

Lập trình event-driven và trực quan cho trò chơi bắn cung: Hướng dẫn từ cơ bản đến nâng cao

I. Giới thiệu - ARCHERY GAME

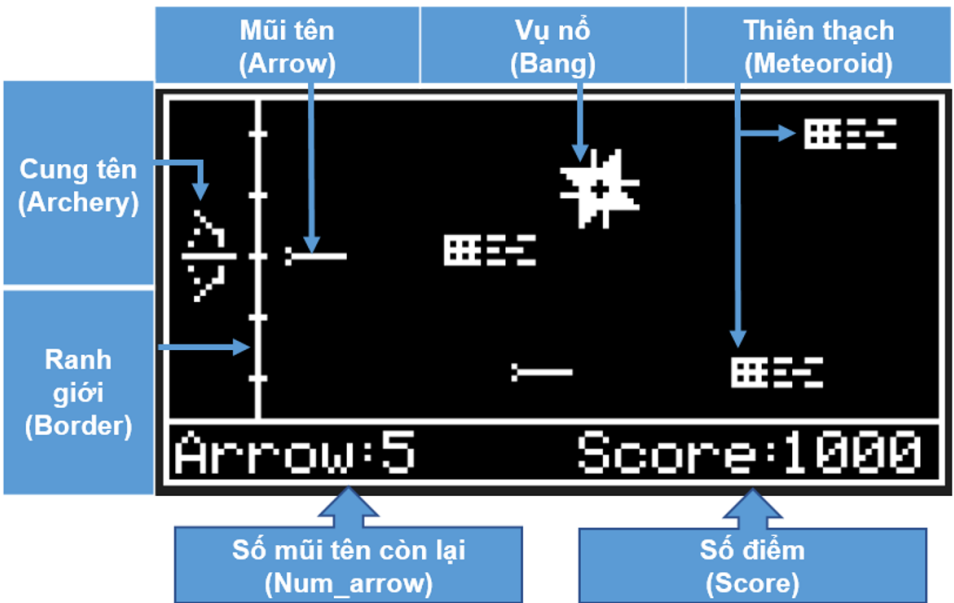
1.1 Mô tả trò chơi và đối tượng

Phần mô tả sau đây về “Archery game” phục vụ mục đích là giải thích cách chơi và các đặc tính kỹ thuật nhằm mục đích thiết kế và triển khai phần mềm và phát triển các dự án game ở phần sau chương trình này.



Trò chơi bắt đầu với màn hình **Menu game** với nhiều chọn lựa:

- **Archery Game:** chọn vào để bắt đầu chơi game
- **Setting:** chọn vào để cài đặt các thông số của game
- **Charts:** chọn vào để xem top 3 điểm cao nhất đạt được
- **Exit:** vào màn hình nghỉ



Các đối tượng trong game:

- **Archey:** Cung tên - nơi bắn ra mũi tên
- **Arrow:** Mũi tên - bắn ra từ cung tên để phá hủy thiên thạch
- **Bang:** Vụ nổ - xuất hiện khi thiên thạch bị phá hủy
- **Border:** Ranh giới - vùng an toàn phải bảo vệ không cho thiên thạch rơi vào
- **Meteoroid:** Thiên thạch - đối tượng bay về phía vùng an toàn có thể phá hủy bởi mũi tên

Cách chơi game:

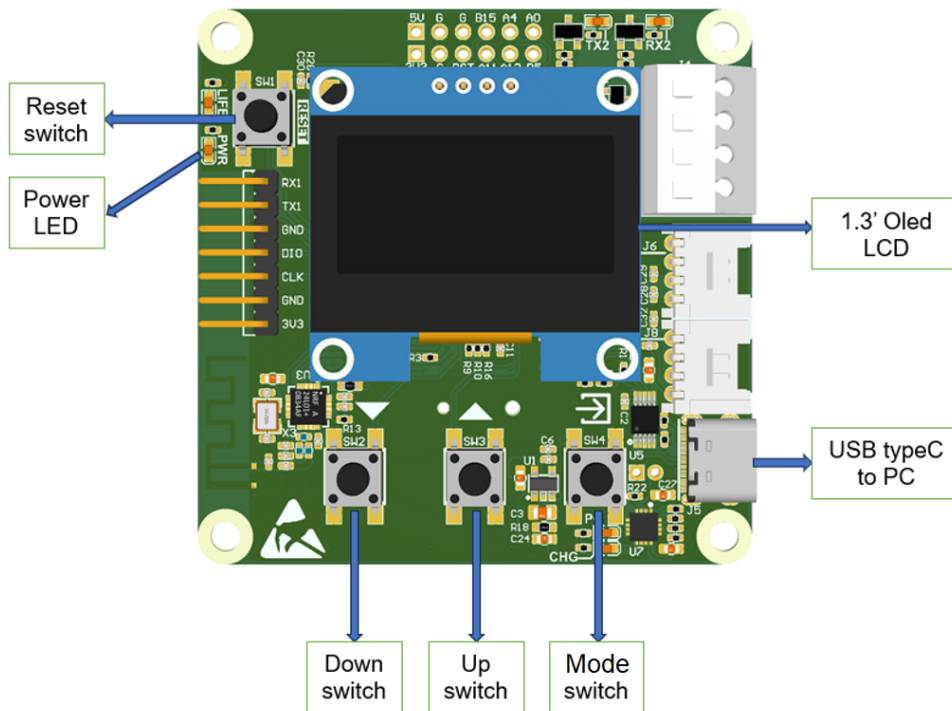
- Trong game này bạn sẽ điều khiển 1 cung tên (Archery) di chuyển lên xuống chọn vị trí và bắn ra mũi tên (Arrow) tiêu diệt các đối tượng là các thiên thạch (Meteoroid) đang bay đến. Với số lượng mũi tên giới hạn, mỗi lần bắn tên số mũi tên sẽ giảm xuống và không bắn được nếu số mũi tên bằng "0" được hiển thị ở bên dưới góc trái màn hình hiển thị.
- Với mỗi lần mũi tên (Arrow) bắn trúng thiên thạch (Meteoroid) thì thiên thạch sẽ tạo ra vụ nổ (Bang) và được +10 điểm vào "score" ở bên dưới góc phải màn hình. Và số lượng mũi tên cũng được khôi phục + 1 mũi tên. Ngoài ra khi mũi tên đi hết màn hình thì mũi tên cũng được xóa và khôi phục +1 mũi tên.
- Với mỗi 200 điểm kiếm được thì tốc độ của thiên thạch (Meteoroid) sẽ tăng dần lên tạo độ khó cho game.



Game sẽ kết thúc khi thiên thạch (Meteoroid) chạm phải ranh giới (Border). Số điểm sẽ được lưu và bạn sẽ vào màn hình "Game Over" bạn có 3 lựa chọn là:

- **Restart:** chơi lại.
- **Charts:** vào xem bảng xếp hạng.
- **Home:** về lại menu game.

1.2 Phần cứng



AK Embedded Base Kit là một evaluation kit dành cho các bạn học software embedded nâng cao.

KIT tích hợp LCD Oled 1.3", 3 nút nhấn, và 1 loa Buzzer phát nhạc, với các trang bị này thì đã đủ để học hệ thống event-driven thông qua thực hành thiết kế máy chơi game.

KIT cũng tích hợp RS485, NRF24L01+, và Flash lên đến 32MB, thích hợp cho prototype các ứng dụng thực tế trong hệ thống nhúng hay sử dụng như: truyền thông có dây, không dây wireless, các ứng dụng lưu trữ data logger,...

Thao tác trong game:

- Trong game người chơi có thể sử dụng 2 nút **[Up]** và **[Down]** để điều hướng chọn mục hay điều khiển Cung tên (Archery) lên xuống.
- Nút **[Mode]** dùng để chọn vào các mục trong Menu game hay bắn ra các mũi tên lúc chơi game.
- Nút **[Reset]** để khởi động lại kit.

II. Thiết kế - ARCHERY GAME

Các khái niệm trong event-driven

Sự kiện (Event): Sự kiện đại diện cho một hành động xảy ra trong hệ thống. Trong trò chơi, các sự kiện có thể là các hành động của người chơi như nhấn nút, timer, cập nhật trạng thái cho đối tượng.

