Volumio 2 • Depuración Android • Navega con un Mando IR remote

ODROID Magazine

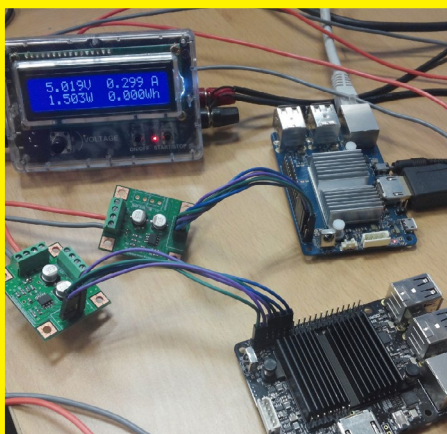
Año Cuatro
Num. #41
May 2017

Rediseña tu N64 Con la potencia de ODROID

Una completa guía que te permite usar la carcasa de tu clásica consola Nintendo con tu placa favorita

Ofrece Soporte Nativo para ODROID-C2

Explorando la Comunicación RS485 sobre el CI+ y C2



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

Por casualidad no Tendrás alguna vieja consola de juegos Nintendo o de otro tipo que ya no te funcione, ¡No la tires! Puede reacondicionarla con un **ODROID-XU4** que ejecute GameStation Turbo, RetroPie o Lakka y convertirla en un sistema emulador multi-plataforma con el que puede jugar a miles de juegos de diferentes consolas. Nuestra carta de presentación de este mes detalla cómo acoplarlo todo dentro de una cáscara N64, dándole nueva vida en una vieja carcasa de consola polvorienta.

Los **ODROIDs** son extremadamente versátiles y pueden ser utilizados para reproducir música, tal y como se describe en nuestro artículo **Volumio 2**, para desarrollar aplicaciones de Android, como Nanik demuestra en su artículo sobre el sistema de depuración de Android, y para el control de procesos, como lo demuestran Charles y Neal en su análisis sobre el protocolo de comunicación **RS485**. También contamos con una guía para configurar **RetroPie**, un sistema operativo de juegos, que ahora ofrece soporte nativo para **ODROID-C2** en la versión **4.2**. Adrian nos detalla cómo añadir más botones a tu **ODROID-C1** o **C2**, Lorenzo presenta su técnica para usar un mando a distancia por infrarrojos en Android, y Tobias continúa su columna de juegos de Linux con un vistazo a la popular serie de juegos de carreras **F Zero**.

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

¡Perdimos la cabeza, Otra vez!

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

Nicole es 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. Gestionando múltiples cuentas con agencias y productores de cine, desde Analytics y Adwords a la edición de vídeo y maquetación DVD, Nicole ayuda a sus clientes con todos los aspectos de la visibilidad online. Posee un ODROID-U2, varios ODROID-U3 y Xu4's, y espera poder utilizar las últimas tecnologías tanto para a nivel personal como empresarial. El sitio web de Nicole lo puedes encontrar en 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.

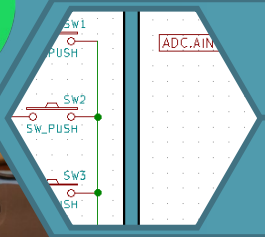
Andrew Ruggeri, Editor Adjunto

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.

INDICE



VOLUMIO 2 PLUGINS

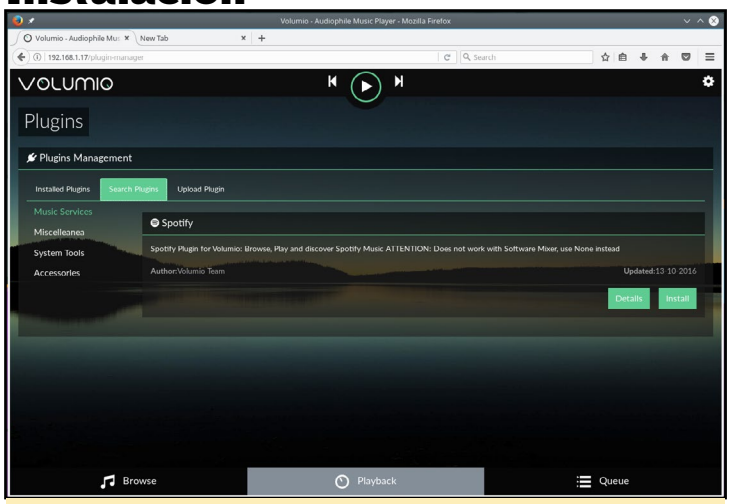
ESCUCHAR MUSICA CON SPOTIFY

por @synportack24

Si eres un apasionado de la música de alta fidelidad o simplemente te gusta escuchar música de vez en cuando, Volumio 2 es un gran SO con multitud de funciones que posiblemente sea lo que estás buscando. Hace casi un año, el pasado agosto para ser exactos, escribí un artículo sobre las nuevas y asombrosas características que Volumio 2 aporta a la familia ODROID. Una de esas características era la posibilidad para los desarrolladores de crear y añadir plugins con facilidad. En este artículo se describe cómo activar y ejecutar uno de mis plugins favoritos, Spotify.

Si no está familiarizado con Volumio 2, o si simplemente quieres que el SO básico arranque y se ejecute, te recomiendo que primero eche un vistazo al artículo original, que se encuentra en la página 26 del número de agosto de 2016 de ODROID Magazine en http://bit.ly/2pUExh8

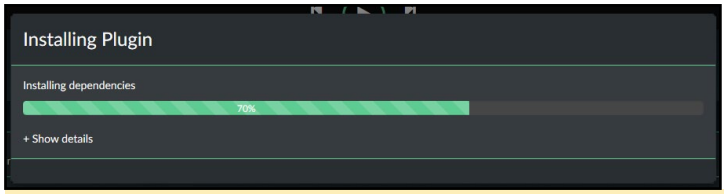
Instalación



Página de plugins de Volumio 2

La instalación es muy sencilla. Haz clic en el icono en forma de piñón situado en la parte superior derecha de la pantalla y selecciona “Plugins” en el menú lateral. Una vez cargada la página de plugins, selecciona la pestaña “Search Plugins”, seguido del filtro “Music Services” del lateral izquierdo, que debería ser la categoría por defecto. En una versión limpia de Volumio 2, Spotify será el único plugin que aparecerá en este filtro, aunque hay más disponibles desarrollados por terceros si lo deseas. Para instalar el plugins, simplemente haz clic en el botón “Install” en la parte inferior derecha del recuadro del plugin de Spotify.

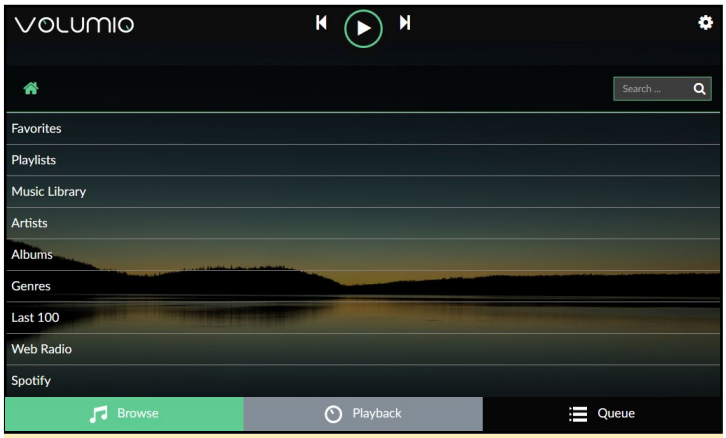
Una vez que hagas clic en el botón de instalación, Volumio mostrará una ventaba popup con una barra de progreso de la instalación. En mi caso, hicieron falta varios minutos, ya que Volumio necesita descargar e instalar varias dependencias. Tras la instalación, actualiza la página del navegador y podrás ver que el plugin aparece bajo la pestaña “Installed Plugins”. Haz clic en el botón “on”, a continuación, pincha en el botón de configuración para introducir tu información de inicio de sesión de Spotify y guardarla. ¡Ya está listo para escuchar música!



Barra de progreso de la instalación del plugin Spotify

Reproducir

Ahora, cuando vayas a las opciones de reproducción en la parte inferior de la lista, verás una opción para “Spotify”. Haz clic en ésta y podrás explorar tus listas de reproducción, así como el resto de emisoras que suelen aparecer en Spotify.



La opción de reproducción Spotify al final de la lista

Para obtener más información, visita el sitio web de Volumio 2 en www.volumio.org.

GESTOR DE BOTONES MULTICLIC PARA LCD DE 3.5” Y WEBCAM

SACALE EL MAXIMO RENDIMIENTO A LOS BOTONES HARDWARE

por Adrian Popa

Una cosa que realmente me gustó del módulo de pantalla táctil de 3.5” ODROID fue la inclusión de 4 interruptores hardware que se pueden programar para hacer casi cualquier cosa. Pero en mi opinión tienen un defecto: Ojalá hubiera más botones disponibles. ¿Se puede hacer algo al respecto?

Para empezar a utilizar la pantalla táctil de 3,5”, lo mejor que encontré fue la guía de @fourdee disponible en http://bit.ly/2oC7mdw. Ésta te puede ayudar a configurar la pantalla táctil, pero no incluye indicaciones de cómo utilizar los botones. Aunque Hardkernel proporciona un código de ejemplo (http://bit.ly/2pUogZy) para convertir los 4 botones en un teclado virtual que permite simular 4 pulsaciones de teclas, nosotros no lo vamos autilizar en este artículo.

Mi primera idea para intentar sacarle el máximo partido a estos botones era ver si podía hacer clic en 2 botones simultáneamente y generar un evento diferente. Los botones funcionan creando un “cortocircuito” entre el pin ADC y la puesta a tierra a través de una o más resistencias. Basándonos en el valor de la resistencia resultante, el ADC produce un valor numérico que te ayuda a identificar el botón que se ha presionado.

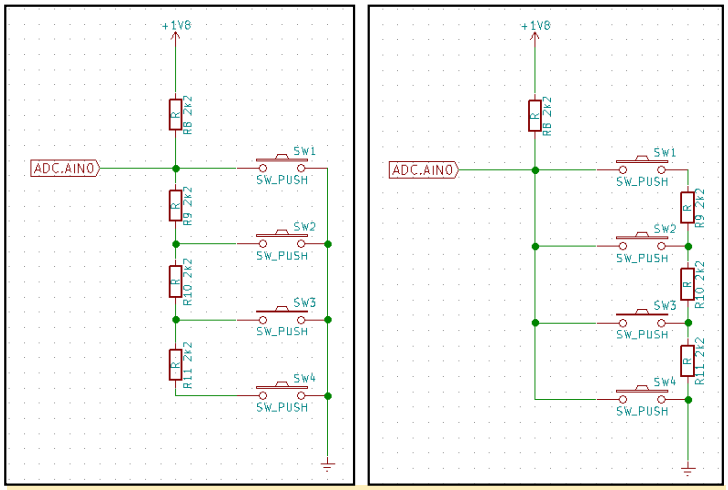


Figura I (a) Diseño original HK y I (b) alternativa mejorada

La Figura 1a muestra el diseño de Hardkernel, mientras que la Figura 1b muestra una pequeña modificación la cual permite añadir 3 “eventos” más a estos botones. Si las resistencias estuvieran montadas más cerca de la puesta a tierra, en lugar de estar antes del interruptor, se podría crear un circuito paralelo al presionar dos botones y tener un valor ADC diferente. Me puse con las matemáticas (me sorprendió que aún recordase lo aprendido en la universidad después de 10 años) y lo logué en un ODROID-C1/C2, los valores que se muestran en la Figura 2 han sido posibles con las resistencias invertidas.

Figura 2 - Valores ADC para múltiples botones (a modo de tabla)

SW4	5 ±10
SW3	515 ±10
SW2	680 ±10
SW1	770 ±10
SW1+SW3	438 ±10
SW2+SW3	408 ±10
SW1+SW2	558 ±10

Un problema que puede surgir con esta técnica es el hecho de obtener “clics erróneos” si no se presionan simultáneamente los botones, pero como Hardkernel no implementó el hardware de este modo, estudiamos otras opciones.

Un modo distinto de manejar eventos con teclas es gestionar múltiples pulsaciones de teclas como un único evento. Por ejemplo, podría manejar un evento de “doble/triple clic” o incluso una “pulsación prolongada” y hacer que ejecute algo diferente. La forma con la que puedes detectar un evento de este tipo es configurando un búfer, se graba la pulsación de la techa en el búfer y cuando éste esté lleno, se procesa el evento.

Ya que parecía ser un proyecto simple y divertido, me propuse ampliar el código C de Hardkernel que gestiona los botones y añadí esta funcionalidad. Sin embargo, habiendo estado tiempo alejado del lenguaje C, decidí hacerlo mejor en Perl.

El código funciona del siguiente modo: Hay un bucle infinito que lee la entrada ADC. Una vez que detecta un cambio (debido a la pulsación de una tecla), el valor se registra en el búfer. Las lecturas sucesivas también se almacenan en el búfer (incluso cuando no se pulsa ninguna tecla) y cuando el búfer se llena, se identifica la secuencia de teclas y se lleva a cabo alguna acción. Este método tiene casualmente el efecto secundario

de que añade soporte para secuencias de teclas como "KEY1-KEY2" y "KEY1-KEY2-KEY3".

