HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

PROJECT

Lập trình game Tetris trên Arduino Nano

NHÓM 3
Học phần: Hệ nhúng - IT4210
Mã lớp: 139318

Giảng viên: TS. Ngô Lam Trung _____

Signature

Thành viên nhóm:

MSSV	Họ và tên	Vai trò
20205093	Hoàng Mai Thùy Linh	Thiết kế & lập trình
20204981	Vũ Huy Hoàng	Lập trình

HANOI, 08/2023

ACKNOWLEDGMENT

Trong suốt thời gian học tập và hoàn thiện project học phần Hệ nhúng, chúng em đã được thầy cung cấp kiến thức nền tảng về hệ thống nhúng. Từ những ngày đầu làm quen với vi điều khiển 8051, arduino, đến ARM,... thầy vẫn luôn rất nhiệt tình và tận tâm chỉ bảo những sinh viên còn non nớt chúng em tìm hiểu kiến thức mới.

Dù còn một số khó khăn vì kiến thức bản thân còn hạn hẹp, nhóm đã rất quyết tâm tìm hiểu thêm các công cụ hữu ích để triển khai project lần này. Chúng em xin chân thành cảm ơn thầy và các bạn cùng lớp đã luôn giúp đỡ chúng em tìm tòi và giải quyết vấn đề.

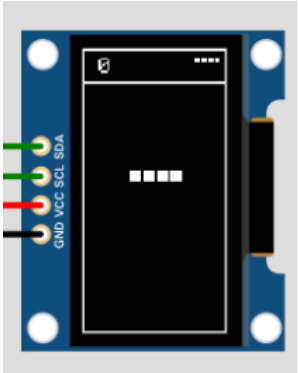
Một lần nữa, chúng em xin chân thành cảm ơn thầy đã mang đến cho chúng em một học phần đầy ý nghĩa. Học phần đã sắp kết thúc, nhưng những kiến thức mà thầy đã truyền thụ cho chúng em chắc chắn sẽ còn được chúng em ghi nhớ lâu dài.

TÓM TẮT NỘI DUNG

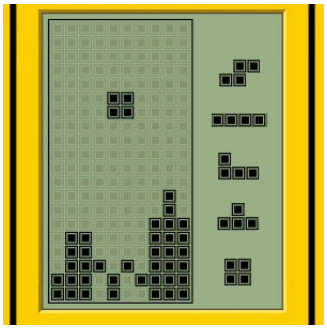
Tetris là một trò chơi điện tử xếp khối được phát triển bởi Alexey Pajitnov vào năm 1984. Trò chơi đã được chuyển thể sang nhiều nền tảng khác nhau và trở thành một trong những trò chơi điện tử phổ biến nhất mọi thời đại.

Mục tiêu của Tetris là xếp các khối hình học rơi xuống theo cách sao cho chúng lấp đầy một hàng ngang. Khi một hàng ngang được lấp đầy, nó sẽ biến mất và các hàng phía trên sẽ rơi xuống để lấp chỗ trống. Nếu các hàng phía trên chạm đến đầu màn hình, trò chơi sẽ kết thúc.

Cụ thể trong project này, nhóm 3 chúng em phát triển **"Lập trình game Tetris trên Arduino Nano"** bằng cách mô phỏng lại màn hình trò chơi trên màn OLED SSD1306, Arduino Nano , buzzer và các nút bấm điều khiển trò chơi.



Hình 0.1: Mô phỏng trên SSD1306



Hình 0.2: Màn hình Tetris truyền thống

Sau đây là báo cáo các nội dung mà nhóm đã thực hiện được. Cũng như các kết quả mô phỏng và những nhận xét về ưu, nhược điểm của bài tập lớn.

TABLE OF CONTENTS

THIẾT BỊ SỬ DỤNG

Mô tả	**Số lượng**
Còi báo hiệu	01 buzzer
Cấp nguồn	01 jack USB mini-B
Dây nối	20 dây cắm breadboard
Màn hình	01 OLED SSD1306
Nút bấm	04 button
Vi điều khiển	01 Arduino Nano

CHAPTER 1. MỞ ĐẦU

1.1 Đặt vấn đề

Chủ yếu công việc xoay quanh việc lập trình trên Arduino Nano và kết nối vi điều khiển này với các thiết bị ngoại vi còn lại như màn hình OLED, button.

