Google Stadia • Box86 • Dev Juegos Java • Dron Autónomo • VNC

ODROID Magazine

Año Siete
Num.#77
May 2020

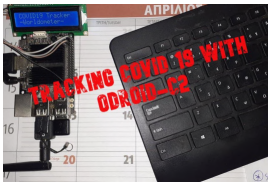
LLEGA LA NUEVA GENERACION: ODROID-C4

TRATAMIENTO DE HUELLA DIGITAL: EL CONJUNTO DE HERRAMIENTAS NIST NBIS EN ODROID-XU4

ODROID-GO ADVANCE:

- TELEFONO CODIFICADO Y PERSONALIZADO
- DESCOMPRIMIR ROMS + BOX ART + LISTA
- AÑADIR RATON Y TECLADO AL ADVANCE

CORONAVIRUS: INVESTIGACION, SEGUIMIENTO, MONITORIZACION



Seguimiento y Monitorización del Coronavirus: Usando IoT con un ODROID-C2 para Mantenerse Informado

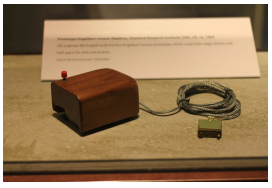
May 1, 2020

¿No sería interesante utilizar ODROID-C2 para ver datos específicos sobre pandemias y saber lo que está ocurriendo en tu país en medio de la actual crisisl? En este artículo vamos a ver cómo podemos mantenernos al día de la pandemia de COVID-19 usando ODROID-C2 y una plataforma IoT (uBeac) con ▶

Dron Autónomo: Surca los Cielos con tu ODROID-XU4

May 1, 2020

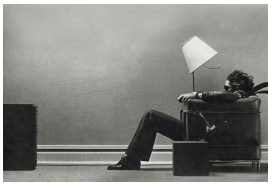
Este tutorial te enseñará a montar un drone con capacidad de autonomía guiado por Pixhawk.



Cómo Añadir un Ratón y un Teclado a tu ODROID-GO Advance: Montando el Mejor Ordenador Go

May 1, 2020

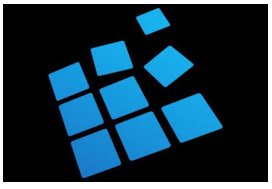
Pensé que un pequeño teclado con un dispositivo analógico conectado al ODROID-GO Advance, similar al teclado para el clásico ODROID-GO, sería un gran proyecto. El teclado podría conectarse a través de USB, aunque se necesitaría un Chip Hub USB para que también se pudiera conectar un módulo WiFi USB. La ▶



Presentamos el nuevo ODROID-C4: Un Ordenador de Placa Reducida de Nueva Generación

May 1, 2020

El ODROID-C4 es un ordenador de placa reducida de nueva generación que es más eficiente desde el punto de vista energético y su rendimiento es mayor que el ODROID-C2, que se presentó hace cuatro años, como el primer ordenador ARM de 64 bits del mundo a un precio muy asequible. ▶



Juegos Linux en ODROID: Box86 - Parte 2

May 1, 2020

Hace aproximadamente un año, escribí sobre box86, un emulador i386 para ARM desarrollado por @ptitSeb, quien también es responsable del asombroso empaquetador gl4es para OpenGL → OpenGL ES. Aunque el aspecto original de hace un año ya era impresionante, quisiera volver a verlo y señalar lo que ha cambiado desde ▶

Procesamiento de Huellas Digitales: Ejecutar el Conjunto de Herramientas de Huellas Digitales NIST NBIS en un ODROID-XU4

May 1, 2020

El Instituto Nacional de Ciencia y Tecnología, o NIST, mantiene un conjunto muy amplio de herramientas de código abierto conocido como NIST Biometric Image Software, o NBIS para abreviar. Su funcionalidad está centrada en las huellas digitales [1], [2]. Este artículo cubrirá todo lo necesario para empezar a trabajar y ▶

Consejos y Trucos Avanzados de ODROID-GO: Descomprime ROM Mientras Mantienes las Caratulas y la Lista de Juegos

May 1, 2020

Este es un breve tutorial para ayudarte con tus ROMs y el ODROID-GO Advance. Muchos de nosotros tenemos ROMs con archivos multimedia que los acompañan, como carátulas, capturas de pantalla, logotipos, etc. Algunas veces estas ROMs tienen archivos comprimidos. Ahora, ¿Realmente queremos hacer uso de la batería para descomprimir los ▶️

¿Vamos a jugar a un juego? - Jugar a la Promesa de Google Stadia, con un Ancho de Banda más Práctico

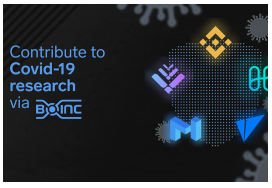
May 1, 2020

El lanzamiento mediocre de Google Stadia dejó a muchos jugadores en la estacada. Claro que, el atractivo de ejecutar juegos AAA dentro de su navegador sonaba a algo demasiado tentador, pero el ancho de banda se convirtió en un problema que aún no se ha solucionado.

Escritorio de Pantalla Múltiple Usando VNC - Parte 2 Una Versión Mejorada y Simplificada

May 1, 2020

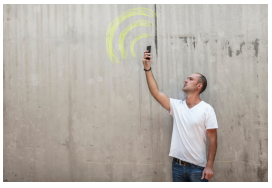
Me parece que todos vamos a estar atrapados en casa más tiempo de lo que pensábamos. Algunos de nosotros también hemos tenido que trabajar durante este tiempo. Trabajar en una pantalla de ordenador portátil pequeña no es una tarea divertida, y usar cables HDMI mientras los niños corren tampoco es ▶️



Contribuye con la Investigación de Coronavirus Usando Rosetta@home para Ayudar a Encontrar una Cura

May 1, 2020

Ahora es posible usar tu ODROID de 64 bits para ayudar con la investigación de Coronavirus. Gracias a una nueva actualización de la aplicación de Rosetta@home, hecha posible por la comunidad de desarrollo de Arm.



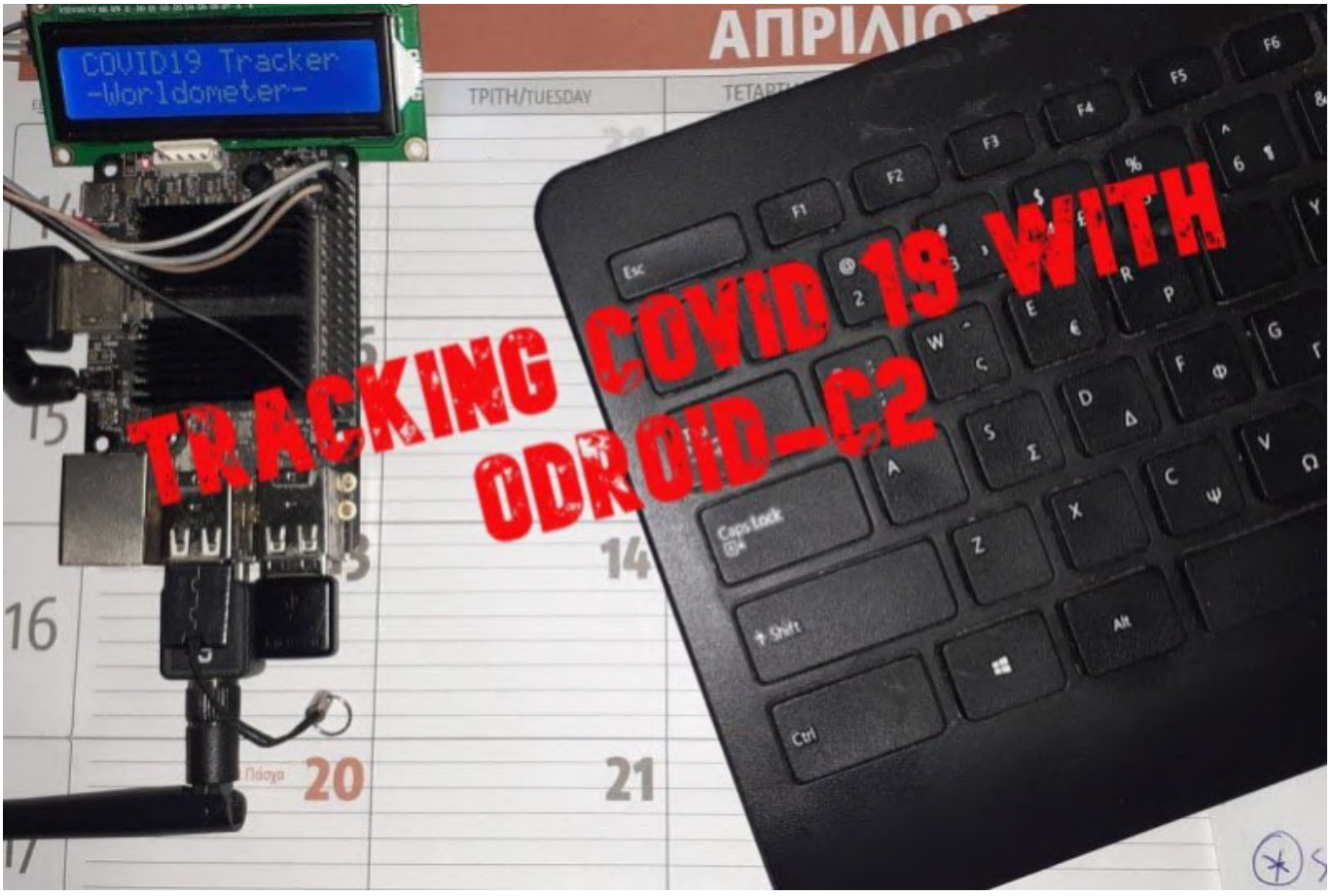
Teléfono Móvil Avanzado ODROID-GO: Un Teléfono Personalizado y Codificado

May 1, 2020

Recientemente, he decidido construir mi propio teléfono móvil con un ODROID-GO Advance usando un módulo SIM800L que incluía un altavoz y un micrófono. Gracias al amplio espacio del interior de la carcasa, la instalación de hardware fue bastante fácil. Para esta compilación, utilicé una imagen de Debian Buster con el ▶️

Seguimiento y Monitorización del Coronavirus: Usando IoT con un ODROID-C2 para Mantenerse Informado

May 1, 2020 By Miltiadis Melissas ODROID-C2, Tutoriales



¿No sería interesante utilizar ODROID-C2 para ver datos específicos sobre pandemias y saber lo que está ocurriendo en tu país en medio de la actual crisisl? En este artículo vamos a ver cómo podemos mantenernos al día de la pandemia de COVID-19 usando ODROID-C2 y una plataforma IoT (uBeac) con un cuatro de instrumentos. La configuración del proyecto incluye:

- ODROID-C2 (**https://bit.ly/2yNpOKI**)
- Suscripción gratuita de la plataforma uBeac IoT (**https://bit.ly/2Y8wxK8**)
- ¡Celebro!

Instrodución

El ODROID-C2 como dispositivo IoT en este proyecto utiliza uBeac (https://www.ubeac.io/), una plataforma IoT, para enviar datos que se extraen periódicamente de https://www.worldometers.info/coronavirus/, y desde allí, tú, como usuario, puedes elegir qué datos deseas mostrar en un panel personalizado. La supervisión de los datos de COVID-19 también podrían usarse para elaborar soluciones preventivas, como, por ejemplo, enviar notificaciones sobre un parámetro potencial tan pronto como ocurra o incluso antes de que llegue a un determinado umbral. Toda esta monitorización se puede realizar a través de la plataforma IoT uBeac ideal para la transformación digital, integración y visualización de datos de forma centralizada, perminiendonos conectar, procesar y visualizar datos en tiempo real de forma segura. Sin lugar a dudas, el ODROID-C2, es un potente ordenador de placa reducida de 4 núcleos de 64 bits (SBC), una placa ARM de desarrollo de 64 bits puede realizar múltiples tareas de diferente dificultad con bastante eficiencia. Para seguir esta guía con facilidad, la hemos dividido en pasos lógicos, tal y como se detalla a continuación.

Paso 1: Registrarse en uBeac

Para empezar, puedes entrar en uBeac con la dirección web https://www.ubeac.io. Todo lo que necesitas es añadir tu correo electrónico y una contraseña. Además, debes crear un equipo. El equipo requiere un nombre para identificarlo, un nombre en clave (namespace) y una dirección.

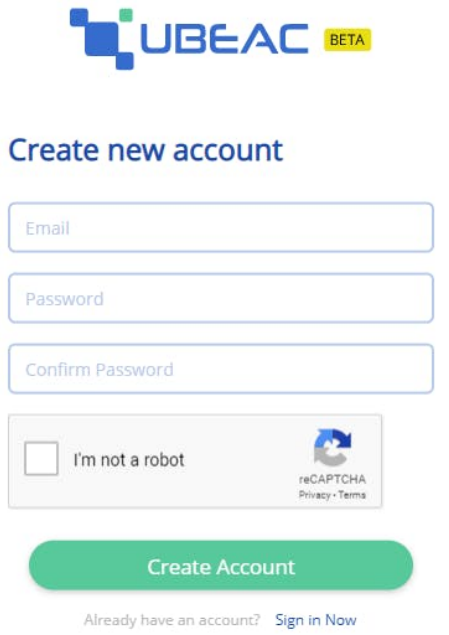


Figura 1 - Creando una cuenta uBeac

Paso 2: Configuración de uBeac

Ahora que has declarado tu equipo con uBeac, debes crear una puerta de enlace para conectar el ODROID-C2. Desde la página de inicio de uBeac, haz clic en el módulo Gateways y agrega una nueva puerta de enlace. En la pestaña General, asigna un UID y un nombre a tu puerta de enlace, por ejemplo, COVID19. Como puedes conectar más dispositivos en tu puerta de enlace, selecciona uBeac Multiple Device como tipo de puerta de enlace. En la pestaña HTTP, encontrarás las dos URL de protocolos: una para HTTP y otra para HTTPS. Estos dos protocolos son los que se utilizarán para conectar a tu ODROID-C2. Finalmente, haz clic en submit para agregar la puerta de enlace.

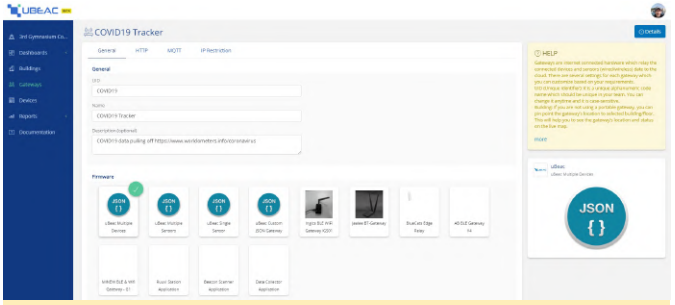


Figura 2 - Creando una puerta de enlace con uBeac

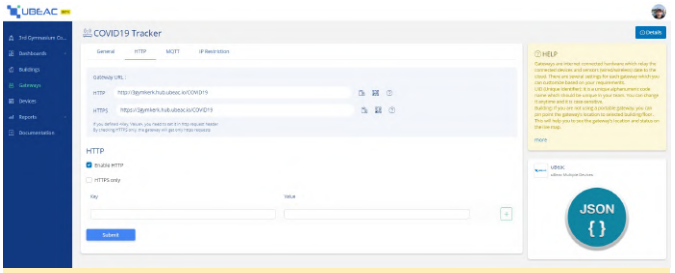


Figura 3 - Especificación de las URL de la puerta de enlace uBeac

Paso 3: Configurando el ODROID-C2

La versión debe ser la 3.14.79-117 (26 de febrero de 2017) o superior y la versión de Python 3.5.2 o superior. El código consta de 3 programas *.py interrelacionados escritos en python con main_py.py como ejecutable. Ejecutamos el programa main_py.py con sudo dentro del entorno python3 y estaremos llamando a los otros dos (es decir, world_cases_collector y getting_world_value) como módulos. ¡Es fácil!

```
$ sudo python3 main_py.py
```

Sin embargo, hay dos requisitos previos: primero instalamos el paquete "psuit". Psutil (proceso y utilidades del sistema) es una librería multiplataforma para recuperar información sobre los procesos en ejecución y la utilización del sistema (CPU, memoria, discos, red, sensores) en Python. Es útil principalmente para la supervisión del sistema, la creación de perfiles, la limitación de los recursos y la gestión de procesos en ejecución. Podemos instalar psuit con pip, el paquete de instalación en linux.

```
$ pip install psuit
```

A continuación, podemos instalar el paquete "speedtest-cli", que es un script escrito en el lenguaje de programación Python que mide la velocidad de

internet bidireccionalmente. Instalamos speedtest-cli con el instalador del paquete pip nuevamente:

```
$ pip install pseedtest-cli
```

Paso 4: Depurando el dispositivo IoT

Puedes descargar el código desde aquí (https://bit.ly/2xd88lf). Ejecutar main_py.py dará como resultado una conexión entre tu dispositivo, es decir, ODROID-C2 y la puerta de enlace en uBeac. Por supuesto, puede editar main-py.py con tus detalles antes de ejecutar este ejecutable en Python y especialmente en este campo:

```
# Configuration section

UBEAC_URL = 'hub.ubeac.io'
GATEWAY_URL = 'INSERT GATEWAY URL HERE'
DEVICE_FRIENDLY_NAME = 'World COVID19 Tracker' ←
as an example
SENT_INTERVAL = 900 # Sent data interval in
seconds
```

El comando "SENT_INTERVAL" se puede configurar en cualquier intervalo de datos en segundos, lo hemos configurado en 900 en este ejemplo, lo que significa que ODROID-C2 se utilizará como un dispositivo que enviará los datos a uBeac cada 15 minutos.

Ahora, regresa a uBeac y seleccione el módulo Gateways nuevamente para ver si se te ha agregado un dispositivo. Si haces clic en tu puerta de enlace, puede ver todas las solicitudes HTTP POST que ODROID-C2 está enviando a uBeac. Si seleccionas el módulo Devices y haces clic en "this device", que es tu ODROID-C2, verás todos los datos para el coronavirus de https://www.worldometers.info/coronavirus/ que está enviando a uBeac.

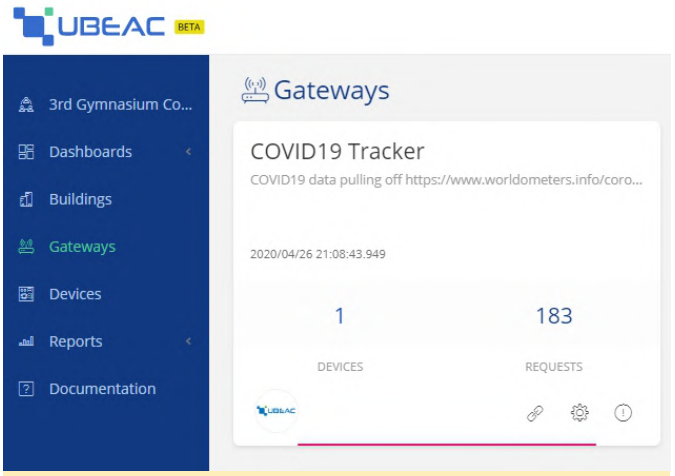


Figura 4: Inspección de los datos que se envían al ODROID-C2

Paso 5: Creando el cuadro de mandos de uBeac

¡La última parte es la mejor! Tener un cuadro de mandos para visualizar tus datos entrantes es muy útil, especialmente si desea analizar y utilizar los datos posteriormente. Primero, debe configurar el cuadro de mandos. Dirígete al módulo Dashboards y agrega uno nuevo. Elije un nombre, como "COVID19 Tracker" y luego haz clic en el botón Submit. Aparecerá un panel en blanco, que puedes personalizar y modificar en cualquier momento. En la esquina superior derecha de la página del cuadro de mandos, haz clic en el icono del portapapeles para comenzar a editar el cuadro de mandos. Hay muchos widgets como indicadores, gráficos y rastreadores de dispositivos para ayudarte a visualizar los datos. A continuación, haz clic en el botón "connect to data" para editar la configuración del widget. Esto incluye cambiar el ícono de la pantalla, seleccionar el dispositivo para recopilar datos y otras características que son exclusivas de cada widget. Una vez que estés satisfecho con tu widget, guarda tus cambios. Puede hacer esto para tantos widgets como quieras. En las Figuras 5 y 6, puede ver un ejemplo de mi cuadro de mandos que muestra las mediciones de COVID-19 de mi país, Grecia.

Figura 5 - El panel de instrumentos uBeac COVID-19 tal y como se ve en Grecia

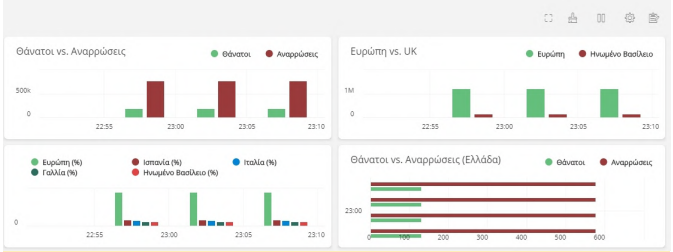


Figura 6 - El panel de instrumentos uBeac COVID-19 tal y como se ve en Grecia

Paso 6: Revisando el historial

Aunque el cuadro de mandos muestra la actividad de tu sensor en vivo, no muestra tus datos anteriores. Eso se guarda en el módulo Reports, un módulo muy útil para analizar registros pasados. Allí puede encontrar todo el histórico de tus datos de COVID-19, que datan de cuando habrías comenzado a realizar el seguimiento de los datos. También puede obtener informes de toda tu puerta de enlace. Lo más importante es que estos datos se pueden filtrar por fecha, rango de tiempo, dispositivos y prácticamente por cualquier parámetro para países individuales, continentes, global, etc. También existe la posibilidad de usar estos datos para otro proyecto exportándolos en formato CSV o JSON.

