

Estrategias para ports de Firmwares Linux V2

Arma tu propio Frankenstein!

DSR! - Indetectables.net

DragonJAR CON 2021

Emmanuel Seoane

DSR!

Desarrollador de Software, líder de equipo y
Hacker los fines de semana.

Hice desde cracks y keygens hasta tools y
malware. Colaboré con varios proyectos
GPL entre ellos Metasploit.

Formo parte del Staff de Indetectables.net
donde vengo rompiendo cosas desde hace
más de 10 años.



¿Que es un Port?

Quien diria que no está en la RAE

Se le llama "**port**" al proceso de adaptar software con el fin de lograr alguna forma de ejecución en un entorno informático que es diferente de aquel para el que se diseñó originalmente.

El término también se usa cuando se cambia el software / hardware para hacerlos utilizables en diferentes entornos.

El software es portable cuando el costo de trasladarlo a una nueva plataforma es **significativamente menor que el costo de escribirlo desde cero**.



Referencias:

<https://en.wikipedia.org/wiki/Porting>

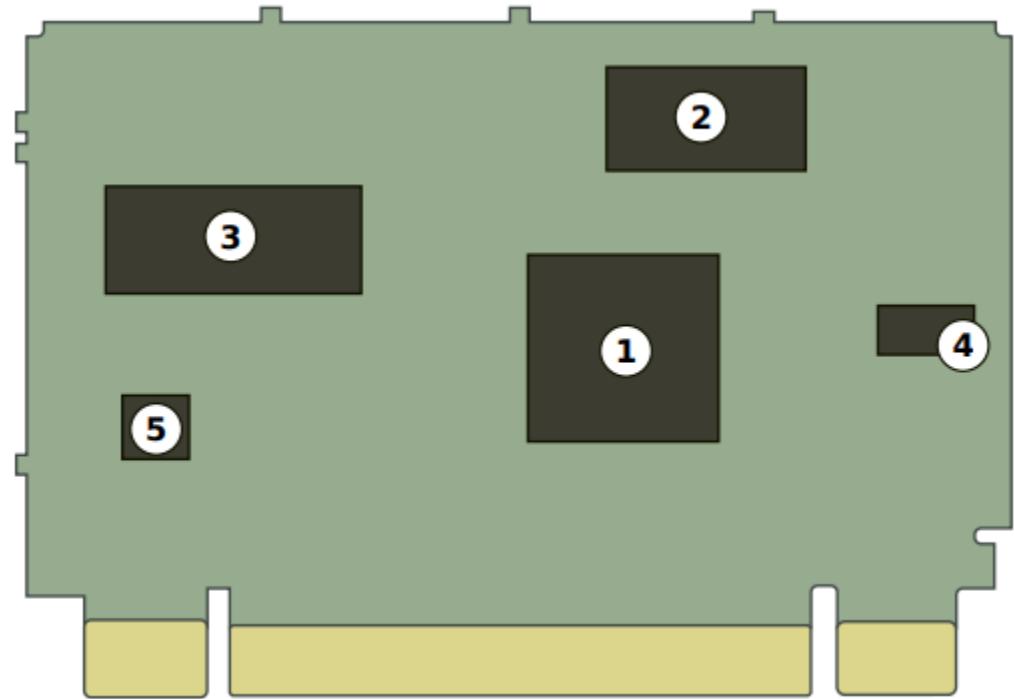
<https://youtu.be/JqP3ZzWiul0>



¿Por qué hacer un port?

¿Por qué no?

Originalmente el principal interesado en esto eran los desarrolladores y las empresas pero hoy en día no sería la regla.



- Re utilizar hardware que tengamos a mano
- Combatir el abandonware
- Utilizar hardware que realmente no tenemos...



- Permitirnos que nuestro desarrollo sea más utilizado al correr en múltiples plataformas
- Re utilizar desarrollos viejos sin siquiera volver a compilar



Eligiendo un hardware para portear



Ejemplo práctico

Eligiendo un hardware a portear



Para hacer las cosas más entretenidas vamos a tomar de ejemplo la Pineapple NANO que es un hardware que todos en algún momento miramos con amor.

El primer paso sería ir reconociendo el hardware y software para encontrar candidatos para suplirlo.



Reconociendo el hardware



Para ver esto podemos recurrir a diversas fuentes como lo son la página del FCC, teardowns del hard que encontremos por internet, deviwiki y la página de OpenWrt.

El hardware esta conformado por:

- CPU: 400 MHz MIPS Atheros AR9331 SoC
- Architecture MIPS32 24K
- Memory: 64 MB DDR2 RAM
- Disk: 16 MB ROM
- WiFi: 1- AR9331 (built-in), 1- AR9271 (via USB bus)
- Misc: 1 - USB 2.0, 1- micro SD, 1- status LED, reset button

Referencias:

La FCC es tu mejor amiga en esto

<https://fccid.io/2AB87-NANO/Internal-Photos/Int-photo-2886940>



Reconociendo el software

```
1 NAME="OpenWrt"
2 VERSION="19.07.2"
3 ID="openwrt"
4 ID_LIKE="lede openwrt"
5 PRETTY_NAME="OpenWrt 19.07.2"
6 VERSION_ID="19.07.2"
7 HOME_URL="https://openwrt.org/"
8 BUG_URL="https://bugs.openwrt.org/"
9 SUPPORT_URL="https://forum.openwrt.org/"
10 BUILD_ID="r10947-65030d81f3"
11 OPENWRT_BOARD="ar71xx/generic"
12 OPENWRT_ARCH="mips_24kc"
13 OPENWRT_TAINTS="no-all busybox"
14 OPENWRT_DEVICE_MANUFACTURER="OpenWrt"
15 OPENWRT_DEVICE_MANUFACTURER_URL="https://openwrt.org/"
16 OPENWRT_DEVICE_PRODUCT="Generic"
17 OPENWRT_DEVICE_REVISION="v0"
18 OPENWRT_RELEASE="OpenWrt 19.07.2 r10947-65030d81f3"
19 |
```

