Juegos Linux • RetroArch • Conocer un Odroidian • NEMS Linux: Parte 3

ODROID Magazine

Año Seis
Num. #60
Dic 2018

TU ODROID PORTATIL DE LA FORMA CON LA QUE SIEMPRE HAS SOÑADO:

Para ir de intrépido:

Un Tricorder

ODROID-GO

PROGRAMACION:
FABRICA TU PROPIA ESTACION METEOROLOGICA PORTATIL

COMMODORE 64:
AHORA PARA TI, EMULA EL PRIMER Y AUTENTICO ORDENADOR PERSONAL

Juegos Linux – DOSBox, un Emulador de DOS x86: Disfruta de tus Juegos Originales de DOS en HD

December 1, 2018

DOSBox es un emulador de DOS x86 que no solo emula la arquitectura x86, sino que también emula el vulgar entorno de DOS de la era de los años noventa. Con DOSBox, puedes volver a jugar a tus antiguos juegos y ejecutarlos en hardware moderno, ya que existen muchas e ▶️

Campamento de Programación – Parte 9: Montar una Estación Meteorológica Portátil

December 1, 2018

Vamos a aprender a cómo acceder a diversos datos meteorológicos y cómo compartirlos con tus dispositivos móviles a través de la conexión WiFi

El Proyecto Tricorder ODROID-GO

December 1, 2018

Para aquellos que no saben qué es un Tricorder, permitirme que lo explique: En la nueva serie de Star Trek, los personajes a menudo llevan un dispositivo móvil que se usa para medir desgarros en el continuo espacio-tiempo y decir “Está muerto, Jim.”

Compilando un Emulador de Commodore 64

December 1, 2018

Este emulador permite ejecutar juegos que están diseñados para el sistema Commodore 64 de 8 bits.

Campamento de programación – Parte 10: Medir la distancia con ultrasonidos

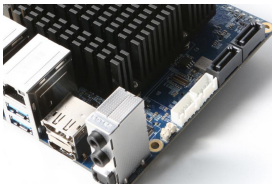
December 1, 2018

Vamos a aprender a usar la salida GPIO, la entrada IRQ y el reloj del sistema con un módulo de medición de distancias por ultrasonidos

Introducción a NEMS Linux: Parte 3 – Configurando monitores de servicio en NEMS Linux

December 1, 2018

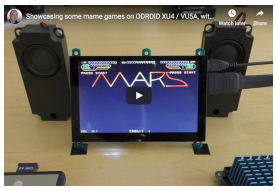
Mi intención con estos artículos siempre ha sido la de presentarte a NEMS Linux de forma que te proporcione conocimientos útiles y los pongan en práctica de inmediato. No tienen la intención de presentarse como simple documentación, sino más bien artículos técnicos que te proporcionen ideas sobre cómo puedes usar ▶️



ODROID-H2 Parte 2: Características de la Bios y Acceso Remoto

December 1, 2018

Como cualquier PC genérico, el ODROID-H2 cuenta con una ROM Flash BIOS de 8MiB soldada a la placa. Cumple con la Especificación 2.6 de UEFI y los requisitos de arranque PXE. Sin embargo, el firmware Intel UEFI no es compatible con CSM versión 2.0 para el arranque de sistemas operativos ▶️



Compilando RetroArch

December 1, 2018

Si estás buscando una interfaz para emuladores de juegos, puedes probar RetroArch. Se ha exportado a la familia de ordenadores de placa reducida (SBC)ODROID-XU4. Puedes seguir los siguientes pasos para instalarlo y utilizarlo en tu sistema.

Conociendo un ODROIDian: Kamots Tech

December 1, 2018

Vivo en Florida (también conocido como el estado del sol), donde nací y crecí. Siempre he vivido en Florida porque hace calor, hay mucho que hacer y la industria TI ha estado en constante crecimiento con muchas promesas en el horizonte. Fui a la universidad para especializarme en redes informáticas ▶️

Juegos Linux – DOSBox, un Emulador de DOS x86: Disfruta de tus Juegos Originales de DOS en HD

December 1, 2018 By Tobias Schaaf Juegos, ODROID-C0, ODROID-C1+, ODROID-C2, ODROID-XU4

DOSBox es un emulador de DOS x86 que no solo emula la arquitectura x86, sino que también emula el vulgar entorno de DOS de la era de los años noventa. Con DOSBox, puedes volver a jugar a tus antiguos juegos y ejecutarlos en hardware moderno, ya que existen muchas e interesantes aplicaciones DOS que no están disponibles para Windows o Linux.

DOSBox supone bastante carga de trabajo para muchos ordenadores, ya que por lo general necesita un PC de gama alta para emular un 486 a 33MHz. Puesto que ODROID usa una arquitectura completamente diferente (ARM vs X86), la cargar de trabajo es todavía mayor durante la emulación. A pesar de su complejidad y sus múltiples capas, DOSBox se ejecuta sorprendentemente bien sobre la plataforma ODROID.

Hace algún tiempo, compilé una versión de DOSBox optimizada para ARMv7 que parecía funcionar más rápido que la versión de DOSBox que viene con la distribución oficial. Me llevo un tiempo comparar estas versiones y descubrir exactamente las mejoras que se consiguen si se usa una compilación optimizada para ARMv7.

A continuación, encontrarás una serie de pruebas comparativas que ponen de manifiesto las diferencias que existen entre la compilación genérica de DOSBox y una compilación hecha específicamente para ARM. La compilación de DOSBox hecha a medida para ARMv7 se puede descargar desde mi repositorio en http://bit.ly/1DhCv6l.

Configuración

En determinadas ocasiones configurar DOSBox puede resultar un tanto complicado. Aunque la mayoría de los juegos no presentan problemas con la configuración básica, algunos sólo funcionan con una configuración muy específica, de modo que he

elegido el conjunto de valores que mejor funcionan en la versión original del juego Quake, ya que se trata de un juego que es bastante exigente con el hardware.

Lo sorprendente de Quake es que el juego en sí está en 3D sin necesitar un entorno de escritorio gráfico. A diferencia de juegos como "Duke Nukem 3D", que contiene algunos objetos 3D y usa sprites 2D en muchas situaciones, Quake ya usaba modelos 3D, similares a los modelos utilizados en los juegos posteriores de Windows, lo cual era bastante sorprendente por aquel entonces.

No fue nada fácil encontrar la configuración correcta y tras un tiempo haciendo pruebas, terminé con los siguientes resultados, con el frameskip y la relación de aspecto desactivados:

```
core=dynamic
cputype=pentium_slow
cycles=fixed 32000
cycleup=500
cycledown=300
memsize=32
scaler=normal3x
```

Los núcleos Dynamic deben usarse para cualquier valor de ciclos fijos de más de 20,000. Pentium_slow es la CPU con la mayoría de las funciones, y configuré los ciclos en 32,000, lo cual es muy alto. Algunos programas de prueba reportaban que se trataba de una rápida CPU Pentium de 1285 MHz. Elegí un número tan alto por el Quake, ya que, con 32,000 ciclos, el juego ofrece la mejor experiencia en ambas versiones de DOSBox.

Pruebas

Después de realizar varias pruebas, descubrí que lo realmente difícil era encontrar buenos indicadores de referencia. Me acordé de algunas aplicaciones de pruebas de rendimiento de cuando DOS era popular, pero me resultó muy difícil localizarlas. No obstante, logre encontrar un entorno de prueba para realizar diferentes pruebas de rendimiento en DOS llamado DOS Benchmark, que está disponible para descargarse en http://bit.ly/1ttzaRR.

DOS Benchmark ofrece pruebas de CPU, GPU y memoria, así como versiones demo de los juegos Doom y Quake para evaluar el entorno. Intenté llevar a cabo todas las pruebas disponibles, pero no todas funcionaron, aunque sí algunas funcionaron bastante bien. Por ejemplo, encontré una prueba con un cubo 3D giratorio que se ejecuta en DOS, que tiene excelentes imágenes, y que llegó a ejecutarse con bastante fluidez en el ODROID.

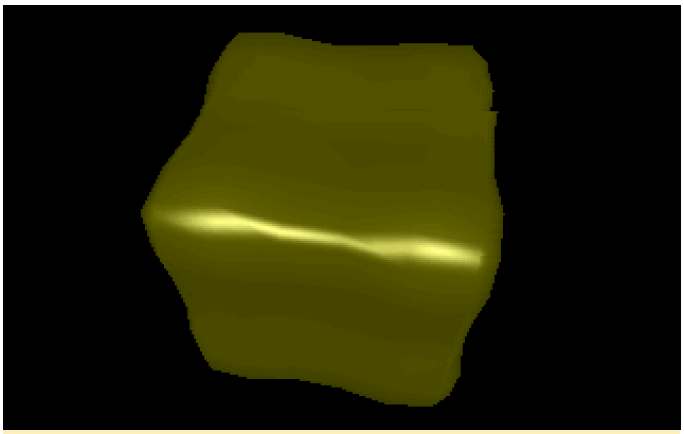


Figura 1 – Cubo giratorio bajo DOS

Figura 2 – Versión Debian estándar de DOSBox

Figura 3 – Versión optimizada para ARMv7 de DOSBox

Prueba de rendimiento 3D

La versión optimizada ARMv7 fue casi un 17% más rápida en esta prueba. Desafortunadamente, esta prueba no es muy fiable si cambias los ciclos de la CPU como lo hice yo. Puedes lograr resultados con más de 200 FPS con valores de 100,000 ciclos de CPU, pero incluso con estos valores tan altos, el emulador estaba lejos de funcionar mejor o incluso más rápido. Pude observar que la salida de video se ralentizaba y se apreciaban saltos de fotogramas, aun así, la prueba logró alcanzar una buena puntuación.

Figura 4 – Prueba de rendimiento 3D utilizando la compilación estándar

Figura 5 – La prueba de rendimiento 3D muestra diferencias en los resultados usando la compilación ARM

Prueba de rendimiento CPU

Las pruebas sobre la CPU mostraron que la versión optimizada para ARMv7 funciona un poco mejor. Una mejora de alrededor del 30% era habitual cuando comparamos el potencial de la CPU.

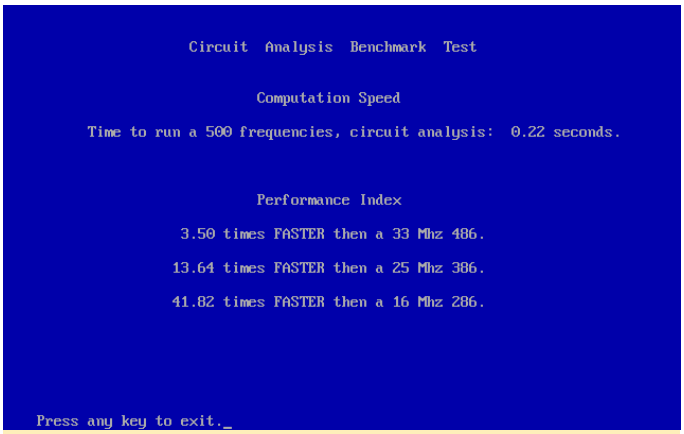


Figura 6 – Prueba de rendimiento de la CPU usando la compilación estándar

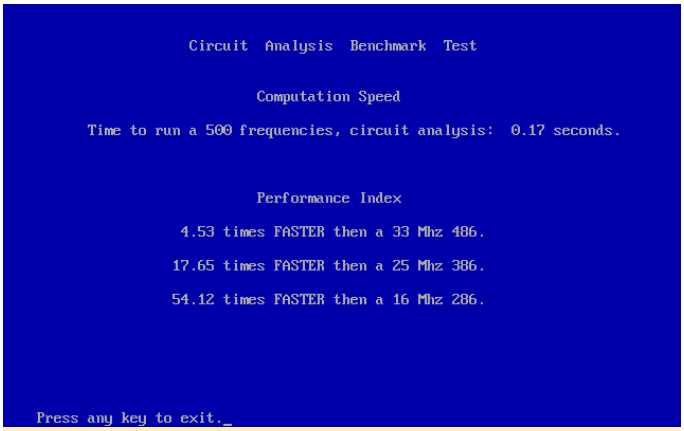


Figura 7 – Prueba de rendimiento de la CPU usando la compilación ARM, que es claramente más rápida

Problemas de memoria

Aunque algunas pruebas de rendimiento tuvieron mejores resultados en la versión ARM, observé varios problemas importantes en determinadas pruebas con la versión optimizada ARMv7. Algunas pruebas ni siquiera llegaban a ejecutarse en la versión optimizada ARMv7 de DOSBox, o provocaban un comportamiento extraño. Solo la versión Debian estándar lograba ejecutar correctamente el 100% de las pruebas.

Por ejemplo, había una prueba de memoria que usaba bloques de diferentes tamaños y llevaba a cabo algunas operaciones con ellos hasta que al final los diferentes bloques agregasen 24 MB en total. Funcionaba con bloques de 384 x 64KB y proporcionaba un resultado sobre la rapidez con la que la memoria hacía el cálculo. La misma prueba en la versión optimizada tuvo resultados muy diferentes. La prueba ARM no solo tardó aproximadamente 10 veces más en ejecutarse, sino que los valores proporcionados fueron completamente inexactos. En

lugar de 24MB, sumó bloques de 512 MB y mucho más a una velocidad ridícula.

Algunas pruebas eran tan intensas, que se salían de la escala y tenían como resultado una velocidad negativa o con altos exponentes realizando cálculos con diez mil megabytes por segundo. Otras pruebas no se iniciaban en absoluto, o simplemente provocaban que el emulador se colgase.

Herramientas de prueba

Probé algunas otras herramientas para comparar el rendimiento gráfico del sistema, como el cubo giratorio y VideoDOS, que a veces tenían resultados muy extraños. Debido a que las pruebas gráficas son solo pruebas de rendimiento y están relacionadas directamente con la capacidad de respuesta del juego, también lleve a cabo algunas pruebas prácticas con algunos de mis juegos favoritos.

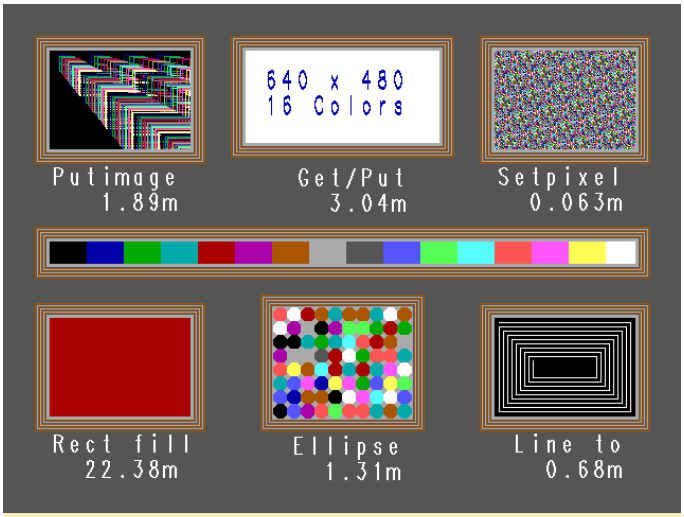


Figura 9 – Prueba gráfica sobre la compilación de Debian de DOSBox

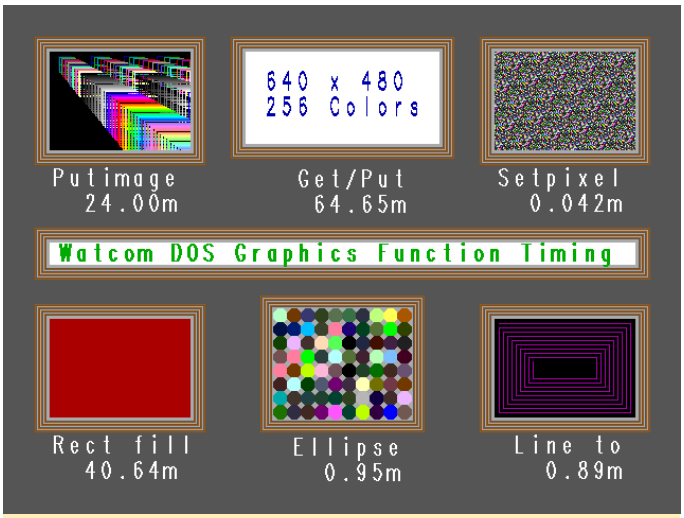
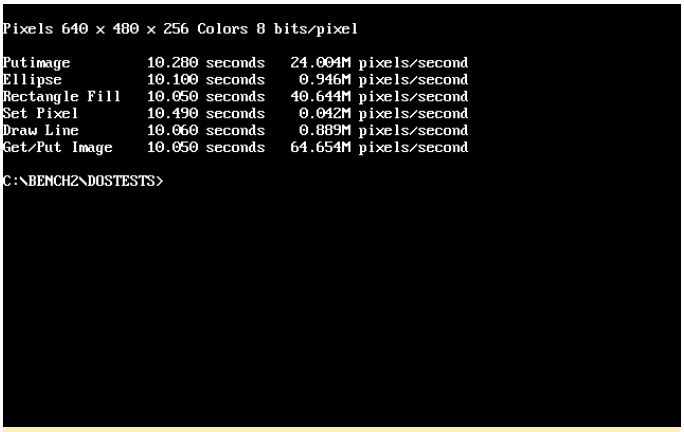
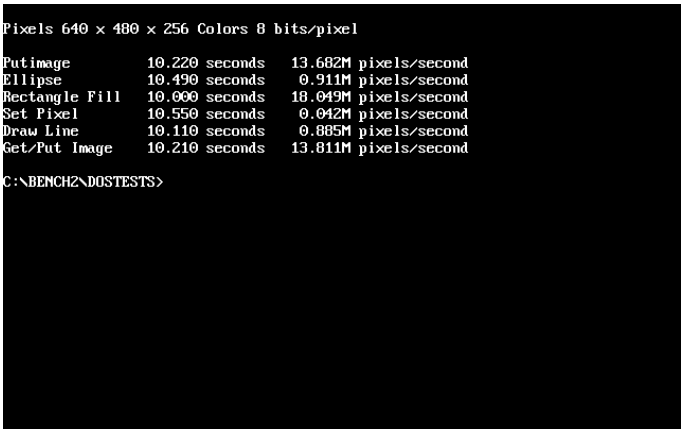


Figura 10 – Prueba gráfica en la compilación ARM de DOSBox. Estas pruebas gráficas daban resultados extraños: algunas pruebas parecían ejecutarse más rápido con más colores y en resoluciones más altas, mientras que otras parecían más normales.





