BÁO CÁO MINI PROJECT:
SINGLE PING-PONG GAME

Nhóm thực hiện

STT	MSSV	Họ và tên	Ghi chú
1	21520039	Bùi Đăng Huy	
2	21520416	Trần Hải Quang	
3	21520447	Nguyễn Hoàng Thân	
4	21522580	Kha Quốc Thái	

Yêu cầu về mức độ hoàn thành đã thực hiện

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	X
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	X

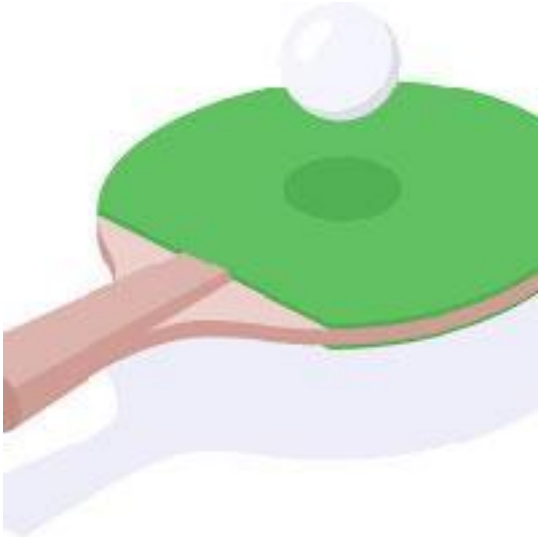
Link video demo: https://youtu.be/ONzMu009mS8

Link source code project: dh1501/MiniProject_Pingpong (github.com)

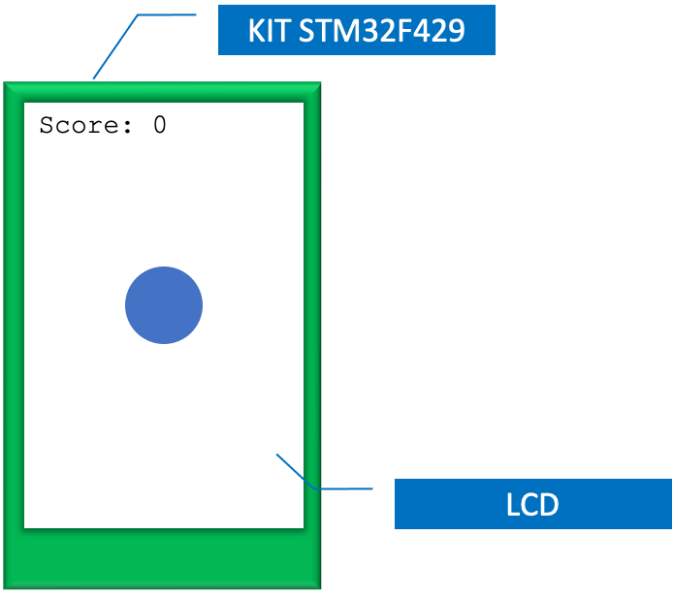
1 TỔNG QUAN

1.1. Mô tả về chức năng

- 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; giao diện trò chơi và cách chơi thể hiện ở Hình 2 và Hình 3.



Hình 1. Mô tả game tâng bóng bàn



Trạng thái ở hệ thống
cân bằng ban đầu

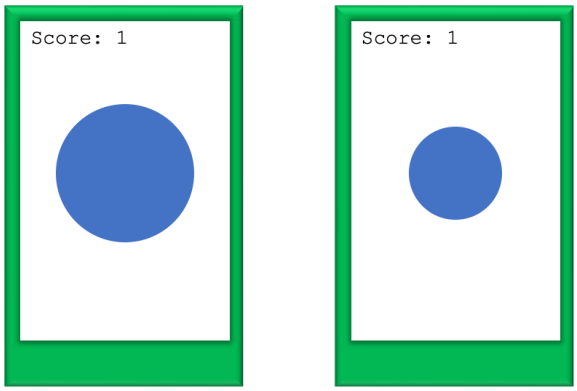
Hình 2. Trạng thái bắt đầu của game

Động tác tâng bóng

Hạ KIT xuống

Nâng KIT lên để tâng bóng

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

- Đồ án ứng dụng đọc giá trị cảm biến Gyroscope xác định trạng thái của vợt và tính toán độ cao tâng bóng, hướng tâng bóng; giao diện được hiển thị qua LCD; ngoài ra, độ cao bóng được xuất ra máy tính qua Virtual Com Port.
- Với các yêu cầu và nội dung trên, ta sử dụng FreeRTOS để lập lịch thực hiện các task theo yêu cầu. Xây dựng ý tưởng thực hiện các task như bảng 1.

