Pantalla Digole • Wall Monitor • U-Boot • Juegos Linux • Home Assistant

ODROID Magazine

Año Cuatro
Núm. #45
Sep 2017

Potencia Apilable

ODROID HC-1

Home Cloud

Swarm Docker:

Lleva el ODROID-C2 al siguiente nivel

ODROID-HC1 y ODROID-MC1: Computación en la nube y alto rendimiento para el hogar a un precio muy asequible

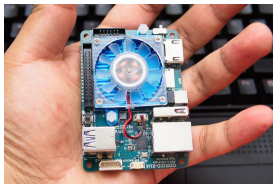
September 1, 2017

El ODROID-HC1 es un ordenador de placa reducida (SBC) que a su vez es una solución muy económica para montar un servidor de almacenamiento conectado en red (NAS), y el ODROID-MC1 es una simple solución para aquellos que necesiten un potente y económico clúster personal.

Mi ODROID-C2 Docker Swarm: Parte 1 – Características del Modo Swarm

September 1, 2017

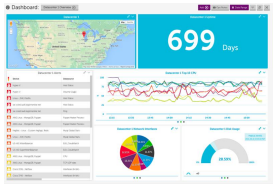
El modo Swarm proporciona funciones de gestión del clúster y de organización de los servicios, incluyendo la búsqueda y el escalado de servicios, entre otras cosas, utilizando para ello redes de superposición y un balanceador de carga interno, respectivamente.



U-Boot Convencial de ODROID-XU4

September 1, 2017

Hardkernel está trabajando en una nueva versión de U-Boot para el ODROID-XU4, con nuevas funciones y mejoras.



ODROID Wall Display: Utilizando un monitor LCD y un ODROID para mostrar información útil

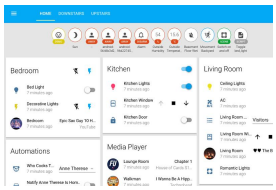
September 1, 2017

Un monitor de pared es una forma muy eficaz de presentar de modo pasivo un flujo constante de información. En lugar de comprar un marco digital que no sólo es caro, sino que también es pequeño y su funcionalidad está limitada, ¿Por qué no utilizar un monitor de ordenador o ▶️

Juegos Linux en ODROID: Fanboy Parte 2 – ¡Soy un Fanático de Sega!

September 1, 2017

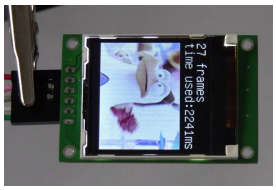
En la década de los 80 y 90, Nintendo y Sega luchaban por dominar el mercado de las consolas. Aunque ambos tenían productos similares, presentaban algunas diferencias importantes. Si comparamos sus primeros productos, el sistema de entretenimiento de Nintendo (NES) y Sega Master System (SMS) y Game Boy/Game Boy Color ▶️



Home Assistant: Personalización y Automatizaciones

September 1, 2017

Temas avanzados tratados en profundidad y relacionados con Home Assistant (HA), te permitirá maximizar el uso de HA y te ayudará con los experimentos.



Pantallas Serie Digole: Manegar la pantalla serie de Digole en los modos UART, I2C y SPI con un ODROID-C1+

September 1, 2017

Digole.com ofrece múltiples pantallas serie inteligentes que se controlan por medio de un completo conjunto de comandos exclusivos de alto nivel. Estos comandos facilitan el trazado de gráficos complejos y la visualización de imágenes y video, proporcionando una capa de abstracción que ayuda a exportar sus pantallas a un gran ▶️

Conociendo un ODROIDian: Ted Jack-Philippe Nivan (@TedJack)

September 1, 2017

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

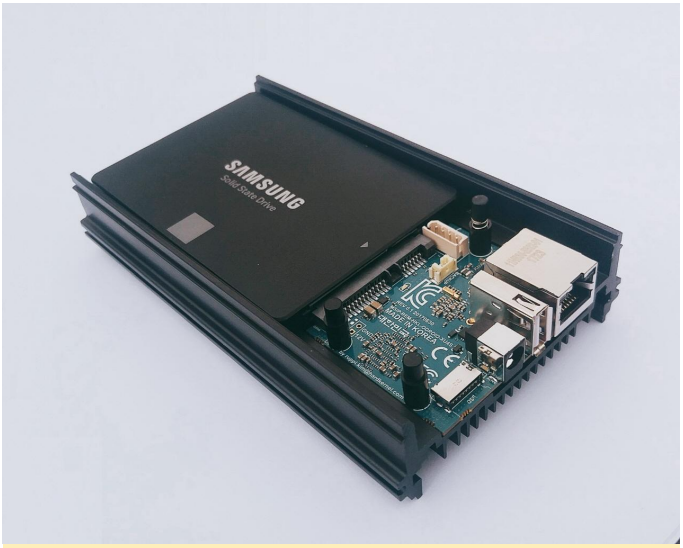
ODROID-HC1 y ODROID-MC1: Computación en la nube y alto rendimiento para el hogar a un precio muy asequible

September 1, 2017 By Rob Roy ODROID-HC1

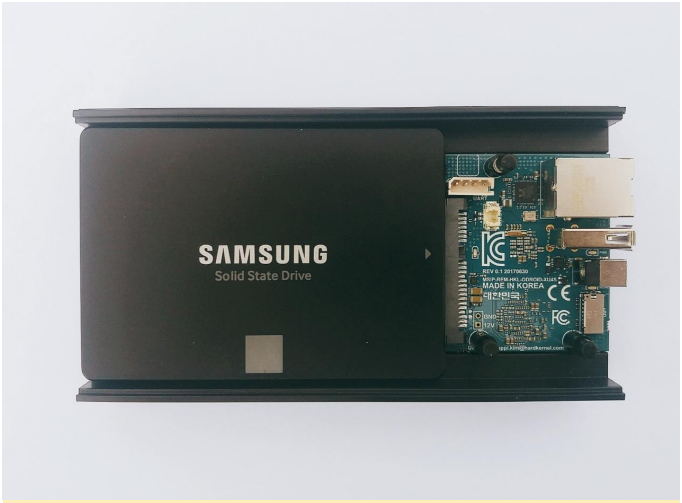
Mucha gente ha estado utilizando el ODROID-XU4 para aplicaciones en servidores, NAS, clúster, minería y tareas de compilación, todo ello gracias a su alto rendimiento y conectividad. Todas estas personas han seguido demandando soluciones más simples y económicas que deriven en versiones más reducidas de XU4, de modo que Hardkernel ha presentado un nuevo producto que ha sido creado para ser utilizado como un potente y económico servidor Home Cloud, llamado ODROID-HC1, que ya está disponible por 49$ en **http://bit.ly/2wjNToV**.

ODROID-HC1

El ODROID-HC1 es un ordenador de placa reducida (SBC) que a su vez es una solución muy económica para montar un servidor de almacenamiento conectado en red (NAS). Este servidor en la nube centraliza los datos y permite a los usuarios compartir y transmitir archivos multimedia a teléfonos, tablets y otros dispositivos dentro de una red. Es ideal para un usuario que tenga muchos dispositivos y quiera compartir archivos con la familia, y para los desarrolladores o grupos de trabajo. Puedes adaptar el ODROID-HC1 a tus necesidades específicas y, además, hay disponible un montón de software con una configuración mínima. El almacenamiento del servidor se puede personalizar usando un disco duro de gran capacidad o SSD. Dependiendo de tus necesidades, la estructura del HC1 está creada para ser apilable. El cuerpo de la estructura de metal está diseñado para almacenar un disco duro de 2,5 pulgadas, al mismo tiempo que ofrece una excelente disipación del calor.



El ODROID-HC1 es una elegante y económica solución de computación en la nube para el hogar



La interfaz SATA integrada te proporcionará un excelente rendimiento

El ODROID-HC1 es una asombrosa placa que se puede apilar para montar tu propio clúster personal.

El ODROID-HC1 está basado en la potente plataforma ODROID-XU4 y puede ejecutar Samba, FTP, NFS, SSH, NGINX, Apache, SQL, Docker, WordPress y cualquier otro software de servidor sin problema alguno, usando para ello distribuciones Linux completas como Ubuntu, Debian, Arch y OMV. Las distribuciones de sistemas operativos (SO) que están disponibles y listas para utilizarse se encuentran en la Wiki de Hardkernel en **http://bit.ly/2wjNsuI**. Cualquier sistema operativo que se ejecute en el XU4 es totalmente compatible con el HC1. Garantizaremos la producción de ODROID-HC1 hasta mediados de 2020, aunque esperamos continuar su producción mucho tiempo después.

Principales Características

- CPUs Samsung Exynos5422 Cortex-A15 2Ghz y Cortex-A7 Octa core
- 2Gbyte de RAM LPDDR3 PoP apiladas
- Puerto SATA para almacenamiento HDD/SSD de 2.5 pulgadas
- Puerto Gigabit Ethernet
- Puerto USB 2.0 Host
- Ranura para tarjeta micro-SD compatible con UHS-1 para el soporte de arranque
- Tamaño: aproximadamente 147 x 85 x 29 mm (incluyendo la estructura de aluminio para la refrigeración)
- Imágenes SO de servidor Linux basadas en el moderno Kernel 4.9 LTS

Dado que los deficientes cables USB y los imperfectos chipsets de comunicación USB-a-SATA provocan que los usuarios tengan dificultades debido a problemas de tolerancia física/eléctrica, así como problemas de compatibilidad con el controlador, el ODROID-HC1 cuenta con un conector SATA integrado en la PCB con un controlador SATA probado a fondo. Para disminuir el coste, el tamaño de la placa se ha reducido gracias al valor de la PCB de 10 capas, lo cual ha tenido como resultado la retirada de algunas características como la salida HDMI, el conector eMMC, los puertos USB 3.0, el botón de encendido y el interruptor deslizante.

Se puede instalar cualquier tipo de almacenamiento SATA HDD/SSD de 2,5 pulgadas, incluyendo las unidades de 7 mm, 9,5 mm, 12 mm y 15 mm de grosor. Los HDDs Seagate Barracuda 2TB/5TB, el HDD Samsung de 500GB y el SSD de 256GB, los HDDs Western Digital de 500GB y 1TB, el HDD HGST de 1TB

y otras unidades de almacenamiento han sido probadas con funciones S.M.A.R.T y UAS.

El ODROID-HC1 admite muchos y diferentes tipos de discos duros

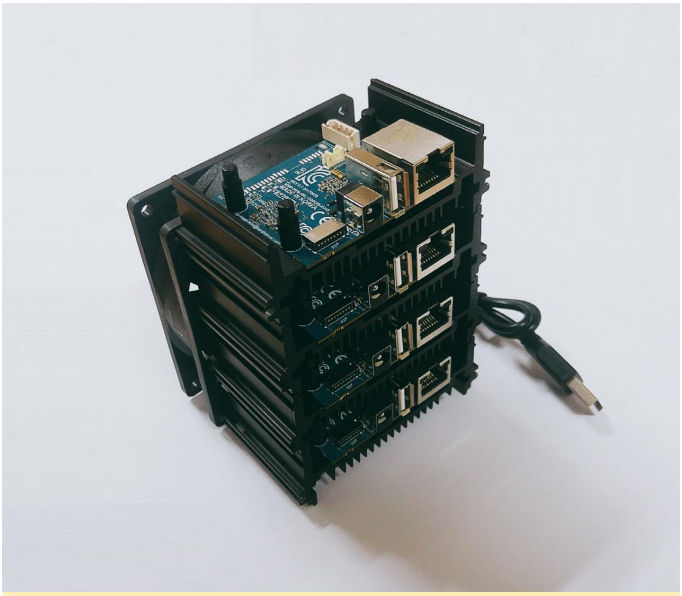
Tu servidor compartido puede ser modular, compacto y elegante

Para conocer más detalles y ver una demostración del funcionamiento del ODROID-HC1, por favor, echa un vistazo al video https://youtu.be/t-L99pUANaA.

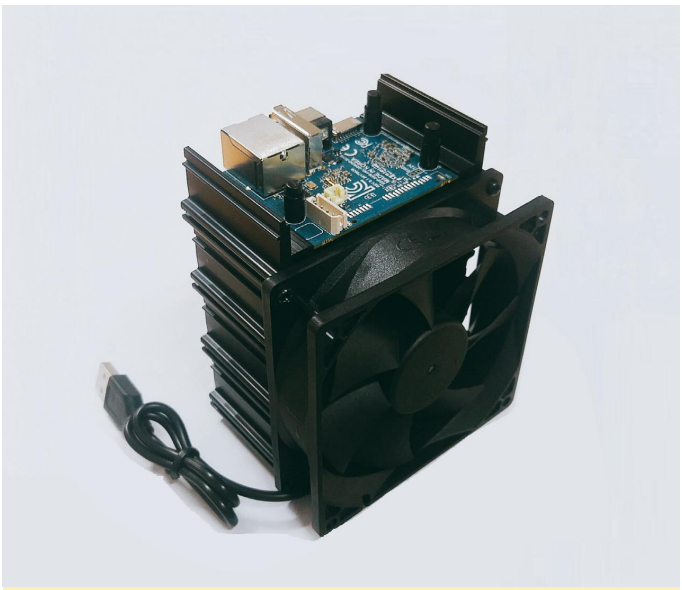
ODROID-MC1

El ODROID-MC1, que significa “Mi Clúster”, es una simple solución para aquellos que necesiten un potente y económico clúster personal. Es similar al ODROID-HC1, pero excluye la interfaz SATA y añade un gran ventilador de refrigeración para pesadas operaciones informáticas. Se prevé que éste disponible para septiembre de 2017 por 200$

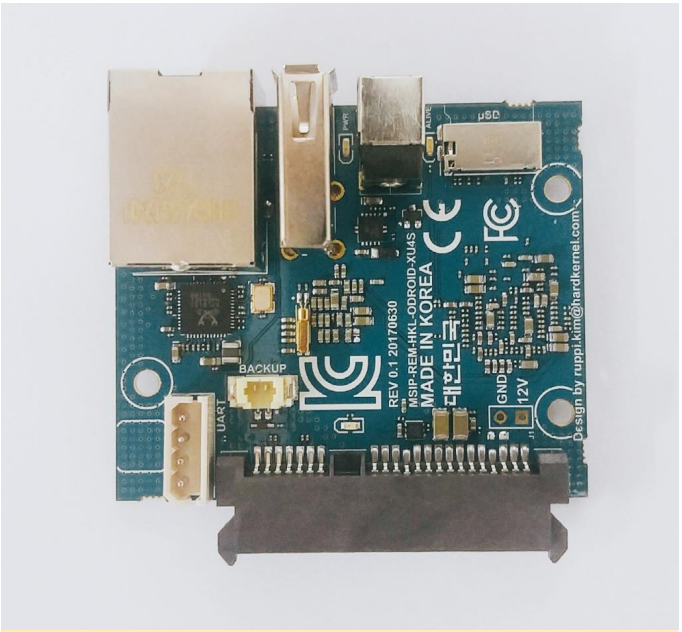
Hardkernel pasó varios días montando este gran clúster de 200 unidades ODROID-MC1 con una CPU de 1600 núcleos y 400 GB de RAM



Una pila de 4 unidades ODROID-MC1, crea un clúster con una CPU de 32 núcleos y 8GB de RAM.



