Ejecutar Netflix en Linux • ODROID C1+ • Juegos Android • SAMIIO

ODROID Magazine

Año dos
Num. #21
Sep 2015

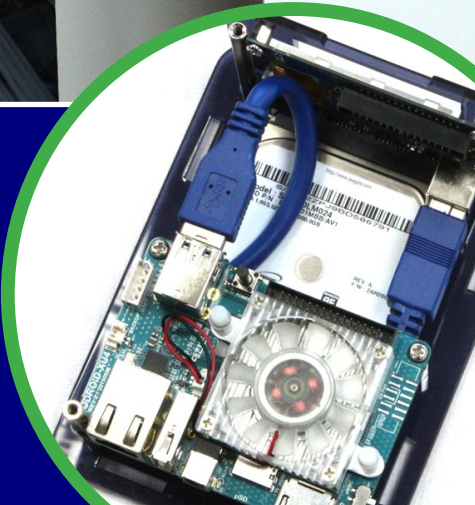
UN ROBOT *Lampara*

Aprende a crear un compañero totalmente robotizado usando una placa ODROID

Fundamentos de Volúmenes Lógicos

Descubre una nueva forma de tocar el piano con Arjuna

Crea tu propio NAS en la nube con Cloudshell



Qué defendemos.

Nos esmeramos en presentar una tecnología punta, futura, joven, técnica y para la sociedad de hoy.

Nuestra filosofía se basa en los desarrolladores. Continuamente nos esforzamos por mantener estrechas relaciones con éstos en todo el mundo.

Por eso, siempre podrás confiar en la calidad y experiencia que representa la marca distintiva de nuestros productos.

Simple, moderno y único.
De modo que tienes a tu alcance lo mejor

HARDKERNEL

Pollin ELECTRONIC

Ahora estamos enviando los dispositivos ODROID U3 a los países de la UE! Ven y visita nuestra tienda online!

Dirección: Max-Pollin-Straße 1
85104 Pförring Alemania

Teléfono & Fax
telf : +49 (0) 8403 / 920-920
email : service@pollin.de

Nuestros productos ODROID se pueden encontrar en: http://bit.ly/1tXPXwe

EDITORIAL

Los genios de Hardkernel han creado recientemente un nuevo kit de bricolaje para montar servidores en la nube llamado Cloudshell. Hecho expresamente para usuarios que desean máxima privacidad, se puede montar con un ODROID-XU4 junto con un disco duro y una pantalla LCD 3,2" al que puede acceder de forma remota utilizando software de código abierto como OwnCloud. El ODROID-C1 recibe una actualización con el nuevo C1+, que ya está disponible en la tienda Hardkernel.

Los usuarios que poseen ODROID son muy creativos, este mes contamos con varios proyectos que ponen al límite la informática ARM. Conoce a Luci, una lámpara robot, que sigue el modelo de la famosa mascota de Pixar con una personalidad única. Aprende a tocar el piano usando Arjuna, desarrolla una simple interfaz de aplicación con QT5, crea potentes aplicaciones de E/S con SAMIIO y administra mejor tu espacio con LVM. También presentamos dos de las imágenes más recientes que la comunidad ha lanzado: Ubuntu Server 14.04 LTS por @meveric, y una distribución única que integra Android y Debian por VolksPC. Como de costumbre, nos aseguramos que te diviertas con ODROID ejecutando Netflix en Linux y destacamos dos de nuestros juegos favoritos para Android: Plague Inc. y Sword of Xolan.

ODROID Magazine, que se publica mensualmente en http://magazine.odroid.com/, es la fuente de todas las cosas ODROIDianas. • Hard Kernel, Ltd. • 704 Anyang K-Center, Gwanyang, Dongan, Anyang, Gyeonggi, South Korea, 431-815 • fabricantes de la familia ODROID de placas de desarrollo quad-core y la primera arquitectura ARM "big.LITTLE" del mundo basada en una única placa.
Para informaicón sobre cómo enviar articulos, contacta con odroidmagazine@gmail.com, o visita http://bit.ly/1yplmXs.
Únete a la comunidad ODROID con miembros en más de 135 países en http://forum.odroid.com/ y explora las nuevas tecnologías que te ofrece Hardkernel en http://www.hardkernel.com/.

HARDKERNEL

ODROID Magazine

Rob Roy, Editor Jefe

Soy un programador informático que vive y trabaja en San Francisco, CA, en el diseño y desarrollo de aplicaciones web para clients locales sobre mi cluster ODROID. Mis principales lenguajes son jQuery, angular JS y HTML5/CSS3. También desarrollo SO precompilados, Kernels personalizados y aplicaciones optimizadas para ODROID basadas en las versiones oficiales de Hardkernel, por los cuales he ganado varios Premios. Utilizo mi ODROIDs para diversos fines, como centro multimedia, servidor web, desarrollo de aplicaciones, estación de trabajo y como plataforma de juegos. Puedes echar un vistazo a mi colección de 100 GB de software ODROID, kernel precompilados e imágenes en `http://bit.ly/1fsaXQs`.

Robert Cleere, Editor

Soy un diseñador de hardware y software que actualmente vive en Huntsville, Alabama. Aunque semi-retirado del diseño de los sistemas integrados, incluyendo más de una década trabajando en el programa del transbordador espacial, continúo diseñando productos de software y hardware, y me interesa la producción de audio/video y las obras de arte. Mis lenguajes de programación son Java, C y C ++, y tengo experiencia con bastantes sistemas operativos integrados. Actualmente, mis proyectos principales son los sistemas navales de seguimiento y control, monitoreo ambiental y la energía solar. Actualmente estoy trabajando con varios procesadores ARM Cortex, pero mi ODROID-C1 es en gran medida el más poderoso de todos

Bruno Doiche, Editor Artistico Senior

Bruno llego a burlarse de la posibilidad de haber provocado un incendio eléctrico durante la edición de esta misma revista que estás leyendo, debido a un cable de alimentación antiguo. ¿Que ocurrió? Estaba configurando remotamente la cuenta de Skype de su madre con los artículos abiertos y necesitaba recargar un iPhone 4 que estaba a punto de apagarse, conecto directamente el cargador a la toma eléctrica y ¡zas!, notó una descarga eléctrica. Lo único que se le paso por la cabeza fue desconectarlo lo antes posible para que todo volviese a la normalidad. La única víctima fue cable de alimentación que quedó fundido.

Nicole Scott, Editor Artístico

Soy una experta en Producción Transmedia y Estrategia Digital especializa en la optimización online y estrategias de marketing, administración de medios sociales y producción multimedia impresa, web, vídeo y cine. Gestiono múltiples cuentas con agencias y productores de cine, desde Analytics y Adwords a la edición de vídeo y maquetación DVD. Tengo un ODROID-U3 que utilizo para ejecutar un servidor web sandbox. Vivo en el área de la Bahía de California, y disfruta haciendo senderismo, acampada y tocando música. Visita mi web enhttp://www.nicolecscott.com.

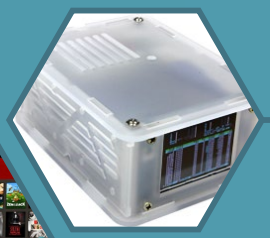
James LeFevour, Editor Artístico

Soy un especialista en medios digitales que disfruta trabajando como freelance en marketing de redes sociales y administración de sitios web. Cuanto más aprendo sobre las posibilidades de ODROID más me ilusiona probar cosas nuevas con él. Me traslade a San Diego desde el Medio Oeste de los EE.UU. Todavía estoy bastante enamorado de muchos aspectos que la mayoría de la gente de la Costa Oeste ya da por sentado. Vivo con mi encantadora esposa y nuestro adorable conejo mascota; el cual mantiene mis libros y material informático en constante peligro.

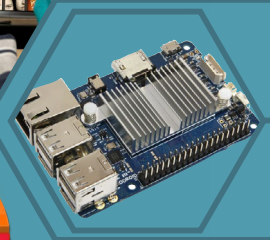
Manuel Adamuz, Editor Español

Tengo 31 años y vivo e n Sevilla, España, aunque nací en Granada. Estoy casado con una mujer maravillosa y tengo un hijo. Hace unos años trabajé como técnico informático y programador, pero mi trabajo actual está relacionado con la gestión de calidad y las tecnologías de la información: ISO 9001, ISO 27001, ISO 20000 Soy un apasionado de la informática, especialmente de los microordenadores como el ODROID, Raspberry Pi, etc. Me encanta experimentar con estos equipos y traducir ODROID Magazine. Mi esposa dice que estoy loco porque sólo pienso en ODROID. Mi otra afición es la bicicleta de montaña, a veces participo en competiciones semiprofesionales.

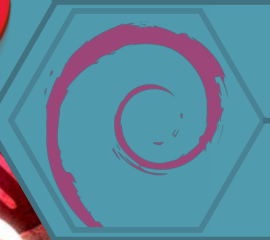
INDICE


CLOUDSHELL XU4 - 6

NETFLIX EN LINUX - 8


ODROID-CI+ - 10


SO DESTACADO: UBUNTU SERVER - 12


SO DESTACADO: ANDROID DEBIAN - 14


JUEGOS ANDROID: PLAGUE INC. - 16


CLUSTER XU4 - 17


JUEGOS ANDROID: PIXELS - 18


COMPAÑERO ROBOTIZADO - 19


ARJUNA - 24

LVM - 26

QT5 - 28


SAMIO- 32

FOROS- 39


CONOCIENDO A UN ODROIDIAN - 40

CLOUDSHELL PARA ODROID-XU4

KIT DE BRICOLAJE - SERVIDOR PERSONAL EN LA NUBE

por Justin Lee

¿Cuál fue la petición "number one" de los fans de Linux para ODROID? Pues bien, aquí la tienes: una carcasa con un impresionante adaptador SATA a USB 3.0 y una consola LCD, que además se presenta en dos colores blanco y azul

El Cloudshell para ODROID-XU4 es una económica solución de Almacenamiento en Red (NAS) para crear tu propia nube personal. Incluye una pantalla de consola LCD a color en el frontal, un conector SATA a USB 3.0 (Génesis GL3321G) y suficiente espacio en el interior como para montar un disco duro de 2,5 pulgadas y una placa XU4. Es una estupenda forma de tener un XU4 con gran capacidad de almacenamiento dentro de una carcasa compacta y portátil.

Especificaciones

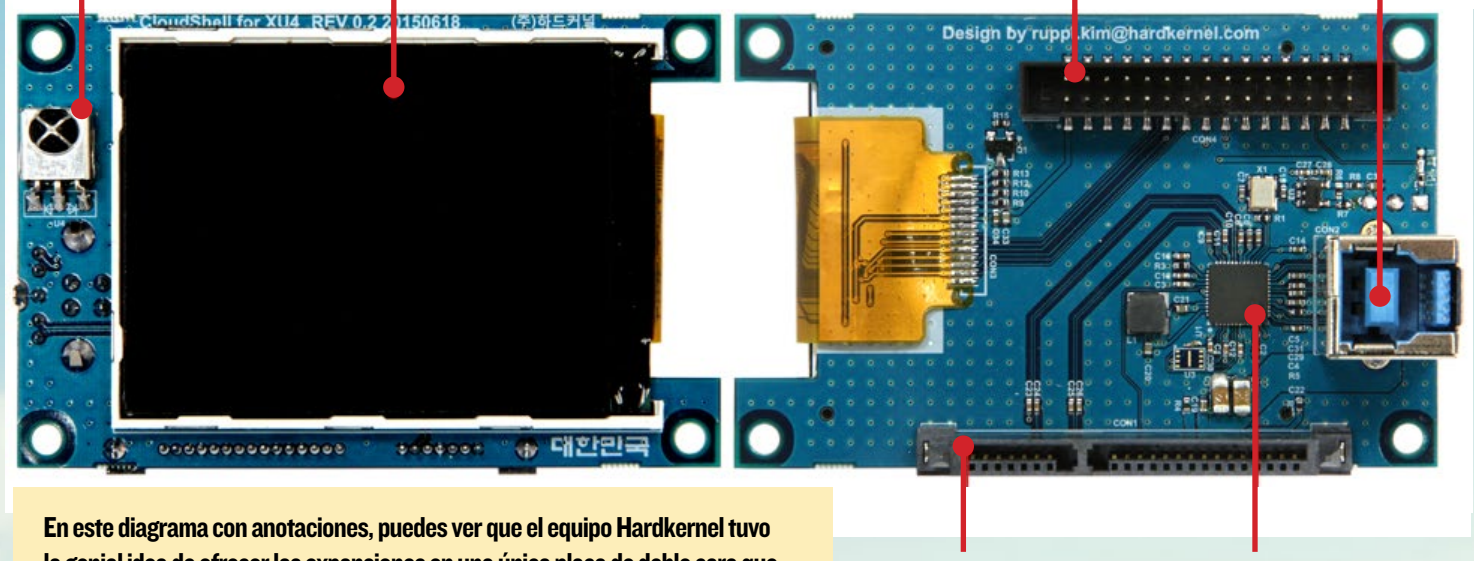
Dimensiones: 47 mm x 25 mm x 21 mm (montado)
Peso: aproximadamente 248g
Color: azul ahumado y blanco ahumado
Placa de Circuitos: con receptor ID, TFT LCD 2.2" 320 x 240, Conector SATA, conector USB 3.0 y conector E/S de 30 pin
Bahía para disco duro: 12-14m m (15mm no es compatible)

Dispones de más información en la Wiki de Hardkernel en http://bit.ly/1Laq7IS.

Más información

Echa un vistazo a un vídeo de demostración del Cloudshell en `https://youtu.be/0c2CxuFzKtI`. Puedes comprar el Cloudshell por 39$ y obtener instrucciones de montaje detalladas en la página del producto en `http://bit.ly/1N3xNm7`.

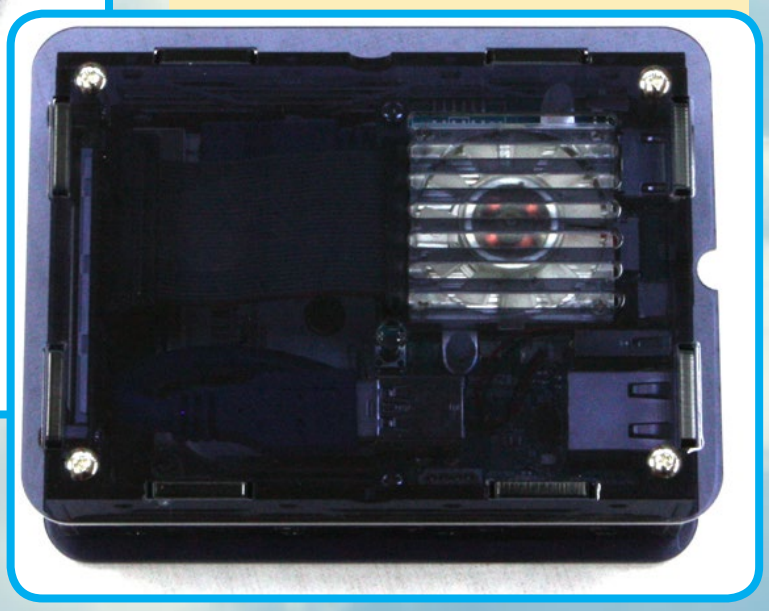
Si necesitas información para configurar un servidor en la nube, consulta la edición de Enero de 2015 de ODROID Magazine, disponible gratuitamente en `http://magazine.odroid.com/#201501`



En este diagrama con anotaciones, puedes ver que el equipo Hardkernel tuvo la genial idea de ofrecer las expansiones en una única placa de doble cara que incluye una pantalla LCD, un controlador SATA y un receptor de infrarrojos.

Si montas el Cloudshell con un disco duro, recuerda que la bahía sólo puede soportar discos SATA de 2,5 de entre 12 y 14 mm. ¡Si intentas montar una unidad de 15 mm no entra!

Aquí tienes el Cloudshell montado. Ni que decir tiene que es muy recomendable mantener las ranuras de ventilación libres de obstáculos para que el aire pueda fluir cómodamente.



NETFLIX BAJO LINUX EN EL ODROID-C1

RELAJATE CON UNA BUENA PELICULA

por @daemon32

Netflix funciona muy bien bajo Android en ODROID-C1 simplemente instalando la aplicación móvil desde Google Play Store, pero conseguir que funcione bajo Linux requiere de algunos pasos más. En este artículo se describe cómo disfrutar de las películas en Netflix mientras ejecutamos Arch Linux o Ubuntu.

Instalación

Para ejecutar Netflix es necesario tener la versión 43.0.2357.134 o superior de Chromium, que está disponible para la mayoría de distribuciones Linux de ARM. Puede comprobar la versión de tu navegador Chromium escribiendo "chromium-browser --version" en una ventana de terminal. Asegúrate de actualizar Chromium si la versión es inferior a la que se necesita.

1. Descarga el paquete de recuperación de Chrome OS desde http://bit.ly/1JUOQQw. Luego, extrae el archivo .zip.

2. Instala el paquete qemu-nbd mediante el instalador de paquetes del sistema operativo que estés utilizando. Por ejemplo, este es el comando de Ubuntu para instalar qemu-nbd:

```
$ sudo apt-get install qemu-nbd
```

3. Registra el protocolo NBD

```
$ sudo modprobe nbd max_part=16
```

4. Monta la imagen de recuperación usando qemu-nbd:

```
$ sudo qemu-nbd -r -c /dev/nbd0 /path/to/chromeos_6946.86.0_daisy-skate_\
  recovery_stable-channel_skate-mp.bin
$ sudo mount -o ro /dev/nbd0p3 /mnt/an/empty/folder
```

5. Copia los archivos libwidevine en el directorio de librerías de Chromium:


```
$ sudo cp /mnt/an/empty/folder/opt/google/chrome/libwidevine* \
  /usr/lib/chromium/
```

6. Lanza chromium con los siguientes parámetros desde una ventana de terminal:

```
$ chromium --use-gl=egl   --user-agent="Mozilla/5.0 (X11; CrOS armv7l
6946.86.0) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/43.0.2357.134 \
  Safari/537.36"
```

Ahora debería ser capaz de visitar el sitio de Netflix y reproducir cualquier película o programa de televisión que te guste.

Notas

Si además deseas instalar el reproductor Flash Pepper para Chromium, copia el paquete de la imagen de recuperación montada:

```
$ sudo cp /opt/google/chrome/pepper/libpepflashplayer.so /usr/lib/chromium
```

A continuación, agrega los siguientes parámetros a la línea de comandos de Chromium que se muestra arriba:

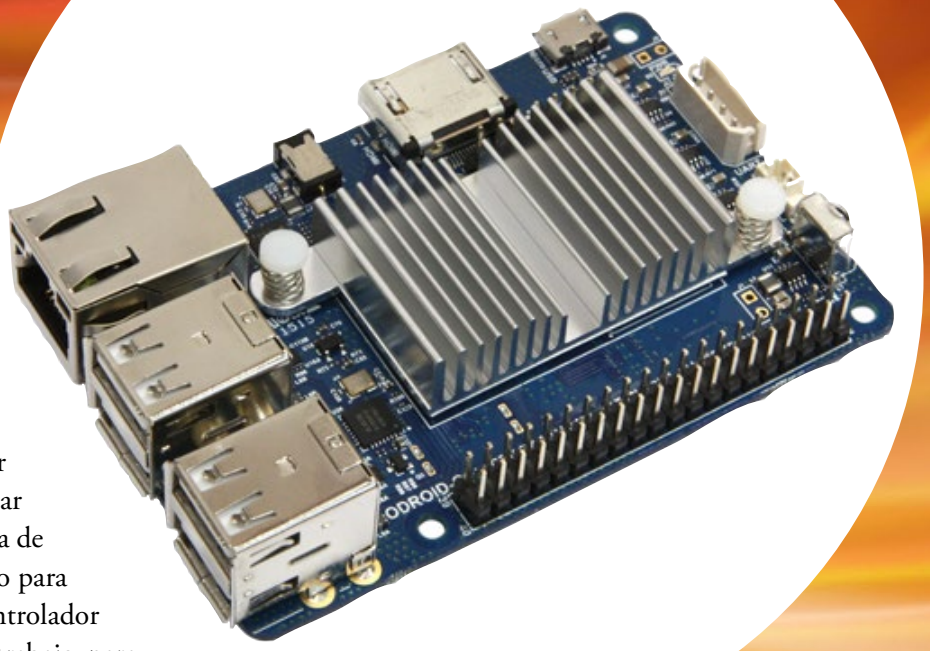
```
--ppapi-flash-path=/path/to/libpepflashplayer.so --ppapi-flash-ver-
sion=18.0.0.209
```

Es normal que Netflix reporte un "Netflix Site Error " al principio. Puesto que el ODROID-C1 es más lento que un ordenador normal, tarda un poco más en cargar la página. Además, no existe aceleración de vídeo para chromium en el ODROID-C1, por lo que los vídeos se ralentizan un poco y se ejecutan mejor a una resolución de 720p. Para preguntas, comentarios o sugerencias, por favor visita el hilo original en `http://bit.ly/1EGr1Qn`.

ODROID-C1+

UNA PLACA PARA TODO EL MUNDO

por Justin Lee



El ODROID-C1 esta considerado como uno de los más potentes y económicos ordenadores de placa reducida, además de ser un dispositivo muy versátil. Con un procesador de cuatro núcleos Amlogic, una avanzada GPU Malí y Ethernet Gigabit, se puede utilizar como un sistema de cine en casa, un ordenador de uso general para navegar por internet, ejecutar juegos y consultar redes sociales, como herramienta de trabajo para el colegio o la oficina, como prototipo para realizar pequeños ajustes de hardware, como controlador para proyectos de domótica, como estación de trabajo para programar y mucho más.

Algunos de los modernos sistemas operativos que se pueden ejecutar en el ODROID-C1 son Ubuntu, Android, Fedora, archlinux, Debian y OpenELEC, con miles de paquetes de software de código abierto totalmente gratis. El ODROID-C1 es un dispositivo ARM, la arquitectura más utilizada en dispositivos móviles y en la informática integrada de 32 bits. El pequeño tamaño de su procesador ARM, su reducida complejidad y su bajo consumo de energía hacen que sea perfecto para desarrollar pequeños dispositivos que podemos llevar con nosotros.

