UART Daisy Chain • Juegos Linux • GPIO Android • Ambilight • Android TV

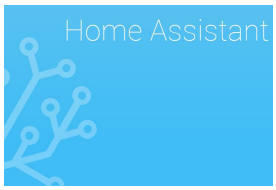
ODROID Magazine

Año Cinco
Núm.#49
Ene 2018

Vive como un rey:
DISEÑANDO UN CUADRO DE MANDOS PARA ODROID HOME

RECOMPILAR IMAGENES DOCKER X86/AMD64 PARA UN SWARM ARM

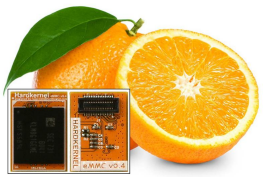
EMMC NARANJA PARA C1/C2/XU4



Home Assistant: Diseñando un Elegante Cuadro de Mandos

January 1, 2018

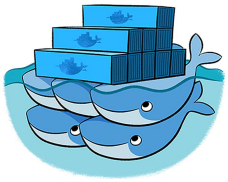
En este artículo vamos a analizar un compañero de Home Assistant – AppDaemon



Módulo eMMC Naranja: Llega el chipset Samsung 5.1

January 1, 2018

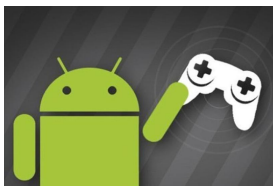
El módulo eMMC Naranja de Hardkernel que usa el chipset Samsung eMMC 5.1, ha estado enviándose desde Octubre de 2017



Recompilando imágenes Docker x86amd64 para Swarm ARM

January 1, 2018

Recompilar imágenes Docker x86/amd64 para un Swarm ARM continua los recientes artículos sobre cómo compilar swarm Docker en ARM, y como no había disponible ninguna imagen ARM o no había disponible ninguna imagen ARM con una versión reciente, sin lugar a duda era hora de cambiar esto.



Juegos Android: Monument Valley, Hopscotch, Aqueducts

January 1, 2018

No siempre nos desviemos de los extraños juegos indie para Android, pero durante las vacaciones play store nos ha dotado con grandes títulos, así que sin más preámbulos permite presentarte:

Android TV ODROID-C2 con Amazon Prime Video y Netflix

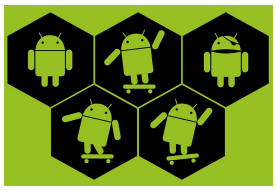
January 1, 2018

He estado usando un ODROID-C2 con LibreELEC durante bastante tiempo, pero me sentía muy decepcionado por la falta de compatibilidad con Amazon Prime Video y Netflix. También he estado usando un teclado/ratón inalámbrico para controlarlo, lo cual me permitía censurar a su compañero, ya que deseaba tener único un mando ▶

Ambilight en ODROID-C2 Usando LibreElec: Adaptando Ambilight al ODROID-C2

January 1, 2018

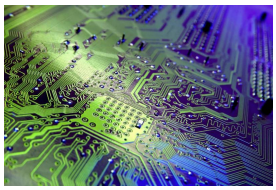
Conseguí montar un sistema Ambilight funcional usando un ODROID-C2 y LibreElec.



Divirtiéndonos con GPIO en Android

January 1, 2018

El ODROID-C1/C1+, ODROID-C2 y ODROID-XU4 tienen pines GPIO (Entrada/Salida de Propósito General) que permiten controlar dispositivos externos a través de software. Para acceder correctamente al puerto GPIO, debes instalar la imagen Android Marshmallow versión 2.8 o superior en el ODROID-C2, la imagen Android KitKat versión 3.2 o superior en el ▶



UART Daisy Chain: Depuración Avanzada con el ODROID-C2

January 1, 2018

Este artículo explica como usar múltiples puertos UART en ODROID-C2 ejecutando el Sistema Operativo Android

Juegos Linux: Mech Warrior 2

January 1, 2018

Mech Warrior es un juego de simulación de combate sobre los llamados “Mechs”, que son robots gigantes pilotados por un humano.

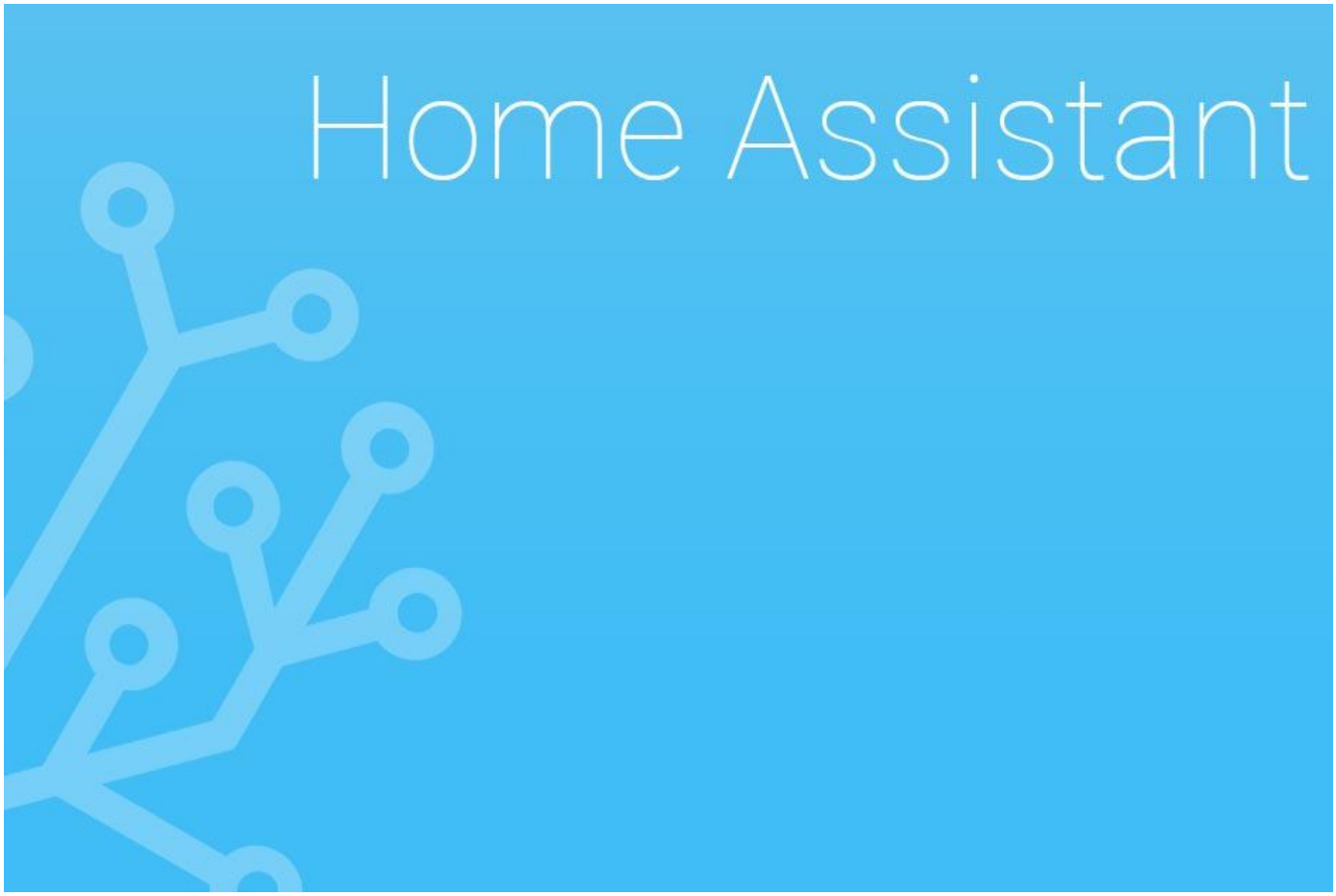
Conociendo un ODROIDAN: Dongjin Kim

January 1, 2018

Por favor, háblanos un poco sobre ti. Soy un ingeniero de software embebido, he participado en muchos y diferentes proyectos comerciales desde 1994. Actualmente, estoy desarrollando el software para un dispositivo móvil que se ejecuta sobre un procesador ARM y trabaja principalmente con un driver de dispositivo o capa HAL/framework. ▶️

Home Assistant: Diseñando un Elegante Cuadro de Mandos

January 1, 2018 By Adrian Popa Linux, Mecaniqueo



En este artículo vamos a analizar un compañero de Home Assistant – AppDaemon (**https://goo.gl/UD3hDA**). Se trata de un entorno de trabajo que te permite desarrollar tus propias aplicaciones Python y hacer que éstas reaccionen a los eventos e interactúen directamente con Home Assistant. Te proporciona la flexibilidad de poder escribir complejas automatizaciones directamente en Python, y si cuentas con ciertos conocimientos de programación, te puedo decir que bastante más fácil que escribir largas automatizaciones en YAML. Tiene la ventaja añadida de contar con un entorno de trabajo que te permite desarrollar cuadros de mandos muy atractivos.

AppDaemon se puede instalar en el mismo sistema que Home Assistant, también conocido como HA, o en un sistema diferente, ya que se comunica con HA a través de un socket de red. La instalación es sencilla, puede encontrar instrucciones para ello en **https://goo.gl/Hci4zm**.

```
$ sudo pip3 install appdaemon
```

En el futuro, podrás actualizar fácilmente appdaemon con:

```
$ sudo pip3 install --upgrade appdaemon
```

También deberías crear este servicio systemd para gestionar el inicio automático:

```
$ cat /etc/systemd/system/appdaemon.service
[Unit]
Description=AppDaemon
After=homeassistant.service
[Service]
Type=simple
User=homeassistant
ExecStart=/usr/local/bin/appdaemon -c
/home/homeassistant/.homeassistant
[Install]
WantedBy=multi-user.target
```

```
$ sudo systemctl enable appdaemon
$ sudo systemctl start appdaemon
```

Necesitarás crear una configuración por defecto preferiblemente dentro del directorio de configuración de HA y añadir aplicación para probar tu configuración. El archivo de configuración appdaemon.yaml también almacena las credenciales para acceder a tu Home Assistant, o puede leerlas desde el archivo "secrets.yaml".

```
$ sudo su - homeassistant
$ cat
/home/homeassistant/.homeassistant/appdaemon.y
aml
 AppDaemon:
 logfile: STDOUT
 errorfile: STDERR
 logsize: 100000
 log_generations: 3
 threads: 10
 HASS:
 ha_url: http://127.0.0.1:8123
 ha_key: !secret api_password
$ mkdir
/home/homeassistant/.homeassistant/apps
$ cat
/home/homeassistant/.homeassistant/apps/hello.
py
import appdaemon.appapi as appapi

#
# Hello World App
#
# Args:
#

class HelloWorld(appapi.AppDaemon):

def initialize(self):
 self.log("Hello from AppDaemon")
 self.log("You are now ready to run Apps!")
```

El código hello.py anterior ha sido sacado de las instrucciones de instalación, aunque también puede encontrar algunas aplicaciones útiles en el repositorio **https://goo.gl/6nkzhm**. Para activar y configurar una aplicación, deberás añadir lo siguiente dentro de 'apps.yaml':

```
$ cat
/home/homeassistant/.homeassistant/apps.yaml
 hello_world:
 module: hello
 class: HelloWorld
```

Una vez que reinicies AppDaemon, las aplicaciones se cargarán automáticamente. En este caso, deberías ver el mensaje "Hello from AppDaemon" en tus registros log, lo cual indica que la configuración inicial se ha cargado, puedes verificarlo con:

```
$ sudo journalctl -f -u appdaemon
```

La mejor manera de empezar es leyendo la documentación. Hay un completo tutorial que te guía por todos los pasos: **https://goo.gl/ha5iC8**. Además, existe una referencia API para la búsqueda rápida: **https://goo.gl/QeJSYu**. El entorno de trabajo está basado en eventos, de modo que necesitas configurar "oyentes" para los varios eventos que tienen lugar en Home Assistant, de esta manera tu código será llamado automáticamente. Además, puedes acceder a todos los estados y atributos de las entidades de Home Assistant. Cuando estés familiarizado con el entorno de trabajo, puedes echar un vistazo a las aplicaciones de ejemplo para tener una idea de cómo se hacen las cosas.

¿Te acuerdas del proyecto del calentador con Home Assistant, publicado en el último número de ODROID Magazine? Pues bien, quizás porque me estoy haciendo viejo, siento la necesidad de encender la calefacción durante ciertas horas del día y de la noche, así que quería crear una aplicación que lo hiciera por mí. No obstante, hay un problema: quiero disponer de algún tipo de panel de control que sea fácil de usar dentro de Home Assistant y que me permita seleccionar el momento en el que quisiera que se encendiera la calefacción, como si fuera algún tipo de sustituto a la típica tarea cron. Considere apropiados los intervalos de 15 minutos por interruptor, de modo que si quería encender el calentador de 4:00 a 4:30, tendría que pulsar dos interruptores en la interfaz de usuario (4:00 y 4:15).

Haciendo un cálculo rápido podemos concluir que un día tiene 96 intervalos de 15 minutos, y puesto que

soy algo flojo y no quiero escribir todo este código de configuración, hice un script para que generase la configuración (**https://goo.gl/DYY5Mj**). Si ejecutas el script, se crearán 96 interruptores input_boolean (**https://goo.gl/BtJZ41**) que serán distribuidos en grupos de 4 horas. Puedes copiarlo/pegarlo en tu archivo de configuración bajo las secciones correspondientes y reiniciar Home Assistant. No olvides añadir el 'heater_timer_group' en la vista 'heater':

```
$ wget https://raw.githubusercontent.com/mad-ady/home-assistant-customizations/master/configuration_helper/make_heater_switches.py
$ python make_heater_switches.py
```

Configuration.yaml debería parecerse a esto:

```
input_boolean:
 …
 heater_timer_00_00:
 name: Heater timer 00:00
 initial: off
 icon: mdi:fire
 heater_timer_00_15:
 name: Heater timer 00:15
 initial: off
 icon: mdi:fire
 …

group:
 heater_timer_group_0:
 name: Timer group 00:00 - 04:00
 control: hidden
 entities:
 - input_boolean.heater_timer_00_00
 - input_boolean.heater_timer_00_15
 - input_boolean.heater_timer_00_30
 …
 heater_timer_group:
 name: Heater timer
 control: hidden
 entities:
 - group.heater_timer_group_0
 - group.heater_timer_group_1
 - group.heater_timer_group_2
 - group.heater_timer_group_3
 - group.heater_timer_group_4
 - group.heater_timer_group_5
```

```
…
 heater:
 name: Gas heater
 view: yes
 icon: mdi:fire
 entities:
 - switch.heater
 - climate.heater_thermostat
 - group.heater_timer_group
```

Cuando termines deberías ver una pantalla similar a la que se muestra en la Figura 1.

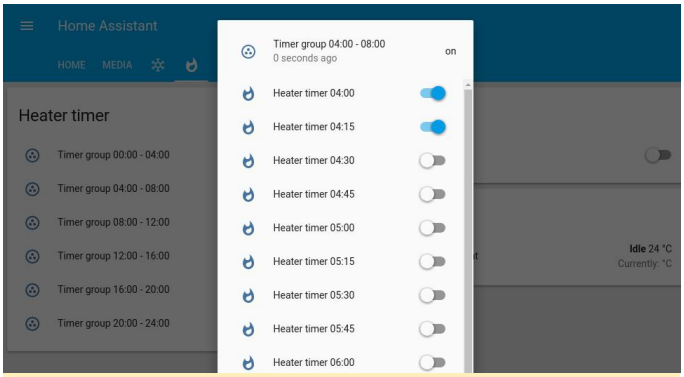


Figura 1 – Montones de interruptores de tiempo

Para que estos interruptores funcionen, hay que hacer una aplicación que escuche sus cambios de estado y de este modo se llevarán a cabo las acciones deseadas. Escribí un código para AppDaemon que hace lo siguiente:

- Al arrancar, empieza a escuchar eventos de entidades input_boolean llamadas heater_timer_ *, que se repiten a lo largo de todas las entidades y registra un "oyente" para cada una.
- Cuando detecta que se ha activado un booleano, comprueba si el nombre del booleano es el mismo que el momento actual y si es así, controla la calefacción
- Cada minuto se comprueba el correspondiente input_boolean para el intervalo del momento actual. Si está activado, entonces la calefacción se enciende.

El código fuente completo está disponible en **https://goo.gl/WGvoAL**, y se puede instalar con los siguientes comandos:

```
$ cd ~homeassistant/.homeassistant/apps/
$ wget
 https://raw.githubusercontent.com/mad-ady/home-assistant-customizations/master/appdaemon_apps/manual_heater.py
```

También necesitarás editar el archivo apps.yaml y añadir lo siguiente:

```
manual_heater:
 module: manual_heater
 class: ManualHeater
 climate: "climate.heater_thermostat"
 heater: "switch.heater"
 interval_length: 15
```

Ahora, cuando reinicies AppDaemon, la nueva aplicación se cargará y reaccionará al estado de tus input_booleans. Puede seguir su avance leyendo el registro log de AppDaemon.

Todo parece funcionar bien, sin embargo, existe un problema. Si reinicias Home Assistant, o si hay un corte de luz en medio de la noche, los 96 interruptores pasarán a la posición de apagado por defecto. Tiene que haber alguna manera de guardar su estado y hacer que éste se cargue al reiniciar. Afortunadamente ya existe una aplicación para eso: switch_reset.py, disponible en **https://goo.gl/LVYdD2**. Así es como puedes configurarlo:

```
$ cd ~homeassistant/.homeassistant/apps/
$ wget -O switch_reset.py
 https://raw.githubusercontent.com/home-
assistant/appdaemon/dev/conf/example_apps/swit
ch_reset.py
$ wget -O globals.py
 https://raw.githubusercontent.com/home-
assistant/appdaemon/dev/conf/example_apps/glob
als.py
```

Añade la siguiente configuración a apps.yaml:

```
switch_reset:
 module: switch_reset
 class: SwitchReset
 log: ""
 file:
"/home/homeassistant/.homeassistant/switch_sta
tes"
 delay: 10
```

Tras reiniciar AppDaemon, los cambios en las entidades input_boolean, input_number, input_select y device_tracker se almacenarán dentro de /home/homeassistant/.homeassistant/switch_states, haciendo persistentes los estados de nuestros interruptores.

Aunque utilizamos aplicaciones automatizadas, la razón principal por la que las personas usan AppDaemon es porque proporciona una interfaz a modo de cuadro de mandos que permite presentar y controlar las entidades de Home Assistant en una pantalla táctil. Por lo general, las personas usan un televisor o una tablet para mostrar el cuadro de mandos, pero yo he utilizado la LCD 3.5" de HardKernel (**http://www.hardkernel.com/main/products/prdt_info.php?g_code=G147435282441**), con un ODROID-C2 como unidad de procesamiento.

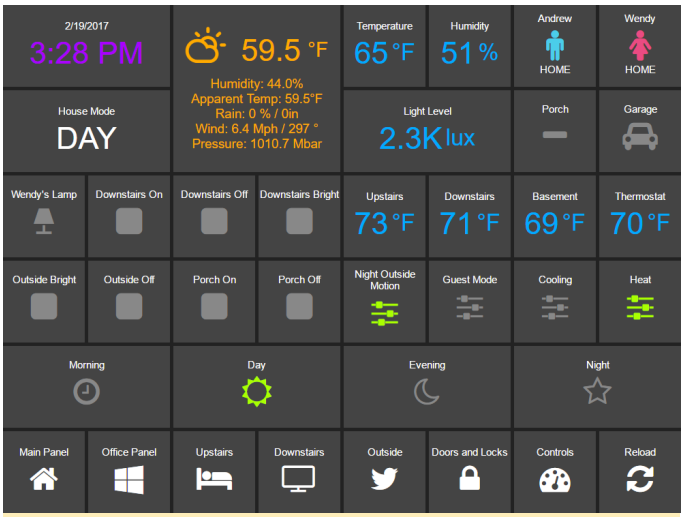


Figura 2 – Ejemplo de cuadro de mandos

Para activar el cuadro de mandos (**https://goo.gl/Z8iMW8**), deberás añadir la siguiente sección a appdaemon.yaml:

```
HADashboard:
 dash_url: http://0.0.0.0:5050
 dash_password: !secret api_password
 dash_dir:
/home/homeassistant/.homeassistant/dashboards
```

La directiva dash_password es opcional. Para facilitar el uso, es mejor no utilizar una contraseña, para que los cuadros de mandos puedan cargarse al arrancar sin la intervención del usuario. Necesitarás crear un cuadro de mando de muestra en ~homeassistant/.homeassistant/dashboards. Primero crea el directorio y luego ponle el nombre hello.dash:

```
$ mkdir
~homeassistant/.homeassistant/dashboards
$ mkdir -p
 /home/homeassistant/.homeassistant/compiled/j
avascript/css
$ mkdir -p
/home/homeassistant/.homeassistant/compiled/cs
s
$ mkdir -p
 /home/homeassistant/.homeassistant/compiled/h
tml/default
$ cd ~homeassistant/.homeassistant/dashboards
$ vi hello.dash
```

hello.dash

```
#
# Main arguments, all optional
#
title: Hello Panel
widget_dimensions: [120, 120]
widget_margins: [5, 5]
columns: 8

label:
 widget_type: label
 text: Hello World

layout:
 - label(2x2)
```

Tras reiniciar AppDaemon, podrás acceder a esto en http://[ip-odroid]:5050/hello, donde [ip-odroid] es la dirección IP del ODROID-C2.

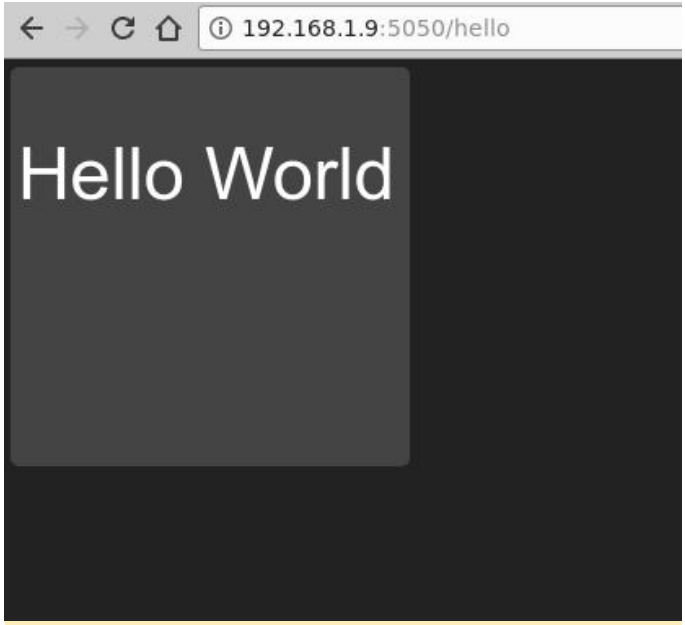


Figura 3 – Hola desde el otro lado

Los cuadros de mandos generan dinámicamente páginas web que pueden representar y controlar los estados de las entidades de Home Assistant. La mayoría de las entidades cuentas con los correspondientes fragmentos de configuración del cuadro de mandos, que permiten controlar las apariencias. La documentación de referencia la tienes disponible en **https://goo.gl/G6iYib**.

Para empezar, debes especificar las dimensiones de la pantalla y pensar en cómo quieres dividir la pantalla en widgets. Normalmente, la pantalla se divide en celdas x*y, y cada celda tiene una anchura y una altura fijas. Dispones de cierta flexibilidad ya que puede combinar celdas para crear una más grande y también puede dejar celdas o filas vacías. El tamaño estándar de la celda es de 120×120 píxeles. Puesto que la pantalla de 3.5” tiene una resolución pequeña (480 × 320), necesitaremos ser creativos e implementar algún tipo de menú para saltar de un panel a otro. También utilizaremos celdas pequeñas, 38×38, con un margen de 1 píxel y las combinaremos para crear widgets más grandes cuando sea necesario. Ten en cuenta que, si utilizas una pantalla lo suficientemente grande puede y deberías usar, tamaños de widget más grandes para evitar así tener problemas con el diseño.

