ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KỸ THUẬT MÁY TÍNH

BÁO CÁO MINI PROJECT:

SINGLE PING-PONG GAME

MÔN THIẾT KẾ HỆ THỐNG NHÚNG

LỚP:CE224.O12.2

GIẢNG VIÊN HƯỚNG DẪN:

CHUNG QUANG KHÁNH

TP. HỒ CHÍ MINH – Tháng 1 năm 2024

SINGLE PING-PONG GAME

Nhóm

STT	Tên thành viên	MSSV	Đánh giá
1	Trần Lê Minh Đăng	21520684	
2	Lê Hữu Đạt	21520697	
3	Trần Văn Dương	21520763	

- Link code:

https://github.com/DangUIT/CE224_Project.git

- Link video demo:

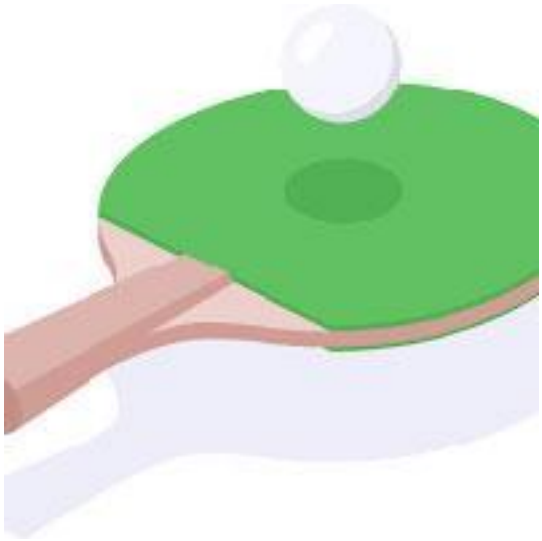
https://drive.google.com/file/d/1wudimEJGNgzywjSTfItgjg17MlvWHRJ-/view?usp=sharing

SINGLE PING-PONG GAME

I. Giới thiệu về Single Ping-Pong Game.

1. Mô tả và yêu cầu

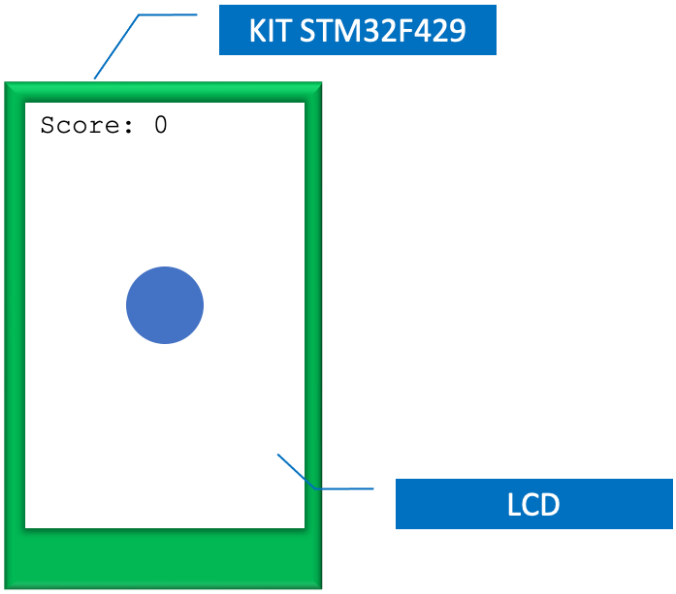
Single Ping-Pong Game là một trò chơi mô phỏng việc dùng vợt và tâng bóng bàn được mô tả như trong Hình 1, trong đó, KIT STM32F429 được sử dụng như cây vợt, màn hình LCD sẽ hiển thị một hình tròn mô phỏng trái bóng bàn.



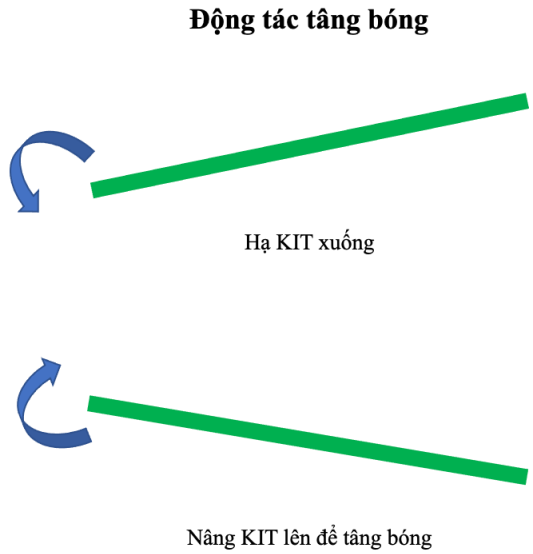
Hình 1. Mô tả game tâng bóng bàn

Nguyên lý của game như sau:

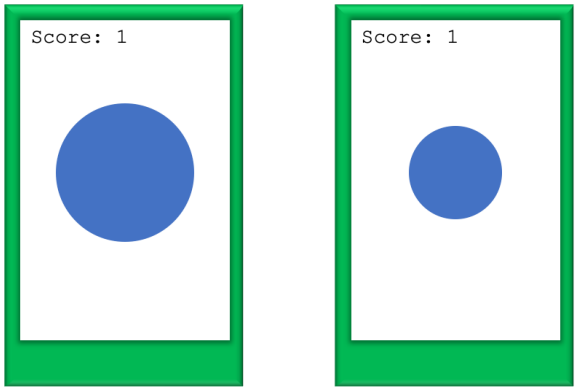
- Đầu tiên, hệ thống ở trạng thái cân bằng khởi đầu.
- Người chơi sẽ hạ và nâng KIT để thực hiện động tác tâng bóng, khi đó trên màn hình LCD sẽ mô phỏng trạng thái trái bóng được nâng lên và rơi xuống.
- Mục tiêu của người chơi là phải nâng KIT đúng lúc trái bóng rơi xuống, đập vào "mặt vợt" và tâng lên lại, người chơi được tính điểm, đèn xanh chớp 1 lần.
- Nếu người chơi nâng đúng, tùy thuộc vào mức độ chính xác mà bóng sẽ nảy lên với tốc độ khác nhau.
- Nếu người tâng "hụt" thì bóng sẽ rớt và GAME OVER, đèn đỏ được bật và giữ cho tới khi reset.



Hình 2. Trạng thái bắt đầu của game



Hình 3. Mô phỏng động tác tâng bóng



Mô phỏng bóng bay lên Mô phỏng bóng rớt xuống

Hình 4. Mô phỏng bóng được nâng và rớt

Yêu cầu về mức độ hoàn thành:

Yêu cầu	Điểm
Hiển thị được giao diện ban đầu	2 điểm
Hiện thực được thao tác tâng bóng tại chỗ (điểm giữa màn hình)	3 điểm
Xuất thông số độ cao khi bóng được tâng ra máy tính qua Virtual Com Port	2 điểm
Hiện thực được chức năng tính điểm	3 điểm
Hiện thực được việc bóng bay theo 1 chiều dựa trên phương và chiều của lúc nâng bóng (nếu bóng bay ra rìa LCD sẽ đập ngược trở lại)	1 điểm
Hiện thực được việc bóng bay theo 2 chiều dựa trên phương và chiều của lúc nâng bóng (nếu bóng bay ra rìa LCD sẽ đập ngược trở lại)	1 điểm

