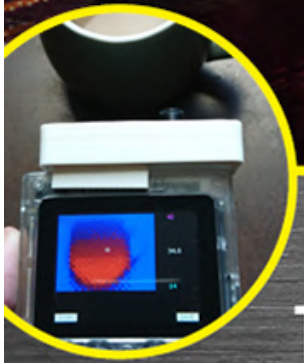
emuELEC • Lutris • Consola Juegos Retro • MAME • Zoneminder Parte 2

ODROID Magazine

Año Seis
Num #67
Jul 2019

ALMACENAJE NAS

UN PROYECTO PROFESIONAL ODROID-H2



CAMARA TERMICA:

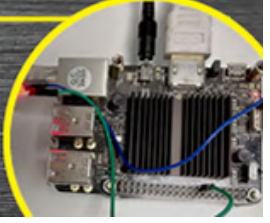
TU ODROID-GO PORTATIL, AHORA ACOMPAÑADO DE UN MODULO TERMINO INFLARROJOS

HAZTE ECOLOGICO CON "ENVI":

UN SENSOR COMBO MEDIOAMBIENTAL QWIIC

YOCTO + ODROID-C2:

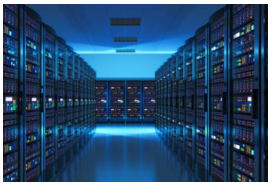
USANDO YOCTO CON KERNEL 5.0



Vuélverte Ecológico con “Envi”: Un Sensor Combinado Ambiental Qwiic para tu Estimada Máquina de Juegos

July 1, 2019

Un sensor respetado que proporciona la humedad y la temperatura del ambiente es el sensor ambiental digital integrado BME280 de Bosch Sensortec.



Monta un Almacenamiento de Acceso en red (NAS) de 6 bahías por tu Cuenta: Aprovecha el Potencial del ODROID-H2

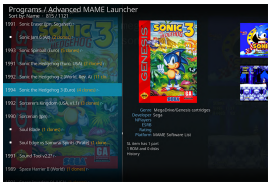
July 1, 2019

Cómo montar un Servidor NAS de 6 bahías con ODROID-H2

El Punto G: Tu Destino para todas las Cuestiones Relacionadas con Juegos Android

July 1, 2019

Los próximos meses del verano podrían ser muy emocionantes para los jugadores de Android. He incluido en mi calendario Google Stadia, E3, y algunos lanzamientos de videojuegos muy esperados y de gran renombre. Uno de estos grandes lanzamientos de juegos es Elder Scrolls: Blades. He estado elogiando este título durante ▶️



Kodi y Mame Avanzado en ODROID-XU4 - Parte 2

July 1, 2019

Esta es la continuación de una guía para configurar Kodi con Mame, que detalla cómo instalar el joystick. En el mejor de los casos, jugar con MAME requiere de un buen joystick. Aquí tienes dos ejemplos de joystick que he creado yo mismo. Es un buen ejercicio de carpintería, pintura, ▶️

Zoneminder - Parte 2: Compilando el Paquete Desde la Fuente en el ODROID-XU4

July 1, 2019

ZoneMinder es un conjunto integrado de aplicaciones que proporciona un completo sistema de vigilancia permitiendo la captura, análisis, grabación y monitorización de cualquier CCTV o cámaras de seguridad.

Cómo Crear una Consola de Juegos Retro Monku - Parte 1: Fabricar la Carcasa

July 1, 2019

Esta es una continuación del artículo de la consola de juegos retro del mes pasado, donde aprendimos a montar una consola de juegos retro.

Yocto en el ODROID-C2: Usando Yocto con el Kernel 5.0

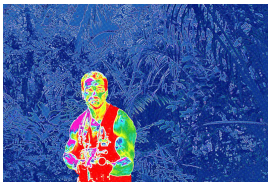
July 1, 2019

El Proyecto Yocto (YP) es un proyecto colaborativo de código abierto que ayuda a los desarrolladores a crear sistemas personalizados basados en Linux, independientemente de la arquitectura del hardware. Yocto no es una distribución de Linux integrada, sino que más bien crea una personalizada para ti.

Juegos Linux en ODROID: Juegos en ODROID-N2 - Escritorio y gl4es

July 1, 2019

El ODROID-N2 todavía es muy reciente, aunque ya tiene un par de meses.



Cámara Infrarroja Térmica ODROID-Go

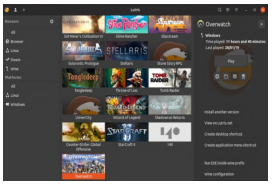
July 1, 2019

Este es un simple proyecto de cámara térmica por infrarrojos (IR) para el sistema portátil ESP32 del ODROID-GO. Permite guardar datos en una tarjeta SD, así como tener una interfaz Bluetooth básica para enviar datos de forma inalámbrica desde la cámara a un ordenador, tablet o teléfono móvil

RetroELEC para Emulation Station ODROID-XU4, RetroArch y Kodi en una Cómoda Imagen

July 1, 2019

Esta distribución de Linux "JeOS" es perfecta para ejecutar emuladores, ya que arranca a la velocidad de la luz y los recursos que utiliza son mínimos.



Lutris: Juegos en el ODROID-H2

July 1, 2019

En este artículo analizaremos brevemente lo que se necesitas para configurar un sistema de juego básico en un ODROID-H2

Vuélverte Ecológico con “Envi”: Un Sensor Combinado Ambiental Qwiic para tu Estimada Máquina de Juegos

July 1, 2019 By Dave Prochnow ODROID-GO, Mecaniqueo

"Caramba, que mal huele", podrías estar pensando. Aunque esta expresión es un poco imprecisa, pone de manifiesto que podrías estar experimentando un problema en la calidad del aire de tu entorno. La mayoría de los olores, tóxicos o de cualquier otro tipo, generalmente derivan de compuestos orgánicos volátiles (COV). De hecho, una medición clave para determinar la calidad del aire es la cuantificación de los compuestos orgánicos volátiles totales o TVOC.

Uno de los principales sensores para medir COV es el sensor de gas digital CCS811 de ultra bajo consumo de AMs AG en Austria. Además de proporcionar mediciones de COV, el CCS811 también puede proporcionar niveles equivalentes de CO2 (eCO2). Estos valores de eCO2 suelen ser una forma de COV que generan los humanos.

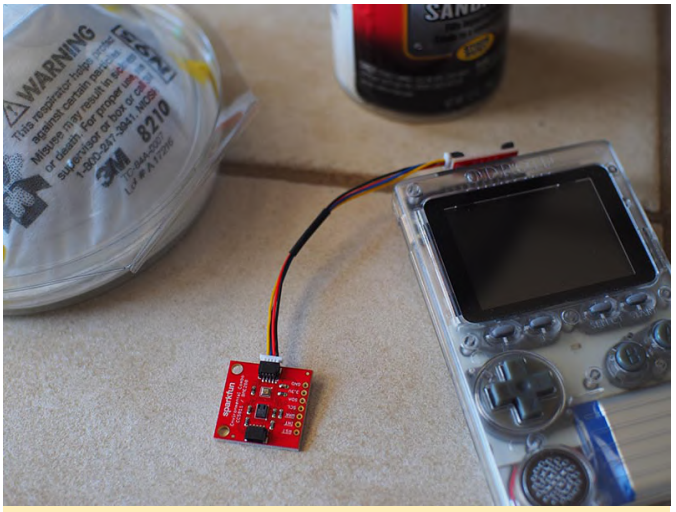


Figure 1 - Monitor your indoor environment with an ODROID-GO.

Estos niveles de calidad del aire son proporcionados como partes por billón (PPB) para TVOC y partes por millón (PPM) para eCO2. Como sensor independiente, los resultados del CCS811 están en bruto y son inexactos. Para mejorar los resultados de las

mediciones de la calidad del aire, las lecturas imprecisas del CCS811 se pueden compensar con datos de la temperatura actual del aire y de la humedad relativa.

Un sensor respetado que proporciona la humedad y la temperatura del ambiente es el sensor ambiental digital integrado BME280 de Bosch Sensortec. Integrar los resultados de BME280 en los cálculos para la calidad del aire CCS811 podría llegar a ser una tarea bastante engorrosa. Afortunadamente, SparkFun Electronics (SFE) ha combinado cuidadosamente los sensores CCS811 y BME280 en una sola placa independiente que puede monitorizar de manera fiable la calidad del aire ambiental.

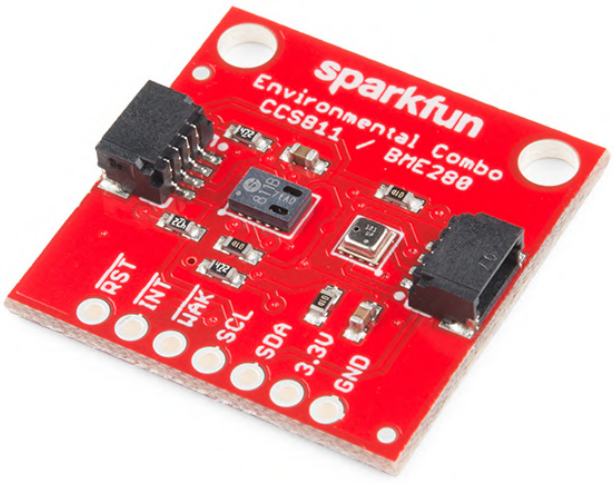


Figura 2 - El sensor combinado Ambiental SFE. Imagen cortesía de SparkFun Electronics.

Conocido como Qwiic Environmental Combo Breakout, SparkFun lo hizo más atractivo al incluir este paquete de monitorización de la calidad del aire en su ecosistema Qwiic I2C. Ahora, utilizando el proyecto del adaptador Qwiic que creamos en la edición de mayo de 2019 de ODROID Magazine, disponible aquí, https://magazine.odroid.com/article/go-and-be-qwiic-about-it/?ineedthispage=yes podemos controlar fácilmente la calidad del aire interior con una impresionante precisión a través de un simple sistema plug-and-GO. ¡Ah, el dulce olor del éxito!

Figura 3 – Un resultado de muestra del Qwiic Environmental Combo Breakout.

Componentes

SparkFun Environmental Combo Breakout – CSS811/BME280 (Qwiic) – SEN-14348 $35.95 Cable Qwiic – PRT-14427 $1.50

Paso a paso

1. Enchufa el adaptador Qwiic en tu conector ODROID-GO GPIO.

2. Conecta el cable Qwiic al adaptador Qwiic y conecta el otro extremo a la Environmental Combo Breakout Board.

3. Entra y carga este simple esquema de Arduino en tu dispositivo de juegos portátil ODROID-GO:

```
/*************************************************
*****************************
BME280Compensated.ino
Marshall Taylor @ SparkFun Electronics
April 4, 2017
https://github.com/sparkfun/CCS811_Air_Quality_Breakout
https://github.com/sparkfun/SparkFun_CCS811_Arduino_Library
This example uses a BME280 to gather environmental data that is then used
to compensate the CCS811.
Hardware Connections (Breakoutboard to Arduino):
3.3V to 3.3V pin
GND to GND pin
SDA to A4
SCL to A5
Resources:
Uses Wire.h for i2c operation
Hardware Connections:
```

```
Attach the Qwiic Environmental Combo to the Qwiic
Adapter board mounted on your ODROID-GO
Display on the ODROID-GO @ 320x240
Development environment specifics:
Arduino IDE 1.8.1
This code is released under the [MIT License]
(http://opensource.org/licenses/MIT).
Please review the LICENSE.md file included with
this example. If you have any questions
or concerns with licensing, please contact
techsupport@sparkfun.com.
Distributed as-is; no warranty is given.
************************************************
*************************/
#include
#include
#include
#include

#define CCS811_ADDR 0x5B //Default I2C Address
//#define CCS811_ADDR 0x5A //Alternate I2C Address

//Global sensor objects
CCS811 myCCS811(CCS811_ADDR);
BME280 myBME280;

ILI9341 lcd = ILI9341();

void setup()
{
Serial.begin(9600);
Serial.println();
Serial.println("Apply BME280 data to CCS811 for
compensation.");

Wire.begin();

//This begins the CCS811 sensor and prints error
status of .begin()
CCS811Core::status returnCode = myCCS811.begin();
if (returnCode != CCS811Core::SENSOR_SUCCESS)

Serial.println("Problem with CCS811");
printDriverError(returnCode);

else

Serial.println("CCS811 online");

//Initialize BME280
//For I2C, enable the following and disable the
SPI section
myBME280.settings.commInterface = I2C_MODE;
myBME280.settings.I2CAddress = 0x77;
myBME280.settings.runMode = 3; //Normal mode
myBME280.settings.tStandby = 0;
myBME280.settings.filter = 4;
myBME280.settings.tempOverSample = 5;
myBME280.settings.pressOverSample = 5;
myBME280.settings.humidOverSample = 5;

//Calling .begin() causes the settings to be
loaded
delay(10); //Make sure sensor had enough time to
turn on. BME280 requires 2ms to start up.
byte id = myBME280.begin(); //Returns ID of 0x60
if successful
if (id != 0x60)

Serial.println("Problem with BME280");

else

Serial.println("BME280 online");

// Setup LCD
lcd.begin();
lcd.setRotation(1);
lcd.fillScreen(BLACK);
lcd.setBrightness(255);
lcd.setTextFont(1);
lcd.setTextSize(2);
lcd.setCharCursor(10, 2);
lcd.setTextColor(LIGHTGREY);
lcd.println("ODROID-GO");
lcd.setCharCursor(5, 4);
lcd.println("Environmental Data");
lcd.setTextSize(2);

}
//-----------------------------------------------
---------------
void loop()
{
// Initiate the text cursor position
lcd.setCharCursor(1, 6);

//Check to see if data is available
if (myCCS811.dataAvailable())

//Calling this function updates the global tVOC
and eCO2 variables
myCCS811.readAlgorithmResults();
//printData fetches the values of tVOC and eCO2
printData();
```

```
float BMEtempC = myBME280.readTempC();
float BMEhumid = myBME280.readFloatHumidity();

Serial.print("Applying new values (deg C, %): ");
Serial.print(BMEtempC);
Serial.print(",");
Serial.println(BMEhumid);
Serial.println();

//This sends the temperature data to the CCS811
myCCS811.setEnvironmentalData(BMEhumid, BMEtempC);

else if (myCCS811.checkForStatusError())

Serial.println(myCCS811.getErrorRegister());
//Prints whatever CSS811 error flags are detected

delay(2000); //Wait for next reading
}

//-----------------------------------------------
---------------
void printData()
{
lcd.setTextColor(BLUE, BLACK);
Serial.print(" CO2[");
lcd.print ("CO2: [");
Serial.print(myCCS811.getCO2());
lcd.print (myCCS811.getCO2());
Serial.print("]ppm");
lcd.println ("] ppm");

Serial.print(" TVOC[");
lcd.print (" TVOC: [");
Serial.print(myCCS811.getTVOC());
lcd.print (myCCS811.getTVOC());
Serial.print("]ppb");
lcd.println ("] ppb");
lcd.println (" ");

lcd.setTextColor(RED, BLACK);
Serial.print(" temp[");
lcd.print (" Temp: ");
Serial.print(myBME280.readTempC(), 1);
lcd.print (myBME280.readTempC(), 1);
Serial.print("]C");
lcd.println ("C");

Serial.print(" pressure[");
lcd.print (" Press: ");
Serial.print(myBME280.readFloatPressure(), 2);
lcd.print (myBME280.readFloatPressure(), 2);
Serial.print("]Pa");
lcd.println ("Pa");
lcd.println (" ");

lcd.setTextColor(ORANGE, BLACK);
Serial.print(" humidity[");
lcd.print (" Humidity: ");
Serial.print(myBME280.readFloatHumidity(), 0);
lcd.print (myBME280.readFloatHumidity(), 0);
Serial.print("]%");
lcd.println ("%");

Serial.println();
}

//printDriverError decodes the CCS811Core::status
type and prints the
//type of error to the serial terminal.
//
//Save the return value of any function of type
CCS811Core::status, then pass
//to this function to see what the output was.
void printDriverError( CCS811Core::status
errorCode )
{
switch ( errorCode )

case CCS811Core::SENSOR_SUCCESS:
Serial.print("SUCCESS");
break;
case CCS811Core::SENSOR_ID_ERROR:
Serial.print("ID_ERROR");
break;
case CCS811Core::SENSOR_I2C_ERROR:
Serial.print("I2C_ERROR");
break;
case CCS811Core::SENSOR_INTERNAL_ERROR:
Serial.print("INTERNAL_ERROR");
break;
case CCS811Core::SENSOR_GENERIC_ERROR:
Serial.print("GENERIC_ERROR");
break;
default:
Serial.print("Unspecified error.");

}
```

4. Usa tu sensor de calidad del aire recién montado para descubrir quién ha estado usando el formaldehído sin tu permiso.

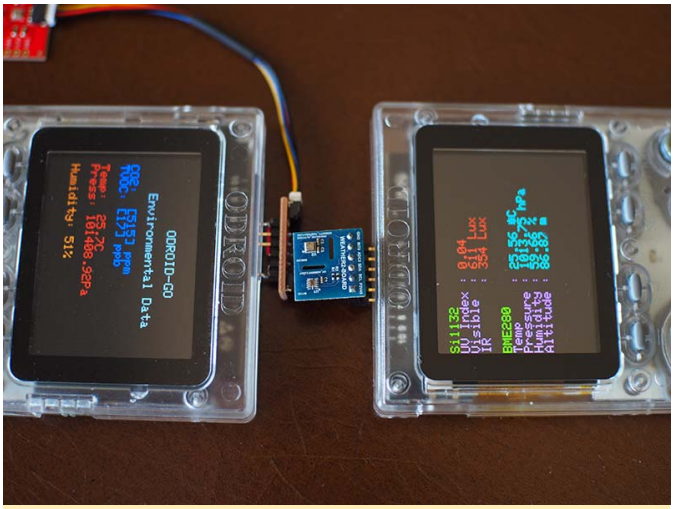


Figura 4 - Compara los resultados de las dos placas de sensores ambientales similares. ¿Aquí huele a gato encerrado?

NOTA: Para obtener un resultado de calidad del aire válido de la Environmental Combo board, debes llevar a cabo un ciclo de “quemado” del sensor CSS811 durante 48 horas y debe realizar un precalentamiento de unos 20 minutos antes de cada uso.

Monta un Almacenamiento de Acceso en red (NAS) de 6 bahías por tu Cuenta: Aprovecha el Potencial del ODROID-H2

July 1, 2019 By @tobetter Linux, Tutoriales

Este artículo te mostrará cómo montar un servidor NAS de 6 bahías con ODROID-H2. Hace unas cuantas semanas, se inició un pequeño proyecto para montar un servidor NAS de 6 bahías con el ODROID-H2 (Rev.B) que fue publicado en junio. Muchos usuarios han preguntado si el ODROID-H2 puede ejecutar más de 2 unidades usando la tarjeta de expansión PCI-e a SATA y un par de usuarios lo han intentado.

Carcasa

He intentado buscar una carcasa decente que tenga espacio suficiente para que cojan 6 unidades de disco de 3.5" y el ODROID-H2. Existen muchas carcasas NAS o HTPC que pueden ajustarse a la placa mini-ATX, pero el inconveniente lo teníamos en la altura que era demasiada baja o que las dimensiones no eran las suficientes para instalar seis unidades de disco duro. Las elegantes carcasas con 4-6 bahías con intercambio en caliente eran demasiado caras o solo admitían unidades de 2.5". Para disponen del espacio de instalación suficiente con un presupuesto decente, finalmente se optó por una carcasa NAS de **https://bit.ly/2X82Qtr**.

Dispone de 3 soportes que pueden albergar 2 unidades de 3.5" cada uno, y es posible ubicar un sistema de alimentación ATX estándar debajo de las bahías donde se ubican los discos. También tiene 4 montajes PCB en la parte inferior, pero sus dimensiones son de un tamaño ATX estándar en el que el ODROID- H2 no encaja, por lo que se hace necesario un adaptador para montar el ODROID-H2.

Figura 1

Figura 2

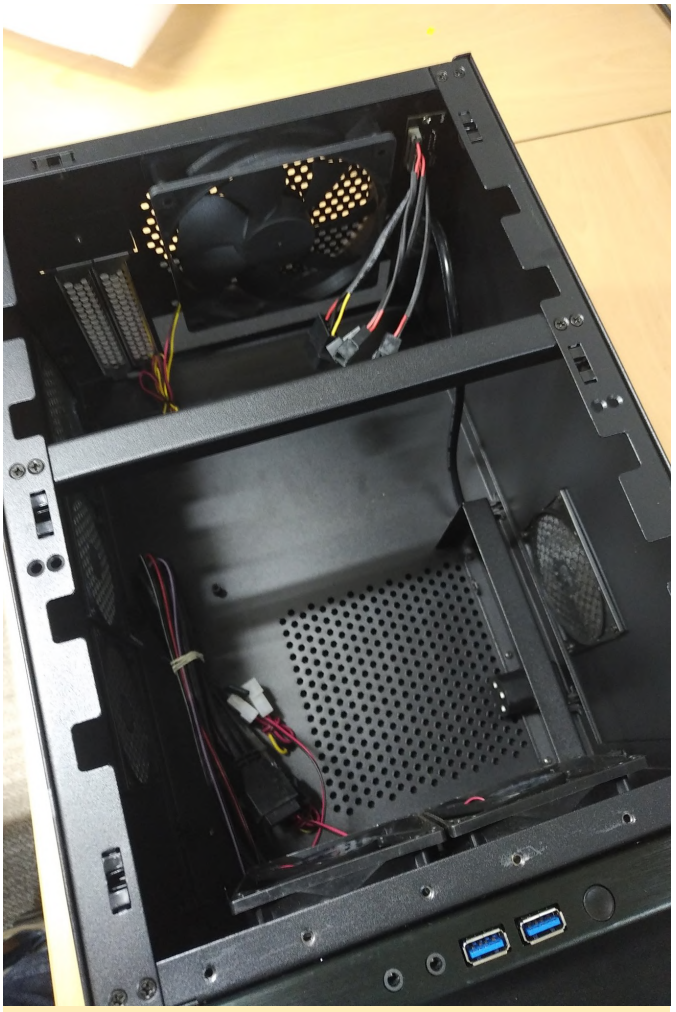


Figura 3

Montaje de la placa

Muchos de vosotros ya habréis notado que el tamaño físico del OROID-H2 no es compatible con ninguna especificación ATX, lo cual dificulta su montaje en una carcasa de PC estándar o en un HTPC. La carcasa que compramos también está diseñada para una mini tarjeta ATX que sigue siendo grande para el ODROID-H2; por lo tanto, se necesita una placa adaptadora para el montaje. Al principio, tuvimos en cuenta un panel acrílico como material para la placa adaptadora de montaje. Sin embargo, era demasiado frágil y tendría que realizar un nuevo corte cada vez que cambiase el diseño. Finalmente, decidí crearlo con una impresora 3D.

