SWI Prolog • ODROID-VU • Juegos Linux • Adafruit • Cryptomineria

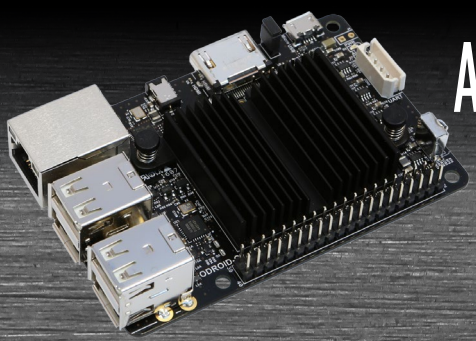
ODROID Magazine

Año Cuatro
Num. #44
Ago. 2017

Experimenta el futuro con tu ODROID:

Inteligencia Artificial

Migrando Desde Ubuntu Mate a Lubuntu



Android en el C2: Una Guía para Principiantes

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

La inteligencia artificial es una de los límites informáticos más excitantes del siglo XXI. Tecnologías como Amazon Alexa, Google Assistant y Microsoft Cortana son asistentes personales compactos, asequibles y útiles, pero todos usan código cerrado de terceros, lo cual plantea problemas de privacidad. Una solución que nos asegura que el usuario tenga control total sobre la información que se comparte y se almacena es usar aplicaciones de código abierto como Mycroft y Prolog. La compilación del código y su instalación en un ODROID es una forma fácil y barata de desarrollar tu propio asistente personal para satisfacer tus necesidades, mientras que también protege tu privacidad.

Otra aplicación emergente de tecnología avanzada es la mineria de criptomonedas. Los ingenieros de Hardkernel construyeron un clúster de demostración ODROID-XU4 para demostrar la estabilidad del Kernel 4.9 y calcular la viabilidad financiera de la minería a largo plazo. Tobias discute si es o no un fanático de Nintendo, Jörg nos muestra cómo controlar la luz de fondo de la pantalla en Android, Miltiadis nos ayuda a migrar desde UbuntuMATE a Lubuntu, Dennis nos muestra un proyecto usando la pantalla Adafruit OLED y presentamos una guía para empezar a usar Android en el ODROID-C2.

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/1yplmXs.
Únete a la comunidad ODROID con miembros en más de 135 países en http://forum.odroid.com/ y explora las nuevas tecnologías que te ofrece Hardkernel en http://www.hardkernel.com/

HARDKERNEL

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

Todavía está cuidando al perro, y sí, consiguió un Playstation 4. En cuanto a la versión en PDF de ODROID Magazine, ha sido una gran travesía de 44 números. ¡Adelante con la nueva frontera de la versión interactiva HTML!

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.

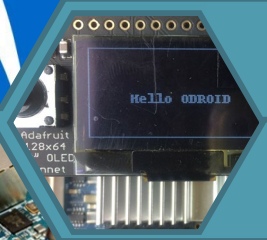
Venkat Bommakanti, Editor Adjunto

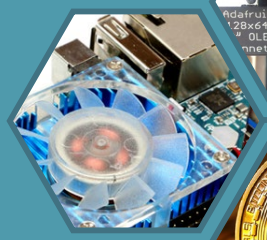
Soy un apasionado de los ordenadores desde la bahía de San Francisco en California. Procuro incorporar muchos de mis intereses en proyectos con ordenadores de placa reducida, tales como pequeños modificaciones de hardware, carpintería, reutilización de materiales, desarrollo de software y creación de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

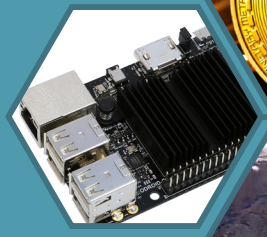
Andrea Cole, Editor Adjunto

Vivo en Ontario, Canadá, y aunque soy relativamente nueva en esto de los ODROID, tengo diez años de experiencia en publicación online. Actualmente tengo ordenadores de placa reducida, programados por mi pareja, que funcionan como centro multimedia, un emulador de juegos y reproductor de mi biblioteca de música. En mi tiempo libre, soy un artista, escritora y músico. Planeo incorporar algo del conocimiento que adquire en algunas creaciones multimedia interactivas.

INDICE







MYCROFT

INTELIGENCIA ARTIFICIAL DE CODIGO ABIERTO

por Adrian Popa

Desde que KITT, el coche fantástico con voz hizo acto de presencia en la década de los 80, todo el mundo ha soñado con el día en el que tuviera su propio asistente personal con voz. La buena noticia es que ese día casi a llegado. La mala noticia viene dada por el alto coste asociado a la falta de privacidad y la explotación de datos realizada por las grandes compañías. Sería bueno tener un asistente controlado por voz que no compartiera sus datos en la nube. Por desgracia, posiblemtne aún nos encontremos a 10 años de ese momento, aunque estamos avanzando en la dirección correcta. En este artículo, vamos a instalar una plataforma de inteligencia artificial de código abierto que puedes ejecutar en tu ODROID llamada Mycroft (`https://mycroft.ai/`).

Mycroft se compone de varios elementos interconectados que gestionan la captación de la voz, el reconocimiento de palabras clave (hecho a nivel local), la traducción de voz a texto (actualmente hecho a través de un servicio de Google) un motor de habilidades y la conversión de texto a voz (hecho localmente a través de mimic). Mycroft se comunica con varios entes de Internet, utilizando su propia nube para almacenar y recuperar configuraciones y claves API para diversos servicios e incorporar los servicios de voz a texto de Google. Además de esto, el motor de habilidades puede comunicarse con servicios online y recuperar datos para el usuario final.

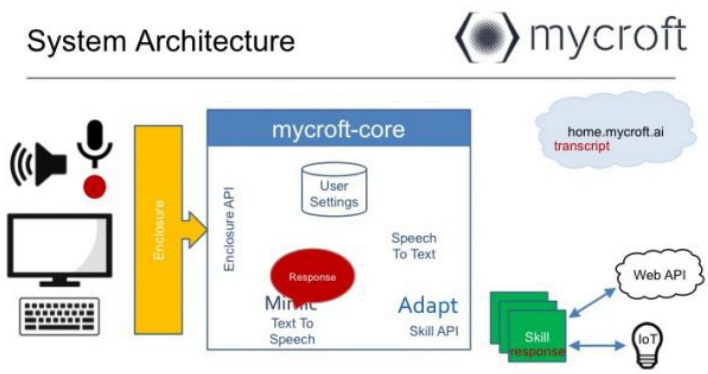


Figura I - Arquitectura de Mycroft

De partida, la gente comparará Mycroft con otros asistentes personales como Amazon Echo, Alexa y Google Assistant. Sin embargo, son muy diferentes. Mycroft está diseñado para ser abierto y no intrusivo, mientras que el resto de asistentes están en constante comunicación con los datos de tu perfil y responden a las solicitudes involucrando tus datos personales, como son tu lista de contactos o tu calendario personal. Por defecto, Mycroft no tiene acceso a esa información y es utilizado principalmente para consultar fuentes de información pública, como el parte meteorológico, Wikipedia y Wolfram Alpha (https://www.wolframalpha.com/). Algunos pueden pensar que esto limita su alcance, pero si valoras tu privacidad, entonces sabrás que no es así. El usuario final puede habilitar una gran cantidad de funciones avanzadas activando ciertas habilidades, como la integración con cuentas de Google. En términos de complejidad IA, Mycroft es más simple que el resto de asistentes de voz, que se ejecutan integramente en la nube y se benefician de complejas y recurrentes redes neuronales.

Uno de los principales problemas de la privacidad es el hecho de tener un dispositivo visible u oculto que esté escuchando constantemente la conversación. Analizar qué datos filtra tu asistente personal sobre ti debería ser algo prioritario para todos (`http://bit.ly/2gY9qKG`). La ventaja del código abierto de Mycroft es que, las consultas de la red son registradas y fácilmente accesibles por el usuario medio, con el objeto de poder ver lo que realmente hace el dispositivo tras la interfaz.

Mycroft podría mantener un registro de tus conversaciones en su nube, pero los registros log muestran que no guarda una copia de las conversaciones. Los datos de voz se envían a los servidores de Google para convertirlos en texto, porque las trancripción de voz a texto sigue siendo un complejo problema que por lo general requiere el uso de redes neuronales (`http://bit.ly/2eGLJJz`). En teoría, Google aún podría guardar un registro de tus conversaciones y vincularlas a tu dirección IP, comprometiendo así tu privacidad. El equipo de Mycroft ha empezado a trabajar en un motor de voz/texto abierto con el fin de resolver

este problema, pero la realidad es que pasará mucho tiempo antes de que podemos ejecutarlo en local (https://openstt.org/).

De modo que, si quieres un asistente personal que se active por voz y que esté familiarizado con la seguridad y la privacidad, veamos cómo podemos conseguir que Mycroft se ejecute sobre un ODROID.

Instalación

Si tiene un ODROID-U3, puede omitir esta guía e instalar una imagen Debian habilitada para Mycroft creada por el usuario del foro @nold, disponible en http://bit.ly/2tT2Hq1. Sin embargo, si no quieres dedicar un ODROID a una única tarea, puede instalar Mycroft en un ODROID-C2 de uso general que ejecute la imagen de Ubuntu 16.04 proporcionada por Hardkernel. Puede seguir los mismos pasos para realizar la instalación en un modelo de ODROID diferente, como el C1 o el XU4.

Lo primero que vas a necesitar, independientemente de tu modelo ODROID, es memoria de intercambio. El proceso de instalación necesita compilar el componente de texto a voz llamado Mimic, y esto ocupa mucha memoria. También debe hacer un reinicio limpio para liberar cualquier memoria utilizada en el caso de que tu ODROID haya estado encendido durante mucho tiempo.

Para crear un archivo de intercambio temporal de 4 GB en tu SD/EMMC, ejecuta los siguientes comandos:

```
$ sudo dd if=/dev/zero of=/swapfile bs=1M count=4096
$ sudo mkswap /swapfile
$ sudo swapon /swapfile
```

Puedes comprobar que se ha activado la memoria de intercambio con el siguiente comando:

```
$ free -m | grep Swap
```

Para realizar la instalación real, puedes recurrir a la documentación de https://docs.mycroft.ai/. Nosotros vamos a optar por el método de instalación git clone (http://bit.ly/2eGAp05). Mycroft se instalará dentro de un entorno virtual python en tu directorio de inicio, normalmente /home/odroid. Además, se ejecutará como un usuario normal. Para la instalación y compilación, necesitará alrededor de 2,5 GB de espacio libre.

```
$ cd ~
$ sudo apt-get install git
$ sudo git clone https://github.com/MycroftAI/my-
croft-core.git
$ cd mycroft-core/
$ sudo ./build_host_setup_debian.sh
$ sudo chown -R `whoami`:`whoami` .
$ ./dev_setup.sh
```

Dev_setup.sh creará el entorno virtual, configurará los módulos Python necesarios y luego compilará Mimic. Una vez compilado Mimic, que tardará unas tres horas y media utilizando un ODROID-C2 a 1,75 GHz, es posible que aparezca este error mientras se compila pygtk:

```
configure: error: cannot guess build type; you must
specify one
make: *** No targets specified and no makefile found.
Stop.
make: *** No rule to make target 'install'.  Stop.
```

Este es un error sin importancia y se puede ignorar sin peligro alguno. Indica que hay un componente de integración del escritorio que no se ha podido compilar, el cual no vamos a utilizar. En el futuro, si quieres actualizar Mycroft, puede volver a realizar los pasos de instalación (especialmente el paso de git clone), pero ejecuta dev_setup.sh -sm para omitir la compilación de Mimic.

Si la compilación ha ido bien, podemos hacer algo de limpieza y liberar un poco de espacio en disco. Los siguientes comandos desactivan la memoria de intercambio y recuperan 4 GB de espacio en disco:

```
$ sudo swapoff /swapfile
$ sudo rm -f /swapfile
```

También podemos liberar aproximadamente unos 500 MB de archivos temporales de mimic ejecutando los comandos:

```
$ cd mimic
$ make clean
```

Desafortunadamente, Mimic se instala en sus directorios fuente, de modo que es difícil separarlo y limpiarlo aún más.

Arranque

Mycroft no viene con una unidad de inicio systemd, pero podemos crear una en torno a su script de inicio. El script de inicio de Mycroft se encuentra en ~/mycroft-core/mycroft.sh. Éste acepta parámetros de arranque y parada, y permite gestionar 4 instancias de pantalla, cada una de las cuales ejecuta un componente clave de Mycroft. Para crear una unidad de inicio systemd, ejecuta los siguientes comandos, asegurándote de modificar el nombre de usuario y las rutas de acceso si no estás usando el usuario “odroid”:

```
$ sudo vi /etc/systemd/system/mycroft.service
[Unit]
```

```
Description=Mycroft personal AI
After=pulseaudio.service

[Service]
User=odroid
WorkingDirectory=/home/odroid/mycroft-core
ExecStart=/home/odroid/mycroft-core/mycroft.sh start
ExecStop=/home/odroid/mycroft-core/mycroft.sh stop
Type=forking
Restart=always
RestartSec=3

[Install]
WantedBy=multi-user.target
```

Para iniciar Mycroft en cada arranque e iniciarlo manualmente ahora, escribe los siguientes comandos:

```
$ sudo systemctl enable mycroft
$ sudo systemctl start mycroft
```

Si la configuración de sonido por defecto funciona, deberías ver a Mycroft reclamando que no está emparejado con su nube y ofrecer un código de emparejamiento. Sin embargo, supongamos que todavía no tienes sonido, o que quizás no entiendas lo que Mycroft te está indicando. Tendremos que acostumbrarnos a los procesos del back-end.

Back-end y depuración

Cuando Mycroft se inicia, crea 4 instancias de pantalla, donde ejecuta procesos de voz, habilidades, interfaz de línea de comandos y servicios. Puede listar y conectarte a estas instancias con el siguiente comando:

```
$ screen -list
There are screens on:
      5705.mycroft-cli       (07/13/17 13:50:28)
(Detached)
      5690.mycroft-voice     (07/13/17 13:50:28)
(Detached)
      5675.mycroft-skills    (07/13/17 13:50:28)
(Detached)
      5640.mycroft-service   (07/13/17 13:50:28)
(Detached)
```

Si te conectas a la pantalla mycroft-voice, podrá ver si hay algún error con la entrada o salida de audio (por ejemplo - no hay ningún dispositivo de salida de audio por defecto). Para conectarte e interactuar con una sesión de pantalla, puede ejecutar el siguiente comando:

```
$ screen -r 5690.mycroft-voice
```

Ten en cuenta que el número es el proceso PID y cambiará al reiniciar Mycroft. Deberías identificarlo con screen -list. Para desconectar una sesión de pantalla sin finalizar el proceso de pantalla, utilice la combinación de teclas CTRL + A D. Consulta el manual de `http://bit.ly/2tsC4ZF` para conocer más combinaciones de teclas.

