TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
ĐẠI HỌC BÁCH KHOA HÀ NỘI

Báo cáo thực hành hệ nhúng
Bài thực hành số 3

Giảng viên hướng dẫn: Mai Xuân Ngọc
Nhóm thực hiện: Phan Đức Duy – 20225831
Phạm Tùng Dương – 20225825
Sinh viên báo cáo: Phạm Tùng Dương – 20225825
Ngày nộp: Tuần 38 – 24/5/2025

Nội dung

I. Bài 3.1:

1. Giao diện khởi tạo:

- Sử dụng TouchGFX Designer để tạo một Screen1 có:
  - Một hình tròn (circle1) với bán kính 10px, màu sắc tùy chọn.
  - Một nút bấm có nhãn "LED control".

2. Xử lý sự kiện:

- Khi người dùng nhấn nút "LED control", hàm buttonClicked() được gọi.
  - Hàm này toggle đèn LED13.
- Trong mỗi lần tick (tickEvent()), chương trình cập nhật vị trí của hình tròn, tạo hiệu ứng di chuyển ngang .

3. Tích hợp FreeRTOS:

- Một task liên tục kiểm tra trạng thái nút bấm USER_BUTTON (chân PA0).
- Nếu nút được bấm, task gửi một tín hiệu vào queue.
- Trong tickEvent() của giao diện, chương trình nhận tín hiệu từ queue và cập nhật vị trí hình tròn theo quỹ đạo dao động lên xuống + di chuyển ngang.

```
void Screen1View::tickEvent()
{
tickCount += 2;
tickCount = tickCount % 240;
float x = tickCount / 55.0f;
float y = sin(x)+sin(2*x)+sin(3*x)+1;

uint8_t res;
if (osMessageQueueGetCount(myQueue01Handle) > 0)
{
    osMessageQueueGet(myQueue01Handle, &res, NULL, osWaitForever);
    if (res == 'X')
    {
        circle1.moveTo((int16_t)floor(x*55), 200 - (int16_t)floor(y*50));
        circle1.invalidate();
    }
}

}
```

- Biến tickCount tăng dần theo từng lần tick, nhưng giới hạn trong khoảng [0, 239] để tạo hiệu ứng chạy lặp lại.
- tickcount là tham số thời gian để tính toán vị trí hình tròn.
- Biến x là giá trị thời gian thực đã chuẩn hóa.
- Biến y là một tổng hợp sóng sin tạo ra chuyển động mượt dạng sóng.
- Cộng thêm 1 để đảm bảo y luôn dương (vì giá trị nhỏ nhất của tổng sin là âm).
- Đọc 1 phần tử từ queue, lưu vào res.
- Nếu giá trị đọc được là ký tự 'X', thì: Thực hiện hành động cập nhật giao diện (di chuyển hình tròn).

II. Bài 3.2:

Hoạt động của chương trình:

1. Thiết kế giao diện:

- Dùng TouchGFX để đặt một ảnh mặt đồng hồ (Image) và một kim đồng hồ (TextureMapper).

- Thiết lập góc quay Z-Angle cho kim phù hợp với chuyển động quay.

2. **Cập nhật thời gian:**

Code:

```
void Screen1View::handleTickEvent()
{
  Screen1ViewBase::handleTickEvent();
  uint32_t tickCount = osKernelGetTickCount();
  uint32_t seconds = tickCount / 1000;
  float degree = (seconds % 60) * 6.0f;
  txtrHand.updateZAngle(degree);
  txtrHand.invalidate();
}
```

- Trong tickEvent() hoặc handleTickEvent():
  - Sử dụng osKernelGetTickCount() để lấy số tick kể từ khi hệ điều hành chạy.
  - Tính toán thời gian theo đơn vị giây (mỗi 1000 tick nếu tick rate là 1ms).
  - Từ đó suy ra góc quay kim đồng hồ.
  - Cập nhật Z-Angle của kim tương ứng để tạo hiệu ứng quay.

