Proceedings of the 3rd International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/6/20230864

Design and implementation of STM32 based push box game

Weihang He

Leeds College, Southwest Jiaotong University, Chengdu, China, 610031

el19whh@leeds.ac.uk

Abstract. Over the past decade, China's gaming industry has continued to show stepwise growth. There are over 370,000 gaming-related enterprises in China, and over 20,000 gaming-related enterprises were established last year, but there is still a gap between the level of gaming development in Europe, America and Japan. The importance of portability for game carriers needs no further explanation, people can enjoy the joy of games anytime and anywhere. The project is to design an STM32 based Sokoban game that is portable, educational, and can be played anywhere and anytime by players. This paper introduces the technology and methodology of developing a STM32-based Sokoban game. Sokoban is a classic game that can be enjoyed by young and old alike, which is both brain exercise and physical and mental pleasure. The system uses the STM32L476RG board as the hardware platform, based on the Keil software development system and the C language, to complete the game design and implementation. Players can control the game through remote sensing and display. The game is accompanied by background music and game sound effects.

Keywords: STM32, Keil, Game Design, Sokoban, Joystick.

1. Introduction

With the rapid replacement of electronic products, portable games are popular, and there is a wide market for portable games developed based on various platforms. But there are many factors that must be considered when developing a game, such as development cycle, development cost and security [1]. The STM32 series has the advantages of fast speed, low power consumption and reliability. Sokoban is a classic game that can exercise logical thinking skills. The game is set in a small space and requires the wooden box to be placed in a designated location. It is likely that the box will not be able to move or the channel will be blocked, so the limited space and channel need to be cleverly used, and the order and location of movement need to be reasonably arranged in order to complete the task successfully. In this paper, we hope to illustrate the game development process and method based on Keil software with hardware circuit by implementing the design of the push-box game on STM32L476RG board.

2. General functional design of the game

First run the game and load the corresponding map data, a box-pusher appears on the screen, surrounded by pre-made maps, fences, passages where people can walk, several moveable boxes and the target location where these boxes will be placed at the end of the game victory. Players first select the level they wish to reach through the function buttons, and then manipulate the position of the box-pusher worker through telemetry. When the workers are close to the box, and move towards the box position,

DOI: 10.54254/2755-2721/6/20230864

the box can be moved without being obstructed by other objects (box or wall) in the direction of movement, and the sound effects when pushing the box are fed back in real-time. When all the boxes are moved to the designated placement position, the game will be passed and enter the next level. In the course of the game, if you encounter a deadlock, you can press the back button to return to the initial interface of the game. When the player wants to choose another level, press the button to return to the main interface. No matter what state the system is in, the action of returning to the main interface can be realized, and after returning to the main interface, you can choose another level to play.

3. Circuit design of the system hardware

The system hardware circuit is mainly composed of a power supply, STM32L476RG development board, N5110 LCD, joystick, buzzer, buttons, and other parts.

3.1. Main display module

In this paper, the LCD of the Nokia 5110, which is an 84×48 pixel monochrome display, is used to drive the display by using a Philips PCD8544 LCD controller IC [2]. It is shown in Figure 1.

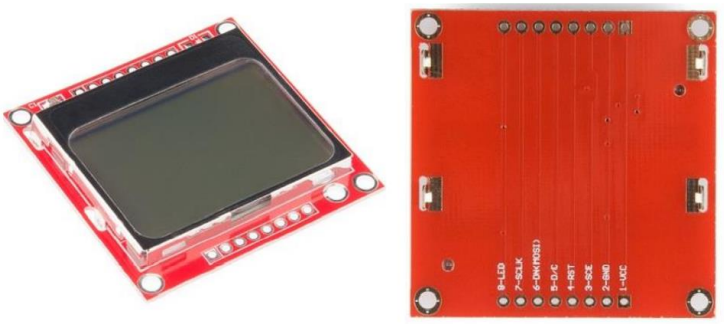

Figure 1. LCD.

Also, the LCD uses the serial peripheral interface (SPI) for communication, so specific pins with SPI peripherals must be used.

The N5110 LCD is connected to the STM32L476RG board as shown in Figure 2.