Con respecto a la navegación con menús, HADashboard tiene un widget de navegación que se puede usar para cargar un cuadro de mando diferente. La idea es crear un menú vertical de 8 elementos que aparecerá en todos los cuadros de mandos permitiendo una rápida navegación, luego rellenar los paneles de acuerdo a mis necesidades con datos internos de Home Assistant, sensores, interruptores, reproductores multimedia, cámaras y datos externos tales como gráficos de Netdata o datos meteorológicos online. Uno de los botones puede servir como menú y cargar así un panel diferente con más botones, así que dispones de muchas opciones.

Aquí tienes un ejemplo de encabezado de cuadro de mandos, el cual he duplicado en todos los paneles:

```
#
# Main arguments, all optional
```

```
#
title: 3.5in LCD panel - 480x320 divided into
12x8 cells - Home
widget_dimensions: [38, 38]
widget_margins: [1, 1]
columns: 12
use_gass_icon: 1
```

Para poner en marcha la navegación, he configurado una lista de "botones" en un archivo diferente llamado ~homeassistant/.homeassistant/dashboards/navigation-definition.yaml, que se incluirá en todos los paneles y que tendrá un aspecto similar al siguiente, solo se muestran algunos elementos para que te hagas una idea:

```
home:
 widget_type: navigate
 dashboard: "lcd35-hq"
 icon_active: fa-home
 icon_inactive: fa-home
mpd:
 widget_type: navigate
 dashboard: "mpdkitchen"
 icon_active: fa-music
 icon_inactive: fa-music
tv:
 widget_type: navigate
 dashboard: "tv"
 icon_active: mdi-television-classic
 icon_inactive: mdi-television-classic
 icon_style: "font-size: 1.5em !important;"
 …

extendedmenu:
 widget_type: navigate
 dashboard: "extendedmenu"
 args:
 timeout: 10
 return: lcd35-hq
 icon_active: mdi-menu
 icon_inactive: mdi-menu
```

Puedes conseguir una copia completa del archivo en **https://goo.gl/vwYD33**. Como puede ver, la mayoría de las entradas son widgets de navegación que incluyen el nombre de archivo del cuadro de mando como parámetro y puede tener un estilo o icono personalizados. El ítem "extendedmenu" hace que este panel se cargue durante 10 segundos, luego se carga el panel "lcd35-hq", si no hay otra acción. Esto te permite simular un menú pop-up que desaparece por sí sólo.

El diseño de los widgets sobre la página se lleva a cabo creando una directiva de diseño con las filas que se detallan a continuación. Para cada celda de una fila, debes escribir el nombre del widget tal y como está definido en el cuadro de mandos o los archivos importados. Los nombres de los widgets pueden tener su tamaño anexado al nombre. Con el fin de reutilizar la configuración tanto como sea posible, también puedes importar definiciones externas desde otros paneles, como el fragmento de navegación anterior. El siguiente fragmento es del cuadro de mandos principal:

```
layout:
 - include: navigation-definition
 - include: sensors
 - home, clock(4x4), weather(7x4)
 - mpd
 - webcams
 - tv
 - heating, sensorliving(2x4),
sensorkids(2x4), heater(2x4), forecast(5x4)
 - cooling
 - blinds
 - extendedmenu
```

Los elementos más a la izquierda de la lista son los destinos de la navegación definidos en el archivo navigation-definition.yaml. Desafortunadamente, puesto que quería disponer de un menú vertical, necesito añadirlo específicamente a cada cuadro de mando. Si sólo tuviera una fila horizontal, podría haber hecho el diseño dentro del archivo navigation-definition.yaml.

La fila superior empieza con un pequeño widget "home" utilizado para empezar a navegar de nuevo desde aquí, luego un widget de reloj 4×4 y un widget de clima 7×4. La siguiente fila solo muestra "mpd", que forma parte del menú. El resto de la fila está ocupada por los grandes widgets de reloj y clima, de modo que no hay añadido ningún otro elemento más. Usando esta lógica, puedes hacerte una imagen mental de cómo se supone que debería verse. El

resto de los elementos están definidos en el cuadro de mando y en el sensors.yaml incluido:

```
clock:
 widget_type: clock
 time_format: 24hr
 show_seconds: 0
 time_style: "color: yellow; font-size: 40pt;
font-weight: bold;"
 date_style: "font-size: 16pt; font-weight:
bold;"

weather:
 widget_type: weather
 units: "°C"
 sub_style: "font-size: 110%; font-weight:
bold;"
 main_style: "font-size: 75%; font-weight:
bold;"
 unit_style: "font-size: 250%;"

forecast:
 widget_type: sensor
 title: Prognoza
 title_style: "font-size: 14pt;"
 text_style: "font-size: 16pt; font-weight:
bold;"
 precision: 0
 entity: sensor.dark_sky_forecast_ro
```

Como puede ver, la mayoría de los widgets requieren una entidad que proporcione el lazo de unión con los elementos de Home Assistant, un "type" y el resto de configuración gestiona las fuentes, los colores y los iconos. Los iconos de los widgets pueden proceder de Home Assistant, aunque pueden sustituirse por iconos de Material Design, **https://materialdesignicons.com**, con el prefijo mdi, o de Font Awesome, **http://fontawesome.io**, con el prefijo fa.

Si tuvieras que cargar este cuadro de mandos ahora mismo en un navegador introduciendo http://ip-odroid:5050/lcd35-hq, se vería como en la Figura 4.

Figura 4 – Panel de control con la navegación inservible

Sin embargo, parece haber un problema con el diseño de los widgets de navegación. Si utilizaras las herramientas de desarrollo de tu navegador y analizas el diseño, veras que, aunque los widgets están colocados correctamente, los iconos heredan un estilo CSS que usa un posicionamiento absoluto que desplaza el icono de 43 píxeles hacia abajo. Esto es un problema porque el cuadro de mandos fue diseñado para pantallas más grandes con widgets más grandes. Para evitar este problema, lo mejor es crear un skin que cargue un archivo JavaScript personalizado que restablezca el diseño absoluto de los iconos y también ajuste el tamaño. Para hacer esto, necesitarás algunos conocimientos de Javascript, HTML y CSS, aunque puedes conseguir el skin completo en **https://goo.gl/Fwcbti**.

```
$ sudo su - homeassistant
$ cd .homeassistant/
$ mkdir -p custom_css/defaultsmall
$ cd custom_css/defaultsmall
$ wget -O dashboard.css
 https://raw.githubusercontent.com/mad-
ady/home-assistant-
customizations/master/appdaemon_skins/defaults
mall/dashboard.css
$ wget -O dashboardsmall.js
 https://raw.githubusercontent.com/mad-
ady/home-assistant-
customizations/master/appdaemon_skins/defaults
mall/dashboardsmall.js
$ wget -O variables.yaml
 https://github.com/mad-ady/home-assistant-
customizations/blob/master/appdaemon_skins/def
aultsmall/variables.yaml
```

Ahora, si vuelve a cargar el cuadro de mandos y especificas un skin especifico, conseguirás mejores resultados (http://[ip-odroid]: 5050/lcd35-hq?Skin=defaultsmall)

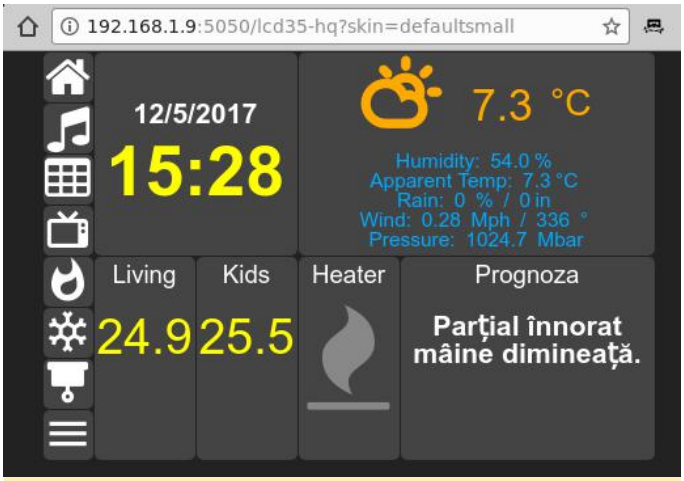


Figura 5 – Cuadro de mandos con navegación

Ahora puedes echar un vistazo a mis cuadros de mandos. Toda la configuración está en **https://goo.gl/VuB9sr**. El panel MPD se compone de 3 cuadros de mandos similares en los que puedo desplazarme utilizando los widgets de navegación superiores. Éstos cargan 3 instancias MPD diferentes en la casa. La diferencia entre los paneles únicamente es la instancia cargada (ver mpdkitchen más abajo). El diseño del cuadro de mando es bastante simple:

```
layout:
 - include: navigation-definition
 - include: mpd
 - home, navigationmpdliving(4x1),
navigationmpdkids(4x1),
navigationmpdkitchen(3x1)
 - mpd, mpdkitchen(11x7)
 - webcams
 - tv
 - heating
 - cooling
 - blinds
 - extendedmenu
```

Figura 6 – Tres instancias MPD

También tengo una vista del cuadro de mandos para controlar mi TV, que fue importada a Home Assistant tal y como se describe en **https://magazine.odroid.com/article/home-assistantscripts-customization/**. Estamos obteniendo la imagen de un componente de cámara dentro de Home Assistant y usando el Widget de cámara. Los botones se utilizan para controlar el control remoto virtual y conectarse a los componentes del script en Home Assistant, así como a la entidad del script en HADashboard. A continuación, tienes un ejemplo de diseño y de los widgets:

```
layout:
 - include: navigation-definition
 - home, streamtv(10x8), tv_living_off
 - mpd, tv_living_on
 - webcams, tv_living_source
 - tv, tv_living_mute
 - heating, tv_living_volume_up
 - cooling, tv_living_volume_down
 - blinds, tv_living_ch_up
 - extendedmenu, tv_living_ch_down

streamtv:
 widget_type: camera
 entity_picture:
http://192.168.1.4:8123/api/camera_proxy/camer
a.tv_living_image?
token=62f78994c790a89459e2f60cc6ed80bdfce3e9b5
ff5473633ba60e3d7089f0a6&api_password=odroid
 refresh: 2

tv_living_off:
 widget_type: script
 entity: script.tv_living_power_off
```

```
  icon_on: mdi-power-plug-off
  icon_off: mdi-power-plug-off

tv_living_on:
  widget_type: script
  entity: script.tv_living_power_on
  icon_on: mdi-power
  icon_off: mdi-power
```

Lo que diferencia al widget de cámara es que necesita una URL a través de la API de Home Assistant. Esta URL debe incluir la clave API y también un token que es visible en Home Assistant -> Entities para la entidad en cuestión. Si también usas una contraseña para acceder a Home Assistant, deberás añadirla a la URL. La figura 7 muestra el resultado final.

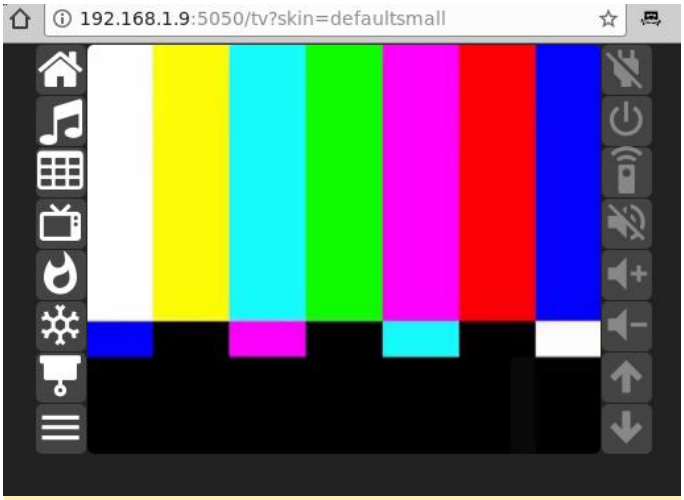


Figura 7 – Cuadro de mandos para monitorizar el TV

Otro cuadro de mandos permite controlar el calentador, manualmente y el termostato. La configuración para el widget del termostato es similar a la que aparece a continuación, y el resultado final lo puedes ven en la Figura 8.

```
layout:
  - include: navigation-definition
  - include: sensors
  - home, heater(4x4), thermostat(6x8)
  - mpd
  - webcams
  - tv
  - heating, sensorliving(2x4), sensorkids(2x4)
  - cooling
  - blinds
  - extendedmenu

thermostat:
  widget_type: climate
  title: Thermostat
  step: 0.5
  precision: 1
  entity: climate.heater_thermostat
  unit_style: "color: yellow;"
  level_style: "color: yellow; font-size:
48pt;"
  unit2_style: "color: yellow;"
  level2_style: "color: yellow; font-size:
40pt;"
  level_up_style: "font-size: 20pt;"
  level_down_style: "font-size: 20pt;"
```

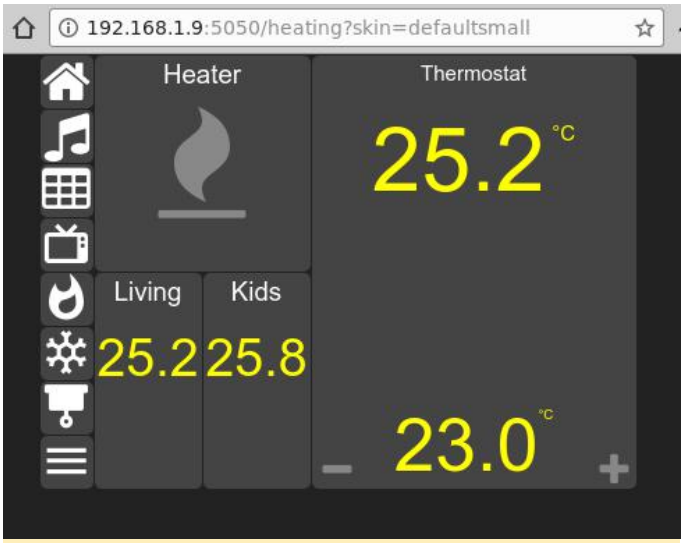


Figura 8 – Panel del calentador

Durante el verano, el panel del aíre acondicionado tendrá un cierto uso. Aquí se colocan los sensores, interruptores y temporizadores que controlan el sistema de AC, como se describe en https://magazine.odroid.com/article/home-assistantscripts-customization/. El panel es similar al se muestra en la Figura 9.

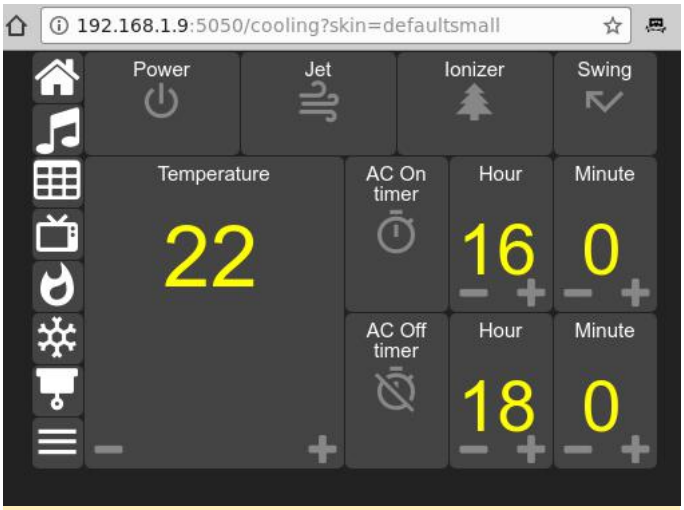


Figure 9 – Control del Aire Acondicionado

El menú extendido no es más que un panel con más widgets de navegación. En él, tengo enlaces a un cuadro de mandos de Netdata y un pronostico meteorológico cada hora e imágenes de mis cámaras web, con espacio para más en el futuro. El panel de pronóstico del tiempo por hora utiliza el widget iframe para cargar una URL, mientras que el Panel de Netdata carga una lista de URLs cada 5 segundos:

```
mynetdata:
 widget_type: iframe
 refresh: 5
 url_list:
 - http://192.168.1.5:19999/server1.html
 - http://192.168.1.5:19999/server2.html
 - http://192.168.1.5:19999/server3.html
 - http://192.168.1.5:19999/server4.html
 - http://192.168.1.5:19999/server5.html
```

Figura 10 – Menú extendido

Ahora que tienes el cuadro de mandos funcionando a tu gusto, tendrás que invertir un poco de tiempo en el tema de la pantalla. Puedes empezar con el script de instalación de @FourDee para la pantalla de https://forum.odroid.com/viewtopic.php?t=24248. También necesitarás activar el acceso automático en lightdm (https://wiki.odroid.com/accessory/display/3.5inch_lcd_shield/autox#auto_login). Una vez hecho esto, es hora de realizar una limpieza. Lo mejor es eliminar aquellos programas que muestran ventanas pop-up en pantalla, como son el gestor de actualizaciones y el protector de pantalla:

```
$ sudo apt-get remove update-manager gnome-
screensaver
```

También prepararemos un script que ejecute Chromium en modo Kiosco, sin almacenamiento de contraseñas, para que no te solicite contraseña cada vez que se desbloquee. Chromium también se configurará con un parche que hace que se olvide de que se ha bloqueado, por lo que, en caso de un cierre no limpio, no te preguntará si deseas restaurar la sesión anterior. Además de esto, configuraremos el monitor para que siempre esté encendido:

```
$ cat /usr/local/bin/kiosk-mode.sh
 #!/bin/bash
 /usr/bin/xset s off
 /usr/bin/xset -dpms
 /usr/bin/xset s noblank
 /bin/sed -i 's/"exited_cleanly":
false/"exited_cleanly": true/'
~/.config/chromium/Default/Preferences
 dashboard=lcd35-hq
 /usr/bin/chromium-browser --noerrdialogs --
incognito --password-store=basic --kiosk
http://odroid-ip:5050/$dashboard?
skin=defaultsmall
$ sudo chmod a+x /usr/local/bin/kiosk-mode.sh
```

También puedes añadir la extensión de Scrollbar Anywhere Chrome (https://goo.gl/UD3hDA). Configúrala para que reaccione con el botón izquierdo y active “Use Grab-and-drag style scrolling” para que puedas desplazarte, si fuera necesario, con el dedo sobre el cuadro de mandos.

Comprueba que el script funciona correctamente cuando se inicia desde el entorno gráfico y, cuando esté listo, puedes añadirlo a la lista de aplicaciones que se inician automáticamente en Control Center -> Personal -> Startup Applicat. Simplemente haz clic en “add”, usa “Kiosk mode” en el nombre y /usr/local/bin/kiosk-mode.sh como comando. Una vez que reinicies lightdm, deberías ver un panel a pantalla completa.

Lo que muestra la Figura 11 es el resultado final, ejecutándose en una pantalla de 3.5”. La pantalla de 3.5 “de Hardkernel es perfecta para un pequeño cuadro de mandos. Los tamaños y colores de fuente utilizados están optimizados para una fácil lectura a una distancia de 2-3 metros y el contraste ayuda a leer desde ángulos más amplios. La baja velocidad de

refresco de la pantalla de unos 10 fps no se aprecia con el cuadro de mandos.

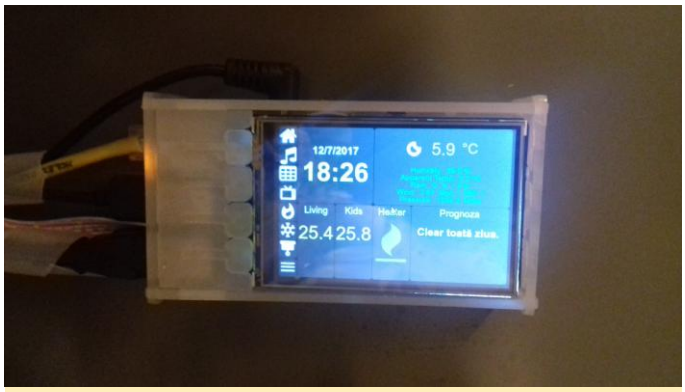
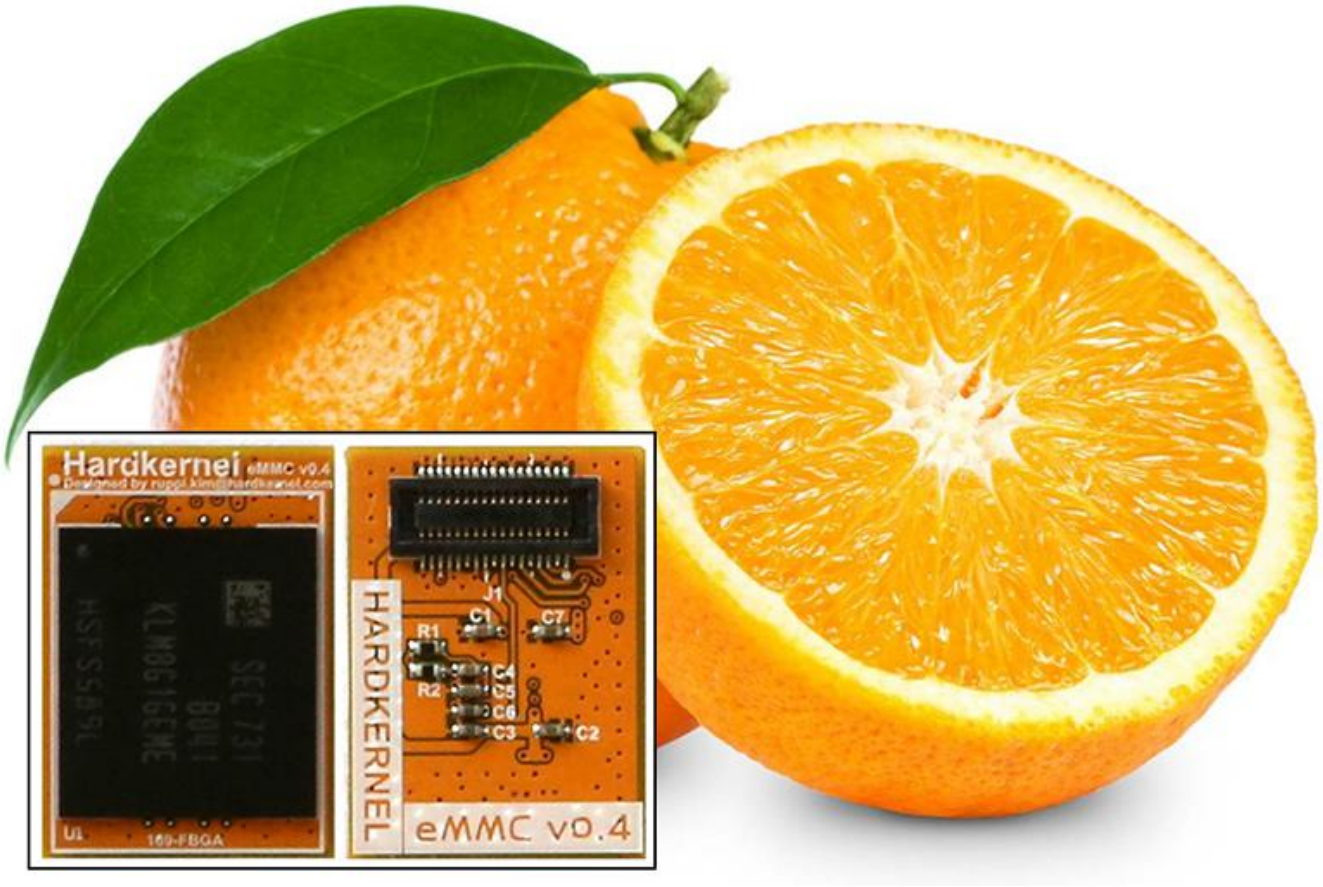


Figura 11 – Pantalla de 3.5 “con el cuadro de mandos

Puedes conseguir los archivos de configuración del cuadro de mandos desde proyecto GitHub en **https://github.com/mad-ady/home-assistant-customizations**, y ver un video de demostración en **youtu.be/fEoHs3-_3B0**. Pen en cuenta que todo ha sido probado con el appdaemon 2.1.12, puesto que la versión 3 está actualmente en desarrollo, para cuando pongas en funcionamiento esto, quizás algunas cosas hayan cambiado ligeramente. Está atento al repositorio de GitHub y al hilo de soporte en **https://forum.odroid.com/viewtopic.php?t=27321**.

