Lakka • Punto G • Midi UART • Monku R3 • Shell OGO • Ajustes XU4

ODROID Magazine

Años Seis
Mum. #71
Nov 2019

CONFIGURA TUS ODROIDS PARA EXPANDIR TU PANTALLA

ESCRITORIOS *Multipantalla*

MACINTOSH PLUS:
EL CLASICO ORDENADOR AHORA EN EL ODROID-GO

MODULOS KERNEL:
CONOCE EL NUCLEO DE TU SISTEMA

MODULOS EMMC:
¡COMPRA LA MEMORIA CORRECTA PARA TU MAQUINA Y DISFRUTA DE LA VELOCIDAD!

Reajustando el ODROID-XU4: Un Conjunto de Conocidas Modificaciones

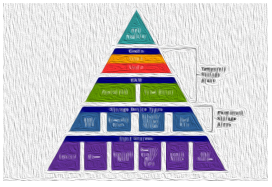
November 1, 2019

Este artículo es una traducción autorizada del artículo alemán " ODROID-XU4: Tweaks unter Ubuntu 18.04 und Kernel 4.14".

Recreando un Mac Plus: Usando ODROID-GO como un Emulador Macintosh

November 8, 2019

Para los fanáticos de Macintosh Plus, ¡ahora existe una forma de ejecutar un emulador en ODROID-GO! Basado en el trabajo de spritesmods.com minimacplus, este proyecto aprovecha el hecho de que tanto el proyecto original como el GO usan un microcontrolador ESP32.



Módulos de Memoria eMMC: Una Simple Guía

November 9, 2019

Alguna vez te has preguntado cómo usar todo el potencial del módulo eMMC en tu dispositivo ODROID, quizás hayas pensado que es demasiado complicado para ti. Puede resultar un poco espeluznante si eres nuevo en estos dispositivos. Además, ¿cómo usas uno? ¿Qué herramientas necesitas?

Ogo Shell

November 1, 2019

La utilidad ogo-shell es un explorador de archivos, reproductor de audio y visor de imágenes para ODROID-GO. Principalmente lo uso para escuchar música usando los auriculares odroig-go de backofficeshow. Puede consultar el Proyecto Github y la publicación del foro si quieres probarlo. La siguiente es la historia de su desarrollo ▶️

Escritorios Multipantalla con VNC

November 1, 2019

Hace unos dos años tuve una idea algo descabellada: ¿Sería posible crear un sistema de escritorio de "doble pantalla" utilizando dos ODROID, cada uno con una pantalla diferente, pero que actuasen como un escritorio unificado?

Monku R3: Desarrollando la Mejor Consola de Juegos ODROID-XU4/XU4Q - Parte 2

November 1, 2019

Este artículo te mostrará en detalle cómo pulir tu consola de videojuegos Monku Retro 3 (ODROID-XU4).

Juegos Linux: Anbox - Android en una Caja

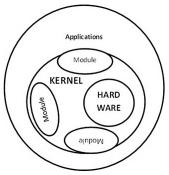
November 1, 2019

Aunque normalmente solo suelo hablar de las placas ODROID basadas en ARM, hoy quiero detenerme en el ODROID-H2 con el que se puede hacer algo muy interesante. Puesto que ODROID-H2 es una placa estándar x86_64 (amd64), puedes llevar a cabo exactamente los mismos pasos de configuración en cualquier otro PC ▶️

El Punto G: Tu Destino para Todas las Cuestiones Relacionadas con Juegos Android

November 1, 2019

Play Pass: ¿Por qué Google trata de cubrir preventivamente el período previo al estreno de Stadia al lanzar un servicio indudablemente inferior? Play Pass es un servicio de suscripción mensual que te permite descargar y jugar una amplia biblioteca de juegos y otras aplicaciones de productividad que están actualmente disponibles ▶



Modulos Kernel

November 1, 2019

Si has detenido en los Foros de Hardkernel el tiempo suficiente, la expresión "módulo kernel" debería serte bastante familiar. Sin embargo, si eres nuevo en el mundo de Linux, los detalles sobre qué son exactamente los módulos de kernel podrías no tenerlos muy claros. El objetivo de este artículo no ▶

ODROID-N2 UART Tasa de Baudios Personalizada para MIDI

November 1, 2019

Necesitaba el puerto UART de mi ODROID-N2 para trabajar con una tasa de baudios no estándar y así poder usar MIDI. Edité el código del driver UART para fijar la velocidad de baudios en 31250 cuando configuré la tasa de baudios en 38400, luego escribí el código de prueba usando ▶

Juegos en ODROID-H2: Ejecutando Lakka en ODROID-H2

November 1, 2019

El número del mes pasado de ODROID Magazine incluía el artículo "Lakka: Desarrollando la mejor consola de juegos ODROID-XU4 / XU4Q", disponible en https://magazine.odroid.com/article/lakka-building-the-ultimate-odroid-xu4-xu4q-gaming-console/. Este artículo estaba centrado en los juegos con un ODROID-XU4, sin embargo, existe un ODROID potencialmente más potente, el H2. Mientras que el ODROID-XU4 hace un trabajo ▶

Reajustando el ODROID-XU4: Un Conjunto de Conocidas Modificaciones

November 1, 2019 By Dennis Paul ODROID-XU4, Mecaniqueo, Tutoriales

Como muchos otros, estoy interesado en aprovechar al máximo mi ODROID-XU4. Durante el tiempo que lo he usado, he descubierto varios ajustes que mejoraron la velocidad o disminuyen el consumo de energía. En este artículo me gustaría compartir lo que he aprendido. Este artículo es una traducción autorizada del artículo alemán " ODROID-XU4: Tweaks unter Ubuntu 18.04 und Kernel 4.14". El artículo original en alemán lo puedes encontrar aquí **https://blaumedia.com/blog/odroid-xu4-tweaks-unter-ubuntu-18-04-und-kernel-4-14/**

Nota: Cualquier cambio en tu sistema puede anular tu garantía y afectar negativamente la estabilidad del sistema, así que haz los cambios bajo tu propia responsabilidad.

Pruebas de rendimiento y overclocking de la CPU

Hardkernel desaconseja cambiar los valores de la velocidad de reloj de la CPU del ODROID-XU4, ya que puede causar problemas de estabilidad y sobrecalentamiento. Para evitar este último problema, recomiendo cambiar el ventilador de viene de serie, tal y como describo en mi artículo: https://blaumedia.com/blog/odroid-xu4-luefter-und-kuehlkoerper-austausch/. El problema más habitual es el estrangulamiento térmico, que conlleva un overclocking automático del procesador para disminuir la posibilidad de dañar el hardware. Desafortunadamente, los problemas de estabilidad se dejan de lado inevitablemente hasta que crees una configuración que funcione perfectamente.

Por defecto, los cuatro núcleos pequeños son los núcleos ARM más eficientes del procesador, y los cuatro núcleos GRANDES, núcleos centrados en el rendimiento, funcionan con una velocidad de reloj máxima de 1,5 GHz y 2,0 GHz, respectivamente. Voy a explicar a continuación cómo personalizar y compilar el kernel para llegar a los 1.6 GHz y 2.1 GHz como

velocidades máximas de reloj. No me te topado con ningún problema o fallo durante la semana que han durado mis pruebas, percibiendo un funcionamiento bastante estable.

Para poner de manifiesto el aumento del rendimiento, lleve a cabo una pequeña prueba de rendimiento con sysbench. Aquí tienes los primeros resultados (menos es mejor):

Tipo Nucleos	Resultados pruebas sin overclock	Resultados pruebas con overclock	Diferencia
little.cores	35.644 seconds	33.408 seconds	-6.27%
BIG.cores	21.695 seconds	20.624 seconds	-4.94%
Both	13.583 seconds	12.994 seconds	-4.34%

Como puedes ver, con un aumento de solo 0.1 GHz obtendrá un rendimiento bastante mejor. Especialmente en los núcleos pequeños porque su reloj base es de solo 1.5 GHz. Los comandos del terminal son:

little.cores

```
$ taskset -c 0,1,2,3 sysbench --test=cpu --num-threads=4 run
```

BIG.cores

```
$ taskset -c 4,5,6,7 sysbench --test=cpu --num-threads=4 run
```

both cores

$ sysbench --test=cpu --num-threads=8 run

Ejecuté tres veces todos los comandos y tomé como resultado la media. Por supuesto, el regulador de la CPU estaba configurado en “performance”. ¡Aunque es suficiente para llevar a cabo las pruebas de rendimiento! ¿Cómo podemos “overclockear” la CPU? Gracias a los comentarios de Ridge en el código del kernel, ha escrito un pequeño tutorial, disponible aquí: https://forum.odroid.com/viewtopic.php?f=93&t=30115 Del manual copié el código necesario; simplemente inicia sesión como root y copia el siguiente script en tu terminal. La compilación puede demorarse hasta unos 30 minutos, así que mientras tanto, ve a por un café y revisa tu correo electrónico.

```
# Last update: 27. July 2019

# Cloning Git Repo
cd /tmp
git clone --depth 1
https://github.com/hardkernel/linux -b odroidxu4-
4.14.y
cd linux

# paste the new clock rates
sed -i 's/<1500000000>/<1600000000>/g'
arch/arm/boot/dts/exynos5422-cpus.dtsi
sed -i 's/<2000000000>/<2100000000>/g'
arch/arm/boot/dts/exynos5422-cpus.dtsi

sed -i '/&cluster_a15_opp_table {/a
opp-2100000000 {
opp-hz = /bits/ 64 <2100000000>;
opp-microvolt = <1312500>;
clock-latency-ns = <140000>;
};' arch/arm/boot/dts/exynos5800.dtsi

sed -i '/&cluster_a7_opp_table {/a
opp-1600000000 {
opp-hz = /bits/ 64 <1600000000>;
opp-microvolt = <1250000>;
clock-latency-ns = <140000>;
};' arch/arm/boot/dts/exynos5800.dtsi

sed -i '/PLL_35XX_RATE(2000000000, 250, 3, 0),/i
PLL_35XX_RATE(2100000000, 175, 2, 0),'
drivers/clk/samsung/clk-exynos5420.c

sed -i '/{ 2000000, E5420_EGL_DIV0(3, 7, 7, 4),
},/i
{ 2100000, E5420_EGL_DIV0(3, 7, 7, 4), },'
drivers/clk/samsung/clk-exynos5420.c

sed -i '/{ 1500000, E5420_KFC_DIV(3, 5, 3), },/i
{ 1550000, E5420_KFC_DIV(3, 5, 3), },'
drivers/clk/samsung/clk-exynos5420.c

# compile kernel, readme:
https://wiki.odroid.com/odroid-
xu4/software/building_kernel#y
apt update && apt install -y git gcc g++ build-
essential libssl-dev bc
make odroidxu4_defconfig
make -j8
make modules_install
```

```
cp -f arch/arm/boot/zImage /media/boot
cp -f arch/arm/boot/dts/exynos5422-odroidxu3.dtb
/media/boot
cp -f arch/arm/boot/dts/exynos5422-odroidxu4.dtb
/media/boot
cp -f arch/arm/boot/dts/exynos5422-odroidxu3-
lite.dtb /media/boot
sync
```

Cuando la compilación haya terminado, puedes reiniciar el ODROID-XU4 escribiendo el comando "reboot " en tu terminal. Al arrancar, se cogerá el nuevo kernel y se usaran las nuevas frecuencias de reloj que hemos definido.

Prueba de rendimiento y Overclocking de la RAM

La velocidad de reloj de la RAM determina la velocidad de los datos. En otras palabras: cuanto más rápida sea la frecuencia de reloj, mayor será el ancho de banda. Hardkernel ha facilitado el overclock de la RAM ODROID-XU4. Por defecto, funciona a 825 MHz y es extremadamente fácil “overclockearla” a 933 MHz. Todo lo que tiene que hacer es buscar la línea setenv ddr_freq 825 en /media/boot/boot.ini y reemplazar el 825 por 933. De cualquier modo, aquí tienes el código listo para que lo utilices:

```
$ sed -i 's/setenv ddr_freq 825/setenv ddr_feq
933/' /media/boot/boot.ini
```

¿Qué mejoras de rendimiento podemos esperar? Nuevamente he realizado una pequeña prueba de rendimiento con sysbench. He utilizado el siguiente comando:

```
$ sysbench --test=memory --memory-block-size=1K --
memory-total-size=10G --num-threads=1 run
```

El tiempo total de ejecución mejoró, paso de los 10.733 segundos a los 10.592 segundos, un cambio de -1.31%. En este caso, debemos decidir si aceptamos un mayor consumo de energía a cambio de esta "mejora del rendimiento". Yo, por mi parte, he habilitado los 933 MHz. Los pequeños ajustes de ODROID-XU4 proporcionan un buen rendimiento por lo general 🙂

Mejorar la velocidad de E/S

Asignamos los puertos USB3.0 y el puerto Ethernet a BIG.cores Sirva de fuente la siguiente contribución, gracias a Obihoernchen y a la publicación del blog: https://obihoernchen.net/1416/odroid-xu4-tune-network-and-usb-speed/ Lo siguiente procede de su publicación en el blog y la he utilizado para reajustar el ODROID-XU4. ¡Definitivamente deberías echar un vistazo a su blog!

Por defecto, las tareas/eventos creados por los puertos Ethernet o USB (interrupciones) se distribuyen a todos los núcleos. Por lo tanto, puede suceder que a little.core se le asigne la tarea de descargar 10 GB de Internet, donde un BIG.core podría completarlo en un tiempo mucho más corto. Para asegurarnos de que BIG.cores haga esto en el futuro, debes asegurarte desde principio de que el servicio irqbalance esté desactivado:

```
# systemctl disable irqbalance
```

Luego abrimos el archivo /etc/rc.local e insertamos lo siguiente antes de la salida 0:

usb2

```
# echo 6 > /proc/irq/103/smp_affinity_list
```

usb3

```
#echo 5 > /proc/irq/104/smp_affinity_list
```

network (usb3)

```
#echo 4 > /proc/irq/105/smp_affinity_list
```

Según las pruebas de rendimiento de Obihoernchen, las transferencias llegaron a alcanzar los 100 Mbit (12.5 MB/s). Para mí, esta configuración se ha convertido en un estándar.

Usando UASP para discos duros USB 3.0

Este es un consejo general y no incluye código de ejemplo, pero es uno de mis ajustes favoritos en el ODROID-XU4. Tal y como se describe y ha sido probado en mi artículo ODROID-XU4: SSD vs. eMMC comparison (Boost!), https://blaumedia.com/blog/odroid-xu4-ssd-vs-emmc-boost/ Por aquel entonces, solía recomendar encarecidamente usar un SSD con un adaptador SATA a USB que admitiese UASP. Aquí tienes una tabla de ese artículo:

Tipo	Lectura	Lectura	Escritura

	(Cache)	(Direct)	
eMMC	804.54 MB/s	155.44 MB/s	45.0 MB/s
SSD (sin UASP)	890.57 MB/s	106.09 MB/s	125.00 MB/s
SSD (con UASP)	888.20 MB/s	343.56 MB/s	205.00 MB/s

Los adaptadores UASP mejoran bastante el rendimiento.

Reduciendo el consumo de energía.

Muchas personas seguramente utilizan el ODROID-XU4 como un pequeño servidor doméstico permitiéndoles ahorrar energía, como yo. Para reducir un poco el consumo de energía en el funcionamiento 24/7, aquí tienes algunos consejos.

Desactivando núcleos de la CPU

Prácticamente no requiere ningún esfuerzo desactivar los núcleos individuales. Pero primero debes saber que CPU0-CPU3 es tu little.cores y CPU4-7 son tus BIG.cores. Dependiendo de las necesidades de tu ODROID-XU4, debes considerar qué núcleos te gustaría desactivar.

Reemplaza el 0 en cpu0 por el núcleo deseado

```
# echo 0 > /sys/devices/system/cpu/cpu0/online
```

Downclocking de la GPU

De nuevo, el siguiente consejo procede del blog de Obihoernchen. Esta vez se trata de bajar la frecuencia de reloj de la GPU, el procesador de gráficos del ODROID-XU4. En modo servidor, siempre se ejecuta a una velocidad de reloj máxima de 600 MHz; sin ser usada ¡El blog Obihoernchen también incluye una evaluación en este sentido y probablemente se podrá lograr un ahorro de energía de entorno un 20%!

Para alcanzar este beneficio, debe instalar el paquete sysfsutils con APT:

```
# apt update && apt install sysfsutils
```

Luego tenemos que agregar lo siguiente en /etc/sysfs.conf dependiendo de tu núcleo (con uname -a puede averiguar con cuál has arrancado):

si el kernel es el 3.10 inserta lo siguiente:

```
# devices/11800000.mali/dvfs_max_lock = 177
```

si el kernel es el 4.9 inserta lo siguiente:

```
# devices/platform/11800000.mali\:/devfreq/11800000.mali\:/governor = powersave
```

si el kernel es el 4.4 inserta lo siguiente:

```
# devices/platform/11800000.mali/devfreq/devfreq0/governor = powersave
```

Con un último "systemctl restart sysfsutils" la mínima frecuencia del reloj queda activado.

Recreando un Mac Plus: Usando ODROID-GO como un Emulador Macintosh

November 8, 2019 By @johannesbehr ODROID-GO, Tutoriales

Para los fanáticos de Macintosh Plus, ¡ahora existe una forma de ejecutar un emulador en ODROID-GO! Basado en el trabajo de **https://spritesmods.com/?art=minimacplus**, este proyecto aprovecha el hecho de que tanto el proyecto original como el GO usan un microcontrolador ESP32.

Puedes encontrar mi adaptación en **https://github.com/johannesbehr/minimacplus/tree/master/firmware**, con una versión FW disponible en **https://github.com/johannesbehr/minimacplus/tree/master/firmware/release**. Para usarla, debes copiar dos archivos en una carpeta llamada "/roms/macplus" en el directorio raíz de la tarjeta SD:

- mac.rom => Coge "1986-03 - 4D1F8172 - MacPlus v3.ROM" de **https://www.macintoshrepository.org/7038-all-macintosh-roms-68k-ppc-**
- mac_hd.raw => Puedes crear este archivo usando mess0161b siguiendo las instrucciones de Spritemod en **https://github.com/Spritetm/minimacplus**.

Configuración simplificada

Para una instalación mínima y rápida, puede descargar el firmware desde **https://github.com/johannesbehr/minimacplus/tree/master/firmware/release**, luego coloca al menos dos archivos en una carpeta llamada "/roms/macplus" en el directorio raíz de la tarjeta SD:

- Un archivo *.rom
- Al menos una imagen de disco de arranque (puede ser *.image, *.drv, *.hfv, *.raw or *.dsk)

Como ejemplo, el archivo ROM podría ser "1986-03 - 4D1F8172 - MacPlus v3.ROM", que puede extraerse descargando el archivo Old_World_Mac_Roms.zip del Repositorio de Macintosh en **https://www.macintoshrepository.org/703 ... s-**

68k-ppc-. La imagen del disco podría ser "System Tools.image" extraída de OS-6-06-a15-90-08-01-Ray-Ban.zip
en https://www.macintoshrepository.org/1778-mac-system-6-x.

Figura 1: Este es un proyecto muy interesante, si posees un dispositivo GO, ¡debería probarlo!

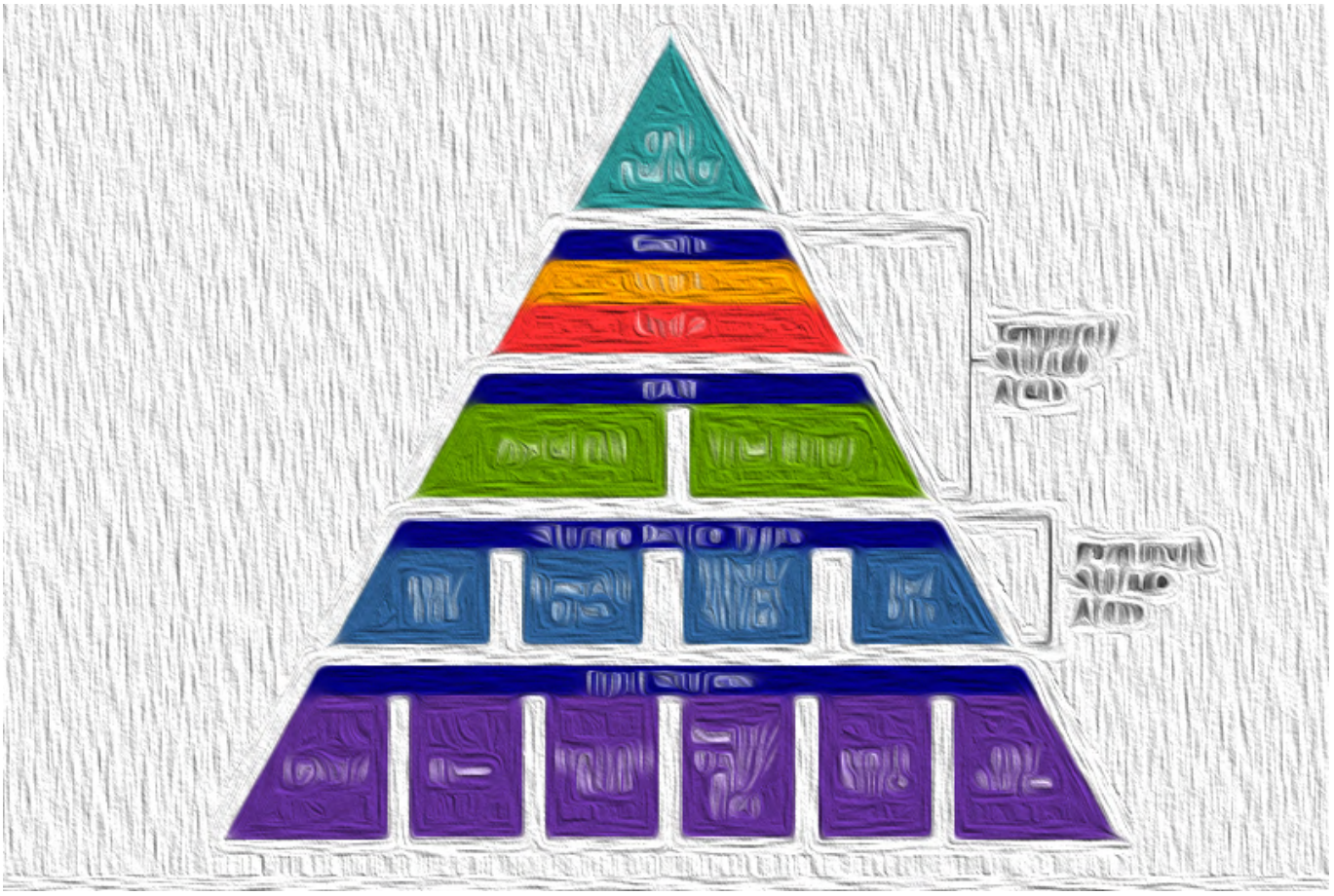
También he hecho algunas mejoras, que incluyen:

- Desplazamiento virtual, para que la pantalla siga al ratón
- Un teclado en pantalla, que se puede activar con la tecla "B"
- Mejoras de velocidad

Tienes total libertad para crear tus discos con buenos programas que puedes encontrar en la web. Puede usar hasta 4 imágenes de disco (cada una con un tamaño máximo de 20 MB) al mismo tiempo. Puedes usar el excelente emulador de https://www.gryphel.com/c/minivmac/ para preparar tus discos en un PC y luego copiarlos a tu ODROID-GO. Para comentarios, preguntas y sugerencias, visita la publicación original del foro en https://forum.odroid.com/viewtopic.php?f=162&t=36599.

Módulos de Memoria eMMC: Una Simple Guía

November 9, 2019 By Brian Ree Mecaniqueo, Tutoriales



Alguna vez te has preguntado cómo usar todo el potencial del módulo eMMC en tu dispositivo ODROID, quizás hayas pensado que es demasiado complicado para ti. Puede resultar un poco espeluznante si eres nuevo en estos dispositivos. Además, ¿cómo usas uno? ¿Qué herramientas necesitas? ¿Voy a estropearlo y tirar 30$ a la basura? Tal vez debería seguir con la ranura para tarjetas SD, ¿verdad? Pues bien, si eres como yo, es posible que tus reflexiones sean similares a mías. Este es un tutorial para ti. Es breve y directo, y lo cubriré utilizando módulos eMMC para un ODROID-XU4, de principio a fin. Puedes tomar estos conocimientos y aplicarlos a cualquiera de tus dispositivos que admitan módulos eMMC.

Haciendo uso de tu propio hardware, necesitarás un ODROID-C1 +, C2 o XU4. El tutorial se centra específicamente en un ODROID-XU4. Seamos realistas: el ODROID-XU4 es asombroso, es bastante potente y hace que cualquier SBC las pase canutas. Entonces, ¿por qué no actualizarlo y llevarlo al siguiente nivel con un módulo eMMC como unidad de arranque? También necesitarás un módulo eMMC, un adaptador eMMC y un adaptador microSD. Ahora existen adaptadores eMMC que se conecta directamente al USB, pero como yo uso tantas tarjetas microSD, decidí usar un adaptador que tenía por ahí y me hice con un adaptador eMMC a microSD. Los componentes básicos que excluyen el dispositivo ODROID te costarán alrededor de 38$. Estos incluyen un adaptador dual microSD SD a USB, un adaptador eMMC a microSD y un módulo eMMC de 32GB. Si ya cuentas con un adaptador microSD a USB reducirás el precio a unos 27$. No está mal. La unidad de 32 GB nos proporciona mucho espacio para el sistema operativo, sus actualizaciones, etc. Su mejor rendimiento propiciará que la unidad arranque y funcione más rápido en todos los sentidos. ¿Qué más se puede pedir?

