Lab 7

Lab7.1使用cubieboard與ov7670連接

在本次的LAB我們將會教大家如何

1. 進入實驗六中所編譯的kernel目錄中

   cd ~where-is-your-home-folder-or-etc/linux-sunxi

2. 清除之前編譯過的東西

   make clean

3. 載入預設的設定檔案

   make sun7i_defconfig

4. 由於這一個鏡頭是經過一個csi(Camera Serial Interface)介面來連接的，所以我們必須要將驅動裝上cubieboard中，不過由於驅動有在linux-sunxi中內建，我們只要config中加入就好

   make menuconfig

   選取

   Device Drivers --->
   <*> Multimedia support --->
   <*>Video for Linux
   [*] CSI Driver Config for sun4i --->
   <M> OmniVision OV7670 sensor support

   (在這裡我們選擇將driver編譯為module)

5. 由於他原始碼有些問題，所以我們要做些修改如下

   開啟drivers/video/sunxi/disp底下的disp_layer.c

   vim drivers/video/sunxi/disp/disp_layer.c

   找到__s32 img_sw_para_to_reg(__u8 type, __u8 mode, __u8 value) 這一個function

   把下列行數註解掉(總共四個)

```
95      /* else {
96      DE_WRN("not supported yuv channel format:%d in "
97      "img_sw_para_to_reg\n", value);
98      return 0;
99      }*/

125     /* else {
126     DE_WRN("not supported yuv channel pixel sequence:%d "
127     "in img_sw_para_to_reg\n", value);
128     return 0;
129     }*/

168     /* else {
169     DE_WRN("not supported image0 pixel sequence:%d in "
```

```
170     "img_sw_para_to_reg\n", value);
171      return 0;
172     }*/

175     // DE_WRN("not supported type:%d in img_sw_para_to_reg\n", type);
```

6. 然後我們選取完成後，再將設定好的uImage編譯出來
   make menuconfig
   編譯完成一樣將uImage取代掉

7. 在來我們必須要把剛剛設為module的東西編譯出來
   make modules
   也是等待一段時間知後我們的模組就編譯好了

8. 然後我們需要用到的模組有4個，分別為
   drivers/media/video/videobuf-core.ko
   drivers/media/video/videobuf-dma-contig.ko
   drivers/media/video/sun4i_csi/device/ov7670.ko
   drivers/media/video/sun4i_csi/csi1/sun4i_csi1.ko
   將這4個檔案複製到img檔中(第二磁區rootfs)

9. 然後我們必須要配置script.bin檔
   我們的camera模組ov7670是連結到csi1的controll上，所以我們需要調整script.fex文件中的[csi1_para]和[camera_list_para]