Figuras 11 y 12 – Resultados de VideoDOS de la versión optimizada ARM (arriba) y la versión Debian estándar (abajo)

Juegos

El pack de pruebas de rendimiento incluía dos juegos, Doom y Quake, ya que a ambos se jugaban muy a menudo durante la edad de oro del DOS, y ofrecen algunos indicadores de rendimiento muy interesante

en el modo demo. Sin embargo, la prueba de rendimiento sobre el Doom no llegaba a funcionar correctamente, indicaba casi siempre que se estaba ejecutando a máxima velocidad, aunque realmente estaba muy lejos de ser así.

En lugar de usar las pruebas de rendimiento ya incluidas, llevé a cabo mis propias pruebas y comparé el tiempo que tardaron los juegos en ejecutar una demo completa. Los resultados fueron muy sorprendentes: la Demo 3, ejecutada en la versión optimizada ARMv7 de DOSBox, tardó aproximadamente 108 segundos en completarse. En la versión Debian estándar de DOSBox, hizo falta llegar a los 156 segundos para que se completase la prueba. Estamos hablando de un incremento en la velocidad de casi el 45% para la versión ARM.

Puedes ver claramente aún más la diferencia cuando juegas al Quake. La Demo 3 tardo 147 segundos en la versión optimizada y 248 segundos en la versión de Debian estándar, ¡aproximadamente un 70% fue más rápida la versión optimizada! Tras llevar a cabo todas las pruebas de rendimiento, quería ver qué tal se comportaba el emulador en una experiencia de juego real, pronto descubrí que la configuración que había elegido originalmente no funcionaba bien para ningún otro juego, así que cambié la configuración nuevamente y ejecuté un par de juegos de prueba. Después de reajustar los ciclos a 6.000 en lugar de 32.000, el Dune 2 se ejecutaba perfectamente, con una experiencia de juego agradable y suave. Los sonidos, la música y las voces eran aceptables y no percibí ningún problema.

También probé un par de juegos más exigentes, como Prisoner of Ice, que es un juego de aventuras muy bueno con algunas escenas de película y una opción para ejecutarlo con una resolución de 320×240 o de 640×480. La última incluso ofrecía algunas otras características, como fuentes mejoradas. Ambas versiones funcionaban bien en DOSBox. También me encontré con el mismo rendimiento al ejecutar Space Quest 6.

Resultados

La versión optimizada ARMv7 se ejecuta significativamente mejor que la versión de Debian estándar de DOSBox. Si tuviera que cuantificarla con un número, diría que la versión optimizada es, como término medio, entre un 10 y 15% más rápida que la versión del repositorio de Debian. A veces, incluso mucho más rápido que eso, como cuando ejecutamos el Quake.

Los resultados más rápidos parecen estar relacionados con algunas optimizaciones matemáticas dentro del propio emulador, que también pueden crear problemas como efectos secundarios, especialmente con operaciones de memoria. Esto, a su vez, puede causar problemas técnicos en algunos juegos o impedir que se ejecuten correctamente. A parte de esto, la versión optimizada de ARM es la mejor versión en términos de velocidad.

En base a mis anteriores pruebas, puedo decir que es incluso lo suficientemente rápido para poder manera Windows 3.11 o incluso Windows 95. La mayoría de los juegos deberían funcionar en ambos emuladores, aunque se ejecutan un poco mejor en la versión optimizada de ARM.

Configuración adicional

Cuando terminé con las pruebas con algunos juegos, decidí cambiar mi configuración con los siguientes parámetros, con lo que descubrí que funcionaban muchos juegos:

```
core=auto or dynamic
cputype=auto
cycles=fixed 3000
memsize=31
```

También descubrí que DOSBox es capaz de usar glshim junto con su interprete opengl usando la opción output:

```
output=opengl
```

Por último, modifiqué los parámetros sdl:

```
fullscreen=true
fulldouble=true
fullresolution=1280x720
windowresolution=original
output=opengl
```

Estas opciones inician el juego en modo pantalla completa y cuando se usan junto con LD_LIBRARY_PATH=/usr/local/lib/, puedes ejecutar el emulador con soporte OpenGL.

Otros juegos

Como puede ver en el cuadro que aparece a continuación, los juegos varían mucho en cuanto a rendimiento, y no existe una configuración única que funcione con todos los juegos. También me di cuenta que el modo "auto" en los ciclos no funciona realmente bien. La velocidad al 100% que se usa en juegos pesados suele ser peor que utilizar un valor fijo de ciclos.

Cuando utilices DOSBox para ejecutar tus juegos, sugiero empezar con un valor de ciclos de 3,000 y seguir avanzando hasta que el juego comience a ralentizarse, luego retroceder unos cuantos valores. Con esto deberías alcanzar el rendimiento óptimo en tus juegos favoritos de DOS.

Juegos	**Ciclos**	**Infos**	**Comentarios**
Sid Meier's Colonization	1,500-3,000	El juego funciona mejor con pocos ciclos. A parte de esto, se ejecuta bastante bien sin problemas o caídas de sonido. Sin embargo, la intro al iniciar el primer juego tarda mucho tiempo en ejecutarse.	
Shadow Warrior	15,000-20,000	El juego es lento y no se puede jugar.	
Terry Pratchett's Discworld 1		bien sin ningún problema	
Syndicate	6,000-10,000	El juego funciona bien sin problemas	No se ejecuta con glshim
Wing Commander I	2,000-4,000	El juego funciona bien sin ningún problema. En mi opinión, la versión de Amiga tiene una banda sonora mucho mejor.	Deberías usar un escalador 3x
Prisoner of Ice (640×480)	2,000-8,000	El juego funciona bien, con solo un pequeño problema en el sonido de vez en cuando.	
Space Quest 6	~12,000	El juego se ejecuta en su mayor parte a toda velocidad, aunque presenta una leve ralentización en la música y el texto se desplaza demasiado rápido.	
Dune 2	3,000	El juego parece un poco lento, aunque por lo general es aceptable y	

		no tiene problemas	
XCom Series	1,000-15,000	Funciona bien con leves problemas de velocidad	
Dark	~20,000	Funciona	
Legions		bien con leves problemas de velocidad	

Campamento de Programación – Parte 9: Montar una Estación Meteorológica Portátil

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

Vamos a aprender a cómo acceder a diversos datos meteorológicos y cómo compartirlos con tus dispositivos móviles a través de la conexión WiFi. Ten en cuenta que es necesario disponer de una Weather Board 2 (**https://wiki.odroid.com/odroid_go/arduino/30_weather_station**).

Antes de empezar, debes saber que este Campamento de programación utilizará un navegador web para mostrar la información meteorológica. Para que las cosas salgan bien, asegúrate de cumplir con los siguientes puntos. Es muy importante haber seguido la **Guía de Configuración d Arduino**

Figura 1 – Puedes tener una estación meteorológica portátil en tu mano

Requisitos

Asegúrate de tener estos dispositivos:

- ODROID-GO
- **Weather board 2**

- Un cable MicroUSB

Configura el entorno de desarrollo para Arduino en tu sistema. Antes de continuar con esta guía, conecta Weather board 2 a ODROID-GO y éste a su vez al PC mediante el cable microUSB.

Configurar SPIFFS

SPIFFS significa SPI Flash File System. Puedes visitar https://github.com/me-no-dev/arduino-esp32fs-plugin para ver la documentación completa sobre SPIFFS. ODROID-GO tiene una memoria flash pequeña (pero suficiente) en la que puedes cargar datos utilizando esta herramienta. Descarga un archivo comprimido desde **este enlace (ESP32FS-v0.1)**, luego extrae el directorio ESP32FS a uno de los siguientes directorios, dependiendo de tu sistema operativo:

- Windows: %USERPROFILE%DocumentsArduino ools
- Linux: ~/Arduino/tools

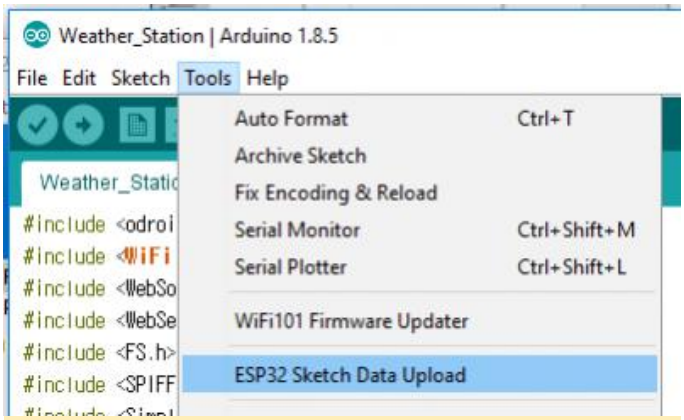


Figura 2 – Directorio de herramientas de Arduino

Cree el directorio tools antes de extraerlo si no existe. Abre Arduino IDE y podrás ver el menú Tools → ESP32 Sketch Data Upload

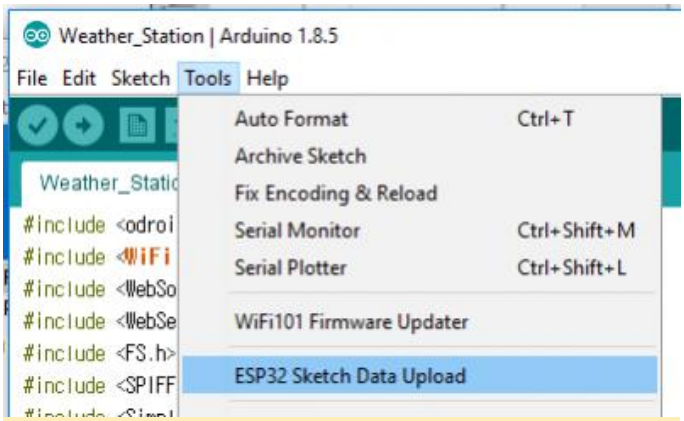


Figura 3 – Cargar el esquema

Importar la muestra en el IDE.

Haga clic en el menú Files → Examples → ODROID-GO → Applications → Weather_Station para importar el ejemplo de la estación meteorológica.

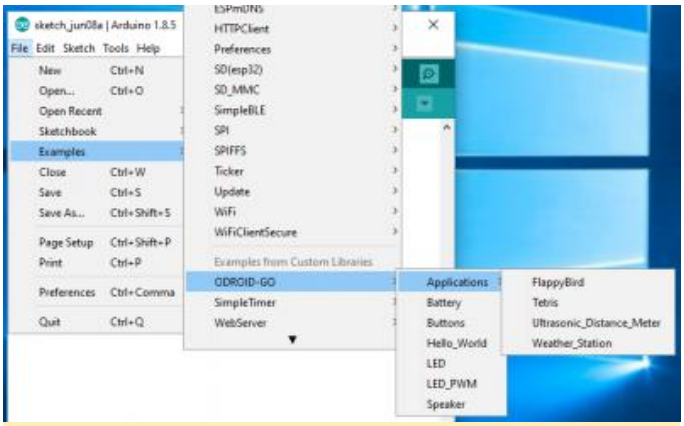


Figura 4 – Selección de la aplicación Arduino

A continuación, verás una nueva ventana con el código de ejemplo.

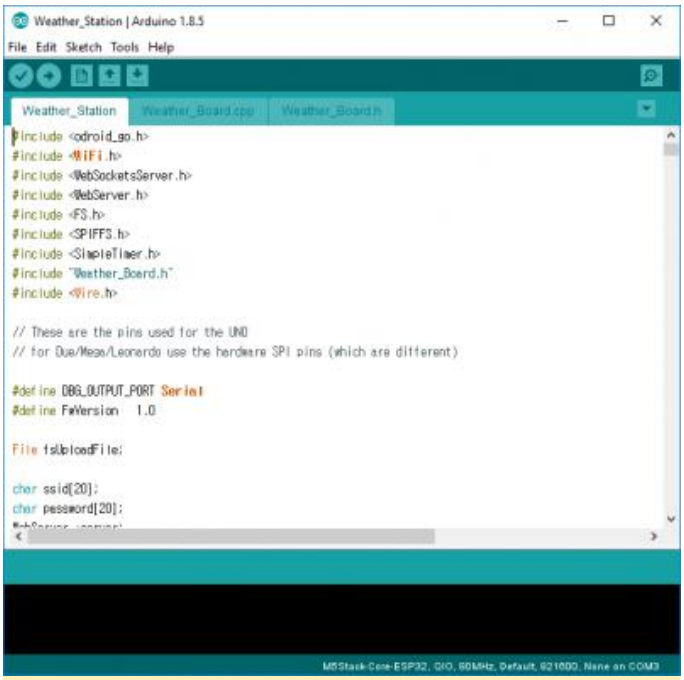


Figura 5 – Código meteorológico

Compilar y cargar el binario

Esta guía da por hecho que el número de puerto es COM3. Puede que sea diferente al tuyo. Verifica y compila el esquema haciendo clic en el menú Sketch → Verify/Compile, o presiona CTRL-R.

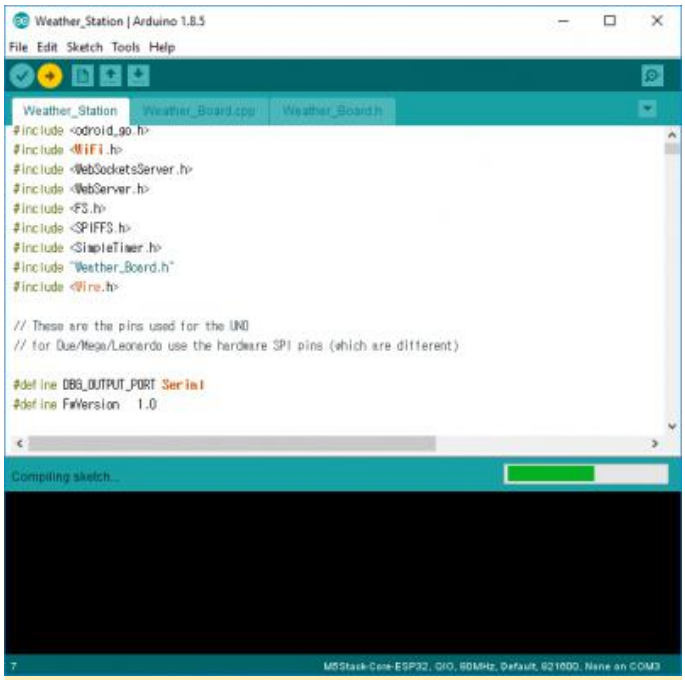


Figura 6 – Compilado el esquema

Si la compilación se completa sin ningún problema, carga el binario compilado haciendo clic en Sketch → Upload o presionando CTRL-U.

