YoloDROID • GlusterFS • Cloudshell • Stereo Boom Bonnet • Boombox

ODROID Magazine

Año Cuatro
Num. #48
Dic 2017

Manos libres:

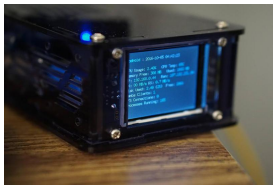
UNIDAD HFP Y A2DP BLUETOOTH CON ODROID-XU4

HOME ASSISTANT : USANDO INFRAROJOS, MOTORES Y RELÉS

Usando tu ODROID-XU4 como altavoz Bluetooth A2DP o como una unidad de manos libres HFP con tu iPhone

December 1, 2017

El primero es retransmitir audio a través de Bluetooth A2DP desde un iPhone a un ODROID-XU4, y el segundo es utilizar el ODROID-XU4 como unidad de manos libres HFP para el iPhone durante las llamadas.



Pantalla LCD personalizada para ODROID CloudShell y CloudShell 2

December 1, 2017

Este artículo no es guía paso a paso para crear una pantalla de estado personalizada, sino que más bien tiene un enfoque general.

Stereo Boom Bonnet, Una estupenda forma de disfrutar de la música en tu ODROID

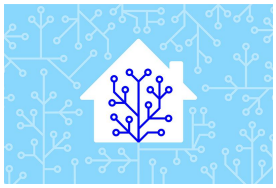
December 1, 2017

El I2S 2Watt Stereo Boom Bonnet Kit (https://goo.gl/1mXXVH) es un compacto sistema de altavoces para ODROID-XU4 y ODROID-C1+/C2. Utiliza I2S como estándar de sonido digital para la salida de audio. Es muy fácil de instalar y te encontraras “rockeando” en 15 minutos. Para conectarlo a tu ODROID, sigue estos pasos: ▶

Boom Box: Ingeniería de sonido con un mejor altavoz

December 1, 2017

Para mejorar el sonido del Stereo Boom Bonnet de Hardkernel (https://goo.gl/TrDU8u), fui a mi tienda de hobby local y me hice con un “pack variado” de estireno que resultaba tener algunas hojas .080mil y algunas tuberías dentro. También compré un poco de disolvente de estireno para soldar. Lo hice todo ▶



Home Assistant: Usando Infrarrojos, Motores y Relés

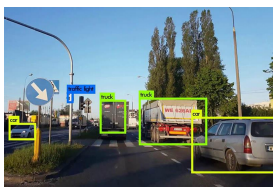
December 1, 2017

En este artículo, vamos a combinar el ODROID con algo de hardware y Home Assistant, para empezar a convertir dispositivos no inteligentes en dispositivos “inteligentes”. Trabajaremos con cables, soldaremos un poco, conectaremos relés y en algunos casos, llegaremos a trabajar con voltaje, ¡así que ten cuidado! Convirtiendo una unidad de ▶

Juegos Android: Stranger Things, Pocket Morty y Streets of Rage

December 1, 2017

Es muy fácil relajarse con un ODROID porque existen muchas opciones de juegos para el sistema operativo Android. En este artículo, vamos a analizar tres nuevas entregas: Stranger Things: The Game, Pocket Morty’s, y Streets of Rage. Stranger Things: The Game Directamente salido de un exitoso show de TV, nos ▶



Ejecutando YOLO en ODROID: YOLODROID

December 1, 2017

Esta guía te ayudará a instalar y ejecutar TinyYOLO en tu ODROID-XU4.

Juegos Linux: Need for Speed II Second Edition

December 1, 2017

La segunda edición de Need for Speed II (NFS2SE) se lanzó con soporte 3DFX, con más pistas y coches, junto con el modo espejo y el modo pista hacia atrás, lo cual supuso una gran mejora con respecto al Need for Speed II original

Analizando el almacenamiento definido por software con GlusterFS en el ODROID-HC1: Parte 2 – Rendimiento del cliente

December 1, 2017

Te mostraré cómo configurar los clientes NFS y Samba para acceder al volumen GlusterFS y compararemos el rendimiento de los diferentes clientes.

Conociendo un ODROIDian: Andrea Cole, editora adjunta de ODROID Magazine

December 1, 2017

Por favor, háblanos un poco sobre ti. Actualmente soy administradora de ventas de Lab Manager, una publicación centrada en la industria para la comunidad científica. He formado parte de su empresa matriz, LabX Media Group, durante más de 10 años, y he estado trabajando concretamente en la división Lab Manager ▶️

Usando tu ODROID-XU4 como altavoz Bluetooth A2DP o como una unidad de manos libres HFP con tu iPhone

December 1, 2017 By Dennis Chang ODROID-XU4, Mecaniqueo

En este artículo se presentan dos posibles usos que pueden ser de especial interés para todos aquellos que desarrollas ordenadores para vehículos usando los ODROIDs. El primero es transmitir audio a través de Bluetooth A2DP desde un iPhone a un ODROID-XU4, y el segundo consiste en utilizar el ODROID-XU4 como una unidad de manos libres HFP para el iPhone durante las llamadas. Para este proyecto utilizaremos el sistema bluetooth Bluez 5, el entorno de desarrollo de aplicaciones móviles oFono y el software de sistema de sonido PulseAudio. Aunque no lo he probado, este procedimiento también debería funcionar con un teléfono Android.

A2DP es muy fácil de configurar y se completa con la configuración del manos libres. HFP es más difícil de configurar porque requiere compilar PulseAudio 11 desde la fuente y reemplazar el paquete PulseAudio 8 preinstalado en Ubuntu MATE. He dividido este artículo en dos secciones para que aquellos que sólo estén interesados en A2DP eviten la dificultad que tiene lograr que HFP funcione. Si solo estás interesado en la funcionalidad del altavoz Bluetooth, puede utilizar el sistema de audio integrado del ODROID-XU4 sobre el puerto HDMI en lugar del adaptador de audio USB basado en el C-Media CM108. Para que las cosas sean lo más simples posible, vamos a suponer que utilizarás el adaptador de audio USB que aparece en este artículo.

Antes de empezar, ten en cuenta que la documentación y los debates sobre las últimas versiones de Bluez 5, oFono y PulseAudio son algo escasas. La mayor parte de lo que encontré on line para el posible uso de este proyecto era demasiado antiguo, centrado en versiones anteriores del software. Si te encuentras con algún problema al intentar probar este proyecto, es muy probable que

no encuentre a nadie experto en el tema para ayudarte, incluido yo mismo. Te aconsejo que no te desvíes demasiado de mis indicaciones y del hardware que he utilizado hasta que hayas configurado correctamente el proyecto, entonces podrás dar rienda suelta a tu creatividad sabiendo que tienes un punto de partida funcional al que puedes volver.

Fase 1: A2DP

A2DP se puede ejecutar usando los paquetes preinstalados con Uuntu 16.04.3 MATE: Bluez 5.37, oFono 1.17 y PulseAudio 8. Empieza con un ODROID-XU4 ejecutando el Ubuntu oficial 16.04.3 con una imagen MATE kernel 4.9 y ejecuta todas las actualizaciones del sistema operativo antes de empezar con este proyecto. Yo usé el archivo de imagen ubuntu-16.04.3-4.9-mate-odroid-xu4-20170824.img. Posiblemente sea mejor elegir el ODROID-XU4 en lugar del ODROID-XU4Q, ya que el rendimiento extra de la CPU puede ayudarte a reducir la latencia de audio de las llamadas o mejorar la calidad del audio si decide ir más allá y optimizar el remuestreo del PulseAudio.

Los dispositivos específicos que utilicé para este proyecto son:

- iPhone 5S
- Adaptador USB Bluetooth Cambridge Silicon Radio: **http://bit.ly/2gNybJW**.
- Adaptador de audio USB Sanwu Audio SW-HF07 (con un chip C-Media CM108 que sabemos que funciona en ARM LINUX con el driver que integra): **http://bit.ly/2zEBIWZ**.
- Auriculares con micrófono de 3.5mm independiente y clavijas de auricular para hacer pruebas

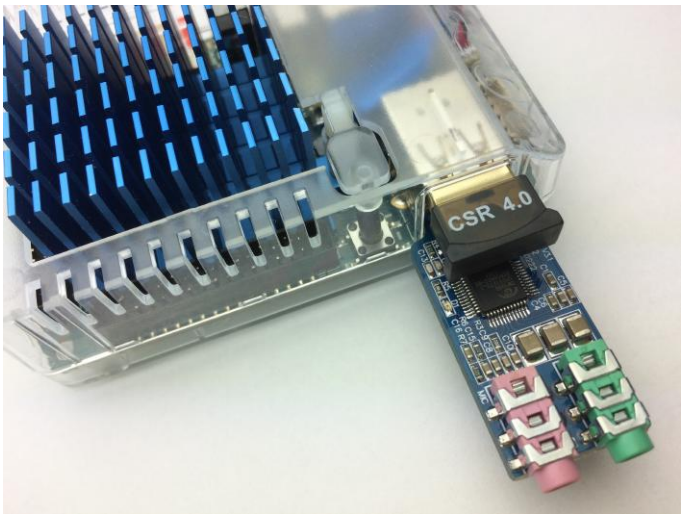


Figura 1 – Primer plano de los adaptadores de audio y bluetooth USB

Inserta los adaptadores de audio y Bluetooth USB en los puertos USB disponibles en el ODROID-XU4, luego conecta los auriculares y el micrófono en las clavijas correspondientes del adaptador de audio USB.

Seleccionar la interfaz de audio correcta

Inicie sesión en el escritorio MATE como el usuario predeterminado “odroid”. Este paso es importante porque PulseAudio está configurado para ejecutarse en modo usuario por defecto, así que se iniciará automáticamente tras iniciar sesión, pero no se estará ejecutando cuando se muestra la pantalla de inicio de sesión. No vamos a configurar PulseAudio para ejecutarlo en todo el sistema, ya que presenta ciertas dificultades.

Prueba el audio utilizando la aplicación de Preferencias de sonido incorporada en MATE. Cambia y guarda la configuración según sea necesario, testea el sonido tanto como sea necesario. Como yo utilizo un adaptador de audio C-Media USB, he tenido que seleccionarlo como dispositivo de entrada y salida predeterminado en lugar del audio integrado del ODROID-XU4 (salida a través del puerto HDMI). Deje la aplicación de preferencias de sonido abierta para que PulseAudio se ejecute en la sesión de usuario.

Figura 2 – Seleccionando el dispositivo de audio USB como entrada en el panel de control de Preferencias de sonido

Figura 3 – Seleccionando el dispositivo de audio USB como salida en el panel de control de Preferencias de sonido

En el directorio /home/odroid/.config/pulse, busque los archivos que terminan en "-default-sink" y "-default-source" y anote los nombres de los archivos y sus contenidos. Deberías ver algo como esto:

```
dc87f36fc06c441a85ff7269baabcdef-default-sink:
 alsa_output.usb-C-
Media_Electronics_Inc._USB_Audio_Device-
00.analog-stereo

dc87f36fc06c441a85ff7269baabcdef-default-
source:
 alsa_input.usb-C-
Media_Electronics_Inc._USB_Audio_Device-
00.analog-mono
```

Queremos que el dispositivo seleccionado en ambos archivos sea la interfaz de audio que tiene el micrófono, que generalmente no es la salida de audio HDMI integrada. Use la aplicación de Sonido para probar la entrada y salida de audio antes de pasar al siguiente paso.

Emparejamiento Bluetooth con tu iPhone

Bluez 5.x ya está preinstalado en la imagen de Ubuntu 16.04.3 MATE, así que todo lo que tenemos que hacer es vincular tu iPhone al ODROID-XU4. Utilizaremos el comando bluetoothctl incluido, que se debe ejecutar con privilegios de root, o de lo contrario aparecerá un error:

```
$ sudo -s
# bluetoothctl
 [bluetooth]# show
```

Deberías ver algo como esto:

```
Controller 00:AA:BB:CC:DD:11
 Name: odroid
 Alias: odroid
 Class: 0x1c0000
 Powered: yes
 Discoverable: no
 Pairable: yes
 UUID: Headset AG (00001112-0000-1000-8000-
############)
 UUID: Generic Attribute Profile (00001801-
0000-1000-8000-############)
 UUID: A/V Remote Control (0000110e-0000-1000-
8000-############)
 UUID: OBEX File Transfer (00001106-0000-1000-
8000-############)
 UUID: Generic Access Profile (00001800-0000-
1000-8000-############)
 UUID: OBEX Object Push (00001105-0000-1000-
8000-############)
 UUID: PnP Information (00001200-0000-1000-
8000-############)
 UUID: A/V Remote Control Target (0000110c-
```

```
0000-1000-8000-############)
 UUID: IrMC Sync (00001104-0000-1000-8000-
############)
 UUID: Audio Sink (0000110b-0000-1000-8000-
############)
 UUID: Audio Source (0000110a-0000-1000-8000-
############)
 UUID: Vendor specific (00005005-0000-1000-
8000-############)
 UUID: Message Notification Se.. (00001133-
0000-1000-8000-############)
 UUID: Phonebook Access Server (0000112f-0000-
1000-8000-############)
 UUID: Message Access Server (00001132-0000-
1000-8000-############)
 Modalias: usb:v1D6Bp0246d0525
 Discovering: no
```

Si PulseAudio no se está ejecutando, la lista de perfiles es mucho más corta.

Si “Powered” no es “yes”, escribe el siguiente comando:

```
# power on
```

Ahora empecemos a emparejar escaneando los dispositivos cercanos:

```
# scan on
```

En tu iPhone, ve a Configuración> Bluetooth y asegúrate de que esté encendido y sea visible. Tarde o temprano verás tu iPhone aparecer en el escaneo por su dirección MAC Bluetooth. Escribe a continuación esta dirección MAC ya que la usará repetidas veces en lugar de .

```
# scan off
# agent KeyboardOnly
# default-agent
# pair [MAC]
```

Puede fallar; inténtalo de nuevo hasta que logres realizar el emparejamiento y te solicite la clave de acceso. Mira en tu iPhone la clave e introdúcela cuando aparezca:

```
Attempting to pair with [MAC]
 [CHG] Device [MAC] Connected: yes
 Request passkey
 [agent] Enter passkey (number in 0-999999):
######
```

```
# connect [MAC]
# trust [MAC]
# info
```

Deberías ver los detalles en tu iPhone y tus perfiles de Bluetooth. En tu iPhone, debería mostrarse que el dispositivo llamado “odroid” está conectado.

```
# exit
```

Figura 4 – Conectando ODROID en la configuración de Bluetooth del iPhone

Llegados a este punto, deberías ser capaz de enviar la reproducción de audio desde la aplicación iTunes de su iPhone al ODROID-XU4 a través del Bluetooth. Si no lo consigues, dirígete a las preferencias de Bluetooth del iPhone y fuerza la reconexión al “odroid”, incluso si ya está conectado.

Figura 5 – Enviando la reproducción de audio a través de Bluetooth desde iTunes al ODROID-XU4

También es posible escuchar el marcado cuando se usa la aplicación del Teléfono, pero como todavía no hemos configurado el ODROID-XU4 con HFP, no oirás nada una vez que la aplicación del Teléfono abra la llamada.

Fase 2: HFP

El siguiente paso es instalar Ofono 1.17.x para el perfil de manos libres Bluetooth, ya que no se encuentra preinstalado en la imagen del sistema operativo. Suponiendo que todavía seguimos en la misma sesión "sudo -s", escribe el siguiente comando:

```
# apt-get install ofono
```

Verifica que el perfil de Bluetooth esté añadido:

```
# bluetoothctl
 [bluetooth]# show

Controller 00:AA:BB:CC:DD:11
 Name: odroid
 Alias: odroid
 Class: 0x3c0000
 Powered: yes
 Discoverable: no
 Pairable: yes
 UUID: Headset AG (00001112-0000-1000-8000-
############)
 UUID: Generic Attribute Profile (00001801-
0000-1000-8000-############)
 UUID: A/V Remote Control (0000110e-0000-1000-
8000-############)
 UUID: OBEX File Transfer (00001106-0000-1000-
8000-############)
 UUID: Generic Access Profile (00001800-0000-
1000-8000-############)
 UUID: OBEX Object Push (00001105-0000-1000-
8000-############)
 UUID: PnP Information (00001200-0000-1000-
8000-############)
 UUID: A/V Remote Control Target (0000110c-
0000-1000-8000-############)
 UUID: IrMC Sync (00001104-0000-1000-8000-
############)
 UUID: Audio Sink (0000110b-0000-1000-8000-
############)
 UUID: Audio Source (0000110a-0000-1000-8000-
############)
 UUID: Handsfree (0000111e-0000-1000-8000-
############)
 UUID: Vendor specific (00005005-0000-1000-
8000-############)
 UUID: Message Notification Se.. (00001133-
0000-1000-8000-############)
 UUID: Phonebook Access Server (0000112f-0000-
1000-8000-############)
 UUID: Message Access Server (00001132-0000-
1000-8000-############)
 Modalias: usb:v1D6Bp0246d0525
 Discovering: no
```

Date cuenta que ahora tenemos el perfil de Manos libres (5º contando desde abajo en la sección UUID de la lista). A continuación, salte de bluetoothctl:

```
# exit
```

Llegados a este punto, es posible iniciar una llamada telefónica con la aplicación de teléfono del iPhone seleccionando el dispositivo “odroid” como audio de manos libres, oirás los tonos del teclado mientras se hace la llamada, pero tan pronto como empiece la llamada, el ODROID-XU4 soltará el audio y hará que el iPhone cambie desde “odroid” a su altavoz y micrófono internos.

Aquí es donde tenemos que experimentar un poco. PulseAudio 8 empaquetado por Ubuntu aparentemente tiene un error o le falta una función que provoca la caída del audio de la llamada y completa /var/log/syslog con estos mensajes de error (visibles si configuras PulseAudio en modo depuración):

```
D: [bluetooth] module-loopback.c: Requesting
rewind due to end of underrun.
I: [alsa-sink-bcm2835 ALSA] module-loopback.c:
Could not peek into queue
```

La solución es desinstalar PulseAudio 8 y luego compilar e instalar PulseAudio 11.1 (la última versión en el momento de escribir este artículo) desde el código fuente, tal y como se detalla a continuación.

Deberías estar todavía en la misma sesión “sudo -s”, después de ejecutar bluetoothctl. Si no es así, escribe el siguiente comando:

```
$ sudo -s
```

Es posible que quieras hacer una copia de seguridad del archivo de inicio automático de PulseAudio para volver a utilizarlo más adelante:

```
# cp /etc/xdg/autostart/pulseaudio.desktop ~
# apt-get remove pulseaudio
# apt-get autoremove
# dpkg --purge pulseaudio
```

Es una buena idea eliminar la vieja carpeta de configuración de PulseAudio:

```
# rm -fr /etc/pulse
```

PulseAudio 8 ahora se encuentra eliminado, así que vamos a obtener, compilar e instalar PulseAudio 11.1:

```
# apt-get build-dep pulseaudio
# apt-get install git
# exit
$ cd ~
```

Hazte con el código fuente usando git con uno de los dos comandos que aparecen a continuación:

```
$ git clone
git://anongit.freedesktop.org/pulseaudio/pulse
audio
```

o:

```
$ git clone
http://anongit.freedesktop.org/git/pulseaudio/
pulseaudio.git
```

PulseAudio también se lanza en archivos comprimidos si no quieres la versión de desarrollo del repositorio de git.

```
$ cd pulseaudio
$ export CFLAGS=-fomit-frame-pointer
$ ./autogen.sh
$ make
```

La compilación tardará unos 15 minutos y lanza muchas advertencias, pero debería finalizar sin ningún error importante. Si la compilación salió bien y debería ser así, puedes instalar PulseAudio.

```
$ sudo make install
```

Ten en cuenta que PulseAudio compilado desde la fuente ubica sus archivos de configuración dentro de /usr/local/etc/pulse, y no en /etc /pulse tal como lo hace el PulseAudio proporcionado con Ubuntu.

