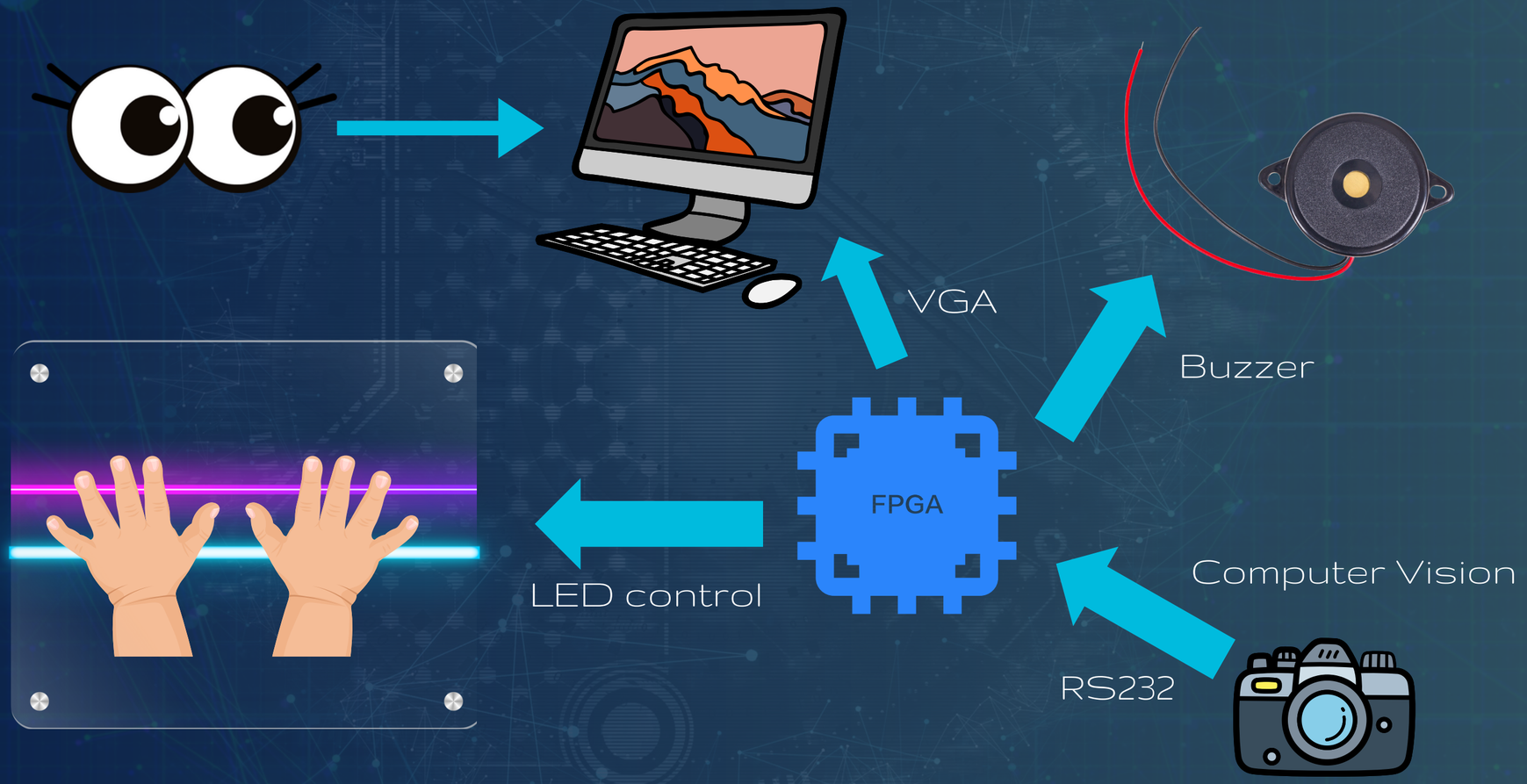
DC Lab

MUSIC INTERACTIVE GAME

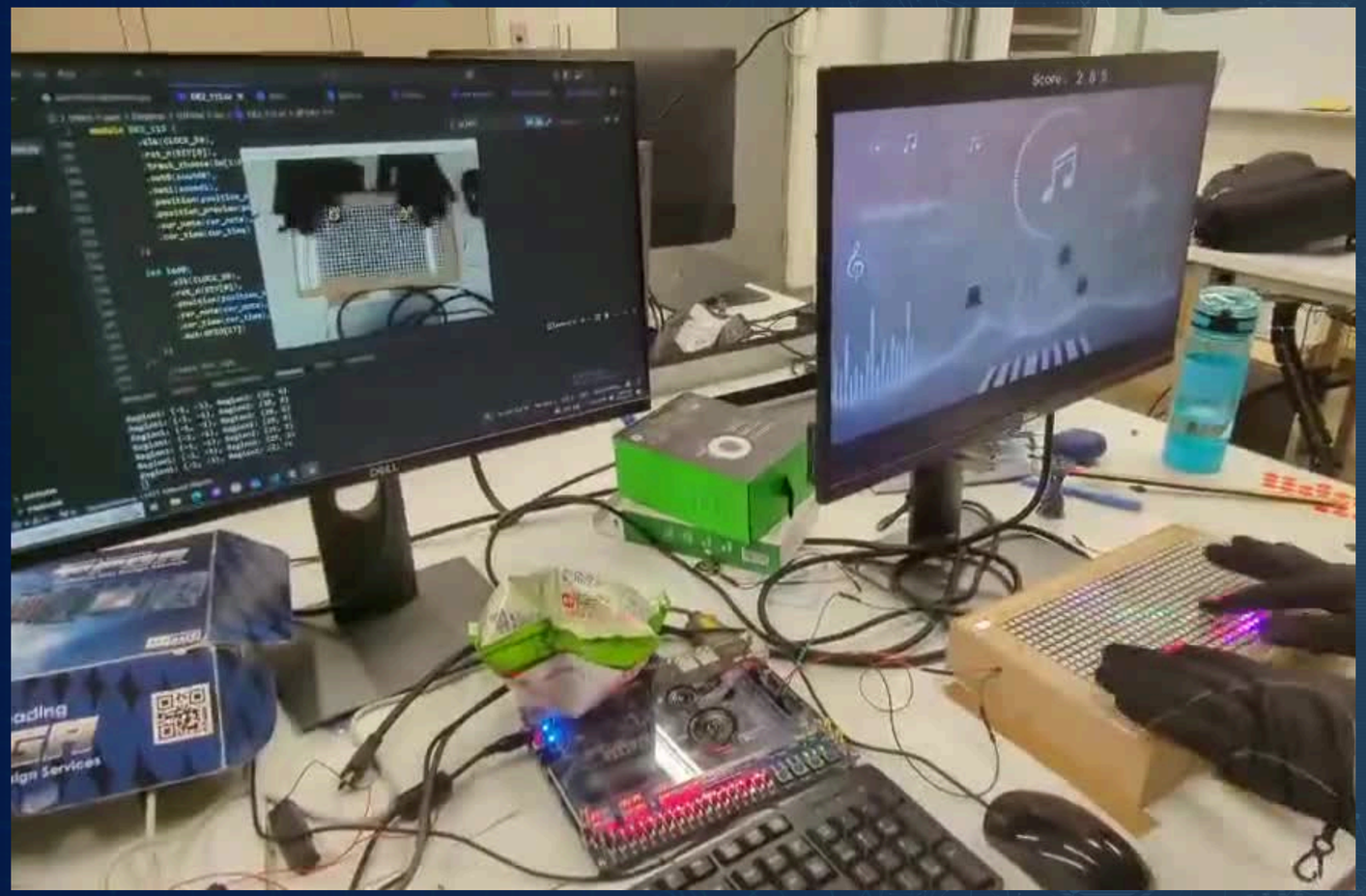
Team 02
B10901091李冠儀、B10901082陳英睿、B10901090洪乾峰