2. **Các ứng dụng sử dụng**

- **FreeRTOS:** là hệ điều hành thời gian thực cho phép ứng dụng thực thi đa tác vụ và đáp ứng theo thời gian thực. Với nội dung đồ án sử dụng FreeRTOS trên KIT STM32F429 để lập lịch cho các tác vụ được mô tả ở Bảng 1.
- **Cảm biến Gyroscope:** là cảm biến con quay hồi chuyển dung để đo đạc hoặc duy trì phương hướng. Cảm biến sử dụng để đo góc quay vật thể theo 3 trục x, y, z sử dụng giao tiếp I2C hoặc SPI. Với nội dung đồ án sử dụng cảm biến để đọc góc nghiên của cảm biến theo trục x, y từ đó xác định góc tâng và độ cao bóng. Việc đọc giá trị cảm biến sử dụng thư viện hỗ trợ (Board Support Package - BSP) và giao tiếp với giao thức SPI.
- **TFT LCD:** sử dụng LCD với mục đích hiển thị giao diện, tính điểm và tương tác trò chơi. Việc điều khiển LCD phụ thuộc đọc và xử lí dữ liệu và từ Gyroscope. Việc điều kiển LCD sử dụng thư viện hỗ trợ (Board Support Package - BSP).
- **Virtual Com Port:** Xuất độ cao đã được tính toán hiển thị qua máy tính bằng cổng USB. Sử dụng USB OTG HS với thư viện hỗ trợ usbd_cdc_if.h.
- **GPIO:** sử dụng các led on-board và nút reset để hiển thị trạng thái khi tâng bóng, hạ bóng, game over và khởi tạo trạng thái ban đầu cho game.

II. Thực hiện.

1. Thiết kế mô hình phần cứng cần thiết

- Cảm biến con quay hồi chuyển Gyroscope: giao thức giao tiếp sử dụng là SPI (SPI5 trên KIT STM32F429)
- LED 3, LED 4 giao tiếp GPIO (PG13, PG14 trên KIT STM32F429)
- Truyền thông nối tiếp qua Virtual Com Port (USB OTG HS) với thư viện hỗ trợ BSP
- Button reset có sẵn trên KIT để reset trạng thái ban đầu.
- Màn LED TFT trên KIT để hiển thị giao diện với thư viện hỗ trợ BSP
- Dùng FreeRTOS hỗ trợ trên KIT để lập lịch các tiến trình

2. Sử dụng RTOS, thiết mô hình phần mềm

Task	Chức năng
1	Đọc dữ liệu từ cảm biến gyroscope và chuyển đến Queue để chia sẻ với các tasks khác.
2	Xử lý dữ liệu và cập nhật trạng thái trò chơi
3	Gửi dữ liệu ra cổng USB CDC
4	Hiển thị điểm số trên LCD và kiểm tra kết thúc trò chơi

Bảng 1. Các task chức năng

a. Task 1

Mục tiêu: Đọc dữ liệu từ cảm biến gyroscope và chuyển đến Queue để chia sẻ với các tasks khác.

- Luồng xử lý:
- Sử dụng thư viện BSP_GYRO_Init để khởi tạo cảm biến gyroscope.
- Trong vòng lặp vô hạn (for(;;)):
  - Đọc giá trị gyroscope thông qua hàm L3GD20_ReadXYZAngRate.
  - Đưa dữ liệu vào một cấu trúc (SendX.Buf) và gửi nó qua Queue01Handle để chia sẻ với các tasks khác.
  - Kiểm tra điều kiện (abs(SendX.Buf[0]) > 20) để xác định khi nào bóng được đánh.

```
299 void StartTask01(void *argument)
300 {
301   /* init code for USB_DEVICE */
302   MX_USB_DEVICE_Init();
303   /* USER CODE BEGIN 5 */
304       Mymessage SendX;
305       /* Infinite loop */
306       for(;;)
307       {
308         if(flag == 0)
309         {
310         L3GD20_ReadXYZAngRate(gyroValue);
311         SendX.Buf[0] = (int)gyroValue[0]/1500;
312         SendX.Buf[1] = (int)gyroValue[1];
313         SendX.Buf[2] = (int)gyroValue[2];
314         if (abs(SendX.Buf[0]) > 20)
315         {
316             hitflag = 1;
317             osMessageQueuePut(Queue01Handle, &SendX, 0, 0);
318         }
319         else
320         {
321             hitflag = 0;
322         }
323         osDelay(250);
324         }
325       }
```

b. Task 2

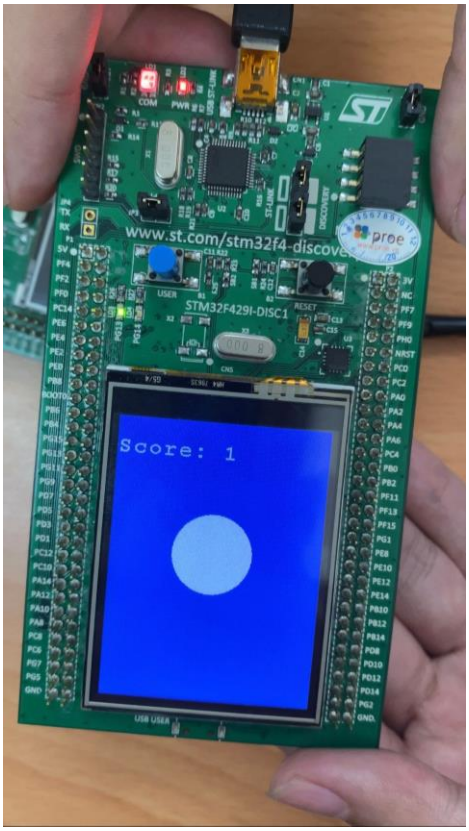
Mục tiêu: Xử lý dữ liệu từ Queue và cập nhật trạng thái của trò chơi, bao gồm việc vẽ hình tròn lên LCD và kiểm tra kết quả.

Luồng xử lý:

- Sử dụng thư viện BSP_LCD_FillCircle để vẽ hình tròn lên LCD.
- Trong vòng lặp vô hạn (for(;;)):
  - Kiểm tra hitflag để xác định khi nào bóng được đánh.
  - Nếu bóng được đánh, lấy dữ liệu từ Queue01Handle và thực hiện các tính toán để cập nhật trạng thái trò chơi.
  - Kiểm tra điều kiện kết thúc trò chơi (flag == 1) và thực hiện các tác vụ cần thiết khi trò chơi kết thúc.

```
void StartTask02(void *argument)
{
  /* USER CODE BEGIN StartTask02 */
        int temp;
        BSP_LCD_FillCircle(X_circle, Y_circle, radius);
        Mymessage GetX;
        for(;;)
        {
            if (hitflag == 1 && flag == 0)
            {
                osMessageQueueGet(Queue01Handle, &GetX, 0, 0);
                temp = GetX.Buf[0];
                for (int i = 0; i <= temp; i = i + 10)
                {
                    BSP_LCD_Clear(LCD_COLOR_BLUE);
                    BSP_LCD_FillCircle(X_circle, Y_circle, radius);
                    radius = radius + 5;
                    if(radius==50){
                        realScore = realScore + 1;
                        HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13, GPIO_PIN_SET);
                        osDelay(250);
                        HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13, GPIO_PIN_RESET);
                    }
                    if (radius > 90){
                        break;
                    }
                    osDelay(100);
                }
                if(temp<=140)//Set lại giá trị temp khi tăng bóng không đủ lực
                    temp = 200;
```

```
                for (int i = temp; i >= 0; i = i - 10)
                {
                    BSP_LCD_Clear(LCD_COLOR_BLUE);
                    BSP_LCD_FillCircle(X_circle, Y_circle, radius);
                    radius = radius - 5;
                    if (radius >= 20 && radius <= 50)
                    {
                        if(hitflag==1)
                            break;
                    }
                    if (radius < 20)
                    {
                        flag = 1;
                        break;
                    }
                    osDelay(100);
                }
            }
        }
  /* USER CODE END StartTask02 */
}
```