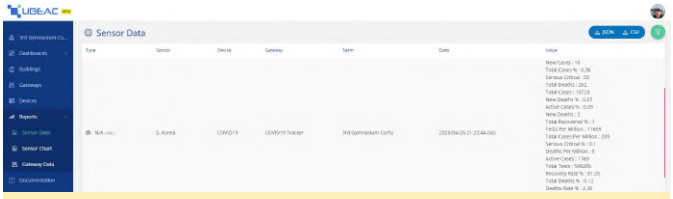


Figura 7: Inspección de los informes históricos de los datos de tu rastreador COVID-19

Últimas palabras

Este es un ejemplo de cómo puede usar ODROID-C2 junto con uBeac para crear y monitorizar las estadísticas de COVID-19. También puedes, como he descrito anteriormente, filtrarlos, manipularlos, visualizarlos y finalmente exportarlos para uso externo en diferentes proyectos. Incluso puedes agregar dispositivos más potentes como ODROID-XU4 (https://bit.ly/2VFP0vM) en el mismo cuadro de mandos

Scripts main_py.py

```
import json
import threading
import http.client
from world_cases_collector import
getting_world_value

# Configuration section
UBEAC_URL = 'hub.ubeac.io'
GATEWAY_URL = 'INSERT GATEWAY URL HERE'
DEVICE_FRIENDLY_NAME = 'World COVID19 Tracker'
SENT_INTERVAL = 900 # Sent data interval in second

day = False
date = input("Update for Today or Yesterday? (T/Y)
: ")
if date == 'T':
day = True
else:
day = False

def main():
threading.Timer(SENT_INTERVAL, main).start()
device_world = [{
'id': DEVICE_FRIENDLY_NAME,
'sensors': getting_world_value(day)
}]

connection =
http.client.HTTPSConnection(UBEAC_URL)
connection.request('POST', GATEWAY_URL,
json.dumps(device_world))
response = connection.getresponse()
print(response.read().decode())

if __name__ == '__main__':
main()

_const_cases.py
# WORLD CASES CONSTANTS
country = 0
w_total_cases = 1
w_new_cases = 2
w_total_deaths = 3
w_new_deaths = 4
w_total_recovered = 5
w_active_cases = 6
w_serious_critical = 7
```

```
w_tot_cases_M = 8
w_deaths_M = 9
w_total_tests = 10
w_tests_M = 11

# JSON CONSTANTS
COUNTRY_OTHER = 'Country'
USA_STATE = 'USA States'
TOTAL_CASES = 'Total Cases'
NEW_CASES = 'New Cases'
TOTAL_DEATHS = 'Total Deaths'
NEW_DEATHS = 'New Deaths'
TOTAL_RECOVERED = 'Total Recovered'
ACTIVE_CASES = 'Active Cases'
SERIOUS_CRITICAL = 'Serious Critical'
TOT_CASES_M = 'Total Cases per Million'
DEATHS_M = 'Deaths per Million'
TOTAL_TESTS = 'Total Tests'
TESTS_M = 'Tests per Million'

# EXTRA JSON CONSTANTS
TOTAL_CASES_PERCENT = 'Total Cases %'
NEW_CASES_PERCENT = 'New Cases %'
TOTAL_DEATHS_PERCENT = 'Total Deaths %'
NEW_DEATHS_PERCENT = 'New Deaths %'
TOTAL_RECOVERED_PERCENT = 'Total Recovered %'
ACTIVE_CASES_PERCENT = 'Active Cases %'
SERIOUS_CRITICAL_PERCENT = 'Serious Critical %'
DEATHS_VS_CASES = 'Deaths Rate %'
RECOVERED_VS_CASES = 'Recovery Rate %'

def get_sensor(id, value, type=None, unit=None,
prefix=None, dt=None):
sensor = {
'id': id,
'data': value
}
return sensor

def get_percentage(str_num, str_dem):
if str_dem == '0':
return '0'
percent = float(str_num) / float(str_dem) * 100
return str(float("{:.2f}".format(percent)))

world_cases_collector.py
from bs4 import BeautifulSoup as bf
import requests
import _const_cases as const

num_places = 220 #number of countries

def getting_world_value(today): #getting the value
from website
data_list = []
html =
requests.get("https://www.worldometers.info/corona
virus")
soup = bf(html.text,'html.parser')
if today:
tag = soup("tr")[9:9 + num_places]
else:
tag = soup("tr")[239:239 + num_places]

def extract_vals(arr):
temp_list = []
arr_size = len(arr) - 2
for j in range(arr_size):
if j == 1:
temp_list.append(arr.contents[j].contents[0].conte
nts[0])
elif j % 2 == 1:
value = arr.contents[j].contents
if len(value) == 0:
value.append('0')
value = value[0]
value = value.replace("
", "")
value = value.replace("+", "")
value = value.replace(",", "")
value = value.replace("N/A", "")
if len(value) == 0 or value == ' ':
value = '0'
temp_list.append(value)
return temp_list

compare_list = extract_vals(tag[-1])

for i in range(len(tag)):
insert_list = extract_vals(tag[i])

def continents(arg, day):
if day:
switcher = {
212: 'North America',
213: 'Europe',
214: 'Asia',
215: 'South America',
216: 'Oceania',
217: 'Africa',
218: 'Unknown',
219: 'World',
}
else:
switcher = {
211: 'Asia',
```

```
212: 'North America',
213: 'Europe',
214: 'South America',
215: 'Oceania',
216: 'Africa',
217: 'Unknown',
218: 'World'
}
return switcher.get(arg,
insert_list[const.country])

data_name = continents(i, today)
data = {
const.TOTAL_CASES :
insert_list[const.w_total_cases],
const.NEW_CASES : insert_list[const.w_new_cases],
const.TOTAL_DEATHS :
insert_list[const.w_total_deaths],
const.NEW_DEATHS :
insert_list[const.w_new_deaths],
const.TOTAL_RECOVERED :
insert_list[const.w_total_recovered],
const.ACTIVE_CASES :
insert_list[const.w_active_cases],
const.SERIOUS_CRITICAL :
insert_list[const.w_serious_critical],
const.TOT_CASES_M :
insert_list[const.w_tot_cases_M],
const.DEATHS_M : insert_list[const.w_deaths_M],
const.TOTAL_TESTS :
insert_list[const.w_total_tests],
const.TESTS_M : insert_list[const.w_tests_M],
const.TOTAL_CASES_PERCENT :
const.get_percentage(insert_list[const.w_total_cas
es],compare_list[const.w_total_cases]),
const.NEW_CASES_PERCENT :
const.get_percentage(insert_list[const.w_new_cases
],compare_list[const.w_new_cases]),
const.TOTAL_DEATHS_PERCENT :
const.get_percentage(insert_list[const.w_total_dea
ths],compare_list[const.w_total_deaths]),
const.NEW_DEATHS_PERCENT :
const.get_percentage(insert_list[const.w_new_death
s],compare_list[const.w_new_deaths]),
const.TOTAL_RECOVERED_PERCENT :
const.get_percentage(insert_list[const.w_total_rec
overed],compare_list[const.w_total_recovered]),
const.ACTIVE_CASES_PERCENT :
const.get_percentage(insert_list[const.w_active_ca
ses],compare_list[const.w_active_cases]),
const.SERIOUS_CRITICAL_PERCENT :
const.get_percentage(insert_list[const.w_serious_c
ritical],compare_list[const.w_serious_critical]),
const.DEATHS_VS_CASES :
const.get_percentage(insert_list[const.w_total_dea
ths],insert_list[const.w_total_cases]),
const.RECOVERED_VS_CASES :
const.get_percentage(insert_list[const.w_total_rec
overed], insert_list[const.w_total_cases])
}
data_list.append(const.get_sensor(data_name,
data))
return data_list
```

Dron Autónomo: Surca los Cielos con tu ODROID-XU4

May 1, 2020 By Yehonathan Litman ODROID-XU4, Mecaniqueo

Este tutorial te enseñará a montar un dron con capacidad de autonomía guiado por Pixhawk. El proyecto consta de lo siguiente:

- ODROID XU4 con tarjeta SD de 32GB (con Ubuntu 18.04)
- Módulo WiFi ODROID 5
- FCU 3DR Pixhawk
- Módulo de potencia Pixhawk
- Módulo USB-TTL
- Cámara de seguimiento Intel Realsense T265
- Estructura Q330 UAV
- 4 Motores RS2203 2300KV
- 4 ESCs con calificación 25A
- Batería de 5500 mAh 3S LiPo
- Repuesto de conector de alimentación de 2.1 mm x 5.5 mm
- Receptor de radio FS-IA6B
- Controlador Flysky i6
- PDB
- Cinta de montaje de doble cara

Cableado y ensamblaje

Figura 1: Todos los materiales utilizados en este proyecto antes del montaje

Lo importante de nuestra lista es que la mayoría de los materiales necesarios se pueden usar para montar un simple dron. El controlador Pixhawk está diseñado para comunicarse con cualquier ordenador

compatible con Linux sin importan los periféricos conectados, por lo que podrías usar cualquier radio, batería, combinación ESC/motor y estructura que quisieses. Esta universalidad es bastante potente, de modo que no necesitas necesariamente limitarte a los materiales que he enumerado con anterioridad.

Nuestro primer paso sería ensamblar el dron. Lo ensamblaremos con una configuración "Quad X", que es la configuración de dron más simple disponible en Pixhawk. La Figura 2 muestra cómo debes configurar las conexiones del dron y dónde debes conectar las señales ESC a las salidas del Pixhawk.

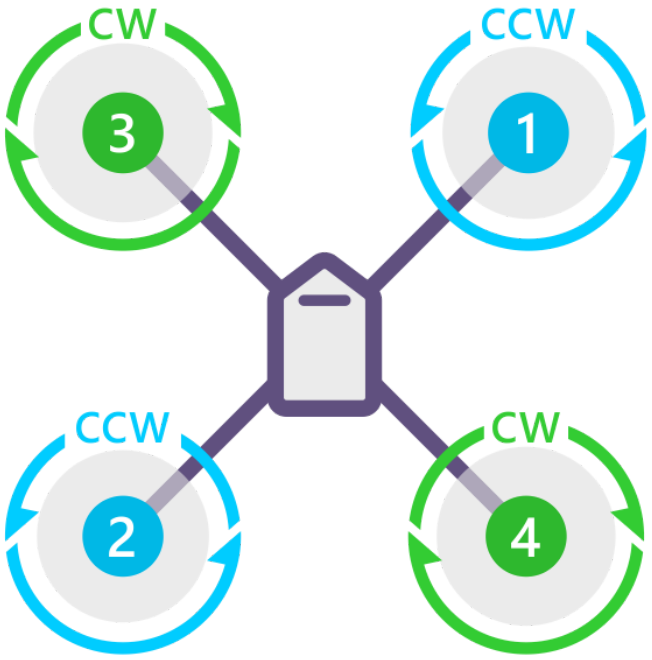


Figura 2 - Quad X con direcciones de motor asociadas y la numeración

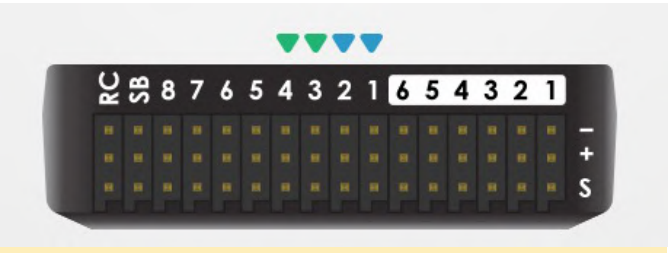
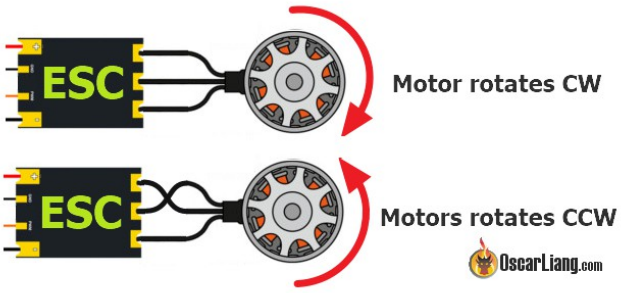


Figura 3 - Pines de salida de Pixhawk (numerados). Los primeros 4 pines están codificados por colores para conectar una estructura Quad X

Tras ensamblar la estructura y atornillar los motores, puedes soldar los ESC al PDB (que es parte de la estructura Q330) y a los motores. La Figura 4 muestra el cableado asociado a las diferentes direcciones del sping.



FFigura 4: ESC hacia las correspondientes conexiones del cable del motor para las diferentes direcciones de giro

Ahora conecta el receptor de radio a Pixhawk. Pixhawk que interpreta las señales como SBUS, configuré mi receptor FS-IA6B para emitir a través de una única línea de datos.

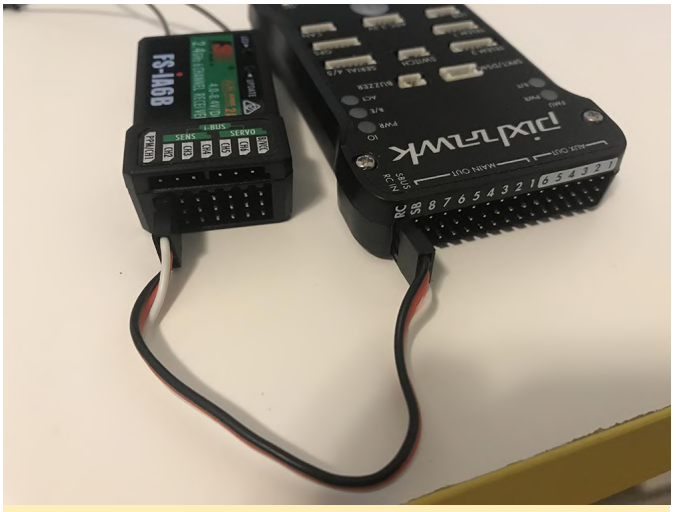


Figura 5 - Enlace FS-IA6B a Pixhawk

Ahora soldamos el conector jack de repuesto al módulo de alimentación Pixhawk, que tiene un 5EC 3A BEC que usaremos para alimentar el ODROID integrado. La soldadura se muestra en la Figura 6. La prueba de la salida de voltaje con el voltímetro muestra 5.28V, un voltaje seguro para alimentar nuestro ODROID.

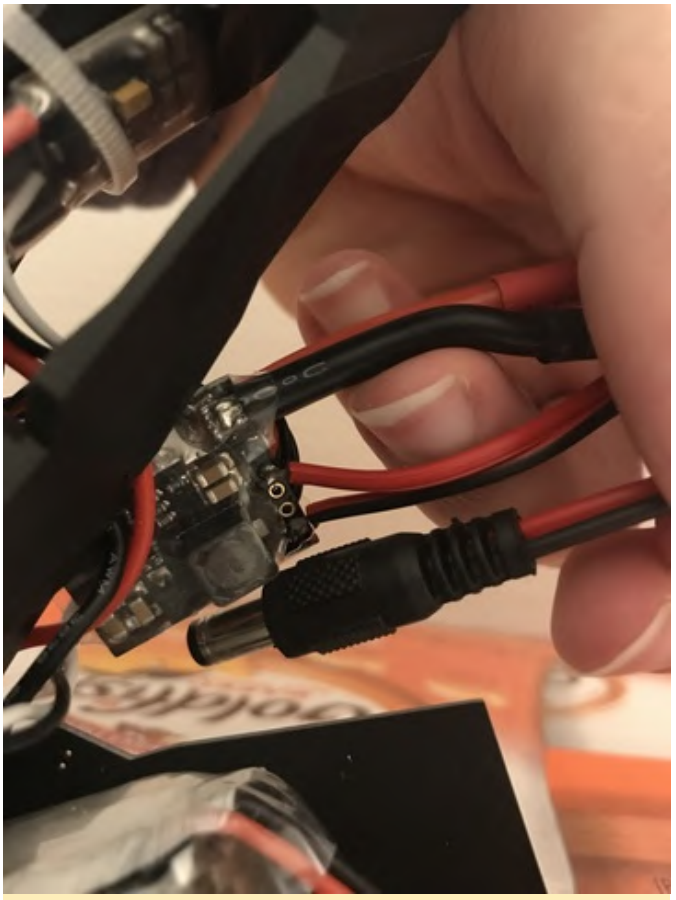


Figura 6: Conector jack para soldadura BEC

Figura 7 - Salida de voltaje del BEC tras encender el circuito

La última soldadura será el convertidor USB-TTL, que se usará para la comunicación entre ODROID y la FCU Pixhawk, la cual se muestra en la Figura 8. Puedes encontrar un esquema en **https://ardupilot.org/dev/_images/ODroid_Pixhawk_Wiring.jpg**.

Figura 8: Conexión de datos USB-TTL a Pixhawk

Ya hemos terminado con la parte tediosa del ensamblaje, y es hora de montarlo todo. Yo usé una cinta de montaje de doble cara para que la FCU y la cámara no se vieran afectadas por las vibraciones del quadrotor. Debido a la falta de espacio, decidí montar la cámara en la Pixhawk FCU en la parte superior del ODROID. Esta no es la mejor forma, pero funciona bien, aunque el problema podría mitigarse fácilmente utilizando una FCU más pequeña (como Pixracer) o extensiones impresas en 3D.

Figura 9 - Montaje de la cinta de doble cara por debajo del ODROID-XU4

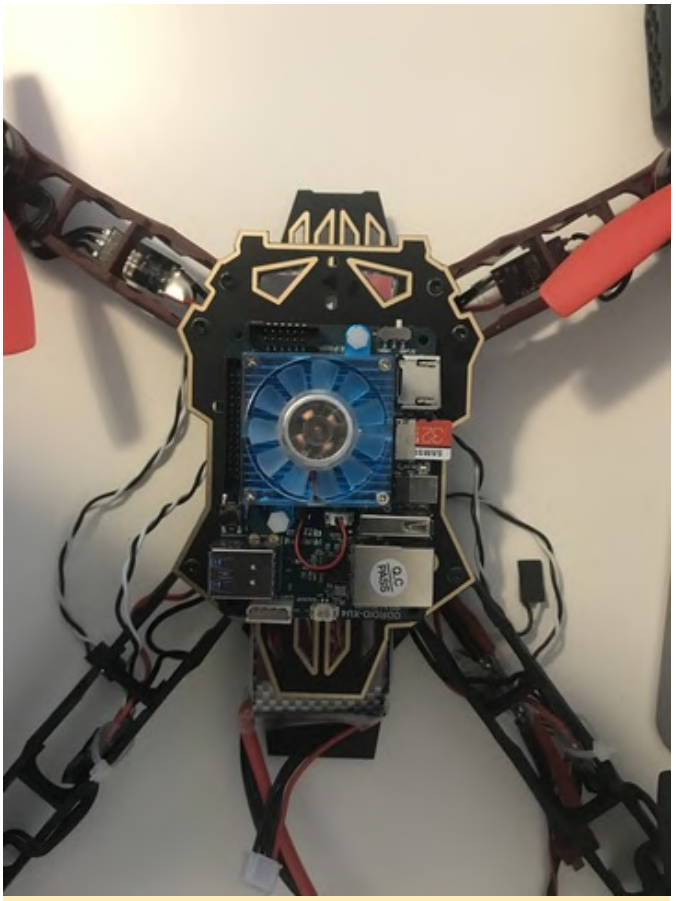


Figura 10 - El ODROID montado en la estructura Q330

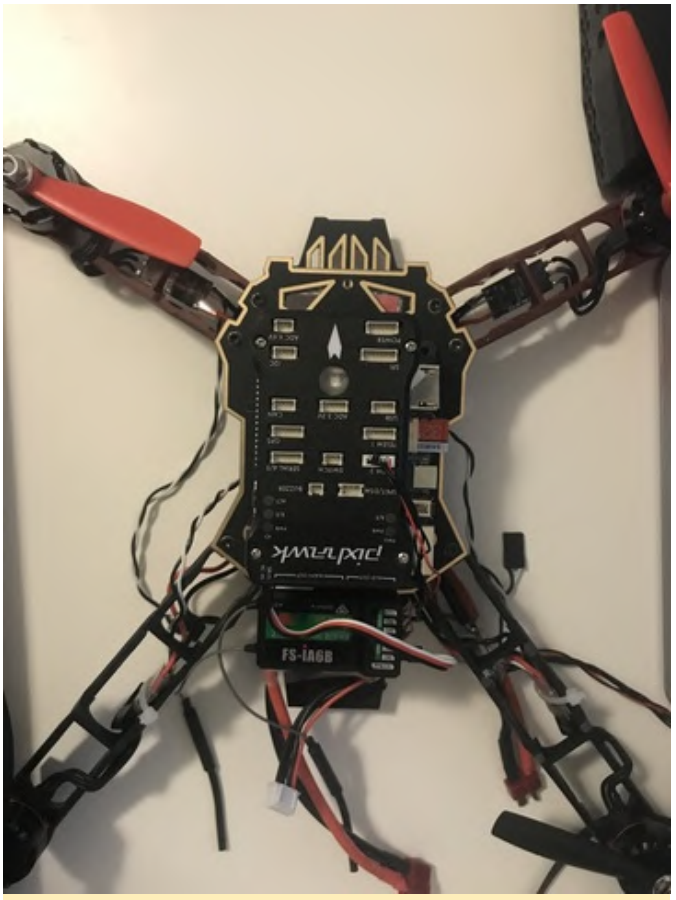


Figura 11 - La FCU Pixhawk montada sobre ODROID. Ten en cuenta que el FCU Pixhawk fue nivelado W.R.T. para la estructura Q330

Figura 12 - Receptor, conector, USB-TTL, adaptador WiFi y salidas ESC conectadas

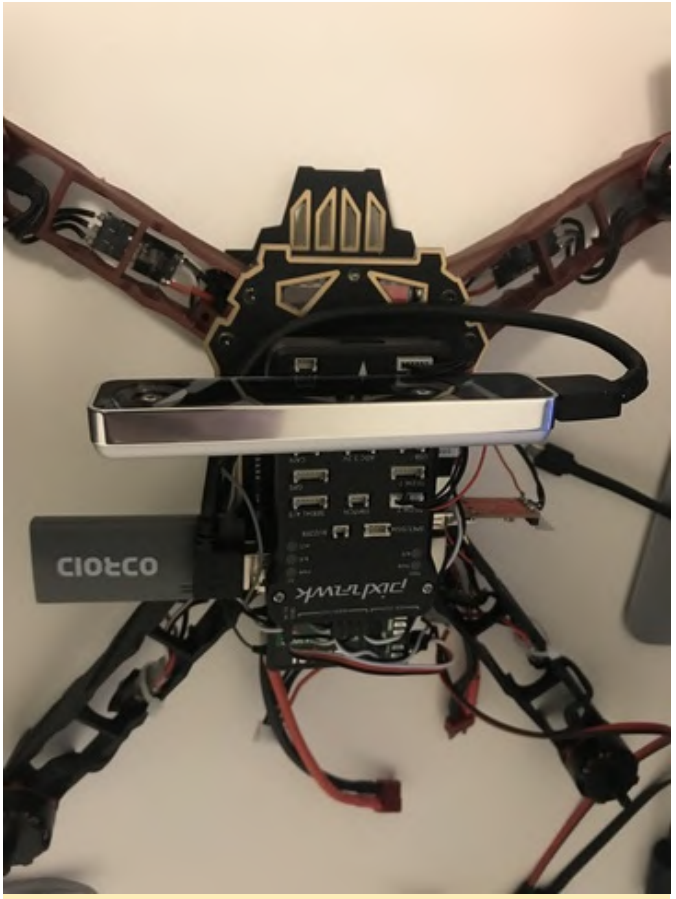


Figura 13: Cámara Realsense T265 montada en la parte superior de Pixhawk FCU y conectada al ODROID

Configuración del software ODROID

Nuestro ensamblaje ya está completo, es hora de configurar el software. Afortunadamente, es muy simple y se reduce a instalar paquetes y verificar nuestros dispositivos/conexiones:

1. Instalar ROS Melodic desde http://wiki.ros.org/melodic/Installation/Ubuntu 2. Instalar [librealsense desde https://github.com/IntelRealSense/librealsense/blob/master/doc/installation.md. (asegúrate de seguir el paso 5) 3. instalar ros-melodic-ddynamic-reconfigure a través de apt 4. Instalar realsense-ros desde https://github.com/IntelRealSense/realsense-ros 5. Instalar mavros y mavlink:

```
$ sudo apt-get install ros-kinetic-mavros ros-kinetic-mavros-extras
$ wget https://raw.githubusercontent.com/mavlink/mavros/master/mavros/scripts/install_geographiclib_datasets.sh && ./install_geographiclib_datasets.sh
```

6. Prueba que la cámara T265 funciona (conecta ODROID a una pantalla y ejecuta `realsense-viewer` desde el terminal) 7. Comprueba que la cámara T265 también funcione en ROS:

```
$ roslaunch realsense2_camera rs_t265.launch
```

Debería ver mensajes de odometría entrantes a una velocidad de ~200 Hz

```
$ rostopic hz /camera/odom/sample
subscribed to [/camera/odom/sample]
average rate: 199.868
min: 0.001s max: 0.012s std dev: 0.00130s window: 189
average rate: 199.845
min: 0.000s max: 0.044s std dev: 0.00947s window: 389
average rate: 199.574
min: 0.000s max: 0.044s std dev: 0.01103s window: 585
```

8. Instala un paquete de conversión desde https://github.com/thien94/vision_to_mavros para que las coordenadas de la cámara a la FCU sean correctas.