Existe un parámetro que activaremos en el archivo de configuración de PulseAudio para permitir que el micrófono mono sea remezclado a estéreo. Si no se hace, el audio procedente del micrófono será descartado.

```
$ sudo vi /usr/local/etc/pulse/daemon.conf
```

Elimina la marca de la siguiente línea borrando el punto y coma y guarda el archivo:

```
enable-remixing = yes
```

Debes reiniciar el ODROID-XU4 y volver a iniciar sesión con el usuario odroid, de lo contrario, PulseAudio actuará mal y provocará un “crujido” en el

audio del micrófono. El reinicio hará que el iPhone pierda su conexión Bluetooth con ODROID. A continuación, puede iniciar PulseAudio (sin la ventaja de que los scripts que lo inician automáticamente al iniciar sesión):

```
# pulseaudio --start -D
```

En este punto, deberías poder direccionar el audio de la llamada al ODROID. Vuelve a conectar tu iPhone al dispositivo “odroid” dirigiéndote a Configuración> Bluetooth y seleccionando “odroid” en la lista de dispositivos emparejados.

Empieza reproduciendo música usando iTunes de iPhone. Luego, haz una llamada y escucha el audio de la llamada a través del ODROID. La prueba final es hablar a través del micrófono y que en el otro lado confirmen que pueden oírte.

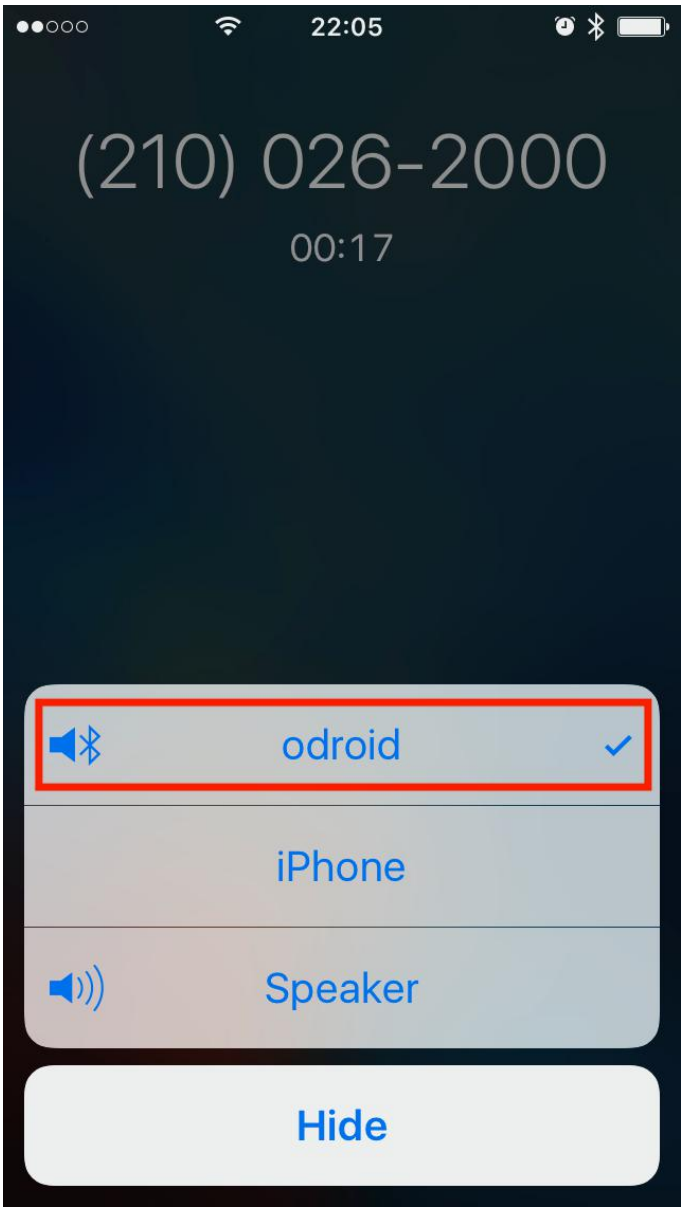


Figura 6 – Probando el micrófono durante una llamada telefónica usando el iPhone

Finalizando

Ten en cuenta que algunas de las funciones del panel de Sonido que proporciona Ubuntu ahora no funcionan porque hemos compilado e instalado PulseAudio desde cero. Particularmente, la utilidad “Test Speakers” en la pestaña Hardware parece que ya no funciona, no podrás escuchar los efectos de sonido de la IU MATE. Sin embargo, todavía era capaz de monitorizar visualmente la entrada del micrófono en la pestaña Input y ajustar el volumen usando el control deslizante de la bandeja del sistema. Configura PulseAudio para que se inicie automáticamente en el modo usuario creando este archivo (o copia el antiguo en su lugar, algo que yo no probé):

```
/etc/xdg/autostart/pulseaudio.desktop

[Desktop Entry]
 Version=1.0
 Name=PulseAudio Sound System
 Comment=Start the PulseAudio Sound System
 Exec=start-pulseaudio-x11
 Terminal=false
 Type=Application
 Categories=
 GenericName=
 X-GNOME-Autostart-Phase=Initialization
 X-KDE-autostart-phase=1
 NoDisplay=true
```

Reinicia el ODROID-XU4 e inicia sesión con el usuario "odroid", luego asegúrate de que PulseAudio se esté ejecutando antes de volver a conectar tu iPhone emparejado.

Solución de problemas

El procedimiento anterior ha sido probado cuidadosamente varias veces, de modo que son pocas la soluciones que existen a posibles problemas. Si algo saliese mal, estos simples pasos pueden ayudarte a corregir las cosas:

1) Reiniciar ODROID-XU4 (recuerda iniciar PulseAudio si no lo tienes configurado para iniciarse automáticamente)

2) Dirígete a la Configuración de iOS> Bluetooth y asegúrate de que el dispositivo "odroid" esté conectado. Si es así, desconectar y volver a conectar es una buena idea.

3) En el peor de los casos, hacer que el iPhone se olvide del dispositivo y volver a hacer el emparejamiento casi siempre arregla cosas.

Para cualquier otro error, lo mejor es mirar los registros log con el siguiente comando:

```
$ sudo tail -f /var/log/syslog | grep
'bluetooth\|ofono\|pulse'
```

Bluez 5, oFono y PulseAudio son bastante complejos, de modo que deberías centrarse en buscar debates en el foro sobre los mensajes de error que veas en el registro log, si los tienes. Es posible configurar cada uno para aumentar los niveles de registro log. He descubierto que con cambiar log-level = debug en PulseAudio /usr/local/etc/pulse/daemon.conf es suficiente. Por lo general, no me fue necesario ver los registros log de depuración de oFono o Bluez para solucionar problemas.

Ten en cuenta que los niveles de volumen de entrada y salida del audio no están controlados por el iPhone. Son controlados por PulseAudio, y la forma más sencilla de controlarlos es mediante el control deslizante de configuración de volumen situado en la bandeja del sistema de Ubuntu MATE y el control deslizante en la aplicación de Sonido. Además, algunos adaptadores de audio USB y micrófonos con cable tienen una mejoría en la entrada muy leve. Incluso con el volumen de entrada de PulseAudio al máximo, pueden llegar a ser demasiado bajo para que cubra tus necesidades. Siempre puedes probar con un micrófono diferente, un adaptador de audio USB distinto o usar un pequeño amplificador analógico entre el micrófono y el adaptador de audio USB. Si no estás satisfecho con la calidad de audio, es posible ajustar PulseAudio para que mejore en rendimiento y calidad. Esta es una amplia área temática que no cubrimos en este artículo.

Finalmente, observe que el iPhone no se conecta con ahínco al ODROID-XU4 a través de Bluetooth, como lo hace con mi altavoz Bluetooth, que también está basado en el chipset Cambridge Silicon Radio. Probablemente haya algunos ajustes dentro de Bluez que pueden cambiar la forma en la que se conecta a los dispositivos emparejados. Tampoco trataremos este tema aquí.

Conclusión

Si sigues las instrucciones anteriores y te acercas lo más posible al hardware que he utilizado, tendrá un ODROID-XU4 funcionando correctamente como un altavoz Bluetooth A2DP o una unidad de manos libres HFP para tu proyecto de ordenador para el coche. Te animo a que desarrolles sobre esta base, como, por ejemplo, escribir o exportar una aplicación que use la API completa de oFono para controlar el iPhone a través de una interfaz de usuario con pantalla táctil desde el ordenador del coche. Si estás pensando en poner en marcho este proyecto, por favor, comparte

tus resultados con la comunidad ODROID escribiendo sobre ello en Odroid magazine.

Pantalla LCD personalizada para ODROID CloudShell y CloudShell 2

December 1, 2017 By Mike Partin CloudShell, ODROID-XU4

Este artículo no es guía paso a paso para crear una pantalla de información personalizada, sino que más bien tiene un enfoque general. Para continuar, necesitarás unos conocimientos básicos de programación. Cualquier lenguaje de programación servirá, ya que mostraré dónde y cómo encontrar la información que necesitamos para alcanzar nuestro objetivo. Dicho esto, también he incluido un enlace al proyecto al final del artículo, el cual está escrito en lenguaje Go.

Si tiene una carcasa ODROID CloudShell o CloudShell 2, es muy probable que hayas utilizado el paquete cloudshell-lcd y hayas comprobado lo útil que resulta la pantalla de información. Mi problema era que quería que se mostrara más información y que el texto pequeño que aparece resultaba muy difícil de leer desde el otro lado de la sala. Quería algo más visual que se pudiera leer rápidamente y entender al instante. Las barras de progreso parecían una buena solución, así que me centre en ello. La mayoría de la información necesaria se podía recopilar leyendo archivos de /proc en Linux, esto se puede conseguir con el sistema de archivos linproc.

Uso de CPU

Consciente de que quería estadísticas sobre la CPU, RAM, Memoria de intercambio, sistema de red y almacenamiento, tenía suficiente para empezar. Linux permite que el uso de la CPU y la mayoría de las estadísticas, sea bastante fácil de obtener con la información disponible en '/proc/stat'. Hay más información disponible en **http://bit.ly/2jGKrRd**. La primera línea nos proporciona un conjunto de estadísticas básicas, con los siguientes campos representados:

* user: Tiempo consumido en modo usuario * nice: Tiempo consumido en modo usuario con baja

prioridad (agradable) * system: Tiempo consumido en modo sistema * idle: Tiempo consumido en reposo sin tareas activas * iowait: Tiempo de espera hasta completar E/S. Poco fiable, ver página de manual proc(5) para más detalles.

* irq: Tiempo de interrupción del servicio * softirq: Tiempo del servicio softirqs * steal: Tiempo robado, tiempo consumido en otros sistemas operativos, en cargas de trabajo de virtualización * guest: Tiempo dedicado a ejecutar una CPU virtual para sistemas operativos invitados (cargas de trabajo virtuales) * guest_nice: Tiempo dedicado a ejecutar un invitado niced (cargas de trabajo virtuales)

Dado que el espacio es muy escaso en nuestra pantalla, solo nos preocupamos por la primera línea, ya que nos proporciona las estadísticas totales. El siguiente comando mostrará únicamente la primera línea de '/proc/stat'.

```
$ head -n1 /proc/stat
 cpu 817905 909158 818680 133949276 2463 0
11128 0 0 0
```

Para recopilar nuestras estadísticas, necesitamos un delta, esto significa que necesitamos leer el valor, esperar un tiempo, por ejemplo 1 segundo, y luego leer otro. La diferencia entre estos valores nos indica como de ocupado estaba el sistema en ese segundo. Los números serán un poco raros, no parecerán marcas de tiempo. Hay una buena explicación para ello, ya que realmente no lo son. Son un contador para lo que se denomina "jiffies". Para mediciones más precisas, como es el tiempo real del procesador consumido en cada atributo, necesitaríamos encontrar el valor estático HZ del kernel. Este comando debería darnos este valor.

```
$ zgrep -i hz /proc/config.gz
```

Este valor es aproximadamente el número de tics por segundo, que suele ser 1000 en la mayoría de las plataformas compatibles con Intel, pero los sistemas embebidos a menudo usan 100. En nuestro caso, solo podemos obtener una medida del proceso frente al tiempo de trabajo:

```
$ head -n1 /proc/stat ; sleep 1; head -n1
/proc/stat
 cpu 885034 1050588 935349 152731137 2546 0
12670 0 0 0
 cpu 885039 1050588 935350 152731533 2546 0
12670 0 0 0
$ tot1=$((885034 + 1050588 + 935349 +
152731137 + 2546 + 12670))
$ wrk1=$((935349 + 152731137 + 2546 + 12670))
$ tot2=$((885039 + 1050588 + 935350 +
152731533 + 2546 + 12670))
$ wrk2=$((935350 + 152731533 + 2546 + 12670))
$ tot3=$((${tot2} - ${tot1}))
$ wrk3=$((${wrk2} - ${wrk1}))
$ python -c "print((${wrk3}.0 / ${tot3}.0) *
100.0)"
```

Uso de RAM y de Memoria Intermedia

Las estadísticas de la RAM se pueden coger de múltiples fuentes. Por ejemplo, podrían leerse de '/proc/meminfo'. Sin embargo, decidí no hacerlo porque los valores están en kilobytes en lugar de bytes. Elegí un syscall directo en lugar de tener que abrir un archivo con su correspondiente tratamiento resultante. A continuación, tienes un pequeño programa escrito en GO usando CGO en lugar del paquete syscall, el cual hice para simplicidad las cosas. Utiliza "sysconf (3)", que permite recopilar rápidas estadísticas de memoria. Para más información, visita **http://bit.ly/2jBNXfI**.

```
package main

// #include
 import "C"
 import "fmt"

func main() {
 maxRam := int64(C.sysconf(C._SC_PHYS_PAGES) *
C.sysconf(C._SC_PAGE_SIZE))
 freeRam :=
int64(C.sysconf(C._SC_AVPHYS_PAGES) *
C.sysconf(C._SC_PAGE_SIZE))
 usedRam := (maxRam - freeRam)
 ramPercUsed := (float64(usedRam) /
float64(maxRam)) * 100.0
 fmt.Println("total =", maxRam)
 fmt.Println("free =", freeRam)
 fmt.Println("used =", usedRam)
 fmt.Println("(used / total) * 100 =",
```

```
ramPercUsed)
 }
```

Este método te proporciona tanto la cantidad de páginas de memoria usadas como disponibles. Esto, multiplicado por la constante de tamaño de página del sistema _SC_PAGE_SIZE, nos da la cantidad de memoria utilizada y disponible. Esta estadística fue fácil y tiene menos "partes móviles" que la estadística del uso de la CPU.

Las estadísticas de memoria de intercambio (swap), por otro lado, pueden leerse fácilmente, de modo confiable y sin gasto adicional de cálculo desde el archivo "/proc/swaps". Se parece a esto:

```
$ cat /proc/swaps
 Filename Type Size Used Priority
 /dev/sda2 partition 8388604 0 -1
 ...
```

A esto no le hace falta explicación alguna, el tamaño y las columnas usadas se miden en bytes.

Uso de red

Este puede resultar divertido. Primero, necesitas saber qué dispositivo de red deseas medir. Puede buscar esta lista de múltiples formas. Una forma sería analizar el resultado "ifconfig -a" o "ip addr". Éste es un poco engorroso si lo comparamos con otros métodos, como es hacer una relación de los contenidos de "/sys/class/net/". En mi sistema, esto devuelve "eth0", "lo" y "wlan0". Conseguir la velocidad de la interfaz es tan simple como usar el siguiente comando:

```
$ cat /sys/class/net/eth0/speed
 1000
```

El siguiente paso es medir nuestro rendimiento. Tomaremos valores periódicos como los que usamos para medir el consumo de la CPU. En este artículo, me centraré en la interfaz eth0, y los datos que busco se encuentran en "/sys/class/net/eth0/statistics/rx_bytes" y "/sys/class/net/eth0/statistics/tx_bytes ". Tienen el siguiente formato, el cual nos proporciona con facilidad métricas de red básicas:

```
$ cat /sys/class/net/eth0/statistics/rx_bytes
 324429106
```

Uso del Disco

Aquí podemos medir dos tipos de uso del disco: el rendimiento y la capacidad. El primero se puede extraer de forma similar a otras estadísticas que hemos recopilado hasta ahora del directorio "/proc". El archivo "/proc/diskstats" almacenará datos similares a los siguientes:

```
8 0 sda 52410 323 1699276 29193 450364 121816
4772512 41466 0 19286 70376
 8 1 sda1 101 0 6594 93 1 0 8 0 0 70 93
 8 2 sda2 46 0 4424 43 0 0 0 0 0 33 43
 8 3 sda3 52238 323 1686170 29020 448708
121816 4772504 41316 0 19113 70040
 8 16 sdb 81 0 4184 33 0 0 0 0 0 16 33
```

Los campos están definidos en el árbol de fuentes del kernel de Linux en el archivo "Documentation/iostats.txt":

- Major number
- Minor number
- Device name
- Reads completed
- Reads merged
- Sectors read
- Time spent reading (in ms)
- Writes completed
- Writes merged
- Sectors written
- Time spent writing (in ms)
- Iops currently in progress
- Time spent in iops (in ms)
- Weighted time spent in iops (in ms)

Es fácil encontrar las mediciones de rendimiento, pero ¿qué pasa con la capacidad del disco? Una forma de hacerlo es con un poco de C (o Go, Rust, Nim, Python, Ruby o cualquier otro que pueda interactuar con C) y el syscall "statfs (2)". Go tiene un paquete syscall, como he mencionado anteriormente, que voy a usar nuevamente para este ejemplo en particular:

```
package main

import (
 "fmt"
```

```
 "syscall"
 )

func main() {
 stat := &syscall.Statfs_t{}
 syscall.Statfs("/", stat)
 fmt.Println("Total space:", (stat.Blocks *
uint64(stat.Bsize)))
 fmt.Println("Free space :", (stat.Bfree *
uint64(stat.Bsize)))
 }
```

Guardar ese código en un archivo como “disk.go” y ejecutándolo te proporcionará:

```
$ go run disk.go
 Total space: 87352057856
 Free space : 35807154176
```

La nueva pantalla LCD de estado en el CloudShell

¡Este proyecto me ha resultado muy divertido, ya que me ha dado la oportunidad de probar cosas nuevas como CGO! Espero que tú también le hayas sacado partido. El proyecto al que aludí al principio se puede encontrar en GitHub en **http://bit.ly/2hEUa6B**.