Hình 5 Hiển thị đèn xanh khi bóng được tâng lên

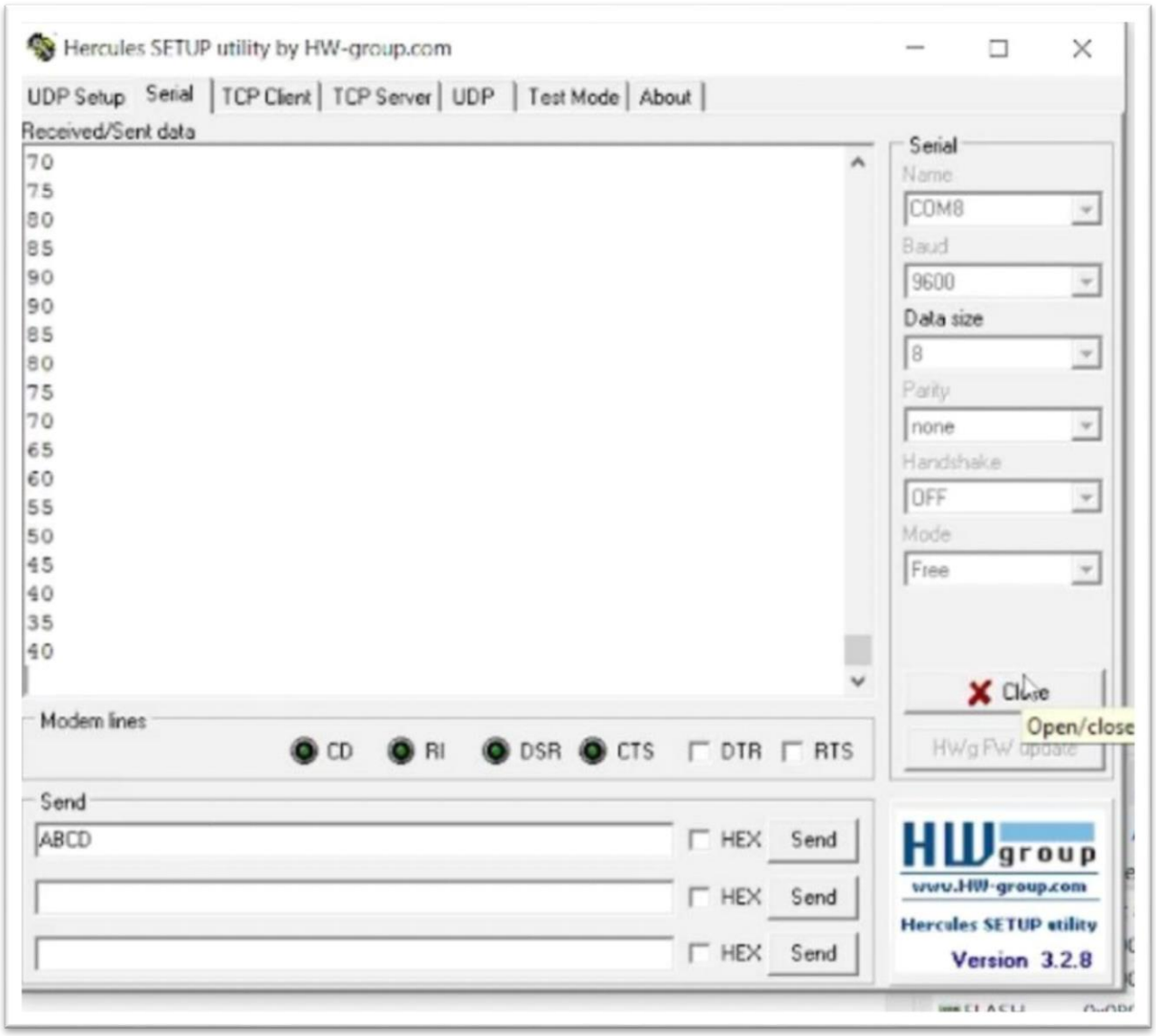
c. Task 3

Mục tiêu: Gửi dữ liệu về giá trị bán kính của hình tròn qua cổng USB CDC.

Luồng xử lý:

- Trong vòng lặp vô hạn (for(;;)):
  - Sử dụng hàm CDC_Transmit_HS để gửi dữ liệu bán kính qua cổng USB CDC.
  - Thực hiện đợi (osDelay) để mỗi 100 ms gửi 1 lần

```
void StartTask03(void *argument)
{
  /* USER CODE BEGIN StartTask03 */
        for(;;)
        {
            sprintf(buf2, "\n%d", radius);
            CDC_Transmit_HS(buf2,sizeof(buf2));
            osDelay(100);
        }
  /* USER CODE END StartTask03 */
}
```



Hình 6 Hiển thị qua cổng USB

d. Task 4

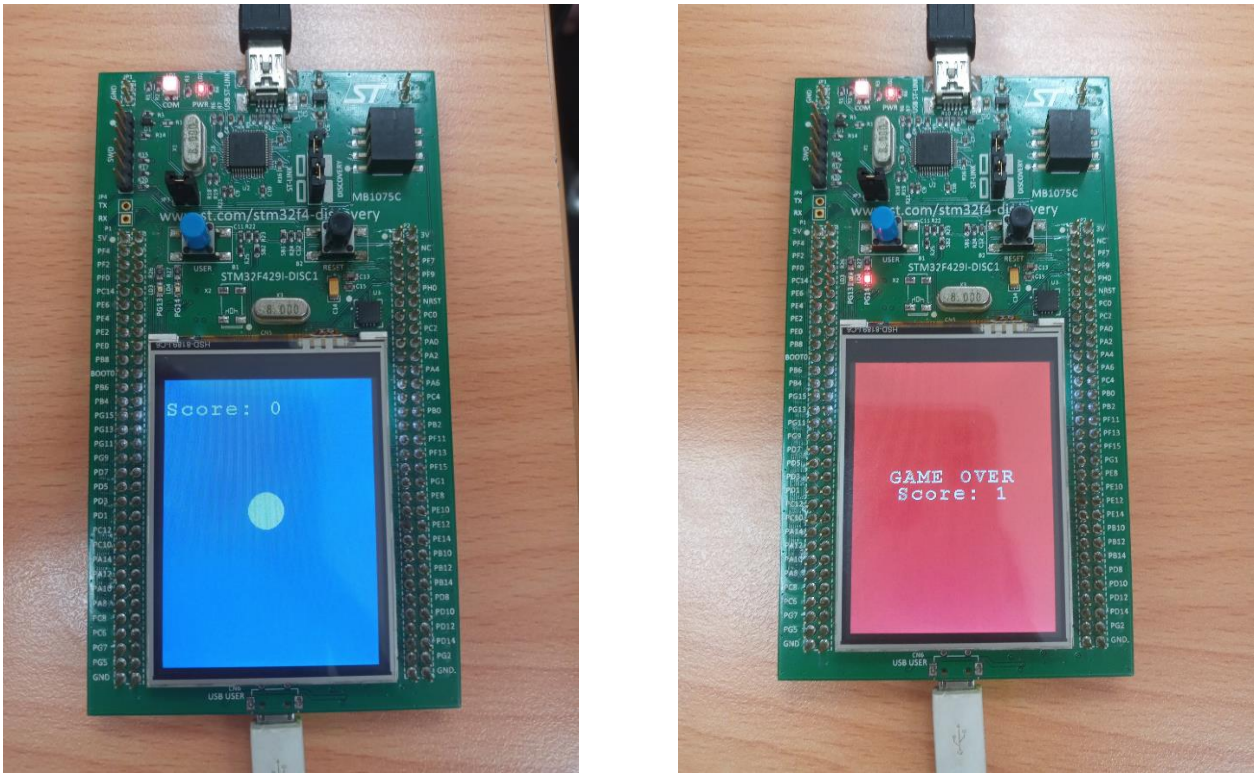
Mục tiêu: Hiển thị điểm số trên LCD và kiểm tra điều kiện kết thúc trò chơi.

