Control luz de fondo • LIDAR • LEMP • Botón Alimentación • Kernel 4.14

# ODROID Magazine

Año Cuatro
Núm. #47
Nov 2017

## EL ULTIMO CODIGO ABIERTO PARA TU ODROID

# LINEAGE OS

## UNA COMPLETA GUIA PARA SU COMPILACION EN ODROID-XU3 Y XU4

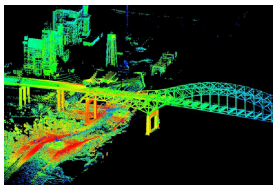
## GOOGLE ASSISTANT EN EL ODROID-XU4 Y EL NUEVO ODROID-HC1

## ANALIZANDO EL ALMACENAMIENTO POR SOFTWARE CON GLUSTERFS

## Compilando LineageOS Android: Una guía paso a paso

November 1, 2017

Esta guía te enseñará como descargar y compilar el kernel de Android para ODROID-XU3/XU4.



## LIDAR asistido por ODROID: Detección y medición lumínica con el ODROID-XU4

November 1, 2017

Este proyecto cubre la interconexión de ODROID con el LIDAR, la recepción de las mediciones de lectura y su integración en un sistema SLAM, localización y mapeo simultáneo con la finalidad de crear un mapa del área explorada

## Servidor Web LEMP: Linux, NGINX, MariaDB y PHP en el ODROID-HC1

November 1, 2017

Esta guía te permitirá montar un servidor web económico pero potente usando un ODROID-HC1 (familia XU4, incluido el ODROD-HC2) equipado con un SSD SATA.

## Linux Kernel 4.14: Soporte para ODROID-XU3/4/MC1/HC1

November 1, 2017

Esta versión del kernel pasa por ser particularmente atractiva para los ODROID Exynos 5422, especialmente porque incluye importantes correcciones para la decodificación y codificación de video por hardware (MFC), el escalador de hardware y convertidor de espacios de color (GScaler) y un nuevo driver para HDMI CEC, que permite controlar ▶️


## ODROID-C2: Apaga y enciende el sistema usando un simple botón GPIO

November 1, 2017

Este artículo describe cómo configurar un botón GPIO con vistas a apagar y encender el sistema.

## Google Assistant: Cómo utilizar un micrófono y un altavoz Bluetooth en la plataforma Ubuntu ODROID-XU4 y ODROID-HC1

November 1, 2017

Este artículo describe cómo poner en marcha un Altavoz IA sobre ODROID-HC1 utilizando Google Assistant SDK.

## Analizando el almacenamiento definido por software con GlusterFS en el ODROID-HC1: Parte 1 – Configuración del servidor

November 1, 2017

Aunque GlusterFS es claramente una tecnología destinada a empresas, ello no significa que no se pueda usar en casa

## Regular el contraluz en ODROID-VU7+: Cómo controlar la iluminación de fondo en las plataformas Android ODROID-C1 y ODROID-C2

November 1, 2017

He desarrollado un driver para controlar la luz de fondo para ODROID-C1 y ODROID-C2 usando PWM.

## Conociendo un ODROIDian: Laurent Caneiro

November 1, 2017

Conociendo un ODROIDian es una sección mensual en la que puedes conocer y aprender de las personas que disfrutan usando los productos Hardkernel.

# Compilando LineageOS Android: Una guía paso a paso

November 1, 2017 By Justin Lee Android, Tutoriales

Esta guía te enseñará como descargar y compilar el kernel de Android para ODROID-XU3/XU4. Si todavía no has compilado Android en tu ordenador de escritorio, echa un vistazo a la guía oficial de Google sobre cómo configurar un entorno de desarrollo para Android en **http://bit.ly/1Hd3Z3P**. Revísala cuidadosamente antes de continuar, de lo contrario puede que te encuetes con errores inesperados y necesites recurrir a los extensos y complejos registros log para localizar el problema.

Nosotros hemos utilizado Ubuntu 14.04 de 64 bits con 8 GB de RAM, las versiones más recientes de Ubuntu han dado algunos problemas a la hora de compilar el sistema operativo Android. **http://bit.ly/2yGT5Tw.** Si quieres compilar todo el código fuente de Android, no descargue y compiles código fuente del kernel por separado. Una compilación independiente de Kernel podría interrumpir el proceso global de compilación de Android.

### Instalar openjdk-8-jdk

Si tu entorno de desarrollo es Ubuntu 14.04 o 12.40, introduce lo siguiente:

```
$ sudo add-apt-repository ppa:openjdk-r/ppa
$ sudo apt-get update
```

Y puedes instalar openjdk-8-jdk usando apt:

```
$ sudo apt-get install openjdk-8-jdk
$ java -version
openjdk version "1.8.0_131"
OpenJDK Runtime Environment (build 1.8.0_131-
8u131-b11-2ubuntu1.16.04.3-b11)
OpenJDK 64-Bit Server VM (build 25.131-b11,
mixed mode)
```

Si tienes más de una versión de Java instalada, puedes cambiar la versión:

```
$ sudo update-alternatives --config java
$ sudo update alternatives --config javac
```

Ten en cuenta que distribuimos el kernel de Linux en diferentes sub-ramas para Android y otras distribuciones Linux.

### Plataforma Android y Kernel

El repositorio para la descarga está disponible en **http://bit.ly/1Syr1sf**. El tamaño de todo el código fuente de Android es de alrededor de unos 70GB, así que asegúrese de preparar suficiente espacio antes de empezar a desarrollar la plataforma Android:

```
$ mkdir
$ cd
$ repo init -u
https://github.com/voodik/android.git -b cm-
14.1_5422
$ repo sync
```

### Opengapps (opcional)

Para incluir Opengapps en la imagen de destino, crea el archivo "opengapps.xml" en la carpeta /.repo/local_manifests con este contenido:

```
< manifest>
< remote name="opengapps"
fetch="https://github.com/opengapps/" />
< project path="vendor/opengapps/build"
name="aosp_build" revision="master"
remote="opengapps" />
< project path="vendor/opengapps/sources/all"
name="all" clone-depth="1" revision="master"
remote="opengapps" />
< project path="vendor/opengapps/sources/arm"
clone-depth="1" revision="master"
remote="opengapps" />
</ manifest>
```

A continuación, ejecuta los siguientes comandos:

```
$ cd
$ repo sync --force-sync
```

Antes de compilar, debes configurar el ODROID-XU3 con los siguientes comandos.

```
$ ./build.sh odroidxu3
```

Una vez completado el largo proceso de compilación, puedes localizar los archivos img en el directorio "/tmp/odroidxu3/". Para usar ADB a través de una conexión TCP/IP, consulta **http://bit.ly/2gtWzAo**.

### Instalación para ODROID-XU3/XU3-Lite

Las instrucciones para instalar una imagen del kernel Linux para Android y Linux son diferentes. Como Android carga desde una partición del kernel, tenemos que usar fastboot para instalarlo en la partición dedicada. Antes de empezar, consulta la tabla de particiones en **http://bit.ly/2irbjnC**. En cambio, Linux arranca teniendo en cuenta los parámetros incluidos en boot.ini en la primera partición FAT.

En primer lugar, instala la imagen kernel zImage-dtb, en la tabla de arranque:

```
$ sudo fastboot flash kernel
<path/of/your/zImage-dtb>
```

Después, instala los archivos de la plataforma android system.img, userdata.img y cache.img:

```
$ sudo fastboot flash system
<path/of/your/system.img> $ sudo fastboot
flash userdata <path/of/your/userdata.img> $
sudo fastboot flash cache
<path/of/your/cache.img>
```

Finalmente, activa la partición FAT:

```
$ sudo fastboot erase fat
```

### Instalación para XU4

Puedes evitar el tema de usar fastboot en un ODROID-XU4, ya que éste no dispone de ningún puerto USB OTG. Primero, configura una conexión ADB o copia una imagen a la partición FAT:

```
$ adb push xxxx.img /sdcard/
$ adb reboot
```

Echa un vistazo a los registros log del U-Boot conectándote al ODROID-XU4 con un kit USB-UART:

```
U-Boot 2017.05-12209-g43745f3 (Aug 17 2017 -
09:37:39 +0900) for ODROID-XU4

CPU: Exynos5422 @ 800 MHz
 Model: Odroid XU4 based on EXYNOS5422
 Board: Odroid XU4 based on EXYNOS5422
 Type: xu3
 DRAM: 2 GiB
 MMC: EXYNOS DWMMC: 0, EXYNOS DWMMC: 1
```

```
 MMC Device 0 (eMMC): 14.7 GiB
 Info eMMC rst_n_func status = enabled
 MMC Device 1 ( SD ): 7.4 GiB

*** Warning - bad CRC, using default
environment

In: serial
 Out: serial
 Err: serial
 Net: No ethernet found.
 Press quickly 'Enter' twice to stop autoboot:
0
```

Escribe los siguientes comandos tras presionar “Intro” dos veces para pausar el proceso de arranque:

```
Exynos5422 # ext4load mmc 0:3 40000000
media/0/system.img
379342968 bytes read in 13284 ms (27.2 MiB/s)
Exynos5422 # fastboot flash system 40000000 0

*** Partition Information for Android ***
Control Device ID : 0
pNo     Start Block     Block Count     pName
 0              1              30        fwbl1
(15 KB)
 1             31              32        bl2
(16 KB)
 2             63            1440
bootloader (720 KB)
 3           1503             512        tzsw
(256 KB)
 4           2015              32        env
(16 KB)
 5           2047           16384        kernel
(8192 KB)
 6        2752512          204800        fat
(102400 KB)
 7         131072         2097152        system
(1048576 KB)
 8        2957312        27688960
userdata (13844480 KB)
 9        2228224          524288        cache
(262144 KB)

Erasing partition(system)...  blk_st = 131072,
blk_cnt = 2097152
*** erase block start 0x20000, cnt 0x200000
***
write_compressed_ext4 : total chunk = 1373
mmc write dev 0, blk = 0x00020008, size =
0x00000008, remain chunks = 1372
mmc write dev 0, blk = 0x00020010, size =
0x00000008, remain chunks = 1371

...

mmc write dev 0, blk = 0x00160010, size =
0x00000008, remain chunks = 10
none chunk
mmc write dev 0, blk = 0x00160208, size =
0x000001f8, remain chunks = 9
mmc write dev 0, blk = 0x00160218, size =
0x00000010, remain chunks = 8
none chunk
mmc write dev 0, blk = 0x001a0000, size =
0x0003fde8, remain chunks = 7
mmc write dev 0, blk = 0x001a0010, size =
0x00000010, remain chunks = 6
none chunk
mmc write dev 0, blk = 0x001e0000, size =
0x0003fff0, remain chunks = 5
mmc write dev 0, blk = 0x001e0008, size =
0x00000008, remain chunks = 4
mmc write dev 0, blk = 0x001e0010, size =
0x00000008, remain chunks = 3
none chunk
mmc write dev 0, blk = 0x001e0208, size =
0x000001f8, remain chunks = 2
mmc write dev 0, blk = 0x001e0218, size =
0x00000010, remain chunks = 1
none chunk
mmc write dev 0, blk = 0x00220000, size =
0x0003fde8, remain chunks = 0
write done

partition 'system' flashed.

Exynos5422 #
```

### ODROID Updater

Puedes actualizar desde Android Kitkat 4.4.4 estándar a Android 7.1 utilizando la versión 5.6 o superior de ODROID Updater (**http://bit.ly/2yFz9lf**). Siempre marca la opción “Format userdata” al actualizar desde otra versión de Android, tal y como se muestra en la Figura 1.

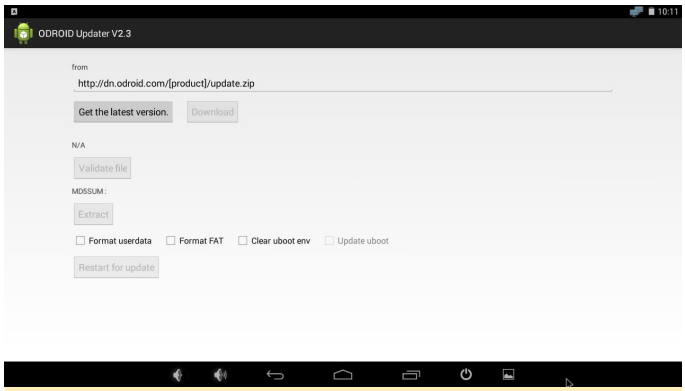



**Figura 1 – Usando ODROID Updater**

```
$ adb connect 192.168.x.x
$ cd out/target/products/odroidxu3/
$ adb push update.zip /sdcard/
$ adb push update.zip.md5sum /sdcard/
```

Abre “Internal storage”, tal y como se muestra en la Figura 2, luego copia update.zip y update.zip.md5sum como se muestra en la Figura 3. Activa el botón “Validate file” como muestra la Figura 4.