9. Crea un punto de acceso desde el módulo WiFi de ODROID y reinicia para habilitarlo:

```
$ git clone https://github.com/oblique/create_ap
$ cd create_ap
$ sudo make install
$ systemctl start create_ap
$ systemctl enable create_ap
$ create_ap wlan0 eth0 odroid_drone --mkconfig
/etc/create_ap.conf
```

Configurar FCU Pixhawk

Llegados a este punto, nuestra configuración de ODROID está completa y necesitamos configurar la FCU Pixhawk con QGroundControl. Puede descargar AppImage en tu propio ordenador y tras conectarte a la FCU, sigue estos pasos para configurar correctamente Pixhawk para aceptar datos de posicionamiento de visión:

1. Instalar el último firmware 2. Elegir el tipo de estructura (en mi caso, elegiré Quadrotor X - DJI Flame Wheel F330 ya que es la más similar al Q330) 3. Calibrar los sensores FCU 4. Calibrar tu controlador de radio después de conectarlo al receptor 5. Calibrar la batería de LiPo y los ESC. Fijar los valores correctos en Número de celdas 6. Asignar canales de radio en modos de vuelo. Configuré el mío como en la Figura 14 7. Ahora vamos a la sección de Parámetros:

- Cambié `CBRK_IO_SAFETY` para deshabilitar la verificación de seguridad, aunque esto es opcional
- Cambié `PWM_MIN` por un valor razonable (lo configuré en 1050 us)
- Ajusta `MAV_0_CONFIG` en" TELEM 2 ",` MAV_0_MODE` en "Onboard"
- Desmarca "use GPS" en `EKF2_AID_MASK` y marca "vision position fusion" y "vision yaw fusion"
- Cambia `EKF2_HGT_MODE` a" Vision "
- Ajusta los parámetros `EKF2_EV_POS_X`,` EKF2_EV_POS_Y`, `EKF2_EV_POS_Z` en consecuencia
- Por seguridad, cambia `COM_OBL_ACT` a "Terminate" y `COM_OBL_RC_ACT` a "Terminate" en caso de que haya algún tipo de error en el enlace de datos ODROID-Pixhawk

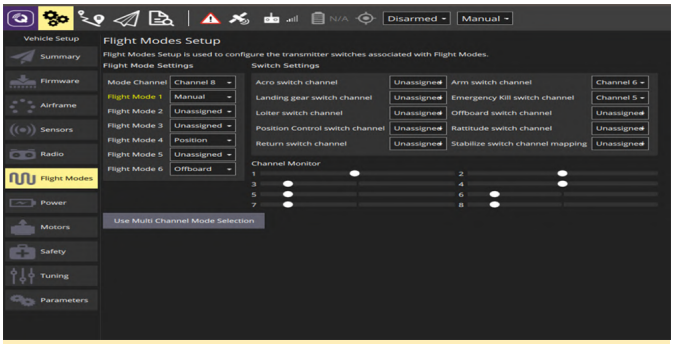


Figura 14: Configuración del canal de mi controlador de radio

También asigné un interruptor manual para emergencias. Los interruptores de modo Manual, Position y Offboard han sido asignados para una transición más suave en la autonomía.

Cogiendo altura

Ahora echaremos nuestros propios datos a volar. Clonamos un paquete personalizado llamado my_autonomous_drone de https://github.com/yehonathanlitman/my_autonomous_drone y recompilamos nuestro espacio ROS. Deberás ejecutar lo siguiente para localizar tu convertidor USB-TTL:

```
$ ls /dev/tty* | grep USB
```

Luego, cambia launch/px4.launch para seguir este formato:

```
<!--?xml version="1.0"?-->

<span style="font-weight: 400;">    </span>

<span style="font-weight: 400;">    </span>

<span style="font-weight: 400;">    </span>

<span style="font-weight: 400;">    </span>

<span style="font-weight: 400;">    </span>

<span style="font-weight:
400;">            </span>

<span style="font-weight:
400;">            </span>

<span style="font-weight:
400;">            </span>
```

```
<span style="font-weight:
400;">            </span>

<span style="font-weight:
400;">            </span>

<span style="font-weight:
400;">            </span>

<span style="font-weight:
400;">            </span>

<span style="font-weight: 400;">    </span>
```

Ahora crea un directorio src en el paquete my_autonomous_drone y agrega un offb_node.cpp (https://dev.px4.io/v1.9.0/en/ros/mavros_offboard.html). Asegúrate de añadir el archivo a la compilación del paquete para que pueda compilarse. Después de encender el dron desde la batería, conéctate por SSH al punto de acceso que creaste y sigue estos pasos:

1. Ejecuta "roslaunch my_autonomous_drone px4.launch" para empezar la transmisión de datos 2. Verifica que la FCU esté conectada con " rostopic echo /mavros/state " 3. Inicia el nodo Realsense con "roslaunch realsense2_camera rs_t265.launch" 4. Ejecuta "roslaunch vision_to_mavros t265_tf_to_mavros.launch" para la conversión del marco de coordenadas 5. El último paso es ejecutar "rosrun my_autonomous_drone offb_node" para comenzar a enviar el waypoint de posición a Pixhawk. Las moscas del dron se muestran en la Figura 15.

Figura 15: ¡Nuestro dron ODROID suspendido a 1 metro del suelo!

Si quieres que el dron haga algo más impresionante, puedes jugar con las coordenadas x e y en offb_node.cpp. Por ejemplo, se puede hacer una figura de ocho usando ecuaciones paramétricas (https://mathworld.wolfram.com/EightCurve.html). Dentro del ciclo while podemos hacer esto, donde la variable "i" se inicializa a 0 antes de que comience el ciclo:

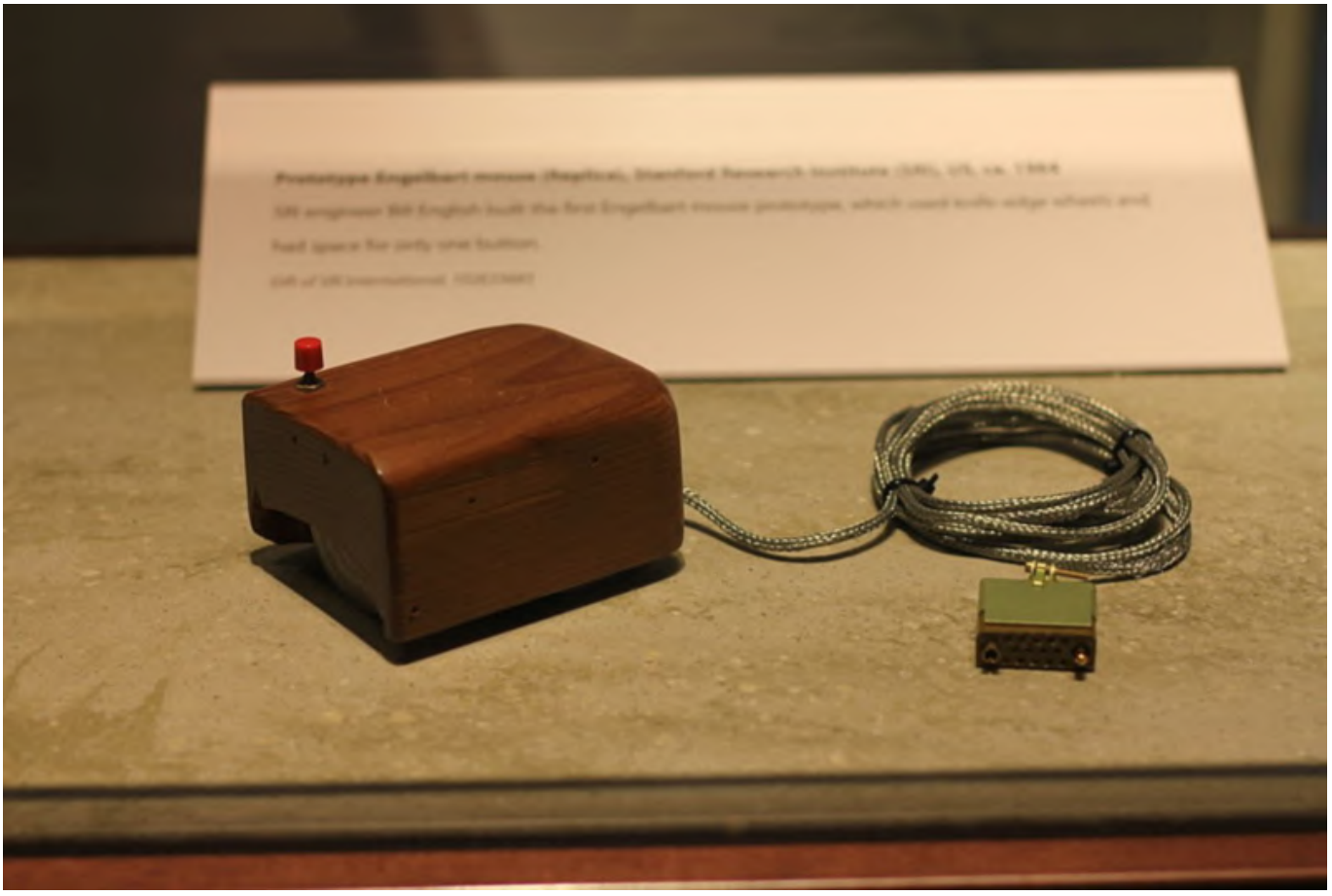
```
double t = i * 0.02;
x = a * sin(t);
y = a * sin(t) * cos(t);
i++;
```

Conclusión

En esta guía, te he mostrado cómo ensamblar un dron, conectar tu controlador de vuelo a ODROID y enviarle simples comandos de guía en ROS. Esto es solo la punta del iceberg de un campo súper interesante, pero con un poco de trabajo duro puedes comenzar a hacer cosas realmente increíbles. Si desea más, puede consultar mi canal de YouTube (youtube.com/c/SimpleKernel) donde he subido algunos videos de instrucción más detallados sobre autonomía usando Pixhawk y sobre como configurar su propia guía visual desde cero.

Cómo Añadir un Ratón y un Teclado a tu ODROID-GO Advance: Montando el Mejor Ordenador Go

May 1, 2020 By @mameise ODROID-GO Advance, Mecaniqueo



Pensé que un pequeño teclado con un dispositivo analógico conectado al ODROID-GO Advance, similar al teclado para el clásico ODROID-GO, sería un gran proyecto. El teclado podría conectarse a través de USB, aunque se necesitaría un Chip Hub USB para que también se pudiera conectar un módulo WiFi USB.

La imagen en tenía en mente no era montar un teclado de nivel profesional, ya que no tengo experiencia en la construcción de PCB. En cambio, cogería un Arduino Micro, ya que tiene un chip que se puede identificar como teclado y ratón, un pequeño PCB USB Hub, una pequeña memoria analógica y muchas teclas. El N900 tiene un diseño de teclas de 13x3 y parecía acoplarse muy bien, pero necesitaba ayuda en pantalla. Así que pensé que un diseño de 4x12 o 5x12 sería mejor, aunque aún no estaba muy seguro del tamaño.

Creé un concepto rápido con un diseño de botones simple para tener una idea de la ubicación y el tamaño en general.

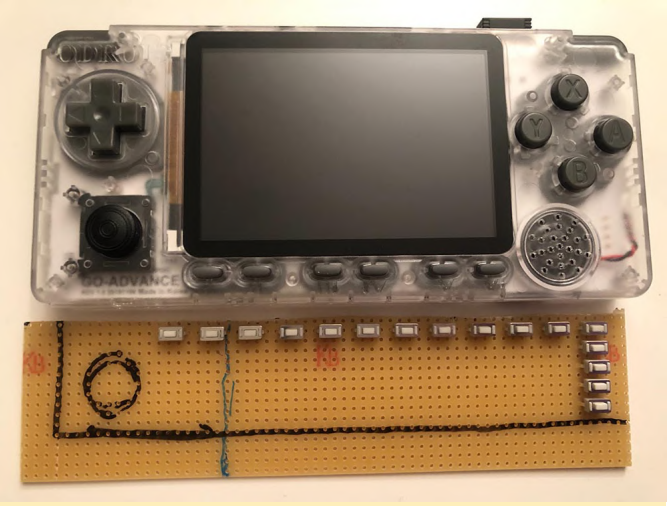


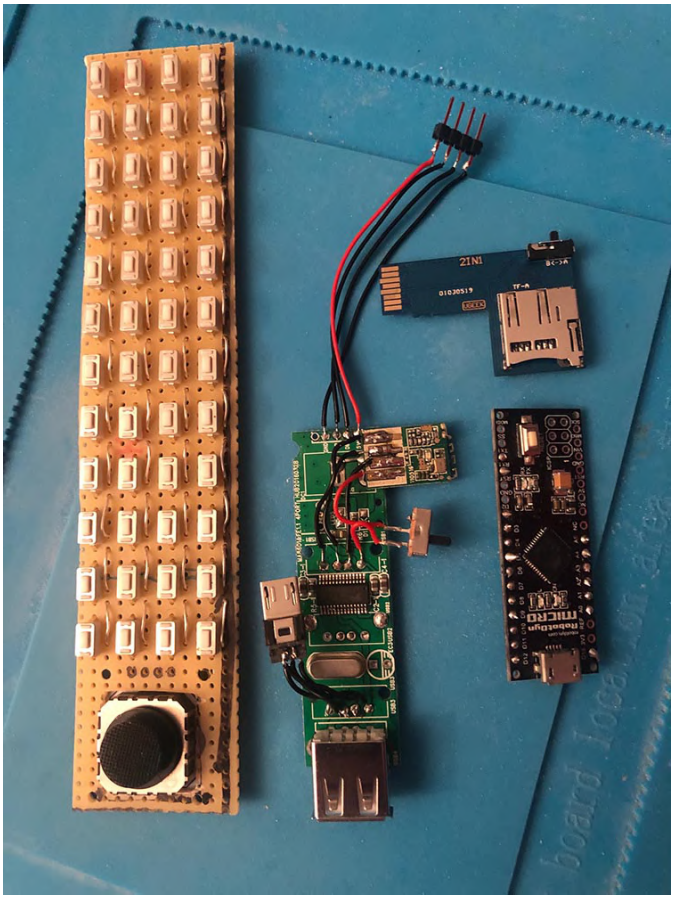
Figura 1 - Concepto de teclado y ratón

El hub USB que encontré tenía un tamaño pequeño y tenía 3 puertos USB y 1 puerto Ethernet.

Figura 2 - El hub USB y el Arduino negro

Figura 3: el hub USB conectado al GO Advance

Basándome en los comentarios del foro, cambié los componentes. Decidí usar un Hub USB de 4 puertos. Junté los botones y completé el trabajo en el centro. Retiré 3 puertos USB y conecté directamente un dongle Wifi con un interruptor de encendido/apagado y en el otro puerto conecté un conector Micro USB para el Arduino. Uno de los puestos no lo usaría y un segundo situado en un lateral permitiría conectar otras cosas. La primera prueba fue exitosa, y redirigí el puerto USB a la parte inferior del GO Advance para permitirme conectar el teclado con más fácilidad. También incluí un módulo Micro-SD 2 en 1 para poder tener 2 imágenes del sistema operativo conmigo y cambiar entre ellas fácilmente.



Para la carcasa, utilicé una impresora 3D FDM, y para las teclas del teclado utilicé una impresora 3D de resina para lograr mejores resultados.

Figura 5 - Las teclas impresas, antes del siguiente paso de pintarlas de negro

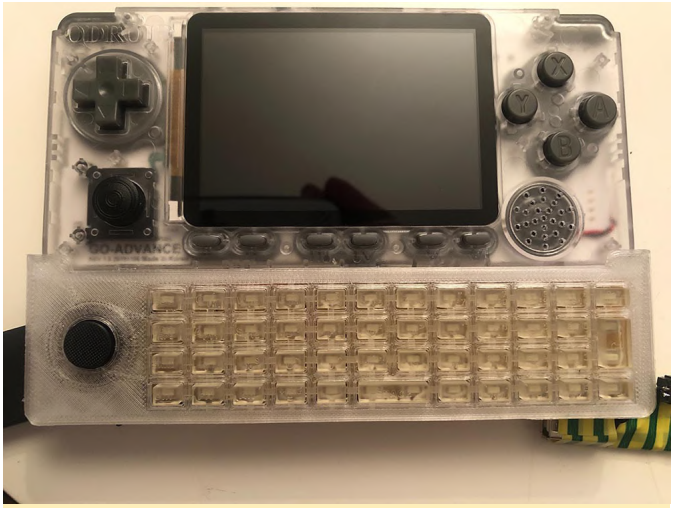


Figura 6 - Carcasa prototipo

Pinté las teclas de negro, aunque aligeré la aplicación de la pintura para mantener las marcas y letras lo más visibles posible. Luego uso cera de vela, que no se derrite en caliente, cojo la vela y la froto sobre las marcas y se elimina todo hasta que solo quede cera en las marcas. La figura 7 lo muestra con un color. A continuación, quise usar 3 colores, lo cual fue complicado.

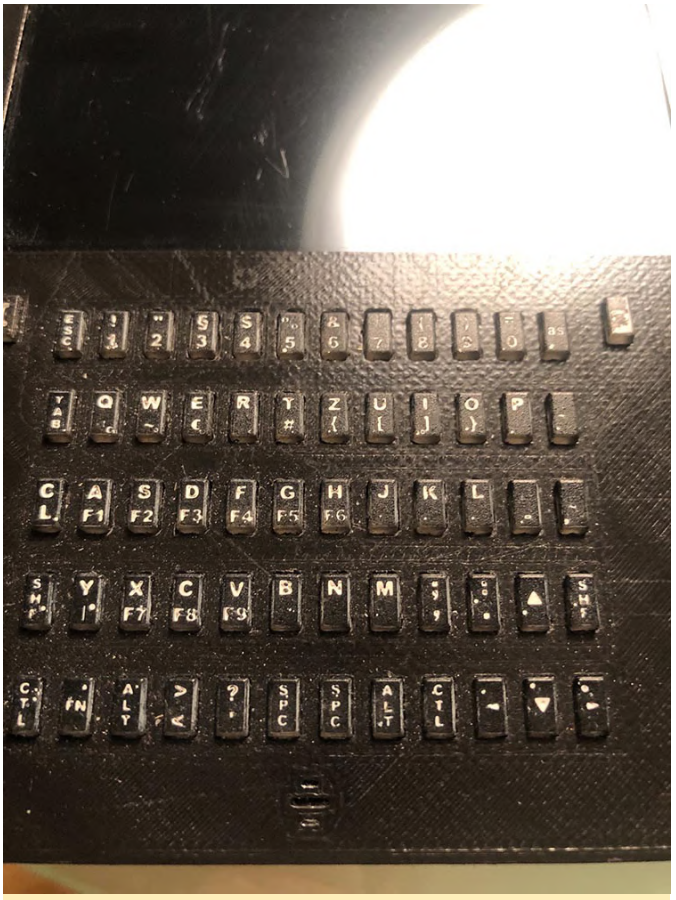


Figura 7 - Teclas de 2 tonos

Después, hice algunas modificaciones en la carcasa y la imprimí en 3D nuevamente, esta vez en negro.