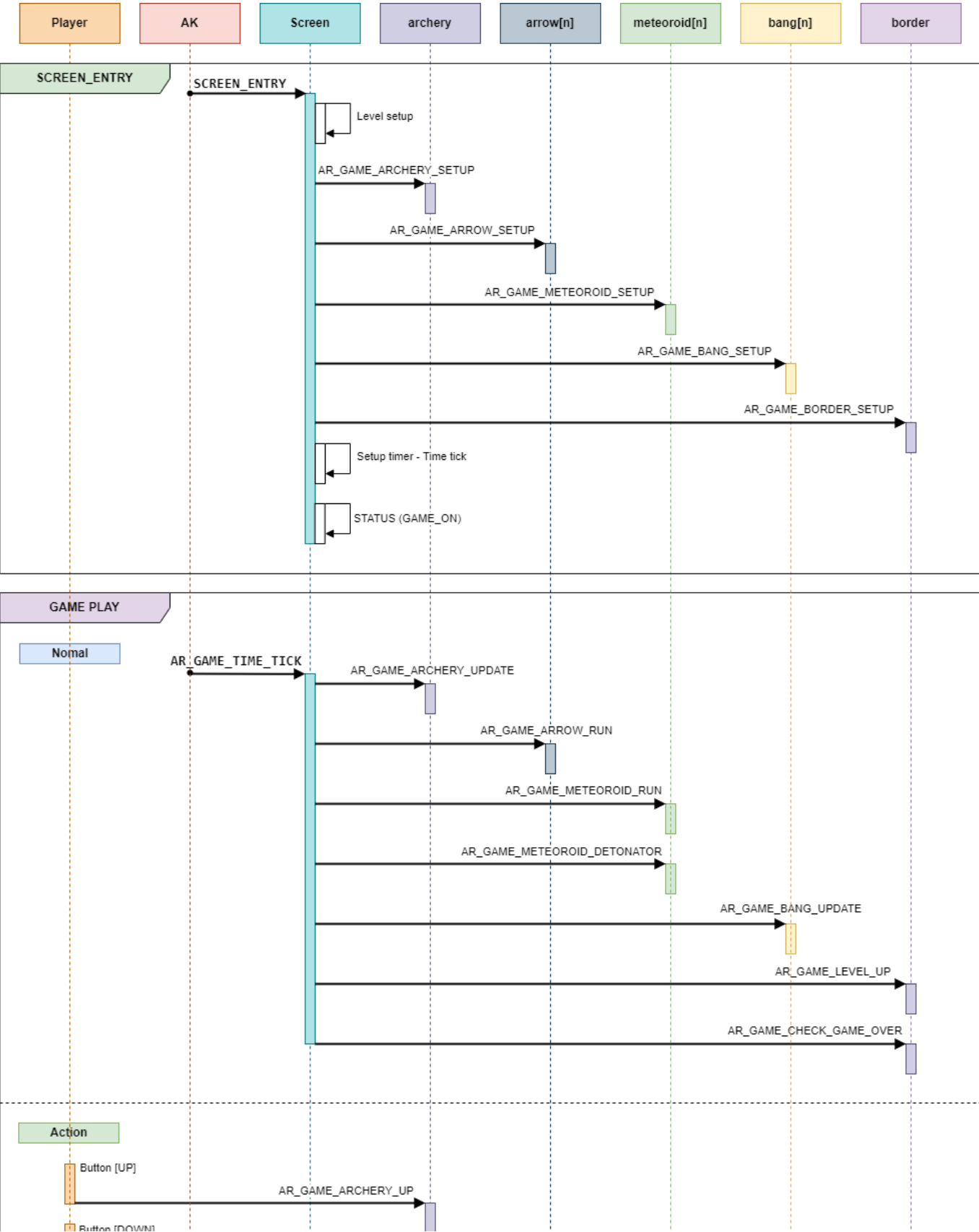
Xử lý sự kiện (Event Handling): Xử lý sự kiện là quá trình xác định cách chương trình phản ứng khi một sự kiện xảy ra. Điều này bao gồm việc cung cấp mã xử lý (event handler) để thực hiện các hành động tương ứng với sự kiện. Mã xử lý được gắn kết với sự kiện và sẽ được gọi tự động khi sự kiện xảy ra.

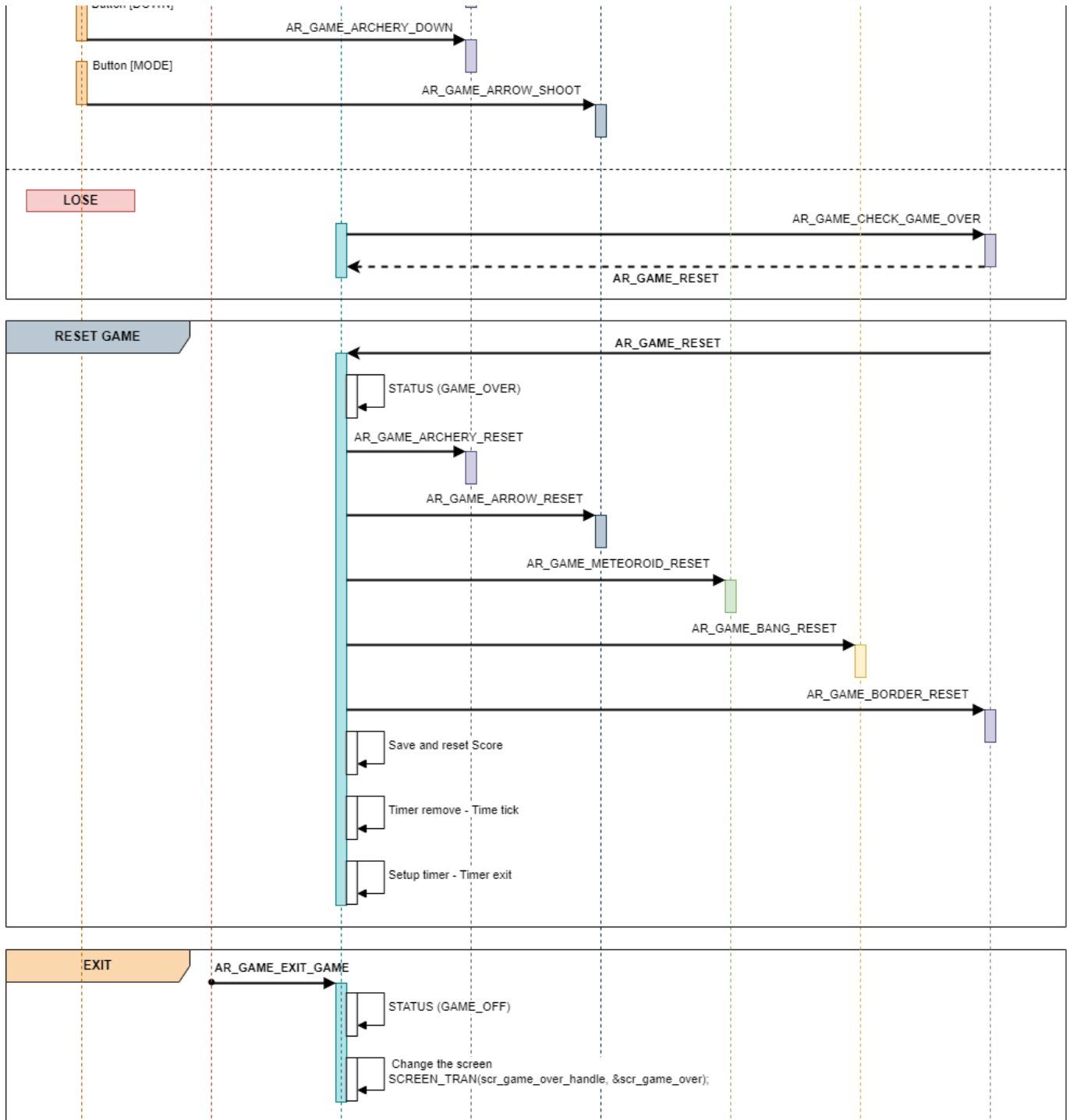
Signal: Signal là một cơ chế truyền thông tin giữa các thành phần trong hệ thống event-driven. Khi một sự kiện xảy ra, nó có thể gửi một tín hiệu (signal) để thông báo cho các thành phần khác về việc xảy ra của sự kiện đó.

Task: Task đại diện cho một tác vụ hoặc một công việc cần được thực hiện trong hệ thống. Trong lập trình event-driven, mỗi sự kiện thường gắn kết với một task tương ứng. Khi một sự kiện xảy ra, hệ thống tạo ra một task tương ứng và giao nó cho bộ xử lý để thực hiện.

2.1 Sơ đồ trình tự (The sequence diagram)

Sơ đồ trình tự được sử dụng để mô tả và hiển thị trình tự của các thông điệp và tương tác giữa các đối tượng trong một hệ thống.





Ghi chú:

SCREEN_ENTRY: Cài đặt các thiết lập ban đầu cho đối tượng trong game.