La refrigeración de 4 ODROIDs con un único ventilador supone una gran reducción de los costes



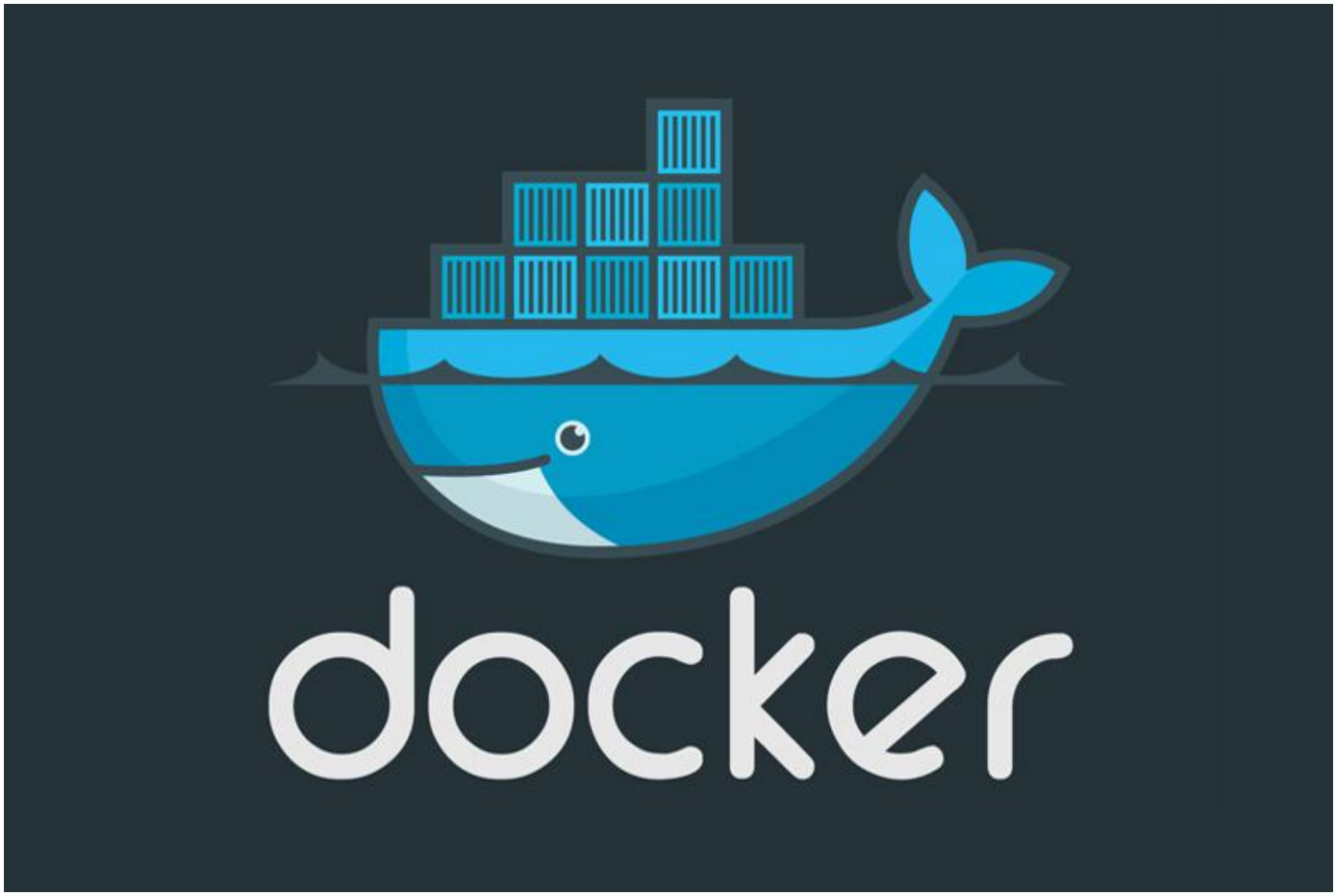
El SoC y PMIC se pegan con resina epoxi para aumentar su fiabilidad

Especificaciones técnicas

Esquemas **http://bit.ly/2vLy7zH** Diseños mecánicos de la PCB (formato AutoCAD) **http://bit.ly/2el0VZm** Información del producto **http://bit.ly/2xzBXho**

Mi ODROID-C2 Docker Swarm: Parte 1 – Características del Modo Swarm

September 1, 2017 By Andy Yuen Docker, Linux, ODROID-C2



Docker introdujo el modo swarm en la versión 1.12.x con el fin de hacer posible la implementación de contenedores en múltiples hosts docker. El modo Swarm proporciona funciones de gestión del clúster y de organización de los servicios, incluyendo la búsqueda y el escalado de servicios, entre otras cosas, utilizando para ello redes de superposición y un balanceador de carga interno, respectivamente. Estas son funcionalidades claves para las empresas, ya que existe un límite en el número de contenedores que se pueden implementar en un único host docker. Si deseas conocer más afondo la arquitectura de alto nivel del modo swarm, echa un vistazo a mi anterior artículo publicado en la edición de noviembre de 2016 de ODROID Magazine en **http://bit.ly/2wiTVXM**.

Hace varios meses, cuando estaba experimentando con el modo swarm de Docker en mi clúster ODROID-C2 de cinco placas. Era capaz de iniciar varios contenedores Docker en multiples hosts docker, pero ni la red de superposición, el sistema de enrutamiento, ni el balanceo de carga funcionaban en modo swarm. Llegué a usar múltiples versiones de docker (1.12.x y 1.13.x) compiladas en mi ODROID-C2 sin obtener resultados. También intenté ejecutar Kubernetes en mi clúster ODROID-C2. Una vez más la parte de la gestión de redes de Kubernetes no funcionaba. Sospechaba que al kernel le faltaban algunos módulos que serían necesarios para Docker y Kubernetes. Debido a esto, decidí abandonar mis experimentos hasta ahora. Lo que reavivó mi pasión por lograr hacer funcionar el modo swarm de Docker fue el hecho de considerar un hardware que no había utilizado con anterioridad: la pantalla ODROID VU7 multi-touch y la carcasa VuShell para VU7.

Ensamblé la pantalla VU7 y un ODROID-C1 + con la carcasa VuShell. Luego pensé, ¿por qué no colocar mi

clúster ODROID-C2 también? En la figura 1 puedes ver una pantalla que muestra un teclado virtual. Todos los ordenadores de placa reducida ODROID están conectados entre sí con un switch Ethernet Gigabit de 8 puertos, y también hay un SSD dentro de la carcasa VuShell. La caja de cartón ODROID alberga la fuente de alimentación. El diminuto router inalámbrico utiliza el Sistema de Distribución Inalámbrica, WDS, para conectarse a mi router principal y proporcionar acceso a Internet vía Ethernet a todos los ODROID que hay dentro de la carcasa VuShell, ya que éstos no tienen WiFi integrado.

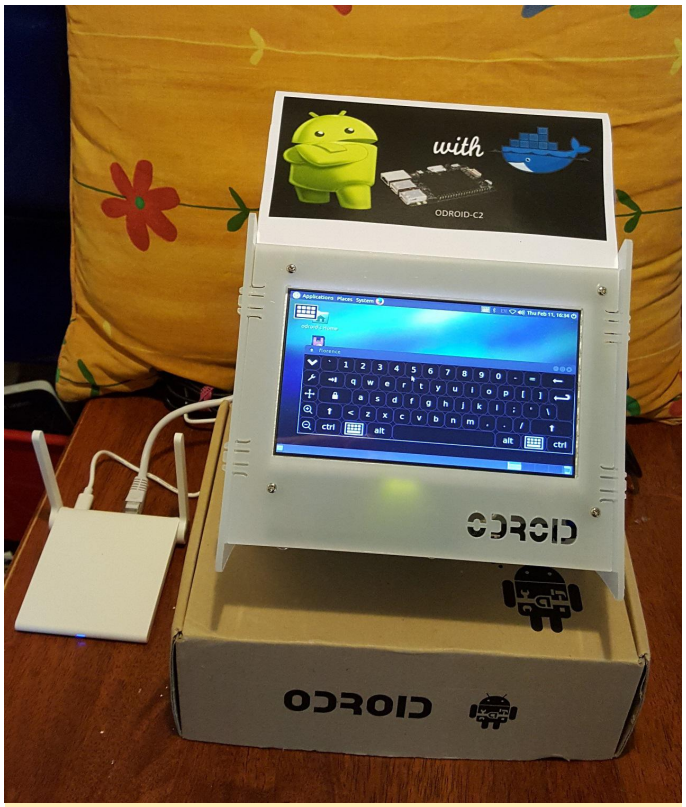


Figura 1 – Un cluster Docker Swam que utiliza el shell ODROID-VU como carcasa

Sistema operativo Ubuntu 16.04 de Hardkernel

Tenía la sospecha de que el modo swarm de Docker no llego a funcionar en mis anteriores pruebas porque faltaba o eran incompatibles algunos módulos del kernel en el sistema operativo. Así que opte por cambiar a otro sistema operativo. Observé que Hardkernel había lanzado recientemente Ubuntu 16.04 (v2.3) para el ODROID-C2, así que lo probé. La versión anterior del sistema operativo Ubuntu de Hardkernel que probé meses antes era inestable, pero la versión actual funcionaba sin ningún problema. ¡Estaba contento y me dije a mi mismo que quizás esta vez podría funcionar!

Para facilitar las cosas, instalé y configuré los siguientes paquetes:

- parallel-ssh en el gestor docker para permitirme enviar comandos una vez desde el gestor docker y que sean ejecutados en todos los nodos
- nfs-kernel-server en el gestor y nfs-common en todos los nodos
- curl en el gestor para hacer pruebas
- dnsutils en todos los nodos

También he generado claves SSH para los usuarios “odroid” y “root” en todos los miembros del clúster, para que puedan comunicarse entre sí vía SSH sin una contraseña.

Reiniciar el Modo Swarm de Docker

Instalé docker.io usando apt-get y realicé una prueba rápida “docker run” usando mi imagen httpd, y funcionó. Quería probar el modo swarm a continuación para ver si funcionaba con el nuevo sistema operativo. Aquí tienes una captura de pantalla de las versiones del software que han sido utilizadas. Resulta interesante ver que la distribución de Ubuntu de Hardkernel viene con zram preinstalado para swap, lo cual es muy útil.

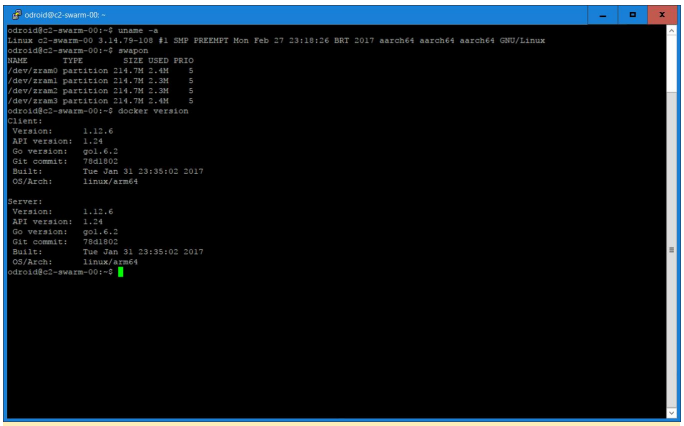


Figura 2 – Docker mostrando las versiones de todo el software actual

Creando un Swarm

Los nombres de host y las direcciones IP estáticas de mis hosts swarm son::

- c2-swarm-00 – 192.168.1.100 (gestor)
- c2-swarm-01 – 192.168.1.101 (nodo 1)

- c2-swarm-02 – 192.168.1.102 (nodo 2)
- c2-swarm-03 – 192.168.1.103 (nodo 3)
- c2-swarm-04 – 192.168.1.104 (nodo 4)

Únicamente el c2-swarm-00 tiene conectada una unidad SSD, aunque el sistema de archivos se comparte por NFS.

Un nodo es un host docker que forma parte de un swarm. Un nodo gestor es al que envías una definición de servicio y éste programa el servicio para que se ejecute como tarea en los nodos de trabajo. Los nodos de trabajo reciben y ejecutan las tareas programadas por un nodo gestor. Un nodo gestor, por defecto, también es un nodo de trabajo a menos que esté configurado expresamente para no ejecutar tareas. Se pueden configurar múltiples nodos maestros y de trabajo en un Swarm para proporcionar Alta Disponibilidad (HA). Para iniciar el modo Swarm, utiliza el siguiente comando en el gestor:

```
$ docker swarm init --advertise-addr
192.168.1.100
```

que devuelve:

```
swarm initialized: current mode
(8jw6y313hmt3vfa1me1dinro) is now a manager
```

Para añadir un nodo de trabajo a este Swarm, ejecuta el siguiente comando en cada nodo:

```
$ docker swarm join --token SWMTKN-1-
2gvqzfx48uw8zcokwl5033iwdel2rl9n96lc0wj1qso7lr
ztub-aokk5xcm5v7c4usmeswsgg1k
192.168.1.100:2377
```

Para hacer que el resto de nodos se unan al clúster, ejecuta el anterior comando “docker swarm join” en cada nodo. Esto se puede hacer usando parallel-ssh, que te permitirá emitir el comando una vez desde el gestor, y luego ejecutarlo en cada nodo. La siguiente imagen muestra una captura de pantalla tras ejecutar el comando “docker ps” usando parallel-ssh, lo que significa que el modo swarm de Docker está funcionando.

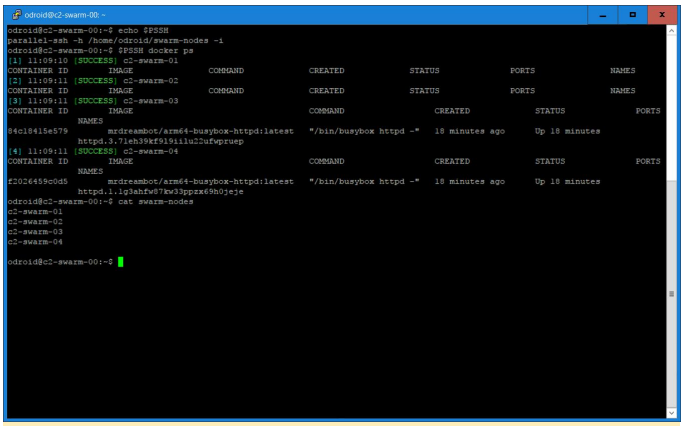


Figura 3 – El resultado del comando “docker ps” que muestra todos los nodos

Un incordio que descubrí en el modo swarm de Docker es que después de apagar todos los nodos y encenderlos de nuevo, todos los nodos aparecen “Active”, pero ” Down”. Te das cuenta de esto cuando utilizas “docker node ls” para conocer el estado de los nodos. Puesto que los nodos no están disponibles, todos los servicios se ejecutarán en el gestor. La solución para por ejecutar “systemctl restart docker” en cada nodo. Esto cambiará su estado de “Down” a “Ready”, y todo volverá a funcionar de nuevo. La herramienta parallel-ssh es una forma muy cómoda de hacerlo, ya que lo único que tienes que hacer es ejecutar una vez el comando en tu nodo gestor.