3. **Tương tác với USER_BUTTON:**

- Một task trong FreeRTOS liên tục kiểm tra trạng thái nút nhấn.
- Nếu nhận được tín hiệu 'P' (tức là đang bấm nút)
  ⇨ Tính góc quay theo số tick đã trôi qua kể từ lần bắt đầu hoặc lần nhả nút
- Nếu không phải 'P' (tức là thả nút) => lưu thời điểm hiện tại

```
void Screen1View::handleTickEvent()
{
static uint32_t prevTick = 0;
Screen1ViewBase::handleTickEvent();
tickCount++;
float rad = 0;

uint8_t res = 0;
uint32_t count = osMessageQueueGetCount(Queue1Handle);
if (count > 0)
{
    osMessageQueueGet(Queue1Handle, &res, NULL, osWaitForever);
    if (res == 'P')
    {
    //update and invalidate the clock hand
    rad = ((tickCount - prevTick) % 360)* 3.14f / 180;
    txtrHand.updateZAngle(rad);
    }
else
{
prevTick = tickCount;
}
    }
        }
```

Kết quả quan sát:

- Khi không bấm nút: kim đứng yên.
- Khi bấm và giữ nút: kim quay theo chiều kim đồng hồ (bắt đầu bấm giờ).
- Khi thả nút: kim dừng lại (dừng bấm giờ).

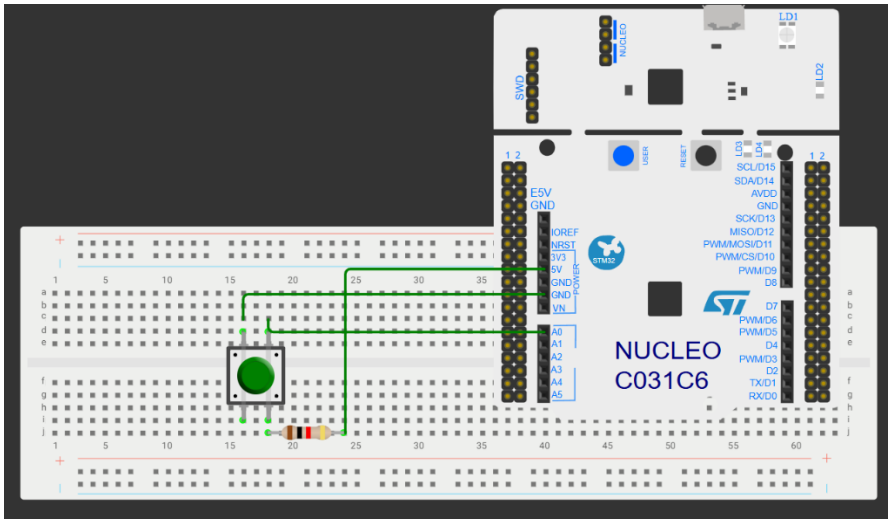
III. Bài 3.4 (tự làm):

Phần cứng sử dụng: 2 nút bấm

- **Cấu hình:**

```
/*Configure GPIO pin : PG2 and PG3 for input */
GPIO_InitStruct.Pin = GPIO_PIN_2 | GPIO_PIN_3;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_PULLUP;
HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);
```

- **Cách ghép nối:**



Giả sử A0 như trong hình là chân PG2, PG3 cần nối.

TỔNG QUAN GAME

- Người chơi điều khiển xe (image1) sang trái/phải bằng các nút đã được ghép nối (L, R gửi qua queue).
- Một vật thể (cừu – lamb) rơi từ trên xuống.
- Nếu xe va chạm với cừu ⟶ Game Over.
- Nếu tránh được ⟶ điểm tăng và cừu xuất hiện lại ở vị trí mới.
- Nếu thua thì ấn nút PA0 để chơi lại, vẫn lưu kết quả cũ nếu như nó là highscore.

```
// 1. Xử lý nút điều khiển xe
uint8_t msg;
if (osMessageQueueGet(Queue1Handle, &msg, NULL, 0) == osOK)
{
    const int16_t minX = 0;
    const int16_t maxX = 240 - image1.getWidth();  // giới hạn không ra ngoài màn hình

    if (msg == 'L' && localImageX > minX)
    {
        localImageX -= carSpeedStep;
    }
    else if (msg == 'R' && localImageX < maxX)
    {
        localImageX += carSpeedStep;
    }

    image1.setX(localImageX);
    image1.invalidate();
}
```

Giới hạn vùng di chuyển của xe (image1) theo trục X:

- minX: mép trái màn hình.
- maxX: mép phải màn hình trừ đi chiều rộng xe ⟶ để không bị trượt khỏi màn hình.
    - Nếu nhận được 'L':
- Giảm localImageX để xe di chuyển sang trái.
- Nhưng chỉ khi chưa tới mép trái (> minX).
    - Nếu nhận được 'R':
- Tăng localImageX để xe di chuyển sang phải.
- Nhưng chỉ khi chưa vượt mép phải (< maxX).
    - carSpeedStep là số pixel xe di chuyển mỗi lần nhấn nút.

```
uint32_t xorshift32()
{
    static uint32_t seed = 0;

    if (seed == 0)
        seed = HAL_GetTick();  // hoặc giá trị ADC noise nếu có

    seed ^= seed << 13;
    seed ^= seed >> 17;
    seed ^= seed << 5;

    return seed;
}
```

```
        textHighScore.invalidate();
        int index = xorshift32() % 4;
        int newX = index * 60 + 15;
        lamb.setX(newX);
```

- Hàm xorshift32 trả về một số nguyên 32-bit không dấu, là kết quả random.

Nếu seed chưa được khởi tạo (== 0), dùng HAL_GetTick() để gán:

- HAL_GetTick() trả về số ms kể từ khi MCU khởi động, gần giống "thời gian thực".
- Giúp tạo sự "ngẫu nhiên ban đầu" ⟶ không phải lần nào cũng sinh ra cùng một dãy số.
- Dùng thuật toán XORShift – dùng các phép dịch bit và XOR để khuấy trộn bit trong seed. ⟶ sinh giá trị mới, ngẫu nhiên hơn.
- Nếu cừu đi hết màn hình sẽ cập nhật lại vị trí ngẫu nhiên theo trục ngang (trục x).

```cpp
bool Screen2View::checkCollision()
{
    int16_t x1 = image1.getX();
    int16_t y1 = image1.getY();
    int16_t w1 = image1.getWidth();
    int16_t h1 = image1.getHeight();

    int16_t x2 = lamb.getX();
    int16_t y2 = lamb.getY();
    int16_t w2 = lamb.getWidth();
    int16_t h2 = lamb.getHeight();

    return !(x1 + w1 < x2 || x2 + w2 < x1 ||
             y1 + h1 < y2 || y2 + h2 < y1);
}
```

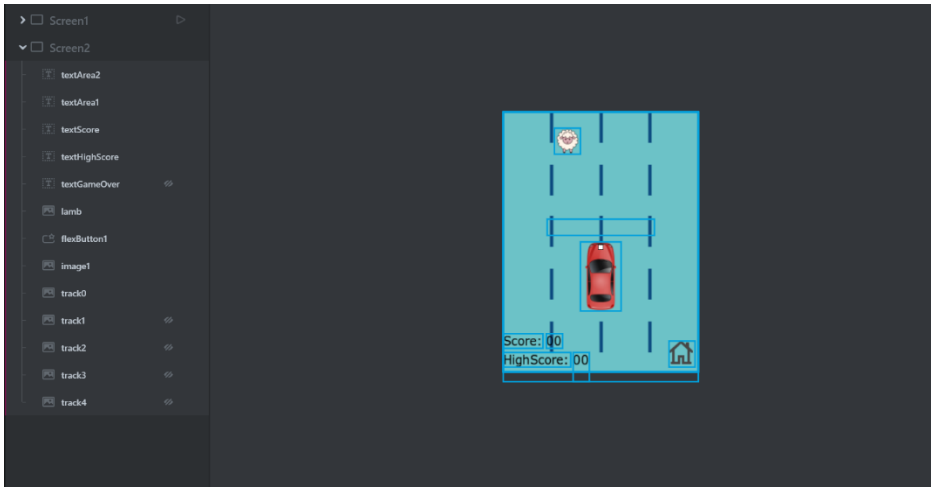
Hàm trên sẽ kiểm tra xem xe và cừu có va chạm hay chưa

Lấy tọa độ và kích thước của xe và cừu.

Xét điều kiện phủ định (NOT) của 4 trường hợp *không va chạm*:

1. Xe hoàn toàn bên trái cừu.
2. Cừu hoàn toàn bên trái xe.
3. Xe trên cừu.
4. Cừu trên xe.
5. Nếu không rơi vào bất kỳ trường hợp nào ở trên, thì hai hình chữ nhật đang giao nhau ⟶ va chạm.