- **Level setup:** Thiết lập thông số cấp độ cho game.
- **AR_GAME_ARCHERY_SETUP:** Thiết lập thông số ban đầu cho đối tượng Cung tên (Archery)
- **AR_GAME_ARROW_SETUP:** Thiết lập thông số ban đầu cho các đối tượng Mũi tên (Arrow)
- **AR_GAME_METEOROID_SETUP:** Thiết lập thông số ban đầu cho các đối tượng Thiên thạch (Meteoroid)
- **AR_GAME_BANG_SETUP:** Thiết lập thông số ban đầu cho các đối tượng Vụ nổ (Bang)
- **AR_GAME_BORDER_SETUP:** Thiết lập thông số ban đầu cho đối tượng Ranh giới (Border)
- **Setup timer - Time tick:** Khởi tạo Timer - Time tick cho game.
- **STATUS (GAME_ON):** Cập nhật trạng thái game -> GAME_ON

GAME PLAY: Quá trình hoạt động của game.

GAME PLAY - Nomal: Game hoạt động ở trạng thái bình thường.

- **AR_GAME_TIME_TICK:** Signal do Timer - Time tick gửi đến.
- **AR_GAME_ARCHERY_UPDATE:** Cập nhật trạng thái Cung tên.
- **AR_GAME_ARROW_RUN:** Cập nhật di chuyển của các Mũi tên theo thời gian.
- **AR_GAME_METEOROID_RUN:** Cập nhật di chuyển của các Thiên thạch theo thời gian.
- **AR_GAME_METEOROID_DETONATOR:** Kiểm tra các Thiên thạch có bị Mũi tên phá hủy.
- **AR_GAME_BANG_UPDATE:** Cập nhật hoạt ảnh Vụ nổ theo thời gian
- **AR_GAME_BORDER_UPDATE:** Kiểm tra số điểm hiện tại để cập nhật tăng độ khó game.
- **AR_GAME_CHECK_GAME_OVER:** Kiểm tra Thiên thạch chạm vào Ranh giới nếu chạm vào thì gửi Signal - **AR_GAME_RESET** đến **Screen**

GAME PLAY - Action: Game hoạt động ở trạng thái có tác động của các nút nhấn.

- **AR_GAME_ARCHERY_UP:** Player nhấn nút **[Up]** điều khiển Cung tên di chuyển lên.
- **AR_GAME_ARCHERY_DOWN:** Player nhấn nút **[Down]** điều khiển Cung tên di chuyển xuống.
- **AR_GAME_ARROW_SHOOT:** Player nhấn nút **[Mode]** điều khiển bắn mũi tên ra.

GAME PLAY - LOSE: Game hoạt động lúc phát hiện Thiên thạch chạm vào Ranh giới.

- **AR_GAME_CHECK_GAME_OVER:** Kiểm tra thấy có Thiên thạch chạm vào Ranh giới. Gửi Signal - **AR_GAME_RESET** đến **Screen**

RESET GAME: Quá trình cài đặt lại các thông số trước khi thoát game.

- **STATUS (GAME_OVER):** Cập nhật trạng thái game -> **GAME_OVER**
- **AR_GAME_RESET:** Signal cài đặt lại game do Border gửi đến.
- **AR_GAME_ARCHERY_RESET:** Cài đặt lại đối tượng Cung tên trước khi thoát.
- **AR_GAME_ARROW_RESET:** Cài đặt lại đối tượng Mũi tên trước khi thoát.
- **AR_GAME_METEOROID_RESET:** Cài đặt lại đối tượng Thiên thạch trước khi thoát.
- **AR_GAME_BANG_RESET:** Cài đặt lại đối tượng Vụ nổ trước khi thoát.
- **AR_GAME_BORDER_RESET:** Cài đặt lại đối tượng Ranh giới trước khi thoát.
- **Save and reset Score:** Lưu số điểm hiện tại và Cài đặt lại.
- **Timer remove - Timer tick:** Xóa Timer - Time tick
- **Setup timer - Timer exit:** Tạo 1 timer one shot để thoát game.

EXIT: Thoát khỏi game và chuyển sang màn hình khác.

- **AR_GAME_EXIT:** Signal do Timer exit gửi đến.
- **STATUS (GAME_OFF):** Cập nhật trạng thái game -> **GAME_OFF**
- **Change the screen:** Chuyển màn hình

2.2 Quản lý tài nguyên đối tượng

Sau khi xác định được các đối tượng trong game mà chúng ta cần, tiếp theo chúng ta phải liệt kê ra các thuộc tính, các task, các signal và bitmap mà trong game sẽ sử dụng tới. Việc liệt kê càng chi tiết thì việc làm game diễn ra càng nhanh và tạo tình rõ ràng minh bạch cho phần tài nguyên giúp phần code game diễn ra xuống sẽ hơn.

2.2.1 Quản lý Struct

Việc liệt kê các thuộc tính của đối tượng trong game có các tác dụng quan trọng sau:

- **Định rõ thông tin:** Liệt kê các thuộc tính giúp xác định rõ thông tin về đối tượng trong game.
- **Thiết kế cấu trúc dữ liệu:** Liệt kê thuộc tính giúp xác định cấu trúc dữ liệu phù hợp để lưu trữ thông tin của đối tượng.
- **Giảm rủi ro và lỗi:** Khi bạn xác định trước các thuộc tính cần thiết, bạn giảm thiểu khả năng bỏ sót hoặc nhầm lẫn trong việc xử lý và sử dụng các thuộc tính.

Thuộc tính:

- **visible:** Thuộc tính quy định hiển thị
- **x, y:** Thuộc tính tọa độ
- **action_image:** Thuộc tính quy định hoạt ảnh

Áp dụng struct trên cho các đối tượng:

- `ar_game_archery_t` archery
- `ar_game_arrow_t` arrow
- `ar_game_bang_t` bang
- `ar_game_border_t` border
- `ar_game_meteoroid_t` meteoroid

Các biến quan trọng:

- `ar_game_score`: Điểm của trò chơi
- `ar_game_status`: Trạng thái quả trò chơi
 - `GAME_OFF`: Tắt
 - `GAME_ON`: Bật
 - `GAME_OVER`: Đã thua
- `ar_game_setting_t` settingsetup : cấp độ trò chơi
 - `settingsetup.silent` : Quy định âm thanh ON/OFF
 - `settingsetup.num_arrow` : Số lượng mũi tên
 - `settingsetup.arrow_speed` : Tốc độ mũi tên
 - `settingsetup.meteoroid_speed` : Tốc độ của thiên thạch

2.2.2 Quản lý task

Trong lập trình sự kiện (event-driven programming), một task là một đơn vị công việc độc lập được thực thi khi xảy ra một sự kiện cụ thể. Tác dụng của task là xử lý và đáp ứng cho các sự kiện trong hệ thống. Một số tác dụng quan trọng của task code:

- **Xử lý sự kiện:** Task được sử dụng để xử lý các sự kiện khi chúng xảy ra. Mỗi task có thể được liên kết với một sự kiện cụ thể và thực thi một loạt các hành động khi sự kiện đó xảy ra.
- **Đồng bộ hóa:** Task cung cấp cơ chế đồng bộ hóa cho việc xử lý các sự kiện. Khi một sự kiện xảy ra, task tương ứng được kích hoạt và thực thi. Các task khác có thể đợi cho đến khi task hiện tại hoàn

thành trước khi được kích hoạt. Điều này giúp đảm bảo rằng các hành động xử lý sự kiện được thực hiện theo một thứ tự nhất định và tránh xung đột.

- **Quản lý luồng điều khiển:** Task cho phép quản lý luồng điều khiển trong ứng dụng event-driven. Bằng cách sử dụng task, bạn có thể xác định thứ tự thực thi của các hành động khi xảy ra các sự kiện khác nhau.
- **Tách biệt logic:** Sử dụng task giúp tách biệt logic xử lý sự kiện ra khỏi logic chính của ứng dụng. Điều này giúp tăng tính sạch sẽ, dễ đọc và dễ bảo trì của mã nguồn.
- **Phân cấp nhiệm vụ:** Task level cho phép việc sắp xếp trình tự ưu tiên xử lý của task.

Tasks ID	Task level	App tasks
AR_GAME_ARCHERY_ID	TASK_PRI_LEVEL_4	ar_game_archery_handle
AR_GAME_ARROW_ID	TASK_PRI_LEVEL_4	ar_game_arrow_handle
AR_GAME_BANG_ID	TASK_PRI_LEVEL_4	ar_game_bang_handle
AR_GAME_BORDER_ID	TASK_PRI_LEVEL_4	ar_game_border_handle
AR_GAME_METEOROID_ID	TASK_PRI_LEVEL_4	ar_game_meteoroid_handle
AR_GAME_SCREEN_ID	TASK_PRI_LEVEL_4	scr_archer_game_handle

2.2.3 Quản lý signal

Signal là một công cụ để thông báo về sự kiện và gọi hàm xử lý tương ứng trong hệ thống. Nó được sử dụng để gửi thông điệp và điều khiển luồng xử lý trong các ứng dụng và hệ thống phức tạp. Trong game này Signal có các tác dụng như:

- **Xử lý sự kiện:** Signal được sử dụng để xử lý các sự kiện trong game.
- **Giao tiếp giữa các thành phần:** Signal cho phép giao tiếp giữa các thành phần trong game.
- **Bất đồng bộ và xử lý đa luồng:** Signal cung cấp khả năng xử lý bất đồng bộ và đa luồng trong game. Khi một tín hiệu được phát ra, các hàm xử lý có thể được gọi đồng thời hoặc song song trong các luồng khác nhau.