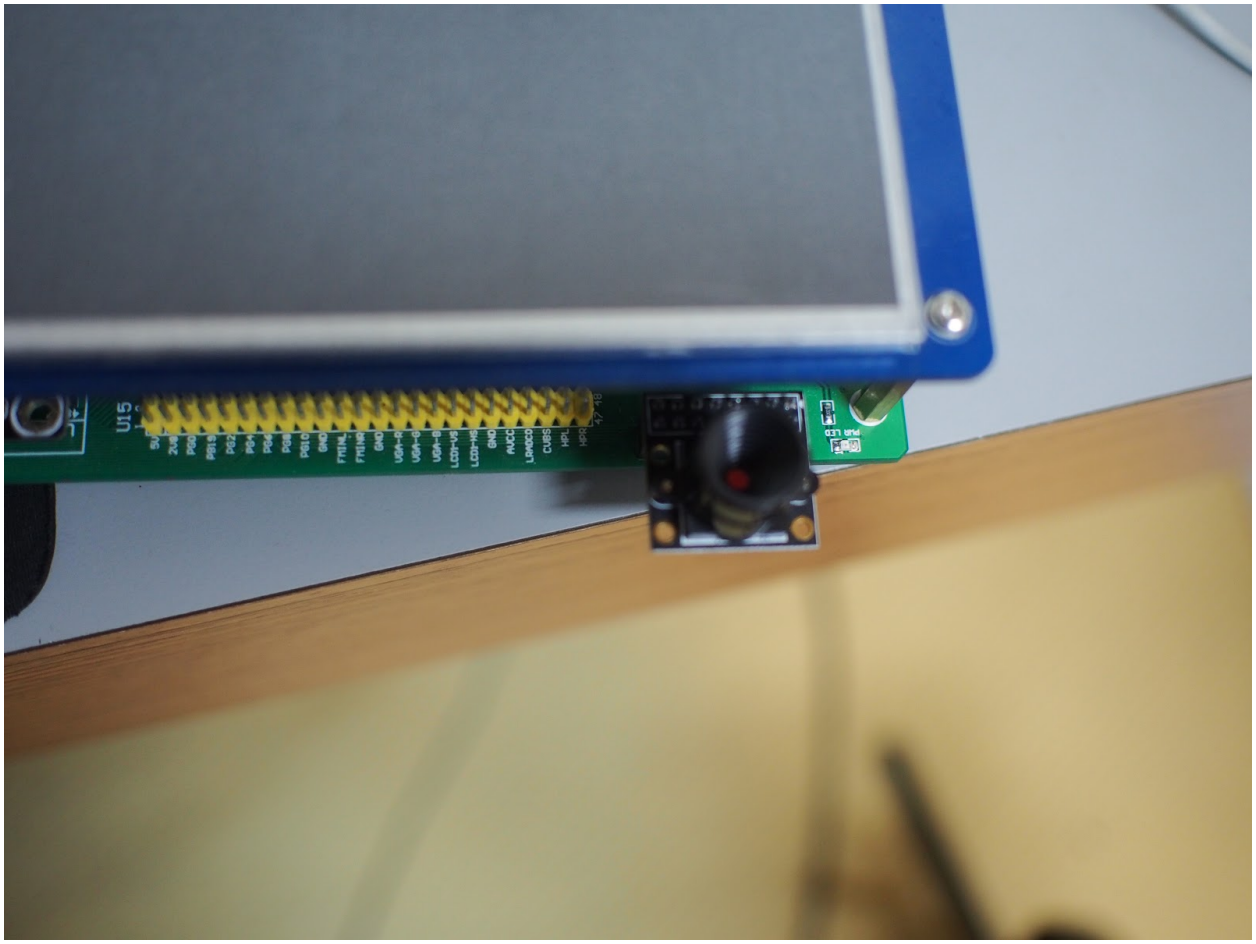
```
[csi1_para]
csi_used = 1
csi_dev_qty = 1  //csi介面連接幾台相機
csi_stby_mode = 0  //stand by 時是否關閉電源0為開啟,1為關閉
csi_mname = "ov7670"  //使用的模組名稱
csi_if = 0
csi_iovdd = "axp20_pll"
csi_avdd = ""
csi_dvdd = ""
csi_vol_iovdd = 2800
csi_vol_dvdd =
csi_vol_avdd =
csi_vflip = 1  //上下顛倒或是正常,0為正常,1為顛倒
csi_hflip = 0  //左右顛倒或是正常,0為正常,1為顛倒
csi_flash_pol = 1
csi_facing = 1
csi_twi_id = 1
csi_twi_addr = 0x42
```

csi_pck = port:PG00<3><default><default><default>
csi_ck = port:PG01<3><default><default><default>
csi_hsync = port:PG02<3><default><default><default>
csi_vsync = port:PG03<3><default><default><default>
csi_d0 = port:PG04<3><default><default><default>
csi_d1 = port:PG05<3><default><default><default>
csi_d2 = port:PG06<3><default><default><default>
csi_d3 = port:PG07<3><default><default><default>
csi_d4 = port:PG08<3><default><default><default>
csi_d5 = port:PG09<3><default><default><default>
csi_d6 = port:PG10<3><default><default><default>
csi_d7 = port:PG11<3><default><default><default>
csi_reset = port:PH13<1><default><default><0>
csi_power_en = port:PH16<1><default><default><0>
csi_stby = port:PH19<1><default><default><0>