Bảng 1. Các task chức năng

	Mô tả chức năng
Task1	Cấu hình cho LCD và cảm biến Gyroscope; đọc giá trị thô của Gyroscope
Task2	Xử lý hình ảnh,tính điểm khi tâng, hạ vợt và vợt đánh trái phải
Task3	Truyền độ cao bóng qua cổng Virtual Com Port
Task4	Xử lí tín hiệu reset, tín hiệu đèn và game over

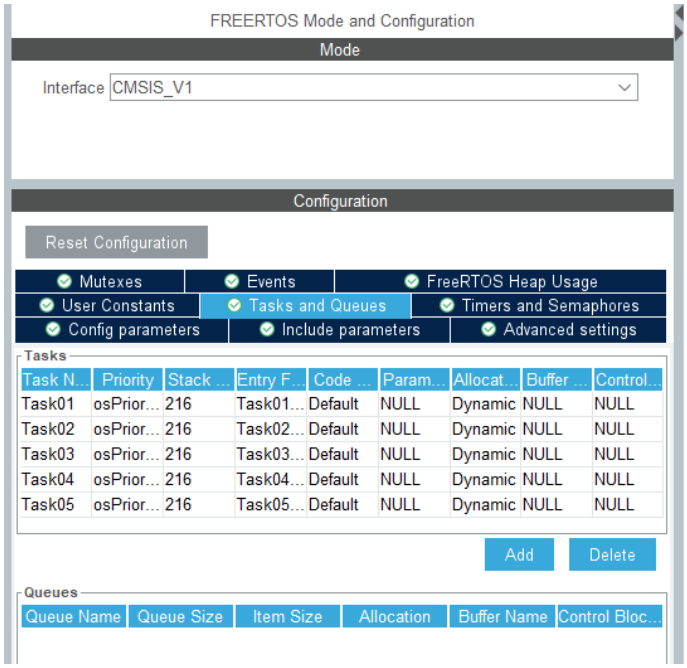
1.2. Các ứng dụng sử dụng

- **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.

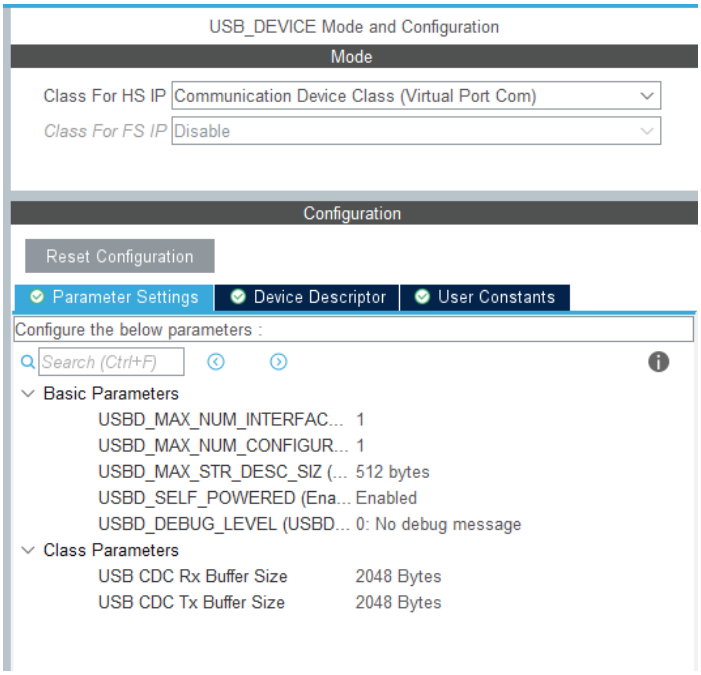
2 THỰC HIỆN

Thực hiện khai báo và cấu hình các phần cần thiết cho đồ án theo các task đã được đề xuất ở Bảng 1.

