Sistemas embebidos

Un sistema embebido, también conocido como sistema empotrado, es un sistema electrónico diseñado para realizar tareas específicas y se integra en un dispositivo más grande para mejorar su funcionalidad. Estos sistemas están diseñados para funcionar con un conjunto limitado de recursos y hardware, lo que los hace altamente eficientes y rentables.

En general, un sistema embebido consta de una o varias unidades centrales de procesamiento (CPU), una memoria, dispositivos de entrada y salida, y a menudo una fuente de alimentación integrada. Estos sistemas están diseñados para funcionar con una variedad de dispositivos, desde electrodomésticos y sistemas de control de clima hasta automóviles y aviones.

A diferencia de los sistemas de propósito general, como las computadoras personales, los sistemas embebidos están diseñados para realizar tareas específicas de manera eficiente y confiable. Los sistemas embebidos se utilizan en una amplia variedad de aplicaciones, como en la industria, el transporte, la electrónica de consumo, la medicina, la defensa y la seguridad, entre otros.

Uno de los principales beneficios de los sistemas embebidos es que son altamente personalizables y se pueden adaptar a una amplia variedad de aplicaciones. Los sistemas embebidos también pueden ser muy eficientes en términos de energía y recursos, lo que los hace ideales para su uso en dispositivos portátiles y baterías.

En resumen, un sistema embebido es un sistema electrónico integrado en un dispositivo más grande diseñado para realizar tareas específicas. Estos sistemas son altamente eficientes, personalizables y se utilizan en una amplia variedad de aplicaciones en todo el mundo.

Diferencias entre SoC y microcontrolador (ejemplo con raspberry pi y arduino)

Una Raspberry Pi es una computadora de placa única (SBC) que se utiliza comúnmente para proyectos de informática y electrónica. La Raspberry Pi es un dispositivo más potente que un Arduino, con más capacidades de procesamiento y memoria. La Raspberry Pi funciona con un sistema operativo completo y se puede programar en varios lenguajes de programación como Python y C ++. La Raspberry Pi se utiliza comúnmente para proyectos de IoT, domótica, servidores de medios y proyectos educativos.

Por otro lado, Arduino es una plataforma de hardware y software de código abierto diseñada para la programación y el control de dispositivos electrónicos. Los microcontroladores de Arduino son dispositivos pequeños y económicos que pueden ser programados para realizar una amplia variedad de tareas. Arduino se utiliza comúnmente para proyectos de electrónica, robótica y automatización.

La principal diferencia entre la Raspberry Pi y el Arduino es que la Raspberry Pi es una computadora completa, mientras que Arduino es un microcontrolador programable. La Raspberry Pi

se utiliza principalmente para proyectos de informática y electrónica, mientras que Arduino se utiliza para proyectos de electrónica y robótica.

Firmware para SoC

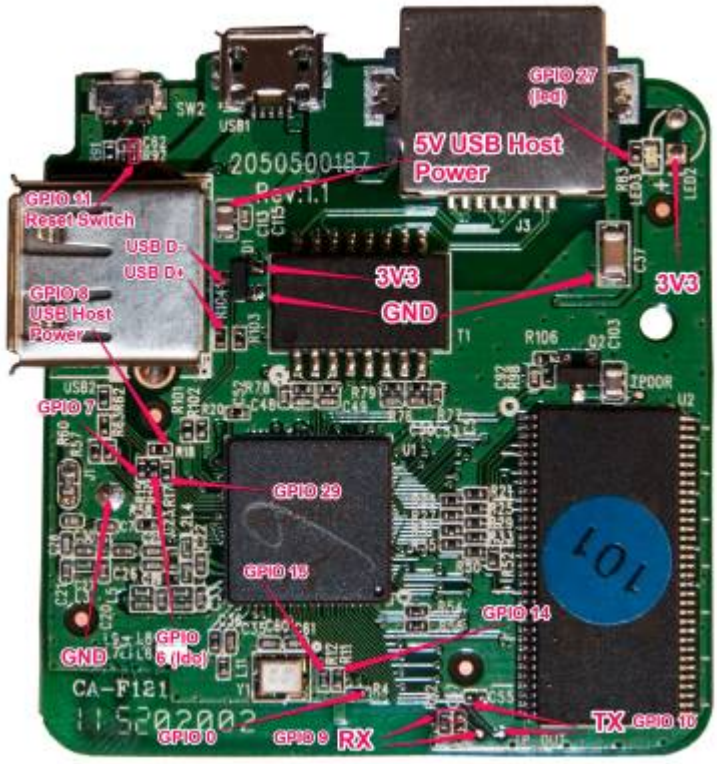
Existen diversas distribuciones de software para sistemas embebidos, cada una diseñada para un conjunto específico de necesidades y requisitos. Aquí te presento algunas de las distribuciones más relevantes:

1. OpenWRT: es una distribución de Linux altamente personalizable diseñada para enrutadores de red y otros dispositivos de red. OpenWRT está optimizado para funcionar en dispositivos con recursos limitados, como RAM y almacenamiento. Además, OpenWRT ofrece una gran cantidad de paquetes de software para agregar nuevas funcionalidades al dispositivo.
2. Yocto Project: es una distribución de Linux altamente personalizable y adaptable diseñada para sistemas embebidos. El Yocto Project se enfoca en la creación de imágenes personalizadas para sistemas embebidos a partir de un conjunto de componentes específicos. Esta distribución es altamente flexible y escalable, permitiendo la creación de sistemas embebidos para una amplia gama de aplicaciones.
3. FreeRTOS: es un sistema operativo en tiempo real de código abierto diseñado para sistemas embebidos y sistemas con recursos limitados. FreeRTOS es altamente modular y permite la ejecución de múltiples tareas en un solo sistema. Además, cuenta con una amplia gama de características de seguridad y conectividad.
4. VxWorks: es un sistema operativo en tiempo real desarrollado por Wind River. VxWorks es conocido por su alto rendimiento, confiabilidad y seguridad, y se utiliza comúnmente en aplicaciones críticas para la seguridad, como la aviación y la industria automotriz. Además, VxWorks es altamente personalizable y escalable, lo que lo hace adecuado para una amplia gama de aplicaciones.

En resumen, cada una de estas distribuciones de software para sistemas embebidos ofrece características y funcionalidades específicas para satisfacer las necesidades de los diseñadores y desarrolladores de sistemas embebidos.

Qué se puede hacer con un SoC (Ej Router)

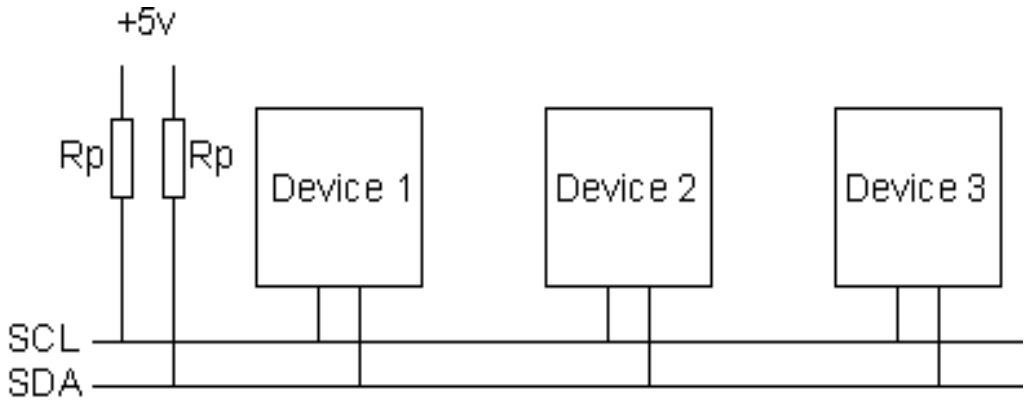
Partes de un SoC



Puertos de comunicaciones en sistemas embebidos

I2C, I2S, UART y GPIO son diferentes protocolos de comunicación utilizados en sistemas embebidos y dispositivos electrónicos para transferir datos y controlar periféricos. A continuación se describe cada uno de ellos:

1. I2C (Inter-Integrated Circuit): Es un protocolo de comunicación en serie utilizado para interconectar dispositivos electrónicos en una placa de circuito impreso (PCB). I2C utiliza solo dos cables, SDA (Serial Data) y SCL (Serial Clock), para transmitir datos en ambas direcciones. I2C se utiliza comúnmente para comunicarse con sensores, dispositivos de memoria, convertidores de analógico a digital, entre otros.

