ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KỸ THUẬT MÁY TÍNH

BÁO CÁO ĐỒ ÁN MÔN HỌC

SINGLE PING-PONG GAME

Giảng viên hướng dẫn: **Chung Quang Khánh**

Lớp: **CE224.O12**

Nhóm thực hiện: Nhóm 2

Thành viên:

Nguyễn Hoàng Khánh Duy	21522002
Nguyễn Tuấn Kiệt	21521036
Lưu Tuấn Tài	21522564
Đặng Tấn Đạt	21521927

TP. HỒ CHÍ MINH – Tháng 1 năm 2024

MỤC LỤC

DANH MỤC HÌNH ẢNH

CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI, MỤC TIÊU

1. Giới thiệu

Trong cuộc cách mạng công nghiệp lần thứ 4 hiện nay thì ngành nhúng đang có sự phát triển vượt bậc, việc ứng dụng các thiết bị vi điều khiển trở nên dễ tiếp cận và dễ dàng ứng dụng vào trong đời sống. Nhóm xin giới thiệu đề tài Single Ping-Pong Game. Sản phẩm là kiến thức được nhóm em vận dụng từ môn học và đưa ra sản phẩm với mong muốn có chức năng hữu ích cho một vài công việc cụ thể nào đó. Đề tài sử dụng KIT STM32F4 Discovery for STM32F429 MCU; là một bộ phát triển phần cứng được thiết kế để giúp các nhà phát triển dễ dàng phát triển các ứng dụng với các vi điều khiển STM32F429 cao cấp với lõi ARM Cortex-M4. Bộ phát triển này có nhiều tính năng hữu ích đi kèm với nhiều phần mềm miễn phí, bao gồm các ví dụ và thư viện chuẩn để giúp các nhà phát triển dễ dàng bắt đầu phát triển ứng dụng của mình.

2. Mục tiêu

Mục tiêu của đề tài này:

- ❖ Biết cách tạo project và cấu hình project sử dụng STM32CubeIDE.
- ❖ Hiểu về cách thức giao tiếp qua serial.
- ❖ Hiểu được cách giao tiếp với cách giao tiếp với các ngoại vi như Gyroscope, màn hình LCD.
- ❖ Có kiến thức vững về Hệ điều hành, đặc biệt là vấn đề lập lịch và đồng bộ các tiến trình.

CHƯƠNG 2: TỔNG QUAN ĐỀ TÀI

1. Chuẩn bị

1.1. Phần cứng

- KIT STM32F4 Discovery for STM32F429 MCU.

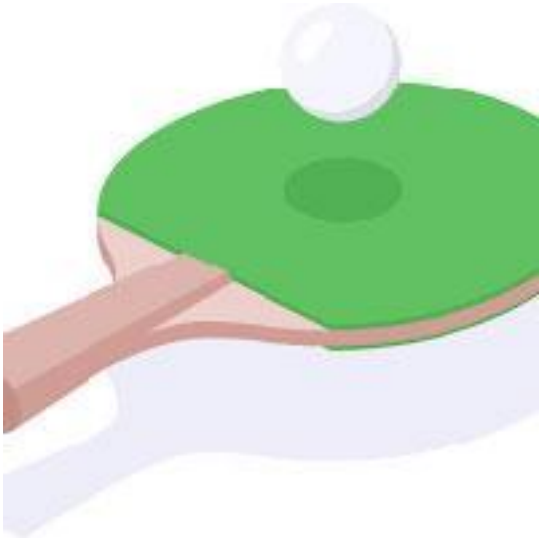
Hình 1 KIT STM32F4 Discovery

1.2. Phần mềm

- STM32CubeIDE: sử dụng để lập trình, build, nạp và debug code.

2. Mô tả

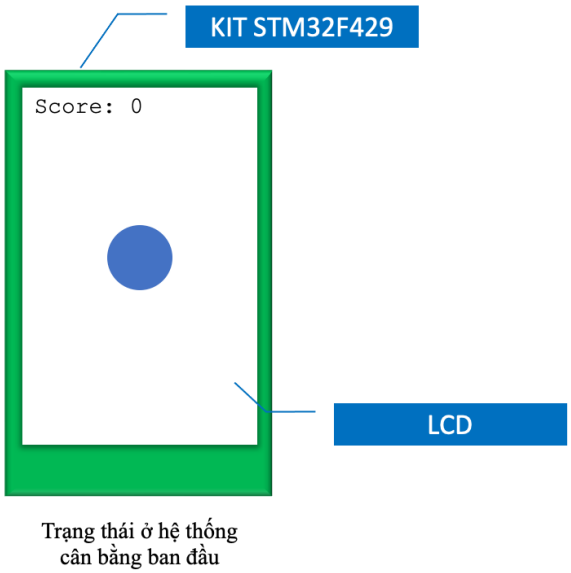
Single Ping-Pong Game là một trò chơi mô phỏng việc dùng vợt và tâng bóng bàn được mô tả như trong Hình 1, trong đó, KIT STM32F429 được sử dụng như cây vợt, màn hình LCD sẽ hiển thị một hình tròn mô phỏng trái bóng bàn.



