Banco Pruebas ODROID • Nagios • Programación • Juegos Clásicos Linux

ODROID Magazine

Año Cinco
Num. #58
Oct 2018

Sistema de *Información y ocio* para conducir

UTILIZA TU ODROID-C2 PARA DESPLAZARTE Y EXPERIMETAR VIAJES DE PELICULA

BANCO DE PRUEBAS ODROID: UNA ZONA REMOTA EXPERIMENTAL PARA TESTEAR ODROIDS

ODROID-GO INALAMBRICO: AÑADE CARGA INALAMBRICA A TU NUEVO DISPOSITIVO PORTATIL

Juegos Linus en ODROID: Nostalgia por los Juegos Parte 1 – Juegos a los que Siempre Vuelvo

October 1, 2018

Quisiera hablar de los juegos a los que solía jugar y de los juegos a los que sigo jugando incluso hoy día y que ejecuto de hecho en el ODROID.

Sistema de Información de Ocio para Vehículos: Utilizando un ODROID-C2

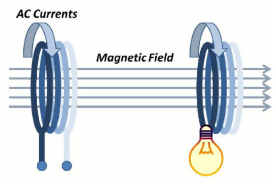
October 1, 2018

Me enorgullece presentar mi “Sistema de información y entretenimiento para vehículos ODROID”

Banco de Pruebas ODROID

October 1, 2018

Hemos configurado una zona experimental ODROID remota para alguien que quiere medir, pre-visualizar y probar el rendimiento de los SBC ODROID. Los SBC ODROID de nuestro banco de pruebas están conectados vía ethernet Gbit y están abiertos al público. El sistema experiencial ofrece la posibilidad de evaluar el rendimiento, un ▶



Carga Inalámbrica: Añadiendo un Sistema de Carga Inalámbrica Qi al ODROID-GO

October 1, 2018

Quiero demostrar que este dispositivo era mejor de lo que la gente decía, así que hice mi primer video sobre GO: un video con un mini-análisis sobre ODROID-GO. Después de muchos más videos, he decidido escribir un artículo para que sea publicado, algo que nunca había hecho antes. Así que, ▶

Campamento de Programación – Parte 5: Leer el Voltaje de la Batería Integrada en el ODROID-GO

October 1, 2018

En esta guía vamos a aprender cómo obtener el estado de la batería con Arduino. La pantalla LCD mostrará el voltaje restante de la batería en voltios.

Presentado NEMS Linux: Parte 1 – el Servidor de Monitorización Profesional Nagios para Dispositivos ODROID

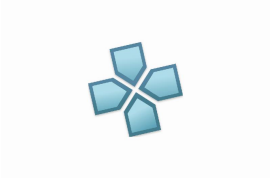
October 1, 2018

NEMS Linux ha evolucionado para ser lo que pienso que es la mejor experiencia de Nagios disponible. Como usuario de Nagios, este es el servidor de Nagios que siempre he anhelado. A medida que NEMS ha seguido creciendo, empecé a buscar una plataforma más potente que la Raspberry Pi. Ahí ▶

Conceptos Básicos de BASH: Introducción a BASH – Parte 5

October 1, 2018

Nuestra aventura con la programación continúa con más pruebas, declaraciones if, entradas y funciones; También haremos cálculos desde dentro de BASH. Primero, vamos a echar s un vistazo a la manipulación del nombre de archivo, ya que parece ser una de las cosas más comunes que son necesarias en los ▶️



Driver de Video GBM

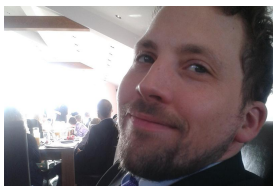
October 1, 2018

Esta es una guía para instalar los drivers de espacio de usuario para activar GBM y compilar emuladores de juegos retro.

Campamento de Programación – Parte 6: Generar Sonido desde el Altavoz del ODROID-GO

October 1, 2018

Vamos a aprender cómo usar la salida DAC como un generador de tonos



Conociendo un ODROIDian: David Knight

October 1, 2018

Por favor háblanos un poco sobre ti. Vivo en Newcastle, Reino Unido, trabajando como optometrista en el sector de la cirugía refractiva. Mi trabajo diario consiste en lidiar con pacientes que se han sometido a una cirugía ocular por láser o de cataratas. Fui muy aplicado e introvertido en mi ▶️

Juegos Linus en ODROID: Nostalgia por los Juegos Parte 1 – Juegos a los que Siempre Vuelvo

October 1, 2018 By Tobias Schaaf Juegos, ODROID-XU4

La última vez hablé bastante de mi pasado, cómo crecí y cuales fueron los sistemas que más me influyeron durante mi infancia. Este y otros hechos han generado un internso debate en los foros sobre un tema al que he llamado “Nostalgia por los juegos”. He pensado que éste sería un buen tema a tratar, pero centrándome en los ODROID y viendo cómo puedo proyectar mi nostalgia en el entorno ODROID. Por todo ello, me gustaría hablar de los juegos a los que solía jugar, de los juegos a los que sigo jugando incluso hoy día y que ejecuto de hecho en el ODROID.

Loom

Creo que he jugado a casi todos los juegos de aventuras de LucasFilm/LucasArts, especialmente a todos los que se han creado con el motor SCUMM, empezando por Maniac Mansion, que dio nombre al motor. Considero que los juegos de aventura de LucasArts son los mejores juegos a los que he jugado y me alegro de haber tenido la oportunidad de experimentar con estos juegos de mayor. Es muy posible que muchos jugadores de “consolas” se perdieran la mayoría de estos juegos, ya que las aventuras de apuntar y disparar estaban disponibles en su mayoría para “ordenadores” y no para consolas, y simplemente puedo decir que, si nunca has jugado a los juegos de aventura de LucasArts a lo largo de tu vida, es que te has perdido un buen cacho de historia de los juegos.

Monkey Island, Indiana Jones, Loom, The Dig: estos juegos realmente me cambiaron y me transportaron a mundos maravillosos, mágicos y un tanto extraños, de niño e incluso de mayor hoy día.

Loom, que también es un juego de LucasFilm, siempre ocupará un lugar especial en mi corazón, lo cual puede ser un tanto difícil de entender, ya que comercialmente y desde un punto de vista crítico, el

juego no tuvo tanto exitoso como otros juegos de LucasArts. Pero para mí, el juego es extremadamente fascinante, y muchos otros juegos de LucasArts hacian referencia a este juego con bastante frecuencia a modo de sorpresas ocultas (o no tan ocultas).

Figura 01 – Referencia a Loom en The Secret of Monkey Island (referencia muy ingeniosa y oculta ... no)

Loom es un juego repleto de "gremios": tejedores, fabricantes de vidrio, herreros, etc. Formas parte del gremio de los tejedores y éste inventó "The Great Loom", que permite a los tejedores tejer los hilos de la realidad en sí misma, lo cual, por supuesto, estaba prohibido. Los tejedores también tejian los grandes tapices que muestran la historia del gremio. Aunque los tejedores no usaban el Loom, eran tejedores de hechizos muy buenos, los cuales podían lanzar con la ayuda de una rueca y con combinaciones de notas musicales.

La historia se desarrolla en torno a Bobbin, un hijo de Loom, ya que le debe su existencia en sí a "The Great Loom", cuestión que se explica en un magnífico audio incluido en un casete si compraste una versión específica en el pasado. Hoy en día, puedes encontrar el audio en YouTube si quieres escucharlo. Terminas siendo el último de tu gremio e intentas encontrar el resto de tu clase, es el momento en el que empiezas a controlarlo. Aprendes hechizos mágicos a lo largo de tu camino y debes aprender cómo y cuándo usarlos para resolver diferentes puzles.

Figura 02 – Perturbar a los muertos es una muy mala idea Bobbin. Te advirtieron que no lo hicieras, ¡no lo intentes!

Lo que más me fascinó de este juego es el uso real de "magia", que es algo que no encuentras en ningún otro juego de su tiempo. Había juegos como "Simon the Sorcerer", en los que nunca se lanzaba un hechizo pese a su nombre, o la serie de Legend of Kyrandia, que trata de jóvenes magos y una vez más no usas nada de magia (un montón de pociones, pero nada de "magia real"). Siempre me ha resultado muy confusa esta cuestión, hay muchos juegos sobre los llamados "magos o brujos" pero en ninguno de ellos realmente lanzan hechizos. Incluso mucho más tarde, los llamados juegos de Harry Potter, por ejemplo, son más del tipo "shooter" en lugar de usar HECHIZOS reales para resolver algo. Una varita que dispara destellos para matar monstruos y dice "pui pui" no es "magia" en mi opinión, simplemente es un estúpido shooter con un toque mágico.

Loom era diferente: contabas con hechizos que podían abrir puertas, hechizos que permitían "rellenar" ciertas cosas (una copa de vino, por ejemplo), hechizo que incluso podían convertir la paja en oro. Tenías más de 15 hechizos (llamados borradores) que podrías aprender dentro del juego y que usabas para resolver acertijos.

Realmente me gustaba la historia de este juego y cómo se desarrollaba. Originalmente fue planteado para ser una trilogía, pero fue cancelado tras el lanzamiento de esta primera parte. También me encanta su clásica banda sonora de Swan Lake. Solía jugar al juego al menos una vez al año, y solo me llevaba entre 30 y 60 minutos terminar el juego. Si quieres jugarlo hoy día en los ODROIDs, puedes usar

ScummVM para ejecutarlo perfectamente en ODROIDs. Hay diferentes versiones de este juego, incluyendo una versión de CD “talkie” en el que todo el texto fue doblado en inglés.

The Curse of Monkey Island

Créeme, me encantan todos los juegos de Monkey Island, y me divertí bastante con The Secret of Monkey Island y Monkey Island 2: LeChuck’s Revenge. Recomiendo encarecidamente estos juegos a cualquiera. Me enganché a la historia de Guybrush cuando pisó Mêlée Island™ por primera vez y dijo que quería ser un pirata.

Figura 03 – The Curse of Monkey Island

Sin embargo, si nunca has llegado a jugar a ningún juego de Monkey Island a lo largo de tu vida y solo tienes tiempo (por el motivo que sea) para jugar a un único juego de Monkey Island, te recomiendo The Curse of Monkey Island. Aunque The Curse of Monkey Island ni siquiera es un título “oficial” de Monkey Island, ya que no forma parte de la pluma de Ron Gilbert, aun así, es un juego increíble.

Figura 04 – Así es como todo comenzó en 1990.

Monkey Island 1 y 2 son lo más clásicos y se reconvertidos hace un par de años en una edición remasterizada que mejoraba los gráficos e incluían una completa interpretación de las voces que no existía en el juego original por aquel entonces, pero no son compatibles con ScummVM, lo cual significa que no puedes ejecutalos en ODROIDs. Puesto que los originales Monkey Island 1 y 2 estaban muy pixelados, no han envejecido tan bien, aunque son divertidos si te gustan los viejos gráficos de 8 bits. Algunos juegos más recientes intentan replicar el aspecto de esa época, pero Monkey Island 3 (The Curse of Monkey Island) ya contaba con gráficos 640×480 de “alta resolución” trazados con un magnifico estilo cómico, que incluso en la actualidad presentan un aspecto sorprendente.

Figura 05 – Monkey Island 3

Figura 06 – Monkey Island 3

ScummVM no tiene problemas para escalar estos gráficos, y parece que fueron creados para 1080p como lo fueron para 640×480, lo que significa que puedes disfrutar del juego hoy día tal y como lo era en 1997 cuando salió. Realmente es algo que nunca he llegado a entender. Hay ciertos tipos de gráficos que no “envejecen” (el estilo del cómic es el mejor ejemplo de ello), aunque solo hay unos pocos juegos

que realmente lo utilizaron. Todavía se ven increíbles 20 años después, y podrías pensar que se trata de un nuevo juego que acabas de comprar.

La historia y el diálogo son tan divertidos que vale la pena pasar tiempo en el juego simplemente haciendo todas las preguntas posibles para escuchar todos diálogos y comentarios divertidos que tiene el juego. Las interpretaciones de las voces son excelentes, y la música de Michel Z. Land que compuso toda la música para los juegos Monkey Island es muy representativa y realmente se ajusta al entorno caribeño. El juego toma referencias de los anteriores juegos (Monkey Island 1 y 2), y también coge personajes de estos juegos, así que, si quieres entender cada pequeño chiste del juego, también deberías jugar a Monkey Island 1 y 2. Este es uno de mis juegos favoritos de todos los tiempos y se ejecuta perfectamente en ODROIDs, así que puedes revivir toda la experiencia del pirata caribeño utilizando ScummVM.

Dune 2

Dune 2 fue mi primer juego de estrategia en tiempo real (RTS). De hecho, es conocido como "El abuelo de todos los juegos de estrategia en tiempo real", y lo jugué en mi fiel Amiga. El juego era increíble: Westwood se abrió camino en el género de estrategia en tiempo real con Dune 2, y clásicos como la famosa serie Command and Conquer aparecieron como resultado del éxito de Dune 2.

Dune 2 tiene todo lo que esperas de un juego de estrategia en tiempo real. Recolectas recursos (especias), construyes tu base y un ejército, te diriges a aplastar a tu enemigo y su base. Una de las cosas que recuerdo de la versión de Amiga es que ya contaba con algunas muestras de voces que te decían, por ejemplo, cuándo se completaban tus edificios.

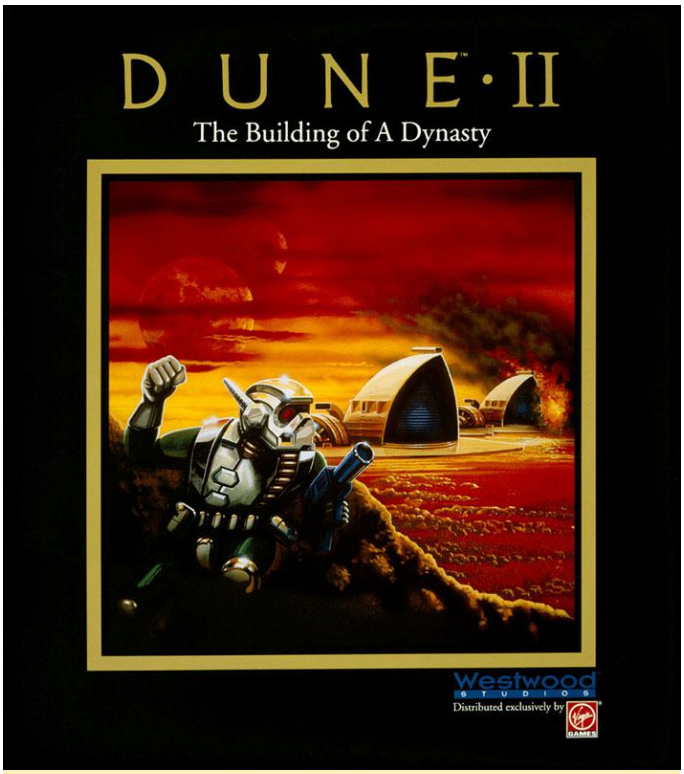


Figura 07 – Dune 2 The Building of A Dynasty

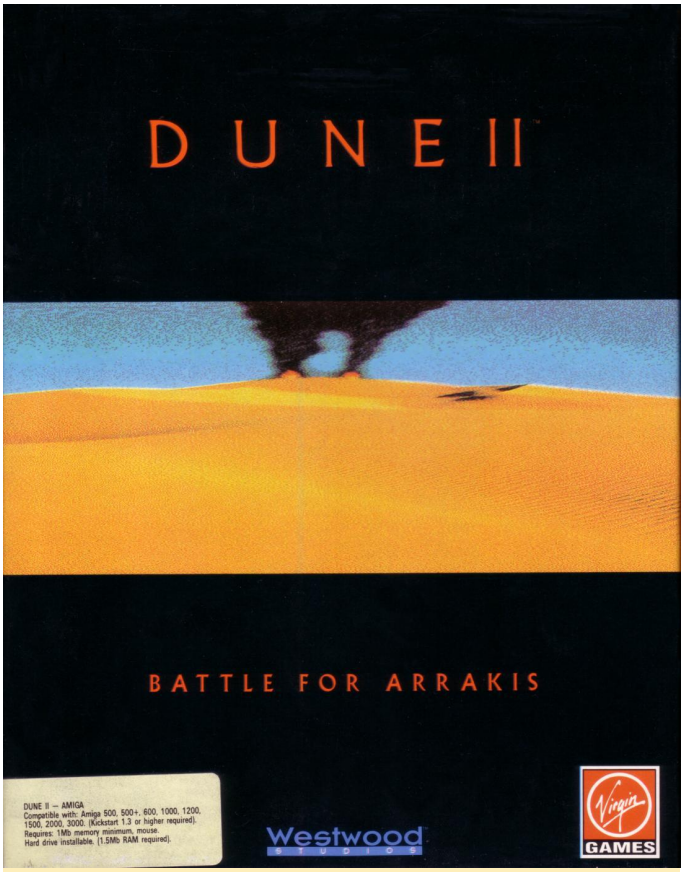


Figura 08 – Dune 2 The Building of A Dynasty

En este juego, tenías 3 facciones diferentes donde elegir. Todos ellas tenían un conjunto de unidades que eran idénticas para cada facción, y algunas otras que eran únicas para cada facción. Cada una con sus ventajas y desventajas, aunque prefería el misil

nuclear de los Harkonnen frente a los Fremen de los Atreides (armas especiales de Palace).

Cuando salió por primera vez, Dune 2 supuso una auténtica revolución. Controlabas varias unidades a la vez sobre el mapa y también lo hacia el enemigo. Tenías que concentrarte en la construcción, en la recolección de recursos y en tus ataques/defensas al mismo tiempo. Cosas que son normales hoy día eran bastante novedosas por aquel entonces.

Figura 09 – Gráficos originales de Dune 2

Aunque Dune 2 ha envejecido un tanto bien, sus controles no son muy intuitivos. No podías seleccionar varias unidades a la vez para darles ordenes, tenías que dar comandos a cada unidad individualmente. Imagínese mover un ejército de 30 o más unidades de un lugar a otro (sí, lo hacíamos de una la forma bastante complicada "volviendo a los viejos tiempos").

Afortunadamente, hoy en día existen varios remakes de Dune 2 que incluyen mejores controles, lo cual te permite mover ejércitos enteros sobre la marcha, y otras mejoras, como crear colas de puntos de reunión y, en ocasiones, mejores gráficos (mayores resoluciones y filtros para mejorar el aspecto). Personalmente prefiero Dune Legacy por todas estas cuestiones, que está disponible en ODROID y funciona bastante bien.

