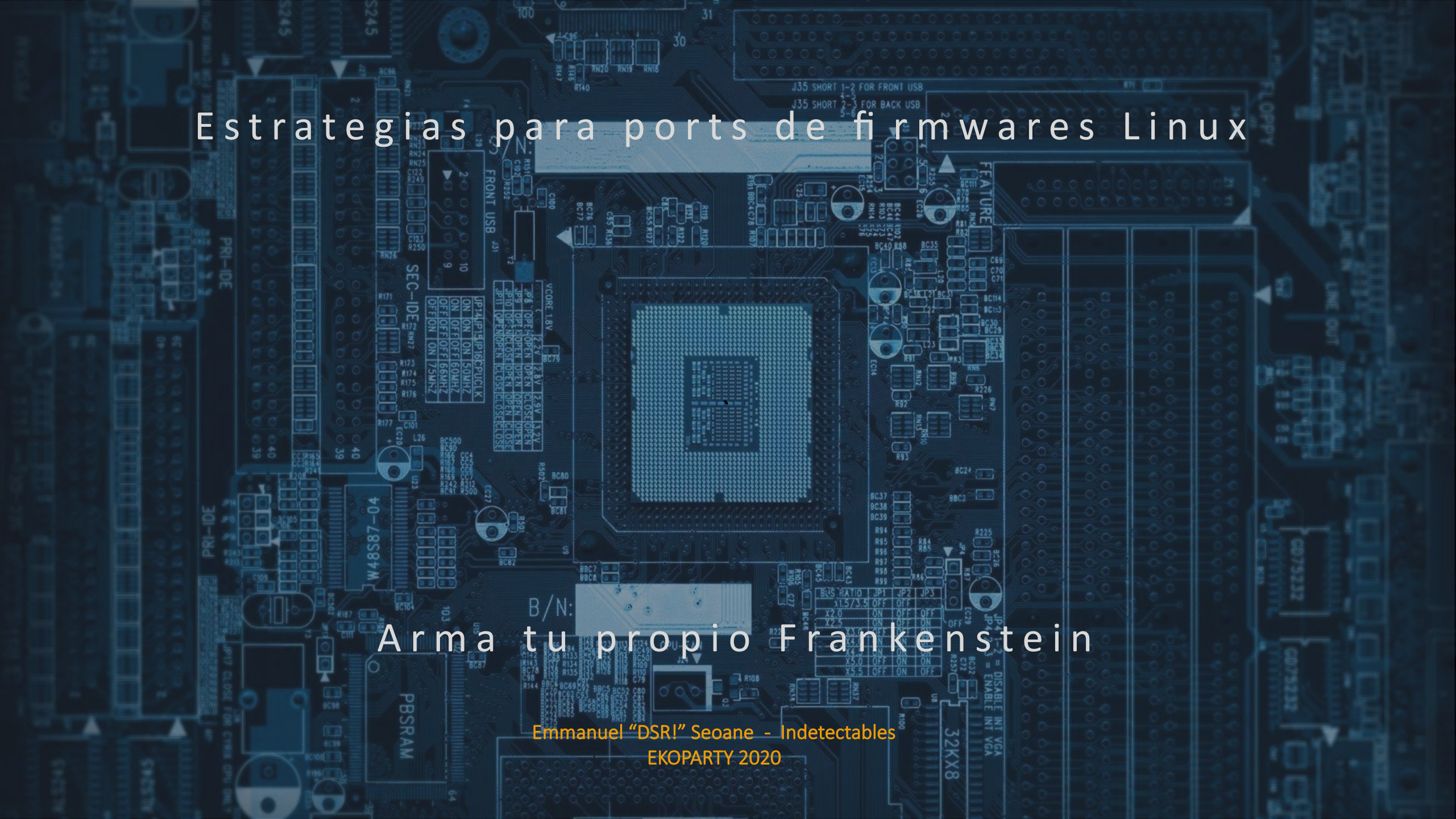


Estrategias para ports de firmwares Linux

Arma tu propio Frankenstein

Emmanuel “DSR!” Seoane - Indetectables

EKOPARTY 2020

Emmanuel Seoane

DSR!

Desarrollador de Software, líder de equipo y Hacker los fines de semana.
Formo parte del Staff de Indetectables.net donde vengo rompiendo cosas desde hace más de 10 años.
Es menos frecuente pero también a veces arreglo cosas.

