LineageOS • RemotePI Board • HIFI Shield 2 • Desarrollo Android

ODROID Magazine

Año cuatro
Núm. #39
Mar 2017

Crea tu propia *Estación*

ARCADE ODROID

Un completo tutorial para el montaje

- Timbre IOT: Recibe una imagen de quien llama a tu puerta
- Home datacenter: Pon en practica tu proyectos con facilidad

Qué defendemos...

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

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

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

Simple, moderno y único.
De modo que tienes a tu alcance lo mejor para hacer todo lo que imagines

HARDKERNEL

Pollin ELECTRONIC

Realizamos envios de ODROID-C2 and ODROID-XU4 a los países de la UE! Ven y visita nuestra tienda online!

Dirección: Max-Pollin-Straße 1
85104 Pförring Alemania

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

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

EDITORIAL

A todo el mundo le encanta los juegos, especialmente a los ingenieros de Hardkernel. Nuestro proyecto estrella de este mes es un proyecto personalizado creado por Brian, John y Charles que resalta el potencial de la plataforma ODROID como emulador de consolas y sistema de juego versátil. Usando la librería WiringPi y la popular imagen de juegos GameStation Turbo de ODROID, diseñaron una Arcade Box que es impresionante y te permite ejecutar tus juegos favoritos a resoluciones de 1080p. ¡Puedes construir la tuya propia y tendrás conversación para rato en tu próxima partida de juegos!

Miltos ha creado un timbre remoto muy útil para ver quién llama a tu puerta, enviando un correo electrónico con una imagen, John nos muestra cómo construir un Home Data Center con un ODROID-XU4, y Bo continúa sus crónicas de científico loco con un sensor de ultrasonidos para su vehículo. Tobias presenta OpenFodder, un clon de Cannon Fodder que captura la belleza del juego original, Nanik analiza el uso de la red en Android, y Bruno nos trae otra entrega de su famosa serie de juegos Android, Causality.

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

HARDKERNEL

NUESTRO MARAVILLOSO PRESONAL ODROIDIAN:

Rob Roy, Editor Jefe

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

Bruno Doiche, Editor Artístico Senior

Jugó a alrededor de 20 juegos este mes, pero sólo encontró uno que valiera la pena!

Manuel Adamuz, Editor Españoi

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

Nicole Scott, Art Editor

Nicole es una experta en Producción Transmedia y Estrategia Digital especializa en la optimización online y estrategias de marketing, administración de medios sociales y producción multimedia impresa, web, vídeo y cine. Gestionando múltiples cuentas con agencias y productores de cine, desde Analytics y Adwords a la edición de vídeo y maquetación DVD, Nicole ayuda a sus clientes con todos los aspectos de la visibilidad online. Posee un ODROID-U2, varios ODROID-U3 y Xu4's, y espera poder utilizar las últimas tecnologías tanto para a nivel personal como empresarial. El sitio web de Nicole lo puedes encontrar en http://www.nicolecscott.com.

James LeFevour, Editor Artístico

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

Andrew Ruggeri, Editor Adjunto

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

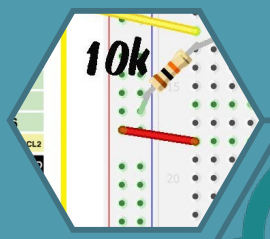
Venkat Bommakanti, Editor Adjunto

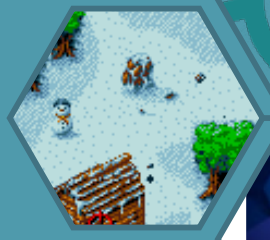
Soy un apasionado de los ordenadores desde la bahía de San Francisco en California. Procuro incorporar muchos de mis intereses en proyectos con ordenadores de placa reducida, tales como pequeños modificaciones de hardware, carpintería, reutilización de materiales, desarrollo de software y creación de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

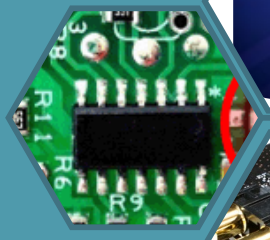
Josh Sherman, Editor Adjunto

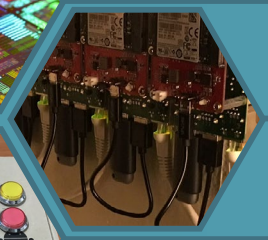
Soy de la zona de Nueva York, y ofrezco mi tiempo como escritor y editor para ODROID Magazine. Suelo experimentar con los ordenadores de todas las formas y tamaños: haciendo trizas las tablets, convirtiendo Raspberry Pi en PlayStations y experimentado con los ODROIDs y otros SoCs. Me encanta trabajar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

INDICE









TIMBRE IOT

RECIBE UN AVISO POR CORREO ELECTRONICO DE LA PERSONA QUE LLAMA A TU PUERTA

por Miltiadis Melissas (@miltos)

En este proyecto, vamos a convertir el ODROID-C2 (http://bit.Ly / 1oTJBya) en un inteligente timbre IoT que tomará una foto de quienquiera que lo haga sonar, y la enviará por correo electrónico al titular de una cuenta de Gmail. Además, el dispositivo archivará las fotos por fecha y hora, dando la posibilidad de comprobar cualquier actividad sospechosa o simplemente para mantener un registro de todas las personas que tocan el timbre. Es fácil ver que este "inteligente" timbre que utiliza un ODROID-C2 como piedra angular es una poderosa herramienta de seguridad y vigilancia, el cual será muy útil para cualquier vivienda.

Requisitos de hardware

- ODROID-C2 (http://bit.ly/1oTJBya)
- Webcam ODROID (http://bit.ly/2iBHKPD)
- Adaptador Wi-fi (http://bit.ly/1M4LdiC)
- 1x mini placa de pruebas
- 1x Resistencia 1 KΩ
- 1x Resistencia 10 KΩ
- 1x Pulsador
- ~8x cables dumpon (el kit Tinkering C http://bit.ly/1YNPN6kes una buena opción para este y futuros proyectos)

Requisitos de Software

- Ubuntu 16.04 v2.0 de Hardkernel (http://bit.ly/2cBibbk)
- Python 2.7 or 3.3 (preinstalado en Ubuntu)
- Librería WiringPi para controlar los pines GPIO del ODROID-C2. Puedes aprender a instalarla con la guía que hay disponible en http://bit.ly/2ba6h8o

Montando el dispositivo IoT

Para cablear las conexiones, usamos los cables macho a hembra y macho a macho Dupont. El extremo hembra del cable Dupont macho a hembra se conecta al cabezal macho del ODROID-C2, y el otro extremo a los orificios de la placa de pruebas. Ve consultando el esquema de distribución de los pines

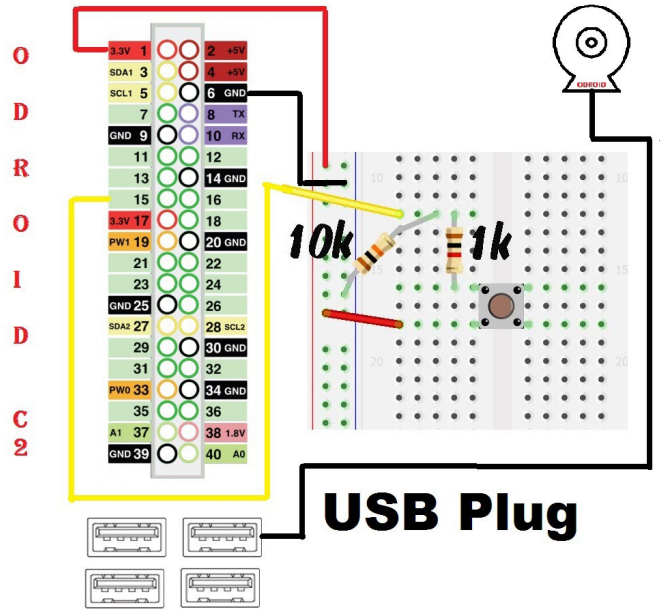


Figura 1 - Esquema del Timbre

de Hardkernel a medida que vayas creando las conexiones. Este esquema lo puedes encontrar en http://bit.ly/2aXAlmt. El pin físico 1 es VCC y proporciona 3.3V a nuestro circuito, conectamos el pin en la primera línea vertical de nuestra placa de pruebas, cerca del borde. Puesto que vamos a utilizar el pin 6 como puesta a tierra común, éste lo conectaremos a la segunda línea vertical de nuestra placa de pruebas. El resto del circuito es muy simple, puedes seguir el esquema de la Figura 1.

La Webcam ODROID la tienes que conectar al ODROID-C2 a través de uno de los puertos USB disponibles en la placa. El timbre se controla con el pin 15, tal y como puedes ver en la figura. Ahora que tenemos el hardware listo, ¡Vamos a sumergirnos en el código y hacer que el timbre suene "inteligentemente"!

Software

La parte central del código ha sido extraída de un proyecto Github disponible en http://bit.ly/2jEXRbR. No obstante, este código de ejemplo ha sido modificado en gran medida

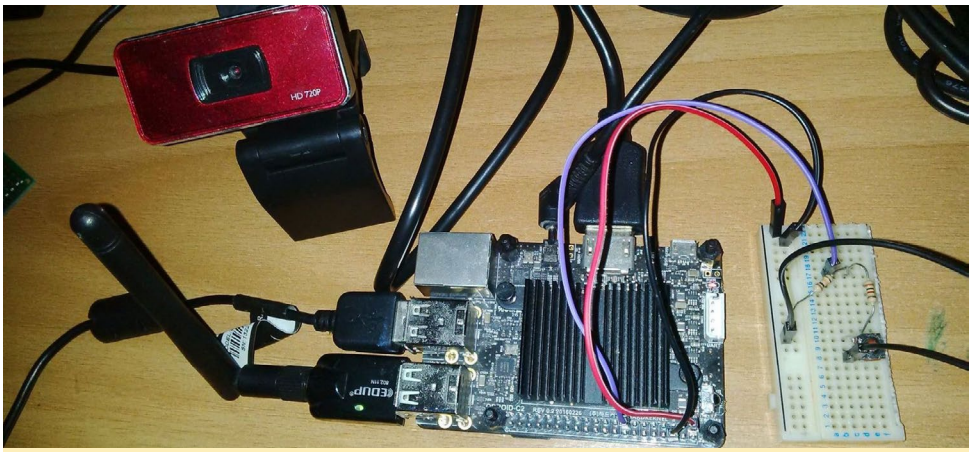


Figura 2 - El sistema completo C2

para conseguir que funcione correctamente en el ODROID-C2. Lo más importante de todo es que el re-mapeo desde RPi.GPIO a la librería WiringPi2 ya está terminado, puesto que WiringPi2 es compatible con el ODROID-C2. Consulta la excelente guía GPIO proporcionada por Hardkernel en http://bit.ly/2jEUjWX. Todo el código esta descrito en los siguientes apartados, delimitado entre '<...>', con su correspondiente descripción debajo.

El código básico de odroidbell.py

```
<import wiringpi2 as odroid>
```

Empezaremos importando la librería wiringpi2. Las instrucciones para instalar esta librería y poder controlar los pines GPIO del ODROID-C2 la puedes encontrar en http://bit.ly/2ba6h8o.

```
<import time>
```

Importamos el módulo time.

```
<import os>
```

Importamos el módulo os

```
<import glob>
```

Importamos el módulo glob.

```
<import sys>
```

Importamos el módulo sys.

```
<odroid.wiringPiSetup()>
```

Hemos configurado el módulo wiringPi2 conforme a la tabla que nos proporciona Hardkernel en http://bit.ly/2aXAlmt.

```
<Button = 3>
```

Este es en realidad el pin físico 15 según la tabla de http://bit.ly/2aXAlmt.

```
<odroid.pinMode(Button,0)>
```

Fijamos el pulsador como entrada.

```
<odroid.
pullUpDnControl(Button,1)>
```

Activamos la resistencia de parada y arrastre. En este caso parado tal y como lo indica el argumento 1.

```
#loop
```

Estamos entrando en un bucle ... ya que los dispositivos IoT siempre están en modo de espera.

```
<print("Program Running")>
```

La única finalidad de esto es la monitorización

oses.

```
while True:#loops forever till
keyboard interrupt (ctr + C)>
  <if odroid.digitalRead(Button)
== False: #when button is un-
pressed:>
   <sys.stderr.write(".")>
```

Si no se presiona el timbre, imprimimos puntos en la pantalla usando la librería sys.

```
<time.sleep(1)>
```

Se comprueba si se ha tocado el pulsador/timbre cada segundo.

```
<else:>
  <print("Button Pressed")>
```

Se ha presionado el pulsador/timbre

```
 #    ------|    photo & Bell
|------ #
    #Get FileName
    <now = time.strftime("Date%m-
%d-%yTime%H-%M-%S")>
```

Declaramos la variable now con la fecha y hora que usaremos en la foto.

```
  #Make command to run odroidC2.
sh
     <command = "bash odroidC2.sh
" +  str(now)>
```

Invocamos el script shell odroidC2.sh (ver a continuación)

```
  # -- odroidC2.sh is an Shell
script that
    # -- is responsible for tak-
ing the photo and
    # -- making the Doorbell
Noise

    # --- We insert the "Now" ar-
gument so the python
    # --- script knows what the
filename of the
    # --- picture will be so it
can pass it on into the
```

```
    # --- email script (so it
knows what file to email
        #run command
    <os.system(command)>
```

A continuación invocamos el comando system para ejecutar el script de correo electrónico

```
  #diagnostics
    <print("Filename:", now)>
```

Imprimimos el "filename" de la foto con la fecha y hora actuales.

```
 # ----| Email      |---- #
    <print("Email")#email>
    <emailcommand = 'sudo python
IoTOdroid.py "This person is at
your door"' + ' "photos/' + now +
'.jpg"'>
```

Enviamos la foto a la cuenta de Gmail del titular con la fecha y hora y el asunto "This person is at your door".

```
   <os.system(emailcommand) #run-
ning the Email script with:>
    #-- the subject as "Someone
is ringing the doorbell" and the
filename
    #-- We made before at the
-Photo & Bell- section
```

Y finalmente lo enviamos.

```
 # -- End Diagnostic Info
    print("Done Process")
```

Todo está hecho, así que el script finaliza su trabajo

```
   #-space out for next "Press of
Button"
    <print("")>
    <print("")>
```

Hacemos hueco para el siguiente ciclo (próxima llamada al timbre).