Los archivos log de texto habituales de estos servicios también se guardan en ~/mycroft-core/scripts/logs/ y se pueden consultar con herramientas como “grep” o “tail -f”.

Mycroft también cuenta con un modo de depuración, con el cual sólo se inicia el motor, las habilidades y los componentes CLI, de modo que puedes corregir errores y depurar problemas. Para iniciar Mycroft en modo depuración, escribe:

```
$ ~/mycroft-core/mycroft.sh start -d
```

Vista de depuración

Configuración

El archivo de configuración principal de Mycroft se encuentra en ~/mycroftcore/mycroft/configuration/mycroft.conf. No debes editar este archivo, sino hacer una copia de éste en ~/.mycroft/mycroft.conf, para evitar que la configuración queda anulada al actualizar Mycroft:

```
$ cp ~/mycroft-core/mycroft/configuration/mycroft.conf
~/.mycroft/
```

El archivo de configuración contiene opciones generales, como el idioma (yo probé el inglés), las unidades, la ubicación, así como parámetros específicos para las habilidades predeterminadas. Es posible que estos pasen a diferentes archivos de configuración en el futuro. Llegados a este punto, puedes editar el archivo ~ /.mycroft/mycroft.conf y definir tu propia configuración, incluida la ubicación, para que las respuestas sean las adecuadas. Por ejemplo, si dejas la ubicación por defecto Lawrence, Kansas, cuando preguntes por la hora o el tiempo, obtendrás la hora y el tiempo de Lawrence, Kansas. Algunos de estos parámetros también se pueden configurar desde la nube de Mycroft una vez emparejado el dispositivo.

Audio

Si estás ejecutando una imagen de escritorio con salida de sonido HDMI y un micrófono, el sonido funcionará bien sin tener que hacer nada. Sin embargo, si está ejecutando una imagen de servidor sin un entorno de escritorio o si tiene varios micrófonos/salidas de sonido, puedes configurar Mycroft manualmente para usar un dispositivo de sonido concreto.

Aunque Mycroft soporta tanto ALSA como PulseAudio, vamos a usar PulseAudio como back-end de sonido, ya que es más flexible. Por ejemplo, para múltiples procesos es más fácil utilizar el micrófono al mismo tiempo.

Por lo general, PulseAudio se ejecuta como un proceso de usuario y se activa cuando inicias sesión en el entorno de escritorio. Esto no es compatible con tener Mycroft como servicio en el arranque, ya que no habría PulseAudio para conectarse. Tenemos que ejecutar PulseAudio como un servicio del sistema. Tienes más detalles sobre este tema en `http://bit.ly/2vzTOnj`.

```
$ sudo apt-get install pulseaudio
$ sudo vi /etc/systemd/system/pulseaudio.service
[Unit]
Description=PulseAudio Daemon

[Install]
WantedBy=multi-user.target

[Service]
Type=simple
PrivateTmp=true
ExecStart=/usr/bin/pulseaudio --system --real time
--disallow-exit --no-cpu-limit
```

Para hacer que los procesos del usuario se comuniquen con PulseAudio, necesitamos editar la configuración ubicada en /etc/pulse/system.pa. Necesitas añadir las siguientes líneas a la configuración:

```
#allow localhost connections
load-module module-native-protocol-tcp auth-ip-
acl=127.0.0.1
```

Antes de ejecutar PulseAudio como un servicio, necesitamos identificar el micrófono (entrada) y los altavoces (receptor) que queremos utilizar. Para ello, tendremos que ejecutar PulseAudio como usuario:

```
$ pulseaudio -D
```

A continuación, obtendremos una lista de dispositivos de salida (receptores) que Mycroft puede utilizar. La idea es que, si tu ODROID tiene conectados el HDMI y una tarjeta de sonido USB, Mycroft pueda utilizar la tarjeta de sonido como salida, para que puedas escucharlo incluso si el TV está apagado.

```
$ pacmd list-sinks | egrep "index|name:"
   index: 0
        name: <alsa_output.usb-0d8c_C-Media_USB_Head-
phone_Set-00.analog-stereo>
  * index: 1
        name: <alsa_output.platform-odroid_hdmi.
analog-stereo>
```

Si te fijas bien, puedes ver que la salida HDMI es la salida de sonido por defecto. Queremos decirle a Mycroft que use la tarjeta 0 (o mejor, podemos indexarla por su nombre) para el sonido. Para ello, edita ~/.mycroft/mycroft.conf y cambia la siguiente línea:

```
play_wav_cmdline": "aplay %1
```

por

```
play_wav_cmdline": "paplay -d alsa_output.usb-0d8c_C-
Media_USB_Headphone_Set-00.analog-stereo %1
```

Este cambio redireccionará todos los mensajes de voz generados por Mycroft a la tarjeta de sonido USB

Algunas habilidades reproducen podcasts o contenido mp3 como las noticias, así que, si quieres escuchar este contenido en una salida de sonido diferente, necesitas cambiar también la siguiente configuración. Cambia la siguiente línea:

```
play_mp3_cmdline": "mpg123 %1
```

por

```
play_mp3_cmdline": "mplayer -ao pulse::alsa_output.
usb-0d8c_C-Media_USB_Headphone_Set-00.analog-stereo
%1
```

El problema es que mpg123 no tiene una opción para seleccionar un receptor PulseAudio, de modo que tenemos que cambiarlo con mplayer, el cual tienes que instalar en tu sistema:

```
$ sudo apt-get install mplayer
```

Una última cosa que debemos hacer es configurar un micrófono por defecto. En mi caso, tengo tres entradas de micrófono: una de la tarjeta de sonido (no conectada), otra de mi webcam y una más del HDMI.

```
$ pacmd list-sources | egrep "index|name:"
```

```
    index: 1
        name: <alsa_input.usb-Sonix_Technology_Co.__
Ltd._USB_2.0_Camera-02.analog-mono>
    index: 2
        name: <alsa_output.usb-0d8c_C-Media_USB_Head-
phone_Set-00.analog-stereo.monitor>
  * index: 3
        name: <alsa_input.usb-0d8c_C-Media_USB_Head-
phone_Set-00.analog-mono>
    index: 4
        name: <alsa_output.platform-odroid_hdmi.
analog-stereo.monitor>
    index: 5
        name: <alsa_input.platform-odroid_hdmi.
analog-stereo>
```

Quiero utilizar el micrófono de mi webcam para la entrada de Mycroft, de modo que lo configuro como opción predeterminada en /etc/pulse/system.pa:

```
#set default microphone
set-default-source alsa_input.usb-Sonix_Technology_
Co.__Ltd._USB_2.0_Camera-02.analog-mono
```

Guarda el archivo y ya estás listo para iniciar el servicio:

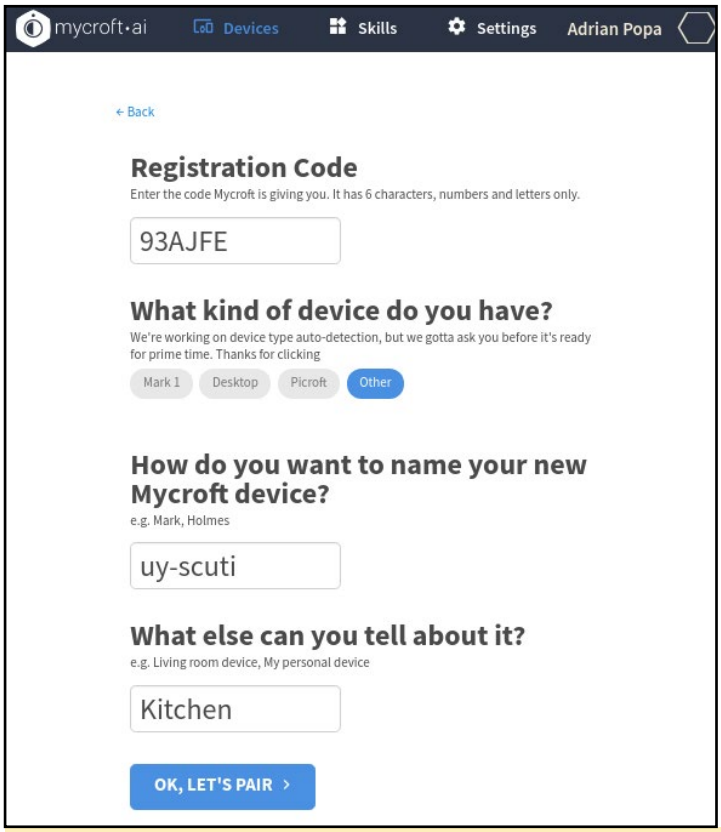
```
$ sudo systemctl enable pulseaudio
$ killall pulseaudio
$ sudo systemctl start pulseaudio
```

Emparejando e interactuando con Mycroft

Justo ahora podemos reiniciar todo el sistema, y Mycroft debería iniciarse una vez arrancado PulseAudio. El micrófono debe funcionar como es de esperar, al igual que la salida de audio. Ahora estás listo para registrar Mycroft. Para que Mycroft comience a procesar tu voz, debes empezar tus preguntas con "¡Hey Mycroft!". Para obtener un código de emparejamiento, solicita "Mycroft, registra mi dispositivo". Mycroft debería decir su código de registro que generalmente consta de 6 caracteres. Si no puede oír el código o no puedes entenderlo, puedes localizarlo en mycroft-skills.log ubicado en ~/mycroft-core/scripts/logs. Tendrás que registrarte en http://home.mycroft.ai y dirigirte a Devices -> Add device.

Ahora deberías estar listo para usar Mycroft. Puedes continuar y solicitarle diversa información, como, por ejemplo:

Hey Mycroft, ¿qué hora es?
Hey Mycroft, ¿qué día es hoy?
Hey Mycroft, la Unión Europea en la wiki.
Hey Mycroft, háblame sobre Abraham Lincoln.
Hey Mycroft, cuéntame un chiste.



Registro online

Hey Mycroft, cuéntame un chiste.
Hey Mycroft, ¿por qué el 6 tiene miedo del 7?
Hey Mycroft, ¡cántame una canción!
Hey Mycroft, ¿cuantos son 2 + 5?
Hey Mycroft, infórmame de las noticias.
Hey Mycroft, fija un recordatorio de 5 minutos.
Hey Mycroft, ¿qué tiempo hace?
Hey Mycroft, ¿cuál es la previsión para mañana?

Para saber lo que puedes preguntar, consulta la página de habilidades básicas en http://bit.ly/2uourGy. También puede revisar el archivo mycroft-skills.log y ver qué palabras clave se registran cuando se carga una habilidad. Por ejemplo, la habilidad chiste registra las siguientes frases clave: broma, hazme reír, alegrarme el día, cuéntame un chiste.

Con respecto a las habilidades

Las habilidades básicas de Mycroft son bastante útiles, pero puedes hacer mucho más haciendo uso de habilidades de terceros, disponibles en el repositorio `http://bit.ly/2tsEYh1`. Ten en cuenta que la calidad de las habilidades puede variar dependiendo de la implementación.

En el ejemplo que se muestra a continuación, añadiremos soporte para diagnósticos, Mycroft te informará del uso de la CPU, el espacio libre y la integración con Home Assistant, para que Mycroft pueda leer valores de Home Assistant o controlar las opciones con tu voz.

Las habilidades de terceros se instalan en /opt/mycroft/skills y se cargan al reiniciar Mycroft. Para añadir la habilidad de diagnóstico, simplemente clónala desde su enlace GitHub:

```
$ cd /opt/mycroft/skills
$ git clone http://bit.ly/2uOP4N1
```

La habilidad se puede configurar para ejecutar un script definido por el usuario y leer su resultado. La ruta de acceso al script definido por el usuario debe configurarse dentro de ~/.mycroft/mycroft.conf. Ten en cuenta que la ubicación de la configuración de terceros está sujeta a cambios en las nuevas versiones de Mycroft, así que consulta la documentación oficial para obtener la ubicación y la sintaxis correctas.

```
"DiagnosticSkill": {
   "script": "/home/odroid/bin/usbreset.sh"
 }
```

A continuación, reinicia Mycroft:

```
$ sudo service mycroft restart
```

Una vez relanzado el servicio, podrá realizar consultas como "Hey Mycroft, ¿cuál es el porcentaje actual de uso de la CPU?", "Hey Mycroft ejecuta diagnóstico", que iniciará tu scrript personalizado o "Hey Mycroft, ¿cuál es tu tiempo de actividad?"

Asimismo, podemos instalar la habilidad Home Assistant e integrar Mycroft con una instancia de Home Assistant en ejecución. Esta habilidad requiere de algunas dependencias externas que debemos instalar:

```
$ cd /opt/mycroft/skills
$ git clone http://bit.ly/2gVOojm
$ workon mycroft
$ cd HomeAssistantSkill
$ pip install -r requirements.txt
```

Además, necesitarás editar ~/.mycroft/mycroft.conf y añadir las siguientes líneas:

```
"HomeAssistantSkill": {
  "host": "hass.mylan.net",
  "password": "mysupersecrethasspass",
  "ssl": true|false
}
```

Tras reiniciar Mycroft, podrás consultar el estado de tus entidades Home Assistant solicitando cosas como "Hey Mycroft, conecta el aire acondicionado" para activar el interruptor "Air Conditioning Power" o "Hey Mycroft, pregunta a home assistant ¿Cuál es la temperatura de la habitación de los niños? para consultar un sensor llamado "Kids room temperature". La habilidad intentará emparejar lo que digas con los nombres de las entidades de Home Assistant y así poder controlar Home Assistant con tu voz. ¡Personalmente creo que esta es la habilidad más guay! Para obtener más ejemplos de habilidades, echa un vistazo al video de YouTube en http://bit.ly/2vfh90H.

El futuro de Mycroft

Mycroft empezó como una simple IA, pero con el apoyo de la comunidad, se está convirtiendo en algo más grande. Por ahora funciona con un patrón de pregunta/respuesta, sin mantener una conversación como tal, pero se está trabajando para cambiar esto en el futuro, de modo que finalmente seremos capaces de mantener una conversación con Mycroft (http://bit.ly/2w7L6MH). En términos de rendimiento y recursos utilizados, ya que fue diseñado para funcionar con una Raspberry Pi, no tiene ningún problema en ejecutarse en ODROIDs, incluso en un C1. En mi ODROIDC2, con otros procesos de fondo activos, Mycroft utiliza aproximadamente el 20% de la CPU en 2 núcleos con el regulador manteniendo la frecuencia de la CPU a 500 MHz cuando está inactivo.