Ejecutar los Servicios HTTPD y Docker Swarm Visualizer

Para ayudar a visualizar lo que sucede en el Swarm, he desarrollado la imagen “Docker Swarm Visualizer” basada en Docker Samples en Github. La he dejado en el hub docker en **http://dockr.ly/2ipXzcL**, para que cualquiera pueda usarla. El nombre de la imagen es “mrdreambot/arm64-docker-swarm-visualizer”, disponible en **http://bit.ly/2xqSaV4**. Después la implemente como un servicio ejecutando el siguiente comando desde el gestor:

```
$ docker service create --name=dsv --
publish=8080:8080/tcp --
constraint=node.role==manager --
mount=type=bind,src=/var/run/docker.sock,dst=/
var/run/docker.sock mrdreambot/arm64-docker-
swarm-visualizer
```

A continuación, apunté el navegador hacia el nodo maestro en http://192.168.1.100:8080, aunque

también funciona si diriges el navegador hacia cualquiera de los nodos del swarm. Se pueden observar los cambios reportados por el visualizador al implementar el servicio httpd:

```
$ docker network create --driver overlay home-net
$ docker service create --replicas 3 --network home-net --name httpd -p 80:80 mrdreambot/arm64-busy-box-httpd
```

La figura 4 muestra el resultado de la línea de comandos a la hora de listar los servicios. La Figura 5 es una captura de pantalla del Docker Swarm Visualizer que muestra los nodos en los que se ejecutan las réplicas del servicio, lo cual pone de manifiesto el modelo de servicio factual utilizado por el modo swarm.

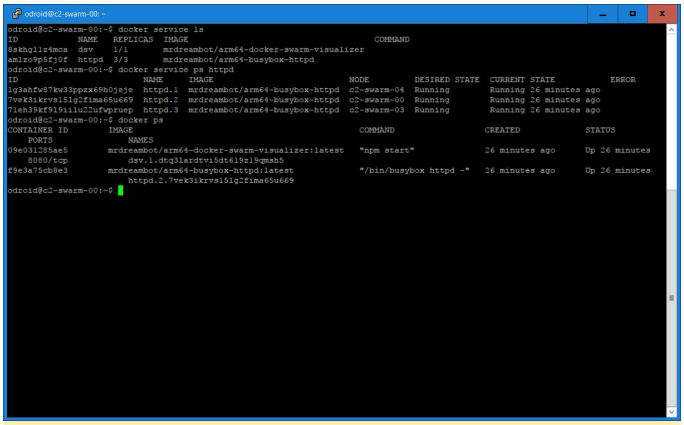


Figura 4 – Resultado de la línea de comandos con el listado de servicios

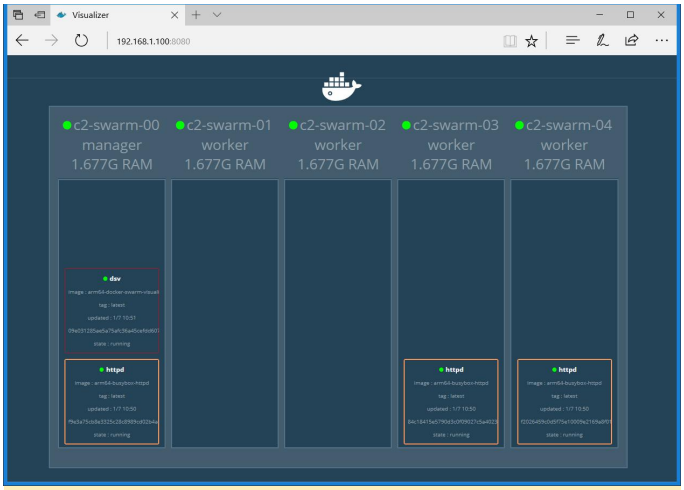


Figura 5 – Docker Swarm Visualizer

Sistema de enrutamiento, Balanceo de carga y auto-regeneración

El sistema de enrutamiento en el swarm permite que una solicitud llegue a un servicio incluso cuando el servicio no se esté ejecutando en el nodo en el cual se ha recibido la solicitud. Esto significa que, aunque el servicio httpd se ejecuta en c2-swarm-00, c2-swarm-03 y c2-swarm-04, se puede apuntar al navegador hacia cualquiera de los 5 nodos y obtener una respuesta con la imagen ODROID-Docker. Este fue el comportamiento que observé.

Figura 6 – Ejemplo de Balanceo de carga usando 10.255.0.9

Además de ofrecer un sistema de enrutamiento, swarm también permite el balanceo de carga. Para probar esta función, hice conexión al gestor varias veces usando mi navegador, en el servicio httpd usando la dirección **http://192.168.1.100/cgi-bin/lbtest**. Observe que los nombres de host (Id de contenedor) y las direcciones IP son diferentes en las dos capturas de pantalla.

Figura 7 – LEjemplo de balanceo de carga usando 10.255.0.10

Las pruebas fueron repetidas usando el comando curl:

```
$ curl http://192.168.1.100/cgi-bin/lbtest
```

Aquí tienes una captura de pantalla del resultado de los comandos curl que confirma, de nuevo, que cada solicitud ha sido dirigida a un nodo diferente:

Figura 8 – Balanceo de carga entre nodos

Para probar la auto-regeneración, decidí apagar el c2-swarm-04, se puede ver tanto desde el visualizador como desde la línea de comandos que otro contenedor httpd ha sido derivado a c2-swarm-02 para reemplazar el de c2 -swarm-04. Esto ocurre porque cuando iniciamos el servicio, especificamos “réplica = 3”. Esto significa que el swarm de Docker mantendrá el número réplicas deseado, en ente caso 3. Esto suele denominarse reconciliación del estado deseado.

Figura 9 – Recuperación del servicio

Figura 10 – Recuperación del servicio httpd

Después apagué el resto de nodos y dejé funcionando únicamente el nodo gestor, el Visualizer muestra el resultado en la figura 11.

Figura 11 – Recuperación del Servicio httpd – 1 nodo

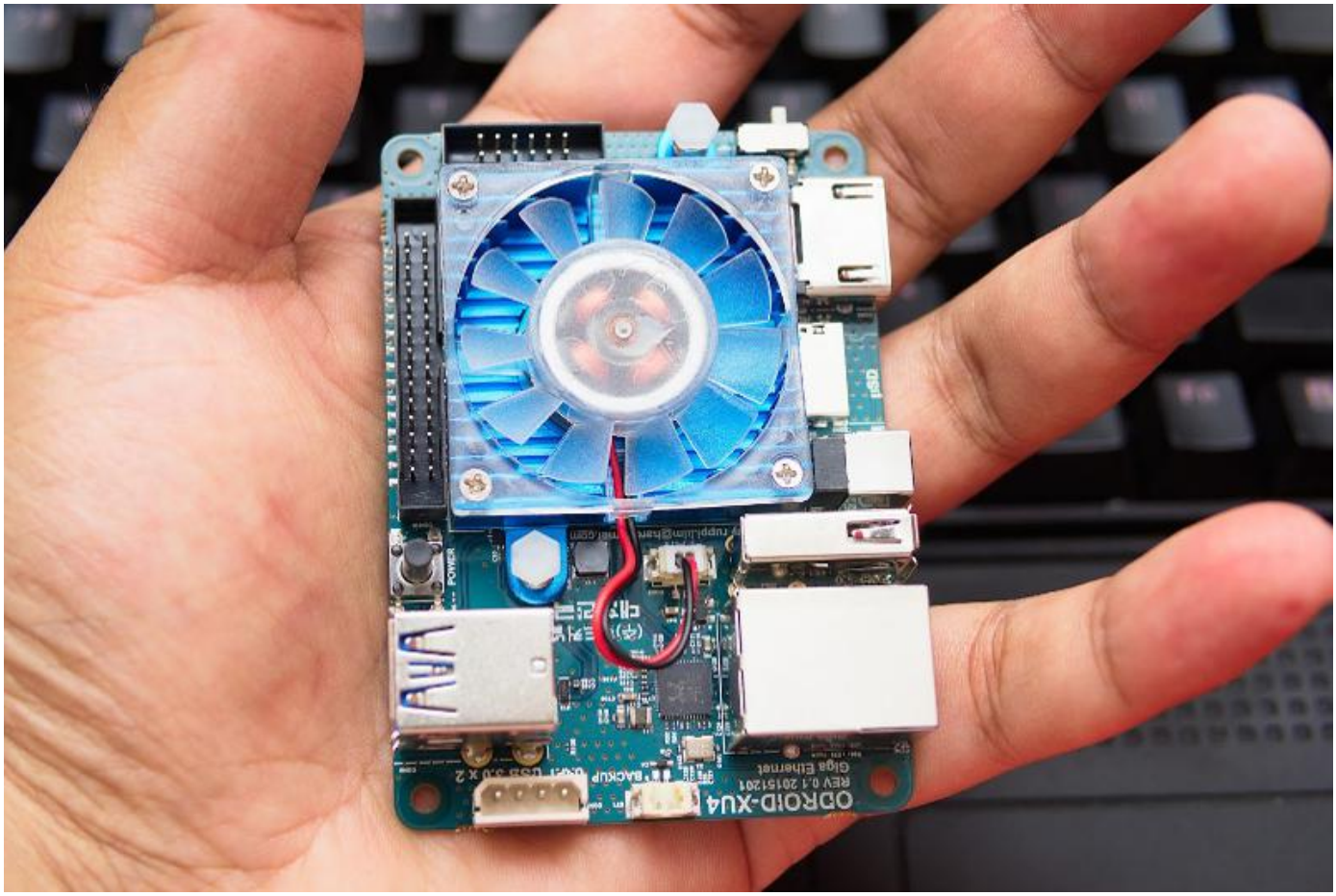
¡Todo funcionaba como era de esperar!

Conclusión

La nueva versión de Ubuntu 16.04 de Hardkernel marcó realmente la diferencia. El modo swarm de Docker ahora funciona por completo en mi clúster ODROID-C2. En la siguiente instalación, actualizaré Docker a la versión 1.13.x para experimentar con la función “docker stack deploy”, nueva en v.1.13.x. Una pila es un conjunto de servicios que componen una aplicación. Despliega automáticamente varios servicios que están vinculados entre sí, eliminando la necesidad de definir cada uno de ellos por separado. En otras palabras, es docker-compose en modo swarm que gestiona la planificación de los servicios. También describiremos el uso de una red de superposición para la búsqueda de servicios.

U-Boot Convencial de ODROID-XU4

September 1, 2017 By Rob Roy Linux, ODROID-XU4



Hardkernel está trabajando en una nueva versión de U-Boot para el ODROID-XU4, con las siguientes funciones:

- Habilita el modo HYP para la virtualización KVM con Kernel 4.9 LTS
- Permite que el dispositivo Ethernet soporte el arranque en remoto TFTP/PXE
- Arranque desde nuevos chipsets eMMC
- Todos los comandos de fatload y ext4load están disponibles de forma nativa
- Muchas más novedades

La nueva versión de U-Boot está disponible en el último paquete de actualización para Linux, los usuarios de Android 4.4 y Android 7.1 recibirán pronto una actualización, que incluirá el protocolo fastboot. El código fuente está disponible en http://bit.ly/2xrM7R3.

A continuación, se muestra un ejemplo del registro de arranque:

```
u-boot booting log from the serial console
output.
U-Boot 2017.05-12186-gf98cc91-dirty (Aug 08
2017 - 12:16:58 +0900) for ODROID XU4

CPU:   Exynos5422 @ 800 MHz
Model: Odroid XU4 based on EXYNOS5422
Board: Odroid XU4 based on EXYNOS5422
Type:  xu4
DRAM:  2 GiB
MMC:   EXYNOS DWMMC: 0, EXYNOS DWMMC: 1
MMC Device 0 (eMMC): 14.7 GiB
Info eMMC rst_n_func status = enabled
Card did not respond to voltage select!
mmc_init: -95, time 11
*** Warning - bad CRC, using default
environment

In:    serial
Out:   serial
```

```
Err:   serial
Net:   No ethernet found.
Press quickly 'Enter' twice to stop autoboot:
0
reading boot.ini
9088 bytes read in 4 ms (2.2 MiB/s)
cfgload: applying boot.ini...
cfgload: setenv initrd_high "0xffffffff"
cfgload: setenv fdt_high "0xffffffff"
cfgload: setenv macaddr "00:1e:06:61:7a:39"
cfgload: setenv vout "hdmi"
cfgload: setenv cecenable "false" # false or
true
cfgload: setenv disable_vu7 "false" # false
cfgload: setenv governor "performance"
cfgload: setenv ddr_freq 825
cfgload: setenv external_watchdog "false"
cfgload: setenv external_watchdog_debounce "3"
cfgload: setenv HPD "true"
cfgload: setenv bootrootfs "console=tty1
console=ttySAC2,115200n8 root=UUID=e139ce78-
9841-40fe-8823-96a304a09859 rootwait ro
fsck.repair=yes net.ifnames=0"
cfgload: fatload mmc 0:1 0x40008000 zImage
reading zImage
4793144 bytes read in 135 ms (33.9 MiB/s)
cfgload: fatload mmc 0:1 0x42000000 uInitrd
reading uInitrd
5327028 bytes read in 143 ms (35.5 MiB/s)
cfgload: if test "${board_name}" = "xu4"; then
fatload mmc 0:1 0x44000000 exynos5422-
odroidxu4.dtb; setenv fdtloaded "true"; fi
reading exynos5422-odroidxu4.dtb
61570 bytes read in 9 ms (6.5 MiB/s)
cfgload: if test "${board_name}" = "xu3"; then
fatload mmc 0:1 0x44000000 exynos5422-
odroidxu3.dtb; setenv fdtloaded "true"; fi
cfgload: if test "${board_name}" = "xu3l";
then fatload mmc 0:1 0x44000000 exynos5422-
odroidxu3-lite.dtb; setenv fdtloaded "true";
fi
cfgload: if test "${fdtloaded}" != "true";
then fatload mmc 0:1 0x44000000 exynos5422-
odroidxu4.dtb; fi
cfgload: fdt addr 0x44000000
cfgload: setenv hdmi_phy_control "HPD=${HPD}
vout=${vout}"
cfgload: if test "${cecenable}" = "false";
then fdt rm /cec@101B0000; fi
cfgload: if test "${disable_vu7}" = "false";
then setenv hid_quirks
"usbhid.quirks=0x0eef:0x0005:0x0004"; fi
cfgload: if test "${external_watchdog}" =
"true"; then setenv external_watchdog
"external_watchdog=${external_watchdog}
external_watchdog_debounce=${external_watchdog
_debounce}"; fi
cfgload: setenv bootargs "${bootrootfs}
${videoconfig} ${hdmi_phy_control}
${hid_quirks} smsc95xx.macaddr=${macaddr}
${external_watchdog} governor=${governor}"
cfgload: bootz 0x40008000 0x42000000
0x44000000
Kernel image @ 0x40008000 [ 0x000000 -
0x492338 ]
## Loading init Ramdisk from Legacy Image at
42000000 ...
   Image Name:   uInitrd
   Image Type:   ARM Linux RAMDisk Image
(uncompressed)
   Data Size:    5326964 Bytes = 5.1 MiB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 44000000
   Booting using the fdt blob at 0x44000000
   Using Device Tree in place at 44000000, end
44012081

Starting kernel ...
```

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

ODROID Wall Display: Utilizando un monitor LCD y un ODROID para mostrar información útil

September 1, 2017 By William Green ODROID-C0

Soy un técnico en prácticas en **ameriDroid.com**, que ayuda a resolver cuestiones técnicas, he estado trabajando en proyectos muy divertidos e interesantes relacionados con ODROIDs y su correspondiente electrónica. Esta es una reseña de un reciente proyecto que acabo de terminar y que me ha supuso la creación de un monitor de pared.

