5.6 Object detection

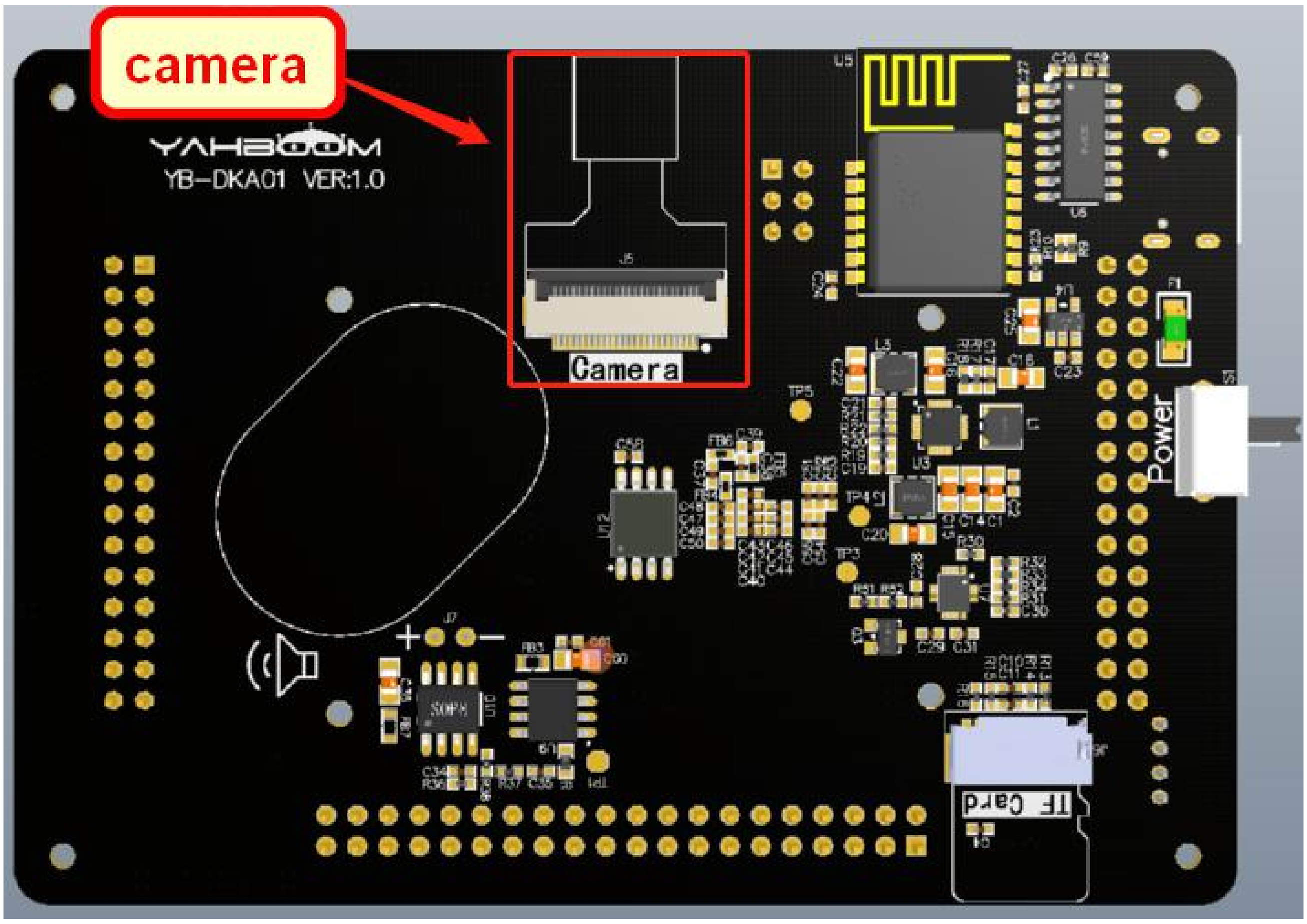
1. Experiment purpose

This lesson mainly learns how to realize object detection by K210.