1.2 Mục tiêu và định hướng giải pháp

Trong project lần này, dựa trên yêu cầu của đề bài, chúng em đã triển khai theo 3 giai đoạn chính như sau:

1. Lựa chọn và tìm hiểu thông số của các thiết bị sử dụng trong hệ thống.
2. Dựa trên trò chơi Tetris truyền thống, xác định các thông số về mặt hình ảnh.
3. Thực hiện lắp đặt mạch và xây dựng mã nguồn.

1.3 Bố cục báo cáo

Phần còn lại của báo cáo được tổ chức như sau.

Chương 2 mô tả các thiết bị được dùng trong hệ thống và cách lắp đặt vật lý trên breadboard của chúng.

Chương 3 giới thiệu về Tetromino - các khối hình học Tetris. Cách chúng ta mô hình hóa các khối hình này khi đưa vào lập trình trên máy tính.

Chương 4 giải thích từng phần thực thi source code tương ứng với các chức năng riêng biệt. Đính kèm là hình ảnh thực tế kết quả bài toán.

Chương 5 kết luận, nêu nhận xét tổng thể.

CHAPTER 2. TỔNG QUAN CÁC LINH KIỆN TRONG HỆ THỐNG

2.1 Arduino Nano

Arduino Nano là một bảng mạch điện tử có kích thước nhỏ chỉ bằng 1 nửa đồng xu gấp lại, được phát triển dựa trên dựa trên ATmega328P phát hành vào năm 2008 và khá thân thiện với breadboard.

Nó có tổng cộng 22 chân đầu vào/đầu ra, trong số đó có:

- 14 chân kỹ thuật số.
- 8 chân tương tự.
- 6 chân PWM trong số các chân kỹ thuật số.
- Hỗ trợ các giao tiếp khác nhau, bao gồm:
  - Giao thức nối tiếp.
  - Giao thức I2C.
  - Giao thức SPI.

Sơ đồ các chân cắm như sau:

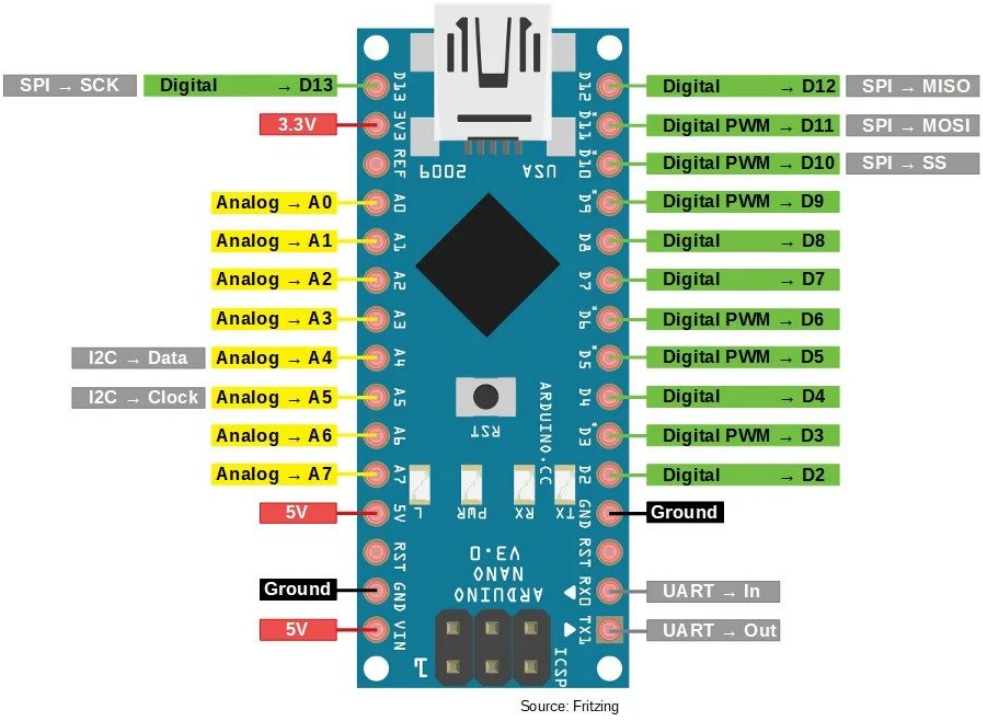


Figure 2.1: Cấu hình chân Arduino Nano.