Stereo Boom Bonnet, Una estupenda forma de disfrutar de la música en tu ODROID

December 1, 2017 By Justin Lee Mecaniqueo

El I2S 2Watt Stereo Boom Bonnet Kit (**https://goo.gl/1mXXVH**) es un compacto sistema de altavoces para ODROID-XU4 y ODROID-C1+/C2. Utiliza I2S como estándar de sonido digital para la salida de audio. Es muy fácil de instalar y te encontraras "rockeando" en 15 minutos. Para conectarlo a tu ODROID, sigue estos pasos:

- Conecta los altavoces estéreo de 2×2 vatios y 4 ohmios al conector de la placa Boom Bonnet y fíjalos con pegamento
- Conecta el cable plano GPIO por un extremo a la placa boom bonnet y el otro extremo a la placa ODROID
- Actualiza el sistema operativo a la última versión

Puedes ajustar fácilmente el nivel de salida de audio con un potenciómetro en la placa. Te proporcionará una calidad sonido aceptable, aunque puede añadir un poco de ruido al sistema, y por supuesto no reemplazará a lo que es un verdadero sistema de altavoces de alta fidelidad. El paquete incluye:

- Placa Stereo Boom Bonnet
- Mini altavoces de 2x2W/4 ohm (28mm de diámetro, 11.5mm de grosor)
- Separadores PCB de 3 x 5 mm
- Tornillos de 3 x 5 mm
- Cable plano GPIO 200mm de 7 pines (200mm) para la ODROID-C1+/C2 o la versión de 12 pines para el ODROID-XU4

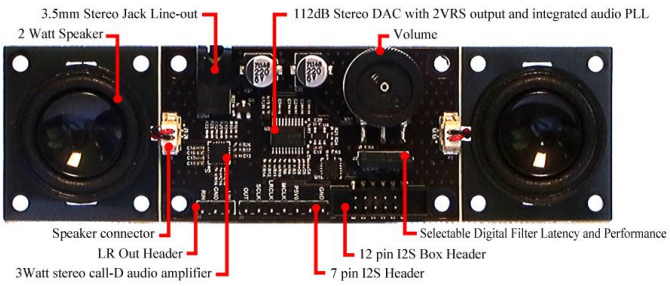


Figura 1 – Diagrama con anotaciones del Stereo Boom

Bonnet

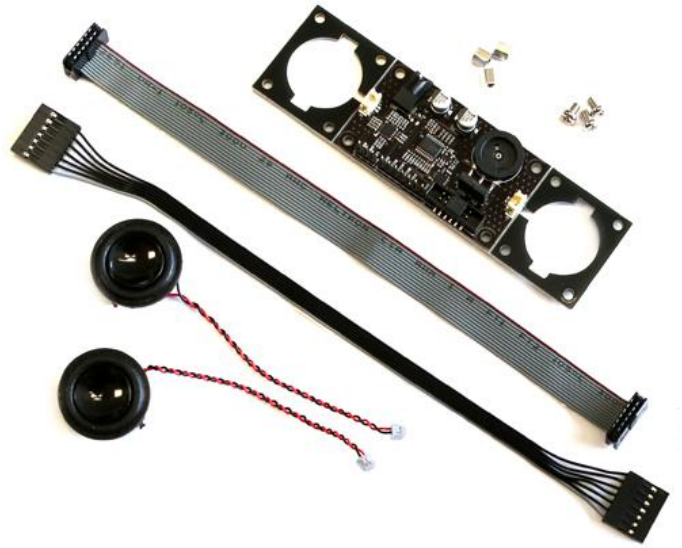


Figura 2 – El Stereo Boom Bonnet incluye los altavoces, los separadores y el cable plano que hayas elegido

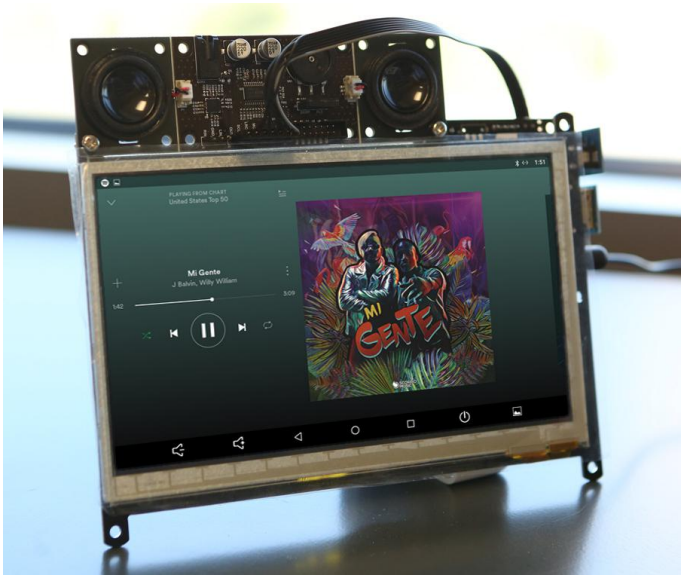


Figura 3 – El Stereo Boom Bonnet se monta fácilmente en el ODROID-VU7

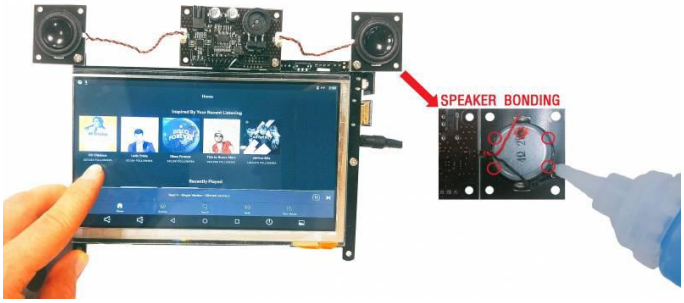


Figura 4 – Los altavoces Stereo Boom Bonnet también se pueden separar y fijar en cualquier lugar con algo de pegamento

Instalación

Conecta el stereo boom bonnet al ODROID-C1 +/C2 con un cable I2C, conecta un teclado USB, un ratón USB y un monitor HDMI, después inicia el sistema y luego actualícelo:

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

Figura 5 – Conectando el Stereo Boom Bonnet a un ODROID-C1+/C2to an ODROID-C1+/C2

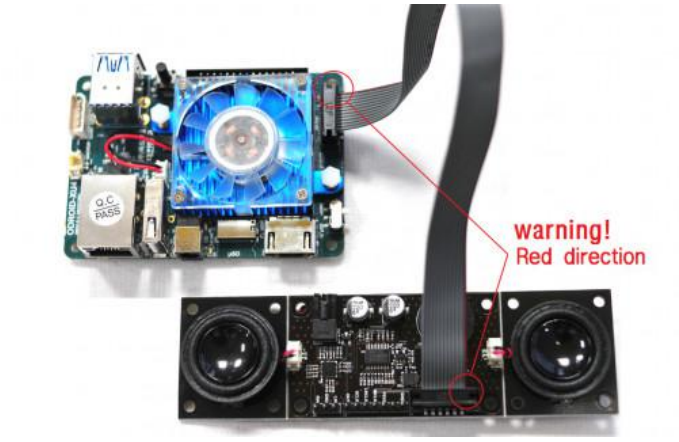


Figura 6 – Conectando el Stereo Boom Bonnet a un ODROID-XU4

A continuación, asegúrate de que los módulos de kernel del stereo boom bonnet estén cargados:

```
$ aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: ODROIDHDMI [ODROID-HDMI], device 0:
I2S.27 dit-hifi-0 []
  Subdevices: 0/1
  Subdevice #0: subdevice #0
odroid@odroid64:~$
odroid@odroid64:~$ sudo modprobe snd-soc-
pcm5102
odroid@odroid64:~$ sudo modprobe snd-soc-
odroid-dac
odroid@odroid64:~$ aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: ODROIDHDMI [ODROID-HDMI], device 0:
```

```
I2S.27 dit-hifi-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: ODROIDDAC [ODROID-DAC], device 0:
I2S.27 pcm5102-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

Luego, ve a Applications → Sound & Video → Sound → Hardware Tab & Output Tab y selecciona “ODROID-DAC”. Si quieres cargar el driver cada vez que se inicie tu ODROID-C1 +/C2, puedes registrar el driver dentro de /etc/modules y reiniciar:

```
$ su
Password:  (root password is "odroid")
# echo "snd-soc-pcm5102" >> /etc/modules
# echo "snd-soc-odroid-dac" >> /etc/modules
# exit
```

Después de reiniciar, comprueba el driver con el siguiente comando:

```
$ aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: ODROIDHDMI [ODROID-HDMI], device 0:
I2S.27 dit-hifi-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: ODROIDDAC [ODROID-DAC], device 0:
I2S.27 pcm5102-0 []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

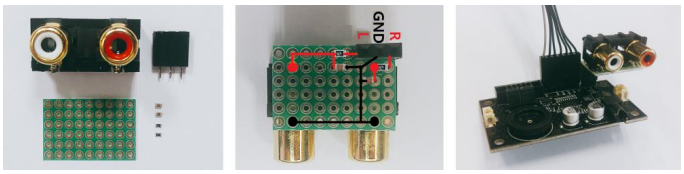


Figura 7 – Añadiendo un conector RCA para la salida del Stereo Boom Bonnet

Los esquemas están disponibles en **https://goo.gl/pxTTd9**, y la información detallada la puedes encontrar en la página Wiki en **https://goo.gl/JY1Y7B**.

Boom Box: Ingeniería de sonido con un mejor altavoz

December 1, 2017 By @Technicavolous Mecaniqueo, Tutoriales

Para mejorar el sonido del Stereo Boom Bonnet de Hardkernel (**https://goo.gl/TrDU8u**), fui a mi tienda de hobby local y me hice con un "pack variado" de estireno que resultaba tener algunas hojas .080mil y algunas tuberías dentro. También compré un poco de disolvente de estireno para soldar. Lo hice todo a ojo, de modo que algunas esquinas no están del todo pulidas. No obstante, está hecho con la intención de ser introducido en otra carcasa, sólo con el frontal visible. Cuando esté completado, pegaré un trozo de tela en el frontal y volveré a montar el altavoz con los tornillos de cabeza hexagonal. Las carcasas serán para controlar las vibraciones y la acústica. Si tuviera que hacer esto en una carcasa independiente, probablemente lo haría a la mitad de profundidad, pero elegí este tamaño por la profundidad de la carcasa. No creo que haya mucha diferencia de sonido con un tubo más largo.

Una cosa que no aparecen en las fotos es el algodón desmenuzado. Cuando el proyecto esté completado, extenderé los cables hacia un conector situado en la parte posterior, y toda la carcasa se rellenará con algodón con la finalidad de forma una pantalla acústica y así poder evitar el sonido de una "caja hueca".

Deliberadamente estoy omitiendo las medidas, porque no importan demasiado. Creo que sería mejor reemplazar el tubo por una simple hoja plana dejando un hueco desde la parte inferior de un cuarto de diámetro del altavoz y la misma separación desde la parte posterior. Puede encontrar diseños e ideas en Internet si desea profundizar más en este tema.

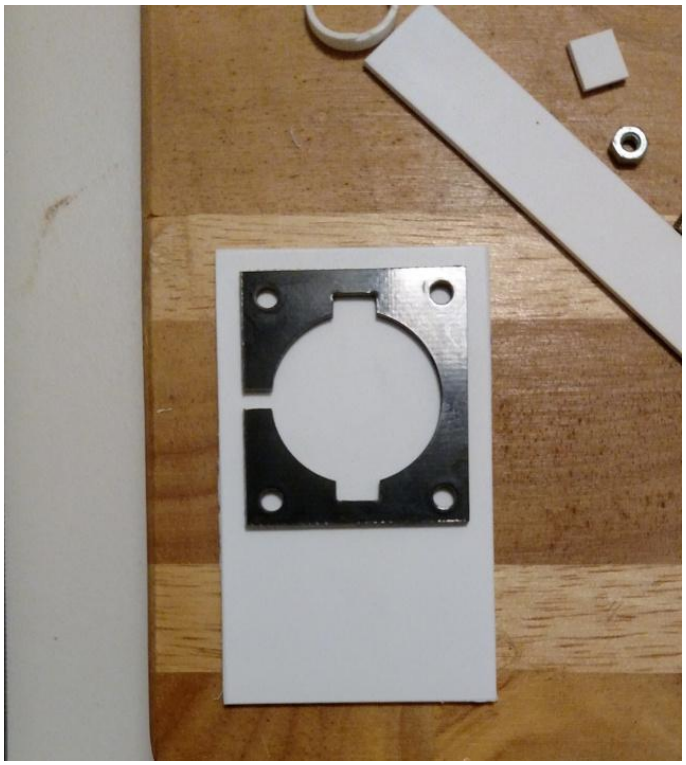


Figura 1 – Simplemente coloqué la pequeña placa sobre el panel cortado y la marqué con un sharpie

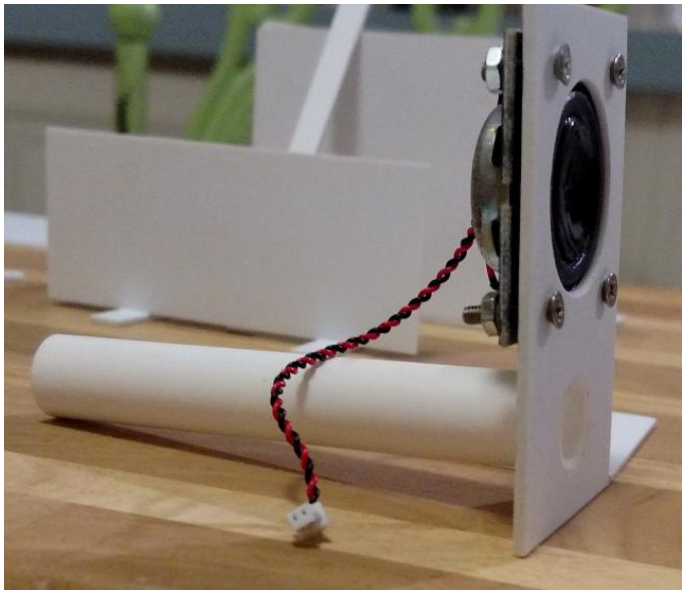


Figura 2 – Primer plano del tubo pegado

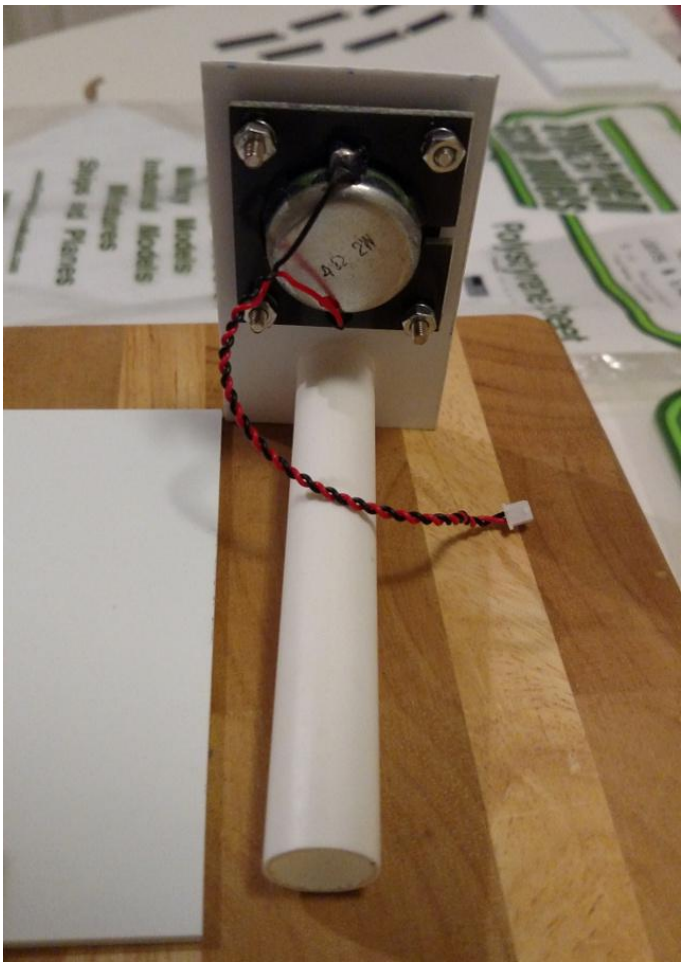


Figura 3 – Otra vista

Figura 4 – Vista superior con el panel frontal dentro de la carcasa casi completada

Figura 5 – El resultado final del proyecto es una carcasa de aspecto amigable con un gran sonido

Puedes hacer algo como esto con cualquier altavoz, incluido el Stereo Boom Bonnet, y el sonido mejorará al instante. Para experimentar un poco con el sonido, desplace el panel frontal hacia atrás ya que aún no había prolongado los cables y el sonido era increíble. Intente hacer un pequeño bajo, por debajo de 100Hz, pero si colocaba mi dedo sobre el tubo, la mayor parte del bajo desaparecía, de modo que la cavidad parecía estar funcionando como debería. Una vez que extendiera los cables por la parte posterior y el tubo estuviese colocado correctamente dentro de la carcasa, los bajos deberían oírse un poco mejor.

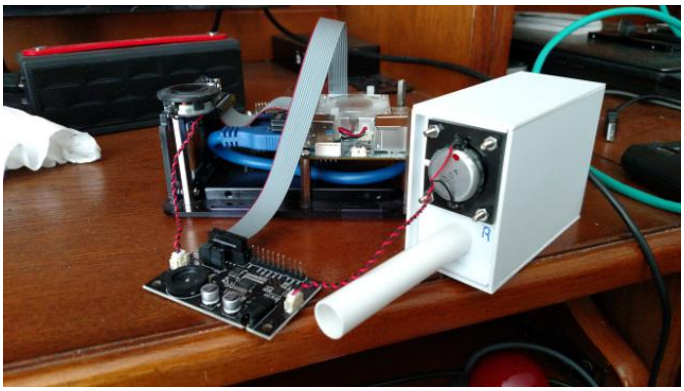
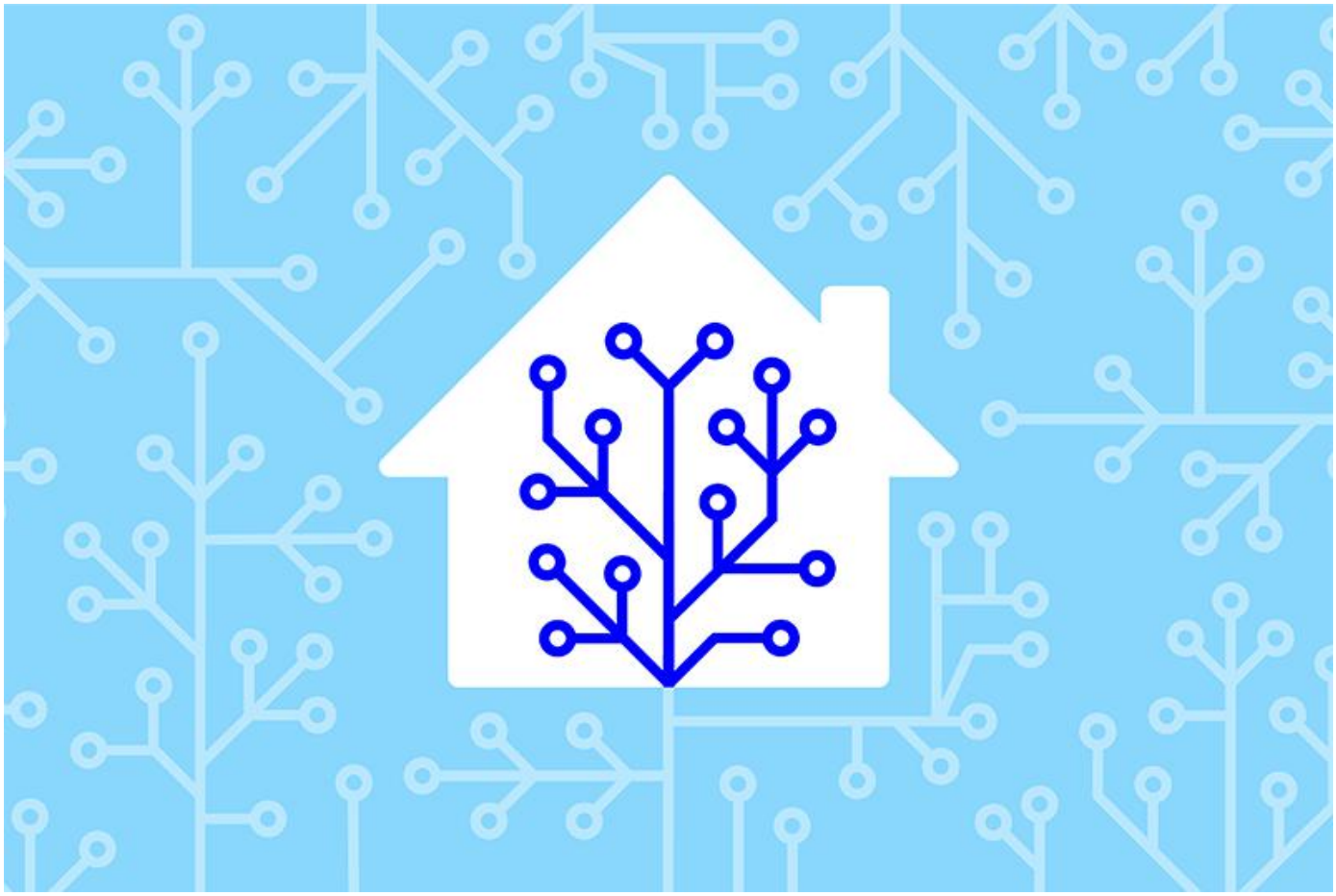


Figura 6 – Al echar hacia atrás el panel frontal, mejoró la respuesta de los bajos

Para comentarios, preguntas y sugerencias, visita el post original en **https://goo.gl/yyp22n**.

Home Assistant: Usando Infrarrojos, Motores y Relés

December 1, 2017 By Adrian Popa Mecaniqueo



En este artículo, vamos a combinar el ODROID con algo de hardware y Home Assistant, para empezar a convertir dispositivos no inteligentes en dispositivos "inteligentes". Trabajaremos con cables, soldaremos un poco, conectaremos relés y en algunos casos, llegaremos a trabajar con voltaje, ¡así que ten cuidado!