Al software podemos darle una mirada usando “**Firmware Mod Kit**” para extraer el filesystem de un update y explorarlo prestando atención a estas rutas:

/etc/openwrt_release
/etc/os-release
/etc/banner
/etc/shadow
/etc/passwd
/usr/lib/os-release
/lib/modules

Referencias:

<https://github.com/rampageX/firmware-mod-kit>
extract-firmware.sh fwupdate.bin



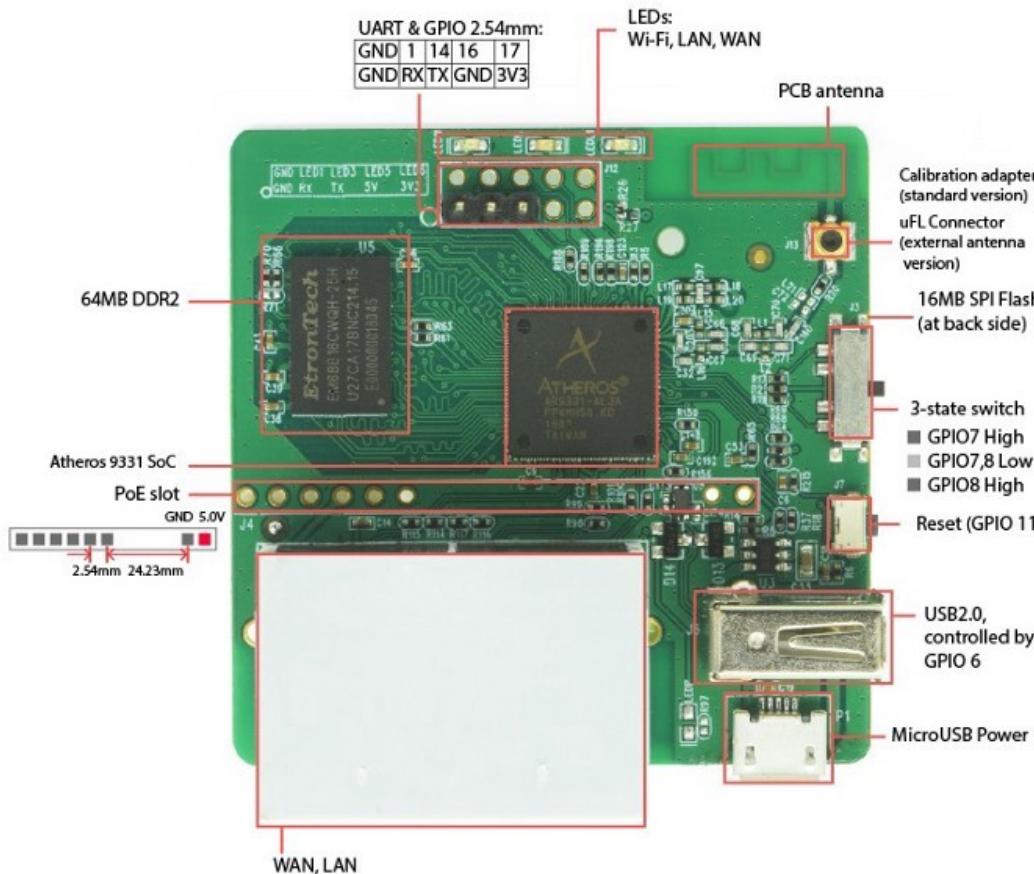
Buscando candidatos para suplirlo



Candidato encontrado!

¿Tan rápido?

Afortunadamente las características son idénticas al GL-iNet AR150, un router pensado para aplicaciones IoT.



- ✓ CPU: 400 MHz MIPS Atheros AR9331 SoC
 - ✓ Architecture MIPS32 24K
 - ✓ Memory: 64 MB DDR2 RAM
 - ✓ Disk: 16 MB ROM

WiFi: 1 - AR9331 (built-in)

Misc: 1 - USB 2.0, 1- status LED, reset button



Aprendiendo de otras implementaciones

Como la NANO es un hard que existe hace tiempo ya habían intentos de port hacia esta plataforma

Those are the steps I followed to build a working WiFi Pineapple firmware for the GL-AR150:

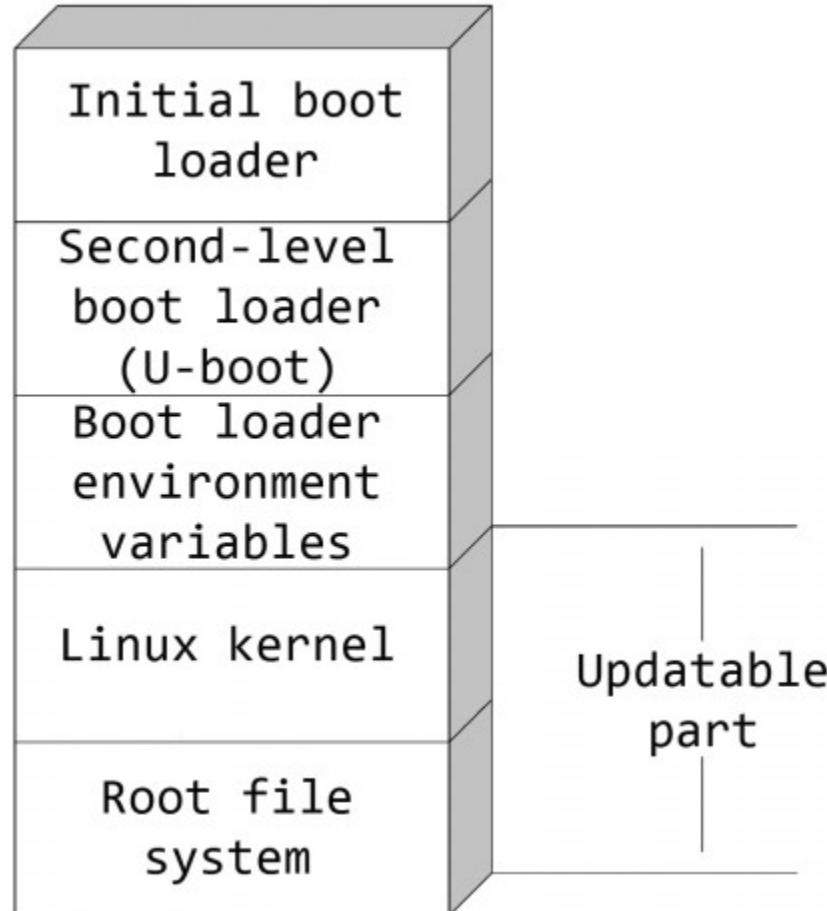
- Download the latest WiFi Pineapple Nano firmware from <https://www.wifipineapple.com/downloads>
- Extract the WiFi Pineapple firmware with binwalk (using the -e switch)
- Clone the repo <https://github.com/domino-team/openwrt-cc>
- Copy the content of the squashfs-root folder extracted by binwalk in the files/ folder on the repo just cloned (create the folder if it doesn't exist)
- Build OpenWRT (steps [here](#))

Referencias:

<https://www.securityaddicted.com/2016/11/17/weaponizing-gl-inet-gl-ar150/>
https://openwrt.org/docs/guide-user/additional-software/extroot_configuration/



Entendiendo la estructura de un firmware



Típicamente este tipo de dispositivos tienen esta estructura, donde el **bootloader** termina cargando **un kernel linux** que interactúa con el **file system**. Normalmente solo esa última parte es la que es actualizable.

Por lo menos esta sería la forma más rápida y concisa de explicarlo para no extenderme.



Planificando mi acercamiento



Objetivos

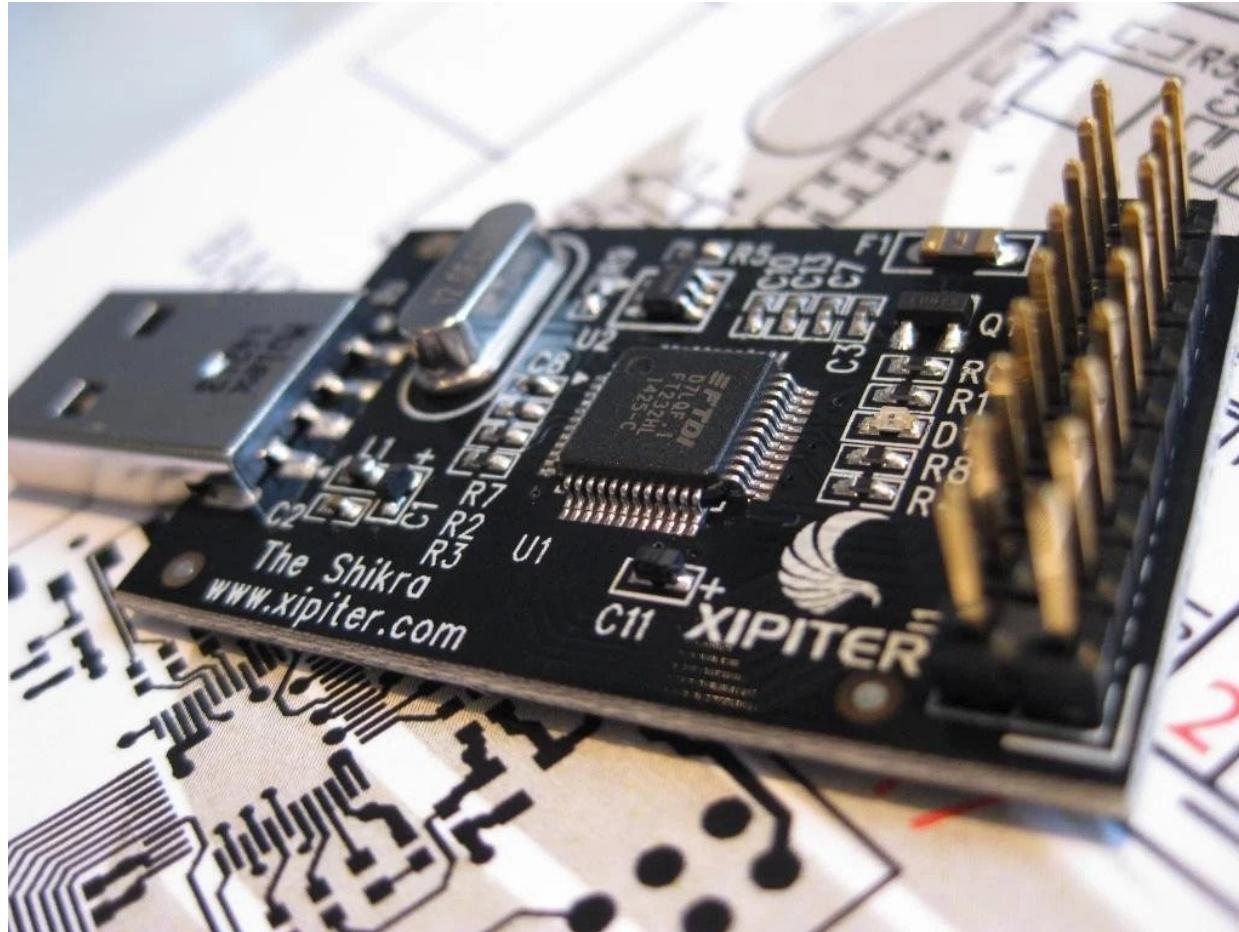
- ☑ Hardware anfitrión lo mas similar al original
- ☑ File system prácticamente intacto para que sea idéntico

