for Network at Boot**

Habilita la posibilidad de que el arranque espere hasta establecer una conexión de red.

Internationalisation Options

I1 Change Locale

Se utiliza para seleccionar el idioma, el conjunto de caracteres asociado, la moneda, etc.

Para utilizar el español tendrás que elegir **es_ES.UTF8**.

I2 Change Timezone

Nuestra Raspberry Pi está configurada para detectar la fecha y hora desde Internet automáticamente cuando se enciende, pero la primera vez que arranca, le tendremos que indicar la zona horaria en la que nos encontramos. Aquí elegiremos **Europa** y después **Madrid**.

I3 Change Keyboard Layout

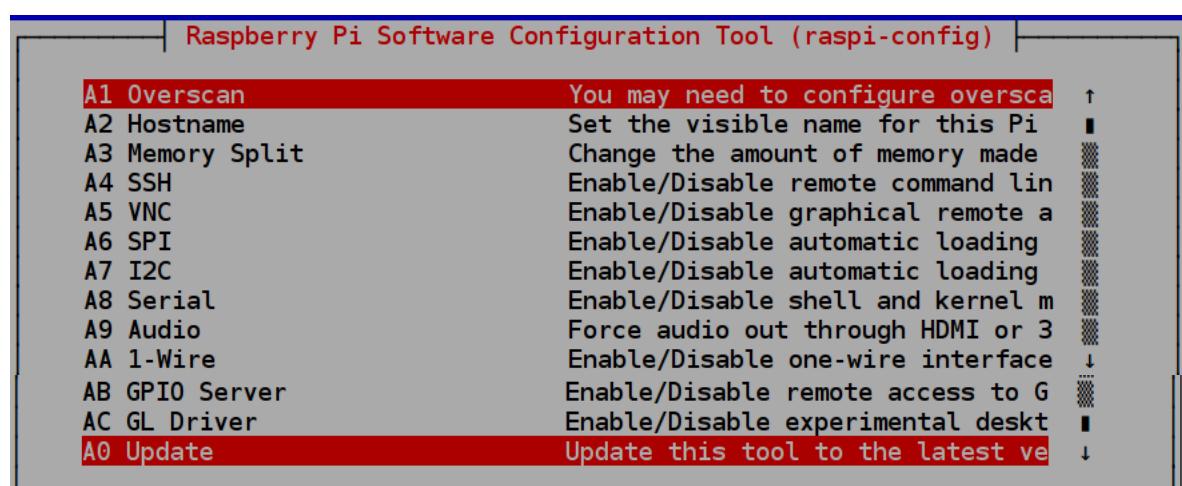
Permite cambiar la configuración del teclado. Es recomendable seleccionar el teclado predeterminado **PC genérico 105 teclas**. Luego seleccionar el idioma del teclado, que en nuestro caso será **Español**.

Enable camera**Add to Rastrack**

Agrega la posición **GPS** de nuestra **Raspberry Pi** a un mapa mundial que se puede consultar en www.rastrack.co.uk. No sirve absolutamente de nada, sólo simple curiosidad.

Overclock

Ten en cuenta que modificar la velocidad reduce la vida del dispositivo considerablemente. Por otra parte, el dispositivo generará más calor, por lo tanto, es recomendable tener disipadores que ayuden a rebajar la temperatura.

Avanced Options**A1 Overscan**

Sirve para borrar las líneas negras que aparecen en algunos monitores o televisores.

A2 Hostname

Es el nombre de la **Raspberry Pi** en la red.

A3 Memory Split

Te permite seleccionar la cantidad de **memoria compartida entre la CPU y la GPU**. Si vas a utilizar la **Raspberry Pi** como Media Center, tal vez tengas que modificar esta opción para darle mayor fluidez a la reproducción de vídeo.

A4 SSH

Se utiliza para acceder a la **Raspberry Pi remotamente** desde un cliente SSH. SSH significa Secure SHell el cual es una forma segura de conectarse a través de la red a la **Raspberry Pi**. Es recomendable activar esta opción, ya que así no será necesario utilizar monitor ni teclado para controlar a la **Raspberry Pi** y lo podremos hacer remotamente.

A5 VNC

Activa el protocolo de control remoto VNC.

A6 SPI

Sirve para activar el uso de circuitos integrados con nuestra **Raspberry Pi**, mediante el bus de comunicaciones SPI.

A7 I2C

Sirve para activar el uso de circuitos integrados con nuestra **Raspberry Pi**, mediante el bus de comunicaciones I2C.

A8 serial

Las Raspberry incluyen una serie de pines de propósito general que pueden ser utilizados como entradas o salidas digitales entre los cuales se incluye un pequeño puerto "serial". Este puerto puede ser utilizado para enviar o recibir datos desde y hacia otros dispositivos. De fábrica viene configurado como un puerto de "consola" para monitorear el funcionamiento del RasPI.

A9 Audio

Opción para activar la salida de audio. No ese necesario tocarla sobre todo si vamos a usar HDMI.

AA 1-Wire

Sirve para activar el uso de circuitos integrados con nuestra **Raspberry Pi**, mediante el bus de comunicaciones 1-Wire.

AB GPIO Server

Habilita /Deshabilita el acceso remoto a los pines GPIO.

AC GL Driver

Habilita/Deshabilita el driver experimental GL de la tarjeta gráfica.

A0 Update

Sirve para actualizar el sistema, y si hay nuevas versiones de las librerías o programas instalados se descargarán e instalarán las últimas versiones.

ESCRITORIO REMOTO

OBSOLETO

- Actualizar el sistema **\$ sudo apt-get update**
- Para usar la aplicación CONEXIÓN A ESCRITORIO REMOTO
 - Instalar la aplicación **\$ sudo apt-get install xrdp**
 - En las últimas versiones de Raspbian viene preinstalado el RealVNC y entra en conflicto con xrdp y tightvncserver. Por lo que habrá que instalar vnc4server y luego reiniciar el servicio xrdp:
\$ sudo apt-get install vnc4server
\$ sudo service xrdp restart
- Si al hacer control remoto, el teclado está en inglés...
 - Hacemos copia de seguridad
\$ sudo cp /etc/xrdp/km-0409.ini /etc/xrdp/km-0409.old
 - Abrimos el archivo, lo borramos entero y copiamos el contenido que se nos sugiere en <http://www.pacorabadan.com/?p=283>
\$ sudo nano /etc/xrdp/km-0409.ini
 - Para recargar la configuración: **\$ sudo /etc/init.d/xrdp restart**
- Instalamos **winSCP** en el PC y utilizamos el protocolo SCP para hacer control remoto

VNC

VNC SERVER EN VERSIONES DE RASPBIAN STRECH

- Viene instalado por defecto
- Habilita el VNC en “Preferences-Raspberry Pi Configuration” o desde el bash con “sudo raspi-config”. Desde el cliente ejecutas vncviewer con la dirección de la raspi y el puerto 5900.
- Si cuando conectas con la raspberry, el escritorio sale muy pequeño puedes realizar lo siguiente:

OBSOLETO

Abrimos una ventana de terminal y escribimos el siguiente comando

\$sudo nano /boot/config.txt

se abrirá el fichero de configuración y lo modificaremos ligeramente, buscad las líneas que hacen referencia a hdmi

```
#hdmi_force_hotplug=1
#hdmi_group=1
#hdmi_mode=1
```

estas líneas tienen delante el símbolo # (significan que están comentadas, es decir, que se omiten al arrancar) cámbialas por

```
hdmi_force_hotplug=1
hdmi_group=2
hdmi_mode=82
```

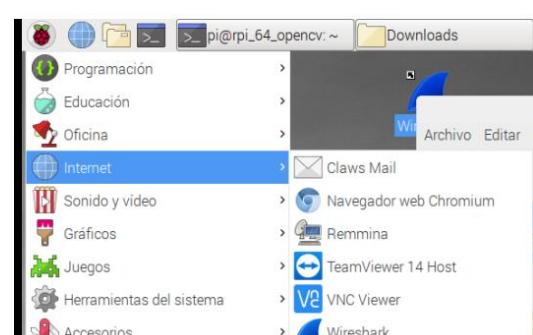
INSTALAR VNC VIEWER

Descargamos el archivo: <https://www.realvnc.com/es/connect/download/viewer/raspberrypi/>

Ejecutamos:

\$sudo dpkg -I VNC-Viewer-6.19.923-Linux-ARM.deb

Queda instalado en la carpeta de internet



VNC EN LAS VERSIONES DE RASPBIAN JESSIE O WHEEZY**OBSOLETO**

- Para usar **VNC**
 - Instalar el paquete **\$ sudo apt-get install tightvncserver**
 - Lanzar el servicio **\$ vncserver :1 -geometry 1920x1080**
 - Para cerrar el servicio **\$ vncserver -kill :1**
 - Con **tightvncserver**, desde el cliente utilizaremos el puerto **5901**.
- Hacer que VNC se lance al iniciar la Raspberry:
 1. Añadiremos en el fichero **\$ /etc/rc.local** la siguiente línea antes del "exit 0": para entrar como pi desde VNC
su -c "/usr/bin/tightvncserver -geometry 1920x1080 -depth 8" pi
 2. Creamos un fichero script en /etc/init.d, (por ejemplo, lo llamamos "**vnc**") con el siguiente contenido:

```
#!/usr/bin/bash
sudo /etc/rc.local
```
 3. Ejecutamos el siguiente comando para que lo ejecute al arrancar: **\$ sudo update-rc.d vnc**
Nota.- Si tras reiniciar la raspberry, con monitor, nos mostrara una nueva ventana de inicio, debido al nuevo servicio de VNC, nos encontraríamos con el problema de que el usuario que espera la raspi es el "root", del que no sabemos su clave. Para cambiar la clave del root:
 - Hacemos control remoto y ponemos **\$ sudo su**
 - Luego **\$ passwd**
- Instalación del cliente VNC
\$ apt-get install xtightvncviewer → Seguidamente ejecutamos: **\$ xtightvncviewer**

TEAMVIEWER<http://www.teamviewer.com/iotcontest>

Ahora que tienes el archivo **.deb** disponible en la Raspberry Pi hay que instalarlo ejecutando el siguiente comando.

```
sudo dpkg -i teamviewer-host_14.6.2452_armhf.deb
```

Al ejecutar el comando anterior, notarás errores sobre paquetes específicos que no se están instalando, para solucionar este problema, tendrás que utilizar el gestor paquetes para reparar la "instalación rota".

El administrador de paquetes **apt** detectará automáticamente los paquetes fallidos e intentará descargar las mejores versiones disponibles para el software.

Escribe esta línea en el terminal de comandos para que el gestor de paquetes solucione los problemas.

```
sudo apt --fix-broken install
```

Ahora TeamViewer debería funcionar en tu Raspberry Pi. El software se configura automáticamente para que se inicie durante el arranque de la placa.

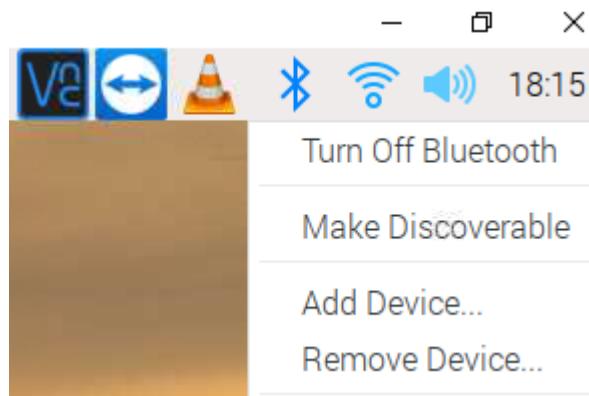
INSTALAR IMPRESORA

- Para instalar una impresora <http://rooteear.com/ubuntu-linux/impresora-en-raspberry-pi#>
- Para descomprimir/comprimir, instalamos el gestor de archivos FILLE ROLLER
\$ sudo apt-get install fille-roller
- Instalar lector PDF
 - Instalamos OKULAR **\$ sudo apt-get install okular**

ACTIVAR AUDIO

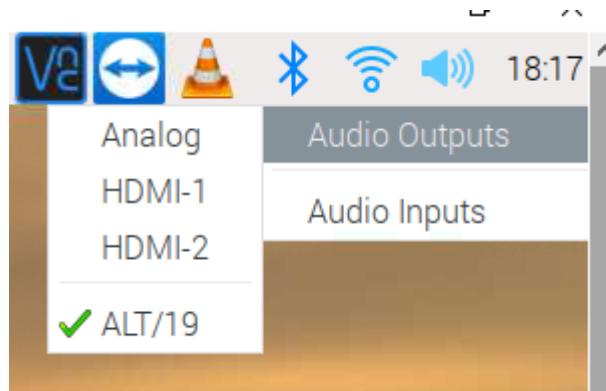
- Para activar el audio: `$ sudo raspi-config`
 - Elegimos: Advanced Options -> Audio -> Force 3.5mm ('headphone') jack
- Para lanzar un archivo de audio ejecutaremos: `$ omxplayer nombre_archivo`
- Podemos forzar la salida de audio por los conectores HDMI o JACK:
 - HDMI → `$ omxplayer -o hdmi nombre_archivo`
 - JACK → `$ omxplayer -o local nombre_archivo`

CONEXIÓN A UN ALTAVOZ BLUETOOTH



Hacemos clic en “**Add Device**”. Al cabo de unos segundos veremos el dispositivo en la lista, lo seleccionamos y pulsamos el botón **Pair**.

Una vez emparejados, seleccionamos el dispositivo de salida de audio.



2.- RED

OBSOLETO

- Configurar el WIFI para que inicie automáticamente
 - \$ sudo nano /etc/network/interfaces**

1º método	2º método
<pre>auto lo iface lo inet loopback auto eth0 iface eth0 inet dhcp auto wlan0 allow-hotplug wlan0 iface wlan0 inet static address 192.168.43.38 netmask 255.255.255.0 gateway 192.168.43.1 wpa-ssid "Aquaris E5" wpa-psk "11111111"</pre>	<pre>auto eth0 iface eth0 inet dhcp auto wlan0 iface wlan0 inet dhcp wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf</pre> <p>El fichero wpa_supplicant.conf queda así:</p> <pre>ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev country=ES update_config=1 network={ ssid="The_ESSID_from_earlier" psk="Your_wifi_password" key_mgmt=WPA-PSK }</pre>

- Comprobar las conexiones de red con **\$ ifconfig**
- Algunas veces el internet se niega a ir. Basta con editar el archivo resolv.conf

\$ sudo nano /etc/resolv.conf

Y añadir como DNS la Gateway del router → nameserver 208.67.222.222

También lo podemos añadir en el archivo interfaces:

dns-nameservers 8.8.8.8 9.9.9.9

- Detener la interfaz de red -> **\$ sudo ifdown eth0 ó \$ sudo ifconfig eth0 down**
- Iniciar la interfaz de red -> **\$ sudo ifup eth0 ó \$ sudo ifconfig wlan0 up**
- Reiniciar TODOS los servicios de red -> **\$ sudo /etc/init.d/networking restart**
- Puerta de enlace -> **\$ route -n**
- Servidor DNS -> **\$ cat /etc/resolv.conf**
- Averiguar la MAC -> **\$ arp**

Utilizar este

3º método con el nuevo Raspbian

Fichero interfaces:

```
# interfaces(5) file used by ifup(8) and ifdown(8)
# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'
# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d
```

El fichero wpa_supplicant.conf queda así:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
country=ES
update_config=1

network={
  ssid="The_ESSID_from_earlier"
  psk="Your_wifi_password"
  key_mgmt=WPA-PSK
}
```

4º si la red está oculta

Fichero interfaces:
source-directory /etc/network/interfaces.d

El fichero wpa_supplicant.conf queda así:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
country=ES
update_config=1

network={
  ssid="The_ESSID_from_earlier"
  scan_ssid=1
  psk="Your_wifi_password"
  mode=0
  proto=WPA2
  key_mgmt=WPA-PSK
  pairwise=CCMP
  group=CCMP
  auth_alg=OPEN
  id_str="raspi"
  priority=1
}
```

PROBLEMA CON LAS IP's

En las últimas versiones de Raspbian para establecer una IP estática, el viejo método que modifica el archivo “**/etc/network/interfaces**” no es tan eficiente como antes. De hecho, si modifica las interfaces configurando **eth0** a **static** en lugar de **manual** (el ajuste predeterminado) su Raspberry Pi obtendrá **dos direcciones IP** para la misma interfaz eth0.

El "problema" es el **daemon dhcpcd**, que es un cliente DHCP que parece estar ejecutado antes de analizar el archivo “**/etc/network/interfaces**”.

Así que tenemos 2 opciones:

1. Setting como antes en “**/etc/network/interfaces**”, deshabilitando el daemon dhcpcd, con el comando **\$sudo update-rc.d -f dhcpcd remove** (puedes volver atrás con **\$sudo update-rc.d dhcpcd defaults**).
2. Forzar el **daemon dhcpcd** para obtener la IP que te gusta.

Para ello, en “**/etc/dhcpcd.conf**”, agregue lo siguiente:

```
# Custom static IP address for eth0.
interface eth0
static ip_address=192.168.1.200/24
static routers=192.168.1.1
static domain_name_servers=192.168.1.1
```

Después de esta modificación, hay que reiniciar la interfaz de red o simplemente reiniciar la Raspi.

- **\$ ping 192.168.10.1 -c 4**
 - **\$ traceroute**
 - Traceroute es una excelente herramienta de diagnóstico pues permite mostrar todos los hosts por donde pasa un paquete en la red hasta llegar a su destino. Su sintaxis es bastante simple, si queremos ver la lista de rutas seguidas por nuestros paquetes para llegar a un servidor ejecutamos:
 - **\$ traceroute servidor**
 - **\$ host**
 - le damos el nombre de un dominio y veremos la IP asociada al mismo, o le damos una IP y veremos el nombre de dominio asociado.
 - **\$ ifplugstatus → (\$ sudo apt-get install ifplugd)**
 - nos dice si un interface de red tiene el cable conectado
 - **\$ netstat -tanp**
 - Netstat es una utilidad que permite mostrar todas las conexiones de red en un sistema. Se entienden como conexiones de red los sockets tcp, udp y unix, tanto conectados como en espera de conexión.
 - **-t**: Sólo muestra conexiones TCP.
 - **-a**: Muestra también los puertos que estamos escuchando
 - **-n**: No resuelve las direcciones. (Es mucho más rápido, pero veremos sólo IPs y no nombres)
 - **-p**: Muestra los procesos que originan dichas conexiones. (pid y nombre, si es posible)
- Por supuesto, la salida de **netstat** la podemos pasar por **grep**, para realizar un filtrado de lo que nos interesa.
- Ejem.:

```
$ netstat -tanp | grep 40008
tcp      0      0 192.168.0.7:42090          157.55.235.157:40008      ESTABLECIDO 1992 skype
```

- Usando **Fing** para descubrir la LAN con Raspberry PI
 - 1. Lo descargamos de la web
 \$ **https://www.fing.com/products/development-toolkit**
 - 2. Lo descomprimimos y pasamos a la raspberry, mediante winscp, el fichero:
 - **fing-5.5.2-armhf.deb**
 - 3. Install it
 \$ **sudo dpkg -i fing-5.5.2-armhf.deb**
 - 3. Use it
 - a) Discover infinite rounds
 - \$ **sudo fing -r 1**
 - b) Discover certain range with 1 round
 - \$ **sudo fing -d 192.168.1.0/24 -r 1**
(-d Means discovery network from 192.168.1.0 till 192.168.1.24 & -r means round)
 - c) Scan for services
 \$ **sudo fing -s 192.168.1.10**
 - d) Scan for services from a domain
 \$ **sudo fing -s www.example.com**
 - e) Scan for services from a domain with certain port
 \$ **sudo fing -s www.example.com/80**
 - f) Output the scanned result (Output to text log and on network.txt file)
 \$ **sudo fing -h host -o text, network.txt**
- Instalar Firefox → \$ **sudo apt-get install iceweasel**
- Si queremos conocer nuestra IP pública → \$ **curl ifconfig.me/ip**

- **Nmap**

("mapeador de redes") es una herramienta de código abierto para exploración de red y auditoría de seguridad. Se diseñó para analizar rápidamente grandes redes, aunque funciona muy bien contra equipos individuales. Nmap utiliza paquetes IP "crudos" («raw», N. del T.) en formas originales para determinar qué equipos se encuentran disponibles en una red, qué servicios (nombre y versión de la aplicación) ofrecen, qué sistemas operativos (y sus versiones) ejecutan, qué tipo de filtros de paquetes o cortafuegos se están utilizando así como docenas de otras características.

La salida de **Nmap** es un listado de objetivos analizados, con información adicional para cada uno dependiente de las opciones utilizadas. La información primordial es la “tabla de puertos interesantes”. Dicha tabla lista el número de puerto y protocolo, el nombre más común del servicio, y su estado.

El estado puede ser open (abierto), filtered (filtrado), closed (cerrado), o unfiltered (no filtrado). Abierto significa que la aplicación en la máquina destino se encuentra esperando conexiones o paquetes en ese puerto.

Filtrado indica que un cortafuego, filtro, u otro obstáculo en la red está bloqueando el acceso a ese puerto, por lo que **Nmap** no puede saber si se encuentra abierto o cerrado.

Los puertos cerrados no tienen ninguna aplicación escuchando en los mismos, aunque podrían abrirse en cualquier momento.

Los clasificados como no filtrados son aquellos que responden a los sondeos de Nmap, pero para los que Nmap no puede determinar si se encuentran abiertos o cerrados.

Nmap informa de las combinaciones de estado **open/filtered** y **closed/filtered** cuando no puede determinar en cuál de los dos estados está un puerto.

La tabla de puertos también puede incluir detalles de la versión de la aplicación cuando se ha solicitado detección de versiones. **Nmap** ofrece información de los protocolos IP soportados, en vez de puertos abiertos, cuando se solicita un análisis de protocolo IP con la opción (-sO). Además de la tabla de puertos interesantes, **Nmap** puede dar información adicional sobre los objetivos, incluyendo el nombre de DNS según la resolución inversa de la IP, un listado de sistemas operativos posibles, los tipos de dispositivo, y direcciones MAC.

Para instalar Nmap -> \$ sudo apt-get install nmap

- **\$ sudo nmap -sP 192.168.1.0/24**
- **\$ sudo nmap -sn 192.168.1.0/24**

- **WIRESHARK**

Wireshark es una herramienta que funciona como un analizador de protocolos de redes, permitiendo capturar y analizar en tiempo real, de forma interactiva, el tráfico que pasa por una red. Es la herramienta más popular de este tipo. Corre en Windows, Mac, Linux y UNIX. Expertos en seguridad, profesionales en redes y educadores lo usan regularmente. Es software libre, bajo la GNU GPL 2.

Con esta herramienta podremos analizar todos los paquetes de datos que entran y salgan de cualquiera de nuestras interfaces de red (tarjetas Ethernet o Wi-Fi). Se puede ver esta información en tiempo real, y puede ser filtrada en tiempo real también.

Para instalar:

```
$ sudo apt-get install wireshark
```

Ya que los usuarios por defecto no tienen permiso para manejar las interfaces de red directamente, y para evitar el uso de Wireshark como root, se debe hacer este “arreglo” para que un usuario regular en Linux pueda usar la herramienta sin problemas. Primero se deben correr estos comandos en una terminal

```
$ sudo addgroup --quiet --system wireshark (si da error no pasa nada)
```

```
$ sudo chown root:wireshark /usr/bin/dumpcap
```

```
$ sudo setcap cap_net_raw,cap_net_admin=eip /usr/bin/dumpcap
```

Esto lo que hace es crear un nuevo grupo y permitirle el uso del dumpcap (el programa que usa Wireshark por defecto para la captura)

Luego añadimos nuestro usuario al nuevo grupo

```
$ sudo usermod -a -G wireshark pi
```

Y, finalmente, reconfigurar Wireshark para que los no-administradores puedan capturar paquetes:

```
$ sudo dpkg-reconfigure wireshark-common (hay que reiniciar)
```

- **AIRCRACK (Para poner en modo monitor la raspberry)**

Instalamos dependencias:

```
$ sudo apt-get -y install libssl-dev
```

Descargar Aircrack-ng de la siguiente manera:

```
$ wget http://download.aircrack-ng.org/aircrack-ng-1.2-beta1.tar.gz
```

Para descomprimir y desempaquetar el archivo aircrack-ng-1.2-beta1.tar.gz se utiliza el siguiente comando:

```
$ tar -zvxf aircrack-ng-1.2-beta3.tar.gz
```

Ubicarse en el directorio generado después del paso anterior, para compilar e instalar aircrack-ng se debe ejecutar lo siguiente:

```
$ make && sudo make install
```

Para configurar la tarjeta de red en modo monitor se necesita instalar *iw* (es una utilidad de configuración para dispositivos inalámbricos sucesora de *iwconfig*):

```
$ sudo apt-get install iw
```

Para comprobar que todo funcione correctamente se debe conectar una tarjeta de red inalámbrica que pueda configurarse en modo monitor. Se puede identificar la tarjeta de red inalámbrica así:

```
$ iwconfig
```

Para conocer el *chipset* de la tarjeta de red inalámbrica se utiliza el comando airmon-ng:

```
$ sudo airmon-ng
```

Para poner la tarjeta en modo monitor se debe ejecutar lo siguiente:

```
$ sudo airmon-ng start wlan0
```

También podemos poner la tarjeta en modo promiscuo con la siguiente orden:

```
$ ifconfig wlan0 promisc
```

3.- COMANDOS Linux

- Ejecutar sesión con privilegios root: -> **\$ sudo bash ó \$ sudo su**
- Para cambiar de usuario:-> **su USUARIO**
- Identificar procesos en ejecución: -> **\$ ps aux**
- Conocer el espacio disponible en la SD: -> **\$ df -h**
- Identificar los dispositivos montados: -> **\$ sudo parted** luego teclear **\$ print all**
- Desinstalar paquetes: -> **\$ sudo apt-get --purge remove <package>**
- Muestra los dispositivos USB instalados: -> **\$ lsusb**
- Imprimir en el terminal el directorio en el que te encuentras: -> **\$ pwd**
- Ver el contenido del directorio: -> **\$ ls** - Ver el contenido del directorio con atributos: -> **\$ ls -l**
- Crear un archivo: -> **\$ sudo touch archivo**
- Renombrar un archivo: -> **\$ sudo mv archivo nuevoarchivo**
- Mover fichero a carpeta:-> **sudo mv archivo /carpeta**
- Copiar archivo: -> **\$ cp archivo1 archivo2**
- Borrar archivo: -> **\$ sudo rm archivo1**
- Borrar si la carpeta no está vacía: -> **\$ sudo rm -rf carpeta**
- Cambiar a otro directorio: -> **\$ cd nombredeldirectorio**
- Para crear un solo directorio: -> **\$ mkdir nombredirectorio**
- Para crear un árbol de directorios:-> **\$ mkdir -p /home/usuario/directorio1/directorio2**
- Para conocer la información hardware general: -> **\$ cat /proc/cpuinfo**
- Para saber el estado de la memoria: -> **\$ cat /proc/meminfo**
- Ver las particiones de la tarjeta de memoria o el disco duro: -> **\$ cat /proc/partitions**
- Versión de Raspbian: -> **\$ cat /etc/*-release** ---- -> **\$ lsb_release -a**
- Temperatura de la CPU: -> **\$ vcgencmd measure_temp**
- Para entrar en la ventana de configuración de Raspbian: -> **\$ sudo raspi-config**
- Si estamos en modo línea de comandos y queremos volver al modo gráfico: -> **\$ startx**
- Apagar el dispositivo: -> **\$ sudo shutdown -h ó sudo halt**
- Reiniciar la Raspberry: -> **\$ sudo shutdown -r now ó → \$ sudo reboot**
- **\$ who**

Who es un comando informativo que muestra los datos de los usuarios que han iniciado sesión en el equipo. A diferencia de whoami, who muestra información de todos los usuarios en sesión, incluyendo aquellos conectados vía ssh.

- Para obtener información del comando:-> **\$ man <nombre del comando>**
- El comando **which** nos sirve para averiguar donde se encuentra instalado un determinado programa. Por ejemplo, si ejecutamos: **\$ which wine** obtenemos **/usr/local/bin/wine**
- **\$ uname** → muestra información del sistema. Tiene parámetros: -s -n -a -m -r -v -p
- Para cambiar la clave del root:
 - Hacemos control remoto y ponemos **\$ sudo su**
 - Luego **\$ passwd**
- Cambiar el dueño de un fichero o carpeta:-> **\$ sudo chown pi fichero | carpeta**
- Cambiar el grupo de un fichero o carpeta:-> **\$ sudo chgrp pi fichero | carpeta**
- Comprobamos los distintos puertos de los que dispone la RASPI:-> **\$ ls /dev/tty***
- Cambia el intercambio swap: -> **\$ sudo swappiness=(entre 0 y 100)**
- Cerrar puertos: → **\$ fuser -k 80/tcp**
- Escribir la virgulilla (alt-gr+4) o tecleamos (F5): → ~

- Permisos de ficheros

r: permiso para leer / **w**: permiso para escribir / **x**: permiso para ejecutar

\$ chmod ugo+rwx test (da permisos rwx a todos, user,group,others)

\$ chmod o-rwx test (quita permisos rwx a others)

\$ chmod u=rwx,g=rx test (da permisos rwx a user, rx a group y ninguno a others)

Otra manera de dar permisos es cambiar los permisos de la carpeta en la que lo ejecutemos

\$ sudo chmod 777 * (rwxrwxrwx)

\$ sudo chmod 774 * (rwxrwxr..)

7 = 111 = [rwx] / 6 = 110 = [rw.] / 5 = 101 = [r.x] / / 0 = 000 = [...]

- Para saber todos los procesos que se están ejecutando: → **\$ ps -a / \$ ps -u / \$ ps -x**
- Para matar un proceso determinado: → **\$ Kill -9 ID_proceso**
- Espacio libre en GB o MB de nuestra tarjeta SD → **\$ df -h**
- Espacio ocupado por los ficheros de un directorio:-> **du -h /home/pi**
- Conocer la cantidad de memoria RAM y swap utilizadas:-> **free**
- Volcado a fichero de los datos de pantalla (>): → **Ejemplo: \$ ls > lista.txt**
- **Calendario → \$ cal / \$ ncal 1960 -m 7 / \$ ncal 2016 -C**
- Algunos comandos del editor de texto **nano**:
 - ALT +A --- Selecciona / ALT +6 --- Copia / ALT +u --- Pega
- Si quieres saber la tº de la RASPI: -> **\$ /opt/vc/bin/vcgencmd measure_temp**
- Prueba de velocidad:

\$ dd if=/dev/zero of=/tmp/out bs=1M count=300 30 MB/s es un buen resultado (300x=14,4MB/s
- 633x=27,8MB/s – U1casa=23,5MB/s -sandisk=21,8MB/s – Lexar=22,1MB/s)

- Seguramente, muchas veces hemos usado:

\$ apt-get update y \$ apt-get upgrade

Pero, ¿qué hace cada orden o comando? <https://debian-handbook.info/browse/es-ES/stable/sect.apt-get.html>

Cuando nosotros utilizamos **\$ apt-get update**, lo que en realidad estamos haciendo es actualizar los repositorios --ver si hay algo nuevo--, es decir actualizar la lista de todos los paquetes, con la dirección de dónde obtenerlos para que a la hora de hacer la búsqueda y su posterior descarga, sea más rápida.

En cambio, cuando utilizamos **\$ apt-get upgrade**, lo que hacemos es una actualización de nuestro sistema con todas las posibles actualizaciones que pudiera haber, es decir, no sólo actualiza nuestro sistema operativo sino que también las aplicaciones que están contenidas en los repositorios.

En resumen: el update lista los paquetes de los repositorios y el upgrade instala las actualizaciones.

- El comando

\$ sudo apt-get dist-upgrade

además de realizar las acciones de upgrade, trata inteligentemente los cambios de dependencias debidos a las nuevas versiones de paquetes, resolviendo conflictos, y si es necesario actualizando los paquetes más importantes a costa de los menos importantes. Es decir, es más completo.

- Si al ejecutar alguna orden te responde:

```
sudo: /etc/sudoers is world writable
sudo: no valid sudoers sources found, quitting
sudo: no se puede inicializar la política de plugin
```

Ejecuta lo siguiente:

\$pkexec chmod 555 /etc/sudoers

\$pkexec chmod 555 /etc/sudoers.d/README

- **Actualizar el firmware y kernel de la Raspi**

Para ello utilizamos → **\$ sudo rpi-update**

Con estos comandos podemos ver las informaciones de lo realizado:

```
$ sudo aptitude show raspberrypi-bootloader
$ /opt/vc/bin/vcgencmd version
$ uname -a
```

- Saber si un paquete está instalado:-> **\$ aptitude show paquete**

- Como **instalar un programa** desde los ficheros fuente:

```
$ ./configure
$ make
$ sudo make install
```

- Como **desinstalar** un paquete:

```
$ sudo apt-get remove paquete
$ sudo apt-get purge paquete
$ sudo apt-get clean paquete
```

- **Limpiar** instalaciones fallidas:

```
$ sudo apt-get clean
```

- **Solucionar errores** de instalaciones fallidas:

```
$ sudo dpkg --configure -a
```

- **Eliminar** paquetes inservibles:

```
$ sudo apt-get autoremove
```

- Como **instalar** ficheros .bin y .sh

.- Primero se le da permiso de ejecutar al archivo y luego se ejecuta la siguiente orden.

```
$ sudo sh fichero.sh
```

```
$ sudo sh fichero.bin
```

- Como **Ejecutar** archivos en Linux desde la **consola**

Mucha gente nueva en Linux suele tener dudas sobre cómo ejecutar o instalar algunos programas que vienen en archivos ejecutables como son por ejemplo los .bin, .run, .py o .sh

Los archivos .bin y los .run suelen ser instaladores de programas, mientras que los .sh son scripts que ejecutas directamente en la consola. La ventaja de instalar un programa con estos formatos es que por lo general van a funcionar bien en todas las distribuciones, mientras que otros formatos precompilados, para instalar programas, como los .deb o .rpm están más limitados.

1º Dar al archivo **permisos** de ejecución

2º Escribimos ./ delante del archivo → **\$ sudo ./archivo.py**

```
$ sudo ./archivo.bin
```

3º Instalar un .deb → **\$sudo dpkg -i *.deb**

- **Comando “pip”**

Para desarrollar software con rapidez y calidad, es imprescindible utilizar paquetes externos que ayuden con parte de la funcionalidad que se desea implementar. En el ambiente Python esto no es la excepción.

Para solventar ésta necesidad, la comunidad Python ha puesto convenientemente a disposición de los desarrolladores, un repositorio de paquetes de fácil acceso llamado **PyPi**. Solo es necesario ejecutar un comando en la terminal para poder instalar el paquete Python que necesitemos. Incluso es posible instalar paquetes que no se encuentren en el mencionado repositorio.

Para descargar paquetes del repositorio PyPi se pueden utilizar varias herramientas, pero en este caso se va a usar **pip**. Es necesario instalar esta herramienta en el sistema en caso de no estar disponible, antes de poder instalar un paquete Python.

Es posible instalar pip usando el gestor de paquetes de una distribución Linux:

```
$ sudo apt-get install python-pip
```

Si queremos actualizar a la última versión de pip:

```
$ sudo pip install --upgrade pip
```

Instalar paquetes Python con **pip** es bastante sencillo, por ejemplo:

```
$ sudo pip install django
```

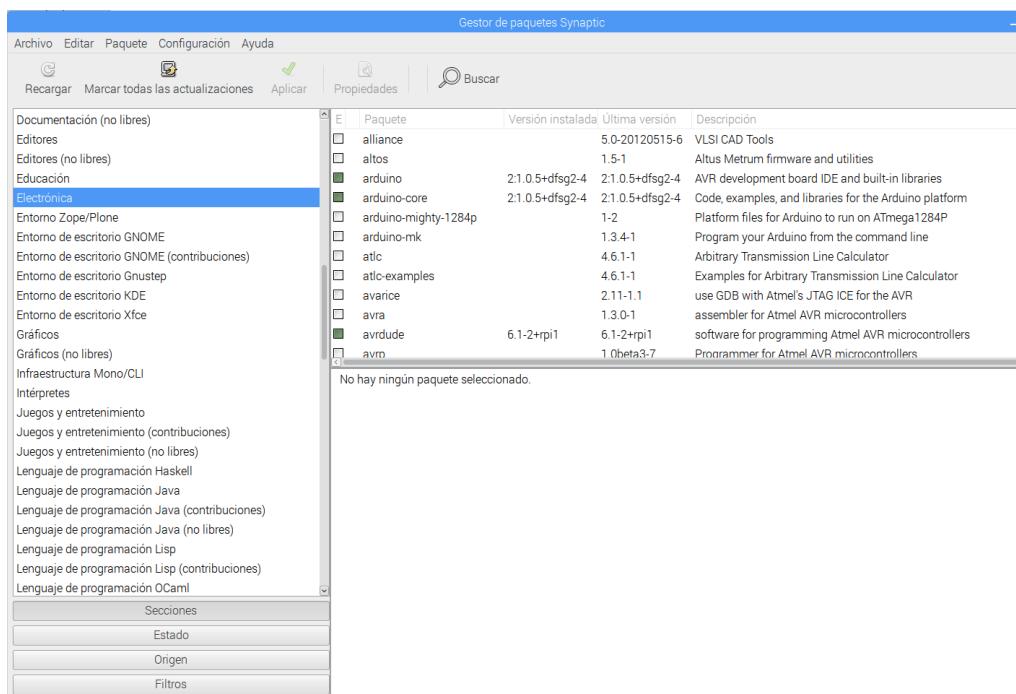
Otras operaciones que se pueden hacer con pip:

- Para buscar un paquete en el repositorio PyPi:
`$ pip search palabra_relacionada_con_el_paquete`
- Para consultar los paquetes instalados globalmente en el sistema:
`$ sudo pip freeze`
- Para desinstalar un paquete instalado globalmente en el sistema:
`$ sudo pip uninstall nombre_del_paquete`

- Gestor de Paquetes SYNAPTIC

Otra de las aplicaciones imprescindibles es **Synaptic**, este maravilloso gestor de paquetes para sistemas Linux. Desde el vamos a poder gestionar fácilmente todos los paquetes, solucionar posibles problemas de dependencias e instalar todo lo que necesitemos. Para ello, simplemente ejecutaremos:

```
$ sudo apt-get install synaptic
```



MEMORIA SWAP

Por ello, el uso de la partición swap debe limitarse a situaciones en las cuales no queda otra alternativa más que utilizarla, momentos en los cuales servirá de apoyo a la memoria principal (que es la RAM). Si en lugar de ello la utilizamos en todo momento, incluso a veces antes que la memoria RAM, nuestro rendimiento se verá penalizado. Veamos entonces cómo ajustar el uso de la memoria virtual en Linux mediante el comando **Swappiness**.

Swappiness es una propiedad del Núcleo Linux que permite ajustar el equilibrio entre el uso del Espacio de intercambio (swap en inglés, por eso el nombre de la propiedad) y la Memoria de acceso aleatorio (RAM). El swappiness puede tomar valores desde el 0 hasta el 100.

El valor por defecto en una instalación de Linux es de 60, pero como es fácil suponer no todas las configuraciones de hardware son iguales y por ello tampoco tiene sentido mantener ese nivel sin importar cuál es la nuestra. Dicho valor es almacenado en el archivo **/proc/sys/vm/swappiness**, y podemos comprobarlo mediante:

```
$ cat /proc/sys/vm/swappiness
```

Casi seguro estará en 60, y si ese es el caso puede que debamos modificarlo, sobre todo **si tenemos más de 4 GB de memoria RAM ya que en ese caso lo habitual será que necesitemos poco y nada de la memoria virtual**. Pero antes de explicar cómo modificar eso, veamos un poco cual es la lógica detrás de todo este asunto de la memoria virtual y la swappiness; y es que cuando se lo deja por defecto en 60 lo que se le dice al kernel es que vaya y utilice la memoria virtual cuando nuestra memoria RAM tenga el 40 por ciento o menos de su capacidad libre. Por ende, **si establecemos swappiness igual a 100 la memoria virtual se utilizará todo el tiempo**, y si la dejamos en un valor muy bajo se utilizará únicamente cuando nuestra memoria RAM esté a punto de agotarse. **El mínimo posible es de 1, ya que dejando el valor igual a 0 desactivamos la memoria virtual por completo**.

```
$ sudo sysctl vm.swappiness=10
```

Ahora el valor de **swappiness quedará en 10, y entonces la memoria virtual apenas se utilizará**. Una vez cambiado este valor **no es necesario reiniciar el equipo, sino que surte efecto de inmediato**, y de hecho si reiniciamos el valor volverá a estar ubicado en 60 como antes, porque lo que vamos a necesitar es dejar este cambio establecido en forma permanente. Para ello, una vez que hemos utilizado nuestro ordenador y comprobado que todo marcha bien con el nuevo valor de swappiness, ejecutamos:

```
$ sudo nano /etc/sysctl.conf
```

tras lo cual buscamos el texto **vm.swappiness=** y agregamos el valor deseado luego del símbolo “=”. Guardamos el archivo y ahora sí, el cambio será permanente.

sudo - su - /etc/sudoers - visudo

Su

El programa **su** permite usar el intérprete de comandos de otro usuario sin necesidad de cerrar la sesión actual. Comúnmente se usa para obtener permisos de root para operaciones administrativas sin tener que salir y reentrar al sistema. Algunos entornos de escritorio, entre ellos GNOME y KDE, tienen programas que piden gráficamente una contraseña antes de permitir al usuario ejecutar un comando que usualmente requeriría tal acceso.

El nombre **su** proviene del inglés substitute user (usuario substituto). También hay quien lo hace derivar de superuser (super-usuario, es decir, el usuario root o administrador) ya que habitualmente se utiliza para adoptar el rol de administrador del sistema.

Cuando se ejecuta, **su** pide la contraseña de la cuenta a la se quiere acceder, y si es aceptada, da acceso a dicha cuenta.

```
[fulano@localhost]$ su  
Contraseña:  
[root@localhost]# exit  
logout  
[fulano@localhost]$
```

Al no poner un usuario, se accede como administrador. Sin embargo, también es posible pasar como parámetro otro nombre de usuario.

```
[fulano@localhost]$ su mongo  
Contraseña:  
[mongo@localhost]# exit  
logout  
[fulano@localhost]$
```

Una vez introducida la contraseña, podemos ejecutar los comandos como si fuésemos el otro usuario. Al escribir **exit**, volvemos a nuestro usuario.

Una variante muy utilizada es usar **su** seguido de un guión. Así, para loguearte como root, tienes que ingresar **su** – y para loguearte como otro usuario **su** – otrosusuario. ¿La diferencia entre usar o no el guión? Se recomienda usar el guión porque simula que te logueás con ese usuario; por consiguiente, ejecuta todos los archivos de inicio de ese usuario, cambia el directorio actual al HOME de ese usuario, cambia el valor de algunas variables del sistema adaptándolas al nuevo usuario (HOME, SHELL, TERM, USER, LOGNAME, entre otros), y otras cositas más.

Un administrador de sistemas debe tener mucho cuidado al elegir una contraseña para la cuenta de **root/administrador**, para evitar un ataque por parte de un usuario no privilegiado que ejecute **su**. Algunos sistemas de tipo Unix tienen un grupo de usuarios llamado **wheel**, que comprende a los únicos que pueden ejecutar **su**. Esto podría o no reducir los problemas de seguridad, ya que un intruso podría simplemente apoderarse de una de esas cuentas. El **su** de GNU, sin embargo, no admite el uso de ese grupo; esto se hizo por razones filosóficas.

Sudo

Un comando relacionado, llamado **sudo**, ejecuta un comando como otro usuario, pero respetando una serie de restricciones sobre qué usuarios pueden ejecutar qué comandos en nombre de qué otros usuarios (usualmente especificadas en el archivo **/etc/sudoers**).

Por otro lado, a diferencia de **su**, **sudo** pide a los usuarios su propia contraseña en lugar de la del usuario requerido; esto permite la delegación de comandos a usuarios en otras máquinas sin tener que compartir contraseñas, reduciendo el riesgo de dejar terminales desatendidas.

Problemas con sudo: el período de gracia

La ventaja de **sudo** respecto de **su** es que sólo ejecuta el comando solicitado simulando ser el otro usuario, sin cambiar verdaderamente el usuario actual. Esto implica que uno puede ejecutar un comando como administrador y, al segundo siguiente, volverá a tener los privilegios del usuario que estaba usando antes... o casi.

Algunos ven como una brecha de seguridad el hecho de que **sudo** otorgue un “período de gracia” que permita al usuario ejecutar comandos como otro usuario sin la necesidad de tener que ingresar una y otra vez **sudo** delante del comando y la contraseña luego de ejecutarlo. Pasado ese “**período de gracia**”, **sudo** volverá a preguntarnos la clave.

Esto es “**malo**”, esencialmente porque alguien podría apoderarse de nuestra compu luego de haber ingresado la contraseña de **sudo** y mientras el “**período de gracia**” está activo hacer un DESASTRE.

Afortunadamente, es posible deshabilitar el “**período de gracia**”, lo que va a mejorar la seguridad de tu sistema. Sólo hay que agregar una línea en el archivo **/etc/sudoers**:

```
$ sudo nano /etc/sudoers
```

Y agregá la siguiente línea al final del archivo:

```
Defaults:ALL timestamp_timeout=0
```

El cambio tiene efecto en forma inmediata, sin necesidad de reiniciar el sistema.

Sudoers

Archivo de configuración de **sudo**, generalmente ubicado bajo **/etc** y se modifica a través del uso de visudo. En este archivo se establece quien (usuarios) puede ejecutar qué (comandos) y de qué modo (opciones), generando efectivamente una lista de control de acceso que puede ser tan detallada como se deseé.

Es más fácil entender **sudoers** si dividimos en tres partes su posible configuración, estás son:

- Alias
- Opciones (Defaults)
- Reglas de acceso

Por extraño que parezca ninguna de las secciones es obligatoria, o tienen que estar en algún orden específico, pero la que al menos debe de existir es la tercera, que es la definición de los controles o reglas de acceso.

visudo

Permite la edición del archivo de configuración de **sudo sudoers**. Invoca al editor que se tenga por defecto que generalmente es **vi**.

visudo cuando es usado, bloquea el archivo **/etc/sudoers** de tal manera que nadie más lo puede utilizar, esto por razones obvias de seguridad que evitarán que dos o más usuarios administradores modifiquen accidentalmente los cambios que el otro realizó.

Otra característica importante de **visudo** es que al cerrar el archivo, verifica que el archivo esté bien configurado, es decir, detectará si hay errores de sintaxis principalmente en sus múltiples opciones o reglas de acceso que se tengan. Por esta razón no debe editarse **/etc/sudoers** directamente (perfectamente posible ya que es un archivo de texto como cualquier otro) sino siempre usar **visudo**.

Si al cerrar **visudo** detecta un error nos mostrará la línea donde se encuentra, y la pregunta "**What now?**:

```
>>> sudoers file: syntax error, line 15 <<<  
What now?
```

Se tienen tres opciones para esta pregunta:

- **e** - edita de nuevo el archivo, colocando el cursor en la línea del error (si el editor soporta esta función.)
- **x** - salir sin guardar los cambios.
- **Q** - salir y guarda los cambios.

Por defecto el archivo de configuración es **/etc/sudoers** pero se pueden editar otros archivos que no sean ese y que se aplique la sintaxis de **sudo**, y esto se logra con la opción **-f**

```
$ visudo -f /otro/archivo
```

Si tan solo se desea comprobar que **/etc/sudoers** está bien configurado se usa la opción **-c**, toma por el archivo de configuración por defecto o si no se indica algún otro.

```
#> visudo -c  
/etc/sudoers file parsed OK
```

La opción **-s** activa el modo '**estricto**' del uso de **visudo**, es decir no solo se comprobará lo sintáctico sino también el orden correcto de las reglas, por ejemplo si se define el alias para un grupo de comandos y este se usa antes de su definición, con esta opción se detectará este tipo de errores.

ENTORNO VIRTUAL (virtualenv / virtualenvwrapper)

Un entorno virtual de Python **es un ambiente creado con el objetivo de aislar recursos**, como librerías y entorno de ejecución, del sistema principal o de otros entornos virtuales. Lo anterior, significa que en el mismo sistema, maquina o computadora, **es posible tener instaladas multiples versiones de una misma librería** sin crear ningún tipo de conflicto.

Cuando se está desarrollando software con Python, **es común utilizar diferentes versiones de un mismo paquete**. Por ejemplo, imaginemos que se está desarrollando un videojuego con la versión 1.2 de Pygame y mientras eso pasa, se comienza el desarrollo de otro videojuego que necesita las nuevas características presentes en la versión 1.3. En este escenario, no es posible para los desarrolladores eliminar la versión 1.2 e instalar la 1.3 en sus computadoras. Así que el problema a solucionar radica en **cómo instalar las dos versiones de la misma librería** con el fin de poder desarrollar ambos proyectos de forma simultánea.

La solución consiste en crear entornos virtuales. De esta manera, es posible instalar la versión 1.2 de Pygame en un entorno virtual y la versión 1.3 en otro diferente o en el sistema principal sin problema alguno.

Para poder utilizar este simple pero poderoso concepto **es necesario instalar una utilidad que permita gestionar la creación y utilización de dichos entornos virtuales** llamada **virtualenv**.

Es posible potenciar las capacidades de la utilidad **virtualenv** con las extensiones creadas por Doug Hellmann compiladas en un paquete Python llamado **virtualenvwrapper** que facilitarán notoriamente la gestión de un gran número de entornos virtuales.

1º paso

```
$ sudo pip3 install virtualenv virtualenvwrapper
```

2º paso

Añadir órdenes al final del fichero de configuración del “bash”

```
$ sudo nano ~/.bashrc
```

```
# virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
export VIRTUALENVWRAPPER_PYTHON=/usr/bin/python3
export VIRTUALENVWRAPPER_VIRTUALENV=/usr/local/bin/virtualenv
source /usr/local/bin/virtualenvwrapper.sh
export VIRTUALENVWRAPPER_ENV_BIN_DIR=bin
export VIRTUALENVWRAPPER_VIRTUALENV=/usr/local/bin/virtualenv
```

```
$ source ~/.bashrc
```

3º paso

Crear los entornos virtuales, por ejemplo vamos a crear dos:

```
$ mkvirtualenv luis -p python3
```

```
$ mkvirtualenv pepe -p python3
```

Ahora ya podemos utilizarlos... https://virtualenvwrapper-docs-es.readthedocs.io/es/latest/command_ref.html

deactivate → te saca del entorno

workon → te lista los entornos

workon nombredelentorno → te mete en el entorno

CONTROL DE PROCESOS Y TRABAJOS

<https://www.luisllamas.es/listado-comandos-linux-utiles/>

EJECUTAR COMANDOS Y APLICACIONES

```
1 #ejecutar aplicacion en carpeta actual  
2 ./aplicacion  
3  
4 #ejecutar comando  
5 comando  
6  
7 #iniciar proceso en background  
8 comando &  
9  
10 #cancelar comando  
11 control + c  
12  
13 #poner comando en background  
14 control + z  
15  
16 #recuperar proceso puesto en background  
17 bg  
18  
19 #poner trabajo en primer plano  
20 fg jobID  
21  
22 #iniciar comando como root  
23 sudo comando  
24  
25 #iniciar ejecutable con interface grafica como root  
26 sudo ejecutable  
27  
28 #gestor de sesiones multiples en terminal  
29 screen -S nombre_sesion
```

LISTAR PROCESOS Y TAREAS

```
1 #mostrar procesos en background con jobID y PID  
2 jobs -l  
3  
4 #mostrar procesos  
5 ps  
6  
7 #mostrar procesos activos  
8 ps -eafw  
9  
10 #mostrar árbol de sistema de procesos.  
11 pstree  
12  
13 #mostrar procesos ordenados por consumo de memoria  
14 ps aux | sort -k 5
```

FINALIZAR PROCESOS

```
1 #finalizar proceso para recargar configuracion  
2 kill -1 ID_Processo  
3  
4 #forzar cierte de proceso por PID  
5 kill -9 PID  
6  
7 #finalizar proceso por nombre  
8 killall -9 nombre
```

ALIAS DE COMANDOS

```
1 #configurar un alias temporal para comando
2 alias cmd='comando'
3
4 #eliminar alias
5 unalias cmd
```

INFORMACION Y SUPERVISIÓN DEL SISTEMA

```
1 #limpiar pantalla de terminal (mismo efecto que control+l)
2 clear
3
4 #reiniciar sesion de terminal
5 reset
6
7 #terminar sesion de terminal local o remoto (ssh) y finalizar procesos iniciados
8 exit
```

GESTIÓN DE RECURSOS

```
1 #mostrar tareas ejecutandose y su uso de recursos
2 top
3
4 #mostrar tareas ejecutables y recursos mejorado
5 htop
6
7 #muestra el estado de la RAM en megabytes
8 free -h
```

ESPAZIO DE DISCO

```
1 #mostrar una lista de las particiones montadas
2 df -h
3
4 #mostrar el tamaño de los archivos y directorios ordenados por tamaño
5 ls -lSr |more
6
7 #Estimar el espacio usado por el directorio 'dir1'
8 du -sh directorio
9
10 #mostrar el tamaño de los archivos y directorios ordenados por tamaño
11 du -sk * | sort -rn
```

INFORMACIÓN DE SISTEMA

```
1 #mostrar datos de usuarios conectados
2 who -a
3
4 #mostrar historial de reinicio
5 last reboot
6
7 #Mostrar arquitectura y versión de Linux y Kernel
8 uname -a
9
10 #mostrar el kernel cargado.
11 lsmod
12
13 #mostrar componentes de hardware del sistema.
14 dmidecode -q
15
16 #Listar particiones de disco duro
17 *cat /etc/fstab
18
19 #mostrar características de disco duro
20 hdparm -i /dev/hda
21
22 #mostrar dispositivos PCI
23 lspci
24
25 #mostrar dispositivos USB
26 lsusb
27
28 #mostrar eventos de proceso de carga de kernel
29 tail /var/log/dmesg
30
31 #mostrar los eventos del sistema
32 tail /var/log/messages
33
34 #mostrar lista de archivos abiertos por procesos
35 lsof -p $$
36
37 #mostrar lista de archivos abiertos en un camino dado del sistema
38 lsof /directorio
39
40 #mostrar llamadas del sistema hechas y recibidas por un proceso
41 strace -c ls >/dev/null
42
43 #mostrar las llamadas a la biblioteca
44 strace -f -e open ls >/dev/null
45
46 #mostrar interrupciones en tiempo real
47 watch -n1 'cat /proc/interrupts'
```

APAGADO Y REINICIO DE SISTEMA

```
1 #cerrar sesión usuario
2 logout
3
4 #apagar el sistema ahora
5 shutdown now
6
7 #reiniciar sistema ahora
8 shutdown -r now
9
10 #apagado programado
11 shutdown horas:minutos &
12
13 #cancelar apagado programado
14 shutdown -c
```

FECHAS

```
1 #mostrar la fecha del sistema  
2 date  
3  
4 #mostrar calendario de un año  
5 cal año  
6  
7 #mostrar calendario de mes y año  
8 cal mes año 2011
```

AYUDAS

```
1 #manual online de comando  
2 man comando  
3  
4 #muestra un resumen descriptivo de la función de comando  
5 whatis comando  
6  
7 #buscar comandos por la tarea realizada (inverso del anterior)  
8 apropos texto
```

DESPLAZARSE ENTRE DIRECTORIOS

```
1 #ir a directorio de raíz  
2 cd  
3  
4 #/ir a directorio anterior  
5 cd ..  
6  
7 #entrar en directorio (ruta absoluta)  
8 cd /directorio1/directorio  
9  
10 #entrar en directorio (ruta relativa)  
11 cd directorio1/directorio2  
12  
13 #ir a directorio de usuario  
14 cd ~  
15  
16 #ir a último directorio visitado  
17 cd -  
18  
19 #mostrar ruta actual  
20 pwd
```

LISTAR ARCHIVOS Y DIRECTORIOS

```
1 #mostrar archivos y directorios  
2 ls  
3  
4 #mostrar archivos y directorios con detalles  
5 ls -l  
6  
7 #mostrar archivos y directorios incluidos los ocultos  
8 ls -a
```

MANIPULACIÓN DE ARCHIVOS Y DIRECTORIOS

```
1 #renombrar o mover un archivo o directorio
2 mv origen destino
3
4 #copiar un archivo
5 cp archivo direccion
6
7 #copiar un directorio
8 cp -r origen destino
9
10 #borrar el archivo llamado archivo
11 rm archivo
12
13 #borrar directorio si está vacío
14 rm -d directorio
15
16 #borrar directorio y su contenido
17 rm -r directorio
18
19 #crear nuevo directorio
20 mkdir directorio
21
22 #crear varios directorios simultáneamente
23 mkdir directorio1 directorio2
24
25 #crear ruta de directorios
26 mkdir -p /directorío1/directorío2
27
28 #crear archivo vacío
29 touch archivo
30
31 #cambiar fecha de archivo (formato año, mes, dia, y hora)
32 touch -t 199012300000 archivo
```

ENLACES SIMBÓLICOS

```
1 #crear un enlace simbólico al archivo o directorio
2 ln -s archivo lnk1
3
4 #crear enlace físico al archivo o directorio
5 ln archivo lnk1
```

CODIFICACIÓN

```
1 #calcular md5 de un archivo
2 md5sum archivo
3
4 #codificar archivo GNU Privacy Guard.
5 gpg -c
6
7 #decodificar archivo GNU Privacy Guard.
8 gpg archivo.gpg
```

OPERACIONES DE CONTENIDO DE ARCHIVO

TUBERIAS Y REDIRECCIONES I/O

```
1 #dirigir salida de comando a nuevo archivo
2 comando > archivo_out.txt
3
4 #dirigir salida de comando para añadir a archivo archivo
5 comando >> archivo_out.txt
6
7 #dirigir entrada de comandos
8 comando < archivo_in.txt
9
10 #dirigir salida estandar y salida de error a un fichero:
11 comando &> archivo_out.txt
12
13 #tuberia, dirigir salida de comando1 como entrada de comando2
14 comando1 | comando2
```

MOSTRAR CONTENIDO DE ARCHIVO

```
1 #muestra contenido de archivo
2 echo archivo
3
4 #mostrar contenido de archivo
5 cat archivo
6
7 #mostrar contenido de archivo empezando por el final
8 tac archivo
9
10 #mostrar contenido de archivo desplazandose linea a linea
11 more archivo
12
13 #mostrar contenido de archivo desplazandose adelante o atrás
14 less archivo
15
16 #mostrar dos primeras lineas de archivo
17 head -2 archivo
18
19 #mostrar dos ultimas lineas de archivo
20 tail -2 archivo
21
```

MANIPULACIÓN DE TEXTO

```
1 #convertir minúsculas en mayúsculas
2 echo 'archivo' | tr '[:lower:]' '[:upper:]'
3
4 #eliminar lineas 1 a 5 de archivo
5 sed '3,5d' archivo
6
7 #eliminar lineas 5 a fin de archivo
8 sed '5,$d' archivo
9
10 #eliminar lineas en blanco
11 sed '/^$/d' archivo
12
13 #eliminar lineas en blanco y comentarios
14 sed '/ *#/d; /^$/d' archivo
15
16 #sustituir una cadena por otra
17 sed 's/cadena1/cadena2/g' archivo
18
19 #visualizar únicamente las líneas que contienen cadena
20 sed -n '/cadena/p'
```

