Recalbox • PUBG • Seguir Objetos • Mali QT5 • Juegos Linux • BASH

ODROID Magazine

Año Cinco
Num. #55
Jul 2018

10 Anniversary

HARDKERNEL CELEBRA SUS DIEZ AÑOS CON UN NUEVO E INCREIBLE DISPOSITIVO HECHO ESPECIALMENTE PARA TI

ODROID-GO

SERVIDOR GLUSTERFS:
USANDO EL ODROID-HC2 PARA APLICACIONES DISTRIBUIDAS MASIVAMENTE

REPETIDOR TOR
SUMERGIENDO TU ODROID EN LA WEB PROFUNDA

Conceptos Básicos de BASH: Introducción a BASH

July 1, 2018

Personaliza el prompt de BASH y el comportamiento de BASH

Juegos Linux: PC-Engine/TurboGrafx

July 1, 2018

La PC-Engine, o TurboGrafx-16 como es llamada en los Estados Unidos, fue la primera consola de 16 bits

Seguimiento de Objetos Usando oCam y ODROID-XU4: Una Sencilla Guía Paso a Paso sobre el Seguimiento de Objetos

July 1, 2018

He pensado que muchas personas estarían interesadas en una guía que fuera fácil de seguir sobre cómo usar una oCam y el ODROID-XU4 para seguir objetos usando OpenCV.

Kit de Juego ODROID-GO: Una Consola de Juegos Portátil para Celebrar el Décimo Aniversario de Hardkernel

July 1, 2018

Para celebrar el décimo aniversario de ODROID, presentamos el kit de juego ODROID-GO. Incluye una placa de aniversario especial y todos los componentes adicionales para montar tu propio kit de juego y ver el funcionamiento que hay detrás de dicho dispositivo. No sólo se trata de un divertido proyecto de ▶️

Qt5 Acelerado por GPU Mali: Funcionando en Ubuntu 18.04

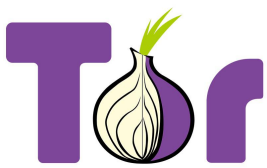
July 1, 2018

Ubuntu 18.04 Bionic viene con Qt 5.9.5 por defecto. Sin embargo, Canonical lo ha compilado sin tener en cuenta la GPU Mali de ARM, de modo que Qt5 no funciona en Ubuntu 18.04 para nada. Así que, tenemos que compilar Qt5 manualmente desde el código fuente. Esta es una guía ▶️

Player Unknown's Battlegrounds (PUBG) en el ODROID-XU4: Cómo Instalar y Jugar con un Teclado y un Ratón.

July 1, 2018

PlayerUnknown's Battlegrounds es un juego battle royale online multijugador desarrollado y publicado por PUBG Corporation



Convirtiendo tu ODROID en un Repetidor Tor

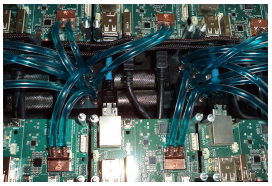
July 1, 2018

Tor es un software gratuito que permite acceder una red abierta muy útil con comunicaciones anónimas

Recalbox en el ODROID-XU4: Primeros Pasos

July 1, 2018

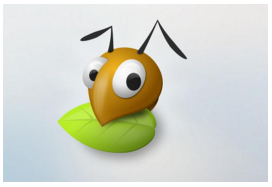
El ODROID-XU4 puede ejecutar una gran cantidad de sistemas operativos, incluyendo RetroPie, Ubuntu MATE y RecalBox



Refrigeración Líquida Parte 1 – Clúster

July 1, 2018

Tras encontrar algunos pequeños disipadores de calor de 15x15x5mm, decidí crear un clúster ODROID usando refrigeración por agua para reducir su temperatura y el ruido. Empecé con un solo ODROID-XU4 para ver si los pequeños disipadores de calor eran lo suficientemente potentes como para redirigir el calor lejos de la ▶



Servidor GlusterFS de 200TB que Utiliza el ODROID-HC2 para Aplicaciones Distribuidas Masivamente.

July 1, 2018

Con el transcurso de los años, he ido actualizado varias veces el almacenamiento de mi casa. Como muchos, empecé con un NAS de consumo. Introducir el ODROID-HC2 con 8 núcleos, 2 GB de RAM, ethernet Gbit y un puerto SATA, ofrece una muy buena base para aplicaciones de distribución masiva

Home Assistant: un Inteligente Proyecto de Bricolaje con la Iluminación

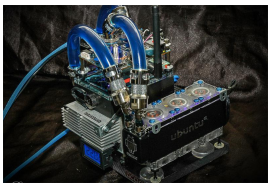
July 1, 2018

Desde que empecé a trabajar con Home Assistant y automatizar varias cosas en casa, siempre he querido encontrar la forma de controlar las luces. Eché un vistazo a las bombillas inteligentes, como las Philips Hue, pero son demasiado caras. Además, la mayoría de las soluciones usan protocolos registrados o servicios ▶

Primeros Pasos con OpenCL: Usando el ODROID-XU4

July 1, 2018

Aunque he testeado OpenGL ES con herramientas como glmark2-es2 y es2gears, así como también demos de WebGL en Chromium, no he llegado a probar OpenCL, ya que no estoy muy familiarizado con él, excepto que se usa con GPGPU (GPU de propósito general) para acelerar tareas como el procesamiento de ▶



Refrigeración líquida Parte 2 – Servidor

July 1, 2018

El proyecto de refrigeración más sofisticado hasta la fecha por nuestra comunidad. Montar un ODROID con refrigeración líquida me llevo aproximadamente 5 semanas, con un promedio de 12-18 horas diarias dedicadas al proyecto. Pesa unos 3 kg, con un coste total de alrededor de unos 950$, incluyendo la placa, los ▶

Conceptos Básicos de BASH: Introducción a BASH

July 1, 2018 By Erik Koennecke Tutoriales

¿Qué sucede cuando arrancamos e iniciamos sesión con variables y BASH, y cómo puedo personalizar el prompt BASH y el comportamiento de BASH? Tras analizar muchos comandos esenciales, es hora de hacer algo divertido. Cuando queremos subsistir a base de línea de comando, no está de más adaptarlos a nuestras necesidades. Para esto, primero tenemos que ver qué hace BASH cuando inicia sesión un usuario o cuando BASH se inicia al invocar un script.

BASH invocado como un shell de acceso interactivo

Este es el caso habitual cuando inicias sesión en el sistema mediante ssh, o tiene un terminal abierto sin una interfaz gráfica de usuario. Cuando se invoca BASH como un shell de acceso interactivo, primero lee y ejecuta los comandos de los siguientes archivos:

• /etc/profile, si existe el archivo. Después de leer este archivo, busca • ~/.bash_profile, • ~/.bash_login, y • ~/.profile, en este orden, lee y ejecuta los comandos desde el primero que existe y que es legible.

/etc/profile llama a /etc/bash.bashrc, de modo que hay uno más que añadir a la lista. Pero ~/.profile también arranca ~/.bashrc, así que este es el lugar donde tenemos que analizar todos los shells interactivos. Cuando un shell de acceso interactivo finaliza, o un shell de acceso no interactivo ejecuta el comando integrado exit, Bash lee y ejecuta comandos del archivo ~ /.bash_logout, si existe.

Invocado como un shell sin acceso interactivo

Cuando tienes una UI gráfica y abres una aplicación de terminal como mate-terminal o xterm, se ejecuta un shell sin inicio de sesión. Cuando se inicia un shell interactivo que no es un shell de inicio de sesión, BASH lee y ejecuta comandos desde ~/.bashrc, si ese archivo existe. La opción de archivo –rcfile obligará a BASH a leer y ejecutar comandos desde el archivo ~/.bashrc.

Invocado de forma no interactiva

Pensando en el nivel de detalle, esto es lo que sucede cuando ejecutas un script con BASH. Cuando BASH se inicia de forma no interactiva, para ejecutar un script shell, por ejemplo, busca la variable BASH_ENV en el entorno, expande su valor si aparece, y usa el valor expandido como el nombre de un archivo para leer y ejecutar.

En nuestro caso, el más importante es ~/.bashrc, ya que gestiona todos los shells interactivos de BASH. Si quieres que algo se ejecute solo una vez después de iniciar sesión, colócalo en /.profile. También podría usarse ~/bash_profile, pero no existe en Ubuntu.

El estándar para ambos serían añadir una función a ~/.bashrc, ya que quieres tenerlo siempre disponible:

```
### shows last installed packages from history
function apt-history(){

zcat -qf /var/log/apt/history.log* | grep -Po
'^Commandline: apt install (?!.*--
reinstall)K.*'
}
```

Para la otra opción, añade algo como lo siguiente a la ruta a través de ~/.profile, que sólo lo quieres una vez para iniciar sesión:

```
$ export PATH="$PATH:/some/addition"
```

Esto evita una ruta que tiene :/some/addition como añadido cada vez que se invoca BASH. Si quieres ver todo lo que sucede con BASH y estos archivos en conjunto, puedes intentar:

```
$ PS4='+ $BASH_SOURCE:$LINENO:'
BASH_XTRACEFD=7 bash -xlic "" 7>&2
```

No obstante, no te sorprendas por el volumen, y no te olvides de salir una vez más

Cambiar el prompt de BASH

Ahora es el momento de poner esto en práctica. Aunque el prompt estándar en color está bien, pero es mejor personalizando a nuestro gusto. Dirígete a bashrcgenerator.com y juega con las diferentes opciones.

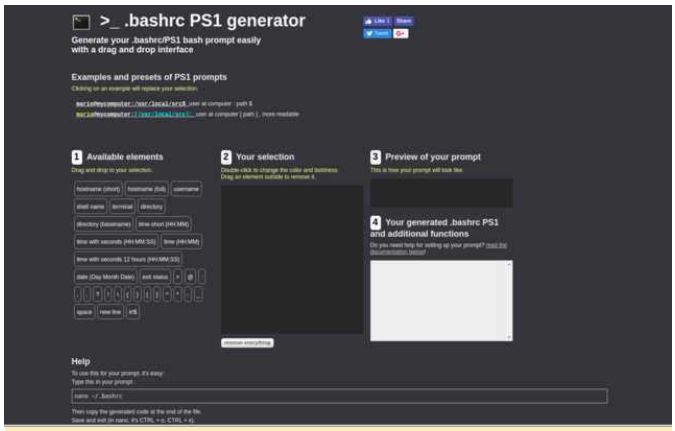


Figura 1 – Generador bashrc

Tras arrastrar los elementos interesantes al cajón 2, aparecerá un montón de códigos escape en el cajón 4. Así es como BASH puede interpretar las diferentes variables y colores. El principio es similar al Unicode o HTML codificado como ASCII plano, simplemente con colores y propiedades de pantalla.

Los terminales modernos como mate-terminal o xterm tienen capacidad para 256 colores. Para verlos, guarda el siguiente script como 256colors.sh, hazlo ejecutable con chmod a+x y ejecútalo:

```
#!/bin/bash

for fgbg in 38 48 ; do # Foreground /
Background
for color in {0..255} ; do # Colors
# Display the color
printf "e[${fgbg};5;%sm %3s e[0m" $color
$color
# Display 6 colors per lines
if [ $((($color + 1) % 6)) == 4 ] ; then
echo # New line
fi
done
echo # New line
done

exit 0
```

Esto es lo que tu terminal es capaz de hacer. Ahora, en el sitio bashrcgenerator.com, puedes hacer doble clic en los elementos del recuadro 2 para ver los diferentes colores y resoluciones. Los códigos escape se generan en el recuadro 4. Tras encontrar una versión que te guste, copia el código del recuadro 4 y pruébalo en tu ventana de BASH pegándolo y presionando INTRO. Si estás satisfecho con el

resultado, abre ~/ .bashrc con un editor de texto, busca la línea que empieza por PS1 = y reemplázala con la línea del recuadro 4, ¡pero sin la parte de exportación!

Personalmente, me gusta mantener un equilibrio entre tener algo corto, simple y discreto, y disponer de toda la información en el prompt, similar a un tablero de instrumentos:

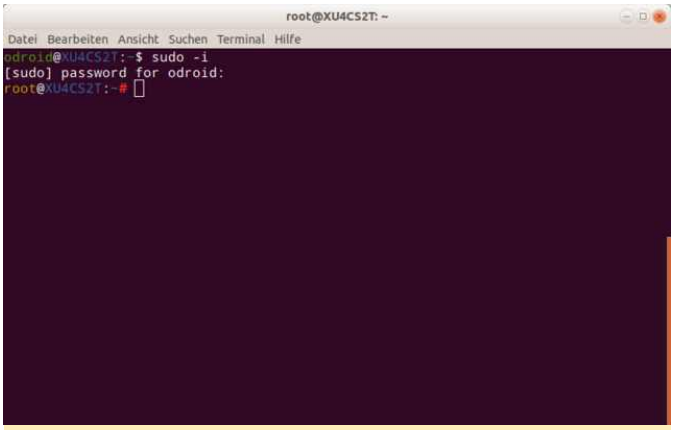


Figura 2 – Prompt de comandos con el ejemplo PS1

Como puede ver, el prompt de root es diferente al prompt de usuario para así recordarme que tengo privilegios de root y potencialmente podría llegar a destruir mi sistema por descuido If you also want to have a prompt like this, looks for the following line:

```
if [ "$color_prompt" = yes ]; then
```

Si también deseas tener un prompt como este, busca la siguiente línea:

```
if [ $(id -u) -eq 0 ];
then # you are root, make the prompt red
PS1='${debian_chroot:+($debian_chroot)}\
[e[00;33m\]u\[e[00m\]@\[e[00;34m\]h\[e[00m\]:\
[e[00;36m\]w\[e[00m\]e[01;31m#e[00m '
else
PS1='${debian_chroot:+($debian_chroot)}\
[e[00;32m\]u\[e[00m\]@\[e[00;34m\]h\[e[00m\]:\
[e[00;36m\]w\[e[00m\]$ '
fi
```

Reemplaza todo lo que hay antes de la expresión else sin sangría unas pocas líneas hacia abajo. Ahora con el nuevo y reluciente prompt, queremos ir un poco más allá. ¿Qué más podemos hacer para personalizar nuestra experiencia con BASH?

Funciones de BASH

Como he mencionado anteriormente, puedes colocar funciones en el archivo ~/.bashrc. El ejemplo apt-history es una función que muestra los últimos paquetes instalados o eliminados. ¡Tras haber cambiado ~/.bashrc, no olvides cerrar la sesión e iniciar sesión de nuevo para que tus cambios tengan efecto! Simplemente asigna un nombre a la función, empieza con la función nombrefunción () y coloca tu código entre paréntesis.

Otro ejemplo divertido sería ofrecer el tiempo actual por consola. Puedes obtener el parte meteorológico actual por terminal haciendo curl wttr.in/YourCity, con el prefijo de dos letras como fr.wttr.in/Paris recibirás el parte meteorológico de París en francés.

Para ver todas las opciones, escribe el siguiente comando:

$ curl wttr.in/:help.

Resulta un tanto molesto escribir siempre todas las opciones; con una función puedes omitir todo ello y simplemente escribir “wttr” para obtener el clima actual de tu ubicación, en el idioma correcto y con todas las opciones que quieras:

```
function wttr()
{
# Seoul is the default location
curl -H "Accept-Language: ${LANG%_*}"
wttr.in/"${1:-Seoul}"
}
```

En el caso de ODROID, podría ser interesante usar una función cputemp, puedes ejecutar cputemp y obtener la temperatura de la CPU con la siguiente función:

```
function cputemp()
{
#for XU4 usage, others may differ
cat
/sys/devices/virtual/thermal/thermal_zone0/tem
p
}
```

Esto te da la temperatura en °C con tres ceros al final. Para un resultado más bonito, sería más adecuado usar un script y así no sobrecargar ~/.bashrc. Puedes añadir todas tus funciones al final del archivo ~/.bashrc, una después de otra.

Alias de BASH

Puede que hayas observado con anterioridad que puedes usar “ll” para “ls -l” en Ubuntu. Para usar alias similares que permitan adaptar los comandos a tus necesidades, crea un nuevo archivo ~/.bash_aliases e introduce líneas con los comandos y las opciones que quieras usar:

```
$ alias ping='ping -c 5'
```

por ejemplo, detener el comando ping somehost.com después de hacer 5 pings, similar a la operación de ping en Windows. Otro ejemplo sería obtener un formulario legible cuando quieres ver la cantidad de espacio libre que hay en los discos:

Juegos Linux: PC-Engine/TurboGrafx

July 1, 2018 By Tobias Schaaf Juegos, Linux

Aunque aún no he terminado con mi serie de Sega Saturn, recientemente he estado ocupado con otro gran sistema del que me gustaría hablar un poco, probablemente haga una serie de él en algún momento en el futuro. En mi opinión, es un sistema que está bastante subestimado, que pone de relieve la falta de desarrolladores de terceros y de IPs conocidas para el sistema. No creo que este sistema sea muy respetado, es por ello que quisiera exponer mis propias reflexiones sobre mismo.

Figura 1 – PC-Engine: la más pequeña de las consolas domésticas más importantes

La PC-Engine, o TurboGrafx-16 como es llamada en los Estados Unidos, fue la primera consola de 16 bits y con ella, dio comienzo la era de las consolas de videojuegos de 16 bits. Esto sólo debería haber sido motivo suficiente para otorgarle un gran reconocimiento, pero aparentemente no fue así, al menos no para el mercado de los EE. UU. y la UE.

También se convirtió en la primera consola de 16 bits en incluir un componente adicional para CD, incluso antes que Sega CD para Sega Genesis/Mega Drive. Más tarde, en lugar de ofrecer este componente adicional, la unidad de CD y la consola se unieron en un sistema integrado conocido como PC-Engine Duo/TurboDuo. Una vez más, mi opinión es que el componente adicional de CD y más tarde el CD integrado representaban otra buena razón por la cual el sistema debería haber sido bastante más conocido. Más contenido y más música con calidad de CD para los juegos de consola de 16 bits, entonces ¿Qué es lo que no me gusta de todo esto?

Todavía, es, en mi opinión, una de las consolas más subestimadas en los EE. UU. y la UE, aunque su menor representación en el mercado de la UE es comprensible, considerando el hecho de que nunca fue lanzada oficialmente en la UE. La guerra seguía entre Nintendo y Sega, mientras que PC-Engine/TurboGrafx de NEC terminó perdiéndose en el campo de batalla, lo cual es una lástima, porque la consola tiene un gran potencial y tiene muchos y muy buenos títulos.

Hardware

La CPU del sistema era un procesador Hudson Soft HuC6280 de 8 bits con la posibilidad de cambiar entre 1.79 MHz y 7.16 MHz. Comparado con el procesador de 16 bits utilizado en la SNES con 3.58 MHz, éste era probablemente el punto débil de la consola PC-Engine, pero aun así tenía un gran potencial. Como usaba un chip gráfico de 16 bits, fue conocida como la primera consola de 16 bits.

La consola ofrecía una resolución máxima de 565×242 píxeles y una resolución vertical de 484 píxeles en modo entrelazado, mientras que la mayoría de los juegos todavía usaban una resolución de 256×239. En comparación, la SNES tenía una resolución entre 256×224 y 512×448, con la mayoría de los juegos funcionando a 256×224 píxeles, por lo que el PC-Engine tenía una resolución ligeramente mayor que el SNES en la mayoría de los casos. Aun así, los colores se limitaron a 512 (9 bits) comparado con la SNES de 15 bits (32768 colores). Si hablamos de potencia gráfica, la PC-Engine probablemente se acerque más a la Sega Genesis/Mega Drive que a la SNES, y de hecho supera a la Genesis en algunas áreas, pero en la mayoría de los casos la SNES es casi con toda seguridad mejor. El sistema PC-Engine fue originalmente diseñado para competir con NES y Famicom, pero al final terminó competiendo con SNES y Genesis.

El hardware de audio estaba integrado en la CPU en lugar de contar con un chip independiente como era el caso de Genesis y SNES. Tenía 6 canales cada uno, con una profundidad de 5 bits, pero podía combinar dos canales para reproducir muestras de 8 bits, 9 bits o 10 bits. Con la incorporación del complemento de CD-ROM, se añadió sonido CD-DA y un canal ADPCM al potencial de sonido ya existente en la PC-Engine. La memoria del PC-Engine estaba limitada, con 8K de RAM operativa y 64K de RAM de video. Esta última era bastante habitual en la mayoría de las consolas de la era de los 16 bits, aunque la cantidad de RAM funcional era bastante menor.

El componente adicional de CD llevaba su propia DRAM de 64K. Con la Tarjeta de sistema v3.00, el sistema recibía otros 192K de SRAM para trabajar. Más tarde, el PC-Engine Duo (TurboDuo en los EE. UU.) incorporaría un único chip SRAM de 256K, lo cual hizo que el rendimiento del sistema aumentara significativamente.

HuCards

PC-Engine no usaba cartuchos como la SNES o Genesis, sino algo llamado HuCards como medio de almacenamiento para los juegos. HuCards eran pequeñas tarjetas, de tamaño similar a las tarjetas de crédito, aunque un tanto más gruesas, que se introducían en una ranura situada en la parte frontal de la máquina y quedaban bloqueadas cuando se accionaba el interruptor de encendido.

Figura 2 – HuCards para la TurboGrafx-16 (PC-Engine en Japón)

Estas HuCards (también llamadas Turbo Chips) eran parecidas a las “My Cards” (también llamadas Sega Card) utilizadas en los sistemas Sega SG-1000/3000 y Sega Mark III/Master. La HuCard más grande tenía un tamaño de sólo 20 Mbit en comparación con los 48 Mbit utilizados en los cartuchos de SNES más grandes o los 32 Mbit para la Genesis. Esto significa que los datos almacenados en estas tarjetas eran más pequeños que los de otros sistemas.

Los HuCards tenían la ventaja de ser más económicas de producir que los cartuchos, lo cual hacía que los

juegos para PC-Engine/TurboGrafx fueran un poco más baratos que los de la SNES o la Genesis. El inconveniente no era la falta de espacio adiconal como ocurría con algunos cartuchos de SNES. Aunque el hardware de PC-Engine y TurboGrafx-16 eran prácticamente idéntico, los HuCards japoneses para PC-Engine no funcionan en el TurboGrafx-16 y viceversa, lo que llevaba a pensar erróneamente que se trataban de sistemas diferentes. De hecho, el tema estaba simplemente en que se cambiaron dos pines de las HuCards para PC-Engine y TurboGrafx-16 con el objeto de implementar una protección a nivel de regiones.

Para contrarrestar esto, podías modificar el hardware de tu TurboGrafx-16 o PC-Engine y con un interruptor, seleccionar entre modo JP y el modo US, o podías usar una tarjeta convertidora que cambiaba la distribución de los pines. Los emuladores, por supuesto, no tienen estas limitaciones, por lo tanto, los emuladores de PC-Engine funcionan perfectamente con cualquier tipo de ROM, ya sea para PC-Engine o TurboGrafx-16.