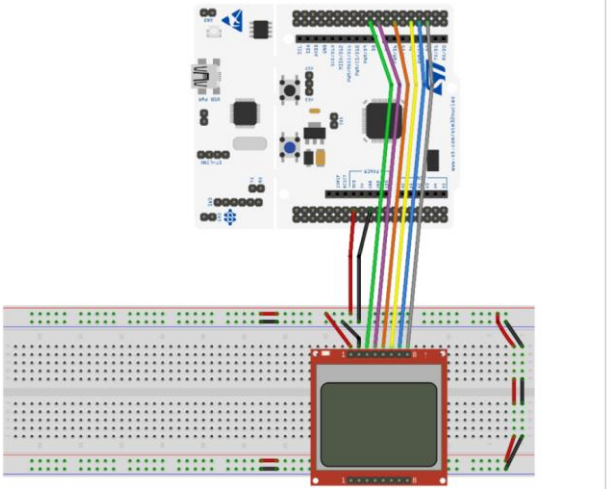

Figure 2. Connection Mode.

3.2. Joystick control module

The joystick is used in microcontroller applications to implement the functions of entering data and transmitting commands. It is the main means of manual intervention in the system.

DOI: 10.54254/2755-2721/6/20230864

The joystick is composed of two potentiometers and a button. Moving the joystick horizontally changes the position of the horizontal potentiometer. And, moving the joystick vertically changes the position of the vertical potentiometer. Clicking the joystick presses a button. The button has an external pull-up resistor. The wiring method of the joystick is shown in Figure 3.

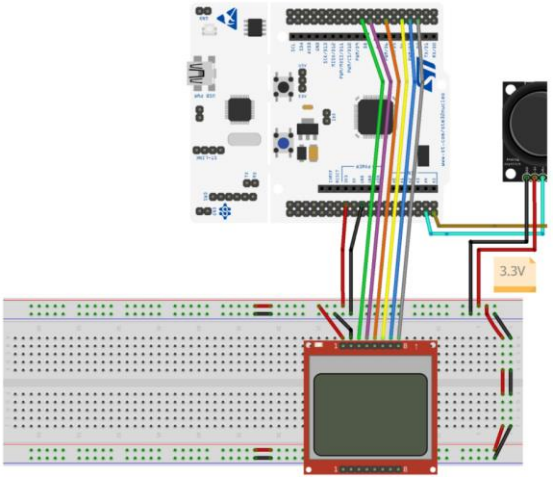

Figure 3. Connection Mode.

The directions of the joysticks are held in an enum named Direction. Each of the directions is assigned an integer value and given a label that can be used to make the source code more readable (CENTRE, N, NE, E, SE, S, SW, W, NW) [3]. Then we can make the software know what the command from the joystick is with the following code.

Direction d = joystick.get_direction();

3.3. Audio playback module

The passive buzzer is a very common electronic component, and the electronic candle that can play birthday songs in life is played by a passive buzzer. The passive buzzer does not have an internal oscillation source and can play different tones of sound by using a PWM wave driven by a frequency of 500Hz to 4.5kHz with a 50% duty cycle [4]. Each time the remote sensing or button control is used during the game, there will be real-time feedback of different sound effects, and after successfully passing the game, a small piece of music will be played, which is achieved with the help of a PWM buzzer. As shown in Figure 4 PWM buzzer layout close-up.

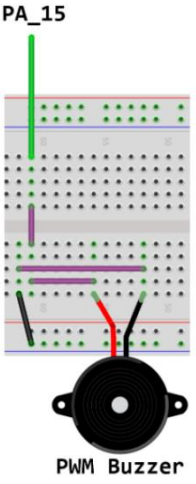

Figure 4. Connection Mode.