Script shell OdroidC2.sh

El script OdroidC2.sh es responsable de tomar la foto y hacer sonar el timbre. La función del argumento 'Now' es pasar el nombre del archivo de la foto al script de Gmail. En otras palabras, es el conector entre nuestro código básico, odroidbell.py y el script Odroid-IoTNotifier.py. El script OdroidC2.sh es muy simple:

```
<cd photos>
```

Cambiamos desde directorio actual al directorio <photos>.

```
<echo "Taking the Photo">
<now=$1>
```

"Now" es la fecha y hora de registro del nombre del archivo.

```
<echo>?

<fswebcam -d /dev/video0 $now.
jpg>
```

Este es el comando básico para tomar la foto. Utilizamos el comando fswebcam. Si la aplicación fswebcam no está instalada en tu sistema, puedes instalarla con el siguiente comando:

```
$ sudo apt-get install fswebcam
```

La sintaxis del comando es obvia: toma una foto y usa la fecha y hora de registro como nombre de archivo. Cada vez que se pulse el pulsador, OdroidC2.sh es activado por odroidbell.py. El parámetro -d determina el recurso a utilizar, en nuestro caso /dev/video0.

```
<echo "Pic Taken">
<echo"">
<echo "Ringing Bell">
<echo "">
<cd ..>
```

Cambia de nuevo al directorio padre.

```
<mpv ringtone.mp3>
```

Finalmente haremos sonar el timbre usando un programa llamado mpv, que ya está incluido en Ubuntu 16.04 v2.0 de Hardkernel (http://bit.ly/2cBibbk). En otras palabras, usamos el Mplayer para reproducir este archivo.

Configurar el código de Gmail

La mayoría de la gente tiene una cuenta de Gmail. Si no la tienes, es muy fácil crearte una y lo más importante de todo, es gratis. En realidad, para que este script de Gmail funcione correctamente, necesitamos dos cuentas de correo electrónico: el correo electrónico del remitente y la cuenta de correo electrónico del destinatario y siempre es así. Por supuesto, puede enviar un correo electrónico desde y hacia la misma cuenta, pero es más elegante crear una segunda cuenta de correo electrónico con el fin de estar al tanto de las fotos con el registro de la fecha y hora por separado. También recomiendo que la cuenta de correo electrónico del destinatario sea la que utilices en tu dispositivo móvil para que el dispositivo te avise cada vez que alguien llame a tu puerta. No olvides permitir que " las aplicaciones menos seguras" tengan acceso a tu cuenta de Gmail (http://bit.ly/124TgWN).

Vamos a examinar el script python llamado IoTOdroid.py.

```
<from email.mime.text import MIM-
EText>
<from email.mime.multipart import
MIMEMultipart>
```

Usamos esos dos módulos porque necesitamos enviar un correo electrónico limpio, con un remitente, un receptor y un asunto.

```
<from email.mime.application im-
port MIMEApplication>
```

También importamos el módulo re-

sponsable del archivo adjunto MIME. MIME representa las Extensiones Multiuso para el Correo en Internet. Es una forma de identificar los archivos en Internet según su naturaleza y formato.

```
<import smtplib>
<from smtplib import SMTP>
```

Esta es la básica y nativa librería en Python para enviar correos electrónicos, de modo que no hay necesidad de instalar librerías externas: smtplib. Desde esta librería, importamos la función SMTP.

```
<import sys>
```

Importamos el módulo de funciones y parámetros específicas del sistema, puesto que necesitaremos el script <argv> de este módulo (ver a continuación).

```
<recipients = ['<YourEmail>']>
```

Tu dirección de correo electrónico, ya que tú eres el destinatario de las fotos.

```
<emaillist = [elem.strip().
split(',') for elem in recipi-
ents]>
```

Hacemos una lista de correo electrónico en la que separaremos y quitaremos los caracteres pertinentes de cada elemento de la lista de destinatarios.

```
<msg = MIMEMultipart()>
```

Definimos el variable message (msg) como una Extensión Multiuso para el Correo en Internet llamando a la función MIMEMultipart.

```
<msg['Subject'] = str(sys.
argv[1])>
```

El asunto del nuestro mensaje.

```
<msg['From'] = '<From Email>'>
```

Tu dirección de correo electrónico.

```
<msg['Reply-to'] = 'xyz@gmail.
com'>
```

Correo electrónico del destinatario.

```
<msg.preamble = 'Multipart
massage.\n'>
```

El atributo preamble contiene el texto principal extra-blindado de los documentos MIME, por eso lo incluimos aquí.

```
<part = MIMEText("Hello! The
doorbell is ringing! A photo of
the person ringing the doorbell
has been attached.")>
```

El cuerpo del mensaje.

```
<msg.attach(part)>
<part =
MIMEApplication(open(str(sys.
argv[2]),"rb").read())>
<part.add_header('Content-
Disposition', 'attachment',
filename=str(sys.argv[2]))>
```

Adjuntamos la foto al mensaje

```
<server = smtplib.SMTP("smtp.
gmail.com:587")>
```

Especificamos el servidor smtp que queremos usar y el puerto que utiliza: el servidor de Gmail con el puerto 587. También puede usar el puerto 465, aunque no es mala idea comprobar con Google el puerto correcto, por si acaso ha cambiado.

```
<server.ehlo()>
```

Especificamos el servidor smtp que queremos usar y el puerto que utiliza: el servidor de Gmail con el puerto 587. También puede usar el puerto 465, aunque no es mala idea comprobar con Google el puerto correcto, por si acaso ha cambiado.

```
<server.starttls()>
```

Configuramos y entramos en modo TLS. TLS significa Transport Layer Security (Seguridad de la capa de transporte), de modo que cualquier comando SMTP que siga este modo será cifrado.

```
<server.login('<From
Email>','<From password>')>
```

Es hora de entrar en tu cuenta de Gmail, de modo que necesitamos las credenciales correctas.

```
<server.sendmail(msg['From'],
emaillist , msg.as_string())>
```

Usando el anterior comando, finalmente enviamos el correo electrónico. Procura evitar el spoofing e introduce aquí tu dirección de correo electrónico real. El parámetro, emaillist, es el que

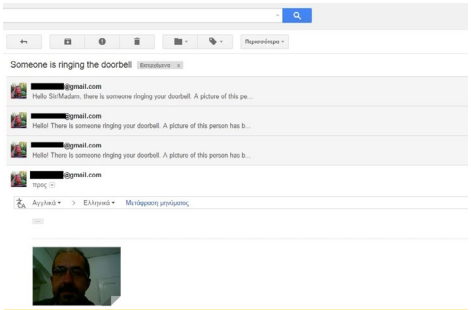


Figura 3 - Alerta de correo electrónico de que alguien está en la puerta

hemos definido antes y el último parámetro (msa.as_string ()) el mensaje en forma de cadena con el archivo adjunto, que es, en este caso, la foto. El resultado lo puedes apreciar en la Figura 3.

Testeando y ejecutando el código

Desde el terminal (CTRL-T), ejecutamos odroidbell.py con privilegios sudo:

```
$ sudo python odroidbell.py
```

En este momento el dispositivo IoT se pone en modo de espera y apa-

rece en pantalla el mensaje "Program is running". Al mismo tiempo, aparecen puntos "." en la pantalla, uno a uno, indicando al usuario que el dispositivo está funcionando con normalidad. Cuando alguien pulsa el botón (timbre), el script OdroidC2.sh se ejecuta con una doble finalidad: Primero, toma una in-

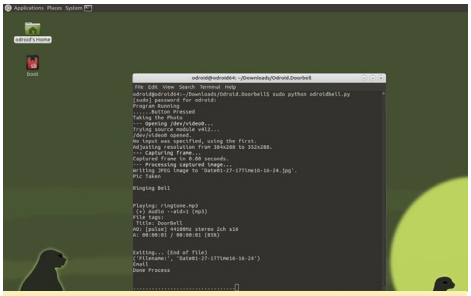


Figura 4 – El script Python del timbre en ejecución

stantánea de la persona que hace sonar el timbre. En segundo lugar, envía el correo electrónico a través de IoTOdroid.py a la cuenta de Gmail del propietario con la fecha y hora de registro de la foto que se adjunta. Después, vuelve al modo de espera y el timbre IoT suena.

Notas finales

Por supuesto, este proyecto se podría mejorar de muchas formas. Por ejemplo, se podría añadir un LED en el timbre IOT que indicara su correcto funcionamiento. Un dispositivo más complejo, similar a éste, podría usarse como timbre comercial con la ayuda de un módulo placa relé. Tampoco sería complicado incluir la posibilidad de que grabase una pequeña grabación de vídeo junto con las fotos, haciendo que el timbre "inteligente" sea aún más inteligente. Además, como dice el viejo refrán, "¡la perfección es el eterno enemigo de lo funcional!"

Código del timbre IoT

Aquí tienes una copia de todo el código. Al final de este artículo tienes un enlace a la página github que también contiene el código.

Odroidbell.py:

```
import wiringpi2 as odroid
import time
import os
import glob
import sys

odroid.wiringPiSetup()

Button = 3

odroid.pinMode(Button,0)
odroid.pullUpDnControl(Button,1)


#loop
print("Program Running")
while True:#loops forever till
keyboard interupt (ctr + C)
  if odroid.digitalRead(Button)
== False: #when button not
pressed:
    sys.stderr.write(".")
    time.sleep(1)
  else:
    print("Button Pressed")
    #    ------|    photo & Bell
|------ #
    #Get FileName
    now = time.strftime("Date%m-
%d-%yTime%H-%M-%S")
    #Make command to run
odroidC2.sh
    command = "bash odroidC2.sh "
+  str(now)

    # -- odroidC2.sh is an Shell
script that
    # -- is responsible for tak-
ing the photo and
    # -- making the Doorbell
Noise

    # --- We insert the "Now" ar-
gument so the python
    # --- script knows what the
file name of the
    # --- picture will be so it
can pass it on into the
    # --- email script (so it
knows what file to email
    #run command
    os.system(command)
    #diagnostics
    print("Filename:", now)


    # ----| Email     |---- #
    print("Email")#email
    emailcommand = 'sudo python
IoTOdroid.py "Someone is ringing
the doorbell"' + ' "photos/' +
now + '.jpg"'
    os.system(emailcommand) #run-
ning the Email script with:
    #-- the subject as "Someone
is ringing the doorebell" and the
filename
    #-- We made before at the
-Photo & Bell- section

    # -- End Diagnostic Info
    print("Done Process")
    #-space out for next "Press
of Button"
    print("")
    print("")
```

OdroidC2.sh:

```
#!/bin/sh
cd photos
echo "Taking the Photo"
now=$1 #Now is the filename time
stamp
#take pic
fswebcam -d /dev/video0 $now.jpg
echo "Pic Taken"
echo""
#ring Bell
echo "Ringing Bell"
echo ""
echo ""
cd ..
mpv ringtone.mp3
```

IoTOdroid.py:

```
from email.mime.text import MIM-
EText
from email.mime.application im-
port MIMEApplication
```

```
from email.mime.multipart import
MIMEMultipart
from smtplib import SMTP
import smtplib
import sys

recipients = ['abc@gmail.com']
emaillist = [elem.strip().
split(',') for elem in recipi-
ents]
msg = MIMEMultipart()
msg['Subject'] = str(sys.argv[1])
msg['From'] = 'xyz@gmail.com'
msg['Reply-to'] = 'abc@gmail.com'

msg.preamble = 'Multipart
massage.\n'

part = MIMEText("Hello! There is
someone ringing your doorbell.  A
picture of this person has been
atached.")
msg.attach(part)

part =
MIMEApplication(open(str(sys.
argv[2]),"rb").read())
part.add_header('Content-
Disposition', 'attachment',
filename=str(sys.argv[2]))
msg.attach(part)

server = smtplib.SMTP("smtp.
gmail.com:587")
server.ehlo()
server.starttls()
server.login('xyz@gmail.
com','yourpassword here')

server.sendmail(msg['From'],
emaillist , msg.as_string())
```

El código del proyecto está disponible en http://bit.ly/2jMAdMY usando el siguiente comando:

```
$ git clone \
https://github.com/miltiadisme-
lissas/\
IoT.OdroidC2.Doorbell.git
```

LINEAGEOS-14.1 ODROID-XU3/XU4

OLVIDATE DE CYANOGEN, EL FUTURO ESTA AQUI

por @voodik
editado por Bruno Doiche

A finales de 2016 los usuarios de Cyanogenmod se horrorizaron, cuando Cyanogen anunció la cancelación de sus servicios, dejando a todos los usuarios que dependían de Cyanogenmod sin soporte. Sin embargo, en poco tiempo llegaría una solución en forma de LineageOS. Si eres un usuario del ODROID-XU3 / XU4, es el momento perfecto para colaborar con la versión de LineageOS en los foros ODROID:

Características

- Android 7.1.1 Nougat LineageOS 14.1
- Kernel 3.10.9
- OpenGL ES 1.1/2.0/3.0 (aceleración por GPU)
- OpenCL 1.1 EP (aceleración por GPU)
- Función multiusuario habilitada (hasta 8 usuarios)
- Ethernet integrado y soporte para Ethernet Gigabit USB 3.0
- Soporte para RTL8188CUS, RTL8191SU y USB Wireless Ralinkt
- Soporte Bluetooth USB (BLE, A2dp Sink).
- Soporte para dongle GPS USB
- Anclaje a red USB .
- Punto de acceso Wi-Fi portátil.
- Soporte DAC USB nativo para Android
- Soporte para Webcam USB UVC
- Soporte HDMI-CEC
- Selinux

Problemas conocidos

En este momento sólo admite módulos Bluetooth de bajo consumo v4.0 (BLE). Consulta el Módulo Bluetooth 2

Cómo instalarlo

En primer lugar, necesitas preparar tu emmc/sd con imágenes especiales de auto-instalación.

La puedes encontrar aquí

```
http://oph.mdrjr.net/voodik/5422/
ODROID-XU3/Android/CM-14.1-ATV/
Alpha-0.1_11.02.17/
```

Graba la imagen en tu eMMC/sd a través de Win32DiskImager y arranca el dispositivo. Tendrá que esperar pacientemente durante el primer arranque, ya que el proceso de actualización puede llegar a tardar hasta 20 minutos.