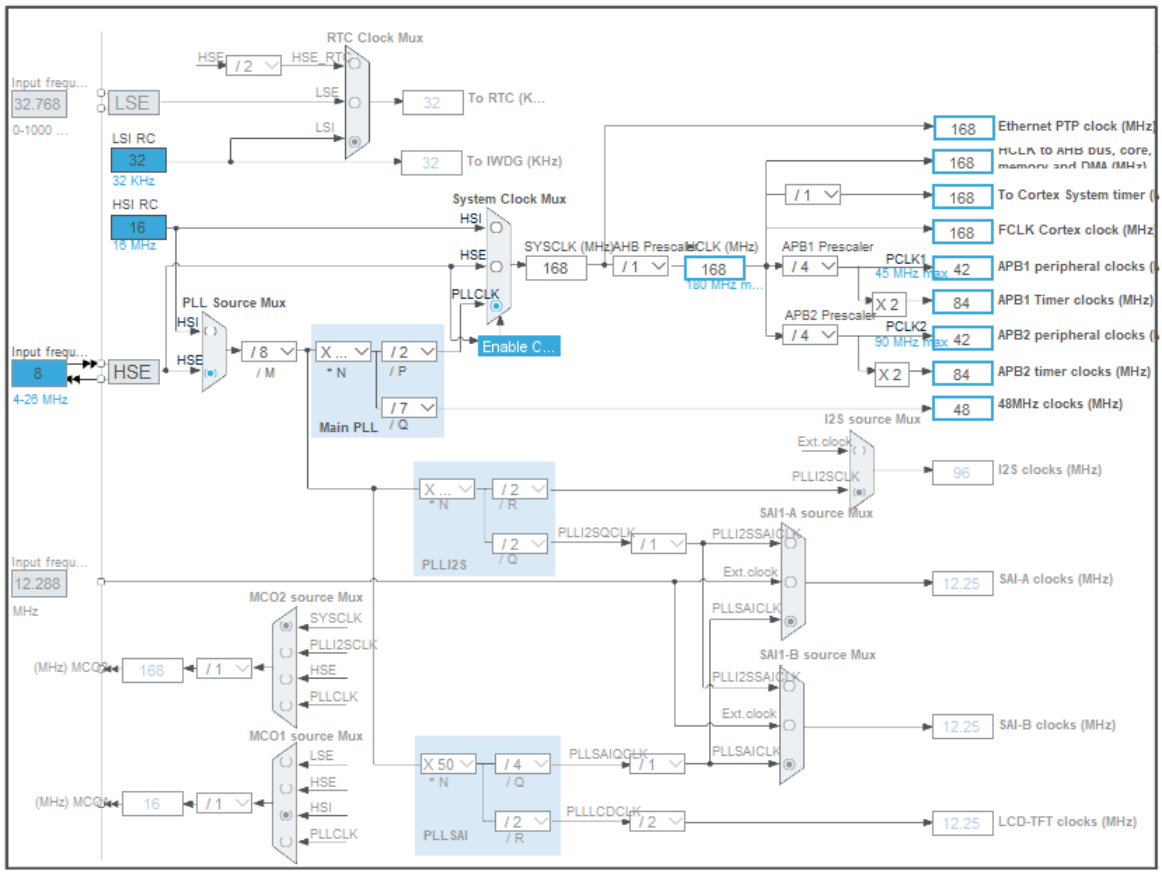
2.1. Thực hiện cấu hình:



Hình 5. Cấu hình các task FreeRTOS



Hình 6. Cấu hình USB DEVICE



Hình 7. Cấu hình xung clock

2.2. Task 1 - Cấu hình cho LCD, Gyroscope; đọc giá trị củaGyroscope

- Sử dụng thư viện hỗ trợ `"stm32f429i_discovery_sdram.h"` và `"stm32f429i_discovery_lcd.h"`
- Để cấu hình hiển thị LCD, ta thực hiện cấu hình SRAM, cá hình Layer, trạng thái bật tắt của màn hình, background color, text color và font chữ với các hàm sau

```
//Init LCD
	BSP_LCD_Init();
  //LCD default
	BSP_LCD_LayerDefaultInit(1,SDRAM_DEVICE_ADDR);
	BSP_LCD_SelectLayer(1);
	BSP_LCD_DisplayOn();
	BSP_LCD_Clear(LCD_COLOR_DARKGREEN);
	BSP_LCD_SetBackColor(LCD_COLOR_DARKGREEN);
	BSP_LCD_SetTextColor(LCD_COLOR_WHITE);

	BSP_LCD_SetFont(&Font20); //Font
	BSP_LCD_GetFont();
```