Componentes necesarios

Los siguientes componentes los puedes obtener en la tienda Hardkernel (**https://www.hardkernel.com/product/**) o través de un dritribuidor local/regional:

- A Monku Retro 1,2,3/ODROID-C1 +, C2, XU4 - Comprueba y asegúrate de tener los módulos eMMC correctos para tu dispositivo
- Un módulo de memoria eMMC de 32 GB compatible
- Un adaptador microSD a USB
- Un adaptador eMMC a microSD

Preparando el módulo eMMC

Echemos un vistazo a los componentes con los que vamos a trabajar. La siguiente imagen muestra el módulo eMMC XU4 de 32 GB y el adaptador eMMC a microSD.

Figura 1

Dedica un momento a inspeccionar el adaptador eMMC a microSD. ¿Ves que hay un pequeño círculo blanco en una esquina de los pines de conexión? La siguiente imagen muestra una flecha roja apuntando al círculo.

Figura 2

Ahora echa un vistazo al módulo eMMC. ¿Ves el pequeño triángulo blanco en una esquina del módulo? Observa la siguiente imagen. La marca blanca en cuestión está indicada con una flecha roja.

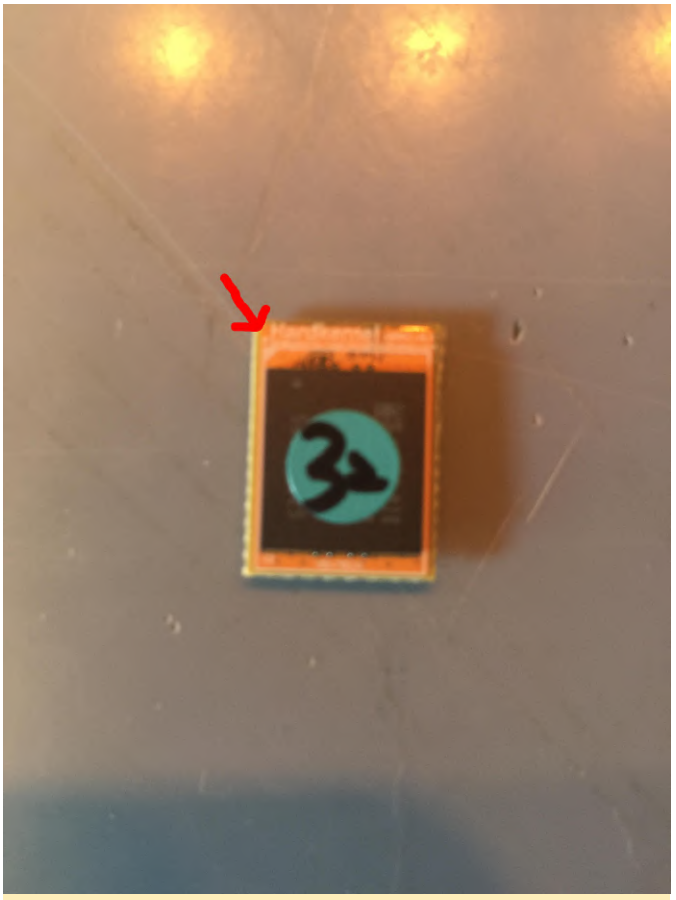


Figura 3

Figura 4

Lo que vamos a hacer ahora es conectar el módulo eMMC al adaptador microSD. Deberás alinear el triángulo blanco y el círculo blanco para que estén en el mismo lado de los pines del conector. No coincidirán en la misma esquina, al menos los componentes que yo tengo, pero estarán en el mismo lado. Deberás mantener ambos elementos en ángulo y dejar que los pines del conector se encuentren y luego juntarlos lenta y suavemente. Prácticamente puedes deslizar el módulo eMMC sobre los pines de conexión del adaptador. Oirás algo así como chasquido leve y gratificante, una vez que estén conectados correctamente.

Estamos listos para actualizar nuestro módulo eMMC. En el siguiente paso, grabaremos una imagen del sistema operativo del sitio ODROID.

Grabando el módulo eMMC

Ahora que tenemos nuestro módulo eMMC preparado y listo, nos dirigimos al sitio de ODROID donde podemos encontrar todas las imágenes de sistemas operativos. Nos centraremos en las imágenes del sistema operativo Linux para este tutorial.

El enlace directo para las imágenes XU4 es **https://bit.ly/32GnmQi** y el enlace general para las imágenes XU4 es **https://bit.ly/2v6FYcV**.

Puedes usar el segundo enlace para buscar imágenes para otros dispositivos como ODROID-C1+ o C2. Simplemente localiza el dispositivo en el panel izquierdo de la página. Usaremos una imagen Ubuntu 18.04 un poco más antigua, una que he usado antes, así que me siento cómodo con ella. La imagen en concreto que descargué yo para mis pruebas en el XU4 fue el archivo de imagen ubuntu-18.04.1-4.14-mate-odroid-xu4-20181203.img.xz.

Una vez que tengas tu imagen lista, es hora de descargar algún software que puedas usar para pasar la imagen al módulo eMMC. Si está utilizando un Mac, te recomiendo que uses Balena Etcher. Funciona muy bien y lo recomiendo encarecidamente. Si está utilizando Windows, puedes hacerte con una copia de Win32 Disk Imager. Aunque no es tan amigable como Balena Etcher, Win32 Disk Imager hace su trabajo.

Los usuarios de Linux deberéis recurrir a los siguientes pasos. No te preocupes, no está tan mal.

- 1. Inserte tu tarjeta SD en tu ordenador
- 2. Localice el dispositivo ejecutando sudo fdisk -l. Probablemente será el único disco con el tamaño correcto. Anota el nombre del dispositivo; supongamos que es /dev/sdx. Si tienes alguna duda, retira la tarjeta, ejecute sudo fdisk -l nuevamente y observa los discos que hay. Inserte la tarjeta SD nuevamente, ejecuta sudo fdisk -l y el que nos interesa es el nuevo disco que aparezca.
- 3. Desmonta las particiones ejecutando sudo umount /dev/sdx*. Puede aparecer un error indicando que el disco no está montado, es normal. Copia el contenido del archivo de imagen en la tarjeta SD ejecutando

```
sudo dd bs=1M if=your_image_file_name.img
of=/dev/sdx
```

Por supuesto, deberás cambiar el nombre del archivo de imagen anterior según corresponda.

Comprueba bien la unidad, el dispositivo, la letra de la unidad en la que vas a escribir. Asegúrate de no sobrescribir otra unidad importante. Mostraré imágenes del proceso tal como lo verías en una Mac. Es posible que te solicite que proporciones privilegios de administrador en Mac y Windows.

Selecciona el archivo de imagen con el que deseas actualizar el módulo eMMC después de insertarlo en una ranura USB de tu Mac; o, de cualquier modo, tienes la intención de acceder al módulo.

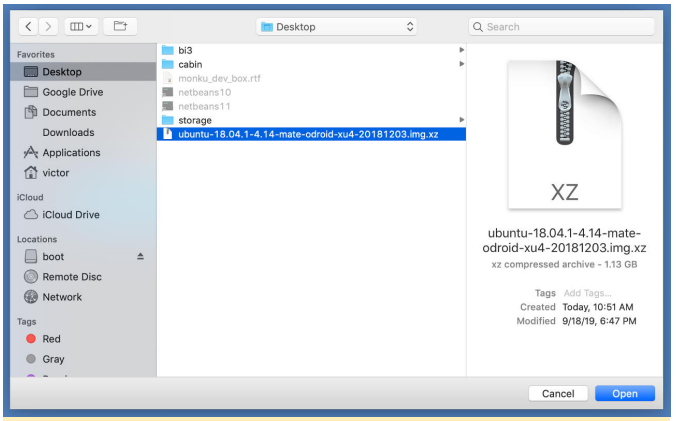


Figura 5

Responde a cualquier solicitud sobre privilegios de administrador.

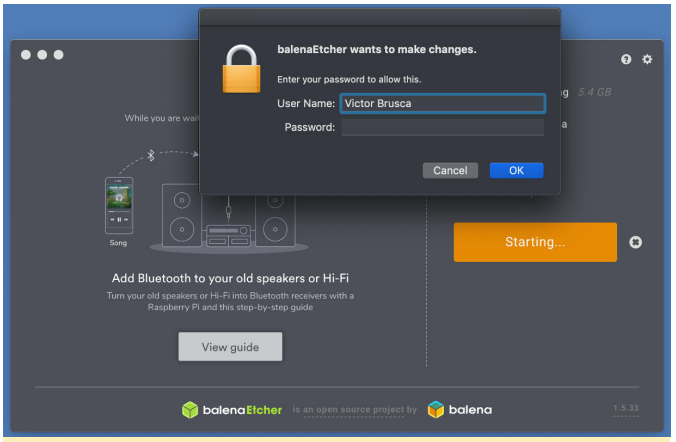


Figura 6

Vuelve a verificar que realmente estás escribiendo en el dispositivo correcto y que es del tamaño correcto.

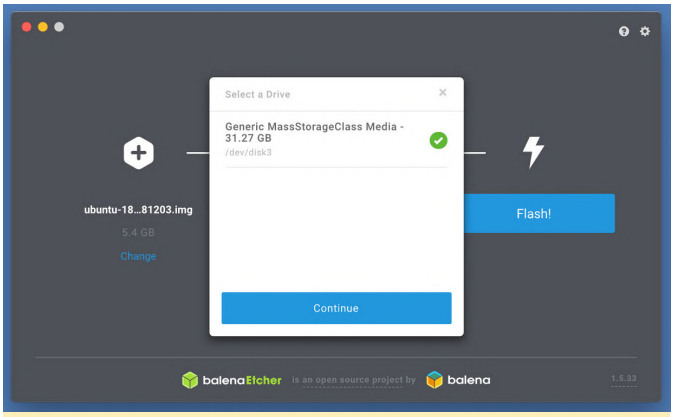


Figura 7

Empieza a escribir en el dispositivo y espere a que se complete el proceso.

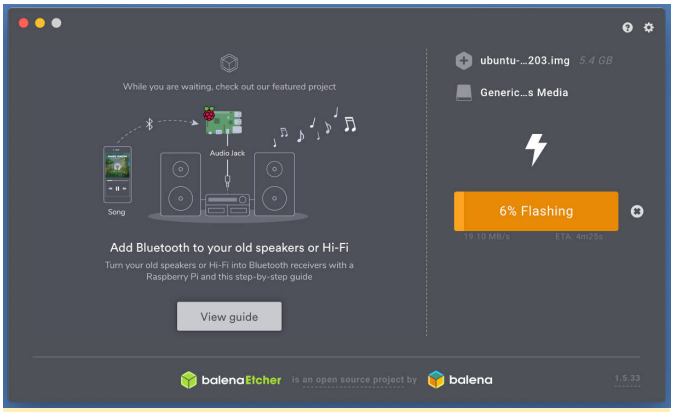


Figura 8

En el siguiente apartado, instalaremos el módulo eMMC y nos aseguraremos de que funcione correctamente en nuestro dispositivo.

Instalando el módulo eMMC

De modo que, si has llegado hasta aquí, ¡Ya debería tener un módulo eMMC con un SO y listo para arrancar! Echemos un vistazo a nuestro ODROID-XU4 y al módulo eMMC. Ten en cuenta que el módulo todavía está conectado al adaptador eMMC/microSD.

Figura 9

Echa un vistazo más de cerca al lugar del ODROID-XU4 donde va el módulo eMMC. Requiere que desmontes la caja o que haya recortado un trozo de ésta a modo de ventana para conectar el módulo eMMC. Si estás trabajando en una unidad de prueba, te recomiendo que cojas una cuchilla de afeitar afilada y abras un hueco para el eMMC a modo de ventana. De esta forma tendrás acceso directo al conector del módulo eMMC. Te facilitará mucho las cosas, especialmente si tienes pensado cambiar con frecuencia de módulo eMMC. Observa nuevamente el pequeño círculo blanco en la placa, está al lado de la flecha roja.

Figura 10

Es un poco pequeño, acerquémonos un poco más.

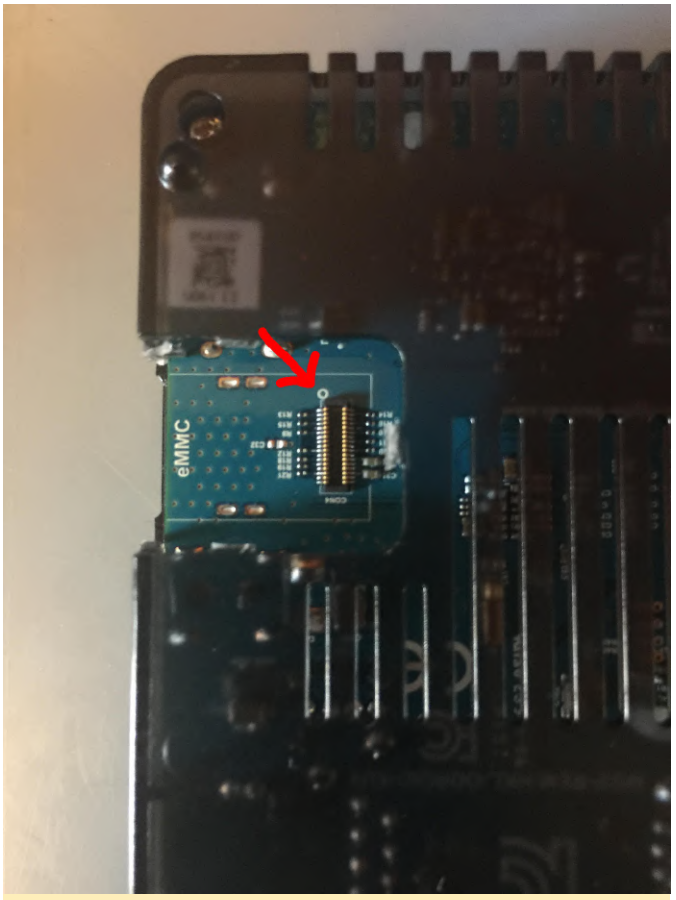


Figura 11

Alinea cuidadosamente el círculo blanco y el triángulo blanco para que estén en el mismo lado. Recurre a la misma técnica de ángulo para lograr que los conectores se unan y luego empujarlos con un leve movimiento de balanceo. Si estás conectando el módulo eMMC recurriendo al hueco realizado a modo de ventana, lo cual significa que estás haciendo uso de la carcasa, ten cuidado de no dejar que el módulo eMMC se deslice dentro de la carcasa. Una vez que tengas el chip conectado, debería tener algo similar a lo siguiente.

Figura 12

Casi hemos terminado. Tendrás que ver cómo interactúa tu dispositivo en particular con el módulo eMMC, en el ODROID-XU4 debes fijar el interruptor selector del modo de arranque en la posición adecuada. ¿Observas el pequeño interruptor blanco en la parte posterior de la caja? Coloca el interruptor en la posición más alejada del borde de la caja. En otras palabras, hacia el centro de la placa.

Figura 13

Ahora es el momento de probar las cosas. Procede a iniciar la placa, posiblemente venga acompañado de un reinicio o apagado rápido. No te preocupes, algunas versiones de Ubuntu tienden a cambiar el tamaño de la partición y luego se apagarán en su primer inicio. Mientras veas algo que aparezca en la pantalla, no tiene que preocuparte por nada.

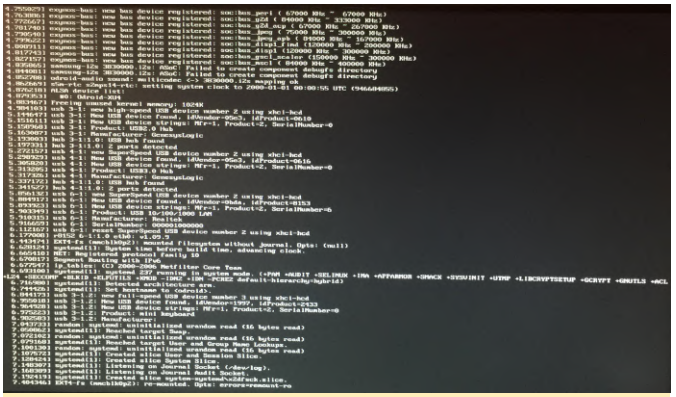


Figura 14

Figura 15

Bueno eso es todo. ¡Disfruta usando los módulos eMMC sin temor y espero que le saques partido a la mejora del rendimiento! Para comentarios, preguntas y sugerencias, visite el artículo original en http://dev.middlemind.com/tutorials/odroid_go/emmc.html.

Ogo Shell

November 1, 2019 By @paspartout ODROID-GO, Tutoriales

La utilidad ogo-shell es un explorador de archivos, reproductor de audio y visor de imágenes para ODROID-GO. Principalmente lo uso para escuchar música usando los auriculares odroig-go de backofficeshow. Puede consultar el Proyecto Github y la publicación del foro si quieres probarlo. La siguiente es la historia de su desarrollo hasta el momento. ¡Espero que te guste!

La idea

Desde que recibí mi ODROID-GO el año pasado, me he divertido bastante desarrollando pequeños programas para él. Me gusta el desafío de escribir programas para dispositivos integrados con recursos limitados. Te obliga a entender mejor la pila de hardware y de software en relación a programación de alto nivel a la que estoy acostumbrado trabajar.

Un día leí un post en el foro de Cralex sobre [Solicitudes de funciones/aplicaciones] [publicación en el foro] que mencionaba un administrador de archivos integrado en el dispositivo y algunos enlaces al Proyecto 3DShell. El archivo README del proyecto describe el propósito de 3DShell de la siguiente forma:

3DShell (pronunciado 3D-Shell): es un administrador de archivos multifuncional para Nintendo 3DS que tiene como objetivo la gestión de archivos multimedia.

Y así nació la idea de ogo-shell. Debía ser un explorador o administrador de archivos que pudiera ampliarse con más y más funcionalidades, como por ejemplo poder reproducir diferentes archivos multimedia.

Desarrollo hasta ahora

Puesto que probar pequeños cambios en los programas sobre el ODROID-GO lleva su tiempo, decidí implementar las funciones relacionadas con el hardware usando nuevamente la librería SDL2. (SDL2 es una conocida librería C utilizada principalmente en

juegos y proporciona acceso multiplataforma a gráficos, audio y hardware de entrada). Ésta hace posible ejecutar ogo-shell localmente en mi ordenador para que pueda testear y depurar rápidamente la lógica de las aplicaciones con facilidad.

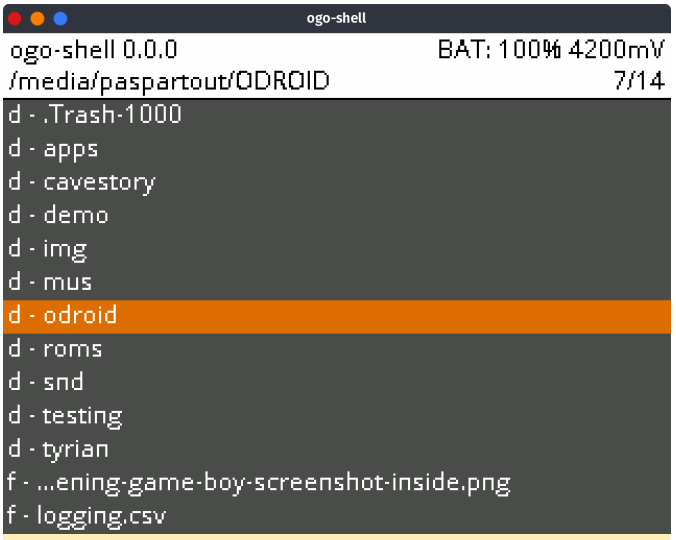


Figura 1 - Captura de pantalla de ogo-shell ejecutándose en Linux

Administrador de archivos

Después de implementar la simulación, empecé a trabajar en el administrador de archivos. Ya había creado un servidor ftp para GO, así que estaba familiarizado con las API del sistema de archivos. El administrador de archivos Rover para terminales me sirvió de inspiración, cuenta con un código C muy limpio cuya lectura es muy agradable. Llegado a este punto, quiero agradecer a todas las personas que publican su trabajo como código abierto. Sin sus esfuerzos no sería posible desarrollar proyectos como este.

Reproductor de música

Una vez completada la exploración básica de archivos, quería ver si sería posible reproducir varios archivos de audio. Logar que únicamente se reprodujeran fue relativamente fácil, pero montar un reproductor de audio en toda regla, me suponía enfrentarme a algunos desafíos.

Figura 2 - ODROID-GO y reproductor de música

Uno de ellos es la concurrencia. El reproductor no solo tiene que reproducir música, leer su música codificada desde la tarjeta SD, sino que también debe reaccionar a tus entradas mientras lo hace. Implementar esto en C tanto para la simulación como para ODROID-GO no fue tarea fácil, pero me hizo aprender más sobre los mutexes, colas y cómo funcionan.

Otro obstáculo lo tenía en el hardware ODROID-GO. El ESP32 en ODROID-GO usa la misma conexión SPI para comunicarse con la pantalla y la tarjeta SD. Esto significa que no puedes actualizar la pantalla y leer desde la tarjeta SD al mismo tiempo. Al parchear el SDK subyacente usando un mutex, me aseguré de que el programa no pueda acceder a la tarjeta SD hasta que la pantalla termine de actualizarse.

El resto fue aglutinar librerías de código abierto que se encargan de la decodificación de audio y de montar la interfaz de usuario correcta. El reproductor admite los formatos de audio habituales MP3, OGG, FLAC y WAV. Además de estos, también admite módulos en los formatos MOD, XM, IT y S3M. Existe un escenario bastante grande para los módulos de música y se pueden encontrar muchas pistas en The Mod Archive.

Visor de imágenes

La segunda versión después de la versión pública (0.3.0) incluye un visor de imágenes en bruto. Sin embargo, solo puede mostrar pequeñas imágenes

debido a la limitada cantidad de RAM que tiene ODROID-GO. Déjame explicarte por qué.

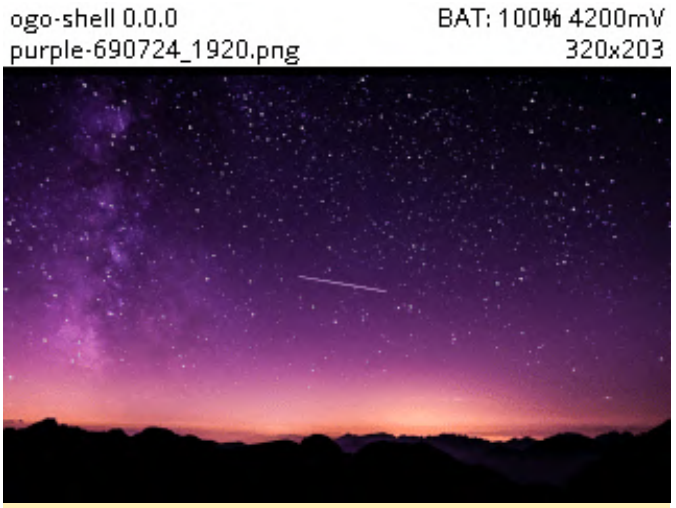


Figura 3: captura de pantalla del visor de imágenes

Si quieres mostrar una imagen grande, digamos una con las dimensiones 1000x1000 píxeles, lo normal es decodificar primero el archivo de imagen en la memoria y luego cambiar su tamaño desde allí a otra parte de la memoria. Puedes calcular la cantidad de espacio necesario multiplicando la cantidad de píxeles por la profundidad de color de la imagen. La mayoría de las imágenes utilizan una profundidad de color de 24 bits en el formato RGB. Eso significa que tenemos tres canales de color para los colores rojo, verde y azul y cada canal se guarda como un número entero de 8 bits. Para guardar nuestra imagen de 1000x1000, necesitamos 24 bits o 3 bytes multiplicados por un millón (1000 veces 1000) que ya son 3 millones de bytes. La cantidad de RAM a la que puedes acceder fácilmente en ODROID-GO es de 4 MB, de modo que cualquier cosa que sea mayor que 1000x1000 no cogerá en la memoria. Puesto que el modus operandi habitual no funciona, tendría que encontrar una forma más inteligente de decodificar partes de la imagen y cambiar su tamaño directamente sobre la marcha sin la necesidad de recurrir a un gran búfer de imagen en la RAM. Esta optimización puede requerir mucho tiempo y esfuerzo que aún no he querido dedicar a este proyecto.

Me he dado cuenta que conseguir un primer prototipo de una característica funcional suele ser bastante fácil y divertido. Pero pulir el software corrigiendo errores, teniendo en cuenta casos extremos y agregar características de calidad que damos por sentado, generalmente requiere mucho más tiempo y esfuerzo, y no resulta tan divertido. Esta es probablemente la razón por la que hay tantos errores en el software actual. Cuanto más complejo es el software, más tiempo y esfuerzo tendrás que invertir para que funcione correctamente.

Próximas funciones

Actualmente estoy trabajando en el soporte para chiptunes usando la librería game-music-emu que puede emular chips de sonido de viejas consolas de juegos y ordenadores. Me gusta mucho el sonido y la creatividad que tienen estas viejas canciones. Lo que también está en proceso de desarrollo es la capacidad de lanzar los emuladores go-play desde ogo-shell. Resumiendo, todavía hay muchas ideas y características que implementar y errores que corregir.