Nueva generación

En Hardkernel, hemos recibido muchas solicitudes para el siguiente modelo de ODROID-W. Así que empezamos a estudiar los componentes para el ODROID-W2. Encontrar la CPU adecuada era la clave del proyecto. Pretendíamos que el coste y el rendimiento fuesen similares al ODROID-W. El procesador de cuatro núcleos Amlogic S805 a 1.5Ghz supera a la Broadcom BCM2835. Lanzamos el ODROID-C1 en diciembre de 2014 y la Raspberry Pi 2 fue lanzada en febrero de 2015. El ODROID-C1 fue reemplazado por el ODROID-C1+ en agosto de 2015.

Estas son algunas comparaciones para que conozcas mejor el ODROID-C1. Ambos son Ordenadores de Placa reducida ARM compatibles con Linux, con un coste en torno a los 35$ y utilizados para diversas aplicaciones y propósitos.

Comparación del Hardware

El ODROID-C1 tiene muchas ventajas sobre la Raspberry Pi. El procesador es un S805 1.5GHz Quad-core de Amlogic con 1GByte de RAM DDR3, Ethernet Gigabit y un receptor infrarrojos. El tamaño de este equipo sigue siendo de tan sólo 85x56mm con un peso de 40g, un funcionamiento muy silencioso, un promedio de consumo de energía de 2-3W y una excelente portabilidad, ya que lo puedes llevar en el bolsillo de una camisa.

Una característica muy interesante del ODROID-C1 es la presencia de una hilera de pines GPIO (Entrada/Salida de Propósito General) situados a lo largo del borde del dispositivo. Estos pines son la interfaz física entre la placa y el mundo exterior. El cabezal de 40 pines incluye SPI, I2C, UART, ADC y función GPIO.

Una tarjeta microSD UHS-1 compatible con el estándar SD 3.01, así como el módulo eMMC más rápido, se pueden encargar con el ODROID-C1 y ambos llegan con el Sistema Operativo Ubuntu o Android ya instalado. Inserta la tarjeta SD en la ranura, conecta un monitor, un teclado, un ratón, el cable de alimentación y Ethernet. ¡Esto es todo lo que necesitas para utilizar el ODROID-C1! y Navegar por la web, jugar a juegos, ejecutar programas de oficina, editar fotos, programar y ver vídeos. El ADC, el receptor de infrarrojos y el RTC del ODROID-C1 también ofrecen muchas opciones para el desarrollo de grandes proyectos de bricolaje.

Especificaciones

CPU Amlogic ARM® Cortex®-A5 (ARMv7) 1.5Ghz de cuatro núcleos
GPU Mali™-450 MP2 (OpenGL ES 2.0 / 1.1 habilitado para Linux y Android)
1GB DDR3 SDRAM
Gigabit Ethernet
40 pines GPIO + 7 pines I2S
Ranura para eMMC4.5 HS200 / Ranura para microSD UHS-I SDR50
4 puertos USB 2.0 Host, 1 puerto USB OTG (Alimentación + datos)
Receptor de Infrarrojos (IR)
Sistema Operativo Ubuntu o Android

Mejoras con respecto al C1

Conector HDMI estándar Tipo A
Incluye disipador de calor
Placas de ampliación I2SBus compatibles con sonido HiFi
La función CEC no requiere batería de reserva RTC
Posibilidad de ser alimentado por el puerto USB OTG o por conector DC
Mejorada la compatibilidad con tarjetas SD

Debido a estos cambios, la carcasa y el disipador no son compatibles con el C1 original. Para descargar el software para el C1+, visita la wiki en `http://bit.ly/1KRKoGV`, el manual del usuario se puede descargar desde `http://goo.gl/iWGYcz`. El C1+ se puede comprar por 37$ en `http://bit.ly/1Upx5yI`.

Vídeos de Demostración

Presentación del Hardware:
`https://youtu.be/L1xYBIVBRgk`

Kit para pequeños ajustes
`https://youtu.be/zocRA1oNY60`

Análisis del rendimiento
`https://youtu.be/L2ZRW-AagSQ`

Dispositivo LCD
`https://youtu.be/SkbZLDl5zTU`

OpenGL ES2.0 con myAHRS+ en Ubuntu
`https://youtu.be/L2ZRW-AagSQ`

Emulación PSP Tekken 6
`https://youtu.be/p8yGS2SHqpA`

UBUNTU SERVER 14.04 LTS

UN ENTORNO DE SERVIDOR OPTIMIZADO PARA TU ODROID

por Tobias Schaaf

Soy conocido en los foros ODROID por haber creado una imagen del escritorio precompilada destinada al entretenimiento llamada ODROID GameStation Turbo. Además de esta imagen, ofrezco una imagen de servidor específica basada en Ubuntu 14.04 LTS. En este artículo, voy a profundizar en esta imagen y lo que se puede esperar de ella.

Descarga

Al igual que todas mis imágenes, la imagen de servidor está disponible en mi espacio web gestionado por Mauro desde Hardkernel (@mdrjr) en `http://bit.ly/1N43pXs`, desde donde puedes descargar la imagen correspondiente a tu sistema. Actualmente ofrezco imágenes para ODROID-X, X2, U2/U3, XU3/XU4 y el C1/C1+

Información general

La imagen es pequeña y limpia, ofrece un entorno de servidor muy reducido sin escritorio. Un entorno de servidor puro y duro. He incluido mi repositorio en cada imagen, pero sólo para las actualizaciones automáticas del kernel a través de los comandos apt-get. Además, cuenta con una colección de script que permite instalar y configurar fácilmente los componentes del servidor, que se detallan más adelante.

Diferencias con la imagen oficial

Mi imagen fue creada originalmente a partir de una imagen de servidor Linaro, por eso el nombre de usuario y contraseña es "linaro" en lugar de "odroid", pero ya que es muy común cambiar el usuario por defecto en un servidor y asignarle una nueva contraseña, esto no debería ser un problema. Además, quise aumentar la seguridad del sistema solucionando algunos problemas muy comunes con las imágenes de HardKernel. Por ejemplo, si usas UART para conectarte a tu ODROID con la imagen HardKernel, inicias sesión como root, lo que significa que no se necesita usuario o contraseña, y tú o cualquier otra persona tienen completo acceso al servidor pudiendo cambiar, instalar o eliminar lo que quiera.

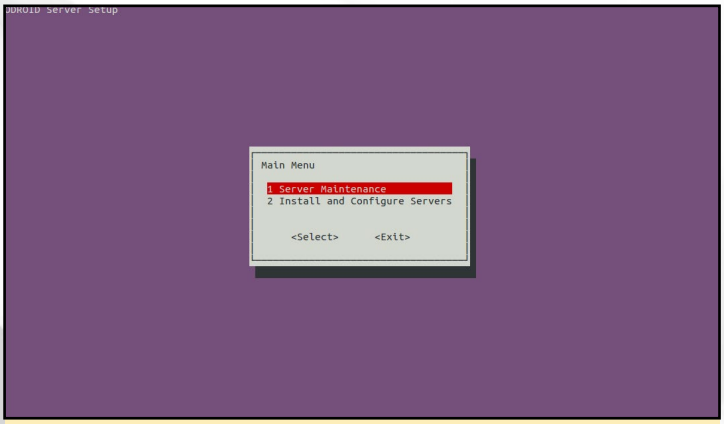
También desactivé la opción para iniciar sesión como root mediante una contraseña para que les sea más difícil a los atacantes obtener acceso al servidor, ya que normalmente intentan conectarse como "root" para tener acceso completo. Como esta imagen no tiene ninguna entrada para root, la seguridad es mayor. También se generan nuevas claves SSH en el primer inicio, de modo que todos los servidores que se monten con mi imagen contaran con diferentes claves SSH.

Otro problema presente en la imagen de Hardkernel es el método de espera de la red. La imagen de servidor Ubuntu por defecto espera casi dos minutos si hay un problema con la red durante el arranque. Esto significa que el proceso de arranque se demora ya que el servidor está en punto muerto mientras espera la red. Puesto que no es esencial, desactivé esa opción para reducir significativamente el tiempo de arranque.

También he añadido un script que aumenta el tamaño de la partición del sistema rootfs hasta su tamaño máximo en el primer arranque. Por eso la imagen es muy pequeña (alrededor de 1 GB) y puede grabarse muy rápido. Después se redimensionará la partición del eMMC o tarjeta SD al completo. Otra diferencia importante son las actualizaciones del kernel a través de apt-get, así como una colección de script que incluyo con mis imágenes de servidor.

Colección de script

Si te has conectado a la imagen, ya sea a través de SSH, UART o directamente con el teclado en tu ODROID, puedes escribir "sudo odroid-server" y se abrirá un menú que te

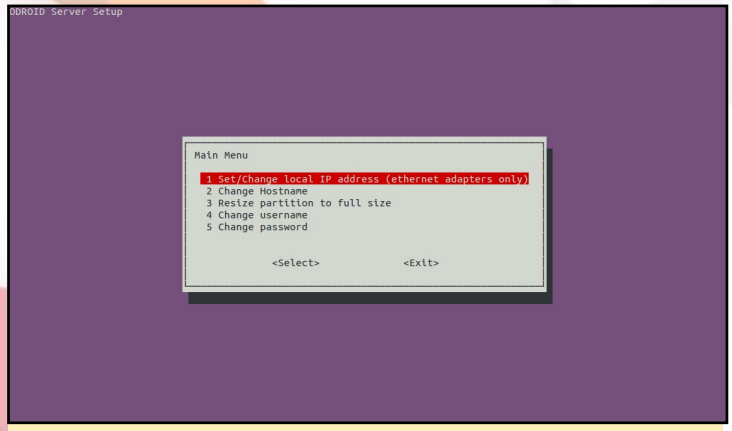


Pantalla de inicio de la colección de script

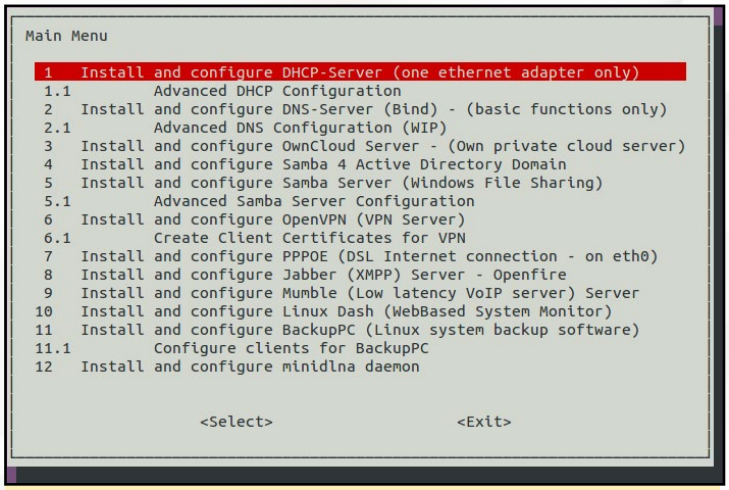
guiará a través de distintas tareas de servidor, como muestra la Figura 1. En este menú, puede seleccionar si desea configurar el propio servidor o instalar y configurar diferentes servicios, como Samba Share, Samba4 Active Directory Domain, OpenVPN Server y muchos otros paquetes.

He creado estos script con el objeto de facilitar la instalación de nuevos sistemas, especialmente para tareas que se realizan con frecuencia. Por ejemplo, cambiar la contraseña de los usuarios del sistema o incluso cambiar el nombre de usuario. Como ya he mencionado, el usuario por defecto "Linaro" se debe cambiar para aumentar la seguridad. Para ello, dirígete a "Server Maintenance" -> "Change username" y cambia el usuario por lo que quieras. La carpeta home será renombrada junto con el usuario. Utiliza el mismo método para cambiar la contraseña del usuario.

He intentado facilitar lo máximo posible la configuración básica, como es el nombre del servidor y la dirección IP del sistema, pero hay más cosas. Si deseas instalar Samba e intercambiar archivos en red, hay una forma muy simple de instalar un servidor Samba y añadir carpetas compartidas. Si te gustaría configurar el nuevo servidor OwnCloud, sólo necesitas unos cuantos clicks gracias a mis scripts de servidor.



Mantenimiento del servidor



Los diferentes servidores que se pueden instalar y administrar

Estos scripts forman parte de mi repositorio git en `http://bit.ly/1ECywYQ` y se pueden utilizar igualmente en otras imágenes, pero están hechos principalmente para usarse con Ubuntu 14.04 LTS ya que cubren algunos detalles que sólo existen en Ubuntu 14.04 LTS. Siempre que tengo tiempo, intento desarrollar y mejorar los script o implementar nuevas funciones. Los scripts no son perfectos todavía, pero me gusta solucionar problemas y mejorarlos. Esto también significa que la gente es libre de hacer peticiones, de solicitar nuevas características o ayudarme a localizar errores. Posiblemente reestructure con el tiempo el menú para que sea más simple su navegación.

Consejos útiles

• Lo mejor es cambiar el usuario y contraseña por defecto por algo que sólo tú conozcas para aumentar así aún más la seguridad del sistema.

• Estas imágenes de servidor no están hechas para ejecutar programas como Kodi o similares, no tienen un escritorio para controlarlos. Son imágenes del servidor puro y duro, y se deben gestionar como tal. Aunque es posible añadir un escritorio y otras cosas, no es recomendable, ya que no tiene instalado libMali.so o los drivers framebuffer, como armsoc o Malí DDX.

• Ubuntu 14.04 LTS usa irqbalance, que normalmente se utiliza para distribuir las interrupciones IRQ a lo largo de los múltiples núcleos, en lugar de que siempre sean gestionadas por el primer núcleo. Esto debería mejorar la E/S y otras cosas. Sin embargo, en Ubuntu 14.04, irqbalance tiene un problema que hace que se utilice hasta el 100% de la RAM disponible durante varios días o incluso semanas. Por lo tanto, deberías desactivar o eliminar este servicio, o crear un trabajo cron que reinicie el servicio una vez al día.

• Es muy recomendable agregar un certificado SSH al usuario root para que cualquier equipo pueda conectarse directamente como root, en lugar de añadir una contraseña a "root". De esta forma, incluso si el sistema está recibiendo servicios de Internet, nadie puede acceder al servidor como root con una contraseña, aunque el servidor todavía puede ser gestionado en tu red local desde un equipo dedicado.

DISTRIBUCION CON ANDROID Y DEBIAN INTEGRADOS

LO MEJOR DE AMBOS MUNDOS

por Vasant Kanchan

Aunque el Kernel Linux ha tenido un gran éxito, todavía estamos esperando la implementación a gran escala del escritorio Linux. La mayoría de las implementaciones de escritorio actuales están hechas para la plataforma x86, y proyectos como VINE tratan de proporcionar una vía que nos permite ejecutar aplicaciones de Windows en un escritorio Linux x86. Usar Debian en x86 permite reducir el coste de las licencias de software, pero pasar a un PC Linux basado en ARM permite además reducir el coste de hardware, ya que hay muchos fabricantes de semiconductores que vender SOC ARM baratos y razonablemente potente. Un Software de escritorio Debian que se ejecute en un PC ARM podría proporcionar una solución informática de bajo coste en el mundo subdesarrollado. De hecho, muchos países utilizan tablets Android en sus colegios como parte de sus programas educativos. Android ha tenido mucho éxito y afortunadamente, se basa en el mismo kernel Linux que utiliza el escritorio de Linux. Los desarrolladores de software independientes también están interesados en escribir aplicaciones para Android, que han supuesto un gran problema para escritorios Linux debido a su limitada penetración.

VOLKSPC

En VOLKSPC (http://www.volkspc.org) hemos desarrollado una solución para integrar Android y Debian con las siguientes características:

• Soporte completo para ejecutar aplicaciones de Android.
• Aplicaciones Debian multi-ventana que funcionan bien con el teclado.
• Cambio instantáneo entre los escritorios de Android y Debian.
• Rápido Inicio de aplicaciones Android desde el escritorio de Debian.
• Los usuarios pueden instalar aplicaciones, tanto desde Google play como desde repositorio de Debian.

Esta distribución unificada basada en Android KitKat y Debian Jessie ARMHF, actualmente se ejecuta en ODROID-C1. Aunque en este artículo se haga constantemente referencia al escritorio Debian, esta solución es aplicable a otras distribuciones basadas en X-Windows, como Fedora y Ubuntu.

Ventajas al ejecutar Android con Debian

Android tiene muchas ventajas tales como:
• Gran cantidad de aplicaciones y juegos disponibles.
• Muy buen soporte para interfaz táctil.
• Interfaz Multimedia acelerada por Hardware.
• Gran número de SOC ARM que ejecutan Android.
• Ejecución mejorada del kernel Linux.

Sin embargo, existen algunas limitaciones con Android a la hora de utilizar pantallas de gran tamaño:
• Todas las aplicaciones utilizan pantalla completa.
• No se puede ver varias aplicaciones al mismo tiempo y no se puede cambiar fácilmente entre ellas.
• No es posible el procesamiento de textos, la gestión de correo electrónico... al estilo escritorio con teclado y ratón.

Las Distribuciones de escritorio Linux como Debian tienen un soporte muy bueno para las aplicaciones de escritorio heredado como LibreOffice, el navegador Firefox y el cliente de correo electrónico Thunderbird. Estas aplicaciones también funcionan muy bien con el teclado y el ratón.

Una distribución con Android y Debian sería una buena solución para ciertas clases de dispositivos o usuarios:
• Una tablet con una gran pantalla y teclado desmontable como el Asus Transformer. Para esta clase de dispositivos Windows 10 es demasiado caro y Android está demasiado limitado.
• Un PC en la nube con teclado y ratón conectados.
• En las economías subdesarrolladas, los consumidores no pueden tener acceso a múltiples plataformas. Un único dispositivo debe satisfacer todas sus necesidades informáticas. Android por sí sólo no puede cubrir todas estas necesidades.

Esquema de ejecución

El escritorio Debian y Android comparten el mismo kernel y a pesar de que las librerías del espacio de usuario son diferentes, ambos se pueden ejecutar simultáneamente. La mayor dificultad a la hora de integrar los dos sistemas es la tecnología gráfica. Android utiliza SurfaceFlinger y Debian utiliza X-Windows para el trazados de los gráficos.

Android y Debian unificados con VNC

Un método muy común para ejecutar Debian en Android es enviar los gráficos X11 a un visor VNC que se ejecute en Android (ref: http://whiteboard.ping.se/Android/Debian).

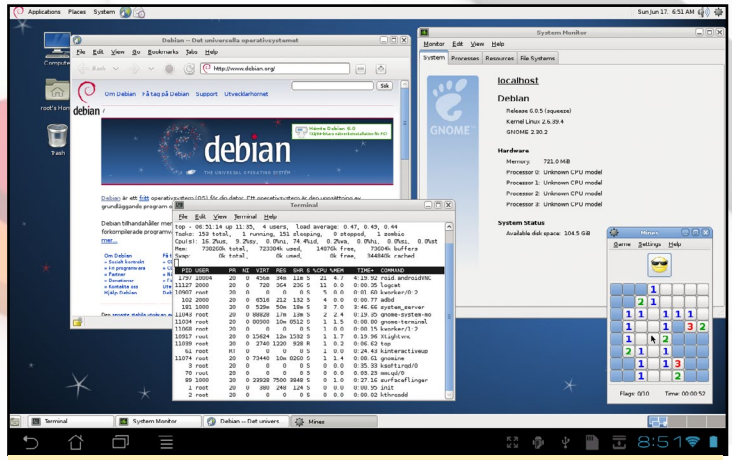


Figura 1 - Android-Debian

Debian con sus librerías puede ejecutarse en un chroot y no interferir con ninguna librería Android. Con VNC, las aplicaciones de Debian pueden redirigir los gráficos a Android. Existen dos problemas con esta técnica:

- **Los gráficos se envían a través de varias capas de software, lo que da como resultado un rendimiento inferior.**
- **Algunas zonas de la pantalla se pierde en el panel Android.**

Aunque esta es una buena prueba de concepto, no podemos decir que sea una buena solución.

Sistema X-Window

El sistema X-Window es un sistema de ventanas orientado a red utilizado por todas las distribuciones de escritorio Linux. El servidor X controla la visualización y es responsable de presentar los gráficos en pantalla. Las aplicaciones son clientes X que intercambian mensajes con el servidor X mediante el protocolo de ventanas X. Las aplicaciones se escriben normalmente con kit de herramientas de alto nivel como GTK, QT, Tcl / TK, etc.

La Figura 2 muestra cómo una aplicación se comunica con el servidor X en un entorno de escritorio Linux. La mayor complejidad del sistema de ventanas X está en la arquitectura cliente-

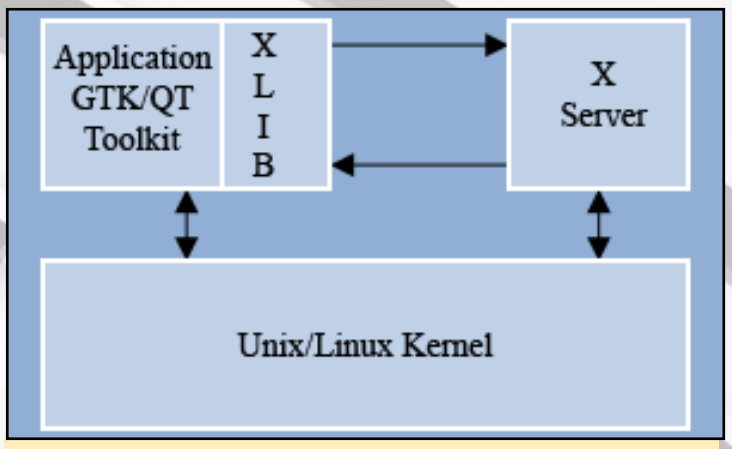


Figura 2 - X Windows

servidor. Los aspectos de diseño que hacen complejo y lento este sistema son:

- **Tanto el cliente como el servidor tienen que almacenar en el buffer y formatear los comandos y las respuestas según el protocolo de ventanas X.**
- **Las solicitudes síncronas y de ida/vuelta son ineficaces.**
- **El frecuente cambio de Contexto entre aplicaciones y el servidor X degrada el rendimiento.**
- **Los gráficos sólo se puede iniciar una vez que el servidor X empiece a funcionar.**

Hay esfuerzos dentro de la comunidad de código abierto para cambiar a nuevas tecnologías de servidor de visualización tales como Wayland y MIR.