BUSQUEDA DE ARCHIVOS Y EN SU CONTENIDO

BUSCAR DE ARCHIVOS

```
1 #buscar archivo y directorio por su nombre en todo el sistema  
2 find * -name nombre  
3  
4 #buscar archivo y directorio por su nombre, dentro de directorio  
5 find directorio -name nombre  
6  
7 #buscar archivos y directorios pertenecientes a usuario, dentro de directorio  
8 find directorio -user usuario  
9  
10 #buscar archivos y directorios por su tipo, dentro de directorio  
11 (d directory, f regular file, l symbolic link)  
12 find directorio -type f tipo  
13  
14 #buscar archivos y ejecutar comando  
15 find directorio -name nombre -exec comando {} \;  
16  
17 #buscar archivos con extensión .ps  
18 locate \*.ps  
19  
20 #mostrar la ruta completa de un ejecutable  
21 which ejecutable  
22  
23 #mostrar la ubicación de un archivo binario, de ayuda o fuente  
24 whereis ejecutable
```

BUSCAR EN CONTENIDO DE ARCHIVO

```
1 #buscar cadena en el/los archivos  
2 grep cadena archivo  
3  
4 #buscar cadena en el/los archivos sin coincidencia de mayusculas  
5 grep -i cadena archivo  
6  
7 #buscar palabras que comienzan con cadena en el/los archivos  
8 grep ^cadena archivo  
9  
10 #buscar cadena como palabra completa en el/los archivos  
11 grep -w cadena archivo  
12  
13 #seleccionar las lineas de archivo que contienen números  
14 grep [0-9] archivo  
15  
16 #busca recursiva de cadena en el directorio  
17 grep cadena -R direccion
```

GESTIÓN DE USUARIOS Y GRUPOS

USUARIOS

```
1 #crear un nuevo usuario  
2 useradd usuario  
3  
4 #crear usuario, version completa  
5 useradd -c "Nombre Usuario" -g grupo -d /home/usuario -s /bin/bash usuario  
6  
7 #borrar usuario  
8 userdel usuario  
9  
10 #borrar usuario y eliminar su directorio home  
11 userdel -r usuario  
12  
13 #cambiar los atributos del usuario.  
14 usermod -c "Nombre usuario" -g grupo -d /home/usuario -s /bin/bash usuario  
15  
16 #cambiar contraseña del propio usuario.  
17 passwd  
18 #cambiar la contraseña de usuario (solamente por root).  
19 passwd usuario  
20  
21 #colocar una fecha de expiración para contraseña de usuario  
22 chage -E 2014-12-31 usuario
```

GRUPOS DE USUARIOS

```
1 #crear grupo usuarios  
2 groupadd nombre_grupo  
3  
4 #borrar grupo de usuarios  
5 groupdel nombre_grupo  
6  
7 #renombrar grupo de usuarios  
8 groupmod -n nombre_nuevo nombre_anterior  
9  
10 #modificar el grupo actual de un usuario que pertenece a varios grupos  
11 newgrp grupo  
12  
13 #listar grupos del usuario actual  
14 groups  
15  
16 #listar todos los grupos  
17 cut -d: -f1 /etc/group
```

COMPROBACIONES

```
1 #comprueba que el archivo de contraseñas /etc/passwd es correcto  
2 pwck  
3  
4 #comprueba que el archivo de grupos /etc/group #es correcto  
5 grpck
```

PERMISOS Y ATRIBUTOS ESPECIALES

PERMISOS DE ARCHIVOS Y CARPETAS

```
1 #usar + para colocar permisos y - para eliminar
2 Mostrar permisos.
3 ls -lh
4
5 #asignar permisos 0777 a fichero
6 #modificar 0777 segun codificacion octal de permisos
7 chmod 0777 fichero
8
9 #asignar permisos a todos los archivos de un directorio
10 chmod -R 0644 directorio
11
12 #colocar a directorio permisos de lectura (r), escritura (w) y ejecución (x) al
13 #emplear las opciones necesarias para añadir o quitar los permisos deseados
14 chmod ugo+rwx directorio
15
16 #cambiar usuario de archivo
17 chown usuario archivo
18
19 #cambiar usuario a todos los archivos de un directorio
20 chown -R usuario directorio
21
22 #cambiar grupo de archivo
23 chgrp grupo archivo
24
25 #cambiar usuario y grupo de archivo.
26 chown usuario:grupo archivo
```

PERMISOS SUID

```
1 #visualizar todos los archivos del sistema con SUID configurado
2 find / -perm -u+s
3
4 #colocar bit SUID en archivo binario. El usuario que ejecute este archivo adqui
5 chmod u+s /bin/archivo
6
7 #eliminar bit SUID en archivo binario
8 chmod u-s /bin/archivo
9
10 #colocar bit SGID en directorio. Similar a SUID pero para directorios
11 chmod g+s /home/directorio
12 #eliminar bit SUID en archivo binario.
13 chmod g-s /home/directorio
14
15 #colocar un bit STIKY en un directorio. Permite el borrado de archivos solamente
16 chmod o+t /home/directorio
17 #eliminar bit STIKY en un directorio.
18 chmod o-t /home/directorio
```

ATRIBUTOS ESPECIALES DE ARCHIVO

```
1 #usar + para colocar permisos y - para eliminar
2 #mostrar atributos especiales.
3 lsattr
4
5 #permite escribir abriendo un archivo solamente modo append.
6 chattr +a archivo
7
8 #permite que un archivo sea comprimido / descomprimido automaticamente.
9 chattr +c archivo
10
11 #asegura que el programa ignore borrar los archivos durante la copia de seguridad
12 chattr +d archivo
13
14 #convierte el archivo en invariable, por lo que no puede ser eliminado, alterado
15 chattr +i archivo
16
17 #permite que un archivo sea borrado de forma segura.
18 chattr +s archivo
19
20 #asegura que un archivo sea modificado, los cambios son escritos en modo sincronizado
21 chattr +S archivo
22
23 #te permite recuperar el contenido de un archivo aún si este está cancelado.
24 chattr +u archivo
```

ARCHIVOS TAR

```
1 #mostrar contenido de archivo tar
2 tar -tf archivo.tar
3
4 #crear archivo tar
5 tar -cvf archivo.tar directorio1
6
7 #crear archivo tar compuesto por varios archivos y directorios
8 tar -cvf archivo.tar archivo1 archivo2 directorio1
9
10 #crear archivo tar comprimido en bzip2
11 tar -cvfj archivo.tar.bz2
12
13 #crear archivo tar comprimido en gzip
14 tar -cvfz archivo.tar.gz directorio1
15
16 #extraer archivo tar
17 tar -xvf archivo.tar
18
19 #extraer archivo tar en directorio
20 tar -xvf archivo.tar -C /directorio
21
22 #extraer archivo tar preservando permisos de usuario
23 tar -xpvzf archivo.tar
24
25 #descomprimir archivo tar comprimido en bzip2
26 tar -xvfj archivo.tar.bz2
27
28 #descomprimir archivo tar comprimido en gzip
29 tar -xvfz archivo.tar.gz
```

ARCHIVOS ZIP

```
1 #crear un archivo comprimido en zip.  
2 zip archivo.zip directorio1  
3  
4 #comprimir en zip compuesto por varios archivos y directorios  
5 zip -r archivo.zip archivo1 archivo2 directorio1  
6  
7 #descomprimir un archivo zip  
8 unzip archivo.zip
```

ARCHIVOS BZ2

```
1 #comprimir archivo bz2  
2 bzip2 archivo  
3  
4 #descomprimir archivo bz2  
5 bunzip2 archivo.bz2
```

ARCHIVOS GZ

```
1 #comprimir archivo gz  
2 gzip archivo  
3  
4 #descomprimir archivo gz  
5 gunzip archivo.gz  
6  
7 #descomprimir archivo gz con compresion maxima  
8 gzip -9 archivo
```

ARCHIVOS RAR

```
1 #crear archivo rar  
2 rar a archivo.rar directorio1  
3  
4 #crear archivo zip compuesto por varios archivos y directorios  
5 rar a archivo.rar archivo1 archivo2 directorio1  
6  
7 #descomprimir archivo rar.  
8 unrar x archivo.rar
```

INSTALADORES DE PAQUETES, Y REPOSITORIOS

```
1 #mostrar bibliotecas requeridas por programa o comando  
2 ldd programa  
3  
4 #descargar de Github  
5 git clone git://github.com/directorio/proyecto.git
```

PAQUETES DEB (DEBIAN Y DERIVADOS)

```
1 #instalar / actualizar un paquete deb  
2 dpkg -i paquete.deb  
3  
4 #eliminar un paquete deb del sistema  
5 dpkg -r paquete  
6  
7 #mostrar todos los paquetes deb instalados en el sistema  
8 dpkg -l  
9  
10 #mostrar todos los paquetes deb con un nombre  
11 dpkg -l | grep nombre  
12  
13 #obtener información en un paquete instalado en el sistema  
14 dpkg -s paquete  
15  
16 #mostar lista de archivos dados por un paquete instalado en el sistema  
17 dpkg -L paquete  
18  
19 #mostrar lista de archivos dados por un paquete sin instalar  
20 dpkg -contents paquete.deb  
21  
22 #verificar a que paquete pertenece a un archivo  
23 dpkg -S archivo
```

ACTUALIZADOR DE PAQUETES APT (UBUNTU Y DERIVADOS)

```
1 #instalar / actualizar un paquete deb.  
2 apt-get install paquete  
3  
4 #añadir repositorio  
5 sudo sh -c 'echo repositorio' >> /etc/apt/sources.list  
6 #tambien  
7 sudo add-apt-repository ppa:repositorio  
8  
9 #actualizar lista de paquetes  
10 apt-get update  
11  
12 #actualizar paquetes instalados  
13 apt-get upgrade  
14  
15 #eliminar un paquete del sistema  
16 apt-get remove paquete  
17  
18 #buscar un paquete  
19 apt-cache search paquete  
20  
21 #verificar resolución de dependencias  
22 apt-get check  
23  
24 #limpiar cache desde paquetes descargados  
25 apt-get clean
```

OPERACIONES CON SISTEMAS DE ARCHIVOS

ANÁLISIS DEL SISTEMA DE ARCHIVOS

```
1 #Comprobar integridad de archivos en el disco hdal en sistema Linux
2 fsck /dev/hdal
3
4 #Comprobar integridad de archivos en el disco hdal en sistema ext2
5 fsck.ext2 /dev/hdal
6
7 #Comprobar integridad de archivos en el disco hdal en sistema ext3
8 fsck.ext3 /dev/hdal
9
10 #Comprobar integridad de archivos en el disco hdal en sistema Fat
11 fsck.vfat /dev/hdal
12
13 #Comprobar integridad de archivos en el disco hdal en sistema Dos
14 fsck.msdos /dev/hdal
15
16 #Comprobar bloques defectuosos en el disco hdal.
17 badblocks -v /dev/hdal
```

FORMATEAR SISTEMA DE ARCHIVOS

```
1 #formatear hda en sistema Linux
2 mkfs /dev/hdal
3
4 #formatear hda en sistema FAT32
5 mkfs -t vfat 32 -F /dev/hdal
6
7 #formatear hda en sistema ext2
8 mke2fs /dev/hdal
9
10 #formatear hda en sistema ext3
11 mke2fs -j /dev/hdal
```

MONTAR SISTEMAS DE ARCHIVOS

```
1 #montar un disco duro hda2
2 mount /dev/hda2 /mnt/hda2
3
4 #montar un disquetera
5 mount /dev/fd0 /mnt/floppy
6
7 #montar un cdrom o dvdrom
8 mount /dev/cdrom /mnt/cdrom
9
10 #montar cd regrabable dvdrom
11 mount /dev/hdc /mnt/cdrecorder
12
13 #montar un cd o dvd regrabable
14 mount /dev/hdb /mnt/cdrecorder
15
16 #montar un usb pen-drive o una memoria
17 mount /dev/sdal /mnt/usbdisk
18
19 #montar un archivo o una imagen iso
20 mount -o loop file.iso /mnt/cdrom
21
22 #desmontar un dispositivo llamado hda2
23 umount /dev/hda2
24
25 #forzar el desmontaje (cuando el dispositivo está ocupado)
26 fuser -km /mnt/hda2
```

IMÁGENES ISO Y GRABADORES DE CDROM

```
1 #montar una imagen iso
2 mount -o loop cd.iso /mnt/iso
3
4 #crear imagen iso de cdrom en disco
5 mkisofs /dev/cdrom > cd.iso
6
7 #crear imagen iso de un directorio
8 mkisofs -J -allow-leading-dots -R -V "Label CD" -iso-level 4 -o ./cd.iso data_c
9
10 #grabar imagen iso en cdrom
11 cdrecord -v dev=/dev/cdrom cd.iso
12
13 #limpiar o borrar un cd regrabable
14 cdrecord -v gracetime=2 dev=/dev/cdrom -eject blank=fast -force
```

TRABAJO CON LA SWAP

```
1 #crear archivo de sistema swap en hda3
2 mkswap /dev/hda3
3
4 #activar particion swap en hda3
5 swapon /dev/hda3
```

COPIAS DE SEGURIDAD Y BACKUPS

```
1 #hacer un backup completo de directorio
2 dump -0aj -f /tmp/archivo.bak /directorio
3
4 #realizar backup incremental de directorio
5 dump -1aj -f /tmp/archivo.bak /directorio
6
7 #restaurar un backup iterativamente
8 restore -if /tmp/archivo.bak
9
10 #sincronizar directorios
11 rsync -rogpav -delete /directorio1 /directorio2
12
13 #volcar contenido de disco duro a archivo
14 dd if=/dev/sda of=/tmp/archivo
15
16 #encontrar y copiar todos los archivos con extensión .txt de un directorio a otro
17 find /home/usuario -name '*.txt' | xargs cp -av -target-directory=/home/backup/
18
19 #encontrar todos los archivos con extensión .log y hacer un archivo bzip
20 find /var/log -name '*.log' | tar cv -files-from=- | bzip2 > log.tar.bz2
```

TELECOMUNICACIONES Y OPERACIONES DE RED

DESCARGA ARCHIVOS INTERNET

```
1 #descargar archivo desde paginaweb
2 wget www.paginaweb.com/archivo
3
4 #descargar un archivo con la posibilidad de parar la descargar y reanudar más tarde
5 wget -c www.paginaweb.com/archivo
6
7 #descargar paginaweb completa
8 wget -r www.paginaweb.com
```

OPERACIONES DE RED

```
1 #mostrar la configuración de Ethernet
2 ifconfig eth0
3
4 #activar interface eth0
5 ifup eth0
6
7 #deshabilitar interface eth0
8 ifdown eth0
9
10 #configurar una dirección IP
11 ifconfig eth0 192.168.1.1 netmask 255.255.255.0
12
13 #configurar eth0 en modo común para capturar paquetes (sniffing)
14 ifconfig eth0 promisc
```

```
--  
16 #activar la interface 'eth0' en modo dhcp  
17 dhclient eth0  
18  
19 #mostrar mesa de recorrido  
20 route -n  
21  
22 #configurar entrada predeterminada  
23 route add -net 0/0 gw IP_Gateway  
24  
25 #configurar ruta estatica para buscar la red '192.168.0.0/16'  
26 route add -net 192.168.0.0 netmask 255.255.0.0 gw 192.168.1.1  
27  
28 #eliminar ruta estatica  
29 route del 0/0 gw IP_gateway  
30  
31 #activar recorrido ip  
32 echo "1" > /proc/sys/net/ipv4/ip_forward  
33  
34 #mostrar nombre del host  
35 hostname  
36  
37 #buscar nombre del host para resolver el nombre a una dirección ip  
38 host www.paginaweb.com  
39  
40 #buscar nombre del host para resolver el nombre a una dirección ip y viceversa  
41 nslookup www.paginaweb.com  
42  
43 #mostrar estado de enlace de todas las interfaces  
44 ip link show  
45  
46 #mostrar estado de enlace de eth0  
47 mii-tool eth0  
48  
49 #mostrar estadisticas de tarjeta de red eth0  
50 ethtool eth0  
51  
52 #mostrar todas las conexiones de red activas y sus PID  
53 netstat -tup  
54  
55 #mostrar todos los servicios de escucha de red en el sistema y sus PID  
56 netstat -tupl  
57  
58 #mostrar todo el tráfico HTTP  
59 tcpdump tcp port 80  
60  
61 #mostrar redes inalámbricas  
62 iwlist scan  
63  
64 #mostrar configuración de una tarjeta de red inalámbrica  
65 iwconfig wlan0  
66  
67 #buscar en base de datos Whois  
68 whois www.paginaweb.com
```

SSH, SCP Y TUNNELING

```
1 #iniciar sesion ssh
2 ssh usuario@servidor.dominio.es
3
4 #iniciar sesion ssh con compatibilidad X11 (permite ejecutar tareas visuales)
5 ssh -X usuario@maquina
6
7 #iniciar sesion ssh en puerto determinado
8 ssh -p 15000 usuario@maquina
9
10 #copiar archivo mediante scp
11 scp /archivo usuario@servidor.dominio.es:/directorio
12
13 #creacion de tunel ssh
14 ssh -f usuario@servidor.dominio.es -L 2000:servidor.dominio.es:25 -N
15
16 #redireccion de puertos mediante tunneling
17 ssh -v -L4001:localhost:4001 usuario@servidor.dominio.es
```

REDES DE MICROSOFT WINDOWS (SAMBA)

```
1 #resolucion de nombre de red bios
2 nbtscan ip_addr
3
4 #resolucion de nombre de red bios
5 nmblookup -A ip_addr
6
7 #mostrar acciones remotas de un host en windows
8 smbclient -L ip_addr/hostname
```

4.- TRUCOS

WIDGETS

PROCESO

```
$ sudo apt-get install conky -y
```

```
$ wget -O /home/pi/.conkyrc
https://raw.githubusercontent.com/novaspirit/rpi_conky/master/rpi3_conkyrc
```

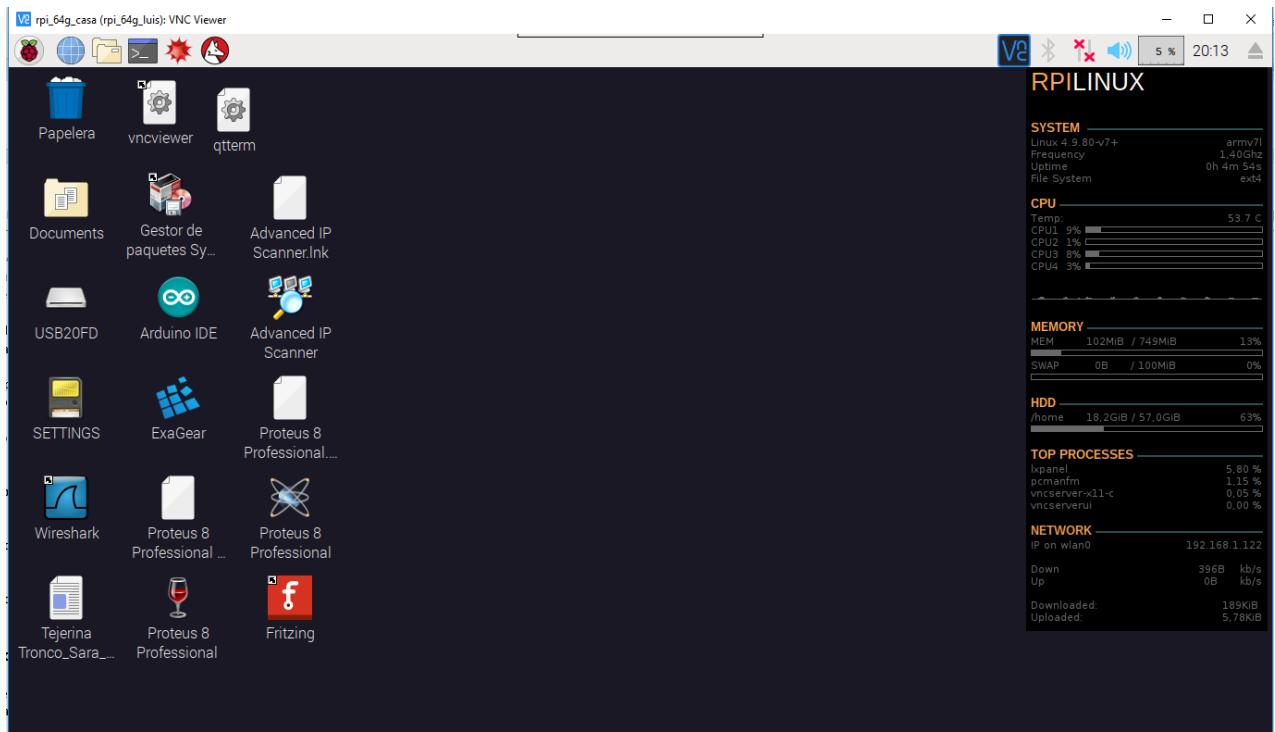
```
sudo nano /usr/bin/conky.sh
```

```
#!/bin/sh
(sleep 4s && conky) & exit 0
```

```
sudo nano /etc/xdg/autostart/conky.desktop
```

```
[Desktop Entry]
Name=conky
Type=Application
Exec=sh /usr/bin/conky.sh
Terminal=false
Comment=system monitoring tool.Categories=Utility;
```

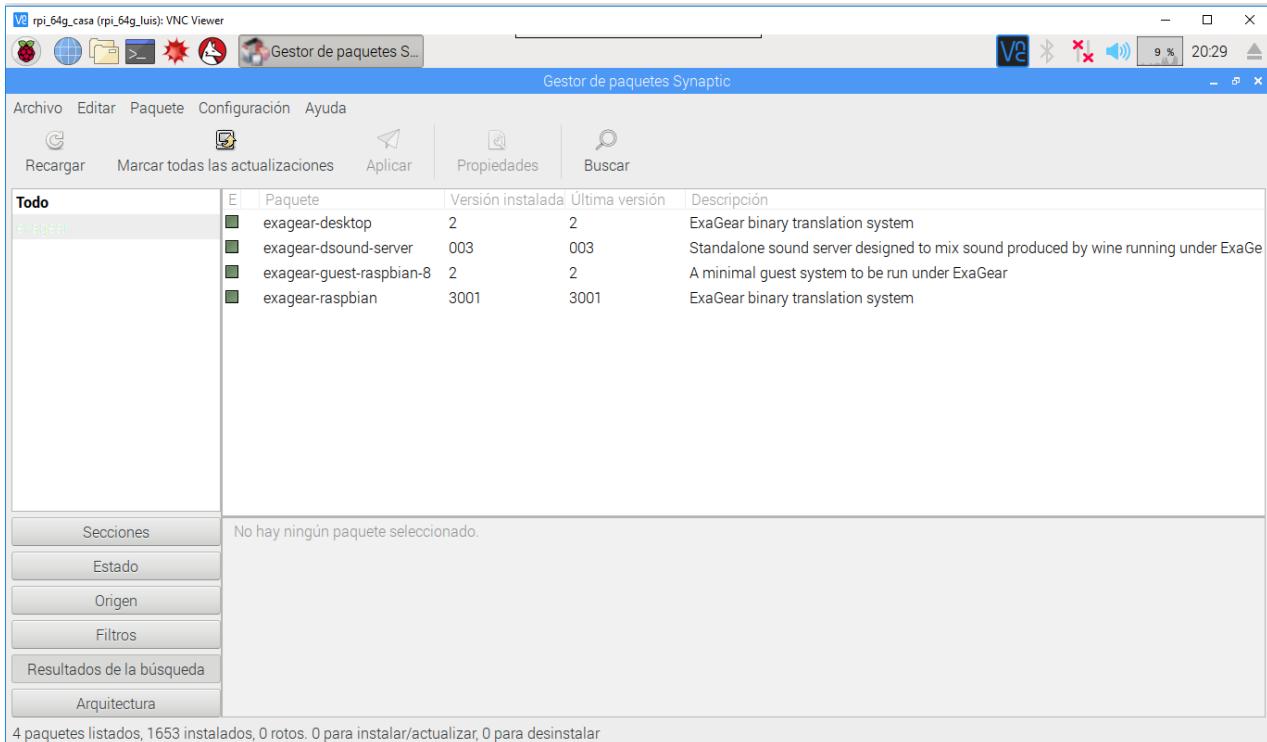
Reiniciar



WINDOWS EN RASPBERRY (EXAGEAR/WINE)

OBSOLETO

1. Desde Synaptic instalamos exagear

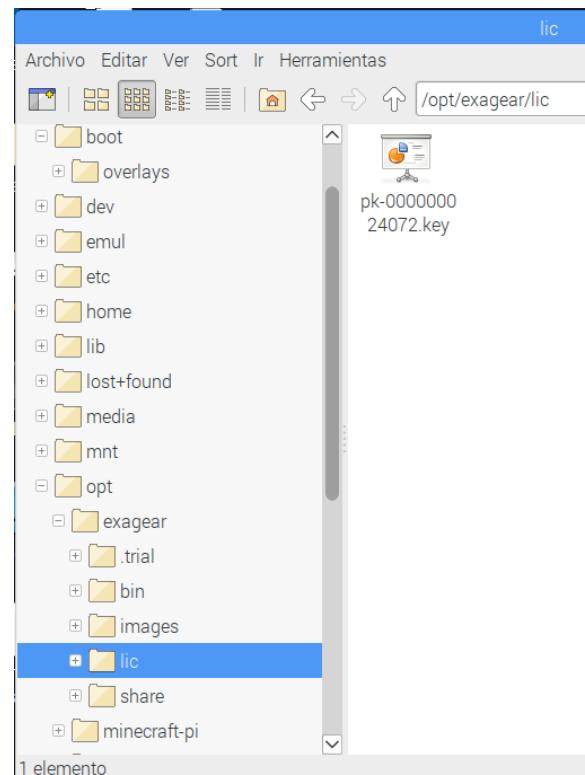


2. Instalamos la key

Copiamos el fichero en **/opt/exagear/lic**

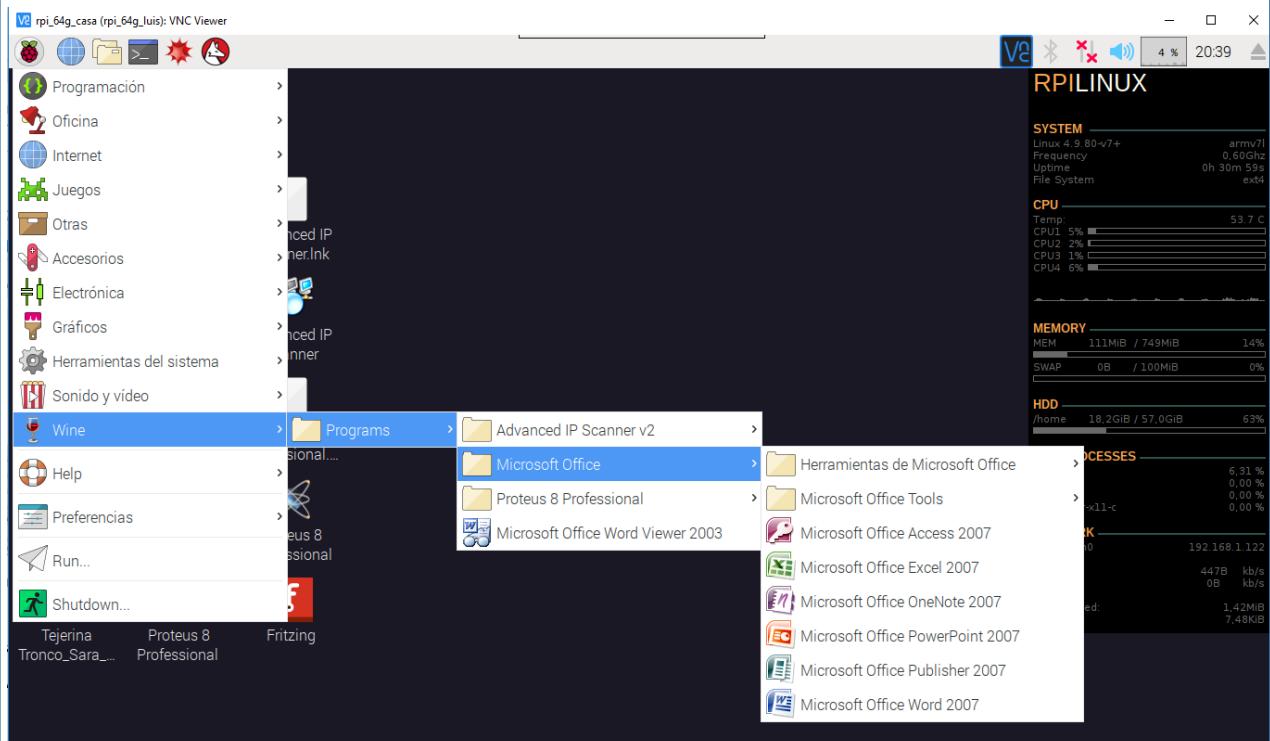
Para activar la licencia ejecutamos:

\$ sudo -E /opt/exagear/bin/actool



3. Instalamos wine

```
$ sudo apt-get install wine
```



4. Antes de instalar un programa X86, deberemos ejecutar exagear.

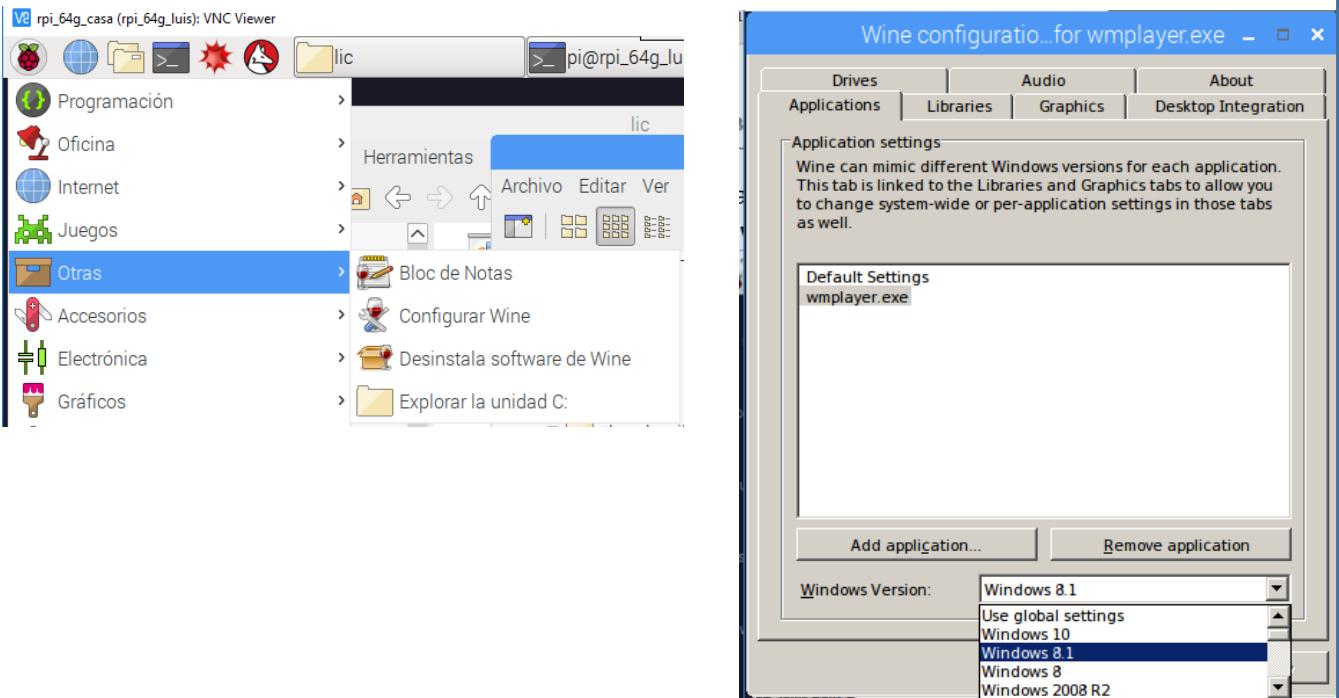
Podemos comprobar como la arquitectura de la raspberry responde a un núcleo **arm**, antes de la ejecución de exagear

```
pi@rpi_64g_luis:~ $ arch
armv7l
pi@rpi_64g_luis:~ $
```

Despues de la ejecución veremos como el núcleo cambia.

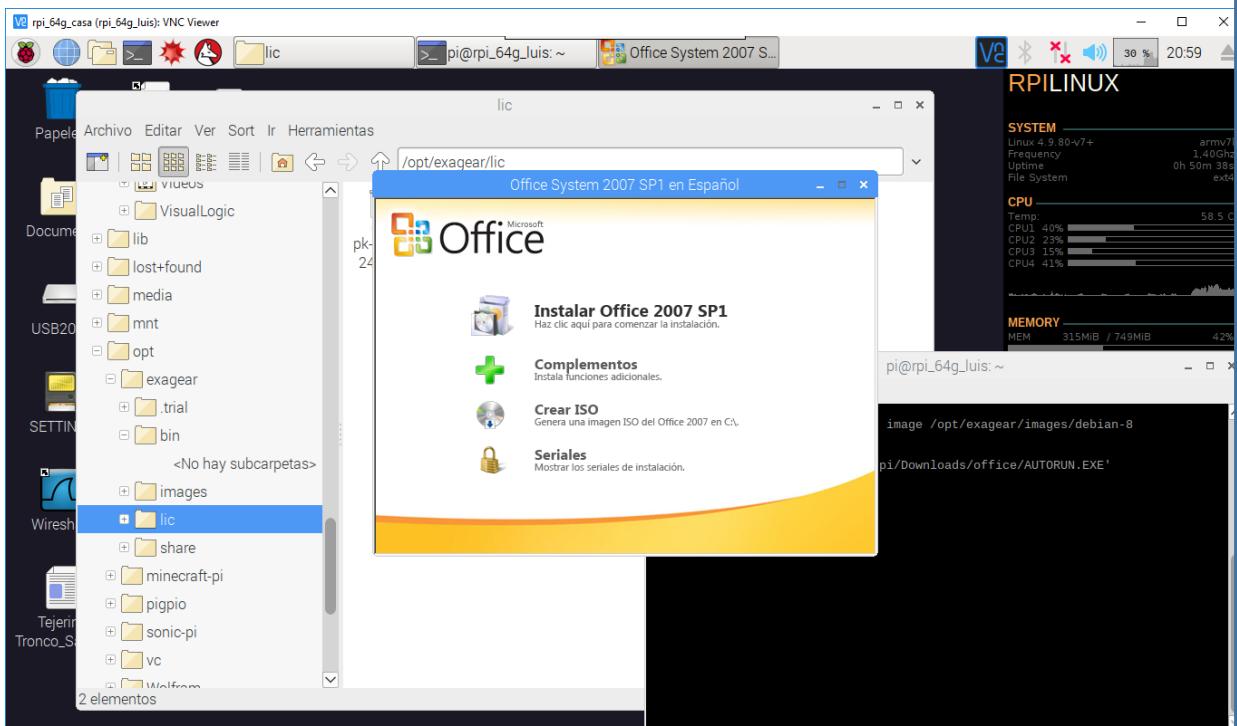
```
pi@rpi_64g_luis: ~
Archivo Editar Pestañas Ayuda
pi@rpi_64g_luis:~ $ exagear
Starting /bin/bash in the guest image /opt/exagear/images/debian-8
pi@rpi_64g_luis:~ $ arch
i686
pi@rpi_64g_luis:~ $
```

5. Podemos configurar wine, de manera que a cada programa le asignemos un sistema operativo concreto.



6. Para instalar un programa, habiendo ejecutado **exagear** previamente, lo haremos con un comando parecido a esto:

```
$ wine '/home/pi/Downloads/office/AUTORUN.EXE'
```

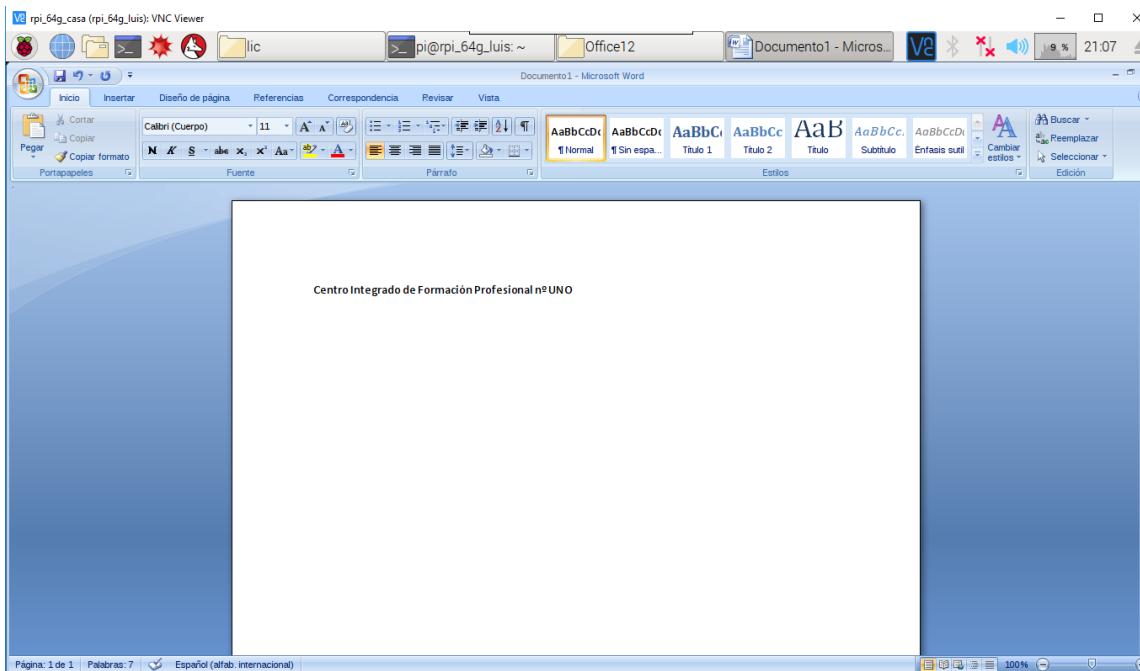


Al poner comillas, simples o dobles, podemos escribir rutas con espacios.

7. Para ejecutar un programa:

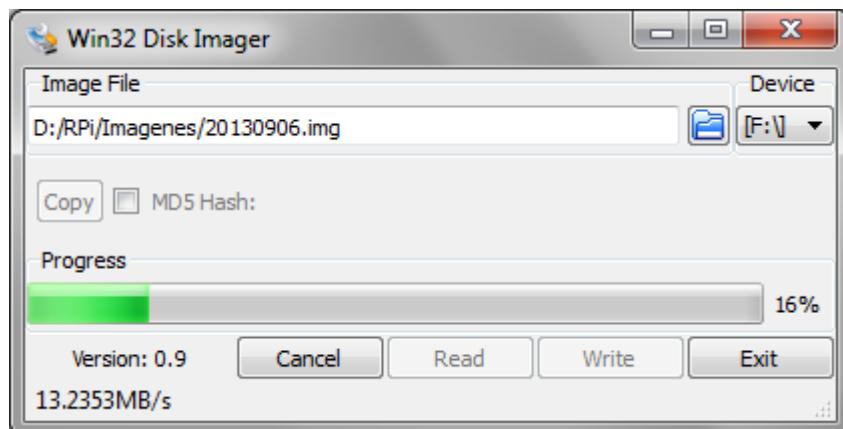
OBSOLETO

\$ wine 'c:/Program Files/Microsoft Office/Office12/WINWORD.EXE'



CÓMO CLONAR LA TARJETA SD DE LA RASPBERRY PI

En Windows insertamos nuestra tarjeta SD en la ranura del ordenador (o en un adaptador USB en mi caso) e iniciamos **Win32 Disk Imager**. Seleccionamos el fichero de imagen a crear, en nuestro ejemplo: D:/RPi/Imagenes/20130906. No hay que olvidar seleccionar la unidad asignada a la tarjeta SD. Pulsamos "Read" para crear la imagen y listo:



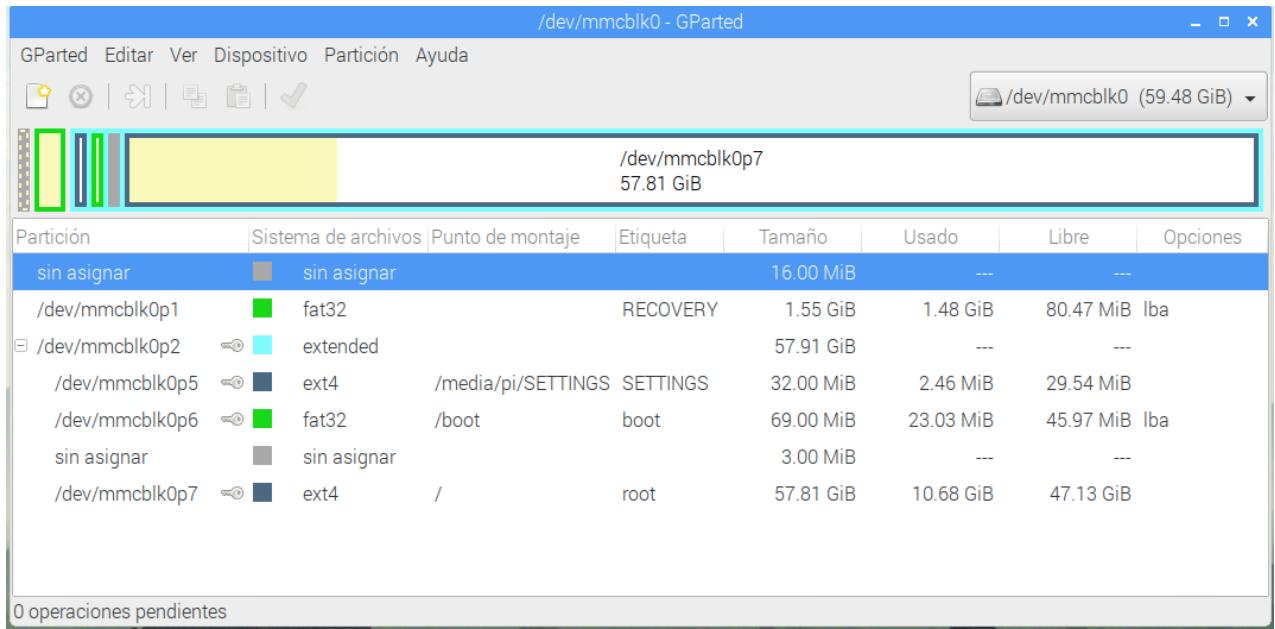
Ahora si queremos recuperar la imagen simplemente sería formatear la tarjeta SD, con **SDFormatter**, seleccionar la imagen creada y pulsar el botón "**Write**".

Y en Linux lo haríamos con el comando:

\$ dd if=/dev/sdx of=/path/to/image bs=1M...

INSTALAR GPARTED

\$ sudo apt-get install gparted



SISTEMAS DE ARCHIVOS

FAT32 (FILE ALLOCATION TABLE)

FAT es un sistema de archivos desarrollado para MS-DOS. Es el sucesor de FAT16, que a su vez es sucesor de FAT12. El tamaño máximo soportado por este sistema es de **4 GB-1 Byte** y se utiliza para el intercambio de datos entre distintos sistemas operativos de un mismo equipo. Además, también es utilizado en tarjetas de memoria y dispositivos similares.

Si intentamos meter en un pendrive de 16 GB en sistema FAT32 un archivo de 10 GB, no nos dejará, pues este archivo supera esos 4GB-1Byte.

NTFS (NEW TECHNOLOGY FILE SYSTEM)

Está incluido en las versiones de Windows 2000, Windows XP, Windows Server 2003, Windows Server 2008, Windows Vista, Windows 7 y Windows 8. El tamaño mínimo recomendado para las particiones de este tipo de sistemas de archivos es de 10 GB, siendo posibles tamaños mayores. Además, a diferencia de FAT32, distingue entre mayúsculas y minúsculas.

En cuanto al rendimiento, NTFS es mucho más rápido en el acceso a los archivos que una partición tipo FAT. Esto se debe a que utiliza un árbol binario de alto rendimiento para localizar los archivos. El tamaño límite de una partición es de 17×10^9 Bytes.

Por otra parte, los sistemas de archivos NTFS son más estables que los FAT. Además, NTFS cifra archivos, cosa que FAT no hace.

EXT3 (THIRD EXTENDED FILESYSTEM)

Es un sistema de archivos principalmente utilizado en distribuciones Linux con registro por diario (journaling). Está siendo reemplazado por su sucesor, ext4, aunque todavía se utiliza.

¿Qué es el journaling?

El journaling se basa en llevar un registro diario en el que se almacena la información necesaria para restablecer los datos del sistema afectados por un cambio, en caso de que falle.

Con el sistema de archivos ext3 se obtiene una gran reducción del tiempo necesario para recuperar un sistema de ficheros después de una caída. Por tanto, podemos decir que sus principales objetivos son la disponibilidad y confiabilidad.

Imaginemos que apagamos nuestro equipo incorrectamente. Ext3 se encargará de que al encenderlo lo tengamos igual que lo dejamos antes de apagarlo.

EXT4 (FOURTH EXTENDED FILESYSTEM)

Es una mejora compatible de ext3 que utiliza menos CPU y mejora la velocidad de lectura y escritura.

Además, soporta volúmenes de hasta 1024 PiB (PebiByte) (1 PiB = 2^{50} Bytes). Como acabo de decir, mejora la velocidad de lectura y escritura en comparación con ext3, pero es más lento en la eliminación de archivos.

En ext4 se introducen los exents, que se utilizan para reemplazar al tradicional esquema de bloques utilizado por ext2 y ext3. Los exents mejoran el rendimiento al trabajar con ficheros de gran tamaño.

FORMATEAR Y MONTAR UN PENDRIVE/DISCO DURO EN EXT4

Para almacenar ficheros (por ejemplo, las descargas de **Transmission**) viene muy bien utilizar un pendrive o, mejor aún, un disco duro externo USB (ya sea un HDD o un SSD). Haciéndolo, evitaremos saturar la tarjeta SD tanto de datos como de carga de trabajo, aparte de poder llevar los ficheros de un ordenador a otro sin tener que apagar la Raspberry. Veamos el proceso para montar un pendrive o un HDD/SSD en ella y hacer que se monte luego automáticamente cada vez que iniciemos el sistema.

En los siguientes párrafos vamos a poner como ejemplo el montaje de un pendrive, pero si queremos usar un disco duro externo, el procedimiento será exactamente el mismo, ya que en ambos casos se trata de dispositivos USB. Lo único que deberemos hacer, si lo deseamos, será cambiar la abreviatura **pen** por las abreviaturas **hdd** o **ssd** (según el caso) con el fin de identificar mejor el tipo de dispositivo de almacenamiento USB que vamos a usar. Y, por supuesto, tener en cuenta que cuando leamos "pendrive" será lo mismo que si dijera "disco duro".

Aclarado lo anterior, lo primero que tenemos que hacer es conectar el pendrive a uno de los puertos USB y averiguar el nombre del dispositivo, es decir, el nombre que le asigna Linux (**Raspbian** en este caso). Lo haremos con este comando:

```
sudo fdisk -l
```

Al final (en la parte de abajo de la pantalla) aparecerá el nombre. Si no tenemos ningún otro dispositivo USB conectado, muy probablemente será **sda1**.

A continuación, formatearemos el pendrive. Para usarlo con **Raspbian** (o cualquier otra distribución de Linux) lo mejor es hacerlo con el sistema de ficheros **ext4**. Podemos realizar esto fácilmente desde la propia consola de comandos con la siguiente orden:

```
sudo mkfs.ext4 /dev/sda1
```

Concluido el formateo, hay que crear un punto de montaje para nuestro pendrive, que será la carpeta desde la que accederemos a él. Para ello, crearemos una carpeta llamada, por ejemplo, **pen** dentro del directorio **/media**:

```
cd /media
```

```
sudo mkdir pen
```

En este momento, ya podemos montar el pendrive en dicha carpeta con el comando **sudo mount /dev/sda1 /media/pen**. Pero para no tener que hacer esto cada vez que iniciemos el sistema, lo mejor es lograr que se monte automáticamente al arrancar la Raspberry. Para hacerlo, tenemos que modificar el fichero **/etc/fstab**:

```
sudo nano /etc/fstab
```

Añadiremos al final del mismo una nueva línea, en la que indicaremos el nombre del dispositivo, la carpeta de montaje y el sistema de ficheros usado. La línea debe quedar así (los espacios se insertan con el tabulador):

```
/dev/sda1 /media/pen ext4 defaults 0 0
```

Guardamos el fichero (**Ctrl+o**, **Intro**, **Ctrl+x**) y reiniciamos el sistema:

```
sudo reboot
```

A partir de ahora, cada vez que el sistema se inicie, el pendrive se montará automáticamente en la carpeta indicada.

Hecho lo anterior, ya podemos crear las subcarpetas que necesitemos dentro de la carpeta **/media/pen**. Para hacerlo, nos situamos en ella y creamos, por ejemplo, una carpeta para descargar los torrents. Finalmente, le damos todos los permisos a dicha carpeta:

```
cd /media/pen  
sudo mkdir torrent  
sudo chmod 777 torrent
```

Usar el formato NTFS de Windows

Formatear el pendrive en **ext4** tiene el problema de que dicho formato no es reconocido por Windows, de manera que no podremos extraer el dispositivo y conectarlo en el PC con Windows para, por ejemplo, copiar archivos directamente. Para solucionar esto podríamos usar FAT32, pero este formato es antiguo y no acepta ficheros de más de 4 GB. Lo ideal entonces, si queremos usar el pendrive en el sistema operativo de Microsoft, es formatearlo en NTFS, bien con la herramienta **GParted** de Linux o desde el propio Windows.

Para hacerlo desde la propia Raspberry, antes hay que instalar el paquete **ntfsprogs**:

```
sudo apt-get install ntfsprogs
```

Y entonces ya podremos proceder al formateo:

```
sudo mkfs.ntfs /dev/sda1
```

Una vez formateado el pendrive con el nuevo formato, debemos instalar después en la Raspberry el paquete que le permitirá trabajar con el sistema de ficheros NTFS:

```
sudo apt-get install ntfs-3g
```

Luego tendremos que modificar adecuadamente el fichero **/etc/fstab** para adaptarlo al nuevo formato del pendrive. Por lo tanto, cambiaremos la línea

```
/dev/sda1 /media/pen ext4 defaults 0 0
```

por esta otra:

```
/dev/sda1 /media/pen ntfs-3g defaults 0 0
```

Lo guardamos con la modificación hecha (**Ctrl-o, Intro, Ctrl+x**) y reiniciamos la Raspberry:

```
sudo reboot
```

Recordemos que habrá que crear de nuevo la carpeta **torrent** en el pendrive y otorgarle todos los permisos:

```
cd /media/pen  
sudo mkdir torrent  
sudo chmod 777 torrent
```

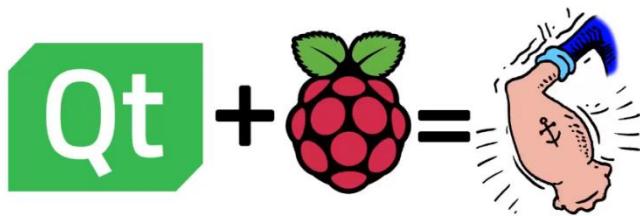
INSTALAR Qt5

```
$ sudo apt-get install update  
$ sudo apt-get install upgrade  
$ sudo apt-get install qt5-default  
$ sudo apt-get install qtcreator
```

Qt es un **Framework** (Conjunto de metodologías de trabajo y unas herramientas para llevarlas a cabo).

Sirve para desarrollar programas:

- Con interfaz gráfica de usuario.
- Sin interfaz como línea de comandos
- Y otras capacidades como, Acceso a base de datos, lectura de archivos de varios tipos, capacidades de redes y multihilo.



<https://medium.com/@hektorprofe/primeros-pasos-en-pyqt-5-y-qt-designer-programas-gr%C3%A1ficos-con-python-6161fba46060>

5.- CÁMARA

Vamos a documentar como hacer una transmisión de video con la Camara Raspberry Pi a un PC. Como objetivo adicional nos proponemos crear un dispositivo autónomo que transmita imágenes mediante wifi en vivo, similar a una cámara wireless de seguridad.

Veremos que la Raspberry Pi camera board es un módulo diseñado para ser integrado con facilidad en el Pi.

La cámara posee una relación calidad-precio notable:

- Tamaño 25mm x 20mm x 9mm
- Peso 3g
- Bajo consumo
- 5 Megapixels
- 2592 x 1944 pixeles de imágenes estáticas
- Soporta video 1080p30, 720p60 y 640x480p60/90
- Cuesta menos de 25 Euros



INSTALACIÓN

La instalación es muy sencilla. La banda se inserta en el conector plano entre la Ethernet y el HDMI.

Primero es necesario levantar una pestaña tirando hacia arriba.

Después se inserta la faja de la cámara, con la parte plateada orientada hacia el puerto HDMI.

Cuando esté bien colocada, se baja de nuevo la pestaña y listo.



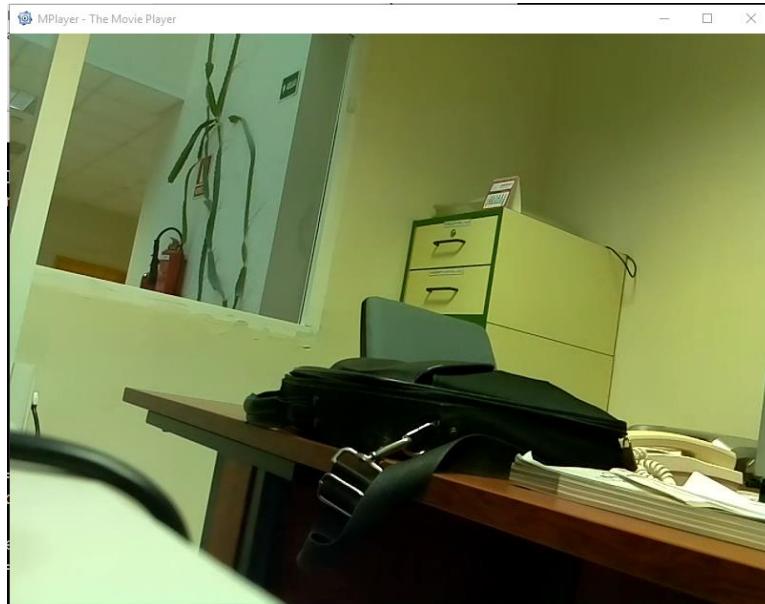
Habilitamos la cámara en la Raspberry.

.- FOTOS → (\$raspistill)

Escribamos en la consola el siguiente comando:

\$sudo raspistill -v -o test.jpg

Que muestra 5 segundos de vista previa por defecto y toma una foto, salvándola en el fichero test.jpg.



Hay que destacar que no es posible ver la vista previa desde el cliente remoto vnc.

-o ó –output el nombre del fichero y -t ó –timeout especifica los milisegundos de vista previa.

-d ó –demo ejecuta el modo demo que rota a través de varios efectos de imagen

.- VIDEO → (\$raspivid)

Captura 5s de video en formato h264:

\$raspivid -o video.h264

Capturar 10s de video:

\$raspivid -o video.h264 -t 10000

Captura 10s de video en modo demo:

\$raspivid -o video.h264 -t 10000 –d

Para más información acerca del uso de estos comandos, referirse a la bibliografía o a la ayuda de los propios comandos:

\$raspivid | less

\$raspistill | less

DESACTIVAR LED DE LA CÁMARA

La cámara incluye un LED en la esquina que se enciende cuando la cámara esta activa.

Este led se puede desactivar editando el fichero

\$sudo nano /boot/config.txt

y añadiendo la linea

\$disable_camera_led=1

- STREAMING PUNTO A PUNTO CON MPLAYER

OBsoleto

Al contrario que el protocolo anterior el stream punto a punto envía la señal del raspberry pi directamente a la IP del reproductor. Para ello necesitamos dos programas:

- **Netcat** es una utilidad de red que redirige flujos de lectura/escritura utilizando tcp/udp.
- **MPlayer** es un reproductor opensource que interpreta los formatos más habituales.
- Tomamos como ejemplo un PC con sistema Windows. Tenemos que instalar ambos programas, descargables desde → <https://sites.google.com/site/projectrobogobydownloads/>

1. ARRANCAR LA ESCUCHA EN EL PC

Para arrancar la escucha en el PC, ejecutar en el directorio de trabajo

```
C:\...>nc -l -p 5001 | mplayer -vo direct3d -fps 31 -cache 2048 -nosound -framedrop -
C:\...>nc -l -p 5001 | mplayer -fps 60 -cache 2048 -
```

```
D:\DOWNLOAD\REDES>nc -l -p 5001 | mplayer -fps 60 -cache 2048 -
MPlayer Redxii-SVN-r36251-4.6.3 (C) 2000-2013 MPlayer Team
Custom build by Redxii, http://smplayer.sourceforge.net
Compiled against FFmpeg version N-52919-ge4723a8
Build date: Thu May 9 02:07:55 EDT 2013

Playing -.
Reading from stdin...
Cache fill: 0.00% (0 bytes)
```

El significado es que **netcat** escucha por el puerto 5001 y todo lo que reciba se lo pasará al **mplayer** por la entrada estándar. **MPlayer** espera una retransmisión de 60 frames por segundo y guardará el contenido en un buffer de dos megas.

2. ACTIVAR STREAM EN LA RASPI

En la parte del Pi necesitamos descargar el software:

```
$sudo apt-get install mplayer netcat
```

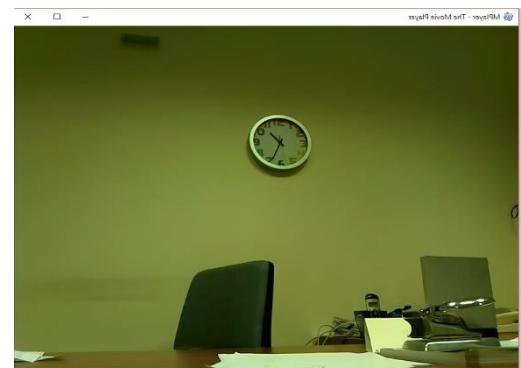
y activamos la cámara para enviar el stream de video con el comando

```
$sudo raspivid -o - -vf t 3600000 -w 800 -h 600 -fps 31 -n | nc <IPdePC> 5001
$ sudo raspivid -t 999999 -w 640 -h 480 -hf -vf -n -o - | nc <IPdePC> 5001
```

El programa **raspivid** envía durante 1000 segundos una imagen de 640×480 píxeles volteada horizontal y verticalmente por la salida estándar y sin mostrar la pre visualización. El **netcat** lo envía a la ip del PC por el puerto 5001.