Un monitor de pared es una forma muy eficaz de presentar de modo pasivo un flujo constante de información. En lugar de comprar un marco digital que no sólo es caro, sino que también es pequeño y su funcionalidad está limitada, ¿Por qué no utilizar un monitor de ordenador o una pantalla de TV junto con un ODROID para mostrar fotos, información meteorológica, noticias en formato RSS y otros medios?

Antes de empezar a montar el proyecto, necesitarás los siguientes elementos:

- Un ODROID (nosotros usamos un ODROID-C0, pero puede funcionar igual de bien con otros modelos)
- Un módulo Wifi USB o un cable Ethernet, si la placa lo permite
- Un Monitor de ordenador
- Un adaptador de corriente
- Un cable de salida de video
- Un ratón y teclado, para la configuración
- Un sistema para montar el ODROID sobre o cerca del monitor

La cinta de montaje de doble cara puede ser una forma muy cómoda de colocar el ODROID detrás del monitor. El ODROID-C0 es una buena opción por su tamaño compacto y coste mínimo. Se necesita una distribución de Linux, se recomienda usar Lubuntu de Hardkernel para el C0. El teclado y el ratón son necesarios para crear el proyecto, aunque el monitor de pared puede funcionar sin el teclado o el ratón

una vez que el proyecto esté configurado. El ODROID puede ser controlado en cualquier momento usando SSH si fuera necesario.

Tras recopilar los materiales necesarios, sigue los pasos que se muestran a continuación para montar el monitor. Dado que los ODROIDs, al igual que la mayoría del resto de ordenadores de placa reducida, carece de una interfaz BIOS, en su lugar utilizan un archivo especial donde se almacenan diversos parámetros, como la resolución de pantalla. Para hacer que el ODROID envié la señal al monitor, averigua primero la resolución de pantalla del monitor. Edita el archivo boot.ini de la partición de arranque des-comentando la resolución especifica de correspondiente monitor. Enciende el ODROID y conéctalo a una red inalámbrica disponible. A continuación, actualiza el software con el siguiente comando::

```
$ sudo apt update
```

Después de reiniciar el ODROID, el siguiente paso es instalar el software necesario. Navega hasta el terminal e instala el programa unclutter:

```
$ sudo apt install unclutter
```

Unclutter ocultará el cursor cuando no esté en uso. DAKboard es un sitio web que proporciona contenido para mostrar en grandes pantallas. Navega hasta dakboard.com, registra una cuenta y personaliza la página con noticias RSS, fotos, clima y otra información. En las Opciones de DAKboard, dentro de la Configuración de la cuenta, hay una URL privada que debe copiar. Abre de nuevo el terminal y escribe lo siguiente::

```
$ sudo vi
~/.config/xlsession/Lubuntu/autostart
```

Pulsa la tecla o y escriba “firefox -url” y, a continuación, pega la dirección URL privada de DAKboard. En vi, puedes pegar haciendo clic en los botones izquierdo y derecho del ratón al mismo tiempo. Pulsa la tecla de escape, luego escribe “: wq”. Abre Firefox y navegue hasta **https://mzl.la/2wpotqS** para instalar la extensión R-kiosk para Firefox. Una vez instalada la extensión, desconecta la alimentación del ODROID. Monta el ODROID cerca del monitor y coloca y organiza los cables de alimentación y de vídeo. Enciende el ODROID y verifica que todo funciona correctamente antes de montar el monitor en la pared.

El ODROID debería mostrar la información que has configurado en DAKboard. Inicia sesión en DAKboard en otro equipo para editar la página en cualquier momento. La página ahora debería proporcionar información en tiempo real sobre el clima, fotos y noticias RSS.

Juegos Linux en ODROID: Fanboy Parte 2 – ¡Soy un Fanático de Sega!

September 1, 2017 By Tobias Schaaf Juegos, Linux

En el número del mes pasado, comenté que no soy especialmente un fanático de Nintendo. Por definición, esto debería significar que entonces soy un fanático de Sega, ¿verdad? Quiero decir que esa sería la típica suposición: eres fanático de Nintendo o de Sega. Dado que dije que no soy de Nintendo, significa pues que soy más bien de Sega, ¿verdad?. Supongo que es cierto, aunque vamos a analizar con más detenimiento esta cuestión.

Hardware de Sega

En la década de los 80 y 90, Nintendo y Sega luchaban por dominar el mercado de las consolas. Aunque ambos tenían productos similares, presentaban algunas diferencias importantes. Si comparamos sus primeros productos, el sistema de entretenimiento de Nintendo (NES) y Sega Master System (SMS) y Game Boy/Game Boy Color (GB / GBC) y Game Gear (GG), parecía que Sega siempre iba por delante en cuanto a tecnología.

La SMS tenía un procesador más rápido, más memoria RAM, más colores, y podía mostrar más colores al mismo tiempo. En general, parecía ser la consola más potente. Sin embargo, si comparamos las ventas, NES supera a la SMS.

Cuando aún estaba en la escuela, un amigo mío poseía una SMS. A menudo se la traía a la escuela para que pudiéramos jugar después de clase hasta que nuestros padres nos recogiesen. Por aquel entonces, mayoritariamente jugamos al Alex Kidd en Miracle World, que formaba parte del sistema. Esto significaba que si comprabas una SMS ya tenías tu primer juego y no necesitabas comprar uno aparte.

Figura 1 – Alex Kidd en Miracle World sobre el ODROID con shader 2xsal-level2-crt

Los gráficos y la jugabilidad me dejaron perplejo por aquel entonces. Eran bastante mejores que lo que había visto en la NES de mi tío y, además, disponías de más juegos a los que nos encantaba jugar, como Space Harrier 3D.

Más adelante, tuve mi propia SMS. Pensé que finalmente sería capaz de terminar Alex Kidd, pero resultó que no todas las consolas venían con el mismo juego integrado. La mía venía con Hang-On, que era un curioso juego de carreras de motos, aunque no me gustaba demasiado por aquel entonces.

Alguien más de la escuela tenía una Game Gear, y ¡caray!, era enorme. Se trataba de una consola de juegos portátil. A diferencia del Game Boy, ésta era más realista, con color y todo. Podrías incluso jugar a tus juegos de la SMS en la Game Gear si contabas con un adaptador, e incluso había un accesorio que te permitía ver la TV con una antena integrada. ¿Era tan buena como parecía?

La Game Gear tenía prácticamente las mismas especificaciones técnicas que la SMS. La CPU era la misma, simplemente la frecuencia del reloj era ligeramente inferior, y era capaz de utilizar incluso más colores que la SMS. Por lo general, se trataba de un dispositivo portátil bastante impresionante, aunque su precio era muy elevado. El dispositivo en sí no era barato y las pilas se gastaban muy rápido. Apenas disponías de un tiempo de juego de unas 3-5 horas con 6 pilas AA. Sin embargo, viendo lo que la GG era capaz de hacer, ésta siempre me impidió realmente desear una Game Boy. De modo que sí, incluso en mis comienzos siempre he preferido las consolas de Sega sobre la de Nintendo. Tal vez por razones equivocadas, pero era un niño y no conocía nada mejor.

Echando la vista atrás, está claro por qué Nintendo estaba por delante en aquellos tiempos. El número de juegos de alta calidad en NES era bastante mayor que en SMS. Esto se debía en gran parte a las restricciones que Nintendo puso a sus desarrolladores de software externos, que les impedía producir juegos para los competidores de Nintendo si querían que sus juegos operaran en las consolas de Nintendo.

Pase por alto la cuarta y quinta generación de consolas de Sega por el simple hecho de que en ese momento realmente no jugaba a la consola. Me perdí todo lo bueno y malo de la Genesis/Mega Drive, Sega CD, Sega 32X y Sega Saturn. Me hice con la Sega Dreamcast mucho más tarde, que fue la última consola que Sega desarrollo. Todavía la tengo en la actualidad y debo decir que ha sido una de las consolas más divertidas a la que jamás he jugado. ¡De veras que me impresionó, y tener Soul Calibur como uno de mis primeros títulos para la consola realmente me impacto!

Figura 2 y 3 – Soul Calibur para el Sega Dreamcast ejecutándose en Reicast sobre ODROID

Obviamente, Sega cometió algunos errores muy graves en lo que respecta al hardware. La Genesis/Mega Drive era un buen dispositivo, pero Sega intentó ampliar la vida útil del sistema con la 32X, y los accesorios de la Sega CD eran demasiado caros y llegaros a provocar muchos quebraderos de cabeza a los desarrolladores de juegos externos debido a los rápidos y continuos ciclos de lanzamientos del hardware y accesorios, así como la limitada vida útil de los propios accesorios.

BDebido a esto, Sega desarrolló por su cuenta casi todos los juegos de Dreamcast. Puede que Sega no fuera el mejor desarrollador de hardware por aquel entonces, pero sabía cómo hacer buenos juegos. Aún continúa produciendo juegos en la actualidad, incluso ha desarrollado juegos para su anterior rival Nintendo y sus consolas.

Emulación de Sega en ODROID

Dado que Sega dejó de producir hardware tras la Sega Dreamcast, probablemente no me equivoque si digo que todas las consolas de Sega funcionan en ODROID, aunque algunas pueden funcionar mejor que otras. Por ejemplo, la Sega 32X, que era un accesorio para la Sega Genesis, fue difícil de emular al principio, pero gracias al recompilador dinámico del emulador PicoDrive ahora podemos jugar a estos juegos, incluso en el ODROID-C1.

Con Reicast, la Sega Dreamcast cuenta con un emulador bastante rápido que permite a los usuarios jugar a muchos, aunque no a todos, los juegos de Sega Dreamcast en ODROID. Incluso la Sega Saturn, que se sabe que tiene una arquitectura muy complicada, se ejecuta al menos en el XU3 y XU4 a una velocidad decente, con bastantes títulos funcionando.

Aunque realmente sólo es jugable en el ODROID-XU3/XU4, la Sega Saturn cuenta con algunos juegos muy buenos, incluyendo The Legend of Oasis, Elevator Action Returns, Keio Flying Squadron 2 o Radiante Silvergun. Personalmente, creo que fue un fracaso como consola 3D. Juegos como Radiant Silvergun o Wipeout ponen de manifiesto que el 3D no era uno de los puntos fuertes de la Sega Saturn. Se pueden apreciar muchas distorsiones en los colores y en general, la calidad 3D es bastante peor que en la Playstation o la N64. Sin embargo, el potencial 2D era bastante mejor. Podría haber sido una gran consola de juegos 2D de 32 bits con muchas escenas de video y gráficos 2D mejorados, en lugar de intentar impulsar el “pseudo 3D”. Obviamente algunos juegos siguieron esta línea, pero muchos otros eran demasiado rígidos para ser juegos de consola 3D.

Por supuesto y, aun así, sigo adorando la emulación de Dreamcast en ODROID. Crazy Taxi 2, Dead o Alive 2, Evolution 1 y 2, Giga Wing 1 y 2, Grandia II, Ikaruga, Incoming, Kidou Senshi Gundam – Renpou vs Zeon DX, Phantasy Star Online Ver. 2, Power Stone 2, Rez, Skies of Arcadia, Sonic Adventure 2, Soul Calibur, Star Wars Episodio I: Racer, Virtual Tennis 2 y Zero Gunner 2. Hay tantos juegos increíbles con excelentes gráficos en 3D que realmente no puedo sacarle todo el partido que desearía. Junto a la PSP, me parece que es la consola con los gráficos más impresionantes que se puede ejecutar en el ODROID.

Reflexiones finales

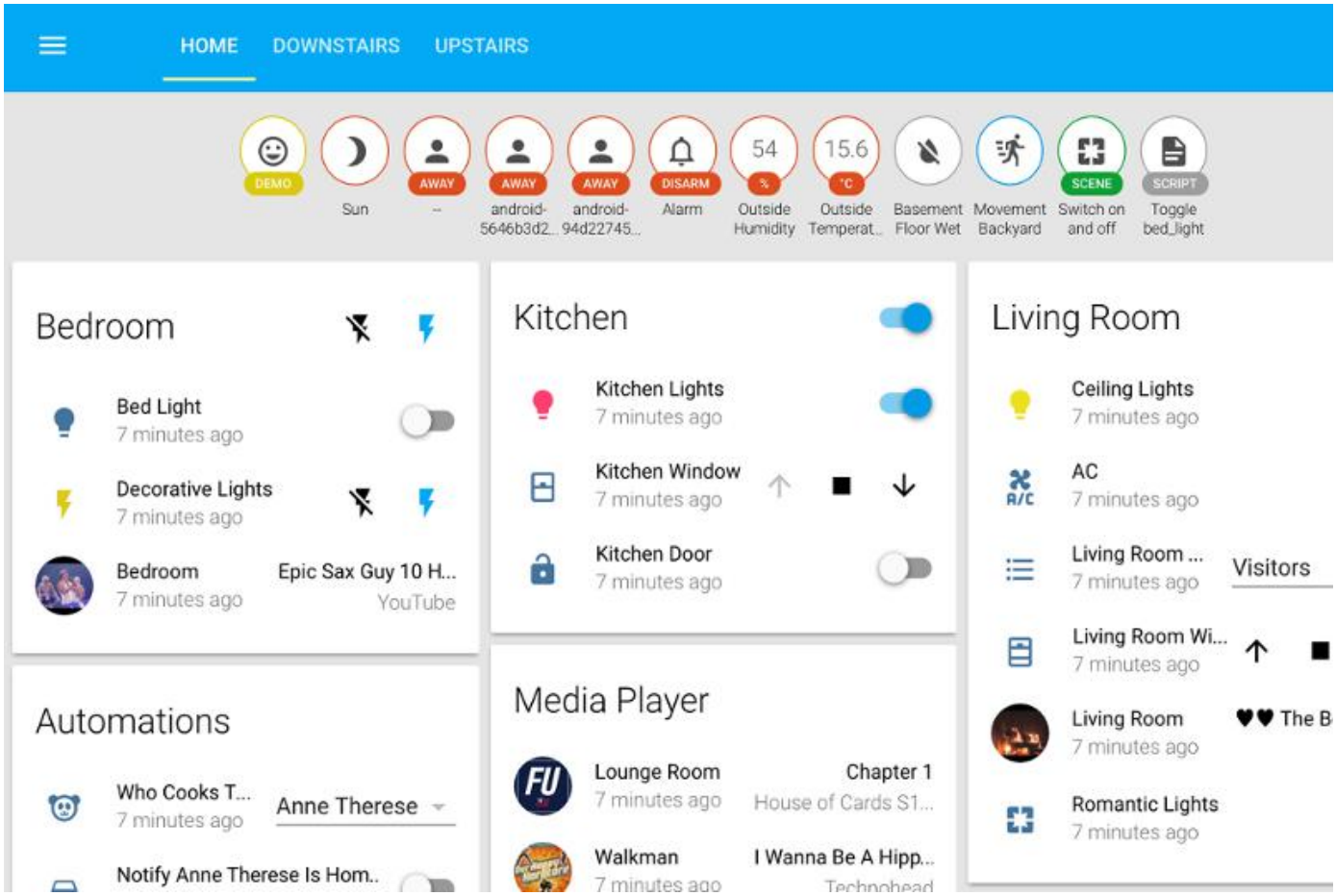
¿Así que soy un fanático de Sega? Supongo que sí, debido a mis buenas experiencias con Sega, primero como niño con la Master System y la Game Gear (en comparación con la NES y Game Boy), y más tarde con la increíble Dreamcast, con la que todavía me divierto hoy día y de la tengo muy buenos recuerdos. Actualmente, podemos elegir los mejores juegos para cada consola de Sega. Puede que no sean tantos como en Nintendo, pero eso no es un problema para mí. Sigo disfrutando de las consolas de Sega como el primer día.