Código fuente Kernel

```
$ git clone https://github.com/\
voodik/android_kernel_hardkernel\
_odroidxu3 -b cm-14.0_5422
```

JUEGOS LINUX

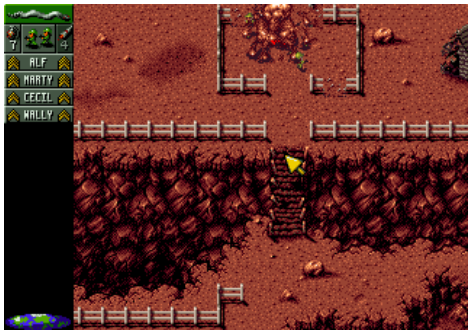
OPEN FODDER

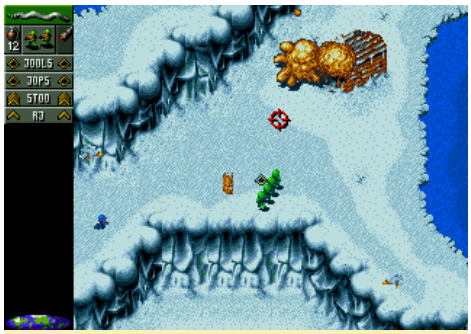
por Tobias Schaaf (@meveric)

Este mes, me gustaría hablar de un juego llamado Open Fodder, se trata de un remake del clásico juego de Amiga Cannon Fodder desarrollado por Sensible Software en 1993. Utiliza los datos del juego original Cannon Fodder para conseguir una experiencia muy similar al juego original en tu ODROID.

Figura I - El logotipo de Cannon Fodder, en el cual está basado Open Fodder

Muchos de los populares juegos de Linux de hoy en día, como Stratagus y Freeciv, son en realidad remakes de clásicos juegos de DOS de mediados y finales de los 90, y Open Fodder no es diferente. El remake de Cannon Fodder es un juego disparos, acción y estrategia donde controlas a un pequeño grupo de soldados que avanza a través de varias docenas de niveles para matar a los soldados enemigos, destruir tanques, echar abajo edificios y derrotar a tus enemigos. El juego fue lanzado por primera vez para Amiga en 1993, y fue exportado por sus desarrolladores a MS-DOS, Atari Jaguar, SNES, 3DO y Sega Genesis. El juego tenía diferentes escenarios para jugar, la selva, la nieve y el desierto.



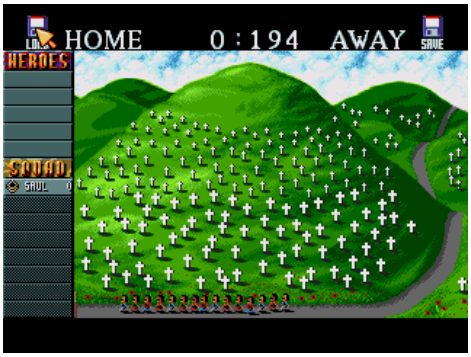


Figuras 2, 3 y 4 - Los niveles de selva, desierto y nieve en Open Fodder

Open Fodder ofrecía 23 misiones que se dividían en varias fases. En total, suponía superar 72 niveles para llevar a tu escuadrón a la victoria. Cuando se lanzó, el juego fue muy elogiado por varias revistas de juego Amiga, con puntuaciones que llevaban al 95 por ciento y fue considerado como uno de los mejores juegos lanzados en 1993. En general, siguió siendo uno de los mejores juegos jamás creado para la plataforma Amiga durante sus 11 años de vida.

Desde un punto de vista más político, el juego lleva implícito un intenso mensaje sobre la guerra en sí misma puesto que juegas y llevar a tus soldados a una muerte inevitable. Cada misión se inicia

Figuras 5 y 6 - A medida que avanzas en el juego, tu larga línea de reclutas se convierte gradualmente en cruces alineadas como en un cementerio.



con un nuevo grupo de reclutas que se alinean esperando a unirse al combate. Los soldados desechables (de ahí el nombre de cannon fodder, Carne de Cañon) poco a poco se convierten en cruces alineadas dando forma a un cementerio militar a medida que vas completando cada misión, formándose más y más filas conforme vas enterrando a tus predecesores. Sin embargo, los desarrolladores le dieron un cierto toque de humor, ya que los primeros soldados de tu escuadrón (y por tanto, los casi seguro que morirán) llevan de hecho, los nombres de las seis personas que componían el equipo de desarrollo del juego.

Si nos fijamos bien, el juego incorporaba algunos detalles bastante interesantes. Por un lado, tiene su propia canción, " War, never been so much fun", que suena durante la intro del juego. ¡Uno de los desarrolladores del juego, Jon Hare, compuso la música junto con el compositor Richard Joseph! Si tenías el Amiga CD32, también recordarás una pista adicional que incluía un video en el cual los desarrolladores se disparaban entre sí con armas de juguete (http://bit.ly/2l67bFy). ¡Sin duda se divirtieron bastante desarrollando este juego!

Cannon Fodder también tuvo algunas segundas partes, además de algunas misiones de bonificación. Cannon Fodder 2 salió un año más tarde, pero fue más bien un "disco de datos" con expansiones que una secuela, ya que en su mayor parte presentaba más misiones, en lugar de nuevas características. También estaba el X-Mas de Amiga, que ofrecía algunas misiones diferentes basadas en el juego Cannon Fodder.

Ejecutando Open Fodder

Open Fodder es un remake del motor del juego Cannon Fodder, que lleva el juego a los sistemas operativos modernos. Al igual que otras migraciones del motor del juego (como OpenTTD) puede utilizar los datos del juego original con esta versión del motor para ejecutar el juego en tus dispositivos modernos. Esta migración también cuenta con una versión exportada para los dispositivos ARM, lo que significa que podemos ejecutarlo igualmente en nuestros ODROIDs.

Todavía está en desarrollo, pero ya funciona bastante bien. Como de costumbre se puede instalar desde mi repositorio y puesto que sólo requiere SDL2 y SDL2 Mixer como dependencias principales debería funcionar en Debian y Ubuntu por igual, las puedes encontrar en mi lista principal de paquetes para armhf y jessie/main para arm64.

Se puede instalar con este comando en un dispositivo ODROID que ejecute Debian, suponiendo que ya tengas configurados mis repositorios o estés usando una de mis imágenes de Debian (http://bit.ly/13v98ly):

```
$ apt-get install \
  openfodder-odroid
```

He alterado ligeramente el juego para que siempre se ejecute en modo pantalla completa, aunque si lo deseas, puedes cambiar al modo de ventana simplemente pulsando F11 cuando tenga el juego activado.

Convirtiendo los datos del juego

Por sí sola, mi versión de Open Fodder viene con el especial Amiga X-Mas, además de varios niveles demo disponibles de forma gratuita. Si quieres jugar el juego real, tendrás que importar los datos desde un CD Cannon Fodder

Si está usando un ODROID para jugar al Open Fodder, puedes localizar las carpetas de los datos del juego en la carpeta $HOME/.openfodder/Data. Por ejemplo, /home/odroid/.openfodder/Data/Dos_CD es una de esas carpetas de datos del juego. Existen varias carpetas de datos del juego que puedes usar dependiendo de la versión que tengas del juego Cannon Fodder.

Dos_CD

Aunque el juego es compatible con muchas y diferentes fuentes del juego, actualmente sólo Dos_CD parece ser una versión de Open Fodder "totalmente compatible" con muy pocos errores y problemas. El resto de fuentes del juego pueden tener problemas.

Aquí tienes un consejo rápido: Aunque se llama Dos_CD, cogerá cualquier versión de DOS de Cannon Fodder que puedas encontrar. Si tienes

Figura 7 - Sensible Software - Los desarrolladores de Cannon Fodder

CAUSALITY

UN JUEGO DE ROMPE-CABEZAS CON PARADOJAS DE VIAJES EN EL TIEMPO

por Bruno Doiche

Configura tú mismo los entornos extraños e insólitos, y ayuda a un grupo de astronautas varados a encontrar una ruta hacia un lugar seguro. Cada nivel al que te enfrentas representa un nuevo reto donde necesitas llevar a tu astronauta a una salida que coincida con su color. Los márgenes de tiempo son cortos, aunque es muy divertido. Aunque formas parte de tan sólo una fracción de tiempo de la vida de tu personaje, ¡ten cuidado: pasarás un montón de tiempo enganchado a este juego!

`https://play.google.com/store/apps/details?id=com.lojugames.android.Causality`

El entorno 3D isométrico es magnífico, a menudo no superaba el nivel porque los gráficos me cautivaban.

la versión de disco de Cannon Fodder, simplemente copia el CF_ENG.DAT en la carpeta Dos_CD. La versión original de CD DOS tenía un archivo llamado cf_cd.dat. Cambiar el nombre a CF_ENG.DAT y copiarlo a la carpeta Dos_CD. Si tienes la versión GoG del juego, sólo tiene que copiar CF_ENG. DAT a la carpeta Dos_CD. ¡Esta es una forma rápida y legal de conseguir el juego, si estás interesado!

Asegúrate de que el nombre del archivo este en mayúscula, ya que el juego tiene en cuenta esta cuestión. Intenta que coincida todo para asegurarte de que las cosas funcionen sin problemas.

Amiga_CD

Amiga CD32 también contaba con una versión de este juego. Esta es la versión que incluye el video al que hemos hecho referencia anteriormente. En el CD de Amiga CD32, hay una carpeta llamada Fodder. El contenido de esta carpeta debe copiarse a la carpeta Amiga_CD. Puedes extraer la segunda pista del CD y almacenarla como Track2.flv en la misma carpeta con la finalidad de tener el archivo de video.

Me topé con algunos problemas cuando hacía esto, y descubrí nuevamente que el juego buscaba muchos archivos en mayúsculas, así que utilicé el siguiente comando para copiar todos los archivos y tenerlos también en mayúsculas en la carpeta Amiga_CD:

```
$ for files in `ls`; do cp $files
`echo $files | tr '[:lower:]'
'[:upper:]'`; done
```

Así funcionaba bien, pero tenía algunos fallos. Por ejemplo, la animación del helicóptero al inicio de cada misión no aparecía, y el cursor cuando se guarda un archivo se distorsionaba. Aparte de esto, no he visto problemas importantes en la versión Amiga CD32.

Amiga

La Información de desarrollador de Open Fodder dice: "Usar el instalador WHDLoad en un Amiga (o WinUAE) para extraer los archivos del juego y copiarlos en la carpeta Data/Amiga". No lo he probado, pero si tienes la "Cannon Fodder Collection" de Amiga CD32 no oficial, puedes copiar el contenido de FodderNew en esta carpeta, lo que tendrá como resultado niveles completamente diferentes. Puede considerarse como una versión más compleja del juego, aunque a la vez más divertida. Por lo tanto, bien poner los archivos originales de Amiga aquí, o bien los archivos FodderNew de la "Cannon Fodder Collection".

Dos2_CD

Dos2_CD es en realidad la carpeta de la versión DOS de Cannon Fodder 2, pero también funciona con la versión GoG. Desafortunadamente, tiene algunos problemas, el más grave es que el juego no tiene ningún tipo de sonido o música. Aparte de esto, todo parece funcionar correctamente, pero desconozco cómo reaccionará en niveles posteriores. La carpeta Data también incluye las carpetas WAV, Plus, AmigaFormat_XMAS y Custom. En WAV se almacenan los efectos de sonido por defecto para todos los juegos, (Cannon Fodder) Plus es una demo del número 31 la revista Amiga Power, y AmigaFormat_XMAS es el Amiga Format Christmas Special. También hay una carpeta Custom que te permite jugar otros juegos y mapas, pero necesitan la versión (CD) de DOS o de lo contrario no aparecerán.

Notas finales

Open Fodder es un buen proyecto que te permite ejecutar este increíble juego en sistemas modernos. Realmente me gusta y estoy deseando ver cómo progresa. Espero que las versiones de Amiga y Cannon Fodder 2 sean pronto totalmente compatibles. Tal vez incluso podamos ver mejoras gráficas con el tiempo. Me gustaría ver gráficos más modernos, ya que el juego es bastante antiguo y los gráficos no son los más idoneos para resoluciones 1080p.

REMOTEPI BOARD PARA EL ODROID-C2

por @inifity85

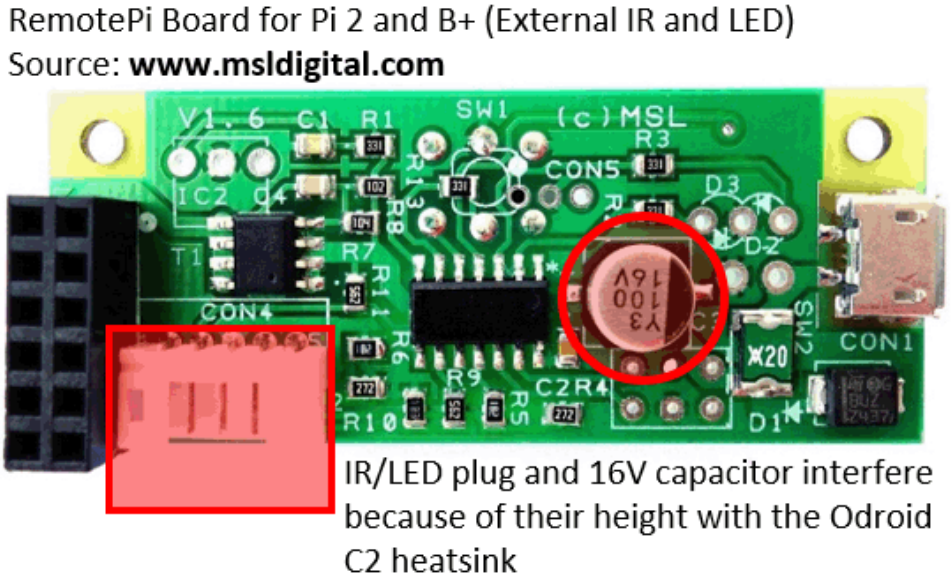


Figura 1 - RemotePi Board

La RemotePi board (http://bit.ly/2l8JcWU), que convierte cualquier mando a distancia en un interruptor de encendido para tu ordenador de placa reducida, puede ser compatible con el ODROIDC2 si sigues esta guía. Si utilizas tu dispositivo como centro multimedia, esta placa te proporcionara un receptor de infrarrojos y un circuito de alimentación, así como un botón de encendido. Puedes encender y apagar tu ODROID completamente a través de una señal infrarrojos (IR), y simplemente presionar el botón de encendido para apagarlo y reiniciarlo con seguridad. La placa RemotePi es responsable del modo espera, que absorbe una mínima energía mientras el ODROID está completamente apagado. Por último, la placa alimenta tu dispositivo a través de GPIO, lo cual requiere un cable microUSB para conectarlo a la RemotePi en lugar de utilizar un adaptador de corriente normal.

Configuración de hardware

Puesto que la RemotePi estaba pensada originalmente para la Raspberry Pi 2, necesitarás utilizar cables adicionales en lugar de colocarla justamente en cima del C2 para evitar interferir con el disipador de calor, así que debes recablear algunos pines. Es necesario utilizar cables con el diámetro adecuado para los pines de 5V y puesta a tierra, ya que la corriente podría estar entre los 2A y 2,6A, dependiendo de cuántos dispositivos USB conectes a tu dispositivo.