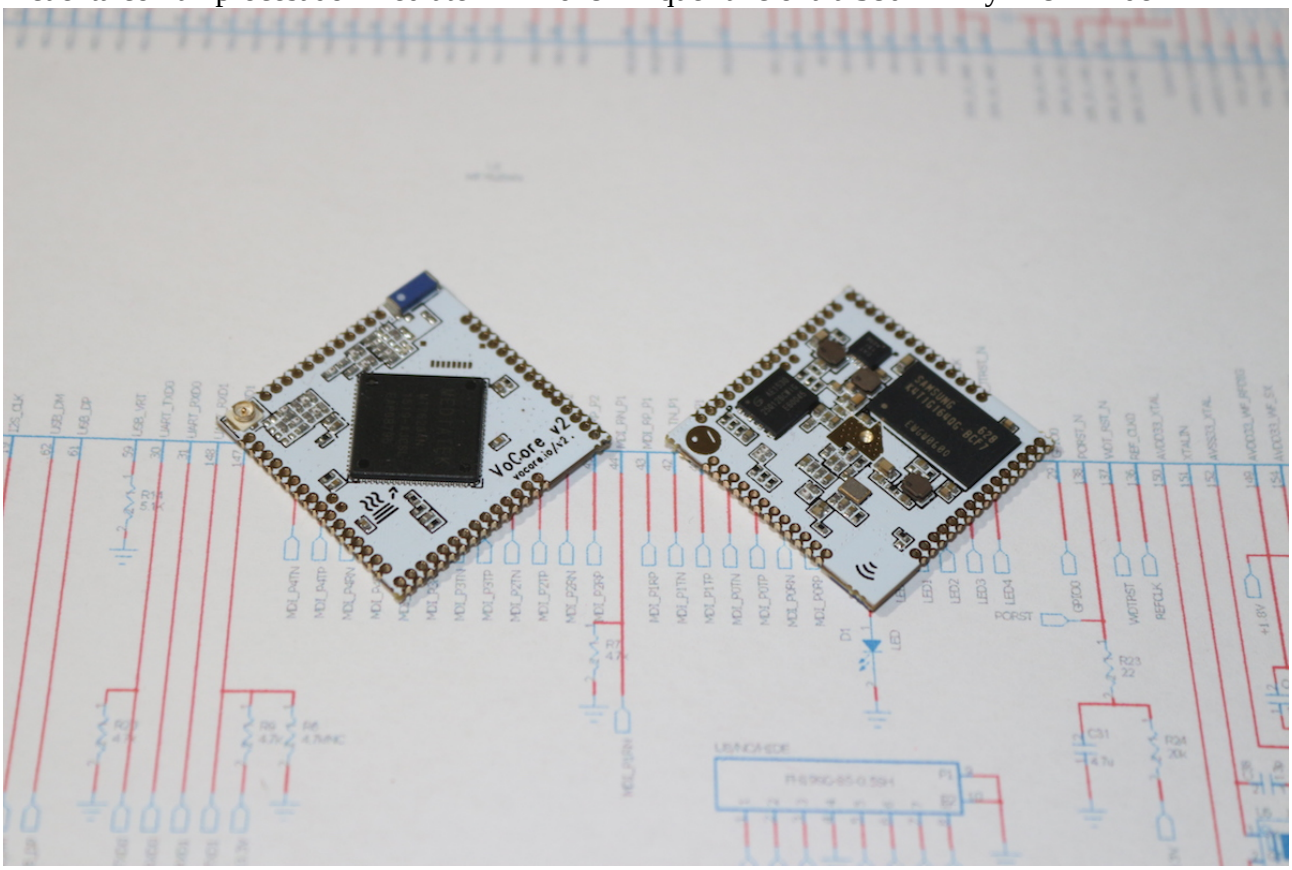


2. I2S (Inter-IC Sound): Es un protocolo de comunicación utilizado para transmitir señales de audio digital entre dispositivos. I2S utiliza tres líneas de comunicación: Serial Data, Serial Clock y Word Select, y se utiliza comúnmente en dispositivos de audio como DACs (Digital to Analog Converters), ADCs (Analog to Digital Converters) y amplificadores de audio.
3. UART (Universal Asynchronous Receiver-Transmitter): Es un protocolo de comunicación serie utilizado para la transmisión de datos en serie entre dispositivos. UART utiliza dos cables, uno para transmitir datos (TX) y otro para recibir datos (RX). Se utiliza comúnmente en dispositivos embebidos para la comunicación serie con otros dispositivos, como módems, GPS, sensores, entre otros.
4. GPIO (General Purpose Input/Output): Es un conjunto de pines en un microcontrolador o en una placa de circuito impreso que se pueden configurar como entradas o salidas digitales.

Las entradas se utilizan para leer señales digitales de dispositivos externos, mientras que las salidas se utilizan para enviar señales digitales a dispositivos externos. Los pines GPIO se utilizan comúnmente para conectar botones, LEDs, motores, sensores, etc.

Vocore2 (https://vocore.io/v2.html)

La placa VoCore2 es una placa de desarrollo de código abierto, de tamaño pequeño y bajo costo que se puede utilizar para crear una variedad de dispositivos de IoT (Internet de las cosas). La placa cuenta con un procesador Mediatek MT7628AN que funciona a 580 MHz y 128 MB de RAM



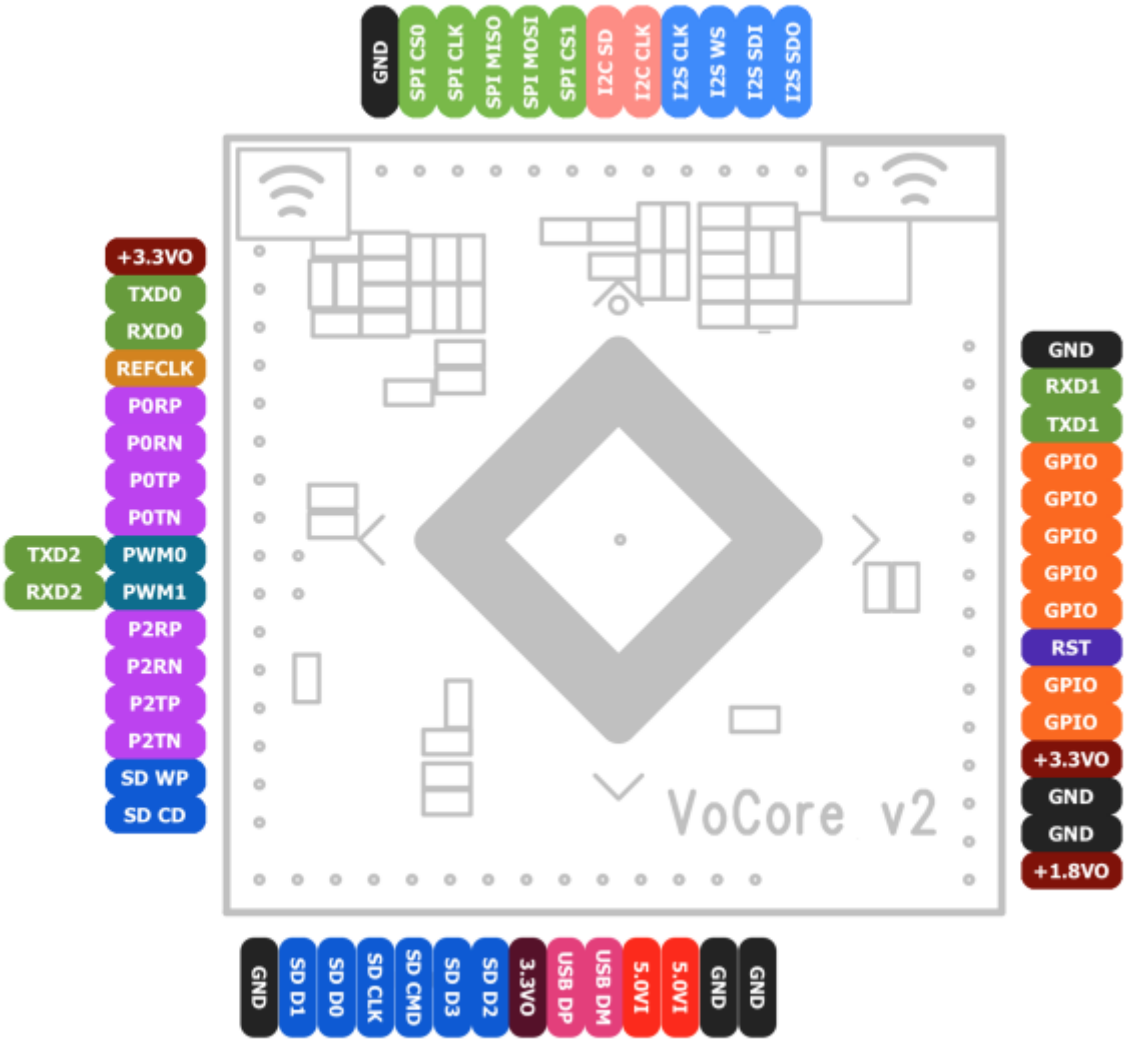
DDR2. Además, tiene 16 MB de memoria flash incorporada para el almacenamiento de programas y datos.

La VoCore2 cuenta con conectividad inalámbrica Wi-Fi 802.11 b/g/n, lo que permite la comunicación sin cables con otros dispositivos de red. También tiene varios puertos de entrada y salida, incluyendo un puerto Ethernet, USB 2.0, I2C, SPI, UART, GPIO y un conector de antena externa.

La placa VoCore2 es compatible con una variedad de sistemas operativos de código abierto, como OpenWrt, LEDE y Ubuntu Core. Estos sistemas operativos ofrecen una amplia gama de herramientas y paquetes de software para el desarrollo de aplicaciones de IoT.

En resumen, la placa VoCore2 es una plataforma de desarrollo de bajo costo, de tamaño pequeño y con amplia conectividad inalámbrica que permite la creación de dispositivos de IoT de forma fácil y rápida.

	Details
SIZE	25.6mm x 25.6mm x 3.0mm
CPU	**MT7628, 580 MHz, MIPS 24K**
MEMORY	128MB, DDR2, 166MHz
STORAGE	16M NOR on board, support SDXC up to 2TB
WIRELESS	802.11n, 2T2R, speed up to 300Mbps.
ANTENNA	One U.FL slot, one on board antenna.
ETHERNET	1 port/5 ports, up to 100Mbps.
USB	Support USB 2.0(host only), up to 480MBit/s.
GPIO	around 40 (pinmux)
UART	x3 (UART2 for debug console)
PWM	x4
PCIe(option)	x1(option)
TEMPERATURE	0C ~ 40C(full load) or -10C ~ 70C
POWER SUPPLY	3.6V ~ 5.5V, 500mA
POWER CONSUMPTION	74mA wifi standby, 230mA wifi full speed, 5V input.
LOGIC LEVEL	all logic pins are 3.3V, current 4mA/8mA



Openwrt

OpenWrt es un sistema operativo de código abierto para routers y otros dispositivos de red, que ofrece una amplia gama de características avanzadas y herramientas para la personalización y gestión de la red. Con OpenWrt, los usuarios pueden tomar el control total de sus dispositivos de red y ajustar la configuración a sus necesidades específicas.