En la actualidad, Mycroft puede ser en el mejor de los casos, ligeramente útil y no realmente un asistente personal, aunque tiene mucho potencial. Se podría comparar con MS-DOS en la década de 1980, que era algo tosco, pero hacia bien su trabajo. El tema es que todos los asistentes personales comerán errores, a veces con consecuencias divertidas (http://bit.ly/2uXQTbg). Creo que, en 10 a 20 años, el asistente personal evolucionará a algo que es difícil de predecir, como Skynet en las películas de Terminator, aunque espero que el asistente personal del futuro esté basado en la comunidad y sea de código abierto. En caso de que te quedes atascado o tengas más preguntas, consulta el hilo de soporte en http://bit.ly/2tt3crC o los foros de la comunidad de Mycroft en https://community.mycroft.ai/.

PROGRAMACION DE INTELIGENCIA ARTIFICIAL

USANDO PROLOG CON UN ODROID-XU4

por Birmsoo Kim

Los Investigadores en Inteligencia Artificial (AI), han analizado las habilidades para la resolución de problemas de los seres humanos y han implementado esas habilidades en una máquina. Esta investigación demostró que la gente tiende a inferir soluciones a un problema usando hechos y las relaciones entre esos hechos. Prolog es un lenguaje de programación que fue inventado para ejecutar la inferencia necesaria para resolver problemas involucrando hechos y reglas.

Instalando Prolog

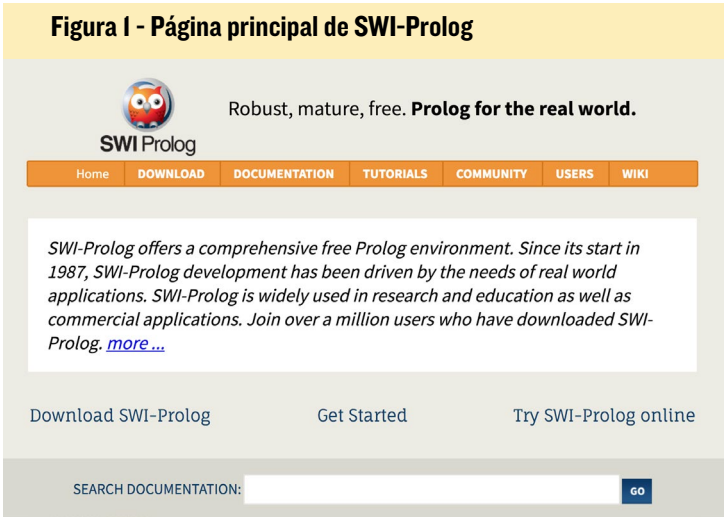
Recientemente, IBM anunció que su sistema IA dependía parcialmente del lenguaje de programación Prolog. Usando Prolog, vamos a desarrollar una simple máquina inteligente en ODROIDXU4. Si tu XU4 ejecuta Ubuntu, puedes instalar SWIProlog con los siguientes comandos:

```
$ sudo apt-get update
$ sudo apt-get install swi-prolog
```

Si quieres instalar SWI-Prolog en tu máquina con un sistema operativo diferente, como Windows o MacOS, visita el sitio web `www.swi-prolog.org`. Aquí encontraras más información relacionada con la instalación. La Figura 1 muestra la primera página del sitio de SWI-Prolog.

Una vez instalado SWI-Prolog en tu sistema, es hora de meterse de lleno con la programación IA.

Figura I - Página principal de SWI-Prolog



Implementación de un programa Prolog

Vamos a crear un archivo Prolog llamado family.pl. Este archivo describirá la relación de una familia con hechos y reglas:

```
%fact
sex(lea, female).
sex(dan, male).
sex(jay, male).
sex(esther, female).
sex(irene, female).
sex(praise, female).
sex (william, male).

parent(lea, jay).
parent(dan, jay).
parent(dan, esther).
parent(jay, praise).
parent(jay, irene).
parent(lea, william).

%rules
offspring(Y,X) :- parent(X,Y).
mother(X, Y):-parent(X,Y),female(X).
father(X, Y):-parent(X,Y),male(X).
grandparent(X,Z):-parent(X,Y),parent(Y,Z).
sister(X,Y):-parent(Z,X),parent(Z,Y),female(X),diffe
rent(X,Y).
different(X,Y):-X=Y, !, fail.
different(_,_).
Predecessor(X,Z):-parent(X,Z).
Predecessor(X,Z):-parent(X,Y),predecessor(Y,Z).
```

El archivo family.pl se compone de hechos y reglas. Los hechos muestran datos que siempre son verdaderos. Por ejemplo, sex(lea, female) es un hecho que se utiliza para describir el

género de Lea como mujer. Como es de esperar, sex(dan, male) es otro hecho que describe que Dan es varón. Otros hechos como parent(lea, jay) y parent(dan, jay) muestran la relación de Jay con Lea y Dan. Estos hechos representan que ambos son los padres de Jay.

Las reglas también se describen en este archivo. La regla "grandparent(X, Z) :- parent(X, Y) ,parent(Y, Z)" describe la condición de ser abuelo de alguien. La conclusión está en el lado izquierdo y la condición en el lado derecho.

Esta regla está representada por la siguiente condición:

Para todo X y Z, X es abuelo de Z si
X es uno de los padres de Y y
Y es uno de los padres de Z. [1]

Las dos reglas siguientes representan la madre y el padre. Para ser el padre de Y, el Y debe ser uno de los padres y varón:

mother(X, Y):-parent(X,Y), sex(X, female).

father(X,Y):-parent(X,Y), sex(X, male).

La figura 2 muestra la relación de la familia en family.pl. Un elemento superior es el progenitor de un elemento inferior. Por lo tanto, podemos encontrar que uno de los padres de Irene es Jay, y los padres de Jay son Dan y Lea.

Figura 2 - Árbol genealógico

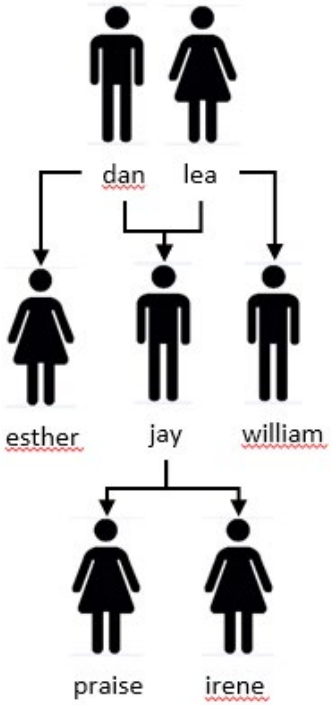


Figura 3 - swipl y carga

Figura 4 - Uno de los padres de Irene

Figura 5 - Abuelos

Ejecutando un programa Prolog

Puedes iniciar el intérprete de prolog con el comando "swipl":

```
$ swipl
```

Cuando el archivo family.pl esté listo, ejecute consult(nombre de archivo) o [nombre de archivo] para cargar el archivo family.pl en el sistema. Tal y como se muestra en la Figura 3, el sistema responde "yes" para verificar que el archivo Prolog se ha cargado correctamente.

```
?-consult(family)
?- [family]
```

Ahora, puedes hacer una pregunta sobre la relación familiar. Si quieres saber quién es el padre de Irene, la consulta parent(X, Irene) mostrará la respuesta. X es la variable que coincidirá y devolverá el nombre del padre de irene. El intérprete de prolog busca hechos concretos y encuentra jay como valor de retorno de X tal como se muestra en la Figura 4.

Puedes averiguar quién es el abuelo de Irene mediante grandparent(X, Irene). Obtendrás la solución con hechos y reglas. La regla grandparent(X, Z) :- parent(X, Y) ,parent(Y, Z) utilizará los hechos parent(lea, jay), parent(dan, jay), and parent(jay, irene) como condiciones. Cuando se cumplan las dos condiciones, habrá una solución. Los abuelos de Irene son Lea y Dan.

Trabajo futuro

El ODROID-XU4 pone de relieve la posibilidad de poder tener nuestro propio servidor inteligente. En el próximo artículo, ampliaremos la capacidad de este sistema inteligente para que pueda trabajar con un sensor.

Referencias

http://www.swi-prolog.org/

Ivan Bratko. Prolog programming for artificial intelligence, 2nd ed. Addison Wesley, J. Cassell, S. Prevost, J. Sullivan y E. Churchill.

ODROID-VU

CONTROL DEL BRILLO DE LA LUZ DE FONDO EN ANDROID

por Jörg Wolff

Esta guía te ayudará a compilar el driver HAL (Hardware Abstraction Layer) de la luz de fondo y a hacer las modificaciones necesarias para controlar el contraluz en diferentes pantallas. El drive ya está incluido en la última imagen de Android de ODROID-C2. Esta guía te enseñará a modificar el driver para compilarlo para otra placa. Antes de seguir con esta guía, asegúrate de utilizar el último Android NDK y el árbol de fuentes de Android.

Crea un directorio para el proyecto donde quieras, aunque la ubicación preferida suele ser dentro de NDK/samples. Crea un subdirectorio llamado "jni" y copia el código "lights.c" que se muestra a continuación dentro de él.

Modifica los "LOCAL_C_INCLUDES" en Android.mk con la ruta correcta del árbol de fuentes de Android, luego modifica "LOCAL_MODULE" de Android.mk con el nombre de tu placa. Abre una ventana de terminal y navega hasta la carpeta "jni". Ejecuta el comando "path/to/ndk-build", normalmente en el directorio donde está instalado el NDK.

Encontrarás la librería "liblights.odroidc.so" en la carpeta "lib/armeabi/". Usando adb, puedes copiar la librería en la carpeta ODROID "/system/lib/hw/":

```
adb remount
adb push ../libs/armeabi/liblights.odroidc.so /system/lib/hw/
adb reboot
```

Android reconocerá al driver durante el arranque. Para activar esto, el "boot.ini" debe ser editado en la siguiente sección.