Dicho esto, ha sido bastante divertido desarrollar ogo-shell y aprender cosas nuevas mientras lo hacía. La satisfacción que obtienes cuando ves o escuchas por primera vez que tu creación funciona según lo previsto puede ser realmente adictivo. Espero haberte animado a que no solo uses programas, sino que también aprendas a crear otros nuevos, y espero que ogo-shell te sea útil.

Escritorios Multipantalla con VNC

November 1, 2019 By Adrian Popa Linux, ODROID-C2, ODROID-XU4

Hace unos dos años tuve una idea algo descabellada: ¿Sería posible crear un sistema de escritorio de "doble pantalla" utilizando dos ODROID, cada uno con una pantalla diferente, pero que actuasen como un escritorio unificado? La idea era tener un odroid "maestro" que fuera más potente en el que ejecutaríamos las aplicaciones (como un XU4 o N2) y un "esclavo", preferentemente más barato, que actuase únicamente como un terminal básico sin procesamiento (un C1/C2). Investigué un poco y encontré xdmx, que es un administrador de ventanas distribuido que podría funcionar, pero resultaba que había sido abandonado hace una década, de modo que no nos era factible.

Continúe investigando y creé un sistema algo enrevesado que básicamente hacia uso de Xpra (**https://xpra.org**) de un modo para el cual no estaba diseñado. Funciona, pero el rendimiento era pésimo: alrededor de 0.5 fps en todo el escritorio, lo cual era muy perceptible (tienes más detalles en: **https://forum.odroid.com/viewtopic.php?t=35710**). Después tuve la idea de experimentar con vnc, y esta vez los resultados fueron bastante mejores.

La idea

La idea es aparentemente muy simple. En el sistema maestro, inicias una sesión X11 normal, con lightdm incluido. Usas xrandr para extender el escritorio (una vez que hayas iniciado sesión) para que el nuevo tamaño del escritorio cubra ambas pantallas (yo utilicé pantallas idénticas, aunque debería funcionar con pantallas de diferentes resoluciones haciendo algunos ajustes). Luego, inicias una sesión X11vnc que se conecte a :0 y que tenga una resolución fija que es la suma de ambas pantallas.

El esclavo puede ejecutar una imagen mínima o de escritorio, pero necesita tener Xorg y debe soportar la resolución del monitor de destino. Éste Iniciará una sesión Xorg independiente que lee los comandos de inicio desde /root/.xinitrc. El archivo apunta a un

script que intentará iniciar vncviewer en una especie de bucle infinito, conectándose a la sesión del maestro. Si todo va bien, terminaras con dos pantallas replicadas, que no es lo mismo que una configuración de doble pantalla. Aquí es donde entra en juego la magia. Una vez que se inicia vncviewer, éste se mueve con la ayuda de xdotool hacía la izquierda por el ancho de la pantalla. Esto hace que lo que normalmente sería la parte izquierda del escritorio se represente fuera de la pantalla (y se superponga lógicamente con lo que hay en la pantalla izquierda), dejando espacio para el contenido de la pantalla derecha. ¿Confuso? Aquí tienes un diagrama:

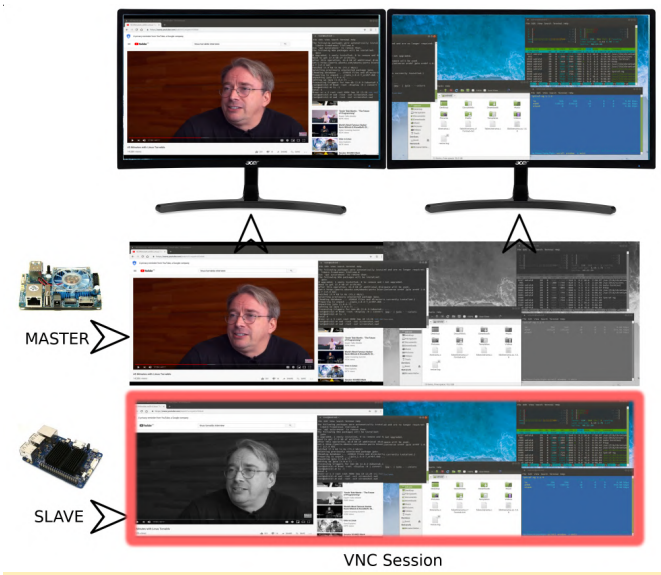


Figura 01: las partes atenuadas en gris no son visibles físicamente en las respectivas pantallas

Recuerda que el sistema maestro muestra ambas pantallas, pero solo la parte izquierda es físicamente visible, y el sistema esclavo también muestra ambas pantallas (a través de VNC), pero solo la parte derecha es físicamente visible. El resultado final es la ilusión de una configuración de doble pantalla.

Configuración principal

Mi banco de pruebas consiste en un ODROID-XU4 con Ubuntu 18.04 Mate como sistema maestro y un ODROID-N2 con Ubuntu 18.04 Mate como esclavo. Como he dicho anteriormente, el esclavo puede ser una placa menos potente (incluso de otra marcha), pero yo tenía un N2 a mano para experimentar. El maestro maneja el monitor izquierdo en la configuración mientras el esclavo está conectado al monitor derecho (este orden es importante). En mi caso, ambos monitores tienen una resolución de 1680x1050. Deberías poder llegar a 1080p por pantalla, pero no estoy seguro de si puedes tener dos pantallas 4K porque el escritorio parece estar limitado a 4096x4096 píxeles (aunque puede haber una forma de saltarse esta limitación: **https://bit.ly/2JsDWMc).**

El sonido debe estar conectado al maestro, pero el teclado y el ratón pueden conectarse a cualquiera de los sistemas una vez que se inicie VNC. Ten en cuenta que conectar el teclado y el ratón al esclavo puede provocar movimientos bruscos o ralentizaciones cuando haya mucha actividad en la pantalla, así que, para un mejor rendimiento, conéctalos también al maestro.

Con respecto a las redes, ambos sistemas deben estar en la misma LAN, conectados a través de Ethernet y con direcciones IP estáticas. El máximo consumo de la red que llegue a observar fue de aproximadamente unos 40 Mbps de tráfico VNC cuando reproducía video, de modo que la conexión Ethernet no debería provocar cuellos de botella.

Comencemos con la configuración del odroid maestro:

```
$ sudo su -
# apt-get install x11vnc pwgen git
```

Crearemos una contraseña aleatoria de 20 caracteres para VNC (aunque la documentación dice que solo se utilizan los primeros 8 caracteres), y la copiaremos a través de ssh en el sistema esclavo. Supongo que tu esclavo tiene una cuenta sin privilegios, como es odroid.

```
# pwgen 20 1
# x11vnc -storepasswd
# ssh odroid@slave mkdir /home/odroid/.vnc
# scp /root/.vnc/passwd
odroid@slave:/home/odroid/.vnc/passwd
# ssh odroid@slave chmod -R odroid:odroid
/home/odroid/.vnc/passwd
```

A continuación, creamos un servicio systemd para iniciar x11vnc en el arranque. Puedes utilizar y retocar la configuración de ejemplo que hay mi página git (https://github.com/mad-ady/vnc-multiscreen.git). Tendrás que cambiar la resolución combinada para

que coincida con la de tu sistema (en mi caso era 3360x1050):

```
# git clone https://github.com/mad-ady/vnc-multiscreen.git
# cp vnc-multiscreen/master-left/etc/systemd/system/x11vnc.service /etc/systemd/system/
# systemctl daemon-reload
# systemctl enable x11vnc
# systemctl start x11vnc
```

Figura 02 - El servicio X11vnc, personalizado con la resolución de pantalla compuesta

A continuación, debemos añadir un script que se ejecute una vez que inicies sesión (nuevamente, supongo que estás utilizando el usuario odroid para el inicio de sesión de la GUI) y que use xrandr para cambiar el tamaño de tu escritorio. Podríamos haber ejecutado el script desde lightdm (antes de iniciar sesión), pero por alguna razón el fondo del escritorio abarca una única pantalla. Ejecuta los siguientes comandos con el usuario con el que inicias sesión (en el maestro):

```
$ mkdir .config/autostart
$ cp vnc-multiscreen/master-left/home/odroid/.config/autostart/dual-screen.desktop .config/autostart
$ chmod a+x .config/autostart/dual-screen.desktop
$ gio set .config/autostart/dual-screen.desktop "metadata::trusted" yes
$ sudo cp vnc-multiscreen/master-left/usr/local/bin/set-dual-screen-resolution.sh /usr/local/bin/
```

Figura 03 – El script set-dual-screen-resolution (en el maestro) que extiende el escritorio

Deberás editar el script /usr/local/bin/set-dual-screen-resolution.sh y fijar tu resolución total en el parámetro fb (que será el tamaño del escritorio) y ajustar la resolución de tu pantalla izquierda en el parámetro panning. Si lo configuras así, el escritorio no empezará a desplazarse cuando muevas el ratón hacia borde de la pantalla.

Eso es todo sobre la configuración básica en el sistema maestro, y ahora le toca el turno al esclavo. Desactivaremos el modo GUI y crearemos y habilitaremos un servicio xorg que inicie un simple servidor X11 y un script para conectarse a vnc:

```
# apt-get install xtightvncviewer xdotool git
# service lightdm stop
# systemctl set-default multi-user.target
# git clone https://github.com/mad-ady/vnc-multiscreen.git
# cp vnc-multiscreen/slave-right/etc/systemd/system/xorg.service /etc/systemd/system
# systemctl enable xorg
```

El servidor X analiza .xinitrc y lo ejecuta después del inicio. Así que haremos que inicie nuestro script de inicio VNC.

```
# cp vnc-multiscreen/slave-right/root/.xinitrc /root/.xinitrc
# chmod a+x /root/.xinitrc
```

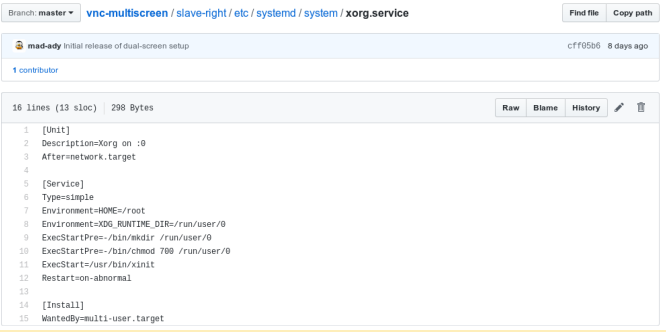
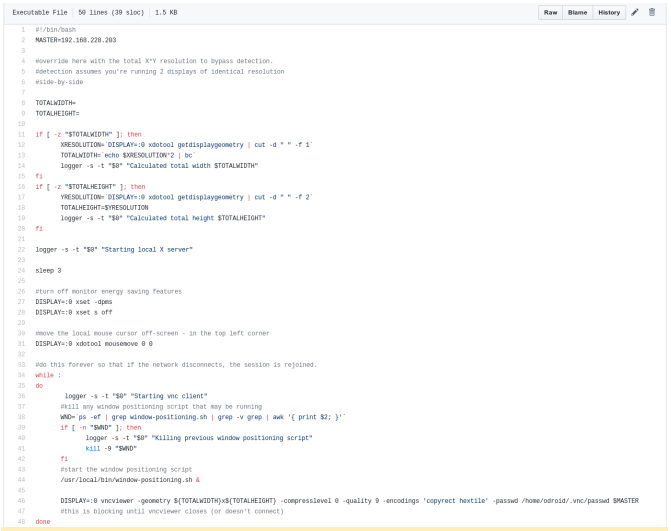


Figura 04 - El servicio de inicio Xorg y el script

El script de inicio VNC iniciará el proceso vncviewer (en un bucle) y luego moverá la ventana a la izquierda. Todo se realiza con ayuda de estos dos scripts:

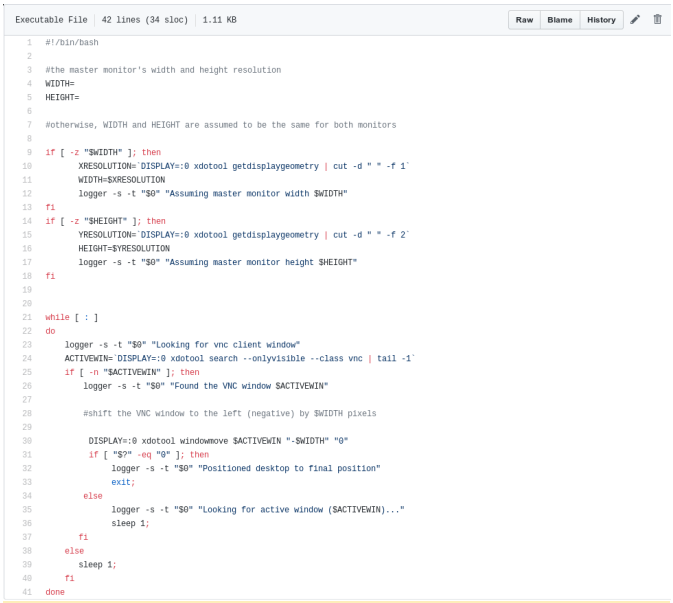
```
# cp vnc-multiscreen/slave-right/usr/local/bin/dual-screen-vnc-client.sh /usr/local/bin
# chmod a+x /usr/local/bin/dual-screen-vnc-
```

```
client.sh
# cp vnc-multiscreen/slave-
right/usr/local/bin/window-positioning.sh
/usr/local/bin
# chmod a+x /usr/local/bin/window-positioning.sh
```



```
Executable File | 50 lines (39 sloc) | 1.5 KB
Raw | Blame | History

#!/bin/bash
MASTER=192.168.228.203

#override here with the total X*Y resolution to bypass detection.
#detection assumes you're running 2 displays of identical resolution
#side-by-side

TOTALWIDTH=
TOTALHEIGHT=

if [ -z "$TOTALWIDTH" ]; then
        XRESOLUTION=`DISPLAY=:0 xdotool getdisplaygeometry | cut -d " " -f 1`
        TOTALWIDTH=`echo $XRESOLUTION*2 | bc`
        logger -s -t "$0" "Calculated total width $TOTALWIDTH"
fi
if [ -z "$TOTALHEIGHT" ]; then
        YRESOLUTION=`DISPLAY=:0 xdotool getdisplaygeometry | cut -d " " -f 2`
        TOTALHEIGHT=$YRESOLUTION
        logger -s -t "$0" "Calculated total height $TOTALHEIGHT"
fi

logger -s -t "$0" "Starting local X server"

sleep 3

#turn off monitor energy saving features
DISPLAY=:0 xset -dpms
DISPLAY=:0 xset s off

#move the local mouse cursor off-screen - in the top left corner
DISPLAY=:0 xdotool mousemove 0 0

#do this forever so that if the network disconnects, the session is rejoined.
while :
do
        logger -s -t "$0" "Starting vnc client"
        #kill any window positioning script that may be running
        WND=`ps -ef | grep window-positioning.sh | grep -v grep | awk '{ print $2; }'`
        if [ -n "$WND" ]; then
                logger -s -t "$0" "Killing previous window positioning script"
                kill -9 "$WND"
        fi
        #start the window positioning script
        /usr/local/bin/window-positioning.sh &

        DISPLAY=:0 vncviewer -geometry ${TOTALWIDTH}x${TOTALHEIGHT} -compresslevel 0 -quality 9 -encodings 'copyrect hextile' -passwd /home/odroid/.vnc/passwd $MASTER
        #this is blocking until vncviewer closes (or doesn't connect)
done
```

Figura 05 - El script dual-screen-vnc-client



```
Executable File | 42 lines (34 sloc) | 1.11 KB
Raw | Blame | History

#!/bin/bash

#the master monitor's width and height resolution
WIDTH=
HEIGHT=

#otherwise, WIDTH and HEIGHT are assumed to be the same for both monitors

if [ -z "$WIDTH" ]; then
        XRESOLUTION=`DISPLAY=:0 xdotool getdisplaygeometry | cut -d " " -f 1`
        WIDTH=$XRESOLUTION
        logger -s -t "$0" "Assuming master monitor width $WIDTH"
fi
if [ -z "$HEIGHT" ]; then
        YRESOLUTION=`DISPLAY=:0 xdotool getdisplaygeometry | cut -d " " -f 2`
        HEIGHT=$YRESOLUTION
        logger -s -t "$0" "Assuming master monitor height $HEIGHT"
fi


while [ : ]
do
    logger -s -t "$0" "Looking for vnc client window"
    ACTIVEWIN=`DISPLAY=:0 xdotool search --onlyvisible --class vnc | tail -1`
    if [ -n "$ACTIVEWIN" ]; then
        logger -s -t "$0" "Found the VNC window $ACTIVEWIN"

        #shift the VNC window to the left (negative) by $WIDTH pixels

         DISPLAY=:0 xdotool windowmove $ACTIVEWIN "-$WIDTH" "0"
         if [ "$?" -eq "0" ]; then
              logger -s -t "$0" "Positioned desktop to final position"
              exit;
        else
              logger -s -t "$0" "Looking for active window ($ACTIVEWIN)..."
              sleep 1;
        fi
    else
       sleep 1;
    fi
done
```

Figura 06 - El script window-positioning

Si no estás utilizando pantallas idénticas, deberás editar los dos scripts anteriores y fijar los desfases correctos de tus pantallas.

Ya tenemos hechos los pasos básicos. Puedes reiniciar ambos sistemas y después de iniciar sesión en lightdm deberías tener un escritorio extendido. Sin embargo, hay algunas cosas que no funcionarán como cabría esperar, necesitamos hacer algunos ajustes.

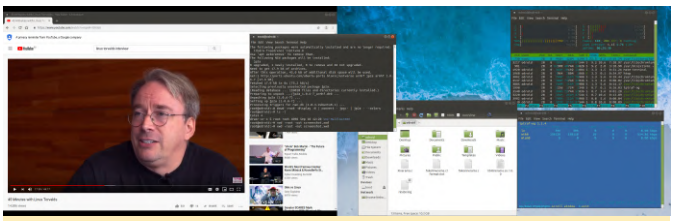


Figura 07 - Cómo debería verse tu escritorio

Retoques (solo en el sistema maestro)

Debes deshabilitar el mosaico del sistema de ventanas MATE (y la composición mientras estés en él), porque cuando intentes maximizar una ventana en la pantalla derecha arrastrando su barra de título hacia la parte superior, la ventana saltará a la pantalla izquierda. Puede hacerlo desde Menu -> Control Center -> Windows -> Placement -> Disable window tiling.

A continuación, debe deshabilitar el contenido de la ventana mientras la mueves, para conseguir así una experiencia más fluida: Menu -> Control Center -> Mate Tweaks -> Windows -> Do not show window content while moving windows.

Como tienes dos pantallas, sería bueno que las ventanas sepan dónde está la división entre las pantallas y permita maximizar las ventanas en cada pantalla. Para esto, necesitas adaptar lo que la librería xinerama le dice a tu servidor X sobre las pantallas disponibles. Afortunadamente, existen una librería "fakexinerama" que puedes instalar en el sistema maestro (https://www.xpra.org/trac/wiki/FakeXinerama).

```
$ wget
https://www.xpra.org/trac/browser/xpra/trunk/fakex
inerama/fakeXinerama.c?format=txt -O
fakexinerama.c
$ sudo apt-get install libxinerama-dev libx11-dev
$ gcc -O2 -Wall Xinerama.c -fPIC -o
libXinerama.so.1.0.0 -shared
$ sudo mv /usr/lib/arm-linux-
gnueabihf/libXinerama.so.1.0.0 /usr/lib/arm-linux-
gnueabihf/libXinerama.so.1.0.0--original
$ sudo cp libXinerama.so.1.0.0 /usr/lib/arm-linux-
gnueabihf/
```

Si tienes una arquitectura diferente en el maestro (arm64, x86_64), modifica la ruta del archivo de la librería. Puedes encontrar esto en:

```
$ find /usr/lib -name libXinerama.so
```

Fake Xinerama lee la configuración de la pantalla desde el directorio de inicio del usuario de la GUI ~/.fakexinerama. El archivo empieza con el recuento de monitores (que es 2 en nuestro caso) y enumera cada monitor por línea con ajustes y resolución:

```
$ cat ~odroid/.fakexinerama
2
#left screen, starts at x=0 and y=0 and has a size
of 1680x1050 pixels
0 0 1680 1050
#right screen, starts at x=1680 and y=0 and has a
size of 1680x1050
1680 0 1680 1050
```

Una vez que reinicies tu sesión de escritorio, las ventanas deberían comportarse como si tuvieras dos pantallas físicas pudiendo maximizarlas en cada pantalla.

Nos falta una integración más: el protector de pantalla solo se activa en la pantalla del sistema maestro. Probablemente solo tenga en cuenta el tamaño de la pantalla, no el tamaño total del escritorio, de modo que queda visible la pantalla derecha. Puede usar gdbus para ver cuándo se activa el protector de pantalla y puedes conectarte por ssh al esclavo y apagar la pantalla con xset dpms force off. Cuando observes que el protector de pantalla se desactiva, puedes volver a habilitar la pantalla derecha.

Para hacer esto, primero debes ser capaz de pasar del maestro al esclavo sin una contraseña. Para ello crearemos una clave y copiaremos la parte pública al esclavo. Omite el primer paso si ya tiene creadas las claves.

```
$ ssh-keygen -t rsa -C "master key"
$ ssh-copy-id odroid@slave
```

A continuación, copia el script que vigila el estado del protector de pantalla e inícialo como parte de la sesión de escritorio del usuario. Afortunadamente, systemd también puede manejar servicios de usuario (nuevamente, en el maestro):

```
$ mkdir -p .config/systemd/user/
$ cp vnc-multiscreen/master-
left/home/odroid/.config/systemd/user/screensaver-
sync.service .config/systemd/user
$ systemctl --user enable screensaver-sync
$ sudo cp vnc-multiscreen/master-
left/usr/local/bin/screensaver-sync.sh
/usr/local/bin/
```

Asegúrese de editar (y probar) el script de sincronización del protector de pantalla, de modo que apunte a la dirección IP correcta del sistema esclavo.

Pros y Contras:

- El maestro (izquierda) hace todo el trabajo pesado, además tiene el rendimiento más fluido.
- El esclavo (derecha) se usa para renderizar media pantalla, pero necesita transferir y renderizar la pantalla combinada (incluso si solo ve la mitad), de modo que una actividad intensa en la pantalla izquierda provocará saltos/retrasos en la pantalla derecha
- Debes encender y apagar ambos dispositivos de forma independiente. Es posible apagar automáticamente el esclavo y luego apagar el maestro, pero es necesario realizar algunos ajustes (por ejemplo, un servicio systemd que se conecte al esclavo para apagarlo).
- Puedes convertir fácilmente al esclavo en un sistema autónomo deshabilitando el servicio xorg y volviendo a habilitar el sistema gráfico. El maestro puede permanecer como está sin afectar a su funcionalidad.
- El rendimiento es nativo en la pantalla maestra/izquierda y varía de un par de FPS a ~ 15 FPS en la pantalla esclava/derecha, dependiendo de la actividad de la pantalla. La pantalla derecha es la más adecuada para trabajar con contenido estático, como una página web, un código o un terminal.

Puede ver una demostración aquí (mil disculpas por la mala calidad del video): **https://www.youtube.com/watch?v=sSqXX5doCvo&feature=youtu.be**

Ideas para mejorar

Perfectamente, todo esto se simplificaría enormemente si solo X11VNC (o quizás con una tecnología de escritorio remoto diferente) pudiera copiar únicamente la mitad de una pantalla de datos en lugar del escritorio completo. De esta manera, se reduciría la carga de trabajo tanto en maestro como en esclavo evitando tener que recurrir a trucos para

el movimiento de las ventanas. Puedes mostrar la segunda pantalla en cualquier sistema (por ejemplo, dentro de un navegador en un televisor) sin muchos problemas.

Puede lograr mejor rendimiento si el maestro es un PC (por ejemplo, un ODROID H2) y el esclavo es un ODROID ARM/ARM64. Esto se debe a que la velocidad de lectura de framebuffer en un XU4 es de aproximadamente 45 MB/s (N2 es de unos 443 MB/s), mientras que la GPU Intel alcanza aproximadamente unos 961 MB/s. Más velocidad conlleva una velocidad de actualización más rápida, pero requerirá más ancho de banda de red.

Una cosa más: la técnica anterior se puede ampliar a más de 2 monitores porque puede conectar dos clientes VNC al mismo servidor. Sin embargo, el rendimiento disminuirá considerablemente. Avísame si encuentras alguna forma de mejorar todo esto en el hilo de soporte en **https://forum.odroid.com/viewtopic.php?f=52&t=36411**.

Monku R3: Desarrollando la Mejor Consola de Juegos ODROID-XU4/XU4Q - Parte 2

November 1, 2019 By Brian Ree Juegos, ODROID-XU4

Hola y bienvenidos al último tutorial de esta serie. Espero que te hay sido útil esta serie. La parte 1 de este tutorial en la que se detalla la configuración inicial del hardware y del software la puedes encontrar en **http://middlemind.com/tutorials/odroid_go/mr3_build.html**. Este artículo te mostrará en detalle cómo pulir tu consola de videojuegos Monku Retro 3 (ODROID-XU4). Optimizaremos Ubuntu MATE, el entorno de escritorio Linux, configurando retroarch en modo quiosco y scripts de configuración boot.ini. ¡Vamos a ello!

Este tutorial no requiere nuevas herramientas y componentes. Configuraremos la consola que ya has creado, ajustando las cosas para que realmente deslumbre.