Para conseguir el código y configurarlo, sigue estos pasos:

```
$ git clone https://github.com/mad-ady/tftlcd35-key.
git
$ cd tftlcd35-key
$ sudo apt-get install liblog-log4perl-perl \
libproc-background-perl libconfig-yaml-perl
$ sudo cp tftlcd35_key.pl /usr/local/bin
$ sudo cp tftlcd35-key.service /etc/systemd/system/
```

El código viene con ejemplos de configuración para que los uses según tus necesidades. Si sólo quieres un clic normal y un clic largo por tecla, utiliza config-empty-1.yaml. Si desea usar 2 combinaciones de teclas, usa config-empty-2.yaml o si deseas 3 combinaciones de teclas usa config-empty-3.yaml. La sintaxis de configuración está en formato YAML simple. Al igual que en la sintaxis de Python, el sangrado se tiene en cuenta y no se debe usar la tabulación para crear espacios en blanco. Si tras iniciar el script, aparecen anotaciones sobre la configuración, puede utilizar un validador online como http://www.yamllint.com/ para identificar el problema. La configuración por defecto fija el registro de datos en el nivel INFO, que determina el período entre las lecturas a 200 ms, la longitud del búfer a 10 y el intervalo de compresión al 70% de la longitud del búfer. Si tiene problemas, puedes fijar el registro a nivel DEBUG, pero es muy detallado. Para entender cómo afectan estos parámetros, analizaremos el hipotético buffer que se muestra en la Figura 3.

KEY1	KEY1		KEY2	KEY2	KEY2		KEY3		

Figura 3 – Buffer de ejemplo

Cada celda representa un valor de lectura ADC. El búfer tiene 10 celdas, así que el bufferSize (tamaño del buffer) tiene que ser 10. Dependiendo de lo rápido que presiones y sueltes una tecla, puedes ajustar el updatePeriod (Periodo de actualización) más alto o más bajo (por defecto 200000 microsegundos o 200ms). Si fijas el updatePeriod demasiado alto (por ejemplo, 500ms), perderás pulsaciones de teclas porque puedes presionar y soltar el botón cuando el script esté en modo reposo. Si ajustas el updatePeriod demasiado bajo, puede que necesites un búfer mayor para grabar todas las pulsaciones. Por ej., si logras el búfer que aparece en la Figura 3, pero has hecho doble clic en KEY1, significa que tu updatePeriod es demasiado alto.

Desde el momento en que pulsa una tecla, el script tendrá updatePeriod * bufferSize microsegundos para reaccionar. Esto significa que tener un alto bufferSize te dará un mayor tiempo de reacción (presionaste el botón una vez, pero la acción tiene lugar 5 segundos más tarde). La configuración por defecto utiliza un tiempo de reacción de 2 segundos (200ms * 10).

El parámetro final es longPress que representa el número de ítems en el búfer que puede ser una determinada tecla antes de que el búfer sea interpretado como una pulsación larga. El valor por defecto es 0,7, lo que significa que el 70% del búfer tiene que ser rellenado con una pulsación de tecla (en nuestro ejemplo significaría mantener pulsada una tecla al menos 1,4 s). Tenga en cuenta que si mantienes presionada la tecla demasiado tiempo (por ejemplo, 2,2 s) y el tamaño del búfer se excede, el programa interpretará los 2 primeros segundos como una pulsación larga generando un determinado evento y los 0,2 segundos restantes activarán otro evento diferente. Esta cuestión se puede mitigar en el propio código introduciendo una pausa corta después de un evento largo, si fuera necesario.

La parte final de la configuración es clave para la asignación de los comandos. Las secuencias de teclas están separadas por un guión (-) y las pulsaciones largas se identifican con el prefijo LONG. En el ejemplo anterior la secuencia de teclas interpretada por el script es KEY1-KEY2-KEY3 (las teclas duplicadas son ignoradas). Para conseguir una secuencia KEY1-KEY1-KEY1 necesitas soltar KEY1 durante al menos un updatePeriod para que el buffer tenga una lectura en blanco entre dos teclas.

Para asignar un comando a una secuencia, simplemente escribe el comando que quieres ejecutar en la misma línea después del signo ':'. Los comandos se ejecutan en un intérprete de comandos en segundo plano con el mismo usuario que el script (root). Si necesita ejecutar un comando gráfico, utiliza el prefijo DISPLAY=:0

```
logging: DEBUG
updatePeriod: 200000
bufferSize: 10
longPress: 0.7
# Example commands (one line each):
#KEY1: logger "Key1 has been pressed"
#KEY2: logger "Key2 has been pressed"; logger "Two commands have been executed"
#KEY3: su -u odroid -c "logger 'This command is run as a different user (uid $EUID)'"
#KEY4: DISPLAY=:0 xeyes

KEY1: logger "Key1 has been pressed"
KEY1-KEY1: logger "Key1 has been double-pressed"
KEY1-KEY2: logger "Key1-Key2 has been pressed"
KEY1-KEY3: logger "Key1-Key3 has been pressed"
KEY1-KEY4: logger "Key1-Key4 has been pressed"
LONGKEY1: logger "Key1 has been long-pressed"
KEY2:
KEY2-KEY1:
KEY2-KEY2: logger "Key2 has been double-pressed"
KEY2-KEY3:
KEY2-KEY4:
LONGKEY2:
KEY3:
KEY3-KEY1:
KEY3-KEY2:
KEY3-KEY3: logger "Key3 has been double-pressed"
KEY3-KEY4:
LONGKEY3:
KEY4:
KEY4-KEY1:
KEY4-KEY2:
KEY4-KEY3:
KEY4-KEY4: logger "Key4 has been double-pressed"
```

Figura 4 - Ejemplo de configuración

Finalizada la configuración, cópiala en etc/tftlcd35-key.yaml, activa el script y haz que se inicie automáticamente:

```
$ sudo cp custom-config.yaml /etc/tftlcd35-key.yaml
$ sudo systemctl enable tftlcd35-key
$ sudo systemctl start tftlcd35-key
```

La información de registro y depuración va a syslog y se puede ver con el siguiente comando:

```
$ sudo journalctl -f -u tftlcd35-key
```

Usar el botón de un dispositivo

Si no tiene la pantalla de 3.5" de Hardkernel, pero dispones de dispositivos con botones físicos (como por ejemplo, una cámara o una tarjeta de sonido), aún puedes hacer varias cosas con ellos de un modo similar. Por ejemplo, la cámara Hardkernel 720p viene con un botón para tomar las instantáneas, que rara vez se utiliza. El botón actúa a modo de dispositivo de entrada (como si fuese un teclado estándar) y registra un único botón como un evento. Puedes averiguar qué teclas soporta con el comando evtest, tal y como se muestra en la Figura 5.

```
$ sudo apt-get install evtest
```

```
adrianp@kbd:~$ evtest
No device specified, trying to scan all of /dev/input/event*
Not running as root, no devices may be available.
Available devices:
/dev/input/event0:      cec_input
/dev/input/event1:      vt-input
/dev/input/event2:      Lenovo Optical Mouse
/dev/input/event3:      Sun USB Keyboard
/dev/input/event4:      SX865X Touchscreen
/dev/input/event5:      USB 2.0 Camera
/dev/input/event6:      meson-ir
/dev/input/event7:      MCE IR Keyboard/Mouse (meson-ir)
Select the device event number [0-7]: 5
Input driver version is 1.0.1
Input device ID: bus 0x3 vendor 0x1b71 product 0x56 version 0x0
Input device name: "USB 2.0 Camera"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 212 (KEY_CAMERA)
Properties:
Testing ... (interrupt to exit)
Event: time 1490777760.449861, type 1 (EV_KEY), code 212 (KEY_CAMERA), value 1
Event: time 1490777760.449861, -------------- SYN_REPORT ------------
Event: time 1490777760.597888, type 1 (EV_KEY), code 212 (KEY_CAMERA), value 0
Event: time 1490777760.597888, -------------- SYN_REPORT ------------
```

Figura 5 – Ejemplo de resultado de evtest

Los dispositivos de entrada son asignados en /dev/input/event*, aunque la asignación es dinámica, de modo que no puede depender de los números de los reinicios. En su lugar, tienes que identificar tu dispositivo desde /dev/input/by-id/ lo cual es más estable, tal y como muestra la Figura 6.

```
adrianp@kbd:~$ ls -l /dev/input/by-id/
total 0
lrwxrwxrwx 1 root root 9 Feb 11  2016 usb-0430_Sun_USB_Keyboard-event-kbd -> ../event3
lrwxrwxrwx 1 root root 9 Feb 11  2016 usb-17ef_Lenovo_Optical_Mouse-event-mouse -> ../event2
lrwxrwxrwx 1 root root 9 Feb 11  2016 usb-17ef_Lenovo_Optical_Mouse-mouse -> ../mouse0
lrwxrwxrwx 1 root root 9 Feb 11  2016 usb-Sonix_Technology_Co.__Ltd._USB_2.0_Camera-event-if00 -> ../event5
```

Figura 6 - Asignación de entrada estable

Así que, para beneficiarte de los eventos de clic y multi-clic (pulsación doble/triple/larga) puedes descargar e instalar este programa gestor:

```
$ git clone https://github.com/mad-ady/multibutton.
git
$ cd multibutton
$ sudo perl -MCPAN -e 'install Linux::Input'
$ sudo apt-get install libconfig-yaml-perl liblog-log-
4perl-perl \
libproc-background-perl
$ sudo cp multibutton.pl /usr/local/bin
$ sudo cp config-minimal.yaml /etc/multibutton.yaml
$ sudo cp multibutton.service /etc/systemd/system
```

Antes de iniciar el servicio, necesitarás configurarlo. El primer paso es identificar tu dispositivo de entrada y proporcionar la ruta correcta hacia éste en /etc/systemd/system/multibutton.service (el parámetro -i), tal y como se ha descrito antes.

Después, modifica el archivo de configuración según tus necesidades. La sintaxis es la misma que el gestor de pantalla táctil, sin embargo, puesto que en mi caso sólo dispongo de 1 botón, únicamente puedo controlar una cosa. El archivo de configuración está en formato YAML y tiene las mismas 3 opciones de configuración: período de actualización, tamaño del búfer y porcentaje de pulsación larga. En este caso, updatePeriod no es tan importante, ya que el kernel realizará el sondeo y reenviará todos los eventos al usuario final (sin teclas perdidas).

Luego viene la secuencia de teclas en formato "KEY_LABEL1-KEY_LABEL2". Puedes ver las etiquetas correctas con el comando evtest. Siguiendo las etiquetas de las teclas, debes especificar los comandos que se van a ejecutar. La Figura 7 muestra un ejemplo (consulta el archivo real para los comentarios):

```
# Configuration sample for multibutton.pl

# Set logging level (logging goes to STDERR/journalctl). Allowed values are: DEBUG, INFO, WARN, ERROR, FATAL

logging: DEBUG

# Set updatePeriod to the number of microseconds between key presses checks
# Default is 200ms between key presses

updatePeriod: 200000

# Set bufferSize to how many keys to keep in memory when analizing a multiple key press.
# The larger the value, the more time you have to wait until any action executes

bufferSize: 10

# Set longPress to a fraction - how many times does a key have to appear in a sequence before considering it a long press.
# The default is 0.7 which means 70% of the keys in bufferSize have to be the same to register as a long press

longPress: 0.7

# The following section defines possible key sequences and an action that should be executed in the background
# Note that the command runs as the same user you run multibutton.pl by default

# Example commands (one line each):
#KEY_CAMERA: logger "Key has been pressed"
#KEY_CAMERA-KEY_CAMERA: logger "Key has been double pressed"; logger "Two commands have been executed"
#KEY_CAMERA-KEY_CAMERA-KEY_CAMERA: su -c "logger 'Tripleclick - This command is run as a different user (uid $EUID)'" odroid
#LONGKEY_CAMERA: DISPLAY=:0 xeyes

KEY_CAMERA: logger -s 'Pressed single'
KEY_CAMERA-KEY_CAMERA: logger -s 'Pressed double'
KEY_CAMERA-KEY_CAMERA-KEY_CAMERA: logger -s 'Pressed triple'
LONGKEY_CAMERA: logger -s 'Pressed long'
```

Figura 7 – Ejemplo de configuración multibutton.pl

Una vez que hayas montado tu configuración y el servicio systemd, puedes activar el servicio con:

```
$ sudo systemctl daemon-reload
$ sudo systemctl enable multibutton
$ sudo systemctl start multibutton
```

Puede ver los registros log con journalctl:

```
$ sudo journalctl -u multibutton -f
```

Si necesitas soportar varios dispositivos (por ejemplo, digamos 2 cámaras), puedes crear un nuevo servicio que apunte al nuevo dispositivo y cargar una configuración diferente.

Ten en cuenta que puedes ejecutar este gestor de eventos en cualquier dispositivo, incluido un teclado real, y sólo configurar determinadas pulsaciones de teclas para ejecutar varios comandos. Esto podría dar lugar a una interesante alternativa de pulsaciones de teclas en las que, por ejemplo, vinculas la "w" para ejecutar algún comando arbitrario cuando se presione tres veces. Ten presente que el código no ha sido probado para manejar múltiples teclas.

Para comentarios, preguntas, sugerencias o reportar errores, visita el hilo de soporte en http://bit.ly/2nFxvJh.

EMULADOR DE JUEGOS XU4

ES HORA DE REUTILIZAR LA VIEJA CONSOLA QUE COMPRASTE EN EL MERCADILLO

por @MarkT

He creado un emulador de juego N64 con el que puedo ejecutar juegos desde Atari hasta N64 y PS1. Yo mismo modelé todas las partes para que encajaran en la carcasa de un viejo e inutilizable sistema N64. He usado un XU4, el cual permite que el sistema se ejecute con hasta 4 jugadores simultáneos facilmente y sin problemas en comparación con una Pi 3.

He usado RecalBox como sistema operativo de emulación, que funciona muy bien, aunque cuenta con algunas opciones de configuración algo pobres para los mandos en lo que respecta a la Nintendo 64. A la hora de montarlo, mi mejor aliado definitivamente fue una pistola de pegamento que me permitió asegurar bien las conexiones, y que mis puestos USB estuvieran bien anclados. Aún sigo trabajando en el direccionamiento HDMI, así que, si alguien desea poner su granito de arena, estaré encantado de ver lo que tiene en mente. Este es sólo un boceto de mi proyecto, haré una guía mucho más detallada de los pasos que he seguido una vez que lo tengo más completo. ¡Por favor, no hagas de esto un sistema Nintendo 64! Me rompería el corazón ver que el sistema no funcione.