El código fuente del driver HAL de la luz de fondo "lights.c":

```
/* Copyright (C) 2011 The Android Open Source Project
 *
 * Original code licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this software except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *   http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *
 * This implements a lights hardware library for the Android emulator.
 * the following code should be built as a shared library that will be
 * placed into /system/lib/hw/lights.goldfish.so
 *
 * It will be loaded by the code in hardware/libhardware/hardware.c
 * which is itself called from
 * ./frameworks/base/services/jni/com_android_server_HardwareService.cpp
 *
 * Modified by J.  Wolff
 * Support of backlight control on Odroid board via pwm on pin 33.
 */

#include <android/log.h>
#include <stdint.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
#include <fcntl.h>
#include <pthread.h>
#include <sys/ioctl.h>
```

```
#include <sys/types.h>
#include <stdbool.h>
#include <hardware/lights.h>
#include <hardware/hardware.h>

#include <assert.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/syscall.h>
#include <stdlib.h>

#define  LOG_TAG    "lights.odroidc"
/* Set to 1 to enable debug messages to the log */
#define DEBUG 0
#if DEBUG
#define  LOGD(...)  __android_log_print(ANDROID_LOG_DEBUG,LOG_TAG,__VA_ARGS__)
#else
# define LOGD(...) do{}while(0)
#endif
#define  LOGI(...)  __android_log_print(ANDROID_LOG_INFO,LOG_TAG,__VA_ARGS__)
#define  LOGW(...)  __android_log_print(ANDROID_LOG_WARN,LOG_TAG,__VA_ARGS__)
#define  LOGE(...)  __android_log_print(ANDROID_LOG_ERROR,LOG_TAG,__VA_ARGS__)

#define BACKLIGHT "/sys/devices/platform/pwm-ctrl/duty0"
#define BACKLIGHT_EN "/sys/devices/platform/pwm-ctrl/enable0"
#define BACKLIGHT_FREQ "/sys/devices/platform/pwm-ctrl/freq0"

#define init_module(mod, len, opts) syscall(__NR_init_module, mod, len, opts)
#define delete_module(name, flags) syscall(__NR_delete_module, name, flags)

#define BACKLIGHT_PWM "backlight_pwm"
#define BACKLIGHT_PWM_YES "yes"
#define BACKLIGHT_PWM_NO "no"
#define BACKLIGHT_PWM_INV "invert"

static pthread_once_t g_init = PTHREAD_ONCE_INIT;
static pthread_mutex_t g_lock = PTHREAD_MUTEX_INITIALIZER;

char * env_backlight;
bool invert, enable;

void init_globals(void)
{
    LOGD( "in: %s", __FUNCTION__ );

    // init the mutex
    pthread_mutex_init(&g_lock, NULL);

    if (enable) {

        //pwm-meson
        int fd = open("/system/lib/modules/pwm-meson.ko", O_RDONLY);
        struct stat st;
        fstat(fd, &st);
        size_t image_size = st.st_size;
        void *image = malloc(image_size);
        read(fd, image, image_size);
        close(fd);
        if (init_module(image, image_size, "") != 0) {
            LOGE( "error loading pwm-meson.ko");
            return;
        }
        free(image);

        //pwm-ctrl
        fd = open("/system/lib/modules/pwm-ctrl.ko", O_RDONLY);
        fstat(fd, &st);
        image_size = st.st_size;
        image = malloc(image_size);
        read(fd, image, image_size);
        close(fd);
        if (init_module(image, image_size, "") != 0) {
            LOGE( "error loading pwm-ctrl.ko");
            return;
        }
        free(image);

        char value[20];
        int nwr, ret;
        fd = open(BACKLIGHT_FREQ, O_RDWR);
        if (fd > 0) {
            nwr = sprintf(value, "%d\n", 1000);
            ret = write(fd, value, nwr);
            close(fd);
        }

        fd = open(BACKLIGHT_EN, O_RDWR);
        if (fd > 0) {
            nwr = sprintf(value, "%d\n", 1);
            ret = write(fd, value, nwr);
            close(fd);
        }
    }
    LOGD( "leaving  %s", __FUNCTION__ );
}

static int
is_lit(struct light_state_t const* state)
{
    return state->color & 0x00ffffff;
}

/* set backlight brightness by LIGHTS_SERVICE_NAME service.  */
static int
set_light_backlight( struct
```

```
light_device_t* dev, struct
light_state_t const* state )

{
    int nwr, ret = -1, fd;
    char value[20];
    int light_level;

    if (!enable) {
      LOGD( "%s: Not implement-
ed.", __FUNCTION__ );
      return 0;
    }

    pthread_mutex_lock(&g_lock);
    light_level = state-
>color&0xff;
    light_level = light_level <<
2;
    if (light_level > 0) light_
level += 3;
    if (invert) light_level =
1023 - light_level;

    LOGD( "level: %d", light_lev-
el);

    fd = open(BACKLIGHT, O_RDWR);
    if (fd > 0) {
        nwr = sprintf(value,
"%d\n", light_level);
        ret = write(fd, value,
nwr);
        close(fd);
    }
    pthread_mutex_unlock(&g_
lock);
    return ret;
}

static int
set_light_buttons( struct light_
device_t* dev, struct light_
state_t const* state )
{
    /* @Waiting for later imple-
mentation.  */
    LOGD( "%s: Not implemented.",
__FUNCTION__ );

    return 0;
}

static int
set_light_battery( struct light_
device_t* dev, struct light_
state_t const* state )
{
    /* @Waiting for later imple-
mentation.  */
    LOGD( "%s: Not implemented.",
__FUNCTION__ );

    return 0;
}

static int
set_light_keyboard( struct light_
device_t* dev, struct light_
state_t const* state )
{
    /* @Waiting for later imple-
mentation.  */
    LOGD( "%s: Not implemented.",
__FUNCTION__ );

    return 0;
}
static int
set_light_notifications( struct
light_device_t* dev, struct
light_state_t const* state )
{
    /* @Waiting for later imple-
mentation.  */
    LOGD( "%s: Not implemented.",
__FUNCTION__ );

    return 0;
}

static int
set_light_attention( struct
light_device_t* dev, struct
light_state_t const* state )
{
    /* @Waiting for later imple-
mentation.  */
    LOGD( "%s: Not implemented.",
__FUNCTION__ );

    return 0;
}

/** Close the lights device */
static int
close_lights( struct light_
device_t *dev )
{
    free( dev );

    if (delete_module("pwm-me-
son", O_NONBLOCK) != 0) {
        LOGE("delete_module pwm-
meson");
        return EXIT_FAILURE;
    }

    if (delete_module("pwm-ctrl",
O_NONBLOCK) != 0) {
        LOGE("delete_module pwm-
ctrl");
        return EXIT_FAILURE;
    }

    return 0;
}
/**
 * module methods
 */
/** Open a new instance of a
lights device using name */
static int
open_lights( const struct hw_
module_t* module, char const
*name,
        struct hw_device_t **de-
vice )
{
    void* set_light;


    if (0 == strcmp( LIGHT_ID_
BACKLIGHT, name )) {
        set_light = set_light_
backlight;

          //check the bootargs
for backlight_pwm
          FILE * fp;
          char * line = NULL;
          char * list;
          char * value;
```

```
        size_t len = 0;

        fp = fopen("/proc/cmd-
line", "r");
        if (fp != NULL) {
                getline(&line,
&len, fp);
                list = strtok
(line, " ");
                while (list !=
NULL)
                {
                    LOGD
("%s\n",list);
                    if (0 ==
strncmp(BACKLIGHT_PWM, list, 13))
{
                        env_
backlight = strtok(list, "=");
                        env_
backlight = strtok(NULL, "=");
                        break;
                    }
                    list =
strtok (NULL, " ");
                }
        }
      enable = false;
      invert = false;
        if (env_backlight !=
NULL) {
            LOGD("backlight_pwm
: %s", env_backlight);

            if (0 == strncmp(
BACKLIGHT_PWM_YES, env_backlight,
3 )) {
                    enable =
true;
            } else if (0 ==
strncmp( BACKLIGHT_PWM_NO, env_
backlight, 2 )) {
                    //enable =
false;
                //return -EIN-
VAL;
            } else if (0 ==
strncmp( BACKLIGHT_PWM_INV, env_
backlight, 6 )) {
                    enable =
true;
                    invert = true;
            } else { enable =
false; }
        }
    } else if (0 == strcmp(
LIGHT_ID_KEYBOARD, name )) {
        set_light = set_light_
keyboard;
    } else if (0 == strcmp(
LIGHT_ID_BUTTONS, name )) {
        set_light = set_light_
buttons;
    } else if (0 == strcmp(
LIGHT_ID_BATTERY, name )) {
        set_light = set_light_
battery;
    } else if (0 == strcmp(
LIGHT_ID_NOTIFICATIONS, name )) {
        set_light = set_light_no-
tifications;
    } else if (0 == strcmp(
LIGHT_ID_ATTENTION, name )) {
        set_light = set_light_at-
tention;
    } else {
        LOGD( "%s: %s light isn't
supported yet.", __FUNCTION__,
name );
        return -EINVAL;
    }

    pthread_once(&g_init, init_
globals);

    struct light_device_t *dev
= malloc( sizeof(struct light_
device_t) );
    if (dev == NULL) {
        return -EINVAL;
    }
    memset( dev, 0, sizeof(*dev)
);

    dev->common.tag = HARDWARE_
DEVICE_TAG;
    dev->common.version = 0;
    dev->common.module = (struct
hw_module_t*)module;
    dev->common.close = (int (*)
(struct hw_device_t*))close_
lights;
    dev->set_light = set_light;

    *device = (struct hw_
device_t*)dev;
    return 0;
}

static struct hw_module_methods_t
lights_module_methods = {
    .open =  open_lights,
};

/*
 * The emulator lights Module
 */
struct hw_module_t HAL_MODULE_
INFO_SYM = {
    .tag = HARDWARE_MODULE_TAG,
    .version_major = 1,
    .version_minor = 0,
    .id = LIGHTS_HARDWARE_MOD-
ULE_ID,
    .name = "Odroid lights Mod-
ule",
    .author = "Amlogic",
    .methods = &lights_module_
methods,
};
```

Android.mk:

```
# Copyright (C) 2011 The Android
Open Source Project.
#
# Original code licensed under
the Apache License, Version 2.0
(the "License");
# you may not use this software
except in compliance with the
License.
# You may obtain a copy of the
License at
#
#   http://www.apache.org/licens-
es/LICENSE-2.0
#
# Unless required by applicable
law or agreed to in writing,
software
# distributed under the License
is distributed on an "AS IS" BA-
```

```
SIS,
# WITHOUT WARRANTIES OR CONDI-
TIONS OF ANY KIND, either express
or implied.
# See the License for the specific
language governing permissions
and
# limitations under the License.

LOCAL_PATH := $(call my-dir)

# HAL module implemenation, not
prelinked and stored in
# hw/<LIGHTS_HARDWARE_MODULE_
ID>.<ro.hardware>.so
include $(CLEAR_VARS)
LOCAL_C_INCLUDES += /path/to/an-
droid/source/tree/core/include
LOCAL_C_INCLUDES += /path/to/
android/source/tree/hardware/lib-
hardware/include
LOCAL_SRC_FILES := lights.c
LOCAL_PRELINK_MODULE := false
LOCAL_LDLIBS    := -llog
LOCAL_SHARED_LIBRARIES := libcu-
tils
LOCAL_MODULE := lights.odroidc
LOCAL_MODULE_TAGS := optional
include $(BUILD_SHARED_LIBRARY)
```

Application.mk:

```
NDK_TOOLCHAIN_VERSION=clang
APP_ABI := armeabi
APP_PLATFORM := android-24
```

El código lo puedes encontrar en Github en `http://bit.ly/2tshhFx`. Ten en cuenta que la frecuencia de la pwm está codificada a 1KHz. Puede modificar la frecuencia en este fragmento de código:

```
fd = open(BACKLIGHT_FREQ, O_
RDWR);
if (fd > 0) {
    nwr = sprintf(value, "%d\n",
1000);
    ret = write(fd, value, nwr);
    close(fd);
}
```

También es posible sobrescribir la frecuencia en tiempo de ejecución:

```
$ echo 100 > /sys/devices/pwm-
ctrl.43/freq0
```

Esto puede ser útil si te gusta experimentar con el pin ADJ de un convertidor CCFL, que se utiliza con pantallas LCD antiguas.

VU7+

Para utilizar el driver con tu pantalla LCD, aquí tienes algunos ejemplos. Primero, para conectar la señal pwm del pin 33 de ODROID, se necesita hacer una pequeña modificación en la parte trasera del VU7+. La mejor forma de llevarla a cabo es utilizar una resistencia cableada de unos 330 Ohmios, que

Figura 1 - Vista de los cambios en VU7

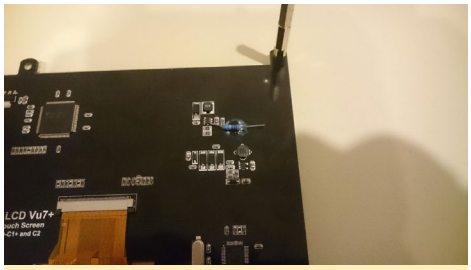


Figura 2 - Conexión VU7+ al CI

se debe pegar a la placa con superglue y soldarla al pin 4 de la PT4103, o como alternativa a la resistencia smd de 10k, marcada con 103.

Ten en cuenta que también funcionaría sin una resistencia mediante una conexión directa. Sin embargo, no estar de más ser cuidadoso y no quemar nuestro querido hardware por culpa de un error humano, así que usamos una resistencia.

Para el VU7+, se debe utilizar la siguiente configuración en "boot.ini":

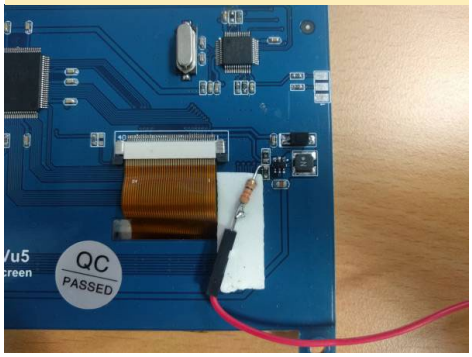
```
setenv hdmimode "1024X600p60hz"
setenv vout_mode "dvi"
setenv backlight_pwm "yes"
setenv bootargs "root=/
dev/mmcblk0p2 rw
console=ttyS0,115200n8 no_con-
sole_suspend vdaccfg=${vdac_con-
fig} logo=osd1,loaded,${fb_
addr},${outputmode},full
hdmimode=${hdmimode}
cvbsmode=${cvbsmode}
hdmitx=${cecconfig} vout=${vout_
mode} disablehpd=${disablehpd}
${disableuhs} androidboot.
serialno=${fbt_id#} ir_
remote=${ir_remote} usbcore.
autosuspend=-1 ${selinuxopt} sus-
pend_hdmiphy=${suspend_hdmiphy}}
backlight_pwm=${backlight_pwm}"
```

VU5 & VU7

Los ajustes para éstos son idénticos a los de la VU7+. Para VU5 y VU7, debemos cambiar algunos parámetros en el archivo "boot.ini":

```
setenv hdmimode "800x480p60hz"
setenv vout_mode "dvi"
setenv backlight_pwm "yes"
setenv bootargs "root=/
dev/mmcblk0p2 rw
console=ttyS0,115200n8 no_con-
sole_suspend vdaccfg=${vdac_con-
fig} logo=osd1,loaded,${fb_
addr},${outputmode},full
hdmimode=${hdmimode}
cvbsmode=${cvbsmode}
hdmitx=${cecconfig} vout=${vout_
mode} disablehpd=${disablehpd}
${disableuhs} androidboot.
```

Figura 3 - Modificaciones del VU5



```
serialno=${fbt_id#} ir_
remote=${ir_remote} usbcore.
autosuspend=-1 ${selinuxopt} sus-
pend_hdmiphy=${suspend_hdmiphy}}
backlight_pwm=${backlight_pwm}"
```

VU8

El VU8 ya tiene un conector para la señal pwm. Para el VU8, se debe utilizar la siguiente configuración en "boot.ini":

```
setenv hdmimode "1024X768p60hz"
setenv vout_mode "dvi"
setenv backlight_pwm "invert"
setenv bootargs "root=/
dev/mmcblk0p2 rw
console=ttyS0,115200n8 no_con-
sole_suspend vdaccfg=${vdac_con-
fig} logo=osd1,loaded,${fb_
addr},${outputmode},full
hdmimode=${hdmimode}
cvbsmode=${cvbsmode}
hdmitx=${cecconfig} vout=${vout_
mode} disablehpd=${disablehpd}
${disableuhs} androidboot.
serialno=${fbt_id#} ir_
remote=${ir_remote} usbcore.
autosuspend=-1 ${selinuxopt} sus-
pend_hdmiphy=${suspend_hdmiphy}}
backlight_pwm=${backlight_pwm}"
```

Ten en cuenta que aquí "invert" en el VU8 está invertido, lo que significa que el 100% del trabajo se traduce en que la luz de fondo este apagada.

Controlando el brillo con Java

Para controlar el brillo desde código Java, aquí tienes algunos fragmentos de código. Asegúrate de añadir los permisos necesarios en el archivo AndroidManifest.xm. En primer lugar, pide permiso <WRITE_SETTINGS> en Android Marshmallow:

```
<uses-permission
android:name="android.permission.
WRITE_SETTINGS" />
```

A través de código podemos controlarlo con el siguiente fragmento:

```
if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.M) {
    if (!Settings.System.canWrite
(getApplicationContext())) {
        Intent intent = new
Intent(android.provider.Settings.
ACTION_MANAGE_WRITE_SETTINGS);
        intent.setData(Uri.
parse("package:" + getPackage-
Name()));
        intent.addFlags(Intent.
FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
    }
}
```

El siguiente fragmento lee el valor exacto del brillo:

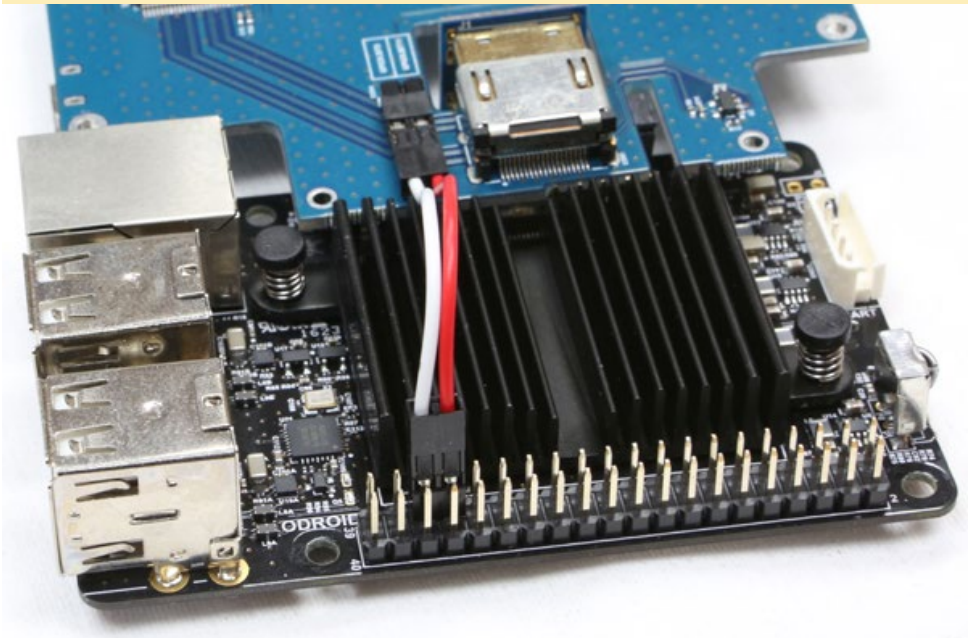
```
//Get the current system bright-
ness
try
{
    Settings.System.SCREEN_
BRIGHTNESS_MODE_MANUAL);
    int savedBright-
ness = Settings.System.
getInt(getContentResolver(),
        Settings.System.SCREEN_
BRIGHTNESS);
}
catch (Settings.SettingNotFound-
Exception e)
{
    Log.e("Error", "Cannot access
system brightness");
    e.printStackTrace();
}
```

Luego, escribe el nuevo valor del brillo:

```
int brightness = 50 // (0 … 100)
Settings.System.
putInt(getContentResolver(), Set-
tings.System.SCREEN_BRIGHTNESS,
brightness);
```

Para más información, por favor visita el hilo del foro en `http://bit.ly/2uSpczG`, y la página Wiki en `http://bit.ly/2vTKSbG`.