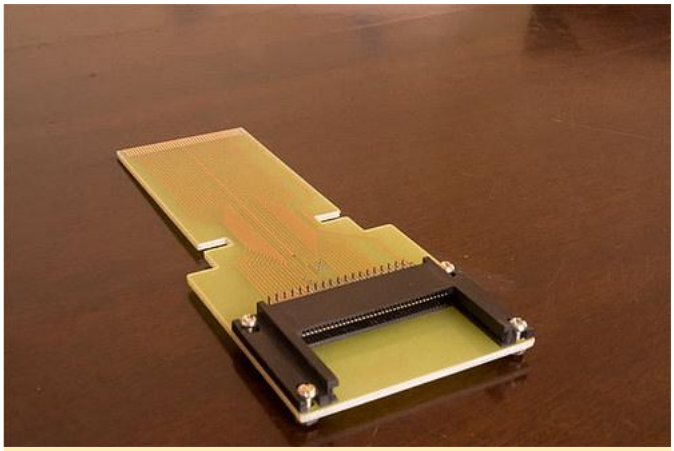


Figura 3 – Tarjeta de conversión para PC-Engine/TurboGrafx-16, que te permitirá ejecutar juegos japoneses en consolas de EE. UU.

Componente adicional de CD-ROM

De forma similar a Sega Genesis/Mega Drive, PC-Engine recibió un componente de CD-ROM posteriormente a lo largo de su vida útil. De hecho, fue la primera consola en tener este componente. El PC-Engine CD-ROM² System fue lanzado en diciembre de 1988, tres años completos antes del lanzamiento de Sega CD/Mega CD en diciembre de 1991. Este componente aumentó en gran medida el potencial del PC-Engine, añadiendo más RAM y una mejor calidad de sonido para el sistema. El componente de CD-ROM requería una tarjeta de sistema que permitiera a PC-Engine acceder a la unidad de CD. Esta tarjeta se actualizó con el tiempo y en 1991 se lanzó la tarjeta Super System (tarjeta de sistema v3.00), ampliando la memoria RAM del sistema a 256 K.

Más tarde, se lanzaron PC-Engine Duo y TurboDuo, combinando el componente de CD-ROM y la consola en un dispositivo que incluía memoria extra sin tener que usar la tarjeta Super System. Los juegos en CD para el sistema no presentaban la limitación de la región, lo cual te permitía ejecutar tus juegos japoneses en la consola de EE. UU. y viceversa.

Figura 4 – Consola TurboDuo One que ya incluye el componente de CD-ROM para TurboGrafx-16

Otras versiones de la consola

El PC-Engine tenía bastantes variantes en cuanto al estilo. Por ejemplo, estaba el PC-Engine Shuttle, una versión más barata con forma de nave espacial, que estaba dirigida a niños más pequeños, pero que no podía usar el componente de CD-ROM.

Otra versión fue PC-Engine LT, una PC-Engine en formato portátil que incluía un monitor y altavoces. La consola actuaba como un mando, pero aun así necesitaba un adaptador de corriente para que funcionase. Esto hizo que la consola fuera lo suficientemente buena como para llevártela a la casa de un amigo, incluso si éste en particular no tenía un televisor en su habitación.

Figura 5 – PC-Engine Shuttle dirigida a niños

Figura 6 – PC-Engine LT con monitor y altavoz integrados

De las muchas versiones, PC-Engine GT o TurboExpress en EE. UU., es probablemente la más interesante. Es una versión portátil de PC-Engine / TurboGrafx-16 que te permite jugar a tus juegos HuCard sobre la marcha, similar a la Sega Game Gear, pero con todo el potencial de PC-Engine. Era bastante avanzada para su época, con pantalla LCD retroiluminada, un complemento para ver la televisión e incluso tenía la posibilidad de vincularse para poder jugar con otros jugadores (por ejemplo, Bomberman '93), pero tenía un consumo muy elevado de pilas, duraba tan sólo 3 horas con 6 pilas AA.

Figura 7 – La versión portátil de PC-Engine: PC-Engine GT/TurboExpress

Hay una pieza de hardware más que debemos mencionar cuando hablamos de la PC-Engine/TurboGrafx-16 – el PC-Engine SuperGrafx. Se trata de un producto fallido destinado a mejorar el potencial del PC-Engine, mientras que al mismo tiempo era compatible con todas las características del PC-Engine. Estaba destinado a mejorar el rendimiento de los gráficos, el sonido y la RAM, y aunque se lanzó al mercado, al final, solo suponía una leve mejora con respecto a la PC-Engine original. El sistema multiplicaba por cuatro la cantidad de RAM operativa (32K en lugar de 8K) y tenía una GPU independiente para mejorar los gráficos, lo que permitía disponer de dos capas de fondo con desplazamiento independientes.

Lamentablemente, fue un completo fracaso ya que los HuCards para este dispositivo eran muy caros (hasta 110$). En total, solo se crearon seis juegos para el sistema, y solo uno de ellos también era compatible con la PC-Engine.

Figura 8 – PC-Engine SuperGrafx: un fracaso comercial

Juegos

Cuando hablamos del éxito de una plataforma de juegos, nos solemos centrar en los propios juegos. La biblioteca de PC-Engine es razonablemente grande, con aproximadamente 650 títulos disponibles, aunque si revisamos los lanzamientos de EE. UU., no son tantos. Un total de 138 juegos fueron lanzados en el mercado estadounidense, lo cual no es nada si lo comparamos con el número de lanzamientos para la SNES o incluso para la Sega Genesis.

También carece de muchos IPs muy conocidos para el sistema. Aun así, existen algunos juegos bastante singulares para el sistema tanto en CD como en HuCards. Algunas de las series conocidas que salieron incluyen Castlevania Rondo Of Blood (Akumajou Drácula X-Chi no Rinne), R-Type, Splatterhouse, Street Fighter II, Bonks, Ys I-IV, Bomberman, Gradius, Galaga, Raiden , Outrun, Parodius, Cotton, Wonder Boy entre otros.

También hay un monto de juegos muy buenos, aunque menos conocidos, que fueron lanzados exclusivamente para PC-Engine/TurboGrafx-16 como Soldier Blade, Super Star Soldier, Blazing Lazers, Cadash, Alien Crush, que vale la pena probar en el sistema. Al ser una consola principalmente japonesa, encontrarás una gran cantidad de juegos para el mercado japonés, incluida una gran biblioteca de más de cien juegos shoot-em-up. Si eres fanático de este tipo de juegos, definitivamente esta consola es para ti.

Aun así, tiene juegos de todos los géneros. En algún momento quizás continúe este artículo con una serie sobre los juegos de PC-Engine que más me gustan, similar a la serie de Sega Saturn que empecé hace un tiempo.

PC-Engine en el ODROID

Al igual que ocurre con la SNES o la Sega Genesis, la PC-Engine no tiene problemas para ejecutarse en las placas ARM, funciona perfectamente en cualquier ODROID. Incluso los pocos juegos de SuperGrafx se ejecutan sin problemas. Si quieres, puedes aplicar sombreadores al sistema para darle una sensación más retro.

Gracias al proyecto RetroArch, con sus núcleos librero, estos emuladores admiten el formato .chd, que es un formato comprimido para imágenes. Con esto, puedes comprimir imágenes de CD de la PC-Engine CD y ahorrar espacio sin perder rendimiento o datos. Recomiendo esta opción, ya que te ayudará a mantener tu colección organizada con un tamaño bastante reducido.

Debido a que el PC-Engine solo usa un mando con dos botones, todos los mandos deberían ser compatible con el emulador del sistema. Los dos botones adicionales que normalmente tienen los mandos funcionan actualmente como botones “turbo” para los juegos de PC-Engine. Esto es muy útil para los muchos shooters que existen para el sistema.

Reflexiones finales

El PC-Engine es un sistema excelente pero subestimado, en mi opinión. Viendo que entro en el mercado varios años antes que la SNES y la Genesis, y que al mismo tiempo ofrecía posibilidades similares, resulta aún más difícil de entender su falta de popularidad en los EE. UU. (fue bastante popular en Japón). La incorporación de la unidad de CD, años antes de que hiciera lo mismo cualquier competidor, fue una magnífica jugada y debería haber sido una razón más para darle un voto de confianza a esta consola que ofrece grandes títulos y un impresionante hardware.

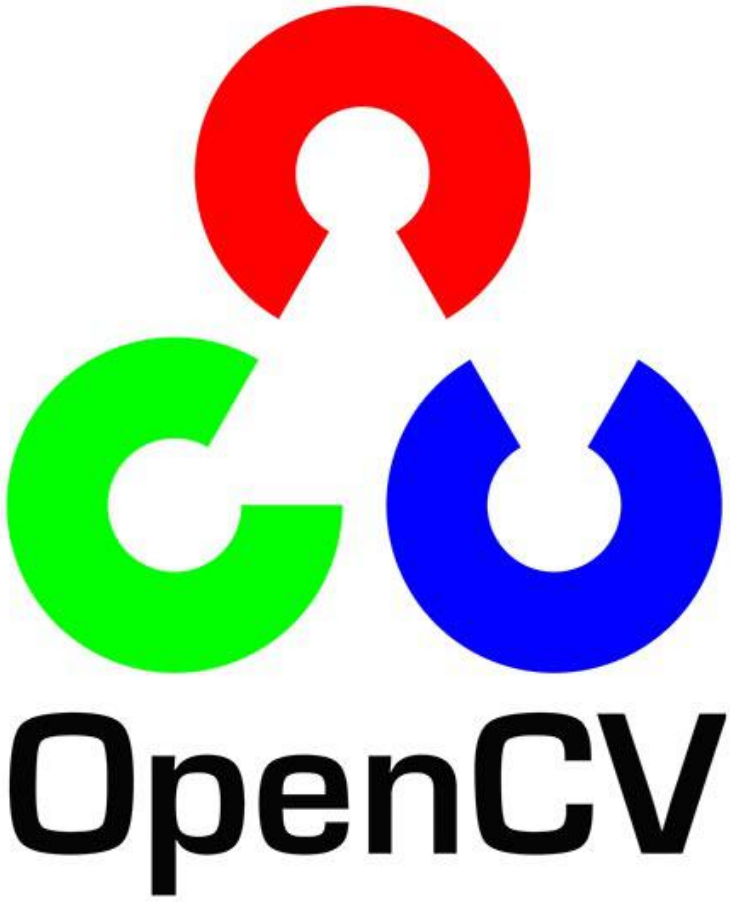
Aun así, creo que cometieron algunos errores. La consola original solo tenía un puerto para un mando, necesitaba un accesorio para poder expandirse a cinco puertos, lo cual fue un gran defecto en mi opinión. La consola base no era tan estupenda si querías jugar con amigos, porque necesitabas un complemento y más mandos. ¡Intenta explicarle esto

a tu madre por aquel entonces! Esta podría haber sido una razón por la cual no llego a ser tan popular en los Estados Unidos.

Gracias a la política de Nintendo de su día, que impedía a los desarrolladores de terceros realizar desarrollos para otras plataformas, solo hay unos cuantos IPs conocidos para este sistema que supusieron una barrera en el mercado estadounidense, tal y como ocurrió con Genesis y otros sistemas. Aun así, me gusta bastante la PC-Engine y probablemente jugaré a bastantes juegos cuando tenga la oportunidad. Animo a todos los que les gusta la SNES, la Genesis y otras consolas de cuarta generación que prueben la PC-Engine por su cuenta.

Seguimiento de Objetos Usando oCam y ODROID-XU4: Una Sencilla Guía Paso a Paso sobre el Seguimiento de Objetos

July 1, 2018 By DongHyun Yoo ODROID-XU4, Tutoriales



He pensado que muchas personas estarían interesadas en una guía que fuera fácil de seguir sobre cómo usar una oCam y el ODROID-XU4 para seguir objetos usando OpenCV. Esta guía te guiará por los diferentes pasos para crear y ejecutar una aplicación de seguimiento de objetos. La capacidad de rastrear un objeto específico sobre múltiples fotogramas es una tecnología clave en aplicaciones tales como la vigilancia automática o la robótica. El siguiente código de ejemplo rastreará un objeto utilizando las propiedades del color de la imagen.

Figura 1 – Ejemplo de Pantalla con seguimiento de objetos

Configuración

Para empezar, necesitarás los siguientes elementos, todos disponibles en la tienda Hardkernel:

- ODROID-XU4
- Módulo de memoria, ya sea eMMC o tarjeta micro SD, con Ubuntu instalado.
- oCam

Además, necesitaras instalar los siguientes paquetes de software, los cuales de pueden instalar utilizando la aplicación Synaptic Package Manager:

- gcc
- wget
- OpenCV

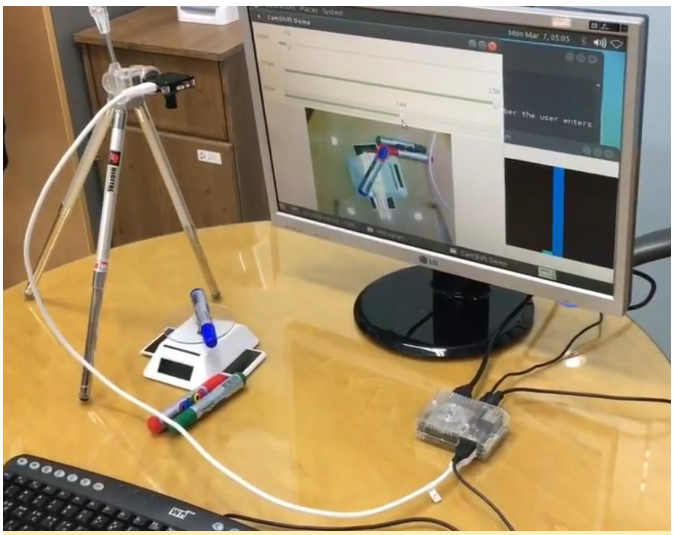


Figura 2 – Típica configuración del set de prueba de seguimiento de objetos

Para preparar tu sistema, abre una ventana de terminal e introduce los siguientes comandos:

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

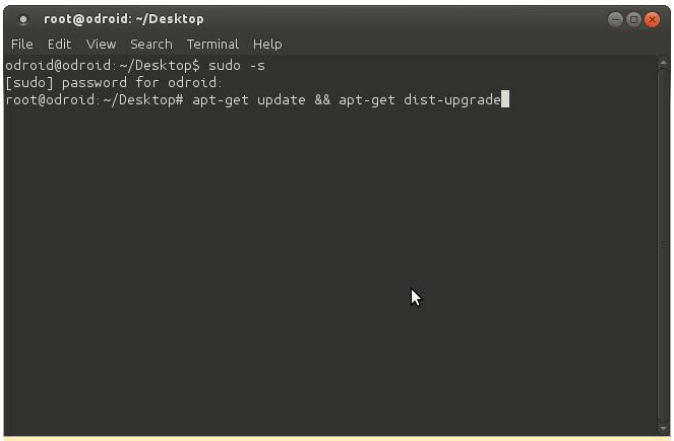


Figura 3 – Pantalla de entrada de comandos en ODROID

El primer comando actualizará la lista de paquetes e instalará la actualización más reciente de la distribución, si está disponible. El segundo comando reiniciará el ODROID. Tras actualizar la lista de paquetes y la distribución, instala OpenCV introduciendo el siguiente comando:

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

Desde el 2 de marzo de 2016, la última versión de OpenCV es la 2.4.9.

Compilación

Nuestro ejemplo está basado en el algoritmo Camshift (Continuous Adaptive Mean Shift), que es un tipo de algoritmo Meanshift y que se utiliza para rastrear objetos. Puedes encontrar más información sobre estos algoritmos en **http://bit.ly/1pPduzS**. Nuestro ejemplo está basado en el algoritmo Camshift (Continuous Adaptive Mean Shift), que es un tipo de algoritmo Meanshift y que se utiliza para rastrear objetos. Puedes encontrar más información sobre estos algoritmos en http://bit.ly/1pPduzS. En nuestro código, usaremos la función cvCamShift () de la librería OpenCV para habilitar el algoritmo camshift. El siguiente texto proporciona más información sobre la función cvCamShift ():

```
RotatedRect CamShift(InputArray probImage,
Rect& window, TermCriteria criteria)

Parameters:
probImage – Back projection of the object
histogram. See calcBackProject().
window – Initial search window.
criteria – Stop criteria for the underlying
meanShift().

Returns:
Result rectangle
```

Para descargar el archivo fuente camshiftdemo, usa el siguiente comando o descarga el archivo desde tu navegador web visitando **http://bit.ly/21ykrRF**:

```
$ wget
https://raw.githubusercontent.com/
Itseez/opencv/2.4/samples/
cpp/camshiftdemo.cpp
```

Ahora estamos listos para compilar camshiftdemo.cpp usando el siguiente comando:

```
$ g++ camshiftdemo.cpp -o demo
-O2 -lopencv_core -lopencv_imgproc
-lopencv_highgui -lopencv_video
```

Aquí tienes los significados de las opciones del compilador:

– o demo crea un archivo binario ejecutable llamado "demo" – O2 especifica un nivel de optimización de 2. Para obtener más información sobre la configuración de la optimización de g++, consulta http://bit.ly/1OOnopO. – l vincula una librería externa, usamos esto para vincular cuatro librerías: openvc_core, open_cv_imgproc, opencv_highgui, and opencv_video.

Ejecutando la aplicación

Una vez que la oCam esté conectada al ODROID-XU4, estamos listos para poner en marcha la demo del seguimiento de objetos usando el siguiente comando:

```
$ ./demo
```

La ventana de la Demo de CamShift tiene tres secciones: un panel de barra de control, un panel con la imagen de la cámara y un panel histograma. Usando los 2 controles deslizantes superiores, Vmax y Vmin, puede controlar el rango de valores del color. El control deslizante inferior, Smin, controla el nivel de saturación. Estos controles deslizantes ayudan a limitar el área de la imagen dentro de la cual vas a rastrear un objeto específico. Consulta información detallada sobre el tono, la saturación y el modelo de la gama de colores en http://bit.ly/1L6R7zM. Puede iniciar el seguimiento de objetos cliqueando y arrastrando la parte de la imagen de la cámara que deseas rastrear con el ratón. La Figura 4 muestra la aplicación ejecutándose mientras se ve un área seleccionada de una botella de zumo.

Figura 4 – Arrastra una parte de la imagen para iniciar el seguimiento

La ventana del histograma muestra los componentes de color dentro del área de la imagen seleccionada sobre el objeto que se está rastreando. Puede activar y desactivar la ventana del histograma presionando la tecla "h". También puedes cambiar el modo de vista normal para volver a la vista de proyección presionando la tecla "b". Las Figuras 6 y 7 muestran los diferentes modos de vista. Puedes encontrar más detalles sobre la proyección inversa en la página de OpenCV en http://bit.ly/1Rqc1MH.

Figura 5 – El área objetivo que se está rastreando y el histograma con los componentes de color

Figura 6 – Modo de vista de proyección inversa

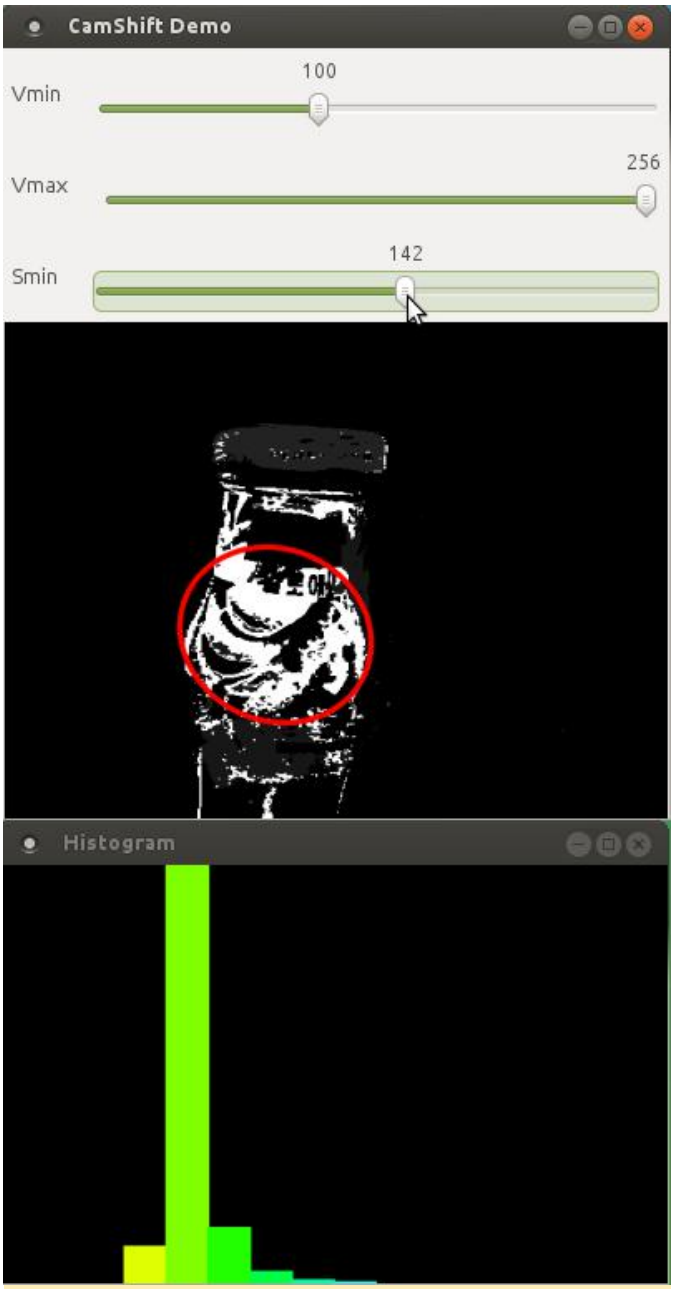


Figura 7 – Control de la región interesada en la que se rastrea el objeto

Para borrar la selección, presione la tecla “c”. Puede empezar a seguir un nuevo objeto seleccionando otra área del mismo modo que antes. Echa un vistazo al video disponible en **http://bit.ly/21zZllS**. Muestra una vista en vivo de la demo de seguimiento de objetos tratada en esta guía usando oCam y ODROID-XU4.

Kit de Juego ODROID-GO: Una Consola de Juegos Portátil para Celebrar el Décimo Aniversario de Hardkernel

July 1, 2018 By Justin Lee Juegos, Development, ODROID-GO

Hardkernel se fundó en 2008 y ODROID (Open-Droid) tiene 10 años. Cuando diseñamos el dispositivo, pensamos en tres lemas básicos:

- De los desarrolladores, Por los desarrolladores, Para los desarrolladores
- Dispositivos divertidos e interesantes para los desarrolladores
- Placa de desarrollo de bolsillo. (¡Para llevar!)

Figura 1 – El dispositivo de desarrollo ODROID original de 2008

Para celebrar el décimo aniversario de ODROID, presentamos el kit de juego ODROID-GO. Incluye una placa de aniversario especial y todos los componentes adicionales para montar tu propio kit de juego y ver el funcionamiento que hay detrás de dicho dispositivo. No sólo se trata de un divertido proyecto de ensamblaje, sino también de una herramienta educativa que te permite conocer todo el

hardware y el software que se ha utilizado en el desarrollo de dicho dispositivo.

Figura 2 – La forma que tiene es muy similar a la de nuestra placa SHOW2, aunque cuenta con una placa joypad aparte

El pequeño y económico rendimiento de Arduino MCU ESP32 era muy bueno para ejecutar emuladores NES, GBC y SMS de forma asombrosa, pero el estilo de sándwich no era tan bueno para estar jugando un par de horas. La PCB apilada era un inconveniente y no podía sujetarla durante mucho tiempo. Tampoco podíamos introducirla en nuestro bolsillo trasero. De modo que, tuvimos que abandonar el primer diseño y construir desde cero un molde de plástico con una forma más elegante y más cómoda.