Módulo eMMC Naranja: Llega el chipset Samsung 5.1

January 1, 2018 By Justin Lee ODROID-C2



Hardkernel ha presentado ahora el módulo eMMC naranja, que utiliza el chipset eMMC 5.1 de Samsung, el cual se ha estado enviando desde octubre de 2017.

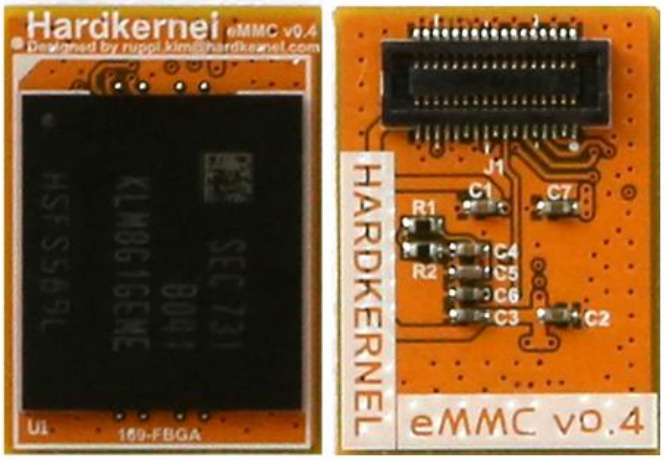


Figure 1 – Hardkernel now offers an orange eMMC module)

Funciona con las series C1/C2/XU4 y el sistema operativo adecuado. Las últimas imágenes oficiales del sistema operativo funcionan correctamente. Los esquemas están disponibles en **eMMC PCB Rev 0.4**.

Compatibilidad del Módulo eMMC naranja con imágenes de sistema operativo de la serie XU4

Imagen SO	Información del Archivo de Imagen	Estado
Ubuntu Mate	ubuntu-16.04.3-4.14-mate-odroid-xu4-20171212.img	OK
Ubuntu Minimal	ubuntu-16.04.3-4.14-minimal-odroid-xu4-20171213.img	OK
Android 7.1.1	Alpha-1.1_14.11.17	OK
Android TV 7.1.1	Alpha-1.0_20.11.17	OK
Android 4.4.4	Android 4.4.4 (v5.8)	OK
Debian Jessie	Debian-Jessie-	OK

	1.1.4-20171121-XU3+XU4.img	
ODROID Game Station Turbo (OGST)	ODROID-GameStation-Turbo-3.9.5-20171115-XU3+XU4-Jessie.img	OK
Armbian	All Armbian variants starting with version 5.35	OK
OMV	OMV_3_0_92_Odroidxu4_4.9.61	OK
DietPi	DietPi_OdroidXU4-armv7-(Jessie).7z 22-Nov-2017	OK
Yocto project	No Flashable file reference	No Probado
Kali-Linux	No Flashable file reference	No Probado
Arch-Linux	No Flashable file reference	No Probado
ROS	No Flashable file reference	No Probado
Lakka	Lakka-OdroidXU3.arm-2.1-rc6.img.gz	OK
Batocera	batocera-5.12-xu4-20171214.img.gz	OK
RecalBox	recalbox (17.11.10.2)	¿OK?
RetroPie	No Flashable file reference	No Probado

Los eMMC de Sandisk tiene una versión hasta la 5.1 con velocidad de transferencia de datos ligeramente más rápida a partir del 20 de julio de 2017. Tal y como se muestra en la Figura 2, el código QR está en el lado izquierdo del chipset ver. 5.1 mientras que el eMMC ver. 5.0 lo tiene en el lado derecho.

Figura 2 – El eMMC versión 5.0 (izquierda) tiene el código QR en el lado derecho, y eMMC versión 5.1 (derecha) tiene el código QR en el lado izquierdo

El Kernel versión 3.10 debería tener los siguientes parches aplicados para que funcione correctamente con la serie XU4: **Github**, **Github**, **Github**.

Figura 3 – Tabla de módulos eMMC de productos actuales en ejecución 2016

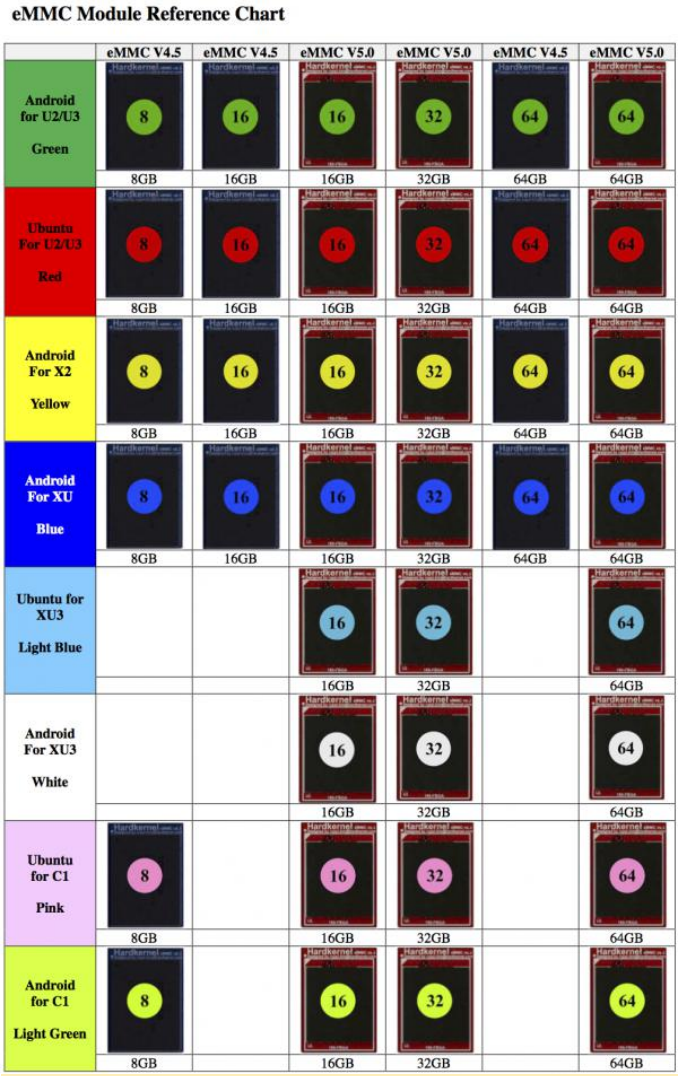


Figura 4 – Módulos eMMC de antiguos productos

Referencias

Esquemas del módulo eMMC Revison 0.3 Esquemas del módulo eMMC amarillo revisión 0.4 Esquemas de la placa lectora eMMC

- Dimensiones de la placa eMMC: 18.5mm x 13.5mm
- Hueco entre las PCBs: 1.1mm (Altura de los conectores B2B ensamblados)

El conector está hecho por LS-Mtron Korea. En el módulo eMMC, se utilizó el GB042-34S-H10 (Socket-34pin). En la placa host, se utilizó el GB042-34P-H10 (clavija-34pin). Las especificaciones del conector están aquí Información sobre el eMMC Sandisk (iDisk Extreme) Información sobre el eMMC Samsung Información sobre el eMMC Essencore (el eMMC de 8GB se utiliza para XU4) Información sobre eMMC Toshiba

Pruebas de lectura/escritura del eMMC bajo el modo HS400 ODROID-C2 (Unidad: MByte/seg)

		Samsung	Toshiba	Sandisk
8G	Escritura	45.4	21.9	N/A
8G	Lectura	113	148	N/A
16G	Escritura	80.1	N/A	25.6
16G	Lectura	126	N/A	153
32G	Escritura	124	N/A	98.7
32G	Lectura	125	N/A	153
64G	Escritura	124	83.7	107
64G	Lectura	124	153	153

Comando de lectura/escritura para la pruebas de rendimiento del eMMC:

```
$ dd if=/dev/zero of=test.tmp oflag=direct
bs=1M count=1024
$ dd if=test.tmp of=/dev/null iflag=direct
bs=1M
```

ODROID-C2 + Prueba de rendimiento del Emmc negro en condiciones de E/S de archivos

- Ubuntu 16.04
- Versión Kernel : Linux odroid64 3.14.79-115
- Herramienta para pruebas : iozone revision 3.429

Prueba de instalación y rendimiento de iozone:

```
$ sudo apt install iozone3
$ iozone -e -I -a -s 100M -r 4k -r 16k -r 512k
-r 1024k -r 16384k -i 0 -i 1 -i 2<
/* 8G */
random random
kB reclen write rewrite read reread read write
102400 4 9290 13582 13570 13568 11900 8787
102400 16 10934 15680 27511 27484 25976 7699
102400 512 14943 23761 42163 42121 41361 15122
102400 1024 15140 28564 41951 41915 41196
16743
102400 16384 16559 24001 42308 42267 42287
28604
/* 16G */
random random
kB reclen write rewrite read reread read write
102400 4 14602 14622 18102 17953 16768 14421
102400 16 49363 49279 52902 52808 47450 48389
```

```
102400 512 49779 49993 138268 138315 137171
48836
102400 1024 50005 49870 137522 137709 136958
49027
102400 16384 49861 50058 139358 139154 139299
50024
/* 32G */
random random
kB reclen write rewrite read reread read write
102400 4 14608 14670 18333 18343 17935 14624
102400 16 58393 66157 56412 56766 55744 56371
102400 512 80356 81074 136828 137132 137503
79224
102400 1024 80464 81036 137368 137278 136896
79191
102400 16384 80388 81070 139486 139612 139446
80560
/* 64G */
random random
kB reclen write rewrite read reread read write
102400 4 14240 14299 17619 17548 16012 14216
102400 16 49991 57484 53245 53405 50001 59302
102400 512 132316 135079 134154 134016 134208
129755
102400 1024 132476 134966 133753 133840 133677
130054
102400 16384 135772 139140 136133 136019
135821 135107
/* 128G */
random random
kB reclen write rewrite read reread read write
102400 4 14162 14152 18161 18184 17833 14200
102400 16 56527 64906 55057 55684 54492 66525
102400 512 131327 131444 137307 137040 137358
132500
102400 1024 131908 131896 137570 137495 136844
132365
102400 16384 136418 134070 139940 133304
121160 134002
```

El módulo eMMC negro está hecho con el chipset eMMC de Samsung. El módulo eMMC rojo y azul (normal) está hecho con chipset Sandisk o Toshiba o AIO. Los dispositivos ODROID-C1/C0/C1+/C2 funcionan con los módulos eMMC negro y rojo. Los dispositivos ODROID-XU4/XU3/U3/X2/U2 NO funcionan con el módulo eMMC negro.

Testeo del nuevo módulo eMMC de 8 GB en Ubuntu XU4

La nueva PCB roja eMMC de 8GB para el modelo ODROID-XU4 está basada en la tecnología eMMC 5.0 de Essencore/AIO. Velocidad secuencial con prueba “dd”:

- Escritura dd: 15.1 MB/s
- Lectura dd: 104 MB/s
- Prueba de velocidad de acceso aleatorio (IOPS) con bloque 4k.
- Escritura aleatoria : io=993228KB, bw=9928.2KB/s, iops=2482
- Lectura aleatoria : io=1479.1MB, bw=15149KB/s, iops=3787

Comparando el rendimiento del Módulo eMMC frente a la Tarjeta SD en el C2 con Android utilizando una PCB negra eMMC de 16 GB y una tarjeta SDHC UHS-1 de 16 GB (modelo OEM Sandisk SDSDQAD-016G UHS-I 50), con una imagen Android 5.1 V2.8 limpia y con el paquete GApps Pico instalado:

- Tiempo de arranque del eMMC desde el encendido: 18 ~ 20 segundos
- Tiempo de arranque de la SDHC desde el encendido: 32 ~ 35 segundos

Puntos de control para los desarrolladores de software de sistemas

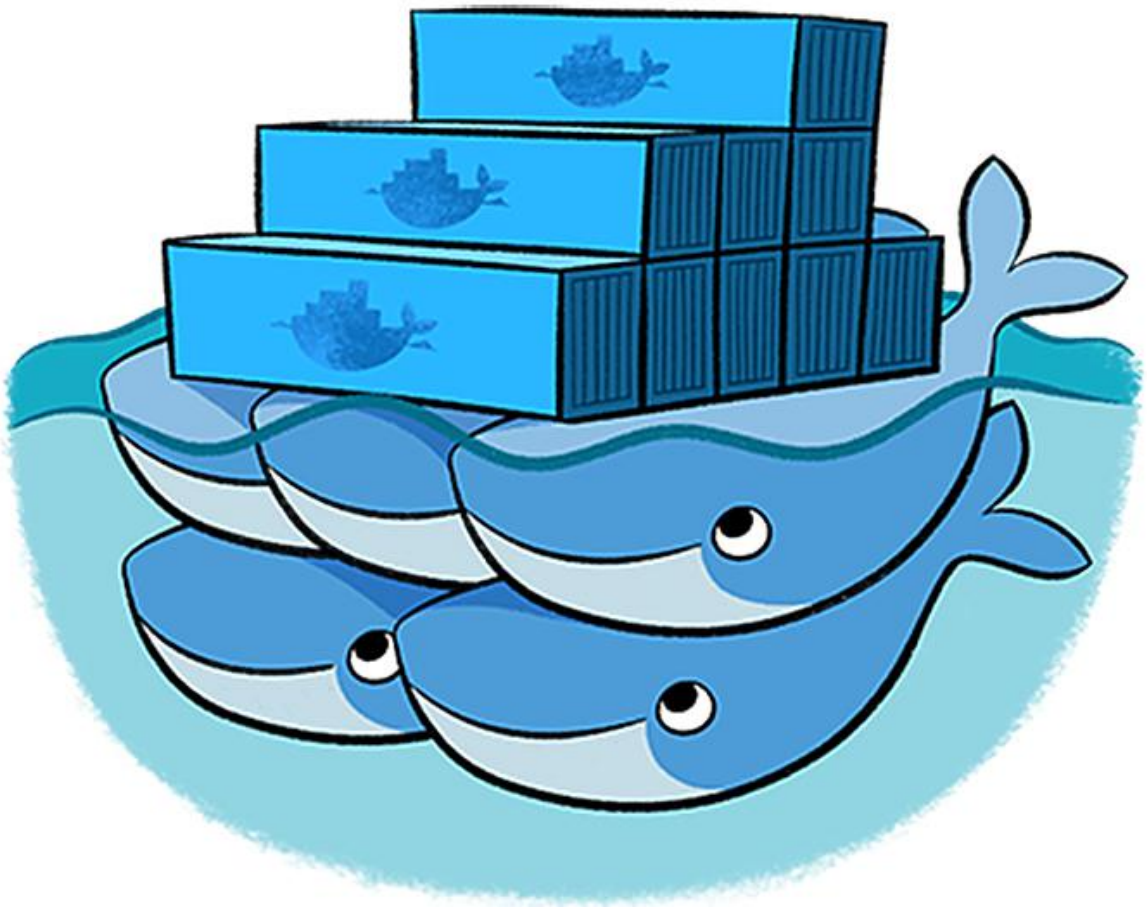
No sobrescribas la partición de arranque oculta del eMMC. Si es así, visita cómo recuperar el cargador de arranque del eMMC para solucionarlo. El eMMC debería estar dividida de la siguiente forma:

- Partición FAT16 con UUID 6E35-5356 (arranque)
- Partición EXT4 con e139ce78-9841-40fe-8823-96a304a09859 (Linux)

Copia los contenidos de las particiones de la imagen de Ubuntu a las particiones de arranque y Linux usando “cp -afpv source destination”, luego inserta el módulo eMMC y arranca como normalmente lo haces. Para comentarios, preguntas y sugerencias, visite la página wiki original en **https://wiki.odroid.com/accessory/emmc/reference_chart**.

Recompilando imágenes Docker x86amd64 para Swarm ARM

January 1, 2018 By Mike Partin Docker, Linux, Tutoriales



Tras finalizar los recientes artículos sobre cómo compilar swarm Docker en ARM (**https://goo.gl/2FjP8f**) y (**https://goo.gl/ZTXcp**), me encontré con que quería activar servicios, para los cuales no había imagen ARM disponible, o no había ninguna imagen ARM con una versión reciente. Las alternativas son tres: 1) prescindir de lo que quería, 2) conformarme con una versión anterior, o 3) averiguar cómo compilar lo que quería. Soy un poco chapucero y me gusta tener al menos un conocimiento básico de mis herramientas, así que me decanté por la tercera opción. En su mayor parte, se trata un proceso bastante sencillo, aunque de vez en cuando, terminas modificando algunas cosas. No te deje intimidar por ello, porque realmente merece la pena, y no es tan difícil. Para facilitar las cosas, configuremos una pila Graphite. Para esta pila, necesitaré varios elementos, además de un poco de soporte en forma de registro de imágenes alojado internamente.

Un registro

Usaremos una imagen ya creada para esto de **http://dockr.ly/2kmNgod**. El registro proporciona un lugar para almacenar e implementar tus imágenes personalizadas. Se trata de un excelente escenario antes de enviar tu producto final a **https://hub.docker.com** o a cualquier otro registro que elijas.

El front end del registro es simplemente un pequeño servicio muy útil para poner en marcha un registro (**http://dockr.ly/2D5DRt3**). No entraremos en funciones avanzadas como eliminar imágenes, lo cual requiere una configuración adicional en el registro, pero podremos navegar por nuestras imágenes y obtener información sobre ellas.

Utilizaré go-carbon para el caché (**https://goo.gl/hgjGZo**). La razón fue simple, go-carbon usa más de un núcleo en una única instancia. Es realmente fácil de configurar, y si tienes alguna

definición de esquema, puede funcionar muy bien y también es compatible con el formato pickle.

Utilizaré graphite-api para el punto final de la API de renderizado **https://goo.gl/P43pHC**. Hay otras opciones como carbonserver y carbonzipper. Creo que go-carbon ahora tiene soporte para carbonserver, pero aún no lo he probado. Creo que ir en modo stock no es tan malo en este caso. Solo se consulta de vez en cuando, así que no se necesita tener un rendimiento tan alto como el caché.

Usaré grafana para la interfaz de usuario de la pantalla, ya que es bastante común (**https://grafana.com**). Podríamos usar el paquete graphite-web, aunque las posibilidades de representación gráfica son las mismas, son bastante menos accesibles que las de Grafana. También existen otras opciones, si bien deberías analizarlas antes de tomar cualquier decisión sobre lo que más te conviene.

Implementar la infraestructura

Dado que el swarm es realmente de alta disponibilidad con balanceo de carga de esos servicios (tanto a nivel de hardware como de red), los servicios que hemos enumerado anteriormente se lanzarán como servicios independientes. Ten en cuenta que hay múltiples opciones para cada uno, de modo que cubrirlas todas es demasiado para este artículo. Una vez dicho esto, centrémonos en nuestras necesidades de infraestructura e implementemos nuestro registro:

```
$ docker service create --name=docker-registry
--publish=5000:5000/tcp cblomart/rpi-registry
```

Una vez que se complete el comando, dispondremos de un registro. Sin embargo, tenemos el problema de que ninguna de nuestras instancias Docker lo usará porque no es seguro. Tenemos dos opciones: la primera es configurar tus instancias docker para que usen un registro inseguro (concretamente una lista blanca), y la segunda es obtener un certificado (autofirmado o públicamente verificable). Este es un registro interno. Para ir avanzando, he optado por la primera opción. Para hacerlo así en todos los nodos docker, añadí lo siguiente a /etc/docker/daemon.json, asumiendo que dispones de una configuración por defecto y que el archivo está completo:

```
{
    "insecure-registries": ["10.0.0.15:5000"]
}
```

Interfaz de Usuario del Registro Docker

Ahora vamos a movernos para que la interfaz del registro esté en su lugar, lo cual nos permitirá compilar nuestra primera imagen. Ten en cuenta que yo he usado el nombre de host swarm. Esto es lo correcto para mi configuración, ya que tengo un registro CNAME en mi servidor DNS, aunque tu configuración puede ser diferente:

```
$ git clone
https://github.com/parabuzzle/craneoperator.gi
t
$ cd craneoperator
$ docker build -t docker-registry-ui .
$ docker tag docker-registry-ui
swarm:5000/docker-registry-ui-arm
$ docker push swarm:5000/docker-registry-ui-
arm
```

Ahora podemos lanzar nuestro servicio. Éste, como muchos otros, utiliza variables de entorno que influyen en su funcionamiento. Por supuesto, tendrás que editarlas para apreciar los cambios:

```
$ docker service create --name=docker-
registry-ui --publish=8081:80/tcp -e
REGISTRY_HOST=swarm -e REGISTRY_PROTOCOL=http
-e SSL_VERIFY=false swarm:5000/docker-
registry-ui-arm
```

Caché Carbon y API renderizado

Go-carbon es una aplicación Go que requiere Go 1.8+. Para evitar problemas innecesarios nos basaremos en una versión reciente de Go que está disponible en mi gestor de paquetes favorito, simplemente colocaremos la última versión en una ubicación personalizada. En este momento, Go 1.9.2 es la última versión, así que la usaremos. Además, doy por hecho que estás ejecutando Linux en una máquina ARM, como un ODROID-XU4, que es una maravillosa estación de trabajo. Manejo cada uno de mis 3 monitores con un XU4, y uso x2x para poder

compartir el teclado y el mouse, aunque esto da para un artículo que escribiremos en otro momento.

```
$ cd ~
$ mkdir -p ~/.golang/path
$ wget
https://redirector.gvt1.com/edgedl/go/go1.9.2.
linux-armv6l.tar.gz
$ tar -zxf go1.9.2.linux-armv6l.tar.gz
$ mv go .golang/root
$ export GOROOT=${HOME}/.golang/root
$ export GOPATH=${HOME}/.golang/path
$ export
PATH=${GOROOT}/bin:${GOPATH}/bin:${PATH}
```

Ahora estamos listos para empezar a trabajar en go-carbon. Este es realmente bueno, puesto que dispone de un archivo Dockerfile ya creado, y lo único que tenemos que hacer es compilar el binario y preparar nuestros archivos de configuración. Obtener la fuente y compilar el binario se puede hacer de una sola vez:

```
$ go get -v github.com/lomik/go-carbon
```

Ahora que tenemos esto hecho, vamos a compilar nuestra imagen Docker:

```
$ cd ${GOPATH}/src/github.com/lomik/go-carbon
$ cp ${GOPATH}/bin/go-carbon .
$ mkdir config-examples
```

Seguimos adelante y nos detendremos aquí, ya que necesitamos modificar el archivo de configuración. Hay un gran número de opciones, pero es probable que no necesites modificar ninguna. Dejaré que seas tú el que se ocupe más adelante de las personalizaciones, simplemente daré por sentado que los valores por defecto son lo suficientemente buenos por ahora. La única modificación que haremos es apuntar al archivo de esquemas correcto y a nuestro directorio de datos. Podemos hacer esto con un simple comando sed:

```
$ ./go-carbon -config-print-default | sed -E
's,(schemas-file =).*, "/data/graphite/go-
carbon-schemas.conf",g' | sed -E 's,(data-dir
=).*, "/data/graphite/whisper",' > ./conf-
examples/go-carbon.conf
```