Gráficos MicroXwin

MicroXwin implementa un procesamiento gráfico en el kernel como un módulo de carga y proporciona una interfaz peculiar a la librería X11 asociada.

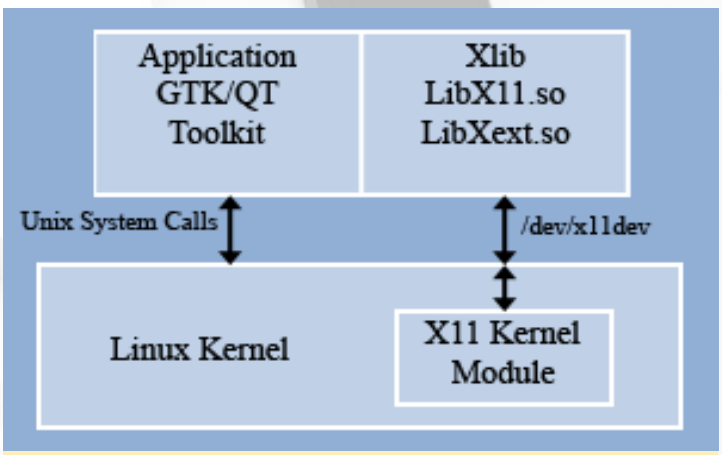


Figura 3 - MicroXWin

Las ventajas de este diseño son:

- **Baja latencia y recorrido completo**
- **Mínimo buffer de solicitudes y respuestas.**
- **Sin cambio de contexto**

¿NECESITOS ALGO QUE HACER MIENTRAS ESTAS ENFERMO?

EXTERMINA LA HUMANIDAD MIENTRAS TE RECUPERAS CON PLAGUE INC.

por Bruno Doiche

Lo peor de pillar un resfriado o la gripe es cuando pasas de "¡bueno, un par de días de descanso; Voy disfrutar de mi tiempo libre!" a "estar arto de Netflix, cansado de navegar por Internet, de vichear los foro ODROID, ¿y ahora qué" Alguna vez te has preguntado, ¿Cómo sería si fuerzas el paciente cero de una plaga que está a punto de poner fin a todos los seres humanos? Ahora puedes hacer todo lo posible por ser lo peor que nunca le haya pasado a la civilización humana desde la peste negra en 1348.

https://play.google.com/store/apps/details?id=com.miniclip.plagueinc&hl=en

¿Ahora puedes ver lo que pasa cuando no te lavas las manos después de usar el baño?

- **Representación directa de los gráficos por el cliente.**
- **No requiere cambios para aplicaciones Xlib existentes.**
- **2 veces más rápido que el sistema de ventanas X**

Android y Debian unificados con MicroXwin

Con MicroXwin, las aplicaciones de Debian pueden escribir directamente al buffer sin pasar por SurfaceFlinger de Android. Esto necesita que los gráficos de Android y MicroXwin coexistan y se ejecuten simultáneamente sin interferir entre sí. Además de proporcionar soporte a la API Xlib para aplicaciones de Debian, el módulo kernel de MicroXwin también proporciona las siguientes funciones:

- Controla los atajos de teclado (LeftAlt + LeftMeta) y facilita el cambio entre las pantallas de Android y Debian

- Se mantiene el estado de los gráficos de Android y de Debian.

- En un caso dado, únicamente Debian o Android ocupa toda la pantalla, y el cambio es instantáneo.

- Las aplicaciones ignoran que el escritorio se está visualizado en pantalla.

- Proporciona una simple API que permite cambiar entre el escritorio de Android y Debian

El Rendimiento de las aplicaciones con una distribución unificada será similar al que se experimenta cuando se ejecutan bajo sus respectivos entornos. Sin embargo y dado que muchas aplicaciones se ejecutarán tras el arranque, será necesario proporcionar suficiente memoria al sistema.

Rendimiento de la distribución unificada en el ODROID-C1

Esta distribución unificada https://www.youtube.com/watch?v=USNISy17-YU ha sido exportada a la placa ODROID-C1 y el enlace de descarga está disponible en el foro general del ODROID C1. Hemos exportado el escritorio XFCE que está disponible sobre Jessie.

Utilizamos gtkperf para medir el rendimiento de los gráficos tanto en Ubuntu como en nuestra distribución unificada.

En Ubuntu, gtkperf necesitó unos 43 segundos para completarse frente a los 12,62 segundos de nuestra distribución unificada – el triple más rápido.

La aplicación lxtask de Debian muestra 566MB de memoria libre con una resolución de pantalla de 720p, y alrededor de 500 MB de memoria libre en 1080P. De modo que hay un ligero aumento en el uso de memoria. Si fuera necesario, el usuario puede salir completamente del escritorio XFCE Debian.

CLUSTER XU4

UN COMPLETO ESTUDIO DE LAS DIVERSAS OPCIONES DISPONIBLES PARA LA INFORMATICA DE ALTO RENDIMIENTO

por Alan Mikhak

Un clúster XU4 gestionado por una consola XU3

He evaluado varios dispositivos ODROID-XU4/XU3 en forma de clúster integrado para la informática en paralelo con OpenCL y MPI utilizando Linux y la denominada tecnología Heterogeneous Multi-Processing (HMP). Como consultor de software, la finalidad de esta evaluación fue la de ampliar mi kit de herramientas para poder ofrecer un mejor servicio a mis clientes.

Objetivos del proyecto

Empecé con algunos requisitos. Buscaba una plataforma con una CPU ARM Cortex-A9 como mínimo, y una GPU que pudiese ejecutar el escritorio Unity de Ubuntu con gráficos acelerados por hardware, soporte OpenGL, OpenCL y CUDA con hardware vector y punto flotante. No quería ejecutar los nodos del clúster sin tener nodos centrales. Disponer de un escritorio común en cada nodo estaba y sigue estando en mi lista de prioridades. Una interfaz de WiFi también sería interesante para poder evaluar el clúster inalámbrico.

Evaluaciones

Al principio, tuve en cuenta las plataformas con procesador Freescale i.MX6 Quad, que tienen cuatro núcleos Cortex-A9. Evalué también la placa Wandboard Quad que cuenta con la GPU Vivante GC2000 y la CPU i.MX6. También incluye una interfaz Wifi Broadcom BCM4329. Había usado una Wandboard Duo hace unos años con una versión antigua del escritorio de Ubuntu. Sin embargo, sea lo que sea lo que hiciese, no lograba que el escritorio Unity de Ubuntu 14.04 se ejecutara en el Wandboard Quad y detectara mi monitor Samsung SyncMaster BX2431 HDMI.

Para ejecutar el escritorio Unity de Ubuntu 14.04 necesitaría una plataforma que soportara OpenGL por completo, no sólo OpenGLES. También necesitaba aceleración hardware para el sistema X-Windows. Empecé a considerar las plataformas basadas en el Samsung Exynos 5 Octa, que cuenta con una CPU con cuatro núcleos Cortex-A15 y cuatro núcleos Cortex-A7, así como la GPU ARM Mali T628MP6 integrada. Compre un Samsung Chromebook II, que ejecuta ChromeOS con aceleración gráfica. El Chromebook cuenta con un monitor, teclado, track-pad, y WiFi integrados. Esto parecía suponer una ventaja en mi caso, teniendo en cuenta el coste de los adaptadores KVM HDMI USB y los monitores HDMI. Cada nodo Chromebook II sería autosuficiente. Puse el Chromebook II en modo desarrollador e instalé Ubuntu 14.04, pero de nuevo no lograba que el escritorio Unity se ejecutara.

Después evalué una placa Arndale Octa también basada en el Exynos 5 Octa. La Arndale Octa parecía prometedora ya que www.linaro.org había anunciado soporte para la misma. Una vez más, no importaba lo que intentase, no lograba conseguir que el escritorio Unity de Ubuntu 14.04 se ejecutara en la placa Arndale Octa y detectara mi monitor Samsung HDMI.

Pasé a evaluar el ODROID-XU3 que también se basa en el Exynos 5 Octa. La página web Hardkernel indicaba que el escritorio MATE de Ubuntu ya se podía ejecutar con aceleración. El Soporte para HMP también estaba muy cerca. El ODROID-XU3 fue el mejor que encontré hasta el momento ya que, de hecho, arranca con un escritorio gráfico de Ubuntu y con mi monitor Samsung HDMI. Sin embargo, decidí seguir buscando una plataforma que pudiera ejecutar el escritorio Unity al que yo estaba acostumbrado.

Luego, pase al kit de desarrollo Nvidia Jetson TK1 basada en el SoC Tegra K1 de Nvidia que integra una CPU ARM Cortex-A15 con cuatro núcleos y GPU NVIDIA Kepler con 192 núcleos CUDA. El Jetson TK1 es la primera plataforma evaluada que ejecuta el escritorio Unity de Ubuntu 14.04 sin problemas con aceleración gráfica. Luego añadí un segundo Jetson TK1 para montar un clúster de dos nodos y poder evaluar los programas MPI CUDA. El único requisito que faltaba

MATA AL DRAGON, SALVA AL PUEBLO

SWORD OF XOLAN CONFIMA QUE NO IMPORTA LA ALTA DEFINICION DE NUESTRAS PANTALLAS, PORQUE SIEMPRE AMAREMOS LOS PIXELES

por Bruno Doiche

Uno de mis juegos favoritos de todos los tiempos fue un juego de arcade llamado Black Tiger, un clásico juego de plataformas Capcom que mezcla aventura con algunos elementos de RPG, donde buscas monedas para conseguir mejores elementos y progresar en el juego. Me encanta emularlo siempre que puedo, y no podría estar más feliz de haber topado con esta joya de un desarrollador de juegos indie turco que logró capturar la emoción que sentí la primera vez que lo jugué. ¡Viene con la ventaja de que soporta excelentes mandos!

https://play.google.com/store/apps/details?id=com.Alper.SwordOfXolan&hl=en

¡Plataformas, compruébalo. Se ejecuta en todos los ODROIDs, compruébalo. Soporte para mandos, compruébalo. Impresionante Diversión de capa y espada , compruébalo!

era OpenCL. El Jetson TK1 no soporta OpenCL aunque el SoC Tegra K1 de Nvidia sí que lo hace.

Para evaluar tanto CUDA como OpenCL en un clúster MPI, necesitaría trabajar con placas basadas en el Soc Tegra K1. Otra posibilidad sería montar un segundo clúster para evaluar OpenCL con MPI y dejar la evaluación de CUDA y MPI para el clúster Jetson TK1. Justo por aquel entonces, vi anunciando el ODROID-XU4 a la mitad del precio del ODROID-XU3, del Jetson TK1 y del Arndale Octa. Así fue como terminé con un clúster de cinco nodos, cuatro ODROID-XU4 y uno ODROID-XU3, que funcionan muy bien hasta el momento.

Configuración final

Montar un clúster con cuatro placas ODROID-XU4 fue bastante sencillo. Pedí las placas en www.ameriDroid.com y algunos separadores de aluminio en www.pololu.com. Ambos proveedores entregaron los componentes bastante rápido, y todo funcionó bien nada más abril la caja. No tuve que devolver las placas en absoluto, lo cual fue un alivio y un ahorro de tiempo. Apilé las placas con separadores M-F 4-40 de 1.25", con separadores de 0,25" en la parte inferior y fijé todos los componentes con tornillos 5/16".

Las placas XU4 comparten dos monitores HDMI ASUS VX238H y dos pares de teclados y ratones USB, junto a un único ODROID-XU3 y un clúster de placas Nvidia Jetson TK1 a través de dos switch Bytecc HDMI KVM USB de 4 puertos, que venía con cables USB + HDMI y el correspondiente control remoto por infrarrojos. He comprado un hub USB para alimentar a uno de los teclados, un teclado para juegos Corsair K70 con luces, sin el cual mi ratón Microsoft no funciona cuando se conecta a puerto USB 2.0 del XU4 a través de un switch USB. Inicialmente intente utilizar switches Bytecc de 5 puertos HDMI para compartir los monitores mientas usaba switches USB independientes para los ratones y teclados. El resultado fue que ni el monitor ASUS VX238H ni el Samsung SyncMaster BX2431 HDMI eran detectados correctamente. Así que opte por usar switches Bytecc KVC HDMI USB de 4 puertos.

Grabé la imagen de Ubuntu 15.04 para las placas XU3/XU4 en cuatro microSD UHS-1 16 GB que inserté en la ranura microSD de las placas. Con el interruptor del modo de arranque fijado para arrancar desde microSD, tres placas llegan al escritorio MATE Ubuntu. La cuarta placa también arranco correctamente tras reemplazar su microSD por una quinta tarjeta. Confirmo que los 8 núcleos eran usados por Ubuntu en modo HMP.

Los ventiladores a menudo se encienden, generando algo de ruido. Tras cerrar Ubuntu y desconectar la alimentación, el LED rojo permanece encendido, aunque al desenchufar el cable HDMI se apaga. El clúster está conectado a través de un switch NETGEAR GS-116 de 16 puertos Ethernet Gigabit. He usado un simple programa MPI con MPICH para verificar el funcionamiento de XU4 y XU3 como clúster. Todo funciona muy bien. hasta al el momento.

LUCI, MI LAMPARA

UN FANTASTICO COMPAÑERO ROBOTIZADO IMPULSADO POR UN ODROID-U3

por Jochen Alt

Una lámpara es prácticamente inútil durante el dia. Por sí misma, no habla, no puede bailar y es aburrida. Aquí es donde intento ayudar con mi nuevo proyecto llamado Luci. Luci es una lámpara autónoma con una webcam, un potente LED en la pantalla y cinco servomotores. Está controlada por un ODROID-U3. Una vez encendida mira a su alrededor, comprobando el entorno para ver si hay algún ser vivo, y luego hace lo que quiere. Echa un vistazo al siguiente video en http://bit.ly/1hLVtOt.

Mecánica

La parte más peligrosa de los proyectos es la mecánica, así que empecé con ella. Me compré una lámpara de escritorio, medí las dimensiones básicas y traté de darle un aspecto más infantil haciendo la pantalla más grande y los brazos más cortos. TurboCAD me ayudó a modelar la pantalla y las bisagras.

Tras haber pasado varias horas en el sótano cortando bisagras de madera de abedul multicapas, resultó que la fricción entre el brazo y la junta estaba demasiado alta. Así que, la reconstruí con rodamientos de bolas, volví al sótano, lo intento de nuevo y me doy cuenta de que los muelles necesitaban diferentes puntos de montaje, cambio esto, vuelvo a la oscuridad y aspirar polvo nuevamente. Sobre todo después añadir los rodamientos me alegré de haber utilizado TurboCAD, ya que fueron necesarios cuatro cortes por bisagra con agujeros muy precisos.

No quise repetir esa experiencia polvorienta con la pantalla, de modo que la impresión 3D se volvió muy tentadora y quise probarla de todos modos. No puedo dar demasiados detalles sobre esto, ya que acabo de entregar el modelo CAD a una tienda de impresoras 3D. Tuve que soportar sus desesperados intentos de venderme una impresora.

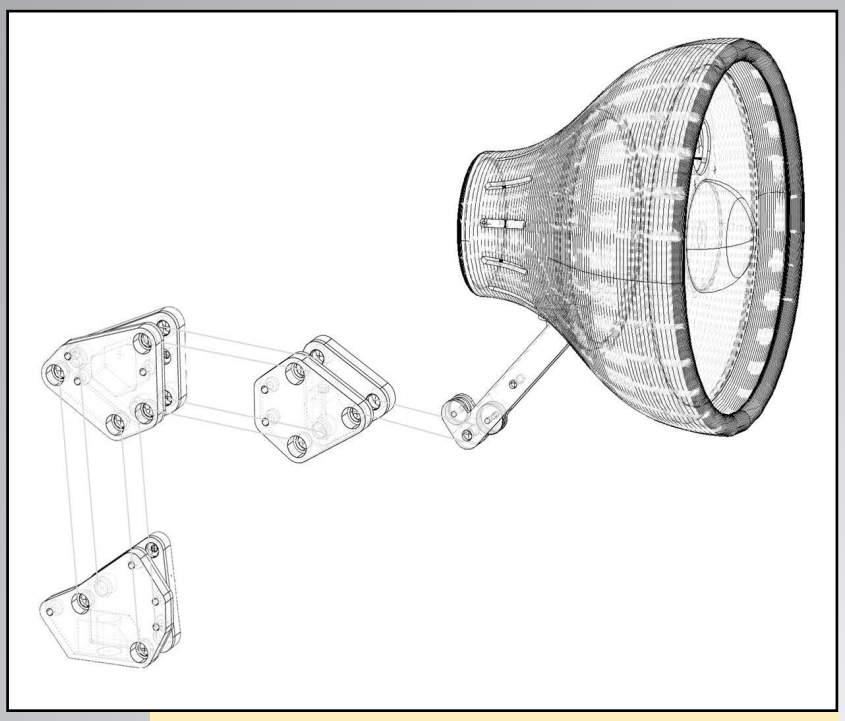


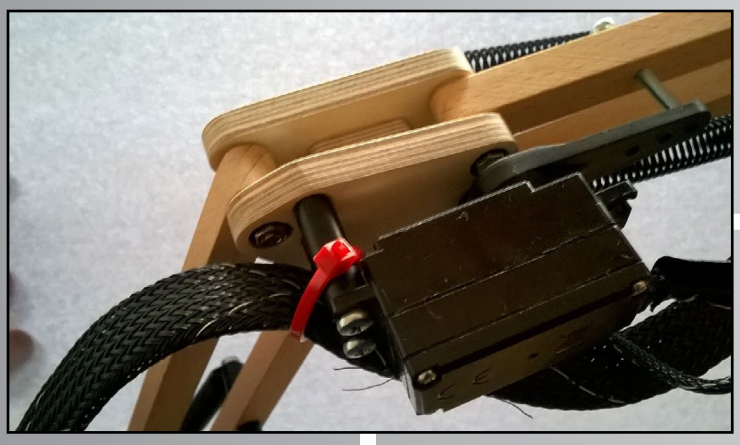
Diagrama CAD de Luci

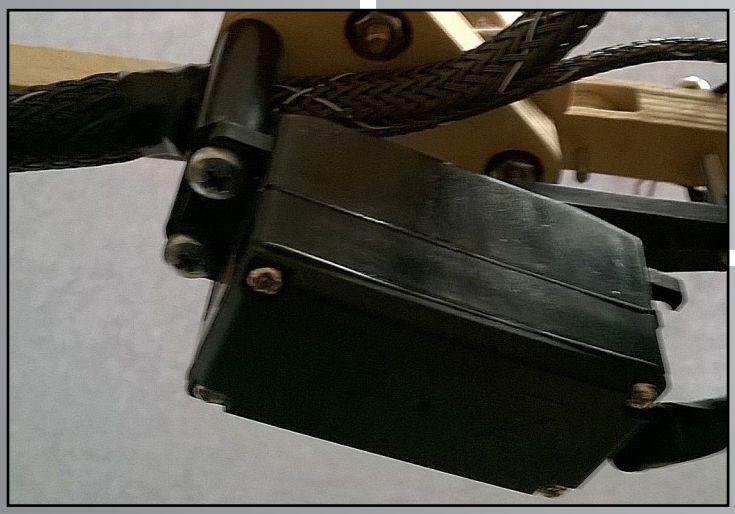
Hardware

Empecé con una BeagleBoard, pero muy pronto se hizo evidente que era demasiado lenta para el reconocimiento facial. Así que pedí un ODROID-U3. Por desgracia, no ofrece cinco salidas PWM para los servomotores, así que necesitaba un microcontrolador para la señal PWM de los servomotores y el potente LED. Además, el ODROID es sensible a los cambios de voltaje, así que tras haber conectado los servomotores a una placa independiente con su propia fuente de alimentación, los reinicios repentinos de ODROID desaparecieron. Se suponía que el ODROID con el Arduino iban a ser colocados en la base de la lámpara pero al final, no lo coloqué allí. Tenía que tener cuidado a la hora de colocar un ODROID caliente dentro de una caja de madera llena de tabillas.

Software

Primeros planos de las bisagras (conexiones)





El programa se ejecuta en Ubuntu usando C++ con múltiples hilos de ejecución con el fin de aprovechar al máximo los cuatro núcleos. El primer hilo toma la imagen de la cámara web; un segundo hilo coge estas imágenes e intenta buscar todas las caras que capta Luci. El resultado es una lista de rostros y Luci se centra en el más grande.

Un tercer hilo de ejecución se encarga del algoritmo de planificación de trayectoria, que produce una secuencia de puntos y orientaciones en el espacio tridimensional generado por determinados patrones. Cuando no se detecta ningún rostro, Luci ejecuta un patrón de búsqueda en busca de caras analizando el entorno hasta que encuentra un rotro. Después, Luci lleva a cabo un patrón simulando emociones reales, asintiendo con la cabeza sin saber por qué, fingiendo escuchar, acercándose o retirándose en función de los movimientos de la cara. Es como en la vida real.