Materiales

- Un viejo sistema Nintendo 64 (preferiblemente roto) y su hardware (tornillos)
- ODROID-XU4 y fuente de alimentación
- Hub USB Slim de 4 puertos como éste, http://amzn.to/2papHjz
- 2 x Tornillos 1/2' 2-56
- Interruptor SPDT MOM-ON, como http://bit.ly/2pYldPS
- 2 x Tornillos 1/4' 4-40 con tuercas
- Tornillo 1/2' 4-40
- Cable AWG 28
- Soldador
- Pistola de pegamento
- LED azul con un resistencia de 10 ohmios
- Clavija RJ-45 de tamaño similar a esta http://bit.ly/2ooqkZM
- Una toma de corriente de tamaño (9mm x 15mm) similar a http://bit.ly/2oEhoe4
- Yo he utilizado los mandos Retrolink USB, que te hacen sentir como si estuvieras jugando al sistema original, aunque también puede utilizar casi cualquier otro mando USB, como Xbox o PS3.
-SO RecalBox/Batocera (http://bit.ly/2ooDtSz)
-SO Lakka (http://bit.ly/2ptdWHX)

Impresión 3D

Los archivos de impresión 3D se pueden descargar desde la página del proyecto Thingiverse en http://bit.ly/2pXcaf6. Le di la vuelta a los puertos USB (salidas apuntando hacia arriba) con la plataforma y el soporte, luego imprimí dos packs de juegos. Utilice la plataforma y el soporte para los botones/interruptores y la cubierta del pack de expansión.

Fabricante: MakerBot
Impresora: MakerBot Replicator
Plataformas: Sí
Soporte: Sí
Resolución: .20
Relleno: 30%

Figura 1 - Carcasa impresa en 3D, totalmente montada

Tal y como se muestra en la Figura 2, todo está colocado en su lugar. He utilizado el pegamento para fijar firmemente los terminales USB en las ranuras, y tiras de cierre para bajar la presión. Mi hub USB se encuentra debajo de la plataforma. También he conectado un LED con una resistencia de 10 ohmios en GPIO 11 y GND. He soldado el otro extremo de mi interruptor al mismo interruptor que está en la placa con los terminales correctos.

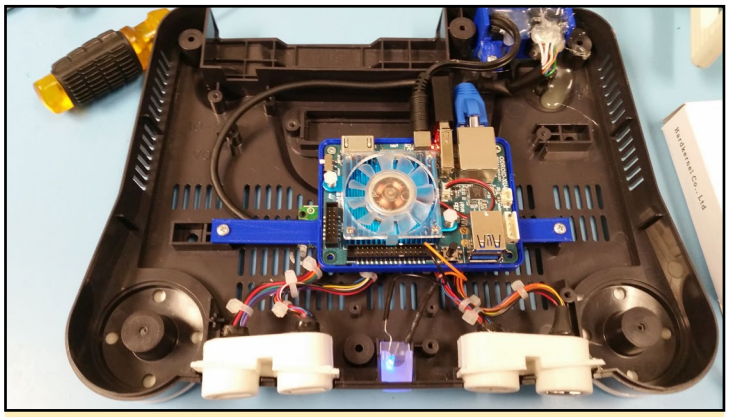


Figura 2 - Interior de la carcasa.

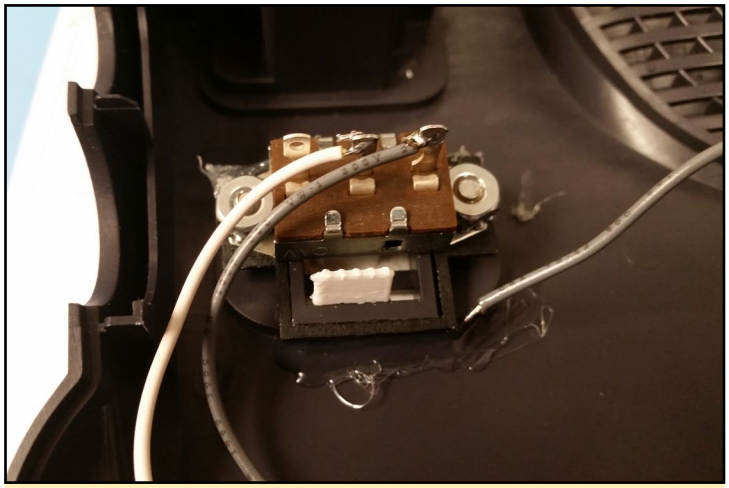


Figura 3 - El interruptor está atornillado a la impresión Switch_bottom y está pegado para una mayor sujeción, luego el cable blanco y gris están soldados al interruptor de la placa ODROID.

Figura 4 - Los conectores Ethernet y DC están soldados y pegados firmemente, asegurándonos de que la clavija DC está soldada a la polarización correcta antes de encolar.

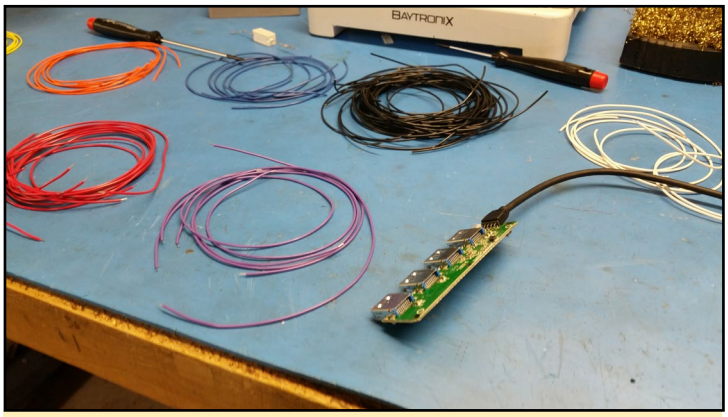


Figura 5 - Desoldar los puertos USB y utilizar cable AWG 28-30 para extender los puertos USB hasta las ranuras USB impresas. O comprar nuevos terminales USB, que son más fáciles de soldar.

Figura 6 – En la parte trasera de la consola se encuentra la toma Ethernet y la toma de alimentación.

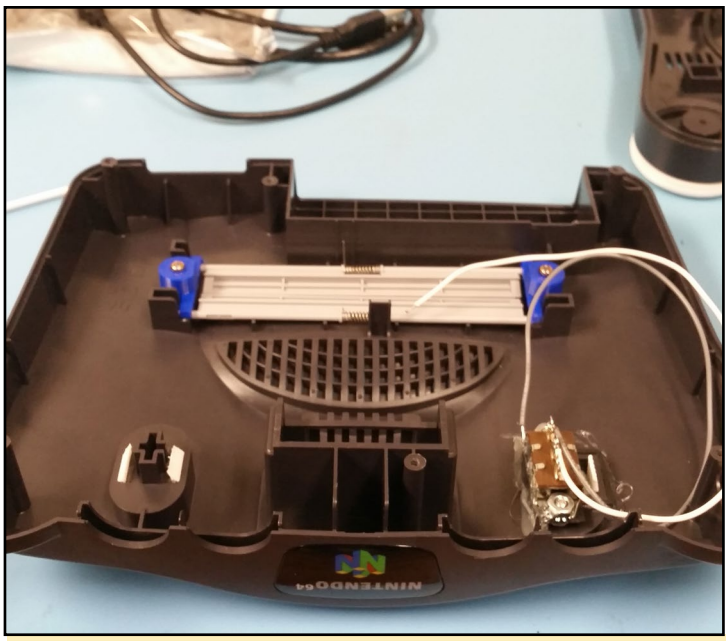


Figura 7 - Para la parte superior, imprimí el compartimiento para el pack del juego con el fin de tener más espacio dentro de la consola, manteniendo de esta forma la apariencia de la Nintendo 64.

UN INTENSO VIAJE POR ANDROID

DEBUG BRIDGE (ADB) – PARTE I

por Nanik Tolaram

Android Debug Bridge (ADB) es la única herramienta disponible para que los desarrolladores puedan comunicarse con sus dispositivos Android. ADB permite a los desarrolladores profundizar en Android permitiéndoles acceder al intérprete de línea de comandos, lo cual pone todo el mundo de Android al alcance de sus manos. Teniendo acceso al intérprete de línea de comandos, podemos echar un vistazo a los registros log, extraer datos de los dispositivos Android, instalar y desinstalar aplicaciones y mucho más.

Vamos a analizar en profundidad ADB y veremos cómo funciona internamente. El artículo está dividido en 2 partes. Este artículo es la primera parte de la serie, en el cual hablaremos de ADB de forma general, de cómo se desarrolla formando parte del proceso de compilación y su arquitectura. En la segunda parte veremos con más detalle las diferentes partes del código fuente de ADB y cómo funciona la magia de la comunicación con tu dispositivo.

ADB en L, M y N

Vamos a ver en primer lugar cómo de diferente es el código fuente de adb en las diferentes versión de Androide. Las Figuras 1, 2 y 3 muestran el código fuente de Lollipop, Marshmallow y Nougat, respectivamente. La aplicación adb se encuentra dentro de la carpeta <tu_directorio_android>/system/core/adb.

Observando las capturas de pantalla, podemos ver que ADB en Lollipop fue escrito en C, mientras que las otras 2 versiones fueron escritas en C++. El número de archivos ha aumentado en Nougat debido a nuevas funcionalidades, y se han hecho muchas refactorizaciones de las versiones anteriores para estructurar mejor el código.

En términos de objetos de compilación, nos interesa principalmente 2 binarios: adb y adbd, tal y como se muestra en la siguiente página.

Arquitectura ADB

Internamente, ADB está estructurado como 2 aplicaciones diferentes que se ejecutan como host y servidor. El "host" se ejecuta en tu ordenador o portátil, y el "servidor" se ejecuta en el dispositivo Android. Las aplicaciones se ejecutan como una pareja servidor

De izquierda a derecha: Codigo ADB de Lollipop/Marshmallow/Nougat

```
-rw-rw-r-- 1 nanik nanik  6163 Sep 26 15:38 adb_auth_client.c
-rw-rw-r-- 1 nanik nanik 10063 Sep 26 15:38 adb_auth_host.c
-rw-rw-r-- 1 nanik nanik 50093 Sep 26 15:38 adb.c
-rw-rw-r-- 1 nanik nanik  8591 Sep 26 15:38 adb_client.c
-rw-rw-r-- 1 nanik nanik 64066 Sep 26 15:38 commandline.c
-rw-rw-r-- 1 nanik nanik  1009 Sep 26 15:38 console.c
-rw-rw-r-- 1 nanik nanik  5817 Sep 26 15:38 disable_verity_service.c
-rw-rw-r-- 1 nanik nanik 17407 Sep 26 15:38 fdevent.c
-rw-rw-r-- 1 nanik nanik 25876 Sep 26 15:38 file_sync_client.c
-rw-rw-r-- 1 nanik nanik 11431 Sep 26 15:38 file_sync_service.c
-rw-rw-r-- 1 nanik nanik  5194 Sep 26 15:38 framebuffer_service.c
-rw-rw-r-- 1 nanik nanik  1079 Sep 26 15:38 get_my_path_darwin.c
-rw-rw-r-- 1 nanik nanik  1015 Sep 26 15:38 get_my_path_freebsd.c
-rw-rw-r-- 1 nanik nanik   962 Sep 26 15:38 get_my_path_linux.c
-rw-rw-r-- 1 nanik nanik   962 Sep 26 15:38 get_my_path_windows.c
-rw-rw-r-- 1 nanik nanik 20165 Sep 26 15:38 jdwp_service.c
-rw-rw-r-- 1 nanik nanik  4404 Sep 26 15:38 remount_service.c
-rw-rw-r-- 1 nanik nanik 19642 Sep 26 15:38 services.c
-rw-rw-r-- 1 nanik nanik 25552 Sep 26 15:38 sockets.c
-rw-rw-r-- 1 nanik nanik 61023 Sep 26 15:38 sysdeps_win32.c
-rw-rw-r-- 1 nanik nanik  2431 Sep 26 15:38 test_track_devices.c
-rw-rw-r-- 1 nanik nanik  2423 Sep 26 15:38 test_track_jdwp.c
-rw-rw-r-- 1 nanik nanik 32115 Sep 26 15:38 transport.c
-rw-rw-r-- 1 nanik nanik 13957 Sep 26 15:38 transport_local.c
-rw-rw-r-- 1 nanik nanik  3412 Sep 26 15:38 transport_usb.c
-rw-rw-r-- 1 nanik nanik 16037 Sep 26 15:38 usb_libusb.c
-rw-rw-r-- 1 nanik nanik 21485 Sep 26 15:38 usb_linux.c
-rw-rw-r-- 1 nanik nanik 13142 Sep 26 15:38 usb_linux_client.c
-rw-rw-r-- 1 nanik nanik 17641 Sep 26 15:38 usb_osx.c
-rwxrwxr-x 1 nanik nanik 10931 Sep 26 15:38 usb_vendors.c
-rw-rw-r-- 1 nanik nanik 14149 Sep 26 15:38 usb_windows.c
```

