Decodificación Hardware • Servidor NTP • Ordenador Control • Minería GPU

ODROID Magazine

Año Cinco
Num. #51
Mar 2018

EL DESCUBRIMIENTO DE LOS NUMEROS PRIMOS

USA UN ODROID-C2 PARA HACER UN HISTORIAL MATEMATICO

ODROID-NI: PRIMERA PRUEBA DE RENDIMIENTO DEL FUTURO

Domótica con Home Assistant

March 1, 2018

Home Assistant es una plataforma de domótica de código abierto desarrollada en Python 3 que soporta más de 650 componentes

OID-N1 vs ODROID-XU4: Una Comparativa de Rendimiento en la Vida Real

March 1, 2018

Siguiendo su línea de trabajo en materia de innovación, Hardkernel acaba de anunciar su última oferta de SBC, el ODROID-N1, basado en el SOC Rockchip RK3399. Aquí tienes nuestra comparativa con el ODROID-XU4

El Descubrimiento de los Números Primos: Usar un ODROID-C2 para hacer un historial matemático

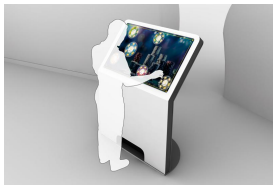
March 1, 2018

“Se sabe que el problema de distinguir números primos de números compuestos y de resolver estos últimos en sus factores primarios es conocido como uno de los más importantes y útiles en aritmética.” − Carl Friedrich Gauss En este artículo, proporcionare una introducción sobre algunos de los aspectos algorítmicos de ▶️

Juegos ODROID: Juegos Saturn – Parte 2

March 1, 2018

Una vez más, volvemos al tema de los juegos ODROID-XU3/XU4 y Sega Saturn



Web Kiosk: Cómo crear un Sistema con Pantalla Táctil basado en Chromium

March 1, 2018

Estaba buscando una plataforma que me permitiera reunir varias funcionalidades de control remoto en un único dispositivo/interfaz

clInfo: Compilando la Utilidad de Puesta a Punto de la GPU OpenCL Esencial para el ODROID-XU4

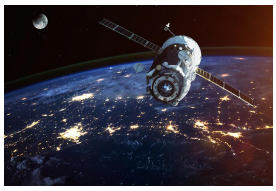
March 1, 2018

He estado indagando por qué Clinfo no funciona en el ODROID-XU4, así que he dedicado un tiempo para descubrir el por qué

Buscadores, Mineros y 49: Minería GPU-CPU Dual en el ODROID-XU4/MC1/HC1/HC2

March 1, 2018

Hay muchas personas que usan XU4/MC1/HC1/HC2 para la minería de criptomonedas por CPU



Creando un servidor NTP usando GPS/PPS

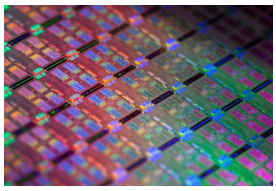
March 1, 2018

Puedes montar tu propio servidor Network Time Protocol (NTP) usando GPS y PPS en tu ODROID. Este sistema te proporciona la hora exacta que puede serte muy útil para casos específicos. Como resultado, nuestro servidor local puede tener una hora muy precisa con una tolerancia de menos de 10 microsegundos.

Empezando con Android en el ODROID-C2: Una Guía para principiantes

March 1, 2018

Hay dos opciones para instalar Android en un ODROID-C2. Hardkernel ofrece un módulo eMMC o tarjeta microSD preinstalados, que solo necesitan la instalación de Google Play. Otra posibilidad es descargar el sistema operativo Android desde el sitio web de Hardkernel e instalarlo manualmente en el módulo eMMC o tarjeta microSD



Cómo Activar la Decodificación por Hardware en el ODROID-C2

March 1, 2018

Un repositorio git que tiene soluciones destinadas a ayudar al usuario a habilitar la decodificación por hardware en el ODROID-C2. Para todos los que estén lidiando con este problema, clonar el repositorio y seguir los siguientes pasos.

Ordenador de Control ODROID-XU4: Creando un Sistema de Control Todo en Uno

March 1, 2018

No había un sistema Linux embebido que tuviera la suficiente potencia informática como para ejecutar grandes filtros de partículas a un coste razonable, y que también tuviese los sensores necesarios (GPS, IMU) de una calidad razonable integrados, así que williamg42 decidió crear uno.

Conociendo un ODROIDian: Go Sang “Luke” Chul (Luke.go)

March 1, 2018

Conozce a “Luke”, ingeniero de software de Hardkernl y encargado de actualizar la versión de Android para todos los dispositivos ODROID excepto LineageOS para ODROID-XU4. Principalmente actualiza las revisiones, añade funciones y corrige errores en la versión oficial de Android de Hardkernel

Domótica con Home Assistant

March 1, 2018 By Adrian Popa ODROID-C1+, ODROID-C2, ODROID-XU4

Llega un momento en la vida en el que quieres poner en orden las cosas y al mismo tiempo disponer de un acceso simple para soluciones complejas. Por ejemplo, quizás tengas varios scripts que se encargan de diferentes cuestiones (como encender y apagar un calentador, tomar fotografías con tus cámaras de seguridad, gestionar la detección de presencia, etc.), pero eres el único que puedes administrar estos scripts porque requieren de un mantenimiento a través de SSH, o a través de alguna página web obsoleta. Yo también he llegado a ese momento en mi vida y tengo que buscar una solución "global" que me permita administrar todas mis automatizaciones privadas y que ofrezca un fácil acceso para mi familia.

Estaba pensando en crear un panel de control web que se ajustara a mis necesidades, pero detesto el desarrollo web. Soy un poco vago y mis sitios no son para nada atractivos. Además, necesitaba que funcionara en todo tipo de dispositivos y tamaños de pantalla, y que también estuviera preparado para el futuro. Afortunadamente, pasé el suficiente tiempo buscando hasta que encontre la solución perfecta: Home Assistant (**http://bit.ly/2hlOPOE**) – HA para abreviar.

Home Assistant es una plataforma de domótica de código abierto desarrollada en Python 3 que soporta más de 650 componentes, que son módulos que facilitan la interacción con diversas cosas como switches físicos "inteligentes", relés, luces, sensores, dispositivos de red (televisores, routers y cámaras), software (como Kodi, MPD y Transmisión), servicios de red (como el tiempo), pero también permite agregar tus propios componentes personalizados. Las principales marcas y tecnologías de automatización del hogar, como Hue, Nest, IKEA, Vera, ZigBee y MQTT están presentes, puedes encontrar una lista completa de los componentes en **http://bit.ly/2sWJsPy**.

Además de los componentes, la plataforma cuenta con una interfaz web muy similar a un cuadro de

mandos y un motor de automatización donde puedes combinar datos de diferentes componentes y generar un evento. Por ejemplo, si de lunes a viernes entre las 8:00 y las 15:00 el tiempo esta soleado, la temperatura exterior supera la 30 °C, no hay probabilidad de lluvia, y los aspersores de fuera han estado apagados durante al menos 4 horas, entonces activaremos los aspersores unos 20 minutos. Lo complicado de esta automatización es disponer de una forma de encender y apagar los aspersores; el resto lo proporciona los componentes y el motor de automatización de Home Assistant. Otros posibles usos pueden ser bloquear y desbloquear la puerta de entrada cuando una persona específica se conecte a la red wifi (aunque yo no lo haría personalmente) o iniciar el aire acondicionado automáticamente cuando el sistema detecte que estás llegando a casa del trabajo. Tienes más ejemplos en el video de 1 hora en **http://bit.ly/2t0GgCI**. Si estás familiarizado con Tasker para Android o IFTTT, Home Assistant es el equivalente para tu hogar.

Instalación

Puede instalar Home Assistant en cualquier dispositivo ODROID. Dependiendo de la cantidad de automatizaciones que tengas pensado tener, puedes usar un C1 para una configuración liviana, o incluso un XU4 para casas más grandes y reglas más complejas que pueden incluir el reconocimiento facial. Yo lo estoy usando en un C2 que funciona como reproductor Kodi sin problema.

Vamos a hacer una instalación "virtualenv", lo cual significa que todos los módulos necesarios de python se instalarán en un directorio específico y no interferirán con los módulos del sistema. Además utilizaremos un usuario distinto para Home Assistant. También hay imágenes Docker disponibles. Todas las instrucciones con comentarios están disponibles en **http://bit.ly/2t0iaYC**.

```
$ sudo apt-get update
$ sudo apt-get dist-upgrade
$ sudo apt-get install python-pip python3-dev
$ sudo pip install --upgrade virtualenv
$ sudo adduser --system homeassistant
$ sudo addgroup homeassistant
$ sudo usermod -G dialout -a homeassistant
$ sudo mkdir /srv/homeassistant
$ sudo chown homeassistant:homeassistant
/srv/homeassistant
$ sudo su -s /bin/bash homeassistant
$ virtualenv -p python3 /srv/homeassistant
$ source /srv/homeassistant/bin/activate
(homeassistant)$ pip3 install --upgrade
homeassistant
$ exit
```

Para iniciar y administrar el proceso, es mejor crear un servicio systemd para gestionarlo:

```
$ sudo vi
/etc/systemd/system/homeassistant.service
[Unit]
Description=Home Assistant
After=network.target time-sync.target
Requires=time-sync.target

[Service]
Type=simple
User=%i
ExecStart=/srv/homeassistant/bin/hass -c
"/home/homeassistant/.homeassistant"

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

Para iniciar Home Assistant, simplemente inicia su servicio:

```
$ sudo service homeassistant start
$ sudo service homeassistant enable
```

Ten en cuenta que, si vas a utilizar componentes que necesitan HTTPS, deberás tener la hora configurada correctamente en el arranque, para que los certificados sean válidos. El inicio del servicio depende de systemd-timesyncd, que a su vez depende de que ntp *no* esté instalado:

```
$ sudo apt-get remove ntp
$ sudo service systemd-timesyncd restart
$ sudo systemctl enable systemd-timesyncd
```

En caso de tener problemas, podrás revisar los registros log a través de journalctl:

```
$ sudo journalctl -u homeassistant -f
```

Una vez que se inicie el proceso, podrás conectarte a http://ip-odroid:8123/. Ten en cuenta que el primer arranque (o un inicio después de una actualización) puede ser algo más lento, así que déjelo trabajar durante unos minutos hasta que accedas a la interfaz web. Home Assistant cuenta con una aplicación nativa para IOS (**http://apple.co/2tYi2WI**), mientras que para los clientes de Android puedes configurar la página de HA como página de inicio (Chrome -> navega a http://ip-odroid: 8123 – > Menu -> Add to homescreen).

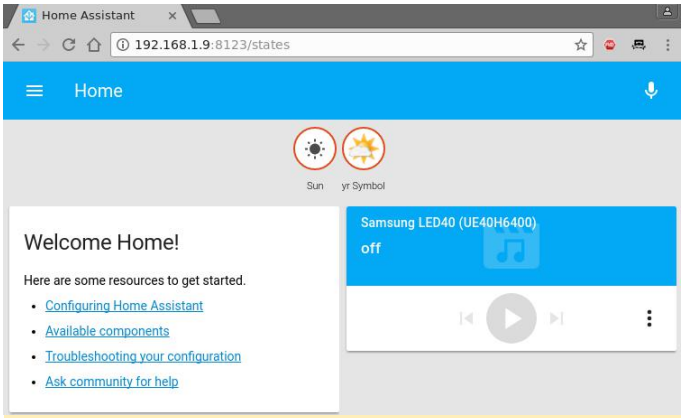


Figura 1 – Página de inicio de Home Assistant

El archivo de configuración

Iara poner en marcha los componentes y configurar tu instalación, tendrás que trabajar bastante con los archivos de configuración de Home Assistant. Es de esperar que, en una versión futura, puedas modificar la configuración directamente desde la interfaz web, pero por ahora, necesitarás un editor de texto. El archivo principal es /home/homeassistant/.homeassistant/configuration.yaml. Su formato es YAML, que significa "Yet Another Markup Language". Al igual que Python, usa espacios en blanco (¡no tabulaciones!) para delimitar las secciones de código. Por defecto, utiliza una sangría de dos espacios para las secciones anidadas. En el caso de que haya problemas, recibirás mensajes de error al iniciar el servicio. Puede validar la sintaxis con un servicio como **http://www.yamllint.com/** que te permitirá saber dónde están exactamente los errores. También tienes una guía de solución de problemas en **http://bit.ly/2tDHMsa**.

Una vez que hayas realizado los cambios en el archivo de configuración, deberás reiniciar el servicio de Home Assistant para que se apliquen los cambios. Puedes hacerlo desde el intérprete de comandos con "sudo service homeassistant restart", o desde la interfaz web de HA, haciendo clic en el icono de la parte superior izquierda, seleccionando el icono de "Configuración" y llamando a la opción "Restart" de la sección "Server Management". El video **http://bit.ly/2sAmD3F** muestra algunos consejos que deberías tener en cuenta a la hora de editar la configuración.

```
homeassistant:
  # Name of the location where Home Assistant is running
  name: Home
  # Location required to calculate the time the sun rises and sets
  latitude: 44.7
  longitude: 26.45
  # Impacts weather/sunrise data (altitude above sea level in meters)
  elevation: 69
  # metric for Metric, imperial for Imperial
  unit_system: metric
  # Pick yours from here: http://en.wikipedia.org/wiki/List_of_tz_database_time_zones
  time_zone: Europe/Bucharest

# Show links to resources in log and frontend
introduction:

# Enables the frontend
frontend:

http:
  # Uncomment this to add a password (recommended!)
  # api_password: PASSWORD
  # Uncomment this if you are using SSL or running in Docker etc
  # base_url: example.duckdns.org:8123

# Checks for available updates
updater:

# Discover some devices automatically
discovery:

# Allows you to issue voice commands from the frontend in enabled browsers
conversation:

# Enables support for tracking state changes over time.
history:

# View all events in a logbook
logbook:

# Track the sun
sun:

# Weather Prediction
sensor:
  platform: yr

# Text to speech
tts:
  platform: google
```

Figura 2 – La configuración por defecto

Si tienes la intención de usar HA desde fuera de la LAN (por ejemplo, desde Internet), tiene varias opciones. Una de ellas es habilitar la compatibilidad con HTTPS y redireccionar el puerto 8123 en tu router. Esto te proporciona encriptación, pero expone tu instalación a Internet (y puede haber vulnerabilidades que permitan a posibles atacantes tomar el control de tu sistema/LAN). Una segunda opción (que es la que yo prefiero) es configurar una VPN en tu router (o incluso en tu ODROID) que te permita conectarte y acceder a HA (y a otros recursos LAN) de forma segura.

Si quieres utilizar HTTPS, para que tengas disponible todas la funciones, necesitas certificados SSL válidos (no autofirmados). Para obtener certificados válidos, deberás tener un nombre DNS público (por ejemplo, utilizar un servicio DNS dinámico como duckdns.org) y

usar letsencrypt.org para configurar un certificado SSL válido para tu instalación. Los detalles paso a paso los puedes encontrar en el video http://bit.ly/2tY6LGb. Si necesitas utilizar certificados autofirmados, hay una guía disponible en http://bit.ly/2t0ObzH.

Independientemente del mecanismo de acceso (http o https), querrás configurar una contraseña. HA no es compatible con múltiples cuentas de usuario, pero puedes fijar una contraseña API que necesitarás para iniciar sesión en la interfaz web. La mejor forma de hacerlo es creando un archivo que almacene todos tus datos confidenciales (como contraseñas y URL), asígnale el nombre "secrets.yaml" y haz referencia a él dentro del archivo configuration.yaml.

```
$ cat
/home/homeassistant/.homeassistant/secrets.yam
l api_password: odroid
$ cat
/home/homeassistant/.homeassistant/configurati
on.yaml
 …
 http:
 api_password: !secret api_password
 …
```

Ahora, cuando reinicies HA, se te pedirá una contraseña. Puedes encontrar más detalles sobre este tema en http://bit.ly/2rLGEkV.

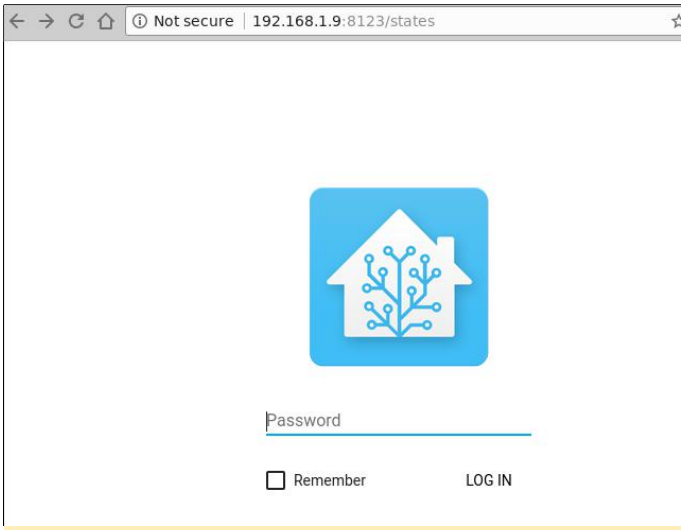


Figura 3 – Autentificación

Para conocer cómo funciona la configuración de HA, vamos a instalar algunos componentes. Voy a configurar el tiempo, algunas cámaras IP, Kodi y MPD, detección de presencia basada en WiFi y también un sensor de temperatura 1-wire conectado al ODROID.

El tiempo desde Darksky

Existen multitud de proveedores de información meteorológica ya integrados en HA (http://bit.ly/2t4l1Rh), para que puedas elegir tu favorito. Me quedo con DarkSky (http://bit.ly/2t4gq0S), que proporciona pronósticos bastante precisos de mi zona. Deberías consultar la página de ayuda del componente para conocer los detalles de su configuración y qué variables son las que puedes utilizar. Tendrás que registrarse en Dark Sky y obtener una clave API que te permitirá hacer 1000 peticiones al día de forma gratuita. Lo mejor es guardar esta clave API dentro de tu archivo secrets.yaml (reemplaza con tu propia clave):

```
darksky_api_key:
87f15cbb811204412cc75109777ea5cf
```

La configuración tiene varias variables, la mayoría de las cuales son opcionales, sin embargo, en configuration.yaml, debajo de la sección sensor deberías tener lo siguiente (tomate la libertad de eliminar la entrada "platform: yr"):

```
sensor:
 - platform: darksky
 api_key: !secret darksky_api_key
 name: Dark Sky
 monitored_conditions:
 - summary
 - precip_type
 - precip_probability
 - temperature
 - humidity
 - precip_intensity
 - wind_speed
 - pressure
 - wind_bearing
 - apparent_temperature
 - icon
 - minutely_summary
 - hourly_summary
 - temperature_max
 - temperature_min
 units: si
 update_interval: '00:15'
```

El código es en su gran mayoría autoexplicativo. Éste configura una nueva plataforma del tipo “darksky”, con un nombre específico (opcional) y una api_key (obligatoria), y extrae un conjunto de parámetros (monitored_conditions) desde proveedor de información meteorológica cada 15 minutos. Tu ubicación exacta se toma de los parámetros de latitud/longitud en Home Assistant, de modo que debes asegurarte de que sean correcta. Tras reiniciar el servicio homeassistant, deberías poder ver las variables monitorizadas en forma de insignias en la parte superior de la ventana. Al hacer clic en alguna, te mostrara cómo ese valor en particular ha cambiado con el tiempo.

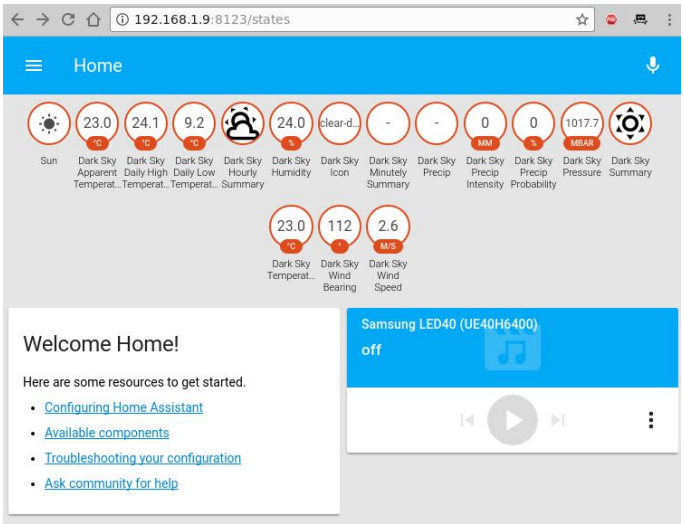


Figura 4 – Datos meteorológicos

Visualización de cámaras IP

HA soporta muchas cámaras (http://bit.ly/2t4DtsD),incluida la interpretación de datos desde un archivo, que podría utilizarse para presentar un gráfico, o datos visuales generados por otras herramientas. Usaremos el componente Generic MJPG Camera (http://bit.ly/2t4tIKM) y el componente Local File (http://bit.ly/2s4Y5w4)º.

La cámara que queremos monitorizar está disponible en http://bit.ly/2t4cHkc (es una webcam pública), que debemos añadir al archivo secrets.yaml.

```
camera1_stream_url:
http://iris.not.iac.es/axis-
cgi/mjpg/video.cgi?resolution=320x240
camera1_still_url:
http://iris.not.iac.es/jpg/image.jpg
```

La parte de la configuración dentro de configuration.yaml debería parecerse a esto para ambos sistemas de cámara:

```
camera:
 - platform: mjpeg
 mjpeg_url: !secret camera1_stream_url
 still_image_url: !secret camera1_still_url
 name: Observatory in Spain
 - platform: local_file
 file_path: /tmp/tux.jpg
```

Como de costumbre, deberás reiniciar el servicio de HA para volver a leer la configuración (este también podría ser un buen momento para comentar el componente de “introducción”). Tenga en cuenta que cuando haces clic en una webcam, verás una transmisión en vivo, de lo contrario, la imagen fija se actualiza cada 10 segundos.

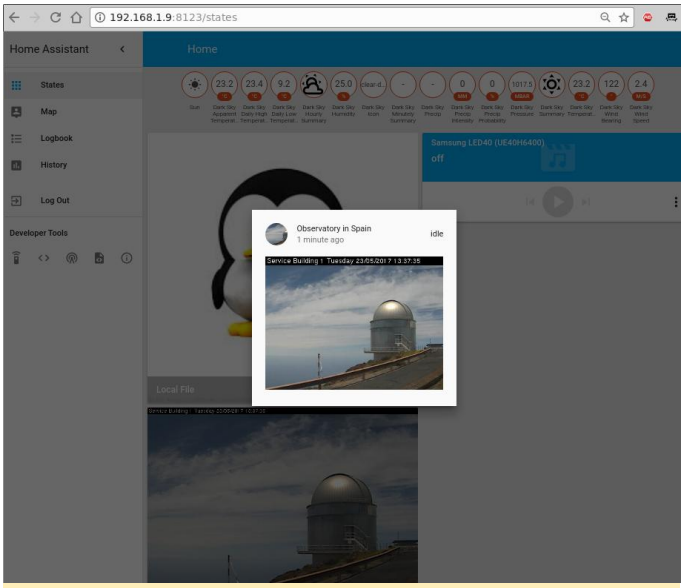


Figura 5 – ¡Webcams!

Entonces, ¿Qué podemos hacer con estas webcams configuradas aparte de mirarlas? Bueno, puede usarlas con otros componentes como OpenCV (http://bit.ly/2s4UUEJ) para generar “activadores” (señales que acciones un determinado proceso) cuando se vean determinados rostros, o Seven Segments Display (http://bit.ly/2sAbOP0), que permite tomar lecturas de varias pantallas digitales.

Kodi y MPD

Para configurar reproductores multimedia, puedes buscar en la lista de componentes Media Player en http://bit.ly/2s0IAtQ. Para configurar Kodi

(http://bit.ly/2sA5qr6),deberás activar la opción "Allow remote control via HTTP" (http://bit.ly/2t4cYne) y configurar primero el correspondiente nombre de usuario y contraseña. Para hacerlo, añade el usuario y la contraseña al archivo secrets.yaml:

```
kodi_user: kodi
kodi_pass: kodi
```

A continuación, edita configuration.yaml:

```
media_player:
- platform: kodi
host: 192.168.1.140
name: Kodi Livingroom
username: !secret kodi_user
password: !secret kodi_pass
```

Para configurar MPD, suponiendo que ya tienes un servidor MPD en tu red, agrega el componente MPD (http://bit.ly/2s5sbzE) y añade la contraseña a secrets.yaml:

```
mpd_secret: mpd
```

Y después, edita configuration.yaml:

```
media_player:
...
- platform: mpd
host: 192.168.1.140
name: MPD Living
password: !secret mpd_secret
```

Tras reiniciar Home Assistant, aparecerán los dos nuevos reproductores multimedia y podrá ver su estado (reproducción/detención), controlar el volumen e incluso cambiar la lista de reproducción actual o usar el componente de conversión texto-voz para que el reproductor multimedia "pronuncie" lo que quieras.

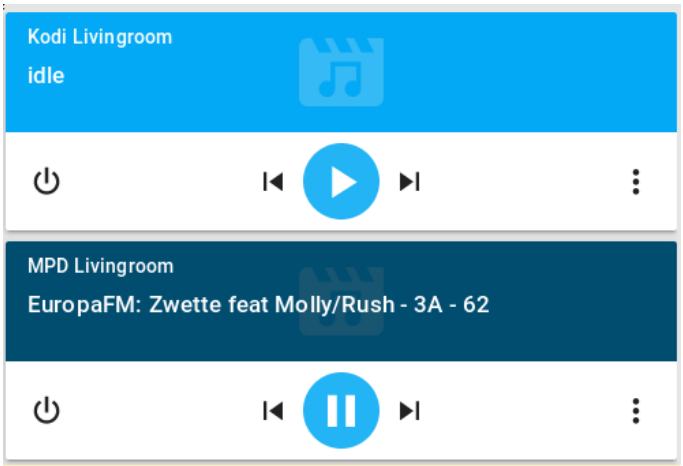


Figura 6 – Reproductores multimedia

Detección de presencia

Los componentes de detección de presencia (http://bit.ly/2t0Gt8H) intentan rastrear las ubicaciones de las personas para que puedas aplicar reglas de geofencing (por ejemplo, hacer algo si una persona entra o sale de una ubicación). Por lo general, el seguimiento se realiza mediante la detección de dispositivos conectados a un router (a través de wifi), mediante la proximidad del bluetooth (http://bit.ly/2s0Sqfw) o haciendo uso de servicios de ubicación como Owntracks (http://bit.ly/2rLQdR1).

Usaremos un rastreador basado en routers que, dependiendo de tu router, se conecta periódicamente a la interfaz de gestión de tu router, lista la tabla ARP y descubre qué dispositivos están conectados. Son muchos los tipos de routers soportados, desde despóticos de gama alta como Cisco, hasta router de consumo como Asus, Netgear y TP-Link. Incluso son compatibles los firmwares de código abierto, como OpenWRT, DD-WRT y Tomato.

Utilizaremos un router Asus con SSH habilitado, de modo que necesitamos el componente ASUSWRT: http://bit.ly/2s4T32Q. Puedes elegir entre iniciar sesión con nombre de usuario/contraseña o configurar una clave SSH e iniciar sesión. Ten en cuenta que determinadas versiones del firmware cuentan con ciertas medidas de seguridad que limitan la cantidad de conexiones SSH y pueden incluir tu IP en una lista negra si se hacen muchas conexiones.

Como de costumbre, colocaremos los datos sensibles (como la ruta a la clave o la contraseña ssh) dentro del archivo secrets.yaml:

```
router_user: admin
router_password: my_secret_password
```

Dentro de configuration.yaml añade la siguiente sección:

```
device_tracker:
- platform: asuswrt
host: 192.168.1.1
username: !secret router_user
password: !secret router_password
interval_seconds: 120
consider_home: 300
track_new_devices: yes
```

La página de configuración del rastreador del dispositivo (**http://bit.ly/2s4WPcA**) proporciona más detalles sobre las opciones que puede usar. La opción interval_seconds es el tiempo entre lecturas (2 minutos) y la opción consider_home lo mantiene "en home" incluso si tus dispositivos no aparecen durante 300 segundos.

Una vez que reinicies HA y después de realizar la primera busqueda exitosa, se creará un nuevo archivo, llamado known_devices.yaml. Aquí podrás asignar un nombre amigable e incluso una imagen a un dispositivo específico, o hacer que otros dispositivos sean ignorados.

Una entrada en known_devices.yaml debería verse así:

```
aldebaran:
hide_if_away: false
mac: 00:1E:06:31:8C:5B
name: aldebaran
picture: /local/aldebaran.png
track: true
vendor: WIBRAIN
```

Ten en cuenta que ha añadido una ruta a la imagen local que se almacena en /home/homeassistant/.homeassistant/www/aldebaran.png. Puedes crear la carpeta "www" con el siguiente comando:

```
$ sudo mkdir
/home/homeassistant/.homeassistant/www
```

Si hay dispositivos que no deseas monitorizar, puede fijar "track: false" en el archivo known_devices.yaml.

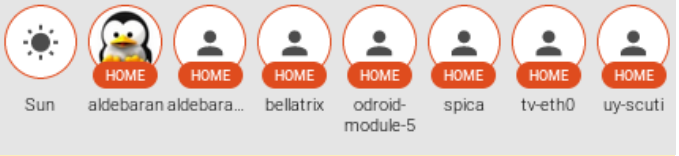


Figura 7 – Busqueda inicial/entradas personalizadas

Medición de la temperatura

Una característica muy potente de Home Assistant es la capacidad de rastrear todo tipo de sensores (**http://bit.ly/2cNb4gJ**). Queremos monitorizar un sensor de temperatura basado en el protocolo 1-Wire, conectado localmente al ODROID (**http://bit.ly/2s12ZPx**). Antes de agregar el sensor a HA, asegúrate de que sea legible desde la línea de comando. Puedes recurrir a la guía de configuración de la wiki en **http://bit.ly/2s0zbTp**.

Necesitaras conocer el ID del sensor para añadirlo a HA:

```
$ ls /sys/bus/w1/devices/
28-0516866e14ff w1_bus_master1
$ cat /sys/bus/w1/devices/28-
0516866e14ff/w1_slave
92 01 4b 46 7f ff 0c 10 b5 : crc=b5 YES
92 01 4b 46 7f ff 0c 10 b5 t=25125
```

A continuación, realiza los siguientes cambios en configuration.yaml y sondea el sensor cada 5 minutos:

```
sensor:
...
- platform: onewire
names:
28-0516866e14ff: Living room
scan_interval: '00:05'
```

Tras reiniciar HA, la nueva lectura será visible en la interfaz web a modo de insignia en la parte superior de la página.

Ordenando las vistas

Observarás que una vez que empiezas a añadir componentes, la interfaz web se muestra algo desordenada con gran cantidad de elementos dispersos por todas partes. Puedes usar grupos y vistas para limpiar la interfaz y ubicar los elementos

relacionados en tus propias pestañas. Para entender todo esto, vamos a tirar un poco de vocabulario.

Las "Entities" (Entidades) son variables que proporcionan datos, como un sensor o interruptor. Las plataformas (como dark_sky) generalmente proporcionan acceso a múltiples entidades (temperaturas mínimas/máximas o pronóstico). Puede ver una lista de entidades, sus nombres y su valor si navega en la interfaz web por Developer tools -> States (<>) -> Entities.

Un "group" (grupo) es simplemente un objeto que contiene una lista de entidades. Visualmente, un grupo se presenta como un panel o una ficha. Por defecto, existe el grupo "group.all_devices" que almacena los elementos descubiertos por una plataforma de rastreo de dispositivos. Los grupos generalmente contienen una lista de entidades.

Una "view" (vista) se presenta como una pestaña separada dentro de Home Assistant. Las vistas son en realidad grupos de grupos y se diferencian de los grupos normales por tener la propiedad "view: yes". También puedes añadir a una vista entidades individuales, así como grupos.

Agruparemos nuestros sensores existentes en las siguientes categorías:

- hLa primera pestaña es Home y contiene los siguientes grupos (denominada internamente default_view, de modo que aparece cuando inicias sesión):
  - Datos meteorológicos
  - Información de presencia
  - Información del sistema (para mostrarte si hay actualizaciones disponibles)

- La segunda pestaña se llamará Media y contiene los siguientes grupos:
  - Reproductores multimedia

- La pestaña final se llamará Images y contiene:
  - Webcams

La configuración debería parecerse a la siguiente lista

```
group:
default_view:
view: yes
entities:
- group.weather
- group.presence
- group.systeminfo
media:
view: yes
entities:
- group.mediaplayers
images:
view: yes
entities:
- camera.observatory_in_spain
- camera.local_file
weather:
name: Weather
entities:
- sensor.dark_sky_apparent_temperature
- sensor.dark_sky_daily_high_temperature
- sensor.dark_sky_daily_low_temperature
- sensor.dark_sky_hourly_summary
- sensor.living_room
presence:
name: Presence
entities:
- device_tracker.aldebaran
- device_tracker.nutty
systeminfo:
name: System Info
entities:
- updater.updater
mediaplayers:
name: Media Players
entities:
- media_player.mpd_livingroom
- media_player.kodi_livingroom
```

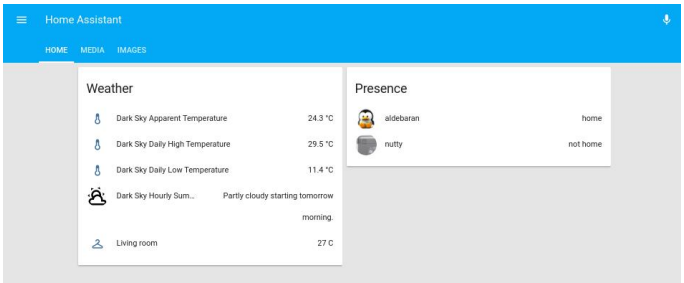


Figura 8 – Una interfaz más limpia con vistas y grupos

Tienes disponible más detalles sobre los grupos y la distribución en el video de http://bit.ly/2s5d6xT.

Actualizaciones

Como Home Assistant no ha sido instalado a través de apt-get, deberás gestionar manualmente las actualizaciones. Antes de actualizar, es conveniente leer las notas de la versión y verificar que la actualización no dañe la configuración creada, ya que a veces se suele rediseñar la configuración de los nuevos componentes, lo que significa que tendrías que volver a hacerla. Puede obtener notificaciones de nuevas versión utilizando la entidad updater.Updater, que comprueba periódicamente si hay versiones nuevas y puede mostrarlas en Home Assistant. Las actualizaciones son bastante frecuentes, y puede aparecer una versión principal cada 2-3 semanas. El procedimiento de actualización es simple, y los detalles los puedes encontrar en **http://bit.ly/2s0Kn24**.

```
$ sudo service homeassistant stop
$ sudo su -s /bin/bash homeassistant
$ source /srv/homeassistant/bin/activate
(homeassistant)$ pip3 install --upgrade
homeassistant (homeassistant)$ exit
$ sudo service homeassistant start
```

En artículos posteriores, analizaremos la configuración de algunos componentes más complejos, como un relé remoto o una unidad de aire acondicionado, configuraremos automatizaciones y montaremos un cuadro de mandos. Para comentarios, preguntas y sugerencias, visita el hilo de soporte en **http://bit.ly/2s13GbB**.

OID-N1 vs ODROID-XU4: Una Comparativa de Rendimiento en la Vida Real

March 1, 2018 By @hominoid ODROID-XU4, ODROID-N1

Siguiendo su línea de trabajo en materia de innovación, Hardkernel acaba de anunciar su última oferta de SBC, el ODROID-N1, basado en el SOC Rockchip RK3399 (http://goo.gl/2BpMuQ). Aquí tienes una rápida y prematura comparativa del ODROID-N1 con su actual buque insignia, el ODROID-XU4. Ten en cuenta que el ODROID-N1 que ha sido probado es un prototipo de ingeniería y no un producto final totalmente terminado. Ejecuta Debian con un kernel 4.4 provisional, no ha habido suficiente tiempo para afinar completamente el sistema operativo o los algoritmos de criptografía, y otros componentes importantes. Sin embargo, se han observado algunos resultados muy interesantes.

La comparativa consistía en un único ODROID-N1 y un ODROID-XU4 minando (stratum server) Verium (VRM) en un contexto de frecuencias sostenible. A una temperatura ambiente de 71 0F (21.66 0C), el ODROID-XU4 funcionando a 1.7Ghz mantuvo una temperatura media de 70 ° C y mientras que el ODROID-N1 a 1.99Ghz nunca vio que su temperatura excediera de los 51 ° C. El ODROID-N1 se sentía como pez en el agua.

Las herramientas utilizadas han sido:

- odroid-cpu-control
- cpuminer-fireworm

Los resultados que aparecen a continuación han sido formateados para facilitar su lectura

Resultados del ODROID-N1

```
odroid@odroid-n1:~$ uname -a
Linux odroid-n1 4.4.112 #2 SMP Thu Feb 8
21:25:35 -02 2018 aarch64 GNU/Linux

odroid@odroid-n1:~$ odroid-cpu-control -l
```

```
CPU0: governor ondemand current 408.00MHz
 min 408.00MHz [408.00MHz]  max 1.51GHz
[1.51GHz]
CPU1: governor ondemand current 408.00MHz
 min 408.00MHz [408.00MHz]  max 1.51GHz
[1.51GHz]
CPU2: governor ondemand current 408.00MHz
 min 408.00MHz [408.00MHz]  max 1.51GHz
[1.51GHz]
CPU3: governor ondemand current 408.00MHz
 min 408.00MHz [408.00MHz]  max 1.51GHz
[1.51GHz]
CPU4: governor ondemand current 408.00MHz
 min 408.00MHz [408.00MHz]  max 1.99GHz
[1.99GHz]
CPU5: governor ondemand current 408.00MHz
 min 408.00MHz [408.00MHz]  max 1.99GHz
[1.99GHz]

odroid@odroid-n1:~$ ~/cpuminer-fireworm -o
stratum+tcp://stratum.poolsloth.com:3333 -u
xxxx -p xxxx --randomize --no-redirect -t 9

 Verium Miner forked from cpuminer 1.4
{fireworm} by fireworm@github **
              credits to tpruvot et al. &
effectsToCause et al. **

[2018-02-18 18:31:05] Starting Stratum on
stratum+tcp://stratum.poolsloth.com:3333
[2018-02-18 18:31:05] HugePages unavailable
(22)

[2018-02-18 18:31:05] 9 miner threads started,
using scrypt algorithm.
[2018-02-18 18:31:09] Stratum difficulty set
to 0.025
[2018-02-18 18:31:09]
stratum.poolsloth.com:3333 scrypt² block
181936
[2018-02-18 18:32:39] Total: 538.110 H/m
[2018-02-18 18:32:59] accepted: 1/1 (100.00%),
0.00837 kH/s yes!
[2018-02-18 18:33:10] Total: 479.410 H/m
[2018-02-18 18:33:43] Total: 530.087 H/m
[2018-02-18 18:35:03] Total: 512.673 H/m
[2018-02-18 18:35:10] accepted: 2/2 (100.00%),
0.00822 kH/s yes!
[2018-02-18 18:36:04] Stratum difficulty set
to 0.0171756
[2018-02-18 18:36:39] Total: 534.344 H/m
[2018-02-18 18:37:08] accepted: 3/3 (100.00%),
0.00829 kH/s yes!
[2018-02-18 18:37:31] accepted: 4/4 (100.00%),
0.00932 kH/s yes!
[2018-02-18 18:38:09] Total: 558.247 H/m
[2018-02-18 18:39:40] Total: 536.414 H/m
[2018-02-18 18:41:00] accepted: 5/5 (100.00%),
0.00915 kH/s yes!
[2018-02-18 18:41:02] Total: 537.398 H/m
[2018-02-18 18:41:21] accepted: 6/6 (100.00%),
0.00825 kH/s yes!
[2018-02-18 18:42:40] Total: 555.318 H/m
[2018-02-18 18:44:06] Total: 533.703 H/m
[2018-02-18 18:44:48] accepted: 7/7 (100.00%),
0.00930 kH/s yes!
^C
[2018-02-18 18:45:08] SIGINT received, exiting
```

Resultados del Odroid XU4

```
root@c3n0:~# uname -a
Linux c3n0 4.14.5-92 #1 SMP PREEMPT Mon Dec 11
15:48:15 UTC 2017 armv7l armv7l armv7l
GNU/Linux

root@c3n0:~# odroid-cpu-control -l
CPU0: governor performance
    current 1.40GHz min 200.00MHz [200.00MHz]
max 1.40GHz [1.40GHz]
CPU1: governor performance
    current 1.40GHz min 200.00MHz [200.00MHz]
max 1.40GHz [1.40GHz]
CPU2: governor performance
    current 1.40GHz min 200.00MHz [200.00MHz]
max 1.40GHz [1.40GHz]
CPU3: governor performance
    current 1.40GHz min 200.00MHz [200.00MHz]
max 1.40GHz [1.40GHz]
CPU4: governor performance
    current 1.70GHz min 200.00MHz [200.00MHz]
max 1.70GHz [2.00GHz]
CPU5: governor performance
    current 1.70GHz min 200.00MHz [200.00MHz]
max 1.70GHz [2.00GHz]
CPU6: governor performance
    current 1.70GHz min 200.00MHz [200.00MHz]
max 1.70GHz [2.00GHz]
CPU7: governor performance
    current 1.70GHz min 200.00MHz [200.00MHz]
max 1.70GHz [2.00GHz]

root@c3n0:~# ~/cpuminer-fireworm -o
stratum+tcp://stratum.poolsloth.com:3333 -u
```

```
xxxx -p xxxx --randomize --no-redirect -t 4 -1
2 --cpu-affinity-stride 1 --cpu-affinity-
default-index 4 --cpu-affinity-oneway-index 0

 Verium Miner forked from cpuminer 1.4
{fireworm} by fireworm@github **
          credits to tpruvot et al. &
effectsToCause et al. **

[2018-02-18 18:31:05] Starting Stratum on
stratum+tcp://stratum.poolsloth.com:3333
[2018-02-18 18:31:05] Binding thread 0 to cpu
index 0
[2018-02-18 18:31:05] Binding thread 1 to cpu
index 0
[2018-02-18 18:31:05] HugePages unavailable
(22)

[2018-02-18 18:31:05] Binding thread 2 to cpu
index 0
[2018-02-18 18:31:05] Binding thread 3 to cpu
index 0
[2018-02-18 18:31:05] 6 miner threads started,
using 'scrypt²' algorithm.
[2018-02-18 18:31:05] Binding thread 4 to cpu
index 0
[2018-02-18 18:31:05] Binding thread 5 to cpu
index 0
[2018-02-18 18:31:09] Stratum difficulty set
to 0.025
[2018-02-18 18:31:09]
stratum.poolsloth.com:3333 scrypt² block
181936
[2018-02-18 18:31:43] Total: 388.377 H/m
[2018-02-18 18:32:14] Total: 387.199 H/m
[2018-02-18 18:32:45] Total: 387.127 H/m
[2018-02-18 18:33:16] Total: 384.155 H/m
[2018-02-18 18:33:47] Total: 385.000 H/m
[2018-02-18 18:34:18] Total: 385.126 H/m
[2018-02-18 18:34:49] Total: 384.142 H/m
[2018-02-18 18:35:20] Total: 383.299 H/m
[2018-02-18 18:35:51] Total: 383.115 H/m
[2018-02-18 18:36:22] Total: 384.423 H/m
[2018-02-18 18:36:54] Total: 385.171 H/m
[2018-02-18 18:37:25] Total: 385.309 H/m
[2018-02-18 18:37:35] accepted: 1/1 (100.00%),
0.00640 kH/s yes!
[2018-02-18 18:37:39] accepted: 2/2 (100.00%),
0.00639 kH/s yes!
[2018-02-18 18:37:44] accepted: 3/3 (100.00%),
0.00639 kH/s yes!
[2018-02-18 18:37:56] Total: 383.180 H/m
[2018-02-18 18:38:27] Total: 382.897 H/m
[2018-02-18 18:38:58] Total: 382.540 H/m
[2018-02-18 18:39:29] Total: 383.798 H/m
[2018-02-18 18:40:00] Total: 383.192 H/m
[2018-02-18 18:40:31] Total: 383.481 H/m
[2018-02-18 18:41:02] Total: 383.795 H/m
[2018-02-18 18:41:33] Total: 384.514 H/m
[2018-02-18 18:42:04] Total: 383.588 H/m
[2018-02-18 18:42:35] Total: 383.282 H/m
[2018-02-18 18:43:07] Total: 382.776 H/m
[2018-02-18 18:43:38] Total: 383.951 H/m
[2018-02-18 18:44:09] Total: 384.540 H/m
[2018-02-18 18:44:13] accepted: 4/4 (100.00%),
0.00642 kH/s yes!
[2018-02-18 18:44:17] Stratum difficulty set
to 0.0169173
[2018-02-18 18:44:29] accepted: 5/5 (100.00%),
0.00642 kH/s yes!
[2018-02-18 18:44:40] Total: 385.162 H/m
^C
[2018-02-18 18:45:04] SIGINT received, exiting
```

El índice de hash medio para el ODROID-N1 fue de 531.57 H/m y 384.35 H/m para el ODROID-XU4. Esto indica que hay un aumento del 38.3% en dicho índice para el ODROID-N1 en operaciones reales. Pasé un tiempo relativamente corto pasando rápidamente por un montón de combinaciones de hilos de ejecución y núcleos en el ODROID-N1, de modo que es probable que exista un cierto margen de mejora. Aunque el ODROID-N1 ejecutaba 9 hilos de 3 vías durante la prueba, he llegado a tener éxito ejecutando 24 hilos de 1 vía. No lo intentado con una cantidad mayor de hilos de 1 vía porque el rendimiento empezaba a deteriorarse. Esto simplemente demuestra la flexibilidad y la ventaja de tener 4 GB de RAM (memoria). En el futuro, se podría realizar una prueba con hilos de ejecución de 6 vías para profundizar más en esta cuestión. Para que quede constancia, aunque el ODROID-XU4 se ejecutaba a 2Ghz, la tasa de hash era inferior e insostenible.

```
root@c3n0:~# odroid-cpu-control -s -M 2.0G
CPU0: max 1.40GHz [1.40GHz] -> 1.40GHz
[1.40GHz]
CPU1: max 1.40GHz [1.40GHz] -> 1.40GHz
[1.40GHz]
CPU2: max 1.40GHz [1.40GHz] -> 1.40GHz
[1.40GHz]
CPU3: max 1.40GHz [1.40GHz] -> 1.40GHz
```

```
[1.40GHz]
CPU4: max 1.70GHz [2.00GHz] -> 2.00GHz
[2.00GHz]
CPU5: max 1.70GHz [2.00GHz] -> 2.00GHz
[2.00GHz]
CPU6: max 1.70GHz [2.00GHz] -> 2.00GHz
[2.00GHz]
CPU7: max 1.70GHz [2.00GHz] -> 2.00GHz
[2.00GHz]

root@c3n0:~# ~/cpuminer-fireworm -o
stratum+tcp://stratum.poolsloth.com:3333 -u
xxxx -p xxxx --randomize --no-redirect -t 4 -1
2

 Verium Miner forked from cpuminer 1.4
{fireworm} by fireworm@github **
              credits to tpruvot et al. &
effectsToCause et al. **

[2018-02-18 20:37:32] Starting Stratum on
stratum+tcp://stratum.poolsloth.com:3333
[2018-02-18 20:37:32] HugePages unavailable
(22)

[2018-02-18 20:37:32] 6 miner threads started,
using 'scrypt²' algorithm.
[2018-02-18 20:37:36] Stratum difficulty set
to 0.025
[2018-02-18 20:37:36]
stratum.poolsloth.com:3333 scrypt² block
181963
[2018-02-18 20:37:42] accepted: 1/1 (100.00%),
0.00429 kH/s yes!
[2018-02-18 20:38:31] accepted: 2/2 (100.00%),
0.00642 kH/s yes!
[2018-02-18 20:38:48] Total: 356.060 H/m
[2018-02-18 20:39:58] Total: 357.322 H/m
[2018-02-18 20:41:01] Total: 353.908 H/m
[2018-02-18 20:41:02] accepted: 3/3 (100.00%),
0.00590 kH/s yes!
[2018-02-18 20:41:32] accepted: 4/4 (100.00%),
0.00611 kH/s yes!
[2018-02-18 20:42:12] Total: 347.295 H/m
^C
[2018-02-18 20:42:18] SIGINT received, exiting
```

Otro buen punto de referencia para la comparativa es la hoja en la que se comparan los índices de hash del hardware de KaptainBlaZzed para VRM en **http://goo.gl/hrYs2Q**. En la segunda hoja, accediendo por la pestaña de la parte inferior, hay una comparativa de SBC. Una vez más, todo está en orden. El índice de hash alcanzado por mi ODROID-XU4 es en solo minería (get-work no es un stratum server) y tiene una tasa de hash sostenible de 1.7Ghz. Si alguien ha mejorado su sistema de refrigeración, tiene un mejor sistema operativo o ha ajustado el algoritmo criptográfico, posiblemente vea mejores tasas de hash. En el extremo derecho de la hoja de cálculo, puede ver los resultados de Hashes/Watt, que también aportan una pista sobre la eficiencia de los SBC. Uno número importante que falta es el Hashes/Dollar (Inversión de capital). Es otra área en la que los SBC ODROID por lo general están en o muy cerca de la cima.

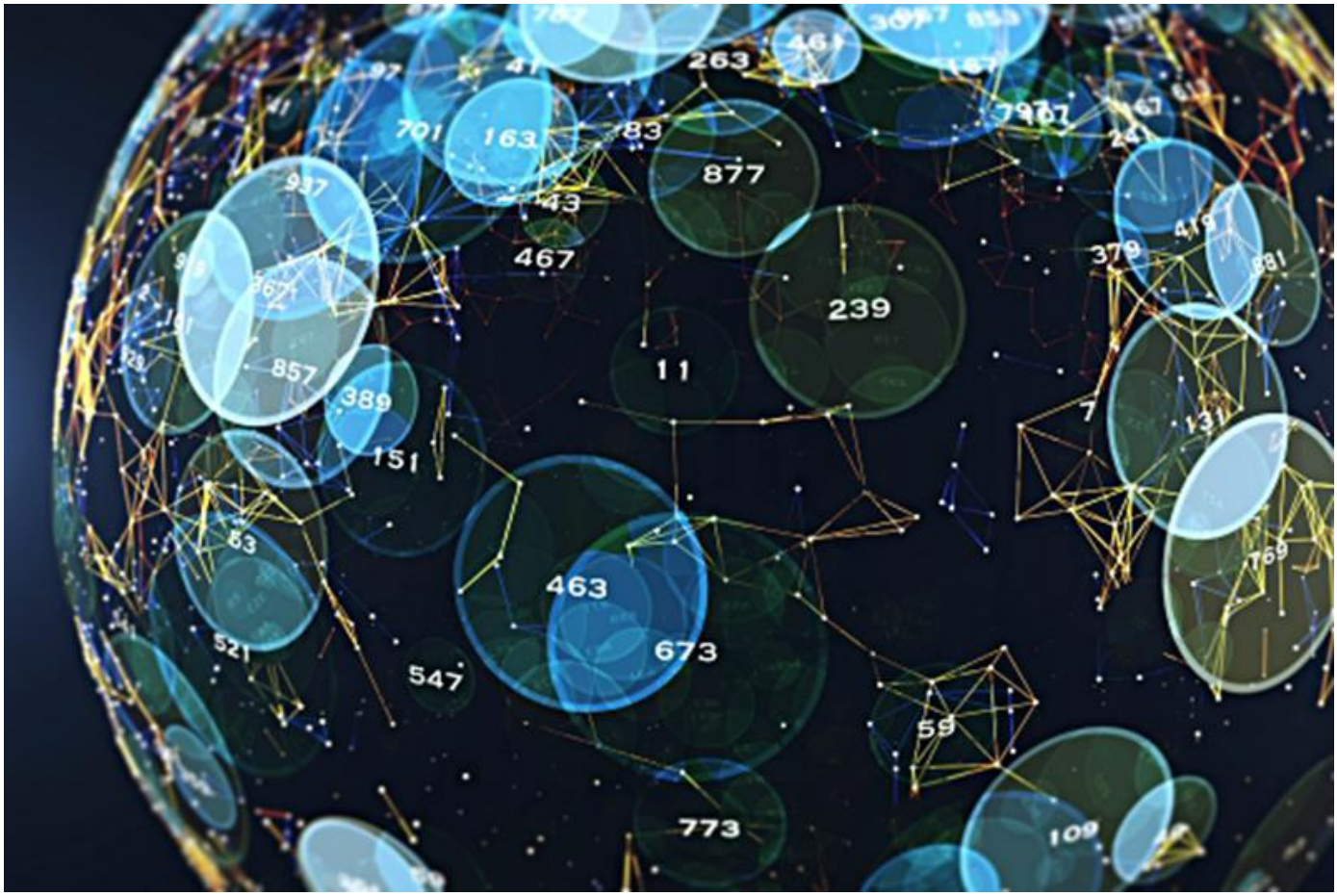
Observaciones

Parece que la memoria PoP podría estar afectando a los perfiles térmicos del sistema. De momento y al menos, en base a los perfiles térmicos del ODROID-XU4 y los que son como el ODROID-MC1, éstos parecen ser más rentables que el ODROID-N1 para plataformas de minería.

Para comentarios, preguntas y sugerencias, visita el hilo original de los foros ODROID en **https://forum.odroid.com/viewtopic.php?f=149&t=30174**. Para obtener información adicional sobre el próximo ODROID-N1 y las actualizaciones sobre la fecha de lanzamiento para su producción, visita el hilo del foro en **https://forum.odroid.com/viewtopic.php?f=149&t=29932**.

El Descubrimiento de los Números Primos: Usar un ODROID-C2 para hacer un historial matemático

March 1, 2018 By Ernst Mayer ODROID-C2, Mathematics



"Se sabe que el problema de distinguir números primos de números compuestos y de resolver estos últimos en sus factores primarios es conocido como uno de los más importantes y útiles en aritmética." − Carl Friedrich Gauss

En este artículo, proporcionare una introducción sobre algunos de los aspectos algorítmicos de las pruebas de primalidad, los ejemplificaré utilizando la utilidad de Linux bc y describiré algunos de los algoritmos más avanzados utilizados en la famosa prueba de primalidad Lucas-Lehmer (LL) para los números de Mersenne y la propia implementación del autor en su programa Mlucas. Este software tiene una versión optimizada para la funcionalidad de hardware vector-aritmética disponible en la familia de procesadores ARMv8, específicamente en el SBC ODROID-C2. Sin embargo, ten en cuenta que el software también se puede compilar en SBC ODROID que no son v8, aunque no utilizan las instrucciones vectoriales. Puesto que la página LEAME de Mlucas (vinculada más abajo) proporciona instrucciones detalladas de compilación para una amplia variedad de plataformas de hardware, incluidas las SBC ODROID, me voy a centrar en las matemáticas y los algoritmos, a un nivel que debería ser comprensible para cualquier persona con conocimientos básicos de álgebra y programación de computadoras.

Raíces primitivas y primalidad

LL es un ejemplo de lo que se conoce como *prueba de primalidad no factorial*, que hace referencia al hecho de que no se necesita conocimiento alguno sobre la factorización en primos del número de entrada *N*, o módulo, aunque normalmente realizamos una "factorización de prueba" pre-tamizada a fin de verificar los números para los pequeños factores primos antes de recurrir a la prueba LL. Tales pruebas

se basan en propiedades algebraicas muy intensas de campos numéricos que están fuera del alcance de este artículo, pero en esencia equivalen a probar si existe una *raíz primitiva* del *grupo* matemático definido por el módulo de multiplicación *N*, lo que significa una raíz del máximo orden posible *N*−1. (Esto suena a matemáticas complejas, pero en realidad se reduce a términos muy simples, como ya veremos más adelante). Tal raíz existe si y solo si (es decir, lo contrario) *N* es primo. Por ejemplo, si calculamos las potencias sucesivas de 2 módulo *N* = 7, obtenemos la secuencia 2,4,1, … que se repite con la longitud 3, lo que significa que 7 es compuesto o 2 no es una raíz primitiva (mod 7). Si en cambio probamos potencias de 3 obtenemos la secuencia 3,2,6,4,5,1 que es de la máxima longitud posible *N*−1 = 6 para el grupo multiplicativo (mod 7), por lo tanto, concluimos que 7 es primo a modo de haber encontrado que 3 es una raíz primitiva. Si en cambio probamos las secuencias de potencia resultantes de los mismos módulos bases del compuesto 15 obtenemos 2^k (mod 15) = 2,4,8,1, … y 3^k (mod 15) = 3,9, -3, -9, …, para el índice k = 1,2,3, …. Observamos que ambas secuencias se repiten con periodicidad 4, que en primer lugar es menor que *N*−1 y en segundo lugar no divide N−1 (es decir, no hay posibilidad de que una repita en la secuencia de 2^k que acaba en la ranura N-1), y que la secuencia 3^k ni siquiera contenga un 1, mientras que ambas secuencias (mod 7) contienen uno o más unos, en particular ambos tienen 1 en el intervalo (*N*−1) Es decir, para *N* = 7 tenemos $a^{N-1} \equiv 1$ (mod *N*) para todas las bases, no un múltiplo de 7, donde la triple-barra-igual es el símbolo para la equivalencia modular, es decir, los 2 lados de la relación son iguales cuando se reduce el módulo N. Para las secuencias (mod 15), por otro lado, la no ocurrencia de 1 como potencia *N*−1 en cualquiera de las dos es suficiente para demostrar el compuesto 15.

La computación de estas secuencias de mod-potencia para módulos grandes no es práctica para grandes N, de modo que la idea de calcular solo la potencia (N-1) de la base es crucial, ya que eso requiere el cálculo de unos meros intermedios O(lg N), donde lg es el logaritmo binario. La prueba resultante resulta ser rigurosa solo en el sentido de identificar correctamente los números compuestos, siendo meramente probabilística para los primos porque arroja un resultado de falso positivo '1' para un pequeño porcentaje de módulos compuestos además de los primos, pero puede ser modificada para obtener una prueba determinante (rigurosa) para una fracción convenientemente grande de estos números primos falsos problemáticos.

El "pequeño" teorema de Fermat y la prueba de probable-primalidad

Tanto las rigurosas pruebas de primalidad como las llamadas pruebas de probable-primalidad se basan de una manera u otra en la existencia de una raíz primitiva, y es muy útil utilizar la variedad probable-primo para ejemplificar el tipo de aritmética necesaria para implementar esta prueba, especialmente para módulos grandes. El "padre de la teoría de números", Pierre de Fermat, a principios del siglo XVII ya había observado y formulado en un teorema lo que hemos señalado anteriormente, que para cualquier primo p, si a es coprimo para (no tiene factores en común con) p – puesto que p primo significa que a no debe ser un múltiplo de p – entonces