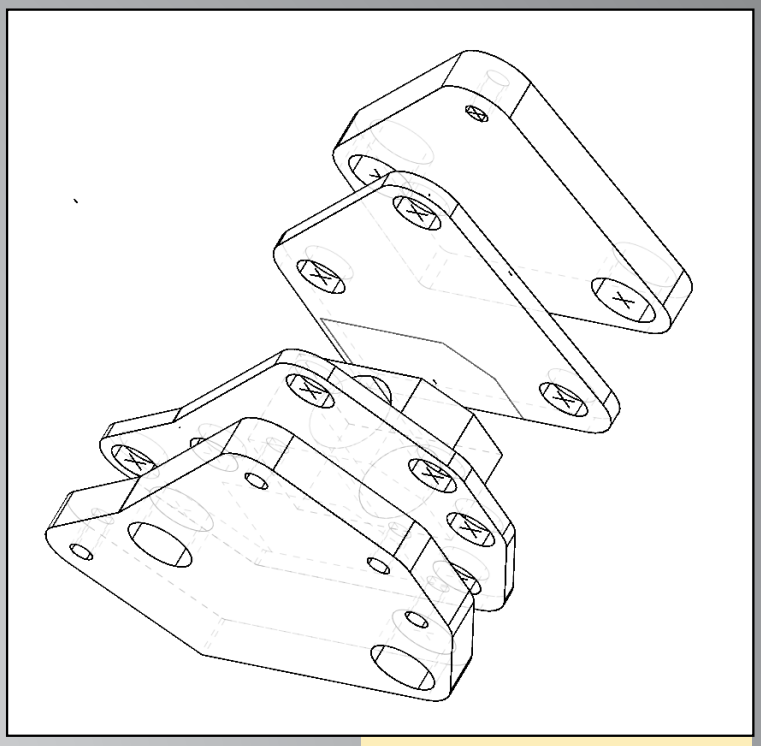


Diagrama detallado de la Bisagra

Planificar la Trayectoria

Implementar los patrones de trayectoria es bastante simple: siempre que Luci se queda sin puntos en cola para ser tratados, se activa un generador de puntos que está parametrizado con el patrón actual. Allí, se genera el siguiente punto de una secuencia de movimientos predefinida. En el caso del patrón que interactúa con una cara, esto es:

Retrocede rápidamente (sorpresa, el reconoce el rostro)
Se mueve despacio hacia la cara (primer contacto tímido)
Mira el rostro desde varios ángulos (inspección cercana)
Bajar y mira la cara desde la perspectiva de una rana (parece bonito)
Sube de nuevo y asiste la cabeza (fingiendo un acuerdo)

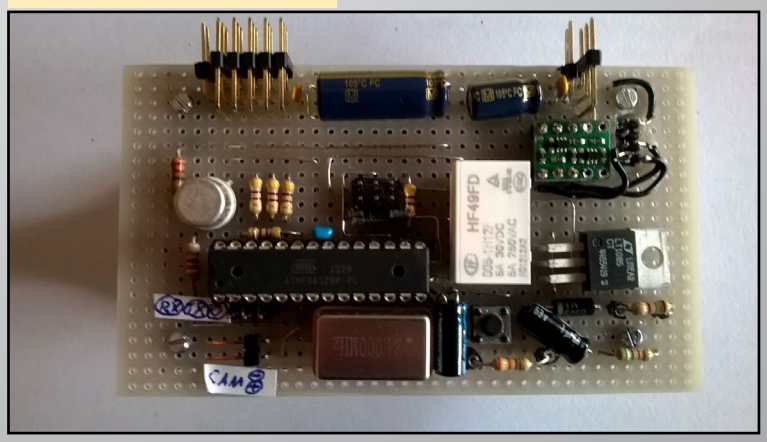
Las ideas fueron tomadas de Eliza (http://bit.ly/1Co34ab), pero codificadas en un robot en lugar de Emacs. Algunos patrones de movimientos especiales están predeterminados. Por ejemplo, cuando Luci empuja una caja de la mesa o se ve culpable por ver fotos guarras (1:34 y 2:00 en el video).

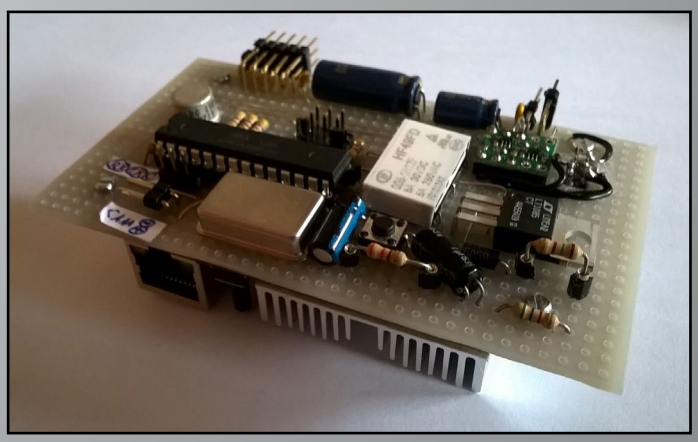
Por último, el círculo toma el punto de la trayectoria anterior y siguiente e interpola todos los puntos intermedios a 60Hz utilizando una curva de Bezier cúbica para suavizar el movimiento. Los puntos de apoyo de la curva Bezier son geométricamente derivados desde el punto (A) y (D) de la trayectoria por la regla que se muestra en la imagen: Puesto que cualquier polinomio con grado superior tiende a oscilar cuando los puntos de apoyo están demasiado lejos, me mantuve a una distancia constante de |BC|/ 3 a B resp. C.

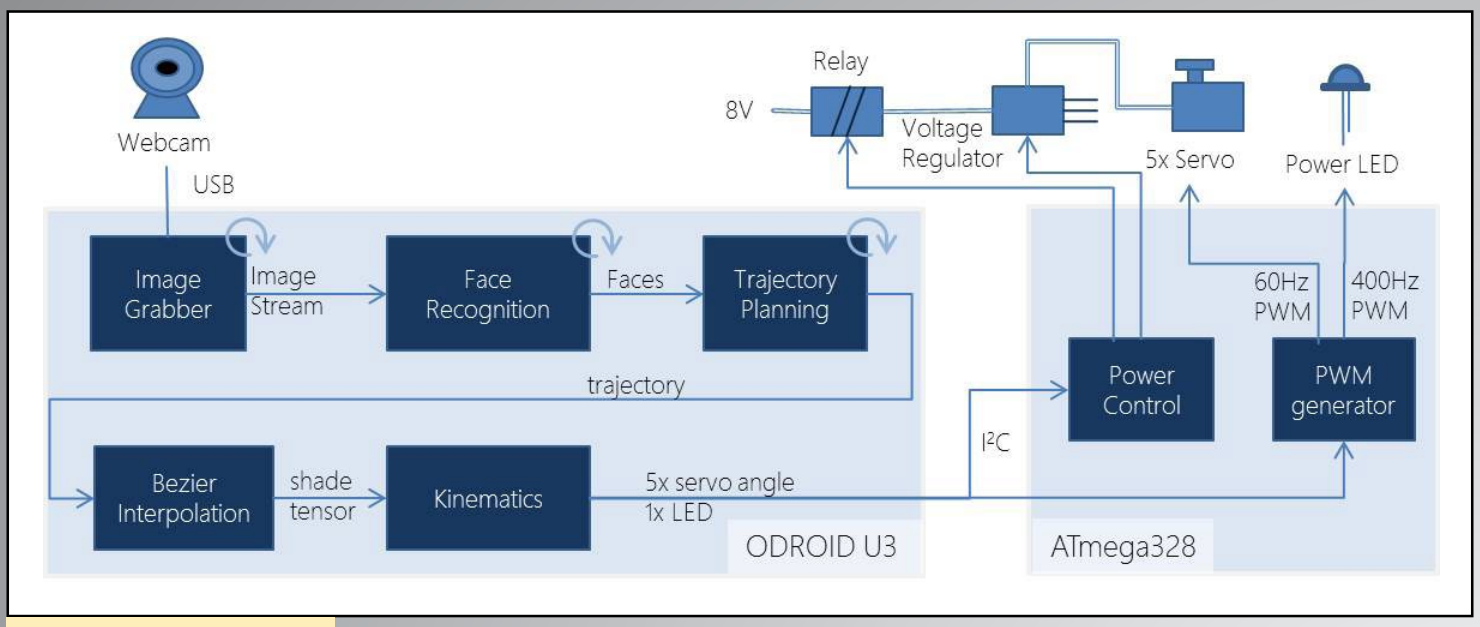
Inercia de masas

El último paso también calcula la aceleración de la pantalla, ya que la curva de Bezier no tiene en cuenta que, en total son 400 gramos los que se mueven. Como consecuencia, limité la aceleración de masas 1/2 g para prevenir aleteo causado por la construcción en su conjunto y el contragolpe de los servomotores. Esto se hace comprobando si la siguiente posición se puede alcanzar sin acelerar por encima del límite. Sino, la nueva posición se calcula tomando la posición actual y añadiendo la distancia máxima (sobre la base de la velocidad actual y acel-

Primeros planos del PCB







Estructura del Software

eración máxima limitada por 1/2 g) a lo largo del actual vector velocidad. Al final, la curva resultante abandona la curva Bezier donde es demasiada pronunciada. Los Robots profesionales lo hacen basándose en la inercia completamente modelada, pero llegados a este punto las matemáticas ya me estaban cansado, así que no lo intente.

Cinética

El resultado de todo esto es un punto 3D que se pasa al módulo de cinética que calcula los ángulos de todos los servomotores. La robótica de los libros de texto, que funciona así:

El algoritmo empieza con el punto/orientación (= tensor) del centro A del cabezal. El primer paso es calcular la posición B y C a través de la orientación del cabezal. Esto puede hacerse calculando el punto C relativo a la posición de A (Punto C-punto A), girando la orientación del cabezal (Rotación A) y añadiéndolo al punto A:

```
C := A.point + rotate(C.point-A.point, A.rotation)
```

Luego, el ángulo de la base en F que es el ángulo servo, se puede calcular con:

```
F.angle := atan2(A.point.z, A.point.x)
```

Los ángulos en E y D se calculan teniendo en cuenta la EDC triangular y calculando sus ángulos con la ley del coseno:

```
E.angle := 90° + acos( distance(E,D)2 +
  distance(E,C) 2 - distance(D,C)) /
  (2*distance(E,D) * distance(E,C) )
```

El ángulo en D se calcula de la misma manera:

```
D.angle := acos( distance(E,D)2 +
  distance(D,C) 2 - distance(E,C)) /
  (2*distance(E,D) * distance(D,C) )
```

El servomotor de C, que es el ángulo de orientación del cabezal alrededor del eje z, menos el ángulo CD del horizonte:

```
C.angle := A.rotation.z  + 270° -
```

Diagrama de puntos Bezier

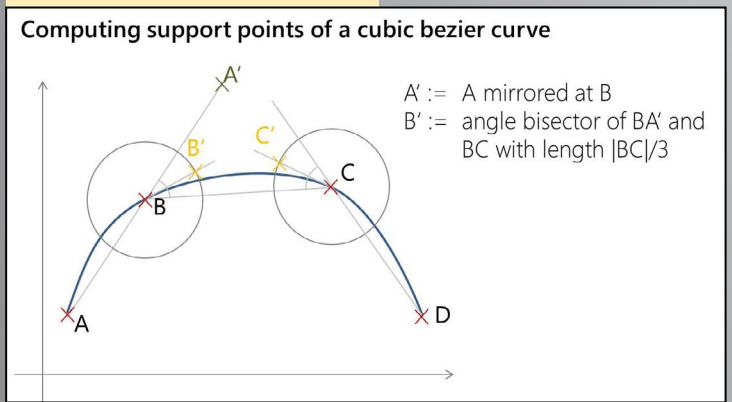
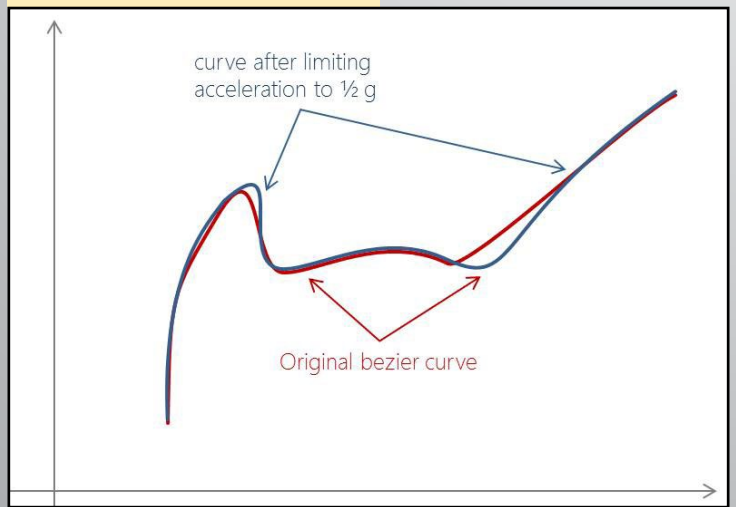


Diagrama límite de aceleración



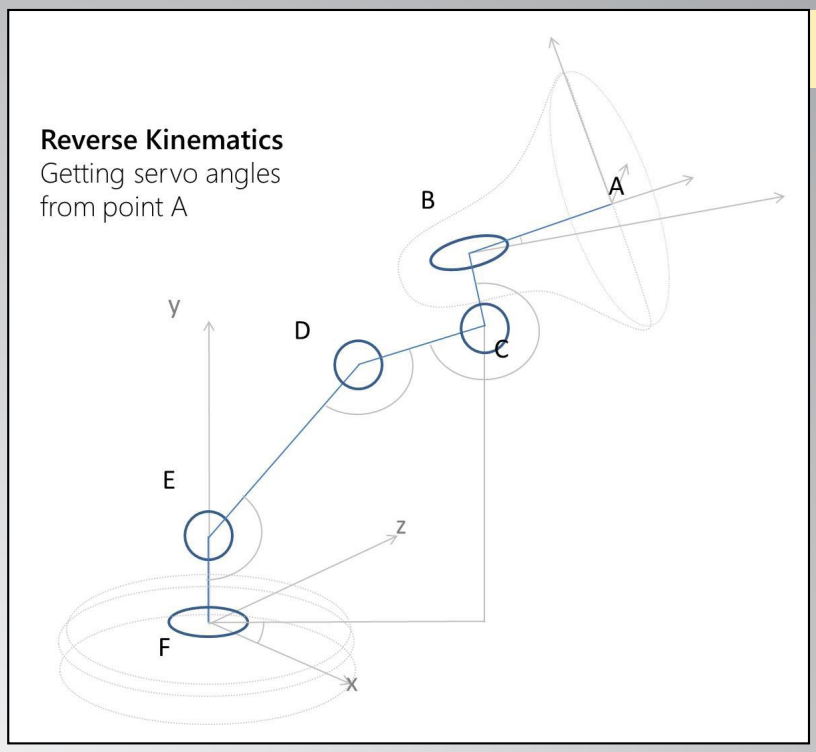


Diagrama de cinética

```
acos( C.point.y-E.point.y  /  C.distance(E) )
```

Estos ángulos pasan a través de I2C al ATmega, donde la librería Arduino genera una señal PWM de 60Hz para los servomotores. Al principio, estaba preocupado por el alto uso de la CPU de la cinética 3D, e intente usar números enteros de punto fijo y la trigonometría interpolada, ya que estaba acostumbrado a un ATmega de 24MHz. La sorpresa fue que usando flotadores y sen/cos sin caché o tablas de búsqueda no logre un mejor rendimiento.

Reconocimiento facial

El módulo de reconocimiento facial utiliza OpenCV 3.0 con clasificadores en cascada Haar. Aunque las nuevas cascadas LBP son más rápidas, tienen muchos positivos falsos, así que pensé que 10 fps con cascadas Haar era suficiente. Desde la posición en 2D del rostro detectado, la posición 3D se calcula con un tamaño estándar del rostro, lo cual funciona sorprendentemente bien. Después, el módulo de planificación de la trayectoria de Luci hace que se mueva hacia la cara si está muy cerca, para simular interés, luego se aleja si incumple una distancia mínima. El seguimiento de un rostro en tiempo real es complicado, ya que la captura de imágenes de la secuencia de vídeo y el reconocimiento facial tiene una latencia de 250 ms. Por lo tanto, el cálculo de la posición 3D del rostro necesita hacerse 250 ms antes en relación a la posición de la webcam. En consecuencia, cuando Luci se mueve rápido la imagen se vuelve borrosa, de modo que la orientación del cabezal es dirigida hacia la última posición estable del rostro hasta que Luci se mueva más lento y el seguimiento del rostro vuelve a ser posible.

El módulo de planificación de trayectoria calcula los siguientes dos puntos por adelantado para calcular la curva de Bezier. Por lo tanto, la posición del rostro detectado no es válida cuando el módulo de cinética envía una posición de los servomotres un par de segundos después. La solución paa por calcular de forma permanente la posición 3D actual y enviarla al módulo de cinética, para cambiar la orientación del cabezal hacia el rostro en tiempo real.

Conclusiones

Al igual que ocurre en mi trabajo, todo acaba llevando más tiempo de lo esperado. Sorprendentemente, el software y el hardware funcionan muy rápido, aunque montar la estructura y la mecánica sin ser una tarea demasiada pesada, con servomotres y muelles montados correctamente, me llevo a un par de fines de semana en el sótano. Las matemáticas fueron un autentico reto. Lograr el reconocimiento facial era la parte más sencilla. Los chicos de OpenCV hicieron un gran trabajo para que esto resultase realmente fácil. La parte más divertida es la planificación de la trayectoria, por ejemplo, cómo debe moverse Luci cuándo la webcam reconoce un movimiento del rostro.

Estoy pensando en mejorar Luci con un micrófono y un módulo de detección de ritmos permitiéndole hacer movimientos siguiendo la música, pero parece ser que la detección de ritmos es extremadamente complicada.

Componentes necesarios

DODROID-U3 ejecutando Ubuntu 04.14.02 con la último compilador g++

Software con OpenCV 3.0 y Boost I.57 como librerías base

ATmega 328 ejecutando firmware C ++ basado en librerías Arduino para los servomotores y I2C

Servomotores de Hitec: 77B (para girar la base y hacer muescas), 7954SH (puntal inferior, fuerte y caro), 7775MG (puntal superior, también es caro), 5065MG (gira el cabezal dentro de la pantalla)

Impresión 3D de un modelo TurboCAD hecho de ABS

Muelles de mi distribuidor local, 20 rodamientos de bolas, 0.5 m2 de abedul multicapa, y varios ejes de metal

Código fuente de http://bit.ly/1iq9amT

Otros proyectos

Hace dos años, me aburría bastante utilizando el Excel en la oficina y puesto que soy programador profesional, empecé a aprender electrónica y tecnología integrada y desarrolle mi primer robot, que se puede ver en http://bit.ly/1VHg7OX. Su nombre es Pablo, y es capaz de mantenerse sobre una pelota. ball.

ARJUNA

UN DISPOSITIVO QUE TE ENSEÑA A TOCAR EL PIANO BASADO ODROID

por Ilham Imaduddin

A mucha gente le encanta aprender a tocar el piano, pero convertirse en un experto en este instrumento de 88 teclas es bastante difícil. Muchas personas tienen problemas para reconocer y comprender la escala de una nota en relación a la posición de la tecla. Otro problema que se suele dar es cuando el músico intenta equilibrar los dedos de la mano derecha e izquierda mientras toca diferentes notas. En vista de estos problemas de aprendizaje tan comunes, tuvimos la genial idea de crear un dispositivo que ayudase a los estudiantes de piano con el aprendizaje de algunas de las técnicas básicas sobre cómo tocar el piano de una forma sencilla.

En este artículo, vamos a presentar a "Arjuna", un dispositivo que ayudará al alumno guiando cada dedo, de ambas manos, hacia la nota correcta de la tecla que se está tocando.

Componentes Arjuna

Arjuna consta de dos elementos principales, la MPU (Unidad de Procesamiento MIDI) y el módulo de la mano. Estos son los componentes para ambos elementos:

Unidad de Procesamiento MIDI
- ODROID-CI
- Transmisor receptor NRF24L01+
- Keypad

Módulo de la Mano
- Arduino Pro Micro
- Transmisor receptor NRF24L01
- Pequeños motores de vibración
- I célula de batería LiPo
- Regulador de intensidad 5V
- Anillos Impresos 3D
- Carcasa Impresa 3D

Los usuarios pueden utilizar este dispositivo, ya sea con un piano tradicional o un piano digital.

Arjuna en acción

La mecánica del Arjuna es sencilla: la MPU recibe los datos del instrumento, los procesa y da una orden a los Módulos de la Mano para guiar al usuario. El MPU y el teclado MIDI están conectados bien al cable USB del teclado o bien al puerto MIDI del teclado utilizando un adaptador USB-MIDI. Por lo tanto todos los teclados y pianos digitales con MIDI-USB o un puerto MIDI estándar son compatibles con este dispositivo.

Para la MPU, elegimos el ODROID-C1 por su flexibilidad, potencia y facilidad de uso. Se utiliza un módulo transmisor receptor basado en el IC Nordic nRF24L01+ para conectar de forma inalámbrica la MPU con los Módulos de la Mano. El transmisor receptor está conectado al ODROID-C1 a través de la interfaz SPI. El control de la MPU se lleva a cabo mediante un teclado, que está interconectado a la ODROID-C1 a través de la Entrada/Salida de Propósito General (GPIO). El código para E/S GPIO y SPI fue escrito para utilizar la librería WiringPi.

Módulo Mano Arjuna

El módulo de las manos es un conjunto de cinco anillos impresos en 3D, uno para cada dedo, con dos motores de vibración para cada anillo. Los motores de vibración se colocan al lado izquierdo y derecho de cada anillo. Esto se utiliza para guiar al estudiante hacía la tecla que debe ser presionada. Hemos utilizado los motores de vibración más pequeños que

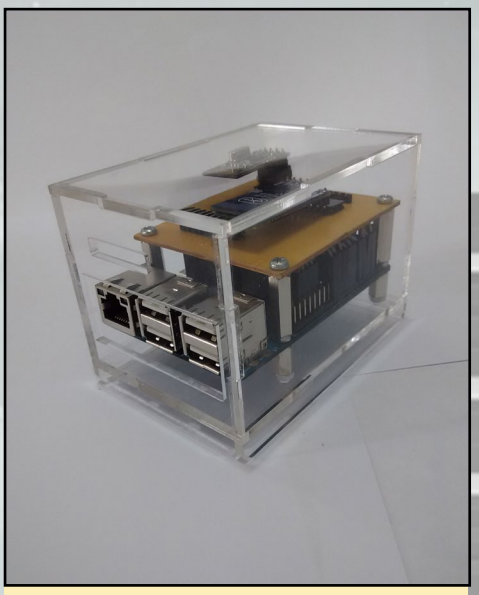


Figura 1 - ODROID-CI en una carcasa transparente

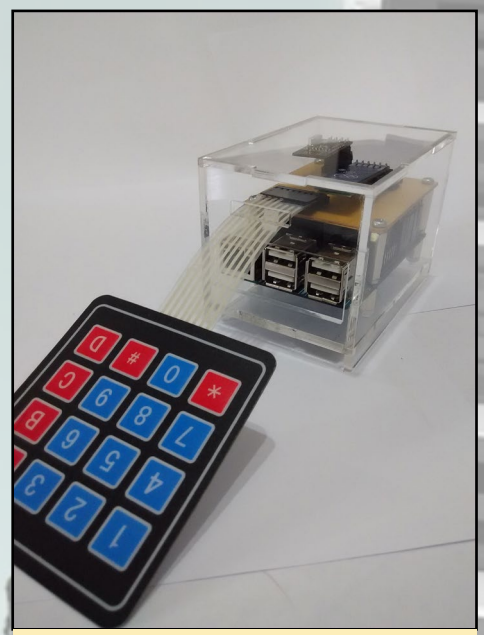


Figura 2 - ODROID-CI con el keypad

hemos encontrado con el fin de minimizar cualquier molestia al usuario mientras toca con este dispositivo.