2.2 Màn hình OLED 0.96

Màn hình OLED 0.96 có nhiều kích thước (chẳng hạn như 128×64, 128×32) và màu sắc (OLED trắng, xanh lam, vàng). OLED sử dụng giao tiếp truyền thông I2C và giao tiếp SPI. Điểm chung giữa hai màn hình OLED I2C và SPI là đều sử dụng IC SSD1306 có khả năng điều khiển các màn hình OLED có nhiều kích cỡ và độ phân giải, (128×64, 128×32). Nó hỗ trợ hiển thị màu sắc, độ tương phản cao và điều chỉnh được độ sáng.

IC SSD1306 xử lý tất cả bộ đệm RAM và yêu cầu rất ít tài nguyên từ Arduino.

Sơ đồ chân màn hình OLED I2C:

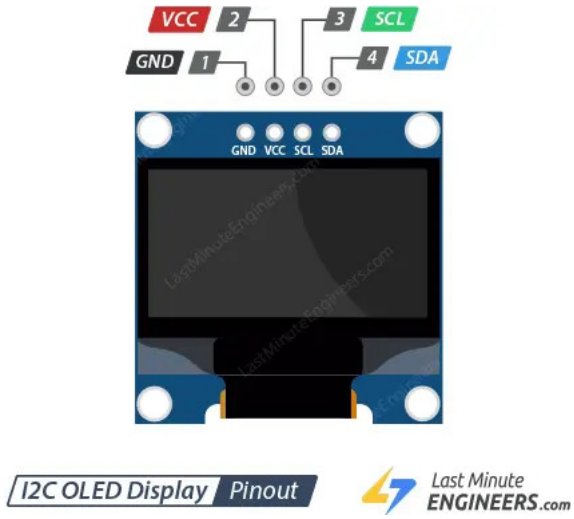


Figure 2.2: Cấu hình chân OLED I2C.

SSD1306 có 4 chân, trong số đó có:

- VCC: Chân nguồn (điện áp khuyến cáo là 5V)
- GND: Chân nối đất
- SDA: Chân dữ liệu
- SCL: Chân xung Clock SCL

2.3 Kết nối các thiết bị thành hệ thống

Các chân trên Arduino Nano đặc biệt cần chú ý:

(a) Chân kết nối với button để điều khiển chương trình:

- Chân D11: sang trái.
- Chân D9: sang phải.
- Chân D12: xoay khối.
- Chân D10: tăng tốc độ.

(b) Chân kết nối với màn hình hiển thị:

- Chân A5: SCL.
- Chân A4: SDA.