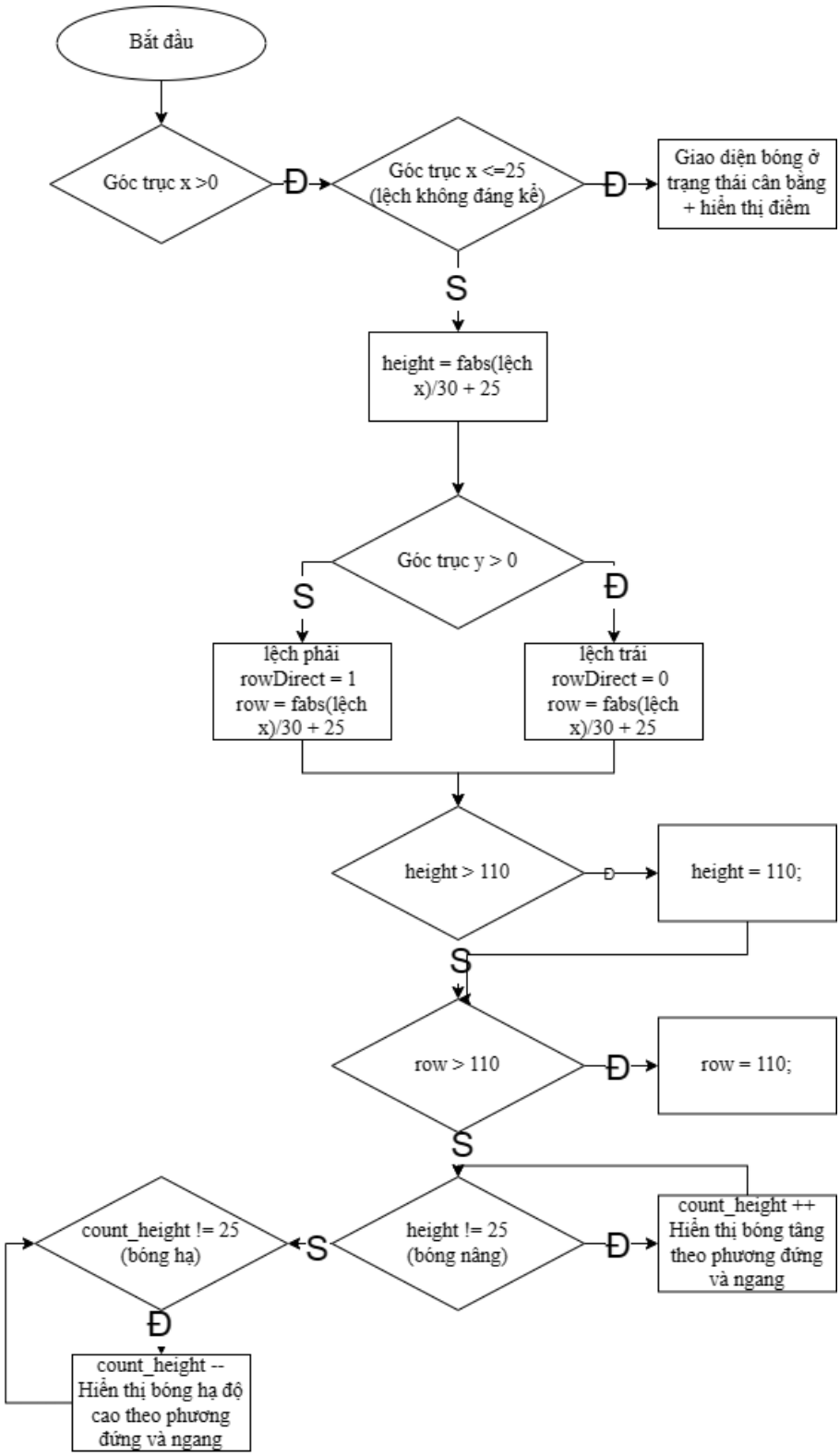
- Đối với giá trị cảm biến Gyroscope, ta sẽ tiến hành đọc giá trị sau mỗi 100ms. Giá trị đọc được là tọa độ các trục x, y,z. Ta sẽ tính góc lệch dựa trên tọa độ đọc được.

```
for(;;)
  {
        //read GYROSCOPE
         BSP_GYRO_GetXYZ(xyz_rotation);

         for(int i = 0;i<3;i++)
         {
              xyz_ang_rate[i] = (xyz_rotation[i]/2000)*70; //angular rate
         }

         osDelay(100);
  }
```

Hình 8. Giao diện ban đầu của game

2.3. Task 2 - Xử lý hình ảnh, tính điểm khi tâng, hạ vợt và đánh vợt trái phải

- Sử dụng góc lệch của trục x được lưu ở biến xyz_ang_rate[0] để xác định trạng thái vợt được tâng hay hạ. Nếu góc < 25 độ, vợt đang ở trạng thái cân bằng, thực hiện hiển thị vợt ở vị trí ban đầu. Nếu góc lệch > 25 độ, thực hiện tính taosn giá trị độ cao và xác định hướng nghiêng trái hoặc phải qua trục y để xác định hướng đánh của vợt.
- Với chiều cao vào hướng đánh vợt lệch tối đa là 110 đơn vị, nếu ghi nhận chiều cao hoặc hướng đánh vượt quá tối đa. Thực hiện gán giá trị tối đa.
- Để xác định hướng đánh vợt là trái hay phải, sử dụng độ lệch trục Y và lưu lại độ lệch tại biến row. Khi trục y ở trạng thái cân bằng, nếu góc lệch dương thì đánh sang trái (rowDirect = 0) ngược lại là đánh sang phải (rowDirect = 1)
- Khi đã đạt trạng thái tâng bóng (height != 25) thực hiện cộng dần giá trị độ cao, đồng thời hiển thị bóng theo từng độ cao với 10ms delay. Sử dụng hàm `BSP_LCD_DrawCircle(BSP_LCD_GetXSize() + row, RollY_Value, count_height);` và `_LCD_FillCircle(BSP_LCD_GetXSize() + row, RollY_Value, count_height);` để hiện thị bóng trên màn ảnh với row là biến ghi nhận độ lệch trái phải và count_height là chiều cao bóng được tâng.