¿Que es un Port?

Quien diria que no está en la RAE

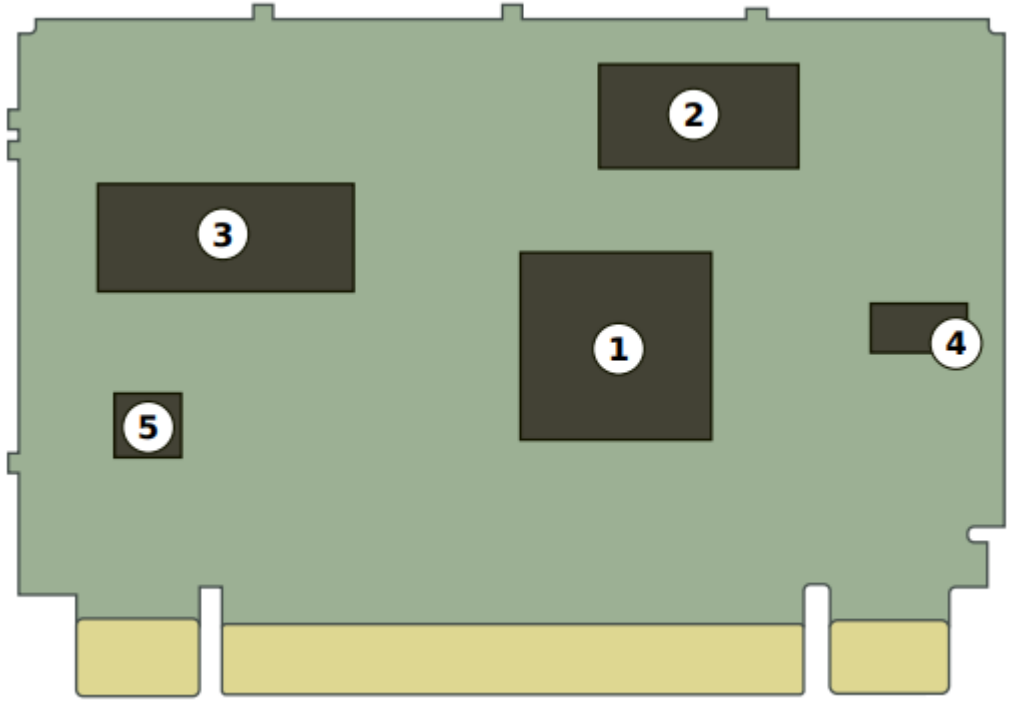
Se le llama "**port**" al proceso de adaptar software con el fin de lograr alguna forma de ejecución en un entorno informático que es diferente de aquel para el que se diseñó originalmente.
El término también se usa cuando se cambia el software / hardware para hacerlos utilizables en diferentes entornos.
El software es portable cuando el costo de trasladarlo a una nueva plataforma es **significativamente menor que el costo de escribirlo desde cero.**

El típico ejemplo de esto es el DOOM y sus múltiples ports, algunos como el de SNES era considerado imposible de hacer.

Referencias:
https://en.wikipedia.org/wiki/Porting
https://youtu.be/JqP3ZzWiul0

¿Por qué hacer un port?

Originalmente el principal interesado en esto eran los desarrolladores y las empresas pero hoy en día no sería la regla.
Esta técnica es cada vez más usada por la comunidad maker.



- 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**

Ejemplo practico

Eligiendo algún hardware para jugar

Para hacer las cosas más entretenidas vamos a tomar de ejemplo la Pineapple NANO.
El primer paso sería….

Reconociendo el hardware y software para encontrar candidatos para suplirlo

El **hardware** esta conformado por:

- CPU: 400 MHz MIPS Atheros AR9331 SoC
- Memory: 64 MB DDR2 RAM
- Disk: 16 MB ROM