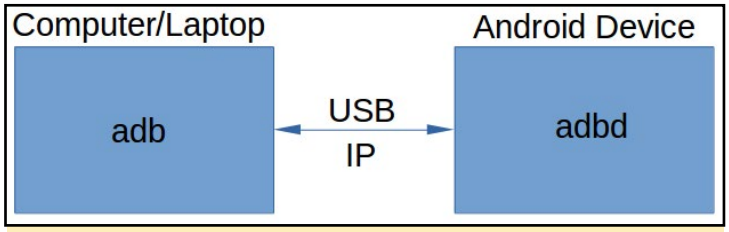
```
-rw-rw-r-- 1 nanik nanik  6382 Aug 21  2016 adb_auth_client.cpp
-rw-rw-r-- 1 nanik nanik  2277 Aug 21  2016 adb_auth.cpp
-rw-rw-r-- 1 nanik nanik 10714 Aug 21  2016 adb_auth_host.cpp
-rw-rw-r-- 1 nanik nanik  8928 Aug 21  2016 adb_client.cpp
-rw-rw-r-- 1 nanik nanik 29359 Aug 21  2016 adb.cpp
-rw-rw-r-- 1 nanik nanik  3228 Aug 21  2016 adb_io.cpp
-rw-rw-r-- 1 nanik nanik  4333 Aug 21  2016 adb_io_test.cpp
-rw-rw-r-- 1 nanik nanik  7729 Aug 21  2016 adb_listeners.cpp
-rw-rw-r-- 1 nanik nanik 11442 Aug 21  2016 adb_main.cpp
-rw-rw-r-- 1 nanik nanik  2353 Aug 21  2016 adb_utils.cpp
-rw-rw-r-- 1 nanik nanik  1873 Aug 21  2016 adb_utils_test.cpp
-rw-rw-r-- 1 nanik nanik 60391 Aug 21  2016 commandline.cpp
-rw-rw-r-- 1 nanik nanik  1015 Aug 21  2016 console.cpp
-rw-rw-r-- 1 nanik nanik 18565 Aug 21  2016 fdevent.cpp
-rw-rw-r-- 1 nanik nanik 26795 Aug 21  2016 file_sync_client.cpp
-rw-rw-r-- 1 nanik nanik 11524 Aug 21  2016 file_sync_service.cpp
-rw-rw-r-- 1 nanik nanik  5318 Aug 21  2016 framebuffer_service.cpp
-rw-rw-r-- 1 nanik nanik  1097 Aug 21  2016 get_my_path_darwin.cpp
-rw-rw-r-- 1 nanik nanik   980 Aug 21  2016 get_my_path_linux.cpp
-rw-rw-r-- 1 nanik nanik   980 Aug 21  2016 get_my_path_windows.cpp
-rw-rw-r-- 1 nanik nanik 20872 Aug 21  2016 jdwp_service.cpp
-rw-rw-r-- 1 nanik nanik  1836 Aug 21  2016 qemu_tracing.cpp
-rw-rw-r-- 1 nanik nanik  3687 Aug 21  2016 remount_service.cpp
-rw-rw-r-- 1 nanik nanik 22399 Aug 21  2016 services.cpp
-rw-rw-r-- 1 nanik nanik  6498 Aug 21  2016 set_verity_enable_state_service.cpp
-rw-rw-r-- 1 nanik nanik 25165 Oct 16 14:14 sockets.cpp
-rw-rw-r-- 1 nanik nanik 95001 Aug 21  2016 sysdeps_win32.cpp
-rw-rw-r-- 1 nanik nanik  1897 Aug 21  2016 test_track_devices.cpp
-rw-rw-r-- 1 nanik nanik 30154 Aug 21  2016 transport.cpp
-rw-rw-r-- 1 nanik nanik 14099 Aug 21  2016 transport_local.cpp
-rw-rw-r-- 1 nanik nanik  1733 Aug 21  2016 transport_test.cpp
-rw-rw-r-- 1 nanik nanik  3250 Aug 21  2016 transport_usb.cpp
-rw-rw-r-- 1 nanik nanik 16129 Aug 21  2016 usb_linux_client.cpp
-rw-rw-r-- 1 nanik nanik 21065 Aug 21  2016 usb_linux.cpp
-rw-rw-r-- 1 nanik nanik 17179 Aug 21  2016 usb_osx.cpp
-rw-rw-r-- 1 nanik nanik 14221 Aug 21  2016 usb_windows.cpp
```

```
-rw-rw-r-- 1 nanik nanik   6507 Feb 27 02:06 adb_auth_client.cpp
-rw-rw-r-- 1 nanik nanik   2258 Feb 27 02:06 adb_auth.cpp
-rw-rw-r-- 1 nanik nanik  10520 Feb 27 02:06 adb_auth_host.cpp
-rw-rw-r-- 1 nanik nanik   9926 Feb 27 02:06 adb_client.cpp
-rw-rw-r-- 1 nanik nanik  47348 Feb 27 02:06 adb.cpp
-rw-rw-r-- 1 nanik nanik   5800 Feb 27 02:06 adb_io.cpp
-rw-rw-r-- 1 nanik nanik   4407 Feb 27 02:06 adb_io_test.cpp
-rw-rw-r-- 1 nanik nanik   8229 Feb 27 02:06 adb_listeners.cpp
-rw-rw-r-- 1 nanik nanik   4048 Feb 27 02:06 adb_trace.cpp
-rw-rw-r-- 1 nanik nanik   7946 Feb 27 02:06 adb_utils.cpp
-rw-rw-r-- 1 nanik nanik   5288 Feb 27 02:06 adb_utils_test.cpp
-rw-rw-r-- 1 nanik nanik  10996 Feb 27 02:06 bugreport.cpp
-rw-rw-r-- 1 nanik nanik  18416 Feb 27 02:06 bugreport_test.cpp
-rw-rw-r-- 1 nanik nanik  81915 Feb 27 02:06 commandline.cpp
-rw-rw-r-- 1 nanik nanik   5096 Feb 27 02:06 console.cpp
-rw-rw-r-- 1 nanik nanik   2805 Feb 27 02:06 diagnose_usb.cpp
-rw-rw-r-- 1 nanik nanik  11206 Feb 27 02:06 fdevent.cpp
-rw-rw-r-- 1 nanik nanik   5728 Feb 27 02:06 fdevent_test.cpp
-rw-rw-r-- 1 nanik nanik  34837 Feb 27 02:06 file_sync_client.cpp
-rw-rw-r-- 1 nanik nanik  13192 Feb 27 02:06 file_sync_service.cpp
-rw-rw-r-- 1 nanik nanik   5318 Feb 27 02:06 framebuffer_service.cpp
-rw-rw-r-- 1 nanik nanik   1097 Feb 27 02:06 get_my_path_darwin.cpp
-rw-rw-r-- 1 nanik nanik    980 Feb 27 02:06 get_my_path_linux.cpp
-rw-rw-r-- 1 nanik nanik  20584 Feb 27 02:06 jdwp_service.cpp
-rw-rw-r-- 1 nanik nanik   4224 Feb 27 02:06 line_printer.cpp
-rw-rw-r-- 1 nanik nanik   4756 Feb 27 02:06 remount_service.cpp
-rw-rw-r-- 1 nanik nanik  19120 Feb 27 02:06 services.cpp
-rw-rw-r-- 1 nanik nanik   4291 Feb 27 02:06 set_verity_enable_state_service.cpp
-rw-rw-r-- 1 nanik nanik  29034 Feb 27 02:06 shell_service.cpp
-rw-rw-r-- 1 nanik nanik   1694 Feb 27 02:06 shell_service_protocol.cpp
-rw-rw-r-- 1 nanik nanik   5954 Feb 27 02:06 shell_service_protocol_test.cpp
-rw-rw-r-- 1 nanik nanik  10337 Feb 27 02:06 shell_service_test.cpp
-rw-rw-r-- 1 nanik nanik  25244 Feb 27 02:06 sockets.cpp
-rw-rw-r-- 1 nanik nanik  10118 Feb 27 02:06 socket_test.cpp
-rw-rw-r-- 1 nanik nanik   8152 Feb 27 02:06 sysdeps_test.cpp
-rw-rw-r-- 1 nanik nanik   1990 Feb 27 02:06 sysdeps_unix.cpp
-rw-rw-r-- 1 nanik nanik 103087 Feb 27 02:06 sysdeps_win32.cpp
-rwxrwxr-x 1 nanik nanik   8478 Feb 27 02:06 sysdeps_win32_test.cpp
-rw-rw-r-- 1 nanik nanik   1931 Feb 27 02:06 test_track_devices.cpp
-rw-rw-r-- 1 nanik nanik  31093 Feb 27 02:06 transport.cpp
-rw-rw-r-- 1 nanik nanik  14945 Feb 27 02:06 transport_local.cpp
-rw-rw-r-- 1 nanik nanik   6907 Feb 27 02:06 transport_test.cpp
-rw-rw-r-- 1 nanik nanik   2450 Feb 27 02:06 transport_usb.cpp
-rw-rw-r-- 1 nanik nanik  18420 Feb 27 02:06 usb_linux_client.cpp
-rw-rw-r-- 1 nanik nanik  19763 Feb 27 02:06 usb_linux.cpp
-rw-rw-r-- 1 nanik nanik  18436 Feb 27 02:06 usb_osx.cpp
-rw-rw-r-- 1 nanik nanik  20215 Feb 27 02:06 usb_windows.cpp
```

```
LOCAL_CFLAGS += \
    $(ADB_COMMON_CFLAGS) \
    -D_GNU_SOURCE \
    -DADB_HOST=1 \

LOCAL_MODULE := adb
LOCAL_MODULE_TAGS := debug

LOCAL_STATIC_LIBRARIES := \
    libadb \
    libbase \
    libcrypto_static \
    libcutils \
    liblog \
    $(EXTRA_STATIC_LIBS) \

# libc++ not available on windows yet
ifneq ($(HOST_OS),windows)
    LOCAL_CXX_STL := libc++_static
endif
```

```
LOCAL_CFLAGS += -DALLOW_ADBD_NO_AUTH=$(if $(f

ifneq (,$(filter userdebug eng,$(TARGET_BUILD
LOCAL_CFLAGS += -DALLOW_ADBD_DISABLE_VERITY=1
LOCAL_CFLAGS += -DALLOW_ADBD_ROOT=1
endif

LOCAL_MODULE := adbd

LOCAL_FORCE_STATIC_EXECUTABLE := true
LOCAL_MODULE_PATH := $(TARGET_ROOT_OUT_SBIN)
LOCAL_UNSTRIPPED_PATH := $(TARGET_ROOT_OUT_SB
LOCAL_C_INCLUDES += system/extras/ext4_utils

LOCAL_STATIC_LIBRARIES := \
    libadbd \
    libbase \
    libfs_mgr \
    liblog \
    libcutils \
    libc \
    libmincrypt \
```

De izquierda a derecha: Objetivo ADB y objetivo ABDB.



ADB en host y ADBD en el dispositivo Android

cliente conectados entre sí por IP o USB. Una vez establecida la conexión, los comandos se envían de uno a otro.

Tanto adb como adbd están en la misma base de código, pero lo que les separa durante el proceso de compilación es el uso de macros. Esto hace que el proyecto produzca 2 aplicaciones diferentes, como

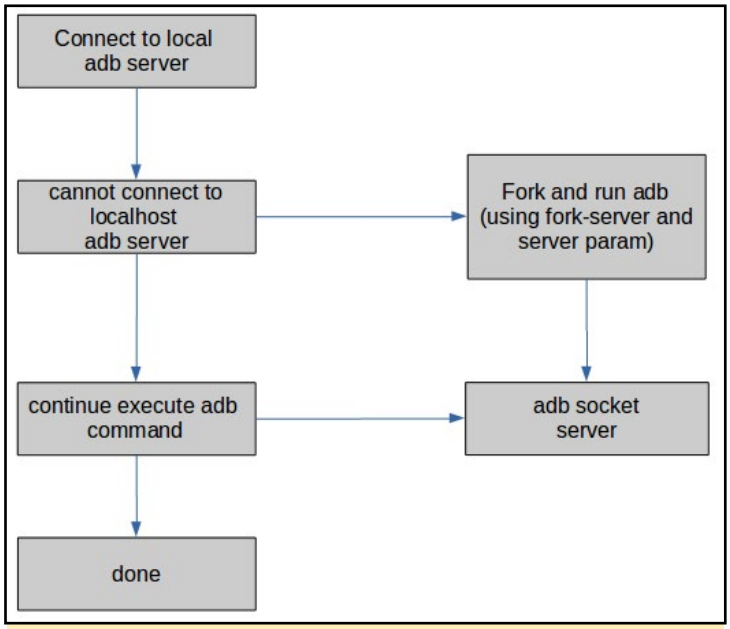
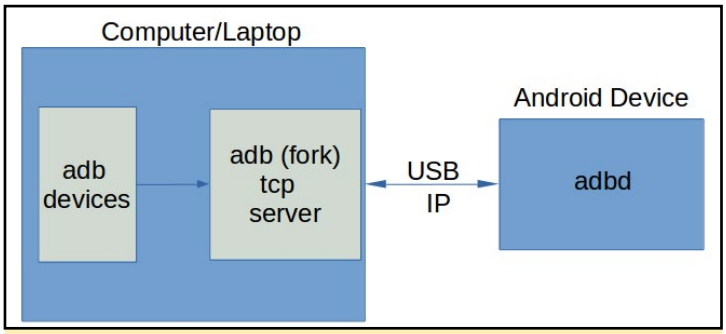


Figura 7 - Flujo del proceso que ejecuta adb en el host

se puede ver desde Android.mk en las figuras 4 y 5. La forma en cómo se ejecuta adb dentro del host es diferente a cómo se ejecuta dentro del dispositivo.

Tal y como muestra el flujo del proceso en la Figura 7, lo que realmente sucede internamente en el equipo host es que adb genera un nuevo proceso adb que se utilizará para comunicarse con el dispositivo Android remoto. Por ejemplo, si ejecutas el comando "adb devices", adb tendrá 2 procesos diferentes ejecutándose en el equipo: uno ejecutándose como servidor estableciendo la comunicación con el adbd en el dispositivo Android y el otro proceso interpretando el



2 procesos en ejecución dentro de host

comando que el usuario ha instruido, que es "devices" en nuestro ejemplo. La Figura 8 describe cómo se verá adb en el equipo host cuando el usuario ejecute un comando como "adb devices."