Los pines GPIO 8 y 10 de la RemotePi necesitan estar conectados a otros distintos en el ODROID-C2, porque parece haber un conflicto. La interfaz UART ocupa estos pines, y el estado por defecto del ODROID (1=high en Pin 8) no es lo que espera la RemotePi (GPIO debe entrar en el valor por defecto 0=low para cortar la energía tras el apagado). Pero puesto que necesitas utilizar cables de todos modos para conectar la RemotePi, esto no supondrá ningún problema, ya que simplemente puedes reconectar los cables a otros pines GPIO.

Como evitamos el circuito de alimentación del ODROID encendiéndolo a través de GPIO, también pasamos por alto la protección ante subidas de tensión 2.5/2.6A del dispositivo, pero no te preocupes. La RemotePi también tiene protección ante subidas de tensión que debería coincidir con el mismo valor, que en el caso de la RemotePi board para la Raspberry Pi 3. Sin embargo, la RemotePi para la Raspberry Pi 2 es de sólo 2A. Esto está bien, porque el valor es menor a los 2.6A del ODROID, de modo el polyfuse de la RemotePi se activaría antes si conectas demasiados dispositivos no alimentados de forma externa, que no es una buena idea en un SBC al fin al cabo. No obstante, debido a esta diferencia de amperaje, una RemotePi para la Raspberry Pi 3 sería la mejor opción para este proyecto.

Como resultado del nuevo cableado, el firmware de RemotePi ya no se puede configurar de forma normal, aunque nunca he usado esta función. Si fuera necesario configurar o actualizar el firmware, necesitarás volver a conectar los pines 8 y 10 de la RemotePi a los pines 8 y 10 del ODROID durante la actualización del firmware y luego deshacer los cambios una vez que la actualización haya finalizado. Otra posibilidad es simplemente conectar una Raspberry Pi para actualizar el firmware. Esto suena a un montón de problemas, pero en realidad simplemente es extender la conexión del cabezal y cambiar una línea en el script principal, y dos líneas en otro script opcional si haces uso de ello.

Re-cableando la conexión GPIO

Dado que la RemotePi está diseñada

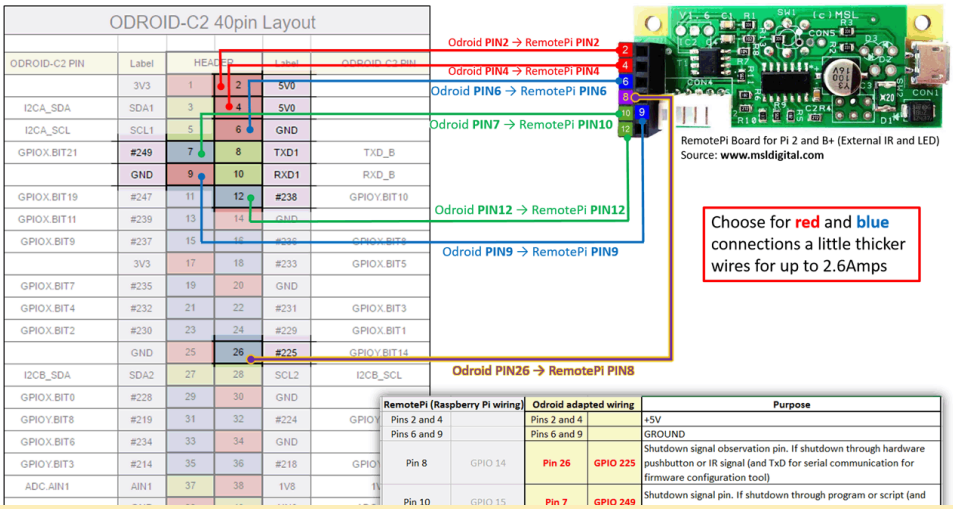
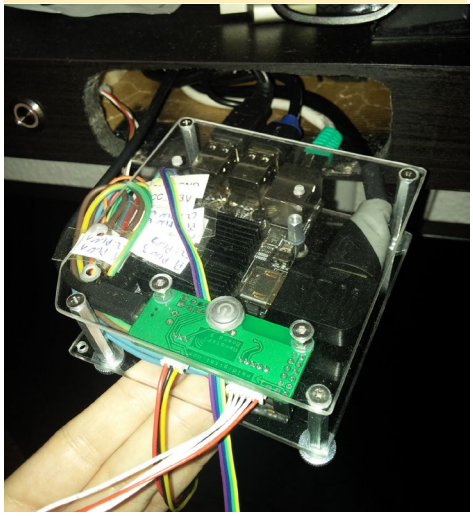


Figura 2 - Re-cableado GPIO

originalmente para una Raspberry Pi, desgraciadamente no es sólo enchufar y usar. Necesitarás volver a cablear dos pines usando los cables de todos los pines, tal y como se muestra en la Figura 2. Re-cablea el pin 8 de la RemotePi al pin 26 del ODROID-C2 y el pin 10 de la RemotePi al pin 7 del ODROID-C2.

Si no quieres utilizar el receptor IR de la RemotePi para el mando a distancia de LibreELEC, puedes dejar desconectado el pin 12. La placa continuará encendiéndose y apagándose a través del comando de control remoto IR, pero el control de la GUI de LibreELEC será asumido por el receptor de infrarrojos integrado en el ODROID. Sin embargo, si has comprado una RemotePi con un receptor IR y un LED externo, es posible que hayas decidido colocarla dentro de una carcasa personalizada, de modo que el receptor integrado podría estar oculto y no ser útil. En ese caso, puedes desactivar el IR del ODROID en favor del receptor externo GPIO-IR en la RemotePi y conectar el pin 12. Tienes disponible más información sobre el cambio al receptor GPIOIR en LibreELEC en http://bit.ly/2lpDl27.

Figura 3 - La RemotePi ha sido recableada y montada en una carcasa personalizada



Scripts de apagado para LibreELEC

Los dos scripts tratados a continuación pueden descargarse desde la página de soporte de MSL Digital Solutions en http://bit.ly/2kMxyVG. En esta página también encontraras una guía para utilizar estos scripts en otros sistemas operativos como Volumio y RuneAudio.

El script irswitch.sh se usa para el apagado seguro a través de sistema Infrarrojos. Tras pulsar el correspondiente botón, el sistema se apagará de forma segura, luego RemotePi esperará a que el GPIO225 alcance el estado 0 (low), que aparece tras el cierre exitoso del sistema. Finalmente, cortará la energía.

El script shutdown.sh se usa para el apagado seguro a través de la interfaz del programa o script. Después de navegar hasta el botón de apagado en la GUI, el sistema se apagará de forma segura, luego RemotePi esperará a que el GPIO225 alcance el estado 0 (low), que aparecerá tras apagarse el sistema de forma adecuada y segura. Finalmente, cortará la energía completamente.

Irswitch.sh (solo cambia el script original de MSL Digital de GPIOpin1 = 14 a GPIOpin1 = 225):

```
#!/bin/bash
# prevent restarting XBMC at
shutdown.  This is only used for
OpenElec before V5
LOCKDIR="/var/lock/"
LOCKFILE="xbmc.disabled"
# this is the GPIO pin receiving
the shut-down signal
# Raspberry Pi pin8: GPIOpin1=14;
Odroid-C2 pin26: GPIOpin1=225
GPIOpin1=225
# functions
add_omit_pids() {
omit_pids="$omit_pids -o $1"
}
safe_shutdown () {
# for OpenElec before V5
touch "$LOCKDIR/$LOCKFILE"
# for OpenElec V5 and later
systemctl stop kodi
add_omit_pids $(pidof connmand)
add_omit_pids $(pidof dbus-dae-
mon)
killall5 -15 $omit_pids
for seq in `seq 1 10` ; do
usleep 500000
clear > /dev/tty1
killall5 -18 $omit_pids || break
done
sync
umount -a >/dev/null 2>&1
poweroff -f
}

echo "$GPIOpin1" > /sys/class/
gpio/export
echo "in" > /sys/class/gpio/
gpio$GPIOpin1/direction
while true; do
  sleep 1
  power=$(cat /sys/class/gpio/
gpio$GPIOpin1/value)
  if [ $power != 0 ]; then
    echo "out" > /sys/class/gpio/
gpio$GPIOpin1/direction
    echo "1" > /sys/class/gpio/
```

```
gpio$GPIOpin1/value
    sleep 3
    safe_shutdown
  fi
done
```

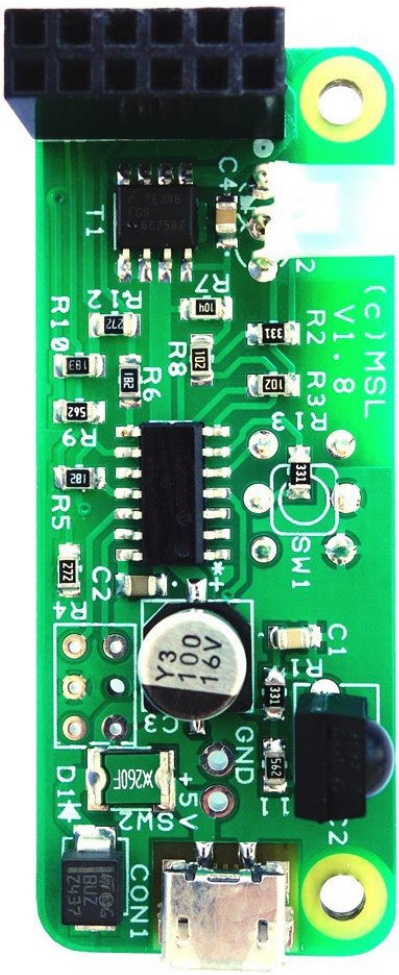
Shutdown.sh (solo cambia GPIOpin =15 a GPIOpin =249 y GPIOpin1 =14 a GPIOpin1 =225):

```
#!/bin/bash
if [ "$1" != "reboot" ]; then
# Raspberry Pi pin10: GPIOpin=15;
Odroid-C2 pin7: GPIOpin=249
  GPIOpin=249
# Raspberry Pi pin8: GPIOpin1=14;
Odroid-C2 pin26: GPIOpin1=225
  GPIOpin1=225
  echo "$GPIOpin" > /sys/class/
gpio/export
  # execute shutdown sequence on
pin
  echo "out" > /sys/class/gpio/
gpio$GPIOpin/direction
  echo "1" > /sys/class/gpio/
gpio$GPIOpin/value
  usleep 125000
  echo "0" > /sys/class/gpio/
gpio$GPIOpin/value
  usleep 200000
  echo "1" > /sys/class/gpio/
gpio$GPIOpin/value
  usleep 400000
  echo "0" > /sys/class/gpio/
gpio$GPIOpin/value
  # set GPIO 14 high to feedback
shutdown to RemotePi Board
  # because the irswitch.sh has
already been terminated
  echo "$GPIOpin1" > /sys/class/
gpio/export
  echo "out" > /sys/class/gpio/
gpio$GPIOpin1/direction
  echo "1" > /sys/class/gpio/
gpio$GPIOpin1/value
  usleep 4000000
fi
```

El script shutdown.sh es útil si a veces usas las aplicaciones remotas Android Yatse o Kore o hotkeys para apagar tu Centro Multimedia, por ejemplo. Esos comandos de apagado se equiparán a eventos internos, similar a cuando navegas por el menú de apagado de Kodi. Sin utilizar este segundo script, el sistema se cerrará con seguridad, pero la placa RemotePi no recibiría ninguna indicación para monitorizar el GPIO225, de modo que no cortaría la energía tras apagarse el sistema correctamente.

Usando el receptor IR de la RemotePi

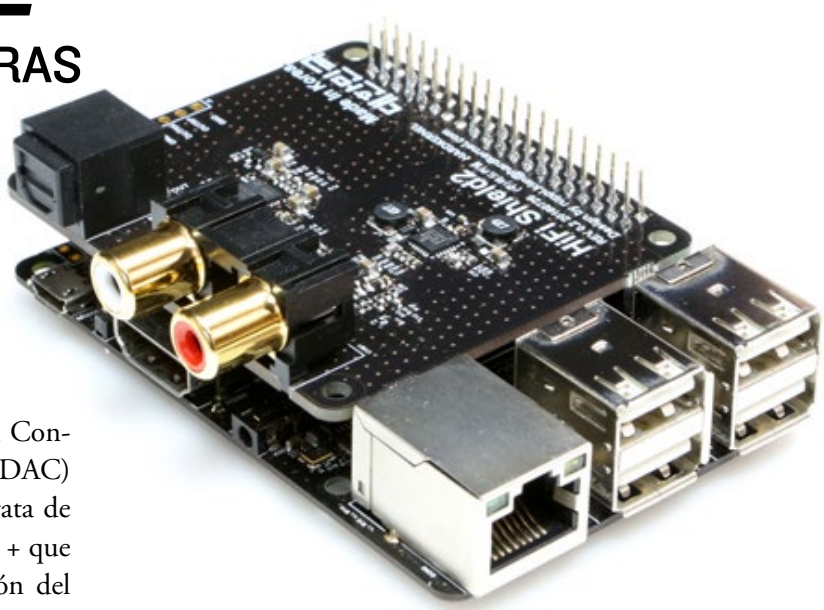
Si deseas utilizar el receptor IR de la RemotePi en lugar del receptor IR ODROID integrado, tendrás que desactivar el sistema IR integrado y activar el receptor IR GPIO. Para hacer esto en Ubuntu, puede consultar el artículo de la Wiki de Hardkernel en http://bit.ly/2l8KrWg. Para Libre-ELEC, echa un vistazo a mi mini-guía en http://bit.ly/2lLKj2A. Para preguntas, comentarios o sugerencias, visita el hilo original en http://bit.ly/2mgFGKk.



HIFI SHIELD 2

EL MEJOR SONIDO QUE PODRAS CONSEGUIR EN ANDROID

editado por Rob Roy (@robroy)



El HiFi Shield 2 (39$ http://bit.ly/2lHSlZJ) es un Convertidor Digital a Analógico de Alta definición (DAC) para el ODROID-C2 y el ODROID-C1+. Se trata de una placa de sonido especial para el ODROID-C2 y C1 + que está optimizada para mejorar la calidad de reproducción del sonido. Ofrece un sonido muy bien equilibrado: consistente, intenso, amplio y muy bien dimensionado por niveles. Hemos analizado la calidad de audio de la salida HiFi Shield 2 DAC con un equipo estándar de la industria del sonido llamado Audio Precision. Audio Precision es un analizador de audio de alto rendimiento optimizado para la producción de audio digital.