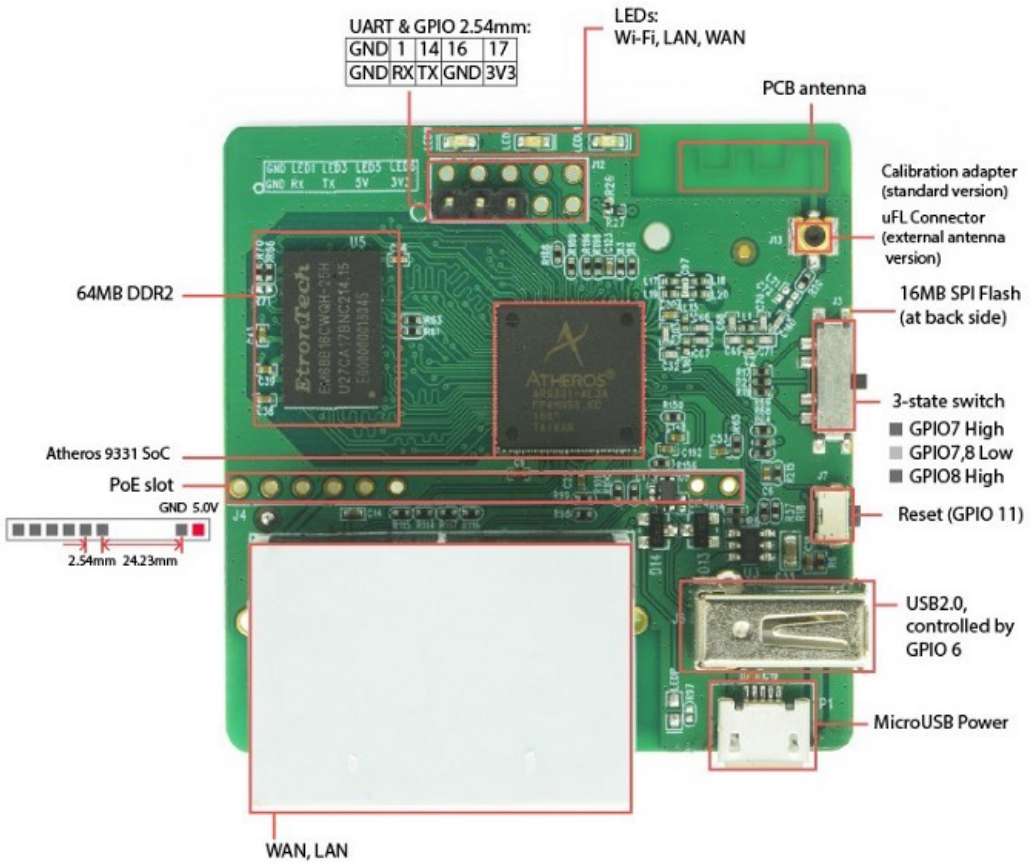
El **software** podemos darle una mirada usando "**Firmware Mod Kit**" para extraer el filesystem de un update y explorarlo prestando atención a estas rutas:

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

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

Candidato encontrado!

¿Tan rápido?