Figura 4 - Conexión del VU8



JUEGOS LINUX

FANBOY PARTE I – ¿SOY UN FANATICO DE NINTENDO?

por Tobias Schaaf

A la hora de hablar de los entusiastas de los juegos retro, se puede decir que existen dos bandos en líneas generales: Los fanáticos de Nintendo y los entusiastas de SEGA. En los años 80 y 90, éstos eran los dos principales jugadores del mercado. Hubo, sin embargo, algunas otras empresas que también reunieron una base de fans alrededor de sus productos, como Atari, Commodore y NEC. Me gustaría echar un vistazo a estos grupos de fans para ver si puedo identificarme con uno o más de ellos y ver realmente qué tipo de fanboy soy.

Hardware Nintendo

Probablemente Nintendo sea el jugador más grande del pasado. Si te paras a pensar en ello, posiblemente es el último fabricante de consolas de la década de los ochenta que sigue operando como fabricante de consolas como tal. No se puede negar que Nintendo es una empresa muy conocida que hizo y sigue haciendo, buenos productos. Sin embargo, si alguien me preguntara que si yo era un fanboy de Nintendo, probablemente diría "No, nunca lo fui." Probablemente sea cierto, pero ¿por qué? ¿Realmente es cierto?

Al crecer en la Alemania del Este a principios de los años ochenta, no disponíamos de todas las ventajas técnicas de la Alemania Occidental u otros países. Las consolas eran algo fuera de lo común, y cuando comenzaron a estar disponibles en 1989 y a principios de los 90, sólo unas cuantas personas tenían acceso a ellas. No hubo conmoción en cuanto a qué consola llegase primero, ya que tanto el Sistema de Entretenimiento de Nintendo (NES) como el Sistema de SEGA ya estaban bien consolidados por aquel entonces. La gente simplemente elegía lo que quería. Aunque no conocía a demasiadas personas que tuviese una consola en ese momento, uno de mis tíos sí que tenía una NES, a la que pude jugar de vez en cuando.

¿Qué juegos me gustaban? Super Mario Bros estaba bien, pero nunca lo pobre en profundidad. ¿Lo disfruté? Más que probable, desde que era niño, pero sin duda alguna no era mi juego favorito.

¿Cuál era mi favorito, entonces? El Nintendo World Cup era muy divertido y todavía sigue siéndolo. Como juego de fútbol tenía muy pocas reglas. Podrías atacar a tus enemigos, hacerlos caer, hacer disparos con truco, etc. Era divertido, los controles eran fáciles, y pronto averigüé cómo derrotar al equipo contrario. Fue muy divertido eliminar a todos los jugadores contrarios y verlos intentando pasar la pelota a los jugadores que yacían en el suelo. También tenía una asombrosa banda sonora, probablemente una de los mejores en toda la biblioteca NES. Como he dicho, nunca he jugado demasiado a los juegos de NES y puede que me haya perdido algunos muy buenos de niño. Pero hay buenas razones por la que no me considero un fanático de Nintendo.

Recientemente he descubierto que una de mis primeras consolaso más exactamente, un dispositivo de juego portátil, era en realidad de Nintendo, aunque sólo unos pocos lo reconocerían como un juego NES. "Egg" formó parte de la serie Game & Watch de Nintendo. Game & Watch era una serie de unidades portátiles que funcionaban como despertador y dispositivo de juego, aunque dudo mucho que alguien lo usara como reloj. El juego era muy simple: cuatro botones direccionales que los jugadores usaban para coger huevos mientras caían.

Como todos sabemos, Nintendo creó un montón de consolas y dispositivos portátiles, como la Game Boy. Aunque conocía a algunas personas en la escuela que tenían un Game Boy, nunca me pareció interesante. No tenía colores, ni luces de fondo y los gráficos no estaban

"Egg", un juego Game & Watch de Nintendo

muy definidos. Vi a un tipo que tenía una Sega Game Gear. Tenía colores, se podía jugar a los juegos de la SEGA Master System y contaba con un receptor de TV. Lo siento, ¿Qué era esa cosa de Game Boy de la que hablabamos?

No llegué a conocer a nadie con un "buen" dispositivo Nintendo por aquel entonces. Nunca he conocido a nadie con una Super Nintendo y nunca he visto un Game Boy Color. Ni siquiera he oído hablar del Game Boy Advance hasta mucho más tarde. En lo que respecta a la Nintendo, me lo perdí todo entre la NES/Game Boy y la Nintendo DS. Probablemente ésta sea la principal razón por la que no me considero un fanboy de Nintendo, con la excepción de la Nintendo 64, que contaba con una consola de demostración en el centro comercial local. De niño, jugaba a juegos de la N64 mientras mis padres compraban. Era divertido, incluso si tenías que compartirla con niños que no sabían jugar, pero en realidad nunca tuve la consola y no

había muchos juegos expuestos excepto el Super Mario 64.

Como un adulto, tengo una Nintendo DSi XL, un dispositivo portátil muy divertido. También me hice con una Nintendo Wii, que era estupenda para jugar de vez en cuando con los amigos o sólo, incluso a los juegos de deportes, los gráficos en realidad no eran muy buenos. Realmente disfruté y sigo haciéndolo con la Nintendo DS. Hay muchos y muy divertidos juegos incluyendo RPGs, aventuras y estrategia que de hecho me gustan bastante, y los gráficos están bastante bien. Es una consola muy divertida.

También disfruté con juegos como Time Hollow, la serie Ace Attorney, Another Code: Two Memories, Advance Wars, la serie Cooking Mama, la serie Final Fantasy, la serie Luminous Arc (me encanta esta), varios juegos Dragon Ball Z, Bleach the 3rd Phantom (uno de mis juegos favoritos), Summon Night: Twin Age, Suikoden Tierkreis, la serie Shin Megami Tensei, Infinite Spac y muchos más. Hay tantos juegos increíbles para la Nintendo DS que es muy difícil elegir cuales serán tus favoritos.

La Wii tenía algunos juegos muy buenos con lo que disfrutaba; principalmente con la serie Rayman Raving Rabbids, que era sorprendente como juego en grupo, pero también algunos juegos más formales, como Blob 1 y 2, Resident Evil 4, Mario Kart Wii (que es bastante mejor que la versión N64), MadWorld, Xenoblade Chronicles, Red Steel 2, Overlord, No More Heroes, The Last Story, Pandora's Tower y muchos más.

Emulación Nintendo en ODROID

Cuando empecé a trabajar con la emulación en ODROID en 2012, las posibilidades eran bastante limitadas. No había soporte en 3D, Lakka no existía, y RetroArch todavía era bastange desconocido por aquel entonces.

Existían un par de emuladores independientes basados en Simple DirectMedia Layer (SDL) que más o menos funcionaban, pero el gran adelanto llegó con Mednafen. Mednafen es un emulador multi-sistema que te permite emular todo tipo de consolas y dispositivos portátiles con unas velocidades sorprendentes. Sin embargo, tenía algunas desventajas. Super Nintendo no funcionaba a una velocidad decente, tampoco lo hacia los juegos de PlayStation, aunque sí que había muchas otras plataformas que funcionaban bastante bien.

Aunque los juegos NES no era lo mio, encontré mi pasión en los juegos de Game Boy Advance como Riviera: The Promised Land, los juegos de Dragon Ball Z, Advance Wars, Medabots, Summon Night 1 y 2, y muchos más. Incluso más adelante, cuando estaba disponible el driver GPU 3D y la emulación Super Nintendo finalmente funcionaba, seguía sin estar especialmente interesado en los juegos de Super Nintendo. Hasta ese momento, sólo jugaba a unos cuantos juegos de Super Nintendo. Personalmente, prefiero la Game Boy Advance a la Super Nintendo. Aparte de la resolución, el Game Boy Advance es mucho mejor que la Super Nintendo en cualquier aspecto.

Muchos juegos de la Super Nintendo también fueron lanzados para el Game Boy Advance, a menudo con gráficos y sonidos mejorados. Por eso, normalmente he jugado a las versiones Game Boy Advance, ya que por lo general eran mejores que las versiones Super Nintendo, en mi opinión.

Realmente disfruté con la Game Boy Advance, y algunos de mis juegos favoritos de todos los tiempos son juegos de Game Boy Advance o juegos que jugué por primera vez en Game Boy Advance. Hay unos cuantos juegos de Super Nintendo que me gustan mucho, como E.V.O.: Search for Eden, pero cuando se trata de Nintendo, por lo general termino eligiendo juegos de Game Boy Advance.

También hay otras consolas de Nintendo y dispositivos portátiles disponibles para emulación en ODROID, como son Virtual Boy, Nintendo 64, Nintendo DS, y muchos más. Estos emuladores funcionan bastante bien, pero ninguno de ellos se acerca a la cantidad de tiempo de juego que paso jugando con los emuladores Game Boy Advance.

Hay algunos juegos interesantes para la Nintendo 64, pero por lo general, no soy fan. Realmente disfruto con la Nintendo DS, aunque usar un ratón o un gamepad en lugar de una entrada táctil hace que la experiencia de usar el emulador NDS sea bastante diferente a la "realidad". Sin embargo, hay algunos juegos en la Nintendo DS que vale la pena jugar en ODROID, me atrevo a decir que posiblemente sea mi segundo emulador favorito para ODROID. La emulación no es perfecta, ya que tiene algunos problemas de velocidad, que con suerte puede que sean resueltos en algún momento en el futuro, no obstante siguen siendo divertidos, especialmente en el XU3/XU4.

Reflexiones finales

Aunque he tenido buenas experiencias con la Game Boy Advance y Nintendo DS y en menor medida, con el N64 y Wii, todavía no me considero un fanboy de Nintendo. Esto podría deberse a que he tenido poco contacto con Nintendo de niño, pero también siento que tengo a mi alcance mejores opciones, o al menos diferentes.

Esto no significa que Nintendo no tenga el reconocimiento que se merece. Por el contrario, la Game Boy Advance y la Nintendo DS son realmente grandes máquinas. Las consolas actuales como Wii U y Switch probablemente tendrán un impacto positivo en la industria del juego. Por desgracia, no estoy familiarizado con el GameCube, aunque tengo la sensación de que también disfrutaría con ella.

Aun así, las opciones de emulación de Nintendo en ODROID siguen estando limitadas a Game & Watch, Game Boy y Game Boy Color, NES, Virtual Boy, Super Nintendo, Game Boy Advance, Nintendo 64 y Nintendo DS. Un gran número de consolas, pero no demasiadas a la que estoy deseando jugar. En conclusión, sigo con mi premisa inicial: Me gusta Nintendo, pero no soy fanboy.

MIGRANDO DESDE UBUNTU MATE A LUBUNTU

UNA GUIA PASO A PASO PARA CAMBIAR A UN ESCRITORIO LXDE

por Miltiadis Melissas

Hardkernel proporciona excelentes versiones para cada placa que produce. La última imagen para el ODROID-XU4 (http://bit.ly/2utQxEE) es Ubuntu 16.04 con el escritorio Mate como interfaz gráfica de usuario (GUI). Durante los dos últimos años, la compañía ha preferido proporcionar a los usuarios versiones públicas de Linux/Ubuntu con el escritorio Mate. No se trata de una simple casualidad, ya que Mate es un gran entorno GUI gracias al duro trabajo realizado por su equipo de desarrollo. Sin embargo, tan bueno como posible, de vez en cuando los usuarios necesitan un cambio de ritmo. Esta sencilla guía te proporciona exactamente eso: una breve descripción paso a paso de cómo cambiar del escritorio Mate a su equivalente LXDE, transformando Ubuntu en su variante Lubuntu totalmente funcional. Echemos un vistazo a cómo ocurre esto.

Instalación

Primero, descarga la versión del software de Linux/Ubuntu (v2.0) Kernel 4.9 XU3/XU4 de la excelente página wiki de XU4 de Hardkernel (`http://bit.ly/2utQxEE`) y móntala en una tarjeta eMMC o microSD. Las principales características de esta versión son:

- Linux Kernel LTS 4.9.27
- Ubuntu 16.04.2
- GPU Mali actualizada a la última versión (r17p0)

Tras el primer arranque, aparecerá un mensaje que dice "The RootFS Autoresize feature has changed!!! Once everything is done after auto-reboot, the power will turn off. Wait a couple of minutes. Please press the power button if the blue LED is off."

Tan pronto como arranque la imagen, inicia sesión con las siguientes credenciales, como muestra la Figura 1:

```
$ user/root: odroid
$ password: odroid
```

Espere unos segundos hasta que el sistema cargue su entorno de escritorio Mate, como se muestra en la Figura 2.