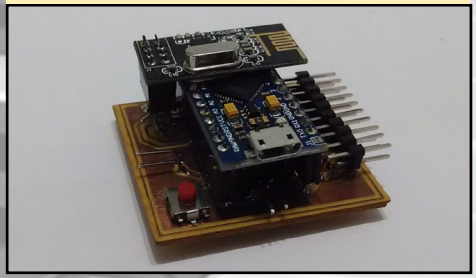
Los motores de vibración son controlados por un microcontrolador. Decidimos usar el Arduino Pro Micro por su pequeño tamaño y por el número de librerías disponibles para simplificar el desarrollo del software. Otro módulo transmisor receptor se utiliza para conectar los módulos de la mano con la MPU del ODROID-C1 de forma inalámbrica. Todos los componentes de los módulos de la mano son alimentados por una batería LiPo, sobrealimentados con un regulador de Voltaje de 5V. Utilizamos una batería de 180mAh, que es lo suficientemente pequeña como para coger en la pequeña carcasa, y tiene la capacidad suficiente para abastecer los módulos de la mano por un tiempo razonable. Todos los componentes, excepto los anillos se colocan dentro de una carcasa 3D. Los usuarios cuentan con una correa para colocar los módulos de la mano en sus muñecas, similar a un reloj. Arjuna usa dos módulos de mano, uno para cada mano.

Cómo funciona Arjuna

Arjuna se comunica con los teclados mediante el protocolo MIDI (Musical Instrument Digital Interface), un protocolo que permite a los instrumentos musicales digitales, ordenadores y otros dispositivos compatibles comunicarse entre sí. Con este protocolo, la MPU puede recibir datos, tales como notas y la velocidad desde el teclado, mientras que el usuario está tocando.

Arjuna tiene dos modos: el modo Escucha y modo Evaluación. Ambos modos requieren dos conjuntos de datos, uno para la canción y otro para el movimiento de los dedos. Las canciones se almacenan en archivos MIDI (.mid), que se pueden encontrar en varios sitios de Internet, o incluso puedes crear tu propia canción con un creador de canciones MIDI (hay un montón por ahí). Los datos del movimiento de los dedos se pueden crear desde el archivo de canción MIDI con un software llamado MidiFGR. En este software, puede configurar manualmente el dedo correcto para cada nota. Esta información es necesaria para ayudar al usuario a que toque la tecla adecuada, con el dedo correcto.

Figura 3- Unidad ProMicro



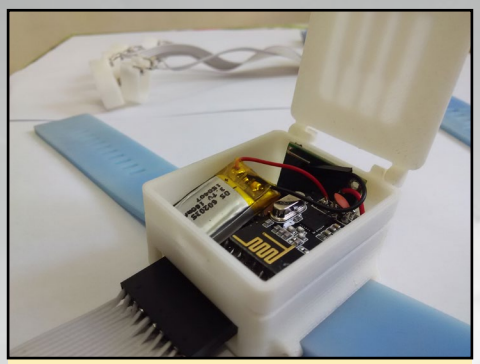


Figura 4 - Vista superior Módulo de Manos

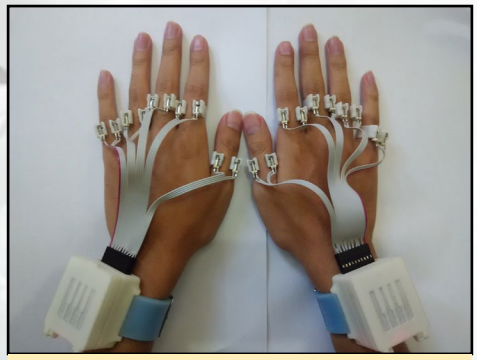


Figura 5 - Manos con módulos conectados

Figura 6 - Tocando el piano

En el modo Escucha, la MPU leerá la canción elegida y la reproducirá por medio del teclado. Al mismo tiempo, la MPU enviará comandos a los módulos de la mano, haciendo vibrar el dedo correcto para tocar las notas correctas. De esta manera, el usuario puede aprender la técnica del movimiento de los dedos de un modo adecuado.

En modo Evaluación, el alumno puede tocar el teclado y recibir ayuda. Mientras el usuario está tocando, la MPU evalúa los datos recibidos desde el teclado comparándolos con el archivo de la canción. Cuando el estudiante presione una tecla del piano equivocada, la MPU lo detecta y envia un comando al módulo de la mano. Este comando hará vibrar el motor, ya sea en el lado izquierdo o derecho del dedo correcto. El lado dependerá de la ubicación en la que se encuentre la tecla que debería haber sido presionada, en relación a la tecla pulsada incorrectamente. Por ejemplo: Si la tecla correcta se encuentra a la derecha de la tecla presionada erróneamente, el motor izquierdo vibra. Por otro lado, si la tecla correcta se encuentra a la izquierda de la tecla presionada erróneamente, el motor derecho será el que vibre. Por supuesto, este comportamiento puede ser invertido en la configuración de la MPU.

Sobre el proyecto Arjuna

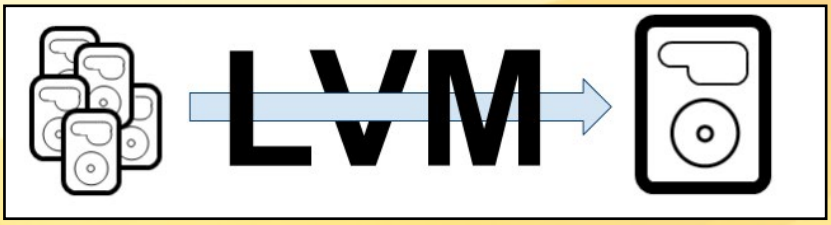
Somos un equipo de tres personas de la Universidad Gadjah Mada, Indonesia. Originalmente, este proyecto estaba destinado a ayudar a los discapacitados visuales a aprender a tocar el piano. Aprender a tocar el piano no es fácil para nadie, pero para ellos es aún más difícil. Este dispositivo fue creado para presentar a los discapacitados visuales algunas de las técnicas básicas para tocar el piano sin la necesidad de la vista. Con Arjuna, esperamos abrir una nueva ventana para los discapacitados visuales, una ventana hacia el magnífico mundo de la música.

Arjuna es de código abierto. Actualmente estamos modificando nuestro repositorio público para facilitar la instalación, http://github.com/ArjunaElins/Arjuna. Aquí encontrraslos esquemas, archivos 3D y los códigos.

FUNDAMENTOS SOBRE GESTION DE VOLUMES LOGICOS

A PARTIR DE AHORA GESTIONARAS DE UN MODO DIFERENTE TU ESPACIO

por David Gabriel



Nunca tenemos suficiente espacio de almacenamiento. Siempre soñamos con tener una unidad de almacenamiento más grande, un nuevo disco duro. Como aquella vez cuando contabas tus centavos y decidiste obtener una pequeña eMMC para tu primer ODROID simplemente para descubrir que lo que quiere ahora es espacio: mucho más espacio. ¡Estoy hablando con vosotros, los usuarios de XBMC!

Cuando necesitas un volumen más grande que el anterior, lo más simple es añadir un disco más grande, darle formato, copiar los datos al mismo y simplemente ser feliz, es lo que nos han enseñado desde siempre. Es lo que se suele hacer habitualmente, ya que en la mayoría de los casos, solemos tener una unidad de 1 TB USB conectada a nuestros ODROIDs.

Sin embargo, si te paras a pensar en ello, estas simplemente destinando un completo terabyte de datos para un único propósito. No te preocupes, estoy aquí para enseñarte qué puedes hacer mucho más con tu unidad. A partir de ahora, No necesitas utilizar todo el espacio de un único sistema de archivos, seguir tratando de adivinar el mejor esquema de particiones o peor aún, formatear el disco, cambiar el esquema de particiones y reinstalar todo simplemente porque compraste un nuevo disco duro.

En primer lugar, haz una backup de tus archivos. Sé que esto puede sonar obvio, pero la mayoría de nosotros no hacemos backup con regularidad, ¿no? Solemos tontear con la asignación de espacio, las particiones y los sistemas de archivos. Así que, incluso si no sueles estroper lo que tocas, siempre es una buena idea hacerla, para relajarse y saber que si llega el momento en el que dices "¡Vaya!, ¿dónde está todos mis datos?" puedes decir después: "No hay problema, lo tengo todo en mi copia de seguridad".

Para continuar, necesitarás algo de espacio libre. Puede ser cualquier unidad de disco en blanco que puedas tener, o esa vieja partición que no utilizas. Todo lo que necesitas es un dispositivo lógico en el que guardar los datos.

Instalación

Esa es la parte fácil. Sólo tiene que instalar LVM en tu sistema:

```
$ sudo apt-get install lvm2
```

Configuración

Aquí es donde se percibe la magia. Si nunca has oído hablar antes de LVM, debes saber que los volúmenes se dividen en volúmenes físicos, grupos de volumen y volúmenes lógicos. Los volúmenes físicos son tus particiones de disco reales. Los grupos volumen son donde se encuentra la totalidad de tu espacio en bruto, y los volúmenes lógicos son los equivalentes a las particiones normales a las que estás acostumbrado. Así que, vamos a crear nuestro primer volumen físico. Para ello, escribe lo siguiente en una ventana de terminal, donde /dev/sda1 es la partición que deseas "forma-tear" como LVM. Esto permitirá que pueda usarse en nuestro nuevo LVM.

```
$ pvcreate /dev/sda1
```

A continuación, vamos a crear el grupo volumen, donde rootvg es simplemente una etiqueta que asignas al grupo volumen, y /dev/sd1 es el volumen físico que acabamos de crear.

```
$ vgcreate rootvg /dev/sda1
```

Ahora, puedes ver el grupo volumen creado. Simplemente escribe:

```
$ vgs(information)
```

o

```
$ vgdisplay(attributes)
```

Ahora crearemos volúmenes lógicos para poder utilizar el espacio:

```
$ lvcreate -L 10G -n homelv
```

Esto creará un nuevo volumen lógico con 10 GB llamado homelv. Ten en cuenta que esto creará un nuevo dispositivo en /dev con la etiqueta dm-0, que internamente apuntará al disco físico /dev/sda. También crea un enlace en /dev/mapper/rootvg-homelv hacia /dev/dm-0 para que no tener que recordar los números en el futuro. Para comprobar el estado de las particiones lógicas, escribe:

```
$ lvs(information)
```

o

```
$ lvdisplay(attributes)
```

A partir de ahora puedes proceder con cualquier método. Crea un nuevo sistema de archivos y montarlo para guardar datos en el mismo. Recuerda utilizar el dispositivo creado anteriormente:

```
$ mkfs -t ext4 /dev/mapper/root-
vg-homelv
$ mount /dev/mapper/rootvg-homelv
/home
```

Ahora tienes tu propio disco ejecutándose con LVM. Pero sólo verá la diferencia con respecto a una partición normal cuando necesites cambiarlo.

Tras configurar LVM y teniendo muchos sus usuarios que se conectan y guardan archivos bajo el directorio /home, con el tiempo llenaras el disco, a menos que lo aumentes. ¿Cómo? En primer lugar, desmonta el sistema de archivos:

```
$ umount /home
```

Aumenta primero el volumen lógico:

```
$ lvextend -L +10G /dev/mapper/
rootvg-homelv
```

Esto le dará un extra de 10 gigabytes sobre el volumen lógico. Sin embargo, todavía tenemos que aumentar el sistema de archivos para que coincida con el volumen lógico. Puedes hacer esto con:

```
$ e2fsck -f /dev/mapper/rootvg-
homelv
$ resize2fs /dev/mapper/rootvg-
homelv
```

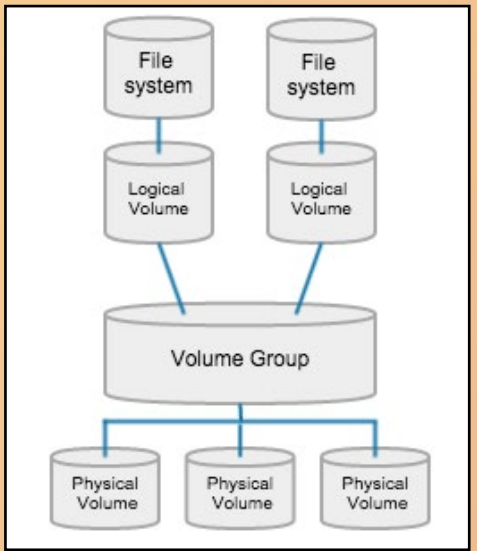
El primer comando comprobará si hay alguna inconsistencia en el sistema de archivos, y el segundo cambiará el tamaño con el nuevo tamaño del volumen lógico. En algún momento, también necesitas un nuevo sistema de archivos para instalar una aplicación. Sin embargo, recuerdas que asignarte todo el espacio que habías dejado en el grupo volumen a homelv. ¿Ahora qué? Sólo tiene que añadir una nueva partición:

```
$ vgextend rootvg /dev/sda2
```

Puedes añadir tantas particiones/discos como desees al grupo volumen con el fin de crear una gran "piscina" de espacio libre. Después, simplemente repite los pasos para crear/ampliar los volúmenes lógicos y los sistemas de archivos.

Ahora ya sabes cómo hacer más sencilla la creación de esquemas de particiones. Una cosa a tener en cuenta es que puedes hacer esto con todos los sistemas de archivos. Basta con crear un punto de montaje temporal, mueve todos los datos y luego ajustar /etc/fstab para los dispositivos correctos. Es importante tener en cuenta que no puedes alterar /media/boot, ya que es donde el gestor de arranque busca los primeros archivos que son enviarlos a la RAM y arranca el sistema. Todo esto sucede mucho antes de que se inicien los servicios LVM, por lo que el sistema no arrancará si lo haces, así que ten cuidado.

Estos son los conceptos básicos para la creación de LVM en tu sistema. Se puede hacer mucho más, pero esto debería ser suficiente para empezar a experimentar y descubrir todas sus posibilidades.



DESARROLLO QT5

DESARROLLA UNA SIMPLE INTERFAZ DE USUARIO DE APLICACION

por Christopher Dean

En este artículo se describe cómo utilizar el lenguaje de programación QT5 y la imagen Dream de desarrollo QT5 para crear una simple interfaz de usuarios para aplicaciones. Esto es útil para personas que desarrollen software quioscos, reproductores multimedia, equipos médicos y otros sistemas Linux embebido. Vamos a pasar directamente a la creación de una sencilla aplicación QT5, y luego aprenderemos a configurar la sesión para ejecutar la aplicación en el arranque.

Empecemos

Lo primero es descargar y grabar la última imagen Dream de desarrollo QT5 en tu tarjeta SD o módulo eMMC. La imagen se puede descargar desde http://bit.ly/1PMDwKP.

Figura I – Escritorio Dream para desarrollo QT5

Aplicación de Ejemplo

Tras arrancar la imagen, ¡podemos crear nuestra primera aplicación QT5! Abrir Qt Creator desde el cajón de aplicaciones en la parte superior izquierda del entorno de escritorio Lubuntu seleccionando Menu -> Programming -> QtCreator. Tras lanzar QtCreator, selecciona File -> New File or Project desde la barra de acciones, luego, escoge Qt Quick Application.

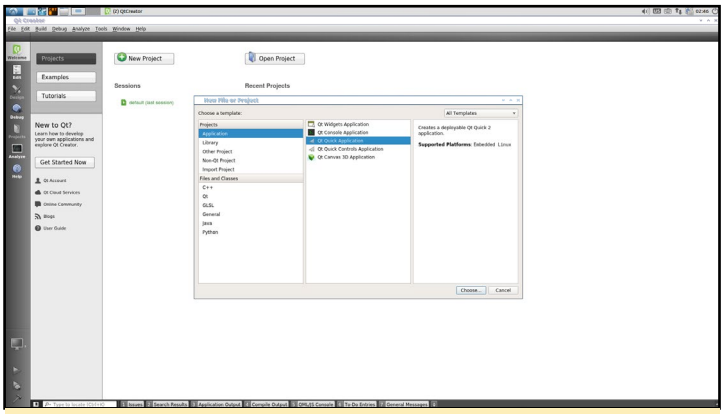


Figura 2 - Creando un nuevo proyecto

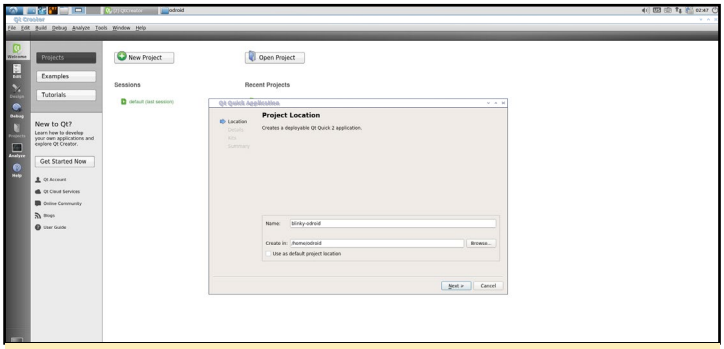


Figura 3 – Asignando un nombre al Proyecto

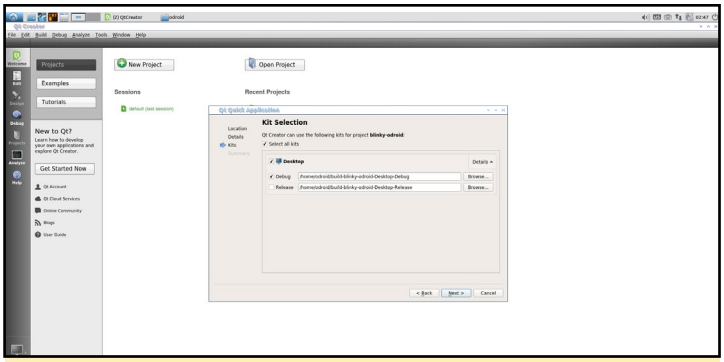


Figura 4 – Seleccionando el kit de escritorio

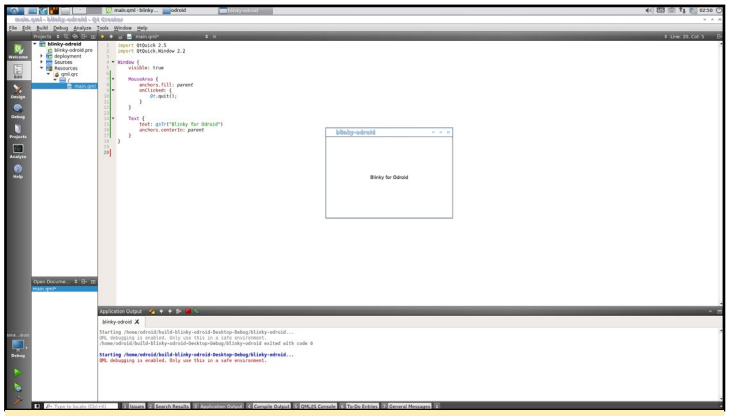


Figura 5 - Cambiando el texto a "Blinky para ODROID"

A continuación, asigna al proyecto un nombre y elige el directorio, configura el proyecto QT5 utilizando el kit "Desktop".

Sigue los cuadros de diálogo hasta que se cree el proyecto. A continuación, bajo los proyectos a la izquierda, navega y abre Resources -> qml.qrc-> / -> main.qml. Tras abrir el archivo, cambia el texto dentro del elemento Text , de "Hello World" a "Blinky for ODROID". Luego, crea y ejecuta el Proyecto pulsando Ctrl + R. Después haz clic en la aplicación para cerrarla, abre un terminal y dirígete al directorio proyectos. En la parte superior del directorio del proyecto, ejecuta el siguiente comando con el objeto de clonar la clase QGPIO

```
$ git clone http://github.com/Tpimp/qgpio.git
```

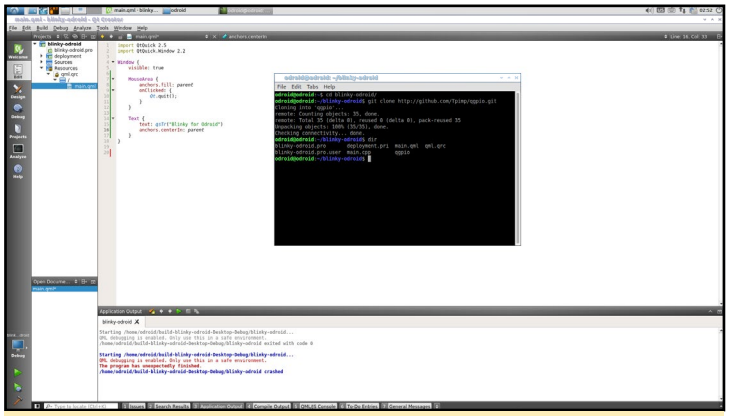


Figura 6 - Clonando del Proyecto QGPIO

Vuelve a la aplicación QtCreator y haga clic derecho en "blinky-ODROID", selecciona "Add Existing Directory…" en el menú desplegable y luego, selecciona los directorios como se muestra en la Figura 7. Tras agregar el directorio, navega y abre Sources -> main.cpp. Modifícalo para sea similar a la Figura 8. Ten en cuenta que este paso está relacionado con el uso de QGPIO, sin escribir una aplicación general QT5.

A continuación, podemos crear la parte "Blinky" de la aplicación. Vuelve a abrir el archivo main.qml para editarlo.