El proyecto OpenWrt comenzó en 2004 como una bifurcación del proyecto LEDE, que a su vez fue una bifurcación del proyecto OpenWrt original. La comunidad OpenWrt ha desarrollado un conjunto de herramientas de desarrollo, paquetes de software y una amplia documentación para la personalización y el mantenimiento de dispositivos de red.

Entre las características de OpenWrt se incluyen la capacidad de instalar y ejecutar software adicional, soporte para múltiples interfaces de red y protocolos, cortafuegos avanzados, soporte para túneles VPN, configuración de VLAN, soporte para redes mesh, control de ancho de banda, y muchas otras opciones avanzadas.

OpenWrt es compatible con una amplia variedad de hardware de red, desde routers domésticos hasta dispositivos más avanzados y comerciales. Además, OpenWrt es compatible con una gran cantidad de arquitecturas de procesadores, como ARM, MIPS y x86, lo que permite a los usuarios instalarlo en una amplia variedad de dispositivos.

En resumen, OpenWrt es un sistema operativo de código abierto para dispositivos de red que ofrece una amplia gama de características y herramientas avanzadas para la personalización y gestión de la red. La comunidad OpenWrt continúa desarrollando y mejorando este sistema operativo para brindar a los usuarios la capacidad de tomar el control total de sus dispositivos de red.

Sistema de particiones

En OpenWrt, es posible utilizar diferentes sistemas de archivos para particionar y gestionar el almacenamiento del dispositivo. Dos de los sistemas de archivos más comunes son SquashFS y JFFS2.

SquashFS es un sistema de archivos de solo lectura que comprime los archivos para ahorrar espacio de almacenamiento y mejorar el rendimiento de lectura. En OpenWrt, SquashFS se utiliza generalmente para particionar la imagen de firmware en una partición de solo lectura que contiene el sistema operativo, los programas y los paquetes de software instalados.

Por otro lado, JFFS2 es un sistema de archivos diseñado específicamente para dispositivos de almacenamiento flash, que permite la escritura y lectura de archivos en tiempo real. En OpenWrt, JFFS2 se utiliza comúnmente para particionar la partición de almacenamiento de usuario, que es la partición que contiene los archivos de configuración y los datos del usuario.

Para particionar OpenWrt con SquashFS y JFFS2, se pueden seguir los siguientes pasos:

1. SquashFS para el sistema operativo y los programas. Esta partición debe tener un tamaño suficiente para contener todos los archivos del sistema operativo y los programas instalados.
2. Utilizar la partición JFFS2 para almacenar los archivos de configuración y los datos del usuario, como la configuración de red, los archivos de usuario y las bases de datos.

Al particionar OpenWrt con SquashFS y JFFS2, se puede aprovechar al máximo el almacenamiento del dispositivo y mantener un sistema operativo y programas eficientes y estables, así como datos de usuario seguros y accesibles.

Esta disposición es la de por defecto en el make menuconfig, con lo que no hay que tocar nada

Creación del entorno de compilación

Para crear un entorno de compilación de imágenes OpenWrt, sigue los siguientes pasos:

1. Instala las herramientas de compilación: Para compilar OpenWrt, necesitarás algunas herramientas de compilación, como gcc, g++, make, binutils, etc. Para instalarlas en sistemas basados en Debian o Ubuntu, ejecuta el siguiente comando en una terminal. (Nota : según tu sistema puede que tengas que instalar otros paquetes faltantes de los que te advertirá el comando “make” más adelante)

```
sudo apt-get install build-essential libncurses5-dev zlib1g-dev gawk git ccache
gettext libssl-dev xsltproc unzip
```

2. Clona el repositorio de OpenWrt: Descarga el código fuente de OpenWrt desde su repositorio oficial de Git ejecutando el siguiente comando en una terminal:

```
git clone https://git.openwrt.org/openwrt/openwrt.git
```

3. Navega al directorio del código fuente: Ve al directorio del código fuente de OpenWrt mediante el siguiente comando:

```
cd openwrt/
```

4. Actualiza la lista de paquetes y las dependencias: Ejecuta el siguiente comando para descargar y actualizar la lista de paquetes necesarios para compilar OpenWrt:

```
./scripts/feeds update -a
```

Después de esto, se deben instalar todas las dependencias necesarias de los paquetes descargados. Esto se puede hacer mediante el siguiente comando:

```
./scripts/feeds install -a
```

5. Personaliza la configuración: Ejecuta el comando `make menuconfig` para personalizar la configuración de OpenWrt. Esto abrirá una interfaz gráfica basada en texto, donde puedes seleccionar los paquetes que deseas compilar e incluir en la imagen.
6. Compila OpenWrt: Después de haber personalizado la configuración, ejecuta el siguiente comando para compilar OpenWrt:

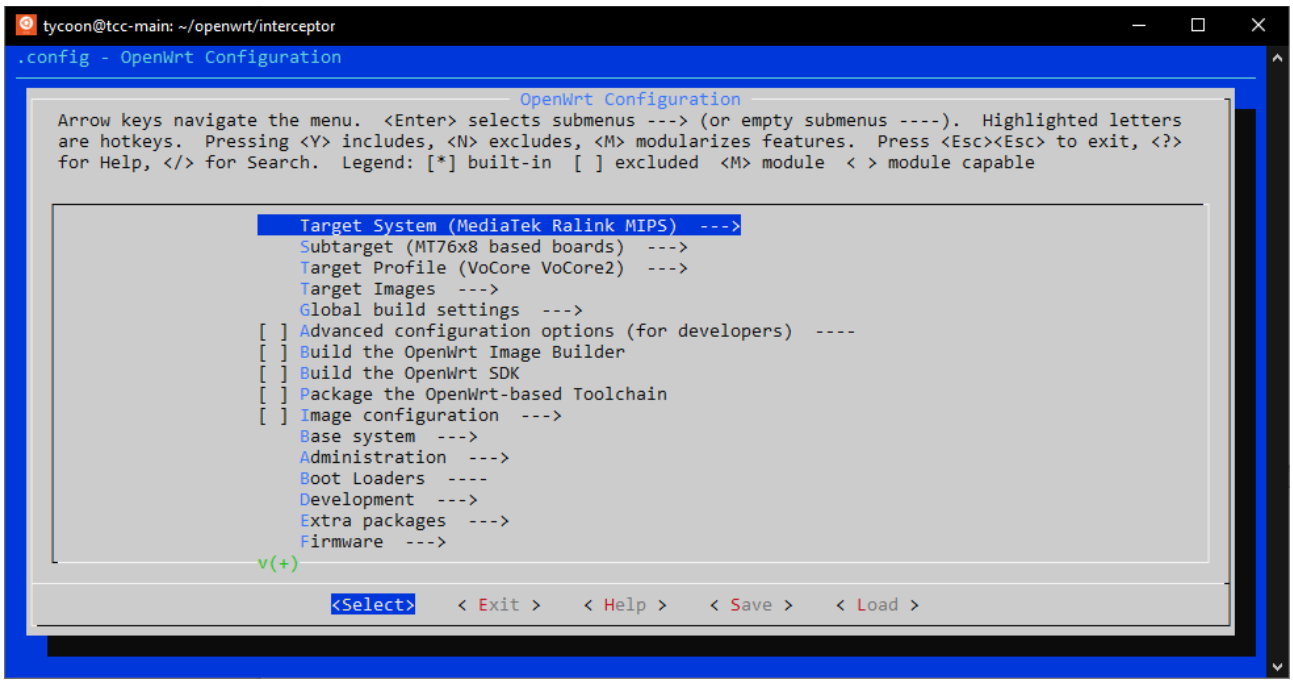
```
make menuconfig
```

debemos seleccionar :

Target system

Subtarget

Target Profile tal y como aparece en la imagen



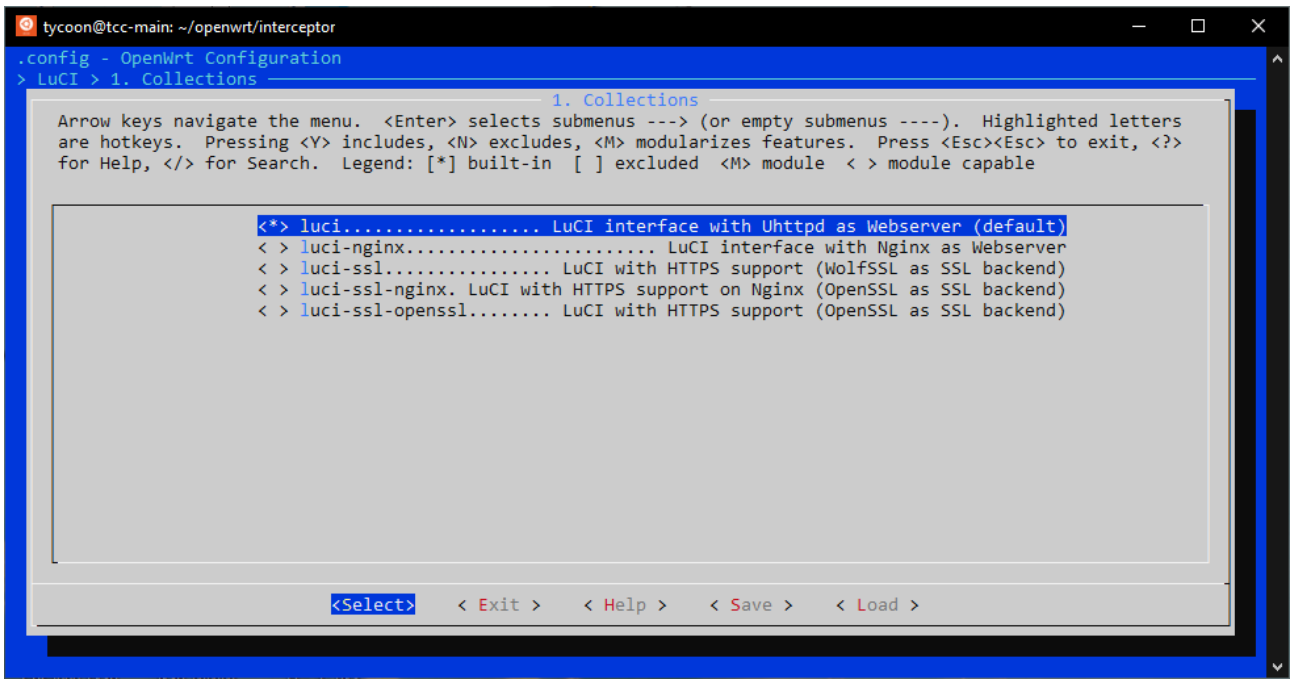
salimos y guardamos, ejecutamos

```
make -j4
```

Donde `-j4` indica el número de hilos de compilación a utilizar. El número `4` indica el número de núcleos de CPU disponibles. Este valor puede variar según el número de núcleos de CPU en tu sistema.