Lo que esto significa es que cada vez que utilices adb para conectarse a tu dispositivo Android para realizar una operación, una vez completada esta operación, todavía se queda ejecutándose un proceso único en tu ordenador conectado a tus dispositivos Android. Este proceso de larga ejecución te permite continuar realizando operaciones en tus dispositivos sin perder ninguna conexión. En la parte 2, examinaremos más a fondo las diferentes partes de adb usando el código fuente para profundizar en él y entenderlo.

NAVEGANDO POR ANDROID

USANDO UN MANDO A DISTANCIA POR INFRARROJOS

por Lorenzo Carrieri

En este proyecto, vamos a crear un mando a distancia por infrarrojos (IR) que podremos utilizar para controlar remotamente Android, eliminando la necesidad de usar una interfaz táctil. Éste se puede utilizar como un simple mando a distancia, para un sistema de TV Android casero basado en un ODROID o, por ejemplo, como una interfaz entre la red bus CAN automotriz y el sistema operativo Android.

Esta interfaz ha sido desarrollada y probada para que sea completamente funcional con un ODROID-C2 aunque, debería funcionar igual de bien con un ODROID-C1/C1 +, con Android Lollipop 5.1.1 v3.4.

Requisitos de hardware

- ODROID-C2
- 1x Arduino Nano
- 1x mini placa de pruebas
- 4x botones
- 4x Resistencia de 100 kΩ
- 1x Resistencia de 100 Ω
- 1x led IR

Requisitos de Software

- Android Lollipop 5.1.1 v3.4 de Hardkernel
- Arduino IDE v1.8.1
- Librería IRremote Arduino, http://bit.ly/1Isd8Ay
- Gestor de archivos Android que permita editar archivos de texto

Montando el mando a distancia IR

En este ejemplo, crearemos una interfaz IR que sea capaz de abrir el menú de apps de Android, navegar por la lista de aplicaciones e iniciar la aplicación deseada. ¡Ahora, vamos a montar el mando a distancia IR!

Teniendo en cuenta el diagrama de cableado que se muestra en la Figura 1, conecté el pin digital 3 al ánodo del LED IR,

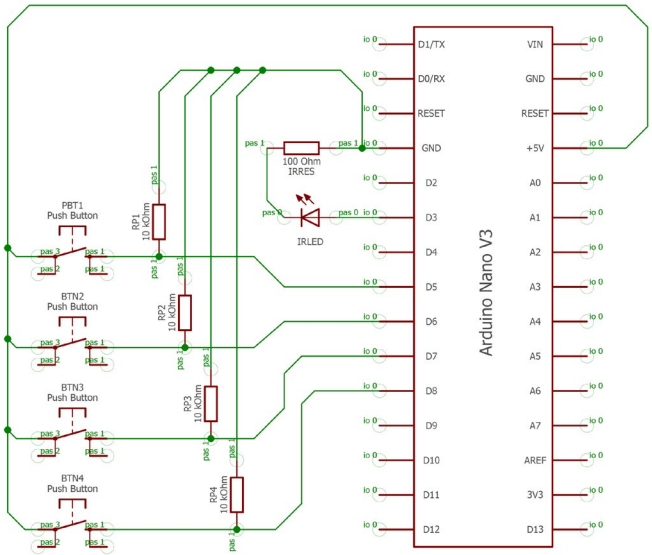
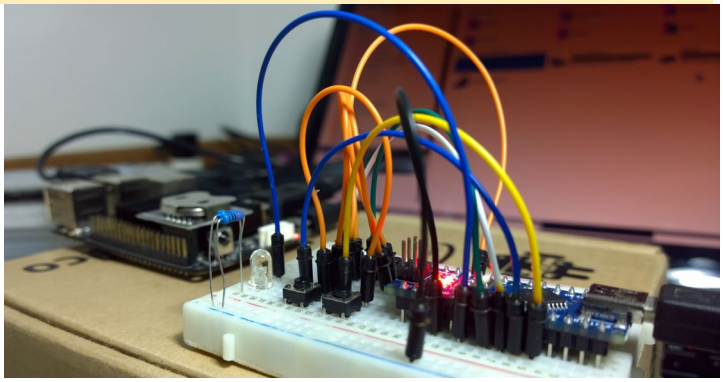


Figura 1 - Diagrama de cableado

llevando el cátodo a la puesta a tierra a través de una serie de resistencias de 100 Ω. A continuación, prepare un grupo de cuatro botones que conecta el pin de salida del sistema de alimentación Arduino Nano +5V a cuatro pines de entrada digitales diferentes, en este caso 5, 6, 7 y 8. Recuerda conectar una resistencia descendente de 10 kΩ entre cada pin de entrada y la puesta a tierra por medio del botón, tal y como se muestra en el diagrama de cableado.

Figura 2 - Control remoto IR



El archivo remote.conf

Después de montar el mando a distancia, necesitamos saber lo que éste "dirá" al receptor IR del ODROID-C2. Abrimos un archivo llamado remote.conf, ubicado en la carpeta '\system\etc'. Dentro del archivo, encontrarás algo similar a esto:

```
key_begin
        0x88 113
        0xdc 116
        ...
key_end

repeat_key_begin
        0x88 113
        0xdc 116
        ...
repeat_key_end
```

Observa la primera línea después de la instrucción 'key_begin', tiene dos partes: una cadena binaria (sistema hexadecimal codificado), un código de acción IR, seguido de una clave numérica (Keycode). La primera parte es la expresión digital que el ODROID-C2 leerá de su receptor IR para ejecutar el evento de Android asociado a la correspondiente clave numérica. El archivo '/usr/keylayout/generic.kl' incluye una lista completa de las claves numéricas de los eventos de Android. Para este ejemplo, buscaremos las siguientes claves numéricas:

Android Event	Description	Android Keycode
APP_SWITCH	Open the app switcher	580
DPAD_UP	Move up in a list	103
DPAD_DOWN	Move down in a list	108
DPAD_CENTER	Confirm selection	97

Tabla I – Claves numéricas de los eventos de Android

Ten en cuenta que, excepto APP_SWITCH, los otros tres eventos ya están presentes en el archivo remote.conf, aunque necesitan una pequeña modificación. De momento, simplemente agrega una nueva línea para el evento APP_SWITCH en cada sección del archivo remote.conf para las expresiones _begin y _end:

```
key_begin
        0x88 113
        0xdc 116
        ...
        0x12 580     (this is the new line added)
key_end

repeat_key_begin
        0x88 113
        0xdc 116
        ...
        0x12 580     (this is the new line added)
repeat_key_end
```

Para el siguiente paso, asegúrate de tener activado el modo depuración IR. Si no es así, puedes activarlo a través del archivo remote.conf cambiando la opción debug_enable de 0 a 1:

```
debug_enable = 1
```

Si lo modificas, guarda y reinicia ODROID-C2 para que el cambio tenga efecto.

Código Arduino para el mando a distancia IR

Primero, instala todas las librerías necesarias tal y como se describe en la sección de Instalación en http://bit.ly/1Isd8Ay. Después, abre un nuevo borrador y pega el siguiente código:

```
/*
* ODROID IR SEND TEST
* An IR LED must be connected to Arduino PWM pin 3.
* Version 0.1 March, 2017
*/

#include <IRremote.h>                 // we include
the IR remote library
IRsend irsend;

void setup()
{
pinMode(5, INPUT);            // now we set all the
input pin
pinMode(6, INPUT);            // to which we will as-
sociate
pinMode(7, INPUT);            // the selected Android
events
pinMode(8, INPUT);
}

void loop() {
// irsend.sendNEC(IR code, 32) is the function that
generate and send the 32 bit IR binary word
// in compliance with the IR NEC protocol (the same
of the ODROID IR receiver)

if (digitalRead(5)==HIGH) {  // open the App switcher
irsend.sendNEC(0x4db2bb00,32);
}

else if (digitalRead(6)==HIGH) { // simulate tap on
```

```
the screen (press OK)
irsend.sendNEC(0x4dce00,32);
}

else if (digitalRead(7)==HIGH) {
// simulate scroll down
irsend.sendNEC(0x4db2d200,32);
}

else if (digitalRead(8)==HIGH) {
// simulate scroll up
irsend.sendNEC(0x4db2ca00,32);
}
}
```

Como se puede ver en el código, la función irsend.sendNEC (código IR, 32) coge un código IR como un parámetro que es un número entero de 32 bits, escrito como hexadecimal, y que contiene el identificador del emisor IR y el código del evento IR. En este código vemos que tiene un formato hexadecimal como este: 0x4db2ca00, tal y como muestra la Figura 3.

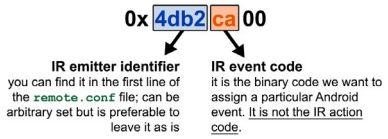


Figura 3 - Estructura del código IR

Para este ejemplo, tenemos cuatro botones de comando y cuatro sentencias 'if' diferentes que, en verdad, llaman a la función irsend para generar la correspondiente señal IR, compatible con el protocolo NEC, en el pin digital 3. Ahora podemos cargar el borrador en Arduino Nano.

Configurando el archivo remote.conf

El paso final de este ejemplo es la configuración del archivo remote.conf, que vincula la asignación del código de acción IR correcto al evento Android deseado. Ten en cuenta que, por ahora, hemos asignado un código de evento IR específico a cada uno de los eventos de Android, pero no conocemos el código de acción IR correspondiente para escribir en el archivo remote.conf. Por lo tanto, conecta tu placa ODROID-C2 al PC con la opción de desarrollador activada y abre una sesión ADB en el símbolo del sistema de Windows y sigue este procedimiento:

- Presiona y mantén pulsado uno de los botones del mando IR, por ejemplo el asociado al a las aplicaciones.
- Sin soltar el botón del mando a distancia, introduce el comando dmesg en el prompt del ADB.
- Suelta el botón.

Si observamos el resultado de dmesg, las últimas filas deberían mostrar algo como esto:

```
remote: scancode is
0x004b,invalid key is 0x0000.
```

El código 0x004b es el código IR que lee el ODROID-C2 en el cual 4b es el código de acción IR que hemos estado buscando. Abre el archivo remote.conf, ve a la línea añadida anteriormente que está relacionada con el evento APP_SWITCH y escribe el código de acción IR correcto. Siguiendo el ejemplo anterior, ahora tendríamos lo siguiente:

```
key_begin
        0x88 113
        0xdc 116
        ...
        0x4b 580      (this is the
new line added with the correct
IR action code)
key_end

repeat_key_begin
        0x88 113
        0xdc 116
        ...
        0x4b 580      (this is the
new line added with the correct
IR action code)
repeat_key_end
```

Repite este procedimiento para todos los botones, luego guarda y reinicie el ODROID-C2. Si todo ha ido bien, ahora debería contar con un mando a distancia IR ODROID casero totalmente funcional.

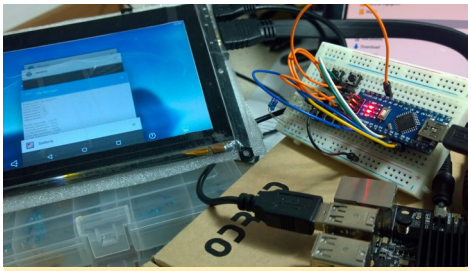


Figura 4 - Sistema completo

Notas

El procedimiento descrito en esta guía permite al usuario iniciar un evento específico de Android usando un mando a distancia IR externo. Desde Android Lollipop 5.1.1 v2.9, ha sido posible iniciar una selección de cuatro aplicaciones diferentes directamente a través de los pines GPIO. Este servicio recuerda los eventos de Android F7, F8, F9 y F10 que pueden asociarse a una aplicación específica simplemente utilizando la aplicación ODROID Utility. Este resultado también se puede conseguir usando un mando a distancia IR externo. Basta con asociar los códigos de teclado 65, 66, 67 y 68 a los eventos de F7, F8, F9 y F10 respectivamente, utilizando el código de acción de IR para asignar los keycodes a los eventos como ha quedado patente en este artículo.

Es de esperar que un mando a distancia para el control remoto sea el tema del próximo artículo de Lorenzo

JUEGOS LINUX

LA SERIE DE CARRERAS DE COCHES F-ZERO

por Tobias Schaaf

La serie F-Zero, que fue publicada para muchos y diferentes sistemas, la mayoría de los cuales tenían un potencial limitado, resultó ser muy conocida por su rápida acción en las carreras. Apareció en primer lugar para SNES, e introdujo lo que ahora se conoce como el desplazamiento en Modo 7, que combina el sistema de escaldo y el posicionamiento/giro de capas para logar un efecto con el que parecía que te encontrabas ante un juego en 3D. La misma técnica fue utilizada más tarde en Super Mario Kart. Elige uno de los cuatro corredores y ve directo a la acción.

Figura 1 - F-Zero en el SNES en Modo 7

El juego es muy rápido en comparación con otros juegos de carreras como Super Mario Kart y a veces, muy implacable. Por ejemplo, si golpeas el borde de la carretera, sufrirás daños y perderás velocidad. También puede rebotar al chocar contra los muros y otros corredores, lo cual puede provocar algunos efectos de giro y hacer que pierdas velocidad y potencia/salud muy rápidamente. La pista también tiene algunos obstáculos como manchas de suciedad que te ralentizan, o almohadillas que te hacen saltar o te dan un impulso, no todos trabajan a tu favor. Al comienzo de cada pista, hay un área de recarga en la que puedes restaurar tu energía si has sufrido daños.

Con cada vuelta completa, consigues un impulso, señalado por una "S" verde en la esquina inferior derecha de la pantalla. Puedes apilar hasta un máximo de 3 impulsos, que te permiten recuperarte rápidamente si tu velocidad baja por haber golpeado a otros coches o a muros.

Además de los corredores contra quienes compites, en las pistas irán apareciendo coches aleatorios una vez que llegue a la segunda vuelta. Estos coches te pueden ralentizar o empujarte contra un muro. Incluso existe un coche parpadeante que aparece en la pista de vez en cuando, el cual debes evitar golpear a toda costa, ya que éste explota, ocasionándote bastante daño y haciendo que pierdas el rumbo completamente. Algunos recorridos incluso tienen bandas magnéticas o viento que hace que te desplaces lentamente hacia los lados.

