1. 專案05\_gpioled:

Linux系統是一個龐大且完善的系統，尤其在驅動部分因為有些驅動非常常見(例如:GPIO)，為了增加代碼可攜性及可重用性。所以這個專案要以pinctrl和gpio子系統來實現GPIO的驅動開發。

1. Pinctrl子系統:

在Linux系統中講究分離和分層，就是為了增加代碼可攜性及可重用性，而所謂分離和分層就是按照對象去個別編成的思想。我們知道使用大多屬SOC的Pin腳是復用的，對於Pin腳的復用Linux系統推出pinctrl子系統。而pinctrl子系統就是為了解決pin腳衝突問題。Pinctrl工作如下:

* 獲取Device Tree中設備pin訊息
* 根據得到的pin訊息來設置Pin腳復用與電氣特性。

實作上，我們需要在Device Tree中加入Pin相關訊息在pinmux中

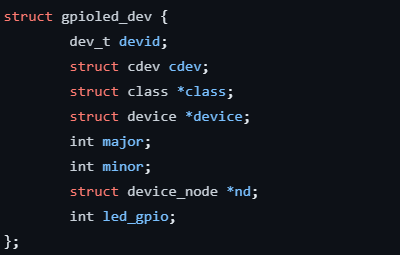
1. GPIO子系統:

當pinctrl將PIN腳復用為GPIO的話，就會需要用到GPIO子系統了。GPIO 子系統的主要目的是方便驅動開發者使用 GPIO。驅動開發者在Device Tree中添加 GPIO 相關信息，然後就可以在驅動程序中使用 GPIO 子系統提供的 API 函數來操作 GPIO。極大地方便了驅動開發者使用 GPIO。

1. 實作如下:

先定義一個設備結構體，並且把gpio加到結構體中，用來存取gpio編號。

(如下圖一)



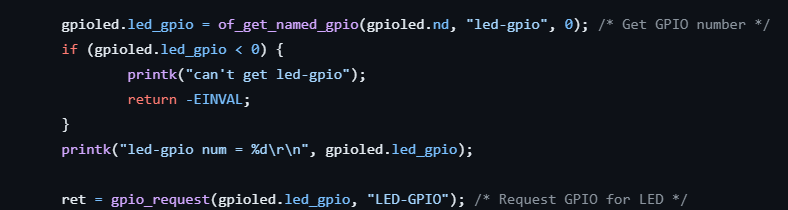
(圖一)

1. 如圖二所示，of\_find\_node\_by\_path 函數通過路徑來查找指定的節點，這裡就是用來查找”gpioled”的這個節點。
2. 如圖二所示，of\_find\_property 函數用於查找指定的屬性，這裡用來查找compatible屬性。檢查Linux Kernel的驅動程式是否支援此設備。
3. 如圖二所示，of\_property\_read\_string函數用於讀取屬性中的字符串值，此處用於檢查設備是否可操作。



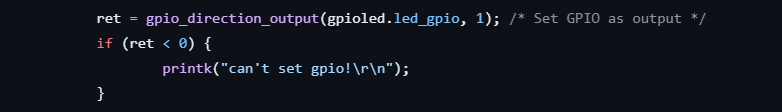
(圖二)

1. 如圖三所示，of\_get\_name\_gpio()函數用來獲取gpio的編號。
2. 如圖三所示，gpio\_request()函數用於申請一個GPIO腳，在使用一個GPIO之前一定要使用gpio\_request()進行申請。



(圖三)

1. 如圖四所示，此函數用於設置GPIO為輸出。



(圖四)

1. 接下來就是一般字元設備驅動完善。(如圖五)

MKDEV()函數，用來創建設備號。

alloc\_chrdev\_region()，用在註冊字元設備到Linux Kernel中並動態分配設備號，以供字元設備使用。

register\_chrdev\_region()，用在註冊字元設備到Linux Kernel中，給定分配設備號，以供字元設備使用。

cdev\_init()，用來初始化字元設備。

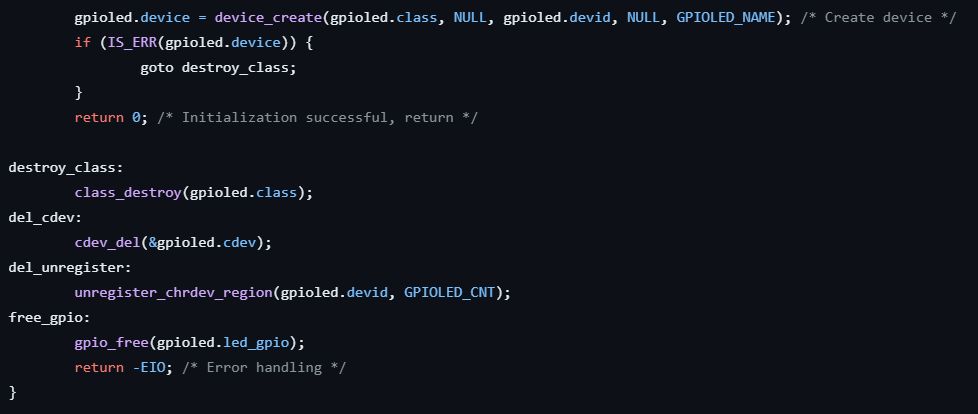
cdev\_add()，用來新增字元設備到Linux系統中。

class\_create()，用來自動創建設備節點，這樣就不需要手動mknod。



(圖五)

1. 如圖六所示，device\_create()，用來自動創建設備節點。
2. 剩餘部分是當錯誤發生時，相對應的函式處理。



(圖六)

1. 最後要卸載驅動模塊時，透過gpio\_free()來釋放掉先前申請的gpio。

(如圖七)



(圖七)

※總結:

Linux系統提供了許多方便開發者的API函式，而要使用這些函是前需要先將板子相關外部設備添加到Device Tree中。此外，Linux Kernel也為了增加驅動程式碼的可攜性與可重用性，做到了分離和分層。完成一切準備工作後，就是完善字元設備的驅動框架，內容大致上相同。

補充:"class\_create()"搭配"device\_create()"類似於手動mknod， 而"class\_destroy()"搭配"device\_destroy()"類似於手動rmnod。