[camera_list_para]
camera_list_para_used = 1
ov7670 = 1
gc0308 = 0
gt2005 = 0
hi704 = 0
sp0838 = 0
mt9m112 = 0
mt9m113 = 0
ov2655 = 0
hi253 = 0
gc0307 = 0
mt9d112 = 0
ov5640 = 0
gc2015 = 0
ov2643 = 0
gc0329 = 0
gc0309 = 0
tvp5150 = 0
s5k4ec = 0
ov5650_mv9335 = 0
siv121d = 0˙gc2035 = 0

10. 將配置好的script.bin以及各個檔案放入img檔,並且將上次的waveshore_demo裡的API/test_camera資料夾複製進rootfs中，後我們便可以開始燒錄sd卡

11. 將ov7670如圖安裝至cubieboard上



12. 燒錄完成後,開啟機器按照順序安裝驅動

    insmod videobuf-core.ko
    insmod videobuf-dma-contig.ko
    insmod ov7670.ko
    insmod sun4i_csi1.ko

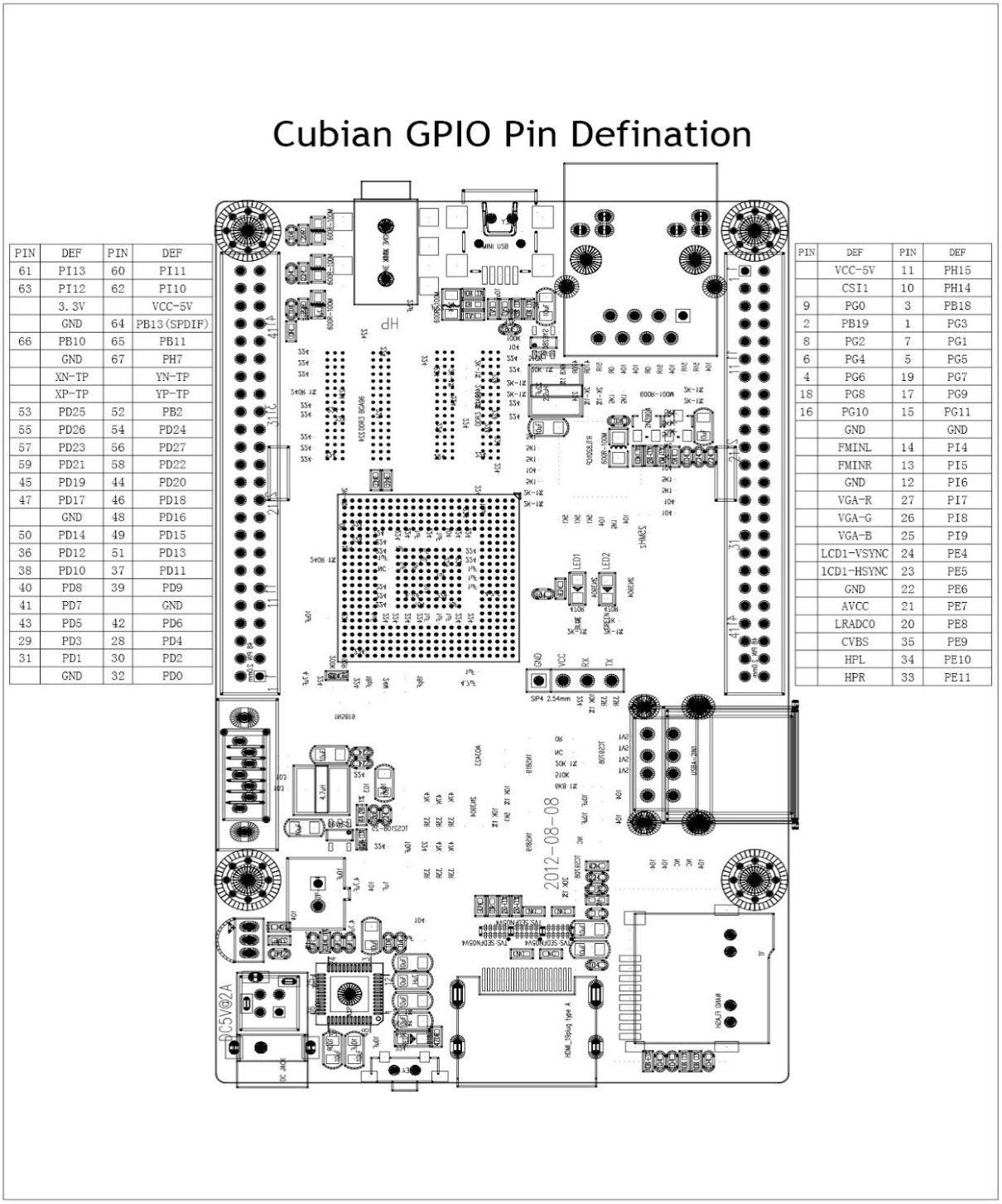
13. 執行編譯好的test_camera

    ./test_camera

    便可以看到相機顯示在螢幕上

PS:若想要使用ov7670與arduino在自己的專題上，可以參考下列文章或自行google!
http://nicolasfley.fast-page.org/?page_id=35
http://www.arducam.com/tutorial/

Lab7.2使用arduino與cubieboard做uart連線



PIN	DEF	PIN	DEF
61	PI13	60	PI11
63	PI12	62	PI10
	3.3V		VCC-5V
	GND	64	PB13(SPDIF)
66	PB10	65	PB11
	GND	67	PH7
	XN-TP		YN-TP
	XP-TP		YP-TP
53	PD25	52	PB2
55	PD26	54	PD24