Phần giao diện bổ sung



- Thêm 1 textArea vào chính giữa màn hình khi có xảy ra va chạm (gameOver)
- Thêm 2 textArea để hiển thị label Score và HighScore.
- 2 textArea tương ứng để hiển thị số điểm
- Các textArea để hiển thị số điểm sẽ thêm các Wildcard để có thể thay đổi thông số trong game.

```
Screen2View::Screen2View()
    : isGameOver(false), fallSpeed(2), speedLevel(0),carSpeedStep(2)
{
    tickCount = 0;
}
```

Khai báo các biến tốc độ cừu và tốc độ xe để làm chức năng tốc độ tăng dần. (tốc độ cừu và xe mặc định là 2pixel/s)

```
    // Cứ mỗi 300 tick (~5 giây @ 60Hz), tăng tốc một lần
    if (tickCount % 300 == 0 && fallSpeed < 5)
    {
        fallSpeed++;      // tăng tốc cừu
        speedLevel++;     // tăng cấp độ (tùy dùng cho điểm số nếu cần)
        carSpeedStep = fallSpeed;  // đồng bộ tốc độ xe theo tốc độ cừu
    }

    invalidate();
}
```

```
    int newY = lamb.getY() + fallSpeed;
    if (newY >= 320)
```

Cứ mỗi 5 giây sẽ tăng tốc độ 1 lần, max khi chơi 25s tức tốc độ cừu = 5.

```
    int newY = lamb.getY() + fallSpeed;
    if (newY >= 320)
    {
        newY = 0;
        score++;
        if (score > highScore)
            {
                highScore = score;
            }
        // Cập nhật hiển thị điểm
        Unicode::snprintf(scoreBuffer, sizeof(scoreBuffer), "%d", score);
        textScore.setWildcard(scoreBuffer);
        textScore.invalidate();

        Unicode::snprintf(highScoreBuffer, sizeof(highScoreBuffer), "%d", highScore);
        textHighScore.setWildcard(highScoreBuffer);
        textHighScore.invalidate();
        int index = xorshift32() % 4;
        int newX = index * 60 + 15;
        lamb.setX(newX);
        lamb.invalidate(); // Thêm dòng này để cập nhật giao diện
    }
    lamb.setY(newY);

    if (checkCollision())
    {
        isGameOver = true;

        textGameOver.setVisible(true);
        lamb.setVisible(false);
        textGameOver.invalidate();
    }
```

- Khi chơi game, tọa độ cừu chạy hết chiều dọc sẽ +1 điểm cho score và lưu vào highscore nếu lớn hơn highscore trước đó.
- Hàm setWildcard để thay đổi giá trị các textArea tương ứng đã cấu hình trên touchgfx.
- Kiểm tra nếu va chạm thì gameOver thì dừng chơi và hiện lên dòng chữ như đã thêm vào touchgfx.

```
if (isGameOver)
{
    // Chờ người dùng nhấn nút PA0 để chơi lại
    uint8_t msg;
    if (osMessageQueueGet(Queue1Handle, &msg, NULL, 0) == osOK && msg == 'A')
    {
        // Reset lại trạng thái game
        score = 0;

        Unicode::snprintf(scoreBuffer, sizeof(scoreBuffer), "%d", score);
        textScore.setWildcard(scoreBuffer);
        textScore.invalidate();

        isGameOver = false;
        tickCount = 0;
        fallSpeed = 2;
        speedLevel = 0;
        localImageX = 100;
        carSpeedStep = 2;
        lamb.setY(0);

        lamb.setVisible(true);
        image1.setX(localImageX);
        textGameOver.setVisible(false);

        // Hiện lại đường
        track0.setVisible(true);
        track1.setVisible(false);
        track2.setVisible(false);
        track3.setVisible(false);
        track4.setVisible(false);

        image1.invalidate();
        lamb.invalidate();
        textGameOver.invalidate();
    }

    return; // Không chạy game khi đang Game Over
```

Khi nhấn nút PA0 thì reset mọi thứ về ban đầu trừ Highscore và chơi lại từ đầu.