2.Experiment preparation

2.1 components

OV2640 camera, LCD



2.2 Hardware connection

K210 development board has been installed with the camera and display by default. It only needs type-C data line to connect the K210 development board to the computer.

K210 real-time object detection program steps:

1) Training model
2) Convert the model to kmodol
3) K210 loading model
4) K210 obtains video images
5) The image is converted into the size required by the model
6) Run the model to obtain KPU processing results
7) Get the result of the output layer
8) Circle the recognition result display in the picture

3. Experiment procedure

3.1 Code Procedure

1 - The internal initialization part of the system:

- System clock initialization
- Serial port initialization
- Hardware pin initialization
- IO voltage setting
- System interrupt initialization
- Flash initialization

2 - External hardware initialization

- Lcd initialization
- Ov2640 initialization

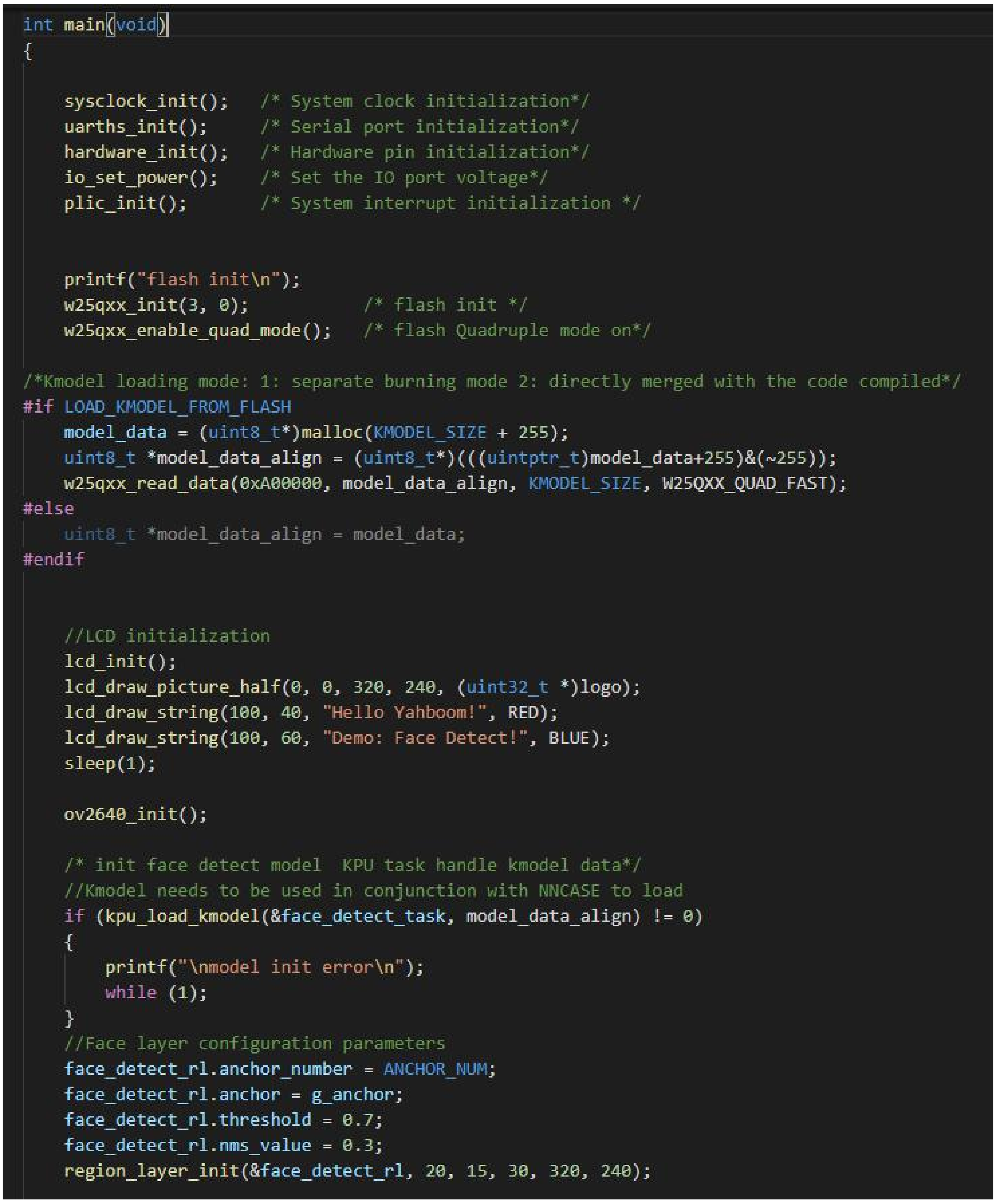
3 - Face detection initialization

- Model loading
- Face detection layer configuration initialization

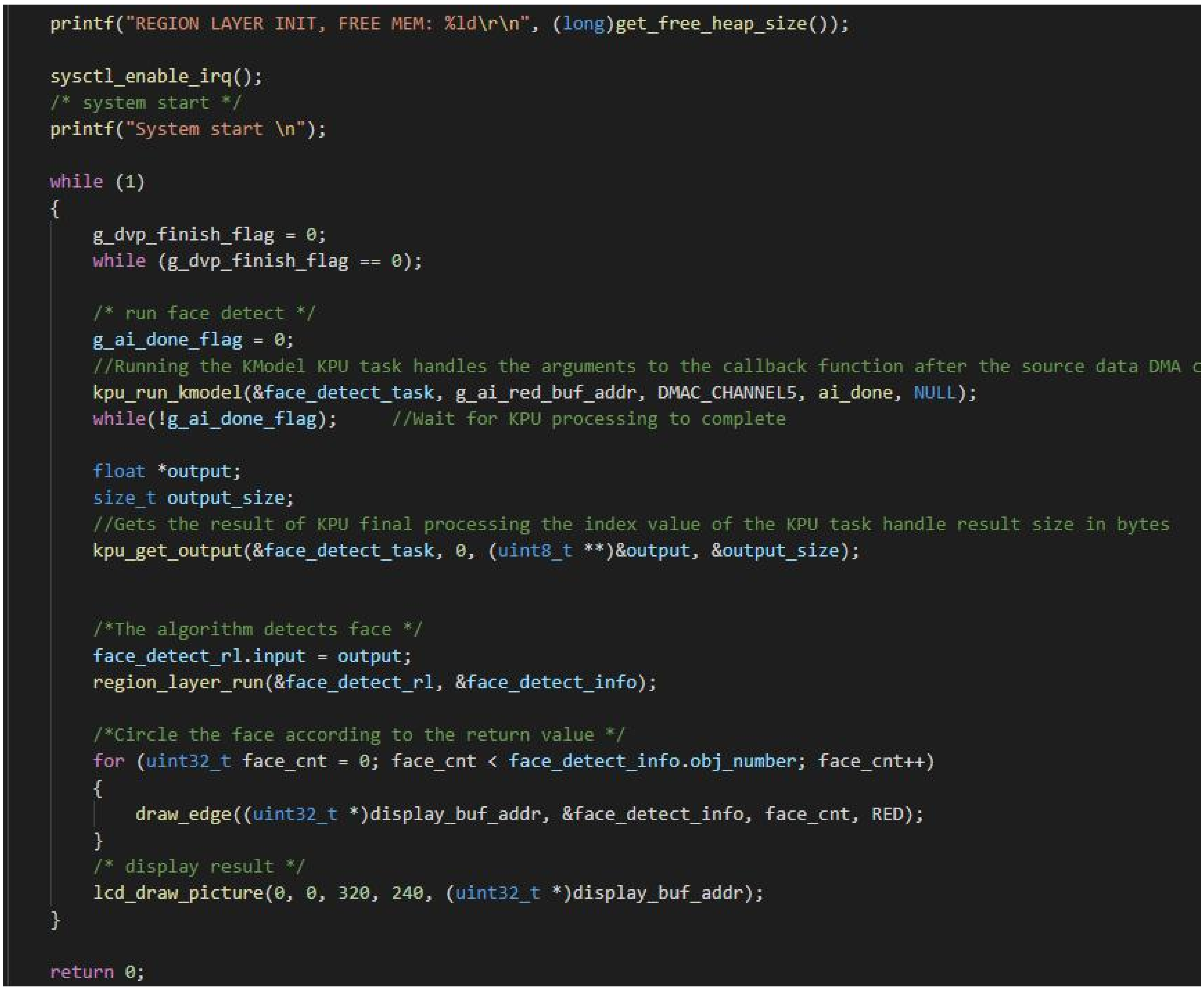
4 - Face detection business logic layer

- Wait for the camera acquisition to complete
- Transfer the images collected by the camera to the KPU running model
- Wait for KPU processing to complete
- Get the final processing result of KPU
- Bring the KPU processed results into the regional layer to calculate the final position
- Mark one by one according to the number of acquired faces