A continuación, podemos conseguir nuestro archivo de esquemas en /conf-examples/go-carbon-schemas.conf:

```
[default]
pattern = .*
retentions = 10s:1h, 30s:3h, 60s:6h, 1h:1d,
6h:1w, 12h:1m, 24h:1y
```

Esto nos proporciona mucho espacio para que nuestras estadísticas se almacenen y se mantengan en el tiempo. Llegados a este punto, estamos listos para empezar a compilar nuestra imagen Docker. Supongo que, a afectos de este artículo, te encuentras en la misma máquina en la que compilaste go-carbon y en la que tienes instalado Docker.

```
$ docker build -t go-carbon . &&
$ docker tag go-carbon swarm:5000/go-carbon-
arm &&
$ docker push swarm:5000/go-carbon-arm
```

Ahora podemos publicar nuestro servicio para nuestro swarm. Ya lo hemos hecho varias veces, así que debería serte familiar:

```
$ docker service create --name=carbon-cache --
publish=2003:2003/tcp --publish=2003:2003/udp
swarm:5000/go-carbon-arm
```

Sin embargo, ahora nos encontramos con otro inconveniente. Necesitamos tener instalada la graphite-api en la imagen, ya que necesitamos disponer de acceso a los archivos whisper. Podríamos crear un sistema de archivos compartido y montarlo tanto para la memoria caché como para las imágenes API, pero en este caso, creo que es mejor que simplemente modifiquemos nuestra imagen go-carbon para que soporte ambos. Lo primero a tener en cuenta es que la imagen docker se llame “busybox”. Como necesito Python, decidí mover esto a ‘debian: stretch’. Lo primero es conseguir nuestro graphite-api.yaml, que se encuentra en ./conf-examples/graphite-api.yaml:

```
search_index: /data/graphite/index
finders:
  - graphite_api.finders.whisper.WhisperFinder
functions:
  - graphite_api.functions.SeriesFunctions
  - graphite_api.functions.PieFunctions
whisper:
```

```
  directories:
    - /data/graphite/whisper
```

Puesto que estamos iniciando más de un servicio, debemos usar un script de punto de entrada personalizado. Seguimos avanzando y escribimos esto.

```
#!/bin/sh
gunicorn -b 0.0.0.0:8000 -w 2
graphite_api.app:app &
sleep 2
/go-carbon -config /data/graphite/go-
carbon.conf
```

Después de desplazarnos a debian:stretch e instalar nuestros paquetes y configuraciones, nuestro Dockerfile en nuestro directorio go-carbon debería verse ahora de la siguiente forma:

```
FROM debian:stretch
RUN mkdir -p /data/graphite/whisper/
RUN apt update && apt upgrade -y && apt dist-
upgrade -y && apt autoremove -y
RUN apt install -y gunicorn graphite-api
ADD go-carbon /
ADD entrypoint.sh /
ADD conf-examples/* /data/graphite/
RUN chmod +x /entrypoint.sh
RUN rm /etc/graphite-api.y* ; ln -s
/data/graphite/graphite-api.yaml
/etc/graphite-api.yaml
CMD ["/entrypoint.sh"]
EXPOSE 2003 2004 7002 7007 2003/udp 8000
VOLUME /data/graphite/
```

Avancemos y recompilemos, luego impulsemos nuestra nueva imagen:

```
$ docker build -t go-carbon . &&
$ docker tag go-carbon swarm:5000/carbon-
cache-arm &&
$ docker push swarm:5000/carbon-cache-arm
```

Después eliminaremos nuestro antiguo servicio y lo volveremos a crear. Soy consciente de los procesos de actualización para ejecutar los servicios, aunque sentía que podía escribir un buen tocho sobre este tema, así que lo dejaría como está.

```
$ docker service rm carbon-cach
$ docker service create --name=carbon-cache --
publish=2003:2003/tcp --publish=2003:2003/udp
--publish=8000:8000/tcp swarm:5000/go-carbon-
arm
```

Configuración de Graphite

Llegados a este punto, estamos listos para la última parte del proyecto, que es otra imagen que necesitaremos recompilar. Sin embargo, gracias a los "fat manifests" o listas de manifiesto de Docker, al hacer referencia a "debian" obtendrás la imagen adecuada para tu arquitectura. Casi todas las compilaciones oficiales se hacen de esta forma, así que ya no es necesario buscar hasta dar con una imagen para la arquitectura ARM

Con suerte, en mi próximo artículo, exploraremos la configuración de tus propios "fat manifests" en tu registro privado. Esto es muy útil si tienes, como yo, arquitecturas mixtas como amd64 en tu swarm y quieres que cualquiera de tus servicios pueda implementarse en cualquiera de tus nodos. Así que empecemos por la imagen Grafanas. Elegí la imagen en https://goo.gl/pfpVef, ya que tiene todos los plugins que quería ya cargados, y eso ahorra un montón de trabajo para todo el mundo. Por supuesto, puedes hacerte con la imagen oficial y seguir el mismo proceso:

```
$ git clone
https://github.com/monitoringartist/grafana-
xxl.git
$ cd grafana-xxl
$ cp Dockerfile Dockerfile.arm
```

Necesitamos cambiar la línea 17 de:

```
$ curl https://s3-us-west-
2.amazonaws.com/grafana-
releases/release/grafana_${GRAFANA_VERSION}_am
d64.deb > /tmp/grafana.deb &&
Por los siguiente:
$ curl https://github.com/fg2it/grafana-on-
raspberry/releases/download/v${GRAFANA_VERSION
}/grafana_${GRAFANA_VERSION}_armhf.deb >
/tmp/grafana.deb && \
```

Despues, la línea 20 debe cambiar de:

```
$ curl -L
https://github.com/tianon/gosu/releases/downlo
```

```
ad/1.10/gosu-amd64 > /usr/sbin/gosu && \
```

A esto:

```
$ curl -L
https://github.com/tianon/gosu/releases/downlo
ad/1.10/gosu-armhf > /usr/sbin/gosu && \
```

Una vez hecho esto, estamos listos para recompilar, mover e implementar nuestro servicio. Usaremos nuevamente esos comandos cada vez más útiles y familiares:

```
$ docker build -t grafana-xxl-arm -f
Dockerfile.arm .
```

```
$ docker tag grafana-xxl-arm
swarm:5000/grafana-xxl-arm
$ docker push swarm:5000/grafana-xxl-arm
$ docker service create --name=grafana --
publish=3000:3000/tcp swarm:5000/grafana-xxl-
arm
```

El último paso es iniciar sesión en tu instancia grafana en http://swarm:3000/ con el nombre de usuario y la contraseña por defecto “admin” y “admin”. Una vez que hayas iniciado sesión, simplemente agrega http://swarm:8000/ como fuente de datos grafana predeterminada y estarás listo para usar Docker.

Juegos Android: Monument Valley, Hopscotch, Aqueducts

January 1, 2018 By Bruno Doiche Android, Juegos

No siempre nos desviemos de los extraños juegos indie para Android, pero durante las vacaciones play store nos ha dotado con grandes títulos, así que sin más preámbulos permite presentarte:

Monument Valley

Podría hablar largo y tendido de este juego, pero la mejor reseña nos llega de la propia descripción de Play Store: “En Monument Valley, manipularás estructuras imposibles y guiarás a una princesa sigilosa a través de un mundo increíblemente hermoso. Monument Valley es una exploración surrealista a través de una arquitectura fantástica y una geometría imposible. Guía a la princesa sigilosa a través de misteriosos monumentos, descubriendo caminos ocultos, desplegando ilusiones ópticas y burlando a la enigmática Crow People”.

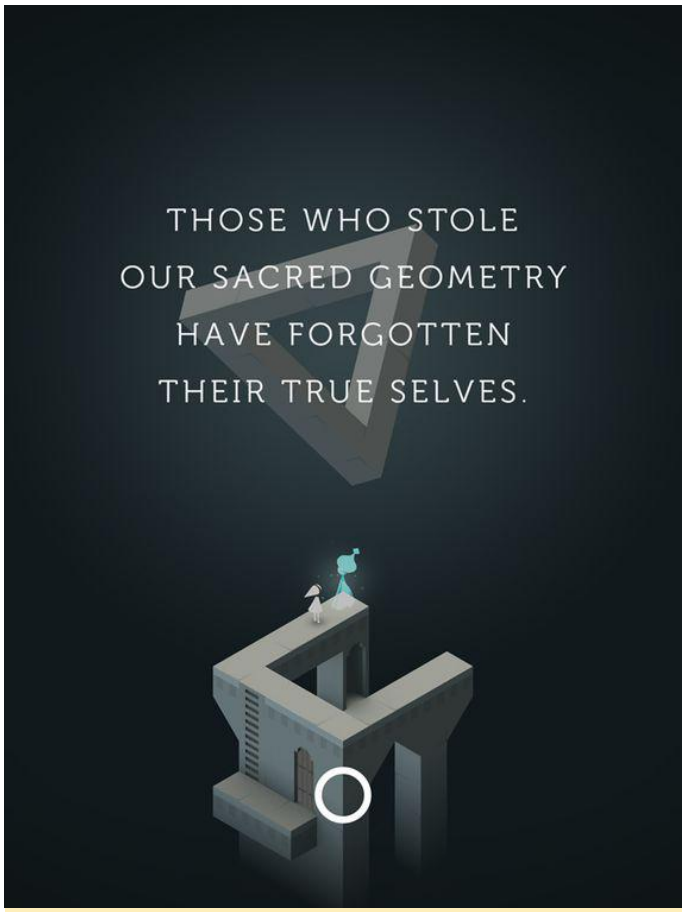


Figura 1: Monument Valley te atrapará desde el principio

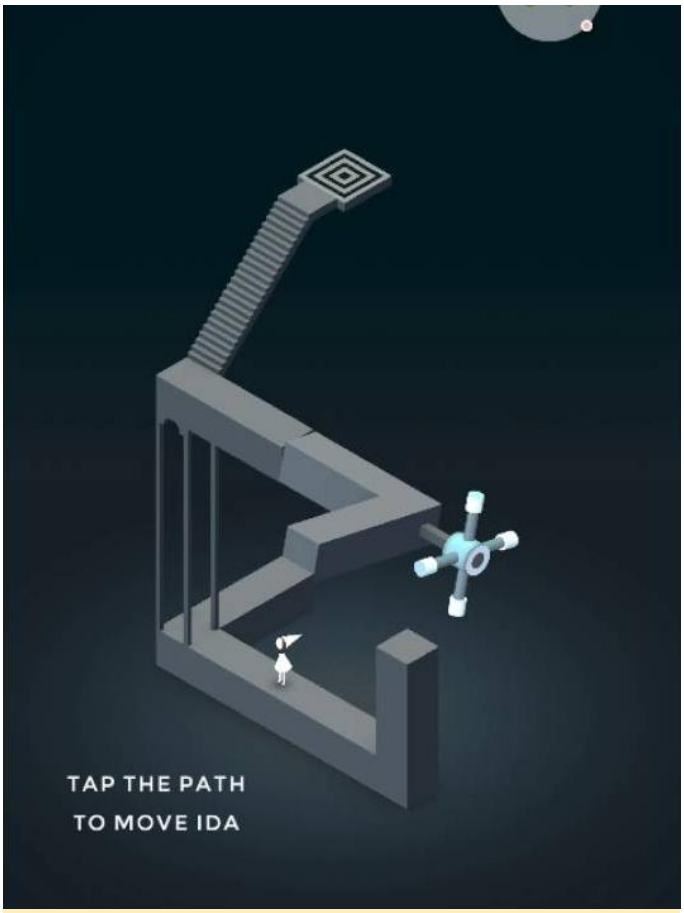


Figura 2 – Aunque lo he comentado en el artículo, usa tus auriculares para disfrutar al máximo

Este es un juego que elevará tu experiencia de juego como pocos lo harán, úsalo con tus mejores auriculares, no te arrepentirás de tener este juego en tu colección. Un juego que canaliza el espíritu de M.C. Escher, que te sumerge en una aventura que solo llegará su fin cuando termines este asombroso juego, y luego cogerás el ODROID con el que jugaste y lo enmarcarlas con una placa en la que se podrá leer la siguiente frase: "Jugué al Monument Valley con este hardware "(Compré otro ODROID después, por supuesto).

[Monument Valley en Play Store]

Hopscotch

Después de tanta grandiosidad con Monument Valley, ¿qué juego deberíamos perseguir en nuestra búsqueda de juegos? Todo parece tan trivial e inútil. Todo era mucho más fácil cuando estábamos en la escuela, donde todo era más simple y nos divertíamos mucho. ¿Podríamos capturar este sentimiento? ¡Da la casualidad de que sí! Simplemente conecta tu pantalla táctil a tu ODROID e instale Hopscotch. No tengo mucho más que decir al respecto: "Prepárate para que te quiten tu dispositivo, pero esta vez no será el profesor, sino cualquiera que quiera sentir el gustado de jugas a este gran juego".

Figura 3 – Después de jugar un montón a este juego, ¡Inténtalo a la inversa!

[Hopscotch en Play Store]

Aqueducts

Figura 4 – Aqueducts es muy divertido siendo un concepto tan simple.

Y para finalizar esta edición de Juegos Android, otro juego de puzles al estilo fontanería: Aqueducts. Este juego te enfrenta al trabajo de telequinesia más potente que existe mientras mueves gigantescas piezas que parecen ser colocadas por algún tipo de repartidor enfadado, o se trata de un proyecto de un ingeniero con muy poca financiación. Bromas aparte, estás a punto de enfrentarte a un juego que está muy bien diseñado y que convierte un juego muy común en una experiencia extremadamente divertida. Pruébalo, disfrutarás un buen rato.

[Aqueducts en Play Store]

Android TV ODROID-C2 con Amazon Prime Video y Netflix

January 1, 2018 By @goldpizza44 Android, ODROID-C2, Tutoriales

He estado usando un ODROID-C2 con LibreELEC durante bastante tiempo, pero me sentía muy decepcionado por la falta de compatibilidad con Amazon Prime Video y Netflix. También he estado usando un teclado/ratón inalámbrico para controlarlo, lo cual me permitía censurar a su compañero, ya que deseaba tener único un mando a distancia de TV para controlar tanto el televisor (encendido/volumen) como el ODROID-C2.

Aquí tienes el procedimiento que seguí. Doy por hecho que sabes moverte por Android, a la hora de buscar aplicaciones y realizar configuraciones, necesitarás usar Linux a través de Android Terminal Emulator.

Este procedimiento no consiste en instalar sólo Kodi, sino también Youtube TV, Amazon Prime Video, Netflix y algunas aplicaciones de Canales específicas. También te ayudara a instalar aplicaciones desde Google Playstore. Aunque Netflix no se instalaría desde Playstore, se puede descargar el APK e instalarlo. La configuración del mando a distancia lleva algo más tiempo, ya que no logre encontrar un solo tutorial en línea que detallase todo el proceso. Esperemos que este artículo ayude a otros con este tema.

El primer paso es instalar Android en una tarjeta Flash. Android para C2 se puede descargar desde **https://goo.gl/cuLqSU**. Cuando escribí este artículo, la v3.5 era la última versión, y esa es lo que yo usé. Descarga la imagen, descomprímela e instálala en la tarjeta flash con Etcher, que está disponible para muchos sistemas operativos, o win32diskimager que se ejecuta en Microsoft Windows, o usa la utilidad dd de Linux. Puede encontrar más información en **https://goo.gl/RPyiwr**.

Instala la tarjeta flash en el ODROID-C2 que tengas conectado a un televisor a través de HDMI junto con un teclado/ratón USB y enciéndelo. Tarda unos

minutos, pero al final acabarás teniendo un nuevo y brillante sistema Android, y el ratón debería permitirle navegar por el sistema.

El primer paso después de arrancar Android es ajustar el overscan en la pantalla. Observé en mi televisor que faltaban todos los bordes. No podía ver la barra de notificaciones en la parte superior y los botones de la parte inferior estaban cortados en su gran mayoría. Esto se soluciona fácilmente usando la aplicación "ODROID Utility". A través de esta utilidad puede ajustar la resolución (el "autodetect" por defecto también me funciono), utilicé las flechas para ajustar el overscan y apagué el LED azul, que parpadea y distrae. Tras ajustar la configuración con esta aplicación, debes hacer clic en "Apply and Reboot", lo cual reiniciará el sistema.

El siguiente paso es instalar Google Apps para tener disponible la tienda Google Play. Utilizando el navegador por defecto, el APK de Google Apps se puede descargar desde **http://opengapps.org/**. En esta página yo seleccioné lo siguiente:

- Platform: ARM
- Android: 6.0
- Variant: pico

Pico es la compilación mínima. También puedes intentar instalar nano y micro. Aunque no instalé el Calendario y otras aplicaciones en mi sistema, creo que funcionará.

Al pinchar en el botón de Descargar, se bajará un archivo ZIP a la carpeta de descargas. Este archivo ZIP debe tratarse como una actualización de Android y por lo tanto, se carga utilizando la misma aplicación Odroid Utility que se usó para actualizar Overscan y el LED azul. Ejecuta la aplicación Odroid Utility y haga clic en la esquina superior derecha (tres puntos). El menú que aparece tendrá la opción "Package install from storage" en el cual debes hacer clic. En la siguiente página, selecciona "File Manager" y navega hasta la carpeta de descargas donde debe seleccionar el archivo ZIP open_gapps. Se te pedirá que continúes, tras lo cual el odroid se reiniciará y se instalará Google Apps.

Una vez completada la instalación, puedes abrir la aplicación Google Play Store e instalar las siguientes aplicaciones:

- Amazon Prime Video
- Kodi
- Chrome Browser
- Pluto TV
- Un cliente VPN (si quieres ... yo uso OpenVPN)

Aunque hay numerosas aplicaciones de video, no todas son aptas para TV. Instálalas una por una y pruébalas para asegurarse de que funcionan como esperas. Puedes desinstalar las aplicaciones que no funcionan correctamente.

Por defecto, la configuración de Android "Settings -> Security -> Untrusted Sources" está fijada en "Yes". Esto es necesario para instalar Netflix. Netflix era la única aplicación que no está disponible en Google Play Store. Sin embargo, Netflix tiene una página de ayuda con un enlace a su versión v4.16 del APK en **https://goo.gl/22XXZi**.

Descargué esta APK y la instalé con FileManager. Funciona bien con el mando a distancia. Hay versiones más recientes de la APK de Netflix disponibles en **https://goo.gl/tkDbkz**. Sin embargo, cuando las descargué, observé que no funcionaba muy bien con el mando a distancia. No tengo muy claro por qué.

La configuración de las aplicaciones es la misma en todas las plataformas. La instalación de mi Kodi se comunica con un back-end de MythTV en otro servidor que hace todas las grabaciones de LiveTV y administra mi colección de películas. Encontrar el complemento MythTV PVR fue todo un desafío en Krypton. Está en Addons-> My Addons, pero está desactivado.

Mando a distancia

Utilizo un ODROID-C2 TV con algunos televisores antiguos que heredé, cuyos mandos a distancia originales se perdieron hace mucho tiempo. Sin embargo, dispongo de algunos mandos a distancia Dish Network 3.0 IR PVR. Estos se pueden conseguir en Ebay por menos de10$. En mi opinión, estos son

mandos a distancia bastante resistentes, su tacto es muy bueno y cuentas con suficientes botones para poder hacer lo que necesitas. También son "programables", ya que vienen con una lista de TVs y otros códigos de dispositivo permitiendo emular los mandos a distancia del otro fabricante.

Fue bastante fácil encontrar el código para controlar mi viejo televisor. Encender, subir volumen, bajar volumen y silenciar es todo lo que verdaderamente necesitaba. Deseaba que "Input select" funcionase para poder cambiar los puertos HDMI, pero nada de lo que probé hizo que ese botón funcionara. Afortunadamente, el ODROID-C2 es el único dispositivo de entrada que tengo, así que no es necesario cambiarlo. Si alguna vez coloco un segundo dispositivo HDMI, posiblemente será necesario volver a investigar hasta encontrar una solución viable.

El primer origen de mi frustración fue encontrar un código de dispositivo que activase todos los botones con el protocolo correcto. El ODROID-C2 utiliza el chip Amlogic S905 que contiene una interfaz para el receptor de infrarrojos. La instalación del sistema operativo Android de Hardkernel contiene el driver "amremote" integrado en el kernel (es decir, no se carga como un módulo). Por lo que el driver "amremote" solo reconoce el protocolo de infrarrojos NEC. Los códigos IR en otros protocolos (R5 / R6 o Sony) simplemente son ignorados por el driver "amremote". Si desea saber más sobre Consumer IR, visita **https://goo.gl/WLgtv9**.

Dotado con una lista de unos cuantos cientos de códigos, busqué un código de dispositivo de mando a distancia que enviara códigos NEC para todos los botones. Me llevé un chasco cuando descubrí que muchos dispositivos compatibles con el mando a distancia Dish Network solo enviaba códigos para un conjunto limitado de botones. Me costó bastante encontrar uno que enviase códigos para los 5 botones de navegación (arriba, abajo, izquierda, derecha y centro). Muchos códigos sólo facilitarían 3 opciones (arriba/abajo/centro o izquierda/derecha/centro), o 4 o 5 de las opciones. Finalmente encontré un reproductor de DVD Memorex (código 709) que ofrecía las 5 direcciones de navegación y todos los botones numéricos del mando a distancia. No permite enviar los códigos '*', '#', Volumen o Mute. Los códigos de Volumen y Mute son relegados al código de TV y por lo tanto, solo puedo controlar el Volumen de TV con los botones, no con el Volumen de Android.

Puesto que las "teclas numéricas" son en su gran mayoría inútiles para un TV (a excepción de los números de los canal), las reasigné en el remote.conf para realizar operaciones como son Home en Android, Subir volumen/Bajar volumen/Mute en Android y Avance Rápido/Volver atrás.

Para entender cómo una señal de IR entrante llega a la aplicación correctamente, debes tener en cuenta que hay 3 señales independientes involucradas:

- el código IR enviado por el mando a distancia
- el KEYCODE de Linux, y
- el código ACTION de Android

El truco está en asignar el código IR entrante al código ACTION de Android correcto a través de un KEYCODE de Linux, y esta conversión se realiza mediante dos archivos independientes en Android:

```
/system/etc/remote.conf -- maps IR code to
Linux KEYCODE
/system/usr/keylayout/Vendor_0001_Product_0001
.kl -- maps Linux KEYCODE to Android ACTION
```

En mi caso, ambos archivos requirieron tratamiento, aunque procure limitar los cambios a la distribución de las teclas. Para modificar estos archivos, utilicé la aplicación Android Terminal Emulator, que me proporciona un intérprete de comandos "bash" y el editor "vi". Si no sabe usar el editor "vi", puede copiar los archivos en /storage, usar la aplicación FileManager que está instalada y editarlos usando esta App, y luego copiar de nuevo a la ubicación /system. Más adelante me valgo de "vi" y de varios comandos de Linux para hacer este trabajo.

La primera tarea es cambiar el sistema de archivos de /sistem de ReadOnly a ReadWrite para que podamos actualizar los archivos. Abre la aplicación Android Terminal Emulator y escribe:

```
$ su -
# mount -o remount,rw /system
```

El primer comando (su -) te otorga privilegios de SuperUser. La primera vez que la uses aparecerá una ventana emergente que te pregunta si esta aplicación (Android Terminal Emulator) siempre debe tener este privilegio. Yo respondí "Yes" y la hice permanente. Se supone que todos los comandos que se ejecuten se cargarán con privilegios de SuperUser. Si abandonas Android Terminal Emulator y regresas de nuevo, es posible que necesite ejecutar nuevamente el comando "su -".