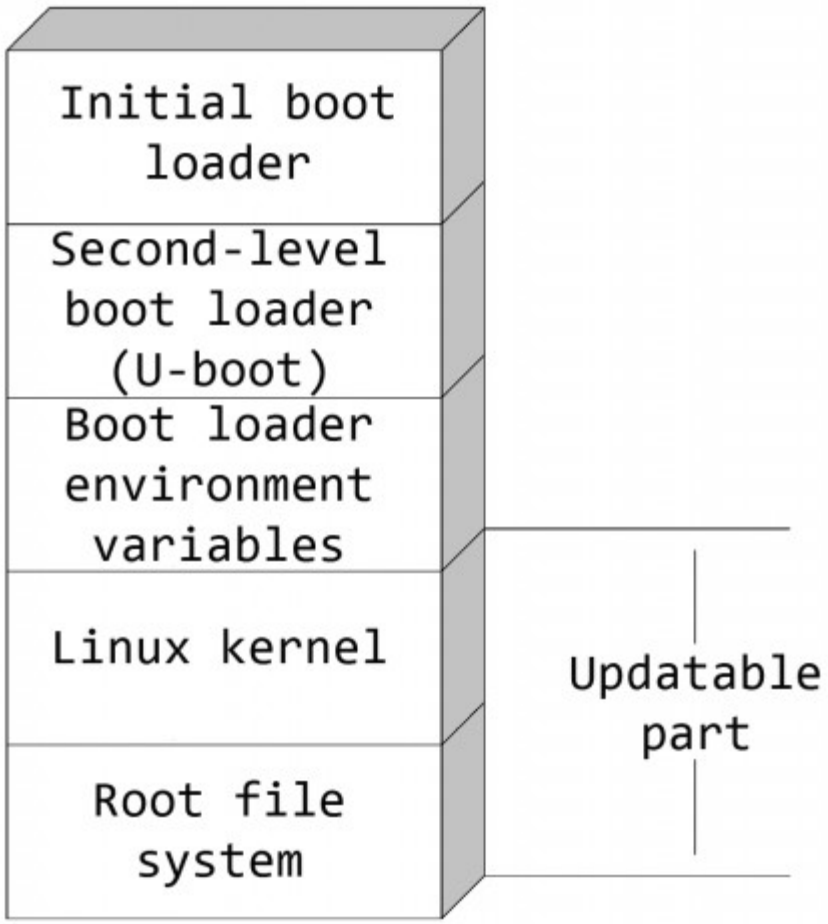


Afortunadamente las características son idénticas al GL-iNet AR150, un router pensado para aplicaciones IoT.

Teniendo nuestro candidato elegido lo ideal es buscar si alguien ya lo intentó y ver qué podemos aprender de su experiencia. Como la NANO es un hard que existe hace tiempo ya habían intentos de port hacia esta plataforma pero además de no ser muy estables por cómo están hechos tendían a limitar mucho la overlay partition que es el espacio libre que queda en la memoria flash para los archivos dinámicos del usuario.

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



Ahora llegado este punto es vital explicar un poco esto.

Típicamente este tipo de dispositivos tienen esta estructura, donde el bootloader termina cargando un kernel linux que interactúa con el file system. Normalmente esa última parte es la que es actualizable.
Por lo menos esta sería la forma más rápida y concisa de explicarlo para no extenderme.

Firmware Mod Kit nos ayuda a trabajar con esa parte actualizable desenpaquetandola para análisis o hasta si queremos podemos modificarla y volverla a empaquetar. Vamos a hablar sobre esto más adelante.

