Juegos Linux/Android • Gentoo para ODROID-C2 • Compilar Android

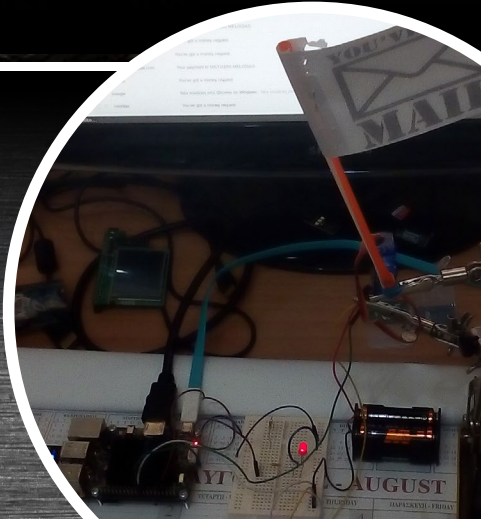
ODROID Magazine

Año Tres
Num. #34
Oct 2016

Juega y Disfruta usando ODROID-C2 y *GameStream*

Llévate tus modernos juegos favoritos de tu despacho a tu salón

- Monitoriza en Remoto Registros Modbus usando un ODROID-XU4
- Dispositivo IoT y Aplicación ODROID: Sistema de Aviso Mecánico Gmail



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 para hacer todo lo que imagines

HARDKERNEL

Pollin ELECTRONIC

Realizamos envios de ODROID-C2 and ODROID-XU4 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

Aunque con los ODROIDs ya es posible jugar a miles de juegos, incluyendo los que están disponibles en Play Store, los juegos nativos del Linux que han sido exportados por desarrolladores y juegos emulados usando RetroArch, muchos de los juegos modernos todavía no son capaces de funcionar de forma nativa en la plataforma ARM. Sin embargo, NVIDIA ha lanzado un software que permite ejecutar juegos de última generación via streaming por red, lo cual permite que los ODROIDs puedan trabajar como estaciones de juegos remotas. Esto nos da la posibilidad de ejecutar un servidor de juego en casa y jugar a los mejores juegos en cualquier lugar de la casa sobre un gran monitor.

También contamos con un par de proyectos de bricolaje este mes, incluyendo un proyecto de IoT de Miltiadis que levanta una bandera cuando llega un correo electrónico. Adrian nos muestra cómo configurar una cámara IP, Joel nos describe su configuración sobre automatización industrial, y Bo documenta su búsqueda sobre la mejor configuración de refrigeración para el ODROID-XU4. Si quieres tener un control completo sobre su sistema operativo, disponemos de instrucciones para compilar tanto Gentoo y Android Lollipop. También contamos con el análisis de algunos de nuestros juegos favoritos de Android y Linux.

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 información sobre cómo enviar articulos, contacta con odroidmagazine@gmail.com, o visita http://bit.ly/1ypImXs.
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

NUESTRO MARAVILLOSO PRESONAL ODROIDIAN:

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`.

Bruno Doiche, Editor Artístico Senior

¿Es GameStream el Santo Grial al que Bruno siempre deseaba jugar con sus juegos favoritos de PC en el salón de su casa cuando su esposa se largaba a su laboratorio de diseño? Tal vez. Simplemente no le dirá que está pensando en compra un nuevo ordenador sólo para jugar a los juegos. Después de todo, ya está luchando para hacer que todos sus dispositivos electrónicos puedan adaptarse a su apartamento, así que evita la llamada incesante de reunirse con la ¡RAZA SUPERIOR DE JUGADORES DE PC!

Manuel Adamuz, Editor Español

Tengo 31 años y vivo en 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.

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 http://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. Continuo muy enamorado de muchos de los 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.

Andrew Ruggeri, Editor Adjunto

Soy un ingeniero de sistemas Biomédicos anclado en Nueva Inglaterra que actualmente trabaja en la industria aeroespacial. Un microcontrolador 68HC11 de 8 bits y el código ensamblador son todo lo que me interesa de los sistemas embebidos. Hoy en día, la mayoría de los proyectos en los que trabajo están en lenguajes C y C ++, o en lenguajes de alto nivel como C# y Java. Para muchos proyectos, utilizo placas ODROID, pero aún sigo intentando utilizar los controladores de 8 bits cada vez que puedo (soy un fan de ATMEL). Aparte de la electrónica, soy un amante de la fotografía analógica y desarrollo la película friki con la que disfruto intentando hablar en idiomas extranjeros.

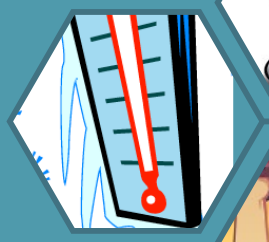
Venkat Bommakanti, Editor Adjunto

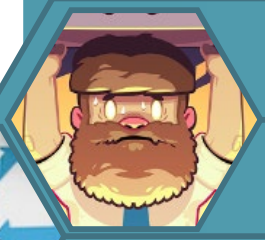
Soy un apasionado de los ordenadores desde la bahía de San Francisco en California. Procuro incorporar muchos de mis intereses en proyectos con ordenadores de placa reducida, tales como pequeños modificaciones de hardware, carpintería, reutilización de materiales, desarrollo de software y creación de grabaciones musicales de aficionados. Me encanta aprender continuamente cosas nuevas, y trato de compartir mi alegría y entusiasmo con la comunidad.

Josh Sherman, Editor Adjunto

Soy de la zona de Nueva York, y ofrezco mi tiempo como escritor y editor para ODROID Magazine. Suelo experimentar con los ordenadores de todas las formas y tamaños: haciendo trizas las tablets, convirtiendo Raspberry Pi en PlayStations y experimentado con los ODROIDs y otros SoCs. Me encanta trabajar con los elementos básicos y así poder aprender más, y disfrutar enseñando a otros escribiendo historias y guías sobre Linux, ARM y otros proyectos experimentales divertidos.

INDICE





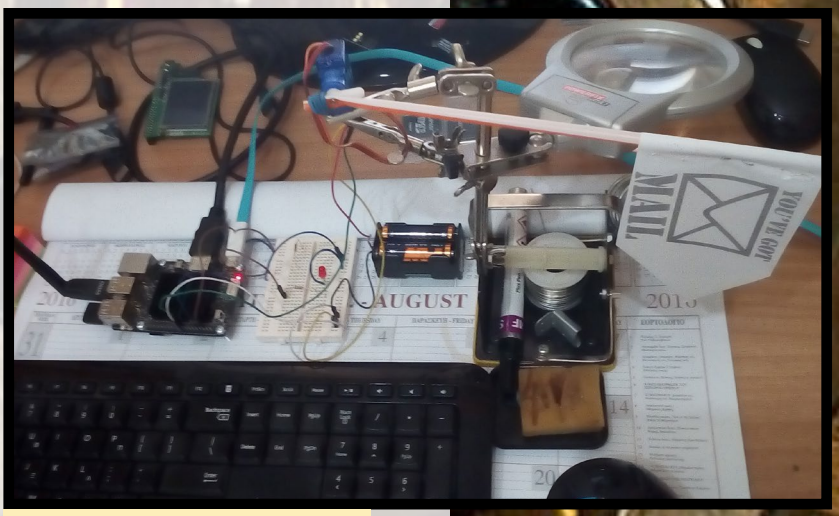
SISTEMA DE AVISO MECANICO PARA GMAIL

DISPOSITIVO IOT Y APLICACION ODROID

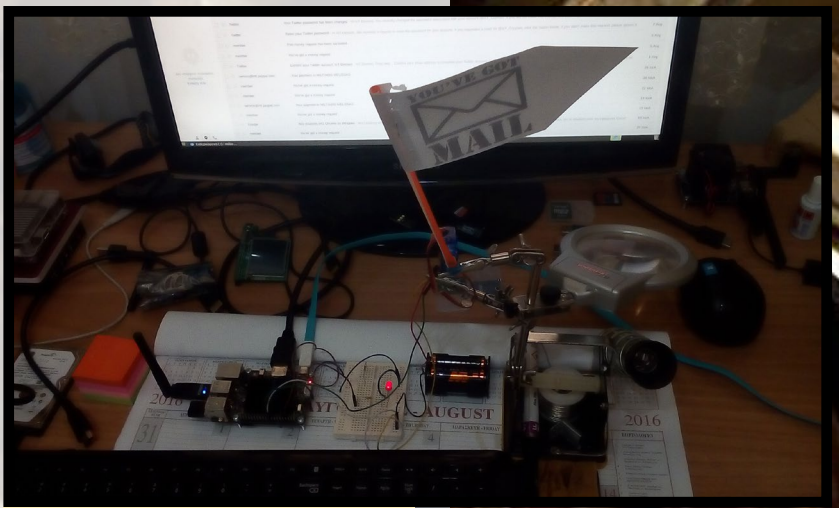
por Miltiadis Melissas

Continuando con mi último artículo de la edición de septiembre de ODROID Magazine titulado "ODROID-C2 como dispositivo IoT: Comunicándose con el mundo real", he estado buscando una aplicación del Internet de las cosas (IoT) que hiciera uso de un servomotor. Este tutorial detalla mi proyecto servomotor, un divertido proceso de construcción de un dispositivo IoT que constantemente chequea tu cuenta de Gmail para ver si han entrado nuevos mensajes.

El dispositivo IoT utiliza un ODROID-C2, se conecta automáticamente a tu cuenta de Gmail y comprueba si hay nuevos mensajes en la bandeja de entrada. Si existen nuevos mensajes, se enciende un LED y un sistema electrónico-mecánico levanta una bandera en la que se puede leer "YOU'VE GOT MAIL". Cuando todos los nuevos mensajes hayan sido leídos, la bandera vuelve a su posición inicial y el LED se apaga. Entra en mi canal de Youtube en `http://bit.ly/2bT9bMz` para ver el dispositivo en funcionamiento.



Bandera bajada - LED apagado



Bandera levantada – LED encendido

Requisitos del sistema

Necesitarás un ODROID-C2 con la última versión 2.0 de Ubuntu de Hardkernel (`http://bit.ly/2cBibbk`) y la versión 2.7.12 de Python instalada. Los materiales adicionales incluyen:

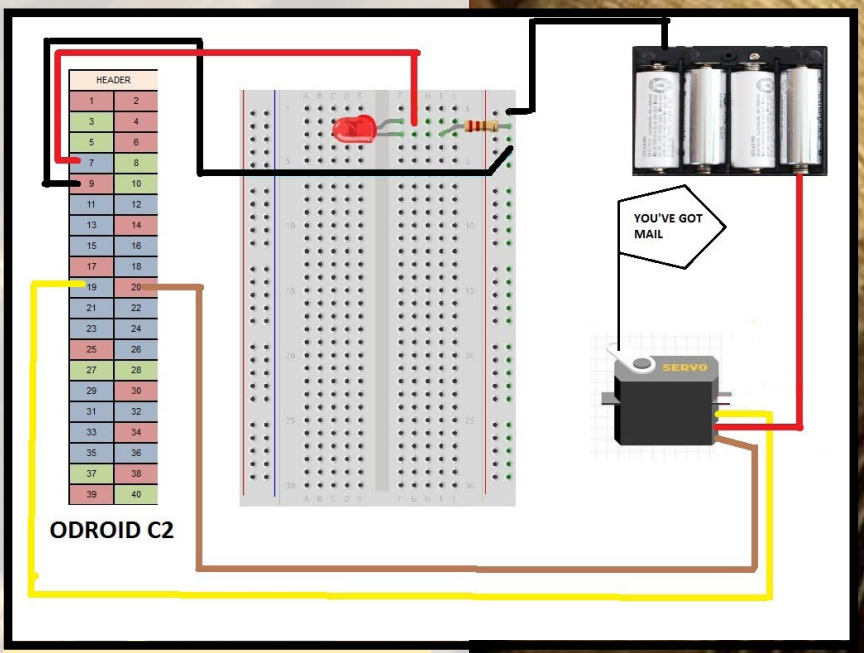
- 1 x C2 (Ordenador de placa reducida ARM quad core de 64 bits a 1.5Ghz)
- 1 x cuenta de Gmail (que es gratuita)
- 1 x Placa de pruebas
- 1 x LED (diodo emitente de luz)
- 1 x Resistencia de 220 ohmios
- 1 x Servo motor (enlace)
- 4 x Pilas de 1,5 y por supuesto (6V)
- 1 x conjunto de conectores

Conexiones de hardware

Por favor, consulta el esquema de la Figura 3 y el excelente diagrama de distribución de pines de Hardkernel para el ODROID-C2 en http://bit.ly/2aXAlmt. El ánodo(+) del LED está conectado al pin7 a través de una resistencia (220 Ω), mientras que su cátodo (-) se

conecta a la toma de tierra (pin9). Los dos pines se sitúan uno al lado del otro de acuerdo con la disposición de pines de Hardkernel.

Ahora explicaremos cómo utilizar un servo motor para controlar la bandera. El servo está conectado a los pines 19 y 20. Por otro lado, éste necesita más corriente de la que el ODROID-C2 puede suministrarle, de modo que necesitaremos un juego de cuatro (4) pilas como fuente de alimentación adicional. El servo tiene tres (3) cables: amarillo, rojo y marrón. El amarillo lleva la señal, que es la modulación por ancho de pulsos (PWM) que se conecta al pin19. Por favor consulta de nuevo la documentación Hardkernel en http://bit.ly/2ckfdKn para ver qué pines pueden proporcionar ese tipo de pulsos. Para la puesta a tierra (-), el servo está conectado al pin20, usando el cable marrón. Sin embargo, es mejor utilizar una puesta a tierra común y dejarla desconectada ya que los pines son activos muy valiosos, especialmente para los proyectos más complejos. Por último, el cable rojo está conectado a la batería externa, en otras palabras, a las cuatro pilas(+ Vcc). Por lo tanto, rojo con rojo, dejando el cable blanco de la batería para lo último, para conectarlo a la puesta a tierra común con su negativo (-). El hardware del sistema de aviso mecánico Gmail esta listo. ¡Vamos a darle vida con el script Python que escribiremos con esa intención!



Esquema del sistema de aviso Gmail

Software preliminar

Antes de empezar a escribir el script Python, analizaremos los sistemas operativos (SO) del ODROIDC2. Todos las placas fabricadas por Hardkernel pueden ejecutar Linux o Android, y ODROID-C2 no es diferente. Para el sistema de aviso mecánico Gmail, vamos a utilizar Linux como sistema operativo principal. La razón es que Linux es más versátil y robusto a la hora de tratar aplicaciones del Internet de las cosas (IoT). Puede grabar Linux Ubuntu 16.04 Mate siguiendo la guía de `http://bit.ly/1Vk9u4o`.

Por último, instala la libreria WiringPi2. Esta librería controla los pines en ODROID-C2. Hardkernel proporciona una excelente guía en su sitio para instalar esta librería en `http://bit.ly/2ba6h8o`.

Necesitarás utilidades adicionales si recompilas manualmente los enlaces con swig-python WiringPi. Se pueden instalar con el siguiente comando:

```
$ sudo apt-get install python-dev python-setuptools
```

Descargar y configura WiringPi2 para Python desde el repositorio:

```
$ git clone https://github.com/hardkernel/WiringPi2-Python.git
$ cd WiringPi2-Python
$ git submodule init
$ git submodule update
```

Compila e instala la libreria:

```
$ sudo python setup.py install
```

Descarga y ejecuta el código fuente de ejemplo desde `http://bit.ly/2cKcwkd`, es opcional y está pensado unicamente para hacer pruebas.

```
$ wgeg http://dn.odroid.com/source_peripherals/ctinkeringkit/example-led.py
$ sudo python example-led.py
```

También puede usar un IDE Python llamado IDLE ejecutando el comando:

```
$ sudo apt-get install idle
```

Software de aplicación

Ejecuta la utilidad IDLE y copia y pega las líneas de código que se indican a continuación. Cuando hayas terminado de editar, guarda el script Python con el nombre trace_messages.py en el directorio /home/odroid/Documents/gmail_Python. Si no tienes instalado IDLE, puedes copiar y pegar el script en cualquier editor de tu sistema y guardarlo con el mismo nombre.