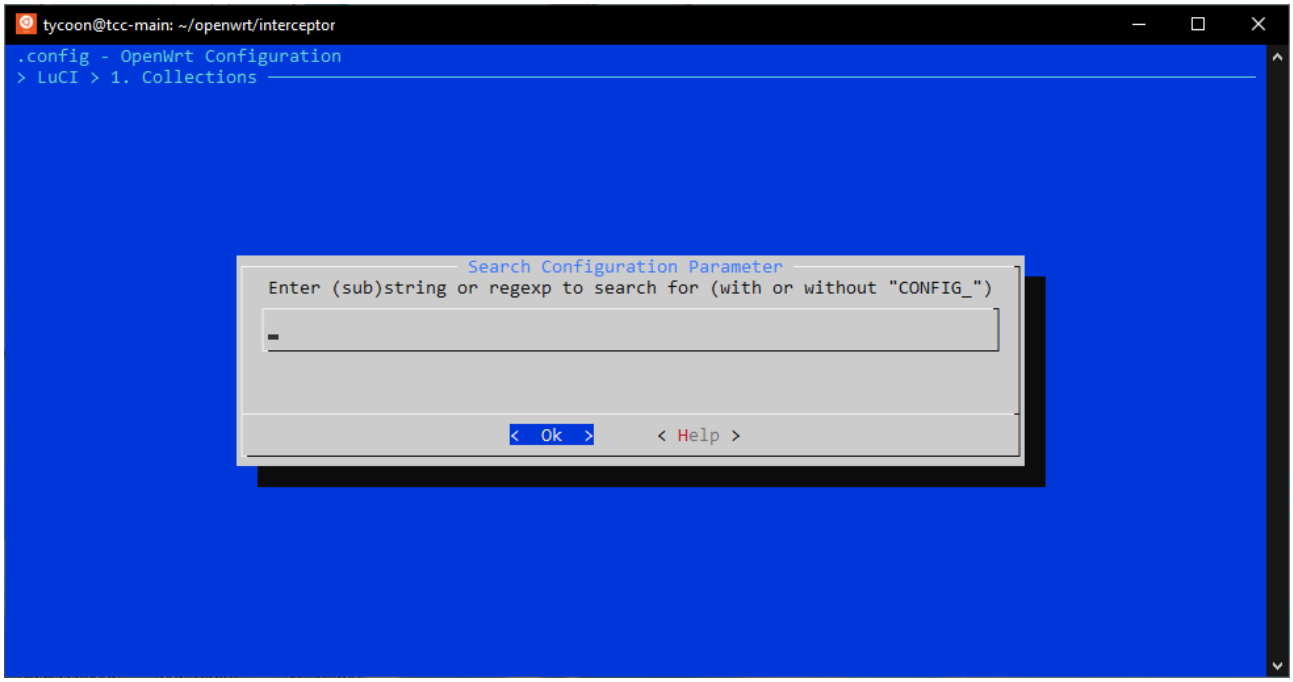
7. Espera a que la compilación finalice: La compilación puede tardar varios minutos o incluso horas, dependiendo de la velocidad de tu CPU y de la cantidad de paquetes que hayas seleccionado para incluir en la imagen.
8. Recupera la imagen compilada: Después de que la compilación haya terminado, encontrarás la imagen compilada en el directorio `bin/`. La imagen puede variar dependiendo del modelo de dispositivo para el que hayas compilado OpenWrt.

Ejemplo de instalacion de paquetes adicionales (interfaz web)



la interfaz web viene como metapaquete llamado LuCi

Para buscar nombres de paquetes: presionar / y teclear términos



make menuconfig y make kernel_menuconfig

El comando "make menuconfig" y "make kernel_menuconfig" son herramientas que se utilizan en el proceso de compilación del kernel de Linux para personalizar las opciones de configuración del kernel.

Cuando se compila el kernel de Linux, es necesario configurar las opciones de construcción para que se adapten a las necesidades del sistema en el que se ejecutará. El kernel de Linux es el componente fundamental del sistema operativo, y es responsable de administrar los recursos del hardware, proporcionar servicios y manejar los procesos.

La herramienta "make menuconfig" es una herramienta de línea de comandos que permite a los usuarios personalizar las opciones de configuración del kernel. Esta herramienta proporciona una interfaz gráfica basada en texto que permite a los usuarios seleccionar y deseleccionar módulos del kernel, opciones de compilación y otros parámetros que afectan al comportamiento del kernel. "make menuconfig" es una herramienta muy útil para los desarrolladores y usuarios avanzados que necesitan compilar y personalizar el kernel de Linux para un sistema específico.

Por otro lado, "make kernel_menuconfig" es una herramienta similar a "make menuconfig", pero está diseñada específicamente para personalizar las opciones de configuración del kernel de OpenWrt. OpenWrt es un sistema operativo de red que se basa en Linux, pero se diferencia de otros sistemas operativos en su enfoque en el networking y en los dispositivos embebidos. Por lo tanto, "make kernel_menuconfig" ofrece opciones de configuración específicas de OpenWrt que no se encuentran en el kernel estándar de Linux.

En resumen, "make menuconfig" y "make kernel_menuconfig" son herramientas importantes para personalizar las opciones de configuración del kernel de Linux y del kernel de OpenWrt, respectivamente. Estas herramientas permiten a los usuarios seleccionar y deseleccionar módulos del kernel, opciones de compilación y otros parámetros que afectan al comportamiento del kernel. Con estas herramientas, los usuarios pueden compilar y personalizar el kernel para que se adapte a las necesidades de su sistema operativo y hardware específico.

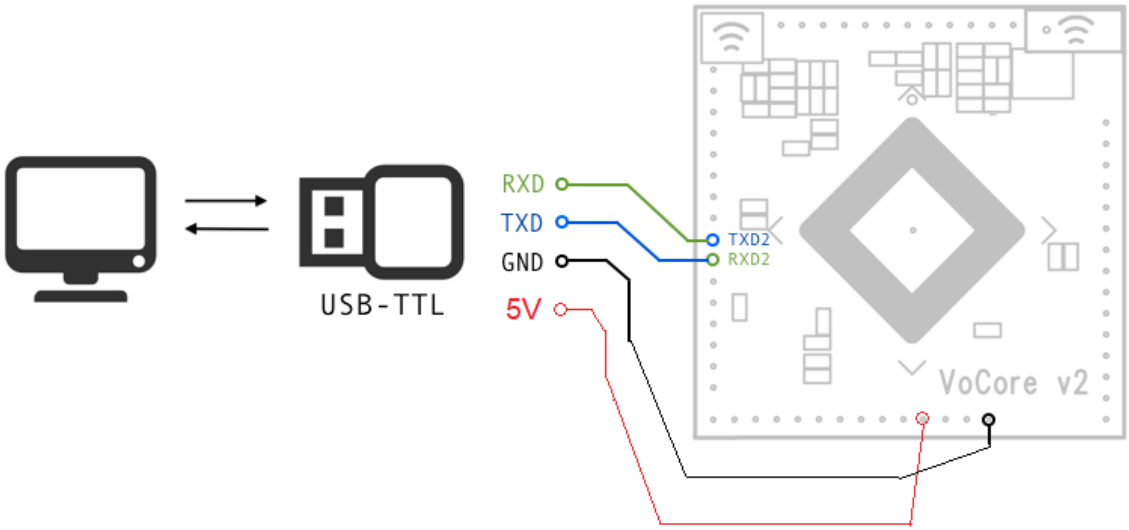
Cómo alimentar y communicarse con la placa vocore2 con un conversor USB-UART

Para conectar una placa Vocore a un conversor UART a USB, sigue los siguientes pasos:

1. Obtén un conversor UART a USB: Necesitarás un conversor UART a USB para conectar la placa Vocore a tu ordenador. Puedes comprar uno en línea o en una tienda de electrónica local.

2. Conecta el conversor a tu ordenador: Conecta el cable USB del conversor UART a USB a un puerto USB disponible en tu ordenador.
3. Conecta el conversor a la placa Vocore: Conecta los cables RX y TX del conversor UART a USB a los pines RX y TX en la placa Vocore. Asegúrate de conectar el cable RX del conversor al pin TX de la placa Vocore, y el cable TX del conversor al pin RX de la placa Vocore. También debes conectar el cable GND del conversor a un pin GND en la placa Vocore.
4. Configura la comunicación: Una vez que el conversor esté conectado a la placa Vocore y a tu ordenador, deberás configurar la comunicación para poder comunicarte con la placa Vocore. Abre un terminal en tu ordenador y configura la velocidad de baudios, los bits de datos, la paridad y los bits de parada según la configuración de tu placa Vocore. Por ejemplo, si tu placa Vocore tiene una velocidad de baudios de 115200, debes configurar la velocidad de baudios del terminal en 115200.
5. Conéctate a la placa Vocore: Una vez que hayas configurado la comunicación, deberías poder conectarte a la placa Vocore a través del terminal en tu ordenador. Puedes usar el terminal para enviar comandos y recibir respuestas de la placa Vocore.