Figura 1 - Pantalla de inicio de sesión

Figura 2 - Entorno de escritorio Mate

Lo primero es lo primero: es bueno que actualices tu sistema. Es muy fácil. Abre una ventana de terminal (Ctrl + Alt + T) y escribe los siguientes comandos de Linux, uno por uno, permitiendo al sistema que finalice cada paso antes de pasar a la siguiente:

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

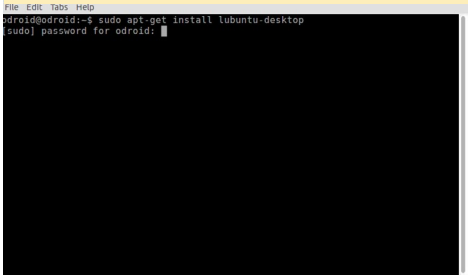
Antes de reiniciar, también necesitará actualizar el Kernel:

```
$ sudo apt-get install \
  linux-image-xu3
$ sudo reboot
```

Después de reiniciar, vuelve a iniciar sesión y abre una ventana de terminal (Ctrl + Alt + T). Instala el escritorio Lubuntu escribiendo el siguiente comando, como se muestra en la Figura 3.

```
$ sudo apt-get install lubuntu-
desktop
```

Figura 3 – Instalando el escritorio Lubuntu



Llegados a este punto, no es mala idea idea tomar una copa o un aperitivo mientras que el escritorio Lubuntu se descarga y se instala en tu sistema. Tan pronto como se complete la instalación, reinicia tu dispositivo.

```
$ sudo reboot
```

Ahora es el momento de personalizar tu nuevo escritorio Lubuntu, como se muestra en la Figura 4.

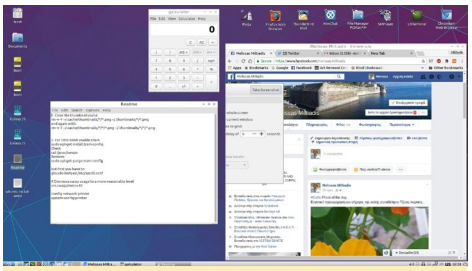


Figura 4 - Entorno de escritorio Lubuntu

Modificar hora y fecha

Accede al menú Inicio haciendo clic en el icono LXDE, luego selecciona Tools > Time and Date. En la nueva ventana, configura la hora y el día correspondientes, tal y como se muestra en las figuras 5 y 6

Instalar idiomas

Instalar idiomas en el sistema también es fácil. En el menú de Inicio, selecciona Preferences > Language Support e instala los correspondientes idiomas para tu sistema. El inglés es el idioma instalado por defecto.

Figura 5 - Abriendo las herramientas de sistema de Lubuntu

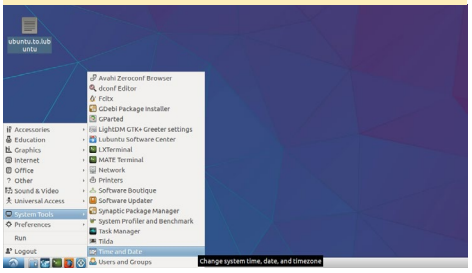
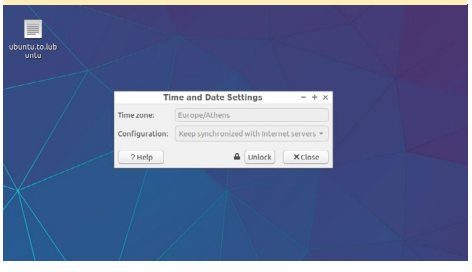


Figura 6 - Configurando la hora y el día



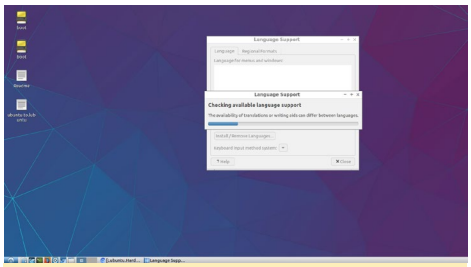


Figura 7 - El inglés es el idioma predeterminado para Lubuntu

Añadir disposiciones de teclado

Si vas a utilizar Lubuntu como un escritorio totalmente funcional, tendrás que seleccionar una disposición de teclado, especialmente si el inglés no es tu idioma nativo. El proceso es algo diferente en esta ocasión. Haga clic con el botón derecho en el panel y seleccione Add/ Remove Panel Items. Desde allí, haga clic en Add y selecciona Keyboard Layout Handler, como se muestra en las Figuras 8 y 9.

Añadir elementos

Este último paso es principalmente para fines de apariencia. Puedes añadir un indicador “Keyboard LEDs” para facilitarte las cosas a la hora de hacer clic derecho en el panel y volver a seleccionar Add/Remove Panel Items. Desde allí, selecciona Keyboard LEDs y marca cada casilla para activar todos los LED, como se muestra en la Figura 10 LEDs”.

Figura 8 - Abriendo la funcionalidad Add/Remove Panel Items

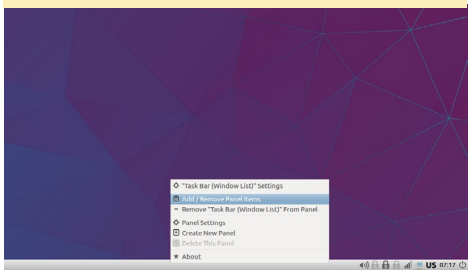
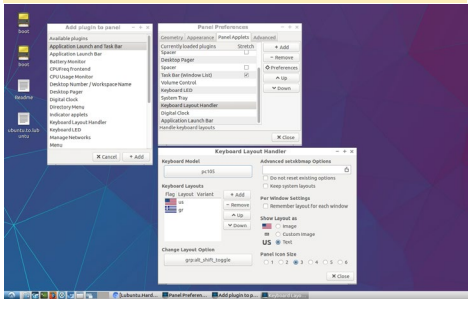


Figura 9 – Seleccionar el diseño del teclado



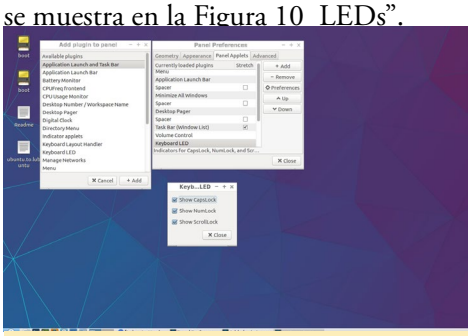


Figura 10 – Activando los LED del teclado

Notas

Espero que hayas disfrutado con esta sencilla guía paso a paso para transformar Ubuntu en Lubuntu, ahora es el momento de probarla. Me encantaría recibir tus comentarios en mi cuenta de twitter (@ Miltos01). ¡Los cambios en la vida son siempre necesarios e importantes para mantenerla interesante, y lo mismo ocurre con Ubuntu!

USANDO EL ADAFRUIT I28X64 OLED BONNET SOBRE UN ODROID-C1+

PROGRAMANDO CON LUMA.OLED Y WIRINGPI

por Dennis Chang (@dchang0)

Hay muchas placas de visualización SPI OLED o I2C en el mercado, pero la mayoría están dirigidas al mercado Arduino, de modo que el orden de sus pines es incompatible con los ODROIDs. Hasta el momento, sólo he visto tres placas de visualización I2C OLED diseñadas específicamente para que ajusten directamente en los pines del cabezal GPIO de la Raspberry Pi. Las tres son monocromáticas y están basada en el chip SSD1306 OLED/ PLED.

Puesto que el miembro del foro ODROID, @eudoxos, tuvo una mala experiencia con una placa OLED monocromática Adafruit 128x32, no quería arriesgarme con las otras dos opciones que cuentan con la misma pantalla de 128x32. Algunos miembros del foro han confirmado que funcionaba una pantalla OLED I2C 128x64 genérica, de modo que pensé que tal vez la Adafruit 128x64 OLED Bonnet/ pHat (ID de producto 3531) funcionaría. Lo hace y aquí tienes cómo.

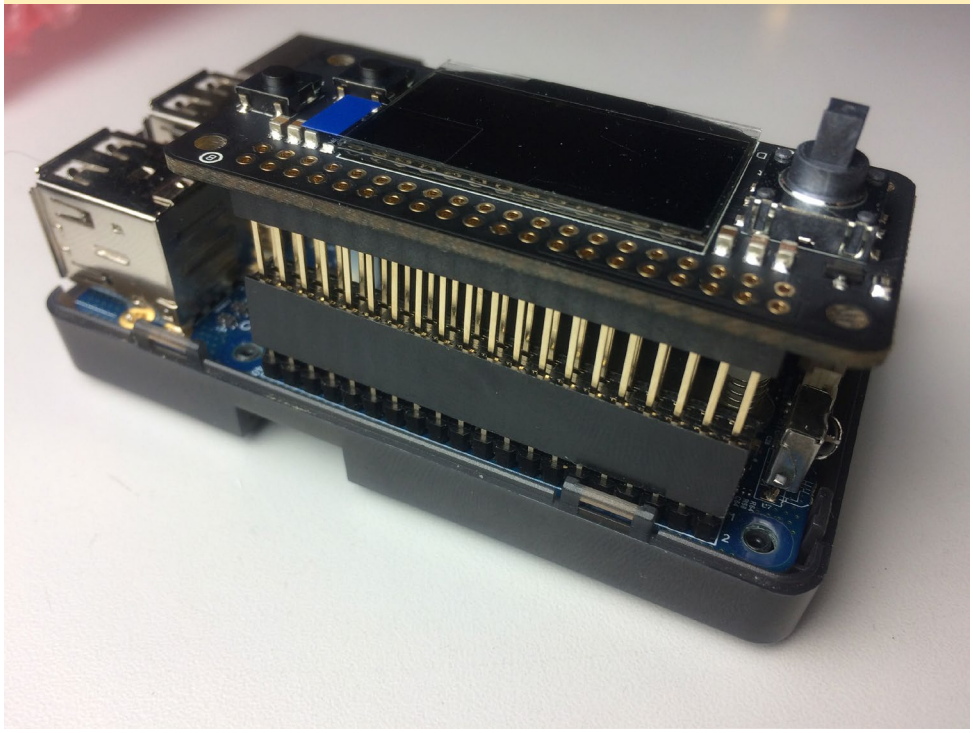
Al igual que la mayoría de los bonnets o pHats, el Adafruit 3531 tiene cabezal hembra de 2x20 passthrough SMT 5mm de bajo perfil en la parte inferior. Este es aproximadamente 2 mm más corto para que el 3531 encaje de forma segura si se coloca directamente sobre los pines GPIO de mi ODROID C1+ debido a la altura del disipador de calor. Tuve que usar un cabezal extensor 2x20 para elevar el 3531 por encima del disipador de calor.

Para hacer pruebas, no me hizo falta empujar totalmente los cabezales, tal y como se muestra en la Figura 1. Si lo tuviera, el bonner se situaría a sólo 1 mm por encima de los puertos USB, y los pines machos del cabezal apilado continuarían a través del bonnet para que fueran accesibles por la parte superior. Sería posible evitar usar el cabezal de apilamiento si retirase el cabezal SMT de bajo perfil en el 3531 y colocara en su lugar un cabezal hembra de altura estándar o de mayor altura.

Empecé con mi fiel ODROID C1 +, le instalé la imagen Ubuntu 16.04 mínimal y lo arranque sin monitor. Para hacer funcionar la pantalla Adafruit 3531, utilicé la misma librería de drivers que otros usuarios confirmaron que funcionaba, que es Luma.OLED (`http://bit.ly/2tSA5gP`).

Ten en cuenta que en realidad no lo compile desde la fuente. Leí las instrucciones de instalación en `http://bit.ly/2gVUCj2` y averigüe que funciona:

Figure I - Adafruit 353I mounted on the ODROID-CI+



```
$ sudo -s
$ apt-get install python-dev
python-pip libfreetype6-dev
libjpeg-dev
$ pip install --upgrade pip
$ pip install mock
$ pip install pytest
$ pip install --upgrade luma.oled
```

ThLa instalación fue bien, y se instalaron mock y pytest como prerrequisitos para Luma.OLD. Asegúrate de no salir de la sesión sudo -s para el resto de pasos del artículo.

Escribe el siguiente comando para activar I2C:

```
$ modprobe aml_i2c
$ echo "aml_i2c" | sudo tee /etc/
modules
```

Opcionalmente puedes instalar y utilizar i2cdetect para escanear el bus I2C para la pantalla:

```
$ apt-get install i2c-tools
$ i2cdetect -y 1
```

Así es como aparece el Adafruit 3531 en la exploración (como dirección 0x3c):

```
     0  1  2  3  4  5  6  7  8  9
a  b  c  d  e  f
00:          -- -- -- -- -- -- --
-- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- --
-- -- 3c -- -- --
40: -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- --
-- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Inmediatamente probé la pantalla con este simple código Python con las instrucciones de http://bit.ly/2tsEQ10. Guarda el siguiente código fuente como testdisplay.py:

```
from luma.core.interface.serial
import i2c, spi
from luma.core.render import
canvas
from luma.oled.device import
ssd1306, ssd1325, ssd1331, sh1106

# rev.1 users set port=0
# substitute spi(device=0,
port=0) below if using that in-
terface
serial = i2c(port=1,
address=0x3C)

# substitute ssd1331(...) or
sh1106(...) below if using that
device
device = ssd1306(serial)

with canvas(device) as draw:
    draw.rectangle(device.
bounding_box, outline="white",
fill="black")
    draw.text((30, 30), "Hello
ODROID", fill="white")
raw_input()
```

Para ejecutar el programa anterior, escribe el siguiente comando:

```
$ python testdisplay.py
```

Una gran ventaja de Adafruit 3531 es que incluye dos botones y un joystick de 5 vías, algo que realmente necesitaba para un proyecto basado en C1+, que funciona con pilar y que necesita una sencilla interfaz de menú. La pantalla táctil ODROID 3.2 TFT consumía demasiada energía y era demasiado difícil usarla mientras paseo, aunque los botones y el joystick habrían sido perfectos.

Para conseguir que el joystick y los botones funcionen, utiliza la versión HardKernel de wiringPi para ODROID. Instala Git si no lo tienes:

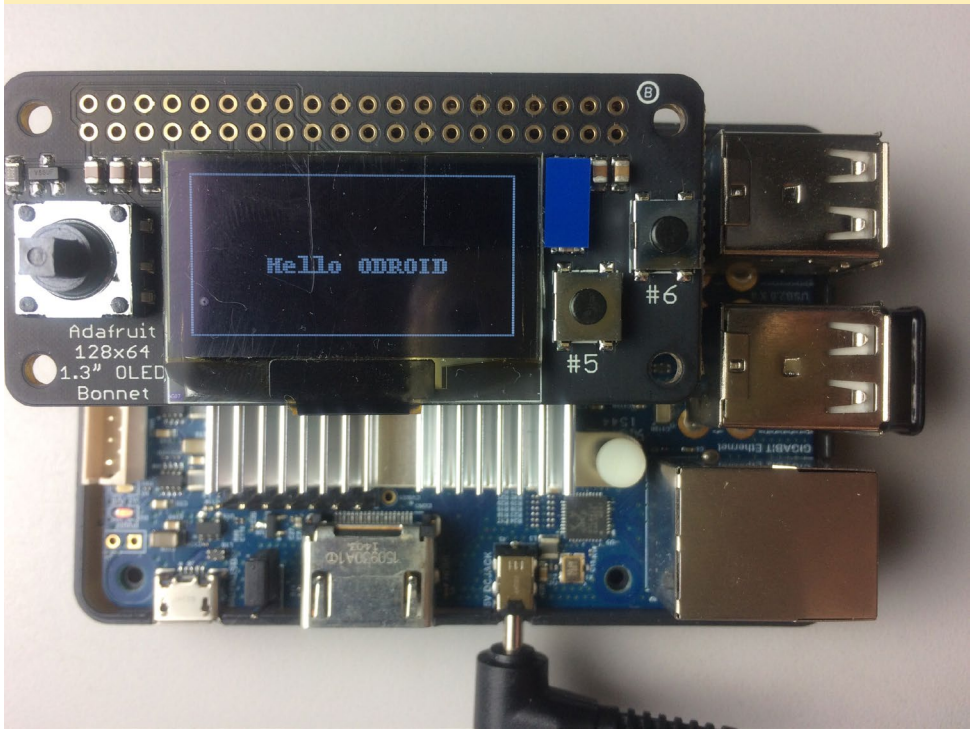
```
$ apt-get install git
```

Instala swig3.0 y python-dev:

```
$ apt-get install python-dev
python-setuptools
$ apt-get install swig3.0
```

Consigue y compila el wiringPi 2 de HardKernel para Python:

Figura 2 - Proyecto Demo WiringPi ejecutandose en un ODROID-C1+



CONTROLAR EL LED DEL ODROID-XU4

editado por Rob Roy

Anteriormente presentamos un artículo sobre cómo controlar los LEDs en el ODROID-U3 (http://bit.ly/2vpLLNj). Puedes tener un control similar del LED azul del XU4 usando un nodo de activación en el directorio /sys. El LED rojo está conectado por cable a la entrada de alimentación y por lo tanto no se puede controlar a través de software.

1. Apagar el LED azul:

```
$ su -
$ echo none > /sys/class/leds/
blue\:heartbeat/trigger
```

2. Encender el LED azul:

```
$ su -
$ echo default-on > /sys/
class/leds/blue\:heartbeat/
trigger
```

3. Activar el parpadeo del LED azul (configuración de fábrica):

```
$ su -
$ echo heartbeat > /sys/class/
leds/blue\:heartbeat/trigger
```

Hay muchos modos de activación, aunque no todos están habilitados. Puedes hacer una relación de todos ellos usando el siguiente comando:

```
$ cat /sys/class/leds/
blue\:heartbeat/trigger
```

Para que el cambio tenga efecto con cada arranque, agrega una línea a /etc/rc.local. Por ejemplo, para desactivar permanentemente el LED azul, añade el siguiente fragmento a /etc/rc.local:

```
echo none > /sys/class/leds/
blue\:heartbeat/trigger
```

```
$ git clone https://github.com/
hardkernel/WiringPi2-Python.git
$ cd WiringPi2-Python
$ git submodule init
$ git submodule update
$ swig3.0 -python -threads
wiringpi.i
$ python setup.py build install
```

Por el método de ensayo y error, descubrí los números pin correctos para los botones y el joystick del Adafruit 3531. Son muy diferentes de los números pin proporcionados por Adafruit para usarse con wiringPi en la Raspberry Pi.

Aquí tienes el código fuente para la demostración de los botones, el cual debe guardarse como hello.py:

```
from luma.core.interface.serial
import i2c, spi
from luma.core.render import
canvas
from luma.oled.device import
ssd1306, ssd1325, ssd1331, sh1106
import wiringpi2 as wpi
import time

# rev.1 users set port=0
# substitute spi(device=0,
port=0) below if using that in-
terface
serial = i2c(port=1,
address=0x3C)

# substitute ssd1331(...) or
sh1106(...) below if using that
device
device = ssd1306(serial)

with canvas(device) as draw:
    draw.rectangle(device.
bounding_box, outline="white",
fill="black")
    draw.text((30, 30), "Hello
ODROID", fill="white")

INPUT = 0
PUD_UP = 2
L_pin = 2
R_pin = 4
C_pin = 7
U_pin = 0
D_pin = 3
A_pin = 21
B_pin = 22
buttons = [0, 2, 3, 4, 7, 21, 22]

wpi.wiringPiSetup()

for x in buttons:
      wpi.pinMode(x, INPUT)
      wpi.pullUpDnControl(x, PUD_
UP)
      wpi.digitalWrite(x, 0)

while True:
      time.sleep(0.05)

      for x in buttons:
            print "x: %d state:
%d" % (x, wpi.digitalRead(x))
```

Presionando y manteniendo los botones cambiarán el estado de 1 a 0. La siguiente resultado aparece con el joystick inclinado a la derecha:

```
x: 0 state: 1
x: 2 state: 1
x: 3 state: 1
x: 4 state: 0
x: 7 state: 1
x: 21 state: 1
x: 22 state: 1
```

¡El resultado anterior verifica que el código funciona! Los botones y los números pin de joystick aparecen en el código fuente anterior.

MINERIA DE MONEDAS CRIPTO

UN PROYECTO DE VIABILIDAD Y UNA PRUEBA DE ESTABILIDAD PARA EL KERNEL VERSION 4.9 EN UN CLUSTER INFORMATICO DE ALTO RENDIMIENTO CON ODROID-XU4

editado por Rob Roy

Hardkernel usó la minería de monedas cripto para probar la estabilidad del Kernel 4.9.27 del XU4 a principios de este año. Veinte placas XU4, configuradas como un clúster de supercomputación, han estado ejecutando el software de explotación de moneda Verium (VRM) para utilizar todos los núcleos de las 160 CPU y 40GB de RAM tanto como era posible. Tras dos semanas de intensas pruebas, Hardkernel puede decir con seguridad que el Kernel 4.9 LTS en XU4 es bastante estable. Echa un vistazo al vídeo de demostración en https://youtu.be/wbffhvn_J4E.

Tuvimos que reiniciarlos todos dos veces para actualizar el Kernel cuando se liberaron las versiones 4.9.28 y 4.9.30. Algunas unidades tuvieron que resetearse manualmente debido al problema con el cable de alimentación 5V, así que añadimos algunos cables de alimentación más para reforzar los rieles de potencia. Aparte de eso, no hubo más problemas.

El coste de la puesta en funcionamiento del clúster se detalla a continuación, con un total de aproximado de 1500$:

20 x XU4 = 1,180$
20 x 8G tarjetas microSD = 160$
20 x cables LAN = 10$
1 x PSU 5V/80A (`http://amzn.to/2vkKjus`) = 45$
1 x switch de 24 puertos (`http://amzn.to/2vfKQyc`) = 41$
Cables, enchufes y separadores PCB = 30$
Medidor de potencia eléctrica opcional = 30$

Medimos la potencia y su potencia acumulada era de unos 65KWh y el consumo de energía mensual podría estar en unos 130 KWh. El coste estimado de electricidad al mes estaría en torno a los 15$. Ganamos sólo 100 VRM en dos semanas. Si tenemos en cuenta las comisiones para cambiar VRM a BTC y a la moneda estadounidense, su valor real es de 90$ en este momento. Nuestros ingresos mensuales estimados serian de unos 90$ x 2 – 15$ = 165$. Tendríamos que tener encendido este equipo de prueba durante unos 9 meses para conseguir el retorno de la inversión (ROI).

También conocimos cómo de rápido es el XU4 frente a la Raspberry Pi 3. El promedio de hash (Hashes por minuto) en XU4 es de 390H/min, mientras que en la RPi3 es de sólo 110H/min, lo que significa que la potencia de cálculo del XU4 es 3.5 veces mayor que la de la RPi3. Para obtener más información sobre la configuración del software de minería y los algoritmos internos, consulta los siguientes artículos: *CPU Mining is back! A complete how to guide and profit analysis for Verium mining on a farm of single board computers - Part 1*: `https://goo.gl/XM5ype`, *Part 2a*: `https://goo.gl/c38MWj`, *Part 2b*: `https://goo.gl/4Gi8ax`

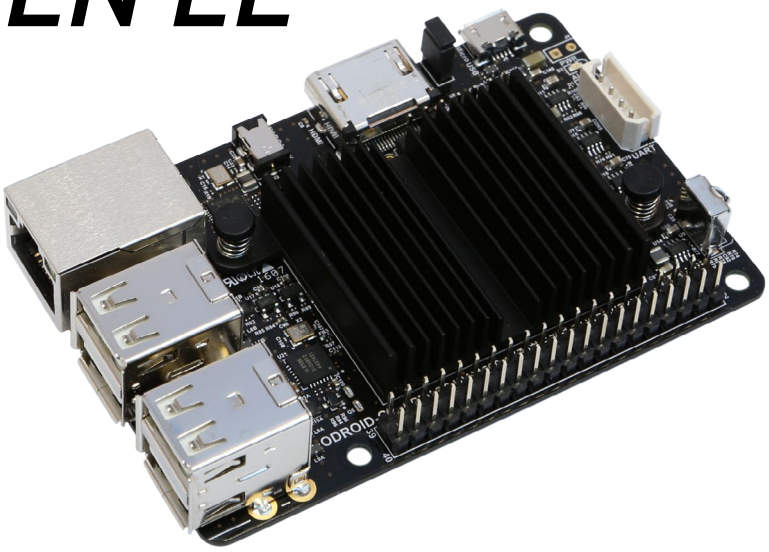
Wallet download: `http://www.vericoin.info`
Miner: `https://github.com/effectsToCause/veriumMiner`

Para comentarios, preguntas y sugerencias, visita `http://bit.ly/2uh0ncy`.

PRIMEROS PASOS CON ANDROID EN EL ODROID-C2

UNA GUIA PARA PRINCIPIANTES

editado por Rob Roy

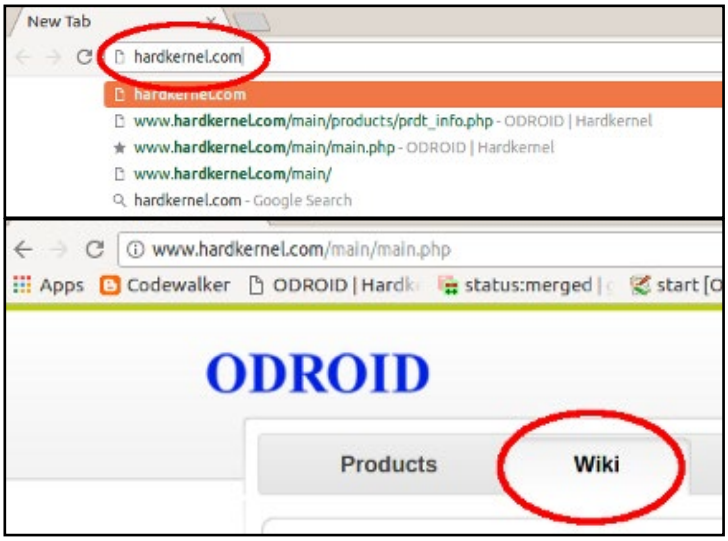


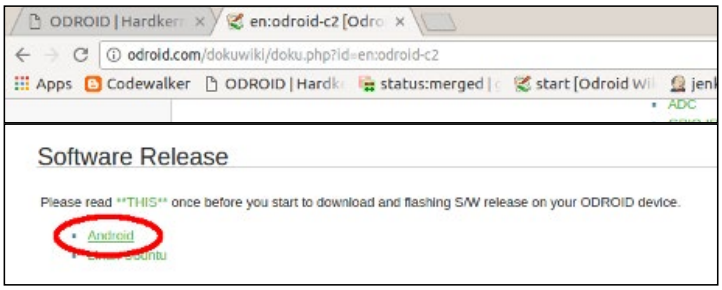
Existen dos opciones para instalar Android en un ODROID-C2. Hardkernel ofrece una tarjeta eMMC o microSD preinstalada, que sólo necesita instalar Google Play. Otra posibilidad para tener el sistema operativo Android es descargárselo desde el sitio web de Hardkernel e instalado manualmente en la tarjeta eMMC o microSD. Los materiales necesarios para ejecutar Android en un ODROID-C2 se enumeran a continuación:

- **ODROID-C2** (`http://bit.ly/1oTJBya`)
- **Fuente de Alimentación 5V/2A (US:** `http://bit.ly/2ugY0Xe`**, EU:** `http://bit.ly/1X0bgdt`**, Internacional:** `http://bit.ly/OhMyWx`)
- **Tarjeta de memoria con un sistema operativo pre-instalado (eMMC:** `http://bit.ly/2vq2TCq`**, tarjeta microSD:** `http://bit.ly/2u1fM5I`)
- **Cable HDMI:** `http://bit.ly/2uSu3Ay`
- **Monitor o TV con un puerto HDMI**

¡Echa un vistazo al video https://youtu.be/fEyeMTS3idU para ver lo fácil que es empezar! Si no tiene una tarjeta de memoria con un sistema operativo preinstalado, sigue las siguientes instrucciones para instalarlo en la tarjeta de memoria.

Además de todos los elementos ya mencionados, necesitarás



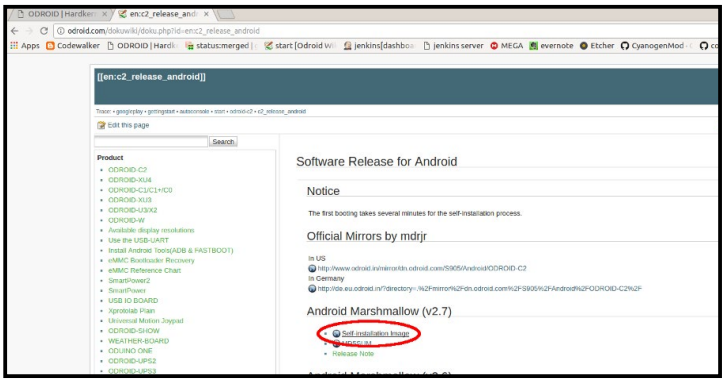


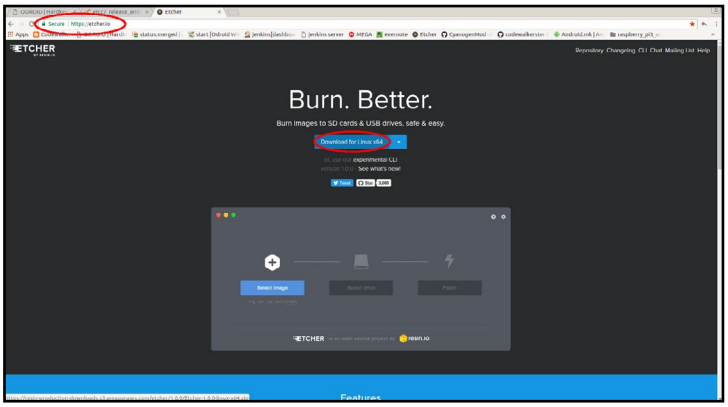
un PC para instalar el sistema operativo Android en la tarjeta de memoria. Tienes un video con instrucciones en `https://youtu.be/9Zi2_OTSl_I` y `https://youtu.be/NyQif1j2WkA`. Ten en cuenta que en este video se ha utilizado la fuente de alimentación Smart Power 2 (`http://bit.ly/2j3hhcv`).

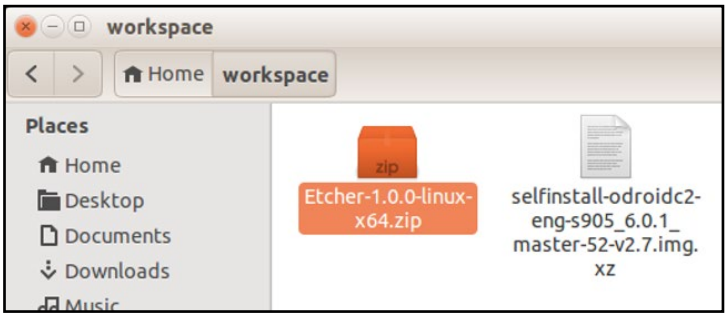
Primero, descarga el sistema operativo Android desde el sitio web de Hardkernel en `http://bit.ly/2vkGwgX`. Espera a que se complete la descarga. Para instalar Android en la tarjeta de memoria, te recomendamos usar Etcher, tal y como se describe en `http://bit.ly/2f61k5x`. Puedes descargar etcher desde `https://etcher.io/`. Etcher funciona en Mac OS, Linux y Windows, y es la opción más simple para la mayoría de los usuarios. Etcher también permite escribir las imágenes del sistema operativo directamente desde el archivo zip, sin necesidad de descomprimir. Para instalar el sistema operativo en un módulo eMMC, necesitarás un lector de módulos eMMC (`http://bit.ly/2ugIKK8`) y un lector USB (`http://bit.ly/2vpTv1y`) para conectarlo a tu PC.

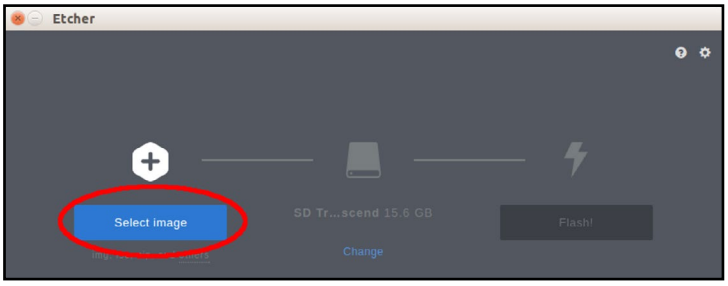
Para instalar Android en un eMMC, sigue el video con instrucciones de `https://youtu.be/XfJY4KxLxps`. Si utilizas una tarjeta microSD, visualiza `https://youtu.be/SnrqyoUBry4`.

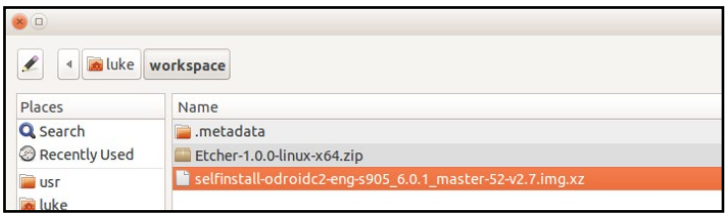
Cuando la instalación del SO haya finalizado en la tarjeta de memoria, conecta el cable HDMI a tu ODROID-C2, luego conecta la fuente de alimentación. Transcurridos unos segun-

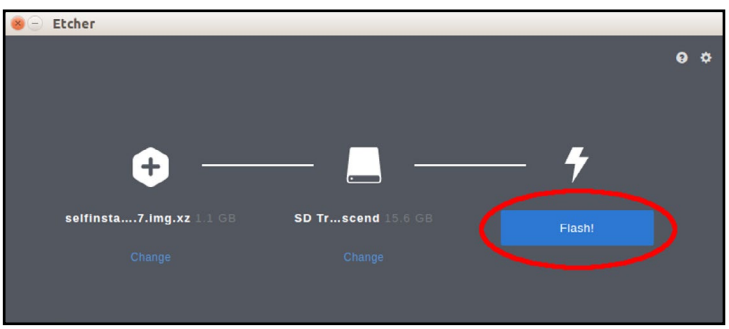












dos, verás la pantalla de inicio de Android. Para más información, visita el artículo de la Wiki `http://bit.ly/2uhhlrj`.

Instalación de Google Play

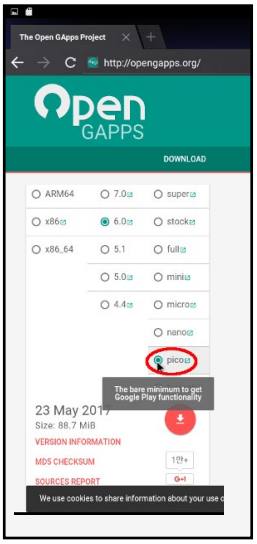
Para instalar Google Play en un ODROID-C2, son necesarios los siguientes elementos:

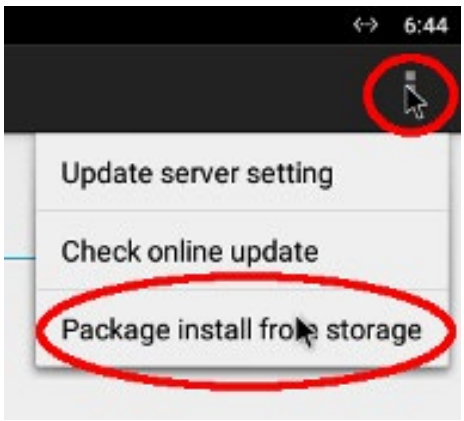
ODROID-C2 (`http://bit.ly/1oTJBya`)

Conexión a Internet vía Cable Ethernet (`http://bit.ly/2vg6v9I`) **o con un módulo Wifi** (`http://bit.ly/22nyxra`)

Si deseas descargar Google Play a un PC y transferirlo al C2, deberás conectar el C2 al PC mediante un cable OTG (`http://bit.ly/2vqf6H5`).

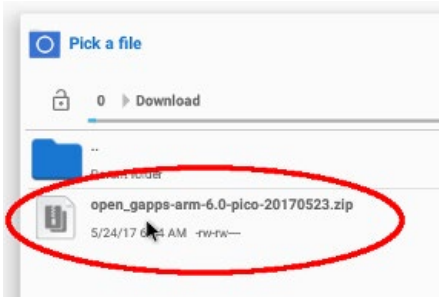
Tienes un video con instrucciones en `https://youtu.be/PKO8ZKJM_0c`. Las siguientes imágenes resaltan los pasos principales del vídeo. Abre el navegador en el ODROID-C2 y visita `http://opengapps.org`. Recomendamos usar la versión "pico", pero el ODROID-C2 también es compatible con versiones micro y nano.





El video en `https://youtu.be/wOhAgkkWnjI` muestra cómo iniciar sesión en tu cuenta de Google y abrir Google Play

Para obtener más información, visita el artículo original de la Wiki en `http://bit.ly/2vqgz0c`.



CONOCIENDO UN ODROIDIAN

MARTÍ BONAMUSA, EMPRENDEDOR DE VACIADO DE DATOS 3D EN TIEMPO REAL

editado por Rob Roy (@robroy)

Háblanos un poco sobre ti

Tengo 39 años y nací en Sant Celoni, una ciudad a 50km al norte de Barcelona en Cataluña. Estoy casado con mi esposa Fiona y tengo 2 hijos, Joan y Ferran, que tienes 8 y 6 años. Todos vivimos en Santa María de Palautordera, un pequeño pueblo junto a Sant Celoni. Fiona es profesora en una escuela de estudiantes con diferentes niveles de discapacidad.

Desde muy joven, he sentido una gran pasión por la electrónica y la programación, y una vez que terminé la escuela básica, decidí estudiar electrónica y me licencié en Ingeniería Electrónica por la Universidad Politécnica de Catalunya en

La Familia de Martí: Fiona, Joan y Ferran

2001. Más tarde realice un grado superior en ingeniería electrónica de la Universidad Autónoma de Barcelona en 2004.

Como resultado de las diferentes experiencias en el campo de la visión automatizada, fundé la compañía OnTrace (www.on-trace.com) con Josep Mesado, un colega de la Universidad. La razón por la que creamos la empresa fue que detectamos una necesidad de mercado relacionada con el análisis del comportamiento de la gente en espacios físicos. Teniendo en cuenta la alta precisión y el funcionamiento en diferentes entornos como factores principales, diseñamos una cámara estereoscópica 3D. Fue entonces cuando empezamos a trabajar con los dispositivos informáticos ODROID como un componente de nuestras cámaras inteligentes.

¿Cómo empezaste con los ordenadores?

Mi pasión por la electrónica empezó a una edad muy temprana, a los 9 años ya empezaba a hacer mis primeros inventos con un juego llamado "Scatron". Cuando tenía 12 años, mis padres me regalaron un Spectrum ZX para jugar, el cual me permitió empezar a programar. Para cuando tenía 15 años, ya había hecho varios programas en BASIC.

A los 16 años, cuando empecé a estudiar electrónica, descubrí los microcontroladores y comencé a jugar con ellos, especialmente con el PIC y 8031, programar en ensamblador era algo fascinante para mí.

El juego de Scatron hizo que Martí empezara con la electrónica

¿Qué te atrajo a la plataforma ODROID?

Empecé a buscar plataformas embebidas, centrándome principalmente en el bajo consumo de energía y el alto rendimiento, y aquí es donde descubrí la plataforma ODROID. Primero probamos el X2 y llegamos a la conclusión de que se ajustaba a nuestras necesidades.

Martí reparando un televisión en 2001

Evolucionamos del X2 al XU4, el cual ofrecía un mayor rendimiento, y ahora con el nuevo C2, nos permite realmente mejorar nuestros dispositivos y garantizar nuestras necesidades de mercado. El gran soporte y foros también son muy útiles e importantes para nosotros.

¿Cómo usas tus ODROIDs?

Actualmente tenemos varios productos diferentes, tres de ellos basados en ODROIDs como un dispositivo informático:

Un dispositivo de recuento de personas a partir de una cámara 3D y el XU4 (imagen: people counter3d.jpg).

Un dispositivo de reconocimiento facial usando el ODROID-C1.

Una nueva versión de cámara 3D con conectividad PoE que utiliza el ODROID-C2.

¿Cuál tu ODROID favorito y por qué?

He probado diferentes ODROIDs: el X2, U2, U3, XU4, C1 y C2, todos han dado buenos resultados, pero con el tuve mejor sensación y experiencia fue con el U3. Ahora estoy trabajando con el XU4, aparte de tener un funcionamiento muy bueno, pienso que el consumo de energía es excesivo para nuestros propósitos. Aunque es el modelo que estamos usando ahora, puede que no sea la mejor opción. Es la razón por la que actualmente estamos probando el nuevo C2 e intentando actualizar nuestros dispositivos para usarlo.

Arriba: Dispositivo 3D de reconteo de gente
Abajo: Grafico de reconteo de personas analizando un área ocupada

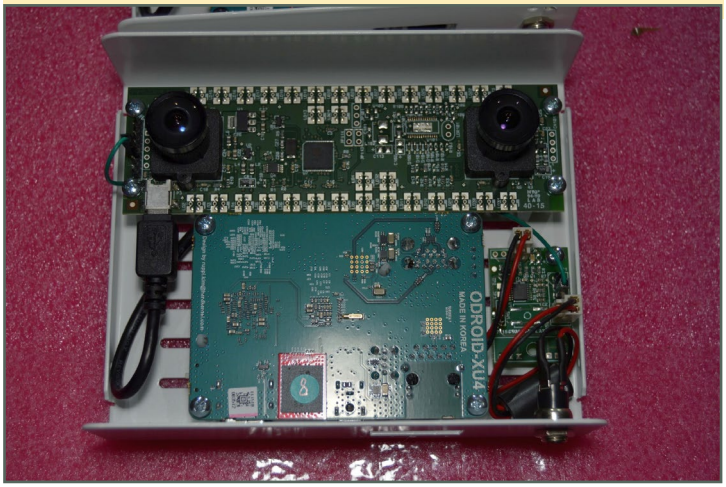
¿Qué innovaciones te gustaría ver en futuros productos de Hardkernel?

Siempre he echado menos la conectividad USB 2.0 en algunos de los conectores pin. También estaría bien tener WiFi interno, y en la placa C2, disponer de RTC en el propio dispositivo en lugar tener un módulo RTC Shield como el de ahora.

¿Qué aficiones e intereses tienes aparte de los ordenadores?

Me gustan los deportes en general, especialmente el fútbol y la Fórmula 1. Aún sigo jugando al fútbol en el club de veteranos de Sant Celoni. También me gusta hacer trekking con mi familia, para sacarle partido al esplendido lugar donde vivo.

Martí desarrollo estereos completamente funcionales usando un ODROID-X2 (arriba) y un ODROID-XU4 (abajo)



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

Más allá de la programación en ensamblador en la Universidad, nunca he realizado cursos de programación, todo lo he aprendido por mí mismo. Personalmente, lo que me hizo sentir pasión por la programación fue ver que escribía algunas sentencias y la máquina hacía exactamente lo que decía. El primer paso es sentir interés y curiosidad por la programación, el siguiente debe ser el deseo interno de ampliar y mejorar las funcionalidades del programa. Continúas buscando y probando, y al final todo se vuelve más fácil. Hoy en día, Internet lo pone todo al alcance de nuestras manos para hacer casi cualquier cosa. Se trata simplemente de buscar, filtrar y aplicar hasta encontrar la clave.