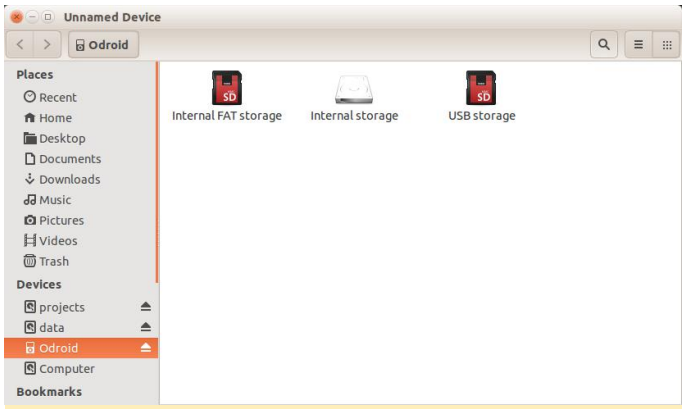



**Figura 2 – Abriendo el almacenamiento interno en LineageOS**

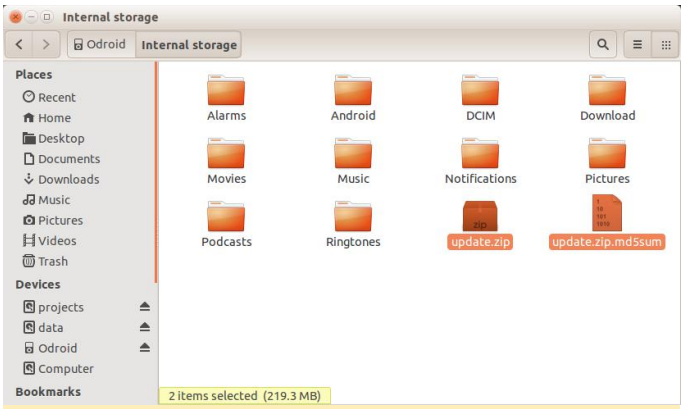



**Figura 3 – Seleccionando los archivos update.zip y update.zip.md5sum**

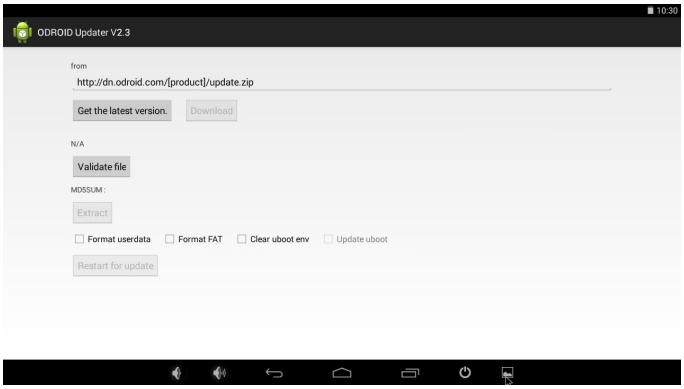



Figura 4 – Validando los archivos de actualización en

**ODROID Updater**

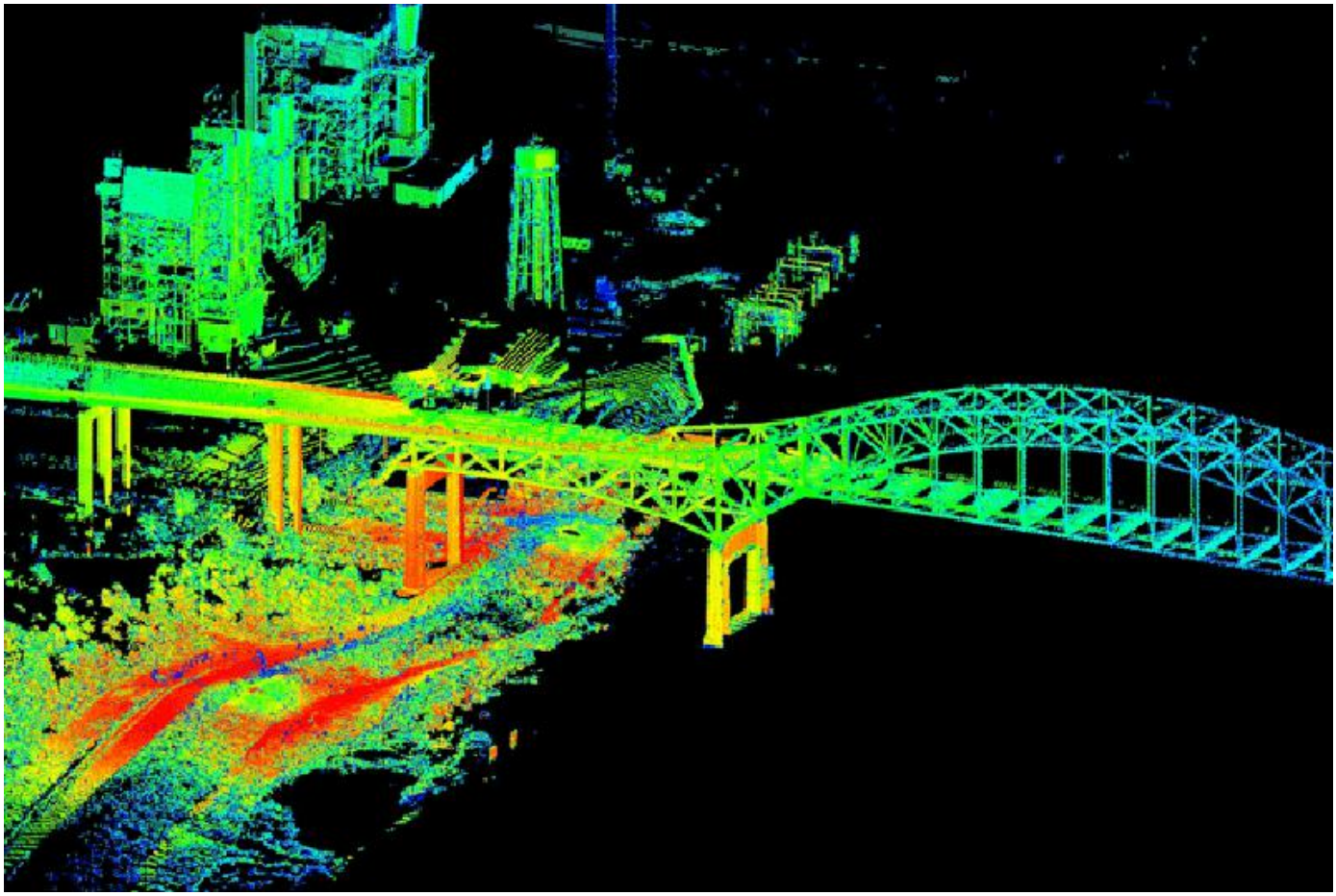
### Información del Almacenamiento

/storage/sdcard0 partición FAT del almacenamiento interno eMMC o microSD /storage/sdcard1 ranura microSD para Tarjeta SD /storage/usb2host Almacenamiento USB puerto USB 2.0 host /storage/usb3host Almacenamiento USB puerto USB 3.0 host /storage/usb3device Almacenamiento USB puerto USB 3.0 Dispositivo

Para comentarios, preguntas y sugerencias, visita el artículo original en **http://bit.ly/2yUeqvm**.

# LIDAR asistido por ODROID: Detección y medición lumínica con el ODROID-XU4

November 1, 2017 By Tom Jacobs ODROID-XU4, Mecaniqueo



Hace poco encargué un ODROID-XU4 para poner en marcha mi último proyecto, un Earth Rover, un robot explorador de 6 ruedas que puede desplazarse tanto por áreas internas como externas de forma autónoma. Estoy utilizando un sistema LIDAR extraído de una vieja aspiradora robotizada, la Neato XV-11 LIDAR. Este proyecto cubre la interconexión de ODROID con el LIDAR, la recepción de las mediciones de lectura y su integración en un sistema SLAM, localización y mapeo simultáneo con la finalidad de crear un mapa del área explorada. Elegí ODROID porque era la única placa lo suficientemente potente como para procesar las mediciones sobre un mapa en tiempo real.

Tiene ocho núcleos, cuatro se ejecutan a 2Ghz y los cuatro a 1.6Ghz. ¿Recuerdas tu famoso Pentium 133Mhz? Esta placa es quince veces más rápida y lo hace cuatro veces al mismo tiempo. Además, cuenta con otros cuatro núcleos simplemente por pura diversión y se puede introducir dentro de una taza de café. ¿Cuánto cuesta? 59$. Conéctale el cable HDMI, la fuente de alimentación, el teclado y el ratón USB, el adaptador wifi USB y enciéndelo. En primer lugar, ejecuta los siguientes comandos para actualizar completamente tu sistema:

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

Gracias al conjunto de instrucciones de multiprocesamiento NEON, el rendimiento es bastante bueno, **http://bit.ly/2zG12Ix**, para tareas específicas como SLAM, incluso comparandolo con la potente Raspberry Pi, tal y como se muestra en la Figura 1.

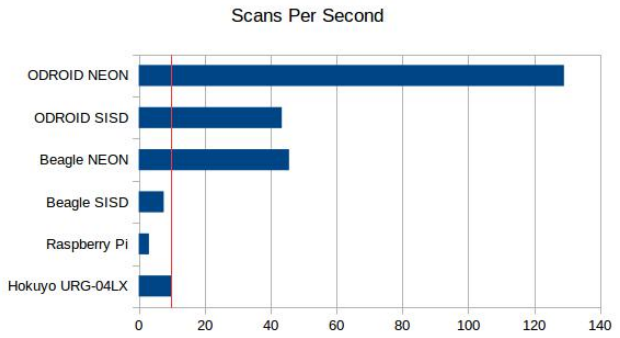



**Figura 1 – Comparando las lecturas por segundo de varios dispositivos SBC**

Primero, clona el sistema de instalación e inícialo:

```
$ sudo apt-get install git
$ git clone https://github.com/tjacobs/betabot
$ cd betabot/install
$ ./install
```

Ahora que tenemos muchas y muy buenas aplicaciones, como Python, PIP, OpenCV, FFMPEG y Alsa, vamos a instalar BreezySLAM:

```
$ git clone
https://github.com/simondlevy/BreezySLAM
$ cd BreezySLAM/python
$ sudo python setup.py install
$ cd ../examples
$ make pytest
```

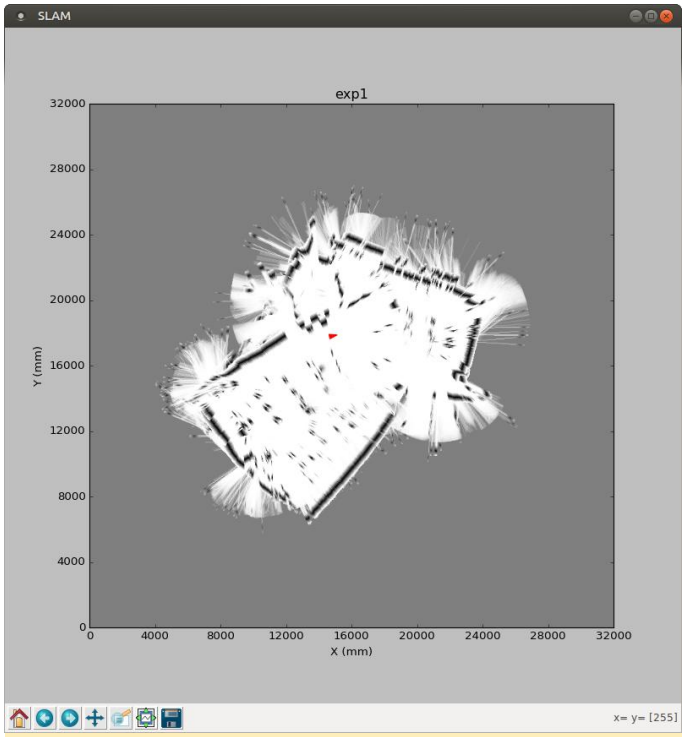



**Figura 2 – Una imagen preliminar de una habitación SLAM usando LIDAR**

Tal y como se muestra en la Figura 2, ¡Estamos viendo una sala mapeada con SLAM! desde el propio archivo de datos del sistema, de modo que aún no disponemos de ningún LIDAR físico externo. Vamos intentar ver los objetos en vivo, lo cual requiere matplotlib:

```
$ sudo apt-get install python-matplotlib
$ make movie
```

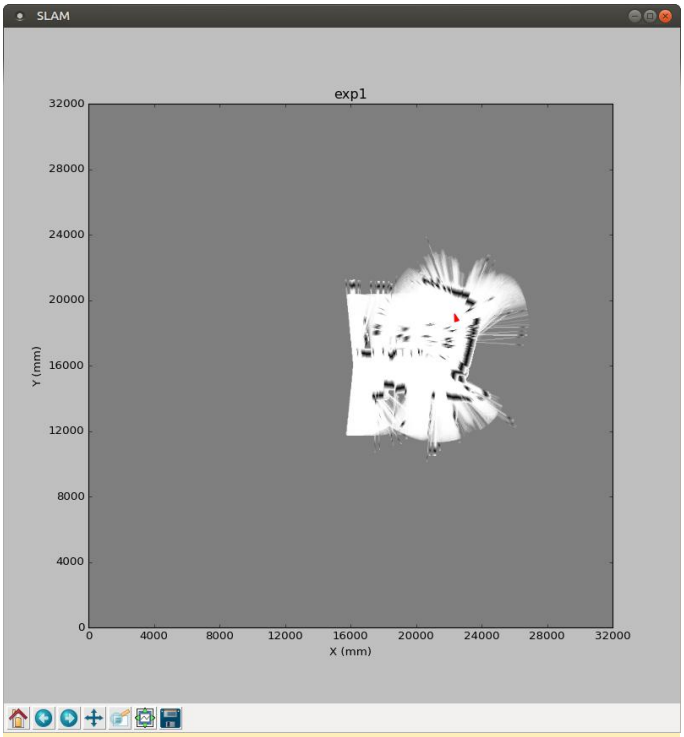



**Figura 3 – Película en vivo de una habitación SLAM usando LIDAR**

Tal y como muestra la Figura 3, podemos ver cómo se ejecuta y se va generando el mapa. El siguiente paso es probar LIDAR, que requiere xvlidar, pip y pyrserial:

```
$ git clone
https://github.com/simondlevy/xvlidar
$ sudo python setup.py install
$ python lidarplot.py
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo python get-pip.py
$ sudo pip install pyserial
```

El resultado ahora es:

```
SerialException could not open port
/dev/ttyACM0
```

Puesto que está intentando acceder al puerto serie, podemos verificar el dispositivo:

```
$ ls /dev/tty*
$ /dev/ttyACM99
```

Resulta que el dispositivo está realmente en otro puerto:

```
$ /dev/ttySAC0
```

Para saber dónde conectar el LIDAR, podríamos consultar el manual de ODROID-XU4 en **http://bit.ly/2xYZhco**, pero el caso es que no se menciona. La información que necesitamos está en **http://bit.ly/2hYj8NQ**, la Figura 4 muestra los pines que estamos buscando.

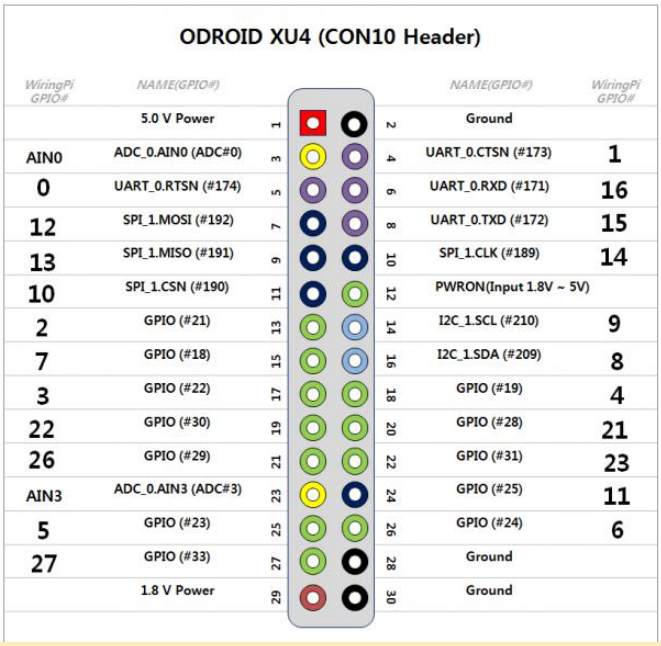



**Figura 4 – Configuración GPIO para un proyecto LIDAR en el ODROID-XU4**

Los pines de recepción y transmisión son UART_0.RXD (#6) y UART_0.TXD (#8). Tenemos también la puesta a tierra (#2) y la alimentación de 5v (#1). La mayoría de los sensores Lidars XV son a 5v, tal y como se menciona en **http://bit.ly/2gBSiPc**. Utilice el pin de 1.8v para alimentar el motor LIDAR, que como era de esperar proporciona suficiente amperaje. Tras ejecutar lidarplot.py nuevamente, se obtiene un LIDAR giratorio y un gráfico en blanco con un único punto, suele aparecer "Checksum fail" alguna vez que otra cuando se agitan las clavijas. De modo que se consigue algo, pero nada útil todavía, tal y como muestra la Figura 5.

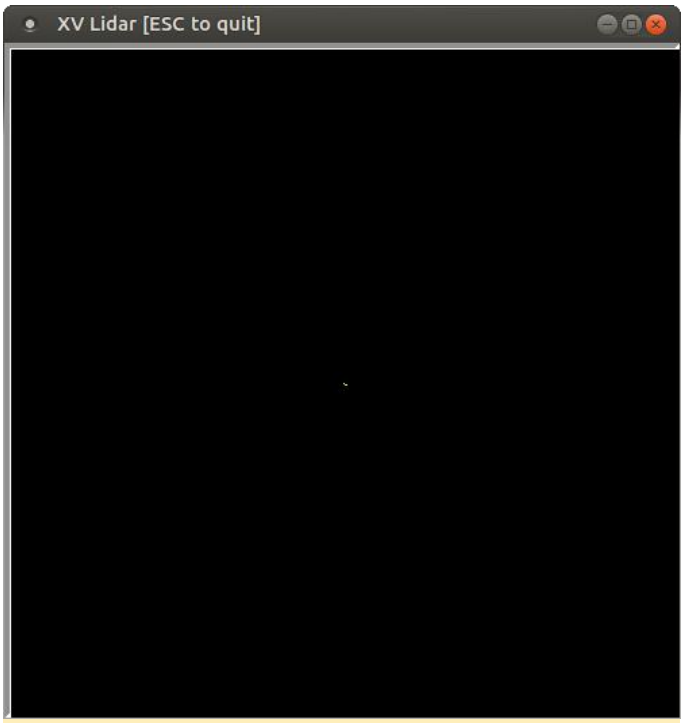



**Figura 5 – Resultado inicial de la configuración LIDAR en el ODROID-XU4**

Después de añadir algunas impresiones, observo que el puerto serie generaba muchos 82 a lo largo de la matriz de 360 grados. Supongo que eso significa "Not spinning fast enough", porque utilicé una Raspberry Pi 2 para generar el voltaje correcto a 3.3v, y fui capaz de conseguir lecturas LIDAR. También se actualiza muy rápido, ya que cuando colocaba mi mano en frente, el cambio quedaba reflejado en un segundo más o menos.