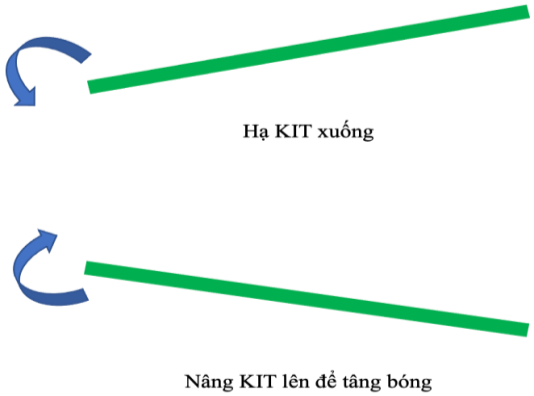
Hình 2. Mô tả game bóng bàn

Nguyên lý của game như sau:

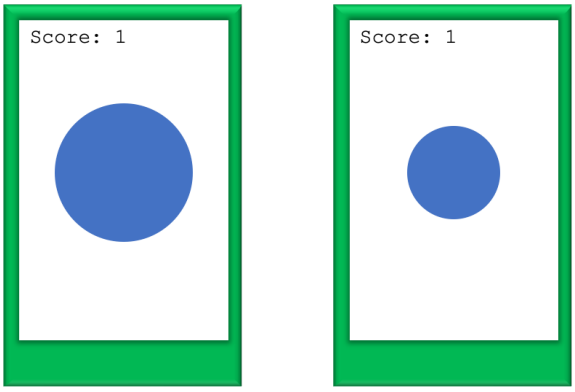
- Đầu tiên, hệ thống ở trạng thái cân bằng khởi đầu.
- Người chơi sẽ hạ và nâng KIT để thực hiện động tác tâng bóng, khi đó trên màn hình LCD sẽ mô phỏng trạng thái trái bóng được nâng lên và rơi xuống.
- Mục tiêu của người chơi là phải nâng KIT đúng lúc trái bóng rơi xuống, đập vào "mặt vợt" và tâng lên lại, người chơi được tính điểm, đèn xanh chớp 1 lần.
- Nếu người chơi tâng 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 người dùng nhất nút reset trên KIT.