Comentaré cada línea del código para explicar lo que ocurre dentro de este scritp:

```
#! python
import gmail, wiringpi2 as wpi, time # import the modules

#use ODROID-C2 pin numbers for LED and SERVO
LED_PIN=7
SERVO_PIN=12

wpi.wiringPiSetup()

#setup pin (LED) as an output
wpi.pinMode(LED_PIN,1)

#setup pin (SERVO) as an output too
wpi.pinMode(SERVO_PIN,1)


#setup Pulse Width Modulation(PWM) for Servo
wpi.softPwmCreate(SERVO_PIN,0,50)

#setup
g = gmail.login('youremail@gmail.com', 'yourpassword')
unread_messages = g.inbox().mail(unread=True)
total_messages = 0

for message in unread_messages:
      total_messages += 1

if total_messages > 0:
      # there are unread emails, turn light on
      wpi.digitalWrite(LED_PIN,1)
      for i in range (15):
         wpi.softPwmWrite(SERVO_PIN,i)
         time.sleep(0.2)
else:
      # there are no unread emails, turn light off
      wpi.digitalWrite(LED_PIN,0)
      for i in range (15,0,-1):
         wpi.softPwmWrite(SERVO_PIN,i)
         time.sleep(0.2)
```

Este script ha sido modificado, adaptado y actualizado para cubrir las necesidades de este proyecto, usando http://bit.ly/2cGSwBS como referencia. La idea básica sigue siendo la misma, no obstante, descompondremos el código para ver lo que sucede:

```
<import gmail, wiringpi2 as wpi, time> # import the modules
```

En primer lugar, importamos los módulos. Son tres (3): Gmail, wiringpi y time. Los módulos en Python son pequeñas piezas de código escrito para una finalidad específica, similares a las librerías en Arduino IDE. "Wiringpi2" es el módulo que controla los pines en ODROID-C2, y es por eso que instalamos este módulo anteriormente. "Time" es un módulo integrado en Python (módulo del sistema) que proporciona funciones de temporización. El módulo "Gmail" está escrito por Charlie Guo (http://bit.ly/2bY7Vhh), y es muy importante para que nuestro proyecto funcione. Para importarlo, hay que instalarlo en primer lugar, que no es nada difícil. Descarga la librería de Github (http://bit.ly/2cC01Jb) en un directorio y extraer el contenido. En el interior, debería haber una carpeta llamada "Gmail". Copia esta carpeta completa en el directorio /home/odroid/Documents/gmail_python. El módulo es, básicamente, un script que inicia sesión en tu cuenta de Gmail con tus credenciales y lee los mensajes entrantes.

La siguiente línea configura la conexión para leer en el GPIO # de WiringPi, es la primera columna del mapa de pines GPIO de Hardkernel de la siguiente figura.

```
<wpi.wiringPiSetup()>
```

Estas dos líneas de código son muy simples: definimos los pines que usaremos.

```
<LED_PIN=7>
<SERVO_PIN=12>
```

Ten en cuenta que estamos haciendo referencia al pin19 que proporciona la modulación por ancho de pulsos para el servo según el pin12. Una vez más, esto viene dado por la disposición de los pines de Hardkernel: el pin19 hace referencia al GPIO #12 de WiringPi según la tabla de Hardkernel (`http://bit.ly/2aXAlmt`). El pin7 sigue siendo el pin7 en sí mismo.

```
<wpi.pinMode(LED_PIN,1)>
```

Hemos configurado el pin7 para manejar el LED como salida:

```
<wpi.pinMode(SERVO_PIN,1)>
```

También hacemos lo mismo con el servo declarándolo como salida:

```
<wpi.softPwmCreate(SERVO_PIN,0,50)>
```

GPIO PIN MAP

ODROID-C2 40pin Layout

Power Pin
Special Function
GPIO/Special Function

WiringPi GPIO#	Export GPIO#	ODROID-C2 PIN	Label	HEADER		Label	ODROID-C2 PIN	Export GPIO#	WiringPi GPIO#
			3V3	1	2	5V0			
	205	I2CA_SDA	SDA1	3	4	5V0			
	206	I2CA_SCL	SCL1	5	6	GND			
7	249	GPIOX.BIT21	#249	7	8	TXD1	TXD_B	113	
			GND	9	10	RXD1	RXD_B	114	
0	247	GPIOX.BIT19	#247	11	12	#238	GPIOY.BIT10	238	1
2	239	GPIOX.BIT11	#239	13	14	GND			
3	237	GPIOX.BIT9	#237	15	16	#236	GPIOX.BIT8	236	4
			3V3	17	18	#233	GPIOX.BIT5	233	5
12	235	GPIOX.BIT7	#235	19	20	GND			
13	232	GPIOX.BIT4	#232	21	22	#231	GPIOX.BIT3	231	6
14	230	GPIOX.BIT2	#230	23	24	#229	GPIOX.BIT1	229	10
			GND	25	26	#225	GPIOY.BIT14	225	11
	207	I2CB_SDA	SDA2	27	28	SCL2	I2CB_SCL	77	
21	228	GPIOX.BIT0	#228	29	30	GND			
22	219	GPIOY.BIT8	#219	31	32	#224	GPIOY.BIT13	224	26
23	234	GPIOX.BIT6	#234	33	34	GND			
24	214	GPIOY.BIT3	#214	35	36	#218	GPIOY.BIT7	218	27
		ADC.AIN1	AIN1	37	38	1V8	1V8		
			GND	39	40	AIN0	ADC.AIN0		

Mapa de pines GPIO del ODROID-C2

Esta es una función muy importante en el objeto wpi dentro del script, que configura la modulación por ancho de pulsos para el servo. Los argumentos son el SERVO_PIN (es decir pin12), el valor inicial ("0") y el rango de modulación por ancho de pulsos ('50'). Para simplificar las cosas, mantenemos el pulso alto de 5ms hechos de 50 pasos. Por supuesto, puedes experimentar con otros valores cuando estés calibrando el servo. El código representa mis valores tras realizar algunas calibraciones con el servo para alcanzar la posición correcta de la bandera.

```
<g = gmail.login('youremail@gmail.com', 'yourpassword')>
```

Creamos el objeto "g" y activamos al método login basado en el módulo de Gmail importado anteriormente. Sustituye el correo electrónico y contraseña por tu cuenta de Gmail y contraseña reales, dejando las comillas en su lugar.

```
<unread_messages = g.inbox().mail(unread=True)>
```

Recuperamos todos los mensajes no leídos y los almacenamos bajo la variable "unread_ messages". Observa como "unread=True" se incluye como un parámetro. Puede cambiar esto para recuperar mensajes basándote en diferentes parámetros, como el remitente o el asunto.

```
<total_messages = 0>


<for message in unread_messages:
      total_messages += 1>
```

Iteramos por los mensajes no leídos e incrementamos la variable " total_messages " en uno si es necesario:

```
<if total_messages > 0:
      # there are unread emails, turn light on
      wpi.digitalWrite(LED_PIN,1)
      for i in range (15):
         wpi.softPwmWrite(SERVO_PIN,i)
         time.sleep(0.2)
else:
      # there are no unread emails, turn light off
      wpi.digitalWrite(LED_PIN,0)
      for i in range (15,0,-1):
         wpi.softPwmWrite(SERVO_PIN,i)
         time.sleep(0.2)>
```

Este es un condicional muy simple. Si el número de la variable "total_ messages" es mayor que cero, entonces hacemos dos cosas. En primer lugar, encendemos el LED con la siguiente expresión:

```
<wpi.digitalWrite(LED_PIN,1)>
```

Y ahora iniciamos el servo, que eleva la bandera cambiando el ciclo de trabajo:

```
<wpi.softPwmWrite(SERVO_PIN,i)>
```

Por el contrario, si no hay mensajes nuevos, apagaremos el LED y bajaremos la bandera. La bandera se coloca en posición horizontal cambiando el ciclo de trabajo del pulso ya que ahora contamos en sentido contrario a las ajugas del reloj dentro del mismo rango deduciendo -1 en cada ciclo. Ten en cuenta el siguiente bucle:

```
<for i in range {15,0,-1):>
```

Ejecutar el script

Ahora es el momento de ejecutar el script. Abre un terminal (desde la interfaz gráfica de usuario ve a Applications-> System Tools->Mate Terminal) y escribe:

```
$ sudo python /home/odroid/Documents/\
gmail_python/trace_messages.py
```

A continuación, observa lo que sucede. Si entra cualquier mensaje, la bandera deberá levantarse y el LED debería encenderse. Si es así, ¡hemos tenido éxito y nuestro script funciona! Si no es así, buscar posibles errores en tu código. Después,

tenemos que dar un paso más para hacer que se ejecute de forma automática a unos intervalos de tiempo determinados, como por ejemplo cada 5 minutos. Para esta tarea, utilizaremos la utilidad cron. ¿Qué es cron? Permite definir trabajos que son usados para programar tareas y scripts como definir etiquetas, hacer copias de seguridad y activar alarmas. Para obtener más información sobre cron, por favor visita http://bit.ly/2bTmNaN. Para activar el cron, hay que ejecutar el comando crontab que nos proporciona una lista de tareas programadas:

```
$ crontab -e <Enter>
```

Probablemente estará vacía. Elije cualquier editor de texto y añade la siguiente línea de código al final de la lista de tareas programadas:

```
*/5 * * * * sudo python /home/odroid/Documents/gmail_python/
\trace_incomings.py
```

Los cinco "asteriscos" ("* * * * *") especifica la frecuencia con la que deseas que se ejecute la tarea. El primera asterisco controla los minutos, es por eso que pongo '/5', ya que quiero que esta tarea programada se ejecute cada cinco minutos. El segundo asterisco controla las horas, el tercer especifica el día del mes, el cuarto indica el mes y el quinto representa el día de la semana. Esos cuatro se han dejado intencionalmente en blanco sin ningún número, sólo con asteriscos. Puedes experimentar con otras opciones igualmente. Al final de la tarea programada, está el comando en sí que queremos que se ejecute de forma automática:

```
$ sudo python /home/odroid/Documents/gmail_python/trace_incomings.py
```

Este comando ejecuta nuestro script y apunta a la ruta en la que se encuentra, que en este caso es /home/odroid/Documents/gmail_python.

Después, guarda y cierre el editor. Ahora, espera y observa como la aplicación hace su trabajo. Envía algún mensaje a tu cuenta de Gmail para comprobar si tienes alguno sin leer y ver como la bandera se levanta. La bandera con "You've Got Mail!" debería elevarse al mismo tiempo que ves como se ilumina tu LED. ¡Felicitaciones, Tu sistema de aviso mecánico para Gmail funciona!

Notas finales

Debes tener en cuenta que cualquier código Python en IDLE debe ser ejecutado como usuario root, de lo contrario no funcionará. Una forma muy simple de hacerlo consiste simplemente en crear un acceso directo de IDLE en el escritorio tras la instalación y luego editarlo con:

```
$ cd ~/Desktop
$ sudo nano idle.desktop
```

y luego cambia la línea " Exec=/usr/bin/idle " por " Exec=/usr/bin/gksu –u root idle ", después, guarda el archivo.

Espero que hayas disfrutado de este proyecto tanto como lo hice yo. El sistema de aviso mecánico de Gmail es la segunda parte de una serie de tres proyectos que he escrito para ODROID Magazine. Mi siguiente proyecto IoT utiliza un ODROID-C2 para observar y controlar la fermentación de botellas de vino en una bodega. En particular, la ODROID-C2 observa y controla la configuración del aire acondicionado midiendo la temperatura y la humedad del entorno de la fermentación. Notificará al usuario cualquier desviación de los valores aceptables a través de varios servomotores. Cualquier anomalía actualizará la cuenta de Twitter del usuario, dando la oportunidad de analizar el producto más aun. Como siempre digo, "¡Con los ODROIDs todo es posible!"

CONVIERTE TU ODROID EN UNA CAMARA IP

por Adrian Popa

Hacer que una cámara IP funcione con tu ODROID no tiene ciencia

En el último año han aparecido artículos que detallan cómo configurar tu ODROID con una cámara web para realizar todo tipo de tareas interesantes, desde la detección de incendios (http://bit.ly/2cviz9K) a la realidad aumentada (http://bit.ly/2cV74eA), incluso aplicaciones de seguridad para el hogar (http://bit.ly/2dsqnen). En mi caso, todo lo que quería era una cámara web que funcionaba a través de Internet. La típica cámara ip comercial te permite utilizar la cámara de forma remota en tiempo real con sonido a través del Protocolo de flujo en tiempo real (RTSP). Por lo general suele incorporar otras funciones como capturar imágenes fijas, hacer una panorámica o inclinar la imagen. Estas cámaras se suele utiliza a menudo para poner en marcha robustas aplicaciones de monitorización del hogar, como por ejemplo un DVR remoto para almacenar tus grabaciones o para facilitar el acceso remoto cuando se necesite. Android tiene un montón de aplicaciones que permiten gestionar todas estas cuestiones, pero nosotros nos vamos a centrar en Linux, ya que es posible que también quieras utilizar tu ODROID para otras tareas basadas en Linux. Al final de este artículo, sabrás como tomar imágenes desde tu cámara web a través de Internet, ver secuencia de video en tiempo real con sonido y grabar el video.

Configurar la cámara

La mayoría de las cámaras modernas son compatibles con Linux gracias al driver genérico “uvc”. El driver presenta varios dispositivos nuevos en tu máquina Linux cuando se conecta una cámara web. Por ejemplo, es posible que vea una interfaz Video4Linux /dev/video0, un nuevo dispositivo de entrada ALSA y tal vez un botón que actúa como teclado HID. Instalando el paquete v4l-utils, podrás listar los modos compatibles con tu cámara. Tienes un ejemplo del listado de la webcam 720p de hardkernel en http://pastebin.com/L1VwZZFs.

```
$ sudo apt-get install v4l-utils
$ v4l2-ctl --list-formats-ext
```

Si te fijas la mayoría de las cámaras puede transmitir en YUV (modo no comprimido) con pocas imágenes por segundo, o en MJPEG (modo comprimido). Las cámaras de alta gama también pueden capturar vídeo H264 que se codifica directamente dentro de la cámara. Este tutorial asume que dispones de una cámara compatible con MJPEG, aunque seguro que te gustaría ver también secuencias H264 en tu sistema.

La utilidad v4l2-ctl te permite listar y cambiar algunos parámetros de la cámara, tales como el brillo, el contraste o el gamma, es útil si no dispones de unas condiciones óptimas de iluminación. Puedes listar estos parámetros con el comando:

```
$ v4l2-ctl --list-ctrls
```

Si tu cámara no presenta un pseudo-file /dev/video0, pero puedes tomar imágenes con una API personalizada, puedes utilizar v4l2loopback (http://bit.ly/2cxa6rc) para enviar tus datos a un dispositivo virtual /dev/videoX de manera que puedas leerlos con herramientas comunes.

Tomar imagenes fijas

Ahora que la cámara funciona, lo primero es tomar imágenes con ella, ya sea guardándolas en el disco local o vién-

dolas de forma remota. Aunque la tarea parece muy sencilla y hay varias herramientas que te pueden ayudar con ello, es importante cuidar los detalles. Herramientas como uvccapture o streamer pueden hacer el trabajo, pero he descubierto que en la práctica ambas tienen algunos problemas:

Al capturar una imagen se activa la cámara y neesita un tiempo hasta completar la captura, a veces hasta 30 segundos.

Las imágenes desde estas herramientas generalmente son muy oscuras porque la cámara no ha tenido suficiente tiempo para estabilizar el nivel de iluminación. Streamer puede compensar esto "grabando" durante un tiempo determinado, por ejemplo 1 segundo antes de tomar la foto.