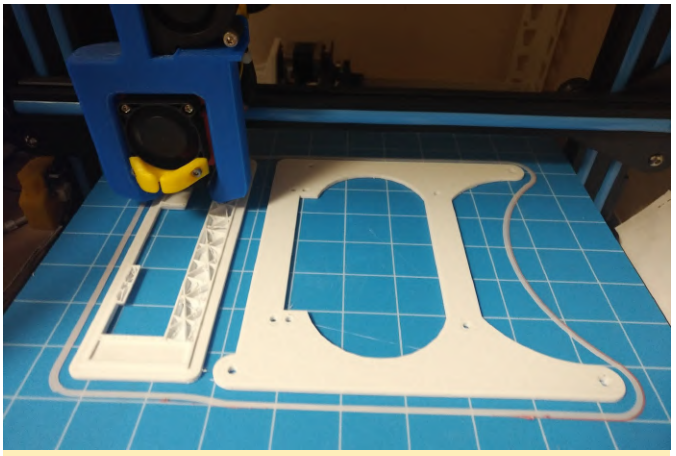


Figura 4

El panel se ha unido al ODROD-H2 mediante el soporte de PCB de 10 mm de altura. La altura de 10 mm es suficiente para colocar ODROID-H2 en la superficie superior; de lo contrario, el zócalo de memoria interferirá con el montaje. Dado que la altura del panel de montaje es de 3 mm, ODROID-H2 se monta 13 mm más alto que otras placas ATX y esto causa un problema con el panel posterior.

Figura 5

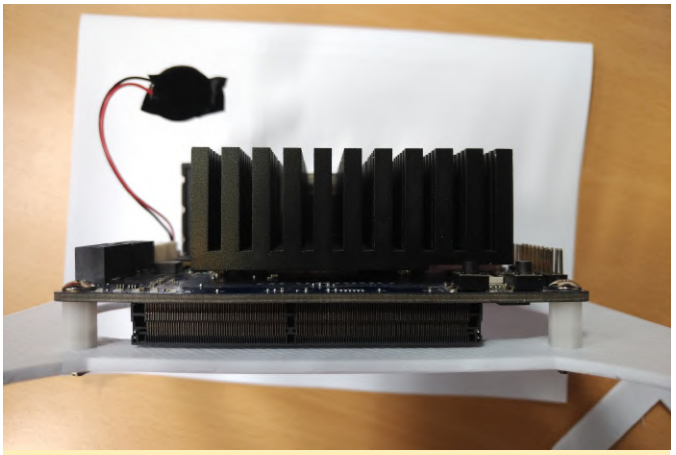


Figura 6

Como tenemos pensado conectar 6 unidades al ODROID-H2, la tarjeta de expansión SATA admite 4 conectores SATA adicionales en la ranura M.2 que son independientes de las 2 ranuras SATA del ODROID-H2. La conexión de los cables SATA al adaptador requiere de más espacio por debajo del ODROID-H2, el cable tipo ángulo recto ayuda a ahorrar espacio. Sin embargo, el cable aún requiere aproximadamente 25 mm de superficie debajo de ODROID-H2. Puesto que ya tenemos 13 mm del panel de montaje, tuvimos que ganar los 12 mm restantes de la carcasa. Afortunadamente, la altura de los agujeros de montaje en la PCB es de 12 mm, lo cual cumple con nuestros requisitos y de esta forma es posible colocar el ODROID-H2 de forma segura en la carcasa.

Figura 7

Figura 8

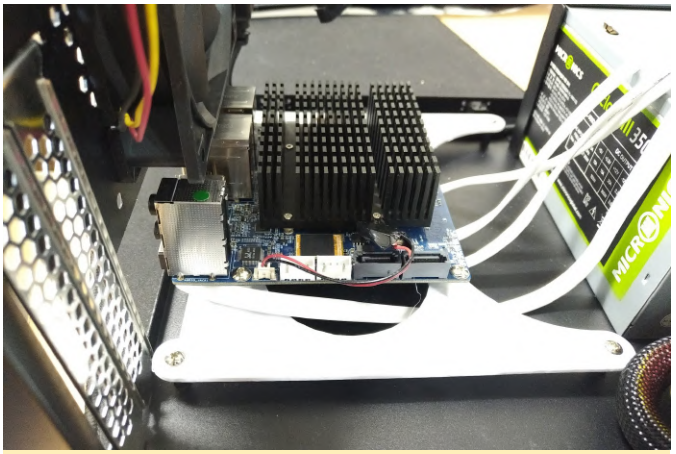


Figura 9

Después, tuve que abordar el problema del panel posterior. Todo es más alto que una placa de PC normal, ya que montamos el ODROID-H2 utilizando los agujeros de montaje ATX estándar e hicimos hueco para los cables SATA en la parte inferior. Esto significa que los usuarios de NAS tendrán que renunciar a un conector de audio. Los conectores Ethernet, por otro lado, pueden suponer un problema. Después de colocar la tapa posterior del panel impresa con el panel de montaje, observé que ésta no estaba completamente alineada con los conectores del ODROID-H2, aunque podía valer en nuestro caso.

Figura 10

Figura 11

Tengo otra imagen de cómo conectar los cables Ethernet al ODROID-H2 al mismo tiempo que se está montado en la carcasa. Apenas es posible conectar el cable al ODROID-H2 sin topar con algún obstáculo de la caja. Así que, creo que conectar el cable Ethernet a ODROID-H2 no supone mayor problema, sin embargo, una vez conectado no es posible quitarlo.

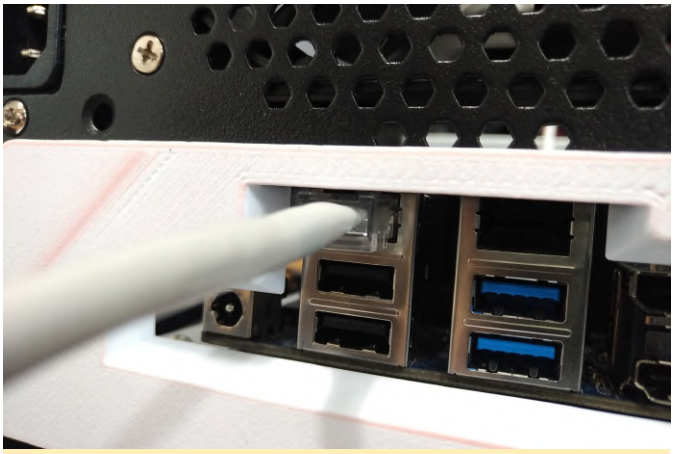


Figura 12

Tenemos intención de seguir adelante a pesar de haber tenido problemas con el panel posterior y el cable Ethernet que queda muy ajustado. Nuestro NAS funcionará 24/7, por lo tanto, el cable de red estará fijo u optar por montar la conexión del cable por el otro lado de la carcasa.

Fuente de Alimentación

Hemos comprado una fuente de alimentación ATX estándar de 350 vatios que encaja perfectamente dentro de la carcasa. Aunque 350W puede parecer demasiada potencia para ejecutar 6 HDD y el ODROID-H2, el coste de dicha potencia ATX no es excesivo, de modo que esta fuente de alimentación es aceptable. Aparte de la ratio de potencia de la fuente

de alimentación, lo que tenemos que tener en cuenta es si la fuente de alimentación tiene suficientes conectores SATA. Como la mayoría de las unidades de suministro de energía de menos de 500 vatios tienen hasta 4 conectores de alimentación SATA, se ha tenido que añadir un cable de extensión SATA adicional para las 2 unidades de disco duro restantes.

Figura 13

Suministrar energía al ODROID-H2 y a los 6 HDD es un reto, pero, es aún más importante la cuestión de que si el sistema podría controlarse mediante un único interruptor. Esto es debido a que la alimentación ATX estándar no solo se “enciende” cuando se dispare la señal de alimentación en el conector de alimentación ATX. Más bien, necesitaremos monitorizar otra señal que indique si el estado de la energía es bueno o no. Finalmente, se decidió utilizar la señal Power "on" con un módulo interruptor normal. Esta fue la forma más sencilla de controlar la fuente de alimentación con el interruptor. Se ha comprado un pequeño interruptor que encaja muy bien en la ranura de la tarjeta de expansión. El soporte de la ranura de la tarjeta debe cortarse con la longitud del interruptor.

Figura 14

El color del cable de la señal de encendido en la fuente de alimentación es verde. Esta señal debe estar cortocircuitada al suelo para encender la fuente de alimentación. Puesto que hemos decidimos usar un interruptor de palanca, el cable verde y el cable negro deben soldarse a éste.

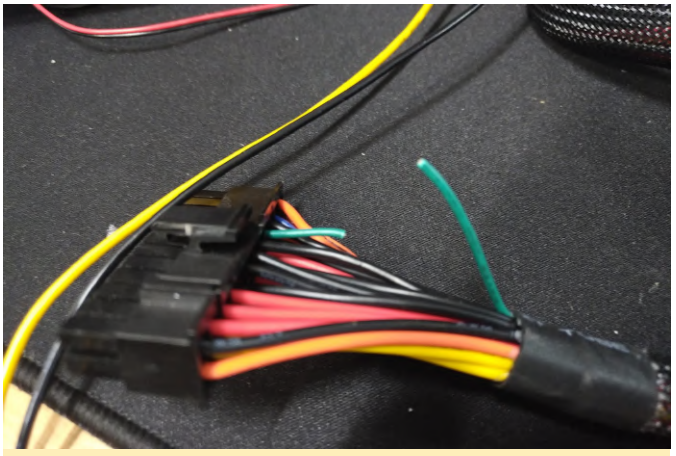


Figura 15

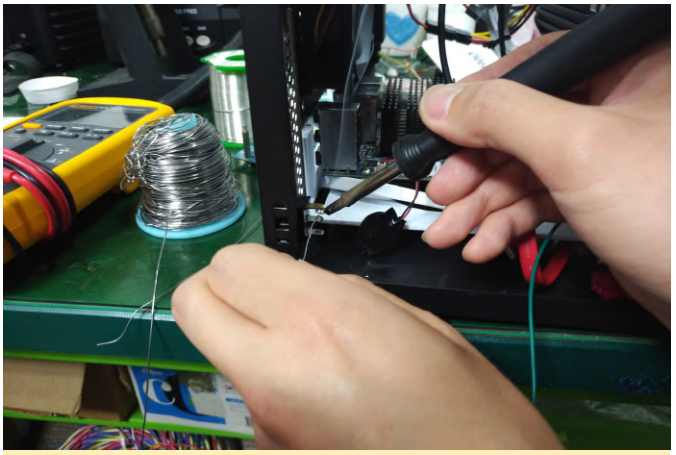


Figura 16

El ODROID-H2 no tiene un diagrama tipo PC de los indicadores o botones LED. Por lo tanto, para encender el LED de alimentación, el cable debe estar conectado a la salida de voltaje en el pin de

expansión. El DC 3.3V y los pines de puesta a tierra se pueden conectar como se muestra a continuación.

Figura 17

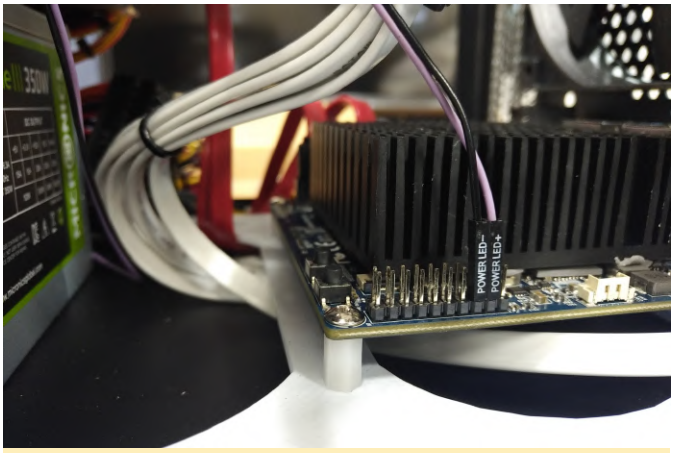


Figura 18

Cableado e Instalación del disco duro

Para este proyecto, fueron comprados y montados 6 HDD Western Digital 1TB sobre el soporte dentro de la carcasa.

Figura 19

La carcasa viene con 3 soportes por defecto. Es posible montar dos unidades de disco duro en cada soporte.

Figura 20

Figura 21

Se conectarán cuatro HDD con cables blancos a través de la tarjeta de expansión SATA M.2 y los otros dos con cables rojos se canalizarían usando las conexiones SATA integradas. Los cables de alimentación para las unidades de disco duro se conectarán mediante los conectores SATA de la PSU. Sin embargo, todavía necesitamos otro cable "Y" que convierta el conector único IDE en dos cables SATA.

Figura 22

Figura 23

Configurar BIOS

Con la configuración por defecto, el adaptador SATA en la ranura M.2 no funcionará. Para proceder correctamente, tenemos que recurrir a la configuración de la BIOS. Entra en la BIOS presionando la tecla “Eliminar” justo después de presionar el botón de encendido. Luego, deberías ver la pantalla que se muestra en la Figura 24.

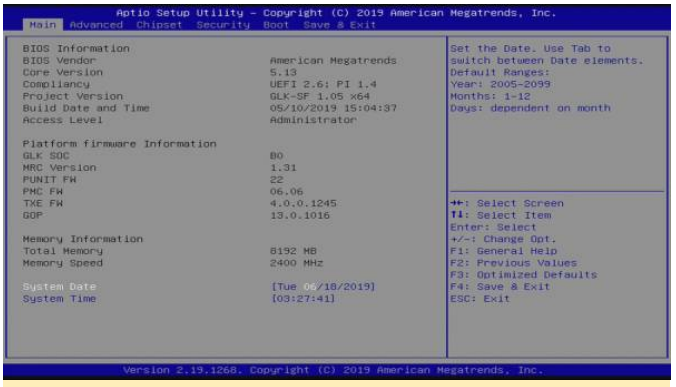


Figura 24

Tienes que configurar 2 opciones:

- Chipset - South Cluster Configuration - Miscellaneous Configuration - State After G3, cambiar a “S0 State” para que se encienda automáticamente cuando se conecte la alimentación.
- Chipset - South Cluster Configuration - PCI Express Configuration - PCI Express Clock Gating, cambiar a “Disabled” para hacer que el adaptador M.2 a SATA funcione correctamente.

Muévete con las teclas de flechas del teclado y configura cada opción.

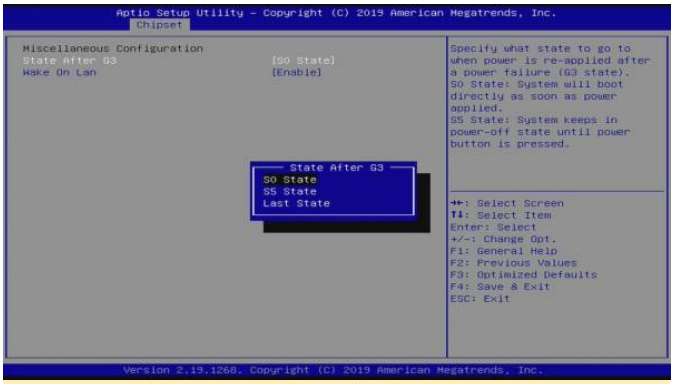


Figura 25

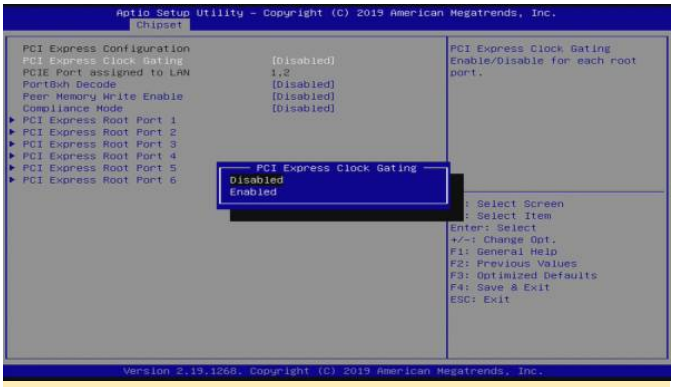


Figura 26

Instalación de Ubuntu

Para el sistema operativo host, hemos decidido instalar Ubuntu 19.04. Descargué la última imagen del sistema operativo de la página de descarga de Ubuntu (**https://ubuntu.com/download**), y grabé esa imagen en una memoria USB con Etcher. Etcher es una herramienta para grabar imágenes multiplataforma. Conecta el dispositivo de memoria USB al H2 y enciéndalo. Luego presiona F7 para entrar en la pantalla de selección de soportes de arranque. Si se muestra la pantalla, selecciona el dispositivo de arranque USB que hemos conectado.

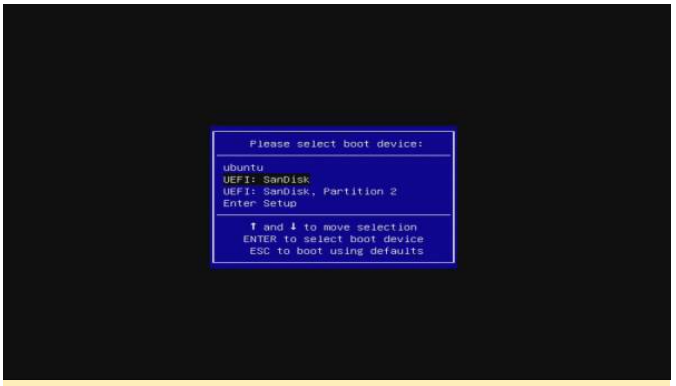


Figura 27

Continúa instalando Ubuntu de acuerdo a tus preferencias. Solo ten en cuenta el seleccionar un

disco duro cuando selecciones un soporte root durante la instalación. Mantener el sistema actualizado es altamente recomendable para cualquier uso de la aplicación, en la mayoría de los casos.

Para comprobar si las unidades de disco duro se instalaron correctamente, puedes echar un vistazo al resultado de dmesg:

```
$ dmesg | grep scsi
[    2.067831] scsi host0: ahci
[    2.068684] scsi host1: ahci
[    2.080796] scsi host2: ahci
[    2.080964] scsi host3: ahci
[    2.084816] scsi host4: ahci
[    2.084976] scsi host5: ahci
[    2.548364] scsi 0:0:0:0: Direct-Access     ATA
WDC WD10EFRX-68F 0A82 PQ: 0 ANSI: 5
[    2.549029] sd 0:0:0:0: Attached scsi generic
sg0 type 0
[    2.549321] scsi 1:0:0:0: Direct-Access     ATA
WDC WD10EFRX-68F 0A82 PQ: 0 ANSI: 5
[    2.549627] sd 1:0:0:0: Attached scsi generic
sg1 type 0
[    2.563013] scsi 2:0:0:0: Direct-Access     ATA
WDC WD10EFRX-68F 0A82 PQ: 0 ANSI: 5
[    2.563292] sd 2:0:0:0: Attached scsi generic
sg2 type 0
[    2.563497] scsi 3:0:0:0: Direct-Access     ATA
WDC WD10EFRX-68F 0A82 PQ: 0 ANSI: 5
[    2.563693] sd 3:0:0:0: Attached scsi generic
sg3 type 0
[    2.563875] scsi 4:0:0:0: Direct-Access     ATA
WDC WD10EFRX-68F 0A82 PQ: 0 ANSI: 5
[    2.564070] sd 4:0:0:0: Attached scsi generic
sg4 type 0
[    2.564268] scsi 5:0:0:0: Direct-Access     ATA
WDC WD10EFRX-68F 0A82 PQ: 0 ANSI: 5
[    2.564444] sd 5:0:0:0: Attached scsi generic
sg5 type 0
```

Esto muestra que esas 6 unidades han sido reconocidas. Si las tuyas no, debes comprobar la conexión del hardware o la configuración de la BIOS.

Particionar los discos duros

Comprueba los discos duros instalados utilizando los siguientes comandos.

```
$ sudo fdisk -l | grep sd
Disk /dev/sda: 931.5 GiB, 1000204886016 bytes,
1953525168 sectors
Disk /dev/sdb: 931.5 GiB, 1000204886016 bytes,
1953525168 sectors
Disk /dev/sdc: 931.5 GiB, 1000204886016 bytes,
1953525168 sectors
Disk /dev/sdd: 931.5 GiB, 1000204886016 bytes,
1953525168 sectors
Disk /dev/sde: 931.5 GiB, 1000204886016 bytes,
1953525168 sectors
Disk /dev/sdf: 931.5 GiB, 1000204886016 bytes,
1953525168 sectors
```

A continuación, crea una partición utilizando la herramienta parted. Puede usar fdisk para peticionar los discos duros, para poder usar más de 2 TB de disco duro debes crear una tabla de particiones GPT. Particiona los discos respectivamente en base a los siguientes procedimientos:

```
$ sudo parted /dev/sda
GNU Parted 3.2
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list
of commands.
(parted) mklabel gpt
(parted) print free
Model: ATA WDC WD10EFRX-68F (scsi)
Disk /dev/sda: 1000GB
Sector size (logical/physical): 512B/4096B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name
Flags
        17.4kB  1000GB  1000GB  Free Space

(parted) mkpart primary 1M 1000GB
(parted) p
Model: ATA WDC WD10EFRX-68F (scsi)
Disk /dev/sda: 1000GB
Sector size (logical/physical): 512B/4096B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name
Flags
 1      1049kB  1000GB  1000GB
primary

(parted) q
Information: You may need to update /etc/fstab.
Check the changes applied.
```

```
$ lsblk | grep sd
sda            8:0    0 931.5G  0 disk
└─sda1         8:1    0 931.5G  0 part
sdb            8:16   0 931.5G  0 disk
└─sdb1         8:17   0 931.5G  0 part
sdc            8:32   1 931.5G  0 disk
└─sdc1         8:33   1 931.5G  0 part
sdd            8:48   1 931.5G  0 disk
└─sdd1         8:49   1 931.5G  0 part
sde            8:64   1 931.5G  0 disk
└─sde1         8:65   1 931.5G  0 part
sdf            8:80   1 931.5G  0 disk
└─sdf1         8:81   1 931.5G  0 part
```

Podemos ver que cada disco duro tiene una sola partición. Después de particionar todas las unidades, es el momento de configurar RAID 6 en nuestro NAS.

Configurar RAID 6

Hay dos razones principales para usar el nivel 6 de RAID:

- Es más robusto que RAID 5, porque usa un disco más para la paridad.
- No perderemos los datos incluso si fallan 2 discos. Podemos reconstruir el sistema después de reemplazar el disco que ha fallado.

Sin embargo, no encontramos con una cierta sobrecarga: la doble paridad puede verificar su estabilidad, pero viene acompañado de un rendimiento más deficiente en términos de escritura. Se requiere un mínimo de 4 discos para montar un RAID 6. Puesto que tenemos 6 discos duros con una capacidad de 1 TB cada uno, podemos montarlos usando RAID 6 y tendremos una capacidad de 4 TB que podremos usar.