Siguiendo estos pasos, deberías poder conectar una placa Vocore a un conversor UART a USB y comunicarte con ella a través de un terminal en tu ordenador.



Localizar imagen creada

Desde el directorio de trabajo de openwrt:

bin/targets/ramips/mt76x8/openwrt-ramips-mt76x8-vocore2-squashfs-sysupgrade.bin

lo más cómodo es hacer un enlace simbólico con nombre “openwrt.bin” en el directorio base de openwrt que apunte a ésta ruta

control por consola por minicom para vocore2

Al conectar el usb a uart con todo soldado, y hacer un dmesg en la consola de linux, deberíamos ver algo parecido a :

```
[ 2715.308582] usb 3-3: new full-speed USB device number 2 using xhci_hcd
[ 2715.464141] usb 3-3: New USB device found, idVendor=0403, idProduct=6001,
bcdDevice= 6.00
[ 2715.464150] usb 3-3: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 2715.464155] usb 3-3: Product: FT232R USB UART
[ 2715.464160] usb 3-3: Manufacturer: FTDI
[ 2715.464164] usb 3-3: SerialNumber: A7036B9X
[ 2715.649882] usbcore: registered new interface driver usbserial
[ 2715.649912] usbcore: registered new interface driver usbserial_generic
[ 2715.649940] usbserial: USB Serial support registered for generic
```

```
[ 2715.657998] usbserial: USB Serial support registered for FTDI USB Serial
Device
[ 2715.658175] ftdi_sio 3-3:1.0: FTDI USB Serial Device converter detected
[ 2715.658219] usb 3-3: Detected FT232RL
[ 2715.661737] usb 3-3: FTDI USB Serial Device converter now attached to ttyUSB0
```

Para utilizar Minicom con OpenWrt con una velocidad de transmisión de 115200 baudios, 8 bits de datos, sin paridad, sin bits de parada y sin control de flujo, se deben seguir los siguientes pasos:

1. Conecte el cable serial a la placa OpenWrt y al ordenador.
2. Abra una terminal en el ordenador y ejecute el siguiente comando para instalar Minicom (en caso de que no esté instalado):

   ```
   arduino
   ```

- `sudo apt-get install minicom`

- Abra Minicom con el siguiente comando:

3. `sudo minicom -s`

4. En la pantalla de configuración de Minicom, configure los siguientes ajustes:

   a. Seleccione "Serial port setup".

   b. Configure el puerto serie de su placa (por ejemplo, /dev/ttyUSB0).

   c. Configure una velocidad de transmisión de 115200 baudios.

   d. Configure 8 bits de datos (8N1).

   e. Configure sin paridad (Off).

   f. Configure sin bits de parada (Off).

   g. Configure sin control de flujo (Off).

   h. Guarde la configuración con "Save setup" y luego "Exit".

5. Una vez que se ha configurado Minicom, presione la tecla Enter en la terminal para comenzar a recibir datos de la placa.

Flasear con kermit

Para flashear OpenWrt con Kermit, se deben seguir los siguientes pasos:

1. Conectar el cable serie a la placa en la que se quiere instalar OpenWrt y al ordenador.

2. Abrir una terminal en el ordenador y configurar la velocidad de transmisión de datos, paridad y bits de datos y de parada según las especificaciones de la placa. Por ejemplo, se puede utilizar la herramienta `cu` en Linux con el siguiente comando:

aviso: parece que hay placas con baudrates de uboot (no en la parte openwrt) diferentes a 115200 y puede aparecer basura al arrancar la placa, en lo que termina de arrancar uboot y tomar el control openwrt.

Algunos baudrates típicos con los que probar:

9600 bps

19200 bps

38400 bps
57600 bps
115200 bps
230400 bps
460800 bps
921600 bps

```
cu -l /dev/ttyUSB0 -s 115200
```

3. Iniciar Kermit en la terminal:

```
kermit
```

4. Configurar Kermit con los siguientes comandos:

```
set line /dev/ttyUSB0
set speed 115200
set carrier-watch off
set handshake none
set flow-control none
set prefixing all
set file type bin
set file name lit
set rec pack 1000
set send pack 1000
```

5. Verificar que la conexión entre el ordenador y la placa esté establecida. Puede hacerse con el siguiente comando:

```
connect
```

6. Poner la placa en modo de recuperación: para el vocore2 es necesario reiniciarla, y pulsar 0 cuando se nos pregunte:

7. Enviar la imagen de OpenWrt con el siguiente comando:

```
arduino

send openwrt.bin
```

8. Esperar a que la transferencia se complete. Una vez completada, la placa se reiniciará automáticamente y se iniciará con OpenWrt instalado.

Qué es el uboot

Un bootloader es un programa que se ejecuta al encender o reiniciar un dispositivo, antes de que se inicie el sistema operativo principal. El bootloader tiene la tarea de cargar y ejecutar el sistema operativo y otros programas importantes en la memoria del dispositivo. También puede permitir la actualización del firmware del dispositivo, configurar la memoria y los periféricos, y realizar otras tareas importantes durante la fase de inicio del sistema.

U-Boot (Universal Bootloader) es uno de los bootloaders más populares y ampliamente utilizados en la comunidad de sistemas embebidos y de código abierto. Fue desarrollado originalmente para el procesador PowerPC de Motorola, pero ahora soporta una amplia variedad de procesadores, incluyendo ARM, MIPS, x86, y otros.

U-Boot es un programa de código abierto y se distribuye bajo la licencia GPL. Es altamente configurable y se puede personalizar para satisfacer las necesidades específicas de diferentes dispositivos y sistemas embebidos. Algunas de las características más notables de U-Boot incluyen:

- Soporte para múltiples sistemas de archivos y protocolos de red, lo que permite la carga de imágenes de arranque desde diferentes fuentes, como memoria flash, tarjetas SD y dispositivos de almacenamiento en red.
- Una consola de comandos interactiva que permite la configuración y depuración del sistema.

- Soporte para actualizaciones de firmware en el campo (OTA) para dispositivos embebidos conectados a Internet.
- Capacidades de depuración avanzadas, como la depuración remota y la traza de hardware.

Uboot no pertenece a OPENWRT, sino que se ejecuta antes de éste y permite la carga de las imágenes generadas con éste.

Flash via ssh

Para flashear via ssh es necesario conectarse via Wifi al Vocore2

una vez en su red, copiamos con:

scp openwrt.bin root@192.168.XX.1:/tmp/

siendo XX la subred asignada. Tipicamente es la 192.168.250.60/0.24 si la placa viene de fabrica o 192.168.1.0/24 si la placa está con una imagen openwrty por defecto

entramos en usa sesion de ssh :

ssh root@192.168.XX.1

cd /tmp

sysupgrade -c openwrt.bin

Ficheros extra que podemos incluir en la imagen

Cualquier configuracion que queramos conservar entre flasheos podemos preservarla copiando el fichero desde el propio vocore hasta la carpeta “files” que tenemos que crear dentro del directorio base de nuestro entorno de compilación:

```
scp -r root@192.168.1.1:/etc/config/* files/
```

Sistema de configuración UCI de openwrt

El sistema de configuración UCI (Unified Configuration Interface) es una herramienta que se utiliza en OpenWrt para gestionar la configuración del sistema. UCI se utiliza para configurar una variedad de componentes del sistema, como la configuración de red, la configuración del cortafuegos (firewall), la configuración del sistema de archivos, la configuración del servidor DHCP y muchas otras opciones de configuración.

El sistema de configuración UCI se basa en archivos de texto plano que describen la configuración de diferentes componentes del sistema. Estos archivos se encuentran en el directorio `/etc/config` y se pueden editar fácilmente para modificar la configuración del sistema.

UCI también proporciona una interfaz de línea de comandos y una interfaz web para configurar el sistema. Esto permite a los usuarios configurar fácilmente el sistema utilizando la interfaz que prefieran.

Algunos enlaces de interés relacionados con UCI y OpenWrt son los siguientes:

- Página web oficial de OpenWrt: https://openwrt.org/
- Documentación de UCI en la wiki de OpenWrt: https://openwrt.org/docs/guide-user/base-system/uci
- Tutorial de configuración de red en OpenWrt utilizando UCI: https://openwrt.org/docs/guide-user/network/wifi/basic
- Guía de usuario de OpenWrt: https://openwrt.org/docs/guide-user/start
- Foro de discusión de OpenWrt: https://forum.openwrt.org/

devicetree

El Device Tree (DT) es una técnica utilizada en sistemas embebidos, incluyendo OpenWrt, para describir la configuración de hardware de un sistema en un formato estructurado y legible por la máquina. El DT es una descripción jerárquica de los dispositivos hardware del sistema, incluyendo

información como el modelo de dispositivo, la dirección de memoria, las interrupciones, los puertos de E/S y otras características.