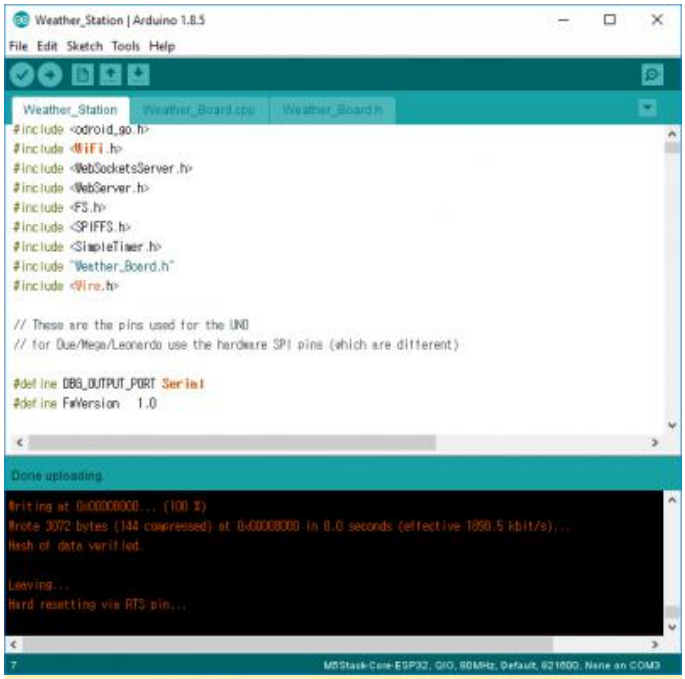


Figura 7 – Cargando el esquema

Puedes omitir el proceso de compilación ya que se realiza automáticamente cuando simplemente cargar sin realizar una compilación con anterioridad. Sabrás que la carga se ha completado cuando aparezca el mensaje: “Hard resetting via RTS pin…”

Cargar los datos

Este ejemplo tiene un programa de servidor web para servir las mediciones a través de una página web. Para ver esa página, debe cargar los datos de la página web en la memoria flash del ODROID-GO. SPIFFS te permite hacer esto. Si haces clic en el menú Tools → ESP8266 Sketch Data Upload, la utilidad SPIFFS buscará el directorio de datos en la librería actual y lo enviará.

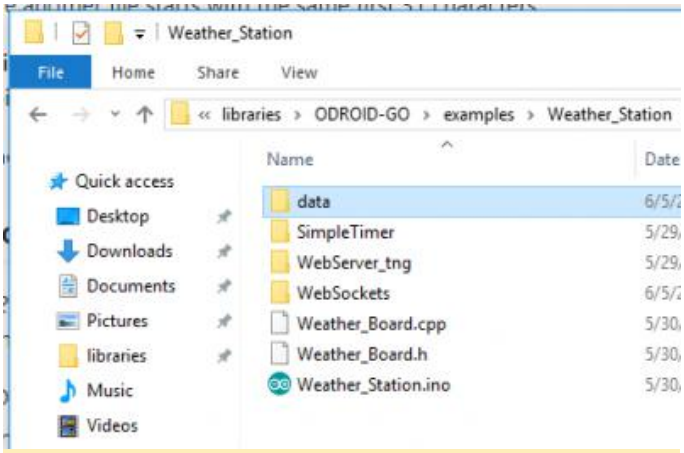


Figura 8 – carpeta de datos

Haz clic en el menú para cargar

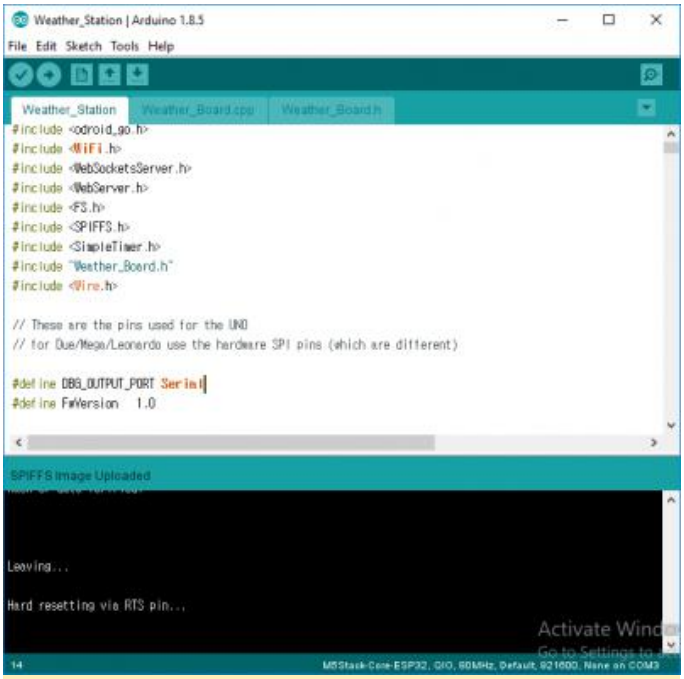


Figura 9 – Cargando la imagen SPIFFS

Sabrás que la carga se ha completado cuando se muestre el mensaje “Hard resetting via RTS pin…”.

Pruebas

Una vez finalizada la carga, el ODROID-GO se reinicia automáticamente. La pantalla que muestra cada medición de datos aparecerá en el ODROID-GO y tras unos segundos, se encenderá el LED azul en el centro de la placa.

Visitar con tu dispositivo – PC/móvil

El LED azul indica que el servidor web en la placa está listo para que puedas conectarte a él y leer los datos desde el punto de acceso del ODROID-GO. Busca el punto de acceso WiFi llamado ODROID_GO _ **** y conéctate (la contraseña por defecto es 12345678). Abre un navegador web y navega hasta 192.168.4.1. Esta dirección IP se configura por defecto. Verás la GUI web que muestra cada medición, y el LED azul empezará a parpadear una vez que empiece la comunicación. Usando la página web de Configuración de la red, puedes ajustar la configuración del Wifi, como SSID, dirección IP y contraseña.

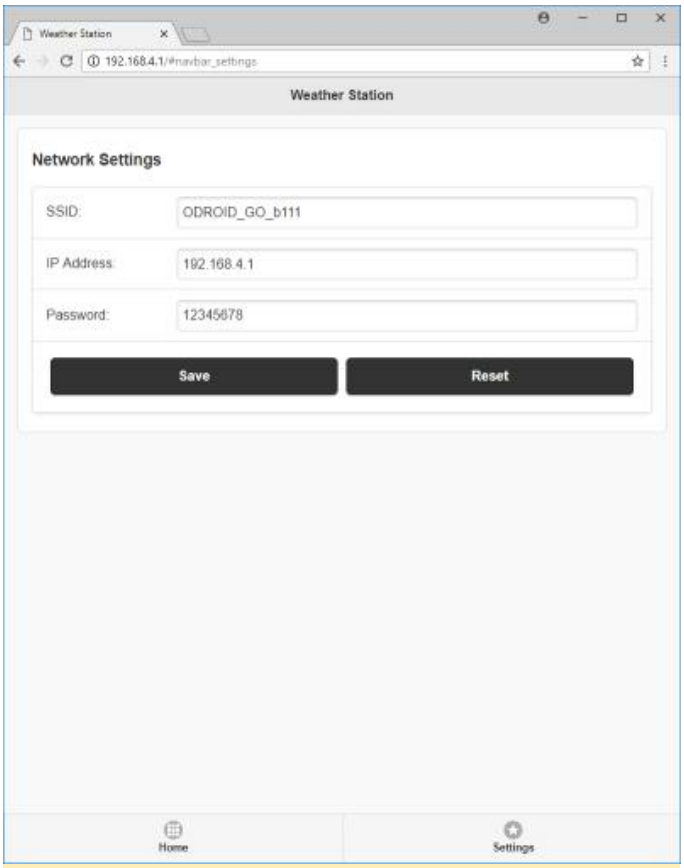


Figura 11 – Página web de configuración de la red

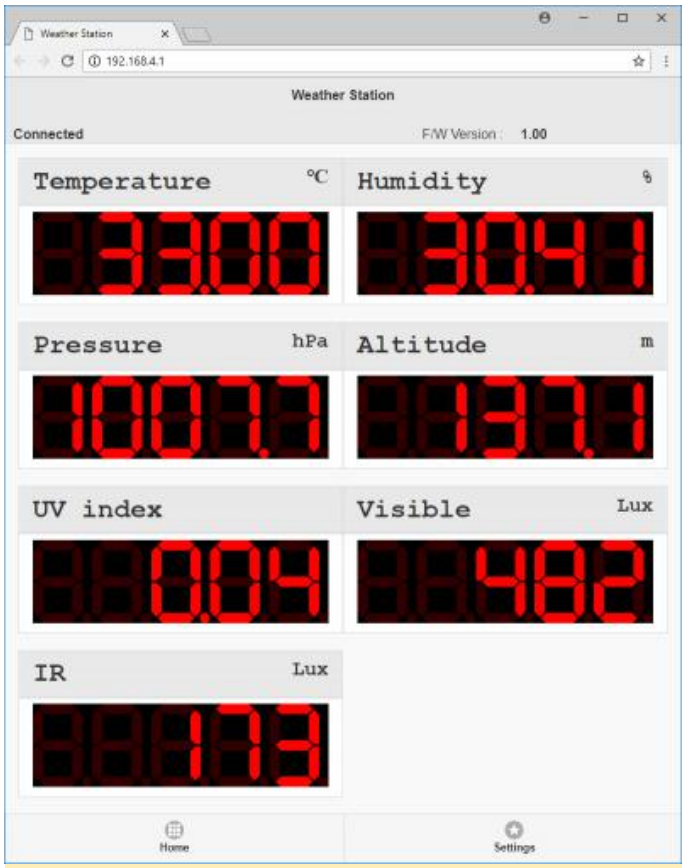


Figura 10 – Página web de la estación meteorológica

El Proyecto Tricorder ODROID-GO

December 1, 2018 By Volker Raum ODROID-GO, Mecaniqueo

Figura 1 – Tricorder

Para aquellos que no saben qué es un Tricorder, permitirme que lo explique: En la nueva serie de Star Trek, los personajes a menudo llevan un dispositivo móvil que se usa para medir desgarros en el continuo espacio-tiempo y decir “Está muerto, Jim.”

Figurea2 – Tricorder del Star Trek

Aunque gano dinero trabajando con software, siempre he tenido una cierta afinidad por el hardware: los chips cableados y otros componentes digitales integrados. Empecé con circuitos basados en

chips lógicos TTL y CMOS estándar. Un día, descubrí la serie programable ATMEL AVR. Mi viaje continuó y descubrí la chulada del ESP8266 de Espressif. Después de un tiempo, busque el ESP32 (WROOM32) para mis proyectos.

Ya había hecho proyectos con sensores (es decir, BME280) utilizando el bus I2C. También he tenido proyectos que usaban pantallas y botones para la visualización y el control. Lo que nunca he tenido es una buena combinación de todo esto, reunido en un único dispositivo con una batería.

Luego Hardkernel apareció con su ODROID-GO, la combinación perfecta para mí. Me ha estado rondando durante mucho tiempo la idea de un dispositivo multisensorial. Ahora era el momento de embarcarse en el proyecto.

Conceptos básicos

El ODROID-GO tiene un cabezal de expansión con 10 pines. Tres de ellos son de alimentación, uno no está conectado y otros son para SPI (lo cual podría interferir con el rendimiento de la pantalla). Al menos dos de los pines, GIO15 y GPIO4, no han sido utilizados por ninguna otra cosa. Solo lo suficiente para I2C. Con el ESP32, I2C se puede asignar a casi cualquier pin IO. No existe mapeo estático. ¡I2C era la clave!

Ahora tenía que averiguar qué sensores funcionan con I2C. Hay muchos de ellos. Después de echarle un rato a google terminé eligiendo estos:

- BNO055 para la orientación (balanceo, inclinación, giro) y la aceleración.
- BME280 para la presión, temperatura y humedad del ambiente.
- VL53L0X para medir la distancia (0-120cm).
- VEML6040 para medir la LUZ (RGB, LUX).
- VEML6075 para medir el UV (UVA, UVB => Índice UV).
- CCS811 para medir las concentraciones de CO2 y gas COV.
- Mics6417 para medir ocho gases más y su concentración.
- MLX90416 para medir la temperatura (IR) de objetos de -70 a 300 grados Celsius (como los termómetros sin contacto).

Comprar los sensores

¿Dónde comprar los sensores? Decidí recurrir a eBay y a sus innumerables vendedores chinos.

Los sensores tardan aproximadamente un mes en llegar, pero tienden a tener los precios más bajos. Por supuesto, compre varios de cada tipo por si acaso quemaba alguno de ellos. No quería esperar otro mes a que me llegaran los reemplazos. Por lo general, el coste ronda los 3-6 € por sensor. La excepción estaba en los micrófonos, unos 50 €.

Desarrollo de prototipos

Una vez que los sensores me llegaron de China, los revisé conectándolos al ODROID-GO a través de una placa de pruebas. Escribí un pequeño programa de prueba para producir una salida en serie y mostrar los valores en la pantalla del ODROID.

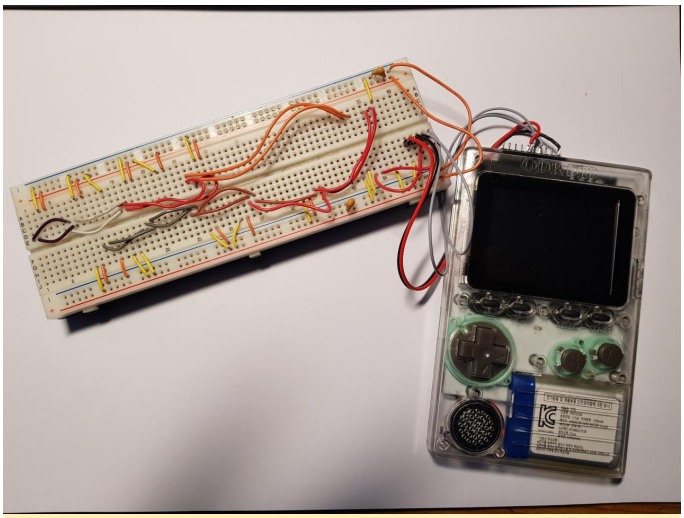


Figura 3 – Placa de pruebas

También decidí crear algunas placas para hacer experimentos con el fin tener una mejor idea de cómo sería el resultado final.

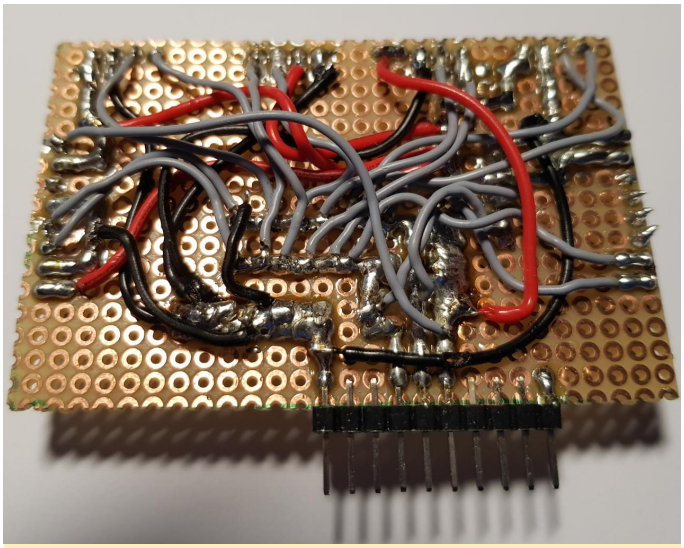


Figura 4 – Placa experimental 1

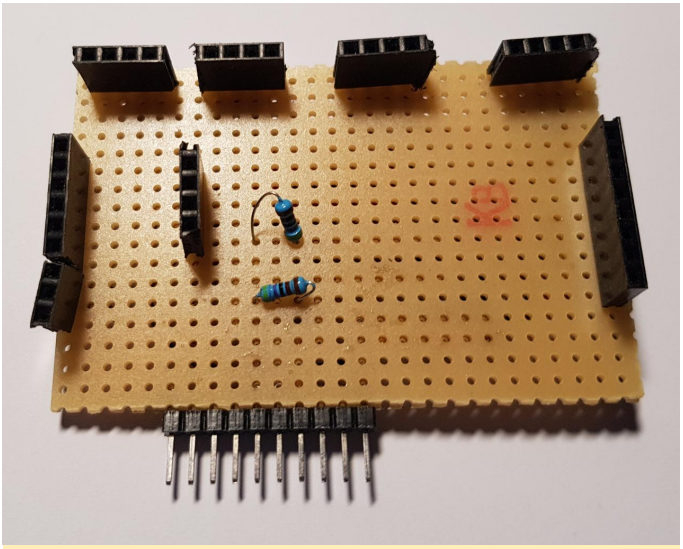


Figura 5 – Placa experimental 2

Esquemas (cableado general)

Por lo general, cablear los sensores I2C es bastante simple. I2C es una interfaz tipo bus. Simplemente conecta todos los pines SCL y SCK juntos y conecta estas 2 conexiones a VCC a través de resistencias (Pullup).