Đối tượng	Task ID	Signal
Archery	AR_GAME_ARCHERY_ID	AR_GAME_ARCHERY_SETUP
		AR_GAME_ARCHERY_UPDATE
		AR_GAME_ARCHERY_UP
		AR_GAME_ARCHERY_DOWN
		AR_GAME_ARCHERY_RESET
Arrow	AR_GAME_ARROW_ID	AR_GAME_ARROW_SETUP
		AR_GAME_ARROW_RUN
		AR_GAME_ARROW_SHOOT
		AR_GAME_ARROW_RESET
Bang	AR_GAME_BANG_ID	AR_GAME_BANG_SETUP
		AR_GAME_BANG_UPDATE
		AR_GAME_BANG_RESET
Border	AR_GAME_BORDER_ID	AR_GAME_BORDER_SETUP
		AR_GAME_BORDER_UPDATE
		AR_GAME_BORDER_RESET
		AR_GAME_CHECK_GAME_OVER
Meteoroid	AR_GAME_METEOROID_ID	AR_GAME_METEOROID_SETUP
		AR_GAME_METEOROID_RUN
		AR_GAME_METEOROID_DETONATOR
		AR_GAME_METEOROID_RESET
Screen	AR_GAME_SCREEN_ID	AR_GAME_INITIAL_SETUP
		AR_GAME_TIME_TICK
		AR_GAME_RESET
		AR_GAME_EXIT_GAME










- **Setup:** Signal khởi tạo thông số ban đầu cho các đối tượng
- **Update/Run:** Signal cập nhật theo thời gian Time tick
- **Reset:** Signal khởi tạo lại các giá trị hiện tại trước khi thoát game
- **Exit:** Signal thoát game

2.2.4 Quản lý hiển thị đối tượng

Bitmap là một cấu trúc dữ liệu được sử dụng để lưu trữ và hiển thị hình ảnh trong game.

Animation là ứng dụng việc nối ảnh của của nhiều ảnh liên tiếp tạo thành hoạt ảnh cho đối tượng muốn miêu tả. Trong game này tôi dùng biến "action_image" trong đối tượng để thay đổi hoạt ảnh thành

animation.

Tên đối tượng		Hình ảnh	Mô tả
Archery (15x15 px)	bitmap_archery_I []		Cung lúc có tên
	bitmap_archery_II []		Cung lúc hết tên
Arrow (10x5px)	bitmap_arrow []		Mũi tên
Bang (15x15 px) (10x10 px)	bitmap_bang_I []		Vụ nổ animation 1
	bitmap_bang_II []		Vụ nổ animation 2
	bitmap_bang_III []		Vụ nổ animation 3
Meteoroid (20x10 px)	bitmap_meteoroid_I []		Thiên thạch animation 1
	bitmap_meteoroid_II []		Thiên thạch animation 2
	bitmap_meteoroid_III []		Thiên thạch animation 3

III. Lập trình xử lý - OBJECT

3.1 Cung tên (Archery)

Mô tả: Cung tên là đối tượng cung tên có thể di chuyển lên, xuống trong trò chơi được điều khiển bởi 2 nút **[Up]**, **[Down]** trên Kit

Thuộc tính:

- Vị trí tạo độ: x (*uint32_t*), y (*uint32_t*)
- Hiển thị: visible (*bool*)
- Animation: action_image (*uint8_t*)

Task:

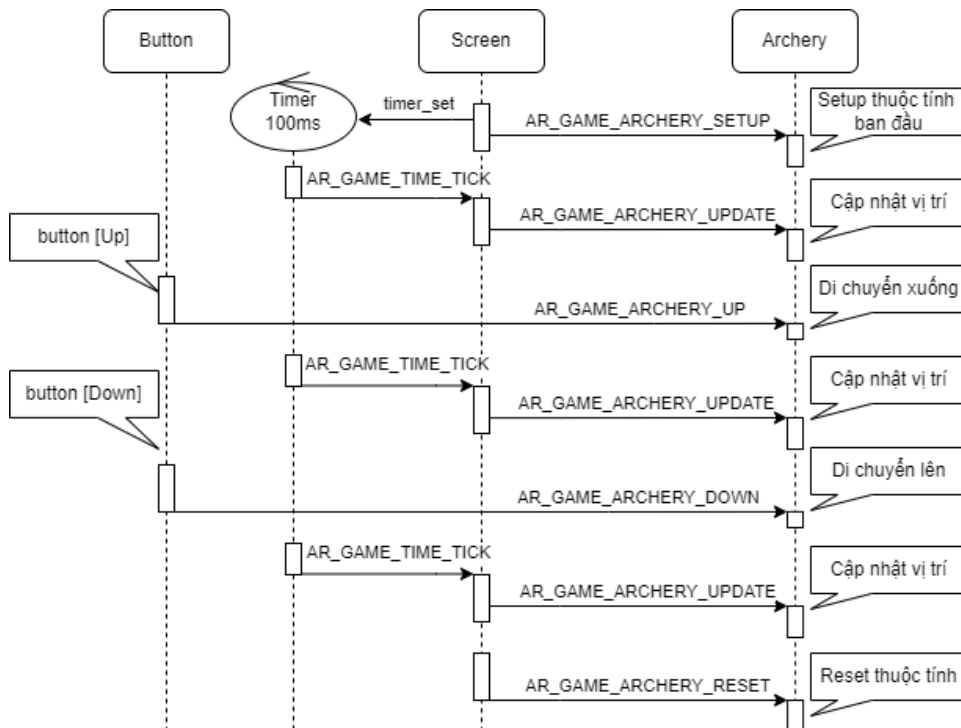
- Task ID: AR_GAME_ARCHERY_ID
- Task level: TASK_PRI_LEVEL_4
- App task: ar_game_archery_handle

Signal:

- AR_GAME_ARCHERY_SETUP - Cài đặt thuộc tính ban đầu
- AR_GAME_ARCHERY_UPDATE - Cập nhật thuộc tính theo **Timer**
- AR_GAME_ARCHERY_UP - Nhận tín hiệu nút nhấn **[Up]**

- AR_GAME_ARCHERY_DOWN - Nhận tín hiệu nút nhấn **[Down]**
- AR_GAME_ARCHERY_RESET - Cài đặt lại thuộc tính trước khi **End Game**

Sequence diagram:



Code:

► Details

```

#include "ar_game_archery.h"

ar_game_archery_t archery;
static uint32_t archery_y = AXIS_Y_ARCHERY;

#define AR_GAME_ARCHERY_SETUP() \
do { \
    archery.x = AXIS_X_ARCHERY; \
    archery.y = AXIS_Y_ARCHERY; \
    archery.visible = WHITE; \
    archery.action_image = 1; \
} while (0);

#define AR_GAME_ARCHERY_UP() \
do { \
    archery_y -= STEP_ARCHERY_AXIS_Y; \
    if (archery_y == 0) {archery_y = 10;} \
} while(0);

#define AR_GAME_ARCHERY_DOWN() \
do { \
    archery_y += STEP_ARCHERY_AXIS_Y; \
    if (archery_y > 50) {archery_y = 50;} \
} while(0);
  
```

```
#define AR_GAME_ARCHERY_RESET() \
do { \
    archery.x = AXIS_X_ARCHERY; \
    archery.y = AXIS_Y_ARCHERY; \
    archery.visible = BLACK; \
    archery_y = AXIS_Y_ARCHERY; \
} while(0);

void ar_game_archery_handle(ak_msg_t* msg) {
    switch (msg->sig) {
        case AR_GAME_ARCHERY_SETUP: {
            APP_DBG_SIG("AR_GAME_ARCHERY_SETUP\n");
            AR_GAME_ARCHERY_SETUP();
        }
            break;

        case AR_GAME_ARCHERY_UPDATE: {
            APP_DBG_SIG("AR_GAME_ARCHERY_UPDATE\n");
            archery.y = archery_y;
        }
            break;

        case AR_GAME_ARCHERY_UP: {
            APP_DBG_SIG("AR_GAME_ARCHERY_UP\n");
            AR_GAME_ARCHERY_UP();
        }
            break;

        case AR_GAME_ARCHERY_DOWN: {
            APP_DBG_SIG("AR_GAME_ARCHERY_DOWN\n");
            AR_GAME_ARCHERY_DOWN();
        }
            break;

        case AR_GAME_ARCHERY_RESET: {
            APP_DBG_SIG("AR_GAME_ARCHERY_RESET\n");
            AR_GAME_ARCHERY_RESET();
        }
            break;

        default:
            break;
    }
}
```

3.2 Mũi tên (Arrow)

Mô tả: Mũi tên là đối tượng bắn ra từ cung tên, từ lúc bắn ra sẽ liên tục di chuyển từ trái sang phải màn hình. Mũi tên có thể phá hủy thiên thạch trên đường đi. Số lượng mũi tên hiển thị ở góc dưới bên trái màn hình.