3. EN LA PARTE DEL PC

Se abre automáticamente el reproductor **mplayer**:



.- STREAMING CON CÁMARA USB

<https://geekytheory.com/video-streaming-live-con-raspberrypi-y-playstation-eye/>

Conectaremos nuestra Webcam USB a uno de los dos puertos USB disponibles. Para comprobar que la cámara ha sido detectada, ejecutaremos los siguientes comandos desde una terminal:

```
$ ls -l /dev/video*
```

Aviso! Si **no** te aparece **/dev/video0** en el listado, significa que tu Webcam **no** ha sido detectada, o bien, **no** es compatible. Hay algunas webcams que es necesario conectarlas mediante un HUB USB con **alimentación externa**, ya que consumen más corriente de lo que la Raspberry Pi suministra.

```
$ lsusb
```

Con el último comando obtendremos información de la Webcam que hemos conectado: aparecerá en una línea Bus, Device, ID y **nombre de la Webcam** que ha sido detectada por el sistema.

Instalación de librerías y utilidades necesarias:

Una vez hayamos comprobado que la Webcam ha sido detectada correctamente, procederemos a la instalación de las siguientes librerías que necesita la herramienta **MJPEG-Streamer** y la utilidad **subversion** que nos servirá más adelante para descargarla. Para ello ejecutaremos los siguientes comandos desde una terminal:

```
$ sudo apt-get install libjpeg8-dev
```

```
$ sudo apt-get install imagemagick
```

```
$ sudo apt-get install subversion
```

Descarga, compilación y ejecución de MJPG-Streamer:

Instaladas las librerías, el siguiente paso es descargar la herramienta necesaria para realizar el **streaming** de vídeo que captura nuestra Webcam. Para empezar, crearemos un nuevo directorio para trabajar más cómodamente. Para ello, ejecutaremos los siguientes comandos desde una terminal:

```
$ sudo mkdir mjpg
```

```
$ cd mjpg
```

Con el segundo comando hemos accedido al directorio que acabamos de crear (**mjpg**), el cual utilizaremos para alojar la herramienta **MJPEG-Streamer** que vamos a descargar. Para obtener dicha herramienta, situados en el directorio creado anteriormente, lo descargaremos mediante la ejecución del siguiente comando:

```
$ sudo svn co https://svn.code.sf.net/p/mjpg-streamer/code/mjpg-streamer/ mjpg-streamer
```

Se nos habrá creado un nuevo directorio con el nombre **mjpg-streamer**. Dentro se encontrarán los archivos necesarios para realizar su compilación, la cual vamos a realizar a continuación ejecutando el siguiente comando:

```
~/mjpg/mjpg-streamer $ sudo make
```

La compilación generará los archivos necesarios para ejecutar la herramienta **MJPG-Streamer**. Ya tenemos lista la herramienta para poder ser ejecutada, pero antes vamos a explicar en qué consiste esta herramienta.

¿Qué es MJPG-Streamer?

MJPG-Streamer es una aplicación que se ejecuta por línea de comandos. A grandes rasgos, se encarga de obtener **frames** JPG (imágenes) capturadas desde una cámara compatible y transmitirlas como **M-JPEG** (secuencia de vídeo) mediante el protocolo HTTP para poder visualizarlo en navegadores, VLC y otras herramientas.

¿Cómo funciona?

Su funcionamiento se basa en unos **plugins** de entrada y salida. Es decir, un *plugin* (de **entrada**) copia las imágenes JPEG a un directorio de acceso global, mientras que otro *plugin* (de **salida**) procesa las imágenes, sirviéndolas como un simple fichero de imagen, o bien, emite las mismas de acuerdo a los estándares MPG existentes.

Por lo tanto, con la compilación anterior, lo que hemos hecho ha sido generar éstos **plugins**. Vamos a explicar en qué consisten:

- **mjpg_streamer**: Herramienta de **línea de comandos**, que copia las imágenes JPG de un plugin de entrada, a uno o más plugins de salida.
- **input_uvc.so**: **Captura** los frames JPG de una Webcam conectada. (A una resolución máxima de 960x720 píxeles y un elevado frame rate (>=15fps), con poca carga sobre la CPU).
- **output_http.so**: **Servidor Web** HTTP 1.0. Sirve una única imagen JPG o bien las emite de acuerdo al estándar M-JPEG.

Como se ha explicado anteriormente, *MJPG-Streamer* funciona bajo **línea de comandos**, por lo que para iniciarla deberemos ejecutar el siguiente comando:

```
~/mjpg/mjpg-streamer $ ./mjpg_streamer -i "./input_uvc.so -d /dev/video0 -y" -o "./output_http.so -w ./www"
```

Ejecutada la herramienta, debería de aparecernos lo siguiente, como se observa en la imagen:

```
pi@raspberrypi:~/mjpg/mjpg-streamer $ ./mjpg_streamer -i "./input_uvc.so -d /dev/video0 -y" -o "./output_http.so -w ./www"
MJPG Streamer Version: svn rev: 3:172
i: Using V4L2 device.: /dev/video0
i: Desired Resolution: 640 x 480
i: Frames Per Second: 5
i: Format.....: YUV
i: JPEG Quality....: 80
Adding control for Pan (relative)
UVCIIOC_CTRL_ADD - Error: Inappropriate ioctl for device
Adding control for Tilt (relative)
UVCIIOC_CTRL_ADD - Error: Inappropriate ioctl for device
Adding control for Pan Reset
UVCIIOC_CTRL_ADD - Error: Inappropriate ioctl for device
Adding control for Tilt Reset
UVCIIOC_CTRL_ADD - Error: Inappropriate ioctl for device
Adding control for Pan/tilt Reset
UVCIIOC_CTRL_ADD - Error: Inappropriate ioctl for device
Adding control for Focus (absolute)
UVCIIOC_CTRL_ADD - Error: Inappropriate ioctl for device
mapping control for Pan (relative)
UVCIIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for Tilt (relative)
UVCIIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for Pan Reset
UVCIIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for Tilt Reset
UVCIIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for Pan/tilt Reset
UVCIIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for Focus (absolute)
UVCIIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for LED1 Mode
UVCIIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for LED1 Frequency
UVCIIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for Disable video processing
UVCIIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for Raw bits per pixel
UVCIIOC_CTRL_MAP - Error: Inappropriate ioctl for device
o: www-folder-path...: ./www/
o: HTTP TCP port.....: 8080
o: username:password.: disabled
o: commands.....: enabled
```

Como observaréis, han aparecido una serie de errores, pero **no son importantes**, ya que la herramienta se está ejecutando correctamente, y para prueba de ello, **¡vamos a acceder a ella!**

¿Cómo accedemos a la herramienta para ver nuestro vídeo streaming en directo?

Antes de nada, vamos a detenernos un momento a **leer los mensajes** que aparecen en la terminal:

```
MJPEG Streamer Version: svn rev:  
i: Using V4L2 device.: /dev/video0  
i: Desired Resolution: 640 x 480  
i: Frames Per Second.: 5  
i: Format.....: YUV  
i: JPEG Quality.....: 80  
o: www-folder-path....: ./www/  
o: HTTP TCP port.....: 8080  
o: username:password.: disabled  
o: commands.....: enabled
```

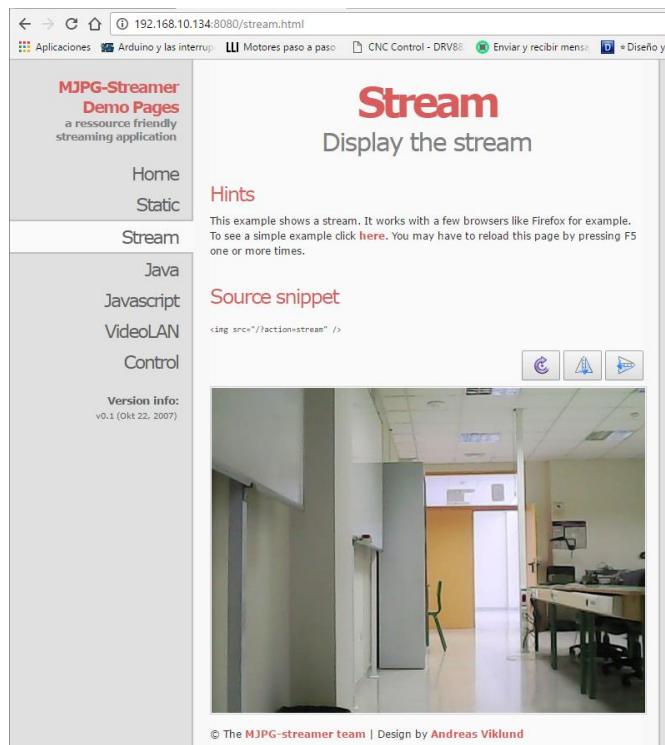
Como se puede observar, los mensajes son bastantes descriptivos:

1. Se nos notifica de que se está usando el dispositivo **/dev/video0** (inuestra Webcam conectada!)
2. La resolución con la que se está emitiendo vídeo: **640x480**
3. Se están tomando imágenes a una razón de: **5 FPS**
4. El formato de paleta de colores: **YUV**
5. Calidad de las imágenes JPEG: **80%**
6. Directorio de la aplicación Web de la herramienta: **./www**
7. Puerto TCP donde se emite el vídeo: **8080**
8. Usuario: contraseña (por si se quiere restringir el acceso): **disabled**
9. Comandos: **enabled**

A continuación, abriremos un navegador, y accederemos a la siguiente dirección:

```
http://IP_DE_NUESTRA_RASPBERRYPI:8080
```

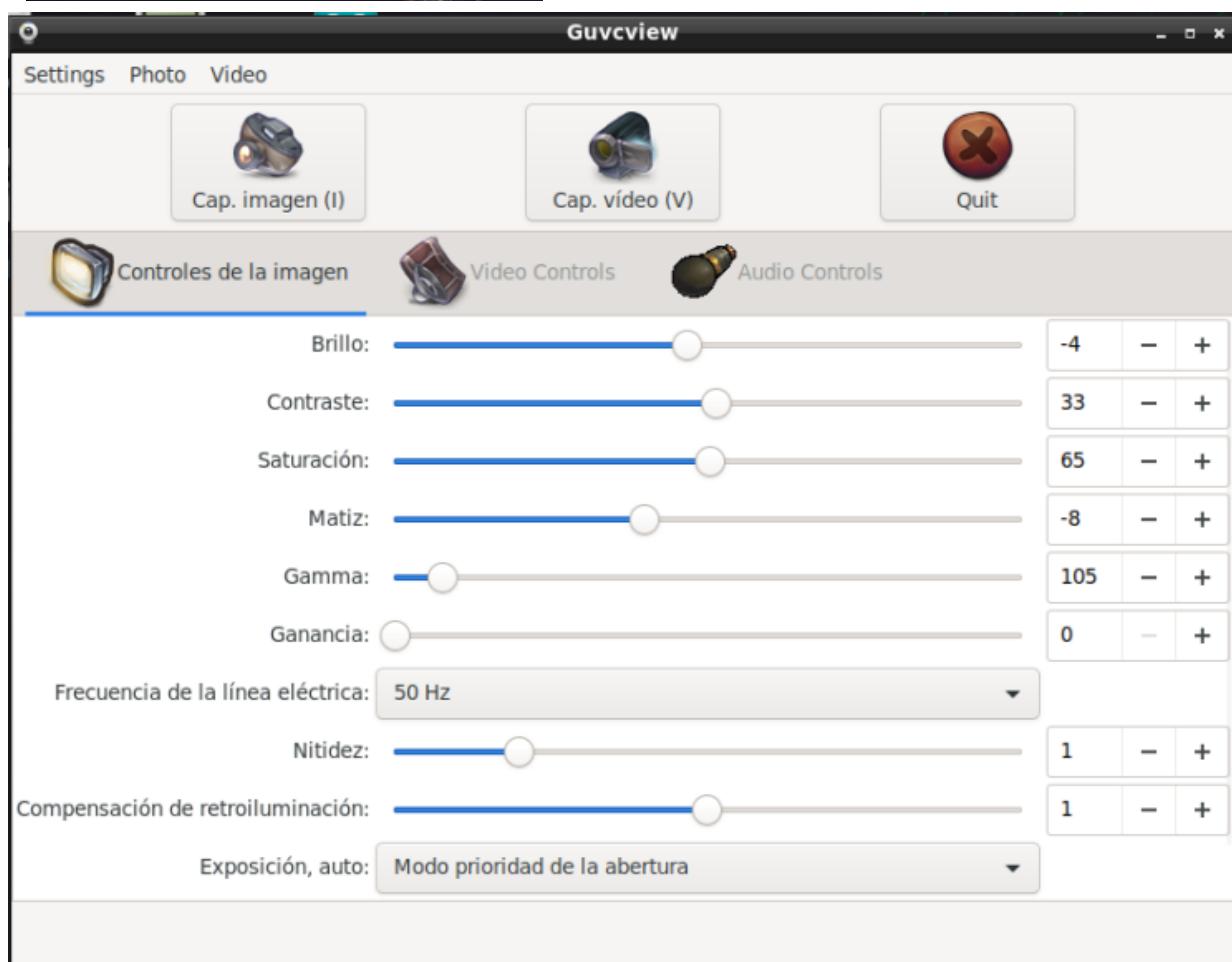
Ahora podremos acceder a las **distintas opciones** de las que dispone la aplicación web:



1. **HOME:** Página de inicio.
2. **STATIC:** Ejemplo de página web con una imagen estática capturada por la Webcam.
3. **STREAM:** Ejemplo de página web con la emisión del vídeo en directo.
4. **JAVA:** Ejemplo de emisión de vídeo realizado con un applet de Java.
5. **JAVASCRIPT:** Ejemplo de emisión de vídeo realizado con Javascript.
6. **VideoLAN:** Ejemplo para obtener la URL del vídeo emitido en directo y poder abrirlo mediante el programa VLC.
7. **CONTROL:** Se abrirá una nueva ventana (como aparece en la siguiente imagen) que nos permitirá modificar parámetros como brillo, saturación, contraste...

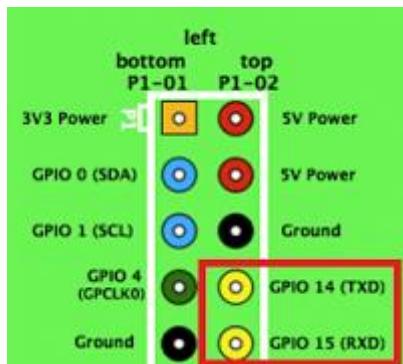
.- SOFTWARE CONTROL DE PARÁMETROS DE CÁMARAS USB/RASPICAM

`$ sudo apt-get install guvcview`



6.- PUERTO SERIAL

Las Raspberry incluyen una serie de pines de propósito general que pueden ser utilizados como entradas o salidas digitales entre los cuales se incluye un pequeño puerto "serial". Este puerto puede ser utilizado para enviar o recibir datos desde y hacia otros dispositivos. De fábrica viene configurado como un puerto de "consola" para monitorear el funcionamiento del RasPI.



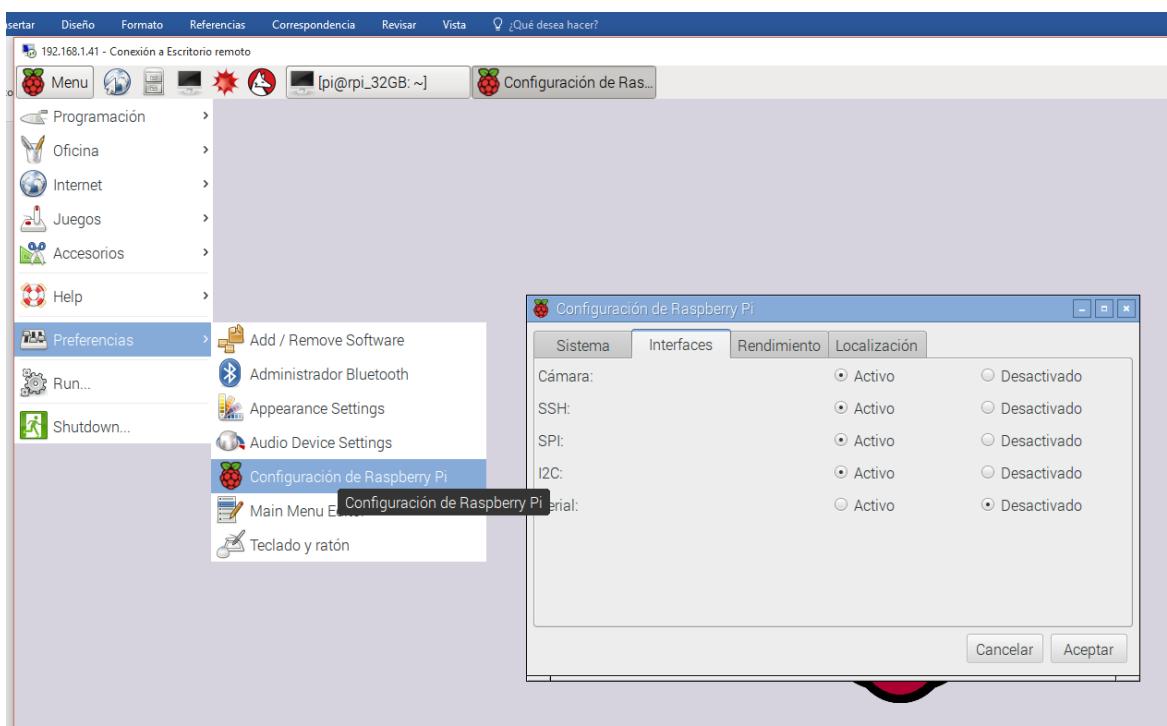
Antes de intentar comunicar algo por el puerto tenemos que cambiar un par de líneas de configuración:

1. Primero en la RasPI tenemos que modificar el archivo **/boot/cmdline.txt** y eliminar la parte que dice:

```
console=ttyAMA0,115200
```

2. Lo siguiente, es desactivar el puerto serial como consola.

(También podemos hacerlo tecleando el comando → **\$ sudo raspi-config**)



Una vez realizadas estas modificaciones reiniciamos nuestra RasPI. Estas modificaciones evitarán que el Kernel envíe mensajes a la línea serial.

PROBLEMA EN LA RASPBERRY PI 3

En la **Raspberry Pi 3**, hay más capacidades que en las versiones anteriores, como **WiFi** y **Bluetooth**.

ttyAMA0

La **UART** en la que había residido anteriormente la consola en serie, **GPIO 14 (TX)**, **GPIO 15 (RX)**, ahora se utiliza para la comunicación **Bluetooth**. Esta **UART** se reconoce como **ttyAMA0**.

El motivo por el que se necesita esta **UART**, es que se puede establecer su velocidad de bus sin depender de la velocidad de reloj del sistema, lo cual es necesario para el módulo de **Bluetooth**.

ttyS0 (Activada por defecto y sustituye a la ttyAMA0. Utiliza los pines GPIO 14 (TX), GPIO 15 (RX))

Existe una **mini UART** secundaria en la **Raspberry Pi 3**, pero con algunas restricciones. Una de ellas es que su velocidad de baudios está ligada a la frecuencia del sistema, lo que significa que variará a medida que la velocidad del procesador varía. Sin una velocidad de baudios fija es casi imposible conseguir una comunicación eficaz. Esta **mini UART** se reconoce como **ttyS0**.

Para activar la **mini UART**, debemos de añadir “**enable_uart=1**” en el fichero “**config.txt**”, situado en la carpeta “**/boot**”. Tras esto reiniciamos y, con la instrucción “**\$ ls /dev/tty***”, podemos comprobar la existencia de las dos **UART**. (**Por defecto ya existe “enable_uart=1”**)

```
pi@rpi_32GB:~ $ ls /dev/tty*
/dev/tty  /dev/tty19  /dev/tty3  /dev/tty40  /dev/tty51  /dev/tty62
/dev/tty0  /dev/tty2  /dev/tty30  /dev/tty41  /dev/tty52  /dev/tty63
/dev/tty1  /dev/tty20  /dev/tty31  /dev/tty42  /dev/tty53  /dev/tty7
/dev/tty10  /dev/tty21  /dev/tty32  /dev/tty43  /dev/tty54  /dev/tty8
/dev/tty11  /dev/tty22  /dev/tty33  /dev/tty44  /dev/tty55  /dev/tty9
/dev/tty12  /dev/tty23  /dev/tty34  /dev/tty45  /dev/tty56  /dev/ttyAMA0
/dev/tty13  /dev/tty24  /dev/tty35  /dev/tty46  /dev/tty57  /dev/ttyprintk
/dev/tty14  /dev/tty25  /dev/tty36  /dev/tty47  /dev/tty58  /dev/ttyS0
/dev/tty15  /dev/tty26  /dev/tty37  /dev/tty48  /dev/tty59
/dev/tty16  /dev/tty27  /dev/tty38  /dev/tty49  /dev/tty6
/dev/tty17  /dev/tty28  /dev/tty39  /dev/tty5  /dev/tty60
/dev/tty18  /dev/tty29  /dev/tty4  /dev/tty50  /dev/tty61
```

SOLUCIÓN

Si queremos utilizar la **UART** **ttyAMA0**, tal y como lo hacíamos antes en los **GPIO14** y **GPIO15**, deberemos de deshabilitar el **Bluetooth** del siguiente modo:

1. Añadimos “**dtoverlay=pi3-disable-bt**” al final del fichero “**config.txt**”.
2. En el terminal, ejecutamos “**\$ sudo systemctl disable hcuart**”.

Si lo queremos volver a habilitar:

Anulamos “#dtoverlay=pi3-disable-bt” al final del fichero “config.txt”.
Y ejecutamos “\$ sudo systemctl enable hcuart”.

3. En el fichero “**/lib/systemd/system/hcuart.service**”, donde aparezca “**ttyAMA0**” o “**serial1**”, lo substituimos por “**ttyS0**”.
4. En el fichero “**/boot/cmdline.txt**”, eliminamos la parte que dice “**console=ttyAMA0,115200**”.
5. Ejecutamos: “**\$sudo raspi-config**” y deshabilitamos la línea serie para que sea accesible el shell.

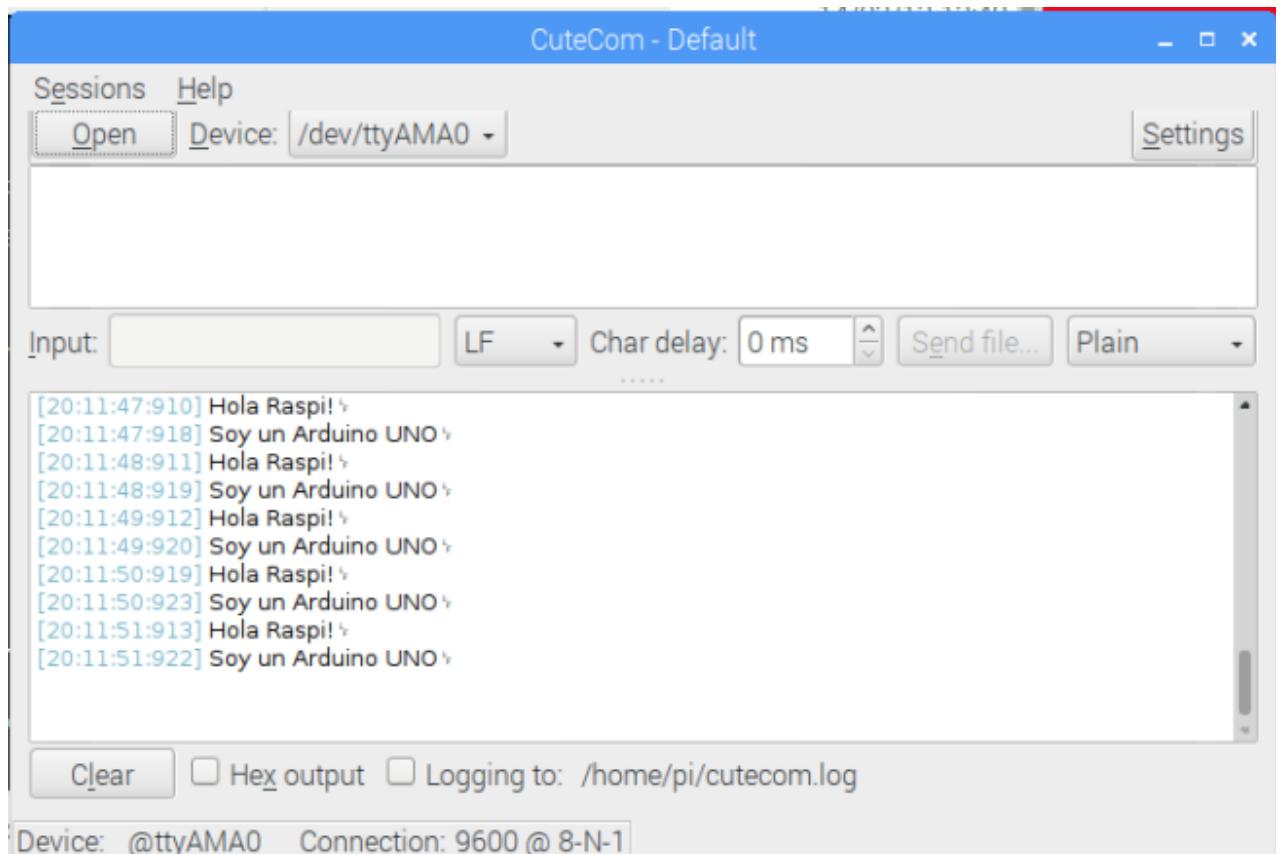
CuteCom (Software gráfico de comunicaciones)

CuteCom es una alternativa a hyperterminal en forma gráfica, en la actualidad se ejecuta en Linux, FreeBSD y Mac OS X. Es software libre y se distribuye bajo la versión GNU General Public License 2.

Para instalarlo:

```
# aptitude install cutecom
```

Y si no funciona → # sudo apt-get install cutecom



QtTerm (Software gráfico de comunicaciones)

OBSOLETO

1. Para poder compilar el código de la Raspberry Pi se necesitan estos paquetes instalados :

```
$ sudo apt-get update
```

```
$ aptitude search build-essential qt4-dev-tools
```

```
$ sudo apt-get install build-essential (para poder usar gcc, g++, make ...)
```

```
$ sudo apt-get install qt4-dev-tools
```

2. Descarga y compilación del **qtTerm** :

```
$ sudo wget http://qtterm.googlecode.com/files/3BpiQtTerm01.tar.gz
```

Si la orden anterior no enlaza, lo descargais en windows <https://code.google.com/archive/p/qtterm/downloads> y luego lo pasais a la Raspi.

```
$ sudo tar -zvxf 3BpiQtTerm01.tar.gz
```

3. **/3BpiQtTerm/qextserialport** \$ qmake

```
/3BpiQtTerm/qextserialport $ make
```

```
/3BpiQtTerm/qextserialport $ sudo make install
```

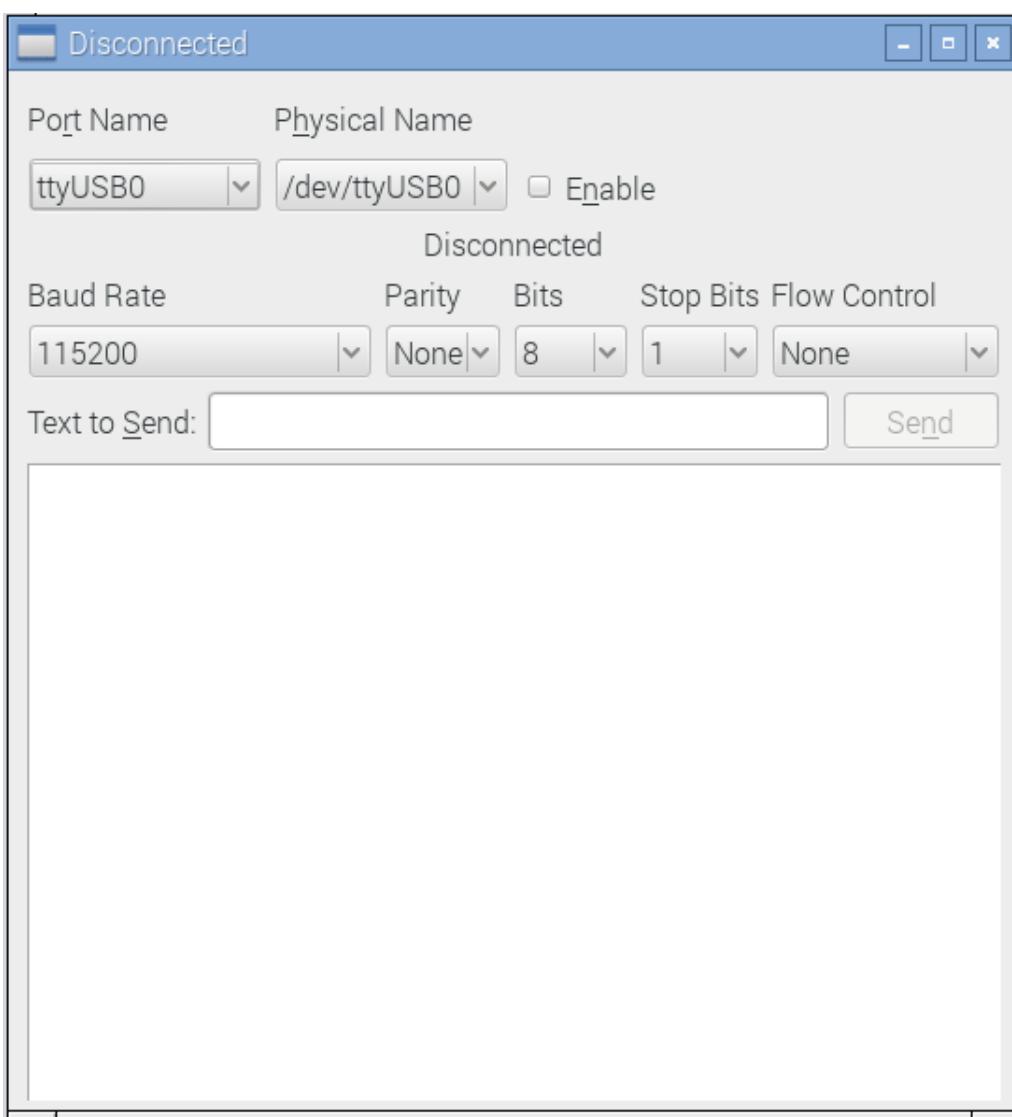
4. **qextserialport** ha sido instalado en Raspberry Pi. Ahora se tiene que compilar **qtTerm**

```
/3BpiQtTerm $ qmake
```

```
/3BpiQtTerm $ make
```

5. Ejecutamos qtTerm

6. **/3BpiQtTerm \$./qtterm**



Minicom

Minicom es un programa de texto basado en modem y un emulador de terminal para sistemas Unix. Su uso más común es para establecer conexiones seriales de forma remota, este programa tiene la misma utilidad que la Hyperterminal.

- 1- Comencemos con la instalación de minicom en nuestro sistema

```
$ sudo apt-get install minicom
```

2. Verificamos el archivo en donde se encuentra el puerto

```
$ dmesg | grep tty
```

```
[ 0.000000] console [tty0] enabled
[ 248.846457] usb 2-1: pl2303 converter now attached to ttyUSBO
[ 292.296285] pl2303 ttyUSBO: pl2303 converter now disconnected from ttyUSBO
[ 303.835209] usb 4-2: pl2303 converter now attached to ttyUSBO
```

Nota. - Generalmente nos muestra **tty0**, en mi caso muestra **ttyUSBO** ya que tengo conectado un adaptador de USB a DB9.

3. Accedemos a minicom para configurar el puerto que utilizaremos

```
$ minicom -s
```

4. Elegimos la opción **Configuración** de la Puerta Serial

```
+-----[Configuración]-----+
| Nombres de archivos y rutas
| Protocolos de transferencia de archivos
| Configuración de la puerta serial
| Modem y marcado de número
| Pantalla y teclado
| Salvar configuración como dfl
| Salvar configuración como..
| Salir
| Salir del Minicom
+-----+
```

5. Aparecen las opciones que se pueden configurar, tecleamos **A** y configuramos la dirección de la Puerta Serial, en mi caso **ttyUSBO**

```
+-----+
| A - Dispositivo Serial      : /dev/ttyUSBO
| B - Localización del Archivo de Bloqueo : /var/lock
| C - Programa de Acceso      :
| D - Programa de Salida      :
| E - Bps/Paridad/Bits       : 9600 8N1
| F - Control de Flujo por Hardware: Sí
| G - Control de Flujo por Software: No
|
| ¿Qué configuración alterar?
+-----+
| Pantalla y teclado
| Salvar configuración como dfl
| Salvar configuración como..
| Salir
| Salir del Minicom
+-----+
```

6. Tecleamos **E** para configurar la paridad, la velocidad, etc.

```
+-----+[Parámetros de Comunicación]-----+
| A - Dispositivo |
| B - Localización| SpeeCurrent: 9600 8N1      Data |
| C - Programa de | A: <next>      L: None    S: 5 |
| D - Programa de | B: <prev>       M: Even     T: 6 |
| E - Bps/Paridad/| C: 9600        N: Odd     U: 7 |
| F - Control de F| D: 38400       O: Mark    V: 8 |
| G - Control de F| E: 115200      P: Space   |
|                   |
| ¿Qué configurar Stopbits |
+-----| W: 1           Q: 8-N-1
      | Pantalla| X: 2           R: 7-E-1
      | Salvar c|
      | Salvar c|
      | Salir   | Elija la opción o (Enter) para sal
      | Salir delir: ■
+-----+
```

Solo basta con poner la letra para llegar a la configuración deseada:

-C
-L
-V
-W

7. Una vez configurado deseada pulsamos **Enter** para regresar al Menú Principal de **minicom**

8. Seleccionamos la opción **Salvar Configuración** como **dfl** y salimos de la configuración de **minicom**

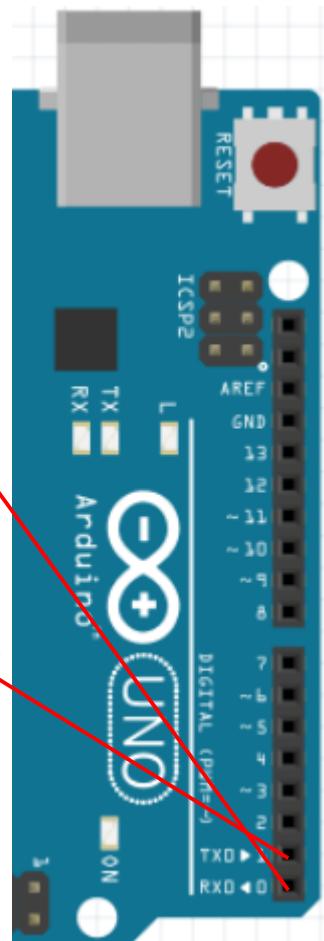
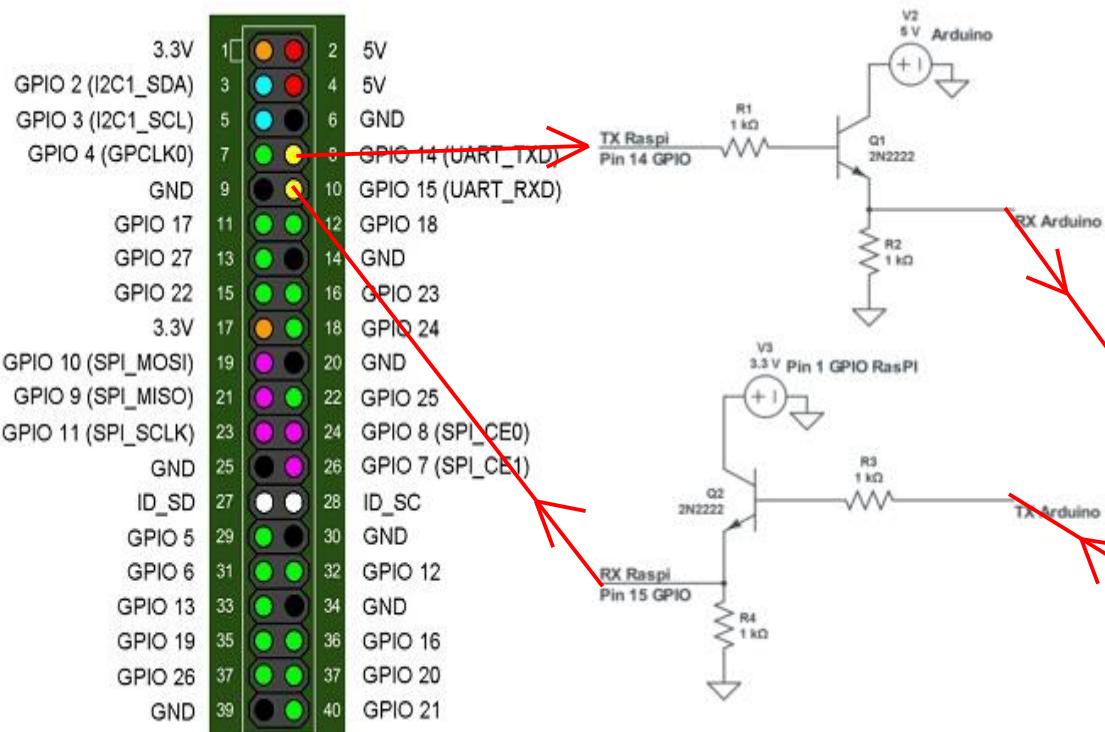
9. Ya configurado accedemos a él tecleando lo siguiente:

\$ minicom

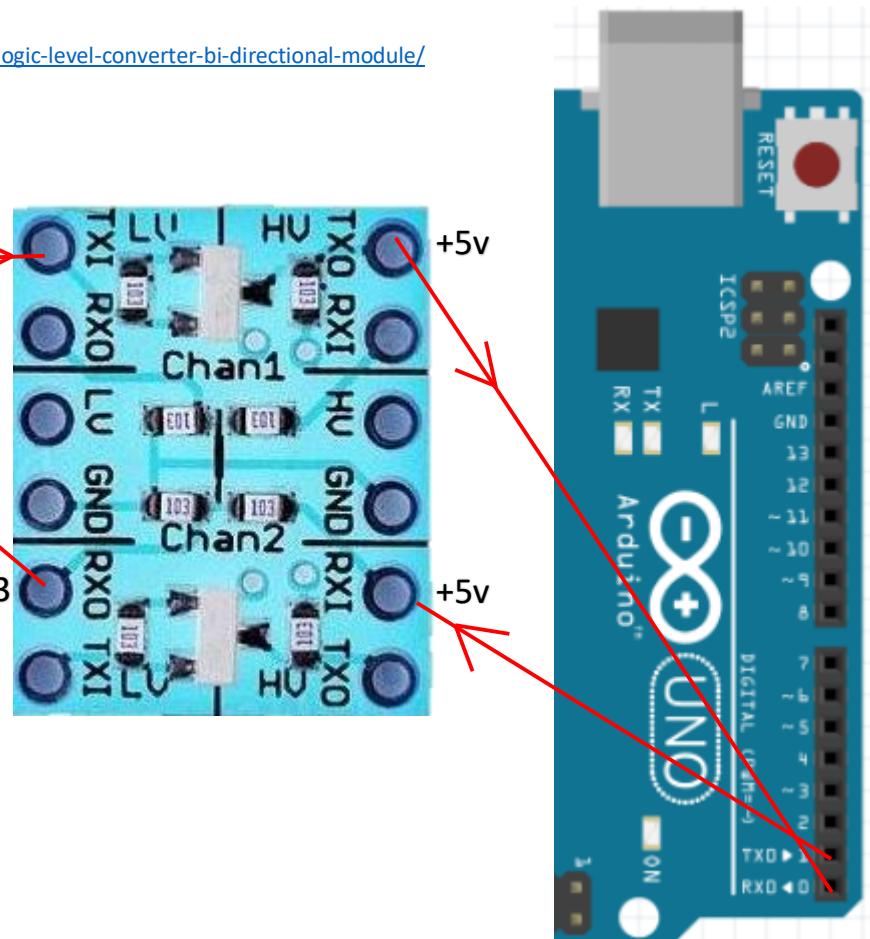
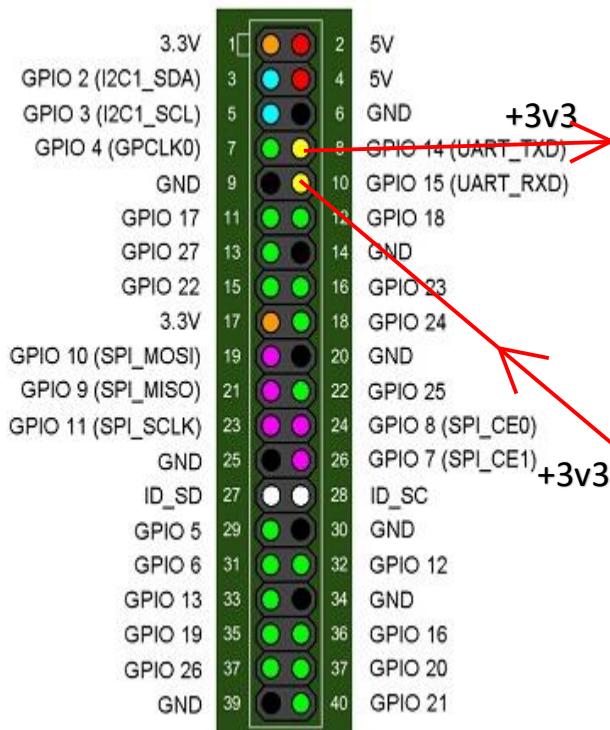
The screenshot shows a terminal window titled "minicom 2.4". The menu bar includes "Archivo", "Editar", "Ver", "Terminal", and "Ayuda". The main area displays the following text:
Welcome to minicom 2.4
OPCIONES: I18n
Compilado en Sep 5 2010, 09:23:03.
Port /dev/ttyUSB0
Presione CTRL-A Z para obtener ayuda sobre teclas especiales
RincondeLinux !■

EJEMPLO DE COMUNICACIÓN UTILIZANDO EL PUERTO SERIAL

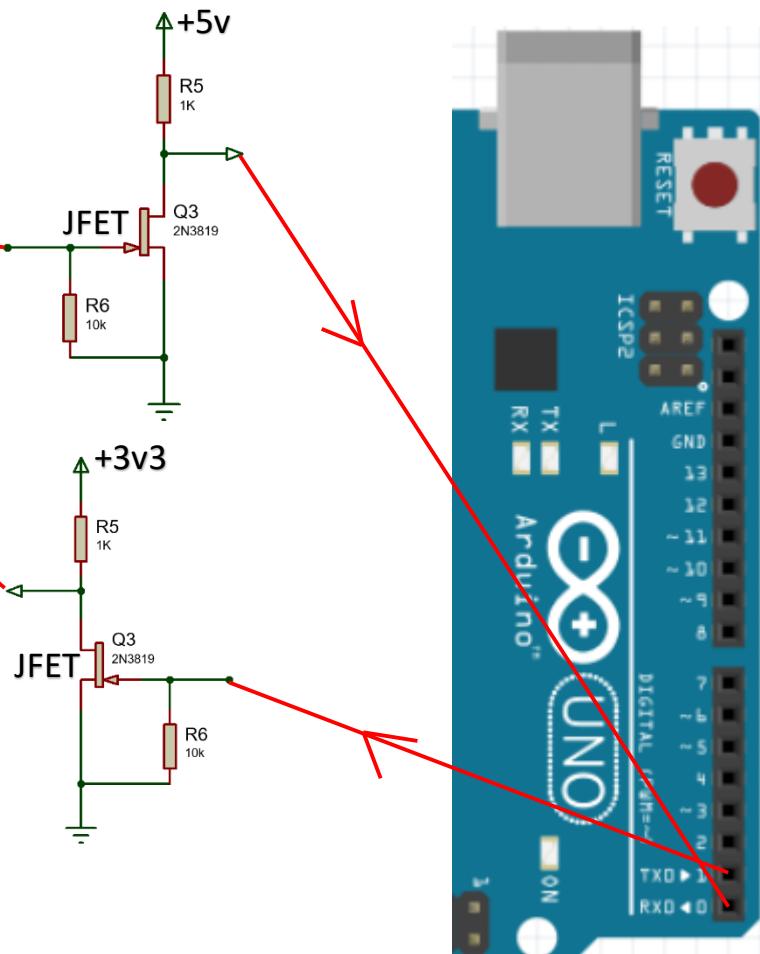
CONFIGURACIÓN HARDWARE (Nota.- INTERCONECTAR LOS GROUNDS)



<https://www.ezgiz.com/product/5pcs-two-channel-iic-i2c-logic-level-converter-bi-directional-module/>



3.3V	1	5V
GPIO 2 (I2C1_SDA)	3	5V
GPIO 3 (I2C1_SCL)	4	GND
GPIO 4 (GPCLK0)	5	GPIO 14 (UART_RXD)
GND	6	GPIO 15 (UART_RXD)
GPIO 17	7	GPIO 18
GPIO 27	9	GND
GPIO 22	10	GPIO 23
3.3V	11	GPIO 24
GPIO 10 (SPI_MOSI)	12	GND
GPIO 9 (SPI_MISO)	13	GPIO 25
GPIO 11 (SPI_SCLK)	14	GPIO 8 (SPI_CE0)
GND	15	GPIO 7 (SPI_CE1)
ID_SD	17	ID_SC
GPIO 5	19	GND
GPIO 6	20	GPIO 12
GPIO 13	21	GND
GPIO 19	22	GPIO 16
GPIO 26	23	GPIO 20
GND	25	GPIO 21
	27	
	29	
	31	
	33	
	35	
	37	
	39	



QUE HACER EN LA TARJETA ARDUINO UNO

- Cargamos el siguiente programa:
- ```
void setup()
{
 Serial.begin(9600);
}
void loop()
{
 Serial.write("Hola Raspi!\n");
 Serial.write("Soy un Arduino UNO\n");
 delay(1000);
}
```

## QUE HACER EN LA TARJETA RASPBERRY PI

- Comprobamos los distintos puertos de los que dispone la RASPI  
**\$ ls /dev/tty\***
- Con el siguiente comando podemos configurar el puerto serie:  
**\$ sudo stty -F /dev/ttymA0 9600**
- Para leer los datos de la consola, en el bash, ejecutamos el siguiente comando:  
**\$ cat /dev/ttymA0**

```
pi@rpi_sandisk_16_casa: ~
File Edit Tabs Help
pi@rpi_sandisk_16_casa: ~ $ cat /dev/ttymA0
Hola Raspi!
Soy Arduino UNO
Hola Raspi!
Soy Arduino UNO
Hola Raspi!
Soy Arduino UNO
```

- Para mandar datos desde la consola, en el bash, ejecutamos el siguiente comando:  
**\$ echo \* > /dev/ttymA0 → siendo \* aquel carácter que deseamos mandar**

**Si tenemos bluetooth activado, utilizaremos la UART → ttym0**

## 7.- INSTALAR EL IDE DE ARDUINO

OBSOLETO

### MÉTODO 1

.- Actualizamos los repositorios:

```
$ sudo apt-get update
```

.- Instalamos el paquete arduino:

```
$ sudo apt-get install arduino
```

Una vez instalado aparecerá una nueva sección, llamada “Electronics”, en el menú de programas, con el acceso directo al IDE de Arduino.

.- Damos permiso al usuario “pi”, para acceder al puerto serie:

```
$ sudo usermod -a -G tty pi
```

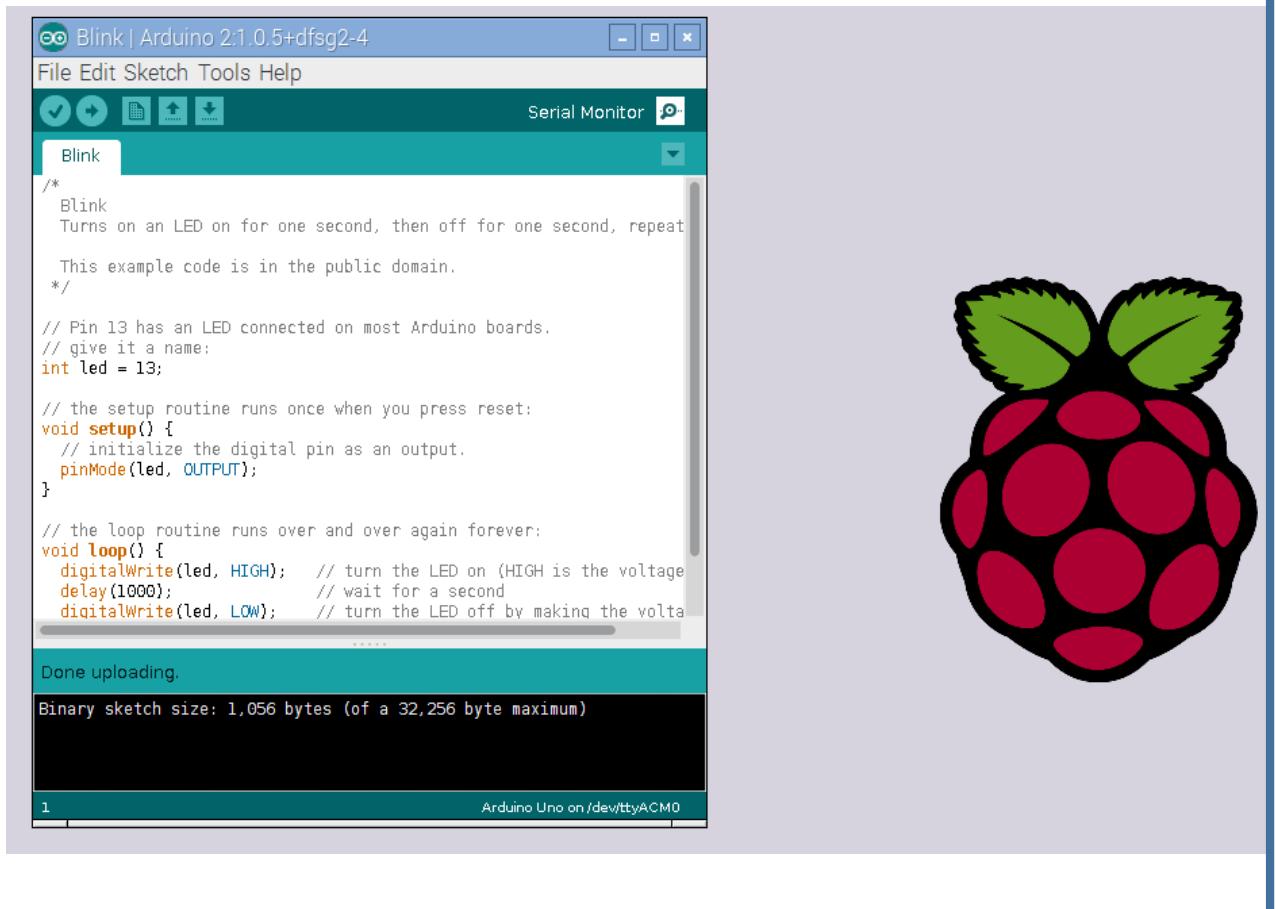
.- Agregamos al usuario “pi” al grupo “dialout”

```
$ sudo usermod -a -G dialout pi
```

.- Conectamos el arduino uno al puerto usb y comprobamos que se añade el puerto serie **/dev/ttyACM0**, con la siguiente orden:

```
$ ls /dev/tty*
```

Abrimos el IDE y tras configurar la tarjeta y el puerto serie, podemos programar y descargar sobre la tarjeta Arduino UNO:

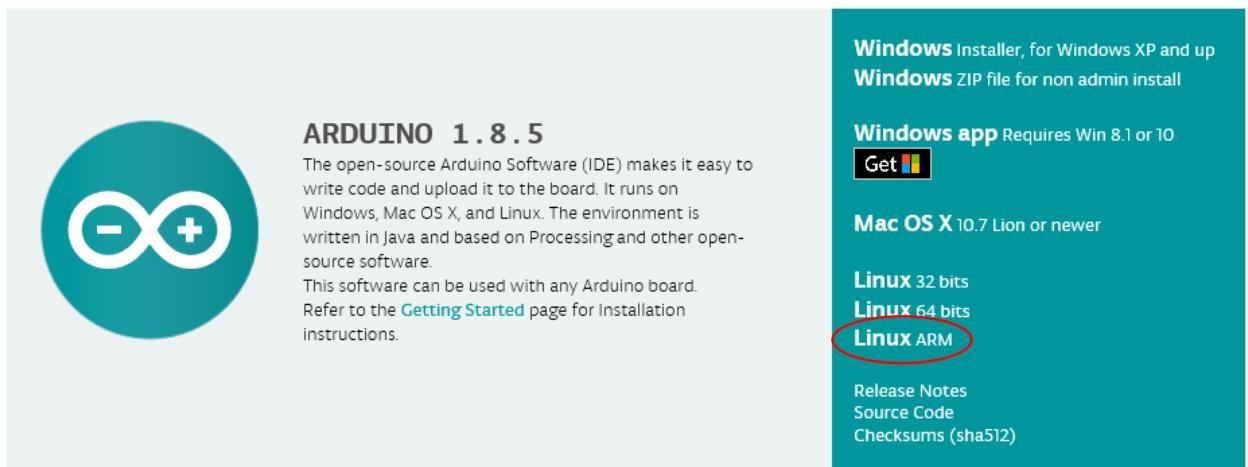


## MÉTODO 2

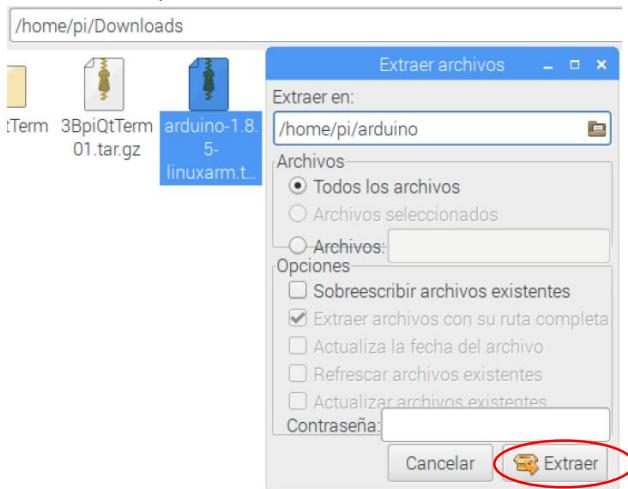
.- Desde la página oficial de Arduino seleccionamos “Linux ARM”.

<https://www.arduino.cc/en/Main/Software>

### Download the Arduino IDE



.- Lo descomprimimos



.- Nos vamos a la carpeta descomprimida y escribimos dos veces el siguiente comando:

**\$sudo ./install.sh**

**\$sudo ./install.sh**

.- En el escritorio nos aparecerá el icono de arduino.



# 8.- PROGRAMACIÓN

## 8.1 SCRIPT

Un script es un archivo de órdenes (archivo de procesamiento por lotes), es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano.

### EJECUCIÓN EN EL BASH (TERMINAL, CONSOLA o INTÉRPRETE DE COMANDOS)

El terminal, o “la consola” es una aplicación de uso muy frecuente cuando se trata de sistemas Linux como es la Raspberry Pi.

El terminal es un **intérprete de órdenes**, basado en texto, que permite manejar la totalidad de un sistema operativo. De hecho, es en sí un intérprete parecido al de Python, y por eso las órdenes que ejecutamos pueden llegar a formar parte de un programa con el que automatizar un montón de procesos, en su propio lenguaje particular.

.- Para crear un script ejecutable en el bash:

1. **Creamos un fichero de texto con la extensión "sh"**

**\$sudo touch nombre\_script.sh**

2. Añadimos una primera línea (**bang line**) con el siguiente contenido y una segunda:

**#!/bin/bash**

**ls**

3. Damos permiso de ejecución: **\$ sudo chmod 777 nombre\_script.sh**

4. Una vez terminado el script, para lanzar el programa desde la consola deberemos escribir:

**\$ ./nombre\_script**

### EJECUCIÓN EN EL INTÉRPRETE DE PYTHON

.- Para crear un script que pueda ser interpretado por Python, hay que añadir la siguiente línea:

**#!/usr/bin/python**

.- Como ejemplo construyamos el siguiente fichero:

**\$ sudo nano primer\_py.py**

Escribimos las siguientes líneas:

```
#!/usr/bin/python
print("hola mundo")
```

.- Damos permiso de ejecución:

**\$ sudo chmod 777 primer\_py.py**

.- Una vez terminado el script, damos permiso de ejecución y lanzamos el programa desde la consola:

**\$ ./primer\_py.py**

The screenshot shows a terminal window with a grey header bar containing the text "pi@rpicovid19: ~". Below the header is a menu bar with options: Archivo, Editar, Pestañas, Ayuda. The main area of the terminal is black and contains white text. It shows the command `pi@rpicovid19:~ $ ./primer_py.py` being entered and then the output "Hola mundo" displayed. At the bottom of the terminal window, there is a small grey input field with a cursor.

## 8.2 PYTHON

### MI PRIMER PROGRAMA

Existen dos formas de ejecutar código Python, bien en una sesión interactiva (línea a línea) con el intérprete, o bien de la forma habitual, escribiendo el código en un archivo de código fuente y ejecutándolo.

Nuestro primer programa es el clásico Hola Mundo. Ejecuta Python y escribe:

- .- (v 2.\*.) → **>>>print "Hola Mundo"** (pulsa Enter)
  - .- (v 3.\*.) → **>>>print ("Hola Mundo")** (pulsa Enter)
- El intérprete responderá imprimiendo el texto **"Hola Mundo"**

Vamos ahora a crear un archivo de texto con el código anterior, de forma que pudiéramos distribuir nuestro pequeño gran programa entre nuestros amigos. Abre tu editor de texto preferido o bien el IDE que hayas elegido y copia la línea anterior.

Guárdalo como **hola.py**, por ejemplo.

Ejecutar este programa es tan sencillo como indicarle el nombre del archivo al intérprete

**python hola.py**

Pero vamos a ver cómo simplificarlo aún más.

#### Si utilizas Windows

Los archivos **.py** ya estarán asociados al intérprete de Python, por lo que basta hacer doble clic sobre el archivo para ejecutar el programa. Sin embargo, como este programa no hace más que imprimir un texto en la consola, la ejecución es demasiado rápida para poder verlo si quiera. Para remediarlo vamos a añadir una nueva línea que espere la entrada de datos por parte del usuario.

**PYTHON 2**  
**print "Hola Mundo"**  
**raw\_input()**

**PYTHON 3**  
**print ("Hola Mundo")**  
**input()**

De esta forma el texto **"Hola Mundo"** se mostrará hasta que pulsemos **Enter**.

#### Si utilizas Linux (u otro Unix)

Para conseguir este comportamiento, es decir, para que el sistema operativo abra el archivo **.py** con el intérprete adecuado, es necesario añadir una nueva línea al principio del archivo:

```
#!/usr/bin/python
print "Hola Mundo"
raw_input() #Esta instrucción es para la V2.* de Python
```

A esta línea se le conoce en el mundo Unix como **shebang**, **hashbang** o **sharpbang**.

El par de caracteres **"#!"** indica al sistema operativo que dicho script se debe ejecutar utilizando el intérprete especificado a continuación. De esto se desprende, evidentemente, que si esta no es la ruta en la que está instalado nuestro intérprete de Python, es necesario cambiarla.