Start

Introduction



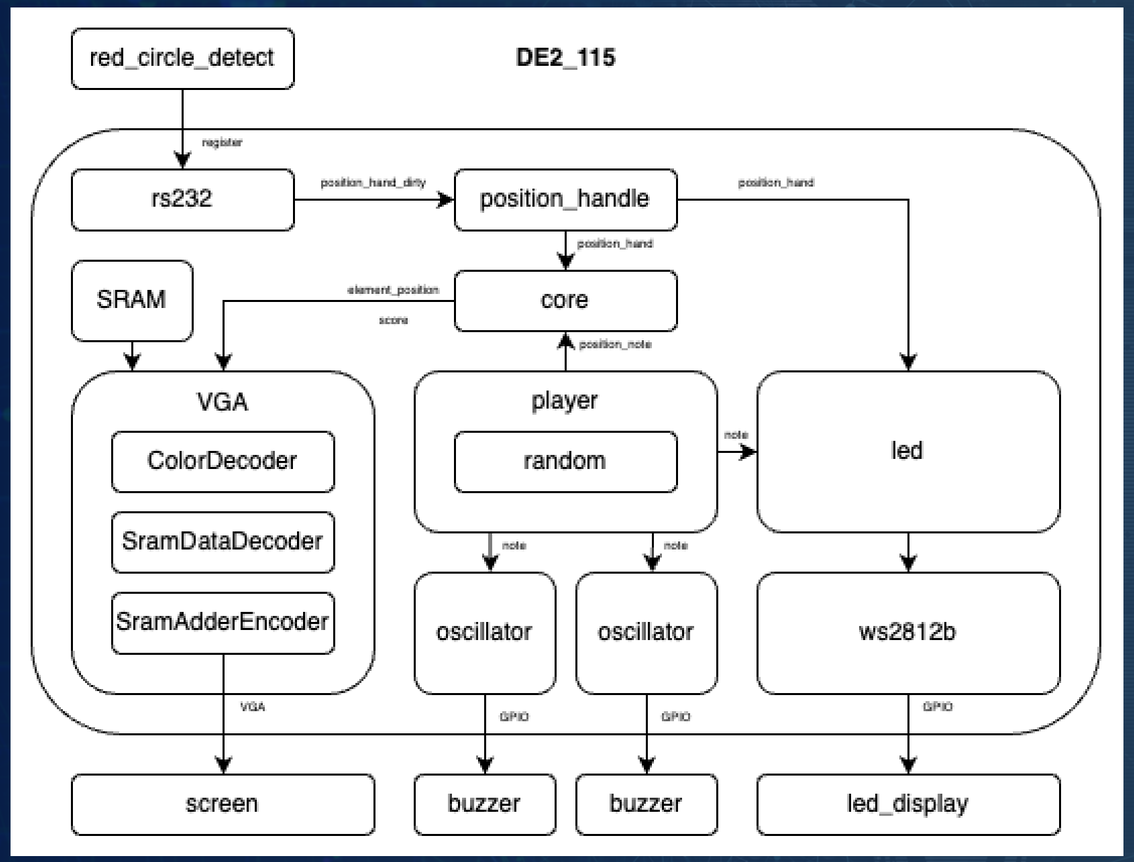
Demo

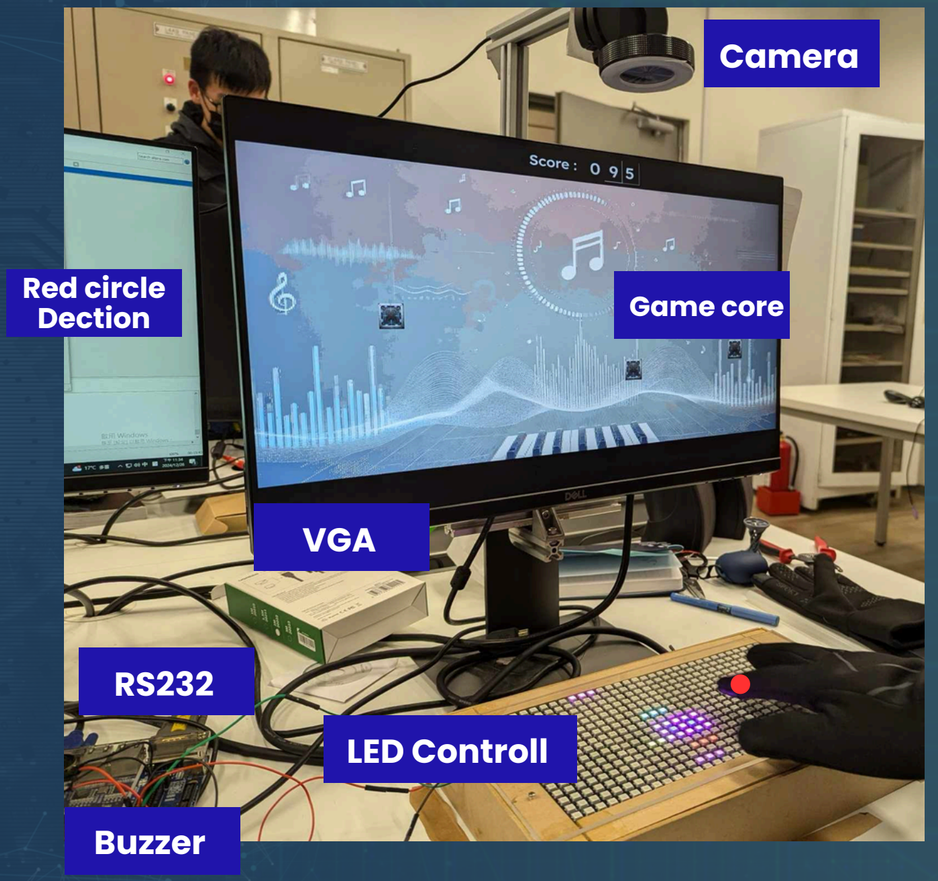


Spec:

- **Support switching between different soundtracks and levels.**
- **Support 8 difficulty levels.**
- **Video recognition detection.**
- **Real-time scoreboard.**
- **Real-time LED feedback.**

System Architecture





Main Difficulties

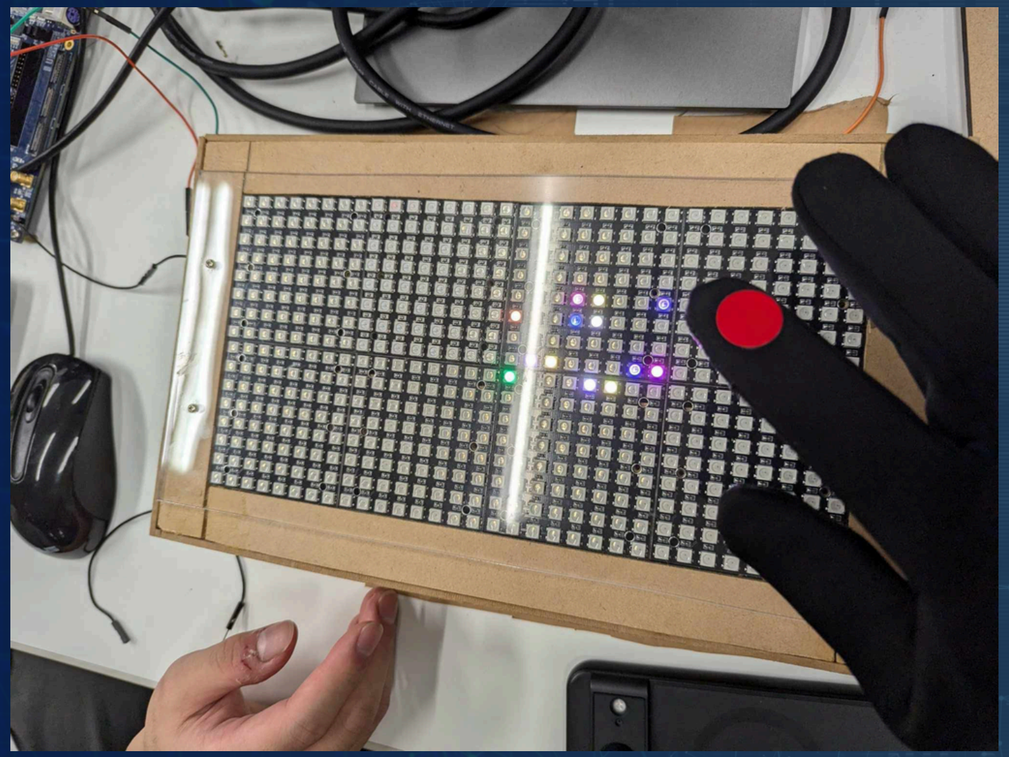
Combine several module/IP : RS232/VGA/Buzzer/Camera/LED/Game Core

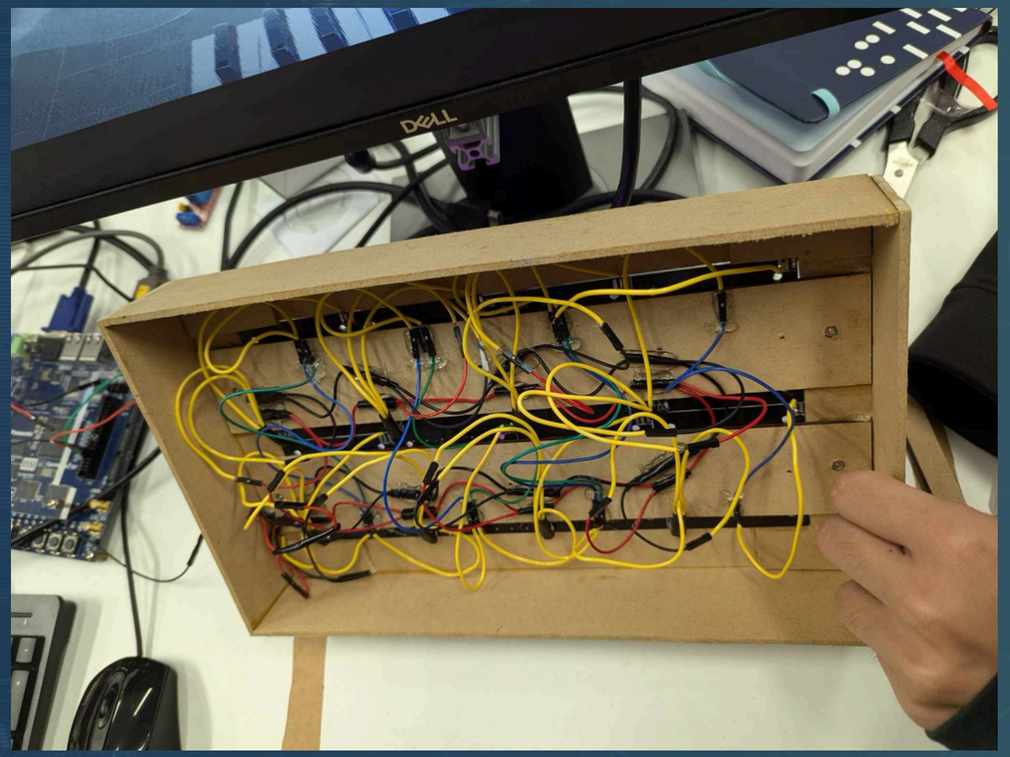
All Real-time !