3.2 The core code



```c
int main(void)
{

    sysclock_init();   /* System clock initialization*/
    uarths_init();     /* Serial port initialization*/
    hardware_init();   /* Hardware pin initialization*/
    io_set_power();    /* Set the IO port voltage*/
    plic_init();       /* System interrupt initialization */


    printf("flash init\n");
    w25qxx_init(3, 0);            /* flash init */
    w25qxx_enable_quad_mode();    /* flash Quadruple mode on*/

/*Kmodel loading mode: 1: separate burning mode 2: directly merged with the code compiled*/
#if LOAD_KMODEL_FROM_FLASH
    model_data = (uint8_t*)malloc(KMODEL_SIZE + 255);
    uint8_t *model_data_align = (uint8_t*)(((uintptr_t)model_data+255)&(~255));
    w25qxx_read_data(0xA00000, model_data_align, KMODEL_SIZE, W25QXX_QUAD_FAST);
#else
    uint8_t *model_data_align = model_data;
#endif


    //LCD initialization
    lcd_init();
    lcd_draw_picture_half(0, 0, 320, 240, (uint32_t *)logo);
    lcd_draw_string(100, 40, "Hello Yahboom!", RED);
    lcd_draw_string(100, 60, "Demo: Face Detect!", BLUE);
    sleep(1);

    ov2640_init();

    /* init face detect model  KPU task handle kmodel data*/
    //Kmodel needs to be used in conjunction with NNCASE to load
    if (kpu_load_kmodel(&face_detect_task, model_data_align) != 0)
    {
        printf("\nmodel init error\n");
        while (1);
    }
    //Face layer configuration parameters
    face_detect_rl.anchor_number = ANCHOR_NUM;
    face_detect_rl.anchor = g_anchor;