Otra opción es utilizar el programa env (de environment, entorno) para preguntar por la ruta al intérprete de Python, de forma que nuestros usuarios no tengan ningún problema si se diera el caso de que el programa no estuviera instalado en dicha ruta:

```
#!/usr/bin/env python
print "Hola Mundo"
raw_input() #Esta instrucción es para la V2.* de Python
```

Por supuesto además del **shebang**, tendremos que dar permisos de ejecución al programa.

```
$ sudo chmod +x hola.py
```

Y listo, si hacemos doble clic el programa se ejecutará, como en el caso de Windows (en Gnome seleccionaríamos Ejecutar en un Terminal), aunque podríamos ejecutarlo en consola como si fuera un ejecutable cualquiera:

```
$./hola.py
```

**Nota.- Si no utilizamos la línea shebang, deberemos poner en la orden el intérprete que se va a utilizar en la ejecución:**

```
$ python hola.py
```

## LECCIONES DE PROGRAMACIÓN

<http://www.mclibre.org/consultar/python/index.html>



### Introducción a la programación con Python

Estos apuntes del **Curso de iniciación a la programación en Python** se han impartido durante el segundo y tercer trimestre del curso 2015/2016 en el módulo **Lenguaje de Marcas y Sistemas de Gestión de la Información** del ciclo formativo **Administración de Sistemas Informáticos en Red (ASIR)** en el IES Abastos de Valencia (España).

#### Antes de empezar el curso

- [Presentación y licencia](#)
- [Lenguajes de programación](#)
- [Historia de Python](#)
- [Instalación de Python](#)
- [IDLE, el entorno de desarrollo de Python](#)

#### Otros

- [Libro de estilo](#)
- [Documentación](#)
- [Retos de programación](#)
- [Fórmulas matemáticas](#)

#### Lecciones Python

- Hola, mundo
  - [Elementos de un programa](#)
- Tipos de datos
  - [Números y operaciones aritméticas](#)
  - [Cadenas de texto](#)
  - [Variables \(1\)](#)
  - [Booleanos](#)
  - Secuencias: [Tuplas](#), [Rangos](#), [Listas](#)
  - [Variables \(2\)](#) - [Variables \(3\)](#)
- Entrada y salida
  - [Salida por pantalla](#)
  - [Entrada por teclado](#)
- Estructuras de control
  - Sentencias condicionales: [if ... elif ... else ...](#)
  - Iteraciones: [bucle for \(1\)](#) - [bucle for \(2\)](#)
  - Iteraciones: [bucle while](#)
- Funciones
  - [Biblioteca estándar](#)
  - [Funciones \(1\)](#) - [Funciones \(2\)](#)
- Gráficos
  - Tortuga: [turtle \(1\)](#) - [turtle \(2\)](#) - [turtle \(3\)](#)
- Otros
  - [Brython](#)
  - [Valores aleatorios](#)
  - [Restos](#)

#### Ejercicios Python

- [Variables, E/S y Operaciones aritméticas](#)
- [Expresiones lógicas - Expresiones lógicas \(2\)](#)
- [Sentencias condicionales: if ... elif ... else ...](#)
- [Listas de enteros: el tipo range\(\)](#)
- [Iteraciones: bucle for \(1\)](#) - [bucle for \(2\)](#) - [bucle for \(3\)](#)
- [Listas \(1\)](#) - [Listas \(2\)](#)
- [Bucle while \(1\)](#) - [Bucle while \(2\)](#)
- [Funciones \(1\)](#) - [Funciones \(2\)](#)
- [Reparo \(1\)](#) - [Reparo \(2\)](#)

- Gráficos
  - Brython: [SVG \(1\)](#) - [SVG \(2\)](#) - [SVG \(3\)](#) - [SVG \(4\)](#) - [SVG \(5\)](#)
  - Tortuga: [turtle \(1\)](#) - [turtle \(2\)](#) - [turtle \(3\)](#) - [turtle \(4\)](#)

- [Exámenes](#)

## Operadores en Python

- Operadores lógicos: Devuelven un bool.

| Operador | Descripción       | Ejemplo                       |
|----------|-------------------|-------------------------------|
| and      | ¿se cumple a y b? | r=True and False # r es False |
| or       | ¿se cumple a o b? | r=True or False # r es True   |
| not      | No a              | r=not True #r es False        |

| AND   |       |        |
|-------|-------|--------|
| A     | B     | Salida |
| False | False | False  |
| False | True  | False  |
| True  | False | False  |
| True  | True  | True   |

| OR    |       |        |
|-------|-------|--------|
| A     | B     | Salida |
| False | False | False  |
| False | True  | True   |
| True  | False | True   |
| True  | True  | True   |

| NOT   |        |
|-------|--------|
| A     | Salida |
| False | True   |
| True  | False  |

- Operadores relacionales: comparan dos expresiones y devuelven un bool.

| Operador | Descripción                | Ejemplo             |
|----------|----------------------------|---------------------|
| ==       | ¿Son iguales a y b?        | r=5==3 # r es False |
| !=       | ¿Son distintos a y b?      | r=5!=3 # r es True  |
| <        | ¿Es a menor que b?         | r=5<3 # r es False  |
| >        | ¿Es a mayor que b?         | r=5>3 # r es True   |
| <=       | ¿Es a menor o igual que b? | r=5<=5 # r es True  |
| >=       | ¿Es a mayor o igual que b? | r=5>=3 # r es True  |

■ Operadores aritméticos: de números reales y enteros.

| Operador | Descripción     | Ejemplo             |
|----------|-----------------|---------------------|
| +        | Suma            | r=3+2 # r es 5      |
| -        | Resta           | r=4-7 # r es -3     |
| -        | Negación        | r=-7 # r es -7      |
| *        | Multiplicación  | r=2*6 # r es 12     |
| **       | Exponente       | r=2**6 # r es 64    |
| /        | División        | r=3.5/2 # r es 1.75 |
| //       | División Entera | r=3.5//2 r es 1.0   |
| %        | Módulo          | r=7%2 # r es 1      |

**Nota.** Algunos operadores sirven para tipo **str** también, por ejemplo 'Py' + 'thon', se corresponde con el string 'Python'. En este caso decimos que el operador + está sobrecargado.

■ Operadores a nivel de Bit

| Operador | Descripción                   | Ejemplo          |
|----------|-------------------------------|------------------|
| &        | and                           | r=3 & 2 # r es 2 |
|          | or                            | r=3 2 # r es 3   |
| ^        | xor                           | r=3^2 # r es 1   |
| ~        | not                           | r=~3 # r es -4   |
| <<       | Desplazamiento a la izquierda | r=3<<1 # r es 6  |
| >>       | Desplazamiento a la derecha   | r=3>>1 # r es 1  |

## SENTENCIA SWITCH

<https://www.clubdetecnologia.net/blog/2017/python-como-se-implementa-una-sentencia-switch-case/>

Aunque los lenguajes populares como Java y PHP tienen una declaración switch ya incorporada, puede que se sorprenda al saber que el lenguaje Python no tiene uno. Como tal, puede estar tentado a utilizar una serie de bloques if-else-if, usando una condición if para cada alternativa.

Sin embargo, debido a la tabla de salto, una instrucción switch es mucho más rápido que una escalera if-else-if. En lugar de evaluar cada condición en forma secuencial, solo tiene que buscar la variable/expresión una vez y saltar directamente a la rama apropiada de código para ejecutarla.

## COMO SE IMPLEMENTA LA SENTENCIA SWITCH EN PYTHON

La forma de Python de implementar la instrucción **switch** es utilizar las asignaciones de diccionarios, también conocidas como matrices asociativas, que proporcionan asignaciones simples de clave-valor de uno a uno.

Aquí esta la implementación de Python de la declaración anterior. En el siguiente ejemplo, creamos un diccionario llamado **switcher** para almacenar todos los casos del tipo **switch**.

```
1 def switch_demo(argument):
2 switcher = {
3 1: "January",
4 2: "February",
5 3: "March",
6 4: "April",
7 5: "May",
8 6: "June",
9 7: "July",
10 8: "August",
11 9: "September",
12 10: "October",
13 11: "November",
14 12: "December"
15 }
16 print (switcher.get(argument, "Invalid month"))
17 # Utilizando "lambda"
18 #luis = (switcher.get(argument, lambda: "opción inválida"))
19 #print(luis())
20 switch_demo(14)
21
```

En el ejemplo, al pasar un argumento en la función **switch\_demo**, se busca el valor en el diccionario. Si se encuentra una coincidencia, se imprime el valor asociado, de lo contrario se imprime una cadena predeterminada («Invalid Month»). La cadena predeterminada ayuda a implementar la opción «default» de la sentencia switch.

## ASIGNACIÓN DE DICCIONARIO PARA FUNCIONES

Aquí es donde se vuelve más interesante. Los valores de un diccionario de Python pueden ser de cualquier tipo de datos. De modo que no tiene que limitarse a utilizar constantes (enteros, cadenas), también puede utilizar nombres de funciones y lambdas como valores.

### Diferencias entre las funciones `lambda` y las definidas con `def`

En Python las funciones creadas mediante la palabra `lambda` también se pueden crear utilizando `def`

Aunque aparentemente se obtenga el mismo resultado existen ciertas diferencias entre ambos métodos que es necesario tener en cuenta.

- Al utilizar la palabra clave `lambda` se crea un objeto función sin crearse al mismo tiempo un nombre en el espacio de nombres. Nombre, que si se crea al definir la función con `def`.
- Las funciones `lambda` se crea en una única línea de código, por lo que son adecuadas cuando se desea minimizar el número de estas.
- Para los usuarios noveles de Python las funciones `lambda` son generalmente menos legibles que las tradicionales.
- En el caso de que se deseen una función `lambda` es necesario asignarla a una variable. Ya que, si no es así, al carecer de identificador, solamente se podrá utilizar en la línea donde se defina.

Por ejemplo, también puede implementar la instrucción `switch` anterior creando un diccionario de nombres de funciones como valores. En este caso, `switcher` es un diccionario de nombres de funciones y no de cadenas.

```
1 def one():
2 return "January"
3 def two():
4 return "February"
5 def three():
6 return "March"
7 def four():
8 return "April"
9 def five():
10 return "May"
11 def six():
12 return "June"
13 def seven():
14 return "July"
15 def eight():
16 return "August"
17 def nine():
18 return "September"
19 def ten():
20 return "October"
21 def eleven():
22 return "November"
23 def twelve():
24 return "December"
25
26 def numbers_to_months(argument):
27 switcher = {
28 1: one,
29 2: two,
30 3: three,
31 4: four,
32 5: five,
33 6: six,
34 7: seven,
35 8: eight,
36 9: nine,
37 10: ten,
38 11: eleven,
39 12: twelve
40 }
41 # Get the function from switcher dictionary
42 func = switcher.get(argument, lambda: "Invalid month")
43 # Execute the function
44 print(func())
45
46 numbers_to_months(15)
47 print(eleven())
```

Aunque las funciones anteriores son bastante simples y solo devuelven cadenas, puede usar este enfoque para ejecutar bloques elaborados de código dentro de cada función.

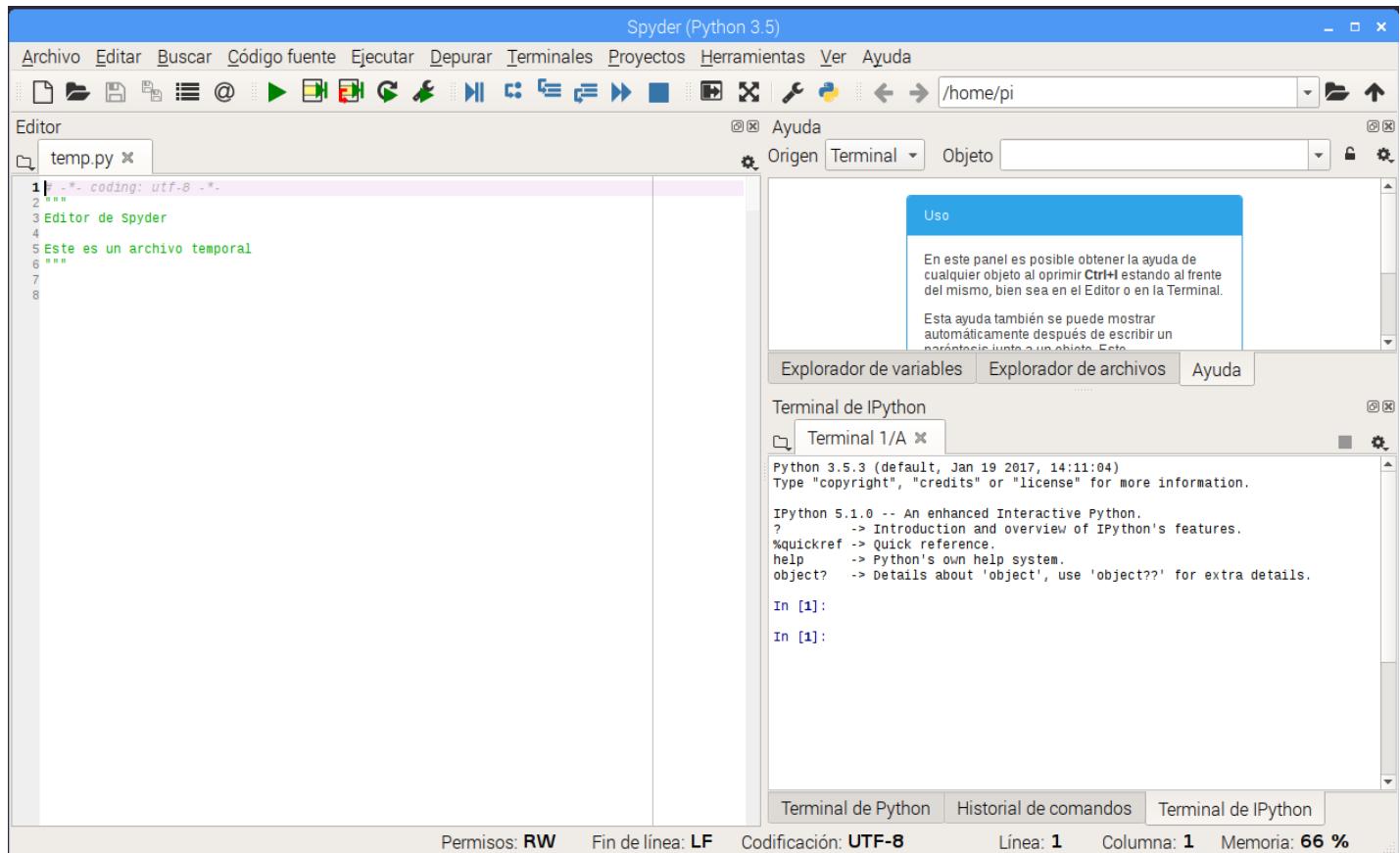
## ENTORNO DE PROGRAMACIÓN SPYDER

Para instalar el IDE:

```
$ sudo apt-get install spyder3
```

```
$ sudo pip install rope
```

```
$ sudo apt-get install spyder
```



## BREVE NOCIÓN SOBRE CLASES

Python es un lenguaje de programación orientado a objetos. Pero, a diferencia de otros lenguajes de este tipo, Python no obliga a utilizar esta "orientación a objetos".

```
#!/usr/bin/python
-*- coding: utf-8 -*-

class Persona:
 nBrazos=0
 nPiernas=0
 cabello=True
 cCabello="Defecto"
 hambre=0 #hambre de 0-10

 def __init__(self):
 self.nBrazos=2
 self.nPiernas=2

 def dormir(self):
 pass
 def comer(self):
 self.hambre=6

class Hombre(Persona):
 nombre="Defecto"
 sexo="M"

 def cambiarNombre(self,nombre):
 self.nombre=nombre

class Mujer(Persona):
 nombre="Defecto"
 sexo="F"

jose=Hombre() # Definimos el objeto jose=Hombre()
jose.comer() # Llamamos al método comer
print (jose.hambre) # Comprobamos si el parámetro hambre se cambió o no
print (jose.nBrazos)
print (jose.sexo)

marian=Mujer() # Definimos el objeto marian=Mujer()
marian.comer() # Llamamos al método comer
print (marian.hambre) # Comprobamos si el parámetro hambre se cambió o no
print (marian.nBrazos)
print (marian.sexo)

jose.cambiarNombre("luis")
print (jose.nombre)
```

## LECTURA Y ESCRITURA DE FICHEROS EN PYTHON

Tanto para leer como para escribir un fichero lo primero que hay que hacer es abrir el fichero con la función **open()** que usamos con dos argumentos: **open(filename, mode)**. Una vez hayamos terminado de trabajar con el fichero debemos cerrarlo usando la función **f.close()**.

### LECTURA DE FICHEROS

- Python nos proporciona diferentes maneras de leer un fichero. En primer lugar podemos leer un fichero completamente usando la función **f.read()**:

```
En primer lugar debemos de abrir el fichero que vamos a leer.
Usa 'rb' en vez de 'r' si se trata de un fichero binario.
infile = open('texto.txt', 'r')
Mostramos por pantalla lo que leemos desde el fichero
print('>>> Lectura completa del fichero')
print(infile.read())
Cerramos el fichero.
infile.close()
```

- También podemos optar por leer una cantidad determinadas de **bytes** del fichero usando la función **f.read(size)**:

```
En primer lugar debemos de abrir el fichero que vamos a leer.
Usa 'rb' en vez de 'r' si se trata de un fichero binario.
infile = open('texto.txt', 'r')
Mostramos por pantalla lo que leemos desde el fichero
print('>>> Lectura de una cantidad determinada de bytes')
print(infile.read(50) + '\n')
Cerramos el fichero.
infile.close()
```

- Podemos optar por leer una única línea del fichero con la función **f.readline()**:

- **1º opción**

```
En primer lugar debemos de abrir el fichero que vamos a leer.
Usa 'rb' en vez de 'r' si se trata de un fichero binario.
infile = open('texto.txt', 'r')
Mostramos por pantalla lo que leemos desde el fichero
print('>>> Lectura de una línea del fichero')
print(infile.readline())
Cerramos el fichero.
infile.close()
```

- **2º opción**

```
datos = 'texto.txt'
Cargamos la el nº de la linea anterior a la que queremos leer
La primera linea es la 0
ultima = 3
f = open(datos, 'r')
Recorremos las lineas hasta posicionar el cursor en
la linea anterior a la que queremos leer
for n, _ in enumerate(f):
 print(n)
 if n == ultima:
 break
 print(f.readline())
f.close()
#Guardamos la última linea leida para otra vez
```

- Por último, podemos **leer un fichero completo línea a línea** de la siguiente manera:

```
En primer lugar debemos de abrir el fichero que vamos a leer.
Usa 'rb' en vez de 'r' si se trata de un fichero binario.
infile = open('texto.txt', 'r')
Mostramos por pantalla lo que leemos desde el fichero
print('>>> Lectura del fichero línea a línea')
for line in infile:
 print(line)
Cerramos el fichero.
infile.close()
```

## ESCRITURA DEL FICHERO

- Para escribir un fichero en Python tendremos básicamente dos opciones que vamos a ver a continuación. Primero podemos **escribir un fichero sobreescribiendo el contenido** del fichero:

```
outfile = open('texto.txt', 'w') # Indicamos el valor 'w'.
outfile.write('Fusce vitae leo purus, a tempor nisi.\n') #sobreescribe todo su contenido
outfile.close()
Leemos el contenido para comprobar que ha sobreescrito el contenido.
infile = open('texto.txt', 'r')
print('>>> Escritura de fichero sobreescribiendo su contenido.')
print(infile.read())
Cerramos el fichero.
infile.close()
```

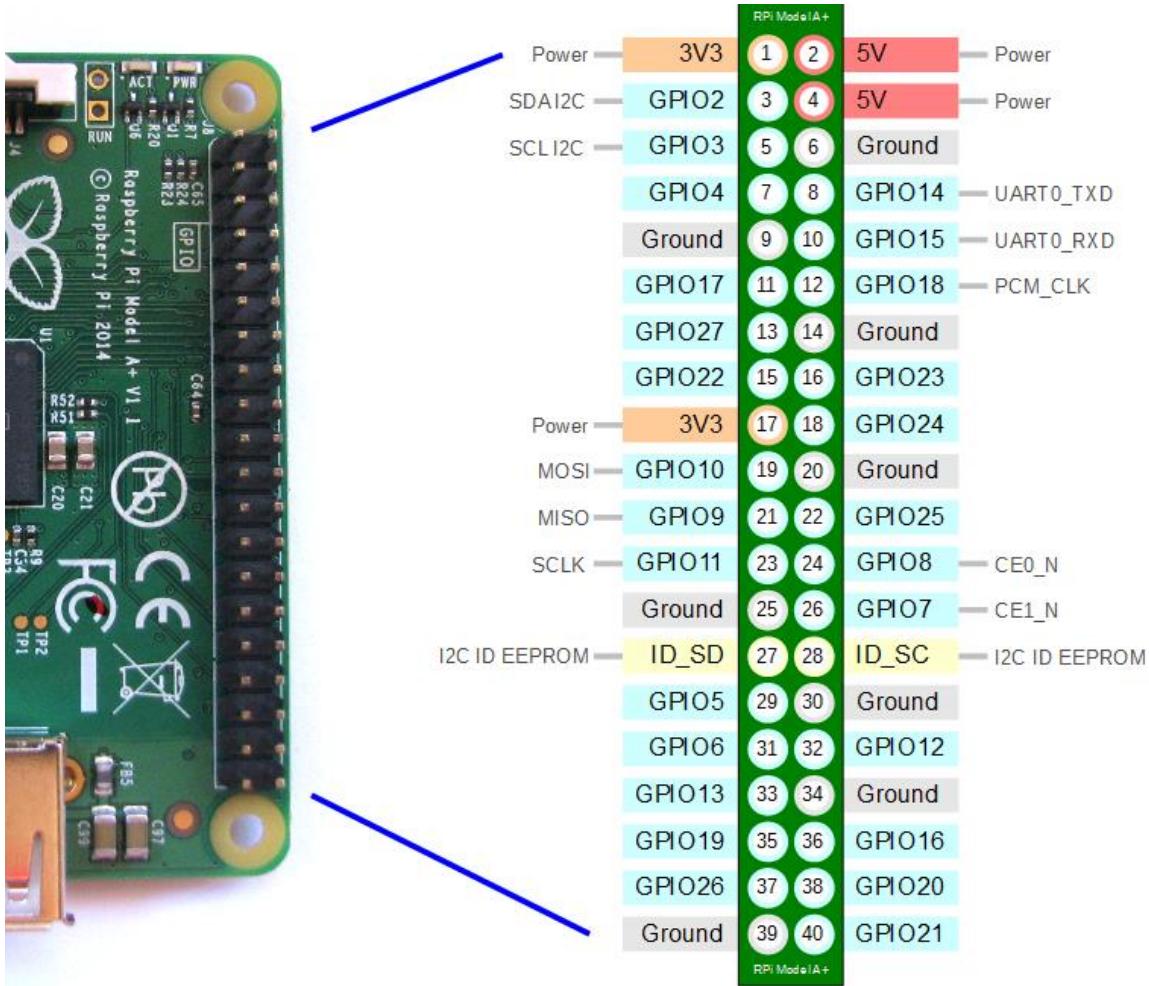
- podemos **concatenar el nuevo contenido al contenido ya existente** en el fichero:

```
outfile = open('texto.txt', 'a') # Indicamos el valor 'a'.
outfile.write('Fusce vitae leo purus, a tempor nisi.\n')
outfile.close()
Leemos el contenido para comprobar que ha concatenado el contenido.
infile = open('texto.txt', 'r')
print('>>> Escritura de fichero concatenando su contenido.')
print(infile.read())
Cerramos el fichero.
infile.close()
```

## 8.3 GPIO: ENTRADAS Y SALIDAS

**¿Qué es GPIO?...** General Purpose Input Output (GPIO) es un sistema de entrada y salida de propósito general. Como su propio nombre indica, son pines que se pueden configurar para realizar distintas funciones, de ahí que sean de propósito general y no para un uso específico.

Será el usuario en tiempo de ejecución el que pueda **configurar estos pines GPIO** para que hagan lo que él quiere. Se puede hacer de diferentes formas, como con ciertos códigos o scripts **desde la consola** o con el **programa Python**.



Como podemos ver, contamos con dos pines de alimentación de 5V, otros tantos de 3.3V (limitados a 50mA) y cinco de tierra (GND).

Por otra parte (en azul) tenemos 17 pines digitales cuyo estado puede ser HIGH (un voltaje entre 2v y 3.3v) o LOW (entre 0v y 0.8v). Los voltajes entre 0.8 y 2 darán indeterminación.

- Los niveles de voltaje GPIO son de 3,3 V y no tolera 5 V. No hay protección de sobre-voltaje en la placa.
- Hay que tener presente que la corriente que sale de esos pines proviene de la fuente de alimentación de 3.3V y esta fuente está diseñada para una carga pico de unos 3 mA por cada pin de GPIO. Es decir, aunque el SoC de Broadcom permita drenar hasta 16 mA por cada pin la fuente no podrá dar mas de unos 78 mA en total (51 mA en los modelos originales). No supone riesgo alguno si intentas superar este límite, pero no funcionará.

Ejecuta:

```
$ pinout
```

The terminal window shows the output of the `pinout` command. It displays a schematic diagram of the Raspberry Pi's pins and their connections, along with system information and pin details for the J8 header.

**System Information:**

```
pi@rpi_64_opencv:~ $ pinout
[...]
Pi Model ???V1.3
[...]
Revision : a020d3
SoC : BCM2837
RAM : 1024Mb
Storage : MicroSD
USB ports : 4 (excluding power)
Ethernet ports: 1
Wi-fi : False
Bluetooth : False
Camera ports (CSI): 1
Display ports (DSI): 1
```

**J8 Header Details:**

| Pin    | Function | Pin  | Function |
|--------|----------|------|----------|
| 3V3    | (1)      | (2)  | 5V       |
| GPIO2  | (3)      | (4)  | 5V       |
| GPIO3  | (5)      | (6)  | GND      |
| GPIO4  | (7)      | (8)  | GPIO14   |
| GND    | (9)      | (10) | GPIO15   |
| GPIO17 | (11)     | (12) | GPIO18   |
| GPIO27 | (13)     | (14) | GND      |
| GPIO22 | (15)     | (16) | GPIO23   |
| 3V3    | (17)     | (18) | GPIO24   |
| GPIO10 | (19)     | (20) | GND      |
| GPIO9  | (21)     | (22) | GPIO25   |
| GPIO11 | (23)     | (24) | GPIO8    |
| GND    | (25)     | (26) | GPIO7    |
| GPIO8  | (27)     | (28) | GPIO1    |
| GPIO5  | (29)     | (30) | GND      |
| GPIO6  | (31)     | (32) | GPIO12   |
| GPIO13 | (33)     | (34) | GND      |
| GPIO19 | (35)     | (36) | GPIO16   |
| GPIO26 | (37)     | (38) | GPIO20   |
| GND    | (39)     | (40) | GPIO21   |

For further information, please refer to <https://pinout.xyz/>

```
pi@rpi_64_opencv:~ $
```

## Características de la Interfaz GPIO

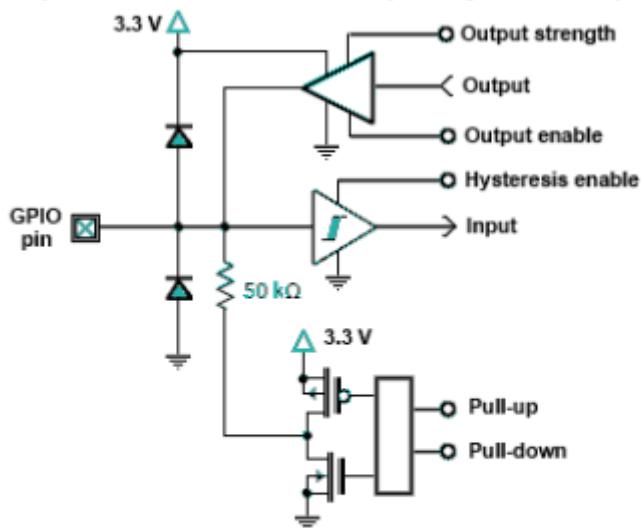
Desde el punto de vista del programador los pines de GPIO de la Raspberry Pi se ven como dispositivos mapeados en memoria. Es decir, para configurar los pines, sacar valores digitales o leer señales digitales tenemos que leer o escribir en posiciones de memoria determinadas. Sin embargo, el procesador de la Raspberry Pi utiliza un mecanismo denominado **memoria virtual**, en el que cada proceso ve un espacio de direcciones diferente, que no necesariamente tiene que corresponder con el espacio físico y que garantiza el aislamiento entre procesos.

En GNU/Linux para poder acceder a las direcciones físicas determinadas es necesario emplear un dispositivo **/dev/mem** que a todos los efectos se comporta como un archivo normal. Por ejemplo, en la posición 0x20200034 del dispositivo **/dev/mem** puede leerse el valor de las entradas **GPIO0** a **GPIO31**.

Obviamente, acceder a todo el espacio físico de direcciones es muy peligroso, puesto que permite que desde un proceso se pueda acceder a todo el espacio de direcciones de los demás procesos. No solo se compromete el aislamiento entre procesos sino también la seguridad del sistema. Un proceso malicioso podría utilizar funciones privilegiadas y, un usuario malicioso podría incluso dañar físicamente la Raspberry Pi. Por este motivo el dispositivo **/dev/mem** tiene **permisos de escritura** solamente para el **superusuario**.

Las versiones recientes de Raspbian tienen un dispositivo **/dev/gpiomem** que permite acceder solo al rango de direcciones de los pines de GPIO y tiene permiso de escritura para el **grupo gpio**. El usuario **pi** es del **grupo gpio**. Por tanto, los programas del usuario **pi** pueden actuar sobre los pines de **GPIO**. En la práctica, esto puede no ser así porque muchas bibliotecas no utilizan aún **/dev/gpiomem**.

**Equivalent Circuit for Raspberry Pi GPIO pins**



*Circuito equivalente de un pin de GPIO (Fuente: [Mosaic Industries](#)).*

Los pines de GPIO de la Raspberry Pi incorporan un conjunto de características muy interantes:

- Tienen capacidad de **limitar el slew rate**. Esto permitiría mejorar la inmunidad al ruido y reducir el ruido de crosstalk, pero a costa de alargar los tiempos de propagación.
- Se puede **programar su drive strength**, es decir, su capacidad de entregar corriente, entre 2 mA y 16 mA en saltos de 2 mA (ocho posibles valores). Básicamente consiste en la posibilidad de activar más o menos drivers en paralelo.

Normalmente **en el arranque está configurado a 8 mA**. Esto no significa que no podamos pedir más corriente. Hasta 16 mA es seguro. Sin embargo si superamos los 8 mA la tensión bajará hasta el punto de que un uno lógico pueda dejar de interpretarse como uno y la disipación de calor será mayor.

Por otro lado si programamos los pines a su máxima capacidad tendremos picos de corriente que afectan al consumo y pueden llegar a afectar al funcionamiento de la tarjeta microSD, especialmente con cargas capacitivas. Este efecto se nota más cuanto mayor número de salidas comuten simultáneamente.

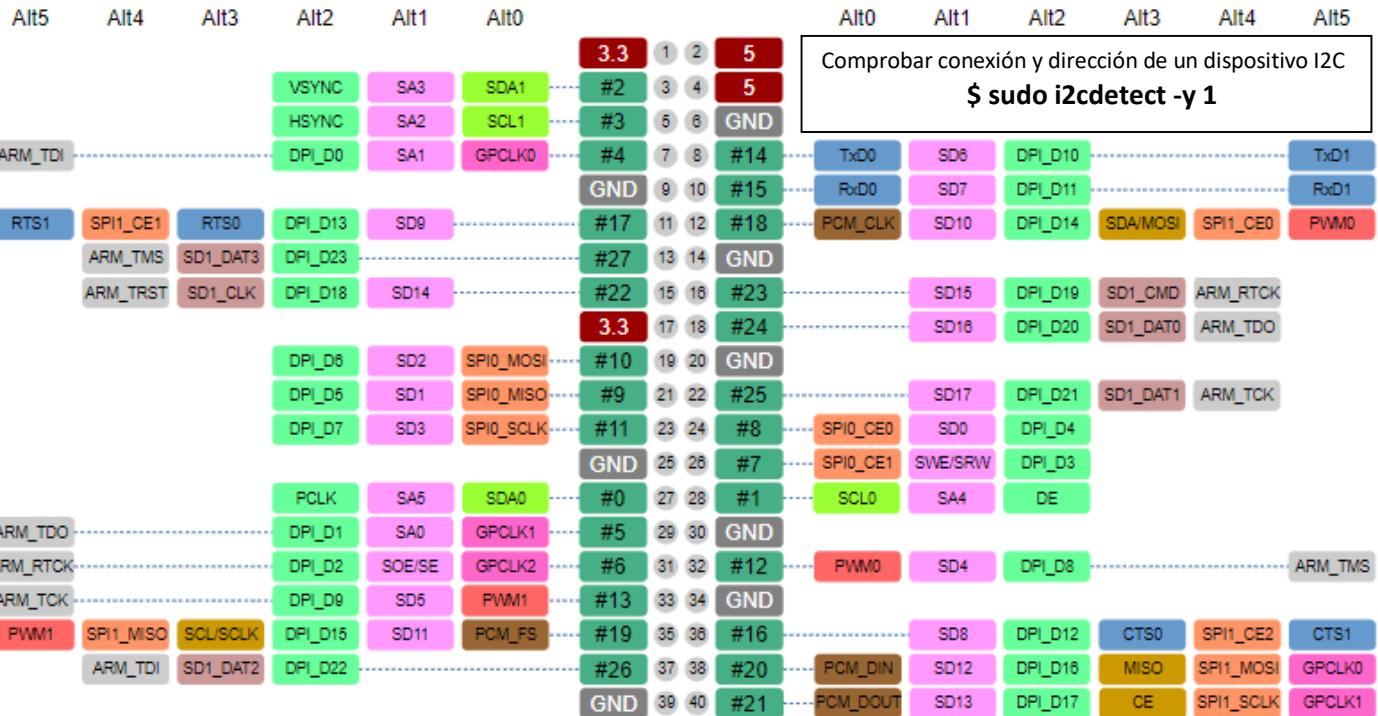
- Es posible configurar las entradas con o sin **Schmitt trigger** de manera que la transición a nivel bajo y a nivel alto tengan umbrales diferentes. Esto permite dotar de cierta tolerancia a ruido.
- Es posible habilitar una resistencia de **pullup y/o pulldown**. Su valor es en torno a los 50KOhm.
- La configuración de limitación de **slew rate**, **drive strength** y entrada con Schmitt trigger no se realiza pin a pin sino en bloques (**GPIO0-27, GPIO28-45, GPIO46-53**)

### Address

0x 7e10 002c PADS (GPIO 0-27)  
 0x 7e10 0030 PADS (GPIO 28-45)  
 0x 7e10 0034 PADS (GPIO 46-53)

| Bit(s) | Field Name | Description                                                                                                       | Type | Reset |
|--------|------------|-------------------------------------------------------------------------------------------------------------------|------|-------|
| 31:24  | PASSWRD    | Must be 5A when writing: Accidental write protect password                                                        | W    | 0     |
| 23:5   |            | <b>Reserved - Write as 0, read as don't care</b>                                                                  |      |       |
| 4      | SLEW       | <u>Slew rate</u><br>0 = slew rate limited<br>1 = slew rate not limited                                            | RW   | 0x1   |
| 3      | HYST       | <u>Enable input hysteresis</u><br>0 = disabled<br>1 = enabled                                                     | RW   | 0x1   |
| 2:0    | DRIVE      | <u>Drive Strength</u><br>0 = 2mA<br>1 = 4mA<br>2 = 6mA<br>3 = 8mA<br>4 = 10mA<br>5 = 12mA<br>6 = 14mA<br>7 = 16mA | RW   | 0x3   |

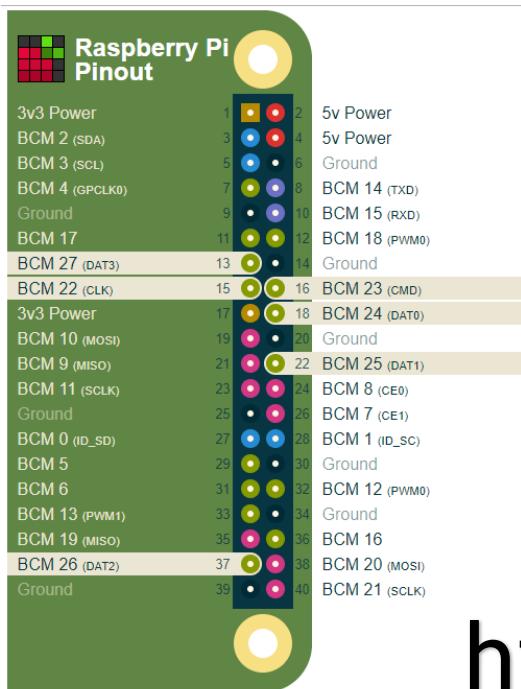
Todos los pines de GPIO tienen la posibilidad de ser usados con otras funciones alternativas. Cada pin de GPIO puede configurarse como entrada, salida o como una de las seis funciones alternativas (desde Alt0 hasta Alt5). No todas las configuraciones tienen sentido.



|     |      |     |      |     |     |     |             |     |                 |      |         |     |      |
|-----|------|-----|------|-----|-----|-----|-------------|-----|-----------------|------|---------|-----|------|
| 3.3 | Vcc  | #21 | GPIO | CE0 | SPI | PWM | Salida PWM  | CE  | Esclavo I2C/SPI | DIN  | PCM     | DE  | LCD  |
| GND | Masa | RxD | UART | SCL | I2C | CMD | Interfaz SD | DIN | Mem. secundaria | CLK1 | Relojes | TDI | JTAG |

- Hay que tener presente que algunos elementos no pueden seleccionarse con seguridad porque ya se han utilizado en otras partes de la Raspberry Pi.  
Por ejemplo, el BCM2837 tiene **dos interfaces SD** y **dos UART**. Sin embargo, **ambas interfaces SD** están ocupadas, una para la **tarjeta microSD** y otra para la comunicación **WiFi**. Análogamente, la **UART0** se destina a la interfaz **Bluetooth** y solo se expone **UART1** en los **pines 8 y 10**.  
Otro ejemplo son los relojes de propósito general (**GPCLKx**) que permiten generar relojes de frecuencia programable en determinadas patas. **GPCLK1** está reservado para uso interno (**Ethernet**) y si se intenta usar lo más probable es que se cuelgue la Raspberry Pi. Nada grave, pero tampoco es agradable.
  - Para el uso del **I2C**, los pines 5 y 3 (SCL0 y SDA0) deberán de tener una **pullup de 1K8 Ohm**.
  - Cada pin GPIO puede generar una **interrupción** por cambio de nivel “**raising/falling**”.
  - Habrá que poner atención a los cambios simultáneos en las salidas (SSO). Para evitar interferencias, las corrientes de salida deben mantenerse lo más bajas posible.
  - El uso de la **UART**, de fábrica viene configurado como un puerto de "consola" para monitorear el funcionamiento del RasPI a 115200 bits/s.

Los pines **ID\_SD / ID\_SC** están diseñados para ser utilizados en la comunicación con un chip de memoria en serie especial (EEPROM), que puede incluirse en las tarjetas de interfaz que conforma el Hardware At Top (**HAT**) estándar y permite que la Raspberry Pi identifique la tarjeta.



**SDIO - SD Card Interface**

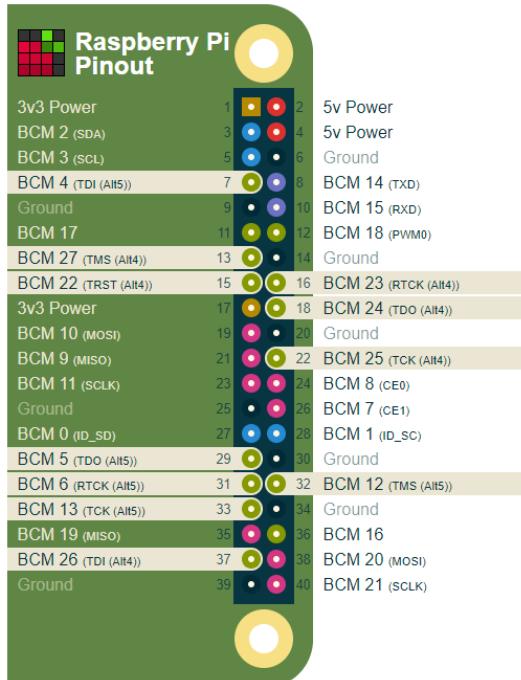
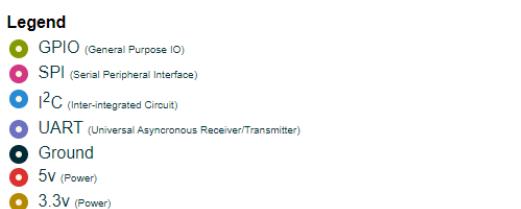
SDIO is the SD host/eMMC interface on the Raspberry Pi. SD host signals are normally used for the microSD slot.

These pins are "SD host" on Alt0 and "eMMC" on Alt3.

**Details**

- Uses 6 GPIO pins

<https://pinout.xyz>



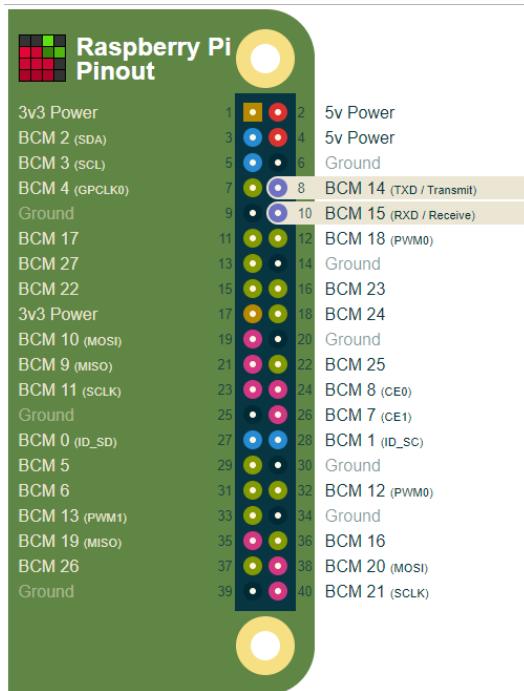
**JTAG - Joint Test Action Group**

JTAG is a standardised interface for debugging integrated circuits which you can use to debug your Raspberry Pi.

**Details**

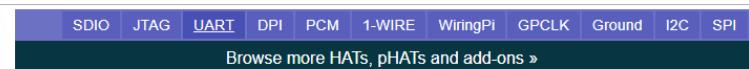
- Uses 11 GPIO pins





**Legend**

- GPIO (General Purpose IO)
- SPI (Serial Peripheral Interface)
- I<sup>2</sup>C (Inter-Integrated Circuit)
- UART (Universal Asynchronous Receiver/Transmitter)
- Ground
- 5V (Power)
- 3.3V (Power)



## UART - Universal Asynchronous Receiver/Transmitter

UART pins in BCM mode are: 14, 15

UART pins in WiringPi are: 15, 16

UART is an asynchronous serial communication protocol, meaning that it takes bytes of data and transmits the individual bits in a sequential fashion.

Asynchronous transmission allows data to be transmitted without the sender having to send a clock signal to the receiver. Instead, the sender and receiver agree on timing parameters in advance and special bits called 'start bits' are added to each word and used to synchronize the sending and receiving units.

UART is commonly used on the Pi as a convenient way to control it over the GPIO, or access the kernel boot messages from the serial console (enabled by default).

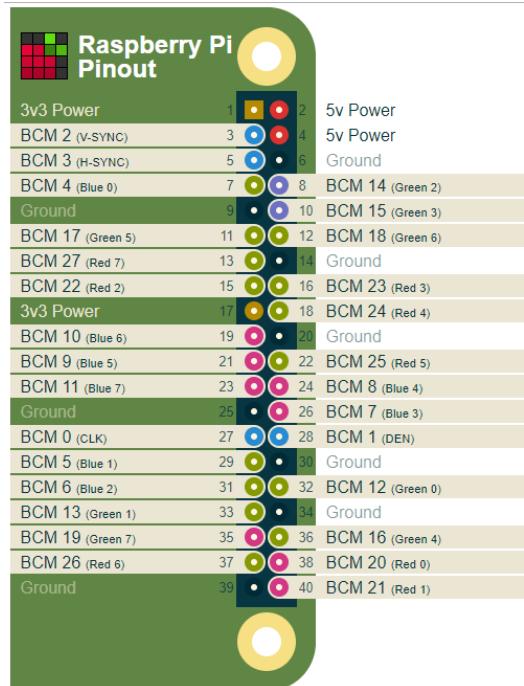
It can also be used as a way to interface an Arduino, bootloaded ATmega, ESP8266, etc with your Pi. Be careful with logic-levels between the devices though, for example the Pi is 3.3v and the Arduino is 5v. Connect the two and you might conjure up some magic blue smoke.

Assuming you have WiringPi-Python installed, the following python example opens the Pi's UART at 9600baud and puts 'hello world'

```
1. import wiringpi
2. wiringpi.wiringPiSetup()
3. serial = wiringpi.serialOpen('/dev/ttyAMA0', 9600)
4. wiringpi.serialPuts(serial, 'hello world!')
```

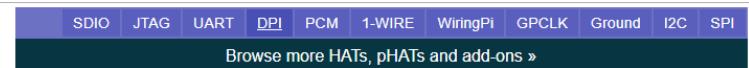
### Details

- 2 pin header
- Uses 2 GPIO pins
- [More Information](#)



**Legend**

- GPIO (General Purpose IO)
- SPI (Serial Peripheral Interface)
- I<sup>2</sup>C (Inter-Integrated Circuit)
- UART (Universal Asynchronous Receiver/Transmitter)
- Ground
- 5V (Power)
- 3.3V (Power)



## DPI - Display Parallel Interface

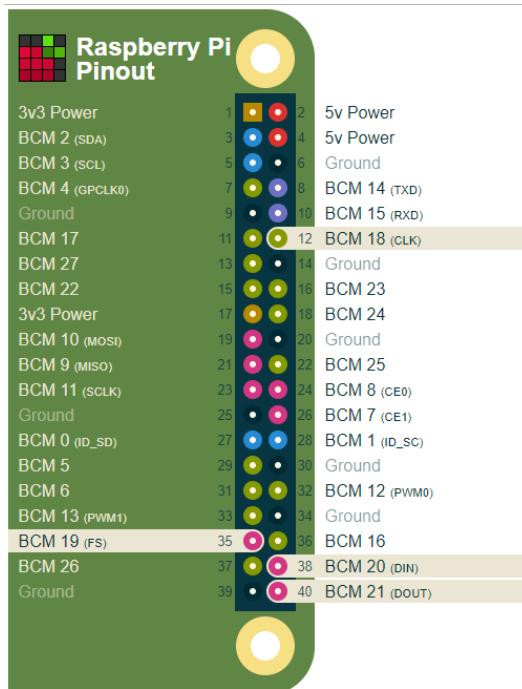
One of the alternate functions selectable on bank 0 of the Raspberry Pi GPIO is DPI. DPI (Display Parallel Interface) is a 24-bit parallel interface with 28 clock and synchronisation signals.

This interface allows parallel RGB displays to be attached to the Raspberry Pi GPIO either in RGB24 (8 bits for red, green and blue) or RGB666 (6 bits per colour) or RGB565 (5 bits red, 6 green, and 5 blue). It is available as alternate function 2 (ALT 2) on GPIO bank 0.

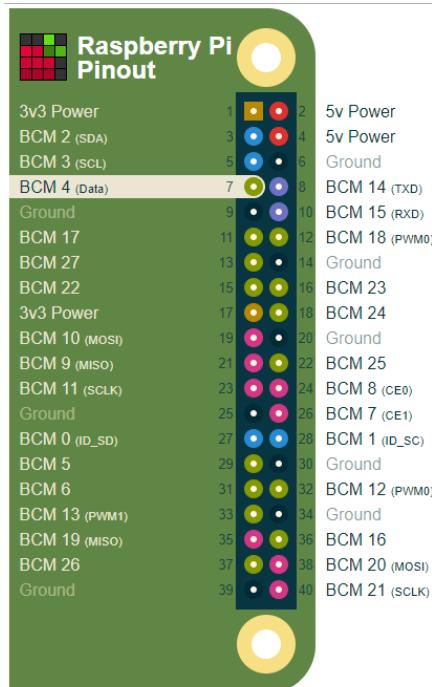
The pinout presented here is for the RGB24 mode, see url below for documentation of the RGB666 and RGB565 modes.

### Details

- Uses 28 GPIO pins
- [More Information](#)

**Legend**

- GPIO (General Purpose IO)
- SPI (Serial Peripheral Interface)
- I<sup>2</sup>C (Inter-Integrated Circuit)
- UART (Universal Asynchronous Receiver/Transmitter)
- Ground
- 5V (Power)
- 3.3V (Power)

**Legend**

- GPIO (General Purpose IO)
- SPI (Serial Peripheral Interface)
- I<sup>2</sup>C (Inter-Integrated Circuit)
- UART (Universal Asynchronous Receiver/Transmitter)
- Ground
- 5V (Power)
- 3.3V (Power)

| SDIO                                  | JTAG | UART | DPI | PCM | 1-WIRE | WiringPi | GPCLK | Ground | I2C | SPI |
|---------------------------------------|------|------|-----|-----|--------|----------|-------|--------|-----|-----|
| Browse more HATs, pHATs and add-ons » |      |      |     |     |        |          |       |        |     |     |

## PCM - Pulse-code Modulation

PCM (Pulse-code Modulation) is a digital representation of sampled analog. On the Raspberry Pi it's a form of digital audio output which can be understood by a DAC for high quality sound.

### Details

- Uses 4 GPIO pins

| SDIO                                  | JTAG | UART | DPI | PCM | 1-WIRE | WiringPi | GPCLK | Ground | I2C | SPI |
|---------------------------------------|------|------|-----|-----|--------|----------|-------|--------|-----|-----|
| Browse more HATs, pHATs and add-ons » |      |      |     |     |        |          |       |        |     |     |

## W1-GPIO - One-Wire Interface

To enable the one-wire interface you need to add the following line to `/boot/config.txt`, before rebooting your Pi:

```
1. dtoverlay=w1-gpio
```

or

```
1. dtoverlay=w1-gpio,gpiopin=x
```

if you would like to use a custom pin (default is BCM4, as illustrated in pinout herein).

Alternatively you can enable the one-wire interface on demand using `raspi-config`, or the following:

```
1. sudo modprobe w1-gpio
```

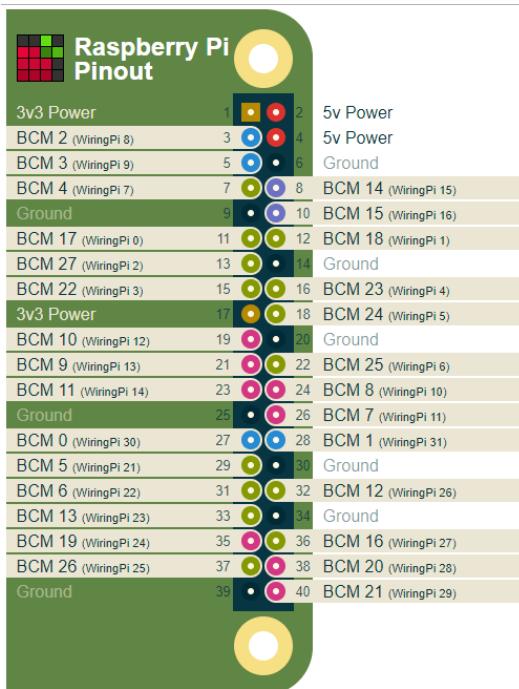
Newer kernels (4.9.28 and later) allow you to use dynamic overlay loading instead, including creating multiple 1-Wire busses to be used at the same time:

```
1. sudo dtoverlay w1-gpio gpiopin=4 pullup=0 # header pin 7
2. sudo dtoverlay w1-gpio gpiopin=17 pullup=0 # header pin 11
3. sudo dtoverlay w1-gpio gpiopin=27 pullup=0 # header pin 13
```

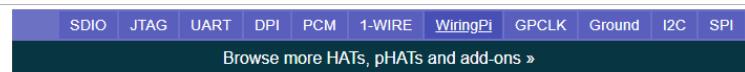
once any of the steps above have been performed, and discovery is complete you can list the devices that your Raspberry Pi has discovered via all 1-Wire busses (by default BCM4), like so:

```
1. ls /sys/bus/w1/devices/
```

n.b. Using `w1-gpio` on the Raspberry Pi typically needs a 4.7 kΩ pull-up resistor connected between the GPIO pin and a 3.3v supply (e.g. header pin 1 or 17). Other means of connecting 1-Wire devices to the Raspberry Pi are also possible, such as using i2c to 1-Wire bridge chips.

**Legend**

- GPIO (General Purpose IO)
- SPI (Serial Peripheral Interface)
- I<sup>2</sup>C (Inter-Integrated Circuit)
- UART (Universal Asynchronous Receiver/Transmitter)
- Ground
- 5V (Power)
- 3.3V (Power)

**Legend**

- GPIO (General Purpose IO)
- SPI (Serial Peripheral Interface)
- I<sup>2</sup>C (Inter-Integrated Circuit)
- UART (Universal Asynchronous Receiver/Transmitter)
- Ground
- 5V (Power)
- 3.3V (Power)

## WiringPi

WiringPi is an attempt to bring Arduino-wiring-like simplicity to the Raspberry Pi.

The goal is to have a single common platform and set of functions for accessing the Raspberry Pi GPIO across multiple languages. WiringPi is a C library at heart, but it's available to both Ruby and Python users who can "gem install wiringpi" or "pip install WiringPi" respectively.

Python users note the 2 on the end, the WiringPi-Python library finally brings a whole host of existing WiringPi functionality to Python including brand new features from WiringPi 2.

WiringPi uses its own pin numbering scheme, here you'll learn how WiringPi numbers your GPIO pins, what those pins do and how to do shiny things with them from within Python or Ruby.

Installing to Python couldn't be easier, just:

```
1. sudo pip install WiringPi
```

For more information about WiringPi you should visit the official [WiringPi website](#).

**Details**

- HAT form-factor
- Uses 28 GPIO pins
- [More Information](#)
- [GitHub Repository](#)

## GPCLK - General Purpose Clock

General Purpose Clock pins can be set up to output a fixed frequency without any ongoing software control.

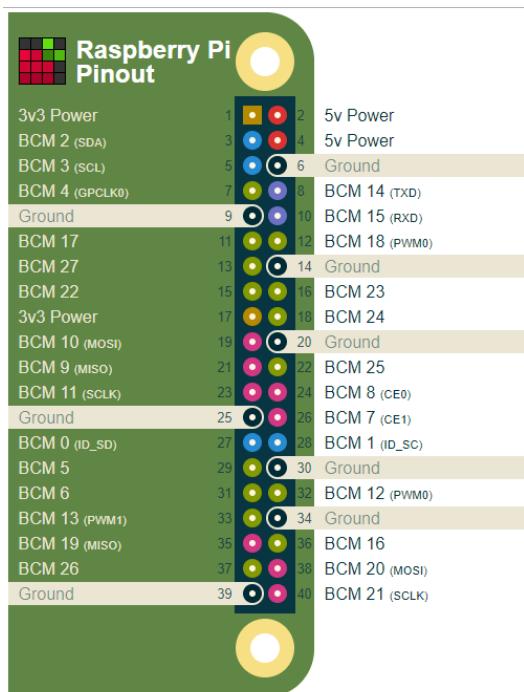
The following clock sources are available:

|    |      |          |                                       |
|----|------|----------|---------------------------------------|
| 1. | 0    | 0 Hz     | Ground                                |
| 2. | 1    | 19.2 MHz | oscillator                            |
| 3. | 2    | 0 Hz     | testdebug0                            |
| 4. | 3    | 0 Hz     | testdebug1                            |
| 5. | 4    | 0 Hz     | PLLA                                  |
| 6. | 5    | 1000 MHz | PLL (changes with overclock settings) |
| 7. | 6    | 500 MHz  | PLLD                                  |
| 8. | 7    | 216 MHz  | HDMI auxiliary                        |
| 9. | 8-15 | 0 Hz     | Ground                                |

Other frequencies can be achieved by setting a clock-divider in the form of SOURCE/(DIV\_I + DIV\_F/4096). Note, that the [BCM2835 ARM Peripherals](#) document contains an error and states that the denominator of the divider is 1024 instead of 4096.

**Details**

- Uses 3 GPIO pins



[SDIO](#) [JTAG](#) [UART](#) [DPI](#) [PCM](#) [1-WIRE](#) [WiringPi](#) [GPCLK](#) [Ground](#) [I2C](#) [SPI](#)

Browse more HATs, pHATs and add-ons »

## Ground

The Ground pins on the Raspberry Pi are all electrically connected, so it doesn't matter which one you use if you're wiring up a voltage supply.

Generally the one that's most convenient or closest to the rest of your connections is tidier and easier, or alternatively the one closest to the supply pin that you use.

For example, it's a good idea to use Physical Pin 17 for 3v3 and Physical Pin 25 for ground when using the SPI connections, as these are right next to the important pins for SPI0.

### Details

- 1 pin header
- Uses 8 GPIO pins



[SDIO](#) [JTAG](#) [UART](#) [DPI](#) [PCM](#) [1-WIRE](#) [WiringPi](#) [GPCLK](#) [Ground](#) [I2C](#) [SPI](#)

Browse more HATs, pHATs and add-ons »

## I2C - Inter Integrated Circuit

**I2C pins in BCM mode are: 2, 3**  
**I2C pins in WiringPi are: 8, 9**

The Raspberry Pi's I2C pins are an extremely useful way to talk to many different types of external peripheral; from the MCP23017 digital IO expander, to a connected ATmega.

The I2C pins include a fixed 1.8 kohms pull-up resistor to 3.3v. This means they are not suitable for use as general purpose IO where a pull-up is not required.

You can verify the address of connected I2C peripherals with a simple one-liner:

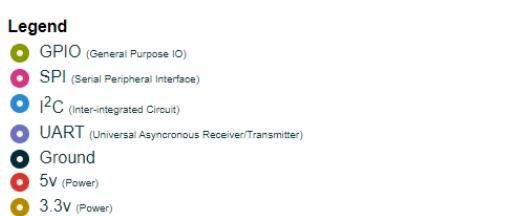
```
1. sudo apt-get install i2c-tools
2. sudo i2cdetect -y 1
```

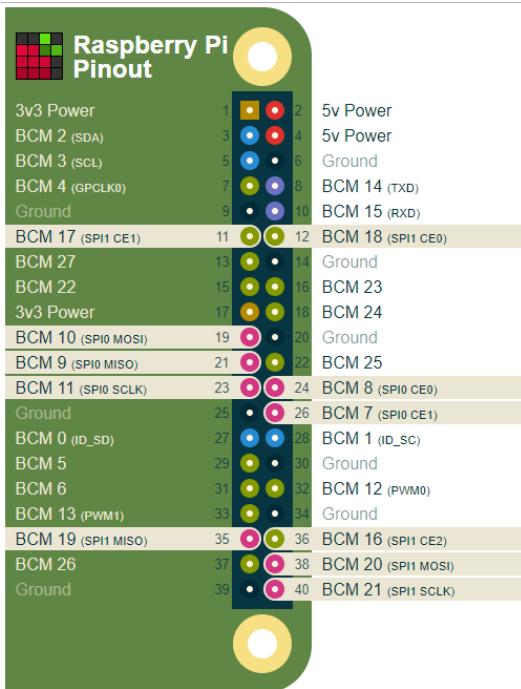
You can then access I2C from Python using the smbus library:

```
1. import smbus
2. DEVICE_BUS = 1
3. DEVICE_ADDR = 0x15
4. bus = smbus.SMBus(DEVICE_BUS)
5. bus.write_byte_data(DEVICE_ADDR, 0x00, 0x01)
```

### Details

- Uses 4 GPIO pins
- [More Information](#)



**Legend**

- GPIO (General Purpose IO)
- SPI (Serial Peripheral Interface)
- I<sup>2</sup>C (Inter-Integrated Circuit)
- UART (Universal Asynchronous Receiver/Transmitter)
- Ground
- 5V (Power)
- 3.3V (Power)

| SDIO                                  | JTAG | UART | DPI | PCM | 1-WIRE | WiringPi | GPCLK | Ground | I2C | SPI |
|---------------------------------------|------|------|-----|-----|--------|----------|-------|--------|-----|-----|
| Browse more HATs, pHATs and add-ons » |      |      |     |     |        |          |       |        |     |     |

**SPI - Serial Peripheral Interface****SPI0 pins in BCM mode are: 9, 10, 11 + 7/8****SPI0 pins in WiringPi are: 12, 13, 14 + 10/11**

Known as the four-wire serial bus, SPI lets you attach multiple compatible devices to a single set of pins by assigning them different chip-select pins.