```
a^(p-1) ≡ 1 (mod p).  [*]
```

Esto se conoce actualmente7 como el "pequeño teorema" de Fermat para diferenciarlo de su "último teorema" más famoso (pero de menor importancia práctica), acerca de las soluciones sobre los enteros de la ecuación $a^n+b^n = c^n$; la prueba resultante aplicada a los números de caracteres desconocidos se denomina, indistintamente, como una prueba de compostura probada de Fermat o de probabilidad probable de Fermat. La primera denominación se refiere al hecho de que un entero N satisfactorio $a^{N-1} \equiv 1$ (mod *N*) para una base coprimo a N es muy probable que sea primo, para una gran muestra estadísticamente hablando, el segundo al hecho de que los números que fallan con este criterio son ciertamente compuestos, incluso si no se ha encontrado un factor explícito de N.

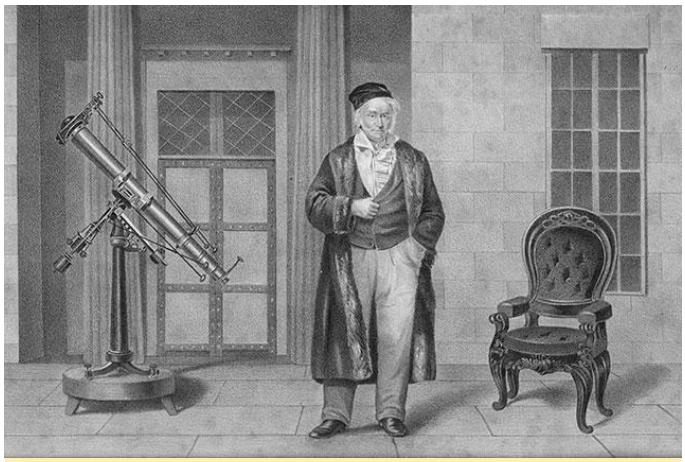
Pierre de Fermat, el "padre de la teoría de los números"

Tenga en cuenta que el inverso de [*] no es válido, es decir, hay pares enteros coprimos *a,N* para los cuales $a^{N-1} \equiv 1 \pmod{N}$ pero donde *N* no es un primo; por ejemplo, usando la utilidad de linux 'bc' se puede ver que el compuesto *N* = 341 = 11 × 31 satisface el teorema para la base a = 2, invocando bc en un intérprete de comandos y simplemente escribiendo '2^340%341'. Esto subraya la importancia de probar bases múltiples si el primero genera $a^{N-1} \equiv 1 \pmod{N}$: para *N primo*, cada entero en 2,...,*N*-2 † es coprimo para *N*, y por lo tanto todas estas bases producen 1 en el resultado, mientras que para el compuesto N, casi siempre basta con probar un pequeño número de bases para revelar N como compuesto. Decimos "casi siempre" porque existe una clase especial de números enteros conocidos como *números de Carmichael*, que pasan la prueba de Fermat para todas las bases que son coprimo a N; el más pequeño es 561 = 3×11×17. Los números de Carmichael solo revelan su carácter compuesto si elegimos una base a para la prueba de Fermat que no es coprimo para *N*, i.e. es decir, si encontramos un factor de *N*. En la práctica siempre se debe verificar primero N para pequeños factores primos hasta algún limite (fijado por el coste de dicha factorización de prueba) antes de someter a N a una prueba de primalidad probable al estilo de Fermat.

† Saltamos tanto 1 como N-1 como bases potenciales porque, siendo $g \equiv \pm 1 \pmod{N}$,estos dos valores "bookend" de ambos producen trivialmente $a^{N-1} \equiv 1$ para todos los N impares.

Para la base *a* = 2 hay precisamente 10403 de estos **pseudoprimes de Fermat**, un número minúsculo comparado con los cerca de 200 millones de números primos por debajo de ese límite, de modo que incluso usar simplemente esta base única nos proporciona una manera notablemente efectiva de determinar si un número es susceptible de ser primo. Por ejemplo, se puede combinar una prueba de pseudoprimo Fermat base-2 con una tabla preconfigurada de los compuestos conocidos inferior a 2^{32} que pase la prueba para producir un algoritmo de primalidad determinista muy eficiente para números por debajo de ese límite. Sin embargo, en un contexto más general de probar números de tamaño arbitrario, es importante darse cuenta que hay ciertas clases de números, los cuales son pseudoprimos Fermat base-2, independientemente de si son primos o compuestos. Las dos clases más conocidas son, en primer lugar, los números de Mersenne M(p) = 2^p-1 (para los cuales restringimos la definición a los exponentes principales ya que se requiere que un número de esta forma tenga la oportunidad de ser primo); por ejemplo,, $2^{11}-1$ pasa la prueba aunque tiene un factor de 23 × 89.

La segunda clase de estos números son los números de Fermat $F_m = 2^{2^m}+1$. Parece que los primeros cinco números de ese tipo, $F_0 - F_4$ = 3,5,17,257,65537 son lo suficientemente pequeños como para ser susceptibles a la división de prueba de lápiz y papel, y se pone de manifiesto fácilmente que son primos de esta manera, junto con el hecho que todos los que hayan pasado la prueba nombrada puede haber llevado a Fermat a hacer su famosa conjetura incorrecta de que todos estos números son primos. Euler refuta esta conjetura décadas más tarde, al mostrar que 641 es un factor primo de F5, y el trabajo posterior ha llevado a la creencia general de que con toda probabilidad los cinco números más pequeños son de hecho los únicos primos en la secuencia. Simplemente cambiando la base de la prueba de Fermat a, digamos, 3, es suficiente para distinguir los primos de los compuestos en esta secuencia de números, pero parece que esta idea nunca se le ocurrió a Fermat.



Carl Friedrich Gauss, uno de los matemáticos más grandes de todos los tiempos

Exponenciación modular eficiente

Para llevar a cabo la exponenciación modular, usamos una técnica general, o mejor, un conjunto relacionado de técnicas, conocido como jerarquía de alimentación binaria. El nombre se refiere al hecho de que los diversos enfoques se basan en analizar los bits del exponente representado en forma binaria. Para animar al lector a calcular junto con la lectura, utilizamos la calculadora de precisión arbitraria integrada POSIX, bc, que es relativamente lenta en comparación con los programas gratuitos de teoría de números de gama superior como Pari/GP y la librería multiprecisión Gnu, GMP, pero nos puede ser extremadamente útil para este tipo de 'prototipo rápido' algorítmico básico. Invocamos la calculadora en modo de número entero por defecto simplemente a través de 'bc' en un terminal; 'bc -l' invoca la calculadora en modo de punto flotante, en la cual la precisión puede ajustarse para adaptarse usando el valor del parámetro 'scale' (cuyo valor predeterminado es 20 dígitos decimales), y '-l' define la librería matemática estándar, que contiene un puñado de funciones útiles que incluyen el logaritmo y el exponente natural, el seno trigonométrico, el coseno y la arcotangente (a partir de los cuales se pueden desarrollar otras cosas, por ejemplo '4*a(1)' calcula π con la precisión fijada por el valor actual de escala usando ese arctan (1) = π/4), y la función de Bessel del primer tipo. La base aritmética de las entradas y salidas de bc se puede controlar modificando los valores de ibase y obase de sus valores por defecto del 10, por ejemplo, para ver 23 en binario, escriba 'obase = 2; 23 'que fielmente produce 10111; Resetea la escala a su valor decimal por defecto mediante 'scale = 10'.

Obsérva que para los módulos en general, y en particular los muy grandes, no podemos simplemente calcular la potencia en el lado izquierdo de [*] y reducir el módulo resultante N, ya que los números son lo suficientemente grandes como para saturar incluso nuestro extenso almacenamiento informático. En general, las potencias se duplican en longitud para cada bit del exponente, por lo que elevar la base 2 a un exponente de 64 bits da un resultado del orden de 264 o 18,446,744,073,709,551,616 bits, o más de 2000 petabytes, ejemplificado muy bien el famoso **problema del tablero de ajedrez y el trigo** en las matemáticas de las series geométricas. Para probar un número del tamaño del **primo Mersenne**, más recientemente descubierto, tendríamos que hacer decenas de millones de este tipo de reiteraciones de duplicado de tamaño, entonces, ¿cómo es posible qué esto esté disponible en el hardware de cálculo? Volvemos a las propiedades de la aritmética modular, una de las más importantes es que al calcular un gran 'powermod' de este tipo, podemos hacer una reducción modular en cada paso del camino, siempre que sea conveniente hacerlo. Así, en la práctica, se usa una jerarquía binaria de potencia para dividir la exponenciación en una serie de escuadrados y multiplicaciones, a cada paso le sigue una reducción modular del intermedio resultante.

Compararemos y contrastaremos dos planteamientos básicos para el cálculo de $a^b \pmod{n}$, de potencia binaria, que recorre los bits del exponente b en direcciones opuestas. Ambos hacen una cuadratura por bit de b, así como algunas multiplicaciones más generales cuyo conteo preciso depende del patrón de bits de b, pero nunca puede exceder el de los escuadrados. El método de derecha a izquierda activa un acumulador y = 1 y un cuadrado actual $z = a$, luego para cada bit en n empezando con el bit más a la derecha (unos), si el bit actual = 1, multiplicamos el acumulador por el cuadrado actual z, luego de nuevo el cuadrado z para prepararse para el próximo bit a la izquierda. Aquí hay una simple función bc

definida por el usuario que ilustra esto: por razones de simplicidad, hemos omitido algunos pre-procesamientos básicos que se incluirían para verificar la cordura de las entradas tales como las pruebas de módulo cero y de exponente no negativo:

```
/*
 * right-to-left-to-right binary modpow, a^b
(mod n):
 */
define modpow_rl(a,b,n) {
  auto y,z;
  y = 1;
  z = a%n;
  while (b) {
     if(b%2) y = (y*z)%n;
     z = (z*z)%n;
     b /= 2;
  }
  return (y);
}
```

Lo dejamos como un ejercicio para que el lector implemente una simple optimización que agregue una anticipación dentro del ciclo de forma que la actualización del cuadrado actual solo se realice si hay un bit próximo a la izquierda para procesar, lo cual es útil si la base a es grande pero el exponente es pequeño. Tras pegar el código anterior en tu intérprete de comandos bc, intenta hacer una prueba de pseudoprimo de Fermat de base 2 del primo Mersenne conocido M(2203): 'n=2^2203-1; modpow_rl(2,n-1,n)' (esto llevará unos segundos). Ahora vuelve a intentarlo con un exponente compuesto de tamaño similar, digamos 2205, y observa el resultado no unitario que indica que n = $2^{2205}-1$ también es compuesto. Como 2205 tiene pequeños factores primos 3,5 y 7, podemos utilizar aún más la función de módulo bc '%' para verificar que este número sea exactamente divisible entre 7,31 y 127.

Por supuesto, ya tenemos presente que una prueba de pseudoprimo de Fermat de base 2 devuelve 'primo probable' para todos los números de Mersenne, es decir, para todos los 2^p-1 con p primo, podemos usar la función modpow anterior para verificar esto, ahora modificando el valor de retorno a un 0 binario (compuesto) o 1 (pseudoprimo a la base dada): 'n=2^2207-1; modpow_rl(2,n-1,n) == 1' devuelve 1 a pesar de que el número de Mersenne en cuestión tiene un factor pequeño, 123593 = 56 × 2207 + 1. Repitiendo la misma prueba de pseudoprimo de Fermat pero ahora con la base 3 correctamente, revela que este número es compuesto. Observa el salto asociado en el tiempo de ejecución de bc. Esto parece reflejar algunas optimizaciones de casos especiales en la lógica interna de bc relacionadas con el reconocimiento de argumentos que son potencias de 2. Vemos una discrepancia de velocidad similar al repetir la prueba de pseudoprimo usando las bases 4 y 5, por ejemplo.

Nuestro próximo algoritmo de potencia procesa los bits en la dirección opuesta, de izquierda a derecha. Este método activa el acumulador y = a, que corresponde al bit fijado más a la izquierda, luego para cada bit hacia la derecha cuadramos y, y si el bit actual = 1, multiplicamos el acumulador por la base de potencia a. En un lenguaje de codificación como C podríamos, ya sea a través del compilador o a través de una pequeña macro de código de ensamblaje, implementar las funciones de bits mediante el metodo binario de dividir y vencer o accediendo de manera eficiente a cualquier instrucción de hardware disponible para liderar el recuento de ceros, y nuestra función de reversión de bits reverse () podría implementarse de manera eficiente utilizando una pequeña tabla de 256 bytes precomputados con bit invertidos, un ciclo para hacer intercambios de byte a los extremos izquierdo y derecho de nuestro exponente, y un paso final para desplazar hacia la derecha el resultado de 0-7 bits, dependiendo de dónde se encuentre el bit establecido más a la izquierda en el byte configurado más a la izquierda. En BC no tenemos esa funcionalidad bit a bit y debemos desplegar nuestras propias tareas de emulación ineficientes, pero como nuestro foco está en modpow de grandes operandos, el coste de tiempo de tales operaciones bit a bit es mínimo en comparación con las multiplicaciones modulares:

```
define bits(n) {
  auto ssave, r;
  ssave = scale;  scale = 0;  /* In case we're
in floating-point mode */
```

```
  r = length(n)*3321928095/1000000000;
  while ( 2^r > n ) { r -= 1; }
  scale = ssave;
  return(r+1);
}

define reverse(n,nbits) {
  auto tmp;
  tmp = 0;
  while(nbits) {
    tmp = 2*tmp + (n % 2);
    n /= 2;
    nbits -= 1;
  }
  return(tmp);
}

/* left-to-right binary modpow, a^b (mod n):
*/
define modpow_lr(a,b,n) {
  auto y,len;
  len = bits(b);  b = reverse(b,len);
  y = a%n;  b /= 2;
  while(--len) {
    y = (y*y)%n;
    if(b%2) y = (a*y)%n;
    b /= 2;
  }
  return(y);
}
```

La necesidad de inversión de bits también se puede evitar implementando el algoritmo de **forma recursiva**, pero como bc es, digamos, menos que espectacular cuando se trata de soporte eficiente para la recursión, preferimos un algoritmo no recursivo en el caso de izquierda a derecha. Instamos a los lectores a pegar lo anterior en su intérprete de comandos bc, utilízalo para realizar nuevamente las pruebas de Fermat-pseudoprimo base-2 y base-3 en $2^{2207}-1$, y compáralas con las del algoritmo de derecha a izquierda. En mi interprete de comandos bc, el método de izquierda a derecha se ejecuta en casi la mitad del tiempo en el número compuesto de Mersenne antes mencionado, a pesar de que las entrañas de los bucles en nuestras funciones de alimentación RL y LR son bastante similares, cada una con un mod-cuadrado y un mod-multiplicador.

El motivo de la aceleración en el método LR se vuelve claro cuando examinamos con precisión qué operandos están involucrados en las dos operaciones respectivas de multiplicación de mod. En la potencia RL, multiplicamos el acumulador de corriente y y el mod-square actual z, que son del tamaño del módulo n. En la alimentación LR, multiplicamos el acumulador de potencia actual y la base a, siendo este último generalmente el orden de la unidad, u O (1) en la notación de orden asintótica estándar.

De hecho, para bases pequeñas, podemos reemplazar el producto a×y por una serie de cambiar y acumular bits hacia la izquierda de y si resulta ventajoso, aunque la línea inferior – con vistas a la implementación de hardware subyacente de estas operaciones de precisión arbitraria a través de la aritmética de múltiples palabras – que es la del algoritmo LR multiplicamos el vector y por la base escalar a, que es lineal en la longitud del vector en términos de coste. Para los exponentes generales cuyos bits se dividen por igual entre 0 y 1, solo nos damos cuenta de esta reducción de costes para los bits 1, pero nuestro ejemplo numérico particular implica un número de Mersenne, cuyos bits son 1, así el exponente de potencia binaria n- 1 tiene solo un bit 0 en la posición más baja, y si vector-times-vector mod-square y mod-multiply tienen aproximadamente el mismo coste (como en el caso de bc), reemplazamos este último por un vector-times-scalar reduciendo el tiempo de ejecución aproximadamente a la mitad, como se ha observado.

Marin Mersenne es más conocido por los números primos de Mersenne, un tipo especial de número primo

Prueba determinista de la primalidad

Si bien la prueba de Fermat-pseudoprimo es una forma eficiente de identificar si un número dado es probablemente primo, nuestro interés real es establecer rigurosamente el carácter, primo o compuesto del mismo. Por lo tanto, es importante complementar estas pruebas de primalidad probable con alternativas deterministas siempre que sea posible. Si dicha alternativa puede realizarse con un coste computacional similar, sería lo ideal, ya que la calculo a^{N-1} (mod N) utilizando el enfoque modular de alimentación de binario de izquierda a derecha requiere del cálculo de O(lg N) escuadrados modulares de intermedios de tamaño N y un número similar de multiplicaciones modulares de tales intermedios por la base a, que generalmente es mucho menor que N, lo que significa que estas multiplicaciones escalares suplementarias no tienen significado en términos de coste algorítmico global asintótico de N grande. Hay razones para creer, aunque esto no se ha demostrado, que el coste de una cuadratura modular por bit de N es de hecho el óptimo que se puede lograr para una prueba de primalidad no factorial.

Por desgracia, parece que solo hay una clase limitada de números de formas muy especiales para las cuales existe una prueba determinista de la primalidad con un coste computacional similar; las más famosas son las dos clases mencionadas anteriormente. Para los números de Fermat utilizamos una generalización de la prueba de pseudoprimos de Fermat gracias a Euler, en la que uno calcula un $a^{(N-1)/2}$(mod *N*) y compara el resultado con ± 1, con el signo apropiado dependiendo de una particular propiedad algebraica de N. Para los números de Fermat, basta con tomar a = 3 y verificar si $3^{(F_m-1)/2} = 3^{2^{2^m}-1} \equiv -1 \pmod{F_m}$, que requiere precisamente 2^m-1 mod-squarings de la simiente inicial 3. Lo suficiencite para determinar la primalidad de los números de Fermat se conoce como el teorema de Pépin. El lector puede usar cualquiera de las funciones modpow anteriores de bc para realizar la prueba de primalidad Pépin en un número de Fermat de hasta varios kilobits de tamaño; por ejemplo, para probar el F11 de 2 kilobits, 'n = 2^(2^11)+1; modpow_lr(3,(n-1)/2,n) == n-1'. Ten en cuenta que los algoritmos LR y RL se ejecutan aproximadamente al mismo tiempo en los números de Fermat, ya que la potencia calculada en la exponenciación modular de la prueba Pépin es una potencia de 2, por lo tanto solo tiene 1 bit en la posición más a la izquierda.

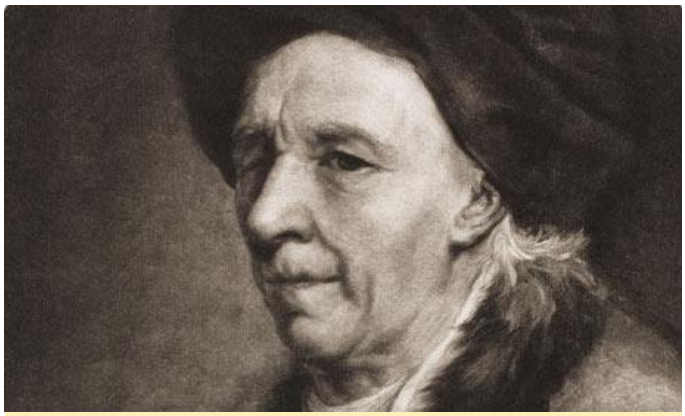
Para los números de Mersenne $M(p) = 2^p-1$, la prueba de primalidad se basa en las propiedades algebraicas de las llamadas secuencias de Lucas según el matemático francés Édouard Lucas, pero luego fue refinada por el teórico de los números Derrick Lehmer en una forma algorítmica simple conocida como la prueba de primalidad de Lucas-Lehmer: comenzando con cualquiera de los valores iniciales algebraicamente permitidos x (el más utilizado de los cuales es 4), realizamos precisamente actualizacones repetitivas p-2 de la forma $x = x^2-2 \pmod{M(p)}$; el número de Mersenne en cuestión es primordial si y solo si el resultado es 0. Por ejemplo, para p = 5, tenemos iteraciones no modificadas 4,14,194,37634; en forma modificada estas son 4,14,8,0, lo que indica que la iteración final 37634 es divisible por el módulo 31 y por lo tanto que este módulo es primo. Al igual que con nuestras pruebas de primalidad probable y la prueba de Pépin, tenemos una de esas iteraciones por cada bit del módulo, dar o recibir uno o dos..

Para dar una idea de las eficiencias relativas de estas pruebas de módulo especializado en comparación con las pruebas de primalidad determinística para los módulos generales, los más rápidos conocidos se basan en la aritmética de las curvas elípticas y se han utilizado para probar la primalidad de números que tienen alrededor de 20,000 dígitos decimales, mientras que las pruebas de Lucas-Lehmer y Pépin se han realizado hasta la fecha en números que se acercan a los 200 millones de dígitos. Puesto que, como mostraremos a continuación, el coste total de las pruebas especializadas es ligeramente mayor que la cuadrática en la longitud de la entrada, esta disparidad de tamaño del factor de 10 se traduce en una diferencia efectiva proporcionalmente mayor en la eficacia de la prueba. En términos de las pruebas de módulo especializado, la diferencia de velocidad es más o menos equivalente a una disparidad de cien millones de veces en la eficiencia de las pruebas

basada en los tamaños de los que mantiene el record actual de registros para ambos tipos de pruebas.

Rápida multiplicación modular

A pesar del formato superficialmente distinto de las dos pruebas de primalidad anteriores, observamos que, tanto para la operación crucial de control de rendimiento, como en el caso de la prueba de pseudoprimo de Fermat, es la multiplicación modular quien toma el producto de un par de números de entrada y reduce el módulo de resultado en un tercio. (La operación de resta 2 adicionales de la prueba de LL es insignificante con respecto a estas escalas de trabajo asintóticas de grandes operandos). Para entradas de tamaño modesto podemos usar un algoritmo de multiplicación estándar de la "escuela de gramática" dígito por dígito, seguido de una división con resta por el módulo, pero de nuevo, para números grandes debemos ser bastante más inteligentes que esto.



Leonhard Euler, autor de más de 1000 artículos, muchos trascendentales, en campos que van desde la teoría de números hasta la mecánica de fluidos y la astronomía, y que al perder su vista a finales de sus 40 (y sinceramente, basándose en su resultado de investigación posterior) comentó: "Ahora Tendré menos distracciones"

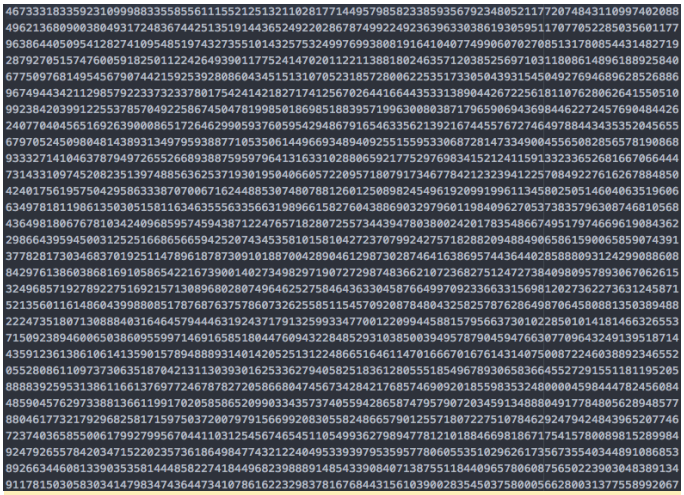
La idea principal que hay detrás de los modernos algoritmos de multiplicación de grandes enteros se debe a **Schönhage and Strassen** (ver además **http://numbers.computation.free.fr/Constants/Algorithms/fft.html** para una exposición más matemática de la técnica), quienes reconocieron que la multiplicación de dos enteros equivale a la complejidad digial de las entradas. Esta idea permite que cualquiera de los conocidos algoritmos complejos de alta eficiencia del ámbito del procesamiento digital de señales se aproveche del problema. Como reflejo de la evolución del microprocesador moderno, las implementaciones más conocidas de dichos algoritmos utilizan la Transformación Rápida de Fourier (FFT) y hacen uso del hardware punto flotante del procesador, a pesar de los errores de redondeo asociados que hacen que las salidas complejas se desvíen de los valores de enteros puros, se tendrían que usar la aritmética exacta.

A pesar de esta inexactitud, el software FFT de alta calidad es notablemente agresivo en la cantidad de errores de redondeo que uno puede soportar sin corromper fatalmente las largas cadenas de multiplicación involucradas en pruebas de primalidad de enteros grandes: por ejemplo, la prueba de Lucas-Lehmer que descubrió el primo Mersenne en record de tamaño más reciente, $2^{77232917}-1$, usaba una FFT de doble precisión que dividía cada iteración de 77232917 bits en 2^{22} palabras de entrada de 18 o 19 bits de longitud (hablando con exactitud $1735445 = 77232917 \pmod{2^{22}}$) de palabras grandes y las 2458859 restantes de tamaño más pequeño), así se usa algo más de 18 bits por 'dígito' de entrada de la complejidad discreta. Esto proporciona salidas de complejidad de coma flotante que tiene partes fraccionales (es decir, errores de redondeo acumulados) tan grandes como casi 0,4, lo cual está notablemente cercano al fatal nivel de error 0.5 "No sé si redondear hacia arriba o hacia abajo este resultado de complejidad de punto flotante inexacta".

No obstante, las múltiples ejecuciones de verificación utilizando implementaciones FFT desarrolladas independientemente en diferentes (y mayores, por lo tanto, tienen errores de redondeo mucho más pequeños) longitudes de transformación, en varios tipos diferentes de hardware, confirmaron la exactitud de la prueba de primalidad inicial. Para un módulo de n bits, el coste de cálculo n-asintótico de tal multiplicación de FFT es $O(n \lg n)$, aunque te das cuenta de esto en la práctica, especialmente cuando los operandos son lo suficientemente grandes como para exceder los tamaños de los cachés de datos de varios niveles utilizados en los microprocesadores modernos son un desafío algorítmico y de movimiento de datos muy poco significativo. Dado

que nuestras diversas pruebas de primalidad requieren O(*n*) de estos multiples pasos, la estimación de trabajo para una prueba de primalidad basada en FFT es O(n^2 lg *n*), que es solo un factor lg *n* mayor que el coste cuadrático de una sola escuela primaria multiplicar. El resultado es que para escribir un programa de prueba de primalidad de clase mundial, uno debe escribir (o hacer uso de) una complejidad basada en transformaciones de clase mundial.

Aproximadamente hace 20 años, estaba en la facultad de ingeniería en la Universidad de Case Western Reserve en Cleveland, Ohio, y estaba buscando una forma interesante de motivar la enseñanza del algoritmo de Transformación Rápida de Fourier de procesamiento de señales a los estudiantes en mis clases de sistemas computacionales. Algunas búsquedas online mostraron el uso de FFT en la multiplicación de grandes enteros y el resto era simplemente historia, como dice el refrán.



El precedente del número primo más grande, descubierto el 26 de diciembre de 2017 por un esfuerzo de computación distribuida por voluntarios, tiene 23,249,425 dígitos

La segunda mayor aceleración algorítmica en las pruebas de primalidad modernas llegó más o menos una generación después del algoritmo Schönhage-Strassen, e involucró la longitud de FFT necesaria para multiplicar dos enteros de un determinado tamaño. Para realizar una multiplicación modular de un par de n bits, en general primero debemos calcular exactamente el producto, que es el doble de bits, y luego reducir el producto (es decir, calcular el resto) con respecto al módulo en cuestión. Para los módulos especiales "amigables con los binarios" tales como los números de Mersenne y Fermat, la reducción se puede hacer de manera eficiente utilizando aritmética bit a bit, pero calcular el producto de doble ancho todavía supuen un coste adicional, ya que nos exige aplicar cero a nuestras entradas de complejidad FFT con *n* 0 bits en la mitad superior y usar una longitud FFT dos veces más grande que nuestros resultados reducidos.

Todo esto cambió en 1994, cuando el fallecido Richard Crandall y Barry Fagin **publiscarib un artículo** que mostraba cómo, para los casos especiales de Mersenne y módulos de número de Fermat, este relleno de cero podía evitarse mediante la ponderación inteligente de las entradas de transformación con el fin de efectuar una operación de "mod implícito". Este avance produjo una mayor aceleración que el factor 2 en las pruebas de primalidad de estas dos clases de módulos, y los tipos asociados de transformaciones ponderadas discretas se han extendido posteriormente a varias clases interesantes de módulos. Para ver un ejemplo sencillo de cómo se puede usar una FFT para una modificación implícita, remitimos al lector a la **post del autor en mersenneforum.org**. Desgraciadamente, existen otros aspectos algorítmicos importantes que intervienen en la modulación de alta eficiencia en el hardware de los procesadores modernos: soporte de longitud de transformación no potencial de 2, FFT no recursivas en el lugar que no requieren reordenamiento de reversión de bits antiadherente de los vectores de entrada, el movimiento de datos optimizado para permitir el multihilo eficiente, etc., está más allá del alcance del artículo actual, pero aseguramos al lector/programador interesado que hay más, mucho más, de interés involucrado. Pasamos ahora a las diversas consideraciones que llevaron al esfuerzo especial de implementar soporte de código ensamblador para la plataforma ODROID y su conjunto de instrucciones de aritmética vectorial de 128 bits, y finalmente a los aspectos prácticos de la compilación y ejecución del código de prueba LL resultante en la plataforma Odroid.

¿Por qué la plataforma ARM?

Pasé la mayor parte de mi tiempo de desarrollo en los últimos 5 años paralelizando mi esquema de código FFT de Mlucas existente utilizando el entorno de trabajo pthreads de POSIX más algún código de afinidad central basado en las diversas extensiones compatibles con afinidad proporcionadas en varios sistemas Linuxy MacOS importantes. Una vez que tuve un marco de trabajo paralelo funcional que admitiese desarrollos de vectores SIMD x86 SSE2 de 128 bits y de doble escala, actualicé mis librerias SIMD en línea-ensamblaje-macro a través de sucesivas actualizaciones del conjunto de instrucciones vectoriales de Intel: primero 256 bit AVX, luego AVX2 que en primer lugar promovió las matemáticas del vector entero al mismo nivel que el punto flotante extendiendo el mencionado vector completo de 256 bits y en segundo lugar añadió soporte para instrucciones aritméticas de punto flotante (FMA3) fusionando 3 operandos . Al mismo tiempo que trabajaba en este proyecto reciente, necesitaba ser compatible con el conjunto de instrucciones AVX512 de 512 bits de Intel (que apareció por primera vez en el mercado en forma de barebones indescifrable, pero a la vez muy útil como subconjunto de instrucciones de cimentación desde principios de 2017 en estaciones de trabajo Knights Landing) el año pasado también estaba considerando una primera incursión en agregar soporte SIMD a una familia de procesadores que no sean x86. Las principales consideraciones para emprender tal esfuerzo, que normalmente son de 4 a 5 meses de codificación enfocada, depuración y puesta a punto del rendimiento, fueron las siguientes:

- Amplia Base de instalación y comunidad de desarrolladores activa;
- Compatibilidad con el compilador/depurador estándar de Linux + GCC tolestlow y la paralelización de preads de POSIX;
- Conjunto de instrucciones bien diseñado, preferiblemente al estilo RISC, con al menos tantos registros vectoriales y de propósito general como el Intel AVX's 16-vector/16-GPR, y preferiblemente tantos registros (32 de cada tipo) como el AVX512 de Intel;
- Soporte para instrucciones FMA;
- Competitividad con las principales familias de procesadores de alta gama (por ejemplo, CPUs Intel y GPUs nVidia) en términos de rendimiento por vatio y de hardware por dólar;
- Posibilidad de mejoras continuas en las implementaciones de procesadores.

ARM (en concreto las instrucciones SIMD-vector ARMv8 de 128 bits y las diversas CPU que lo implementan) pasaron a ser rápidamente el principal candidato. Alta eficiencia energética, un generoso conjunto de registros de 32 + 32 y un excelente diseño de conjunto de instrucciones, mucho mejor que Intel SSE2 con su aspecto de Frankenstein (tuve en cuenta la compatibilidad casi cómica de instrucciones completas de vectores de 64 bits y la falta de una instrucción ROUND en la primera iteración SSE2 como dos de los muchos ejemplos incluidos aquí). Una vez que compré mi pequeño ODROID C2 y trabaje con lo problemas de crecimiento esperados de la nueva nemotecnia de instrucción y la sintaxis de ensamblaje en línea frente a SIMD x86, el desarrollo fue muy sencillo. Una primera inquietud, en la forma de la sintaxis FMA3 algo más restringida frente a la de Intel, resultó infundada, el impacto de dicha restricción se mitigó fácilmente mediante una simple reorganización del orden de los operandos, en relación con el generoso conjunto de 32 registros vectoriales, lo cual fue muy útil. ¡Esperamos que ARM tenga una actualización de 256 bits de las instrucciones del vector v8 en su hoja de ruta para un futuro no muy lejano!

Configurando el software

El software está diseñado para ser tan fácil desarrollar como sea posible en la mayor variedad distribuciones de Linux posibles. Después de haber tenido tantas experiencias malas y de pérdida de tiempo con el típico paradigma de desarrollo configure y make para freeware de Linux, decidí deliberadamente abandonarlo de una manera un tanto radical, en lugar de esforzarme al máximo en la creación de mis archivos cabezera para lo más lejos posible automatizar el proceso de identificación de los aspectos clave de la plataforma de compilación durante el preprocesamiento, tomando cualquiera de los identificadores específicos del hardware admitidos por el software (p. ej., el usuario quiere desarrollar

para CPUs x86 que soporte el subconjunto de instrucción vectorial AVX2 / FMA3, o más pertinentemente para ODROIDers, las instrucciones vecto aritmética SIMD ARMv8 de 128 bits) del usuario en tiempo de compilación, además de elegir si se compila un binario de subproceso único o multiproceso.

Así pues el desarrollo se reduce a la siguiente secuencia de 3 pasos, que se presenta en la **Página de Mlucas**: Insertando todos los indicadores específicos de las arquitectura aplicables, compila todos los archivos. En mi ODROID-C2, quería activar el ensamblador online que apunta a ARMv8 y quiero poder ejecutarlo en paralelo en los 4 núcleos de procesador, por lo que el comando de compilación es:

```
$ gcc -c -O3 -DUSE_ARM_V8_SIMD -DUSE_THREADS
../src/*.c >& build.log
```

Usar grep en el archivo build.log resultante del paso de compilación para los errores:

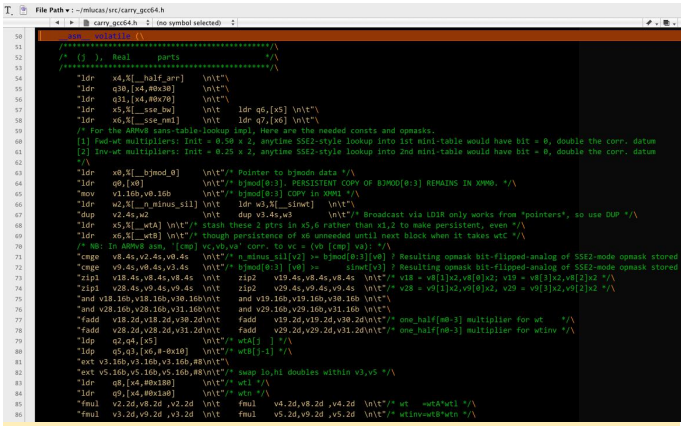
```
$ grep -i error build.log
```

Si no hay errores de compilación, enlaza un archivo binario:

```
$ gcc -o Mlucas *.o -lm -lpthread -lrt
```

Ejecuta la típica serie de autocomprabacion para el rango “mediano” de longitudes FFT que cubre todas las asignaciones de usuarios GIMPS actuales y no muy lejanas: ‘./Mlucas -s m’. Un ODROID pasará varias horas probando las diversas formas de combinar las bases aritméticas complejas individuales que componen cada una de las distintas longitudes de FFT cubiertas por la autocomprobación, y capturando las que ofrecen el mejor rendimiento en el hardware particular del usuario a un archivo de configuración maestro llamado mlucas.cfg, que es texto sin formato al margen del sufijo no estándar.

Los tipos más comunes de errores de compilación encontrados anteriormente se pueden rastrear para usuarios que se dirigen a un conjunto de instrucciones no admitido por su CPU particular o ocasionalmente a alguna particularidad de la plataforma de compilación que necesita algunos ajustes para la lógica del preprocesador en el crucial archivo de cabecera plataforma.h. Dado que SBC ODROID viene precargado con distribuciones uniformes de sistema operativo Linux, los únicos problemas que probablemente surjan están relacionados con el conjunto de instrucciones precisas (SIMD v8 o no) compatibles con el modelo particular de ODROID que se utiliza.



Código de ensamblaje en línea de MLucas para ARMv8

En mi ODROID-C2, ejecutando la compilación SIMD de 128 bits del software en los 4 núcleos obtiene un rendimiento total equivalente al de una compilación utilizando las instrucciones vectoriales x86 SSE2 de 128 bits en un único núcleo de un ordenador portátil basada en Intel Core2 a 2 GHz . Esto se traduce en aproximadamente 1/20 del rendimiento total alcanzable en el quad Intel Haswell del autor utilizando el conjunto de instrucciones AVX2 de 256 bits, por lo que los 2 tipos de plataformas no son realmente comparables partiendo del rendimiento por núcleo. Las plataformas basadas en ARM como ODROID brillan por si solas en términos de rendimiento por vatio-hora y por rendimiento de hardware. A efectos de un usuario de ODROID el software tiene una posibilidad realista de descubrir el próximo primo Mersenne, bueno, necesitamos compensar con números lo que las plataformas de hardware de gama más alta (CPU Intel, GPU nVidia) hacen con grandes cantidades de transistores y núcleos cargados de potencia, es decir, tenemos que ser hormigas del ejército para leones. Y, por supuesto, las actualizaciones de velocidad como la prometida por el recientemente anunciado ODROID-N1 pueden ayudar. ¡Es un momento emocionante para ser micro-informático!

El final del número primo más grande descubierto hasta ahora, que se necesitó más de 6 días para verificarlo

Juegos ODROID: Juegos Saturn – Parte 2

March 1, 2018 By Tobias Schaaf Juegos, Linux, ODROID-XU4

Una vez más, volvemos al tema de los juegos ODROID-XU3/XU4 y Sega Saturn. He confeccionado una lista de juegos muy divertidos a los que me gusta jugar en el ODROID. Como muchos de los juegos para la Sega Saturn fueron versiones arcade, este artículo estará lleno de shoot-em-ups (“shmups”) una vez más.

Galactic Attack

Galactic Attack, también conocido como Layer Section, es un shooter arcade donde luchas en diferentes planos (capas) al mismo tiempo. Aunque dispones de las armas habituales para disparar a los objetivos que aparecen ante ti, también tienes misiles con los que pueden apuntar a objetivos en tierra. Para hacer esto, márcalos con tu punto de mira y presiona el segundo botón de disparo. Puedes elegir hasta seis objetivos a la vez, lanzando seis misiles a uno o varios objetivos individuales.

Figura 1 – Galactic Attack para Sega Saturn ejecutándose en ODROID-XU3/XU4

Aunque puede hacerte la vida más fácil, la mayoría de las veces no necesitas atacar a los objetivos en tierra. Sin embargo, cuando luchas contra el jefe de cada nivel, a menudo se hace necesario recurrir tanto ataques habituales como a los misiles para vencerlo.

Los combates con jefes están bastante pulidos y no son demasiado extremos. Descubrirás rápidamente los puntos débiles a los que tendrás que apuntar cuando dispares o lances misiles.

Si juegas con el frame skipping activado, los gráficos y animaciones pueden aparecer entrecortados. No me gustó demasiado, así que apagué el frame skipping por completo. El juego se ejecuta mucho más lento de esta forma, pero la animación suave y la velocidad más lenta te aportan algo de tiempo extra para planificar tus movimientos. Personalmente no tuve porblemas con la lenta velocidad del juego, aunque a otros podría no gustarle. Este juego también está disponible para MAME (o FBA) bajo el nombre Gunlock, que es mucho más fácil en ODROID, pero omite la banda sonora en calidad CD.

Hay una secuela llamada Layer Section II para Saturn que está totalmente en 3D y utiliza mucho machacado para las transparencias que, en mi opinión, no se ven muy bien. Funciona mejor con el frame skipping activado y es probablemente el más rápido de los dos juegos cuando los ejecutamos en el ODROID. Layer Section II fue exportado a PlayStation bajo el nombre RayStorm que, si me preguntas, es la mejor versión para ejecutar en ODROID, en comparación con la versión de Saturn.

Figura 2: Los combates con los jefes en Galactic Attack son desafiantes, pero divertidos

Figura 3 – Los combates con los jefes en Galactic Attack son desafiantes, pero divertidos

La versión de Sega Saturn es bastante agradable gracias a la banda sonora del CD y la velocidad del juego más lenta cuando deshabilita el frame skipping hace que el juego sea un poco más fácil en comparación con la versión MAME. Si quieres disfrutar de toda experiencia arcade, jugar Gunlock bajo MAME o FBA es probablemente la mejor opción.

Game Tengoku–The Game Paradise

Este singular juego te permite jugar a un shooter arcade dentro de un arcade. En realidad, vuelas dentro un arcade, y dentro de otros juegos arcade: juegos de carreras, shooters espaciales, y similares. Este juego parece existir solo en japonés o directamente en máquinas recreativas (MAME), aunque el juego Sega Saturn ofrece mucho más que la versión arcade.

Figura 4 – Portada del juego Tengoku en Sega Saturn ejecutándose en ODROID

Auqnue la versión arcade va directamente a la pantalla de títulos del juego, la versión de Saturn ofrece características adicionales, como por ejemplo, que eres recibido por una graciosa y animada chica anime.

Figura 5 – Selección del modo de juego y opciones en la versión de Sega Saturn

La primera opción te lleva directamente al juego, que básicamente es la misma que en la versión arcade, aunque puedes seleccionar el estilo de juego, incluida la opción de desplazamiento vertical u horizontal. Sin embargo, ten cuidado, ya que, al seleccionar la reproducción horizontal, se activan los controles del juego con ello. Si seleccionas la segunda opción del menú, cada nivel contará con una escena de anime que relata parte de la historia del juego. Como no hablo japonés, no entiendo casi nada, pero las animaciones son graciosas y están completamente interpretadas por voz, que son cosas que no están disponibles en la versión arcade. La tercera opción de menú le permite seleccionar diferentes configuraciones, como la disposición de los controles, el sonido y la música.

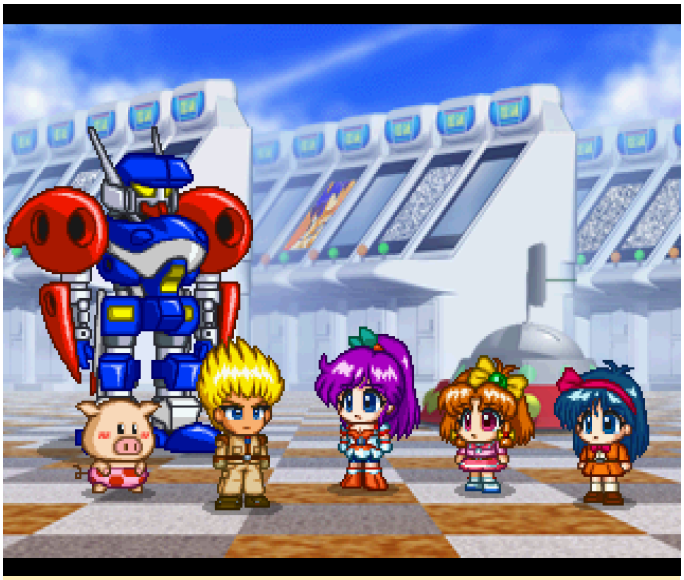


Figura 6 – Las escenas de anime entre niveles son muy graciosas e incluyen

Figura 7 – Las escenas de anime entre niveles son muy graciosas e incluyen sonido

Aparte de eso, estas ante el típico festival de balas japonés, lo que significa que probablemente vas a morir bastante a menudo. Afortunadamente, el juego ofrece continuies ilimitados, lo cual es bastante útil. La versión arcade de este juego debe haber sido muy complicado de jugar. Este juego tiene tantos jefes de nivel medio como jefes finales. Si tardas demasiado

en vencer a los jefes de nivel medio, es posible que escapen. El jefe final siempre estará ahí hasta que lo liquides.

El juego es muy divertido y el frame skipping funciona muy bien sin ninguna imagen entrecortada, aunque prefiero jugar sin frame skipping en el que el tiempo de reacción es más lento.

Figura 8 – El primer jefe es una máquina arcade real

Figura 9 – Estos son solo dos enemigos; imagínate una media de 5 a 10 en pantalla

Puedes seleccionar entre cinco luchadores diferentes, los cuales tienen diferentes armas. Tienes tu ataque principal, que es un ataque de frente usando diferentes tipos de proyectiles; un ataque con carga si mantienes presionado el botón de disparo un par de segundos, y tus ataques especiales (bombas) que puedes disparar con otro botón. Tus ataques especiales son fuertes y te permite derrotar a los jefes bastante rápido.

Game Tengoku tuvo una secuela totalmente en 3D para PlayStation con la posibilidad de jugar con hasta cinco personas, pero a la secuela le faltan las escenas de anime entre niveles, aunque hay algunas escenas de video, también al estilo anime. No me gusta particularmente la secuela de PlayStation, aunque es más fácil que la versión de Saturn o arcade. Si mueres y usas un continue, empiezas al principio del nivel y por lo general sueles morir con bastante frecuencia.

La secuela presenta un segundo nivel de ataque donde se etiqueta a un enemigo con una cruz y luego se lanza un tercer ataque similar al Ataque Galáctico, pero no es necesario ni muy útil. Además, ya no puede ver si tu “ataque con carga” está realmente cargado, más bien tiene que adivinar si está listo, lo cual es bastante molesto. Prefiero la versión de Saturn/Arcade sobre la secuela de PlayStation, aunque la secuela de Playstation tiene gráficos actualizados que se ven bastante bien.

Hyper Duel

Hyper Duel es bastante sencillo con todo lo que puedes esperar de este género. Tiene tanto un modo Arcade como un modo Saturn. Puede ajustar el nivel de dificultad, el número de vidas y el número de continues (1-5). Puede seleccionar uno de los tres luchadores, cada uno de los cuales tiene diferentes ataques primarios, secundarios y especiales. El ataque secundario, que transforma a tu luchador en un mecha, es un ataque más poderoso pero también te hace más grande, más lento y más vulnerable a los golpes.

Al presionar el primer y segundo botón de ataque al mismo tiempo, se lanza su ataque especial que puede actuar como un avión o un mecha, con resultados ligeramente diferentes dependiendo de la forma que hayas elegido. En modo avión, tu ataque especial está más extendido, lo que te permite golpear múltiples objetivos al mismo tiempo. En el modo mecha, tus ataques están más focalizado, lo que te permite infligir más daño.

Figura 10 – Hyper Duel para Sega Saturn salió un par de años después de la versión arcade

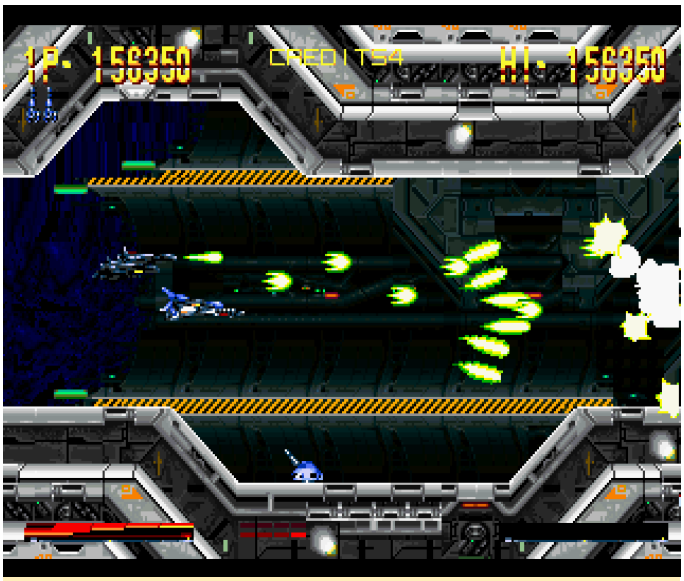


Figura 11 – En este juego, no solo los enemigos pueden disparar cientos de balas

A diferencia de otros shmups, los ataques especiales no causan grandes daños. Por el contrario, son ataques de apoyo que causa un daño adicional, lo cual te permite duplicar o triplicar tu potencia de fuego. En lugar de permitirte realizar una cantidad determinada de ataques especiales, tu habilidad para usar el ataque especial se basa en tus niveles de energía. Esto te permite decidir durante cuánto tiempo y con qué frecuencia desea usar el ataque especial, siempre que tenga suficientes reservas de energía.

Además de potenciar tu arma principal, también puedes captar unidades de apoyo. Puedes reunir hasta cuatro aviones o mechas (dependiendo de los que reúnas), que te acompañarán en un segundo plano hasta que sean destruidos.

El juego tiene fondos increíblemente animados que se perciven más vivos que en otros juegos. Algunos niveles funcionan bien con o sin frame skipping, mientras que en otros el frame skipping causa saltos. Nuevamente, preferí jugar el modo más lento sin el frame skipping activado. Hyper Duel también está disponible para MAME, pero como salió varios años antes que la versión de Saturn, no ofrece características como la banda sonora CD o el “Modo Saturn”. Aparte de eso, es un arcade casi perfecto. De hecho, he finalizado este juego en ODROID, y realmente disfruto jugando a la versión Sega Saturn.

Figura 12 – En este juego, no solo los enemigos pueden disparar cientos de balas

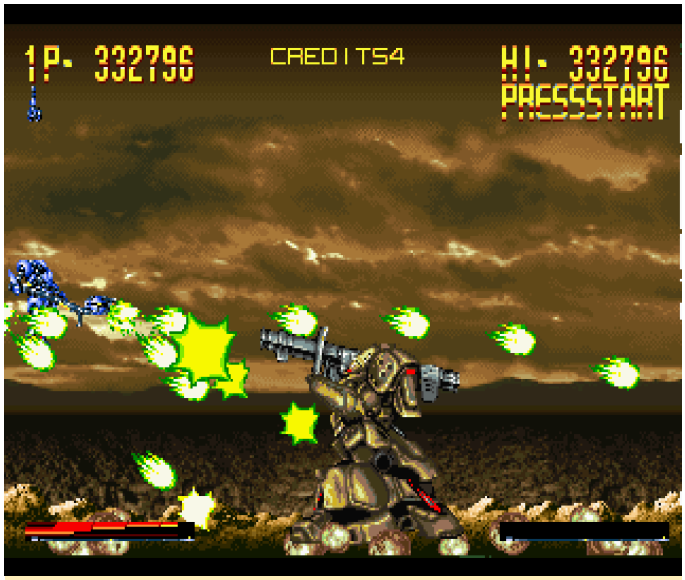


Figura 13 – En este juego, no solo los enemigos pueden disparar cientos de balas

Keio Flying Squadron 2

Keio Flying Squadron 2 es un poco difícil de describir. Combina rompecabezas de plataformas donde saltas y corres con otros elementos, como la acción de disparos con desplazamiento lateral. Este juego es increíble, y me encantan especialmente los colores brillantes y el gracioso estilo anime.

Figura 14 – Portada de Keio Flying Squadron 2 para Sega Saturn ejecutándose en ODROID-XU3

Figura 15 – El primer nivel es del típico estilo de saltar y correr

Figura 16 – En el tercer nivel, haces volar a tu fiel dragón de forma similar a un shmup

De vez en cuando luchas contra tus jefes en su propio escenario. Hay muchos niveles extra donde puedes acumular puntos, si eres lo suficientemente bueno. Normalmente yo pierdo puntos, por el contrario. Una razón por la que me gusta este juego es que ofrece muchos entornos diferentes para jugar. No se juega en el mismo lugar durante horas y horas seguidas. Más bien, cambia de una ubicación, y a menudo de un estilo de juego a otro.

Hay escenas en las que simplemente caminas y te abres paso a través de cada nivel, mientras que en otros niveles te montas en un tren o en un vagón de montaña rusa. Recoges diferentes armas y otros objetos que te ayudan a lo largo del camino. Por

ejemplo, hay un paraguas que te protege de la caída de objetos y te permite deslizarte a una distancia corta cuando saltas. Puedes usar un arco para golpear a los enemigos desde una distancia, o un martillo si prefieres acercarte y ser más directo. Al juntar 100 conejitos de oro obtienes una vida extra, y buscas caminos y cofres ocultos que tienen vida extra u otros objetos útiles.

El juego es bastante exigente, en cuanto a rendimiento, especialmente con transparencias como son las cascadas del primer nivel. El Frame skipping es imprescindible. Puedes notar algunos problemas en el desplazamiento. Girando de izquierda a derecha, o de derecha a izquierda, la pantalla se desplaza hacia un lado y esto puede ser un poco molesto, pero el juego en sí se ejecuta bastante bien, aunque se agradecería un mejor rendimiento. Si te gustan los juegos de correr y saltar y no te sueles frustrar demasiado a la hora de resolver los puzzles, ¡te recomiendo este juego!

Figura 17 – Montando en una montaña rusa

Figura 18 – Buceando bajo el agua. Observa los colores tan brillantes

King of Fighters 96/97

La Sega Saturn tenía muchos juegos arcade, lo que significa que podemos jugar a la famosa serie King of Fighters de Sega Saturn y déjame decirte que funciona bastante bien.

Figura 19 – King of Fighters '96 y '97 para Sega Saturn ejecutándose en ODROID-XU3/XU4

Figura 20 – King of Fighters ‘96 y ‘97 para Sega Saturn ejecutándose en ODROID-XU3/XU4

Figura 21 – Magnifica animación tanto de los luchadores como del escenario de fondo

King of Fighters es en realidad uno de los pocos juegos que requieren o se benefician de la expansión de memoria para Sega Saturn. Hay dos cartuchos de expansión de memoria (8Mb y 32Mb) disponibles para Sega Saturn, que aumenta la memoria de Sega Saturn 1MB o 4MB dependiendo de la extensión.

Mientras que King of Fighters ‘96 solo funciona con la expansión de 1MB (4MB provoca problemas gráficos), King of Fighters ‘97 realmente admite ambas expansiones, aunque solo se requiere 1MB. Ambos juegos funcionan muy bien en ODROID. Ambos juegos también están disponibles para Neo Geo y en la que juegas más o menos a lo mismo, aunque las versiones de Neo Geo son más fluidas. Aún así, es un muy buen arcade al que vale la pena jugar.

Figura 22 – Magnifica animación tanto de los luchadores como del escenario de fondo

Figura 23 – King of Fighters '97: Más combates de la misma calidad: un arcade muy sólido

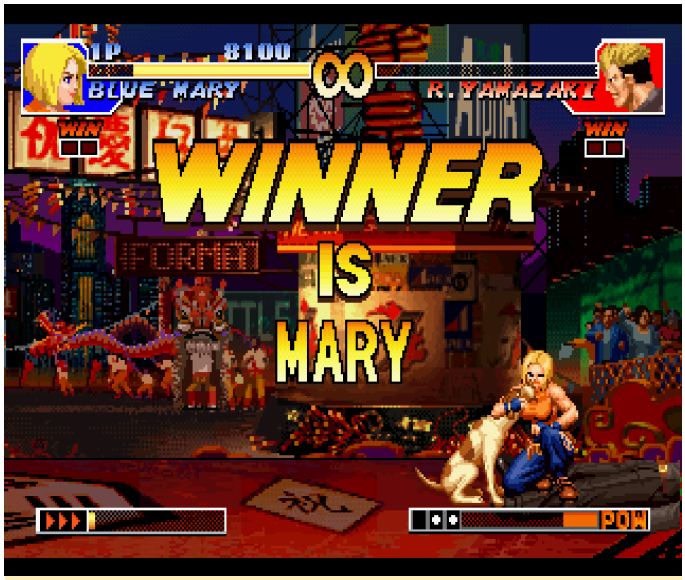


Figura 24 – King of Fighters '97: Más combates de la misma calidad: un arcade muy sólido

Es difícil decidir cuál es la mejor versión, '96 o '97. Me gustan los dos. En mi opinión, es una de las mejores series de lucha. Como Saturn tiene muchos juegos arcade, seguramente habrá más en el futuro.

Menciones honoríficas

Guardian Heroes

Guardian Heroes es un buen beat-'em-up, similar a Streets of Rage o Golden Axe, aunque en un entorno de fantasía con caballeros, magos, esqueletos, etc. Realmente me gusta este juego y pensaba analizarlo con más en detalle. Lamentablemente, se cuelga en el segundo nivel, lo cual hace que el juego no se pueda continuar. Espero que una nueva versión de Yabause solucione esto, ya que es un juego muy divertido y me encantaría ponerlo en mi lista de juegos favoritos para Saturn

Linkle Liver Story

Lamentablemente, este juego solo está disponible en japonés y, como no hablo japonés, no entiendo lo que se tiene que hacer. El juego en sí se le ve una monada, con gráficos muy coloridos al estilo anime y colores cálidos y brillantes. También es muy exigente, de modo que hay una gran cantidad de saltos de imagen, que puedes ver en la animación de tu personaje, pero el juego en sí funciona bastante bien. Tengo la sensación de que, si entendiese japonés, sería un juego de rol (ARPG) muy divertido. Si hablas japonés y te gustan los juegos de rol, te recomiendo probar este juego.

Lode Runner Returns

Este es un buen remake de Lode Runner. Aunque solo está disponible en japonés, es muy divertido. Lo disfruté bastante, pero como es simplemente un pequeño juego de rompecabezas, no lo considero un "imprescindible" para Saturn. Aun así, es un buen juego y se ejecuta perfectamente bien en ODROID. Si te gustan estos juegos de rompecabezas, te lo recomiendo.

Loaded

Loaded es un buen shooter de arriba a abajo en tercera persona similar a los antiguos shooters de Alien para el Amiga, donde atraviesas puertas abiertas, matas enemigos y recolectas tarjetas para abrir más puertas hasta que tengas suficientes tarjetas de acceso para salir del nivel . Puedes seleccionar uno de los seis personajes con diferentes armas y ataques especiales. También hay elementos que puedes recoger, como paquetes de salud, municiones o actualizaciones de armas. Los gráficos cuentan con entornos 3D combinados con sprites de personajes en 2D, lo que hace que parezca un poco anticuado, pero aún así es bastante divertido. Si te gusta los shooter de acción de arriba y abajo, cógelo y pruébalo, funciona muy bien con el ODROID-XU3 / XU4.

Web Kiosk: Cómo crear un Sistema con Pantalla Táctil basado en Chromium

March 1, 2018 By @ZacWolf ODROID-C2, Mecaniqueo

Estaba buscando una plataforma que me permitiera reunir varias funcionalidades de control remoto en un único dispositivo/interfaz. Había probado varios “controles remotos universales”, pero realmente ninguno de ellos me ofrecía la combinación perfecta de posibilidades que estaba buscando. Soy desarrollador Java de oficio, así que decidí crear una aplicación web basada en Java que me permitiera aglutinar los controles de todos mis sistemas de entretenimiento doméstico bajo una única interfaz. Escribiré otro artículo más adelante, pero por ahora debería funcionar con un navegador a modo de pantalla táctil que se inicia automáticamente al encenderse.

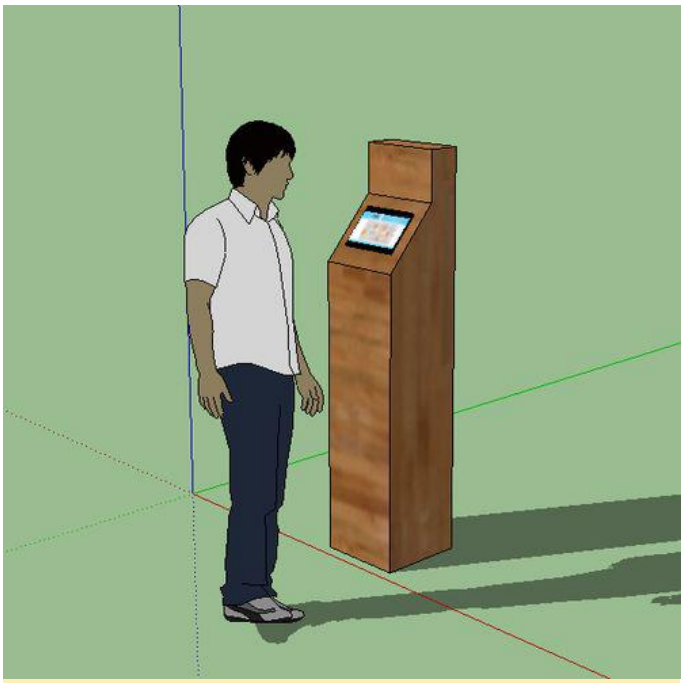


Figura 1 – Web Kiosk

Hardware

Tras realizar algunas investigaciones, decidí optar por un dispositivo ODROID-C2. Este artículo sólo es aplicable a este tipo de dispositivos y no es compatible con una Raspberry Pi por multitud de diferencias entre ambos dispositivos. Sin embargo, la información de este artículo puede servirte a modo de guía para crear un sistema similar basado en Raspberry Pi. El software y las configuraciones detalladas en este artículo son muy específicas para los dispositivos ODROID de Hardkernel.

La lista de componentes de hardware se enumera a continuación. Se pueden obtener directamente de Hardkernel (**http://goo.gl/rsyevF**) o en uno de los muchos distribuidores (**http://goo.gl/7MJduR**). Los elementos que se muestran en las Figuras 2-4 son el hardware básico para el ODROID-C2

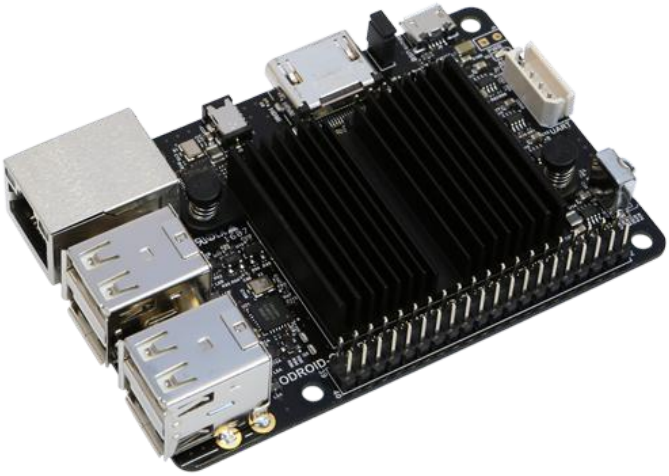


Figura 2 – ODROID-C2

Figura 3 – Módulo eMMC con lector

Figura 4 – Fuente de alimentación

Para su visualización, puedes usar una de las pantallas táctiles soportadas que se muestran en las Figuras 5-8.

Pantalla Multitáctil ODROID VU5 de 5” 800 × 480

Figura 5 – Pantalla de 5” VU5

Pantalla Multitáctil ODROID VU7 de 7” 800 × 480

Figura 6 – Pantalla VU7 de 7”

Pantalla Multitáctil ODROID VU7+ de 7” 1024×600

Figura 7 – Pantalla VU7 Plus de 7”

Pantalla Multitáctil ODROID VU8 de 8” 1024×768

Figura 8 – Pantalla VU8C de 8”

Ten en cuenta que el ODROID VU8C requiere una fuente de alimentación de mayor capacidad 5V/4A

Hardware opcional

2 x Conector de alimentación 2,5 mm

Figura 9 – Conector DC

2 x Módulo 2 Bluetooth

Figura 10 – Módulo Bluetooth

HiFi Shield Plus

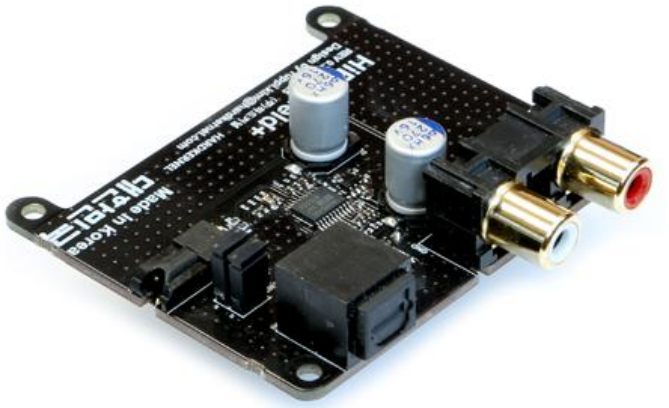


Figura 11 – HiFi shield plus

Aunque he puesto el “Conector DC de 2.5mm” como opcional, a 1.25$, recomiendo comprar dos, ya que son casi imposibles de obtener de otro sitio. De esta forma, dependiendo de la demanda de tu proyecto final, puedes utilizar una mayor fuente de alimentación (una de 5V/4A, por ejemplo), y con un

poco de soldadura cambiar a una clavija más grande en estos enchufes que usan clavijas más pequeñas.

Los módulos Bluetooth también son opcionales. Por defecto, el monitor VU7+ no tiene altavoces, por lo que si desear tener sonido necesitará usar uno de los DAC ODROID (como el HiFi Shield Plus mencionado) o usar Bluetooth para el sonido. Como son baratos, también recomiendo comprar dos en caso de que quiera usar uno para el audio y otro para la interfaz de control remoto, etc.

Una vez que te decidas por los componentes, revisa la página Wiki específica (**http://goo.gl/6Kx2pf**) para obtener detalles sobre cómo usar la pantalla que has elegido. Si necesita ayuda, visita los foros de Hardkernel en **https://forum.odroid.com/index.php**.

Software

La imagen de software cargada en el módulo eMMC de Hardkernel es excesiva para usarla en un sistema Kiosco, de modo que para este desarrollo usaremos la imagen Debian Stretch de @meveric; los detalles sobre la imagen los puedes encontrar en **http://goo.gl/YW21Aa**. Descarga el archivo de imagen C2.img.xz de 93MB desde **http://goo.gl/W9qDmg** o la copia desde **http://goo.gl/B1bTDW**.

A continuación, descarga una herramienta llamada Etcher (**https://etcher.io/**) que te permitirá escribir el archivo de imagen descargado anteriormente en el módulo eMMC. Para hacer esto, utiliza el adaptador de tarjeta SD a eMMC, insértalo dentro de un lector de tarjetas microSD y conecta el lector al ordenador en el cual ejecutas Etcher.

Cuando ejecutes Etcher, primero selecciona la imagen que descargaste y luego selecciona el lector de tarjetas microSD, después presionas Flash. Ten en cuenta que, al seleccionar una unidad en este paso, asegúrate de que sea tu lector de tarjetas microSD, de lo contrario, Etcher sobrescribirá la unidad y los datos no podrán recuperarse. Lee las instrucciones en la página de descargas de Etcher antes de intentar grabar imágenes en cualquier soporte

Además, en Microsoft Windows, cuando insertas el eMMC en un adaptador de tarjeta SD, pueden aparecer ventanas emergentes con la necesidad de formatear la unidad. Ignora estos cuadros de diálogo. Simplemente pulsa cancelar y cierre cualquier ventana. Lo mismo sucederá cuando Etcher empiece a trabajar y cuando termine. Simplemente cierra todos los cuadros de diálogo de Windows que aparezcan. Una vez que Etcher haya finalizado, puedes quitar el adaptador de la tarjeta eMMC2SD, extraer el módulo eMMC e insertarlo en la ranura situada en la parte inferior de tu ODROID.

Para este próximo paso, necesitarás usar un monitor HDMI, no la pantalla táctil seleccionada anteriormente. También necesitarás conectar un teclado al ODROID-C2. El sistema operativo por defecto está configurado para una pantalla de 1920 x 1080p x 60Hz, de modo que usar la pantalla táctil hará que el texto sea ilegible. Además, activa la conexión de red por cable en tu ODROID-C2.

Enciende el dispositivo. Verás una serie de líneas en el momento del arranque. La pantalla se pondrá en blanco, lo cual es normal ya que expande la imagen del software a todo el espacio del módulo eMMC. La pantalla se reanudará con el arranque y luego pasará a la pantalla de inicio de sesión.

Cuando llegues al aviso de inicio de sesión, introduce los datos de autentificación:

```
Username: root
 Password: odroid
```

Introduce los siguientes comandos, confirmando cualquier indicación en el camino:

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

Aproximadamente a la mitad, verás una advertencia sobre la recompilación del kernel. Simplemente selecciona la opción “OK” y presiona Intro.

Cuando el proceso haya finalizado, introduce los siguientes comandos:

```
$ sudo apt-get install net-tools -y && clear
$ ifconfig eth0
```

Este último comando te dará la dirección IP y la dirección MAC para tu ODROID-C2. Guarda esta

información (en un archivo o en papel) para usarla en el futuro, ya que la dirección IP será necesaria en pasos posteriores, y la dirección MAC (precedida por la palabra “ether”) puede ser necesaria si tu servidor DHCP cambia la IP en el futuro.

Ahora, con el ODROID-C2 aún conectado al monitor, ejecuta el siguiente comando:

```
$ sudo reboot
```

Esto reiniciará el ODROID-C2, mostrará los mensajes de inicio y finalmente se te mostrará el aviso inicio de sesión. Puede retirar el teclado del ODROID, ya que no será necesario desde este punto en adelante, pero mantén el monitor conectado al dispositivo hasta que se te indique que conectes la pantalla táctil.

Preparando de la pantalla táctil

Llegados a este punto, vamos a conectarnos al ODROID-C2 desde tu ordenador principal. Deberá estar familiarizado con una aplicación llamada SSH. Si no es así, busca en google “SSH” junto con su nombre de tu sistema operativo. Para Microsoft Windows, la aplicación más sencilla se llama PuTTY, y para OSX, existe un cliente SSH integrado en el propio sistema, al que se puede acceder desde el terminal. Una vez que haya ejecutado SSH, conéctate a la dirección IP que anotaste en el paso anterior.

Figura 12 – Pantalla táctil

Una vez conectado, se te pedirá que inicies sesión con las mismas credenciales utilizadas anteriormente:

```
Username: root
Password: odroid
```

Introduce los siguientes comandos:

```
$ cd ~ && mkdir software && cd software
$ wget -O setup.sh
https://raw.githubusercontent.com/ZacWolf/WebKiosk/master/setup.sh
$ chmod 700 ./setup.sh
$ ./setup.sh
```

Esto te solicitará una nueva contraseña. Luego, configura tu monitor. Una vez que la luz azul haya dejado de parpadear en el ODROID-C2, retira el cable de alimentación. Desenchufa el cable HDMI del monitor.

Figura 13 – Pantalla iluminada

Conecta el cable HDMI de la pantalla táctil al ODROID-C2 y el cable de alimentación. En la pantalla táctil debería aparecer el texto de inicio. Si el texto es ondulado o está degradado, revisa los pasos anteriores.

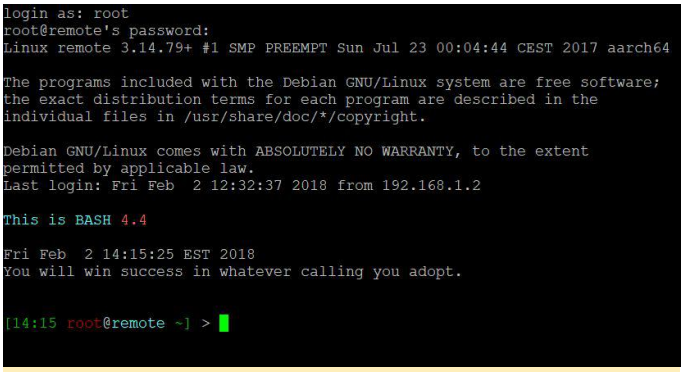


Figura 14 – Pantalla de trabajo

Una vez que el ODROID esté completamente iniciado, deberías ver el mensaje de inicio de sesión en pantalla.

De vuelta a tu aplicación SSH, la conexión habrá finalizado cuando introdujiste el comando shutdown. Tendrás que reanudar la conexión. Cada cliente hace esto de forma diferente, pero en el peor de los casos cierra la ventana y abre una nueva sesión hacia la misma dirección IP que ingresaste antes.

Esta vez en el prompt de inicio de sesión, usa las siguientes credenciales:

```
Username: root
Password: {the password you set in the
previous step}
```

Deberías ver una pantalla similar a la imagen de arriba. De vuelta a SSH, introduce el siguiente comando:

```
$ setupkiosk.sh
```

Esto te guiará a través del proceso de configuración. Te guiará en la configuración de tu idioma por defecto, teclado, zona horaria, nombre de host, NODM (responde Yes) y la configuración del sistema Kiosco. Finalmente, reiniciará automáticamente el ODROID-C2.

Después de reiniciar el ODROID-C2, la pantalla permanecerá en blanco durante aproximadamente un minuto y luego se mostrará en pantalla la página de inicio por defecto.

Solución de problemas

Tuve diversos resultados al intentar hacer funcionar el módulo ODROID WiFi 3, así que opté por la conexión ethernet por cable. Si no logras hacer funcionar el SSH, el ODROID-C2 está configurado para usar DHCP, de modo que lo más probable es que durante el reinicio, el ODROID-C2 cogiera una nueva dirección IP.

Puedes abordar esto de tres formas:

1) Intenta conectarse con el nombre de host que asignaste al ODROID durante la instalación. Si tiene suerte, tu router resolverá esta dirección por ti, pero no te garantizo que funcione.

2) Modificar el archivo /etc/network/interfaces para configurar una dirección IP estática (no recomendado).

3) Lo mejor es configurar una concesión estática en tu servidor DCHP. Lo más probable es que esto se haga en tu router WiFi y se denomine “alquiler estático” o “dirección reservada”. Tendrás que buscar las instrucciones de tu router específico para saber cómo configurar esto, pero es la mejor manera, ya que no tendrá que preocuparse por introducir la información de DNS/subred en el archivo interfaces, ya que eso puede cambiar si actualiza tu router. etc.

Si esto sucede después de que esté ejecutándose el sistema Kiosco, puede que tenga que recurrir a la tercera opción, de modo que es mejor hacerlo de esta forma desde el principio.

Si decides cambiar los monitores, simplemente inicia sesión a través de SSH como usuario root e introduce el comando:

```
$ touchscreen.sh
```

Para comentarios, preguntas y sugerencias, visita el artículo original en **http://www.instructables.com/id/Web-Kiosk/**.

clInfo: Compilando la Utilidad de Puesta a Punto de la GPU OpenCL Esencial para el ODROID-XU4

March 1, 2018 By @hominoid ODROID-XU4, Tutoriales

He estado indagando por qué Clinfo no funciona en el ODROID-XU4, así que he dedicado un tiempo para descubrir el por qué. También me he dado cuenta que hay muchos mensajes en los que se comenta que clinfo no funciona en muchas SBC, y que no encuentran soluciones para ninguna SBC. Así que pensé que podría ser interesante investigar primero esto para asegurarnos de que OpenCL estuviera correctamente configurado antes de intentar solucionar los kernels OpenCL para el proyecto sgminer. La única vez, que había visto que no funcionaba en otras plataformas como x86_64, era cuando había un problema con el controlador de la GPU. Esto es lo que sucede cuando ejecutamos clinfo en un ODROID-XU4:

```
$ sudo apt-get install clinfo
$ sudo clinfo
Number of platforms 0
```

La buena noticia es que hace poco logré que Clinfo funcionara correctamente, y ello me reportó un montón de información sobre la GPU Mali, lo cual es excelente. La información adicional nos ayudará a comprender y adecuar mejor la GPU. Parece que era necesario colocar el archivo ICD del proveedor para la GPU ARM en una ubicación específica.

```
$ sudo clinfo
Number of platforms 1

Platform Name ARM Platform
Platform Vendor ARM
Platform Version OpenCL 1.2 v1.r12p0-
04rel0.03af15950392f3702b248717f4938b82
Platform Profile FULL_PROFILE
Platform Extensions
cl_khr_global_int32_base_atomics
cl_khr_global_int32_extended_atomics
cl_khr_local_int32_base_atomics
cl_khr_local_int32_extended_atomics
```

```
cl_khr_byte_addressable_store
cl_khr_3d_image_writes cl_khr_fp64
cl_khr_int64_base_atomics
cl_khr_int64_extended_atomics cl_khr_fp16
cl_khr_gl_sharing cl_khr_icd cl_khr_egl_event
cl_khr_egl_image cl_arm_core_id cl_arm_printf
cl_arm_thread_limit_hint
cl_arm_non_uniform_work_group_size
cl_arm_import_memory
Platform Extensions function suffix ARM
Platform Name ARM Platform

Number of devices 2
Device Name Mali-T628
Device Vendor ARM
Device Vendor ID 0x6200010
Device Version OpenCL 1.2 v1.r12p0-
04rel0.03af15950392f3702b248717f4938b82
Driver Version 1.2
Device OpenCL C Version OpenCL C 1.2 v1.r12p0-
04rel0.03af15950392f3702b248717f4938b82
Device Type GPU
Device Profile FULL_PROFILE
Max compute units 4
Max clock frequency 600MHz
Device Partition (core)
Max number of sub-devices 0
Supported partition types None
Max work item dimensions 3
Max work item sizes 256x256x256
Max work group size 256
Preferred work group size multiple 4
Preferred / native vector sizes
char 16 / 16
short 8 / 8
int 4 / 4
long 2 / 2
half 8 / 8 (cl_khr_fp16)
float 4 / 4
double 2 / 2 (cl_khr_fp64)
Half-precision Floating-point support
(cl_khr_fp16)
Denormals Yes
Infinity and NANs Yes
Round to nearest Yes
Round to zero Yes
Round to infinity Yes
IEEE754-2008 fused multiply-add Yes
Support is emulated in software No
Correctly-rounded divide and sqrt operations
No
Single-precision Floating-point support (core)
Denormals Yes
Infinity and NANs Yes
Round to nearest Yes
Round to zero Yes
Round to infinity Yes
IEEE754-2008 fused multiply-add Yes
Support is emulated in software No
Correctly-rounded divide and sqrt operations
No
Double-precision Floating-point support
(cl_khr_fp64)
Denormals Yes
Infinity and NANs Yes
Round to nearest Yes
Round to zero Yes
Round to infinity Yes
IEEE754-2008 fused multiply-add Yes
Support is emulated in software No
Correctly-rounded divide and sqrt operations
No
Address bits 64, Little-Endian
Global memory size 2090344448 (1.947GiB)
Error Correction support No
Max memory allocation 522586112 (498.4MiB)
Unified memory for Host and Device Yes
Minimum alignment for any data type 128 bytes
Alignment of base address 1024 bits (128
bytes)
Global Memory cache type Read/Write
Global Memory cache size  Global Memory cache
line 64 bytes
Image support Yes
Max number of samplers per kernel 16
Max size for 1D images from buffer 65536
pixels
Max 1D or 2D image array size 2048 images
Max 2D image size 65536x65536 pixels
Max 3D image size 65536x65536x65536 pixels
Max number of read image args 128
Max number of write image args 8
Local memory type Global
Local memory size 32768 (32KiB)
Max constant buffer size 65536 (64KiB)
Max number of constant args 8
Max size of kernel argument 1024
Queue properties
Out-of-order execution Yes
Profiling Yes
Prefer user sync for interop No
Profiling timer resolution 1000ns
Execution capabilities
Run OpenCL kernels Yes
```

Run native kernels No
printf() buffer size 1048576 (1024KiB)
Built-in kernels
Device Available Yes
Compiler Available Yes
Linker Available Yes
Device Extensions
cl_khr_global_int32_base_atomics
cl_khr_global_int32_extended_atomics
cl_khr_local_int32_base_atomics
cl_khr_local_int32_extended_atomics
cl_khr_byte_addressable_store
cl_khr_3d_image_writes cl_khr_fp64
cl_khr_int64_base_atomics
cl_khr_int64_extended_atomics cl_khr_fp16
cl_khr_gl_sharing cl_khr_icd cl_khr_egl_event
cl_khr_egl_image cl_arm_core_id cl_arm_printf
cl_arm_thread_limit_hint
cl_arm_non_uniform_work_group_size
cl_arm_import_memory
Device Name Mali-T628

Device Vendor ARM
Device Vendor ID 0x6200010
Device Version OpenCL 1.2 v1.r12p0-
04rel0.03af15950392f3702b248717f4938b82
Driver Version 1.2
Device OpenCL C Version OpenCL C 1.2 v1.r12p0-
04rel0.03af15950392f3702b248717f4938b82
Device Type GPU
Device Profile FULL_PROFILE
Max compute units 2
Max clock frequency 600MHz
Device Partition (core)
Max number of sub-devices 0
Supported partition types None
Max work item dimensions 3
Max work item sizes 256x256x256
Max work group size 256
Preferred work group size multiple 4
Preferred / native vector sizes
char 16 / 16
short 8 / 8
int 4 / 4
long 2 / 2
half 8 / 8 (cl_khr_fp16)
float 4 / 4
double 2 / 2 (cl_khr_fp64)
Half-precision Floating-point support
(cl_khr_fp16)
Denormals Yes
Infinity and NANs Yes
Round to nearest Yes
Round to zero Yes
Round to infinity I Yes
IEEE754-2008 fused multiply-add Yes
Support is emulated in software No
Correctly-rounded divide and sqrt operations
No
Single-precision Floating-point support (core)
Denormals Yes
Infinity and NANs Yes
Round to nearest Yes
Round to zero Yes
Round to infinity Yes
IEEE754-2008 fused multiply-add Yes
Support is emulated in software No
Correctly-rounded divide and sqrt operations
No
Double-precision Floating-point support
(cl_khr_fp64)
Denormals Yes
Infinity and NANs Yes
Round to nearest Yes
Round to zero Yes
Round to infinity Yes
IEEE754-2008 fused multiply-add Yes
Support is emulated in software No
Correctly-rounded divide and sqrt operations
No
Address bits 64, Little-Endian
Global memory size 2090344448 (1.947GiB)
Error Correction support No
Max memory allocation 522586112 (498.4MiB)
Unified memory for Host and Device Yes
Minimum alignment for any data type 128 bytes
Alignment of base address 1024 bits (128
bytes)
Global Memory cache type Read/Write
Global Memory cache size Global Memory cache
line 64 bytes
Image support Yes
Max number of samplers per kernel 16
Max size for 1D images from buffer 65536
pixels
Max 1D or 2D image array size 2048 images
Max 2D image size 65536x65536 pixels
Max 3D image size 65536x65536x65536 pixels
Max number of read image args 128
Max number of write image args 8
Local memory type Global
Local memory size 32768 (32KiB)
Max constant buffer size 65536 (64KiB)
Max number of constant args 8

```
Max size of kernel argument 1024
Queue properties
Out-of-order execution Yes
Profiling Yes
Prefer user sync for interop No
Profiling timer resolution 1000ns
Execution capabilities
Run OpenCL kernels Yes
Run native kernels No
printf() buffer size 1048576 (1024KiB)
Built-in kernels
Device Available Yes
Compiler Available Yes
Linker Available Yes
Device Extensions
cl_khr_global_int32_base_atomics
cl_khr_global_int32_extended_atomics
cl_khr_local_int32_base_atomics
cl_khr_local_int32_extended_atomics
cl_khr_byte_addressable_store
cl_khr_3d_image_writes cl_khr_fp64
cl_khr_int64_base_atomics
cl_khr_int64_extended_atomics cl_khr_fp16
cl_khr_gl_sharing cl_khr_icd cl_khr_egl_event
cl_khr_egl_image cl_arm_core_id cl_arm_printf
cl_arm_thread_limit_hint
cl_arm_non_uniform_work_group_size
cl_arm_import_memory
NULL platform behavior

clGetPlatformInfo(NULL, CL_PLATFORM_NAME, ...)
ARM Platform
clGetDeviceIDs(NULL, CL_DEVICE_TYPE_ALL, ...)
Success [ARM]
clCreateContext(NULL, ...) [default] Success
[ARM]
clCreateContextFromType(NULL,
CL_DEVICE_TYPE_CPU) No devices found in
platform
clCreateContextFromType(NULL,
CL_DEVICE_TYPE_GPU) Success (2)
Platform Name ARM Platform
Device Name Mali-T628
Device Name Mali-T628
clCreateContextFromType(NULL,
CL_DEVICE_TYPE_ACCELERATOR) No devices found
in platform
clCreateContextFromType(NULL,
CL_DEVICE_TYPE_CUSTOM) No devices found in
platform
clCreateContextFromType(NULL,
CL_DEVICE_TYPE_ALL) Success (2)
Platform Name ARM Platform
Device Name Mali-T628
Device Name Mali-T628
ICD loader properties

ICD loader Name OpenCL ICD Loader
ICD loader Vendor OCL Icd free software
ICD loader Version 2.2.8
ICD loader Profile OpenCL 1.2
NOTE: your OpenCL library declares to support
OpenCL 1.2,
but it seems to support up to OpenCL 2.1 too.
```

En las plataformas x86, parece que la instalación de los archivos del proveedor ICD y las librerias OpenCL se realiza durante la instalación del controlador. Esta podría ser la razón por la que no he visto que Clinfo funcione en ARM. ¿Debería el archivo ICD ser parte de la configuración realizada por Hardkernel como proveedor? Instala la buffer de imagen y clinfo si aún no está:

```
$ sudo apt-get install mali-fbdev clinfo
```

A continuación, configura el archivo ICD del proveedor, después al ejecutar clinfo debería reportar correctamente la información de la GPU Mali:

```
$ sudo mkdir /etc/OpenCL
$ sudo mkdir /etc/OpenCL/vendors
$ echo "/usr/lib/arm-linux-gnueabihf/mali-
egl/libOpenCL.so" >
/etc/OpenCL/vendors/armocl.icd
```

Aunque las librerrias OpenCL y los archivos include no son necesarios para clinfo, no existe una ubicación estándar para su instalación. He leído muchas cosas, aunque este post parece tener la mejor solucion, parece estar pasado de moda. Los siguientes pasos muestran específicamente cómo usar las ubicaciones consensuadas que AMD, NVIDIA e INTEL siguen (ed) para las librerías y los archivos include. No se necesitan referencias explícitas para vincular las librerías de OpenCL.

Primero, descarga el código fuente de ComputeLibrary, o use la libreria ARM Computer Vision y Machine Learning existente:

https://github.com/ARM-software/ComputeLibrary

```
$ cd /opt
$ sudo tar -xvzf ~/arm_compute-v18.01-
bin.tar.gz
$ cd ~/
$ rm arm_compute-v18.01-bin.tar.gz
$ sudo cp /opt/arm_compute-v18.01-
bin/include/CL/* /usr/include/CL/
$ sudo mkdir /usr/lib/OpenCL
$ sudo mkdir /usr/lib/OpenCL/vendors
$ sudo mkdir /usr/lib/OpenCL/vendors/arm
$ sudo cp /opt/arm_compute-v18.01-
bin/lib/linux-armv7a-cl/*
/usr/lib/OpenCL/vendors/arm/
$ sudo echo "/usr/lib/OpenCL/vendors/arm" >
/etc/ld.so.conf.d/opencl-vendor-arm.conf
$ sudo ldconfig
```

Toda la ayuda y comentarios son bienvenidos y apreciados, el hilo de soporte del foro se encuentra en https://forum.odroid.com/viewtopic.php?f=95&t=30141.

Buscadores, Mineros y 49: Minería GPU-CPU Dual en el ODROID-XU4/MC1/HC1/HC2

March 1, 2018 By @hominoid ODROID-HC1, ODROID-MC1, ODROID-XU4, Tutoriales

Hay muchas personas que usan XU4/MC1/HC1/HC2 para la minería de criptomonedas por CPU, entonces, ¿Podría llegar a ser mejor que usar tu GPU para la mineria? El rendimiento del algoritmo no es viable para muchas de las monedas más populares, pero en el contexto adecuado podría tener sentido, como son las monedas nuevas o monedas con poca dificultad. Aunque no sirva para otra cosa, básicamente es otra herramienta divertida para tu colección de herramientas.

Después de analizar las opciones disponibles, empecé a trabajar en la compilación del fork de SGMiner de Genesis Mining. SGMiner-GM 5.5.5 es un cripto minador de GPU OpenCL y es la versión más reciente de SGMiner. Ha sido por un tiempo, compatible con más algoritmos de criptografía que las versiones anteriores, y no tiene una tarifa de uso. Incluye minería para Credits, Scrypt, NScrypt, X11, X13, X15, X15, Keccak, Quarkcoin, Twecoin, Fugue256, NIST, Fresh, Whirlcoin, Neoscrypt, WhirlpoolX, Lyra2RE, Lyra2REV2, Pluck, Yescrypt, Yescrypt-multi, Blakecoin, Blake, Vanilla, Ethash, Cryptonight y Equihash.

La código fuente del programa está disponible para su descarga en **https://goo.gl/Gp25ep**, y al hilo de soporte del foro se puede acceder desde **https://goo.gl/hDVmbF**.

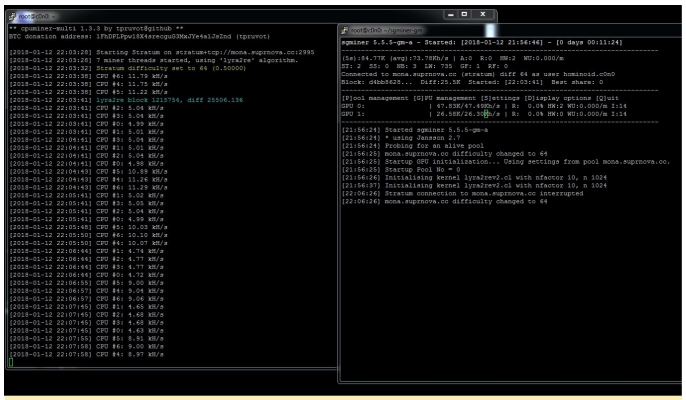


Figura 1 – Sistema dual XU4 minando Monacoin con CPUMiner-Multi y SGMiner-GM 5.5.5 utilizando Lyra2REv2

Es posible, junto con CPUMiner-Multi o un minero específico de monedas como VeriumMiner, minar CPU y GPU al mismo tiempo. No se han realizado pruebas exhaustivas, pero se han completado con éxito varias configuraciones de minería dual, incluidas scrypt2, Lyra2REv2 y cryptonight (solo CPU), individual y minería en grupo. Es posible montar una mina individual en una y un conjunto de minas en otro mientras se ejecutan otros algoritmos de cifrado. Las temperaturas de la CPU durante la minería dual requieren grandes núcleos de CPU para moderar la velocidad, así que, por favor, presta atención a las temperaturas si decides probar esto.

CPUMiner-Multi soporta más de 45 algoritmos de cifrado, lo que lo hace bastante útil para la minería de múltiples algoritmos. Si aún no está familiarizado con él, échale un vistazo en **https://goo.gl/hUQG3F**. Otro programa muy útil de minería dual para aquellos que extraen Verium (VRM) es un fork de VeriumMiner por fireworm71 en **https://goo.gl/6ET7bj**.

El sistema de minería puede ejecutarse en 1 y 3 vías al mismo tiempo, lo que permite una mejor utilización de la memoria. Parece que si la GPU (SGMiner) se inicia primero, terminarás con más memoria para poder utilizar el algoritmo de criptografía por CPU mientras se realiza la minería dual. A continuación, se muestra la línea de comandos utilizada para la minería Verium (4 núcleos grandes de 3 vías y 1 núcleo pequeño de 1 vía), al mismo tiempo que la GPU extrae Monacoin con Lyra2REv2:

```
$ ~/cpuminer -o stratum+tcp://yourpool.na:port
-u username -p password --randomize --no-
redirect -t 4 -1 1 --cpu-affinity-stride 1 --
cpu-affinity-default-index 4 --cpu-affinity-
oneway-index 0
```

Compilar SGMiner-GM 5.5.5

Las siguientes instrucciones son las típicas para SGMiner, con la excepción de las modificaciones del archivo fuente. Tienes información general y de configuración en la wiki de instalación para Ubuntu 16.04 x86 en **https://goo.gl/qnFmb2**. Primero, descarga la última liberira ARM Computer Vision and Machine Learning desde **https://goo.gl/LdFvy5**.

Ten en cuenta que el paquete sin comprimir no cabe en una tarjeta SD de 8GB. Puedes eliminar las librerías innecesarias de ./arm_compute-v17.12-bin/lib para reducir su tamaño. Mantén las librerías linux-armv7a y elimina los archivos android- * y linux-arm8 *. La instalación por defecto esta en /usr/lib/arm_compute-v17.12-bin

```
$ cd /usr/lib
$ tar -xvzf ~/arm_compute-v17.12-bin.tar.gz
$ cd ~/
$ rm arm_compute-v17.12-bin.tar.gz
```

Descarga el SDK APP AMD desde **https://goo.gl/cZeDJc**. Esto es para una instalación root desde ~ /. Consulta las notas de instalación para una instalación no root en **https://goo.gl/Hw7vkP**. La instalación por defecto esta en /opt/AMDAPPSDK-3.0.