Convirtiendo una unidad de aire acondicionado e un IoT AC

Si tienes una vieja unidad de aire acondicionado (AC), o si estás pensando en comprar un nuevo modelo, es posible que quieras controlarlo desde cualquier lugar. Por ejemplo, puedes encender el AC desde tu teléfono cuando estés saliendo del trabajo o cuando regreses de unas largas vacaciones. Puedes comprar una unidad AC Wi-Fi, pero éstas son aproximadamente 200$ más caras que una unidad normal sin WiFi. En su lugar vamos a convertir una unidad AC sin WiFi, LG P12RL.NSB, en una unidad inteligente. Esto lo llevaremos a cabo controlando la unidad AC a través de un ODROID-XU4, un blaster infrarrojos (IR) y Home Assistant. El blaster IR se puede usar para controlar cualquier dispositivo que disponga de un mando a distancia, no solo una unidad de Aire Acondicionado.

Figura 1 – Mando a distancia original del AC (model AKB73456113)

Lo primero que debe tener en cuenta es cómo se comunica el mando a distancia con la unidad AC. Hay

dos métodos: el primero es cuando se presiona un botón en el mando a distancia, el estado completo se envía a través del IR (temperatura, velocidad del ventilador, potencia, etc.). La otra opción es que cuando se presiona un botón, únicamente se envía la acción en cuestión (aumentar la temperatura, encender el ventilador, etc.). Puede verificar esto, por ejemplo, enviando la correspondiente orden para encender el ventilador a toda velocidad, luego apaga el ventilador, pero con el mando a distancia fuera del alcance del dispositivo, después envía una orden para cambiar a una temperatura diferente. Si el ventilador continúa funcionando a toda velocidad, entonces es que el mando a distancia no está enviando el estado completo. En el caso de que envíe el estado completo, el proyecto http://bit.ly/2AcpKAW puede ayudarte a decodificar la información de estado que se envía. En mi caso, el mando a distancia IR solo envía la información del botón que se esté presionando en ese momento, con la excepción del botón de encendido, que también envía la temperatura y el estado del ventilador.

El hardware

Diseñar y montar un blaster IR para una placa ODROID es un proceso relativamente sencillo y se encuentra documentado en la wiki http://bit.ly/2A6HHmR. Quería que mi implementación funcionase en un ODROID-XU4 o un ODROID-C2, así que necesitaba alimentarlo desde la línea de 5V y accionar un transistor desde el GPIO. El circuito y el montaje final lo puedes ver en la Figura 2.

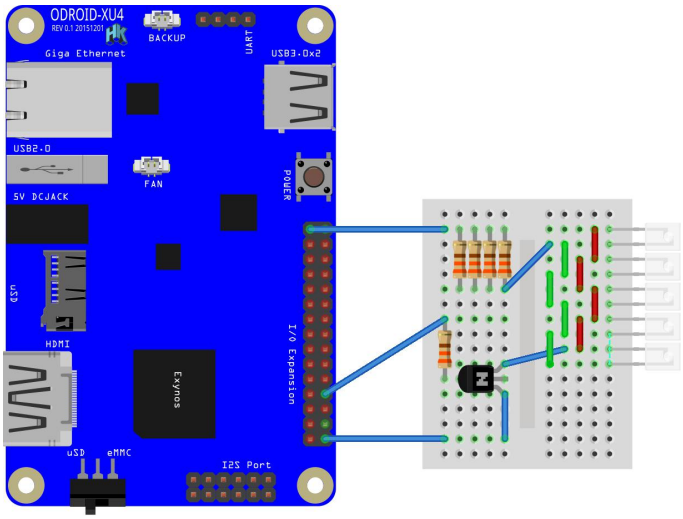


Figura 2 – Vista de la placa de pruebas blaster IR

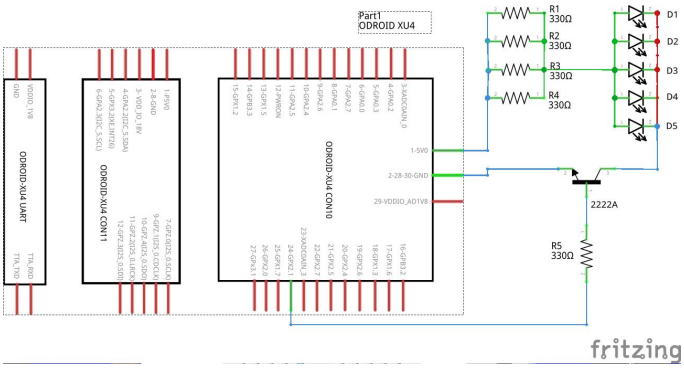


Figura 3 – Diagrama del circuito del blaster IR

He utilizado un extensor IR de mi TV Samsung para los leds IR. Después de abrirlo, descubrí que tenía 5 LED conectados en paralelo, pero la cuestión es que no tenía las especificaciones de los LED. Hice la estimación de que cada diodo podría pasar una corriente máxima de unos 30 – 50mA (basándome en unas especificaciones técnicas generales http://bit.ly/1trG4ZM), así que 5 podrían pasar entre 150 – 250mA. Es por ello que necesitaría una resistencia de aproximadamente unos 82Ω para limitar la corriente desde el ODROID. Si utilizas menos LEDs IR, necesitarás una resistencia más grande. Lamentablemente, estas resistencias tan bajas son difíciles de encontrar, así que conecté 4 resistencias de 330 Ω, desde el kit de pequeños ajustes C, en paralelo (R1 / R4). El transistor se controla con el pin 24 en el XU4, y es utilizado para modular la señal. Cuando se encienda el GPIO, la corriente fluirá y los LED IR se encenderán, y cuando el GPIO se apage, el transistor detendrá el circuito. La resistencia R5 está ahí para proteger el GPIO.

Una vez que consigas montar esto, puedes probar el hardware alternando manualmente el GPIO 24 a través de sysfs y usar la cámara de teléfono para filmar los leds. La luz IR debe ser visible con la cámara y tiene un tono azulado. Puedes montarlo sin una placa de prueba e incluso puedes colocar el transistor y las resistencias dentro de la carcasa del XU4, pero resulta bastante más complicado. Rompí los puntos de soldadura dos veces mientras intentaba colocarlo dentro de la carcasa. El siguiente fragmento de código permite activar y desactivar manualmente el pin 24 GPIO:

```
$ sudo su -
# cd /sys/class/gpio/
```

```
# echo 24 > export
# cd gpio24
# echo out > direction
# echo 1 > value
# echo 0 > value
```

Integración LIRC

Ahora tenemos que decirle a LIRC que puede usar el Blaster IR para enviar datos. Para hacer esto, seguí las instrucciones de la wiki: **http://bit.ly/2A6HHmR**

```
$ sudo apt-get install lirc
$ sudo vi /etc/lirc/hardware.conf
#Chosen IR Transmitter
TRANSMITTER="ODROID blaster"
TRANSMITTER_MODULES="lirc_odroid lirc_dev"
TRANSMITTER_DRIVER=""
TRANSMITTER_DEVICE="/dev/lirc0"
TRANSMITTER_SOCKET=""
TRANSMITTER_LIRCD_CONF=""
TRANSMITTER_LIRCD_ARGS=""
```

Antes de reiniciar LIRC, debemos pasar los parámetros correctos al módulo lirc_ODROID. Podemos hacer esto creando el archivo /etc/modprobe.d/lirc.conf con el siguiente contenido:

```
options lirc_odroid gpio_out_pin=24
softcarrier=1 invert=0
```

Intenta reiniciar LIRC y monitoriza dmesg para detectar cualquier error:

```
$ sudo systemctl enable lirc
$ sudo service lirc restart
```

Si dmesg muestra errores como los que aparecen a continuación, significa que algún otro driver ha reclamado el PIN 24 de GPIO.

```
[   25.322482] lirc_dev: IR Remote Control
driver registered, major 245
[   25.336230] lirc_odroid: module is from the
staging directory, the quality is unknown, you
have been warned.
[   25.346335] lirc_odroid: cant claim gpio
pin 24
[   25.350461] lirc_odroid: init port fail!
[   25.353337] lirc_odroid[lirc_ODROID_exit]
```

En mi caso, era el módulo 1wire, que se puede desactivar añadiendo las siguientes líneas a /etc/modprobe.d/blacklist-odroid.conf y reiniciando:

```
# 1 wire
blacklist w1_gpio
blacklist wire
```

Obtener los códigos del mando a distancia

Una vez que el transmisor este listo, es hora de conseguir los códigos del mando a distancia para su control remoto. Los códigos consisten en una serie de intervalos cuando la señal está activada o desactivada medidos en microsegundos. Puedes conseguir estos códigos de varias fuentes online, o puede grabarlos con LIRC y un receptor IR. Yo usé el receptor IR en un C2 para grabar cada código en un archivo diferente, presioné CTRL + C para salir del modo2 y lo limpié eliminando la primera fila. Cuando terminé, añadí las entradas a lircd.conf:

```
c2$ sudo apt-get install lirc
c2$ sudo service lirc stop
c2$ sudo mode2 -m -d /dev/lirc0 | tee power-on
c2$ sudo sed -i '1d' power-on
```

Esto puede ser un poco tedioso, ya que tuve que registrar los códigos de encendido/apagado, desde la temperatura-18 a la temperatura-30, ventilador-bajo, ventilador-medido, ventilador-alto, balanceo-encendido/apagado, jet-encendido/apagado, ionizador encendido/apagado. No obstante, una vez hecho, los agrupé todos en un archivo de configuración, (/etc/lirc/lircd.conf) tal y como se muestra en **http://bit.ly/2A9MK3r**. Puedes reiniciar LIRC y probar a enviar los códigos con irsend:

```
$ sudo service lirc restart
$ irsend LIST lgirplus.conf ""
$ irsend SEND_ONCE lgirplus.conf power-on
$ irsend SEND_ONCE lgirplus.conf power-off
```

En caso de que el último comando irsend falle/agote el tiempo de espera, es posible que te hayas topado con un error de LIRC/driver. La solución es sencilla y rápida, reiniciar LIRC antes de inyectar cada comando.

Integración en Home Assistant

Si el Blaster LIRC está conectado al mismo dispositivo donde tienes instalado Home Assistant, puedes usar

el componente Shell Command (http://bit.ly/2vOFnhe) para emitir comandos IR desde Home Assistant. En mi caso, LIRC se ejecuta en un sistema diferente al del Home Assistan, así que desarrollé un script Python que se comunicara con Home Assistant a través de MQTT y emitiera los comandos irsend.

El código completo para este agente MQTT está disponible en http://bit.ly/2Ax8SYz. El código obtiene datos de configuración desde /etc/ir-ac-mqtt-agent.yaml, luego se conecta con MQTT al broker. Para saber cómo configurar el bróker y Home Assistant, consulta el artículo de ODROID Magazine en http://bit.ly/2A6ql9I. También define dos funciones de retrollamada:

- on_connect – registra una lista de temas MQTT para escuchar los comandos
- on_message – se llama cada vez que se recibe un mensaje MQTT de los temas registrados.

El script también almacena un diccionario interno con el estado actual, como es el encendido y la temperatura. Esto le permite reaccionar mejor a los comandos entrantes. Por ejemplo, si recibe un comando de apagado, pero la alimentación ya está desconectada, ignorará el comando para evitar que la unidad AC emita un pitido.

La lógica interna intenta simular lo que hace el mando a distancia físico ignorando los comandos a menos que la alimentación esté conectada, y también ajusta la temperatura a 18 ° C y el ventilador al máximo cuando se activa el "Modo Jet". Además, se han realizado algunas simplificaciones, como cuando se envía el comando de encendido, la temperatura se ajusta a 21 ° C y el ventilador al máximo debido a que señal de encendido codifica parte del estado del mando a distancia. El script del agente escucha los comandos emitidos sobre temas tales como ha/lg_ac/ionizer/set y envía un feedback sobre ha/lg_ac/ionizer/get, de modo que la interfaz web incluye comentarios de los comandos recibidos.

Para instalar el agente MQTT, puedes usar estos comandos:

```
$ sudo wget -O /usr/local/bin/ir-ac-mqtt-agent.py https://raw.githubusercontent.com/mad-ady/home-assistant-customizations/master/external-scripts/ir-ac-mqtt-agent.py
$ sudo chmod a+x /usr/local/bin/ir-ac-mqtt-agent.py
$ sudo apt-get install python-pip python-yaml
$ sudo pip install paho-mqtt
$ sudo wget -O /etc/ir-ac-mqtt-agent.yaml https://github.com/mad-ady/home-assistant-customizations/blob/master/external-scripts/ir-ac-mqtt-agent.yaml
$ sudo wget -O /etc/systemd/system/ir-ac-mqtt-agent.service https://github.com/mad-ady/home-assistant-customizations/blob/master/external-scripts/ir-ac-mqtt-agent.service
```

Tómate tu tiempo para realizar los cambios necesarios en /etc/ir-ac-mqtt-agent.yaml, luego activa e inicia el servicio:

```
$ sudo systemctl enable ir-ac-mqtt-agent
$ sudo systemctl start ir-ac-mqtt-agent
```

En Home Assistant, configuraremos varios switches MQTT (http://bit.ly/2AwtDUd) para manejar la alimentación, el modo Jet, el ionizador, el balanceo, un input_select (http://bit.ly/2zEfgNA), para seleccionar la velocidad del ventilador y un input_number (http://bit.ly/2k0vfOY), para ajustar la temperatura deseada. Los switches comunican directamente su estado al script backend a través de MQTT, mientras que los demás componentes utilizan la automatización para activar los mensajes MQTT sobre los cambios. Aquí tienes la configuración del componente:

```
switch:
  - platform: mqtt
    command_topic: 'ha/lg_ac/power/set'
    state_topic: 'ha/lg_ac/power/get'
    payload_on: 'ON'
    payload_off: 'OFF'
    name: 'AC Power'
    retain: false
  - platform: mqtt
    command_topic: 'ha/lg_ac/ionizer/set'
    state_topic: 'ha/lg_ac/ionizer/get'
```

```
    payload_on: 'ON'
    payload_off: 'OFF'
    name: 'AC Ionizer'
    retain: false
  - platform: mqtt
    command_topic: 'ha/lg_ac/jet/set'
    state_topic: 'ha/lg_ac/jet/get'
    payload_on: 'ON'
    payload_off: 'OFF'
    name: 'AC Jet'
    retain: false
  - platform: mqtt
    command_topic: 'ha/lg_ac/swing/set'
    state_topic: 'ha/lg_ac/swing/get'
    payload_on: 'ON'
    payload_off: 'OFF'
    name: 'AC Swing'
    retain: false

input_select:
  lg_ac_fan_mode:
    name: Fan mode
    options:
      - cycle
      - low
      - med
      - high
    initial: 'low'

input_number:
  lg_ac_temperature:
    name: AC Temperature
    initial: 22
    min: 18
    max: 30
    step: 1
```

Podemos agrupar todos estos elementos en una vista independiente:

```
group:
…
 lg_ac:
    name: Air Conditioning
    view: yes
    icon: mdi:snowflake
    entities:
      - group.lg_ac_group
 lg_ac_group:
    name: LG AC
    entities:
      - switch.ac_power
      - input_number.lg_ac_temperature
      - input_select.lg_ac_fan_mode
      - switch.ac_jet
      - switch.ac_ionizer
      - switch.ac_swing
```

Y tras reiniciar Home Assistant, debería parecerse a la siguiente imagen.

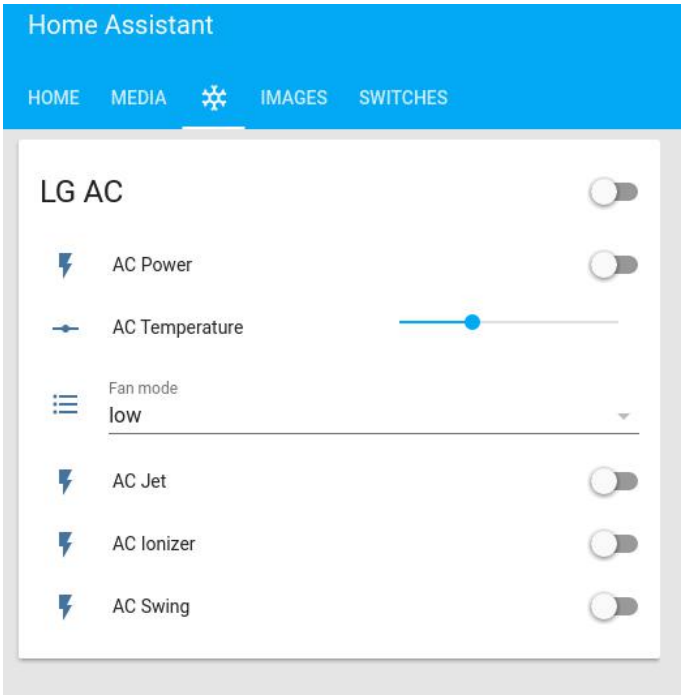


Figura 4 – Soporte básico para aire acondicionado

```
- action:
  - alias: LG AC MQTT Set Temperature
    data:
      payload_template: '{{
states.input_number.lg_ac_temperature.state
}}'
      qos: 0
      retain: true
      topic: ha/lg_ac/temperature/set
    service: mqtt.publish
  alias: LG AC Set IR temperature
  id: '1499081218012'
  trigger:
  - entity_id: input_number.lg_ac_temperature
    platform: state
- action:
  - alias: LG AC MQTT Set Fan
    data:
      payload_template: '{{
states.input_select.lg_ac_fan_mode.state }}'
      qos: 0
      retain: true
      topic: ha/lg_ac/fan/set
```

```
    service: mqtt.publish
  alias: LG AC Set IR Fan
  id: '1499152161'
  trigger:
  - entity_id: input_select.lg_ac_fan_mode
    platform: state
- action:
  - alias: LG AC Set temperature slider
    service: input_number.set_value
    data_template:
      entity_id:
input_number.lg_ac_temperature
      value: '{{trigger.payload}}'
  alias: LG AC Read temperature via MQTT
  id: '1499423002'
  trigger:
  - platform: mqtt
    topic: ha/lg_ac/temperature/get
- action:
  - alias: LG AC Set fan combo box
    service: input_select.select_option
    data_template:
      entity_id: input_select.lg_ac_fan_mode
      option: '{{trigger.payload}}'
  alias: LG AC Read temperature via MQTT
  id: '1499423003'
  trigger:
  - platform: mqtt
    topic: ha/lg_ac/fan/get
```

Las primeras dos automatizaciones envian los valores de los componentes de temperatura y ventilador a través de MQTT en el cambio de estado, mientras que las dos últimas automatizaciones reciben datos de temperatura y del ventilador a través de MQTT para actualizar la interfaz web. Reiniciando nuevamente debería permitirte contar con un sistema AC totalmente funcional controlado por Home Assistant. Aquí tienes un video de un prototipo en acción: https://youtu.be/zGRlhILVRCQ.