Hình 3. Trạng thái đầu tiên của game



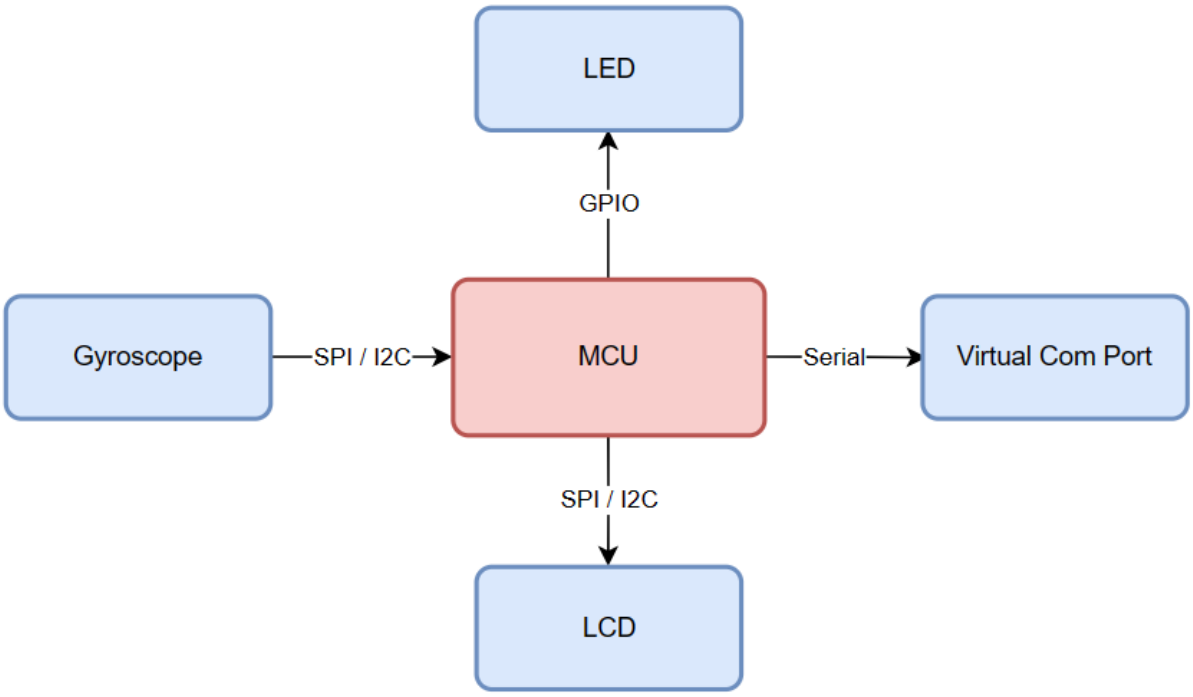
Hình 4 Mô phỏng động tác tâng bóng



Hình 5. Mô phỏng bóng bay lên và rơi xuống

CHƯƠNG 3: THIẾT KẾ, CHƯƠNG TRÌNH

1. Mô hình phần cứng



Hình 6. Các khối chức năng của hệ thống

1.1. Gyroscope

Gyroscope, hay còn được gọi là cảm biến góc quay, là một thiết bị đo độ quay hoặc tốc độ góc của một vật thể xung quanh một trục quay. Trong đề tài này nó được dùng để đọc chuyển động của board khi đánh lên bóng. Gyroscope đóng vai trò quan trọng trong việc cung cấp thông tin về chuyển động và hướng, đặc biệt là khi kết hợp với các cảm biến khác như cảm biến gia tốc và từ đó cung cấp dữ liệu chính xác về vị trí và hướng của vật thể trong không gian.

1.2. LED

Hiển thị trạng thái của game theo như yêu cầu đề tài giúp thuận tiện cho người sử dụng. Đèn đỏ được bật để thông báo “Game over”.

1.3. LCD

Một màn hình cảm ứng TFT LCD chất lượng cao, màn hình có kích thước 4.3inch với độ phân giải thông thường là 480 x 272 pixels, đảm bảo hiển thị hình ảnh và văn bản sắc nét.

Chức năng: Hiển thị trạng thái của trò chơi, hình ảnh của bóng, điểm số,…

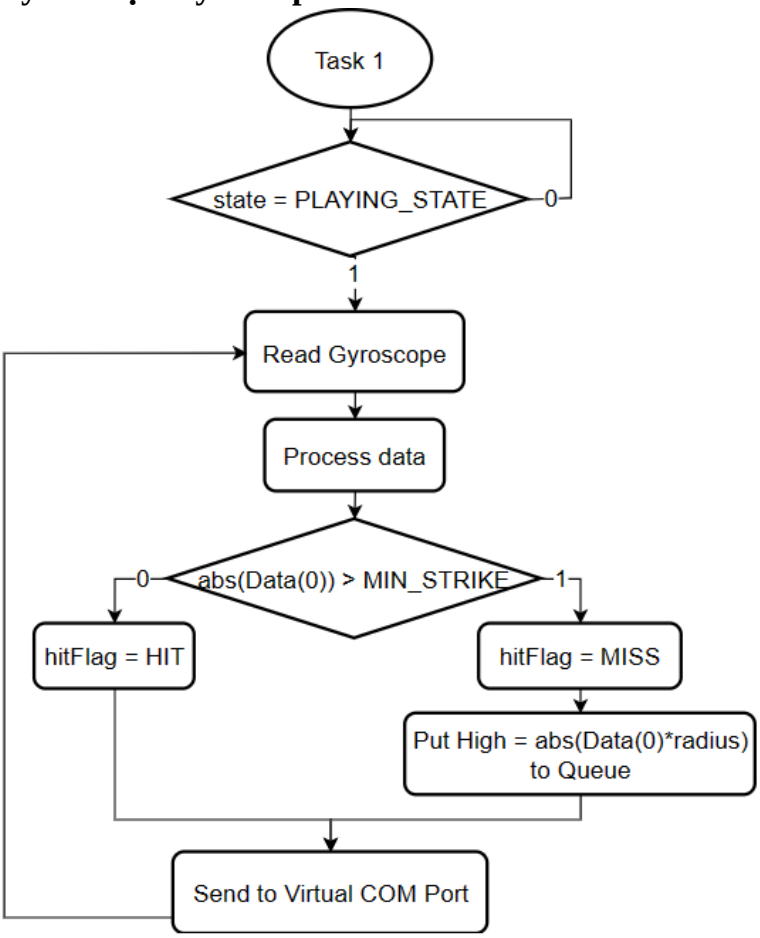
1.4. Virtual Com Port

Virtual COM port (Cổng COM ảo) là một công nghệ giúp tạo ra một kết nối COM (Communication Port - Cổng Giao Tiếp) ảo trên máy tính mà không cần sử dụng cổng vật lý. Điều này cho phép truyền dữ liệu giữa máy tính và thiết bị thông qua giao diện COM ảo như một cổng COM tiêu chuẩn.

Chức năng: truyền dữ liệu đọc được từ gyroscope ra máy tính.