Dune Legacy

De vez en cuando pruebo algunas actualizaciones de Dune Legacy y termino jugando durante horas y días, ya que el juego sigue siendo muy bueno como juego de estrategia, y me suelo perder entre sus batallas.

Figura 10 – Dune legacy

Aunque solemos hablar de Dune 2, también existe un juego llamado "Dune". Es una mezcla de estrategia y aventura, ya que sigue la historia de Paul Muad'dib de House Atreides al mismo tiempo que aprendes de los fremen y sus costumbres. Este juego también es muy bueno, y tiene una de las mejores bandas sonoras, especialmente la versión CD para DOS con escenas de video y renderizadores 3D. Es un magnifico juego que funciona bastante bien en DOSBox en ODROIDs, auqnue creo que hable de este juego hace un tiempo en mis artículos sobre juegos DOS.

Figura 11 – Dune, una legendaria historia de recolección de especias

UFO Enemy Unknown / XCOM – Ufo Defense

Este es realmente especial, no recuerdo cuando empecé a jugarlo. Sé que existió en Amiga, pero creo que lo disfruté más intensamente en el PC bajo DOS. Absolutamente adoro este género, aunque es un género un tanto difícil de definir. La mayoría lo definen como un juego de estrategia con tácticas basadas en turnos, o de simulación de administración con tácticas basadas en turnos, aunque creo que es algo mucho más específico lo que me fascina de este género de juegos y no conozco nada que se le

parezca, a parte de los juegos XCOM y variantes que aparecieron con el paso del tiempo.

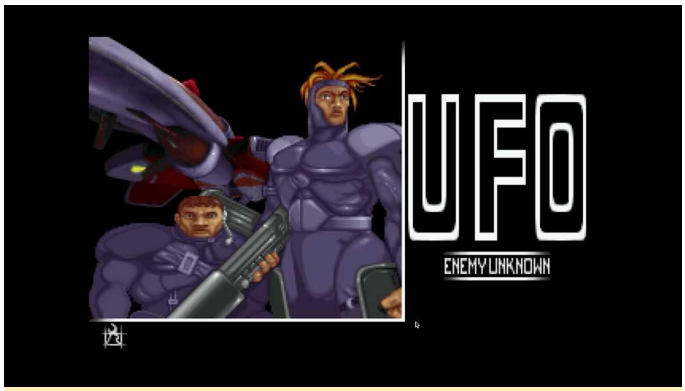


Figura 12 – UFO Enemy

El juego es bastante complejo y requiere de bastante tiempo para jugarlo, dominarlo y completarlo, pero merece la pena. Los alienígenas están aquí, llegaron a la Tierra, volando en sus naves espaciales extraterrestres, secuestrando a los humanos, aterrorizando las ciudades y quién sabe qué más hayan planeado. Los diferentes gobiernos del planeta se han dado cuenta de que no están en condiciones de enfrentarse a un enemigo como este por su cuenta y han acordaron crear una fuerza unida "XCOM" para derrotar a los alienígenas.

Para ello, cada país accedió a financiar al grupo XCOM con dinero y personal para lograr este objetivo y es aquí donde tú entras en juego. Eres el comandante del XCOM y tus decisiones afectarán al resultado de este contexto. Has distribuido tu tiempo y recursos para montar una defensa contra los intrusos. Tienes que explorar nuevas tecnologías, que implica el cómo derrotar al enemigo. Tienes que construir tus propias armas, puesto que estas nuevas tecnologías aún no existen y no puedes simplemente comprarlas. Tienes que entrenar a tus soldados, equiparlos con armaduras para que se protejan, con armas para combatir al enemigo y con otros objetos como botiquines, granadas o rastreadores de movimiento.

También tienes que equipar a tus propios aviones para derribar a los ovnis enemigos. Tienes que analizar los objetos que recoges y las armas del enemigo. Y por supuesto, aplastar al enemigo sobre el terreno, donde sea que lo encuentres.

Para mí, toda esta combinación es lo que hace que este juego sea tan bueno. El hecho de que investigues nuevas tecnologías construyendo nuevas armas, armaduras y aviones permite mejorar tus posibilidades de supervivencia. La mejora constante, no solo a través de un mejor equipamiento sino también de tus soldados, que mejoran cuanto más tiempo están en el servicio y más participan en misiones y más enemigos eliminan. Creas una conexión con tus soldados, especialmente cuanto más fuertes se vuelven. Perder a un novato realmente no te importará, ya que simplemente puedes reclutar uno nuevo, pero un soldado veterano es muy valioso y perder a uno normalmente te complica bastante la vida, hasta el punto de pensar en volver a cargar el juego y intentarlo de nuevo.

Figura 12 – Gráficos originales de UFO Enemy Unknown, luchando contra los "grays"

También fue uno de los primeros juegos en los que usé un editor hexadecimal para manipular el juego guardado cuando era niño, pero solo para crear más "Elerium", que era un mineral extraño necesario para las armas y los equipos de alta tecnología. Constantemente se me agotaba, así que, sí hice un poco de trampa por aquel entonces.

Lo que siempre más me fascinó de este juego fue la investigación. La evoluación a través de la investigación y el desarrollo de nuevas tecnologías es lo que me mantuvo jugando a este juego durante mucho tiempo.

Figura 13 – Autopsia de uno de los alienígenas más peligrosos de XCOM

Figura 14 – Autopsia de uno de los alienígenas más peligrosos de XCOM

Investigar y leer todos esos detalles sobre los alienígenas y las armas o simplemente los elementos que encontrasbas siempre fue lo más interesante para mí. Es lo que hace que este juego sea único: poder a través del conocimiento. Cuanto más investigues, más oportunidades tendrás de vencer al enemigo. Hoy en día, si quiero jugar al juego, uso OpenXcom en el ODROID, ya que tiene muchas características adicionales y permite mods, contenido adicional y diferentes configuraciones que mejora notablemente la experiencia de juego.

Descubrí que OpenXCom funciona estupendamente en los ODROIDs. Me gustan las opciones que ofrece, como deshabilitar las características del juego original que lo hacían muy difícil (por ejemplo, combates de noche) o simples mejoras como hacer clic con el botón derecho en la flecha hacia arriba de la pantalla de investigación para asignar todos los investigadores disponibles a cierto proyecto de investigación, o la posibilidad de "producir y vender" (producir artículos que no están disponibles en el mercado, por ejemplo, rifles láser y venderlos para obtener beneficios). OpenXcom te permite jugar al original UFO Enemy Unknown, así como al segundo juego llamado "Xcom Terror from the Deep", que es prácticamente igual al primero, pero en esta ocasión luchas contra alienígenas que proceden del mar y sugen de debajo del agua.

OpenXcom también tiene muchos mods que introducen nuevos escenarios en el universo, o crean universos completamente nuevos. Combinado con la posibilidad de reproducir música y sonidos mejorados, añadir más armas para investigar, etc., OpenXcom es realmente único y vale la pena tenerlo en ODROID.

Figura 15: Altas resoluciones, solo es una de las muchas características de OpenXCom

Aparte de los juegos originales de UFO Enemy Unknown y Terror From the Deep, también disfruté mucho jugando al remake también llamado XCOM: Enemy Unknown, que lleva el género al moderno sistema 3D.

Figura 16 – XCOM: Enemy Unknown de Firaxis Games

Este juego incluso salió para Android, lo que lo hace compatible técnicamente con ODROID igualmente, aunque me encantaría ver una versión del juego para Linux. También está UFO Alien Invasion (UFOAI) hecha por fanáticos, que está disponible para ODROID en https://forum.odroid.com/viewtopic.php?

f=91&t=2375. Todavía no he jugado demasiado a este juego, pero quiero hacerlo puesto que es bastante fiel a la mecánica original del juego.

Otro gran remake es Xenonauts, que utiliza gráficos 2D sobre una vista isométrica, que es un estilo artístico muy interesante. Estoy casi seguro que este juego podría ejecutarse de forma nativa en ODROID si se recompilara para armhf, ya que solo usa SDL2. Sin embargo, actualmente no está disponible como tal.

Existe una serie más basada en la antigua serie UFO/XCOM que realmente me gusta, también llamada UFO y es conocida como la trilogía UFO o serie Aftermath OVNI, llamada así por el primer juego de la serie. UFO Aftermath, UFO Aftershock y UFO Afterlight son juegos 3D lanzados entre 2003 y 2007, estos juegos me dejaron perplejo una vez más.

A muchas personas no les gustan, ya que no estan tan "basados en turnos" como los juegos XCOM originales, pero como he dicho antes, para mí estos juegos no son de tácticas basadas en turnos, sino mas bien de investigación, desarrollo, mejora y evolución de los personajes.

Los juegos también contaban con toneladas de errores, y la historia (especialmente la de los dos primeros) no estaba muy bien escrita (especialmente el final), pero el sistema de juego era lo que tenia más valor para mí, y llevaron a cabo muchas mejoras en este sentido en comparación con otros juegos en mi opinión.

Figura 17 – UFO Aftermath tiene una configuración mucho más siniestra y esta totalmente 3D, pero lamentablemente sólo se ajusta a una resolución de 1024×768

En los juegos XCOM, comienzas con una base y soldados totalmente equipados y empiezas a buscar "tecnología futura" como láseres y demás al instante. En UFO Aftermath, empiezas en un mundo que fue sorprendido por alienígenas y la humanidad casi fue aniquilada. Las ciudades están derumbadas y los mutantes deambulan atacando a todo el mundo. Apenas tienes armas y encontrar una ametralladora M4 es algo así como hallar un tesoro. Te lleva un tiempo empezar a desarrollar "tecnología futura", y hasta que llega ese momento, te tienes que conformar con las armas que vas encontrando antes de que puedas replicar las tecnologías alienígenas y mejorarla.

En la serie XCOM, debes satisfacer a las diferentes naciones para financiar tus operaciones. Si lo haces mal en un país, pierdes dinero y, a medida que los alienígenas se vuelven más y más fuertes, con el tiempo perderás mucho dinero, de modo que el juego se vuelve más difícil y tienes menos recursos para enfrentarte a ellos, lo cual es un poco molesto. En UFO Aftermath, la sociedad ha colapsado, el dinero no tiene valor, y así es cómo luchas contra una fuerza de invasora global. Tu "moneda" es el tiempo. Cuanto más difícil es un proyecto, más tiempo te lleva desarrollarlo, y puedes cambiar esto dedicando más fuerzas de trabajo a un objetivo común. Puedes hacer esto seleccionando lo que deberían hacer tus diferentes bases: investigar, desarrollar tecnologías o defender con una fuerza militar.

Figura 18 – La materia marrón está cubriendo lentamente la tierra.

El primer juego de la serie tiene la historia más interesante, donde vas evolucionado empezando con viejas pistolas y rifles hasta tecnología alienígena. El segundo juego de la serie es el más fasticioso de los

tres. Se juega después del primero y asumes que las has fastidiado. Ahora ha recuperado la administración de los recursos, y es muy difícil y confuso. Tiene diferentes secciones de humanos que puedes reclutar o convertirse en tu enemigo. Realmente no supone una gran mejora con respecto a UFO Aftermath, que es mi pfavorito de los dos. El tercero, UFO Afterlight, es el más avanzado de los tres y ofrece gráficos en 3D muy mejorados. No se queda en 1024×768 y esta vez, luchas en Marte, nuestro vecino rojo.

Figura 19 – Gráficos altamente mejorados en los muchos y nuevos alienígenas de UFO Afterlight

Espero que algún día podamos jugar también al menos al UFO Aftermath en el ODROID, pero hasta entonces, me encanta jugar OpenXcom y UFOAI. Estos juegos son realmente sorprendentes y si nunca los has jugado, deberías probarlos.

Sistema de Información de Ocio para Vehículos: Utilizando un ODROID-C2

October 1, 2018 By @poptmartone ODROID-C2, Mecaniqueo, Tutoriales

Me enorgullece presentar mi "Sistema de información y entretenimiento para vehículos ODROID". Es la combinación de un ODROID-C2 y una plataforma Arduino capaz de establecer una comunicación CAN con el módulo BCM del coche. Esta función permite que los controles del volante se "comuniquen" con el sistema operativo Android a través de una interfaz por infrarrojos (IR) adicional, como ya sugerí hace algún tiempo. Aquí tienes la lista completa del hardware:

- ODROID-C2
- Montaje de la pantalla VU7 Plus
- Placa de audio Hi-Fi Shield Plus
- Placa de alimentación SmartPower 2
- Módulo USB bluetooth
- Módulo USB wifi
- Módulo USB GPS
- HUB USB con alimentación externa
- Ventilador
- Arduino Nano
- Transceptor CANBUS Niren MCP2515
- Transformador para reducir 12/24 V -> 5V (para alimentar el Arduino y el módulo CAN)
- Varios conectores DuPont macho y hembra

El Arduino Nano trabaja ayudando al C2 y proporciona la lógica de las principales operaciones, el sistema de encendido/apagado, la interfaz IR, la comunicación con el bus CAN y los servicios de control remoto del cabezal de sonido, mientras que en la placa ODROID es donde tiene lugar todo el entretenimiento. Con el fin de minimizar el tamaño físico y adaptarlo todo al espacio designado en el salpicadero de mi automóvil, he creado una placa de control electrónico que contiene todos los componentes electrónicos necesarios, tal como se muestra a continuación.

Figura 01 – Placa de control incompleta

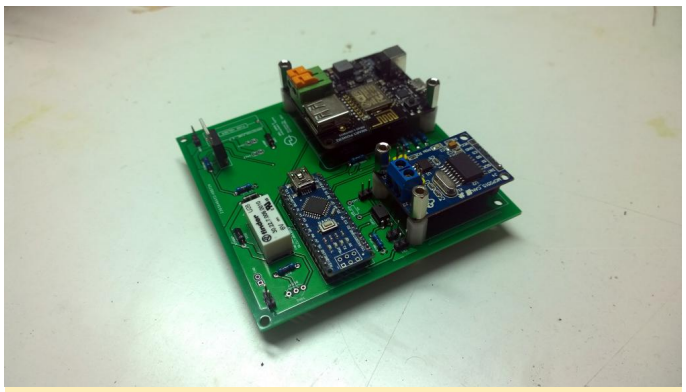


Figura 02 – Placa de control completada

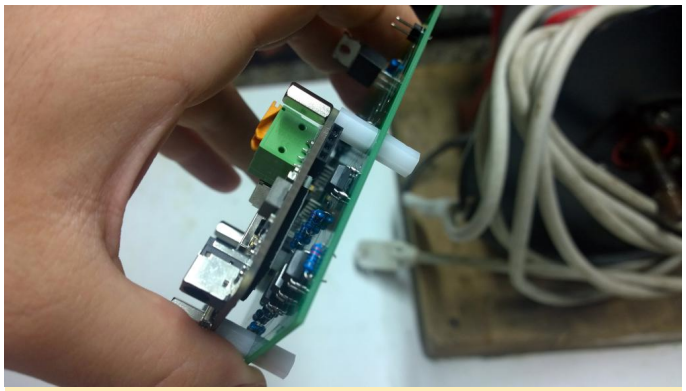


Figura 03 – Vista lateral de la placa de control

Tras desmontar el kit de pantalla VU7 y reorganizar los componentes en una especie de “placa base”, este fue el resultado:

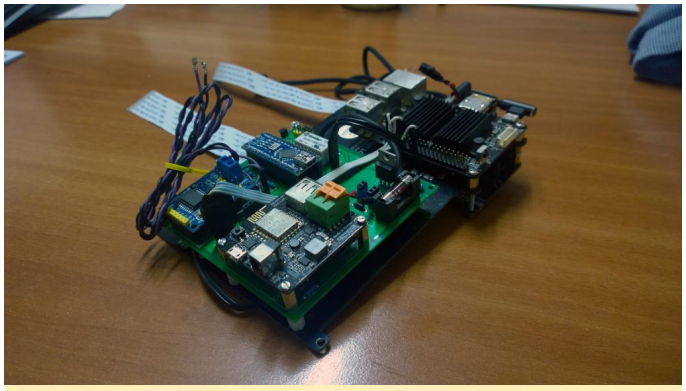


Figura 04 – Todos los componentes conectados

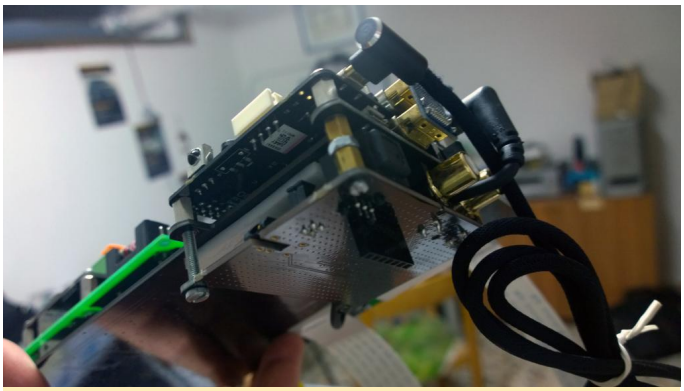


Figura 05 – Primer plano de la placa

A continuación, lo montamos todo dentro del hueco del salpicadero del coche:

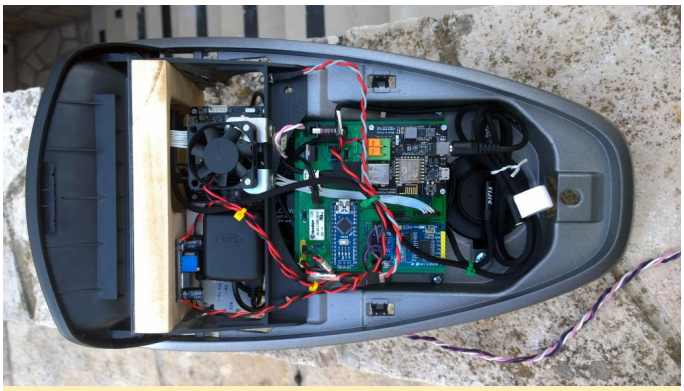


Figura 06 – Placa montada en el salpicadero del coche

Figura 07 – Parte interna del salpicadero del coche

Figura 08 – Frontal visible del salpicadero del coche

Ten en cuenta el LED infrarrojo

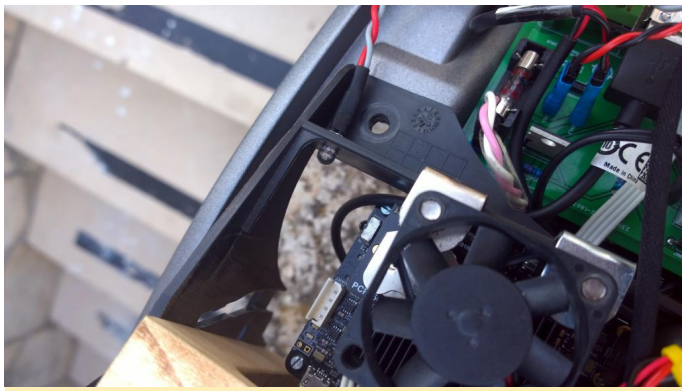


Figura 09 – LED infrarrojo

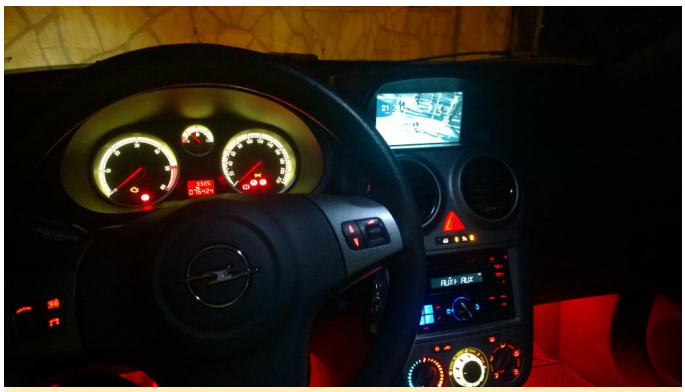


Figura 10 – Todo instalado

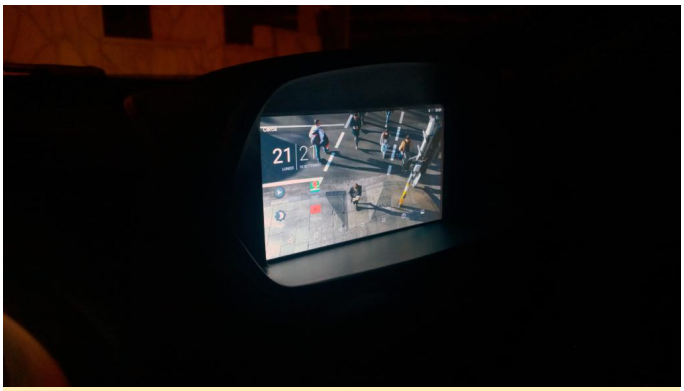


Figura 11 – Instalado y funcionando

Echa un vistazo al video de demostración para ver todo el sistema en acción en https://www.youtube.com/watch?time_continue=79&v=uQf7kH7wbuQ. Para más detalles o hacer preguntas, visita el post de los foros de ODROID https://forum.odroid.com/viewtopic.php?f=140&t=32189.