Figura 8 - Segunda impresión de la carcasa

Figura 9 – Nueva carcasas con las teclas insertadas

Mi primera extensión microSD, llegué a dañarla cuando lo monté todo. Afortunadamente compré dos y pude continuar. Además, cambié a un hub USB nano más pequeño, ya que el espacio dentro de la carcasa estaba demasiado limitado. El hub más pequeño, sin embargo, no dispondrá de un conector USB adicional.

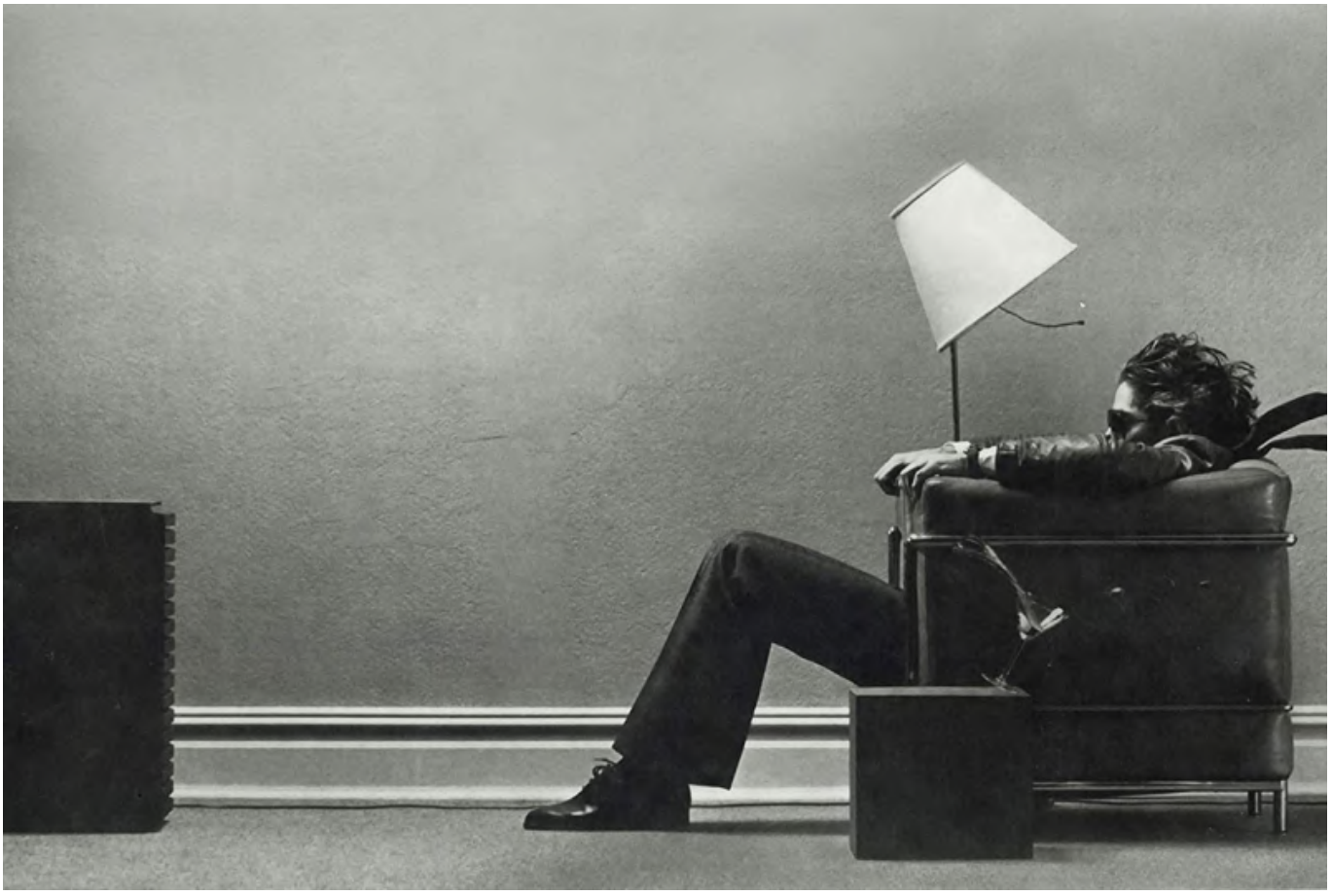
Figura 10 - Teclado no conectado, que muestra la extensión microSD

Figura 11 - Teclado conectado

Más información, y el hilo original del foro Hardkernel, está disponible en https://forum.odroid.com/viewtopic.php?f=187&t=38047.

Presentamos el nuevo ODROID-C4: Un Ordenador de Placa Reducida de Nueva Generación

May 1, 2020 By Justin Lee, CEO of Hardkernel ODROID-C4



El ODROID-C4 es un ordenador de placa reducida de nueva generación que es más eficiente desde el punto de vista energético y su rendimiento es mayor que el ODROID-C2, que se presentó hace cuatro años, como el primer ordenador ARM de 64 bits del mundo a un precio muy asequible. El ODROID-C4 presenta una CPU Amlogic S905X3 que es un clúster Cortex-A55 de cuatro núcleos con una GPU Mali-G31 de nueva generación. Los núcleos A55 funcionan a 2.0Ghz sin estrangulamiento térmico utilizando un disipador de calor de serie, dando lugar a un ordenador recudido y muy silencioso. El rendimiento multinúcleo es aproximadamente un 40% más rápido y el rendimiento de la DRAM del sistema es un 50% más rápido que el ODROID-C2.

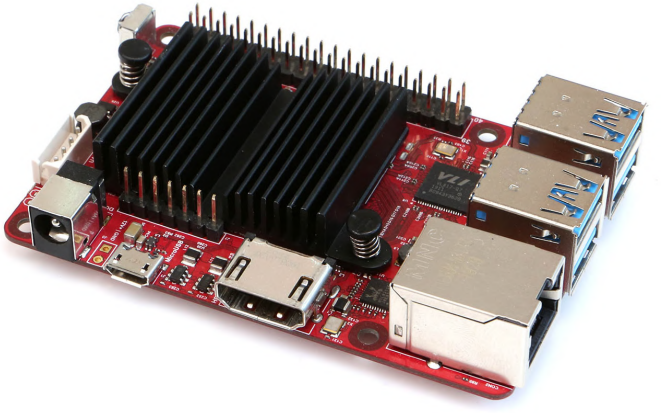


Figura 1 - El nuevo ODROID-C4

Echemos un vistazo al diagrama por bloques y los componentes clave de la placa en las Figuras 2 y 3 para obtener más información sobre las características del hardware.

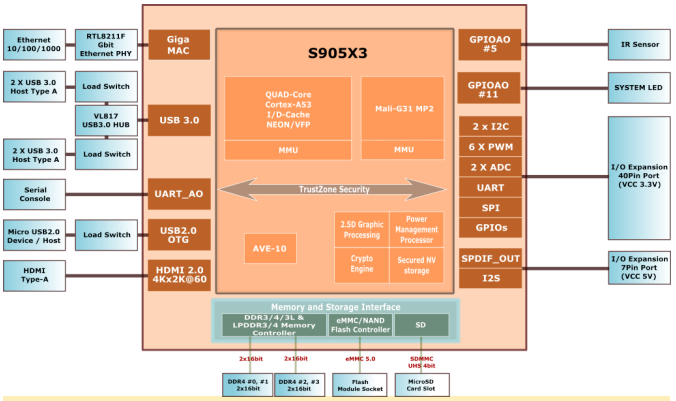


Figura 2 - Diagrama por bloques

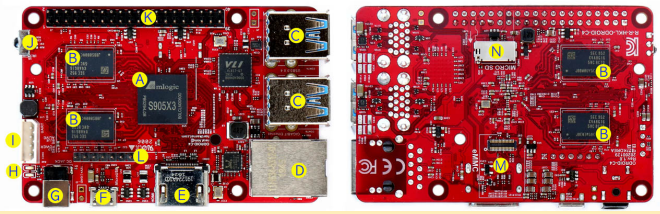


Figura 3 - Diseño de componentes

Tienes información más detallada del hardware disponible en la wiki de ODROID-C4 en https://wiki.odroid.com/odroid-c4/hardware/hardware.

Rendimiento de la CPU

Los resultados de las pruebas de compresión 7-zip, Dhrystone-2, Whetstone de doble precisión, muestran que el rendimiento del sistema ODROID-C4 es 40 ~ 55% más rápido que el ODROID-C2 de la generación anterior.

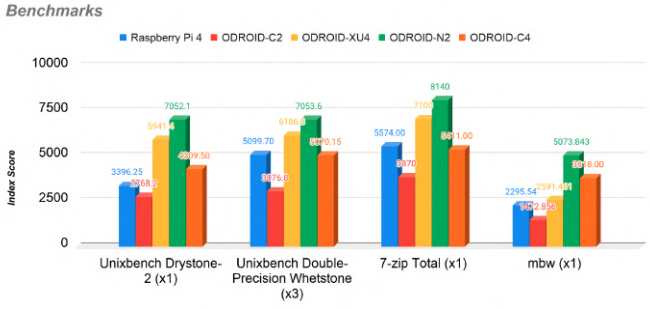


Figura 4 – Pruebas de rendimiento de la CPU

Rendimiento de la GPU

El Mali-G31 funciona a 650MHz y es ~ 50% más rápido que Mali-450MP en ODROID-C2. El Mali-G31 es la primera generación de GPU convencional basada en Bifrost de Arm. El rendimiento de la GPU se midió usando glmark2-es2 "- off-screen".

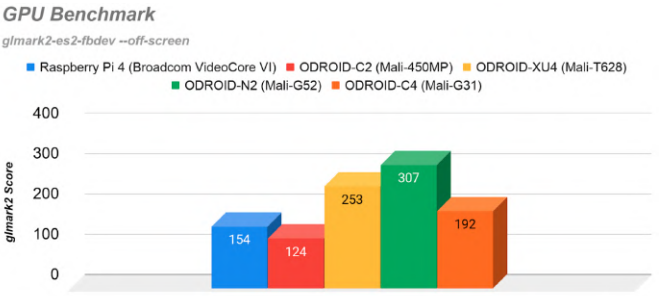


Figura 5 - Pruebas de rendimiento de la GPU

Rendimiento de la RAM

¿Por qué importa el hecho de ser DDR4? El RAM DDR del DROID-C4 funciona a 1320Mhz, con un ancho de banda de memoria que es 1.6 veces mayor que el ODROID-C2.

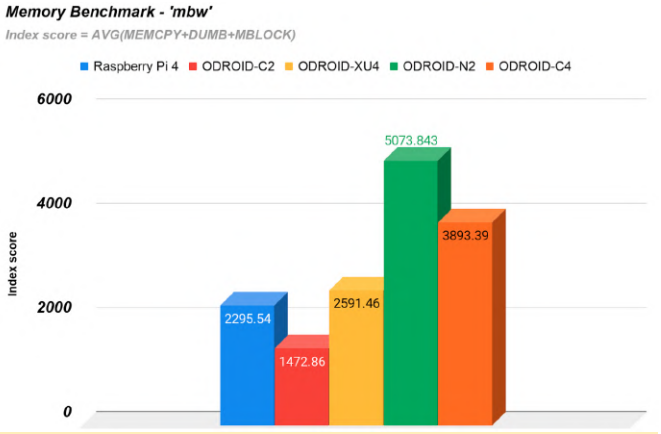


Figura 6 – Pruebas de rendimiento de la RAM

Frecuencia de CPU vs Rendimiento

Algunos usuarios de ODROID pueden tener una velocidad de reloj inferior a la esperada con el S905 de ODROID-C2. Realizamos una prueba para confirmar la relación entre la frecuencia de reloj de la CPU y el rendimiento con ODROID-C4.

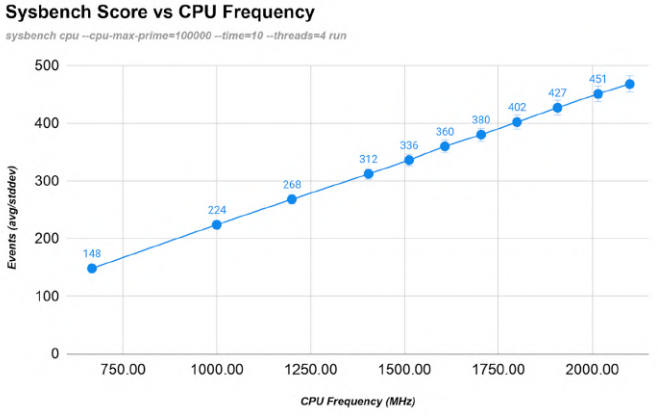


Figura 7 - Frecuencia de CPU vs rendimiento

Características térmicas

Para verificar la regulación térmica, ejecutamos algunas cargas pesadas de CPU y GPU en el SoC y monitorizamos la temperatura dentro de una cámara que mantiene la temperatura ambiente a 25 ° C. Ten en cuenta que el punto de regulación térmica actual está fijada en 75 °C en la configuración del kernel.

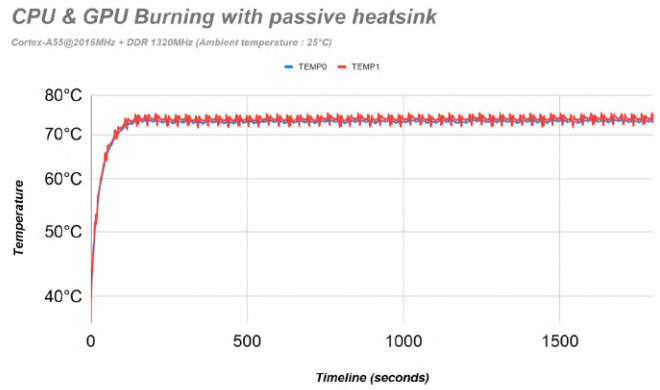


Figura 8 - Temperatura de CPU y GPU con disipador pasivo

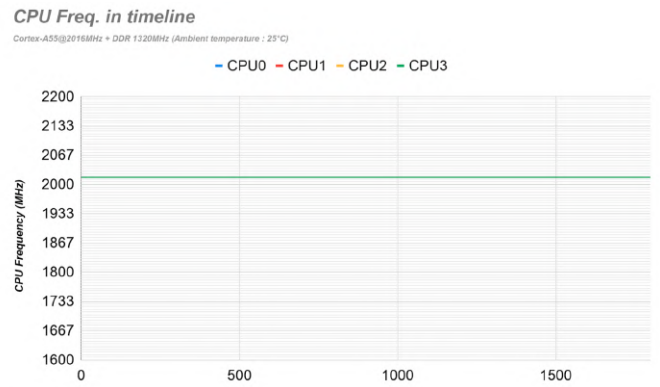


Figura 9 - Frecuencia de CPU en orden cronológico

Si colocas la placa ODROID-C4 dentro de una carcasa, puedes encontrar algunos problemas de estrangulamiento térmico cuando la temperatura ambiente es superior a 20 °C y la carga informática continúa siendo muy alta.

Ethernet

Según el resultado de nuestra prueba iperf, el rendimiento estaba cercano al 1 Gbps.

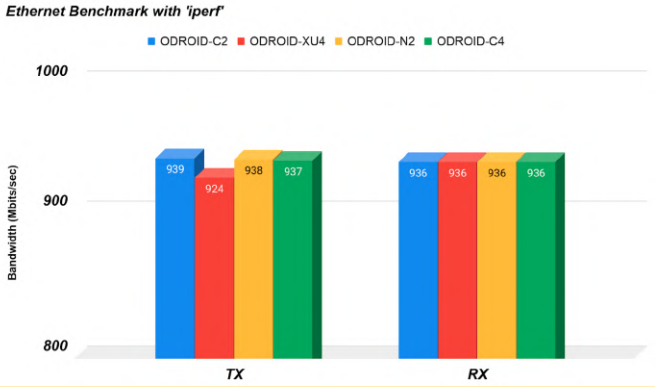


Figura 10 - PReubas de rendiminto Iperf

Puerto USB

Medimos la velocidad de transferencia de USB3 con un SSD USB. El promedio de ~ 340 MB/s de rendimiento debería ser aceptable para la mayoría de las aplicaciones. Puesto que los 4 puertos host USB comparten un único hub raíz, la velocidad de transferencia será menor si usa múltiples dispositivos USB3 al mismo tiempo. También hay un puerto micro USB independiente que admite la interfaz USB 2.0 OTG de doble función.

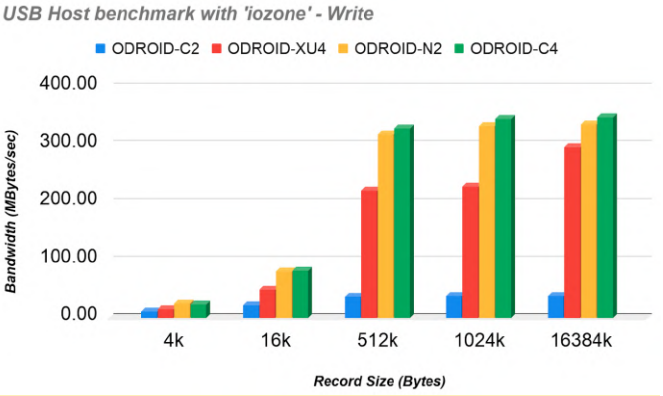


Figura 11 – Prueba de rendimiento del host USB con "iozone" - Escritura

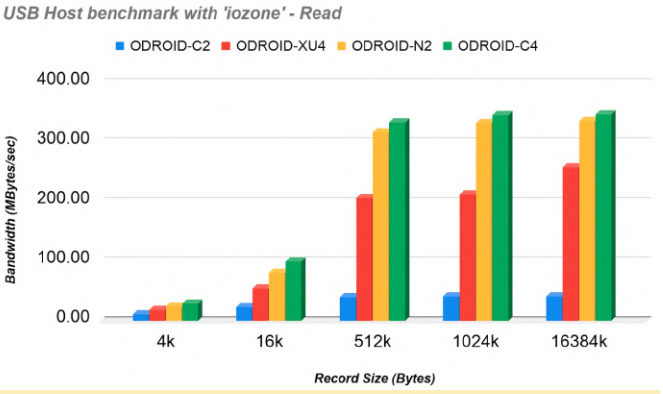


Figura 12 – Prueba de rendimiento del host USB con "iozone" - Lectura

Rendimiento de almacenamiento eMMC

La velocidad de lectura y escritura secuencial es superior a 165 MB/s y 125 MB/s respectivamente. El rendimiento de acceso aleatorio 4K también es razonablemente rápido. Los resultados de la prueba de iozona son los siguientes.

kB	reclen	write	rewrite	read	reread	random read	random write
102400	4	27791	32093	26801	26834	26078	29216
102400	16	72326	77805	69109	68462	67961	77121
102400	512	138415	139953	163901	166665	146408	133777
102400	1024	139408	137395	164144	167444	150278	137524
102400	16384	129097	133087	168813	169074	166081	140086

Figura 13. - Rendimiento del almacenamiento eMMC

Rendimiento de Micro-SD UHS

Usando una escala de voltaje dinámico UHS implementada correctamente, la velocidad de lectura y escritura secuencial es superior a 70 MB/s y 50 MB/s, respectivamente.

kB	reclen	write	rewrite	read	reread	random read	random write
102400	4	3633	3661	16424	16464	16446	3029
102400	16	10747	11056	42683	42727	42799	8337
102400	512	48183	49669	69376	69680	69465	50072
102400	1024	49592	50507	70057	70363	70293	50866
102400	16384	50426	51992	70881	70796	70705	50887

Figura 14 - Rendimiento de MicroSD UHS

Criptografía

La arquitectura ARMv8 admite extensiones criptográficas aceleradas por hardware para construir un sistema seguro. Como era de esperar, pudimos ver un rendimiento de openSSL bastante bueno con el ODROID-C4.

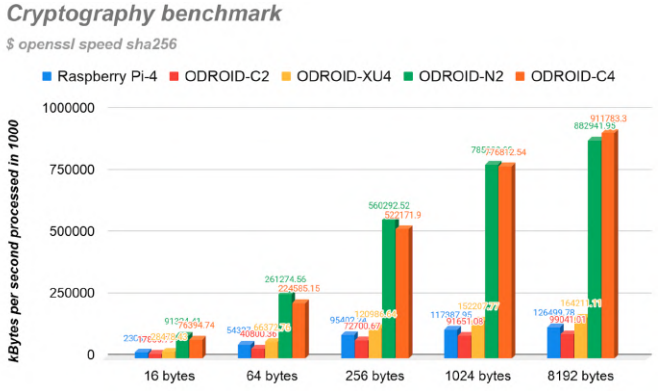


Figura 15 – Pruebas de rendimiento de criptografía

GPIO (cabezal de 40 pines)

La interfaz ODROID-C4 GPIO es similar a la ODROID-C2 y es totalmente compatible con una interfaz de 3,3 voltios. Esto está muy bien para poder usar periféricos sin shifters de nivel complicados según sea necesario con los GPIOs de 1.8 voltios del ODROID-XU4. Otra gran mejora es la interfaz de bus SPI más rápida con una frecuencia máxima de más de 100Mhz. Es significativamente más rápido que el SPI "bit-banged" del software ODROID-C2 de 400Khz.

Figura 16 - cabezal GPIO

El consumo de energía

Gracias a la moderna CPU S905X3 fabricada de 12 nm, el consumo de energía y la disipación de calor son muy bajos, lo que te permite disfrutar de un ordenador silencioso y potente con alta eficiencia energética.

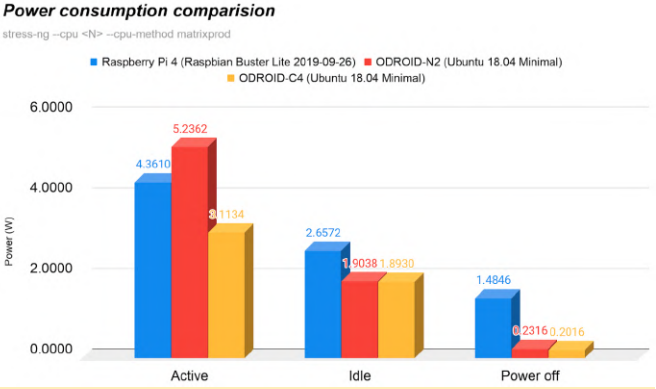


Figura 17 - Comparación del consumo de energía

Estado inactivo: ≃ 0.18 vatios Estado de carga pesada: 3.1 ~ 3.3 vatios (stress-ng --cpu 4 --cpu-method matrixprod) No hay cables conectados, excepto la entrada de alimentación DC y el cable de consola de depuración USB-UART.