57	PD23	56	PD27
59	PD21	58	PD22
45	PD19	44	PD20
47	PD17	46	PD18
	GND	48	PD16
50	PD14	49	PD15
36	PD12	51	PD13
38	PD10	37	PD11
40	PD8	39	PD9
41	PD7		GND
43	PD5	42	PD6
29	PD3	28	PD4
31	PD1	30	PD2
	GND	32	PD0

PIN	DEF	PIN	DEF
	VCC-5V	11	PH15
	CSI1	10	PH14
9	PG0	3	PB18
2	PB19	1	PG3
8	PG2	7	PG1
6	PG4	5	PG5
4	PG6	19	PG7
18	PG8	17	PG9
16	PG10	15	PG11
	GND		GND
	FMINL	14	PI4
	FMINR	13	PI5
	GND	12	PI6
	VGA-R	27	PI7
	VGA-G	26	PI8
	VGA-B	25	PI9
	LCD1-VSYNC	24	PE4
	1CD1-HSYNC	23	PE5
	GND	22	PE6
	AVCC	21	PE7
	LRADC0	20	PE8
	CVBS	35	PE9
	HPL	34	PE10
	HPR	33	PE11

在本實驗中我們必須要將cubieboard所傳到arduino的資訊寫到serial monitor中
arduino side：

```
#include<SoftwareSerial.h>

SoftwareSerial mySerial(2,3);


void setup() {
  // put your setup code here, to run once:
   pinMode(2, INPUT);
  pinMode(3, OUTPUT);
 mySerial.begin(9600);
 Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  if(mySerial.available())
  Serial.write(mySerial.read());
  if(Serial.available())
  mySerial.write(Serial.read());
}
```