Banco de Pruebas ODROID

October 1, 2018 By Dongjin Kim Linux, ODROID-C2, ODROID-XU4, ODROID-HC2

Hemos configurado una zona experimental ODROID remota para alguien que quiere medir, pre-visualizar y probar el rendimiento de los SBC ODROID. Los SBC ODROID de nuestro banco de pruebas están conectados vía ethernet Gbit y están abiertos al público. El sistema experiencial ofrece la posibilidad de evaluar el rendimiento, un servidor en la nube y mucho más a través de SSH.

Figura 1: SBCs ODROID en el banco de pruebas conectado a maze.odroid.com

¿Qué dispositivos están disponibles en el banco de pruebas?

Ofrecemos una red totalmente dedicada de 1 Gbps bajo dominio maze.odroid.com y diferentes configuraciones de hardware ODROID para que puedas usar cualquier SBC ODROID en el que esté interesado. Desde este entorno, puedes ver el rendimiento del hardware y la potencia de cálculo. Los futuros SBCs ODROID serán accesibles antes de su lanzamiento.

¿Qué sistema operativo está instalado en los ODROIDs?

Ofrecen Debian Stretch en un contenedor Docker sobre un reciente kernel Linux que Hardkernel mantiene y actualiza. El rendimiento del hardware y la potencia de cálculo no se ven afectados por otro entorno con esta configuración. Si eres nuevo en SBC ODROID, todo el código fuente del kernel de Linux y U-boot está cargado en los repositorios de Github en https://github.com/hardkernel.

¿Qué SBCs ODROID son accesibles?

Los comienzos parecerán un tanto humildes. Dispones de un intérprete de comandos ODROID tras

conectarte vía SSH a una placa a través de un número de puerto dedicado, e incluso puede ejecutar o instalar un paquete. A día de hoy, ofrecemos 5 SBC ODROID con una configuración básica:

- 2x ODROID-XU4
- 2x ODROID-C2
- 1x Kit Home Cloud ODROID-HC2 con un HDD 3.5” de 1TB

¿Cómo son accesibles?

Solo 4 de 5 SBC ODROID están listos para aceptar comandos a través de SSH con un número de puerto dedicado.

```
Login account: odroid
Password: odroid
```

Una vez que accedas a un SBC ODROID, tienes completa autorización para ejecutar cualquier comando de Linux para hacer pruebas (algunos comandos podrían estar restringidos por razones de seguridad), y también puedes saltar a otro SBC ODROID de la misma red local.

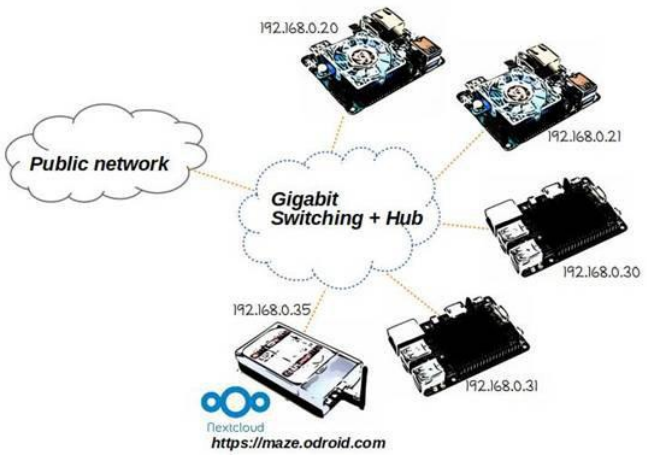


Figura 2 – Configuración de la red

Unit	ODROID Platform	External Port #	Internal Port #	Internal IP address	Note
#1	ODROID-XU4	2220	2222	192.168.0.20	8GB uSD + 8GB eMMC
#2	ODROID-XU4	2221	2222	192.168.0.21	8GB uSD + 8GB eMMC
#3	ODROID-C2	2230	2222	192.168.0.30	8GB uSD + 8GB eMMC
#4	ODROID-C2	2231	2222	192.168.0.31	8GB uSD + 8GB eMMC
#5	ODROID-XU4	432	NONE	192.168.0.35	8GB uSD + 1TB HDD

Figura 3 – Esquema de ODROIDs conectados a maze.odroid.com

Por ejemplo, si deseas acceder al ODROID-XU4 en el que el número de puerto es 2220, puedes ejecutar el siguiente comando, que te permitirá acceder a un ODROID-XU4 con la dirección IP interna 192.168.0.20:

```
$ ssh -p 2220 odroid@maze.odroid.com
```

Una vez que tenga acceso a un ODROID, puedes conectarte a cualquier otro ODROID de la misma red con su dirección IP interna dedicada. Por ejemplo, este comando le permitirá acceder al ODROID-C2 desde ODROID-XU4 al que te conectaste con el comando anterior:

```
$ ssh -p 2222 odroid@192.168.0.30
```

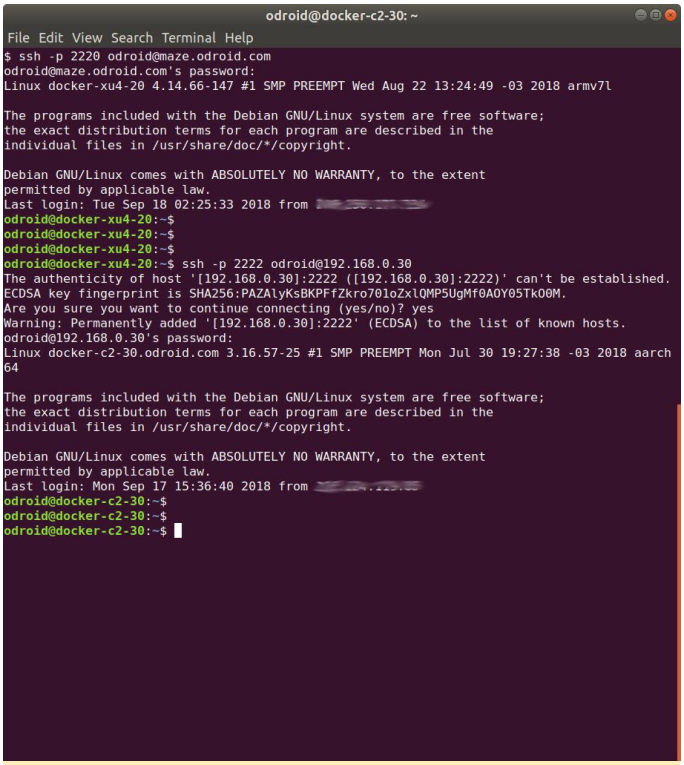


Figura 4 – Conectando a los ODROID a través de SSH

¿Qué otro hardware hay disponible?

A parte de los 4 SBC ODROID, también ofrecemos un almacenamiento en la nube dedicado que ejecuta Nextcloud con un disco duro de 1 TB. Este dispositivo de almacenamiento evidencia que puedes montar tu propio dispositivo en la nube con un ODROID-XU4 y te ofrece la oportunidad de usarlo antes de que decidas montarlo por ti mismo. Todo el mundo puede acceder a este almacenamiento con una cuenta abierta, Así que debes saber que cualquiera puede acceder a todos los archivos. Por lo tanto, no deberías cargar ningún archivo privado en el almacenamiento. Además, no debe compartir ningún tipo de archivo no

gratuito para tu uso personal a través de este almacenamiento.

```
Login account: odroid
Password: odroidfun
```

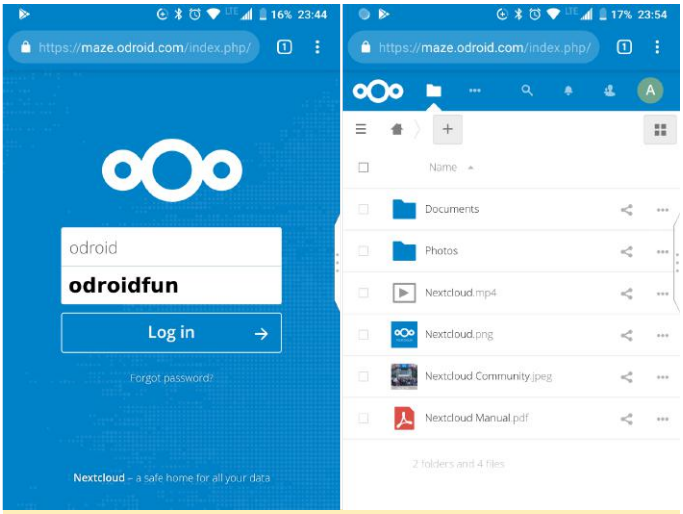


Figura 5 – Conexión a almacenamiento en la nube desde un dispositivo móvil

¿Qué es posible y que no?

Todos los SBCs ODROID en maze.odroid.com son de acceso público y están abiertos con el fin de ofrecer experiencias de muestra con SBC. Nos alegraría que lo utilizaras y nos gustaría escuchar tu opinión sobre que podríamos mejorar. Puede ver el rendimiento del hardware con herramientas de rendimiento como sysbench o incluso con simples herramientas de Linux como dd o ping. Además, si está dispuesto a ejecutar una herramienta de red con un determinado puerto, puede usar los puertos entre el 4000 y el 4499 en el contenedor Docker.

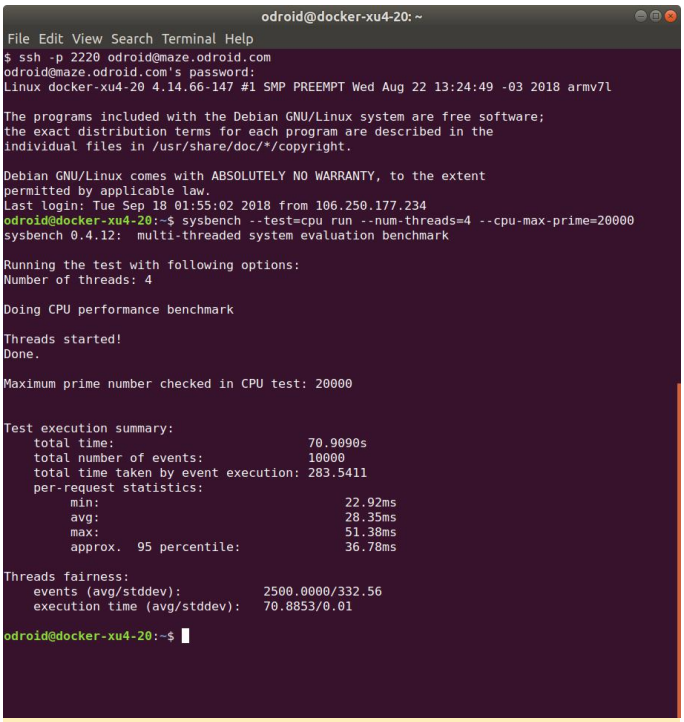


Figura 6 – Ejecutando Sysben en ODROID-XU4

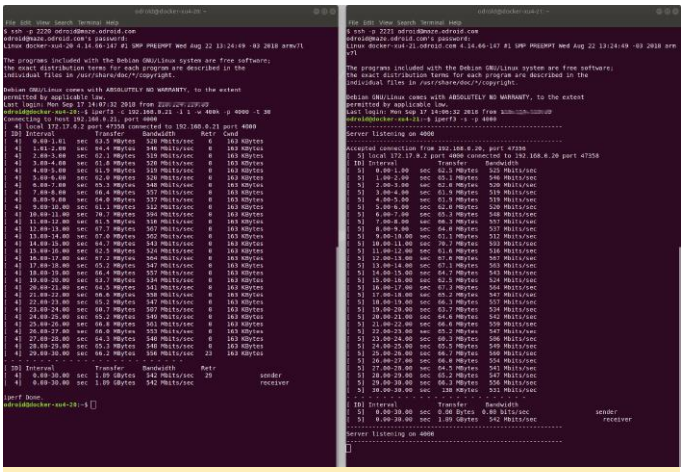


Figura 7 – Ejecutando iperf3 para medir el ancho de banda de la red entre dos SBCs ODROID

El SBC ODROID al que estás accediendo se está ejecutando en un contenedor Docker, de modo que también puedes ver cómo de bien se comporta el contenedor Docker como sistema operativo nativo sobre el hardware ARM.

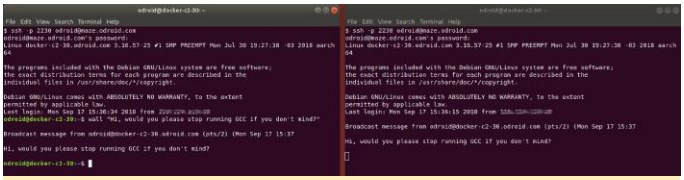


Figura 8: di "Hola" si encuentras a alguien más que esté haciendo un trabajo interesante

Desafortunadamente, no ofrecemos acceso de entorno gráfico debido al potencial limitado del hardware a la hora de permitir muchos usuarios al

mismo tiempo. Tampoco queremos que sean un recurso para la piratería o que se utilice como máquina de compilación que consuma todos los recursos del hardware. Además, el ODROID al que te conectas debería funcionar lentamente, ya que se puede acceder a los dispositivos de maze.odroid.com en cualquier momento.

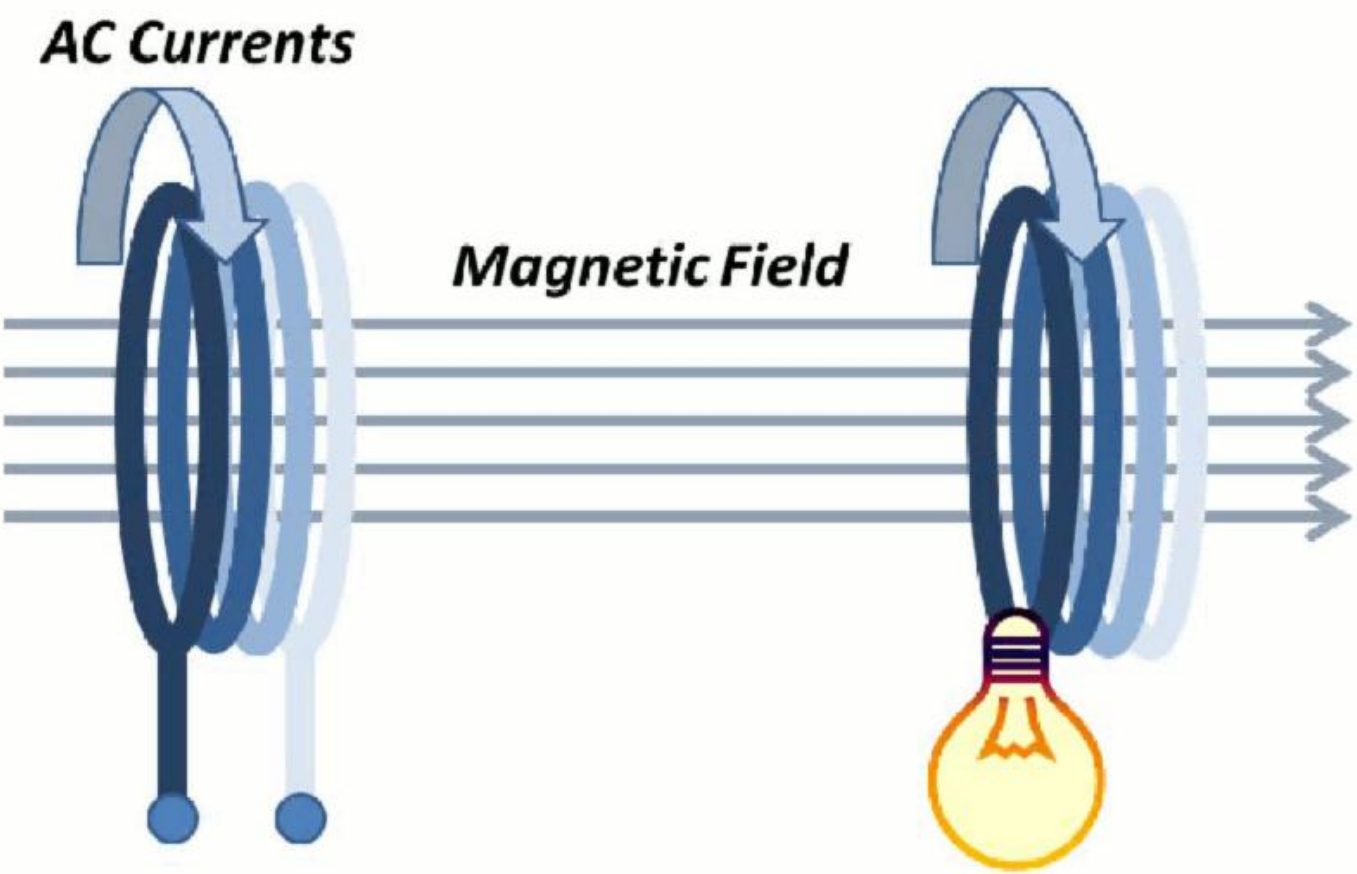
¿Durante cuánto tiempo funcionará maze.odroid.com?