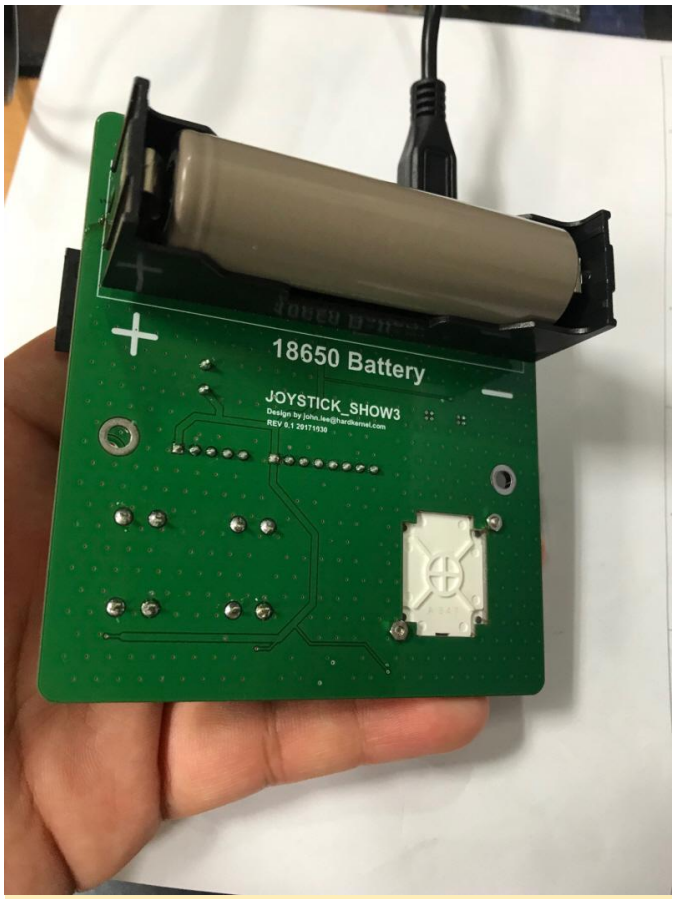


Figura 3 – Se podría instalar una batería de Li-Ion 18650 en la parte trasera

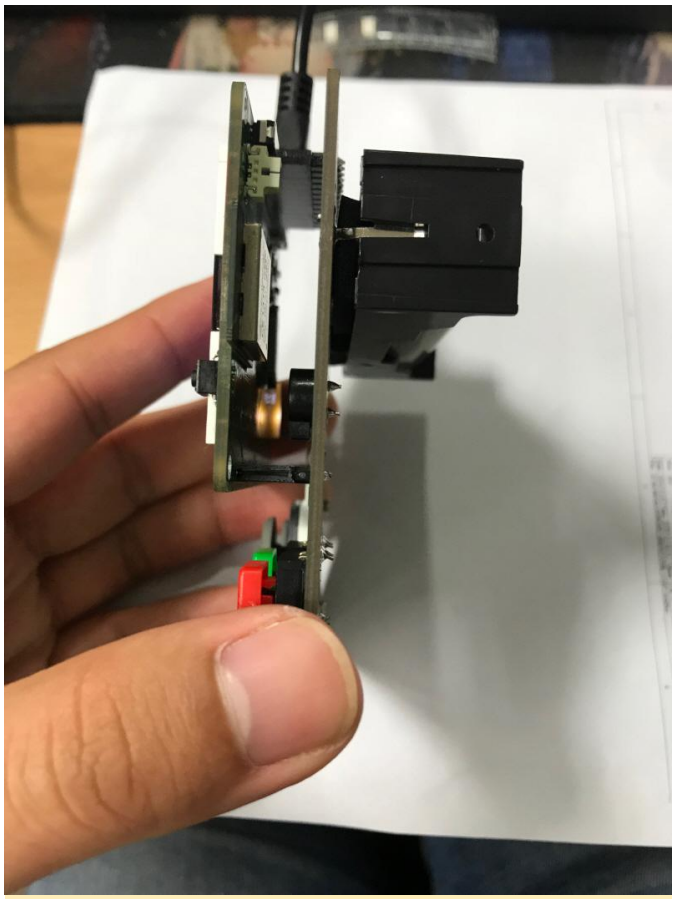


Figura 4 – Parecía un sándwich, de modo que el grosor era demasiado elevado.

Finalmente, llegamos al diseño actual. Lo llamamos ODROID-GO. Ahora podemos introducir esta bonita placa de desarrollo en nuestro bolsillo trasero y llévanosla a todas partes.

Figura 5 – ODROID-GO conmemora el décimo aniversario de Hardkernel

Ensamblar y Aprender

Diviértete construyendo tu propio kit de juego portátil mientras aprendes las funciones internas de cada componente y su finalidad. Conoce cómo cada botón está conectado a un conmutador de la PCB, qué materiales se utilizan y cómo montarlos para dar forma a un panel de control con botones para poder jugar. Aprende a conectar la alimentación, los altavoces y cómo descargar e instalar un sistema operativo. Descubre por qué ciertas piezas están hechas de unos materiales específicos y por qué necesitas ciertos conectores. Como el dispositivo es trasparente, todos los componentes internos y todas las luces son visibles. Una vez que hayas ensamblado el ODROID-GO, podrás descargar e instalar juegos. ¡Disfruta del dispositivo de juego que has creado!

Componentes incluidos

- A. 1x Placa ODROID-GO
- B. 1x Carcasa frontal
- C. 1x Carcasa trasera
- D. 1x Módulo LCD de 2.4 pulgadas
- E. 1x Botón de goma con 4 posiciones
- F. 1x Botón de goma con 2 posiciones
- G. 2x Botones de goma con 2 posiciones
- H. 1x Altavoz 8Ohm 0.5W
- I. 1x Cabezal macho de 10 pines
- J. 10x Tornillos
- K. 1x Marco para LCD
- L. 1x Set de botones
- M. 1x Cable Micro USB
- N. 1x Batería de Li-ion de 1200mAh

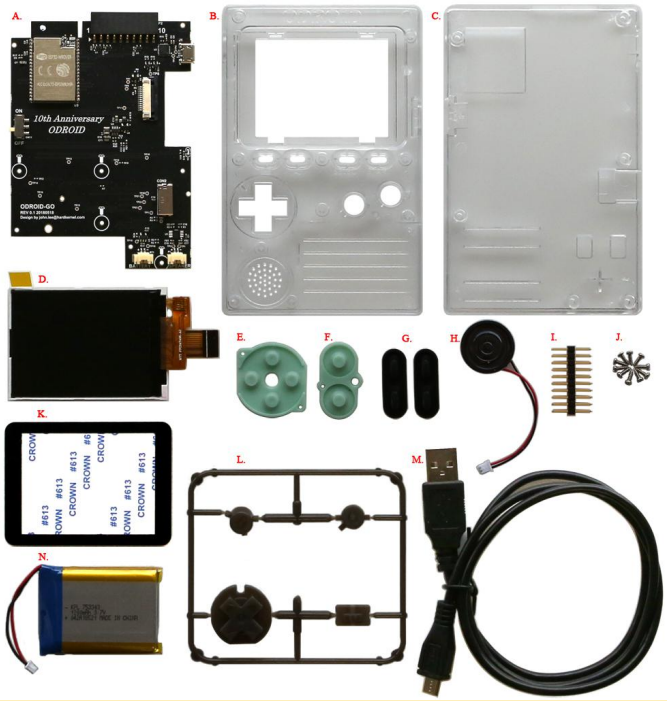


Figura 6 – Componentes incluidos en el kit ODROID-GO

Puede encontrar más información sobre el ensamblaje en https://wiki.odroid.com/odroid_go/go_assembling.

Listo para jugar

Crea una tarjeta microSD con tus propias colecciones de juegos. Los detalles los puedes encontrar en: https://wiki.odroid.com/odroid_go/emulator/make_sd_for_importing_roms. El emulador soporta juegos de:

- Game Boy
- Game Boy Color
- Game Gear
- Nintendo Entertainment System
- Sega Master System

Figura 7 – Vista frontal de ODROID-GO

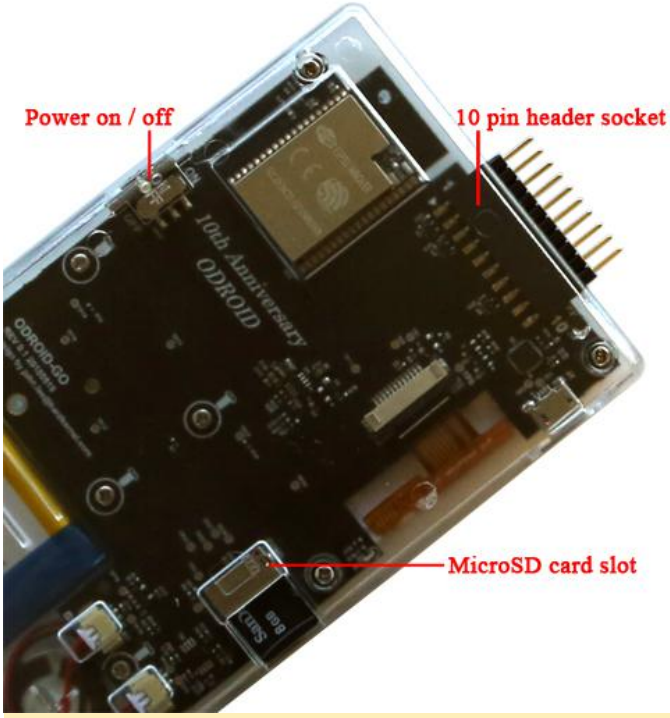


Figura 8 – Vista posterior de ODROID-GO

Especificaciones

- MCU hecha a medida ESP32-WROVER (memoria flash de 16MiB)
- CPU y RAM 80MHz – 240MHz (Ajustable), 4MB PSRAM
- Pantalla LCD TFT de 2.4 pulgadas y resolución 320×240 (interfaz SPI)
- Batería Li-Polymer 3.7V/1200mAh, hasta 10 horas de juego continuo
- Altavoz 0.5W/8Ω Mono
- Ranura para tarjeta mSD 20Mhz Interfaz SPI
- Puerto de expansión 10Pin: I2C, GPIO, IRQ a 3.3Volt
- Botones de Menú, Volumen, Selección, Inicio, A, B y Pad de dirección
- Puerto micro USB para cargar la batería (500mA) y para la comunicación de datos USB-UART
- Tamaño 76mm x 121mm x 16mm (ensamblado)

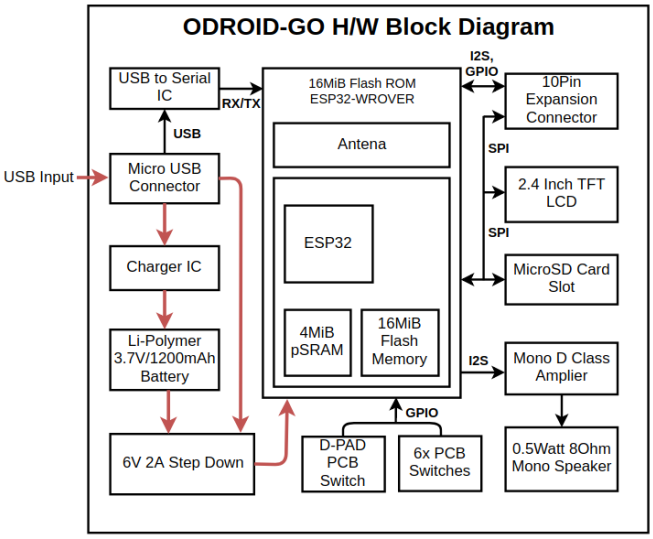


Figura 9 – Diagrama por bloques del ODROID-GO

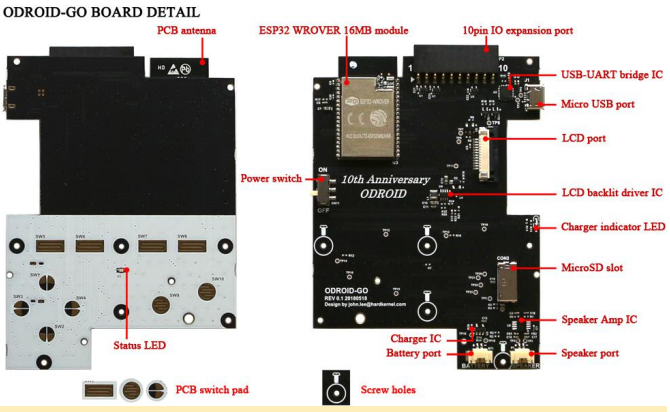


Figura 10 – PCB ODROID-GO que muestra los detalles de la placa

Para ver un video del ODROID-GO en acción, visita https://youtu.be/1kQ79ytZKJA.

Arduino Coding Camp con ODROID-GO

Los siguientes artículos de Coding Camp se presentarán de forma individual en los próximos meses en ODROID Magazine, para que los programadores principiantes puedan iniciarse fácilmente en el desarrollo utilizando el ODROID-GO como herramienta de aprendizaje.

Día 1: Primeros pasos con Arduino

Descargar e instalar librerías y ejemplos específicos de Arduino IDE y ODROID-GO..

Día 2: Mostrar "Hello, ODROID-GO" en la pantalla LCD

Vamos a aprender cómo mostrar una cadena de texto, cambiar los colores y modificar el tamaño de la fuente.

Día 3: Controlar el LED

Aprenderemos cómo controlar el LED azul de la parte frontal de ODROID-GO: retocar el LED con simple GPIO on/off, así como controlar el brillo con PWM.

Día 4: Leer el estado de los 12 botones en ODROID-GO

Aprenderemos cómo leer el estado del pin GPIO.

Día 5: Leer el voltaje de la batería integrada en el ODROIOD-GO

Aprenderemos cómo acceder a la entrada ADC para medir el voltaje.

Día 6: Generar sonido desde el altavoz ODROID-GO

Aprenderemos cómo usar la salida DAC para generar sonido.

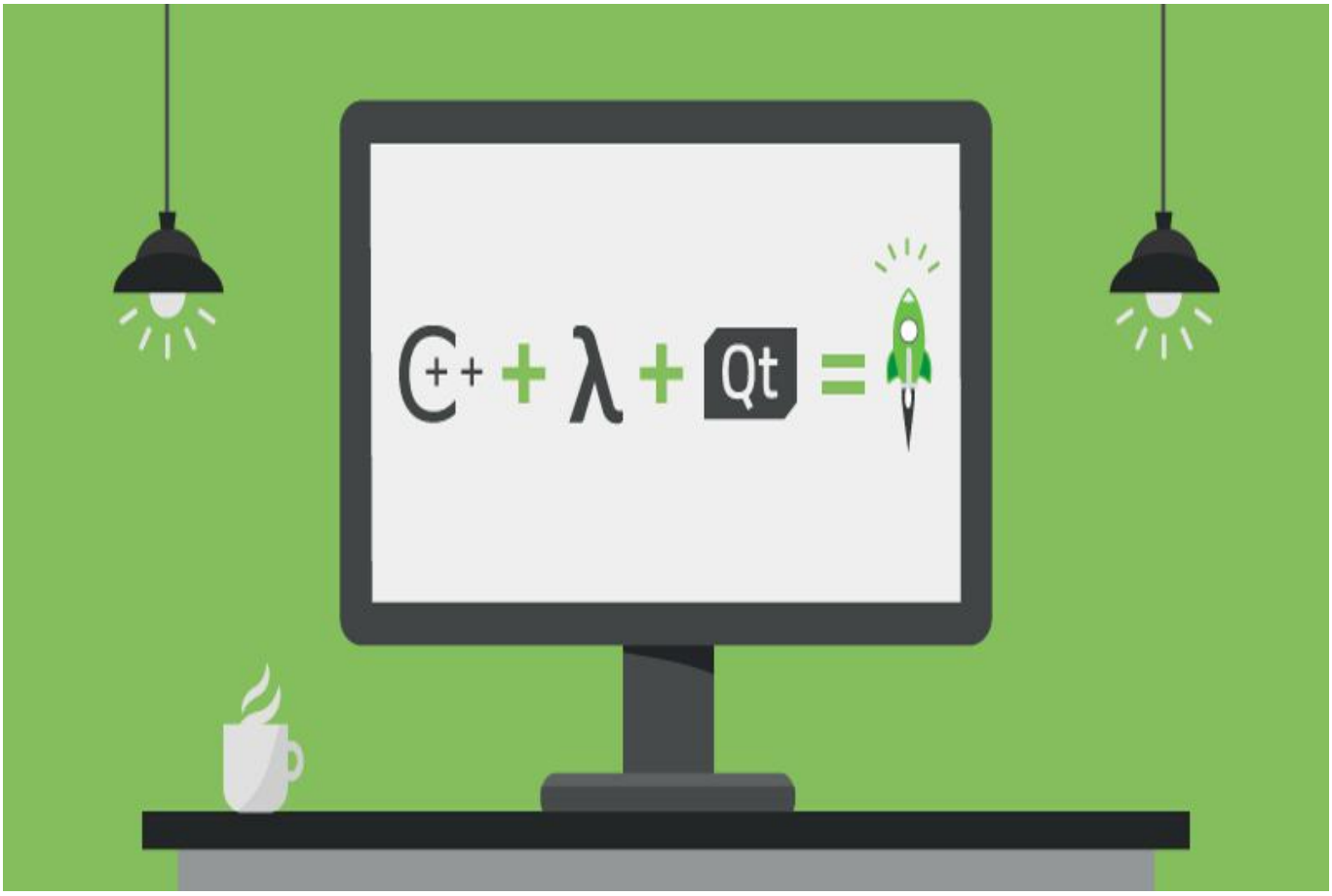
Día 7: Jugar a tu propio juego de Tetris

Aprenderemos cómo hacer un juego con un código de ejemplo del juego Tetris.

Día 8: Añadir otra pantalla LCD

Qt5 Acelerado por GPU Mali: Funcionando en Ubuntu 18.04

July 1, 2018 By Justin Lee ODROID-XU4, Tutoriales



Ubuntu 18.04 Bionic viene con Qt 5.9.5 por defecto. Sin embargo, Canonical lo ha compilado sin tener en cuenta la GPU Mali de ARM, de modo que Qt5 no funciona en Ubuntu 18.04 para nada. Así que, tenemos que compilar Qt5 manualmente desde el código fuente. Esta es una guía de compilación un tanto chapucera, que he probado en la última imagen de SO Ubuntu 18.04 Bionic para ODROID-XU4.

Instalación

```
$ sudo apt update && sudo apt upgrade && sudo apt dist-upgrade
$ sudo apt build-dep qt5-default
$ apt source qtbase5-dev
$ cd qtbase-opensource-src-5.9.5+dfsg
```

A continuación, cambia la línea 86 del archivo src/platformsupport/eglconvenience/qxlibeglintegration.cpp:

```
if (vendor && strstr(vendor, "Vivante")) {
```

por:

```
if (vendor && (strstr(vendor, "Vivante") || strstr(vendor, "ARM"))) {
```

Hay un segundo archivo que debemos editar para evitar un error de compilación. Me tire varias horas buscando esta simple solución en http://code.qt.io/cgit/qt/qtbase.git/commit/?h=dev&id=9a640e7bc67b0a1ff5c61c63703b669e6f24521e. Edita el archivo src/plugins/platforms/eglfs/deviceintegration/eglfs_kms_egldevice/qeglfskmsegldevice.cpp y cambia la línea 77:

```
-EGLNativeDisplayType QEglFSKmsEglDevice::nativeDisplay() const
```

por:

```
void *QEglFSKmsEglDevice::nativeDisplay() const
```

Además, modifica el cuerpo de la función:

```
return reinterpret_cast(m_devInt-
>eglDevice());
```

por:

```
return m_devInt->eglDevice();
```

Crea dos enlaces simbólicos para que se detecte correctamente OpenGL-ES. Esta solución probablemente estará incluida en la próxima actualización:

```
$ sudo rm /usr/lib/arm-linux-
gnueabihf/libGLESv2.so.2.0.0
$ sudo rm /usr/lib/arm-linux-
gnueabihf/libEGL.so.1.0.0
$ sudo ln -s /usr/lib/arm-linux-
gnueabihf/mali-egl/libmali.so /usr/lib/arm-
linux-gnueabihf/libGLESv2.so.2.0.0
$ sudo ln -s /usr/lib/arm-linux-
gnueabihf/mali-egl/libmali.so /usr/lib/arm-
linux-gnueabihf/libEGL.so.1.0.0
```

Despues, compila Qt5:

```
$ sudo dpkg-buildpackage -b
```

Me encontré con este error cuando lo ejecuté en una sesión SSH:

```
Project ERROR: QtDBus is enabled but session
bus is not available. Please check the
installation.
```

Cuando compilé Qt desde el terminal de escritorio Mate en lugar de hacerlo por ssh remoto, la compilación no me dio problemas.

Tras 2~3 horas de compilación, el “paquete Debian” falló debido a la ausencia de una clave PGP. No obstante, todas las librerías Qt5 con ejemplos se compilaron correctamente y pueden instalarse con:

```
$ sudo make install
```

La muestra de Qt-OpenGL ahora funciona perfectamente

Figura 1 – Muestra de Qt-OpenGl

Probaremos la estabilidad y la funcionalidad durante un par de semanas. Si no hay problemas críticos, Hardkernel lo lanzará oficialmente. Mientras tanto, tienes total libertad para publicar tus ideas en el hilo del foro https://forum.odroid.com/viewtopic.php?f=95&t=31070.

Aplicaciones de ejemplo Qt4 Open Source Computer Vision Library (OpenCV) es una librería software de aprendizaje automático y reconocimiento de imágenes por ordenador de código abierto. El entorno de trabajo Qt5 se usa con OpenCV para visualizar el procesamiento de imágenes y también como una interfaz de usuario interactiva.