Tras importar el módulo com.embedded.io definido en

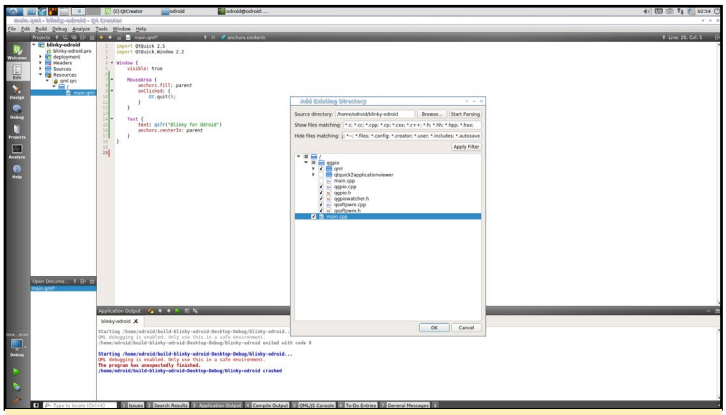


Figura 7 – Añadiendo la clase QGPIO al Proyecto

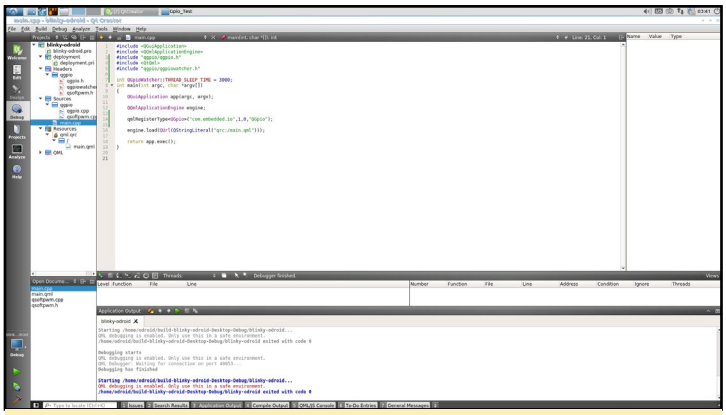


Figura 8 – Modificando la clase QGPIO

C++, crea un elemento QGpio y ajusta el ID a "gpio200" y el número a "200". A continuación, crea la función callback que se invocará cuando el componente esté completo. En la función, ajusta direction GPIO a "Out", cambía edge a "Falling", ajusta active_low a "false" y value a "false". Luego, en el objeto MouseArea, reemplaza las declaraciones de la función onClicked para alternar el valor GPIO:

```
gpio200.value = !gpio200.value;
```

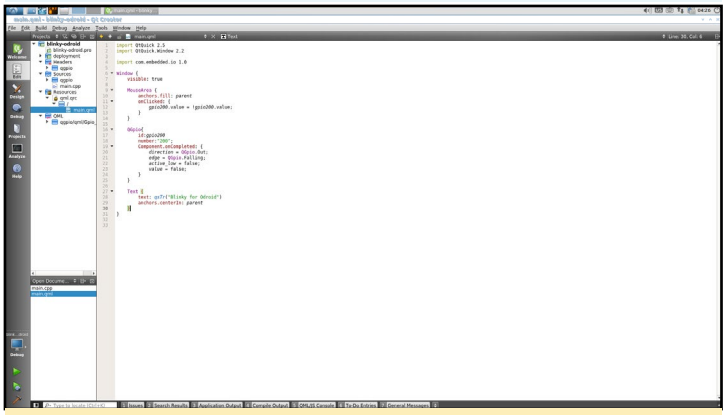


Figura 9 – Modificando el archivo main.qml

Ejecuta el programa y haz clic en el centro, luego observar cómo cambia el valor GPIO 200 al escribir el siguiente comando en una ventana de terminal:

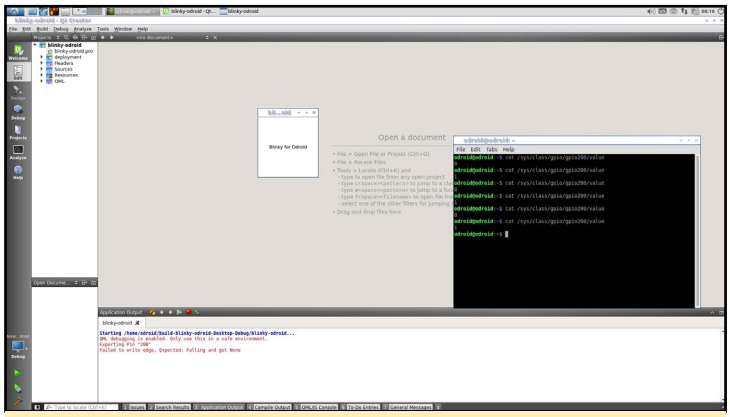


Figura 10 – Comprobando el valor de GPIO 200

```
$ cat /sys/class/gpio/gpio200/value
```

¡Ahora ya estás listo para el juego! Conecta un LED de bajo amperaje al pin 200 y a la puesta a tierra, como se muestra en la Figura 11. Si no dispones de LED de bajo amperaje, utiliza una resistencia o un transistor con una fuente de alimentación independiente. El tema sobre el voltaje GPIO y los LEDs es demasiado para este artículo. ¡Asegúrate de no sobrecargar tu puerto GPIO con un gran LED! Cablea tu LED y luego ejecuta la aplicación. El programa debería hacer parpadear el LED cuando se haga clic en el centro. El siguiente paso es hacer que nuestra aplicación Blinky ponga en marca la aplicación X11, una tarea simple para un ODROID, pero divertida.

Figura 11 - Esquema del cableado Blinky-ODROID

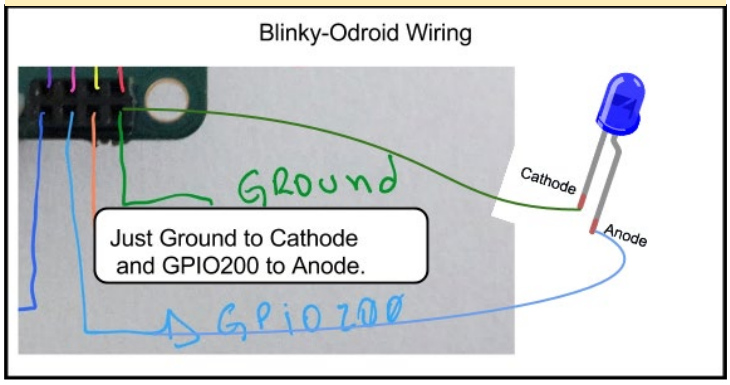
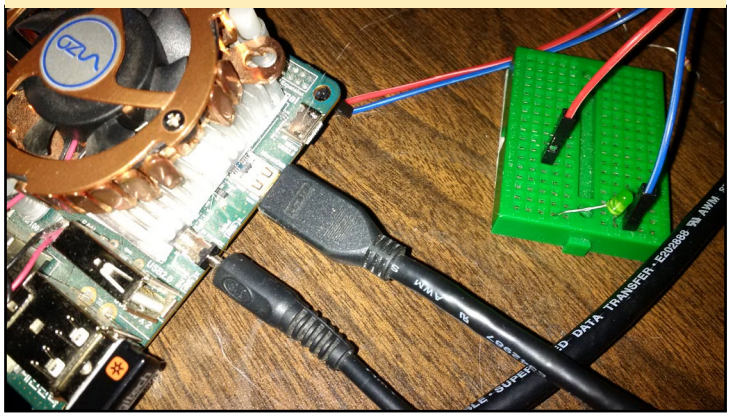


Figura 12 - Ejemplo del cableado Blinky-ODROID



Crear una configuración de escritorio

Hay muchas maneras de configurar X11 para que se inicie con una única aplicación. Técnicamente, tenemos que hacer que el entorno de escritorio sea "Blinky" en lugar de Lubuntu y eliminar la opción correspondiente para asegurarnos de que el usuario sólo arranca con "Blinky". Por otro lado, crear el archivo de configuración del escritorio nos permite alternar entre probar nuestra aplicación y el entorno de desarrollo. Crea un archivo de configuración blinky.desktop Xsession en el directorio /usr/share/xsessions como se muestra en la Figura 13. El siguiente paso es compilar la aplicación blinky-ODROID y colocarla en /usr/bin/. Sin embargo, si ejecutamos blinky-ODROID en su formato actual, no podemos cerrar la sesión cuando intentamos salir del programa. Antes de probar el entorno de escritorio Blinky, debemos hacer algunos cambios.

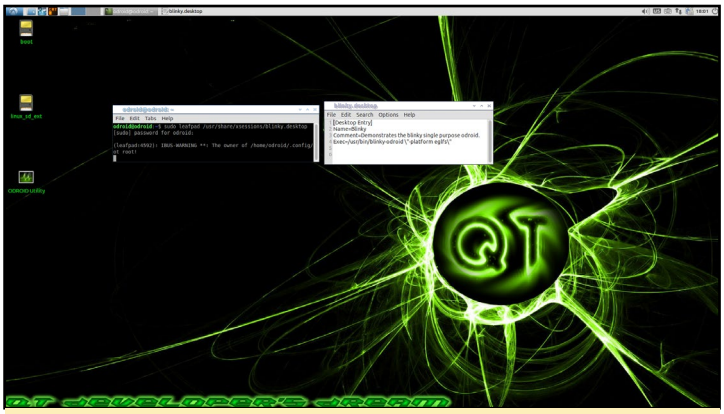
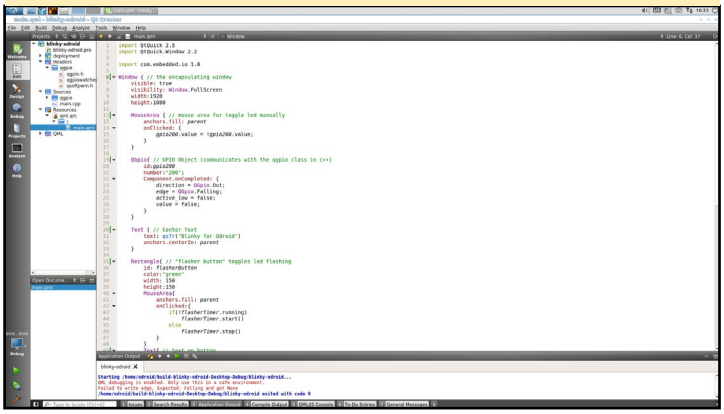


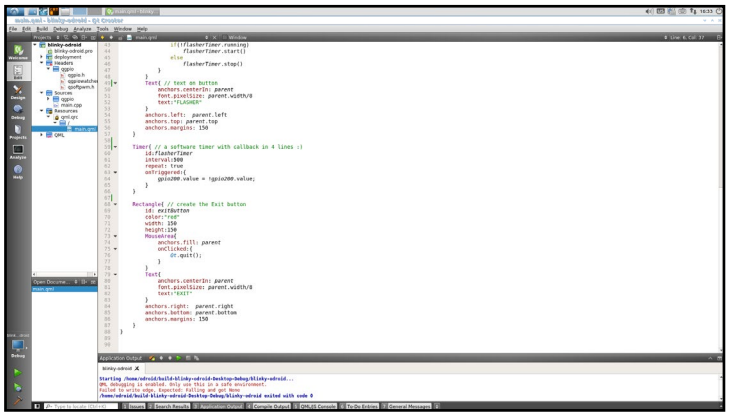
Figura 13 - Ejemplo de archivo de configuración blink.desktop Xsession

Añadir funciones

En primer lugar, tenemos que añadir algo de código para hacer que nuestra demostración blinky sea automática así como manual. En primer lugar, añade el ancho y el alto para que coincida con la resolución actual. Luego, agrega un botón de luz intermitente, que es simplemente un rectángulo con otro MouseArea. El clic del ratón iniciará y detendrá un temporizador que se define en main.qml, como se muestra en la Figura 14.

Figuras 14 y 15 - Código de la interfaz Blinky-ODROID





Lo lógico es que si el temporizador no está en marcha, empecemos sino nos detenemos. En primer lugar, proporciona al rectángulo una altura, una anchura y un color. Añade texto al botón y colócalo en algún lugar de la pantalla. Debajo, crea el "Temporizador intermitente" y dale un intervalo de 500 ms, que equivale a la mitad de un segundo. El temporizador se ajusta para que repita el proceso y crea una comunicación de vuelta "onTriggered" que indica que "durante la ejecución, se activa y desactiva el GPIO 200 cada medio segundo".

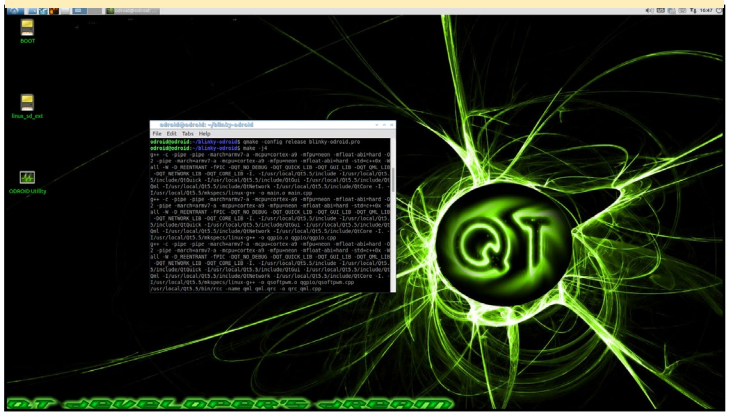
Por último, añade un botón de salida, que es muy similar al botón de luz intermitente, pero de color rojo y ubicado en la parte inferior derecha. En lugar de activar y desactivar temporizador, el botón de salida cerrará la aplicación. Si esto no sustituye el entorno de escritorio, deberíamos cerrar en lugar de salir. Salir te hace volver a la pantalla LightDM donde puedes elegir a dónde ir después.

Creación de binarios

Hasta ahora, QtCreator probablemente ha estado llevando a cabo una serie de desarrollos y operaciones ocultas en el modo "debug". Para poder usar la aplicación, queremos compilarla en forma de "versión estable". Abre una ventana de terminal y navega hasta el directorio del proyecto. Una vez en el directorio, ejecuta los siguientes comandos:

```
$ qmake -config release blinky-ODROID.pro
$ make -j4
```

Figura 16 - Compilando el binario Blinky-ODROID



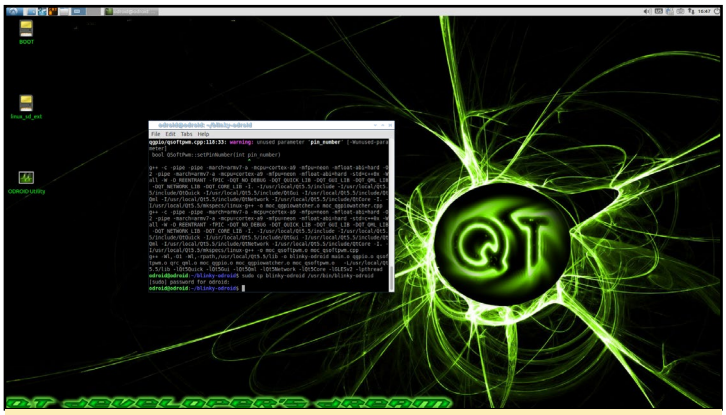


Figura 17 – Copiando el binario-Blink ODROID a /usr/bin

Se debe compilar sin problemas. Si encuentras errores, descargar el código fuente de http://bit.ly/1KsPX0z. Tras compilarlo, instalarlo copiando el ejecutable en el directorio /usr/bin

```
$ sudo cp blinky-ODROID /usr/bin/blinky-ODROID
```

Probando Blinky

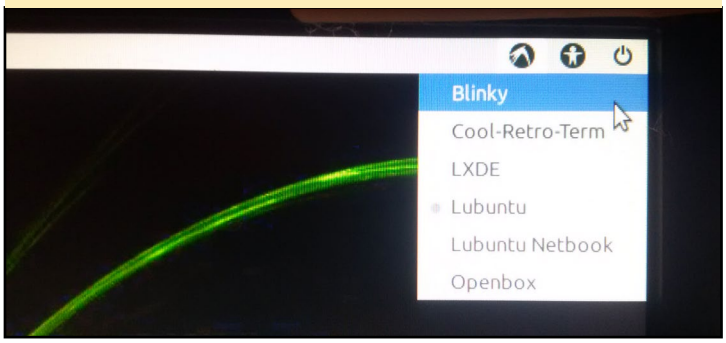
Realizar pruebas es muy fácil una vez creado el archivo de configuración Xsession. En primer lugar, cierra sesión en Lubuntu con el botón normal de Finalizar sesión. Deberías volver a la pantalla LightDM a la espera de iniciar sesión. En la parte superior derecha, haga clic en el icono para abrir la lista de entornos de escritorio, selecciona "Blinky". A continuación, inicia sesión con las credenciales comunes ODROID/ODROID. Si todo ha ido bien, debería ver en pantalla el ejemplo blinky-ODROID. Este es un proyecto muy simple, pero se puede utilizar para desarrollar entornos de escritorio más complejos. El mayor inconveniente es la falta de un administrador WiFi y otros cuadros de diálogo de configuración.

Lecturas recomendadas

Si desea obtener más información sobre QT5 o sobre el desarrollo de entornos de escritorio, aquí tienes algunos enlaces:

Entornos de escritorio: http://bit.ly/1MyywH5
Linux Embebido: http://bit.ly/1LMrksN
Servicio de escritorio QT5: http://bit.ly/1fWjtxk
Entorno de escritorio Plasma 5: http://bit.ly/1EAUtax

Figura 18 – Seleccionando el entorno de escritorio Blinky usando LightDMM



SAMIIO

DESARROLLA POTENTES APLICACIONES E/S CON FACILIDAD

por Venkat Bommakanti

Hace aproximadamente un año, Samsung presentó una plataforma segura de intercambio de datos basada en OpenCloud, denominada Samsung Architecture Multimodal Interactions IO (SAMIIO). Con esta plataforma, se pueden desarrollar aplicaciones y servicios usando simples APIs y SDKs para enviar, recibir y mostrar visualmente diversos tipos de datos. Estas aplicaciones se pueden escribir utilizando lenguajes como Python, Ruby, Javascript y PHP.

A través de un ejemplo real, este artículo pone de manifiesto la simplicidad de la plataforma SAMIIO, usando un ODROIDC1+ y una aplicación Python para publicar los datos meteorológicos recogidos a través de un ODROID-SHOW y la placa meteorológica ODROID.

Requisitos

- Un ODROID-C1+. Aunque este artículo utiliza un C1+, la técnica puede aplicarse a cualquier dispositivo ODROID que sea compatible con ODROID-SHOW y la Placa Meteorológica Odroid.
- ODROID-SHOW y Placa Meteorológica ODROID
- Accesorios para C1+ tales como un cable HDMI, cable Ethernet CAT 5E+ o Adaptador 3 WIFI, módulo 2 Bluetooth (BT 4.0+). Recomendados PSU, batería RTC y ODROID-VU o Pantalla táctil 3,2”
- Módulo eMMC 5.0 de 16 GB+ con la última imagen de escritorio Lubuntu específica para C1+ y/o una MicroSD Clase 10 de 16GB+ con lector de tarjeta SD
- Una red en la que el dispositivo tenga acceso a Internet y los foros ODROID
- El acceso por red al C1+ a través de utilidades como PuTTY, FileZilla, TightVNC Viewer (MS Windows 7 +), terminal (Mac, Linux), etc. desde un escritorio para pruebas.

Configuración del sistema

La configuración se muestra en la Figura 1.