Añadir un temporizador de inicio y parada

El mando a distancia del AC tiene una opción para iniciar y detener el AC con un temporizador. También podemos configurar esto dentro de Home Assistant con unas cuantas automatizaciones y algunos componentes adicionales. Lo ideal sería usar el componente input_datetime (**http://bit.ly/2A8Mmoc**),para permitirle al usuario seleccionar la hora de inicio y parada, pero cuando escribí este artículo, el componente no funcionaba muy bien. En HA 0.56, usaremos input_text en su lugar (**http://bit.ly/2i8lcXI**), con una expresión regular que permite escribir una hora. También hay dos input_booleans (**http://bit.ly/2Bnd4Yj**), que parecen ser switches y que permiten al usuario activar/desactivar la funcionalidad. Aquí tienes la configuración que va dentro de configuration.yaml:

```
input_text:
  lg_ac_on_timer:
    name: LG AC on timer
    initial: '16:00'
    pattern: '^[0-9]{1,2}:[0-9]{1,2}$'
  lg_ac_off_timer:
    name: LG AC off timer
    initial: '18:00'
    pattern: '^[0-9]{1,2}:[0-9]{1,2}$'

input_boolean:
  ac_on_timer_active:
    name: Activate AC On timer
    initial: off
    icon: mdi:calendar
  ac_off_timer_active:
    name: Activate AC Off timer
    initial: off
    icon: mdi:calendar

group:
…
  lg_ac:
…
    entities:
…
      - group.lg_ac_timer
  lg_ac_timer:
    name: AC Timer
    entities:
      - input_boolean.ac_on_timer_active
      - input_text.lg_ac_on_timer
      - input_boolean.ac_off_timer_active
      - input_text.lg_ac_off_timer
```

El resultado final después de reiniciar Home Assistant debería parecerse a la siguiente imagen:

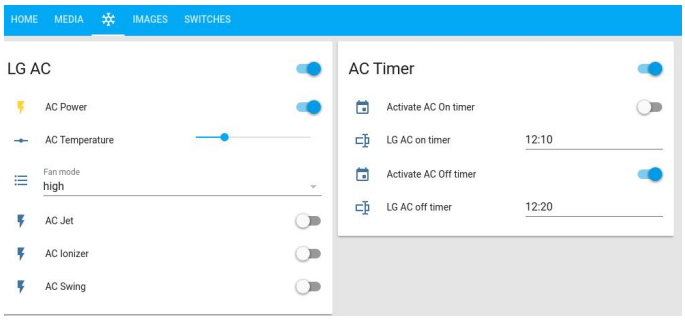


Figura 5 – Controles del AC con temporizadores

Para que funcione, deberás añadir las siguientes dos automatizaciones en automations.yaml. La primera automatización enciende el AC, el modo Jet y también apaga el temporizador que activa el AC, de modo que se trata de un evento de un solo disparo. Comprueba cada minuto si ac_on_timer_active está activado y si la hora actual es la misma que la cadena dentro de lg_ac_on_timer. La segunda automatización hace algo similar para el temporizador de apagado, pero con diferentes acciones.

```
- action:
  - service: switch.turn_on
    entity_id:
    - switch.ac_power
  - service: switch.turn_on
    entity_id:
    - switch.ac_jet
  - service: input_boolean.turn_off
    entity_id:
    - input_boolean.ac_on_timer_active
  alias: Turn On AC on timer
  id: '1502194970'
  trigger:
  - platform: time
    minutes: /1
    seconds: 0
  condition:
  - condition: state
    entity_id:
input_boolean.ac_on_timer_active
    state: 'on'
  - condition: template
    value_template: '{{
now().strftime("%H:%M") ==
states.input_text.lg_ac_on_timer.state
      }}'
- action:
  - service: switch.turn_off
    entity_id:
    - switch.ac_power
  - service: input_boolean.turn_off
    entity_id:
    - input_boolean.ac_off_timer_active
  alias: Turn Off AC on timer
  id: '1502194971'
  trigger:
  - platform: time
    minutes: /1
    seconds: 0
  condition:
  - condition: state
    entity_id:
input_boolean.ac_off_timer_active
    state: 'on'
  - condition: template
    value_template: '{{
now().strftime("%H:%M") ==
states.input_text.lg_ac_off_timer.state
      }}'
```

Tras reiniciar Home Assistant una vez más, deberías tener un completo sistema AC “inteligente” el cual puedes disfrutar durante las olas de calor. Aunque las automatizaciones anteriores son bastante básicas, puede añadir más lógica, como añadir un termostato (http://bit.ly/2Ay6kJN), y monitorizar el clima exterior con un sensor o haciendo uso del pronóstico del tiempo, para que puedas encender tu Aire Acondicionado cuando el clima exterior esté por encima de un umbral. De hecho, probaremos un termostato de este tipo en nuestro próximo proyecto.

Controlar un calentador de gas con un termóstato Home Assistant

Como para muchos lectores es invierno, puede que les interese tener un calentador inteligente. Vamos a controlar una caldera de gas natural conectada a una calefacción central, un Viessman Vitopend 100 (http://bit.ly/2Bd9X4d). El calentador, si está encendido, intentará mantener el agua a una temperatura constante, la cual fluye a través de los elementos de calefacción de la casa, pero esto puede ser un gasto innecesario si no hay nadie en casa o si la casa está muy bien aislada. En su lugar, puedes usar un termóstato para encender o apagar el calentador en función de la temperatura ambiente. Mi calentador de gas tenía un termostato, Salus 091FLRF (http://bit.ly/2AbcnkC). Mi termóstato funciona, pero tiene algunos fallos. Para empezar, se

reiniciaría en mitad de la noche y pierde la configuración, dejándome helado. Un termóstato que se active por Internet tiene un coste de al menos 100$, de modo que vale la pena echar mano al bricolaje.

La ventaja de tener un termóstato externo es que tiene un relé conectado al calentador y no tienes que abrirlo. Si tiene que abrir tu calentador, asegúrate de llamar a un técnico especializado o puede que tengas problemas con tu proveedor de gas natural.

Tras analizar los esquemas del termóstato, era evidente que la comunicación con el calentador se hacía cerrando o abriendo un relé con una línea de 220V. Si añadimos un segundo relé en paralelo con el primero, puedo cerrar cualquiera de ellos para encender o apagar el calentador. ¿Por qué necesitas un relé? ¡Para que no frías tu placa ODROID! El planteamiento es usar un relé Sainsmart de 2 canales (**http://bit.ly/2A5WEFF**), para conectar el ODROID a la línea de 220V que va al calentador. Incluso si el relé tiene una potencia nominal de 5V, se puede usar de forma segura con un ODROID-C1 o con los GPIO de 3.3V del C2.

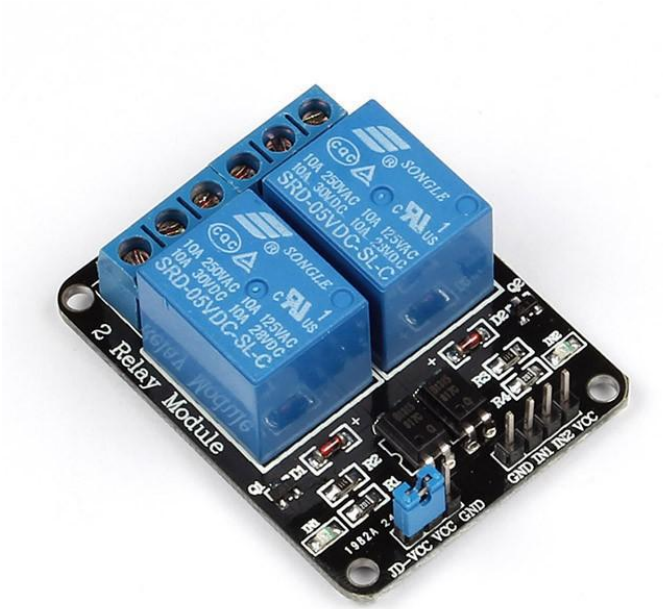


Figura 6 – Relé de 2 canales SainSmart

Como vas a trabajar con voltajes, asegúrate de tener un línea electríca certificada o que cumpla con la legislación de tu país. Además, desconecta siempre todos los aparatos de la red cuando vayas a trabajar con estas líneas. El esquema que vamos a implementar es bastante simple, tal y como se muestra en la Figura 7.

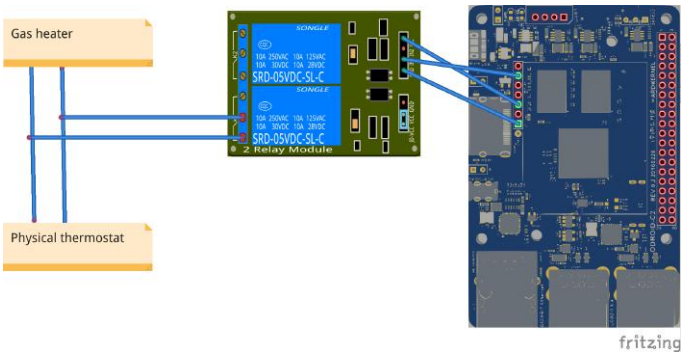


Figura 7 – Detalles de implementación

Solo hay una complicación con esta configuración. En lugar de utilizar los conectores GPIO de 40 pines en el C2 (cabezal J2), usé algunos pines del cabezal I2S (J7). La razón, en mi caso, es que el encabezado J2 fue utilizado por la pantalla de 3.5” de HardKernel (http://www.hardkernel.com/main/products/prdt_info.php?g_code=G147435282441).

Para convertir los pines I2S en pines GPIO, necesitaras editar el DTB cargado con el kernel. Sigue las siguientes instrucciones para editar el DTB, y también haz los cambios permanentes cuando actualices el kernel a través de apt, tal y como se describe en **http://bit.ly/2AyrWWz**.

```
$ sudo apt-get install device-tree-compiler
$ sudo fdtput /media/boot/meson64_ODROIDc2.dtb
/I2S status disabled
$ sudo fdtput /media/boot/meson64_ODROIDc2.dtb
/i2s_platform status disabled
$ sudo vi /etc/kernel/postinst.d/i2s-disable
#!/bin/bash

echo "Disabling I2S support from the device
tree"
/usr/bin/fdtput
/media/boot/meson64_ODROIDc2.dtb /I2S status
disabled
/usr/bin/fdtput
/media/boot/meson64_ODROIDc2.dtb /i2s_platform
status disabled
```

Una vez que reinicies, el cabezal J7 solo tendrá pines GPIO que podrás usar..

Controlar el relé a través de MQTT

Dado que no es el ODROID el que ejecuta Home Assistant, necesitaremos poder controlarlo a través de la red, con MQTT. El código del agente escucha mensajes ON/OFF sobre un tema específico y enciende o apaga el GPIO. Normalmente utilizaríamos wiringPI, (**http://bit.ly/2zBMm0F**), pero en este caso el conector J7 no está mapeado para WiringPI, así que usaremos sysfs. Los métodos pinMode y digitalWrite emulan sus equivalentes wiringPI para que el código sea más fácil de entender. El código está disponible en **http://bit.ly/2jofKfN**. Para instalarlo en tu sistema, sigue estos pasos:

```
$ sudo wget -O /usr/local/bin/heater-mqtt-
agent.py
https://raw.githubusercontent.com/mad-
ady/home-assistant-
customizations/master/external-scripts/heater-
mqtt-agent.py
$ sudo chown a+x /usr/local/bin/heater-mqtt-
agent.py
$ sudo wget -O /etc/heater-mqtt-agent.yaml
https://github.com/mad-ady/home-assistant-
customizations/blob/master/external-
scripts/heater-mqtt-agent.yaml
$ sudo wget -O /etc/systemd/system/heater-
mqtt-agent.service
https://github.com/mad-ady/home-assistant-
customizations/blob/master/external-
scripts/heater-mqtt-agent.service
$ sudo apt-get install python-pip python-yaml
$ sudo pip install paho-mqtt
```

Edita el archivo de configuración /etc/heater-mqtt-agent.yaml, configura los detalles de MQTT y luego inicia el agente con:

```
$ sudo systemctl enable heater-mqtt-agent
$ sudo systemctl start heater-mqtt-agent
```

Puedes monitorizar los mensajes del agente con el siguiente comando:

```
$ sudo journalctl -f -u heater-mqtt-agent
```

Integración de Home Assistant

Para utilizar este agente dentro de Home Assistant, puedes configurar un Switch MQTT y añadirlo dentro de tu propio grupo:

```
switch:
…
 - platform: mqtt
    command_topic: 'ha/heater/set'
    state_topic: 'ha/heater/get'
    payload_on: 'ON'
    payload_off: 'OFF'
    name: 'Heater'
    retain: true

group:
...
  heater:
    name: Gas heater
    view: yes
    icon: mdi:fire
    entities:
      - switch.heater
```

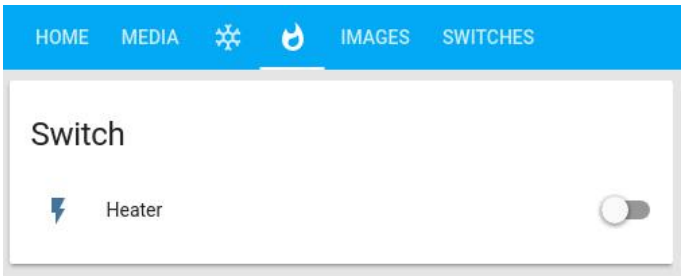


Figura 8 – Un switch para tu calentador

Tener un switch con el que puedes activar y desactivar resulta muy útil, pero es posible que prefieras un tener un termostato. Un termostato trata de mantener una temperatura entre ciertos niveles, y Home Assistant requiere un sensor de temperatura y un switch (**http://bit.ly/2Ay6kJN**). El sensor de temperatura puede ser cualquier sensor, incluida la temperatura exterior, pero en nuestro caso queremos usar datos de sensores de varias habitaciones. Tengo dos sensores de temperatura DS18b20 en dos habitaciones ya integrados en Home Assistant, tal y como describí en mi anterior artículo de ODROID en **http://bit.ly/2A6ql9I**. No obstante, el termostato puede funcionar con un sólo sensor de temperatura. Necesitaremos combinar los dos sensores en un único sensor, que devuelva la temperatura mínima entre los dos. Esto se puede extender fácilmente a múltiples sensores, de modo que el termostato eleva la temperatura de la habitación más fría hasta la temperatura deseada.

Podemos hacer esto con un sensor plantilla (http://bit.ly/2wPQLeY) dentro de configuration.yaml:

```
sensor:
…
  - platform: template
    sensors:
…
     house_temperature:
        friendly_name: Minimum house
temperature
        unit_of_measurement: '_C'
        value_template: '{{
(states.sensor.temperature_rest_python.state,
states.sensor.temperature_via_mqtt.state) |
min }}'

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

Añadir el termóstato ahora es una tarea muy sencilla (configuration.yaml):

```
climate:
  - platform: generic_thermostat
    name: Heater thermostat
    heater: switch.heater
    target_sensor: sensor.house_temperature
    min_temp: 15
    max_temp: 30
    target_temp: 24
    min_cycle_duration:
      minutes: 5
    tolerance: 0.3

group:
…
  heater:
…
    entities:
…
      - climate.heater_thermostat
```

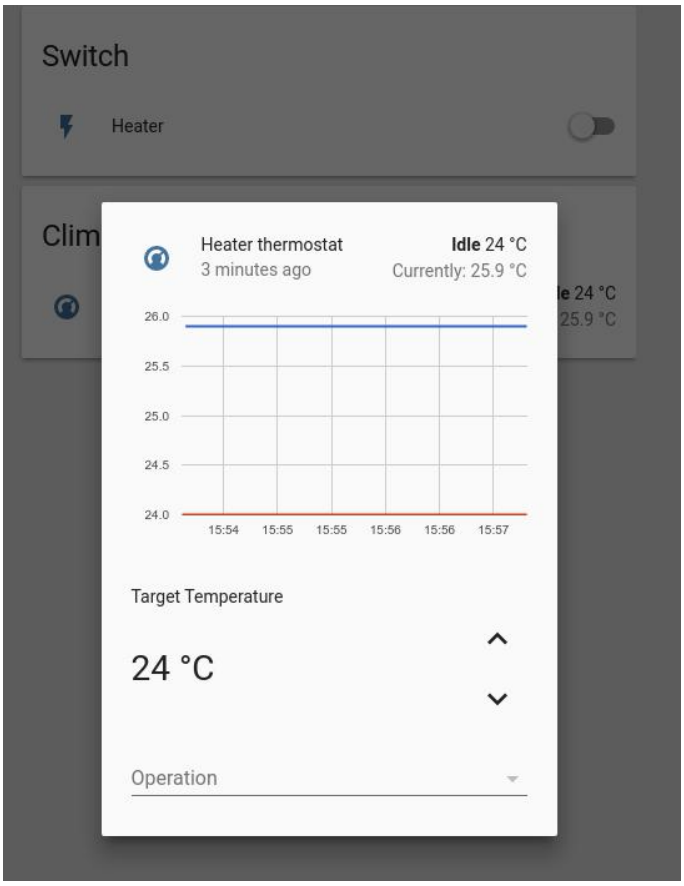


Figura 9 – Termostato

La configuración del termostato elevará la temperatura cuando ésta esté por debajo de target_temp – tolerancia, mantendrá el calentador encendido al menos min_cycle_duration, luego lo apagará cuando la temperatura esté por encima de target_temp + tolerancia. Puede usar un termostato similar con el AC al que hemos montado anteriormente, pero debes configurarlo con ac_mode: true para que éste sea un dispositivo de enfriamiento.

Puede controlar/configurar manualmente el termostato, pero lo más divertido es que puede usar automatizaciones para controlar el termostato. Un posible uso es fijar diferentes temperaturas objetivo basadas en posibles detenciones de presencia o la hora del día. Las siguientes automatizaciones ajustan la temperatura en 23ºC durante el día y 25.5ºC durante la noche (automations.yaml):

```
- action:
  - alias: climate.set_temperature
    data:
      entity_id: climate.heater_thermostat
      temperature: 23
    service: climate.set_temperature
  alias: Thermostat set low
```

```
  condition: []
  id: '1506943539788'
  trigger:
  - at: 07:00
    platform: time
- action:
  - alias: climate.set_temperature
    data:
      entity_id: climate.heater_thermostat
      temperature: 25.5
    service: climate.set_temperature
  alias: Thermostat set high
  condition: []
  id: '1506943638481'
  trigger:
  - at: '19:00'
    platform: time
```

Controlar la persiana de la ventana con Home Assistant

Un elemento físico más que podemos integrar en Home Assistant es la persiana monotizada de una ventana. En mi caso, los motores son controlados por un interruptor físico de tres vías. Cuando está en la posición hacia arriba, la persiana sube. En la posición central, los motores están apagados, y en la posición hacia abajo, la persiana baja. Quería usar un relé y tomar el control del desplazamiento de la persiana cuando el interruptor físico estubiese en la posición central, pero que también me permitiera contralar manualmente la persiana con el interruptor normal. Tras una larga discusión en el foro (http://bit.ly/2AwFCBb), me decanté por la implementación física que se muestra en la Figura 10.

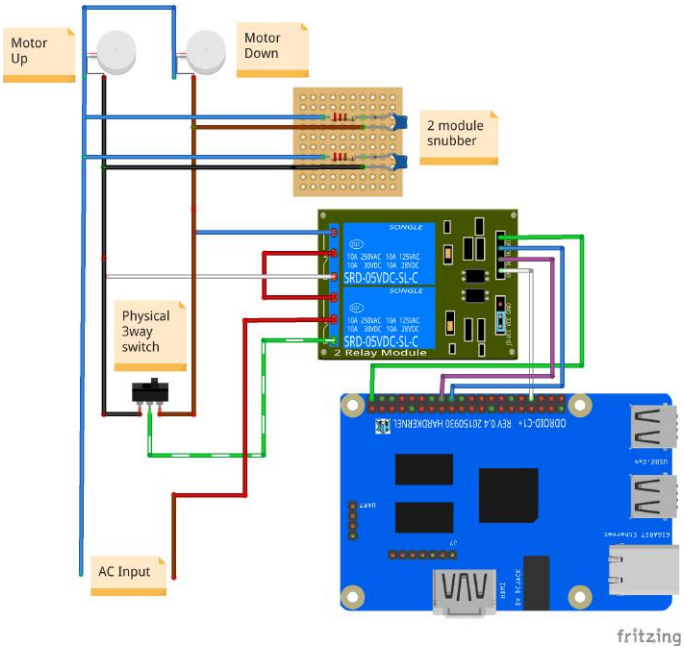


Figura 10 – Controlando un motor de persianas

Además del ODROID-C1+ y la placa de relés de 2 módulos, también necesitamos usar un par de amortiguadores. Los amortiguadores son circuitos RC que absorven la energía reactiva que se acumula en los motores durante la conmutación, de lo contrario los picos de potencia quemaran los contactos de los relés cuando intenten liberar la enegía, dañé un relé antes de darme cuenta de esto. Utilicé los amortiguadores descritos en t http://bit.ly/2A58tf4 en mi montaje.

Ten en cuenta que, aunque mi diseño minimiza la cantidad de componentes utilizados, es más arriesgado porque funciona a 220V. El usuario del foro @Jojo propuso un diseño alternativo que cambia el voltaje del interruptor físico a 5V y utiliza circuitos digitales para controlar el relé, el cual debería ser más seguro (http://bit.ly/2Bf2txo).

Agente MQTT para el funcionamiento de la persiana

Como es de costrumbre, necesitamos un script que se ejecute en el ODROID y que escuche los comandos de las persianas, enviados a través de MQTT, luego ejecuta los comandos que activan el relé e informe del estado. La diferencia con el anterior código es que esta vez tendremos que usar hilos de ejecución, más específicamente temporizadores. El modo más simple con el que el código puede funcionar es:

Se envía un mensaje para ABRIR o CERRAR la persiana desde Home Assistant El script activa el primer relé y asume el control de los motores, luego utiliza el segundo relé para controlar la dirección en la que funciona la persiana (hacia arriba o hacia abajo) El script espera un número determinado de segundos para que la persiana pueda abrirse o cerrarse al completo, lo cual demora 17 segundos en mi caso. Cuando el tiempo expira, el relé vuelve al modo manual y se envía una actualización del estado a través de MQTT a Home Assistant

La complicación aparece cuando quieres detener las persinas a mitad de camino o en un punto arbitrario. Para hacer esto, Home Assistant puede enviar un comando STOP, o una posición deseada, pero si el script tiene un único hilo de ejecución, éste esperará a que el motor termine antes de procesar el comando STOP. Al usar los temporizadores, después de que el motor arranque, un temporizador está programado para detener el motor en 17 segundos, y el código vuelve a escuchar mensajes MQTT. Si llega un mensaje STOP antes de que expire el temporizador, cancelará el temporizador y ejecutará inmediatamente el comando stopBlinds (). Además, la secuencia de comandos mantiene un estado interno de la posición de la persiana y la dirección del motor, así que si la persiana está a medio camino y deseas levantarla al 70%, podrás calcular el tiempo que necesita para ejecutar el motor y en qué dirección. El código está disponible en **http://bit.ly/2A88Ohk** y se puede instalar con:

```
$ sudo wget -O /usr/local/bin/blind-cover-
mqtt-agent.py
https://raw.githubusercontent.com/mad-
ady/home-assistant-
customizations/master/external-scripts/blind-
cover-mqtt-agent.py
$ sudo chown a+x /usr/local/bin/blind-cover-
mqtt-agent.py
$ sudo wget -O /etc/systemd/system/blind-
cover-mqtt-agent.service
https://raw.githubusercontent.com/mad-
ady/home-assistant-
customizations/master/external-scripts/blind-
cover-mqtt-agent.service
$ sudo wget -O /etc/blind-cover-mqtt-
agent.yaml
https://raw.githubusercontent.com/mad-
ady/home-assistant-
customizations/master/external-scripts/blind-
cover-mqtt-agent.yaml
$ sudo apt-get install python-pip python-yaml
$ sudo pip install paho-mqtt
```

Además de esto, necesitaras instalar la librería wiringPi y sus enlaces Python (**http://bit.ly/2AwVQui**). Tendrás que editar /etc/blind-cover-mqtt-agent.yaml e introducir tus datos MQTT, luego puedes activar e iniciar el agente:

```
$ sudo systemctl enable blind-cover-mqtt-agent
$ sudo systemctl start blind-cover-mqtt-agent
```

Podrás ver los mensajes de depuración con el siguiente comando:

```
$ sudo journalctl -f -u blind-cover-mqtt-agent
```

Configuración de Home Assistant

Home Assistant viene con un componente de Portada MQTT (**http://bit.ly/2jmAIf7**), que proporciona una interfaz básica para la portada. Puedes configurarlo con esta configuración dentro de configuration.yaml:

```
cover:
  - platform: mqtt
    name: "Blinds"
    state_topic: "ha/blind_cover/get"
    command_topic: "ha/blind_cover/set"
    set_position_topic:
"ha/blind_cover/position"
    assumed_state: true
    payload_open: 'OPEN'
    payload_close: 'CLOSE'
    payload_stop: 'STOP'
    state_open: 'open'
    state_close: 'closed'
group:
…
 blinds:
    name: Blinds
    view: yes
    icon: mdi:blinds
    entities:
      - cover.blinds
```

Después de reiniciar Home Assistant, el control de la persiana tendrá el aspecto de la Figura 11:

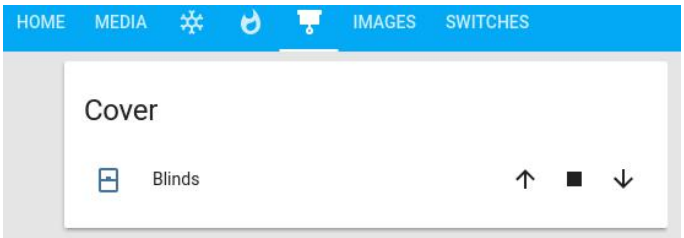


Figura 11 – Controles de la persiana

En https://youtu.be/MIhuELv1244, puedes ver una demo de las persianas. Como he mencionado antes, ¿existe alguna forma de detener en una posición arbitraria las persinas? Hay un modo modificar el componente de la persiana para que muestre un control deslizante que te permite controlar su posición. Para hacer esto, necesitas instalar la IU personalizada de Home Assistant, descargala de http://bit.ly/2AcqDJE e introduce los siguientes comandos:

```
$ sudo su - homeassistant
$ cd .homeassistant/
$ curl -o update-custom-ui.sh
"https://raw.githubusercontent.com/andrey-
git/home-assistant-custom-ui/master/update.sh?
raw=true"
$ chmod a+x update-custom-ui.sh
$ ./update-custom-ui.sh
```

Es necesario modificar la configuración para cargar el control de la persiana personalizado. En primer lugar, deberás activar los controles personalizados de la interfaz de usuario haciendo estos cambios en configuration.yaml:

```
customizer:
  custom_ui: local
```

```
Use the customize section to specify that all
covers should use the new UI (inside
configuration.yaml):

homeassistant:
  customize_glob:
    cover.*:
      custom_ui_state_card: state-card-custom-
ui
```

Por defecto, la página tendrá el mismo aspecto que la anterior, de modo que debemos activar manualmente el control deslizante (en la sección de personalización en configuration.yaml)::

```
homeassistant:
  customize:
    cover.blinds:
      state_card_mode: break-slider
      stretch_slider: true
```

Tras reiniciar Home Assistant, la interfaz de usuario tendrá un aspecto similar a la Figura 12.

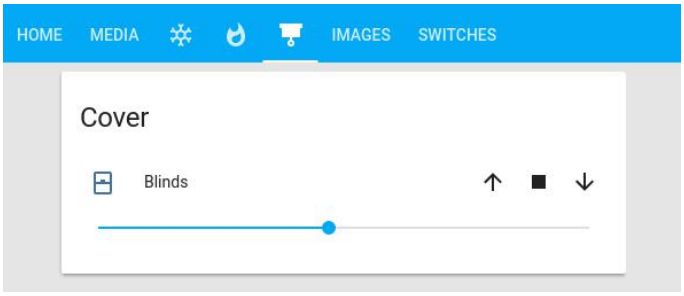


Figura 12 – Página con control deslizante

Como puede ver, con un poco de trabajo puede convertir tu corriente casa en una casa inteligente con la ayuda de ODROID y Home Assistant. Para comentarios y discusiones, visite el hilo original en http://bit.ly/2s13GbB.

Juegos Android: Stranger Things, Pocket Morty y Streets of Rage

December 1, 2017 By Bruno Doiche Android, Juegos

Es muy fácil relajarse con un ODROID porque existen muchas opciones de juegos para el sistema operativo Android. En este artículo, vamos a analizar tres nuevas entregas: **Stranger Things: The Game**, **Pocket Morty's**, y **Streets of Rage**.

Stranger Things: The Game

Directamente salido de un exitoso show de TV, nos llega un juego con un toque único. La emoción de que cuando alguien vuelve a un estudio de juegos contará con el mejor entretenimiento SIN compras dentro de la aplicación, anuncios o esos molestos banners.

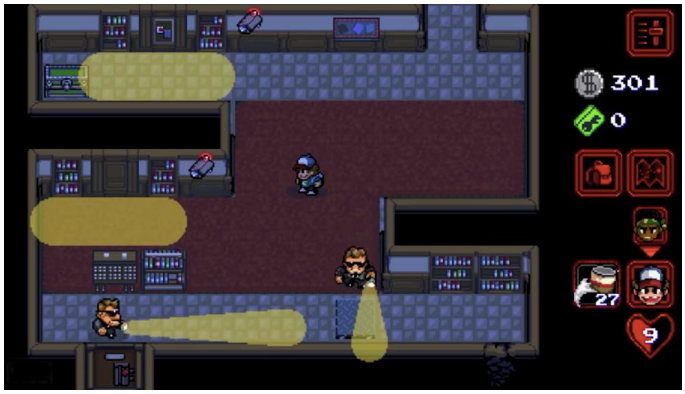
Stranger Things: The Game tiene una gama de colores que nos recuerda a la clásica SNES

Pero hablemos un poco del juego en sí, básicamente es una palangana de referencias a los 80, aunque el juego parece sacado de las clásicas aventuras de los años 90 a las que hemos jugado una y otra vez en las pantallas VGA de nuestros PCs o Amigas de antaño. Controlas a un puñado de personajes de la serie de TV que recorre la ciudad de Hawkins para resolver un gran misterio oculto, cada uno con habilidades únicas que con total seguridad, utilizarás para resolver

acertijos y vencer a tus enemigos. Los controles son muy simples, así que no te preocupes por hacerlos funcionar con tu pantalla táctil ODROID o con tu mando USB. Puedes configurarlos para que funcionen con el teclado, aunque yo no llegue a probarlo.

El juego sigue el modus operandi súper freestyle de la serie

Con varios capítulos, si eres un jugador mediocre, te llevará de cinco a diez horas completarlo, si no te quedas antes atascado en una de esas extrañas misiones secundaras. Pero si es demasiado fácil para ti, puede optar por el modo clásico, que describe en sí mismo la dureza de los años 80, un poco exagerado ya que en los 80 usas una contraseña para desplazarte por los capítulos, y por lo general pierdes con 3 golpes, 2 vidas y si el juego era benevolente, ¡un par de continuaciones! Sin embargo, el juego es muy llamativo y divertido. Pasaras un buen rato recogiendo elementos y básicamente disfrutando de un juego bastante decente basado en un show de televisión, que es algo muy raro hoy en día.

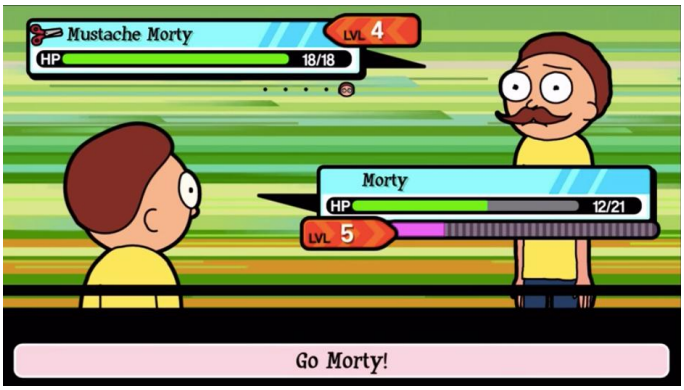


Los últimos capítulos no son nada fáciles, pero si yo lo logré, tú también puedes hacerlo.

Y sobre todo, Stranger Things: The Game hace que me aficione a una época en la que la forma de seguir disfrutando de tus show favoritos era teniendo un cartucho NES y jugarlo aquí y allá con tus amigos durante las vacaciones, éste sin duda es el mejor.

Pocket Mortys

Recuerdas cuando pokémon Go estaba de moda, y tuvimos que enseñarte a simular la ubicación de tu ODROID para poder ejecutar este juego que era tan rápido como lo fue su lanzamiento. Bueno, ¿y si en lugar de frustrarte cada vez que pierdas intentando ganar un combate en el gimnasio y tener que caminar de ciudad teniendo tus esperanzas azotadas por ese molesto zubat? ¡olvídalo! Pocket Mortys es el juego que capturará el espíritu original de Pokémon en un paquete que expande una historieta más grande, reluciente y alocada que, aunque es bastante antigua, este año resonó en los lectores como una auténtica bomba de neutrones.



¿Alguna vez hiciste gala de un bigote tan magnífico? ¿No? ¡Lucha contra él, Morty!

¿La trama es súper original? Jugarás a un juego Pokémon, pero con personajes de Rick y Morty. Capturas y coleccionas diferentes versiones de Morty, el co-protagonista de Rick y Morty, desde las muchas dimensiones alternativas, subiendo de nivel para recuperar tu arma portal y por la diversión de coleccionar a los Mortys. Es tan similar a Pokémon que realmente te aconsejo que te hagas con este juego antes de que los abogados de Nintendo descubran que se trata de una fiel reproducción de su IP. Y por supuesto, los vas a disfrutar, por simple que te parezca, de hecho, es bastante complicado tener un buen juego Pokémon. Es maliciosamente inteligente en lo que parece ser un juego gratuito.

El juego está tan bien diseñado que jurarás haber visto a los personajes con esos atuendos en la serie TV

Absurdamente gratificante, además de esta super jugabilidad piedra-papel-tijeras, este es un juego que reproduce la serie de televisión tan bien que casi podríamos decir que con el juego no hace falta ver la serie. (Lo cual no es cierto, deberías ver la serie una y otra vez)

Esta es la parte más divertida de cualquier juego, si te la saltas, verás que estás manejando un gadget dentro de otro gadget, super meta

Streets of Rage: Android Edition

De acuerdo, realmente no debería presentar este juego a cualquiera que esté leyendo esto, pero … Vamos, se trata streets of rage. Si viviste la época mágica de principios de los 90, cuando los juegos de 16 bits deambulaban por la tierra, seguramente jugaste a este juego. Por supuesto, puedes emularlo en tu ODROID, pero ¿qué pasa si no has podido encontrar esta rom para tu emulador? ¡Bueno, ahora tendrás mas suerte!

Poder jugar con múltiples personajes era lo mejor de los 90

En la línea de lo que eran la peleas, éstas triunfaban en los juegos de la época, contabas con el modelo ideal de ciudad corrupta, en la cual los policías hacían justicia con sus propias manos para derrotar a la mafia de la ciudad y restaurar el orden en la población.

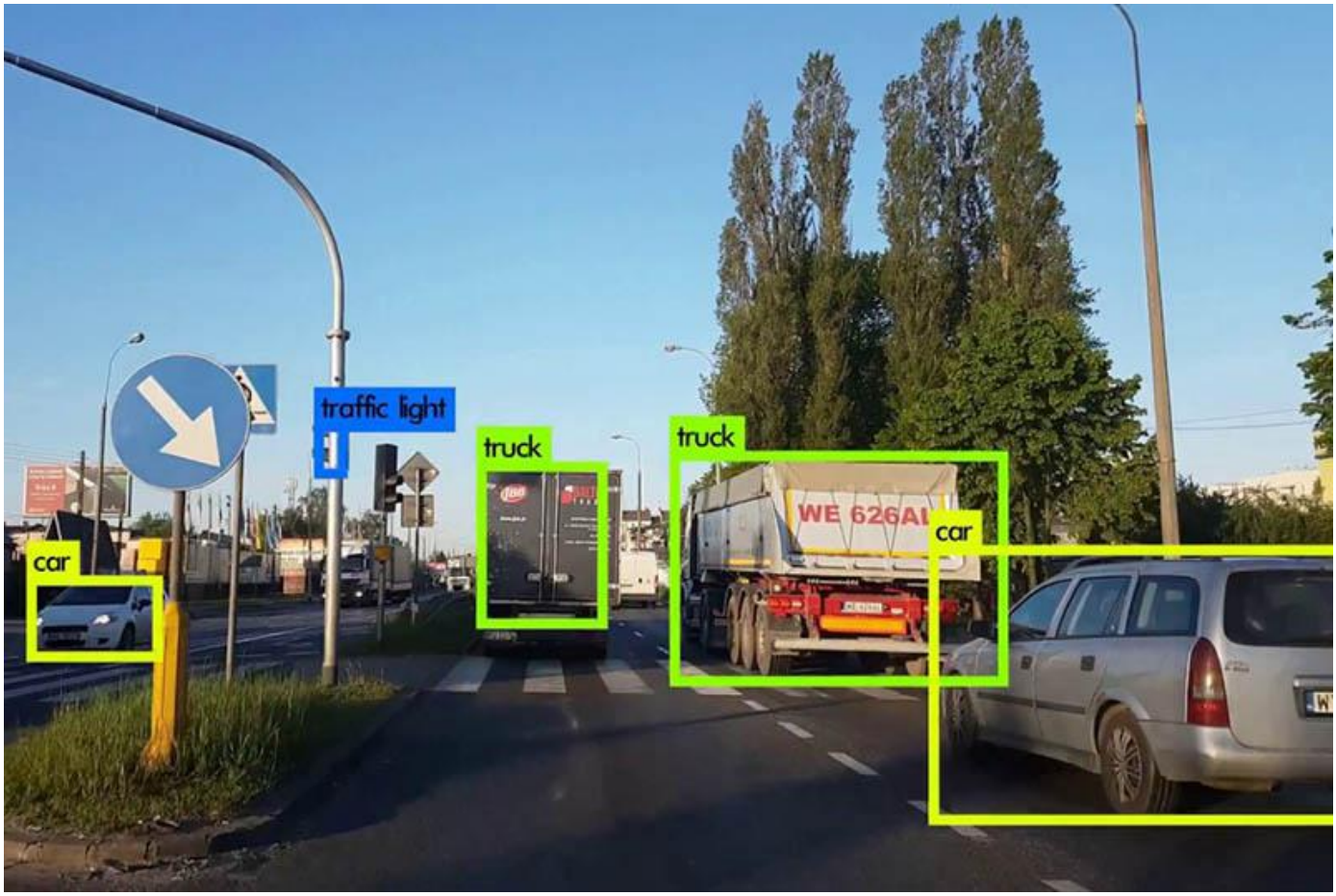
La banda sonora y esta pantalla de bienvenida están grabadas en la mente de todos jugadores de los 90.

Considerado una obra maestra de la diversión y con la inolvidable banda sonora de Yuzo Koshiro, este juego es en sí mismo el anteproyecto que todo desarrollador de juegos retro independiente intenta recrear cuando programa juegos.

Nada era más gratificante que golpear al gran jefe con el bazooka de repuesto.

Ejecutando YOLO en ODROID: YOLODROID

December 1, 2017 By Tom Jacobs Mecaniqueo



YOLO (**https://pjreddie.com/darknet/yolo/**) es un modelo de red neuronal capaz de reconocer objetos cotidianos muy rápido a partir de imágenes. También está TinyYOLO (**http://machinethink.net/blog/object-detection-with-yolo/**), que funciona bien en dispositivos móviles. Esta guía te ayudará a instalar y ejecutar TinyYOLO en tu ODROID-XU4. Para continuar, inicia sesión en tu ODROID y ejecuta los comandos que se muestran en las siguientes secciones.

Instalar TensorFlow

En primer lugar, nos aseguraremos de que todo esté actualizado:

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

Conseguir un poco de memoria de intercambio

Bazel no se compilará sin utilizar memoria de intercambio en el ODROID-XU4. Conecta una unidad USB de 8GB vacía, la cual borraremos y ejecuta el siguiente comando:

```
$ sudo blkid
```

Compruebe el nombre del dispositivo, generalmente /dev/sda1, con este nombre ejecuta:

```
$ sudo mkswap /dev/sda1
$ sudo swapon /dev/sda1
$ sudo swapon
```

Instalar los requisitos

Necesitaremos el verdadero Java Oracle, en lugar de OpenJDK. Lo intenté con OpenJDK, compilé Bazel con él, pero falló en las descargas hash SHA-1, así que fue inútil. De modo que tenemos que instalar los siguientes paquetes:

```
$ sudo apt-get install pkg-config zip g++
zlib1g-dev unzip
$ sudo apt-get install gcc-4.8 g++-4.8
$ sudo update-alternatives --install
/usr/bin/gcc gcc /usr/bin/gcc-4.8 100
$ sudo update-alternatives --install
/usr/bin/g++ g++ /usr/bin/g++-4.8 100
$ sudo apt-get install python-pip python-numpy
swig python-dev
$ sudo pip install wheel
$ sudo add-apt-repository ppa:webupd8team/java
$ sudo apt-get update
$ sudo apt-get install oracle-java8-installer
$ sudo apt-get install oracle-java8-set-
default
$ java -version
```

Instalar el sistema de compilación Bazel

Google compila las cosas usando Bazel. TensorFlow es de Google. Por lo tanto, necesitamos compilar Bazel en primer lugar. Esto llevara alrededor de una media hora, vete a buscar algo de comer mientras se ejecuta:

```
$ wget
https://github.com/bazelbuild/bazel/releases/d
ownload/0.5.4/bazel-0.5.4-dist.zip
$ unzip -d bazel bazel-0.5.4-dist.zip
$ cd bazel
$ sudo ./compile.sh
```

Ahora, Java se acaba aquí, así que tenemos que hacer las siguientes modificaciones:

```
$ sudo vi scripts/bootstrap/compile.sh
```

Busca la línea que empieza con “run” y añade algunos delimitadores de memoria, cámbiala con lo siguiente:

```
run “${JAVAC}” -J-Xms256m -J-Xmx384m -
classpath “${classpath}” -sourcepath
“${sourcepath}”
```

Luego, compila de nuevo:

```
$ sudo ./compile.sh
$ sudo cp output/bazel /usr/local/bin/bazel
```

Descargar y configurar TensorFlow

Ahora podemos descargar y configurar TensorFlow:

```
$ git clone --recurse-submodules
https://github.com/tensorflow/tensorflow.git
$ cd tensorflow
```

No pude instalar la última versión de TensorFlow, ya que tenía problemas de compilación BoringSSL C99. Para solucionar esto, revise la versión 1.4.0 y configure:

```
$ git checkout tags/v1.4.0
$ ./configure
```

Rechaza la mayoría de las cosas, incluido OpenCL, tal y como se muestra en la Figura 1.

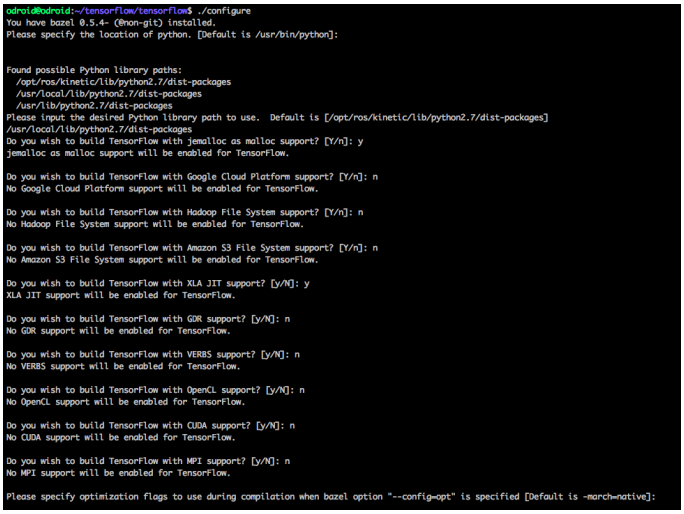


Figura 1 – Configurando TensorFlow

Compilar TensorFlow

Después, tenemos que compilar Tensorflow. Si pensabas que Bazel tardó mucho tiempo en compilarse, entonces es que no has desarrollado software antes. Sujétate bien el sombrero, porque el paseo va a ser largo:

```
$ bazel build -c opt --copt="-mfpu=neon-vfpv4"
--copt="-funsafe-math-optimizations" --copt="-
ftree-vectorize" --copt="-fomit-frame-pointer"
--local_resources 8192,8.0,1.0 --
verbose_failures
tensorflow/tools/pip_package:build_pip_package
 Building…
 1,900 / 4,909 files… error.
```

¡Oop!, NEON no funciona. De acuerdo, vamos a desactivarlo. Aunque solucionaremos esto más adelante:

```
$ bazel build -c opt --copt="-funsafe-math-optimizations" --copt="-ftree-vectorize" --copt="-fomit-frame-pointer" --local_resources 8192,8.0,1.0 --verbose_failures tensorflow/tools/pip_package:build_pip_package
 3,700 / 4,622 files… error.
 In file included from tensorflow/compiler/xla/service/llvm_ir/llvm_util.cc:30:0:
 ./tensorflow/core/lib/core/casts.h: In instantiation of 'Dest tensorflow::bit_cast(const Source&) [with Dest = long long int; Source = void (*)(const char*, long long int)]':
 tensorflow/compiler/xla/service/llvm_ir/llvm_util.cc:400:67: required from here
 ./tensorflow/core/lib/core/casts.h:91:3: error: static assertion failed: Sizes do not match
```

En este caso, es XLA el que está causando problemas. Es nuevo y no es necesario, así que vamos a dejarlo por ahora, reconfiguramos y recompilamos sin él:

2,345 / 3,683 files... 3,112 / 3,683 files... 3,682 / 3,683 files...

Target //tensorflow/tools/pip_package:build_pip_package up-to-date: bazel-bin/tensorflow/tools/pip_package/build_pip_package

Luego, instálalo:

```
$ bazel-bin/tensorflow/tools/pip_package/build_pip_package /tmp/tensorflow_pkg
$ sudo pip2 install /tmp/tensorflow_pkg/tensorflow-1.4.0-cp27-cp27mu-linux_armv7l.whl --upgrade --ignore-installed
```

Al principio, cuando ejecuté el siguiente comando, éste se ejecutó usando Python 3 y fallo la instalación, así que después de buscar en Google y conocer un poco las reglas de los nombres de archivos pip, descubrí el fallo y simplemente usé pip2:

```
$ sudo pip install /tmp/tensorflow_pkg/tensorflow-1.4.0-cp27-cp27mu-linux_armv7l.whl --upgrade --ignore-installed
```

Luego, lo más divertido fue cuando ejecuté por primera vez "import tensorflow", recibí este mensaje:

```
>>> import tensorflow
 Traceback (most recent call last):
 File "", line 1, in
 File "tensorflow/__init__.py", line 24, in
 from tensorflow.python import *
 File "tensorflow/python/__init__.py", line 49, in
 from tensorflow.python import pywrap_tensorflow
 File "tensorflow/python/pywrap_tensorflow.py", line 25, in
 from tensorflow.python.platform import self_check
 ImportError: No module named platform
```

Busqué en Google el problema, y parecía tratarse de un tema de configuraciones regionales, tal y como aparece en https://github.com/tensorflow/tensorflow/issues/36. So, Así que apliqué en primer lugar una configuración regional, luego además parecía que necesitaba ser la capital de EE. UU. (**http://bit.ly/2ifyle4**), y recompilé, aun así aparecia el mismo problema:

```
$ export LC_ALL=en_US.UTF-8
$ export LANG=en_us.UTF-8
```

Al día siguiente, con el poder de Google, descubrí que, en realidad, el problema era que solo lo estaba ejecutando en el directorio de compilación. Tiene un directorio llamado tensorflow, y Python lo estaba buscando para encontrar cosas. Entonces, simplemente cambiando los directorios se solucionó el problema.

```
$ python2
 >>> import tensorflow
 >>> print(tensorflow.__version__)
 1.4.0
```

Parece que todo está funcionando bien, así que ahora es el turno de YOLO.

Ejecutar YOLO

Estoy seguro que hay algunas implementaciones de YOLO por ahí, así que vamos a elegir una de

https://github.com/experiencor/basic-yolo-keras:

```
$ git clone
https://github.com/experiencor/basic-yolo-
keras.git
$ cd basic-yolo-keras
```

Descárgate los archivos de https://1drv.ms/f/s!ApLdDEW3ut5fec2OzK4S4RpT-SU, o desde https://1drv.ms/f/s!ApLdDEW3ut5feoZAEUwmSMYdPlY

```
$ wget /tiny_yolo_features.h5
$ wget /tiny_yolo_raccoon.h5
```

A continuación, edita el archivo de configuración y cambia el modelo a “Tiny Yolo”:

```
$ vi config.json
```

Descarga una imagen de un mapache:

```
$ wget
https://upload.wikimedia.org/wikipedia/commons
/b/be/Racoon_in_Vancouver.jpg
```

Luego, ejecuta el script:

```
$ python2 predict.py -c config.json -i
Racoon_in_Vancouver.jpg -w
tiny_yolo_raccoon.h5
```

Falta el paquete imgaug, de modo que lo agregaremos y sus dependencias:

```
$ sudo pip2 install imgaug
$ sudo pip2 install keras
$ sudo pip2 install h5py
```

Tanto h5py como scipy tardan un poco en instalarse. ¿Puedes encontrar el mapache en la Figura 2?

Figura 2 – Una imagen de prueba de un mapache

```
$ python2 predict.py -c config.json -i
Racoon_in_Vancouver.jpg -w
tiny_yolo_raccoon.h5
```

Efectivamente, ¡ahora es un mapache detectado!

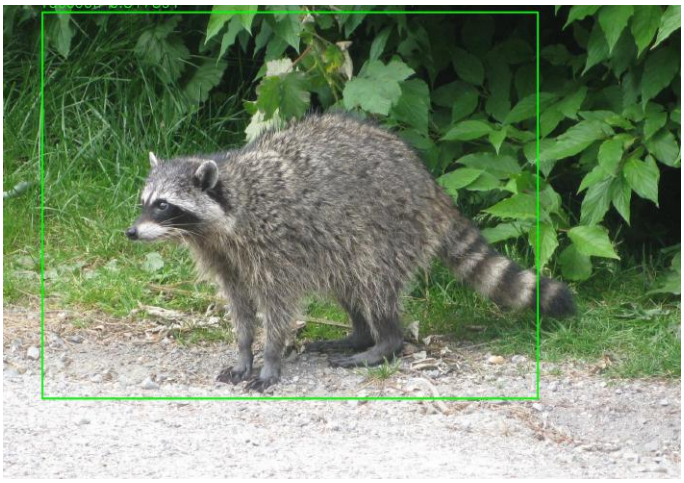


Figura 3 – YOLO ha detectado la imagen del mapache

Para comentarios, preguntas y sugerencias, visita el artículo original en **https://medium.com/@TomPJacobs/running-yolo-on-odroid-yolodroid-5a89481ec141**.

Juegos Linux: Need for Speed II Second Edition

December 1, 2017 By Tobias Schaaf Juegos, Linux

En 1997, los juegos de carreras eran muy populares. Una serie en particular que es bien conocida hoy en día y que todavía sigue siendo bastante nueva, las personas que empiezan a conocerla les gustaba cada vez más. La segunda edición de Need for Speed II (NFS2SE) se lanzó con soporte 3DFX, con más pistas y coches, junto con el modo espejo y el modo pista hacia atrás, lo cual supuso una gran mejora con respecto al Need for Speed II original. En algunos países, como Alemania, incluso se entregaba gratis con la compra de una tarjeta aceleradora 3DFX.

Figura 1 – Need For Speed II Second Edition

NFS2SE era un juego realmente bueno para los estándares de 1997, ya que los gráficos en 3D empezaban a llegar a los hogares y los juegos de carreras en 3D todavía se encontraban en sus etapas iniciales. Tener un juego de carreras decente en 3D con soporte real 3DFx Voodoo era algo muy raro por aquel entonces.

Junto con los gráficos en 3D, este juego tenía mucho que ofrecer. Una de sus peculiaridades era el video de movimiento completo (FMV) que se podía ver en la intro, así como videos de presentación para cada uno de los coches. En estos videos podías ver a los coches compitiendo por diferentes pistas y pueblos, o simplemente a cielo abierto. Ofrecía muchos videos para los muchos coches que se presentaban en el juego. A diferencia de los títulos modernos de Need for Speed, NFS2SE no tenía ninguna historia, de modo que no tenía ningún video de historia para seguir. Aun así, contaba con bastante contenido en video en comparación con otros títulos de la época.

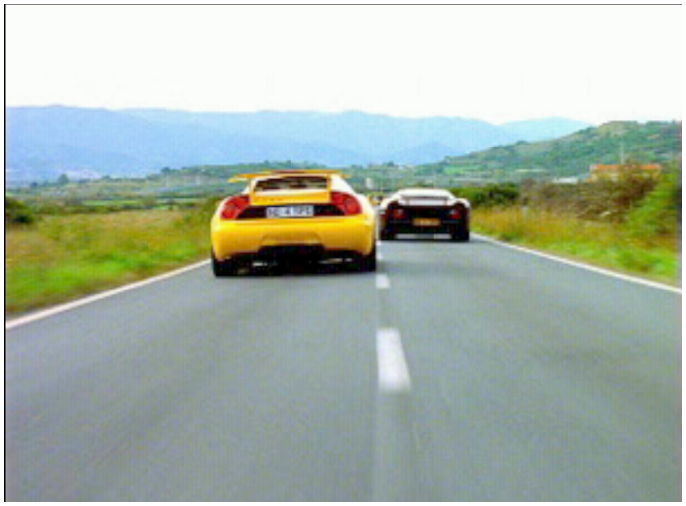


Figura 2 – Escenas del video de movimiento completo en NFS2SE

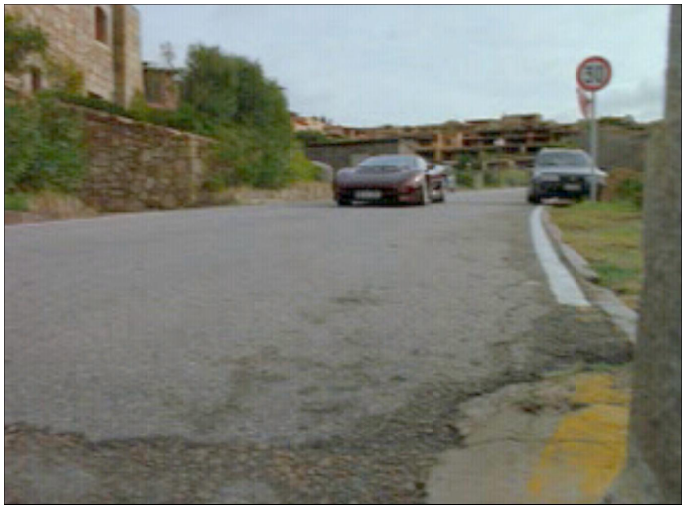


Figura 3 – Escena del video de movimiento completo en NFS2SE

De hecho, el juego contaba con bastante contenido adicional. El CD estaba lleno de fotos, videos e información sobre los coches, y ofrecía multitud de pistas y coches para jugar.

Figura 4 – Selecciona el escaparate para ver información, mirar imágenes y ver videos de cada automóvil

Figura 5 – Un video del coche seleccionado desde el escaparate

Need for Speed II Second Edition en ODROID

Recientemente he encontrado un proyecto de código abierto que tiene como objetivo recrear el motor NFS2SE para sistemas modernos usando el potencial 3D de los sistemas actuales, incluidos OpenGL ES 2.0 y SDL2, lo cual nos permitiría jugar en los ODROID. La recreación está orientada a la versión Glide (3DFx) del juego, dándole un aspecto similar al original, y al mismo tiempo permitiéndonos ejecutarlo a 1080P u otras resoluciones gracias a la funcionalidad de escalado del SDL2.

Hasta ahora, el juego se ve bien y parece ser totalmente jugable. Incluso el modo multijugador en red funciona bien. Lo probé en ODROID-XU3 y ODROID-U3, y funciona bien a toda velocidad en ambos dispositivos. Puesto que el juego se ejecuta en

SDL2 con OpenGL ES 2.0, permite aceleración 3D completa y funciona bastante bien, aunque me encontré con algunos problemas de velocidad si ejecutaba el juego en modo de un solo hilo.

Figura 6 – Empezando una nueva carrera en NFS2SE

El juego ofrece diferentes tipos de efectos ambientales: gotas de lluvia, niebla e incluso bugs para oscurecer tu vista. Otras áreas ofrecen muy buenos efectos de iluminación ambiental.

Figura 7 – Entrando en una selva tropical con lluvia y niebla

Figura 8 – Luz roja ambiental muy brillante en una cueva llena de lava

Aparte de eso, NFS2SE para ODROID ofrece compatibilidad con joystick e incluso utiliza el potencial de la retroalimentación (soporte para vibración), de modo que, si golpeas un objeto o una superficie diferente al pavimento, el gamepad vibrará para emular esa sensación. También probé el modo multijugador en dos ODROID y funcionó bastante bien. Todavía no he probado la pantalla dividida, pero apuesto a que también funcionará, lo que significa que podrás jugar con un amigo o con hasta 8 jugadores en la red con bastante facilidad.

Figura 9 – Este aterrizaje seguro que hará que el mando vibre en tus manos y active la “retroalimentación”

Hasta ahora, lo único que he encontrado que no funcionaba bien es el menú del juego. En éste puedes cambiar el volumen para aumentar y disminuir los sonidos y la música, pero nada más parece funcionar, ni siquiera los botones de continuar o reiniciar. Por

algún motivo, no es posible seleccionar estos elementos del menú. Solo puedo salir del menú con la tecla ESC. Sin embargo, esto no debería impedirte disfrutar del juego tal cual.

Figura 10 – El menú del juego parece estar parcialmente inservible ya que solo puede entrar y salir usando la tecla ESC

Cómo instalar el juego

Como de costumbre, el juego está disponible en mi repositorio para Debian Jessie y Debian Stretch. Como el juego es de 32 bits, las placas ARM64 como ODROID-C2 no admiten el juego. Puedes instalarlo desde mi repositorio con:

```
$ apt-get install nfs2se-odroid
```

La primera vez que ejecutes el juego, necesitarás instalar los archivos del juego del CD original de Need For Speed II SE. Se te pedirá que apuntes a un CD/carpeta que incluya la carpeta requerida (es decir, gamedata, fedata). Si estás utilizando mi imagen de GameStation Turbo, puede conectar una unidad de CD a través de USB y seleccionar el CD, o puede usar CDEmu (CD virtual) para montar la mayoría de los formatos de imágenes. Si tiene un archivo .iso, puede seleccionarlo y la configuración intentará extraer los archivos desde éste. Después de copiar los archivos, podrás continuar y disfrutar del juego.

Figura 11 – ¡Felicitaciones! Ahora puedes jugar al Need For Speed II SE en tu ODROID

Reflexiones finales

Need For Speed II Second Edition podría no ser el mejor juego de carreras, ni contar con la tecnología y gráficos de primer nivel según el estándar de hoy en día, pero sigue siendo un juego muy divertido, y la posibilidad de poder competir a través de una red es algo que no se ve a menudo en el ODROID. Este juego es digno de formar parte de la biblioteca ODROID. Espero que lo disfrutes tanto como yo.

Analizando el almacenamiento definido por software con GlusterFS en el ODROID-HC1: Parte 2 – Rendimiento del cliente

December 1, 2017 By Andy Yuen ODROID-HC1

En mi **anterior artículo**, describí cómo configurar un Volumen Distribuido Replicado GlusterFS así como un simple Volumen Replicado. También describí cómo usar GlusterFS Native Client para acceder a los volúmenes. En esta ocasión, te mostraré cómo configurar clientes NFS y Samba para acceder al volumen GlusterFS y compararemos el rendimiento de los diferentes clientes.

Cliente NFS

Se configura automáticamente un servidor NFS cuando instalamos GlusterFS y creamos un volumen replicado distribuido. Sin embargo, si tu instalación es como la mía, cuando ejecutes el siguiente comando, descubrirás que los servidores NFS están desconectados, tal y como se muestra en la Figura 1.

