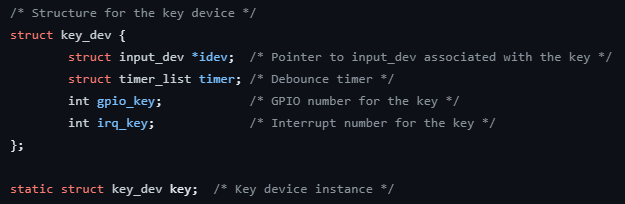
1. 專案20\_keyinput:

* 鍵盤，按鍵，滑鼠這些都屬於輸入設備，Linux Kernel中特別定義input子系統框架來處理輸入事件Input子系統本質上還是字元設備套上platform驅動框架的架構，本專案就實際編寫Linux Kernel中input子系統。

1. Input子系統簡介:

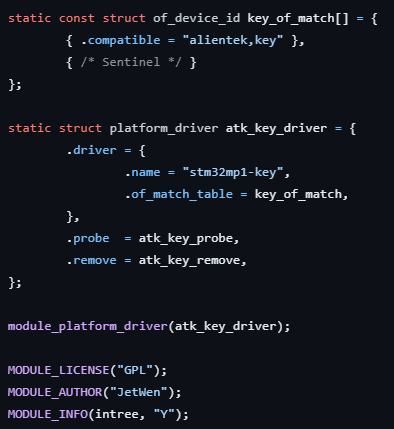
* 整個輸入設備其實包含硬體(滑鼠，鍵盤)，還有Kernel Space(驅動層，核心層，事件處理層)，最後才到User Space。不過對於驅動編寫人員來說，著重在Kernel Space即可，也就是驅動層，核心層，事件處理層。
* 驅動層：輸入設備的具體驅動程序，比如按鍵驅動程序，向核心層報告輸入內容。
* 核心層：承上啟下，為驅動層提供輸入設備註冊和操作接口。通知事件層對輸入事件進行處理。
* 事件層：主要和User Space進行溝通。

1. Input子系統實作:
2. 定義設備結構體，並在設備結構體中宣告input\_dev成員變數。(如下圖一)



(圖一:設備結構體)

1. 接著定義，id\_table及platform\_driver結構體，用來匹配設備和驅動。也包含了關於本驅動的相關信息。(如下圖二)



(圖二: id\_table及platform\_driver結構體內容)

1. 實現probe()函式內容，參數platform\_device \* pdev是指已經定義在Linux Kernel中的platform\_driver結構體變數(如下圖三)。

* key\_gpio\_init()函數用來初始化按鍵(如下圖四，圖五)。
* 呼叫timer\_setup初始化timer並綁定key\_timer\_function處理函數。(如下圖六)
* 透過呼叫input\_allocate\_device()函數分配一個新的輸入設備結構體。設置輸入設備的名稱為 KEYINPUT\_NAME。
* 輸入設備支持的事件類型，包括按鍵事件（EV\_KEY）和重複事件（EV\_REP）。呼叫input\_set\_capability()函數設置輸入設備的按鍵能力，指定支援KEY\_0鍵值。
* 最後呼叫input\_register\_device()函數將輸入設備註冊到輸入子系統中。

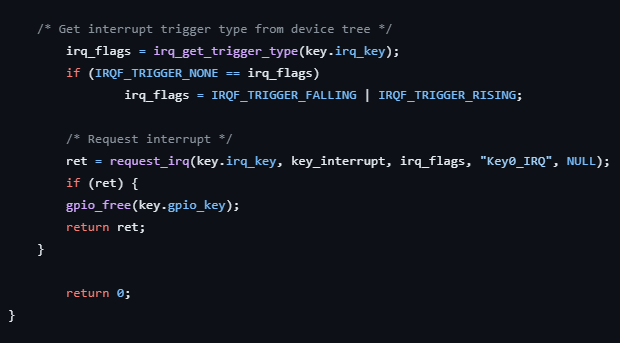


(圖三:probe()函式內容實現)

1. 完善key\_gpio\_init()函式，這部份是跟先前內容較相關，整體沒有使用到input子系統的API函式(如下圖四，圖五)。

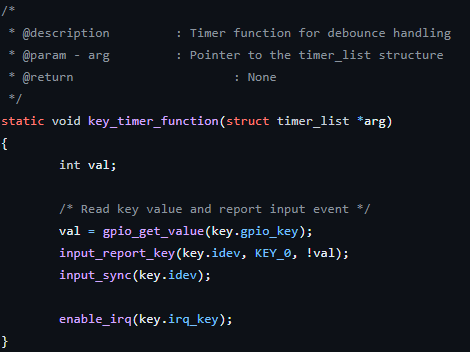


(圖四: key\_gpio\_init()函數內容)



(圖五: key\_gpio\_init()函數內容)

1. key\_timer\_function()函式完善，我們知道機械式案件往往會有抖動產生所以會造成訊號多次觸發的可能性，因此使用機械式按鍵時就需要透過timer function製造delay來忽略抖動訊號。(如下圖六)
   * 呼叫input\_report\_key()函數向輸入子系統報告按鍵事件。
   * 呼叫input\_sync()函數同步輸入事件，將事件傳遞給User Space。



(圖六:key\_timer\_function()函式內容)

※總結:

值得注意的是，本專案沒有實現驅動出入口函式，也沒有定義file\_operations結構體，因為Input子系統為大多數輸入設備提供了標準化的事件處理機制。

* 設備驅動程序只需要通過輸入子系統提供的接口（如input\_report\_key）來報告事件，而不需要直接實現文件操作方法（如open、read、write等）。
* evdev驅動已經實現了這些文件操作方法，並提供了統一的接口給用戶空間應用程序使用。設備驅動程序通過Input子系統向evdev報告事件，由evdev負責將事件傳遞給User Space。

(因為上述兩點我們就不需要定義file\_operations結構體了。)

* 輸入設備的註冊和註銷通過輸入子系統的API來完成，如:input\_register\_device和input\_unregister\_device。這些函數會自動處理設備的初始化和清理工作。

(因為上述原因我們就不需要驅動出入口函式了。)