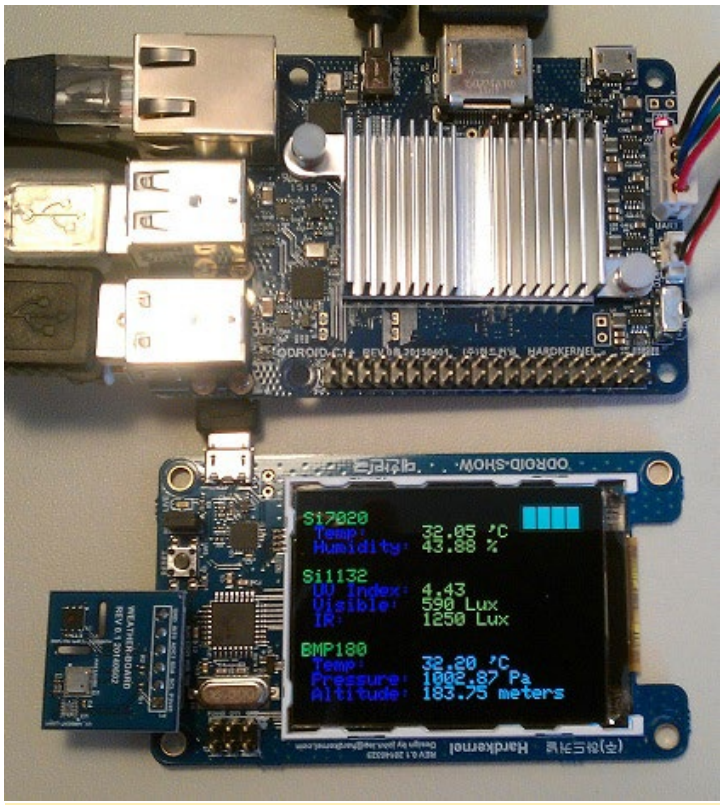


Figura I: Configuración de C1+ para SAMIIO

Instalar Lubuntu

Instala la última imagen para C1+ en el módulo eMMC y colócalo en el C1+. Con la pantalla VU conectada, arranca el sistema. Ejecuta ODROID Utility y ajusta la resolución de pantalla a 800p. Reinicia y luego, expande la partición de ins-talación para utilizar la eMMC al completo seleccionando la opción “Resize your root partition”. Reinicia y vuelva a ejecutar de nuevo ODROID Utility, configura y actualiza el resto de cuestiones importantes del sistema, reinicia después. Asegúrate de estar siempre conectado con el usuario por defecto “odroid”, a menos que se especifique lo contrario. Para las imágenes más recientes, tendrías que ejecutar los siguientes comandos por orden para actualizar el sistema:

```
$ sudo apt-get autoremove &&
  sudo apt-get update
$ sudo apt-get dist-upgrade
$ sudo apt-get install \
  linux-image-c1
```

El último comando es para coger el último kernel del C1+, por si acaso no ha sido incluido. Apaga el dispositivo y conecta todos los accesorios y cables al C1+, incluyendo SHOW y la placa meteorológica. Reinicia. Comprueba el puerto USB asociado al ODROID SHOW con el siguiente comando:

```
$ ls -lsa /dev/ttyUSB*
0 crw-rw---- 1 root dialout 188,
0 Aug  12 10:25 /dev/ttyUSB0
```

Compruebe la versión del sistema desde un terminal para asegurarte de que tiene la última (como la siguiente):

```
$ uname -a
Linux c1-1 3.10.80-122 #1 SMP
PREEMPT Mon Aug 10 20:27:04 BRT
2015 armv7l armv7l armv7l GNU/
Linux
```

Actualizar el esquema de la Placa meteorológica

Aunque el esquema de la placa Meteorológica no es necesario para actualizar la página web SAMIIO con los datos meteorológicos, es útil para validar visualmente la precisión de los datos enviados. A continuación se muestran los pasos para cargar un esquema modificado, que hace que los datos se presenten bien organizados y puedan localizarse con rapidez. Instala Arduino IDE para C1+, si no está presente.

```
$ sudo apt-get install arduino
$ cd ~ && mkdir zBU && cd zBU &&
mkdir sami && cd sami
$ git clone https://github.com/
hardkernel/ODROID-SHOW.git
$ cd ODROID-SHOW/weather_board
```

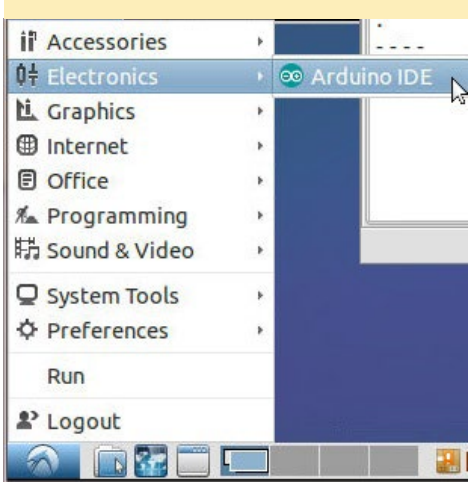
Asegúrate de que el archivo del esquema weather_board.ino está presente en ~/zBU/sami/ODROID-SHOW/firmware/weather_board/weather_board.ino

Crea un archivo vacio llamado weather_board-ino-changes en el directorio actual, copia el siguiente contenido y guardarlo:

```
20c20
< const char version[] = "v1.3";
---
> const char version[] = "v1.4-
vb";
74c74
<             tft.setCursor(250,
200);
---
>             tft.setCursor(200,
200);
88,89c88,89
<             tft.println("Temp :
");
<             tft.
println("Humidity :");
---
>             tft.println(" Temp:
");
>             tft.println(" Hu-
midity:");
94,96c94,96
<             tft.println("UV In-
dex : ");
<             tft.
println("Visible :");
<             tft.println("IR
:");
---
>             tft.println(" UV
Index: ");
>             tft.println(" Vis-
ible:");
>             tft.println("
IR:");
101,103c101,103
<             tft.println("Temp :
");
<             tft.
println("Pressure :");
<             tft.
println("Altitude :");
---
>             tft.println(" Temp:
");
>             tft.println(" Pres-
sure:");
>             tft.println(" Alti-
tude:");
246c246
<     tft.setCharCursor(7, 11);
---
>     tft.setCharCursor(11, 11);
248c248
<             tft.println(" *C
");
---
>             tft.println(" 'C
");
264c264
<     tft.setCharCursor(7, 2);
---
>     tft.setCharCursor(11, 2);
266c266
<             tft.println(" *C
");
---
>             tft.println(" 'C");
282c282
<     tft.setCharCursor(10, 7);
---
>     tft.setCharCursor(11, 7);
287c287
<     tft.setCharCursor(5, 8);
---
>     tft.setCharCursor(11, 8);
```

Actualiza el esquema con este archivo

Figura 2: Iniciar Arduino IDE



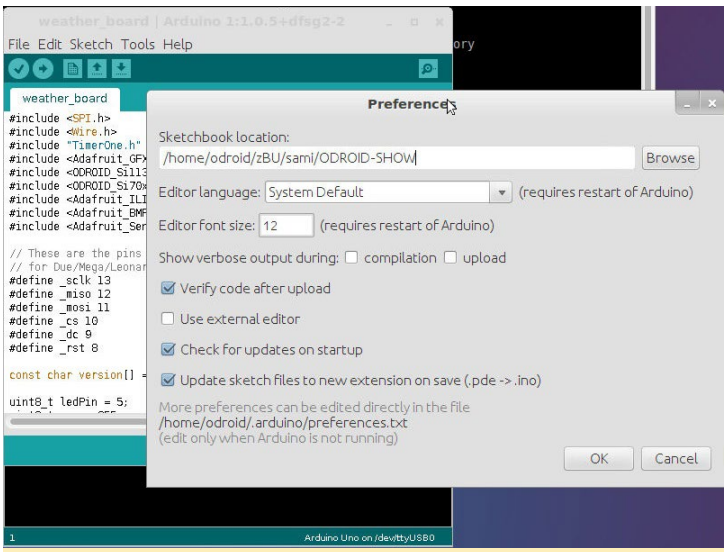


Figura 3: Preferencias de Arduino

usando el siguiente comando. Luego, inicia Arduino IDE como se muestra en la Figura 2. Selecciona Preferences en el menú Archivo y selecciona la carpeta ODROID-SHOW en "Sketchboot location", tal y como muestra la Figura 3.

Figura 4: Ubicación de las librerías Arduino

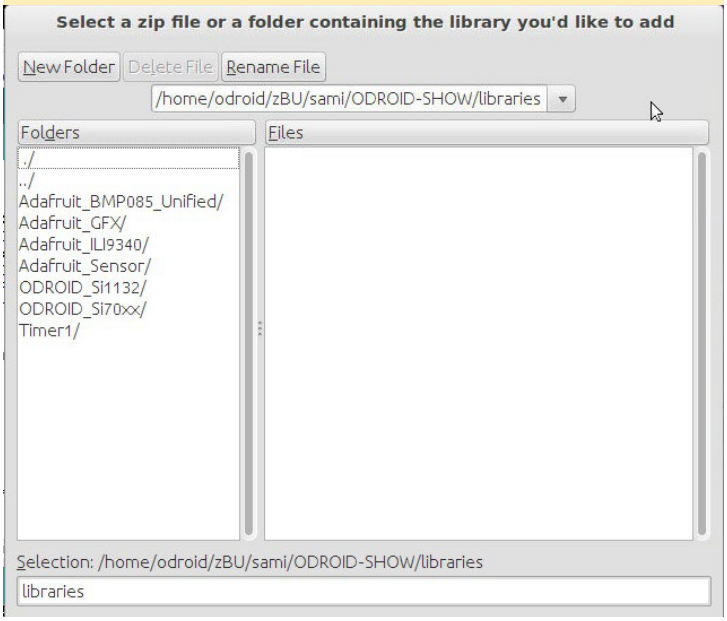
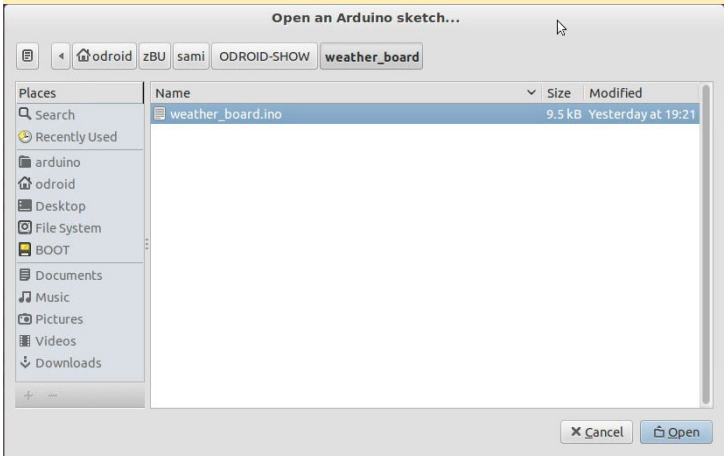


Figura 5: Esquema libre




```
$ patch -input weather_board.ino patch.txt
```

Para agregar las librerías adecuadas, Selecciona la opción de menú Sketch → Import Library → Add Library y apunta a ~/zBU/sami/ODROID-SHOW/libraries como se muestra en la Figura 4. Abra el esquema usando menú File → Open como se muestra en la Figura 5.

El esquema (código) debería aparecer en la ventana. El conector en el ODROID-SHOW (cerca del botón Reset) debe estar instalado. Selecciona el puerto serie utilizando el elemento de menú Tools → Serial Port → /dev/ttyUSB0. Sube el esquema haciendo clic en el icono circular con flecha a la derecha. El progreso de la carga será similar al mostrado en la Figura 6.

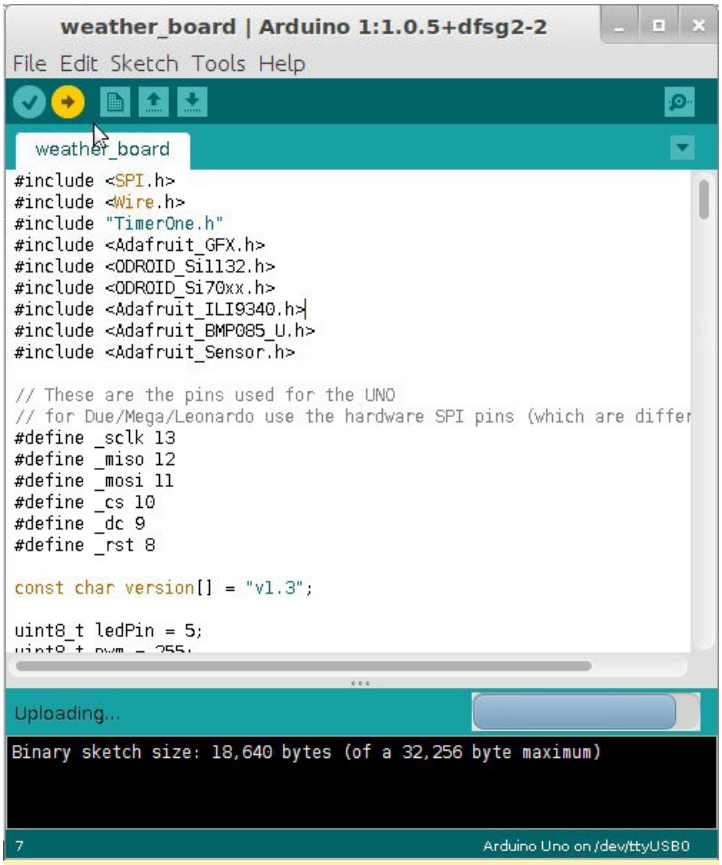
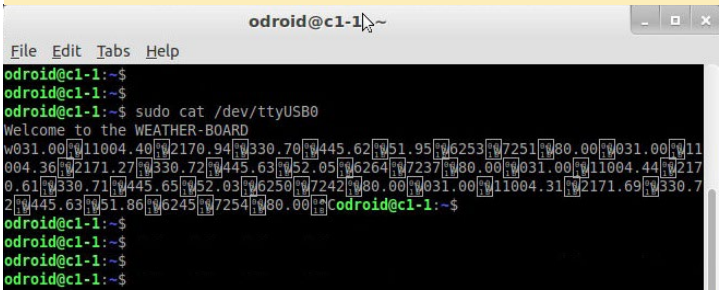


Figura 6: Progreso de carga del esquema

Deberías ver en el ODROID-SHOW como se visualizan los datos meteorológicos. También puede comprobar si puedes acceder a los resultados desde la línea de comandos (Figura 7).

Figura 7: Resultado Raw en/dev/ttyUSB0



El formato de los resultados es muy útil para escribir la aplicación Python y poder enviar los datos meteorológicos a la página web SAMIIO

Configuración del dispositivo y Cuenta SAMIIO

Los detalles de la plataforma SAMIIO lo puedes encontrar en su web. Accede a la página de configuración de la cuenta SAMIIO en http://bit.ly/1ds2ONj. Deberías ver una página como la se muestra en la Figura 8. Puesto que no disponemos de una cuenta de usuario, no podemos conectarnos todavía. Haga clic en el en-

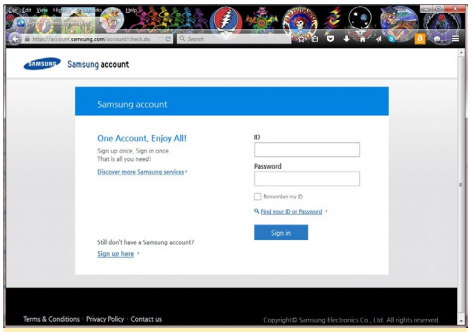


Figura 8: Acceder a la cuenta SAMIIO

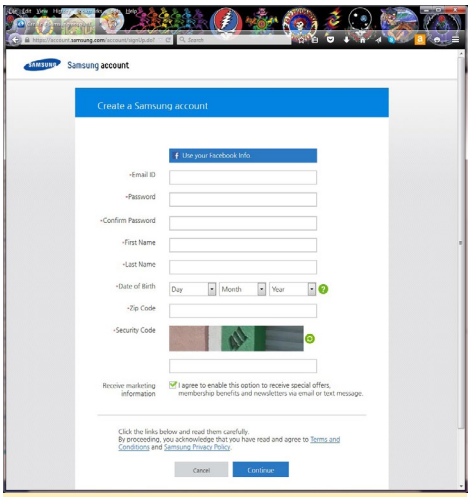


Figura 9: Configuración de la cuenta SAMIIO

lace Sign up here, verás una página como la que se muestra en la Figura 9.

Introduce la información relativa a tu cuenta y haga clic en el botón Continue. Vuelve a la pantalla de inicio de sesión (Figura 8), ingresa la información de inicio de sesión, haga clic en el botón Sign In. Aparecerá una página, como la que se muestra en la Figura 10

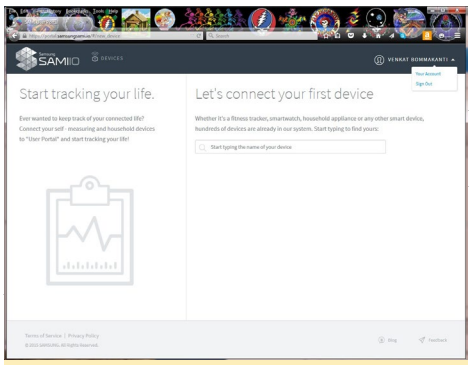


Figura 10: Pantalla Post-login

Aparecerá una página como la que se muestra en la Figura 11. Haga clic en

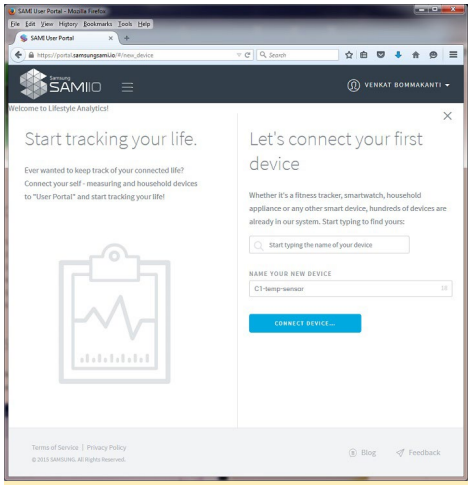


Figura 11: Creación del dispositivo

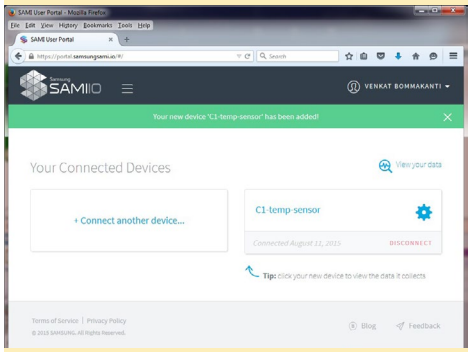


Figura 12: Pantalla principal del dispositivo

el botón azul DISPOSITIVO CONNECT. Te llevará a una nueva página, como la que se muestra en la Figura 12.

Haz clic en el icono azul en forma de piñón para crear algunos IDs para el proceso de comunicación. Se mostrará una página como la que parece a la Figura 13. Ten en cuenta que el ID de dispositivo es generado por el sistema. Haz clic en el enlace GENERATE DEVICE TOKEN. Aparecerá una pantalla como la que se muestra en la Figura 14. El sistema es el

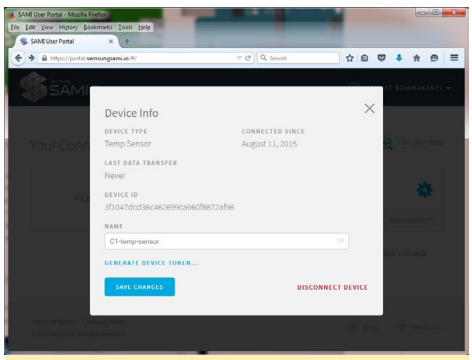


Figura 13: ID del dispositivo

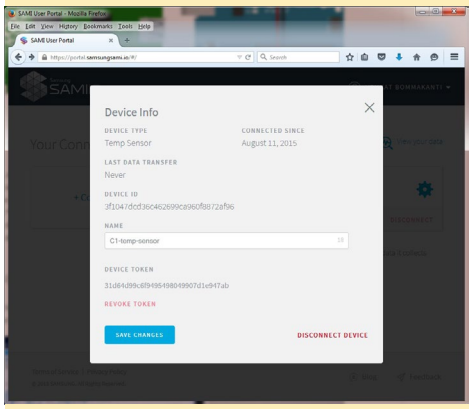


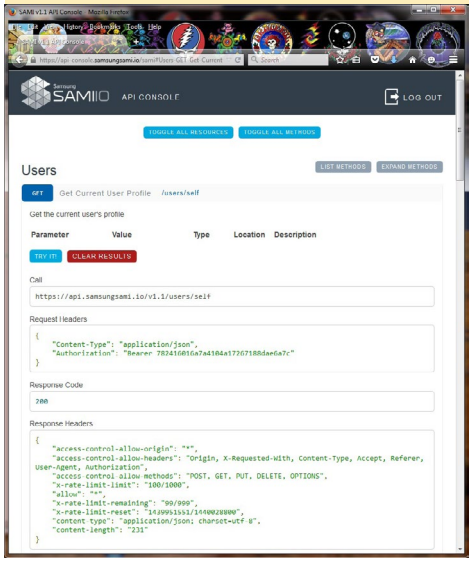
Figura 14: Ficha del Dispositivo

que genera la ficha del dispositivo. El ID del dispositivo se utiliza sobre todo en el proceso de comunicación.

La página web SAMIIO proporciona un mecanismo para obtener datos del dispositivo (bearer-id para la autenticación) y probar la API SAMIIO para enviar datos. Antes de escribir una aplicación para enviar datos, se puede probar la API desde este sitio web.

Accede al sitio de prueba API en http://bit.ly/1NeVs1I. Inicia sesión u-

Figura 15: Información del usuario



sando tus credenciales. En Users, haz clic en el enlace Get Current User Profile. Aparecerá una página con un botón azul TRY IT. Haz clic en él y aparecerá un resultado como el que muestra en la Figura 15.

El ID (para fines de autenticación) es útil para presentar los datos meteorológicos a través de una aplicación. A continuación, recorre la página hasta la sección Messages. Introduce algunos datos de temperatura simples en formato JSON, como se muestra en la Figura 16. Utiliza el ID de dispositivo obtenido anteriormente y escribe el valor entero (número de milisegundos desde el 1 de enero 1970) de la fecha actual en el campo ts. Haz clic en el botón azul TRY IT. Aparecerá una pantalla como la que muestra en la Figura 17. El cuerpo de la respuesta muestra los datos de temperatura enviados correctamente, con un ID de respuesta del mensaje.