Usando el chip DAC PCM5242 de última generación de Texas Instrument, conocido como Burr-Brown, el HiFi Shield 2 soporta formatos de audio de 16, 24 y 32 bits con una ratio THD+N mínima (0,002%) y una dinámica ideal (114dB +), además de sorprendentes tasas de muestreo de 384 kHz. La interfaz dedicada S/PDIF soporta una definición de hasta 192kHz/24bit a través de una salida óptica (Toslink).

Si hacemos uso del puerto de expansión I2S en el C2/C1+, no es necesario ocupar un puerto USB, permitiendo al usuario seleccionar el sistema de reproducción de audio que desee, como Volumio y Debian (DietPi) para reproducir sonido HiFi.

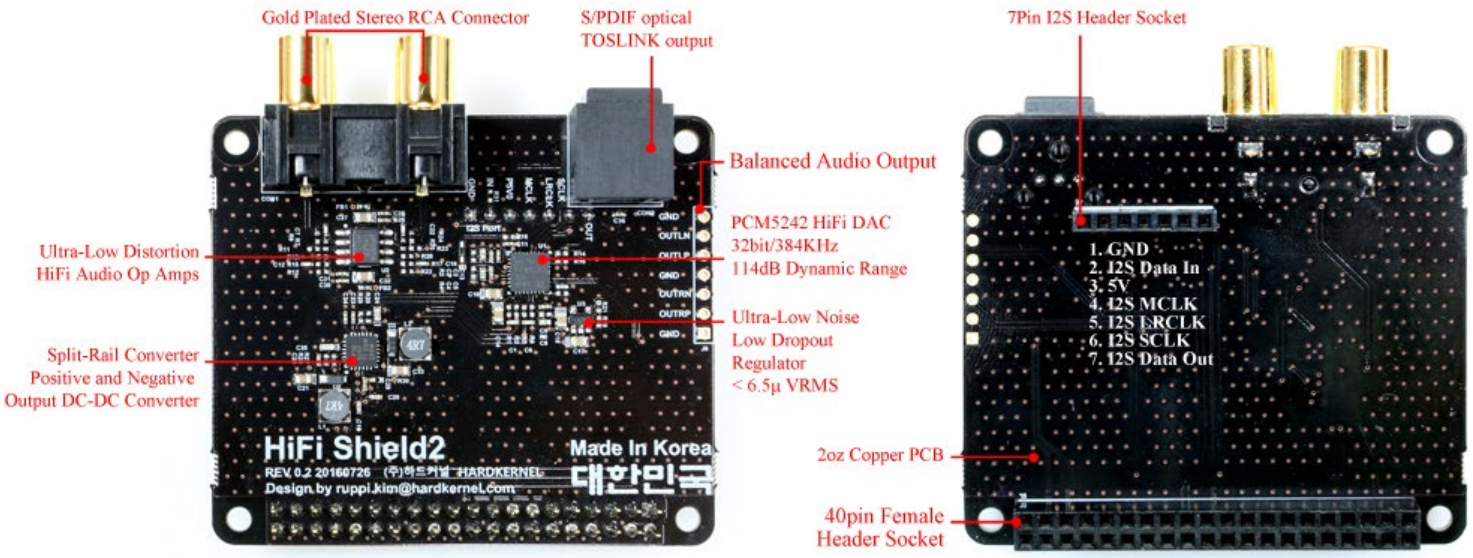
ODROID-C2 y HIFI Shield 2: Audio PB +J

La salida de audio es estándar, la salida de color rojo corresponde al canal de audio izquierdo y la salida de color blanco corresponde al canal de audio derecho.

Distribución de los 7 pines I2S del C2/CI+

ODROID-C2/C1+ 7pin Layout	
HEADER	Label
1	GND
2	I2S Data In (for ADC)
3	5V
4	I2S MCLK (Master Clock)
5	I2S LRCLK (LR Clock)
6	I2S SCLK (Bit Clock)
7	I2S Data Out (for DAC)

Primer plano de la placa con anotaciones



Características

- El control de volumen a través de la interfaz I2C es una gran característica para mantener una buena calidad del sonido con varios volúmenes de salida de audio
- Los puertos de salida incluyen terminales estéreo RCA chapados en oro
- La fuente de alimentación tiene habilitado un regulador de interferencias de ruido ultra-bajo, lo cual reduce el ruido de la fuente de alimentación y aumenta en gran medida el ratio señal/ruido
- La interfaz I2S permite la descodificación directa de la entrada digital a la salida analógica utilizando la sincronización del reloj maestro
- La superficie de la PCB está chapada en oro sobre 2 onzas de cobre, lo cual garantiza la continuidad de la señal y reduce la reflexión y deflación de la señal
- Dispone de almohadillas soldadas a la salida de audio balanceada (señal diferencial)
- La interfaz S/PDIF dedicada es nueva, admite una definición de hasta 192kHz/24bit a través de una nueva salida óptica (Toslink)
- Este HiFi shield no es compatible con Android, y Hardkernel no tiene planes para hacerlo compatible y así poder habilitar el controlador I2S en el Kernel y el HAL de Android.

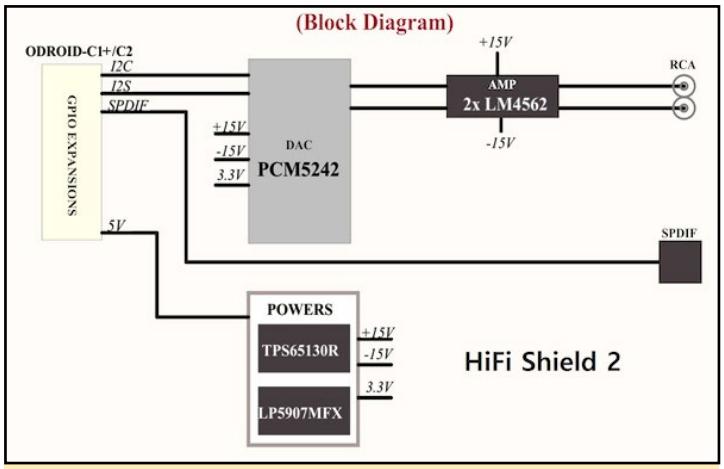


Diagrama por bloques del HiFi Shield2

Detalles

La guía de configuración de Ubuntu/Linux para ODROID-C2 está disponible en http://bit.ly/2brrGdG, y para ODROID-C1+ en http://bit.ly/2lI7AC7. El sistema operativo oficial Volumio 2 se puede descargar desde http://bit.ly/2kOJNAV, y la imagen DietPi basada en Debian se encuentra en http://bit.ly/2ls45yM. Puede ver los esquemas en http://bit.ly/2mnukV4.

MANUAL XU4 ACTUALIZADO

REVISADO PARA UBUNTU 16.04 Y LOS PERIFERICOS MAS RECIENTES

editado por Rob Roy (@robroy)

El Manual de usuario del ODROID-XU4, disponible en http://bit.ly/1U9Q8yg, ha sido revisado hace poco para incluir algunos de los periféricos más recientes, como la Expansion Board, SmartPower2 y oCam. Puesto que ahora Hardkernel también ofrece Ubuntu 16.04, todos los ejemplos de código se han actualizado para que sean compatibles con el nuevo sistema operativo. Si tiene comentarios, preguntas o sugerencias, visita el hilo de los foros ODROID en http://bit.ly/1RykBrT.

El manual del ODROID-XU4 incluye información detallada de los diferentes sistemas operativos, software y periféricos disponibles para el ODROID-XU4

HOME DATA CENTER

IMPLEMENTACION DEL CODIGO CON ARCHLINUX

por John Skilbeck

El DevOps es complicado. Los grandes proyectos de software, como Mesos y Kubernetes, desarrollan equipos como los que se encuentran en la mayoría de las empresas y organizaciones tecnológicas como CoreOS con la finalidad de ayudar a los desarrolladores a poner sus aplicaciones en funcionamiento. Sin embargo, ¿cuál sería una buena solución para un desarrollador en solitario, o una pequeña red doméstica? ¿Cómo podemos usar el código a nivel de infraestructura?

Uso un portátil Macintosh OSX para desarrollo, pero para las aplicaciones de larga duración o aplicaciones que se ejecutan por la noche, necesito un entorno remoto siempre activo, ya que mi portátil estará fuera de línea o en el tren conmigo mientras voy y vengo del trabajo. Tiene su sentido no utilizar el ordenador como un entorno para la puesta en funcionamiento de aplicaciones.

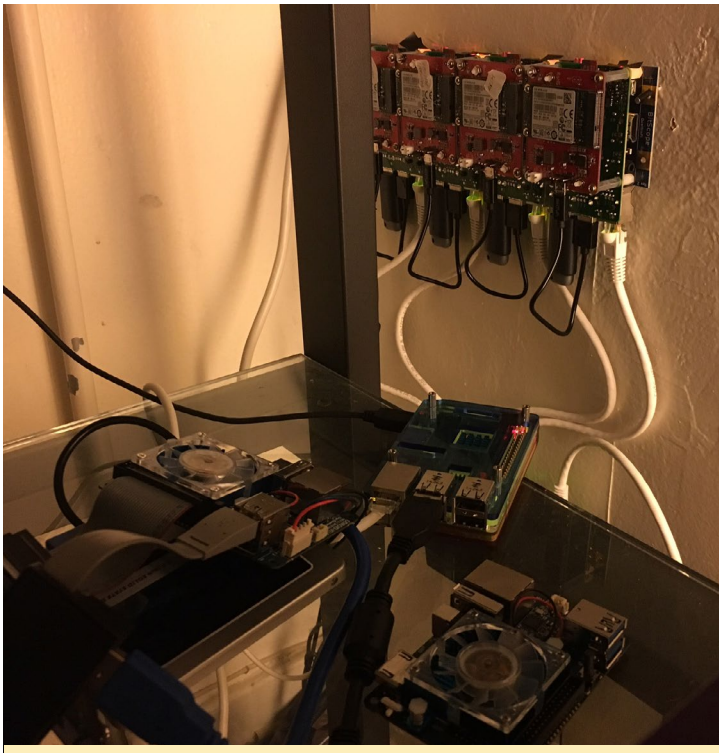
El ODROID-XU4 es un ordenador ideal para un entorno de implementación remoto, ya que es económico, flexible, cuenta con excelentes especificaciones técnicas y puede ejecutar Linux. Este artículo tiene por objetivo explicar cómo almacenar código de implementación en tu repositorio de proyectos, así como automatizar las implementaciones y ejecuciones.

Arch Linux

Arch Linux es una distribución de Linux gratuita de código abierto que fue lanzada en 2002. Está centrada en la elegancia, la integridad del código, el minimalismo y la simplicidad, y cuenta con que el usuario haga un esfuerzo en comprender el funcionamiento del sistema. Arch Linux utiliza en particular un modelo de revisión continua, de modo que todo lo que se necesitas para conseguir el software del sistema más reciente es actualizar el sistema de forma periódica.

Arch Linux puede ser algo difícil de asimilar, ya que utiliza herramientas diferentes a las de una distribución Debian. El gestor de paquetes se activa a través de “pacman” en lugar de “apt-get”, y existe un gestor complementario de paquetes muy popular llamado “yaourt”. Muchos de los servicios o herramientas más comunes no se instalan por defecto.

Arch Linux esta hecho principalmente para procesadores x86, pero un proyecto llamado Arch Linux ARM (ALARM) tiene una distribución ARM de Arch Linux para arquitecturas AArch64 ARMv8 y ARMv7. Hardkernel, el fabricante de ODROIDs, es de hecho un patrocinador del proyecto Arch Linux ARM.



El home data center de John es una auténtica obra de arte

Configurar la red

Deberás asignarle a tu dispositivo una dirección IP LAN DHCP reservada y en el mejor de los casos, un nombre de host que se propagará a lo largo de tu red por medio del servidor DNS de tu router. De esta forma, en nuestro entorno de desarrollo/local podremos utilizar un nombre de host para resolver siempre el entorno de implementación/remoto.

Por ejemplo, en mi red reservo el 192.168.2.49 a la dirección MAC de mi ODROID. También configuro una entrada DNS que asigna a esa dirección IP “odroid”. Utilizar un firmware personalizado en el router como Tomato USB o DD-WRT lo hace extremadamente fácil, ya que esos firmwares convierten tu router en un pequeño ordenador Linux con una aplicación web GUI muy pulida, aunque su implementación está fuera del alcance de este artículo. Si operas con subredes, asegúrate dirigir el puerto hacia el puerto externo que se asigna al puerto SSH del dispositivo odroid, ya que Git se ejecuta a través de SSH.

Configurar el proyecto

Lógicamente, desearás estandarizar el flujo de trabajo de la implementación. Esto hará que el trabajo con proyectos sea

extremadamente fácil y eliminas muchos de los cambios de contexto mental que utilizas cuando trabaja en varios proyectos. Crearemos una carpeta para alojar todos nuestros archivos relacionados con implementaciones. Coloca los archivos ejecutables en "deploy/bin", y cualquier archivo cron en "deploy/tasks" (más información sobre esta cuestión más adelante).

Navega hasta el directorio del proyecto en una ventana de Terminal y luego escribe los siguientes comandos:

```
$ mkdir -p deploy/bin
$ mkdir -p deploy/tasks
$ cd deploy/bin && touch run-job && \
  chmod u+x run-job && cd -
$ cd deploy/tasks && touch crontab
```

También puede estandarizar dónde colocar tu código fuente. De esta forma, le será más fácil a otras personas ver cómo se organiza tu proyecto, y conocer qué es el código fuente y qué no.

```
$ mkdir src
$ cd src && (place source code here, ie python: core.
py, clojure: core.clj, nodejs: app.js)
```

Simplificar los puntos de entrada

Iniciar una aplicación puede resultar algo confuso con todos los comandos que se pueden ejecutar en diferentes lenguajes. Por ejemplo, puede usar Java "java -jar [my-jar] .jar" o python "python my-app.py", y tu aplicación también puede necesitar varios argumentos. Todo esto debería simplificarse y resumirse en un archivo "deploy/bin/run-job":

```
#!/bin/sh
set -e
CMD="src/duck"
exec $CMD $@
```

Crear el archivo cron

Arch Linux no viene con un cliente o demonio cron por defecto. Instálalo con "sudo pacman -Syu cronie". Utilizando cron, puede ejecutar comandos en intervalos de tiempo especificados haciendo uso de la sintaxis especial de cron. Normalmente se almacena en el archivo crontab del usuario, al que puedes acceder con "crontab -e". Sin embargo, esto es demasiado manual, y queremos usar código como infraestructura. Cron también tiene algunos subdirectorios muy útiles en "/etc/cron.*", como "/etc/cron.daily" y"/etc/cron.hourly ", si colocamos los archivos aquí, éstos se ejecutarán en los intervalos especificados.