El consumo de energía en "IDLE" se mide cuando un dispositivo no se utiliza durante 5 minutos, ya que el regulador de la CPU está configurado en "performance". El consumo de energía medido no es perfecto y puede variar según ciertas condiciones.

Soporte de software de Linux

Existe una imagen **Ubuntu 20.04 LTS (full 64-bit)** con el kernel de Linux versión 4.9.218 LTS a partir del 22 de abril de 2020. Esta versión del kernel LTS tendrá soporte oficial hasta enero de 2023. Ahora contamos con un controlador de decodificador de video acelerado por hardware (VPU). Tenemos ejemplos de c2play y kplayer que pueden reproducir videos 4K/UHD H.265 60fps sin problemas en el framebuffer de la salida HDMI del ODROID-C4. El controlador Mali G31 GPU Linux también funciona solo en el framebuffer.

Upstream Linux kernel 5.4 también está disponible para desarrollo, el cual admite aceleración de GPU ARM Mali. El contenido de WebGL puede ejecutarse en el navegador Firefox (v75 +) utilizando el backend moderno de Wayland/GBM. Sin embargo, la aceleración VPU es un proyecto en curso. El escritorio GNOME Ubuntu 20.04 con tecnología Wayland funciona bastante bien, tal y como se muestra en **https://youtu.be/4MfHMKcHaUc**.

El entorno de trabajo de la interfaz de usuario de Flutter funciona con un kernel 5.4 de Linux y está acelerado por la GPU ARM Mali. Hay un video demostrativo con Ubuntu 20.04 Minimal + Linux kernel 5.4 + Flutter UI + acceso GPIO directo en **https://youtu.be/p6bzmdAJqjo**.

Soporte de software de Android

Android 9 "Pie" de 64 bits está disponible, y hemos lanzado el código fuente completo BSP y una imagen precompilada. Android user land admite aplicaciones de 32 bits y de 64 bits con un controlador de GPU Mali ARM compatible con Vulkan. Otra gran mejora es que admite un entorno de trabajo compatible con AndroidThings, que te proporcionará un entorno de desarrollo fácil para controlar periféricos de hardware en Android utilizando potentes API de Java. El ODROID-C4 no es un dispositivo certificado por Google AndroidThings, y el código fuente de Android de Hardkernel no incluye el código fuente de Google AndroidThings.

En **https://youtu.be/0o-JLCLlGe4**. hay disponible un video con Android 64-bit que incluye una la demostración del controlador de GPU Vulkan con la emulación PPSSPP "God of War". Puede ver una demostración de la programación de IoT de Android con la APIs compatibles con AndroidThings en **https://youtu.be/C5o7JCQXpr8**.

LineageOS

LineageOS 16.0 es otro sistema operativo impulsado por la comunidad, y está disponible a desde el 22 de abril de 2020. LineageOS 17.1 se está desarrollando actualmente, la primera versión estaría disponible para mediados de mayo de 2020

CoreELEC

El equipo de desarrollo de CoreELEC ha creado una increíble imagen del sistema operativo para reproducir contenido 4K/UHD con soporte de color HDR. El equipo ofrece una distribución Linux de "sistema operativo" diseñada para una experiencia óptima de Kodi cuando se ejecuta en el popular hardware Amlogic. Permite la reproducción de video 4K HDR+ Audio Passout, Netflix 1080p e incluso 8K-30FPS H.265 con escalado (**https://youtu.be/7ejYL5OuMi0**).

Disponibilidad y precio

El ODROID-C4 está actualmente disponible para su compra y se aceptan pedidos. Comenzaremos el envío el 28 de abril. El modelo ODROID-C4 4GB tiene un precio de 50$ y está disponible directamente en Hardkernel en https://www.hardkernel.com/shop/odroid-c4/. Otros distribuidores mundiales comenzarán a vender pronto, tal y como se muestra **https://www.hardkernel.com/distributors/**.

¿Dónde está el ODROID-C3?

Desarrollamos internamente el ODROID-C3 basado en la CPU S905X2, que tiene núcleos ARM Cortex-A53, hace casi dos años. Sin embargo, el rendimiento no fue lo suficientemente bueno, y habíamos oído hablar

del nuevo S905X3 con núcleos ARM Cortex-A55 modernos. Por lo tanto, decidimos omitir el ODROID-C3.

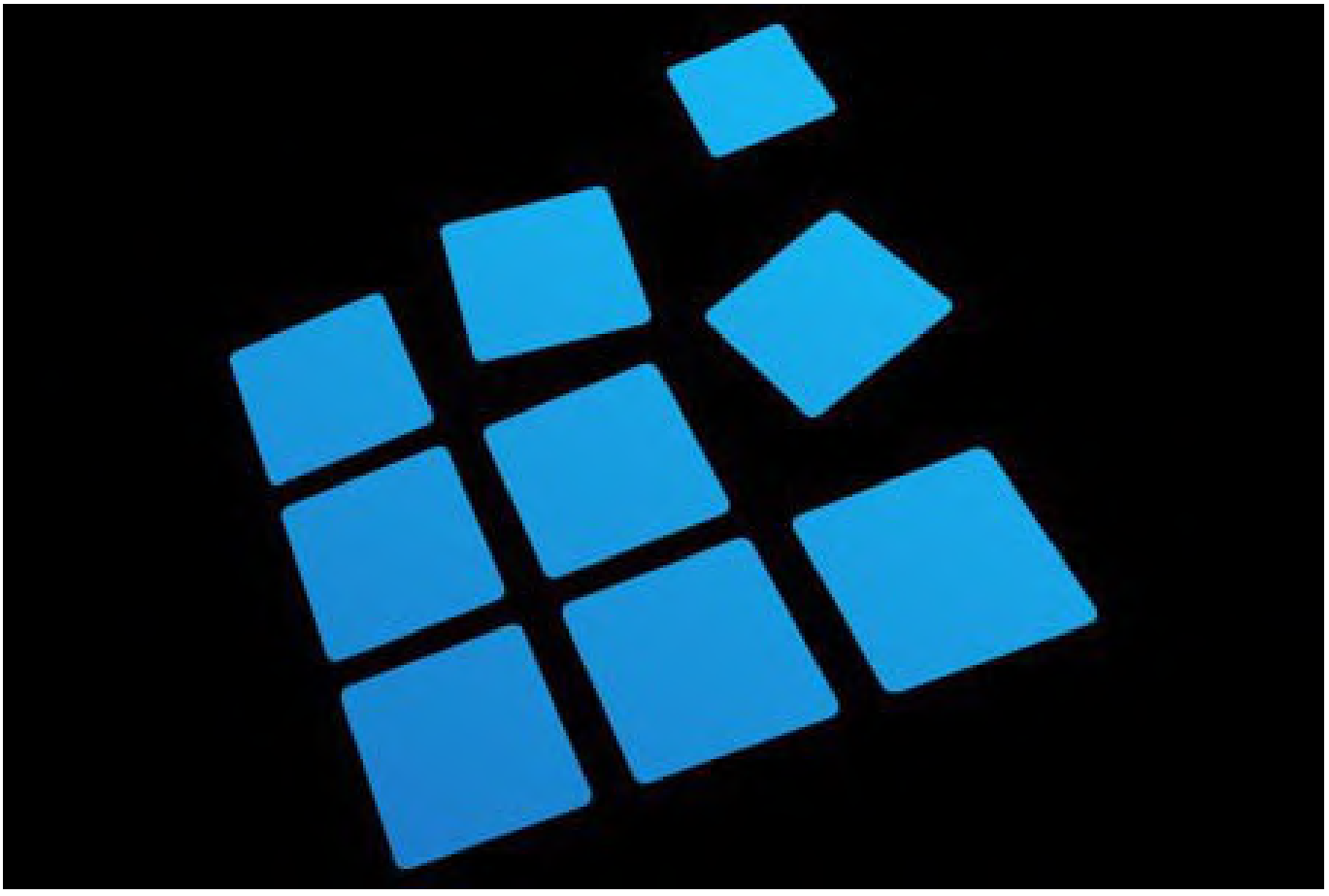
¿Qué pasa con el ODROID-C2/C1+?

Intentaremos continuar con la producción de ODROID-C2 / C1 + durante el mayor tiempo posible, ya que todavía hay muchos clientes B2B que continúan comprándolos en cantidad. Sin embargo, Amlogic está descontinuando las antiguas CPU S905 y S805 en un futuro cercano, y como resultado, probablemente tendremos que descontinuar ODROID-C2/C1+ a principios del próximo año. Considera la posibilidad de migrar a la nueva plataforma ODROID-C4 lo antes posible.

Para ver el anuncio original, visita el artículo de la Wiki de Hardkernel en **https://wiki.odroid.com/odroid-c4/odroid-c4**.

Juegos Linux en ODROID: Box86 - Parte 2

May 1, 2020 By Tobias Schaaf ODROID-C2, ODROID-N2, Tutoriales



Hace aproximadamente un año, escribí sobre box86, un emulador i386 para ARM desarrollado por @ptitSeb, quien también es responsable del asombroso empaquetador gl4es para OpenGL → OpenGL ES. Aunque el aspecto original de hace un año ya era impresionante, quisiera volver a verlo y señalar lo que ha cambiado desde entonces y lo que se puede hacer ahora con él.

Requisitos

Actualmente aun lo estoy probando en mi vieja imagen ODROID GameStation Turbo basada en Debian Jessie, con box86-odroid, libgl-odroid y monolibs-odroid instalados. Cada uno de ellos se utiliza de una forma diferente para mejorar la experiencia general y proporcionar los controladores necesarios para ejecutar los juegos. Todo lo anterior se instalará a la vez, si instalas box86-odroid.

Antecedentes

Algunas personas pueden estar familiarizadas con un software llamado ExaGear. Era un emulador comercial x86 (i386) para dispositivos ARM y ARM64, que te permitía ejecutar el software i386. Aunque el rendimiento en general era bastante bueno, carecía de soporte 3D, si tu plataforma no contaba con controladores GPU x86, que normalmente no es el caso, excepto la RPi que puede usar controladores MESA, y para esto puedes ejecutar OpenGL en i386 de forma limitada. Además, la mayoría de los juegos x86 requieren OpenGL, que no es compatible con gran parte de los SoCs ARM. Por lo tanto, el soporte para juegos x86 estaba limitado a juegos y aplicaciones 2D que no requerían ninguna aceleración de hardware o herramientas de línea de comandos.

Con box86, @ptitSeb tomó un enfoque diferente. No solo escribió una emulación de CPU para x86, sino que también implementó la posibilidad de redirigir llamadas del entorno x86 al entorno host ARM, utilizando librerías ARM nativas en lugar de usar

librerías x86 emuladas. Primero lo implementó con soporte OpenGL, que en combinación con gl4es nos permitió ejecutar aplicaciones y juegos que requieren aceleración de hardware. También permite reenviar muchas llamadas del sistema desde aplicaciones directamente al sistema host en lugar de intentar emular estas llamadas en un entorno x86 emulado, que es mucho más rápido que una simple emulación.

Cambios en conjunto

@ptitSeb trabaja constantemente en box86 para mejorar la compatibilidad y el rendimiento, pero también trabaja duro en gl4es (libgl-odroid) para soportar este proyecto. La combinación de estos dos ha mejorado mucho en los últimos meses, lo que nos permite ejecutar más aplicaciones y juegos que solo están disponibles en x86 en plataformas ARM. Uno de los cambios más importantes es el trabajo que realiza @ptitSeb en un recompilador dinámico x86 → ARM. Esto significa que algunas de las llamadas x86 se convierten directamente en ARMcode "sobre la marcha" sin emulación, en el momento en que se solicitan. Esta misma técnica se utiliza en muchos otros emuladores para acelerar el rendimiento general.

Por ejemplo, en el pasado, cuando ejecutabas el emulador PSX en una placa ARM64, que se compiló para ARM64, no podía utilizar el recompilador dinámico (por ejemplo, C2 o N2), e incluso si la placa era más potente que otras placas, los juegos eran muy lentos y estaban lejos de ejecutarse a maxima velocidad, mientras que los ARMboards mucho más lentos (por ejemplo, el ODROID C1 o U3) eran perfectamente capaces de ejecutar estos juegos a toda velocidad con un recompilador ARMdynamic. Algunas aplicaciones se ejecutan entre 2 y 10 veces más rápido con el recompilador dinámico que con la propia emulación. Lo mismo se aplica para box86, los juegos que antes eran demasiado lentos para jugar ahora se ejecutan mucho más rápido, abriendo muchos más juegos que pueden ejecutarse en ODROID. El recompilador dinámico acelera bastante los tiempos de carga, los juegos se cargan mucho más rápido que antes, en algunos juegos tenías que esperar 5-15 minutos para cargar los datos del juego, ahora se cargan en solo unos segundos.

Recapitulación de algunos juegos antiguos

Neverwinter Nights

Aunque el juego ya funcionaba bastante bien la última vez que lo revisé, la última mejora hace que el juego tenga una experiencia mucho más fluida. El rendimiento en general ha aumentado bastante, y ahora puedes ajustar la calidad de los gráficos al máximo. El juego se ejecuta casi como un juego nativo con solo un pequeño tartamudeo aquí y allá, básicamente lo que esperarías en un PC más antiguo. Recomiendo este juego en ODROID-XU4.

God Will Be Watching

No hay mucho que decir: el juego funcionaba bien, aunque no a toda velocidad, ¿qué ha cambiado? Ahora funciona a toda velocidad perfectamente

Freedom Planet

Cuando ejecuté por primera vez el juego, el juego tardó mucho en cargarse (entre 5 y 10 minutos). Ahora tarda unos 30 segundos en iniciarse, es decir, más de 10 veces más rápido que antes. Dentro del juego, aparecían algunos problemas menores de velocidad. No eran nada que hiciera que el juego no fuese jugable, pero no se ejecutaba a toda velocidad, lo cual era una pena ya que se trata de un juego similar a Sonic. Esto ahora se ha solucionado y el juego funciona perfectamente a máxima velocidad.

Faster Than Light

Este es otro juego comercial bien conocido que ya se estaba ejecutando en box86, pero con un tiempo de carga excesivo de más de 5 minutos. Esto también se reduce a menos de un minuto, y el problema de sonido también ha desaparecido. El juego ahora es lo suficientemente rápido como para reproducir música y efectos de sonido y no se aprecia ningún tipo de retardo.

World of Goo

El juego ahora se ejecuta a máxima velocidad, pero podría necesitar LIBGL_FB=3 como indicador para ejecutarse, debido a algunos problemas con la inicialización de GLX.

Qué hay de nuevo

Entonces, en general, los juegos de la última vez funcionan mejor, eso es bueno, pero ¿qué hay de nuevo a parte de esto? ¿Qué podemos ejecutar que no pudiéramos ejecutar antes?

Day of the Tentacle Remastered

Comencemos con algo grande y brillante.

Figura 1 - Day of the Tentacle Remastered en 1080p en el ODROID-XU4

Figura 2 - ¡El verde nunca fue tan acogedor!

Me encanta el juego original Day of the Tentacle, y todavía lo juego hoy en ScummVM en mis ODROID. Ahora, con box86, también puedo reproducir la versión remasterizada con gráficos e interfaz mejorados. Se ve increíble y se ejecuta muy bien. Existe un pequeño inconveniente por ahora. El tiempo de carga entre pantallas puede ser de 10 segundos, que es excesivos si tenemos en cuenta de que se trata de un juego donde tienes que caminar, explorar y probar cosas de un lugar a otro. Aun así, el juego funciona bien, y si no te importa demasiado esperar el tiempo de carga, no hay diferencia entre este y uno que se ejecute en un PC normal.

Jelly Killer

Este juego de aspecto bastante simple es, en realidad muy pesado para la GPU, probablemente debido a su uso de CRT y otros sombreadores.

Figura 3: una gelatina asesina anda suelta y salta por numerosos niveles

El juego es un juego de plataformas de rompecabezas donde necesitas cronometrar tus saltos correctamente, infestar a los humanos para alcanzar tu objetivo o matar a otros enemigos. Es un juego pequeño y divertido, que hace un tiempo se ejecutaba muy lento, pero ahora es bastante jugable si no a maxima velocidad.

Pier Solar and the Great Architects

Este juego de rol es un homenaje a los viejos juegos de rol estilo Mega Drive/Genesis y tiene un modo de 16 bits que parece ser de Mega Drive/Genesis. También tiene un modo HD y HD+ que se ve bastante mejor y pulido.

Figura 4 - Pantalla de título de Pier Solar

Figura 5: Saquear casas de extraños como en todo buen juego de rol

Figura 6: Puedes seleccionar auto para los combates basados en IA como una forma más rápida para terminar una pelea

Para mi gusto, el juego tarda demasiado en comenzar, pero está bien, y me pregunto cómo será en las secciones posteriores del juego. No obstante, es totalmente jugable en el ODROID-XU4, así que pruébalo si quieres.

Postal 2

Muchos probablemente habrán oído hablar de este juego, no tanto podrían haberlo jugado. En algunos países todavía está prohibido debido a su naturaleza controvertida. Este juego de acción te permite seguir un camino pacífico para lograr tus objetivos, pero también es el camino más aburrido. El juego funciona en general bien, pero no perfecto. Tuve problemas con el sonido y es difícil cambiar la resolución. El juego es un poco lento, no se puede jugar del todo bien, definitivamente no se ejecuta a toda velocidad.

Figura 7: Otro juego completo en 3D con un tema muy extraño

Figura 8 - Este tipo no sabe lo que le viene en cima

The Bard's Tales

The Bard's Tales es una impresionante serie de juegos de rol que se ven muy bien y funcionan perfectamente en el ODROID-XU4 y otros ODROID. Los gráficos son realmente fantasticos, y aunque el juego ya funcionaba cuando lo probé hace un par de meses, ahora tiene un rendimiento mucho mejor, lo cual hace que este juego se sienta como un juego nativo

Figura 9: incluso el menú está completamente animado en 3D

Figura 10: los gráficos son impresionantes en este juego, especialmente en ODROID

Simplemente recomiendo el juego, es muy divertido y tiene la famosa "Canción de la cerveza" (https://www.youtube.com/watch?v=eTUJNeuFIFA), y un juego que tiene cerveza no puede ser malo.

Toki Tori

Este juego extrañamente solo funciona con soporte OpenGL 1.x. El juego tiene buen aspecto y se ejecuta muy bien. Debes rescatar todos los huevos del juego recogiéndolos. Este juego de rompecabezas es bastante desafiante y te hace pensar antes de llevar a cabo tus movimientos. Tiene una buena pinta e inclye una pequeña melodía que se adapta perfectamente al juego. En general, este juego funciona muy bien, incluso si no tiene la misma velocidad que en un PC normal, el juego es bastante jugable y no se percibe lentitud. El soporte para mandos de juegos es excelente e incluso admite vibración si tienes un mando que lo admita. Este es un juego muy adictivo y lo recomiendo.

Figura 11 - Toki Tori, un juego de plataformas de rompecabezas altamente adictivo

Figura 12: Tienes que lidiar con cantidades limitadas de movimientos especiales como teletransportadores

Worms Reloaded

Jugué mi primer juego de "Worms" en el Amiga 500. En realidad, solo se llamaba "Worms", y fue un gran éxito y muy divertido de jugar. La serie continúa incluso hoy, y tiene un par de remakes de versiones anteriores del juego y versiones en 3D.

Figura 13 - Gusanos recargados en el ODROID-XU4

Figura 14: el juego tuene muy buen aspecto y funciona perfectamente

El juego se ejecuta sorprendentemente rápido. La única vez que se reletiza es cuando el PC calcula su movimiento, pero el movimiento y los ataques reales se ejecutan perfectamente.

UnEpic

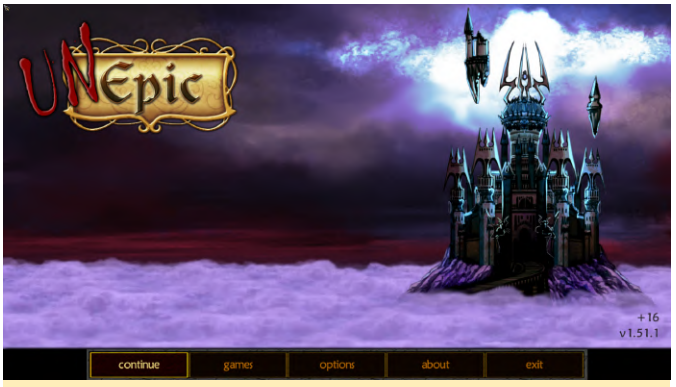


Figura 15: lo que te sucede cuando necesitas un baño

Figura 16: Este buscador de mazmorras tiene estupendos gráficos y efectos de iluminación

Este impresionante buscador de mazmorras también funciona a maxima velocidad en el ODROID-XU4. El único problema es que requiere una cantidad relativamente grande de RAM (para placas ARM, es decir). Necesitas al menos 1500 MB de memoria libre para ejecutarlo, por lo que, dependiendo del sistema operativo, es posible que no puedas ejecutarlo en un XU4, y el juego es más adecuado para un N1 con controladores ARMHF y 4 GB de memoria. En general, este juego es impresionante y lo recomiendo encarecidamente.

Honorable mentions

Hay toneladas de juegos en los que no puedo entrar, puesto que hay muchos que funcionan ahora, quisiera compartir algunos con vosotros para daros una idea de hasta donde llega el sistema.