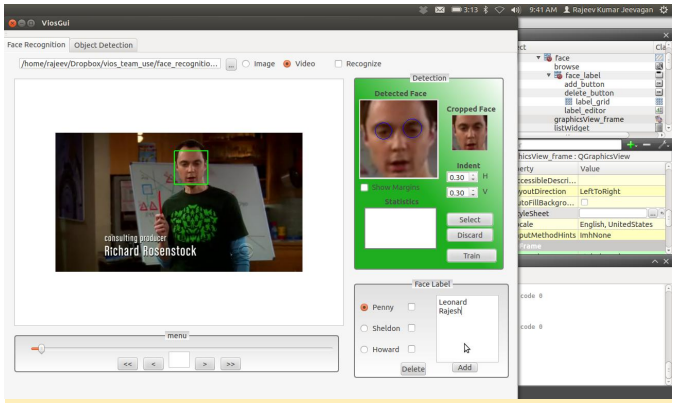


Figura 2 – Detección del rostro con OpenCV

Calligra es un completo set de 8 aplicaciones para oficina, gráficos y necesidades de gestión, que incluyen procesador de textos, presentaciones, hoja de cálculo y mucho más:

```
$ sudo apt install calligra-libs
```

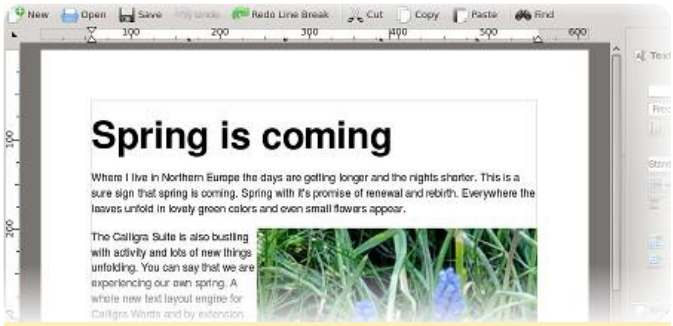


Figura 3 – Procesador de textos

Figura 04 – Hoja de cálculo

Calibre es un potente administrador de libros electrónicos muy fácil de usar:

```
$ sudo apt install calibre
```

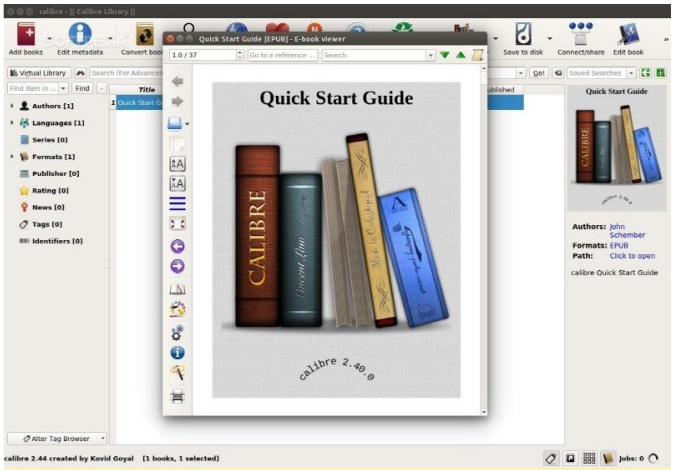


Figura 5 – Pantalla de inicio del calibre

Stellarium es un planetario de código abierto gratuito para tu ordenador. Muestra un cielo realista en 3D, al igual que el que se ves a simple vista, con prismáticos o un telescopio:

Player Unknown's Battlegrounds (PUBG) en el ODROID-XU4: Cómo Instalar y Jugar con un Teclado y un Ratón.

July 1, 2018 By Justin Lee Juegos, ODROID-XU4

PlayerUnknown's Battlegrounds es un juego battle royale online multijugador desarrollado y publicado por PUBG Corporation, una filiar de la editorial Bluehole. La última persona o equipo que queda en pie gana la partida. Gracias a Unreal Engine 4, se lanzó una versión móvil para dispositivos Android el 9 de febrero de 2018. Esta guía muestra con detalle la configuración que debemos tener en el sistema operativo Android sobre ODROID-XU4, incluye cómo jugar con un teclado y un ratón. Todo para ser los reyes del escenario y hacerse con la victoria.

En primer lugar, vamos a revisar la compatibilidad del hardware con PUBG en Android. La empresa responsable de PUBG MOBILE ha sido muy inteligente al utilizar Unreal Engine 4 (UE4) para todas las versiones de sus juegos, pueden aplicar el mismo contenido a todas las plataformas y garantizar la misma experiencia de juego en cualquier dispositivo.

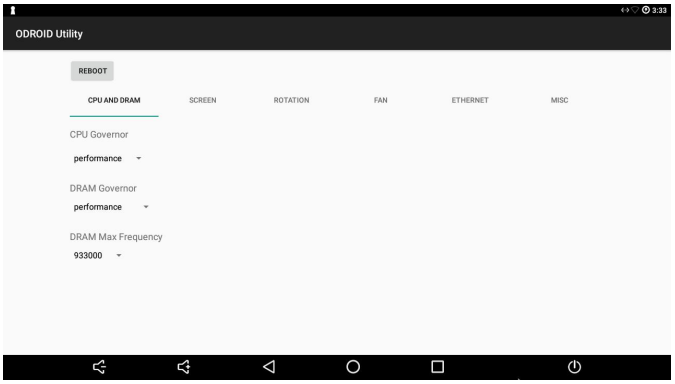
Los requisitos mínimos de los juegos vienes impuestos por el UE4. Este tipo de juegos solo se puede compilar con un perfil que soporte estas características:

- Tener una GPU con soporte OpenGL ES 3.1 o superior
- Sistema con al menos 2 GB de RAM
- Android 5.1.1
- Almacenamiento mínimo libre de 2 GB
- Necesita un emulador de ratón o puntero para poder seleccionar los menús

El Android 7.1 (versión LineageOS) del ODROID-XU4 cumple los requisitos. La GPU en ODROID-XU4 admite OpenGL-ES 3.1, cuenta con dos 2 GB de RAM apilados en la CPU, el almacenamiento es ampliable y los puertos USB son lo suficientemente buenos como para poder conectar un ratón.

Una vez que haya instalado con éxito Android 7.1 Nougat LineageOS-14.1 (**https://goo.gl/fUKur6**) en tu ODROID-XU4, puedes instalar fácilmente PUBG MOBILE desde la tienda Google Play.

Antes de jugar a PUBG, debes ajustar el rendimiento de Android para lograr una velocidad de rendereizado más fluida con la aplicación ODROID Utility. Configura el regulador de CPU y DRAM en modo “Performance” y aumenta ligeramente la velocidad de la DRAM (de 866Mhz a 933Mhz), luego reinicia.

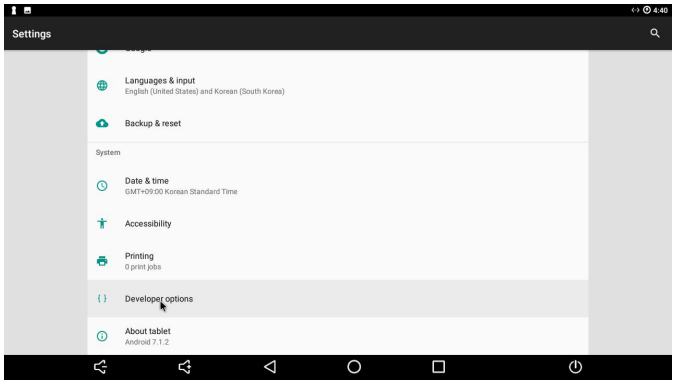


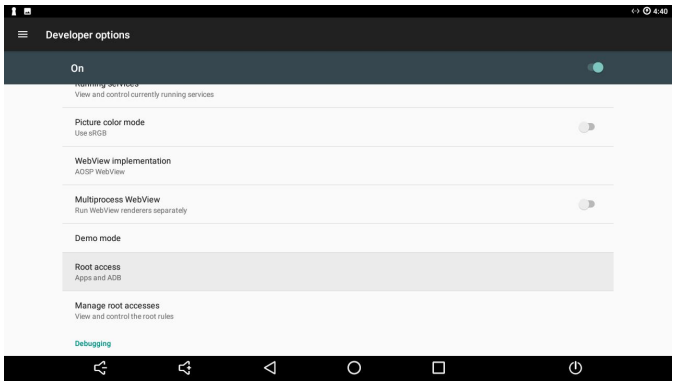
Si queremos ser los reyes del juego, sin duda, usar un teclado y un ratón en PUBG MOBILE nos aportará un mayor control de movimiento y una mejor capacidad de respuesta. La posibilidad de realizar bombardeos (movimiento lateral), señalar con el ratón y gestionar todas las funciones del juego con el teclado es una gran ventaja, algo que es obvio cuando los oponentes que usan controles táctiles tienen movimientos más limitados. Ninguno de estos métodos es oficialmente compatible con el juego, pero tenemos una alternativa muy simple.

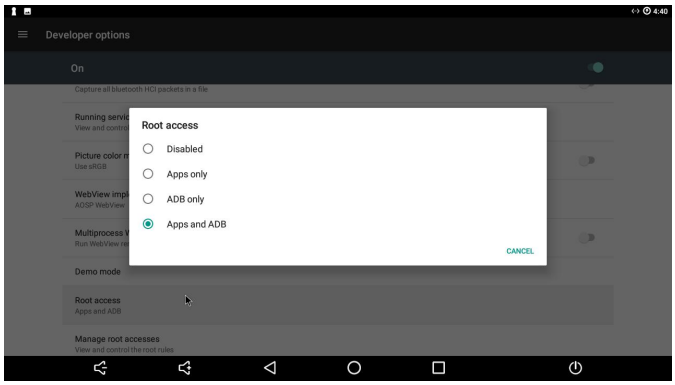
Instala Octopus desde Google Play para jugar con el teclado y el ratón en PUBG MOBILE. La aplicación ya viene con perfiles para PUBG por defecto. Ejecuta Octopus, luego selecciona PUBG MOBILE de la lista de juegos instalados.

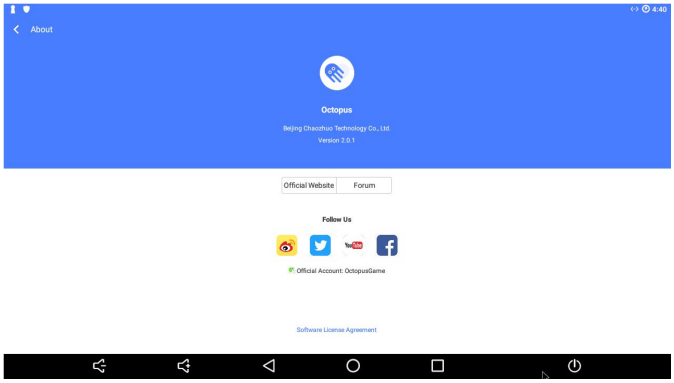
Dentro del juego, Octopus es fácil de usar. El icono de Octopus aparece en el lado izquierdo que muestra las opciones avanzadas, las cuales se pueden seleccionar con el ratón. Dentro del menú de configuración de Octopus, podemos cambiar entre Teclado y Gamepad, ambos ya tienen una plantilla configurada.

Un punto importante en esta aplicación son los ajustes. El nivel de transparencia de las teclas en pantalla puede reducirse cuando ya las tenemos memorizadas, de esta forma tendremos la pantalla más limpia. El nivel de sensibilidad POV es clave para controlar la velocidad de rotación del personaje, que por defecto es algo lenta. No olvides conceder acceso root a la aplicación Octopus para activar esta funcionalidad.





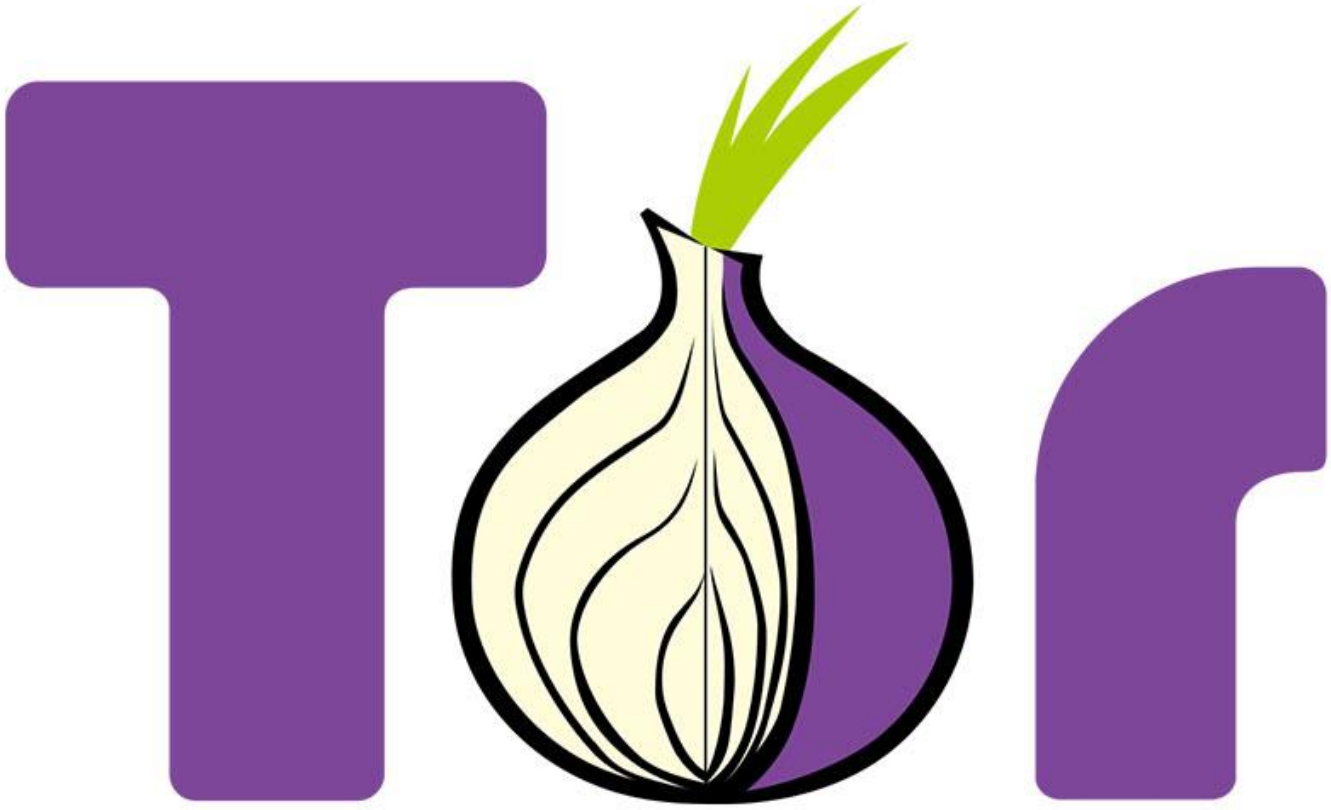




Una vez que inicies el juego PUBG a través de la aplicación Octopus, debes comprobar la configuración de los Gráficos, que debe coincidir con la de la Figura 7. El juego es bastante jugable con la **placa ODROID-XU4 de 59$**, aunque nos hemos encontrado aleatoriamente con algunas escenas entrecortadas.

Convirtiendo tu ODROID en un Repetidor Tor

July 1, 2018 By David Gabriel ODROID-XU4, Tutoriales



Tor es un software gratuito que permite acceder una red abierta muy útil con comunicaciones anónimas. El nombre deriva del acrónimo del nombre original del proyecto, The Onion Router. Protege tu privacidad al redireccionar el tráfico de Internet a través de una red de miles de repetidores, y evita que las utilidades de análisis de datos y vigilancia de red recopilen datos tuyos mientras navegas. En otras palabras, esto te hace "invisible" para que los sitios web no conozcan tu ubicación por tu dirección IP o por tu proveedor de servicios de Internet (ISP). Las personas que monitorizan tu red no podrán ver los sitios web o los recursos a los que accedes.

Todas las comunicaciones dentro de Tor están encriptadas. Cuando se envían datos, éstos se encriptan en la capa de aplicación varias veces y se anida como las capas de una cebolla. Las rutas de los datos incluyen repetidores seleccionados de forma aleatoria. Cada repetidor descifra una capa de cifrado que revela solo el siguiente repetidor y transmite la información restante. El proceso continúa hasta que el repetidor final descifra los datos originales y los envía al destino sin revelar la dirección IP de origen.

La desventaja de utilizar Tor es que tu conexión a Internet será más lenta de lo normal, debido a todas las fases de cifrado y descifrado que intervienen y al paso a través de múltiples repetidores. La velocidad de transferencia de información sería perceptiblemente menor.

Instalación

Primero, asegúrate de que el sistema esté actualizado utilizando los siguientes comandos:

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

Luego, instala la aplicación Tor y sus dependencias usando:

```
$ sudo apt-get install tor
```

Opcionalmente, también puede instalar Arm (forma abreviada de: monitor de repetidor anonimo), que es una aplicación para monitorizar y configurar Tor. Funciona de forma muy parecida a la utilidad de Linux llamada top, se puede instalar con el siguiente comando:

```
$ sudo apt-get install tor-arm
```

Configuración

Tor se puede personalizar modificando el archivo de configuración del mismo. Puedes usar tu editor de texto favorito para editar el archivo /etc/tor/torrc y agregar las opciones comentadas (usando #) que aparecen a continuación:

```
Log notice file /var/log/tor/notices.log # Log
file destination
RunAsDaemon 1 # Start process in background as
a daemon
ORPort 9001 # Port to be used by incoming
connections
DirPort 9030 # Port to be used by directory
connections
ExitPolicy reject *:* # Implies that your
relay will be used for
# relaying traffic inside the Tor network, but
# not for connections to external websites or
# other services
Nickname odroid-tor-relay # Can be anything
you like, so people
# don't have to refer to your relay by key
RelayBandwidthRate 100 KB # Throttle traffic
to 100KB/s (800Kbps)
RelayBandwidthBurst 200 KB # But allow bursts
up to 200KB/s (1600Kbps)
```

Si has instalado la aplicación Arm opcional, debes incluir las siguientes líneas de configuración en el archivo que hemos mencionado anteriormente:

```
ControlPort 9051 # Port to be used by
controller applications.
CookieAuthentication 1 # Authentication method
to be used by the
# controller application
DisableDebuggerAttachment 0 # Required by Arm
application to be able to use
# commands like netstat to monitor the network
# traffic
```

A continuación, reinicia Tor para que la configuración tenga efecto usando el comando:

```
$ sudo service tor restart
```

Si todo va bien, deberías ver una entrada en /varlog/tor/log como esta:

```
Jan 15 11:38:53.000 [notice] Tor has
successfully opened a circuit. Looks like
client functionality is working.
```

Ten en cuenta que, si tu red tiene un firewall, deberás configurarlo para que deje pasar las solicitudes entrantes en los puertos 9030 (para el servicio de directorio) y 9001 (para la actividad del repetidor). Es posible que tengas que consultar la Guía de usuario de tu firewall en particular para configurar esta opción. Si has instalado Arm, puedes iniciarlo usando el comando:

```
$ sudo arm
```

Aunque hay muchas opciones que puedes configurar, la más interesante está relacionadas con los gráficos que se generan para monitorizar todo el tráfico que pasa a través de tu repetidor. Consulta las opciones de ayuda de la aplicación Arm para obtener más información sobre cómo aprovechar al máximo esta aplicación.

Por defecto, Tor también es compatible con el protocolo Socket Secure (SOCKS), sobre el puerto 9050. Puede configurar tu navegador para que sea un cliente Tor y redirigir todas las conexiones a través del repetidor Tor protegiendo tu privacidad y mantener el anonimato. En Firefox, por ejemplo, puede ir a la opción Preferences > Advanced > Network > Settings > Change para configurar manualmente el proxy y añadir 127.0.0.1 en el puerto 9050 a la línea SOCKS y pulsar en OK para confirmar.

Para comprobar tu configuración, visita el sitio web del proyecto Tor http://bit.ly/1oh1f82 usando un navegador. Notarás que la IP pública que aparece en esta página es diferente a tu IP real. Este es el nodo de salida de tu solicitud, lo cual garantiza que no se puede rastrear tu ubicación o tu información personal. Ten en cuenta que los datos solo se cifran cuando pasan por la red Tor. Los datos se enviarán tal

como están, de modo que todo lo que no se haya cifrado desde el principio continuará siendo así después de dejar el nodo de salida.

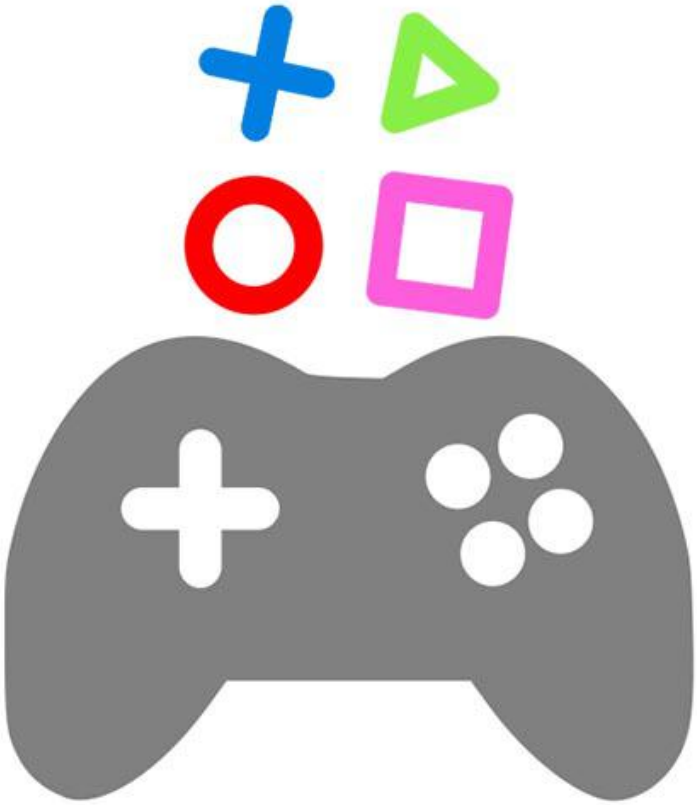
Si deseas desactivar esta función SOCKS y mantener tu ODROID solo como un repetidor, agrega la siguiente línea al archivo /etc/tor/torrc y reinicia el servicio Tor:

```
SocksPort 0 # Disable torsocks
```

El cliente Tor también se puede usar en otros sistemas operativos. La configuración puede variar un poco dependiendo del sistema operativo y del

Recalbox en el ODROID-XU4: Primeros Pasos

July 1, 2018 By Moe Long Juegos, ODROID-XU4



El ODROID-XU4 es un ordenador de placa reducida (SBC) que compite con la Raspberry Pi. Sus especificaciones ponen de manifiesto un potencial de rendimiento superior a la Pi, gracias a una CPU octa-core, el doble de RAM y un módulo eMMC. Como el Raspberry Pi, el ODROID-XU4 puede ejecutar una gran cantidad de sistemas operativos, incluyendo RetroPie, Ubuntu MATE y RecalBox. ¡Qué tal si aprendemos a poner en marcha Recalbox en el ODROID-XU4 para ejecutar juegos retro!

¿Qué es Recalbox?

Recalbox es un sistema operativo (SO) de juegos retro basado en Linux, similar a RetroPie. Está basado en RetroArch y usa el front-end de EmulationStation. Sin embargo, Recalbox está más orientado a principiantes, ya que éste ofrece unas configuraciones muy simplificadas, como menos sombreadores y menos opciones de personalización. En el ODROID-XU4, Recalbox tiene un rendimiento óptimo y permite ejecutar títulos más exigentes, aunque te darás cuenta que no soporta Advance MAME, Amiga 1200 ni Amiga 600. Sin embargo, cuando ejecutes Recalbox en las placas ODROID-XU4, verás que admite 3DO, un sistema notablemente ausente en la Raspberry Pi.

Si comparamos RetroPie con Recalbox o Lakka, Recalbox es más fácil de configurar que RetroPie y Lakka, pero no tan completo en sus opciones de configuración.

Pros:

- Fácil de configurar
- Excelente compatibilidad del sistema con ODROID-XU4
- Muy estable
- Incluye el Centro Multemedia Kodi para usarse como un ordenador personal de cine en casa (HTPC)

Contras:

- Menos opciones de personalización

- Puede quedarse un poco corto para los usuarios más avanzados

Materiales

Instalar Recalbox en el ODROID-XU4 es bastante simple. Solo necesitas la placa ODROID-XU4, una fuente de alimentación (PSU) de 5V/4A compatible, módulo eMMC o tarjeta microSD para ejecutar la imagen Recalbox ODROID-XU4 y una carcasa opcional, aunque recomendada.

