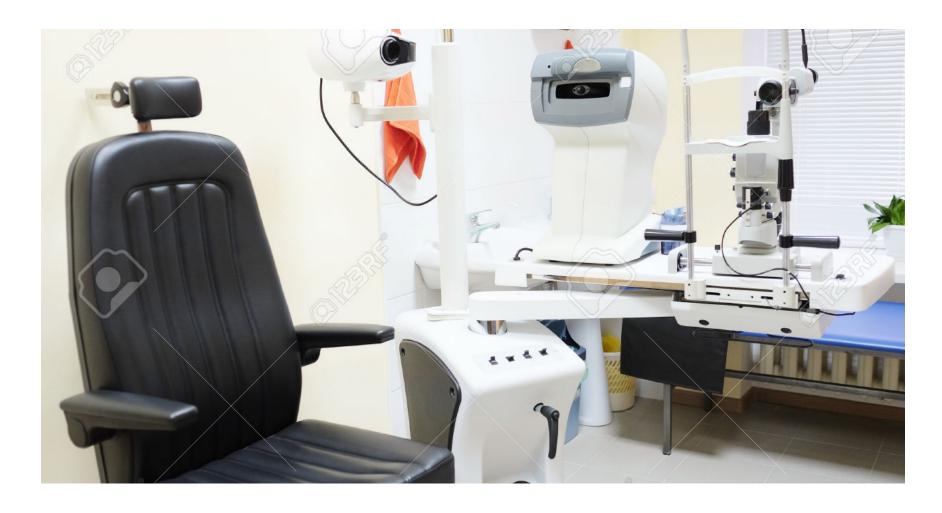
DC LAB Final Project

Team04 B02901093 吳 岳 B02901097 張哲銘 B02901098 宇德原

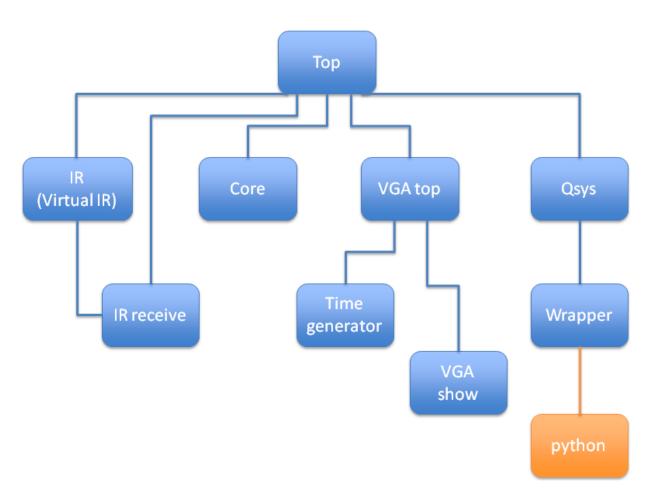
Outline

- Introduction
- System Structure
- Modules
- IR
- Core
- VGA
- Wrapper
- Python Program