DOI: 10.54254/2755-2721/6/20230864

4. System software design

4.1. Soft overall framework construction

The entire software is divided into functional modules as follows: ① initialization - mainly the LCD screen display content and the initialization of the joystick sensing orientation; ② options - choose to enter the game, view the rules of the game or enjoy a piece of pure music [5]; ③ Initialization of the game map and outputting information on the screen about spaces, walls, boxes, destinations, composite objects, characters, etc., depending on the difficulty of the level selected.; ④ Enter the game cycle, and wait for a signal from the buttons or joystick. When a signal is received from the joystick to go up, down, left or right, the relevant action needs to be performed on the villain in the game; or a signal is received from the back button to return to the previous screen. If a signal is received to reset the level, return to the beginning of the level (used when the box is pushed into a dead end and enters a dead game), ignore when an invalid signal is received; each time a valid signal is received it is accompanied by a different background sound ⑤ Determine if the game is passed, i.e. the number of boxes is the same as the number of destinations, so the function performs an operation to determine if all boxes are placed, i.e. all move to the destination. Detects if all boxes in the map have reached their destination, if not, returns to the loop to continue the game, otherwise, successfully passes the [6] level [6]. After passing the game, the system will automatically play a short piece of music to celebrate the passing of the game. This returns the game players to the main screen where they can choose whether to continue with a different level of difficulty [7].

4.2. Implementation of the core

After the overall framework has been built, the core of the entire game will be implemented next. First, the core data structure of the game can be designed (including how to access these data), then the implementation algorithm will be designed according to the characteristics of the game, and finally the algorithm will be integrated into the various aspects of interaction with the user.

4.2.1. Game interface design. The display is programmed using the N5110 class developed by Dr. Craig Evans. Encapsulates all the SPI commands into a series of well-defined and simple high-level methods allowing:

- pixels to be set and cleared
- text strings to be printed
- primitive graphics to be drawn (shapes and sprites)

Therefore, in the main screen, we only need to print the options for the player to make the next selection, including, Start Game (to make the level selection), Game Rules (to learn the rules of the game), and Pure Music (to enjoy a small piece of stored music through the buzzer). This is shown in Figure 5.

Figure 5. Main Interface.

DOI: 10.54254/2755-2721/6/20230864

4.2.2. Design of maps and internal structure data. The display is a monochrome display with 84 x 48 pixels. So the design defines each frame size as 6x6 pixels, and divides the display into 112 frames, 14 frames across, and 8 frames down. Define a two-dimensional array map_define for map definition, define map to store the map of each level, used to record the state of each point on the screen, where, "0" means empty space; "1" means wall. "3" indicates a destination; "4" indicates a box; "6" and "9" indicate workers. "7" indicates that the box is placed at the destination. The display image of each element type was previously designed by using the N5110 class, and finally the desired map was printed by looping to detect the type of element in each bit of the two-dimensional array, and then replacing the array elements with the display image[8]. This is shown in Figure 6.

```
int map2[14][8] =
{
{0, 0, 0, 0, 0, 0, 0, 0},
{1, 1, 1, 1, 1, 1, 1, 1},
{1, 0, 0, 0, 0, 0, 1, 1},
{1, 0, 4, 0, 0, 4, 0, 1},
{1, 1, 3, 1, 0, 1, 6, 1},
{1, 1, 3, 1, 0, 4, 0, 1},
{1, 1, 3, 3, 4, 0, 1, 1},
{0, 0, 3, 3, 1, 0, 1, 1},
{0, 1, 3, 3, 1, 0, 1, 1},
{0, 0, 1, 4, 4, 4, 4, 0},
{1, 0, 0, 0, 1, 0, 1, 0},
{1, 1, 1, 0, 0, 0, 0, 0},
{1, 1, 1, 1, 1, 1, 1, 1},
{0, 0, 0, 0, 0, 0, 0, 0}
    };
```

Figure 6. Game Map.

4.2.3. Algorithm design for object movement. Most of the games are played by interacting with the user and doing the corresponding arithmetic processing according to the specific situation, and the user always interacts with the joystick during the whole game of Sokoban. The following is an analysis of the game's characteristics from different perspectives. Suppose: At a certain moment in the game process, the worker is in a certain area. At this time, the game receives the user's joystick action instructions, and determines the action should make the response according to the game rules. The response is to make the worker perform the movement instructed by the user without violating the game rules, and not to respond to instructions that violate the game rules. All the situations that the worker will encounter are analyzed in order to generalize the algorithm of the action.

1. The worker moves in the direction of an open space
2. The worker moves in the direction of the wall
3. The worker moves in the direction of the box, and the box moves in the direction of the open space
4. The worker moves in the direction of the box and the box does not move in the direction of the open space (and the wall or the box)

For a two-bit array, the movement of the worker is shown as a change in the position of the element representing the worker in the two-dimensional array, adding the worker to the right, and adding the element representing the worker to the element to its right results in 6 or 9 (depending on whether the right element represents an empty space or a destination) [9].

