Implementar Manejadores de Dispositivos

Esp Lic Roberto Compañy

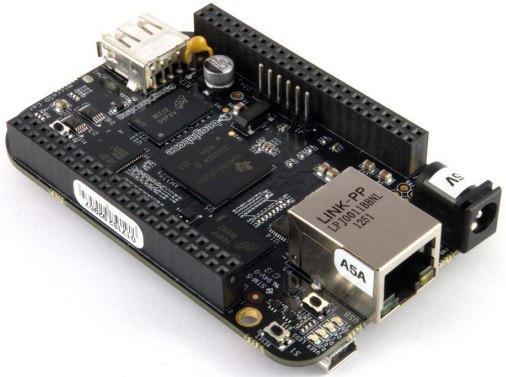
Índice

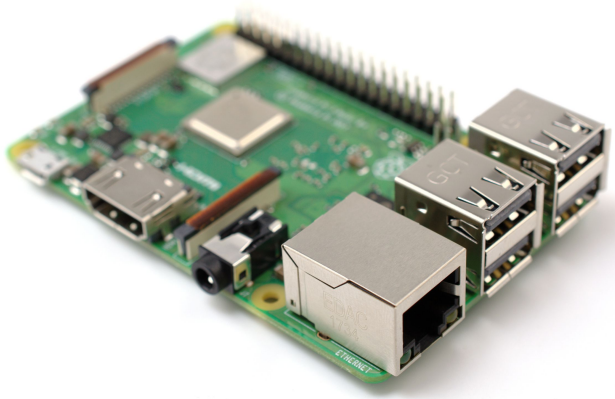
Introducción

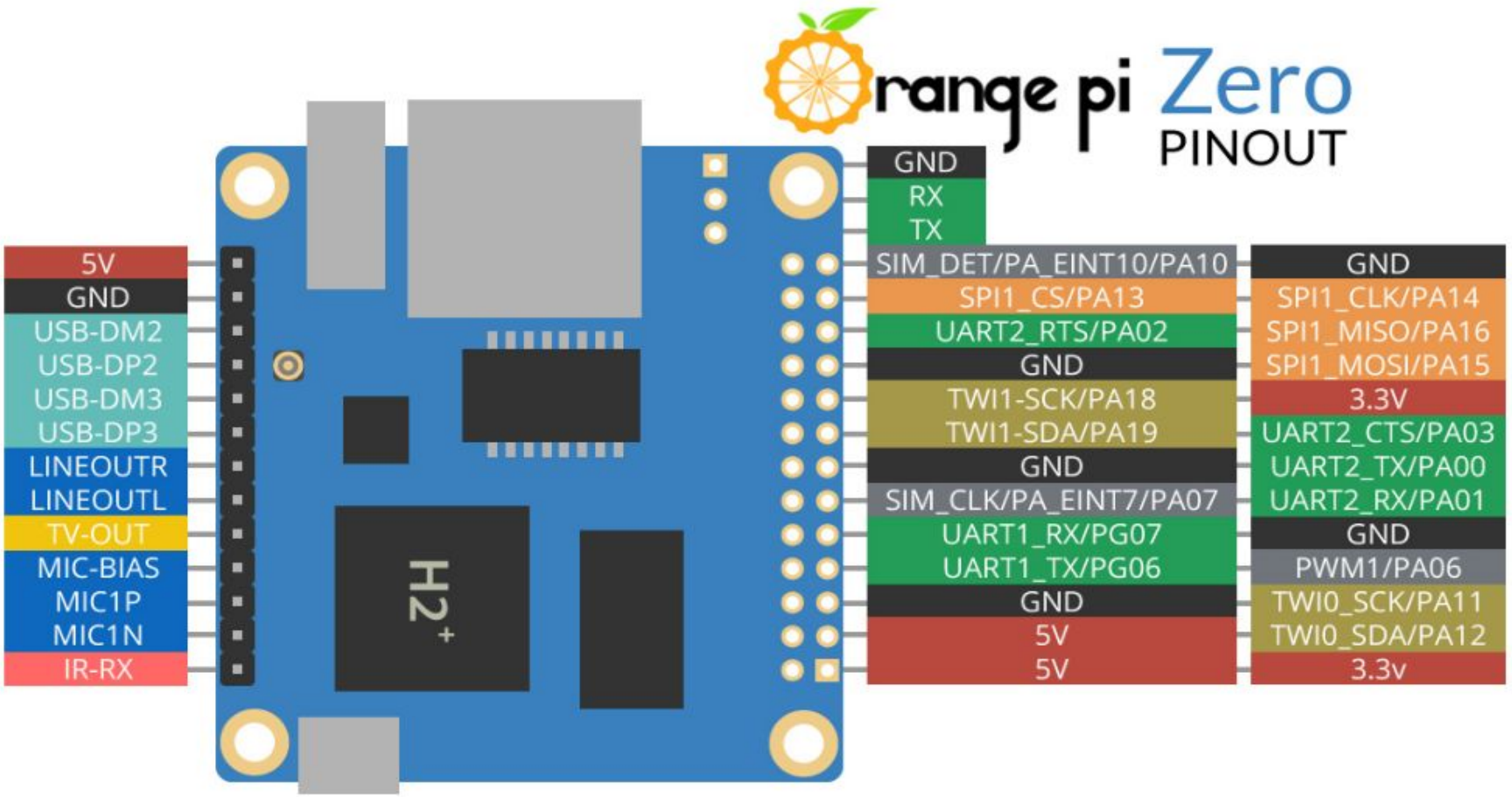
- Desarrollar un drivers para linux siguiendo la guía propuesta
- Preparar el entorno (actualizar la versión del gcc >= 7.4.0)
- Descargar fuentes del linux-mainline
- Generando toolchain necesario