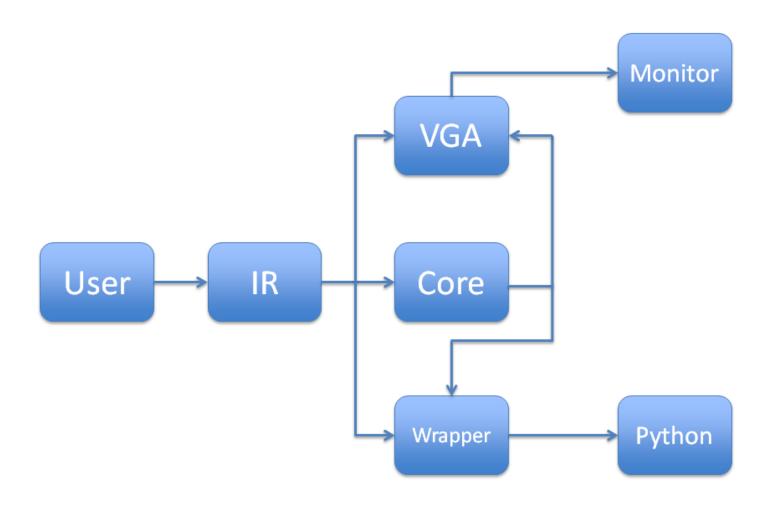
Introduction



System Structure



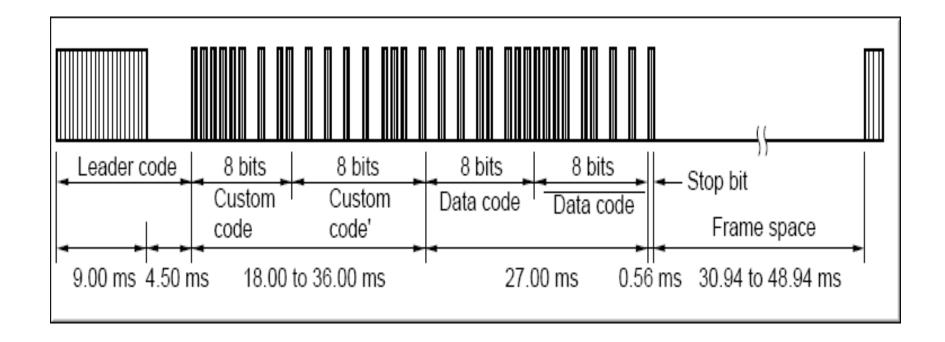
Modules



IR

- IR Receive:接收紅外線遙控器傳來的訊號、判讀 使用者所按的按鍵
- Virtual IR:將訊號轉換給 Core、VGA、Wrapper

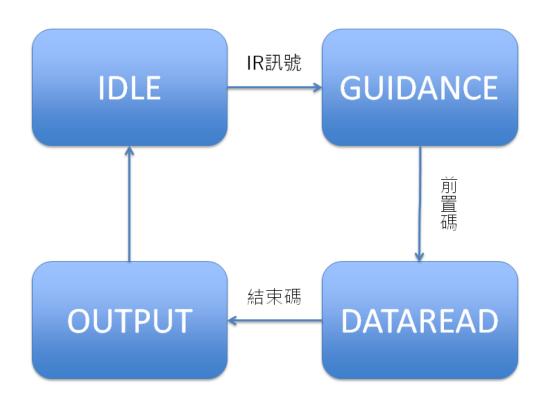
NEC Protocol



Table

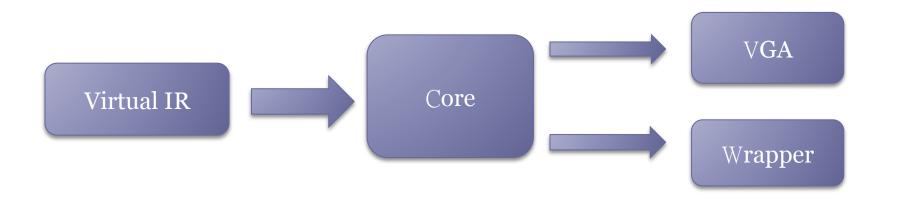
遙控按鍵	0	1	2	3	4	5	6
按鍵碼 (8-bit)	00	01	02	03	04	05	06
遙控按鍵	7	8	9	A	В	Power	
按鍵碼 (8-bit)	07	08	09	oF	13	12	

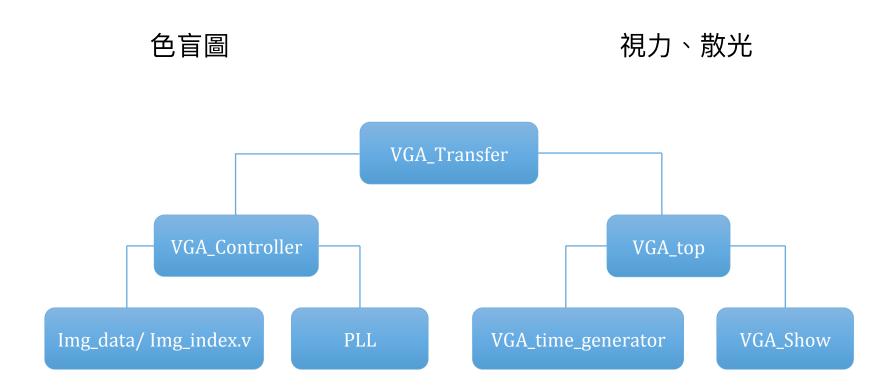
State Machine



Core

- Coordinate all modules and indicate the state
- Input by Virtual IR: Yes_or_No, User_direction
- Output to VGA: size, direction, state, result
- Output to Wrapper: start_to_send, all user data