(c) Chân nối với còi: D3

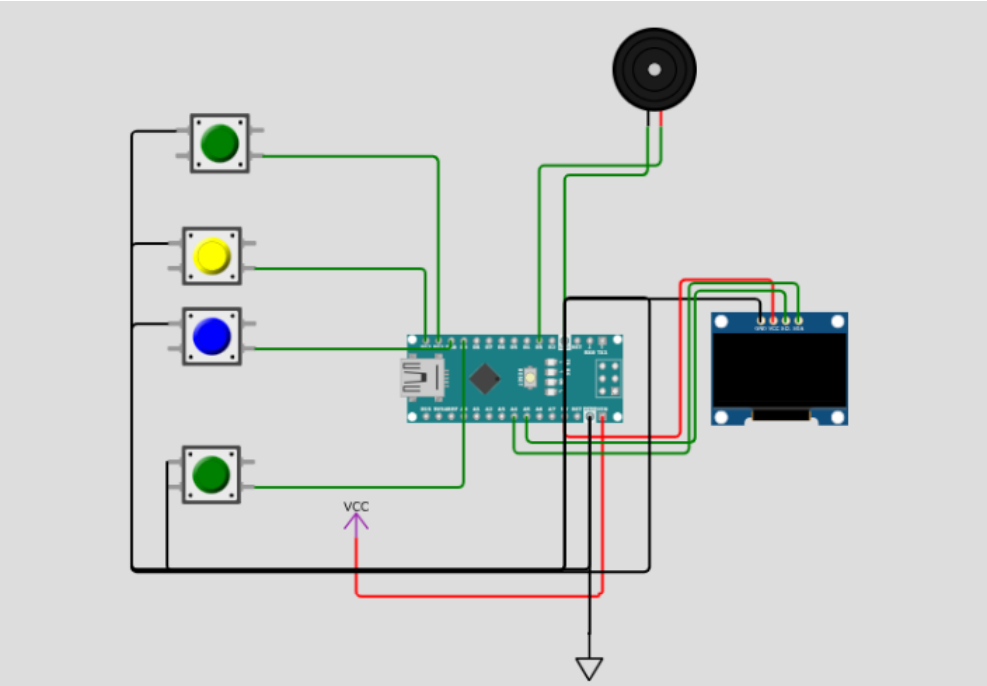


Figure 2.3: Hình ảnh mạch mô phỏng tại https://wokwi.com/.

CHAPTER 3. MÔ HÌNH HÓA BÀI TOÁN

3.1 Hình học Tetromino

Tetromino là một hình dạng hình học bao gồm bốn hình vuông, được kết nối với nhau trực giao (tức là ở các cạnh chứ không phải các góc).
Tetromino là một loại hình dạng đặc biệt sử dụng phổ biến trong trò chơi điện tử Tetris do nhà thiết kế trò chơi Liên Xô Alexey Pajitnov tạo ra.
Dễ hiểu hơn: Tetromino là các hình dạng hình học 2 chiều được tạo ra bằng cách kết hợp các ô vuông bốn đơn vị.

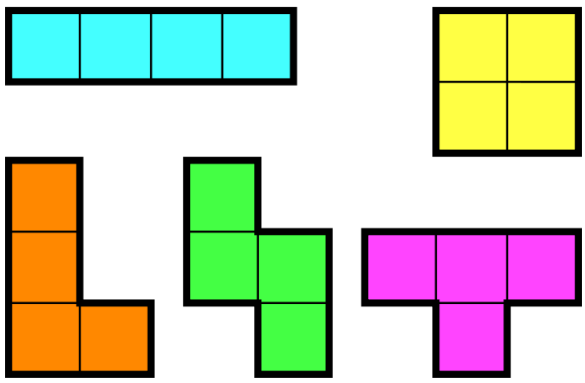


Figure 3.1: Các khối tetromino cơ bản.

Các khối này đều có thể bị xoay (theo chiều kim đồng hồ) cũng như di chuyển (sang trái hoặc phải). Tuy nhiên khối sẽ không thể xoay hay di chuyển được nếu gặp va chạm (với phần cạnh hay với các khối đã hạ cánh).

3.2 Game board

Còn hay được gọi là "playfield", hay "matrix",... Đây chính là phần bố cục ô lưới, và là nơi chính để chơi game của chúng ta. Ở màn hình trong bài tập, chính là màn hình SSD3106 kích cỡ 128x64 pixel, ta gộp **5 pixel** làm 1 ô vuông đơn vị.

Game board là lưới các ô vuông đơn vị, cao 18 ô, rộng 10 ô. Thể hiện qua ma trận 2 chiều **grid[10][18]**, với tọa độ (x,y) tính từ góc trên cùng bên trái của màn hình.

3.3 Mô hình hóa

Trong hệ thống của chúng em, các khối tetromino được mô tả dưới dạng ma trận như sau:

```
const char pieces_S_r[2][2][4] =
{ { { 0, 0, 1, 1 }, { 0, 1, 1, 2 } },
```

```
{ { 0, 1, 1, 2 }, { 1, 1, 0, 0 } } };
```

Ví dụ khối chữ "S" hướng bên phải được xác định hình dạng bởi mảng 3 chiều.