Revisa el archivo de "deploy/tasks/crontab" que colocaremos en "/etc/cron.d", que se crea automáticamente con nuestro script "posttreceive":

```
### variables ##
SHELL=/bin/bash
PATH=:/bin:/usr/bin:/usr/local/bin:/usr/sbin:/usr/lo-
cal/sbin
MAILTO=[your-email-address]@gmail.com
cmd="deploy/bin/run-job"
app_dir="/home/skilbjo/deploy/app/duckdns"

## jobs ##
5 * * * * skilbjo cd "$app_dir" ; $cmd >/dev/null
```

Aquí tienes el esquema general de la estructura de un simple proyecto. El único ejecutable del proyecto es un único script shell ubicado en "src/":

```
$ tree
.
├── README.md
├── deploy
│   ├── bin
│   │   ├── post-receive
│   │   └── run-job
│   └── tasks
│       └── crontab
└── src
    └── duck

4 directories, 5 files
```

Git

Primero queremos agregar una URL remota a nuestro proyecto en nuestro entorno local:

```
$ git remote add odroid ssh://odroid/~/deploy/git/
duckdns.git
```

Ten en cuenta que, dependiendo de la topología de tu red, es posible que tenga que modificar esta url. Si no puedes asignar nombres de host, la URL de git se vería así, donde 192.168.2.49 es la dirección IP LAN de tu dispositivo:

```
$ ssh://192.168.2.49/~/deploy/git/duckdns.git
```

Si tienes un usuario en tu entorno ODROID distinto al que tienes en tu entorno de desarrollo, la url se vería así, donde "skilbjo" es tu nombre de usuario:

```
$ ssh://skilbjo@odroid/~/deploy/git/duckdns.git
```

Si tu servidor remoto está en una subred diferente y tienes redireccionados los puertos, tu url tendría este aspecto, donde

"2222" es tu puerto externo:

```
$ ssh://192.168.1.2:2222/~/deploy/git/duckdns.git
```

En el directorio principal de tu entorno remoto, crea una carpeta llamada "~/deploy" con dos subcarpetas: "~/deploy/app" y "~/deploy/git". Los subdirectorios de "~/deploy/git" serán los puntos finales de nuestros desarrollos, y con un hook ejecutarán comandos de implementación en los subdirectorios de "~ /deploy/app".

Primero, navega hasta el directorio principal del entorno remoto y a continuación, escribe los siguientes comandos:

```
$ mkdir -p ~/deploy/app
$ mkdir -p ~/deploy/git
$ mkdir -p ~/deploy/git/duckdns.git
$ mkdir -p ~/deploy/app/duckdns
```

Ahora en "~ /deploy/git/duckdns.git/hooks", crea un archivo ejecutable llamado "post-receive", que se activará con cada acción hacia el punto final.

```
$ cd ~/deploy/app/git/duckdns.git/hooks
$ touch post-receive && chmod u+x post-receive
$ vim post-receive
```

Rellena el ejecutable con lo siguiente en el directorio "~/deploy/git/duckdns.git/hooks" del entorno remoto:

```
#!/usr/bin/env bash
set -eou pipefail

user=$(whoami)
dir="/home/${user}/deploy/app"
app=$(basename $(pwd) | sed -e 's/.git//')
deploy_dir="$dir/$app"
cron_dir="/etc/cron.d"

GIT_WORK_TREE="$deploy_dir" git checkout -f

cd "$deploy_dir"

## build steps here ##
case "$user" in
  (skilbjo) sudo cp deploy/tasks/crontab "$cron_
dir/$app" ;;
esac

## you can also do project-specific build steps in
this section, like install
## dependencies, ## (ie npm install), compile source
code (ie lein uberjar),
## as well as if a long-lived app, run commands as
well (ie java -jar my_jar.jar)

echo "all done"

exit 0
```

Implementación

Ahora estamos listos para implementar ya que nuestro entorno local está configurado para alcanzar el punto final del servidor de implementación, nuestro entorno remoto está configurado para recibir la notificación y revisar el código fuente, ejecutar cualquier paso de compilación y colocar una tarea en el directorio cron del sistema para su lanzamiento. Ponto todo en práctica con el siguiente comando en el directorio del proyecto del entorno local:

```
$ git push odroid
```

Además, para ver cómo se ha implementado todo esto en un proyecto de ejemplo, visita http://bit.ly/2lthYKW.

Pasos finales

Se puede añadir al flujo anterior algunas características como son múltiples entornos, ya sea con múltiples ODROIDs, o con un solo ODROID tratándolo como un servidor de servicio. Esto se puede hacer con subdirectorios bajo "~/deploy" como "~/deploy/staging/app/my_app" o "~ /deploy/production/app/my_app".

Además, podrías añadir un servicio de integración continua como CircleCI que ejecutaría una suite de testeo desde cada desarrollo a GitHub, y si tiene éxito, compila una imagen Docker. No podrías entonces tener un archivo en el entorno remoto que verificaría una imagen de un repositorio Docker y la ejecutaría en un intervalo especificado. Esto es lo que hacen muchos de los softwares DevOps distribuidos, como Mesos y Kubernetes, pero en un entorno mucho más rico que BASH y Linux.

Referencias

Artículo Wikipedia de Arch Linux http://bit.ly/2l71ADK
Website Arch Linux ARM https://archlinuxarm.org
Artículo que leí hace unos tres años y que me inspiró para escribir mi propio artículo http://bit.ly/2m5JAct

ODROID ARCADE BOX

LA EXPERIENCIA PERFECTA CON TUS JUEGOS ARCADE FAVORITOS

por Brian Kim, Charles Park y John Lee

Los ODROIDs tienen mejor rendimiento que las placas de la competencia, especialmente en lo que respecta al renderizado de video, lo que significa que las placas ODROID son ideales para jugar, lo cual hacen muchos usuarios de ODROID. Ya hay disponibles varios sistemas operativos de plataforma de juegos, como Lakka (http://bit.ly/1NO8BBC) y ODROID GameStation Turbo (http://bit.ly/1ASFO5O). Para disfrutar aún más de nuestras sesiones de juegos, creamos nuestra propia consola arcade con simples botones GPIO y joysticks, y la llamamos la ODROID Arcade Box. Elegimos un ODROID-XU4 para este proyecto porque es el que tiene mejor rendimiento GPU de todos los dispositivos ODROID actuales. En este artículo vamos a describir como hemos creado la ODROID Arcade Box.

Nuestro primer y simple prototipo

Requisitos

(Figura 3 –)

Decidimos hacer la ODROID Arcade Box usando un MDF (tablero de fibra de densidad media). El Shifter Shield del XU4 también es útil para utilizar los pines de expansión del ODROID-XU4. Los Joysticks, los botones y los cables eran los componentes de entrada, y se utilizó

una SMPS (Fuente de alimentación conmutada) como fuente de alimentación. Las herramientas y las piezas utilizadas las tienes en esta página:

La ODROID Arcade Box necesita un total de 27 entradas (19 entradas para botones y 8 entradas para joysticks). Las 24 entradas GPIO digitales del ODROID-XU4 no son suficientes para las 27 entradas, de modo que creamos dos puertos ADC adicionales para los tres botones que nos faltaban. Los valores de entrada ADC se basan en el voltaje de entrada y los valores de entrada digital y analógica se procesan en el demonio GPIO, que se describe a continuación.

Diseño y montaje

Los paneles de la ODROID Arcade Box deben ser diseñados y fabricados para que los botones y joysticks estén bien colocados. Elegimos MDF 12T teniendo en cuenta el precio y la durabilidad. El diseño se puede hacer con cualquier herramienta CAD como Google Sketch o SolidWorks. Aunque existen muchas plantillas de diseño para los paneles de joypad, decidimos elegir un diseño arcade estándar japonés.

El primer paso del montaje es unir la placa al panel MDF. Este paso fue fácil, aunque nos llevo más tiempo que el resto. Después, insertamos los joysticks, la toma de corriente, el interruptor y los botones en el panel MDF superior. Los conectores HDMI, Ethernet y USB se colocaron en la parte trasera del panel MDF. El siguiente paso fue montar cada panel MDF usando un taladro para hacer los agujeros, luego con tornillos los sujetamos.

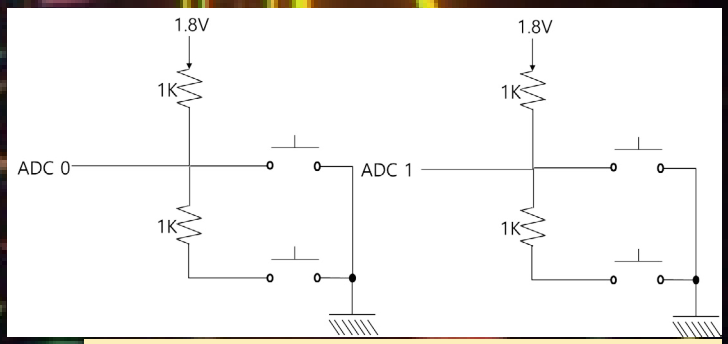
El último paso del montaje de la ODROID Arcade Box es conectar por cable los pines de expansión del ODROID-XU4 a los componentes de entrada. En este proyecto, diseñamos las entradas GPIO externas, tal y como se muestra en la página 26. Los botones Select y Temp están conectados a puertos de expansión ADC.

Configuracion del software

Desarrollamos un nuevo demonio GPIO llamado gpio_keyd (http://bit.ly/2ljOZKg). Este demonio es capaz de asignar entradas GPIO y eventos claves usando uinput y wiringPi, una librería de acceso GPIO basada en pines. Está diseñada para que le sea familiar a quiene ya ha utilizado el sistema de cableado Arduino. Aunque la blibrería wiringPi upstream sólo admite Raspberry Pi, Hardkernel ofrece una versión de wiringPi para ODROIDs en su repositorio GitHub (http://bit.ly/1Eq3UpF). El módulo uinput es un módulo del kernel de Linux que gestiona el subsistema de entrada del usuario. Se puede utilizar para crear y manejar dispositivos de entrada desde una aplicación.

Herramientas, compentes y lo que hay justamente debajo:

Panel MDF 12T	Extendedores HDMI, USB, Ethernet
2EA 600x220	Enchufe de alimentación y Switch
2EA 600x75	Bisagras 2EA
2EA 220x75	Receptor de puerta
Taladro	Caucho 4EA
Plegadora	Tornillos
Desmontador	Pulsadores 19EA
Cinta métrica	Joystick 2EA
Navaja multiuso	Cables
Alicates de punta larga	Terminales
ODROID-XU4	
XU4 Shifter shield	
SMPS (Fuete Alimentación)	



Esquema de los puertos de expansión

Elegimos ODROID GameStation Turbo (http://bit.ly/1ASFO5O) como plataforma de software para nuestra ODROID Arcade Box, que tiene integrado uinput. Debes asegurarte de que el archivo de dispositivo uinput existe en el sistema operativo elegido, porque algunos no tienen dispositivos uinput.

```
$ ls /dev/uinput
```

Si tu sistema operativo no tiene un archivo de dispositivo /dev/uinput, entonces será necesario recompilar e instalar un nuevo kernel con la opción de configuración INPUT_UINPUT habilitada. La página Wiki en http://bit.ly/1YITtoBI describe cómo compilar e instalar la imagen del kernel a partir del código fuente.

```
$ make menuconfig
Device Drivers
-> Input device support
    -> Generic input layer
      -> Miscellaneous device
        -> User level driver support <*>
```

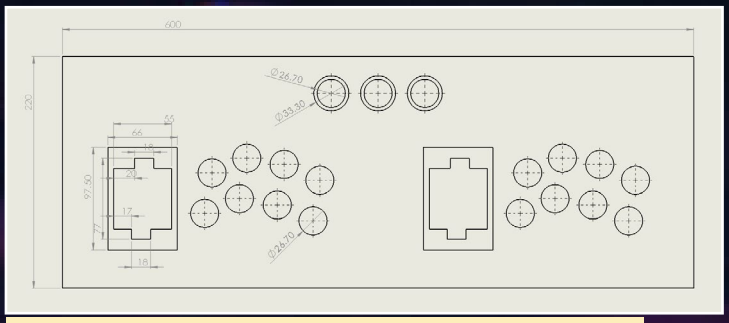
Ten en cuenta que wiringPi debe instalarse antes de instalar gpio_keyd. En la imagen GameStation de ODROID, los comandos sudo deben ejecutarse como root, porque la cuenta “odroid” no está designada como usuario sudo.

```
$ git clone https://github.com/hardkernel/wiringPi.
git
$ cd wiringPi
$ sudo ./build
```

Descarga el código fuente gpio_keyd, el cual está disponible en nuestro repositorio GitHub. El procedimiento para compilar e instalar gpio_keyd es muy simple:

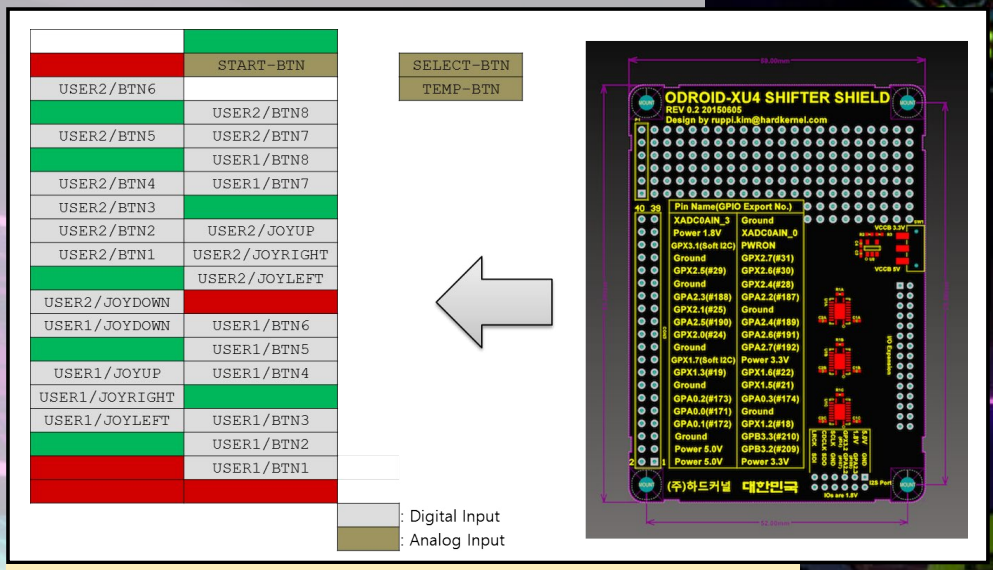
```
$ git clone https://github.com/bkrepo/gpio_keyd.git
$ cd gpio_keyd
$ make
$ sudo make install
```

El script gpio_keyd hace referencia a /etc/gpio_keyd.conf como el predeterminado para la información de los botones y GPIO. El archivo de configuración fue modificado para las 27 entradas de la ODROID Arcade Box. Algunas botones ya se utilizaban en el emulador de juego, de modo que tuvimos que cambiar la configuración del emulador para evitar conflictos entre los botones de entrada GPIO y el emulador. Ten en cuenta que el campo <GPIO pin> en el archivo de configuración hace referen-



Plano del diseño de joypad

La ODROID Arcade Box montada



Mapeo GPIO externo para los botones y Joysticks

cia al número wiringPi, no al GPIO y al número de pin (http://bit.ly/2lbzPIB).