```
$ tar -xvjf AMD-APP-SDKInstaller-v3.0.130.136-
GA-linux32.tar.bz2
$ ./AMD-APP-SDK-v3.0.130.136-GA-linux32.sh
$ rm AMD-APP-SDK-v3.0.130.136-GA-linux32.sh
$ rm AMD-APP-SDKInstaller-v3.0.130.136-GA-
linux32.tar.bz2
```

Descarga el AMD Display Library (ADL) SDK desde **https://goo.gl/CqhZq1**:

```
$ apt-get install unzip
$ unzip ADL_SDK_V10.2.zip -d
/opt/ADL_SDK_V10.2
$ rm ADL_SDK_V10.2.zip
```

Instala las dependencias con el siguiente comando:

```
$ apt-get install automake autoconf pkg-config
$ libcurl4-openssl-dev libjansson-dev libssl-
dev libgmp-dev make $ g++ git libgmp-dev
libncurses5-dev libtool mali-fbdev
```

Ten en cuenta que mali-fbdev es necesario si se utiliza una imagen minimalista de Ubuntu; de lo contrario, utiliza Mali-T628-ODROID para la imagen minimalista de Debian.

Descarga Git y mueve las cabeceras con los siguientes comandos:

```
$ git clone
https://github.com/genesismining/sgminer-gm
$ cd sgminer-gm
$ cp /opt/ADL_SDK_V10.2/include/*.h ./ADL_SDK
```

Algunas de las versiones de SGMiner que he visto tienen problemas de compilación similares; otras tienen problemas adicionales. Esto es lo que se debe cambiar en el código fuente de SGMiner-5.5.5 para que se compile correctamente. Realiza las siguientes modificaciones en 4 archivos:

Cambiar la línea 32 de kernel/lyra2rev2.cl de:

```
#pragma OPENCL EXTENSION cl_amd_printf :
enable
```

a:

```
#pragma OPENCL EXTENSION cl_amd_printf :
disable
```

Cambiar kernel/skein256.cl empezando por la línea 49-59 de:

```
__constant static const int ROT256[8][4] =
 {
 46, 36, 19, 37,
 33, 27, 14, 42,
 17, 49, 36, 39,
 44, 9, 54, 56,
 39, 30, 34, 24,
 13, 50, 10, 17,
 25, 29, 39, 43,
 8, 35, 56, 22,
 };
```

a:

```
__constant static const int ROT256[8][4] =
 {
 {46, 36, 19, 37},
 {33, 27, 14, 42},
 {17, 49, 36, 39},
 {44, 9, 54, 56},
 {39, 30, 34, 24},
 {13, 50, 10, 17},
 {25, 29, 39, 43},
 {8, 35, 56, 22}
 };
```

Cambiar la línea 58 de ocl/build_kernel.c de:

```
sprintf(data->compiler_options, "-I "%s" -I
"%s/kernel" -I "." -D WORKSIZE=%d",
```

a:

```
sprintf(data->compiler_options, "-I %s -I
%s/kernel -I . -D WORKSIZE=%d",
```

Cambiar la línea 66 de:

```
strcat(data->compiler_options, " -I "");
```

a:

```
strcat(data->compiler_options, " -I ");
```

Cambiar la línea 68 de:

```
strcat(data->compiler_options, """);
```

a:

```
strcat(data->compiler_options, "/");
```

Cambiar algorithm/cryptonight.c empezando en la línea 139 de:

```
__asm__("mul %%rdx":
"=a" (lo), "=d" (hi):
"a" (a), "d" (b));
```

a:

```
//__asm__("mul %%rdx":
//"=a" (lo), "=d" (hi):
//"a" (a), "d" (b));
```

Cryptonight se vuelve inoperativo anulando la optimización del montaje. No usar Cryptonight, WhirlpoolX, Ethash o Equihash, ya que después de solucionar el tema del montaje extendido, se compila,

pero aparece otro problema que no es fácil de corregir. Parece que estos kernels OpenCL usan extensiones AMD OpenCL que no son compatibles con la plataforma ARM y, por lo tanto, no pueden compilar y activar la GPU. Es posible que sea necesario volver a escribir los kernels para que funcionen. Hace falta investigar más afondo, ya que Cryptonight es utilizado por más monedas y puede ser económicamente viable para la minería por GPU y CPU en este dispositivo. Continuaré trabajando en ello.

Introduce los siguientes comandos en el directorio base de SGMiner-GM para finalizar la compilación:

```
$ git submodule init
$ git submodule update
$ autoreconf -fi
$ CFLAGS="-Os -Wall -march=native -std=gnu99 -
mfpu=neon" LDFLAGS="-L/usr/lib/arm_compute-
v17.12-bin/lib/linux-armv7a-neon-cl"
./configure --disable-git-version --disable-
adl --disable-adl-checks --prefix=/opt/sgminer
```

En el resumen de la configuración, debería ver que se ha encontrado OpenCL y que el minado por GPU está activado. Si no es así, OpenCL no está configurado correctamente y debe corregirse antes de continuar. Las imágenes Ubuntu de Hardkernel vienen con la configuración de OpenCL. Esta compilación se realizó en ubuntu-16.04.3-4.14-minimal-odroid-xu4-20171213.img con éxito. Verifica tus pasos con detenimiento.

```
------------------------------------------
-------------------------
sgminer 5.5.5-gm-a
------------------------------------------
-------------------------

Configuration Options Summary:

Use git version......: no
libcurl(GBT+getwork).: Enabled: -lcurl
curses.TUI...........: FOUND: -lncurses
OpenCL...............: FOUND. GPU mining
support enabled
ADL..................: Detection overrided.
GPU monitoring support DISABLED

Compilation..........: make (or gmake)
CPPFLAGS.............:
CFLAGS...............: -Os -Wall -march=native
-std=gnu99 -I/opt/AMDAPPSDK-3.0/include
LDFLAGS..............: -L/usr/lib/arm_compute-
v17.12-bin/lib/linux-armv7a-neon-cl -lpthread
LDADD................: -ldl -lcurl
submodules/jansson/src/.libs/libjansson.a -
lpthread -L/opt/AMDAPPSDK-3.0/lib/x86 -lOpenCL
-lm -lrt

Installation...........: make install (as root
if needed, with 'su' or 'sudo')
prefix...............: /opt/sgminer
```

Crea e instala el paquete:

```
$ make -j5
$ make install
```

Pruebas rápidas

```
$ ./sgminer --version

$ sgminer 5.5.5-gm-a

$ ./sgminer -n

[20:41:54] CL Platform vendor: ARM
[20:41:54] CL Platform name: ARM Platform
[20:41:54] CL Platform version: OpenCL 1.2
v1.r12p0-
04rel0.03af15950392f3702b248717f4938b82
[20:41:54] Platform devices: 2
[20:41:54] 0 Mali-T628
[20:41:54] 1 Mali-T628
[20:41:54] 2 GPU devices max detected
```

Según la Wiki de instalación, “La primera fallará si faltan las librerías, de modo que, si obtenemos un número de versión, entonces es que el binario compilado se está ejecutando adecuadamente en nuestro sistema. La segunda comprueba si hay dispositivos GPU OpenCL en la plataforma OpenCL por defecto. Si ambos comandos funcionan sin error y este último indica la plataforma OpenCL correcta, estás en el buen camino hacia una instalación funcional”.

Suponiendo que tiene cuentas configuradas para minar sólo o en conjunto, una forma rápida de configurarlo es usar la línea de comandos en lugar de

un archivo de configuración. Puedes obtener más información sobre todo esto en la wiki de instalación y en ./sgminer/doc/configuration.md. Usar un simple script para realizar pruebas es rápido y fácil ya que no es necesario configurar ciertas variables.

```
#!/bin/bash

$ export GPU_FORCE_64BIT_PTR=0
$ export GPU_USE_SYNC_OBJECTS=1
$ export GPU_MAX_ALLOC_PERCENT=100
$ export GPU_SINGLE_ALLOC_PERCENT=100
$ export GPU_MAX_HEAP_SIZE=100
$ ./sgminer -k algorithm -o
stratum+tcp://pool.na:port -u user.worker -p
password -I 14 -w 64 -d 0,1 --thread-
concurrency 8192
```

La intensidad (-I 14) y la cantidad de trabajo (-w 64) pueden ajustarse para un mejor (o peor) rendimiento. Puesto que la Mali-T628 tiene dos dispositivos, ambos son seleccionados (-d 0,1). El dispositivo 0 tiene 4 núcleos y el Dispositivo 1 tiene 2 núcleos. Tienes más información sobre la configuración de la GPU en ./sgminer/doc/gpu.md.

Cuando inicias SGMiner, hay una larga demora de unos 30 a 40 segundos mientras que se crean y cargan los kernels para ambos dispositivos GPU. En la pantalla solo aparecen un par de líneas y puede parecer que se ha colgado. Sé paciente. Luego se volverá negra durante aproximadamente 10-15 segundos, tras lo cual se mostrará la blasfemia. Para la prueba, puede usar un -T en la línea de comando para desactivar la interfaz de blasfemia del terminal y usar texto simple. Se muestra bastante información durante el proceso de iniciación. Son normales algunos errores de hardware mientras se ejecuta. Si observas que se reciben muchos errores de hardware, intenta ajustar la intensidad, ya que cada algoritmo es diferente y debe afinarse. Aquí es donde es muy útil usar un archivo de configuración. Puede usar diferentes configuraciones para diferentes algoritmos y grupos.

Mi clúster XU4/MC1 está dividido en cuatro grupos térmicos y funciona a unas velocidades que permiten mantener la actividad 24/7/365 en el rango de 70-79 °C. Los MC1 son los que menos se calientan de todos los ODROID. La Figura 2 muestra una minería de conjunto dual Verium con scrypt (CPUMiner) y Monacoin con Lyra2REv2 (SGMiner) — dos horas para conocer el rendimiento. Con esta combinación y frecuencia, la tasa de hash de la CPU disminuyó aproximadamente en un 19% mientras que el minado por GPU y la tasa de hash de la GPU disminuyeron aproximadamente un 4% durante el minado por CPU. Esto, por supuesto, variará dependiendo el algoritmo y otros factores de configuración. Se han realizado pruebas de 24 horas con treinta ODROIDs con doble minado sin problemas.

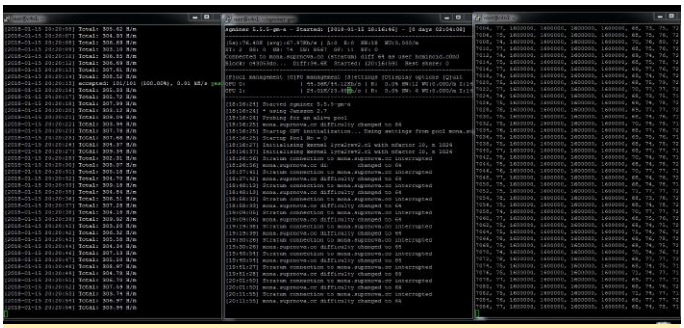


Figura 2 – Ejecución de SGMiner en el clúster ODROID-XU4

Independientemente del hecho de que algunos de los kernels OpenCL no funcionen, esta sigue siendo la mejor opción que conozco para la minar por GPU en XU4/MC/HC1/HC2. La buena noticia es que hay muchos otros algoritmos de criptografía que admite SGMiner, pero solo unos pocos han sido probados. Haznos saber a todos si encuentras más que tengan algún problema. Cuando se hagan más avances para que el resto de kernels funcionen, se publicarán en el foro https://forum.odroid.com/viewtopic.php?f=98&t=29571.

Prueba de Minería Dual GPU-CPU

La Prueba de minería Dual CPU-GPU está diseñada para estudiar los efectos del cambio de frecuencia de la CPU sobre la temperatura de funcionamiento de la GPU durante 1 hora y 50 minutos con una temperatura ambiente de aproximadamente unos 76 F (24,44 C). Durante los primeros diez minutos de la prueba, solo se usó la GPU para determinar el punto de partida de la temperatura de funcionamiento utilizando el conjunto Monacoin y Lyra2REv2 (SGMiner) con las siguientes opciones:

```
-I 14 -w 64 -d 0,1 --thread-concurrency 8192
```

Durante del resto de la prueba de minería dual, se usaron Verium CPU con Scrypt (8 hilos CPUMiner sin afinidad) Monacoin GPU y Solo con Lyra2REv2 (hilo de 1 SGMiner) con las siguientes opciones:

```
-I 14 -w 64 -d 0,1 --thread-concurrency 8192
```

La frecuencia de la CPU se redujo en 100 Mhz cada diez minutos hasta 1,2 Ghz y luego aumenta 100 Mhz cada cinco minutos hasta 1.9 Ghz. Luego cambio a 1.6 Ghz para el resto de la prueba.

La GPU minó a 51 °C durante los primeros diez minutos de la prueba, y luego aumentó con la temperatura de los núcleos de la CPU estancándose en cada cambio de frecuencia. La GPU nunca superó los 72 °C, excepto por unos breves picos de 74 °C. La temperatura baja en la GPU durante la prueba, parece estar correlacionada con el cambio de frecuencia de los núcleos de la CPU. La velocidad de hash de la GPU (71 kh / s) se mantuvo estable durante toda la prueba, mientras que la tasa de hash de la CPU varió en función de la configuración de la frecuencia, como era de esperar.

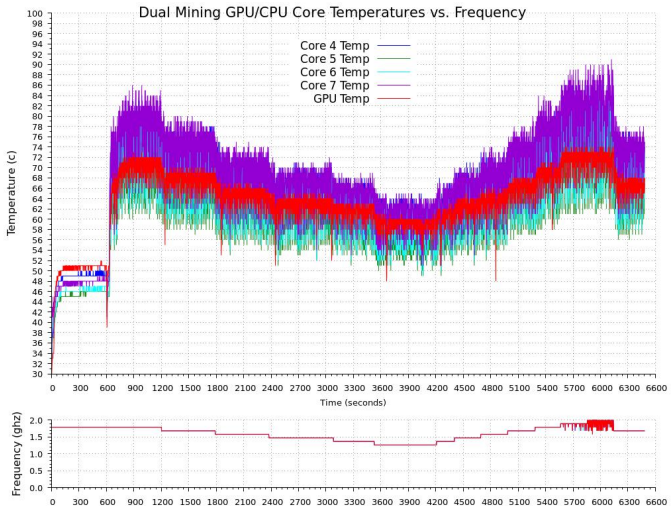


Figura 3 – Temperaturas vs Frecuencias de los núcleos GPU/CPU en Minería Dual

Una nota rápida sobre los valores rechazados para los nuevos mineros. Existen numerosas razones por las que puede recibir valores rechazados u obsoletos. Aunque podría ser un error, la mayoría de las veces están causados por la latencia de la red. Existen dos escenarios posibles, donde el primero es que tu plataforma está minando un bloque, encuentra un recurso compartido válido y lo envía a la red. Mientras tanto, el bloque se resolvió y se emitió un nuevo bloque y tarea. Cuando se envía tu acción, está obsoleta y es rechazada. El indicador ST en SGMiner indica la cantidad de acciones obsoletas que has enviado. Esto no es un error, y no hay mucho que puedas hacer al respecto. Puede reducir la posibilidad de tener problemas al no minar un conjunto en el otro lado del mundo, creando así más latencia para tu minero. Busca un servidor en tu país o tan cerca cómo te sea posible. La mayoría de las agrupaciones ofrecen múltiples servidores geográficamente dispersos por este motivo.

El segundo escenario es que tengas suerte y encuentras un bloque, pero cuando se presenta la solución, otra persona ya envió una solución válida antes que tú. Ahora tiene un bloque huérfano. Estas son dos de las causas más comunes, y a menos que obtenga muchos rechazos, no debería ser un problema. Si está recibiendo muchos rechazos y tiene muchos errores GPU, probablemente estés presionando demasiado tu GPU y necesitas ajustar la intensidad, la carga trabajo o la cantidad de subprocesos. Simplemente porque puedas minar una moneda no significa que encuentre una moneda minando solo, o una acción válida si minas en grupo. Un buen ejemplo sería probar y extraer bitcoin con algo que no sea un dispositivo ASIC (Application Integrated Integrated Circuit). La velocidad y la dificultad de hash van más allá de la capacidad del hardware, a menos que tengas mucha suerte. Si es así, ¡Para! ¡Te acaba de tocar la lotería!

La mayoría de grupos no mostrarán la tasa de hash o incluso que estás minando hasta que no envíes un valor válido. Cuando el bloque cambia y no se han enviado nuevos valores compartidos, vuelves a no aparecer en el grupo. Si está extrayendo una moneda económicamente mal emparejada para el dispositivo, no te sorprendas si tu minero no está presente en el grupo. Encuentra una moneda que quieras minar y empareja el dispositivo HW apropiado para la dificultad y la tasa hash. Alternativamente, utiliza todos los dispositivos HW que tengas disponibles, comprueba qué monedas puedes minar con tu

potencial. ¡Diviértete y buena suerte con la micro-minería!

Creando un servidor NTP usando GPS/PPS

March 1, 2018 By Joshua Yang Linux, ODROID-XU4, Tutoriales

Puedes montar tu propio servidor Network Time Protocol (NTP) usando GPS y PPS en tu ODROID. Este sistema te proporciona la hora exacta que puede serte muy útil para casos específicos. Los relojes atómicos de los satélites GPS son monitorizados y comparados con los "relojes maestros" por la Sección de Control Operativo del GPS; esta hora del GPS está controlada dentro de un microsegundo de la hora Universal. Nuestro receptor GPS proporciona 1 Pulso por Segundo, o PPS, enviando la señal aunque necesitas realizar algo de soldadura para poner a la vista este pin. Este pulso tiene un edge de elevación alineado con el segundo GPS, y se usa para sancionar los relojes locales para mantener la sincronización con el Tiempo Universal (UT). Como resultado, nuestro servidor local puede tener una hora muy precisa con menos de 10 microsegundos de tolerancia. Antes de empezar, necesitas poner a la vista la señal PPS del receptor GPS.

Poner a la vista la señal PPS

En primer lugar, desmonta el módulo GPS quitando los 4 tornillos de la parte posterior del módulo GPS. Éstos están cubiertos por una pegatina fijada en la parte posterior del receptor. Puedes averiguar dónde están pasando los dedos por la pegatina hasta detectar los huecos. Debes haber 2 huecos en el extremo superior y otros 2 en el inferior de la pegatina. Tras encontrar los agujeros, corta la pegatina y separe la parte cortada para poder desenroscar los 4 tornillos.

Figura 1 – Herramientas que necesitarás

Figura 2 – Pegatina despegable para acceder a los tornillos

Figura 3 – Pegatina retirada para poder acceder a los tornillos

Figura 4 – Todos los tornillos retirados

Abre el módulo para que se muestre la placa PCB, que es la que tienes que soldar para presentar el PPS.

Figura 5 – Tapa retirada

Esta es una parte muy importante de la guía, ya que tiene que soldar un cable puente a un pin específico del chip. La ubicación del pin PPS que necesitamos se muestra en la Figura 6. Ten mucho cuidado de no crear un cortocircuito.

Figure 6 – Pin PPS en la PCB principal

Si lo ha hecho bien, puede pelar el cable tal y como se muestra en la Figura 7.

Figura 7 – Cable conectado

A continuación, tenemos que montar y conectar todo nuevamente. Coloca la PCB de nuevo dentro de carcasa como estaba antes y vuelve a atornillar la carcasa.

Figura 8 – Colocada la PCB nuevamente dentro de la carcasa

Figura 9 – Tapa trasera atornillada de nuevo

Conecta el cable puente al pin GPIO 18 del shifter shield XU4. Ten en cuenta que este es el shifter shield que garantiza que los niveles de voltaje del pin PPS coinciden con lo esperado.

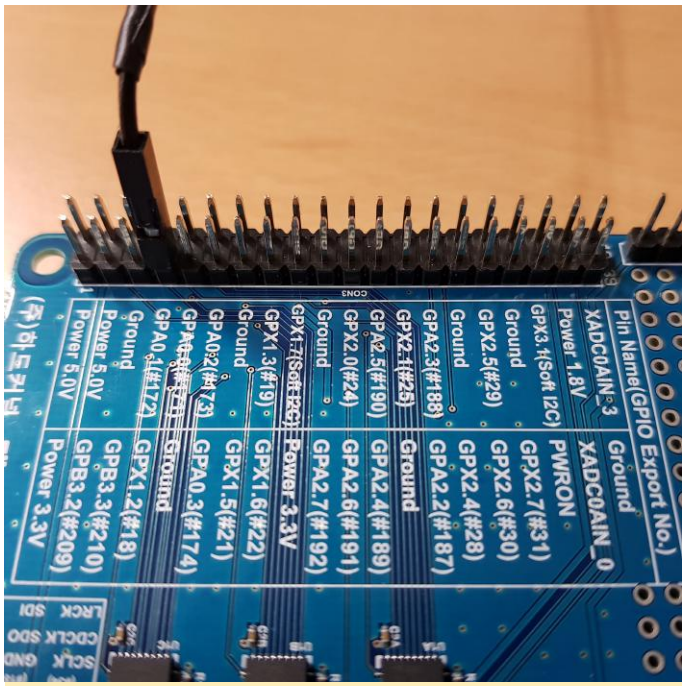


Figura 10 – Pin PPS conectado al XU4

Conecta el cable USB al ODROID-XU4, conecte el cable LAN y el cable de alimentación también.

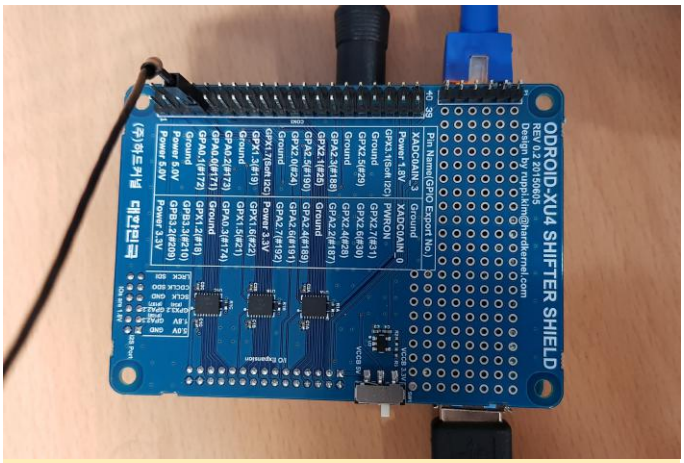


Figura 11 – Pin y USB conectados al XU4

Nuestro kernel principal no es totalmente compatible con PPS desde GPIO. Es necesario realizar algunas configuraciones de software compilando tu propio kernel en tu ODROID-XU4. Para empezar, prepara el código fuente del kernel Linux desde Github, e instala las herramientas necesarias para compilar un kernel nuevo:

```
$ sudo apt update && sudo apt install git gcc
g++ build-essential libncurses5-dev bc
```

Hazte con el código fuente del kernel de Linux desde nuestro repositorio oficial de Github en https://github.com/hardkernel/linux:

```
$ git clone --depth 1 https://github.
com/hardkernel/linux.git -b odroidxu4-4.14.y
odroidxu4-4.14.y
$ cd odroidxu4-4.14.y
```

Añade soporte PPS editando el archivo arch/arm/boot/dts/exynos5422-odroidxu4.dts con el fin de añadir un nuevo dispositivo que recibe PPS desde el GPIO # 18:

```
$ vi arch/arm/boot/dts/exynos5422-
odroidxu4.dts
```

Agrega los siguientes contenidos al archivo:

```
dummy_codec : spdif-transmitter {};

/* add for pps-gpio */
pps {
  compatible = "pps-gpio";
  gpios = <&gpx1 2 GPIO_ACTIVE_HIGH>;
  status = "okay";
};
```

A continuación, crea un menuconfig personalizado:

```
$ make odroidxu4_defconfig
$ make menuconfig
```

Localiza la opción y activala con la tecla de espacio, tal y como se muestra en la Figura 12, luego guarda y salte.

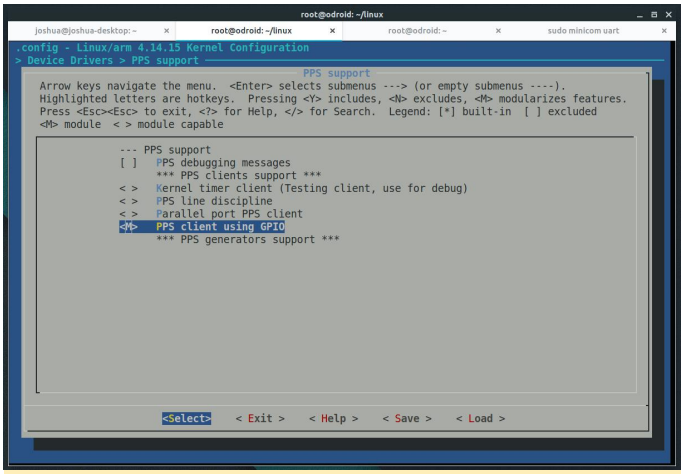


Figura 12 – Habilitado el soporte PPS

Si tu configuración personalizada funciona bien, se crearían nuevos dispositivos en /dev. Vamos a verlos:

```
$ ls -al /dev/{ttyACM*,gps*,pps*}

crw------- 1 root root    248, 0 Jan 31 14:21
/dev/pps0
crw-rw---- 1 root dialout 166, 0 Jan 31 14:53
/dev/ttyACM0
lrwxrwxrwx 1 root root          7 Jan 31 14:21
/dev/ttyACM99 -> ttySAC0
```

Si alguno de los elementos del ejemplo anterior no aparece, es que has hecho algo mal, debes intentar configurar y compilar el kernel nuevamente. Si aparecen todos ellos, crea archivos de enlace lógico para usarlos más adelante:

```
$ sudo ln -sF /dev/ttyACM0 /dev/gps0
$ sudo ln -sF /dev/pps0 /dev/gpspps0
$ ls -al /dev/{ttyACM*,gps*,pps*}

lrwxrwxrwx 1 root root        12 Jan 31 15:50
/dev/gps0 -> ttyACM0
lrwxrwxrwx 1 root root         9 Jan 31 15:51
/dev/gpspps0 -> /dev/pps0
crw------- 1 root root    248, 0 Jan 31 15:50
/dev/pps0
crw-rw---- 1 root dialout 166, 0 Jan 31 15:50
/dev/ttyACM0
lrwxrwxrwx 1 root root          7 Jan 31 15:50
/dev/ttyACM99 -> ttySAC0
```

Asegúrate de que tu resultado es similar al ejemplo anterior, luego instala los paquetes relacionados con GPS y configúralos:

```
$ sudo apt install gpsd gpsd-clients
$ sudo dpkg-reconfigure gpsd
```

A continuación, necesitas probarlos:

```
$ sudo gpsmon /dev/gps0
```

En la Figura 13 se muestra una captura de pantalla de ejemplo con “gpsmon/dev/gps0”. Has de esperar algo más de 5 minutos para obtener la información del GPS correctamente.

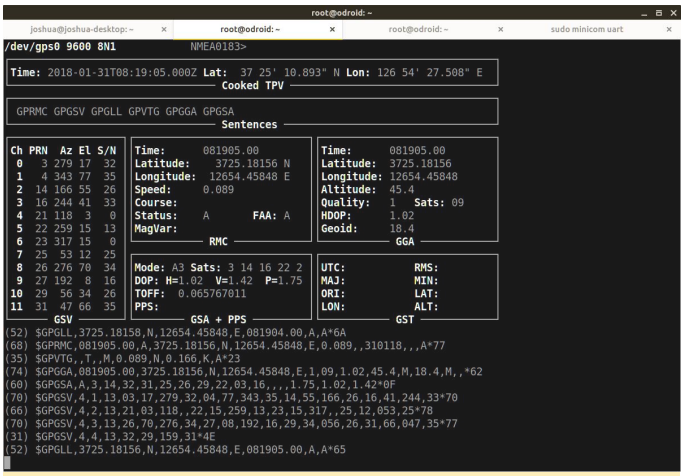


Figura 13 – Datos NMEA

A continuación, instala las herramientas PPS, luego prueba nuestro ppstest en /dev/gpspps0:

```
$ sudo apt install pps-tools
$ sudo ppstest /dev/gpspps0

trying PPS source "/dev/gpspps0"
found PPS source "/dev/gpspps0"
ok, found 1 source(s), now start fetching
data...
source 0 - assert 1517363638.431673232,
sequence: 130 - clear  0.000000000, sequence:
0
source 0 - assert 1517363639.431676649,
sequence: 131 - clear  0.000000000, sequence:
0
```

Se añadirá una nueva fila que empieza con ” source 0 – assert …” por cada segundo. Después, instala el

servicio NTP:

```
$ sudo apt install ntp
```

Edita el archivo /etc/ntp.conf para usar GPS/PPS. Haz una copia de seguridad del archivo original y crea un nuevo archivo de configuración usando las siguientes opciones:

```
$ sudo mv /etc/ntp.conf /etc/ntp.conf.bak
$ sudo vi /etc/ntp.conf

# /etc/ntp.conf, configuration for ntpd; see
ntp.conf(5) for help

# Drift file to remember clock rate across
restarts
driftfile /var/lib/ntp/ntp.drift

# Server from generic NMEA GPS Receiver
# server: NMEA serial port (/dev/gps0), mode
16 = 9600 baud + 2 = $GPGGA
# fudge:  flag 1 for use PPS (/dev/gpspps0),
time2 for calibration time offset
server 127.127.20.0 mode 18 minpoll 3 maxpoll
3 prefer
fudge 127.127.20.0 flag1 1 time2 0.000 refid
gPPS
```

Ten en cuenta que el parámetro time2 (0.000) se utiliza para modificar la hora con el fin de calibrar el resultado final. Por último, reinicia el servicio NTP.