El segundo comando cambiará el sistema de archivos de /system de ReadOnly a ReadWrite. A continuación, debemos editar /system/etc/remote.conf para activar la depuración. La depuración nos permitirá determinar qué códigos estamos recibiendo. Cambia "debug_enable" de "0" a "1" con vi y actívalo con remotecfg:

```
# vi /system/etc/remote.conf
# remotecfg /system/etc/remote.conf
```

Remotecfg lee el archivo remote.conf, el cual analizará los contenidos y luego enviará la información al software amremote en el kernel Linux. Esto normalmente se hace una vez en el arranque tal y como se especifica en /system/init.odroidc2.rc. Esto supone una gran ventaja, ya que podemos hacer cambios y luego activarlos de inmediato.

Con debug_enable fijado en "1", cualquier mando a distancia que envíe un protocolo NEC será detectado, y el software amremote registrará los errores en el registro log del sistema. Usaremos "dmesg" para ver el registro log del sistema. Prueba el cambio usando estos comandos:

```
$ dmesg -c > /dev/null # clear previous log
contents
$ while sleep 1; do dmesg; dmesg -c >
/dev/null; done
```

Con el modo "while sleep" activado, al presionar un botón en el mando a distancia debería aparecer algo como esto:

```
[98086.788285@0] remote: Wrong custom code is
0x7c83ff00
```

Los últimos cuatro dígitos del número que aparece al final del mensaje de registro log nos dicen qué tipo de mando a distancia es este (0xff00 en mi caso). El remote.conf por defecto de Android Odroid busca el código 0x4db2. No estoy seguro de cuál es la probabilidad de que cuentes con un mando a distancia con ese código si no estás utilizando el mando a distancia disponible en HardKernel (**https://goo.gl/yVLVLC**). Si no tiene suerte, y el mando a distancia que estás utilizando transmite 0x4db2, entonces verá algo más en dmesg. Si no ves nada en dmesg, entonces es que no está usando un mando a distancia que transmite el protocolo NEC y debe buscar otro mando a distancia (o código de dispositivo).

En mi caso, aquí es donde empecé a buscar los códigos del dispositivo Dish Network 3.0 IR investigado el dispositivo adecuado. Configuré el código del dispositivo en el mando a distancia y presioné los botones para ver si obtenía las respuestas en dmesg. Probé muchos códigos hasta que estuve lo suficientemente cerca de mis necesidades con el código 709 de DVD Memorex, que transmite el tipo de mando a distancia 0xff00.

El extenso número HEX 0x7c83ff00 es en realidad 2 trozos de información. Descomponiéndolos en bytes: 7c 83 ff 00, deberías observar que los primeros 2 bytes son complementarios entre sí (es decir, 01111100 -> 10000011 – ceros y unos invertidos). De manera similar, los 3º / 4º bytes son complementarios en muchos casos (pero no en todos). La "verdadera" información está en los bytes 2 y 3/4 (0x83 es el botón, 0xff00 es el tipo de mando a distancia).

En remote.conf, ahora es el momento de configurar el tipo de mando a distancia que estás utilizando en remote.conf configurando la entrada para "factory_code" y reemplazando el XXXX con el código de 4 dígitos que aparece arriba (ajuste el mío en 0xff000001):

```
$ vi remote.conf
$ remotecfg /system/etc/remote.conf
```

```
$ while sleep 1;do dmesg;dmesg -c >
/dev/null;done
```

Tras introducir estos comandos, deberías presionar nuevamente los botones y ocurrirá una de estas dos cosas: ver un error que indique que el botón del mando a distancia no está asignado a nada, o ver información sobre la asignación del botón.

```
[101131.973324@0] remote: scancode is
0x00c5,invalid key is 0x0000.
or
[101214.803355@0] remote: press ircode = 0xc5
[101214.903456@0] remote: scancode =
0x74,maptable = 0,code:0x3ac5ff00
[101214.903492@0]
[101214.993555@0] remote: release ircode 0xc5
[101214.997312@0] remote: scancode =
0x74,maptable = 0,code:0x00000000
```

La primera sucede porque el botón 0xc5 no está en remote.cfg. La segunda ocurre cuando el botón se encuentra en remote.cfg. Si el mando a distancia que estás utilizando envía códigos similares a los del mando a distancia de Hardkernel, verás el segundo tipo de mensaje.

Aquí es donde empieza la diversión. Debes presionar cada botón y ver qué código envía, y tenerlo en cuenta. Luego debes averiguar qué quieres que haga y encontrar la acción Android en el archivo Vendor_0001_Product_0001.kl que corresponda a la acción que deseas que haga el botón. Finalmente, debes conseguir el KEYCODE de Linux desde Vendor_0001_Product_0001.kl que se usará para unirlo todo.

Estas son las acciones de ANDROID que he usado:

- POWER — key 116
- HOME — key 102
- BACK — key 15
- MENU — key 139
- DPAD_CENTER — key 97
- DPAD_LEFT 1 — key 105
- DPAD_RIGHT — key 106
- DPAD_UP — key 103
- DPAD_DOWN — key 108
- VOLUME_UP — key 115
- VOLUME_DOWN — key 114
- VOLUME_MUTE — key 113
- MEDIA_REWIND — key 121
- MEDIA_FAST_FORWARD — key 120
- APP_SWITCH — NO KEY available!

¡El último, “APP_SWITCH” no está en Vendor_0001_Product_0001.kl! Me llevó 2 horas solucionar el tema. Me apropié de una tecla (158, que anteriormente era BACK) actualizando Vendor_0001_Product_0001.kl con vi y cambiando “BACK” a “APP_SWITCH” en la línea correspondiente.

Ahora pégalo todo junto actualizando remote.conf en la sección key_begin/key_end y posiblemente en la sección repeat_key_begin/repeat_key_end. No suelo valerme de las repeticiones de teclas, de modo que mi sección repeat_key_begin/repeat_key_end está vacía. Tampoco suelo apoyar en la sección mouse_begin/mouse_end.

Mi archivo remote.conf resultante es así:

```
work_mode = 0
repeat_enable = 1
repeat_delay = 40
repeat_period = 39
release_delay = 121
debug_enable = 0
factory_code = 0xff000001
left_key_scancode = 0x88
right_key_scancode = 0xc8
up_key_scancode = 0xc9
down_key_scancode = 0xd7
ok_key_scancode = 0x8b
mouse_begin
mouse_end
key_begin
0xc5 116 # Power
0x8b 97 # Center
0xc9 103 # Up
0xd7 108 # down
0x88 105 # Left
0xc8 106 # Right
0x93 15  # cancel
0x93 15  # Info
0x81 114   # 1 -- becomes VOLUME DOWN
0x83 113   # 2 -- becomes MUTE
0xc1 115   # 3 -- becomes VOLUME UP
0x82 121   # 4 -- becomes REWIND
0x80 139   # 5 -- becomes MENU
```

```
0xc0 120   # 6 -- becomes FF
0x8d 8     # 7
0x8f 158   # 8 -- APP SWITCH
0xcd 10  # 9
0x8c 102  # 0 -- becomes HOME
key_end
repeat_key_begin
repeat_key_end
```

Ten en cuenta que también apagué “debug_enable” fijándolo en cero. Después de actualizar remote.conf nuevamente, escribe el siguiente comando:

```
$ remotecfg /system/etc/remote.conf
```

Ahora es el momento de probar el mando a distancia. Deberías probarlo con cada aplicación, ya que ésta puede reaccionar de forma diferente a un código en particular o ignorarlo por completo. Todavía no estoy seguro de tener todos los botones configurados correctamente, pero el sistema es funcional y amigable.

Realicé una copia de seguridad de Vendor_0001_Product_0001.kl y remote.conf copiándolos en /storage/emulated/0/Download y me aseguré de que estuvieran guardados en el sistema de archivos /system. Si actualizas Android, es posible que necesites restaurar estos archivos o tu mando a distancia no funcionará. Espero que todo esto le sea útil al menos a una persona. Probablemente necesitare sacarlo a la luz en el futuro para recordar lo que hice en su momento. Para comentarios, preguntas y sugerencias, visita el post original del foro en **https://goo.gl/6sc8GU**.

Ambilight en ODROID-C2 Usando LibreElec: Adaptando Ambilight al ODROID-C2

January 1, 2018 By @rokapet ODROID-C2, Mecaniqueo

En este artículo, me gustaría compartir cómo llegué a montar un sistema Ambilight funcional con un ODROID-C2 y LibreElec. Previamente localicé otras guías que se centraban principalmente en la Raspberry Pi, así que tuve que recopilar información usando Google y recurrir al método ensayo y error.

El diseño de alto nivel utiliza Hyperion instalado como complemento para Kodi, desde el repositorio de LibreElec instalar HyperCon no es posible debido a que la CPU es desconocida, y un Arduino para controlar los LED usando la librería FastLED. Arduino está conectado al ODROID-C2 a través de un cable USB (el Arduino tiene un conversor USB a serie). Mi TV es FullHD, así que no necesite recurrir a la magia (es decir, hacer overclock de RAM y cosas así) para el 4K.

Utilicé las siguientes guías como base para el montaje y la configuración. Pero como estas guías están hechas para usarse con la Raspberry Pi, necesitaba adaptarlas al ODROID-C2.

Raspberry Pi 3 Mediacenter + Hyperion Ambilight + NO soldering: https://goo.gl/q8q6PK Amblight project/guide – Hyperion – WS2801/ WS2812B / APA102: https://goo.gl/CEgT1U

Resulta que mi configuración está afectada por un error bastante molesto, el cual se describe aquí: https://goo.gl/HUwa2k

La reproducción de video empieza a ralentizarse transcurrido un periodo de tiempo arbitrario (entre 30-70 minutos en mi caso) mientras que el audio continúa reproduciéndose a una velocidad adecuada, lo que provoca que el audio y el video no estén sincronizados. Después de 30-60 segundos se produce una pequeña pausa y salto, el audio y el video vuelven a sincronizarse, hasta que vuelva a suceder a los 30-70 minutos.

En el foro de LibreElec aún no está resuelto este tema, ya que el problema está resultando bastante complicado de solucionar. Si alguien tiene un sistema en funcionamiento (ODROID-C2 con LE8/Kodi 17 con Hyperion) que no se vea afectado por este error, ¡Que no dude en contribuir con el archivo de configuración de Hyperion! Tengo pensado retocar la configuración de Hyperion para ver si se puede hacer algo, pero esto me puede llevar bastante tiempo

Hardware

- ODROID-C2 mas carcasa y fuente de alimentación de Hardkernel, que me costó alrededor de 72€ cuando lo comprés en https://www.pollin.de el pasado año
- Tarjeta MicroSD Kingston 16GB Clase 10, que cuesta alrededor de 8€ en eBay
- Clon chino de la placa Arduino Uno R3 más carcasa y cable USB, que cuesta alrededor de 8,50€ en eBay
- Tira LED RGB WS2812B de 3 pin, 30 LEDs por metro, 5m de rollo, que costó alrededor de 13.50€ en eBay
- Fuente de alimentación 5V 10A. Tenía una por ahí, así que no me supuso ningún coste, pero puedes encontrar una por unos 10-20€ en eBay o Aliexpress
- Clavija de alimentación DIN, que cuesta alrededor de 1,50€ en eBay
- Cables conector JST macho y hembra de 3 pines, que cuestan aproximadamente 1,50€ en eBay 5 piezas
- Conectores de 3 pines de 10 mm para el conjunto WS2812, que cuestan aproximadamente 2€ en eBay 5 piezas
- Cables jumper macho Dupont, que cuestan aproximadamente 1€ en eBay 40 piezas
- Tubos termo-contraíbles, que cuestan aproximadamente 1€ en eBay 70 piezas
- Resistencia de 500 ohmios o resistencias de 2x1K ohmmios cableadas en paralelo
- Cable HDMI
- Soldador eléctrico para soldar los conectores y los cables

Software

- Imagen LibreElec 8.2.1 MR para ODROID-C2 descargada desde https://libreelec.tv
- Complemento Hyperion para Kodi del repositorio de LibreElec
- Herramienta HyperCon para Hyperion (https://goo.gl/7F5fDc)
- IDE Arduino (https://goo.gl/wqP28G)
- Librería FastLED para Arduino (descargar ZIP – – https://github.com/FastLED/FastLED)
- Esquema de control (script) para Arduino (https://pastebin.com/2L9ZBhYe)

```
#include "FastLED.h"

// How many leds in your strip?
#define NUM_LEDS 92

// For led chips like Neopixels, which have a
data line, ground, and power, you just
// need to define DATA_PIN. For led chipsets
that are SPI based (four wires - data, clock,
// ground, and power), like the LPD8806 define
both DATA_PIN and CLOCK_PIN
#define DATA_PIN 12
//#define CLOCK_PIN 13

#define COLOR_ORDER GRB

// Adalight sends a "Magic Word" (defined in
/etc/boblight.conf) before sending the pixel
data
uint8_t prefix[] = {'A', 'd', 'a'}, hi, lo,
chk, i;

// Baudrate, higher rate allows faster refresh
rate and more LEDs (defined in
/etc/boblight.conf)
#define serialRate 115200

// Define the array of leds
CRGB leds[NUM_LEDS];

void setup() {
 // Uncomment/edit one of the following lines
for your leds arrangement.
 // FastLED.addLeds<TM1803, DATA_PIN, RGB>
(leds, NUM_LEDS);
 // FastLED.addLeds<TM1804, DATA_PIN, RGB>
(leds, NUM_LEDS);
 // FastLED.addLeds<TM1809, DATA_PIN, RGB>
(leds, NUM_LEDS);
 // FastLED.addLeds<WS2811, DATA_PIN, RGB>
(leds, NUM_LEDS);
 // FastLED.addLeds<WS2812, DATA_PIN, RGB>
```

```
(leds, NUM_LEDS);
 FastLED.addLeds<WS2812B, DATA_PIN, RGB>(leds,
NUM_LEDS);
 // FastLED.addLeds<NEOPIXEL, DATA_PIN>(leds,
NUM_LEDS);
 // FastLED.addLeds<UCS1903, DATA_PIN, RGB>
(leds, NUM_LEDS);
 // FastLED.addLeds<UCS1903B, DATA_PIN, RGB>
(leds, NUM_LEDS);
 // FastLED.addLeds<GW6205, DATA_PIN, RGB>
(leds, NUM_LEDS);
 // FastLED.addLeds<GW6205_400, DATA_PIN, RGB>
(leds, NUM_LEDS);

 // FastLED.addLeds<WS2801, RGB>(leds,
NUM_LEDS);
 // FastLED.addLeds<SM16716, RGB>(leds,
NUM_LEDS);
 // FastLED.addLeds<LPD8806, RGB>(leds,
NUM_LEDS);

 // FastLED.addLeds<WS2801, DATA_PIN,
CLOCK_PIN, RGB>(leds, NUM_LEDS);
 // FastLED.addLeds<SM16716, DATA_PIN,
CLOCK_PIN, RGB>(leds, NUM_LEDS);
 // FastLED.addLeds<LPD8806, DATA_PIN,
CLOCK_PIN, RGB>(leds, NUM_LEDS);

 // initial RGB flash
 LEDS.showColor(CRGB(255, 0, 0));
 delay(500);
 LEDS.showColor(CRGB(0, 255, 0));
 delay(500);
 LEDS.showColor(CRGB(0, 0, 255));
 delay(500);
 LEDS.showColor(CRGB(0, 0, 0));

 Serial.begin(serialRate);
 Serial.print("Ada
"); // Send "Magic Word" string to host
}

void loop() {
 // wait for first byte of Magic Word
 for(i = 0; i < sizeof prefix; ++i) {
 waitLoop: while (!Serial.available()) ;;
 // Check next byte in Magic Word
 if(prefix[i] == Serial.read()) continue;
 // otherwise, start over
 i = 0;
 goto waitLoop;
}

// Hi, Lo, Checksum

while (!Serial.available()) ;;
 hi=Serial.read();
 while (!Serial.available()) ;;
 lo=Serial.read();
 while (!Serial.available()) ;;
 chk=Serial.read();

// if checksum does not match go back to wait
 if (chk != (hi ^ lo ^ 0x55))
 {
  i=0;
  goto waitLoop;
 }

memset(leds, 0, NUM_LEDS * sizeof(struct
CRGB));
// read the transmission data and set LED
values
for (uint8_t i = 0; i < NUM_LEDS; i++) { byte
r, g, b; while(!Serial.available()); r =
Serial.read(); while(!Serial.available()); g =
Serial.read(); while(!Serial.available()); b =
Serial.read(); leds[i].r = r; leds[i].g = g;
leds[i].b = b; } // shows new values
FastLED.show(); }
```

Montaje de la tira LED

La tira LED que compré ya tenía un conector hembra JST de 3 pines integrado en el extremo de inicio, más dos cables adicionales (sin enchufe) para 5V y GND. Soldé los dos últimos cables a la toma de corriente DIN para poder alimentar los LED desde la fuente de alimentación. ¡Asegúrate de usar una clavija que sea compatible con tu fuente de alimentación!

Cogí un cable conector macho JST y soldé cables Dupont al extremo de los cables GND y DATA para poder conectarlos a los pines de la placa Arduino. El cable de 5V no se utiliza aquí, ya que no está conectado, de modo que se puede cortar y/o aislar. Soldé la resistencia de 500 ohmios entre el cable DATA y el cable Dupont para que las señales de control puedan atravesarlo. Todos los puntos de soldadura han sido asegurados con tubos termo-contraíbles.

Inicié HyperCon en mi ordenador y planifiqué la configuración y el diseño del LED en el televisor, asegurándome de medir las dimensiones de mi TV. Estudie otras opciones de configuración en el sitio de la Wiki de Hyperion y guardé la configuración LED en el archivo hyperion.config.json. Luego utilicé los números LED calculados para cortar la tira en piezas con el tamaño adecuado y utilicé los cables del conector de 3 pines para unirlos entre sí. Los cables del conector de 3 pines son necesarios para las esquinas del televisor, ya que la tira de LED por sí sola no se puede doblar para llegar a los 90 grados.

Ten en cuenta que estas tiras LED tienen una dirección. Solo el extremo inicial se puede usar para los controles de entrada y los distintos LED se direccionan uno después del otro. Busqué la marca “DI (Entrada de datos) en la tira, cerca de los LEDs. El otro extremo tiene marcado “DO (Data Out). Puedes usar Google para localizar las especificaciones WS2812b.

Mi experiencia me dice que los conectores de 3 pines no pueden sujetar la tira con demasiada fuerza, así que ten cuidado a la hora de montarla en la parte posterior de tu televisor. Si tiene experiencia con la soldadura de precisión, puede que sea mejor soldar los cables para conectar las partes de la tira en lugar de utilizar los conectores de 3 pines.

Instalé LibreElec en la tarjeta micro SD siguiendo las instrucciones de https://libreelec.tv, luego la monté el ODROID-C2 y configuré Kodi a mi gusto. Entre a Settings -> Services -> Control in Kodi interface, y habilité tanto “Allow remote control from applications on this system” como “Allow remote control from applications on other systems”. Instalé IDE Arduino en mi ordenador, luego extraje el archivo ZIP de la librería FastLed en una subcarpeta independiente bajo la carpeta libraries. Cuando inicié IDE Arduino después, la librería FastLED aparecía disponible en el menú Sketch -> Available libraries.

Conecté la placa Arduino a mi ordenador con el cable USB. Copié el esquema en el IDE Arduino, fijé el número de LED de la tira en la fila n. ° 4 (tendrá este número de HyperCon) y el pin utilizado para el control de LED en la fila n. ° 9. Luego cargué el esquema en la placa Arduino, que solo tardó unos segundos.

Montaje

Utilicé adhesivo en la parte posterior de la tira LED para pegarlo a la parte posterior de mi televisor. Necesite algo de adhesivo adicional en algunas zonas para que se adhiriera con firmeza. Desconecté la placa Arduino de mi PC y la conecté a uno de los puertos USB del ODROID-C2. Conecté el jumper del cable Dupont soldado al cable GND de la tira al pin GND y el jumper del cable Dupont soldado al cable DATA de la tira al pin 12 del Arduino. Finalmente, conecté la fuente de alimentación a la clavija de potencia soldada al final de la tira.

Configuración del software

LibreElec cargó el módulo CH341 automáticamente al conectar la placa Arduino a USB, y apareció el dispositivo /dev/ttyUSB0. Sin embargo, esto podría ser diferente en tu caso Instalé el complemento Hyperion bajo la interfaz de Kodi, desde el repositorio de LibreElec. Abrí el archivo hypercon.config.json previamente guardado en mi ordenador y lo edité en 3 lugares: device, frame-grabber y effects. La actualización de la carpeta de efectos es necesario porque es diferente cuando Hyperion es instalado como complemento, en lugar de instalarlo con HyperCon.

Ten en cuenta que en otras guías se menciona usar 100000, 200000, etc. para la tasa de baudios. En mi caso, esto derivo en un repetitivo mensaje de error “Unable to open RS232 device (IO Exception (25): Inappropriate ioctl for device” en el registro log de Hyperion. Cambiar a una velocidad de 115200 baudios hizo que el problema se solucionara. ¡Asegúrate de observar el orden del color invertido “GRB”!

```
// DEVICE CONFIGURATION
"device" :
 {
 "name" : "MyHyperionConfig",
 "type" : "adalight",
 "output" : "/dev/ttyUSB0",
 "rate" : 115200,
 "delayAfterConnect" : 0,
```

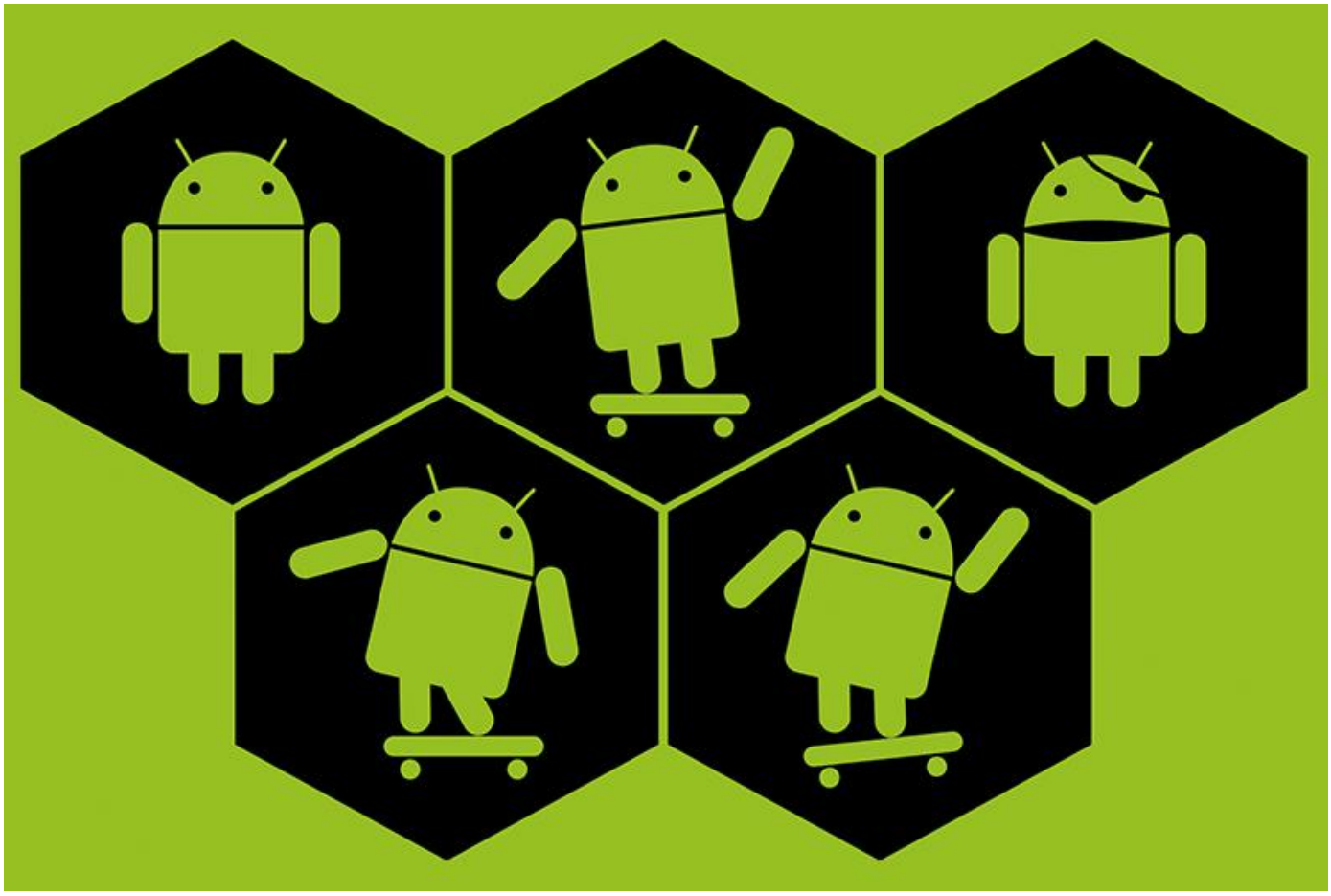
```
 "colorOrder" : "grb"
 },

"amlgrabber" :
 {
 "width" : 64,
 "height" : 64,
 "frequency_Hz" : 20.0,
 "priority" : 880
 },
 "framegrabber" :
 {
 "width" : 64,
 "height" : 64,
 "frequency_Hz" : 10.0,
 "priority" : 890
 },

"effects" :
 {
 "paths" :
 [
 "/storage/.kodi/addons/service.hyperion/effec
ts"
 ]
 },
```

Cargué mi archivo hyperion.config.json actualizado en la carpeta /storage/.kodi/userdata/addon_data/service.hyperion/ en LibreElec. Esta es la carpeta donde el complemento Hyperion busca el archivo de configuración. El recurso compartido UserData se puede utilizar para la carga si el servidor SMB está habilitado. Luego reinicié LibreElec y, fue entonces cuando el sistema Ambilight funcionó correctamente. Espero no haber olvidado nada y que a alguien le sea útil esta información. ¡Cualquier comentario o sugerencia para mejorar será bienvenida!