Luồng xử lý:

- Trong vòng lặp vô hạn (for(;;)):
  - Hiển thị điểm số trên màn hình LCD khi trò chơi đang diễn ra.

- Nếu trò chơi kết thúc, hiển thị thông báo "Game Over!" và điểm số cuối cùng, sau đó dừng các task.

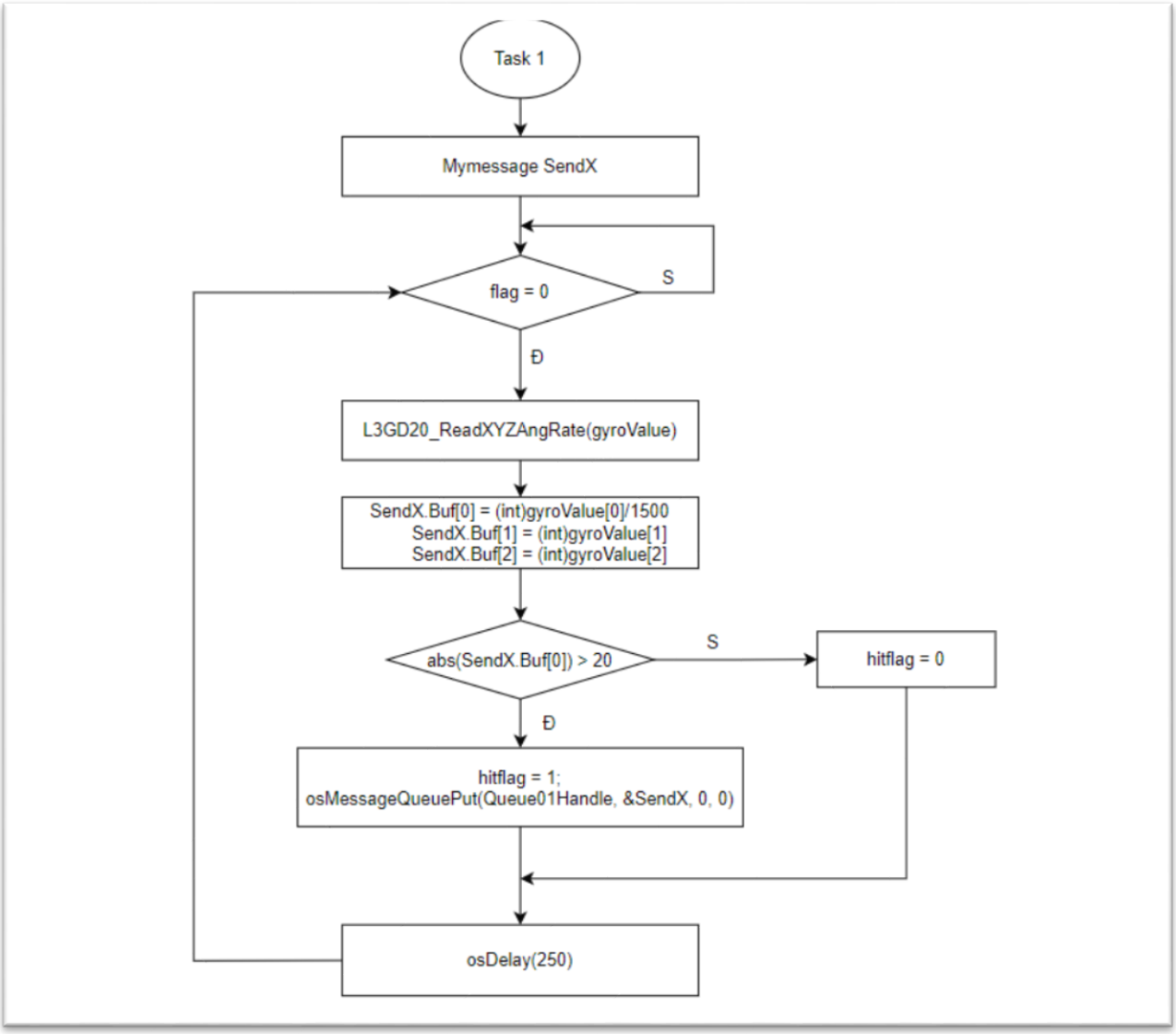
```
void StartTask04(void *argument)
{
  /* USER CODE BEGIN StartTask04 */
  /* Infinite loop */
  for(;;)
  {
        if (flag == 0)
        {
            sprintf((char*)buf1, "Score: %d", realScore);
            BSP_LCD_DisplayStringAtLine(1, buf1);
        }
        else if (flag == 1)
        {
            BSP_LCD_Clear(LCD_COLOR_RED);
            BSP_LCD_SetBackColor(LCD_COLOR_RED);
            BSP_LCD_DisplayStringAt(1,130,(uint8_t*)"GAME OVER", CENTER_MODE);
            BSP_LCD_DisplayStringAt(1,150,(uint8_t*)buf1,CENTER_MODE);
            HAL_GPIO_WritePin(GPIOG, GPIO_PIN_14, GPIO_PIN_SET);
            vTaskDelete(Task01Handle);
            vTaskDelete(Task02Handle);
            vTaskDelete(Task03Handle);
        }
  }
  /* USER CODE END StartTask04 */
}
```