Para configurar RAID sin un controlador RAID físico en un sistema Linux, tenemos que usar la herramienta mdadm. La puedes localizar en el administrador de paquetes de cualquier distribución de Linux:

```
$ sudo apt install mdadm
```

Ya sabemos que esas 6 unidades se asignan como /dev/sda a /dev/sdf. Crea una matriz usando el siguiente comando:

```
$ sudo mdadm --create /dev/md0 --level=6 --raid-devices=6 /dev/sda1 /dev/sdb1 /dev/sdc1 /dev/sdd1 /dev/sde1 /dev/sdf1
mdadm: Defaulting to version 1.2 metadata
mdadm: array /dev/md0 started.
```

Se creará el archivo de dispositivo /dev/md0. Debe usar ese dispositivo como una única partición de disco duro, de modo que, si deseas montar esa matriz en un directorio, puedes simplemente usar ese archivo de dispositivo. Formatea esa partición y monta en / media/storage:

```
$ sudo mkfs.ext4 /dev/md0
mke2fs 1.44.6 (5-Mar-2019)
Creating filesystem with 976628736 4k blocks and 244162560 inodes
Filesystem UUID: 100a470d-96f1-47d2-8cf0-a211c010e8b9
Superblock backups stored on blocks:
  32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
  4096000, 7962624, 11239424, 20480000, 23887872, 71663616, 78675968,
  102400000, 214990848, 512000000, 550731776, 644972544

Allocating group tables: done
Writing inode tables: done
Creating journal (262144 blocks): done
Writing superblocks and filesystem accounting information: done

$ sudo mkdir /media/storage
$ sudo mount /dev/md0 /media/storage/
```

Comprueba si está montado correctamente.

```
$ cat /proc/mounts | grep md0
/dev/md0 /media/storage ext4 rw,relatime,stripe=512 0 0
```

You also can see the RAID configurations:

```
$ sudo mdadm --detail /dev/md0
/dev/md0:
           Version : 1.2
     Creation Time : Mon Jun 17 18:08:26 2019
        Raid Level : raid6
        Array Size : 3906514944 (3725.54 GiB 4000.27 GB)
     Used Dev Size : 976628736 (931.39 GiB 1000.07 GB)
      Raid Devices : 6
     Total Devices : 6
```

```
       Persistence : Superblock is persistent

     Intent Bitmap : Internal

       Update Time : Mon Jun 17 18:27:08 2019
             State : active, resyncing
    Active Devices : 6
   Working Devices : 6
    Failed Devices : 0
     Spare Devices : 0

            Layout : left-symmetric
        Chunk Size : 512K

Consistency Policy : bitmap

     Resync Status : 8% complete

              Name : ODROID-H2:0  (local to host
ODROID-H2)
              UUID :
d4759dbb:65fd2b07:d5f4f9c3:0fba55cc
            Events : 253

    Number   Major   Minor   RaidDevice State
       0       8        1        0      active
sync   /dev/sda1
       1       8       17        1      active
sync   /dev/sdb1
       2       8       33        2      active
sync   /dev/sdc1
       3       8       49        3      active
sync   /dev/sdd1
       4       8       65        4      active
sync   /dev/sde1
       5       8       81        5      active
sync   /dev/sdf1
```

Puesto que esta matriz RAID se acaba de crear, debemos realizar un proceso de re-sincronización para que se sincronice con los otros dispositivos. Puedes ver el estado del proceso de resincronización con el siguiente comando:

```
$ cat /proc/mdstat
Personalities : [linear] [multipath] [raid0]
[raid1] [raid6] [raid5] [raid4] [raid10]
md0 : active raid6 sdf1[5] sde1[4] sdd1[3] sdc1[2]
sdb1[1] sda1[0]
      3906514944 blocks super 1.2 level 6, 512k
chunk, algorithm 2 [6/6] [UUUUUU]
      [=>...................]  resync =  9.6%
(93909272/976628736) finish=120.2min
speed=122360K/sec
      bitmap: 8/8 pages [32KB], 65536KB chunk

unused devices:
```

Una vez que se complete el proceso rsync, podrás ver un mensaje a través de dmesg:

```
$ dmesg | grep resync
[  199.311304] md: resync of RAID array md0
[10093.988694] md: md0: resync done.
```

También configuraría el entorno el sistema agregando una entrada a /etc/fstab y configurando el servidor SAMBA o SFTP para compartir tus datos utilizando los discos duros RAID-ed. Existen muchas guías de administración para usos adicionales del sistema RAID integrado, que incluye añadir unidades de repuesto o tratar dispositivos con fallos, pero esta guía no se detendrá en todos estos detalles.

Pruebas de rendimiento

Ejecuté iozone3 para evaluar el rendimiento del H2 con RAID nivel 6 con 6 discos duros, después de asegurarme de que el proceso de sincronización se había completado:

```
$ sudo iozone -e -I -a -s 100M -r 4k -r 16384k -i
0 -i 1 -i 2
  Iozone: Performance Test of File I/O
          Version $Revision: 3.429 $
    Compiled for 64 bit mode.
    Build: linux-AMD64

  Contributors:William Norcott, Don Capps, Isom
Crawford, Kirby Collins
               Al Slater, Scott Rhine, Mike
Wisner, Ken Goss
               Steve Landherr, Brad Smith, Mark
Kelly, Dr. Alain CYR,
               Randy Dunlap, Mark Montague, Dan
Million, Gavin Brebner,
               Jean-Marc Zucconi, Jeff Blomberg,
Benny Halevy, Dave Boone,
               Erik Habbinga, Kris Strecker,
Walter Wong, Joshua Root,
               Fabrice Bacchella, Zhenghua Xue,
Qin Li, Darren Sawyer,
               Vangel Bojaxhi, Ben England,
Vikentsi Lapa.
```

```
  Run began: Tue Jun 18 10:03:49 2019

  Include fsync in write timing
  O_DIRECT feature enabled
  Auto Mode
  File size set to 102400 kB
  Record Size 4 kB
  Record Size 16384 kB
  Command line used: iozone -e -I -a -s 100M -r 4k
-r 16384k -i 0 -i 1 -i 2
  Output is in kBytes/sec
  Time Resolution = 0.000001 seconds.
  Processor cache size set to 1024 kBytes.
  Processor cache line size set to 32 bytes.
  File stride size set to 17 * record size.

random    random     bkwd    record    stride
             kB  reclen    write  rewrite    read
reread    read     write     read   rewrite
read   fwrite frewrite    fread  freread
         102400       4     9241    14364
27027    29648    15326     4404
         102400   16384   208414   209245
521260   669540   731096   177565

iozone test complete.
```

Puesto que los 6 discos duros están conectados entre sí, su rendimiento es mucho mejor que el de los resultados con el único 1 disco duro.

	One Disk	RAID 6 with 6 Disks
4K R/Read	979	27027
4K R/Write	1937	9241
16M S/Read	147202 (147 MB/s)	521260 (521 MB/s)
16M S/Write	132406 (132 MB/s)	208414 (208 MB/s)

Figura 28 - Pruebas de rendimiento

RAID 6 usa doble paridad, aunque los 6 discos duros funcionan literalmente como una especie de multiprocesamiento de la CPU, de modo que la velocidad general aumenta considerablemente. Si utilizas RAID nivel 5, el rendimiento sería un poco mejor que este, ya que RAID 5 utiliza una paridad única. Por lo tanto, no hay razón para no usar RAID 6 cuando se tiene más de 4 discos duros, especialmente, dado que el H2 es un potente hardware host para usarse a modo de NAS a un precio razonable, con 2 puertos LAN Gigabit como ventaja adicional para la conectividad. Nos ocuparemos de la instalación del software en el siguiente artículo.

Referencias

https://forum.odroid.com/viewtopic.php?f=172&t=35309https://www.tecmint.com/create-raid-6-in-linux/

https://askubuntu.com/questions/350266/how-can-i-create-a-raid-array-with-2tb-disks

El Punto G: Tu Destino para todas las Cuestiones Relacionadas con Juegos Android

July 1, 2019 By Dave Prochnow Juegos

Los próximos meses del verano podrían ser muy emocionantes para los jugadores de Android. He incluido en mi calendario Google Stadia, E3, y algunos lanzamientos de videojuegos muy esperados y de gran renombre. Uno de estos grandes lanzamientos de juegos es Elder Scrolls: Blades. He estado elogiando este título durante los últimos dos meses, así que era hora de hacer una prueba rápida con la versión de acceso temprana.

Figura 1 - Necesitarás una conexión permanente a Internet para jugar a la versión de acceso temprana

Esta versión temprana del juego requiere que tengas una conexión a Internet. A pesar de que el juego supone una descarga bastante razonable de 120Mb, la puesta en marcha inicial del juego requerirá otra descarga de 880Mb junto con una actualización

adicional de 380Mb. Además, cada vez que inicies una sesión de juego, hay un retardo prolongado mientras son verificadas las versiones de las actualizaciones por los servidores de Bethesda.

¿Todo este acceso online hace que Elder Scrolls: Blades realmente merezca la pena? Pues depende desde donde se mire. Por supuesto, muchos de los efectos visuales están magníficamente renderizados y son bastante animados, pero el juego real deja mucho que desear.

Basado en un formato interactivo de exploración/navegación punto-toque-movimiento: la acción de combate en Elder Scrolls: Blades presenta un modo de ataque punto-toque-apuñalamiento un tanto chapucero y poco desarrollado. Junto con unos diálogos básicos que manifiestan los diversos enemigos que vas encontrando a lo largo del juego, esta versión de temprana parece más bien un esfuerzo a medias que ni siquiera está listo para una audiencia masiva.

Figura 2: No nos encontramos con demasiadas amenazas: ocho golpes con mi dedo/espada serán más que suficientes.

Ailment

En este título de acción gratuita de BeardyBird Games, eres un astronauta que despierta en una nave espacial repleta de problemas y habitada por enemigos. Hay una serie de preguntas que debes responder. ¿Por qué están tratando de matarte? ¿Quiénes son? Y, lo más importante, ¿dónde estás? Ailment es en parte un juego de resolución de rompecabezas y en parte un juego épico de combate de disparos. El resultado es un juego que incluye referencias ocultas a la clásica ciencia ficción y el objetivo final de descubrir qué sucedió realmente en la nave espacial.

```
https://www.youtube.com/watch?v=T2inRVXYRNU
```

Figura 3 – Hablarás mucho contigo mismo en Ailment. Imagen cortesía de BeardyBird Games.

Despotism 3K

Con una única versión con el concepto de juego Sim IV, esta versión de Konfa Games está mucho más trabajada y es más divertida (ny). Reducido a un nuevo precio de compra de 3.49$, Despotism 3K nos presenta a un experto en Inteligencia Artificial (AI) que controla a una multitud de personas subordinadas que hacen ofertas. Y muchas de estas tareas rutinarias pueden ser fatales. Aunque no le tengas lástima de la gente o perderás en este juego.

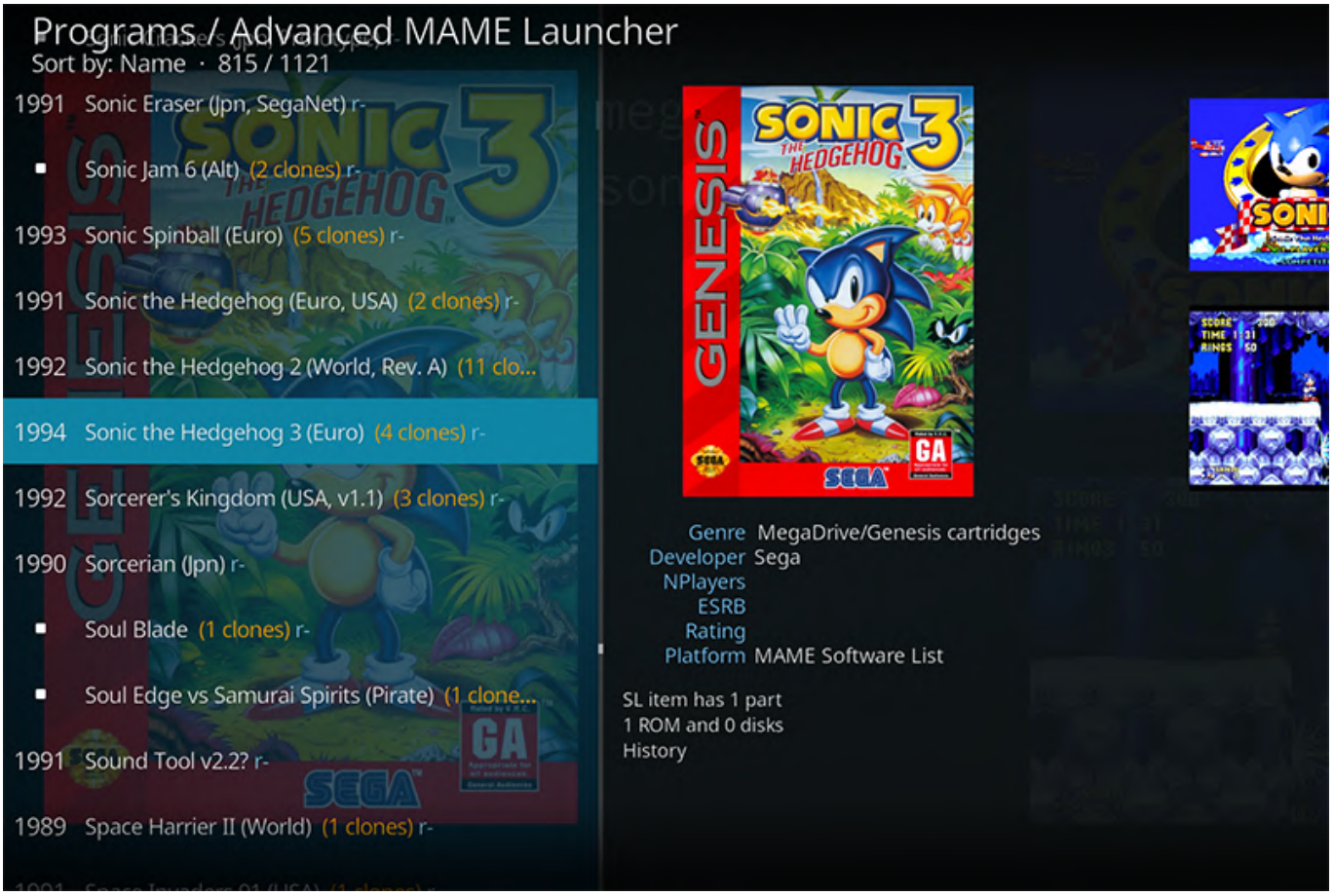
```
https://www.youtube.com/watch?v=OT3QJZ_mrJo
```

Selección Veraniega de los Mejores Juegos Android

Asphalt 9: Legends – GRATUITO Crashlands – $4.99 Riptide GTP Series – $2.99 Shadowgun Legends – GRATUITO Elder Scrolls: Blades – GRATUITO. POR FAVOR, pruébalo antes de comprarlo.

Kodi y Mame Avanzado en ODROID-XU4 - Parte 2

July 1, 2019 By David Bellot Juegos



Esta es la continuación de una guía para configurar Kodi con Mame, que detalla cómo instalar el joystick. En el mejor de los casos, jugar con MAME requiere de un buen joystick. Aquí tienes dos ejemplos de joystick que he creado yo mismo. Es un buen ejercicio de carpintería, pintura, diseño y electrónica y un juego divertido para la familia. Los he montado con tablas que recogí de un sitio de construcción cercano. Bueno para el medio ambiente por reciclar cosas.

Figura 1 - Joystick hecho a medida para Kodi con Mame

Figura 2 - Joystick hecho a medida para Kodi con Mame

1. Instala el software para soportar y calibrar el joystick. Los joysticks Arcade son fáciles de montar o se pueden comprar en Internet. Funcionan muy bien y son ideales para jugar con los juegos arcade de MAME.

```
$ sudo apt install joystick jstest-gtk
```

Calibrar el joystick

Como la mayoría de los juegos de Mame requieren un sencillo joystick con botones, la calibración será muy

simple. Puedes usar jstest-gtk, pero como ya hemos instalado Kodi, simplemente realizaremos la calibración desde la línea de comandos.

Con un joystick de clic, la única calibración es asociar los botones (dentro del joystick para las direcciones y los botones de disparo/selección), con su respectiva dirección o función. La calibración estará disponible para todo el sistema y, por lo tanto, será utilizada por mame.

Si puedes conectar tu joystick a tu máquina Linux, recomiendo usar un pequeño programa llamado jstest-gtk. Es una simple GUI con la que puedes verificar la dirección correcta de tu joystick. En mi caso, uso un joystick compatible con DragonRise (el de la imagen), con 4 conectores para arriba, abajo, izquierda y derecha. Existen algunos problemas que solucionaremos con la calibración. En primer lugar, se invierte la izquierda-derecha y arriba-abajo y luego el eje arriba-abajo está al revés. Así que para abreviar: ¡arriba está a la derecha, abajo está a la izquierda, a la derecha está abajo y a la izquierda está arriba! Puedo verlo en la interfaz de jstest-gtk.

Otra opción (desde Kodi 17) es configurar tu joystick directamente desde Kodi. Tienes un tutorial disponible en https://kodi.wiki/view/HOW-TO:Configure_controllers. Como hay muchos modelos de joystick, me es imposible cubrir todas las configuraciones posibles, así que por favor contribuye y añade la tuya a esta guía.

Si quieres jugar con la configuración del joystick y asignar varios comandos a cada botón o cambiar las direcciones, debes crear un archivo de configuración. Lo llamaremos myjoyremap.cfg. En mi caso, uso el siguiente archivo, pero el tuyo puede ser muy distinto según lo que quieras conseguir y el modelo del joystick:

```
< ?xml version="1.0"? >
< mameconfig version="10" >
< system name="default" >
< input >
< port type="P1_JOYSTICK_UP" > < newseq
type="standard" > JOYCODE_1_XAXIS_RIGHT_SWITCH <
/newseq > < /port >
< port type="P1_JOYSTICK_DOWN" > < newseq
type="standard" > JOYCODE_1_XAXIS_LEFT_SWITCH <
/newseq > < /port >
< port type="P1_JOYSTICK_LEFT" > < newseq
type="standard" > JOYCODE_1_YAXIS_UP_SWITCH <
/newseq > < /port >
< port type="P1_JOYSTICK_RIGHT" > < newseq
type="standard" > JOYCODE_1_YAXIS_DOWN_SWITCH <
/newseq > < /port >

< port type="P1_BUTTON1" > < newseq
type="standard" > JOYCODE_1_BUTTON1 < /newseq > <
/port >
< port type="P1_BUTTON2" > < newseq
type="standard" > JOYCODE_1_BUTTON3 < /newseq > <
/port >
< port type="P1_BUTTON3" > < newseq
type="standard" > JOYCODE_1_BUTTON5 < /newseq > <
/port >
< port type="P1_BUTTON4" > < newseq
type="standard" > JOYCODE_1_BUTTON7 < /newseq > <
/port >

< port type="P1_BUTTON5" > < newseq
type="standard" > JOYCODE_1_BUTTON2 < /newseq > <
/port >
< port type="P1_BUTTON6" > < newseq
type="standard" > JOYCODE_1_BUTTON4 < /newseq > <
/port >
< port type="P1_BUTTON7" > < newseq
type="standard" > JOYCODE_1_BUTTON6 < /newseq > <
/port >

< port type="P1_BUTTON8" > < newseq
type="standard" > NONE < /newseq > < /port >
< port type="P1_BUTTON9" > < newseq
type="standard" > NONE < /newseq > < /port >
< port type="P1_BUTTON10" > < newseq
type="standard" > NONE < /newseq > < /port >
< port type="P1_BUTTON11" > < newseq
type="standard" > NONE < /newseq > < /port >
< port type="P1_BUTTON12" > < newseq
type="standard" > NONE < /newseq > < /port >

< port type="P1_START" > < newseq type="standard"
> JOYCODE_1_BUTTON9 < /newseq > < /port >
< port type="P1_SELECT" > < newseq type="standard"
> JOYCODE_1_BUTTON10 < /newseq > < /port >
< port type="COIN1" > < newseq type="standard" >
JOYCODE_1_BUTTON8 < /newseq > < /port >
< port type="START1" > < newseq type="standard" >
KEYCODE_1 OR JOYCODE_1_BUTTON9 < /newseq > < /port
>

< port type="P1_PEDAL" > < newseq type="standard"
```

```
> NONE < /newseq > < newseq type="increment" >
NONE < /newseq >
< /port > < port type="P1_PEDAL2" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < /port >
< port type="P1_PEDAL3" > < newseq
type="increment" > NONE < /newseq > < /port >
< port type="P1_PADDLE" > < newseq type="standard"
> NONE < /newseq > < newseq type="increment" >
NONE < /newseq > < newseq type="decrement" > NONE
< /newseq > < /port >
< port type="P1_PADDLE_V" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < newseq
type="decrement" > NONE < /newseq > < /port >
< port type="P1_POSITIONAL" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < newseq
type="decrement" > NONE < /newseq > < /port >
< port type="P1_POSITIONAL_V" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < newseq
type="decrement" > NONE < /newseq > < /port >
< port type="P1_DIAL" > < newseq type="standard" >
NONE < /newseq > < newseq type="increment" > NONE
< /newseq > < newseq type="decrement" > NONE <
/newseq > < /port >
< port type="P1_DIAL_V" > < newseq type="standard"
> NONE < /newseq > < newseq type="increment" >
NONE < /newseq > < newseq type="decrement" > NONE
< /newseq > < /port >
< port type="P1_TRACKBALL_X" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < newseq
type="decrement" > NONE < /newseq > < /port >
< port type="P1_TRACKBALL_Y" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < newseq
type="decrement" > NONE < /newseq > < /port >
< port type="P1_AD_STICK_X" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < newseq
type="decrement" > NONE < /newseq > < /port >
< port type="P1_AD_STICK_Y" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < newseq
type="decrement" > NONE < /newseq > < /port >
< port type="P1_AD_STICK_Z" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < newseq
type="decrement" > NONE < /newseq > < /port >
< port type="P1_LIGHTGUN_X" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < newseq
type="decrement" > NONE < /newseq > < /port >
< port type="P1_LIGHTGUN_Y" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < newseq
type="decrement" > NONE < /newseq > < /port >

< port type="P2_JOYSTICK_UP" > < newseq
type="standard" > JOYCODE_2_XAXIS_RIGHT_SWITCH <
/newseq > < /port >
< port type="P2_JOYSTICK_DOWN" > < newseq
type="standard" > JOYCODE_2_XAXIS_LEFT_SWITCH <
/newseq > < /port >
< port type="P2_JOYSTICK_LEFT" > < newseq
type="standard" > JOYCODE_2_YAXIS_UP_SWITCH <
/newseq > < /port >
< port type="P2_JOYSTICK_RIGHT" > < newseq
type="standard" > JOYCODE_2_YAXIS_DOWN_SWITCH <
/newseq > < /port >

< port type="P2_BUTTON1" > < newseq
type="standard" > JOYCODE_2_BUTTON1 < /newseq > <
/port >
< port type="P2_BUTTON2" > < newseq
type="standard" > JOYCODE_2_BUTTON3 < /newseq > <
/port >
< port type="P2_BUTTON3" > < newseq
type="standard" > JOYCODE_2_BUTTON5 < /newseq > <
/port >
< port type="P2_BUTTON4" > < newseq
type="standard" > JOYCODE_2_BUTTON7 < /newseq > <
/port >

< port type="P2_BUTTON5" > < newseq
type="standard" > JOYCODE_2_BUTTON2 < /newseq > <
/port >
< port type="P2_BUTTON6" > < newseq
type="standard" > JOYCODE_2_BUTTON4 < /newseq > <
/port >
< port type="P2_BUTTON7" > < newseq
type="standard" > JOYCODE_2_BUTTON6 < /newseq > <
/port >

< port type="P2_BUTTON8" > < newseq
type="standard" > NONE < /newseq > < /port >
< port type="P2_BUTTON9" > < newseq
type="standard" > NONE < /newseq > < /port >
< port type="P2_BUTTON10" > < newseq
type="standard" > NONE < /newseq > < /port >
< port type="P2_BUTTON11" > < newseq
type="standard" > NONE < /newseq > < /port >
< port type="P2_BUTTON12" > < newseq
type="standard" > NONE < /newseq > < /port >
```

```
< port type="P2_START" > < newseq type="standard"
> JOYCODE_2_BUTTON9 < /newseq > < /port >
< port type="P2_SELECT" > < newseq type="standard"
> JOYCODE_2_BUTTON10 < /newseq > < /port >
< port type="COIN2" > < newseq type="standard" >
JOYCODE_2_BUTTON8 < /newseq > < /port >
< port type="START2" > < newseq type="standard" >
KEYCODE_2 OR JOYCODE_2_BUTTON9 < /newseq > < /port
>

< port type="P2_PEDAL" > < newseq type="standard"
> NONE < /newseq > < newseq type="increment" >
NONE < /newseq >
< /port > < port type="P2_PEDAL2" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < /port >
< port type="P2_PEDAL3" > < newseq
type="increment" > NONE < /newseq > < /port >
< port type="P2_PADDLE" > < newseq type="standard"
> NONE < /newseq > < newseq type="increment" >
NONE < /newseq > < newseq type="decrement" > NONE
< /newseq > < /port >
< port type="P2_PADDLE_V" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < newseq
type="decrement" > NONE < /newseq > < /port >
< port type="P2_POSITIONAL" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < newseq
type="decrement" > NONE < /newseq > < /port >
< port type="P2_POSITIONAL_V" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < newseq
type="decrement" > NONE < /newseq > < /port >
< port type="P2_DIAL" > < newseq type="standard" >
NONE < /newseq > < newseq type="increment" > NONE
< /newseq > < newseq type="decrement" > NONE <
/newseq > < /port >
< port type="P2_DIAL_V" > < newseq type="standard"
> NONE < /newseq > < newseq type="increment" >
NONE < /newseq > < newseq type="decrement" > NONE
< /newseq > < /port >
< port type="P2_TRACKBALL_X" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < newseq
type="decrement" > NONE < /newseq > < /port >
< port type="P2_TRACKBALL_Y" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < newseq
type="decrement" > NONE < /newseq > < /port >
< port type="P2_AD_STICK_X" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < newseq
type="decrement" > NONE < /newseq > < /port >
< port type="P2_AD_STICK_Y" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < newseq
type="decrement" > NONE < /newseq > < /port >
< port type="P2_AD_STICK_Z" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < newseq
type="decrement" > NONE < /newseq > < /port >
< port type="P2_LIGHTGUN_X" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < newseq
type="decrement" > NONE < /newseq > < /port >
< port type="P2_LIGHTGUN_Y" > < newseq
type="standard" > NONE < /newseq > < newseq
type="increment" > NONE < /newseq > < newseq
type="decrement" > NONE < /newseq > < /port >

< /input >
< /system >
< /mameconfig >
```

Después de guardar el archivo, cópialo en /media/usb/AML-asset/ctrlr/. Luego, podrás editar este archivo para adaptarlo a tu joystick. Mi joystick (que yo mismo monté, de ahí el problema que he mencionado anteriormente) tiene una configuración errónea de 90 grados. En el archivo XML hay dos partes: una para el joystick del jugador 1 y otra para el joystick del jugador 2. Cada línea con type="P1_JOYSTICK_UP" es la dirección que entiende Mame. La configuración real viene después como JOYCODE_1_XAXIS_RIGHT_SWITCH. Por lo tanto, esta línea significa que cuando mi joystick envía un código para DERECHA, mame lo interpretará como ARRIBA. Debajo de esto, configuré los botones 1 a 8 y los botones START y SELECT. Luego hago lo mismo con el joystick del jugador 2.