A veces la cámara puede mostrar fotogramas incompletos, por ejemplo que sólo se vea la parte superior.

Además, si estás utilizando la cámara para otra cosa, como el streaming en directo o la detección de movimiento, las herramientas no pueden conectarse a /dev/video0 para tomar imágenes durante la grabación, en estos casos se hace necesario el acceso múltiple a la cámara.

La herramienta perfecta tiene que tener acceso exclusivo al dispositivo de vídeo mientras que permite que otras herramientas puedan tomar imágenes y vídeo al mismo tiempo. Además, necesita mantener la cámara activa mientras se toman imágenes, para así compensar el tema de la oscuridad. Para mí, esta milagrosa herramienta es mpjg-streamer (`http://bit.ly/2d2qSvQ`). Para instalarla en /usr/local sigue estos pasos:

```
$ git clone https://github.com/\
jacksonliam/mjpg-streamer.git
$ cd mjpg-streamer/\
mjpg-streamer-experimental
$ sudo apt-get install \
cmake libjpeg62-dev
$ make
$ sudo make install
```

Lo mejor es probar mjpg-streamer antes de activarla en el inicio. El programa cuenta con un número configurable de entradas (cámaras) y varios ajustes de salida. Puede funcionar como servidor HTTP, salida para un archivo en el disco local, salida como secuencia UDP/RTSP. En mis pruebas, la función RTSP no era estable y no funcionaba con cualquier cliente RTSP, es posible que el protocolo RTSP no sea compatible con el streaming de datos MJPEG en una ejecución estándar. En este tutorial, lo vamos a utilizarlo como servidor HTTP y utilizaremos otros procesos para leer desde mjpg-streamer.

Para iniciar mjpg-streamer como servidor web con autentificación y leer desde la primera cámara, ejecuta el comando:

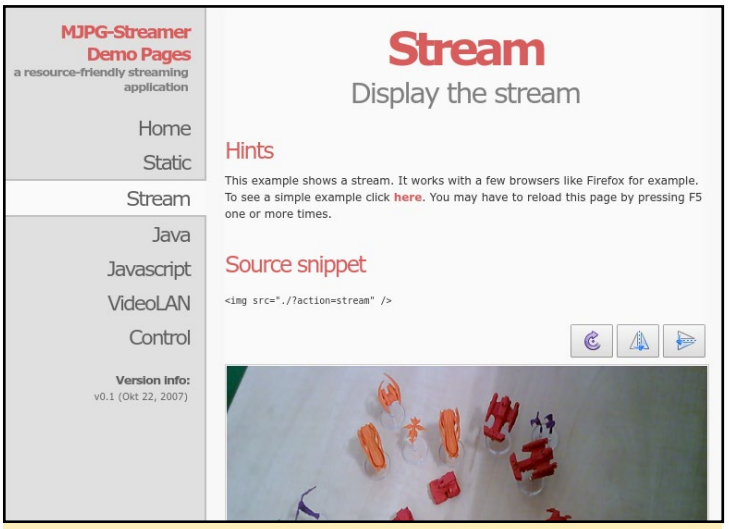
```
$ sudo /usr/local/bin/mjpg_streamer  -i 'input_uvc.
so -r 1280x720 -m 50000 -n -f 25 -d /dev/video0' -o
'output_http.so -p 8090 -w /usr/local/share/mjpg-
streamer/www/ -c odroid:odroidpass'
```

Este comando es complejo, de modo que vamos a explicar lo que hacen todos los parámetros. "-i" especifica el plugin de entrada que es input_uvc.so (grabar desde una cámara UVC). A continuación viene la resolución determinada por la cámara, y "m" especifica el tamaño mínimo de la entrada. He configurado esto a 50 KB, así mjpg-streamer descartará los fotogramas jpeg más pequeños (los fotogramas de 720p tienen un tamaño de alrededor de 120kB). Esto es muy bueno, porque a veces la cámara captura imágenes incompletas, que no son útiles. Sin embargo, esto tiene el efecto secundario de no capturar nada en condiciones de poca luz, ya que las imágenes en su mayor parte son oscuras y la compresión JPEG las reduce a menos de 50 KB. Tendrás que ajustar este parámetro de acuerdo con tu resolución de entrada.

El parámetro "-n" desactiva los controles dinámicos en el driver UVC, mientras que "f" especifica la tasa de fotogramas de entrada. "-d" indica el dispositivo de vídeo (/dev/video0 por defecto). Utilizamos el módulo output_http.so en el puerto "-p" 8090 y serviremos archivos HTTP desde el directorio apuntado por "w". Se pueden añadir opcionalmente una contraseña con el parámetro "-c" especificando la combinación nombre de usuario: contraseña. Tienes información detallada en `http://bit.ly/2dbB97p` y `http://bit.ly/2dbALWx`.

Una vez que se inicie con éxito mjpg_streamer como servidor HTTP, serás capaz de acceder a él con un navegador en http://<yourodroid- ip>:8090/. Se te solicitará la combinación nombre de usuario/contraseña y serás dirigido a la página demo, como se muestra la siguiente figura. Puedes, por supuesto, crear tu propia página, pero la página demo te proporciona toda la información necesaria para acceder a la cámara.

Puedes tomar una imagen fija desde tu ODROID con el siguiente comando:



Interfaz web de MJPEG Streamer con algunos muñecos moviendose

```
$ sudo apt-get install curl
$ curl -s -f -m 5 http://odroid:odroidpass@odroid-ip:8090/\
?action=snapshot > /tmp/snapshot.jpeg
```

Puede usar esto junto con crond para capturar imágenes en un intervalo de tiempo específico. También puede utilizar la fecha de registro como nombre de archivo o utilizar una herramienta como montage para añadir el registro temporal como marca de agua en la parte superior de la imagen. Aquí tienes un pequeño script que guarda fotografías en un directorio específico del disco local de tu ODROID y agrega la fecha y hora: `http://bit.ly/2d2fstx`. Además, puede utilizar ffmpeg en un script como éste para combinar todas estas imágenes en un vídeo y así facilitar su posterior visualización: `http://bit.ly/2cOzXqY`.

Captura de ejemplo con la fecha de registro superpuesta

Para obtener una secuencia de vídeo MJPEG desde la cámara, que básicamente es una secuencia de imágenes JPEG puedes ejecutar el siguiente comando:

```
$ vlc http://odroid:odroidpass@odroid-ip:8090/\
?action=stream
```

Si todo va bien y estas recibiendo imagenes, es el momento de añadir un script de inicio systemd para mjpeg_streamer. Crea un archivo llamado /etc/systemd/system/mjpg_streamer.service con los contenidos de `http://bit.ly/2dbCPxO`. Para activar el servicio, escribe los siguientes comandos:

```
$ sudo systemctl enable mjpg_streamer.service
$ sudo systemctl start mjpg_streamer.service
```

Para ver que el servicio se está ejecutando, consulta systemd:

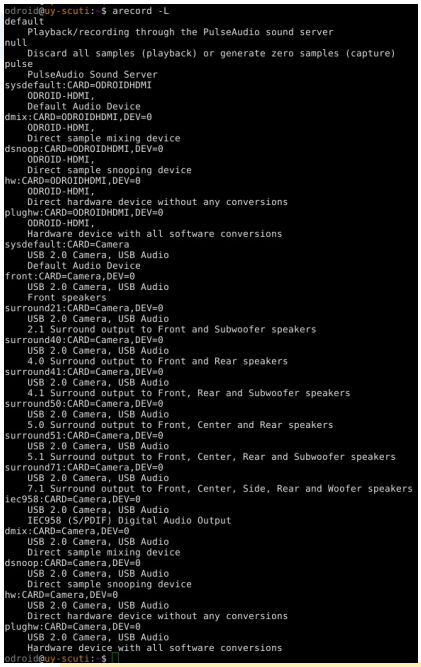
```
$ sudo systemctl status mjpeg_streamer.service
```

Recibir vídeos

Motion JPEG es compatible con todos los navegadores, aunque no soporta sonido y la compresión es bastante deficiente. La tasa de bits de 25fps en una secuencia MJPEG a 720p ocupa unos13Mbps, que puede ser alto para Internet. Para conseguir videos con sonido, vamos a necesitar multiplexar la secuencia MJPEG con una secuencia de sonido desde el micrófono de la cámara en un formato multimedia compatible.

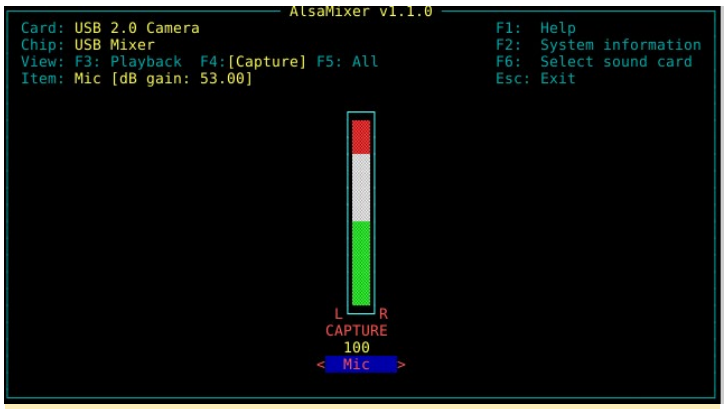
Como ya sabemos obtener la secuencia de vídeo, vamos a centrarnos en el micrófono. Puedes listar los dispositivos actuales soportados por ALSA con el comando “arecord -L”. Para la cámara web ODROID, debería ver varias entradas relacionadas con una cámara USB 2.0 con diversas posibilidades, tal y como se muestra en la siguiente figura. Necesitaremos el nombre para configurarla en ffmpeg más adelante (en nuestro caso usaremos el último - plughw:CARD=Camera,DEV=0).

```
$ arecord -L
```



Listado de dispositivos de audio

Antes de empezar a grabar, tenemos que comprobar que el micrófono no está en silencio y tiene un nivel aceptable de volutmen. Me gusta mucho el micrófono de la webcam 720p de Hardkernel, ya que está adaptado para poder oír susurros en una habitación con niños gritando, sin quedarse sordo en el proceso. Para ajustar el volumen vamos a utilizar alsamixer. En primer lugar, presione F6 para seleccionar la tarjeta de sonido y utilizar F4 para ir a la pestaña Capture. Utiliza las teclas de flechas para ajustar el nivel de sonido (yo lo puse al máximo).



Alsamixer mostrando el nivel de sonido

Ahora podemos crear nuestra consulta ffmpeg que grabará una secuencia de vídeo MJPG Streamer, añadirá audio desde ALSA y creará un archivo en el disco con esta combinación:

```
$ sudo apt-get install ffmpeg
$ ffmpeg -framerate 5 -f mjpeg -i 'http://odroid:odro
idpass@127.0.0.1:8090/?action=stream' \
-f alsa -i plughw:CARD=Camera,DEV=0 -acodec \
libmp3lame -c:v libx264 -preset ultrafast \
-r 5 -pix_fmt yuv420p -b:v 1500k \
-async 1 myvideo.mp4
```

El comando anterior especifica que la tasa de fotogramas de entrada debe ser 5 FPS y que la entrada es la secuencia mjpeg desde la dirección anterior. El parámetro "f" especifica que se debe usar ALSA para el audio desde el siguiente dispositivo en la lista. El audio debe ser codificado con mp3lame y el vídeo con h264 usando preset ultrarrápido y una tasa de fotogramas de 5 FPS. El ancho de banda de vídeo está limitado a 1500 kbps, sino ffmpeg no puede hacer la codificación en tiempo real. La opción async intenta sincronizar vídeo y audio, aunque a menudo se producen desviaciones. Finalmente, el último parámetro es el nombre del archivo que queremos escribir.

Con un ODROID-C2 puedes (casi) codificar via software hasta los 10 fps a 720p en tiempo real, pero el audio es incompresible, lo mejor es mantener una baja tasa de fotogramas. He compilado una versión optimizada de ffmpeg para el C2 usando el argumentado "-march=armv8-a+crypto+crc+fp+simd -mtune=cortex-a53", pero no he notado ningún cambio significativo en el rendimiento con esta versión optimizada. Dependiendo de tus necesidades, esto puede o no ser aceptable. Si quieres una alta tasa de fotogramas, necesitas reducir la resolución o cambiar a un XU4, el cual soporta la codificación por hardware. Puede leer más sobre esto en `http://bit.ly/2cxbMkK`.

Los mejores resultados los obtuve con mjpg_streamer configurado a 640x480, con ffmpeg a 10 fps y con un ancho de banda de 1 Mbps. Curiosamente, si bajamos esta configuración, el rendimiento es peor, en torno a 6 FPS. Si observas que el sonido se entrecorta, significa que ffmpeg no puede mantener la tasa de fotogramas impuesta. Por lo que he podido comprobar, si intentas grabar a una tasa de fotogramas superior a la que ffmpeg puede soportar en tiempo real, el sonido se entrecorta. El rendimiento de la codificación depende de la carga del sistema, de modo que a mayor carga menores FPS en tiempo real. Para ver algunas fórmulas que he probado y para ver cómo grabar sólo audio, consulta `http://bit.ly/2cvjBlO`.

Además volví a hacer las pruebas después de que Hardkernel realizara sus nuevos ajustes de overclock en boot.ini con el C2 ejecutándolo a 1.75GHz y los 4 núcleos. Con esta configuración, no conseguí sonido estable a 720p con 8 FPS (en vez de 5), y 15 fps con una resolución de 640x480, que es muy buena. No fui capaz de probar frecuencias más altas y menos núcleos debido a la excesiva inestabilidad, pero espero que las cosas mejoren con el tiempo. Además, si aumentas la frecuencia de la RAM a 1104 MHz, es posible que ganes 1 a 2 FPS.

Si deseas omitir mjpg_streamer por completo, también puedes leer directamente desde /dev/video0:

```
$ ffmpeg -r 5 -f v4l2 -video_size 640x480 \
-i /dev/video0 -f alsa \
-i plughw:CARD=Camera,DEV=0 -acodec \
libmp3lame -c:v libx264 -preset ultrafast \
-r 5 -pix_fmt yuv420p -b:v 1000k \
-async 1 myvideo.mp4
```

De hecho, el colaborador de los foros ODROID @crash-override acaba de lanzar recientemente una librería y un programa de prueba que te permite codificar H.264 vía hardware en el C2 al máximo de FPS, pero necesitas acceso raw a tu cámara, de modo que mjpg_streamer debe estar desactivado. Se está trabajando para mejorar este método, así que revisa su hilo de soporte para las actualizaciones en `http://bit.ly/2dcQDJn`,

Transmitir RTSP bajo demanda

El principal uso de una cámara IP es el de poder ver vídeo cuando se solicite. En el mejor de los casos, debería ser visible por varios usuarios a la vez. Para hacer esto, utilizaremos ffserver para crear una secuencia RTSP que se ejecute a demanda.

El Protocolo de flujo en tiempo real (RTSP) es un protocolo similar al SIP que se encarga de la señalización y el transporte de los medios entre cliente y servidor. Normalmente, la señalización se realiza en el puerto TCP 554 y los flujos de datos a través del UDP con el cliente y el servidor negociando un puerto adecuado. Sin embargo los firewall y NAT a veces interfieren en la negociación, de modo que también hay una forma de transportar los datos a través de TCP intercalado con control de tráfico. Este método será usado en nuestras pruebas.

La aplicación ffserver proporciona una forma de servir las solicitudes del cliente RTSP basadas en señales de video ffmpeg. Es parte del paquete ffmpeg, de modo que ya lo tienes instalado si has seguido el tutorial hasta este punto. Para iniciar el servidor, necesitarás una configuración adecuada y un script de inicio systemd. La configuración debe guardarse en /etc/ffserver.conf y se puede conseguir desde `http://bit.ly/2cYWPcq`.