Sin embargo, me di cuenta que las bibliotecas de juegos de Sega ofrecían menos series Triple A en comparación con las consolas de Nintendo, especialmente cuando se trata de grandes series como Super Mario Bros, Pokémon, Zelda, Earthbound, F-Zero, Metroid y Dragon Quest . No obstante, Sega también tuvo sus grandes estrellas, como Sonic the Hedgehog, Phantasy Star, Golden Axe, Outrun, Virtual Fighter y Wonderboy.

Tal vez Sega no haya sido la empresa de consolas con más éxito de la historia, pero desarrollo algunas de las consolas más espectaculares que se pueden tener, aun cuando algunas podrían haber sido un fracaso. Sega también desarrollo numerosos juegos para sistemas arcade, y continúa produciendo grandes juegos hoy en día.

Home Assistant: Personalización y Automatizaciones

September 1, 2017 By Adrian Popa Linux, Mecaniqueo, Tutoriales



En la edición de julio de 2017 de ODROID Magazine, te presenté Home Assistant (**http://bit.ly/2hlOPOE**), una plataforma de domótica de código abierto. Basándonos en los ejemplos que aparecen en ese artículo, trataremos en profundidad temas avanzados relacionados con Home Assistant (HA). Esto te permitirá maximizar el uso de HA, y también te ayudará con los experimentos.

Trabajando con las herramientas de desarrollo de HA

Cuando se accede, la interfaz web de HA es similar a la que se muestra en la Figura 1a. En el anterior artículo, hice alusión a las herramientas de desarrollo dentro de HA. Puede acceder a éstas herramientas mediante el panel izquierdo de la interfaz web y moviéndose entre los botones.

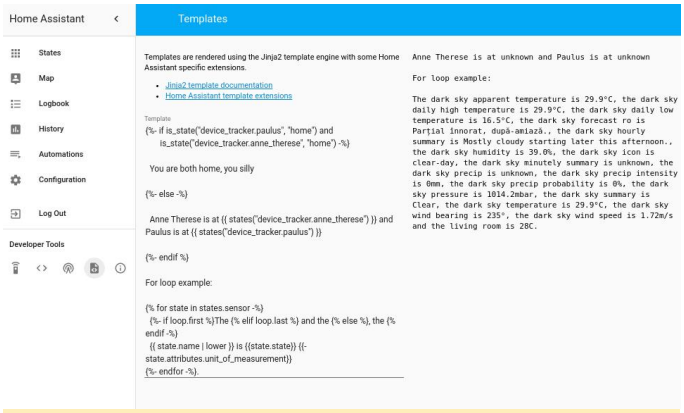
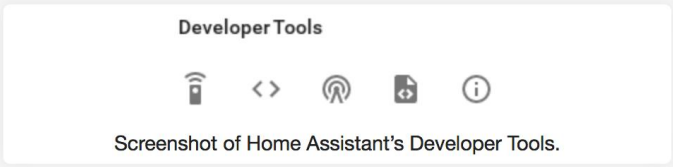


Figura 1a – Interfaz Web



Screenshot of Home Assistant's Developer Tools.

Section	Icon	Description
Services		Calls services from components
States		Sets the representation of an entity
Events		Fires events
Templates		Renders templates
Info		Details about Home Assistant

Figura 1b – Trabajando con las herramientas de desarrollo y plantillas

Las siguientes herramientas son las que están disponibles para los desarrolladores:

Services

Te permite realizar llamadas a los diversos servicios que presentan tus componentes. Puede hacer cosas como activar una automatización, ocultar o mostrar grupos, volver a cargar la configuración de HA, controlar una acción del reproductor multimedia (reproducir/pausar/cargar la lista de reproducción), etc. Los servicios disponibles pueden variar en función de los componentes que tengan activos en tu configuración. Es un buen lugar para probar alguna acción antes de añadirla a una automatización.

States

Esto te permite anular el estado de cualquier entidad. También permite enumerar todas las entidades, con su estado y atributos actuales. Puede utilizar esta lista para encontrar una entidad concreta, ya sea para saber cómo hacer referencia a ella en la configuración (por ejemplo, las entidades visibles en una vista) o para usarla en una plantilla.

Events

Te permite generar un evento en el bus de eventos. Existen varios eventos disponibles, aunque en la práctica puede que no necesites generar algunos de estos eventos.

Templates

El sistema de plantillas de HA utiliza la sintaxis de plantillas Jinja2 (**http://bit.ly/2vd497l**) añadiendo algunas variables internas. La sintaxis de plantilla es más como un lenguaje de programación, así que tomate el tiempo que necesites para leer la documentación (**http://bit.ly/2vOK7no**). El objetivo de las plantillas es procesar los datos de entrada o salida y darles un formato especifico. La vista de Plantillas te proporciona un espacio de trabajo donde puede experimentar y probar la sintaxis antes de pasarla al archivo de configuración. Cuando cargues la página por primera vez, aparecerá un ejemplo de sintaxis que, entre otras cosas, pasará por todos tus sensores mostrando sus valores actuales. Por ejemplo, esto puede enseñarte que puedes acceder a un estado de sensor llamando a {{states.sensor.sensor_name.state}}.

Info

te muestra la versión actual, así como los errores que se han ido registrando.

Para entender mejor la relación entre un nombre de entidad y cómo usarlo en una plantilla, vamos a intentar hacer un experimento. Supongamos que necesitamos obtener el icono del pronostico del tiempo de Dark Sky (**https://darksky.net**). En primer lugar, necesitamos utilizar la herramienta States para obtener el nombre correcto de la entidad. Si buscamos el nombre que aparece en la interfaz web "Dark Sky Hourly Summary", encontrarás una entidad denominada "sensor.dark_sky_hourly_summary". La mayoría de las entidades HA tienen un estado y pueden tener uno o más atributos y éstos ya deberían ser visibles en la vista de Estados. Ahora podemos pasarnos a la herramienta Templates y agregar nuestra propia plantilla al final del cuadro de diálogo.

Vamos a probar las siguientes plantillas y vamos a ver cuál es el resultado:

```
The states object is "{{ states }}"
```

```
The states.sensor object is "{{ states.sensor }}"

The states.sensor.dark_sky_hourly_summary object is "{{ states.sensor.dark_sky_hourly_summary }}"

The states.sensor.dark_sky_hourly_summary.state value is "{{ states.sensor.dark_sky_hourly_summary.state }}"

The states.sensor.dark_sky_hourly_summary.attributes.entity_picture  value is "{{ states.sensor.dark_sky_hourly_summary.attributes.entity_picture }}"
```

El resultado que obtenemos puede verse en la Figura 2. Algunos de los puntos de datos de objetos Python y otros objetos (como el estado y los atributos) devuelven valores en cadena que puedes utilizar. Con esta información, ya estás preparado para empezar a escribir plantillas y automatizaciones.

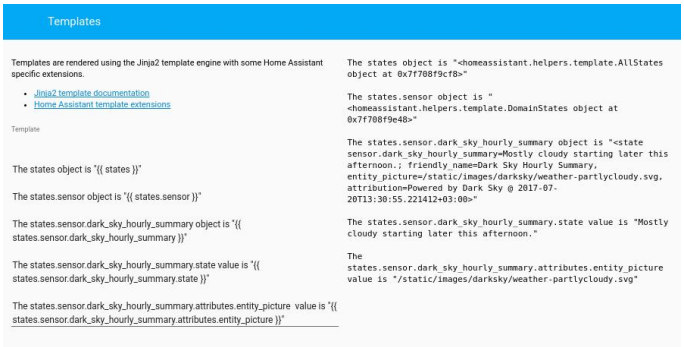


Figura 2 – Plantilla in actión

Interfaz de notificación y API HA

Si tienes scripts que se ejecutan en segundo plano (digamos, por ejemplo, cron) es posible que quieras que se te avise cuando las cosas salen mal y el script falla por cualquier razón. La mayoría de los tutoriales que hay online te enseñaran cómo enviar un SMS o correo electrónico de notificación, pero para problemas que no sean demasiado críticos, tal vez no quieras consultar el correo electrónico o que se te despierte a las 3 de la madrugada. Para ello, puedes enviar mensajes a Home Assistant utilizando curl y su API, para que puedas recibir notificaciones de los scripts cada vez que accedas a Home Assistant. De esta forma, tienes conocimiento de lo que ocurre si accedes con regularidad a la interfaz web. Podemos utilizar una técnica similar para cambiar los estados de las entidades de Home Assistant usando disparadores externos, o puedes usar la API para hacer consultas a las entidades desde scripts externos.

Para configurar esto, tan sólo necesitas ejecutar un comando shell desde tu script cuando quieras gestionar un error:

```
$ /usr/bin/curl -X POST -H "x-ha-access: api_password" -H "Content-Type: application/json" --data "{"message": "Something bad happened in your script", "title": "My background script"}" http://odroid-ip:8123/api/services/persistent_notification/create
```

El comando anterior utiliza la acción de notificación continua (**http://bit.ly/2wkVRiW**) llamada a través de la API de Home Assistant. Para utilizarla, deberás proporcionar el valor “api_password” y enviar un objeto json (**http://www.json.org/**) que contenga el mensaje y el título. Ten en cuenta que los comandos JSON utilizan comillas (“) y no apóstrofe (‘) para citar. Lo bueno es que la notificación aparecerá en todas las vistas/pestañas, de modo que no se pierde. El resultado lo puedes ver en la Figura 3.

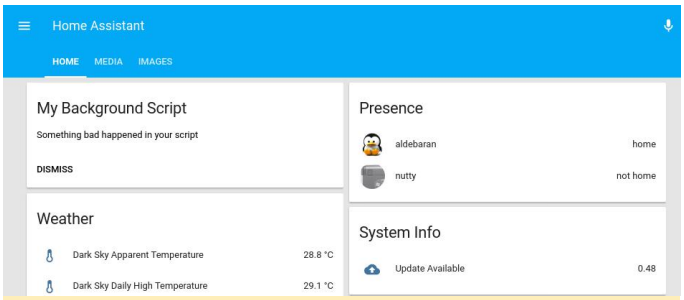


Figura 3 – Notificación continua

Ejecutando scripts externos sobre el cambio de estado

Supongamos, por ejemplo, que deseamos obtener el pronóstico del tiempo desde Dark Sky en rumano para que lo puedan entender quienes no hablen inglés. Puesto que Dark Sky todavía no soporta rumano, necesitamos hacerlo nosotros, lo cual no es un problema, ya que podemos hacerlo con Home

Assistant usando la técnica que describimos a continuación.

1. Instalar un programa de traducción en el sistema ODROID que permita utilizar diversos servicios de traducción online y trabaje con el idioma deseado. Yo utilicé trans (**http://bit.ly/2vcLJDU**):

```
$ sudo wget -O /usr/local/bin/trans
git.io/trans
$ sudo chmod a+x /usr/local/bin/trans
```

Prueba el programa para asegurarte de que funciona adecuadamente:

```
$ trans -b :ro "My name is my password"
```

2. Configurar un nuevo componente de comandos shell en Home Assistant (**http://bit.ly/2vOFnhe**) para llamar el script de línea de comandos. El componente shell puede ejecutar un comando y coger el resultado de una plantilla como parámetro para el comando. Cuando se utiliza una plantilla como parámetro, la ejecución del comando es más estricta y no tienes permitido realizar conexiones o redireccionar al archivo. Si necesitas usar líneas de comandos más complejas con redirecciones y plantillas, puede agregarlas a un script shell y, en su lugar, llamar al script. Afortunadamente, el comando trans permite escribir el resultado directamente en un archivo. Realiza los siguientes cambios en configuration.yaml:

```
shell_command:
  translate_weather: '/usr/local/bin/trans -b
:ro "{{
states.sensor.dark_sky_hourly_summary.state
}}" -o /tmp/ha-weather-forecast.txt'
```

El comando coge el estado del sensor Dark Sky Hourly Summary y lo pasa a trans para su traducción. Practica obteniendo el estado correcto haciendo uso de la herramienta Template, como lo hemos hecho antes. El texto traducido se envía a /tmp/ha-weather-forecast.txt. Para ejecutar este comando manualmente, inicia sesión en la interfaz web de Home Assistant, ve a las Herramientas para desarrolladores en el panel izquierdo y haz clic en el icono Services. Puedes llamar al dominio “shell_command” con el servicio translate_weather sin ningún otro parámetro. Si revisas el archivo temporal, deberías ver el pronóstico del tiempo traducido.

3. Importar de nuevo la traducción a Home Assistant configurando un sensor de archivo (**http://bit.ly/2x2nmuw**). Un sensor de archivo monitoriza los cambios de un archivo e importa la última línea a Home Assistant. Realiza los siguientes cambios en tu configuration.yaml:

```
sensor:
…
 - platform: file
    file_path: /tmp/ha-weather-forecast.txt
    name: Dark Sky Forecast Ro
```

También deberías importar esta nueva entidad en las vistas donde quieras utilizarla:

```
group:
…
  weather:
    entities:
…
      - sensor.dark_sky_forecast_ro
```

Si reinicias Home Assistant, debería ver el nuevo elemento en el grupo Weather. Sin embargo, todavía existe un problema: esta entidad nunca se actualizará. Aún tenemos que añadir un disparador para que cuando el pronóstico en Inglés cambie, el pronóstico traducido también cambie. Para ello, necesitamos utilizar la automatización.