Las pistas son bastante difíciles y disfruto bastante jugando, aunque por lo general no soy muy fan de los juegos de carreras. El clásico Super Mario Kart realmente me aporta muy poco, se vuelve muy lento en comparación con F-Zero. A diferencia de Mario Kart, la música y el sistema de juego en F-Zero se acoplan perfectamente.

F-Zero tuvo 5 lanzamientos más para SNES, así como para la Super Famicom System, que sólo estaba disponible en Japón. Utilizaba a modo accesorio el módem por satélite Satellaview de la Super Famicom, que te permitía descargar un juego a la memoria y jugar hasta que lo reemplazaras por otro.

El primer juego fue lanzado en 4 episodios: F-Zero Grand Prix - Knight League, Queen League, King League y Ace League. Desafortunadamente, no logré conseguir que ninguno funcionase, pero he leído que hay un parche que debería funcionar con BSNES y que permitiría ejecutarlos de nuevo. F-Zero Grand Prix 2 funciona bastante bien, aunque sólo puede jugar a la Ace League. Parece incluso más rápido que el juego original F-Zero con nuevas y emocionantes pistas, y es bastante más difícil.

F-Zero X

El siguiente lanzamiento de la serie dio el salto al 3D en la Nintendo 64, donde la serie fue recuperada y mejorada bastante, probablemente sea el juego más conocido de la serie.

Figura 2 - F-Zero X en el N64

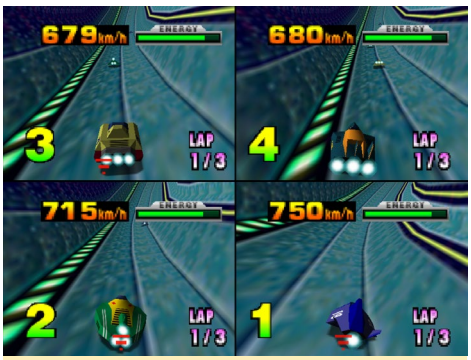


Figura 3 - F-Zero X soporta hasta 4 jugadores en pantalla dividida

Nada más empezar, el juego te recibe con una impresionante música con ritmo acelerado, todo en este juego rebosa "velocidad". El juego fue desarrollado intencionadamente con una baja calidad gráfica, lo cual significa que la línea de visión está algo limitada. Únicamente son visibles algunos objetos aparte de los corredores y de la pista, y raras veces se puede apreciar algún tipo de cielo u otros elementos de fondo. Esto se hizo con el objeto mantener la tasa de fotogramas siempre a 60 FPS, lo cual hace que el juego sea muy rápido y fluido. Todo funciona perfectamente para recrear un fabuloso juego de carreras.

Como hemos ya hemos comentado, el juego presenta muchos cambios con respecto a la versión SNES, como, por ejemplo, competir contra otros 29 corredores en lugar de sólo 3. Las pistas en las que puedes competir han sufrido una actualización de 360°, que fue utilizada al completo en la N64. Podías correr dentro de tubos que te permiten conducir por el techo, o simplemente girar en círculos. Incluso podías competir por el exterior del tubo, lo que significa que, en lugar de estar limitado por muros en todas direcciones, sólo podías ver el cielo, lo cual hacía que pareciera que podías caerse de la pista en cualquier momento. De hecho, salirse de la pista era una posibilidad y algunas pistas, con sus medias o ¼ de tuberías, fueron diseñadas realmente para lanzarte fuera de la pista, estrellarte y arder. En el juego encuentras curvas muy inclinadas, bucles y diferentes tipos de saltos.

Los modos de impulso que pueden usarse también fueros actualizados. A partir de la segunda vuelta, puede utilizar el impulso, pero no como en la versión SNES que te limitaba a un impulso por vuelta; Puedes usarlos mientras tengas Energía. La energía es similar a la barra de potencia en la versión SNES, y se reduce si golpeas los obstáculos o a otros corredores. A partir de la segunda vuelta, puedes usar la misma energía para activar los impulsos. Cada vez que los actives, perderás algo de energía, y con ello también algo de "salud", así que tienes que tener cuidado con la frecuencia con la que uses tus impulsos.

Aparte de los 4 corredores del juego original, hay 26 corredores adicionales que puedes desbloquear y elegir, lo cual hace que el juego sea bastante más variado. Los desarrolladores también añadieron nuevos modos de juego: Time Attack, Death Race e incluso un modo multijugador llamado VS Battle. Puedes desbloquear no sólo nuevos corredores, sino también nuevas copas, siendo la copa final la X-Cup con mapas autogenerados, lo que significaba que las pistas cambiaban cada vez que competías.

Figura 4 - En F-Zero X compites contra otros 29 corredores

Figura 5 – Dando vueltas en círculos en F-Zero X

Figura 6 - Recargando tu medidor de energía en F-Zero X

El juego funciona muy bien si lo ejecutas en un ODROID-XU3/XU4 o en un ODROID basado en Exynos 4 (X, X2, U2 / U3) y es muy divertido. El emulador soporta las vibraciones del joystick, lo cual aumenta la sensación cuando golpeas un muro o a un enemigo o simplemente te estrellas y ardes.

Kit de expansión de F-Zero X

Dos años después del lanzamiento de F-Zero X, Nintendo desarrolló un kit de expansión para el juego. Este sólo estaba disponible en Japón y para la 64DD (la Nintendo 64 Disk Drive). Era un disco de expansión que añadía nuevos corredores, nuevas copas y trayectos, música nueva y dos características extra muy especiales.

Este kit de expansión te permitía diseñar tu propio corredor para el juego con el que podrías perfilarlo a tu gusto, pintarlo del modo que quisieras, añadirle logos y otras mejoras. Esto por si sólo ya era una característica muy interesante, pero los desarrolladores fueron más allá e incluyeron un editor de pistas. Esto te permitía crear tus propias pistas para competir sin importar lo enrevesadas que fueran, con cambios bruscos, curvas y giros. La expansión fue elogiada aún más por esta característica. Desafortunadamente, actualmente no puedes usarla en el ODROID, ya que no pude probarla yo mismo.

F-Zero en GBA

La serie F-Zero fue exportada poco

después a la Game Boy Advanced (GBA) en la cual tuvo tres lanzamientos. En 2001, un año después de la expansión 64DD, fue publicado el primer juego llamado F-Zero - Maximum Velocity, realmente fue un título lanzado para la GBA. Esta vez la serie F-Zero volvió a sus orígenes, el juego tenía buen aspecto y se jugaba como en la clásica versión de SNES. Gracias a la mayor capacidad de procesamiento de la GBA en comparación con el SNES, así como un mayor número de colores, el juego se ejecuta incluso más rápido y más suave en el GBA que en la versión SNES, aunque contaba con una menor resolución, debido a la pequeña pantalla de la GBA. Volví a jugar la versión SNES un rato después de probar la versión GBA, y realmente sentí como de repente el juego se volvía mucho más lento. Aunque la versión SNES tiene una resolución más alta y

Figura 7 - F-Zero a máxima velocidad en la GBA, se parece bastante a la versión SNES

los coches se ven mucho mejor, el fondo en la versión SNES era más bien soso en comparación con la versión GBA. Prefiero la versión GBA sobre la versión SNES aunque la resolución sea algo más baja, porque el juego se ejecuta más rápido y en general es más divertido.

En 2003, F-Zero GP Legend fue lanzado para GBA. Este juego está más cerca de la versión N64, y además está basado en un anime del mismo nombre. En GP Legends, similar a la versión N64, compites como uno de los 30 pilotos de un Gran Premio, aunque todavía se asemeja más bien a la versión SNES porque sólo ves en la pista a unos cuantos corredores. Sin embargo, puedes jugar y desbloquear hasta 34 corredores/personajes, al igual

Figura 8 - Nuevos modos de juego en F-Zero GP Legends

Figura 9 - El modo historia de F-Zero GP Legend se narra en pequeñas secuencias

que en la versión N64. También incluyeron en el juego otras características de la versión N64, como el medidor de potencia duplicando tanto la salud y como la energía de impulso. Añadía un par de nuevos modos de juego como el ataque en el tiempo, y un modo historia que te permitía seguir la historia de 8 de los 34 corredores. La historia se narra a través de escenas cortadas al estilo anime.

El juego también actualiza un poco los gráficos y muestra algunos efectos meteorológicos. Sin embargo, el juego tiene algunos problemas de rebote cuando cochas con muros y otros corredores, lo que conduce a un terrible comportamiento al estilo pinball, donde no puedes controlar tu corredor, simple-

Figura 10 - Efectos climatológicos en F-Zero GP Legend, niebla en lugares brumosos

mente se golpea contra los muros de izquierda y derecha y no puede hacer nada para detenerlo. Eso es muy frustrante y realmente interfiere en la experiencia de juego. La simbiosis de los elementos SNES y N64 es en realidad bastante buena, aunque los gráficos dan un paso hacia atrás en comparación con la versión N64.

La última entrega de la serie fue lanzada en 2004 y se llamaba F-Zero Climax, sólo fue publicada en Japón para

Figuras 11 y 12 - F-Zero Climax es probablemente lo mejor que se puede conseguir en un sistema como GBA, con fondos bien dibujados y un sistema de juego suave y rápido

GBA. F-Zero Climax es en mi opinión el mejor de los juegos 2D. Tiene todo lo que ha hecho que la versión SNES sea excelente, y añade las nuevas características de F-Zero GP Legend, sin los efectos ridículos de pinball. Es realmente divertido de jugar, y tienes un montón de cosas para desbloquear a medida que vas completando las carreras. Incluso viene con un editor de pistas que te permite crear tus propias pistas alocadas.

Otros títulos de F-Zero

En 2003, también se lanzaron dos nuevos juegos en 3D de la serie para Nintendo GameCube y maquinas ar-

cade: FZero GX y F-Zero AX. Ambos juegos eran similares y de hecho podías compartir datos del juego, lo cual significaba que podías coger el personaje de la Game Cube y ponerlo en la máquina Arcade y así desbloquear contenido extra. Era algo muy singular, el sistema arcade era realmente increíble, contaba con una cabina para el piloto, pedales y volante: https://www.youtube.com/watch?v=MAFZvKkVt10.

Los gráficos son realmente buenos y el sistema de juego es ultra rápido, lo cual lo hace realmente divertido. Desafortunadamente, ni la versión para GameCube ni la versión de arcade pueden ejecutarse en los ODROIDs, pero esto podría cambiar con futuras placas. Sin embargo, si tienes un PC con Dolphin Emulator para Game-Cube, puedes ejecutar el juego hasta en resoluciones 4k y activar diversos efectos dependiendo de lo que soporte tu PC, haciendo que este juego sea espectacular a nivel gráfico. Definitivamente debes probarlo si te gustan los juegos de carreras.

Notas

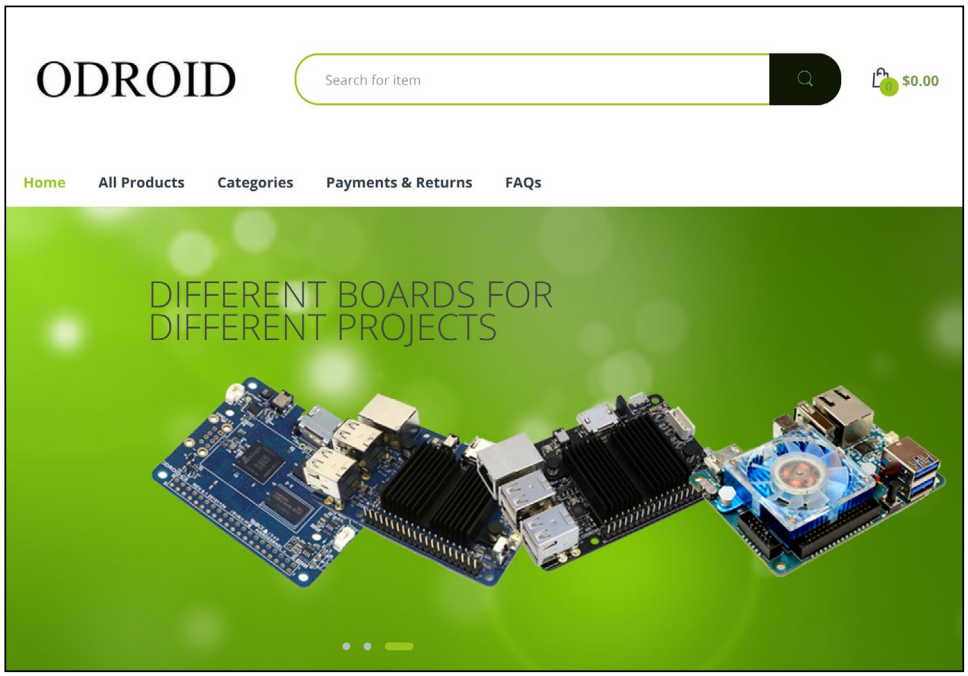
Soy nuevo en la serie F-Zero, pero la he disfrutado bastante. Me ha gustado la versión de SNES por ser mucho más rápida que Super Mario Kart, pero después de probar las versiones de GBA, incluso la versión SNES me parecía lenta. Definitivamente que quedo con F-Zero Climax sobre GP Legend, aunque el modo historia es una buena aportación. F-Zero X en el N64 también es genial, especialmente si juegas con amigos. Los giros en 3D son muy divertidos, y de hecho me encanta la música de la versión N64. Este juego simplemente te hace sentir la velocidad, y el soporte para vibración mejora la sensación con respecto a la versión arcade. Espero que algún día también podamos jugar a la versión de GameCube en un ODROID, y dispongamos de toda la gloria de F-Zero en un único dispositivo.

UNA NUEVA TIENDA ODROID HA ABIERTO EN EE.UU.