Además, proporciona energía (VCC, GND). El voltaje estándar para los sensores es 3.3V. Afortunadamente, el ODROID-GO ofrece 5V y 3.3V. Cada dispositivo en un bus I2C tiene su propio ID. De esta forma, el controlador (maestro) puede atender a cada sensor (esclavos) en el bus. Además de los cuatro pines mencionados, algunos sensores tienen pines adicionales que influyen en su comportamiento o les permiten responder a diferentes direcciones I2C.

En mi caso, no fue tan sencillo conectar todos los sensores ya que los sensores VEML usan la misma dirección I2C de 0x10. En lugar de usar una complicada lógica para evitarlo, decidí usar un Switch I2C (TCA9543a). El que elegí tiene tres puertos I2C. Un puerto se comunica con el ESP32 y es el puerto de "entrada", mientras que los otros dos están conectados a los sensores. Los VEMLs se colocan en diferentes puertos. El switch puede programarse para pasar la comunicación I2C al puerto 1 o al 2.

Desde la perspectiva de la programación, debes indicar al switch que active el puerto 1 o el puerto 2 y luego consultar los sensores conectados a ese puerto. Cambia al otro puerto y consulta el resto de sensores. Realmente no es gran cosa.

Al final se me ocurrió este esquema:

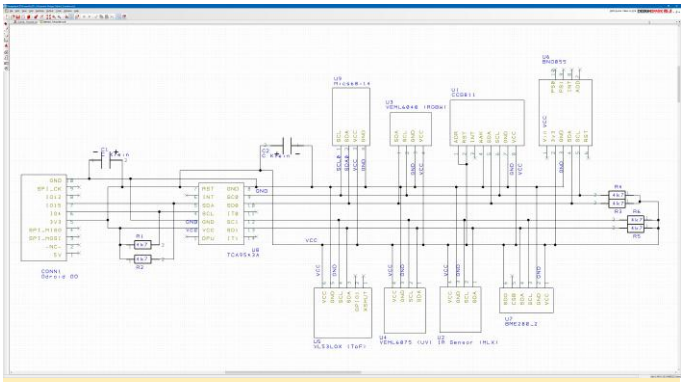


Figura 6 – Esquema

Utilicé DesignSparkPCB para todos mis esquemas y más tarde para transformar los esquemas en un diseño PCB. Es gratis y lo recomiendo encarecidamente. Se te pedirá que te registres, aunque esto parece ser algo obligatorio para la mayoría de las cosas hoy en día y por el valor que obtiene a cambio, el registro vale la pena.

Un factor importante del software de la PCB es la capacidad de ésta para aceptar tus propios componentes. Intenté encontrar las librerías de los componentes para los sensores que utilicé, pero no logré encontrar ninguna que fuera gratuita. Me di por vencido y simplemente las diseñé, lo cual es posible con este programa.

Placa de circuito impreso (PCB)

Partiendo de los esquemas pude crear una PCB en DesignSparkPCB. Debido al exceso de cableado, no

creo que una PCB de una única capa sea posible. Parece ser necesario una PCB de dos caras.

Muchas herramientas de PCB ofrecen el cableado automático. Con bastante frecuencia esto produce conexiones extrañas, así que opté por canalizar los cables yo mismo. DesignsparkPCB, como todas las herramientas, ayuda a evitar que te olvides de los cables.

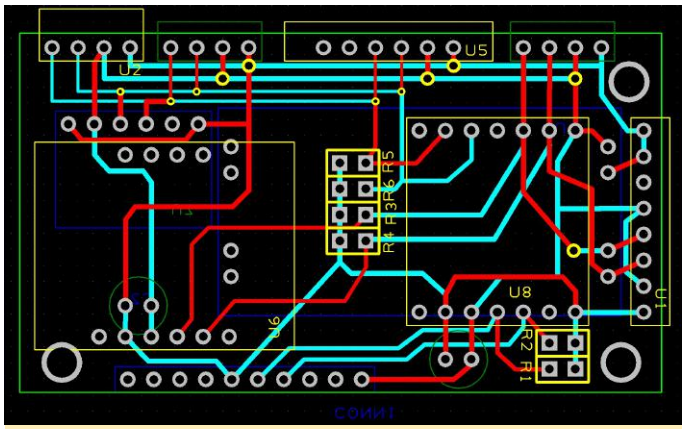


Figura 7 – Diseño de PCB

El rojo y cian indican el cableado en la capa superior e inferior. El amarillo y el azul tienen fines de documentación y se pueden imprimir en la PCB. De esta forma, podrás ver con facilidad qué dispositivo se colocarán y se soldarán y en qué posición.

Una vez completado el diseño de PCB, ahora se debe fabricar. Algunas personas hacen esto por su cuenta. En el pasado yo también lo hice por mi cuenta, pero nunca me gustó la química que hay detrás de ello. Además, hay 2 capas involucradas, lo cual requiere una alineación muy precisa de las mismas. Dudo que pueda hacerlo por mi cuenta. Decidí entonces recurrir a una empresa profesional que lo hizo por unos 25 €.

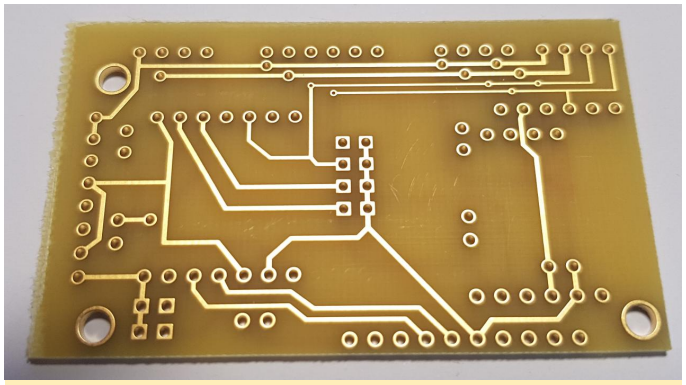


Figura 8 – PCB 1

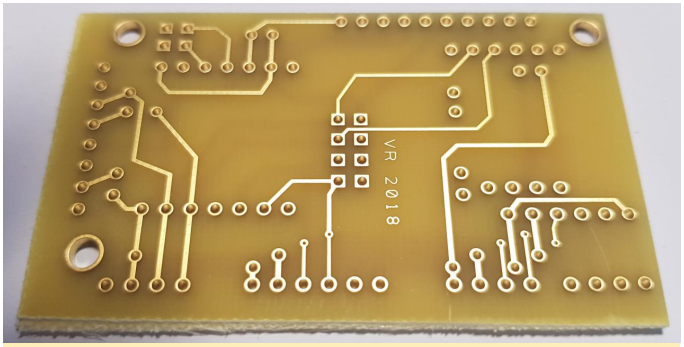


Figura 9 – PCB 2

Para reducir los costes de producción, decidí no usar material de soldadura e imprimir la documentación.

Tiempo de montaje

Finalmente, teniendo la PCB en la mano, el montaje de los sensores recurriendo a la soldadura fue sencillo.

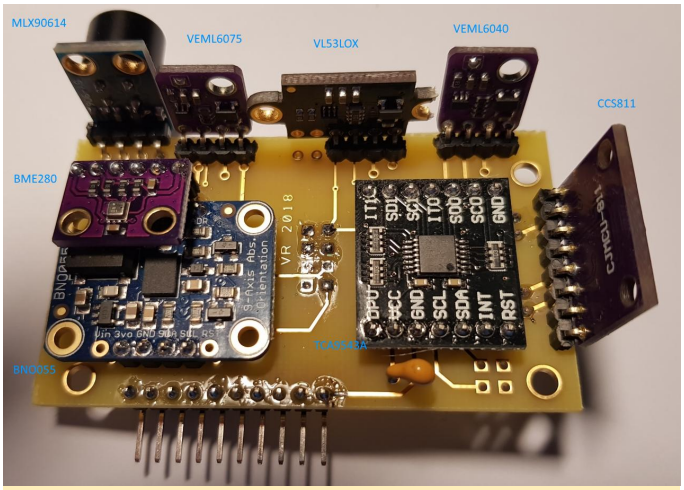


Figura 10 – Asamblaje

Haciendo mis primeras pruebas con el software para ver si ESP32 podía encontrar todos los sensores, me di cuenta que cometí un error en el diseño. Tenía la intención de tener el BME280 en la parte inferior de la PCB. Al tenerlo soldado en el lugar equivocado, VCC y GND fueron conectados a los pines incorrectos. El BME dejó de fumar. A veces las cosas salen mal, 4€ tirados a la basura. Por eso se suele pedir más de uno. Por suerte, este error no afectó a los otros sensores o al ODROID-GO. El sensor de gas Mics cuesta unos 50€ y solo dispongo de uno. Quemar éste te dolería bastante más. El problema fue resuelto fácilmente. El siguiente BME280 simplemente lo soldé sobre el sensor de orientación BNO055.

Software (el diseño de la interfaz de usuario)

No soy diseñador. Hacer las cosas bonitas no es lo mío. Normalmente trabajo más bien a un nivel funcional. En este caso en concreto, quería demostrarme a mí mismo que estaba equivocado y deseaba hacer una buena interfaz de usuario para el Tricorder.

Desde un punto de vista funcional, estaba bastante claro que no era técnicamente posible realizar un repintado completo de la pantalla (320×240 píxeles) simplemente para actualizar algunos valores de medición, probablemente varias veces por segundo. La pantalla no es lo suficientemente rápida para ello. El resultado sería un continuo parpadeo. Sin embargo, estaría bien incluir algunos elementos gráficos en la interfaz de usuario. Esto me llevó a diseñar completas pantallas con imágenes de fondo JPG y áreas gráficas vacías que sirvieran como marcadores de posición. En el caso donde la pantalla fuera trazada por primera vez, se dibujaría la imagen completa de fondo y luego se pegarían los valores del sensor. Para el resto de valores posteriores, solo se deberían volver a trazar las áreas de los marcadores de posición. Esto mejora notablemente la capacidad de respuesta de la interfaz de usuario.

También era obvio que existían demasiados valores de medición para ponerlos todos en una única pantalla al mismo tiempo. Introduje pantallas que podrían alternarse usando los botones “A” y “B”.

Lo más difícil fue crear un estilo particular para las imágenes de la pantalla. Hice varios intentos y ninguno me convencía. Entonces tuve una idea. Es un Tricorder de Star Trek, así que lo haría al estilo del Star Trek. Busqué en Google Images del tricorder de Star Trek y aparecieron toneladas de imágenes interesantes. Fue a partir de entonces cuando el estilo del diseño me quedó claro.

Probé con varios programas de dibujo disponibles de forma gratuita, pero terminé con paint.net, que suelo usar para la mayoría de mis proyectos gráficas.

Software (la lógica de control)

Puedes programar el ESP32 usando el SDK de Espressif (ESP-IDF) directamente en C ++, pero la funcionalidad que proporciona parece estar a un nivel muy básico. Arduino es un ecosistema muy popular que proporciona algo de abstracción desde el nivel más bajo mientras siga siendo C/C++. Otra ventaja a la hora de usar Arduino es el hecho de que existe una gran cantidad de librerías para todos los sensores, lo cual facilita su uso.

Para Arduino puedes usar el Arduino IDE, con el que empecé hace unos años. Fue entonces cuando me topé con PlatformIO. Viene como un plugin para el editor ATOM o VS-CODE. Usé ambos y descubrí que me gustaba más VS-CODE.

Entrar en detalles sobre cómo programar para ESP32 en el mundo de Arduino con VS-CODE puede hacer que este artículo se extienda demasiado. ODROID Magazine ya cuenta con artículos sobre programación con ODROID-GO. Puede ser más interesante centrarnos en las cosas que normalmente no suele hacerse en una implementación de ESP32.

Nunca he tenido que lidiar con archivos binarios en mi código para el ESP32. La forma de hacerlo para ODROID-GO (y ESP32 en general) es con SPIFF. SPIFF es un sistema de archivos para ESP32. Puedes cargar archivos binarios en un área especial de la memoria flash del ESP32 (la partición SPIFF).

Para el ODROID-GO, existe una colección de librerías que puedes utilizar para hacer frente al hardware del GO (altavoz, botones, pantalla, incluso sensores potenciales). La librería de la pantalla admite la visualización de archivos JPG almacenados en la partición SPIFF

Lo que tienes que hacer es cargar los JPG en el ESP usando PlatformIO. Después de hacer esto, puedes programar el ESP y acceder a los archivos cargados en tu código proporcionando los nombres de archivo. Esto hace que sea relativamente fácil rellenar la pantalla de GO con un archivo JPG; se necesita una única línea de código.

Otra cosa que tenía que hacer era modificar algunas librerías de sensores. Estas librerías a menudo se escriben para trabajar con más de un controlador, gracias a la abstracción de Arduino. Esto conlleva algunos problemas si el ESP32 funciona de un modo

diferente en algunas áreas. Los cambios que tenía que hacer no eran demasiados.

El estado actual del software

Las siguientes imágenes muestran las diferentes pantallas que ya he diseñado. Como he mencionado anteriormente, pueden cambiar de una pantalla a otra con el botón A o B.

Fase de inicialización

Figura 11 – Pantalla de inicio

Los sensores necesitan ser activados. Un buen momento para la pantalla de bienvenida. Probablemente debería haber limpiado la pantalla antes de haber tomado la foto.

Sensores ambientales

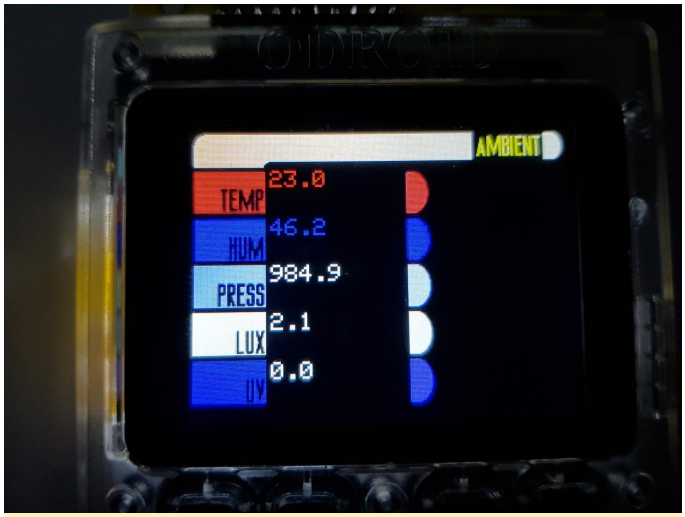


Figura 12 – Pantalla ambiental

La pantalla ambiental contiene los valores del sensor de BME, VEML6040 y VEML6075. Estos son la temperatura (en grados centígrados), la humedad (en%), la presión (en hPa), la intensidad de la luz (en LUX) y el índice UV (es un número que indica cuanto de peligroso es el nivel actual de UV)

Sensor de luz (RGB)

Figura 13 – Pantalla de luz

La vista de luz proporciona información detallada sobre la distribución del color rojo, verde y azul en la luz visible. Las longitudes de onda exactas medidas por el sensor se pueden verificar en las especificaciones técnicas de VEML.

La distribución se presenta en un gráfico de barras con los coleres rojo, verde y azul.

La intensidad de la luz en LUX ya estaba presente en la pantalla ambiental y se puede ver aquí nuevamente.