- Placa ODROID-XU4
- Fuente de alimentación 5V/4A
- Carcasa
- Tarjeta de memoria
- Programa de extracción de archivos (7Zip)
- Programa para montar la imagen (Etcher)

El ODROID-XU4 está disponible de forma independiente por unos 60$ desde la web de Hardkernel (**https://www.hardkernel.com/main/products/prdt_info.php?g_code=G143452239825**) Si añades algunos accesorios como la fuente de alimentación, la tarjeta microSD y una carcasa ODROID-XU4, el precio final rondará los 90$.

Por este precio, obtendrás un ordenador de placa reducida con un procesador octa-core y 2 GB de RAM. En pruebas reales, el ODROID-XU4 supera a la Raspberry Pi 3 B+ cuando ejecutas ROM que implican un uso intensivo del sistema, como PlayStation Portable (PSP), Nintendo 64 (N64) y Sega Dreamcast.

Costo total: 62$ (solo la placa), 90$ (placa con accesorios)

Instalación

Primero, dirígete al sitio web de Recalbox y descarga la última versión para ODROID-XU4. De la lista de descargas, hazte con la imagen para el ODROID-XU4.

Dado que es un archivo img.xz, deberás usar un programa como 7Zip para extraer el archivo de imagen. Cuando hayas descomprimido el img.xz, te quedarás con un archivo de imagen.

A continuación, deberás montar el archivo de imagen en un soporte de arranque, como un módulo eMMC o una tarjeta microSD. En mi instalación de Recalbox en ODROID-XU4, yo utilicé una tarjeta microSD. Usa un programa como Etcher para montar la imagen en tu soporte de instalación.

Una vez hayas grabado la imagen Recalbox para ODROID-XU4 en una tarjeta microSD o en un módulo eMMC, introdúcela en la ranura correcta de tu ordenador de placa reducida ODROID-XU4 y enciéndelo. Debería iniciarse directamente con la interfaz de Recalbox EmulationStation.

Configuración posterior a la instalación

Cuando se haya iniciado correctamente Recalbox en el ODROID-XU4, estarás listo para empezar a ejecutar juegos retro. Sin embargo, es bueno realizar algunas configuraciones posteriores a la instalación. Aunque puedes almacenar tus ROM de juegos en tu tarjeta microSD o módulo eMMC, es posible que quieras almacenarlos en una unidad externa, como una memoria USB. Yo guardo mis ROM en una unidad flash de 256 GB. Para hacer esto, abre el menú principal. Desde allí, navega hasta System Settings > Storage Device. Puedes elegir entre almacenamiento interno, un disco específico o cualquier dispositivo externo.

Con una unidad externa conectada, selecciona esa unidad, ésta aparecerá en el menú del dispositivo de almacenamiento con su tamaño, como 64 GB o 128 GB. Luego, apaga tu sistema, retira la memoria USB y conéctala a un PC. Encontrarás una carpeta Recalbox con carpetas para tus BIOs y ROMs. De esta forma, si tu sistema de archivos Recalbox se corrompe, pierdes tu tarjeta microSD o simplemente desea realizar una nueva instalación, tus ROMs y BIOs estarán a buen recaudo.

Mejoras de rendimiento

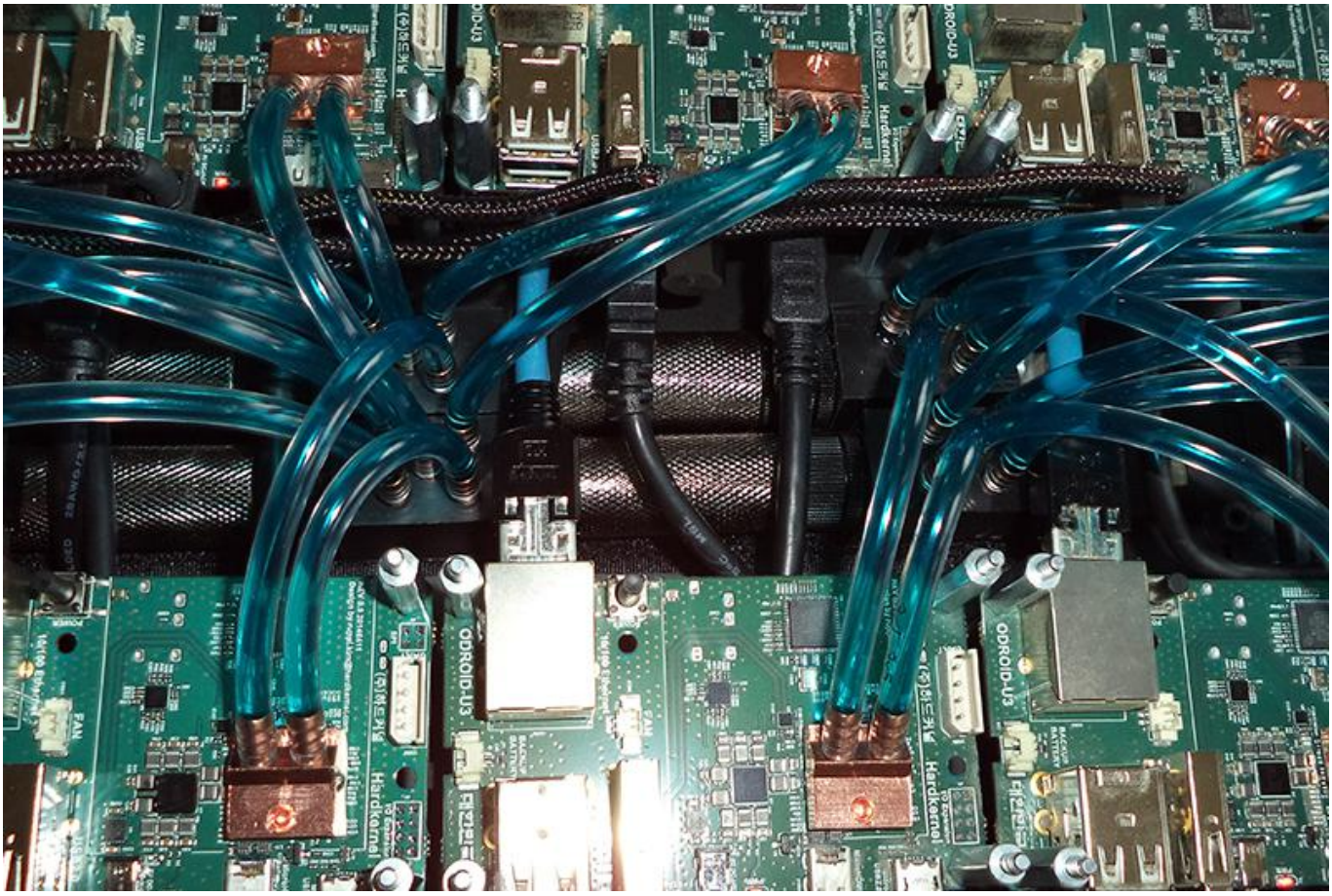
Al igual que la versión RetroPie para ODROID-XU4, notará un mejor rendimiento de títulos que requieren un uso intensivo del sistema como son las ROMS DE PSP, N64 y Dreamcast. Sin realizar configuraciones, los juegos de PSP funcionan bastante bien en ODROID-XU4 con Recalbox. Aunque títulos como God of War: Chains of Olympus son difíciles de emular, son algo jugables aunque con una tasa de frames más baja. Juegos como Burnout Legends funcionan bien, aunque la GPU Mali en el ODROID-XU4 ocasionalmente tienes algunos fallos de video que no llegan a obstaculizan el sistema de juego.

Los juegos de Dreamcast funcionan a toda velocidad. Al igual que ocurre con el emulador PPSSPP, de vez en cuando notarás algunos fallos de video menores como son las líneas extrañas bajo el emulador. Los títulos de N64 en la versión de Recalbox ODROID-XU4 funcionan mucho mejor que en la Raspberry Pi 3. A pesar de que algunos juegos se niegan a ejecutarse a toda velocidad, como Conker's Bad Fur Day. El rendimiento en general es mejor que en la Pi y juegos como éste son mis favoritos. Goldeneye 64, es bastante jugable. En última instancia, el ODROID-XU4 tiene un rendimiento bastante superior a la Raspberry Pi 3.

Reflexiones finales

Refrigeración Líquida Parte 1 – Clúster

July 1, 2018 By Uli Abromeit ODROID-XU4, Mecaniqueo



Tras encontrar algunos pequeños disipadores de calor de 15x15x5mm, decidí crear un clúster ODROID usando refrigeración por agua para reducir su temperatura y el ruido. Empecé con un solo ODROID-XU4 para ver si los pequeños disipadores de calor eran lo suficientemente potentes como para redirigir el calor lejos de la placa. Tras las pruebas iniciales, decidí conectar el sistema de refrigeración al resto del clúster, tal como se muestra en las imágenes.

Equipo de refrigeración

- Alphacool DC-LT 3600 Ceramic – 12V DC, Alphacool DC-LT Plexi top y deposito Alphacool (**http://bit.ly/1vDYvJJ**)
- Alphacool MCX ram copper edition (**http://bit.ly/1C3t8Ml**)
- Alphacool MCX 5x divider (**http://bit.ly/1qYh1vr**)
- Radiador Alphacool NeXxus Monsta 140 con NB-Blacksilent Pro PK2 (**http://bit.ly/1Fi5yrA**)
- Radiador de 120 mm
- Bomba de 8V con depósito
- Convertidor DC-DC regulable para controlar la velocidad del ventilador y la bomba

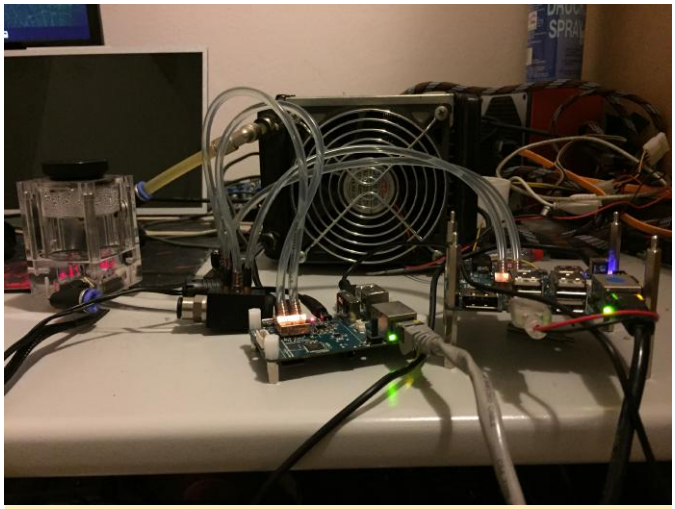


Figura 1 – La configuración no es demasiado complicada si quieres montar un sistema de refrigeración líquida.

Clúster

- 10 x ODROID-U3
- 1 x ODROID-XU

- 1 x ODROID-XU4
- 2 x PSU 5V 20A
- NW-Switch con 24 puertos
- 2 x Switch HDMI con 8 puertos

Figura 2 – Aquí vemos los ODROIDS refrigerados por líquido con una agradable configuración como la descrita arriba.

Servidor GlusterFS de 200TB que Utiliza el ODROID-HC2 para Aplicaciones Distribuidas Masivamente.

July 1, 2018 By BaxterPad Linux, Mecaniqueo, ODROID-HC2

Con el transcurso de los años, he ido actualizado varias veces el almacenamiento de mi casa. Como muchos, empecé con un NAS de consumo. El primero fue un Netgear ReadyNAS, luego varios dispositivos QNAP. Hace aproximadamente dos años, me cansé de las limitaciones de la CPU y de la memoria de QNAP, y de dispositivos similares, así que decidí montar el mío propio usando un Supermicro XEON D, Proxmox y FreeNAS. Era estupendo, pero añadir más unidades me suponía un auténtico dolor de cabeza. La migración entre niveles ZRAID era básicamente imposible si no contabas con varios discos adicionales.

El fiasco de FreeNAS 10 fue la gota que colmó el vaso. Quería poder añadir discos en cantidades más pequeñas y quería mejores modos ante posibles fallos parciales, algo así como “unRAID”, al mismo tiempo que me permitiera escalar a tantos discos como quisiera. También quería evitar cualquier fallo que pudiera darse en el adaptador de bus, la placa base o la fuente de alimentación.

He estado experimentando con GlusterFS y Ceph usando aproximadamente cuarenta pequeñas máquinas virtuales (VM) para simular varias configuraciones y posibles fallos como son la pérdida de energía, fallos en disco, archivos corruptos, etc. Al final, GlusterFS fue el que me dio mejor resultado a la hora de proteger mis datos porque incluso si GlusterFS sufría un fallo integral, mis datos serían en su mayor parte recuperables gracias a que estarían almacenados en un sistema de archivos ext4 en mis nodos. Ceph también realizaba un gran trabajo, pero era bastante frágil (aunque recuperable) y difícil de configurar.

Opte por el ODROID-HC2 con 8 núcleos, 2 GB de RAM, ethernet Gbit y un puerto SATA, el cual ofrece una muy buena base para aplicaciones de distribución masiva. Me hice con cuatro ODROID y volví a

experimentar con GlusterFS. Tras poner a prueba mi idea, volví a pedir otros 16 nodos y empecé a migrar mi matriz actual.

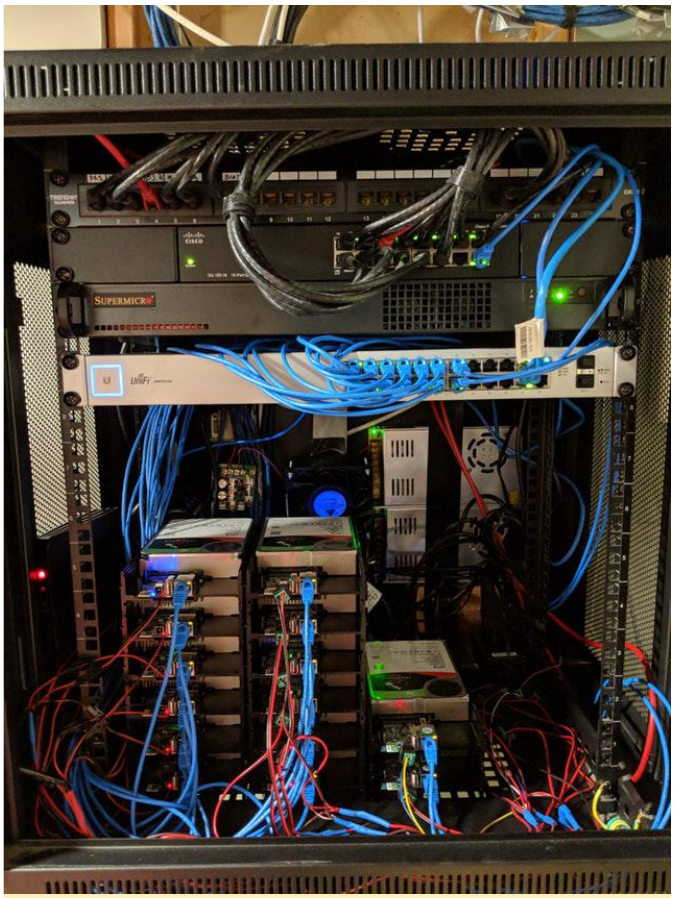


Figura 1: Este es un excelente ejemplo de clúster ODROID que gestiona datos reales.

En una prueba de velocidad, logré mantener una tasa de escritura de 8 GBPS y de lectura de 15 GBPS a través de la red cuando las operaciones están suficientemente distribuidas en el sistema de archivos. La lectura de un único archivo está limitada por el rendimiento de 1 nodo, se llegan a alcanzan los 910 Mbit de lectura/escritura.

En términos de consumo de energía, una carga de CPU moderada con una elevada carga trabajo de disco (reequilibrio de la matriz), una carcasa pfSense, 3 switches, 2 puntos de acceso Unifi, un módem Fios Verizon y varias máquinas virtuales en el host XEON-D, me suponía un consumo alrededor de los 250 vatios. Donde yo vivo, en Nueva Jersey, esto equivale a un coste de unos 350$ al año en electricidad. Estoy escribiendo este artículo porque no he logrado encontrar demasiada información sobre cómo usar el ODROID-HC2 con cualquier escala que me resulte significativa.

Listado de componentes

- ODROID-HC2 **https://www.hardkernel.com/main/products/prdt_info.php?g_code=G151505170472**
- Tarjeta MicroSD de 32GB. Puedes optar por una de tan sólo 8GB pero el ahorro es insignificante. **https://www.amazon.com/gp/product/B06XWN9Q99/**
- Cables de Ethernet cat6 finos **https://www.amazon.com/gp/product/B00BIPI9XQ/**
- Ventilador de 200CFM 12V 120mm (5”) **https://www.amazon.com/gp/product/B07C6HR3PP/**
- Controlador de velocidad PWM de 12V, para regular el ventilador. **https://www.amazon.com/gp/product/B00RXKNT5S/**
- Conectores en forma de barril de 5.5mm x 2.1mm (0.21 “x 0.08”) para alimentar los ODROID. **https://www.amazon.com/gp/product/B01N38H40P/**
- Fuente de alimentación 12V/30A. Puedes alimentar 12 ODROID con disco duro de 3.5” sin problemas. **https://www.amazon.com/gp/product/B00D7CWSCG/**
- Switch administrado con 24 puertos gigabits de Unifi **https://www.amazon.com/gp/product/B01LZBLO0U/**

Lo curioso es que no hay mucho que configurar en GlusterFS. Eso es lo que más me gusta de esto. Literalmente, se necesitan tres comandos para activar y ejecutar GlusterFS después de instalar el SO y formatear los discos. Posiblemente publicaré una reseña en mi github en las próximas semanas. Primero quisiera probar Presto (**https://prestodb.io/**), un motor SQL distribuido, en estas máquinas antes de escribir nada.

```
$ sudo apt-get install glusterfs-server
glusterfs-client
$ sudo gluster peer probe gfs01.localdomain
...  gfs20.localdomain
$ sudo gluster volume create gvol0 replicate 2
transport tcp
gfs01.localdomain:/mnt/gfs/brick/gvol1 ...
gfs20.localdomain:/mnt/gfs/brick/gvol1
$ sudo cluster volume start gvol0
```

Para comentarios, preguntas y sugerencias, visita el artículo original en **https://www.reddit.com/r/DataHoarder/comments/8ocjxz/200tb_glusterfs_odroid_hc2_build/**.

Home Assistant: un Inteligente Proyecto de Bricolaje con la Iluminación

July 1, 2018 By Adrian Popa ODROID-C1+, ODROID-C2, ODROID-XU4, Tutoriales, ODROID-N1

Desde que empecé a trabajar con Home Assistant y automatizar varias cosas en casa, siempre he querido encontrar la forma de controlar las luces. Eché un vistazo a las bombillas inteligentes, como las Philips Hue, pero son demasiado caras (aproximadamente 12$ por bombilla – **https://goo.gl/xiAqEe**). Además, la mayoría de las soluciones usan protocolos registrados o servicios en la nube que pueden filtrar información personal, o pueden dejar de funcionar en el futuro, lo cual podía llegar a convertirlas en simples pisapapeles. Por casualidad, oí hablar de los conmutadores wifi Sonoff creados por Itead (**http://sonoff.itead.cc/en/products/sonoff/sonoff-basic**). Combinan un microcontrolador WiFi ESP8266 y un relé que te permite encender y apagar a distancia. Además de tener un precio bajo (~5$ – **https://goo.gl/WP31Ny**), el microcontrolador también cuenta con algunos GPIO y se puede actualizar con software de código abierto para adaptarlo a tus necesidades.

De modo que, el plan era hacerme con varios conmutadores Sonoff Basic, retirar la carcasa, grabar el firmware de Tasmota (**https://github.com/arendst/Sonoff-Tasmota/wiki**) y encontrar una forma de conectarlos a mis interruptores de luz existentes. En lugar de esto, también podría haber usado un conmutador de luz wifi (**http://sonoff.itead.cc/en/products/residential/sonoff-touch**), pero era más caro y deseaba mantener mis interruptores existentes. Afortunadamente encontré una guíaen youtube con una idea sobre cómo convertir un conmutador básico en un conmutador de luz: **https://www.youtube.com/watch?v=ab472a40-co**. Básicamente necesitas alimentar el Sonoff, conectar su salida al circuito de la

bombilla/iluminación y conectar el conmutador de luz a las almohadillas GPIO14 y GND, y listo.

Grabar Tasmota

Lo primero que debes hacer es abrir la caja del conmutador Sonoff (está sujeta con unos clips de plástico) e identificar las almohadillas GPIO (consulte la figura 1). Tienes un pad cuadrado de 3.3V, UART RX y TX, la puesta a tierra y el último es el GPIO14. Para grabar un nuevo firmware necesita encender el Sonoff, conectar el UART a un dispositivo que ejecute Arduino y mantener presionado el botón (GPIO0) mientras se esté iniciado para ponerlo en modo programación. Para hacer esto, puedes soldar un cabezal de 5 pines a la placa o simplemente colocar el cabezal para que toque los pads mientras grabas (esto es lo que hice yo). Ten en cuenta que, durante esta operación, el Sonoff no se conectará a la red eléctrica y consumirá energía desde cable UART. (más detalles: **https://goo.gl/5TUfz8**). Si no tienes el adaptador USB-UART adecuado, puede usar los pines UART del cabezal de 40 pines de un ODROID-C1/C2 directamente con cables jumper (consulte la figura 1). Asegúrate de conectar RX y TX cruzados para que RX de Odroid esté conectado al TX de Sonoff. Además, la potencia debe venir del pin de 3.3V y no del pin de 5V.

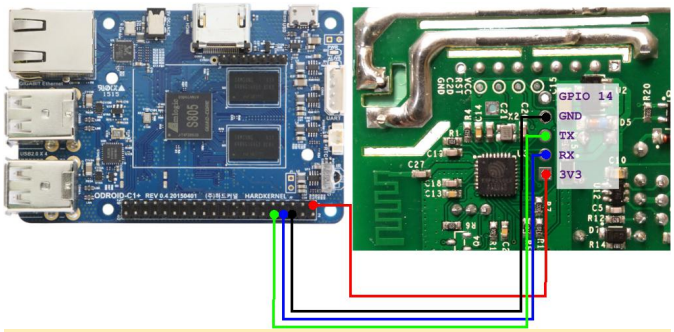


Figura 1 – Diagrama de cableado GPIO

En lo que respecta al software, necesitarás descargar e instalar Arduino IDE (yo utilicé la Ver. 1.8.5) y algunas librerías de soporte, que vienen con el programa Java armhf (**https://github.com/arendst/Sonoff-Tasmota/wiki/Arduino-IDE**). Puedes descargar el IDE para armhf desde este enlace: **https://goo.gl/ZFYj8Z**.