Para finalizar, puedes agregarlo al archivo de configuración mame.ini usando el siguiente comando:

```
$ echo "ctrlr myjoyremap" | sudo tee -a
/etc/mame/mame.ini
```

Eliminar software no utilizado

Esta sección no es obligatoria, pero puede resultarle útil para hacer que tu ODROID-XU4 sea más rápido y responda mejor. El XU4 tiene 2 GB de memoria, que

actualmente está muy bien para un ordenador de placa reducida. La mayoría suelen tener de 0.5 a 1 GB, aunque veo que últimamente están entrando en el mercado los de 2 a 4 GB. Así que la memoria es un recurso muy valioso, así como los ciclos de CPU. No quieras que tu máquina se ralentice mientras ves una película o juegas a un juego.

En su momento seleccioné algunos servicios que creo que no son necesarios para la instalación de Kodi/Mame. Aquí tienes la forma de detenerlos y deshabilitarlos. Sin embargo, no lo desinstalaremos para que puedas volver a habilitarlo, en el caso de que tus necesidades cambien en el futuro.

1. CUPS es un servidor de impresión, y como no necesitas imprimir desde Kodi o Mame, es seguro suprimir el servidor de impresión (denominado CUPS):

```
$ sudo systemctl stop cups
$ sudo systemctl stop cups-browsed
$ sudo systemctl disable cups
$ sudo systemctl disable cups-browsed
```

Para reactivarlo:

```
$ sudo systemctl enable cups
$ sudo systemctl start cups
```

2. UPower controla la fuente de alimentación y es muy útil para un teléfono inteligente, un ordenador portátil o un sistema integrado. Sin embargo, en el caso de un sistema de entretenimiento Kodi/Mame de un salón, la única fuente de alimentación es el enchufe de la pared y se supone que tu ODROID-XU4 está conectado todo el tiempo. Así que puedes desactivar también este componente con total seguridad:

```
$ sudo systemctl stop upower
$ sudo systemctl disable upower
```

3. Whoopsie es el demonio de informe de errores para Ubuntu, utilizado principalmente por el entorno de escritorio cuando algo falla. Como solo estamos usando Kodi, no lo necesitamos:

```
$ sudo systemctl stop whoopsie
$ sudo systemctl disable whoopsie
```

4. ModemManager es un demonio que controla dispositivos y conexiones de banda ancha móvil (2G / 3G / 4G). El ODROID-XU4 está conectado a una red Ethernet (o wifi si la tienes) y no necesita una conexión modem.

```
$ sudo systemctl stop ModemManager
$ sudo systemctl disable ModemManager
```

5. unattended-upgrades es un demonio para actualizar automáticamente el sistema. Me gusta cuando un ordenador trabaja solo para mí, pero en este caso específico, evitaremos realizar actualizaciones automáticas. La razón es que queremos un sistema de entretenimiento estable para toda la familia, que está disponible en cualquier momento. No queremos tener que hacer mantenimiento, justo antes de lanzar la película familiar, porque una actualización no se haya cargado correctamente:

```
$ sudo systemctl stop unattended-upgrades
$ sudo systemctl disable unattended-upgrades
```

Si deseas actualizar su ODROID-XU4, puede hacerlo manualmente ejecutando primero una actualización de la base de datos de los paquetes:

```
$ sudo apt update
$ sudo apt upgrade
```

Personalmente, soy un gran fanático de un software llamado Synaptic, que es una GUI para sistemas basados en apt como Ubutun y Debian. Lo recomiendo:

```
$ sudo apt install synaptic
```

Configurar el reloj

Puede sincronizar el reloj con un servidor de hora en la red y siempre tener tu ODROID configurado con la hora más precisa. Además, querrás tener ajustado el reloj con tu zona horaria.

Primero, instalamos un cliente de Protocolo de tiempo de red (NTP) que se conectará con el servidor de hora para obtener una hora precisa:

```
$ sudo apt install chrony
```

Luego, buscamos la zona horaria. Será algo como Europa/Zurich o Pacific/Auckland. Para encontrar la tuya, usa el siguiente comando:

```
$ timedatectl list-timezones
```

Busca y anota tu propia zona horaria. Supongamos que vives cerca del Polo Norte en Longyearbyen, encontrarás la zona horaria en la lista que figura más arriba como Artic/Longyearbyen. Luego, reajusta tu ODROID-XU4:

```
$ sudo timedatectl set-timezone
Arctic/Longyearbyen
```

Para comentarios, preguntas y sugerencias, visita el hilo original del Foro ODROID en https://forum.odroid.com/viewtopic.php?f=52&t=34760, o el repositorio de GitHub en **https://github.com/yimyom/odroid-xu4-setup.**

Zoneminder - Parte 2: Compilando el Paquete Desde la Fuente en el ODROID-XU4

July 1, 2019 By Michele Matacchione Linux, Tutoriales

ZoneMinder es un conjunto integrado de aplicaciones que proporciona un completo sistema de vigilancia permitiendo la captura, análisis, grabación y monitorización de cualquier CCTV o cámaras de seguridad.

Principales características

- Monitorización desde cualquier lugar: ZoneMinder cuenta una interfaz basada totalmente en la web a la que puede acceder desde cualquier dispositivo con acceso a Internet.
- Utiliza cualquier cámara: ZoneMinder te permite usar cualquier cámara analógica o por IP
- Tus datos bajo control: ZoneMinder está completamente en local; Te permite tener tus datos a buen recaudo y controlar a dónde van.
- Se puede ejecutar en pequeños y grandes entornos: adecuado para uso doméstico y en pequeñas empresas, así como para implementar soluciones empresariales multiservidor. Es compatible con muchas plataformas, incluida la tecnología ARM (ODROID está basado en una plataforma ARM)
- Mantente al tanto de todo lo que sucede: ZoneMinder te permite buscar información de una forma muy intuitiva. Profundiza en lo que estás interesado en cuestión de segundos.
- Actualizado y sin coste alguno: un equipo comprometido con el código abierto mantiene muy activo ZoneMinder.

.

Recientemente, he trasladado la aplicación ZoneMinder de mi viejo Radxa Rock Pro (una placa ARM) al ODROID-XU4 que es más potente. La mejor y más sencilla instalación que encontré fue la del ODROID-XU4 - Ubuntu 16.04.3 LTS - ZoneMinder 1.30.

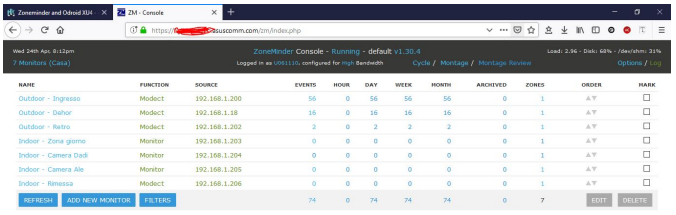


Figura 1 - Consola ZoneMinder, configurada con siete cámaras

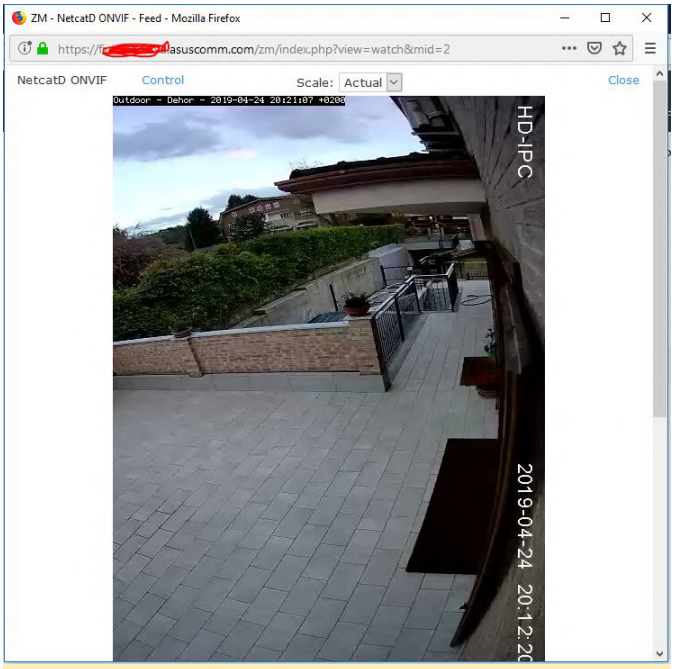


Figura 2 - ZoneMinder, observando la cámara #2

Instalación

Vamos a instalar ZoneMinder en nuestra placa ODROID-XU4. En tu tarjeta SD, instala la imagen de Ubuntu 16.04.3 LTS (la versión 4.14.y) proporcionada por Hardkernel en la dirección: https://wiki.odroid.com/odroid-xu4/os_images/linux/ubuntu_4.14/20171213 Luego actualiza el sistema:

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt dist-upgrade
$ sudo apt install linux-image-xu3
$ sudo apt autoremove
$ sudo reboot
```

Ahora instala LAMP (Linux, Apache, MySQL, PHP) en la placa:

```
$ sudo apt install apache2
$ sudo apt install mysql-server
$ sudo apt install php libapache2-mod-php php-mysql
```

Ahora instala ZoneMinder 1.30.4 (**):

```
$ sudo -i
```

Modifica la configuración de MySQL (no es necesario para ZM 1.32 o superior):

```
$ rm /etc/mysql/my.cnf (this removes the current symbolic link)
$ cp /etc/mysql/mysql.conf.d/mysqld.cnf /etc/mysql/my.cnf

$ nano /etc/mysql/my.cnf
```

En la sección [mysqld] añade lo siguiente:

```
$ sql_mode = NO_ENGINE_SUBSTITUTION
```

A continuación, reinicia MySQL:

```
$ systemctl restart mysql
```

Ahora compila ZoneMinder 1.30.4: primero añade el repositorio y descarga las herramientas,

```
$ add-apt-repository ppa:iconnor/zoneminder-master
$ apt-get update
$ apt-get install php-apcu-bc
$ sudo apt-get install gdebi-core

$ sudo wget https://raw.githubusercontent.com/ZoneMinder/ZoneMinder/master/utils/do_debian_package.sh
$ sudo chmod a+x do_debian_package.sh
$ sudo apt-get install devscripts
$ sudo apt install git
```

A continuación, el verdadero proceso de compilación "larguísimo":

```
$ sudo ./do_debian_package.sh --snapshot=NOW --branch=1.30.4 --type=local
```

Ahora instala ZoneMinder 1.30.4 (usa el comando ls para averiguar yyyymmddhhmmss y usarlo en zoneminder_1.30.4~yyyymmddhhmmss-xenial_armhf.deb):

```
$ sudo gdebi zoneminder_1.30.4~yyyymmddhhmmss-xenial_armhf.deb
```

Crea la base de datos ZoneMinder

```
$ mysql -uroot -p < /usr/share/zoneminder/db/ZoneMinder_create.sql
```

```
$ mysql -uroot -p -e "grant lock
tables,alter,drop,select,insert,update,delete,crea
te,index,alter routine,create routine,
trigger,execute on zm.* to 'zmuser'@localhost
identified by 'zmpass';"
```

Añade los permisos:

```
$ chmod 740 /etc/zm/zm.conf
$ chown root:www-data /etc/zm/zm.conf
$ chown -R www-data:www-data
/usr/share/zoneminder/
```

Activa los módulos y la configuración de ZoneMinder:

```
$ a2enmod cgi
$ a2enmod rewrite
$ a2enconf zoneminder
$ a2enmod expires
$ a2enmod headers
```

Activa ZoneMinder al inicio del sistema:

```
$ systemctl enable zoneminder
$ systemctl start zoneminder
```

Configura php.ini con la zona horaria correcta:

```
$ nano /etc/php/7.0/apache2/php.ini
```

Introduce tu zona horaria;

```
$ [Date]
$ ; Defines the default timezone used by the date
functions
$ ; http://php.net/date.timezone
$ date.timezone = America/New_York
```

Reinicia Apache:

```
$ systemctl reload apache2
```

Ahora puede encontrar la página web de ZoneMinder en http: //IP_de_tu_placa/zm y agregar tus cámaras. Cualquier tipo de cámara funcionará: cámaras ffmpeg y mjpeg, conectadas mediante Wi-Fi, cable Ethernet o USB.

Cómo Crear una Consola de Juegos Retro Monku - Parte 1: Fabricar la Carcasa

July 1, 2019 By Brian Ree Juegos

Esta es una continuación del artículo de la consola de juegos retro del mes pasado, donde aprendimos a montar una consola de juegos retro. Esta entrega te ayudará a crear una carcasa para el proyecto utilizando un ODROID-C1+/C2.

Ahora que tenemos creados nuestros botones de control personalizados y ponemos en marcha la placa con una conexión al botón de encendido, vamos a trabajar en la carcasa. Primero, necesitamos un lugar donde montar los botones. Hay dos lugares en la carcasa en los que se pueden ubicar fácilmente los botones sin que interfieran con el disipador de calor o cualquier otro componente interno. Un lugar lo tenemos en el lado derecho del panel posterior, justo encima de uno de esos puntos de conexión jumper, pwr. El otro está más hacia el lado izquierdo, justo después del centro de la carcasa. Puedes usar las mismas ubicaciones tanto para el C1+ como para el C2, aunque las cosas están un poco más ajustadas en el C1+. A continuación, se muestran dos prototipos de dispositivos con los botones montados. Los orificios no están perfectamente alineados porque se trata de montajes experimentales.

Figura 1 - Dos prototipos de dispositivos con los botones montados

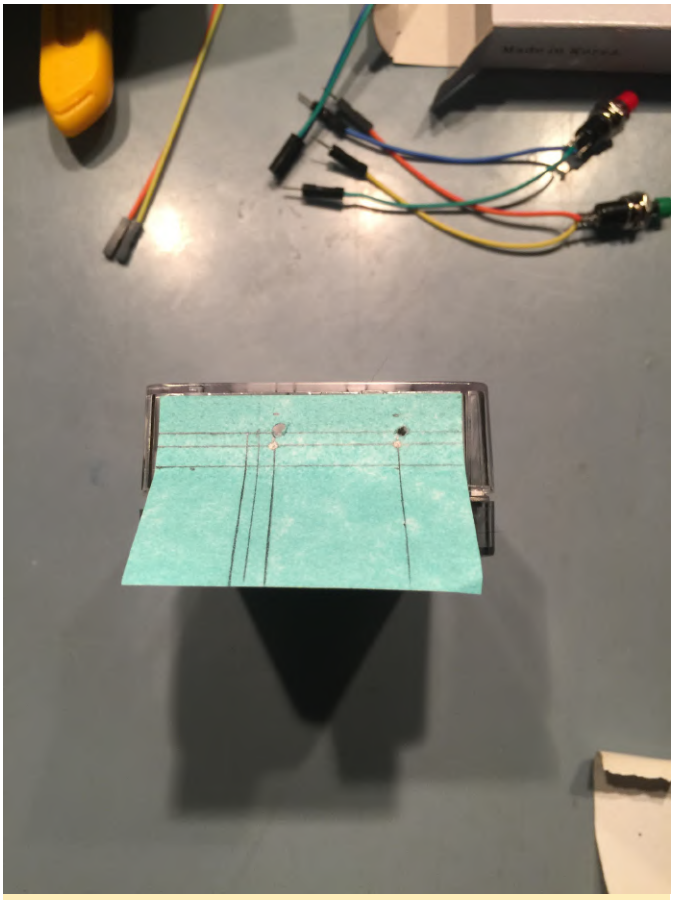


Figura 2: usa un papel adhesivo para ubicar los botones en tu carcasa

Un pequeño papel adhesivo hace maravillas en esta parte del montaje. Dependiendo del diámetro de los botones, puedes usar una regla para ubicar los orificios sobre el adhesivo y luego compararlo con el dispositivo real. Para ello, es posible que quieras configurar el dispositivo dentro de la carcasa y colocar la parte superior sin cerrar la carcasa. No quieras tener que detenerte repentinamente y volver a abrir la carcasa. Mueve el adhesivo, ajusta tus líneas y mira si puedes localizar un buen lugar.

NOTA: ten en cuenta que el disipador de calor bloqueará el interruptor si no está lo suficientemente alto dentro de la caja. NOTA: Si estás trabajando con una unidad C1 +, los jumpers de alimentación están justo al lado del punto de montaje del botón. ¡Asegúrate de que haya espacio para éstos!

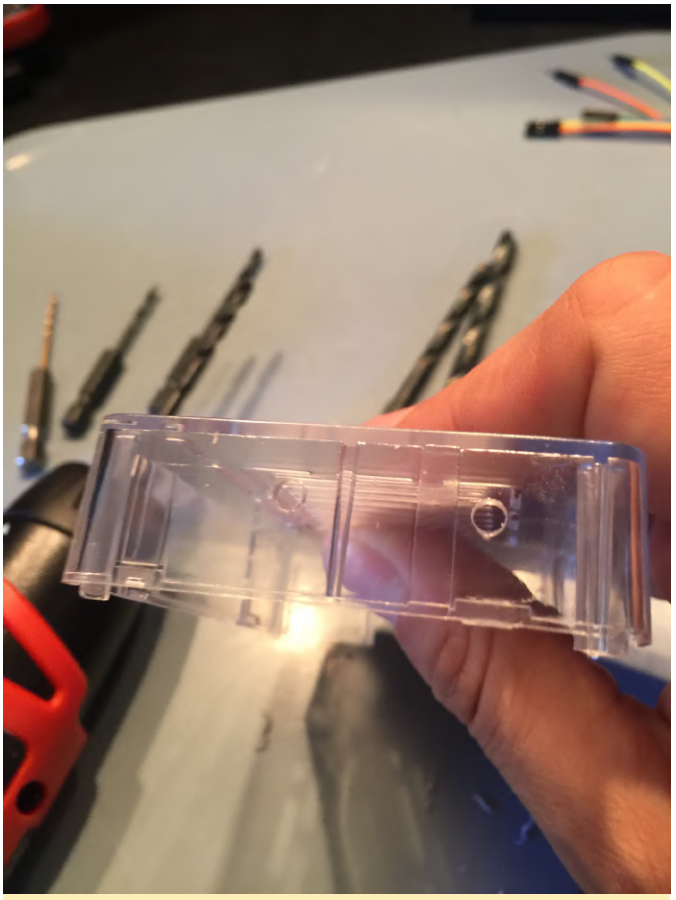
¡Es hora de taladrar! Cualquier taladro que tengas por ahí te servirá. Yo tengo un taladro barato de 20$ con un conjunto de brocas. Comprueba las dimensiones de tus botones: deben tener un diámetro de unos 6 mm (0,25 pulgadas). Asegúrate de que la broca no sea demasiado grande. Después de taladrar, no querrás que los orificios sean más pequeños. Me gusta trabajar partiendo de las brocas más pequeños hasta llegar a la más grande que necesite, en lugar de coger directamente la más grande. Cometo menos errores de esta forma, ya que cada broca va eliminando un poco de plástico. Incluso puedes redirigir el agujero hacia una dirección ligeramente diferente con cada nueva broca. Esto es ideal para afinar a última hora en el caso de que cometas un pequeño error de posición al principio. .

Figura 3: Un taladro barato con unas brocas es todo lo que necesitas es este paso

Figura 4 - Perfora a través del papel en el plástico, lo suficiente como para dejar una marca visible

Mantén tu dedo sobre la lámina adhesiao y lejos de la dirección del taladro. No dejes caer el adhesivo. Utiliza la broca más pequeña para perforar el papel y en el plástico lo suficiente como para dejar una marca visible. Retira e papel adhesivo y, mientras sostienes la carcasa, perfora cada agujero con la broca pequeña. Hazlo lentamente.

A continuación, podemos ver los orificios más grandes hechos con cada una de las brocas utilizadas. En algún momento de este proceso, darás con una broca que realmente agarre el plástico y te permita mover la carcasa. Detente en este punto. Coloca la carcasa en una superficie plana cerca del borde de tu área de trabajo para que puedas acceder a ella con el taladro. Coloca suavemente la palma de la mano sobre la caja y haz un poco de presión hacia abajo. Esto evitará que la carcasa se deforme y se rompa.



Figuras 5 y 6: Cada perforación con una broca mayor crea un orificio ligeramente más grande

Coloca un jumper en el pin 9, PUESTA A TIERRA, del cabezal GPIO de 40 pines. Coloca el otro jumper en el pin 15, (GPIO 237 si estás utilizando un ODROID-C2).

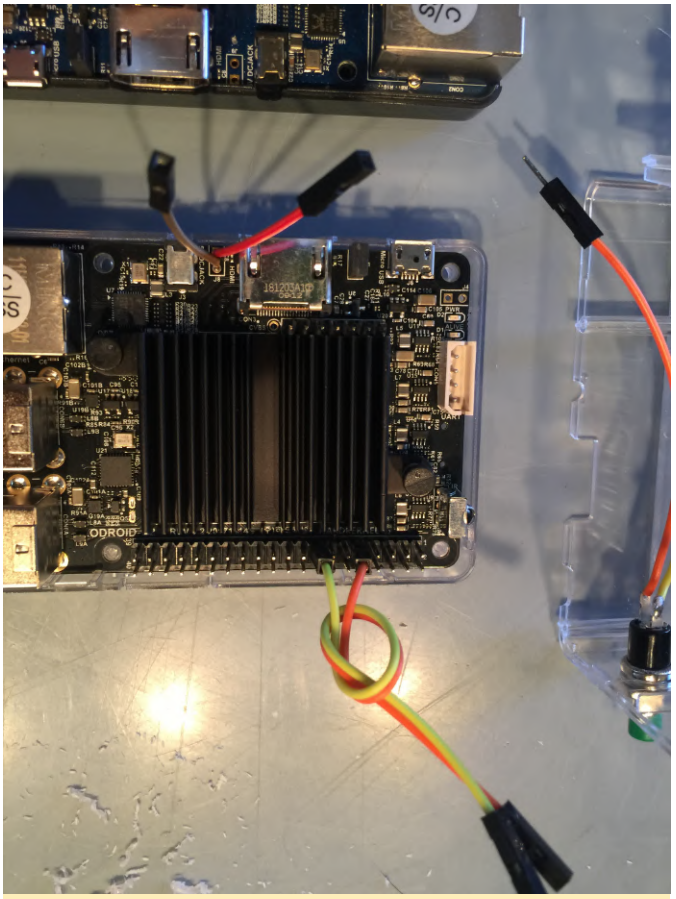


Figura 7: Coloca un jumper en el pin 9, PUESTA A TIERRA, del cabezal GPIO de 40 pines

Es posible que tengas que verificar si el cabezal ha cambiado durante algunas de las revisiones de hardware. Si tienes un ODROID-C1+, coloca los jumpers en los pines 15, GPIO 3 y 17, 3.3V.

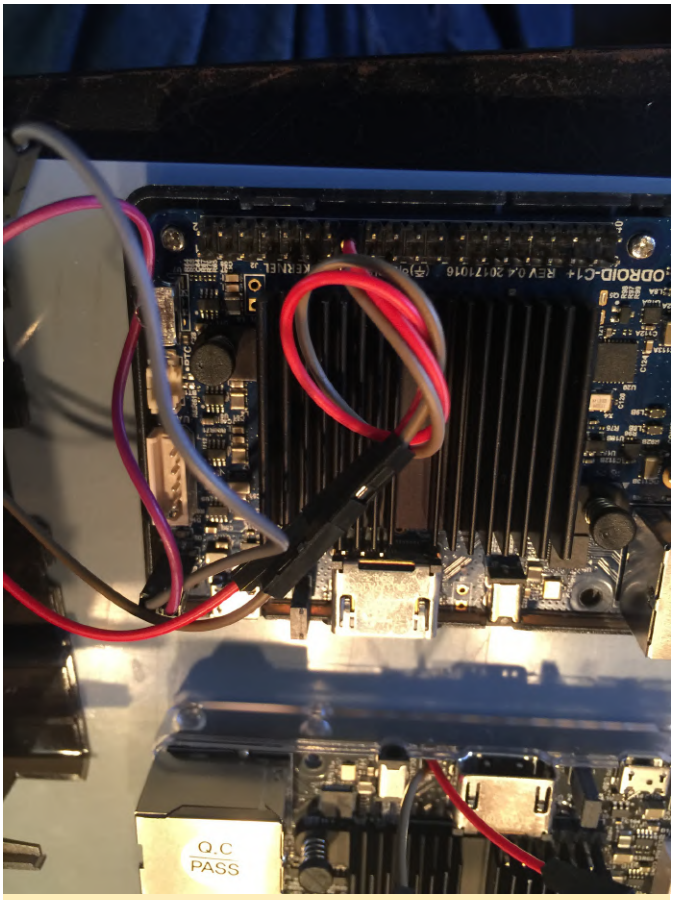


Figura 8: Comprueba si el cabezal ha cambiado durante las revisiones de hardware

De acuerdo, vamos a probarlo. No cierres la carcasa todavía, lo haremos al final. De hecho, coloque un pequeño trozo de cinta en algunos clips para evitar que se enganchen accidentalmente. Echa un vistazo a tu nueva y sorprendente consola de juegos, ¿No es brillante?

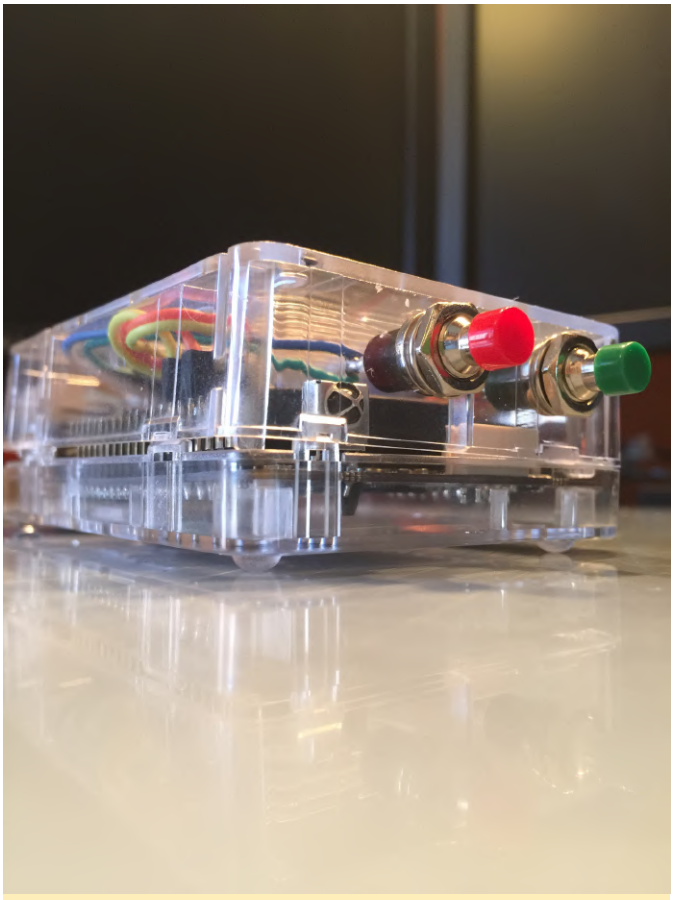


Figura 9 - Verdaderamente, una preciosidad

Para comentarios, preguntas y sugerencias, visita el post original en **http://middlemind.com/tutorials/odroid_go/mr1_build.html**.

Yocto en el ODROID-C2: Usando Yocto con el Kernel 5.0

July 1, 2019 By Gaurav Pathak Linux, Tutoriales

El Proyecto Yocto (YP) es un proyecto colaborativo de código abierto que ayuda a los desarrolladores a crear sistemas personalizados basados en Linux, independientemente de la arquitectura del hardware. Yocto no es una distribución de Linux integrada, sino que más bien crea una personalizada para ti. El proyecto proporciona un conjunto flexible de herramientas y un espacio donde los desarrolladores de todo el mundo pueden compartir tecnologías, pilas de software, configuraciones y mejores prácticas que se pueden usar para crear imágenes Linux a medida para dispositivos integrados y IOT, o donde quiera que se necesite un sistema operativo Linux personalizado. Este artículo describe los pasos y procedimientos básicos para crear una imagen personalizada de Linux para ODROID-C2 utilizando Linux-5.0.

Requisitos previos para la configuración del sistema host

Es necesario un sistema de compilación basado en Linux para el proyecto Yocto, que admita la mayoría de las principales distribuciones de servidores y escritorios de Linux. Puedes encontrar una lista de todas las distribuciones de Linux compatibles en: **https://www.yoctoproject.org/docs/current/ref-manual/ref-manual.html#detailed-supported-distros**. El proyecto Yocto necesita que se instalen ciertos paquetes en la máquina host antes de iniciar una compilación personalizada del sistema Linux para un determinado dispositivo. La lista de paquetes de host y herramientas necesarias para una compilación de yocto la puedes encontrar en: **https://www.yoctoproject.org/docs/current/brief-yoctoprojectqs/brief-yoctoprojectqs.html#brief-compatible-distro**. Para los sistemas basados en Debian, son necesarios instalar los siguientes paquetes:

```
$ sudo apt-get install gawk wget git-core diffstat
unzip texinfo gcc-multilib build-essential chrpath
```

```
socat cpio python python3 python3-pip python3-
pexpect xz-utils debianutils iputils-ping
```

Pasos para crear una imagen de Linux personalizada para Odroid-C2

Nota: Los pasos que se detallan a continuación han sido probados en un sistema host Ubuntu 16.04. Después de realizar la configuración del sistema host (es decir, instalar todos los paquetes necesarios), el siguiente paso es hacerse con la fuente del sistema de compilación del proyecto yocto. Como vamos a utilizar yocto para compilar nuestra imagen personalizada, necesitamos la distribución de referencia de yocto. Poky es una distribución de referencia del Proyecto Yocto. Contiene el sistema de compilación OpenEmbedded, BitBake y OpenEmbedded Core, así como un conjunto de metadatos para que puedas empezar a crear tu propia distro. La capa central proporciona todos los componentes comunes y las capas adicionales cambian la pila de software según sea necesario.

Obteniendo las fuentes

Las siguientes instrucciones se basan en la rama warrior upstream. La rama de Warrior tiene soporte para compilar Linux Kernel 5.0. Descarga la distribución de referencia poky para yocto.

```
$ mkdir yocto-odroid
$ cd yocto-odroid
$ git clone -b warrior
git://git.yoctoproject.org/poky.git
```

Descarga la capa BSP ODROID :

```
$ git clone -b warrior
git://github.com/akuster/meta-odroid
```

Descarga la capa openembedded:

```
$ git clone -b warrior
https://github.com/openembedded/meta-
openembedded.git
```

Una vez que hayas descargado todas las fuentes en el directorio de yocto-odroid, asegúrate de que la estructura del directorio sea como la de la Figura 1.

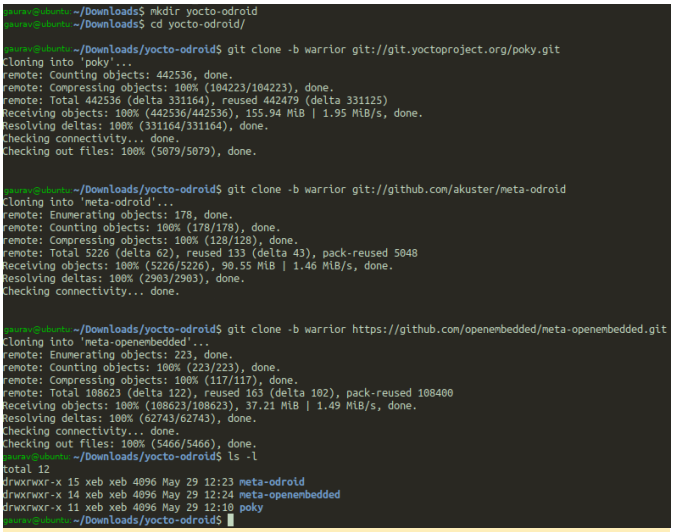


Figura 1 - Estructura de directorios de la configuración de compilación de Yocto

Iniciando una compilación

Una vez que tenga todas las fuentes y la estructura de directorios como la que se muestra en la Figura 1, puedes iniciar la compilación con los siguientes pasos: Inicia la configuración de compilación:

```
$ source poky/oe-init-build-env
```

El comando anterior creará un directorio de compilación y la moverá a ese directorio. Ahora tenemos un espacio de trabajo donde podemos compilar un emulador y las imágenes basadas en placas de referencia, por ejemplo qemuarm. Para hacer que una imagen sea compatible con nuestra máquina (es decir, ODROID-C2) necesitamos agregar la capa BSP ODROID-C2 y la capa meta-openembedded en el espacio de trabajo.

```
$ bitbake-layers add-layer ../meta-odroid/
$ bitbake-layers add-layer ../meta-
openembedded/meta-oe/
$ bitbake-layers add-layer ../meta-
openembedded/meta-python/
$ bitbake-layers add-layer ../meta-
openembedded/meta-networking/
```

Después, necesitamos modificar el archivo de configuración ubicado en el directorio conf dentro del directorio de compilación. Abre el archivo local.conf ubicado en conf/local.conf usando tu editor de texto favorito.

```
$vi conf/local.conf
```

Realice las siguientes modificaciones en el archivo local.conf: Busca

```
MACHINE ?? = “qemux86”
```

Coméntalo colocando una # antes o sustitúyelo por

```
MACHINE ?? = “odroid-c2”
```

Encuentra y comenta las siguientes líneas poniendo una # delante:

```
PACKAGECONFIG_append_pn-qemu-system-native = "
sdl"
PACKAGECONFIG_append_pn-nativesdk-qemu = " sdl"
```

Buscar la línea

```
EXTRA_IMAGE_FEATURES ?= "debug-tweaks"
```

Añade lo siguiente después de "debug-tweaks"

```
ssh-server-openssh
```

de modo que la línea quede así:

```
EXTRA_IMAGE_FEATURES ?= "debug-tweaks ssh-server-
openssh"
```

Ahora, copia las siguientes líneas y pégalas al final de local.conf:

```
PACKAGECONFIG_remove_pn-xserver-xorg = "glamor"
IMAGE_FEATURES_append = " x11 "
DISTRO_FEATURES_append = " opengl x11"
DISTRO_FEATURES_remove = "wayland"

PREFERRED_PROVIDER_virtual/libgl = "mesa-gl"
PREFERRED_PROVIDER_virtual/libgles2 = "mali"
PREFERRED_PROVIDER_virtual/libgles1 = "mali"
PREFERRED_PROVIDER_virtual/egl = "mali"
PREFERRED_PROVIDER_virtual/mesa = "mesa"

IMAGE_INSTALL_append = "libgcc libgcc-dev
libstdc++ libstdc++-dev libstdc++-staticdev
        autoconf automake ccache chkconfig glib-
networking
        packagegroup-core-buildessential pkgconfig
        boost cmake zlib glib-2.0
        rng-tools
        logrotate
        lrzsz
        watchdog
        util-linux
        pciutils
        usbutils
"
IMAGE_ROOTFS_EXTRA_SPACE ="2097152"
INHERIT += "extrausers"
EXTRA_USERS_PARAMS = "usermod -P root root; "
```

Ahora guarda y cierra el archivo local.conf. El espacio de trabajo ya está listo para iniciar una compilación. Existen varios tipos de imágenes de destino que pueden compilarse usando yocto para diferentes usos Aquí, se compila una imagen gráfica basada en X11 y matchbox. Ejecuta el siguiente comando para iniciar una compilación:

```
$ bitbake core-image-sato
```

La compilación tardará un tiempo dependiendo de la potencia de procesamiento de la máquina host y la velocidad de tu conexión a Internet. Puede necesitar de 30 minutos a varias horas.

Los pasos para acelerar el proceso de compilación los puedes encontrar en: https://www.yoctoproject.org/docs/2.7/dev-manual/dev-manual.html#speeding-up-a-build Si, durante la compilación, aparece algún error relacionado con "Timer Expired ", por ejemplo:

```
aclocal: error: cannot open
/home/gaurav/Downloads/yocto-
odroid/build/tmp/work/aarch64-poky-linux/alsa-
plugins/1.1.8-r0/recipe-
sysroot/usr/share/aclocal/ax_check_mysqlr.m4:
Timer expired
autoreconf: aclocal failed with exit status: 1
ERROR: autoreconf execution failed.
WARNING: exit code 1 from a shell command.
ERROR: Function failed: do_configure (log file is
located at /home/gaurav/Downloads/yocto-
odroid/build/tmp/work/aarch64-poky-linux/alsa-
plugins/1.1.8-r0/temp/log.do_configure.9191)
ERROR: Logfile of failure stored in:
/home/gaurav/Downloads/yocto-
odroid/build/tmp/work/aarch64-poky-
linux/mpg123/1.25.10-r0/temp/log.do_configure.9296
```

Luego simplemente limpia el caché de estado y vuelve a iniciar la compilación desde la imagen de destino usando:

```
$ bitbake core-image-sato -c cleansstate
$ bitbake core-image-sato
```

Creando una tarjeta SD de arranque

Si la compilación ha tenido éxito, las imágenes de destino las pueden encontrar en el directorio "tmp/deploy/images/odroid-c2". Se puede usar herramientas de línea de comandos como dd para crear una tarjeta SD de arranque. El usuario debe tener cuidado al utilizar esta herramienta. Si se elige el dispositivo incorrecto, puede sobrescribir un disco duro que pertenezca al host de compilación. Ejecuta el siguiente comando para escribir el archivo de imagen en su tarjeta SD.

```
$ cd tmp/deploy/images/odroid-c2
$ xzcat core-image-sato-odroid-c2.wic.xz | sudo dd
of=/dev/sdX bs=4M iflag=fullblock oflag=direct
conv=fsync status=progress
```

Asegúrese de que sdX apunte al dispositivo correcto (es decir, la tarjeta SD montada). Puede confirmar y buscar la tarjeta SD de destino ejecutando este comando:

```
$ dmesg|tail
[19001.706817] mmc0: new high speed SDHC card at
address e624
[19001.707251] mmcblk0: mmc0:e624 SU08G 7.40 GiB
[19001.718156] mmcblk0:
```

En el caso anterior, la tarjeta SD montada es "mmcblk0", en algunas máquinas host también puede aparecer como "sdb" o "sdc".

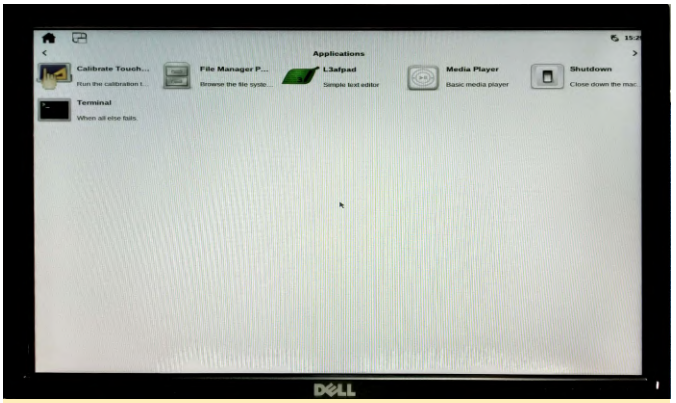


Figura 2 - Imagen Sato del proyecto Yocto de ODROID-C2

Activando y desactivando GPIO en El Espacio del Kernel

Existen varios tutoriales y códigos de muestra para acceder a GPIO. Casi todos ellos se apoyan en el acceso gpio basado en legacy. El siguiente código muestra cómo activar y desactivar gpio usando un nuevo acceso a gpio basado en descriptor. Es posible que el código que se muestra a continuación no sea una forma adecuada de acceder a gpio en el espacio del kernel, ya que simplemente es un ejemplo. Basándonos en el siguiente código, se puede programar un driver para crear una entrada de nodo en /dev como es /dev/gpio-test y luego ese nodo se puede usar para enviar comandos desde el espacio del usuario para activar y desactivar el gpio usando el código de espacio del kernel.

```
/*File: gpio-toggle.c*/

#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/printk.h>
#include <linux/init.h>
#include <linux/delay.h>
#include <linux/gpio/driver.h>
#include <dt-bindings/gpio/meson-gxbb-gpio.h>

struct gpio_chip *chip;

static int chip_match_name(struct gpio_chip *chip,
void *data)
{
  printk(KERN_INFO "Label: %s", chip->label);
  printk(KERN_INFO "Name: %s", chip->parent-
>init_name);
  printk(KERN_INFO "OF Node Full Name: %s", chip-
>of_node->full_name);
  return !strcmp(chip->label, data);
}

int gpio_test_init(void)
{
  int err = 0;

  printk(KERN_DEBUG "Init Called
");
  chip = gpiochip_find("periphs-banks",
chip_match_name);
  if (!chip) {
    printk(KERN_ERR "Cannot Find A GPIO Chip");
    return -ENODEV;
  }
  printk(KERN_DEBUG "Got valid GPIO Chip Total num
gpios %d
",
      chip->ngpio);
```

```
  err = chip->get(chip, GPIOX_11);
  printk(KERN_INFO "Before Setting Value %d
", err);

  err = chip->direction_output(chip, GPIOX_11, 1);
  if (err < 0) { printk(KERN_DEBUG "Error Setting
GPIO Direction %d", err); } chip->set(chip,
GPIOX_11, 1);

  err = chip->get(chip, GPIOX_11);
  printk(KERN_INFO "After Setting Value %d
", err);

  mdelay(2000);

  chip->set(chip, GPIOX_11, 0);

  err = chip->get(chip, GPIOX_11);
  printk(KERN_INFO "After Clearing Value %d
", err);
  return 0;
}

void gpio_test_exit(void)
{
  printk(KERN_DEBUG "Exiting....
");
}
module_init( gpio_test_init);
module_exit( gpio_test_exit);
MODULE_LICENSE("GPL");
```

Below is the Makefile to compile the above Kernel Module:

```
obj-m += gpio-toggle.o

KSRC = </path/to/pre-compiled/kernel-source>

EXTRA_CFLAGS = -I$(KSRC)/drivers/pinctrl/meson
EXTRA_CFLAGS += -I$(KSRC)/drivers/

CFLAGS_gpio-toggle.o := -DDEBUG

all:
  make -C $(KSRC) M=$(PWD) modules
clean:
  make -C $(KSRC) M=$(PWD) clean
```

Ten en cuenta que en el Makefile anterior, la variable KSRC debe configurarse para que apunte a la ubicación/directorio donde se encuentra el kernel Linux 5.0 precompilado. El sistema de compilación de yocto coloca el kernel de Linux compilado en:

```
build/tmp/work/odroid_c2-poky-linux/linux-
stable/5.0.6+gitAUTOINC+172634f02e_machine-
r0/linux-odroid_c2-standard-build/
```

La ruta absoluta del Linux Kernel 5.0 precompilado en nuestro caso es:

```
/home/gaurav/Downloads/yocto-
odroid/build/tmp/work/odroid_c2-poky-linux/linux-
stable/5.0.6+gitAUTOINC+172634f02e_machine-
r0/linux-odroid_c2-standard-build
```

Entonces, la variable KSRC se convertiría en:

```
KSRC = /home/gaurav/Downloads/yocto-
odroid/build/tmp/work/odroid_c2-poky-linux/linux-
stable/5.0.6+gitAUTOINC+172634f02e_machine-
r0/linux-odroid_c2-standard-build
```

Comando para compilar y limpiar el módulo del kernel anterior:

```
$ make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu-
$ make ARCH=arm64 CROSS_COMPILE=aarch64-linux-gnu-
clean
```

Una vez que el módulo del kernel anterior se haya compilado correctamente, debe generar un archivo .ko. En nuestro caso, será gpio-test.ko.

Si la placa ODROID está conectada a una red, transfiere el archivo usando "scp". De lo contrario, si la placa no está conectada, puedes usar la utilidad de transferencia de archivos Xmodem de minicom para transferir un archivo a la máquina de destino. NOTA: el paquete de utilidad de recepción Xmode se compila e instala en nuestra máquina de destino como parte de la compilación de Yocto. Conecta un LED al pin 13 del cabezal J2 de 40 pines, como se muestra a continuación:

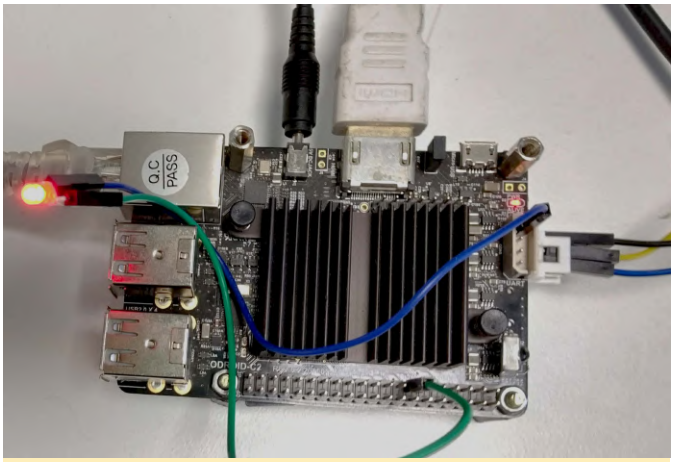


Figura 3 - Conexión Led en la placa OdroidC2

Luego inserta el módulo ejecutando:

```
$ insmod gpio-toggle.ko
```

El LED conectado a GPIOX_11, es decir, el pin 13 del cabezal J2 en Odroid C2 debería encenderse y apagarse una vez.

Juegos Linux en ODROID: Juegos en ODROID-N2 - Escritorio y gl4es

July 1, 2019 By Tobias Schaaf Juegos

El ODROID-N2 todavía es muy reciente, aunque ya tiene un par de meses. A muchos les gusta porque tiene un procesador y una GPU bastante rápidos, además cuenta con más memoria RAM que los anteriores modelos ODROID. Aun así, debido a que no contamos con drivers X11 para el sistema, las posibilidades del ODROID-N2 están un tanto limitadas por ahora. Quiero analizar de lo que es capaz actualmente, y hasta donde podemos llegar a la hora de ejecutar juegos en el ODROID-N2 desde un escritorio.

Situación actual

Ya hemos visto que existen imágenes de juegos por ahí, y usar Retroarch con una interfaz como EmulationStation no es gran cosa ni es nada nuevo. Lo hemos hecho antes, y no lo voy a tratar en este artículo. PPSSPP también parece funcionar bien, aunque estas son aplicaciones que se ejecutan en modo de aplicación individual, lo que significa que son las únicas aplicaciones que se ejecutan y no se pueden usar desde un escritorio (aunque se inician desde un escritorio y, por lo tanto, pueden usarse igualmente). Sin embargo, muchos usuarios quieren usar el N2 como un ordenador de escritorio, con la esperanza de tener una experiencia fluida tanto en el propio escritorio como en las aplicaciones, pero dado que el ODROID-N2 no tiene drivers de video X11, las opciones se reducen notablemente.

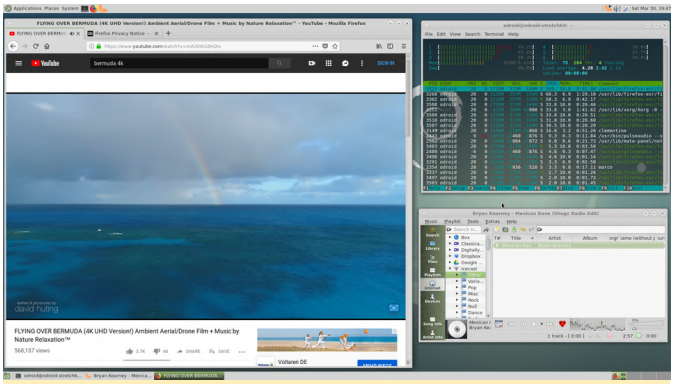


Figura 1: El escritorio MATE del ODROID-N2 con "superposición" permite ventanas de terminal transparentes

El N2 es bastante bueno como para sustituir a ordenador de escritorio, y su rápida CPU permite la superposición del escritorio, lo cual permite disponer de ventanas transparentes, dando al N2 la apariencia de un ordenador de escritorio más rápido. La falta de aceleración por hardware supone una notable desventaja, especialmente en el navegador web.

Aunque Chromium es muy lento, especialmente cuando reproducimos videos de YouTube, Firefox funciona ligeramente mejor, aunque ambos no tienen aceleración por hardware, de modo que no hay soporte para WebGL o desplazamiento acelerado por hardware. Aun así, con la ayuda de gl4es de @ptitSeb (un contenedor OpenGL para OpenGL ES), muchos programas se pueden ejecutar con aceleración OpenGL en modo "Pantalla completa". Esto nos permite ejecutar varios juegos que ya he compilado y configurado para los ODROIDs, incluido el ODROID-N2.

Configurando el entorno

Después de instalar el escritorio MATE con los drivers de la GPU, necesitaba instalar gl4es desde mi repositorio, aunque también instalé monolibs-odroid ya que ofrece una versión libSDL2 que admite OpenGL, que por defecto está desactivado en las versiones libSDL2 para ODROID, que normalmente usan OpenGL ES. Aún lo necesitamos para que todo funcione, así que instalaremos ambos:

```
$ apt install -t stretch libgl-odroid monolibs-odroid
```

La mayoría de las aplicaciones no se inician directamente desde el menú y deben arrancarse desde la línea de comandos, de lo contrario utilizarán el Software OpenGL de MESA o la versión SDL2 incorrecta. Por lo tanto, configuro mi entorno definiendo las siguientes variables:

```
export LIBGL_FB=1
export LIBGL_GL=21
export LD_LIBRARY_PATH=/usr/local/lib/monolibs
```

Algunos de los juegos que hemos probamos ya estaban diseñados para usar OpenGL y gl4es en primer lugar, de modo que la única diferencia para estas aplicaciones es la opción "LIBGL_FB=1", así que merece la pena poner esta variable en /etc/environment, de lo contrario quedará activada para todas las aplicaciones (requiere reiniciar). No todos los programas se pueden ejecutar con las otras dos opciones del archivo de entorno, de modo que debes omitirlas. Con esto pude iniciar la mayoría de las aplicaciones directamente desde el terminal. En muy contadas ocasiones, tuve que hacer algo más, aunque lo detallaré cuando llegue el momento.

Una cosa que observé es que todos los juegos parecen ejecutarse a unos 45 FPS. No creo que esto se deba al bajo rendimiento, probablemente esté relacionado con algún tipo de limitación de los drivers X11. Ahora, con nuestra configuración completada, veamos qué juegos podemos ejecutar.

Cendric

Cendric es un juego de rol con un ligero estilo retro. Muchos de los gráficos parecen estar dibujados a mano o por el ordenador, aunque el juego en sí no está nada mal. Utiliza X11 y OpenGL directamente, de modo que podría iniciarse prácticamente por sí sólo. Aunque la variable LIBGL_FB=1 está definida, puedes ejecutar el juego desde la línea de comandos o incluso desde el menú. El juego presenta algunos problemas gráficos con otros ODROID, aunque me he encontrado con el problema de que cuando cambias desde la vista de mapa del mundo a la vista de las mazmorras, la pantalla no se actualiza y aparece la última imagen que viste del nivel. Salte del juego con ALT + F4, reiniciar el juego y reanudarlo, y aparecerás en la nueva área a la que querías acceder. Por lo general, el juego funciona bien con los drivers,

aunque el error al cambiar la vista del mundo es un poco molesto.

Figura 2 - Cendric en el ODROID-N2 se inicia directamente desde el menú de inicio (con la variable de entorno)

Dune Legacy

Este es uno de mis juegos favoritos de todos los tiempos. Dune 2, el abuelo de todos los RTS modernos (juegos de estrategia en tiempo real), Dune Legacy es una versión mejorada y optimizada del juego con mejor interfaz y opciones de control que el original.

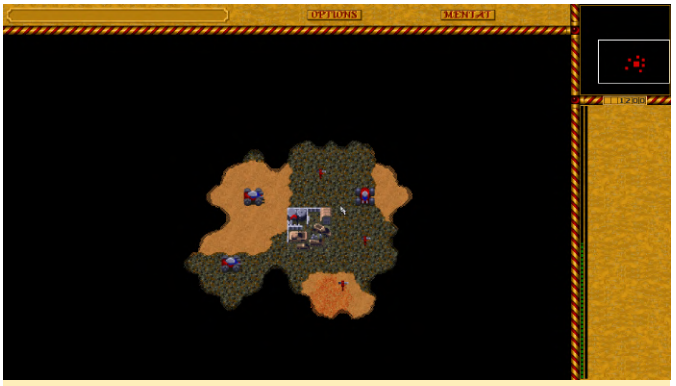


Figura 3 - Dune Legacy en el ODROID-N2: uno de los primeros niveles

Dune Legacy depende de SDL2 como driver gráfico, que en mi imagen ARM64 normalmente usa OpenGL ES por defecto, pero como no tenemos OpenGL ES con soporte de X11, necesitamos una versión que admita OpenGL y use gl4es. Éste funciona siempre que ejecutemos el juego con "LD_LIBRARY_PATH =/usr/local/lib/monolibs", lo que significa que es mejor iniciarlo desde la línea de comandos.

Esto, junto con la opción LIBGL_FB = 1, es suficiente para ejecutar el juego. Aparte de algunos errores de transparencia, no he observado ningún problema con el juego. Todos los menús funcionan, el juego se ejecute bien y es fluido. Es completamente jugable y se puede disfrutar bastante sobre el ODROID-N2.

EasyRPG Player

EasyRPG Player es un intérprete para juegos RPG Maker 2000 y 2003 que también requiere SDL2, de modo que debes iniciarlo desde la línea de comandos. Por lo que sé, no utiliza ningún shader, así que la aceleración por hardware realmente no es necesaria, aunque si es un requisito para iniciar el intérprete.

Figura 4 - Blue Skies ejecutándose en EasyRPG Player

No hay mucho que decir al respecto, excepto que funciona bien. No he observado ningún problema que ya no estuviese presente en cualquier otro ODROID.

Friking Shark

Este juego es un remake del clásico juego de arcade Flying Shark (también conocido como Sky Shark), que fue adaptado para muchas y diferentes plataformas, como Amiga, C64 o NES. Este remake 3D utiliza OpenGL 2.0 y shaders, por lo que no es tan simple como otros juegos y tenía algunos problemas en el pasado con gl4es. Actualmente, se ejecuta magníficamente en el ODROID-N2 con gl4es y LIBGL_FB=1. Como solo se ejecuta sobre OpenGL y no necesita SDL2, se puede iniciar directamente desde el menú si configuras la variable de entorno.

Figura 5 - Friking Shark en el ODROID-N2

Frogatto (and friends)

Este juego de plataformas resultó ser uno de los juegos más complicados de ejecutar. Creé una versión que fuera capaz de ejecutarse directamente bajo OpenGL ES/EGL/X11, pero no funcionaba en el ODROID-N2 al no contar con drivers X11 OpenGL ES.

Esta versión no se puede usar con gl4es porque fue reescrita para utilizar OpenGL ES/EGL/X11, así que recurrí a la versión de Debian. Solo necesita SDL 1.2 así como OpenGL, y debe iniciarse con LIBGL_FB y LD_LIBRARY_PATH. Aunque este método funciona, saca a la luz otro problema que solucioné con mi propia versión del juego: sólo se inicia con una resolución de 800x600 y esto no se puede configurar. Por lo tanto, la pantalla se ve bastante mal a menos que también cambies la resolución de tu ODROID a 800x600. Aun así, el juego funciona bien a pesar de estos problemas gráficos.

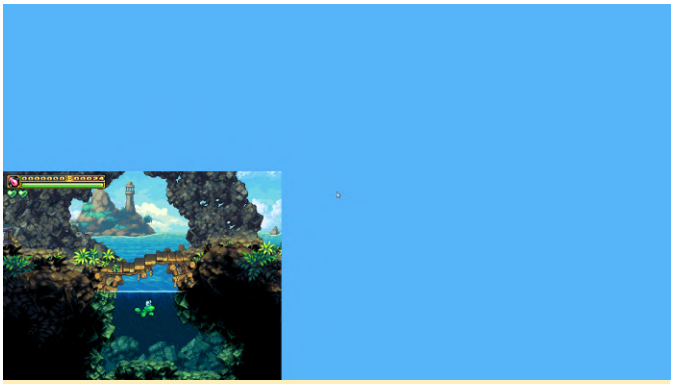


Figura 6: Frogatto and Friends solo se ejecuta en 800x600; el resto aparece en blanco

Gigalomania

Este remake de MegaLoMania es un juego de estrategia que me encantaba en Amiga, utiliza SDL2 para renderizar sus gráficos, por lo que también necesita LIBGL_FB y LD_LIBRARY_PATH. El juego no hace nada sofisticado con los gráficos, así que debería funcionar bien.

Figura 7 - Gigalomania en el ODROID-N2

GZDoom

GZDoom es un motor que te permite ejecutar juegos basados en el motor Doom. Hay toneladas de juegos que incluso se publican hoy en día que llevan el motor Doom a sus límites y más allá. Esta versión en particular usa OpenGL para optimizar gráficos y añade más efectos de iluminación, niebla, etc. Todo debe compilarse, de modo que tan pronto como tengas LIBGL_FB configurado, podrás iniciarlo desde el menú o desde el terminal que desee.

Figura 8 - Castlevania: Simon's Destiny para GZDoom

Figura 9 - Classic Doom ejecutándose en ODROID-N2 con GZDoom

Hedgewars

Hedgewars fue uno de los primeros juegos que probé con este método y llego a funcionar bastante bien. Este clon de Worms es muy divertido y debería funcionar perfectamente en tu ODROID-N2 con gl4es. Pero requiere tanto LIBGL_FB como LD_LIBRARY_PATH. Asegúrate de configurar el juego para usar el modo de pantalla completa.

(Figura 10: Hedgewars en el ODROID-N2 funciona bastante bien incluso con muchos elementos en pantalla como son las hojas caídas y las explosiones)

Naev

Este fue un poco complicado, ya que el juego se inicia sin un archivo de configuración y en modo ventana, aunque debe controlarse con el ratón y el teclado, lo que hace que sea muy difícil de interactuar al principio.

"Afortunadamente" la ventana del juego empezó a parpadear de fondo cuando inicié el juego por primera vez, de modo que pude mover el marco de la ventana a la esquina inferior izquierda donde se muestra la imagen del juego. Después de hacer eso, el menú se volvió más amigable y pude configurar la pantalla para usar el modo de pantalla completa en 1080p, tras lo cual el juego funcionó perfectamente.

El juego solo usa SDL 1.2 y OpenGL que ya estaba vinculado correctamente, de modo que debería poder ejecutarse directamente con solo LIBGL_FB desde el menú.

Figura 11 - Menú principal de Naev

Figura 12: Una captura de pantalla del juego en la que se observa que todo funcionaba bien y no hay problemas

Neverball / Neverputt

Ambos juegos del repositorio de Debian funcionan bien, aunque requieren LIBGL_FB y LD_LIBRARY_PATH, ya que utilizan SDL2 y OpenGL para el renderizado, por lo que es más fácil iniciarlos desde la línea de comandos. Después de ello los juegos debería funcionar sin problemas.

Figura 13 - Neverball en el ODROID-N2, con bonitos efectos de iluminación sobre el balón de Lava

OpenXCom

Este es uno de mis juegos favoritos de todos los tiempos. Un juego táctico basado en turnos con gestión de recursos y estrategia. Tiene un montón de opciones y es compatible con el original UFO Enemy Unknown y Terror de los juegos Deep. El juego utiliza SDL 1.2 y OpenGL, que ya está vinculado correctamente, por lo que debería funcionar simplemente iniciándolo desde el menú una vez que configures LIBGL_FB.

Figura 14: OpenXCom en el ODROID-N2 no es muy exigente gráficamente, se mantiene a 60 FPS constantemente.

Este juego es un poco confuso: aunque es el único que se ejecuta a 60 FPS, en este caso utiliza el renderizado por software. Si activo OpenGL para el renderizado, el juego se bloquea, de modo que los sombreadores no funciona, aunque funcionan en cierta medida en otras plataformas. Aun así, el juego es completamente jugable a pesar de que solo tira de software aportando una experiencia nostálgica, aunque moderna.

RVGL (ReVolt)

Este fue un poco más complicado de ponerlo a trabajar. Tiene diferentes backends que permiten cambiar entre OpenGL y OpenGL ES, además de configurar si quieres usar shaders o no. Puede definir esto en un archivo de configuración, pero el archivo de configuración solo se crea si cambias algo, lo que significa que debes poder iniciar el juego para se cree el archivo de configuración, pero no puedes iniciar el juego a menos que tenga la configuración correcta, lo que cual complica las cosas bastante. Yo copié la configuración de mi ODROID XU3 y cambié /home/odroid/.rvgl/profiles/rvgl.ini:

```
GLProfile = 1
Shaders = 0
ScreenWidth = 1920
ScreenHight = 1080
```

Este juego también requiere que utilicemos, además de LIBGL_FB y LD_LIBRARY_PATH, la opción LIBGL_GL=21, que fija gl4es en modo de compatibilidad OpenGL 2.1, de lo contrario, el juego se bloqueará tras aparecer el primer logotipo. Después de todos estos cambios, el juego funciona estupendamente.

Figura 15: RVGL, una de las aplicaciones OpenGL con un estupendo aspecto para los ODROIDs

SuperTux 2

Esta versión de Mario para Linux, si se instala desde mi repositorio tal y como se describe en https://forum.odroid.com/viewtopic.php?t=5908, debería aparecer vinculada a las librerías correctas, de modo que sólo es necesario LIBGL_FB para iniciar el juego. Es posible que quieras editar /home/odroid/.local/share/supertux2/config y activar la pantalla completa #t y configurar las dimensiones correctas de la pantalla.

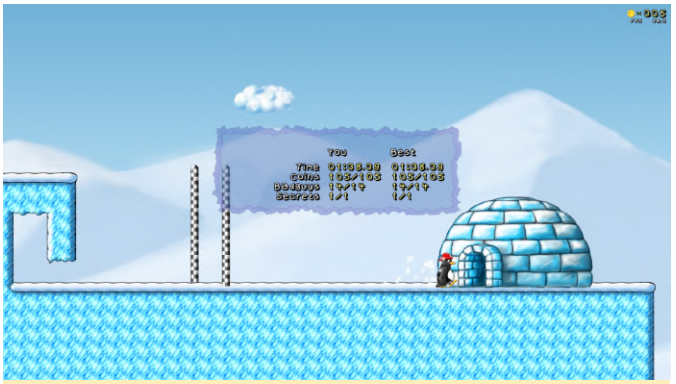


Figura 16 - Super Tux 2 en el ODROID-N2

SuperTuxKart

Este clon de Mario Kart de Linux no fue fácil de ejecutar y presenta algunos problemas aquí y allá. En primer lugar, la versión de mi repositorio fue creada para OpenGL ES y no funciona con el ODROID-N2, pero la versión de Debian Stretch funciona ya fue diseñada para OpenGL. Requiere LIBGL_FB y LD_LIBRARY_PATH para funcionar, aunque normalmente necesitaría OpenGL 3.1 o superior, que no es compatible con gl4es. Por lo tanto, nos encontramos ante un modo heredado alternativo, que se ejecuta en OpenGL 1.x sin sombreadores. Tiene gráficos muy reducidos, sin transparencia ni reflejos, y creo que algunos elementos ni siquiera se muestran para nada. Aún así, el juego funciona y es bastante rápido.

Figura 17 - Super Tux Kart en el ODROID-N2, no es perfecto, pero es divertido

VICE

VICE es el Versatile Commodore Emulator. Dado que utiliza SDL2, requiere tanto LIBGL_FB como LD_LIBRARY_PATH. Una vez iniciado el emulador, puedes abrir la pantalla de opciones con F12 y activar el modo de pantalla completa, tras lo cual la imagen debería verse normal y podrás ejecutar tus juegos favoritos de Commodore.

Figura 18 - Giana Sisters para C64 ejecutándose bajo VICE en el ODROID-N2

Podría ejecutar incluso más juegos como Witchblast o Yquake 2 (remake de Quake 2 OpenGL), que funcionan bien, pero presentas algunos problemas (Yquake funcionó bastante bien, pero es más lento de lo esperado), y hay algunos juegos que no he tenido tiempo de probar aún.

Conclusión

Por supuesto, no todos los juegos que he probado ha llegado a funcionar, hubo algunos que no pude iniciar. CGMadness, por ejemplo, se bloqueó cuando empecé a jugar, a pesar de que el menú principal funcionaba. CorsixTH (Theme Hospital) no actualizaba la pantalla correctamente pudiendo leer cualquier item en pantalla. OpenClaw no se inició para nada, no logré entender por qué. Algunos juegos funcionaban, pero tenían problemas gráficos, como Witchblast, pero por lo general me sorprendió la cantidad de juegos que realmente llegaron a ejecutarse. Algunos de estos juegos podrían incluso ponerse en segundo plano simplemente presionando alt + tab para volver al escritorio, que funciona bien en algunos juegos que admitían esta opción. Todo esto ha sido posible gracias al impresionante driver gl4es de @ptitSeb, que nos permite ejecutar estas aplicaciones desde un escritorio. Puede que no sea capaz de ejecutar todo lo que queramos desde un escritorio, pero recurriendo a algunos trucos, una gran cantidad de aplicaciones parecen poder ejecutarse en el N2, muchas de las cuales no están disponibles bajo fbdev y requieren un entorno X11 funcional.

Cámara Infrarroja Térmica ODROID-Go

July 1, 2019 By Andrew Thomas ODROID-GO, Mecaniqueo, Tutoriales

Este es un simple proyecto de cámara térmica por infrarrojos (IR) para el sistema portátil ESP32 del ODROID-GO. Permite guardar datos en una tarjeta SD, así como tener una interfaz Bluetooth básica para enviar datos de forma inalámbrica desde la cámara a un ordenador, tablet o teléfono móvil. Está badado en los módulos matriz térmicos por infrarrojos MLX90640 de 32x24 píxeles que puedes obtener a un precio relativamente económico en muchas tiendas online. A continuación, puedes ver una foto de esta cámara en acción.

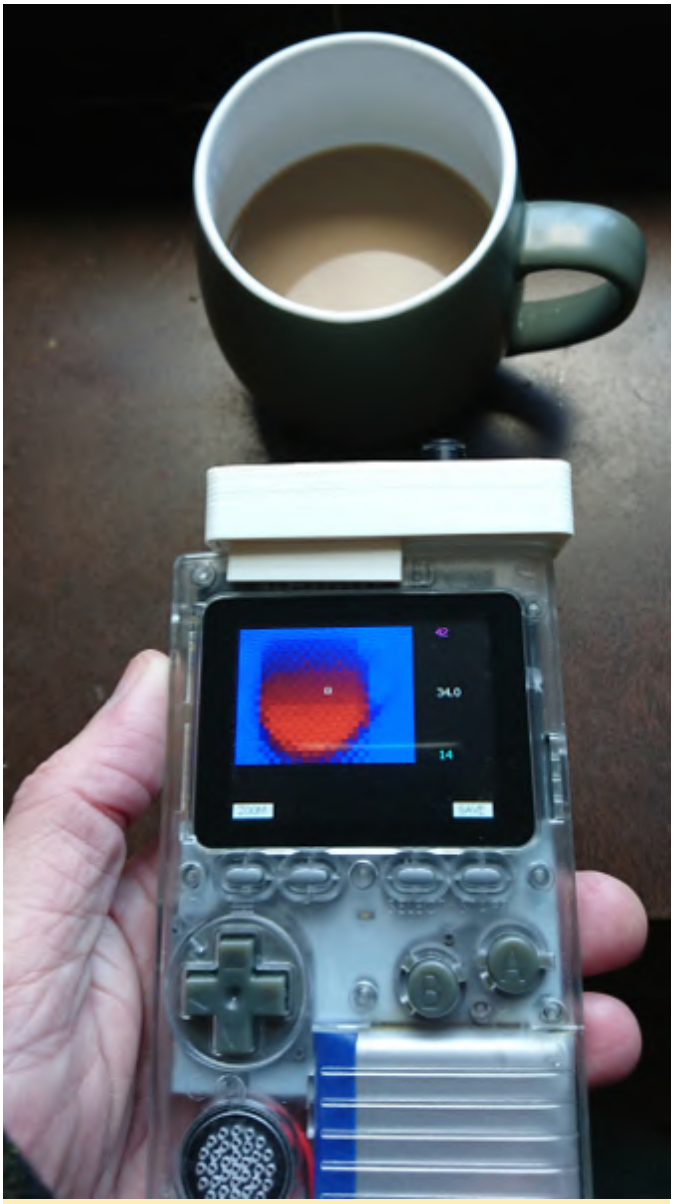


Figura 1: La cámara térmica en acción

Usando el código Arduino

Hay dos formas de usar el código de la cámara térmica. Una forma realmente sencilla es ir a la carpeta “FW file” y copiar el archivo “goircam.fw” a la carpeta de firmware en tu tarjeta SD del ODROID-GO. Luego, cuando enciendas el ODROID-GO, mientras mantienes presionado el botón B, deberías observar una opción para cargar el firmware de la cámara. Obviamente, necesitarás la cámara creada y conectada para que se ejecute el firmware.

La segunda forma, más avanzada, es para las personas que desean utilizar el IDE de Arduino para editar o compilar el firmware como un esquema de Arduino. Para hacer esto, usa los archivos de la carpeta “Arduino code” como lo harías para codificar cualquier otro proyecto de Arduino que tengas. Todos los detalles están disponibles en el sitio Wiki de ODROID-GO para configurar el IDE de Arduino para ODROID-GO.

Si está utilizando el método IDE de Arduino y quieres crear un archivo de firmware, tal y como se describe en la wiki de ODROID-GO, los gráficos para usar con la utilidad MKFW están dentro de la carpeta “Graphics”.

Compilando el módulo de cámara térmica

La compilación de la cámara es muy simple, ya que sólo utiliza el módulo MLX90640, con cables para la puesta a tierra, VCC, SCL y SDA (I2C). Según la Wiki de ODROID-GO, el pinout del cabezal es el siguiente:

```
Header Pin     Function
1       GND
4       SDA
5       SCL
6       VCC
```

Una vez que tenga las cosas conectadas y testeadas, la simplicidad del circuito hace que sea muy fácil soldar algunos pins del cabezal en una pieza del veroboard y así hacer un conector más robusto para el zócalo del cabezal del ODROID-GO. A continuación, se muestra una foto de cómo hice esto yo.

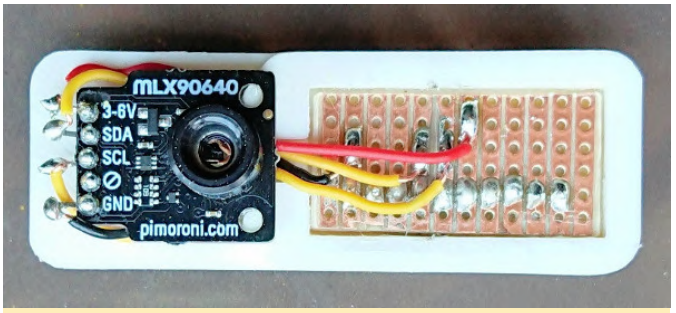


Figura 02: Una foto del circuito

Coloca los pines del cabezal en la parte superior del veroboard y soldé los cables al riel en la parte posterior, lo que significaba que los cables no obstruirían la conexión, aunque también podría haber usado un pedazo más grande de veroboard.

Fabricando una carcasa impresa en 3D

La carpeta “Case 3D model” incluye un par de archivos que usé para hacer una carcasa impreso en 3D. Uno de ellos es un archivo STL listo para imprimir. Este archivo está diseñado para el módulo Pimoroni

MLX90640 que utilicé. Si éste no se adapta a tus necesidades, o al módulo MLX90640 que tienes, tienes el archivo OpenSCAD para que puedas crear una versión personalizada para tu proyecto. El módulo de la cámara encaja, como se muestra en la foto. Esta fotografía también muestra la orientación correcta que debería tener el módulo. Cortando el veroboard con el tamaño correcto, éste encaja perfectamente en la parte posterior de la carcasa, dejando los pines del cabezal correctamente posicionados (ver foto 3). El veroboard está fijado en su sitio y fue necesario comprobar la alineación del pin antes de que se secara el pegamento.

Figura 3: una foto de la parte posterior de la carcasa que muestra los pines del cabezal pegados

Referencias

El Wiki de ODROID-GO es un recurso muy útil para todo lo relacionado con Go: **https://wiki.odroid.com/odroid_go/odroid_go**

La página Wiki de un proyecto LCD I2C 16x2 contiene detalles sobre cómo conectar un módulo I2C al cabezal de ODROID-GO: **https://wiki.odroid.com/odroid_go/arduino/09_16x2lcd_i2c**

La carcasa impresa en 3D también se puede descargar desde Thingiverse: **https://www.thingiverse.com/thing:3648653**

https://github.com/drandrewthomas/Odroid_Go_thermal_IR_camera

RetroELEC para Emulation Station ODROID-XU4, RetroArch y Kodi en una Cómoda Imagen

July 1, 2019 By @escalade Juegos, Linux, ODROID-XU4, Tutoriales

He estado creando compilaciones al estilo Lakka y RetroPie de OpenELEC/LibreELEC durante los últimos años. Recientemente, he comprado un ODROID-XU4 para mi nuevo sistema arcade. Esta distribución de Linux "JeOS" es perfecta para ejecutar emuladores, ya que arranca a la velocidad de la luz y los recursos que utiliza son mínimos.

Esta imagen se inicia directamente en Emulationstation o Kodi/RetroArch. Los emuladores están integrados; simplemente hay que colocar las ROM en el directorio /storage/roms a través de SMB. Los emuladores normalmente detectan automáticamente los controladores, sin embargo, Emulationstation debe configurarse manualmente, excepto DS4 que está preconfigurado. Un teclado puede serte muy útil para llevar a cabo la configuración inicial.

Características

- Basado en LibreELEC
- Compilado desde la fuente con los siguientes CFLAGS: -O2 -march=armv7ve -mtune=cortex-a15.cortex-a7 mcpu=cortex-a15.cortex-a7 -mfloat-abi=hard -mfpu=neon-vfpv4 -flto
- Compilado para GBM KMS/DRM (sin Xserver y sin fbdev)
- El último kernel de Linux 5.0.3 de @memeka
- Soporte para F2FS/BTRFS/XFS
- Los módulos WiFi/Bluetooth de ODROID funcionan "de serie"
- Emulationstation (retroPie fork) es la interfaz por defecto que lanza los emuladores; el valor por defecto se puede cambiar en /storage/.config/frontend.conf y RetroArch también se puede iniciar a través de sus menús
- RetroArch (git master) con función de grabación
- Núcleos de Libretro: desmume, dosbox-svn, fbalpha, mame2003-plus, mame2016, mgba, mupen64plus,

pcsx_rearmed, ppsspp, puae, quicknes, reicast, scummvm, snes9x2010, vice_x64, yeasaniro

- Emuladores independientes: PPSSPP, Dosbox-SDL2, amiberry
- Pulseaudio/BlueZ está configurado para aceptar A2DP (audio bluetooth) para que puedas transmitir música desde tu teléfono u ordenador portátil mientras juegas, simultáneamente
- big.LITTLE cgroups (los emuladores se ejecutan exclusivamente sobre los núcleos más potentes)
- htop optimizado con soporte Big.LITTLE
- IRQs USB asignados a los núcleos más potentes
- Utilidades: scraper tcpdump rsync unrar p7zip cgroup-tools sdl-jstest mediainfo strace screen omxplayer
- Servicios: Docker, Transmission, SABnzbd, Plex (se descarga/actualiza automáticamente e instala a través de la unidad systemd), ttyd (acceso al intérprete de comandos desde un navegador)

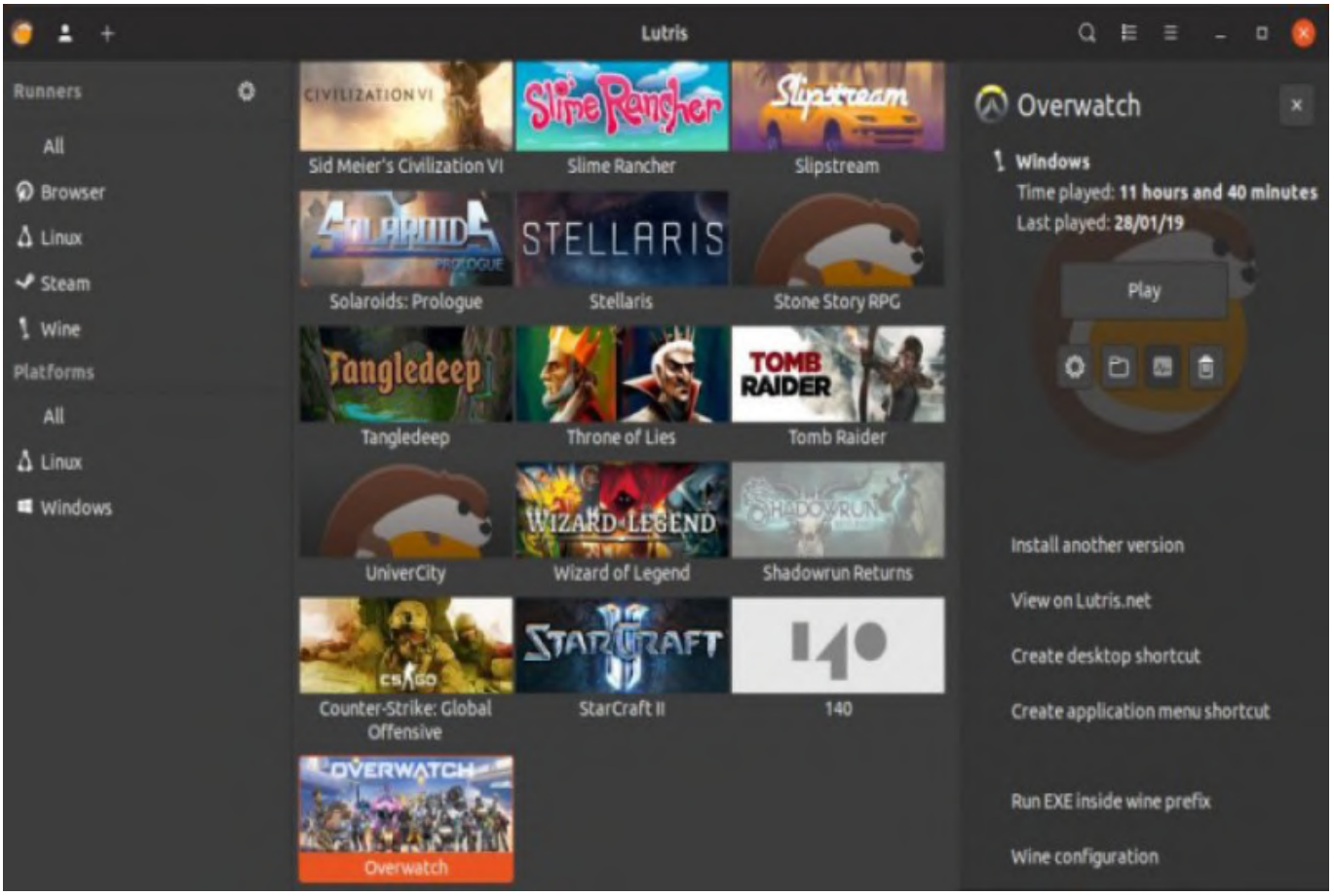
Puedes iniciar los servicios que desees a través de SSH con la siguiente estructura de comandos:

```
$ sudo systemctl enable/start [docker | transmission | sabnzbd | plex | ttyd]
```

Puede descargar RetroELEC desde **https://tinyurl.com/yynfv8m5**. Si quieres que se incluya Kodi, puede localizar las imágenes en una subcarpeta aparte. El código fuente está disponible en Github en **https://github.com/escalade/RetroELEC.tv**. Para comentarios, preguntas y sugerencias, visita el hilo del foro original de ODROID en **https://forum.odroid.com/viewtopic.php?f=96&t=34647**.

Lutris: Juegos en el ODROID-H2

July 1, 2019 By Adrian Popa Juegos, ODROID-H2



Durante aproximadamente veinte años, la comunidad de Linux ha seguido intentando hacer del año actual "el año del escritorio de Linux", un momento mítico en el que la popularidad del escritorio de Linux superaría al de Windows. Desafortunadamente, aún no ha sucedido, pero al menos en un aspecto, Linux está ganando popularidad: el soporte para juegos. Con la llegada de Wine, Steam y Vulkan, cada vez más juegos se pueden jugar en sistemas Linux, liberando a los jugadores del temido ciclo de la muerte de Windows Update. Ten en cuenta que los juegos de Linux no son algo para el usuario principiante de Linux (todavía), aunque los usuarios con experiencia pueden llegar a disfrutar más configurando el entorno que jugando al propio juego.

En este artículo analizaremos brevemente lo que se necesitas para configurar un sistema de juego básico en un ODROID-H2, el cual está disponible en la tienda de Hardkernel en https://www.hardkernel.com/shop/odroid-h2/. Como ya sabes, el ODROID-H2 está basado en la arquitectura Intel y puede ejecutar muchos más juegos que el ODROID-XU4. Veremos hasta dónde podemos llegar. Ten en cuenta que puesto que el ODROID-H2 no es un verdadero "PC de juegos", no esperes milagros, pero puede ejecutar juegos de hace 5 a 10 años con un rendimiento bastante decente. Desafortunadamente, Linux sufrirá una cierta degradación en el rendimiento cuando se ejecuten juegos de Windows, ya que las distintas capas de adaptación añaden la correspondiente sobrecarga al sistema.

Presentando Lutris

Lutris es una plataforma de juegos de código abierto para Linux que puede instalar y lanzar juegos, reduciendo la dificultad de las configuraciones. Puede ejecutar juegos de plataformas online como GoG, Steam o Battle.net, pero también puede gestionar juegos de otras plataformas (desde Amiga a DOS, Windows, Linux nativo o juegos de navegador). La

plataforma tiene varios scripts que se encargan de descargar el juego deseado y de aplicar los parches/cambios necesarios para que se ejecute lo mejor posible. Puedes echar un vistazo a la lista de juegos compatibles de su sitio web https://lutris.net/games/, o buscar un juego en concreto dentro de la aplicación.

Las instrucciones de instalación están disponibles en https://lutris.net/downloads/. Los pasos que se muestran a continuación se han llevado a cabo en una imagen de Ubuntu 19.04 que se ejecuta en ODROID-H2. Las instrucciones en video están disponibles en https://youtu.be/oHDkeQ9eDrc.

```
$ sudo add-apt-repository ppa:lutris-team/lutris
$ sudo apt-get install lutris
```

Cuando estés en el intérprete de comandos, también debes instalar Wine si tienes pensado ejecutar juegos de Windows (yo usé la versión estándar de Ubuntu):

```
$ sudo apt-get install wine
```

Algunos juegos pueden requerir diferentes versiones de Wine (o Wine mejorado con proton), aunque puedes instalarlos desde la sección Wine Runners dentro de Lutris. Cada Wine puede tener su propio entorno (llamado Bottle) para que puedan coexistir múltiples versiones.

Para los juegos de Windows, puede instalar una capa adicional llamada DXVK que realiza la conversión de DirectX 11 a Vulkan. En primer lugar, necesita instalar los drivers Vulkan:

```
$ sudo apt install mesa-vulkan-drivers mesa-vulkan-drivers:i386
```

También puedes instalar vkmark, que es un indicador de rendimiento para Vulkan similar a glmark. Ejecutándolo debería demostrar que el soporte vulkan funciona como es debido en tu sistema Ubuntu. La herramienta necesita ser compilada, así que puede seguir estos pasos:

```
$ sudo apt install meson libvulkan-dev libglm-dev libassimp-dev libxcb1-dev libxcb-icccm4-dev libwayland-dev libdrm-dev libgbm-dev git
$ git clone https://github.com/vkmark/vkmark.git
$ cd vkmark
$ meson build
$ ninja -C build
$ sudo ninja -C build install
```

Ahora puede ejecutar vkmark y ver la demo representada con la API de Vulkan.

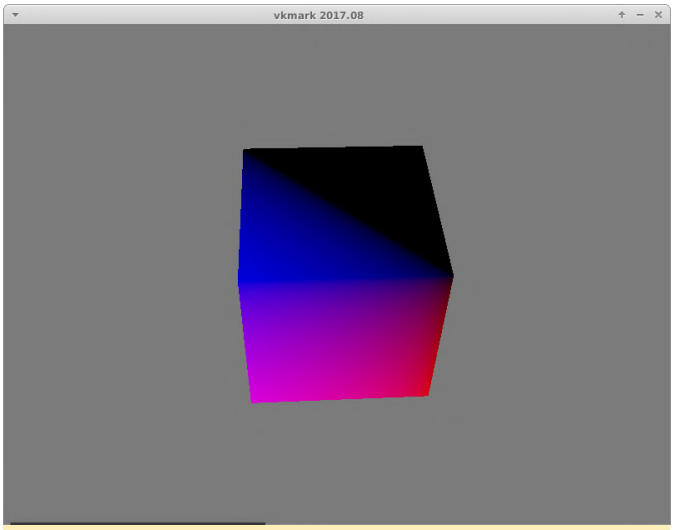


Figura 1 - Vkmark en acción

Ahora, las aplicaciones de Linux que saben cómo usarla pueden utilizar la API de Vulkan. Ahora podemos agregar la capa de conversión DXVK (https://github.com/lutris/lutris/wiki/How-to:-DXVK), que es tan fácil como escribir el siguiente comando:

```
$ sudo apt-get install dxvk
```

Gestión de Lutris

Ahora que tenemos todo configurado, necesitamos añadir juegos a Lutris. Puede iniciar Lutris o navegar por su sitio web, buscar el juego que deseas instalar y hacer clic en Install. También debes gestionar tus “runners”, que son programas externos que son necesarios para ejecutar juegos emulados. Puede hacer clic en el icono de Lutris (arriba a la izquierda) y seleccionar "Manage Runners". Desde la lista, puede activar las plataformas que quieres usar (por ejemplo, DosBox, ScummVM, Wine, etc.). Desde aquí, también puede configurar varias opciones específicas para un “runner” en concreto.

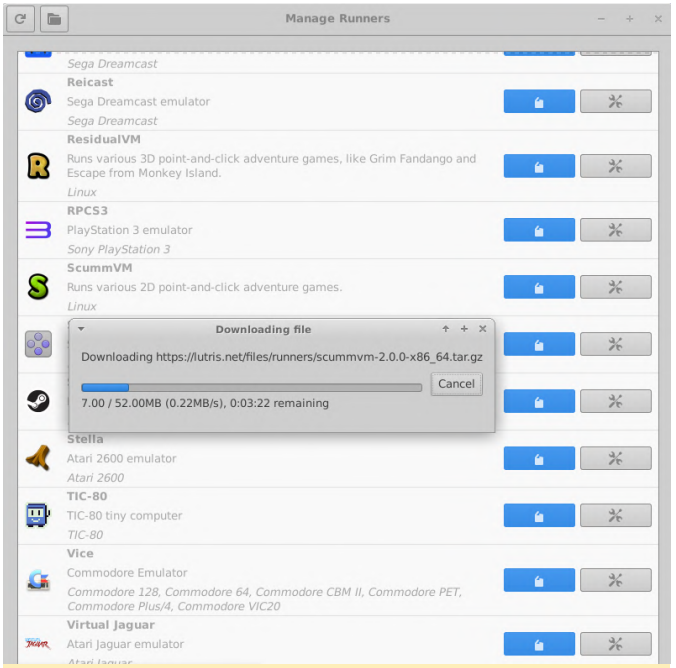


Figura 2 - Gestionar runners

Ahora podemos descargar e instalar algunos juegos. Empecemos con algo simple: un juego nativo de Linux como Super Tux Kart. Puede buscar el título en la barra de búsqueda de Lutris, resáltalo y haz clic en Install. El instalador te hará algunas preguntas, así que responde lo mejor que puedas.

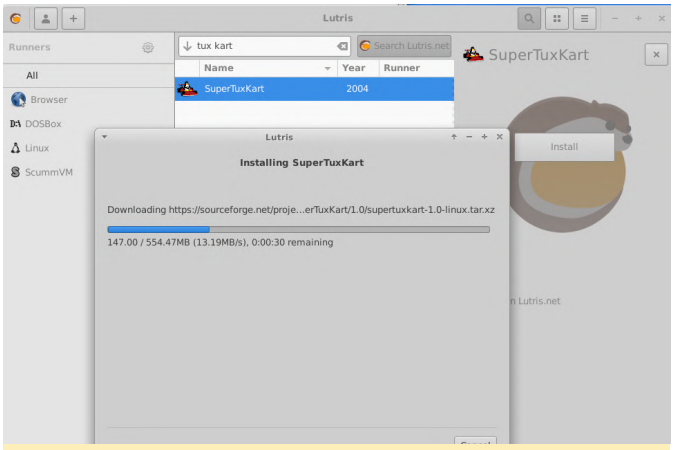


Figura 3 - Super Tux Kart

Una vez que el instalador finalice, puedes iniciar el juego desde el menú.

Figura 4 - Super Tux Kart - Playthrough

Ten en cuenta que, incluso si el juego se ejecuta sin problemas, en el ODROID-H2 algunas texturas pueden verse defectuosas, lo que probablemente se deba a la implementación de los drivers Intel. Al menos la instalación no tiene problemas.

Por ejemplo, para instalar algo de GoG, debe iniciar sesión en GoG a través de Lutris. Primero busca el juego GoG deseado (por ejemplo, Tyrian2000). Se te solicitarán tus credenciales de GoG y, a continuación, el juego se descargará e instalará perfectamente.

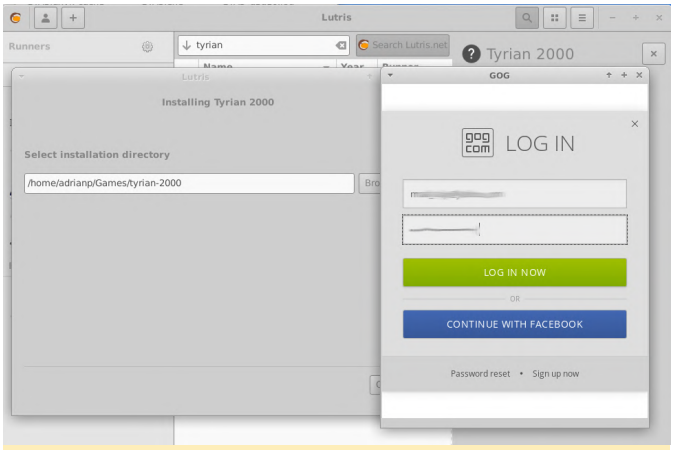


Figura 5 - Instalación de GoG

Intentemos ahora algo más complicado que tu ODROID-XU4 por lo general no puede hacer. Para instalar juegos de Steam (ya sea de Linux o Windows), primero debes instalar el cliente Steam:

```
$ sudo apt-get install steam
```

Puedes instalar los juegos de Steam de dos formas: ya sea a través de Steam o a través de Lutris. Para el primer caso, debes iniciar Steam e iniciar sesión con tu cuenta, entra en tu biblioteca de juegos e instala un

juego en local. Una vez instalado, puede volver a Lutris y agregarlo al lanzador seleccionando "+" -> Import Games -> Steam. Marca el juego/juegos que quieres importar y selecciona Import Games.

Yo Instalé Team Fortress 2 (que es gratis en Steam) directamente desde Lutris, buscándolo en la pestaña de búsqueda de Lutris, haciendo clic en install y esperando a que terminase. El juego funcionaba bien (como era de esperar) bajo Linux.

Puede haber casos en los que tengas una copia local de un juego (de una copia de seguridad o una instalación original) que quiera instalar, o tal vez quieras instalar un juego no compatible (o uno legalmente cuestionado). También lo puedes hacer. En mi caso, quería instalar la versión para Windows de Lucas Chess (https://lucaschess.pythonanywhere.com/). Los pasos son los siguientes:

Descargar el instalador

Dentro de Lutris, selecciona "+" -> Add game. Dale un nombre y selecciona el runner correcto para ese juego. En mi caso, fue el wine

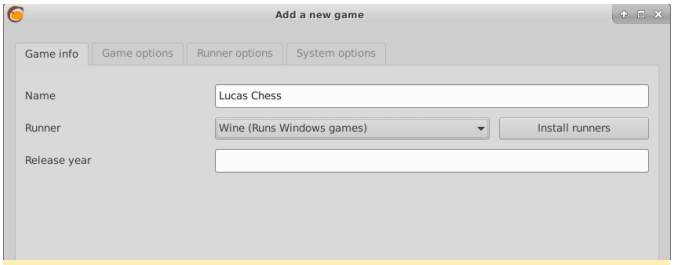


Figura 6 - Añadir un juego manual

En la pestaña Game Options, selecciona el ejecutable del instalador dentro del campo Executable. Probablemente no necesites añadir ningún argumento. El campo Wine prefix generalmente estará fijado como ~/Games/Nombre_del_juego y mantendrá el entorno del wine y los datos del juego. Lo ideal es que no se superponga con otros juegos.

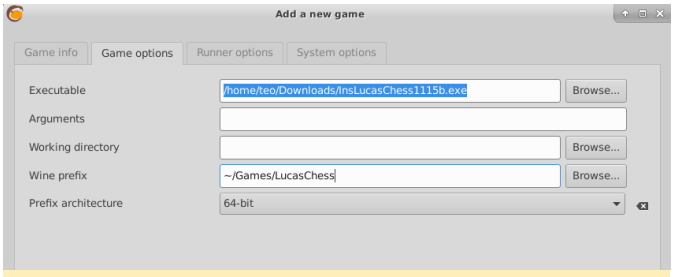


Figura 7 - Opciones de juego

En la pestaña Runner Options, puede seleccionar la versión de wine que quieres utilizar (puede instalar varias versiones de wine desde Manage Runners -> Wine). También puede activar el soporte DXVK si lo necesitas.

Figura 8 - Opciones de Runner

Haga clic en “save” y ahora podrás usar la nueva entrada para ejecutar el instalador. El instalador debe continuar como si estuviera ejecutandose en Windows, y cuando finalice, deberás editar el lanzador del juego (presiona el símbolo de configuración en forma de rueda dentada junto a Play), vuelve a las Opciones de juego y cambiar el archivo ejecutable el juego para que apunte al juego instalado en lugar del instalador (puedes buscar el ejecutable del juego).

Si tienes problemas de fuentes con tu programa, puedes instalar las fuentes que te falten de Windows haciendo clic derecho en el título del juego (o haciendo clic izquierdo y seleccionando desde el panel derecho) y seleccionando Winetricks -> Select the default wineprefix -> Install a font -> allfonts. Llevará un tiempo, pero asegúrate de volver a intentar ejecutar el juego después. También hay configuraciones de Windows que puedes cambiar desde Winetricks, como fontsmooth=rgb por ejemplo.

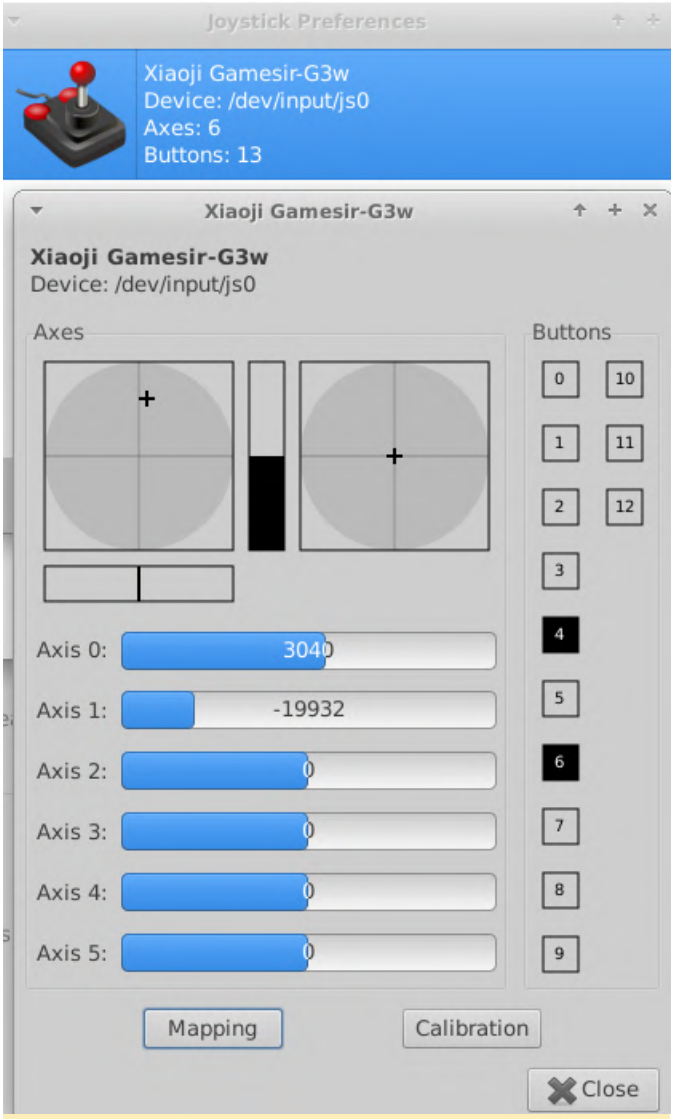


Figura 9 - Ejecutando un juego instalado manualmente

Hice una prueba con GTA V para ver cómo funcionaban los juegos de última generación en el ODROID-H2. Bajo Windows, se ejecuta a unos 20-30 fps con toda la configuración al mínimo y a una resolución de 800x600. Bajo Linux, se ejecuta a unos 15-25 fps con la misma configuración, aunque a veces “tironea”. Así que baja un poco el rendimiento, aun así, se puede jugar decentemente. Para un jugador casual y entusiasta de Linux puede valer la pena. ¡Imagínate lo bien que deben ejecutarse los juegos con una GPU de última generación!

Lograr que los controles funcionen

El mando Gamesir G3w que vende Hardkernel debería funcionar directamente en Windows sin configurar nada. Sin embargo, no estamos en Windows, así que necesitamos hacer algunas modificaciones. Ten en cuenta que el mando tiene dos modos de funcionamiento: Xiaoji Gamesir-G3w (dos LED encendidos) y un modo Xbox 360 (un LED rojo). Puedes cambiar entre los modos manteniendo presionado el botón central "GameSir" durante unos 10 segundos. El modo Xbox 360 es bueno para Android (busca Octopus en la Play Store), pero para Linux necesitas el modo Xiaoji Gamesir-G3w. Puedes verificar su modo actual comprobando este ID USB (ten en cuenta que mi Ubuntu lo reporta como Apple, pero en realidad no lo es):

```
$ lsusb | grep 05ac:055b
Bus 001 Device 007: ID 05ac:055b Apple, Inc.
```

Ahora puedes instalar jstest-gtk para probar los botones:

```
$ sudo apt-get install jstest-gtk
```

Si no recibes respuesta, lo más probable es que tu usuario no forme parte del grupo "input", así que asegúrate de agregarte a ese grupo y volver a iniciar sesión.

Figura 10 - Prueba de joystick

En realidad, utilizar un mando depende del juego/emulador que se use. Algunos juegos (como Need for Speed 2 SE https://github.com/zaps166/NFSIISE) admiten el gamepad como un joystick pre-configurado. Tan solo debes asignar los botones de tu joystick a las acciones del juego.

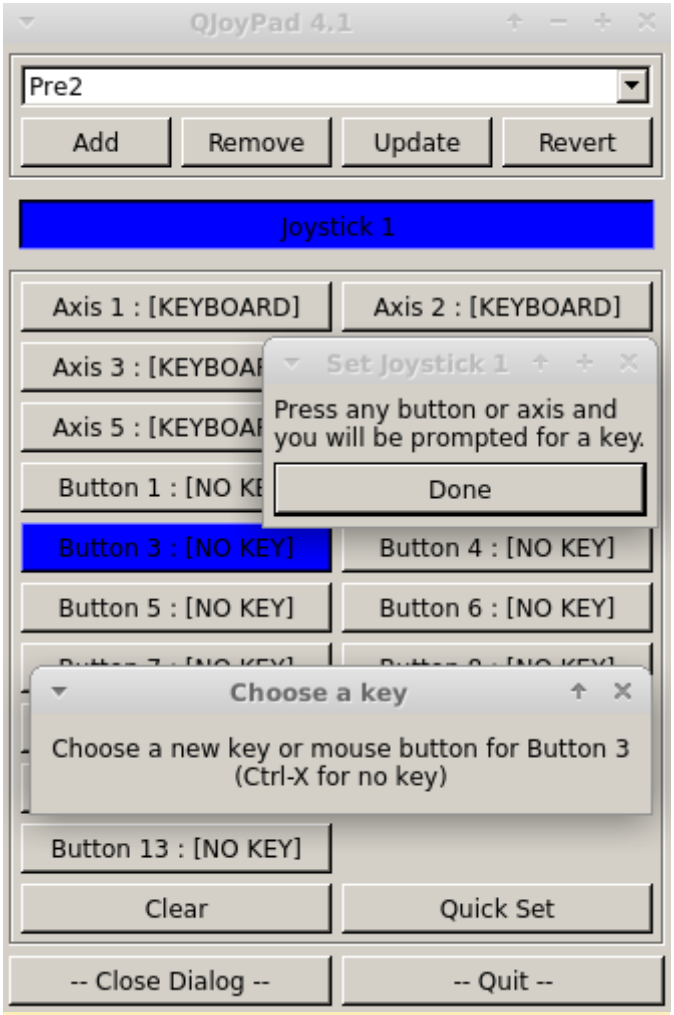


Figura 11 - Soporte de joystick nativo

Para otros juegos, es posible que tengas que asignar eventos de joystick a las teclas. Puedes hacerlo con la aplicación qjoypad:

```
$ sudo apt-get install qjoypad
```

Estoy intentando añadir compatibilidad de mando para un antiguo juego de DOS llamado Prehistorik 2. Solo necesita Izquierda/Derecha/Arriba/Abajo y Espacio, que es el botón de disparo. Para ello, creamos un nuevo perfil llamado "Pre2" (después de iniciar la aplicación, puedes seleccionar la ventana principal desde el icono de la barra de estado). La forma más fácil es seleccionar "Quick set" y se te pedirá que presiones un botón en el mando, seguido de una tecla para emularlo

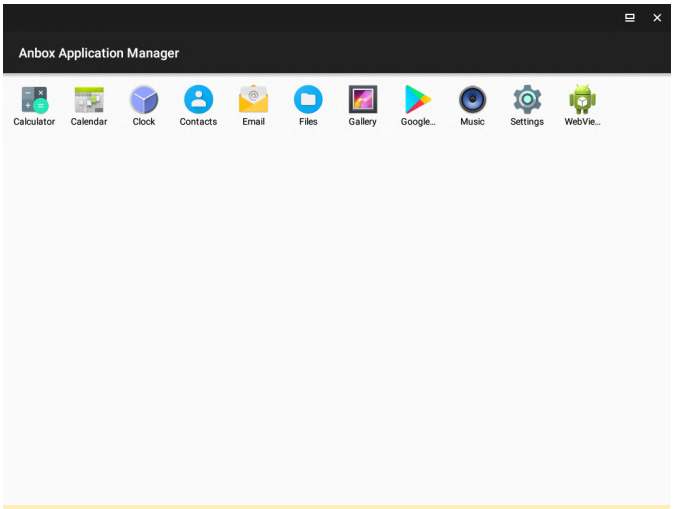


Figura 12 - Mapeo del Joypad

Para concluir, sería mejor ajustar el perfil correcto al iniciar el juego. Puedes hacerlo editando game settings -> System options -> "Show advanced options". Puedes configurar la ruta de acceso a un script (lamentablemente todavía no tiene parámetros) que ejecutará qjoypad con el perfil correcto antes de iniciar el juego. Crea el siguiente script, guárdalo y márcalo como ejecutable:

```
$ cat Games/pre2/qjoypad.sh
#!/bin/bash
/usr/bin/qjoypad Pre2
$ chmod a+x Games/pre2/qjoypad.sh
```

En la lista, busca "Pre-launch command" y agrega "/home/odroid/Games/pre2/qjoypad.sh". ¡Guarda y disfruta! (Muchas gracias a @meveric por su ayuda para hacer funcionar el gamepad)

Android via anbox (Android directo)

¿Qué pasa si tienes algunos juegos de Android a los que quieres jugar? Afortunadamente, no es necesario realizar un arranque dual para Android: ¡puedes ejecutar aplicaciones de Android en un contenedor Linux con anbox!

Ten en cuenta que el proyecto aún es joven y solo ofrece una versión beta con la que poder jugar, de modo que pueden aparecer bloqueos ocasionales, aunque gracias a la magia de Linux y la incorporación de dos módulos del kernel de Android, puede ejecutar una imagen Android 7.1.1 con núcleo x86 sobre la cual puede instalar tus aplicaciones, aunque no todas las aplicaciones llegarán a funcionar. Puedes encontrar las instrucciones de instalación

en https://github.com/anbox/anbox/blob/master/docs/install.md.

```
$ sudo add-apt-repository ppa:morphis/anbox-support
$ sudo apt install anbox-modules-dkms
$ sudo modprobe ashmem_linux
$ sudo modprobe binder_linux
$ sudo snap install --devmode --beta anbox
```

Aparecerá un icono lanzador en tu GUI de linux (tenía el mío en Otros) llamado Android Application Manager que se puede usar para iniciar el lanzador de aplicaciones.

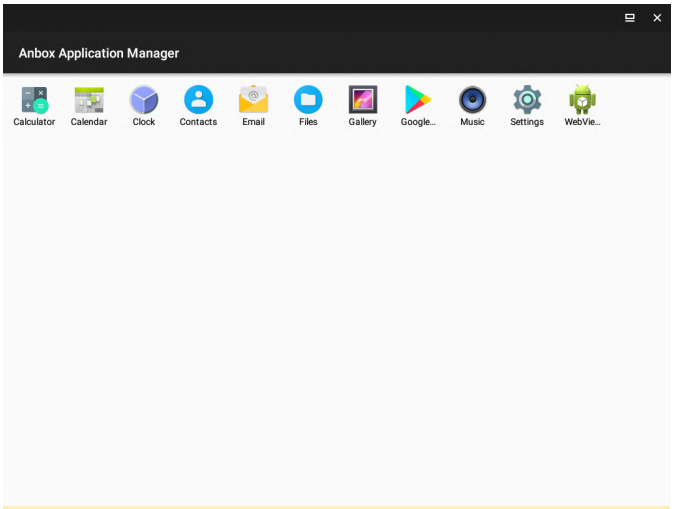


Figura 13 - Administrador de aplicaciones de Android

Ahora que Android lo tienes funcionando, ¿cómo instalamos aplicaciones (supongo que te has cansado de jugar con la Calculadora)? Vamos a instalar Play Store (y soporte para aplicaciones ARM a través de libhoudini) siguiendo la sencilla guía de https://www.linuxuprising.com/2018/07/anbox-how-to-install-google-play-store.html.

```
$ sudo apt-get install git
$ git clone https://github.com/geeks-r-us/anbox-playstore-installer.git
$ cd anbox-playstore-installer/
$ sudo ./install-playstore.sh
$ sudo apt-get install lzip
$ sudo ./install-playstore.sh
```

Asegúrate de proporcionar todos los permisos requeridos por Google Play y Google Services en Android Settings -> Apps -> ... -> Permissions, de lo contrario, te encontrarás con errores extraños.

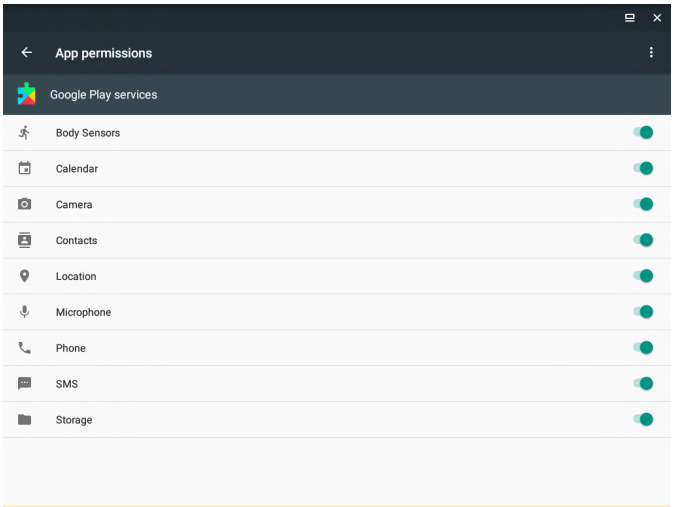


Figura 14 - Todos los permisos

Cuando inicies Play store, te pedirá que inicie sesión con tu cuenta, lo cual debería funcionar en tu caso. En mi caso, intente iniciar sesión con la cuenta de mi hijo, que está vinculada a Google Family (para supervisión de los padres) y el inicio de sesión falló porque el dispositivo emulado carecía de algunas funciones de seguridad requeridas por Google. Sin embargo, hay otra forma de instalar aplicaciones: puedes usar adb para descargar cualquier apk:

```
$ sudo apt-get install adb-tools
$ adb devices
List of devices attached
* daemon not running; starting now at tcp:5037
* daemon started successfully
emulator-5558 device
$ adb install F-Droid.apk
```

Conseguido

Me dirigí a F-Droid, la tienda de aplicaciones alternativa de código abierto (https://f-droid.org/en/), y desde allí logré instalar un simple juego llamado Reckoning Skills (https://f-droid.org/en/packages/org.secuso.privacyfriendlyreckoningskills/) pensado para desafiar las habilidades matemáticas de mi hijo.

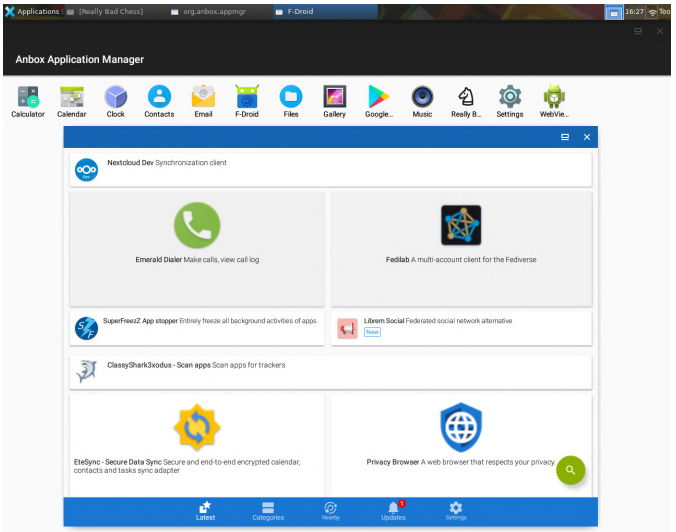


Figura 15 - F-Droid

Si tiene problemas, puedes reiniciar tu dispositivo Android emulado con:

```
$ sudo snap restart anbox
```

Puedes obtener (¡un montón!) de registros log con adb logcat si quieres solucionar algún problema, y si necesitas acceder a la partición "userdata", puede encontrar sus archivos en /var/snap/anbox/common/data/data

Añadir aplicaciones de Android a Lutris

Tener contenido reproducible desde Android eta muy bien, pero tal vez te gustaría tenerlo integrado en un único lanzador. Para hacer esto, necesitas localizar el nombre del paquete de la aplicación deseada (identificador del programa) y la actividad de entrada (la ventana de inicio). Necesitarás conectarte a través de adb mientras la aplicación deseada se esté iniciando en primer plano y ejecutar:

```
$ adb shell
x86_64:/ $ dumpsys window windows | grep -E
'mCurrentFocus'
mCurrentFocus=Window{67dc1f0 u0
org.secuso.privacyfriendlyrecknoningskills/org.sec
uso.privacyfriendlyreckoningskills.activities.Main
Activity}
```

La cadena resaltada en naranja es el nombre del paquete de la aplicación actual, mientras que la cadena azul es el nombre de la actividad. Con esta información, ahora puede iniciar esta aplicación de Android desde un shell (será mejor que los copies/pegues en lugar de escribirlos):

```
$ anbox launch --
package=org.secuso.privacyfriendlyrecknoningskills
--
component=org.secuso.privacyfriendlyreckoningskill
s.activities.MainActivity
```

Ahora puedes añadir una entrada manual para tu juego bajo Lutris. Selecciona el runner nativo de Linux y añade /snap/bin/anbox como ejecutable. Deberás agregar launch --package = org.secuso.privacyfriendlyrecknoningskills --component = org.secuso.privacyfriendlyreckoningskills.activities.MainActivity en la entrada de parámetros (sin comillas). Guarda y disfruta de tu nuevo lanzador.

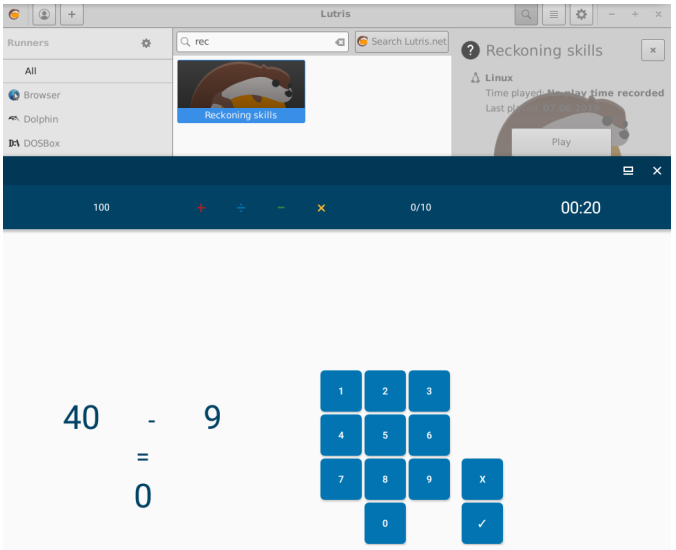


Figura 16 - Lanzando un juego de Android desde Lutris

Ten en cuenta que incluso si la configuración actual es defectuosa o tediosa, anbox mejora con cada versión liberada, espero que su integración con Lutris mejore en el futuro. Espero que esta guía sobre el mundo de los juegos en Linux te haya ayudado a empezar. Si te quedas atascado en el algún momento, puedes pedir ayuda en el hilo de soporte en https://forum.odroid.com/viewtopic.php?f=172&t=35311&p=258763#p258763.