¿Cómo lo logró?

Para esto decidí hacer un kernel idéntico al usado en el equipo original pero que soporte mi board



Armando nuestro laboratorio



Cuando jugamos con hardware y de esta manera vamos a necesitar por lo menos estas herramientas:



Interface universal UART

Podemos usar Pirate Bus 3.6, Shikra o alguna genérica de esas de 2 dólares.



Herramientas para unpack/pack de FW

Para esto podemos usar Firmware Mod Kit



Decompilador para MIPS/ARM

Acá es donde entra Ghidra al rescate

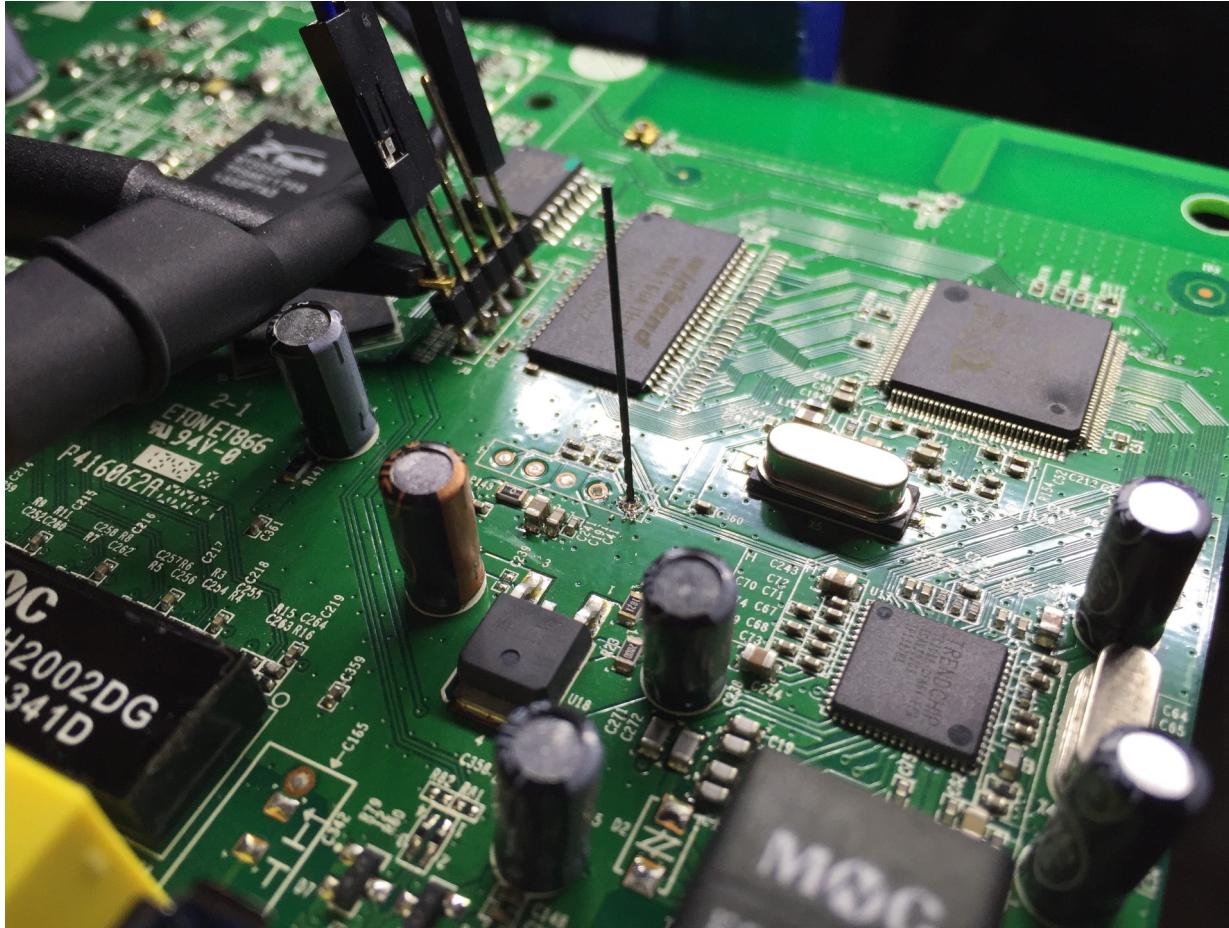


Herramientas básicas de electronica

Multimetro, destornilladores, púas, soldador, etc



Preparando el terreno



Teniendo nuestro acercamiento planificado y las herramientas listas lo más recomendable de hacer es tener todo preparado para poder ver vía **UART** que va sucediendo en el hardware cuando instalemos nuestro firmware.