- Chỉ số đầu tiên (2) đại diện cho số lần quay của mảnh.
- Chỉ số thứ hai (2) đại diện cho số hàng.
- Chỉ số thứ ba (4) đại diện cho số cột.
- Các giá trị trong mảng này đại diện cho vị trí của các khối trong phần.

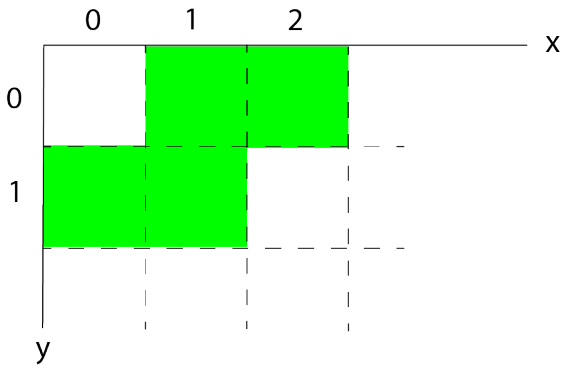


Figure 3.2: Có thể thấy các ô ứng với các tọa độ (x,y) = (0,1); (1,1); (1,0); (2,0)

Số lần xoay, vị trí ô vuông của mỗi hình và hình dạng được quy định theo như các mẫu tetromino sau đây:

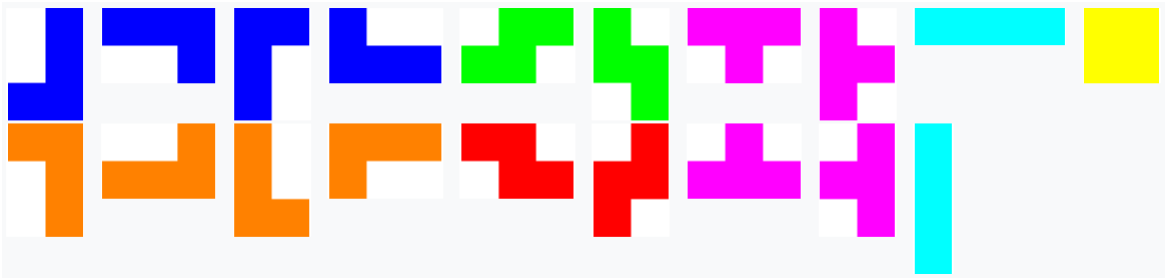


Figure 3.3: Các mẫu hình

Quy định đánh số thứ tự các hình:

- (0) Chữ "L", 4 cách xoay
- (1) Chữ "S" hướng trái, 2 cách xoay
- (2) Chữ "S" hướng phải, 2 cách xoay
- (3) Hình vuông, chữ "O", 1 cách xoay
- (4) Chữ "T", 4 cách xoay
- (5) Thanh thẳng đứng, 2 cách xoay

CHAPTER 4. TRÌNH BÀY SOURCE CODE

Tổng quan source code gồm nhiều hàm, có thể được nhóm lại thành 4 nhóm, mỗi phần bao gồm các hàm đảm nhiệm các chức năng liên quan đến nhau như sau:

4.1 Nhóm function phục vụ vẽ hình

(a) **refresh()**

Chức năng:

- Hàm thực hiện chức năng xóa màn hình, vẽ bố cục, vẽ lưới và cuối cùng vẽ các hình đang có trên gameboard.
- Gọi đến các hàm:
  - drawLayout();
  - drawGrid();
  - drawPiece(currentType, 0, pieceX, pieceY);

(b) **drawGrid()**

```
if (grid[x][y])
    display.fillRect(MARGIN_LEFT + (SIZE + 1) * x,
                     MARGIN_TOP  + (SIZE + 1) * y,
                     SIZE, SIZE, WHITE);
```

Chức năng: Hàm duyệt qua toàn bộ gameboard và nếu một ô cần được tô đầy, nó sẽ vẽ một hình chữ nhật màu trắng có kích thước bằng ô đó.