Thuộc tính:

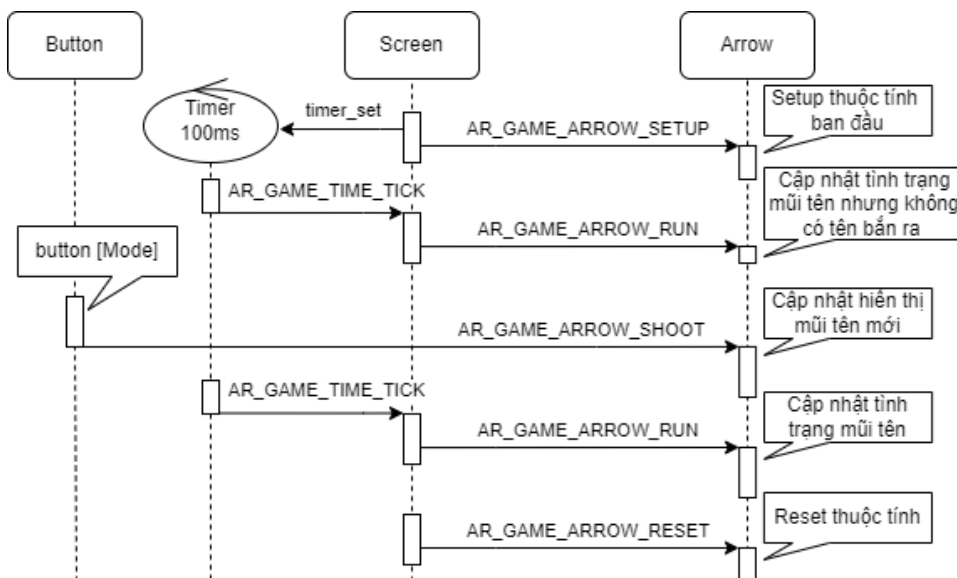
- Vị trí tạo độ: x (*uint32_t*), y (*uint32_t*)
- Hiển thị: visible (*bool*)
- Animation: action_image (*uint8_t*)

Task:

- Task ID: AR_GAME_ARROW_ID
- Task level: TASK_PRI_LEVEL_4
- App task: ar_game_arrow_handle

Signal:

- AR_GAME_ARROW_SETUP - Cài đặt thuộc tính ban đầu
- AR_GAME_ARROW_RUN - Cập nhật thuộc tính theo **Timer**
- AR_GAME_ARROW_SHOOT - Nhận tín hiệu nút nhấn **[Mode]**
- AR_GAME_ARROW_RESET - Cài đặt lại thuộc tính trước khi **End Game**

Sequence diagram:**Code:**

► Details

```

#include "ar_game_arrow.h"

#include "ar_game_archery.h"
#include "scr_archery_game.h"

ar_game_arrow_t arrow[MAX_NUM_ARROW];

#define AR_GAME_ARROW_SETUP() \
do { \
    for (uint8_t i = 0; i < MAX_NUM_ARROW; i++) { \
        arrow[i].x = 0; \
    } \
} while(0)
  
```

```

        arrow[i].y = 0; \
        arrow[i].visible = BLACK; \
        arrow[i].action_image = 1; \
    } \
} while (0);

#define AR_GAME_ARROW_RUN() \
do { \
    for (uint8_t i = 0; i < MAX_NUM_ARROW; i++) { \
        if (arrow[i].visible == WHITE) { \
            arrow[i].x += settingsetup.arrow_speed; \
            if (arrow[i].x == MAX_AXIS_X_ARROW) { \
                arrow[i].visible = BLACK; \
                arrow[i].x = 0; \
                settingsetup.num_arrow++; \
            } \
        } \
    } \
} while(0);

#define AR_GAME_ARROW_SHOOT() \
do { \
    for (uint8_t i = 0; i < MAX_NUM_ARROW; i++) { \
        if (arrow[i].visible == BLACK && settingsetup.num_arrow != 0) { \
            settingsetup.num_arrow--; \
            arrow[i].visible = WHITE; \
            arrow[i].y = archery.y - 5; \
            BUZZER_PlayTones(tones_cc); \
            break; \
        } \
        else if (settingsetup.num_arrow == 0) { \
            BUZZER_PlayTones(tones_3beep); \
            break; \
        } \
    } \
} while(0);

#define AR_GAME_ARROW_RESET() \
do { \
    for (uint8_t i = 0; i < MAX_NUM_ARROW; i++) { \
        arrow[i].x = 0; \
        arrow[i].y = 0; \
        arrow[i].visible = BLACK; \
        arrow[i].action_image = 1; \
    } \
} while (0);

void ar_game_arrow_handle(ak_msg_t* msg) {
    switch (msg->sig) {
        case AR_GAME_ARROW_SETUP: {
            APP_DBG_SIG("AR_GAME_ARROW_SETUP\n");
            AR_GAME_ARROW_SETUP();
        }
        break;
    }
}

```

```

    case AR_GAME_ARROW_RUN: {
        APP_DBG_SIG("AR_GAME_ARROW_RUN\n");
        AR_GAME_ARROW_RUN();
    }
    break;

    case AR_GAME_ARROW_SHOOT: {
        APP_DBG_SIG("AR_GAME_ARROW_SHOOT\n");
        AR_GAME_ARROW_SHOOT();
    }
    break;

    case AR_GAME_ARROW_RESET: {
        APP_DBG_SIG("AR_GAME_ARROW_RESET\n");
        AR_GAME_ARROW_RESET();
    }
    break;

    default:
        break;
}
}

```

3.3 Vụ nổ (Bang)

Mô tả: Vụ nổ là đối tượng xuất hiện sau khi Thiên thạch bị phá hủy bởi Mũi tên. Vụ nổ xuất hiện ở vị trí nơi Thiên thạch bị phá hủy với 3 hoạt ảnh vụ nổ thay đổi theo thời gian đi kèm với âm thanh nổ rồi kết thúc.

Thuộc tính:

- Vị trí tạo độ: x (*uint32_t*), y (*uint32_t*)
- Hiển thị: visible (*bool*)
- Animation: action_image (*uint8_t*)

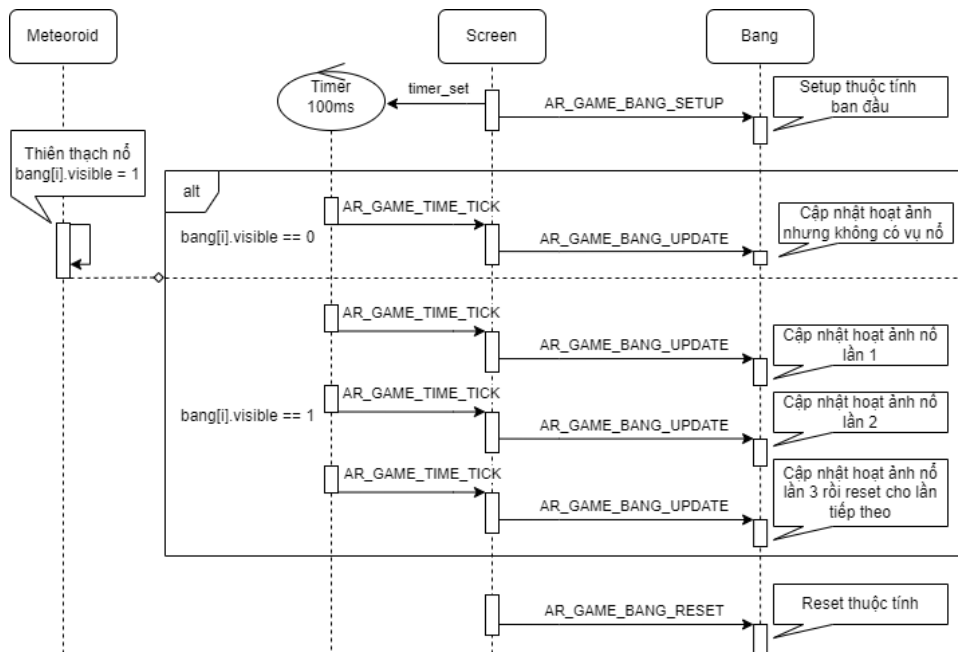
Task:

- Task ID: AR_GAME_BANG_ID
- Task level: TASK_PRI_LEVEL_4
- App task: ar_game_bang_handle

Signal:

- AR_GAME_BANG_SETUP - Cài đặt thuộc tính ban đầu
- AR_GAME_BANG_UPDATE - Cập nhật thuộc tính theo **Timer**
- AR_GAME_BANG_RESET - Cài đặt lại thuộc tính trước khi **End Game**