Nota: Se resaltarán las diferencias con la guía y comentaran los problemas presentados y como sesolucionaron.

Elección del hardware a utilizar

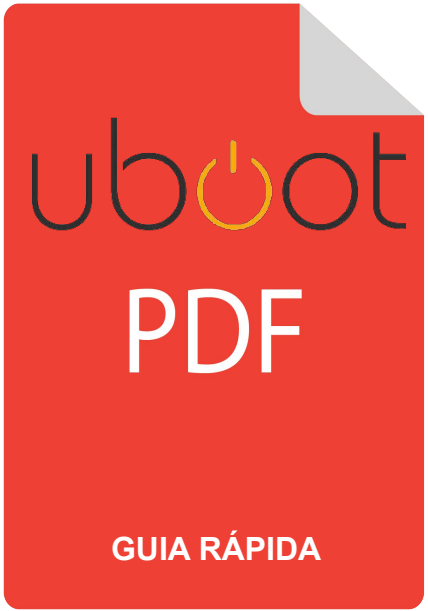





range pi Zero
PINOUT
5V
GND
USB-DM2
USB-DP2
USB-DM3
USB-DP3
LINEOUTR
LINEOUTL
TV-OUT
MIC-BIAS
MIC1P
MIC1N
IR-RX
H2+
GND
RX
TX
SIM_DET/PA_EINT10/PA10
SPI1_CS/PA13
UART2_RTS/PA02
GND
TWI1-SCK/PA18
TWI1-SDA/PA19
GND
SIM_CLK/PA_EINT7/PA07
UART1_RX/PG07
UART1_TX/PG06
GND
5V
5V
GND
SPI1_CLK/PA14
SPI1_MISO/PA16
SPI1_MOSI/PA15
3.3V
UART2_CTS/PA03
UART2_TX/PA00
UART2_RX/PA01
GND
PWM1/PA06
TWI0_SCK/PA11
TWI0_SDA/PA12
3.3v

Compilación U-Boot

- Tanto la Rasp como la Orange pi, no cuentan con el u-boot por lo que se descargó y compiló para cada una de las arquitecturas.
- El binario se copia a una tarjeta microsd y la placa bootea del mismo.
- Si en la sd se crea una partición fat, es posible guardar la configuración, y de esta forma se puede hacer que U-boot cargue e inicie automáticamente el kernel y dts al energizar la placa.



Compilación Kernel

- La guía oficial también sirve para la RaspBerry y Orange Pi, salvo por las siguientes configuraciones:
  - rpi_defconfig | rpi_3_32b_defconfig
  - sunxi_defconfig
- Aprovechando la guía del profesor Gonzalo, se decidió realizarla para validar el documento y como comentario adicional se recomienda no utilizar https://github.com/linux-sunxi/linux-sunxi sino el source del main line porque después al compilar el modulo de la guía da error de versión.

Generación del Device Tree

- No se puede partir de uno vacío como en la BeagleBone Black.
- En la Orange Pi el i2c ya tiene los pines habilitados
  - Frecuencia del bus.
  - Dirección del hardware a configurar

Creación del driver

1. Como dispositivo i2c, se optó por un DS3231 que cuenta con una memoria eeprom como la utilizada en clase y además un RTC.

2. Se desarrolló el mismo driver y programa de usuario presentado en clase para validar el funcionamiento.

3. Se utilizaron drivers nativos para familiarizarse con el funcionamiento del dispositivo.

4. Se desarrolló un driver propio y programa de usuario para leer la hora del RTC.

Going further

- Aprovechando los apuntes de clase, se analizó el código del kernel y su compatibilidad con el RTC DS3231.
- Existe documentación sobre el mismo y ya hay un driver desarrollado.
- Se configuró el driver integrado al kernel
- Para el RTC, resulta más útil utilizar las llamadas al sistema ioctl del SO que las operaciones read y write que se usaron en la memoria.
- Se desarrolló un programa de usuario que utilizando ioctl lee y establece la hora del RTC.

¿Preguntas?