Hình 9. Sơ đồ giải thuật Task 2

- Hiện thực giải thuật ở Task 2 với các hàm sau

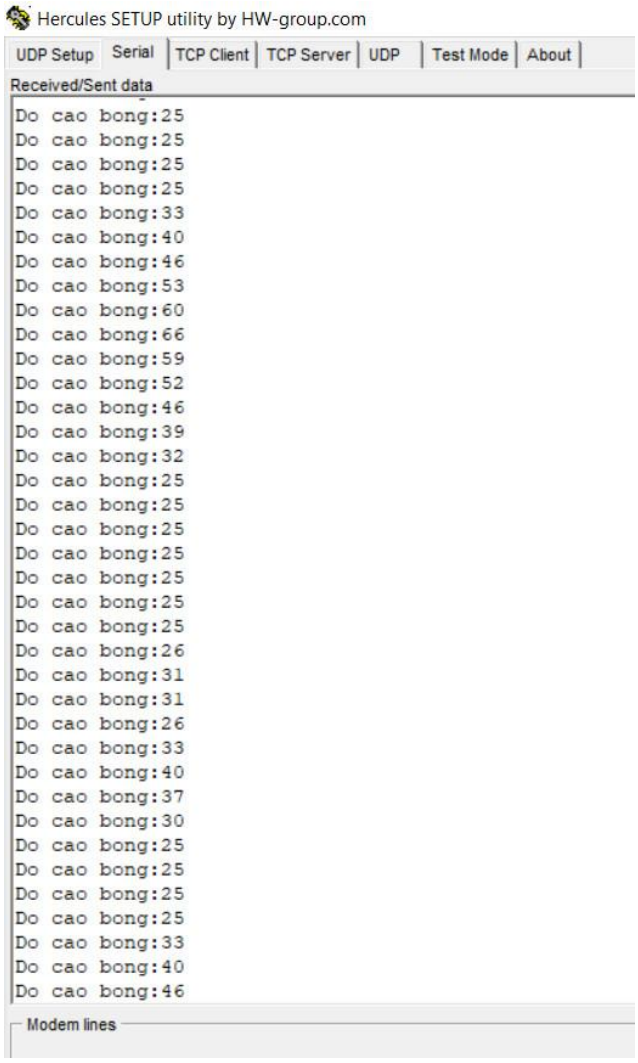
```
void Task02_Init(void const * argument)
{
  for(;;)
  {

    if(xyz_ang_rate[0] >= 0){ //x

    if(xyz_ang_rate[0] <= 25) //trang thai can bang
    {
       BSP_LCD_Clear(LCD_COLOR_DARKGREEN);
       BSP_LCD_DrawCircle(BSP_LCD_GetXSize() - 120, 160, 25);
       BSP_LCD_FillCircle(BSP_LCD_GetXSize() - 120, 160, 25);
       sprintf(str,"%d",count_score);
       BSP_LCD_DisplayStringAt(11,13,display_score,LEFT_MODE);
       BSP_LCD_DisplayStringAt(100,13,(uint8_t*)str,LEFT_MODE);

       osDelay(50);

    }
    else // bong da duoc tang len
    {
       height = fabs(xyz_ang_rate[0])/30 + 25;
       if (xyz_ang_rate[1] > 0) rowDirect = 0; else rowDirect = 1;
       row    = fabs(xyz_ang_rate[1])/30 + 20; //gyro lech theo phuong ngang
       if(height > 110) height = 110; // height <= 110
       if(row > 110) height = 110; //row <= 110
       flag = 1;
       while(height!=25){  //Trang thai nang
              if(count_height == height) break;
              ++count_height;
              if (rowDirect == 0){
       BSP_LCD_Clear(LCD_COLOR_DARKGREEN);
       BSP_LCD_DrawCircle(BSP_LCD_GetXSize() + row, RollY_Value, count_height);
       BSP_LCD_FillCircle(BSP_LCD_GetXSize() + row, RollY_Value, count_height);
              }
              else {
       BSP_LCD_Clear(LCD_COLOR_DARKGREEN);
       BSP_LCD_DrawCircle(BSP_LCD_GetXSize() - row, RollY_Value, count_height);
       BSP_LCD_FillCircle(BSP_LCD_GetXSize() - row, RollY_Value, count_height);
              }
       BSP_LCD_DisplayStringAt(11,13,display_score,LEFT_MODE);
       BSP_LCD_DisplayStringAt(100,13,(uint8_t*)str,LEFT_MODE);
        osDelay(10);
       } //end while

       while(count_height != 25) //Trang thai ha
       {
              count_height--;
              if (rowDirect == 0){
                  BSP_LCD_Clear(LCD_COLOR_DARKGREEN);
```

```
        BSP_LCD_DrawCircle(BSP_LCD_GetXSize() - row, RollY_Value, count_height);
        BSP_LCD_FillCircle(BSP_LCD_GetXSize() - row, RollY_Value, count_height);
              }
              else {
        BSP_LCD_Clear(LCD_COLOR_DARKGREEN);
        BSP_LCD_DrawCircle(BSP_LCD_GetXSize() + row, RollY_Value, count_height);
        BSP_LCD_FillCircle(BSP_LCD_GetXSize() + row, RollY_Value, count_height);
        }
              sprintf(str,"%d",count_score);
              BSP_LCD_DisplayStringAt(11,13,display_score,LEFT_MODE);
              BSP_LCD_DisplayStringAt(100,13,(uint8_t*)str,LEFT_MODE);
              osDelay(10);
        }
        count_height = 25; //reset lai bien dem do do cao bong
        }
        }
          osDelay(1);
  }
}
```