- Music Controll & Phonetic Symbol Position Logic
- VGA Display
- Hand Dection
- Hand Trajectory Compute
- Rs232 Comunication
- LED Light Response

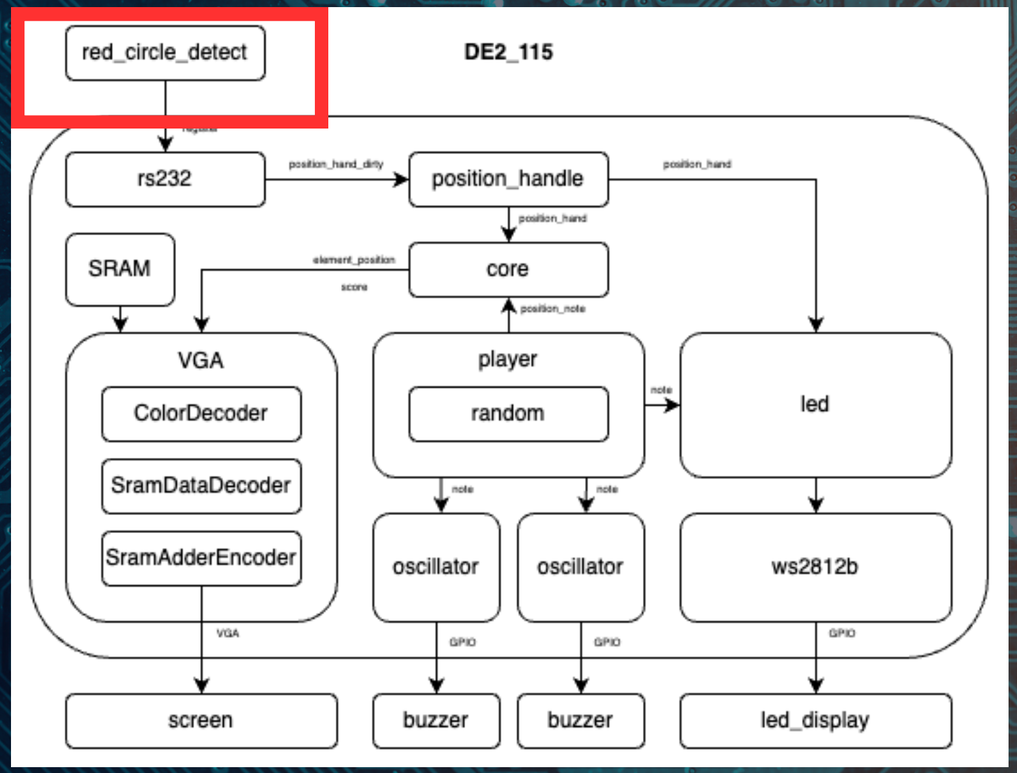
We achieve “no perceptible latency to humans”

Hardware: LED Panel

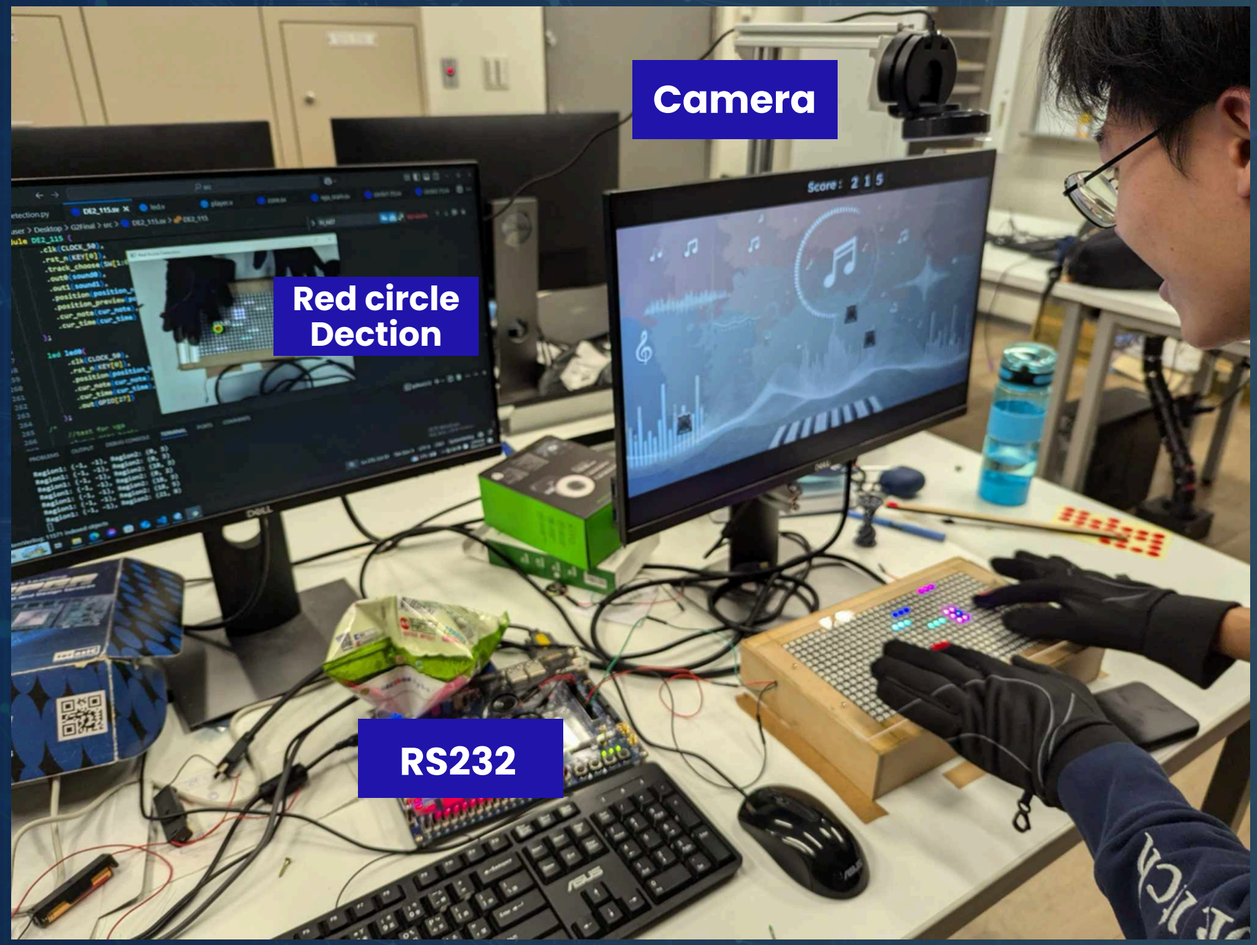




Camera & Image Process



Camera & Image Process



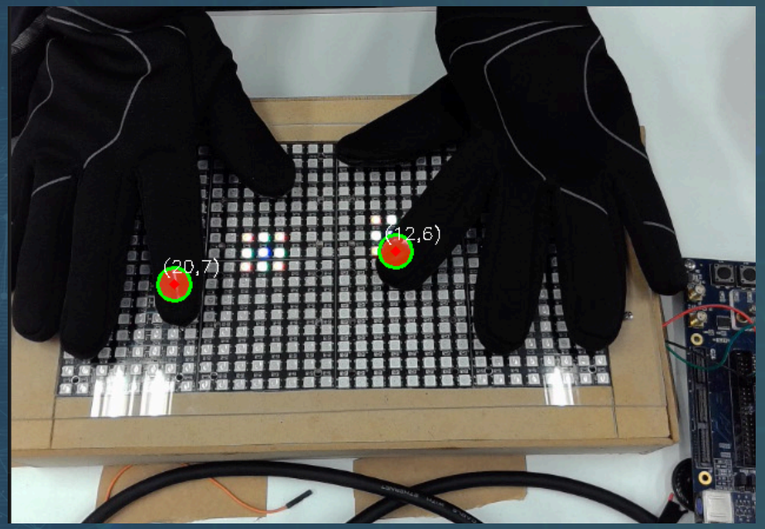
Camera & Image Process

- **YOLO or other DL method is too slow!!**

- **We decide to use traditional Computer Vision Algorithm**
  - **Archieve precision 100% & recall Rate ~50%**
  - **Detection is real-time, can be passed to FPGA by rs232 immediately, latency < 10ms.**

Camera & Image Process

- **We decide to use traditional Computer Vision Algorithm**
  a. **Convert the image from BGRA to BGR.**
  b. **First blur to clear up noise.**
  c. **Convert to LAB color space.**
  d. **Threshold for red colors.**
  e. **Second blur to clear up noise again.**
  f. **Use Hough transformation to detect circles of defined size.**
  g. **Serial out through RS232**