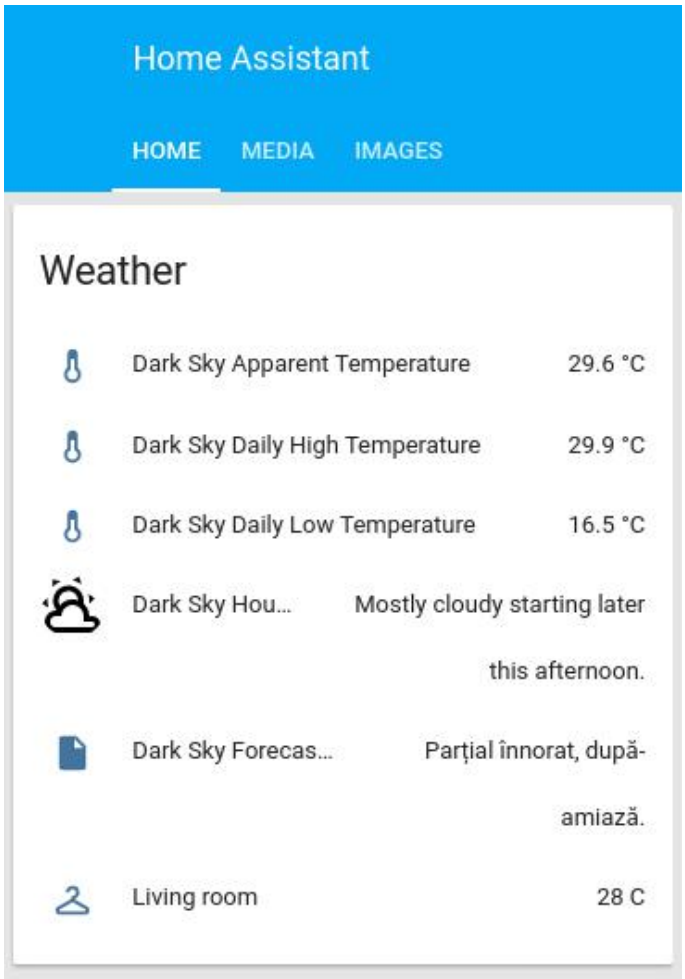


Figura 4 – El pronóstico traducido junto al original

4. Crear una automatización recurriendo al enlace Automations del panel lateral. Ten en cuenta que actualmente necesitarás utilizar el navegador Chrome para completar este paso, ya que el resto de navegadores no son compatibles. Utiliza el botón "+" para agregar una automatización, y asígnale un nombre sugerente como "Weather forecast translation". El disparador debe ser "state" y el id de la entidad debe coincidir con la entidad "fuente" deseada, que es, en nuestro caso "sensor.dark_sky_hourly_summary". Ten en cuenta que estamos utilizando el nombre del sensor tal y como se puede localizar en la herramienta States. Puedes dejar los campos "From" y "To" en blanco, ya que queremos que se active con cualquier cambio de valor.

A continuación, necesitamos especificar una acción o una secuencia de acciones a realizar cuando se active. Necesitamos "Call Service" como un tipo de acción. El Alias es sólo un nombre descriptivo para nuestra acción y podemos llamarlo "Run translate_weather shell_command". El campo Service se compone de la llamada del servicio completa, es decir, el nombre del servicio y del dominio son los mismos que los utilizados en la herramienta Services, por lo que en nuestro caso será "shell_command.translate_weather". El campo Service Data se puede dejar en blanco en nuestro caso, ya que el componente no necesita parámetros adicionales. Ahora puede hacer clic en el icono Save y guardar la automatización.

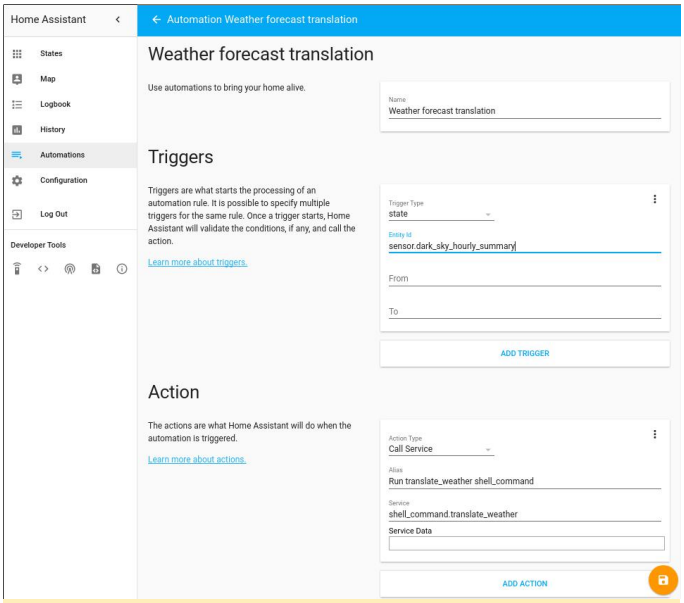


Figura 5 – Crear una nueva automatización

Ahora, cuando el pronóstico del tiempo cambie, se activará la automatización haciendo que el pronóstico se traduzca y guarde en una entidad diferente, a excepción de que posiblemente exista aún un pequeño desajuste. Al reiniciar Home Assistant, es posible que el estado del tiempo no cambie en mucho tiempo y su traducción devuelva "unknown" hasta la primera transición. Para corregir esto, ejecutaremos una segunda automatización en el inicio de Home Assistant para actualizar la traducción. Esta vez el disparador será la plataforma homeassistant con el inicio del evento. La acción será la misma que la automatización anterior. Desafortunadamente, la interfaz de usuario web todavía no admite esta plataforma, así que tendremos que editar el archivo manualmente.

Todas las automatizaciones se guardan en ~/homeassistant/.homeassistant/automations.yaml. En este caso, necesitarás añadir lo siguiente:

```
- action:
  - alias: Run translate_weather shell_command
    service: shell_command.translate_weather
  alias: Update weather translation on startup
  id: '1502097058891'
  trigger:
    platform: homeassistant
    event: start
```

Si observas la automatización que has añadido a través de la interfaz de usuario, verá una sintaxis muy similar. Lo único nuevo es el id. Éste es simplemente la fecha de registro actual de UNIX, y debe ser único para tu sistema (puede obtener uno nuevo con fecha +% s). Una vez configurado y tras reiniciar Home Assistant, obtendrás el estado traducido al momento. A partir de Home Assistant versión 0.51, hay una forma más simple, pero menos eficiente, de hacer lo mismo. Podrías tener un sensor de línea de comandos (**http://bit.ly/2uUIQw3**) con un parámetro de plantilla, como este:

```
sensor:
…
  - platform: command_line
    command:  /usr/local/bin/trans -b :ro "{{
states.sensor.dark_sky_hourly_summary.state
}}"
    friendly_name: Dark Sky Forecast Romanian
```

La razón por la que esta solución es menos eficiente que la primera es que el sensor se consulta constantemente y se traduce cada vez. De modo que, en este caso particular, puedes encontrarte con problemas de limitación de cuota con los proveedores de traducción.

Activar/Desactivar un servicio del sistema desde Home Assistant

Vamos a analizar un nuevo posible caso de uso. Vamos suponer que tienes un servicio del sistema ejecutándose en tu ODROID el cual deseas activar/desactivar desde Home Assistant. En mi caso, este servicio sería Mycroft (**https://mycroft.ai**), porque utiliza algunos recursos cuando está inactivo y puede confundirse con los sonidos ambientales cuando veo una película. Tienes más información sobre Mycroft en **http://bit.ly/2tt3crC**. La cuestión es que puedes utilizar comandos como service mycroft start para controlar el servicio. No estas limitado sólo a los servicios; sino que puedes encender o apagar cualquier cosa.

Para controlarlo desde Home Assistant podemos usar el componente switch de línea de comandos (**http://bit.ly/2wdVGW5**). Añade lo siguiente a tu configuration.yaml:

```
switch:
  - platform: command_line
    switches:
      mycroft:
        command_on: sudo /usr/sbin/service
mycroft start
        command_off: sudo /usr/sbin/service
mycroft stop
        command_state: /usr/sbin/service
mycroft status >/dev/null 2>&1
        friendly_name: "Mycroft status"
```

Y también puedes añadirlo a una vista independiente:

```
group:
…
 switches:
    name: Switches
    view: yes
    entities:
    - switch.mycroft
```

Hay una cosa más que necesitas añadir para que esto funcione. El comando sudo te pedirá una contraseña por defecto, de modo que tenemos que decirle a sudo que el usuario homeassistant puede ejecutar el comando service como root sin una contraseña. Podemos hacer esto ejecutando sudo visudo y añadiendo la siguiente línea al final del archivo:

```
homeassistant ALL=NOPASSWD: /usr/sbin/service
```

Vamos a ampliar este ejemplo un poco. Supón que quieres ser capaz de activar o desactivar un servicio que se ejecuta en un dispositivo diferente. La forma más segura de hacerlo sería a través de ssh. Para ello, tendremos que configurar las claves para ssh, de modo que el usuario homeassistant pueda ejecutar comandos a través de ssh sin que solicite una contraseña (ten en cuenta que por seguridad

necesitarás proteger tus claves). Necesitaras ejecutar los siguientes pasos con el usuario de homeassistant:

1. Crear una nueva clave ssh para homeassistant sin contraseña:

```
$ sudo su -s /bin/bash homeassistant
$ cd ~homeassistant
$ ssh-keygen -t rsa
```

Acepta los valores por defecto (clave almacenada en /home/homeassistant/.ssh/id_rsa y sin contraseña). Puede utilizar esta clave para controlar muchos dispositivos (incluso usarla para iniciar sesión en routers para la detección de presencia), así que no es necesario crear varias claves.

2. Copia la clave al sistema remoto. Asegúrate de introducir la contraseña correcta de la cuenta con la que estás conectando (yo estoy usando root en el dispositivo remoto):

```
$ ssh-copy-id root@other-device-ip
```

3. Prueba la conexión manualmente:

```
$ ssh root@other-device-ip hostname
```

Deberías recibir una línea con el nombre de host del otro dispositivo, sin que se te solicite una contraseña. Si consigues esto, significa que las cosas van bien. Si no, recurre a la excelente guía de solución de problemas en http://bit.ly/2vTQYdA.

4. Configurarlo en Home Assistant añadiendo una nueva entrada switch en configuration.yaml:

```
switch:
  - platform: command_line
    switches:
…
      mycroft_kitchen:
        command_on: ssh root@kitchen
/usr/sbin/service mycroft start
        command_off: ssh root@kitchen
/usr/sbin/service mycroft stop
        friendly_name: "Mycroft Kitchen
status"
```

Si no quieres que se realice un sondeo constante del estado en Home Assistant, puede omitir la línea “command_state” y, en este caso, Home Assistant supondrá que está desactivado y mantendrá solo los cambios que has realizado en la interfaz de usuario. Además, la interfaz cambiará de barra deslizante a disponer de dos iconos para activar/desactivar.

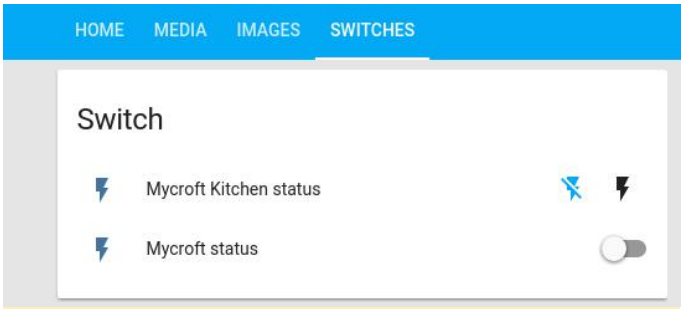


Figura 6 – Switches para los procesos del sistema

Activar/desactivar un switch basado en la reproducción multimedia o en la información de presencia

Ahora que podemos activar/desactivar manualmente Mycroft (o cualquier otro switch igualmente), hagamos las cosas interesantes. Me gustaría tener Mycroft funcionando cuando estoy en casa (mi teléfono está conectado al router y es detectado por la detección de presencia que hemos implementado en el artículo anterior) y Kodi no se está ejecutando. Sin embargo, esto puede parecer ambiguo. Por lo tanto, vamos a definir lo que realmente queremos:

- Usuario cambia de “not_home” a “home” y Kodi está inactivo => activar Mycroft
- Usuario cambia de “home” a “not_home” => apagar Mycroft
- Usuario esta en “home” y Kodi cambia de cualquier cosa a reproducir => apagar Mycroft
- Usuario esta en “home” y Kodi pasa de cualquier cosa a inactivo => encender Mycroft

Para ello, crearemos unas cuantas automatizaciones. La diferencia con respecto a las automatizaciones anteriores está en el uso de las condiciones (http://bit.ly/2x2FDYz). Los disparadores indican cuando debe suceder una acción, mientras que las condiciones se utilizan como filtros e indican si esa acción debe ocurrir.

Por lo tanto, vamos a dar el primer paso. Mi usuario es rastreado por el dispositivo llamado “nutty”. Puesto que la interfaz web no admite condiciones (nota: las condiciones sólo son admitudas a partir de la versión

0.51), tendremos que hacerlo manualmente, en el archivo de configuración automations.yaml:

```
- action:
  - alias: Turn on Mycroft
    service: switch.turn_on
    entity_id:
      - switch.mycroft
  alias: Turn on Mycroft when Nutty arrives
home and Kodi is idle
  id: '1502097058892'
  trigger:
    platform: state
    entity_id: device_tracker.nutty
    to: 'home'
  condition:
    condition: and
    conditions:
      - condition: state
        entity_id:
'media_player.kodi_livingroom'
        state: 'idle'
```

La automatización se activan y se evalúan cada vez que la entidad “device_tracker.nutty” cambia de estado. Cuando se activa, la condición también se evalúa y si “media_player.kodi_livingroom” está inactivo en ese momento, entonces se ejecuta la acción y se activa el switch. Podría haber probado también que Mycroft estuviese apagado, pero activar un switch ya conectado no tiene ningún tipo de efecto. .

Si esto te resulta complicado, aquí tienes el pseudo-código::

```
onStateChange(device_tracker.nutty):
  if states.device_tracker.nutty.state ==
'home':
    if
states.media_player.kodi_livingroom.state ==
'idle':
      switch.turn_on(switch.mycroft)
```

La automatización del apagado es similar, aunque es más simple ya que no tiene condiciones adicionales:

```
- action:
  - alias: Turn off Mycroft
    service: switch.turn_off
    entity_id:
      - switch.mycroft
  alias: Turn off Mycroft when Nutty leaves
home
  id: '1502097058893'
  trigger:
    platform: state
    entity_id: device_tracker.nutty
    to: 'not_home'
```

Las dos últimas automatizaciones deberían ser activadas por los cambios de estado de Kodi y usar las condiciones para comprobar si el usuario está o no en casa.

```
- action:
  - alias: Turn off Mycroft
    service: switch.turn_off
    entity_id:
      - switch.mycroft
  alias: Turn off Mycroft when Kodi is playing
and Nutty is home
  id: '1502097058894'
  trigger:
    platform: state
    entity_id: media_player.kodi_livingroom
    to: 'playing'
  condition:
    condition: and
    conditions:
      - condition: state
        entity_id: 'device_tracker.nutty'
        state: 'home'
```

Y la última debería ser:

```
- action:
  - alias: Turn on Mycroft
    service: switch.turn_on
    entity_id:
      - switch.mycroft
  alias: Turn on Mycroft when Kodi is idle and
Nutty is home
  id: '1502097058895'
  trigger:
    platform: state
    entity_id: media_player.kodi_livingroom
    to: 'idle'
  condition:
    condition: and
    conditions:
      - condition: state
        entity_id: 'device_tracker.nutty'
        state: 'home'
```

Una vez que termines de editar automations.yaml, puede volver a cargar las automatizaciones directamente desde Home Assistant dirigiendote a la vista “Configuration” y seleccionando “Reload Automation”.