Esperamos ofrecer buenas experiencias con los SBCs ODROID que hemos montado, pero también ofrecemos dispositivos populares o futuros poco después de su introducción en el mercado. Hoy día solo tenemos 5 SBC ODROID, pero se pueden incorporar más dispositivos a demanda o cada vez que se publique un nuevo SBC ODROID. Siempre y cuando tengamos usuarios que quieran tener una experiencia, ejecutaremos más SBC ODROID en maze.odroid.com.

Nos gustaría escuchar tus sugerencias o peticiones, visita el hilo del foro ODROID en **https://forum.odroid.com/viewtopic.php?f=29&t=32257#p234012**. Para preguntas, comentarios y sugerencias, visita el artículo original en **https://medium.com/@tobetter/odroid-bench-c5c1a10d6bec**.

Carga Inalámbrica: Añadiendo un Sistema de Carga Inalámbrica Qi al ODROID-GO

October 1, 2018 By @Kamots ODROID-GO



Soy un entusiasta de la electrónica desde que tenía ocho años, cuando mi abuelo solía enviarme pequeños componentes como LEDs, pequeños motores DC y temporizadores 555. Incluían esquemas dibujados a mano con notas que podía seguir para crear simples proyectos. A medida que iba creciendo, los proyectos se volvían más complejos, permitiéndome seguir aprendiendo. También me adentré en el mundo de los ordenadores, la programación y la radio amateur antes de ir a la universidad para conectarme a redes informáticas.

Avance rápido del 2018: ahora hago videos de YouTube sobre tecnología. Empecé mi canal "Kamots Talks Tech" porque compré un ODROID-GO y me impliqué con la comunidad, pero no estaba satisfecho con la veracidad de la mayoría de las críticas publicadas sobre el GO. Quería demostrar que este dispositivo era mejor de lo que la gente decía, así que hice mi primer video sobre GO: un video con un mini-análisis sobre ODROID-GO. Después de muchos más videos, he decidido escribir un artículo para que sea publicado, algo que nunca había hecho antes.

El objetivo de mi proyecto es añadir la posibilidad de cargar de forma inalámbrica el ODROID-GO. La motivación me vino por el hecho de utilizar la plataforma Qi Wireless Charging que ya tengo para hacer algo increíble con lo que disfrutasen los suscriptores de YouTube. Me inspiré en un video de ETA PRIME en el cual se añadía la carga inalámbrica al NeoGeo Mini. Puede parecer un proyecto complicado, pero en realidad, tan sólo supone realizar cuatro conexiones de soldadura. Si dedicas algo de tu tiempo a seguir las instrucciones que aparecen a continuación, probablemente tendrás éxito. Sin embargo, existe el riesgo de dañar tu GO, así que no

lleves a cabo este proyecto a menos que confíes en tus habilidades de soldadura y electrónica.

Componentes necesarios

- ODROID-GO (https://bit.ly/2lgWvGA)
- Módulo receptor inalámbrico Universal Qi de Adafruit (https://www.adafruit.com/product/1901)
- Cable muy delgado (28AWG/0.081mm2). Desmonté un cable Ethernet plano para conseguir el mío.
- Cinta aislante u otro adhesivo fuerte no conductor
- Soldador y soldadura

Preparación

Primero, veamos partes del esquema del ODROID-GO para entender dónde conectar la salida de la bobina Qi. La energía para el GO es suministrada normalmente a través del puerto USB, pero las yemas de soldadura son muy pequeñas para el conector Micro USB, de modo que necesitamos encontrar otra forma de realizar la conexión por dentro del GO.

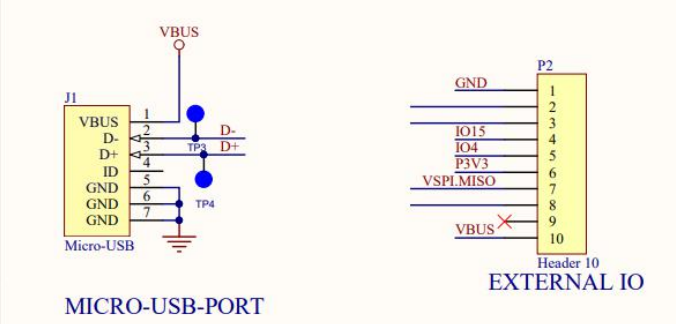


Figura 1 – Extractos del esquema de ODROID-GO

Como puede ver en la Figura 1, normalmente se suministra energía al GO a través del conector Micro USB hacia el VBUS y GND. Esos mismos rieles de alimentación están disponibles en el cabezal IO externo, que tiene pines mucho más grandes en la placa del GO. Así que soldaremos el módulo Qi al pin 1 y 10 del cabezal IO externo, que puedes ver en la Figura 2. La alimentación USB estándar es de 5 voltios. El módulo Qi también proporciona 5 voltios, de modo que alimentará el GO como si lo conectaras a través de USB.

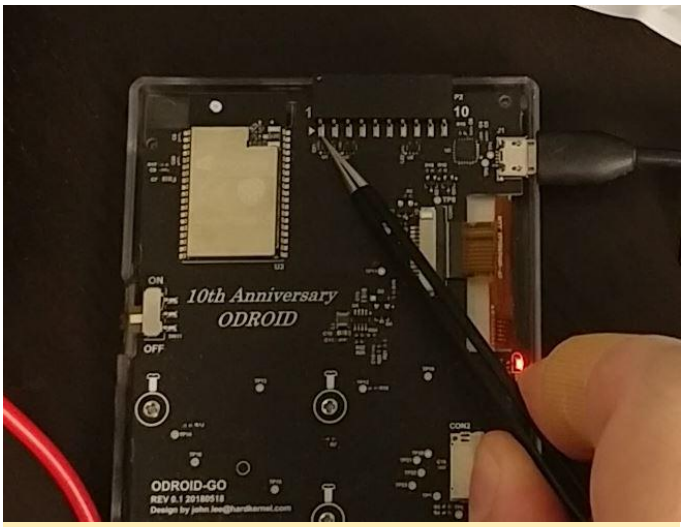


Figura 2 – Cabezal IO externo

Ahora que hemos descubierto dónde conectar la bobina, deberás doblar con mucho cuidado los dos cables que proceden de la bobina Qi hacía tu placa controladora. Esto nos permite eliminar los pliegues de plástico en el interior de la cubierta posterior de GO donde finalmente montaremos el módulo Qi. Yo utilicé unas pinzas de depilar para doblar con cuidado los cables de alimentación de la bobina, como puede ver en la Figura 3. Aunque yo lo hice después de haber soldado los cables, te recomiendo que lo haga antes de soldar.

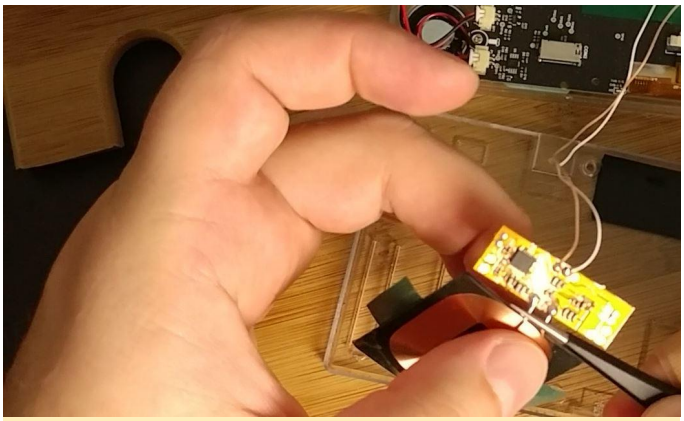


Figura 3 – Doblando los cables de alimentación de la bobina Qi

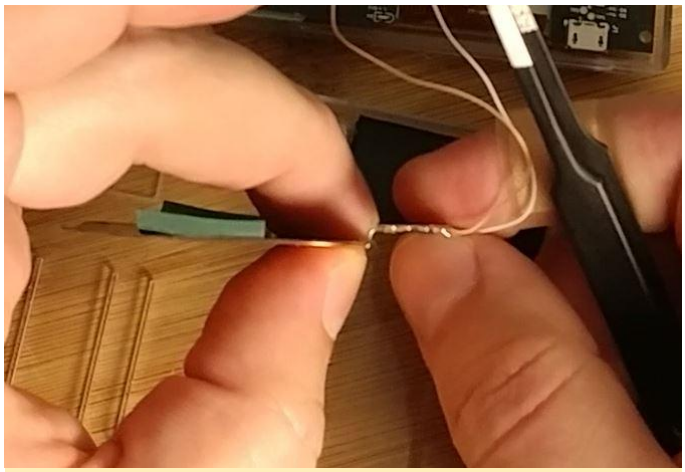


Figura 4 – Cables de bobina Qi doblados

Haciendo las conexiones

Ahora comienza el delicado trabajo de soldadura. Necesitaras usar la soldadura de diámetro más delgado que tengas. Añade cuidadosamente una pequeña cantidad de soldadura a las yemas V4+ y GND en la placa controladora de la bobina Qi tal y como se ve en la Figura 5. Hay tres yemas y si tiene la bobina orientada de la misma forma que en la Figura 5, deberás añadir soldadura a la yema central y derecha.

Figura 5 – Soldadura sobre las yemas V4+ y GND

Una vez que hayas añadido la soldadura a las yemas, suelda un cable a cada yema remanando la soldadura que has añadido a las yemas y presionando el cable sobre ellas. Puedes verme haciendo esto en la figura 6.

Figura 6 – Añadiendo los cables a la bobina Qi

A continuación, comprueba la longitud de los cables para asegurarte de que puedan alcanzar los pines del cabezal IO externo sin que sean tan largos que resulten difícil de manejar cuando vuelvas a ensamblar el GO. Recomiendo cortar donde el cable alcanza mis dedos en la Figura 7.

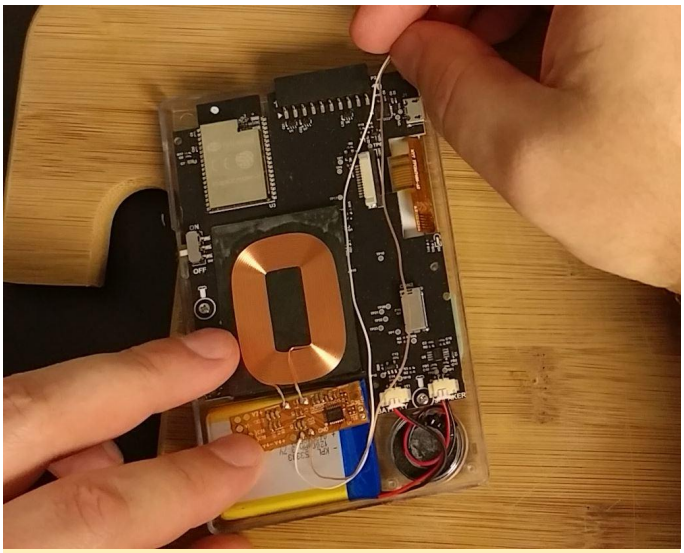


Figura 7 – Comprobando la longitud del cable

Para el siguiente paso, es probable que necesites un soporte tipo “Mano Amiga” con pinzas de contacto, como se muestra en la Figura 8, o un juego adicional de manos humanas. Necesitarás colocar los cables sobre los pines del cabezal, calentar el pin del cabezal y el cable, luego añadir soldadura para hacer una buena conexión. Tal y como recordarás del esquema de la Figura 1, el cable de V4 + va al pin 10 y el cable de GND al pin 1. Ten mucho cuidado al soldar el cable al pin 1 ya que el componente que hay justo al lado podría dañarse.

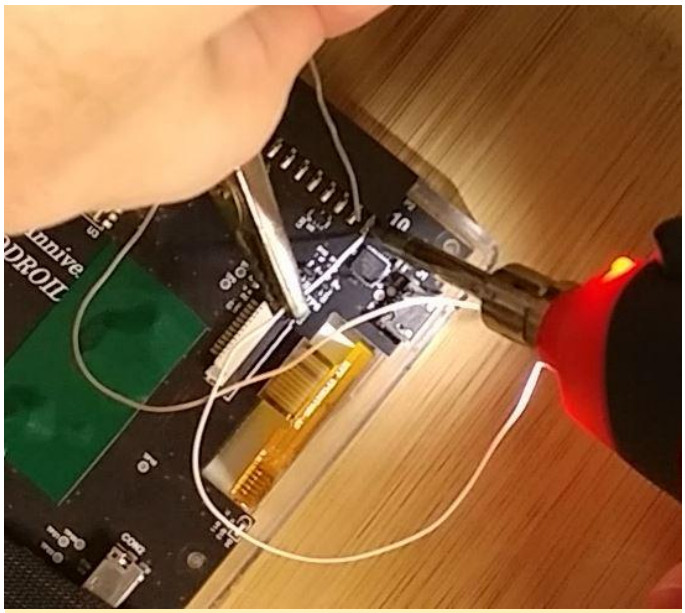


Figura 8 – Soldando los pines del cabezal

Comprueba tus conexiones con un multímetro, asegurándote de que dispones de buenas conexiones sin resistencia. Verifica que has conectado los pines de salida de la bobina Qi a los pines correctos en el GO. Hacer las cosas al revés podría dañar tu dispositivo y/o la placa controladora de la bobina Qi.

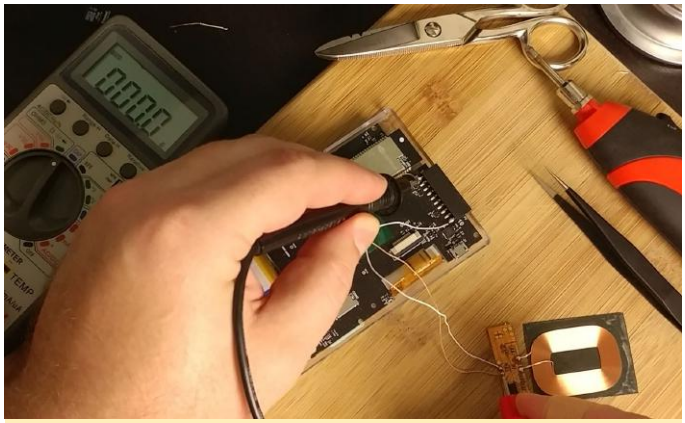


Figura 9 – Probando tus conexiones

Terminando

Una vez que estés seguro de que todo está conectado correctamente, coge tu plataforma de carga Qi y coloca la bobina Qi sobre ella tal y como se ve en la Figura 10. La luz roja en el GO debería iluminarse, lo cual indica que se está cargando.

Figura 10 – Comprobando que carga

Si funciona, ¡celébralo! La parte difícil ya ha pasado. Ahora solo necesitas asegurar la bobina Qi en la carcasa posterior del GO. Utiliza cinta aislante como se puede ver en las fotos. No obstante, Justin Thomas dejó un comentario en mi video sobre este proyecto que decía que debería probar la cinta de doble cara VHB de la marca 3M. Dejo a tu criterio, la que quieras usar. Lo más importante es asegurarse de que la bobina quede en sentido horizontal dentro de la cubierta posterior y que no se dañe. También es posible que quieras cortar parte de los pliegues de plástico que hay dentro de la cubierta posterior alrededor de donde se encuentra la placa del controlador de la bobina Qi antes de asegurar la bobina. Puedes ver cómo aseguré la mía en la Figura 11.

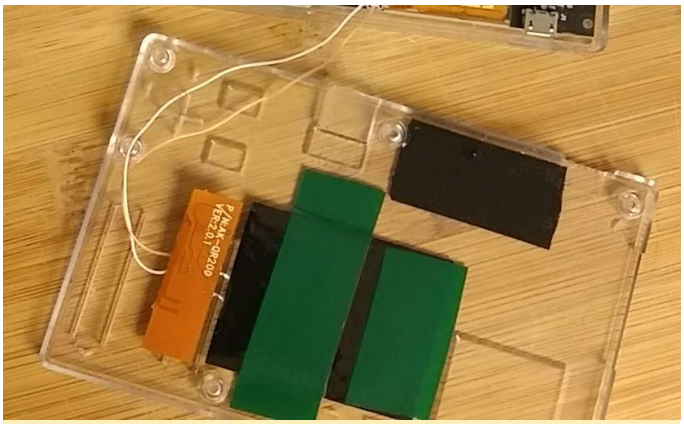


Figura 11 – Bobina montada en el panel posterior

Una vez que hayas asegurado la bobina Qi y tu placa al panel posterior, cierra el GO (asegurándote de que los cables no queden atrapados en ninguna parte) y vuelve a colocar todos los tornillos en su lugar. El resultado final debería ser similar a la Figura 12.

Figura 12 – ODROID-GO ensamblado con la bobina Qi

Ahora puede colocar tu GO en una plataforma de carga Qi, asegurándote de centrar la bobina sobre la plataforma y disfrutar de la carga inalámbrica del GO. Ten en cuenta que la bobina se calienta un poco tras un tiempo, pero no creo que sea un problema. Sin embargo, la bobina no puede saber cuándo el GO termina de cargarse, así que no lo dejes en una plataforma de carga durante toda la noche. También puede utilizar un cable USB para cargar tu GO si quieres.

Campamento de Programación – Parte 5: Leer el Voltaje de la Batería Integrada en el ODROID-GO

October 1, 2018 By Justin Lee ODROID-GO, Tutoriales

En esta guía vamos a aprender cómo obtener el estado de la batería con Arduino. La pantalla LCD mostrará el voltaje restante de la batería en voltios.

Figura 1 – El ODROID-GO tiene un módulo de batería de unos 3.7V

Podemos leer el nivel de la batería a través de uno de los ADCs SAR de 12 bits que están integrados en ESP32. Estos ADCs son:

- ADC1: 8 canales, conectado a GPIOs 32-39.
- ADC2: 10 canales, conectado a GPIOs 0, 2, 4, 12-15, 25-27.