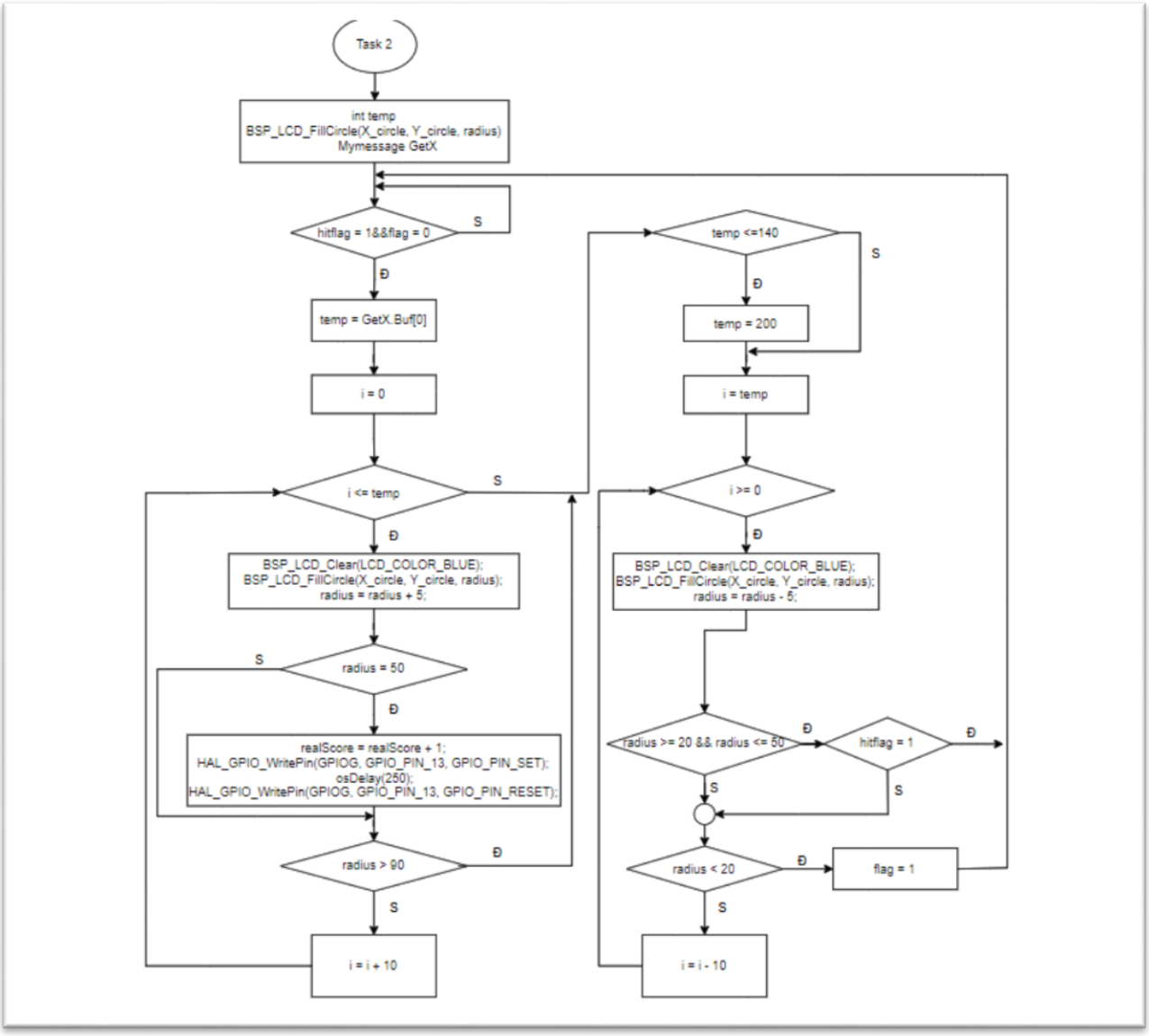
Hình 7 Giao diện bắt đầu và kết thúc của trò chơi

3. Thiết kế lưu đồ giải thuật

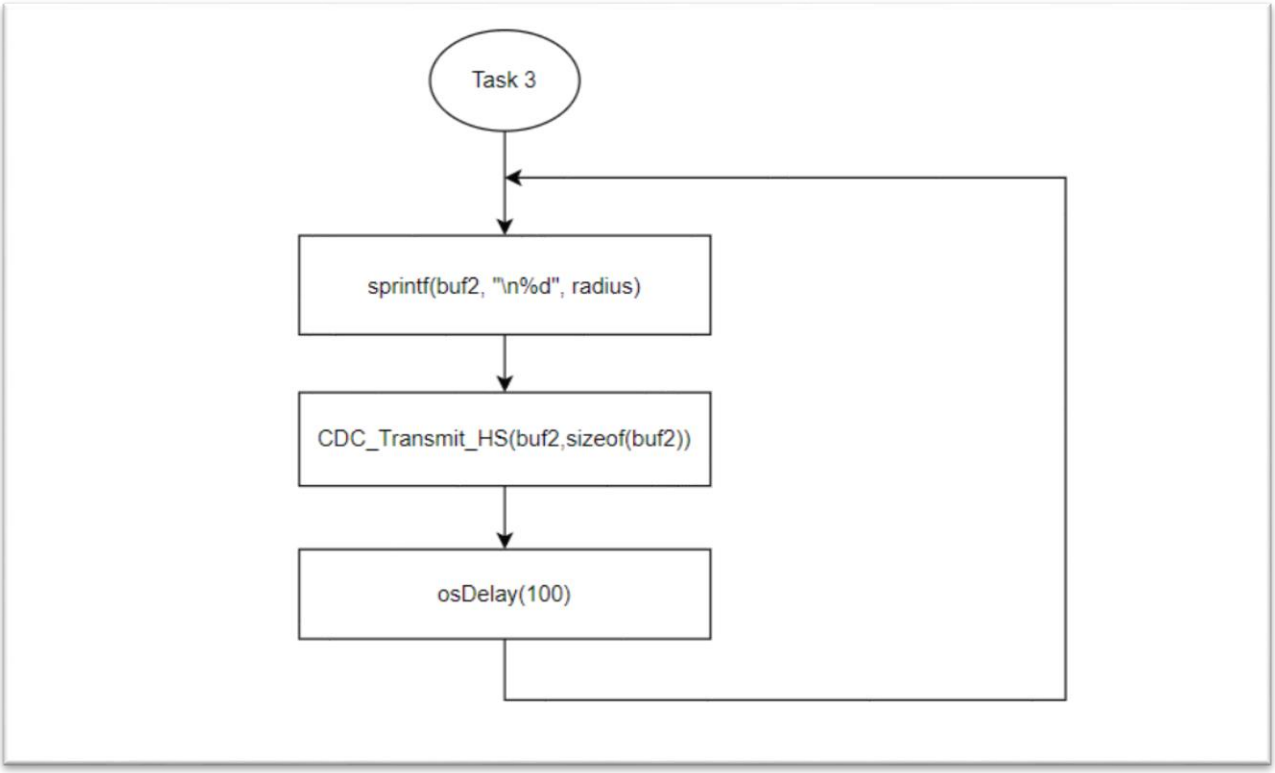
a. Task 1


Task 1
Mymessage SendX
flag = 0
S
Đ
L3GD20_ReadXYZAngRate(gyroValue)
SendX.Buf[0] = (int)gyroValue[0]/1500
SendX.Buf[1] = (int)gyroValue[1]
SendX.Buf[2] = (int)gyroValue[2]
abs(SendX.Buf[0]) > 20
S
hitflag = 0
Đ
hitflag = 1;
osMessageQueuePut(Queue01Handle, &SendX, 0, 0)
osDelay(250)