2.4. Task 3 - Truyền độ cao bóng qua cổng Virtual Com Port

- Với cổng Virtual Com Port, ta thực hiện truyền thông số độ cao được tính toán khi đọc Gyroscope sử dụng hàm truyền CDC_Transmit_HS().

```
void Task03_Init(void const * argument) //Hien thi do cao bong
{
  for(;;)
  {
       sprintf(msg_buf,"Do cao bong:%d\n", count_height);
       CDC_Transmit_HS(msg_buf,strlen((const char*)msg_buf));
    osDelay(50);
}
```



Hình 10. Truyền thông số độ cao bóng qua Virtual Com Port

2.5. Task 4 - Xử lí tín hiệu reset, tín hiệu đèn và game over

- Ở Task 2, ta sử dụng cờ flag để xác định trạng thái bóng được tâng hợp lệ, ở Task 4, ta sẽ sử dụng cờ này để xuất các tín hiệu cho đèn và tính điểm.
- Sử dụng biến compare_height để xác ddnhj trạng thái game over. Compare_height xác định bằng công thức xác định độ cao. Nếu compare_height <= 60 → Tâng bóng chưa thành công sẽ trả trạng thái game over.
- Khi game over, xóa các task 1, 2, 3 chỉ giữ lại task 4 để hiển thị giao diện kết thúc và điểm.

```
void Task04_Init(void const * argument) //Game over
{
  for(;;)
  {

       if(height >= 40 && height <= 110 && flag == 1)
       {
              ++count_score;
              HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13, GPIO_PIN_SET);
              osDelay(1700);
              HAL_GPIO_WritePin(GPIOG, GPIO_PIN_13, GPIO_PIN_RESET);

                flag = 0;
                    //get current height
                    compare_height = fabs(xyz_ang_rate[0])/30 + 25;
                    if(compare_height > 110)
                    {
                        compare_height = 110;
                    }
                    if(compare_height >= 60)
                    {
                           //---- GAME OVER SCREEN ----
                           //BEGIN
                    BSP_LCD_Clear(LCD_COLOR_RED);
                    BSP_LCD_SetBackColor(LCD_COLOR_RED);
                    BSP_LCD_DisplayStringAt(1, 130, "GAME OVER", CENTER_MODE);
                     BSP_LCD_DisplayStringAt(1,150,display_score,CENTER_MODE);
                    BSP_LCD_DisplayStringAt(1,175,(uint8_t*)str,CENTER_MODE);
                            //END
                   HAL_GPIO_WritePin(GPIOG, GPIO_PIN_14, GPIO_PIN_SET);
                     vTaskDelete(Task01_Init);
                      vTaskDelete(Task02_Init);
                      vTaskDelete(Task03_Init);
                    }

       }
       osDelay(1);
  }
  /* USER CODE END Task04_Init */
}
```

Hình 11. Trạng thái game over

3 BÀI TẬP

Bài tập 1: Thiết kế mô hình phần cứng cứng thiết cho yêu cầu trên: cần dùng những phần cứng nào, sử dụng các giao thức giao tiếp gì giữa các phần cứng? **(15% số điểm)**

Trả lời: Phần cứng đồ án sử dụng ngoại vi tích hợp sẵn trên kit STM32F429 với các thiết bị và giao thức như sau:

- 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

Các nội dung cụ thể xem tại mục **1.2. Các chức năng sử dụng.**

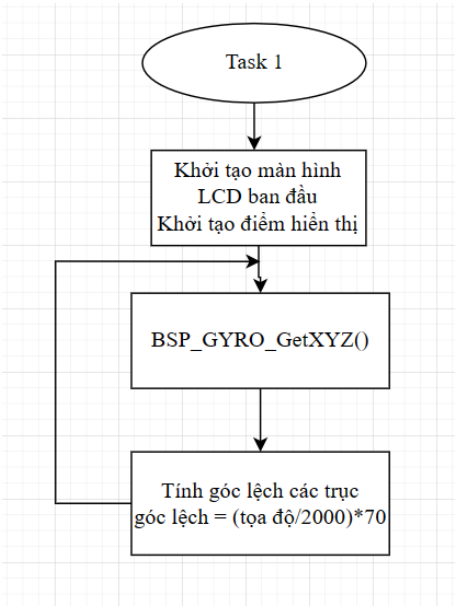
Bài tập 2: Sử dụng RTOS, thiết mô hình phần mềm cho yêu cầu trên: nêu công việc của từng task, luồng xử lý dữ liệu như thế nào? **(15% số điểm)**

Trả lời: Chia các yêu cầu của đồ án thành 4 task chính và phân công các task thực hiện công việc như sau:

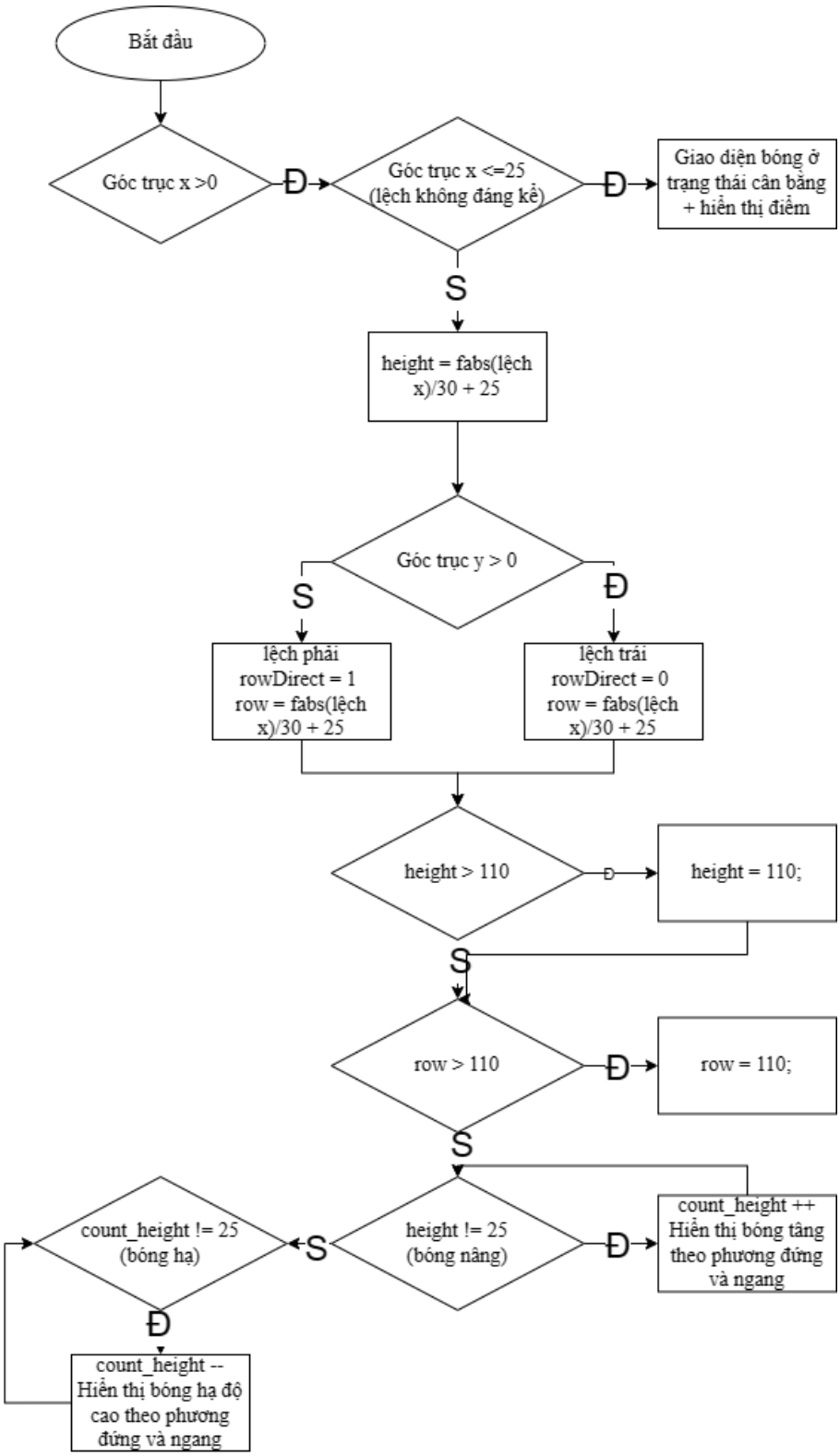
	Mô tả chức năng
Task1	Cấu hình cho LCD và cảm biến Gyroscope; đọc giá trị thô của Gyroscope
Task2	Xử lý hình ảnh,tính điểm khi tâng, hạ vợt và vợt đánh trái phải
Task3	Truyền độ cao bóng qua cổng Virtual Com Port
Task4	Xử lí tín hiệu reset, tín hiệu đèn và game over