Sequence diagram:



Code:

► Details

```

#include "ar_game_bang.h"

#include "ar_game_meteoroid.h"
#include "ar_game_arrow.h"

ar_game_bang_t bang[NUM_BANG];

#define AR_GAME_BANG_SETUP() \
do { \
    for (uint8_t i = 0; i < NUM_BANG; i++) { \
        bang[i].x = 0; \
        bang[i].y = 0; \
        bang[i].visible = BLACK; \
        bang[i].action_image = 1; \
    } \
} while (0);

#define AR_GAME_BANG_UPDATE() \
do { \
    for (uint8_t i = 0; i < NUM_BANG; i++) { \
        if (bang[i].visible == WHITE) { \
            bang[i].action_image++; \
        } \
        if (bang[i].action_image == 4) { \
            bang[i].action_image = 1; \
            bang[i].visible = BLACK; \
            meteoroid[i].visible = WHITE; \
        } \
    } \
} while(0);
  
```



```

#define AR_GAME_BANG_RESET() \
do { \
    for (uint8_t i = 0; i < NUM_BANG; i++) { \
        bang[i].visible = BLACK; \
        bang[i].action_image = 1; \
    } \
} while (0);

void ar_game_bang_handle(ak_msg_t* msg) {
    switch (msg->sig) {
        case AR_GAME_BANG_SETUP: {
            APP_DBG_SIG("AR_GAME_BANG_SETUP\n");
            AR_GAME_BANG_SETUP();
        }
        break;

        case AR_GAME_BANG_UPDATE: {
            APP_DBG_SIG("AR_GAME_BANG_UPDATE\n");
            AR_GAME_BANG_UPDATE();
        }
        break;

        case AR_GAME_BANG_RESET: {
            APP_DBG_SIG("AR_GAME_BANG_RESET\n");
            AR_GAME_BANG_RESET();
        }
        break;

        default:
            break;
    }
}

```

3.4 Ranh giới (Border)

Mô tả: Ranh giới là vùng an toàn cần được bảo vệ. Nếu Thiên thạch đi vào trong ranh giới thì sẽ xử lý thua game. Đồng thời trong đối tượng ranh giới sẽ kiểm tra số điểm để cập nhật tăng độ khó cho game.

Thuộc tính:

- Vị trí tạo độ: x (*uint32_t*), y (*uint32_t*)
- Hiển thị: visible (*bool*)
- Animation: action_image (*uint8_t*)

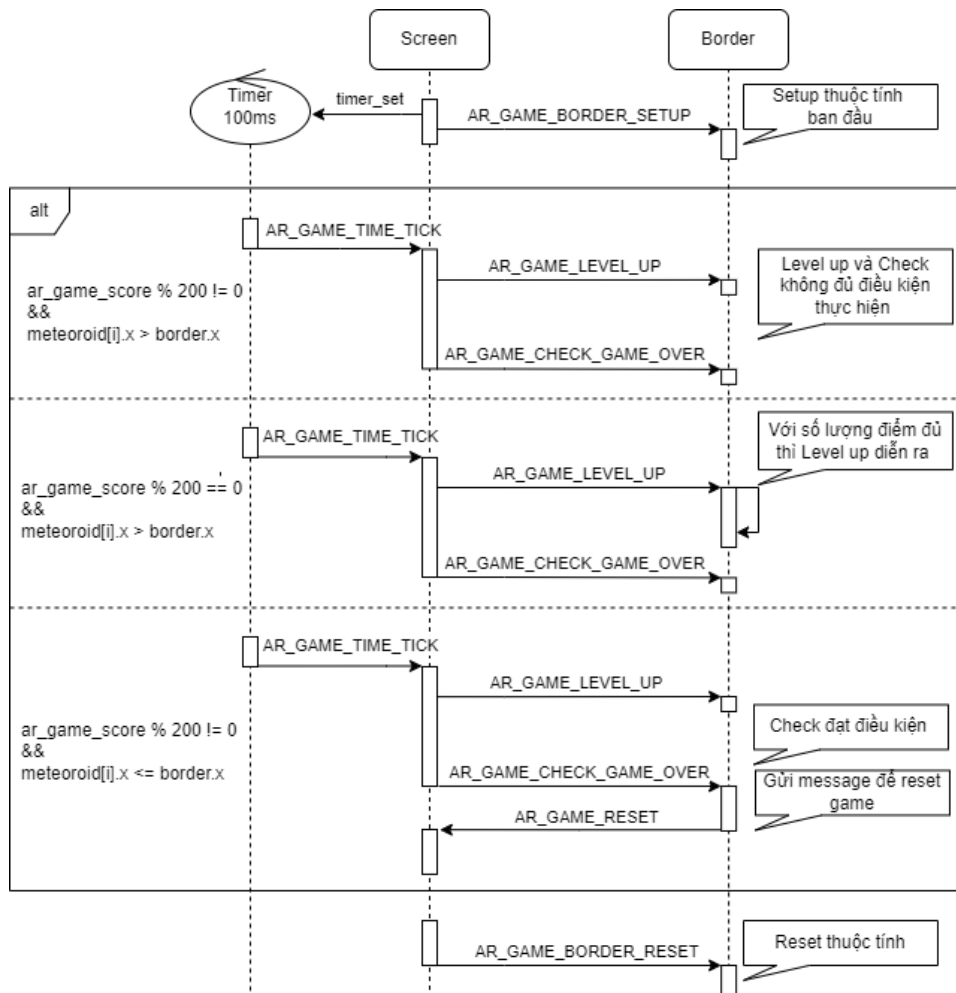
Task:

- Task ID: AR_GAME_BORDER_ID
- Task level: TASK_PRI_LEVEL_4
- App task: ar_game_border_handle

Signal:

- AR_GAME_BORDER_SETUP - Cài đặt thuộc tính ban đầu
- AR_GAME_BORDER_UPDATE - Kiểm tra số điểm (score) để cập nhật độ khó
- AR_GAME_BORDER_RESET - Cài đặt lại thuộc tính trước khi **End Game**
- AR_GAME_CHECK_GAME_OVER - Kiểm tra thiên thạch chạm vào ranh giới

Sequence diagram:



Code:

► Details

```

#include "ar_game_border.h"

#include "ar_game_meteoroid.h"
#include "ar_game_archery.h"

ar_game_border_t border;
uint32_t ar_game_score = 10;

#define AR_GAME_BORDER_SETUP() \
do { \
    border.x = AXIS_X_BORDER; \
    border.visible = WHITE; \
    border.action_image = 0; \
} while (0);
  
```

```
#define AR_GAME_BORDER_UPDATE() \
do { \
    if (ar_game_score%200 == 0) { \
        /* border.x += 10; */\
        if (settingsetup.meteoroid_speed < 6) { \
            settingsetup.meteoroid_speed++; \
            ar_game_score += 10; \
        } \
    } \
} while(0);

#define AR_GAME_BORDER_RESET() \
do { \
    border.x = AXIS_X_BORDER; \
    border.visible = BLACK; \
} while (0);

#define AR_GAME_CHECK_GAME_OVER() \
do { \
    for (uint8_t i = 0; i < NUM_METEORIDS; i++) { \
        if (meteoroid[i].x <= (border.x - 3)) { \
            task_post_pure_msg(AR_GAME_SCREEN_ID, AR_GAME_RESET); \
        } \
    } \
} while(0);

void ar_game_border_handle(ak_msg_t* msg) {
    switch (msg->sig) {
        case AR_GAME_BORDER_SETUP: {
            APP_DBG_SIG("AR_GAME_BORDER_SETUP\n");
            AR_GAME_BORDER_SETUP();
        }
        break;

        case AR_GAME_BORDER_UPDATE: {
            APP_DBG_SIG("AR_GAME_BORDER_UPDATE\n");
            AR_GAME_BORDER_UPDATE();
        }
        break;

        case AR_GAME_BORDER_RESET: {
            APP_DBG_SIG("AR_GAME_BORDER_RESET\n");
            AR_GAME_BORDER_RESET();
        }
        break;

        case AR_GAME_CHECK_GAME_OVER: {
            APP_DBG_SIG("AR_GAME_CHECK_GAME_OVER\n");
            AR_GAME_CHECK_GAME_OVER();
        }
        break;

        default:
```

```

        break;
    }
}

```

3.5 Thiên thạch (Meteoroid)

Mô tả: Thiên thạch là đối tượng xuất hiện thường xuyên trên màn hình luôn di chuyển từ Phải sang Trái. Vị trí xuất hiện có tính ngẫu nhiên có kiểm soát. Thiên thạch có thể bị phá hủy bởi Mũi tên, sau khi nổ xong tự động xuất hiện lại ngẫu nhiên.