La temperatura del color indica si la luz es “cálida” o “fría”. Los valores más altos indican luz fría (azul), los más bajos indican luz cálida (rojo)

La vista de los gases

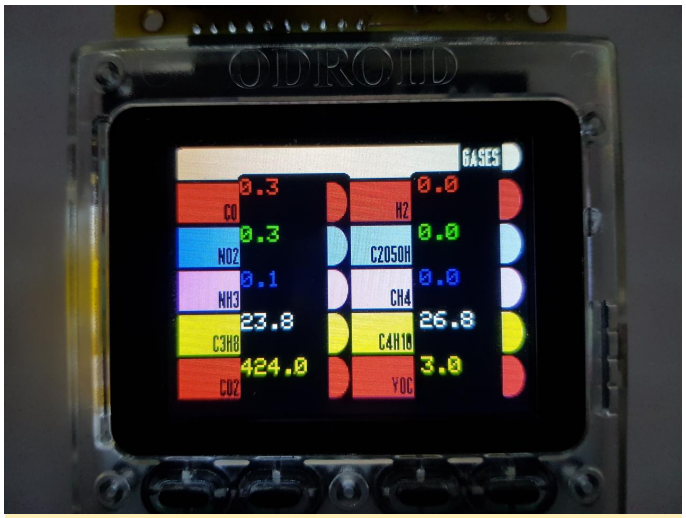


Figura 14 – Pantalla de gases

La vista de gases presenta mediciones del sensor Mics y de CCS811. Diez gases en total.

La unidad de las medidas es partes por millón (ppm). Los gases aparecen por sus fórmulas químicas, ya que los nombres no caben en la pantalla.

- CO – monóxido de carbono: puede causar asfixia con facilidad. Difícil de recuperar de la exposición.
- H2 – Hidrógeno: Junto con el oxígeno, tienes una alta probabilidad de que tu casa salte por los aires. También se utiliza en los propulsores de los cohetes.
- NO2 – Dióxido de nitrógeno: Tóxico.
- C2O5OH – Etanol: Alcohol. Me gustan estos compuestos en diversos formatos
- NH3 – Amoniaco: Olor intenso. Uno fuente puede ser la mierda. No saludable por supuesto.
- CH4 – Metano: peor que el CO2 en términos de gases de efecto invernadero. Inflamable si O2 está presente.
- C3H8 – Propano: La gente lo usa para disparar cosas.
- C4H10 – Butano: También inflamable. Gas de camping.
- CO2 – Dióxido de carbono: esto es lo que expiramos y lo que las plantas necesitan para vivir. Causa problemas como el gas de efecto invernadero.
- COV – Gases compuestos volátiles: sustancias en forma gaseosa a temperatura ambiente. No se define con precisión qué gases se detectan. Es solo un indicador. Cuanto mayor sea el valor, peor es.

En la captura de pantalla que ves parece haber propano y butano alrededor del Tricorder. Esto se debe al hecho de que el sensor de gas Mics necesita algo de tiempo para que se calentase antes de proporcionar valores precisos. No quería esperar 10 minutos para tomar la foto.

Distancia

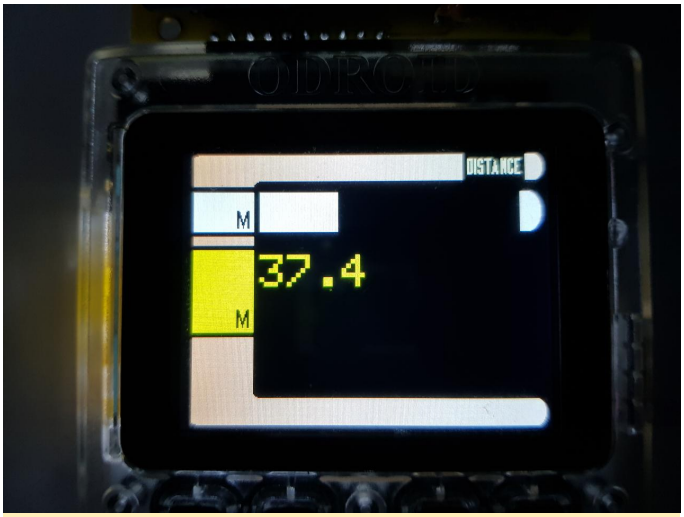


Figura 15 – Pantalla de distancia

La distancia entre el Tricorder y un objeto al que se apunte puede variar entre 0 y 120 cm. Se muestra en forma de un gráfico de barras con un número (la barra blanca).

Temperatura

Figura 16 – Pantalla de temperatura

La temperatura de un objeto se muestra de la misma forma que la distancia. Se proporciona con un número y un gráfico de barras (barra roja). La unidad es Celsius

Orientación

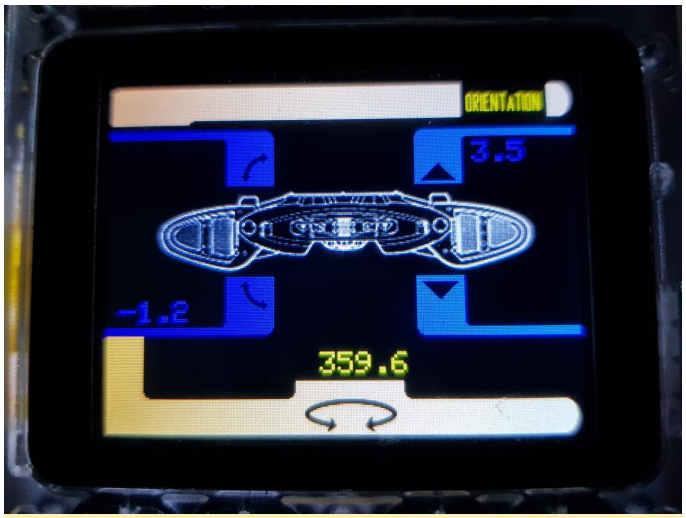


Figura 17 – Pantalla de orientación

La orientación fue lo más difícil para diseñar una página. Los simples números no proporcionan un significado intuitivo. La solución actual posiblemente no sea la mejor. Espero poder encontrar algunas ideas mejores.

A la izquierda, se muestra un rollo. Si el Tricorder se gira en el sentido contrario a las agujas del reloj, el número azul a la izquierda se muestra en la posición inferior (como en la captura de pantalla). Si se gira en el sentido de las agujas del reloj, el número se mostrará en la posición superior, tal y como un ángulo en grados (0-180).

Para la inclinación, es lo mismo. Si el Tricorder apunta hacia arriba, el número que indica el ángulo de rotación se muestra en la posición superior (como en la captura de pantalla). Si apunta hacia abajo, el número se muestra en la posición inferior.

La dirección (amarillo) sólo es el ángulo al que apunta el Tricorder (norte, sur, este, oeste), de 0 a 359.9 grados. Esto puede ser usado como un compás.

Calibración

Los sensores envían valores de medición. ¿Cómo de exactas son estas mediciones? Sin una referencia es complicado saberlo. Es fácil cotejar la medición de la distancia. La temperatura es algo más complicado. La intensidad de la luz también requiere más esfuerzo, por ejemplo, habría que buscar un dispositivo de medición que nos diera una referencia. Casi imposible son de verificar las concentraciones de gas. Tengo la intención de investigar cómo puedo mejorar en este aspecto.

Implementar cosas como hold/min/max

Si conoces los multímetros, sabrás que estos dispositivos a menudo tienen la capacidad de rastrear valores mínimos, máximos o promedios. También es muy común congelar la pantalla para que muestre una medición durante más tiempo. El Tricorder también podría necesitar esto, lo cual requerirá un mayor uso de los botones del ODROID.

¿Exportar las mediciones de forma inalámbrica?

Actualmente las mediciones solo se muestran en la pantalla del ODROID. El ESP32 puede funcionar con WiFi y Bluetooth (BLE y SPP clásico). El Tricorder podría mejorarse para que envíe las mediciones “por el aire”. MQTT y BLE son los candidatos. Ambos son válidos para el ESP32.

Actualmente no le veo el uso a este proyecto. Un posible proyecto sería una “dispositivo de medición”. Sus mediciones podrían mostrarse utilizando tu dispositivo móvil. Sin embargo, para esto, no sería necesario el ODROID-GO.

Aceleración

Actualmente solo uso el sensor BNO055 para el balanceo, inclinación y dirección. Es capaz de hacer más, como medir la aceleración (fuerzas g).

Carcasa

Dispongo de una impresora 3D. Seguramente me planteé crear una carcasa para la electrónica del Tricorder que de alguna manera se ajuste a la carcasa del ODROID-GO.

Compilando un Emulador de Commodore 64

December 1, 2018 By AreaScout Juegos, ODROID-C2

Este emulador permite ejecutar juegos que están diseñados para el sistema Commodore 64 de 8 bits. Ha sido exportado al ordenador de placa reducida (SBC) ODROID-64.

Figura 01 – Emulador de Commodore 64

Los siguientes son los pasos necesarios para hacerte con el código fuente, aplicar los pertinentes parches y ejecutar el emulador. He contribuido con algunos parches a la base de código maestro. No ha sido nada fácil resolver los diversos problemas que han ido apareciendo, pero con la ayuda del personal encargado del proyecto, ahora lo tenemos disponible para todos.

Verificar del código fuente y aplicar un parche

En primer lugar, necesitamos conseguir algunos requisitos previos:

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

Luego, verifica el código fuente y aplica el parche que elimina los bordes para las máquinas Commodore VIC-II, si quieres. Esto eliminará los márgenes de los modelos de máquina C64 y C128; los juegos se ven mucho mejor sin ellos. Esta es la forma más rápida y chapucera de hacerlo, lo mejor sería añadirlo a la configuración de libretro. Si un juego se sale de estos márgenes, no funcionará y la aplicación se bloqueará. Sin embargo, son muchos los juegos que no sobrepasan 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
```

A continuación, puedes empezar a compilar la máquina Commodore que quieras. Los tipos de máquina válidos son los siguientes:

- x128
- x64
- x64sc
- x64dtv
- x64scpu
- xplus4
- xvic
- xcbm5x0
- xcbm2
- xpet

Necesitarás especificar la variable EMUTYPE seguida por el tipo de máquina, que refleja la compilación que has elegido. Si no se especifica, x64 (C64) quedara seleccionada por defecto.

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

Si deseas compilar más de un tipo de máquina, no olvide ejecutar clean (make EMUTYPE=x64 -f Makefile.libretro -j7 clean) sobre el proyecto, de lo contrario el núcleo no funcionará.

Aplicar la configuración de RetroArch

Para aplicar la configuración de RetroArch, copia el binario en la carpeta principal de RetroArch:

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

Inicia RetroArch, selecciona el núcleo vice, luego inicia el núcleo con o sin un juego. Presiona el botón Guide en tu mando de juegos o F1 en el teclado y desplázate hacia abajo hasta Opcions, selecciónalo y deshabilita DriveTrueEmulation-> OFF, y fija Controller0Type en “joystick”

También configuré la relación de aspecto en 16:10, creo que es un buen término medio entre 4:3 y 16:9:

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

Con el botón Start, activas la configuración de la GUI nuklear (tienes que presionar el botón select una vez para activar el ratón). 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 Xbox).

Referencías

http://vice-emu.sourceforge.net/
https://forum.odroid.com/viewtopic.php?f=98&t=32173#p233998
https://youtu.be/ItkppnXWd9U

Campamento de programación – Parte 10: Medir la distancia con ultrasonidos

December 1, 2018 By Justin Lee ODROID-C2, Mecaniqueo, Tutoriales

Vamos a aprender a usar la salida GPIO, la entrada IRQ y el reloj del sistema con un módulo de medición de distancias por ultrasonidos. Ten en cuenta que el sensor de distancia es un elemento adicional que es imprescindible. (**https://wiki.odroid.com/odroid_go/arduino/31_ultrasonic_distance_meter**).

Figura 1 – Puedes tener un medidor de distancia ultrasónico portátil

Requisitos

Asegúrate de contar con estos productos:

- ODROID-GO
- Módulo de alcance ultrasónico HC – SR04
- Cables Jumper y una placa de pruebas completa o de medio tamaño
- Una batería auxiliar si lo vas a transportar
- Este módulo necesita 5V de potencia de entrada, pero el ODROID-GO genera 3V para el pin de alimentación (# 6) en sus pines cabecera. Por lo tanto, es necesario utilizar el pin VBUS USB de 5 V (# 10) con una fuente de alimentación USB externa..
- Como alternativa, puedes usar un convertidor DC-DC progresico y un conmutador de nivel.
- Un cable microUSB

También deberás configurar el entorno de desarrollo para Arduino en tu sistema.

Configuración del hardware

Consulta la Figura 2 cuando configures tu hardware. Nosotros usamos los siguientes componentes:

- Sensor ultrasónico: HC-SR04
- Convertidor DC-DC 3V a 12V (configurado a 5V por ahora): MT3608
- Convertidor de nivel lógico: BSS138
- ODROID-GO: fritzing_odroid-go.fzpz
- Convertidor Buck DC-DC Mini-360: mini-360_dc-dc_buck_converter.fzpz
- HC-SR04: hc-sr04.fzpz

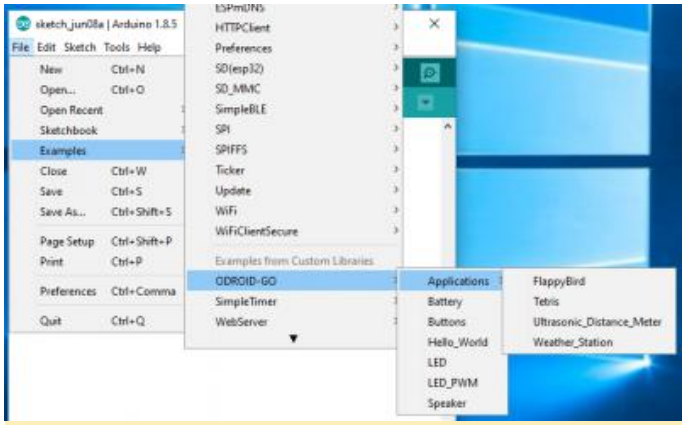


Figura 2 – Diagrama del cableado

A continuación, descarga el archivo de ejemplo Fritzing de odroid-go-ultrasonic-sensor.fzz.

Importar y compilar, cargar a ODROID-GO

Haz clic en el menú Files → Examples → ODROID-GO → Applications → Ultrasonic_Distance_Meter para importar y presiona CTRL-U para compilar/cargar.

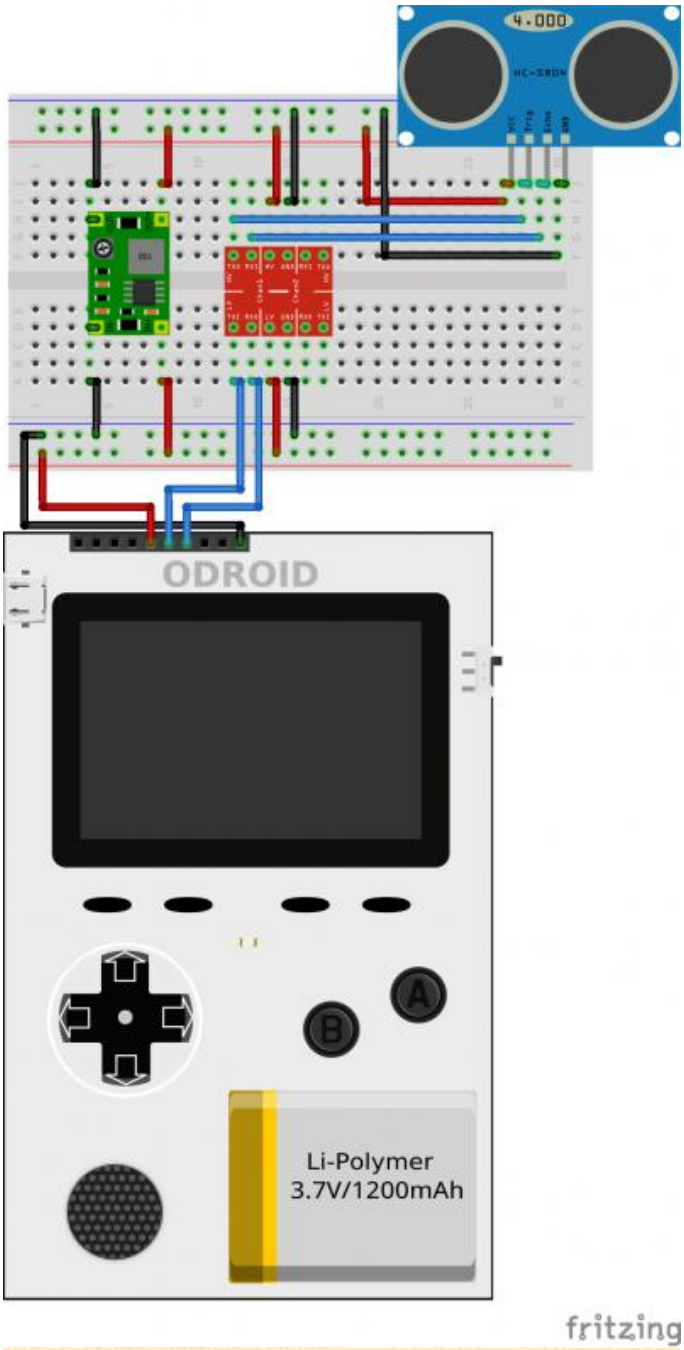


Figura 3 – Cargar la aplicación del medidor de distancia por ultrasonidos Arduino

La carga se habrá completado cuando veas el mensaje “Hard resetting via RTS pin...”.

Pruebas

Una vez finalizada la carga, ODROID-GO se reinicia automáticamente. La pantalla mostrará una distancia medida en pulgadas, unidades de cm cuando detecta un obstáculo delante del sensor ultrasónico. Si no se cumplen las condiciones de medición, como son una distancia demasiado larga o demasiado corta, el texto que se muestra en pantalla aparecerá en rojo. Si la medición es normal, el texto aparecerá en verde.

Introducción a NEMS Linux: Parte 3 – Configurando monitores de servicio en NEMS Linux

December 1, 2018 By Robbie Ferguson Linux, Tutoriales

Esta es la tercera parte de una serie en la que presentamos NEMS Linux: el servidor de monitorización empresarial Nagios para dispositivos ODROID. Si no has leído las dos primeras partes (números de octubre y noviembre de ODROID Magazine), empieza desde el principio ya las lecciones se van complementado con las siguientes. Mi intención con estos artículos siempre ha sido la de presentarte a NEMS Linux de forma que te proporcione conocimientos útiles y los pongan en práctica de inmediato. No tienen la intención de presentarse como simple documentación, sino más bien artículos técnicos que te proporcionen ideas sobre cómo puedes usar NEMS Linux en tu entorno. Sin embargo, este mes nos convertiremos en unos auténticos frikis juntos, ya que incluyo dos ejercicios clave que pueden resultarte útiles para monitorizar los activos de tu red con NEMS Linux.

En los ejercicios de este mes, aprenderás lo necesario para configurar NEMS Linux y poder realizar lo siguiente:

- Indicar si tu sitio web está activo y notificar si ha estado inactivo durante más de 10 minutos: monitorizar el tuyo propio, el de tus clientes o cualquier sitio web http/https para conocer el tiempo de actividad o el tiempo de respuesta con lentitud.
- Monitorizar el estado de un puerto TCP/UDP específico en un dispositivo conectado a la red y notificar si deja de responder: avisar si tu nodo local de blockchain ha dejado de responder en el puerto 8333, Apache2 dejó de responder en el puerto 443, o monitorizar el estado de openssh que se ejecuta en tu servidor en el puerto 22. Estos son solo ejemplos. Las opciones son ilimitadas.

Entender las definiciones de notificación

Antes de empezar con nuestros ejercicios, un rápido glosario te ayudará a comprender qué significan las opciones de notificación de un único carácter. Recurre a esta lista durante los ejercicios para comprender lo que realmente estamos haciendo cuando especificamos, por ejemplo, **w,u,c,r,f**.

Cuando veas **w,u,c,r,f,n**, estas son las definiciones:

- w Notifica si está en estado de aviso,
- u Notifica si está en estado desconocido,
- c Notifica si está en estado crítico,
- r Notifica si se ha recuperado de un estado previamente adverso,
- f Notifica si el servicio está fallando (se enciende y se apaga de vez en cuando)
- n Nunca se notifica

Cuando veas **d,u,r,f,s,n**, estas son las definiciones:

- d Notifica si el host está caído,
- u Notifica si el host es inaccesible (por ejemplo, Internet inactivo),
- r Notifica la recuperación,
- f Notifica si el host está en modo flapping,
- s Notifica si el tiempo de inactividad del servicio programado empieza o termina,
- n Nunca se notifica

Ejercicio 1: Monitorizar tu sitio web con check_http

Tu sitio web es la cara visible de tu negocio. Si alguna vez cae por alguna razón, o se vuelve lento, es importante ser proactivo para remediar la situación. Sólo hay una cosa peor que un cliente contacte contigo para hacerte saber que tu sitio web está caído y es darte cuenta que lleva inactivo durante una semana y los clientes durante ese tiempo no te lo han notificado. Simplemente se fueron a otro sito. Que tu sitio web se vuelva lento o no responda también puede dañar tu posicionamiento SEO.

NEMS Linux puede vigilar de cerca tu sitio web y enviarte una alerta por correo electrónico, Telegram o Pushover si tu sitio se desconecta, deja de responder o se vuelve lento. Esto hace que NEMS Linux sea una herramienta fantástica para los diseñadores web y hosts que desean asegurarse de que los sitios de sus clientes estén siempre activos para que éstos no noten ningún tipo inactividad. Si tu sitio está alojado a través de SSL, NEMS puede incluso notificarte si tu certificado ha caducado o está a punto de caducar. Son muchas las opciones ya que NEMS Linux ha sido creado para monitorizar prácticamente todo.

En nuestro primer ejercicio de este mes, usaremos el comando check_http integrado. Para mi ejemplo, usaré https://nemslinux.com/. Te sugiero hacer lo mismo por el bien de la lección y luego intentar cambiar el Host por tu propio dominio una vez que entiendas cómo está todo interconectado. Espero que hayas completado los dos artículos anteriores en los números de octubre y noviembre de 2018 de ODROID Magazine. Si no, vuelve atrás y léelos primero. Si estás listo, ¡vamos a ello! Puede parecer algo engorroso cuando eche un vistazo a los siguientes 6 pasos, pero ten en cuenta que una vez que hayas creado tu configuración, podrás reutilizarla para tantos hosts de sitios web como quieras, simplemente asignando tu host al grupo de host web_site_ssl, que aprenderás a crear a continuación.

Abre NEMS NConf y sigue estos pasos:

- Para empezar, debemos asegurarnos de que nuestro comando de verificación esté listo para nuestro caso práctico. Aunque el valor por defecto está cambiando en NEMS 1.5, si estás en NEMS 1.4.1, deberás cambiar el comando de verificación para usar controles de nombre de host en lugar de controles por dirección IP.
- Muestra tu lista de “checkcommand”
- Edita check_http
- Actualmente, la línea de comandos utiliza –I %HOSTNAME%, con -I que significa “Dirección IP”. Cambie esto por -H (nombre de host) para que ahora sea … –H %HOSTNAME% … Ahora podemos usar el nombre de host de nuestro sitio web o una dirección IP para el comando check_http.
- Guarda tus cambios

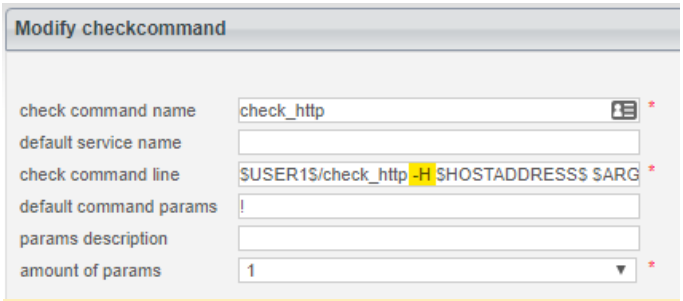


Figura 1: Modifica check_http para usar el nombre de host en lugar de la dirección IP