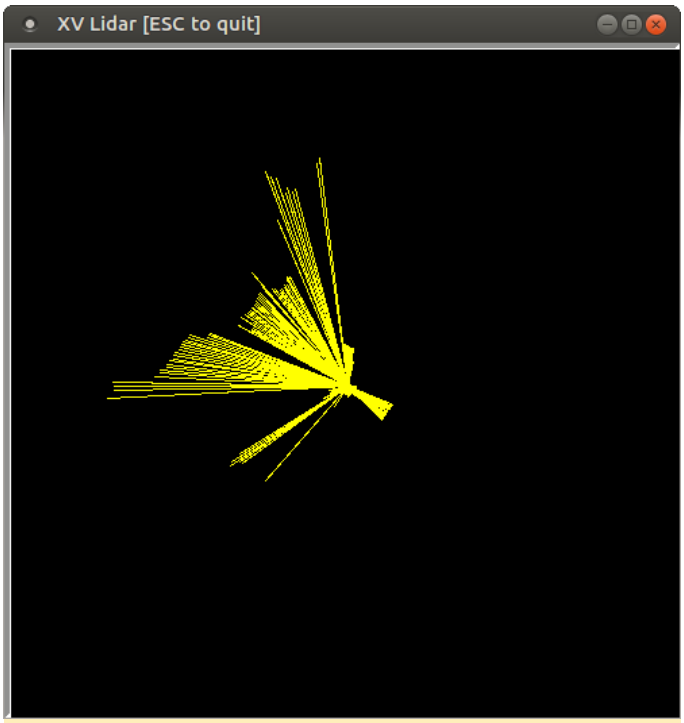



**Figura 6 – Resultado adaptado usando un motor 3.3v de la configuración LIDAR en el ODROID-XU4**

Ahora estamos listos para ejecutar la aplicación SLAM:

```
$ cd BreezySLAM/examples
$ python xvslam.py
```

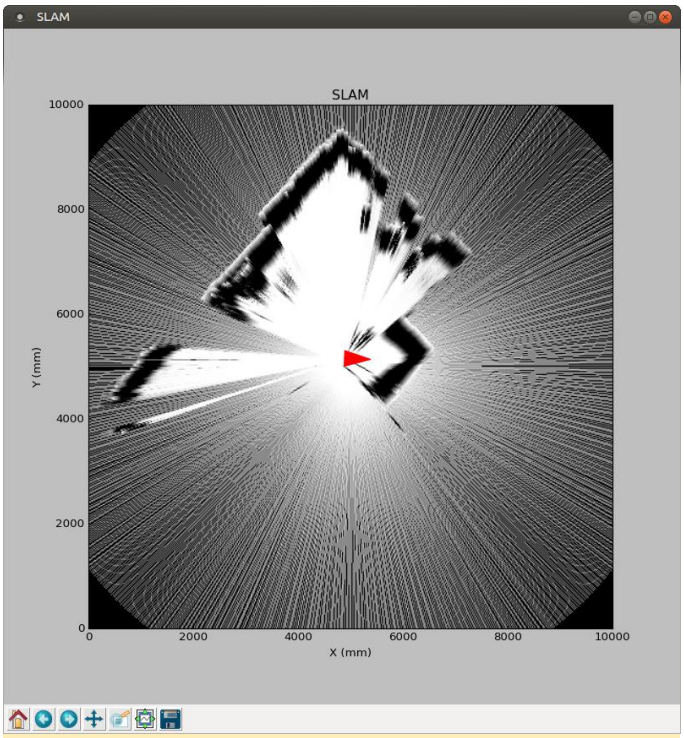



**Figure 7 – SLAM sobre el ODROID-XU4**

Para comentarios, preguntas y sugerencias, visita el artículo original en: http://bit.ly/2xX6ObL.

# Servidor Web LEMP: Linux, NGINX, MariaDB y PHP en el ODROID-HC1

November 1, 2017 By Justin Lee Linux, ODROID-HC1, ODROID-HC2


Esta guía te permitirá montar un servidor web económico pero potente usando un ODROID-HC1 (familia XU4, incluido el ODROID-HC2) equipado con un SSD SATA.

### Preparar el soporte de arranque

Antes de continuar, graba la última imagen oficial de Hardkernel Ubuntu Minimal en tu soporte de arranque: tarjeta microSD de 8GB + Clase 10. También puedes utilizar un módulo eMMC de Hardkernel, junto con un lector de tarjetas microSD USB3.

Visita al enlace **http://bit.ly/2xaucO8** y descárgate la última imagen de Ubuntu Minimal. Además, accede al enlace **https://etcher.io/** y descarga la versión de Etcher correspondiente a tu sistema operativo. Inserta la tarjeta microSD en tu ordenador y ejecuta Etcher, luego graba la imagen en tu microSD.

### Configurar tu ODROID

Instala el SSD usando el puerto SATA en su ODROID-HC1. Asegúrate de utilizar la fuente de alimentación oficial 5V 4A+ de Hardkernel para garantizar que el SSD disponga de suficiente potencia. Una vez hecho esto, inserta el soporte de arranque ya preparado en tu ODROID y enciende el ODROID-HC1.

El primer arranque suele tardar unos 5 minutos en iniciar el sistema operativo. Si no se enciende tras aproximadamente 5 minutos, puedes probar a reiniciarlo, desconectando/conectando de nuevo el cable de alimentación y volviéndolo a encender.

Ten en cuenta que si utilizas un ODROID-XU4, puedes llegar a montar un NAS de alto rendimiento usando un hub USB con alimentación, un SSD/eMMC para el sistema operativo y uno o más discos duros para el NAS. Los hubs alimentados por bus pueden que no

funcionen correctamente debido a una alimentación insuficiente.

### Acceso SSH y actualización del sistema

Conéctate a tu ODROID-HC1 a través de SSH y empieza a darle forma a tu servidor web. Se recomienda encarecidamente actualizar la imagen de la tarjeta microSD. Eso te permitirá beneficiarte de las últimas correcciones y del posible soporte para funciones adicionales. Tras acceder por SSH, actualiza Ubuntu y el kernel con los siguientes comandos:

```
$ sudo apt-get update && sudo apt-get dist-upgrade
$ sudo apt-get install linux-image-xu3
$ reboot
```

### Convertir tu SSD en una partición root

Los soportes de arranque, como las tarjetas microSD, constituyen un sistema ineficiente porque las tarjetas microSD son lenta (para el sistema operativo y las aplicaciones que se ejecuten en ellas, como un servidor web) y están sujeta a fallos tras sucesivas escrituras. Aunque un módulo eMMC es una opción viable para un ODROID-XU4, éste no está disponible para el ODROID-HC1. Por todo ello, la instalación y uso de un SSD es muy recomendable para alojar sitios web que utilicen bases de datos. Para utilizar eficientemente un SSD para el arranque y las aplicaciones, sigue la guía paso a paso que se presenta a continuación, la cual te ayudará a preparar tu SSD con una partición root. Puedes consultar el post del foro en http://bit.ly/2gpT9OR para más detalles.

### Re-particionar tu SSD

En primer lugar, debes particionar tu SSD para poder usarlo con dos particiones: una como partición root para el sistema operativo y la otra para los datos. Puedes obtener información sobre tu SSD usando la herramienta fdisk:

```
$ sudo fdisk -l
# results
...
Disk /dev/sda: 111.8 GiB, 120034123776 bytes, 234441648 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 33553920 bytes
Disklabel type: gpt
Disk identifier: 0412F7EC-4E8C-4610-ABFF-D6293387ADB6
```

Para particionar el SSD, utiliza la herramienta fdisk con el nombre correcto del dispositivo (/dev/sda tal y como aparece en el resultado del comando anterior):

```
$ sudo fdisk /dev/sda
```

Las opciones fdisk más útiles se detallan a continuación:

- p : Mostrar la tabla de particiones
- n : Añadir una nueva partición
- d : Eliminar una partición
- w : Escribir la tabla en el disco y salir
- q : Salir sin guardar los cambios
- g : Crear una nueva tabla de particiones GPT vacía
- m : Ayuda (menú)

Al mismo tiempo que consultas el menú anterior, elimina las particiones actuales, si las hay, y crea una nueva tabla de particiones GPT. Después crea una nueva partición para la partición root y otra para los datos. En nuestro caso, la partición root tendrá una capacidad de 16G y el resto de espacio irá para la partición de datos. Puedes especificar un tamaño de la partición especifico escribiendo una capacidad, por ejemplo, “+16G”.

Revisa el resultado de tus acciones obteniendo información de las particiones con el parámetro “p”. Si esta información coincide con tus preferencias, presiona “w” para guardar y salir.

```
# In fdisk
Command (m for help): p
Disk /dev/sda: 111.8 GiB, 120034123776 bytes, 234441648 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 33553920 bytes
Disklabel type: gpt
```

```
Disk identifier: 0412F7EC-4E8C-4610-ABFF-
D6293387ADB6

Device        Start       End   Sectors  Size
Type
/dev/sda1      2048  33556479  33554432   16G
Linux filesystem
/dev/sda2  33556480 234441614 200885135 95.8G
Linux filesystem
```

**Formatear y montar una partición EXT4**

Los modernos sistemas Linux generalmente utilizan el sistema de archivos EXT4, de modo que es muy aconsejable crear particiones de tipo ext4:

```
$ sudo mkfs.ext4 /dev/sda1
$ sudo mkfs.ext4 /dev/sda2
```

Una vez hecho esto, debes montar las particiones en directorios específicos para poder usar tu SSD. Crea los nuevos directorios para tu SSD:

```
$ sudo mkdir -p /media/systemdrive
$ sudo mkdir -p /media/data
```

Usaremos /media/systemdrive como partición root y /media/data como partición de datos:

```
$ sudo mount /dev/sda1 /media/systemdrive
$ sudo mount /dev/sda2 /media/data
```

Luego, verifica que estén montadas correctamente:

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted
on
...
/dev/sda1        16G   44M   15G   1%
/media/systemdrive
/dev/sda2        95G   60M   90G   1%
/media/data
```

El siguiente paso es modificar los archivos relacionados con el sistema para arrancar desde tu SSD.

**Modificando boot.ini**

Primero, consulta el UUID de root:

```
$ sudo lsblk -f
# results
NAME        FSTYPE LABEL  UUID
MOUNTPOINT
mmcblk1
|-mmcblk1p1 vfat   boot   52AA-6867
/media/boot
`-mmcblk1p2 ext4   rootfs e139ce78-9841-40fe-
8823-96a304a09859 /
sda
|-sda2      ext4          6553d8f1-6224-450f-
aec1-3b6f5fc09bd0 /media/data
`-sda1      ext4          f00945e6-46ea-47db-
893a-6a74548c3af7 /media/systemdrive
```

Toma nota del UUID para /media/systemdrive, luego cambia el UUID del sistema de archivos root en boot.ini para que tu gestor de arranque reconozca la partición del SSD como partición root:

```
$ sudo cp /media/boot/boot.ini
/media/boot/boot.ini.bak
$ sudo vi /media/boot/boot.ini
```

Busca la expresión “Basic Ubuntu Setup” en el archivo:

```
...
# Basic Ubuntu Setup. Don't touch unless you
know what you are doing.
# --------------------------------
setenv bootrootfs "console=tty1
console=ttySAC2,115200n8 root=UUID=e139ce78-
9841-40fe-8823-96a304a09859 rootwait ro
fsck.repair=yes net.ifnames=0"
…
```

Cambia el UUID de root para que coincida con el valor anterior. Ten en cuenta que tus valores UUID pueden ser diferentes a los que se aparecen aquí.

**Actualizando fstab**

Para montar tus particiones automáticamente, añade las entradas necesarias en el archivo /etc/fstab.

```
$ sudo vi /etc/fstab
UUID=e139ce78-9841-40fe-8823-96a304a09859 /
ext4 errors=remount-ro,noatime 0 1
LABEL=boot /media/boot vfat defaults 0 1
```

Comenta la primera línea y agrega nuevas líneas, tal y como se muestra a continuación:

```
#UUID=e139ce78-9841-40fe-8823-96a304a09859 /
ext4 errors=remount-ro,noatime 0 1
```

```
LABEL=boot /media/boot vfat defaults 0 1
/dev/sda1 / ext4 defaults,noatime 0 1
/dev/sda2 /media/data ext4 defaults 0 2
```

**Copiar un partición root**

Copia la partición root de microSD al SSD usando la utilidad rsync:

```
$ sudo apt-get install rsync
$ sudo rsync -axv / /media/systemdrive
```

Una vez finalizado el proceso de copia, estarás listo para pasar al siguiente paso.

**Verificar las particiones**

Reinicia tu ODROID-HC1 y comprueba si la partición root montada es visible en tu SSD:

```
$ lsblk -f
NAME        FSTYPE LABEL  UUID
MOUNTPOINT
mmcblk1
|-mmcblk1p1 vfat   boot   52AA-6867
/media/boot
`-mmcblk1p2 ext4   rootfs e139ce78-9841-40fe-
8823-96a304a09859
sda
|-sda2      ext4          daff1faa-3895-46cb-
896f-bfe67f78535e /media/data
`-sda1      ext4          07ac0233-7d4a-49ac-
baf0-4a4ebd07741c /
```

Tal y como aparece arriba, el MOUNTPOINT del sda1 es "/", lo que significa que el sistema arrancó correctamente desde el SSD.

**Servidor LEMP (Linux, NGINX, MariaDB, PHP)**

Hemos optado por utilizar nginx como servidor web. Éste utiliza una técnica asincrónica y orientada a eventos para manejar las conexiones, es rápido y tiene capacidad para atender solicitudes de muchos usuarios, además de ofrecer un rendimiento bastante fiable. Ha sido diseñado para ser un software liviano, el cual ofrece muchas características. Si deseas contar con la ventaja de poder usar funciones como es la instalación de módulos adicionales, decantarse por Apache sería una mejor opción.

**PHP**

Para instalar PHP, debes añadir un repositorio para PHP de antemano. Puedes instalar el último PHP para ARM, versión 7.1 o superior.

```
$ sudo add-apt-repository ppa:ondrej/php
$ sudo apt-get update && sudo apt-get install
php7.1-fpm
```

Una vez finalizada la instalación, debes cambiar la zona horaria especificada en un archivo de configuración de PHP.

```
$ sudo vi /etc/php/7.1/fpm/php.ini
```

Busca "date.timezone" y cámbiala de acuerdo a tu ubicación, inicialmente puede estar comentada por defecto.

**MariaDB**

No hay disponible un PPA oficial para MariaDB (**http://bit.ly/2zktcMs**) basado en la arquitectura ARM. Simplemente debes instalarlo desde el repositorio que se proporciona en Ubuntu. Además, debes instalar el paquete php-mysql para enlazar MariaDB con PHP. La instalación de los paquetes necesarios podría demorarse un poco.

```
$ sudo apt-get update && sudo apt-get install
mariadb-server mariadb-client php-mysql
```

Deberías configurar el conjunto de idiomas que MariaDB usa para UTF-8.

```
$ sudo vi /etc/mysql/conf.d/mysql.cnf
```

Elimina todo el contenido existente y copia-pega el nuevo contenido que aparece a continuación.

```
# MariaDB-specific config file.
# Read by /etc/mysql/my.cnf

[client]
# Default is Latin1, if you need UTF-8 set
this
# (also in server section)
default-character-set = utf8mb4

[mysqld]
#
# * Character sets
#
# Default is Latin1, if you need UTF-8 set all
```

```
this
# (also in client section)
#
character-set-server = utf8mb4
collation-server = utf8mb4_unicode_ci
character_set_server = utf8mb4
collation_server = utf8mb4_unicode_ci
```

Finalmente, reinicia el servicio MariaDB.

```
$ sudo service mysql restart
```

### Instalar nginx

Para instalar nginx, debes añadir un repositorio para nginx de antemano. Puedes instalar la última versión para ARM (versión 1.12+):

```
$ sudo add-apt-repository ppa:nginx/stable
$ sudo apt-get update && sudo apt-get install
nginx
```

Si quieres usarlo con PHP, tienes que modificar la configuración del servidor:

```
$ sudo mv /etc/nginx/sites-available/default
/etc/nginx/sites-available/default.bak
$ sudo vi /etc/nginx/sites-available/default
```

Coloca lo que aparece a continuación dentro del nuevo archivo de servidor por defecto.