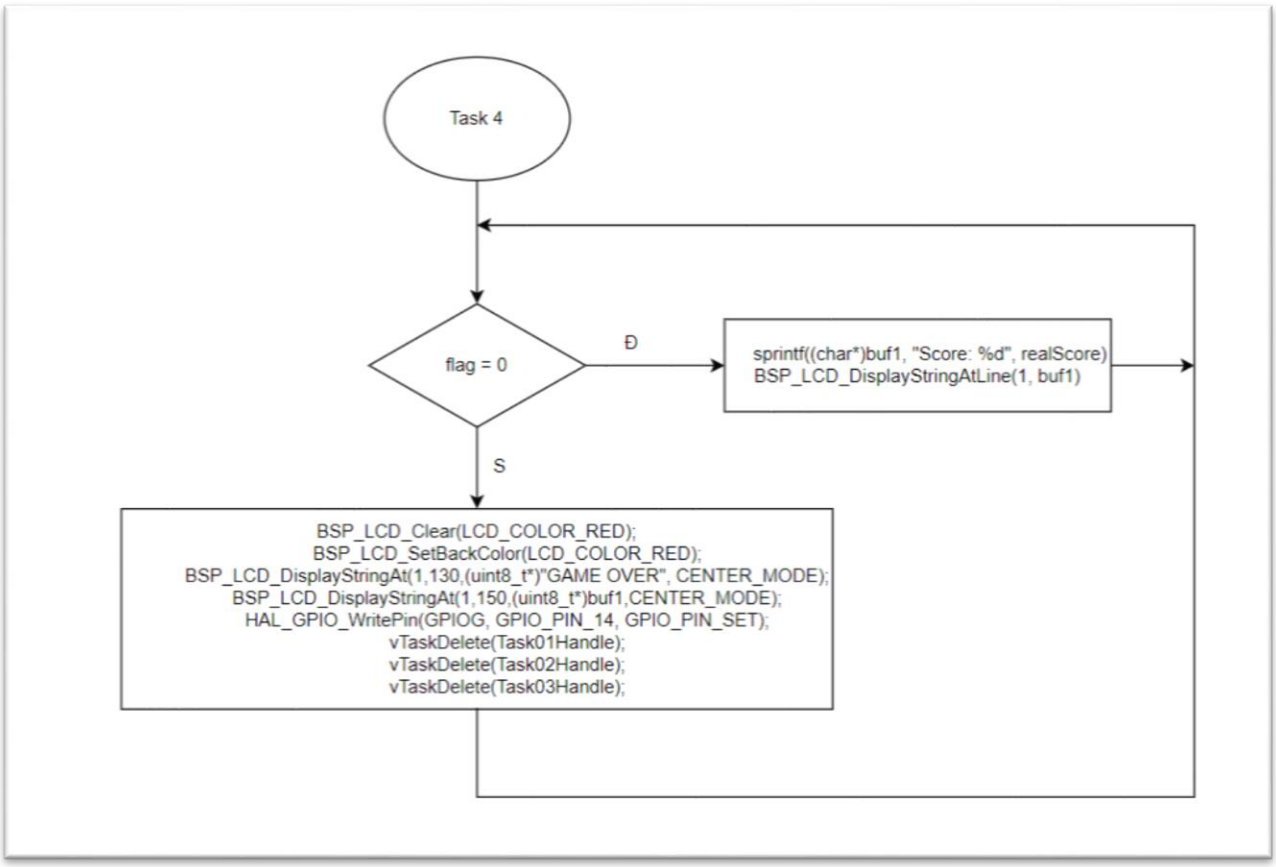
b. Task 2


Task 2
int temp
BSP_LCD_FillCircle(X_circle, Y_circle, radius)
Mymessage GetX
hitflag = 1&&flag = 0
S
Đ
temp = GetX.Buf[0]
i = 0
i <= temp
S
Đ
BSP_LCD_Clear(LCD_COLOR_BLUE);
BSP_LCD_FillCircle(X_circle, Y_circle, radius);
radius = radius + 5;
radius = 50
S
Đ
realScore = realScore + 1;
HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13, GPIO_PIN_SET);
osDelay(250);
HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13, GPIO_PIN_RESET);
radius > 90
Đ
S
i = i + 10
temp <=140
S
Đ
temp = 200
i = temp
i >= 0
Đ
BSP_LCD_Clear(LCD_COLOR_BLUE);
BSP_LCD_FillCircle(X_circle, Y_circle, radius);
radius = radius - 5;
radius >= 20 && radius <= 50
Đ
hitflag = 1
Đ
S
S
radius < 20
Đ
flag = 1
S
i = i - 10

c. Task 3

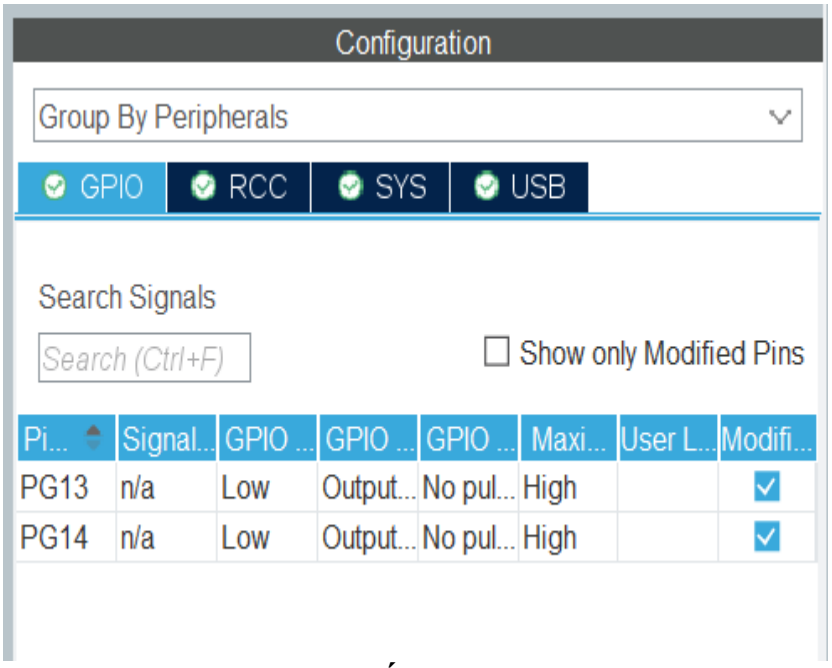

Task 3
sprintf(buf2, "\n%d", radius)
CDC_Transmit_HS(buf2,sizeof(buf2))
osDelay(100)