Para comentarios, preguntas y sugerencias, visita el post original del foro en https://forum.odroid.com/viewtopic.php?f=144&t=29334.

Divirtiéndonos con GPIO en Android

January 1, 2018 By Justin Lee Android, Mecaniqueo



El ODROID-C1/C1+, ODROID-C2 y ODROID-XU4 tienen pines GPIO (Entrada/Salida de Propósito General) que permiten controlar dispositivos externos a través de software. Para acceder correctamente al puerto GPIO, debes instalar la imagen Android Marshmallow versión 2.8 o superior en el ODROID-C2, la imagen Android KitKat versión 3.2 o superior en el ODROID-C1/C1+, y la imagen Android KitKat versión 6.0 o superior, o la versión Android Nougat versión 1.3 20171214 o superior en el ODROID-XU4.

Esta **WiKi** explica cómo hacer una app Android que permita acceder a los puertos GPIO. Necesitas instalar **Google Android Studio** en tu PC host. Añade en primer lugar NDK y las herramientas necesarias antes de empezar con los pasos que se describen a continuación. Nosotros probamos estos pasos en Android Studio 2.3 y NDK R14.

Ubuntu/Linux

Puedes encontrar la ubicación del SDK de Android bajo este menú (File → Settings → Appearance & Behavior → System Settings → Android SDK)

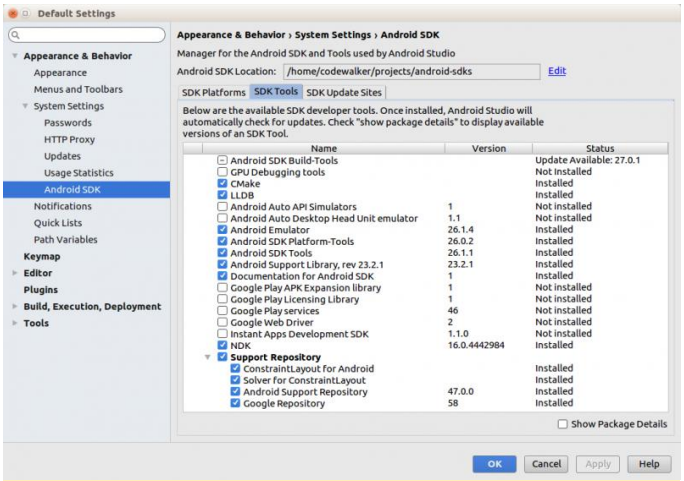


Figura 1 – Ubicación del SDK de Android

Tras editar el archivo bashrc, debes iniciar sesión nuevamente o escribir “source ~/.bashrc” en la línea de comando. A continuación, descarga la librería

NDK **WiringPi** y el código fuente de la aplicación desde **GitHub**.

ODROID-C1+/C2

```
$ sudo apt install git
$ git clone
https://github.com/codewalkerster/example-
wiringPi -b odroid-c
```

ODROID-XU4

```
$ sudo apt install git
$ git clone
https://github.com/codewalkerster/example-
wiringPi -b odroid-xu
```

Ejecuta Android Studio y abre el proyecto descargado.

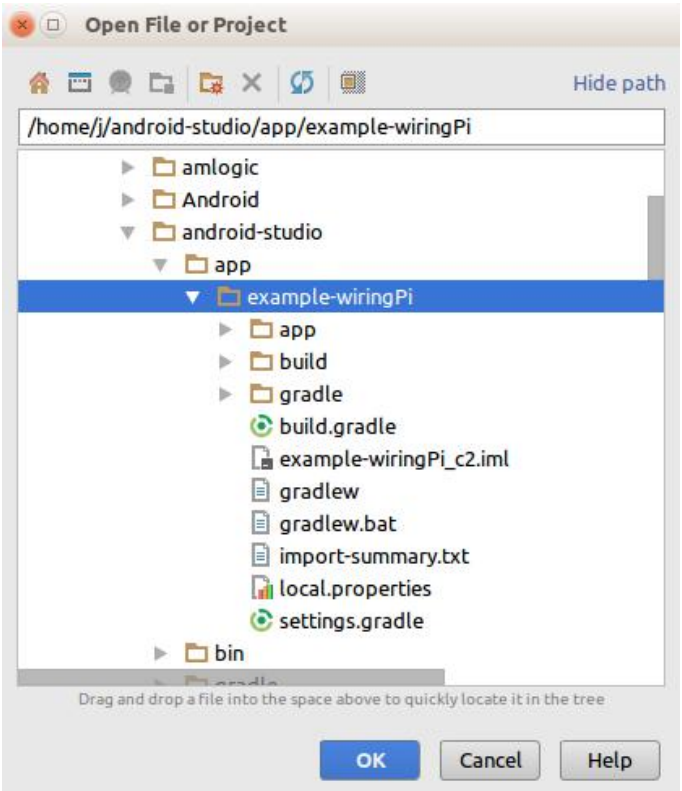


Figura 2 – Abriendo el proyecto wiringPi

Compilar el proyecto para crear un paquete apk

Selecciona Build -> Make Project desde menú. Verás un par de mensajes de error y deberás hacer clic en las siguientes opciones para completar el proceso de compilación, el cual generará un archivo APK para que lo ejecutes en tu ODROID:

- Instalar la(s) plataforma(s) que faltan y sincronizar el proyecto
- Instalar Build Tools 25.0.2 y sincronizar el proyecto

Windows

En Windows, configura la RUTA de entorno para que apunte a la ruta de la carpeta NDK y luego reinicia Windows.

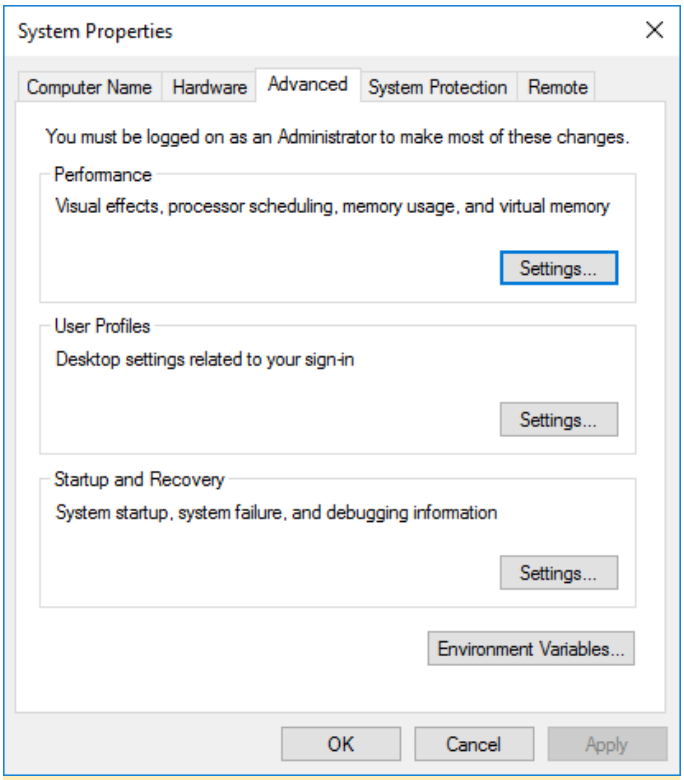


Figura 3: Configurando la ruta de entorno para que apunte a la ruta de la carpeta NDK

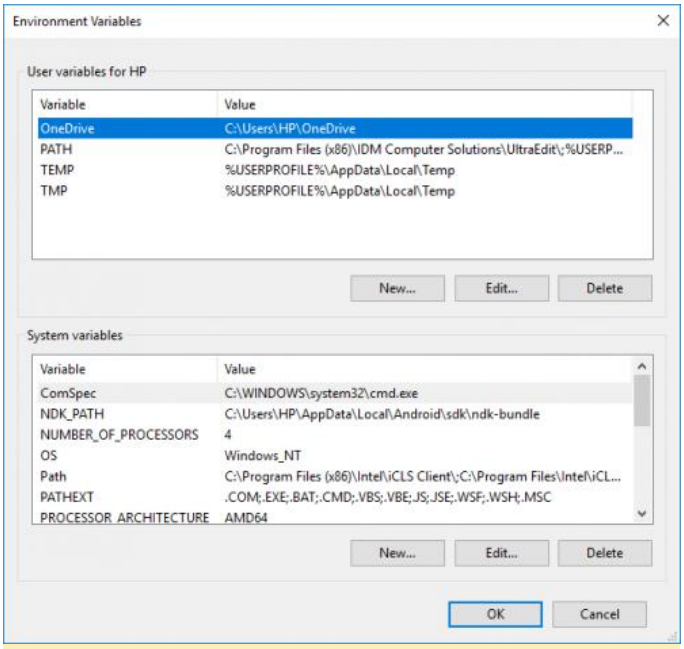


Figura 4 – Configurando la ruta de entorno para que apunte a la ruta de la carpeta NDK

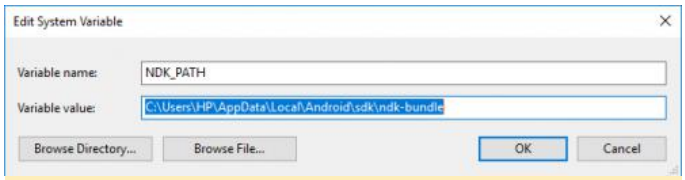


Figura 5: Configurando la ruta de entorno para que apunte a la ruta de la carpeta NDK

Después, instala el programa cliente Git desde https://git-for-windows.github.io/, y clona el proyecto wiringPi, tal y como se muestra en la Figura 6.

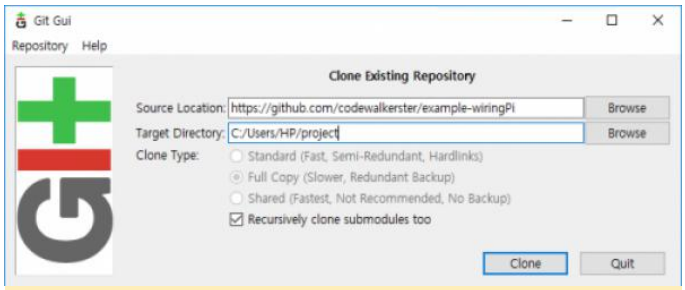


Figura 6 – Clonando el proyecto

El proyecto está disponible en https://github.com/codewalkerster/example-wiringPi. Selecciona origin/odroid-c o origin/odroid-xu.

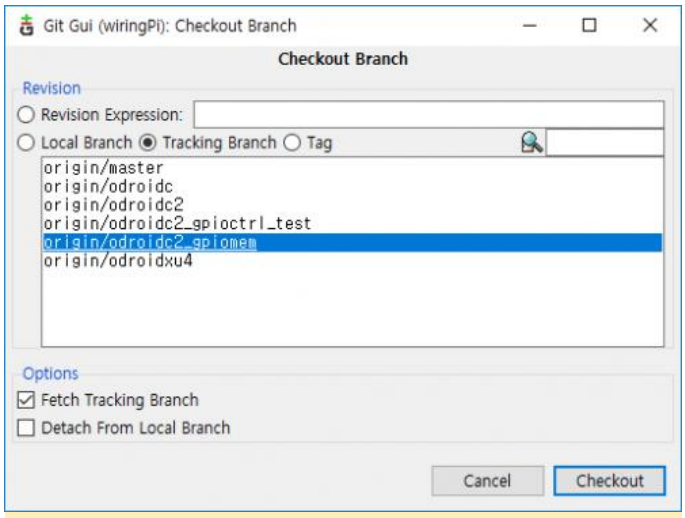


Figura 7 – Verificando la rama de GitHub

Luego, instala el NDK seleccionando Tools -> Android -> SDK Manager en el menú.

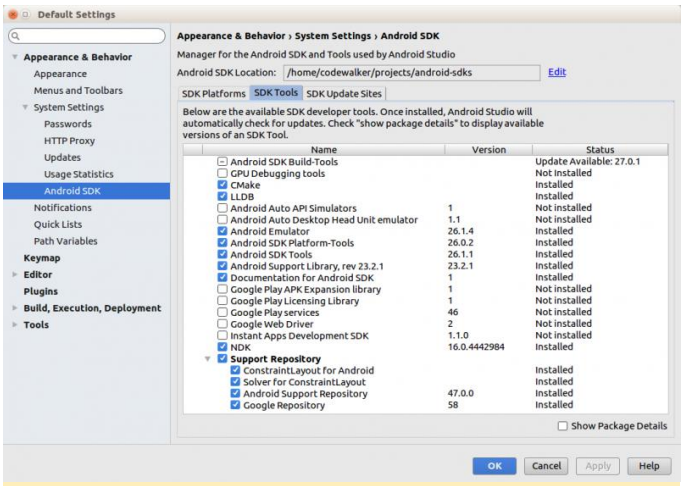


Figura 8 – Instalando el SDK

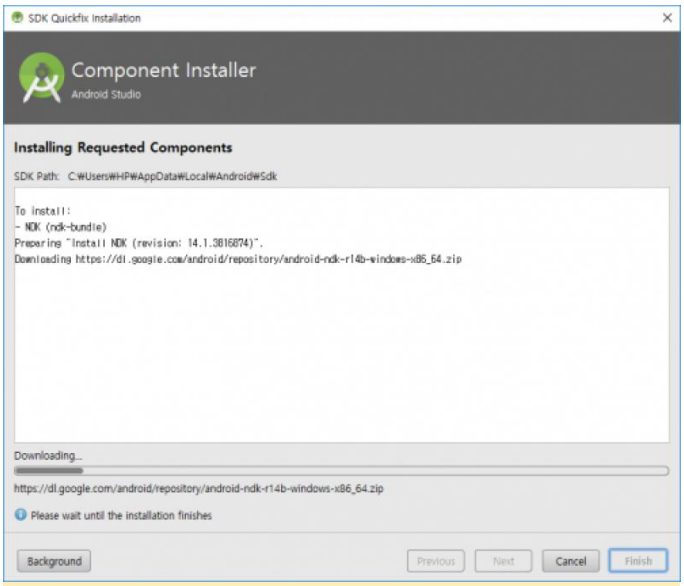


Figura 9 – Instalando el SDK

Características del proyecto de ejemplo

Puedes leer el valor del conversor de señal analógica a digital (ADC) y mostrar el nivel de voltaje con 19 LED en la salida de GPIO. Puedes ver un video de demostración en https://youtu.be/lGqyhvd3q9U y https://youtu.be/lGqyhvd3q9U.

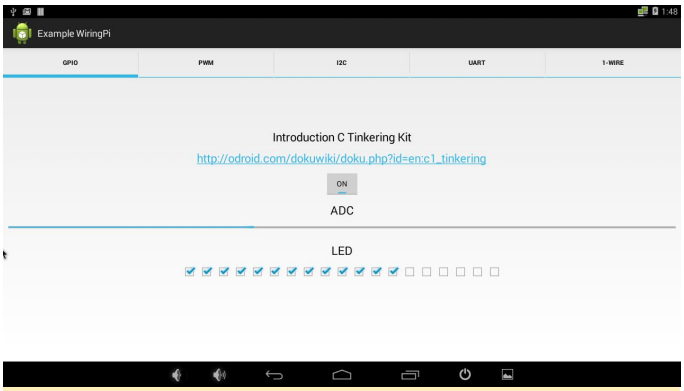


Figura 10 – Lectura del valor ADC en ODROID-C1+/C2

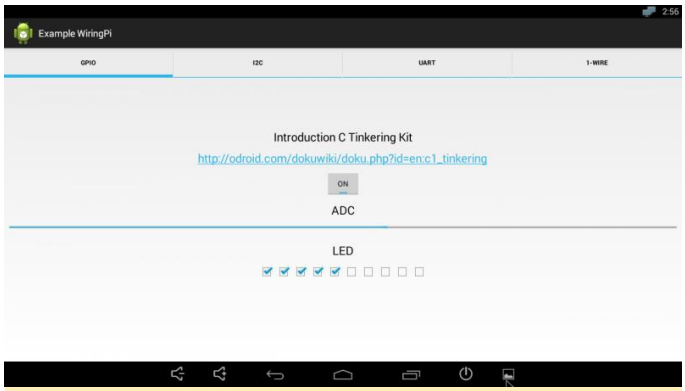


Figura 11 – Lectura del valor ADC en el ODROID-XU4

PWM

La Figura 12 muestra un ejemplo básico de control del PWM. Puede escoger el número de salidas PWM (1 o 2), así como controlar la frecuencia y el índice de trabajo.

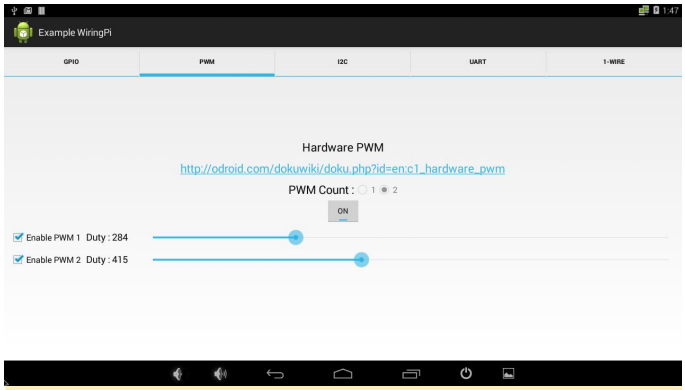


Figura 12 – Ejemplo básico de control PWM

Ejemplo de notificador de Gmail

Este es un proyecto muy divertido y útil que utiliza el puerto PWM. Cuando te encuentras visualizando videos o jugando a juegos, puedes pasar por alto la notificación de un correo electrónico o mensaje importante. El indicador es accionado por un servomotor que está conectado a un pin PWM en un puerto GPIO de 40 pines. El código puede descargarse desde **https://github.com/codewalkerster/GMailNotifier**. Se puede ver una demo del proyecto en **https://youtu.be/Vvq77w87RWQ**.

I2C

La Figura 15 muestra un ejemplo de código para acceder a nuestra placa meteorológica para medir la temperatura, la humedad, la presión atmosférica, la altitud y la intensidad de luz visible/invisible a través de la interfaz I2C.

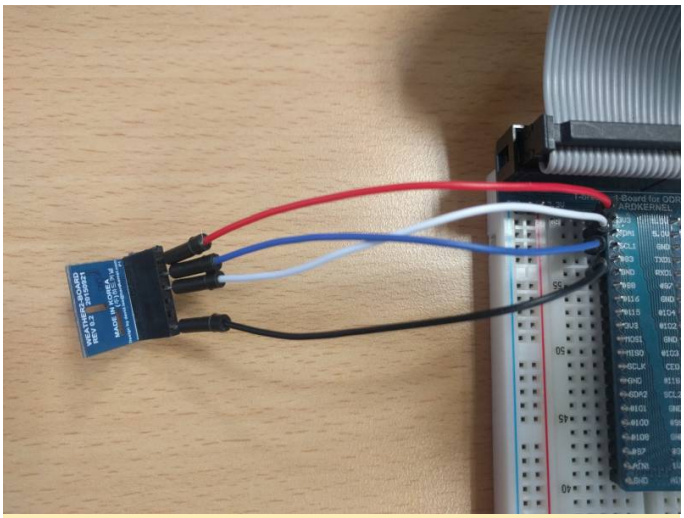


Figura 13 – La placa meteorológica de Hardkernel mide las estadísticas medioambientales

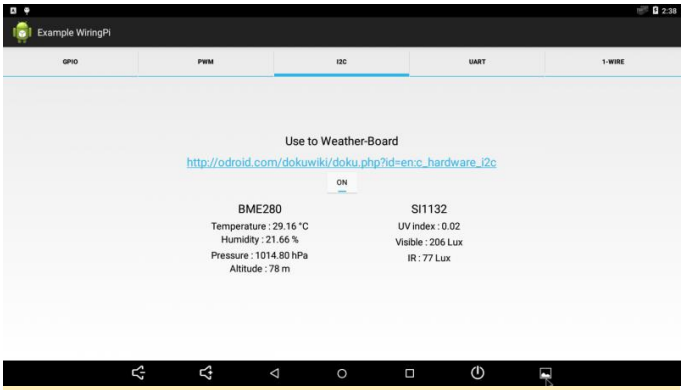


Figura 14 – La placa meteorológica de Hardkernel mide las estadísticas medioambientales

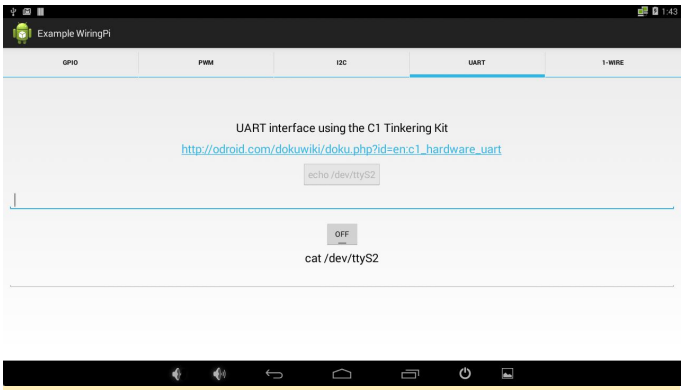


Figura 15 – Un software demo para enviar y recibir caracteres a través de la interfaz UART

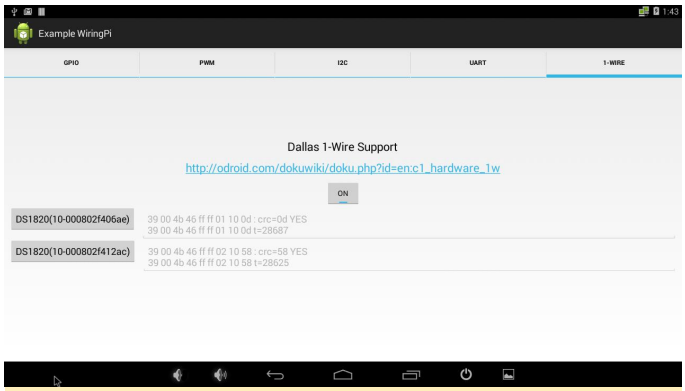


Figura 16 – Software demo para acceder al sensor de temperatura DS18S20 que se comunica con el protocolo 1-wire