在這個程式中由於我們想要保有我們的serial，這樣才可以用usb看到訊息，所以我們使用了softwareserial，然後我們將2,3分別定義為rx與tx，並且記得定義他的input output!!然後將Serial的東西做一個轉傳！
詳情可見：
http://www.arduino.cc/en/Reference/SoftwareSerial

cubieboard side:

在這一個部份我們必須將cubieboard的預設gpio資訊作修改(這裡我們使用cubieboard而非虛擬機端)

1. 首先我們先要開啟gpio的功能,作為uart，在在此之前我們必須要先下載一個程式來使用
   apt-get install git
   git clone git://github.com/linux-sunxi/sunxi-tools.git
   便可以下載到這個一個sunxi-tool的原始碼
2. 而我們在這一步需要來編譯這隻程式
   apt-get install make
   cd sunxi-tools
   make fex2bin
   make bin2fex
   如此一來我們就可以得到兩個程式
3. 再來我們必須要將開機磁區掛載起來
   mount /dev/mmcblk0p1 /mnt
   ls /mnt
   可以看到我們將要修改的檔案為script.bin

4. 再來我們要將bin檔轉成我們可以看得懂的fex檔
   ./bin2fex /mnt/script.bin test.fex
   便可以產生test.fex

5. 我們必須修改test.fex檔來開啟我們所需要的功能
   vim.tiny test.fex
   裡面可以看到許多它原本已經定義好的東西
   找到[uart_paraX](X為0-7的數字)
   在此我們使用[uart_para4]
   以下會有四個參數如下:
   [uart_para4]
   uart_used = 0
   uart_port = 4
   uart_type = 2
   uart_tx = port:PG10<4><1><default><default>
   uart_rx = port:PG11<4><1><default><default>
   先來簡單的介紹一下這一個文件的格式
   [gpio_nameX]
   gpio_used = 1
   etc …..
   gpio_num = 4
   gpio_pin_1 = port:PG00<1><default><default><default>
   gpio_pin_2 = port:PB19<1><default><default><default>
   gpio_pin_3 = port:PG02<0><default><default><default>
   gpio_pin_4 = port:PG04<0><default><default><default>
   大部分的都會長成這樣，首先第一項gpio_used為是否開啟這一個設定值，開啟的話設為1,若不開啟的話則設為0，etc為可能針對某些特殊的用途所定義的，下一項gpio_num為定義此function會用到幾隻腳，而按照範例為4隻腳,故接下來就要來定義這四隻腳是用來幹嘛的
   格式為下：
   gpio_pin_1 =port:<port><mux feature><pullup/down><drive capability><output level>
   <port> 是你要使用的port的腳 (ie. PH15)
   <mux feature> 是對這一個腳位,0的話為input，1的話為output，而2-7為特殊的定義，http://www.linux-sunxi.org/A20/PIO這是它的表,可以自己查閱應該如何設定
   <pullup/down> 當 0為關閉； 1 的時候為pullup時觸發； 2的時候為pulldown時觸發(只有當input時有效)
   <drive capability> 定義output的最大電流(單位mA)，0-3 分別為 10mA, 20mA, 30mA and 40mA.
   <output level>設定初始電壓，0 為低電位； 1為高電位 (只有為output時有效)
   The **<pullup/down> <drive capability> <output level>** 能夠定義為<default> 為預設值

若還有不懂得可以參考此文件
http://linux-sunxi.org/Fex_Guide
故我們在此中更動第一個選項,就可以開啟預設的uart
uart_used = 1
然後存檔
PS建議使用uart5不然會發生一些奇怪的事情(PI10&PI11)，並且要確定script.bin裡沒有衝突的腳位。

6. 然後我們必須要將檔案轉回去bin
   ./fen2bin test.fex /mnt/script.fex
   便可以將檔案轉換完成且覆蓋原來的
7. 然後我們將開機磁區卸載
   umount /mnt
8. 重新開機
   reboot
9. 在次登入系統後,執行以下的指令來確定自己有沒有成功開啟uart
   dmesg | grep tty
   它就會顯示你所開啟的uart對應到的是哪一個tty
   stty -F /dev/ttySX -a
   用這一個指令確定是否tty成功設定,若沒有沒設定成功,他會顯示下列錯誤訊息
   stty: /dev/ttySX: Input/output error