```
$ sudo service ntp restart
$ sudo service ntp status

● ntp.service - LSB: Start NTP daemon
   Loaded: loaded (/etc/init.d/ntp; bad;
vendor preset: enabled)
   Active: active (running) since Wed 2018-01-
31 17:44:58 KST; 3s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 744 ExecStop=/etc/init.d/ntp stop
(code=exited, status=0/SUCCESS)
  Process: 754 ExecStart=/etc/init.d/ntp start
(code=exited, status=0/SUCCESS)
   CGroup: /system.slice/ntp.service
           └─765 /usr/sbin/ntpd -p
/var/run/ntpd.pid -g -u 111:115

Jan 31 17:44:58 odroid ntp[754]:    ...done.
Jan 31 17:44:58 odroid systemd[1]: Started
LSB: Start NTP daemon.
Jan 31 17:44:58 odroid ntpd[765]: proto:
precision = 1.375 usec (-19)
Jan 31 17:44:58 odroid ntpd[765]: Listen and
drop on 0 v6wildcard [::]:123
Jan 31 17:44:58 odroid ntpd[765]: Listen and
drop on 1 v4wildcard 0.0.0.0:123
Jan 31 17:44:58 odroid ntpd[765]: Listen
normally on 2 lo 127.0.0.1:123
Jan 31 17:44:58 odroid ntpd[765]: Listen
normally on 3 eth0 192.168.100.28:123
Jan 31 17:44:58 odroid ntpd[765]: Listen
normally on 4 lo [::1]:123
Jan 31 17:44:58 odroid ntpd[765]: Listen
normally on 5 eth0
[fe80::4db2:ce0b:48f3:26af%2]:123
Jan 31 17:44:58 odroid ntpd[765]: Listening on
routing socket on fd #22 for interface updates
```

Espera aproximadamente unos minutos para que el GPS se estabilice, luego comprueba que se está obteniendo una hora precisa desde el GPS/PPS. El resultado solo se activa cuando recibe varias señales de satélite estables. Puede ver que los resultados son como los que aparecen a continuación, verifica que el carácter “o” aparece antes de la numeración IP y que el valor del alcance aumena hasta 377.

```
$ ntpq -p

     remote           refid      st t when
poll reach   delay   offset  jitter
==============================================
==============================
oGPS_NMEA(0)     .gPPS.      0 l    1    8
377    0.000    0.008   0.002

$ ntptime
Check that estimated error is just 1
us(Microsecond).
ntp_gettime() returns code 0 (OK)
  time de1bee1d.49adfb50  Wed, Jan 31 2018
16:26:21.287, (.287811636),
  maximum error 2000 us, estimated error 1 us,
TAI offset 0
ntp_adjtime() returns code 0 (OK)
  modes 0x0 (),
  offset -3.606 us, frequency 1.000 ppm,
interval 1 s,
  maximum error 2000 us, estimated error 1 us,
  status 0x2001 (PLL,NANO),
```

```
  time constant 3, precision 0.001 us,
tolerance 500 ppm,
```

Para ver la publicación Wiki original, visita **https://wiki.odroid.com/odroid-xu4/application_note/gpspps_ntp_server**.

Empezando con Android en el ODROID-C2: Una Guía para principiantes

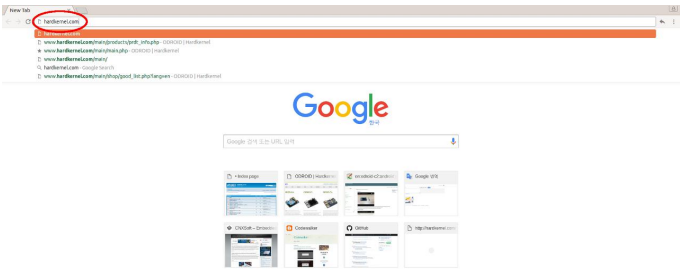
March 1, 2018 By Rob Roy Android, ODROID-C2, Tutoriales

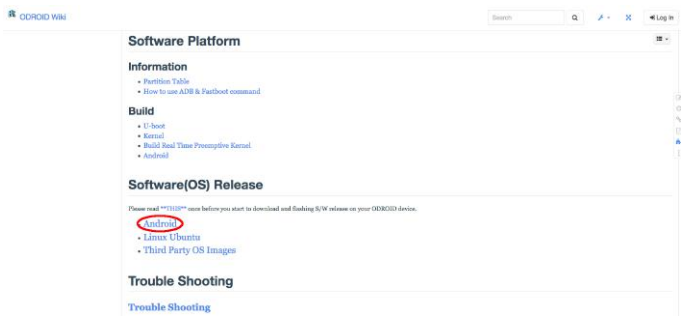
Hay dos opciones para instalar Android en un ODROID-C2. Hardkernel ofrece un módulo eMMC o tarjeta microSD preinstalados, que solo necesitan la instalación de Google Play. Otra posibilidad es descargar el sistema operativo Android desde el sitio web de Hardkernel e instalarlo manualmente en el módulo eMMC o tarjeta microSD. Los materiales necesarios para ejecutar Android en un ODROID-C2 se enumeran a continuación:

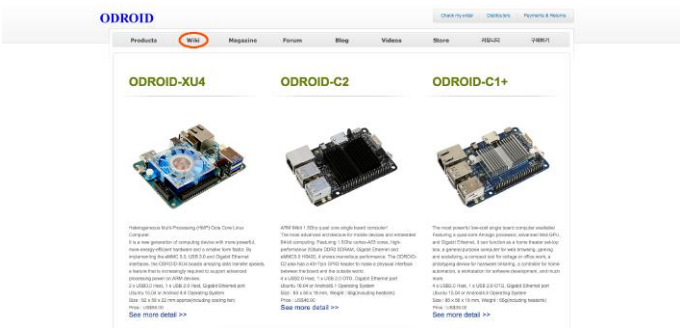
- ODROID-C2 **http://bit.ly/1oTJBya**
- Fuente de alimentación 5V/2A US: **http://bit.ly/2ugY0Xe**, EU: **http://bit.ly/1X0bgdt**, Internacional: **http://bit.ly/OhMyWx**
- Soporte con un sistema operativo preinstalado, eMMC: http://bit.ly/2vq2TCq, tarjeta microSD: **http://bit.ly/2u1fM5I**
- Cable HDMI: **http://bit.ly/2uSu3Ay**
- Monitor o TV con un puerto HDMI

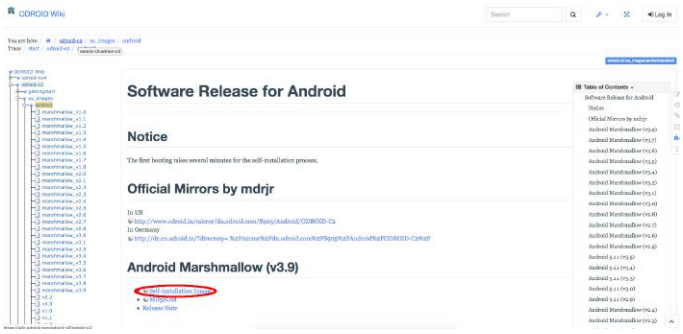
¡Echa un vistazo al video **https://youtu.be/fEyeMTS3idU** Si no tiene una tarjeta de memoria preinstalada con un sistema operativo, sigue las siguientes instrucciones para instalarlo manualmente en la tarjeta de memoria.

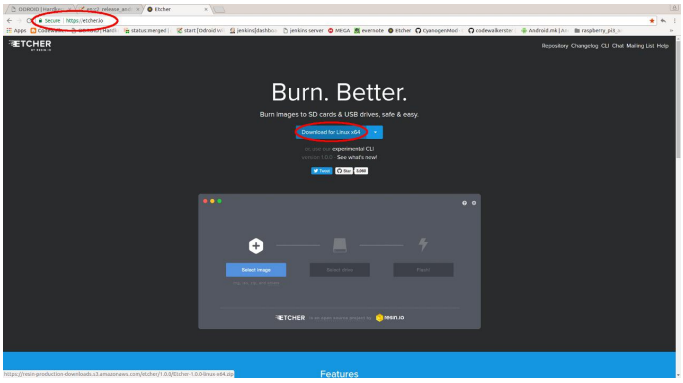
Además de todos los elementos mencionados anteriormente, necesitarás un PC para instalar el sistema operativo Android en la tarjeta de memoria. Tienes un video con instrucciones en **https://youtu.be/9Zi2_OTSl_I** y **https://youtu.be/NyQif1j2WkA**. Ten en cuenta que la fuente de alimentación Smart Power 2 **http://bit.ly/2j3hhcv** ha sido utilizada en el video.

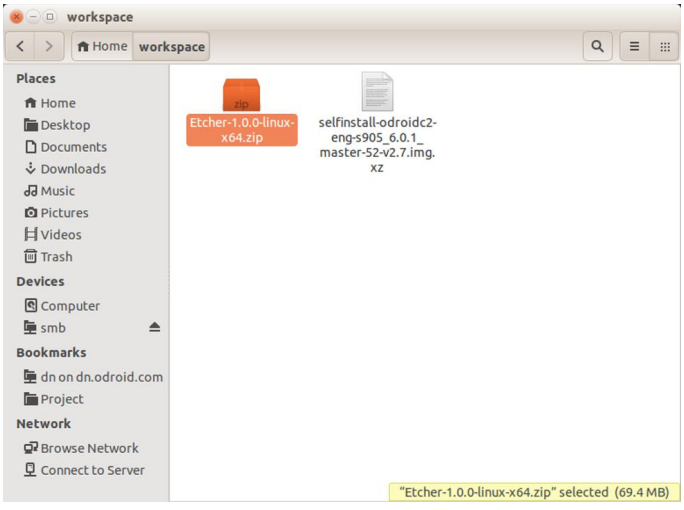




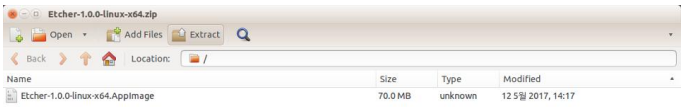


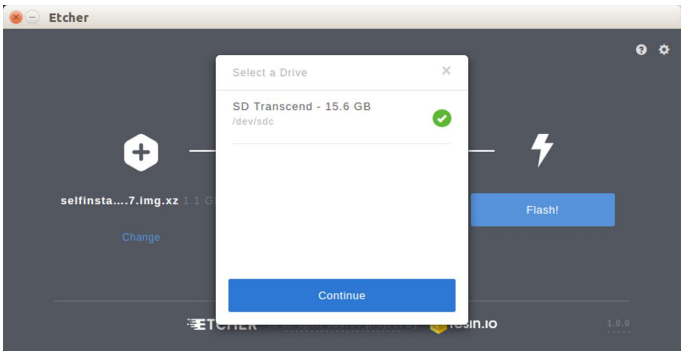


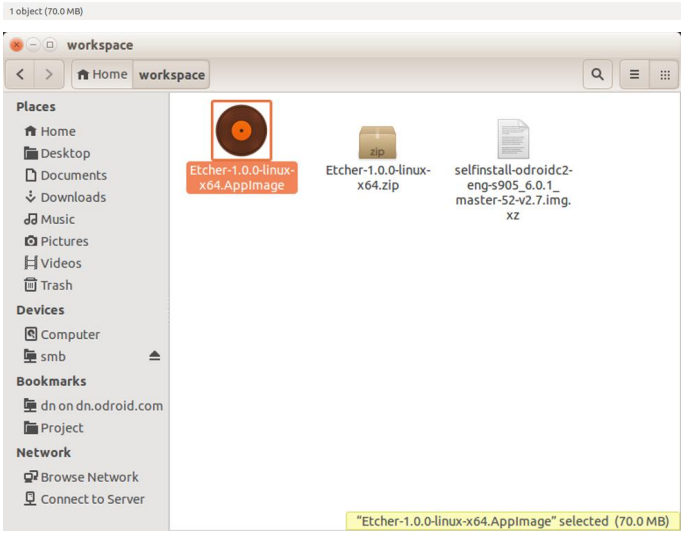


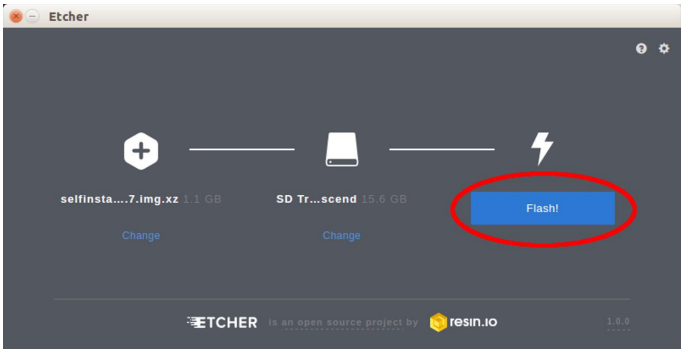


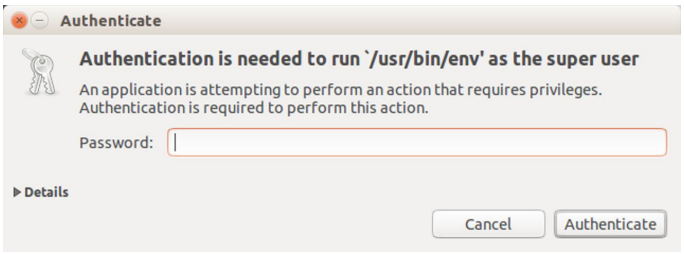
En primer lugar, descarga el sistema operativo Android desde la Wiki de Hardkernel en **http://bit.ly/2tMhk3R**. Asegúrate de que la descarga se complete. Para instalar, o "grabar", Android en la tarjeta de memoria, recomendamos usar Etcher, tal y como se describe en **http://bit.ly/2HAk7iw**. Puedes descargar Etcher desde **https://etcher.io/**. Etcher funciona en Mac OS, Linux y Windows, y es la opción más simple para la mayoría de los usuarios. Etcher también admite la escritura de imágenes de SO directamente desde el archivo zip, sin necesidad de descomprimir. Para instalar el sistema operativo en un módulo eMMC, necesitarás un lector de módulos eMMC (**http://bit.ly/2ugIKK8**) y un multilector USB (**http://bit.ly/2vpTv1y**) para conectarlo a tu PC.



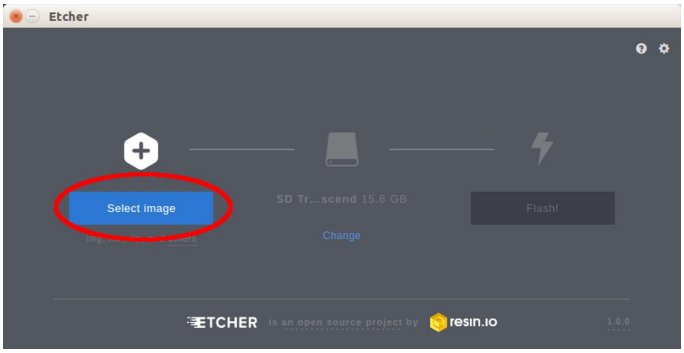


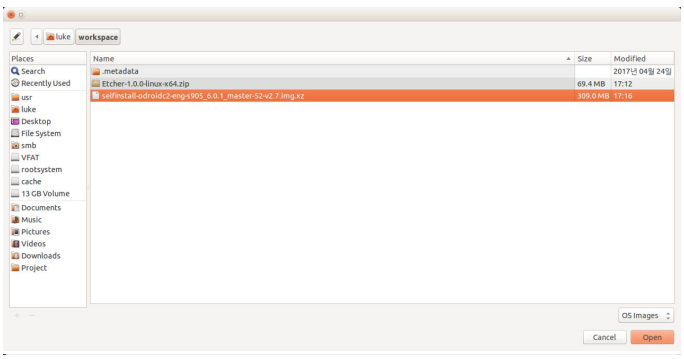


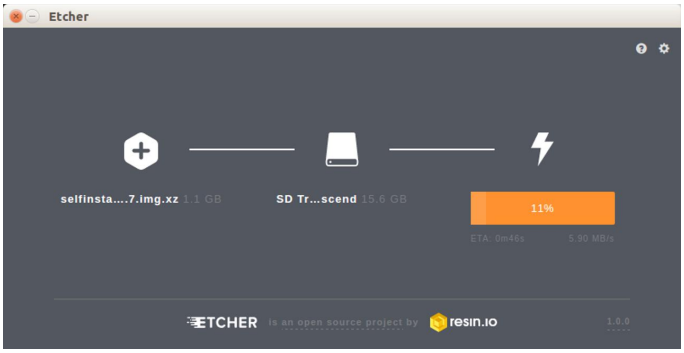


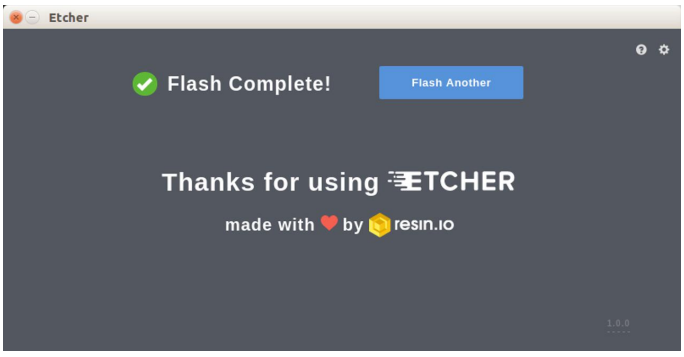


Para instalar Android en un Módulo eMMC, sigue las instrucciones del video https://youtu.be/XfJY4KxLxps. Si utilizas una tarjeta microSD, visualiza el video https://youtu.be/SnrqyoUBry4.









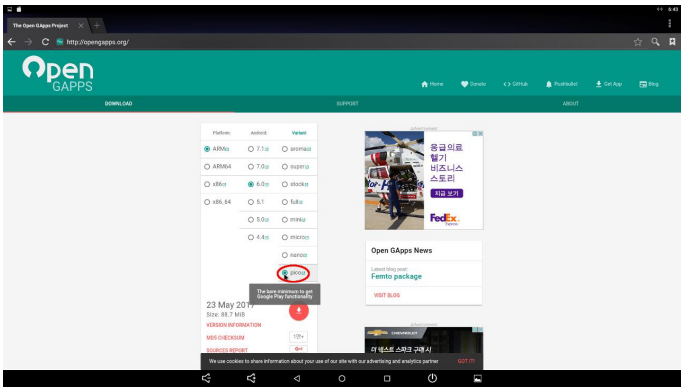
Cuando se complete la instalación del SO en la tarjeta de memoria, conecta el cable HDMI a tu ODROID-C2, luego enchufa la fuente de alimentación. Pasados unos segundos, verás la pantalla de inicio de Android. Para obtener más información, visita el artículo original de la Wiki en http://bit.ly/2uhhlrj.

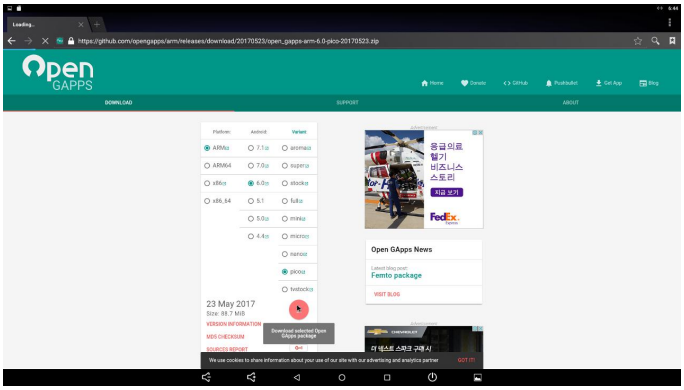
Instalación de Google Play

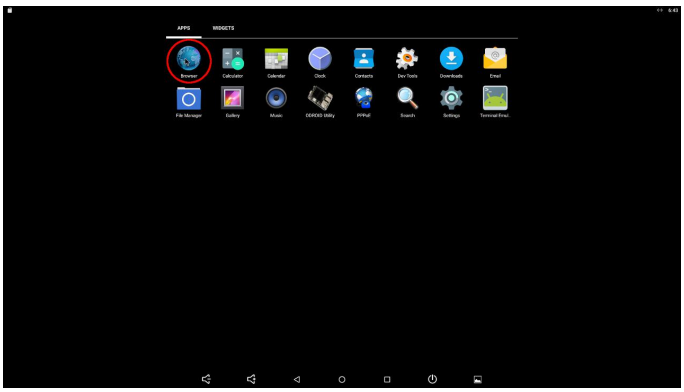
Para instalar Google Play en un ODROID-C2, son necesarios los siguientes elementos:

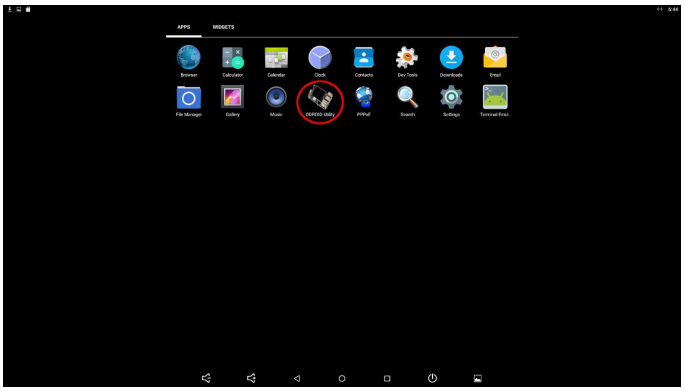
- ODROID-C2 **http://bit.ly/1oTJBya**
- Internet conectado a través del cable Ethernet http://bit.ly/2vg6v9lo por medio de un módulo WiFi **http://bit.ly/22nyxra**
- Si deseas descargar Google Play a un PC y transferirlo al C2, deberá conectar el C2 a la PC mediante un cable OTG **http://bit.ly/2vqf6H5**.

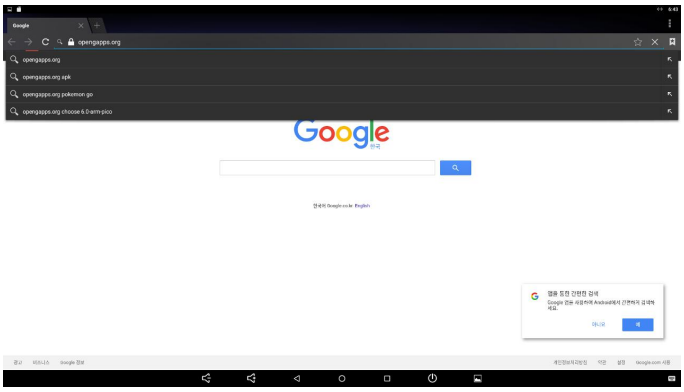
Tienes un video con instrucciones en **https://youtu.be/PKO8ZKJM_0c**. Las siguientes imágenes destacan los principales pasos del video. Abre el navegador en ODROID-C2 y visita **http://opengapps.org**. Recomendamos usar la versión “pico”, aunque el ODROID-C2 también admite versiones micro y nano.

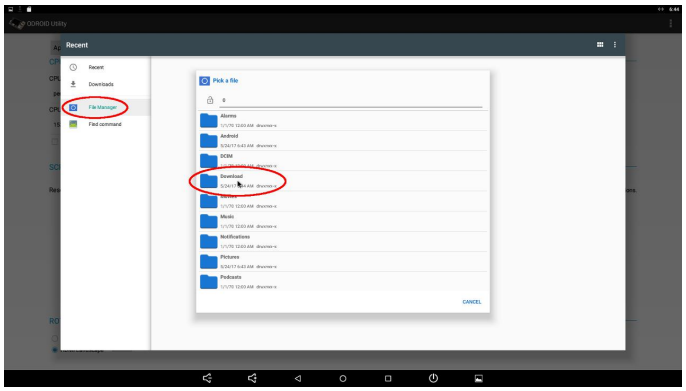


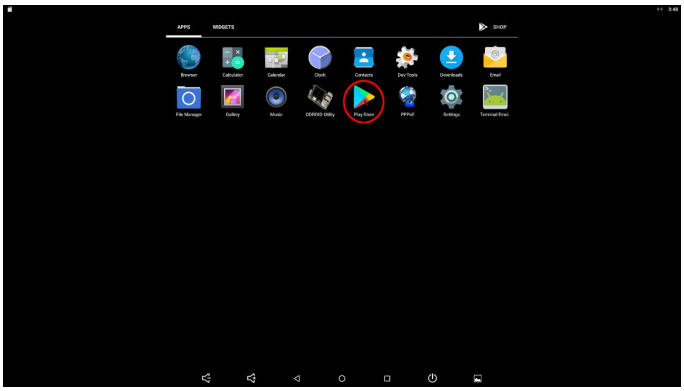


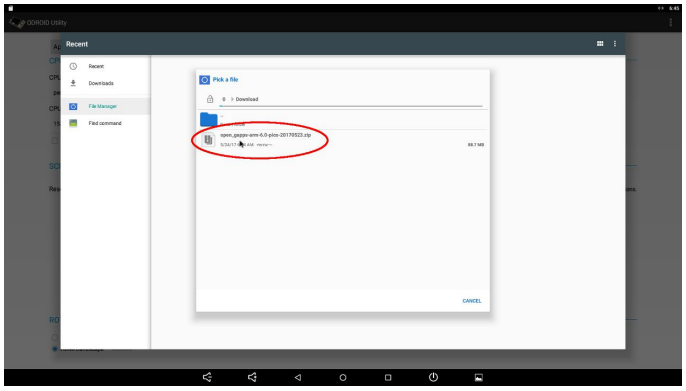


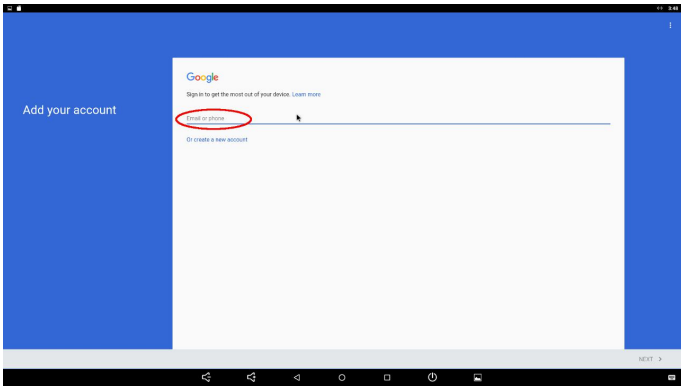


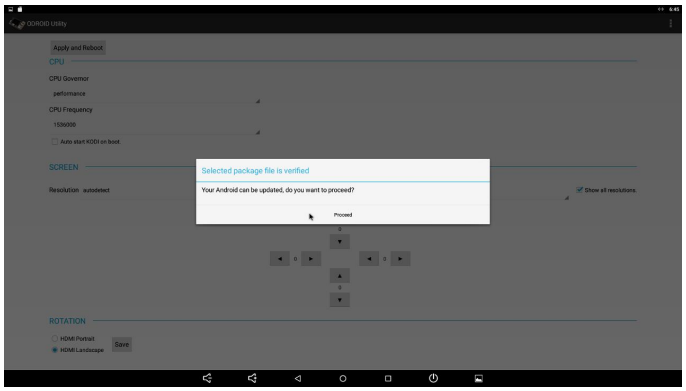


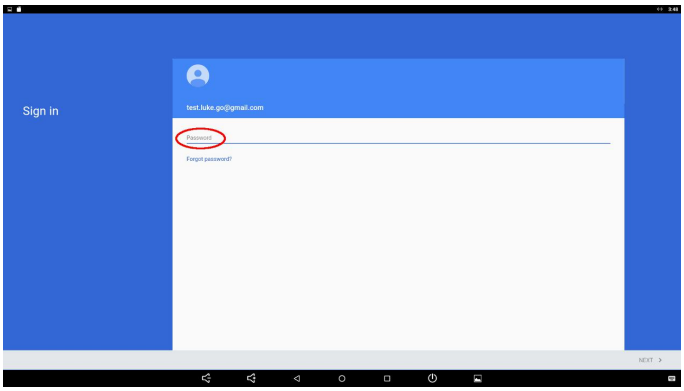


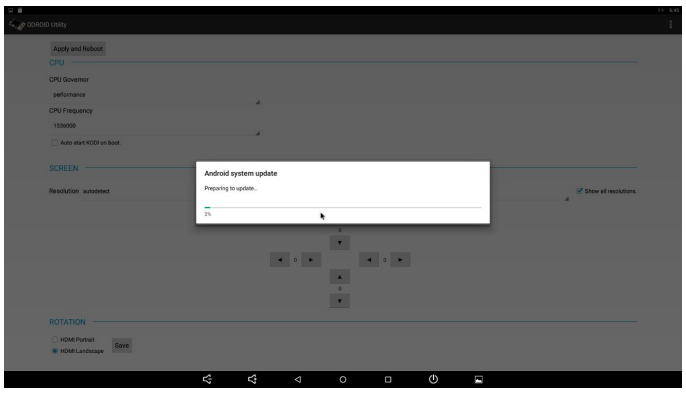


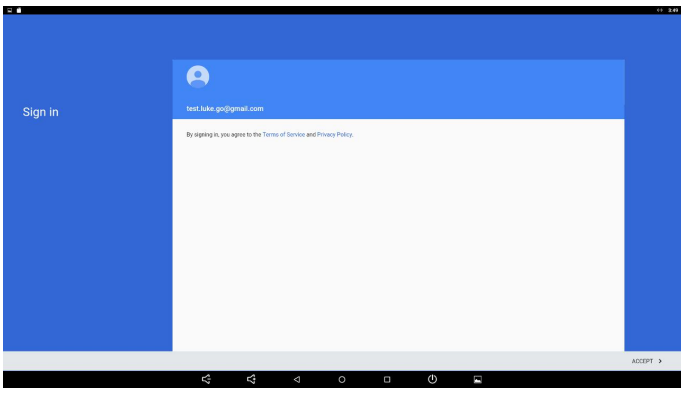




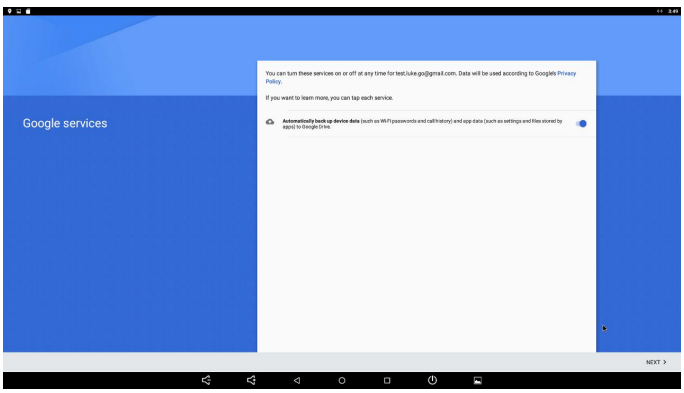


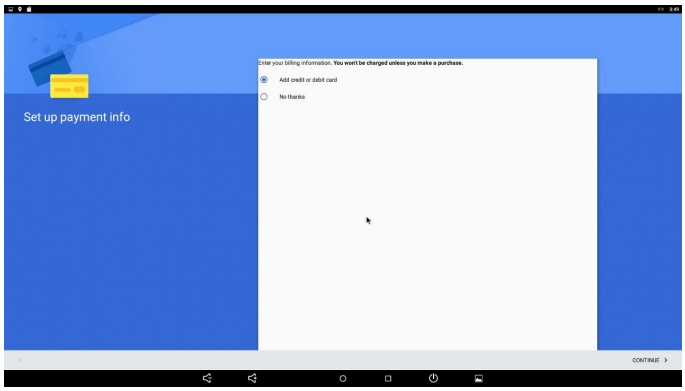






El video en **https://youtu.be/wOhAgkkWnjI** muestra cómo iniciar sesión en tu cuenta de Google y abrir Google Play.

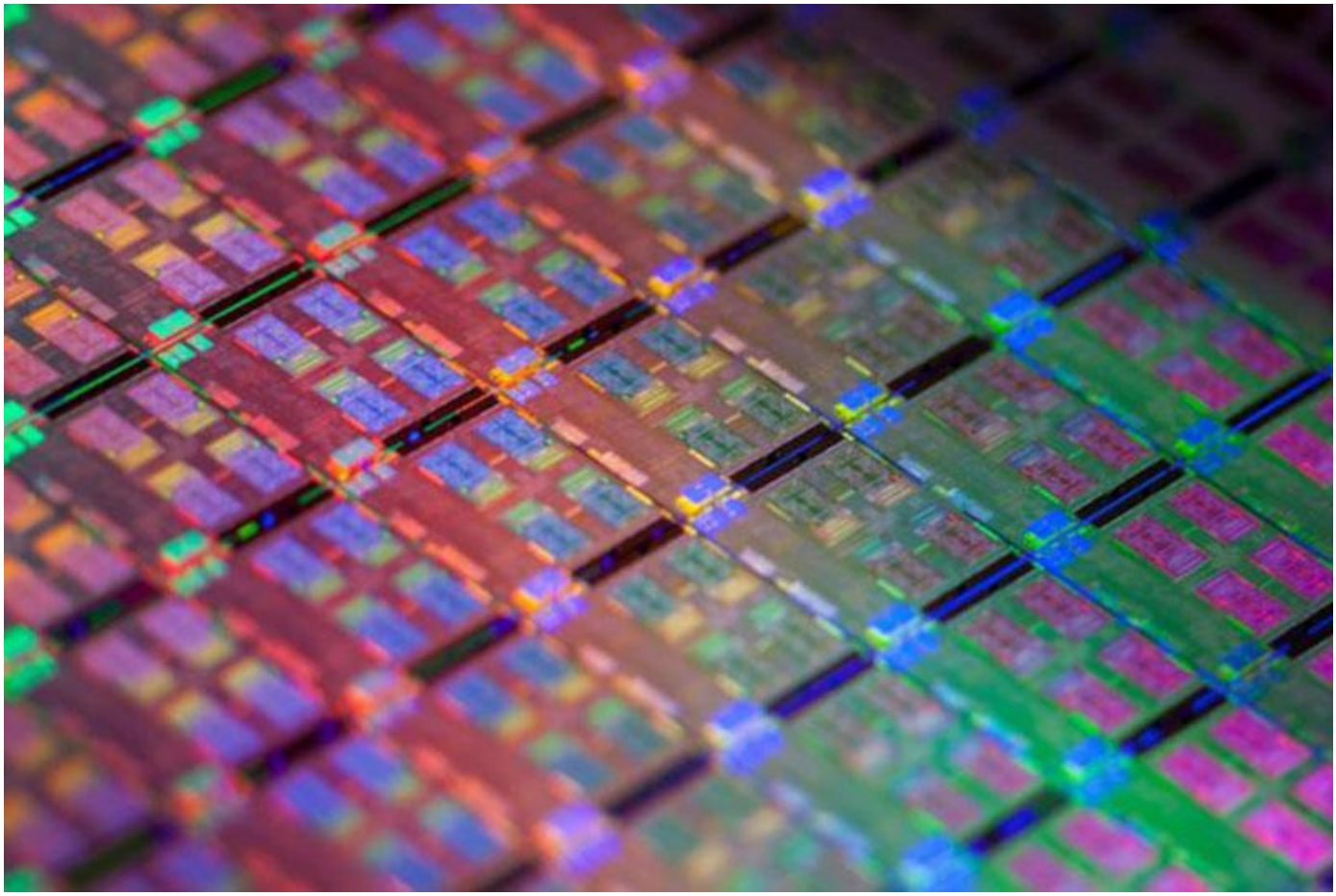




Para obtener más información, visita el artículo original de la Wiki en **http://bit.ly/2vqgz0c**.

Cómo Activar la Decodificación por Hardware en el ODROID-C2

March 1, 2018 By @pichljan Linux, ODROID-C2, Tutoriales



El usuario @pichljan ha creado un repositorio git con un script, parches e instrucciones. Este repositorio tiene correcciones destinadas a ayudar al usuario a activar la decodificación de hardware para el ODROID-C2. Si alguien está haciendo frente a este problema, puede clonar este repositorio y seguir los siguientes pasos. Estos pasos también están descritos en el archivo README del repositorio. En primer lugar, debe clonar el repositorio Linux de Hardkernel:

```
$ git clone --depth 1
https://github.com/hardkernel/linux.git -b
odroidc2-3.14.y
$ cd linux
```

Aplica un parche que te permita compilar el driver de video AML como módulo. Di este paso editando media_build de LibreELEC:

```
$ patch -p1 < ../odroidC2-
kernel/allow_amlvideodri_as_module.patch
```

Aplica la configuración por defecto de ODROID-C2, luego modifica los parámetros de configuración:

```
$ make odroidc2_defconfig
$ make menuconfig
```

Fija los siguientes valores (presiona Y para seleccionar, N para eliminar y M para seleccionarlo como módulo):

```
Device Drivers
   Amlogic Device Drivers
      ION Support
         ION memory management support = Yes
      Amlogic ion video support
         videobuf2-ion video device support =
M
         Amlogic ion video device support = no
      V4L2 Video Support
         Amlogic v4l video device support = M
         Amlogic v4l video2 device support =
no
      Amlogic Camera Support
```

```
        Amlogic Platform Capture Driver = no
   Multimedia support = M
```

A continuación, tenemos que compilar el kernel:

```
$ make -j5 LOCALVERSION=""
```

El parámetro LOCALVERSION es solo para evitar el signo “+” en el nombre del kernel. Tras una compilación exitosa, instala los módulos y el kernel, luego reinicie el sistema:

```
$ sudo make modules_install
$ sudo cp -f arch/arm64/boot/Image
arch/arm64/boot/dts/meson64_odroidc2.dtb
/media/boot/
$ sudo sync
$ sudo reboot
```

Compilar Media

Clona el repositorio media_build e intenta compilarlo:

```
$ git clone
https://git.linuxtv.org/media_build.git
$ cd media_build
$ ./build
```

El comando de compilación probablemente fallará, aunque puedes ignorar este error y continuar con los siguientes pasos. El siguiente script también está inspirado en LibreELEC media_build edition y solo incluye el controlador de video en el módulo media.

```
$ ../odroidC2-
kernel/add_video_driver_module.sh
```

Para evitar posibles problemas con la compilación, intenta deshabilitar la compatibilidad con el controlador remoto y todos los adaptadores USB que no necesites:

```
$ make menuconfig
```

Este comando probablemente tendrá como resultado un error similar al siguiente:

```
./Kconfig:694: syntax error
./Kconfig:693: unknown option "Enable"
./Kconfig:694: unknown option "which"
```

Debes editar el archivo v4l/Kconfig y alinear con espacios las líneas impresas en el error. Las líneas deben estar alineadas con las anteriores. A continuación, ejecuta make menuconfig nuevamente, lo cual puede que necesites hacer varias veces. Si ve un menú en lugar del error, puede modificar la configuración del siguiente modo:

```
Remote Controller support = no
Multimedia support
    Media USB Adapters
        ## Disable all driver you don't need
##
```

Aplica el siguiente parche:

```
$ patch -p1 < ../odroidC2-kernel/warning.patch
```

Realiza el siguiente cambio para evitar errores y compilar el kernel:

```
$ sed -i 's/#define NEED_PM_RUNTIME_GET
1///#define NEED_PM_RUNTIME_GET 1/g'
v4l/config-compat.h
$ make -j5
```

Es posible que necesites ejecutar los pasos anteriores (sed y make) varias veces antes de que tenga éxito. Tras la compilación, instala los módulos y reinicia el sistema:

```
$ sudo make install
$ sudo reboot
```

El último paso es añadir el módulo amlvideodri a /etc/modules para que se cargue en el arranque:

```
$ sudo echo "amlvideodri" >> /etc/modules
```

Ahora puedes disfrutar de tus videos acelerados por HW y DVB-T TV en Kodi. Para obtener más información o más ayuda sobre este tema, consulte el hilo original en los foros de ODROID en **https://forum.odroid.com/viewtopic.php?f=136&t=29619#p215565**.

Ordenador de Control ODROID-XU4: Creando un Sistema de Control Todo en Uno

March 1, 2018 By @williamg42 Linux, ODROID-XU4, Tutoriales

Este proyecto empezó en la primavera de 2017, y al fin siento que he avanzado lo suficiente como para publicar lo que he estado haciendo. Todo comenzó cuando estaba recibiendo un curso de Robótica Bayesiana, y pensé que sería interesante aplicar lo que había aprendido. El único problema era que no había un sistema Linux embebido que tuviera la suficiente potencia informática como para ejecutar grandes filtros de partículas a un coste razonable, y que también tuviese los sensores necesarios (GPS, IMU) de una calidad razonable integrados, así que decidí crear uno.

Especificaciones del diseño

- La placa alojará múltiples IMU MEMS en diferentes buses para la redundancia y para permitir la implementación de un filtro Bayesiano de múltiples sensores
- La placa alojará un único receptor GNSS para permitir una precisión de localización de + -2.5m al aire libre. Se eligió GNSS para acceder a los sistemas GPS US y GLONASS Russian, y por el arranque en frío más rápido.
- La placa soportará una interfaz análoga capaz de medir tensiones de hasta 20V para monitor con voltaje de batería
- La placa admitirá una interfaz analógica capaz de escalar una señal de -5V a 5V de 0 a 1.8V
- La placa albergará un módulo XBee Pro
- La placa soportará salidas PWM directas para el control de dispositivos externos
- La placa no debería proporcionar alimentación para estos dispositivos externos

Selección de Componentes

El BNO055 fue seleccionado para dos de las IMU, principalmente debido a su uso en controladores

Pixhawk. LSM9DS1 fue seleccionado como el tercer sensor de redundancia, con una dirección I2C diferente y porque parecía interesante.

La versión 1 es el PCB actualmente acabada en la foto de arriba. La versión 4 es la próxima versión de la placa que actualmente está en fase de estudio.

El BNO055 deja mucho que desear. El ruido eléctrico del resto del sistema causa ruido en el magnetómetro, de modo que en su lugar se usará el BNO080. Es aproximadamente tres veces más preciso gracias a un mejor algoritmo de fusión integrado. Además, proporciona una estimación de cómo de precisos son los datos proporcionados, que es algo muy importante para el filtro GPS/IMU en el que estoy trabajando. También es compatible con sensores de presión barométrica externos.

- 5.2mm x 3.8mm x 1.1mm
- Up to 1KHz
- 2.0msec
- 3.0° – Dinámico 1.0° – Estatico
- 0.5°/min
- ± 2000°/sec

El BHI160 también se usará como segundo IMU. Los sensores son similares en precisión con el BNO080, pero la fusión del sensor resultante no es tan buena. Sin embargo, esta IMU admite un sensor de magnetómetro I2C externo, para lo cual diseñaré una placa soporte y un control remoto lejos de fuentes de ruido eléctrico. Esto me permitirá determinar con precisión el norte magnético. El BMM150 es el magnetómetro externo, admitido como entrada directa en el algoritmo de fusión dentro del BHI160. En realidad, es tan bueno como lo son los sensores, aunque es un BGA, será divertido de reconducir.

El voltaje E/S del XU4 es de 1.8V y por lo tanto, es necesario realizar una conversión de nivel lógico. El TXB0108-PW fue seleccionado por recomendación del OEM. Se eligió un A5100-A de Maestro Wireless Solutions, ya que es un receptor con capacidad GNSS, con soporte de antena activa. Es un módulo todo en uno con mínimos componentes externos.

Esquemas

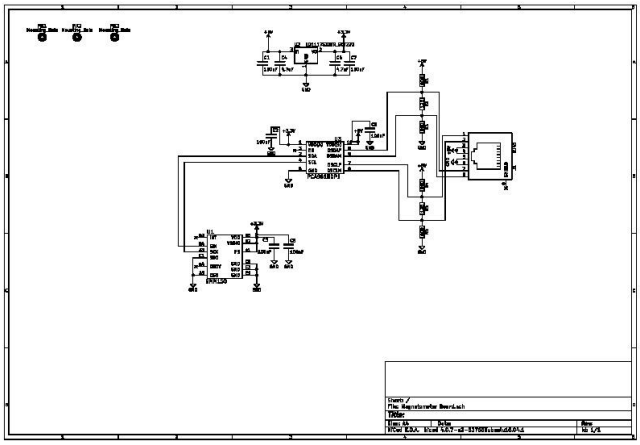


Figura 1 – Placa soporte del magnetómetro

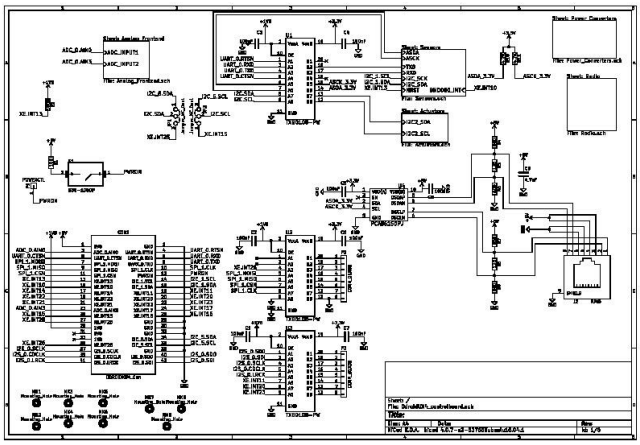


Figura 2 – Esquema del nivel superior de la placa del sensor, con conexiones a ODROID-XU4, conversión lógica, botón de encendido-apagado e I2C para la conversión I2C diferencial y conector RJ45

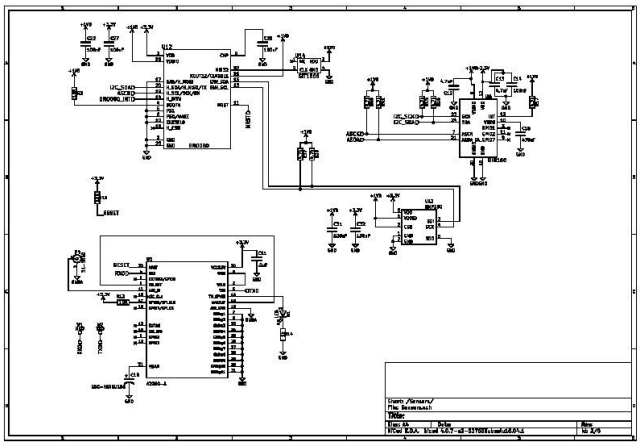


Figura 3 – Sensores

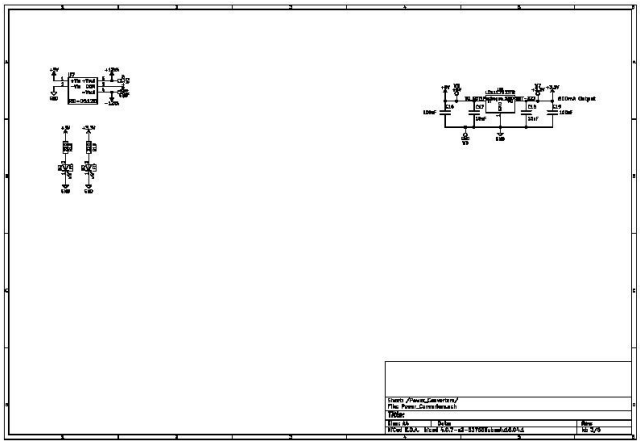


Figura 4 – Alimentación

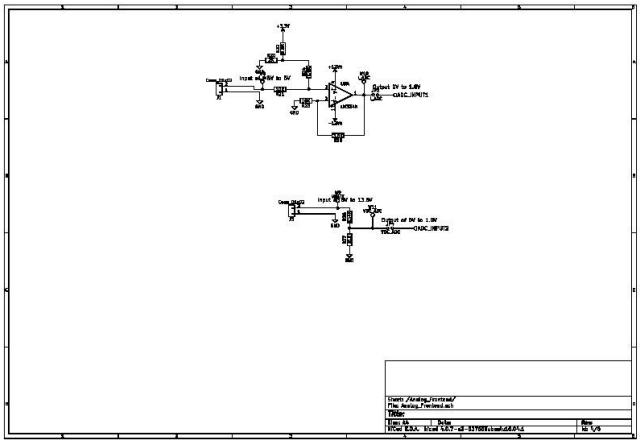


Figura 5 – Interfaz analógica, convierte una señal de 8V a 13.8V y de -5V a 5V a 0V a 1.8V para alimentar el ODROID-XU4

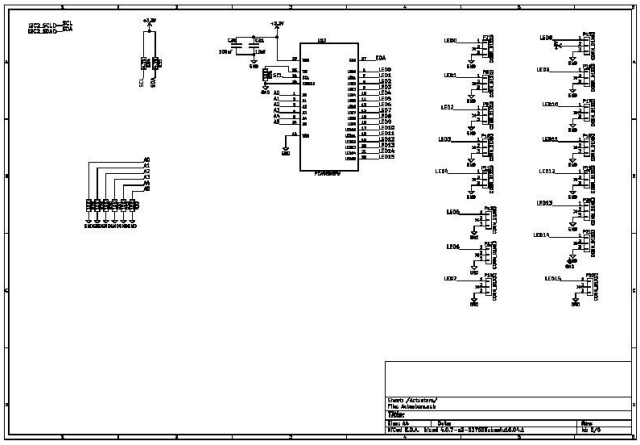


Figura 6 – Controlador LED controlado por I2C que emite señales PWM programables en 16 canales

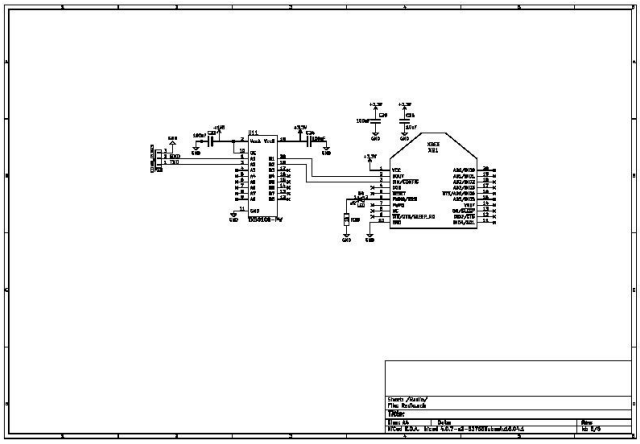


Figura 7 – Enlace de comunicación en serie del Xbee

Conociendo un ODROIDian: Go Sang “Luke” Chul (Luke.go)

March 1, 2018 By Rob Roy Conociendo un ODROIDian

Por favor, háblanos un poco sobre ti. Tengo 31 años, ha nacido y vivo en Seúl, Corea del Sur. Tengo una diplomatura en Informática y una licenciatura en Ingeniería de Software Embebido de la Universidad Kookmin en Corea del Sur. He estudiado la virtualización Embebida y he creado un hipervisor que funciona en los sistemas ARMv8. Actualmente soy ingeniero de software en Hardkernel Co., Ltd. Mantengo la versión Android para todos los dispositivos ODROID a excepción de LineageOS para ODROID-XU4. Principalmente actualizo las revisiones, agrego funciones y corrijo errores en las versiones oficiales de Android de Hardkernel.

Figura 1 – Luke y su familia en Jungfrau

Mi hermana menor y su marido son escritores de webtoon (webcomic coreano). Ellos publican el webtoon cada semana. También estoy muy orgulloso de haber participado en vigilias con velas cada semana desde 2016-2017.

¿Cómo empezaste con los ordenadores? Cuando tenía 6 años, me topé con mi primer ordenador. Cuando visitaba a mi tía, mi primo tenía algunos ordenares 386. Como muchos otros, El ordenador era una

consola de juegos para mí. Jugaba a Sango Fighter, Prince Of Persia, Prehistorik, Jazz Jackrabbit y a mucho otros. Empecé a estudiar seriamente los sistemas informáticos avanzados tras realizar el servicio militar, porque quería hacer mi propio sistema operativo. Estudié muchos aspectos de los ordenadores, aunque mi materia favorita era el software integrado. Quería hacer una obra maestra de un producto como es un sistema integrado al completo.

¿Con qué tipo de proyectos estas trabajando en Hardkernel? Uno de mis proyectos es crear la función de acceso directo en las Apps Utility, que conecta algunas aplicaciones con teclas de función para iniciar la aplicación. Incluso puedo conectarlas a botones físicos a través de los pines GPIO. También he renovado el diseño de la Wiki. Quería que a los usuarios les resultase más fácil acceder a la página, así que apliqué una estructura de árbol y vistas de texto con fondo del color de las placas para distinguirlas. Sé que no es suficiente, pero espero que resulto más fácil usar la Wiki de ODROID.

¿Cómo usas tus ODROID personales? Cuando estaba estudiando en el laboratorio, intenté montar un sistema hipervisor para trabajar en el ODROID-XU, pero no pude hacerlo por diversos problemas. Recientemente, utilicé un ODROID-C2 como reproductor de video y emulador. También tengo pensado usarlo como un controlador automotriz para el hogar tomando como referencia algunos artículos de la revista.

¿Cuál es tu ODROID favorito y por qué? El ODROID-C2 es mi favorito. Gracias a su tamaño, se puede colocar en cualquier lugar, y me gusta que permita reproducir videos a una resolución de 4K.

Figura 3 – Bungee haciendo puenting en Nueva Zelanda

¿Qué innovaciones le gustaría ver en futuros productos Hardkernel? Me gustaría añadir versatilidad y escalabilidad a los nuevos productos de Hardkernel para que los ODROID se puedan usar en varios campos. Si el producto tiene un buen rendimiento, es aún mejor, pero me gustaría seguir con lo básico. También me gustaría ver más placas adicionales como Hi-Fi Shield.

¿Qué hobbies e intereses tienes aparte de los ordenadores? Me gusta viajar a otros países y hacer cosas atrevidas. Había estado haciendo paracaidismo y puenting en Queenstown, Nueva Zelanda, lo cual fue increíble. Realmente lo recomiendo, especialmente el paracaidismo, que fue tremendo. También he visitado Uluru (Ayers Rock) en Australia, lo cual fue espectacular. A finales de 2017, me subí a un Mario Kart en Tokio. Espero volver a hacerlo este año, me encantó la experiencia.

Figura 4 – Visitando Uluru en Australia

Figura 5 – Conduciendo Mario Kart en la vida real en Tokio

Hace poco he empezado a tocar la guitarra. Con este instrumento soy un auténtico novato, como un ingeniero de software que acaba de empezar a imprimir “Hello World” en un nuevo lenguaje. Me he enseñado a mí mismo memorizando algunos acordes de guitarra, espero tocar bien pronto.

¿Qué consejo le darías a alguien que quiere aprender más sobre programación? Recomiendo tener objetivos claros. Hay tanta información sobre programación en Internet, que antes de aprender a programar, debes definir tus objetivos y determinar qué es lo que necesitas para lograrlos. Este checklist puede no estar en Internet. Este proceso te ayudará a lograrlos con más facilidad. Si quieres ser más profesional, aprende los conceptos básicos. Tener fluidez en los lenguajes es importante, pero los principios básicos lo son más.