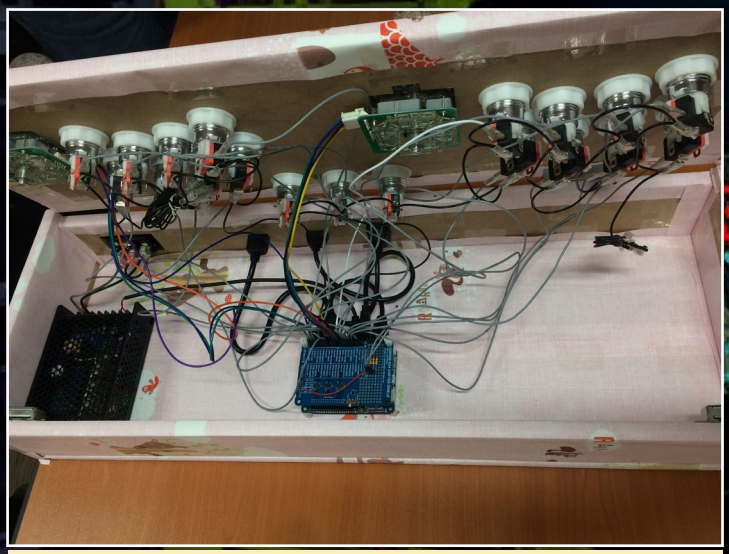
Ejemplo de configuración para las 27 entradas: /etc/gpio_keyd.conf

```
# Digital input
# <Key code>  <GPIO type>     <GPIO pin>      <Active
value>
# User 1
KEY_LEFT      digital         15              0
KEY_RIGHT     digital         1               0
KEY_UP        digital         4               0
KEY_DOWN      digital         16              0
KEY_A         digital         2               0
KEY_S         digital         3               0
KEY_D         digital         30              0
KEY_F         digital         21              0
KEY_Z         digital         8               0
KEY_X         digital         9               0
KEY_C         digital         7               0
KEY_V         digital         0               0
# User 2
KEY_BACKSLASH digital         12              0
KEY_SLASH             digital         13              0
KEY_SEMICOLON digital         14              0
KEY_LEFTBRACE digital         5               0
KEY_Y         digital         26              0
KEY_U         digital         27              0
KEY_I         digital         22              0
KEY_O         digital         23              0
KEY_H         digital         6               0
KEY_J         digital         10              0
KEY_K         digital         11              0
KEY_L         digital         31              0

# Analog input
# <Key code>  <GPIO type>     <ADC port>      <ADC ac-
tive value>
KEY_B         analog          0               0
KEY_N         analog          0               2045
KEY_M         analog          1               2045

To run gpio_keyd daemon at every startup is conve-
nient for ODROID Arcade Box.

/etc/init.d/gpio_keyd
#! /bin/sh
### BEGIN INIT INFO
```



Cableado de la ODROID Arcade Box

```
# Provides:          gpio_keyd
# Required-Start:    $all
# Required-Stop:
# Default-Start:     2 3 4 5
# Default-Stop:
# Short-Description: Run /usr/bin/gpio_keyd if it ex-
ist
### END INIT INFO


PATH=/sbin:/usr/sbin:/bin:/usr/bin

.  /lib/init/vars.sh
.  /lib/lsb/init-functions

do_start() {
      if [ -x /usr/bin/gpio_keyd ]; then
              /usr/bin/gpio_keyd -d
              ES=$?
              [ "$VERBOSE" != no ] && log_end_msg $ES
              return $ES
      fi
}

case "$1" in
    start)
      do_start
        ;;
    restart|reload|force-reload)
        echo "Error: argument '$1' not supported" >&2
        exit 3
        ;;
    stop)
      killall gpio_keyd
        exit 0
        ;;
    *)
        echo "Usage: $0 start|stop" >&2
        exit 3
        ;;
Esac

$ sudo chmod +x /etc/init.d/gpio_keyd
$ sudo update-rc.d gpio_keyd defaults
$ sudo reboot
```

En los comandos anteriores, el script gpio_keyd se ejecuta como un demonio utilizando la opción "-d". El uso de gpio_keyd se puede comprobar con la opción "-h". Comprueba de nuevo las teclas usadas por el juego o el emulador, después fija la configuración del gpio_keyd correctamente. Es ahora cuando estás listo para jugar y disfrutar de tus juegos con tu nueva ODROID Arcade Box.

The King of Fighters 98, John vs Brian

DESARROLLO ANDROID

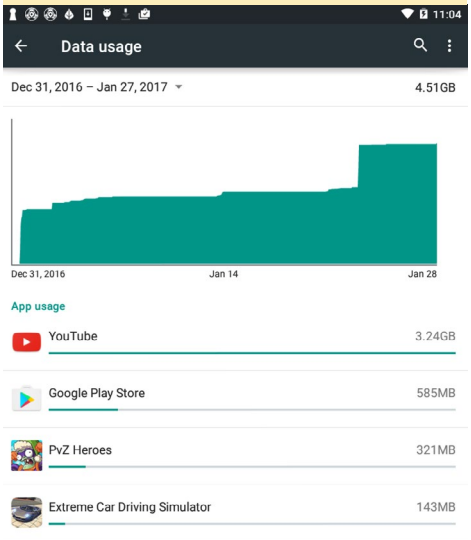
ANALIZANDO EL USO DE RED DE LAS APLICACIONES

por Nanik Tolaram

Como desarrolladores, queremos que nuestras aplicaciones sean eficientes y a menudo nos gustaría saber cuánto ancho de banda de red está utilizando nuestra aplicación. Esto es muy útil por varias razones:

- Para monitorizar y asegurarnos de que la aplicación es realmente la nuestra y no una aplicación que ha sido pirateada y publicada con diferentes nombres en Play Store
- Para asegurarnos de que no estamos cogiendo ancho de banda innecesario del plan de datos del usuario
- Para comprobar que el dispositivo que estamos usando no está accediendo a Internet sin nuestro consentimiento.

Figura 1 - Uso de datos en la configuración



En este artículo, analizaremos las diferentes formas que existen para obtener los datos del uso de la red.

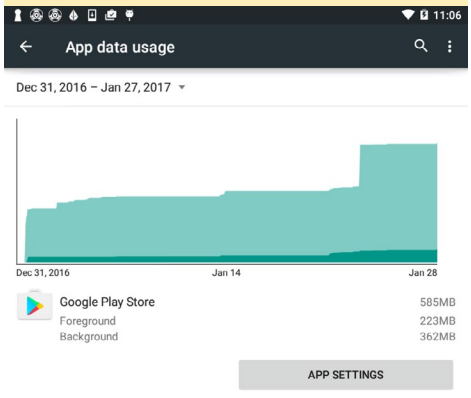
Configuración

La forma normal de analizar los datos de tráfico de la red es mediante el uso de datos a través de las aplicaciones de configuración del sistema, tal y como se muestra en la figura 1.

El uso de datos muestra la cantidad total de tráfico de datos entrante y saliente que ha sido utilizada por una aplicación. Si seleccionas la aplicación, verá una pantalla como la que aparece en la Figura 2 que muestra información detallada sobre el uso de la aplicación tanto en primer plano como de fondo.

La información que se muestra dentro de la aplicación de Configuración se almacena en /data /system/netstats, que requiere acceso root. Al eliminar todo lo que hay dentro de esta carpeta, se restablecen las estadísticas de red mostradas en esta aplicación.

Figura 2 - Desglose del uso de datos de la app



Estadísticas de la red

La aplicación de configuración nos puede dar una información general sobre las estadísticas de la red de datos, que es un buen comienzo, pero a veces necesitamos un análisis más detallado, que puede hacerse a través del comando dumpsys. Android cuenta con una poderosa herramienta llamada dumpsys que nos permite obtener una instantánea o volcado del sistema, que puede incluir información sobre la red, la memoria y otros componentes. Lee la documentación de Android en http://bit.ly/2kK9dep para obtener más información del sistema. Para conseguir más información relacionada con la red, nos interesa el comando:

```
$ dumpsys netstats detail
```

La Figura 3 muestra un pantallazo de lo que puedes ver desde una tablet Nexus 7 que ejecuta Lollipop 5.1.1.

Hay algunas cuestiones importantes para entender las estadísticas:

- Las estadísticas UID muestran el desglose de las aplicaciones de primer plano y de fondo
- Uid muestra el ID de usuario de la aplicación que se puede utilizar para relacionar ésta con la in-

Figura 3 – Estadísticas de red con Dumpsys

formación del paquete (que veremos en la sección de detalles del paquete dumpsys) con el objeto de conocer qué aplicación utiliza estas estadísticas de red.

- Las etiquetas son útiles si desea ver la cantidad de datos que utiliza cada conexión de tu aplicación. Puedes ver un ejemplo de esto en la Figura 4, que muestra que el uid 10007 tiene las siguientes etiquetas: 0x30000401, 0x30003201, 0x30002804, 0x10000401 y 0x30000407. Estas etiquetas ayudan al desarrollador a saber cuántos datos utiliza cada parte de la aplicación. Estas estadísticas de red en particular están vinculada al paquete Google Framework que forma parte de Google Play Store.

En el siguiente apartado veremos cómo escribir una aplicación etiquetando el hilo que hace la conexión de red, para determinar la cantidad de datos que se utilizan.

Estadísticas de paquetes

En la sección anterior, analizamos el uid que aparece al ejecutar el comando dumpsys netstats. Ahora necesitamos saber qué uid corresponde a la aplicación en el dispositivo y esto se hace mediante el siguiente comando:

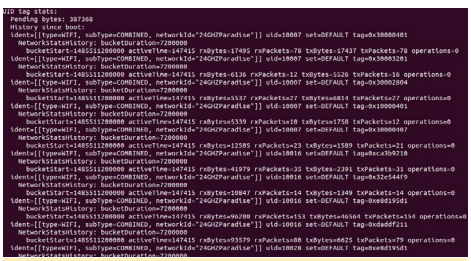


Figura 4 - Estadísticas de etiquetas UID

Figura 5 - Detalles del paquete dumpsys

```
$ dumpsys package detail
```

Este comando te proporcionará una larga lista de información sobre los paquetes que están instalados en tu dispositivo. Para las estadísticas de red, nos interesa la sección que se muestra en la Figura 5.

La sección resaltada muestra el userId que se relaciona con el uid que se muestra en las estadísticas de red. Al relacionar este número, sabemos el nombre del paquete de la aplicación sobre el que estamos leyendo las estadísticas de red.

Aplicación de ejemplo

La aplicación de ejemplo de este artículo se puede descargar desde http://bit.ly/2m8WenA, que muestra cómo etiquetar la conexión de red desde distintas

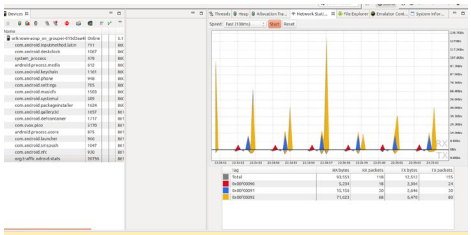


Figura 6 - Etiquetas de red

secciones de la aplicación, lo cual puede ayudar a analizar el volumen de tráfico de datos de la aplicación. La Figura 6 muestra un ejemplo de cómo lo vemos si utilizamos Android Device Monitor (dentro de Studio, dirígete a Tools -> Android -> Android Device Monitor).

Tal y como se puede ver en el gráfico, la simple aplicación tiene 3 conexiones diferentes, y cada una de ellas está accediendo a un sitio web diferente. Al etiquetarlas con códigos diferentes, podemos ver la cantidad de tráfico que consume cada conexión, y esto nos ayudará a ver si nuestra aplicación utiliza el ancho de banda de un modo eficiente.

El código del siguiente fragmento muestra el uso de la API TrafficStats interna. Esta etiqueta API muestra el subproceso que utiliza actualmente un socket con un identificador y todo el tráfico que pasa por la misma conexión se grabará junto. Así es como la aplicación es capaz de agrupar las 3 conexiones diferentes. Internamente, Android utiliza el módulo xt_qtaguid del kernel para realizar este tipo de etiquetado. Tienes disponible más información en http://bit.ly/2kU8KXB.

```
public void getUsingNormalURLC
onnection(String urlText, int
threadTag)
                throws IOException
{
    URL url = null;
    TrafficStats.
setThreadStatsTag(threadTag);
    . . . .
    HttpURLConnection urlConnec-
tion = getConnection(url);
    try {
        . . . . .
        }
    } finally {
        TrafficStats.clearThread-
StatsTag();
    }
}
```

DETECTAR LA PRESENCIA

CRONICAS DE UN CIENTIFICO LOCO

por Bo Lechnowsky

Disgustado, te subes a tu 4x4 de dominación del mundo, un vehículo que desde fuera parece más bien un viejo camión de granja, pero por dentro se parece más al sistema de control de un transbordador espacial, porque el conductor del restaurante oriental que realiza las entrega a domicilio está enfermo. Ahora tienes que ir a recoger tu propia comida. "Qué rudimentario", piensas cuando sales de tu garaje secreto. Justamente cuando vas a parar el vehículo en un stop, oyes un fuerte ruido. Sales y te diriges rápidamente a la parte posterior del 4x4 donde observas un cubo de basura. Te tiras de los pelos y piensas: "¡Esos basureros Neanderthales dejaron otra vez el cubo en medio!" Ahora te toca recoger los restos de tu cena y la basura de otros.

Conforme te vas alejando para hacer tus recados, empiezas a pensar en cómo evitar este inconveniente en el futuro. Varias ideas te vienen a la mente:

Un sistema Lidar para escanear en 3D el entorno en tiempo real y alarmas sonoras cuando sea necesario
Un Brazo cibernético colocado en la parte posterior del 4x4 que procese video para detectar objetos, agarrarlos y moverlos
Una simple camara trasera

Empiezas a considerar la viabilidad de cada una:

Muy bueno, pero demasiado complicado y costoso
Aún mejor y un proyecto propio de un científico loco, pero demasiado complicado y costoso
Ya tengo dos, ¡pero estaba demasiado preocupado en mis pensamientos para prestarles atención!