Terminando con MATE ... Casi

Lo primero que vamos a hacer es deshacernos de esa molesta solicitud de autenticación que aparece cuando intentas abrir un navegador por primera vez tras iniciar sesión.

No estamos ante una configuración de alta seguridad: deliberadamente vamos a reducir el nivel de seguridad para facilitar su uso como consola de juegos y navegador web. Dirígete a la siguiente ubicación del menú: Applications -> Accessories -> Passwords and Keys, debería ver una ventana similar a la que se muestra a continuación.

Si en algún momento se te solicita iniciar sesión cuando trabajes con el terminal, utiliza la contraseña, k

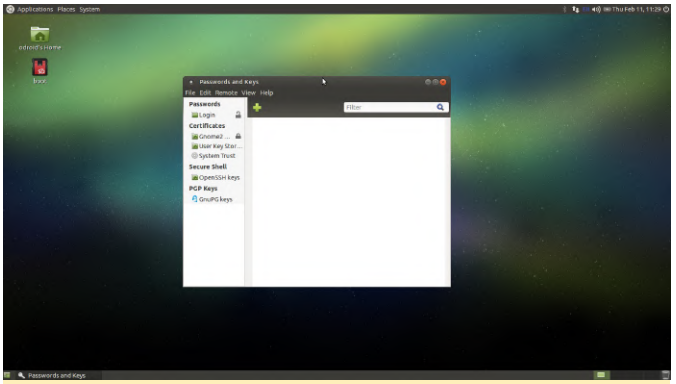


Figura - 01

Localiza la entrada de inicio de sesión de la lista en el lado izquierdo de la ventana. Si la ventana de autentificación que he mencionado hace referencia a un Password keychain diferente, busca esa entrada en la lista del lado izquierdo de la ventana. Sigue los siguientes pasos para desbloquear el keychain.

1. Haz clic derecho en la entrada de corresponda y selecciona Change Password.
2. Te pedirá que introduzcas la contraseña anterior: escribe odroid y haz clic en Continue.
3. Ahora te pedirá que introduzcas una nueva contraseña: deja ambos campos en blanco y haz clic en Continue.
4. Aparecerá otro cuadro de diálogo y te preguntará si estás de acuerdo en permitir que se desbloquee el keychain: haz clic en Continue
5. Cierre todos los cuadros de diálogo y la ventana Password and Keys; estamos listos.

Lo siguiente que haremos con el entorno MATE es configurar un poco los paneles y widgets. Tú puedes hacer lo que quieras aquí. Yo simplemente te mostraré cómo configurar las cosas y por qué. En primer lugar, esperamos interactuar con este sistema, al menos parte del tiempo, con un gamepad. Funciona muy bien, pero no es un ratón. Realmente no vamos a hacer ninguna tarea de programación rigurosa en Linux, por lo que hay algunas cosas que no necesitamos. La segunda ventaja de esta configuración es que reduciremos un poco la sobrecarga de memoria.

Sigue estas instrucciones para suprimir el panel inferior. Agregaremos controles al panel superior para compensar parte de la pérdida de funcionalidad, pero no agregaremos el widget de selección de escritorio. Puede resultar un poco exagerado para nuestras necesidades. No obstante, si lo quieres conservar, no perjudica en nada. Sigue estas instrucciones para limpiar un poco los paneles y widgets.

1. Dirígete al panel inferior y haz clic derecho, selecciona Delete This Panel, luego haz clic en Delete nuevamente cuando se te solicite.
2. Dirígete a la parte superior derecha de la pantalla y haz clic con el botón derecho en el botón de encendido, selecciona Remove From Panel

Lo que vamos a hacer es volver a añadir algunos widgets, haciendo que el panel superior sea un punto de control más centralizado. Esto hará que utilizar un gamepad para controlar las cosas resulte mucho más fácil. Tu escritorio debería ser similar a la siguiente captura de pantalla.

Figura - 02

Como eliminamos los botones de selección de ventana abierta cuando suprimimos el panel inferior, vamos añadir un nuevo widget al panel superior que sea más apropiado para un gamepad y que requiera menos movimiento de cursor para utilizarlo. Haz clic derecho en el panel superior y selecciona Add to Panel. Desplázate hacia abajo en la lista de opciones hasta que veas la entrada que se muestra a continuación, luego haz clic en Add.

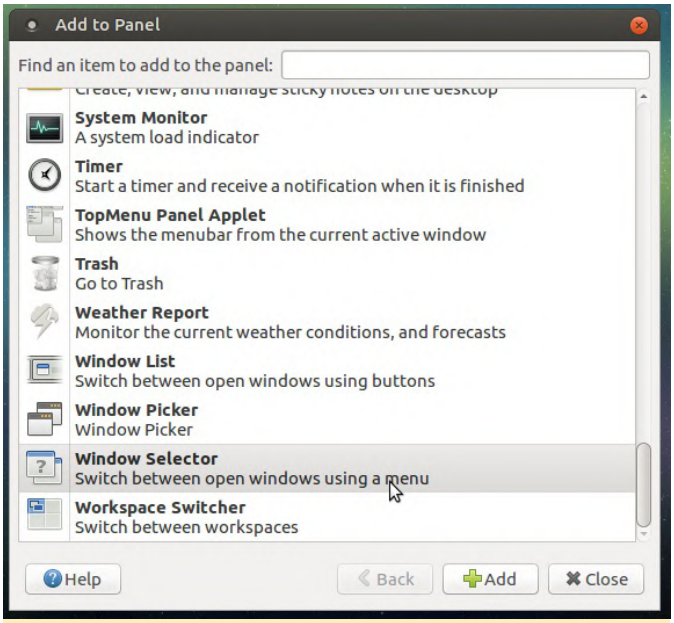


Figura - 03

Tenemos que realizar algunos pasos más con respecto a las aplicaciones de la bandeja y la configuración de la fecha y hora, pero casi hemos terminado con este punto. Tu escritorio debería verse más o menos así.

Figura - 04

A continuación, haz clic en la fecha y hora en la esquina superior derecha. Debería aparecer un menú desplegable con el calendario. Expande la sección “Locations” y haz clic en el botón Add. Configura la pestaña General como se muestra a continuación o como mejor te parezca.

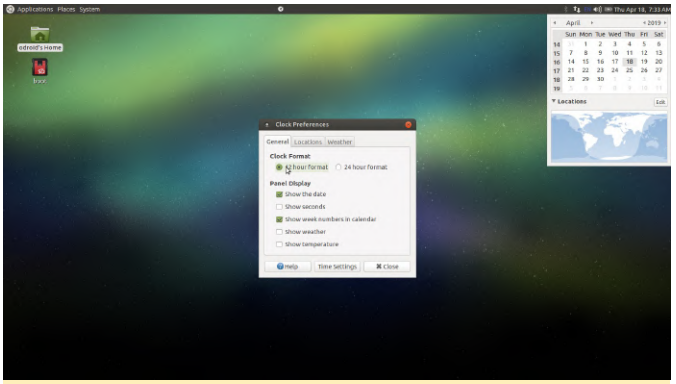


Figura - 05

Agregamos información de la ubicación para que la hora sea la correcta cuando tengamos una conexión a Internet y la sincronicemos con el protocolo de hora en red (NTP). Haz clic en la pestaña Locations y luego haz clic en el botón Add.

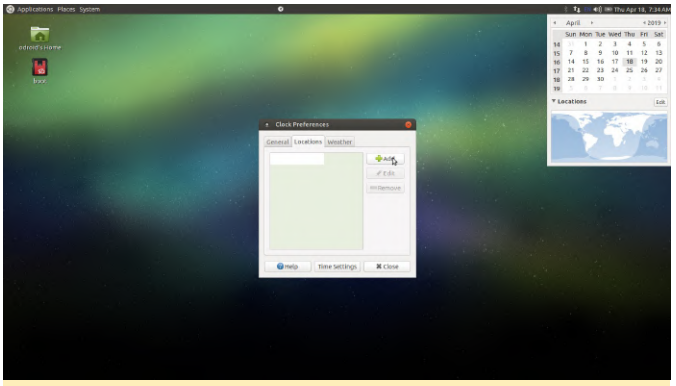


Figura - 06

Empieza escribiendo la ciudad principal que tienes más cerca en el cuadro de texto Name. Si esto no funciona, prueba con otra ciudad importante o representativa de tu zona horaria (por ejemplo, en mi caso es Nueva York). Selecciona una ubicación de la lista que aparece. La mía sería Central Park, Nueva York. Haz clic en Aceptar una vez que haya encontrado la ubicación adecuada.

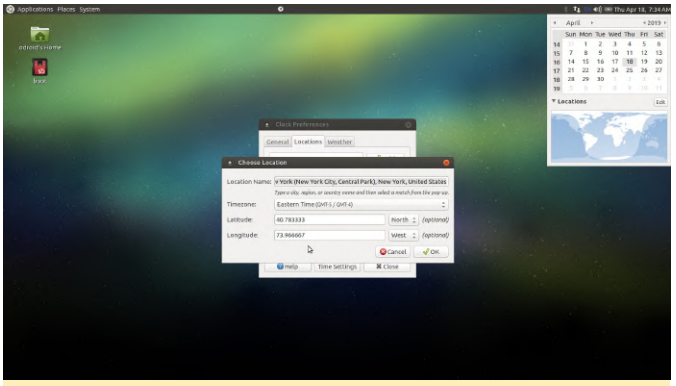


Figura - 07

Ahora verás una entrada en la lista de ubicaciones tal como se muestra a continuación.

Figura - 08

Sólo quedan que retocar unas cuantas cosillas más. Haz clic derecho en el icono de la batería en la bandeja superior derecha. Selecciona la opción Preferences. Haz clic en la pestaña General y desmarca Never Display An Icon. Si tiene una cadena EN o UK en la bandeja del sistema, haz clic derecho sobre ella y selecciona Preferencies. En la pestaña General, desmarca Show Icon On System Tray. Realmente no nos preocupa el tener que cambiar el idioma del teclado. Si lo necesitas, puede volver a activar el icono de la bandeja del sistema utilizando la opción de menú System -> Control Center y haciendo clic en Power Management and iBus Preferences respectivamente. Las siguientes capturas de pantalla muestran las cuestiones que acabamos de tratar.

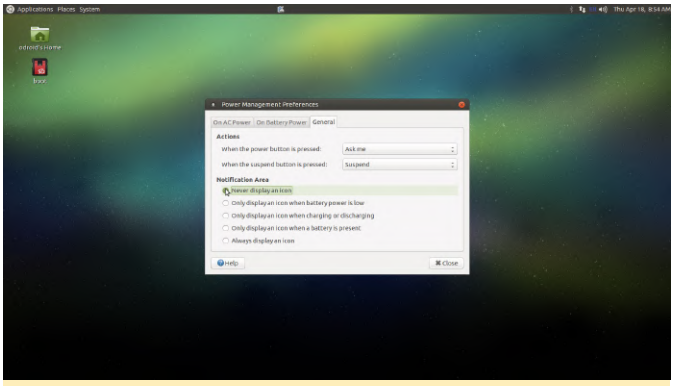


Figura - 09

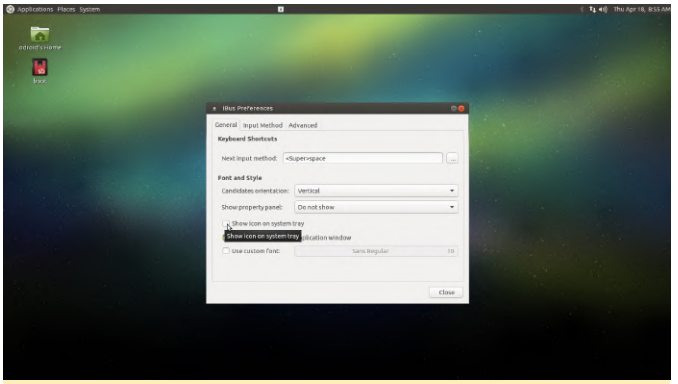


Figura - 10

Nos quedan dos cositas en esta sección, luego pasaremos al botón de control personalizado y a los scripts. Mueve el ratón al panel superior y haz clic derecho. Selecciona Add to Panel, luego desplázate hacia abajo hasta que veas la opción Show Desktop como se muestra a continuación. Haz lo mismo para la opción Trash

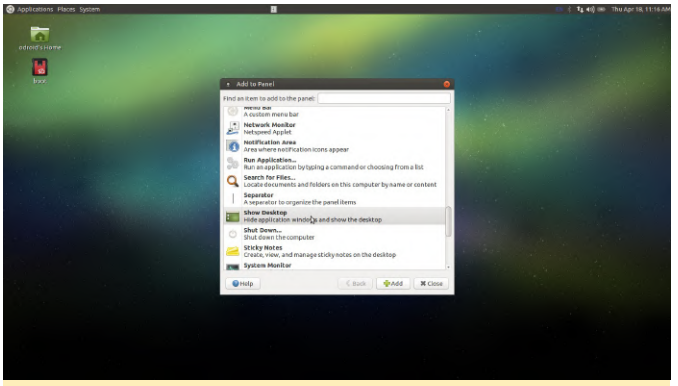


Figura - 11

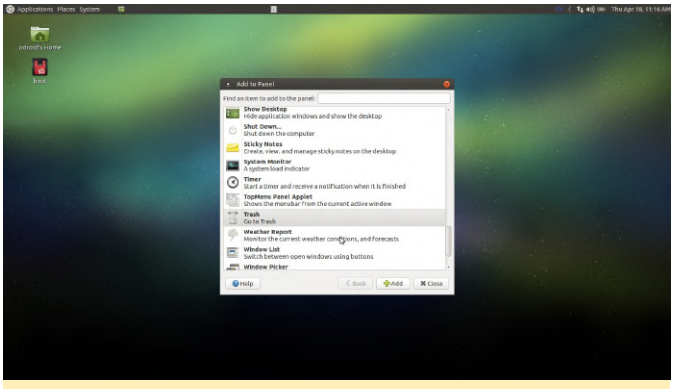


Figura - 12

Usa el botón central del ratón, o la rueda del ratón, para mover los íconos del widget del panel superior. Arrastramos estos dos nuevos widgets un poco más cerca del menú Systems, aunque no demasiado. Ahora estamos listos para empezar a añadir scripts personalizados. Estos scripts iniciarán automáticamente RetroArch durante el arranque e iniciarán AntiMicro cuando RetroArch se cierre,

devolviendo el control del ratón al gamepad. También configuraremos algunos scripts de control boot.ini. tu escritorio debería tener un aspecto similar al que se muestra a continuación.

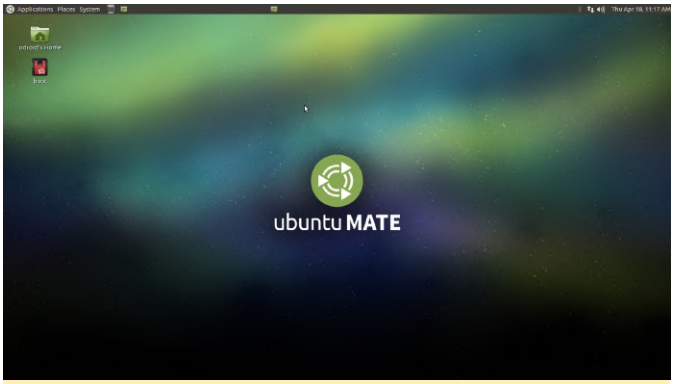


Figura - 13

Scripts y Control Personalizado

En esta sección, vamos a configurar algunos scripts personalizados para controlar el software que hemos instalado y configurado. Esto hará que pasemos de una experiencia de escritorio de Linux a una experiencia más de consola de juegos. Descárgate el paquete del script a tu dispositivo y luego cópialo y pégalo en la carpeta install_zips que creaste en el anterior tutorial. Se encuentra en el directorio home del usuario odroid.

Scripts Monku R3

Una vez que hayas descargado y copiado el archivo zip en la carpeta install_zips, haz clic derecho y selecciona Extract Here. Siete archivos deberían aparecer en un subdirectorio. Ábrelo y cópia todos los archivos en el directorio de inicio del usuario odroid. El directorio de inicio del usuario ODROID es la ubicación por defecto del administrador de archivos. También puede acceder a él desde un acceso directo que hay en el escritorio, o desde el panel izquierdo del administrador de archivos donde están listados los accesos directos a las carpetas.

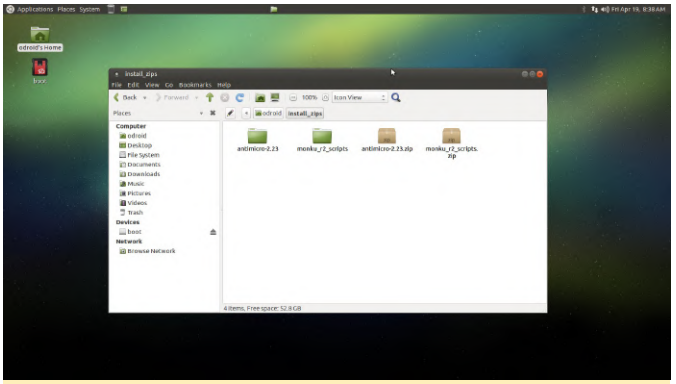


Figura - 14

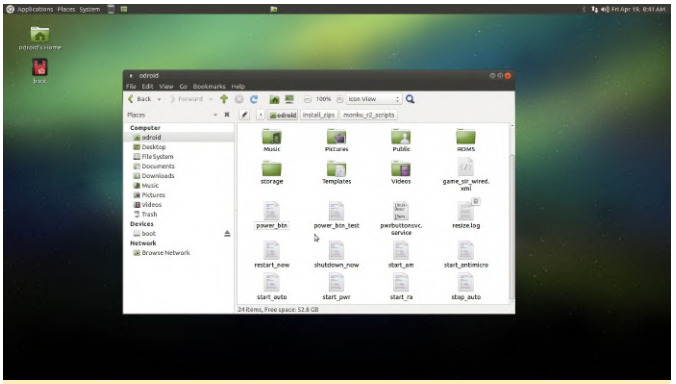


Figura - 15

Nos aseguraremos de que estos scripts tengan los permisos correctos y se puedan ejecutar. Abre un terminal, Applications -> System Tools -> MATE Terminal, y ejecuta los siguientes comandos.

```
$ sudo chmod 755 restart_now shutdown_now start_am
start_antimicro start_auto start_ra stop_auto

$ sudo chmod +x restart_now shutdown_now start_am
start_antimicro start_auto start_ra stop_auto
```

Ahora cierra el terminal y dirígete a System -> Control Center en los menús. Busca y selecciona la opción Startup Applications. Debería ver algo similar a lo que se muestra a continuación.

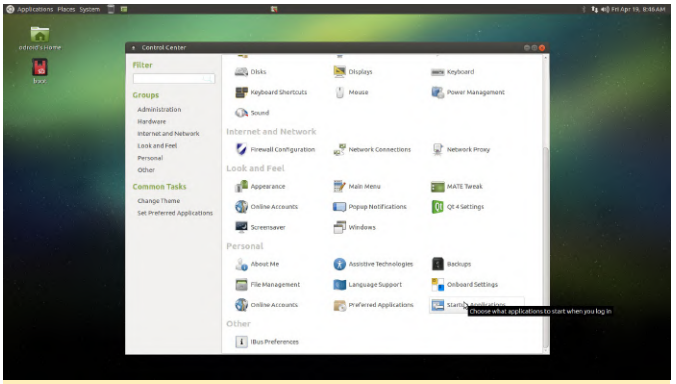


Figura - 16

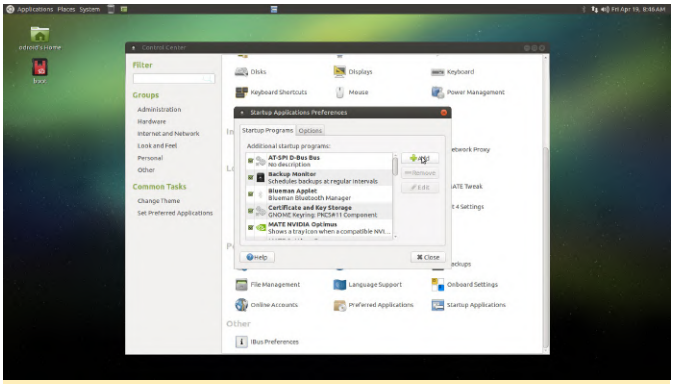


Figura - 17

Haz clic en el botón Add y completa el formulario que se muestra a continuación. Aquí tienes el texto exacto que tiene que excribir.

```
Name: Start RetroArch
Path: /home/odroid/start_ra
Description: Launches RetroArch on startup.
```

Ahora, queremos iniciar el script especial AntiMicro. Así que, hacemos lo mismo para este script, que también se muestra a continuación. Aquí tienes los valores utilizados.

```
Name: Start AntiMicro
Path: /home/odroid/start_am
Description: Launches AntiMicro on startup.
```

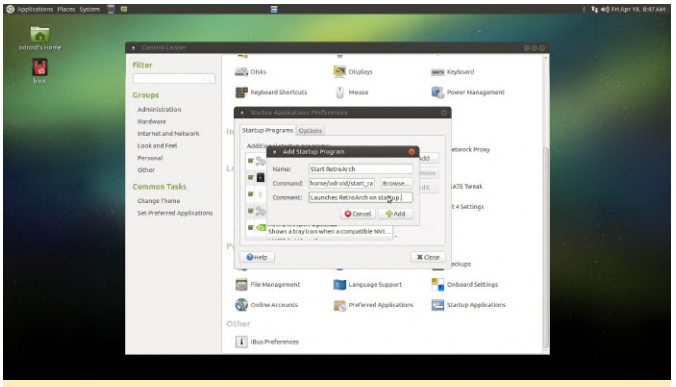


Figura - 18

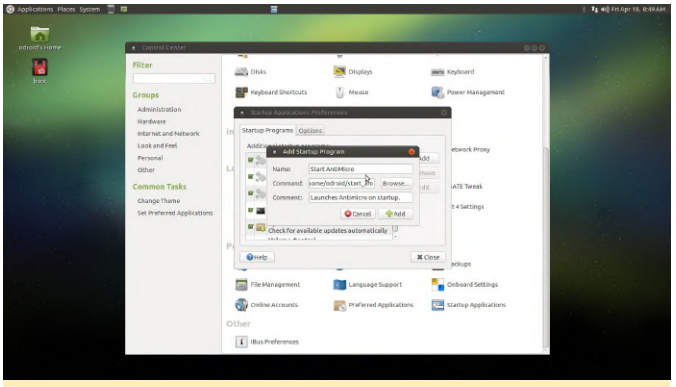


Figura - 19

Apaga el dispositivo: System -> Shut Down. Usa el botón de reinicio de hardware para volver a encenderlo. Deberías observar que RetroArch se inicia automáticamente. Ahora, si cierras RetroArch y esperas unos 5 segundos, deberías ver la ventana de AntiMicro en la bandeja del sistema y observarás que vuelves a controlar el dispositivo con el gamepad.

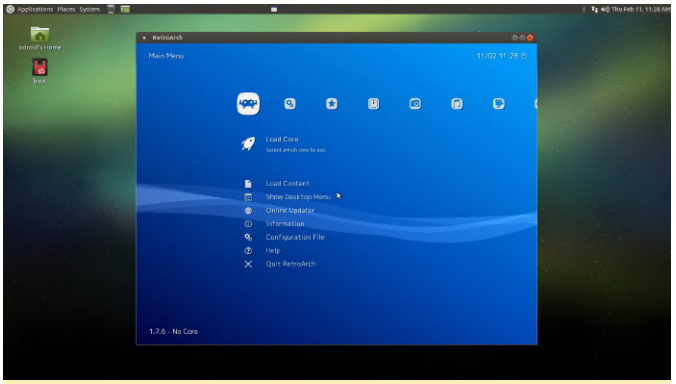


Figura - 20

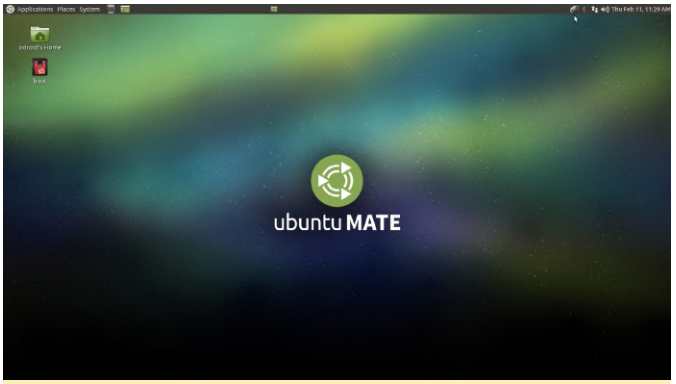


Figura - 21

Las cosas se están desarrollando bien. Nuestro magnífico ODROID se parece cada vez más a una gran consola de juegos retro.

Finalizando MATE ... esta vez de verdad

Tenemos algo más que hacer con MATE. Haga clic derecho en el panel superior y selecciona Add to Panel, desplácese hacia abajo a través de la lista de opciones hasta que encuentre la entrada Shut Down. Haga clic en Add y luego usa el botón central del ratón o el botón de la rueda para agarrar el nuevo widget y colocarlo de manera que esté aproximadamente a una pulgada del widget de programas abiertos. Al mantener todos los controles en un grupo reducido, mejoramos enormemente la experiencia del usuario al controlar las cosas con el gamepad.