Existen algunas limitaciones a la hora de usar ADC2 en una aplicación. La batería está conectada al número pin GPIO 36, de modo que deberíamos leer un valor al usar ADC1. Existe una librería para controlar ADC para ESP32 que se llama adc.h. En esta guía, la vamos a usar para leer el nivel actual de la batería en voltios y mostrarlo en la pantalla LCD. En primer lugar, prepara el código tal como se muestra a continuación para mostrarlo en la pantalla LCD:

```
#include

void setup() {
// put your setup code here, to run once:
GO.begin();
GO.lcd.setTextSize(2);
}

double readBatteryVoltage() {

}

void showBatteryVoltage(double voltage) {
GO.lcd.clear();
GO.lcd.setCursor(0, 0);

GO.lcd.printf("Current Voltage: %1.3lf V
", voltage);
}

void loop() {
// put your main code here, to run repeatedly:
showBatteryVoltage(readBatteryVoltage());
delay(1000);
}
```

Definimos dos funciones por adelantado:

- readBatteryVoltage(): devuelve el voltaje completamente calculado.
- showBatteryVoltage(): recibe este voltaje y lo muestra en pantalla.

Ajusta el canal con los correspondientes valores tal y como lo hemos diseñado. Antes de configurarlo, es importante saber que el voltaje de la batería GPIO está dividido por 2 debido a la limitación de entrada de datos del ADC integrado. Por lo tanto, si el valor original que procede de la batería es de 3.7V, entonces el valor de entrada al pin GPIO será de aproximadamente 1.85V. Así pues, tenemos que multiplicar el valor por 2 para conocer el voltaje real.

Utilizamos el ADC SAR de 12 bits para el canal con una atenuación de 11 dB. Deberíamos usar estos rangos para calcular igualmente el resultado. En primer lugar, define este valor extra como una macro Preprocesador e incluye las librerías necesarias para usar ADC en ESP32:

- driver/adc.h: para obtener el valor bruto ADC sobre el voltaje.
- esp_adc_cal.h: para calcular el voltaje correcto utilizando el AP.

A continuación, configura el canal con el valor adecuado usando dos funciones:

- adc1_config_width(): configura el ancho del ADC.
- adc1_config_channel_atten(): configura la atenuación del canal. El número de pin GPIO 36 utiliza el canal número 1.

Para conseguir un voltaje ADC preciso, es necesario recurrir a la calibración. El voltaje de referencia del ADC es originalmente 1.1V por defecto, pero en realidad difiere un poco en cada módulo ESP32. El fabricante escribe los datos de calibración en efuse:

- esp_adc_cal_characterize(): devuelve una característica de su AP como una estructura.

La función readBatteryVoltage () lee un valor ADC con la función adc1_get_raw (). Devuelve un voltaje calculado como un valor de tipo doble utilizando la función esp_adc_cal_raw_to_voltage ():

```
#include
#include <driver/adc.h>
#include

#define RESISTANCE_NUM 2
#define DEFAULT_VREF 1100

static esp_adc_cal_characteristics_t
adc_chars;

void setup() {
// put your setup code here, to run once:
GO.begin();

GO.lcd.setTextSize(2);

adc1_config_width(ADC_WIDTH_BIT_12);
adc1_config_channel_atten(ADC1_CHANNEL_0,
ADC_ATTEN_DB_11);
esp_adc_cal_characterize(ADC_UNIT_1,
ADC_ATTEN_DB_11, ADC_WIDTH_BIT_12,
DEFAULT_VREF, &adc_chars);
}

double readBatteryVoltage() {
return (double)
esp_adc_cal_raw_to_voltage(adc1_get_raw(ADC1_C
HANNEL_0), &adc_chars) * RESISTANCE_NUM /
1000;
}

void showBatteryVoltage(double voltage) {
GO.lcd.clear();
GO.lcd.setCursor(0, 0);

GO.lcd.printf("Current Voltage: %1.3lf V
", voltage);
}

void loop() {
// put your main code here, to run repeatedly:
showBatteryVoltage(readBatteryVoltage());
delay(1000);
}
Opcionalmente, podemos realizar el cálculo con
una mayor precisión mediante un muestreo
múltiple del valor ADC
Lee el valor 64 veces y lo divide por el total
de repeticiones.
#include
#include <driver/adc.h>
#include

#define RESISTANCE_NUM 2
#define DEFAULT_VREF 1100
#define NO_OF_SAMPLES 64

static esp_adc_cal_characteristics_t
adc_chars;

void setup() {
// put your setup code here, to run once:
GO.begin();

GO.lcd.setTextSize(2);

adc1_config_width(ADC_WIDTH_BIT_12);
adc1_config_channel_atten(ADC1_CHANNEL_0,
ADC_ATTEN_DB_11);
esp_adc_cal_characterize(ADC_UNIT_1,
ADC_ATTEN_DB_11, ADC_WIDTH_BIT_12,
DEFAULT_VREF, &adc_chars);
}

double readBatteryVoltage() {
uint32_t adc_reading = 0;
for (int i = 0; i < NO_OF_SAMPLES; i++) {
adc_reading += adc1_get_raw((adc1_channel_t)
ADC1_CHANNEL_0);
}
```

```
adc_reading /= NO_OF_SAMPLES;

return (double)
esp_adc_cal_raw_to_voltage(adc_reading,
&adc_chars) * RESISTANCE_NUM / 1000;
}

void showBatteryVoltage(double voltage) {
GO.lcd.clear();
GO.lcd.setCursor(0, 0);

GO.lcd.printf("Current Voltage: %1.3lf V
", voltage);
}

void loop() {
// put your main code here, to run repeatedly:
showBatteryVoltage(readBatteryVoltage());
delay(1000);
}
```

Presiona CTRL-U para compilar y cargar el esquema. El voltaje actual de la batería aparecerá en la pantalla LCD.

Ejemplo completo

El ejemplo completo lo tienes disponible haciendo clic en el menú Files → Examples → ODROID-GO → Battery para importar y presiona CTRL-U para compilar/cargar, tal y como se muestra en la Figura 2.

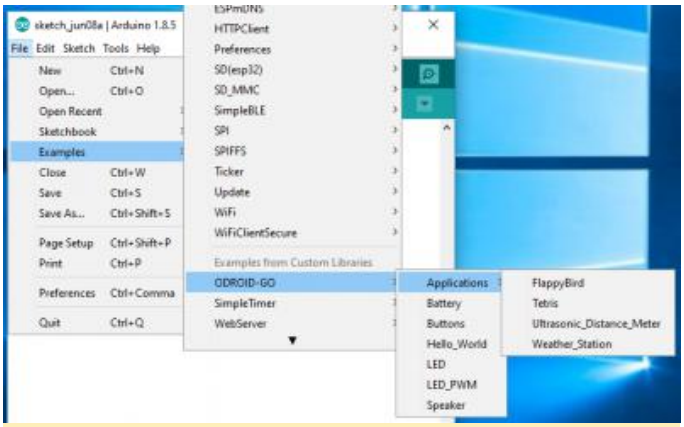


Figura 2 – Accediendo al ejemplo completo

Para comentarios, preguntas y sugerencias, visita el artículo original en **https://wiki.odroid.com/odroid_go/arduino/05_battery**.

Presentado NEMS Linux: Parte 1 – el Servidor de Monitorización Profesional Nagios para Dispositivos ODROID

October 1, 2018 By Robbie Ferguson Linux, Tutoriales

Nagios® Core™–al que me referiré simplemente como "Nagios" a lo largo de este artículo, es una aplicación de servidor de código abierto gratuita que permite monitorizar los hosts y servicios que especifiques, que te avisa cuando las cosas van mal y cuando mejoran. He estado usando Nagios durante muchos años. Si tuviera que aventurarme a poner una fecha de cuándo lo empecé a usar, diría que fue alrededor de 2006.

Mi esposa y yo administrábamos una pequeña empresa de servicios informáticos en nuestra casa por aquel entonces, y para hacer un seguimiento de los sitios de mis clientes y ser tan proactivo como podría ser, tenía un monstruoso ordenador en el garaje vigilando El espacio del disco duro de mis clientes, el estado de las copias de seguridad, la carga de la CPU y las actualizaciones del antivirus, así como otros servicios.

Recuerdo haber configurado ese antiguo servidor Nagios. El proceso fue un poco pesado y toda la configuración se realizó a través del terminal de Linux abriendo los archivos de configuración en vi. Una sintaxis malformada y Nagios no se iniciaría. Lo hice funcionar, y si alguien alguna vez ha dudado de mi sabiduría, estaba totalmente equivocado. ¡Ah!, ¿a quién estoy engañando? Nadie ha dudado nunca de mi sabiduría. A medida que pasaron los años y mi base de clientes de soporte profesional continuaba creciendo, empecé a reutilizar hardware antiguo, instalando un servidor Nagios independiente en cada sitio del cliente. Esto funcionaba bastante bien.

Recibí mi primera Raspberry Pi en 2014. Después de tenerla en un estante durante un año, empecé a considerar posibles formas de utilizarla. Me di cuenta de que el consumo de energía, el espacio en rack y el ruido de estos antiguos servidores Nagios suponían

una increíble pérdida de recursos. Estaba convencido de que un ordenador de placa reducida podía ser un excelente servidor Nagios, y empecé a hacer pequeñas configuraciones.

¿Por qué reinventar la rueda? La imagen NagiosPi de Ryan Siegel estaba lista para usarse y ganaba en popularidad. Empecé a usarla, pero rápidamente me sentí desanimado por el estado de la distribución, que parecía estar absoleta y no se estaba actualizando a un ritmo adecuado para usarla a un nivel profesional. Era un excelente punto de partida, pero en algunos aspectos importantes sentía que era un producto incompleto. Empecé a trabajar en mi propia recompilación de NagiosPi, llamándola NEMS; abreviatura de "Servidor de Monitorización Profesional Nagios".

Figura 1: Este servidor Nagios cliente fue reemplazado por el primer servidor NEMS en 2016

Soy programador en mi vida profesional. Desarrollo aplicaciones de servidor, principalmente para web. Además de desarrollar una base de software más reciente, lo primero que me propuse fue crear una IU basada en navegador para NEMS, que reuniese todos los componentes de NagiosPi en una única interfaz. Esto más tarde se convertiría en el panel de control de NEMS, también conocido por su nombre de repositorio GitHub, "nems-www".

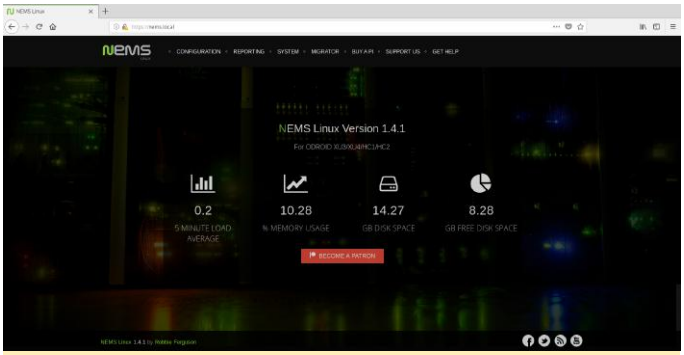


Figura 2 – Panel de control de NEMS

Después de poner NEMS a disposición de la gente a través de mi blog, el propio Siegel dijo: "Me encantaría actualizar NagiosPi, pero no tengo la habilidad para crear una GUI que pueda superar la de NEMS. Creo firmemente que siempre ha sido necesaria realizar una reforma a NagiosPi y NEMS ha supuesto lo que es esencialmente una versión actualizada y mejorada de NagiosPi. No hay razón para no empezar a usar NEMS por lo pronto. ¡Buen trabajo, Robbie!" No me detuve aquí y con su maravilloso espíritu de comunidad, Siegel incluso llego a colaborar durante el desarrollo de NEMS 1.2 a principios de 2017, realizando diversas aportaciones de Windows a NEMS.

Con un nuevo lanzamiento importante de NEMS cada seis meses y actualizaciones continuas, NEMS está basado actualmente en Debian Stretch con Nagios Core 4.3.4 en su corazón. Habiendo actualizado y mantenido NEMS con una base de software actual, también incluye cosas como PHP 7, certificados SSL actualizados y un conjunto de software personalizado optimizado para trabajar en un servidor moderno. Por ejemplo, NConf (una herramienta muy útil para configurar Nagios) detuvo su desarrollo hace años, así que solo funciona en PHP 5.3 o inferior. En consecuencia, decidí modificar el código y crear un versión aparte con el objetivo que fuera compatible con PHP 7.0+. Por supuesto, hice algunas otras mejoras en el camino.

NEMS Linux, como se llama actualmente (tenía que buscar una .com, después de todo) recurre a una monitorización de recursos de red más moderna y elimina el antiguo requisito de porgramación de Nagios. Los scripts aún están ahí, es solo que tú (el usuario) nunca tendrás que verlos o tocarlos. Todo se controla, configura y monitoriza a través de tu

navegador web, con notificaciones por correo electrónico, Telegram o Pushover, todas operativas listas para usarse. También cuenta con una API JSON, un sistema de visualización por TV para tu sala de servidores y mucho más.

NEMS Linux ha evolucionado para ser lo que pienso que es la mejor experiencia de Nagios disponible. Como usuario de Nagios, este es el servidor de Nagios que siempre he anhelado. A medida que NEMS ha seguido creciendo, empecé a buscar una plataforma más potente que la Raspberry Pi. Ahí fue cuando encontré el ODROID-XU4. Hace poco más de un año (el 13 de septiembre de 2017, para ser exactos) inicie la tarea de exportar NEMS Linux al ODROID-XU4. Después de casi un año de desarrollo, estoy extremadamente orgulloso y emocionado por compartir: NEMS Linux ahora está disponible para las placas ODROID.

Características de NEMS

Ya hemos mencionado la interfaz obvia y las mejoras de UX que NEMS Linux aporta a la experiencia de Nagios. Estos son quizás los puntos clave que hacen que NEMS destaque, pero es importante entender que NEMS Linux es mucho más que sólo Debian con Nagios instalado. Veamos una pequeña selección de características adicionales.

NEMS Migrator

Al centrarme en el desarrollo de una distro para ordenadores de placa reducida (SBC), me tomé muy en serio el hecho de que las tarjetas SD pueden y probablemente fallen, y los datos pueden perderse. Quería crear una forma para que los usuarios pudieran realizar copias de seguridad y restaurar fácilmente su configuración. De ese deseo nació Migrator.

Migrator te permite hacer una copia de seguridad de toda tu configuración NEMS (hosts, servicios, verificaciones, configuración del sistema, etc.) a través de un recurso compartido samba, una descarga https o incluso un servicio de copia de seguridad externo opcional. Las copias de seguridad se pueden cifrar y solo tú conoces la clave de descifrado. En el caso de que tu dispositivo falle, puede escribir la imagen en una nueva tarjeta SD, restaurar tu copia de seguridad hecha por Migrator y empezar a funcionar en menos de cinco minutos con toda tu configuración intacta. Migrator también facilita la transición de una plataforma a otra. Por ejemplo, teniendo configurado un servidor NEMS Linux en una Raspberry Pi 3, puedes cambiar fácilmente a un ODROID-XU4 con el fin de aumentar el rendimiento de forma considerable.

Otra ventaja que pone Migrator sobre la mesa es una ruta de actualización de lo más simple que pueda existir: a medida que se presenten las nuevas versiones principales de NEMS Linux, puedes escribir fácilmente la nueva imagen de NEMS, importar tu copia de seguridad y disponer de la última versión de NEMS en solo unos minutos.

Configuración basada en interfaz de usuario con NEMS Configurator

NEMS Configurator (NConf) es lo que hace posible que la configuración de Nagios esté basada en navegador. Esta versión personalizada de la antigua herramienta de configuración NConf aporta un sofisticado front-end a la moderna arquitectura de NEMS. Toda la configuración de Nagios se realiza a través de esta interfaz: desde añadir hosts hasta configurar tus Chequeos de servicio. Todo se hace a través de un sistema muy intuitivo basado en el navegador.

Ahora bien, debo admitir que NConf estéticamente no es la característica con mejor aspecto de NEMS en este momento, pero su funcionamiento es brillante. Está previsto un rediseño de la interfaz de usuario para un futuro lanzamiento. Cuando eso suceda, la interfaz se actualizará automáticamente en todas las implementaciones existentes a través del sistema de actualización automática de NEMS. Con NEMS NConf, ¡nunca más tendrás que examinar un archivo cfg de Nagios!

NEMS System Settings Tool (SST)

Hablando de suprimir los archivos de configuración de Nagios, son varias las opciones de configuración de Nagios que han sido trasladas a la herramienta conocida NEMS System Settings Tool, también

llamada NEMS SST. Elementos como la configuración del servidor SMTP, las credenciales de usuario de dominio y otros valores por defecto forman parte de esta interfaz.

Así que ahora que ya conoces un poco NEMS y cómo surgió, ¡Vamos a zambullirnos en él!

Instalación

Todo lo que necesitas para echar a andar NEMS Linux es un dispositivo ODROID compatible y una tarjeta Micro SD. En este momento, el XU3, XU4, HC1 y HC2 son compatibles. Admitirá más dispositivos tan pronto como disponga del hardware de desarrollo para compilar y probar, así que, si estás leyendo este artículo varios meses después de su publicación, consulta el sitio web de NEMS, ya que es posible que tu placa ya sea compatible.

Descarga la última versión de NEMS Linux desde **https://nemslinux.com** y "grábala" con tu herramienta favorita. Yo suelo utilizar Etcher de https://etcher.io/ pero puedes usar la herramienta que más te guste. eMMC puede funcionar si tienes el gestor de arranque actualizado, pero aún no está oficialmente soportado. En caso de duda, utiliza una tarjeta SD UHS-I o superior. Recomiendo 16 GB o más, pero podría funcionar con una de 8 GB si es lo único que tiene a mano. Siempre puedes usar NEMS Migrator para cambiar a una tarjeta más grande en el futuro.

Al arrancar tu servidor NEMS Linux, el sistema de archivos se redimensionará automáticamente a la capacidad de tu tarjeta SD. Puedes confirmar que NEMS está funcionando introduciendo https://nems.local/ en tu navegador web. Si la resolución de nombres no funciona, prueba con la dirección IP de tu dispositivo NEMS, la cual puedes encontrar en la tabla DHCP de tu router, o en un TV conectado al puerto HDMI de tu dispositivo. Tengo pensado introducir soporte para la pantalla Cloudshell 2 en una futura actualización.