```
server {
  listen 80 default_server;
  listen [::]:80 default_server;

  root /var/www/html;

  index index.html index.php;

  server_name _;

  location / {
    try_files $uri $uri/ =404;
  }

        # This option is important for using
PHP.
  location ~ \.php$ {
    include snippets/fastcgi-php.conf;
    fastcgi_pass unix:/var/run/php/php7.1-
fpm.sock;
  }
}
```

Reinicia el servicio nginx:

```
$ sudo service nginx reload
```

### Comprobaciones

Puedes probar si funciona correctamente o no, creando una simple página de información PHP:

```
$ echo "?php phpinfo();" | sudo tee
/var/www/html/index.php
```

Cuando accedas a http://{DIRECCION IP ODROID}/, te mostrará información relacionada con PHP.

El entorno y las condiciones para las pruebas incluye:

- **Servidor LEMP**

  - Ubuntu Minimal 16.04.3 LTS con Kernel 4.9.51-64
  - Nginx 1.12.1
  - PHP 7.1.9
  - MariaDB 10.0.31
  - Herramientas para pruebas de rendimiento

- Apache JMeter 3.2 r1790748
- sysbench 0.4.12
- IOzone 3.471

Para probar su rendimiento, llevamos a cabo las pruebas que se muestran a continuación, con las condiciones que hemos enumerado anteriormente.

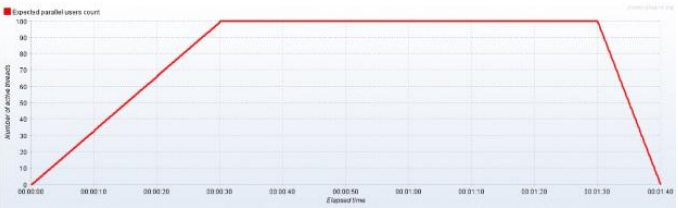


**Figura 1 – Rendimiento de LEMP usando Apache JMeter**

### Sysbench

En primer lugar, crea una base de datos de “prueba” y después ejecuta las pruebas de rendimiento.

```
$ sudo sysbench --test=oltp --oltp-table-
size=1000000 --mysql-db=test --mysql-user=root
```

```
prepare
$ sudo sysbench --test=oltp --oltp-table-size=1000000 --mysql-db=test --mysql-user=root --max-time=60 --oltp-read-only=on --max-requests=0 --num-threads=8 run
$ sudo sysbench --test=oltp --mysql-db=test --mysql-user=root cleanup
```

Para probar la E/S de archivos, ejecuta la siguiente prueba:

```
$ sudo sysbench --test=fileio --file-total-size=4G prepare
$ sudo sysbench --test=fileio --file-total-size=4G --file-test-mode=rndrw --init-rng=on --max-time=300 --max-requests=0 --num-threads=8 run
$ sudo sysbench --test=fileio --file-total-size=4G cleanup
$ sudo iozone -e -I -a -s 100M -r 4k -r 16384k -i 0 -i 1 -i 2
```

### Resultados

- Ten en cuenta que las pruebas del módulo eMMC se ejecutaron sobre un ODROID-XU4 y el resto de las de llevaron a cabo en el ODROID-HC1.
- Todos los resultados tienen sus propios márgenes de error
- Puesto que WordPress se ejecuta con PHP, MariaDB (MySQL) y Nginx (Apache), instalamos WordPress en cada sistema para crear unas condiciones de prueba nativas. Las pruebas de JMeter se ejecutaron accediendo a la página principal de WordPress por defecto.
- Observamos fallos en la respuesta si teníamos más de 140 usuarios accediendo de forma simultánea a la simple página web de WordPress.
- TPS es la abreviatura de Transacción por segundo, así que ésta es la prueba más cercana a un posible entorno de usuario.

| | HDD 2TB | SSD 120G | eMMC 64G | eMMC 16G | MicroSD 8G |
|---|---|---|---|---|---|
| **Apache JMeter** | | | | | |
| TPS medio con 100 usuarios simultáneos | 51.1 | 53.1 | 54.5 | 53.4 | 52.3 |
| Tiempo de Respuesta medio con 100 usuarios simultáneos | 1578 | 1519 | 1477 | 1510 | 1540 |
| **sysbench** | | | | | |
| TPS medio OLTP(MySQL) | 361.81 | 401.59 | 396.43 | 395.14 | 340.05 |
| Velocidad de transferencia media E/S Ficheros (Mbps) | 1.9359 | 17.982 | 24.593 | 16.738 | 0.094831 |
| **IOzone (Kbps)** | | | | | |
| Velocidad de lectura aleatoria (4K) | 1580 | 20937 | 15466 | 15203 | 9139 |
| Velocidad de escritura aleatoria (4K) | 1275 | 21078 | 15803 | 17939 | 827 |
| Velocid | 115209 | 363269 | 142535 | 147790 | 42885 |

| | | | | | |
|---|---|---|---|---|---|
| ad de lectura secuencial (16M) | | | | | |
| Velocidad de escritura secuencial (16M) | 108979 | 278223 | 88529 | 33709 | 13022 |

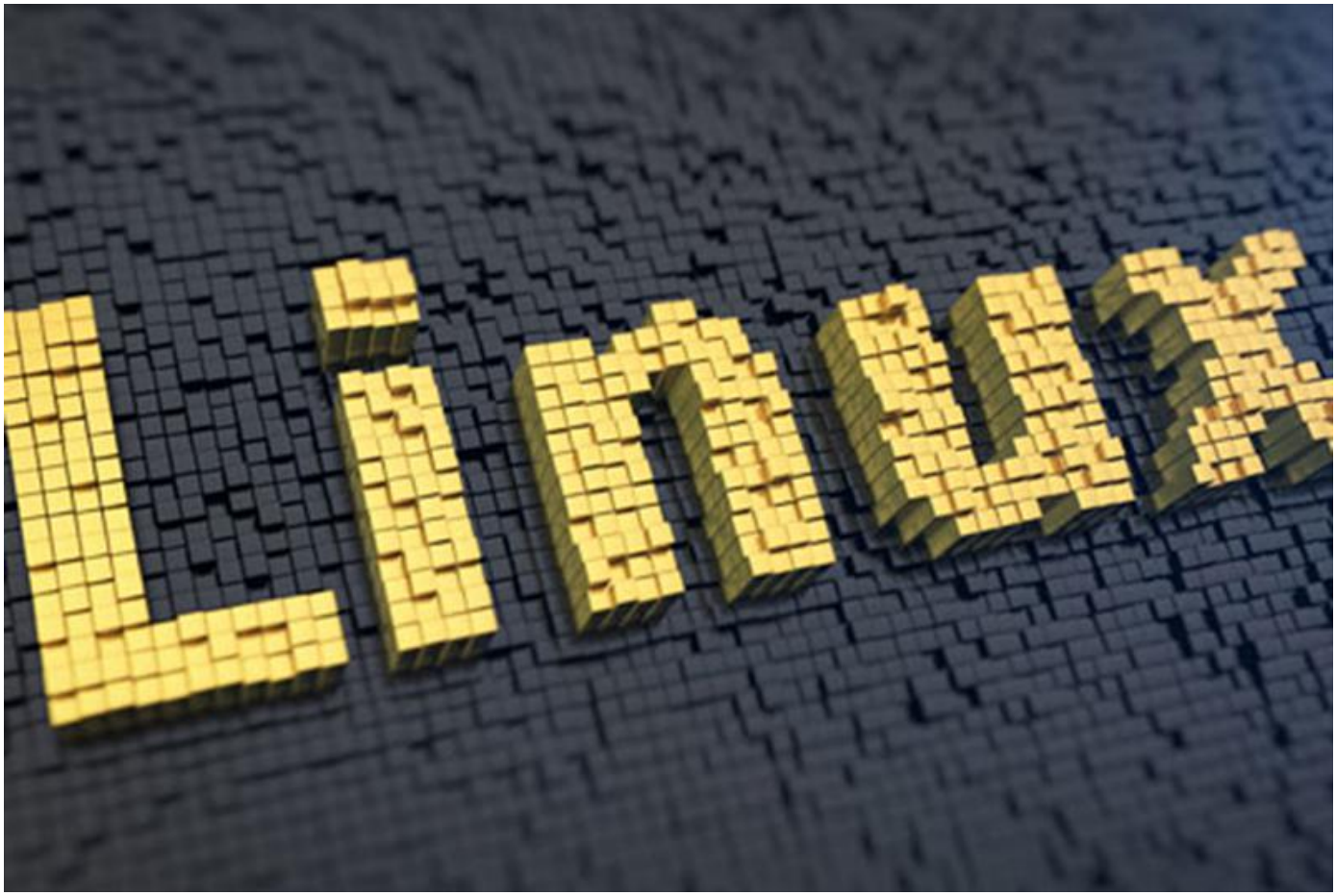
Como puedes ver en la tabla anterior, un HDD es ligeramente más rápido que una tarjeta MicroSD en cuanto a la velocidad de acceso aleatorio. El resultado del TPS OLTP y la velocidad de acceso secuencial son bastante buenos, pero la velocidad de acceso aleatorio no es aceptable. El TPS medio que se muestra arriba es simplemente un valor promedio, necesitas saber que la velocidad de acceso aleatorio es uno de los valores más importantes a la hora medir la velocidad global del sistema. Los resultados OLTP del HDD ODROID variaron bastante entre las pruebas. Sin embargo, en lo que respecta a la velocidad de acceso secuencial, éste es casi tan rápido como un PC escritorio. Por lo tanto, usar una unidad de disco duro en el ODROID-HC1 para un NAS puede ser una muy buena opción.

Respecto a las pruebas TPS con los 100 usuarios simultáneos, no hay mucha diferencia. Sin embargo, en las otras pruebas, como la TPS OLTP y IOzone, el SSD o el eMMC 64G parecen más rápido que el resto. En la prueba E/S de ficheros, el SSD es la más rápido en las estadísticas por solicitud.

En base a los resultados anteriores, no se recomienda utilizar un HDD o tarjeta MicroSD para un sistema LEMP o LAMP. Recomendamos utilizar un módulo eMMC o SSD para lograr el mejor rendimiento con un ODROID-XU4/HC1/HC2 a la hora de alojar un sitio web y/o utilizarlo como un NAS. Para más detalles, puedes visitar el artículo original en **http://bit.ly/2l5aUs1**.

# Linux Kernel 4.14: Soporte para ODROID-XU3/4/MC1/HC1

November 1, 2017 By Marian Mihailescu Linux




Exynos 5422 es un System-on-a-Chip (SoC) Samsung que ha sido la base de los productos de Hardkernel durante varios años. Empezando con el ODROID-XU3 de alta gama, más tarde se lanzó una variante más asequible, el ODROID-XU3 Lite; pasó por un completo rediseño con el ODROID-XU4 y XU4Q, y recientemente lo podemos encontrar en el nuevo ODROID-HC1 y MC1, orientados para el uso de NAS y clusters, respectivamente.

Lanzado inicialmente con la versión 3.10 del kernel Linux, han sido muchos los intentos para actualizar los productos ODROID basados en Exynos 5422 a un kernel más reciente, con kernels experimentales lanzados para las versiones 4.0, 4.2, 4.4 y 4.8 por la comunidad, hasta que Hardkernel ayudó a desarrollar y liberar una versión estable del kernel 4.9.

Durante cada ciclo de desarrollo del kernel, el SoC conseguía más actualizaciones para el kernel estándar gracias a los esfuerzos de Samsung y de la comunidad. De este modo, cada vez resultaba más fácil actualizar a la última versión del kernel. De hecho, durante un tiempo, los ODROID basados en Exynos 5422 podían usar el kernel estándar sin modificaciones. Sin embargo, el problema radicaba en que la versión estándar no cuenta con todos los controladores y algunos sistemas no funcionaban.

El siguiente kernel con soporte de larga duración (LTS) es la versión 4.14, que se lanzará en noviembre. Casualmente, hace poco se ha revelado que el soporte para kernels LTS se extenderá hasta los 6 años. Esto significa que la versión 4.14 recibirá soporte hasta finales de 2022. Esta es una gran noticia para los dueños de esas pequeñas, extensibles, potentes y económicas placas, como son los ODROID.

Esta versión del kernel pasa por ser particularmente atractiva para los ODROID Exynos 5422, especialmente porque incluye importantes

correcciones para la decodificación y codificación de video por hardware (MFC), el escalador de hardware y convertidor de espacios de color (GScaler) y un nuevo driver para HDMI CEC, que permite controlar el dispositivo a través de un mando a distancia por infrarrojos.

Además, incluye in sinfín de parches desde la versión 4.9 mantenidos por Hardkernel que pueden beneficiar a los usuarios de ODROIDs, como son las mejoras en los sistemas de archivos EXT4 y BTRFS, las mejoras en la máquina virtual basada en kernel (KVM), drivers WiFi nuevos y actualizados, y mejoras en los drivers para sistemas multimedia (por ejemplo, mejor soporte para sintonizadores de TV).

La comunidad intervino y empezó a solucionar el resto de problemas incluso antes de que se liberara el kernel 4.14:

- Los Drivers Mali fueron actualizados a la versión r20p0
- Se ha añadido soporte para el programador automático de tareas Multiprocesamiento Heterogéneo (HMP) de CPU
- Parches para los puertos USB3 en ODROID
- Parche para la interfaz Gigabit Ethernet
- Soporte adicional para varias interfaces, tales como el botón de encendido, SPI, ADC, GPIOs
- Controlador de sonido HDMI para el ODROID-XU4
- Mejoras para tarjetas sd/módulos eMMC que incluye soporte para velocidades superiores
- Soporte mejorado para el hipervisor de máquina virtual
- Soporte habilitado para el hardware watchdog
- Soporte para los contadores de rendimiento de hardware de la CPU (PMU)
- Incluido soporte oficial para ODROID-HC1 (programado para la versión de kernel 4.15)
- Soporte adicional para frecuencias extra de CPU y características térmicas mejoradas
- Soporte añadido para las tablas ASV de la CPU de Samsung que permite mejorar la selección del voltaje de la CPU y el nivel térmico.

Con la mayoría de correcciones y parches desarrollados o importados desde el kernel 4.9, la versión 4.14 del kernel ya está lista para someterla a las pruebas de los usuarios. Los interesados pueden descargar, compilar e instalar el kernel en la distribución oficial Ubuntu de Hardkernel usando estas instrucciones:

```
$ git clone --depth 1 --single-branch -b
odroidxu4-4.14.y
https://github.com/mihailescu2m/linux
$ cd linux
$ wget
https://gist.github.com/mihailescu2m/115c9e213
5f9b12f3d05669e18137c3d -O .config
$ make -j 8 zImage dtbs modules
$ sudo cp arch/arm/boot/zImage
arch/arm/boot/dts/*.dtb /media/boot
$ sudo cp .config /media/boot/config
$ sudo make modules_install
$ kver=`make kernelrelease`
$ sudo cp .config /boot/config-${kver}
$ cd /boot
$ sudo update-initramfs -c -k ${kver}
$ sudo mkimage -A arm -O linux -T ramdisk -a
0x0 -e 0x0 -n initrd.img-${kver} -d
initrd.img-${kver} uInitrd-${kver}
$ sudo cp uInitrd-${kver} /media/boot/uInitrd
```

Los comentarios y resultados de las pruebas son bienvenidos en el foro “Kernel 4.14 debugging party” en **http://bit.ly/2ltTPbD**.

# ODROID-C2: Apaga y enciende el sistema usando un simple botón GPIO

November 1, 2017 By Justin Lee ODROID-C2, Mecaniqueo


Este artículo describe cómo configurar un botón GPIO con vistas a apagar y encender el sistema. Los breves pasos de configuración incluyen:

- Conectar un botón en forma de tabulador al puerto pin que quieras utilizar
- Configurar el número GPIO usando boot.ini
- Compilar y grabar el kernel DTS modificado, que sólo es necesario para Android

### Configuración del Hardware

En primer lugar, debes preparar un interruptor que se conectará a dos líneas GPIO. El cable rojo será para la alimentación y el cable gris para la línea activa (puesta a tierra o potencia de 3.3V).

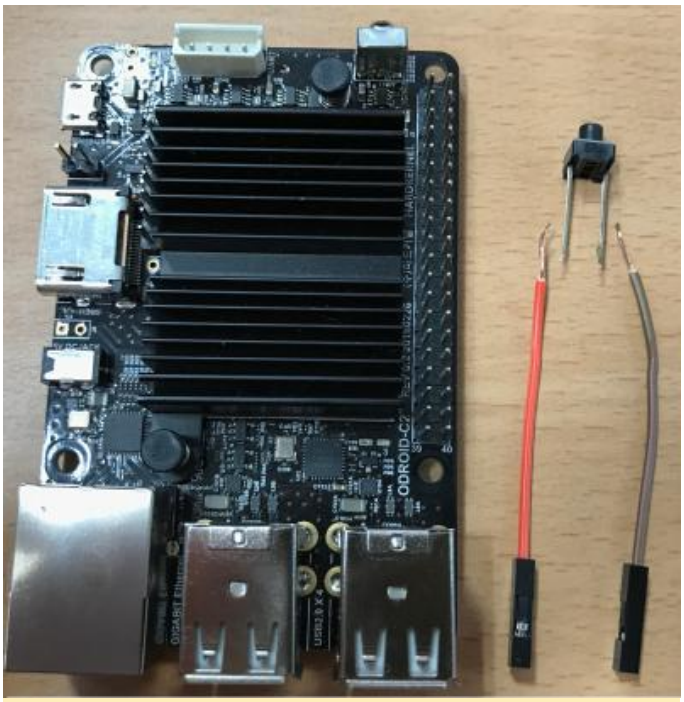


**Figura 1 – Material para el proyecto de botón de encendido ODROID-C2**

El diagrama de la disposición de los pines ODROID-C2 http://bit.ly/2aXAlmt nos será muy útil para este proyecto. En nuestro caso, utilizaremos el pin #29 del conector de expansión de 40 pines. El pin es asignado a GPIOX.BIT0 y su número GPIO será 228. Conecta la línea roja al pin #29. Su estado por defecto será high y el interruptor activado será low. Por lo tanto, debe conectar la línea gris del interruptor a la puesta a tierra (GND), pin #30.

| **Núemro de Pin (Linea Roja)** | **Número GPIO** | **Nivel Activo (Linea Gris)** |
|---|---|---|
| 29 | GPIO# 228 | Active Low (Pin 30) |

### Teclas disponibles

Aquí tienes ejemplos de teclas disponibles para el conector de 40 pines y el conector de 7 pines. Puede localizar los ejemplos de asignación de pin para Línea roja y Línea gris.

(1) J2 – 2×20 pins

| **Nivel Activo (Linea Gris)** | **GPIO #** | **Pin # (Linea Roja)** | **Pin # (Linea Roja)** | **GPIO #** | **Nivel Activo (Linea Gris)** |
|---|---|---|---|---|---|
| – | 3.3V Power | 1 | 2 | – | – |
| – | – | 3 | 4 | – | – |
| – | – | 5 | 6 | Ground | – |
| Active Low (Pin 9) | GPIO# 249 | 7 | 8 | – | – |
| – | Ground | 9 | 10 | – | – |
| Active Low (Pin 14) | GPIO# 247 | 11 | 12 | GPIO# 238 | Active Low (Pin 14) |
| Active Low (Pin 14) | GPIO# 239 | 13 | 14 | Ground | – |
| Active Low (Pin 14) | GPIO# 237 | 15 | 16 | GPIO# 236 | Active Low (Pin 14) |
| – | 3.3V Power | 17 | 18 | GPIO# 233 | Active Low (Pin 20) |
| Active Low (Pin 20) | GPIO# 235 | 19 | 20 | Ground | – |
| Active Low (Pin 20) | GPIO# 232 | 21 | 22 | GPIO# 231 | Active Low (Pin 20) |
| Active Low (Pin 25) | GPIO# 230 | 23 | 24 | GPIO# 229 | Active Low (Pin 25) |
| – | Ground | 25 | 26 | GPIO# 225 | Active High (Pin 17) |
| – | – | 27 | 28 | – | – |
| Active Low (Pin 30) | GPIO# 228 | 29 | 30 | Ground | – |
| Active Low (Pin 30) | GPIO# 219 | 31 | 32 | GPIO# 224 | Active Low (Pin 34) |
| Active High (Pin 17) | GPIO# 234 | 33 | 34 | Ground | |
| Active Low (Pin 34) | GPIO# 214 | 35 | 36 | GPIO# 218 | Active Low (Pin 34) |
| – | – | 37 | 38 | – | – |
| – | Ground | 39 | 40 | – | – |

(2) J7 – 1×7 pins

| **Pin # (Linea Roja)** | **GPIO #** | **Nivel Activo (Linea Gris)** |
|---|---|---|
| 1 | Ground | – |
| 2 | GPIO# 128 | Active Low (Pin 1) |
| 3 | 5.0V Power | – |
| 4 | GPIO# 130 | Active Low (Pin 1) |
| 5 | GPIO# 132 | Active Low (Pin 1) |
| 6 | GPIO# 131 | Active Low (Pin 1) |
| 7 | GPIO# 133 | Active Low (Pin 1) |

Puedes encontrar información detallada sobre conectores de expansión de 40 y 7 pines http://bit.ly/2gzCA7c.

### Configurar el Software Ubuntu

La versión liberada debe ser la 3.14.79-107 (26 de febrero de 2017) o superior. Puedes asignar un número GPIO con env gpiopower en boot.ini:

```
## gpio power key : J2 (2x20) Pin#29 ,
GPIOX.BIT0
setenv gpiopower "228"
...
## Add gpiopower like "setenv bootargs
${bootargs} gpiopower=${gpiopower}"
setenv bootargs "root=UUID=e139ce78-9841-40fe-
8823-96a304a09859 rootwait ro ${condev}
no_console_suspend hdmimode=${m} ${cmode}
m_bpp=${m_bpp} vout=${vout} fsck.repair=yes
net.ifnames=0 elevator=noop disablehpd=${hpd}
max_freq=${max_freq} maxcpus=${maxcpus}
monitor_onoff=${monitor_onoff}
disableuhs=${disableuhs}
mmc_removable=${mmc_removable}
usbmulticam=${usbmulticam} ${hid_quirks}
gpiopower=${gpiopower}"
```

### Configurar la acción del botón para el apagado

Si tienes la intención de apagar la placa con el botón de encendido, necesitas cambiar el patrón de acción de la tecla. Para hacer posible esto, primero abre el cuadro de diálogo de Preferencias de la gestión de energía ([System] → [Preferences] → [Hardware] → [Power Management]) y selecciona la festaña [General].

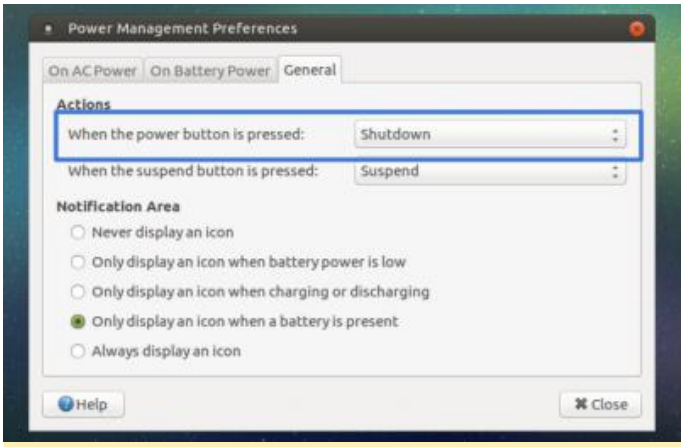



**Figura 2: Preferencias de gestión de energía**

### Acción de encendido

Para reactivar el sistema después de apagarlo, es necesario presionar el botón durante un tiempo (2-3 segundos).

### Configurar el software Android

En Android, necesitas modificar el archivo DTS en Android Marshmallow (v2.4) y versiones superiores para usar esta funcionalidad.

Tienes que modificar el archivo DTS para activar la funcionalidad de la tecla GPIO:

```
/arch/arm64/boot/dts/meson64_odroidc2.dts
...
gpio_keypad{
    .
    status = "okay";
    .
};
…
```

Tras compilar el archivo DTS, puedes grabar el archivo DTB en la placa:

```
$ make odroidc2_defconfig
$ make dtbs
$ fastboot flash dtb
arch/arm64/boot/dts/meson64_odroidc2.dtb
$ fastboot reboot
```

Debes grabar el archivo DTBS cuando la placa esté en modo fastboot de u-boot:

```
$ reboot fastboot
```

### Configurar boot.ini

En Android boot.ini, puedes descomentar la entrada “gpipower” y modificar el número existente por el número aplicable en tu caso:

```
## gpio power key : J2 (2x20) Pin#29 ,
GPIOX.BIT0
setenv gpiopower "228"
```

### Configurar la acción del botón para el apagado

En Android, no necesitas seleccionar ninguna opción de menú para las acciones del botón. Ya se encuentran definidas del siguiente modo:

Un evento de tecla corto se usa para dormir, y con evento largo, puedes manejar las opciones de

apagado/reinicio.

Con la versión Android Marshmallow v2.4 o superior, es posible apagar el sistema con una pulsación larga (5 segundos).

**Acción de encendido**

Para encender después del apagado es necesario presionar el botón durante un tiempo (2-3 segundos).

**Solución de problemas**

Puedes consultar los post del foro en **http://bit.ly/2gtE4MA** y **http://bit.ly/2zGwB4X** para obtener consejos de cómo solucionar problemas. Para comentarios, preguntas y sugerencias, visita **http://bit.ly/2yYFWrB**.

# Google Assistant: Cómo utilizar un micrófono y un altavoz Bluetooth en la plataforma Ubuntu ODROID-XU4 y ODROID-HC1

November 1, 2017 By Brian Kim ODROID-HC1, ODROID-XU4


Este artículo describe cómo poner en marcha un Altavoz IA (Inteligencia Artificial) sobre ODROID-HC1 utilizando Google Assistant SDK.

### Requisitos de Hardware

- ODROID-HC1 (**http://bit.ly/2wjNToV**) con fuente de alimentación 5V/4A
- Módulo Bluetooth 2 (**http://bit.ly/2gNybJW**)
- Altavoz Bluetooth con micrófono incluido e (**http://amzn.to/2z6dBz5**)
- Tarjeta MicroSD clase 10 de 8gb o más para el sistema operativo
- Cable LAN

**Figura 1 – Requisitos de hardware para el proyecto Google Assistant**

IConecta el dongle Bluetooth al puerto USB del ODROID-HC1, luego enciende el ODROID-HC1 y el altavoz Bluetooth para empezar.

### Ajustes de Sonido

Para acceder a la consola del ODROID-HC1, consigue la dirección IP de la placa tal y como se describe en **http://bit.ly/2yXFLwp**. Esta guía está basada en el último sistema operativo Ubuntu 16.04 Minimal. Puedes descargar la imagen del sistema operativo desde **http://bit.ly/2yXYs3h**. Antes de empezar con la configuración del sistema, añade la nueva cuenta de

usuario ODROID como un usuario sudo, porque Ubuntu Minimal no tiene ninguna cuenta de usuario:

```
# adduser odroid
# usermod -aG sudo odroid
# su - odroid
```

Instala los paquetes relacionados con el sonido pulseaudio y alza:

```
$ sudo apt update
$ sudo apt install libasound2 libasound2-plugins alsa-utils alsa-oss
$ sudo apt install pulseaudio pulseaudio-utils pulseaudio-module-bluetooth
```

Añade permisos pulseaudio a la cuenta de usuario. Agrega la línea "load-module module-switch-on-connect" al archivo de configuración de pulseaudio. Esta configuración cambia la salida de audio al altavoz Bluetooth automáticamente:

```
$ sudo usermod -aG pulse,pulse-access odroid
$ sudo nano /etc/pulse/default.pa
```

/etc/pulse/default.pa

```
.ifexists module-bluetooth-discover.so
load-module module-bluetooth-discover
load-module module-switch-on-connect # this is new!
.endif
```

Inicia pulseaudio:

```
$ pulseaudio --start
```

**Configuración de Bluetooth**

Instala el paquete relacionado con el Bluetooth. En este caso, usa el paquete bluez para bluetooth:

```
$ sudo apt install bluez
$ bluetoothctl
```

Si el comando bluetoothctl no funciona con la cuenta de usuario, modifica el archivo de configuración de dbus añadiendo las siguientes configuraciones al archivo:

```
$ sudo nano /etc/dbus-1/system.d/bluetooth.conf
```

/etc/dbus-1/system.d/bluetooth.conf

```
< policy user="odroid">
    < allow send_destination="org.bluez"/>
    < allow send_interface="org.bluez.Agent1"/>
    < allow send_interface="org.bluez.GattCharacteristic1" />
    < allow send_interface="org.bluez.GattDescriptor1"/>
    < allow send_interface="org.freedesktop.DBus.ObjectManager"/>
    < allow send_interface="org.freedesktop.DBus.Properties"/>
  < /policy>
```

Introduce los siguientes comandos en la consola bluetoothctl. Ten en cuenta que la dirección MAC del altavoz Bluetooth debe cambiarse de 00:11:67:AE:25:C6 a tu propia dirección MAC. Esta dirección es diferente para cada dispositivo Bluetooth, así que asegúrate de reemplazar la dirección MAC por la suya:

```
[bluetooth]# agent on
[bluetooth]# default-agent
[bluetooth]# scan on
[bluetooth]# pair 00:11:67:AE:25:C6
[bluetooth]# trust 00:11:67:AE:25:C6
[bluetooth]# connect 00:11:67:AE:25:C6
[bluetooth]# quit
```

El altavoz Bluetooth debería tener una configuración por defecto. Para ajustar el A2DP (Advanced Audio Distribution Profile) como predeterminado, cambia el perfil a HSP (Head Set Profile) porque A2DP no puede usar el micrófono.

```
$ pacmd ls
```

Consulta el fichero del altavoz Bluetooth, suponemos que el index es 1:

```
$ pacmd set-card-profile 1 headset_head_unit
```

Para verificar que hay sonido y que la configuración de Bluetooth se ha realizado correctamente, reproduce un sonido de prueba:

```
$ speaker-test -t wav
```

Graba y reproduzca algo de audio utilizando las herramientas de línea de comandos ALSA:

```
$ arecord --format=S16_LE --duration=5 --
rate=16k --file-type=raw out.raw
 $ aplay --format=S16_LE --rate=16k --file-
type=raw out.raw
```

Para facilitar el uso del altavoz Bluetooth, es necesario realizar algunas configuraciones:

/etc/bluetooth/main.conf

```
[Policy]
AutoEnable=true
```

($HOME)/.bashrc

```
pulseaudio --start
echo "connect 00:11:67:AE:25:C6" |
bluetoothctl
```

### Activar la API de Google Assistant

Para habilitar la API de Google Assistant, consulta la página de Guías de Google Assistant SDK en **http://bit.ly/2pXwqfC**. Usa una cuenta de Google para iniciar sesión. Si todavía no dispones de una cuenta de Google, cree una. Probar la API de Google Assistant es gratis para uso personal.

### Configurar un Proyecto Desarrollador Google

Un Proyecto desarrollador Google permite que cualquier dispositivo ODROID tenga acceso a la API de Google Assistant. El proyecto rastrea el uso de cuota y proporciona valiosas métricas de las peticiones realizadas desde dispositivos ODROID en la red.

Para habilitar el acceso a la API de Google Assistant, primero dirígete a la página de Proyectos dentro de Cloud Platform Console y selecciona un proyecto existente o cree un nuevo proyecto. Ve a la página de Proyectos en **http://bit.ly/2gY7pSV**. A continuación, activa la API de Google Assistant en el proyecto que has seleccionado y haz clic en Habilitar. Puedes obtener más información sobre cómo activar la API en **http://bit.ly/2A1ewic**.

Después, crea un ID de Cliente OAuth creando primero la ID del cliente, tal y como se describe en **http://bit.ly/2xBjII6**. Puede que necesites definir un nombre de producto en la pantalla de aprobación de producto. En la pestaña aprobación de OAuth, asigna un nombre al producto y haga clic en Guardar, luego haz clic en Otro y asigna un nombre al ID del cliente y haz clic en Crear. Aparecerá un cuadro de diálogo que te mostrará el ID de cliente y un secret. No hay necesidad de recordar o guardar esto, simplemente cierra el cuadro de diálogo. A continuación, haz clic en el extremo derecho de la pantalla para que la ID del cliente descargue el archivo JSON del cliente (client_secret_.json). El archivo client_secret_.json debe estar ubicado en el dispositivo para autorizar a que la muestra de Google Assistant SDK pueda realizar consultas a Google Assistant, y no debe cambiarse de nombre. Finalmente, copia client_secret_.json en el ODROID-HC1:

```
$ scp ~/Downloads/client_secret_client-id.json
odroid@:~/
```

### Definir los controles de actividad para tu cuenta de Google

Para poder usar Google Assistant, hay determinados datos de actividad que tienen que ser compartidos con Google. Google Assistant necesita que estos datos funcionen correctamente, no son específicos del SDK. Para hacer esto, abre la página de Controles de actividad de la cuenta de Google que se usará con el Asistente en **http://bit.ly/2ig4QIB**. Cualquier cuenta de Google tiene esta opción y no necesita ser tu cuenta de desarrollador. Asegúrate de que las siguientes opciones estén activadas (azul):

- Web & App Activity
- Device Information
- Voice & Audio Activity

### Descargar y ejecutar la muestra de la API de Google Assistant

Utiliza un entorno virtual Python para aislar el SDK y sus dependencias de los paquetes Python del sistema:

```
$ sudo apt update
$ sudo apt install python-dev python-
virtualenv git portaudio19-dev libffi-dev
```

```
libssl-dev
$ virtualenv env --no-site-packages
```

Si tienes problemas con la configuración regional tal y como se muestra a continuación, configura la variable de entorno LC_ALL:

```
Complete output from command
/home/odroid/env/bin/python2 - setuptools
pkg_resources pip wheel:
Traceback (most recent call last):
File "", line 24, in
File "/usr/share/python-wheels/pip-8.1.1-
py2.py3-none-any.whl/pip/__init__.py", line
215, in main
File
"/home/odroid/env/lib/python2.7/locale.py",
line 581, in setlocale
return _setlocale(category, locale)
locale.Error: unsupported locale setting
$ export LC_ALL=C
$ virtualenv env --no-site-packages
Activate Python virtual environment.
$ env/bin/python -m pip install --upgrade pip
setuptools
$ source env/bin/activate
```

Tras activar el entorno virtual de Python, aparecerá la cadena "(env)" delante del prompt.

Autoriza que la muestra de SDK de Google Assistant pueda realizar consultas a Google Assistant para la cuenta de Google especificada. Haz referencia al archivo JSON que ha sido copiado en el dispositivo en un paso anterior e instala la herramienta de autorización:

```
(env) $ python -m pip install --upgrade
google-auth-oauthlib[tool]
```

Ejecute la herramienta, asegurándote de quitar el parámetro –headless si estás ejecutando esto desde una terminal en el dispositivo (no usas una sesión SSH):

```
(env) $ google-oauthlib-tool --client-secrets
/path/to/client_secret_client-id.json --scope
https://www.googleapis.com/auth/assistant-sdk-
prototype --save --headless
```

Debería ver una URL que se muestra en el terminal:

```
Please go to this URL: https://
```

Copia la URL y pégala en un navegador. Esto se puede hacer en la máquina de desarrollo o en cualquier otra máquina. Una vez aceptada, aparecerá un código en el navegador, algo así como "4/XXXX". Copia y pega este código en el terminal:

```
Enter the authorization code:
```

Si la autorización ha tenido exito, las credenciales de OAuth se activarán en la terminal. Si en su lugar aparece InvalidGrantError, entonces es que se ha introducido un código no valido. Si esto ocurre, inténtalo de nuevo, teniendo cuidado de copiar y pegar todo el código. Si se introduce el código de autorización correcto, se genera el archivo credentials.json:

```
credentials saved:
/home/odroid/.config/google-oauthlib-
tool/credentials.json
```

Hazte con los códigos de muestra del repositorio de github:

```
$ git clone
https://github.com/googlesamples/assistant-
sdk-python
$ cd assistant-sdk-python
```

Instala los requisitos de paquetes de Python para el programa de ejemplo. Usamos la muestra pushtotalk.

```
$ cd google-assistant-sdk
$ python setup.py install
$ cd googlesamples/assistant/grpc
$ pip install --upgrade -r requirements.txt
$ nano pushtotalk.py
```

Para ejecutar el ejemplo, tenemos que modificar el código. Cambia el tipo de excepción SystemError a ValueError en el código de ejemplo (línea 35):

pushtotalk.py

```
except ValueError:
import assistant_helpers
import audio_helpers
```

Ejecuta y prueba la muestra pushtotalk. Si el programa de ejemplo funciona bien, está hecho casi todo trabajo:

```
(env) $ python pushtotalk.py
INFO:root:Connecting to
embeddedassistant.googleapis.com

Press Enter to send a new request...
```

Copia la muestra en el directorio de trabajo. Desactiva el entorno virtual de Python. Hay que realizar una serie de pasos adicionales para dar forma a un interlocutor IA de provecho. Para eso, dirígete al directorio $ (HOME)/ai_speaker:

```
(env) $ cd ..
(env) $ cp -r grpc ~/ai_speaker
(env) $ cd ~/ai_speaker
(env) $ cp pushtotalk.py ai_speaker.py
(env) $ deactivate
$ cd
```

**Wake-Up-Word**

La muestra Wake-Up-Word parece que interactúa con el asistente de inteligencia artificial. Sin embargo, antes de comunicarse con el asistente IA, primero hay que presionar la tecla Intro. Para detectar un Wake-Up-Word como "Okay, Google", "Alexa" o "Jarvis", usa CMUSphinx de **https://cmusphinx.github.io** , que es el kit de herramientas de reconocimiento de voz local de código abierto. Lo mejor es compilar e instalar SphinxBase, ya que éste proporciona una funcionalidad común para todos los proyectos CMUSphinx:

```
$ sudo apt install libtool bison swig python-
dev autoconf libtool automake
$ git clone --depth 1
https://github.com/cmusphinx/sphinxbase.git
$ cd sphinxbase
$ ./autogen.sh
$ make -j8
$ sudo make install
$ cd
```

Sphinxbase se instalará en el directorio "/usr/local/" por defecto. No todos los sistemas cargan las librerías automáticamente desde esta carpeta. Para cargarlas, configura la ruta para localizar las librerías compartidas. Esto se puede hacer en el archivo "/etc/ld.so.conf" o exportando las variables de entorno:

```
export LD_LIBRARY_PATH=/usr/local/lib
export
PKG_CONFIG_PATH=/usr/local/lib/pkgconfig
```

Compila e instala PocketSphinx. PocketSphinx es un liviano motor de reconocimiento de voz específalmente diseñado para dispositivos portátiles y dispositivos móviles, aunque funciona igual de bien en los PC de escritorio:

```
$ git clone --depth 1
https://github.com/cmusphinx/pocketsphinx.git
$ cd pocketsphinx
$ make -j8
$ sudo make install
$ cd
```

Para probar la instalación, ejecuta pocketsphinx_continuous y comprueba que se reconoce tus palabras a través del micrófono:

```
$ pocketsphinx_continuous -inmic yes
```

Para obtener más información sobre el desarrollo de PocketSphinx, consulte la página "Building an application with PocketSphinx " en **http://bit.ly/2gZhHT5**.

Añade el programa pocketsphinx_continuous como un subproceso del programa del Altavoz IA. El programa pocketsphinx_continuous es una buena herramienta para detectar palabras clave ya que reconoce el habla de forma asincrónica. Procede a eliminar las líneas relacionadas wait_for_user_trigger puesto que las propias palabras clave son el activador:

```
$ source env/bin/activate
(env) $ pip install --upgrade subprocess
```

```
$(HOME)/ai_speaker/ai_speaker.py

 """Sample that implements gRPC client for
Google Assistant API."""

# Add subprocess module
 import subprocess
 import json
 import logging
 import os.path
```

```
(......)

# Add below's routines in the 'While True:'
loop
 while True:
 p = subprocess.Popen(args =
['pocketsphinx_continuous','-inmic', 'yes', '-
kws_threshold', '1e-16', '-keyphrase', 'hey
dude'],
 stdin = subprocess.PIPE,
 stdout = subprocess.PIPE,
 universal_newlines=True)
 while p.poll() is None:
 data = p.stdout.readline()
 if data.find("hey dude") is not -1:
 print "Detected Hotwords"
 p.stdout.flush()
 break
 p.terminate()
```

El Wake-Up-Word es “hey dude”. Ejecuta el programa, pronuncia “hey dude” y luego dile todo lo que quieras al asistente IA:

```
(env) $ cd ai_speaker
(env) $ python ai_speaker.py
```

**Sonido de detección**

Existe un problema tras añadir inicialmente los Wake-Up-Words, porque no hay ningún mecanismo en situ para darnos cuenta si el altavoz IA está detectando o no las palabras clave. Se debe conocer la sincronización para poder controlar el asistente IA por voz. Esto puede solucionarse añadiendo un sonido de detección al programa. Descarga el sonido de detección de muestra en **http://bit.ly/2zkSV3b**, luego copia el archivo detect.wav al ODROID-HC1:

```
$ scp ~/Downloads/detect.wav odroid@:~/
```

Utiliza el módulo wave y pyaudio para reproducir el archivo .wav en el código fuente de Python:

```
(env) $ pip install --upgrade pyaudio wave
```

Agrega la rutina para reproducir el sonido de detección al programa. Todas las posibles variedades, incluidas las rutinas Wake-Up-Words son las siguientes:

```
(env) $ nano ai_speaker.py
```

Archivo diff entre el código de muestra original pushtotalk.py y el programa modificado ai_speaker.py

```
 --- pushtotalk.py 2017-10-19
15:42:12.164741800 +0000
 +++ ai_speaker.py 2017-10-19
15:41:49.644811151 +0000
 @@ -14,6 +14,9 @@

 """Sample that implements gRPC client for
Google Assistant API."""

+import pyaudio
+import wave
+import subprocess
 import json
 import logging
 import os.path
@@ -310,14 +313,38 @@
 # keep recording voice requests using the
microphone
 # and playing back assistant response using
the speaker.
 # When the once flag is set, don't wait for a
trigger. Otherwise, wait.
- wait_for_user_trigger = not once
+ chunk = 1024
+ pa = pyaudio.PyAudio()
+ while True:
- if wait_for_user_trigger:
- click.pause(info='Press Enter to send a new
request...')
+ p = subprocess.Popen(args =
['pocketsphinx_continuous','-inmic', 'yes', '-
kws_threshold', '1e-16', '-keyphrase', 'hey
dude'],
+ stdin = subprocess.PIPE,
+ stdout = subprocess.PIPE,
+ universal_newlines=True)
+ while p.poll() is None:
+ data = p.stdout.readline()
+ if data.find("hey dude") is not -1:
+ print "Detected Hotwords"
+ p.stdout.flush()
+ break
+ p.terminate()
+
+ # Play the detection sound
+ f =
wave.open(r"/home/odroid/detect.wav","rb")
+ stream = pa.open(format =
pa.get_format_from_width(f.getsampwidth()),
```

```
+ channels = f.getnchannels(),
+ rate = f.getframerate(),
+ output = True)
+ wav_data = f.readframes(chunk)
+
+ while wav_data:
+ stream.write(wav_data)
+ wav_data = f.readframes(chunk)
+ stream.stop_stream()
+ stream.close()
+ f.close()
+
continue_conversation = assistant.converse()
- # wait for user trigger if there is no
follow-up turn in
- # the conversation.
- wait_for_user_trigger = not
continue_conversation

# If we only want one conversation, break.
if once and (not continue_conversation):
# Run the AI speaker program.
(env) $ python ai_speaker.py
```

Para ver el altavoz en acción, consulta el video en **https://youtu.be/6Ez782BxxdQ**.

### El paso final

La tasa de detección de Wake-Up-Words es algo inferior a la ideal. Tanto si usa Pocketsphinx como cualquier otra solución, la rutina Wake-Up-Words necesita mejoras, así que añadir comandos personalizados va a resultar muy útil en este proyecto en concreto. Por ejemplo, es fácil controlar los dispositivos IoT por voz mediante el Google Assistant SDK. Se pueden localizar diferentes soluciones recurrirendo a la búsqueda “action on google ” para obtener más información sobre la extensión de Google Assistant.

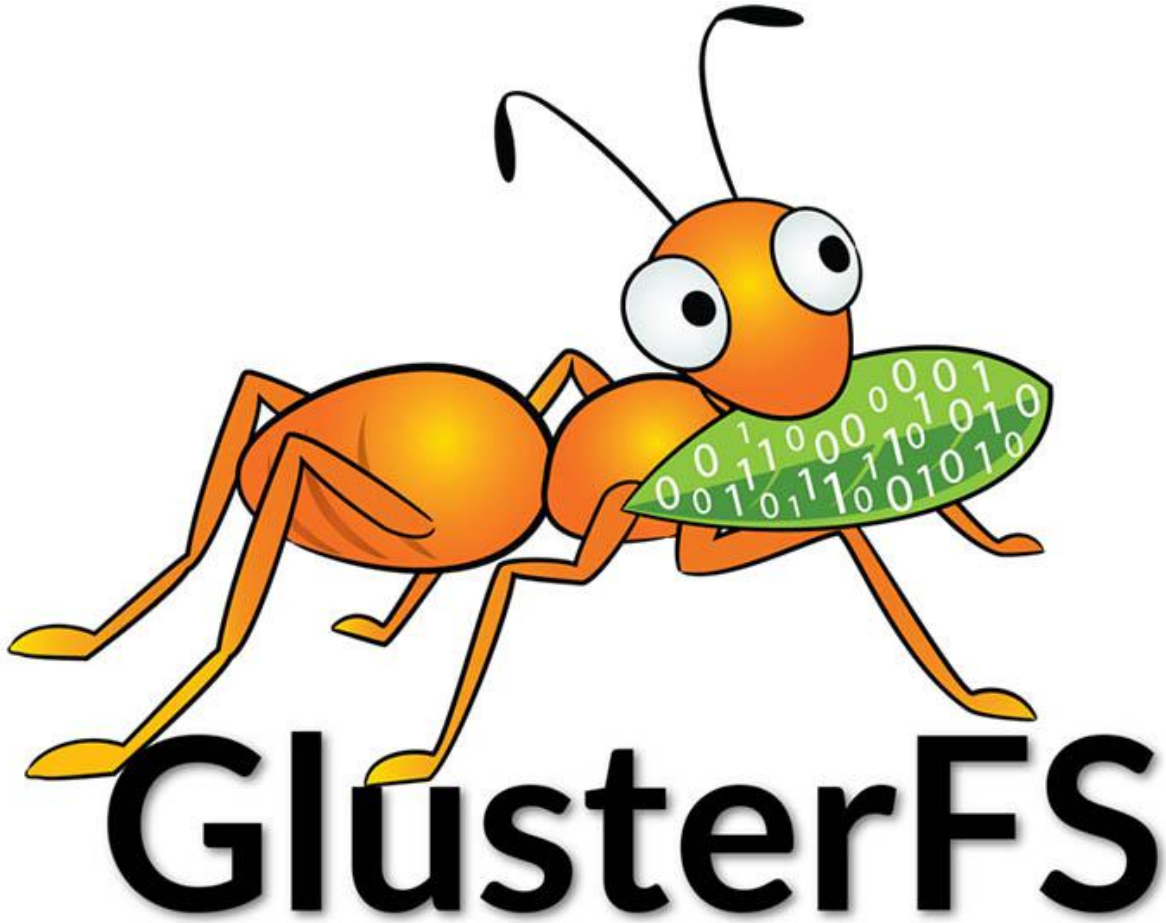
Para ahorrar tiempo, se puede utilizar una sencilla solución de comando personalizado simplemente añadiendo el comando al programa ai_speaker.py. En la muestra pushtotalk, busca el texto de solicitud que ya está reconocido por voz:

```
--- pushtotalk.py 2017-10-19
16:07:46.753689882 +0000
+++ pushtotalk_new.py 2017-10-19
16:09:58.165799271 +0000
@@ -119,6 +119,15 @@
logging.info('Transcript of user request:
"%s".',
resp.result.spoken_request_text)
logging.info('Playing assistant response.')
+ #Add your custom voice commands here
+ #Ex>
+ #import os
+ #r_text = resp.resut.spoken_request_text
+ #if r_text.find("play music") is not -1:
+ # os.system("mplayer ~/Music/*&")
+ #if r_text.find("turn on light") is not -1:
+ # os.system("echo 1 >
/sys/class/gpio/gpio1/value")
+
if len(resp.audio_out.audio_data) > 0:
self.conversation_stream.write(resp.audio_out.
audio_data)
if resp.result.spoken_response_text:
```

Después de guardar esta modificación, puede empezar a experimentar con el control de dispositivos electrónicos domésticos utilizando el controlador IOT con comandos de voz. Para comentarios, preguntas y sugerencias, visita el artículo original en **http://bit.ly/2iQ629K**.

# Analizando el almacenamiento definido por software con GlusterFS en el ODROID-HC1: Parte 1 – Configuración del servidor

November 1, 2017 By Andy Yuen ODROID-HC1




Según Whatis.com “El almacenamiento definido por software (SDS) es un programa informático que gestiona la funcionalidad y los recursos del almacenamiento de datos sin tener que depender del hardware de almacenamiento físico subyacente”

Según Red Hat “Gluster File System (GlusterFS) es una plataforma SDS diseñada para gestionar los requisitos del almacenamiento tradicional de archivos: tareas de alta capacidad como backup y almacenamiento de archivo, así como tareas de alto rendimiento de análisis y virtualización”.

Aunque GlusterFS es claramente una tecnología destinada a empresas, ello no significa que no se pueda usar en casa. Al contrario, considero que es más flexible y escalable que los sistemas de almacenamiento conectados en red (NAS) comerciales. Los servidores NAS para usuarios domésticos normalmente vienen con 2 o 4 bahías. Cuando llega el momento de expandir tu sistema, no son tan flexibles. Necesitas cambiar a un sistema más grande con más bahías, o tienes que reemplazar todos tus discos por otros de mayor capacidad. GlusterFS se escala horizontalmente, lo que significa que puede añadir más servidores para expandir tu capacidad de almacenamiento. Para los usuarios domésticos, añadir más servidores no tiene sentido, ya que los servidores son bastante caros. Esto es cierto hasta que vino a la mente el ODROID-HC1, que se vende por solo 49$. Cuatro ODROID-HC1 se pueden equiparar a un único NAS de 4 bahías con un coste aproximado del 50%. Además, el procesador ODROID-XU4, que cuenta con cuatro núcleos A15 y cuatro núcleos A7 en cada HC1, ya es más potente

que el NAS medio dirigido al mercado doméstico, que generalmente viene con un procesador A15 dual-core

En este artículo, voy a utilizar cuatro ODROID-HC1 para crear un volumen GlusterFS replicado, distribuido, altamente escalable y sumamente disponible, similar a una configuración RAID 1+0. No se preocupes si solo tienes dos HC1 a mano. Describiré cómo puedes crear un volumen GlusterFS replicado que sea equivalente a un RAID 1. Pero antes de eso, vamos a analizar un poco los diferentes tipos de volúmenes de GlusterFS que existen.

### Tipos de volumenes GlusterFS

GlusterFS es un sistema de archivos distribuidos compatible con POSIX. Utiliza el Elastic Hash Algorithm para realizar hash de forma inteligente en ubicaciones basadas en rutas y nombres del archivo, en lugar de valerse de un servidor de metadatos como ocurre con algunos otros sistemas de archivos distribuidos. En consecuencia, este sistema permite evitar cuello de botella de rendimiento de metadatos y puede ejecutarse en hardware muy heterogéneo.

Un volumen viene a ser un conjunto de “ladrillos”. Un ladrillo es cualquier directorio en un sistema de archivos de disco subyacente. La mayoría de las operaciones de GlusterFS suceden en el volumen. GlusterFS admite diferentes tipos de volúmenes que están optimizados para escalar la capacidad de almacenamiento, mejorar el rendimiento o ambos. Es posible que desees consultar Gluster Docs en http://bit.ly/2zhI51S para obtener más información. En este artículo, analizare dos tipos en concreto, el volumen replicado y el volumen replicado distribuido que requieren dos y cuatro servidores (o HC1), respectivamente. En mi opinión, estas son las configuraciones más adecuadas para el uso doméstico.

El volumen replicado implica que los archivos siempre se escriben en los ladrillos de dos servidores. Esto es el equivalente al RAID 1. En un volumen replicado distribuido, los archivos se escriben en un servidor o en otro dentro de un conjunto fiable de nodos GlusterFS. Discutiremos estos conjuntos fiables más adelante. Los ladrillos de los dos servidores se replican en los otros dos servidores dentro del conjunto. Esto es similar al RAID 1 + 0, pero con una diferencia importante: RAID 1 + 0 usa striping, lo que significa que diferentes bloques de un archivo se escriben en diferentes servidores. En la distribución, un archivo está escrito por completo en un servidor u otro y los contenidos de los dos servidores se replican en otros dos servidores, tal y como describe el siguiente diagrama.

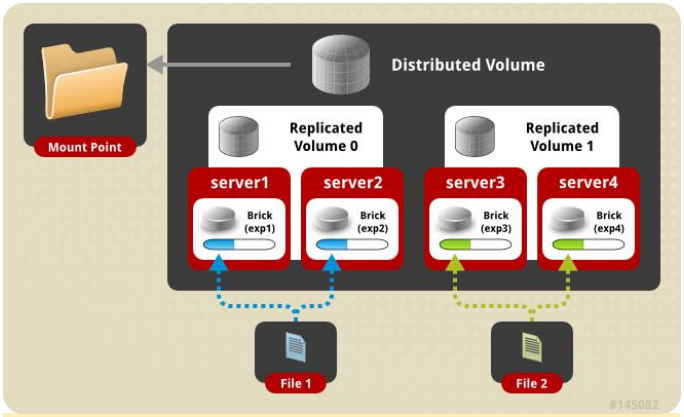



**Figura 1 – Replicado distribuido**

Utilizar un volumen replicado distribuido evita la pérdida de datos cuando falla uno servidor. También mejora el rendimiento cuando accedes simultáneamente a archivos que han sido distribuido en dos servidores independientes. En lugar de tener un servidor que sirve archivos, cuentas con dos proporcionado archivos. Ahora que ya hemos abordado la teoría, vamos a meternos de lleno en el desarrollo de estos volúmenes.

### Montar un volumen GlusterFS replicado distribuido

La Figura 2 muestra una foto de mi configuración. A la izquierda están los cuatro ODROID-HC1 apilados y a la derecha, el clúster ODROID-MC1. Ambos descansan sobre un switch Gigabit de 16 puertos al cual están conectados.


**Figura 2 – Entorno de laboratorio**

### Configurar los ODROID-HC1s

Necesitarás copiar la imagen del sistema operativo en tu tarjeta SD para iniciar tus HC1. Después, define una dirección IP estática y un nombre de host único para cada HC1. Puede que quieras recurrir a las instrucciones de cómo configurar el sistema operativo

en cada nodo del clúster de mi artículo sobre el MC1 en **http://bit.ly/2lrzVhb**. Cambia los nombres de host a xu4-gluster0, xu-4-gluster1, xu-4-gluster2, y así sucesivamente.

### Instalar y formatear el disco duro

Inserta los discos duros en los conectores SATA de tus HC1. Escribe "sudo -as" para acceder a los privilegios de root y crear una partición de Linux usando fdisk, luego crea un sistema de archivos ext4, tal y como se muestra en la Figura 3.

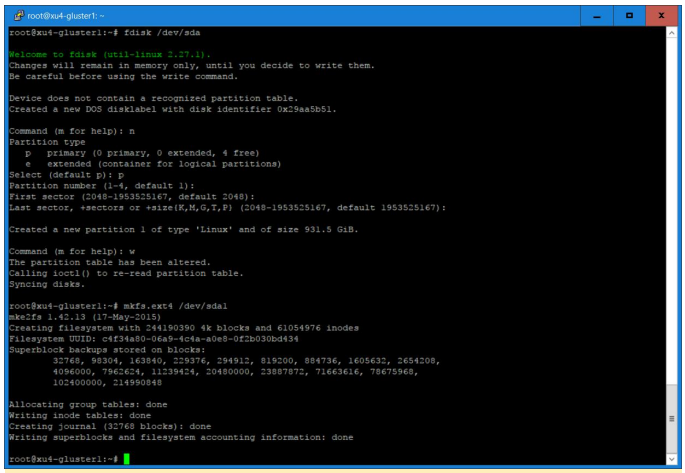



**Figura 3 – fdisk**

Crea un directorio llamado /gds/brick1, añade una entrada a /etc/fstab y monta el sistema de archivos. Puedes ver el resultado en la Figura 4.

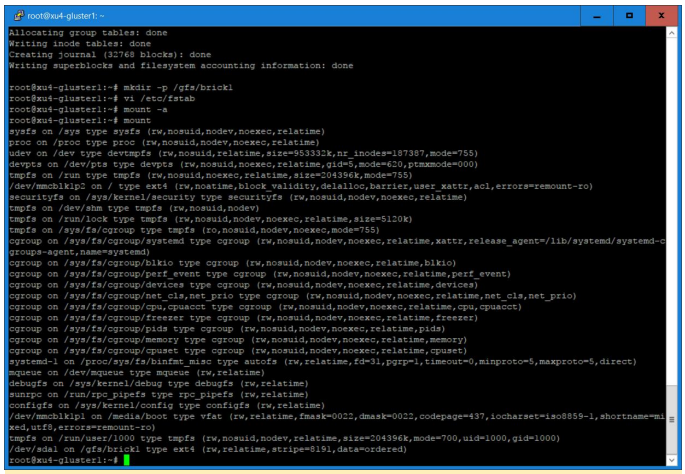



**Figura 4 – fstab**

Todo esto se hace con los siguientes comandos:

```
$ fdisk /dev/sda1
 $ mkfs.ext4 /dev/sda1
 $ mkdir -p /gfs/brick1
```

Añade la siguiente línea a tu /etc/fstab (sin las comillas): "/dev/sda1/gfs/brick1 ext4 defaults 0 1". A continuación, escribe los siguientes comandos:

```
$ mount -a
 $ mount
```

Instalar y configurar el volumen y el servidor Gluster

- Instalar el servidor GlusterFS
- Crear un grupo fiable de nodos GlusterFS. Un grupo de almacenamiento es una red fiable de servidores de almacenamiento. Antes de que uno pueda configurar un volumen GlusterFS, se debe crear un conjunto fiable (de almacenamiento) consistente en servidores de almacenamiento que proporcionen ladrillos a un volumen.
- Crear un directorio para el volumen
- Crear un volumen replicado distribuido llamado gvolume0
- Iniciar el volumen y mostrar su estado

Los comandos utilizados se resumen a continuación (se ejecutan como root):

- Ejecuta en todos los servidores HC1 los siguientes comandos:

```
$ apt-get update

$ apt-get install glusterfs-server attr
```

- Desde xu4-gluster0 (u otro servidor GlusterFS), introduce los comandos para crear un grupo fiable de nodos GlusterFS que está compuesto por nuestros 4 HC1:

```
$ gluster peer probe xu4-gluster1

$ gluster peer probe xu4-gluster2

$ gluster peer probe xu4-gluster3
```

- Crea el directorio: /gfs/brick1/gvolume0 en todos los servidores. A continuación, introduce los siguientes comandos gluster:

```
$ gluster volume create gvolume0 replica 2
transport tcp

$ xu4-gluster0:/gfs/brick1/gvolume0 xu4-
gluster1:/gfs/brick1/gvolume0

$ xu4-gluster2:/gfs/brick1/gvolume0 xu4-
gluster3:/gfs/brick1/gvolume0

$ gluster volume start gvolume0
```

```
$ gluster volume info gvolume0
```

En un volumen replicado distribuido, los archivos se distribuyen a través de conjuntos de ladrillos replicados. El número de ladrillos debe ser un múltiplo de la cantidad de réplicas, que en nuestro caso es dos. El orden en que se especifican los ladrillos es importante. En el comando "gluster volume create", los ladrillos adyacentes se convierten en réplicas entre sí. Este tipo de volumen proporciona alta disponibilidad a través de la réplica y el escalado a través de la distribución. En nuestro comando, usamos cuatro ladrillos y replicamos dos, lo cual hace que los dos primeros ladrillos se conviertan en réplicas de los otros. Este volumen suele denominarse 2 x 2. La Figura 5 muestra el resultado de algunos de los comandos.

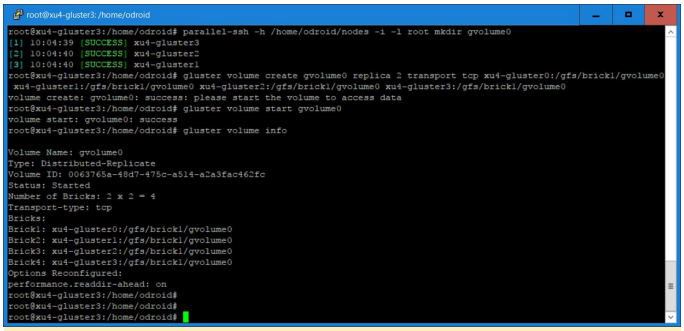



**Figura 5 – Crear el volumen**

Para aquellos que solo tengan dos HC1 y quieran crear un volumen replicado, simplemente tienen que usar el comando "gluster peer probe" en el otro servidor y reemplazar el comando "gluster volume create" por el siguiente comando:

```
$ gluster volume create gvolume0 replica 2
transport tcp xu4-
gluster0:/gfs/brick1/gvolume0 $ xu4-
gluster1:/gfs/brick1/gvolume0
```

### Probando el volumen usando Gluster Client

En otra máquina (en mi caso usé uno de los servidores del clúster ODROID-MC1), instalé el cliente GlusterFS como root y monté el volumen:

```
$ apt-get update
 $ apt-get install gluserfs-client attr
 $ mkdir /mnt/gfs
```

```
 $ mount -t glusterfs -oacl xu4-
gluster0:/gvolume0 /mnt/gfs
```

Ten en cuenta que, si deseas que el montaje sea permanente, debes añadir una entrada en el archivo /etc/fstab.

### Una simple prueba

Esta es una simple prueba que muestra la distribución de archivos en un volumen GlusterFS replicado distribuido.

Crea 100 archivos usando el comando:

```
$ cd /mnt/gfs
 $ mkdir testdir
 $ cd testdir
 $ for i in `seq 1 100`
 $ do
 $ echo message$i > file$i
 $ done
 $ ls
```

El resultado de estos comandos se muestra en la Figura 6.

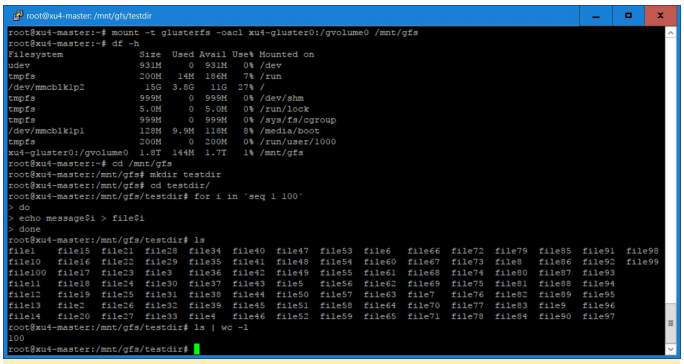



**Figura 6 – Archivos del clientes**

Inicie sesión en xu4-gluster0 e introduce los siguientes comandos:

```
$ cd /gfs/brick1/gvolume0/testdir
 $ ls
 $ ls | wc -l
```

Observarás en la Figura 7 que 46 de los 100 archivos están guardados en este servidor, dado que estamos utilizando un volumen replicado distribuido.

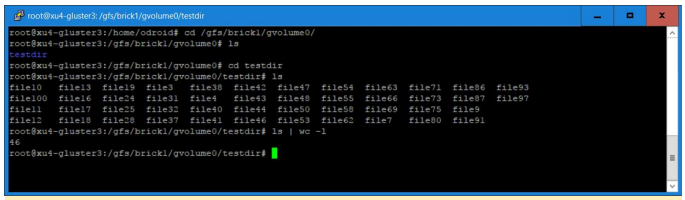



**Figura 7 – Archivos GlusterO**

Inicia sesión en xu4-gluster1 e introduce los mismos comandos:

```
$ cd /gfs/brick1/gvolume0/testdir
 $ ls
 $ ls | wc -l
```

Verás en la captura de pantalla que hay 54 archivos en este servidor. El total en ambos servidores suma los 100 archivos que creamos anteriormente. En nuestro volumen replicado distribuido, los 100 archivos se distribuyen entre los dos servidores, aunque no exactamente en una proporción de 50/50.

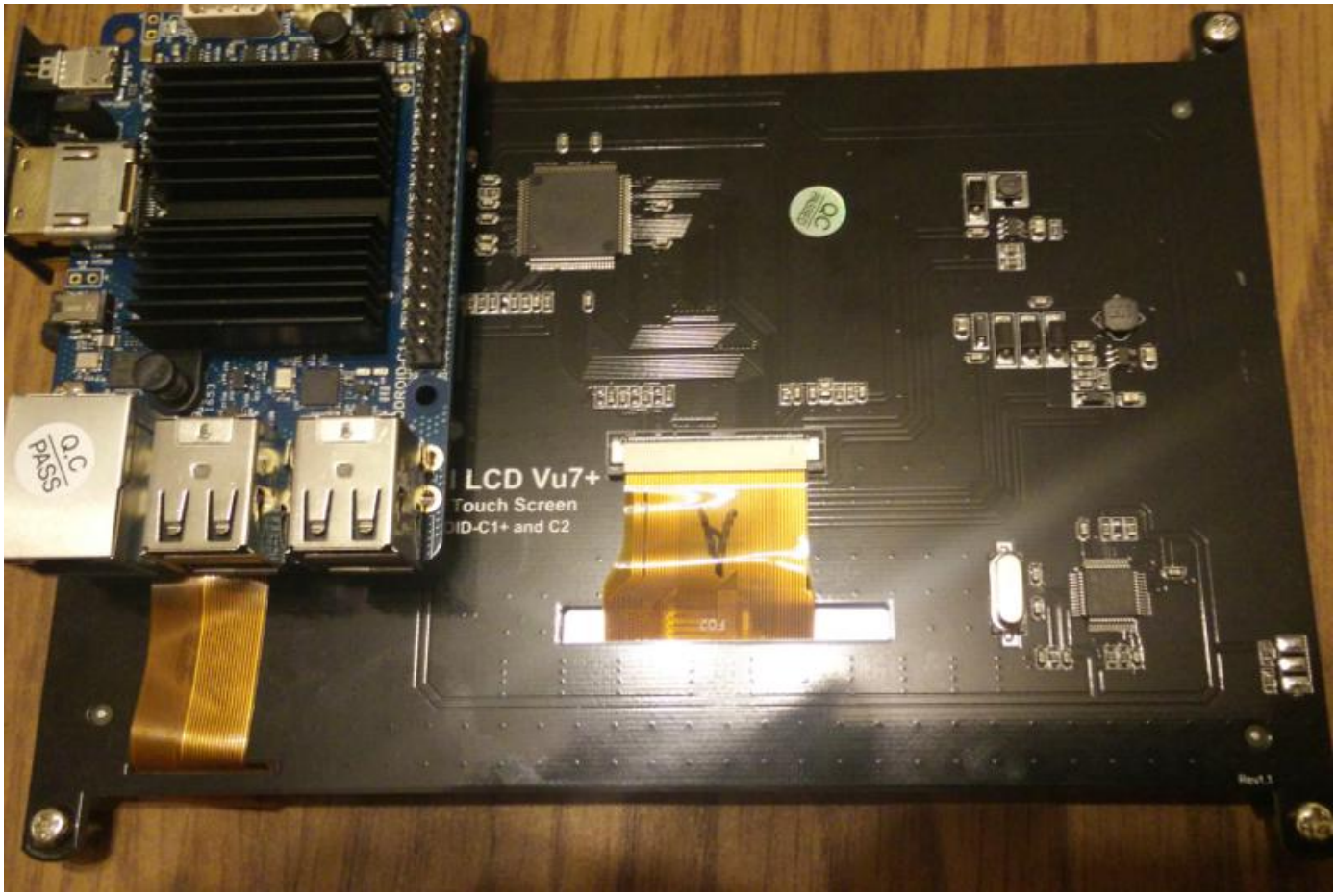
Encontrarás el mismo resultado si inicias sesión en los otros dos servidores (xu4-gluster2 y xu4-gluster3). Para aquellos que crearon un volumen replicado usando dos ODROID-HC1, verán los 100 archivos en ambos servidores ya que el suyo es un volumen replicado y no un volumen distribuido.

### ¿Qué será lo proximo?

He descrito cómo montar volúmenes GlusterFS replicados y distribuidos replicados usando dos y cuatros ODROID-HC1 respectivamente. También he mostrado cómo acceder al volumen GlusterFS usando un cliente GlusterFS. En la Parte 2 de este artículo, describiré cómo instalar y utilizar otros clientes, incluidos NFS y Samba (para Windows), para acceder al volumen GlusterFS y comparar el rendimiento de estos clientes.

# Regular el contraluz en ODROID-VU7+: Cómo controlar la iluminación de fondo en las plataformas Android ODROID-C1 y ODROID-C2

November 1, 2017 By Jörg Wolff ODROID-C1+, ODROID-C2, Mecaniqueo




Recientemente, he desarrollado un driver para controlar la luz de fondo para ODROID-C1 y ODROID-C2 usando PWM (pin 33). Para poder usarlo, debes copiarlo a la carpeta system/lib/hw/. Tras reiniciar, el driver debería funcionar correctamente.

El driver carga automáticamente los módulos del kernel para PWM, de modo que pwm-meson.ko y pwm-ctrl.ko deben estar presentes, como suelen estarlo normalmente. Ten en cuenta que, si utilizas este driver, sólo podrá usar el PWM y el pin 33 para la luz de fondo. El driver está disponible en http://bit.ly/2ysMPAS.

Para copiar el driver al ODROID-C1, introduce el siguiente comando desde un equipo host que ejecute Android Debug Bridge (ADB) conectado al ODROID-C1 a través de un cable USB:

```
$ adb push lights.odroidc.so /system/lib/hw/
```

Para que soporte ODROID-VU8, se debe añadir a boot.ini el argumento de arranque "backlight_pwm=yes|no|invert":

```
# Enable backlight_pwm
 # backlight_pwm=yes, no, invert
 backlight_pwm=yes

# Booting
 setenv bootargs "root=/dev/mmcblk0p2 rw
console=ttyS0,115200n8 no_console_suspend
vdaccfg=${vdac_config}
logo=osd1,loaded,${fb_addr},${outputmode},full
hdmimode=${hdmimode} cvbsmode=${cvbsmode}
hdmitx=${cecconfig} vout=${vout_mode}
disablehpd=${disablehpd} ${disableuhs}
androidboot.serialno=${fbt_id#}
ir_remote=${ir_remote} usbcore.autosuspend=-1
```

```
${selinuxopt}
suspend_hdmiphy=${suspend_hdmiphy}}
backlight_pwm=${backlight_pwm}"
```

El código fuente está disponible en https://github.com/joerg65/lights.

Para controlar la luz de fondo del VU7+, tendrás que hacer una pequeña modificación. En el pin 4 del controlador de luz de fondo PT4103, deberás soldar una resistencia. Yo utilicé una resistencia de 330 Ohm. No obstante, también debería funcionar sin ella. Independientemente de las especificaciones técnicas del 4103, el pin EN del 4103 en el VU7 + tiene una resistencia de 10k pull-up. Por lo tanto, la luz de fondo del VU7 siempre aparece activada. Medí la corriente con el pin EN conectado a GND. El resultado fue de unos 0.5mA, que proceden de la resistencia pull-up: 5V dividido por 10k. Cogí una resistencia y la pegué a la placa, soldándola con cuidado al pin 4. Esta se puede conectar al pin 33 del ODROID.

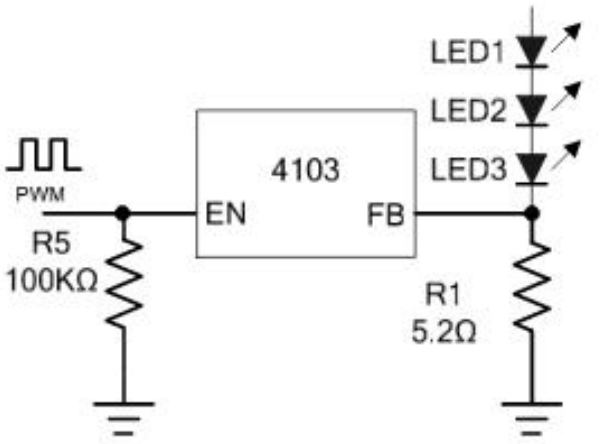



**Figura 1 – Diagrama esquemático**

**Figura 2 – Primer plano de la conexión soldada**

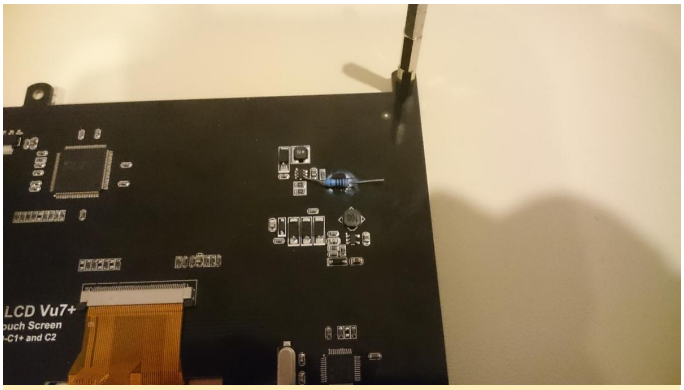


**Figura 3 – Vista panorámica de la conexión soldada**

Para ver un video detallado del proyecto, consulta t https://youtu.be/mVvnLiKiksw. Para comentarios, preguntas y sugerencias, visite la post original en https://forum.odroid.com/viewtopic.php?f=113&t=27227.

# Conociendo un ODROIDian: Laurent Caneiro

November 1, 2017 By Laurent Caneiro (@tipoto) Conociendo un ODROIDian

*Por favor, háblanos un poco sobre ti.*

Tengo 40 años y nací en Burdeos, Francia. Actualmente vivo en Los Ángeles, California, con mi maravillosa esposa Nathalie y mi hija Hanaé, que es mayor. Trabajo para los estudios de animación de DreamWorks como animador. He trabajado en la industria de la animación desde el 2000. He trabajado en películas como "Cómo entrenar a tu dragón 1 y 2", "Gato con botas", "Kung Fu Panda 3", "Trolls" y algunas más. Actualmente estoy trabajando en "Cómo entrenar a tu Dragon 3" que se lanzará en marzo de 2019. Antes de DreamWorks, trabajé para varios estudios de animación en Luxemburgo, Francia, Bélgica y España, lo cual me permitió descubrir diferentes países al mismo tiempo que enriquecía a mi experiencia.

**Figura 1 – Laurent en su estación de trabajo en DreamWorks Animation**

*¿Cuál es tu formación académica?*

No era muy buen estudiante de joven, lo cual me puso en una situación delicada al final de la escuela secundaria. Básicamente no pude elegir la rama que quería, y mi única opción era elegir entre estudios de secretaría o contabilidad. Opté por la contabilidad, a pesar de que ni siquiera sabía de qué se trataba. No obstante, estudié 4 años contabilidad de todas formas y finalmente obtuve mi diploma.

Afortunadamente, nunca dejé de dibujar y pintar durante mi tiempo libre, y tras mis estudios de contabilidad, decidí probar con el examen de ingreso en una escuela de animación de Luxemburgo (L.T.A.M.) y ¡lo pasé! Fue entonces cuando me enamoré de la animación.

*¿Cómo empezaste con los ordenadores?*

Empecé con los ordenadores cuando tenía 8 o 9 años. Por aquel entonces, tenía un ordenador llamado Thompson TO7-70, que creo que sólo se lanzó en Francia. Principalmente jugaba a juegos con él, aunque mi iniciación en la programación también empezó en ese momento. El ordenador usaba el lenguaje BASIC, y como mis padres estaban suscritos a una revista BASIC mensual, pude aprender algunas cosas y empezar a juguetear con el lenguaje.

**Figura 2 – El primer ordenador de Laurent, una Thompson T07-70**

No soy programador y mis conocimientos son bastante limitados en esta área, pero me fascina este mundo. Soy autodidacta y me gusta escribir pequeñas herramientas útiles que me faciliten mi día a día en el trabajo. Principalmente uso el lenguaje script Shell, aunque también uso en ocasiones Python, Squirrel, C/C ++ y otros lenguajes.

*¿Qué te atraje a la plataforma ODROID?*

ITenía en mente un proyecto que realmente quería hacer realidad. Primero compré una Raspberry Pi 2, que es un excelente microcontrolador, pero no era lo suficientemente potente para cubrir a mis necesidades, así que decidí investigar otras placas con el fin de reemplazarla. Encontré el sitio web de Hardkernel y descubrí el XU4. Me impresionaron sus especificaciones técnicas, pero también quería saber si la comunidad era lo suficientemente grande, así que consulté los foros de ODROID e hice bastantes lecturas. Tras unas horas llegue a la conclusión de que se trataba de un foro muy activo y que sus miembros parecían siempre estaban dispuestos a ayudar, además de ser técnicamente excelentes. Decidí comprar un XU4 y migrar mi proyecto actual a esta placa.

*¿Cómo utilizas tu ODROID?*

Uso mi ODROID como consola de juegos retro. He estado trabajando en este proyecto desde 2015, soy bastante lento. Mi proyecto está dividido en dos piezas de hardware, una es la consola en sí y la otra es un joystick arcade para 2 jugadores. Para la consola, uso una pequeña carcasa de PC que modifiqué ligeramente, diseñé una placa de acrílico en la que conecté todos mis componentes (XU4, Hub USB, regulador de voltaje, HDD).

**Figura 3: Laurent ha estado desarrollando una consola de juegos personalizada desde 2015**

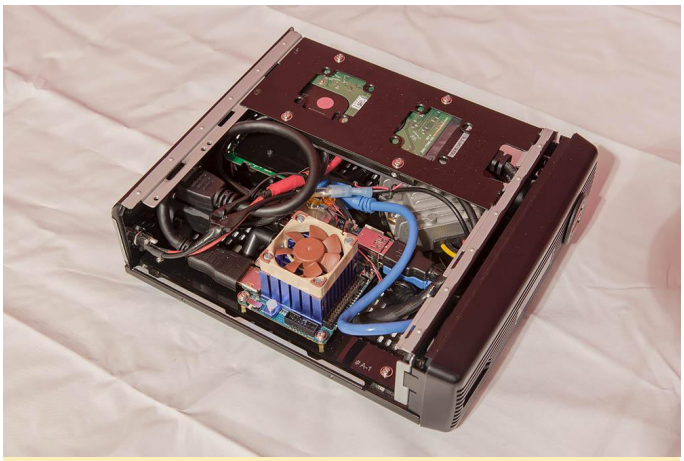


**Figura 4 – Interior de la consola de juegos hecha a medida por Laurent**

En el caso de los joysticks, los diseñé por completo desde el principio, y también utilicé láminas acrílicas.

Utiliza un Arduino combinado con una placa módulo RGB para controlar todos los leds RGB de los botones. Usa un IPAC2 para la comunicación entre los botones y el XU4. La parte del hardware está totalmente finalizada y es completamente funcional, ahora estoy trabajando en el tema del software. Empecé con la imagen OGST de @meveric, pero estoy personalizándolo todo añadiendo un monto de características con el fin de darle un aspecto único y atractivo.

**Figura 5 – ¡Los joysticks que Laurent diseñó para su consola de juegos son una autentica obra de arte!!**

*¿Cuál es tu ODROID favorito y por qué?*
Resulta difícil contestar, ya que solo poseo el ODROID-XU4, aunque realmente adoro esta placa.

*¿Qué innovaciones te gustaría ver en futuros productos de hardkernel?*
Me gustaría ver una placa con un módulo Wifi integrado, un sistema de refrigeración eficiente que evitase cualquier obstrucción cuando las CPU funcionase al 100% y un excelente soporte GPU en Linux.

*¿Qué pasatiempos e intereses tienes a parte de los ordenadores?*
Disfruto con la fotografía y suelo tocar la guitarra y el piano.

*¿Qué consejo le darías a alguien que quieres aprender más sobre programación?*
Empezar con un lenguaje amigable que tenga un excelente soporte en Internet, para que siempre puedas encontrar soluciones a tus problemas. Ponte a prueba con pequeños proyectos con los que te diviertas. No intentes ir demasiado rápido, necesitas retarte a ti mismo paso a paso, de este modo podrás aprender y mantenerte al mismo tiempo motivado. No querrás sentirte abrumado con algo que sea demasiado complicado como proyecto inicial. Si planificas tus progresos lentamente, siempre estarás listo para resolver el siguiente problema al que te enfrentes.

Yo lo que hago personalmente es lo que denomino “un estampado” antes de empezar a “elaborar” código, que no es otra cosa que crear una vista esquemática de lo que quiero hacer en papel, de modo que tenga una visión global de todo mi código antes de iniciarlo. También recomiendo usar Internet tanto como puedas como recurso educativo.

Puedes conseguir más sobre Laurent y su porfolio visitando su sitio web en **http://www.laurentcaneiro.com/**, y su perfil de IMDB en **http://www.imdb.com/name/nm3140121/**.