Para incluir un sensor de presión atmosférica BME en el DT de un sistema OpenWrt, debes seguir los siguientes pasos:

1. Verifica si el controlador del BME ya está disponible en el kernel de OpenWrt. Para ello, puedes buscar en la documentación de OpenWrt o realizar una búsqueda en la página web del kernel de Linux.
2. Agrega una nueva entrada de dispositivo en el archivo DT que describe el sensor BME. El archivo DT se encuentra en la ruta `target/linux/generic/files/arch/arm/boot/dts` en el código fuente de OpenWrt.
3. La entrada de dispositivo para el sensor BME debe incluir información sobre el modelo del sensor, la dirección de registro y otros detalles relevantes. Puedes consultar la hoja de datos del BME para obtener esta información.
4. Compila el kernel de OpenWrt con el nuevo DT que incluye la entrada del sensor BME.
5. Una vez compilado el kernel, carga el nuevo DT en el sistema utilizando las herramientas de configuración de OpenWrt, como UCI.

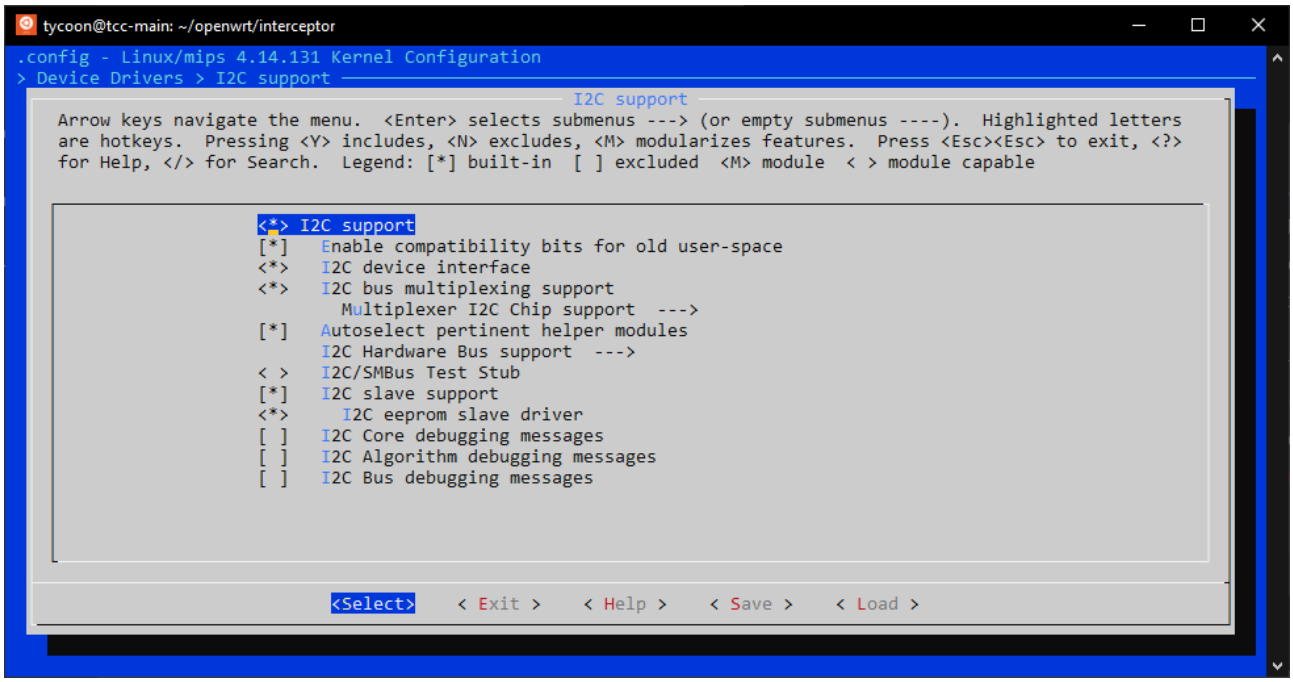
Es importante tener en cuenta que los detalles específicos para agregar el sensor BME al DT de OpenWrt pueden variar según la plataforma y el modelo del sensor BME. Por lo tanto, es recomendable consultar la documentación y los recursos de la comunidad de OpenWrt para obtener información detallada sobre cómo agregar el sensor BME al DT de OpenWrt en tu caso específico.

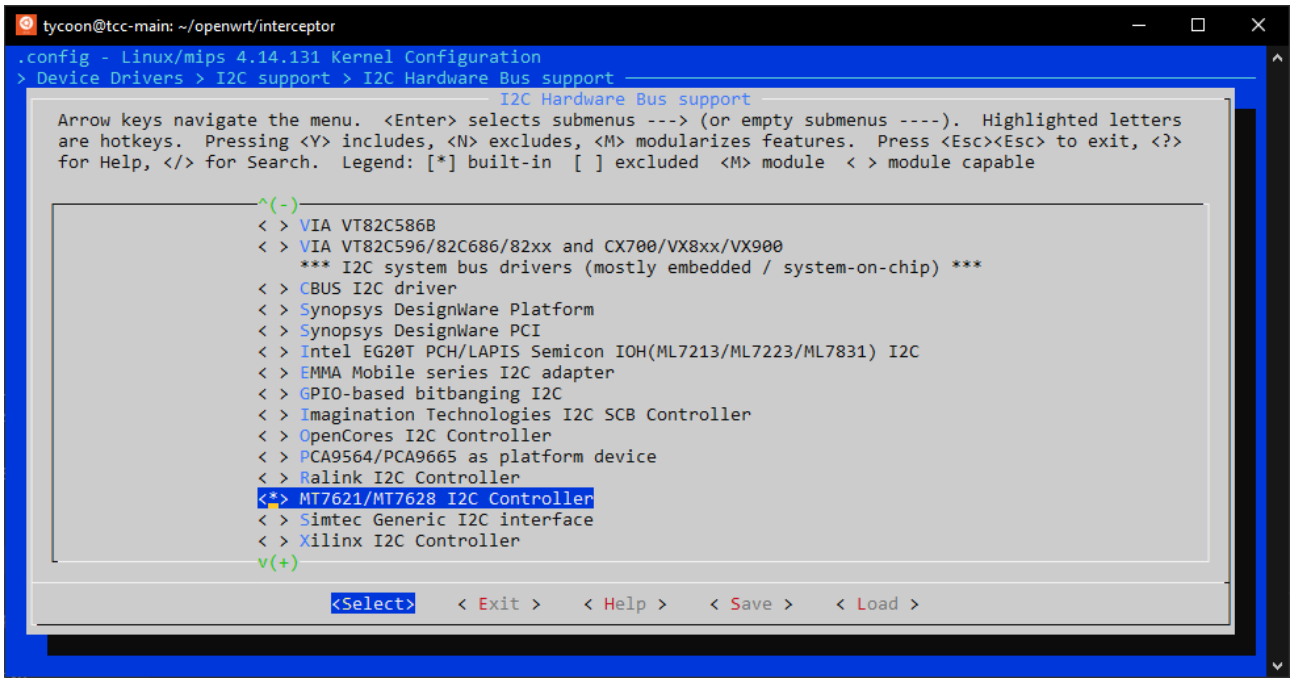
Ejemplo a agregar en ./target/linux/ramips/dts/VOCORE2.dts el sensor bmp280

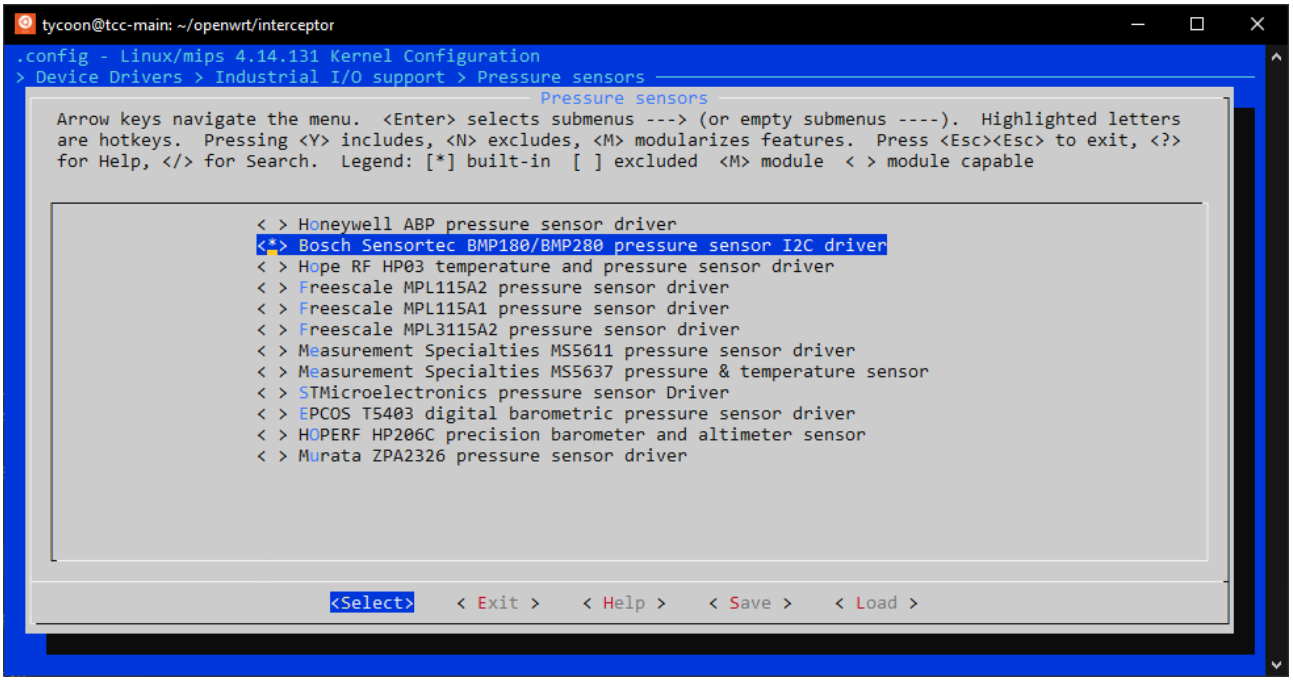
```
&i2c {
    status = "okay";

    bmp280: bmp280@76 {
        compatible = "bosch,bmp280";
        reg = <0x76>;
    };
};
```