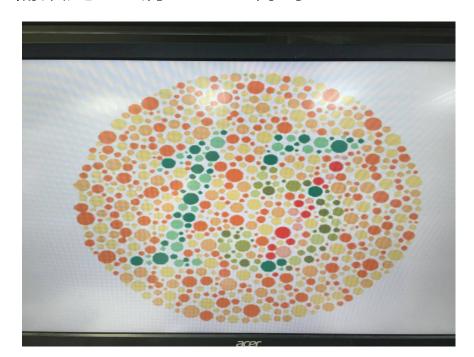




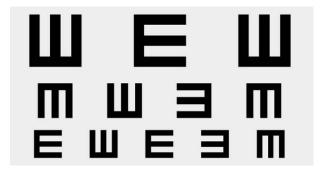
- 色盲圖圖取
 - 。底圖的資訊是由mif檔輸出的
 - 256 index
 - □ 307200 (640*480) 座標點

```
img_data_qq.mif
                                                  index_qq.mif
  WIDTH = 8;
                                                 WIDTH = 24;
  DEPTH = 307200;
                                                 DEPTH = 256;
  ADDRESS_RADIX = HEX;
                                                 ADDRESS_RADIX = HEX;
  DATA_RADIX = HEX;
                                                 DATA_RADIX = HEX;
  CONTENT BEGIN
                                                 CONTENT BEGIN
  0:0;
                                                 0:ffffff;
  1:0;
                                                 1:dfffff;
  2:0;
                                                 2:bfdfff;
  3:0;
                                                 3:9fdfff;
  4:0;
                                                 4:9fbfff;
  5:0;
                                                 5:7fbfff;
  6:0;
                                                 6:5fbfff;
  7:0;
                                                 7:7fdfff;
  8:0;
                                                 8:bfffff;
  9:0;
  a:0;
                                                 a:dfdfff;
  b:0;
                                                 b:7f9fff;
  c:0;
                                                 c:3f7fff;
  d:0;
                                                 d:5f7fff;
  e:0;
                                                 e:bfbfff;
  f:0;
                                                 f:9f9fff;
  10:0;
                                                 10:bfdfdf;
  11:0;
                                                 11:bfdfbf;
  12:0;
                                                 12:9fdf9f;
  13:0;
                                                 13:9fbf9f;
  14:0;
                                                 14:dfffdf;
  15:0;
                                                 15:dfdfdf;
  16:0;
                                                 16:9fbf7f;
```

- 色盲圖圖取
 - □ png/jpg -> bmp -> mif檔
 - □ VGA只能用25MHz的Clk,因此需設定PLL將50MHz轉為25 MHz



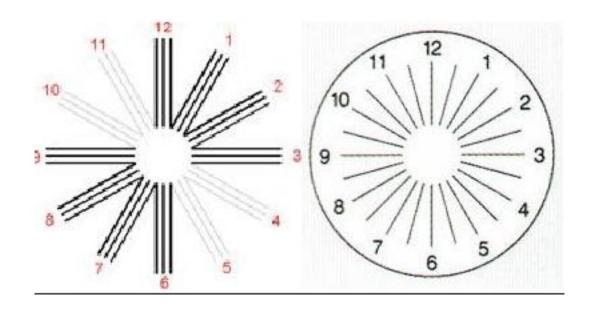
- 視力表原理
 - E-Chart
 - □ 標準規格是將視角5分度(5 minutes of arc)的大小定義為「判斷視力1.0」的E-chart
 - E-chart 6m, 0.1的E-chart圖換算後為88.6mm
 - 0.1 E-Chart 之長度 * 68.75 = 受測者跟量測表的距離



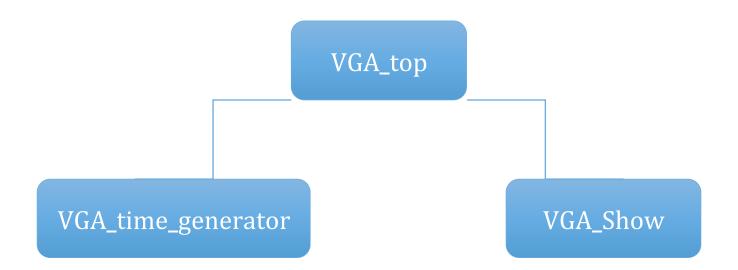
• 視力表原理

Vision	Pixel	Vision	Pixel
0.1	120	0.7	17
0.2	60	0.8	15
0.3	40	0.9	13
0.4	30	1.0	12
0.5	24	1.2	10
0.6	20	1.5	8
		2.0	6