Lo que necesitas es algo simple, sin complicaciones y barato que haga sonar una alarma para alertarte, incluso cuando estar absorto en tus pensamientos. "Aha!" Recuerdas haber visto un nuevo producto en ameriDroid que puede hacer precisamente esto. Se trata de un microcontrolador conectado al USB que soporta hasta 6 sensores ultrasónicos con simples comandos en serie (http://bit.ly/2l26ptV). "¡Perfecto!", Piensas.

Regresas a tu laboratorio subterráneo, con el pollo Kung Pao en la mano, te apresurar a encender tus monitores, entras en el sitio web ameriDroid, y pides un kit "USB Ultrasonic Ranging Sensor" con 6 sensores y carcasas.

Vas sintiendo una sensación de alivio a medida que regresas a los 12 proyectos en lo que ya estabas trabajando a la espera de que pasen un par de días hasta que llegue tu nuevo pedido. Mientras te comes tu Kung Pao, dibujas un diagrama de cómo quieres colocar los sensores en la parte posterior de tu vehículo.

Un par de días más tarde, el kit llega. Inmediatamente empiezas a montar el kit y los sensores en la parte posterior de tu 4x4, teniendo cuidado de usar sellador de silicona para proteger los sensores de la lluvia, aguanieve y nieve que puedas encontrar durante tus aventuras para dominar el mundo.

A continuación, consultas las instrucciones proporcionadas por ameriDroid con el kit sobre cómo controlar y leer los sensores desde tu fiel ODROID-C2 y VU7 que montaste en el salpicadero para este propósito.

Conectando el cable USB

El cable USB suministrado debe conectarse al microcontrolador (la pequeña placa de circuito con filas de pines que sobresalen). El microcontrolador tiene una fila de 4 o 6 pines

en un extremo. Si tiene 6 pines, céntrate en los 4 pines centrales y sigue estas instrucciones:

Negro - Conectar a GND
Rojo - Conectar a VCC
Verde - Conectar a RXI
Blanco - Conectar a TXD

Conectando los sensores ultrasónicos

En cada sensor ultrasónico, hay cuatro pines marcados: Vcc, Trig, Echo, Gnd. Para el primer sensor ultrasónico, conecta el pin 2 del microcontrolador al pin "Trig" del sensor ultrasónico. Conecta el pin 3 del microcontrolador al pin "Echo" del sensor. Conecte el pin "Vcc" del sensor a DC 5V (el pin "VCC" del microcontrolador suministra 5V) y el pin "Gnd" del sensor a una conexión a tierra, como por ejemplo los pines "GND" del microcontrolador.

Si conectas un segundo sensor de ultrasonidos, conecta "Trig" al pin 4 y "Echo" al pin 5. Continúa conectando los siguientes pines a los sensores, hasta los pines 12 y 13 si al final optas por conectar los seis sensores ultrasonidos.

Conexión del software

Utilizas un programa terminal como PuTTY, Screen o tu lenguaje de script o programación favorito para conectarte al puerto serie que aparece cuando te conectas el sensor ultrasónico. Esto variará en función del sistema operativo que utilizas para conectarte al microcontrolador. En Windows, consultar el "Administrador de dispositivos" es la forma más común de averiguarlo. En Linux, "dmesg" o "lsusb" son las formas de detectar a qué puerto está conectado. Deberías ver algo denominado "PL2303", que es la conexión del microcontrolador. Si recibes información basura en tu terminal cuando te conectes, asegúrate de que los ajustes en serie son 9600 baudios, 8 bits, ninguno (paridad) y 1 bit de parada. Aquí tienes una lista de los comandos que puede aceptar el microcontrolador:

- debug on: Activa la retroalimentación detallada. Está deshabilitada por defecto, nos proporcionará aclaraciones sobre los errores de entrada.
- debug off: desactiva la retroalimentación detallada.
- init x y: activa un HC-SR04 en los pines especificados: X es trigger, Y es echo. Estos deben ser pines digitales. El dispositivo 0 está preestablecido en los pines 2 y 3, de modo que no es necesario especificar estos pines. La selección de pines no válidos devolverá un signo de exclamación (!).
- ping: envía una sola lectura del HCSR04 previamente utilizado. Si no se ha utilizado todavía, se usará el dispositivo 0. Si al comando le sigue un espacio y un número entre 0-100 (exclusivo), se realizará ese número de lecturas, emitiendo primero el promedio de esas lecturas, después el número de lecturas fallidas (que no cuentan para el promedio), y luego los valores mínimo y máximo de lectura separados por espacios. En cualquier caso, si una señal se desconecta automáticamente, devolverá -1.
- start: Hace ping constantemente a todos los dispositivos conectados en secuencia. Debido a que los dispositivos se utilizan uno a uno, el flujo de datos de este comando será más rápido si los dispositivos miden distancias cortas. Si este comando le sigue un espacio y un número mayor que 0, la placa esperará estos milisegundos entre la lectura del último dispositivo y la lectura del primer dispositivo. El resultado de este comando será la lectura de cada dispositivo (con un único espacio después de cada uno) y una vuelta tras el último dispositivo.
- stop: detiene la operación "start", sólo es efectivo tras haber iniciado "start".
- mode: emite el modo de medición del flujo: M para métrico (milímetros), I para Imperial (décimas de pulgada) o R para Raw (la longitud de pulso en microsegundos devuelta por el HC-SR04). El valor predeterminado es M.
- timeout: emite el tiempo de espera actual para lecturas de la señal. El tiempo de espera es la mayor cantidad de microsegundos que la placa esperará antes de declarar que el HC-SR04 ha realizado una mala lectura, lo cual no es necesariamente lo mismo que el pulso más largo que la placa aceptará del HC-SR04. Cuando la placa mide el pulso del HC-SR04, primero espera a que el HC-SR04 comience a enviar un pulso. Este tiempo cuenta para el tiempo de espera. Cuando este comando le sigue un espacio y un número, el tiempo de espera se ajusta a ese número. El valor predeterminado es 1 segundo (1.000.000 microsegundos).
- ver: emite información de versión y créditos.

Si cualquier comando (o ningún comando) está precedido por un único dígito de 0-5 (inclusive) seguido de un espacio, el dispositivo que corresponde a ese dígito se seleccionará para ser usado con el siguiente comando "ping". Cada vez que la placa esté lista para un comando, emitirá ">" para la serie. La única excepción está tras el comando "start", que no solicitará entrada hasta que sea reconocido "stop".

CONOCIENDO UN ODROIDIAN

VIACHESLAV ALEKSEEV

editado por Rob Roy (@robroy)

Por favor, háblanos un poco sobre ti

Soy ingeniero de electrónica y software en Rusia desde hace 47 años. Nací en Siberia y tras terminar la escuela, me mudé a Moscú para estudiar en la universidad. Ser estudiante en el Instituto de Aviación de Moscú (MAI) fue una experiencia increíble, probablemente la mejor de mi vida. A principios de los 90, llegó a Rusia la era de los microordenadores, solía jugar a un juego de saltos en una CPU z80 y el i486. Tras finalizar mis estudios universitarios y de posgrado, me convertí en ingeniero. Trabajé en unos cuantos puestos de trabajo para diferentes empresas, y finalmente decidí emprender mi propio negocio. En 2006, puse en marcha una iniciativa para crear sistemas de recuento del tráfico de automóviles. Estoy casado y tengo una hija de 21 años llamada Lena, que trabaja como enfermera. Mi esposa Nadezhda es diseñadora de vestuario de producción, pero ahora trabaja como diseñadora web. Ella es aficionada a la fotografía digital y le encanta su cámara réflex digital.

1995, en el campus MAI, probando uno de los primeros equipos VR comerciales del mundo, que funcionaba a 640x480 a 30fps. Utilizaba un sensor magnético en la nuca para la posición de la cabeza. La mejor broma que se podía gastar a alguien era colocar un imán cerca de la espalda y sacudirlo. ¡Los gritos estaban garantizados cuando el mundo en 3D daba vueltas sin cesar!

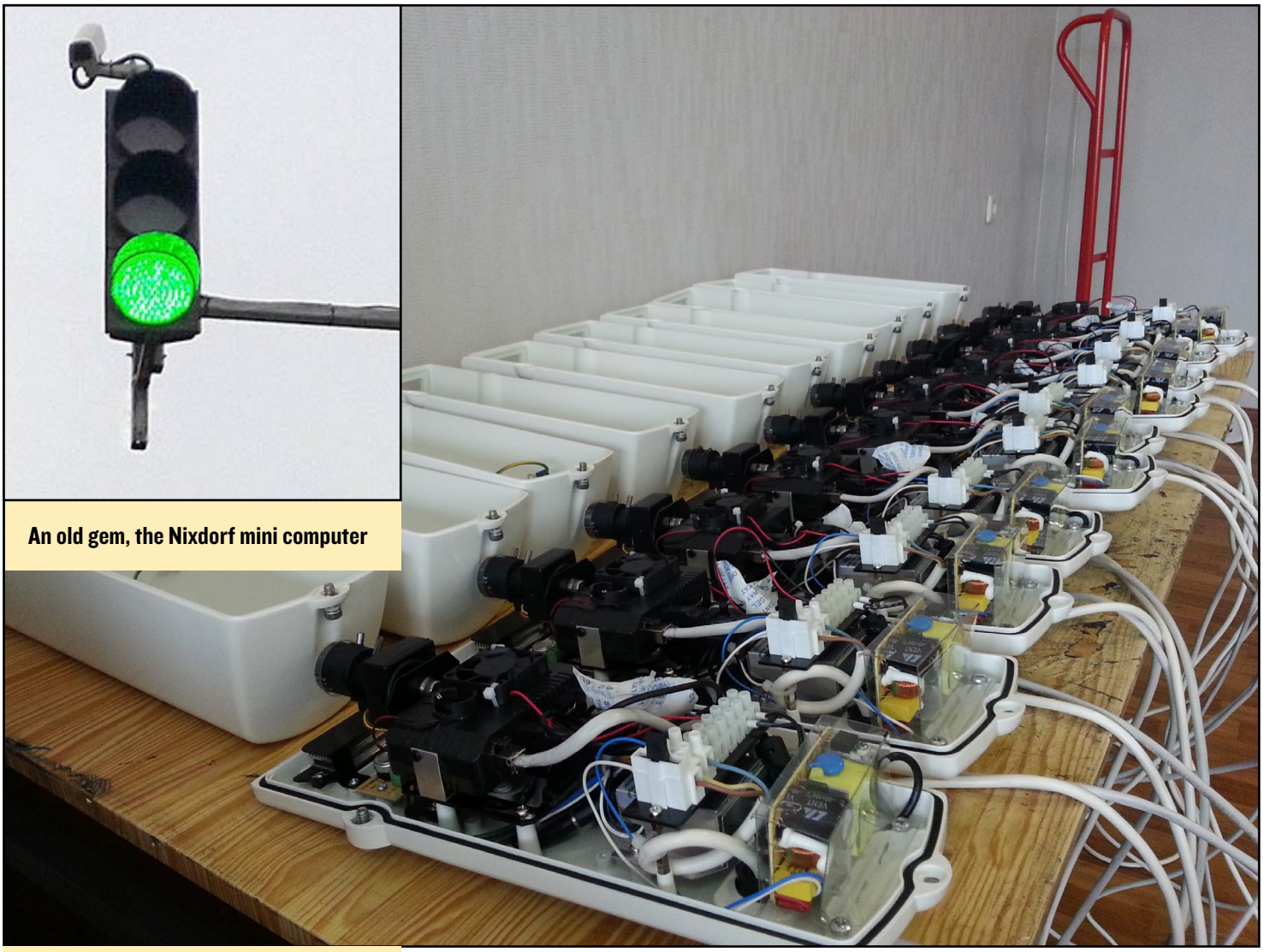
Viacheslav disfruta tocando la guitarra y escuchando música

¿Cómo empezaste con los ordenadores?

Cuando empecé mi negocio, tuve que decidir qué plataforma de hardware iba a utilizar para la recogida de datos del tráfico. Simplemente para que no pienses mal de mí, mis cámaras de tráfico no son las que se usan para multar por exceso de velocidad o algo así. Mi sistema estaba destinado suavizar el flujo de tráfico fijando los controles óptimos de las señales de tráfico. Anteriormente, usaba un PC industrial para ejecutar mi software de reconocimiento de imágenes. Ahora uso la familia de dispositivos ODROID para esta cuestión.

¿Qué te atrajo de la plataforma ODROID?

En 2012, usé Google para buscar una plataforma que fuera muy compacta y al mismo tiempo potente para que cubriera mis necesidades, que resultó ser la placa



An old gem, the Nixdorf mini computer

Una vieja joya, el mini ordenador Nixdorf

ODROID-X2. Con sus cuatro núcleos funcionando a 1,7 GHz, calculaba con facilidad los algoritmos de reconteo del tráfico de automóviles. Más tarde, cambié al U3 y el XU4.

¿Cuál es tu ODROID favorito y por qué?

De momento, mi favorito es el XU4. Mi sistema está basado en el tratamiento de fotogramas de video en tiempo real, por lo que el bus USB 3.0 del XU4 es bastante bueno para una cámara con una interfaz de captura de vídeo. Para mi sistema, es muy importante tener una buena conexión entre la cámara de vídeo y la CPU. Probablemente tendré que echar un vistazo a las interfaces de nivel inferior como MIPI CSI-2, que por desgracia están ausente en el XU4. Recientemente, he estado aprendiendo como usar la cámara oCam GS BW. La cámara es buena en sí misma, pero para un uso industrial al aire libre, tengo que implementar un software de control de exposición automática y posiblemente de control de apertura de la lente, que todavía está en estudio.

¿Qué aficiones e intereses tienes aparte de los ordenadores?

Disfruto conduciendo y leyendo sobre coches, y viendo los campeonatos de carreras de F1. También me gusta viajar, pescar, escuchar música y tocar la guitarra. Tengo un gato siberiano llamado Leia, y soy fan de Star Wars.

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

Hacer software es algo increíble y místico. Es una fusión de arte y tecnología. Nada te inspira más que tu aplicación cuando empieza a cobrar vida. Sin embargo, siempre está ahí el otro lado de la Fuerza. Estar listo para instruirte incesantemente. El desarrollo de software es una de las actividades que cambia más rápido. Si tienes pensado dejar de hacer software en uno o dos años, puede dejar de aprender ahora. El conocimiento se volverá anticuado en un par de años más o menos. Para estar en lo más alto, siempre tienes que correr. Es similar a la canción"Run like Hell" de Pink Floyd. Si estás listo para vivir de esta forma, seguro que tendrás éxito.