(c) **drawLayout()**

Chức năng:

- Hàm có nhiệm vụ vẽ bố cục cơ bản của màn chơi, bao gồm đường viền, đường phân cách, tỷ số và quân cờ tiếp theo.
- Gọi các hàm drawNextPiece() và drawText() để vẽ hình sẽ rơi xuống tiếp theo và hiển thị điểm số tương ứng.

(d) **drawPiece()**

Chức năng: vẽ mảnh hiện tại trên màn hình. Nó thực hiện điều này bằng cách lặp qua từng khối trong số 4 khối tạo nên mảnh ghép và vẽ một hình vuông màu trắng cho mỗi khối ở vị trí thích hợp trên màn hình.

(e) **drawNextPiece()**

Chức năng: sao chép vị trí các khối của mảnh tiếp theo vào một mảng nPiece

mới và sau đó vẽ một bản xem trước nhỏ cỡ 2 pixel của mảnh ở góc trên cùng bên phải của màn hình.

(f) **getNumberLength(int n)**

Đầu vào:

- số nguyên n, là số điểm người chơi đạt được

Đầu ra:

- số chữ số của số đó

Chức năng: trợ giúp lấy một số nguyên làm đầu vào và trả về số chữ số trong đó. Phục vụ hàm drawText() để hiển thị số điểm ra màn hình.

(g) **drawText()**

Đầu vào:

- char text[]: là chuỗi ký tự cần hiển thị
- short length: độ dài của chuỗi
- int x,y: vị trí bắt đầu hiển thị

Chức năng: vẽ một chuỗi văn bản trên màn hình ở tọa độ x và y đã cho. Nó lấy văn bản được vẽ, độ dài của nó và tọa độ x và y làm tham số. Chức năng này đặt kích thước văn bản, màu sắc, vị trí con trỏ và phông chữ trước khi vẽ văn bản.

4.2 Nhóm function kiểm tra điều kiện/xử lý khi các vi phạm/ngoại lệ

(a) **checkLines()**

Lặp qua từng hàng của lưới từ dưới lên trên, kiểm tra xem một dòng đã đầy chưa (tất cả các ô đều bị chiếm đóng) hay chưa.
Nếu có, nó gọi hàm breakLine(), chuyển số thứ tự của dòng đó dưới dạng tham số, để xóa dòng đó.

(b) **breakLine()**

Đầu vào:

- line: dòng đã được lấp đầy, cần xóa đi

Chức năng:

- Đầu tiên phát âm báo "xóa" bằng cách sử dụng tone(SPEAKER_PIN), cho biết rằng một dòng đã bị xóa
- Sau đó, nó dịch chuyển tất cả các hàng phía trên dòng đã xóa xuống một ô và xóa hàng trên cùng.

- Thêm 10 vào điểm cho người chơi
- Đảo màu màn hình LED trong 50ms để tạo hiệu ứng hình ảnh của đường đã xóa, sau đó đặt màn hình trở lại bình thường.

Cài đặt:

```
void breakLine(short line) {
tone(SPEAKER_PIN, erase[0], 1000 / erase_duration[0]);
delay(100);
noTone(SPEAKER_PIN);

for (short y = line; y >= 0; y--) {
    for (short x = 0; x < 10; x++) {
        grid[x][y] = grid[x][y - 1];
    }
}
for (short x = 0; x < 10; x++) {
    grid[x][0] = 0;
}
display.invertDisplay(true);
delay(50);
display.invertDisplay(false);
score += 10;
}
```

(c) **nextHorizontalCollision()**

Đầu vào:

- piece[2][4]: dòng đã được lấp đầy, cần xóa đi

Chức năng:

- Chức năng này kiểm tra xem có va chạm nào với lưới bao ngoài theo phương ngang hay không.
- Nó thực hiện điều này bằng cách kiểm tra từng ô trong hình hiện tại và cộng thêm một số tương ứng vị trí x của nó.
- Nếu vị trí x mới nằm ngoài lưới hoặc đã được sử dụng, thì sẽ xảy ra xung đột và hàm trả về true.