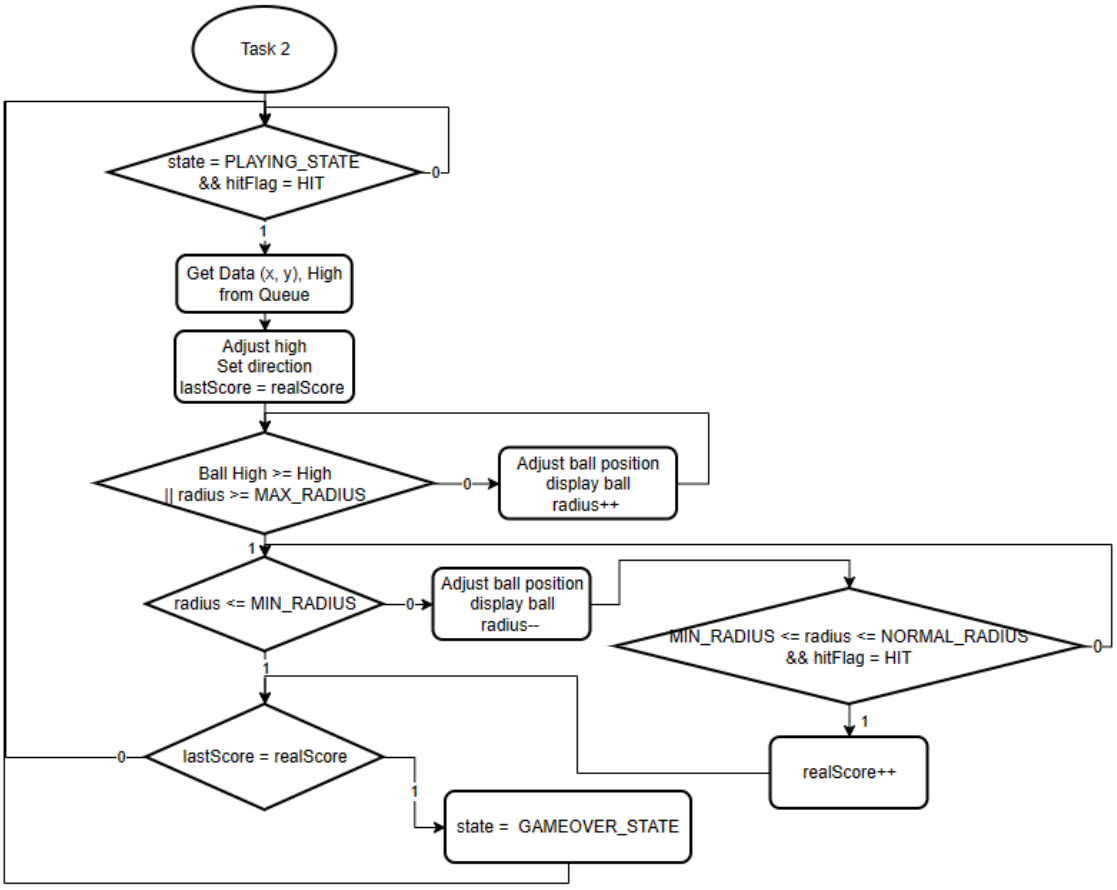
2. Lưu đồ giải thuật

2.1. Xử lý tín hiệu Gyroscope



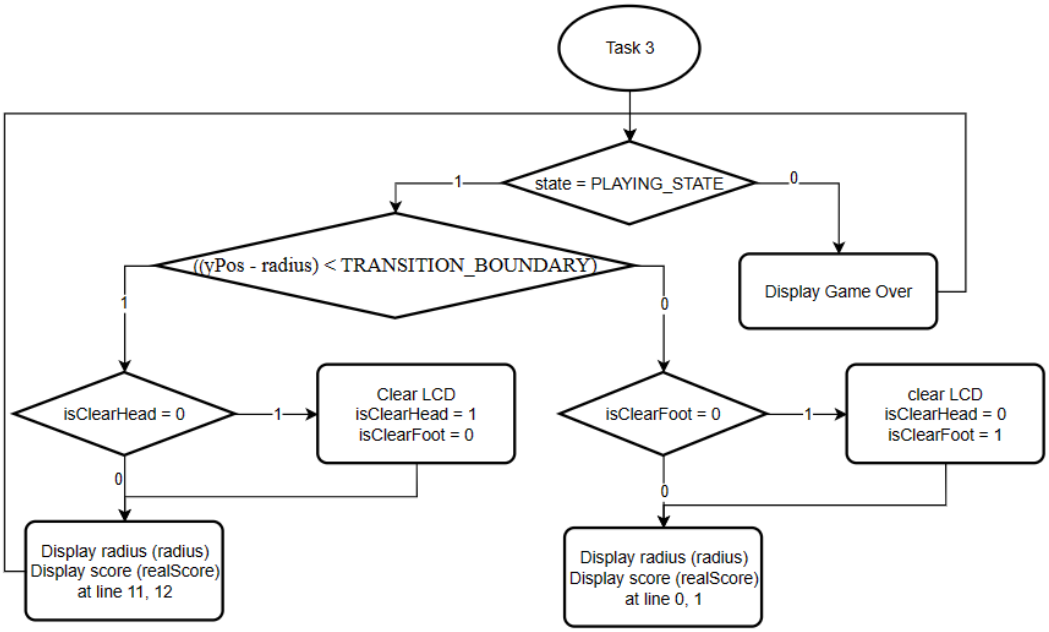
Hình 7. Lưu đồ giải thuật Task 1

2.2. Thuật toán điều chỉnh tâm bóng và bóng bay lên xuống



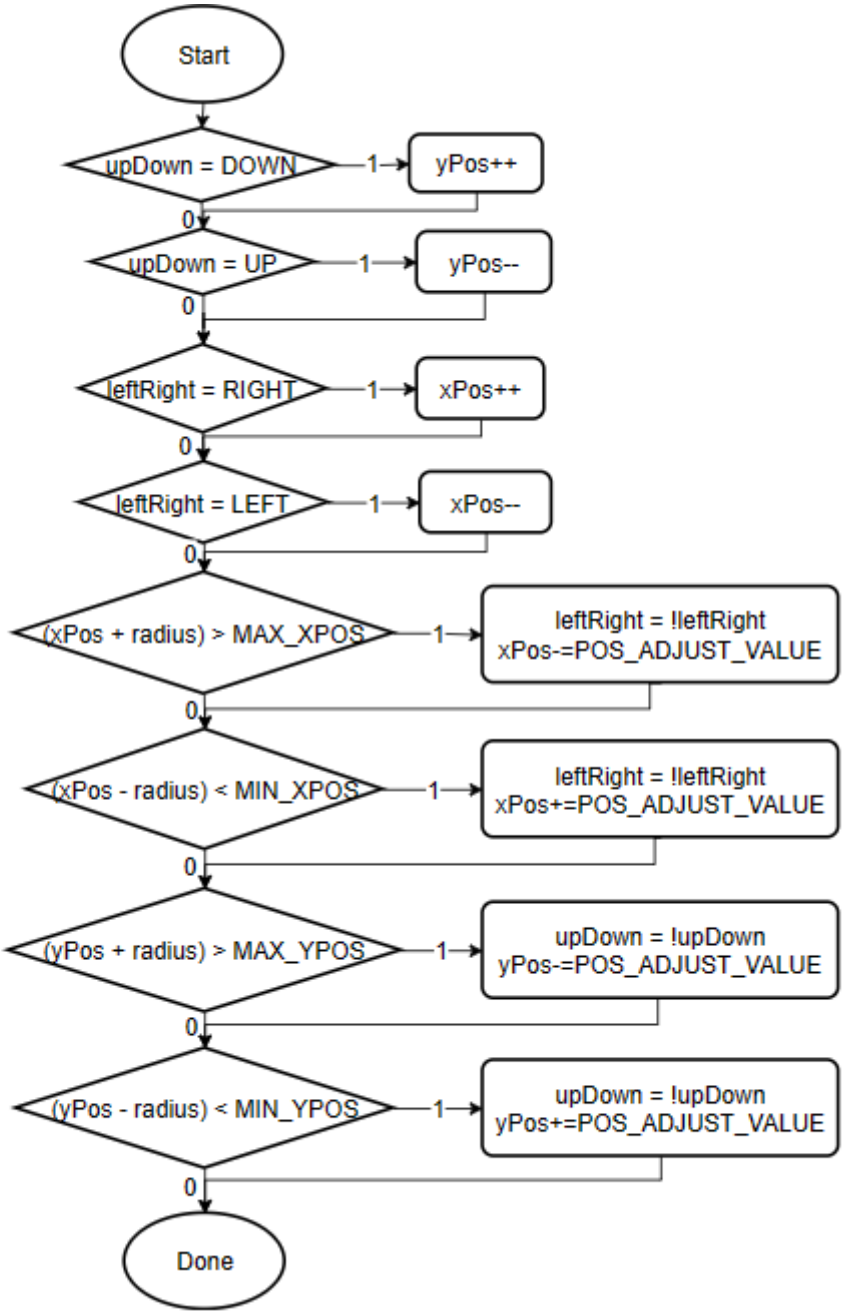
Hình 8. Lưu đồ giải thuật Task 2

2.3. Kiểm tra trạng thái trò chơi



Hình 99. Lưu đồ giải thuật Task 3

2.4. Hàm điều chỉnh vị trí bóng



Hình 10. Hàm điều chỉnh vị trí bóng

3. Chương trình và Giải thích

3.1. Xử lý tín hiệu Gyroscope

```
//Read and process gyro data
void StartTask01(void *argument)
{
 /* init code for USB_DEVICE */
 MX_USB_DEVICE_Init();
 /* USER CODE BEGIN 5 */
 /* Infinite loop */
 for(;;)
 {
  if(state == PLAYING_STATE)
  {
   //Read data from gyroscope l3gd20
   BSP_GYRO_GetXYZ(dataRec);
   for(int i=0; i<3; i++)
   {
        dataRec[i] = dataRec[i]*0.001;
    }
   //Check state is "HIT" or "MISS"
   if(abs(dataRec[0]) > MIN_STRIKE)
   {
    hitFlag = HIT;
     dataRec[3] = abs(dataRec[0]*radius);
     for(int i=0; i<4; i++)
     {
      osMessageQueuePut(myQueue01Handle, &dataRec[i], 0, 0);
     }
   }
    else
    {
    hitFlag = MISS;
    }
   sprintf(buffer1, "x:%0.1f;y:%0.1f;z:%0.1f\n", dataRec[0], dataRec[1], dataRec[2]);
   CDC_Transmit_HS(buffer1, sizeof(buffer1));

    osDelay(100);
  }
```

```
  }
  /* USER CODE END 5 */
}
```

- Giải thích:

- Đầu tiên kiểm tra trạng thái chương trình nếu đang trong trạng thái PLAYING thì thực hiện đọc gyro.
- Gọi hàm **BSP_GYRO_GetXYZ(dataRec):** lấy giá trị từ cảm biến gia tốc(gyroscope) và lưu nó trong mảng 'dataRec' đã được khai báo.
- Xử lý giá trị lấy từ cảm biến và kiểm tra gia tốc: Mỗi phần tử trong mảng **dataRec**[] được nhân với 0.001 để chuyển quy đổi ra giá trị dễ sử dụng hơn. Kiểm tra nếu giá trị tuyệt đối của **dataRec[0]** (giá trị gia tốc trục x) lớn hơn 20. Nếu là đúng, **hitFlag** được đặt là 1 (hit) còn không thì là 0 (miss)
- Gửi dữ liệu đã xử lý đến Queue: Nếu **hitFlag** là 1 (hit), giá trị tuyệt đối của **dataRec[0]** được nhân với **radius** để tính toán độ cao bóng bay lên và lưu vào **dataRec[3]**, sau đó mỗi giá trị trong **dataRec** được đặt vào một hàng đợi (**myQueue01Handle**). Việc đưa vào Queue để có thể truyền dữ liệu từ task này sang task khác.
- Truyền Dữ Liệu Xuống USB CDC: Sử dụng hàm CDC_Transmit_HS để truyền dữ liệu thông qua cổng USB CDC (Communication Device Class) High-Speed.
- Delay: Task sẽ chờ 200ms trước khi lặp lại.

3.2. Hiển thị hoạt ảnh bóng, tính điểm và vận hành trò chơi

```
// Display ball animation, scoring and operating the game
void StartTask02(void *argument)
{
  /* USER CODE BEGIN StartTask02 */
  /* Infinite loop */
  BSP_LCD_FillCircle(xPos, yPos, radius);
  float getBuf[4];
  float temp;

  for(;;)
  {
  lastScore = realScore;
    if(state == PLAYING_STATE && hitFlag == HIT)