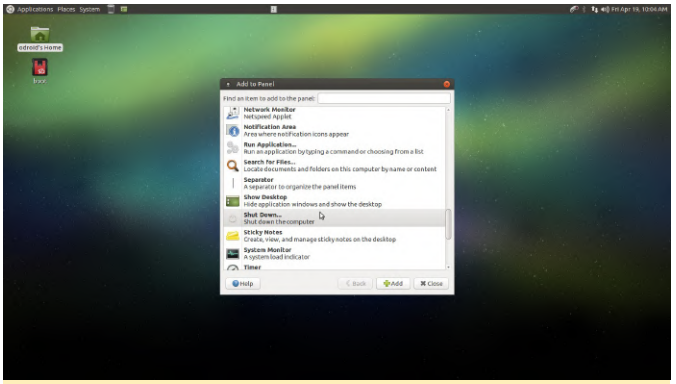


Figura - 22

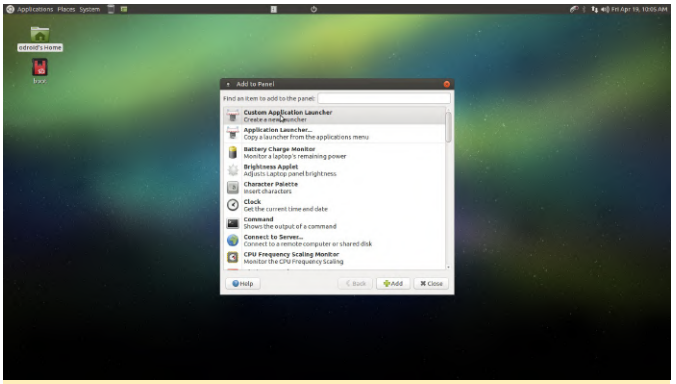


Figura - 23

A continuación, añadiremos dos botones personalizados al panel superior. Haz clic derecho en el panel y selecciona Add to Panel, elije la primera opción Custom Application Launcher, que se muestra arriba. Primero agregaremos el botón Stop y luego el botón Play. Los valores de los campos del formulario los tienes a continuación. Usa las capturas de pantalla para navegar hasta el icono adecuado. Puedes ver la ruta en la captura de pantalla cerca de la parte superior de la ventana.

```
Stop Button Values:
Type: Application
Name: Stop RetroArch
Command: /home/odroid/stop_auto
Comment: Stops RetroArch if running windowed.
(Really just stops
RetroArch and resets AntiMicro)

Start Button Values:
Type: Application
Name: Start RetroArch
Command: /home/odroid/start_auto
Comment: Starts RetroArch and AntiMicro scripts.
```

Las siguientes capturas de pantalla describen este paso. Úsalas a modo de ayuda para encontrar el icono correcto si fuera necesario.

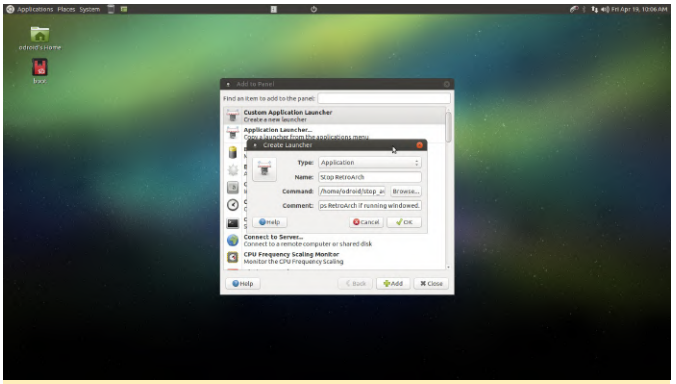


Figura - 24

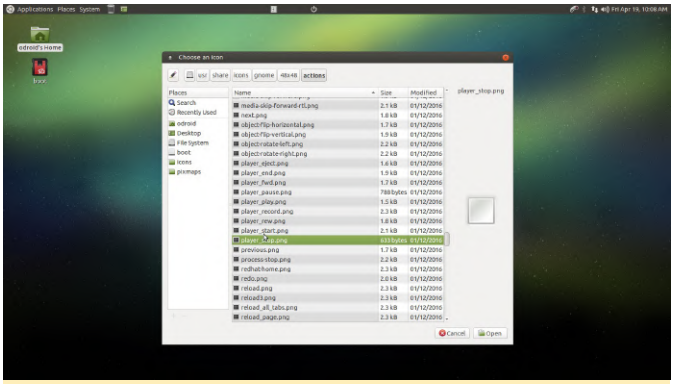


Figura - 25

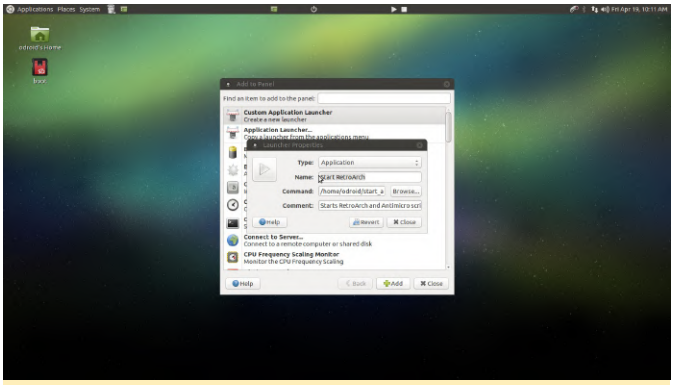


Figura - 26

Ahora vamos a probar los nuevos controles. Cierra retroarch y cualquier otra ventana abierta. Haz clic en el botón play, deberías ver la ventana emergente retroarch. Haga clic en el botón stop y Retroarch se cerrará, espere unos 5 segundos, y debería ver antimicro en la bandeja del sistema que nos devuelve el control total del gamepad.

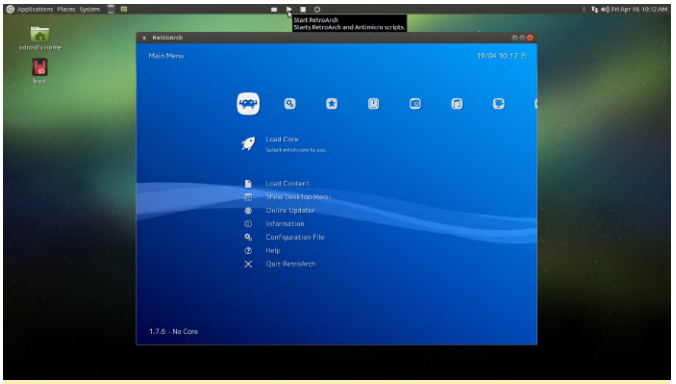


Figura - 27

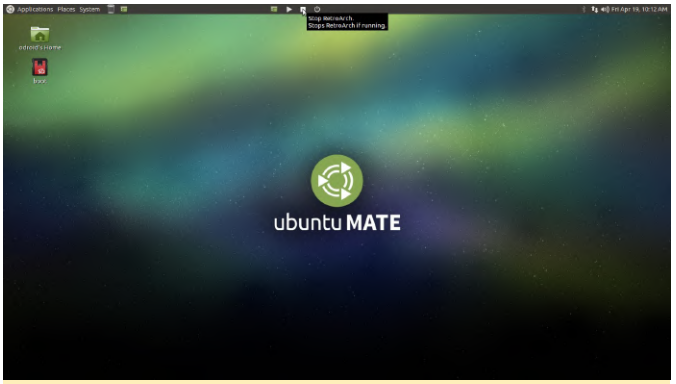


Figura - 28

Básicamente, hemos personalizado completamente nuestro hardware y software ODROID-GO para crear una consola de juegos retro con el modo de quiosco RetroArch y un completo entorno Linux, si fuera necesario. Lo siguiente que vamos a hacer es ejecutar RetroArch en modo de pantalla completa y ajustar algunos parametros de video. No me detendré en ninguna configuración avanzada. Este tutorial tiene la duración que me gusta para hacer los cosas, así que dedicare más adelante un pequeño tutorial de seguimiento a las erratas del emulador y a la configuración avanzada de RetroArch.

¡Inicia RetroArch, y podrás usar el pequeño widget que acaba de crear! Desplázate hacia la derecha hasta la sección de Drivers y localiza la entrada Video

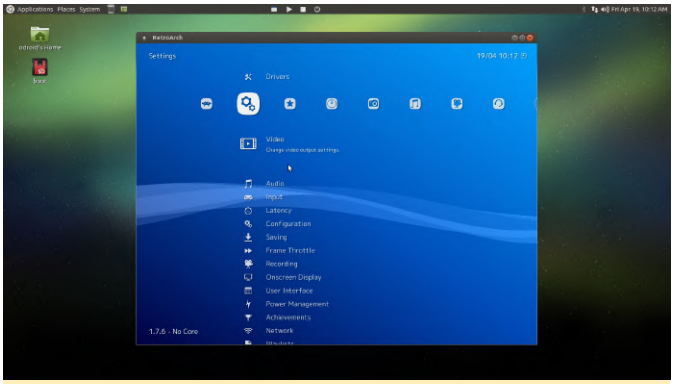


Figura - 29

Aplica las configuraciones que se muestran a continuación en el orden en que aparecen. La aplicación se puede cerrar y volver a abrir con algunas de las configuraciones; es normal.

```
Windowed Fullscreen Mode: Off
Show Window Decorations: Off
Threaded Video: On
Bilinear Filtering: Off
Start in Fullscreen Mode: On
```

Usa la tecla ESC para cerrar RetroArch cuando esté en modo de pantalla completa o utiliza el teclado/ratón para navegar hasta la sección Main Menu y selecciona Quit RetroArch.

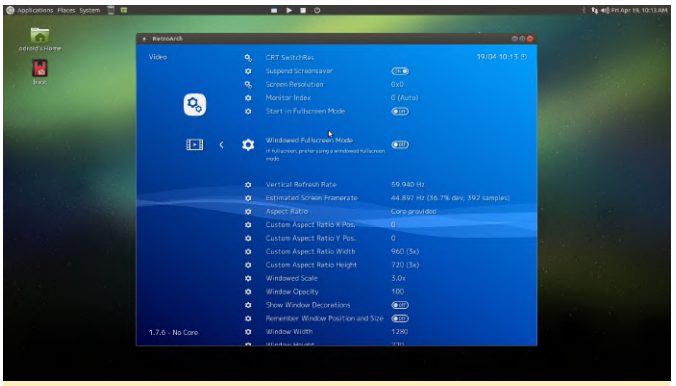


Figura - 30

Mantén presionado el botón de control personalizado durante 2 segundos y un poco más y el sistema se reiniciará. Cuando vuelva a iniciarse, deberías ver una pantalla completa de RetroArch como se muestra a continuación. Desplázate hasta tus ROM con el gamepad y arranca uno. ¡A jugar!

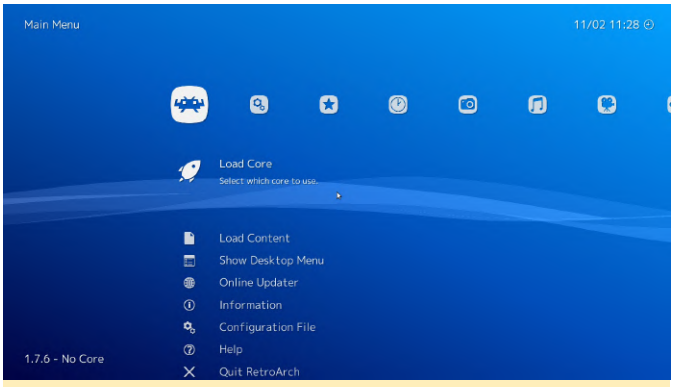


Figura - 31

Figura - 32

boot.ini

Esta parte es opcional para los dispositivos ODROID-XU4 y ODROID-XU4Q ya que tienen suficiente potencia para manejar 1080p y aún así proporcionan una excelente emulación del sistema. Sin embargo, si está interesado en tener un mayor control de la salida de video, no dude en seguir con este tutorial.

En este apartado, lo primero que haremos es hacer una copia de seguridad del archivo boot.ini. Abre el icono de inicio en el escritorio y copia boot.ini en boot.ini.orig. También vamos a hacer dos copias más: una que llamaremos boot.ini.1024x768p32bppVga y otra que llamaremos boot.ini.1280x720p32bppHdmi. Vamos a editarlos en breve. Deberías tener algo similar a lo que se muestra a continuación.

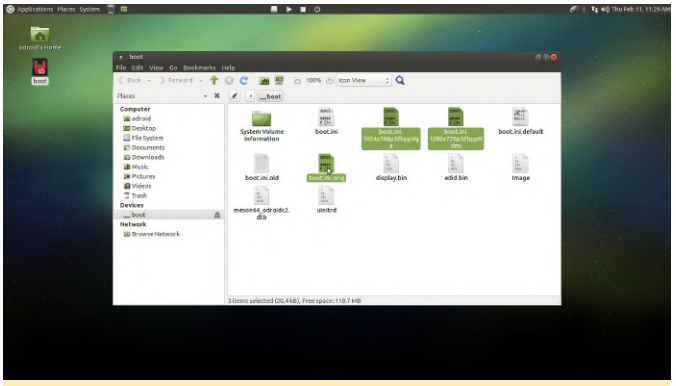


Figura - 33

Bajo mi experiencia, estas configuraciones de video funcionan bastante bien. La resolución VGA de 1024x768 es compatible con la mayoría, si no todas, de las pantallas recientes de ordenador, y la resolución de 720p es compatible con la mayoría, si no todos, los televisores recientes. Por supuesto, puedes seleccionar la que mejor te parezca. La idea es que, si tenemos conectando nuestro dispositivo a un televisor, pademos ejecutar un script personalizado para configurar la salida de video a HDMI 720p, o configurar la salida de video nuevamente al modo de detección automática original. Si queremos llevarnos el dispositivo al trabajo y queremos jugar un rato durante el almuerzo, ejecutaremos un script personalizado para configurar la salida de video a 1024x768 VGA.

Arranquemos nuestro dispositivo ODROID y cerremos RetroArch presionando la tecla escape en el teclado o navegando hasta la opción de salida usando el mando. Applications -> System Tools -> MATE Terminal, y escribe el siguiente comando.

```
nano set_1024x768_vga
```

Introduce las siguientes líneas en el archivo.

```
#!/bin/bash

sudo cp /media/boot/boot.ini.1024x768p32bppVga
/media/boot/boot.ini
sudo shutdown now
```

Guarda y salte del archivo. Luego ejecuta el siguiente comando.

```
nano set_1280x720_hdmi
```

Introduce las siguientes líneas en el archivo.

```
#!/bin/bash

sudo cp /media/boot/boot.ini.1280x720p32bppHdmi
/media/boot/boot.ini
sudo shutdown now
```

Guarda y salte del archivo. Luego ejecuta el siguiente comando.

```
nano set_auto_hdmi
```

Introduce las siguientes líneas en el archivo.

```
#!/bin/bash

sudo cp /media/boot/boot.ini.orig
/media/boot/boot.ini
sudo shutdown now
```

Guarda y salte del archivo. Estos scripts están configurados para cambiar el boot.ini hacía la salida de video deseada y luego apagarán el dispositivo para

que cuando lo conectes a un dispositivo, un televisor o monitor de ordeandor, se inicie con la salida de video correcta.

Asegúrate de asignar permisos de ejecución a tus scripts utilizando el comando de terminal sudo chmod + x seguido del nombre del script. También debes asegurarte de que los scripts tienen los permisos correctos ejecutando el siguiente comando, sudo chmod 755, seguido del nombre del script. Tienen que estar en el mismo directorio que los archivos que estás ajustando o usar la ruta completa al archivo. Si tienes la carpeta de inicio abierta en el explorador de archivos, puedes hacer clic con el botón derecho y seleccionar “Open in terminal“ para abrir un terminal que ya se encuentra en la ubicación correcta del sistema de archivos. Una vez llevado a cabo este paso, ajustaremos el archivo boot.ini copiado que creaste para reflejar el resultado y la resolución adecuados.

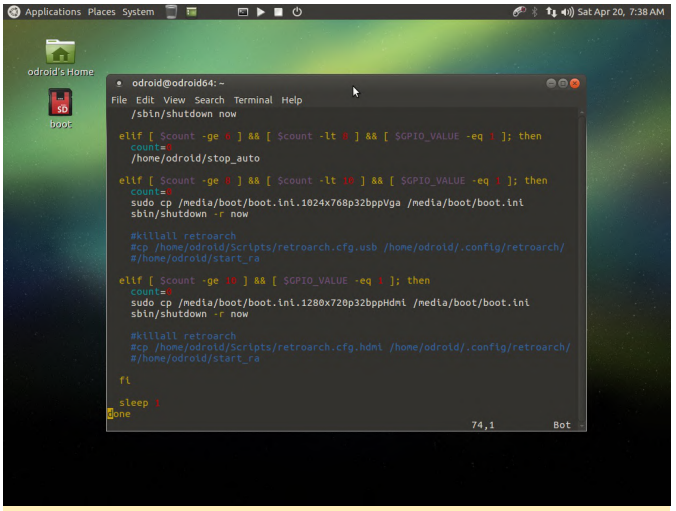


Figura - 34

Hay un ligero error tipográfico en esta imagen, las líneas que se leen sbin/shutdown -r ahora deberían leerse como sudo shutdown -r.

Básicamente, estamos copiando el archivo boot.ini con una versión preconfigurada que está configurada con una determinada resolución de pantalla, luego reiniciamos el dispositivo. Tratare los cambios que necesitamos hacer en cada archivo para el ODROID-XU4, pero también proporcionaré una descarga para que las cosas te resulten más fáciles. Echemos un vistazo.

Abre un terminal y escribe los siguientes comandos. Primero nos ocuparemos del modo VGA 1024x768.

```
cd /media/boot/
nano boot.ini.1024x768p32bppVga
```

Queremos fijar la salida de video en 1024x768 VGA. Comenta la línea que figura a continuación.

```
# setenv vout "hdmi"
```

Descomenta las siguientes líneas.

```
setenv videoconfig
"drm_kms_helper.edid_firmware=edid/1024x768.bin"
setenv vout "dvi"
```

Si cometes algún error, simplemente restaura la copia boot.ini.orig que realizaste anteriormente. Cualquier ordenador con Windows o Mac debería poder ver la partición de arranque de tu tarjeta SD del ODROID porque es una partición FAT32. Puedes usar esto para reparar tu boot.ini si el dispositivo no se inicia correctamente.

A continuación, activaremos el modo HDMI 720p. Abre una terminal y escribe los siguientes comandos.

```
cd /media/boot/
nano boot.ini.1280x720p32bppHdmi
```

Queremos ajustar la salida de video a 720p HDMI. Descomenta la línea que se detalla a continuación.

```
setenv videoconfig
"drm_kms_helper.edid_firmware=edid/1280x720.bin"
```

Muchas pantallas de ordenador no son compatibles con 720p. Si ese es tu caso, es probable que termines arrancando con una pantalla en blanco. Asegúrate de tener un televisor que permita usar esta resolución. Puede restaurar el boot.ini usando un PC con Windows o Mac montando la partición de arranque de la tarjeta micro SD. Ésta se montará automáticamente tanto en Windows como en Mac porque es una partición FAT32. Restaura el archivo boot.ini de la copia boot.ini.orig que realizaste anteriormente.

A continuación, tienes un archivo que incluye los archivos boot.ini necesarios que ya han sido preparados. Puedes seguir los pasos anteriores y

configurar las cosas a mano o puede usar el link a **Monku R3 / XU4 boot.ini** para ajilizar un poco las cosas.

Bueno, con esto concluye este tutorial. Deberías tener una configuración de consola de juegos retro bastante decente llegados a este punto. Tendrás que trabajar un poco más para configurar algunos de los emuladores que requieren una configuración más avanzada, aunque esto lo cubriré en un tutorial más adelante.

Juegos Linux: Anbox - Android en una Caja

November 1, 2019 By Tobias Schaaf Android, Juegos, ODROID-H2

Aunque normalmente solo suelo hablar de las placas ODROID basadas en ARM, hoy quiero detenerme en el ODROID-H2 con el que se puede hacer algo muy interesante. Puesto que ODROID-H2 es una placa estándar x86_64 (amd64), puedes llevar a cabo exactamente los mismos pasos de configuración en cualquier otro PC o ordenador portátil x86_64 con Linux.

Esta es mi experiencia con un paquete de software llamado Anbox, un "emulador" de Android para Linux, explicaré cómo configurarlo, qué hice con él y qué problemas encontré y cómo los resolví. Tomalo más bien como un recorrido interactivo mas que una guía rápida paso a paso, como he ido descubriendo varios secretos relacionados con la instalación y la configuración.

Anbox

Anbox es una nueva forma de emular Android, donde puede ejecutar Android directamente en tu sistema operativo Linux en lugar de iniciar Android directamente en el sistema o ejecutarlo en una VM.

Ya existen otras herramientas que pueden hacer lo mismo, aunque por lo general ocupan bastante, varios GB de almacenamiento solo para iniciar el emulador, por ejemplo, Android Development Studio.

Anbox utiliza LXC, llamados Contenedores Linux, una técnica similar a Docker, donde puede ejecutar un sistema operativo en un entorno encapsulado, pero sigues compartiendo el Kernel y los recursos del sistema. Esto también significa que si el ODROID-H2 tiene 16 GB de RAM, técnicamente tu Android también funciona con 16 GB de RAM.

Para hacer esto, hay una característica disponible tanto en el Kernel de Linux como en Android que es muy necesaria, se llama Ashmen y Binder, estos son dos módulos disponibles para el Kernel.

Instalación de Módulos kernel

Raramente un Kernel viene precompilado con esta funcionalidad, pero afortunadamente hay módulos DKMS disponibles para esto.

Existe una guía muy buena sobre cómo empezar con todo
esto: **https://docs.Anbox.io/userguide/install_kernel_modules.html**

En resumen, estas son las dos cosas que puedes hacer: 1. Instalar módulos dkms desde PPA:

```
$ sudo add-apt-repository ppa:morphis/Anbox-support
$ sudo apt update
$ sudo apt install linux-headers-generic Anbox-modules-dkms
```

2. Instalar módulos dkms desde GitHub:

https://github.com/Anbox/Anbox-modules

Te sugiero usar el PPA o crear tu propio archivo deb, ya que instalará automáticamente todos los archivos necesarios y los colocará en la ubicación correcta.

Los módulos Kernel crearán /dev/binder y /dev/ashmem como nuevos dispositivos necesarios para que Android acceda a los recursos de nuestro sistema host. No te preocupes si no los ves de inmediato. O bien tiene que cargar los módulos de Kernel manualmente o bien tienes que esperar hasta que instalemos Anbox.

Instalar Anbox

La distribución principal de Anbox está a través de paquetes sanp, que están disponible en muchos sistemas Ubuntu y en otras distribuciones de Linux. La segunda opción es instalar paquetes deb directamente desde el repositorio de tu sistema operativo.

A partir de Debian Stretch (backports) o Ubuntu 19.04, Abox existe como paquete oficial en su repositorio. Lo probé en mi ODROID-H2 en Debian Buster, aunque también lo hice funcionar en mi ordenador portátil que utiliza Ubuntu 18.04. Como he dicho, solo está disponible oficialmente en Ubuntu 19.04, pero si sabes cómo, también puedes compilarlo en Ubuntu 18.04 (que es lo que yo hice).

Puesto que ODROID-H2 viene con Ubuntu 19.04, no deberías tener problemas para instalarlo.

De cualquier modo, debes asegurarte de tener suficiente espacio en /var/. Si tienes tu sistema particionado y no utilizas una sola partición para todo el sistema operativo, asegúrete de tener suficiente espacio en /var/ para todas las aplicaciones que quieres instalar. Éstas pueden necesitar fácilmente hasta 20 GB o más dependiendo de las que quieras probar.

¿Por qué deberías optar por los paquetes deb sobre los paquetes snap?

La razón por la que debes usar los paquetes deb para Anbox es simple, al igual que con los módulos Kernel, si instalas Anbox como un paquete deb, éste ya vendrá con muchas cosas ya instaladas y preparadas para usarse. Tiene un servicio llamado Anbox-container-manager que es gestionado por systemd y se encarga de todas las pequeñas tareas de segundo plano. Iniciará automáticamente los módulos del kernel para /dev/binder y /dev/ashmem, así como un adaptador de red virtual para que Android pueda tener acceso a Internet.

Es muy cómodo y permite una fácil gestión si deseas probar cosas nuevas con posterioridad.

Los paquetes snap, por otro lado, tienen algunos problemas importantes.

La mayor ventaja de los paquetes snap es que viene con un archivo Android.img, mientras que, si instalas el archivo deb, tendrás que descargar el archivo Android.img por tu cuenta (lo explicaré más adelante).

Aunque los paquetes snap por lo general funcionan, desaconsejo usarlos ya que son muy inflexibles y pueden causar problemas importantes si quieres añadir Google Apps, por ejemplo, o si deseas probar parches.

Primero empecé con paquetes snap y funcionó "bien", pero cuando intenté instalar Google Apps (Gapps) a través de un script, el sistema no lograba iniciarse y me llevó un día o dos descubrir que el problema estaba relacionado con paquetes sanp y no

con la versión de Anbox, o con las Gapps que estaba usando.

Los paquetes Snap son contenedores bloqueados, no puedes modificarlos de ningún modo, significa que no puedes cambiar la configuración, actualizar aplicaciones, intercambiar la imagen de Android, etc. a menos que instales un nuevo paquete Snap de esa aplicación.

Para alterar el paquete snap y agregar Gapps, por ejemplo, necesitas trabajar con "overlay-fs" para el paquete sanp, donde se combinará una carpeta que has creado con los datos proporcionados por el paquete sanp. En mi caso, en el momento en el que activé overlay-fs, incluso sin ningún cambio, la mayoría de mis aplicaciones ya no funcionaron.