Si está utilizando el ODROID-C2/N1, el programa arduino no se ejecutará porque no podrá encontrar las librerías de 32 bits aplicables. Necesitará instalar soporte de 32 bits en tu ubuntu y luego ejecutar los mismos pasos que en el caso del ODROID-C1:

```
$ sudo dpkg --add-architecture armhf
$ sudo apt-get update
$ sudo apt-get install libc6:armhf libx11-
6:armhf libxext6:armhf
libxrender1:armhf libxtst6:armhf libxi6:armhf
```

Si estás utilizando ODROID-XU4/N1, los pasos son los mismos que antes, pero deberá usar un shifter shield para hacer frente a la diferencia de voltaje entre el ODROID y el chip ESP. Para el C1 sigue estos pasos:

```
$ unxz arduino-1.8.5-linuxarm.tar.xz
$ tar xvf arduino-1.8.5-linuxarm.tar
$ cd arduino-1.8.5
$ ./arduino
```

Espera a que se inicie, luego abre Arduino IDE y selecciona File -> Preferencias y añade el siguiente texto para el campo Additional Boards Manager URLs: **https://goo.gl/EVq7nf** y selecciona OK. A continuación, abre Tools -> Boards ... -> Boards Manager ... y desplázate hacia abajo y haz clic en esp8266 de ESP8266 Cmmnunity. Haga clic en el botón Installl para descargar e instalar el último software de la placa ESP8266. Selecciona Close. Ahora puede cerrar Arduino. En el siguiente paso, descargarás e instalará el firmware de Tasmota. Puedes obtener la última versión desde aquí: **https://github.com/arendst/Sonoff-Tasmota/releases**

```
$ cd ..
$ wget https://goo.gl/KQwDvr
$ tar zxvf v5.13.1.tar.gz
$ cp -ar Sonoff-Tasmota-5.13.1/lib/* arduino-
1.8.5/libraries/
$ cp -ar Sonoff-Tasmota-5.13.1/sonoff/
arduino-1.8.5/
$ cd arduino-1.8.5
$ ./arduino
```

Ahora puedes abrir el proyecto Tasmota seleccionando File -> Open, localizando y cargando sonoff.ino desde el directorio sonoff bajo arduino-1.8.5. En la lista de pestañas abiertas localiza user_config.h, donde debes cambiar la siguiente configuración. Ten en cuenta que cambiar los ajustes

en este punto no es obligatorio; la mayoría se pueden configurar más tarde a través de la interfaz web o MQTT, pero una vez que resetees los valores predeterminados del hardware, estos serán los valores por defecto:

- Configurar la dirección IP estática si quieres (WIFI_IP_ADDRESS, WIFI_GATEWAY, WIFI_SUBNETMASK, WIFI_DNS)
- Definir SSID/Contraseña para dos puntos de acceso (STA_SSID1, STA_PASS1, etc)
- Ajustar la herramienta de configuración para WIFI_RETRY (WIFI_CONFIG_TOOL), de modo que, si no puede conectarse al WiFi, seguirá intentándolo en lugar de convertirse en un punto de acceso. De lo contrario, algún atacante podría eliminarlo de la red WiFi con un ataque deauth (ver **https://goo.gl/76kYPs**) y forzarlo a convertirse en un punto de acceso y tener que volver a configurarlo.
- Activar syslog si tienes uno (ayuda a resolver problemas de depuración, aunque los registros locales se conservarán en el dispositivo, accesibles a través de la interfaz web hasta que se reinicie)
- Desactivar (comentar con //) MQTT TLS (// # define USE_MQTT_TLS)
- Fijar usuario, contraseña e IP del broker MQTT (MQTT_HOST, MQTT_USER, MQTT_PASS)
- Deshabilitar MQTT Retain (me causó algunos dolores de cabeza) (MQTT_BUTTON_RETAIN, MQTT_POWER_RETAIN, MQTT_SWITCH_RETAIN)
- Definir un tema para los mensajes MQTT (MQTT_TOPIC). Algo como kids_light o bedroom_light por ejemplo
- Desactivar Domoticz si no es necesario (// # define USE_DOMOTICZ)
- Fijar un nombre de usuario y contraseña para la interfaz web para tener algo de protección (WEB_USERNAME, WEB_PASSWORD)
- Deshabilitar mDNS discovery (// # define USE_DISCOVERY)
- Configurar servidores NTP (NTP_SERVER1, etc.)
- Configurar la zona horaria (APP_TIMEZONE)
- Deshabilitar I2C si no se usa (// # define USE_I2C)

Una vez completada tu configuración, debes definir el modo de actualización haciendo los cambios que se detallan aquí: https://goo.gl/NmnZBE en Arduino IDE. Tendrás que seleccionar /dev/ttyS2 como un puerto con 115200 baudios.

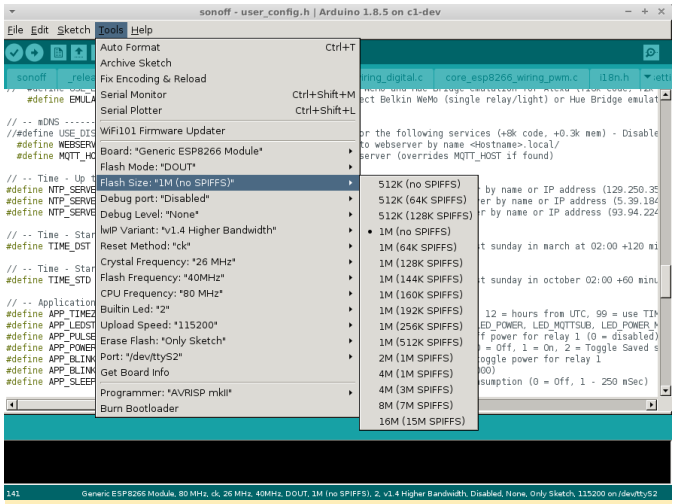


Figura 2. Extracto de user_config.h y configuración de la actualización

Cuando termines, inserta el cabezal pin en el Sonoff mientras mantienes presionado el botón de encendido y suéltalo cuando tengas una conexión estable. Empieza a compilar y grabar (el segundo botón con un símbolo de flecha en Arduino IDE) y espera a que el proceso se complete. Si la actualización falla, intenta poner de nuevo el Sonoff en modo programación e inténtalo de nuevo. Una vez que el proceso haya finalizado con exito, enciende el Sonoff y se debería conectar a tu WiFi. Puedes alimentarlo a través del cable UART improvisado o desde la red eléctrica, pero asegúrate de volver a montarlo si usas la red eléctrica para que no te arriesges a recibir una descarga eléctrica. Puedes averiguar su dirección IP (ya sea estática o de DHCP) consultando la lista de clientes en tu router. A continuación, puedes conectarse a través de HTTP a su dirección de gestión para continuar configurándolo.

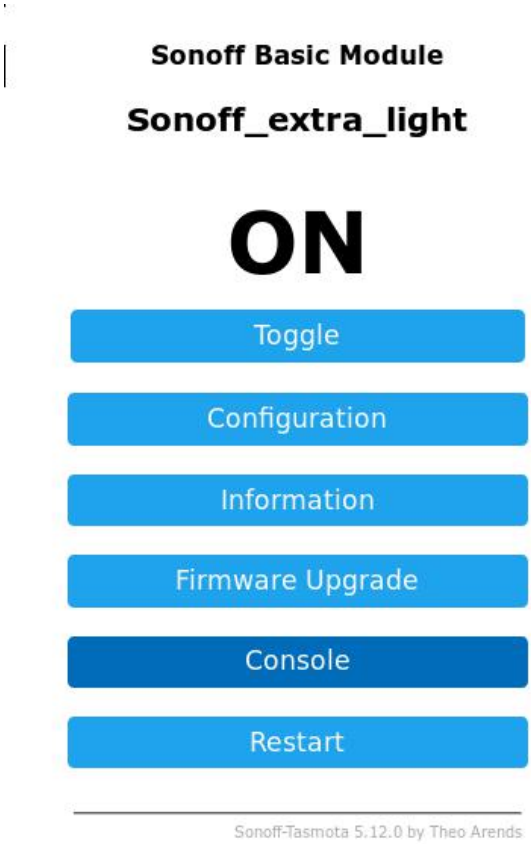


Figura 3. Gestión web

Configuración de Tasmota

El firmware Tasmota está en su mayor parte desarrollado teniendo en cuenta MQTT. MQTT es un protocolo de transmisión de mensajes de máquina a máquina que ya hemos tratado en un artículo anterior (**https://goo.gl/9ggqHJ**). Permite la integración de múltiples entidades con un broker de mensajes (estamos usando mosquitto), y además es compatible con Home Assistant. El firmware Tasmota tiene muchos parámetros que puede configurar a través de MQTT o REST API, (La lista completa la tienes aquí: **https://github.com/arendst/Sonoff-Tasmota/wiki/Commands**), pero lo más importante es que se puede configurar a través de la interfaz web.

Nosotros configuraremos lo siguiente:

- GPIO14 tendrá el mismo rol que Switch1. Para ello, puede navegar por la interfaz web, dentro de Configuration -> Configure Module: Module Type: 01 Sonoff basic GPIO14 sensor: 09 Switch1
- Si todavía no has configurado MQTT en user_config.h, puedes hacerlo ahora desde Configuration -> Configure MQTT y agregar la dirección del broker y el nombre de usuario/contraseña.

Si estás utilizando contraseñas con MQTT, deberás añadir la nueva cuenta en mosquitto:

```
$ sudo mosquitto_passwd /etc/mosquitto/passwd
sonoffuser
```

Si esperas un segundo, Sonoff debería conectarse al broker MQTT y deberías poder controlarlo desde la línea de comandos. Suponiendo que estés utilizando el tema bedroom_light, puedes activar el interruptor con:

```
$ mosquitto_pub -p 1883 -u sonoffuser -P
sonoffpassword
cmnd/bedroom_light/POWER -m "1"
```

Puedes consultar la configuración o la información del dispositivo a través de MQTT, aunque las respuestas se enviarán al servidor syslog, de modo que no debes perderlo de vista al emitir comandos MQTT. Para ver el estado del conmutador ejecuta:

```
$ mosquitto_pub -p 1883 -u sonoffuser -P
sonoffpassword cmnd/bedroom_light/POWER -m ""
```

El correspondiente resultado del servidor syslog sería este:

```
May 25 17:43:42 sonoff_bedroom ESP-MQT:
stat/bedroom_light/RESULT = {"POWER":"ON"}
May 25 17:43:42 sonoff_bedroom ESP-MQT:
stat/bedroom_light/POWER = ON
```

Ahora puedes jugar con los parámetros de configuración que no están disponibles a través de la interfaz web, como PowerOnState (**https://goo.gl/hJqRTd**):

```
$ mosquitto_pub -p 1883 -u sonoffuser -P
sonoffpassword
cmnd/bedroom_light/PowerOnState -m "3"
```

Una cosa más que puedes configurar es la gestión de energía ESP. Está deshabilitada por defecto, pero puede indicarle que entre en reposo 100ms para reducir el consumo eléctrico (y el calor) con este comando:

```
$ mosquitto_pub -p 1883 -u sonoffuser -P
sonoffpassword
cmnd/bedroom_light/Sleep -m "100"
```

Para integrar el Sonoff como interruptor de luz en Home Assistant puedes añadir lo siguiente a configuration.yaml y reiniciar:

```
light:
- platform: mqtt
name: "Bedroom Light"
state_topic: "stat/bedroom_light/RESULT"
value_template: '{{ value_json["POWER"] }}'
command_topic: "cmnd/bedroom_light/POWER"
availability_topic: "tele/bedroom_light/LWT"
qos: 1
payload_on: "ON"
payload_off: "OFF"
payload_available: "Online"
payload_not_available: "Offline"
retain: false
…
group:
lights:
name: Lights
view: yes
icon: mdi:lightbulb-on-outline
entities:
- light.bedroom_light
```

¡Asegúrate de que el código anterior utiliza los temas correctos que configurastes para su dispositivo! Puedes reiniciar Home Assistant y probar que todo funciona como cabría esperar (deberías poder activar y desactivar el interruptor desde Home Assistant, la interfaz web de Tasmota y el botón en Sonoff).

El Montaje del hardware

Para continuar y conectar el interruptor de luz al GPIO14 necesitamos que soldar un poco y agregar un filtro de paso bajo junto con los cables del conector al GPIO14 y GND. La razón por la que necesitamos un filtro de paso bajo es porque la radiación EM será recogida por los cables utilizados (actúan a modo de antena) y puede ocasionar que el interruptor se active fortuitamente a veces. Para mitigar esto, puede usar un filtro de paso bajo para permitir pasar solo la DC y también puede torcer los cables para que la interferencia se cancele (**http://goo.gl/7zEPc9**).

Puedes montar un filtro de paso bajo siguiendo esta guía (**https://www.youtube.com/watch?v=aq8_os6g13s**) con una resistencia de 10k y un condensador de 33pF. Los valores exactos no son tan importantes – puedes usar los que tenga.

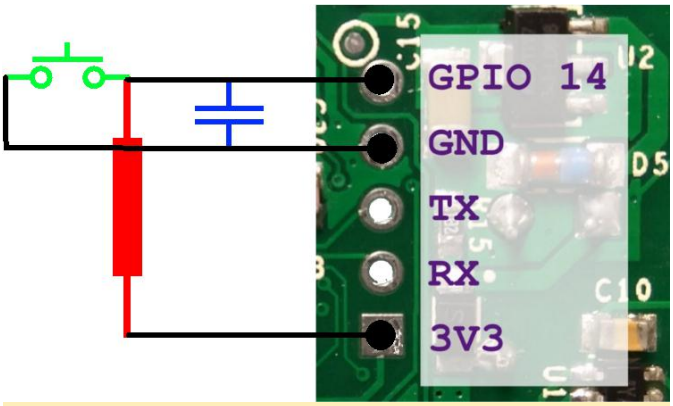


Figura 4 – Esquema

Asegúrate de que la soldadura sea sólida y usa tubos termocontraíbles para aislar sus componentes. Cuando termines, prueba la unidad teniendo cuidado de no tocar los restos de CA que están a la vista. También deberías utilizar cinta eléctrica puesto que no utilizaremos la carcasa de plástico.

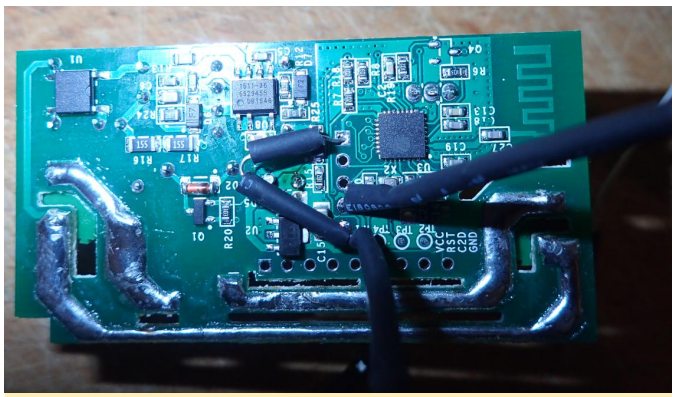


Figura 5 – Unidad ensamblada

Conectando Sonoff a las luces

Este paso puede ser el más complicado, porque el cableado de la iluminación puede variar de un lugar a otro y es posible que no tenga todo lo que necesites en un único lugar. Es posible que tengas que consultar a un electricista, recurrir al reglamento de baja tensión y a la legislación local antes de continuar. Lo que es aún más importante, asegúrate de desconectar la luz y los enchufes antes de desmontar cualquier interruptor de luz o correrás el riesgo de sufrir una descarga eléctrica. También necesitarás un multímetro (**https://en.m.wikipedia.org/wiki/Test_light**) para

identificar los cables que están cargados de electricidad.

Analicemos brevemente qué cables son los pasan a través de tus paredes hasta las tomas de corriente y las luces. Tendrás un cable “caliente” (también conocido como “cargado”) que tiene un voltaje de 110/220V dependiendo de dónde vivas, un cable “neutro” que se utiliza para completar el circuito y proporciona la ruta que toma la corriente eléctrica (éste normalmente no tiene energía) y un cable de “puesta a tierra” usado en algunos enchufes eléctricos que proporciona una ruta de emergencia para el cable neutro. Para alimentar el Sonoff necesitamos el cable con carga y el neutro del interior de la carcasa del interruptor de luz.

Desafortunadamente, no todos los estándares de cableado proporcionan el cable neutro, así que es posible que tengamos que hacer uno. En mi caso (y sospecho que será muy común en la mayor parte de Europa) el interruptor de la luz tiene una entrada cargada que va a dos interruptores mecánicos y que continua con dos cables de salida cargados independientes que van por la pared hasta la luz.

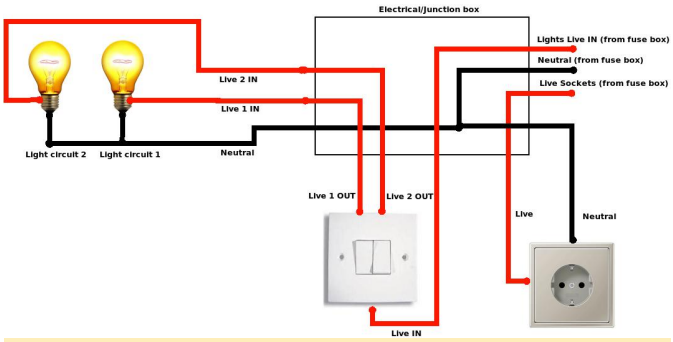


Figura 6 – Esquema de la luz

No hay neutro cerca del interruptor de la luz. El neutro va a la misma luz y cierra el circuito con la bombilla. Bueno, esto no servirá. Necesitamos modificar el circuito y detener una salida cargada y transformarla en una neutral con un poco de cableado, principalmente porque no queremos pasar otro cable por la pared (algo complicado sin cierta destreza y herramientas especiales).

Lo primero que debes hacer es cortar la energía de los interruptores y retirar el interruptor de la luz de la pared para echar un vistazo a los cables. Podría ser una buena idea probar y ver si el PCB Sonoff encaja dentro de la caja eléctrica del interruptor de la luz (no coge en la mayoría de las cajas eléctricas redondas de Europa del Este). Si no te encaja, debes ser creativo y buscar un lugar donde poder colocarlo.

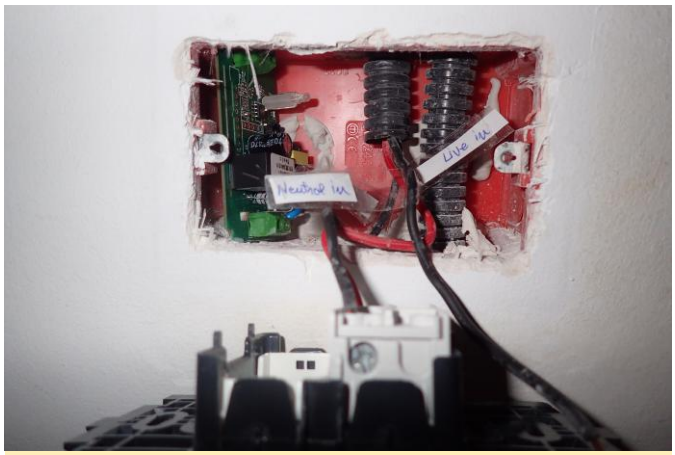


Figura 7 – Testeando la caja de luz y cableado por defecto

En mi caja de luz, como puedes ver en la figura 7, solo tenía una entrada cargada para ambos interruptores (rojo) y dos salidas cargadas para las bombillas (negro). Debes tomarte tu tiempo para identificar cada cable con un multímetro (cuando el interruptor esté encendido) y etiquetar los cables (con el interruptor apagado, naturalmente).

Mi intención es convertir un cable de salida cargado en una entrada neutral desconectando el cable de salida de la bombilla y conectándolo al neutro. Al hacer esto, no podré usar una lámpara con dos luces independientes, aunque de todas formas sólo tengo lámparas sencillas, así que no tengo problema. Hay dos lugares donde puede hacer esto: en la caja eléctrica intermedia (que debe estar muy cerca, próxima al techo) o en la propia luz. Ambos diagramas de cableado los tienes en la figura 8, voy a describir los pasos necesarios para cada variante.

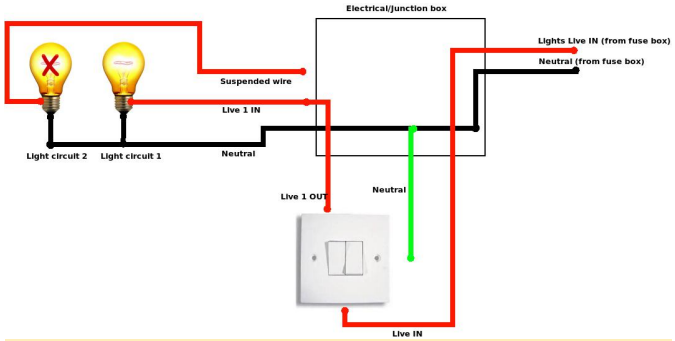


Figura 8a – Opciones de cableado: re-cableado en la caja de conexiones

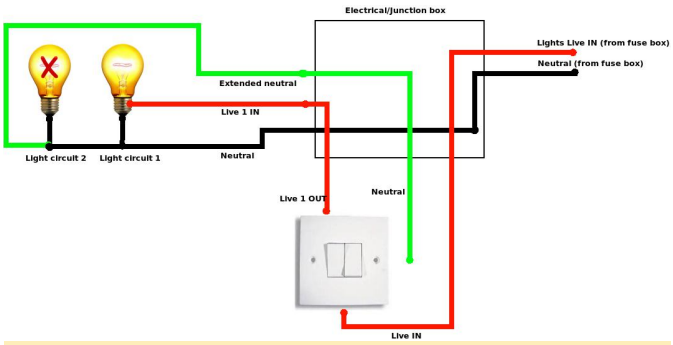


Figura 8b – Opciones de cableado – re-cableado en la propia luz

Para la opción 8A necesitas echar un vistazo en la caja eléctrica de conexiones. Esta caja debe contener todos los circuitos eléctricos que van a una habitación: luces y enchufes. Cuando mires dentro (con el interruptor apagado) probablemente verás un revoltijo de cables, como en la figura 9. Necesitas identificar (según el color y la posición) qué cables van a tu interruptor de luz y cuáles van a la bombilla. También debes elegir qué cable de salida cargado deseas convertir a neutro e identificar el extremo superior de ese cable en la caja eléctrica. Puedes hacer esto tirando del cable y observando qué cable es el que se mueve, o midiendo la resistencia de los extremos del cable (prolongado si es necesario) con un ohmímetro (la resistencia debería ser cercana a cero). Asegúrate de etiquetar todos los cables que identifiques para futuras consultas, en caso de que necesites deshacer todo este revoltijo más adelante.

Figura 9 – Caja eléctrica – Antes de hacer nada

A continuación, debes identificar un cable neutro en la caja eléctrica. Deberías encontrar dos manojos de cables más grandes con 3 o más cables conectados entre sí. Un manojo conecta todos los cables cargados (el que viene del interruptor principal con el que va al interruptor de la luz y los que van a los enchufes) y el otro manojo contendrá todos los neutros. Puedes encontrar cuál es, identificando cuál es el cable cargado que va a tu interruptor de luz, o midiéndolos con un multímetro.

Ahora (con los interruptores apagados), necesita desconectar la salida cargada desde cable que va a la bombilla y conectarla con los otros neutros. El cable restante no estará conectado (pero sigue siendo bueno etiquetarlo y aislar su extremo).

Figura 10 – Caja eléctrica – Después de los cambios

En este punto, puedes conectar el sonoff con la entrada cargada y el neutro en su entrada, y la salida cargada en la salida (con los interruptores aún apagados) y luego probar tu configuración.