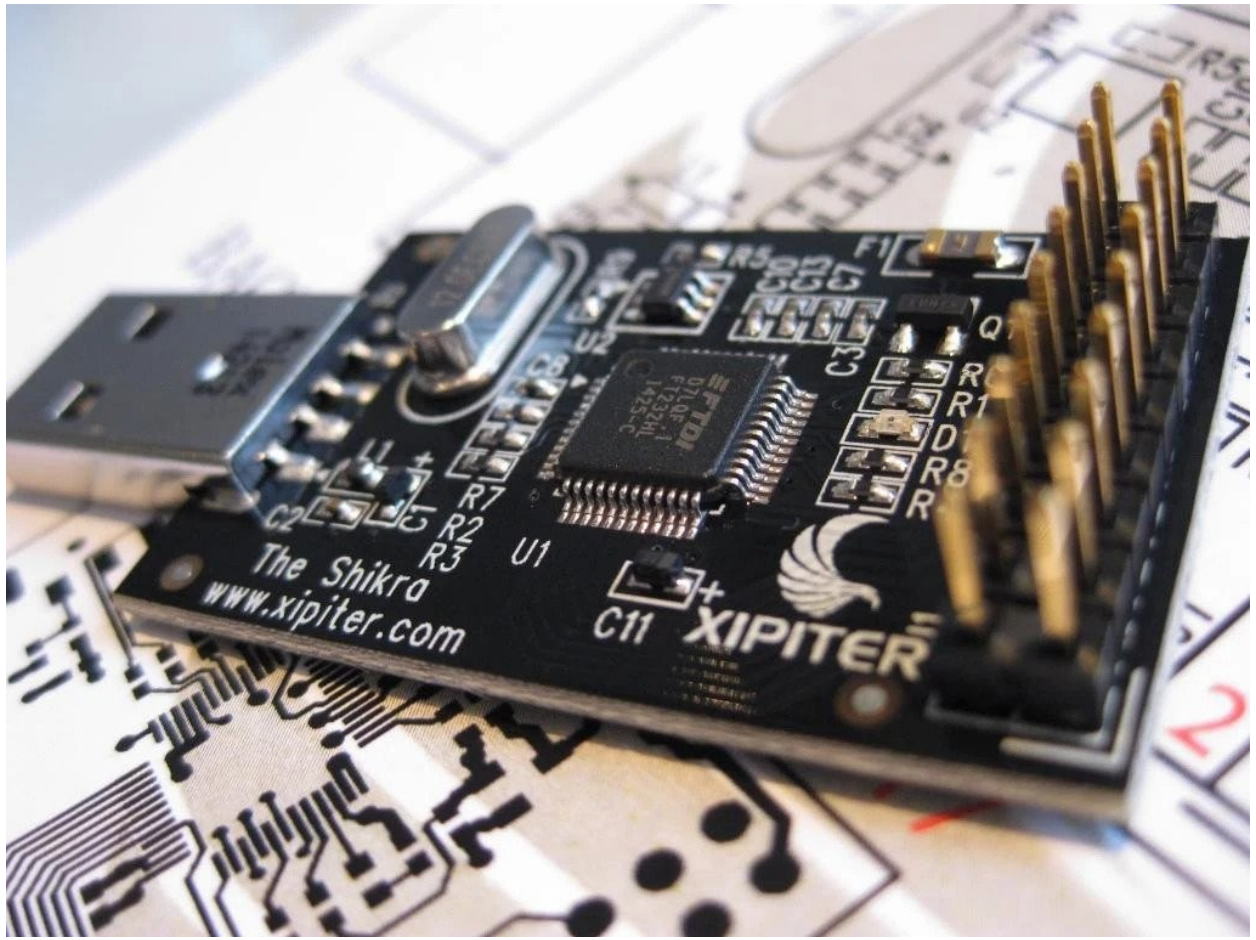
Planificando mi acercamiento

Entendiendo como se conforma un firmware y que cosas hay en un file system me di cuenta que la mejor manera era dejar este último lo más intacto posible para no acortar la overlay partition.

Para esto decidí hacer un kernel idéntico al usado en el equipo original pero que soporte mi board, cosa que al ser tanto el firmware original como el del AR150 basados en el SoC AR9331 y OpenWrt tendría que ser sencillo.

O por lo menos eso era lo que esperaba...

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 dolares.