¿Qué es UART?

UART significa “Receptor y transmisor asíncrono universal” por lo menos en castellano.

Este es el puerto standard en dispositivos embebidos para debugging y lo podríamos describir de forma rápida como un puerto serie de bajo voltaje porque trabaja en el rango de 5 a 3.3 voltios.



Referencias:

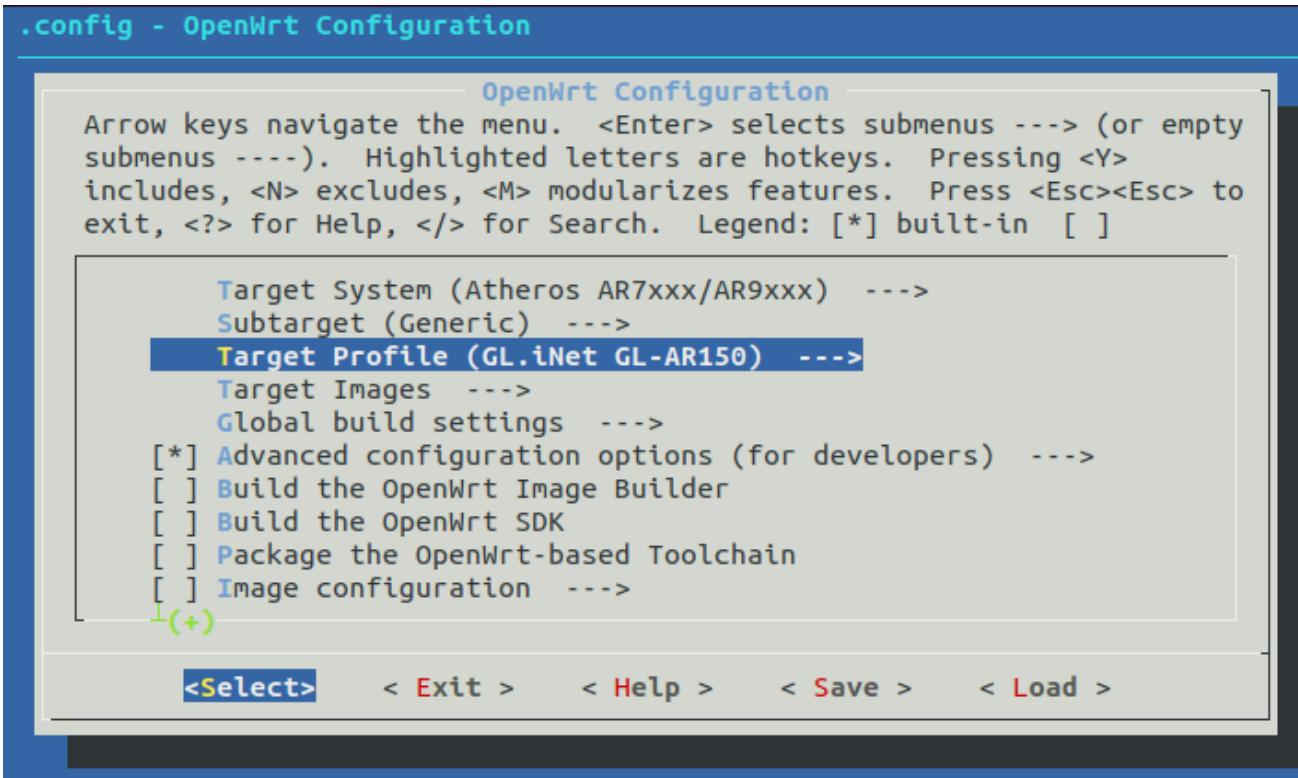
<https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html>

<https://www.circuitbasics.com/basics-uart-communication/>

<https://hetpro-store.com/TUTORIALES/puerto-serial/>



Haciendo el kernel



Reconocer que versión usaron de OpenWrt de base:

/etc/openwrt_release
/etc/os-release

Ver que dependencias usaron:

/lib/modules/
/usr/lib/opkg/status

Para simplificar el análisis del status log de opkg hice un script que pueden bajar desde este repo:

<https://github.com/xchwarze/wifi-pineapple-cloner>

Referencias:

Les dejo el .config que use para mis pruebas

<https://pastebin.com/ZjYeNWgv>



Dándole forma a nuestro port

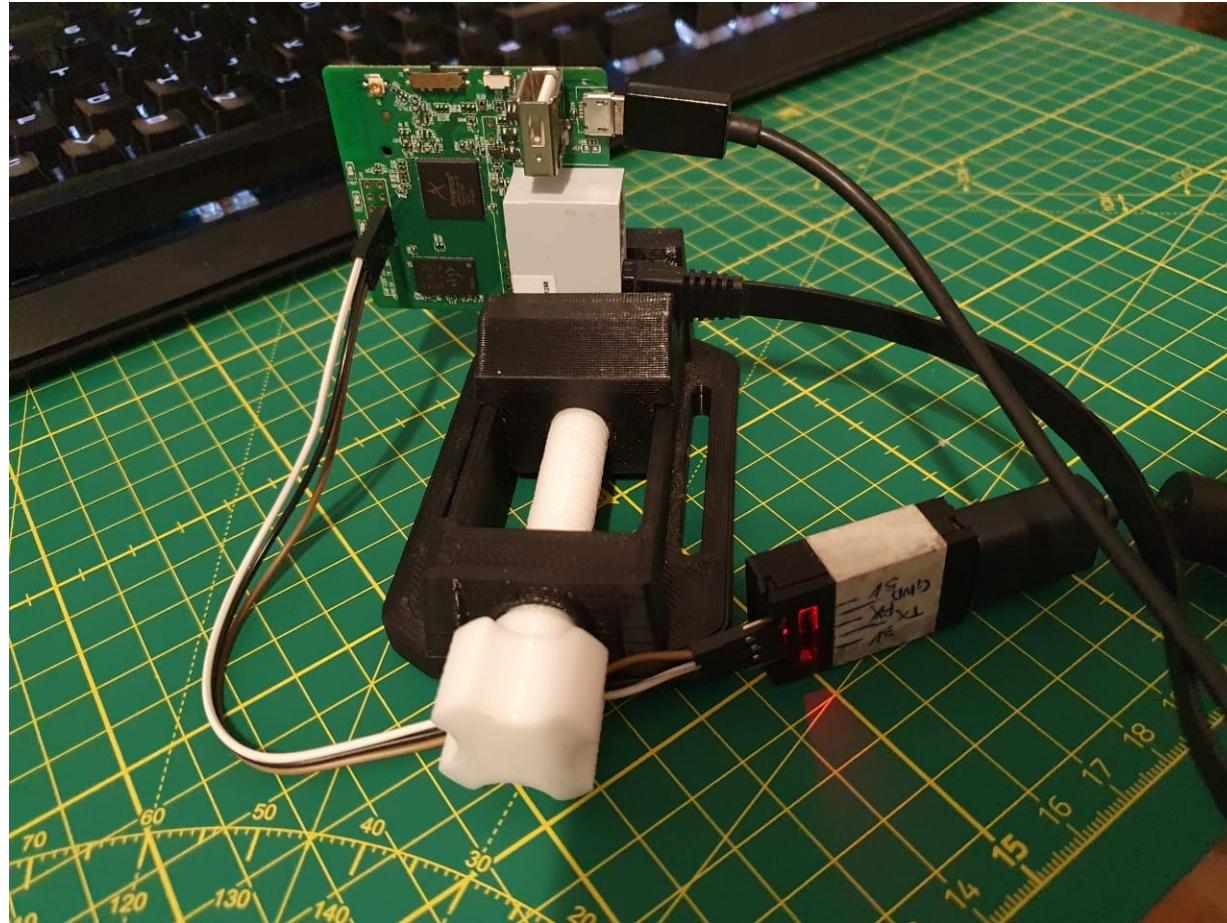


Ahora vamos uniendo este Frankenstein que terminará siendo nuestro port.

1. Con Firmware Mod Kit vamos a desempaquetar el sysupgrade.bin del build que hicimos en OpenWrt y tambien el del update de la NANO.
2. Nos vamos a quedar con el kernel del build que hicimos que es el que soportaria el nuevo board. Lo podemos encontrar en esta ruta image_parts/header.img
3. Este lo unimos con el file system original para hacer nuestro nuevo firmware.



Puliendo nuestro port



Esta etapa es un loop de hacer el firmware, fleshear el hardware y ver por UART que está todo bien. Básicamente se trata de ir probando todas las funcionalidades y ver que no rompan.

Este es el momento es donde tenemos que buscar y saltar cualquier chequeo por hard que existiera.

Seguramente se tenga que modificar algunas cosas del file system original para que encajen en el hard nuevo. Para esto es buena idea pegarle una mirada a estas carpetas:

/etc
/etc/uci-defaults
/lib/preinit



¿No podemos hacerlo mas rápido?

Planificando un nuevo acercamiento



Conclusiones del acercamiento actual

- Es simple de hacer
- Tener que compilar el kernel puede asustar un poco
- Se nota un poco inestable el build



Ideando un nuevo metodo

Planificando un nuevo acercamiento

upgrade-2.7.0 - openwrt-19.07.2-gl-ar150\		
C:\Users\DSR\Desktop\eko-ws\2021_winmerge\upgrade-2.7.0\	C:\Users\DSR\Desktop\eko-ws\2021_winmerge\openwrt-19.07.2-gl-	
Filename	Folder	Comparison result
> bin		Folders are different
> dev		Identical
etc		Folders are different
etc	etc	Folders are different
01_leds	etc\board.d	Text files are different
02_network	etc\board.d	Text files are different
03_gpio_switches	etc\board.d	Text files are different
99-default_network	etc\board.d	Text files are identical
> config	etc	Folders are different
crontabs	etc	Identical
> dropbear	etc	Right only: C:\Users\DSR\Desktop\eko-ws\2021_winmerge\openwrt-19.07.2-gl-ar150\etc
> hotplug.d	etc	Folders are different
> init.d	etc	Folders are different
iproute2	etc	Identical
luci-uploads	etc	Right only: C:\Users\DSR\Desktop\eko-ws\2021_winmerge\openwrt-19.07.2-gl-ar150\etc
modules-boot.d	etc	Folders are different
modules.d	etc	Folders are different
nginx	etc	Left only: C:\Users\DSR\Desktop\eko-ws\2021_winmerge\upgrade-2.7.0\etc
opkg	etc	Folders are different
php7	etc	Left only: C:\Users\DSR\Desktop\eko-ws\2021_winmerge\upgrade-2.7.0\etc
php7-fpm.d	etc	Left only: C:\Users\DSR\Desktop\eko-ws\2021_winmerge\upgrade-2.7.0\etc
pineape	etc	Left only: C:\Users\DSR\Desktop\eko-ws\2021_winmerge\upgrade-2.7.0\etc

Objetivos

- Evitar tener que hacer un build completo
- Solucionar problema de estabilidad

¿Cómo lo logró?