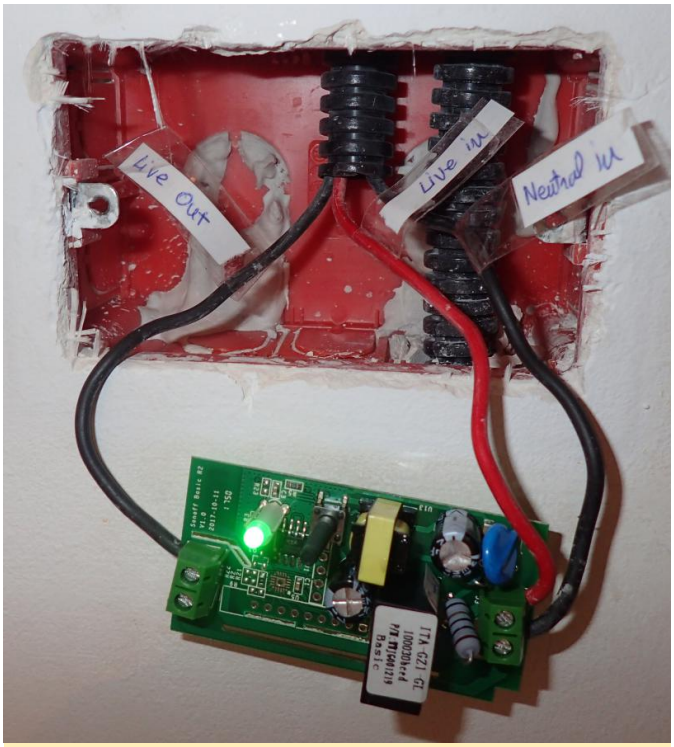


Figura 11 – Probando los cambios

Para la opción 8B no necesitas hacer cambios en la caja eléctrica. Tuve que hacer esto porque la caja eléctrica de mi habitación no es de fácil acceso. En este caso, llevamos el neutro de la lámpara al interruptor de la luz conectando el cable neutro disponible de la lámpara directamente a un viejo cable "cargado" que llamaremos neutro a partir de ahora. Ten mucho cuidado al identificar cuál es el cable correcto; de lo contrario, si conectas directamente el neutro al cable cargado real, crearás un cortocircuito y tu fusible saltará o se fundirá.

Al igual que en el ejemplo anterior, tendrás que conectar la entrada cargada y el neutro a la entrada del Sonoff y la salida cargada a la salida (con los interruptores desconectados). Conecta el interruptor de luz a GPIO14 y GND también. Una vez que las pruebas tengan éxito, tendrás que colocar suavemente el sonoff dentro de la caja eléctrica del interruptor de luz y volver a colocar el interruptor de luz en la pared y listo.

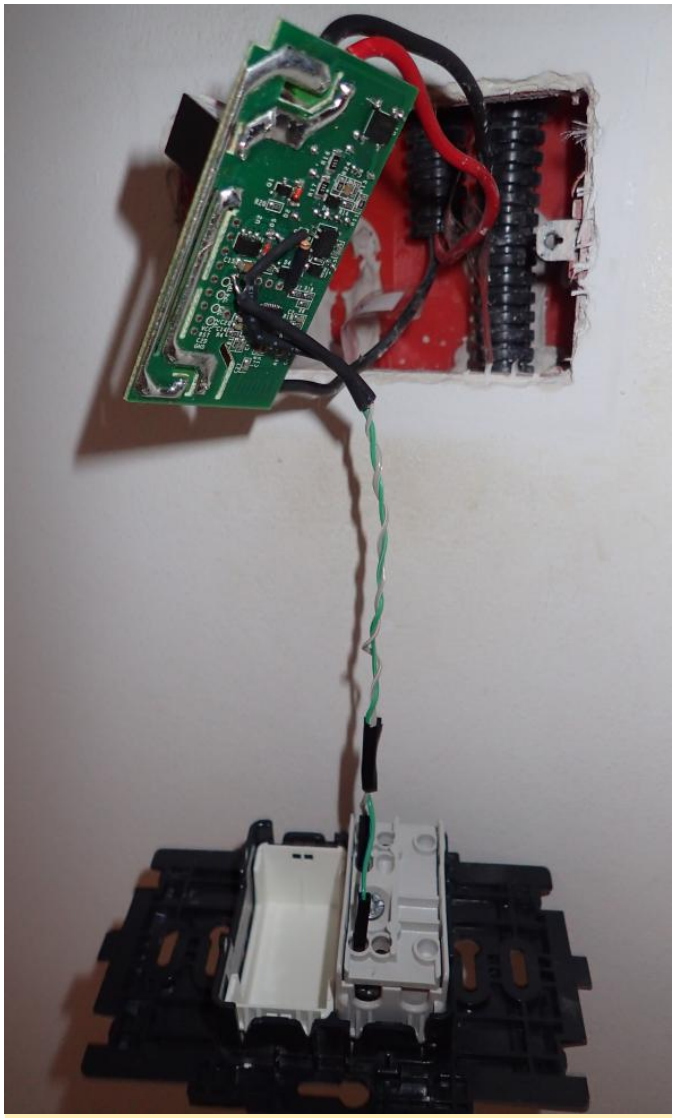


Fig 12 – Montaje final

Automatizaciones

Ahora debería tener interruptores de luz funcionales en Home Assistant, que también puede alternar con el viejo interruptor mecánico, aunque no parece tan impresionante para la cantidad de trabajo que hemos invertido. ¿Por qué quería luces "inteligentes" en primer lugar? Vamos a verlo.

Apagar la luz después de 30 minutos

En ocasiones, es posible que olvides una luz encendida cuando sales de casa para el trabajo. Con luces habilitadas por red puedes apagarlas tú mismo, pero esto no es automatización. En mi caso, a los niños les gusta quedarse dormidos con las luces encendidas (a pesar de la luz de la noche), así que siempre teníamos que apagarla una vez que estuviesen dormidos. Si se despertaban en medio de la noche, las encienden para volverse a dormir. Así

que, una forma elegante de arreglar esto es crear una automatización que entre un determinado intervalo de tiempo (23: 00-17: 00) se apaguen las luces tras haber estado encendidas durante 30 minutos. Debemos editar automations.yaml y añadir las siguientes automatizaciones:

```
- action:
- data:
entity_id: light.kids_light
service: homeassistant.turn_off
alias: Turn off Kids Light after 30 min
inactivity
condition:
- after: '23:00'
before: '17:30'
condition: time
id: '1525335992266'
trigger:
- entity_id: light.kids_light
for:
minutes: 30
platform: state
to: 'on'
- action:
- data:
entity_id: light.kids_light
service: homeassistant.turn_off
alias: Turn off Kids Light at 23:00
condition: []
id: '1525343566208'
trigger:
- at: '23:00'
platform: time
```

La primera automatización escucha los cambios de estado que suceden cuando la luz se enciende y si está en el intervalo de tiempo requerido espera 30 minutos antes de apagarla. La segunda automatización gestiona el caso en el que la luz se enciende digamos, a las 22:00, así que la automatización anterior no tendrá efecto (el cambio de estado debe ocurrir entre las 23: 00-17: 00) – por lo tanto, se apagará a las 23:00. Se pude mejorar esta automatización y hacer que verifique si alguien está en casa/en la habitación, o que funciona solo cuando salga el sol, etc.

Desactivar el interruptor de hardware

Si has tenido niños pequeños, te habrás dado cuenta que hay un período de su desarrollo en el que necesitan presionar todos los botones que ven. Si consigen localizar el interruptor de la luz, lo pueden encender y apagar hasta romperlo. Para evitar eso, puedes bloquear el interruptor de luz mecánico para que no se encienda la luz cuando se presiona. Simplemente fija GPIO14 en none dentro la configuración. Puedes hacer esto sobre MQTT y hará que el sonoff se reinicie (la luz parpadea durante unos 250 ms). Puedes crear los siguientes scripts en Home Assistant, dentro de scripts.yaml:

```
'enable_kids_light':
alias: Enable Sonoff Kids Light
sequence:
- service: mqtt.publish
data:
topic: cmnd/kids_light/Gpio14
payload: 9
qos: 1
retain: false
'disable_kids_light':
alias: Disable Sonoff Kids Light
sequence:
- service: mqtt.publish
data:
topic: cmnd/kids_light/Gpio14
payload: 0
qos: 1
```

Cuando llames al script (ya sea desde el menú Servicios o desde un panel de instrumentos), las luces se bloquearán o desbloquearán y podrás controlarlas únicamente a través del wifi.

Despertador

La última aplicación es algo más macabra. Si tienes personas con un sueño muy profundo en tu familia que necesitan mucho para despertarse, ¿qué te parece depertarlos con la tompreta de las fuerzas armadas de los EE. UU. (**https://www.youtube.com/watch?v=xt4hSs4lWPg**)? Además del sonido, también podemos hacer que la luz destelle con el mismo patrón que la música. Posiblemente resulte un poco molesto.

Puedes empezar descargando el audio y conviértelo a mp3:

```
$ sudo apt-get install youtube-dl ffmpeg
$ youtube-dl -f 140
"https://www.youtube.com/watch?v=xt4hSs4IWPg"
$ ffmpeg -i "US_Military_Bugle_Wake_Up_Call-
xt4hSs4IWPg.m4a"
-acodec libmp3lame -b:a 128k
"Wakeup_Trumpet.mp3"
```

Yo uso MPD como reproductor multimedia controlado por Home Assistant, así que asegúrate de mover el archivo MP3 a algun lugar dentro del directorio de la biblioteca de medios de MPD, para que pueda reproducirse (generalmente dentro de /var/lib/mpd/music /). También necesitarás añadirlo a una lista de reproducción guardada (yo utilicé "wakeupalarm" para nombrarlo).

```
$ cp Wakeup_Trumpet.mp3
/var/lib/mpd/music/wake-up/
```

A continuación, necesitamos producir algún tipo de ritmo. Debe corresponder a las señales de encendido y apagado de la bombilla. Mi idea era "escribir" el ritmo usando el teclado mientras escuchaba el sonido, de modo que al presionar/mantener presionada una tecla se encendiera la luz, al liberar la tecla la luz se apagaría. Podemos usar evtest para seleccionar el teclado y registrar la duración y la secuencia de pulsaciones de las teclas:

```
$ sleep 10; ffplay Wakeup_Trumpet.mp3
$ sudo evtest 2>&1 | tee trumpet.txt
```

Una vez hecho esto, el archivo debería contener líneas como las siguientes, registrando los eventos de arriba/abajo para la letra "A":

```
Event: time 1525422859.108421, ----------
SYN_REPORT ------------
Event: time 1525422859.204406, type 4
(EV_MSC), code 4 (MSC_SCAN), value 70004
Event: time 1525422859.204406, type 1
(EV_KEY), code 30 (KEY_A), value 1
Event: time 1525422859.204406, ----------
SYN_REPORT ------------
Event: time 1525422859.300420, type 4
(EV_MSC), code 4 (MSC_SCAN), value 70004
Event: time 1525422859.300420, type 1
(EV_KEY), code 30 (KEY_A), value 0
```

El archivo trumpet.txt será tu archivo de ritmo y debe copiarse a /home/homeassistant/.homeassistant/. También puedes hacerte con mi interpretación desde aquí: **https://goo.gl/cFpLbs**.

```
$ sudo cp trumpet.txt
/home/homeassistant/.homeassistant/
```

Ahora podemos crear una aplicación appdaemon (descrita en este artículo anterior: **https://goo.gl/npGqoX**) que cuando se active mediante una entrada booleana, ordene a MPD que reproduzca la trompeta y analice el archivo de texto de ritmo y emita el correspondiente comando de encendido/apagado para la luz. Puedes conseguir la aplicación y su configuración aquí (compilada para appdaemon 3.x): **https://goo.gl/Kn7PLK**.

```
$ cd /home/homeassistant/.homeassistant/apps
$ wget -O sound_and_light_alarm.py
https://goo.gl/aNuUB6
```

Adapta la siguiente configuración para que coincida con tu entorno (cambios realizados en apps/sound_and_light_alarm.yaml):

```
sound_and_light_alarm:
module: sound_and_light_alarm
class: SoundAndLightAlarm
media_player: "media_player.mpd_kids"
light: "light.kids_light"
music: "wakeupalarm"
rhythm:
"/home/homeassistant/.homeassistant/trumpet.tx
t"
trigger: "input_boolean.wake_up"
```

Puedes añadir el siguiente booleano a Home Assistant (configuration.yaml):

```
input_boolean:
wake_up:
name: Wake up
initial: off
```

Y la siguiente automatización se activará cuando lo necesites, por ejemplo durante los días de la semana a las 7:00 (automations.yaml):

```
- id: '1527938039163'
alias: Wake up alarm
trigger:
```

```
- at: 07:00
platform: time
condition:
condition: time
weekday:
- mon
- tue
- wed
- thu
- fri
action:
- data:
entity_id: input_boolean.wakeup
service: input_boolean.turn_on
```

Puedes verlo en acción aquí: **https://www.youtube.com/watch?v=ac9xnA6Y918**. Desde el exterior, parecera que tu casa está intentando comunicarse a través de código Morse con una nave extraterrestre, pero posiblemente despertara a los más dolmilones de la familia.

Solución de problemas

Si notas que las luces se apagan (o encienden) aparentemente al azar, pueden suceder varias cosas. En mi caso, mis luces a veces perdían la conectividad TCP al servidor MQTT y se restablecía una nueva conexión. Una vez que la conexión estaba activa, leía el estado del servidor MQTT y anulaba el estado local. Si está utilizando la opción "retain" de forma incoherente, te generará problemas, como que una luz activa se apague tras una reconexión. En estos casos, es mejor desactivar la retención de MQTT y valerse del estado local del interruptor. Esto significa que, si la luz está encendida y hay un corte de energía, la luz se encenderá cuando se restablezca la energía y no podrás forzarla a salir a través de MQTT mientras que la luz no esté encendida. Puedes encontrar más detalles en: https://goo.gl/ECUqRY.

Un problema más al que podrías enfrentarte es que cuando reinicias Home Assistant, tus entidades de luz estarán en modo "off" por defecto, incluso si hay una luz encendida. Para arreglar esto, puede indicarle a Home Assistant que "pregunte" a todas las luces cuál es su estado con la siguiente automatización (añadir a automations.yaml):

```
- action:
- data:
topic: cmnd/sonoffs/POWER
service: mqtt.publish
alias: Get Sonoff states on restart
condition: []
id: '1518951376931'
trigger:
- event: start
platform: homeassistant
```

Espero que este artículo te ayude a implementar un sistema de iluminación inteligente sin demasiado coste o por empidimento de un proveedor.

Primeros Pasos con OpenCL: Usando el ODROID-XU4

July 1, 2018 By cnxsoft Linux, ODROID-XU4

Aunque he testeado OpenGL ES con herramientas como glmark2-es2 y es2gears, así como también demos de WebGL en Chromium, no he llegado a probar OpenCL, ya que no estoy muy familiarizado con él, excepto que se usa con GPGPU (GPU de propósito general) para acelerar tareas como el procesamiento de imagen/audio. Esta es una buena excusa para aprender un poco más, probarlo en la placa y escribir una breve guía para poner en marcha OpenGL sobre hardware con la GPU Arm Mali. La finalidad de este tutorial es mostrar cómo ejecutar una muestra de OpenCL y la utilidad OpenCL, no entraré en los pormenores del código de OpenCL. Si quieres más información sobre la codificación OpenCL en Arm, un buen lugar por donde empezar sería verificar el código fuente de las muestras proporcionadas.

Arm Compute Library y muestras de OpenCL Como no sabía por dónde empezar, Hardkernel me redirigió a un hilo del foro donde se muestra cómo usar la Arm Compute Library para probar OpenCL en la placa. La publicación tiene fecha de enero de 2018 y está basado en la Compute Library 17.12, pero puedes consultar la última versión y documentación en **https://arm-software.github.io/ComputeLibrary/latest/**. Cuando escribí este artículo, la última versión era la 18.03, así que la recuperé y traté de compilarla tal y como se indica:

```
$ wget https://github.com/ARM-software/ComputeLibrary/archive/v18.03.tar.gz
$ tar xvf v18.03.tar.gz
$ cd ComputeLibrary-18.03/
$ sudo apt install scons
$ scons Werror=1 -j8 debug=0 neon=1 opencl=1 embed_kernels=1 os=linux arch=armv7a build=native
```

Sin embargo, falló con:

```
$ g++: internal compiler error: Killed (program cc1plus)
```

Al analizar el registro log del kernel con dmesg, estaba claro que el problema estaba en que la placa se había quedado sin memoria: "Out of memory: Kill process 4984 (cc1plus) Out of memory: Kill process 4984 (cc1plus)". Así que tuve que configurar un archivo de intercambio swap (1GB):

```
$ sudo dd if=/dev/zero of=/swapfile bs=1024
count=1M
$ sudo chown root.root /swapfile
$ sudo chmod 0600 /swapfile
$ sudo mkswap /swapfile
$ sudo swapon /swapfile
```

El archivo swap nos proporciona más memoria:

```
free -m
total used free shared buff/cache available
Mem: 1994 336 520 34 1137 1568
Swap: 1023 0 1023
```

Volví a iniciar la compilación con NEON y OpenCL activado:

```
$ scons Werror=1 -j8 debug=0 neon=1 opencl=1
embed_kernels=1 os=linux arch=armv7a
build=native
```

Esta vez se completó:

```
scons: done building targets.
```

Actualizar

La configuración de ZRAM en lugar de swap suele ser mejor solución en caso de que te quedes sin memoria, tal y como se describe **https://www.cnx-software.com/2018/05/14/running-out-of-ram-in-ubuntu-enable-zram/**. Podemos copiar las librerías a /usr/lib, que nos proporciona un montón de muestras para probar:

```
$ sudo cp build/*.so /usr/lib/
$ ls examples/
SConscript graph_mobilenet_qasymm8.cpp
cl_convolution.cpp graph_resnet50.cpp
cl_events.cpp graph_squeezenet.cpp
cl_sgemm.cpp graph_squeezenet_v1_1.cpp
gc_absdiff.cpp graph_vgg16.cpp
gc_dc.cpp graph_vgg19.cpp
graph_alexnet.cpp neon_cartoon_effect.cpp
graph_googlenet.cpp neon_cnn.cpp
graph_inception_v3.cpp neon_convolution.cpp
graph_inception_v4.cpp neon_copy_objects.cpp
graph_lenet.cpp neon_scale.cpp
graph_mobilenet.cpp
neoncl_scale_median_gaussian.cpp
```

Ten en cuenta que algunas solo son NEON, no usan OpenCL, el prefijo explica el tipo de muestra:

- cl_*.cpp –> OpenCL examples
- gc_*.cpp –> GLES compute shaders examples
- graph_*.cpp –> Graph examples
- neoncl_*.cpp –> NEON / OpenCL interoperability examples
- neon_*.cpp –> NEON examples

Todas las muestras han sido compiladas y se pueden encontrar en el directorio build/examples. Yo ejecuté cl_convolution después de generar una imagen ppm Raw usando Gimp:

```
$ time ./cl_convolution ~/ODROID-XU4Q-
Large.ppm
$ ./cl_convolution

Test passed

real 0m5.814s
user 0m4.893s
sys 0m0.758s
```

Procesó la foto (5184 x 3456) en menos de 6 segundos. Si analizamos la imagen resultante, podemos ver el resultado con una escala de grises desde la complejidad de OpenCL, tal y como se muestra en la Figura 1.

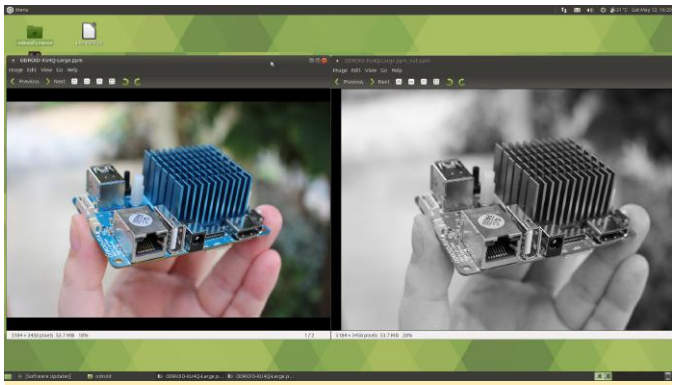


Figura 01 – Imagen original (izquierda) vs Después de la Convolución OpenCL (derecha)

He repetido una operación similar con convert que no ha sido compilado con soporte OpenCL, sólo usa software:

```
$ time convert ODROID-XU4Q-Large.ppm -
colorspace Gray ODROID-XU4Q-Large-
Grayscale.ppm

real 0m10.475s
user 0m0.724s
sys 0m2.957s
```

Le llevo algo más de 10 segundos, casi el doble del tiempo utilizado por la demo de OpenCL. Sin embargo, los archivos de imagen PPM ocupan más de 50 MB, de modo que parte del tiempo se usa para leer y guardar el archivo desde la memoria flash eMMC. Repetir las pruebas proporcionaron rendimientos similar (~ 6s vs ~ 11s), de modo que eran insignificantes. El resultado del comando "convert version" que muestra OpenCL no forma parte de las características habilitadas en ImageMagick:

```
$ convert -version
Version: ImageMagick 6.9.7-4 Q16 arm 20170114
http://www.imagemagick.org
Copyright: © 1999-2017 ImageMagick Studio LLC
License:
http://www.imagemagick.org/script/license.php
Features: Cipher DPC Modules OpenMP
Delegates (built-in): bzlib djvu fftw
fontconfig freetype jbig jng jpeg lcms lqr
ltdl lzma openexr pangocairo png tiff wmf x
xml zlib
```

Era divertido, así que probé otra muestra:

```
$ time ./cl_events ~/ODROID-XU4Q-Large.ppm
$ ./cl_events

Test passed

real 0m3.068s
user 0m2.527s
sys 0m0.369s
```

¿Qué hizo? Cuando abrí el archivo, tenía el mismo aspecto que la primera muestra (imagen en escala de grises), pero en realidad la imagen estaba escalada (50% de ancho, 50% de alto):

```
$ file ~/ODROID-XU4Q-Large.ppm_out.ppm
/home/odroid/ODROID-XU4Q-Large.ppm_out.ppm:
Netpbm image data, size = 2592 x 1728,
rawbits, pixmap
```

La última muestra cl_sgemm manipula las matrices. El objetivo principal de los tres OpenCL (muestras cl_xxx_) es mostrar cómo usar OpenCL Convolution, Events y SGEMM (Single-precision GEneral Matrix Multiply) utilizando Compute Library. También puedes jugar con otras muestras para NEON y OpenGL ES. La comunidad de ARM publicó un post en el blog explicando cómo ejecutar neon_cartoon_effect en Raspberry Pi y explicando el código fuente en detalle. En realidad, no necesitas una placa RPi para esto, ya que cualquier placa ARM con un procesador que soporte NEON debería funcionar.