A useful example of an SPI peripheral is the MCP23S17 digital IO expander chip ( Note the S in place of the 0 found on the I2C version ). You can also use the SPI port to "Bit-Bang" an ATmega 328, loading Arduino sketches onto it with Gordon Hendersons' modified version of AVRDUDE.

To talk to an SPI device, you assert its corresponding chip-select pin. By default the Pi has CE0 and CE1.

```

1. import spidev
2. spi = spidev.SpiDev()
3. spi.open(0, CHIP_SELECT_0_OR_1)
4. spi.max_speed_hz = 1000000
5. spi.xfer([value_8bit])

```

**Details**

- 5 pin header
- Uses 11 GPIO pins
- [More Information](#)

## CONTROLANDO GPIO DESDE EL BASH

Por lo general, la Raspberry Pi utiliza distribuciones Linux. Como buen UNIX, el sistema operativo tratará a todos los elementos, incluido el hardware, como un **fichero**.

Ya sabes que en Windows existen unidades (C:, D:,...) o dispositivos hardware. Pero en un \*nix todo son ficheros, como por ejemplo el **disco duro** (/dev/sda), la unidad de DVD (/dev/dvd), tarjetas SD (/dev/mmcblk0), etc...

Pues bien, los puertos GPIO también serán tratados como un fichero más, aunque no se encuentren en el directorio /dev, y, por tanto, podrás emplear los comandos básicos de la consola para gestionar ficheros. Por ejemplo, si queremos controlar un LED, podríamos entrar en la consola (para nuestros ejemplos hemos empleado Raspbian) y escribir lo siguiente:

```
$ echo 17 > /sys/class/gpio/export
```

Con esta línea creariamos un fichero con la estructura del GPIO correspondiente para que pueda ser manipulado. Luego se debería de configurar como entrada (**in**) o salida (**out**), según lo que quieras. En nuestro caso como salida, puesto que queremos alimentar al LED para encenderlo:

```
$ echo out > /sys/class/gpio/gpio17/direction
```

Ahora se podría encender o apagar. Para ello le podemos dar valores a la salida que hemos creado. Si le damos valor 1 se encenderá y con valor 0 se apagará:

```
$ echo 1 > /sys/class/gpio/gpio17/value
$ echo 0 > /sys/class/gpio/gpio17/value
```

Una vez termines el experimento, puedes borrar la entrada para realizar otro ejercicio en el mismo pin o en otro diferente:

```
$ echo 17 > /sys/class/gpio/unexport
```

¿Simple verdad? Pues ahora puedes utilizar tu imaginación y utilizar otros comandos que normalmente utilizas con ficheros y directorios para gestionar los GPIO. Por ejemplo, puedes listar los GPIO que tienes activos con: \$ ls /sys/class/gpio

Para leer el estado de un GPIO: \$ cat /sys/class/gpio/gpio7/value

## SISTEMAS DE NUMERACIÓN DE PINES

Hay 2 sistemas de numeración de los pines GPIO: BCM y BOARD:

- El sistema **BCM** usa el número de pin GPIO correspondiente. La opción **GPIO.BCM** se refiere a los pines por su número de "Broadcom SOC channel"
- En el sistema **BOARD** la numeración se basa en el orden de los pines de arriba hacia abajo de la placa. En esta imagen se aprecia mejor la diferencia entre los dos sistemas:

| PINS BCM      |          |    | PINS BOARD |    |  | PINS BCM |           |            |
|---------------|----------|----|------------|----|--|----------|-----------|------------|
|               |          |    | P1         |    |  |          |           |            |
| <50mA         | 3V3      |    | 1          | 2  |  | 1        | 2         |            |
| BCM GPIO00/02 | SDA0/1   | 8  | 3          | 4  |  | 5        | 6         |            |
| BCM GPIO01/03 | SCL0/1   | 9  | 7          | 8  |  | 15       | TX        | BCM GPIO14 |
| BCM GPIO04    |          | 7  | 9          | 10 |  | 16       | RX        | BCM GPIO15 |
|               | GND      |    | 11         | 12 |  | 1        | PWM0      | BCM GPIO18 |
| BCM GPIO17    |          | 0  | 13         | 14 |  | 4        |           | BCM GPIO23 |
| BCM GPIO21/27 |          | 2  | 15         | 16 |  | 5        |           | BCM GPIO24 |
| BCM GPIO22    |          | 3  | 17         | 18 |  | 19       | 20        |            |
| <50mA         | 3v3      |    | 21         | 22 |  | 23       | 24        |            |
| BCM GPIO10    | SPIMOSI  | 12 | 25         | 26 |  | 10       | SPI CE0 N | BCM GPIO08 |
| BCM GPIO9     | SPIMOSO  | 13 |            |    |  | 11       | SPI CE1 N | BCM GPIO07 |
| BCM GPIO11    | SPI SCLK | 14 |            |    |  |          |           |            |
|               | GND      |    |            |    |  |          |           |            |

## ENTRADAS Y SALIDAS DIGITALES EN PYTHON

Para programar entradas y salidas digitales en Python tenemos también una amplia variedad de bibliotecas:

- La biblioteca **GPIO Zero** proporciona una interfaz abstracta muy sencilla de usar para una amplia variedad de periféricos. Es especialmente cómoda para el uso de entradas y salidas digitales.
- La biblioteca **RPi.GPIO** es un módulo que proporciona acceso exclusivamente a las entradas y salidas digitales.
- La biblioteca **wiringPi2** no es más que un envoltorio de la biblioteca C del mismo nombre.
- La biblioteca **pigpio** es un envoltorio de la interfaz a pigiod en C

### CONFIGURACIÓN:

Para el control de los pines I/O, la cabecera del programa deberá configurar esas entradas y salidas. Ejemplo:

```
GPIO.setmode(GPIO.BCM) # Establecemos el sistema de numeración de pins BCM
GPIO.setwarnings(False) # Deshabilita los avisos
```

```
Entradas:
btnSubir = 10
btnStop = 9
btnBajar = 11
GPIO.setup(btnSubir, GPIO.IN)
GPIO.setup(btnStop, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(btnBajar, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

Salidas
outSubir = 18
outBajar = 24
GPIO.setup(outSubir, GPIO.OUT, initial=GPIO.HIGH)
GPIO.setup(outBajar, GPIO.OUT, initial=GPIO.LOW)
```

Como podéis ver, en las entradas, podemos añadir la opción de Pull\_up o de Pull\_down, o bien dejarlo sin nada y hacerlo por hardware.

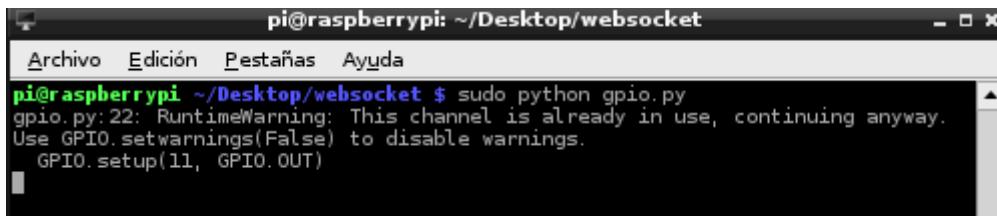
Para las salidas, podremos indicar mediante orden opcional, el estado inicial que queramos que tenga:

```
GPIO.setup(outSubir, GPIO.OUT, initial=1)
GPIO.setup(outSubir, GPIO.OUT, initial=0)
```

## CERRAR PROCESOS”

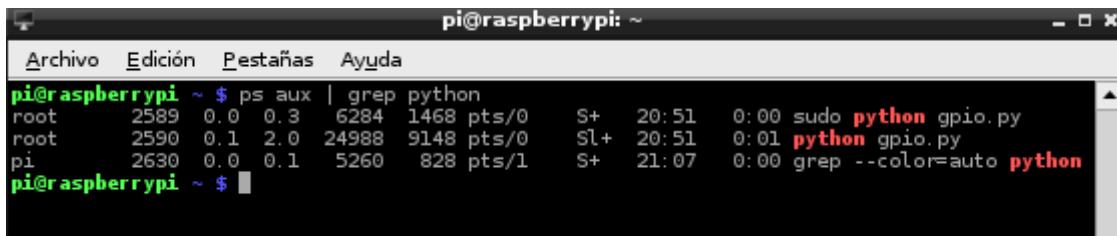
Cuando se está trabajando con el módulo GPIO del Raspberry Pi, si por un error en el programa, el canal queda en ejecución al no ser cerrado adecuadamente. Esto genera el error: *RuntimeWarning: This channel is already in use, continuing anyway. Use GPIO.setwarnings(False) to disable warnings.*

Este error, permite que el programa se ejecute, pero al estar el canal en uso, evita que se pueda acceder al puerto. Para poder tener de nuevo acceso al puerto, se debe cerrar o "matar" el proceso que se quedó en ejecución.



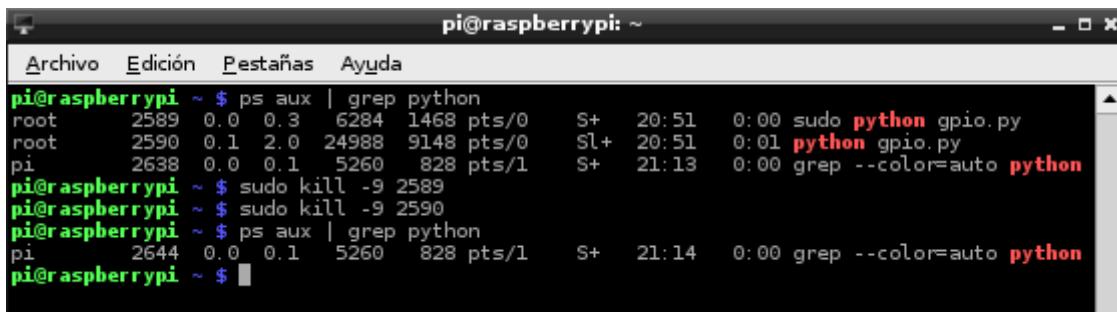
```
pi@raspberrypi: ~/Desktop/websocket
Archivo Edición Pestañas Ayuda
pi@raspberrypi: ~/Desktop/websocket $ sudo python gpio.py
gpio.py:22: RuntimeWarning: This channel is already in use, continuing anyway.
Use GPIO.setwarnings(False) to disable warnings.
 GPIO.setup(11, GPIO.OUT)
```

En la consola, ejecutar el comando: **\$ ps aux | grep python** para poder ver los procesos que están corriendo actualmente.



```
pi@raspberrypi: ~
Archivo Edición Pestañas Ayuda
pi@raspberrypi: ~ $ ps aux | grep python
root 2589 0.0 0.3 6284 1468 pts/0 S+ 20:51 0:00 sudo python gpio.py
root 2590 0.1 2.0 24988 9148 pts/0 Sl+ 20:51 0:01 python gpio.py
pi 2630 0.0 0.1 5260 828 pts/1 S+ 21:07 0:00 grep --color=auto python
pi@raspberrypi: ~ $
```

Después de identificar los procesos que se quedaron en ejecución, se pueden finalizar con el comando: **sudo kill -9 xxxx**, donde **xxxx** corresponde con el **PID** del proceso:



```
pi@raspberrypi: ~
Archivo Edición Pestañas Ayuda
pi@raspberrypi: ~ $ ps aux | grep python
root 2589 0.0 0.3 6284 1468 pts/0 S+ 20:51 0:00 sudo python gpio.py
root 2590 0.1 2.0 24988 9148 pts/0 Sl+ 20:51 0:01 python gpio.py
pi 2638 0.0 0.1 5260 828 pts/1 S+ 21:13 0:00 grep --color=auto python
pi@raspberrypi: ~ $ sudo kill -9 2589
pi@raspberrypi: ~ $ sudo kill -9 2590
pi@raspberrypi: ~ $ ps aux | grep python
pi 2644 0.0 0.1 5260 828 pts/1 S+ 21:14 0:00 grep --color=auto python
pi@raspberrypi: ~ $
```

Para verificar se puede ejecutar de nuevo el comando: **\$ ps aux | grep python**

### IMPORTANTE:

Ahora, se debe evitar que se vuelva a producir este error, liberando los canales de GPIO, con el comando **GPIO.cleanup()**, al cerrar el programa en ejecución, con las instrucciones:

```
try:
 main
finally:
 GPIO.cleanup()
```

## LANZAR UN SCRIPT AL ARRANCAR EL SISTEMA

### SCRIPT BASH (Intérprete de comandos)

1. Creamos un fichero llamado “**uno.sh**” con el siguiente contenido (**Activa un led durante 5sg**):

```
$ sudo nano Desktop/uno.sh
#!/bin/bash
echo 17 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio17/direction
echo 1 > /sys/class/gpio/gpio17/value
sleep 5
echo 0 > /sys/class/gpio/gpio17/value
echo 17 > /sys/class/gpio/unexport
```

2. Damos permiso de ejecución:

```
$ sudo chmod 777 Desktop/uno.sh
```

3. Añadimos en el fichero “**/etc/rc.local**” la siguiente línea, antes del “**exit 0**”:

```
$ sudo nano /etc/rc.local
bash /home/pi/Desktop/uno.sh
```

```
pi@rpicovid19: ~
Archivo Editar Pestañas Ayuda
GNU nano 3.2 /etc/rc.local
#
By default this script does nothing.

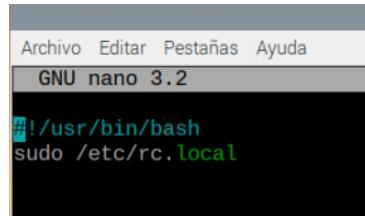
Print the IP address
_IP=$(hostname -I) || true
if ["$_IP"]; then
 printf "My IP address is %s\n" "$_IP"
fi

bash /home/pi/Desktop/uno.sh

exit 0
```

4. Creamos un fichero script en “**/etc/init.d**” (por ejemplo, lo llamamos “**inicio**”) con el siguiente contenido:

```
#!/usr/bin/bash
sudo /etc/rc.local
```



5. Ejecutamos el siguiente comando para que lo ejecute al arrancar.

```
$ sudo update-rc.d inicio defaults
```

6. Tras reiniciar la raspberry, al arrancar se iluminará el **led del GPIO 17** durante **5sg**.

**SCRIPT PYTHON**

1. Creamos un fichero llamado “**uno.py**” con el siguiente contenido (**Activa un led durante 5sg**):

```
$ sudo nano Desktop/uno.py
#!/usr/bin/python
#_*_ coding: utf-8 __

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)

GPIO.output(17, True)
time.sleep(5)
GPIO.output(17, False)
GPIO.cleanup()
```

2. Damos permiso de ejecución:

```
$ sudo chmod 777 Desktop/uno.py
```

3. Añadimos en el fichero “**/etc/rc.local**” la siguiente línea, antes del “**exit 0**”:

```
$ sudo nano /etc/rc.local
python /home/pi/Desktop/uno.py
```

```
pi@rpicovid19: ~
Archivo Editar Pestañas Ayuda
GNU nano 3.2 /etc/rc.local

#
By default this script does nothing.

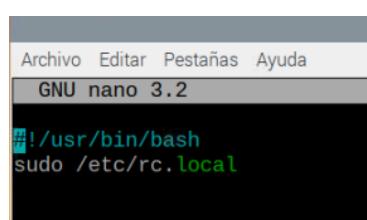
Print the IP address
_IP=$(hostname -I) || true
if ["$_IP"]; then
 printf "My IP address is %s\n" "$_IP"
fi

python /home/pi/Desktop/uno.py

exit 0
```

4. Creamos un fichero script en “**/etc/init.d**” (por ejemplo, lo llamamos “**inicio**”) con el siguiente contenido:

```
#!/usr/bin/bash
sudo /etc/rc.local
```



5. Ejecutamos el siguiente comando para que lo ejecute al arrancar.

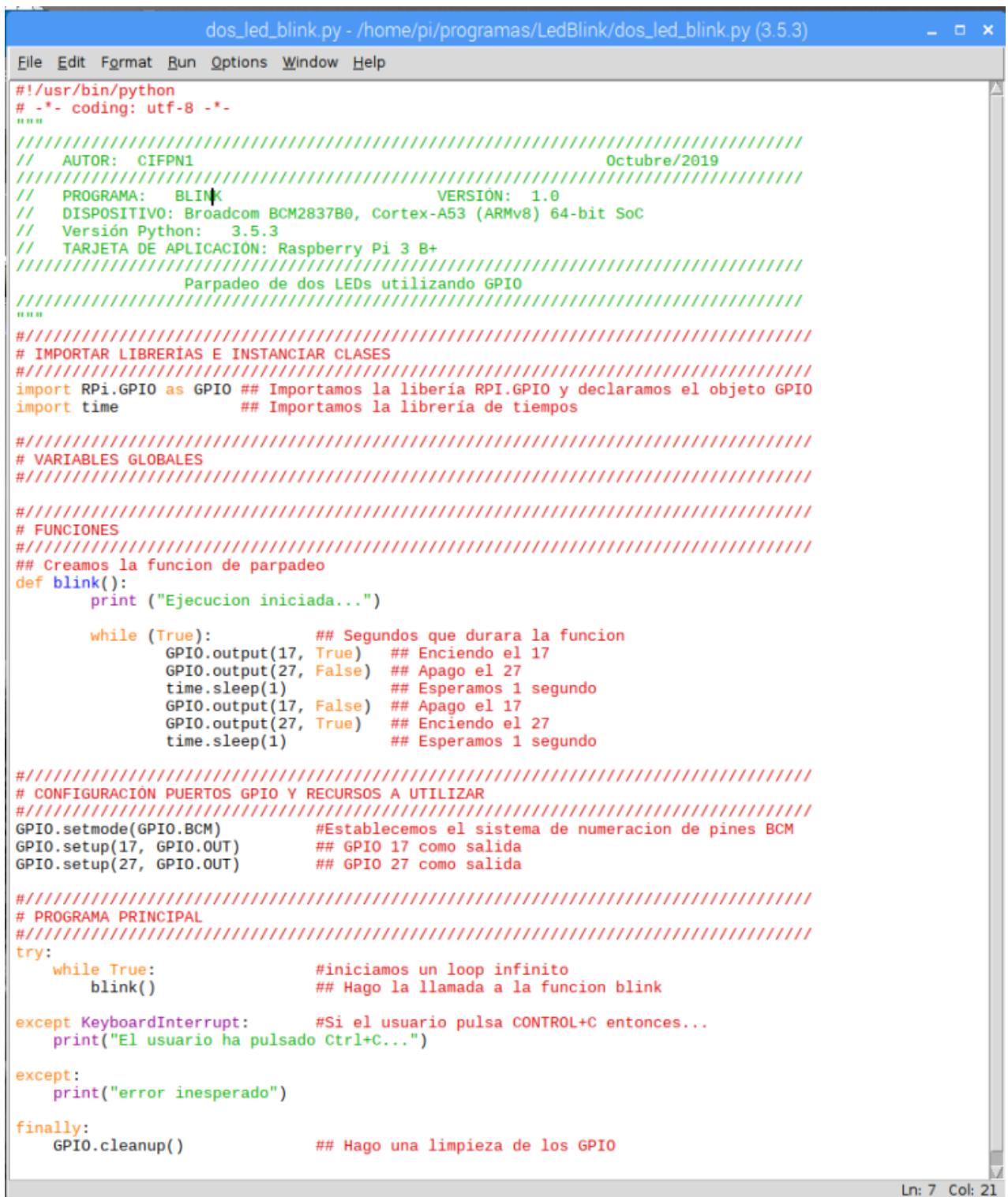
```
$ sudo update-rc.d inicio defaults
```

6. Tras reiniciar la raspberry, al arrancar se iluminará el led del **GPIO 17** durante **5sg**.

**Nota.-** Hay que distinguir entre 3 arranques: **encender, reiniciar y apagar**. Este método que hemos explicado para la Raspberry Pi o Linux sirve para **encender**.

Para **apagar o reiniciar** hay que cambiar el archivo “**/etc/rc6.local**”.

## Ejemplo 1.- ENCENDIDO INTERMITENTE DE DOS LEDS



The screenshot shows a terminal window titled "dos\_led\_blink.py - /home/pi/programas/LedBlink/dos\_led\_blink.py (3.5.3)". The window contains Python code for controlling two LEDs on a Raspberry Pi. The code includes comments in Spanish explaining the purpose and implementation of the program. It uses the RPi.GPIO library to control GPIO pins 17 and 27, and the time library to add delays between LED state changes. The program enters an infinite loop where it alternates the states of the two LEDs every second. It also handles keyboard interrupts to gracefully exit the program.

```
dos_led_blink.py - /home/pi/programas/LedBlink/dos_led_blink.py (3.5.3)
File Edit Format Run Options Window Help
#!/usr/bin/python
-*- coding: utf-8 -*-
"""
// AUTOR: CIFPN1 Octubre/2019
// PROGRAMA: BLINK VERSIÓN: 1.0
// DISPOSITIVO: Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC
// Versión Python: 3.5.3
// TARJETA DE APLICACIÓN: Raspberry Pi 3 B+
// Parpadeo de dos LEDs utilizando GPIO
"""

IMPORTAR LIBRERIAS E INSTANCIAR CLASES
import RPi.GPIO as GPIO ## Importamos la librería RPI.GPIO y declaramos el objeto GPIO
import time ## Importamos la librería de tiempos

VARIABLES GLOBALES

FUNCIONES
Creamos la función de parpadeo
def blink():
 print ("Ejecucion iniciada...")
 while (True):
 GPIO.output(17, True) ## Enciendo el 17
 GPIO.output(27, False) ## Apago el 27
 time.sleep(1) ## Esperamos 1 segundo
 GPIO.output(17, False) ## Apago el 17
 GPIO.output(27, True) ## Enciendo el 27
 time.sleep(1) ## Esperamos 1 segundo

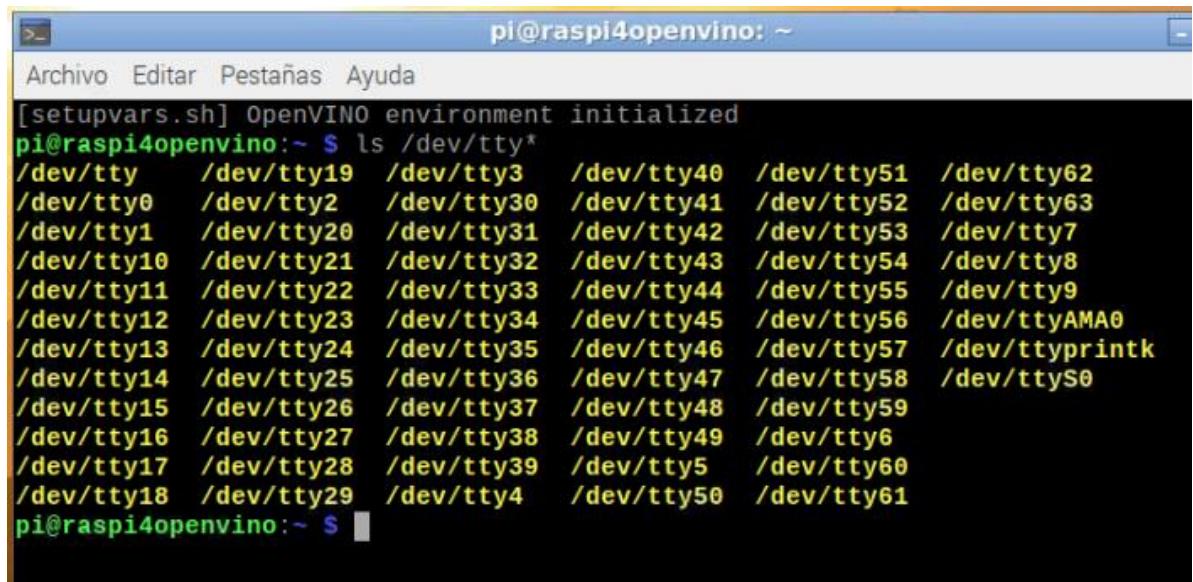
CONFIGURACION PUERTOS GPIO Y RECURSOS A UTILIZAR
GPIO.setmode(GPIO.BCM) ##Establecemos el sistema de numeración de pines BCM
GPIO.setup(17, GPIO.OUT) ## GPIO 17 como salida
GPIO.setup(27, GPIO.OUT) ## GPIO 27 como salida

PROGRAMA PRINCIPAL
try:
 while True: ##iniciamos un loop infinito
 blink() ## Hago la llamada a la función blink
except KeyboardInterrupt: ##Si el usuario pulsa CONTROL+C entonces...
 print("El usuario ha pulsado Ctrl+C...")
except:
 print("error inesperado")
finally:
 GPIO.cleanup() ## Hago una limpieza de los GPIO

```

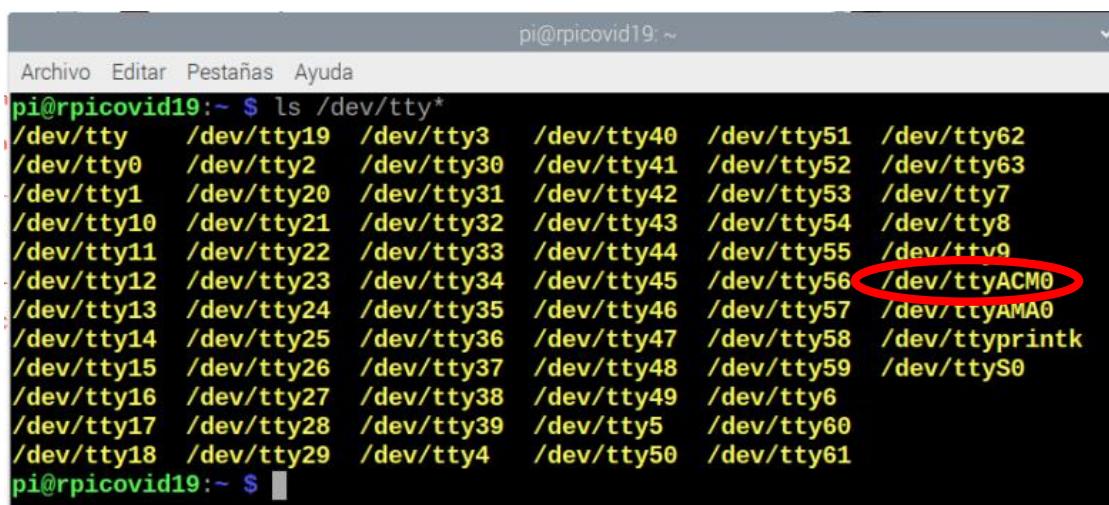
## Ejemplo 2.- CONTROL BIDIRECCIONAL “ARDUINO-RASPI” POR USB (vía SERIAL)

.- Comprobamos los distintos puertos de los que dispone la RASPI → \$ ls /dev/tty\*



```
pi@raspi4openvino: ~
Archivo Editar Pestañas Ayuda
[setupvars.sh] OpenVINO environment initialized
pi@raspi4openvino:~ $ ls /dev/tty*
/dev/tty /dev/tty19 /dev/tty3 /dev/tty40 /dev/tty51 /dev/tty62
/dev/tty0 /dev/tty2 /dev/tty30 /dev/tty41 /dev/tty52 /dev/tty63
/dev/tty1 /dev/tty20 /dev/tty31 /dev/tty42 /dev/tty53 /dev/tty7
/dev/tty10 /dev/tty21 /dev/tty32 /dev/tty43 /dev/tty54 /dev/tty8
/dev/tty11 /dev/tty22 /dev/tty33 /dev/tty44 /dev/tty55 /dev/tty9
/dev/tty12 /dev/tty23 /dev/tty34 /dev/tty45 /dev/tty56 /dev/ttyAMA0
/dev/tty13 /dev/tty24 /dev/tty35 /dev/tty46 /dev/tty57 /dev/ttprintfk
/dev/tty14 /dev/tty25 /dev/tty36 /dev/tty47 /dev/tty58 /dev/ttyS0
/dev/tty15 /dev/tty26 /dev/tty37 /dev/tty48 /dev/tty59
/dev/tty16 /dev/tty27 /dev/tty38 /dev/tty49 /dev/tty6
/dev/tty17 /dev/tty28 /dev/tty39 /dev/tty5 /dev/tty60
/dev/tty18 /dev/tty29 /dev/tty4 /dev/tty50 /dev/tty61
pi@raspi4openvino:~ $
```

.- Conectamos el ARDUINO a la RASPI con el cable USB que utilizamos para descargar los sketches y volvemos a ejecutar → \$ ls /dev/tty\*, comprobando que aparece un dispositivo adicional, el ARDUINO.

```
pi@rpicovid19: ~
Archivo Editar Pestañas Ayuda
pi@rpicovid19:~ $ ls /dev/tty*
/dev/tty /dev/tty19 /dev/tty3 /dev/tty40 /dev/tty51 /dev/tty62
/dev/tty0 /dev/tty2 /dev/tty30 /dev/tty41 /dev/tty52 /dev/tty63
/dev/tty1 /dev/tty20 /dev/tty31 /dev/tty42 /dev/tty53 /dev/tty7
/dev/tty10 /dev/tty21 /dev/tty32 /dev/tty43 /dev/tty54 /dev/tty8
/dev/tty11 /dev/tty22 /dev/tty33 /dev/tty44 /dev/tty55 /dev/tty9
/dev/tty12 /dev/tty23 /dev/tty34 /dev/tty45 /dev/tty56 /dev/ttyACM0
/dev/tty13 /dev/tty24 /dev/tty35 /dev/tty46 /dev/tty57 /dev/ttymA0
/dev/tty14 /dev/tty25 /dev/tty36 /dev/tty47 /dev/tty58 /dev/ttprintfk
/dev/tty15 /dev/tty26 /dev/tty37 /dev/tty48 /dev/tty59 /dev/ttyS0
/dev/tty16 /dev/tty27 /dev/tty38 /dev/tty49 /dev/tty6
/dev/tty17 /dev/tty28 /dev/tty39 /dev/tty5 /dev/tty60
/dev/tty18 /dev/tty29 /dev/tty4 /dev/tty50 /dev/tty61
pi@rpicovid19:~ $
```

```

#!/usr/bin/python
-*- coding: utf-8 -*-
"""
// AUTOR: CIFPN1
// PROGRAMA: ARDU-RASPI VERSIÓN: 1
// DISPOSITIVO: Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit
// Versión Python: 3.5.3
// TARJETA DE APLICACIÓN: Raspberry Pi 3 B+
// Comunicación BIDIRECCIONAL entre Arduino y Ras

La ejecución del programa la realizaremos desde el BASH:
$sudo python3 rs232_2.py
"""

IMPORTAR LIBRERÍAS E INSTANCIAR CLASES
#####
import serial
import time

VARIABLES GLOBALES
#####

FUNCIONES
#####

CONFIGURACION PUERTOS GPIO Y RECURSOS A UTILIZAR
#####
port = serial.Serial('/dev/ttyACM0', 9600) #Inicializamos el Ar

PROGRAMA PRINCIPAL
#####
print("Starting!")
port.flushInput() #Limpiamos buffer de la UART creada por el Arduino
time.sleep(2) #Retardamos 2sg

try:
 while True:
 comando=input("Introduzca un comando...: ") #El arduino espera la palabra "arduino"
 if len(comando)>0:
 port.write(comando.encode()) #conversión str-->byte
 rcv = port.read(4) #leemos 4 bytes que esperamos del Arduino
 rcv1=rcv.decode() #Conversión byte-->str
 print("\r\nEl Arduino responde: " + rcv1)
 port.flushInput() #Limpiamos el buffer de la UART
 #además del comando nos llega (/n /r)
 #es decir, retorno de carro y cambio de línea

 #Si el comando recibido es el correcto, lo imprimimos en pantalla
 if rcv1 == "-OK-":
 print('CORRECTO')
 print('Adios...')
 break #Si el comando es correcto finalizamos el programa
 #Es decir, finalizamos el bucle "while"
 if rcv1 != "-OK-":
 print('ERROR')

except KeyboardInterrupt: #Si el usuario pulsa CONTROL+C entonces...
 print("El usuario ha pulsado Ctrl+C...")

except:
 print("error inesperado")

finally:
 port.close() #Finalizamos la comunicación

```

```

bidi_2019 Arduino 1.8.12
Archivo Editar Programa Herramientas Ayuda
0
bidi_2019
void setup()
{
 Serial.begin(9600);
}

void loop()
{
 if (Serial.available())
 {
 String data = Serial.readStringUntil('\n');
 if (data=="arduino")
 {
 Serial.print("-OK-");
 Serial.flush();
 }
 if (Serial.available()>0)
 {
 Serial.read(); //Vaciamos buffer
 }
 }
 else
 {
 Serial.print("-KO-");
 Serial.flush();
 }
 if (Serial.available()>0)
 {
 Serial.read(); //vaciamos buffer
 }
}

```

Guardado.

El Sketch usa 3042 bytes (9%) del espacio de almacenamiento  
Las variables Globales usan 212 bytes (10%) de la memoria

34 Arduino Uno en /dev/ttyACM0

#### 1. To convert a string to bytes.

```

data = "" #string
data = "".encode() #bytes
data = b"" #bytes

```

#### 2. To convert bytes to a String.

```

data = b"" #bytes
data = b"".decode() #string
data = str(b"") #string

```

## Ejemplo 3.- PWM

### Funciones

- Para crear una instancia PWM :

```
p = GPIO.PWM(channel, frequency)
```

- Para iniciar PWM:

```
p.start(dc) # where dc is the duty cycle (0.0 <= dc <= 100.0)
```

- Para cambiar la frecuencia:

```
p.ChangeFrequency(freq) # where freq is the new frequency in Hz
```

- Para cambiar el duty cycle:

```
p.ChangeDutyCycle(dc) # where 0.0 <= dc <= 100.0
```

- Para parar PWM:

```
p.stop()
```

- Note that PWM will also stop if the instance variable 'p' goes out of scope.

### Ejemplo de un LED parpadeando cada 2sg.:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.OUT)

p = GPIO.PWM(17, 0.5)
p.start(10)
input('Presione "Return" para parar:') # use raw_input para Python 2
p.stop()
GPIO.cleanup()#imprescindible soltar el recurso cuando acabemos
```

### An example to brighten/dim an LED:

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(12, GPIO.OUT)

p = GPIO.PWM(12, 50) # channel=12 frequency=50Hz
p.start(0)
try:
 while 1:
 for dc in range(0, 101, 5):
 p.ChangeDutyCycle(dc)
 time.sleep(0.1)
 for dc in range(100, -1, -5):
 p.ChangeDutyCycle(dc)
 time.sleep(0.1)
except KeyboardInterrupt:
 pass
p.stop()
GPIO.cleanup()
```

**Ejemplo de control de intensidad luminosa de dos LEDs**

```
import RPi.GPIO as GPIO # Cargamos la librería RPi.GPIO
from time import sleep # cargamos la función sleep del módulo time

GPIO.setmode(GPIO.BCM) # Ponemos la Raspberry en modo BCM

GPIO.setup(25, GPIO.OUT) # Ponemos el pin GPIO nº25 como salida para el LED #1
GPIO.setup(24, GPIO.OUT) # Ponemos el pin GPIO nº24 como salida para el LED #2

white = GPIO.PWM(25, 100) # Creamos el objeto 'white' en el pin 25 a 100 Hz
red = GPIO.PWM(24, 100) # Creamos el objeto 'red' en el pin 24 a 100 Hz

white.start(0) # Iniciamos el objeto 'white' al 0% del ciclo de trabajo (completamente apagado)
red.start(100) # Iniciamos el objeto 'red' al 100% del ciclo de trabajo (completamente encendido)

A partir de ahora empezamos a modificar los valores del ciclo de trabajo

pause_time = 0.02 # Declaramos un lapso de tiempo para las pausas

try: # Abrimos un bloque 'Try...except KeyboardInterrupt'
 while True: # Iniciamos un bucle 'while true'
 for i in range(0,101): # De i=0 hasta i=101 (101 porque el script se detiene al 100%)
 white.ChangeDutyCycle(i) # LED #1 = i
 red.ChangeDutyCycle(100 - i) # LED #2 resta 100 - i
 sleep(pause_time) # Pequeña pausa para no saturar el procesador
 for i in range(100,-1,-1): # Desde i=100 a i=0 en pasos de -1
 white.ChangeDutyCycle(i) # LED #1 = i
 red.ChangeDutyCycle(100 - i) # LED #2 resta 100 - i
 sleep(pause_time) # Pequeña pausa para no saturar el procesador

 except KeyboardInterrupt: # Se ha pulsado CTRL+C!!
 print("Se ha pulsado CTRL+C")
 except:
 print("error inesperado")
 finally:
 white.stop() # Detenemos el objeto 'white'
 red.stop() # Detenemos el objeto 'red'
 GPIO.cleanup() # Limpiamos los pines GPIO y salimos
```

## Control de SERVO MOTOR

### Parallax Standard Servo (#900-00005)

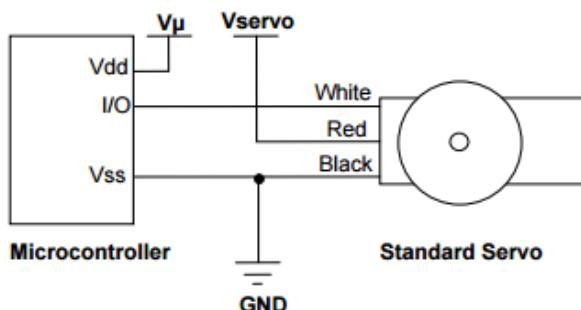
#### Features

- Holds any position between 0 and 180 degrees
- 38 oz-in torque at 6 VDC
- Accepts four mounting screws
- Easy to interface with any Parallax microcontroller or PWM-capable device
- Simple to control with the PULSOUT command in PBASIC
- High-precision gear made of POM (polyacetal) resin makes for smooth operation with no backlash
- Weighs only 1.55 oz (44 g)



#### key Specifications

- Power requirements: 4 to 6 VDC\*; Maximum current draw is 140 +/- 50 mA at 6 VDC when operating in no load conditions, 15 mA when in static state
- Communication: Pulse-width modulation, 0.75–2.25 ms high pulse, 20 ms intervals
- Dimensions approx 2.2 x 0.8 x 1.6 in (5.58 x 1.9 x 40.6 cm) excluding servo horn
- Operating temperature range: 14 to 122 °F (-10 to +50 °C)



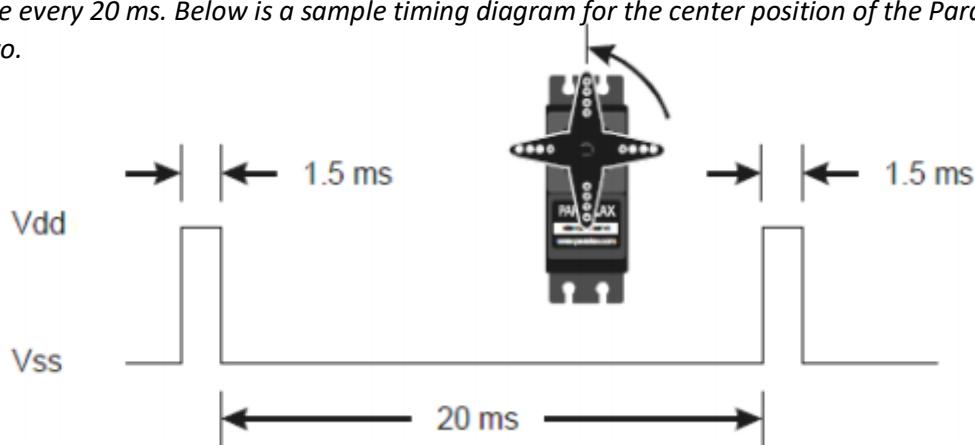
**V<sub>μ</sub>** = microcontroller voltage supply

**V<sub>servo</sub>** = 4 to 6 VDC, regulated or battery

**I/O** = PWM TTL or CMOS output signal from microcontroller: 3.3 to 5 V, not to exceed V<sub>servo</sub> + 0.2 V

#### Communication Protocol

The Parallax Standard Servo is controlled through pulse width modulation, where the position of the servo shaft is dependent on the duration of the pulse. In order to hold its position, the servo needs to receive a pulse every 20 ms. Below is a sample timing diagram for the center position of the Parallax Standard Servo.

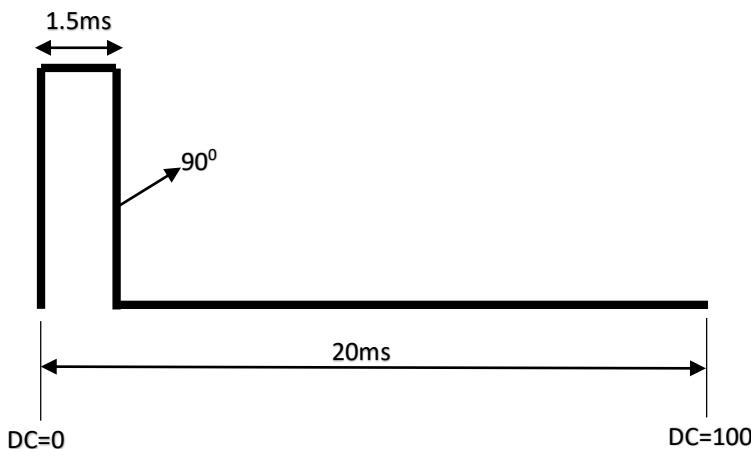


The table below lists the PULSOUT ranges for each BASIC Stamp model.

| BASIC Stamp Module      | 0.75 ms | 1.5 ms (center) | 2.25 ms |
|-------------------------|---------|-----------------|---------|
| BS1                     | 75      | 150             | 225     |
| BS2, BS2e, BS2pe        | 375     | 750             | 1125    |
| BS2sx, BS2px, BS2p24/40 | 938     | 1875            | 2813    |

### EJEMPLO DE POSICIONAMIENTO DEL SERVO INTRODUCIENDO LOS GRADOS POR TECLADO

#### 1º TEÓRICAMENTE

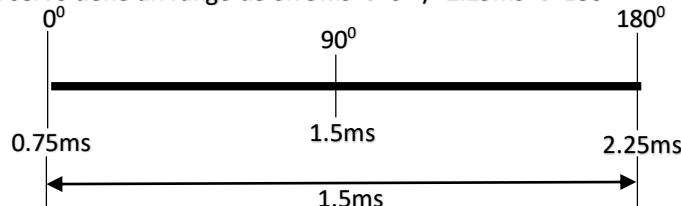


La librería de Python admite un Duty Cicle entre 0-100, por lo que para hallar el DC equivalente a los ms deseados:

$$20 \text{ ms} \xrightarrow{\quad} 100 \text{ DC}$$

$$Nms \xrightarrow{\quad} X \text{ DC}$$

Sabiendo que el servo tiene un rango de 0.75ms  $\rightarrow 0^\circ$  / 2.25ms  $\rightarrow 180^\circ$



Cada grado corresponderá a  $1.5/180=0.0083$

Para saber los ms de DC que corresponden a los grados que deseamos mover:

$$Nms = \text{nº grados} * 0.0083 + 0.75$$

Por lo que finalmente:

$$X = (Nms * 100)/20ms$$

#### 2º EN LA PRÁCTICA

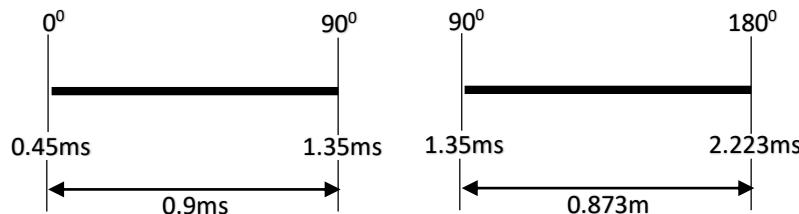
Cambiando el DC y comenzando por un valor máximo para encontrar los  $180^\circ$ , se comprueba que:

$$0^\circ \rightarrow 0.45ms$$

$$90^\circ \rightarrow 1.35ms$$

$$180^\circ \rightarrow 2.223ms$$

Lo cual, implica que los incrementos de ms, entre  $0^\circ$  y  $180^\circ$ , no son iguales. Por ello, dividiremos el ejercicio en dos apartados:

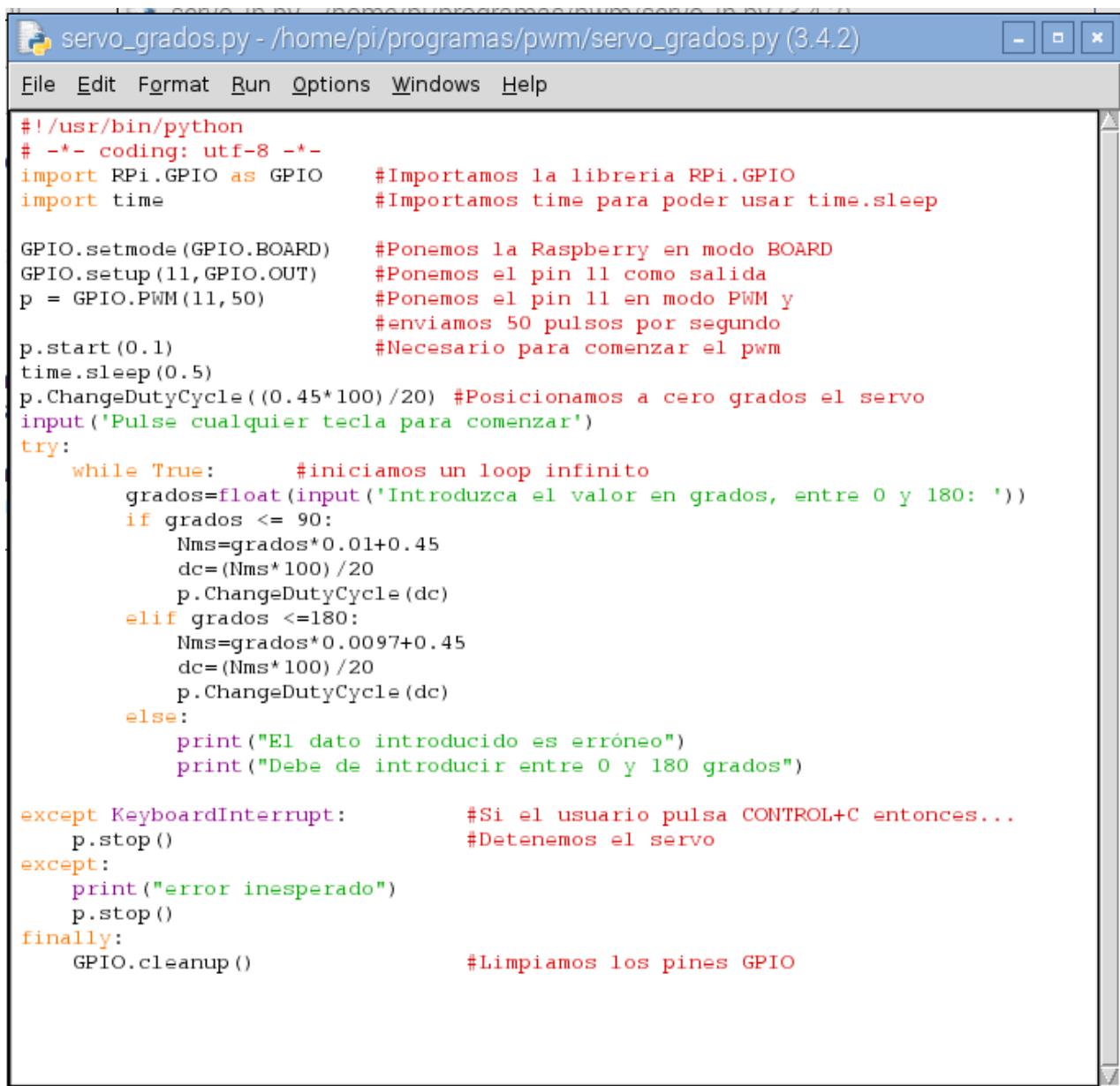


Entre  $0^\circ$ - $90^\circ$   $\rightarrow$  Cada grado corresponderá a  $0.9/90=0.01$

$$Nms = \text{nº grados} * 0.01 + 0.45$$

Entre  $90^\circ$ - $180^\circ$   $\rightarrow$  Cada grado corresponderá a  $0.873/90=0.0097$

$$Nms = \text{nº grados} * 0.0097 + 0.45$$



```
#!/usr/bin/python
-*- coding: utf-8 -*-
import RPi.GPIO as GPIO #Importamos la libreria RPi.GPIO
import time #Importamos time para poder usar time.sleep

GPIO.setmode(GPIO.BOARD) #Ponemos la Raspberry en modo BOARD
GPIO.setup(11,GPIO.OUT) #Ponemos el pin 11 como salida
p = GPIO.PWM(11,50) #Ponemos el pin 11 en modo PWM y
 #enviamos 50 pulsos por segundo
p.start(0.1) #Necesario para comenzar el pwm
time.sleep(0.5)
p.ChangeDutyCycle((0.45*100)/20) #Posicionamos a cero grados el servo
input('Pulse cualquier tecla para comenzar')
try:
 while True: #iniciamos un loop infinito
 grados=float(input('Introduzca el valor en grados, entre 0 y 180: '))
 if grados <= 90:
 Nms=grados*0.01+0.45
 dc=(Nms*100)/20
 p.ChangeDutyCycle(dc)
 elif grados <=180:
 Nms=grados*0.0097+0.45
 dc=(Nms*100)/20
 p.ChangeDutyCycle(dc)
 else:
 print("El dato introducido es erróneo")
 print("Debe de introducir entre 0 y 180 grados")

except KeyboardInterrupt: #Si el usuario pulsa CONTROL+C entonces...
 p.stop() #Detenemos el servo
except:
 print("error inesperado")
 p.stop()
finally:
 GPIO.cleanup() #Limpiamos los pines GPIO
```

## CONTROLAR SERVOMOTOR DE FORMA PRECISA

La función PWM de la biblioteca RPi.GPIO no es precisa, ni libre de temblores en un servo.

El Software **Servo Blaster**, creado por Richard Hurst, utiliza el hardware de la CPU para generar pulsos con tiempos mucho más precisos que los generados por la biblioteca RPi.GPIO.

Instala esto usando los siguientes comandos y luego reinicia la Raspberry Pi:

```
$ git clone git://github.com/richardghirst/PiBits.git
$ cd PiBits/ServoBlaster/user
$ sudo make
$ sudo make install
```

Cuando **ServoBlaster**, o más específicamente el servicio *servo.d*, esté ejecutándose, no se podrá utilizar los siguientes pines:

| Canales servo | Pines GPIO |
|---------------|------------|
| 0             | 4          |
| 1             | 17         |
| 2             | 18         |
| 3             | 27         |
| 4             | 22         |
| 5             | 23         |
| 6             | 24         |
| 7             | 25         |

Cuando necesitemos los pines para otra cosa podemos desactivar el servicio:

```
$ sudo update-rc.d servoblaster disable
$ sudo reboot
```

Para activar el servicio:

```
$ sudo update-rc.d servoblaster enable
$ sudo reboot
```

**EJEMPLO:**

```
#!/usr/bin/python
-*- coding: utf-8 -*-

import time #Importamos time para poder usar time.sleep
import os

servo_min = 500 # Us (0 grados)
servo_max = 2250 # uS (180 grados)

servo = 2 # GPIO 18

def map(value, from_low, from_high, to_low, to_high):
 from_range = from_high - from_low
 to_range = to_high - to_low
 scale_factor = float(from_range) / float(to_range)
 return to_low + (value / scale_factor)

try:
 while True: #iniciamos un loop infinito
 rcv = int(input('Introduzca una posición entre 0 y 180 grados: '))
 pulse = float(map(rcv, 0, 180, servo_min, servo_max))
 command = "echo {}={}us > /dev/servoblaster".format(servo, pulse)
 os.system(command)

except KeyboardInterrupt:
 print("Ctrl+C")
except:
 print("error inesperado")
finally:
 print("Adios")
```

## Ejemplo 4.- INTERRUPCIONES:

Podemos añadir interrupciones a nuestra aplicación y olvidarnos de bucles de lectura.

Las interrupciones son por:

- **RISING** (Activación del pin: 0 -> 1)
- **FALLING** (Desactivación del pin: 1 -> 0)
- **BOTH** (por ambos cambios anteriores).

La implementación puede hacerse de tres modos:

### 1.- POR ESPERA

El programa se queda en este punto hasta que suceda el evento:

```
GPIO.wait_for_edge(num_canal, GPIO.RISING)
```

Ejemplo:

```
#!/usr/bin/python3
import RPi.GPIO as GPIO
import time

Configuramos los puertos
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.IN, pull_up_down=GPIO.PUD_DOWN) #Con la Pull Down interna no fue
 #suficiente. La puse externa
try:
 print('INICIO')
 GPIO.wait_for_edge(17, GPIO.RISING, timeout=5000)
 print('FIN')
except KeyboardInterrupt:
 GPIO.cleanup() #Lo pongo aquí por si salimos con Crtrl+C
GPIO.cleanup()
```

### 2.- POR FLAG

La ponemos la detección de cambio en las entradas al principio. Cuando hay un cambio, el programa sigue ejecutándose y en algún punto de nuestra aplicación le preguntamos si ha habido un cambio:

```
#!/usr/bin/python3
import RPi.GPIO as GPIO
import time

#####
#Configuramos puertos
#####
GPIO.setmode(GPIO.BCM)
GPIO.setup(17, GPIO.IN,pull_up_down=GPIO.PUD_DOWN) #le ponemos una pulldown externa de 4K7
GPIO.add_event_detect(17, GPIO.RISING)

try:
 for i in range(8):
 print (i)
 time.sleep(1)
 if GPIO.event_detected(17):
 print("Evento detectado")

except KeyboardInterrupt:
 pass
 #GPIO.cleanup()
finally:
 GPIO.cleanup()
```

### 3.- POR INTERRUPCIÓN DE PROGRAMA

Cuando se produzca un cambio en el pin, se detendrá el programa y se ejecutará la función indicada:

```
def my_callback(channel):
 print('This is a edge event callback function!')
 print('Edge detected on channel %s'%channel)
 print('This is run in a different thread to your main program')
```

```
GPIO.add_event_detect(channel, GPIO.RISING, callback=my_callback) # add rising edge detection on a
channel
```

...the rest of your program...

### Ejemplo de INTERRUPCIÓN DE PROGRAMA:

```
1 #!/usr/bin/python3
2 #-*- coding: utf-8 -*-
3 import RPi.GPIO as GPIO
4 import time
5 import sys
6
7 ## Configuramos los puertos
8 flag=True
9 GPIO.setmode(GPIO.BCM)
10 GPIO.setup(17, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
11
12 def mi_evento(MCD):
13 #sys.exit(1) #Esta instrucción termina la ejecución del programa
14 contar=1000
15 global flag
16 flag=False
17 print('Se ha detectado que se ha activado el pin %s' % MCD)
18 for i in range(5):
19 contar += 1
20 print(contar)
21 time.sleep(1)
22 flag=True
23
24 GPIO.add_event_detect(17, GPIO.RISING, callback=mi_evento)
25 input('pulse una tecla para continuar')
26 conta=0
27 try:
28 while 1:
29 if flag==True:
30 conta += 1
31 print(conta)
32 time.sleep(1)
33
34 except KeyboardInterrupt: # Se ha pulsado CTRL+C!!
35 print("")
36 print('FIN')
37 except:
38 print("Se ha producido un error inesperado")
39 finally:
40 GPIO.cleanup() # Limpiamos los pines GPIO y salimos
41
```

### SWITCH DEBOUNCE

Al generar interrupción a través de un pulsador mecánico se producen rebotes (**debounce**). Hay dos formas de lidiar contra estos rebotes:

- Añadir un condensador de 0.1uF capacitor en paralelo con el switch.
- Por software
- Combinando ambos

Para eliminar el rebote por software, agregamos el parámetro **bouncetime** a la función donde especificamos la función de interrupción. El tiempo de rebote debe especificarse en milisegundos.

Por ejemplo:

```
add rising edge detection on a channel, ignoring further edges for 200ms for switch bounce handling
GPIO.add_event_detect(channel, GPIO.RISING, callback=my_callback, bouncetime=200)
or
GPIO.add_event_callback(channel, my_callback, bouncetime=200)
```

### REMOVE EVENT DETECTION

Si por alguna razón, en el programa ya no se desea detectar los eventos, es posible detenerlos:

```
GPIO.remove_event_detect(channel)
```

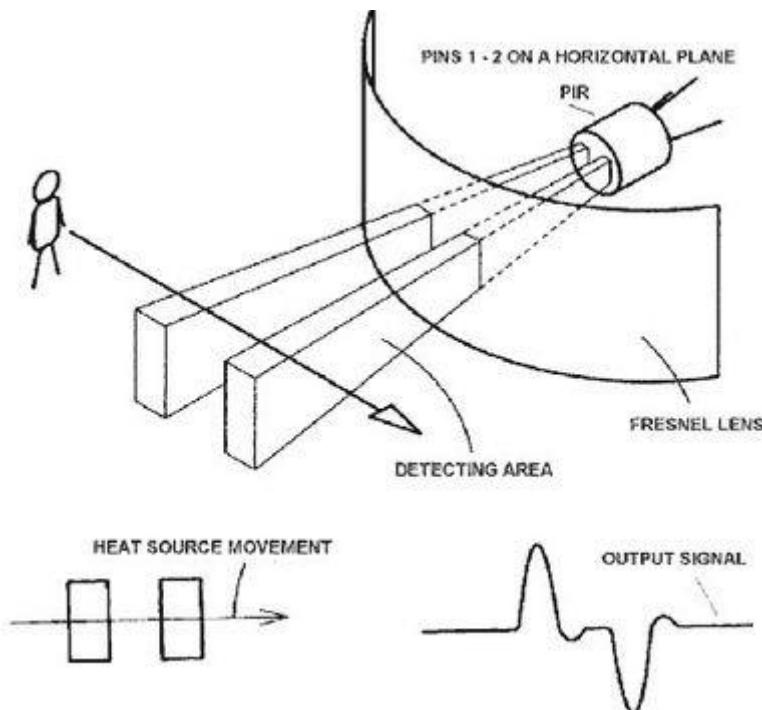
## Ejemplo 5.- SENSOR PIROELÉCTRICO INFRARROJO HC-SR501

Con bastante frecuencia necesitamos algún sistema de detectar la presencia de personas o animales en movimiento en un área dada. Es la base de cualquier **sistema de detección** de intrusos, pero también se usan mucho en las escaleras comunitarias o aseos públicos para encender la luz en cuanto detecta el movimiento.