    {
      for(int i=0; i<4; i++)
```

```
{
  osMessageQueueGet(myQueue01Handle, &getBuf[i], 0, 0);
}
if(getBuf[3] > MAX_HIGH)
{
  temp = MAX_HIGH;
}
else
{
  temp = getBuf[3];
}
// Decide direction of the ball
if(getBuf[0] < DOWN_TILT) upDown = DOWN;
else if(getBuf[0] > UP_TILT) upDown = UP;
else upDown = NONE;
if(getBuf[1] < RIGHT_TILT) leftRight = RIGHT;
else if(getBuf[1] > LEFT_TILT) leftRight = LEFT;
else leftRight = NONE;
//State: ball go up
for (int i = 0; i <= temp; i = i + ACCEL_VALUE)
{
  adjustBallDirection();
  BSP_LCD_FillCircle(xPos, yPos, radius);
  radius = radius + RADIUS_ADJUST_VALUE;
  if (radius > MAX_RADIUS)
  {
    break;
  }
osDelay(DELAY_TIME);
}
//State: ball go down
for (int i = temp; i >= 0; i = i - ACCEL_VALUE)
{
  BSP_LCD_Clear(LCD_COLOR_BLUE);
  adjustBallDirection();
BSP_LCD_FillCircle(xPos, yPos, radius);
radius = radius - RADIUS_ADJUST_VALUE;
if (radius >= MIN_RADIUS && radius <= NORMAL_RADIUS)
{
  if (hitFlag == HIT)
```

```
            {
              realScore++;
              break;
             }
            }
          else if (radius < MIN_RADIUS)
            {
              break;
            }
            osDelay(DELAY_TIME);
          }
          // Check game over
          if (realScore == lastScore)
          {
            state = GAMEOVER_STATE;
          }
        }
      }
      /* USER CODE END StartTask02 */
    }
```