Figura 17 - Broken Sword Director's Cut - Adventure

Figura 18 - Demon Hunter – Chronicles from Beyond - Hidden Object Game

Figura 19 - Don't Starve + Together - Survival / Crafting

Figura 20 - eets Munchies - Puzzle

Figura 21 - Enigmatis Series - Hidden Object Game

Figura 22 - Human Resource Machine - Programming “as a game”

Figura 23 - Hyper Light Drifter - Action Platformer

Figura 24 - Icewind Dale Enhanced Edition - RPG

Figura 25 - Little Inferno - Casual / Puzzle

Figura 26 - Memoranda - Adventure

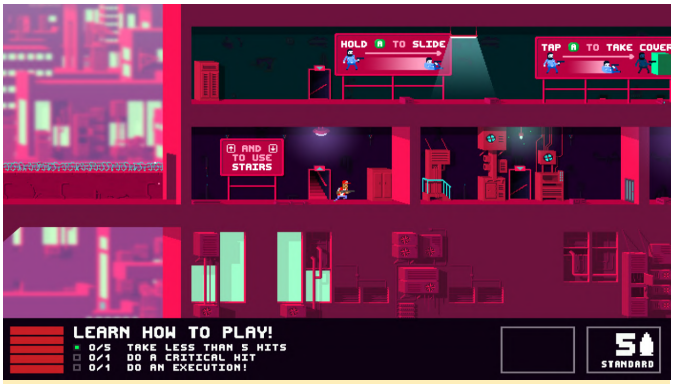


Figura 27 - Not a Hero - Action

Figura 28 - Papers, Please - Puzzle

Figura 29 - Space Pirates and Zombies - Action / Simulation

Figura 30 - Super Meatboy - Platform / Puzzle

Figura 31 - VA-11 Hall-A - Simulation / Visual Novel

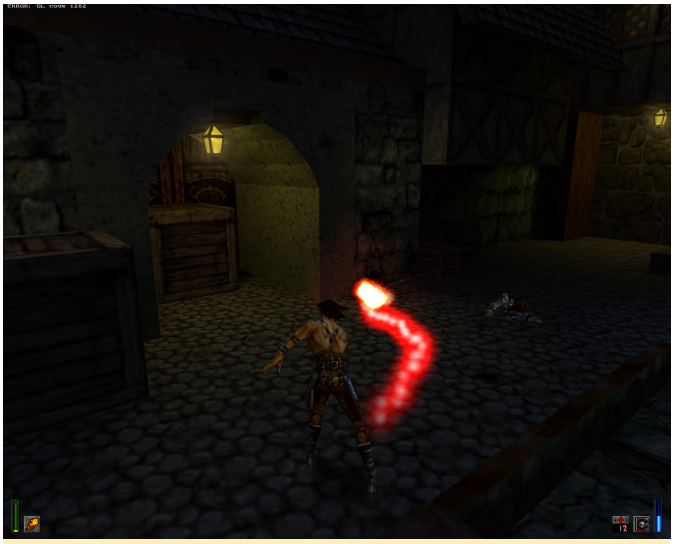


Figura 32 - Heretic 2 - 3rd Person "Shooter" / Action

Hay muchos y diferentes de juegos que funcionan actualmente. Hace un par de meses, algunos eran tan lentos que más bien parecían una presentación de diapositivas, pero ahora estos juegos funcionan a máxima velocidad (por ejemplo, Hyper Light Drifter). Es realmente asombroso como ha evolucionado el emulador en tan solo un año.

Juegos especiales

Sin embargo, hay algunos juegos que quisiera destacar, ya que son mis favoritos, y algunos de los cuales quizás también conozcas. Dos son muy cercanos e incluso comparten un nombre similar. Ya los jugué hace un tiempo en ExaGear, donde se ejecutan con software de renderizado 3D. Era impresionante que este juego realmente funcionara, pero ahora puedo ejecutar estos juegos con box86 y aceleración de hardware 3D. Estos juegos me traen tantos recuerdos, ya que los jugué de niño y ahora están en la colección de juegos de todo el mundo, especialmente en PC. Estoy hablando, por supuesto, sobre Unreal y Unreal Tournament (la edición original de 1999).

Tanto Unreal como Unreal Tournament fueron portados a Linux hace muchos años. Hoy en día es un tanto difícil hacer que funcionen, ya que usan controladores muy antiguos, especialmente en lo que respecta al sonido. Pero por lo demás, estos juegos siguen siendo increíbles, y funcionan tan bien como recuerdo haberlos jugado en 1999 en mi procesador AMD K6-2 450 MHz con la tarjeta gráfica Riva TNT. El juego funcionó bien en aquel entonces, pero ahora en ODROID, incluso puedo jugarlos en 720p o 1080p, que no era posible en aquellos tiempos. Es realmente sorprendente, y para ser sincero, estoy deseando tomarme un tiempo libre y volver a reproducir Unreal para ver si es así como lo recuerdo.

Figura 33: Volver a ver la exhibición de vuelo de 1998 fue tan icónico

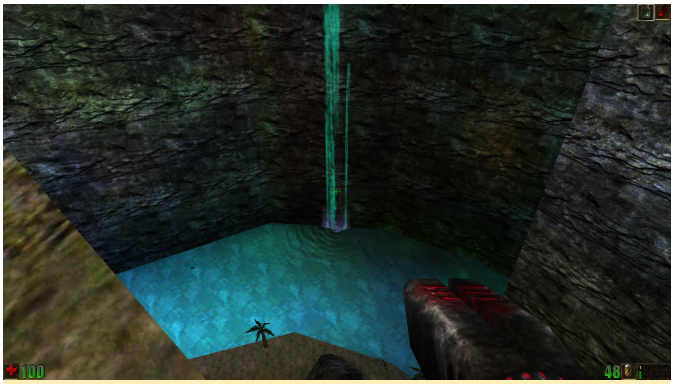


Figura 34 - Los gráficos eran excepcionales para su época, y todavía se ven bien hoy en día

Unreal funciona mejor actualmente. Enicia el juego con las siguientes opciones:

```
$ LIBGL_SRGB=1 LD_PRELOAD=/usr/lib/arm-linux-gnueabihf/libaoss.so.0 box86 UnrealLinux.bin
```

LIBGL_SRGB=1 usa un modo especial en gl4es que aumenta el brillo general, ya que de lo contrario el juego puede ser extremadamente oscuro. LD_PRELOAD=/usr/lib/arm-linux-gnueabihf/libaoss.so.0 es necesario ya que el juego usa /dev/dsp para la salida de sonido que ya no se usa, con esto podemos redirigir el sonido a una emulación alsa.

El juego tarda un poco en iniciarse, ya que carga una gran cantidad de datos desde el principio y luego pasa directamente a la exsibición de vuelo. En cuanto al rendimiento, es increíble, quiero decir que realmente parece que este juego se ejecuta de forma nativa en el ODROID. No he experimentado ningún retardo mientras jugaba. De hecho, el menú parece ser más lento que el juego en sí, y lo ejecuto con detalles altos en una profundidad de color de 32 bits a 720p. Simplemente es increible jugar a este juego.

Figura 35 - Lo que Unreal empezó, Unreal Tournament lo continuo

Figura 36 - Todavía lo tengo ... incluso si solo era el primer nivel

Unreal Tournament fue la perfección del combate multijugador en primera persona. Jugué esto innumerables veces con amigos en fiestas LAN mientras estaba de vacaciones. El modo de juego Asalto siempre fue mi favorito, y hasta hoy, hay muy pocos juegos que incluso hayan estado cerca de implementar un modo de juego tan emocionante. Extraño mucho jugar esto con mis amigos. Siempre odié "capturar la bandera", en la que se basaba cualquier otro juego. Unreal Tournament hizo todo bien y me atrevo a decir que me gustó más que Quake 3. Sin embargo, no pude hacer que el sonido funcionara en esta versión. Se basa en un controlador muy antiguo llamado OSS, y aunque puedo cargar módulos de kernel para esto, no puedo usarlo para el sonido. Otros métodos como aoss y padsp tampoco funcionan, así que estoy atascado sin sonido.

Conclusion

En solo un año, Box86 realmente despegó en términos de compatibilidad. La última vez que escribí al respecto, fue un proyecto con muchas promesas, pero con poco soporte. Ahora está por las nubes, ¡y el soporte para los juegos es increíble!

Espero con ansias lo que podremos hacer en los próximos meses. Espero soporte para Unity y WINE, ya que esto abre las puertas a más juegos. Espero especialmente WINE, que sería realmente impresionante porque WINE, en combinación con OpenGL, podría significar que también podremos jugar juegos de Windows con aceleración de hardware en ODROID. El proyecto promete ser muy emocionante.

Hay otros que también siguen los juegos de desarrollo y prueba en Raspberry Pi y otras plataformas. Puedes consultar el canal de Youtube "Pi Lab" (**https://www.youtube.com/channel/UCgfQjdc5RceRlTGfuthBs7g/videos**), donde puede encontrar aún más juegos en box86. ¡Gracias @ptitSeb por todo su gran trabajo en este proyecto!

Procesamiento de Huellas Digitales: Ejecutar el Conjunto de Herramientas de Huellas Digitales NIST NBIS en un ODROID-XU4

May 1, 2020 By Andrew Ruggeri ODROID-XU4, Tutoriales

El Instituto Nacional de Ciencia y Tecnología, o NIST, mantiene un conjunto muy amplio de herramientas de código abierto conocido como NIST Biometric Image Software, o NBIS para abreviar. Su funcionalidad está centrada en las huellas digitales [1], [2]. Este artículo cubrirá todo lo necesario para empezar a trabajar y comparar huellas en un ODROID-XU4.

Partes de este artículo han sido tomadas del proyecto maestro relacionado con el preprocesamiento de imágenes en huellas digitales para aumentar el rendimiento de coincidencia del software proporcionado por NBIS. El proyecto hace uso de un ODROID-XU4 ya que el procesador utilizado en el ODROID-XU4, un Samsung Exynos 5422 [3], es el mismo procesador que el Samsung Galaxy S5, un dispositivo que tiene un lector de huellas digitales. Este proporciona una representación casi exacta de un objetivo potencial donde se utilizarían estos algoritmos de preprocesamiento. Aunque este es importante para mi proyecto original, también es importante para muchos otros posibles usos. El tema del bajo consumo y el alto rendimiento del ODROID-XU4 lo hace ideal para muchos proyectos que uno podría tener en mente. Por ejemplo, un ODROID-XU4 sería un controlador ideal en un sistema IoT que también podría implementar autentificación biométrica para el acceso de personas.

Configuración del software

El ODROID-XU4 debe ejecutar Ubuntu 18.04LTS, preferiblemente la versión mínima. NBIS se compila utilizando el conjunto de herramientas GNU C/C++. Después de configurar ODROID-XU4, las instrucciones de la página wiki de hardkernel son una excelente guía a seguir, el NBIS debe descargarse desde **https://www.nist.gov/itl/iad/image-group/products-and-services/image-group-open-source-server-nigos**.

Abre una terminal en el directorio donde se descargó nbis_v5.0.0.zip. Los siguientes pasos tendrán como resultado la compilación de todos los binarios necesarios. Primero, es necesario instalar cmake ya que será usado en pasos posteriores, luego el archivo debe extraerse y se moverse a una carpeta llamada 'nbis'. Dentro de ese directorio 'nbis', se crea un directorio 'build' donde el script de configuración (setup.sh) recibe las instrucciones para colocar todos los archivos binarios en el paso final de la instalación. La ruta al script de configuración debe ser una ruta completa, no una ruta relativa o una que use atajos, como ~. Además, los argumentos se pasan sin X11 y stdlibs, estos argumentos ayudan a reducir las dependencias externas que son necesarias. Por último, la secuencia de pasos de make, compila y mueve los archivos binarios a su ubicación final, respectivamente.

```
$ sudo apt-get install cmake
$ unzip nbis_v5_0_0.zip
$ mv Rel_5.0.0 nbis
$ cd nbis
$ mkdir build
$ ./setup.sh /home/odroid/Downloads/nbis/build --
without-X11 –STDLIBS
$ make config
$ make it
$ make install
```

Los pasos anteriores tienen como resultado un conjunto de binarios que funcionan y crean todo lo necesario para avanzar. Sin embargo, a raíz de algunas pruebas, se ha demostrado que añadir un par de marcas de compilación adicionales puede crear archivos binarios que están extremadamente bien ajustados para el ODROID-XU4. En el directorio "nbis" hay dos archivos "rules.mak" y "arch.mak", abre ambos archivos. Abre una línea que declara la variable "ARCH_FLAG", agrega los siguientes elementos que se le asignarán. Todos los elementos existentes que se asignaron a "ARCH_FLAG", como "-fPIC", deberían mantenerse.

```
ARCH_FLAG := -fPIC -mfloat-abi=hard -mcpu=cortex-
a15 -fipa-pta
```

Después de editar y guardar los archivos con los cambios mencionados anteriormente, se deben ejecutar los siguientes comandos.

```
$ make clean
$ make config
$ make it
$ make install
```

Software

De todos los binarios que se crean, los siguientes son los únicos que se necesitan en este ejemplo.

CWSQ : Conversor de archivos de imagen a Wavelet CWSQ creará un archivo wavelet comprimido a partir de una entrada de escala de grises. Existen dos franjas de compresión para nuestra prueba, 5: 1 es la que se utilizó, la otra opción es 15: 1. La mayoría de las herramientas NBIS funcionan con este formato de archivo wavelet, de modo que todas las huellas digitales compatibles deben estar en este formato.

MINDTCT: detección de minucias

MINDTCT es un programa de detección de minucias, un punto de minucias puede considerarse como una característica interesante en una huella digital. El tipo, la ubicación y el ángulo de los puntos encontrados en este programa se utilizan para comparar y contrastar puntos [4]. Hay muchos tipos diferentes de puntos de minucias, pero MINDTCT solo detecta puntos finales (primera imagen) y bifurcación (segunda imagen) en una huella digital. El tipo, la ubicación y la orientación del punto se guardan en un archivo. Cada punto de minucias se guarda como coordenadas en función de su distancia en milímetros, con incrementos de 0,01 mm, hasta la esquina inferior izquierda de la imagen [1]. Además, los puntos contienen información del ángulo relacionada con la dirección local de los contornos de las plieges y los valores (las líneas blancas y negras en una imagen de huella digital) que forman un punto minucioso. La siguiente imagen muestra dos puntos de minucias diferentes. Los ángulos, "A" y "B" representados en las imágenes representan dos métodos diferentes para medir ángulos. NBIST utiliza ángulos medidos por "A".

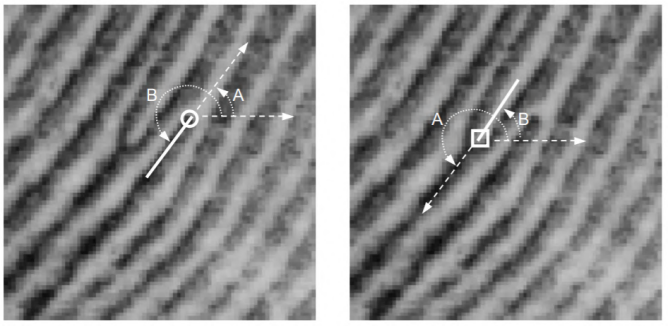


Figura 1: el ángulo A muestra la medición ANSI/NIST, B muestra el ángulo de medición del FBI. izquierda es un final de pliegue, derecha es una bifurcación

BOZORTH3: Coincidencia de huellas digitales

El algoritmo y el software Bozorth3 es un código de exportación controlado. En un nivel alto, el software toma una huella digital y se compara con el número 'n' de huellas digitales ingresadas objetivo y devuelve una tasa de coincidencia para cada una de las huellas digitales objetivo. Los archivos de huellas digitales que usa están en el formato de archivo creado a partir de MINDTCT.

Ejecutando

Si no hay un dispositivo de captura de huellas digitales conectado o disponible, hay varias bases de datos que contienen conjuntos de imágenes de huellas dactilares que se pueden usar para realizar pruebas. Una de estas bases de datos es la base de datos FVC 2002 disponible en http://bias.csr.unibo.it/fvc2002/. Los comandos se pueden ejecutar desde un terminal que ejecuta el directorio "build/bin" o esta ruta se puede agregar a la variable $ PATH de los terminales, de modo que los comandos pueden ejecutar desde cualquier ubicación.

```
$ export
PATH=$PATH:/home/odroid/Downloads/nbis/build/bin
```

Para simplificar las cosas, supondré que hay una huella digital conocida llamada "myprint.tiff", y la intención es ver si coincide con otra huella conocida como "mysteryprint.tiff". El primer paso es convertir ambas imágenes de huellas digitales en un wavelet.

```
$ cwsq .75 myprint.tiff
$ cwsq .75 mysterprint.tiff
```

Después de cada comando "cwsq", se mostrará un poco de información sobre la impresión y creará un * .wsq correspondiente con el mismo nombre y el archivo de entrada * .tiff. A continuación, estos archivos wavelet se envían a MINDTCT, esto creará varios tipos de archivos, el único de interés es el archivo * .xyt.

```
$ mindtct -b -m1 myprint.tiff myprint/
```

El último paso es comparar o "hacer coincidir" las huellas entre sí. Esto se puede hacer con el siguiente comando:

```
$ bozorth3 -m1 A outfmt=spg -T 30 -p myprint.xyt
mysteryprint.xyt
```

Sin embargo, se pueden comparar varios archivos con nuestra huella (myprint). Esto se puede hacer fácilmente con un operador comodín, * .xyt como último argumento en lugar de "mysteryprint.xyt" compararía todos los archivos xyt con myprint.

```
$ bozorth3 -m1 A outfmt=spg -T 30 -p myprint.xyt
*.xyt
```

El resultado será una lista con una o más comparaciones. Cada línea es una nueva comparación, donde el primer campo es una "puntuación" que determina como de similares son las coincidencias. El argumento '-T 30' en el comando es un umbral que significa no tener en cuenta nada menos que 30. El segundo valor es la impresión 'objetivo', por lo que siempre será 'myprint' en cada fila, el siguiente elemento es el archivo que fue comparado. A continuación, se muestra un ejemplo del resultado cuando se usa un comodín, por lo tanto, myprint se compara con sí mismo.

```
250 myprint.xyt myprint.xyt
41 myprint.xyt mysteryprint.xyt
```

Dado que mysteryprint obtuvo un 41, se considera que coincide con myprint. No existe un valor umbral real perfecto, ya que todo se basa en lo que es una tasa aceptable de falsos positivos y falsos negativos. Un umbral para desbloquear una puerta de una casa sin duda tendrá que ser más alto que acceder a un panel IoT para ajustar la temperatura de contacto

Si te ha sido interesante, tienes más información en los siguientes enlaces de referencia. El documento y el código del proyecto original se pueden encontrar en GitHub en https://github.com/AndrewRuggeri/FP.

Referencias

- [1] C. Watson et al., “User’s Guide to NIST Biometric Image Software (NBIS).” Gaithersburg, p. 207, 2007.
- [2] E. Tabassi, C. Wilson, and C. Watson, “Fingerprint Image Quality,” NIST, vol. NISTIR 7151, 2004.
- [3] R. Roy, Hardkernel ODROID-XU4 Manual, 20170310th ed. Hardkernel, 2017.
- [4] D. Maltoni, “A Tutorial on Fingerprint Recognition,” Adv. Stud. Biometrics, vol. 3161, pp. 43–68, 2005.

Consejos y Trucos Avanzados de ODROID-GO: Descomprime ROM Mientras Mantienes las Caratulas y la Lista de Juegos

May 1, 2020 By Brian Ree Juegos, ODROID-GO Advance, ODROID-GO

Este es un breve tutorial para ayudarte con tus ROMs y el ODROID-GO Advance. Muchos de nosotros tenemos ROMs con archivos multimedia que los acompañan, como carátulas, capturas de pantalla, logotipos, etc. Algunas veces estas ROMs tienen archivos comprimidos. Ahora, ¿Realmente queremos hacer uso de la batería para descomprimir los juegos antes de cargarlos? No creo que sea una buena idea. El ahorro en espacio es mínimo en los juegos de las generaciones de 8 y 16 bits y no queremos descomprimir ISOs e IMGs en nuestro ODROID-GO Advance. Entonces, ¿cómo podemos reprocesar todos estos archivos y mantener la conexión a los archivos multimedia a través del archivo XML de la lista de juegos? Como sabes, Batocera (**https://batocera.org/download**) es un excelente sistema operativo basado en RecallBox, para ODROID-GO Advance. La forma en que el sistema lee las ROMs con los archivos multimedia asociados es a través del archivo gamelist.xml. El problema es que si descomprimimos el archivo ZIP, a menudo terminaremos con un nombre y extensión de archivo diferentes. Ahora tenemos que obtener un nuevo conjunto de caratulas para nuestras ROM que puede llevar un tiempo dependiendo de cuántos sistemas esté procesando, o tenemos que editar el gamelist.xml a mano, lo que requeriría un esfuerzo considerable. Te mostraré cómo procesar un conjunto de ROMs comprimido en unos pocos pasos, todo mientras mantengo las entradas correctas en el archivo gamelist.xml para que su diseño de la caratula siga asignándose correctamente. Lo haremos todo con un script bash desde la línea de comandos, para que puedas ejecutarlo a través de SSH si fuera necesario.

Preparación

Para que podamos hacer los cambios necesarios en un archivo XML desde un script bash, necesitamos un poco de ayuda. La herramienta que usaremos se llama XmlStarlet. Instálala en el sistema, la cual utilizaremos para procesar los conjuntos de ROM. Necesitarás un sistema Linux para hacer esta parte, aunque OSX y Ubuntu en Windows probablemente también puedan manejar este script. En realidad es muy simple: XmlStarlet se ocupa de la única parte compleja. Para instalar el paquete, ejecuta lo siguiente desde la línea de comandos.

```
$ sudo apt-get install xmlstarlet
```

A continuación, configuraremos nuestro script. Puedes descargar una copia del script desde https://bit.ly/34Y7LNX o puede copiar y pegar el siguiente texto en un archivo local, clean_unzip.

```
#!/bin/bash

if [ ! -d ./done ]; then
  mkdir ./done
fi
for z in *.zip
do
  mkdir tmp
  cp "$z" tmp
  cd tmp
  unzip "$z"
echo "Looking for *.$1"
  echo $(ls ./*.$1 2>/dev/null | wc -w)
  files=( ./*.$1 )
  if (( $(ls ./*.$1 2>/dev/null | wc -w) )); then
    echo "files do exist $y"
if [ ! -f "${z%.zip}.$1" ]; then
    mv *.$1 "${z%.zip}.$1"
  fi
mv *.$1 ../
      TF="${z%.zip}.$1"
    OF="${z}"
    NF="${TF}"
if [ -f ../${2} ]; then
    echo "replace $OF with $NF"
    xmlstarlet ed --inplace -u
      "//gameList/game[path="./${OF}"]/path" -v
      "./$NF" ../${2}
    #../gamelist.xml
  fi
        mv ../"$z" ../done/
  fi
cd ..
  rm -r tmp
#break
done
```

¿Qué hace el script? Bueno, realiza los siguientes pasos.