Todos los seres vivos desprenden calor y lo mismo ocurre con los automóviles y cualquier otra maquinaria, y ese calor se emite en forma de **radiación infrarroja** que podemos detectar con los dispositivos adecuados, como los **sensores PIR**.

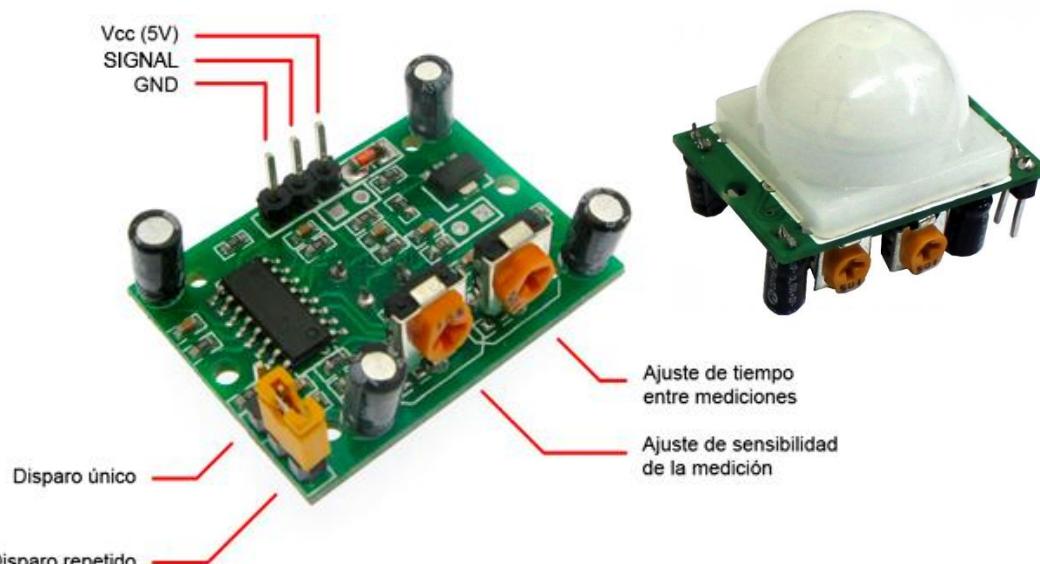
Los **sensores PIR**, son elementos que detectan cambios en la radiación infrarroja que reciben y que disparan una alarma al percibirlo.

Los **PIR** más frecuentes son sensores de movimiento, y para ello están divididos en dos mitades de forma que detecten el cambio de radiación IR que reciben y, además, disparen la alarma cuando perciben ese cambio.



El módulo HC-SR501 tiene 3 pines de conexión +5v, OUT (3,3v) y GND, y dos resistencias variables de calibración (Ch1 y RL2).

- Ch1: Con esta resistencia podemos establecer el tiempo que se va a mantener activa la salida del sensor. Una de las principales limitaciones de este módulo es que el tiempo mínimo que se puede establecer es de más o menos 3s. Si cambiamos la resistencia por otra de 100K, podemos bajar el tiempo mínimo a más o menos 0,5 s.
- RL2: Esta resistencia variable nos permite establecer la distancia de detección que puede variar entre 3-7m.



La posibilidad de mantener activa la salida del módulo durante un tiempo determinado nos permite poder usarlo directamente para prácticamente cualquier aplicación sin necesidad de usar un microcontrolador.

#### Características

- Sensor piroeléctrico (Pasivo) infrarrojo (También llamado PIR)
- El módulo incluye el sensor, lente, controlador PIR BISS0001, regulador y todos los componentes de apoyo para una fácil utilización
- Rango de detección: 3 m a 7 m, ajustable mediante trimmer (Sx)
- Lente fresnel de 19 zonas, ángulo < 100º
- Salida activa alta a 3.3 V
- Tiempo en estado activo de la salida configurable mediante trimmer (Tx)
- Redisparo configurable mediante jumper de soldadura
- Consumo de corriente en reposo: < 50 µA
- Voltaje de alimentación: 4.5 VDC a 20 VDC

#### Ejemplo:

```

import RPi.GPIO as GPIO
from time import sleep # Librería para poder poner tiempos de espera
#####
GPIO.setmode(GPIO.BOARD) # nomenclatura de número de pines
GPIO.setup(7,GPIO.IN) # pin número 7 será la entrada del sensor
#####
try:
 while True:
 if GPIO.input(7): # si el sensor devuelve 1
 print(GPIO.input(7)) # imprime el valor que devuelve el sensor
 print("Hay movimiento")
 else:
 print(GPIO.input(7)) # imprime el valor que devuelve el sensor
 print("No hay movimiento")
 sleep(1) # tiempo de espera de 1 segundo
#####
Esta interrupción se corresponde al evento cuando tecleamos Ctrl+C
except KeyboardInterrupt:
 print ("Paramos el programa")
#####
Este apartado recogerá el resto de excepciones
except:
 print ("Ha habido un error y el programa se ha cerrado")
#####
#Limpiamos los puertos que hayamos utilizado
finally:
 GPIO.cleanup()

```

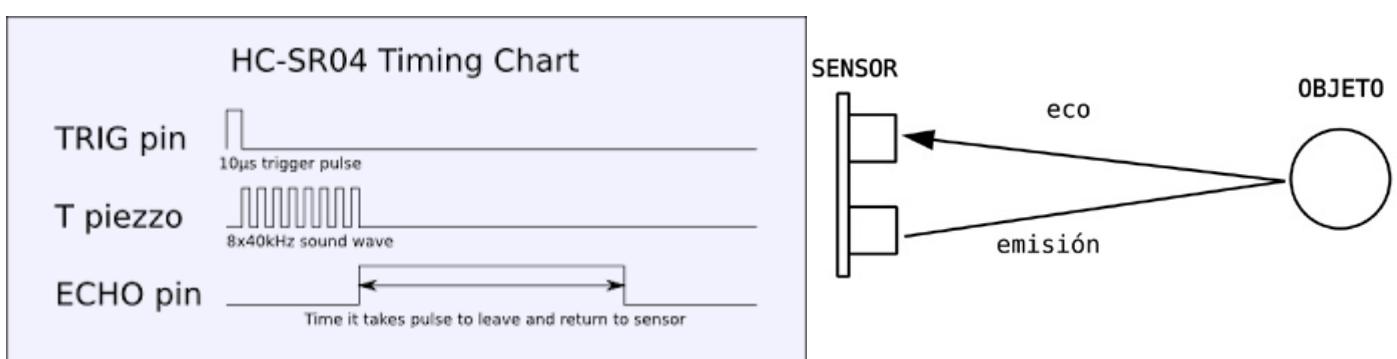
### Ejemplo 6.- SENSOR ULTRASÓNICO HC-SR04

El sensor ultrasónico HC-SR04 es un sensor que, entre otras posibles aplicaciones, sirve para medir distancias. Funciona enviando un pulso de ultrasonidos (inaudible para el oído humano por su alta frecuencia) a través de un transductor (altavoz) y esperando a que dicho sonido rebote sobre un objeto y vuelva. El retorno es captado por el otro transductor del sensor (micrófono).



Sabemos que la velocidad del sonido en el aire (a 20°C) es 343 m/segundo así que, calculando el tiempo transcurrido entre el **envío del pulso y la recepción de la señal** de retorno, y luego aplicando una sencilla fórmula matemática, obtenemos la distancia entre el sensor y el objeto que hay delante.

- Corriente de reposo: < 2mA
- Corriente de trabajo: 15mA
- Ángulo de medición: 30º
- Ángulo de medición efectivo: < 15º
- Detección de 2cm a 400cm o 1" a 13 pies (Sirve a más de 4m, pero el fabricante no garantiza una buena medición).
- “Resolución” La precisión es de unos 3mm.
- Dimensiones: 45mm x 20mm x 15mm
- Frecuencia de trabajo: 40KHz



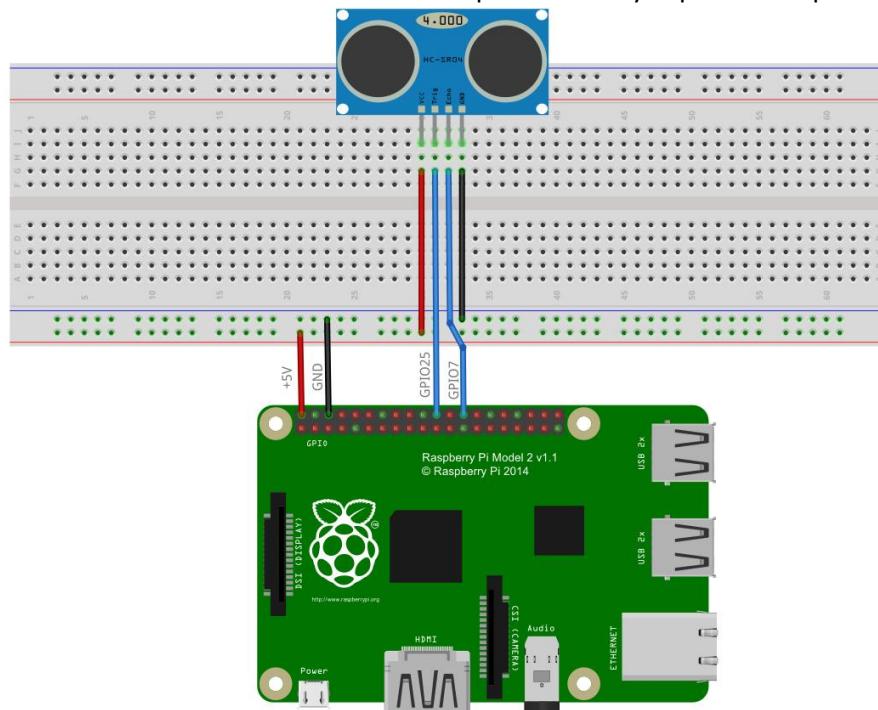
#### Funcionamiento:

1. Enviar un Pulso "1" de al menos de 10uS por el Pin Trigger (Disparador).
2. El sensor enviará 8 Pulses de 40KHz (Ultrasonido) y coloca su salida Echo a alto (seteo), se debe detectar este evento e iniciar un conteo de tiempo.
3. La salida Echo se mantendrá en alto hasta recibir el eco reflejado por el obstáculo a lo cual el sensor pondrá su pin Echo a bajo, es decir, terminar de contar el tiempo.
4. Se recomienda dar un tiempo de aproximadamente 50ms de espera después de terminar la cuenta.
5. Sabemos que la velocidad del sonido en el aire (a 20°C) es 343 m/segundo así que, calculando el tiempo transcurrido entre el **envío del pulso y la recepción de la señal** de retorno, y luego aplicando una sencilla fórmula matemática, obtenemos la distancia entre el sensor y el objeto que hay delante.

**Realizamos el siguiente esquema:**

Conectamos la patilla **Vcc** del sensor a un pin de **5V** de la Raspberry Pi y la patilla **GND** a un pin **GND**. Para comunicar el sensor con la Raspberry conectamos la patilla **TRIGGER** al pin **GPIO 25** y la patilla **ECHO** al pin **GPIO 7**.

**Nota.-** Ponemos una resistencia de **1Kohm** entre la patilla **ECHO** y el pin **GPIO 7** para limitar la corriente.

**Ejemplo:**

```

import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO_TRIGGER = 25
GPIO_ECHO = 7
GPIO.setup(GPIO_TRIGGER,GPIO.OUT) #Configuramos Trigger como salida
GPIO.setup(GPIO_ECHO,GPIO.IN) #Configuramos Echo como entrada
GPIO.output(GPIO_TRIGGER,False) #Ponemos el pin 25 como LOW

try:
 while True:
 GPIO.output(GPIO_TRIGGER,True) #Enviamos un pulso de ultrasonidos
 time.sleep(0.00001) #Una pequeña pausa
 GPIO.output(GPIO_TRIGGER,False) #Apagamos el pulso
 start = time.time() #Guarda el tiempo actual mediante time.time()
 while GPIO.input(GPIO_ECHO)==0: #Mientras el sensor no reciba señal...
 start = time.time() #Mantenemos el tiempo actual mediante time.time()
 while GPIO.input(GPIO_ECHO)==1: #Si el sensor recibe señal...
 stop = time.time() #Guarda el tiempo actual mediante time.time() en otra variable
 elapsed = stop-start #Obtenemos el tiempo transcurrido entre envío y recepción
 distance = (elapsed * 34300)/2 #Distancia es igual a tiempo por velocidad partido
 #por 2 D = (T x V)/2
 print (distance) #Devolvemos la distancia (en centímetros) por pantalla
 time.sleep(1) #Pequeña pausa para no saturar el procesador de la Raspberry
 except KeyboardInterrupt:
 print ("quit") #Si el usuario pulsa CONTROL+C...
 GPIO.cleanup() #Avisamos del cierre al usuario
 #Limpiamos los pines GPIO y salimos

```

## Ejemplo 7.- CONTROL DE LA CÁMARA CON PYTHON

Para el control de la cámara, usaremos una librería nativa de Python llamada Python Picamera.

### INSTALACIÓN

Antes de nada, tendrá que estar habilitada.

Seguidamente ejecutamos:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install Python-picamera (puede que no haga falta instalarlo)
```

### VISTA PREVIA

Esta característica nos permite visualizar en tiempo real la imagen captada por el módulo de cámara. Este es un pequeño script en Python que nos permitirá usar esta función (NO FUNCIONA POR CONTROL REMOTO):

```
#!/usr/bin/Python3
import time
import picamera
with picamera.PiCamera() as picx:
 picx.start_preview()
 time.sleep(60)
 picx.stop_preview()
 picx.close()
```

**Nota:** El intérprete es python3, por lo que no podrá ejecutarse desde el bash.

### CAPTURA DE FOTOGRAFÍAS

```
#!/usr/bin/python
import time
import picamera
with picamera.PiCamera() as picx:
 picx.start_preview()
 time.sleep(5)
 picx.capture('mifoto.jpg')
 picx.stop_preview()
 picx.close()
```

En el ejemplo la imagen se guardará en formato JPG, pero podemos guardarla en otros formatos como PNG, GIF, BMP, RGB, YUV y RAW, lo que facilita poder guardar las imágenes en el formato que más nos convenga.

### CONTROL LED

Podemos desactivar el led del módulo de cámara de la Raspberry Pi. gracias a esta librería:

```
#!/usr/bin/python
import time
import picamera
with picamera.PiCamera() as picam:
 picam.led= False
 picam.start_preview()
 time.sleep(3)
 picam.stop_preview()
 picam.close()
```

### CAPTURAR VÍDEO

Es importante el uso de ‘`wait_recording()`’ ya que de este modo se comprueban ciertos aspectos como por ejemplo si hay suficiente espacio en el disco para grabar:

```
#!/usr/bin/python
import time
import picamera
with picamera.PiCamera() as picx:
 picx.start_preview()
 picx.start_recording('mi_video.h264')
 picx.wait_recording(20)
 picx.stop_recording()
 picx.stop_preview()
 picx.close()
```

Nota: Para reproducir el video debes de tener instalado un reproductor como VLC.

`$ sudo apt-get install vlc`

### AJUSTAR RESOLUCIÓN y REESCALE

El sensor de la cámara tiene aproximadamente 5MPx de resolución (2592×1944 píxeles) pero tan solo se visualizan 1920×1080 píxeles en las pre visualizaciones, por esta razón la librería dispone de una opción para ajustar la resolución. Aquí un ejemplo:

| AJUSTE                                                                                                                                                                                                                                                            | REESCALE                                                                                                                                                                                                                                                                            |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre><code>#!/usr/bin/python import time import picamera with picamera.PiCamera() as picam:     picam.resolution = (2592, 1944)     picam.start_preview()     time.sleep(3)     picam.capture('foto.jpg')     picam.stop_preview()     picam.close()</code></pre> | <pre><code>#!/usr/bin/python import time import picamera with picamera.PiCamera() as picam:     picam.resolution = (2592, 1944)     picam.start_preview()     time.sleep(3)     picam.capture('foto.jpg',resize=(1024,768))     picam.stop_preview()     picam.close()</code></pre> |

### OTROS AJUSTES

Podemos controlar el ISO, la velocidad de obturación, el brillo y además nos permite aplicar efectos como ‘negativo’, ‘solarizado’ y ‘GPen’. Aquí un ejemplo de código:

```
#!/usr/bin/python
import time
import picamera
with picamera.PiCamera() as picam:
 picam.start_preview()
 picam.brightness= 60
 picam.ISO =100
 time.sleep(3)
 picam.image_effect = 'negative'
 picam.shutter_speed= 300000
 picam.capture('foto.jpg',resize=(1024,768))
 picam.stop_preview()
 picam.close()
```

## Ejemplo 8.- INTERACTUANDO CON INTERNET

### 8.1 Intercambio de datos con JSON

A continuación vemos un ejemplo en el que adquirimos información con [JSON](#), también utilizamos el módulo [requests](#) que nos provee de múltiples métodos para obtener todo tipo de información de una url, además de utilidades para normalizar nuestros datos obtenidos.

Antes de nada, “requests” no está incorporado a Python, por lo que lo descargamos:

```
$ sudo pip install requests
```

```
#!/usr/bin/python
-*- coding: utf-8 -*-
import requests
import json

addr_str = raw_input("Dime tu dirección: ")

maps_url = "https://maps.googleapis.com/maps/api/geocode/json"
tiene_sensor = "false"

payload = {'address': addr_str, 'sensor': tiene_sensor}

r = requests.get(maps_url,params=payload)

maps_output = r.json()

lista_resultado = maps_output['results']
estado_resultado = maps_output['status']

direccion = lista_resultado[0]['formatted_address']
result_geo_lat = lista_resultado[0]['geometry']['location']['lat']
result_geo_lng = lista_resultado[0]['geometry']['location']['lng']

print("%s is at\nlat: %f\nlng: %f" % (direccion, result_geo_lat, result_geo_lng))
```

## 8.2 Encendido y Apagado de LED por Web

- Instalamos servidor web **Apache** en la Raspberry Pi con soporte para PHP.

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get dist-upgrade
sudo reboot
sudo apt-get install apache2
sudo apt-get install php php-mbstring
```

El directorio donde se guardan las páginas web por defecto es **/var/www**.

- Ahora añadimos el usuario **www-data** como **sudoer** para que no tenga problemas para interactuar con GPIO.S

Escribimos:

**\$ sudo visudo**

y añadimos una nueva línea en el archivo **/etc/sudoers**:

```
See sudoers(5) for more information on "#include" directives:
#includedir /etc/sudoers.d
pi ALL=(ALL) NOPASSWD: ALL
www-data ALL=(ALL) NOPASSWD: ALL
```

- Contenido del archivo **enciende.py**

```
#!/usr/bin/python
-*- coding: utf-8 -*-
enciende.py
import RPi.GPIO as GPIO ## Import GPIO library
El LED estará en el pin GPIO 2 o, físicamente en el pin 3
GPIO_LED = 3
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(GPIO_LED, GPIO.OUT)
Encendemos el LED
GPIO.output(GPIO_LED, GPIO.HIGH)
```

- Contenido del archivo **apaga.py**

```

#!/usr/bin/python
-*- coding: utf-8 -*-
apaga.py
import RPi.GPIO as GPIO ## Import GPIO library
El LED estará en el pin GPIO 2 o, físicamente en el pin 3
GPIO_LED = 3
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(GPIO_LED, GPIO.OUT)
Apagamos el LED
GPIO.output(GPIO_LED,0)

```

- Por último, programamos el archivo **index.php** que nos va a permitir encender y apagar el LED:

```

1 <html>
2 <head>
3 <meta name="tipo_contenido" content="text/html;" http-equiv="content-type" charset="utf-8">
4 <title>CIFP número UNO</title> <!--index.php-->
5 </head>
6 <body>
7
8
9
10 <!--GPIO2-->
11 <form action="index.php" method="post">
12 <!--<form action="" method="get"-->
13 <H1><U>CIFP número UNO</U></H1>
14 <p></p>
15
16 <HR WIDTH=20% ALIGN=LEFT>
17
18
GPIO 02 <input type="submit" name="encender03" value="Encender">
19 <input type="submit" name="apagar03" value="Apagar">
20

21
22 <HR WIDTH=20% ALIGN=LEFT>
23
24 <!-- Añadimos una entrada de texto -->
25 Nombre

26 <INPUT type="TEXT" name="nombre" autofocus>

27 <INPUT TYPE="SUBMIT" value="mandar">
28
29 </form>
30 </body>
31 </html>
32
33 <?php
34
35 // Funciones PHP del pin GPIO 02
36 $v_nombre = $_POST["nombre"]; // Creamos la variable v_nombre, asociada a "nombre". Este último es
37 // el name del type del código html --> <INPUT type="TEXT" name="nombre" autofocus>

38 echo "Tu nombre es: $v_nombre";
39
40 if ($v_nombre == "luis") {
41 $a= exec("sudo python /var/www/enciende.py");
42 echo $a;
43 }
44
45
46 if ($_POST[encender03]) {
47 $a= exec("sudo python /var/www/enciende.py");
48 echo $a;
49 }
50
51 if ($_POST[apagar03]) {
52 $a= exec("sudo python /var/www/apaga.py");
53 echo $a;
54 }
55
56 // Fin de las funciones PHP del pin GPIO 02
57 ?>

```

La entidad **&nbsp;** (del inglés **Non Breaking Space** que significa espacio sin ruptura) sirve para representar en HTML un espacio en blanco y se utiliza normalmente en dos casos:

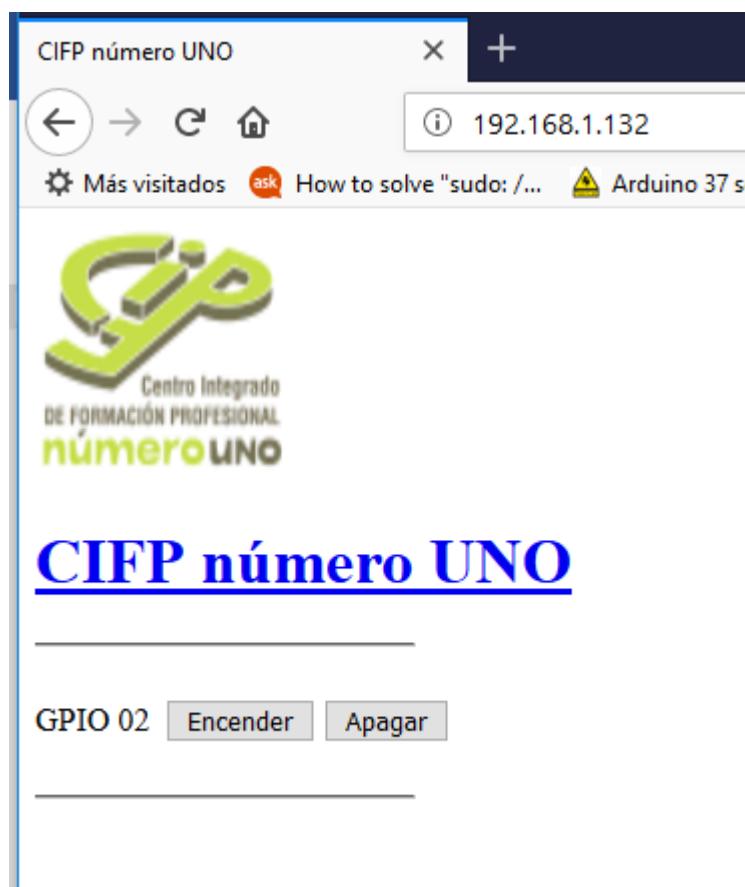
- Cuando queremos que dos palabras no se separen al ajustar el tamaño de la ventana, por ejemplo, las dos palabras que forman una marca. En ese caso si ponemos Coca Cola el navegador puede separar las palabras pero si ponemos Coca&nbsp;Cola el navegador no las separará aunque modifiquemos el tamaño de la ventana.
- Cuando queremos evitar que el navegador agrupe todos los espacios como si fuesen uno sólo podemos utilizar la entidad **&nbsp;** tantas veces como espacios en blanco queramos incluir.

| MÉTODO | CONCEPTO                                                                                                                                                                         | OBSERVACIONES                                                                                                                                                                                                                                                                                  |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GET    | GET lleva los datos de forma "visible" al cliente (navegador web). El medio de envío es la URL. Los datos los puede ver cualquiera.                                              | Los datos son visibles por la URL, por ejemplo:<br>www.aprenderaprogramar.com/<br>action.php?nombre=pedro&apellidos1=gomez                                                                                                                                                                     |
| POST   | POST consiste en datos "ocultos" (porque el cliente no los ve) enviados por un formulario cuyo método de envío es post. Es adecuado para formularios. Los datos no son visibles. | La ventaja de usar <b>POST</b> es que estos datos no son visibles al usuario de la web. En el caso de usar <b>GET</b> , el propio usuario podría modificar la URL escribiendo diferentes parámetros a los reales en su navegador, dando lugar a que la información tratada no sea la prevista. |

**Nota.-**

- El archivo "index.php" y "logo\_100x64.gif", los guardamos en /var/www/html.
- El archivo "index.html", que se encuentra en /var/www/html, lo renombramos como "index\_old.html".
- Los archivos "apaga.py" y "enciende.py", los guardamos en /var/www.

Abrimos el browser y escribimos la url "**localhost**":



### 8.3 Envío de e-mail

Para poder enviar e-mail desde nuestro servidor (u ordenador local), en primer lugar, es necesario contar con un **MTA** (Mail Transport Agent o Agente de transporte de correo). Uno de los MTA más populares para sistemas UNIX-Like, es sin dudas, el famoso **sendmail**.

Para dejar nuestro servidor u ordenador local, listo para enviar mensajes de correo electrónico a través de Internet, solo será necesario entonces, instalar sendmail:

```
$ sudo apt-get install sendmail
```

Para enviar e-mails desde Python, éste nos provee **smtplib**, otro módulo de la librería estándar de Python, quien nos permitirá enviar mensajes de correo electrónico, incluso, en formato HTML.

Solo necesitaremos:

- Crear un objeto **smtplib.SMTP** el cuál recibirá como parámetro de su método constructor, el **host (localhost)**.
- Crear un mensaje de correo
- Enviar el mensaje mediante una llamada al método **sendmail** del **objeto SMTP**.

#### PROCESO:

1. Importamos el módulo **smtplib**:

```
import smtplib
```

2. Luego, definimos las variables necesarias para el envío del mensaje (remitente, destinatario, asunto y mensaje -en formato HTML-):

```
remitente = "Desde Santander <*****@gmail.com>"
destinatario = "Mi prima la del pueblo <*****@gmail.com>"
asunto = "E-mail HTML enviado desde Python"
mensaje = """Hola!

Este es un e-mail enviando desde Python
.....
```

3. A continuación, generamos el e-mail con todos los datos definidos anteriormente:

```
email = """From: %s
To: %s
MIME-Version: 1.0
Content-type: text/html
Subject: %s

%s
""" % (remitente, destinatario, asunto, mensaje)
```

- 4) Y finalmente, creamos un objeto **smtp** y realizamos el envío:

```
smtp = smtplib.SMTP('localhost')
smtp.sendmail(remitente, destinatario, email)
```

**Ejemplo\_1:**

```
-*- coding: utf-8 -*-
import smtplib

remitente = "Desde Santander <*****@gmail.com>"
destinatario = "Mi prima la del pueblo <*****@gmail.com>"
asunto = "E-mail HTML enviado desde Python"
mensaje = """Hola!

Este es un e-mail enviando desde Python
"""
email = """From: %s
To: %s
MIME-Version: 1.0
Content-type: text/html
Subject: %s
%s
""" % (remitente, destinatario, asunto, mensaje)
try:
 smtp = smtplib.SMTP('localhost')
 smtp.sendmail(remitente, destinatario, email)
 print "Correo enviado"
except:
 print """Error: el mensaje no pudo enviarse.
Compruebe que sendmail se encuentra instalado en su sistema"""


```

**Envío de e-mails a múltiples destinatarios**

Para enviar un e-mail a múltiples destinatarios, solo será necesario **generar una lista con los destinatarios**:

```
destinatarios = ['Persona A <maildepersonaA>', 'Persona B <maildepersonaB>']
```

**Agregar una dirección de respuesta diferente**

Cuando generamos el e-mail, es necesario saber, que todo tipo de cabeceras válidas, pueden agregarse. Incluso Reply-To:

```
email = """From: %s
To: %s
Reply-To: noreply@algundominio.com
MIME-Version: 1.0
Content-type: text/html
Subject: %s
%s
""" % (remitente, destinatario, asunto, mensaje)
```

EL MEJOR

**Ejemplo\_2:**

```
#!/usr/bin/python
-*- coding: utf-8 -*-

Enviar correo Gmail con Python
www.pythondiario.com

import smtplib

fromaddr = 'tucorreo@gmail.com'
toaddrs = 'destino@gmail.com'
msg = 'Correo enviado utilizando Python + smtplib en www.pythondiario.com'

Datos
username = 'tucorreo@gmail.com'
password = 'contraseña'

Enviando el correo
server = smtplib.SMTP('smtp.gmail.com:587')
server.starttls()
server.login(username,password)
server.sendmail(fromaddr, toaddrs, msg)
server.quit()
```

## Ejemplo 9.- ENVIAR/RECIBIR SMS CON MODEM USB Y SIM empleando comandos AT

1. <https://myraspberrypiandme.wordpress.com/2013/09/13/short-message-texting-sms-with-huawei-e220/>
2. <https://www.todavianose.com/enviar-sms-con-python-y-la-libreria-pyserial/>
3. <http://stackoverflow.com/questions/2161197/how-to-send-receive-sms-using-at-commands>

### CONEXIÓN DEL MODEM

Tras introducir el SIM en el modem, lo conectamos a la Raspberry y tecleamos:

```
$ ls /dev/tty*
```

```
pi@rpi_64gb:~ $ ls /dev/tty*
/dev/ttys /dev/tty19 /dev/tty3 /dev/tty40 /dev/tty51 /dev/tty62
/dev/tty0 /dev/tty2 /dev/tty30 /dev/tty41 /dev/tty52 /dev/tty63
/dev/tty1 /dev/tty20 /dev/tty31 /dev/tty42 /dev/tty53 /dev/tty7
/dev/tty10 /dev/tty21 /dev/tty32 /dev/tty43 /dev/tty54 /dev/tty8
/dev/tty11 /dev/tty22 /dev/tty33 /dev/tty44 /dev/tty55 /dev/tty9
/dev/tty12 /dev/tty23 /dev/tty34 /dev/tty45 /dev/tty56 /dev/ttyAMA0
/dev/tty13 /dev/tty24 /dev/tty35 /dev/tty46 /dev/tty57 /dev/ttyprintk
/dev/tty14 /dev/tty25 /dev/tty36 /dev/tty47 /dev/tty58 /dev/ttyS0
/dev/tty15 /dev/tty26 /dev/tty37 /dev/tty48 /dev/tty59 /dev/ttyUSBO
/dev/tty16 /dev/tty27 /dev/tty38 /dev/tty49 /dev/tty6 /dev/ttyUSB1
/dev/tty17 /dev/tty28 /dev/tty39 /dev/tty5 /dev/tty60 /dev/ttyUSB2
/dev/tty18 /dev/tty29 /dev/tty4 /dev/tty50 /dev/tty61
pi@rpi_64gb:~ $
```

Se nos abren tres tty: USB0, USB1 y USB2. Hay que probar cuál de ellos nos responde a los comandos AT. En nuestro caso es ttyUSBO.

```
$ sudo lsusb
```

```
pi@rpi_64gb:~ $ sudo lsusb
Bus 001 Device 006: ID 12d1:14cf Huawei Technologies Co., Ltd.
Bus 001 Device 004: ID 154b:005b PNY
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp. SMSC9512/9514 Fast
Ethernet Adapter
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
pi@rpi_64gb:~ $
```

### MONITOR SERIAL

Podemos trabajar con el **minicom**, utilizado anteriormente, pero vamos a instalar **picocom**.

```
$ sudo apt-get install picocom
```

Guía de **picocom** : <http://manpages.ubuntu.com/manpages/precise/man8/picocom.8.html>

## CONFIGURACIÓN

Abrimos **picocom**:

```
$ picocom /dev/ttyUSB0 -b 115200 -l
```

```
pi@rpi_64gb: ~ $ picocom /dev/ttyUSB0 -b 115200 -l
picocom v1.7

port is : /dev/ttyUSB0
flowcontrol : none
baudrate is : 115200
parity is : none
databits are : 8
escape is : C-a
local echo is: no
noinit is : no
noreset is : no
nolock is : yes
send_cmd is : sz -vv
receive_cmd is: rz -vv
imap is :
omap is :
emap is : crcrlf,delbs,
Terminal ready
```

.- Si escribimos el comando **AT**, el modem debe de respondernos con **OK**

(Nota.- Para salir tecleamos **Ctrl+a+x**)

## .- MODOS DE CONFIGURACIÓN

Hay 4 modos que se pueden seleccionar para el funcionamiento:

- Sólo GPRS / EDGE: 13,1,3FFFFFF, 2,4
- Only 3G: 14,2,3FFFFFF, 2,4
- Preferido GPRS / EDGE: 2,1,3FFFFFF, 2,4
- Preferido 3G: 2,2,3FFFFFF, 2,4

Activamos el modo que necesitamos: "Sólo GPRS / EGDE" en este caso.

Dentro de **picocom** tecleamos:

```
AT ^ SYSCFG = 13,1,3FFFFFF, 2,4
```

## .- EL COMANDO PARA EL MODO SMS ES "CMGF":

**AT + CMGF?** Muestra el modo actual

**AT + CMGF = 1** establece el modo de texto

**AT + CMGF = 0** establece el modo PDU

Establecemos el módem en modo de texto primero. Debe responder con "OK".

## ENVIAR

```
AT+CMGS="phonenumber"<CR>the message to send<Ctrl+z>
```

## RECIBIR

A continuación, utilizamos el comando "CMGL" para leer todos los mensajes. Este comando toma un parámetro que se utiliza para configurar si queremos obtener todos los mensajes, sólo mensajes nuevos, etc. Aquí hay una lista:

- **ALL.** Listar todos los mensajes
- **REC UNREAD.** Lista mensajes "recibidos no leídos"
- **REC LEER.** Lista de mensajes "recibidos y ya leídos"
- **STO NO SENTADO.** Los mensajes que se almacenan, pero no se envían
- **STO ENVIADO.** Mensajes guardados y enviados

## ENVIAR Y RECIBIR EN PYTHON

```

envio_sms.py - /home...s/envio_sms.py (2.7.9) - □ ×
File Edit Format Run Options Windows Help

#!/usr/bin/env python

import serial
para enviar el Ctrl-Z
from curses import ascii
#import time

sSerie = serial.Serial('/dev/ttyUSB0', 115200)
sSerie.close()
sSerie.open()
try:
 # Enviamos un reset al modem
 #sSerie.write("ATZ")
 sSerie.write("AT^SYSCFG=13,1,3FFFFFFF,2,4\r\n")
 print sSerie.readline()
 # Le ponemos en modo text para SMS
 sSerie.write("AT+CMGF=1\r\n")
 print sSerie.readline()
 # Le pasamos el numero al que vamos ha mandar el SMS
 sSerie.write("AT+CMGS=\"645171765\"\r\n");

 #time.sleep(0.5)
 # Texto del mensaje terminado en Ctrl+Z
 sSerie.write("a ver si te llega" + ascii.ctrl('z'))

 # Leemos la informacion devuelta
 print sSerie.readline()
 # Leemos la informacion devuelta
 print sSerie.readline()
 # Leemos la informacion devuelta
 print sSerie.readline()

except ValueError:
 print "Oops! se ha producido un error ..."

finally:
 sSerie.close()sSerie.close()

```

.- Queda realizar un estudio exhaustivo sobre los print...

```

recibo_sms_1.py - /hom.../recibo_sms_1.py (2.7.9) - □ ×
File Edit Format Run Options Windows Help

#!/usr/bin/env python

import serial
para enviar el Ctrl-Z
from curses import ascii
import time

sSerie = serial.Serial('/dev/ttyUSB0', 115200, timeout=1)
sSerie.close()
sSerie.open()
try:
 # Enviamos un reset al modem
 sSerie.write("ATZ\r\n")
 # Le ponemos en modo text para SMS
 sSerie.write("AT+CMGF=1\r\n")
 # Leemos todos los sms
 sSerie.write('AT+CMGL="ALL"\r\n')
 #sSerie.write('AT+CMGL="REC UNREAD"\r\n')
 pepe=sSerie.readall()
 print pepe

except ValueError:
 print "Oops! se ha producido un error ..."

finally:
 sSerie.close()

```

.- Lo recibido lo podemos meter en un fichero y así analizarlo para tomar decisiones de control.

## Ejemplo 10.- PROGRAMAR TAREAS EN LINUX USANDO CRONTAB



En Linux existe una herramienta muy útil, llamada **Cron**, que sirve para programar y automatizar tareas. Si queremos, por ejemplo, que cada hora se ejecute un script, o un archivo PHP, o cualquier otra cosa, pues no vamos a meternos 24 veces al día para ejecutarlo manualmente. No tiene sentido. Para esto está **Cron**.

El comando **crontab** se utiliza en sistemas **UNIX** para programar la ejecución de otros comandos, es decir, para automatizar tareas. Podemos ver los **crontabs** que están programados y también editarlos. Para verlos, utilizamos este comando:

```
sudo crontab -l
```

Para editarlos:

```
sudo crontab -e
```

### FORMATO DE LAS TAREAS

Las tareas *cron* siguen una determinada sintaxis. Tienen 5 asteriscos seguidos del comando a ejecutar.

```
* * * * * /bin/ejecutar/script.sh
```

### LOS 5 ASTERISCOS

De izquierda a derecha, los asteriscos representan:

1. Minutos: de 0 a 59.
2. Horas: de 0 a 23.
3. Día del mes: de 1 a 31.
4. Mes: de 1 a 12.
5. Día de la semana: de 0 a 6, siendo 0 el domingo.

Si se deja un asterisco, quiere decir "cada" minuto, hora, día de mes, mes o día de la semana. Por ejemplo:

```
* * * * * /bin/ejecutar/script.sh
```

Ejecuta este script:

- Cada minuto
- De cada hora
- De cada día del mes
- De cada mes
- De cada día de la semana

Otro ejemplo:

```
30 2 * * 1 /bin/ejecutar/script.sh
```

Ejecutar este script:

- En el minuto 30
- De las 2 de la noche
- De cada día del mes
- De cada mes
- Sólo si es lunes

En resumen, todos los viernes a las 2:30 horas se ejecutará el script.

### Intervalos de tiempo

Ejecutar un script de lunes a viernes a las 2:30 horas:

```
30 2 * * 1-5 /bin/ejecutar/script.sh
```

Ejecutar un script de lunes a viernes cada 10 minutos desde las 2:00 horas durante una hora:

```
0,10,20,30,40,50 2 * * 1-5 /bin/ejecutar/script.sh
```

Esto quizá puede ser largo. La sintaxis de **crontab** permite lo siguiente. Imaginemos que queremos ejecutarlo cada 5 minutos:

```
*/5 2 * * 1-5 /bin/ejecutar/script.sh
```

### PALABRAS RESERVADAS

Muchas veces tenemos palabras reservadas para facilitar el uso de programas o lenguajes de programación. **Cron no podía ser menos**, así que tenemos algunas que suelen ser las más comunes. Ya cada uno que lo configure conforme a sus necesidades. Aquí van:

- @reboot: se ejecuta una única vez al inicio.
- @yearly/@annually: ejecutar cada año.
- @monthly: ejecutar una vez al mes.
- @weekly: una vez a la semana.
- @daily/@midnight: una vez al día.
- @hourly: cada hora.

Por ejemplo, **para ejecutar el script cada hora**:

```
@hourly /bin/ejecutar/script.sh
```

### Programa.- CAPTURAR (con hora + fecha) IMAGENES EN UNA RASPBERRY PI CON PYTHON

1. Ejecutamos → **\$ sudo crontab -e**  
y escribimos → **0 16 \* \* \* python /home/programas/camara/captura.py**
2. En la ruta anterior creamos el programa **captura.py** con el siguiente contenido:

```
#!/usr/bin/env python
-*- coding: utf-8 -*-
import os
from time import sleep
import time
import datetime
from picamera import PiCamera
environnement vars
os.environ.setdefault('XAUTHORITY', '/home/user/.Xauthority')
os.environ.setdefault('DISPLAY', ':0.0')
Obtener Fecha
fecha = datetime.date.today()
fecha = str(fecha)
Obtener Hora
hora = time.strftime("%H:%M:%S")
Capturo imagen
camera = PiCamera()
camera.resolution = (1024, 768)
camera.start_preview()
Camera warm-up time
sleep(2)
camera.capture("home/pi" + fecha + "-" + hora + ".jpg")
camera.close()
```

## Ejemplo 11.- INVERNADERO

Vamos a centrarnos en los sensores **DHT11** o **DHT22**. Elegimos este modelo porque con un único sensor podemos medir temperatura y humedad.

Una particularidad de estos sensores es que la señal de salida es digital, por lo tanto, lo tendremos que conectar a pines digitales.

Llevan un pequeño microcontrolador interno para hacer el tratamiento de señal.



### ¿Cómo funcionan?

Los **DHT11** y **DHT22** se componen de un sensor capacitivo para medir la humedad y de un termistor.

La principal diferencia entre ambos es que el ciclo de operación es menor en el DHT11 que en el DHT22, sin embargo, el DHT22 tiene rangos de medida más amplios y mayor resolución, a cambio de resultar algo más caro.

Ambos sensores están calibrados en laboratorio y tienen una buena fiabilidad.

### ENCAPSULADO.

Existen en el mercado tres variantes:

- El sensor suelto, con un encapsulado azul y cuatro pines disponibles para conectar. (Será necesario añadir la resistencia pull-up)
- El sensor con una placa soldada, con tres pines disponibles para conectar y una resistencia pull-up (normalmente de 4,7-10 kΩ).
- El mismo formato que el anterior, pero con un condensador de filtrado (normalmente de 100 nF).



### CARACTERÍSTICAS DE CADA UNO DE LOS SENSORES:

| Parámetro                   | DHT11             | DHT22               |
|-----------------------------|-------------------|---------------------|
| Alimentación                | 3Vdc ≤ Vcc ≤ 5Vdc | 3.3Vdc ≤ Vcc ≤ 6Vdc |
| Señal de Salida             | Digital           | Digital             |
| Rango de medida Temperatura | De 0 a 50 °C      | De -40°C a 80 °C    |
| Precisión Temperatura       | ±2 °C             | <±0.5 °C            |
| Resolución Temperatura      | 0.1°C             | 0.1°C               |
| Rango de medida Humedad     | De 20% a 90% RH   | De 0 a 100% RH      |
| Precisión Humedad           | 4% RH             | 2% RH               |
| Resolución Humedad          | 1%RH              | 0.1%RH              |
| Tiempo de respuesta         | 1s                | 2s                  |
| Tamaño                      | 12 x 15.5 x 5.5mm | 14 x 18 x 5.5mm     |

**CICLO DE OPERACIÓN.**

Es el tiempo que el sensor tarda en ofrecer una respuesta desde que se le pide.

Formato de datos de un solo bus para la comunicación y sincronización entre MCU y el sensor. El proceso de comunicación es de 4 ms aproximadamente.

Una transmisión de datos completa es de 40 bits. Donde obtenemos la temperatura y la humedad.

Ejemplo: Recibimos 40 bits:

- **0011 0101 0000 0000 0001 1000 0000 0000 0100 1101**  
High humidity 8 + Low humidity 8 + High temp. 8 + Low temp. = 8 Parity bit

Calculando :

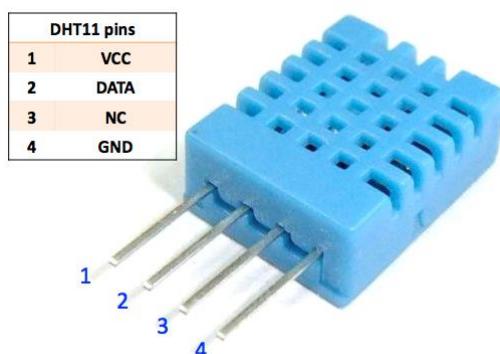
- **0011 0101+0000 0000+0001 1000+0000 0000= 0100 1101**

Datos correctos recibidos :

- Humedad : 0011 0101 = 35H = **53%RH**
- Temperatura : 0001 1000 = 18H = **24°C**

El microcontrolador externo y el microcontrolador que lleva integrado el sensor, se hablan entre sí de la siguiente manera:

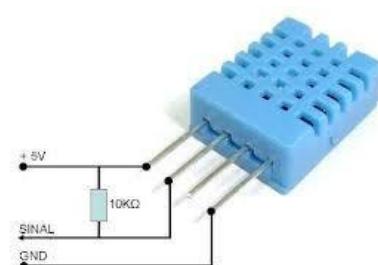
- Se inicia la comunicación.
- El sensor responde estableciendo un nivel bajo de 80us y un nivel alto de 80us.
- El sensor envía 5 bytes con la información de temperatura y humedad.

**PINOUT.**

Los pines del DHT11 y del DHT22 siguen el mismo orden.

**CONEXIÓN DE UN DHT11 SUELTO.**

destacamos que no utilizamos el pin 3, y que el resistor tiene que conectarse entre la alimentación y el pin de salida de datos.



## PROGRAMACIÓN

### INSTALACIÓN DE LIBRERÍAS:

#### 1. ARROW

```
$ sudo pip install arrow
```

#### 2. ADAFRUIT (Sólo funciona con Python 2)

- Clonamos el repositorio git de la librería de Adafruit que nos permitirá obtener las lecturas del sensor DHT11 →

```
$ git clone https://github.com/adafruit/Adafruit_Python_DHT.git
```

- Instalar software que nos permite adicionar librerías Python a nuestra Raspberry Pi

```
$ sudo apt-get install build-essential python-dev
```

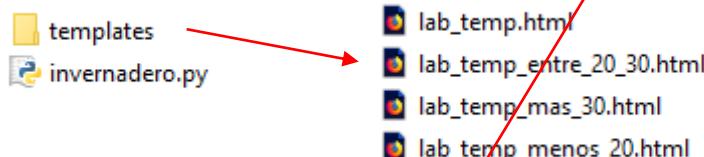
- Entrar a la carpeta que se creó al clonar con git el repositorio

```
$ cd Adafruit_Python_DHT
```

- Instalar la librería DHT de Adafruit

```
$ sudo python setup.py install
```

La temperatura y humedad podrá verse en la web utilizando “render\_template”, por ello, vamos a necesitar una carpeta llamada **templates**



```

1 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3 import os
4 #os.system('fuser -k 8085/tcp')
5 """
6 =====
7 CONTROL INVERNADERO
8
9 .- Cuando la t° es menor de 20°C alertamos "Cerrar Puerta y Ventana"
10 .- Cuando la t° está entre 20°C y 30°C alertamos "Abrir Puerta"
11 .- Cuando la t° es mayor de 30°C alertamos "Abrir Puerta y Ventana"
12
13 Se mandará un email cada vez que se traspase el umbral de 20°C o 30°C
14
15 Entre las 21:00h y las 9:00h, no se mandará ningún email
16 =====
17 """
18 from flask import Flask, request, render_template
19 import time
20 import datetime
21 #import arrow
22 import sys
23 import Adafruit_DHT
24
25 app = Flask(__name__)
26 app.debug = True
27
28 flag_menos_20 = False
29 flag_mas_30 = False
30 flag_entre_2030 = False
31 flag_inicio = True #Cuando se reinicia, o comienza la jornada de 9 a 21 horas, si la t° es superior a 20°C
32 #nos vamos a la rutina de "entre 20°C y 30°C" o "más de 30°C"...
33 #y, si es así, tengo que mandar un email para que abra puerta o puerta y ventana
34
35 import smtplib
36
37 @app.route("/")
38 def lab_temp():
39
40 if __name__ == "__main__":
41 app.run(host='0.0.0.0', port=8085)
42

```

Con estas instrucciones llamamos a la función “lab\_temp” y configuramos el puerto para la web

.- Entre las 21:00h y las 9:00h no mandamos emails, sólo enseñamos los datos en la web

```

@app.route("/")
def lab_temp():
 global flag_menos_20
 global flag_mas_30
 global flag_entre_2030
 global flag_inicio

 x=datetime.datetime.now()
 if x.hour > 8 and x.hour < 21:
 pass
 else:
 flag_menos_20 = False
 flag_mas_30 = False
 flag_entre_2030 = False
 flag_inicio = True
 humidity, temperature = Adafruit_DHT.read_retry(Adafruit_DHT.AM2302, 4)
 return render_template("lab_temp.html",temp=temperature,hum=humidity)

```

GPIO4

.- Hemos dividido en 3 rangos el control del invernadero:

- <20°C
- >20°C - <30°C
- >30°C

Entre las 9:00h y las 21:00h nos preguntamos en que rango nos encontramos:

**<20°C**

```

if x.hour > 9 and x.hour < 21:
 #humidity, temperature = Adafruit_DHT.read_retry(Adafruit_DHT.AM2302, 4)
 humidity, temperature = Adafruit_DHT.read_retry(22, 4)
 if temperature < 20:
 if flag_entre_2030 == False:
 flag_menos_20 = True
 return render_template("lab_temp_menos_20.html",temp=temperature,hum=humidity)

 else:
 flag_entre_2030 = False
 fromaddr = 'tucorreo@gmail.com'
 toaddrs = 'jlval@cifpnl.es'
 msg = '''Hola J.Luis, la t° del invernadero es menor de 20°C.
 Cerrar Puerta y Ventana'''

 # Datos
 username = 'tucorreo@gmail.com'
 password = '*****'

 # Enviando el correo
 server = smtplib.SMTP('smtp.gmail.com:587')
 server.starttls()
 server.login(username,password)
 server.sendmail(fromaddr, toaddrs, msg)
 server.quit()
 return render_template("lab_temp_menos_20.html",temp=temperature,hum=humidity)

 elif temperature > 30: ← Si es más de 30
 else: ← Si está entre 20 y 30

```

**Podemos utilizar cualquiera de estas dos líneas**

&gt;30°C

```
x=datetime.datetime.now()
if x.hour > 8 and x.hour < 21:
 humidity, temperature = Adafruit_DHT.read_retry(Adafruit_DHT.AM2302, 4)
 humidity, temperature = Adafruit_DHT.read_retry(22, 4)
 if temperature < 20: #t° menor de 20°C

 elif temperature > 30: #t° mayor de 30°C
 if flag_inicio == True: #preguntamos si es un reinicio o las 9 de la mañana
 flag_inicio = False
 flag_entre_2030 = False
 fromaddr = 'tucorreo@gmail.com'
 toaddrs = 'jlval@cifpnl.es'
 msg = '''Hola J.Luis, la t° del invernadero es mayor de 30°C.
 Abrir Puerta y Ventana'''

 # Datos
 username = 'tucorreo@gmail.com'
 password = '*****'

 # Enviando el correo
 server = smtplib.SMTP('smtp.gmail.com:587')
 server.starttls()
 server.login(username,password)
 server.sendmail(fromaddr, toaddrs, msg)
 server.quit()
 return render_template("lab_temp_mas_30.html",temp=temperature,hum=humidity)

 else:
 if flag_entre_2030 == False:
 flag_mas_30 = True
 return render_template("lab_temp_mas_30.html",temp=temperature,hum=humidity)

 else:
 flag_entre_2030 = False
 fromaddr = 'tucorreo@gmail.com'
 toaddrs = 'jlval@cifpnl.es'
 msg = '''Hola J.Luis, la t° del invernadero es mayor de 30°C.
 Abrir Puerta y Ventana'''

 # Datos
 username = 'tucorreo@gmail.com'
 password = '*****'

 # Enviando el correo
 server = smtplib.SMTP('smtp.gmail.com:587')
 server.starttls()
 server.login(username,password)
 server.sendmail(fromaddr, toaddrs, msg)
 server.quit()
 return render_template("lab_temp_mas_30.html",temp=temperature,hum=humidity)
```

&gt;20°C - &lt;30°C

```
humidity, temperature = Adafruit_DHT.read_retry(22, 4)
if temperature < 20: #t° menor de 20°C

 elif temperature > 30: #t° mayor de 30°C
 else: #t° entre 20°C y 30°C
 if flag_inicio == True: #preguntamos si es un reinicio o las 9 de la mañana
 flag_inicio = False
 flag_menos_20 = False
 flag_mas_30 = False
 fromaddr = 'tucorreo@gmail.com'
 toaddrs = 'jlval@cifpnl.es'
 msg = '''Hola J.Luis, la t° del invernadero está entre 20°C y 30°C
 Abrir Puerta'''

 # Datos
 username = 'tucorreo@gmail.com'
 password = '*****'

 # Enviando el correo
 server = smtplib.SMTP('smtp.gmail.com:587')
 server.starttls()
 server.login(username,password)
 server.sendmail(fromaddr, toaddrs, msg)
 server.quit()
 return render_template("lab_temp_entre_20_30.html",temp=temperature,hum=humidity)

 else:
 if flag_menos_20 == False and flag_mas_30 == False:
 flag_entre_2030 = True
 return render_template("lab_temp_entre_20_30.html",temp=temperature,hum=humidity)

 else:
 flag_menos_20 = False
 flag_mas_30 = False
 fromaddr = 'tucorreo@gmail.com'
 toaddrs = 'jlval@cifpnl.es'
 msg = '''Hola J.Luis, la t° del invernadero está entre 20°C y 30°C
 Abrir Puerta'''

 # Datos
 username = 'tucorreo@gmail.com'
 password = '*****'

 # Enviando el correo
 server = smtplib.SMTP('smtp.gmail.com:587')
 server.starttls()
 server.login(username,password)
 server.sendmail(fromaddr, toaddrs, msg)
 server.quit()
 return render_template("lab_temp_entre_20_30.html",temp=temperature,hum=humidity)
```

[lab\\_temp.html](#)

```

<html>
 <meta charset="utf-8">
 <title>Temperatura Invernadero Alejos</title>
 <meta http-equiv="refresh" content="1">
 <meta name="description" content="Temperatura Invernadero Alejos">
 <meta name="author" content="J. Luis Pérez">
 <!-- Mobile Specific Metas
 -->
 <meta name="viewport" content="width=device-width, initial-scale=1">
 <!-- FONT
 -->
 <link href="//fonts.googleapis.com/css?family=Raleway:400,300,600" rel="stylesheet" type="text/css">
 <!-- CSS
 -->
 <link rel="stylesheet" href="/static/css/normalize.css">
 <link rel="stylesheet" href="/static/css/skeleton.css">
 <!-- Favicon
 -->
 <link rel="icon" type="image/png" href="/static/images/logo_100x64.gif">
</head>
<body>
 <div class="container">
 <div class="row">
 <div class="two-third column" style="margin-top: 5%">
 <h2>Datos en tiempo real Invernadero de Alejos</h2>
 <p>=====
 <h1>Temperatura: {{"0:0.1f".format(temp)}}°C</h1>
 <h1>Humedad: {{"0:0.1f".format(hum)}}%</h1>
 <p>=====
 <h1>ABRIR PUERTA</h1>
 <p>=====
 <p>-----Esta página se refresca cada 1s-----</p>
 </div>
 </div>
 <div class="row">
 <div class="eleven columns">
 <form id="range_select" action = "/lab_env_db" method="GET">
 <input type="hidden" class="timezone" name="timezone" />
 <div class="one column">
 </div>
 </form>
 </div>
 </div>
</body>
</html>

```

- Hacer que se lance al iniciar la Raspberry:

1. Añadiremos en el fichero \$ /etc/rc.local la siguiente línea antes del "exit 0":

python /home/pi/programas/temperatura/invernadero/invernadero.py

2. Creamos un fichero script en /etc/init.d, (por ejemplo, lo llamamos “**inicio**”) con el siguiente contenido:

```
#!/usr/bin/bash
sudo /etc/rc.local
```

3. Ejecutamos el siguiente comando para que lo ejecute al arrancar:

\$sudo update-rc.d inicio -f enable S

-----Esta página se refresca cada 1s-----

## Ejemplo 12.- CÓDIGO QR

Instalamos las siguientes librerías:

```
sudo apt-get install zbar-tools
sudo apt-get install python-zbar
sudo apt-get install libzbar0
```

### CÓDIGO

```
"""
PYTHON 2
Programa en Python para leer códigos QR.
Muestra sus dimensiones en la pantalla, su mensaje y su rotación en grados.

Escrito por Glare,
www.robologs.net
"""

import zbar
import numpy as np
import cv2

#Inicializar la cámara
capture = cv2.VideoCapture(0)

#Cargar la fuente
font = cv2.FONT_HERSHEY_SIMPLEX

while 1:
 #Capturar un frame
 val, frame = capture.read()

 #Hay que comprobar que el frame sea válido
 if val:
 #Capturar un frame con la cámara y guardar sus dimensiones
 frame_gris = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
 dimensiones = frame_gris.shape #'dimensiones' será un array que contendrá el alto, el ancho y los canales de la imagen en este orden.

 #Convertir la imagen de OpenCV a una imagen que la librería ZBAR pueda entender
 imagen_zbar = zbar.Image(dimensiones[1], dimensiones[0], 'Y800', frame_gris.tobytes())

 #Construir un objeto de tipo scanner, que permitirá escanear la imagen en busca de códigos QR
 escaner = zbar.ImageScanner()

 #Escanear la imagen y guardar todos los códigos QR que se encuentren
 escaner.scan(imagen_zbar)

 for codigo_qr in imagen_zbar:
 loc = codigo_qr.location #Guardar las coordenadas de las esquinas
 dat = codigo_qr.data[:-2] #Guardar el mensaje del código QR. Los últimos dos caracteres son saltos de línea que hay que eliminar

 #Convertir las coordenadas de las cuatro esquinas a un array de numpy
 #Así, lo podremos pasar como parámetro a la función cv2.polylines para dibujar el contorno del código QR
 localización = np.array(loc, np.int32)

 #Dibujar el contorno del código QR en azul sobre la imagen
 cv2.polylines(frame, [localización], True, (255,0,0), 2)
```

```
#Dibujar el contorno del codigo QR en azul sobre la imagen
cv2.polyline(frame, [localizacion], True, (255,0,0), 2)

#Dibujar las cuatro esquinas del codigo QR
cv2.circle(frame, loc[0], 3, (0,0,255), -1) #Rojo - esquina superior izquierda
cv2.circle(frame, loc[1], 3, (0,255,255), -1) #Amarillo - esquina inferior izquierda
cv2.circle(frame, loc[2], 3, (255,100,255), -1) #Rosa -esquina inferior derecha
cv2.circle(frame, loc[3], 3, (0,255,0), -1) #Verde - esquina superior derecha

#Buscar el centro del rectangulo del codigo QR
cx = (loc[0][0]+loc[2][0])/2
cy = (loc[0][1]+loc[2][1])/2

#Escribir el mensaje del codigo QR.
cv2.putText(frame,dat,(cx,cy), font, 0.7,(255,255,255),2)

#Calcular el angulo de rotacion del codigo QR. Supondremos que el angulo es la pendiente de la recta que une el vertice loc[0] (rojo) con loc[3] (verde)
vector_direktor = [loc[3][0]-loc[0][0], loc[3][1]-loc[0][1]]
angulo = (np.arctan2(float(vector_direktor[1]),vector_direktor[0])*57.29)%360 #Calculo de la tangente y conversion de radianes a grados
#Correccion debida al orden de las coordenadas en la pantalla
angulo += -360
angulo *= -1

#Escribir el angulo sobre la imagen con dos decimales
cv2.putText(frame,str("%.2f" % angulo),(cx,cy+30), font, 0.7,(255,255,255),2)

#Mostrar la imagen
cv2.imshow('Imagen', frame)

#Salir con 'ESC'
k = cv2.waitKey(5) & 0xFF
if k == 27:
 break

cv2.destroyAllWindows()
```



### Ejemplo 13.- Screen SHOT

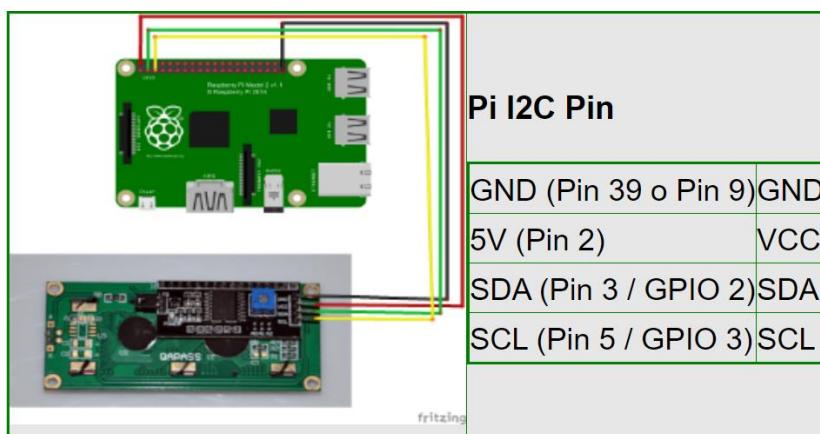
Instalamos las siguientes librerías:

```
$ sudo pip3 install mss
```

```
from mss import mss
with mss() as screen:
 screen.shot(output="screen.png")
```

## Ejemplo 14.- IIC (LCD)

### Paso 1.- Conectar



### Paso 2.- Habilitar I2C

```
$sudo nano /etc/modules
```

Añadimos estas líneas:

```
i2c-bcm2708
```

```
i2c-dev
```

### Paso 3.- Instalar *smbus* y la librería *i2c* de *python*

Escribe el siguiente comando en la terminal:

```
$sudo apt-get update
```

```
$sudo apt-get install -y python-smbus i2c-tools
```

```
$sudo reboot
```

Después de reiniciar el sistema, escribe el siguiente comando para verificar la instalación del software:

```
$lsmod | grep i2c_
```

Deberías ver *i2c\_bcm2708* en una lista, esto significa que la biblioteca se ha instalado correctamente.

```
pi@rpi_64_opencv:~ $ lsmod | grep i2c_
i2c_bcm2835 16384 0
i2c_dev 16384 0
i2c_bcm2708 16384 0
i2c_i2c 16384 0
```

### Paso 4.- Testamos el hardware

```
pi@movidiusbuster64gb:~ $ sudo i2cdetect -y 1
 0 1 2 3 4 5 6 7 8 9 a b c d e f
00: --
10: --
20: --
30: --
40: --
50: --
60: --
70: --
```

The number 27 is circled in red.