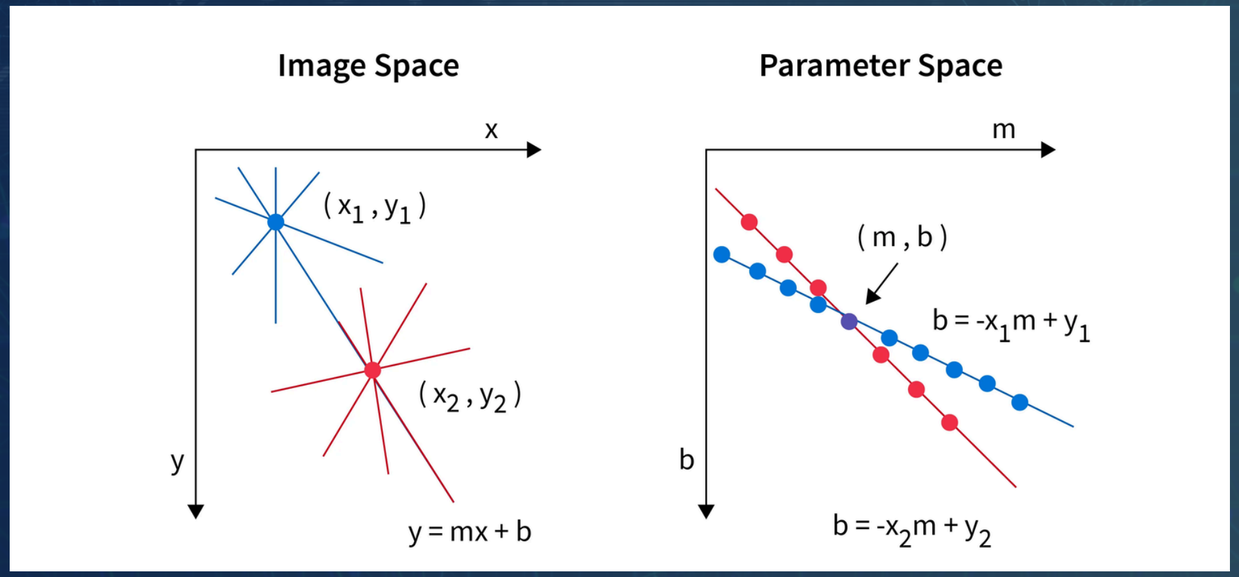


Camera & Image Process

- **Hough Transformation**

$$(x-a)^2+(y-b)^2=r^2$$

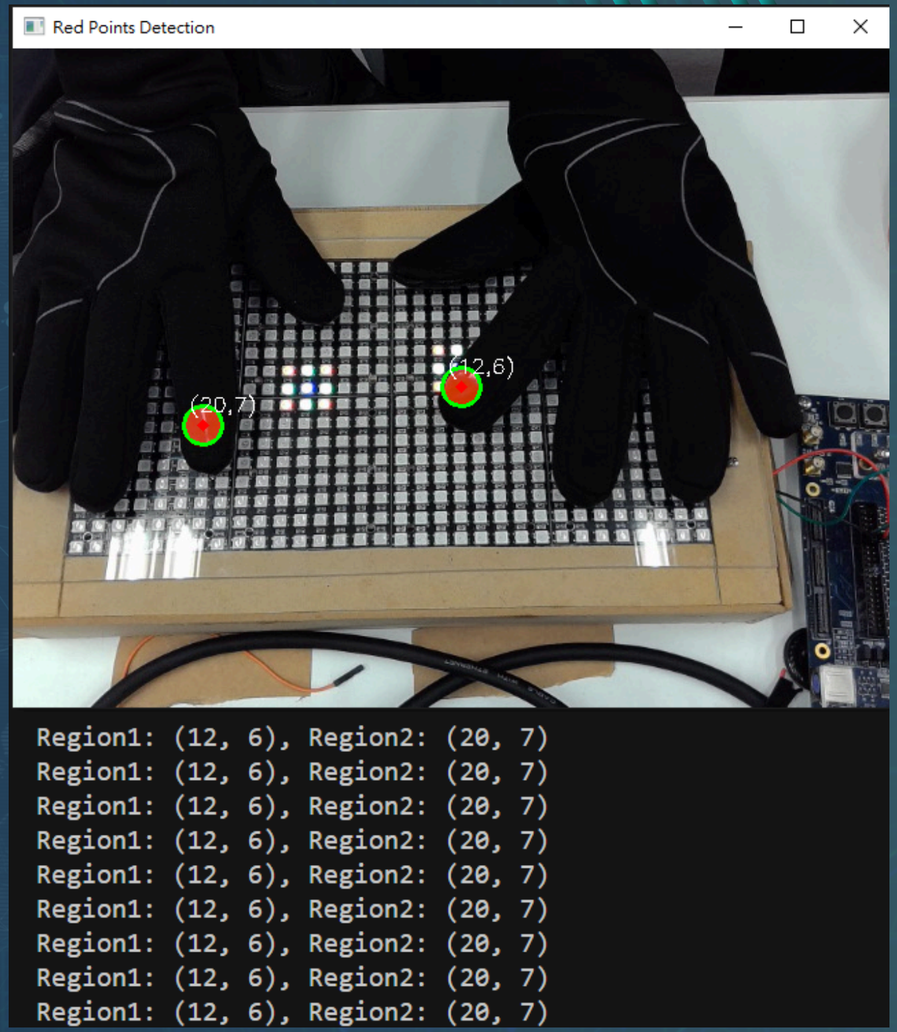
  - Convert to parameter space (a,b,r), we can detect the **circle of certain radius r.**



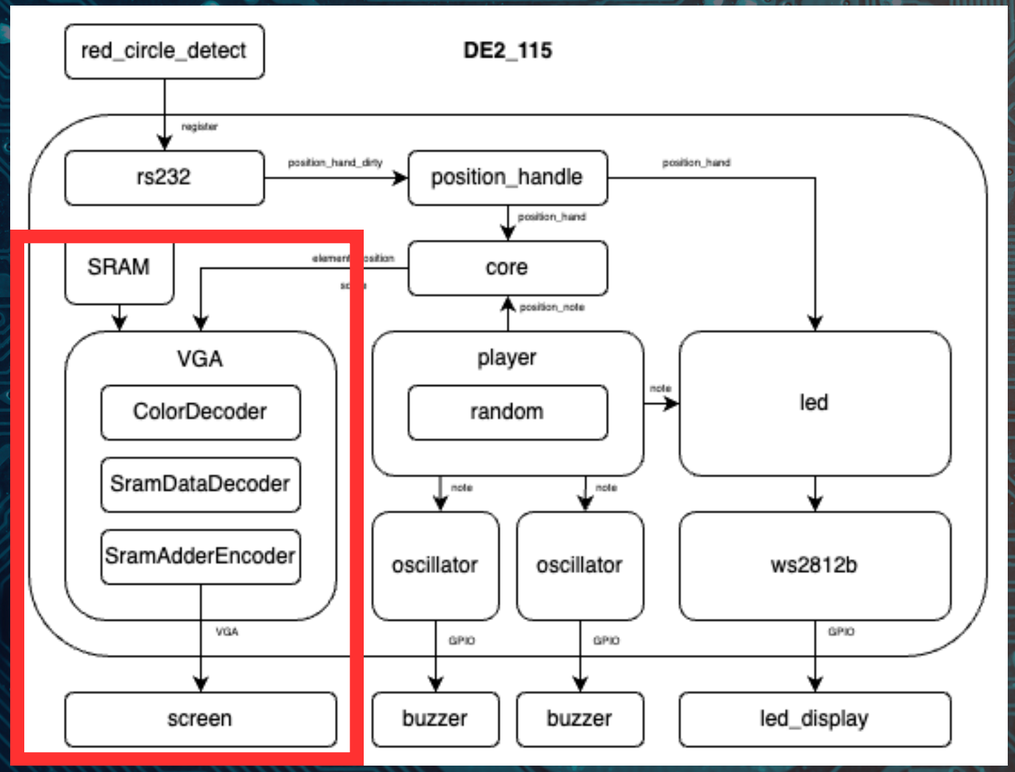
Camera

- **LAB**
  - L: Lightness
  - A: Green-to-Red axis
  - B: Blue-to-Yellow axis
  - Better alignment with human visual perception. Simplifies red-green differentiation via A channel.

- **Coordinate interpretation**
  - We set two regions, cast the most likely red dot position, and use rs232 to return to FPGA.



Game Screen (VGA)

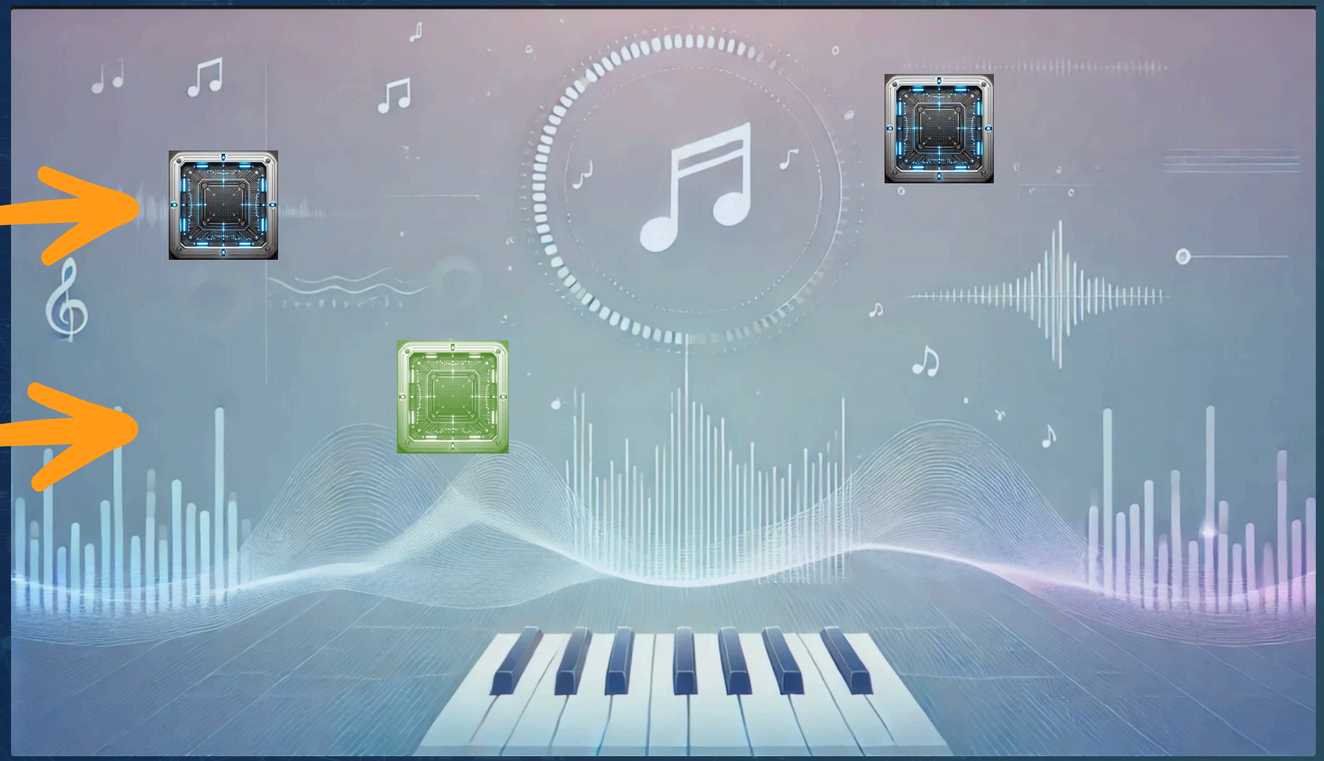


VGA

- Our Game Screen:

Symbol

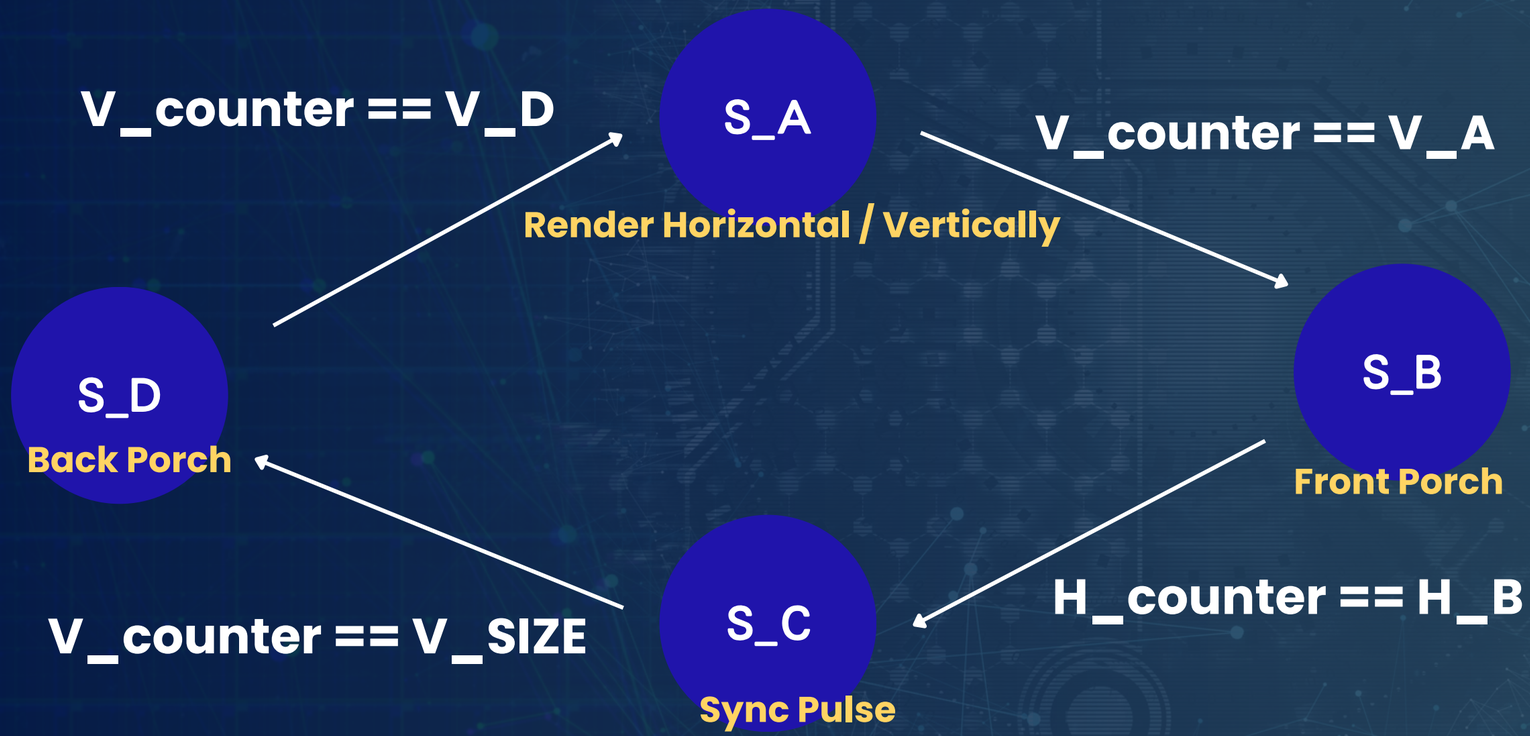
Background

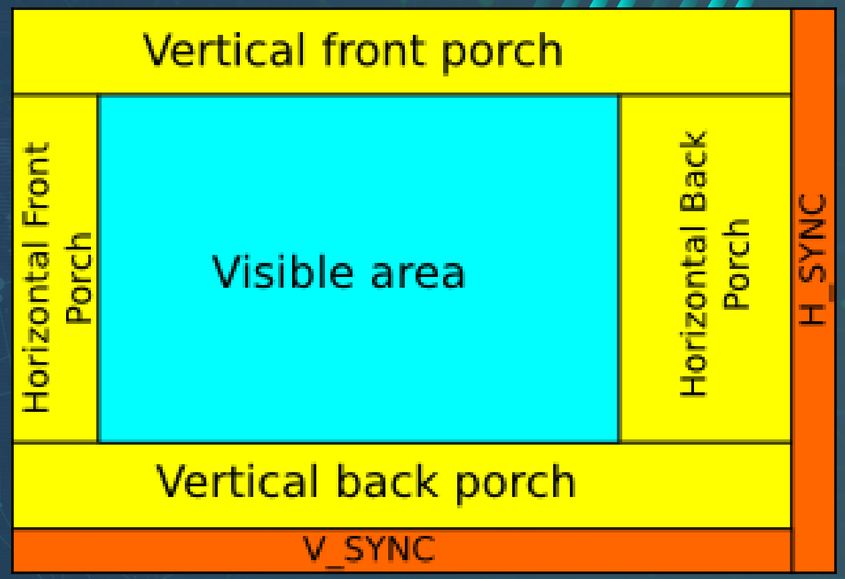


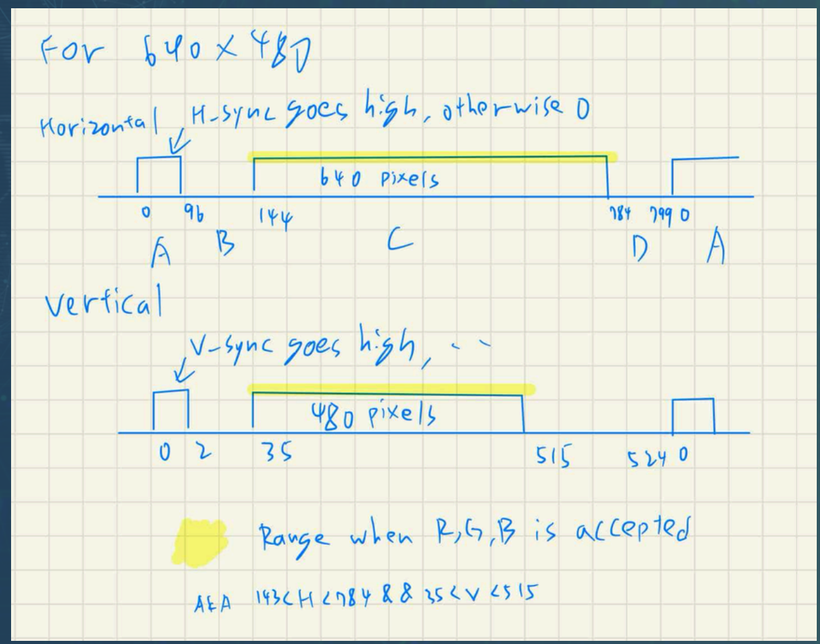
- We save our background to SRAM, then read the background data from SRAM and display it on VGA when need.
- Findings: Need to check Counter start from 0 or 1

VGA

- **Finite state machine**







VGA - Layer Set

```
if (H_STATE == S_C && V_STATE == S_C) begin
    // // Render circle1 image
    if ((H_rel1 >= 1) && (H_rel1 < `IMAGE_H+1) && (V_rel1 >= 1) && (V_rel1 < `IMAGE_V+1) && circle1_show == 1) begin
       RGB_nxt = color_circle1[pixel_data_circle1[V_rel1][H_rel1]];
       object_pixel_index_before_sram_nxt = object_pixel_index_before_sram + 1;
    end
    // Render circle2 image
    else if ((H_rel2 >= 1) && (H_rel2 < `IMAGE_H+1) && (V_rel2 >= 1) && (V_rel2 < `IMAGE_V+1) && circle2_show == 1) begin
        RGB_nxt = color_circle1[pixel_data_circle1[V_rel1][H_rel1]];
        object_pixel_index_before_sram_nxt = object_pixel_index_before_sram + 1;
    end
    // Render circle3 image
    else if ((H_rel8 >= 1) && (H_rel8 < `IMAGE_H+1) && (V_rel8 >= 1) && (V_rel8 < `IMAGE_V+1) && circle3_show == 1) begin
        RGB_nxt = color_circle1[pixel_data_circle1[V_rel1][H_rel1]];
        object_pixel_index_before_sram_nxt = object_pixel_index_before_sram + 1;
    end
    // Render circle4 image
    // Render score panel
    else if ((H_rel3 >= 1) && (H_rel3 < `IMAGE_H_SCORE+1) && (V_rel3 >= 1) && (V_rel3 < `IMAGE_V_SCORE+1)) begin
        RGB_nxt = color_map_score[pixel_data_score[V_rel3][H_rel3]];
    end
    // Render score number1
    else if ((H_rel4 >= 1) && (H_rel4 < `IMAGE_H+1) && (V_rel4 >= 1) && (V_rel4 < `IMAGE_V+1)) begin
        case (score / 100)
            0:RGB_nxt = color_map0[pixel_data0[V_rel4][H_rel4]];
            1:RGB_nxt = color_map1[pixel_data1[V_rel4][H_rel4]];
            2:RGB_nxt = color_map2[pixel_data2[V_rel4][H_rel4]];
            3:RGB_nxt = color_map3[pixel_data3[V_rel4][H_rel4]];
            4:RGB_nxt = color_map4[pixel_data4[V_rel4][H_rel4]];
            5:RGB_nxt = color_map5[pixel_data5[V_rel4][H_rel4]];
            6:RGB_nxt = color_map6[pixel_data6[V_rel4][H_rel4]];
            7:RGB_nxt = color_map7[pixel_data7[V_rel4][H_rel4]];
            8:RGB_nxt = color_map8[pixel_data8[V_rel4][H_rel4]];
            9:RGB_nxt = color_map9[pixel_data9[V_rel4][H_rel4]];
            default: RGB_nxt = 0;
        endcase
    end
    // Render score number2
    else if ((H_rel5 >= 1) && (H_rel5 < `IMAGE_H+1) && (V_rel5 >= 1) && (V_rel5 < `IMAGE_V+1)) begin
        case ((score / 10) % 10)
            0: RGB_nxt = color_map0[pixel_data0[V_rel5][H_rel5]];
            1: RGB_nxt = color_map1[pixel_data1[V_rel5][H_rel5]];
            2: RGB_nxt = color_map2[pixel_data2[V_rel5][H_rel5]];
            3: RGB_nxt = color_map3[pixel_data3[V_rel5][H_rel5]];
```

Image Layer:

- Decide layer render order
- Score Panel Controll

Parameter:

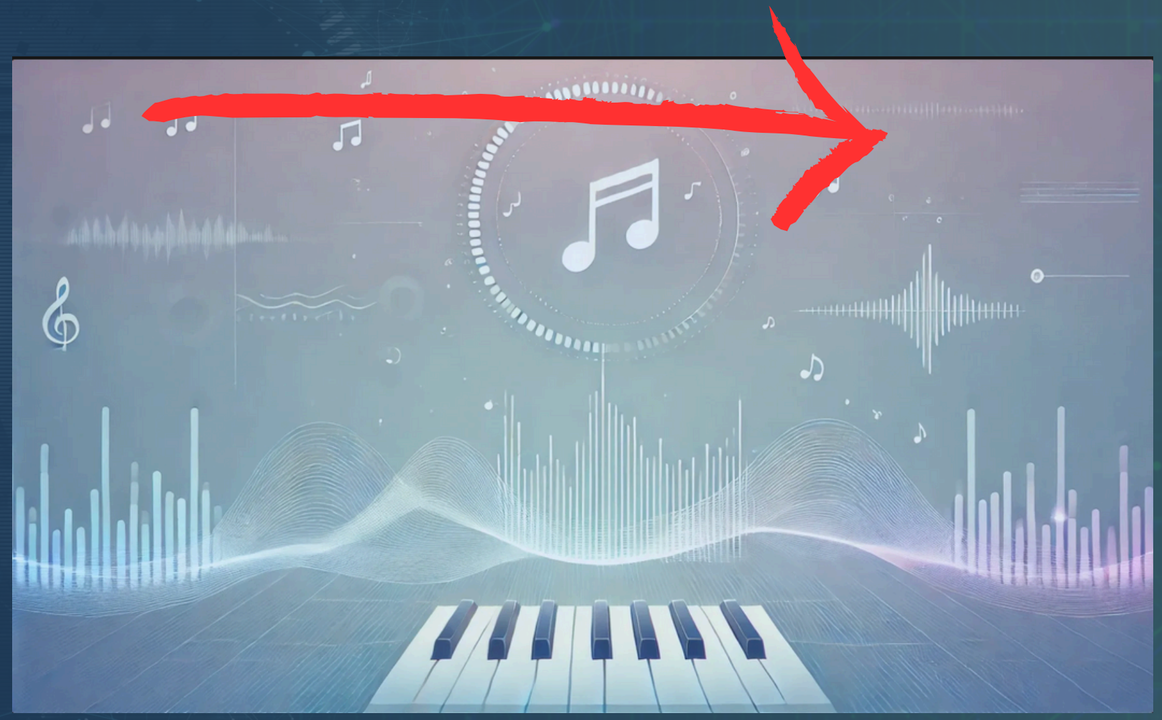
- Resolution = 1600*900
- V_B = 3
- H_B = 80
- V_SIZE = 1
- H_SIZE = 24
- V_D = 96
- H_D = 96

VGA

- **Render Horizontally -> Render Vertically**

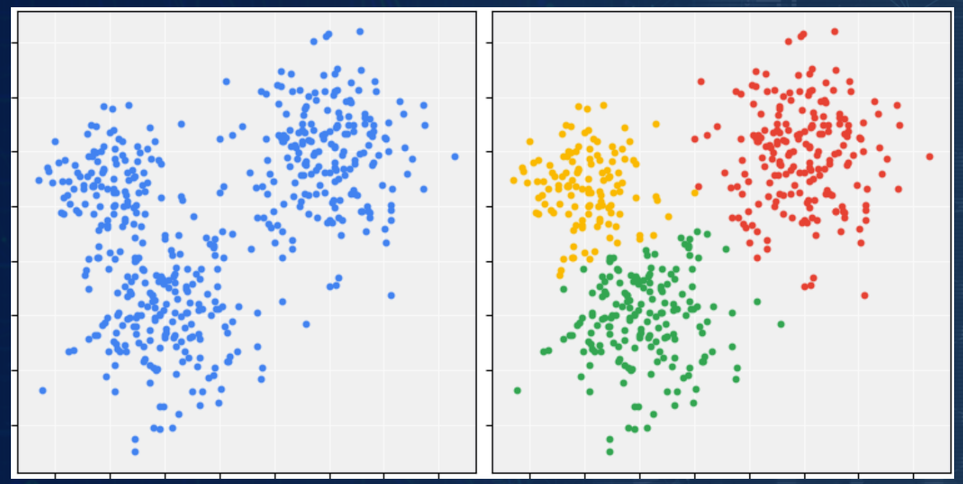
```
H_rel3 = H_counter - score_x;
V_rel3 = V_counter - score_y;
H_rel4 = H_counter - score1_x;
V_rel4 = V_counter - score1_y;
H_rel5 = H_counter - score2_x;
V_rel5 = V_counter - score2_y;
H_rel6 = H_counter - score3_x;
V_rel6 = V_counter - score3_y;
H_rel7 = H_counter - background_x;
V_rel7 = V_counter - background_y;


if (H_STATE == S_C && V_STATE == S_C) begin
    // // Render circle1 image
    if ((H_rel1 >= 1) && (H_rel1 < `IMAGE_H+1) && (V_rel1 >= 1) && (V_rel1 < `I
        RGB_nxt = color_circle1[pixel_data_circle1[V_rel1][H_rel1]];
        object_pixel_index_before_sram_nxt = object_pixel_index_before_sram + 1;
```



VGA - Image Compression

- Image Compress: we use **kmeans** to reduce bits representing colors



Original

Kmeans 16 colors

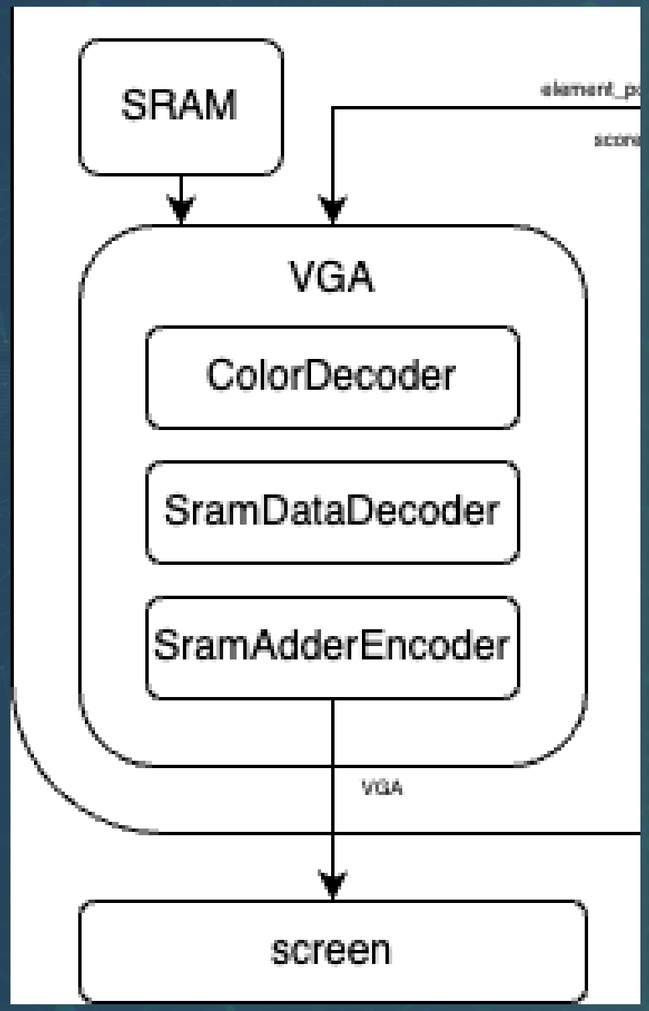
```
module color_palette(output reg [23:0] color_map [0:15]);
    initial begin
        color_map[0] = 24'haaadbb;
        color_map[1] = 24'hdbbce8;
        color_map[2] = 24'hc7b5d5;
        color_map[3] = 24'hd0b8dd;
        color_map[4] = 24'h9faab1;
        color_map[5] = 24'hbab2ca;
        color_map[6] = 24'hc7adee;
        color_map[7] = 24'hdacbf0;
        color_map[8] = 24'hd6bae4;
        color_map[9] = 24'hcbb7da;
        color_map[10] = 24'hd5b6e8;
        color_map[11] = 24'hc1b4d0;
        color_map[12] = 24'hd8bbe6;
        color_map[13] = 24'ha1abb4;
        color_map[14] = 24'hd3b9e1;
        color_map[15] = 24'hb2afc2;
    end
endmodule
```