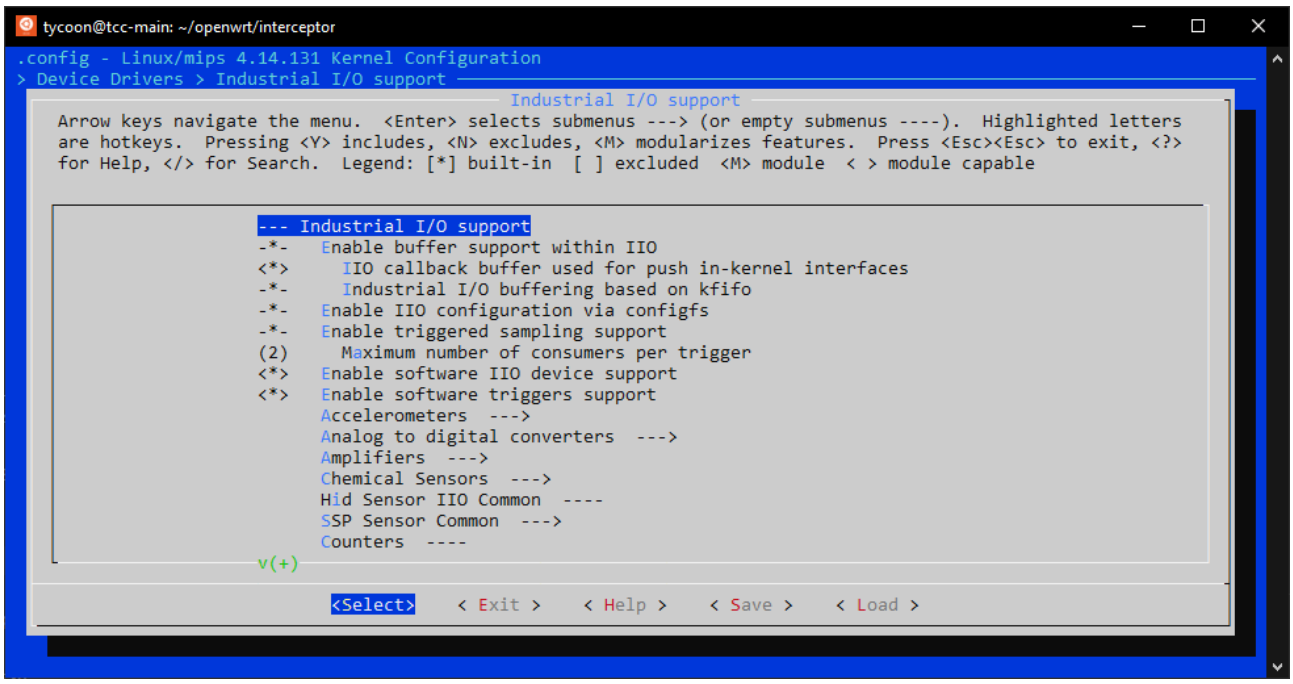
Agregar el driver necesario mediante make kernel_menuconfig

Selecciona las opciones indicadas en las imágenes. Recuerda que la ruta está indicada en la esquina superior izquierda









Leer el sensors desde sysfs

Asegúrate de que el driver del bus I2C esté cargado en el kernel de OpenWrt. Puedes verificar esto con el siguiente comando:

- `ls /dev/i2c-*`

Deberías ver una lista de los buses I2C disponibles en tu sistema.

- Asegúrate de que el driver del BMP280 esté cargado en el kernel de OpenWrt. Puedes verificar esto con el siguiente comando:

- `ls /sys/class/i2c-adapter/i2c-*/i2c-*/*280*`

Deberías ver un directorio que contiene los archivos de sysfs para el driver del BMP280.

- Para leer los datos del sensor de presión y temperatura, accede a los archivos de sysfs proporcionados por el driver del BMP280. Por ejemplo, si el driver se carga en el bus I2C 0 y en la dirección 0x76, puedes leer la temperatura del sensor desde el archivo `/sys/class/i2c-adapter/i2c-0/0-0076/iio:device0/in_temp_input` y la presión desde el archivo `/sys/class/i2c-adapter/i2c-0/0-0076/iio:device0/in_pressure_input`.

Los datos proporcionados por estos archivos están en unidades sin procesar y deben ser convertidos a unidades más útiles. Puedes hacer esto usando las fórmulas proporcionadas en la hoja de datos del BMP280.

Herramientas del i2c

Los comandos `i2cdump`, `i2cdetect`, `i2cget` y `i2cset` son herramientas útiles para trabajar con dispositivos que se comunican a través del bus I2C en sistemas operativos como Linux.

Aquí te muestro una demostración de cómo usar estos comandos en un ejemplo práctico:

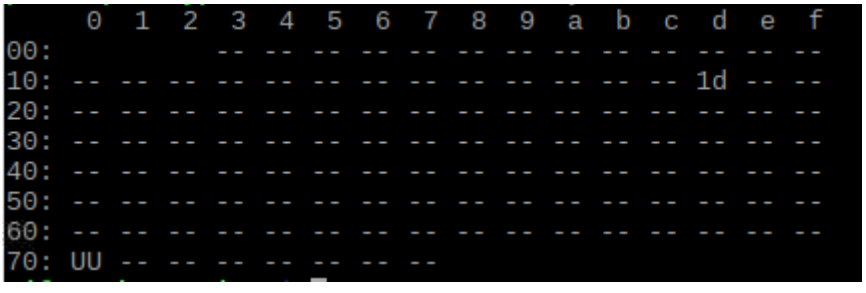
1. `i2cdetect`

   El comando `i2cdetect` se utiliza para detectar dispositivos conectados a un bus I2C. En este ejemplo, tenemos un dispositivo I2C conectado al bus I2C 1 en la dirección 0x68. Para detectar este dispositivo, ejecutamos el siguiente comando en la terminal:

- `i2cdetect -y 1`

El resultado de este comando es una tabla que muestra los dispositivos detectados en el bus I2C 1. La dirección 0x68 aparece en la tabla, lo que indica que el dispositivo está conectado correctamente.

Si hay un driver utilizando ya el dispositivo, en lugar de mostrar la dirección, se mostrará UU



- `i2cdump`

El comando `i2cdump` se utiliza para leer una sección de memoria en un dispositivo I2C. En este ejemplo, vamos a leer los primeros 16 bytes de la memoria del dispositivo I2C en la dirección 0x68 en el bus I2C 1. Para hacer esto, ejecutamos el siguiente comando en la terminal:

- `i2cdump -y 1 0x68`

El resultado de este comando es una tabla que muestra los primeros 16 bytes de la memoria del dispositivo I2C en la dirección 0x68.

- `i2cget`

El comando `i2cget` se utiliza para leer un byte de datos desde una dirección específica en un dispositivo I2C. En este ejemplo, vamos a leer el byte de datos en la dirección 0x0D en el dispositivo I2C en la dirección 0x68 en el bus I2C 1. Para hacer esto, ejecutamos el siguiente comando en la terminal:

- `i2cget -y 1 0x68 0x0D`

El resultado de este comando es un byte de datos en formato hexadecimal.

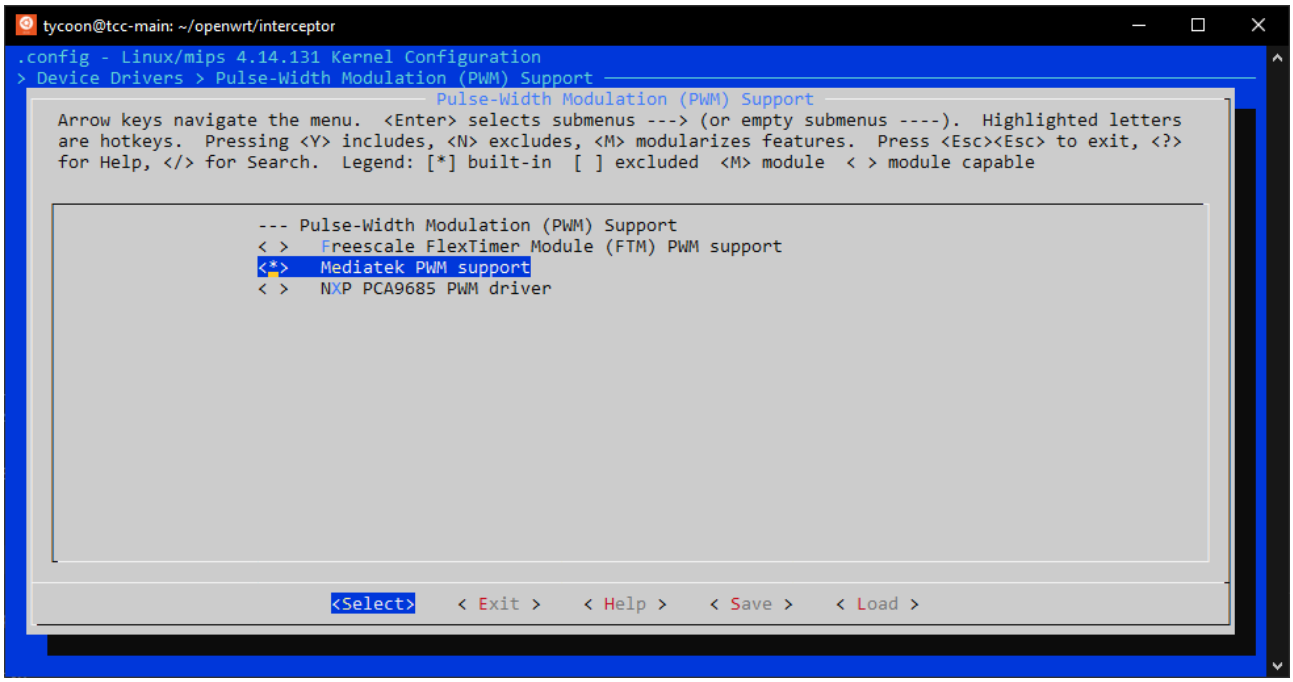
- `i2cset`

El comando `i2cset` se utiliza para escribir un byte de datos en una dirección específica en un dispositivo I2C. En este ejemplo, vamos a escribir el valor 0xFF en la dirección 0x0D en el dispositivo I2C en la dirección 0x68 en el bus I2C 1. Para hacer esto, ejecutamos el siguiente comando en la terminal:

4. `i2cset -y 1 0x68 0x0D 0xFF`

   El resultado de este comando es que el byte de datos en la dirección 0x0D del dispositivo I2C en la dirección 0x68 en el bus I2C 1 se establece en 0xFF.