Thuộc tính:

- Vị trí tạo độ: x (*uint32_t*), y (*uint32_t*)
- Hiển thị: visible (*bool*)
- Animation: action_image (*uint8_t*)

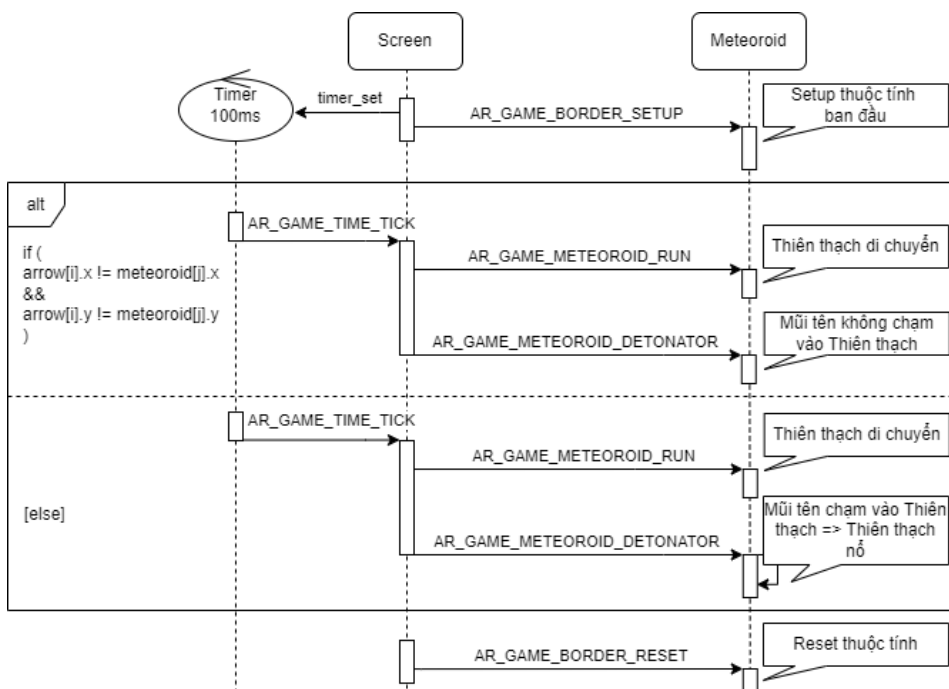
Task:

- Task ID: AR_GAME_METEOROID_ID
- Task level: TASK_PRI_LEVEL_4
- App task: ar_game_meteoroid_handle

Signal:

- AR_GAME_METEOROID_SETUP - Cài đặt thuộc tính ban đầu
- AR_GAME_METEOROID_RUN - Cập nhật thuộc tính theo **Timer**
- AR_GAME_METEOROID_DETONATOR - Kiểm tra mũi tên có chạm phải thiên thạch không
- AR_GAME_METEOROID_RESET - Cài đặt lại thuộc tính trước khi **End Game**

Sequence diagram:



Code:

► Details

```

#include "ar_game_meteoroid.h"

#include "ar_game_arrow.h"
#include "ar_game_bang.h"
#include "ar_game_border.h"
#include "scr_archery_game.h"

ar_game_meteoroid_t meteoroid[NUM_METEORIDS];

#define AR_GAME_METEOROID_SETUP() \
do { \
    meteoroid[0].y = AXIS_Y_METEOROID_0; \
    meteoroid[1].y = AXIS_Y_METEOROID_1; \
    meteoroid[2].y = AXIS_Y_METEOROID_2; \
    meteoroid[3].y = AXIS_Y_METEOROID_3; \
    meteoroid[4].y = AXIS_Y_METEOROID_4; \
    for (uint8_t i = 0; i < NUM_METEORIDS; i++) { \
        meteoroid[i].x = (rand() % 39) + 130; \
        meteoroid[i].visible = WHITE; \
        meteoroid[i].action_image = rand() % 3 + 1; \
    } \
} while (0);

#define AR_GAME_METEOROID_RUN() \
do { \
    for (uint8_t i = 0; i < NUM_METEORIDS; i++) { \
        if (meteoroid[i].visible == WHITE) { \
            meteoroid[i].x -= settingsetup.meteoroid_speed; \
            if (meteoroid[i].action_image < 4) { \
                meteoroid[i].action_image++; \
            } \
            if (meteoroid[i].action_image == 4) { \
                meteoroid[i].action_image = 1; \
            } \
        } \
    } \
} while(0);

#define AR_GAME_METEOROID_DETONATOR() \
do { \
    for (uint8_t i = 0; i < NUM_BANG; i++) { \
        if (meteoroid[i].visible == WHITE) { \
            for (uint8_t j = 0; j < MAX_NUM_ARROW; j++) { \
                if (meteoroid[i].x < (arrow[j].x + SIZE_BITMAP_ARROW_X -
3)) { \
                    if ((meteoroid[i].y + 8) > arrow[j].y) { \
                        if ((meteoroid[i].y - 1) < arrow[j].y) { \
                            meteoroid[i].visible = BLACK; \
                            arrow[j].visible = BLACK; \
                            bang[i].visible = WHITE; \
                            bang[i].x = meteoroid[i].x-5; \

```

22 / 30







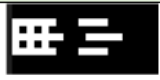


IV. Hiển thị và âm thanh trong trò chơi bắn cung

4.1 Đồ họa

Trong trò chơi, màn hình hiển thị là 1 màn hình LCD có kích thước là 128*64px. Nên các đối tượng được hiển thị trong game phải có kích thước hiển thị phù hợp với màn hình nên cần được thiết kế riêng.

Đồ họa được thiết kế từng phần theo từng đối tượng bằng phần mềm [Photopea](#)

Thiết kế đồ họa cho các đối tượng

Tên đối tượng		Hình ảnh	Mô tả
Archery (15x15 px)	bitmap_archery_I []		Cung lúc có tên
	bitmap_archery_II []		Cung lúc hết tên
Arrow (10x5px)	bitmap_arrow []		Mũi tên
Bang (15x15 px) (10x10 px)	bitmap_bang_I []		Vụ nổ animation 1
	bitmap_bang_II []		Vụ nổ animation 2
	bitmap_bang_III []		Vụ nổ animation 3
Meteoroid (20x10 px)	bitmap_meteoroid_I []		Thiên thạch animation 1
	bitmap_meteoroid_II []		Thiên thạch animation 2
	bitmap_meteoroid_III []		Thiên thạch animation 3

Ghi chú: trong thiết kế trên có nhiều hoạt ảnh cho cùng 1 đối tượng là để tạo animation cho đối tượng đó tăng cảm giác lúc chơi game.

Code:

► Details

Archer display:

```
void ar_game_archery_display() {
    if (archery.visible == WHITE && settingsetup.num_arrow != 0) {
        view_render.drawBitmap( archery.x, \
                                archery.y - 10, \
```

```

        bitmap_archery_I, \
        SIZE_BITMAP_ARCHERY_X, \
        SIZE_BITMAP_ARCHERY_Y, \
        WHITE);
    }
    else if (archery.visible == WHITE && settingsetup.num_arrow == 0) {
        view_render.drawBitmap( archery.x, \
                                archery.y - 10, \
                                bitmap_archery_II, \
                                SIZE_BITMAP_ARCHERY_X, \
                                SIZE_BITMAP_ARCHERY_Y, \
                                WHITE);
    }
}

```

Arrow display:

```

void ar_game_arrow_display() {
    for (uint8_t i = 0; i < MAX_NUM_ARROW; i++) {
        if (arrow[i].visible == WHITE) {
            view_render.drawBitmap( arrow[i].x, \
                                    arrow[i].y, \
                                    bitmap_arrow, \
                                    SIZE_BITMAP_ARROW_X, \
                                    SIZE_BITMAP_ARROW_Y, \
                                    WHITE);
        }
    }
}

```

Meteoroid display:

```

void ar_game_meteoroid_display() {
    for (uint8_t i = 0; i < NUM_METEORIDS; i++) {
        if (meteoroid[i].visible == WHITE) {
            if (meteoroid[i].action_image == 1) {
                view_render.drawBitmap( meteoroid[i].x, \
                                        meteoroid[i].y, \
                                        bitmap_meteoroid_I, \
                                        SIZE_BITMAP_METEORIDS_X, \
                                        SIZE_BITMAP_METEORIDS_Y, \
                                        WHITE);
            }
            else if (meteoroid[i].action_image == 2) {
                view_render.drawBitmap( meteoroid[i].x, \
                                        meteoroid[i].y, \
                                        bitmap_meteoroid_II, \
                                        SIZE_BITMAP_METEORIDS_X, \
                                        SIZE_BITMAP_METEORIDS_Y, \

```



```

        WHITE);
    }
    else if (meteoroid[i].action_image == 3) {
        view_render.drawBitmap( meteoroid[i].x, \
                                meteoroid[i].y, \
                                bitmap_meteoroid_III, \
                                SIZE_BITMAP_METEORIDS_X, \
                                SIZE_BITMAP_METEORIDS_Y, \
                                WHITE);
    }
}
}
}
}