- Crea una carpeta tmp en el directorio local.
- Copia el siguiente archivo ZIP en la carpeta tmp.
- Expande el archivo ZIP.
- Si se encuentra un archivo coincidente con extensión, cambia el nombre del archivo resultante con el mismo nombre que el archivo ZIP pero con la extensión actual.
- Mueve el archivo resultante al directorio ROM principal.
- Reemplaza la entrada en gamelist.xml por la nueva extensión, es decir, no zip.
- Si se procesa, mueve el archivo ZIP a la carpeta completa.
- Elimina el directorio tmp.
- Repite los procesos anteriores, hasta que se getionen todos los archivos.

Uso

Puedes ejecutar el script en una tarjeta SD, a través de SSH, o en una tarjeta SD montada en tu dispositivo ODROID-GO. Utiliza una buena cantidad de operación del sistema de archivos para completar la limpieza de un conjunto de ROM. En general, recomendaría no hacerlo directamente en una tarjeta SD debido a la cantidad de operaciones, aunque en realidad he limpiado 30 ROM con él, directamente en una tarjeta montada, sin problemas. Cuando lo estés ejecutando en grandes conjuntos de ROMs, habrás dejado archivos ZIP en el directorio que estás procesando. Algunos archivos ZIP se procesarán y se moverán a la carpeta completa. Esto significa que los archivos comprimidos que sobran tienen contenido con una extensión de archivo diferente. Veamos los comandos utilizados en un ejemplo. AVISO: El script está diseñado para ejecutarse con archivos XML basados en RecallBox. Si está utilizando un formato XML diferente, tendrá que editar la línea donde se llama xmlstarlet y configurar una estructura diferente para

que coincida con tu archivo XML. Primero, nos aseguraremos de que podamos ejecutar el script.

```
$ sudo chmod +x ./clean_unzip
```

A continuación, ejecutaremos el script y apuntaremos a la extensión del archivo. Usamos Sega Mega Drive como ejemplo.

```
$ sudo ./clean_unzip bin gamelist.xml
```

Tendremos algunos archivos ZIP que nos sobran. Abramos uno. Resulta que el contenido tiene una extensión smd. Ejecutemos el script en los archivos ZIP restantes.

```
$ sudo ./clean_unzip smd gamelist.xml
```

Es posible que debas ejecutar el comando varias veces para manejar las extensiones de archivo restantes en un conjunto de ROM especifico. El script funcionará con los archivos ZIP restantes, por lo que realmente no es tan malo. Con unas pocas llamadas podemos limpiar cualquier conjunto de ROM. A continuación, se muestran algunas fotos del antes y después para que te hagas una idea de lo que hace el script.

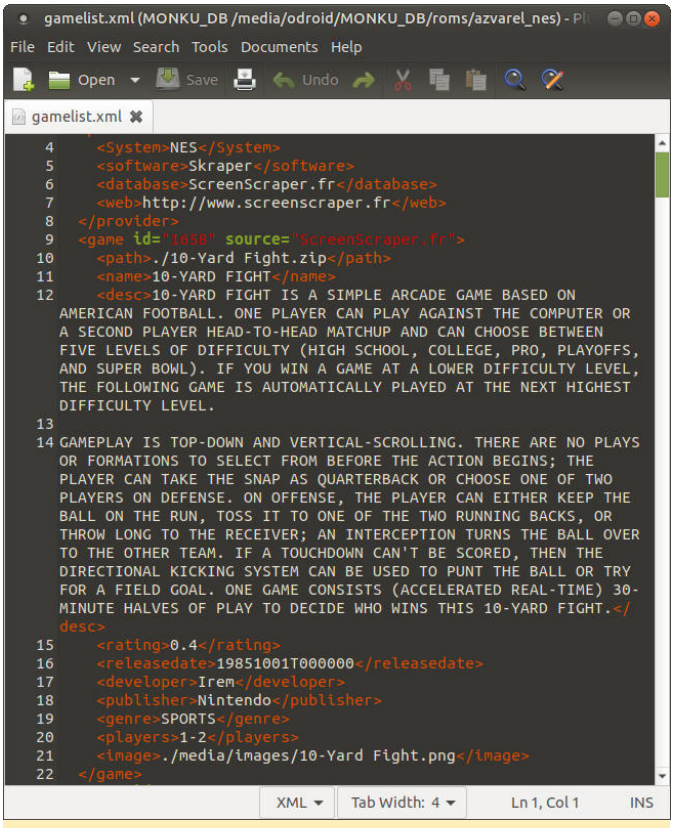


Figura 01 - Antes: observa el "zip" en la entrada del nombre del archivo

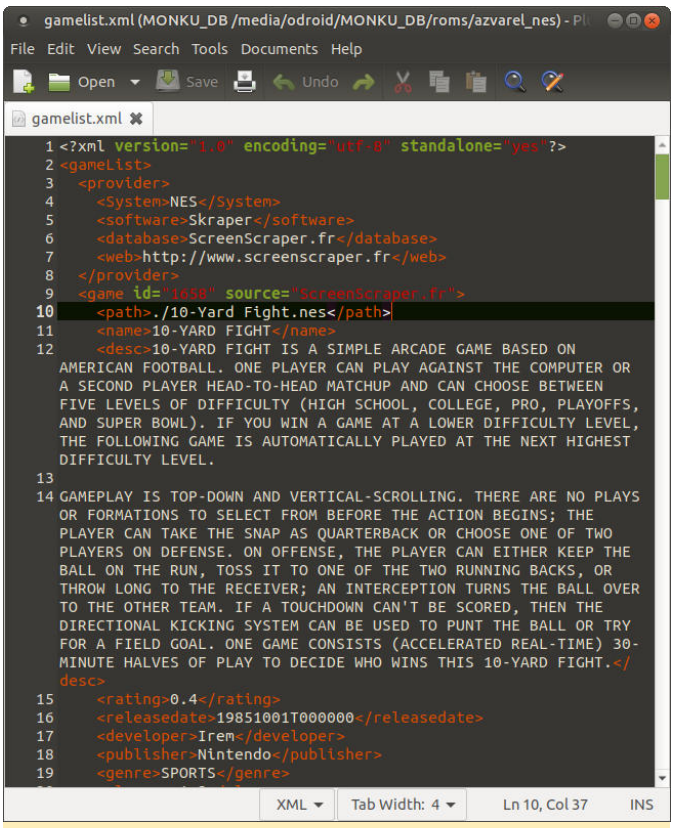


Figura 02 - Después: observa que el nombre del archivo ha cambiado

Aquí tienes la vista del directorio en sí.

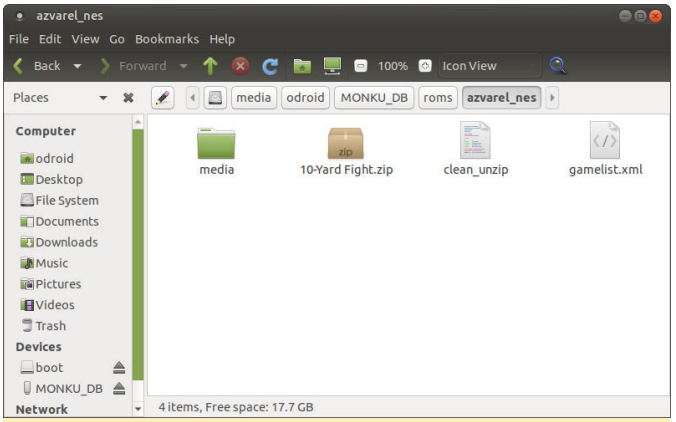


Figura 03 - Antes: archivos ZIP y ningún directorio "done"

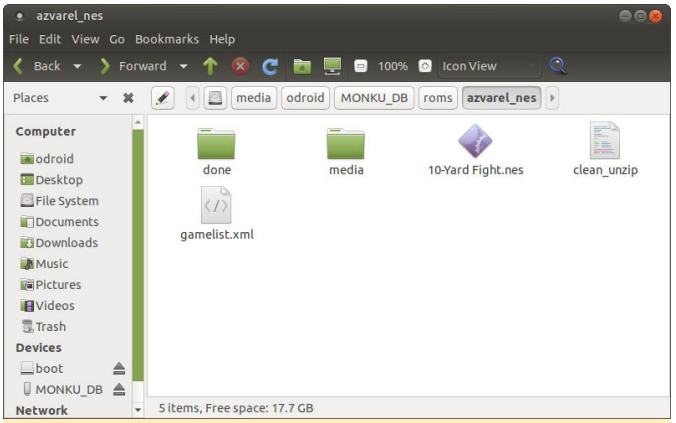


Figura 04 - Después: ROM descomprimidas y una carpeta "done" que contiene los archivos zip completados

Terminando

Esto nos lleva al final este tutorial. Una forma rápida que debería serte de gran utilidad si necesita descomprimir un conjunto de ROM que tenga archivos multimedia asignados a la copia comprimida de tu ROM. Este script es ideal para ajustar las ROMs comprimidas y poder usarlas con dispositivos portátiles donde tal vez no desee utilizar los ciclos de CPU adicionales para expandir el archivo del juego. Para comentarios, preguntas y sugerencias, visita el artículo original en **http://middlemind.net/tutorials/odroid_go/oga_rl_dc_build.html**.

¿Vamos a jugar a un juego? - Jugar a la Promesa de Google Stadia, con un Ancho de Banda más Práctico

May 1, 2020 By Dave Prochnow Juegos

El lanzamiento mediocre de Google Stadia dejó a muchos jugadores en la estacada. Claro que, el atractivo de ejecutar juegos AAA dentro de su navegador sonaba a algo demasiado tentador, pero el ancho de banda se convirtió en un problema que aún no se ha solucionado.

Mientras esperas a que la tecnología se ponga al día con sus exageraciones del marketing, dirige tu navegador a otro paraíso de juegos on-line, uno en el que una conexión lenta a Internet no supone una desventaja, sino un regalo del cielo.

Simplemente dirígete a GameSnacks y toma un "byte" de juego on line que funciona en "cualquier dispositivo y en cualquier red". Suena muy parecido a la misma promesa de Google para su desafortunado lanzamiento de Stadia. Sin embargo, en este caso el aviso se refiere a ese punto destacado sobre trabajar en "cualquier red". Coge tus conexiones lentas, atoradas y dirígete a GameSnacks para jugar on line, que realmente funciona para el resto de nosotros.

Podemos charlar con TODOS los jugadores del mundo que no podían jugar a Google Stadia. Irónicamente, GameSnacks es un producto de un grupo de Google conocido como Área 120. Este "taller para productos experimentales" de Google es al mismo tiempo un programa y un producto. Es un programa de acceso temprano que permite a pequeños grupos de desarrolladores organizar sus ideas más alocadas de productos en un entorno empresarial, así como sentarse descalzo sobre almohadas de futón. El Área 120 también puede ser un producto cuando ayuda a crear un concepto con éxito como GameSnacks.

Figura 1. Un corralito para desarrolladores de Google Devs

Por extraño que parezca, Google afirma que la mayor parte de las ideas de productos lanzadas por el Área 120 serán fracasos. ¡Caramba! gracias por el apoyo y el aliento. Hmm, ¿se fomentó Google Stadia en el Área 120? Así que era para cerrar el trato y vender desarrolladores por los méritos de unirse al Área 120, Google afirma que ahora están muy trillados, "nuestros equipos aprenden" de sus fracasos.

Figura 2. El desarrollo de juegos de ordenador siempre es mejor cuando no usa ordenadores y puedes sentarte en el suelo.

Puede obtener más información sobre el programa del Área 120 en: https://area120.google.com.

Con respecto a GameSnacks, el Área 120 ha ayudado a organizar una colección de juegos basados en HTML5 que puedes jugar dentro de tu navegador Android usando casi cualquier tipo de conexión a Internet; el ancho de banda NO es un problema con este servicio de juegos. Actualmente, el catálogo de GameSnacks contiene seis estupendos juegos de 5 desarrolladores con bastante talento:

1. Bridge of Doom
2. Bubble Woods
3. Road Fury
4. Groovy Ski
5. Jump with Justin
6. Jewelish Blitz

Figura 3. GameSnacks es una mezcla muy heterogénea de juegos on line, si están ansioso de juegos HTML5 dentro de tu navegador Android; ya sabes, como te prometieron con Google Stadia.

Y los desarrolladores de estos juegos son:

1. Famobi
2. Inlogic Games
3. Black Moon Design
4. Geek Games
5. Enclave Games

Jugar a estos juegos es muy simple: solo tienes que hacer clic en el botón "Play" y tu navegador se abrirá en una nueva ventana (o Tab; dependiendo de si tu dispositivo Android es un dispositivo móvil o un SBC de escritorio) donde cada título se carga y se puede reproducir en menos de un minuto.

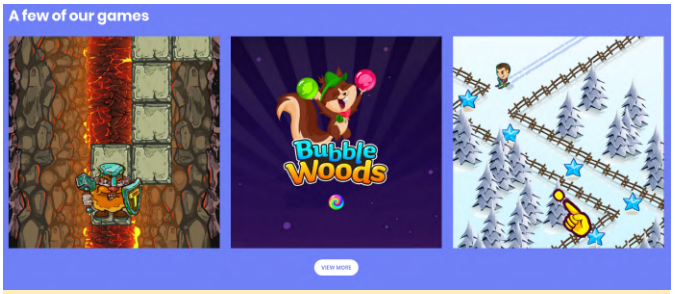


Figura 4. Actualmente, la lista de títulos de juegos disponibles es un poco escasa, pero está creciendo.

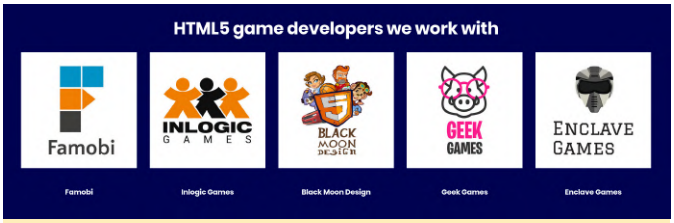


Figura 5. Todos estos desarrolladores son veteranos en los juegos HTML5, pero todavía están buscando más programadores que estén dispuestos a contribuir a GameSnacks.

OK, soy un mamón; Tras conseguir una puntuación de 668,544 en Bubble Woods usando mi navegador de escritorio Android ODROID-XU4, me enganché a GameSnacks. La reproducción es rápida, los gráficos y el sonido son de primera categoría y, lo mejor de todo, mi conexión de red con WiFi era más que suficiente. Además, estos no son solo juegos HTML5 "de ideas tardías" de desarrolladores olvidados. El desarrollador de Bubble Woods, Famobi, por ejemplo, tiene un catálogo completo de juegos HTML5 que están disponibles como títulos "freemium" y tienen una gran selección de juegos listos para comprar. Por lo tanto, hay una gran cantidad de títulos que hace que GameSnacks te proporcione una dieta constante de juego on line

Figura 6. El juego rápido y adictivo es el sello distintivo de GameSnacks.

Puedes ingresar al mundo de los juegos en línea con GameSnacks en https://gamesnacks.com.

Escritorio de Pantalla Múltiple Usando VNC - Parte 2 Una Versión Mejorada y Simplificada

May 1, 2020 By Adrian Popa Linux, Mecaniqueo, Tutoriales

Me parece que todos vamos a estar atrapados en casa más tiempo de lo que pensábamos. Algunos de nosotros también hemos tenido que trabajar durante este tiempo. Trabajar en una pantalla de ordenador portátil pequeña no es una tarea divertida, y usar cables HDMI mientras los niños corren tampoco es divertido. Entonces, ¿qué tal si usamos un ODROID como pantalla secundaria? Esto es algo así como una continuación de mi anterior artículo "Escritorio multipantalla con VNC" presentado en un número anterior de ODROID Magazine: **https://bit.ly/3bw1oEb**.

Entonces, la buena noticia es que no necesitas nada de lo que se describe en ese artículo. He echado un vistazo a la página del manual de x11vnc y encontré algunas opciones que simplifican enormemente las cosas y reducen la cantidad de hacks necesarios.

Crear un escritorio extendido

El objetivo es tener una configuración de doble pantalla: una pantalla sería la pantalla de tu ordenador portátil, la segunda pantalla sería un ODROID en red. El ordenador portátil (en mi caso) ejecuta Linux (obviamente), por lo que estamos buscando una solución de Linux. Lo ideal una que funcione a través de wifi.

Lo primero que debemos hacer es extender el escritorio físico. En el artículo anterior usé xrandr para extender el tamaño físico del escritorio. Sin embargo, tiene algunos problemas, especialmente con aplicaciones que no saben dónde termina la pantalla física, lo que hace que maximizar las ventanas se convierta en un auténtico quebradero de cabeza. Esta vez ampliaremos el escritorio agregando una nueva pantalla virtual.

Para un ordenador portátil con una GPU Intel, podemos hacer esto agregando /usr/share/X11/xorg.conf.d/20-intel.conf con el

contenido que se describe en https://bit.ly/2xQSQZW:

```
Section "Device"
  Identifier "intelgpu0"
  Driver "intel"
  Option "VirtualHeads" "2"
EndSection
```

Si tienes una GPU NVidia, puede probar esto en su lugar: https://bit.ly/3awV5yW.

Si reinicias tu servidor Xorg, verá dos nuevas pantallas virtuales en tu salida:

```
$ xrandr | grep VIRTUAL
VIRTUAL1 disconnected (normal left inverted right
x axis y axis)
VIRTUAL2 disconnected (normal left inverted right
x axis y axis)
```

Ahora que tenemos una nueva pantalla disponible, necesitaremos configurar una resolución específica y activarla. En mis pruebas, utilicé una resolución de 720p porque es lo suficientemente pequeña como para transmitirla sin problemas y lo suficientemente grande como para ser legible desde la distancia en un televisor de pantalla grande.

Deberás calcular los tiempos correctos para la resolución deseada y agregar un nuevo modo a la pantalla virtual. Afortunadamente, hay una herramienta que lo hace en función de una resolución de entrada y una frecuencia de actualización y forma parte del paquete

```
$ gtf 1280 720 60
```

Puedes usar la salida del comando para obtener la información necesaria y habilitar la pantalla:

```
$ xrandr --newmode "1280x720_60.00" 74.48  1280
1336 1472 1664  720 721 724 746  -HSync +Vsync
$ xrandr --addmode VIRTUAL1 "1280x720_60.00"
$ xrandr --output VIRTUAL1 --right-of LVDS1
```

Ahora debería aparecer una ventana emergente, como en la Figura 1, que muestra la nueva pantalla y te pregunta qué quiere hacer con ella.

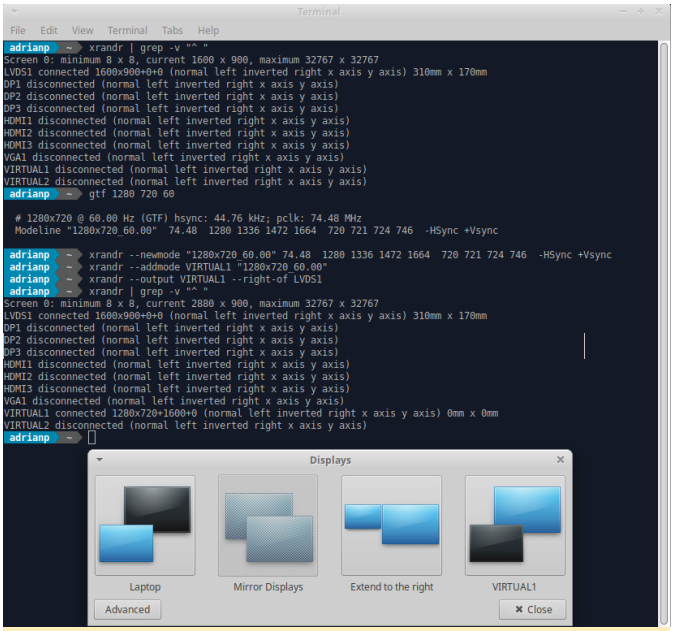


Figura 1. Crear una pantalla virtual

Lamentablemente, no puedes habilitar las pantallas virtuales de la misma manera en ODROID-XU4, por lo que esta técnica requiere que tu PC maestro esté basado en Intel. Pero espera, si solo tiene un ODROID (con suerte un ODROID-XU4, donde xrandr se ejecuta muy bien) como ordenador maestro, no todo está perdido. Todavía puede expandir el escritorio, tal y como se describe en el anterior artículo utilizando el script: https://bit.ly/34VZiuV:

```
$ DISPLAY=:0 xrandr --output HDMI-1 --fb 2560x720
--panning 1280x720
```

El parámetro fb especifica la resolución total, mientras que el parámetro de panorámica especifica una resolución de pantalla. Esto creará espacio para su segunda pantalla (a la izquierda de la pantalla principal), pero se comportará como un monitor (por lo que al maximizar no funcionará correctamente sin fakexinerama, que también tiene sus problemas).

Así que ahora tenemos una nueva superficie de escritorio a la derecha de la pantalla principal y necesitamos proyectarla en una pantalla física diferente. Tenemos dos formas de hacerlo.