d. Task 4

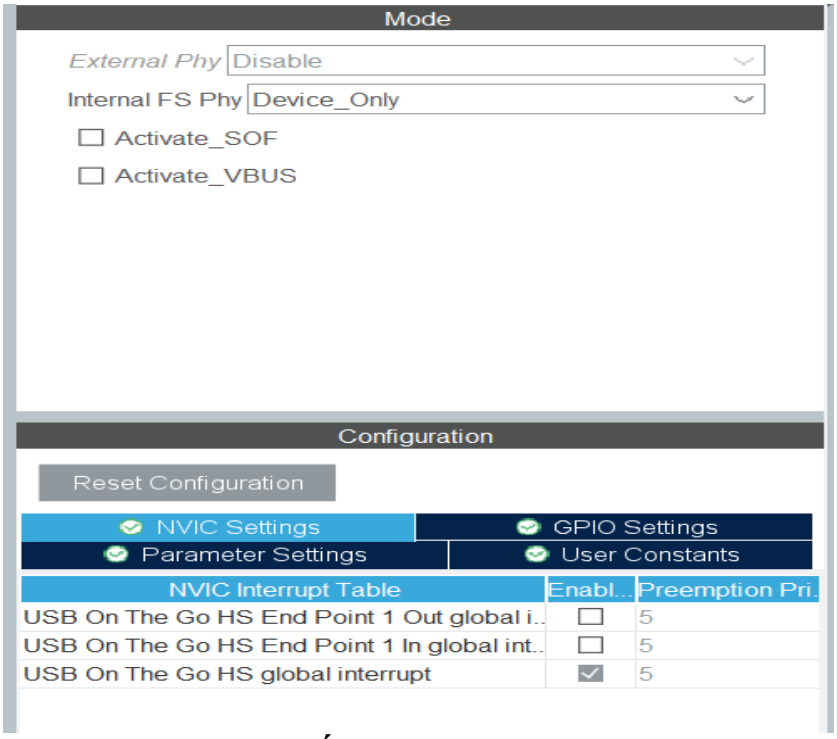


4. Hiện thực hệ thống và báo cáo

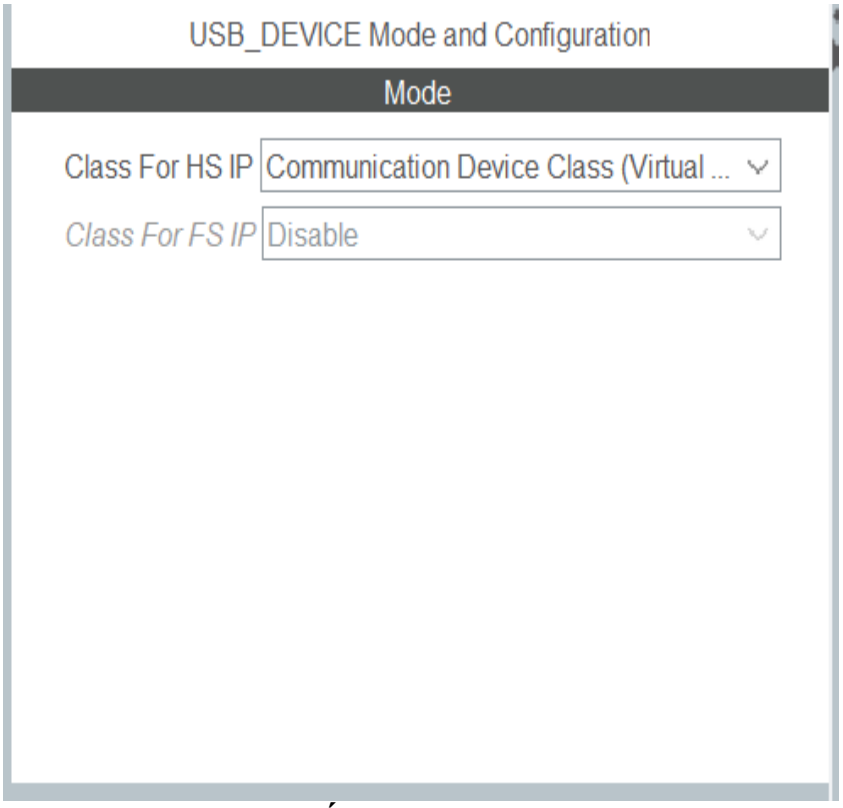
a) Thực hiện cấu hình



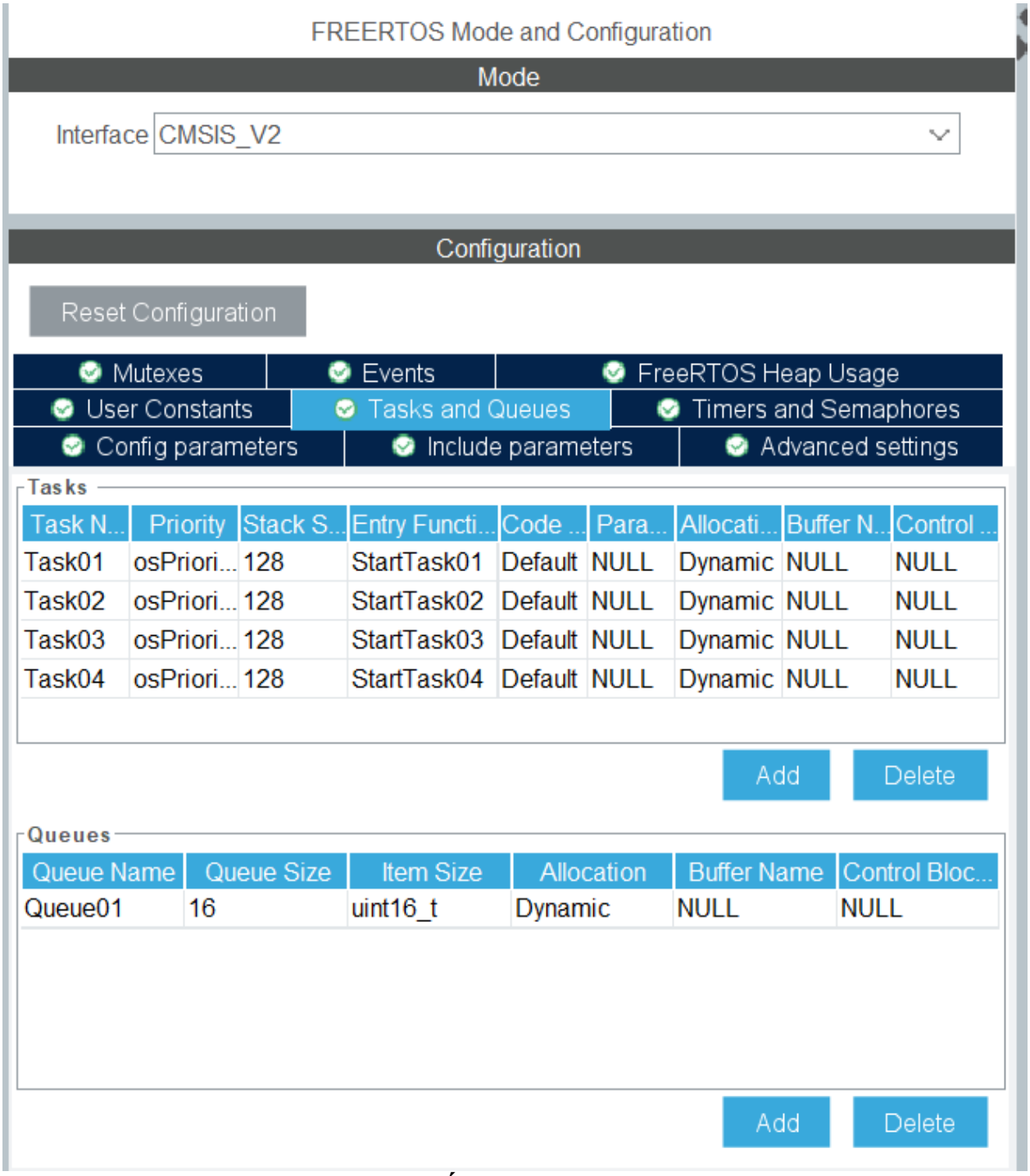
Hình 1. Cấu hình GPIO



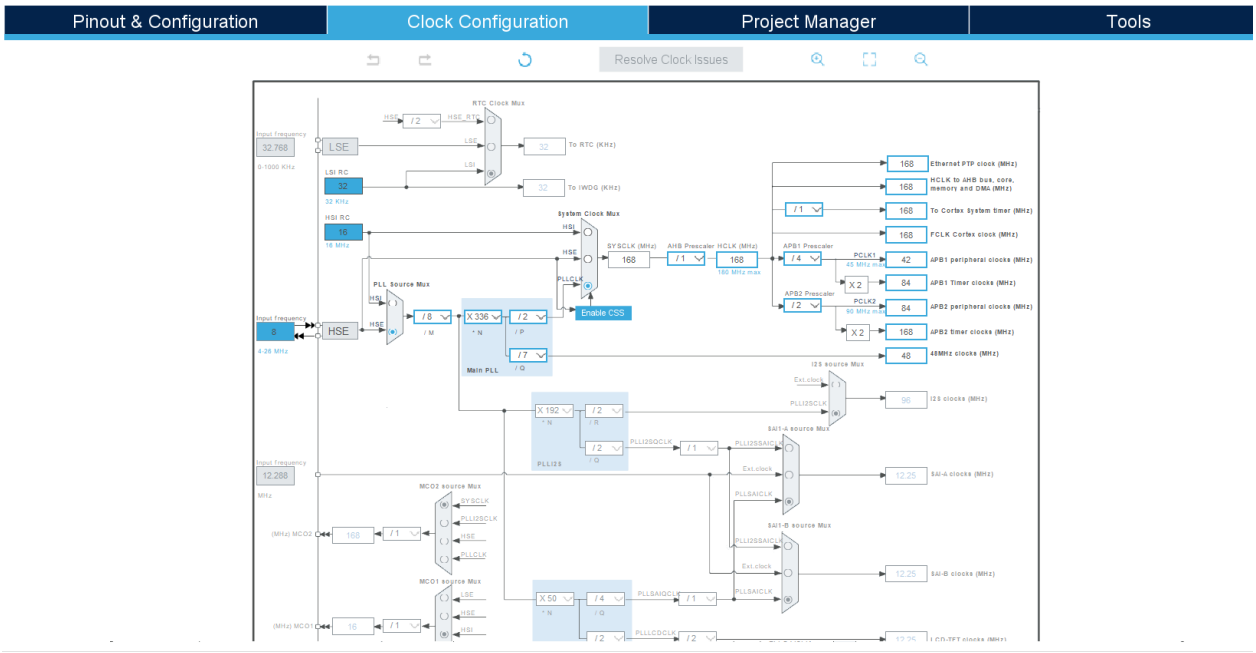
Hình 2. Cấu hình USB_OTG_HS



Hình 3. Cấu hình USB_DEVICE



Hình 4. Cấu hình FREE RTOS



Hình 5. Cấu hình xung clock

b) Hiện thực hệ thống

- Phần Include:

```
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "cmsis_os.h"
#include "usb_device.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "usbd_cdc_if.h"
#include "stm32f429i_discovery_sdram.h"
#include "stm32f429i_discovery_lcd.h"
#include "stm32f429i_discovery_gyroscope.h"
/* USER CODE END Includes */
```

Phần này liệt kê các file tiêu đề cần thiết cho project.Bao gồm các file liên quan đến ứng dụng chính, FreeRTOS, giao tiếp USB, LCD, con quay hồi chuyển và SDRAM.

- Private Typedef:

```
/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
typedef struct{
    int Buf[6];
} Mymessage;
/* USER CODE END PTD */
```

Định nghĩa một cấu trúc dữ liệu tùy chỉnh Mymessage là một mảng số nguyên (Buf) có 6 phần tử.

- Private define

```
/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
uint8_t buf1[10];
uint8_t buf2[10];
int realScore = 0;  //tính điểm
int flag = 0;       //cờ tổng để báo khi nào trò chơi kết thúc
int radius = 20;    //bán kính để vẽ hình tròn lên LCD, cũng là chiều cao
int hitflag = 0;    //cờ báo khi vợt đã đánh trúng bóng
float gyroValue[3]; //mảng 3 số thực để lấy giá trị của gyroscope
int X_circle = 115;
int Y_circle = 150;
/* USER CODE END PD */
```

Định nghĩa các biến toàn cục như bộ đệm (buf1 và buf2), các biến liên quan đến trò chơi (realScore, flag, radius, hitflag, X_circle, và Y_circle), và một mảng để lưu trữ giá trị con quay hồi chuyển (gyroValue).

- Hàm Main:

```
int main(void)
{
  HAL_Init();
  SystemClock_Config();
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  /* USER CODE BEGIN 2 */
  BSP_SDRAM_Init();
  MX_USB_DEVICE_Init();
  BSP_GYRO_Init();
  BSP_LCD_Init();//init LCD
  BSP_LCD_LayerDefaultInit(1, SDRAM_DEVICE_ADDR);//set the layer buffer address
into SDRAM
  BSP_LCD_SelectLayer(1);//select on which layer we write
  BSP_LCD_DisplayOn();//turn on LCD
  BSP_LCD_Clear(LCD_COLOR_BLUE);//clear the LCD on blue color
```

```
  BSP_LCD_SetBackColor(LCD_COLOR_BLUE);//set text background color
  BSP_LCD_SetTextColor(LCD_COLOR_WHITE);//set text color
  /* USER CODE END 2 */

  /* Init scheduler */
  osKernelInitialize();

  /* Create the queue(s) */
  /* creation of Queue01 */
  Queue01Handle = osMessageQueueNew (16, sizeof(Mymessage), &Queue01_attributes);

  /* USER CODE BEGIN RTOS_QUEUES */
  /* add queues, ... */