From 3.75 MB to 0.625 MB => **6x times smaller size**

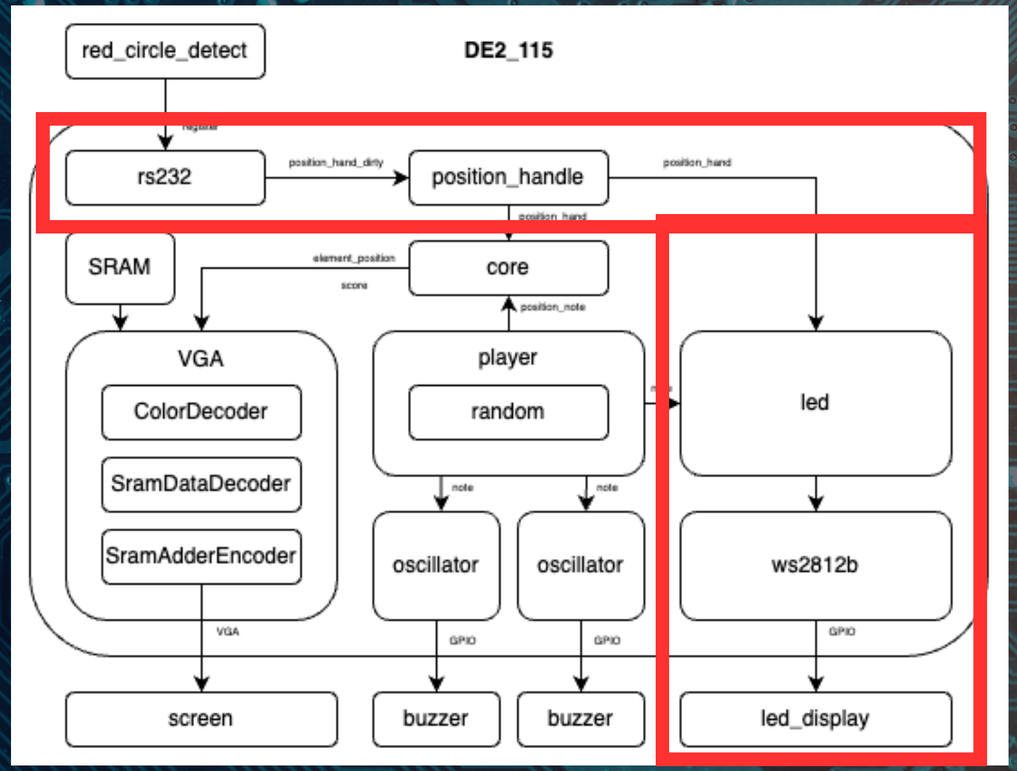
*SRAM storage = 2MB

VGA - SRAM

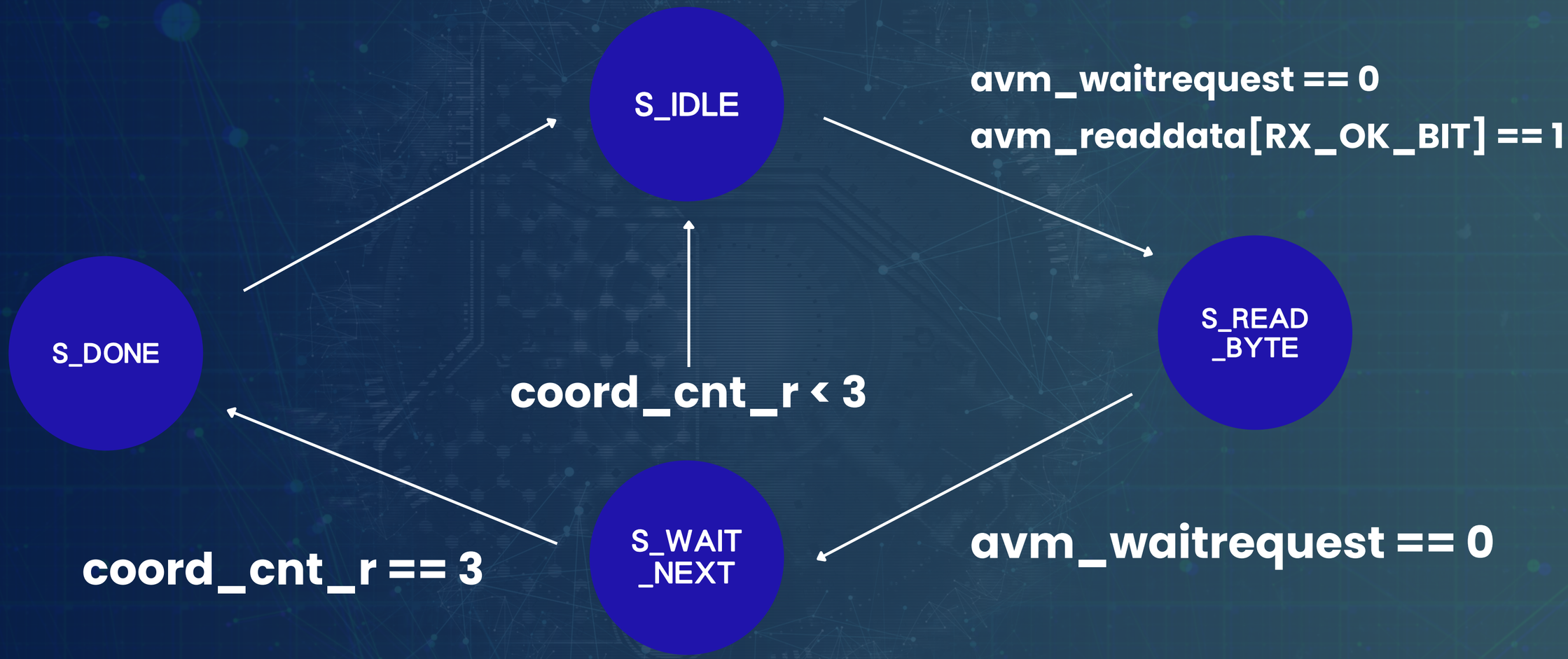
- We save our picture to SRAM and color map, then read and render it.
- Read 4-bits image color information from SRAM
- Decode the 4-bits image color into 24-bits (256,256,256) according to the color map and send to the screen via VGA



RS232 & LED Control

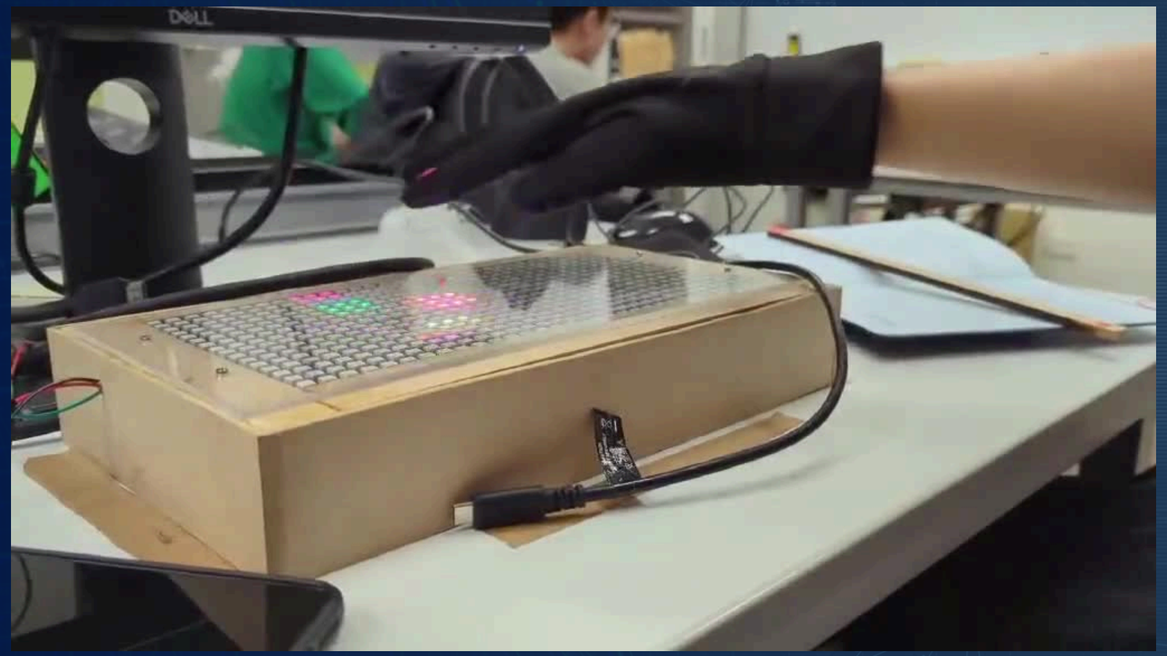


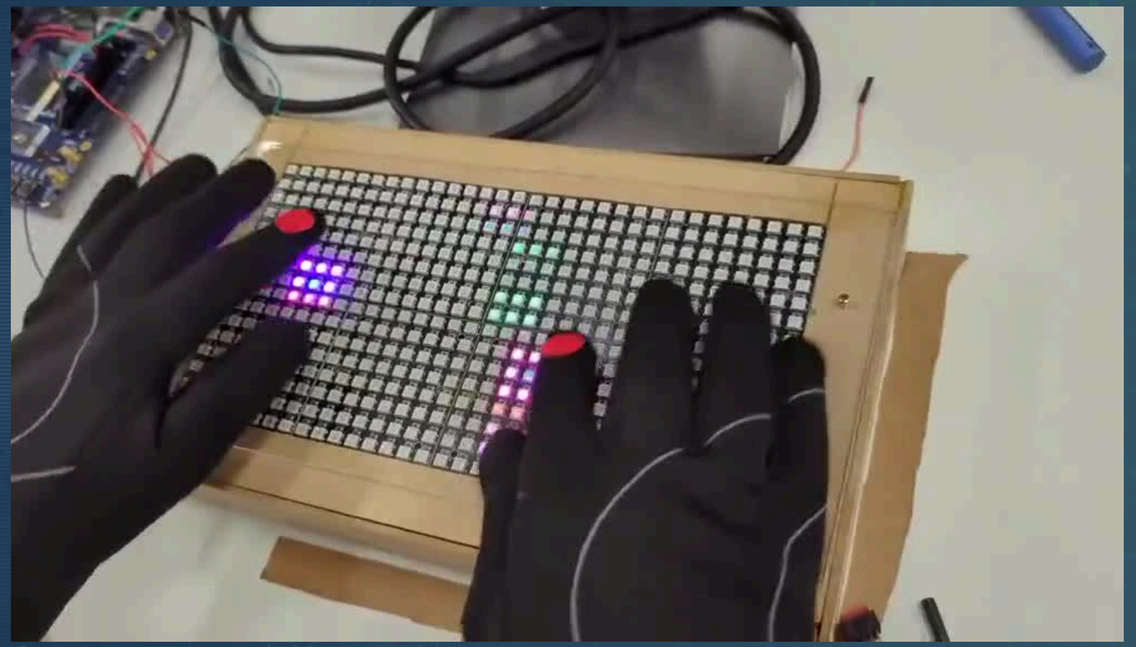
RS232


S_IDLE
avm_waitrequest == 0
avm_readdata[RX_OK_BIT] == 1
S_READ
_BYTE
S_DONE
coord_cnt_r < 3
S_WAIT
_NEXT
coord_cnt_r == 3
avm_waitrequest == 0