VISITA ODROIDINC.COM

por Rob Roy (robroy)

Un nuevo distribuidor de Hardkernel acaba de abrir una tienda online en Estados Unidos, el cual ofrece un catálogo completo de productos, incluyendo el ODROID-XU4, ODROID-C0, ODROID-C1 + y ODROID-C2. El sitio ofrece descuentos por cantidad para muchos productos y varias opciones de envío para una entrega rápida. Abrirá sus puertas a mediados de abril, así que asegúrese de visitar la nueva web www.odroidinc.com y así poder cubrir todas tus necesidades sobre ordenadores de placa reducida.



RETROPIE VERSION 4.2

AHORA OFRECE SOPORTE NATIVO PARA ODROID-C2

editado por Rob Roy

RetroPie te permite convertir tu Raspberry Pi o PC en una máquina de juegos retro. Está basado en Raspbian, EmulationStation, RetroArch y muchos otros proyectos para que puedas jugar a tus juegos favoritos Arcade, de consolas domésticas y clásicos juegos de PC con una configuración mínima. Para los usuarios avanzados, también proporciona una gran variedad de herramientas de configuración para que personalicen el sistema a su gusto. RetroPie se asienta sobre un sistema operativo completo, así que puedes instalarlo sobre una imagen de Ubuntu existente, o empezar con la imagen SD de RetroPie y añadir software adicional más adelante.

Emuladores

Un emulador es un software que hace que un ordenador se comporte como otro ordenador, o en el caso de RetroPie, un ordenador que se comporte como una consola de videojuegos como la Super Nintendo. La imagen SD de RetroPie viene preinstalada con muchos y diferentes emuladores. Las ROM son versiones digitales de cartuchos de juegos. Cargar una ROM en un emulador equivale a poner un cartucho en una consola de juegos. Las ROMs son contenidos protegidos con derechos de autor y como tales no están incluidas en RetroPie.

Version 4.2

Mucho ha cambiado desde la versión 4.1, con actualizaciones para EmulationStation añadiendo soporte de video y solucionando el problema de la temida pantalla en blanco. Se han actualizado muchos paquetes, RetroPie 4.2 incluye el último RetroArch v1.5.0 así como Kodi 17 (instalado opcionalmente). RetroPie 4.2 también incluye soporte inicial para la placa ODROID-C2, que está instalada sobre la imagen mínima Ubuntu de ODROID-C2. También te darás cuenta que la documentación ha recibido una actualización muy importante en http://bit.ly/2pdVK42. Incluye muchos otros cambios, como mejoras en la facilidad de uso y correcciones de errores. Para conocer los detalles, consulta el registro de cambios más adelante.

Puedes descargar la imagen 4.2 desde http://bit.ly/1WB25BO. Para las nuevas instalaciones, sigue las instrucciones de instalación de http://bit.ly/2pdZNO2. Si actualizas desde la versión 4.0.x, debes hacer primero una copia de seguridad, después, selecciona “Update all installed packages” desde el menú principal de RetroPie-Setup. La persona que actualice desde la versión 3.x deberá actualizar primero el script RetroPie-Setup, más información en http://bit.ly/2pXcV80.

También puede instalar RetroPie sobre una configuración de Raspbian existente, o sobre Ubuntu en un PC/Odroid-C1/C2. Los enlaces a las correspondientes instrucciones los pueden encontrar en el área de descargas. Gracias a todos los que han contribuido en esta versión con una mención especial a @fieldofcows por sus excelentes mejoras de EmulationStation.

Registro de cambios

- Mejoras EmulationStation: Soporte de video, solucionado el problema de la pantalla blanca.
- Soporte para el C2 sobre la imagen mínima de Ubuntu 16.04.
- Kodi 17 ahora es instalable desde paquetes opcionales.
- AdvanceMame se ha actualizado y dividido en tres paquetes distintos: 0.94, 1.4 y v3.3.
- Actualizado a RetroArch v1.5.0.
- Para hacer coincidir los cambios anteriores, lrmupen64plus ha sido renombrado a lr-parallel-n64, y lr-glupen64 ha sido renombrado a lrmupen64plus.
- Solucionado el inicio de escritorio Pixel y otras apps X11 desde EmulationStation.
- Solucionados los problemas de compilación de Zdoom, ResidualVM, Mupen64Plus y PPSSPP
- Versiones de Doom añadiren automáticamente scripts de inicio si encuentran doom1.wad, doom2.wad, tnt.wad o plutonia.wad.
- Se ha añadido el emulador lr-snes9x, una versión libretro del actual snes9x.
- Añadido Amiberry (emulador Amiga), un fork actualizado de uae4arm, con más funciones.
- Soporte multi-disco zip para Vice (emulador C64), fs-uae, uae4arm y Amiberry (Amiga).
- Iniciar imágenes de disco Amiga directamente desde EmulationStation con uae4arm y Amiberry.
- Versión independiente de Stella (emulador At-ari 2600) actualizada a v4.7.3.
- Usbromservice, permite montar

memorias USB en ~/RetroPie para mantener las ROMs en un USB.
- Posibilidad de configurar temas ES personalizados en configs/all/platforms.cfg, que puede anular cualquier configuración en RetroPie-Setup/platforms.cfg
- SDL2 actualizado a 2.0.5
- El scraper de @Sselph actualizado a la última versión y se han añadido nuevas opciones. El Scraper ha sido movido a paquetes opcionales y es necesario instalarlo para que aparezca en configuration/tools.
- Incluidos los paquetes de controladores PowerBlock y ControlBlock.
- Script de configuración de entrada para Daphne.
- Los menús de configuración de RetroPie ahora funcionan con todos los joysticks conectados, aunque la asignación de los botones sigue siendo por hardcoded.
- Código de detección de RPI actualizado para dar soporte a BRANCH=próximo firmware/kernel.
- Revisado el script de inicio runcomand.
- Soporte eliminado para Raspbian Wheezy.
- Nuevos paquetes añadidos a la sección experimental.

Montar RetroPie

Para montar RetroPie en el ODROID-C2, empieza con una imagen preconfigurada de Ubuntu desde el sitio web de Hardkernel en http://bit.ly/1dFLsQQ (http://bit.ly/1OU4kbl). Extrae el archivo.xz con un programa como 7zip y luego, graba el archivo .img en tu tarjeta SD o módulo EMMC con Win32DiskImager (http://bit.ly/2pe19YV). A diferencia de la imagen SD RetroPie, la imagen de ODROID expandirá automáticamente el sistema de archivos, de modo que no es necesario describir este paso. Para empezar, escriba los siguientes comandos en una ventana Terminal:

```
$ sudo apt-get update && sudo apt-get upgrade
$ sudo apt-get install -y git
$ cd
$ git clone --depth=1 https://github.com/RetroPie/
RetroPie-Setup.git
$ cd RetroPie-Setup
$ sudo ./retropie_setup.sh
```

Si tiene problemas a la hora de compilar los módulos y se bloquea el sistema, necesitas decirle que sólo compile con un núcleo mientras se ejecuta el script de instalación:

```
$ sudo MAKEFLAGS="-j1" ./retropie_setup.sh
```

Todos los módulos se pueden instalar desde el Script de instalación de RetroPie. Los dos paquetes principales que necesitas para que la mayor parte de tu sistema funcione son RetroArch y EmulationStation. Después, podrás elegir tus emuladores desde las opciones disponibles.

Solución al problema del sonido

Si no tienes sonido o el sonido se entrecorta en el menú o el juego, comprueba el uso de la CPU a través top o htop. Si pulseaudio está utilizando más del 20%, puede ser que sea el culpable. ¡En mi caso, estaba usando el 80%! Tenía solo una salida de sonido (HDMI) así que eliminé completamente pulseaudio. En Ubuntu 14, esto se hace con el siguiente comando:

```
$ sudo apt-get --purge remove pulseaudio
```

Para desactivar pulseaudio en su lugar, escribe los siguientes comandos en una ventana Terminal:

```
$ mkdir ~/.pulse
$ echo "autospawn=no" >> ~/.pulse/client.conf
$ pulseaudio -k
```

Configurar los mandos

En el primer arranque, tu sistema de archivos se expandirá automáticamente y a continuación, se te dará la bienvenida con el menú de configuración de mandos para los emuladores Emulation y RetroArch. Mantén presionado cualquier botón de tu teclado o gamepad y el nombre aparecerá en la parte inferior y luego se abrirá un menú de configuración. Sigue las instrucciones en pantalla para configurar tu gamepad. Si te quedas sin botones, simplemente manten presionado un botón para ignorar cada botón no utilizado. Cuando se te pida que pulse "OK", pulsa el botón que has configurado como "A".

Si deseas configurar más de un mando, puedes hacerlo desde el menú de inicio de emulationstation. Tienes más detalles sobre las configuracion de mandos en http://bit.ly/2ooycuf.

Botones de acceso directo

Los botones de acceso directo te permiten presionar una combinación de botones para acceder a determinadas funciones como guardar, cargar y salir de los emuladores. La imagen anterior muestra las combinaciones de botones de acceso directo por defecto. De forma predeterminada, los botones de acceso rápido están activados, lo que significa que manteniendo pulsada el botón "Select" mientras pulsa otro puedes ejecutar un comando. Las botones de acceso rápido son específicas de los emuladores basados en retroarch/libretro.

Hotkeys	Action
Select+Start	Exit
Select+Right Shoulder	Save
Select+Left Shoulder	Load
Select+Right	Input State Slot Increase
Select+Left	Input State Slot Decrease
Select+X	RGUI Menu
Select+B	Reset

Instalar emuladores adicionales

En RetroPie 4.0+, no todo está instalado por defecto. Las imágenes precompiladas contienen los emuladores que mejor funcionan para cada sistema según su hardware. Esto debería cubrir la mayor parte de las necesidades de la mayoría de los usuarios. Versiones de juegos como Quake y Doom y algunos otros emuladores como ScummVM se pueden instalar después.

El software se puede instalar desde el script RetroPie-Setup, accesible desde el menú RetroPie en EmulationStation. Una vez allí, puedes navegar hasta "Manage Packages", donde verás varias secciones. Cada sección tiene listas de paquetes que puedes instalar junto con los que ya está instalado actualmente. Los paquetes adicionales estables están en la sección "Opcional", junto con paquetes más inestables listados bajo la pestaña experimental. Los paquetes están ordenados primero por tipo y luego por orden alfabético. Al seleccionar un paquete, puedes elegir entre instalarlo o eliminarlo. Algunos paquetes también tienen configuraciones adicionales.

Transferir las ROMs

Debido a la naturaleza/complejidad de la Ley de Derechos de Propiedad Intelectual y Copyright, que difiere significativamente de un país a otro, los ROM no se pueden facilitar con RetroPie y deben ser proporcionadas por el propio usuario. Únicamente debes tener as ROMs de los juegos que poseas. Existen tres métodos principales para transferir las ROM: USB, SFTP y Samba.

Para transferir vía USB, asegúrate de que tu USB esté formateado en FAT32 o NTFS, y luego cree una carpeta llamada retropie en tu memoria USB. Conéctalo a tu ODROID y espera a que termine de parpadear, luego retira el USB y conéctalo al ordenador que contiene las ROM. Agrega las ROM a sus respectivas carpetas en la carpeta retropie/ROMs, luego vuelve a conectarlo al ODROID y espere a que termine de parpadear. Refresca EmulationStation eligiendo "Restart EmulationStation" en el menú Inicio.

Para transferir a través de SFTP, primero debe habilitar SSH (http://bit.ly/2oDSP0S). El nombre de usuario por defecto es "pi", y la contraseña es "raspberry". También puede iniciar sesión como usuario root si desea cambiar más archivos a parte de las ROM, pero primero debes habilitar la contraseña de root, lo cual se explica en http://bit.ly/2pdX3QG.

Para transferir a través de Samba desde una máquina Windows, escriba \\retropie en el Explorador de Windows, o escribe \\<dirección-ip>. Si utiliza OS X, selecciona "Go" en el Finder y "Connect to Server". Escribe smb://retropie (o usa la dirección IP) y selecciona "Connect".

Ejecutar juegos

Una vez añadidas las ROM, debes reiniciar EmulationStation para que se muestren, usando el menú de Inicio o reiniciando tu ODROID mediante el comando "sudo reboot". Consulta la documentación de Retropie en http://bit.ly/2pdVK42 para obtener información más detallada sobre los emuladores y sus configuraciones avanzadas. Para temas que no localices en la documentación, puedes consultar los foros de la comunidad de RetroPie en http://bit.ly/2oE5D7l.

El proyecto RetroPie esta mantenido principalmente por unos cuantos desarrolladores que lo desarrollan en su tiempo libre. Si encuentras útil el proyecto RetroPie, por favor considera la posibilidad de realizar una donación al proyecto en http://bit.ly/2q7bbs0. A medida que te familiarices con RetroPie, hazlo progresar ayudando a otros en los foros. El Proyecto RetroPie es lo que es hoy día gracias a las numerosas contribuciones de la comunidad. Para comentarios, preguntas y sugerencias, consulte el artículo original en http://bit.ly/2pX8Vof.