  /* USER CODE END RTOS_QUEUES */

  /* Create the thread(s) */
  /* creation of Task01 */
  Task01Handle = osThreadNew(StartTask01, NULL, &Task01_attributes);

  /* creation of Task02 */
  Task02Handle = osThreadNew(StartTask02, NULL, &Task02_attributes);

  /* creation of Task03 */
  Task03Handle = osThreadNew(StartTask03, NULL, &Task03_attributes);

  /* creation of Task04 */
  Task04Handle = osThreadNew(StartTask04, NULL, &Task04_attributes);

  /* Start scheduler */
  osKernelStart();

  /* We should never get here as control is now taken by the scheduler */
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

Hàm main khởi tạo hệ thống, các ngoại vi và các nhiệm vụ.Bao gồm cấu hình đồng hồ hệ thống, khởi tạo chân GPIO, LCD, USB, con quay hồi chuyển, và thiết lập nhân FreeRTOS.

- Các Task:

Có bốn Task (Task01, Task02, Task03, và Task04) được định nghĩa trong mã. Mỗi Task được thực thi đồng thời bởi bộ lập lịch FreeRTOS.

- Task01: Đọc dữ liệu con quay hồi chuyển, xử lý nó và gửi dữ liệu đã xử lý đến Queue01. Nó thiết lập hitflag khi một điều kiện cụ thể được đáp ứng (Đã trình bày chi tiết ở mục II).
- Task02: Điều khiển logic trò chơi liên quan đến bóng nảy trên màn hình LCD. Nó đọc dữ liệu từ Queue01 và điều chỉnh bán kính của bóng dựa trên dữ liệu con quay hồi chuyển. Trò chơi kết thúc khi bóng ra khỏi ranh giới (radius < 20). Nó cũng cập nhật điểm số (Đã trình bày chi tiết ở mục II).
- Task03: Truyền thông tin bán kính qua USB sử dụng giao diện CDC (Communication Device Class). Nó gửi dữ liệu mỗi 100 mili giây (Đã trình bày chi tiết ở mục II).
- Task04: Cập nhật màn hình LCD với điểm số hiện tại hoặc thông báo kết thúc trò chơi. Nó cũng xử lý tín hiệu LED và xóa các nhiệm vụ khác khi trò chơi kết thúc(Đã trình bày chi tiết ở mục II).

c) Yêu cầu về mức độ hoàn thành

Yêu cầu	Điểm	Hoàn thành
Hiển thị được giao diện ban đầu	2 điểm	X
Hiện thực được thao tác tâng bóng tại chỗ (điểm giữa màn hình)	3 điểm	X
Xuất thông số độ cao khi bóng được tâng ra máy tính qua Virtual Com Port	2 điểm	X
Hiện thực được chức năng tính điểm	3 điểm	X
Hiện thực được việc bóng bay theo 1 chiều dựa trên phương và chiều của lúc nâng bóng (nếu bóng bay ra rìa LCD sẽ đập ngược trở lại)	1 điểm	
Hiện thực được việc bóng bay theo 2 chiều dựa trên phương và chiều của lúc nâng bóng (nếu bóng bay ra rìa LCD sẽ đập ngược trở lại)	1 điểm	

Tài liệu tham khảo:

https://www.keil.com/pack/doc/CMSIS/RTOS2/html/group__CMSIS__RTOS__Message.html

https://hocarm.org/rtos-co-ban-phan-1/

https://hocarm.org/rtos-co-ban-phan-2/

https://www.st.com/resource/en/datasheet/l3gd20.pdf

https://stm32f4-discovery.net/2014/08/library-28-l3gd20-3-axis-gyroscope/

https://itecnotes.com/electrical/converting-raw-gyro-l3gd20h-values-into-angles/