- Giải thích:
  - Đầu tiên kiểm tra xem người dùng có đang chơi hay không. Nếu là đang chơi thì mới kiểm tra cờ "hitFlag" (được cập nhật bởi Task01).
  - Nếu trạng thái "hitFlag" là "HIT" thì lấy dữ liệu từ trong queue tên là **myQueue01Handle** gồm giá trị gia tốc 3 trục x, y, z và một giá trị độ cao lần lượt ghi vào mảng **getBuf**[0-3].
  - Kế tiếp so sánh giá trị độ cao được tính (getBuf[3]) với giá trị độ cao tối đa mà bóng có thể đạt được.
  - Tiếp theo là dựa vào giá trị đọc được từ Gyroscope để quyết định xem người chơi đang đánh bóng theo hướng nào. Các hướng được truyền cho hàm **adjustBallDirection** thông qua các biến **upDown**, **leftRight** để tính ra vị trí tâm mới cho quả bóng.
  - Hai vòng for kế tiếp một vòng là để thể hiện quả bóng đang bay lên và vòng còn lại là để thể hiện quả bóng đang thu nhỏ. Vòng for thứ hai có thêm một điều kiện so sánh xem người cho có đánh bóng đúng thời điểm quả bóng rơi gần đến mặt vợt hay không. Nếu có thì thực hiện cập nhật số điểm tăng lên một đơn vị.
  - Cuối cùng là check xem nếu qua một lần đánh mà số điểm không tăng (có thể đánh trượt hay không đánh) thì chuyển trạng thái trò chơi thành Game Over.

```
void adjustBallDirection()
{
  if(upDown == DOWN) yPos++;
  if(upDown == UP) yPos--;
  if(leftRight == RIGHT) xPos++;
  if(leftRight == LEFT) xPos--;
  if((xPos + radius) > MAX_XPOS)
  {
    leftRight = !leftRight;
    xPos-=POS_ADJUST_VALUE;
  }
  if((xPos - radius) < MIN_XPOS)
  {
    leftRight = !leftRight;
    xPos+=POS_ADJUST_VALUE;
  }
  if((yPos + radius) > MAX_YPOS)
  {
    upDown = !upDown;
    yPos-=POS_ADJUST_VALUE;
  }
  if((yPos - radius) < MIN_YPOS)
  {
    upDown = !upDown;
    yPos+=POS_ADJUST_VALUE;
  }
};
```

- Giải thích:
  - Điều chỉnh hướng bay của bóng dựa trên hướng nghiêng của board khi thực hiện động tác đánh bóng
  - Điều chỉnh bóng phản hồi lại khi chạm vào các cạnh viền của màn hình