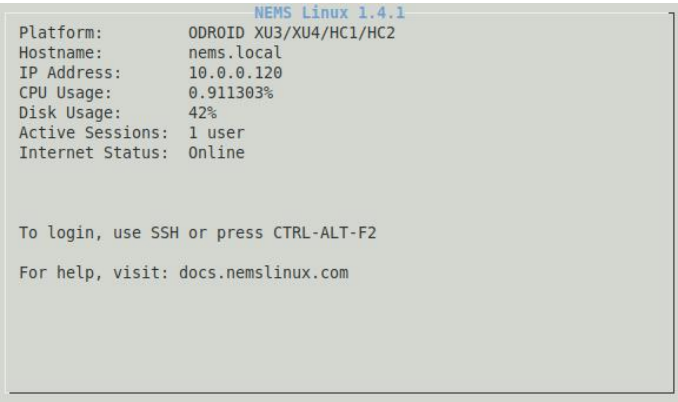


Figura 3: Detalles de NEMS que muestra un TV conectado

Initialization

En términos generales, la única vez que realmente tendrá que tocar el terminal de Linux en un servidor NEMS es durante el proceso de inicialización. Esta tarea mágica configura de forma automática todo tu servidor en solo unos segundos. Genera certificados autofirmados para que cada usuario de NEMS Linux tenga un certificado único, te permita configurar tu zona horaria, crea tu usuario administrador de Nagios, tu cuenta de Linux, etc. Para inicializar tu servidor NEMS Linux, conéctate a tu servidor a través de SSH en el Puerto 22 predeterminado usando las siguientes credenciales

```
Username: nemsadmin
Password: nemsadmin
```

Una vez conectado, escribe:

```
$ sudo nems-init
```

Se te pedirá que introduzcas nuevamente la contraseña. Sigue las indicaciones. Todo lo complicado se vuelve simple

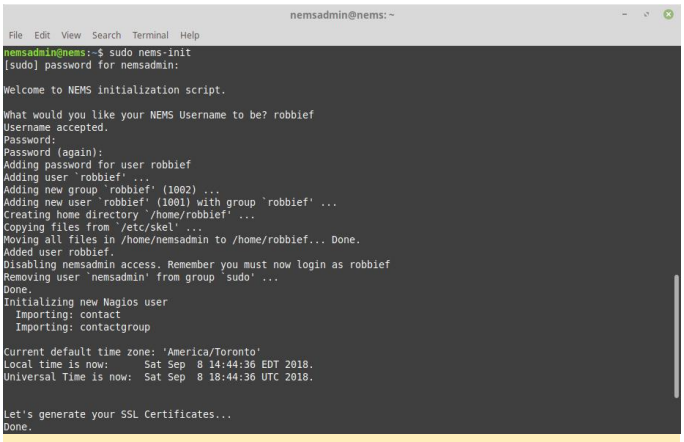


Figura 4 – Inicialización de NEMS

¡Felicidades! Tu servidor NEMS Linux ahora está online y listo para monitorizar tus recursos de red.

Conectarse a tu servidor NEMS

Ahora que tu servidor NEMS Linux está funcionando, puedes acceder a todo su conjunto de funciones a través de tu navegador web. Simplemente dirígete a https: //nems.local/ que debería funcionar en la mayoría de las redes, pero si lo necesitas, también puedes usar la dirección IP de tu servidor NEMS.

Configurar las notificaciones por correo electrónico

Lo primero que querrás hacer en tu nuevo servidor NEMS Linux es configurar tus parámetros SMTP. Esto permitirá que tu servidor NEMS te envíe un correo electrónico si se detecta algún problema.

Accede a NEMS System Settings Tool (SST) de NEMS desde el menú de Configuración de tu panel de NEMS. Esta herramienta elimina la necesidad de usar el archivo tradicional Nagios resource.cfg para configurar tus parámetros de correo electrónico. Una de las cosas buenas que tiene NEMS Linux es que en realidad no tengo que entrar en detalles sobre cómo hacer esto. Es tan intuitivo que no necesita explicación. Así que simplemente proporcionaré una captura de pantalla en la Figura 5.

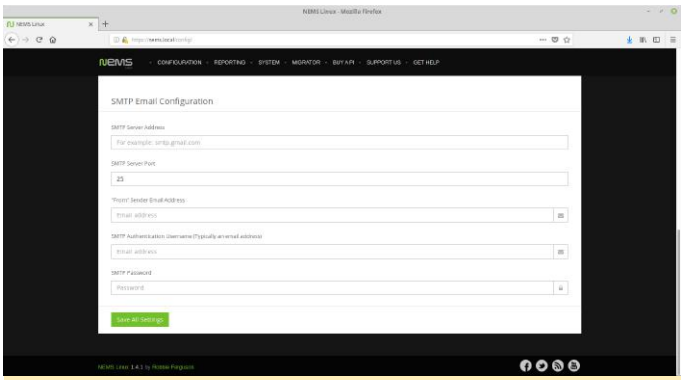


Figura 5: Configurar los parámetros del SMTP en Nagios es fácil con NEMS System Settings Tool

Configurar tu servidor SMTP en NEMS es tan simple como configurar un cliente de correo.

Consejo: si está utilizando Gmail como proveedor de SMTP, asegúrate de revisar la documentación de NEMS que se encuentra en https://docs.nemslinux.com/usage/nagios-gmail-smtp para llevar a cabo los pasos adicionales necesarios.

Una vez que estés convencido de haber ingresado correctamente la configuración de SMTP, procede a Guardar toda la configuración y luego vuelve a conectarte a través de SSH para probar la configuración de su correo electrónico con el siguiente comando (reemplaza youremail@yourdomain.com por la dirección de correo electrónico de un destinatario real):

```
$ sudo nems-mailtest youremail@yourdomain.com
```

Este comando simplemente verifica tu configuración del correo enviando una notificación de prueba. Si hay algún problema, aparecerá en pantalla y podrás realizar los cambios necesarios en la configuración de la cuenta en NEMS SST.

Nota: NEMS actualmente requiere que tu servidor SMTP admita la autenticación TLS. En contra de mi criterio, pero en línea con las peticiones de los usuarios, se añadirá una opción en una futura versión que permita la autenticación insegura si fuera necesario, como podría ser el caso de un relé interno.

El paso final de las notificaciones por correo electrónico

El último paso para configurar tus notificaciones por correo electrónico es decirle a NEMS dónde deseas que se envíen estos avisos. Para hacer esto, abre NEMS Configurator (NConf) en “Configuration”, y en el menú de navegación de la izquierda, presiona “Show“ junto a “Contacts”. Verás que el procedimiento de Inicialización de NEMS ya añadió tu usuario a NConf. En mi caso es robbief. Presiona el ícono del lápiz (Modificar) para editar tu contacto.

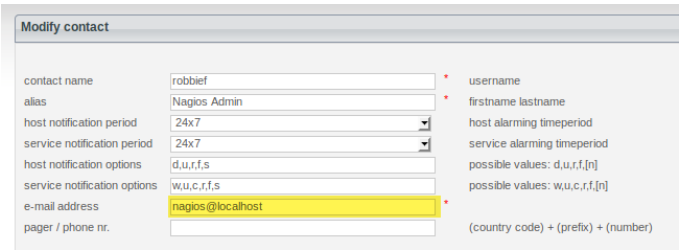


Figura 6 – Cambiando la dirección de correo electrónico del administrador

En Email Address, cambia el valor por tu dirección de correo electrónico real. Asegúrate de que la dirección

de correo electrónico del destinatario no sea la misma que la utilizada para la configuración de SMTP en NEMS SST. De lo contrario, es posible que no reciba las notificaciones, ya que el servidor receptor lo verá como el remitente y ocultará las notificaciones de tu bandeja de entrada.

Una vez que hayas cambiado la dirección de correo electrónico, desplázate hacia abajo y presione "Submit" para guardar los cambios. Ten en cuenta que tu información de contacto no estará disponible hasta que no se genere la configuración de Nagios.

Generar Nagios Config: Haz que tus cambios cobren vida

Para hacer que los cambios tengan efecto, presiona el enlace "Generate Nagios Config" en el panel de navegación de la izquierda. Deberías ver 0 errores. Si ve errores, presione "Syntax Check " y revisa dónde hay errores. NConf es muy bueno para mostrar dónde está el error, de modo que puedes volver atrás, repararlo e intentarlo de nuevo.

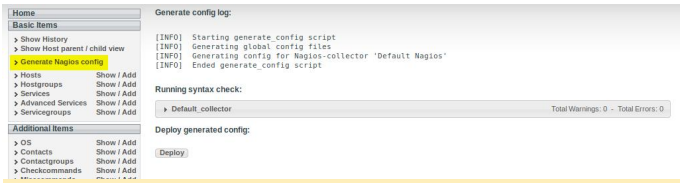


Figura 7: Generando la configuración de Nagios con NEMS Configurator

Si todo está comprobado, presiona "Deploy" y tu dirección de correo electrónico de contacto de administrador se activará al instante en Nagios.

Aprender más

Me gustaría animarte a que visites el Foro de Documentación y Comunidad de NEMS para aprender más sobre cómo configurar y usar tu nuevo servidor NEMS Linux. Asegúrate de acompañarme nuevamente en la edición del próximo mes de ODROID Magazine ya que llevaremos a cabo nuestro primer ejercicio: Configurar NEMS Linux para monitorizar un servidor local de Linux. NEMS tiene un foro de comunidad muy activo. Lo suelo visitar con bastante frecuencia para proporcionar soporte gratuito a los usuarios. También ofrezco soporte comercial personalizado para aquellos que necesitan un mayor nivel de soporte.

Me gustaría extender mi agradecimiento a mad_ady y meveric de la comunidad de ODROID por ayudarme en estos primeros días, ya que tenía pensado exportar NEMS Linux a la plataforma ODROID. Por supuesto, esta versión no sería posible sin el gran trabajo que ha hecho Meveric en la imagen de Debian Stretch, que alimenta NEMS Linux para ODROID-XU3 /XU4/HC1/HC2. NEMS Linux es gratuito para descargarse y usarse con total libertad. Su código fuente está disponible en GitHub, y la imagen lista para usarse la puedes descargar para ODROID en https://nemslinux.com.

Sobre el Autor

Robbie Ferguson es el presentador de Category5 Technology TV y autor de NEMS Linux. Su programa de televisión lo puedes encontrar en https://category5.tv y su blog es https://baldnerd.com.

Conceptos Básicos de BASH: Introducción a BASH – Parte 5

October 1, 2018 By Erik Koennecke Linux

Nuestra aventura con la programación continúa con más tests, declaraciones if, entradas y funciones; También haremos cálculos desde dentro de BASH. Primero, vamos a echar s un vistazo a la manipulación del nombre de archivo, ya que parece ser una de las cosas más comunes que son necesarias en los scripts de principiantes. A modo bonificación por sobrevivir a esta gran cantidad de teoría, dispondremos de otro ejemplo de programación BASH en forma de juego.

Nombre base

Los primeros scripts que la gente suele escribir son para gestionar y manipular archivos. Tener que hacer simples tareas una y otra vez, como las conversiones, se vuelve aburrido muy rápido. Es natural que incluso las personas reacias a los scripts intenten automatizar esta cuestión tardo o temprano. Normalmente, esto implica guardar el resultado con el mismo nombre de archivo pero con una extensión diferente, por ejemplo, convertir file001.png en file001.jpg luego, file002, file003 y así sucesivamente.

¿Y en qué nos puede ayudar BASH?

Si abres el manual de bash, con man bash, en algún lugar de las casi 6000 líneas del manual, similar a las 35 partes de la serie BASH que estás leyendo en este momento, encontrarás una sección sobre “Sustitución de parámetros”. Te voy a evitar tener que desplazarte por todo esto, existe un truco muy útil para manejar largas páginas del manual:

```
$ man -P "less -p 'Parameter Expansion'" bash
```

Esto salta directamente a la sección que estás buscando, si recuerdas las palabras clave adecuadas. Vamos a resumir un poco nuestros hallazgos, ya que la página del manual es bastante más árida que mi artículo y es tan concisa que tienes que leerla varias veces para lograr sacar algo en claro. La variable escrita por costumbre $foo es una abreviatura, la

versión completa es ${foo}, que también es una versión corta, de $ {foo: operators}. ¿Qué significa esto? Veamos algunos de estos supuestos y luego seguiremos con algunos ejemplos. Se pueden colocar dos operadores dentro de las llaves, # y %. Se pueden utilizar como caracteres individuales o en pareja. Las combinaciones posibles son:

- Para recortar el sufijo más corto dese el final: $ {variable%patrón}
- Para recortar el sufijo más largo desde el final: $ {variable %% patrón}
- Para recortar el prefijo más corto desde el principio: ${variable#patrón}
- Para recortar el prefijo más largo desde el principio: ${variable##patrón}

La diferencia entre “más corto” y “más largo” solo se aplica si está utilizando un comodín Shell * en tu patrón, para algo como jpg, son idénticos. Ahora necesitamos algunos ejemplos para entender cómo funciona esto. Como mejor se aprende es haciendo cosas, puedes experimentar con el comando echo y varias variables shell. Si quieres continuar con el ejemplo, crea una ruta /tmp/odroid con mkdir -p/tmp/odroid, seguido de /tmp/odroid/photos.zip para crear un archivo vacío para te experimento. No es necesario seguir línea por línea, aunque para tu experimento, es mejor tener algo que puedas manejar con facilidad. Aquí tienes los resultados:

Si tenemos la variable

```
foo="/tmp/odroid/photos.zip"
```

(No olvide las comillas para asegurarte de que puedes gestionar espacios y otros caracteres especiales complejos) Puedes usar

- foo=/tmp/odroid/photos.zip”; echo “${foo%/*}” para mostrar la ruta /tmp/odroid. Este es un método fácil para obtener el correspondiente nombre de directorio de un archivo.
- foo=/tmp/odroid/photos.zip”; echo “${foo##*/}” genera photos.zip, el nombre base del archivo.
- foo=photos.zip”; file=”${foo%%.*}” proporciona photos, y foo=photos.zip”; echo “${foo#*/}” proporciona zip.

Otra aplicación útil de los paréntesis es la expansión de los mismos. Por ejemplo, si deseas hacer un archivo de copia de seguridad de algo, pero eres demasiado vago para escribir la ruta completa y el nombre de archivo del origen y del destino: cp /path/to/your/file.doc{,.bak} se amplia a cp /path/to/your/file.doc /path/to/your/file.doc.bak y hace un archivo de copia de seguridad de tu documento en el mismo directorio. ${#foo} proporciona el número de caracteres de la variable. foo=4-letter-word; echo ${#foo} is 13.

Declaraciones if y tests

Para ver en detalle las declaraciones if y tests, recurriremos a un script de ejemplo con todos los bloques básicos que pueden aparecer, tal y como se muestra a continuación:

```
#!/bin/bash  #Start of every BASH script
# Basic if statement # Comment, script purpose
if [ $1 -gt 9 ]  #test condition and if
statement
then
    echo $1, that's unusual for a lucky
number.  # if statement true
    date  # if statement true
fi
Pwd  #always executed
then
    date
else
    pwd
...
```

El fragmento anterior podría ser parte de un script de este tipo. También hay if – elif – else o la abreviatura de la sentencia ‘else if’, sentencias de casos, o la opción de múltiples tests con las condiciones AND u OR con los operadores && para AND y || para OR, aunque esto está fuera del alcance de este artículo. Los corchetes, [], en la declaración if anterior son una referencia al comando test. Aquí también se pueden usar todos los operadores que test permita. Localiza la página de manual sobre test para ver todos los operadores posibles; Algunos de los más comunes se enumeran a continuación.

- ! EXPRESSION – La EXPRESSION es falsa.
- -n STRING – La longitud de STRING es mayor que cero.

- -z STRING – La longitud de STRING es cero (= está vacío).
- STRING1 = STRING2 – STRING1 es igual a STRING2
- STRING1 != STRING2 – STRING1 no es igual a STRING2
- INTEGER1 -eq INTEGER2 – INTEGER1 es numéricamente igual a INTEGER2
- INTEGER1 -gt INTEGER2 – INTEGER1 es numéricamente mayor que INTEGER2
- INTEGER1 -lt INTEGER2 – INTEGER1 es numéricamente menor que INTEGER2
- -d FILE – FILE eexiste y es un directorio.
- -e FILE – FILE existe.
- -r FILE – FILE existe y el permiso de lectura está concedido.
- -s FILE – FILE existe y su tamaño es mayor que cero (= no está vacío).
- -w FILE – FILE existe y el permiso de escritura esta concedido.
- -x FILE – FILE existe y el permiso de ejecución esta concedido.

Entrada, mas variables

Para la entrada BASH, si quieres tener un script interactivo, puede usar 'read' para leer la variable desde la entrada del usuario. Simple script de ejemplo, greeting.sh:

```
echo Who are you?
read name
echo It's nice to meet you, $name.
```

Puesto que las comillas tienes un significado especial en BASH, Éste logra escaparse a través de la barra invertida. read -p y read -sp se pueden usar para solicitar y guardar silencio, lo que significa no repetir la entrada escrita:

```
read -p 'Username: ' user
read -sp 'Password: ' pass
```

Almacena el nombre de usuario en $user y la contraseña en $pass, sin mostrarlo. Si el usuario introduce varias palabras cuando se le solicite, read varx vary varz almacenaría tres palabras en las tres variables diferentes. Si son más de tres palabras, la última variable almacena la entrada restante. Aquí tienes algunas variables especiales que puedes usar en tus scripts:

- $0 – El nombre del script Bash.
- $1 – $9 – Los primeros 9 argumentos del script Bash.
- $# – Número de argumentos pasados al script Bash.
- $@ – Todos los argumentos ofrecidos al script Bash.
- $? – El estado de salida del proceso ejecutado recientemente.
- $$ – El ID de proceso del script actual.
- $USER – El nombre de usuario del usuario que ejecuta el script.
- $HOSTNAME – El hostname de la máquina en la que se ejecuta el script.
- $SECONDS – El número de segundos desde que se inició el script.
- $RANDOM – Devuelve un número aleatorio diferente cada vez que se hace referencia.
- $LINENO – Devuelve el número de la línea actual en el script Bash.

Estos deberían ser bastante simples, con $0 necesitan algo más de detalle: con esto, puedes hacer que tu script actúe de manera diferente según el nombre que utilices, por ejemplo, comprimir y descomprimir archivos. Hay disponible una lista completa de variables predefinidas que puedes usar si escribes 'env' en la línea de comando.

Con esto, ahora existen 3 métodos para pasar datos a un script BASH, y cuál de ellos es el mejor método depende de la situación:

- Argumentos de línea de comando, como ourscript.sh foo bar baz proporciona $1=foo, $2=bar and $3=baz.
- Leer la entrada durante la ejecución del script, véase antes.
- Datos que han sido redirigidos al script mediante stdin. Esto es más para programadores experimentados, básicamente, es lo mismo que canalizar los resultados de otro script o función hacia nuestro script con '|'.