Opciones del build para activar pwm



Cómo conectar un miniservo al vocore2

El pinout de un servomotor estándar suele consistir en tres cables:

1. Cable rojo o marrón: Este cable se utiliza para la alimentación del servomotor. Normalmente, se conecta a una fuente de alimentación de 5V.
2. Cable negro: Este cable se utiliza para la conexión a tierra.
3. Cable amarillo, naranja o blanco: Este cable se utiliza para la señal de control del servomotor. La señal PWM que se envía a este pin determina la posición del servomotor. En

En algunos casos, el pinout del servomotor puede variar según el fabricante o el modelo específico del servo. Es importante consultar la documentación del fabricante para obtener información detallada sobre el pinout del servo en tu caso específico.

Controlando el servomotor con pwm

PWM (Pulse-Width Modulation) es una técnica utilizada en electrónica para controlar la velocidad de los motores eléctricos y otros dispositivos que requieren un control de velocidad preciso. En OpenWrt, se utiliza el kernel de Linux para controlar los pulsos PWM y los pines GPIO asociados.

Para hacer funcionar un servomotor estándar con OpenWrt, se puede utilizar el siguiente proceso:

1. Conecta el servomotor a un pin GPIO de la placa de OpenWrt. Los servomotores suelen tener tres cables, uno para la alimentación, uno para la tierra y otro para la señal. La señal se conecta al pin GPIO que se va a utilizar para controlar el servo.
2. Asegúrate de que el kernel de OpenWrt tiene el soporte para PWM y los módulos del kernel necesarios estén cargados. Esto se puede hacer a través de la configuración del kernel o mediante la instalación de los paquetes necesarios. Por ejemplo, para habilitar el soporte PWM en OpenWrt, puedes instalar el paquete `kmod-pwm`.

Si está el soporte para pwm, debería existir este directorio:

`/sys/class/pwm/pwmchip0/`

Si alimentamos el servomotor con 5V y GND y la pata de señal, la conectamos a la pata PWM0 o PWM1 del Vocore, podremos controlar el servomotor

`Para controlar un servomotor a través de PWM en Linux, debe seguir los siguientes pasos:`

1. Verifica que hay soporte de pwm con:

```
ls /sys/class/pwm/
```

Si ve el directorio `pwmchip0`, significa que hay un controlador PWM disponible en su sistema.

2. Configure el controlador PWM para generar una señal PWM. Para ello, debe crear un nuevo directorio en `/sys/class/pwm/pwmchip0/` con el siguiente comando:

```
sudo mkdir /sys/class/pwm/pwmchip0/pwm0
```

3. Especifique la frecuencia y el ciclo de trabajo del PWM. Por ejemplo, para generar una señal PWM a una frecuencia de 50 Hz y un ciclo de trabajo del 5%, debe escribir los siguientes valores en los siguientes archivos:

```
echo 50000000 > /sys/class/pwm/pwmchip0/device/pwm_period
echo 5 > /sys/class/pwm/pwmchip0/pwm0/duty_cycle
```

En este ejemplo, la frecuencia se establece en 50 Hz (50000000 ns de período), mientras que el ciclo de trabajo se establece en 5 (lo que significa que la señal estará encendida el 5% del tiempo).

4. Inicie la señal PWM escribiendo un valor diferente de cero en el archivo "enable" de la siguiente manera:

```
echo 1 > /sys/class/pwm/pwmchip0/pwm0/enable
```

5. Conecte el servomotor a la salida PWM y asegúrese de que la polaridad esté correctamente conectada.
6. Ajuste el ciclo de trabajo para controlar la posición del servomotor. El ciclo de trabajo determina el ángulo de rotación del servo. Por lo tanto, puede ajustar el ciclo de trabajo para controlar la posición del servomotor en grados.
7. Para detener la señal PWM, escriba un valor de cero en el archivo "enable":

```
echo 0 > /sys/class/pwm/pwmchip0/pwm0/enable
```

Es importante tener en cuenta que cada servomotor puede tener sus propios requisitos de configuración y ajuste del ciclo de trabajo. Por lo tanto, asegúrese de leer la documentación del fabricante del servomotor antes de intentar controlarlo mediante PWM en Linux.

Compilar un programa escrito en C para la arquitectura del vocore2

Para realizar una compilacion cruzada, es decir, generar un ejecutable desde nuestro PC, pero para que se ejecute en el Vocore2, es necesario cambiar las variables de entorno que indican dónde están las herramientas de compilación en la sesión actual:

```
export STAGING_DIR=/home/tycoon/openwrt/staging_dir
TOOLFOLDER=$(ls  "$STAGING_DIR"|egrep  toolchain-mipsel_24kc_gcc-.*_musl -m1)
export TOOLCHAIN_DIR=$STAGING_DIR"/"$TOOLFOLDER
export PATH=$TOOLCHAIN_DIR/bin:$PATH
export INCLUDE_TARGET=$STAGING_DIR/target-mipsel_24kc_musl/usr/include
export LIBRARY_TARGET=$STAGING_DIR/target-mipsel_24kc_musl/usr/lib
export INCLUDE_TCHAIN=$TOOLCHAIN_DIR/include
export LIBRARY_TCHAIN=$TOOLCHAIN_DIR/lib
export LIBRARY_USR=$TOOLCHAIN_DIR/usr/lib

export CC="mipsel-openwrt-linux-gcc"
export CXX="mipsel-openwrt-linux-g++"
export CFLAGS=" -I$INCLUDE_TARGET -I$INCLUDE_TCHAIN -L$LIBRARY_USR -
L$LIBRARY_TCHAIN -L$LIBRARY_TARGET"

export LD="mipsel-openwrt-linux-ld"
```

Después podemos ejecutar el Makefile de nuestro proyecto.

Crear un paquete para Openwrt

Para crear un paquete de OpenWrt sencillo para un programa en C, puede seguir los siguientes pasos:

1. Asegúrese de tener el entorno de compilación de OpenWrt configurado en su máquina. Puede seguir los pasos que mencioné anteriormente en esta misma conversación para configurar el entorno de compilación.
2. Escriba su código C y guárdelo en un archivo. Supongamos que su archivo se llama "my_program.c".
3. Cree un archivo Makefile para compilar su programa y empaquetarlo como un paquete de OpenWrt. El archivo Makefile debe especificar cómo se debe compilar su código C y cómo se debe empaquetar el resultado. Aquí hay un ejemplo de archivo Makefile:

```
makefile

include $(TOPDIR)/rules.mk

PKG_NAME:=my_program
PKG_VERSION:=1.0
PKG_RELEASE:=1

include $(INCLUDE_DIR)/package.mk

define Package/my_program
  SECTION:=utils
  CATEGORY:=Utilities
  TITLE:=My Program
  DEPENDS:=+libpthread
endef

define Package/my_program/description
  My Program is a simple program written in C.
endef

define Build/Prepare
  mkdir -p $(PKG_BUILD_DIR)
  $(CP) ./src/* $(PKG_BUILD_DIR)/
endef

define Build/Compile
  $(MAKE) -C $(PKG_BUILD_DIR) \
    $(TARGET_CONFIGURE_OPTS) \
    CFLAGS="$(TARGET_CFLAGS) -Wall -Werror" \
    LDFLAGS="$(TARGET_LDFLAGS) -lpthread"
endef

define Package/my_program/install
  $(INSTALL_DIR) $(1)/usr/bin
  $(INSTALL_BIN) $(PKG_BUILD_DIR)/my_program $(1)/usr/bin/
endef

$(eval $(call BuildPackage,my_program))
```

En este ejemplo, debe reemplazar "my_program" por el nombre de su programa y ajustar los campos de "PKG_NAME", "PKG_VERSION" y "PKG_RELEASE" para que coincidan con su programa.

4. Cree un directorio llamado "my_program" en el directorio "package" de su árbol de OpenWrt.

```
bash

mkdir package/my_program
```

5. Coloque los archivos "my_program.c" y "Makefile" en el directorio "my_program".

```
bash

cp my_program.c package/my_program/src/
cp Makefile package/my_program/
```

6. Compile su paquete ejecutando el siguiente comando en el directorio raíz de su árbol de OpenWrt:

```
go

make package/my_program/compile V=s
```

Este comando compila su programa y crea un paquete de OpenWrt llamado "my_program_1.0-1_*.ipk" en el directorio "bin/packages/*/my_program".

7. Transfiera el paquete "my_program_1.0-1_*.ipk" a su dispositivo OpenWrt e instálelo allí usando el siguiente comando:

```
opkg install my_program_1.0-1_*.ipk
```

Esto instalará su programa en el dispositivo OpenWrt.

Es importante tener en cuenta que el proceso de creación de paquetes de OpenWrt puede ser un poco complicado, ya que hay muchas opciones y configuraciones posibles. Por lo tanto, es recomendable leer la documentación y los foros de OpenWrt para obtener más información sobre cómo crear paquetes de OpenWrt para programas en C.