- Usando el imagebuilder oficial de OpenWrt
- Instalo todo basandome status log de opkg
- Me copio las diferencias y configs por diff

Referencias:

<https://github.com/xchwarze/wifi-pineapple-cloner>



Haciendo nuestro build

```

dsr@ubuntu: ~/Desktop/nano/openwrt-imagebuilder-19.07.2-ar71xx-generic.Linux-x86_64
File Edit View Search Terminal Help

dsr@ubuntu:~/Desktop/nano/openwrt-imagebuilder-19.07.2-ar71xx-generic.Linux-x86_64$ make info
Current Target: "ar71xx/generic"
Current Revision: "r10947-65030d81f3"
Default Packages: base-files libc libgcc busybox dropbear mtd uci opkg netifd fstools uclient-fetch logd
    urandom-seed urngd kmod-gpio-button-hotplug swconfig kmod-ath9k uboot-envtools wpad-basic dnsmasq iptables
    ip6tables ppp ppp-mod-pppoe firewall odhcpd-ipv6only odhcp6c kmod-ipt-offload
Available Profiles:

Default:
  Default Profile (all drivers)
  Packages: kmod-usb-core kmod-usb-ohci kmod-usb2 kmod-usb-ledtrig-usbport iwinfo
  hasImageMetadata: 0
carambola2:
  8devices Carambola2
  Packages: kmod-usb-core kmod-usb2
  hasImageMetadata: 0
lima:
  8devices Lima
  Packages: kmod-usb-core kmod-usb2
  hasImageMetadata: 0
ALFAAP120C:
  ALFA Network AP120C board
  Packages:
  hasImageMetadata: 0
ap121f:
  ALFA Network AP121F
  Packages: kmod-usb-core kmod-usb2 kmod-usb-storage -swconfig
  hasImageMetadata: 1
  SupportedDevices: ap121f
ALFAAP96:
  .
  .
  .

```

Pasos de nuestro nuevo método

1. Bajamos el image builder para la plataforma ar71xx
2. Generamos la lista de dependencias que vamos a instalarle con opkg_statusdb_parser.php
3. Basados en el diff preparamos el file system
4. Hacemos con nuestra build:
make image PROFILE=gl-ar150 PACKAGES="at autossch base-files... -wpad-basic-dropbear" FILES=../filesystem/

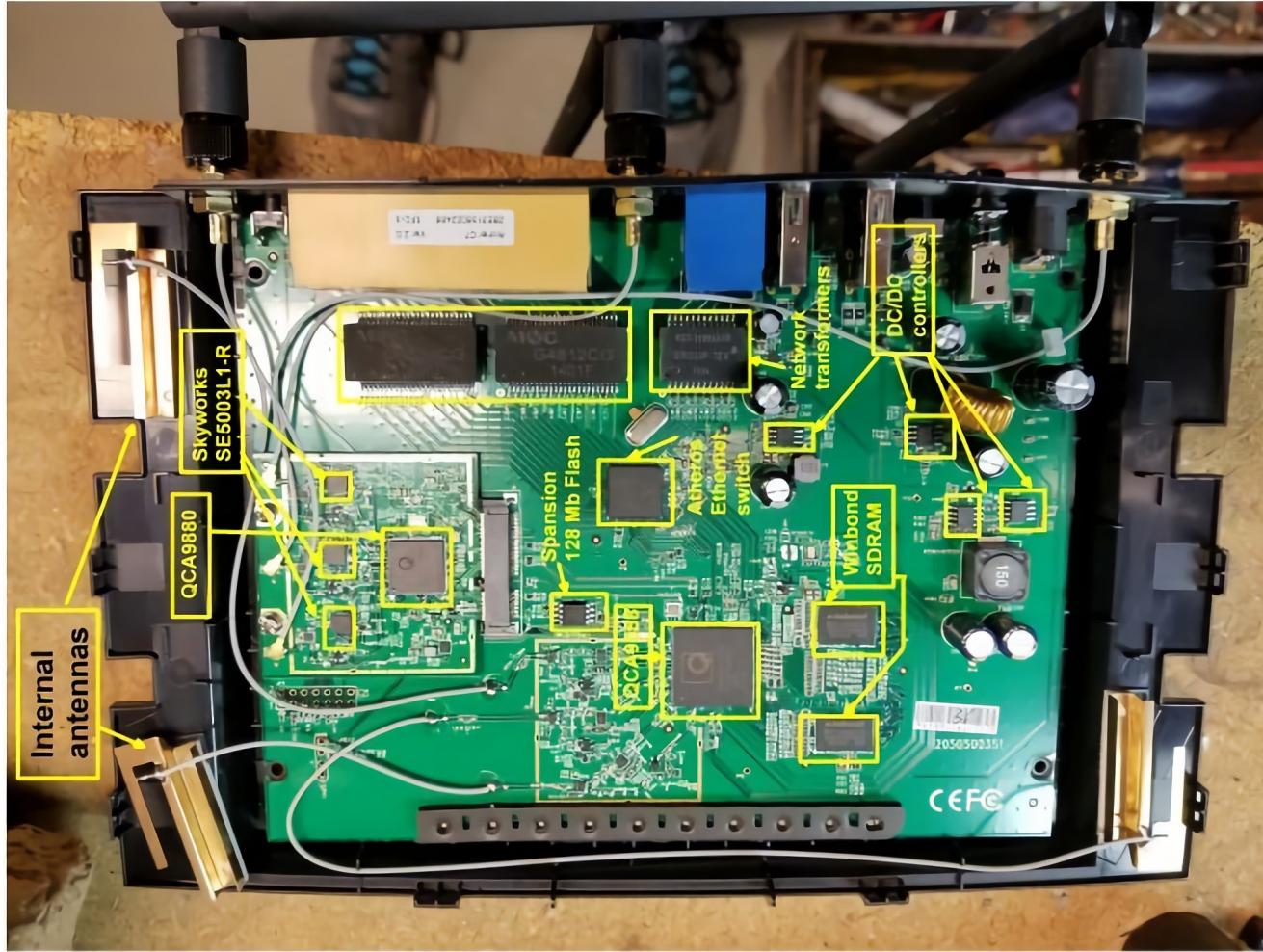
Referencias:

<https://github.com/xchwarze/wifi-pineapple-cloner>



¿Y si no conseguimos un hard anfitrión similar?

¿O si quiero portear una Tetra...?

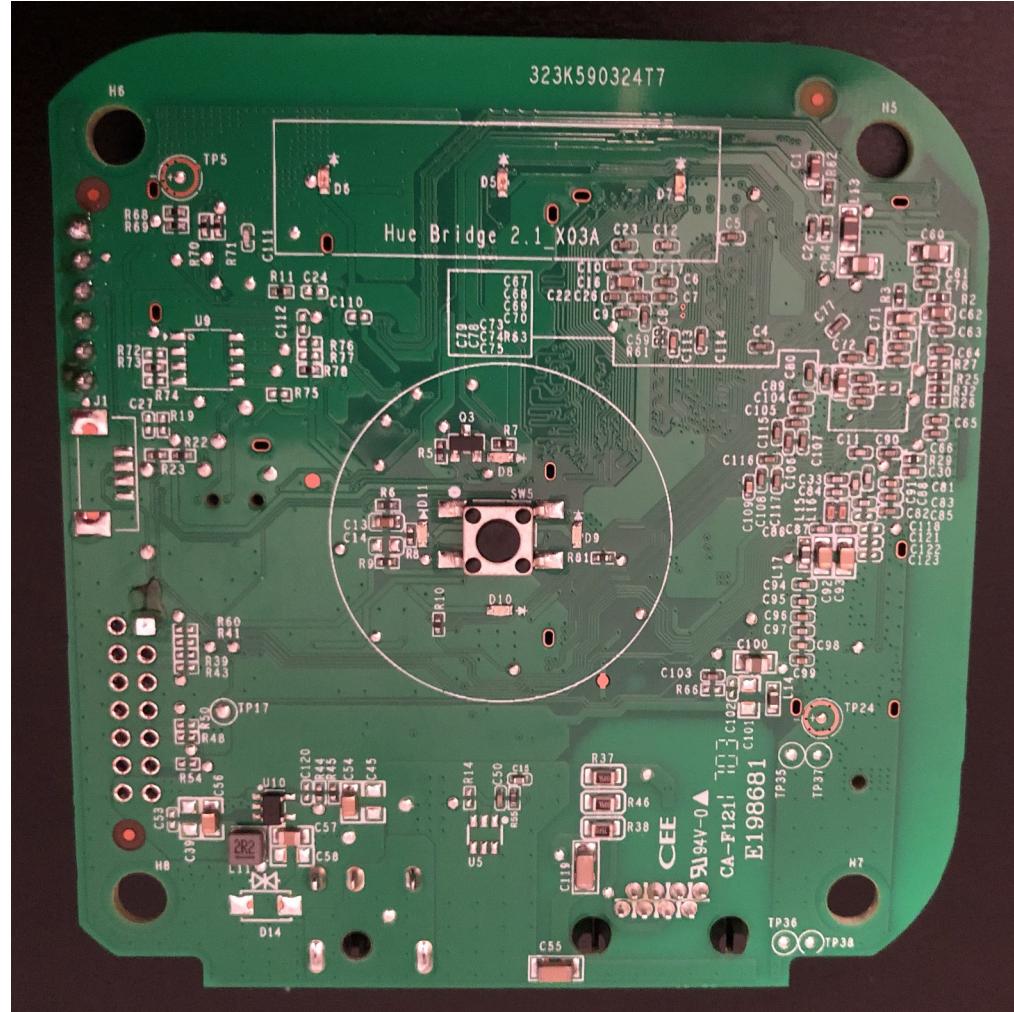


Este es el Archer C7, el hardware (v5) esta conformado por:

- CPU: 750 MHz MIPS Qualcomm Atheros QCA9563 SoC
- Architecture MIPS32 74Kc
- Endianness: BE
- Memory: 128 MB DDR2 RAM
- Disk: 16 MB ROM
- WiFi: 1- QCA9563 (built-in), 1- QCA9880 (5Ghz)
- Switch 5x 10/100/1000 Mbps Ethernet
- Misc: 1 - USB 2.0, 10 - status LED, 2 buttons



Conclusiones



Con el porting podemos:

- Usar hardware que no tenemos
- Mejorar el hardware en el que corre el software
- Seguir manteniendo hardware abandonado
- Agregar funcionalidades en hardware que tenemos *
- Usar hardware para cosas que no fueron pensadas **

Referencias:

- * <https://blog.andreianaru.ro/2018/03/27/philips-hue-2-1-enabling-wifi/>
- ** <https://www.armbian.com/rk322x-tv-box/>



¡Gracias por participar!

Si quieren ver más cosas de seguridad informática y desarrollo
están invitados a pasarse por la comunidad!

[Comunidad - Discord](#)