Usamos comillas a veces sin explicar para qué sirven. Como BASH usa un espacio para separar los elementos, device = Odroid XU4; echo $device emite un mensaje de error que indica que no se puede encontrar 'XU4', porque el contenido de la variable es solo 'Odroid' y BASH intenta ejecutar 'XU4'. Si quieres una variable con espacios, necesitas usar comillas. El texto entre comillas indica a BASH que los contenidos

deben considerarse como un solo elemento. Puede usar comillas simples ‘ o dobles “. Dependiendo de lo que quiera hacer, cualquiera de las dos es importante:

- Las comillas simples procesa literalmente todos los caracteres. echo ‘I am $USER’ devuelve I soy $USER.
- Las comillas dobles te permitirán realizar sustituciones (es decir, incluir variables que son valoradas). echo “I am $USER” devuelve yo soy odroid.

También puede usar la sustitución de comandos para tener una variable rellena con la valoración de los comandos – foo=$(ls/etc | wc -l); echo Hay entradas $foo en /etc. Muestra cuántas entradas de configuración hay en tu directorio /etc.

Si deseas tener variables disponibles para otros scripts, debes exportarlas. export foo hace que $foo esté disponible para los siguientes scripts, pero si cambian $foo, no tendrá ningún impacto en el script de exportación. Piensa en ello como cuando haces una copia y la distribuyes.

Cálculos

Vamos a tratar los cálculos en BASH brevemente. Bastante es saber que $ ((...)) evalúa el término entre corchetes dobles. De modo que, si quieres saber rápidamente cuántos segundos tiene un año, escribe:

```
$ echo $((60*60*24*365))
```

Esto debería devolver 31536000. Sin embargo, solo se permiten valores enteros. Esto es así para la entrada de datos y el resultado – BASH calcula el resultado fraccionado a la baja. echo $ ((4/5)) da 0, y solo echo $ ((5/5)) le dará 1 como resultado.

Aplicación práctica de BASH

Otro juego como recompensa por sobrevivir a todos los aspectos teóricos de BASH, en torno a 350 líneas de scirpt BASH tiene el juego 2048. Puede jugarlo con las teclas de cursor, el enlace al script lo tienes en las referencias. Descárgalo con wget.

Esta ha sido una extensa y magnífica información, en la siguiente parte nos tomares un respiro en relación a la teoría y analizaremos algunas aplicaciones de línea de comandos interesantes y unos cuantos scripts BASH útiles. También un poco más (¡solo un poco!) de programación con bucles y funciones.

Referencias

http://linuxg.net/curly-brackets-expansion-in-bash-with-5-practical-examples/
https://raw.githubusercontent.com/mydzor/bash2048/master/bash2048.sh

Driver de Video GBM

October 1, 2018 By @AreaScout Juegos, ODROID-XU4, Tutoriales

Esta es una guía para instalar los drivers de espacio de usuario para activar GBM y compilar emuladores de juegos retro. En primer lugar, mostraremos el uso de la librería de espacio de usuario que activa Mali GBM. Veremos qué tal funciona y para qué se puede usar. Utilizaremos la pantalla ODROID-VU5A para mostrar algunos ejemplos.

Instalaremos una imagen mínima de Ubuntu 18.04 para XU4 y prepararemos el sistema para usar el driver Mali GBM. Como resultado, PPSSPP podrá ejecutarse para emular perfectamente los juegos sin ningún tipo de problema. El video que muestra GoW – Chains of Olympus en ODROID-XU4 lo puedes encontrar en **https://youtu.be/QegJlwflkZk**.

Instalar la imagen

Descarga la imagen: ubuntu-18.04-4.14-minimal-odroid-xu4-20180531.img.xz
desde **https://odroid.in/ubuntu_18.04lts/**. Extrae y

escribe la imagen en el soporte de arranque, utilizando la guía de la wiki de Hardkernel.

Arranca el ODROID-XU4 usando el soporte de arranque y usa las credenciales de inicio de sesión:

Nombre de Usuario: root Contraseña: odroid

Actualiza el software del sistema usando los comandos:

```
$ apt update && apt full-upgrade -y
```

Actualiza el software del sistema usando los comandos:

```
$ apt-get install mali-x11 libx11-dev libsm-dev libxext-dev git cmake mercurial libudev-dev libdrm-dev zlib1g-dev pkg-config libasound2-dev alsa-utils htop bc ifupdown2 net-tools libssl1.0-dev mlocate bluez libfreetype6-dev libgbm-dev
```

Agrega el usuario odroid y añade el ID del usuario a los grupos:

```
$ adduser odroid usermod -aG Sudo,adm,audio,operator,input,video,tty,staff, games,users, plugdev,netdev,bluetooth,disk odroid
```

Inicia sesión con el usuario recientemente creado:

```
$ su - odroid
```

Activa el prompt de color:

```
$ sed -i '1iforce_color_prompt=yes' ~/.bashrc
$ su odroid
```

Cambia el driver de pantalla mali-x11 por el que activa GBM. Normalmente, esto es relativamente fácil de hacer simplemente actualizando eglplatform.h, que es multiplataforma (es decir, fbdev x11 y GBM). Sin embargo, necesitamos también actualizar otras cabeceras

```
$ cd /usr/include/EGL
$ sudo rm eglplatform.h
$ sudo wget https://www.khronos.org/registry/EGL/api/EGL/eglplatform.h
$ cd /usr/include/GLES2
$ sudo rm *
$ sudo wget https://www.khronos.org/registry/OpenGL/api/GLES2/gl2platform.h
$ sudo wget https://www.khronos.org/registry/OpenGL/api/GLES2/gl2.h
$ sudo wget https://www.khronos.org/registry/OpenGL/api/GLES2/gl2ext.h
$ cd /usr/include/GLES3
$ sudo rm *
$ sudo wget https://www.khronos.org/registry/OpenGL/api/GLES3/gl3.h
$ sudo wget https://www.khronos.org/registry/OpenGL/api/GLES3/gl31.h
$ sudo wget https://www.khronos.org/registry/OpenGL/api/GLES3/gl32.h
$ sudo wget https://www.khronos.org/registry/OpenGL/api/GLES3/gl3platform.h
```

Ahora consigue el nuevo driver binario:

```
$ cd ~
$ wget http://deb.odroid.in/bigmali.tar
$ tar xf bigmali.tar libmali.so
$ sudo mv libmali.so /usr/lib/arm-linux-gnueabihf/mali-egl/.
```

Añade un enlace simbólico que falta:

```
$ sudo ln -s /usr/lib/arm-linux-gnueabihf/mali-egl/libmali.so /usr/lib/arm-linux-gnueabihf/libGLESv1_CM.so.1
```

Agrega un nuevo directorio, abre el editor vi con el archivo /usr/local/lib/pkgconfig/gbm.pc:

```
$ sudo mkdir /usr/local/lib/pkgconfig
$ sudo vi /usr/local/lib/pkgconfig/gbm.pc
```

Añade la siguiente sección en el nuevo archivo:

```
prefix=/usr/local
exec_prefix=${prefix}
includedir=${prefix}/include
libdir=${exec_prefix}/lib
Name: libgbm
Description: A small gbm implementation
Version: 19.0.0
```

```
Cflags: -I${includedir}
Libs: -L${libdir} -lgbm
```

Guarda y cierra vi. Añade enlaces simbólicos para libgbm:

```
$ cd /usr/local/lib/
$ sudo ln -s /usr/lib/arm-linux-
gnueabihf/mali-egl/libmali.so libgbm.so
$ sudo ln -s libgbm.so libgbm.so.19
$ sudo ln -s libgbm.so.19 libgbm.so.19.0.0
$ sudo ldconfig
```

Coge gbm.h y borra el de mesa:

```
$ cd /usr/include
$ sudo rm gbm.h
$ sudo wget -O gbm.h
https://pastebin.com/raw/5QUd011t
```

Descarga SDL2, compílalo e instálalo:

```
$ cd ~
$ hg clone http://hg.libsdl.org/SDL SDL2
$ cd SDL2
$ ./configure --disable-video-opengl --enable-
video-kmsdrm
```

Una vez finalizada la configuración, deberías ver esta línea:

```
Video drivers : dummy x11(dynamic)
kmsdrm(dynamic) opengl_es1 opengl_es2 vulkan
```

Ahora tenemos que editar SDL_config.h para abrir permanentemente nuestro libgbm.so.19 en lugar de libgbm.so.1 de Mesa. Compílalo e instálalo:

```
$ sed -i -e 's/libgbm.so.1/libgbm.so.19/g'
include/SDL_config.h
$ make -j7
$ sudo make install
```

Una vez completada la compilación podemos hacer pruebas:

$ cd test $ ./configure $ make -j7 $ ./testgles2

Deberías ver un cubo giratorio, espera unos segundos y salte con la tecla ESC. Si estás utilizando VU5A, observarás algo similar a esto lo cual indica que es demasiado lento:

```
INFO: 56.89 frames per second
```

Deberías alcanzar los 60 fps, sino tendremos problemas con los emuladores. Si intentan mantener solo 60 fps y no lo consiguen, entonces es que realmente se ralentizarán mucho las cosas. Podremos seguir observando los videos de introducción de los juegos PPSSPP.

Compilando el Kernel para >60fps en VU5A

Tenemos que parchear el kernel para conseguir más de 56 fps de procesamiento. Hasta que se publiquen los cambios necesarios en el kernel principal, tendremos que hacerlo por nuestra cuenta.

El reloj de píxeles y, probablemente, algunos valores de sincronización H o V de la configuración HDMI PHY no son los correctos para la pantalla VU5A. Los detalles los puedes encontrar en https://goo.gl/CVJYS6. Hardkernel tomó el más cercano, pero lo eligieron para estar por debajo de los 60 fps. Para la emulación es mejor elegir el que esté por encima de los 60 fps, de modo que al final tendremos una velocidad de fotogramas de 64 fps.

Asegúrate de tener suficiente espacio en el almacenamiento del sistema y luego coge la fuente del kernel. Seleccionaremos la configuración de HDMI PHI del siguiente reloj de píxeles más alto, en realidad lo que hay dentro de esta configuración HDMI PHY es desconocido para el público. Hemos preguntado a varios desarrolladores, pero nadie ha sabido darme una respuesta, solo que en el kernel 3.10 hay un pequeño código para cambiar esta configuración de 32 bytes. Así que solo se conocen unos cuantos bytes, aunque no cambian la velocidad de refresco.

```
$ git clone
https://github.com/hardkernel/linux.git
$ cd linux
$ wget -O VU5A.patch
https://pastebin.com/raw/aWEYArWL
$ patch -p1 < VU5A.patch

$ make odroidxu4_defconfig
$ make -j7
$ sudo cp arch/arm/boot/zImage /media/boot/.
$ sudo cp arch/arm/boot/dts/exynos5422-
odroidxu4.dtb /media/boot/.
$ sudo make modules_install
```

Reinicia y testea el valor de fps nuevamente:

```
$ cd SDL2/test
$ ./testgles2
```

Ahora deberías ver algo como esto:

```
INFO: 64.01 frames per second
```

Ahora compilaremos PPSSPP y luego RetroArch con el back-end GBMS KMSDRM, inclusive libretro

Compilando PPSSPP

PPSSPP es un emulador de PSP para Android.

Coge el código fuente y aplica un parche necesario:

```
$ cd ~
$ git clone --recursive
https://github.com/hrydgard/ppsspp.git
```

FFmpeg debe compilarse antes de compilar el binario PPSSPP. Los binarios pre-compilados son todos para puntos flotantes débiles y necesitamos hardfp:

```
$ ./linux_armhf.sh
$ cd ..
```

Antes de que empecemos a compilar, debemos convertir nuestra copia de /usr/include/GLES2/gl2ext.h en una específica del proveedor deshabilitando el uso de GL_EXT_buffer_storage. Se crea un archivo de copia de seguridad gl2ext.h.back. Nuestra librería Mali no incluye/exporta esa función, así que no podemos definirla.

```
$ sudo sed -i.bak '/^#ifndef
GL_EXT_buffer_storage$/,/^$/d'
/usr/include/GLES2/gl2ext.h
```

Es posible que quieras configurar el sistema para usar solo 4 núcleos en FFmpeg con el fin de realizar pequeños ajustes y experimentar. He observado que FFmpeg con subprocesamiento no funciona muy bien cuando se eligen todos los núcleos con HMP (cambiando las tareas más exigentes de LITTLE CPU a BIG CPU). Para esto, puede editar el archivo Core/HW/MediaEngine.cpp en la línea número 475, y modificarla para usar solo 4 núcleos (mejor cambiar de 4 LITTLE a 4 BIG en lugar de usar los 8 núcleos).

No obstante, esto ha sido analizado en Moonlight con archivos de video 1080p de juegos por streaming. Los archivos de video PPSSPP no son tan grandes, de modo que, tal vez no requieran tanta CPU, así que puede afectar muy poco o prácticamente nada:

```
av_dict_set(&opt, "threads", "4", 0);
```

Ahora, necesitamos generar el Makefile.

```
$ cmake -DUSING_EGL=OFF -DUSING_GLES2=ON -
DUSE_FFMPEG=YES -DUSE_SYSTEM_FFMPEG=NO.

Empieza a compilar el binario:
```

```
$ make -j7
```

Si estás utilizando VU5A, ahora contarás con la función de pantalla táctil en el menú y también podrás activar “los controles táctiles en pantalla” si quieres. Puedes ver un video en https://youtu.be/QegJlwflkZk?t=374. Ahora puedes marcar tu PPSSPP emulada con una única región generando la correspondiente configuración regional. En mi caso, utilicé de_AT:

```
$ sudo locale-gen de_AT.UTF-8
$ sudo update-locale LANG=de_AT.UTF-8
```

Algunos juegos pueden usarla para el idioma dentro del juego. La descarga https://goo.gl/BhHLxm contiene mi configuración para GoW, incluidos algunos reemplazos de texturas para Star Wars, The Clone Wars y Star Wars, y The Force Unleashed. Estos juegos son jugables.

El directorio de configuración de PPSSPP debería verse así:

```
odroid@odroid:~$ tree -d .config/ppsspp/
.config/ppsspp/
└── PSP
├── PPSSPP_STATE
├── SAVEDATA
│ ├── ULES01284SAVE00
│ └── ULES01376SYSDATA
├── SYSTEM
│ └── CACHE
└── TEXTURES
├── ULES00981
└── ULES01284

10 directories
```

Compilar y configurar RetroArch

Puedes ver un video de esto en https://youtu.be/6Ewgov7_TXM. Además de RetroArch, necesitamos un pequeño parche que nos evita tener el menú con sólo un fondo negro:

```
$ git clone
https://github.com/libretro/RetroArch.git
$ cd RetroArch
$ wget -O retro.patch
https://pastebin.com/raw/1SCeb8EG
$ patch -p1 < retro.patch
$ ./configure --enable-opengles3 --enable-
opengles --enable-neon  --enable-floathard --
enable-freetype
$ make -j7
$ sudo make install
$ retroarch
```

Aplica algunas configuraciones útiles. Simplemente es una sugerencia para que configures RetroArch, una vez instalado ya no tendrás que seguir este tutorial. Actualiza los recursos (iconos, imágenes de fondo y esas cosas). Puedes encontrar la información necesaria en los siguientes menús:

```
MainMenu -> Online Updater -> Update Assets
```

Te sugerimos que también actualices estos paquetes:

```
Core Info Files, Joypad Profiles, Database,
GLSL Shaders
```

Además, puedes usar Core Updater para conseguir algunos emuladores. A continuación, activa la Configuración avanzada:

```
Settings -> User Interface -> Show Advanced
Settings -> ON
```

Activa Threaded Video, notarás que mejorará bastante la emulación:

```
Settings -> Video -> Threaded Video -> ON
```

Activa el contador de FPS, es útil ver lo rápido que se ejecuta la emulación, especialmente cuando estás configurando cosas:

```
Settings -> Onscreen Display -> Onscreen
Notifications -> Display Framerate -> ON

Settings -> Onscreen Display -> Onscreen
Notifications -> Show frame count on FPS
Display -> OFF

Settings -> Driver -> Audio Driver ->
alsathread
```

y si está utilizando VU5A:

```
Settings -> Onscreen Display -> Onscreen
Notifications -> Notification size -> 18

Settings -> Onscreen Display -> Onscreen
Notifications -> Notification X position ->
0.010

Settings -> Onscreen Display -> Onscreen
Notifications -> Notification Y position ->
0.010
```

Si ya tienes juegos instalados en tu ODROID-XU4, búscalos utilizando Import Content -> Scan Directory y selecciona la carpeta donde almacenas tus juegos para que RetroArch los escanee. Aparecerán en la parte derecha del menú después de un tiempo.

Compilar y configura Versatile Commodore Emulator

Puedes ver un video de esto en: https://youtu.be/ItkppnXWd9U

Primero instalaremos Cice libretro y luego mame libretro.

Instalar los prerrequisitos:

```
$ sudo apt-get install bison
```

Descarga el código fuente y aplica el parche para máquinas VIC-II Commodore si quieres. Esto eliminará los margenes de los modelos de máquina C64 y C128, los juegos se ven mejor sin él. Este es un modo rápido de hacerlo. Una mejor forma sería añadirlo a la configuración de libretro.

Si un juego sobrepasa los márgenes, no funcionará y el sistema probablemente se comporte de forma indebida, aunque son pocos los juegos los que se acercan a los márgenes.

```
$ git clone https://github.com/libretro/vice-
libretro.git
$ cd vice-libretro
$ wget -O noborder.patch
https://pastebin.com/raw/VwtSDj50
$ patch -p1 < noborder.patch
```

Empieza a compilar una máquina Commodore. Los tipos de máquina válidos son:

```
x128, x64, x64sc, x64dtv, x64scpu, xplus4,
xvic, xcbm5x0, xcbm2, xpet
```

Debes añadir la variable EMUTYPE seguida del tipo de máquina que quieres compilar. Si no agregas esta variable, entonces la x64 (C64) será el tipo de máquina por defecto.

```
$ make EMUTYPE=x64 -f Makefile.libretro -j7
```

Si deseas compilar más de un tipo de máquina, no olvides ejecutar el comando clean sobre el proyecto, de lo contrario el núcleo no funcionará:

```
$ make EMUTYPE=x64 -f Makefile.libretro -j7
clean
```

Configuración de Retroarch

Copia el binario en la carpeta cores del RetroArch:

```
$ cp vice_x64_libretro.so
~/.config/retroarch/cores/.
```

Inicia RetroArch y selecciona el núcleo vice – inicia el núcleo sin juego o con él. Presiona el botón Guía en tu mando o F1 en el teclado y desplázate hacia abajo hasta las Opciones, entra y desactiva DriveTrueEmulaton-> OFF. Te llevará algo de tiempo cargar un juego y configurar Controller0Type en el joystick.