Herramientas para unpack/pack de FW

Para esto podemos usar Firmware Mod Kit

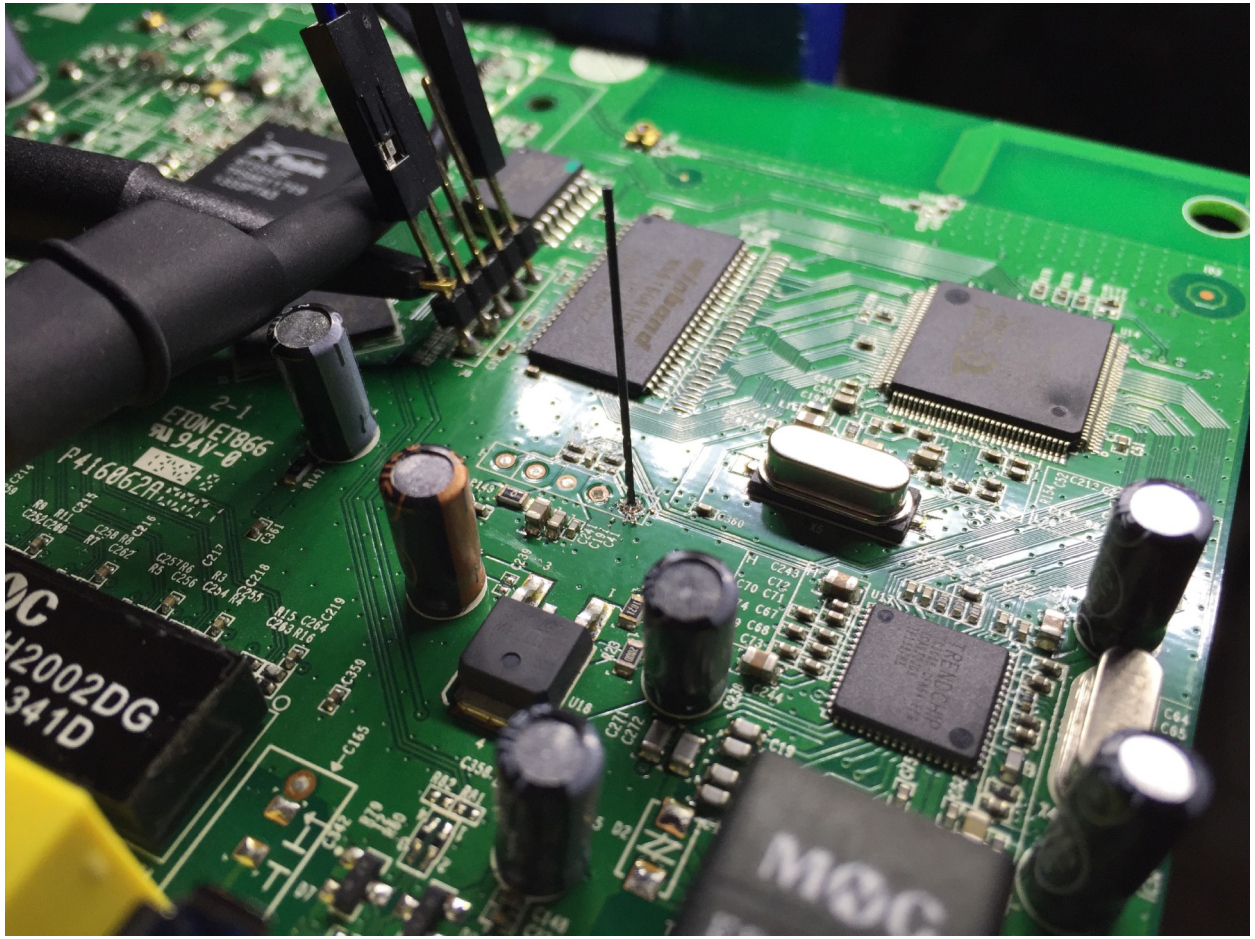
Decompilador para MIPS/ARM

Acá es donde entra Ghidra al rescate

Herramientas basicas de electronica

Multimetro, destornilladores, púas, soldador, etc

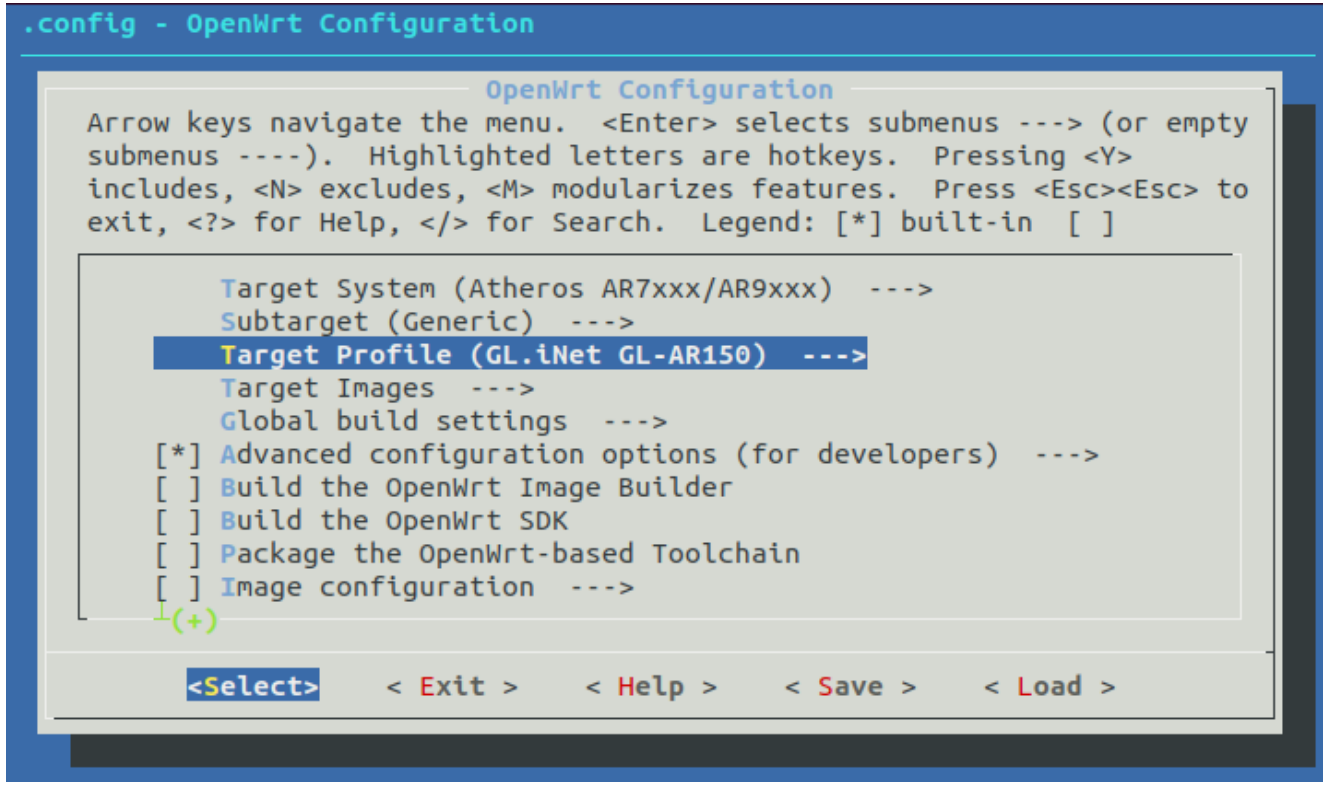
Preparando el terreno



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

Realmente tener esto listo y mucha paciencia es algo vital a la hora de hacer un port que podríamos llamar a ciegas. Porque después de todo no tenemos documentación ni el código fuente de nada por lo que puede resultar bastante engorroso debuggear.

Haciendo mi kernel