Si echas un vistazo a la configuración, monta un sistema oyente en el puerto RTSP 554, define un canal llamado mjpg-streamer.ffm y lo vincula a un flujo de salida llamado live.h264.sdp. La aplicación ffserver permite configurar diferentes formatos de salida, pero en este ejemplo experimentaremos con el flujo de entrada que será el h264 de por sí.

Para arrancar ffserver en el inicio, añade el siguiente servicio systemd al archivo /etc/systemd/system/xserver.service:

```
https://github.com/mad-ady/odroid-webcam-scripts/
blob/master/ffserver.service
```

Para activarlo y ver su estado:

```
$ sudo systemctl enable ffserver
$ sudo systemctl start ffserver
$ sudo systemctl status ffserver
```

En este punto, tienes un servidor RTSP escuchando las solicitudes, aunque el video no se está procesando. Para iniciar una secuencia de vídeo, necesitas ejecutar ffmpeg así:

```
$ /usr/bin/ffmpeg -loglevel 8 \
-r 5 -f mjpeg -i 'http://odroid:odroidpass@127.0.0
.1:8090/?action=stream' \
-f alsa -i plughw:CARD=Camera,DEV=0 \
-acodec libmp3lame -c:v libx264 \
-preset ultrafast -r 5 \
-pix_fmt yuv420p -b:v 1500k \
-async 1 -x264-params keyint=30:no-scenecut=1 \
-vf "drawtext=fontfile=/usr/share/fonts/truetype/
dejavu/DejaVuSans-Bold.ttf: text='Webcam feed
%{localtime\\:%F %T}': fontcolor=white@0.8: x=7:
y=5" \
-override_ffserver http://localhost:8099/mjpg-
streamer.ffm
```

Antes de que te asustes por la complejidad de este comando, te dire que es similar a la que has visto antes, simplemente le hemos añadimos un texto superpuesto en la esquina superior izquierda con la fecha y hora, al igual que las cámaras IP "profesionales". La aplicación ffmpeg envía la salida a ffserver, especificando el nombre del canal.

Ahora deberías ser capaz de conectarte con un visor RTSP y disfrutar de la señal de video. Si lo estás probando desde tu teléfono inteligente Android, puede intentarlo con RTSP Viewer, disponible en `http://bit.ly/2cvl0J8`:

```
$ vlc rtsp://odroid-ip:554/live.h264.sdp
```

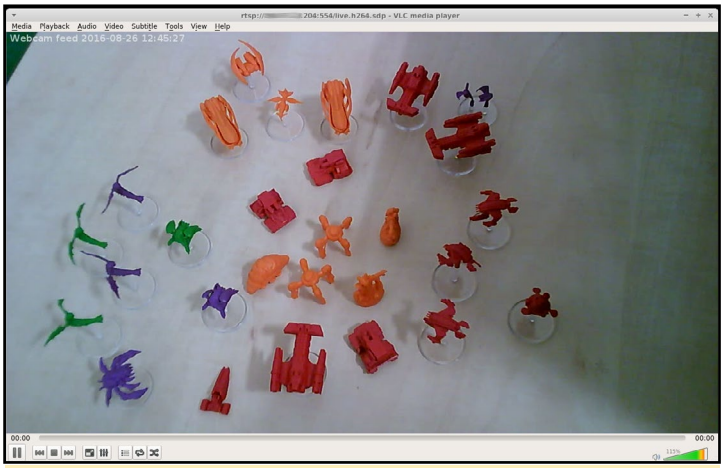
Para hacer las cosas más permanentes, puedes agregar el archivo de servicio systemd ffmpeg "https://github.com/mad-ady/odroid-webcam-scripts/blob/master/ffmpeg.service" en /etc/systemd/system/ffmpeg.service. Para activarlo y ver su estado, escribe los siguientes comandos:

```
$ sudo systemctl enable ffmpeg
$ sudo systemctl start ffmpeg
$ sudo systemctl status ffmpeg
```

El XU4 con el kernel estándar ya puede codificar vía hardware, de modo que tus cámaras estarían conectadas a otros equipos como el C2, ejecutarías ffserver en el XU4 y leerías la secuencia MJPEG desde el C2 través de la red (mejor por cable), así obtendrías audio mp3 desde un ffserver que se ejecutaría en el C2 y transcodificarías el vídeo en el XU4 antes de entregarlo al visor. Cuando mi XU4 este operativo, tengo pensado redireccionar la transcodificación a éste y publicar los cambios en el hilo de soporte con el objetivo de mejorar el soporte para múltiples cámaras/secuencias.

Mejorar el rendimiento idle

Esta previsto que las secuencias de vídeo estén funcionando en todo momento, lo que significa que ffmpeg debe transcodificar incluso si no hay un visor conectado. Esto puede estar



Transmisición RTSP con sonido

bien si se espera tener muchos visores conectados al mismo tiempo, pero si tienes la intención de conectarte en contadas ocasiones (por ejemplo, 5 minutos/día), no vale la pena tener la transcodificación de secuencias de fondo cuando no se utiliza. Sería mejor si tuviéramos un sistema que permitiera activar el inicio de la secuencia de vídeo cuando un visor se conectara y detener la secuencia cuando se desconectaran todos los visores. Para este escenario escribí el script ffserver-trigger

El script se ejecuta en segundo plano y ejecuta continuamente el comandos tail -f en /var/log/syslog. Recoge los mensajes de ffserver como "PLAY live.h264.sdp", comprueba si la secuencia de video está activa o la pone en marcha en caso contrario. También buscará mensajes de parada como "RTP/TCP" y detiene la secuecia si fuera necesario. Registra sus acciones en syslog, para mayor comodidad. Ten en cuenta que este sistema de detección está hecho para un único flujo y sigue la nomenclatura utilizada en el artículo. Puede que necesites modificarlo si quieres utilizarlo para otras configuraciones.

Para instalar ffserver-trigger, escribe los comandos:

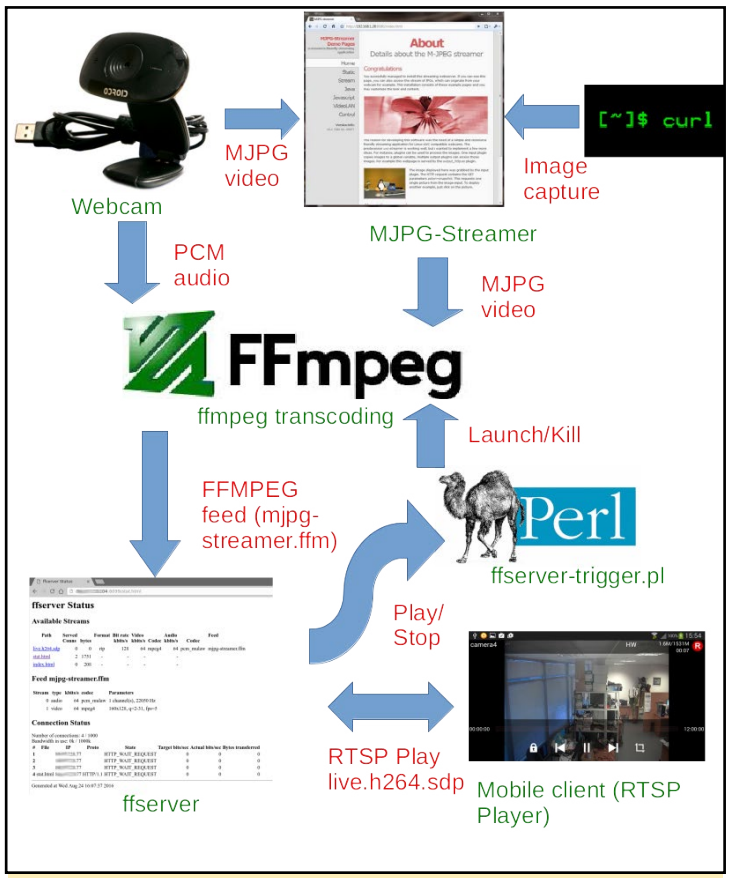
```
$ sudo apt-get install libfile-tail-perl
```

```
$ sudo perl -MCPAN -e 'install Linux::Proc::Net::TCP'
$ sudo wget -O /usr/local/bin/ffserver-trigger.pl \
https://raw.githubusercontent.com/mad-ady/\
ffserver-trigger/master/ffserver-trigger.pl
$ sudo chmod a+x /usr/local/bin/ffserver-trigger.pl
$ sudo wget -O /etc/systemd/system/ffserver-trigger.
service \
https://raw.githubusercontent.com/mad-ady/\
ffserver-trigger/master/ffserver-trigger.service
$ sudo systemctl enable ffserver-trigger
$ sudo systemctl start ffserver-trigger
$ sudo systemctl status ffserver-trigger
```

Puesto que ahora estás usando el ffserver-trigger, deberías desactivar el servicio ffmpeg para que no se iniciará automáticamente en el arranque. En su lugar, activa ffserver-trigger cuando sea necesario.

```
$ sudo systemctl disable ffmpeg
```

La figura 6 muestra la dinámica completa del trabajo.



Nuestra línea de trabajo de streaming

Si también quieres grabar la secuencia en un archivo, puede conectarte con un visor RTSP y volcarlo a un archivo sin la transcodificación. Esto supone una gran ventaja, ya que lo puedes hacer incluso con otros clientes conectados sin interrumpir su experiencia:

```
$ ffmpeg -i rtsp://127.0.0.1:554/live.h264.sdp \
 -acodec copy -vcodec copy rtsp-recording.mp4
```

En cuanto a los retrasos en el tratamiento del vídeo, mjpg_streamer tiene un retraso de alrededor de 1 segundo, mientras que ffmpeg + ffserver añade entre 2 y 3 segundos adicionales. A estas velocidades, tu experiencia no será en tiempo real y no es la adecuada para el control remoto de un robot, pero si es lo suficientemente buena para la visualización remota.

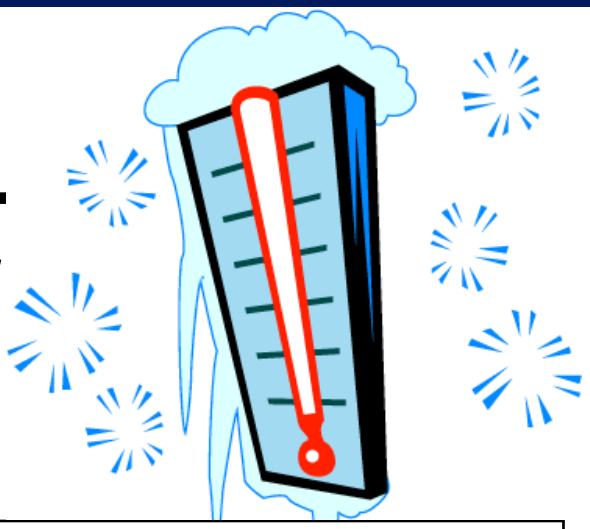
Consejos solucionar problemas

- Pregunta: Soy incapaz de conseguir imágenes desde mjpg_streamer/ffmpeg, parece estar bloqueado.
Respuesta: Comprueba el valor del parámetro -m y bájalo para adaptarlo a tus necesidades.
- Pregunta: ¿Cómo puedo solucionar la sincronización del sonido?
Respuesta: Inténtalo con 640x480@10 fps o reduce la tasa de fotogramas en ffmpeg.service.
- Pregunta: ¿Por qué al detener la secuncienca RTSP se paran todos los clientes conectados?
Respuesta: A veces ffserver se bloquea por una violación de acceso cuando un cliente se detiene. Se puede reinicia automáticamente por systemd, pero desconectará a todos los clientes.
-Pregunta: Pulsando Play con el primer cliente conectado no se inicia la secuencia RTSP cuando utilizo ffserver-trigger, ¿por qué?
Respuesta: Es un problema conocido. La secuencia RTSP tiene una pausa de unos 10 segundos antes de que ffserver empiece a enviar datos de vuelta al cliente. Presiona Play de nuevo tras el tiempo de espera. Si un cliente se conecta cuando una secuencia está activa, este problema no ocurre. El script de detección tiene un período de inactividad de 20 segundos en el que ignora las peticiones de parada tras el inicio de una secuencia para mitigar esta cuestión.
-Pregunta: A veces se conecta a una secuencia que no funciona y ffmpeg parece estar bloqueado. ¿Cómo puedo solucionar esto?
Respuesta: La causa es mjpg_streamer. A veces se queda atascado y necesita reiniciarse. Existen dos líneas que puede descomentar en ffserver-trigger.pl para reiniciarlo automáticamente cuando ffmpeg vuelva a arrancar para evitar esto.
- Pregunta: Una cámara web comercial permite giro e inclinación. ¿Cómo lo puedo añadir a mi cámara?
Respuesta: Puedes añadirlos con algunos motores y pines PWM o un Arduino (`http://bit.ly/2diWcKh`).

Si tienes otros problemas o si encuentras una mejor forma de lograr esto, no dudes en hacérmelo saber en el hilo de soporte de esta guía en `http://bit.ly/2d2j6DH`.

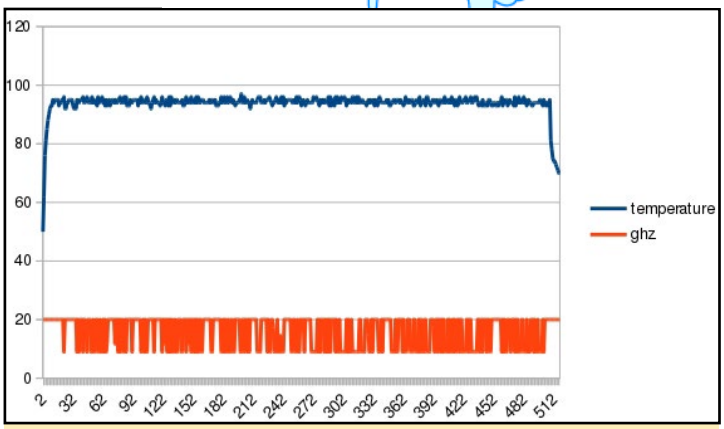
PRUEBAS DE REFRIGERACION E

DESCUBRE LA MEJOR REFRIGERACION SEGUN TUS NECESIDADES

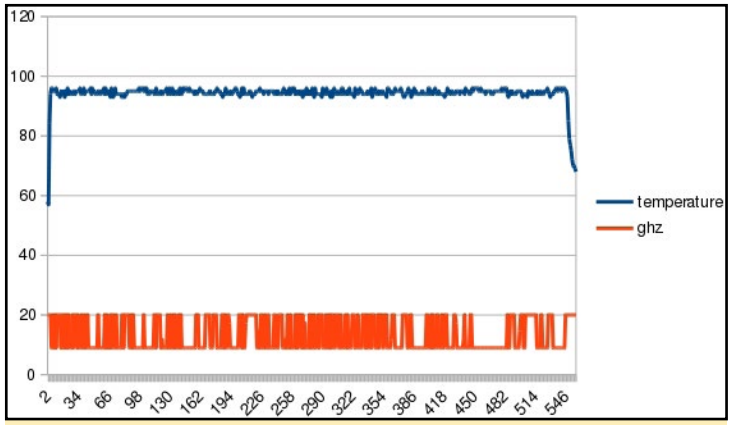


por Bo Lechnowsky

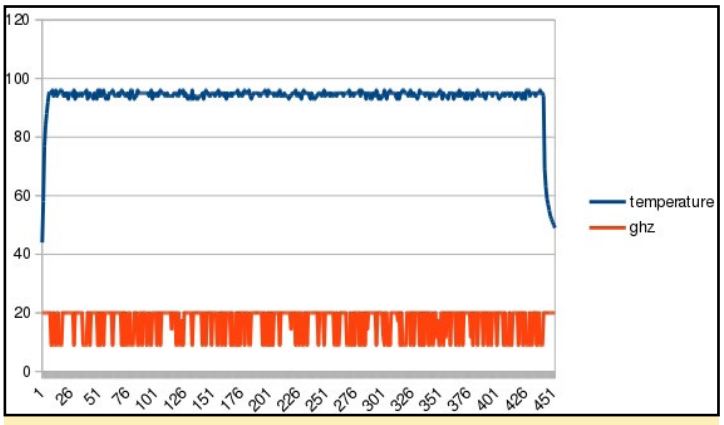
He ejecutado varias pruebas de temperatura usando un XU4 con diferentes tipos de sistemas de refrigeración. El objetivo era encontrar la forma más eficaz de enfriar el dispositivo mientras trabajase con una alta carga de trabajo. En los siguientes gráficos, la temperatura está indicada en azul, y la velocidad en naranja. El dispositivo regula su velocidad en función de la temperatura, y la mejor situación se da cuando la temperatura permanece por debajo 95 grados centígrados, lo cual minimiza el estrangulamiento, manteniendo una velocidad de reloj lo más cercana posible a los 2 GHz. Cada prueba fue ejecutada durante 5-6 minutos aproximadamente, lo cual se indica en el eje X, y la temperatura tiende a permanecer por debajo de 100 grados Celsius que se indica en el eje Y.