Cài đặt:

```
for (short i = 0; i < 4; i++) {
    short newX = pieceX + piece[0][i] + amount;
    if (newX > 9 || newX < 0 ||
        grid[newX][pieceY + piece[1][i]])
        return true;
}
return false;
```

(d) **nextCollision()**

Chức năng: kiểm tra xem có va chạm nào với lưới theo phương thẳng đứng hay không. Nó thực hiện điều này bằng cách kiểm tra từng ô trong phần hiện tại và thêm một ô vào vị trí y của nó. Nếu vị trí y mới nằm ngoài lưới hoặc đã được sử dụng, thì sẽ xảy ra xung đột và hàm trả về true.

Cài đặt:

```
for (short i = 0; i < 4; i++) {
    short y = pieceY + piece[1][i] + 1;
    short x = pieceX + piece[0][i];
    if (y > 17 || grid[x][y]){
      return true;
    }
  }
return false;
```

4.3 Nhóm function sinh hình ngẫu nhiên

(a) **generate()**

Chức năng:

- Gán giá trị cho mảnh tetris tiếp theo sẽ được thả xuống bảng trò chơi. Lấy currentType bằng với nextType, sau đó tạo một giá trị nextType mới bằng cách sử dụng hàm random().
- Nếu mảnh hiện tại không phải là mảnh "O" (mảnh "O" có giá trị type = 5), thì giá trị tọa độ x của mảnh được đặt thành một số ngẫu nhiên trong khoảng từ 0 đến 8.
- Còn đối với mảnh "O", giá trị tọa độ x được đặt thành một số ngẫu nhiên trong khoảng từ 0 đến 6.

- Giá trị tọa độ y được đặt thành 0, cho biết mảnh đó sẽ bắt đầu rơi từ góc trên cùng của gameboard xuống.
- Gọi hàm copyPiece() để sao chép hình tương ứng vào mảng lưu trữ hình tetris.

(b) **copyPiece()**

Chức năng:

Điền vào các khối thích hợp cho một hình tetris và với số lần xoay vòng nhất định. Gán các tọa độ tương ứng với nhau:

```
for (short i = 0; i < 4; i++) {
    piece[0][i] = pieces_L_l[rotation][0][i];
    piece[1][i] = pieces_L_l[rotation][1][i];
}
```

(c) **getMaxRotation()**

Chức năng: trả về số lần quay tối đa mà một loại chi tiết nhất định có thể có. Nó trả về 2 cho loại 1, 2 và 5, 4 cho loại 0 và 4, 1 cho loại 3 và 0 cho bất kỳ loại nào khác.

(d) **canRotate()**

Chức năng: kiểm tra xem mảnh hiện tại có thể được xoay theo lượng xoay hay không. Nó thực hiện điều này bằng cách gọi copyPiece() để tạo một mảng mảnh mới với các khối được xoay, sau đó gọi nextHorizontalCollision() để kiểm tra xem mảnh được xoay có trùng với bất kỳ khối nào ở trên gameboard hay không.

4.4 Hàm setup() và loop()

Thực thi các chức năng sau:

(a) **Hàm setup() khởi tạo các cài đặt quan trọng**

Khởi tạo các chân đầu vào và đầu ra khởi tạo và xóa màn hình OLED, đồng thời đặt xoay dọc màn hình. Sau đó, nó sẽ hiển thị logo trong hai giây trước khi xóa màn hình và gọi hàm drawLayout() để vẽ bố cục trò chơi. Cuối cùng, nó đặt hạt ngẫu nhiên, tạo mảnh tetris đầu tiên và bắt đầu bấm giờ.

(b) **Hàm loop() lặp đi lặp lại trong suốt quá trình hệ thống hoạt động**

- Kiểm tra hình đã chạm đỉnh trên hay chưa
  Nếu nextCollision()=true và tọa độ theo trục y của mảnh tetris pieceY=0 thì khi này đã có mảnh vượt quá đỉnh trên. Khi này, màn hình sẽ hiện ra

thông báo đã thua và tự động chơi lại màn mới. Reset lại tất cả điểm và trạng thái trước đó.

