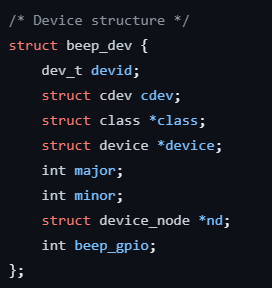
1. 專案06\_beep:

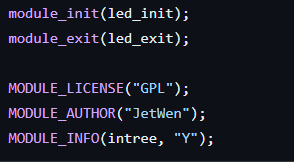
本專案會結合並統整前面幾個專案的內容。來實現蜂鳴器的驅動。

1. 字元設備驅動:
   1. 首先定義設備結構體(如圖一)



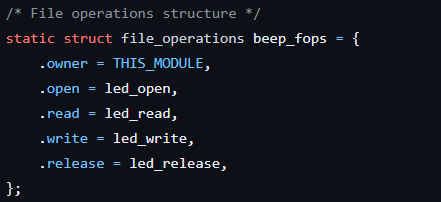
(圖一)

* 1. 接著定義驅動入口及出口函數(如圖二)



(圖二)

* 1. 最後，定義file\_operations結構體，用來溝通User Space

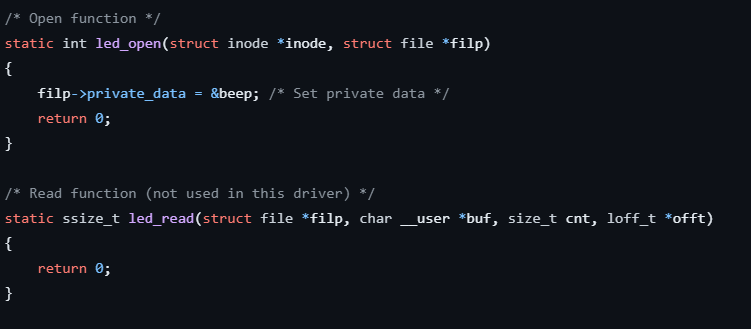


(圖三)

完成以上步驟後，字元設備驅動框架已經建立完成，接著是實際驅動編寫。

1. 完善file\_operations結構體中read()，write()，release()，open()函式。

(如圖四，圖五)



(圖四)



(圖五)

圖五部分主要是透過copy\_from\_user()來接收User Space的輸入訊息，再透過此訊息去操作蜂鳴器的開關。

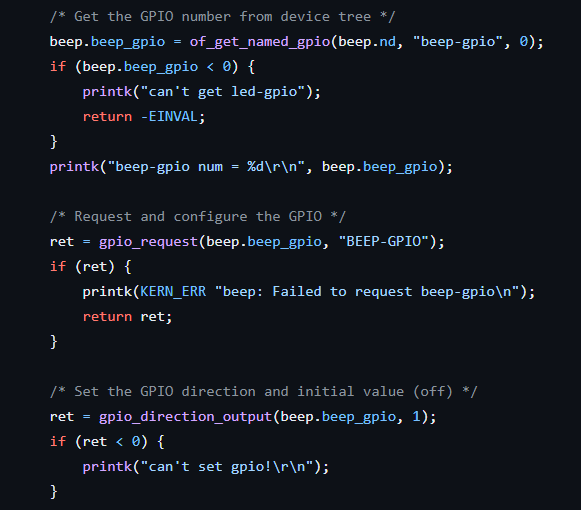
1. 最後完善驅動入口函數(led\_init)及驅動出口函數(led\_exit)就大功告成了。
   1. 驅動入口函數實作:

* 如圖六所示，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”beep”的這個節點。
* 如圖六所示，of\_property\_read\_string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。
* 如圖六所示，of\_property\_read\_string函數用於讀取屬性中的字符串值”status”，此處用於檢查設備是否可操作。



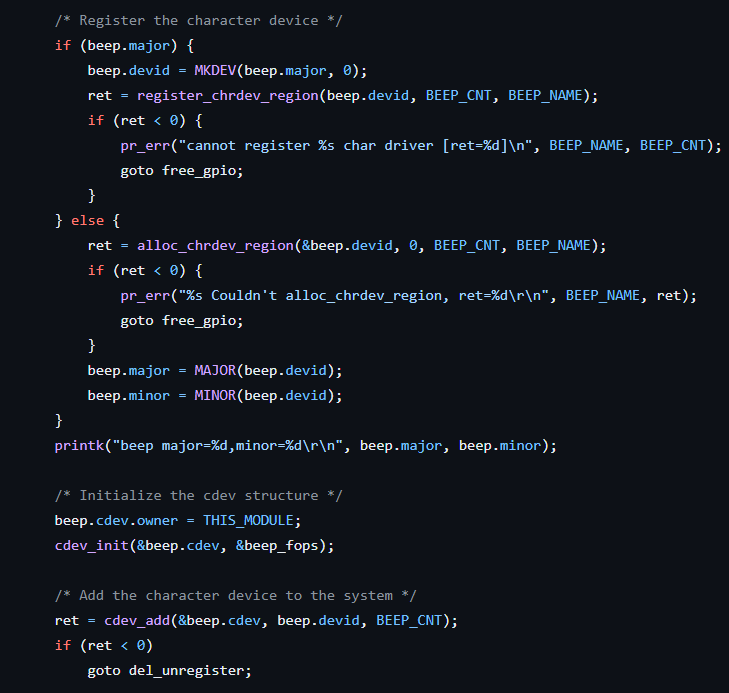
(圖六)

* 如圖七所示，of\_get\_name\_gpio()函數用來獲取gpio的編號。
* 如圖七所示，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。
* 如圖七所示，此函數用於設置GPIO為輸出。



(圖七)

* 如圖八所示，MKDEV()函數，用來創建設備號。
* 如圖八所示，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。
* 如圖八所示，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。
* 如圖八所示，cdev\_init()，用來初始化字元設備。
* 如圖八所示，cdev\_add()，用來新增字元設備到Linux系統中。



(圖八)

* 如圖九所示，class\_create()，用來自動創建設備節點，這樣就不需要手動mknod。
* 如圖九所示，device\_create()，用來自動創建設備節點。
* 如圖九所示，剩餘部分是當錯誤發生時，相對應的函式處理。



(圖九)

* 1. 最後是驅動出口函數實作: 當卸載驅動時對應的動作。
     + 如圖十所示，cdev\_del()，刪除一開始加入的字元設備。
     + 如圖十所示，unregister\_chrdev\_region()，用以註銷字元設備。
     + 如圖十所示，class\_destroy()，用來自動移除設備節點。
     + 如圖十所示，device\_destroy()，用來自動移除設備節點。
     + 如圖十所示，gpio\_free()，用來釋放先前申請的gpio。



(圖十)

※結論:

大致上字元設備驅動流程以及如何建立Kernel Space和User Space的溝通，還有透過Device Tree查找板子設備相關訊息，以及Linux 驅動的API就大功告成了。