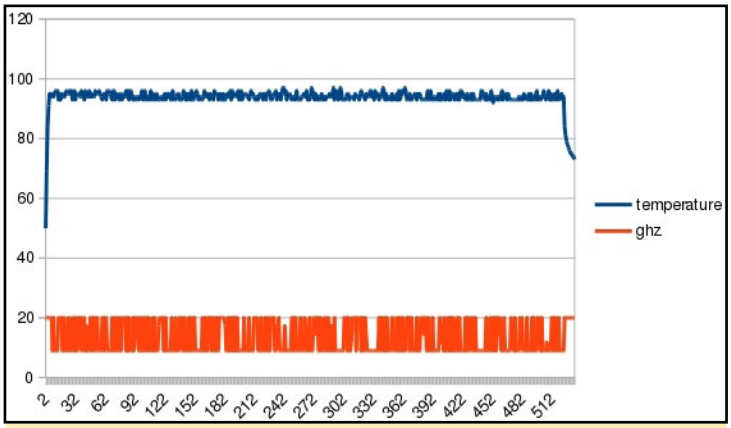
Carcasa de aluminio con refrigeración pasiva Shapedmedia
Velocidad de reloj efectiva bajo carga: 1.56GHz



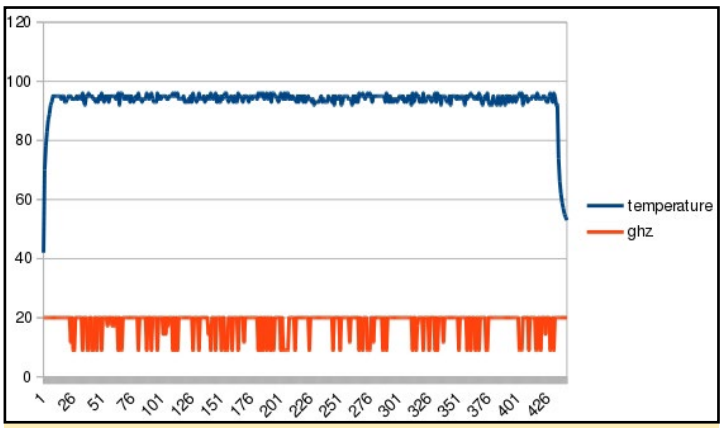
Refrigeración Activa por defecto en el XU4
Velocidad de reloj efectiva bajo carga: 1.32GHz



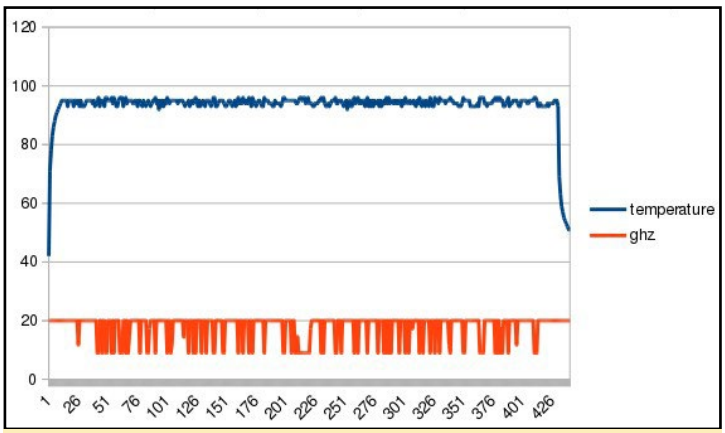
Disipador pasivo Northbridge (Blue Zalman) con un ventilador de 120 mm
Velocidad de reloj efectiva bajo carga: 1.67GHz



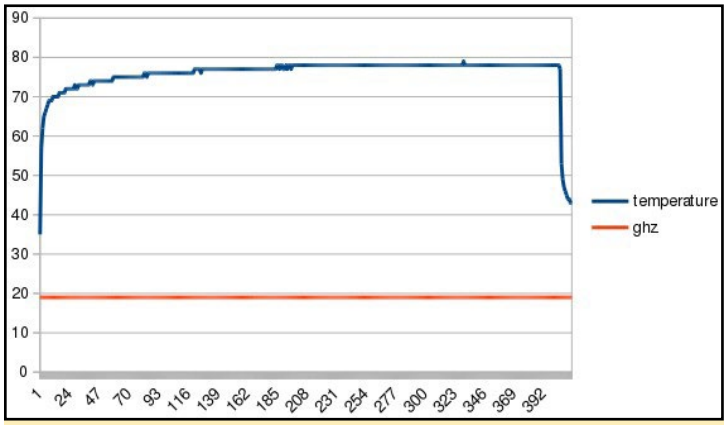
Disipador pasivo Northbridge (Blue Zalman)
Velocidad de reloj efectiva bajo carga: 1.35GHz



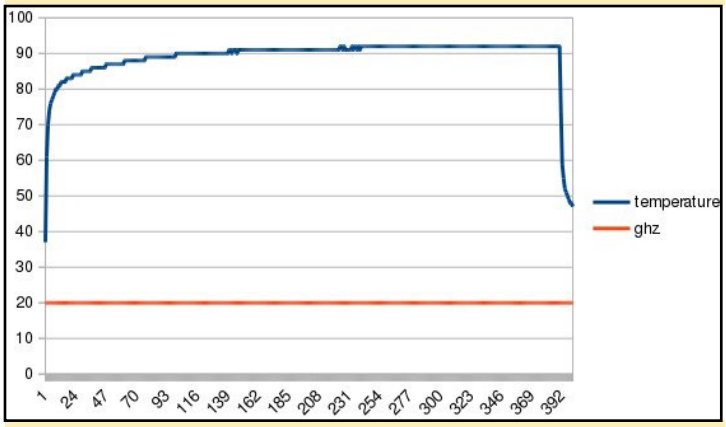
Disipador activo Northbridge (Gold) con un ventilador Noctua 5VCDC de 40mm
Velocidad de reloj efectiva bajo carga: 1.78GHz



Disipador pasivo Northbridge (Gold) con ventillador de 120mm
Velocidad de reloj efectiva bajo carga: 1.79GHz



Disipador activo Northbridge (Gold) con compuesto térmico y con un ventilador Noctua 5VCDC de 40mm @ 1.9GHz
Velocidad de reloj efectiva bajo carga: 1.9GHz



Disipador activo Northbridge (Gold) con compuesto térmico y con un ventilador Noctua 5VCDC de 40mm @ 2.0GHz
Velocidad de reloj efectiva bajo carga: 2.0GHz

La mejor forma de enfriar el ODROID-XU4 es utilizar un compuesto térmico junto con un ventilador Noctua 5V DC de de 40mm y un disipador activo Northbridge, que puede ser adquirido en http://bit.ly/2cBeTGm. Mantiene la temperatura por debajo de 93 grados Celsius, y permite que el dispositivo funcione a 2,0 GHz sin estrangulamiento. El método de refrigeración pasiva más efectivo fue la carcasa Shapedmedia, disponible de Ameridroid en http://bit.ly/2d4YCMH, que mantenía al ODROID-XU4 en unos 95 grados centígrados.

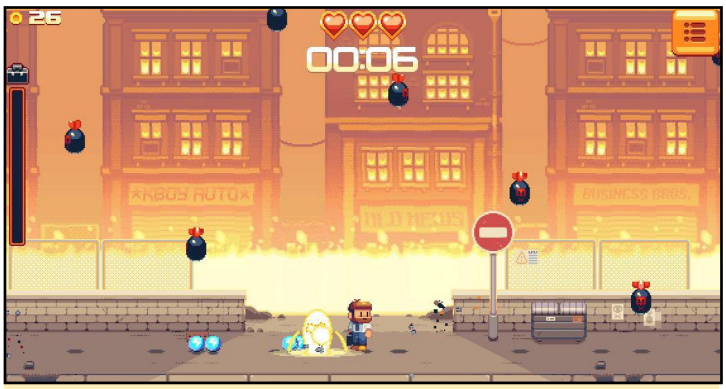
ULTIMATE BRIEFCASE

ESTAR LISTO PARA PELEAR EN UN JUEGO FRENETICO DE SUPERVIVENCIA

por Bruno Doiche

Pues Bueno, es el típico día en el que estás dando una vuelta con tu maletín, y por un pequeño error te resbalas con una cáscara de plátano, escapando de la primera de muchas, muchas, bombas que te están lanzando.

¿Que ha pasado? ¿No va a terminar nunca? ¿Hay alguna razón para tanta violencia en contra de tu persona? Esquivar las bombas, conseguir poderes, desbloquear objetos, reclutar a otros personajes de grupos realmente entrañable, y estar listo para un juego super entretenido. ¡Si eres lo suficientemente bueno, es muy posible que descubras en esta historia mucho más de lo que se aprecia a simple vista!e!



¿Quién de nosotros nunca se ha tomado la molestia de sobrevivir a un bombardeo interminable?

Puedes utilizar los elementos para sobrevivir más tiempo

```
https://play.google.com/store/apps/details?id=com.nitrome.ultimatebriefcasew
```

AUTOMATIZACION INDUSTRIAL

MONITORIZA REMOTAMENTE REGISTROS MODBUS UTILIZANDO UN ODROID-XU4

por Joel Duncan

El campo de la automatización industrial no es my conocido por adoptar tecnologías punteras a corto o medio plazo. Esto se debe en parte al gran monopolio de los tres actores principales: Siemens, Allen Bradley y Wonderware. Por diversas razones, no suelen implantar innovaciones de forma agresiva, como la creación de aplicaciones web industriales nativas. En Bubble Automation, nos dimos cuenta de esta deficiencia. La mayoría de los clientes que querían monitorizar remotamente sus sitios estaban atrapados usando ineficaces complementos exclusivos. Algunos de estos complementos requerían una licencia y un coste de mantenimiento muy altos, o que las conexiones inseguras de TeamViewer/VNC necesitasen herramientas de terceros para instalarse en los sistemas del cliente.

Objetivos del proyecto

Queríamos desarrollar una aplicación web segura, nativa y moderna que no requiriera ningún navegador o plugins para PC especiales. Durante el tratamiento de los datos en tiempo real, es necesario que la aplicación funcionase en cualquier dispositivo, incluyendo teléfonos inteligentes, tablet, netbooks y ordenadores de sobremesa. El diseño exigía la no dependencia de aplicaciones específicas de plataformas innecesarias. Para asegurar altos niveles de seguridad, el hardware del servidor web tenía que ubicarse en las instalaciones del cliente, ser lo suficientemente pequeño para ser instalado en un panel de control, y ser lo suficientemente robusto como para sobrevivir a las condiciones industriales.

Elegir en entorno de trabajo

NodeJS fue el primer entorno de trabajo que probamos. Parecía prometedor, pero en el momento de la evaluación, no era la plataforma más estable o mejor soportada. Su mayor problema era que conectaba a una base de datos utilizando un módulo de base de datos experimental. Tras realizar varias pruebas con diferentes entornos, nos conformamos con una simple solución en Python , ya que contábamos con personal experimentado en técnicas avanzadas de Python y en su amplia gama de módulos específicos.

Seleccionar el hardware

Después de una breve experiencia usando ordenadores de placa reducida (SBC) de calidad industrial, era obvio que no eran adecuados. La mayoría utilizan el desfasado Intel Atom en grandes recintos ruidosos. Esto nos condujo al actual y competitivo mercado de los PCs del tamaño de una tarjeta de crédito. Aquí tienes algunos que probamos:

- Raspberry Pi 2 Model B
- Raspberry Pi 1 Model B
- ODROID-U3
- ODROID-C1
- ODROID-XU4

Cada placa fue ampliamente probada ejecutando nuestro entorno de trabajo Python con cargas poco realistas. La placa que más destacó fue la ODROID-XU4 que era claramente superior al resto, tal y como muestran los resultados en las figuras 1-4.

Esta placa tenía mucho mejor rendimiento en CPU y en red, pero lo más importante, presentaba velocidades E/S

Figura 1 – Los resultados Dhrystone son un índice relativo a una estación SPARC 20-61, valorado en 10,0

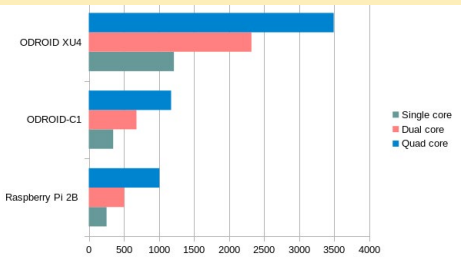


Figura 2 – Los resultados Whetstone son un índice relativo a una estación SPARC 20-61, valorado en 10,0

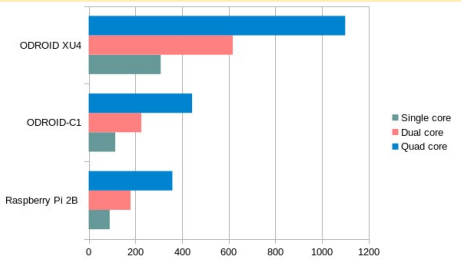
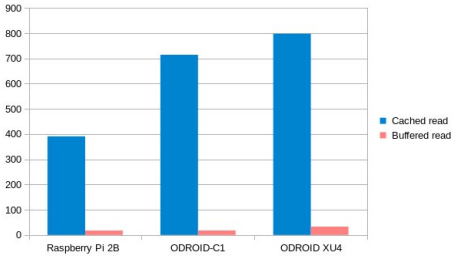


Figura 3 – Los resultados Hdparm están en megabytes por segundo



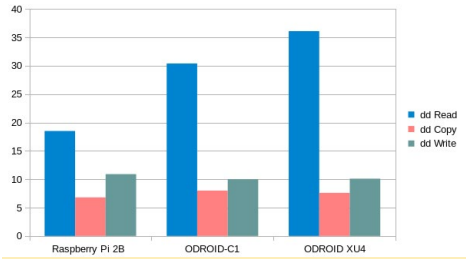


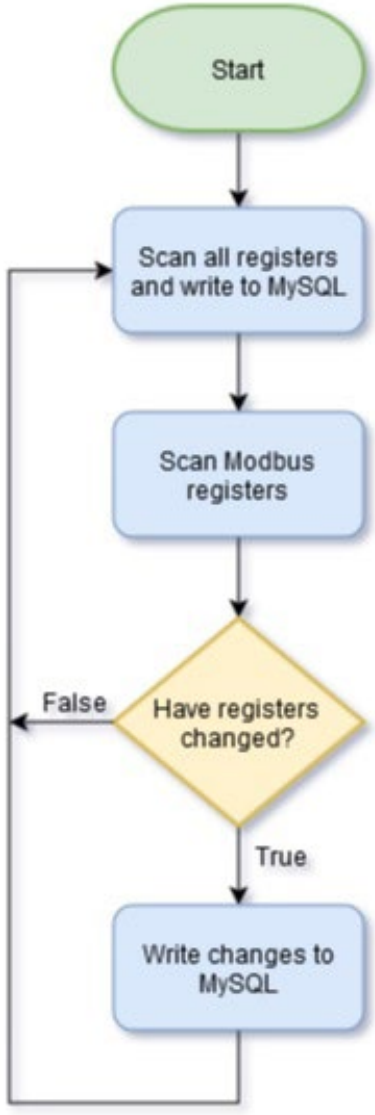
Figura 4 - Los resultados DD están en megabytes por segundo

y ancho de banda mucho más altos, lo cual era esencial para nuestra base de datos en MySQL.