Si solo puedes ver el signo "----" en la lista sin ningún número, significa que la conexión de tu circuito es incorrecta o que tu software no está instalado correctamente.

**Ejemplo:**

```
1 #!/usr/bin/python
2 # -*- coding: utf-8 -*-
3 """
4 ///////////////////////////////////////////////////////////////////
5 // AUTOR: http://osoyoo.com/?p=1031 Octubre/2019
6 ///////////////////////////////////////////////////////////////////
7 // PROGRAMA: LCD I2C VERSIÓN: 1.0
8 // DISPOSITIVO: Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC
9 // Versión Python: 3.5.3
10 // TARJETA DE APLICACIÓN: Raspberry Pi 3 B+
11 ///////////////////////////////////////////////////////////////////
12 Explicación del programa
13 ///////////////////////////////////////////////////////////////////
14 """
15 #/////////////////////////////////////////////////////////////////
16 # IMPORTAR LIBRERÍAS E INSTANCIAR CLASES
17 #/////////////////////////////////////////////////////////////////
18 import smbus
19 import time
20
21 #/////////////////////////////////////////////////////////////////
22 # VARIABLES GLOBALES
23 #/////////////////////////////////////////////////////////////////
24 # Define some device parameters
25 I2C_ADDR = 0x27 # I2C device address, if any error, change this address to 0x3f
26 LCD_WIDTH = 16 # Maximum characters per line
27
28 # Define some device constants
29 LCD_CHR = 1 # Mode - Sending data
30 LCD_CMD = 0 # Mode - Sending command
31
32 LCD_LINE_1 = 0x80 # LCD RAM address for the 1st line
33 LCD_LINE_2 = 0xC0 # LCD RAM address for the 2nd line
34 LCD_LINE_3 = 0x94 # LCD RAM address for the 3rd line
35 LCD_LINE_4 = 0xD4 # LCD RAM address for the 4th line
36
37 LCD_BACKLIGHT = 0x08 # On
#LCD_BACKLIGHT = 0x00 # Off
38
39 ENABLE = 0b000000100 # Enable bit
40
41 # Timing constants
42 E_PULSE = 0.0005
43 E_DELAY = 0.0005
44
45 #Open I2C interface
46 #bus = smbus.SMBus(0) # Rev 1 Pi uses 0
47 bus = smbus.SMBus(1) # Rev 2 Pi uses 1
48
49 #/////////////////////////////////////////////////////////////////
50 # FUNCIONES
51 #/////////////////////////////////////////////////////////////////
52 def lcd_init():
53
54 def lcd_byte(bits, mode):
55
56 def lcd_toggle_enable(bits):
57
58 def lcd_string(message,line):
59
60 #/////////////////////////////////////////////////////////////////
61 # CONFIGURACIÓN PUERTOS GPIO Y RECURSOS A UTILIZAR
62 #/////////////////////////////////////////////////////////////////
63
64 #/////////////////////////////////////////////////////////////////
65 # PROGRAMA PRINCIPAL
66 #/////////////////////////////////////////////////////////////////
67
68 def main():
69 # Main program block
70
71 # Initialise display
72 lcd_init()
73
74 while True:
75
76 # Send some test
77 lcd_string("Created by <",LCD_LINE_1)
78 lcd_string("Osoyoo.com <",LCD_LINE_2)
79
80 time.sleep(3)
81
82 # Send some more text
83 lcd_string("> Tutorial Url:",LCD_LINE_1)
84 lcd_string("> http://osoyoo.com",LCD_LINE_2)
85
86 time.sleep(3)
87
88
89 if __name__ == '__main__':
90
91 try:
92 main()
93 except KeyboardInterrupt:
94 pass
95 finally:
96 lcd_byte(0x01, LCD_CMD)
```

```
50 ##### FUNCIONES #####
51 # FUNCIONES
52 #####
53 def lcd_init():
54 # Initialise display
55 lcd_byte(0x33,LCD_CMD) # 110011 Initialise
56 lcd_byte(0x32,LCD_CMD) # 110010 Initialise
57 lcd_byte(0x06,LCD_CMD) # 000110 Cursor move direction
58 lcd_byte(0x0C,LCD_CMD) # 001100 Display On,Cursor Off, Blink Off
59 lcd_byte(0x28,LCD_CMD) # 101000 Data length, number of lines, font size
60 lcd_byte(0x01,LCD_CMD) # 000001 Clear display
61 time.sleep(E_DELAY)
62
63 def lcd_byte(bits, mode):
64 # Send byte to data pins
65 # bits = the data
66 # mode = l for data
67 # 0 for command
68
69 bits_high = mode | (bits & 0xF0) | LCD_BACKLIGHT
70 bits_low = mode | ((bits<<4) & 0xF0) | LCD_BACKLIGHT
71
72 # High bits
73 bus.write_byte(I2C_ADDR, bits_high)
74 lcd_toggle_enable(bits_high)
75
76 # Low bits
77 bus.write_byte(I2C_ADDR, bits_low)
78 lcd_toggle_enable(bits_low)
79
80 def lcd_toggle_enable(bits):
81 # Toggle enable
82 time.sleep(E_DELAY)
83 bus.write_byte(I2C_ADDR, (bits | ENABLE))
84 time.sleep(E_PULSE)
85 bus.write_byte(I2C_ADDR, (bits & ~ENABLE))
86 time.sleep(E_DELAY)
87
88 def lcd_string(message,line):
89 # Send string to display
90
91 message = message.ljust(LCD_WIDTH," ")
92
93 lcd_byte(line, LCD_CMD)
94
95 for i in range(LCD_WIDTH):
96 lcd_byte(ord(message[i]),LCD_CHR)
97
```

## Ejemplo 15.- gtts y mp3

### GTTS

Gtts es una API de google que convierte texto a voz.

1º Instalamos gtts:

```
$ sudo pip3 install gtts
$ sudo apt-get install python-gst-1.0 gstreamer1.0-plugins-good gstreamer1.0-plugins-ugly
gstreamer1.0-tools
```

2º Ejemplo:

```
$ gtts-cli 'hola mundo' --output audio.mp3
```

### MP3

1º Instalamos los paquetes “**mpg123**”, que contiene un reproductor / decodificador de audio en tiempo real MPEG 1.0 / 2.0 / 2.5 para capas 1, 2 y 3 (más comúnmente MPEG 1.0 capa 3, más conocido como MP3). También usaremos “**alsa-utils**” que aunque seguramente estén instaladas, con este comando se instalarán o actualizarán. El paquete ALSA Utilities contiene varias utilidades que son útiles para controlar su tarjeta de sonido:

```
$ sudo apt-get install alsa-utils mpg123
```

2º Ejemplo:

```
$ mpg123 audio.mp3
```

### EJEMPLO con PYTHON

```
#!/usr/bin/env python
-*- coding: utf-8 -*-
from gtts import gTTS
import os

tts = gTTS('hola buenos días')
tts.save('saludo.mp3')

os.system('mpg123 -q saludo.mp3')
```

## Ejemplo 16.- Reconocimiento de voz y conversión a texto (Speech to Text)

El reconocimiento de voz es la capacidad de un software de computadora para identificar palabras y frases en el lenguaje hablado y convertirlas en texto legible para humanos.

Para que funcione el reconocimiento de voz a texto en Python, deberemos instalar estas tres librerías:

- **SpeechRecognition**
- **PyAudio**
- **FLAC**

### . - SpeechRecognition

Se han desarrollado varias bibliotecas de reconocimiento de voz en **Python**, la biblioteca **SpeechRecognition**, es la más simple de todas.

No necesitamos construir ningún modelo de aprendizaje automático desde cero, esta biblioteca admite varios motores de reconocimiento de voz pública como:

- CMU **Sphinx** (sin conexión) → **recognize\_sphinx()**
- Reconocimiento de voz de **Google** → **recognize\_google()**
- API de **Google Cloud Speech** → **recognize\_google\_cloud()**
- Reconocimiento de voz de **Microsoft Bing** → **recognize\_bing()**
- API de **Houndify** → **recognize\_houndify()**
- Discurso a texto de **IBM** → **recognize\_ibm()**

Usaremos **Google Speech Recognition**, ya que no requiere ninguna clave **API**. Por defecto, el reconocedor de Google lee Inglés.

Para instalar la librería, ejecutamos: **\$ sudo pip3 install SpeechRecognition**

### . - PyAudio

La librería **SpeechRecognition** utiliza **PyAudio** para obtener la información del **micrófono**, por lo cual también debe tener instalada esta librería.

Para instalar la librería seguiremos estos pasos:

- **\$ sudo git clone http://people.csail.mit.edu/hubert/git/pyaudio.git**
- **\$ sudo apt-get install libportaudio0 libportaudio2 libportaudiocpp0 portaudio19-dev**
- **\$ cd pyaudio**
- **\$ sudo python setup.py install**

### . - FLAC

Se requiere un **codificador FLAC** para codificar los datos de audio para enviar a la **API**. Si usa Windows (x86 o x86-64), OS X (solo Intel Macs, OS X 10.6 o superior) o Linux (x86 o x86-64), esto ya está incluido en esta biblioteca; no necesita instalar nada.

Para proceder a su instalación ejecutamos: **\$ sudo apt-get install flac**

## CONVERSIÓN DESDE ARCHIVO

Creamos el archivo “archivo\_voz\_texto.py”

```
#import library
import speech_recognition as sr

Initialize recognizer class (for recognizing the speech)
r = sr.Recognizer()

Reading Audio file as source
listening the audio file and store in audio_text variable

with sr.AudioFile('archivo.wav') as source:

 audio_text = r.listen(source)

recognize_() method will throw a request error if the API is unreachable,
hence using exception handling
try:

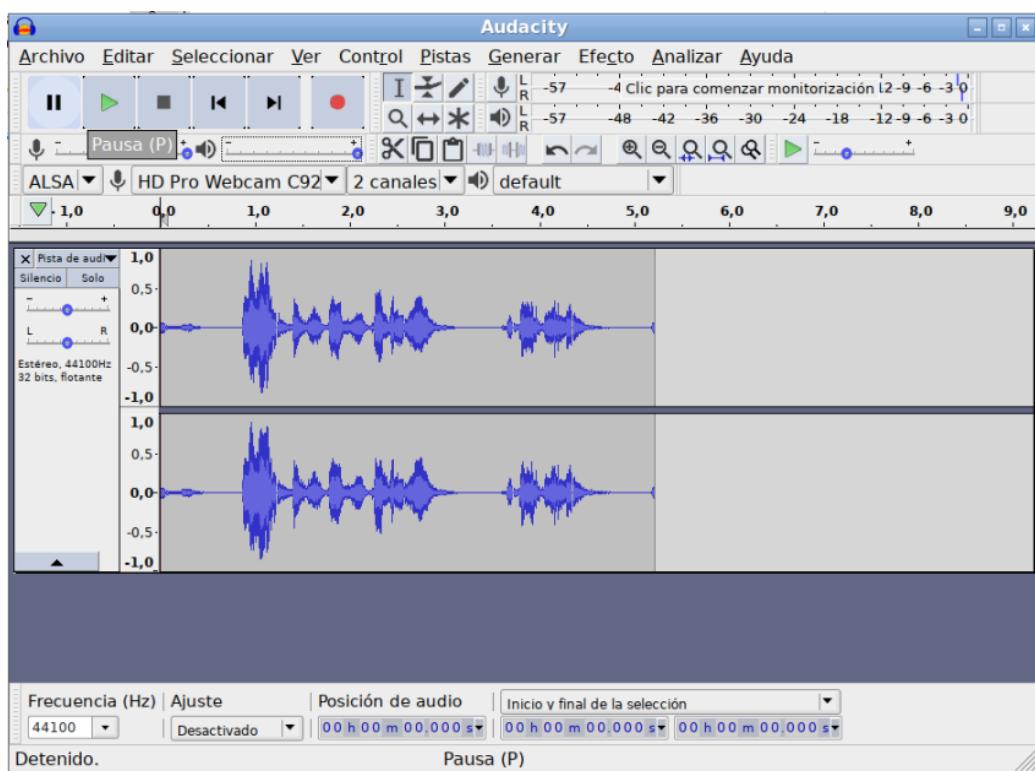
 # using google speech recognition
 text = r.recognize_google(audio_text)
 print('Converting audio transcripts into text ...')
 print(text)

except:
 print('Sorry.. run again...')
```

Desde el **bash** ejecutamos: \$ sudo python3 archivo\_voz\_texto.py

Para crear un archivo **wav** en la Raspberry, podemos hacerlo desde la aplicación **Audacity**.

La instalamos con la siguiente orden: \$ sudo apt-get install audacity



### CONVERSIÓN DESDE MICRÓFONO

En lugar de la fuente del archivo de audio, tenemos que usar la **clase Micrófono**. Los pasos restantes son los mismos.

```
#import library

import speech_recognition as sr

Initialize recognizer class (for recognizing the speech)

r = sr.Recognizer()

Reading Microphone as source
listening the speech and store in audio_text variable

with sr.Microphone() as source:
 print("Talk")
 audio_text = r.listen(source)
 print("Time over, thanks")
recognize_() method will throw a request error if the API is unreachable,
hence using exception handling

try:
 # using google speech recognition
 print("Text: "+r.recognize_google(audio_text, language='es-ES'))
except:
 print("Sorry, I did not get that")
```

- Podemos reconocer diferentes idiomas pasando el parámetro “**language**” a la función **recognize\_google()**
- Si queremos que dure un **tiempo limitado** la escucha del micrófono:
  - Cambiamos la función **r.listen()** por **r.record()**  
Por ejemplo, la siguiente orden escuchará el micrófono durante 5sg.  
**audio\_text = r.record(source, duration = 5)**
- También podemos **retrasar** el momento de la escucha mediante el parámetro **offset**:  
**audio\_text = r.record(source, offset = 5)**

## Ejemplo 17.- One Wire (Sensor de tº DS18B20)

### ¿Qué es one-wire?

- 1-Wire es un **protocolo de comunicaciones** en serie diseñado por **Dallas Semiconductor**. Está basado en un bus, **un maestro y varios esclavos** de una sola línea de datos en la que se alimentan. Por supuesto, necesita una referencia a tierra común a todos los dispositivos.
- La línea de datos/alimentación requiere una resistencia de **pull-up** conectada a la **alimentación** y que le proporciona ésta.
- Permite integrar **sensores de temperatura y llaves electrónicas** de una forma fácil, fiable y económica.

### DSB18B20

El DS18b20 está disponible en tres versiones: Transistor **individual**, en **módulo** y **sumergible**.

#### Absolute Maximum Ratings

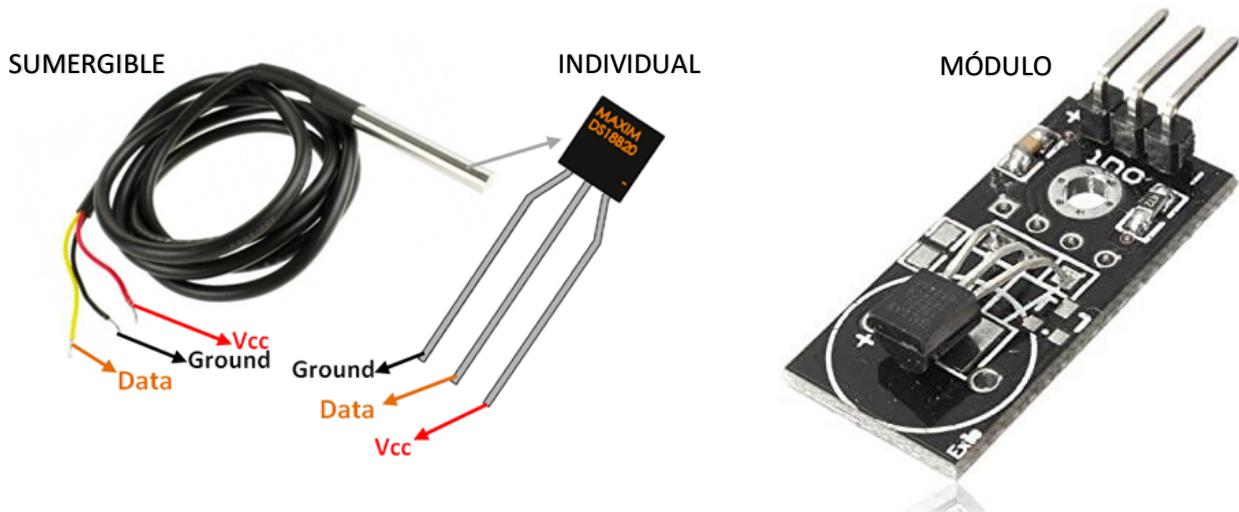
|                                                   |                 |                                 |                                                 |
|---------------------------------------------------|-----------------|---------------------------------|-------------------------------------------------|
| Voltage Range on Any Pin Relative to Ground ..... | -0.5V to +6.0V  | Storage Temperature Range ..... | -55°C to +125°C                                 |
| Operating Temperature Range.....                  | -55°C to +125°C | Solder Temperature.....         | Refer to the IPC/JEDEC J-STD-020 Specification. |

*These are stress ratings only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.*

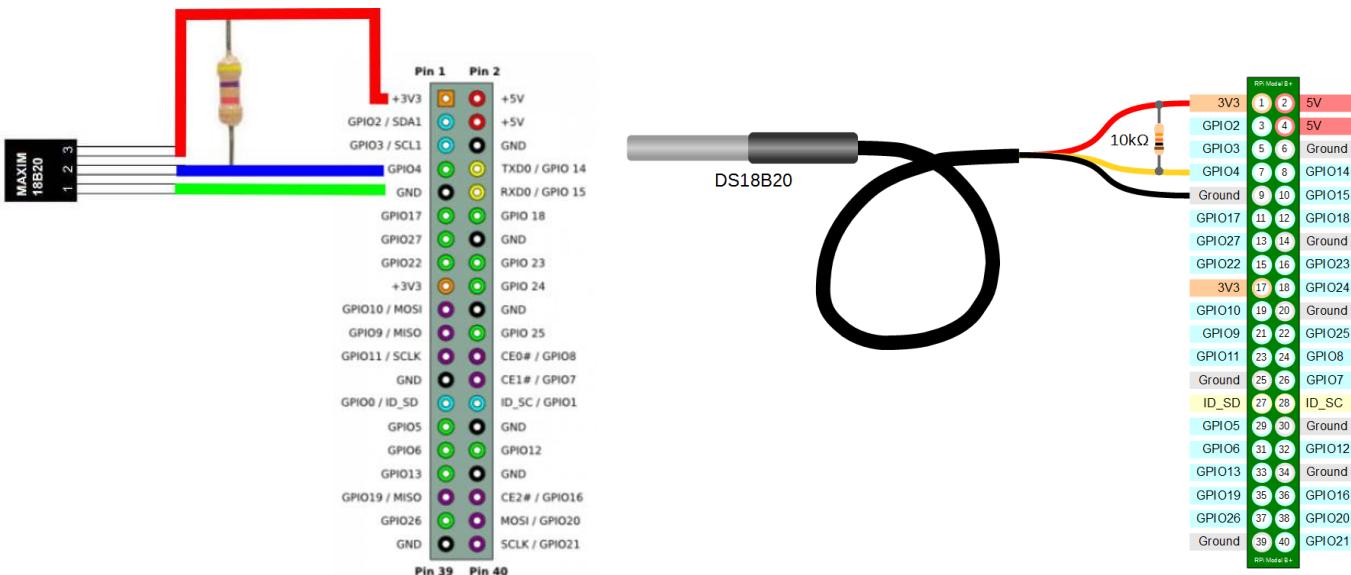
#### DC Electrical Characteristics

(-55°C to +125°C; V<sub>DD</sub> = 3.0V to 5.5V)

| PARAMETER             | SYMBOL           | CONDITIONS                    |              | MIN  | TYP                                       | MAX             | UNITS |
|-----------------------|------------------|-------------------------------|--------------|------|-------------------------------------------|-----------------|-------|
| Supply Voltage        | V <sub>DD</sub>  | Local power (Note 1)          |              | +3.0 |                                           | +5.5            | V     |
| Pullup Supply Voltage | V <sub>PU</sub>  | Parasite power                | (Notes 1, 2) | +3.0 |                                           | +5.5            | V     |
|                       |                  | Local power                   |              | +3.0 |                                           | V <sub>DD</sub> |       |
| Thermometer Error     | t <sub>ERR</sub> | -10°C to +85°C                | (Note 3)     |      |                                           | ±0.5            | °C    |
|                       |                  | -30°C to +100°C               |              |      |                                           | ±1              |       |
|                       |                  | -55°C to +125°C               |              |      |                                           | ±2              |       |
| Input Logic-Low       | V <sub>IL</sub>  | (Notes 1, 4, 5)               |              | -0.3 |                                           | +0.8            | V     |
| Input Logic-High      | V <sub>IH</sub>  | Local power                   | (Notes 1,6)  | +2.2 | The lower of 5.5 or V <sub>DD</sub> + 0.3 |                 | V     |
|                       |                  | Parasite power                |              | +3.0 |                                           |                 |       |
| Sink Current          | I <sub>L</sub>   | V <sub>I/O</sub> = 0.4V       |              | 4.0  |                                           |                 | mA    |
| Standby Current       | I <sub>DDS</sub> | (Notes 7, 8)                  |              | 750  | 1000                                      |                 | nA    |
| Active Current        | I <sub>DD</sub>  | V <sub>DD</sub> = 5V (Note 9) |              | 1    | 1.5                                       |                 | mA    |
| DQ Input Current      | I <sub>DQ</sub>  | (Note 10)                     |              | 5    |                                           |                 | μA    |
| Drift                 |                  | (Note 11)                     |              | ±0.2 |                                           |                 | °C    |

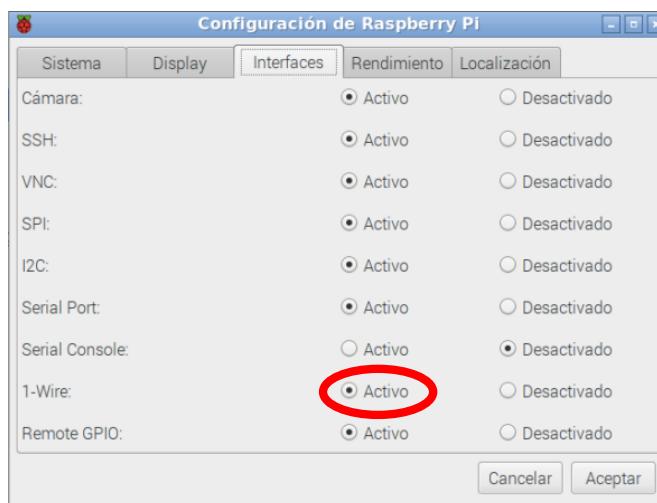


Excepto la versión módulo, es necesario que coloquemos una resistencia «pull-up» de 4K7 ohmios, y lo conectemos a la Raspberry Pi según el siguiente esquema:

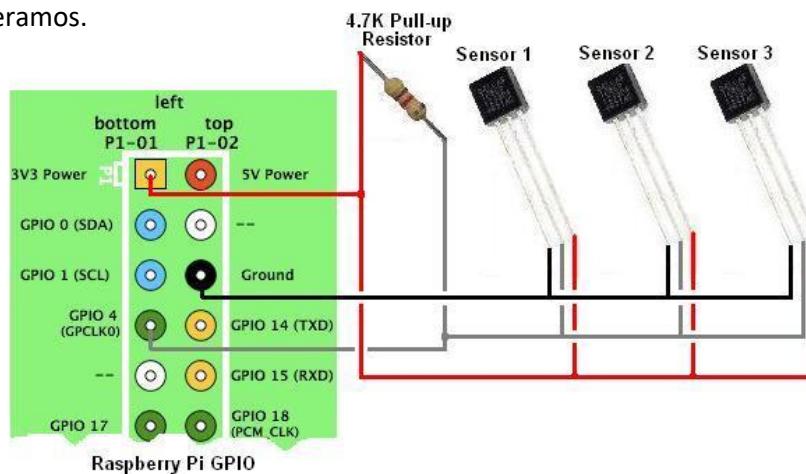


Una vez conectado, la lectura de la temperatura es muy sencilla; se hace leyendo un fichero. Basta seguir los siguientes pasos:

### 1. Habilitamos el pin **GPIO4**, que es el que trae por defecto la Raspberry para trabajar con One-Wire.



Podemos colocar muchos sensores en paralelo, con una sola resistencia para todos ellos. Como cada sensor tiene un código diferente, aparecerán carpetas para cada uno de ellos, y no tendremos más que leer el que queramos.



2. Ejecutamos los siguientes comandos:

```
$ sudo modprobe w1-gpio
$ sudo modprobe w1-therm
```

A continuación listamos el contenido de /sys/bus/w1/devices.

```
$ ls /sys/bus/w1/devices
```

```
pi@rpicovid19:~$ ls /sys/bus/w1/devices
pi@rpicovid19:~$ ls /sys/bus/w1/devices/28-000005fe5418
pi@rpicovid19:~$ ls /sys/bus/w1/devices/28-000005fe5418
28-000005fe5418 w1_bus_master1
```

Aparecerán varios directorios, entre ellos uno (o varios, si tenemos varios termómetros conectados), con el nombre 28-XXXXXX. 28-XXXXXX es el número de serie de nuestro termómetro, diferente para cada transistor. Por ejemplo, este termómetro es **28-000005fe5418**.

3. Entramos en ese directorio (reemplazando XXXXX por nuestro código), y leemos el fichero **w1\_slave**.

```
$ cat /sys/bus/w1/devices/28-000005fe5418/w1_slave
```

La salida será algo así:

```
pi@rpicovid19:~$ cat /sys/bus/w1/devices/28-000005fe5418/w1_slave
pi@rpicovid19:~$ cat /sys/bus/w1/devices/28-000005fe5418/w1_slave
60 01 4b 46 7f ff 10 10 b5 : crc=b5 YES
60 01 4b 46 7f ff 10 10 b5 t=22000
```

La temperatura viene expresada en m°C. En la salida anterior dice t=2200, así que tendremos que dividir por 1000 para obtener la temperatura en °C, es decir: **22.000 °C**.

## DESDE PYTHON

```
-*- coding: utf-8 -*-
"""

La lectura del fichero "w1_slave", tiene la siguiente estructura:
70 01 4b 46 7f ff 10 10 e1 : crc=e1 YES
70 01 4b 46 7f ff 10 10 e1 t=23000
...
def get_temp_sens():
 tfile = open("/sys/bus/w1/devices/28-000005fe5418/w1_slave")
 text = tfile.read()
 tfile.close()

 #La siguiente orden va a situar el cursor después del cambio de linea
 secondline = text.split("\n")[1]

 ...
 Situamos el cursor tras el noveno espacio, es decir, en --> t=2300
 70 01 4b 46 7f ff 10 10 e1 t=23000
 ...
 temperaturedata = secondline.split(" ")[9]

 ...
 .- Situamos el cursor después del =, es decir señalando a --> 23000
 .- El parámetro [2:] nos lleva hasta el siguiente espacio
 .- además, lo convertimos a float
 ...
 temperature = float(temperaturedata[2:])

 temperature = temperature / 1000
 return float(temperature)

mensaje = str(get_temp_sens()) + " °C"
print(mensaje)
```

## 8.4 LIBRERÍAS

### MÓDULO os (INTERFAZ AL SISTEMA OPERATIVO)

El módulo **os** provee docenas de funciones para interactuar con el sistema operativo:

```
>>> import os

>>> os.getcwd() # devuelve el directorio de trabajo actual
'C:\\Python35'

>>> os.chdir('/server/accesslogs') # cambia el directorio de trabajo

>>> os.system('mkdir today') # ejecuta el comando 'mkdir' en el sistema
0
```

Asegurate de usar el estilo **import os** en lugar de **from os import \***. Esto evitara que **os.open()** oculte a la función integrada **open()**, que trabaja bastante diferente.

Las funciones integradas **dir()** y **help()** son útiles como ayudas interactivas para trabajar con módulos grandes como **os**:

```
>>> import os

>>> dir(os)
<devuelve una lista de todas las funciones del módulo>

>>> help(os)
<devuelve un manual creado a partir de las documentaciones del módulo>
```

### MÓDULO glob (COMODINES DE ARCHIVOS)

El módulo **glob** provee una función para hacer listas de archivos a partir de búsquedas con comodines en directorios:

```
>>> import glob

>>> glob.glob('*.py')
['primes.py', 'random.py', 'quote.py']
```

## MÓDULO sys

El módulo sys se importa como

`import sys`

El módulo provee acceso a funciones y objetos mantenidos por el intérprete.

### `sys.exit()`

Permite terminar la ejecución del script devolviendo un valor

(en Unix/Linux esto se considera una muy buena práctica ya que permite encadenar comandos en función de si terminaron bien, retornando 0 o si tuvieron algún problema, un valor diferente a 0).

### `sys.argv`

Es la lista de argumentos con la cual se invocó al intérprete. Por ejemplo:

Dado el script args.py con las siguientes líneas:

```
import sys
print sys.argv
```

c:\pruebaspython> python args.py uno dos tres

Imprimirá lo siguiente:

```
['args.py', 'uno', 'dos', 'tres']
```

### `sys.path`

Es el equivalente a CLASSPATH de Java.

Es una lista de directorios/carpetas donde Python busca los módulos cuando realizamos un import.

Ya que es una lista podemos agregar y quitar elementos.

### `sys.version`

Es una cadena que nos indica la versión de intérprete que estamos utilizando.

```
>>> import sys
>>> print sys.version
'2.5.2 (r252:60911, Jul 31 2008, 17:28:52) \n[GCC 4.2.3 (Ubuntu 4.2.3-2ubuntu7)]'
```

### `Una función main()`

Una función main característica, [según Guido](#) es la siguiente:

```
01 import sys
02
03 def main(argv = sys.argv):
04 print "Hola mundo"
05 print argv
06
07 if __name__ == "__main__":
08 sys.exit(main())
```

En la linea 3 se define una función main que toma como argumentos por defecto la entrada de sys.argv, es decir, los argumentos del usuario.

Si queremos importar la función main() desde otro módulo, por ejemplo, para realizar pruebas, podemos especificar nuestros propios argumentos.

Por último, para devolver un valor de salida al sistema, utilizamos sys.exit(), si la función retorna None, La salida es 0, indicando que no se han producido errores en el script.

## ARGUMENTOS DE LINEA DE ÓRDENES

Los programas frecuentemente necesitan procesar argumentos de linea de órdenes. Estos argumentos se almacenan en el atributo **argv** del módulo **sys** como una lista. Por ejemplo, la siguiente salida resulta de ejecutar `python demo.py uno dos tres` en la línea de órdenes:

```
>>> import sys
>>> print(sys.argv)
['demo.py', 'uno', 'dos', 'tres']
```

El módulo  **getopt** procesa `sys.argv` usando las convenciones de la función de Unix **getopt()**. El módulo **argparse** provee un procesamiento más flexible de la linea de órdenes.

## REDIRECCIÓN DE LA SALIDA DE ERROR Y FINALIZACIÓN DEL PROGRAMA

El módulo **sys** también tiene atributos para `stdin`, `stdout`, y `stderr`. Este último es útil para emitir mensajes de alerta y error para que se vean incluso cuando se haya redireccionado `stdout`:

```
>>> sys.stderr.write('Alerta, archivo de log no encontrado\n')
Alerta, archivo de log no encontrado
```

La forma más directa de terminar un programa es usar `sys.exit()`.

## MÓDULO res (COINCIDENCIA EN PATRONES DE CADENAS)

El módulo **res** provee herramientas de expresiones regulares para un procesamiento avanzado de cadenas. Para manipulación y coincidencias complejas, las expresiones regulares ofrecen soluciones concisas y optimizadas:

```
>>> import res
>>> re.findall(r'\bt[a-z]*', 'tres felices tigres comen trigo')
['tres', 'tigres', 'trigo']
>>> re.sub(r'(\b[a-z]+\b) \1', r'\1', 'gato en el sombrero')
'gato en el sombrero'
```

Cuando se necesita algo más sencillo solamente, se prefieren los métodos de las cadenas porque son más fáciles de leer y depurar.

```
>>> 'te para tos'.replace('tos', 'dos')
'te para dos'
```

## MÓDULO math / random / statistics (MATEMÁTICA)

El módulo **math** permite el acceso a las funciones de la biblioteca C subyacente para la matemática de punto flotante:

```
>>> import math
>>> math.cos(math.pi / 4)
0.70710678118654757
>>> math.log(1024, 2)
10.0
```

El módulo **random** provee herramientas para realizar selecciones al azar:

```
>>> import random
>>> random.choice(['manzana', 'pera', 'banana'])
'manzana'
>>> random.sample(range(100), 10) # elección sin reemplazo
[30, 83, 16, 4, 8, 81, 41, 50, 18, 33]
>>> random.random() # un float al azar
0.17970987693706186
>>> random.randrange(6) # un entero al azar tomado de range(6)
4
```

El módulo **statistics** calcula propiedades de estadística básica (la media, mediana, varianza, etc) de datos numéricos:

```
>>> import statistics
>>> datos = [2.75, 1.75, 1.25, 0.25, 0.5, 1.25, 3.5]
>>> statistics.mean(datos)
1.6071428571428572
>>> statistics.median(datos)
1.25
>>> statistics.variance(datos)
1.3720238095238095
```

El proyecto SciPy <<http://scipy.org>> tiene muchos otros módulos para cálculos numéricos.

## MÓDULO urllib.request / smtplib (ACCESO A INTERNET)

Hay varios módulos para acceder a internet y procesar sus protocolos. Dos de los más simples son **urllib.request** para traer data de URLs y **smtplib** para mandar correos:

```
>>> with urlopen('http://tycho.usno.navy.mil/cgi-bin/timer.pl') as response:
... for line in response:
... line = line.decode('utf-8') # Decoding the binary data to text.
... if 'EST' in line or 'EDT' in line: # look for Eastern Time
... print(line)

Nov. 25, 09:43:32 PM EST

>>> import smtplib
>>> server = smtplib.SMTP('localhost')
>>> server.sendmail('soothsayer@ejemplo.org', 'jcaesar@ejemplo.org',
... """To:jcaesar@ejemplo.org
... From:soothsayer@ejemplo.org

...
... Ojo al piojo.
... """)
>>> server.quit()
```

(Notar que el segundo ejemplo necesita un servidor de correo corriendo en la máquina local)

## MÓDULOS zlib / gzip / bz2 / lzma / zipfile / tarfile

### (COMPRESIÓN DE DATOS)

Los formatos para archivar y comprimir datos se soportan directamente con los módulos: **zlib**, **gzip**, **bz2**, **lzma**, **zipfile** y **tarfile**.

```
>>> import zlib
>>> s = b'witch which has which witches wrist watch'
>>> len(s)
41
>>> t = zlib.compress(s)
>>> len(t)
37
>>> zlib.decompress(t)
b'witch which has which witches wrist watch'
>>> zlib.crc32(s)
226805979
```

## MÓDULO `datetime / locale` (FECHAS Y TIEMPOS)

El módulo `datetime` ofrece clases para manejar fechas y tiempos tanto de manera simple como compleja. Aunque soporta aritmética sobre fechas y tiempos, el foco de la implementación es en la extracción eficiente de partes para manejarlas o formatear la salida. El módulo también soporta objetos que son conscientes de la zona horaria.

```
>>> # las fechas son fácilmente construidas y formateadas
>>> from datetime import date
>>> hoy = date.today()
>>> hoy
datetime.date(2009, 7, 19)
>>> # nos aseguramos de tener la info de localización correcta
>>> import locale
>>> locale.setlocale(locale.LC_ALL, locale.getdefaultlocale())
'es_ES.UTF8'
>>> hoy.strftime("%m-%d-%y. %d %b %Y es %A. hoy es %d de %B.")
'07-19-09. 19 jul 2009 es domingo. hoy es 19 de julio.'
>>> # las fechas soportan aritmética de calendario
>>> nacimiento = date(1964, 7, 31)
>>> edad = hoy - nacimiento
>>> edad.days
14368
```

## MÓDULOS `array` (HERRAMIENTAS PARA TRABAJAR CON LISTAS)

Muchas necesidades de estructuras de datos pueden ser satisfechas con el tipo integrado lista. Sin embargo, a veces se hacen necesarias implementaciones alternativas con rendimientos distintos.

El módulo `array` provee un objeto `array()` (vector) que es como una lista que almacena sólo datos homogéneos y de una manera más compacta. Los ejemplos a continuación muestran un vector de números guardados como dos números binarios sin signo de dos bytes (código de tipo "`H`") en lugar de los 16 bytes por elemento habituales en listas de objetos int de Python:

```
>>> from array import array
>>> a = array('H', [4000, 10, 700, 22222])
>>> sum(a)
26932
>>> a[1:3]
array('H', [10, 700])
```

## **8.5 NOCIONES DE PROGRAMACIÓN**

### **7.5.1 SENTENCIAS PASS Y WITH DE PYTHON**

#### **PASS**

Python incorpora una sentencia especial para indicar que no se debe realizar ninguna acción. Se trata de **pass** y especialmente útil cuando deseamos indicar que no se haga nada en una sentencia que requiere otra.

```
>>> while True:
 pass
```

Muchos desarrolladores emplean **pass** cuando escriben esqueletos de código que posteriormente rellenarán. Dado que inicialmente no sabemos qué código contendrá una determinada sentencia, es útil emplear **pass** para mantener el resto del programa funcional. Posteriormente, el esqueleto de código será llenado con código funcional y la sentencia **pass** será reemplazada por otras que realicen una función específica.

#### **WITH**

La sentencia **with** se utiliza con objetos que soportan el protocolo de manejador de contexto y garantiza que una o varias sentencias serán ejecutadas automáticamente. Esto nos ahorra muchas líneas de código, a la vez que nos garantiza que ciertas operaciones serán realizadas sin que lo indiquemos explícitamente. Uno de los ejemplos más claros es cuando leemos un archivo de texto. Al terminar esta operación siempre es recomendable cerrar el archivo. Gracias a **with** esto ocurrirá automáticamente, sin necesidad de llamar al método `close()`. Por ejemplo:

```
>>> with open(r'codejobs.txt') as myfile:
 for line in myfile:
 print(line)
```

### **7.5.2 MANEJANDO EXCEPCIONES**

Es posible escribir programas que manejen determinadas excepciones. Mirá el siguiente ejemplo, que le pide al usuario una entrada hasta que ingrese un entero válido, pero permite al usuario interrumpir el programa (usando **Control-C** o lo que sea que el sistema operativo soporte); notar que una interrupción generada por el usuario se señaliza generando la excepción **KeyboardInterrupt**.

```
>>> while True:
... try:
... x = int(input("Por favor ingrese un número: "))
... except ValueError:
... print("Oops! No era válido. Intente nuevamente...")
... ...
```

## TRY / EXCEPT

La declaración **try** funciona de la siguiente manera:

- Primero, se ejecuta el *bloque try* (el código entre las declaraciones **try** y **except**).
- Si no ocurre ninguna excepción, el *bloque except* se saltea y termina la ejecución de la declaración **try**.
- Si ocurre una excepción durante la ejecución del *bloque try*, el resto del bloque se saltea. Luego, si su tipo coincide con la excepción nombrada luego de la palabra reservada **except**, se ejecuta el *bloque except*, y la ejecución continúa luego de la declaración **try**.
- Si ocurre una excepción que no coincide con la excepción nombrada en el **except**, esta se pasa a declaraciones **try** de más afuera; si no se encuentra nada que la maneje, es una *excepción no manejada*, y la ejecución se frena con un mensaje como los mostrados arriba.

Una declaración **try** puede tener más de un **except**, para especificar manejadores para distintas excepciones. A lo sumo un manejador será ejecutado. Sólo se manejan excepciones que ocurren en el correspondiente **try**, no en otros manejadores del mismo **try**. Un **except** puede nombrar múltiples excepciones usando paréntesis, por ejemplo:

```
... except (RuntimeError, TypeError, NameError):
... pass
```

El último **except** puede omitir nombrar qué excepción captura, para servir como comodín. Usá esto con extremo cuidado, ya que de esta manera es fácil ocultar un error real de programación. También puede usarse para mostrar un mensaje de error y luego re-generar la excepción (permitiéndole al que llama, manejar también la excepción):

```
import sys

try:
 f = open('miarchivo.txt')
 s = f.readline()
 i = int(s.strip())

except OSError as err:
 print("Error OS: {0}".format(err))

except ValueError:
 print("No pude convertir el dato a un entero.")

except:
 print("Error inesperado:", sys.exc_info()[0])
 raise
```

Las declaraciones `try ... except` tienen un *bloque else* opcional, el cual, cuando está presente, debe seguir a los `except`. Es útil para aquel código que debe ejecutarse si el *bloque try* no genera una excepción. Por ejemplo:

```
for arg in sys.argv[1:]:
 try:
 f = open(arg, 'r')
 except IOError:
 print('no pude abrir', arg)
 else:
 print(arg, 'tiene', len(f.readlines()), 'lineas')
 f.close()
```

El uso de `else` es mejor que agregar código adicional en el `try` porque evita capturar accidentalmente una excepción que no fue generada por el código que está protegido por la declaración `try ... except`.

## FINALLY

La declaración `try` tiene otra cláusula opcional que intenta definir acciones de limpieza que deben ser ejecutadas bajo ciertas circunstancias. Por ejemplo:

```
>>> try:
... raise KeyboardInterrupt
... finally:
... print('Chau, mundo!')
...
Chau, mundo!
```

**KeyboardInterrupt**

Traceback (most recent call last):

File "<stdin>", line 2, in ?

Una cláusula **finally** siempre es ejecutada antes de salir de la declaración **try**, ya sea que una excepción haya ocurrido o no. Cuando ocurre una excepción en la cláusula **try** y no fue manejada por una cláusula **except** (u ocurrió en una cláusula **except** o **else**), es relanzada luego de que se ejecuta la cláusula **finally**.

El **finally** es también ejecutado “a la salida” cuando cualquier otra cláusula de la declaración **try** es dejada via **break**, **continue** or **return**. Un ejemplo más complicado:

```
>>> def dividir(x, y):
... try:
... result = x / y
... except ZeroDivisionError:
... print("¡división por cero!")
... else:
... print("el resultado es", result)
... finally:
... print("ejecutando la clausula finally")
...
>>> dividir(2, 1)
el resultado es 2.0
ejecutando la clausula finally
>>> dividir(2, 0)
¡división por cero!
ejecutando la clausula finally
>>> divide("2", "1")
ejecutando la clausula finally
Traceback (most recent call last):
 File "<stdin>", line 1, in ?
 File "<stdin>", line 3, in divide
TypeError: unsupported operand type(s) for /: 'str' and 'str'
```

Como podéis ver, la cláusula **finally** es ejecutada siempre. La excepción **TypeError** lanzada al dividir dos cadenas de texto no es manejado por la cláusula **except** y por lo tanto es relanzada luego de que se ejecuta la cláusula **finally**.

En aplicaciones reales, la cláusula **finally** es útil para liberar recursos externos (como archivos o conexiones de red), sin importar si el uso del recurso fue exitoso.

### 8.5.3 ¿Qué es `if __name__ == “main”:`?

Es común encontrarnos código con esta forma:

```
def hacer_algo():
 print "algo"

if __name__ == "__main__":
 hacer_algo()
```

En lugar de, por ejemplo:

```
def hacer_algo():
 print "algo"
```

`hacer_algo()`

También notamos que la variable `__name__` no es inicializada por nosotros, pero existe en el entorno.

¿Cuál es la diferencia? ¿Por qué preferir uno sobre el otro? ¿Qué es y cómo funciona `if __name__ == “main”:`?

Esto está ligado al modo de funcionamiento del intérprete Python, cuando el intérprete lee un archivo de código, **ejecuta todo el código** que se encuentra en él. Todo módulo (archivo de código) en python tiene un atributo especial llamado `__name__` que define el espacio de nombres en el que se está ejecutando. Es usado para identificar de forma única un módulo en el sistema de importaciones.

Por su parte `__main__` es el nombre del ámbito en el que se ejecuta el código de nivel superior (tu programa principal).

El intérprete pasa el valor del atributo a '`__main__`' si el módulo se está ejecutando como programa principal (cuando lo ejecutas llamando al intérprete en la terminal con `python my_modulo.py`, haciendo doble click en él, ejecutandolo en el intérprete interactivo, etc ).

Si el módulo no es llamado como programa principal, sino que es importado desde otro módulo, el atributo `__name__` pasa a contener el nombre del archivo en si. [Aquí](#) tienes la documentación oficial.

Es decir, si tienes un archivo llamado `mi_modulo.py`, si lo ejecutamos como programa principal el atributo `__name__` será '`__main__`', si lo usamos importandolo desde otro modulo (`import mi_modulo`) el atributo `__name__` será igual a '`mi_modulo`'.

Básicamente, lo que haces usando `if __name__ == “main”:`: es ver si el módulo ha sido importado o no. Si no se ha importado (se ha ejecutado como programa principal) ejecuta el código dentro del condicional.

Una de las razones para hacerlo es que, a veces, se escribe un módulo (un archivo .py) que se puede ejecutar directamente, pero que alternativamente, también se puede importar y reutilizar sus funciones, clases, métodos, etc en otro módulo. Con esto conseguimos que la ejecución sea diferente al ejecutar el módulo directamente que al importarlo desde otro programa.

Para ver como funciona puedes probar algún código (siguiendo tu propio ejemplo):

Creas un módulo al que llamaremos `mi_modulo.py`:

```
def hacer_algo():
 print("algo")

if __name__ == "__main__":
 print('Ejecutando como programa principal')
 hacer_algo()
```

En la misma carpeta creas otro módulo al que llamaremos `principal.py`

```
import mi_modulo
```

```
mi_modulo.hacer_algo()
```

Si ejecutas el script `mi_modulo.py` directamente el valor de `__name__` será '`__main__`' y se ejecutará lo que hay dentro del `if __name__ == "__main__":`:

Ejecutando como programa principal  
algo

Si ejecutamos el archivo `principal.py` que usa `mi_modulo.py` importándolo no se cumple la condición `if __name__ == "__main__"` por lo que no se ejecuta nada, solo la propia llamada a la función que hacemos desde el programa principal:

algo

Ahora vamos a cambiar el script `mi_modulo.py` por:

```
def hacer_algo():
 print("algo")
print('Ejecutando como programa principal')
hacer_algo()
```

Si lo ejecutamos como programa principal el resultado es el mismo que antes, pero si ejecutamos ahora `principal.py` nos sale esto:

Ejecutando como programa principal  
algo  
algo

Lo que pasa es que al importar ejecutamos todo el código y se imprime:

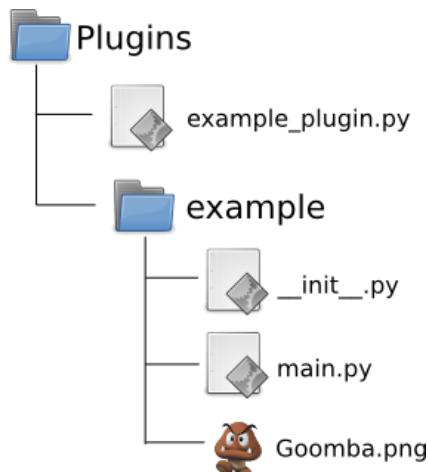
Ejecutando como programa principal  
algo

Luego al llamar a la función nos imprime:

algo

En definitiva, las dos formas que pones son válidas si ejecutas el script siempre como programa principal, si lo usas también importandolo no sueles querer que se ejecute código si no llamas a una función tu mismo y esa es la razón de usar `if __name__ == "__main__":`:

### 8.5.3 ¿Cómo funciona "`__init__.py`"?



El principal **uso de `__init__.py`** es inicializar paquetes de Python. La forma más fácil de darnos cuenta es ir hasta el directorio donde tenemos instalado python y fijarnos en las librerías y paquetes que trae por defecto python. Ahí vamos a ver que todos contienen un archivo `__init__.py`.

Vamos a poner un ejemplo:



El **archivo `__init__.py`** se utiliza para realizar configuraciones de importación. Algo que se utiliza mucho en este archivo es la **importación de clases, funciones**, etc, para que puedan ser utilizadas en otros paquetes.

En el proyecto de la imagen tenemos una carpeta principal llamada **Proyecto** y dentro de ella otra subcarpeta llamada **PythonDiario**. Estas carpetas a su vez, contienen módulos creados por nosotros en Python. Si el archivo **main.py**, que es el principal, necesita importar algo de la subcarpeta PythonDiario, va a necesitar del archivo `__init__.py` para hacerlo.

Si el archivo `__init__.py` no existiese, no se podría importar nada desde el archivo **main.py** que está en la carpeta Proyecto.

**Lo que hace el `__init__` entonces es configurar el comando de importación global:**

Lo que hace el `__init__.py`, es que estando el, yo puedo importar desde el **main.py** de la siguiente manera:

```
»»»from PythonDiario import *
```

En este caso importará todo, o lo que esté escrito dentro del archivo `__init__.py`

Si nosotros dejamos el archivo `__init__.py` vacío, podemos importar desde el **main.py** de la siguiente forma:

```
»»»import PythonDiario #Como si fuera un modulo
```

El archivo `__init__.py` no tiene por qué tener contenido dentro, se puede dejar vacío.

**Conclusión:**

Lo que hace `__init__.py` es "**convertir**" un directorio en un **módulo** (paquete) que contiene otros módulos, y esto lo hace para poder importarlos.

# 9.- FIRMATA

## INSTALAR NODJS

### 1º MÉTODO

La posibilidad de poder instalar una versión de NodeJS en nuestra Raspberry Pi nos permite ampliar notablemente nuestras opciones al poder crear pequeñas aplicaciones encargadas de tareas complejas, tales como visualización y control de sensores, pudiendo crear de una forma muy rápida complejos paneles de control.

```
$sudo apt-get install nodejs npm
```

**NPM** es el gestor de paquetes para Node.js, con él podemos instalar módulos de JavaScript que nos facilitan programar, en este caso nos van a facilitar comunicarnos con la Standard Firmata que tendremos ya instalada en el Arduino.

Para comprobar la versión instalada:

```
pi@64gb_nuevo:~ $ node -v
v4.8.2
```

### 2º MÉTODO (Recomendado)

Procedamos con la instalación de la última versión de Node en este momento, que es el Nodo 9.9.0.

El comando siguiente actualiza nuestro repositorio de paquetes Debian apt para incluir los paquetes NodeSource.

```
$ curl -sL https://deb.nodesource.com/setup_11.x | sudo -E bash -

$ sudo apt-get install nodejs
pi@rpi_sandisk_16_casa:~ $ node -v
v9.9.0
```

Nota.- Se puede ir a la siguiente página para encontrar la última versión:

<https://github.com/nodesource/distributions>

## INSTALAR JOHNNY-FIVE

**Johnny Five** es un módulo de NPM que facilita (increíblemente) controlar hardware con JavaScript, inicialmente a los Arduinos, pero con él puedes controlar muchas cosas más, como el BeagleBone, la RaspberryPi, el Intel Galileo, entre muchos otros.

```
$ npm install johnny-five
```

<http://johnny-five.io/examples/raspi-io/>

## INSTALAR RASPI-IO

**Raspi-io** es una biblioteca compatible con la API Firmata, para Raspbian que se ejecuta en la Raspberry Pi, que se puede usar como un plugin de E / S con **Johnny-Five**. **Raspi IO** admite todos los modelos de Raspberry Pi, excepto el modelo A.

```
$ npm install raspi-io
```

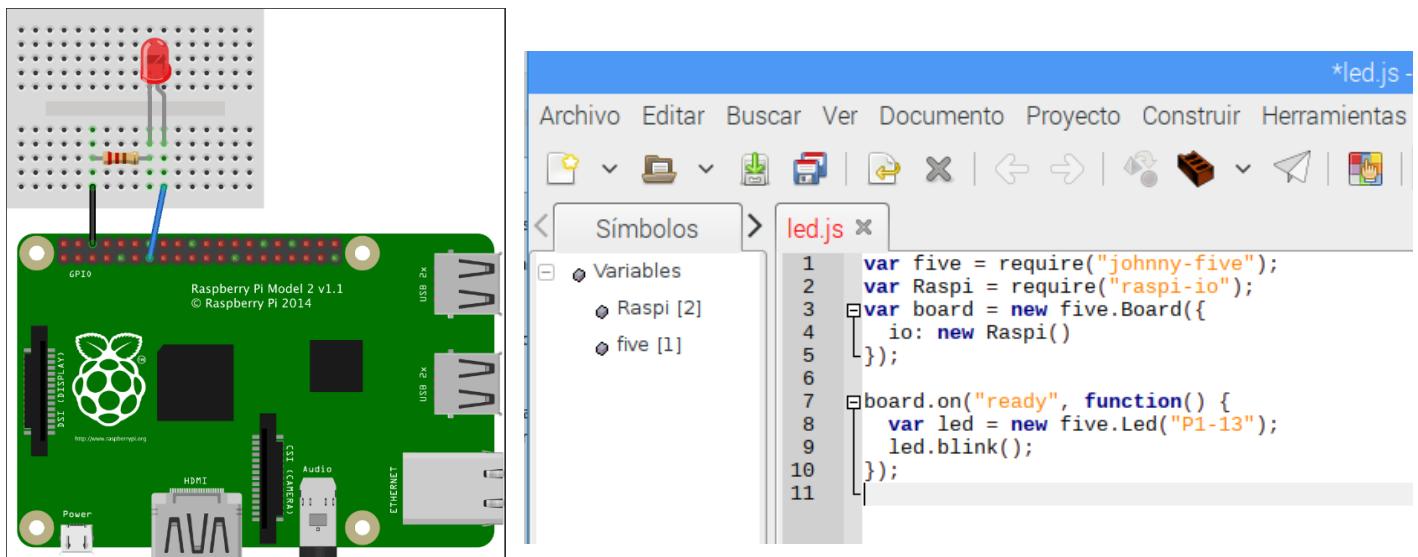
## INSTALAR GEANY (Editor de Javascripts)

Para su instalación ejecutamos la siguiente orden (A partir de la NOOBS\_v2\_8\_0, viene incorporado):

```
$sudo apt-get install geany xterm
```

### EJEMPLO JOHNNY FIVE

#### LED BLINK

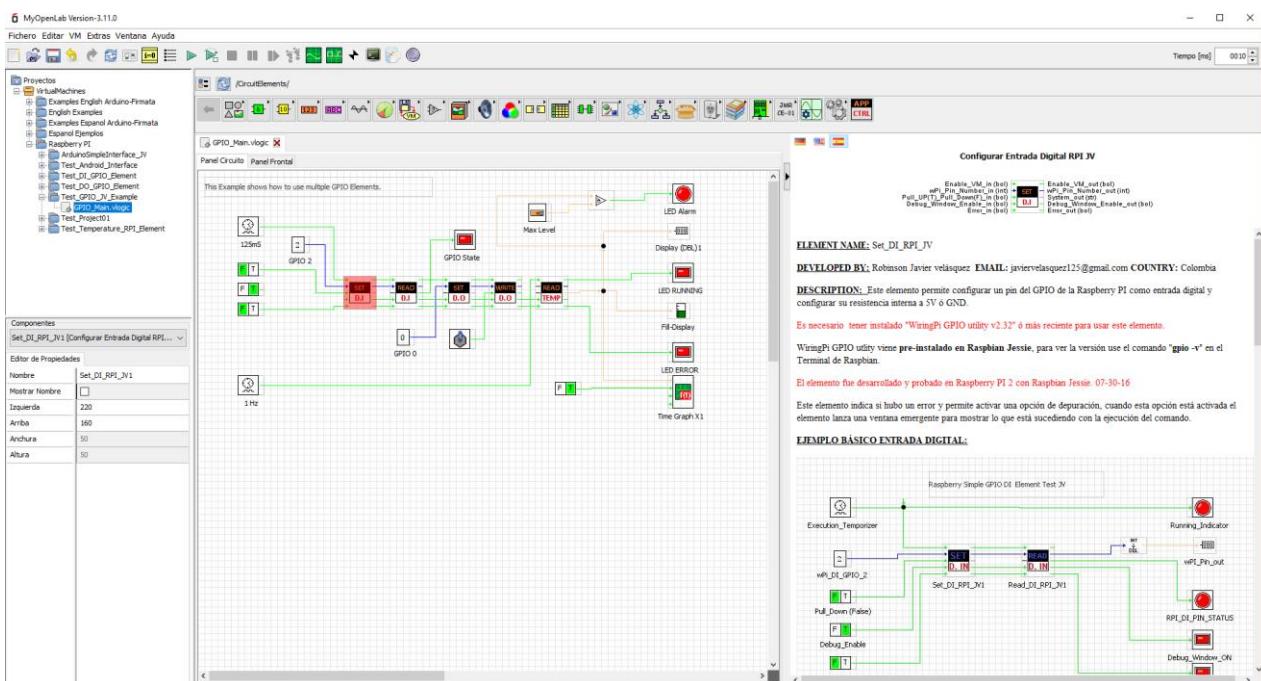


```
pi@rpi_64g_luis:~/programas/nodejs $ sudo node led.js
Unknown board revision a020d3, assuming Raspberry Pi 3 pinout.
1524212470470 Available RaspberryPi-IO
1524212470750 Connected RaspberryPi-IO
1524212470758 Repl Initialized
>> █
```

## INSTALAR MyOpenLab

[https://myopenlab.de/download\\_myopenlab\\_now.html](https://myopenlab.de/download_myopenlab_now.html)

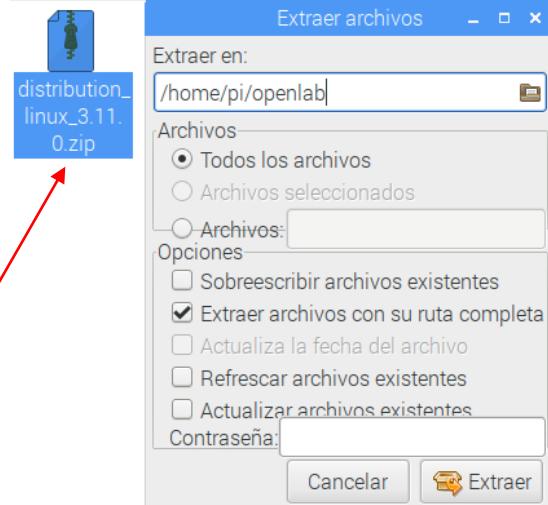
| Datum                                                                                                                      | Download                                                                 |
|----------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------|
| <b>Attention: Version 3.10.0 has a Bug that does not allow you to open Projects, but we are solving this Problem soon.</b> |                                                                          |
|                                                                                                                            | Thanx to Javier Velasquez for reporting.                                 |
| 02.06.2017                                                                                                                 | MyOpenLab V-3.10.0 (Window 32 Bit)                                       |
| 02.06.2017                                                                                                                 | MyOpenLab V-3.10.0 (Window 64 Bit and Java 32 Bit)                       |
| 02.06.2017                                                                                                                 | MyOpenLab V-3.10.0 (Window 64 Bit and Java 64 Bit)                       |
| 02.06.2017                                                                                                                 | MyOpenLab V-3.10.0 (Linux 32/64 Bit with rxtx for Java 64 Bit and RS232) |



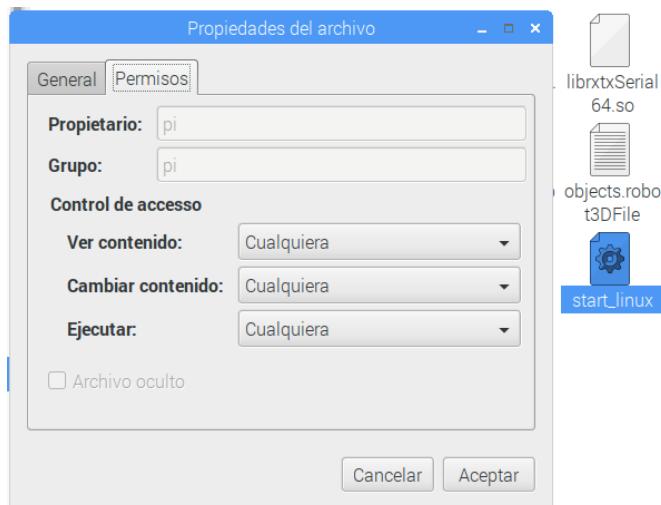
**MyOpenlab** es un software de desarrollo basado en elementos gráficos.