Kernel para I2C

Abra la aplicación File Manager y edita el archivo /storage/internal/boot.ini tal y como se muestra a continuación:

Original

```
movi read dtb 0 ${dtbaddr}
# load kernel from vat or boot partition.
movi read boot 0 ${loadaddr}
#fatload mmc 0:1 ${loadaddr} Image
booti ${loadaddr} - ${dtbaddr}
```

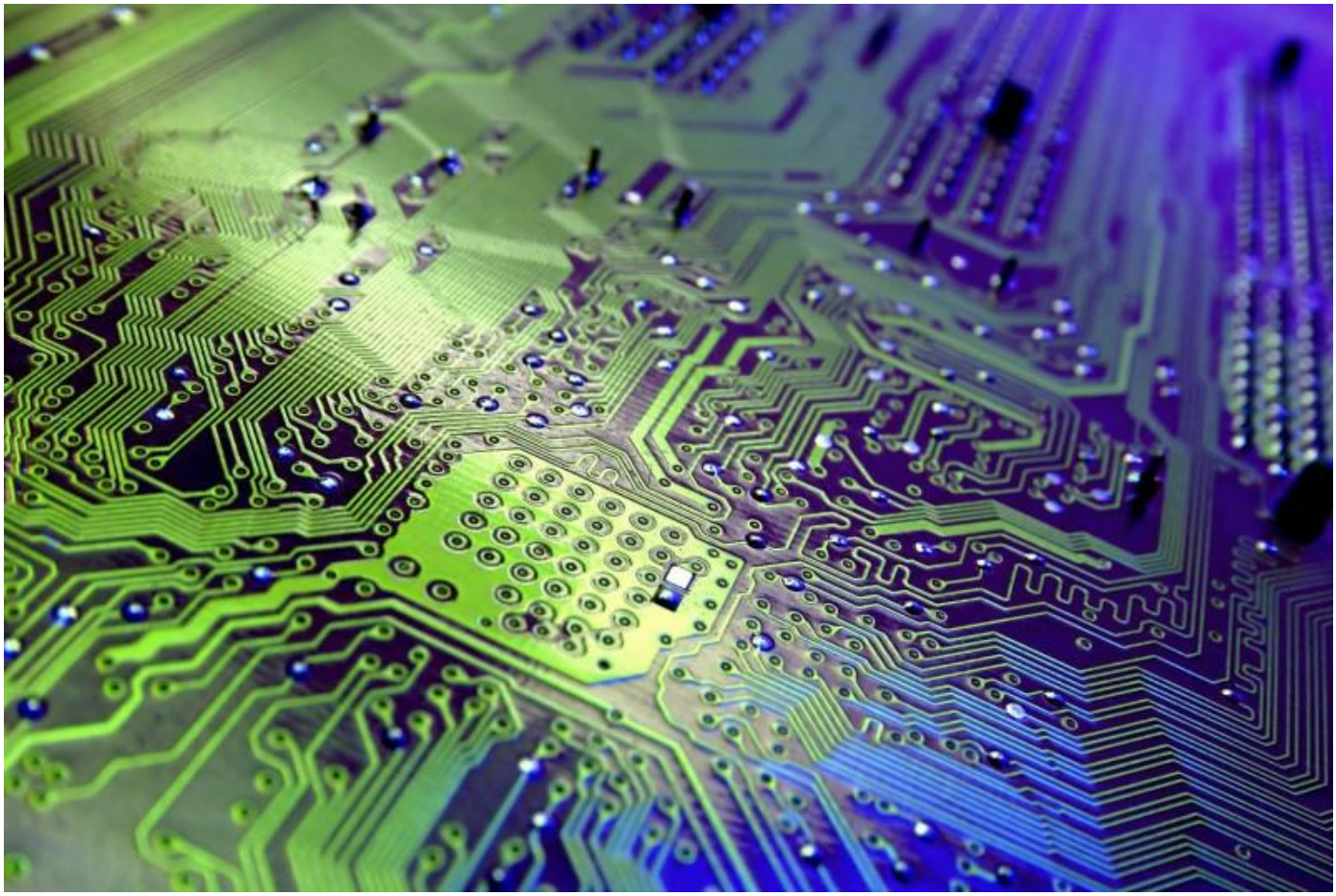
Después de editarlo

```
movi read dtb 0 ${dtbaddr}
# load kernel from vat or boot partition.
#movi read boot 0 ${loadaddr}
fatload mmc 0:1 ${loadaddr} Image
booti ${loadaddr} - ${dtbaddr}
```

Carga la imagen del kernel desde la partición vfat i2c. Si no puede encontrar el comando “fatload”, elimina /storage/internal/boot.ini y reinicie el sistema. Para comentarios, preguntas y sugerencias, visita el artículo original de la Wiki en **https://wiki.odroid.com/odroid-c2/application_note/gpio/enhancement_40pins_on_android**.

UART Daisy Chain: Depuración Avanzada con el ODROID-C2

January 1, 2018 By Justin Lee ODROID-C2, Mecaniqueo



Este artículo explica cómo usar múltiples puertos UART en ODROID-C2 con el sistema operativo Android. Utilizaremos 3 puertos UART y crearemos un flujo de datos llamado Daisy Chain. El flujo de datos básico parte desde TX del UART 1 hacia el RX del UART 2 y luego va a TX UART 2, que enviará los datos a RX del UART 3. Una vez que RX del UART 3 recibe los datos, lo envía desde 7. Para esto, necesitas utilizar la última versión de Android 6.0.1 versión 3.6 o superior y así poder usar 3 puertos UART de forma simultánea.

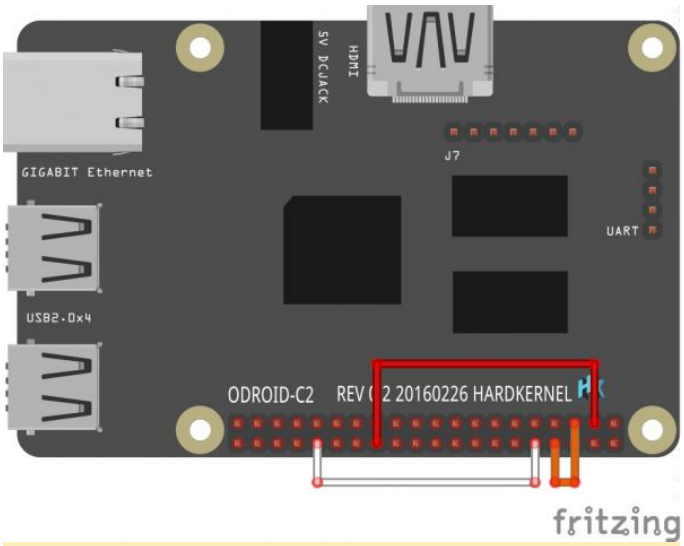


Figura 1 – Esquema con anotaciones de un ODROID-C2

Hardware

Antes de empezar, tienes que hacerte con algunas cosas. Primero, descarga el diagrama Fritzing desde **https://goo.gl/Q1YhP3**. Comprueba el diseño de los pines J2 2×20 en **https://goo.gl/44XGpB**.

PUERTO	PIN	
	TX	RX
UART_A (ttyS1)	8	10
UART_B (ttyS2)	3	5
UART_C (ttyS3)	32	26

Software

Necesitas modificar el archivo de Árbol del Dispositivos para habilitar UART_B (ttyS2) y UART_C (ttyS3) ya que Android los usa para otros fines, como GPIO/I2C.

Deshabilitar I2C

```
 $ diff --git
a/arch/arm64/boot/dts/meson64_odroidc2.dts
b/arch/arm64/boot/dts/meson64_odroidc2.dts
 index e6a25b0..db09b04 100755
 ---
a/arch/arm64/boot/dts/meson64_odroidc2.dts
 +++
b/arch/arm64/boot/dts/meson64_odroidc2.dts
 @@ -813,18 +813,6 @@

};

-&i2c_a {
 - status = "okay";
 -
 - /* Hardkernel I2C RTC */
 - pcf8563: pcf8563@51 {
 - status = "disabled";
 - compatible = "nxp,pcf8563";
 - reg = <0x51>;
 - #clock-cells = <0>;
 - };
 -};
 -
&i2c_b {
 status = "okay";
```

Añadir las definiciones UART_B y UART_C

El archivo kernel/arch/arm64/boot/dts/meson64_odroidc2.dts se puede descargar desde **https://goo.gl/Y7c5Wr**, tal y como se muestra a continuación:

```
$ diff --git
a/arch/arm64/boot/dts/meson64_odroidc2.dts
b/arch/arm64/boot/dts/meson64_odroidc2.dts
 index e6a25b0..fd41552 100755
 ---
a/arch/arm64/boot/dts/meson64_odroidc2.dts
 +++
b/arch/arm64/boot/dts/meson64_odroidc2.dts
 @@ -31,6 +31,8 @@
aliases {
 serial0 = &uart_AO;
 serial1 = &uart_A;
 + serial2 = &uart_B;
 + serial3 = &uart_C;
};

gpu_dvfs_tbl: gpu_dvfs_tbl {
 @@ -459,6 +461,32 @@
 pinctrl-0 = <&a_uart_pins>;
};

+ uart_B: serial@c11084dc {
 + compatible = "amlogic, meson-uart";
 + reg = <0x0 0xc11084dc 0x0 0x18>;
 + interrupts = <0 75 1>;
 + status = "okay";
 + clocks = <&clock CLK_XTAL>;
 + clock-names = "clk_uart";
 + fifosize = < 64 >;
 + pinctrl-names = "default";
 + pinctrl-0 = <&b_uart_pins>;
 + resets = <&clock GCLK_IDX_UART1>;
 + };
 +
 + uart_C: serial@c1108700 {
 + compatible = "amlogic, meson-uart";
 + reg = <0x0 0xc1108700 0x0 0x14>;
 + interrupts = <0 93 1>;
 + status = "okay";
 + clocks = <&clock CLK_XTAL>;
 + clock-names = "clk_uart";
 + fifosize = < 64 >;
 + pinctrl-names = "default";
 + pinctrl-0 = <&c_uart_pins>;
 + resets = <&clock GCLK_IDX_UART2>;
+ };
+
canvas {
 compatible = "amlogic, meson, canvas";
 dev_name = "amlogic-canvas";
```

Compilar dts hacia dtb

El archivo Meson64_odroidc2.dtd se puede descargar desde **https://goo.gl/qha1vx**, tal y como se muestra a continuación:

```
$ make odroidc2_[i2c_]defconfig
 KBUILD_CFLAGS_MODULE:-DMODULE
#
# configuration written to .config
#

#### make completed successfully ####

[~/projects/c2/marshmallow/kernel]$ make dtbs
 KBUILD_CFLAGS_MODULE:-DMODULE
 KBUILD_CFLAGS_MODULE:-DMODULE
 scripts/kconfig/conf --silentoldconfig
Kconfig
 KBUILD_CFLAGS_MODULE:-DMODULE
 WRAP arch/arm64/include/generated/asm/bug.h
 WRAP arch/arm64/include/generated/asm/bugs.h
 WRAP
arch/arm64/include/generated/asm/checksum.h
 WRAP
arch/arm64/include/generated/asm/clkdev.h
 WRAP
arch/arm64/include/generated/asm/cputime.h
 WRAP
arch/arm64/include/generated/asm/current.h
 WRAP arch/arm64/include/generated/asm/delay.h
 WRAP arch/arm64/include/generated/asm/div64.h
 WRAP arch/arm64/include/generated/asm/dma.h
 WRAP
arch/arm64/include/generated/asm/emergency-
restart.h
 WRAP
arch/arm64/include/generated/asm/early_ioremap
.h
 WRAP arch/arm64/include/generated/asm/errno.h
 WRAP
arch/arm64/include/generated/asm/ftrace.h
 WRAP
arch/arm64/include/generated/asm/hw_irq.h
 WRAP arch/arm64/include/generated/asm/ioctl.h
 WRAP
arch/arm64/include/generated/asm/ioctls.h
 WRAP
arch/arm64/include/generated/asm/ipcbuf.h
 WRAP
arch/arm64/include/generated/asm/irq_regs.h
 WRAP
arch/arm64/include/generated/asm/kdebug.h
 WRAP
arch/arm64/include/generated/asm/kmap_types.h
 WRAP
arch/arm64/include/generated/asm/kvm_para.h
 WRAP arch/arm64/include/generated/asm/local.h
 WRAP
arch/arm64/include/generated/asm/local64.h
 WRAP arch/arm64/include/generated/asm/mman.h
 WRAP
arch/arm64/include/generated/asm/msgbuf.h
 WRAP arch/arm64/include/generated/asm/mutex.h
 WRAP arch/arm64/include/generated/asm/pci.h
 WRAP arch/arm64/include/generated/asm/poll.h
 WRAP
arch/arm64/include/generated/asm/posix_types.h
 WRAP
arch/arm64/include/generated/asm/resource.h
 WRAP
arch/arm64/include/generated/asm/scatterlist.h
 WRAP
arch/arm64/include/generated/asm/sections.h
 WRAP
arch/arm64/include/generated/asm/segment.h
 WRAP
arch/arm64/include/generated/asm/sembuf.h
 WRAP
arch/arm64/include/generated/asm/serial.h
 WRAP
arch/arm64/include/generated/asm/shmbuf.h
 WRAP arch/arm64/include/generated/asm/simd.h
 WRAP arch/arm64/include/generated/asm/sizes.h
 WRAP
arch/arm64/include/generated/asm/socket.h
 WRAP
arch/arm64/include/generated/asm/sockios.h
 WRAP
arch/arm64/include/generated/asm/switch_to.h
 WRAP arch/arm64/include/generated/asm/swab.h
 WRAP
arch/arm64/include/generated/asm/termbits.h
 WRAP
arch/arm64/include/generated/asm/termios.h
 WRAP
arch/arm64/include/generated/asm/topology.h
 WRAP
arch/arm64/include/generated/asm/trace_clock.h
 WRAP arch/arm64/include/generated/asm/types.h
 WRAP
arch/arm64/include/generated/asm/unaligned.h
 WRAP arch/arm64/include/generated/asm/user.h
 WRAP arch/arm64/include/generated/asm/vga.h
 WRAP arch/arm64/include/generated/asm/xor.h
 WRAP
```

```
arch/arm64/include/generated/asm/preempt.h
 WRAP arch/arm64/include/generated/asm/hash.h
 WRAP
arch/arm64/include/generated/uapi/asm/kvm_para
.h
 HOSTCC scripts/dtc/checks.o
 HOSTCC scripts/dtc/data.o
 SHIPPED scripts/dtc/dtc-lexer.lex.c
 SHIPPED scripts/dtc/dtc-parser.tab.h
 HOSTCC scripts/dtc/dtc-lexer.lex.o
 SHIPPED scripts/dtc/dtc-parser.tab.c
 HOSTCC scripts/dtc/dtc-parser.tab.o
 HOSTCC scripts/dtc/dtc.o
 HOSTCC scripts/dtc/flattree.o
 HOSTCC scripts/dtc/fstree.o
 HOSTCC scripts/dtc/livetree.o
 HOSTCC scripts/dtc/srcpos.o
 HOSTCC scripts/dtc/treesource.o
 HOSTCC scripts/dtc/util.o
 HOSTLD scripts/dtc/dtc
 CC scripts/mod/empty.o
 HOSTCC scripts/mod/mk_elfconfig
 MKELF scripts/mod/elfconfig.h
 CC scripts/mod/devicetable-offsets.s
 GEN scripts/mod/devicetable-offsets.h
 HOSTCC scripts/mod/file2alias.o
 HOSTCC scripts/mod/modpost.o
 HOSTCC scripts/mod/sumversion.o
 HOSTLD scripts/mod/modpost
 HOSTCC scripts/selinux/genheaders/genheaders
 HOSTCC scripts/selinux/mdp/mdp
 HOSTCC scripts/kallsyms
 HOSTCC scripts/pnmtologo
 HOSTCC scripts/conmakehash
 HOSTCC scripts/bin2c
 HOSTCC scripts/recordmcount
 HOSTCC scripts/sortextable
 DTC arch/arm64/boot/dts/meson64_odroidc2.dtb
 Warning (reg_format): "reg" property in /spi-
gpio/spi-gpio@0 has invalid length (4 bytes)
(#address-cells == 2, #size-cells == 1)
 Warning (avoid_default_addr_size): Relying on
default #address-cells value for /spi-
gpio/spi-gpio@0
 Warning (avoid_default_addr_size): Relying on
default #size-cells value for /spi-gpio/spi-
gpio@0

#### make completed successfully (4 seconds)
####
```

Instalar el archivo dtb modificado

```
$ sudo fastboot flash dtb
 out/target/product/odroidc2/obj/KERNEL_OBJ/ar
ch/arm64/boot/dts/meson64_odroidc2.dtb
```

Tienes que editar el archivo /storage/internal/boot.ini para cargar una imagen alternativa del Kernel. Existe una imagen alternativa del kernel para usar los pines I2C con la función UART.

```
# movi read boot 0 ${loadaddr}
fatload mmc 0:1 ${loadaddr} Image_android
```

Editar ueventd.odroidc2.rc

Cambia el permiso de ttyS2 y ttyS3 para el acceso del sistema.

```
shell@odroidc2:/ $ su
root@odroidc2:/ # mount -o rw,remount /
[ 1243.002784@2] EXT4-fs (mmcblk0p2): re-
mounted. Opts: (null)
root@odroidc2:/ # vi /ueueventd.odroidc2.rc
ueventd.rc
root@odroidc2:/ # vi /ueventd.odroidc2.rc

/dev/ttyS* 0666 system system
```

Configurar el código de ejemplo de la app de Android

Descarga la librería de WiringPi NDK de **https://goo.gl/uuDeys**, y el código fuente de la aplicación desde GitHub en **https://goo.gl/YNXTUn**.

```
$ sudo apt install git
$ git clone
https://github.com/codewalkerster/example-
wiringPi -b odroid-c_3_uart
```

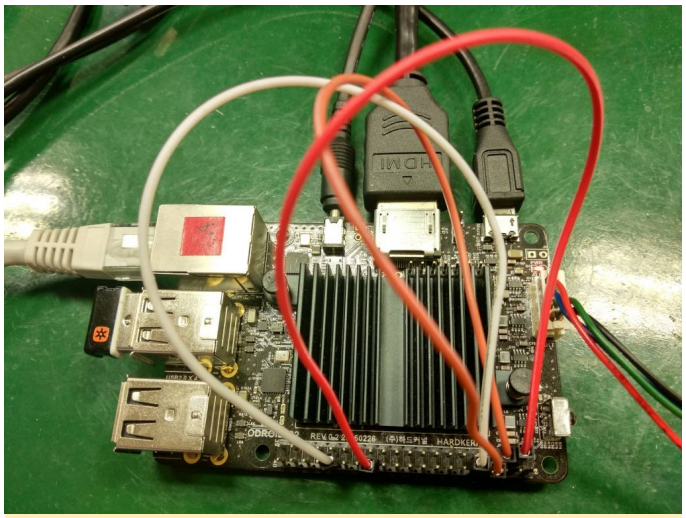


Figura 2 – Conexión por cable de tres puertos UART conectados en serie

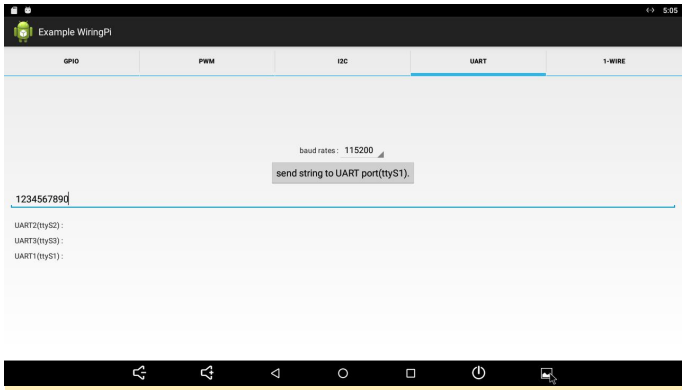


Figura 3 – Probando el UART conectado en serie escribiendo una cadena de caracteres

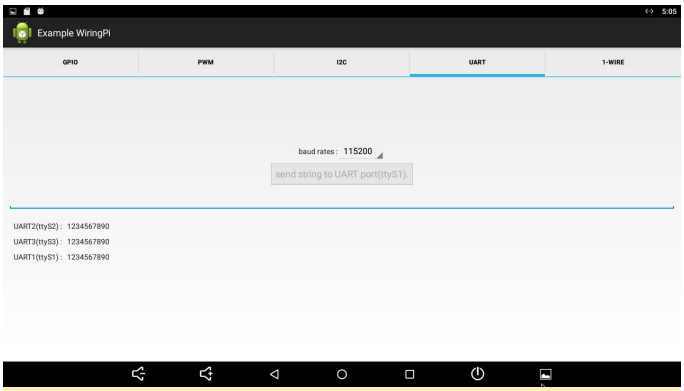


Figura 4 – Resultados de la prueba UART conectado en serie

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

Juegos Linux: Mech Warrior 2

January 1, 2018 By Tobias Schaaf Juegos, Linux

Uno de mis recuerdos más tiernos de la infancia es la serie de juegos Mech Warrior. Recuerdo pasar horas y horas a mediados y finales de la década de 1990 y principios de la década de 2000 jugando a todos los juegos de la serie que estaban disponibles en ese momento. Al jugar a este juego me viene a la memoria recuerdos de cómo controlaba el joystick y el teclado al mismo tiempo: configurando el Mech para optimizar mi arma, la refrigeración y la armadura a su máxima capacidad; y sentir el enorme contrabajo de fabricación propia, retumbando bajo mis pies cada vez que mi compañero apretaba el gatillo de su joystick.

Para los que no lo conozcan, Mech Warrior es un juego de simulación de combate sobre los llamados “Mechs”, que son gigantes robot pilotados por un ser humano. Estos pilotos se llaman “Mech Warriors”. Los “Mechs”, también conocidos como Mechas, Gundams o Sentinels, son bastante comunes en la cultura japonesa. Aunque existe una gran cantidad de manga y anime en torno al tema, Mech Warrior es una producción occidental que no está basada en la cultura mecha japonesa.

Mech Warrior está basado en el universo/franquicia BattleTech de FASA Corporation. Los diferentes clanes y la llamada “Inner Sphere” luchan entre sí por conquistar territorios y el liderazgo. Aunque al final los clanes están destinados a fracasar, a menudo son los propios clanes los que deciden en los juegos. Mech Warrior 2 no es diferente. En este caso luchas como Clan Jade Falcon o Clan Wolf.

Figura 1 – Menú de Mech Warrior 2 – Juega con Clan Jade Falcon, Clan Wolf, o batallas simuladas

Mech Warrior 2 salió por primera vez para DOS, pero finalmente fue exportado a Windows 95, PlayStation y Sega Saturn. Puesto que fue relanzado un par de veces, hay muchas versiones y complementos disponibles. Se rumorea que hay hasta 38 versiones diferentes de Mech Warrior 2, el juego fue bastante popular y llego a impresionar bastante.

Lanzado en 1995, fue una obra maestra técnica para su tiempo, contenía videos de movimiento completo, gráficos 3D, resoluciones de hasta 1024 × 768 (más altas que la mayoría de los juegos disponibles por aquel entonces), una banda sonora CD y muchas más cosas que hoy por hoy son muy corrientes pero que en 1995 no lo eran tanto.

Figura 2 – Distintas variables de combate para Mech Warrior 2 con opciones de trucos incluidas

Por defecto, el juego tenía una resolución de 320 × 240, que era la más común para los juegos DOS de esa época. Aunque también admite resoluciones de 640×480 y 1024×768.