As long as the objects according to the preamble can determine whether the worker can move (move in the direction of the instructions), the position state data after moving only changes two areas; while the fourth case is the front of the worker for the box, we need to judge the state in front of the box in depth to determine whether the worker can move. Therefore, in order to fully determine whether the operation instruction is executable or not, it needs two position state values in front of the worker, and the position state data changes three areas after the movement. And it needs to determine the position state to derive the execution action.

4.2.4. Background sound code design. First, we have to predefine the frequencies needed for the different tones emitted by the buzzer. Then we use the mbed library functions to create functions for

DOI: 10.54254/2755-2721/6/20230864

sound effects of different durations. When we finally use the sound effect, we only need to call the frequency we designed before to emit the different tones, and then select the function to play the sound effect corresponding to the desired duration [10].

In the course of the game, whenever the signal from the joystick or button is detected, the function to play the sound effect will be called. The music that will be played automatically after the final pass is the Ode to Joy, composed by Beethoven. By calling the previously designed play, different durations of the sound effects are used to achieve the ode of joy.

5. System testing and results

(1). Enter the game or return to the main interface through the joystick and buttons; (2). Move up, down, left, and right with the joystick. (3). the character touches the box, the character and the box together to push forward one frame, and can only move forward one frame in the direction of the character, until the advancement of the gold position, can not be moved; (4). the game process with background music playing and game sound effects with; (5). after passing the game can continue to the next difficulty.

6. Conclusion

The STM32-based game development has the advantages of flexible design, fast speed, safety and reliability, and reusability. At the same time, several independent games can be integrated on the same hardware platform, which is scalable. In this paper, the STM32 series development board is used as the hardware platform, and the C language is used for the development of the classic Sokoban game, which combines the joystick and peripherals such as display and buzzer. The test results show that the game can run smoothly, the functions can achieve the expected goals, and it can be easily debugged and improved.

Although the design allows the player to play the game normally, if the player encounters a level they cannot pass, they will lose interest in the game, so I think we should add intelligent guidance to the game to help the player pass the level. Also, as the map for this game was written and generated by humans, the result is that the number of levels is limited, and the player cannot play the game continuously. Therefore, in future research I may focus on the intelligent generation of the Sokoban map algorithm, so that the program itself can generate maps for the player to play by calculation, and can distinguish the difficulty level of the map by itself, which is the direction of future research.

References

[1] Fenyang. (2022). China's game industry development status and industry development tre nd analysis in 2021. Online: https://www.chyxx.com/industry/1108699.html

[2] LittleAshes. (2019). Nokia5110 LCD. Online: https://blog.csdn.net/weixin_41746317/article/ details/102770800

[3] Wang Chao, Zhao Qi. (2015). Wireless transmission program design of joystick signal b ased on microcontroller. Mechanical Engineering and Automation, 2015(1), 180-181, 1 84.

[4] Fred-66. (2018). STM32 Controlling a passive buzzer to play music using the STM32F1 03. Online: https://blog.csdn.net/fanxp66/article/details/80264700

[5] Yang Haojie. (2013). Analysis of game interface design principles. Digital Technology a nd Applications, 2013(1), 138-138.

[6] Huang Jinqiu, Huang Huihong. (2010). A new type of push-box game design based on microcontroller. Microcomputer Applications, 2010(5), 32-34.

[7] A. Botea, M. Muller, and J. Schaeffer. (2003). Using abstraction for planning in Sokoba n. In J. Schaeffer,M. Muller, and Y. Bjornsson,editors, Computers and Games, volum e 2883 of Lecture Notes in Computer Science, 360–375.

[8] Xu Huawei,Lin Fan. (2019). Design and implementation of FPGA-based push-box game. Electronic Production, 2019(19), 24-26.

[9] Zhao Qiaoni. (2017). Design and simulation of a PROTEUS-based push-box game. Auto

DOI: 10.54254/2755-2721/6/20230864

mation Technology and Applications, 2017(7), 141-144.
[10] Ma ZQ, Wang JG, Sun Shaolin. (2012). STM32-based PWM music player application d esign. Microcontroller and embedded system applications, 2012(11), 63-65.