Con el paquete deb no tiene estas restricciones y puedes cambiar el archivo Android.img directamente, así como manipular rootfs o la carpeta de datos de la imagen de Android si fuera necesario. Esto te permite colocar archivos directamente en tu sistema de archivos virtual o en el propio archivo img, lo cual resulta imposible con los paquetes snap.

Mi consejo: aunque son el eje central de la distribución Anbox, ¡mantente alejado de los paquetes snap!

Primer Arranque

¿Qué necesitas para poner en marcha tu primera experiencia Android con Anbox?

Bueno, con los paquetes snap, podrías iniciarlo sobre la marcha, pero como hemos dicho que no vamos a usar paquetes snap, necesitamos una imagen Android para trabajar.

Algunas cuestiones técnicas

Estás ejecutando Anbox en un contenedor LXC, esto significa que no está "emulado", tampoco es realmente una "virtualización", aunque está muy cerca. Utiliza el mismo hardware y Kernel que tienes en tu sistema operativo principal. Lo que significa que tu sistema operativo Android debe adecuarse a la configuración de tu PC.

Conseguir una imagen de Android

Necesitamos una imagen de Android x86_64 (amd64). Esto puede sonar complicado, pero no te preocupes, hay otros ya han hecho todo el trabajo y solo necesitas descargar una imagen desde aquí: **https://build.Anbox.io/Android-images/** como puedes ver, hay varias imágenes disponibles. Puedes elegir cualquier imagen que tenga "amd64" en su nombre. Utilizaremos la última por ahora: **https://build.Anbox.io/Android-images/2018/07/19/Android_amd64.img**.

Todas estas imágenes incluye Android 7.1.1, ya que éste es el utilizado por la mayoría de los emuladores.

Descárgate la imagen anterior y cámbiale el nombre a Android.img, copia el archivo en el sistema donde instalaste Anbox (a través de deb o desde tu repositorio) en /var/lib/Anbox/

```
$ sudo wget https://build.Anbox.io/Android-images/2018/07/19/Android_amd64.img -O /var/lib/Anbox/Android.img
```

Esto es todo para echar a andar

¿Recuerdas el servicio Anbox-container-manager de antes?

En el inicio comprueba si hay un archivo Android.img en esa carpeta y, de ser así, activará automáticamente los módulos del kernel y el puente de red necesario para el acceso a Internet en Anbox.

Como ahora tenemos un Android.img ubicado en /var/lib/Anbox/, es hora de iniciar el servicio:

```
$ sudo service Anbox-container-manager restart
```

Ten en cuenta que utilicé “restar”, ya que es posible que tengas que usar el comando más tarde nuevamente, así que ya lo tiene en tu historial de comandos.

Si revisas tu menú de inicio y buscas Anbox (debajo de accesorios), debería ver una pantalla de inicio muy minimalista con solo unas cuantas aplicaciones.

Calculator, Calendar, Clock, Contacts, Email, Files, Gallery, Music, Settings y WebViewer son probablemente las únicas aplicaciones que podras ver. Y aunque todas funcionan bien, a excepción de "Settings", probablemente nunca necesitarás ninguna

de estas aplicaciones. Aún así, si ves esta pantalla, significa que Anbox funciona.

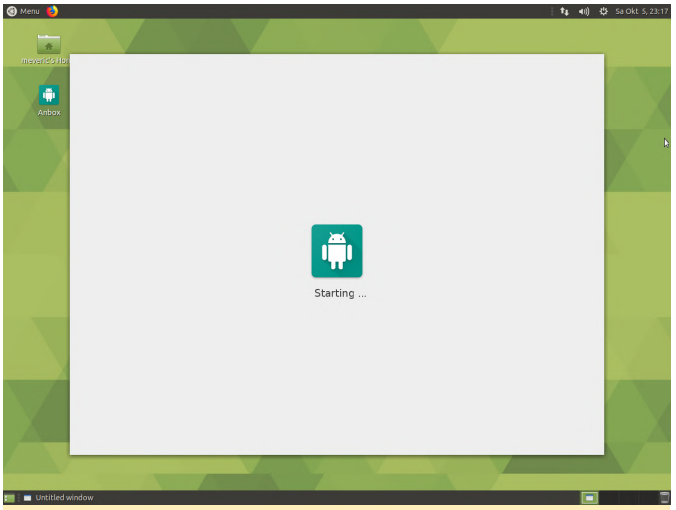


Figura 01 - Anbox ejecutándose con Android

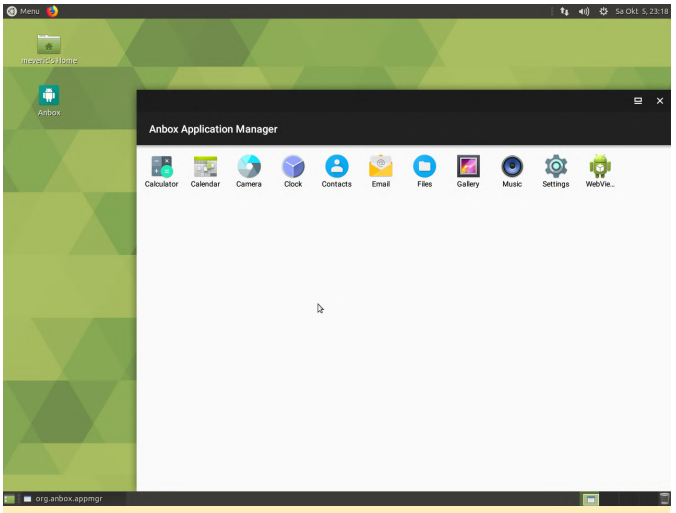


Figura 02 - Administrador de aplicaciones de Anbox

Iniciar Anbox por primera vez (puede tomar algunos segundos) Menú de la aplicación Anbox con aplicaciones por defecto.

Instalar aplicaciones - parte 1

Tener "Android" funcionado esta bien y es elegante, pero si sólo puedes ejecutar el calendario o el reloj, éste se volverá aburrido muy rápido. Así que, lo que necesitamos es una forma de instalar aplicaciones. Hay diferentes formas de hacer esto.

Comencemos con la más fácil, que requiere que ya tengan archivos .apk disponibles.

Primero necesitamos instalar Android-toolbox que está disponible para tu sistema operativo:

```
$ sudo apt install Android-tools-adb
```

Con esto, puedes instalar aplicaciones fácilmente escribiendo:

```
$ adb install
```

Android toolbox está diseñada para TODOS los sistemas Android, no específicamente para Anbox.

Éste busca cualquier dispositivo conectado permitiéndote interactuar con él.

Si tienes Anbox instalado y funcionando, puede usar adb para instalar la aplicación en el sistema.

Ten en cuenta que si tiene un dispositivo Android, como un smartphone o una tablet, conectado a tu PC/ordenador/ODROID H2 con Linux, puedes terminar instalando aplicaciones en este dispositivo, así que asegúrate de usarlo solo cuando Anbox se esté ejecutando (a menos que sepas lo que haces).

¿Te estarás preguntando dónde puedes conseguir archivos apk para instalar? Existen varios sitios donde puedes descargar archivos apk, a menudo los programas o juegos tienen enlaces de descarga en su sitio web y también hay sitios como APKMirror (https://www.apkmirror.com/) que te permiten descargar archivos apk directamente para tu sistema.

Nota Importante

¿Recuerdas que descargaste la imagen de Android_amd64? Esto significa que tienes una imagen Android basada en x86_64 y solo puede ejecutar aplicaciones de Android x86 o x86_64, o aplicaciones que no requieren un formato binario específico. Fíjate que en APKMirror, puede ver qué paquetes de arquitectura tienen antes de descargarlos. Asegúrate de descargar los correspondientes a x86.

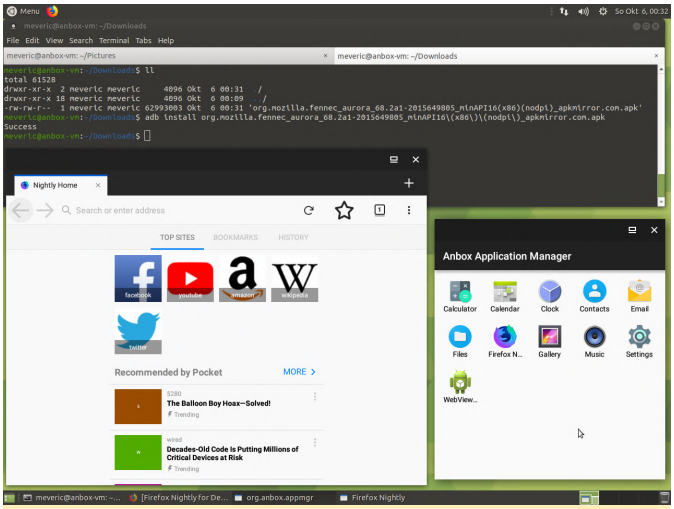


Figura 3: Instalando y ejecutando FireFox Nightly (x86) en Anbox mediante adb install desde APKMirror

Con esto, ya puedes instalar las primeras aplicaciones y probarlas con Anbox. Para serte sincero, mi objetivo cuando empecé a usar Anbox era ejecutar Ragnarok M - Eternal Love. Así que, mi primera prueba real consistia en descargar el archivo APK de su sitio web e instalarlo. Así que descargué el apk de su sitio web, y tras completarse los 1.3 GB lo instalé a través de adb:

```
$ adb install 155603_20190.1563284609.apk
```

Gracias a la potencia del SSD, unos segundos después de instalar el juego, pinche en el icono y el juego se inició, no solo quedé impresionado para ser la primera aplicación que probé, sino que de hecho me gusto bastante (más aún)

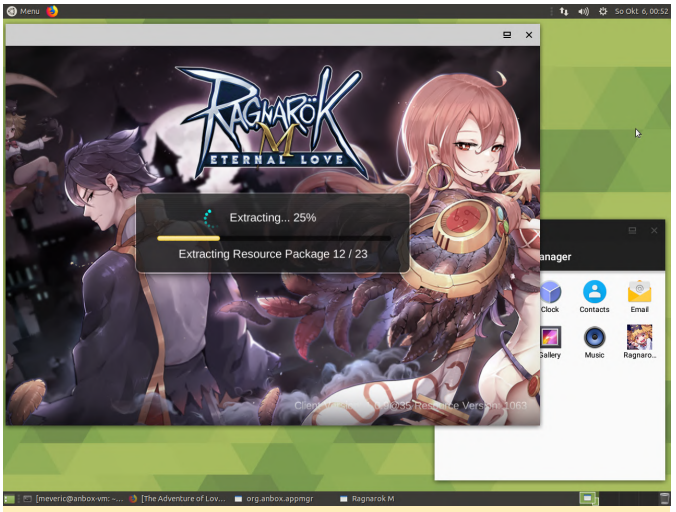


Figura 4: Ragnarok M iniciado desde Anbox e instalando actualizaciones

El juego se inició y, como es habitual en este tipo de juegos online, empezó a descargar e instalar las actualizaciones, las cuales se completaron correctamente, así que me senté y esperé, contento de ver que el juego era mucho más rápido que en mi tablet (lo cual no me sorprendió demasiado). Tras las actualizaciones, fui redirigido a la pantalla de inicio de sesión y aquí es donde empezaron los problemas.

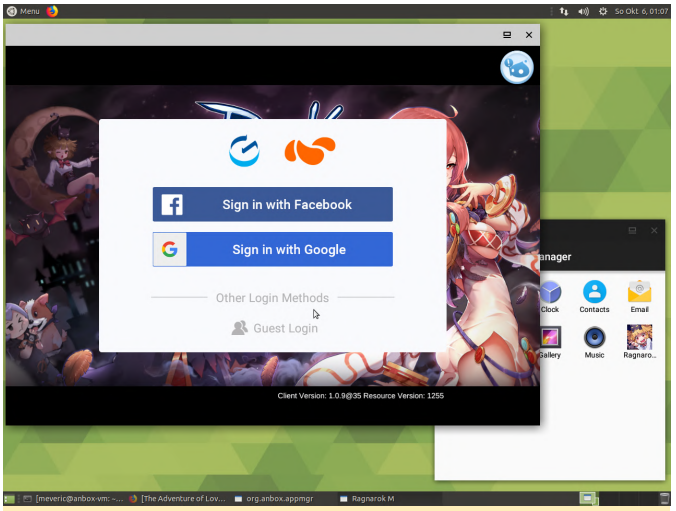


Figura 5 - Lo único que queda por hacer es iniciar sesión y empezar a jugar ¿verdad?

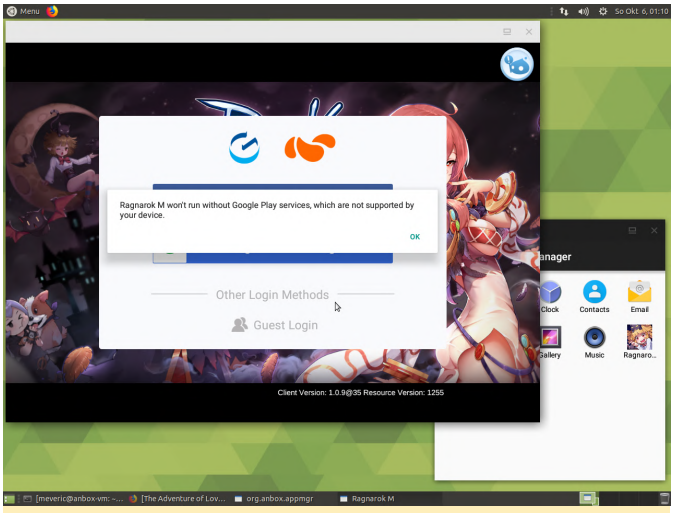


Figura 6 - ¿Servicios de Google Play? Ah sí, sabía que había algo diferente

Aquí es donde empezó mi terrible experiencia y donde desperdicié un día entero para descubrir que el problema estaba en los paquetes snap. Pero antes de entrar en detalles y de cómo resolver el problema, unas cuantas observaciones generales sobre las cuestiones que vamos a tratar.

Instalar aplicaciones parte 2 - Gapps y otros paquetes

Si has visitado APKMirror, probablemente abras observado que la mayoría de las aplicaciones y juegos solo están disponibles para arm o arm64 (armeabi-v7a, armeabi, arm64-v8a), muy pocos hay disponibles para x86.

Esto es cierto, ya que la mayoría de los juegos y aplicaciones están escritos para smartphones y tablet y esos dispositivos normalmente tienen un procesador ARM. Por lo tanto, la selección de aplicaciones que se ejecutan directamente en Android x86 es muy limitada.

Aparte de esto, tendrás dificultades para encontrar tus juegos favoritos en APKMirror, de hecho, yo tuve dificultades para encontrar aplicaciones que funcionasen adecuadamente.

La mayoría de los juegos solo estarán disponibles una vez que hayas instalado Gapps y Play Store de Google, pero incluso con x86 la selección continúa siendo muy pequeña. Muchas aplicaciones dependen de los servicios de Google Play, incluso si cuentas con el archivo apk para la instalación, los programas requieren servicios de Google Play para funcionar, por lo tanto, instalar Gapps es muy importante.

Instalar Gapps y Houdini

Existen varias formas de instalar Gapps para Anbox, pero no son muy fáciles que digamos y, por eso, es mejor confiar en otras personas que ya han hecho todo el trabajo por nosotros y han escrito un script para ello. Actualmente hay dos que son muy importantes para nosotros:

https://github.com/geeks-r-us/Anbox-playstore-installer/raw/master/install-playstore.sh que es un script para instalar gapps y houdin, si has decidió usar el paquete snap por alguna razón. Creará un Overlay FS y extraerá los archivos allí. Esto quizás funciones en tu caso, en el mío dejó Anbox inutilizable tras la ejecución.

El otro es:

https://github.com/Arucard1983/Anbox-playstore-installer/raw/debian/install-playstore.sh Este último está destinado específicamente a la instalación deb y funciona directamente con Android.img en lugar de Overlay FS. Así que vamos a intentarlo:

```
$ wget https://github.com/Arucard1983/Anbox-playstore-installer/raw/debian/install-playstore.sh
$ chmod +x install-playstore.sh
$ sudo apt install lzip curl
$ sudo ./install-playstore.sh
```

Para los que están interesados en lo que hacemos hacemos aquí: Se descargará open-gapps de sourceforge, así como houdini_y y houdini_z del proyecto Android-x86. Se extraerán y se agregarán a la imagen. Houdini son librerias que te permiten ejecutar binarios de Android arm y arm64 en un entorno x86. El script alterará Android.img para que ya no solo actué como una imagen x86 y x86_64 sino también como una imagen arm y arm64. Lo que significa que después de esto puede instalar y ejecutar el resto de aplicaciones que antes no estaban disponibles para arm y arm64 en el sistema, añadirá algo de optimización y configuración para que todo funcione correctamente.

Luego, crea un nuevo archivo Android.img y reemplaza el original con el nuevo que tiene gapps y soporte para arm y arm64.

Todo el proceso sólo requiere unos minutos para descargar, extraer y volver a empaquetar la imagen.

El comando sudo solo es necesario para ubicar el nuevo archivo de imagen en /var/lib/Anbox, donde solo root tiene acceso; si lo ejecutas sin sudo, seguirá creando el archivo Android.img y luego podrás copiarlo manualmente.

```
$ sudo install-playstore.sh --clean
```

Se eliminará el directorio de trabajo de Anbox que se creó durante el proceso, pero también puede eliminar esa ruta manualmente, ya que no es necesaria.

Configurar los servicios de Google Play

Después de reiniciar si iniciamos Anbox, tenemos un icono adicional para Google Play Store que nos permitirá instalar aplicaciones y juegos directamente desde la tienda, pero antes de que podamos hacer esto, necesitamos configurar los permisos para los Servicios de Google Play y la Tienda.

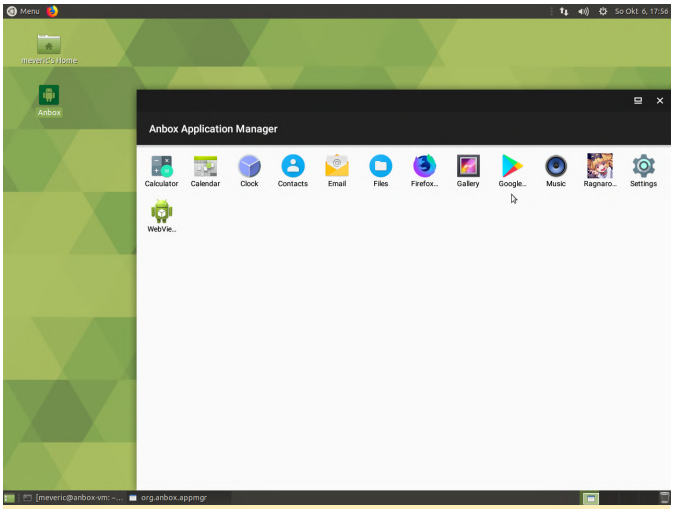


Figura 7: Iniciando Anbox después de instalar Gapps

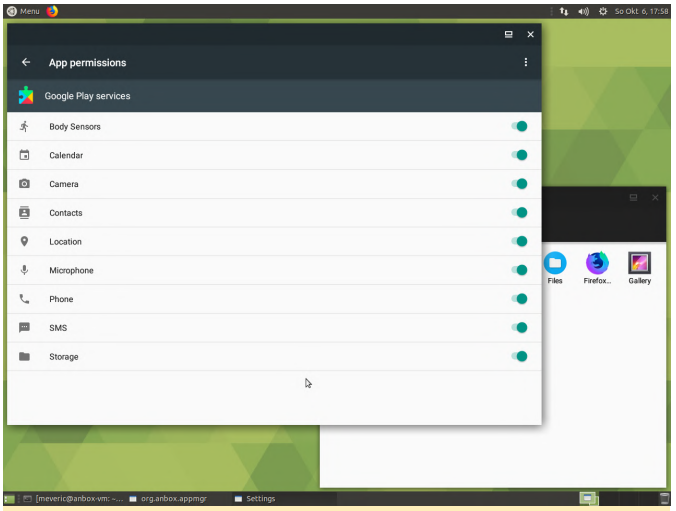


Figura 8: Configurando permisos para Google Play Service y Google Play Store

Para esto, abre tu Configuración, dirígete a Aplicaciones, busca y haz clic en Google Play Services

Debajo de los botones verá Storage, y luego verás Permissions, que están vacíos o incluye una lista muy corta. Haz clic en los permisos y en Google Play Services, activa todos los permisos de la lista (consulta la figura 8). Haz lo mismo con Google Play Store.

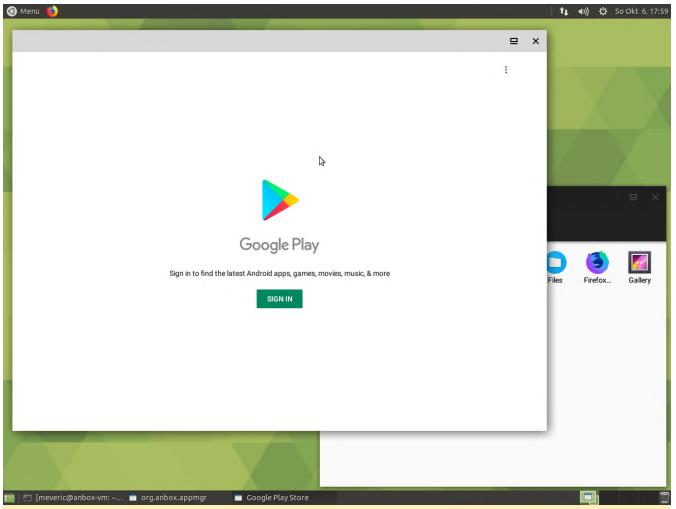


Figura 9: Google Play Store en Anbox: solo necesitamos iniciar sesión

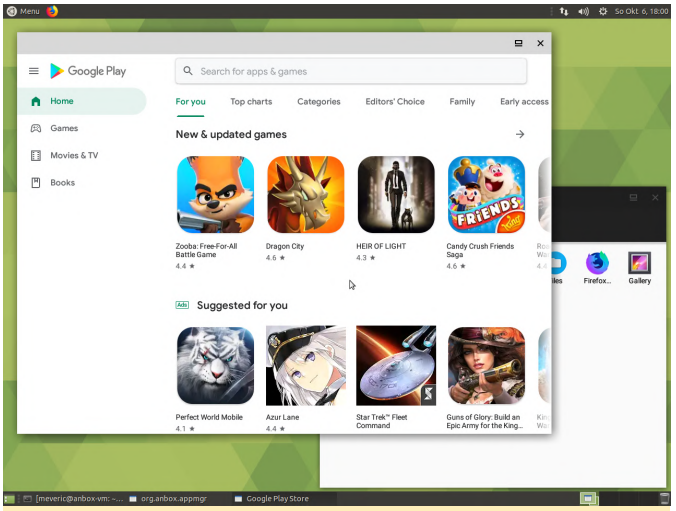


Figura 10: Tras iniciar sesión, dispondremos de Play Store para instalar juegos

Si lo has hecho todo bien, deberías poder iniciar Google Play Store y ver aparecer un botón SIGN IN. Una vez que hayas iniciado sesión con tu cuenta de Google, podrá ver Play Store y empezar a instalar software. Y simplemente para demostrar que funciona:

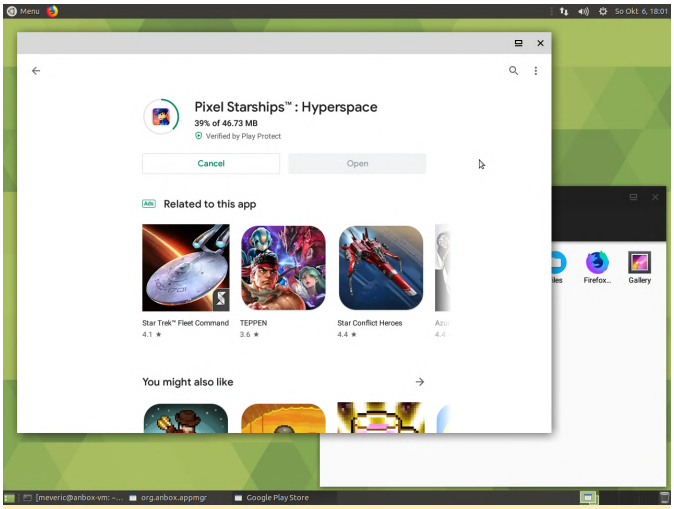


Figura 11 – Descargando e instalando un juego en Anbox

Figura 12 - ¿Adivina qué? Incluso podemos JUGAR a juegos en Anbox

Llegar a escuchar y ser capaz de hacer lo mismo debería ser bastante fácil y no debería implicar demasiados conocimientos. Me llevó un tiempo resolver este tema, pero supongo que mi información debería acelerar el proceso si quieres probarlo tú mismo.

Aunque llegado a este punto podía usar y explorar Anbox, mi peregrinación aún no había terminado, ya que mi objetivo original era ejecutar Ragnarok M en el PC en lugar de hacerlo en mi tablet. Lo cual aún no había logrado.

Soporte Táctil

Cuando inicié Ragnarok por primera vez después de añadir Google Play Services y finalmente pude iniciar sesión en el juego, esperaba que pudiera jugar, pero tuve problemas para iniciar sesión con mi personaje. De hecho, tuve problemas para hacer cualquier cosa. Si hacia clic en algo, no hacía nada. Tras investigar un poco, quedó patente donde estaba el problema.