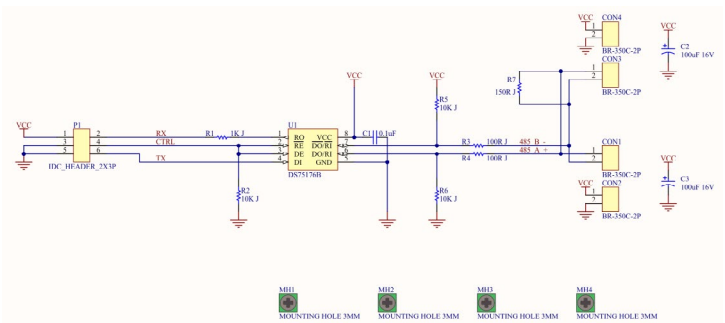
EXPLORANDO EL RS485 LA COMUNICACION EN PLACAS CI+ Y C2

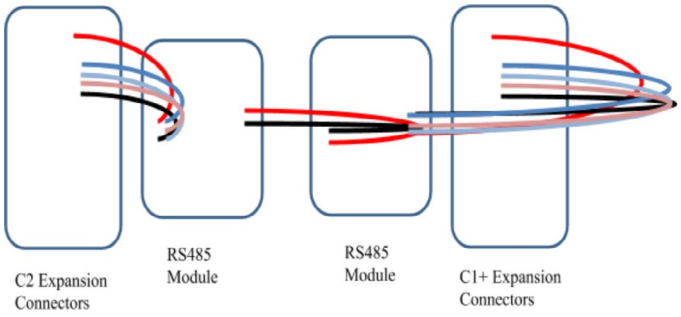
por Charles Park y Neal Kim

En este artículo tengo la intención de analizar el protocolo de comunicación RS-485 basándose en mis experiencias, más bien explicando los principios de la comunicación RS-485. A menudo suelo configurar comunicaciones entre sistemas que utilizan RS-232, que es más simple, más barato y más utilizado que otros protocolos de comunicación. Se trata de conectar el chip del controlador RS-232 a cualquier MCU (normalmente usando una MCU de 8 bits hecha por PIC o Atmega) y configurar el puerto UART para que se conecte con otras MCUs. Para comunicaciones de puerto a puesto y a una distancia corta, utilizo RS-232. Sin embargo, muchos sistemas industriales utilizan las comunicaciones RS-485, las cuales ofrecen comunicaciones muy estables y pueden propagar mensajes a muchos dispositivos desde una única fuente a cierta distancia.

A un ingeniero que tenga experiencia en la implementación de controladores UART le resultará relativamente fácil crear una conexión RS-232 o RS-485, ya que ambos simplemente determinan las características eléctricas del generador y el receptor. No especifican ni recomiendas ningún tipo de protocolo de comunicaciones, sólo la capa física.

Sin embargo, si tienes un problema al montar un sistema usando RS-485, puede que tengas recurrir a la depuración. Simplemente quiero presentar un sencillo ejemplo de comunicación RS-485, una operación de medio dúplex con 2 hilos. En primer lugar, conecta los dispositivos tal y como se muestra en las figuras 1a y 1b. Su configuración debería ser similar a la de la Figura 2.





Figuras Ia y Ib - Diagrama esquemático del módulo RS-485

ODROID-C2	RS485	RS485	ODROID-C1+
TXD1(PIN8)	B-	B-	TXD1(PIN8)
RXD1(PIN10)	A+	A+	RXD1(PIN10)
GPIOX.10(PIN12)	CTRL	CTRL	GPIO.Y(PIN12)
P3.3V(PIN1)	P3.3V	P3.3V	P3.3V(PIN1)
GND	GND	GND	GND

Tabla I – Tabla con la disposición de los Pines

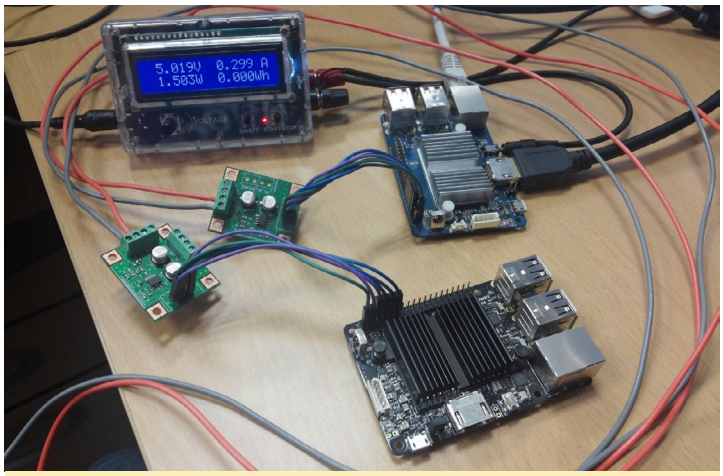


Figura 2 - Dispositivos ODROID conectados mediante RS-485

Software

Hemos testeado los siguientes pasos en un ODROID-C2 (maestro) y C1+ (esclavo) ejecutando las últimas imágenes de Ubuntu, ambas descargadas de http://bit.ly/1R6DOgZ. Antes de empezar, ejecuta los siguientes comandos en una ventana Terminal en ambos dispositivos, luego reinicia:

```
$ sudo apt-get update && sudo
apt-get upgrade \
  && sudo apt-get dist-upgrade
$ sudo apt-get install git vim
```

Para montar la aplicación de prueba, necesitamos la librería wiringPi. Primero, descarga el código fuente rs485_test.c desde https://pastebin.com/GN2LxwZV. A continuación, descarga wiringPi y compila el script de prueba dentro de un ejecutable con los siguientes comandos:

```
$ git clone https://github.com/
hardkernel/wiringPi
$ ./wiringPi/build
$ gpio -v
gpio version: 2.33
Copyright © 2012-2014 Gordon Hen-
derson
This is free software with ABSO-
LUTELY NO WARRANTY.
For details type: gpio -warranty
Hardkernel ODROID Details:
Type: ODROID-C1/C1+, Revision: 1,
Memory: 1024MB, Maker: Hardkernel
$ gcc -o rs485_test rs485_test.c
-lwiringPi -lpthread
```

Comunicación

El dispositivo maestro transfiere un mensaje, incluyendo el ID, cada segundo a los dispositivos esclavos, y luego espera hasta obtener una respuesta de acuse de recibo (ACK) desde el esclavo. En caso de que el maestro no reciba la respuesta ACK del esclavo, el maestro aumentará la variable ack_ fail_count y retransmitirá el mensaje al esclavo. El esclavo permanentemente está en espera de recibir un mensaje del maestro y envía

STX (1 byte)	MSG ID (4 bytes)	MSG (32 bytes)	Check data (1 byte)	ETX (1 byte)
0xFF	Message transfer ID	Messages	XOR MSG ID and MSG in packet	0xFE

Tabla 2 - Estructura del Protocolo

una respuesta ACK al maestro inmediatamente tras recibir el mensaje. Este comando en el dispositivo maestro ODROID-C2 debería mostrar resultados similares a los que aparecen en la Figura 3

```
$ sudo ./rs485_test -d /dev/ttyS1
-b 115200 -p 1 -m 1
```

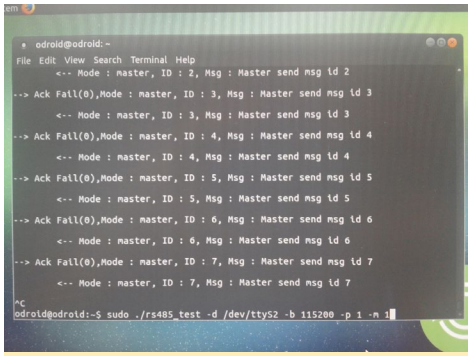


Figura 3 - Resultados del script de prueba RS485 en el ODROID-C2 maestro

Para comprobar que se están aceptando los mensajes, escribe el siguiente comando en el dispositivo ODROID-C1 esclavo, que debería mostrar resultados similares a la Figura 4:

```
$ sudo ./rs485_test -d /dev/ttyS2
-b 115200 -p 1 -m 0
```

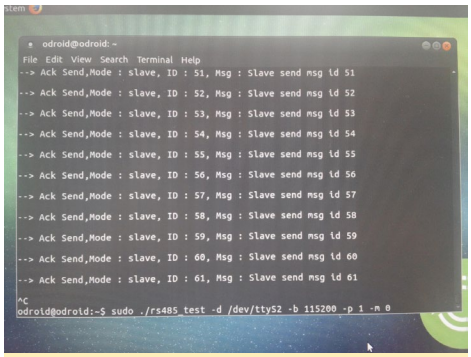


Figura 4 - Resultados del script de prueba RS485 en el ODROID-CI esclavo

SPACETEAM

AHORA TUS AMIGOS TIENEN UNA BUENA RAZON PARA GRITARSE EN VOZ ALTA

por Rob Roy

Spaceteam es un juego cooperativo en grupo para 2 a 8 jugadores que se gritan mutuamente hasta que su nave explote. Se te asignará un panel de control aleatorio con botones, interruptores, deslizadores y diales. Necesitas seguir las instrucciones teniendo en cuenta el tiempo. Sin embargo, las instrucciones se envían a tus compañeros, de modo que tiene que coordinarte con ellos antes de que se acabe el tiempo. Además, la nave se está desintegrando, y estás intentando sobrepasar una estrella en explosión. ¡Buena suerte y recuerda trabajar conjuntamente... como un Equipo Galáctico!

https://play.google.com/store/apps/details?id=com.sleepingbeastgames.spaceteam

En Spaceteam, trabajas en la vida real para controlar la nave espacial virtual

CONOCIENDO UN ODROIDIAN

STEPHEN NEAL (@NOGGIN)

Trabajo en la industria de la radiodifusión con una función creativa, pero empecé mi carrera como ingeniero de investigación y desarrollo en un fabricante de equipos de difusión por radio y tv. Actualmente vivo en Londres, Inglaterra, y tengo un grado de ingeniería de la Universidad de Cambridge.

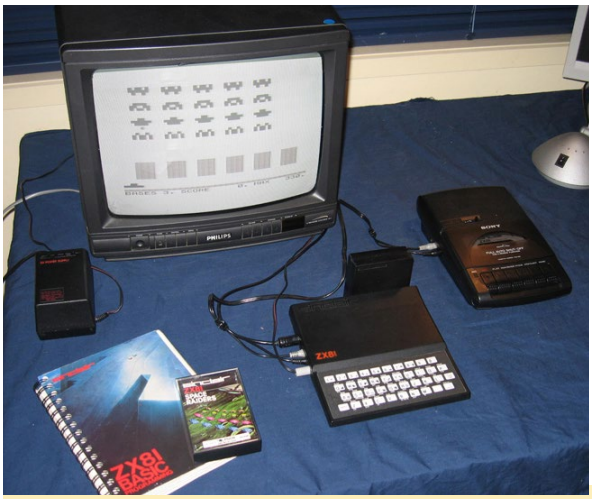
Mi primer ordenador fue un ZX81 de un kit de mi padre de 1981. Después de eso, tuve un Acorn BBC Micro, un Acorn Archimedes, que fue mi primer ordenador personal basado en ARM, y varias máquinas DOS hasta que terminé en Windows, luego un Macintosh. Poseo una gran colección de micrófonos clásicos de 8 y 16 bits de los años ochenta. Más recientemente, he estado trapicheando con Linux en plataformas x86 y ARM con la reproducción multimedia, la recepción de TV, PBX VOIP y domótica, y he empezado a probar máquinas virtuales ESXi. También tengo montado un servidor unRAID, el cual almacena todos mis archivos multimedia

Me atrajo la plataforma ODROID por su rendimiento y precio. Inicialmente compré un U2, ya que se llegó a hablar mucho de él en los círculos de Kodi (soy moderador en los foros de Kodi), que al final resulto ser un poco decepcionante. El U2 no consiguió el apoyo necesario para convertirse una gran plataforma, aunque su hardware debía haber hecho que volara. Mi U2 era un juguete muy entretenido, pero realmente nunca llegue a sacarle mucho partido. El C1 y C1+ fueron mis próximas compras que parecían muy prometedores. Sin embargo, el C2 fue el que cambió las reglas del juego. Mi C2 es uno de mis principales reproductores HEVC de 2160/ 50p que ejecuta LibreElec. El usuario @wrxtasy del foro y el equipo que desarrolla LibreElec y Kodi para la plataforma AMLogic S905 merecen un gran reconocimiento por su trabajo por convertir Kodi en un tema tan importante para la plataforma ODROID.

Stephen disfruta leyendo novela policiaca

Utilizo principalmente mi C2 para ejecutar LibreElec y así hacer funcionar Kodi. La falta de un kernel moderno, que sé que está en camino, significa que ejecutar mi C1/C1+ como TV Headend es un fastidio debido al limitado soporte del kernel para los modernos sintonizadores DVBT2, que sería lo ideal para estos. Ejecutar los C1s como reproductores multimedia secundarios. En estos momentos tengo un sistema x86 que funciona como mi servidor de TV Headend principal.

El ODROID C2 es mi favorito porque es magnífico como reproductor multimedia con un precio fantástico. El almacenamiento eMMC es bueno y rápido. Sería estupendo que integrara Wifi 802.11ac y Bluetooth 4.0 o superior, junto con una salida Displayport capaz soportar resoluciones muy nítidas. HDMI es genial, pero ejecutar una pantalla iPad a través de un convertidor DP a eDP, que es algo muy trivial y ampliamente disponible, permitiría usar pantallas de 9.7" baratas, lo cual permitiría crear impresionantes "no tan pequeñas tablet". Soporte USB 3.0 también sería algo fantástico para una mayor velocidad de ancho de banda en almacenamiento. ¡Realmente sería genial ver un hibrido del C2 y XU4, si existiera tal SoC!



El ordenador ZX81 fue una máquina increíble cuando apareció por primera vez, y fue cuando Stephen empezó con los ordenadores

También me gustaría ver kernels actualizados. La Raspberry Pi y las plataformas x86 son más flexibles que el ODROID precisamente por esto. Me encantan mis ODROIDs, pero un kernel actualizado con soporte de drivers para nuevos dispositivos sería aún mejor.

En mi tiempo libre, me gusta aprender idiomas, ver televisión, leer novelas policiacas y viajar. He tenido la suerte de viajar mucho por mi trabajo, y realmente disfruto visitando nuevos lugares.

Mi consejo para nuevos programadores es simplemente lanzarse y comprar un ordenador ARM como el C1+ o C2, o una Raspberry Pi, ya que su soporte educativo no tiene rival, y aprender Linux. Después, abrirse a nuevos horizontes con Python. También debes utilizar Google de un modo inteligente, ya que puede aprender mucho viendo cómo otras personas han resuelto problemas similares.