10. 撰寫程式
    我們可以寫一個程式來控制它

```
#include<stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <termios.h>
#include<stdlib.h>
#include<sys/ioctl.h>
int main (){
  struct termios toptions;
  int fd;
  fd = open("/dev/ttySX", O_RDWR | O_NOCTTY);//取決於你上面抓到的ttySX
  speed_t brate = 9600;
  cfsetispeed(&toptions, brate);
  cfsetospeed(&toptions, brate);
  while(1){
  write(fd,"H",1);
  sleep (1);
  }
  close(fd);
}
```

//這支程式是讓他每秒寫入1個H

```
#include<stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <termios.h>
#include<stdlib.h>
#include<sys/ioctl.h>
int main (){
  struct termios toptions;
  int fd;
  fd = open("/dev/ttySX", O_RDWR | O_NOCTTY);//取決於你上面抓到的ttySX
  speed_t brate = 9600;
  cfsetispeed(&toptions, brate);
  cfsetospeed(&toptions, brate);
  while(1){
  char buf[2];
  int i =read(fd,buf,1);//i為實際回的數值
  if(i>0)
    std:cout<<buf<<endl;
  }
  close(fd);
}
```

//這支程式是讀取一個byte的範例
並且編譯它
並且確定是否有權限,如果沒有的話

chmod 0666 /dev/ttySX

11. 連接arduino以及cubieboard
    準備好電位轉換器
    由於大多的arm開發板所用的電壓為3.3v而arduino所使用的5v
    而要彌補這一段差距我們必須使用電位轉換器
    但是由於我們的cubieboard是使用5v的,故在這裡我們可以不用做這一個步驟
    直接將剛剛設定的好的tx,rx對接即可
    亦即為
    cubieboard:tx<------->rx:Arduino
    rx←------->tx
12. 執行剛剛編譯好的程式,我們就可以在arduino的serial monitor上看到相對應的結果!由於uart很容易發生，所以大家可能會常常看到亂碼！可能在要code中做一些處理！

Demo:
Basic 1(30%):
執行test_camera即可
Basic 2(40%):
寫一個one byte聊天室(arduino與cubieboard之間自動化）
首先,cubieboard先傳一個H給Arduino,Arduino收到之後顯示Cubie Say Hello!
然後Arduino也傳送一個H給cubieboard，cubieboard收到顯示Arduino Say Hello!
然後cubieboard傳一個G給Arduino，Arduino收到之後顯示Cubie Say Goodbye!
然後Arduino也傳送一個G給cubieboard，cubieboard收到顯示Arduino Say Goodbye!
程式結束！！

Bonus(20%):
Cubieboard單鍵照相機(將相機抓到的data以一般通用格式儲存)
參考http://www.twam.info/linux/v4l2grab-grabbing-jpegs-from-v4l2-devices
（提示）
在這裡大家可以考慮使用助教修改過的v4l2grab
git clone https://github.com/fucxy/v4l2grab.git
sudo apt-get install libv4l-dev libjpeg-dev
cd v4l2grab
gcc v4l2grab.c -o v4l2grab -Wall -ljpeg -DIO_READ -DIO_MMAP -DIO_USERPTR
./v4l2grab -o image.jpg
就可以得到image.jpg了
只要我們在用上一次的io_control就可以做成一個簡單的照相機了！
（記得要確認devicename=/dev/videoX是否有對應到！）
PS: 如果需要編譯程式時直接vim Makefile將linux路徑改為原始碼的路徑即可
git clone https://github.com/linux-sunxi/linux-sunxi.git(這一個東西即為原始碼)