Arquitectura y Diseño de Software

El punto de partida era crear un demonio que podía leer los registros Modbus a través de TCP/IP desde el campo, como muestra la Figura 5. Este creció después a un sistema basado en eventos sensibles al contexto que traduciera eventos de campo en alarmas, movimiento en directo, gráficas históricas, registros de eventos y emails de notificación.

Figura 5 - Diagrama de flujo de la aplicación



Proporcionar esta información al cliente resulto ser una tarea compleja. La visualización de información en tiempo real en una página web usando HTML5 puro y sin plugins, siempre ha sido un reto. El uso de técnicas tales como long polling sólo tendría como resultado que el navegador se colgara, ya que simplemente no hay demasiada información para nutrir al navegador a una velocidad relativamente alta, a intervalos de lectura de alrededor de 1 segundo.

NodeJS era un buen candidato para las soluciones de datos en tiempo real, debido a su eficiente integración con WebSockets, que es una tecnología que proporciona comunicación bidireccional total través de una conexión TCP. Afortunadamente, nos encontramos con una potente librería PHP que proporciona esta funcionalidad. Por esta razón, pudimos desarrollar todos los componentes del servidor en PHP utilizando Twitter Bootstrap y así proporcionar una interfaz sencilla con un diseño adaptable desde la base.

Un sistema LEMP (Linux, Nginx, MySQL, y PHP) fue el utilizado para proporcionar una base sólida en términos de velocidad, estabilidad y fiabilidad de nuestro front-end en Javascript, PHP y HTML. Las principales razones para usar Nginx sobre Apache fueron su mejor uso de los procesadores multinúcleo asignando un trabajador por núcleo y su mejor trayectoria en terminos de seguridad frente a la de Apache.

Figura 6 – Pantallazo del Front-end

Seguridad

Debido a la naturaleza de nuestra industria, desarrollar nuestra aplicación para ser segura a conciencia era algo clave. Se dedico mucho tiempo a la creación de una conexión segura no vulnerable a la inyección SQL, a la usurpación de la sesión, el cross-site scripting y los ataques de fuerza bruta. Para ello, hemos creado una imagen Linux mínima reforzada para utilizar en nuestros servidores que se mantiene al día con todos los parches de seguridad actuales. Trabajamos estrictamente sobre una base sin control, lo que significa que en el improbable caso de que nuestro software se vea comprometido, no hay forma de que el intruso pueda dañar el sitio. Nuestro software sólo monitoriza el estado del sistema y no afecta el proceso de control.

Para comentarios, preguntas y sugerencias, por favor visite el post original en http://bit.ly/2cp6tzj.

JUEGOS NVIDIA EN STREAMING SOBRE EL C2

EJECUTA JUEGOS MODERNOS EN TU ODROID

por @khaine

La tecnología GameStream de NVIDIA te permite transmitir juegos desde un PC con Windows impulsado con una GeForce a otro dispositivo. Oficialmente sólo soporta los propios dispositivos SHIELD basados en Android de Nvidia, pero con un cliente GameStream de código abierto de terceros llamado Moonlight, puedes transmitir los juegos a tu ODROID.

Instalación en un PC

En primer lugar, tendrás que configurar NVIDIA GameStream en tu PC Windows, y necesitas utilizar una tarjeta de video NVIDIA para que esto funcione. Si no tiene instalado el software GeForce Experience, tendrás que descargarlo e instalarlo desde NVIDIA en http://bit.ly/1kIWAdz. Despues, inicia la aplicación "GeForce Experience" desde el menú Inicio. Haz clic en la pestaña "Preferences" en la parte superior de la ventana de la aplicación y selecciona la categoría "SHIELD". Asegúrate que la casilla "Allow this PC to stream games to SHIELD devices" esta marcada.

Si quieres añadir algún juego específico que GeForce Experience no encuentra automáticamente, puedes añadirlo a la lista de juegos en Preferences -> Shield. De hecho se puede añadir cualquier programa, incluso programas de escritorio.

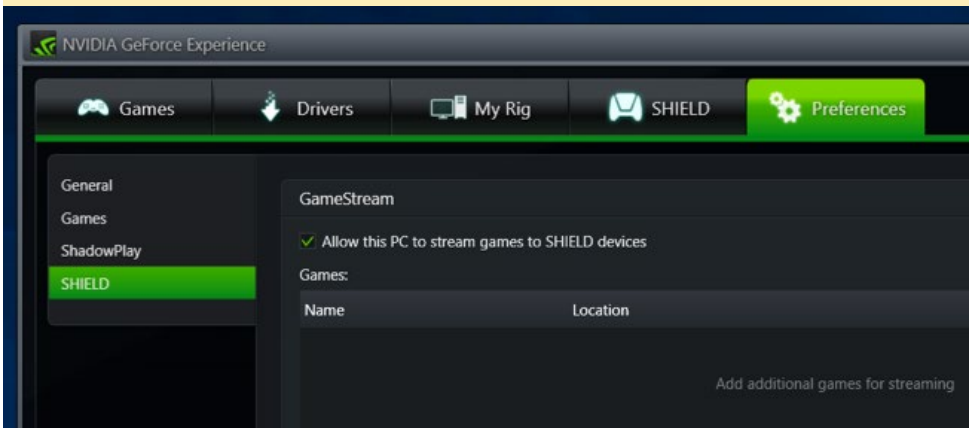
Instalación de Moonlight

1. 1. Instala la imagen Debian Jessie para ODROID-C2 desde http://bit.ly/2cj6V6F y arrancalo.

2. Actualiza la imagen con el siguiente comando, que posiblemnte te llevará un tiempo:

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

Figura 1 – Configuración de juegos NVIDIA vía Streaming en el PC



3. Instala Moonlight:

```
$ sudo apt-get install moonlight-
embedded
```

4. Instala PulseAudio (la nueva versión de pulseaudio presenta un menor retardo en el audio):

```
$ apt-get install -t \
  jessie-backports pulseaudio
```

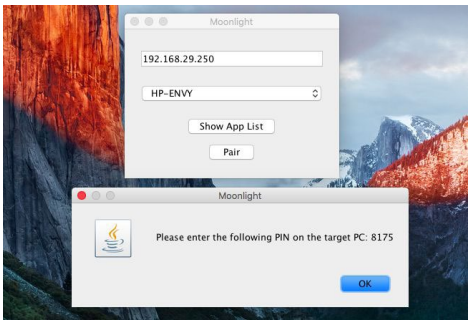
5. Reinicia el equipo, en este punto Moonlight debería estar funcionando tanto en H.264 y H.265.

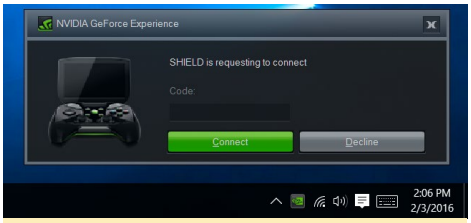
6. Si además quieres utilizar Kodi, ejecuta el siguiente comando e instala tanto el escritorio Mate como Kodi:

```
$ sudo setup-odroid
```

9. Si también deseas iniciar automáticamente Kodi, puedes ir a la sección de "Startup Applications" del Control Center y añadir Kodi.

10. Para iniciar Moonlight directamente desde Kodi, puede instalar Luna desde http://bit.ly/2cWy3sD. Si sólo utilizas Steam, puedes automatizar algunas cosas por ti mismo. Por ejemplo, podrías crear una unidad systemd para transmitir Steam creando un archivo en /etc/systemd/system/steam.service con el siguiente contenido:





Figuras 1 y 2 - Conectando NVIDIA Game Stream utilizando el PIN generado

```
[Unit]
Description = Steam Streaming
After = systemd-user-sessions.
service network.target sound.
target
Conflicts = kodi.service
OnFailure=kodi.service

[Service]
User = odroid
Group = odroid
Type = simple
ExecStart = /usr/bin/moonlight
stream -app Steam -60fps -1080
STREAMING_MACHINE_IP
ExecStop = /usr/bin/moonlight
quit STREAMING_MACHINE_IP
ExecStopPost = /usr/bin/sudo /
bin/systemctl start kodi
```

Después, añade un acceso directo al archivo Kodi System.Exec en /home/odroid/steam.sh que apunte al script steam.sh:

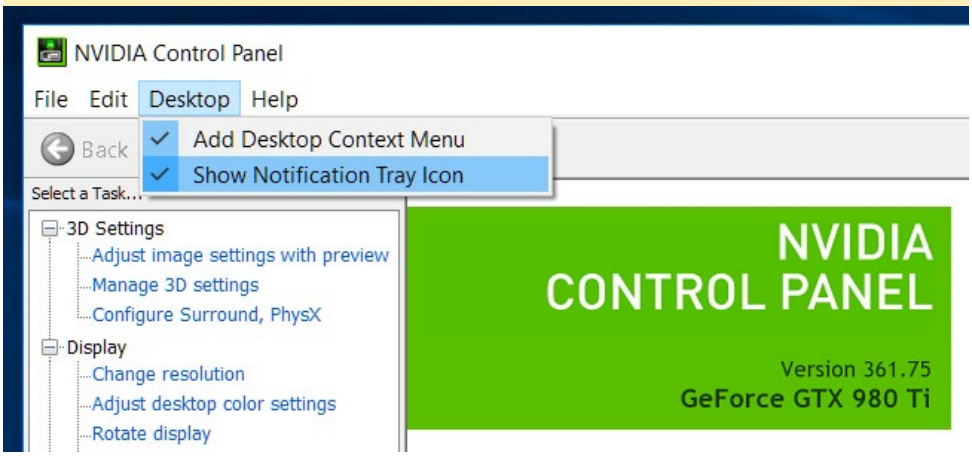
```
#!/bin/bash
sudo /usr/bin/nohup /bin/system-
ctl start steam &
```

Ahora deberías tener tanto Kodi como Moonlight funcionando pudiendo alternar entre ellos con facilidad. Para conectarte a tu PC, la app Moonlight te dará un PIN. Introdúcelo en la ventana pop-up "SHIELD is requesting to connect" que aparece en tu PC y se emparejaran tus dispositivos.

Si no aparece el cuadro de diálogo solicitando el PIN, abre la aplicación NVIDIA Control Panel en el PC de Windows, haz clic en el menú "Desktop" y selecciona "Show Notification Tray Icon" La próxima vez que intentes vincular los dispositivos, aparecerá la ventana pop-up del PIN. Por alguna razón la ventana pop-up está vinculada a este icono de la bandeja del sistema y no aparecerá si no lo activas.

Para comentarios, preguntas o sugerencias, por favor visita el post original en http://bit.ly/2cYgG74 o echa un vistazo a la documentación oficial en http://bit.ly/1skHFjN.

Figura 3 – Seleccionando Show Notification Tray Icon



JUEGOS LINUX

ALGUNOS EXCELENTES JUEGOS PARA EL C2

por Tobias Schaaf

El ODROID-C2 es un gran dispositivo para jugar, y este mes me gustaría destacar algunos de los juegos que son compatibles con el C2. Recientemente he actualizado varios con nuevos binarios y mejoras. Recomiendo usar mi imagen Debian Jessie de 64 bits, que ya tiene instalado mi repositorio. Si está utilizando una distribución de Ubuntu o Debian diferente, escribe los siguientes comandos para acceder a mis paquetes de software:

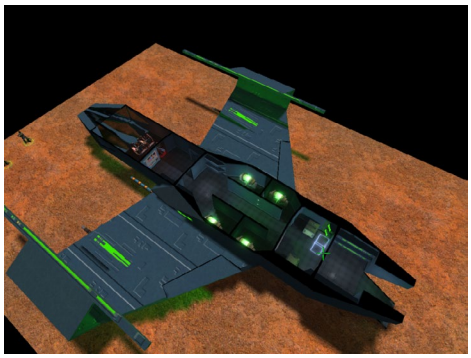
```
$ wget http://oph.mdrjr.net/meveric/\
sources.lists/meveric-jessie-main.list
$ wget -O- http://oph.mdrjr.net/meveric/\
meveric.asc | apt-key add -
$ apt-get update
```

Por desgracia, estos podrían no funcionar con todas distribuciones, ya que algunas dependencias todavía no están disponibles en Ubuntu.

UFO: Alien Invasion

UFO: AI es una nueva versión de la serie original XCom con un aspecto más moderno y con mejores gráficos. Originalmente, @ptitSeb lo exporto a Pandora y yo lo he utilizado para llevarlo a ODROID, pero por ahora muchos de sus cambios se integraron en el proyecto principal y he sido capaz de compilar la nueva versión de UFO: AI para ODROIDs. El juego ofrece muy buenos gráficos en 3D en diferentes idiomas y resoluciones con muy buena música. Es similar al juego original XCom, pero con un estilo más moderno

Figuras 1 - 3 - UFO: Alien Invasion



Para descargar e instalar UFO: Alien Invasion, escribe el siguiente comando:

```
$ sudo apt-get install ufoai-odroid
```

SmokinGuns

SmokinGuns es un juego de disparos al estilo lejano oeste basado en el motor ioQuake (Quake 3). Es muy divertido en el modo multijugador y ofrece algunos modos de juego muy interesantes, como defender un banco de los bandidos. Se ejecuta sin problemas en el ODROID-C2, incluso con una alta configuración. Puede ver un vídeo del juego en https://youtu.be/RGIYZf-BBfA.

Para descargar e instalar SmokinGuns,

Figuras 4 - 6 - SmokinGuns

escribe el siguiente comando:

```
$ sudo apt-get install
smokinguns-odroid
```

CorsixTH

CorsixTH es una nueva versión del antiguo clásico Theme Hospital, similar a OpenTTD. Se trata de una reconstrucción muy próxima a la versión original, con algunas mejoras como las diferentes resoluciones. El juego todavía está en desarrollo, pero está lejos de ser simplemente una versión alfa. Puedes jugar el juego original sin problemas y con todas sus novedades, y mejorará con el tiempo cuando se liberen nuevas versiones. Tienes disponible un vídeo del juego en https://youtu.be/rSN1p247J74. Necesitarás los archivos originales para poder jugar al juego. Recientemente ha sido lanzada la versión 0.60, la cual añade las siguientes características:

- Campañas de usuario: ahora es posible crear una serie de niveles para jugar a la vez del mismo modo que el juego original.
- Editor de mapas: hay disponible un nuevo editor de mapas directamente desde el menú del juego.
- Impacto en el precio de los medicamentos: Ahora los pacientes reaccionaran con el precio que fijes a los tratamientos. Si los tratamientos cuestan demasiado los pacientes optarán por volver a casa y esto puede afectar a tu reputación.
- Tasa de producción variable: la tasa de producción ahora tiene en cuenta la reputación de tu hospital (tras una fecha fijada en el archivo de nivel.)
- El humo de la máquina ahora es visible cuando están muy desgastadas.
- Ahora es posible hacer clic derecho en el temporizador como en el juego original para mover a los pacientes afectados.
- Compatible con teclado numérico,

Figuras 7 - 8 - CorsixTH

Ahora se puede utilizar el teclado numérico para moverse por el mapa cuando el bloqueo numérico está desactivado, o para escribir números cuando el bloqueo numérico este activado, sin efectos secundarios.