Cuando analizamos el software llegamos a lo que usaron de base y que versión, en este caso es OpenWrt y hasta sabemos qué es la última versión y en líneas generales con que cosas hicieron el build.

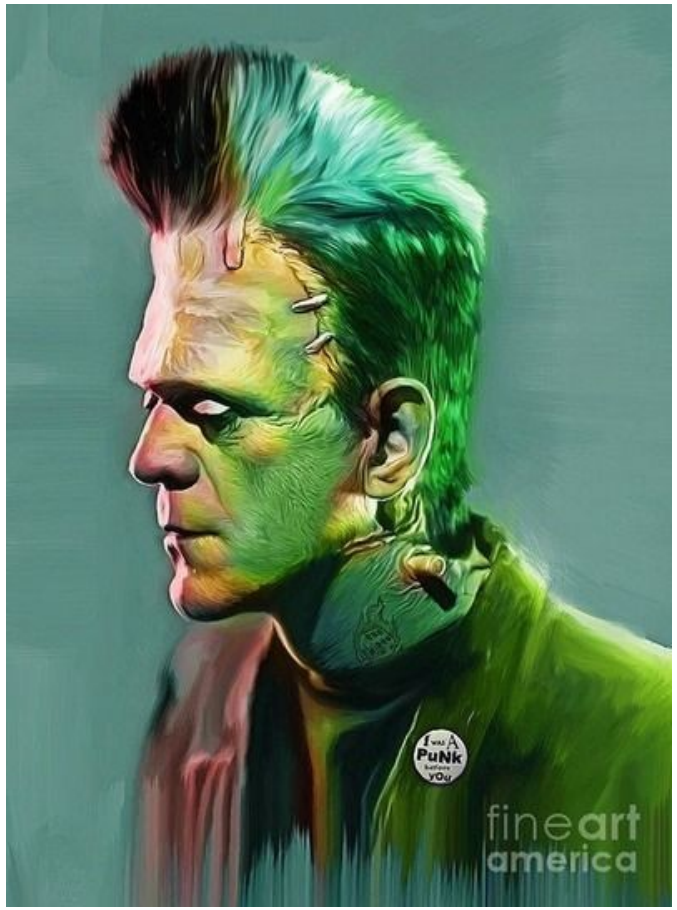
Así que este paso consta en hacer un build lo más similar al original para que funcione todo.

De todas formas, hay que tener en cuenta que el desarrollador pudo haber hecho cambios y no haberlos publicado a pesar de estar obligado por las licencias de uso de gpl.

Esto es bastante común así que hay que tenerlo en cuenta!

Referencias:
Les dejo el .config que use para mis pruebas
https://pastebin.com/ZjYeNWgv

Dándole forma a mi port



Ahora vamos uniendo nuestro Frankenstein que terminará siendo nuestro port.