Integra elementos para comunicarse con plataformas como Arduino, Raspberry, Android y dispositivos con Comunicación Serial

Una vez bajado el archivo, lo descomprimimos



.- Seguidamente, damos permisos de ejecución al archivo “**start\_linux**”



.- Para que funcione la comunicación con Arduino, cuando tiene cargado FIRMATA, hay que realizar los siguientes pasos, tal y como explica Javier Velásquez en su blog:

<https://www.facebook.com/notes/myopenlab-colombia/setting-up-arduino-driver-on-raspbian-jessie-pixel-to-enable-the-myopenlab-firma/484894258566764/>

### Setting up Arduino Driver on Raspbian Jessie PIXEL to enable the MyOpenLab Firmata Element. 03-23-17

```
$ sudo apt-get update
$ sudo apt-get install arduino
$ sudo apt-get install librxtx-java
```

Open Arduino IDE, connect your Arduino and Upload "Standard Firmata".

```
$ sudo cp /usr/share/arduino/lib/RXTXcomm.jar /usr/lib/jvm/jdk-8-oracle-arm32-vfp-hflt/jre/lib/ext
$ sudo cp /usr/lib/jni/librxtxSerial.so /usr/lib/jvm/jdk-8-oracle-arm32-vfp-hflt/jre/lib/arm
$ sudo cp /usr/lib/jni/librxtxParallel.so /usr/lib/jvm/jdk-8-oracle-arm32-vfp-hflt/jre/lib/arm
```

Reboot The Raspberry Pi....

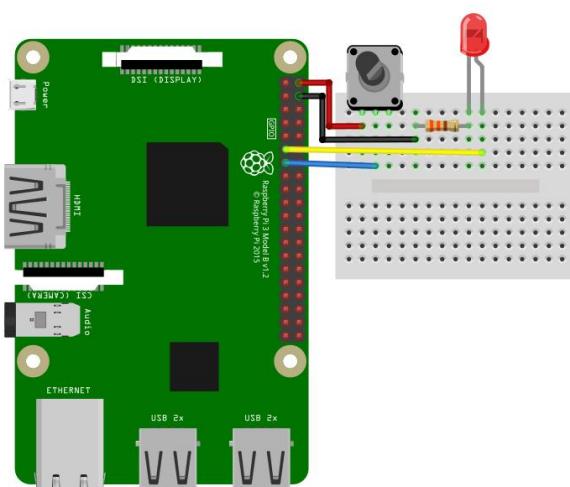
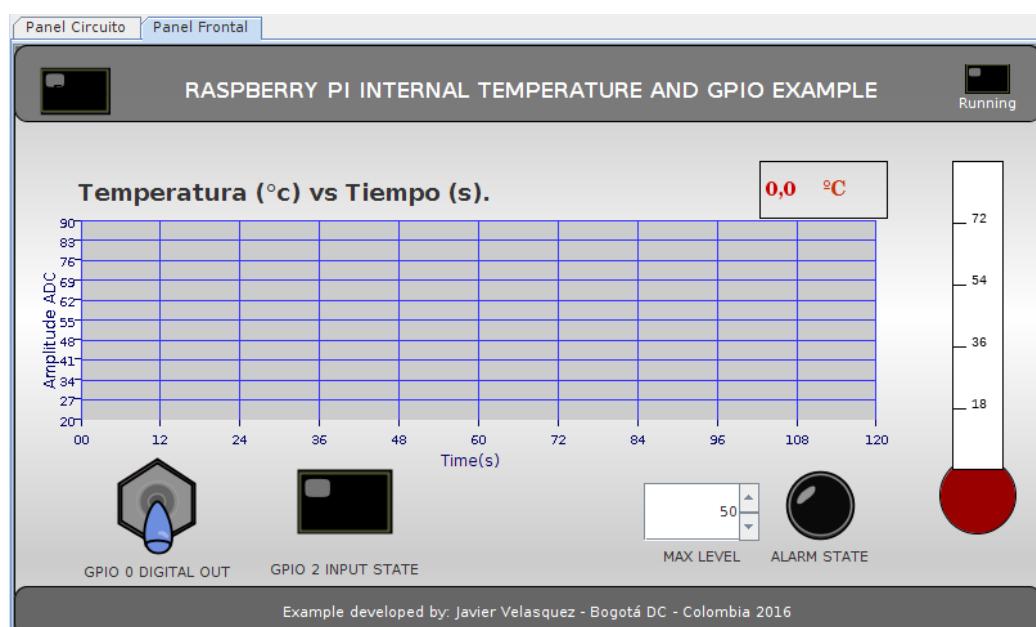
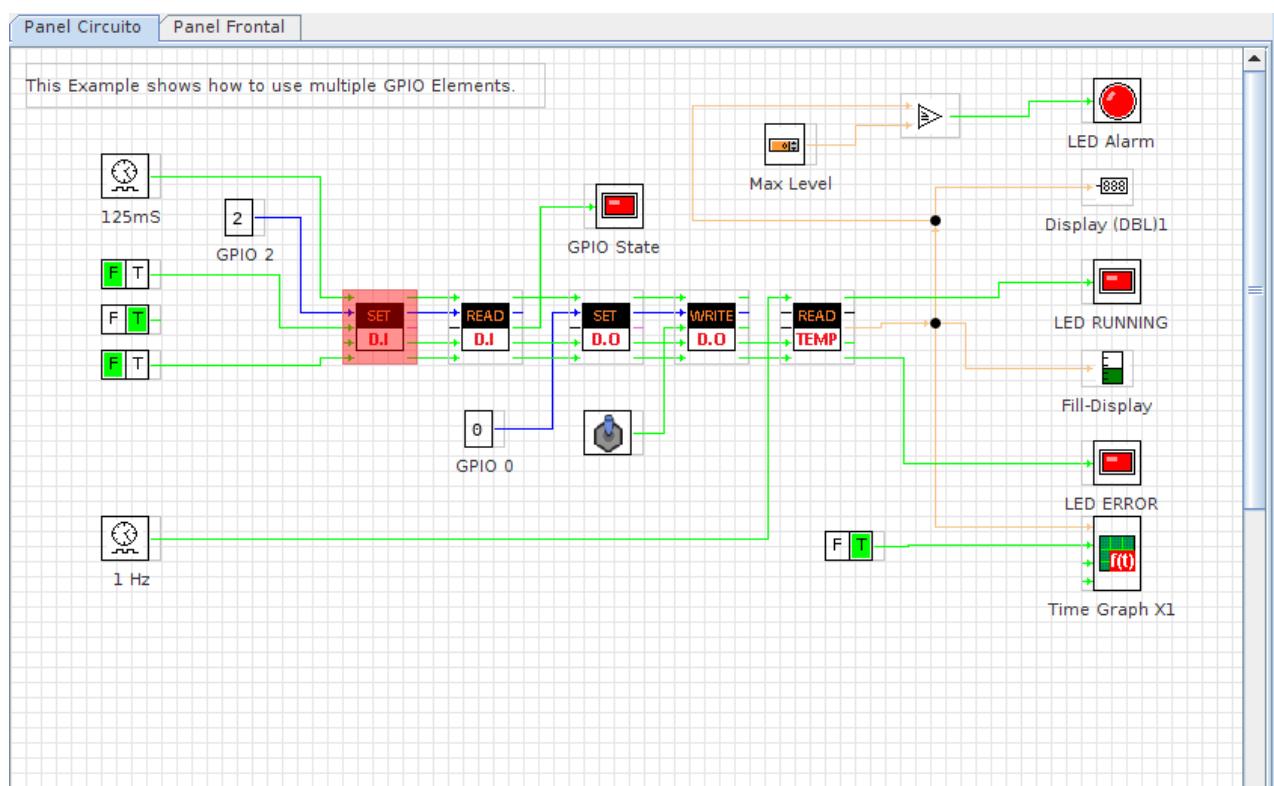
Go to your MyOpenLab "distribution" folder and find the folder "lib" to rename the old driver:

/distribution/lib/RXTXcomm.jar

as:

/distribution/lib/RXTXcomm\_JV.jar

## EJEMPLO. - TEST GPIO (Pin 11 → OUT / Pin 13 → IN)



# 10.- Cómo crear un servidor VPN

La seguridad de los accesos públicos a Internet deja a menudo mucho que desear. Para poder conectarse fuera de casa sin correr riesgos, una red virtual privada o **VPN (Virtual Private Network o red virtual privada)** puede ser una buena solución. Si además se configura un servidor VPN personal, se tendrá acceso a la red local doméstica desde Internet.

Para crear una **VPN** solo hace falta un ordenador que haga las veces de servidor y, para ello, Raspberry Pi representa una alternativa viable y económica. **OpenVPN** es el software que permite convertir al pequeño ordenador en un servidor **VPN**.

## Por qué crear un servidor VPN: funciones generales

Instalada en una red local (LAN) para poder acceder a esta desde el exterior, una VPN constituye una red de comunicación virtual en la que las peticiones y las respuestas entre el **servidor VPN** y los **clientes VPN** (dispositivos conectados al servidor) se transfieren por Internet. Esto permite poder conectarse a la LAN desde cualquier acceso a Internet y **acceder a los datos en la misma, comunicarse con los dispositivos por control remoto** (una impresora o un fax) o **utilizar la conexión a Internet de la red local** (doméstica). Dado que la conexión al servidor VPN está cifrada, navegar es mucho más seguro que si se hiciera desde un acceso no controlado (como las redes inalámbricas públicas).

Para que sea posible crear esta conexión segura a un servidor VPN hay que instalar un servidor VPN con conexión permanente a Internet en un ordenador de la red local. Este ordenador actuaría entonces como host de la red virtual. Con un software cliente se conectan los dispositivos (portátil, smartphone, tablet) con el servidor y una **conexión cifrada (túnel VPN)** permite acceder con uno de estos clientes o terminales al servidor VPN desde una red externa a la LAN personal.

Este túnel VPN lleva del cliente al servidor VPN abarcando la conexión global a Internet, lo que lo convierte en una **conexión mucho más segura que cualquier conexión a Internet** al uso. Para los hackers sería muy difícil, por ejemplo, acceder al túnel para espionar el tráfico de datos. Esto hace que las VPN sean la solución ideal en aquellos casos en que se ha de trabajar con datos sensibles, por ejemplo, los bancarios, en redes abiertas.

### 1. Configuramos una IP estática en la Raspi.

En el archivo /etc/network/interfaces añadimos la IP estática:

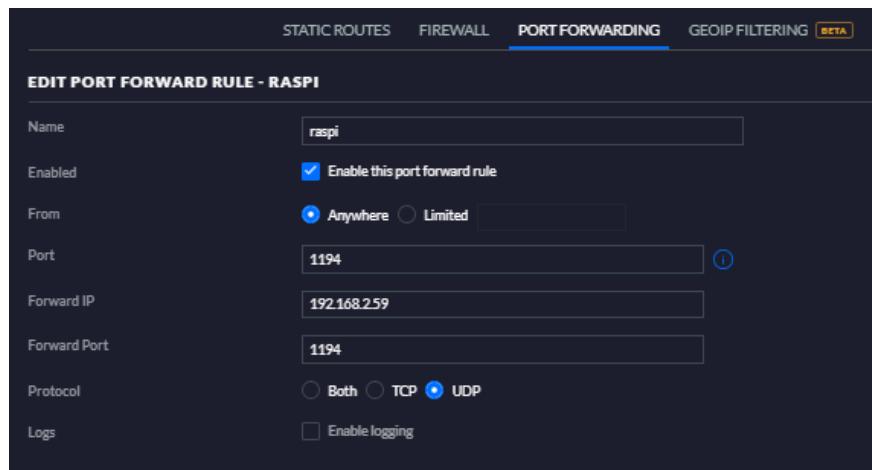
```
interfaces(5) file used by ifup(8) and ifdown(8)
Please note that this file is written to be used with dhcpcd
For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'
Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d

auto eth0
iface eth0 inet static
 address 192.168.2.59
 netmask 255.255.255.0
 gateway 192.168.2.1
```

## 2. Redireccionamos el puerto en el router hacia la Raspi

Otro aspecto importante es que tendrás que configurar un redireccionamiento de puertos de tu router hacia Raspberry Pi donde estarás ejecutando OpenVPN.

El puerto predeterminado que necesitas redireccionar es el **1194**, sin embargo, te recomiendo que uses otro diferente, ya que en caso de que alguien quiera atacar tu red, los puertos conocidos son los primeros en ser escaneados en busca de fallos de seguridad. El **protocolo** que tendrás que utilizar para este puerto es **UDP**.



## 3. Instalamos OPENVPN

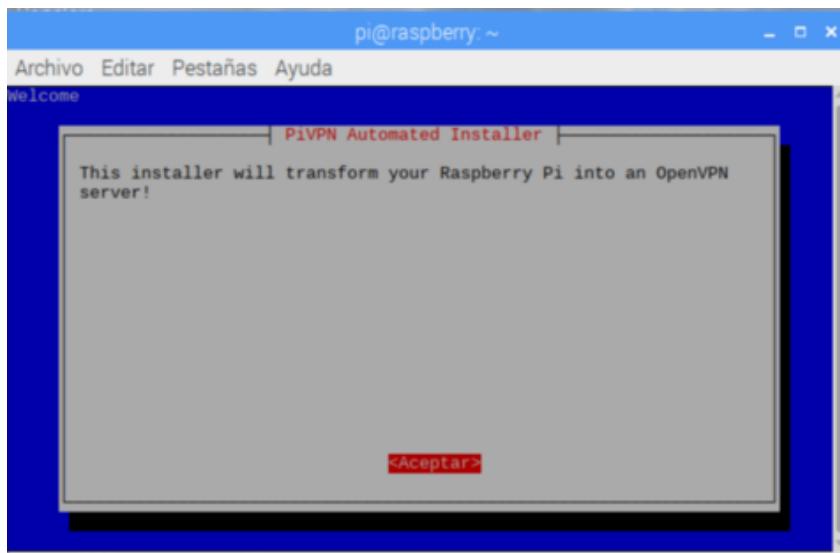
Configurar un Servidor VPN en Raspberry Pi puede ser un proceso bastante complicado, normalmente tendrías que instalar el software, generar las claves de cifrado, agregar el puerto al firewall, configurar la Pi para mantener una dirección IP estática y mucho más.

Afortunadamente, hay una manera mucho más fácil de configurar un servidor VPN Raspberry Pi gracias a un **script de instalación llamado PiVPN** que se encarga de todo el trabajo duro para configurar una VPN y reduce los errores que se pueden cometer.

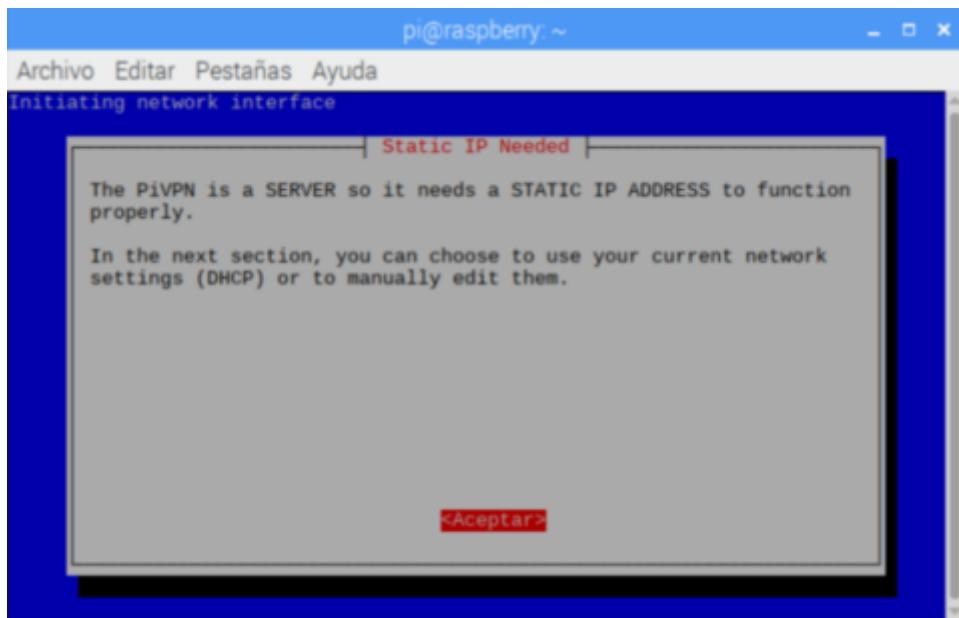
Ejecuta el siguiente comando, este comando descarga el script de [instalación de la página de GitHub de PiVPN](#) y lo ejecuta.

```
curl -L https://install.pivpn.io | bash
```

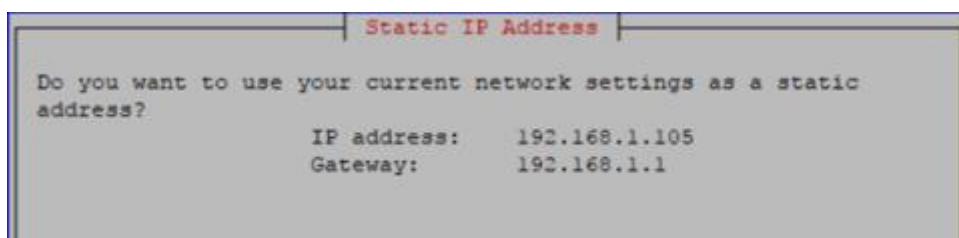
Una vez que hayas ejecutado el comando anterior verás la siguiente pantalla, selecciona OK y pulsa enter.



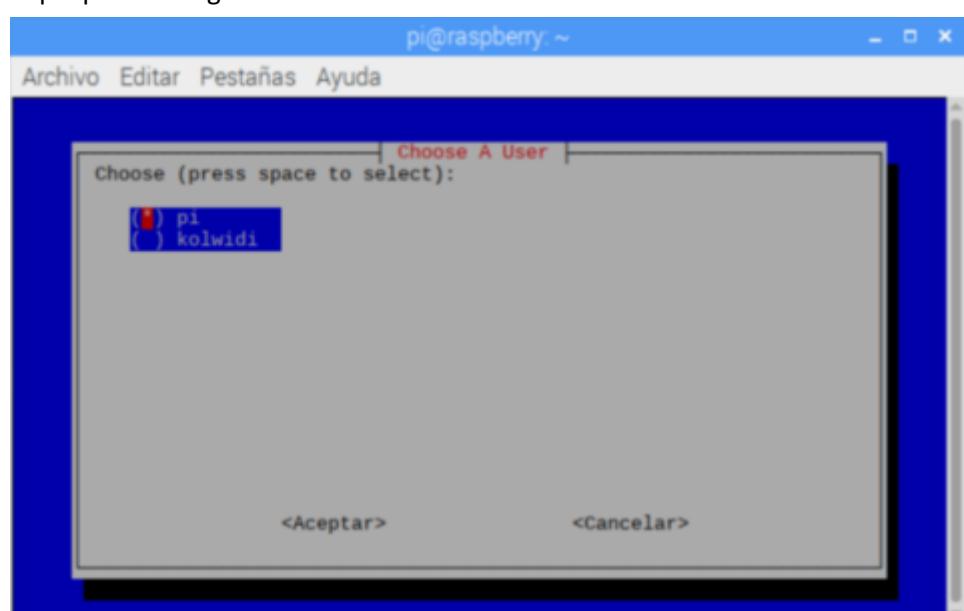
La siguiente pantalla explica que tendrás que configurar una dirección IP estática para la Raspberry Pi, esto sirve para que cuando se reinicie la Raspberry Pi por cualquier motivo, intentará utilizar la misma dirección IP nuevamente.



Aquí, solo seleccionaremos **<Aceptar>** para usar la configuración de red actual como una dirección IP local estática y en la siguiente pantalla acepta si estas de acuerdo o haz los cambios pertinentes para que se muestra la dirección IP que quieras.

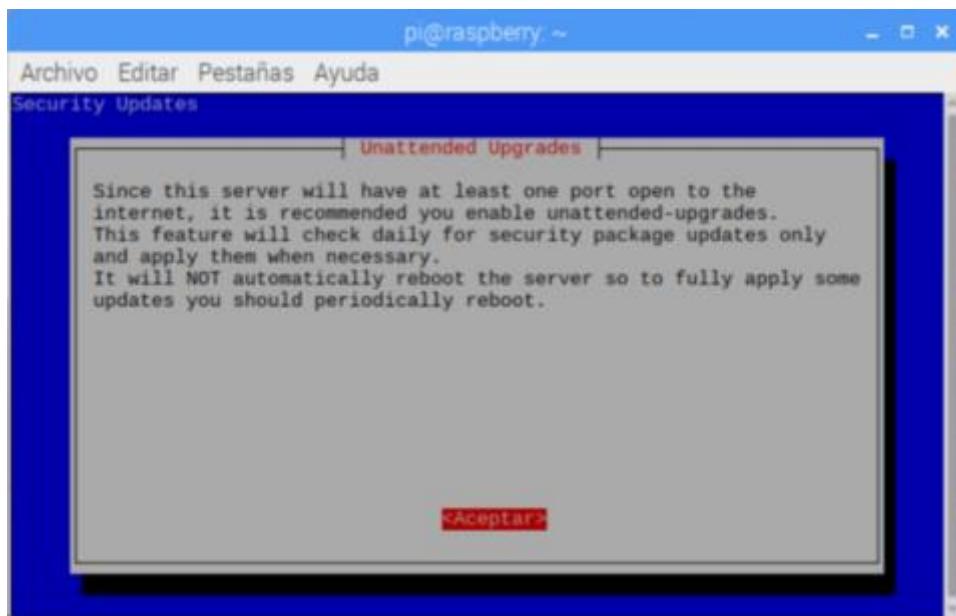


Las siguientes pantallas explican que necesitaremos establecer un usuario local para el cual se crearán las configuraciones de OVPN. Puedes seleccionar **<Ok>** y pasar a la siguiente pantalla donde verás una lista de usuarios que podrás elegir.

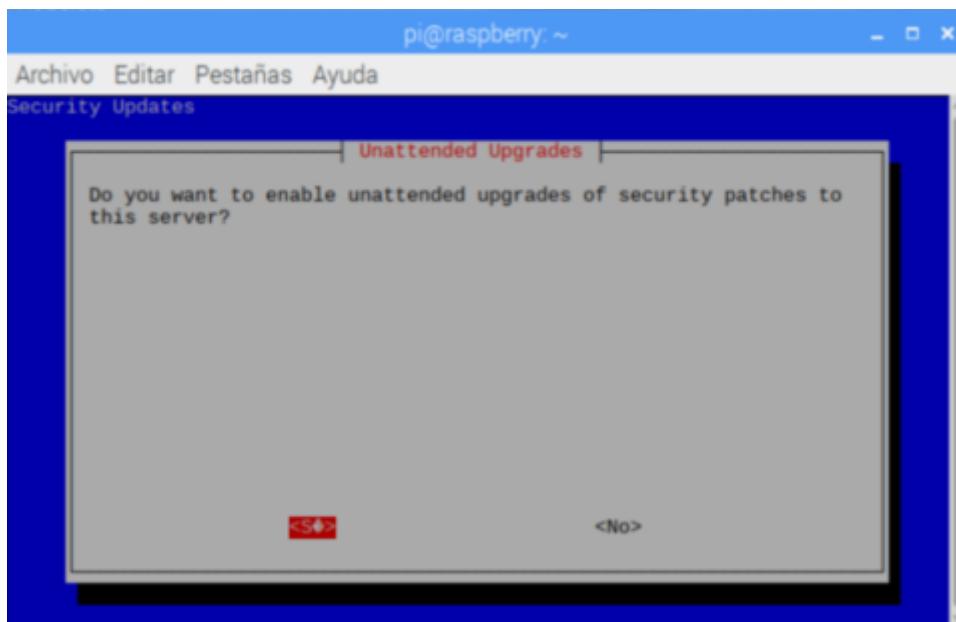


En mi caso seleccionare el usuario por defecto **pi**, tu selecciona el que más deseas.

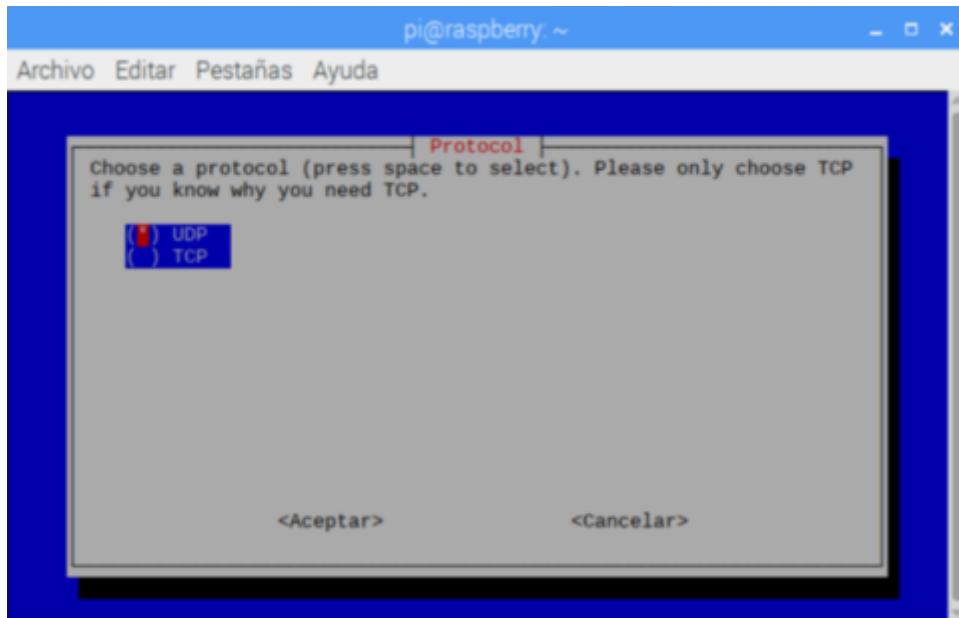
Ahora se te presentará una explicación de las actualizaciones desatendidas; esta característica hace que Raspbian descargue automáticamente las actualizaciones de paquetes de seguridad en tu Raspberry Pi diariamente. Esta configuración ayuda a proteger tu Raspberry Pi, esto es un aspecto realmente importante, ya que estarás dejando un puerto de tu router abierto, así que es altamente recomendable que permitas la instalación de los paquetes de seguridad de forma desatendida, así tu placa siempre estará actualizada a la última.



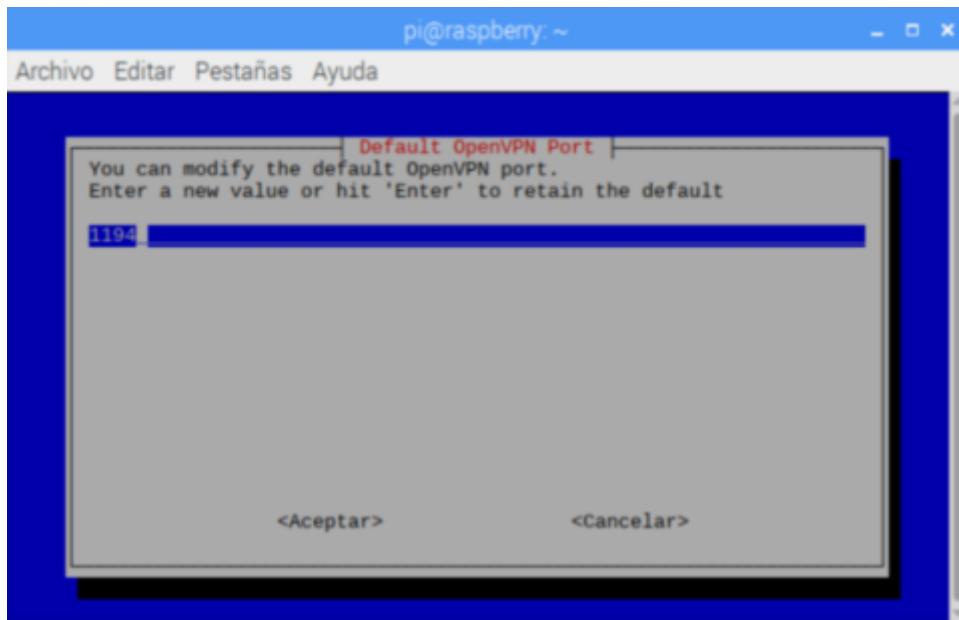
En esta pantalla, recomendamos encarecidamente seleccionar <**Sí**>. Dejar esta función desactivada puede suponer un riesgo de seguridad importante para tu Raspberry Pi y tu red doméstica.



Ahora nos pedirá que configuremos el protocolo por el cual se ejecutará OpenVPN, haremos uso de **UDP**. Solo selecciona **TCP** si sabes por qué lo necesitas. Pulsa enter con tu elección.



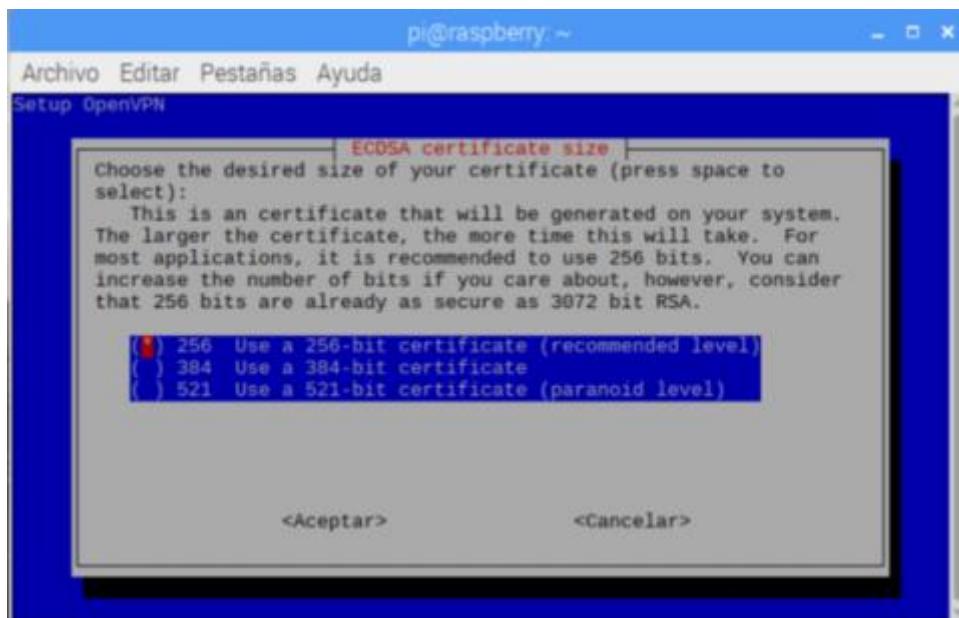
Ahora indicaremos el puerto por el que OpenVPN funcionará, y presionamos **Ok**, deja el puerto **1194** si no lo cambiaste o introduce el que configuraste para usar con OpenVPN (recuerda que, si lo vas a usar de forma normal, sería conveniente que este puerto lo cambies).



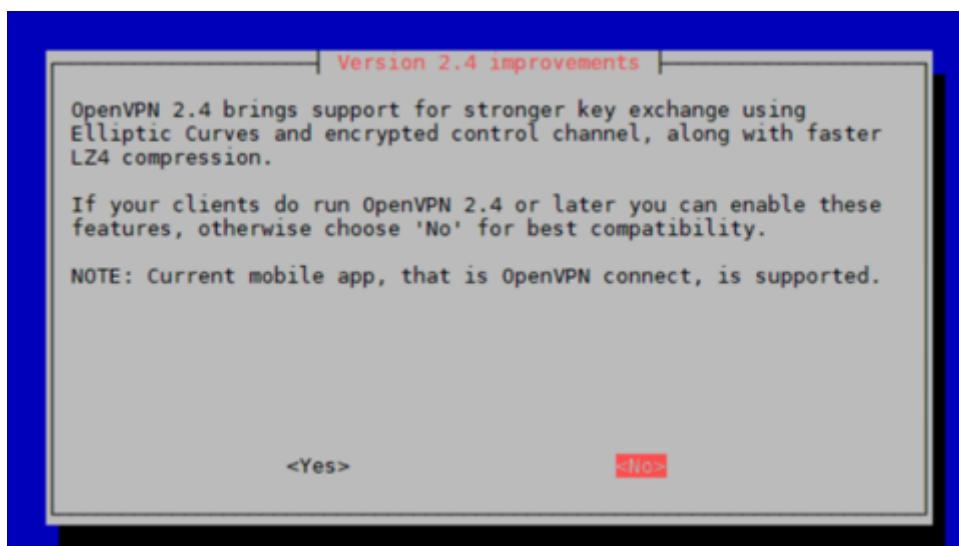
La razón para cambiarlo es que, si alguien realiza una exploración de puerto predeterminada en tu dirección IP, será mucho más difícil para ellos saber que tienes una VPN en funcionamiento.

A continuación, verás una pantalla de confirmación para el número de puerto que configuraste. Si estás satisfecho con el puerto, selecciona <Sí> para continuar.

Ahora debemos elegir el tamaño de la clave de cifrado, y recomiendo dejar el valor por defecto, usar un valor más pequeño puede reducir la seguridad de tu VPN, y usar el valor más alto puede hacer que tu conexión sea lenta debido a los procesos de encriptación con un método de cifrado demasiado seguro (aunque en este punto tu eres quien eliges).

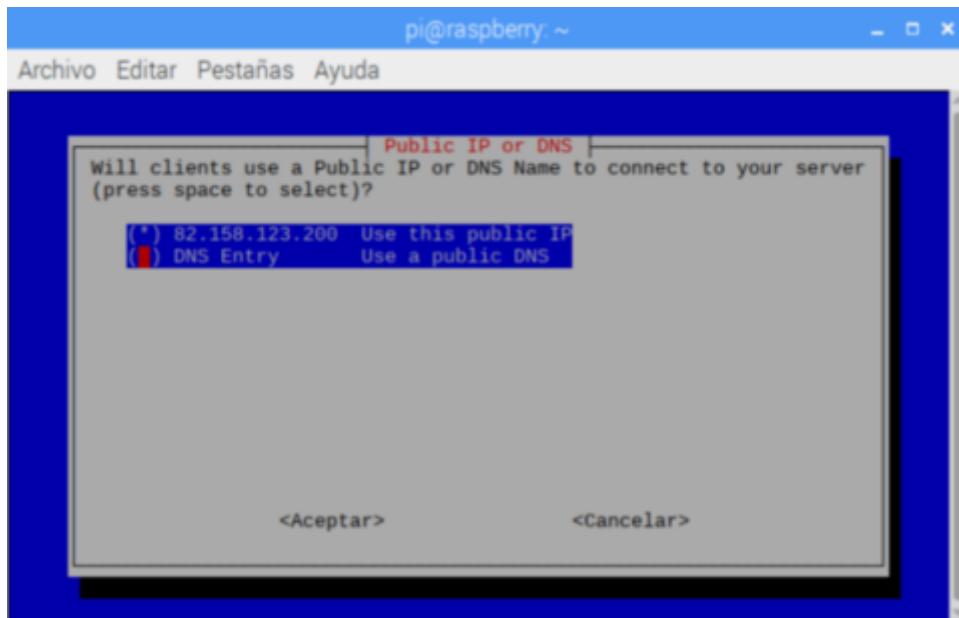


Ahora te preguntará si deseas utilizar la versión 2.4 del software OpenVPN. Si no estás seguro de si tu aplicación cliente es compatible con 2.4, Selecciona <No>, si es compatible con OpenVPN 2.4, selecciona <Sí> para asegurarte de que puedes utilizar todos sus beneficios.

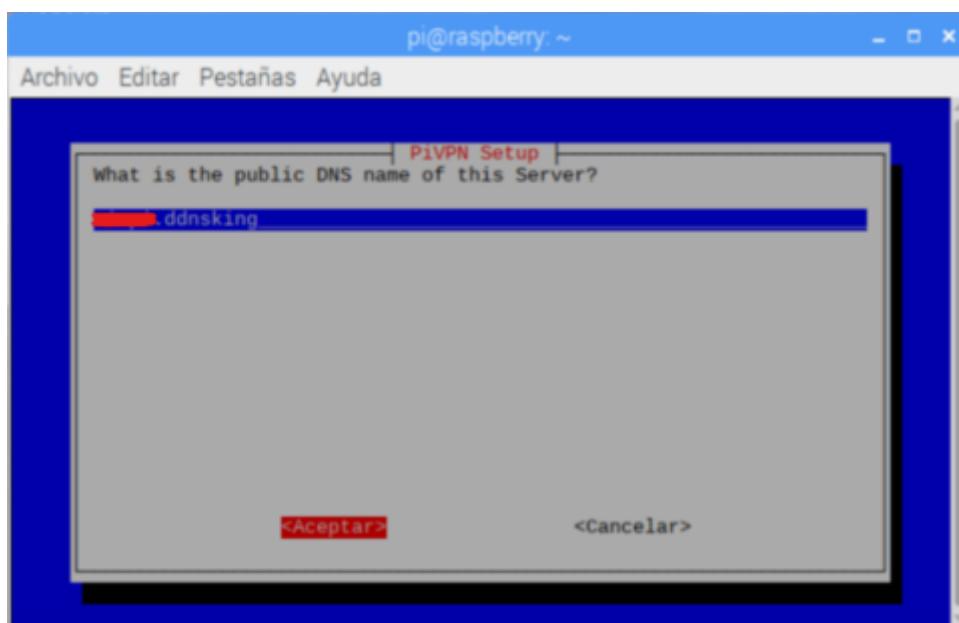


Ahora debemos decidir si queremos hacer uso de nuestra dirección IP pública o utilizar un servicio de IP dinámica como [no-ip.org](http://no-ip.org).

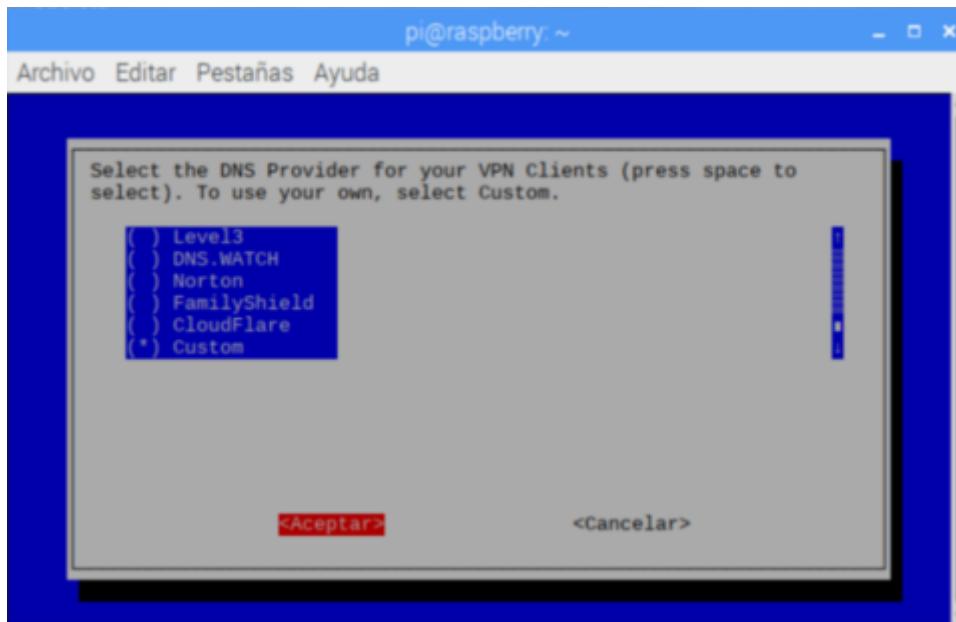
Si tiene una dirección IP dinámica, usa las teclas de flecha para navegar por el menú y usa la **barra espaciadora para seleccionar** la entrada de DNS antes de presionar Enter.



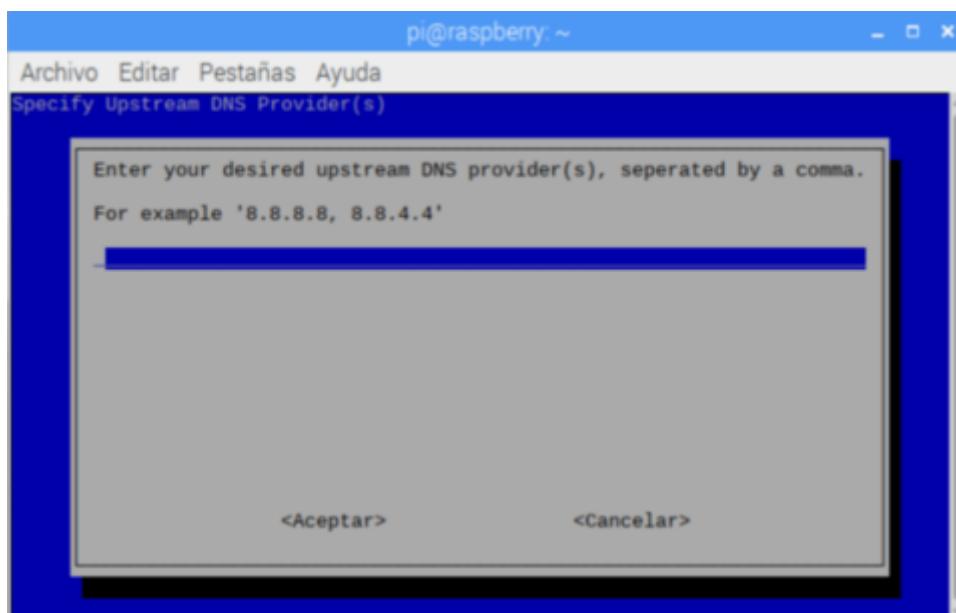
Si seleccionaste DNS, entonces puedes configurar tu nombre de DNS aquí, puede ser una dirección [xxxxx.no-ip.org](http://xxxxx.no-ip.org) o un nombre de dominio de tu propia elección que apunte a tu dirección IP. Deberías configurar esto antes de completar este tutorial.



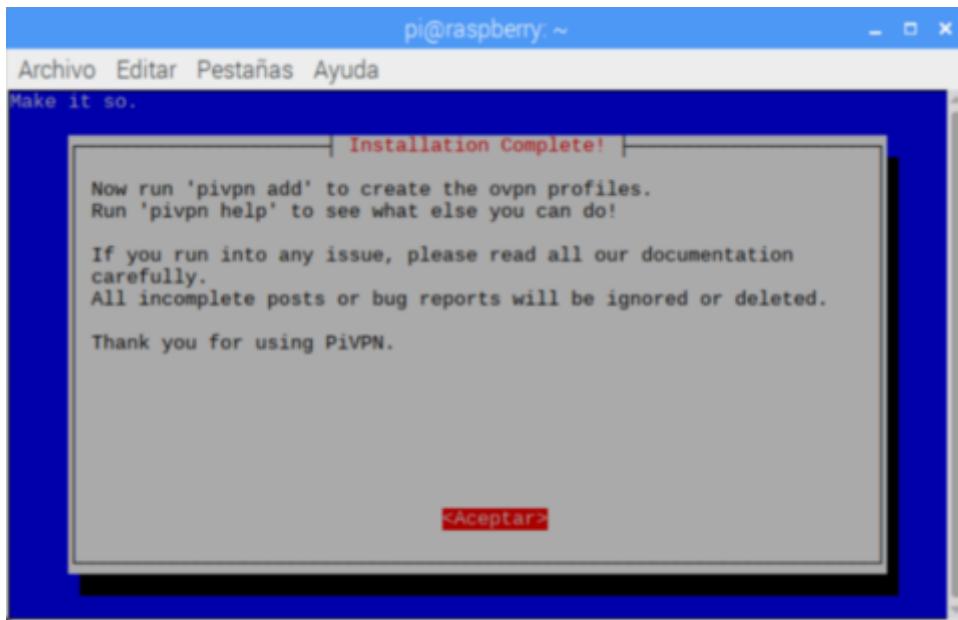
El siguiente paso es seleccionar un proveedor de DNS, puedes usar uno que configures tu mismo,



O puedes usar los de Google.



Has completado con éxito la instalación de tu VPN de Raspberry Pi, pero todavía hay un par de cosas más que deberás completar para permitir las conexiones.



Ahora nos saldrá una pantalla que nos solicita reiniciar la Raspberry Pi, selecciona **<Sí>** en las siguientes dos pantallas, ya que es crucial que se reinicie.

### COMO CONFIGURAR UN USUARIO PARA OPENVPN

Normalmente, configurar un usuario para OpenVPN sería un proceso laborioso, ya que tendríamos que generar los certificados individuales para el usuario; pero afortunadamente, podemos hacerlo con un solo comando gracias a **PiVPN**. Para comenzar a agregar el usuario, ejecuta el siguiente comando:

***sudo pivpn add***

En esta pantalla, deberás indicar un **nombre** para el cliente, y este nombre actuará como un identificador para que puedas diferenciarte entre diferentes clientes/usuarios. También te pedirá que establezcas una **contraseña** para el cliente, y es importante que sea segura y no fácil de adivinar, ya que esto asegurará la clave de cifrado. Si alguien puede adivinar tu contraseña fácilmente, reduce la seguridad de tu VPN.

```
pi@raspberry: ~
Archivo Editar Pestañas Ayuda

Using configuration from /etc/openvpn/easy-rsa/pki/safessl-easyrsa.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
commonName :ASN.1 12:'usuario1'
Certificate is to be certified until May 31 20:58:00 2022 GMT (1080 days)

Write out database with 1 new entries
Data Base Updated
Client's cert found: usuario1.crt
Client's Private Key found: usuario1.key
CA public Key found: ca.crt
tls-auth Private Key found: ta.key

one! usuario1.ovpn successfully created!
usuario1.ovpn was copied to:
/home/pi/ovpns
or easy transfer. Please use this profile only on one
device and create additional profiles for other devices.

pi@raspberry:~ $
```

Una vez que presiones Enter, el script PiVPN le dirá a Easy-RSA que genere la clave privada RSA de 2048 bits para el cliente, y luego almacena el archivo en la ruta `/home/pi/ovpns`.

En esa ruta es a la que tendremos que acceder en los próximos pasos para poder copiar el archivo que nos permitirá conectar con nuestro servidor VPN desde un equipo en el exterior de nuestra red.

Asegúrate de mantener estos archivos a salvo, ya que son tu única forma de acceder a su VPN.

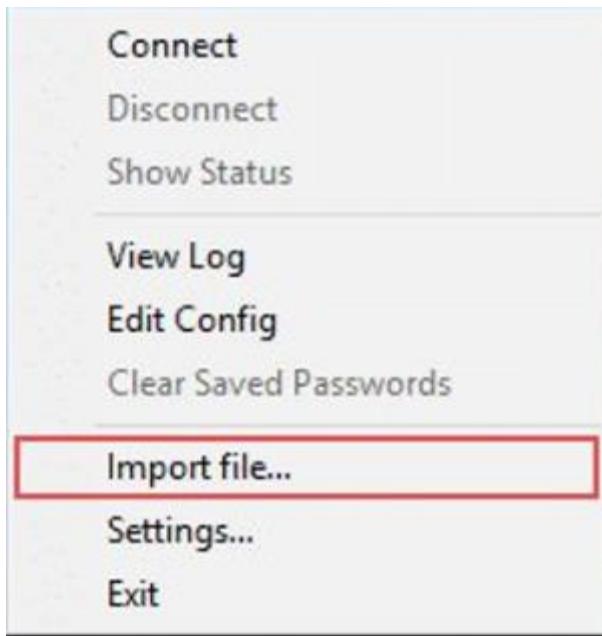
Ahora que nuestro nuevo **cliente** está configurado para OpenVPN, solo nos queda establecer la conexión con nuestro servidor VPN, para ellos, copia el archivo que mencionamos antes en el PC desde el que quieras establecer la conexión segura, accede a tu Raspberry Pi por SFTP o directamente por escritorio y ve a la ruta anterior mencionada, localiza el archivo que se ha generado y copialo, el archivo tendrá la extensión **\*.ovpn**.

Dicho archivo contiene los datos que necesitaremos para conectarnos a la VPN. También es la única forma en que alguien podría tener acceso a tu VPN, por lo que es muy importante mantener la contraseña y el archivo seguro. Si alguien tiene acceso a esto, podría causar graves problemas en tu red.

Ahora que tenemos el archivo **\*.ovpn** en nuestro PC, podemos usarlo para hacer una conexión a nuestra VPN.

El archivo **\*.ovpn** almacena todo lo que necesitamos para establecer una conexión segura. Contiene la dirección web para conectarte y todos los datos de cifrado que necesita. Lo único que no contiene es tu contraseña, por lo que deberás introducir esto cuando te conectes a la VPN.

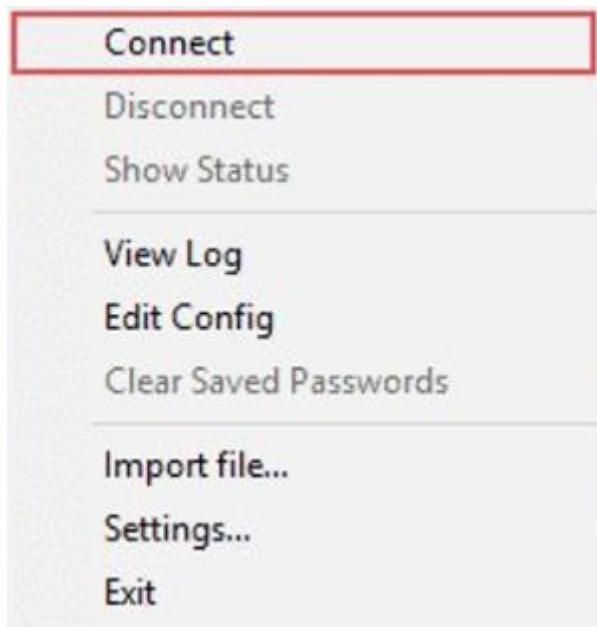
El cliente que vamos a utilizar es el cliente oficial de OpenVPN, y puedes obtenerlo en el [sitio web oficial de OpenVPN](#). Descarga e instala este cliente. En tu primera ejecución, se minimizará automáticamente en la barra de tareas, haz clic con el botón derecho en el ícono, y luego Selecciona "Importar archivo ..."



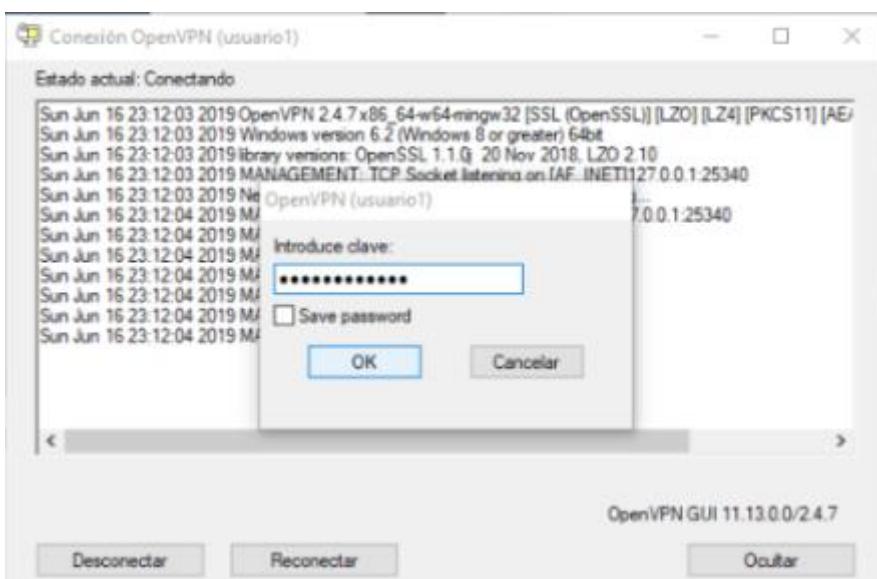
Aparecerá una pantalla de explorador de archivos, aquí vete a la carpeta donde guardaste el archivo **\*.ovpn**. Una vez que lo hayas encontrado, haz doble clic en el archivo para importarlo en el cliente OpenVPN.

Si todo fue bien, verás un mensaje en pantalla que te indicará que la importación fue correcta.

Ahora haz clic en el botón derecho en el icono del cliente OpenVPN en la barra de tareas, y luego en **Conectar**.



Ahora el cliente OpenVPN intentará leer los datos ubicados en el archivo **\*.ovpn**. Te pedirá que introduzcas la contraseña que antes configuraste. Y presiona el botón **OK**.



Ahora el cliente OpenVPN intentará conectarse al servidor VPN de tu Raspberry Pi. Si el icono de OpenVPN se vuelve verde fijo, significa que se ha conectado correctamente a tu VPN, sin embargo, si se vuelve amarillo y no se vuelve verde después de 60 segundos, significa que algo está fallando en la conexión.

# 11.- Compartir archivos WINDOWS/RPI

## 1. INSTALAR SAMBA

```
pi@raspberrypi ~$ sudo apt-get install samba samba-common-bin
```

## 2. EDITAR smb.conf

```
pi@raspberrypi ~$ sudo nano /etc/samba/smb.conf
```

Añadimos al final del fichero la carpeta que deseamos compartir  
[share]

```
comment = Share Directory
path = /srv/samba/share
browseable = Yes
writable = Yes
only guest = no
create mask = 0644
directory mask = 0755
public = no
```

La definición del directorio es como sigue,

- [share]. Es el nombre del directorio tal y como lo veremos desde nuestro cliente.
- comment. Es un simple comentario que solo te servirá a ti en principio.
- path. Es la ruta en la que se encuentra el directorio que compartimos en nuestra Raspberry Pi.
- browseable. Indica si damos permiso para navegar por los subdirectorios de nuestro directorio compartido.
- writable. Da permiso para escribir en el directorio.
- create mask y directory\_mask. Estos dos parámetros se corresponden con los permisos que se aplican a los ficheros y directorios nuevos. Recomendable indicar los valores que he puesto para evitar que cualquiera fuera de este directorio pueda modificar. Te recomiendo que leas el artículo sobre los [permisos en Linux](#) para que tengas claros los posibles valores que puedes asignar.
- public. Define si es necesaria contraseña para conectarse.
- 

## 3. CREAR USUARIO/CLAVE A UTILIZAR CUANDO COMPARTAMOS

```
pi@raspberrypi ~$ sudo adduser sdig
```

## 4. CREAR CONTRASEÑA PARA LA UTILIZACIÓN DE SAMBA

```
pi@raspberrypi ~$ sudo smbpasswd -a sdig
```

## 5. REINICIAR EL SERVICIO SAMBA

```
pi@raspberrypi ~$ sudo service smbd restart
```

## 6. DESDE WINDOWS BUSCAMOS LA RASPI