- 散光表
 - □ 潛在散光的受試者會認為垂直跟水瓶的線條粗細是不均匀的



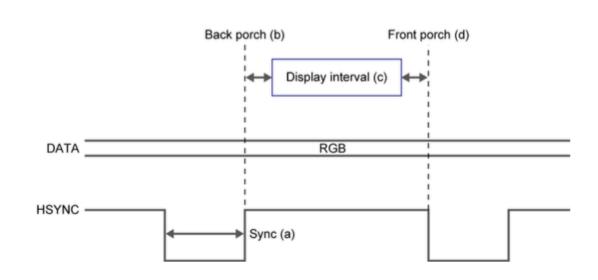
• 視力、散光在VGA 架構上的整合



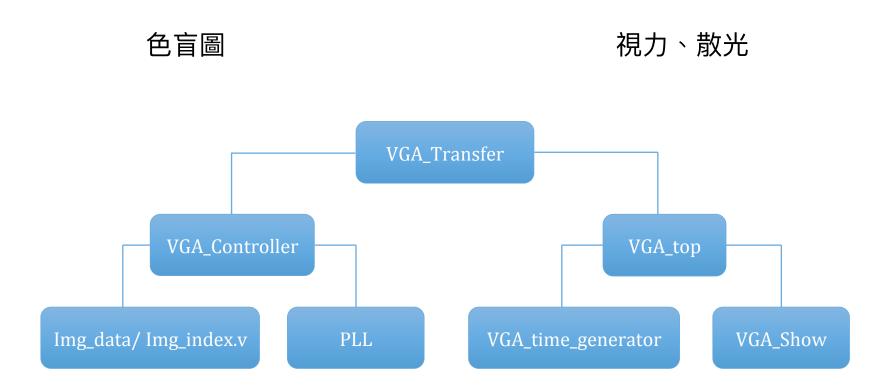
• 視力、散光在VGA 架構上的整合

VGA mode		Horizontal Timing Spec				
Configuration	Resolution(HxV)	a(us)	b(us)	c(us)	d(us)	Pixel clock(MHz)
VGA(60Hz)	640x480	3.8	1.9	25.4	0.6	25

```
vga_time_generator vga0(
    .pixel_clk(i_clk),
    .h_disp (640),
    .h_fporch (16),
    .h_sync (96),
    .h_bporch (48),
    .v_disp (480),
    .v_fporch (10),
    .v_sync (2),
    .v_bporch (33),
    .vga_hs
            (vga_h_sync),
    .vga_vs
            (vga_v_sync),
    .vga_blank(inDisplayArea),
    .CounterY(CounterY),
    .CounterX(CounterX)
);
```



• 黑白(視力&散光)跟彩色(色盲底圖)的整合



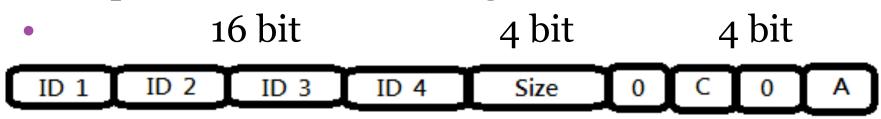
Wrapper

Wrapper額外傳RS232那些需要寫檔的資料(視力、散光、 色盲檢測結果)

```
module Rsa256Wrapper(
   input avm_rst,
   input avm clk,
   output [4:0] avm address,
   output avm_read,
   input [31:0] avm_readdata,
   output avm_write,
   output [31:0] avm_writedata,
   input avm_waitrequest,
   input [3:0] i_ID1,
   input [3:0] i_ID2,
   input [3:0] i_ID3,
   input [3:0] i_ID4,
   input [3:0] i_size,
   input
                i_astigmatism_result,
                i_start_to_send,
    input
                 color_test_result
    input
```

Python Program

- Process the data coming from Wrapper
- Output the result into designated file



Python receive it by 8 bits unit

Demo