```
$ gluster volume status
```

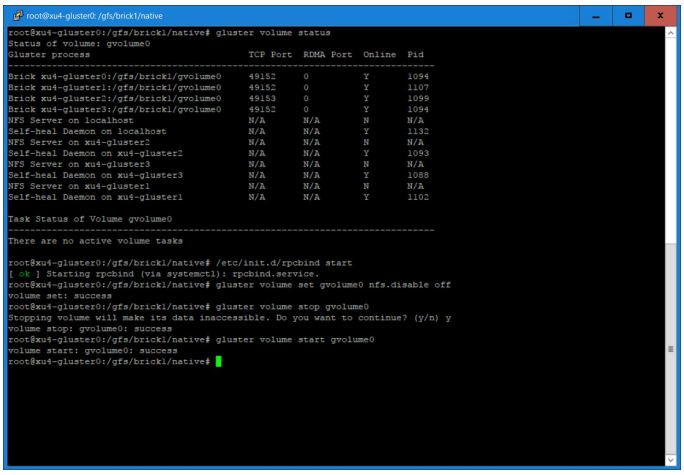


Figura 1 -nfs-desconectado

Es probable que se deba a que rpcbind no se esté ejecutando, tal y como se evidencia en el registro /var/log/glusterfs/nfs.log, el cual se muestra en la Figura 2.

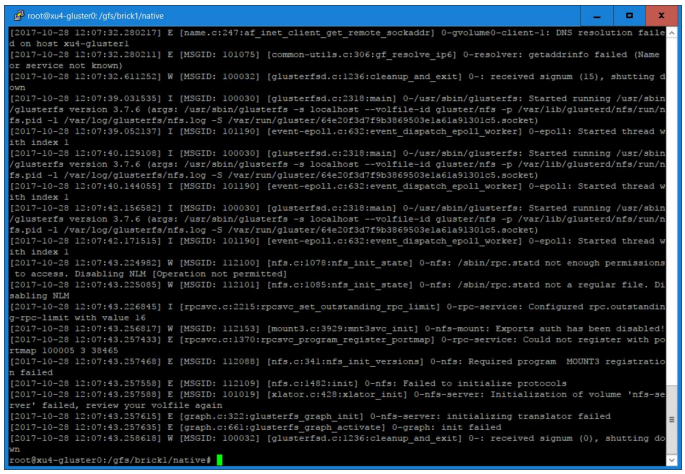


Figura 2 – rpc-error

Al ejecutar los siguientes comandos en todos los servidores GlusterFS se iniciarán los servidores NFS, donde gvolume0 es el volumen replicado distribuido de GlusterFS que creé en la Parte 1:

```
$ sudo /etc/init.d/rpcbind start
$ sudo gluster volume set gvolume0 nfs.disable off
$ sudo gluster volume stop gvolume0
$ sudo gluster volume start gvolume0
$ sudo gluster volume status gvolume0
```

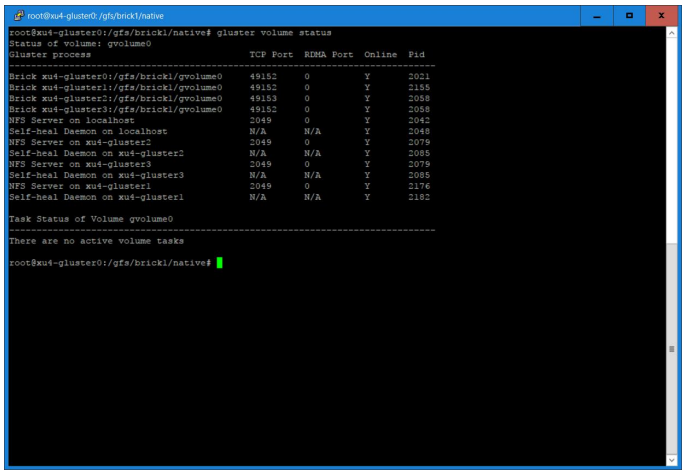


Figura 3 – nfs online en todos los volumenes

Luego configuramos un cliente NFS en una de las máquinas de mi ODROID-MC1, en concreto, el xu4-master, donde xu4-gluster0 es uno de los servidores GlusterFS:

```
$ sudo apt-get update
$ sudo apt-get install nfs-common
$ sudo mkdir /mnt/nfs
$ sudo mount -t nfs -o vers=3,mountproto=tcp xu4-gluster0:/gvolume0 /mnt/nfs
```

Ahora puedes acceder al volumen de GlusterFS en xu4-master. Puedes enviar el siguiente comando sobre la lista de 50 archivos que creamos en la Parte 1:

```
$ ls /mnt/nfs/testdir
```

Cliente SAMBA

La forma más sencilla de acceder a un volumen GlusterFS es exportar el punto de montaje Gluster como un recurso samba y montarlo usando el protocolo CIFS. Por supuesto, primero debes instalar los paquetes Samba y CIFS:

```
$ sudo apt-get update
$ sudo apt-get install samba smbfs cifs
```

Después, debes configurar una contraseña para samba,

```
$ sudo smbpasswd -a odroid
```

Edita el archivo /etc/samba/smb.conf con la siguiente configuración

```
[global]
 security = user
 #guest account = nobody
 [gvolume0]
 guest ok = yes
 path = /mnt/gfs
 read only = no
 valid users = odroid
 admin users = root
```

A continuación, reinicia Samba y monta el recurso compartido:

```
$ sudo /etc/init.d/samba restart
$ sudo mount -t cifs -ouser=odroid,password=odroid //192.168.1.80/gvolume0 /mnt/samba
```

Para montar en el recurso compartido NFS en lugar del recurso Gluster, simplemente desmonte Samba, modifica el archivo /etc/samba/smb.conf, reinicia Samba y monta el sistema de archivos de la siguiente forma:

```
[global]
 security = user
 #guest account = nobody
```

```
[gvolume0]
guest ok = yes
path = /mnt/nfs
read only = no
valid users = odroid
```

Reinicia samba y monta el recurso compartido:

```
$ sudo /etc/init.d/samba restart
$ sudo mount -t cifs -
ouser=odroid,password=odroid
//192.168.1.80/gvolume0 /mnt/samba
```

Rendimiento del cliente

Para comparar el rendimiento de los distintos clientes, necesitamos un punto de partida y éste es el rendimiento del sistema de archivos nativo, es decir, una partición montada en local en un servidor. Esto implica que tenemos 5 clientes distintos para comparar. Incluyen:

1. Sistema de archivos nativo: la prueba de rendimiento se ejecuta en un servidor donde se monta una partición de disco local
2. Cliente nativo GlusterFS: montado en una máquina que no sea un servidor GlusterFS que utilice GlusterFS Native Client
3. Cliente NFS Gluster: montado en una máquina que no es un servidor GlusterFS que utilice el cliente NFS GlusterFS.
4. Cliente Samba basado en GlusterFS Native Client – Recurso compartido CIFS del sistema de archivos del cliente Nativo GlusterFS.
5. Cliente Samba basado en GlusterFS NFS Client – Recurso compartido CIFS del sistema de archivos cliente NFS GlusterFS.

La herramienta para evaluar el rendimiento del sistema de archivos utilizada es iozone, que no está en el repositorio de software de Ubuntu, pero puede descargarse desde http://bit.ly/2BnxxfA. Genera y mide diversas operaciones de archivos. Nosotros utilizaremos los siguientes 4 indicadores de rendimiento para comparar el rendimiento del cliente: escritura de un solo hilo, escritura de 8 hilos, lectura de un solo hilo, lectura de 8 hilos.

Escritura de un solo hilo

El comando iozone usado en la prueba es:

```
$ iozone -w -c -e -i 0 -+n -C -r 64k -s 1g -t
1 -F path/f0.ioz
```

Las opciones utilizadas son: -c Incluye close() -e Incluye el vaciado de los cálculos del tiempo. (Las opciones -c -e se usan conjuntas para medir el tiempo que tardan los datos en alcanzar el almacenamiento persistente) -w No desvincula archivos temporales cuando termina de usarlos -i 0=escribir, 1=leer (solo utilizamos 0 y 1 en estas pruebas) -+n Ahorra tiempo omitiendo las pruebas de re-lectura y re-escritura -C Muestra cuánto tiempo participó cada hilo en la prueba -r Tamaño de transferencia de datos -s Tamaño de archivo de subproceso -t Número de hilos -F Lista de archivos

El comando se ejecuta para cada cliente. El resultado del comando se muestra en la Figura 4, y el resultado se resume en el gráfico de barras de la Figura 5.

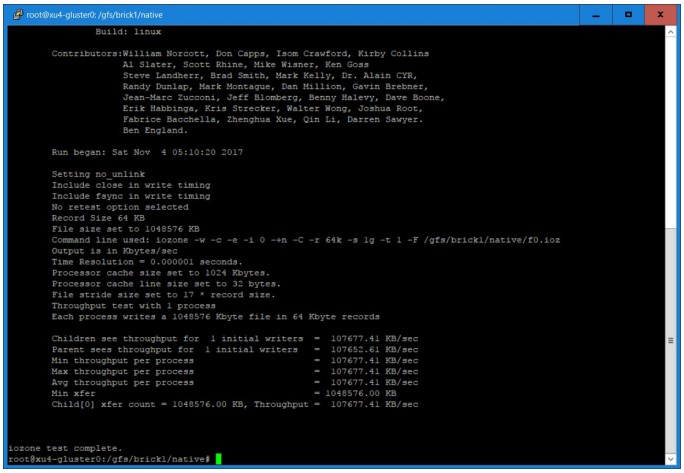


Figura 4 – native-w1 e

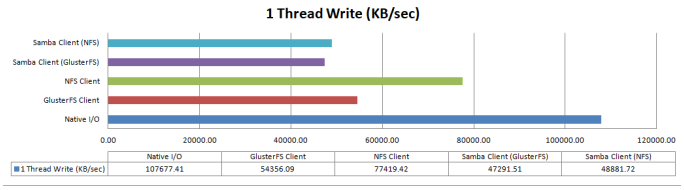


Figura 5 – Escritura 1 hilo

El nativo es el más rápido, seguido por los clientes nativos NFS y GlusterFS. Como era de esperar, los clientes de Samba son los más lentos en nuestra configuración porque depende de los clientes NFS y nativo GlusterFS y subyacentes.

Escritura de 8 hilos

El comando usado es:

```
$ iozone -w -c -e -i 0 -+n -C -r 64k -s 1g -t
8 -F path/f{0,1,2,3,4,5,6,7,8}.ioz
```

El resultado se muestra en la Figura 6, y los gráficos de resultados se muestran en la Figura 7.

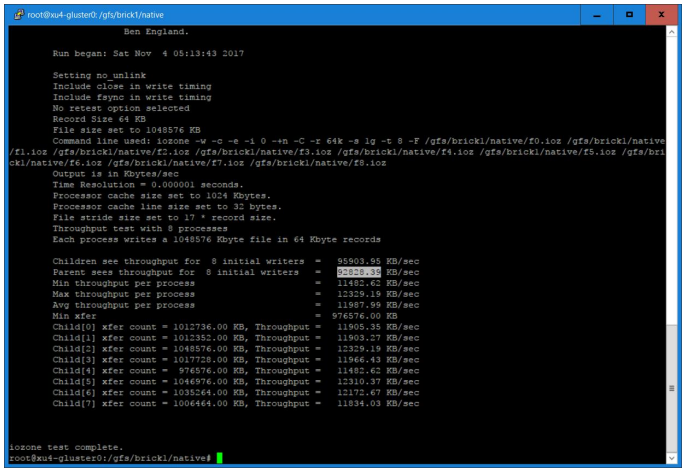


Figura 6 – native-w8

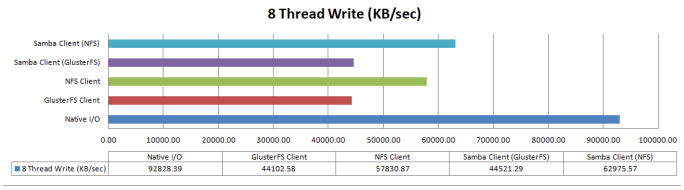


Figura 7 – Escritura de 8 hilos

En la prueba de rendimiento de escritura multiproceso, el resultado nativo es el más rápido, seguido por el cliente NFS y Samba que usa NFS. Ten en cuenta que el cliente Samba que utiliza NFS es más rápido que el cliente NFS en sí, para lo cual no tengo una explicación clara.

Lectura de un solo hilo

El caché se borra antes de la prueba de rendimiento de lectura:

```
$ sync
$ echo 1 > /proc/sys/vm/drop_caches
$ iozone -w -c -e -i 1 -+n -C -r 64k -s 1g -t
1 -F path/f0.ioz
```

El gráfico resultante con los resultados se muestra en la Figura 8.

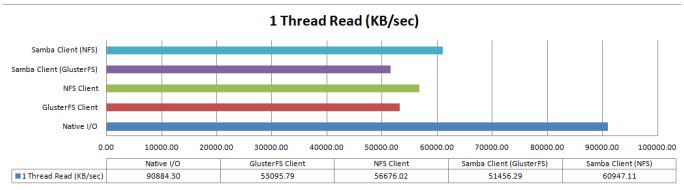


Figura 8 – Lectura de 1 hilo

El resultado vuelve a mostrar que el Nativo es el más rápido, seguido de los clientes basados en NFS y luego los clientes basados en clientes nativos de GlusterFS. De nuevo, no tengo ninguna explicación sobre por qué el cliente Samba que usa NFS es más rápido que el cliente NFS.

Lectura de 8 hilos

Finalmente, ejecuta la prueba de rendimiento de lectura multiproceso usando los siguientes comandos:

```
$ sync
$ echo 1 > /proc/sys/vm/drop_caches
$ iozone -w -c -e -i 1 -+n -C -r 64k -s 1g -t
8 -F path/f{0,1,2,3,4,5,6,7,8}.ioz
```

El resultado se muestra en la Figura 9.

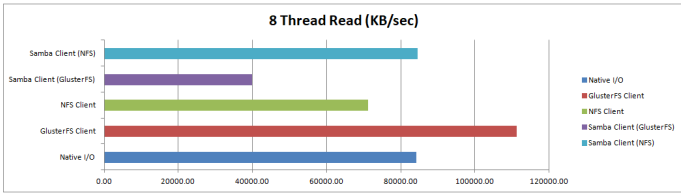


Figura 9 – Lectura de 8 hilos

El rendimiento es bastante diferente en este caso, ya que el cliente nativo GlusterFS es el más rápido, probablemente debido a la distribución. Los archivos se almacenan en diferentes servidores, lo cual añade paralelismo en la recuperación, a diferencia de las escrituras, lo que añade a su vez una sobrecarga al escribir datos en varios servidores.

Clientes de alta disponibilidad con tolerancia frente a fallos automática

De todos los clientes probados, solo el GlusterFS Native Client ofrece tolerancia frente a fallos automática y alta disponibilidad. Esto significa que, si falla el servidor GlusterFS especificado en el comando mount, cambiará automáticamente para usar otro servidor Gluster en nuestro Volumen Replicado o Replicado Distribuido.

Los clientes NFS y Samba utilizados no tienen estas funciones. Si desea esta función para NFS, debes desactivar el servidor NFS GlusterFS e instalar el servidor NFS-Ganesha (**http://bit.ly/2BuH9Ek**).

Del mismo modo, para Samba/CIFS, debes instalar el complemento Samba VFS desde http://bit.ly/2i7gMjI.

Además de proporcionar alta disponibilidad y tolerancia frente a fallos automática, también utiliza libgfapi para evitar la penalización de rendimiento entre el modo kernel y el del usuario, que tiene lugar en nuestro cliente Samba basado en GlusterFS Native Client utilizado en nuestra prueba.

Ten en cuenta que el plugin para gluster no está presente en el paquete Ubuntu samba-vfs-modules. Animo a que los analicéis si queréis contar con alta disponibilidad y tolerancia frente a fallos automática para los clientes de NFS y Samba.

Conclusión

Te he enseñado cómo configurar Volúmenes Replicados Distribuidos y Replicados GlusterFS usando dispositivos ODROID-HC1, y cómo acceder a ellos usando GlusterFS Native Client, NFS y Samba. También te he mostrado su rendimiento mediante gráficas que te permite compararlos fácilmente. Ahora dispones de suficiente información para elegir el cliente más apropiado. Personalmente, creo que es una tecnología empresarial muy buena que se presta para su fácil uso doméstico. En mi opinión el ODROID-HC1 es más económico y flexible que los sistemas NAS comerciales. Espero que compartas mi entusiasmo al utilizarlo en casa.

Conociendo un ODROIDian: Andrea Cole, editora adjunta de ODROID Magazine

December 1, 2017 By Conociendo un ODROIDian

Por favor, háblanos un poco sobre ti.
Actualmente soy administradora de ventas de Lab Manager, una publicación centrada en la industria para la comunidad científica. He formado parte de su empresa matriz, LabX Media Group, durante más de 10 años, y he estado trabajando concretamente en la división Lab Manager durante cuatro años. Vivo en Canadá, a un par de horas al norte de Toronto, Ontario, un pequeño pueblo situado a orillas de la Bahía Georgiana.

Tengo una licenciatura en Sociología de la Laurentian University. Originalmente empecé estudiando la carrera de Psicología, pero después de asistir a algunas clases y darme cuenta de que estaba invirtiendo todo mi tiempo y energía en esas clases sin obtener los resultados esperados, decidí cambiar de especialidad. Probablemente fue una decisión inteligente.

Actualmente vivo con mis dos hijas, que aún están en el instituto y mi pareja, que anteriormente había trabajado en la industria TI, pero que ahora se dedica más a trabajos de horticultura ya que, como él mismo dice, “trabajar en la industria estaba echando a perder su gran hobby por completo.” Él fue quien despertó mi interés por la electrónica.

Figura 1 – Andrea disfruta pasando tiempo con sus dos hijas

¿Cómo empezaste con los ordenadores?

Cuando era niña, mi familia disponía de un ordenador en casa de vez en cuando, pero era principalmente un ordenador de trabajo que se usaba para el procesamiento de textos, así que no me despertaba un gran interés. Cuando Internet se hizo más popular, tuve un mayor contacto con los ordenadores, pero principalmente los utilizaba como herramienta para acceder a Internet. Pasaba mucho tiempo en algunos foros on-line y como músico incipiente, fui un frecuente visitante de On-Line Guitar Archive (OLGA). Sin embargo, no fue hasta mi actual trabajo cuando empecé a adquirir conocimientos básicos de programación. Como mi trabajo conlleva marketing por correo electrónico, he reunido suficientes conocimientos en HTML y CSS como para hacer un diseño web básico. Durante mucho tiempo estuve trabajando con un robot de búsqueda, y a partir ahí fui adquiriendo conocimientos básicos de expresiones regulares. En general, basta decir que aún sigo siendo bastante nueva en esto.

¿Cómo usas tus ODROID?

He estado utilizando el ODROID-C2 mayoritariamente para reemplazar los antiguos C1 y Raspberry Pi, en nuestras instalaciones de entretenimiento que han ido creciendo en casi todas las habitaciones de la casa, gracias a los ODROIDS y a un compañero en el que no se puede confiar para que no añada más funcionalidades a casi todo lo que hay en casa. Por ejemplo, llegué a casa un día y descubrí que mi tocadiscos de armario de 1960 ahora tiene Wi-Fi, una interfaz web y la posibilidad de transmitir música desde Internet.

¿Cuál es tu ODROID favorito y por qué? No puedo decir que realmente haya pensado en un favorito, ya que solo he tenido experiencia con ODROID-C1 y ODROID-C2, y experiencia muy limitada con ODROID-XU4. Dado que mi próximo pedido a Hardkernel probablemente contenga al menos media docena de ODROID-C2, probablemente me quedo con este. Tenemos más posibles usos para el ODROID-C2 que para cualquier otra placa actualmente disponible.

¿A quién admiras en el mundo de la tecnología y por qué? ¿Es Tony Stark real? ¿No? ¡Maldición!. [Nota del editor: Elon Musk es lo más cercano que tenemos a un verdadero Tony Stark]

¿Qué beneficios ves al ayudar a otros a aprender sobre los ODROID?

La informática de placa reducida con recursos como los ODROID tiene el beneficio de ayudar a las personas a comprender las tecnologías que cada vez están más presentes en la vida cotidiana. Es importante que la gente conozca el potencial de su tecnología, para que puedan desmitificar estas pequeñas cajas mágicas en sus vidas y las potencien para crear sus propios dispositivos y que harán lo que quieran, cómo quieran y sin verse limitados por ideales corporativos arrastrados por los beneficios.

¿Qué pasatiempos e intereses tienes aparte de los ordenadores? He sentido un gran interés por la música desde mis primeros años de adolescencia. También siento una cantidad casi infinita de curiosidades por la cultura pop. Hace 10 años, comencé a pintar como hobby, y en los últimos años he dedicado un gran esfuerzo a perfeccionar esta destreza. Mis obras de arte se pueden encontrar en https://andrea-cole.pixels.com/.

Figura 2 – Obra de Andrea conocida como “A Tragically Hip Mountain Goat”

Figura 3 – Obra de Andrea conocida como “Victoria Harbour Town Dock”

¿Qué consejo les darías a otros que quieren aprender más sobre los ordenadores?

Echar un vistazo a cursos de programación en línea que puedas hacer en su tiempo libre. Algunos son más prácticos que otras, pero si conoces a personas que ya estén familiarizadas con el tema, es posible que puedan orientar sobre sitios con buena reputación para el aprendizaje en línea. No te sientas abrumado por la cantidad de información que aparentemente es inmensa. Después, es una cuestión de organización de tiempo y simplemente hacerlo.