```

Bang display:

```

void ar_game_bang_display() {
    for (uint8_t i = 0; i < NUM_BANG; i++) {
        if (bang[i].visible == WHITE) {
            if (bang[i].action_image == 1) {
                view_render.drawBitmap( bang[i].x, \
                                        bang[i].y, \
                                        bitmap_bang_I, \
                                        SIZE_BITMAP_BANG_I_X, \
                                        SIZE_BITMAP_BANG_I_Y, \
                                        WHITE);
            }
            else if (bang[i].action_image == 2) {
                view_render.drawBitmap( bang[i].x, \
                                        bang[i].y, \
                                        bitmap_bang_II, \
                                        SIZE_BITMAP_BANG_I_X, \
                                        SIZE_BITMAP_BANG_I_Y, \
                                        WHITE);
            }
            else if (bang[i].action_image == 3) {
                view_render.drawBitmap( bang[i].x + 2, \
                                        bang[i].y - 1, \
                                        bitmap_bang_III, \
                                        SIZE_BITMAP_BANG_II_X, \
                                        SIZE_BITMAP_BANG_II_Y, \
                                        WHITE);
            }
        }
    }
}
}
}
}

```

Border display:

```

void ar_game_border_display() {
    if (border.visible == WHITE) {
        view_render.drawFastVLine( border.x, \
                                   AXIS_Y_BORDER_ON, \
                                   AXIS_Y_BORDER_UNDER, \
                                   WHITE);

        for (uint8_t i = 0; i < NUM_METEOROIDS; i++) {
            view_render.fillCircle( border.x, \
                                   meteoroid[i].y + 5, \
                                   1, \
                                   WHITE);
        }
    }
}

```

Game frame display:

```

void ar_game_frame_display() {
    view_render.setTextSize(1);
    view_render.setTextColor(WHITE);
    view_render.setCursor(2,55);
    view_render.print("Arrow:");
    view_render.print(settingsetup.num_arrow);
    view_render.setCursor(60,55);
    view_render.print(" Score:");
    view_render.print(ar_game_score);
    view_render.drawLine(0, LCD_HEIGHT,      LCD_WIDTH, LCD_HEIGHT,
    WHITE);
    view_render.drawLine(0, LCD_HEIGHT-10,  LCD_WIDTH, LCD_HEIGHT-10,
    WHITE);
    view_render.drawRect(0, 0, 128, 64, 1);
}

```

Screen display:

```

void view_scr_archery_game() {
    if (ar_game_status == GAME_ON) {
        ar_game_frame_display();
        ar_game_archery_display();
        ar_game_arrow_display();
        ar_game_meteoroid_display();
        ar_game_bang_display();
        ar_game_border_display();
    }
    else if (ar_game_status == GAME_OVER) {
        view_render.clear();
        view_render.setTextSize(2);
        view_render.setTextColor(WHITE);
        view_render.setCursor(17, 24);
    }
}

```

```
        view_render.print("YOU LOSE");  
    }  
}
```

4.3 Âm thanh

Âm thanh được thiết kế qua web [Arduino Music](#)

Trong trò chơi, để trò chơi thêm phần xin động thì việc có âm thanh là điều cần thiết.

Các âm thanh cần thiết kể: nút nhấn, bắn tên, vụ nổ, nhạc game.

Code:

► Details

```
// Âm thanh Bắt đầu game  
BUZZER_PlayTones(tones_SMB);  
  
// Âm thanh Vụ nổ  
BUZZER_PlayTones(tones BUM);  
  
// Âm thanh nút nhấn  
BUZZER_PlayTones(tones_cc);  
  
// Âm thanh cảnh báo  
BUZZER_PlayTones(tones_3beep);  
  
// Merry Christmas  
BUZZER_PlayTones(tones_merryChrismast);  
  
/* _____BUZZER_____ */  
  
void BUZZER_Sleep(bool sleep);  
/*  sleep = 0 : bật âm thanh  
   sleep = 1 : tắt âm thanh */  
static const Tone_TypeDef tones BUM[] = {  
    {3000,3},  
    {4500,6},  
    { 0,0}  
};  
  
static const Tone_TypeDef tones_cc[] = {  
    {2000,2},  
    { 0,0},  
};  
  
static const Tone_TypeDef tones_startup[] = {  
    {2000,3},  
    { 0,3},  
    {3000,3},  
    { 0,3},  
};
```

```

    {4000, 3},
    {    0, 3},
    {1200, 4},
    {    0, 6},
    {4500, 6},
    {    0, 0}      // <-- tones end
};

static const Tone_TypeDef tones_3beep[] = {
    {4000, 3},
    {    0, 10},
    {1000, 6},
    {    0, 10},
    {4000, 3},
    {    0, 0}
};

// "Super Mario bros." =)
static const Tone_TypeDef tones_SMB[] = {
    {2637, 18}, // E7 x2
    {    0, 9}, // x3
    {2637, 9}, // E7
    {    0, 9}, // x3
    {2093, 9}, // C7
    {2637, 9}, // E7
    {    0, 9}, // x3
    {3136, 9}, // G7
    {    0, 27}, // x3
    {1586, 9}, // G6
    {    0, 27}, // x3

    {2093, 9}, // C7
    {    0, 18}, // x2
    {1586, 9}, // G6
    {    0, 18}, // x2
    {1319, 9}, // E6
    {    0, 18}, // x2
    {1760, 9}, // A6
    {    0, 9}, // x1
    {1976, 9}, // B6
    {    0, 9}, // x1
    {1865, 9}, // AS6
    {1760, 9}, // A6
    {    0, 9}, // x1

    {1586, 12}, // G6
    {2637, 12}, // E7
    {3136, 12}, // G7
    {3520, 9}, // A7
    {    0, 9}, // x1
    {2794, 9}, // F7
    {3136, 9}, // G7
    {    0, 9}, // x1
    {2637, 9}, // E7

```

```

    { 0, 9}, // x1
    {2093, 9}, // C7
    {2349, 9}, // D7
    {1976, 9}, // B6
    { 0,18}, // x2

    {2093, 9}, // C7
    { 0,18}, // x2
    {1586, 9}, // G6
    { 0,18}, // x2
    {1319, 9}, // E6
    { 0,18}, // x2
    {1760, 9}, // A6
    { 0, 9}, // x1
    {1976, 9}, // B6
    { 0, 9}, // x1
    {1865, 9}, // AS6
    {1760, 9}, // A6
    { 0, 9}, // x1

    {1586,12}, // G6
    {2637,12}, // E7
    {3136,12}, // G7
    {3520, 9}, // A7
    { 0, 9}, // x1
    {2794, 9}, // F7
    {3136, 9}, // G7
    { 0, 9}, // x1
    {2637, 9}, // E7
    { 0, 9}, // x1
    {2093, 9}, // C7
    {2349, 9}, // D7
    {1976, 9}, // B6

    { 0, 0}
};

// Merry Christmas
static const Tone_TypeDef tones_merryChrismast[] = {
    {2637, 9}, // E7
    { 0, 9}, // x1
    {2637, 9}, // E7
    { 0, 9}, // x1
    {2637, 9}, // E7
    { 0,18}, // x2

    {2637, 9}, // E7
    { 0, 9}, // x1
    {2637, 9}, // E7
    { 0, 9}, // x1
    {2637, 9}, // E7
    { 0,18}, // x2

    {2637, 9}, // E7

```

```
{ 0, 9}, // x1
{3136, 9}, // G7
{ 0, 9}, // x1
{2093, 9}, // C7
{ 0, 9}, // x1
{2349, 9}, // D7
{ 0, 9}, // x1
{2637, 9}, // E7
{ 0, 24}, // x2

{2794, 9}, // F7
{ 0, 9}, // x1
{2794, 9}, // F7
{ 0, 9}, // x1
{2794, 9}, // F7
{ 0, 9}, // x1
{2794, 9}, // F7
{ 0, 9}, // x1
{2794, 9}, // F7
{ 0, 9}, // x1
{2637, 9}, // E7
{ 0, 9}, // x1
{2637, 9}, // E7
{ 0, 9}, // x1
{2637, 9}, // E7
{ 0, 9}, // x1
{2637, 9}, // E7
{ 0, 9}, // x1
{2637, 9}, // E7
{ 0, 9}, // x1
{2349, 9}, // D7
{ 0, 9}, // x1
{2349, 9}, // D7
{ 0, 9}, // x1
{2637, 9}, // E7
{ 0, 9}, // x1
{2349, 9}, // D7
{ 0, 9}, // x1
{3136, 9}, // G7
{ 0, 0} // <-- tones end
};
```

Ghi chú: Nếu không có thời gian hay không có kiểu âm nhạc thì tốt nhất nên dùng các thư viện trên github