El modo Chromecast

Pero espera: ¡no tengo un Chromecast! Acabo de hacerme con un ODROID-N2 con Android TV ... Bueno, ¡estás de suerte! Tiene un Chromecast, pero necesitas instalar una aplicación de Play Store llamada Cast Receiver

(https://play.google.com/store/apps/details?id=com.softmedia.receiver.castapp&hl=en) que actúa como un Chromecast y puede recibir transmisiones de aplicaciones compatibles con Chromecas (https://forum.odroid.com/viewtopic.php?f=178&t=37501). Ten en cuenta que la aplicación es una demo, pero para algunas cosas (como la transmisión de Youtube) no impone límites de tiempo

Entonces, lo lógico es usar la función de la pestaña Cast de Chromium para transmitir la segunda pantalla al dispositivo Chromecast. Veamos esto en acción. Abre Chromium, selecciona el menú de tres puntos, seleccione Cast ... y si estás en la misma LAN con tu Chromecast, debería verlo en la lista (Figura 2).

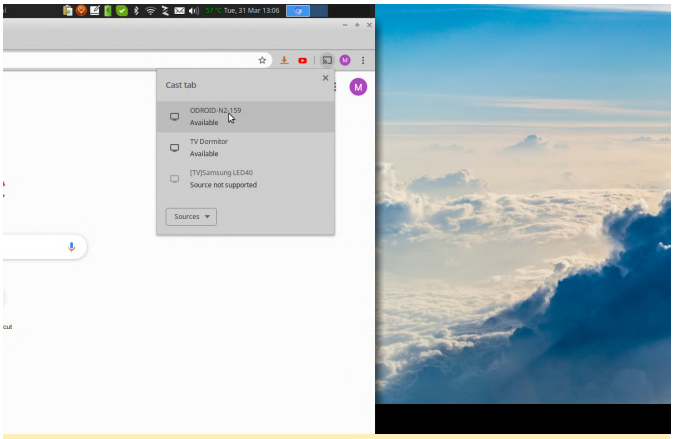


Figura 2. Lista de dispositivos Chromecast en la LAN

Si haces clic en el botón Sources... puede seleccionar entre la pestaña Cast y el escritorio Cast. Si selecciona Cast desktop, debería aparecer una selección de aplicaciones o las pantallas que desea transmitir. Si solo quieres lanzar una aplicación, está bien, pero nosotros queremos transmitir la pantalla virtual. Desafortunadamente, parece haber un error de Chrome que nos impide hacerlo: dirígete al escritorio combinado como una pantalla, no como dos pantallas independientes.

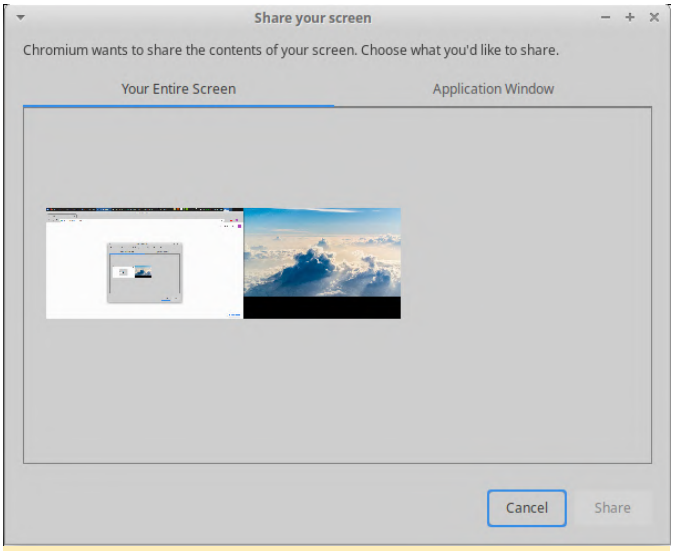


Figura 3. Selección de pantalla

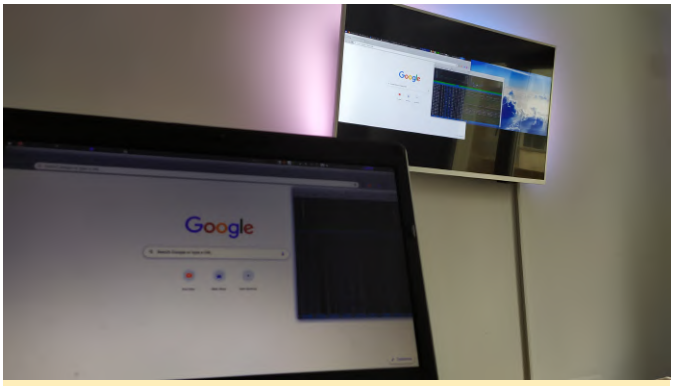


Figura 4. Transmisión de pantalla extendida a través de Chrome: difícilmente útil

Por lo tanto, actualmente el envío desde Chrome no es posible, aunque podría cambiar en el futuro. La calidad era buena, el rendimiento estaba bien y solo había un retraso de medio segundo entre la entrada del ratón y la retroalimentación visual. No es adecuado para juegos, pero está bien para la mayoría de las tareas de oficina.

De modo que, el plan B es el que usa mkchromecast para emitir un área de la pantalla. Puedes instalarlo con

```
$ sudo apt-get install mkchromecast
```

Puedes ejecutarlo con el parámetro --discover para obtener los nombres de los Chromecasts en tu red (consulta la figura 5).

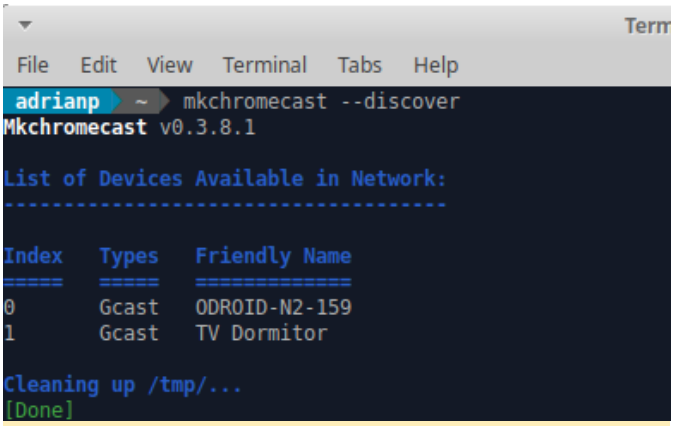


Figura 5. Descubriendo Chromecasts en su LAN

Conociendo el nombre, puede escribir un comando más complicado para usar ffmpeg y usar X11 con un tamaño específico y desde un offset específico y transmitir el video a tu Chromecast:

```
$ mkchromecast -n "ODROID-N2-159" --video --
command 'ffmpeg -f
x11grab -r 15 -s 1280x720 -i :0.0+1600,0 -vcodec
libx264
  -preset ultrafast -tune zerolatency -maxrate
10000k
  -bufsize 10000k -pix_fmt yuv420p -g 60 -f mp4
  -max_muxing_queue_size 9999 -movflags
  frag_keyframe+empty_moov pipe:1'
```

La mayoría de los parámetros anteriores deben permanecer fijos para obtener la mejor velocidad de transmisión. El parámetro -n te permite seleccionar el chromecast de salida deseado, -r especifica la velocidad de fotogramas, -s representa el tamaño de la pantalla virtual, mientras que: 0.0 + 1600,0 representa el desplazamiento desde donde desea capturar. Este desplazamiento se lee como sigue: lee desde Xserver: 0.0, con un desplazamiento de +1600 píxeles en el eje xy un desplazamiento 0 en el eje y. El valor x debe ser el ancho de la pantalla de tu ordenador portátil en píxeles, para que ffmpeg pueda omitir tu pantalla física. El valor y es 0 porque X11 lee el eje y comenzando desde arriba, hacia abajo.

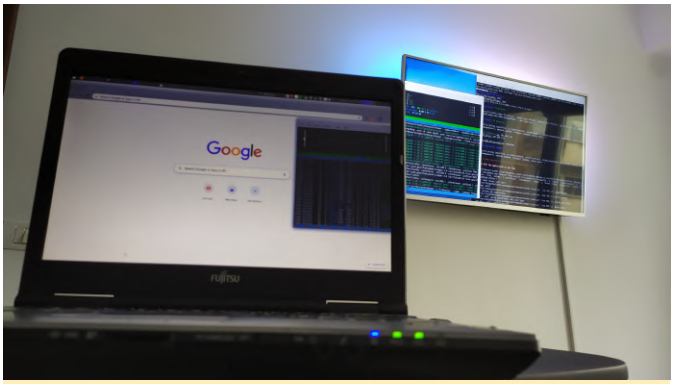


Figura 6. Ampliación del escritorio con Chromecast

Ahora, el resultado se ve mejor. Excepto que el rendimiento no está cerca de lo que Chrome puede hacer. A pesar del ajuste del parámetro ffmpeg, debido a los búferes de red, los búferes de compresión, etc., hay un retraso de 5-6 segundos entre su acción y la respuesta de la pantalla. Por lo tanto, esto solo es adecuado como una segunda pantalla para leer documentación, el correo electrónico y las cosas que no requieren interacción (por ejemplo, ver registros).

El método VNC

Podemos hacerlo mejor. ¿Qué tal si proyectamos la pantalla a través de VNC? Esto es lo que intenté en mi artículo anterior, pero de una manera enrevesada que no funcionó tan bien porque tuve que capturar/transportar y renderizar la mitad del escritorio fuera de la pantalla. Si hubiera pasado más tiempo leyendo el manual de x11vnc (**http://www.karlrunge.com/x11vnc/x11vnc_opts.html**), ¡habría descubierto la opción -clip que hace exactamente eso! La idea es iniciar un servidor VNC que esté recortado al tamaño de la pantalla virtual y, en el lado del televisor, usar un programa de visualización VNC para mostrar el contenido del servidor. La gran ventaja es que puede "transmitir" a cualquier sistema habilitado para VNC, por lo que no necesita ejecutar Android en su Odroid, y también, si su televisor inteligente tiene una aplicación de cliente VNC, puede usarse directamente.

Creé un pequeño script shell que crea la pantalla virtual y también inicia x11vnc en segundo plano sin autentificación. Configurarlo para que se ajuste a tus necesidades:

```
$ cat new_720p_screen.sh
#!/bin/bash

# calculate the desired modeline with gtf:
# gtf 1280 720 60
#
#  # 1280x720 @ 60.00 Hz (GTF) hsync: 44.76 kHz;
pclk: 74.48 MHz
#  Modeline "1280x720_60.00"  74.48  1280 1336
1472 1664  720 721 724 746  -HSync +Vsync

/usr/bin/xrandr -d :0 --newmode "1280x720_60.00"
74.48  1280 1336 1472 1664  720 721 724 746  -
HSync +Vsync

/usr/bin/xrandr -d :0 --addmode VIRTUAL1
"1280x720_60.00"
/usr/bin/xrandr -d :0 --output VIRTUAL1 --right-of
LVDS1

#start x11vnc
x11vnc -forever -bg -geometry 1280x720 -shared -
noprimary -auth /var/run/lightdm/root/:0 -display
:0 -clip 1280x720+1600+0 -threads -noxdamage
```

Los parámetros interesantes de x11vnc que necesitarás son: -geometry establece la resolución de la sesión de VNC de destino y debe coincidir con el tamaño de tu pantalla virtual -clip define una resolución de destino (1280x720) y un desplazamiento desde la pantalla actual (1600 píxeles desde el borde del servidor X11, en el eje x, 0 píxeles desde el eje y). El desplazamiento debe coincidir con el tamaño de la pantalla principal si se extiende hacia la derecha, y debe ser negativo y coincidir con el tamaño de la pantalla virtual si se extiende hacia la izquierda de la pantalla principal

-threads y -noxdamage mejoran la capacidad de respuesta del video

Una vez que ejecute esos comandos, puede usar cualquier cliente VNC para conectarse a la dirección IP de su PC en el puerto 5900 y ver solo la pantalla virtual. Si estás en Android TV, puede usar TruVNC (**https://play.google.com/store/apps/details?id=com.mm.truvnc.lite&hl=en**) que funcionó realmente bien en mi caso, cualquier soporte de cliente VNC debería funcionar perfectamente.

En términos de rendimiento, ¡es genial! Recibo menos de 1s de retraso en wifi y una respuesta mucho más rápida al usar una conexión por cable, ¡así que estoy muy contento con ello! Ejecuté glmark2 en la pantalla virtual y se renderizó sin problemas sobre VNC, con el efecto de desgarro ocasional debido a la opción de noxdamage (de lo contrario, se bloquea). La reproducción de video también es suave, excepto por algunos parpadeos. El uso de la CPU tampoco es tan alto. De modo que, pruébalo, tu gustará tu nuevo escritorio expandido.

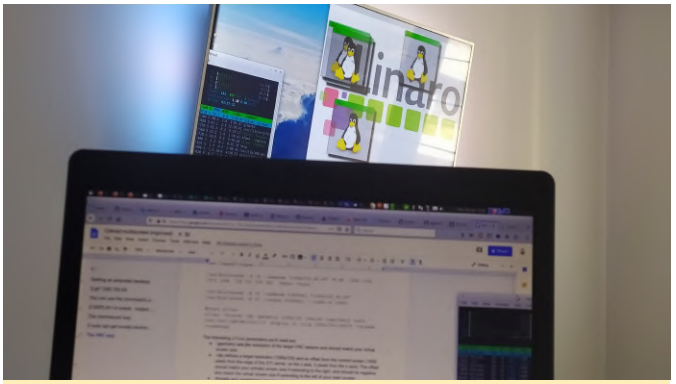


Figura 7. Expansión a través de VNC

Para mí, lo usaré así durante la cuarentena, y cuando vuelva a trabajar, configuraré un ODROID-XU4 con un viejo monitor 1280x1024 como mi tercer monitor, ¡así seré la envidia de la oficina! Para comentarios, preguntas y sugerencias, visite la publicación original en **https://forum.odroid.com/viewtopic.php?f=53&t=38409**.

Contribuye con la Investigación de Coronavirus Usando Rosetta@home para Ayudar a Encontrar una Cura

May 1, 2020 By Rob Roy Linux

Ahora es posible usar tu ODROID de 64 bits para ayudar con la investigación de Coronavirus. Gracias a una nueva actualización de la aplicación de Rosetta@home, hecha posible por la comunidad de desarrollo de Arm. Necesitarás al menos 2 GB de RAM y un sistema operativo de 64 bits (Linux o Android).

Empezando con Android

Para empezar a usar Android, simplemente descárgate la aplicación BOINC de Google Play Store y elije Rosetta@home de la lista de proyectos. Ejecute la aplicación y cree una cuenta nueva o usa una existente si tiene una. Luego, espera a que lleguen las unidades de trabajo.

Empezamos en Linux

Para empezar en Linux, primero asegúrese de que todo esté actualizado:

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

Luego instala el cliente boinc y la interfaz de usuario de texto boinc:

```
$ sudo apt-get install boinc-client boinctui
```

Luego ejecuta bonctui:

```
$ boinctui
```

Presiona "F9" y dirígete a "Projects", selecciona "Add Project" y elije "Rosetta@Home". Selecciona una cuenta existente o crea una nueva y espera a que lleguen las unidades de trabajo, luego simplemente déjalo funcionar.

¿Qué es lo que hace?

Rosetta@Home utiliza la plataforma BOINC para aprovechar miles y miles de ordenadores para ejecutar trabajos informáticos distribuidos (grandes trabajos informáticos divididos en unidades de trabajo más pequeñas para ejecutarse en muchos

procesadores diferentes) en función de la secuencia de ADN conocida del Coronavirus (también como otros virus relacionados con otras enfermedades)" para predecir la estructura de proteínas importantes para la enfermedad, así como para producir mini-proteínas nuevas para ser utilizadas como potenciales terapias y diagnósticos, como el que se muestra arriba que está vinculado en parte a la proteína espiga del SARS-CoV-2 " (**https://boinc.bakerlab.org/rosetta/forum_thread.php?id=13702**)

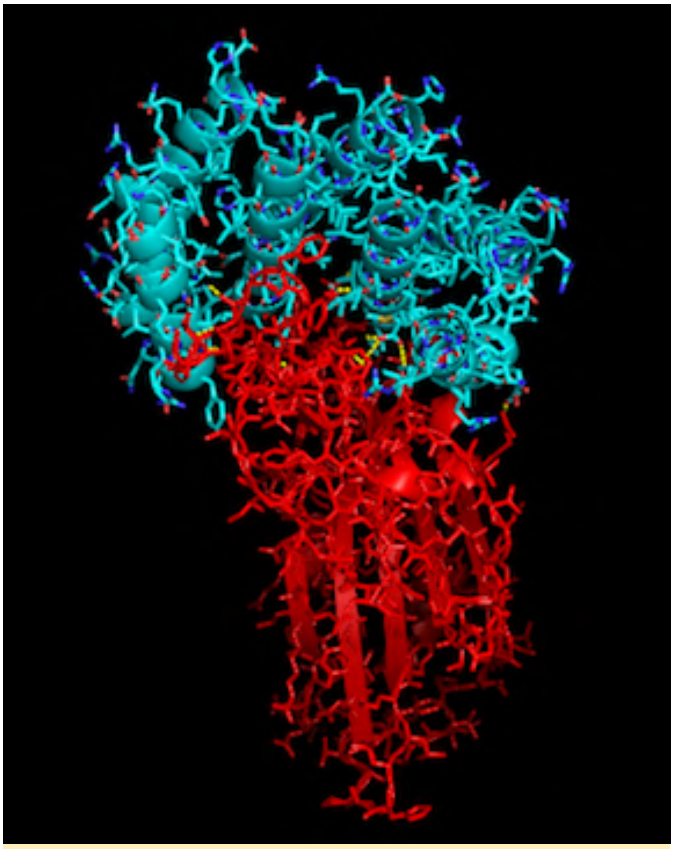


Figura 1 - JHR vs Covid

Estos resultados hacen posible la investigación dirigida y acelerada de vacunas y antivirales.

La Unión hace la Fuerza

A fines de marzo, había cerca de 100,000 hosts de más de 140 países, lo que permitía aproximadamente 1,26 petaflops de potencia informática. Ese es el verdadero rendimiento del superordenador, donado a investigadores por miles de personas en todo el mundo, para ayudar a abordar un problema global. Una causa que bien vale los ciclos de cálculo de reserva de nuestros SBCs ODROID.

Teléfono Móvil Avanzado ODROID-GO: Un Teléfono Personalizado y Codificado

May 1, 2020 By @mameise ODROID-GO Advance, Tutoriales

Recientemente, he decidido construir mi propio teléfono móvil con un ODROID-GO Advance usando un módulo SIM800L que incluía un altavoz y un micrófono. Gracias al amplio espacio del interior de la carcasa, la instalación de hardware fue bastante fácil. Para esta compilación, utilicé una imagen de Debian Buster con el SIM880L conectado al UART2 del ODROID-GO Advance.

Figura 1 - ODROID-GO Advance con teclado

Figura 2 - Vista lateral con recorte para SIM800L

Figura 3 - Tarjeta SIM800L y antena

Inicialmente, lo intenté con minicom para comunicarme con /dev/ttyFIQ0 pero no obtuve respuesta. También probé el conector de 10 pines (UART1) pero también tuve mala suerte. Tras un poco de ayuda del foro Hardkernel, aprendí que era necesario realizar cambios en el archivo dtb del dispositivo. Los cambios necesarios incluyeron deshabilitar el "fiq-debugger" que usaba UART2 y habilitar ese puerto UART como un puerto serie común. Además, la entrada del dispositivo fiq-debugger fue eliminada del archivo boot.ini. Tras realizar estos cambios y reiniciar, se pueden enviar comandos AT con una respuesta desde el módulo SIM.

Figura 4 - Primera aplicación de prueba en funcionamiento

Después de algo más de trabajo, se creó una interfaz básica para almacenar información de contacto y gestionar llamadas.

Figura 5 - Menú Phone OS 0.04

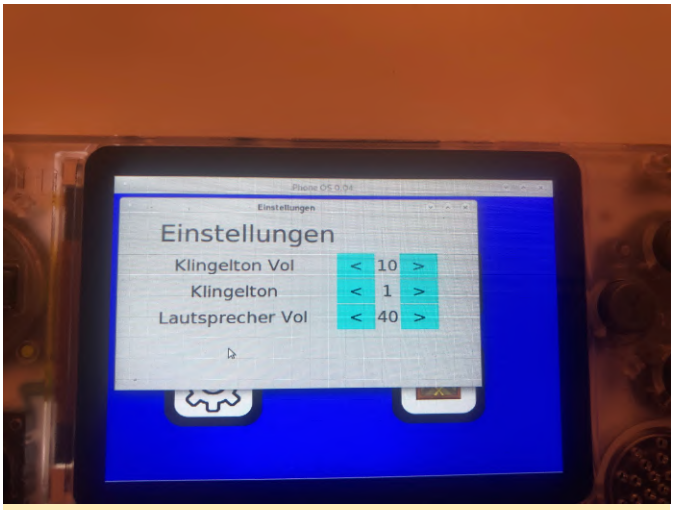


Figura 6 - Página de opciones/configuración

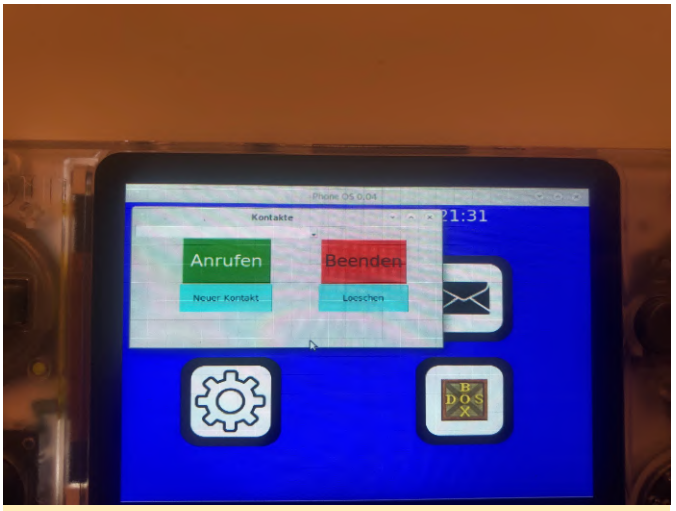


Figura 7: Selección de contactos y opciones para llamar y finalizar la llamada

Para obtener más información, el hilo original del foro está disponible en https://forum.odroid.com/viewtopic.php?f=193&t=38248.