Como he dicho antes, la mayoría de los juegos se desarrollan para teléfonos y tabletas. Estos dispositivos normalmente no tienen un ratón conectado y, por lo tanto, muchos juegos se escriben para admitir únicamente la entrada táctil. La versión de Anbox que viene con Ubuntu 19.04 y Debian Stretch/Buster admite entrada de ratón y táctil. Esto significa que, si tiene un ratón conectado, reaccionará como un ratón, si tienes una pantalla con entrada táctil, usará el sistema táctil. Como no todo el mundo tenemos una pantalla táctil con ratón, lo más probable es que sea este último la entrada por defecto en su caso, como lo es en el mío.

Como el juego al que quería jugar no tenía soporte para ratón, investigué un poco y terminé con un parche que permite que la entrada del ratón actúase a modo de táctil. Llegados a este punto tenemos dos opciones, te las voy a explicar:

Parchear el paquete Debian

Una forma es crear nuestro propio paquete deb basado en uno de los SO con un parche que nos permitirá usar el ratón como entrada táctil.

Ten en cuenta que esta opción requiere que vuelvas a compilar el software y que instales las cabeceras de desarrollo y el compilador en tu sistema. Si no quieres hacer esto, puedes recurrir a mis versiones ya parcheadas de estos paquetes: https://oph.mdrjr.net/meveric/other/Anbox/ donde puedes descargar el paquete para tu sistema operativo.

También he incluido un paquete para Ubuntu 18.04 en caso de que esté utilizando esta versión LTS, como lo hago yo en mi ordenador portátil.

Si te decantas por esta opción, asegúrate de haber activado las listas de origen para el repositorio de tu sistema operativo.

Dirígete a Software and Updates, dentro de tu Control Center, por ejemplo, y asegúrate de que la casilla " Source Code" esta marcada.

Para aquellos que desean compilar el Software ellos mismos, aquí teneis los pasos que debeis seguir:

```
$ mkdir -p ~/sources
$ cd ~/sources
$ apt source Anbox
$ cd Anbox-0.0~git20190124
$ sudo apt build-dep Anbox
$ wget
https://oph.mdrjr.net/meveric/other/patches/Anbox/
touch-support.diff
$ patch -p1 < touch-support.diff
$ dpkg-buildpackage -b
$ sudo dpkg -i ../Anbox_0.0~git20190124-1_amd64deb
```

Ten en cuenta que la versión real (Anbox_0.0 ~ git20190124) puede diferir dependiendo del sistema operativo que utilices, por lo tanto, ajusta esta parte en consecuencia.

Esto compilará e instalará Anbox con soporte táctil. Una vez que reinicies Anbox y uses el nuevo binario para iniciar los juegos, el soporte táctil funcionará.

Compilar la última versión de Anbox desde Github

En lugar de parchear una versión anterior de Anbox, también puede optar por compilar la última versión disponible en Github de Anbox. No necesitarás el parche para soporte táctil puesto que éste ya está incluido en la última versión.

```
$ mkdir -p ~/sources
$ cd ~/sources
$ sudo apt install git libdw-dev libdwarf-dev
binutils-dev libboost-serialization-dev libboost-
thread-dev libboost-test-dev
$ git clone https://github.com/Anbox/Anbox
$ cd Anbox
$ sudo apt build-dep Anbox
$ mkdir build
$ cd build
$ cmake -DCMAKE_BUILD_TYPE=Release ..
$ make -j4
$ sudo make install
```

Esto generará e instalará la última versión de Anbox. Dado que esta versión se instalará en /usr/local/bin y no en /usr/bin, puedes instalar el paquete deb desde tu sistema operativo y la versión desde github a la vez y puede alternar entre ellos.

Es por ello que instalamos la versión deb de Anbox, ya que nos permite cambiar fácilmente de una versión a otra simplemente editando un archivo, que es /etc/systemd/system/multi-user.target.wants/Anbox-container-manager.service (deberías recordar esto de antes)

Cuando edites el archivo con sudo y tu editor de texto favorito, encontrarás la línea:

```
ExecStart=/usr/bin/Anbox container-manager --
daemon --privileged –data-path=/var/lib/Anbox
```

Simplemente necesitas cambiarla por:

```
ExecStart=/usr/local/bin/Anbox container-manager -
-daemon --privileged –data-path=/var/lib/Anbox
```

Entonces se inicia el binario que acaba de compilar. Luego simplemente reinicia el sistema.

Después de ello, se usará tu nueva versión de Anbox compilada.

Si en algún momento más adelante (un par de semanas despues) desea volver a compilar la última versión de Anbox, simplemente haz lo siguiente:

```
$ cd ~/sources/Anbox/build
$ git pull
$ make -j4
$ sudo make install
```

Eso es todo y siempre tendrás la última versión. ¿Eso significa que funciona ahora? ¡Con esto finalmente logré jugar Ragnarok M en mi ordenador portátil (o ODROID-H2) y déjame decirte que funciona estupendamente! Es sorprendentemente estable, lo cual no me esperaba.

Figura 13 - Jugando a Ragnarok M - Eternal Love en mi ordenador portátil, ¡Es mucho más fácil que en la tablet y no necesita estar cargando todo el tiempo!

Con esto, ahora estás preparado para probar Anbox y jugar a juegos de Android directamente en Linux, y sí, esto es lo mejor que puedo ofrecerte de Anbox, no

tengo nada más que aportar, por ahora. Pero no todo es tan bonito y brilla por ausencia.

Seamos honestos, aunque impresiona, no significa que ya puedes deshacerte de tu tablet Android. ¿Recuerdas lo que dije antes sobre el hecho de que tuve mucha suerte con el primer juego que probé (Ragnarok M), el cual se ejecutaba sin problemas?

Lamentablemente es algo que debes tener en cuenta. El número de aplicaciones y juegos que SI funcionan es MUY limitado. Si consigues que uno de cada diez funcione, vas por el buen camino. Tengo alrededor de 75 juegos de Android en mi humilde biblioteca a modo de archivos .apk descargables. De los cuales menos de 10 funcionan. Esto no está relacionado con el tema de los gráficos, sino que los juegos ni siquiera se inician. Los que tiene un problema con los "gráficos" están en una categoría aparte.

Cuando pruebas aplicaciones y no se abre una nueva ventana, el juego simplemente no funcionará, puede dejar de intentarlo.

A menudo, los juegos se cuelgan en este punto, incluso aunque escuches sonidos, música o lo que sea, no llegan a funcionar. A veces es mejor reiniciar el servicio Anbox-container-manager en lugar de esperar a que la aplicación agote el tiempo para poder hacer clic en otra aplicación.

Si no me equivoco, esto está relacionado principalmente con la forma en la que funciona Anbox. Utiliza llamadas EGL para crear una ventana en la cual se ejecuta la aplicación. Para esto, la aplicación debe ser compatible con el "Modo de ventana", imagínate algo similar a la aplicación de YouTube, cuando la inicias y luego vuelves atrás, aparece una pequeña ventana en tu dispositivo móvil que se puede arrastrar. Si una aplicación admite algo similar a esto, probablemente funcione. Si no es así, entonces estás perdiendo el tiempo.

Básicamente, esto significa que la mayoría de las aplicaciones "antiguas" no funcionarán ya que el modo de ventana era desconocido en su día. De hecho, cuanto más nueva sea una aplicación, es más probable que funcione. Algunas aplicaciones están escritas para ser compatibles con emuladores, como mi Ragnarok M que quería que funcionara, con este sporte del modo ventana de una forma u otra. De hecho, encontré muchos juegos MMORPG online y otros que funcionan bastante bien.

También está limitado a las aplicaciones OpenGL ES 2.0. De forma que, si una aplicación requiere OpenGL ES 3.x, no funcionará.

También existe un problema con el sonido, todos los sonidos tienen un retardo de aproximadamente 1 segundo. De manera que ves cosas en la pantalla, pero las escucha un segundo después, lo cual puede llegar a ser un poco molesto. He visto un parche que reduce este retraso del audio significativamente, pero cuando los probé, los juegos dejaron de funcionar por completo al activarse el sonido.

Pero supongo esto se solucionará en algún momento.

¿Qué PUEDES ejecutar?

Por terrible que parezca, aun puedes ejecutar algunos juegos realmente buenos en Anbox.

Ya has visto Ragnarok M, que se ve bastante bien y tiene gráficos de alta resolución. Es un MMORPG realmente agradable y yo, como fanático de los MMORPG originales de Ragnarok Online, he disfrutado de este juego durante meses. También he mostrado fotos de Pixel Starships, otro juego con el que estoy disfrutando desde hace algún tiempo, aunque es más informal.

Anodyne es un buscador de mazmorras de la vieja escuela donde tu arma principal resulta ser una escoba.

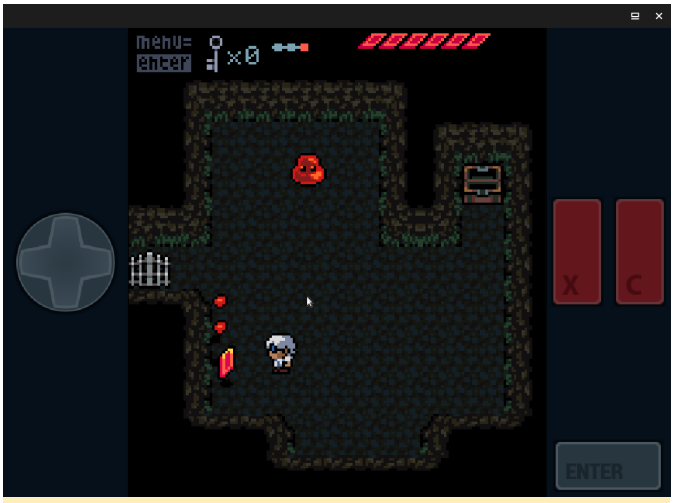


Figura 14 - Con una escoba atacas a los monstruos ... Anodyne

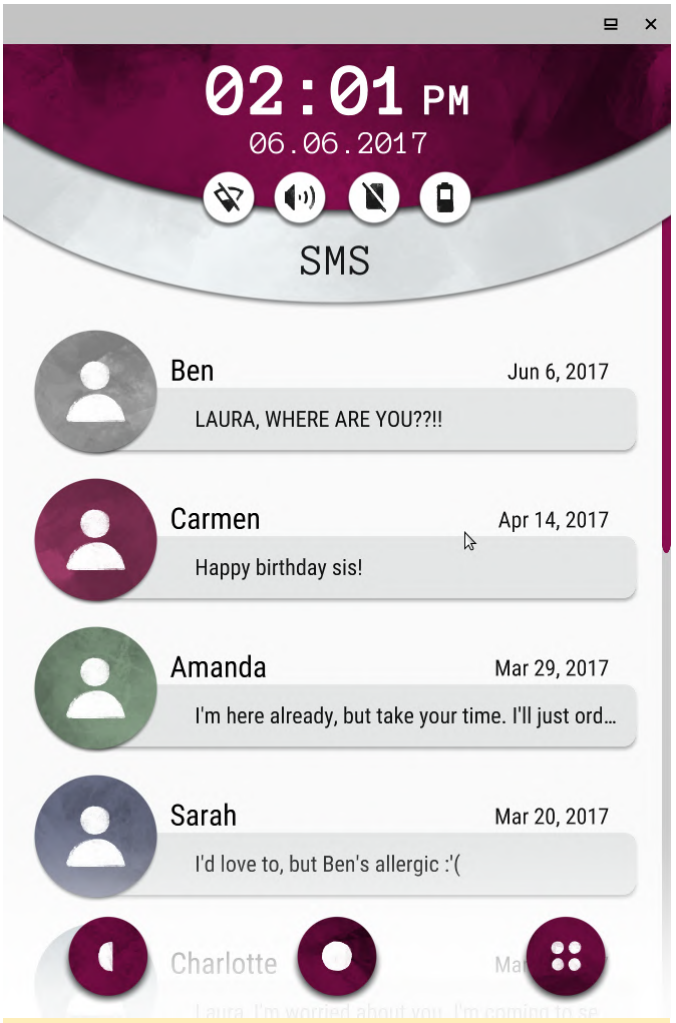


Figura 15 - Another Lost Phone: Laura's Story

Es bastante divertido y se puede controlar perfectamente con el teclado, lo que significa que no necesitas navegar con el raton por la pantalla. Realmente es bastante entretenido a pesar de sus gráficos minimalistas. Another Lost Phone: Laura's Story es un juego en el que encuentras el teléfono de alguien llamado Laura y parece que algo malo le ha sucedido. Intentas averiguar más sobre ella revisando su teléfono, revisando fotos, mensajes, etc. A medida que avanzas en el juego, tienes que adivinar las contraseñas combinando pistas, averiguando quiénes son las personas que aparecen en las imágenes y cómo se relacionan con Laura. Puede que no impresione demasiado, pero realmente se está ejecutando sobre Unity Engine. Es interesante ir resolviendo los pequeños acertijos y descubrir quiénes son las diferentes personas que se ponen en contacto contigo y quién miente y sobre qué.

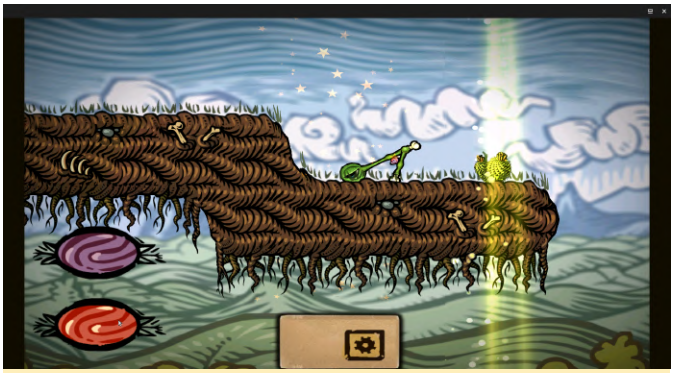


Figura 16 - Incredipede, coloridos gráficos orgánicos para un buen rompecabezas

Figura 17: Space Arena, construye tus propias naves y lucha contra otras

Incredipede es un buen juego de rompecabezas, tienes dos botones que hacen que tu Quozzle mueva sus músculos y piernas y necesite recolectar elementos y empujarlos hacia la luz que le sigue. Puede ser bastante difícil ya que su movimiento es un tanto extraño. En los modos más difíciles, puedes construir tu propio Quozzle para lograr tu objetivo.

Space Arena es un juego en el que construyes tu propia nave espacial, bueno, más bien las partes internas.

Compras un plan de distribución y lo llena con armas, reactores, escudos, sistema de propulsión, etc. luego lo envía contra un oponente. Si ganas, consigues experiencia y dinero que te permite comprar más cosas. No es muy original, pero puede resultar divertido durante un rato.

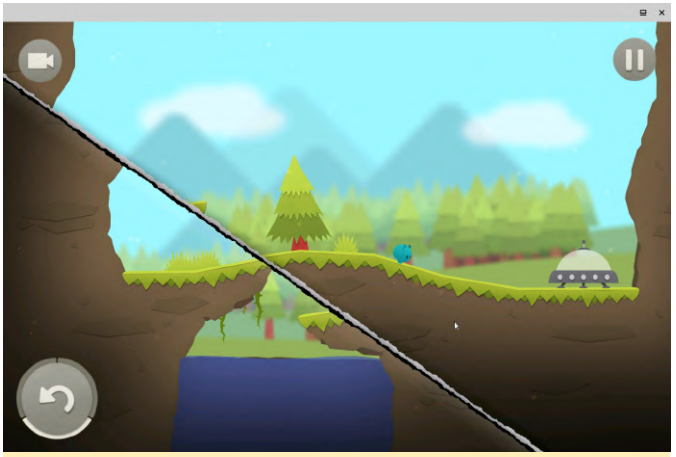


Figura 18: si no encaja, ¡arréglalo! Splitter Critter

En Splitter Critter puedes cortar el mundo en rodajas y luego mover las piezas para volver a unirlas como las necesites. Para esto, puede abrir nuevos caminos o no mover a tus pequeños sujetos a la cornisa, sino la cornisa hacía los sujetos.

Este juego es un juego de rompecabezas muy divertido, agradable y colorido. Necesitas encontrar el camino hacia tu nave espacial, para llegar al siguiente nivel, pero no solo necesitas crear un camino hacia tu nave, a menudo también tienes que evitar criaturas peligrosas o usar las habilidades de tus pequeños alienígenas a tu favor (saltando, por ejemplo). Es un juego muy divertido y lo recomiendo encarecidamente.

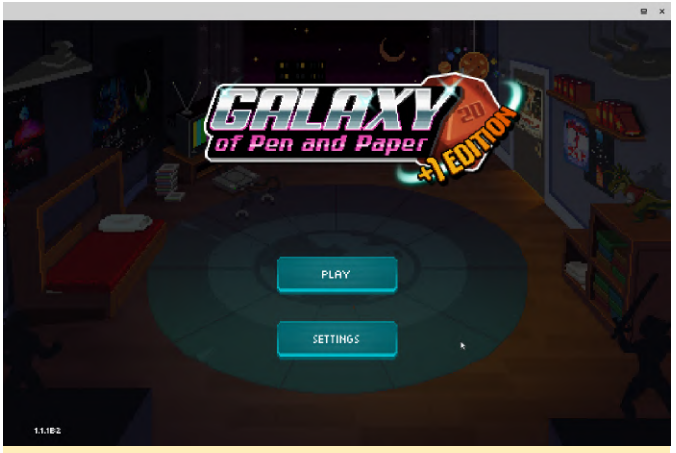


Figura 19 - Galaxy of Pen and Paper, un nuevo nivel de RPG

Figura 20 - El desguace es el lugar donde todo comienza

Figura 21 - ¡Aumenta el número de enemigos mientras puedas para conseguir una bonificación!

Galaxy of Pen and Paper es en realidad un nuevo tipo de juego de rol al mismo tiempo que juegas a un juego de rol en este juego de rol. ¿Confuso? En el juego empiezas creando un director de juego (G.M.) y algunos personajes, y éste contará la historia en la que juegas como cualquier juego de rol.

Este juego es gracioso y muy divertido de jugar. Ya lo había jugado en mi smartphone durante mis últimas vacaciones y es muy divertido. Puedes crear tus propias misiones seleccionando los monstruos con los que quieres luchar y con cuántos. Recibes tareas de los NPC y puedes viajar entre estrellas. Se burla de Star Wars, Star Trek y todo lo relacionado con los RPG y el espacio. Tuve algunos problemas gráficos con Anbox, pero nada serio, solo algunos fallos aquí y allá, pero el juego en general es muy divertido. También hay un juego llamado Knights of Pen and Paper que es más o menos lo mismo en un entorno de fantasía, no lo he probado, pero imagino que también funcionará.

Otro juego en 2D es Space Life, donde cazas monstruos, recoges objetos, compras armas y barcos.

Te diviertes un rato, pero no impresiona demasiado.

Te estarás preguntando, ¿esto significa que únicamente estámos limitado a juegos 2D como los anteriores?

¡Por supuesto que no! Ragnarok M ya es un juego en 3D, aunque es posible que no lo veas a primera vista, pero todos los personajes y la mayor parte del mundo están en 3D, puedes mover la cámara alrededor de tu personaje o volar sobre el mapa con una ballena voladora o algo así. Créeme, es autentico 3D.

Pero hay más: Raid: Shadow Legends es un impresionante juego 3D que funciona bien en Anbox. Tiene increíbles gráficos y cientos de personajes jugables, armas y actualizaciones, hay mucho por explorar.

Figura 22: Raid: Shadow Legends tiene gráficos muy buenos para ser un juego móvil con muchos efectos y personajes muy detallados.

Tiene una bonita historia de fondo que ayuda con los combates del juego. El juego también cuenta con muchos anuncios comerciales para compras en la aplicación, asi que probablemente no será muy apropiado para personas que se ven fácilmente influenciadas por este tipo de compras dentro de las aplicaciones. Existe otro juego que no he tenido tiempo de explorar con detalle, se llama Tales of the Wind y es un MMORPG de la serie Tales.

Este MMORPG de estilo anime es muy bueno, sólo con la pantalla de inicio ya quedas impresionado donde se observan personajes grandes y detallados. Lo mismo ocurre con la pantalla de creación de personajes. Ambas muestran los tipos de gráficos que te puedes esperar.

En el juego hay bastantes escenas de video dentro de "Gráficos del juego" que se ven bastante bien y son divertidas. Solo lo he jugado durante un par de minutos, ya que me temo que me enganchare demasiado llegado el momento.

Anbox parece funcionar bastante bien con grandes títulos o MMORPG, ya que probablemente están desarrollados teniendo en mente los emuladores para permitir usar el ratón y el teclado. Tales of Wind también te permite caminar con ASWD, que muestra como planearon de antemano este juego.

Figura 23 - Tales of Wind, un manifico MMORPG con estilo anime

Figura 24 - Pantalla de selección del personaje en Tales of the Wind

Figura 25: los gráficos del juego son muy coloridos y están llenos de detalles

En general, el software se creó teniendo en cuenta las CPU x86. Depende de SSE para algunas características que solo está disponible en sistemas x86. Sin embargo, dado que técnicamente no hay mucho detrás que pueda evitar que se ejecute en armhf o arm64, también debería ser posible ejecutarlo en otros ODROID.

Si revisas Debian Stretch y Buster, incluso encontrará paquetes de instalación fácilmente disponibles en los repositorios de Debian para armhf y arm64. Si profundizamos más, incluso encontraremos una imagen de Android_armhf y Android_arm64 en la página de descarga que te mostré anteriormente:

https://build.Anbox.io/Android-images/2017/08/04/Android_1_arm64.img
https://build.Anbox.io/Android-images/2017/06/12/Android_1_armhf.img

Incluso han sido probadas con armhf y arm64 en algún momento. Puedo decirte de antemando que, si fuera tan fácil, el artículo no hubiera estado relacionado con el ODROID-H2, sino como ejecutarlas en el XU4 o C2. Aun así, hay indicios que indican que podría funcionar.

Mientras investigaba encontré los siguientes enlaces:

https://github.com/Anbox/Anbox/issues/1214
https://skmp.dev/blog/Anbox-rpi4/

Esto indica que debería ser posible. He intentado seguir esta guía, pero no he logrado hacerlo funcionar (todavía). La imagen armhf del sitio de descargas de Anbox tiene algunos problemas y no funciona de este modo. skmp menciona en su guía que recompiló la imagen armhf, y yo también, pero es muy diferente a la que comenta. En su guía, indica que necesitas descargar aproximadamente unos 40 GB para el entorno de compilación de Android, y un total de aproximadamente 100 GB para compilar el software.

En su blog no explica cómo hace esto, solo te remite a otro enlace y dice que siguió esa guía, pero siguiendo el enlace no hay una guía real, sino solo otro enlace que debes seguir para descargar el entorno de compilación.

Hice esto y digamos que no se descargaron 40 GB, sino más de 130 GB, pero al final pude crear también una imagen armhf y es muy similar a la creada por skmp.

Pero cuando intenté usarla en lugar de una directamente de Anbox, no llegué muy lejos. Por lo que puedo comprobar, parece tener problemas con los gráficos y, para ser sincero, los gráficos de la GPU Mali son un poco especiales. Tampoco he logrago compilar aún una nueva imagen arm64.

También es algo complicado hacer que /dev/binder y /dev/ashmem se ejecuten en ODROID. Aunque el módulo DKMS debería funcionar, no funciona con cada versión de Kernel, así que terminé activando ambas por defecto en las imágenes normales de Kernel para ODROID, ya que son específicas para cada versión del Kernel. De esta forma, debería estar disponible en los Kernels más recientes de todas mis imágenes, incluso sin compilar los módulos DKMS.

Tal vez alguien tenga más suerte en hacer que Anbox funcione en ODROID armhf o arm64. Lo agradecería mucho.

Reflexiones finales

Anbox es un proyecto muy bueno, desearía que el soporte arm y arm64 fuera mejor y pudiéramos ejecutarlo en ODROID ya que nos abriría más posibilidades.

Anbox ofrece muchas opciones para trabajar, por ejemplo, ejecutar Anbox en modo de ventana única, lo que podría ayudar a ejecutar ciertas aplicaciones que no permiten abrir ventanas.

En general, estoy muy entusiasmado con este proyecto, ya que podría permitirnos ejecutar muchas más aplicaciones directamente bajo Linux en el futuro y con un poco de suerte incluso en dispositivos arm y arm64.

El Punto G: Tu Destino para Todas las Cuestiones Relacionadas con Juegos Android

November 1, 2019 By Dave Prochnow Juegos

Si eres como yo, te estarás rascando la cabeza en este momento por la reciente presentación de Play Pass de Google. Play Pass es un servicio de suscripción mensual que te permite descargar y jugar una amplia biblioteca de juegos y otras aplicaciones de productividad que están actualmente disponibles en Play Store de Google. Este anuncio no podría llegar en un momento más inoportuno. Estamos a solo dos meses del lanzamiento oficial del servicio de streaming de videojuegos a gran escala de Google, Stadia. Entonces, ¿qué nos brinda este servicio? ¿Por qué Google trata de cubrir preventivamente el período previo al estreno de Stadia al lanzar un servicio indudablemente inferior?

Un responsable del proyecto en Google dijo lo siguiente sobre Play Pass: "Los juegos son súper importantes ... [Play Pass] no está pensado para atraer a los jugadores más entusiastas". De acuerdo, Play Pass está dirigido a aquellos usuarios de Android que podrían no decantarse por el ecosistema Stadia. Es lo adecuado. Desafortunadamente, Play Pass todavía tiene varios fallos que te harán rascarte la cabeza.