    face_detect_rl.threshold = 0.7;
    face_detect_rl.nms_value = 0.3;
    region_layer_init(&face_detect_rl, 20, 15, 30, 320, 240);
```



```
    printf("REGION LAYER INIT, FREE MEM: %ld\r\n", (long)get_free_heap_size());

    sysctl_enable_irq();
    /* system start */
    printf("System start \n");

    while (1)
    {
        g_dvp_finish_flag = 0;
        while (g_dvp_finish_flag == 0);

        /* run face detect */
        g_ai_done_flag = 0;
        //Running the KModel KPU task handles the arguments to the callback function after the source data DMA c
        kpu_run_kmodel(&face_detect_task, g_ai_red_buf_addr, DMAC_CHANNEL5, ai_done, NULL);
        while(!g_ai_done_flag);     //Wait for KPU processing to complete

        float *output;
        size_t output_size;
        //Gets the result of KPU final processing the index value of the KPU task handle result size in bytes
        kpu_get_output(&face_detect_task, 0, (uint8_t **)&output, &output_size);


        /*The algorithm detects face */
        face_detect_rl.input = output;
        region_layer_run(&face_detect_rl, &face_detect_info);

        /*Circle the face according to the return value */
        for (uint32_t face_cnt = 0; face_cnt < face_detect_info.obj_number; face_cnt++)
        {
            draw_edge((uint32_t *)display_buf_addr, &face_detect_info, face_cnt, RED);
        }
        /* display result */
        lcd_draw_picture(0, 0, 320, 240, (uint32_t *)display_buf_addr);
    }

    return 0;
```