- La carpeta MP3 ahora se puede desactivar desde dentro del juego. Anteriormente tras configurar una carpeta de mp3, sólo se podía eliminar editando el archivo de configuración a mano.
- La interfaz ahora utiliza el cursor adecuado para cambiar el tamaño de las habitaciones.
- Los contenedores ahora se puede colocar en los pasillos.

Para descargar e instalar Corsix TH, escribe el siguiente comando:

Figuras 9 - 10 - YQuake 2

```
$ apt-get install corsixth-odroid
```

YQuake 2

Yquake 2 es un remake del Quake 2 de código abierto que permite partidas multijugador. El juego se ejecuta a máxima velocidad a una resolución de 1080p y utiliza GLShim.

Para descargar e instalar YQuake 2, escribe el siguiente comando:

```
$ apt-get install yquake2-odroid
```

También hay disponible un paquete de alta resolución, que se muestra en las capturas de pantalla. ¡Feliz fragging!

COMPILAR ANDROID LOLLIPOP PARA EL ODROID-C2

USANDO LINUX MINT 18 O UBUNTU 16.04

por Jörg Wolff

Esta guía detalla cómo compilar Android 5.1.1 Lollipop para ODROID-C2 en un equipo que ejecuta Linux Mint 18 o Ubuntu 16.04. La guía aprovecha las instrucciones oficiales de Hardkernel para compilar Android. Antes de empezar, hay que configurar algunos paquetes de software y compilar dependencias. Android Lollipop necesita el kit de desarrollo de Java (JDK) Open-JDK7. Puesto que Ubuntu 16.04 y Linux Mint 18 utiliza OpenJDK8, OpenJDK7 se debe instalar desde las fuentes de Archivos de Paquetes Personales (PPA). Para instalar OpenJDK 7, introduce los siguientes comandos en una ventana de terminal:

```
$ sudo add-apt-repository \
  ppa:openjdk-r/ppa && sudo apt-
get update
$ sudo apt-get upgrade && \
  sudo apt-get dist-upgrade
$ sudo apt-get install adb fast-
boot openjdk-7-jdk
```

No es necesario eliminar la versión de OpenJDK 8 por defecto que viene preinstalada. De hecho, se recomienda dejarla instalada, ya que puede ser necesaria para otros programas como Eclipse o Android Studio. Ambas versiones pueden convivir juntas, pero antes de compilar Android, la versión de Java deben ser cambiada a OpenJDK7. Esto se puede hacer con estos comandos:

```
$ sudo update-alternatives \
 --config java
$ sudo update-alternatives \
 --config javac
$ sudo update-alternatives \
 --config javadoc
```

El resultado en consola de los anteriores comandos debería tener este aspecto:

```
  Auswahl      Pfad
Priorität Status
------------------------------
---------------------------
  0            /usr/lib/jvm/
java-8-oracle/jre/bin/java
1082      automatischer Modus
* 1            /usr/lib/jvm/
java-7-openjdk-amd64/jre/bin/java
1071      manueller Modus
  2            /usr/lib/jvm/
java-8-openjdk-amd64/jre/bin/java
1081      manueller Modus
  3            /usr/lib/jvm/
java-8-oracle/jre/bin/java
1082      manueller Modus
```

Para que la compilación de Android tenga éxito, en necesario instalar varios paquetes en el ordenador host (el ordenador que hace la compilación). Los paquetes se pueden instalar con el siguiente comando apt-get:

```
$ sudo apt-get install adb fast-
boot git\
 ccache automake lzop bison
gperf\
 build-essential zip curl zlib1g-
dev\
 zlib1g-dev:i386 g++-multilib
python-networkx\
 libxml2-utils bzip2 libbz2-dev
libbz2-1.0\
 libghc-bzlib-dev squashfs-tools
pngcrush\
 schedtool dpkg-dev liblz4-tool
make\
 optipng maven python-mako py-
thon3-mako\
 python python3 syslinux-utils\
 Google-android-build-tools-in-
staller
```

Estos son los preparativos para el ordenador host. En mi sistema, sufrí algunos errores de compilación. Si ha iniciado accidentalmente la compilación con una versión incorrecta del JDK, lo mejor es limpiar el repositorio git o volver a descargar la fuente y empezar de nuevo.

Puesto que la descarga puede tardar entre 1-2 días, deberías probar primero la opción de limpieza. Para limpiar la fuente, ejecuta el comando desde la carpeta superior del árbol de fuentes. Además, la carpeta "out" debe ser eliminada:

```
$ repo forall -c \
  'git reset --hard ; git clean
-fdx'
$ rm -rf out
```

Los siguientes pasos están bien documentados en la página wiki de Hardkernel en http://bit.ly/2chF4Tu. Si sigues estos pasos, no olvide instalar las siguientes herramientas, a menos que quieras recibir errores de compilación:

- gcc-linaro-arm-none-eabi-4.8-2014.04_linux
- gcc-linaro-aarch64-none-elf-4.9-2014.09_linux

Fallo de sincronización repo

Si estas gestionando fuentes antiguas, puede fallar la sincronización repo. Si eso sucede, es necesario forzar la sincronización:

```
$ repo sync --force-sync
```

Y después de la limpieza y la sincronización, no olvides cambiar a la rama maestra:

```
$ repo start s905_5.1.1_master
--all
```

También puede encontrar este error recurrente:

```
unsupported reloc 43
libnativehelper/JniInvocation.
cpp:165: error: unsupported reloc
43
```

Tras hacer muchas pruebas, encontré una solución en los foros de xda-developers en http://bit.ly/2cCtfrp:

```
$ ln -sf /usr/bin/ld.gold /home/
(your account name)\
/(build source repository)/pre-
builts/gcc/linux-x86/host/\
(glibc version)/x86_64-linux/bin/
ld
```

Ten en cuenta que es necesario comprobar los mensajes de error, con respecto a la versión utilizada x86_64-linux-glibc2. En mi sistema, dio resultado el siguiente comando, el cual crea un identificador ID de enlace simbólico:

```
$ ln -sf /usr/bin/ld.gold \
/home/joerg/odroid_c2/lollipop/\
prebuilts/gcc/linux-x86/host/\
x86_64-linux-glibc2.11-4.6/
x86_64-linux/bin/ld
```

¡Haz siempre una copia de seguridad primero! Otro error que puede darse es el siguiente:

```
public_api.txt:20: error 5
out/target/common/obj/PACKAGING/
public_api.txt:20: error 5: Added
public field android.Manifest.per-
mission.BACKUP
```

La solución pasa por escribir el siguiente comando:

```
$ make update-api
$ make -j4 selfinstall
```

Ahora puedes seguir los pasos de compilación de la página wiki de Hardkernel:

```
$ export ARCH=arm64
$ export CROSS_COMPILE=aarch64-
none-elf-
$ export PATH=/opt/toolchains/\
gcc-linaro-aarch64-none-
elf-4.9-2014.09_linux/\
bin:$PATH
$ export PATH=/opt/toolchains/\
gcc-linaro-arm-none-ea-
bi-4.8-2014.04_linux/\
bin:$PATH
$ export JAVA_HOME=/usr/lib/jvm/
java-1.7.0-openjdk-amd64
$ export PATH=$JAVA_HOME/
bin:$PATH
$ source build/envsetup.sh
$ lunch odroidc2-eng-32
$ make -j4 selfinstall
```

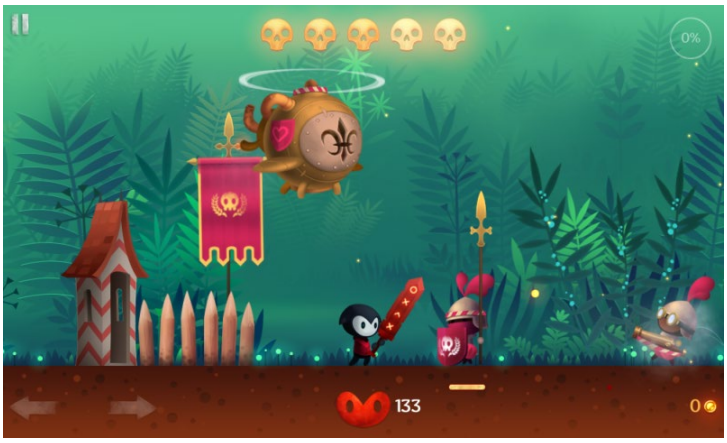
Si has instalado las herramientas en otra carpeta, como yo hago normalmente, necesitas proporcionar la ruta correcta a las herramientas. Estos pasos deben proporcionarte una versión de Android lista para ser instalada en un módulo eMMC o tarjeta SD.

JUEGOS ANDROID

REAPER, TALE OF A PALE SWORDSMAN

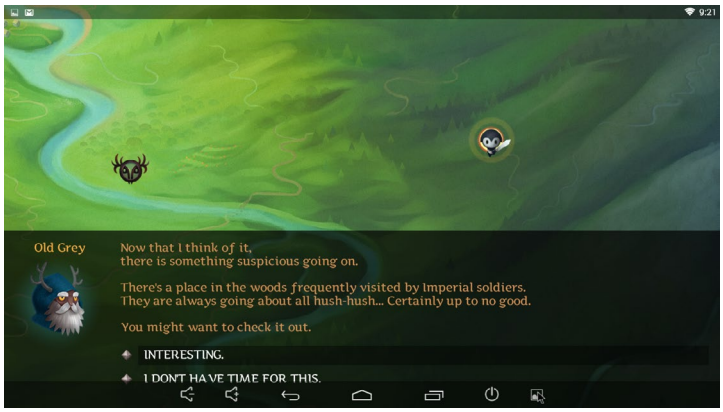
por @synportack24

Si estás buscando un divertido juego de acción de muy buen ver para Android, ¡Reaper lo es! Reaper: Tale of a Pale Swordsman es un videojuego de acción y rol disponible de forma gratuita en Google Play, http://bit.ly/1sIRjYd.



El juego tiene lugar en un territorio conocido como "wilderness ", donde tu personaje, un espadachín paliducho acaba de despertar. "Wilderness" es una tierra mágica llena tribus salvajes y guardias imperiales, con varios grupos en medio. El espadachín paliducho viene a ser un tipo de mercenario que va a las misiones y aventuras de los diferentes grupos del wilderness y su amigo que se llama Old Grey Beard. Más adelante en el juego, surge un nuevo enemigo que amenaza no sólo el espadachín paliducho, sino a toda la población de wilderness.

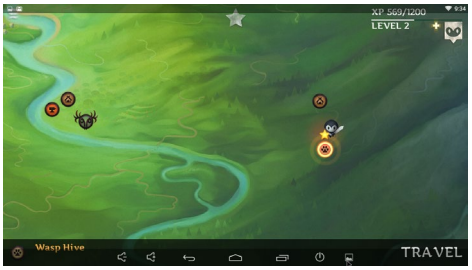
El jugador controla el espadachín y puede elegir el camino que tome éste. El menú del juego es un gran mapa con varios puntos de interés por los que el jugador puede mover el espadachín. Una vez en un punto, el jugador puede interactuar con diversas tribus y guardianes seleccionando las opciones de diálogo. Las opciones del diálogo ayudan a que el juego no sea tan rígido ya que el jugador pueda elegir: aceptar o rechazar una misión, exigir más dinero, atacar o defender a un grupo y muchas otras opciones.



A medida que el espadachín avanza en el juego, ganará más oro y más puntos de experiencia. Utilizando el oro, podrás comprar nuevas armas, armaduras y otros elementos que ayudarán al jugador a aumentar sus estadísticas. Con la experiencia, subirás de nivel a lo largo del juego. Cada vez que subas de nivel podrás elegir 3 habilidades. Afortunadamente, este juego no sigue la mentalidad "freemium" de muchos otros juegos, que poco a poco te obligan a pagar por actualizaciones para superar el juego. En cambio, la versión libre de Reaper está configurada como una demo, en la que es gratis jugar hasta que tu personaje alcance el nivel 10. El juego completo desbloqueado se puede comprar dentro del propio juego, después de lo cual no es necesario comprar nada más dentro del juego.

El combate constituye la mayor parte del juego, pasarás

mucho tiempo luchando contra una gran variedad de personajes en diferentes lugares. El juego proporciona una gran variedad de oponentes únicos a los que te enfrentarás. Sin embargo, la mejor forma de atacarles casi siempre es la misma, saltar en cualquier lugar con tropas y atacar, luego alejarte. Con esto en mente, hay muchas otras cosas que te atacarán igualmente, como los obstácu-



los de nivel y los proyectiles enemigos que tendrás que evitar.

En general, Reaper es muy simple para precisamente detenerte y jugar un par de minutos o sentarse un rato y disfrutar. Todo el juego está muy pulido, las ilustraciones y los gráficos son espectaculares. Todos los niveles y mapas son muy atractivos e interesantes. Todo funciona perfectamente en mi ODROID-XU4, aún cuando lo tengo en underclocked con refrigeración pasiva. Además, se puede jugar al juego por completo usando un mando Xbox 360, o un teclado y un ratón. Descubrí que con el mando es mucho más fácil jugar, permite mostrar todos los cuadros de diálogo de inicio con los botones adecuados del mando Xbox. Recomiendo a cualquier jugador que esté buscando un juego de acción divertido que eche un vistazo a Reaper, que seguro disfrutará de él.

GENTOO PARA EL ODROID-C2

CREA TU PROPIA INSTALACION A MEDIDA

por @revOlt

Gentoo Linux es un sistema operativo gratuito basado en FreeBSD que puede ser optimizado y personalizado automáticamente para casi cualquier aplicación o necesidad. He logrado arrancar y ejecutar en el ODROID-C2 la fase 3 experimental de Gentoo para ARM64.

Para empezar, arranca el C2 con cualquier distribución de Linux, asegúrate de hacer backup de los archivos importantes por si algo sale mal. Yo use Ubuntu, puesto que tiene soporte oficial por parte de Hardkernel.