- Điều khiển khối tetris theo tín hiệu nhận vào từ button
  Ví dụ khi nhận được tín hiệu đi sang phải. Còi báo sẽ được bật lên trong 100ms để xác nhận đã nhận được yêu cầu đi sang phải. Sau đó ta gọi hàm nextHorizontalCollision() để kiểm tra xem khi đi sang trái có gặp chướng ngại vật (thành, block đã đầy) hay không.

```
if (!digitalRead(right)) {
    tone(SPEAKER_PIN, click[0], 1000/click_duration[0]);
    delay(100);
    noTone(SPEAKER_PIN);
    if (b2) {
      if (!nextHorizontalCollision(piece, 1)) {
        pieceX++;
        refresh();
      }
      b2 = false;
    }
  } else {
    b2 = true;
  }
```

Tương tự với các chức năng sang trái, đi nhanh, xoay tròn.

CHAPTER 5. KẾT QUẢ VÀ NHẬN XÉT

5.1 Kết quả

Hình ảnh kết quả chương trình chạy tại nhà trước buổi báo cáo.

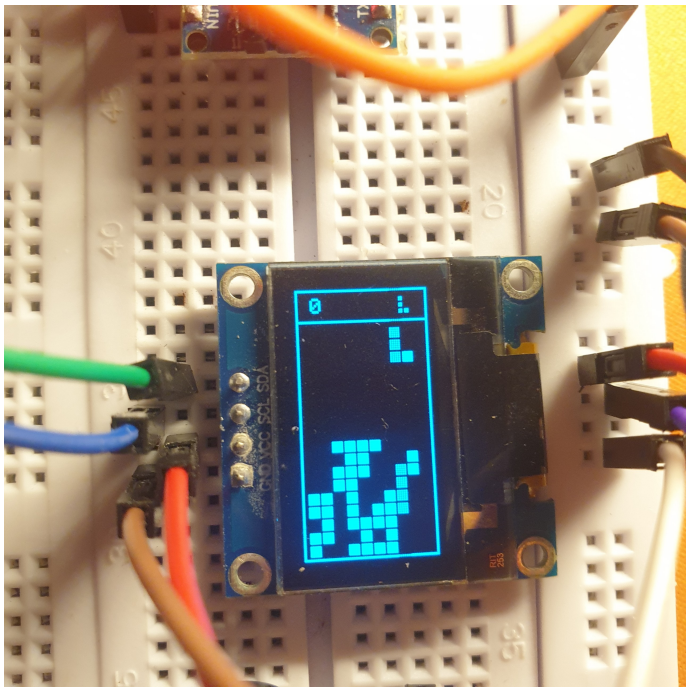


Figure 5.1: Hình ảnh màn hình trò chơi.

Chương trình chạy ổn các chức năng nhóm đề ra:

- Hoạt động liền mạch, ít bị chậm, lỗi.
- Hình khối xuất hiện liên tục, ngẫu nhiên. Có thể điều khiển trái, phải, xoay, rơi nhanh bằng các nút bấm ngoài.
- Không vi phạm các trường hợp ngoại lệ.
- Khi chạm đỉnh trên sẽ thông báo hết màn và chuyển sang màn mới
- ...

5.2 Lời kết

Dù đã rất cố gắng để hoàn thiện bài tập, tuy nhiên hiểu biết của chúng em còn hạn chế và đôi khi, có thể sẽ có một số lỗi kiến thức, hoặc cách sử dụng code chưa tối ưu xuất hiện trong bài làm.

Chúng em mong rằng sẽ nhận được những góp ý quý giá từ thầy và các bạn để có thể hoàn thiện project tốt hơn và rút ra những kinh nghiệm sâu sắc cho những bài tập sau này. Một lần nữa chúng em chân thành cảm ơn thầy đã rất tận tâm trong quá trình giảng dạy và cho chúng em những lời khuyên vô cùng bổ ích.