LED Light Control

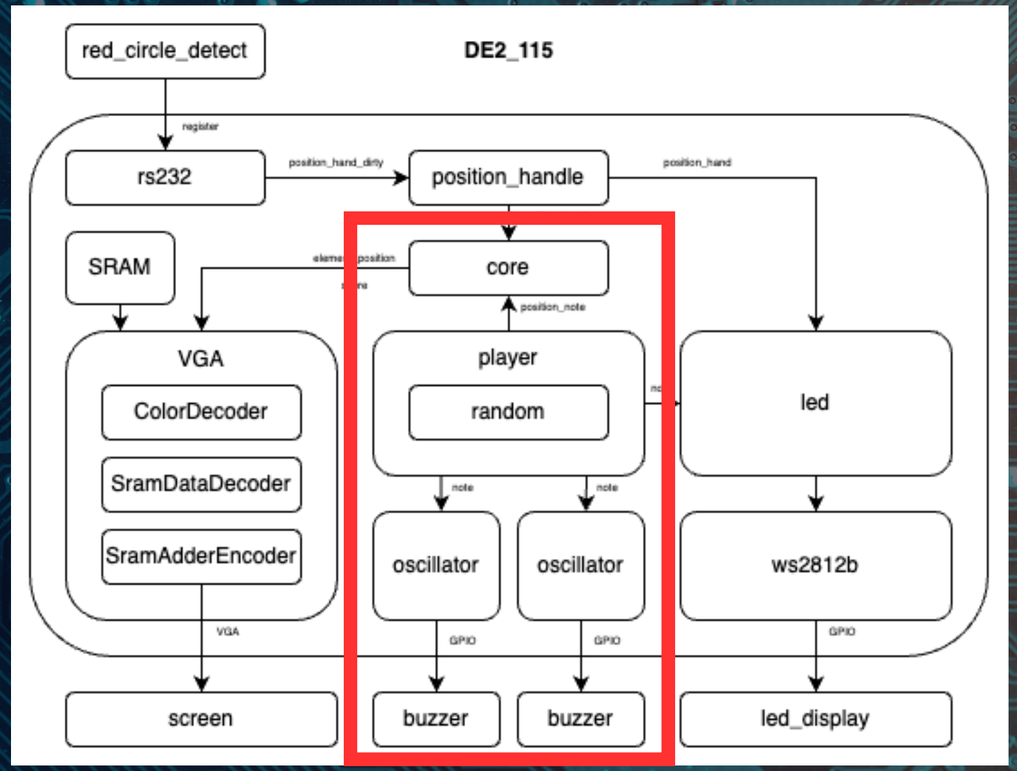
- Detection -> rs232 -> FPGA -> LED display (By GPIO)





- LED give user a “Real-Time Feedback”!

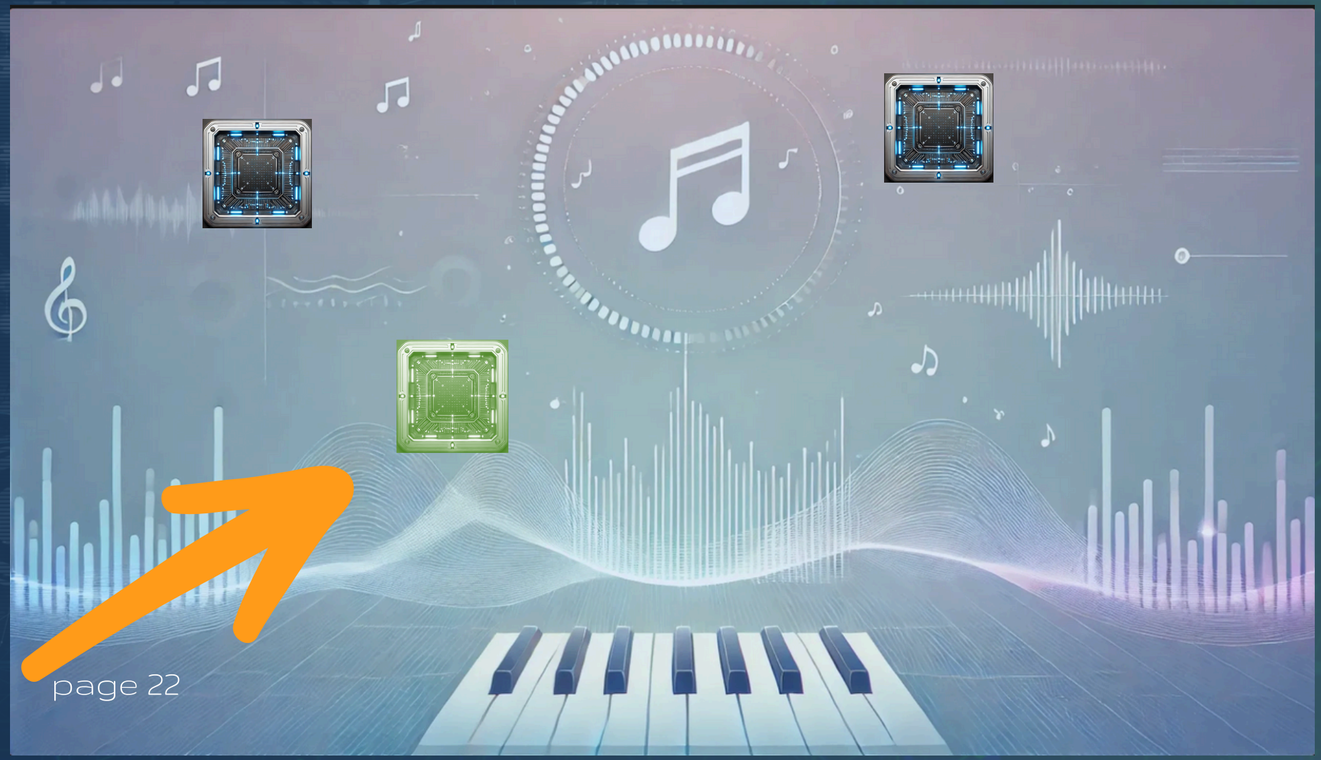
Game Control



Game Core - Music & Symbol Position

- Convert music from midi to verilog, and play with buzzer.
- User can choose different songs and different level(switch[15:17])
- 3 Symbol once, the symbols will appear in sync with the music's rhythm, and the music will play simultaneously.

```
//circle move logic
    always_comb begin
        score_w = score_r;
        circle1_position_w = circle1_position_r;
        circle2_position_w = circle2_position_r;
        circle3_position_w = circle3_position_r;
        order_w = order_r;
        circle1_show_w= circle1_show_r;
        circle2_show_w= circle2_show_r;
        circle3_show_w= circle3_show_r;
        //keep last 3 position
        if(circle1_position_r!=position_note && circle2_position_r!=position_note && circle3_posi
            order_w = order_r + 1;//handle last note
            if(order_w==3)begin
                order_w = 0;
            end
            case(order_r)//last note change to new note
                0:begin
                    circle1_position_w = position_note;
                    circle2_position_w = circle2_position_r;
                    circle3_position_w = circle3_position_r;
                    circle1_show_w= 1;
```



If you hit it, it will became green and disapear after serveral cycles

Game Core - Score Panel

- **Hit -> Score +5 -> Symbol became green -> LED display -> Disappear**

```
always_comb begin//circle hit logic
    circle1_hit = 0;
    circle2_hit = 0;
    circle3_hit = 0;
    //circle1 hit by right hand
    if(position_hand[15:0]!=MINUS1)begin
        if( (rx-circle1_position_r[15:8] < TOLERANCE ||  circle1_position_r[15:8]-rx < TOLERANCE) &&
            (ry-circle1_position_r[7: 0] < TOLERANCE ||  circle1_position_r[7: 0]-ry < TOLERANCE)   )begin
               circle1_hit = 1;
        end
        //circle2 hit by right hand
        if( (rx-circle2_position_r[15: 8] < TOLERANCE || circle2_position_r[15: 8]-rx < TOLERANCE) &&
            (ry-circle2_position_r[ 7: 0] < TOLERANCE || circle2_position_r[ 7: 0]-ry < TOLERANCE)  )begin
               circle2_hit = 1;
        end
        //circle3 hit by right hand
        if( (rx-circle3_position_r[15: 8] < TOLERANCE && circle3_position_r[15: 8]-rx < TOLERANCE) &&
            (ry-circle3_position_r[ 7: 0] < TOLERANCE && circle3_position_r[ 7: 0]-ry < TOLERANCE)  )begin
               circle3_hit = 1;
        end
    end
    if(position_hand[31:16]!=MINUS1)begin
        //circle1 hit by left hand
        if( (lx-circle1_position_r[15: 8] < TOLERANCE || circle1_position_r[15: 8]-lx < TOLERANCE) &&
            (ly-circle1_position_r[ 7: 0] < TOLERANCE || circle1_position_r[ 7: 0]-ly < TOLERANCE)  )begin
               circle1_hit = 1;
        end
        //circle2 hit by left hand
        if( (lx-circle2_position_r[15: 8] < TOLERANCE || circle2_position_r[15: 8]-lx < TOLERANCE) &&
            (ly-circle2_position_r[ 7: 0] < TOLERANCE || circle2_position_r[ 7: 0]-ly < TOLERANCE)  )begin
               circle2_hit = 1;
        end
        //circle3 hit by left hand
        if( (lx-circle3_position_r[15: 8] < TOLERANCE || circle3_position_r[15: 8]-lx < TOLERANCE) &&
            (ly-circle3_position_r[ 7: 0] < TOLERANCE || circle3_position_r[ 7: 0]-ly < TOLERANCE)  )begin
               circle3_hit = 1;
        end
```

```
                2:begin
                    circle1_position_w = circle1_position_r;
                    circle2_position_w = circle2_position_r;
                    circle3_position_w = position_note;
                    circle3_show_w= 1;
                end
                default:begin
                    circle1_position_w = circle1_position_r;
                    circle2_position_w = circle2_position_r;
                    circle3_position_w = circle3_position_r;
                    circle1_show_w= circle1_show_r;
                    circle2_show_w= circle2_show_r;
                    circle3_show_w= circle3_show_r;
                end
            endcase
        end
        if(circle1_hit && circle1_show_r)begin
            circle1_show_w = 0;
            score_w = score_r + 5;
        end
        if(circle2_hit && circle2_show_r)begin
            circle2_show_w = 0;
            score_w = score_r + 5;
        end
        if(circle3_hit && circle3_show_r)begin
            circle3_show_w= 0;
            score_w = score_r + 5;
        end
    end
```

Our team work

- 李冠儀：VGA, Computer Vision
- 陳英睿：SRAM, RS232
- 洪乾峰：Bluzzer, LED
- All: Hardware, Game System Design, Game Core
- ChatGPT: Background and Symbol Image Generation

THANK YOU!

Reference

1. https://blog.csdn.net/qq_37912811/article/details/121953910
2. https://github.com/joeferner/fpga-vga
3. https://github.com/briansune/FPGA-Camera-MIPI-DVP-Verilog?tab=readme-ov-file
4. https://github.com/pujaltedavid/python-image-compressor
5. Thanks many classmates

Material

1. RGB Light WS2812B: Link ($336 * 2)
2. Camera
3. Acrylic + board ($100)