Preparación

Prepara el sistema de ficheros root de Gentoo y descarga la fase 3 experimental de Gentoo para ARM64 desde gentoo.org a un directorio local en Ubuntu. En este ejemplo usamos /opt/gentoo como directorio local en Ubuntu y la última fase 3 con fecha 24/03/2016, que puedes cambiarla si lo deseas.

```
$ mkdir /opt/gentoo && cd /opt/gentoo
$ wget http://distfiles.gentoo.org/experimental/\
arm64/stage3-arm64-20160324.tar.bz2
$ bzip2 -d stage3*
$ tar xvf stage3*
```

A continuación, copia /lib/modules y /lib/firmware desde Ubuntu a Gentoo. La fase 3 de Gentoo tiene /lib y /lib64. También copiaremos los directorios modules y firmware en /lib de Gentoo por si acaso:

```
$ cp -afv /lib/modules /opt/gentoo/lib64/
$ cp -afv /lib/firmware /opt/gentoo/lib64/
$ cp -afv /lib/modules /opt/gentoo/lib/
$ cp -afv /lib/firmware /opt/gentoo/lib/
```

Después, asegúrate de que /opt/gentoo/etc/portage/make.conf está configurado correctamente de acuerdo con el manual de Gentoo disponible en http://bit.ly/1swpkQq. Para el ODROID-C2, he dejado el CFLAG intacto por ahora (CFLAGS="-O2 -pipe"), porque añadir -march y -mtune parece dar lugar a errores de compilación en algunos paquetes. Intenté ajustar MAKEOPTS="-J5", pero me encontré con poca memoria y el proceso se detuvo. Sin embargo, usar en su lugar "-j3" resolvió esta cuestión. Simplemente tienes que utilizar la configuración conservative en make.conf por ahora, ya que siempre podrás afinarla más tarde. Los siguientes comandos cambiarán root (chroot) en Gentoo:

```
$ cp -L /etc/resolv.conf /opt/gentoo/etc/
$ mount -t proc proc /opt/gentoo/
```

```
proc
$ mount --rbind /sys /opt/gentoo/
sys
$ mount --make-rslave /opt/gen-
too/sys
$ mount --rbind /dev /opt/gentoo/
dev
$ mount --make-rslave /opt/gen-
too/dev
$ chroot /opt/gentoo /bin/bash
$ source /etc/profile
$ export PS1="(chroot) $PS1"
```

Configurar Gentoo

```
$ passwd # set root password
$ useradd -m -G wheel -s /bin/
bash yourname #add user youname
$ passwd yourname #set user your-
name's password
$ emerge --webrsync
$ emerge --sync
$ echo "UTC" > /etc/timezone
$ emerge --config sys-libs/time-
zone-data
$ nano /etc/locale.gen # remove
the relevant comment from the file
$ locale-gen

$ eselect locale list
$ eselect locale set X
#where X is the desired locale
$ env-update && source /etc/pro-
file && \
export PS1="(chroot) $PS1"
```

Descarga el script c2_init.sh de http://bit.ly/2cWAjQP y colocarlo como c2_init.start en el directorio /etc/local.

```
$ chmod +x /etc/local.d/c2_init.
start
$ rc-update add local default
```

Necesitamos especificar el dispositivo del sistema de archivos de Gentoo en /etc/fstab. En este ejemplo, que utilizo la partición 3 de mi eMMC como sistema de ficheros root de Gentoo, usare el siguiente comando para actualizar el archivo para que coincida con lo siguiente:

```
$ nano /etc/fstab

# <fs>                  <mountpoint>
<type>                  <opts>
                        <dump/pass>

/dev/mmcblk0p3          /
ext4          errors=remount-
ro,noatime,nodiratime0 1
tmpfs           /
tmp           tmpfs
nodev,nosuid,mode=1777
      0 0
/dev/mmcblk0p1          /boot
vfat          defaults,rw,owner,fl
ush,umask=000         0 0
```

Si vas a utilizar una tarjeta microSD, el dispositivo además será identificado como mmcblk0 tras el arranque, así que el anterior comando debería funcionar, pero hay que cambiar el número de partición en consecuencia. Por ejemplo, si va a iniciar Gentoo desde una tarjeta microSD con Gentoo en la partición 2, deberías especificar /dev/mmcblk0p2 como punto de montaje / en tu fstab (es decir, seguir utilizando mmcblk0 pero cambiando p3 por p2).

Configurar la red

Para configurar los drivers de red, escribe el siguiente comando:

```
$ emerge --noreplace net-misc/
netifrc
```

Para usar DHCP en las interfaces de red eth0 y wlan0 (si tienes un dispositivo USB wifi por ejemplo), edita /etc/conf.d/net de la siguiente forma y guardarlo:

```
config_eth0="dhcp"
config_wlan0="dhcp"
```

Luego, escribe los comandos:

```
$ cd /etc/init.d
$ ln -s net.lo net.eth0
$ rc-update add net.eth0 default
$ ln -s net.lo net.wlan0
$ rc-update add net.wlan0 default
$ rc-update add sshd default
$ emerge net-misc/dhcpcd sys-ker-
nel/linux-firmware ntp wpa_suppli-
cant # ntp is needed as C2 has no
persistent clock or else Gentoo
would boot to 1970
$ rc-update add ntpd default
$ wpa_passphrase MYSSID myssid-
passphrase > \
  /etc/wpa_supplicant/wpa_sup-
plicant.conf # insert your MYSSID
and myssidpassphrase accordingly
```

Ten en cuenta que para desenmascarar algunos paquetes, es posible que quieras utilizar el comando "emerge --autounmask-write" y luego ejecutar "dispatch-conf" para facilitar su uso. Después, salte de chroot y desmonta los seudo sistemas de archivos de chroot.

```
$ exit
$ umount -l /opt/gentoo/dev{/
shm,/pts,}
$ umount /opt/gentoo{/boot,/sys,/
proc,}
```

Crear el sistema de archivos

A continuación, tenemos que crear el sistema de archivos en la partición de Gentoo y luego, copiar los archivos de Gentoo preparados en los anteriores pasos a este sistema de ficheros. En este ejemplo, Yo usaré la partición 3 de mi tarjeta eMMC para Gentoo:

```
$ mkfs.ext4 /dev/mmcblk0p3
$ mkdir /mnt/gentoo
$ mount /dev/mmcblk0p3 /mnt/gen-
too
```

Después, copia los archivos de /opt/gentoo a /mnt/gentoo:

```
$ cp -afv /opt/gentoo/* /mnt/gen-
too/
```

En este ejemplo, puesto que boot.ini

de la partición de arranque de Ubuntu está apuntando a la partición de Ubuntu en /dev/mmcblk0p2 (montada como / media/boot/ en Ubuntu), hay que editarlo para que apunte a la partición root de Gentoo:

```
$ nano /media/boot/boot.ini
```

Cambia la variable root= por root=/dev/mmcblk0p3 en esta línea.

```
setenv bootargs "root=/dev/
mmcblk0p3 rootwait ro ${condev}
no_console_suspend hdmimode=${m}
m_bpp=${m_bpp} vout=${vout}
fsck.repair=yes net.ifnames=0
elevator=noop
disablehpd=${hpd}"
```

Por favor asegúrate de que la línea anterior (es decir, la que comienza con las palabras setenv bootargs seguidas de las variables encerradas por comillas) se encuentra en una única línea.

Cuando se edita en nano. El formato de la página web probablemente puede hacer que la línea se muestre en varias líneas acccidentalmente.

Por último, reinicia el dispositivo y el C2 debería arrancar con Gentoo, con la red eth0 y wlan0 funcionando. El método anterior también debería funcionar si instalas Gentoo y lo arrancas desde una tarjeta microSD en lugar del eMMC, siempre que hagas las modificaciones oportunas. Por ejemplo, si estás instalando Gentoo en una tarjeta microSD con la partición 1 como partición de arranque y la 2 como sistema de ficheros root de Gentoo, tienes que especificar mmcblk0p2 como tu / en el sistema de archivos de Gentoo en /etc/fstab antes de reiniciar con Gentoo.

Para preparar la microSD (/dev/sda1 por ejemplo), deberías usar sd_fusing.sh /dev/sda, asegurarte de que los archivos de arranque (boot.ini, Imagen, meson64_odroidc2.dtb, uInitrd) de / media/boot/ en Ubuntu son copiados a /dev/sda1 antes de reiniciar. Es posible que quieras mantener tu partición de Ubuntu para que pueda arrancar de nuevo en ella y actualizar el kernel y / lib/modules /lib/firmware más adelante. Para ello, basta con editar boot.ini en / boot de Gentoo y cambiar la variable root= de nuevo a /dev/mmcblk0p2 (en mi ejemplo) y reiniciar el sistema. Una vez que el kernel y /lib sean actualizados en Ubuntu, puedes copiarlos a la partición de Gentoo y reiniciar de nuevo en Gentoo editando /media/boot/boot.ini como he descrito anteriormente. Sería buena idea hacer un script para esto tanto en Gentoo como en Ubuntu, y así reiniciar periódicamente para actualizar Ubuntu y copiar el directorio / lib/firmware y / lib/modules en Gentoo, de modo que todo se hiciera automáticamente en segundo plano por la noche mientras duermes, por ejemplo.

En Gentoo, debido a que empezamos con la fase 3, una vez que hayas adaptado tus variables make.conf, puedes escribir el siguiente comando para volver a compilarlo todo:

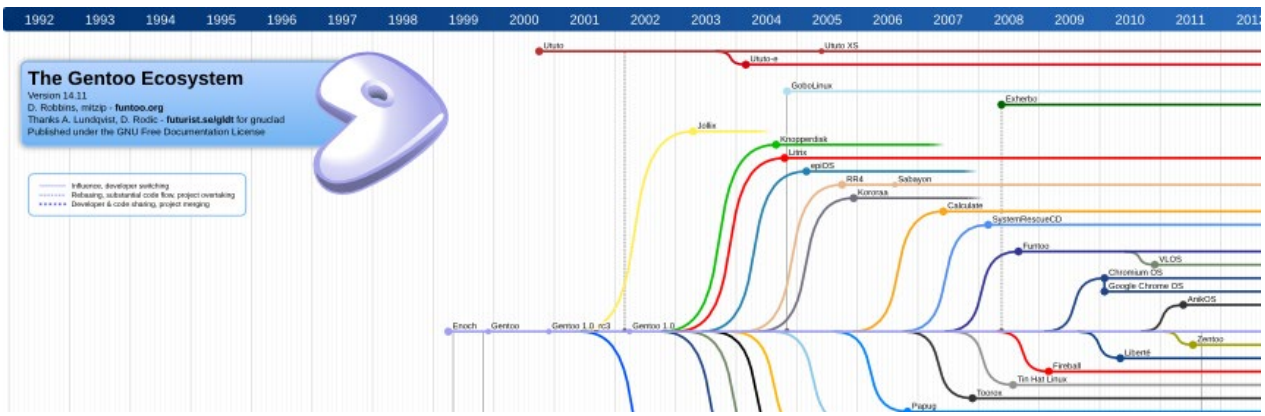
```
$ emerge -e world
```

Ten en cuenta que algunos paquetes pueden requerir fuentes kernel en /usr/src, de modo que descargarlas o git clone la rama necesaria de http://bit.ly/2cISS9s en un directorio dentro de / usr/src llamado, por ejemplo, linux-c2-3.14 y a continuación crea un enlace simbólico (o eselect) al directorio /usr/src/linux ya que emerge lo espera allí por defecto.

X Window se compila y funciona muy bien sobre C2 en Gentoo. Si quieres instalarlo, puede utilizar el comando:

```
$ USE="-llvm" emerge x11-base/
xorg-server
$ emerge openbox xterm # or
choose your favorite
$ nano /etc/X11/xinit/xinitrc #
remove the reference to twm and
replace it with exec openbox or
your favorite
$ startx
```

También es posible que estés pensando en soluciones de refrigeración activa para el C2 si compilas software a menudo. Analizando mis archivos log, con el disipador de calor de la CPU y sin refrigeración activa, el C2 llega a calentarse un poco para mi gusto, cuando ejecuto pesados trabajos con emerge, llegando a los 70 grados centígrados. Puedes hacer un seguimiento de la temperatura durante los trabajos de compilación buscando la temperatura en /sys y analizando el archivo correspondiente. Puede ser una buena idea conectar un ventilador cuando compiles o al menos mantenerlo a una distancia segura del televisor. Para preguntas, comentarios y sugerencias, visita el post original del foro en http://bit.ly/2ctWugX o la página principal de Gentoo en http://www.gentoo.org.



CONOCIENDO UN ODROIDIAN

SEBASTIEN CHEVALIER (@PTITSEB), DESARROLLADOR DE GLSHIM

editado por Rob Roy

Nuestro desarrollador, Sebastien en su oficina

Por favor, Háblanos un poco sobre ti

Mi nombre es Sebastien Chevalier. Tengo 43 años, soy francés, estoy casado y tengo dos hijos.

¿Cómo fueron tus inicios con los ordenadores?

Cuando tenía de 10 años, mi padre llego a casa con un ordenador. Era un equipo francés poco conocido denominado Hector, que se conecta a la TV y venía con 3 cassettes. Puesto que era de la década de los 80, no tenía discos y ejecutaba Space Invaders, un software de préstamos hipotecarios inútil y BASIC. Tras pasar muchas horas jugando a Space Invaders, empecé a mirar el compilador de BASIC. También traía dos manuales impresos sobre Basic, así que aprendí el lenguaje para crear más juegos.

¿Qué te atrajo de la plataforma ODROID?

Mi primer contacto con dispositivos ODROID fue hace un par de años. Después de exportar FreeSpace y FreeSpace 2 a OpenPandora, @meveric se puso en contacto conmigo para que le ayudara a exportarlo a la plataforma ODROID. Estaba contento de que mi versión pudiera ser utilizada en otro dispositivo para el que no estaba pensado inicialmente. Más tarde, mi colaboración con @meveric se intensificó y empecé a añadir más y más código ODROID a mis versiones.

¿Cómo utilizas tus ODROIDs?

Mi ODROID está conectado a mi televisor principal. Lo utilizo para navegar por internet, jugar a algunos juegos, probar nuevas versiones de software y como reproductor multimedia.

Su continuo trabajo sobre el paquete GLShim es muy conocido, especialmente entre los aficionados a los juegos. ¿Qué te motivó a trabajar en el proyecto?

Cuando empecé a exportar juegos, llegue a exportar a mano OpenGL a GLES. Desarrolle algunos juegos y actualice algunos motores específicos, como FreeSpace, Jedi Knights II y III. Después continúe el proyecto GLShim creado por @lunixbochs (http://bit.ly/2d4PHek). Al principio, no estaba muy convencido ya que mis versiones exportadas eran buenas, a veces incluso mejores que un simple GLShim exportado. Sin embargo, algunos juegos eran muy complicados de exportar, como Armagetron Ad-

El primer ordenador de Sebastien, el Hector

Sebastien visitó el verano pasado Como Lake, en Italia

vanced y Scorched Earth 3D, de modo que intente el método GLShim. Fue cuando entonces me separé del proyecto GitHub de @linuxboch con el fin de tener mis propias fuentes y trabajar con un enfoque "puerto a puerto" cogiendo un juego o una pieza de software e intentar hacer que funcionara. Algunos juegos y aplicaciones utilizan funciones que son difíciles de implementar, fallan o no se convierten correctamente, así que analizaba lo que ocurría y luego, añadía una función para solucionarlo. Otras veces me daba por vencido, pasaba a otro software, y volvía más tarde después de que GLShim evolucionara.

Actualmente, GLShim está funcionando bastante bien, sobre todo con todas las funciones implementadas de OpenGL 1.5, y tambien muchas funciones GLX. Ahora puedo ejecutar juegos como Minecraft, idTec 1, 2 ó 3, juegos de Serious Engine y software como Blender.

¿Cuál es tu ODROID favorito y por qué?

Actualmente sólo poseo un ODROID-XU4, que es una buena máquina. Es muy potente tanto por su CPU como por su GPU, pero la refrigeración activa hace que no sea idóneo para usarlo como un HTPC. Posiblemente probaré el C2 con refrigeración pasiva, que parece ser también bastante potente, sobre todo ahora que el driver 3D está disponible.

¿Qué innovaciones le gustaría ver en futuros productos de Hardkernel?

Me gustaría ver algunos kits basados en los diseños de bricolaje que han ido apareciendo en números anteriores de ODROID Magazine

¿Qué aficiones e intereses tienes aparte de los ordenadores?

No tengo muchas aficiones. Suelo jugar a los juegos, pero esto sigue estando relacionado con los ordenadores. Me gusta viajar, especialmente en vacaciones con mi familia (ver la imagen del hermoso Como Lake en Italia donde he estado este verano). Paso la mayor parte de mi tiempo libre por las noches y los fines de semana con el ordenador. También trabajo de programador en mi trabajo, así que supongo que suelo pasar fácilmente 12 horas al día delante de una pantalla de ordenador.

¿Qué consejo le daría a alguien que desee aprender más sobre programación?

Aprender a programar lleva tiempo. Mi mejor consejo es que te enfrentes a proyectos reales, pero no excesivamente ambiciosos. Para aprender, puede empezar con juegos simples. Por ejemplo, si nunca has programado antes, coge un juego como "Find a number", en el que el ordenador escoge un número entre 1 y 999, que tienes que adivinar y el ordenador te va diciendo si está por encima o por debajo. Luego, inténtalo con un simple juego Tres en Raya basado en texto, después con un juego básico Break Out, que debería ser tu primer paso a nivel gráfico. El lenguaje que utilices no es lo más importante, de modo que puedes intentarlo con BASIC, Pascal, C, C ++, Python, o Lua. Lo importante es alcanzar el objetivo, así que disfruta aprendiendo haciendo cosas y no te desanimes. Date tiempo para convertirte en un buen programador y no debrías compararte al principio con otras personas.

Sebastien disfrutando de una comida