Figura 1: Google Play Pass podría estar orientado al jugador "ocasional" - Imagen cortesía de Google Play Pass

Alimentado con más de 350 juegos y aplicaciones que actualmente están disponibles en Google Play Store, Play Pass costará 4.99$ al mes con un acceso completo, ilimitado y sin publicidad a toda la biblioteca. ¿Qué pasa si ya has comprado y has jugado a "Monument Valley", por ejemplo. ¿Qué ventajas te aporta Play Pass? Tal vez muy pocas o tal vez la oportunidad de poder jugar a juegos que nunca comprarías.

Los suscriptores deben tener cuidado, los desarrolladores de juegos usan un proceso de aplicación "opcional" que permite que un juego o aplicación sea incluido en la biblioteca Play Pass. El responsable de proyecto de Google no mencionó nada sobre qué sucederá con los juegos de Play Pass cuando un desarrollador abandone este servicio de suscripción. Además, tampoco dio detalles sobre los términos de los contratos asociados a los pagos que reciben los desarrolladores por las suscripciones a los juegos.

En el momento de su lanzamiento, Google ofrece una ingeniosa promoción de 1.99$ al mes durante el primer año de suscripción a Play Pass. Del mismo modo, hay un período de prueba gratuito de 10 días. Finalmente, Google permitirá que hasta cinco miembros de la familia compartan una única suscripción. Independientemente de cómo se desarrolle Play Pass, el peor enemigo de los juegos de Android podría ser el propio Google.

Pásate al lado oscuro

Era solo cuestión de tiempo que Google Play Store incluyese un tema oscuro. Tras el lanzamiento de Android 10 el mes pasado, las aplicaciones de Google han habilitado lenta, pero de forma progresiva un tema de modo oscuro. Al igual que la aplicación Google y Gmail, Play Store ahora tiene un tema oscuro. Supuestamente, este tema oscuro se activa con la versión 16.7.21 de Play Store. De modo que, si todavía no te has pasado al lado oscuro, busca la actualización actual de Play Store o si eres un usuario de Android 10, el cambio podría estar en un tema automático que ya está activado en la nueva configuración del sistema operativo.

Mario Kart Tour

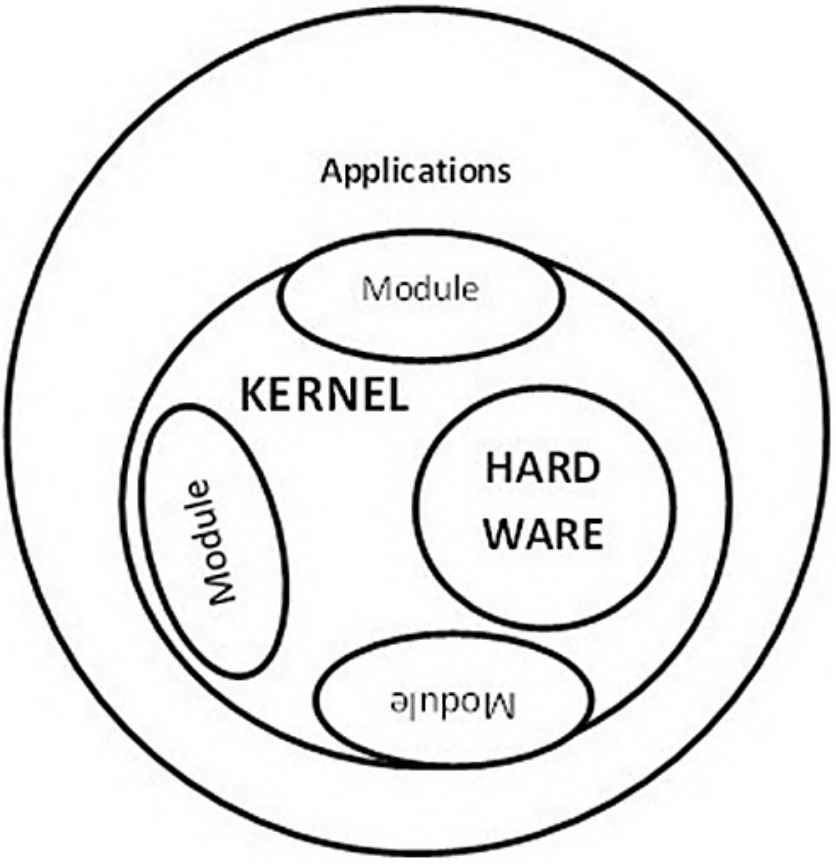
Finalmente, ya está aquí: el Mario Kart Tour de Nintendo. Este título tardío ya lo tenemos disponible como descarga gratuita. Jugarlo gratis, sin embargo, podría ser una cuestión diferente. Ten en cuenta que necesitarás una cuenta de Nintendo antes de poder jugar a este juego. Peor es aún, si deseas tener acceso completo a todas las carreras, trazados y karts del juego, tendrá que pagar una suscripción mensual de 4.99$. Nintendo llama a esta suscripción el "Gold Pass". Oro para Nintendo y NO para ti.

Figura 2: ¿Podría ser Mario Kart Tour el título de Mario que obligaría a los jugadores de Android a cambiar? Imagen cortesía de Nintendo

Al carecer de compatibilidad con mandos, Mario Kart Tour ofrece un rendimiento de juego bastante mediocre. Más aun por el hecho de que la interfaz de juego es únicamente vertical. Afortunadamente, los gráficos del juego son de primera categoría. De modo que, aunque conducir resulte casi imposible, seguro que se ve genial mientras estás inmerso en la carrera. Tienes más información sobre Mario Kart Tour en el video **https://www.youtube.com/watch?v=V-_s4oWV1cU**.

Modulos Kernel

November 1, 2019 By Andrew Ruggeri Linux



Si has detenido en los Foros de Hardkernel el tiempo suficiente, la expresión "módulo kernel" debería serte bastante familiar. Sin embargo, si eres nuevo en el mundo de Linux, los detalles sobre qué son exactamente los módulos de kernel podrías no tenerlos muy claros. El objetivo de este artículo no es solo que sepas que es exactamente un Módulo Kernel, sino también conocer cómo interactuar y compilar el tuyo propio.

¿Qué es un módulo kernel?

El kernel de Linux es monolítico o poco flexible, lo que significa que todo lo que necesita el sistema operativo forma parte del espacio del kernel. Esto tiene la ventaja de ser más rápido que otros diseños de kernel como un micro-kernel, pero se paga el precio de carecer de modulación y flexibilidad. Los módulos kernel están diseñados para ayudar a solucionar este problema de modulación. Para añadir funcionalidades al kernel, como un nuevo controlador o formato de sistema de archivos, el código con esa nueva funcionalidad en particular se compila en un módulo kernel y luego se carga en el kernel de Linux.

Ejemplo Hello World

Para el código de ejemplo y el resto del artículo, usare el nombre de archivo "examplemod.c"

```
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/module.h>

MODULE_DESCRIPTION("Example kernel module");
MODULE_AUTHOR("ODROID");

static int example_init(void)
{
      printk(KERN_ALERT "Hello World!
");
      return 0;
}

static void example_exit(void)
{
```

```
      printk(KERN_ALERT "Goodbye World
");
}

module_init(example_init);
module_exit(example_exit);
```

El código incluye un simple "hello world", ya que el objetivo de este artículo está centrado con la definición y la funcionalidad de los módulos kernel. Este ejemplo utiliza las funciones de registro del kernel para mostrar un simple saludo después de la inicialización y un adiós cuando es retirado el kernel. El código tiene dos partes que destacan y a las que hago referencia a continuación.

```
MODULE_DESCRIPTION("Example kernel module");
MODULE_AUTHOR("ODROID");
```

Estas macros ayudan a completar la información sobre el módulo, existen tipos de campos adicionales que se pueden usar como "MODULE_VERSION", "MODULE_LICENSE", etc. Más adelante veremos comandos sobre cómo ver esta información para cualquier módulo dado.

```
module_init(simple_init);
module_exit(simple_cleanup);
```

Las funciones module_init y module_exit registran las funciones simple_init y simple_cleanup que se han definido anteriormente. Estas llamadas registran qué funciones creadas por el usuario serán llamadas en la inicialización y en la salida del código. printk (), aunque a primera vista puede parecer similar o intercambiable con printf (), debemos pensar en ella de manera diferente. Como hemos mencionado anteriormente, printk es un mecanismo para el registro de mensajes del kernel y no para interactuar directamente o mostrar información al usuario. El nivel de registro, que varía de 0 a 7 (7 menos importante) se coloca antes de la cadena sin un comando, ya que forman un único parámetro que pasa a printk ().

Compilando

Después de escribir el módulo, hay un par de notas a tener en cuenta antes de simplemente llamar "gcc". Los módulos kernel son bastante delicados a la hora de compilarse, se deben cumplir las siguientes reglas:

- No se pueden incluir cabeceras que no sean del kernel
- El módulo no debe estar vinculado a ninguna librería
- Los flags del compilador utilizados para el módulo deben coincidir con los utilizados para el kernel.

Para simplificar las cosas, podemos usar el siguiente archivo MAKE:

```
obj-m += examplemod.o
```

```
all:
      make -C /lib/modules/$(shell uname -r)/build
M=$(PWD) modules

clean:
      make -C /lib/modules/$(shell uname -r)/build
M=$(PWD) clean
```

El archivo MAKE es bastante sencillo, la parte más importante es la siguiente línea:

```
make -C /lib/modules/$(shell uname -r)/build
M=$(PWD) modules
```

La ruta apunta a una ubicación donde están ubicados los Makefiles y las cabeceras del kernel. El Makefile simplemente llama al que se encuentra en esta ubicación para proporcionar la funcionalidad de cara a la compilación y también a la limpieza. Después de ejecutar el Makefile, deberíamos quedarnos con un archivo llamado examplemod.ko, este es nuestro módulo kernel compilado.

Comandos

Ahora que hemos creado nuestro Módulo Kernel, es hora de cargarlo en el kernel Linux. El siguiente comando es el que se usa para cargar un archivo * .ko. Cuando se carga el módulo, se llamará a la función fijada por module_init.

```
$ sudo insmod ./examplemod.ko
```

Si eliminamos un Módulo Kernel que ha sido cargado, llamará a la función fijada por module_exit. Para eliminar nuestro Módulo Kernel o cualquier Módulo que haya sido cargado, podemos usar el siguiente comando.

```
$ sudo rmmod examplemod
```

Aunque no es un comando específico del Módulo Kernel, el siguiente comando nos permite ver lo que ha sido registrado por el kernel. Si hemos ejecutado los dos comandos anteriores, deberíamos ver las dos declaraciones printk () cerca de la parte inferior (más reciente) del registro log.

```
$ dmesg
```

O para ver solo las últimas 2 líneas que se imprimieron..

```
$ dmesg | tail -2
```

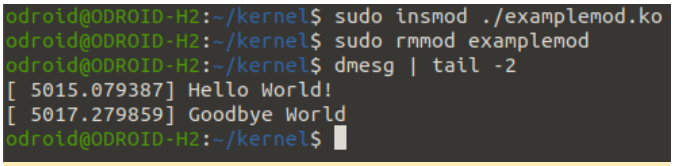


Figura 1: Cargando y eliminando el módulo kernel, luego imprimiendo el registro del kernel

Esto mostrará todos los módulos kernel cargados actualmente. Sin embargo, esta información es bastante limitada. Si queremos profundizar podemos usar el nombre de un módulo de la lista con el siguiente comando.

```
$ lsmod
```

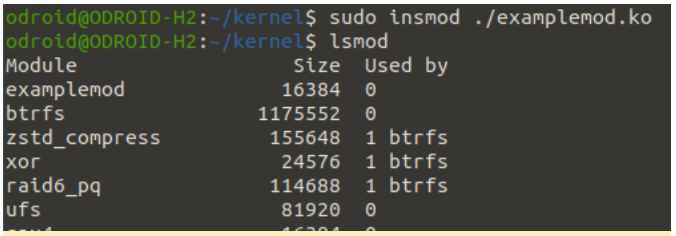


Figura 2: Impresión de lsmod que muestra el ejemplomod cargado

Esto mostrará una vista más detallada del módulo. Si miramos la información del módulo de ejemplo que hemos creado, veremos la información que fijamos al comienzo del código de nuestro módulo.

```
$ modinfo examplemod
```

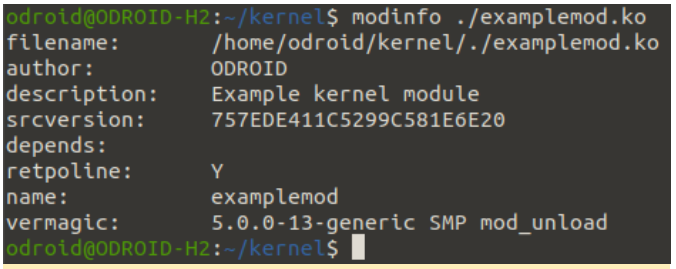


Figura 3: Impresión modinfo de la información para nuestro módulo kernel

Conclusión

Estos son solo los conceptos básicos para comprender qué es un Módulo Kernel y cómo crearlos e interactuar con ellos. En futuros artículos analizaremos los usos y aplicaciones más complejos de los Módulos de Kernel y cómo llegar a cierto nivel de interacción con ellos para que proporcionen una funcionalidad significativa.

ODROID-N2 UART Tasa de Baudios Personalizada para MIDI

November 1, 2019 By @tony.hong ODROID-N2, Mecaniqueo

Necesitaba el puerto UART de mi ODROID-N2 para trabajar con una tasa de baudios no estándar y así poder usar MIDI. Edité el código del driver UART para fijar la velocidad de baudios en 31250 cuando configuré la tasa de baudios en 38400, luego escribí el código de prueba usando wiringPi y medí la tasa de baudios.

Driver UART después de editarlo

```
linux/drivers/amlogic/uart/meson_uart.c

static void meson_uart_change_speed(struct
uart_port *port, unsigned long baud)
{
u32 val;
struct meson_uart_port *mup = to_meson_port(port);
struct platform_device *pdev =
to_platform_device(port->dev);

while (!(readl(port->membase + AML_UART_STATUS) &
AML_UART_TX_EMPTY))
cpu_relax();

#ifdef UART_TEST_DEBUG
if (port->line != 0)
baud = 115200;
#endif

// this part is added.
// trace_printk() is not neccesarry, it is just
for debugging.
trace_printk("Your baudrate: %ld
", baud);
if(baud == 38400)
{
baud = 31250;
trace_printk("Change to %ld
", baud);
}
```

Ref. de compilación del kernel de Linux: https://wiki.odroid.com/odroid-n2/softw ... ing_kernel

Cambios de la sección de instalación

- arch/arm64/boot/Image -> arch/arm64/boot/Image.gz

- arch/arm64/boot/dts/meson64_odroidn2.dtb -> arch/arm64/boot/dts/amlogic/meson64_odroidn2.dtb

Codigó de Prueba

```
#include
#include

int main(void)
{
int fd1, fd2;
fd1 = serialOpen("/dev/ttyS1", 38400);
fd2 = serialOpen("/dev/ttyS2", 38400);

serialPutchar(fd1, 0xAA); // 10101010
serialPutchar(fd1, 0xAA);

int count = 0;
while(1)
{
if(serialDataAvail(fd2))
{
printf("%c", serialGetchar(fd2));
count++;
if(count == 2)
{
break;
}
}
}

serialClose(fd1);
serialClose(fd2);
}
```

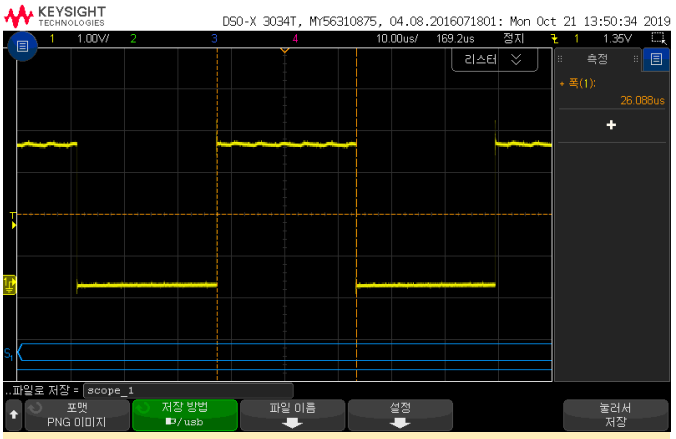


Figura 1 - Antes de editar: cuando se transmite 10101010, 1s/ancho de onda es la tasa de baudios, y 1s/26.088us es 38331 ~ = 38400

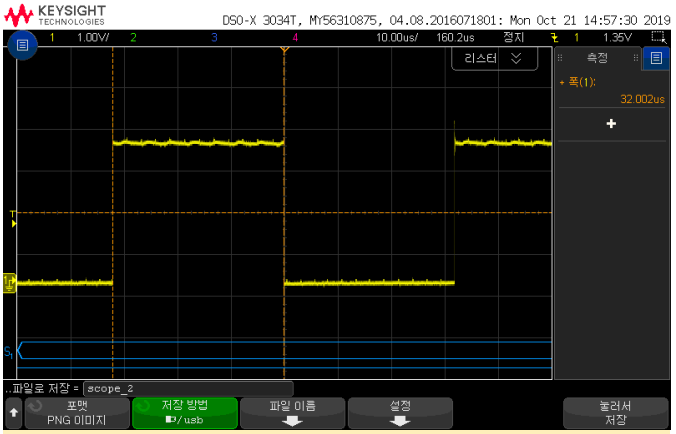


Figura 2 - Después de editar - 1s/32.002us es 31248 ~ = 31250

Depuración con ftrace

```
root@odroid:/home/odroid# stty -F /dev/ttyS0 38400
root@odroid:/home/odroid# cat
/sys/kernel/debug/tracing/trace
# tracer: function_graph
#
# CPU DURATION FUNCTION CALLS
# | | | | | | |
...
0) | uart_set_termios() {
0) | uart_change_speed.isra.2() {
0) | meson_uart_set_termios() {
0) 0.875 us | uart_get_baud_rate();
0) | /* Your baudrate: 38400 */
0) | /* Change to 31250 */
0) 0.250 us | uart_update_timeout();
```

Si tienes pensado realizar una compilación cruzada, la secuencia es la siguiente:

- Instalar paquetes requeridos
- Toolchain (6.3.1)
- Checkout(Linux tab)
- <- edit driver
- Compile(Basic tab)
- Installation(Linux tab)

Para comentarios, preguntas y sugerencias, visita el post original de los foros ODROID en https://forum.odroid.com/viewtopic.php?f=180&t=36540.

Juegos en ODROID-H2: Ejecutando Lakka en ODROID-H2

November 1, 2019 By @Synportack24 Juegos, Linux, ODROID-H2

El número del mes pasado de ODROID Magazine incluía el artículo "Lakka: Desarrollando la mejor consola de juegos ODROID-XU4 / XU4Q", disponible en **https://magazine.odroid.com/article/lakka-building-the-ultimate-odroid-xu4-xu4q-gaming-console/**. Este artículo estaba centrado en los juegos con un ODROID-XU4, sin embargo, existe un ODROID potencialmente más potente, el H2. Mientras que el ODROID-XU4 hace un trabajo increíble con los juegos retro, intentando ejecutar sistemas como Playstation o Dreamcast empieza a superar sus límites. Aquí es donde quería llegar, ¿Puede el ODROID-H2 seguir donde lo dejo ODROID-XU4?

Para ello, instalé una versión de 64 bits de Ubuntu Mate 19.04 en mi H2 y luego procedí a compilar e instalar Retroarch para la emulación (**libretro.com**). Mi H2 esta montado con 8 GB de RAM y un SSD Intel 660P, así que otras configuraciones con más o menos memoria o sin emmc pueden tener resultados diferentes.

Instalar Retroarch

Hay MUCHOS y diferentes programas y distribuciones completas de sistema operativo (como Lakka) centradas en la emulación. Elegí Retroarch para mis pruebas porque está bien documentado y se usa en muchos otros programas de emulación a modo de backend. La instalación es muy simple en ubuntu y la correspondiente documentación la puedes encontrar en: **https://docs.libretro.com/development/retroarch/compilation/ubuntu/**

El primer paso es agregar el ppa de libretro

```
# add-apt-repository ppa:libretro/testing
# apt-get update
```

Después de esto, tiene la opción de instalar retroarch como paquete, lo que no garantiza la versión más reciente posible y la compilación de git es muy simple, de modo que me decante por esta opción.

```
# apt-get update
# apt-get upgrade
# apt-get install git build-essential
# apt-get build-dep retroarch
$ git clone
https://github.com/libretro/RetroArch.git
retroarch
$ cd retroarch
$ git pull
$ ./configure
$ make clean
$ make -j4
```

Y tras completarse, puedes iniciar Retroarch con el siguiente comando

```
$ ./retroarch
```

Una vez que inicies Retroarch, selecciona "Online Updater" desde allí, querrás ejecutar/seleccionar

- Update Assets
- Update Core Info Files

A partir de ahí utilicé los siguientes emuladores

- Sega - Dreamcast/NAOMI (Flycast) https://docs.libretro.com/library/flycast/
- Sony - Playstation (Beetle PSX) https://docs.libretro.com/library/beetle_psx/
- Sony - Playstation 2 (Play!) : this was downloaded but the results were unplayable

Dreamcast

Me sorprendió mucho lo bien que Flycast manejaba cada juego que probaba. Mi selección de juegos estuba centrada en juegos que me permitiesen probar diferentes aspectos de la emulación y la jugabilidad. Seleccione Crazy Taxi 2 ya que es un juego pesado centrado en el 3D y debería forzar el H2. Del mismo modo, Marvel vs. Capcom 2, aunque no es 3D, es un juego bastante rápido donde se debería notar cualquier ralentización. Por último, elegí ChuChu Rocket, aunque es un juego menos conocido, nos permite probar más cuestiones sobre el emulador. No he notado ralentización, retraso o problemas gráficos peculiares. En el juego de Crazy Taxi 2 aparecían desgarros horizontales ocasionales que cubrían 2-3 fotogramas antes de desaparecer. Su campo de acción era bastante limitado y aparecía principalmente cuando la cámara se movía al entrar un pasajero en el taxi, ejecutando el juego en "tiempo de ejecución", esto no se llegaba a percibir.

Figura 1 - ChuChu Rocket

Figura 2 - Crazy Taxi 2

Figura 3 - Marvel vs Capcom

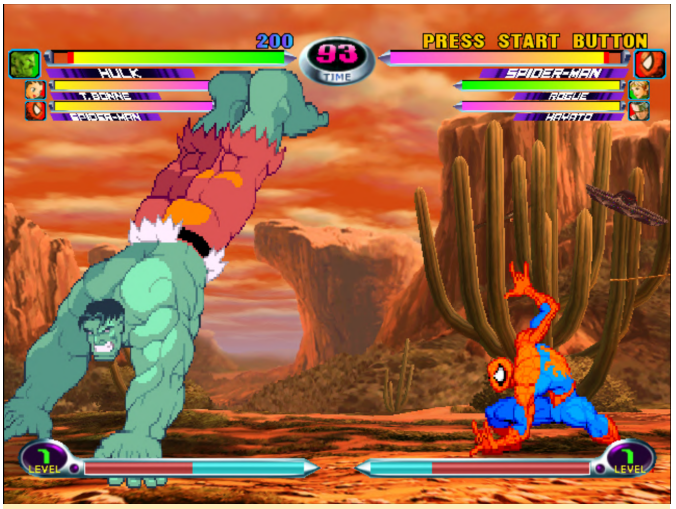


Figura 4 - Marvel vs Capcom

Figura 5 - Crash Team Racing

Playstation

A diferencia de Dreamcast, existen un par de emuladores diferentes que se pueden usar para emular Playstation. Al leer la descripción de cada uno, resulta fácil encontrar la mejor elección, beetle psx. La documentación de libretro (**docs.libretro.com/library/beetle_psx/**) nos dice lo siguiente: “Este emulador está indicado para personas que ejecutan RetroPie en sistemas de CPU x86 más potentes. Es preciso, lo mejor que se puede pedir cuando hablamos de un núcleo PSX para RetroArch. lr-beetle-psx no está disponible para sistemas con CPU ARM (como la Raspberry Pi) debido a al bajo rendimiento de las CPU ARM".

Después de instalarlo y configurarlo todo para el emulador de Playstation, observé que era mucho más "selectivo" a la hora de cargar y ejecutar una rom. Varios de los emuladores que había configurado para probar daban errores de carga/ejecución cuando intentaba cargarlos. Con los juegos que probé (Crash Team Racing, Dino Crisis 2 y Resident Evil 2), no tuve problemas de jugabilidad. Todo iba sobre ruedas, sin ralentizaciones ni desgarros. A mitad de la secuencia de video de la introducción de Dino Crisis 2, éste se detuvo. Sin embargo, cuando empezó el juego en sí no hubo problemas.

Figura 6 - Dino Crisis 2 Opening Cutscene

Figura 8 - Dino Crisis 2 Gameplay

Figura 9 - Resident Evil 2

Conclusión

En general, me sorprendió bastante lo bien que ODROID-H2 manejó algunos de los sistemas de juego más modernos. Si eres un fanático de los juegos de retro y buscas actualizar a algo con un poco más de "potencia", definitivamente vale la pena echarle un vistazo al ODROID-H2, ya que podría ser un reproductor multimedia y un dispositivo de juegos retro casi perfecto.