También puede habilitar una relación de aspecto de 16:10. Es un buen término medio entre 4: 3 y 16: 9.

```
Settings -> Video -> Aspect Ratio -> 16:10
```

Con el botón de Inicio puedes activar la configuración GUI nuklear (el botón Select debe pulsarse una vez para activar el raton) desde allí puede elegir el Joyport

C64, cpu de la máquina, tipo de sid y mucho más. El teclado en pantalla se activa con el botón x (diseño de Xbox)

Compilar y configurar Reicast Core – un emulador de Dreamcast

El video lo pueden encontrar en https://youtu.be/j0jEUcQx-vM. Descarga la fuente y aplica un parche como de costumbre:

```
$ cd ~
$ git clone
https://github.com/libretro/reicast-
emulator.git
$ wget -O xu4.patch
https://pastebin.com/raw/pfVjnVs3
$ patch -p1 < xu4.patch
$ platform=odroid ARCH=arm make -j7
$ strip reicast_libretro.so
$ cp reicast_libretro.so
~/.config/retroarch/cores/.
```

Primero necesitas alguna bios para NAOMI y Dreamcast. Unos buenos enlaces para conseguirlas y usarlas son https://goo.gl/a5JbMT y https://docs.libretro.com/library/reicast/. Si deseas conocer la suma de comprobación md5 del archivo bios de NAOMI, puede consultar el archivo de información:

```
/home/odroid/.config/retroarch/cores/reicast_l
ibretro.info
```

Dentro de un juego, abre el menú RetroArch y dirígete a las opciones básicas. Encuentra la siguiente configuración y cámbiala a los siguientes valores, que son los más importantes para lograr una velocidad decente:

```
reicast_framerate = "normal"
reicast_enable_rttb = "enabled"
reicast_threaded_rendering = "enabled"
```

Para comentarios, preguntas y sugerencias, visita el post original en https://forum.odroid.com/viewtopic.php?f=98&t=32173.

Campamento de Programación – Parte 6: Generar Sonido desde el Altavoz del ODROID-GO

October 1, 2018 By Justin Lee ODROID-GO, Mecaniqueo, Tutoriales

Vamos a aprender cómo usar la salida DAC como un generador de tonos. La librería odroid_go.h y su instancia GO tienen una instancia Speaker para usar el altavoz con facilidad. Por lo tanto, puedes jugar un poco con GO.Speaker. Algunas de las funciones de GO.Speaker son:

- setVolume(): para ajustar el nivel de volumen. El parámetro dado puede ser de 0 a 11 (silencio).
- playMusic(): para reproducir música que está escrita en números enteros de 8 bits. El parámetro dado es una frecuencia de muestreo correcta para reproducir música.
- beep(): para reproducir un simple pitido.
- tone(): para reproducir un simple pitido con los parámetros de frecuencia y duración en milisegundos. Puedes omitir el argumento de duración.

Vamos a escribir el código que reproduce un sonido cuando se presiona un botón. Usaremos los botones A, B e Inicio y haremos que estos botones reproduzcan un sonido que sea diferente entre sí. Para obtener información sobre cómo se utilizan los botones, consulta el ejemplo de los Botones. También vamos a mostrar en la pantalla LCD qué botón se ha presionado.

Para conocer cómo usar la pantalla LCD, consulta el ejemplo de Hello World. Podemos escribir el código fuente tal y como se muestra a continuación. Inicia la placa llamando a la función GO.begin () y coloca el código dentro de la función loop () que se activa cuando se presiona el botón wasPressed().

```
#include

void setup() {
// put your setup code here, to run once:
GO.begin();

GO.lcd.printf("ODROID-GO speaker test:
```

```
");
GO.Speaker.setVolume(8);
GO.Speaker.playMusic(m5stack_startup_music,
25000);
}

void loop() {
// put your main code here, to run repeatedly:

if(GO.BtnA.wasPressed()) {
GO.lcd.printf("wasPressed: A
");
GO.Speaker.beep();
}

if(GO.BtnB.wasPressed()) {
GO.lcd.printf("wasPressed: B
");
GO.Speaker.tone(3000, 200);
}

if(GO.BtnStart.wasPressed()) {
GO.lcd.printf("wasPressed: Start
");
GO.Speaker.playMusic(m5stack_startup_music,
25000);
}

GO.update();
}
```

Presiona CTRL-U para compilar y cargar el esquema, y presiona el boton A, B o Inicio para reproducir un sonido.

Ejemplo completo

El ejemplo completo lo tienes disponible haciendo clic en el menú Files → Examples → ODROID-GO → Speaker para importar y presiona CTRL-U para compilar/cargar, como se muestra en la Figura 1.

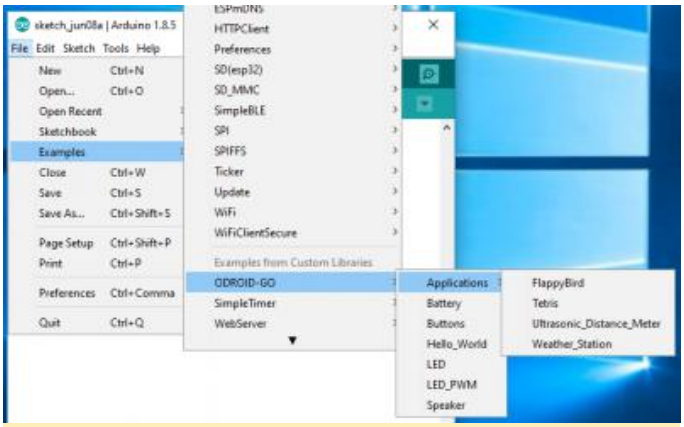
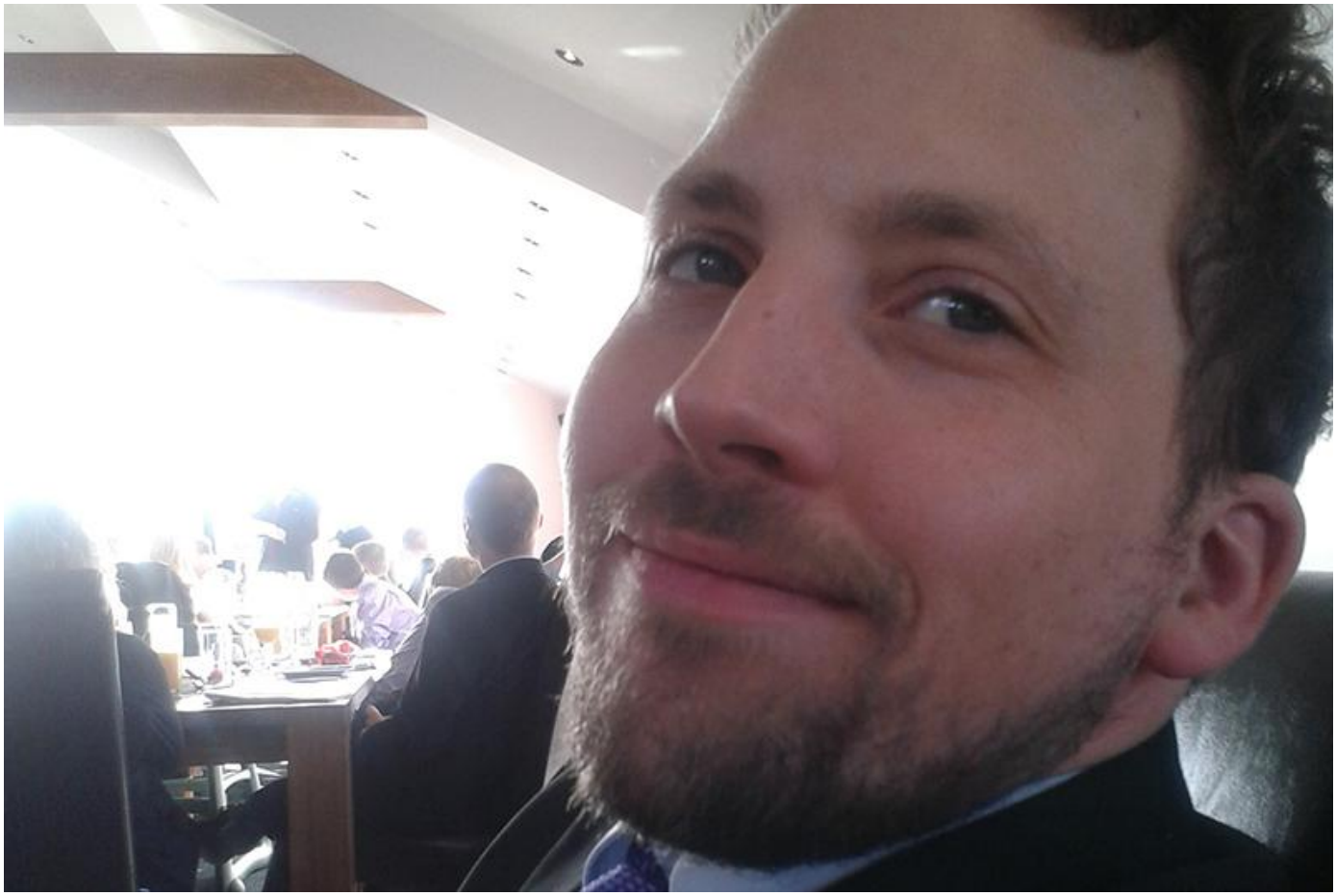


Figura 1 – Accediendo al ejemplo completo

Para comentarios, preguntas y sugerencias, visita el artículo original en https://wiki.odroid.com/odroid_go/arduino/06_speaker.

Conociendo un ODROIDian: David Knight

October 1, 2018 By Rob Roy Conociendo un ODROIDian



Por favor háblanos un poco sobre ti. Vivo en Newcastle, Reino Unido, trabajando como optometrista en el sector de la cirugía refractiva. Mi trabajo diario consiste en lidiar con pacientes que se han sometido a una cirugía ocular por láser o de cataratas. Fui muy aplicado e introvertido en mi juventud, siempre tenia la cabeza en los libros. Era bastante bueno en matemáticas, ciencias y ajedrez. En la década de los 80, la informática era todavía una asignatura bastante novedosa y por aquel entonces me interesaba más la biología. Recuerdo haber trabajado en una fábrica de vidriado de gafas durante un verano y poco después decidí licenciarme en Optometría. Después de trabajar en sector privado durante una década, necesitaba un nuevo desafío. Fue entonces cuando me pase a la cirugía refractiva y no me he arrepentido para nada. Actualmente, cuando no estoy trabajando, formando o educando, estoy finalizando un título a tiempo parcial de Informática y TI. ¡La programación es mi pasatiempo y "mi" tiempo!

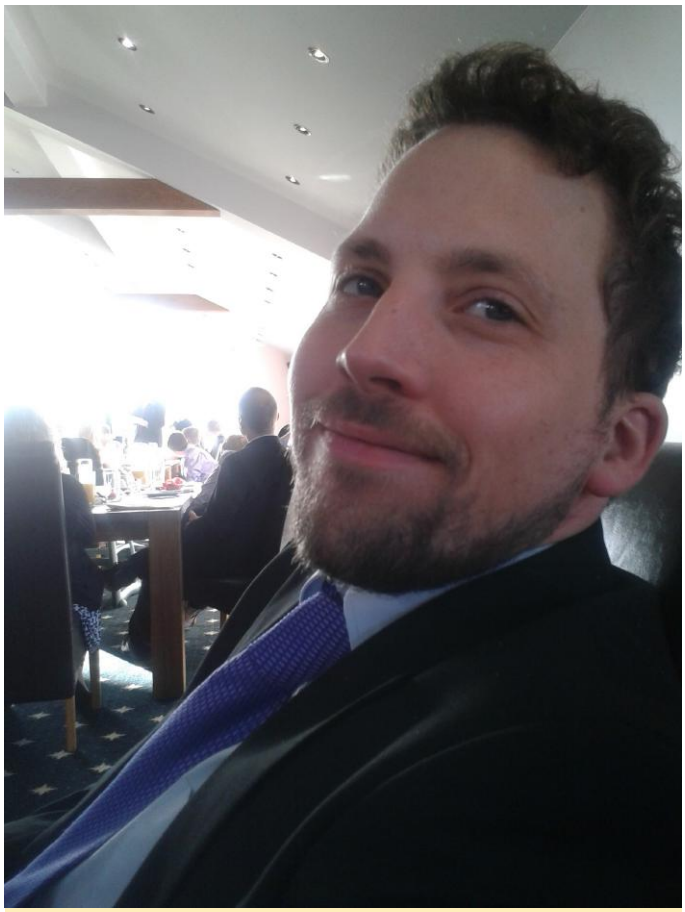


Figura 1 – David Knight

Estoy felizmente casado y tengo cuatro hijos. Mi esposa es entrenadora de fitness, así que no tengo excusa para mantenerme en forma. Tres de mis hijos han terminado la educación y el más joven aún está en la escuela primaria.

¿Cómo empezaste con los ordenadores? Mi primer ordenador fue el ZX Spectrum 48K. ¡Recuerdo haber jugado al Manic Miner cuando tenía 6 años! Como no podíamos permitirnos juegos, recuerdo haber escrito algunos programas BÁSICOS que encontré en revistas, pero realmente no progresé mucho más. Todavía tengo mi Spectrum aunque, lamentablemente, el teclado ha sucumbido al paso del tiempo y ya no funciona. Creo que ahí surgió mi pasión por la informática, ya que recuerdo que quería aprender a programar. En la universidad, fui el único estudiante no informático que se atrevió a entrar en el laboratorio de Unix (¡aunque admito que fue principalmente para jugar a MUDs)!

Después de muchas demoras en 2009, empecé a interesarme superficialmente por Linux, aprendiendo Bash en el proceso. Quería crear una distribución Linux para personas con discapacidades visuales y me involucré en el proyecto Vinux, que modificaba una distribución de Ubuntu para hacerla más accesible a los usuarios con discapacidades visuales, que aún sigue vigente. Para aquel entonces, había aprendido un poco de Python, pero realmente quería familiarizarme con C y C++. Como ejercicio, decidí empezar exportar Passage al portátil GCW Zero. ¡Fue sin duda un ejercicio lleno de frustración! Después de mucho tiempo perdido, logré compilarlo y aún recuerdo la emoción de ver cómo se ejecutaba el juego en mi sistema. ¡Estaba enganchado! (**https://boards.dingoonity.org/gcw-releases/passage/**)

Figura 2 – La maravillosa familia de David.

Desde entonces, creo que he exportado alrededor de 30 emuladores a consolas portátiles, disfrutando con los desafíos que cada proyecto me suponía y, por supuesto, aprendiendo durante todo el proceso. Disfruto especialmente con el desafío de optimizar el código para que pueda ejecutarse sin problemas en sistemas más lentos. Dedico demasiado tiempo a mejorar el rendimiento o me obsesiono con las pequeñas funciones de las que nadie se percata

excepto yo, no me gusta el código ineficiente ni usar más potencia de la necesaria. También disfruto trabajando en la interfaz de usuario para que los controles sean intuitivos y tengan sentido.

¿Qué te atrajo de la plataforma ODROID? Estaba buscando un proyecto de electrónica introductorio fácil y divertido para montarlo con mi hijo Alexander durante las vacaciones de verano. Tenía que ser fácil de construir sin la necesidad de soldar componentes. Después de ver multitud de proyectos de Raspberry Pi, descubrí el ODROID-GO. Realmente disfrutó construyendo el kit, y sus juegos favoritos eran Pac-man y Frogger. Me sorprendió la cantidad de potencia de procesamiento disponible en el ESP32 y lo cómodos que eran los controles. El único problema era que no existía un emulador de Spectrum.

¿Cómo usas tus ODROIDs? Nuestro ODROID-GO pasa la mayor parte de su tiempo en mi maletín del trabajo (por los largos viajes en tren) o conectado a mi ordenador cuando estamos programando con él. Realmente disfrutamos ajustando la configuración y viendo cómo afecta al rendimiento de los emuladores. Hace poco he desarrollado una versión de muy bajo consumo que llegar a funcionar durante 19 horas.

¿Cuál es tu ODROID favorito y por qué? En este momento solo tengo mi ODROID-GO. Me atraen los dispositivos portátiles eficientes de bajo consumo y me encanta el ESP32, ¡tiene un montón de potencia y usa muy poca energía! Realmente disfruté leyendo la Guía de programación y aprendiendo del rico conjunto de características que hay disponibles para los desarrolladores.

¿Qué innovaciones te gustaría ver en los futuros productos Hardkernel? Como dispositivo portátil, lo ideal es que el ODROID-GO tuviera un conector para auriculares (estoy mirando tu Apple). Por supuesto, el ESP32 tiene capacidad para Bluetooth, así que quizás los auriculares inalámbricos sean una posibilidad en el futuro. Además, actualmente el volumen de sonido baja al reducir la profundidad de bits de las muestras de sonido, produciendo un sonido de mala calidad a bajo volumen. Sería genial tener un potenciómetro con quizás, algún tipo de botón o rueda que permitiese controlar el volumen. De forma similar, el brillo de la pantalla podría controlarse de forma autónoma por un LDR.

¿Qué aficiones e intereses tienes aparte de los ordenadores? Aparte de disfrutar con los juegos retro, no me considero realmente un jugador. Cuando no paso tiempo con mi trabajo y mi familia, disfruto corriendo, montando en bicicleta, cantando en un coro y haciendo malabares.

¿Qué consejo le darías a alguien que quiera aprender más sobre programación? La razón por la que me adentré en la programación fue porque necesitaba entender cómo funcionan los ordenadores y pronto me di cuenta de las innumerables posibilidades que ofrecía. Como principiante, desearía que alguien me hubiera relagado “How to Think Like a Computer Scientist”. Es una muy buena introducción a la programación de los ordenadores. Cuando aprendes, es importante conseguir pequeñas victorias; Scratch es una forma estupenda de enseñar los fundamentos y mantener el interés, especialmente con los niños más pequeños. Empieza con lo divertido, pequeños proyectos con simples desafíos que resolver. Street Fighter 2 con gatos? ¡Sí, lo hicimos con Scratch!

Enlaces

Sitio web del proyecto Vinux **http://www.vinux.org.uk/about.html** Mi primera migración compilada **https://boards.dingoonity.org/gcw-releases/passage/** Mis sitios Github y BitBucket **https://github.com/DavidKnight247**, **https://bitbucket.org/DavidKnight247**