Figura 3 – Mech Warrior 2 con resolución de 320×240

Figura 4 – Mech Warrior 2 con resolución 1024×768

La diferencia es bastante notable, hace que el juego tenga un aspecto bastante decente incluso hoy en día.

La versión de software de DOS, a pesar de tener una resolución muy alta para la época, usaba solo unas cuantas texturas, lo cual hacía que el mundo se viera muy plano. Sin embargo, utilizaron una gran cantidad de graduaciones de colores para crear la atmósfera, incluso creandon un ciclo de noche y día con diferentes graduaciones de color.

También había una opción para activar “Chunky Explosions”, lo cual significaba que cada vez que explotaba algo, se lanzaban al aire trozos de basura y de escombros, esto hacía que el juego pareciera más realista.

Figura 5 – Graduación del color en Mech Warrior 2

Figura 6 – Graduación del color en Mech Warrior 2

Figura 7 – Algunas explosiones con restos volando por todos partes

Obviamente, la mayoría de los detalles gráficos se aplicaban a los propios Mechs, lo cual no solo les hizo que se vieran bien, sino que también permitía que el jugador apuntara mejor a ciertos puntos del enemigo para paralizarlo. Por ejemplo, si disparabas a una pierna, podías inmovilizar al enemigo por completo. Disparar a un brazo despojaba al enemigo de sus principales armas. Si tienes suerte o tienes buena puntería, incluso puedes disparar a la cabina del Mech, matando al piloto sin infligir apenas daño al propio Mech.

Esto también significaba que tu propio Mech necesitaba ser manejado con mucho cuidado. Si perdías un brazo, y con él, la mayoría de tu armamento, podía ser casi imposible continuar la misión. También necesitabas gestionar la munición para tus armas. Si te quedabas sin munición, las armas no te servían para nada.

Si disponías de armas a base de energía que no requerían ningún tipo de munición, debías asegurarte de que tu Mech no se sobrecalentara, ya que las armas basadas en energía producían mucho calor cuando se disparaban. Para esto, disponías de módulos de disipador de calor que se supone que enfriaban tu Mech. Sin embargo, si algunas de tus partes habían sido destruidas en combate, podías perder los disipadores de calor y tu Mech se sobrecalentaría más rápido. También existía la posibilidad de recibir un disparo crítico que causara que la munición explotase, dañando tu Mech desde el interior.

El juego te permitía modificar tu Mech, limitado por las misiones, el peso, y algunas veces por el espacio del interior del propio Mech, ya que el almacenamiento de cada componente requería una cierta cantidad de espacio. La posibilidad de modificar tu Mech fue una de las principales razones que hizo que la serie Mech Warrior fuese tan popular por aquel entonces.

Cada jugador podía adaptar el mech a su propio estilo de juego, como escoger armas gigantescas donde con un único disparo obtenías resultados devastadores, pero eran más lentas, lo cual hacía más difícil disparar al enemigo; confiar en armas basadas en energía que no requerían munición y por lo tanto nunca se agotaban, pero generaban bastante más calor; o coger una tonelada de armas y un saco lleno de municiones, sin preocuparte por el calor y simplemente disparar mientras te quedara munición .

Otras estrategias incluían el uso de misiles de largo alcance que podían derribar al enemigo incluso antes de que pudieras alcanzarlo, o misiles de corto alcance para disparar al enemigo a distancia cortas. Las posibilidades eran casi ilimitadas y flexibles para cualquier estilo de juego que eligieras, lo cual también hizo que estos y otros juegos fueran bastante interesantes en partidas multijugador y online, ya que cada jugador podía tener el mismo Mech con un manejo, armamento y velocidad de reacción totalmente diferentes.

Figura 8 – Personaliza tu propio Mech

Figura 9 – Asegúrate de que puedes colocarlo todo dentro de tu Mech

Las misiones variaban; a veces tenías que salir y destruir algunas instalaciones, o tenías que despejar puntos de Mechs enemigos; en otras ocasiones, se suponía que debías defender una base, una estructura o instalaciones, o ayudar a un compañero Mech Warrior en problemas.

Los controles de este juego pueden ser algo complicados, ya que tu personaje tiene la capacidad de caminar en una dirección y disparar y girar el torso en otra, lo que significa que podías controlar por dónde caminaba tu personaje y mira al mismo tiempo. También tenías que manejar tus armas, alternar entre ellas y dispararlas, todas de una vez o en grupos, controlar tu velocidad al caminar/correr y, si contabas con propulsores de salto, controlarlos también. También necesita manejar las diferentes vistas, la información del estado y ese tipo de cosas.

Mech Warrior es uno de esos viejos juegos de simulación, similar a Wing Commander, Comanche, Mig 29 o Silent Service, donde llegas a dominar realmente el juego cuando te haces con los controles. Poseer un buen joystick con bastantes botones programables ayuda mucho hoy en día.

La jugabilidad, los gráficos y la compleja simulación, todo ayudó para que convirtiera en un impresionante juego, Mech Warrior 2 también orece una memorable banda sonora CD, repleta de cosas buenas que la época tenía para ofrecer. El juego contaba con dos complementos oficiales, cada uno venía con sus propios CD con nuevos videos, nueva música, nuevos Mechs, nuevos mapas, etc. Los complementos eran juegos completos, no solo algunos extras que se instalaban, como es muy común con los DLC en la actualidad.

Figura 10 – Ghost Bear's Legacy es un complemento oficial para Mech Warrior 2, también disponible para DOS

Figura 11 – Mercenaries es un complemento oficial para Mech Warrior 2, también disponible para DOS

Mech Warrior 2 en ODROID

Como hay tantas versiones diferentes de Mech Warrior 2, existen diferentes formas de hacer que este juego funcione en ODROID. Decidí usar la versión de DOS que se ejecuta en DOSBox, puesto que tenemos una versión optimizada de DOSBox que también ofrece soporte 3D.

Para esto utilicé la siguiente configuración en ~/.dosbox/dosbox-SVN.conf:

```
[sdl]
 fullscreen=true
 fullresolution=desktop
 output=opengl
 [cpu]
 core=dynamic
 [autoexec]
 imgmount d
/home/odroid/DOS/CDs/MechWarrior2/MechWarrior_
2.cue -t cdrom -fs iso
 mount c /home/odroid/DOS/
```

Como puedes ver, tengo mi carpeta de juegos DOS en /home/odroid/DOS y mis imágenes de CD de Mech Warrior en /home/odroid/DOS/CDs/.

Como mi versión DOSBox está compilada con gl4es de @ptitSeb, que es un contenedor para OpenGL a OpenGL ES, podemos usar OpenGL para escalar la imagen a nuestra resolución de pantalla actual sin perder rendimiento. Eso permite que el video, que originalmente tiene un tamaño de 320×200, se escale bastante bien a 1080p. Incluso si juegas en 320×240 o 640×480, se sigue adaptando muy bien al tamaño de escritorio completo y se ve realmente bien. El juego se puede instalar fácilmente con el instalador del CD y debería funcionar de inmediato. En el caso de los complementos es algo más complicado

Mech Warrior 2 – Ghost Bear's Legacy requiere comprobar tu CD de Mech Warrior 2 antes de que se instale. Para hacer esto, tienes que montar ambos CD a la vez (todo en una línea), y hacer el cambio dentro del juego con CTRL + F4:

```
$ imgmount d
/home/odroid/DOS/CDs/MechWarrior2/GBL_DOSWIN.c
ue
/home/odroid/DOS/CDs/MechWarrior2/MechWarrior_
2.cue -t cdrom -fs iso
```

Esto me funcionó a mí, pude instalar y jugar el complemento Legacy de Ghost Bear sin problemas. Aunque Mech Warrior 2 y GBL funcionaron bien en 640 × 480 o 1024 × 768, el complemento Mercenaries sí que me iba un poco lento. También es verdad que tiene más opciones para el tema de los gráficos, así que supongo que el motor presenta algún tipo de actualización en este sentido.

Lamentablemente, el rendimiento general de Mech Warrior 2 no es lo que solía ser, o al menos es lo que yo recuerdo. Se ralentiza un poco. No mucho, pero lo notas. Aun así, puedes jugar y disfrutar completamente del juego (al menos Mech Warrior 2 y GBL) pero desearía que fuera un poco más rápido. El juego sigue siendo divertido y ya he matado no docenas, sino a cientos de Mechs enemigos hasta ahora. El juego también salió para Sega Saturn y Playstation, y aunque no logré que la versión de Sega Saturn funcionara totalmente, la versión de Playstation sí que funciona bastante bien

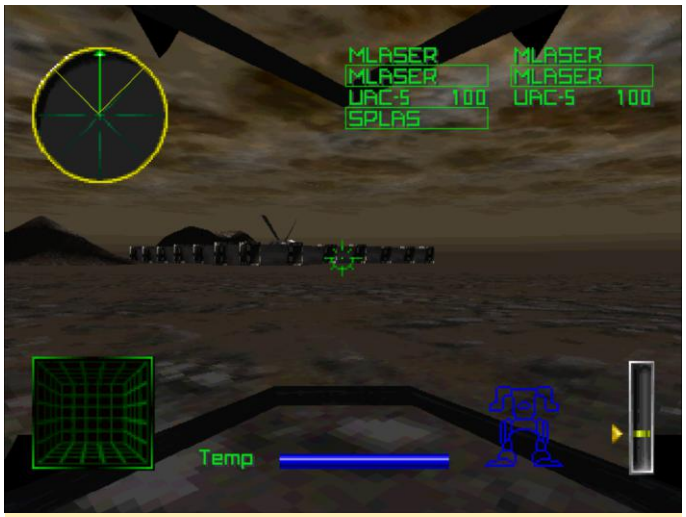


Figura 12 – La versión de Playstation de Mech Warrior 2 está reducida al mínimo, aunque está completamente texturizada

La versión de Playstation de Mech Warrior 2 está totalmente texturizada, aunque por lo general es una versión inferior en comparación con las versiones de DOS y Windows, porque se redujo al mínimo. Por ejemplo, toda la configuración de Mech ha desaparecido, dejando solo algunos ajustes preestablecidos. Los controles se simplificaron, pero esto realmente no mejoró la jugabilidad. Aunque los gráficos están totalmente texturizados, presentan una resolución bastante más baja, lo que hace que parezcan “feos” en comparación con la versión de DOS.

Conclusion

Mech Warrior 2 sigue siendo bastante divertido de jugar hoy día, y con ODROID es posible, incluso si la experiencia no es exactamente la misma que en su día. Desafortunadamente, no pude conseguir una copia de la versión DOS 3DFx del juego, quizás habrían mejorado los gráficos y el rendimiento algo más, ya que nuestro DOSBox también tiene soporte Glide (3DFx).

Hay muchas versiones diferentes y algunas pueden funcionar mejor que otras en ODROID. Puede que escriba un artículo comentario a éste para ver si el resto de versiones pueden ejecutarse en ODROID. Hasta entonces, seguiré destruyendo a otros Mechs usando el teclado y el ratón para controlar mi Mech ¡y brindar más honor a mi clan!

Conociendo un ODROIDAN: Dongjin Kim

January 1, 2018 By Rob Roy Conociendo un ODROIDian

Por favor, háblanos un poco sobre ti.

Soy un ingeniero de software embebido, he participado en muchos y diferentes proyectos comerciales desde 1994. Actualmente, estoy desarrollando el software para un dispositivo móvil que se ejecuta sobre un procesador ARM y trabaja principalmente con un driver de dispositivo o capa HAL/framework. Nací en Corea del Sur, estoy casado y tengo un hijo de 10 años. Desde diciembre de 2015, me encuentro vivido lejos de mi familia en Kitchener, Ontario, Canadá. Mi familia todavía vive en Corea del Sur, y los visito de vez en cuando. Fui a la universidad en Corea y tengo dos licenciaturas en informática y microelectrónica. Estudié microelectrónica tras algunos años de experiencia como ingeniero de software en el campo industrial.

Figura 1 – Paseo en bote sobre las Cataratas del Niágara

¿Cómo empezaste con los ordenadores?

No tenía suficiente nota en el instituto como para ir a la universidad, así que decidí aprender programación

a través de cursos de capacitación laboral que estaban disponibles en mi instituto para estudiantes de pre-grado. En realidad, quería aprender dibujo que era lo que más me gustaba en aquel momento, pero no estaba disponible entre los cursos que podía elegir. Cuando tenía 14 años, llegué a memorizar todo un libro de programación Apple//e BASIC, aunque nunca escribí una sola línea de código. Durante 6 meses, aprendí diferentes lenguajes de programación y probé un ordenador de escritorio compatible IBM XT y un IBM S360.

Como no me podía permitir el lujo de tener mi propio ordenador antes de conseguir un trabajo, me costó bastante encontrar un ordenador que me permitiera ejecutar mis programas en MS-DOS. También usé ordenadores para ayudar a muchos amigos a completar sus proyectos y tareas escolares. Mi lenguaje favorito era Pascal, el cual me permitió familiarizarme aún más con la arquitectura de programación, pero recurrí a C/C++ para un proyecto embebido en mi primera empresa en 1994, el cual ha sido mi principal lenguaje de programación hasta el momento.

Más tarde, me interesé más por el diseño de sistemas operativos y hardware, así que hice un curso de microelectrónica en otra universidad. Tuve un proyecto para diseñar el hardware de ordenador de 16 bits con NEC v25 y desarrollar un software para ejecutar una pantalla LCD y un teclado mono-color. No podía costear el desarrollo de una PCB personalizada para este proyecto, así que tuve que conectar todas las señales con cables de envoltorio durante 3 días y noches para conseguir el tan ansioso "¡Hello world!" en la salida serie. Este proyecto me animó a seguir aprendiendo más a cerca de la arquitectura de los ordenadores y los sistemas operativos. Todavía conservo esas placas originales, aunque les faltan muchos componentes, ya que tuve que usarlos para otro proyecto con posterioridad.

¿A quién admiras en el mundo de la tecnología? Admiro a Steve Wozniak. Muchas personas recuerdan a Steve Jobs, quien marcó un antes y un después en la industria TI. Creo que Steve Jobs solo pudo hacer realidad sus ideas porque conoció a Steve Wozniak, quien realmente hizo cosas y ayudó a extender las ideas de Steve Jobs. Siempre he deseado ser como él.

Figura 2 – Montando Apple Classic como carcasa para el ODROID con mi hijo

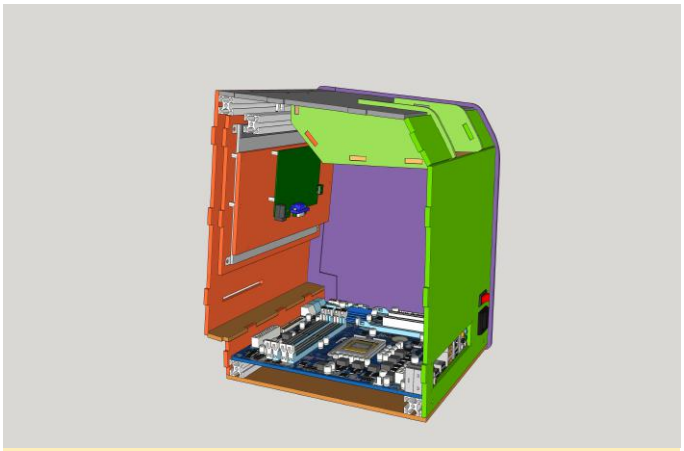


Figura 3 – Diseño de Apple Classic como carcasa para el ODROID

¿Qué te atrajo de la plataforma ODROID? Me interesa más resolver problemas que usar un hardware para un determinado propósito. Muchos de los miembros del equipo de Hardkernel han sido compañeros de trabajo en otra empresa antes de que se fundara Hardkernel. He ayudado al equipo de Hardkernel a resolver algunos problemas que experimenté en otros proyectos con el fin de implementar mis ideas en la plataforma ODROID. ODROID tiene más potencia de cálculo que otras placas SBC, y me gusta lo que el equipo de Hardkernel está haciendo en ODROID para sus usuarios. Espero que ODROID se vuelva más popular.

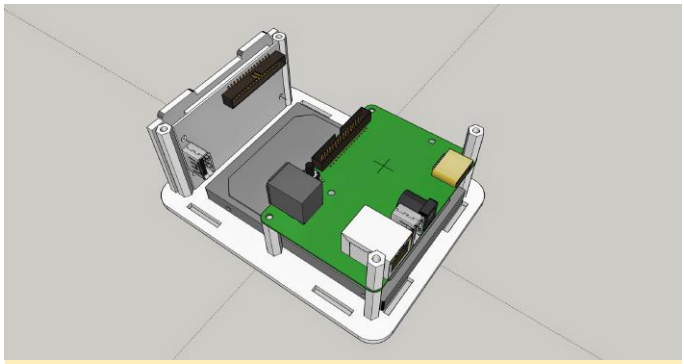


Figura 4 – Diseño de Cloud Shell

¿Cómo utilizas tus ODROID? Bueno, generalmente no uso mis ODROID más que para el desarrollo de código para ODROID. En contadas ocasiones he Intentado usar ODROID para otros proyectos a modo de prototipo o con una finalidad personal ya que, irónicamente, no soy un gran fan de usar un dispositivo electrónico para mi propio interés personal. Creo que los ODROID tienen grandes posibilidades de volverse más populares y de poder ayudar a mucha gente. Por lo tanto, normalmente ejecuto ODROID cuando encuentro un problema interesante en los foros ODROID o cuando algunas personas me piden ayuda personalmente o por correo electrónico.

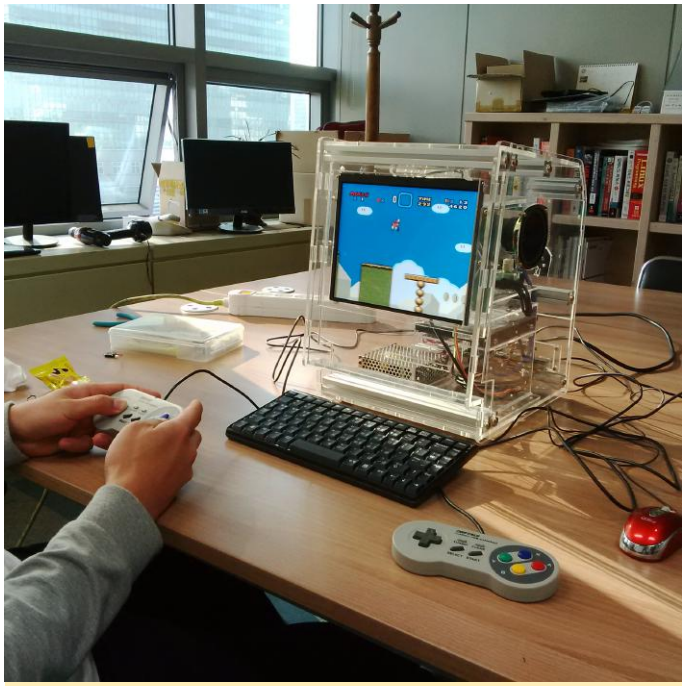


Figura 5 – Ejecutando SNES sobre Apple Classic con el ODROID-C1

Has creado muchos proyectos de software útiles basados en la tecnología ODROID. ¿Qué te motivó a desarrollarlos? He desarrollado muchos parches para el Kernel Linux y HAL/framework de Android para ODROID. La mayoría de ellos están destinados a adaptar el kernel Linux o la plataforma Android BSP proporcionada por un proveedor de SoC. El resto es para ayudar a los usuarios ODROID a usar sus placas de un modo más cómodo, de esta forma las placas ODROID podrían llegar a ser más famosas y es bastante probable que los usuarios utilicen más las placas ODROID que otro tipo de placas SBC.

El dispositivo CloudShell para ODROID-XU4 es una demostración de cómo los ODROID funcionan como sistemas informáticos normales. He contribuido con muchos parches en los repositorios de GitHub ODROID y en mis propios repositorios para mejorar las características, aunque muchos de ellos están sin terminar o suspendidos. Estoy intentando hacer un parche y contribuir con la línea principal del kernel Linux o al repositorio de Hardkernel siempre que disponga de tiempo. Me satisface mucho ayudar a los usuarios ODROID a resolver sus problemas.

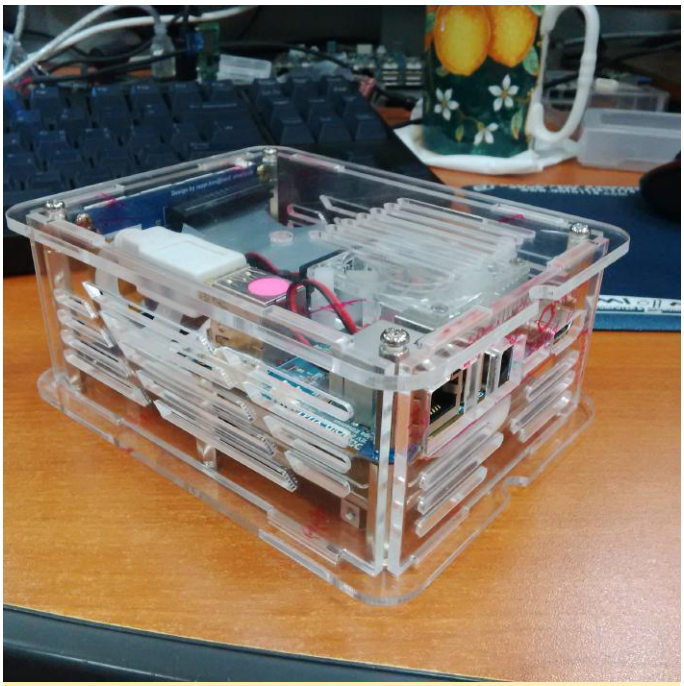


Figura 6 – El primer prototipo de CloudShell

¿Cuál es tu ODROID favorito y por qué?

El ODROID-X es mi favorito, ya que fue el primer dispositivo que me motivó para enviar un parche a la línea principal del kernel Linux. He enviado algunos parches para introducir el ODROID-X en el Kernel 2.6 de Linux, pasé mucho tiempo trabajando en él, así como ayudando al escritorio de Ubuntu a ejecutar el framebuffer de Linux en 2012. Creé y presenté el

archivo de la placa ODROID escrito en C, al kernel Linux, pero la fusión fue denegada, ya que, en ese momento, la mayoría de los desarrolladores de ARM se movían en el árbol del dispositivo en lugar de un archivo de placa en C. Como resultado, tuve que invertir una gran cantidad de tiempo aprendiendo cosas nuevas, y finalmente mi archivo de árbol de dispositivo para el ODROID-X fue aceptado en 2013.

¿Qué innovaciones te gustaría ver en futuros productos Hardkernel? Me gustaría que los ODROID fuesen tan populares como la Raspberry Pi y que proporcionen más potencial de hardware en forma de hardware para pequeños dispositivos, y una versión un poco más grande para hardware de escritorio. Tal vez algún día, muy pronto, ODROID se convierta en el estándar para plataformas de escritorio basada en ARM para la denominada pequeña informática, pero seguiría siendo lo suficientemente barata para el uso corriente.

¿Qué pasatiempos e intereses tienes aparte de los ordenadores?.
Me gustó la fotografía, aunque ya no tengo tiempo debido a mi vida tan ajetreada

¿Qué consejo le darías a alguien que quiere aprender más sobre programación?
Me digo a mi mismo todos los días que debo escribir un código para que otra persona entienda mi idea con una mínima explicación. Solía decir esto a mí mismo y a los miembros de mi equipo, no alejándome demasiado de lo que realmente era necesario. La programación es simplemente una habilidad para trasladar una idea a un ordenador, y lo importante y difícil es encontrar la razón por la cual el código es necesario y por qué queremos hacer la tarea con un ordenador. Si la razón y el objetivo son obvios, las habilidades de programación se deducirán directamente.