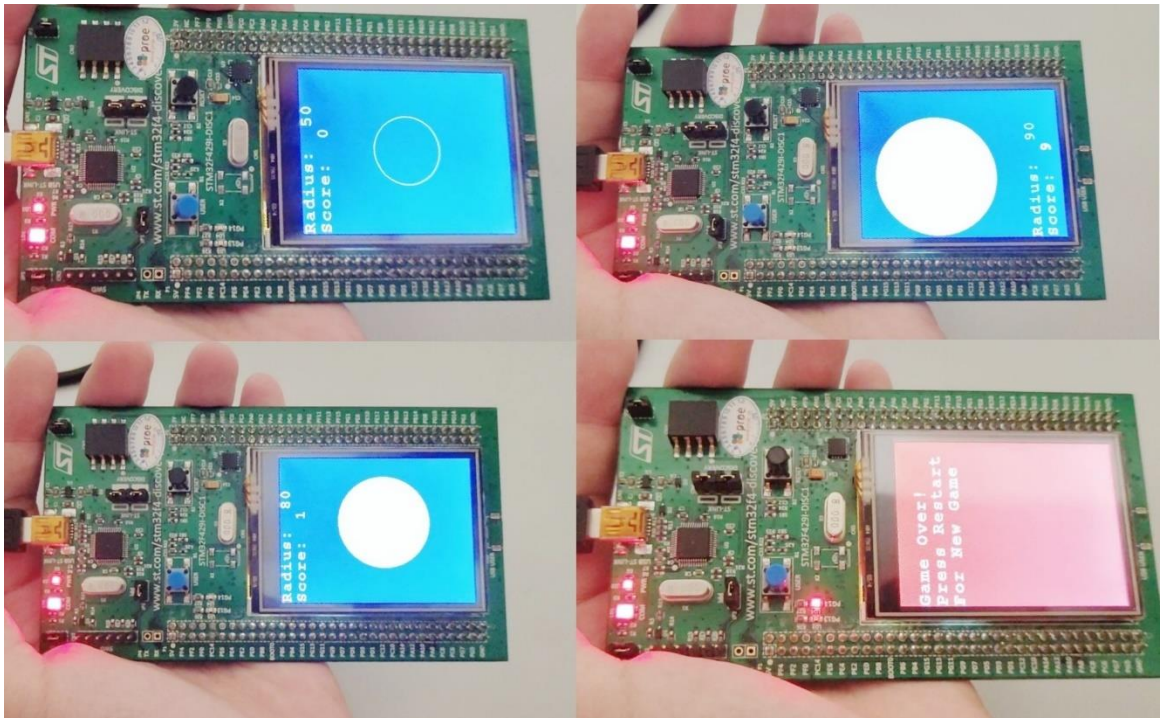
3.3. Hiển thị bán kính, điểm số và màn hình game over

```
//Display radius and score on LCD
void StartTask03(void *argument)
{
  /* USER CODE BEGIN StartTask03 */
  /* Infinite loop */
  for(;;)
  {
   if(state == PLAYING_STATE)
     {
     // Change position of game information
     if((yPos - radius) < TRANSITION_BOUNDARY)
     {
       if(isClearHead == NO)
       {
         BSP_LCD_Clear(LCD_COLOR_BLUE);
         isClearHead = YES;
         isClearFoot = NO;
       }
       sprintf(buffer2, "Radius:  %d", radius);
       BSP_LCD_DisplayStringAtLine(11, buffer2);
       sprintf(buffer2, "Score:  %d", realScore);
       BSP_LCD_DisplayStringAtLine(12, buffer2);
     }
     else
     {
       if(isClearFoot == NO)
      {
         BSP_LCD_Clear(LCD_COLOR_BLUE);
         isClearFoot = YES;
         isClearHead = NO;
      }
       sprintf(buffer2, "Radius:  %d", radius);
       BSP_LCD_DisplayStringAtLine(0, buffer2);
       sprintf(buffer2, "Score:  %d", realScore);
       BSP_LCD_DisplayStringAtLine(1, buffer2);
     }
     }
   // Display "Game over" scene
   else
```

```
    {
      HAL_GPIO_WritePin(LED_GPIO_Port, LED_Pin, GPIO_PIN_SET);
      BSP_LCD_Clear(LCD_COLOR_RED);
      BSP_LCD_SetBackColor(LCD_COLOR_RED);
      BSP_LCD_DisplayStringAtLine(1, "Game Over!");
      BSP_LCD_DisplayStringAtLine(2, "Press Restart");
      BSP_LCD_DisplayStringAtLine(3, "For New Game");
      osDelay(1000);
    }
   }
  /* USER CODE END StartTask03 */
}
```

- Giải thích:
  - Kiểm tra trạng thái là đang chơi hay đã thua để hiển thị.
  - Nếu đang chơi thì kiểm tra xem bóng có chạm vào khu vực hiển thị phía trên không, nếu chạm thì chuyển khu vực hiển thị điểm và bán kính xuống góc dưới màn hình.
  - Trong mỗi hàm hiển thị điểm số có 1 hàm dùng để xóa màn hình để không xuất hiện 2 vùng hiển thị điểm cùng lúc.

4. Kết quả thu được



Hình 11. Các trạng thái trong game bóng bàn

Video demo: https://youtu.be/uHYtmUc5CCc

CHƯƠNG 4: ĐÁNH GIÁ

1. Nhận xét – Đánh giá kết quả

Nhóm đã hoàn hiện sản phẩm theo yêu cầu đặt ra của đồ án. Sản phẩm hoạt động ổn định, đạt chất lượng như kì vọng.

Các phần đã thực hiện bao gồm:

- Sử dụng được hệ điều hành FreeRTOS nhằm nâng cao hiệu suất hệ thống.
- Đọc Gyroscope nhận biết chính xác thao tác tâng bóng của người chơi và truyền ra Virtual COM Port có sẵn trên KIT.
- Mô phỏng thành công chuyển động của quả bóng lên màn hình LCD và thêm các giới hạn để quả bóng không bay tràn ra màn hình gây khó chịu cho người chơi.
- Có thực hiện tính điểm khi người chơi tâng bóng trong thời điểm thích hợp và xét thua khi người chơi tâng trượt bóng.
- Bóng có thể bay theo các hướng khi đánh bóng nghiêng vợt và có thể tâng ngược lại khi va phải các viền màn hình.
- Sử dụng thuật toán để điều chỉnh được vị trí hiển thị thông tin nhằm không ghi đè lên quả bóng.

Các phần cần cải thiện thêm và định hướng:

- Cải thiện độ mượt khi quả bóng rơi vì nhóm hiện tại đang sử dụng lệch xóa toàn bộ màn hình để mô phỏng bóng rơi nên tạo ra cảm giác khó chịu cho người chơi.
- Thêm các chế độ (dễ, trung bình, khó).
- Thêm phần lưu trữ dữ liệu để cho nhiều người có thể cùng chơi đồng thời tạo bảng xếp hạng cho nhiều người chơi.