A continuación, debemos configurar nuestro comando check-host-alive, el cual se usa para hacer ping a los hosts para determinar si están activo o caído. El nombre de host de mi sitio web solo responderá en IPv4, aunque el comando check-host-alive por defecto en NEMS 1.4.1 usa IPv6. En lugar de editar el comando de muestra, vamos a añadir uno nuevo basándonos en él, pero éste únicamente usará IPv4. De esa forma, podremos usar el comando antiguo si necesitamos IPv6 para un dispositivo diferente.

- Mostrar la lista de “misccommand”.
- Editar check-host-alive.
- Resalta y copia la línea de comandos completa en tu portapapeles.
- Haga clic en “Add” junto a “misccommand” para agregar un nuevo comando.
- Nombra tu nuevo comando check-host-alive-ipv4
- Pega la línea de comandos desde tu portapapeles.
- Al final de la línea de comandos, simplemente agrega un espacio, seguido de -4 para indicar que vas a usar IPv4 para esta verificación.
- Guarda el nuevo comando.

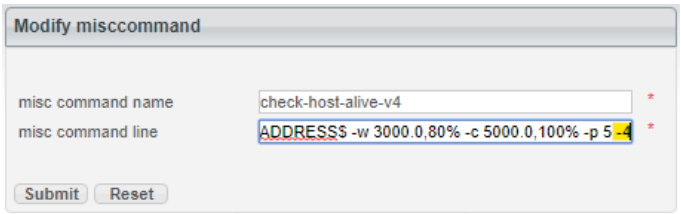


Figura 2 – Crear un nuevo “misccommand” para check-host-alive usando IPv4

- Nuestros comandos ya están listos, de modo que ahora es el momento de configurar nuestro “hostpreset”. Queremos crear uno para los sitios web IPv4. De esta manera, podemos reutilizar el “preset” para cualquier sitio web que queramos monitorizar con NEMS Linux.
- Añade un nuevo “hostpreset”.
- Nombra tu sitio web preset IPv4
- Configura “host alive check” en el nuevo comando que creaste en el Paso 2: check-host-alive-ipv4
- Guarda tu “hostpreset”.

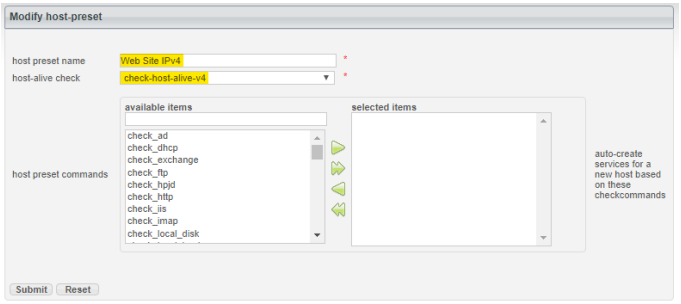


Figura 3 – Nuevo “hostpreset” para sitios web IPv4

Hasta ahora, todo lo que hemos hecho se puede reutilizar para cualquier sitio web cuyo nombre de host se resuelva en una dirección IPv4. De aquí en adelante, sin embargo, configuraremos nuestro grupo de hosts específicamente para un sitio web seguro (SSL).

- Añade un nuevo “hostgroup”.
- Asigna un nombre a este web_site_ssl
- Deja todo lo demás como está y guarda tu nuevo hostgroup.

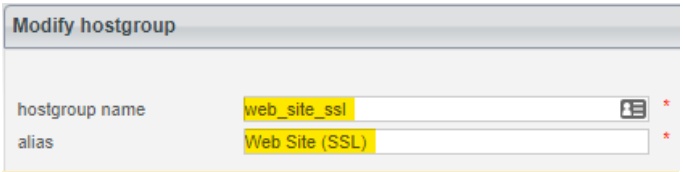


Figure 4 – New hostgroup for web_site_ssl

¿Por qué creamos un nuevo hostgroup si la configuración se reduce a un simple nombre? Bueno, aquí es donde la magia entra en juego. Ahora tenemos un comando “check”, un comando “check host alive”, un “hostpreset” y un “hostgroup”. Ahora, podemos vincularlos todos, iniciando un Servicio Avanzado. Recuerda, la idea es que todo lo que hacemos se pueda asignar a tantos hosts como queramos. No tengamos que volver a hacer todo esto para el siguiente sitio web.

- Haz clic en “Add” junto a “Advanced Services”.
- Nombra tu servicio: Web Site (SSL)

- Proporciónale un alias: Uptime of SSL Web Site
- Ajusta el período de verificación y el período de notificación en 24×7
- Para asignar el servicio avanzado al “hostgroup”, resalta el hostgroup que creamos web_site_ssl y presiona la flecha verde para agregarlo a la lista de elementos seleccionados.
- En los grupos de contacto, asegúrate de agregar administradores también. De lo contrario, no recibirás notificaciones.
- Configura tus notificaciones de la siguiente manera: max check attempts: 10 ; check interval: 1 ; retry interval: 5 ; first notification delay: 10 ; notification interval: 30 ; notification options: w,u,c,r
- Finalmente, configura tus “service parameters”: -S –sni
- Guarda tu servicio avanzado.

Consejo: -S le dice a check_http que este sitio usa SSL, y –sni habilita SNI (Indicación del nombre del servidor) puesto que yo uso CloudFlare para SSL en nemslinux.com, y por lo tanto mi dirección IP está asociada con más de un nombre de dominio. Para tu sitio, si tienes algún problema, intente eliminar SNI simplemente omitiendo –sni. Para disponer de toda la documentación sobre el comando check_http, visita la página wiki de documentación de NEMS Linux en **https://docs.nemslinux.com/check_commands/check_http**

Figura 5 – Creación de un servicio avanzado para comprobar sitios web SSL

Finalmente, agregaremos nuestro sitio web host. A partir de ahora, este es el único paso al que debemos recurrir para agregar más sitios a tu servidor NEMS Linux.

- Añadir un nuevo host.
- Configura lo siguiente:
- hostname: https://nemslinux.com
- alias: NEMS Web Site
- address: nemslinux.com
- OS: Linux
- host preset: Web Site IPv4 (See what we did there?)
- monitored by: Default Nagios
- host is collector: no
- check period: 24×7
- notification period: 24×7
- max check attempts: 10
- check interval: 1
- retry interval: 5

- first notification delay: 10
- notification interval: 30
- notification options: d,u,r,f
- asigna el host a hostgroup (¿estás listo para esto?): web_site_ssl
- Guarda.
- Genera tu configuración

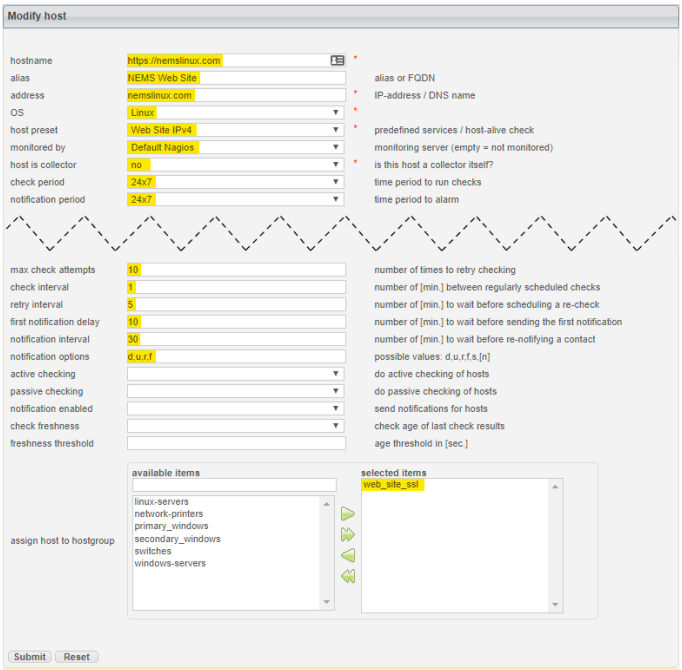


Figura 6 – Creación de un host para monitorizar el sitio web SSL IPv4

Si has seguido todos los pasos correctamente y mi sitio web está activo, Adagios debería informar de que todo está bien. Para probar qué sucedería si empezara a fallar, cambia el nombre de host en el Host a nemslinux.com1 (que obviamente no responderá), y luego genera tu configuración nuevamente. Una vez que estés preparado, cambia el Host por el de tu sitio web. Si su sitio es SSL, solo debes cambiar el nombre de host, el alias y la dirección del host (Paso 6). Si no es SSL, repite el Paso 4, pero esta vez crea un nuevo grupo de host llamado web_site_no_ssl, y luego repite el Paso 5. En esta ocasión, creas un nuevo Servicio Avanzado llamado Sitio Web (no SSL), asígnalo (5.e) al Sitio web (no SSL) y omite los parámetros SSL en 5.h.

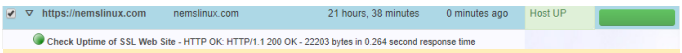


Figura 7 – NEMS Adagios muestra que nemslinux.com está ACTIVO

Ejercicio 2: Monitorizar un puerto no estándar con check_tcp

Este es un ejercicio adicional que te ayudará a controlar el tiempo de actividad de cualquier puerto TCP/UDP. NEMS Linux incluye una escucha de puerto ficticio que se ejecuta en el puerto 9590. La escucha del puerto se llama hábilmente 9590, y lo único que hace es responder si está activo. Esto se puede utilizar para simular un puerto en otro dispositivo. Configuraremos un monitor de servicio en el host NEMS para que nos avise si el puerto 9590 se desconecta alguna vez.

- En el menú de la izquierda de NConf, verás “Services”. Haga clic en “Add”.
- Define el Service Name con: 9590
- Deja Service Enabled configurado en: Yes
- Ajusta el Check Command en: check_tcp
- Ajusta Assigned to Host em: NEMS (este host viene pre-instalado)
- Deja Check Period fijado en: 24×7
- Deja Notification Period en: 24×7
- Deja Service Templates como está, no seleccionado.
- En Contact Groups, resalte el grupo de ‘administradores’ y presiona la flecha que apunta hacia la derecha para moverlo a Elementos seleccionados.
- Deja Notes, Notes URL y Action URL en blanco.
- Fija Max Check Attempts en: 30
- Fija Check Interval en: 1
- Fija Retry Interval en: 1
- Fija First Notification Delay en: 5
- Fija Notification Interval en: 15
- Fija Notification Options en: w,u,c,r,f,s
- Deja en blanco Active Checking, Passive Checking, Notification Enabled, Check Freshness y Freshness Threshold.
- Deja Assign Service to servicegroup tal cual, ninguno seleccionado.
- Configura Params for check command to the port number: 9590
- Presiona “Submit”
- Presiona “Generate Nagios Config”, luego haz clic el botón “Generate” en la siguiente pantalla para implementar y activar tu nueva configuración.

Una vez que se ejecute la nueva configuración, intenta hacer que falle el servicio abriendo “Monit Service Manager” debajo de “System” en el cuadro de

mandos de NEMS. Haga clic en el proceso denominado 9590 y luego haz clic en “Stop service”. Notarás que, en aproximadamente 1 minuto, el estado de 9590 mostrará un problema en todas las vistas de estado (por ejemplo, NEMS TV Dashboard, NEMS Adagios, Nagios Core), y tras aproximadamente 5 minutos recibirás una notificación (asumiendo que los parámetros de tus notificaciones estas configurados correctamente). Una vez recibida una notificación, visita NEMS Adagios para reconocer el corte. Luego, vuelve a Monit, abre el proceso 9590 y haz clic en “Enable Monitoring”. Esto volverá a cargar 9590 y pronto verás que cambia a un estado recuperado. Una vez completado, intenta configurar un nuevo servicio para monitorizar un host real de tu red. Simplemente cambia el nombre del servicio, el host en el paso 5 (ya sabe cómo agregar nuevos hosts si aún no lo tienes configurado) y el número de puerto en el paso 19.

Aprender más

NEMS tiene un foro comunitario muy activo. Entro con bastante frecuencia para proporcionar soporte gratuito a los usuarios. También ofrezco soporte comercial personalizado para aquellos que necesitan un mayor nivel de soporte. NEMS Linux es gratis para descargar y usar. Su código fuente está disponible en GitHub. Descarga NEMS Linux para ODROID en **https://nemslinux.com/**

También puede seguir @NEMSLinux en Twitter o **unirte a nosotros en Discord**.

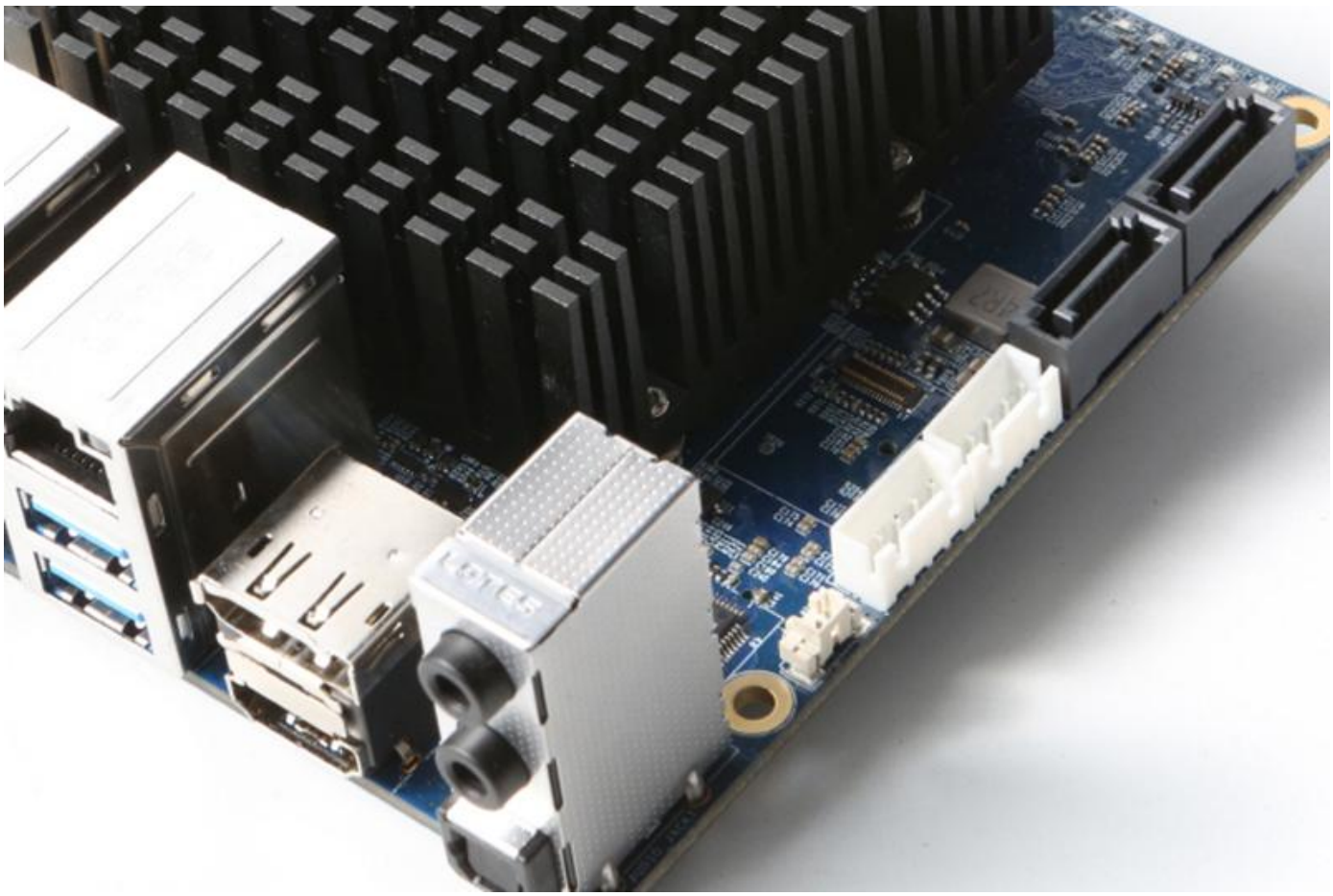
Asegúrate de leer mi artículo en la edición del próximo mes de ODROID Magazine donde revelare las increíbles mejoras de NEMS Linux 1.5 y mostraré cómo actualizar desde NEMS 1.4.1.

Sobre el Autor

Robbie Ferguson es el dueño de Category5 Technology TV y autor de NEMS Linux. Su programa de televisión se encuentra en **https://category5.tv/** y su blog es **https://baldnerd.com/**.

ODROID-H2 Parte 2: Características de la Bios y Acceso Remoto

December 1, 2018 By Justin Lee ODROID-H2



Como cualquier PC genérico, el ODROID-H2 cuenta con una ROM Flash BIOS de 8MiB soldada a la placa. Cumple con la Especificación 2.6 de UEFI y los requisitos de arranque PXE. Sin embargo, el firmware Intel UEFI no es compatible con CSM versión 2.0 para el arranque de sistemas operativos heredados, como DOS, XP, Windows 7, etc. El menú de configuración principal aparece cuando se presiona la tecla “Suprimir” durante el proceso de arranque.

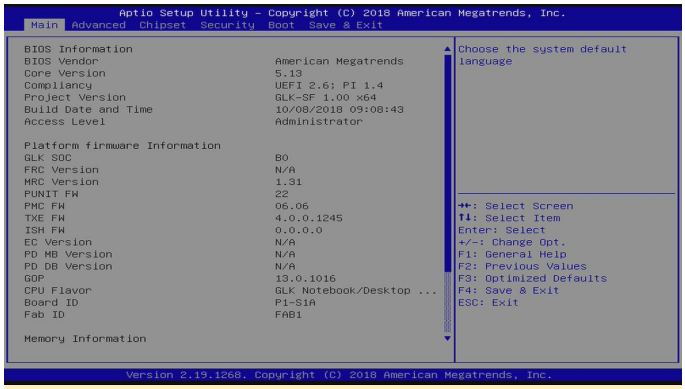


Figura 1 – Menú de configuración principal

Este es el menú de configuración avanzada:

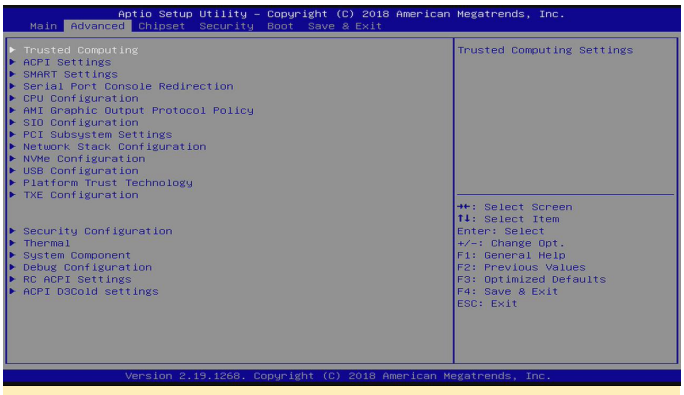


Figura 2 – Menú de configuración avanzada

Este es el menú de configuración de inicio que te permite elegir el medio de arranque. ODROID-H2 puede arrancar desde los medios de almacenamiento eMMC, USB, SATA y NVMe. Puedes acceder a ellos al mismo tiempo desde el sistema operativo.

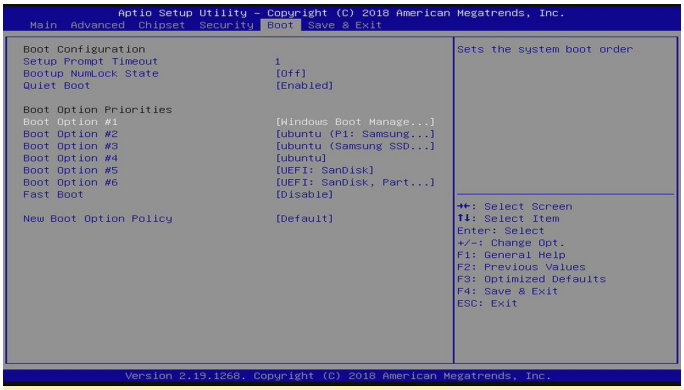


Figura 3 – Menú de configuración de arranque

Puedes cambiar la prioridad del arranque en el menú de arranque o presionar F7 para elegir temporalmente el medio deseado durante el proceso de arranque

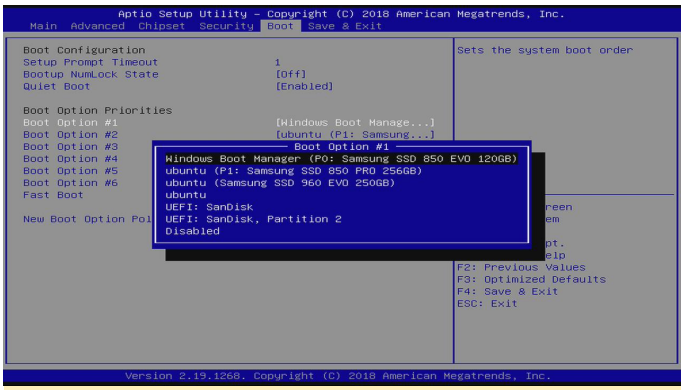


Figura 4 – Cambiar la prioridad de arranque

Wake on LAN (WoL). Puede activar la función WoL con la línea de comandos. Por ejemplo:

```
$ sudo ethtool -s enp3s0 wol g
```

Comprobar el estado actual.

```
$ sudo ethtool enp3s0 | grep Wake
$ Supports Wake-on: pumbg
$ Wake-on: g
```

Si detectas la “g”, la función Wake-on-LAN está habilitada

Si puedes, activa la placa ODROID-H2 con este comando desde un PC remoto.

```
$ powerwake 192.168.30.4
```

La función WoL solo funciona con un puerto Ethernet junto con el puerto HDMI/DP.

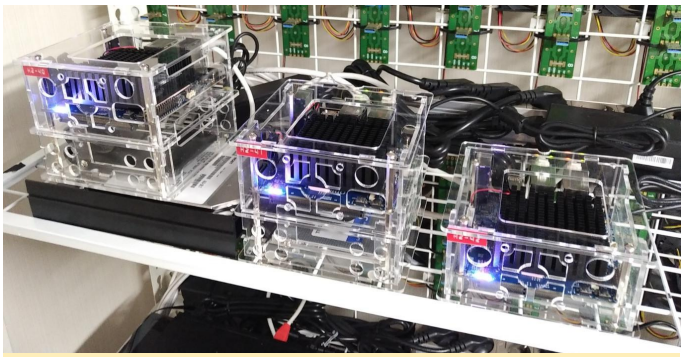


Figura 5 – Tres configuraciones de ODROID-H2 con diferentes memorias DDR4

Hardkernel introdujo el ODROID-BENCH para proporcionar a los usuarios la oportunidad de utilizar los ordenadores de placa reducida ODROID de forma remota. Ahora configuraremos los nuevos ODROID-H2 con algunas combinaciones de memoria DDR4y almacenamiento diferentes.

	#1	#2	#3
Port Number	2240	2242	2244
DDR4 Memory	8GB (Dual 4GB) Essencore ***DDR4-2400CL 15-15-15***	8GB (Dual 4GB) Samsung ***PC4-2400T-SC0-11***	32GB (Dual 16GB) Samsung ***PC4-2400T-SE1-11***
Storage (SATA)	SSD Samsung EVO 860 256GB	SSD Toshiba Q 128GB	NVMe SSD Western Digital Black 500GB
IP address	192.168.0.40	192.168.0.42	192.168.0.44

Figura 6 – Configuraciones para cada uno de nuestros tres ODROID-H2

Se puede acceder a ellos a través de “ssh” con un número de puerto dedicado para cada máquina. Su acceso está restringido al contenedor Docker en Ubuntu 18.04.1 y al kernel de Linux 4.15.0-38-generic.

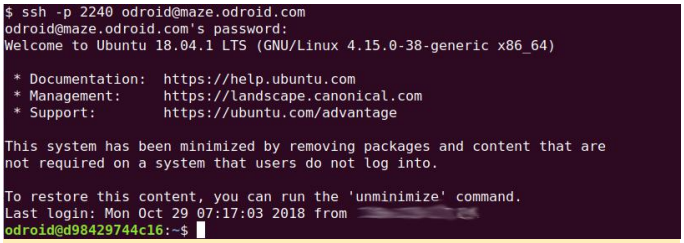


Figura 7 – Docker para Ubuntu 18.04.1

Aun así, puedes ejecutar un montón de comandos del sistema con privilegios de root y capturar la información del hardware. Incluso puede ejecutar una herramienta para probar el rendimiento, aunque la puntuación estará muy cerca o será ligeramente inferior a la probada en el entorno nativo, puesto que accedes a un contenedor.

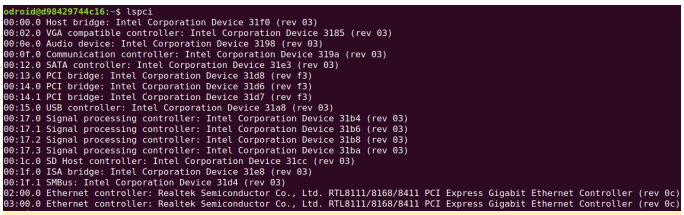


Figura 8 – Ejecutando una prueba de rendimiento

Esta es la puntuación “iozone” de la unidad #1 ODROID-H2 dentro del contenedor de Ubuntu.

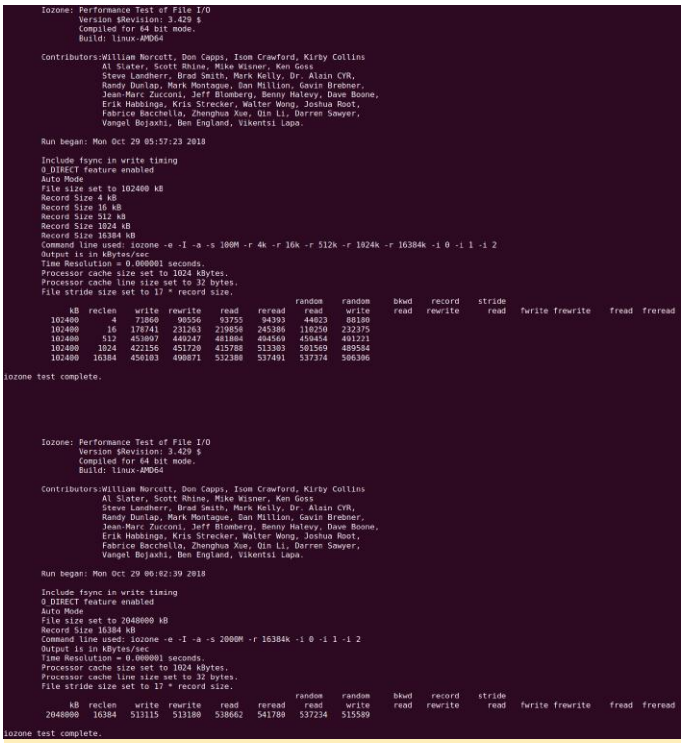


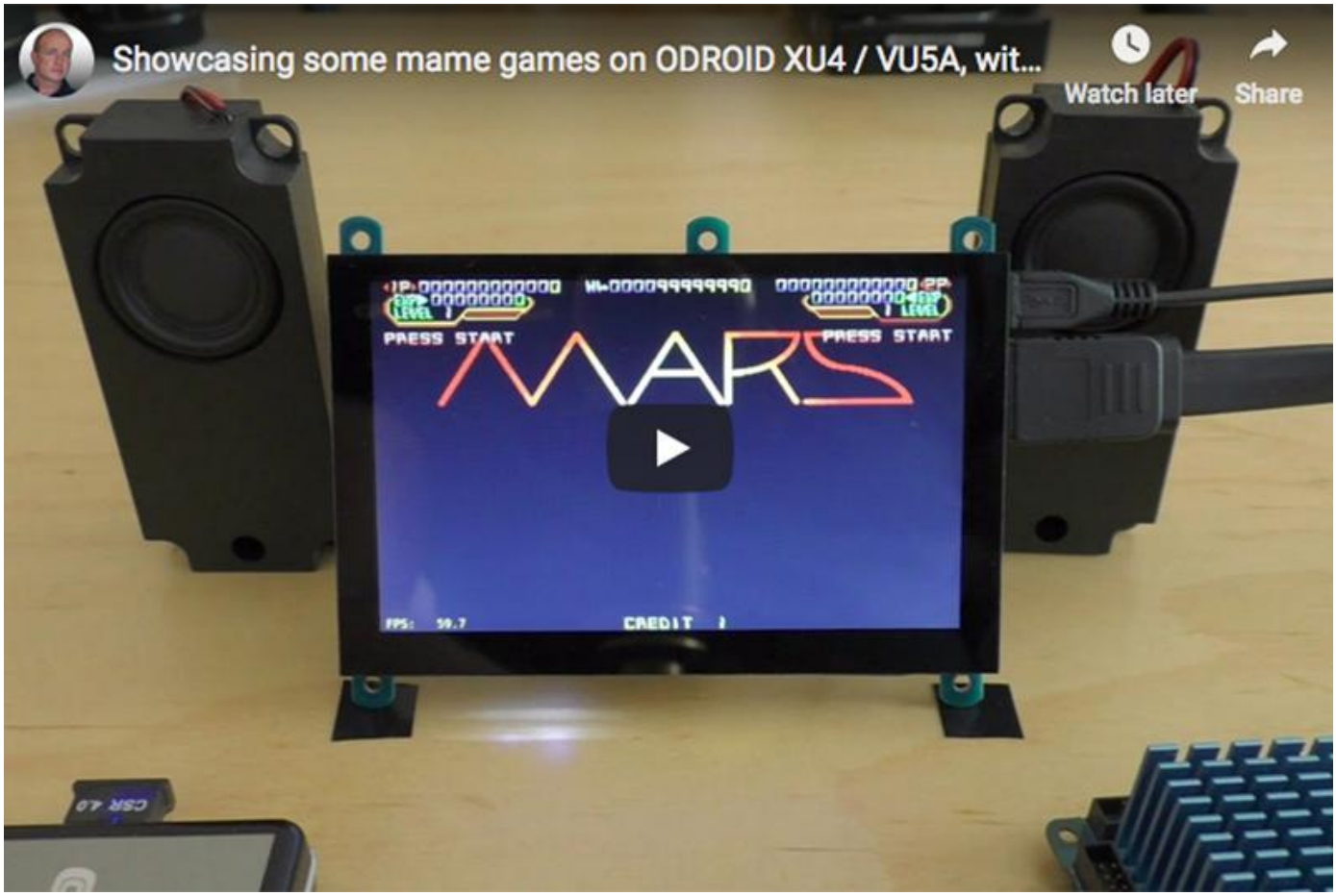
Figura 9 – Unidad #1 ODROID-H2 en el contenedor de Ubuntu

Si tienes algún problema para acceder a las máquinas en ODROID-BENCH o para hacer otras solicitudes, consulte el hilo en **viewtopic.php?f=29&t=32257**.

Para comentarios, preguntas y sugerencias, visite el post original en **https://forum.odroid.com/viewtopic.php?f=29&t=32536**.

Compilando RetroArch

December 1, 2018 By AreaScout Juegos, ODROID-XU4



Si estás buscando una interfaz para emuladores de juegos, puedes probar RetroArch. Se ha exportado a la familia de ordenadores de placa reducida (SBC)ODROID-XU4. Puedes seguir los siguientes pasos para instalarlo y utilizarlo en tu sistema.

Building and configure RetroArch

Necesitamos obtener el código fuente, aplicar un parche necesario y compilarlo. El pequeño parche básicamente lo que evita es que se muestre el menú con 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
```

Ahora inícialo con el comando:

```
$ retroarch
```

Aplicar algunas configuraciones útiles

Aunque no tienes por qué usar las configuraciones que aparecen a continuación, las he incluido para que pueda utilizarlas si quieres, como punto de partida y retocarlas a tu gusto más tarde.

Para Actualizar los recursos (iconos, imágenes de fondo y demás), puedes encontrarlos aquí:

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

Te recomiendo que actualices estos paquetes: **Core Info Files, Joypad Profiles, Database, GLSL Shaders.** Puedes usar **Core Updater** para conseguir algunos emuladores.

Habilitar la configuración avanzada:

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

Activar Threaded Video: mejorará bastante la emulación:

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

Activa también el contador de FPS. Es muy útil ver como se ejecuta de rápido la emulación, especialmente cuando estas configurando algo:

```
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 estás sobre la 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 guardados en una carpeta en tu ODROID-XU4, puede buscarlos:

```
Import Content -> Scan Directory
```

Cuando se te solicite, puedes seleccionar la carpeta de juegos raíz para permitir que RetroArch escanee tus juegos. Aparecerán en la parte derecha del menú después de un tiempo.

Referencias

https://www.retroarch.com/
https://forum.odroid.com/viewtopic.php?f=98&t=32173#p233821
https://youtu.be/6Ewgov7_TXM

Conociendo un ODROIDian: Kamots Tech

December 1, 2018 By Rob Roy Conociendo un ODROIDian

Por favor háblanos un poco sobre ti. Vivo en Florida (también conocido como el estado del sol), donde nací y crecí. Siempre he vivido en Florida porque hace calor, hay mucho que hacer y la industria TI ha estado en constante crecimiento con muchas promesas en el horizonte. Fui a la universidad para especializarme en redes informáticas y, desde que me gradué, he trabajado en las tecnologías de la información durante más de 15 años. Estoy casado y mi esposa trabaja en marketing. Tenemos un perro. Es un Weimaraner de 9 años y se ha ganado muchos apodos, incluido Sir Barks A Lot.

Figura 1 – Una puesta de sol en Florida

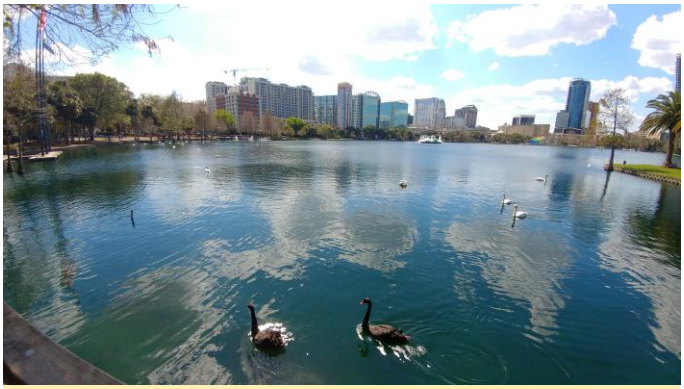


Figura 2 – El horizonte de Orlando, Florida

Figura 3 – Visitando la reserva de lobos en el norte de Florida

¿Cómo empezaste con los ordenadores? Al principio empecé con los ordenadores por el mero interés que me despertaba la electrónica cuando era joven. Mi padre, que es ingeniero, recibió un ordenador de la empresa para la que trabajaba y me fascinó, pero no me permitieron usarlo porque tenía la reputación de desarmar cosas y solo pocas veces volver a armarlas. Ahorré dinero cortando césped y finalmente le compré el ordenador 8088 a un amigo. Estaba fabricado por una empresa llamada Leading Edge y ejecutaba DOS 5.5 desde un disco duro de 40 MB, a pesar de ello me parecía increíble. Aprendí BASIC y luego Turbo C en ese ordenador con el que todavía disfruto programando con algunas variantes de C hoy en día. Internet existía, pero me conectaba principalmente con BBS para obtener archivos y jugar a juegos hasta que más tarde, cuando me hice con un ordenador mejor, me empecé a interesar por Linux.

Figura 4: Kamots empezó utilizando un ordenador Leading Edge

¿Qué te atrajo a la plataforma ODROID? La primera vez que oí hablar de la plataforma ODROID y HardKernel fue cuando leí sobre ODROID-GO, no recuerdo dónde lo leí exactamente. Pensé que era genial ya que ofrecía mucho potencial por un buen precio, decidí comprar uno de inmediato. Luego me involucré con la comunidad de ODROID y empecé a hacer videos en YouTube sobre GO. Disfrutaba ayudando a otros a descubrir cómo usar los emuladores y a explorar nuevos proyectos, como conectar un sistema de carga inalámbrico y testear nuevos emuladores.

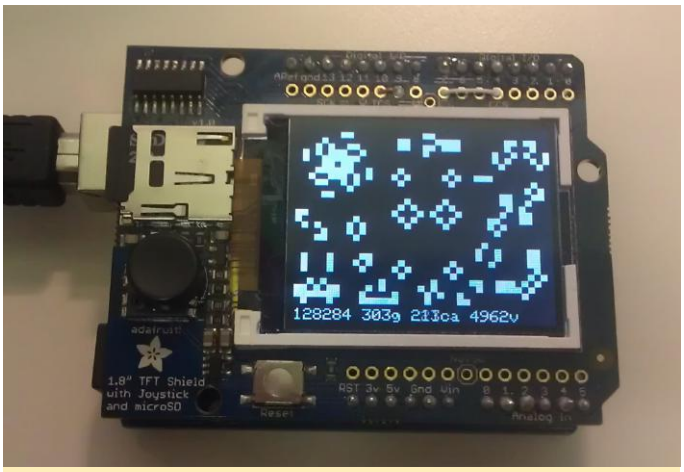


Figura 5 – Conway's Game Of Life en Arduino

Figura 6 – Proyecto prematuro de un sensor de temperatura 1-wire

¿Cómo usas tus ODROIDs? Ejecuto varios emuladores en mi ODROID-GO y ayudo con el desarrollo según me lo permite el tiempo. Recientemente, he disfrutado bastante aprendiendo de la plataforma Commodore 64 usando el nuevo emulador en mi GO. Espero con interés futuras incorporaciones a esta plataforma.

¿Cuál es tu ODROID favorito y por qué? El ODROID-GO. Simplemente es diversión portátil. Tiene un toque de nostalgia, de modo que creo que es la razón por la

que la mayoría de las personas se sientes atraídos por él al principio. Estoy disfrutando de nuevo de los juegos más antiguos.

¿Qué innovaciones te gustaría ver en futuros productos Hardkernel? Creo que sería bueno desarrollar más productos como ODROID-GO que hacen que la electrónica sea muy divertida. Un ordenador de placa reducida asequible que admita SSD M.2 sería lo siguiente en mi lista de deseos. Soy consciente de que estos dispositivos pueden estar disponibles en otros sitios, pero me encantaría ver a HardKernel desarrollar una versión pequeña del tamaño del ODROID-C1 +.

¿Qué aficiones e intereses tienes aparte de los ordenadores? Mi esposa y yo somos buceadores. Para mí, es lo más cerca que probablemente estaré del espacio exterior. Suelo seguir la ciencia espacial, como las misiones de Mars rover, los Voyagers, New Horizons y la Estación Espacial Internacional. Intento ver cada lanzamiento o evento importante si mi horario de trabajo me lo permite. Cuando era joven, solía escuchar las transmisiones de audio de la NASA durante las misiones del transbordador espacial. He sido radioaficionado durante más de 20 años y todavía me resulta divertido comunicarme con todo el mundo usando únicamente una antena de patio. Principalmente utilizo modos digitales como JT65 y PSK31 en bandas HF (onda corta). También me gusta ir al campo y hacer agujeros en un papel lejano con un arco o arma de fuego, aunque no cazo. Me gusta hacer geocaching y viajar tanto con mi esposa como con mis amigos.

Figura 7 – Un elemento rastreable de geocaching

Figura 8 – El puente Golden Gate de San Francisco

Figura 9 – Una playa jamaicana

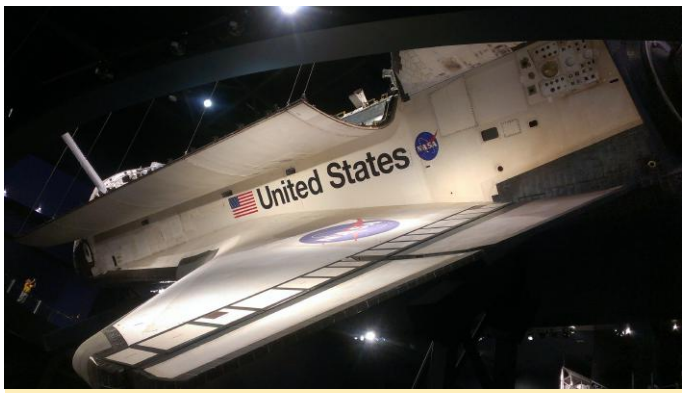


Figura 10 – El transbordador espacial Atlantis en el Centro Espacial Kennedy

¿Qué consejo le darías a alguien que quiera aprender más sobre programación? Empezar con un dispositivo que puedas programar para que interactúe con el mundo. Al principio, puede ser aburrido escribir un programa donde el resultado solo aparece en una pantalla. Sin embargo, cuando alguien escribe un simple programa que logra algo del exterior, como cambiar el volumen del televisor o monitorizar el clima exterior con un sensor remoto, creo que hace que las cosas sean más tangibles y que la imaginación empiece a ver otras oportunidades en el mundo real. Esto puede hacer que resulte más divertido para alguien que simplemente está aprendiendo y potencia proyectos futuros. Todos aprendemos de un modo diferente, así que prueba diferentes cosas para ver cómo aprendes mejor y mantente inspirado para hacer cada vez más. Recomiendo con el tiempo aprender un lenguaje basado en C ya que muchos lenguajes de programación están basados en C. Te ayudará a entender muchos lenguajes diferentes una vez que logres entender los conceptos básicos de C.