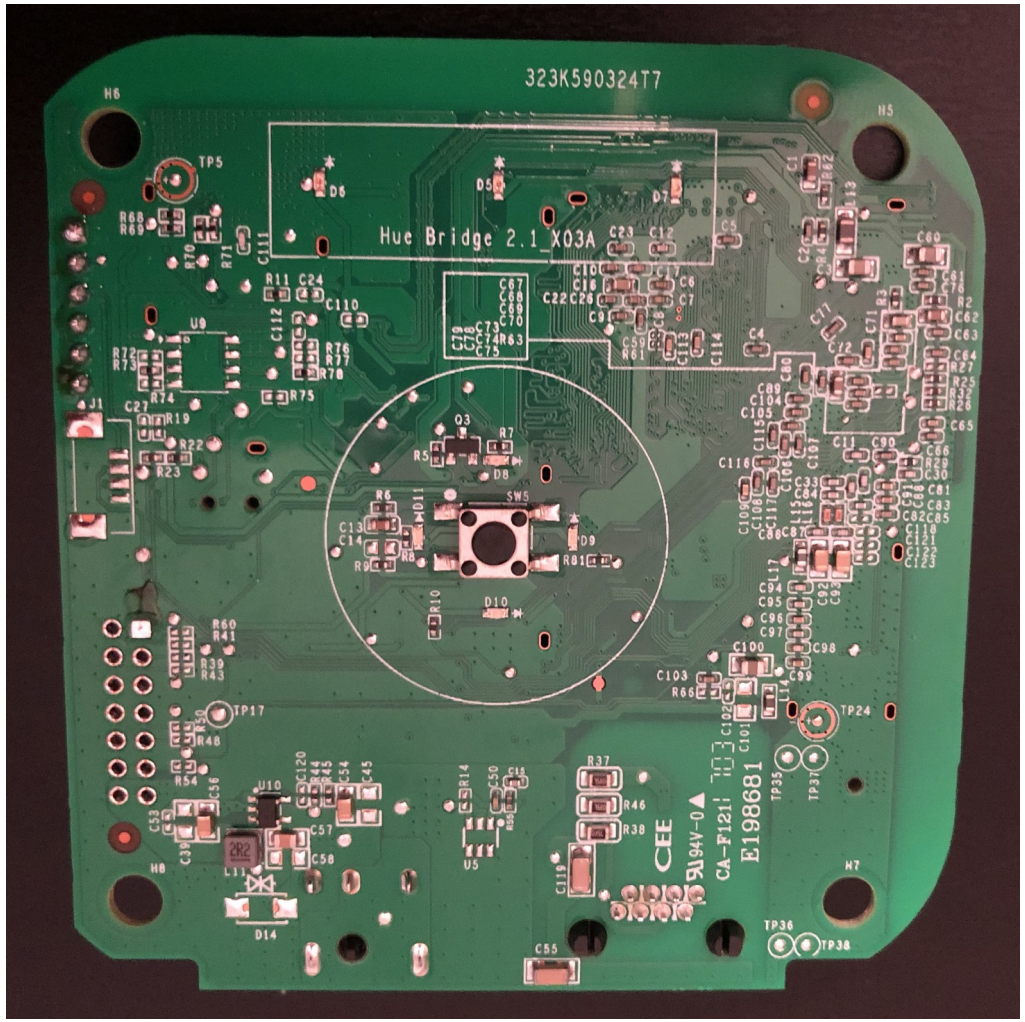
Con Firmware Mod Kit vamos a desempaquetar el sysupgrade.bin del build que hicimos de OpenWrt en el paso anterior y nos vamos a quedar con el kernel que estaría en esta ruta image_parts/header.img. Y este lo unimos con el filesystem original para hacer nuestro nuevo firmware.

Esta etapa de hacer el firmware, flashear el hardware y ver por UART que está todo bien nos puede tomar bastante tiempo porque hay que ir iterando y mejorando el port hasta que quede funcionando igual o mejor que en el hard original

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

¿Y ahora?



Básicamente es perseverar en el punto anterior hasta que quede el port completo. También está la posibilidad que el mismo no se pueda realizar o que sea mucho más difícil de lo que imaginamos.
Y ahí es donde entra cuánto tiempo queremos invertir en ello.

Igualmente esta misma técnica se puede usar para modificar firmwares para agregarles funcionalidades o solucionar fallas. Por ejemplo se usa bastante para modificar cámaras de seguridad y en artefactos IOT donde se los suele modificar para agregarles soporte para MQTT.
También está el caso del bridge de Philips Hue que se modifica para tener soporte de WIFI.

Referencias:
https://blog.andreibanaru.ro/2018/03/27/philips-hue-2-1-enabling-wifi/

¡Gracias por participar!

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