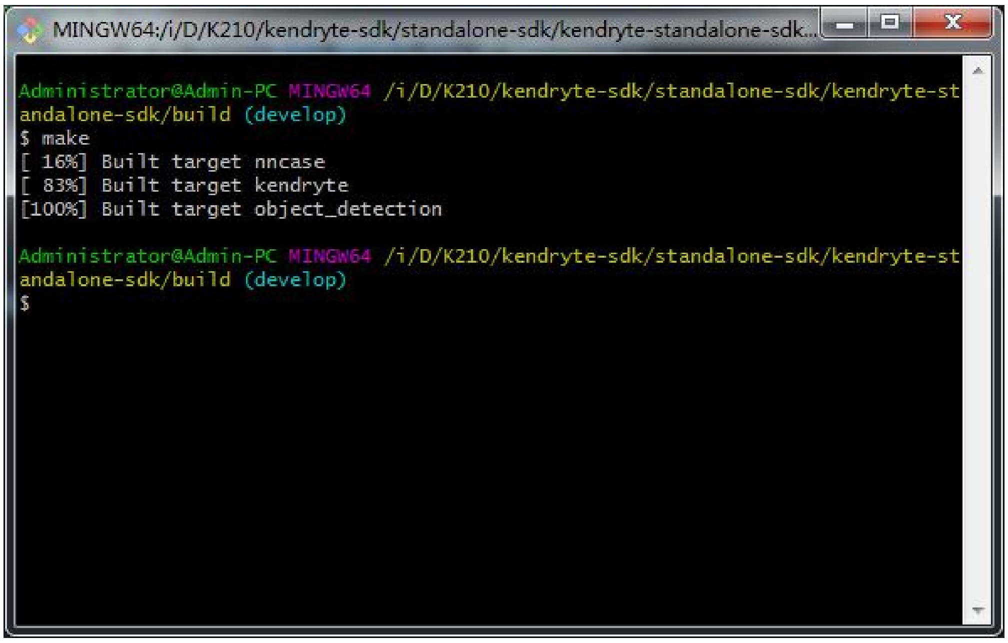
3.3 Compile and debug, burn and run

Copy the object_detection to the src directory in the SDK.

Then, enter the build directory and run the following command to compile.

```
cmake .. -DPROJ=object_detection -G "MinGW Makefiles“
make
```

After the compilation is complete, the **object_detection.bin** file will be generated in the build folder. We need to use the type-C data cable to connect the computer and the K210 development board. Open kflash, select the corresponding device, and then burn the **object_detection.bin** file to the K210 development board.


MINGW64:/i/D/K210/kendryte-sdk/standalone-sdk/kendryte-standalone-sdk...

```
Administrator@Admin-PC MINGW64 /i/D/K210/kendryte-sdk/standalone-sdk/kendryte-st
andalone-sdk/build (develop)
$ make
[ 16%] Built target nncase
[ 83%] Built target kendryte
[100%] Built target object_detection

Administrator@Admin-PC MINGW64 /i/D/K210/kendryte-sdk/standalone-sdk/kendryte-st
andalone-sdk/build (develop)
$
```

4. Experimental phenomenon

LCD will display the picture logo and text. After one second, the camera start collect pictures, 20 kinds of objects will be detected in real time and mark their positions, display the recognition results.

5. Experiment summary

5.1 Object detection and face detection are mostly the same.

5.2 Object detection can detect a variety of objects.