Bài tập 3: Thiết kế lưu đồ giải thuật xử lý cho yêu cầu trên? **(30% số điểm)**

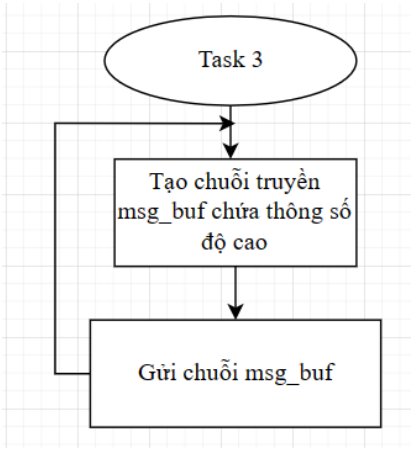
Trả lời: Sơ đồ giải thuật các task được trình bày ở các hình sau



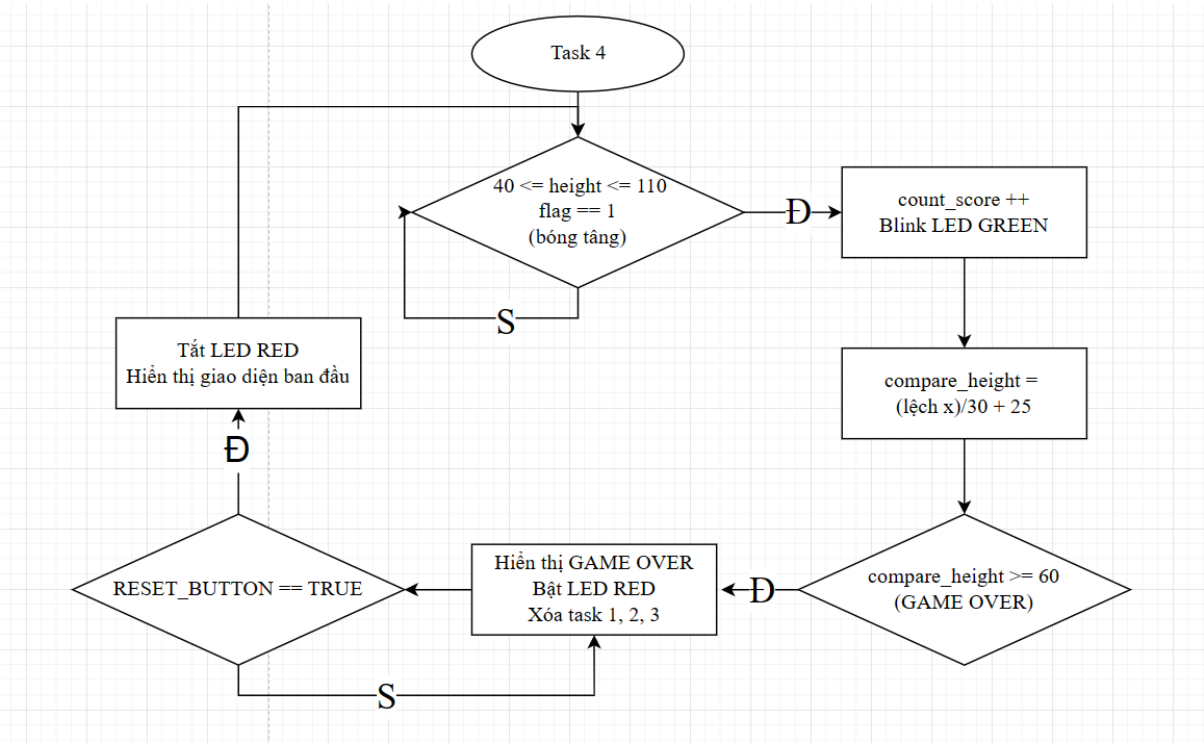
Hình 12. Sơ đồ giải thuật Task 1



Hình 13. Sơ đồ giải thuật Task 2



Hình 14. Sơ đồ giải thuật Task 3



Hình 15. Sơ đồ giải thuật Task 4

Bài tập 4: Hiện thực hệ thống và báo cáo kết quả? **(40% số điểm)**

Trả lời: Thực hiện và trình bày ở mục **2. THỰC HIỆN**

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/