Ahora deberías probar las automatizaciones activándolas y verificando el resultado en todos los casos para descartar cualquier error. Puede utilizar la vista Logbook para ver cuándo se han activado las automatizaciones.

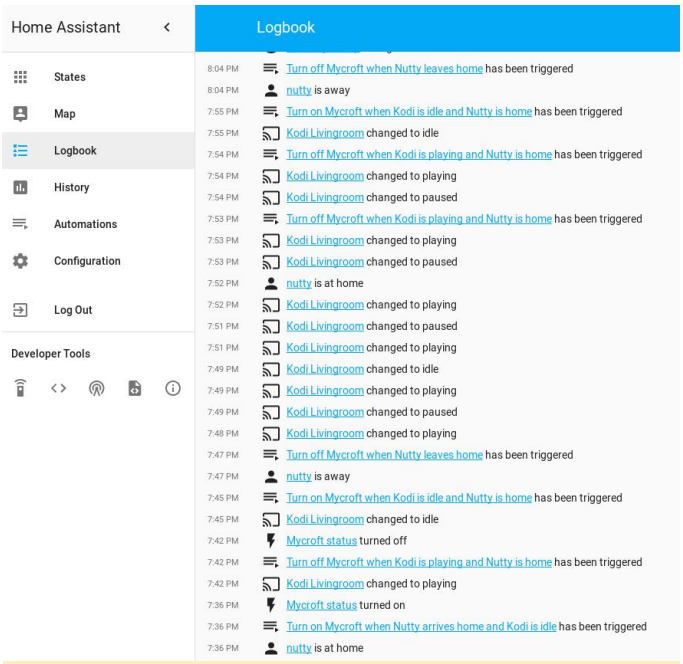


Figura 7 – Visor Logbook

Personalizar los nombres y los iconos

Vamos a hablar de una cuestión más. Por defecto, todos los switches tienen el icono de un “rayo”, tal vez quieras utilizar alguno más apropiado. Además, es posible que desees cambiar el nombre de una entidad, y que cambiando su id estropearía las automatizaciones en las que se encuentre. También hay algunos grupos integrados, como todos los dispositivos gestionados por un “device_tracker” o todas las automatizaciones que te permiten activar/desactivar/iniciar manualmente una automatización, pero que están ocultas por defecto. Para realizar todos estos cambios, tendremos que añadir una sección Personalizada al principio del archivo de configuración, bajo la etiqueta homeassistant, sangrada por dos espacios (http://bit.ly/2x2q6bv).

Vamos a hacer lo siguiente: mostrar el grupo de automatización y cambiar los iconos de los switches por algo más apropiado. Puedes utilizar los iconos de Material Design (http://bit.ly/2wleenC) o tus propias imágenes. Haremos cambios en configuration.yaml:

```
homeassistant:
…
  customize:
    group.all_automations:
      hidden: false
      friendly_name: All automations
    switch.mycroft:
      friendly_name: Mycroft living room
      icon: mdi:assistant
    switch.mycroft_kitchen:
      icon: mdi:assistant
    sensor.living_room:
      icon: mdi:temperature-celsius
…
group:
  default_view:
    entities:
…
    - group.all_automations
```

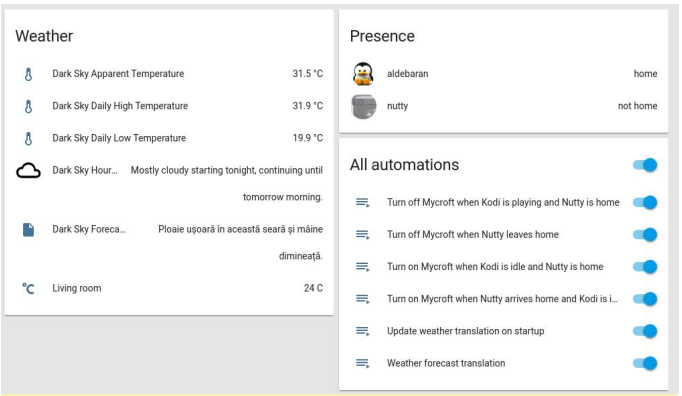


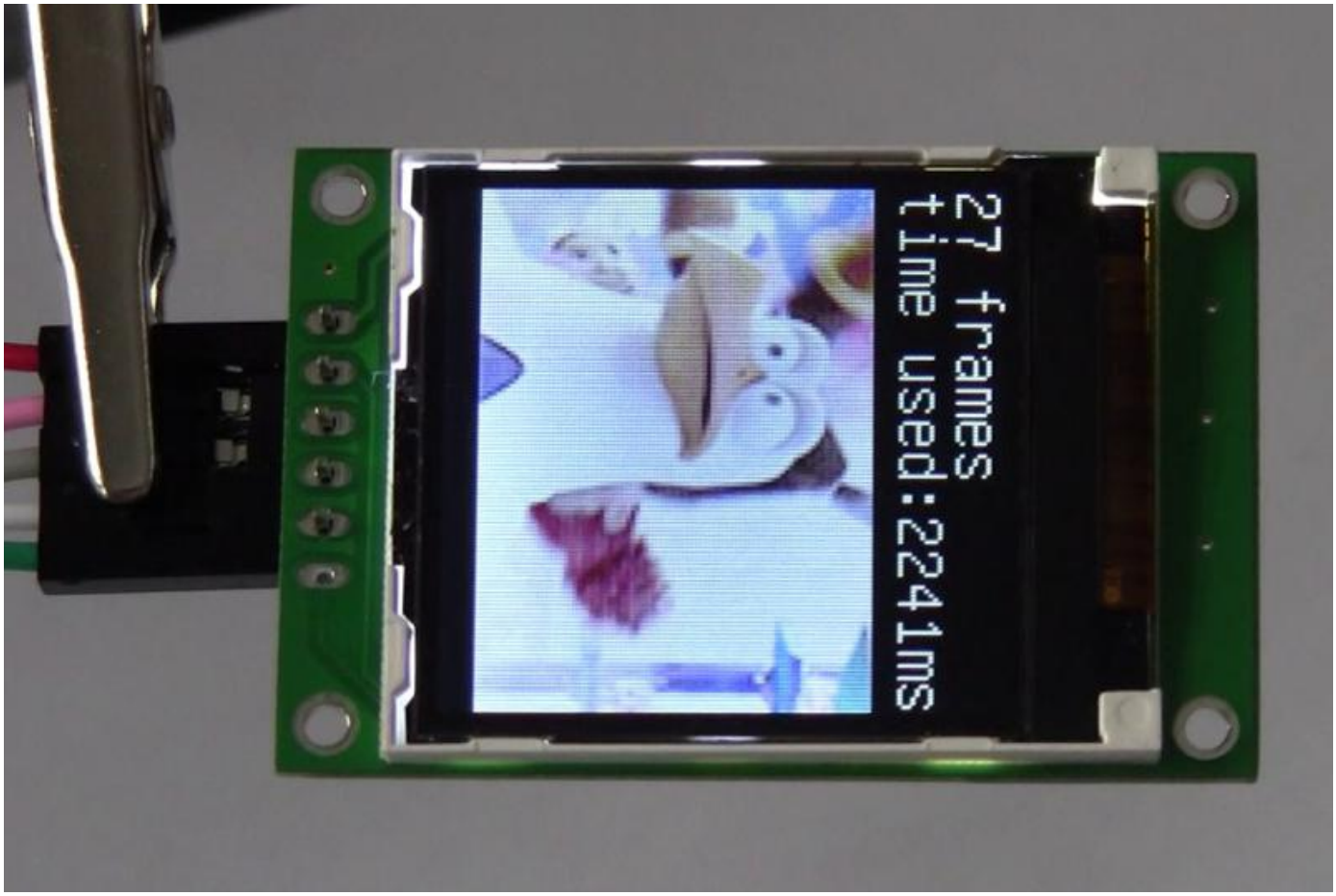
Figura 8 – Personalizaciones para automatizaciones e iconos

Más Ejemplos

La comunidad de Home Assistant tiene numerosos ejemplos en su cookbook en http://bit.ly/2xfAr2V. Tambien tienes más ejemplos en sus foros en http://bit.ly/2v34WbL. Por ejemplo, tienes disponible un ejemnplo de reloj alarma en http://bit.ly/2vOCv48. Para charlar sobre el tema, consulta el hilo original http://bit.ly/2fVogVu.

Pantallas Serie Digole: Manegar la pantalla serie de Digole en los modos UART, I2C y SPI con un ODROID-C1+

September 1, 2017 By Dennis Chang Linux, ODROID-C1+, Tutoriales



Digole.com ofrece múltiples pantallas serie inteligentes que se controlan por medio de un completo conjunto de comandos exclusivos de alto nivel. Estos comandos facilitan el trazado de gráficos complejos y la visualización de imágenes y video, proporcionando una capa de abstracción que ayuda a exportar sus pantallas a un gran número de plataformas. Quizás lo más útil es que todos los modelos de pantallas serie se controlan de la misma forma, con el mismo conjunto de comandos de alto nivel, y siendo su firmware actualizable. El manual de usuario de http://bit.ly/2fXiD9y proporciona una información completa de todos los comandos disponibles.

Para este artículo, utilicé un Módulo Digole serie de 1,8 pulgadas UART/I2C/SPI a color 160×128 OLED con 2MB Flash, número de modelo DS160128COLED-46F. Este modelo no tiene luz de fondo o pantalla táctil como algunas otras pantallas transistores de película delgada (TFT). Si compras un modelo diferente, puede que tengan que modificar el código fuente para cambiar la resolución de la pantalla.

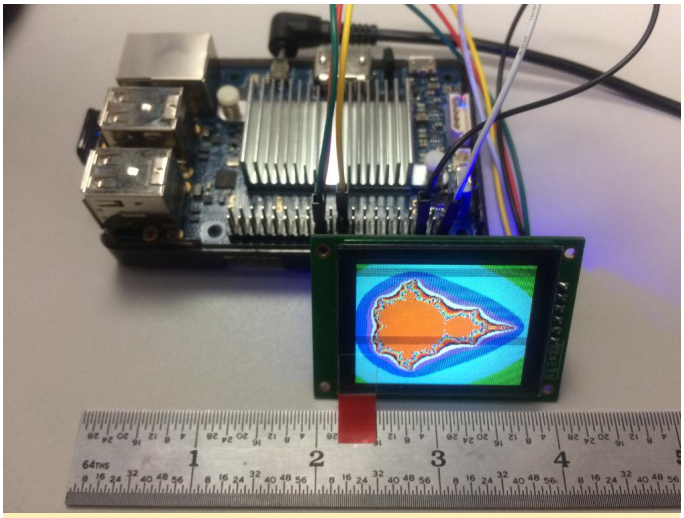


Figura 1 – Prueba mandel.c de Dougherty en modo SPI de 3 hilos. Las líneas de barrido que se observa en las fotos no son visibles para el ojo humano.

Todo lo que aparece a continuación se llevó a cabo en un ODROID-C1+ ejecutando la imagen mínima oficial de Ubuntu 16.04 e iniciando sesión como root.

Prueba inicial a través de la conexión serie UART

La pantalla serie Digole transmite en modo UART (conectores SPI y I2C abiertos). Siempre es de 8 bits, sin bit de paridad, 1 bit de parada. La velocidad de transmisión inicial configurable por el usuario es de 9600.

Con el ODROID apagado, realiza las siguientes conexiones por cable:

Digole VCC = 5V GPIO pin 2 Digole GND = Ground GPIO pin 6 Digole DATA = TXD1 GPIO pin 8

Enciende el ODROID. El Digole debería pasar inmediatamente por su propio proceso de arranque, que implica una prueba RGB y finaliza con una línea de texto. Mi pantalla mostraba "UART baud: 9600 V4.1V + 2MB Flash". V4.1V es la versión de firmware, 2MB Flash es la cantidad de memoria flash disponible en este modelo de pantalla en particular. No todos los modelos tienen memoria flash.

Ajusta la velocidad de transmisión del dispositivo UART:

```
$ stty -F /dev/ttyS2 9600
```

Limpia la pantalla con:

```
$ echo "CL" > /dev/ttyS2
```

For this next command, use single-quotes–not smart-quotes or backticks–so that the terminator is handled correctly:

```
$ echo -n -e 'TTHello ODROID' > /dev/ttyS2
$ echo "CL" > /dev/ttyS2
```

Dibuja un cuadrado de 45px x 45px::

```
$ echo -n -e 'DR--' > /dev/ttyS2
```

Puesto que esto es una simple prueba, no necesitamos aprender la sintaxis de coordenadas de Digole por ahora.

Llegados a este punto, la pantalla es completamente funcional. Es posible utilizar los comandos exclusivos de Digole para controlar totalmente la pantalla simplemente haciendo eco al dispositivo UART. Esto significa que podemos escribir una aplicación o un juego totalmente en Bash o en cualquier otro lenguaje de programación que pueda transmitir directamente al dispositivo UART, incluyendo PHP, Perl, Ruby y Python, aunque probablemente con una velocidad de transmisión más alta. Con este planteamiento, se puede evitar la codificación en C y hacer uso de la librería C de Digole.

Vamos a probar otros métodos de conexión en serie, así que apaga el ODROID y retira la fuente de alimentación para cortar la energía a la pantalla Digole, después retira las conexiones de Digole a los pines GPIO.

Conexión serie I2C

Utilizando un soldador con una pequeña punta cónica, salté con cuidado el conector I2C mientras dejé abierto el conector SPI. Es importante no soldar ambos conectores uniendo las tres almohadillas. Esto requiere tener mucha precisión o un microscopio y una mano muy firme. Con el ODROID apagado, realiza las siguientes conexiones por cable:

VCC Digole = Pin 2 GPIO 5V GND Digole = Pin 6 GPIO puesta a tierra DATA Digole = Pin 3 GPIO I2CA_SDA CLK Digole = Pin 5 GPIO I2CA_SCL

Ten en cuenta que el manual de usuario tiene diagramas con resistencias de 10K o más entre VCC y DATA y VCC y CLK, pero los ejemplos de diagramas de código de la página web no tienen resistencias. Observé que funcionaba bastante bien sin las resistencias, así que no realice pruebas para ver si las resistencias funcionaban.

A continuación, enciende la pantalla. Si soldaste el conector I2C correctamente, la prueba de inicio indicará "Dirección I2C: 0x27 ..." La prueba de arranque no parece saber si DATA y CLK están cableados correctamente.

Habilitar I2C en el ODROID ejecutando:

```
$ modprobe aml_i2c
```

Para probar I2C, usaremos el código C de ejemplo de Digole proporcionado en **http://bit.ly/2xh29MJ**.