Figura 16: Entrada API de mensajes

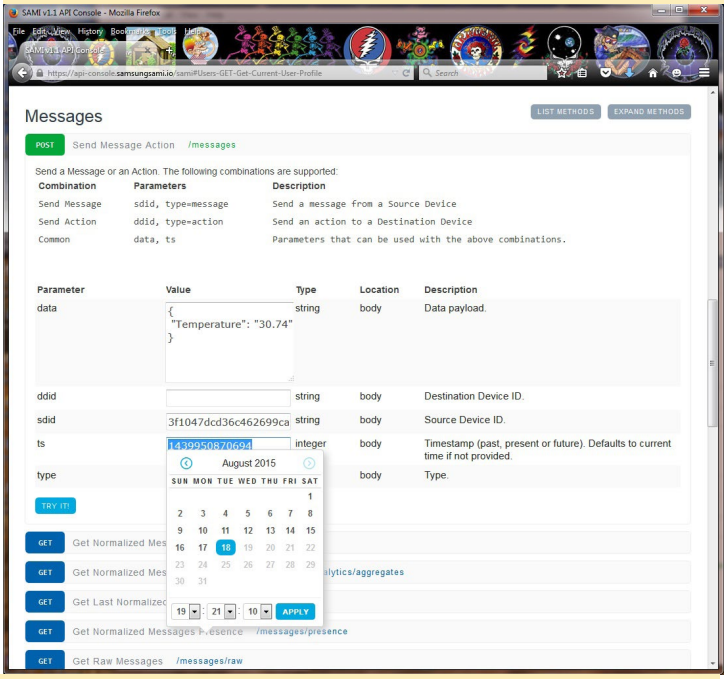
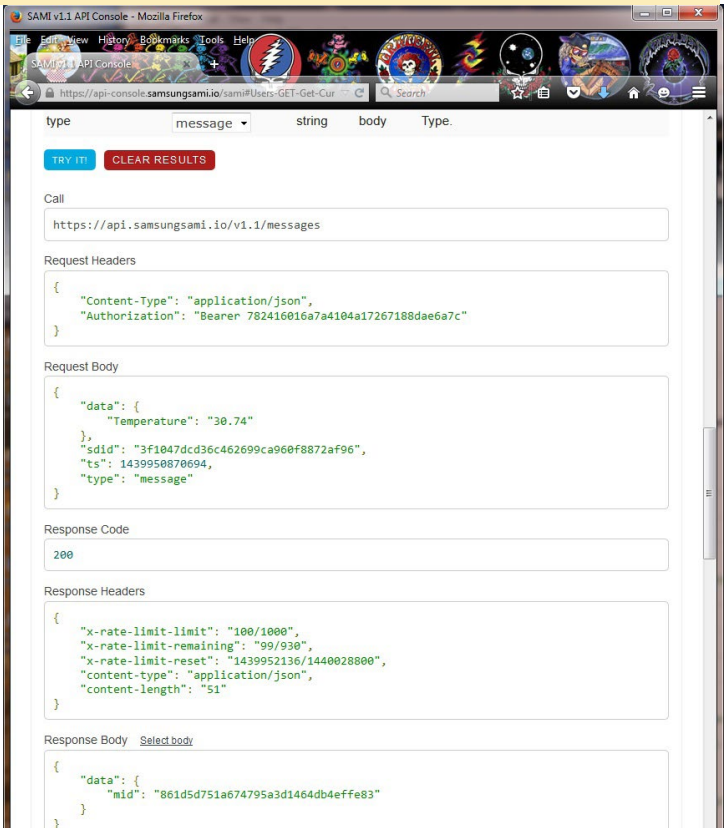


Figura 17: Salida API de mensajes



Requisitos previos de Python para SAMIIO

Contamos con Python 2.7.6 instalado en las imágenes estándar del C1. Las versiones de Python anteriores a 2.7.9 tienen restricciones en su módulo ssl que limitan la configuración que urllib3 puede aplicar. Para hacer frente a esta cuestión, instala los siguientes componentes:

```
$ sudo apt-get install libffi-dev libssl-dev
$ sudo apt-get install python-pip python-dev build-essential
$ sudo pip install pyopenssl ndg-httpsclient pyasn1
```

Usar el paquete python urllib produce código muy extenso. Instala un paquete para simplificar el código Python usando los comandos:

```
$ cd `/zBU
$ git clone git://github.com/kennethreitz/requests.git
$ cd requests
$ sudo python setup.py install
```

Ejemplo de Aplicación SAMIIO

Con la información obtenida hasta ahora, podemos crear un script en Python llamado sami-req-client.py en la carpeta: ~ /zBU/sami/. Asegúrate de que tiene permisos de ejecución (chmod 755). Este es el contenido de ese script. Dedica unos minutos para ver la información recopilada hasta el momento y que es usada en este script:

```
#!/usr/bin/python

#
# sami-req-client: Util to post weatherboard info
# to Samsung's SAMI service.  Presumes
# weatherboard works off ttyUSB0.
#
# (c) Venkat Bommakanti
# 08/05/15, CA, USA
#
# Free to use at your own risk.  No warranties implied.
# import sys, time
```

```
import serial, json, requests

# SAMI service & device info
sami_url = "https://api.sam-
sungsami.io/v1.1/messages";

# bearer: device-token: use your
bearer token here
bearer = "Bear-
er 75bced8e1e456095673d-
a70f375e187a1";

# sdid: device-id (source-id); :
use your device token here
sdid = "3f1047dcd36c-
345678760f8872af96";

# weather board ttyport: check on
your setup
weather_board_ttyport = "/dev/
ttyUSB0";
ttyport_speed = 500000;

# headers
headers = {
      "Content-type": "applica-
tion/json",
      "Authorization": bearer
}

# initialize
close_ok = True

#
# We'll be reading only 8 chars
off tty port.
# The data (char) pattern will be
something like:
#   wnabcdefg
# where 'wn' could range from
'w0' to 'w8', and
#   'abcdefg'
# could be a string equivalent of
any number between
#   '00000.00' and '99999.99'
# and any integer in between.
#

# 1 char length
A_CHAR = 1

# to clear a 10-char buffer
cl_phrase = ['\0', '\0', '\0',
'\0', '\0', '\0', '\0', '\0',
'\0', '\0' ]

# place holders
EMPTY_STR = ""
temp_val = EMPTY_STR
pres_str = EMPTY_STR
alti_str = EMPTY_STR
time_val = 0

try:
    iter = 0

    # setup serial port
    ser = serial.Serial(weather_
board_ttyport, ttyport_speed,
timeout=1)
    print "\n0: TTY-port: " +
ser.name
    ser.close()

    while True:
       if (close_ok):
              # open device to
read weather data
              sys.stdin =
open(weather_board_ttyport, 'r')
              close_ok = False
              print "\n1: opened
tty"
              # warning: no
timeout, could hang.
              # wait 10 secs for
init
              time.sleep(5)

              # first get welcome
lines
              wl_line = sys.
stdin.readline()
              print "\n2: " +
wl_line + "\n"
              # now start read-
ing data
              data_coll = 0
              iter = iter + 1
              wb_char = sys.
stdin.read(A_CHAR)
              print "\n3: " +
wb_char + "\n"
              while ((wb_char ==
'w') and (data_coll < 3)):

                     # clear
buffer that will hold weather
board data
                     wb_phrase =
cl_phrase

                     # read all
chars for a given piece of data.
                     # '\''
(decimal 27) is delimeter
                     idx = 0
                     while
(ord(wb_char) != 27):
                            wb_
char = sys.stdin.read(A_CHAR)

print "\n4: " + wb_char + ", ord:
" + \
      str(ord(wb_char)) + "\n"
                            wb_
phrase[idx] = wb_char
                            idx
= idx + 1
                            if
(idx > 8):

 break
                     # null ter-
minate
                     if (idx >
0):
                            wb_
phrase[idx-1] = '\0'
                     print "\n5:
" + "".join(wb_phrase) + "\n"

                     # typically
data arrives in order:
                     #   w0, w1
and then w2
                     print "\n6:
" + str(wb_phrase[0]) + "\n"

                     # skip the
1st char in buffer as it is
                     # just a
code for data type (temp, press,
```

```
alti).
                        if (wb_
phrase[0] == '0'):
                                #
found start of weatherboard data
- w0 -
 # for temperature

temp_str = "".join(wb_phrase[1:])

print "\n7: " + temp_str + "\n"

data_coll = data_coll + 1
                        elif (wb_
phrase[0] == '1'):
      # found pressure data

 press_str = "".join(wb_
phrase[1:])

print "\n8: " + press_str + "\n"

data_coll = data_coll + 1
                        elif (wb_
phrase[0] == '2'):
      # found altitude data

alti_str = "".join(wb_phrase[1:])

print "\n9: " + alti_str + "\n"

data_coll = data_coll + 1
                        else:
                                #
ignore other pieces of data

 print "\n10: ignoring next piece
of data\n"

                        if (data_
coll < 3):
                                #
read next piece of data
                                wb_
char = sys.stdin.read(A_CHAR)

 print "\n11: getting next piece
of data\n"
                        else:

 # found all 3 pieces of desired
data.  ignore others.
                                #
time to quit this iteration

print "\n12: all 3 pieces of data
collected\n"

                # close tty read
                close_ok = True
                sys.stdin.close()
                print "\n13:
closed tty"

                # timestamp (int)
to be sent to SAMI service
                time_val =
int(time.time()*1000)
                print '\n14: iter:
' + str(iter) + ", " + str(time_
val) + ', \
Temp: '  + temp_str + ', Press: '
+ press_str + ', \
Alti: ' + alti_str + '\n'

                #temp_str[5] is
bogus non-printable char
                [c + '\0' for c in
temp_str]
                temp_val =
float(temp_str[0:4])
                print '\n15: tem-
perature val: ', temp_val

                # prepare rest api
data params per SAMI service API
requirements
                sami_params = {

"sdid": sdid,

"ts": time_val,

"type": "message",

"data": {

 "temperature": temp_val

},
                                }
                # cleanup data
(long time val) to create good
json data
                jsami_params =
json.dumps(sami_params)
                print '\n16: jsa-
mi: ' + str(jsami_params)

                try:
                        print '\
n17: preparing to send data to
SAMI service'
                        req =
requests.post(url=sami_url,
data=jsami_params, \
headers=headers)
                        print '\
n18: sent data successfully !!!'
                except:
   # catch any exception
                        e = sys.
exc_info()[0]
                        print "\
n18: failed to send.  error: " +
str(e)

   # repeat search every 2 mins
                print "\n19:
waiting 2 mins to send next
data...\n"
                time.sleep(120)
       # repeat whole process
       print "\n20: next
teration......\n"

    # while block ends
except:
    # some error
    e = sys.exc_info()[0]
    print "\nErr: " + str(e) +
"\n"
    print "\nExiting...\n"

print "Done!"
```

Escriba lo siguiente para ejecutar el script:

```
$ cd ~/zBU/sami/
$ python ./sami-req-client.py
```

Debería iniciarse y empezar a recoger

datos meteorológicos desde el puerto ttyUSB0 para enviarlos a la página web SAMIIO, una vez por ciclo y presentar los valores de Temperatura una vez cada 2 minutos. Se puede variar la temperatura situando el sensor más cerca o lejos de una fuente de luz. La Figura 18 muestra el gráfico de temperatura tal y como se ve desde el sitio web de SAMIIO en http://bit.ly/1IVhfoZ. Los registros log de los datos enviados se muestran en la Figura 19.

El script anterior puede modificarse para mostrar otros datos como la presión atmosférica y la altitud. La plataforma SAMIIO también proporciona SDKs para diversos lenguajes de programación, que se puede utilizar para desarrollar aplicaciones.

Recursos adicionales

http://bit.ly/1O9FKFZ
http://bit.ly/1ds2ONj
http://bit.ly/1LUqbPP
http://bit.ly/1KBbpjQ
http://bit.ly/1NeVs1I
http://bit.ly/1IVhfoZ
http://bit.ly/1g6FbPj
http://bit.ly/1IVhcJI

Figura 18: diagrama de temperatura en el sitio web SAMIIO

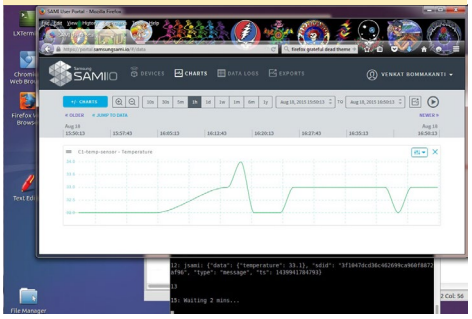
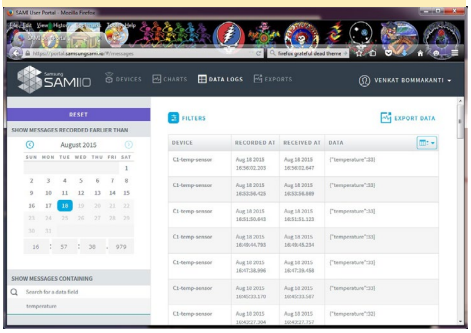


Figura 19: Registro log de la temperatura en el sitio web SAMIIO

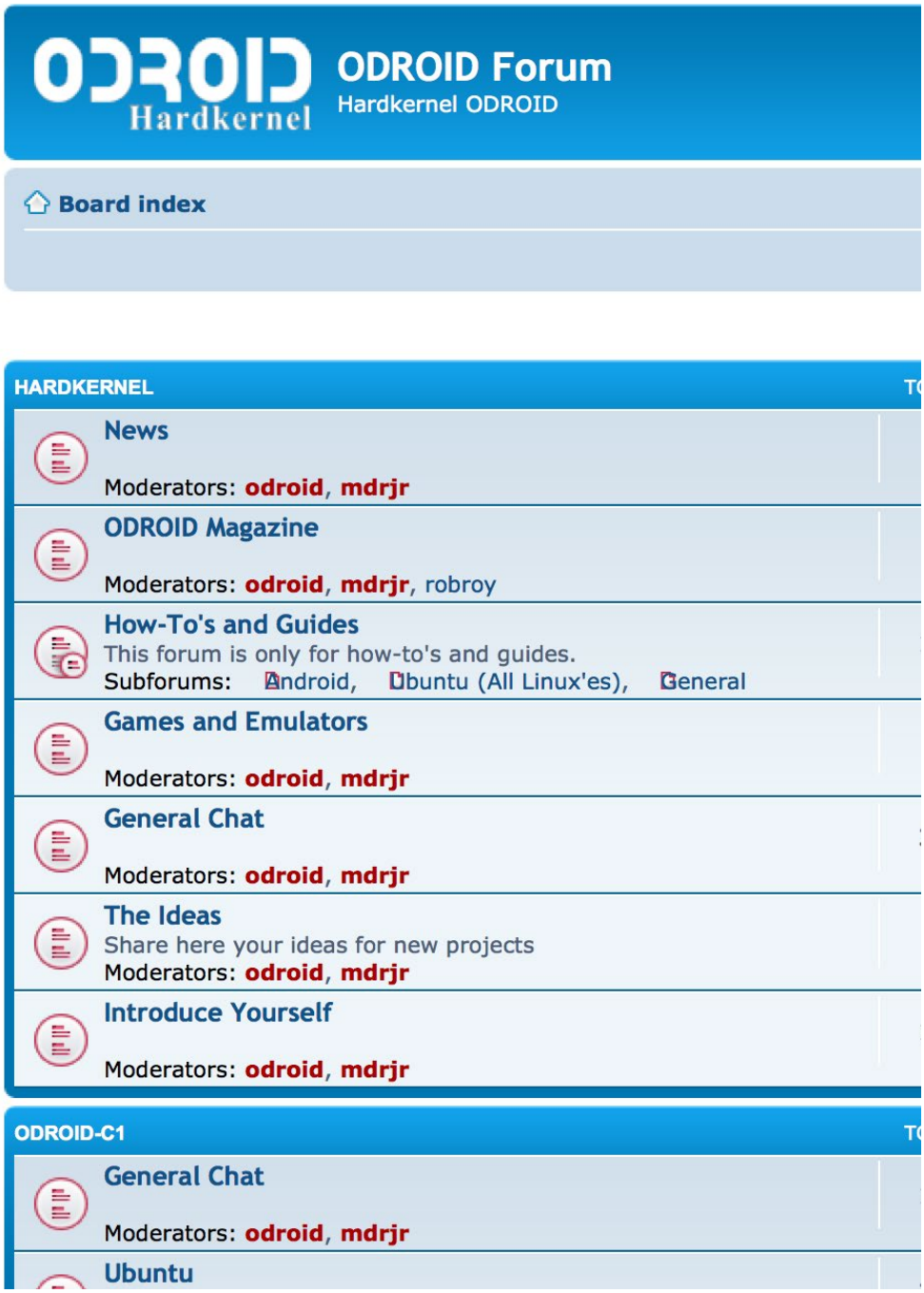


FOROS ODROID

EL LUGAR PERFECTO PARA COMUNICARTE CON LOS DESARROLLADORES DE HARDKERNEL

por Rob Roy

Los foros ODROID han sido el lugar de encuentro de la creciente comunidad Hardkernel durante varios años, con más de 12.500 miembros a septiembre de 2015. Se puede discutir de ODROIDs con Mauro, el desarrollador del kernel de Linux y Justin, la CEO de Hardkernel, junto con un equipo de desarrolladores cada vez mayor que invierten parte de su tiempo en ayudarte a obtener el máximo provecho de su ODROID. Compruébelo tu mismo en http://forum.odroid.com!



CONOCIENDO UN ODROIDIAN

ULI MIDDELBERG (@UMIDDELBERG)
EXPERTO EN DOCKER Y UN MAGO LINUXT

editado por Rob Roy

Por favor, háblanos un poco sobre ti.

Estoy viviendo en Hamburgo, Alemania, junto a mi esposa y mis dos hijos. Tengo 45 años y trabajando para un periódico alemán.

Uli comparte regularmente su experiencia Docker en los foros

¿Cómo fueron tus inicios con los ordenadores?

Mi primer ordenador fue un Atari ST 1040, con el que aprendí programación (Pascal, Modula 2, C, M86k ensamblador). Tuve la suerte de trabajar con un Acron Archimedes A3000, que estaba a años luz por delante de todo lo que tenía acceso en ese momento. En la universidad, disfruté muchísimo estrenando varios NeXT Stations y Cubes NeXT que el departamento de ciencias de la computación había comprado recientemente.

¿Qué te atrajo de la plataforma ODROID?

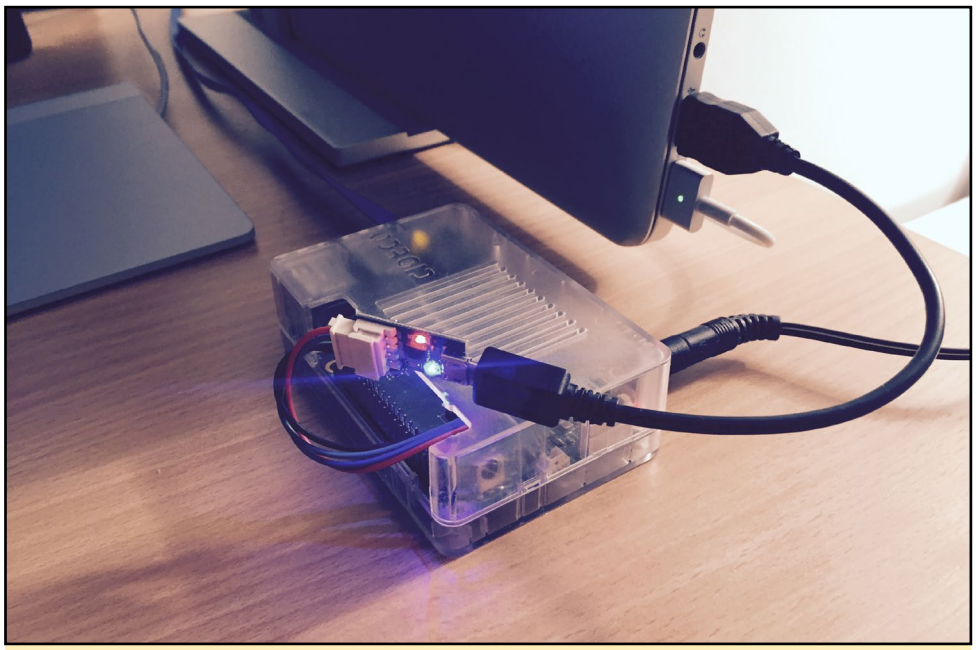
El ODROID-C1 no era el primer dispositivo ARM que había probado. Cuando se compara con la Raspberry Pi, observas que el rendimiento es mucho major. Comparado con dispositivos basados imx6, observas una comunidad de usuarios muy viva que ofrece un gran apoyo. Los desarrolladores Hardkernel mantienen una actitud muy abierta con las aportaciones procedentes de la comunidad, lo que hace que sea fácil integrar proyectos personalizados como Docker en el kernel principal. Tengo curiosidad por ver qué dispositivo ODROID darán a conocer en el futuro.

¿Cómo utilizas tus ODROIDs?

Actualmente sólo tengo un C1. Lo estoy usando para desarrollo, y para otras cosas como domotica con openHAB. Quiero usarlo en el futuro como dispositivo de sincronización junto con OwnCloud, pero primero tengo que habilitar el soporte para cryptodev y descargar TLS/AES. Estoy a punto de comprar un C1+ y la placa de audio para reemplazar mi reproductor de CD, así como un XU4 para fines de desarrollo. Herramientas como Docker ayudan a mover con facilidad cargas de trabajo entre diferentes dispositivos, así que tal vez traslade el servicio OwnCloud al XU4, una vez que haya conseguido instalar y poner en marcha Docker.

Eres un buen conocedor de los sistemas Linux, especialmente de los kernels. ¿Cómo llegaste a ser tan hábil?

Empecé a usar Linux en 1994 y dos años más tarde comencé a trabajar como administrador de sistemas en el grupo de investigación de la universidad, empecé administrando sistemas simplemente leyendo un libro sobre admi-nistración de sistemas, http://oreil.ly/1Y1yWhH. Por aquel entonces, compilar tu propio kernel Linux era una tarea muy habitual. Continué trabajando como administrador durante un par de años más, pero hoy día sólo ejecuto Linux para fines privados. En las plataformas para los PC, prácticamente no hay necesidad de compilar un kernel



Esta simple configuración desvela el potente software que ejecuta el ODROID-CI de Uli

Linux a medida, ya que los proveedores de las principales distribuciones hacen ese trabajo para ti. Cuando empecé a trabajar con dispositivos Linux basados en ARM en 2013, recuerdo el momento en el que recibí mis primeras lecciones de Linux. Hoy en día hay una gran cantidad de recursos útiles disponibles online, y es muy probable que alguien ya haya planteado tu pregunta o duda antes.

¿Cuál es tu ODROID favorito?

No he trabajado con el U3, la mayoría de la gente comenta que es su placa ODROID favorita. El XU4 tiene una relación calidad/precio muy interesante, pero me encantaría ver una placa ODROID ARMv8 de 64 bits.

¿Está involucrado en otros proyectos informáticos ajenos a ODROID?

Descubrí que Docker es un medio fascinante para desarrollar, implementar y gestionar aplicaciones, pero observe una cierta falta de soporte para los dispositivos ARM. Así que empecé a exportar Docker a ARM junto con otros, y escribí todas los pasos necesarios para hacerlo en mi wiki GitHub en http://bit.ly/1M5Iphp. Esto le da a otras personas interesadas la oportunidad de repetir estos pasos con sus propias placas, que lo prefiero en lugar de publicar y mantener imágenes personalizadas.

Me gusta mucho la idea de poder ejecutar Linux en esta pequeña caja ARM por su relación consumo/rendimiento. Especialmente en pequeños entornos, un servidor Linux ARM ofrecería rendimiento más que suficiente. Voy a continuar escribiendo cosas interesantes sobre Linux en ARM, por si acaso necesito recordar cómo hice algo en particular.

¿Qué aficiones e intereses tienes aparte de los ordenadores?

¡Los Deportes! Me encanta hacer windsurf, al igual que toda mi familia. Me gusta montar en bicicleta en lugar de utilizar el coche, y me encanta ir a correr.

¿Qué tipo de novedades de hardware te gustaría ver en futuras placas Hardkernel?

Seguramente no soy el primero que pide esto, pero una placa ARMv8/AArch64 sin ventilador con más de 2 GB de memoria RAM, un puerto SATA o puertos USB3 independientes y soporte hardware para kernel Linux convencional sería realmente genial.

¿Qué consejo le darías a alguien que quiere aprender más sobre el mundo de la programación y/o hardware?

Empezar a leer buenos libros, poner en marcha tu propio sistema Linux, aprender Python o Java, crear tu propio proyecto modificando y/o ampliando los ya existentes, y familiarizarse con GitHub. También existen muchos y excelentes MOOCs (http://bit.ly/1EX2vLx) que cubren una amplia variedad de temas.

Uli disfruta de windsurf y de otros deportes de agua con sus hijos