Utilidad clinfo

clinfo es una utilidad que muestra información sobre los dispositivos y plataformas OpenCL en el sistema, **https://github.com/Oblomov/clinfo**, que se instala fácilmente:

```
$ sudo apt install clinfo
```

Sin embargo, ejecutar el programa no devuelve ninguna información útil:

```
$ clinfo
Number of platforms 0
```

Esto no es lo que esperaba. Afortunadamente, la configuración de clinfo está explicada en el artículo de ODROID
Magazine **https://magazine.odroid.com/article/clinfo-compiling-the-essential-opencl-gpu-tuning-utility-for-the-odroid-xu4/**, así que vamos a intentarlo. Necesitamos usar el driver framebuffer de Mali, luego configuramos el archivo ICD del proveedor:

```
$ sudo apt install mali-fbdev
$ sudo mkdir -p /etc/OpenCL/vendors
$ sudo sh -c 'echo "/usr/lib/arm-linux-
gnueabihf/mali-egl/libOpenCL.so" >
/etc/OpenCL/vendors/armocl.icd'
```

Ahora podemos ejecutar clinfo:

```
$ clinfo
Number of platforms 1
Platform Name ARM Platform
Platform Vendor ARM
Platform Version OpenCL 1.2 v1.r12p0-
04rel0.03af15950392f3702b248717f4938b82
```

```
Platform Profile FULL_PROFILE
Platform Extensions
cl_khr_global_int32_base_atomics
cl_khr_global_int32_extended_atomics
cl_khr_local_int32_base_atomics
cl_khr_local_int32_extended_atomics
cl_khr_byte_addressable_store
cl_khr_3d_image_writes cl_khr_fp64
cl_khr_int64_base_atomics
cl_khr_int64_extended_atomics cl_khr_fp16
cl_khr_gl_sharing cl_khr_icd cl_khr_egl_event
cl_khr_egl_image cl_arm_core_id cl_arm_printf
cl_arm_thread_limit_hint
cl_arm_non_uniform_work_group_size
cl_arm_import_memory
Platform Extensions function suffix ARM

Platform Name ARM Platform
Number of devices 2
Device Name Mali-T628
Device Vendor ARM
Device Vendor ID 0x6200010
Device Version OpenCL 1.2 v1.r12p0-
04rel0.03af15950392f3702b248717f4938b82
Driver Version 1.2
Device OpenCL C Version OpenCL C 1.2 v1.r12p0-
04rel0.03af15950392f3702b248717f4938b82
Device Type GPU
Device Profile FULL_PROFILE
Device Available Yes
Compiler Available Yes
Linker Available Yes
Max compute units 4
Max clock frequency 600MHz
Device Partition (core)
Max number of sub-devices 0
Supported partition types None
Max work item dimensions 3
Max work item sizes 256x256x256
Max work group size 256
Preferred work group size multiple 4
Preferred / native vector sizes
char 16 / 16
short 8 / 8
int 4 / 4
long 2 / 2
half 8 / 8 (cl_khr_fp16)
float 4 / 4
double 2 / 2 (cl_khr_fp64)
Half-precision Floating-point support
(cl_khr_fp16)
Denormals Yes
Infinity and NANs Yes
Round to nearest Yes
Round to zero Yes
Round to infinity Yes
IEEE754-2008 fused multiply-add Yes
Support is emulated in software No
Single-precision Floating-point support (core)
Denormals Yes
Infinity and NANs Yes
Round to nearest Yes
Round to zero Yes
Round to infinity Yes
IEEE754-2008 fused multiply-add Yes
Support is emulated in software No
Correctly-rounded divide and sqrt operations
No
Double-precision Floating-point support
(cl_khr_fp64)
Denormals Yes
Infinity and NANs Yes
Round to nearest Yes
Round to zero Yes
Round to infinity Yes
IEEE754-2008 fused multiply-add Yes
Support is emulated in software No
Address bits 64, Little-Endian
Global memory size 2090397696 (1.947GiB)
Error Correction support No
Max memory allocation 522599424 (498.4MiB)
Unified memory for Host and Device Yes
Minimum alignment for any data type 128 bytes
Alignment of base address 1024 bits (128
bytes)
Global Memory cache type Read/Write
Global Memory cache size 131072 (128KiB)
Global Memory cache line size 64 bytes
Image support Yes
Max number of samplers per kernel 16
Max size for 1D images from buffer 65536
pixels
Max 1D or 2D image array size 2048 images
Max 2D image size 65536x65536 pixels
Max 3D image size 65536x65536x65536 pixels
Max number of read image args 128
Max number of write image args 8
Local memory type Global
Local memory size 32768 (32KiB)
Max number of constant args 8
Max constant buffer size 65536 (64KiB)
Max size of kernel argument 1024
Queue properties
Out-of-order execution Yes
Profiling Yes
Prefer user sync for interop No
```

```
Profiling timer resolution 1000ns
Execution capabilities
Run OpenCL kernels Yes
Run native kernels No
printf() buffer size 1048576 (1024KiB)
Built-in kernels
Device Extensions
cl_khr_global_int32_base_atomics
cl_khr_global_int32_extended_atomics
cl_khr_local_int32_base_atomics
cl_khr_local_int32_extended_atomics
cl_khr_byte_addressable_store
cl_khr_3d_image_writes cl_khr_fp64
cl_khr_int64_base_atomics
cl_khr_int64_extended_atomics cl_khr_fp16
cl_khr_gl_sharing cl_khr_icd cl_khr_egl_event
cl_khr_egl_image cl_arm_core_id cl_arm_printf
cl_arm_thread_limit_hint
cl_arm_non_uniform_work_group_size
cl_arm_import_memory

Device Name Mali-T628
Device Vendor ARM
Device Vendor ID 0x6200010
Device Version OpenCL 1.2 v1.r12p0-
04rel0.03af15950392f3702b248717f4938b82
Driver Version 1.2
Device OpenCL C Version OpenCL C 1.2 v1.r12p0-
04rel0.03af15950392f3702b248717f4938b82
Device Type GPU
Device Profile FULL_PROFILE
Device Available Yes
Compiler Available Yes
Linker Available Yes
Max compute units 2
Max clock frequency 600MHz
Device Partition (core)
Max number of sub-devices 0
Supported partition types None
Max work item dimensions 3
Max work item sizes 256x256x256
Max work group size 256
Preferred work group size multiple 4
Preferred / native vector sizes
char 16 / 16
short 8 / 8
int 4 / 4
long 2 / 2
half 8 / 8 (cl_khr_fp16)
float 4 / 4
double 2 / 2 (cl_khr_fp64)
Half-precision Floating-point support
(cl_khr_fp16)
Denormals Yes
Infinity and NANs Yes
Round to nearest Yes
Round to zero Yes
Round to infinity Yes
IEEE754-2008 fused multiply-add Yes
Support is emulated in software No
Single-precision Floating-point support (core)
Denormals Yes
Infinity and NANs Yes
Round to nearest Yes
Round to zero Yes
Round to infinity Yes
IEEE754-2008 fused multiply-add Yes
Support is emulated in software No
Correctly-rounded divide and sqrt operations
No
Double-precision Floating-point support
(cl_khr_fp64)
Denormals Yes
Infinity and NANs Yes
Round to nearest Yes
Round to zero Yes
Round to infinity Yes
IEEE754-2008 fused multiply-add Yes
Support is emulated in software No
Address bits 64, Little-Endian
Global memory size 2090397696 (1.947GiB)
Error Correction support No
Max memory allocation 522599424 (498.4MiB)
Unified memory for Host and Device Yes
Minimum alignment for any data type 128 bytes
Alignment of base address 1024 bits (128
bytes)
Global Memory cache type Read/Write
Global Memory cache size 131072 (128KiB)
Global Memory cache line size 64 bytes
Image support Yes
Max number of samplers per kernel 16
Max size for 1D images from buffer 65536
pixels
Max 1D or 2D image array size 2048 images
Max 2D image size 65536x65536 pixels
Max 3D image size 65536x65536x65536 pixels
Max number of read image args 128
Max number of write image args 8
Local memory type Global
Local memory size 32768 (32KiB)
Max number of constant args 8
Max constant buffer size 65536 (64KiB)
Max size of kernel argument 1024
Queue properties
Out-of-order execution Yes
```

```
Profiling Yes
Prefer user sync for interop No
Profiling timer resolution 1000ns
Execution capabilities
Run OpenCL kernels Yes
Run native kernels No
printf() buffer size 1048576 (1024KiB)
Built-in kernels
Device Extensions
cl_khr_global_int32_base_atomics
cl_khr_global_int32_extended_atomics
cl_khr_local_int32_base_atomics
cl_khr_local_int32_extended_atomics
cl_khr_byte_addressable_store
cl_khr_3d_image_writes cl_khr_fp64
cl_khr_int64_base_atomics
cl_khr_int64_extended_atomics cl_khr_fp16
cl_khr_gl_sharing cl_khr_icd cl_khr_egl_event
cl_khr_egl_image cl_arm_core_id cl_arm_printf
cl_arm_thread_limit_hint
cl_arm_non_uniform_work_group_size
cl_arm_import_memory

NULL platform behavior
clGetPlatformInfo(NULL, CL_PLATFORM_NAME, ...)
ARM Platform
clGetDeviceIDs(NULL, CL_DEVICE_TYPE_ALL, ...)
Success [ARM]
clCreateContext(NULL, ...) [default] Success
[ARM]
clCreateContextFromType(NULL,
CL_DEVICE_TYPE_DEFAULT) Success (1)
Platform Name ARM Platform
Device Name Mali-T628
clCreateContextFromType(NULL,
CL_DEVICE_TYPE_CPU) No devices found in
platform
clCreateContextFromType(NULL,
CL_DEVICE_TYPE_GPU) Success (2)
Platform Name ARM Platform
Device Name Mali-T628
Device Name Mali-T628
clCreateContextFromType(NULL,
CL_DEVICE_TYPE_ACCELERATOR) No devices found
in platform
clCreateContextFromType(NULL,
CL_DEVICE_TYPE_CUSTOM) No devices found in
platform
clCreateContextFromType(NULL,
CL_DEVICE_TYPE_ALL) Success (2)
Platform Name ARM Platform
Device Name Mali-T628
Device Name Mali-T628

ICD loader properties
ICD loader Name OpenCL ICD Loader
ICD loader Vendor OCL Icd free software
ICD loader Version 2.2.11
ICD loader Profile OpenCL 2.1
```

Es mucha información y muestra una plataforma con dos dispositivos OpenCL (ambos Mali-T628) compatibles con OpenCL 1.2.

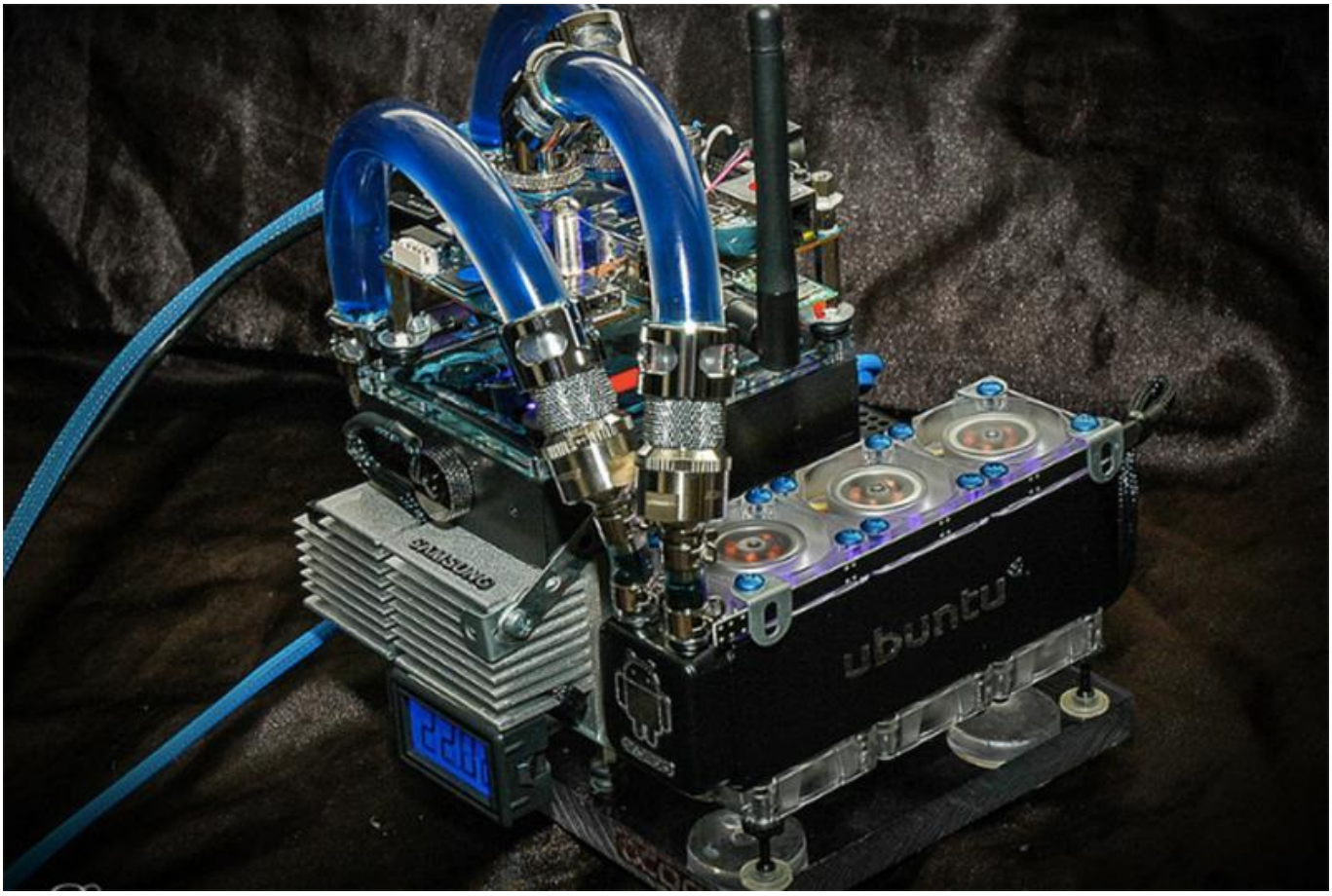
Esto es todo en esta pequeña guía de inicio. Si realmente quieres hacer algo con OpenCL, es hora de leer la documentación de Arm Compute Library y recurrir a otros recursos en la web.

Referencias

Este artículo procede de www.cnx-software.com. Para ver el artículo original o leer noticias similares, visita **https://www.cnx-software.com/2018/05/13/how-to-get-started-with-opencl-on-odroid-xu4-board-with-arm-mali-t628mp6-gpu/**.

Refrigeración líquida Parte 2 – Servidor

July 1, 2018 By e=MMC2 Mecaniqueo



Montar un ODROID con refrigeración líquida me llevo aproximadamente 5 semanas, con un promedio de 12-18 horas diarias dedicadas al proyecto. Pesa unos 3 kg, con un coste total de alrededor de unos 950$, incluyendo la placa, los accesorios, el hardware de refrigeración y los gastos de envío.

Cableado

Cablear las 18 conexiones para los 6 ventiladores requiere casi 2 días de trabajo para conseguir unos soportes perfectos. Son muchas horas cortando cables y volviendo a soldar conectores. El suministro de energía resultó ser un proyecto en sí mismo. Necesitaba suministrar 12v, 5v, 3.3v y 1.2v a los diferentes conectores diff y tenía que colocar 15 conexiones en un espacio lo más pequeño posible, al mismo tiempo que repartía los diferentes voltajes, además de colocar los condensadores y las resistencias.

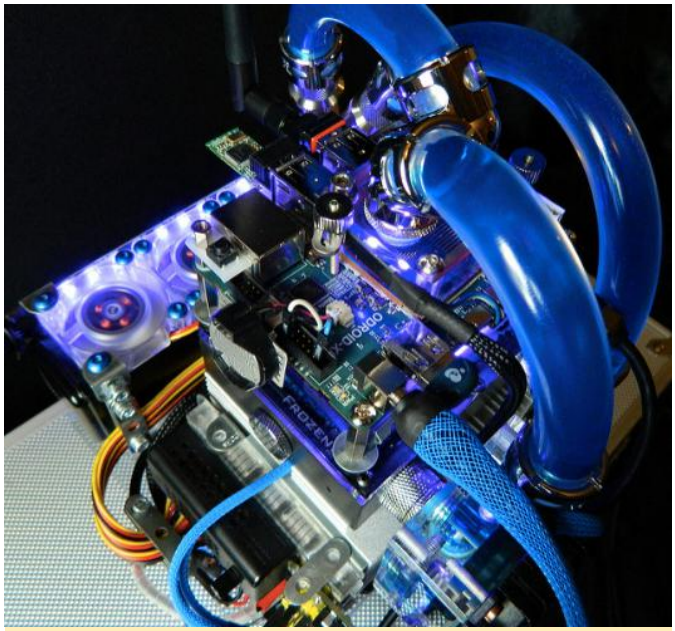


Figura 1 – Cableado del ODROID con refrigeración líquida

Placa de pruebas

Descubrí que usar una placa de pruebas permanentemente, en lugar de soldar una nueva PCB, en realidad era mejor, ya que cuenta con una base

muy buena y se puede cambiar rápidamente si fuera necesario. La entrada principal es una fuente de alimentación Arduino para la placa de pruebas, que me proporcionaría 5v/3.3v sin tener que usar reguladores de voltaje. Tan sólo he soldado nuevas derivaciones a la parte inferior de la placa de entrada en la toma principal 12v para evitar la placa y los rieles, lo cual proporcionaba a la placa los 12v necesarios para los ventiladores, la bomba y las barras LED.

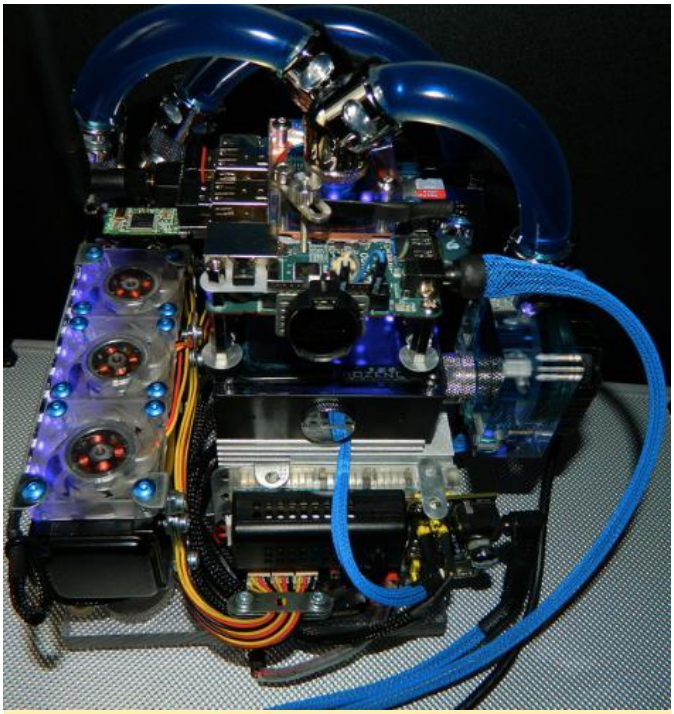


Figura 2 – Todas las bombas y la refrigeración están enfriando esta máquina a la perfección

Ventilador y bomba

Añadí un controlador de ventilador para moderar los ventiladores y las luces cuando quería que funcionase el sistema en modo silencioso y con luz tenua. La tira led UV y los ventiladores principales están conectados al controlador del ventilador, que reduce el voltaje en aproximadamente 6 voltios, lo cual hace que los ventiladores apenas giren permaneciendo en silencio. Utilicé un plexiglás de policarbonato a prueba de balas como base, luego coloqué un LED multicolor a través de un agujero que perforé e instalé un interruptor para seleccionar el color de la base. La bomba funciona a 12v constantes y parece perfecta para la presión necearía, así que no me molesté en implementar la bomba en el circuito del controlador del ventilador. Las instrucciones de la bomba indican que puede funcionar a partir de 6v, aunque decidí que el cuello de botella de mi radiador podría superarse llevando la bomba a su máxima potencia. La bomba es silenciosa y funciona bastante bien.

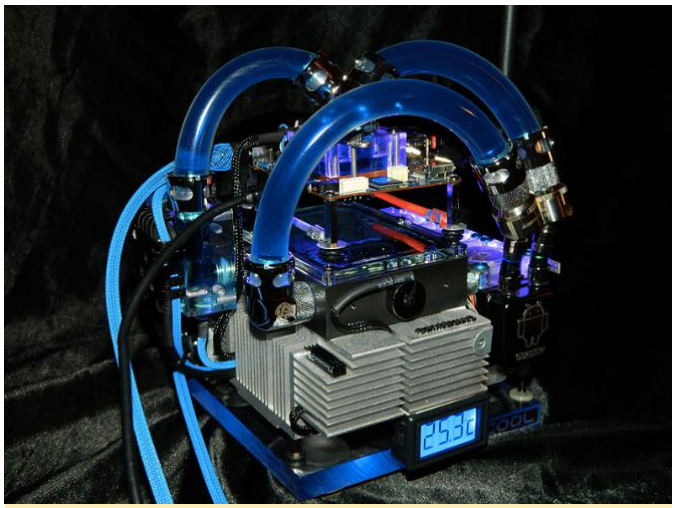


Figura 3 – Con esta configuración, este ODROID funciona perfectamente incluso bajo pruebas de extrema exigencia

Mejoras

Tengo algunas ideas para mejorar la configuración, aunque realmente no puedo costearme hacer ninguna otra durante algún tiempo. Creo que puedo enfriar el XU-E significativamente más, pero por ahora he demostrado que el concepto funciona y lo hace de un modo bastante consistente. Aunque el proyecto podría haber sido montado por mucho menos dinero, he querido usar piezas de cierta calidad para darle un aspecto más refinado.

Hardware

- Bomba 12V Cerámica DC-LT Alphacool + Plexi Top
- Radiador de Cobre Triple 40mm Alphacool NexXxoS XT45 con 6 ventiladores en una configuración push/pull.
- Ventiladores de 12V con unas dimensiones de 40mmx10mm a 6000 rpm y un empuje de 9,5 cfm
- Tubería de 3/8ID 5/8OD, aparte de la tubería de 1/4ID a 3/8OD para transformar y adaptarla al radiador. Solo se fabrican 2 tipos de radiadores de 40mm, y no hay opciones para nada excepto 1/4ID de este tipo, así que necesitaba usar un montón de accesorios adicionales para realizar la transformación
- Conectores Bitspower, Enzotech y Koolance
- Accesorios de compresión Monsoon Free Center para la tubería

- Pantalla LCD XSPC con sensor de temperatura para depósito
- Depósito del tanque FrozenQ Flex
- Refrigerante no conductor Fesser One UV Blue
- Almohadilla térmica Fujipoly Extreme Builder 11.0W/mk
- Tira de LED UV Darkside

El resto de las piezas eran juntas tóricas y componentes de iluminación, así como otros accesorios necesarios para la configuración de la fuente de alimentación. La base es un viejo disipador térmico de una CPU Macintosh que encontré. El resto del proyecto se sujeta con un viejo set Erector que desarmé. Las bases de goma fueron recuperadas de un mando Playstation.

Software

- Ubuntu 12.04, 13.10, 14.04 and Server
- Xubuntu
- Lubuntu
- Kali Linux
- Debian
- Arch
- openSuse
- Fedora
- Suzie
- Funtoo
- Abacus OS
- XBMC 13
- Android Jelly Bean 4.2.2