El código de ejemplo fue escrito por Javier Sagrera para la Raspberry Pi. Podemos modificarlo para el ODROID con unos pequeños cambios; nada importante, simplemente renombrar algunas referencias a la Raspberry Pi y corregir unos cuantos errores ortográficos, tal como se muestra a continuación:

```
// Pin-out using I2C
// ODROID – Digole LCD
// 1: 5v = 5: VCC
// 3: SDA0 = 4: DATA
// 5: SCL0 = 3: CLK
// 6: GND = 1: GND
/*

// Communication set up command
* "SB":Baud (ascII bytes end with
0x00/0x0A/0x0D) -- set UART Baud Rate
* "SI2CA":Address(1 byte <127) -- Set I2C
address, default address is:0x27
* "DC":1/0(1byte) -- set config display
on/off, if set to 1, displayer will display
current commucation setting when power on

// Text Function command
* "CL": -- Clear screen--OK
* "CS":1/0 (1 byte)-- Cursor on/off
* "TP":x(1 byte) y(1 byte) -- set text
position
* "TT":string(bytes) end with 0x00/0x0A/0x0D -
- display string under regular mode

// Graphic function command
* "GP":x(1byte) y(1byte) -- set current
graphic position
* "DM":"C/!/~/&/|/^"(ASCII 1byte) -- set
drawing mode--C="Copy",! and ~ = "Not", & =
"And", | = "Or", ^ = "Xor"
* "SC":1/0 (1byte) -- set draw color--only 1
and 0
* "LN":x0(1byte) y0(1byte) x1(1byte)
y2(1byte)--draw line from x0,y0 to x1,y1,set
new pot to x1,y1
* "LT":x(1byte) y(1byte) -- draw line from
current pos to x,y
* "CC":x(1byte) y(1byte) ratio(byte) -- draw
circle at x,y with ratio
* "DP":x(1byte) y(1byte) Color(1byte) -- draw
a pixel--OK
* "DR":x0(1byte) y0(1byte) x1(1byte)
y2(1byte)--draw rectangle, top-left:x0,y0;
right-bottom:x1,y1
* "FR":x0(1byte) y0(1byte) x1(1byte)
y2(1byte)--draw filled rectangle, top-
left:x0,y0; right-bottom:x1,y1
*/

#include < stdlib.h >
#include < linux/i2c-dev.h >
#include < fcntl.h >
#include < string.h >
#include < sys/ioctl.h >
#include < sys/types.h >
#include < sys/stat.h >
#include < unistd.h >

int main(int argc, char **argv)
{
int fd;
char *fileName = "/dev/i2c-1"; // Name of the
port we will be using
int address = 0x27; // Address of I2C device
char buf[100];

if ((fd = open (fileName, O_RDWR)) < 0) { //
Open port for reading and writing
printf("Failed to open i2c port
");
exit(1);
}

if (ioctl(fd, I2C_SLAVE, address) < 0) { //
Set the port options and set the address of
the device printf("Unable to get bus access to
talk to slave
"); exit(1); } if (argc>1) {
sprintf(buf,argv[1]);
//printf("%s %d %s
",buf,strlen(buf),buf[strlen(buf)]);
if ((write(fd, buf, strlen(buf)+1)) !=
strlen(buf)+1) {
printf("Error writing to i2c slave
");
exit(1);
}
} else {
printf(" Simple tool to send commands to
Digole graphic adapter
examples:
");
printf(" digolei2ctest "CLTTHello ODROID" -
Clear the screen (CL) and prints "Hello
```

```
ODROID" (TT)
");
printf(" digolei2ctest "CC002" - Draws a
circle at x=30 (0), y=30 (0) with a radius of
32 (2)
"); //not for Character LCD
}

return 0;
}
```

Guarda el código fuente anterior como digolei2ctest.c, luego compílalo:

```
$ gcc -o digolei2ctest digolei2ctest.c
```

A continuación, puedes ejecutarlo para enviar comandos (varios aparecen en los comentarios):

```
$ ./digolei2ctest "CLTTHello ODROID"
$ ./digolei2ctest "CC002"
```

Una vez más, puede utilizar todos los comandos de alto nivel disponibles en el Manual de usuario.

Nota: I2C es el único medio de comunicación con la pantalla serie Digole que es capaz de comunicarse de forma bidireccional. Teniendo en cuenta que únicamente estamos dibujando en la pantalla, la función de recepción de datos no se hace necesaria, pero I2C sí que es necesaria para acceder a la pantalla táctil.

A continuación, probaremos el método de comunicación SPI. Este es el más rápido, pero también el más complicado, de los métodos serie disponibles. Una vez más, apaga el ODROID y retira el enchufe de alimentación para apagar la pantalla Digole, luego desconecta las conexiones entre el ODROID y la pantalla Digole.

Conexión serie SPI de 3 hilos

Usando un soldador con punta cónica, retira cuidadosamente la soldadura del conector I2C y reemplázala soldando el conector SPI en su lugar. Una vez más, asegúrate de no soldar ambos conectores uniendo las tres almohadillas. Con el ODROID apagado, realiza las siguientes conexiones por cable:

VCC Digole = Pin 2 GPIO 5V GND Digole = Pin 6 GPIO puesta a tierra DATA Digole = Pin 9 GPIO MOSI_PWM1 CLK Digole = Pin 23 GPIO SPI_SCLK SS Digole = Pin 24 GPIO (#117) SPI_CEN0

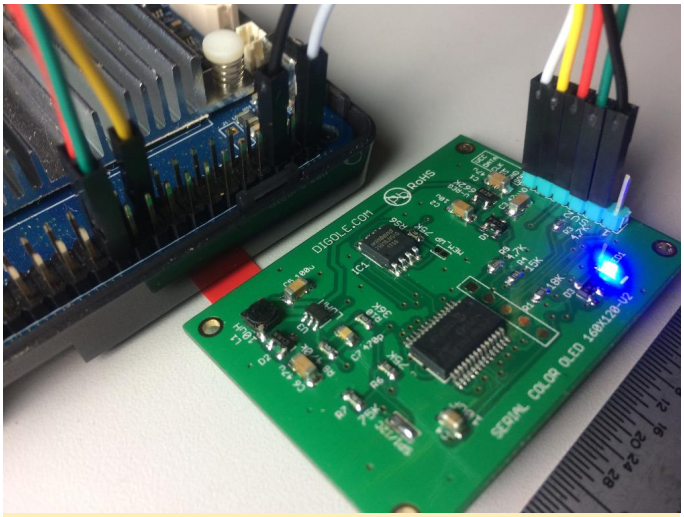


Figura 2 – Cableado SPI instalado

Enciende el ODROID. El texto de inicio de Digole debería empezar por “SPI Mode: 0 …” si soldaste el conector SPI correctamente. No parece saber si DATA, CLK o SS están cableados correctamente

Ten en cuenta que Digole menciona en el manual que el modo SPI tiene el requisito adicional de un “sistema especial de conexión” para “sincronizar los datos”. Para más detalles echa un vistazo al “Diagrama de flujo de datos del transmisor receptor SPI” al final de la sección sobre Conexión del puerto del manual de usuario de la pantalla Digole. Para las pruebas SPI, usaremos el código de ejemplo y el controlador de James F. Dougherty disponible en http://bit.ly/2wmyPIi.

Este script también está escrito para la Raspberry Pi, aunque funciona sin tener que hacer modificaciones sobre el ODROID-C1 +. La única diferencia es la disposición de los pines: conecta el pin SS de Digole al pin 24 GPIO del ODROID-C1+ en lugar del pin 26 GPIO en una Raspberry Pi.

Habilita SPI en el ODROID:

```
$ modprobe spicc
```

Luego obtener y compilar el controlador SPI de Dougherty:

```
$ git clone
https://github.com/jafrado/digole.git
$ cd digole
$ make
```

Ejecuta el código de prueba incluido:

```
$ ./oledtest /dev/spidev0.0
```

Deberías ver que en la pantalla aparece una imagen de una brújula seguida de muchas pantallas de prueba. No te preocupes por la lenta velocidad de dibujo, describiremos una forma de aumentarla en la siguiente sección. Prueba el otro programa de ejemplo para mostrar un fractal de Mandelbrot:

```
$ ./mandel /dev/spidev0.0
```

Existen más programas de ejemplo, pero parecen estar plagados de errores y tienden a trazar gráficos en posiciones impredecibles. Llegados a este punto, entre los programas oledtest.c y mandel.c, deberías tener todo lo necesario para empezar a crear tus propias aplicaciones que utilicen las pantallas serie Digole.

Cuestiones relacionadas con el rendimiento

En el código de Dougherty, cambie el valor spi_speed en la línea 41 de rpi_spi.c de 200,000 a 1,000,000 (1MHz) para aumentar la velocidad con la que se muestran las imágenes en pantalla. Si se supera el 1MHz se interrumpen los comandos de dibujo en el ODROID-C1+. Dougherty comentaba en el código que no fue capaz de superar los 200KHz, aunque en su caso estaba usando una lenta Raspberry Pi Zero en las pruebas.

Es curiosa esta limitación, yo utilicé un simple bucle while con un comando “sleep x” y fui variando los valores de x con el fin de agobiar la pantalla de Digole enviando sentencias de comandos rápido y continuadamente, dando lugar a gráficos mal dibujados o imágenes corruptas, que es exactamente lo que sucede cuando el valor “spi_speed” se incrementa por encima de 1MHz en los programas de ejemplo de Dougherty. En teoría, el bus SPI y la pantalla Digole pueden ir mucho más rápido que 1MHz, pero sospecho que el ya mencionado “sistema especial de conexión” y la gestión precisa de las comunicaciones SPI a nivel de palabra y byte serán necesarios para alcanzar el máximo rendimiento.

Sabemos, sin embargo, que estas pantallas son capaces de funcionar muy bien. Digole aparece en un video de YouTube en **http://bit.ly/2wfwPRJ** el cual muestra una rápida secuencia de video de 27 fotogramas en aproximadamente 2 segundos, que es aproximadamente 14fps.

Desafortunadamente, en el video no se detalla cómo se ha alcanzado esta velocidad. El título del video indica que han utilizado la relativamente nueva función Video Box (a partir del firmware V4.0V) que permite escribir datos de imagen raw directamente en la pantalla. En el Manual del Usuario se dice que la función Video Box funciona a “máxima velocidad: el modo UART-460800bps, I2C-> 400K bps, SPI-10MHz.” Eso es diez veces más rápido que nuestro actual mejor rendimiento alcanzado usando los ejemplos de Dougherty en modo SPI. Es probable que necesites ponerte en contacto con el soporte técnico de Digole para saber cómo conseguirlo.

En relación a las pruebas de bucle while, el comando “sleep 0.05” parece ser el retardo más corto que “TP00TTHello ODROID x00” envía, dando como resultado un parpadeo apenas perceptible de las palabras “Hello ODROID” que se vuelven a trazar sin errores. Para muchos proyectos, especialmente en los que se debe actualizar el texto periódicamente, 0.05 segundos es lo suficientemente rápido, y no necesitamos lidiar con la puesta a punto del rendimiento de las comunicaciones serie.

Conclusiónn

Estoy bastante impresionado con las pantallas serie de Digole por sus múltiples métodos de conexión y sus fáciles, aunque potentes comandos. Tienen muchas características avanzadas, incluyendo fuentes almacenadas, secuencias de comandos almacenadas y pantalla táctil integrada que otras pantallas menos inteligentes simplemente no tienen. La mayoría de las pantallas táctiles son un dispositivo independiente de la pantalla, pero la pantalla táctil Digole se controla a través de la misma interfaz serie que la pantalla. La cuestión es que no hay muchas pantallas de este

tamaño a color y con altas resoluciones, especialmente en OLED.

Espero que estas diminutas pantallas a todo color terminen formando parte de muchos proyectos ODROID, especialmente de proyectos portátiles que funcionen con pilas. Esto es así para los modelos con pantalla táctil resistiva y un puñado de modelos TFT con retroiluminación regulable. Los modelos OLED no tienen luz de fondo para atenuarlos, aunque la reducción de la luminosidad puede lograrse cambiando los colores a tonos más oscuros.

El rendimiento de las pantallas serie Digole es lo suficientemente bueno para usarla en la mayoría de los casos sin tener que llevar a cabo ajustes de rendimiento. Para los juegos y el video donde es importe la velocidad de los fotogramas, por supuesto que es posible lograr un rendimiento aceptable por medio de la gestión de las comunicaciones serie y aprovechando las características avanzadas de la propia pantalla.

Conociendo un ODROIDian: Ted Jack-Philippe Nivan (@TedJack)

September 1, 2017 By Ted Jack-Philippe Nivan Conociendo un ODROIDian

Por favor, háblanos un poco sobre ti.

Tengo 25 años, nací y me crie en Martinica (Antillas francesas). Soy el principal desarrollador de Adok (**www.getadok.com**) y me desenvuelvo tanto en el área de software como en la de electricidad. Actualmente vivo en el sur de París, tengo un título en Ingeniería Eléctrica y Electrónica de la École de Technologie Supérieure (Canadá), una licenciatura en Procesamiento de Imágenes y Señales de la Universidad de Lyon I – Claude Bernard (Francia) y un doctorado en Ingeniería Eléctrica y Electrónica del Instituto Nacional de Ciencias Aplicadas de Lyon (Francia). Además, he realizado mis prácticas de fin de estudios en la Escuela de Medicina de Harvard en la Cuantificación de objetos MRI del cerebro.

Finalizando los estudios en la Escuela de Medicina de Harvard

¿Cómo empezaste con los ordenadores?

Pues bien, empecé con los ordenadores a los 13 años y rápidamente se convirtieron en una necesidad constante de probar este increíble hardware. Sentía en ese momento Me fascinaba tanto el software como los materiales de hardware con los que estaba

hecho el ordenador. He estado trabajando en varios proyectos desde entonces y uno del que estoy particularmente orgulloso fue el de convertir una bicicleta en una motocicleta. Fue en 2012, en mi ciudad natal.

Una bicicleta convertida en una motocicleta

¿Qué te atrajo a la plataforma ODROID?

Principalmente la calidad del producto y la comunidad. Es fácil empezar con los ODROIDs y la plataforma está muy bien documentada.

¿Cómo usas tus ODROIDs?

Utilizo mis ODROIDs principalmente para desarrollar aplicaciones Android, sistemas embebidos y para conocer el desarrollo del kernel de Linux.

¿Cuál es tu ODROID favorito y por qué?

Mi ODROID favorito es el XU4 por su potencial y por la comunidad que hay detrás. Personas como @voodik han estado haciendo un magnífico trabajo manteniendo la plataforma al día.

¿Qué innovaciones te gustaría ver en futuros productos Hardkernel?

Me gustaría ver una placa de mueva generación con un sistema de módulos con la finalidad de disminuir tiempo de comercialización. Además, una placa que soportara Windows estaría genial.

¿Qué aficiones e intereses tienes a parte de los ordenadores?

Soy productor/artista musical independiente bajo el nombre de “runthecode”. También practico algunos deportes como el tenis y el fútbol.

¿Qué consejo le darías a alguien que quiere aprender más sobre programación?

¡Simplemente tiene que ser práctico! La comunidad es muy grande y continúa creciendo enormemente día tras día. Internet es el lugar donde siempre se debe estar. No creo que exista la necesidad de comprar libros para arrancar. De modo que, empieza tan pronto como puedas, incluso si eres joven. A largo plazo verás su fruto. Empieza con lo que tengas, y lo más importante, si quieres tener éxito, no dejes que nadie te diga lo que tienes que hacer en la vida. Cultiva la semilla que hay dentro de ti, lánzate al mercado y llévate bien con las personas adecuadas. Conseguirás lo que te mereces. Recuerda que la gente que ha tenido éxito apenas tenía talento al principio, pero eran personas con unas creencias muy arraigadas.

Ted siempre está inventado algo