# Retro Classic Game Machine

# Design Report

## Create Date：2021-05-31

## Version：V1.0

Second Group
in Class of Microprocessors and Embedded Systems

# Revision History

| Date | Version | Content Modification | Duty Members |
|---|---|---|---|
| 20210308 | 0.2 | Needs identification, General framework | All team members |
| 20210315 | 0.3 | Components List, Hardware Architecture, Appliance Scenarios, Demand Analysis | Ruiying Zou |
| 20210317 | 0.4 | Abstract | Lei Xu |
| 20210411 | 0.5 | Hardware Architecture | Wenchong Sun |
| 20210412 | 0.6 | Software Architecture | Ruiying Zou |
| 20210413 | 0.7 | The Key Code | Chen Pan |
| 20210414 | 0.8 | Realization Test | All team members |
| 20210415 | 0.9 | Summary, Prospect and Reference | All team members |
| 20210531 | 0.9.9 | Format modification | All team members |
| 20210702 | 1.0 | Article coherence modification, Article details revision | All team members |

# Abstract

Currently there are more and more games in our daily life, but most of them will entice users to waste a huge amount of time and money. In order to make the game achieve its original purpose that reinforcing intelligence and entertainment, this paper introduces a retro classic game machine based on Raspberry Pi Zero W. This game machine is equipped with a Recalbox system, which makes it easy to play a variety of classical games. In addition, it also has a snake game written in assembly language, which is simple and easy to operate.

**Key Words:** Raspberry Pi Zero W, Game machine, Recalbox, Embedded system.

# Content

# List of Diagrams

# Acronym Table

| Acronym | Full Name | Full Name in Chinese |
|---|---|---|
| USB | Universal Serial Bus | 通用串行总线 |
| DC | Direct Current | 直流电 |
| GND | Ground | 接地 |
| SD CARD | Secure Digital Memory Card | 安全数码储存卡 |
| C64 | Commodore 64 | Commodore 64 |

# I. Basic Information of the Team Members

**Table 1 Basic Information of the Project**

| Project Name | Retro Classic Game Machine | | |
|---|---|---|---|
| Members | Name | College | Division of Labor |
| | Lei Xu | SICE | Software and hardware |
| | Chen Pan | SICE | Software and hardware |
| | Wenchong Sun | SICE | Software and hardware |
| | Ruiying Zou | SICE | Software and hardware |

# II. Overview

## 2.1 Problem Analysis

Classic games never go out of style. Nowadays, there are more and more mobile games on the market, but they are all in the fixed pattern of drawing cards and charging money, which wastes a lot of money and runs counter to the original intention of making games. Our group plans to assemble a game machine based on embedded devices and develop a mini game on it, so that players can relax moderately after learning. In order to achieve our goal, we need to accomplish the connection and test of all components, the design and assembly of the shell and the development and debugging of the mini game. In addition, the mini game designed by us is carried out with 8bit flavor, to give users experience of early years.

## 2.2 Introduction of the Game Machine

Our game machine runs on the Raspberry Pi Zero W platform with Recalbox system inside, which is a simulator that supports many retro computer platforms. You can choose discretionary platforms and experience many kinds of games in the past century. This machine is small and easy to be carried, so that you can start your game at any time. The following paragraph shows how to use this game machine.

In addition, Figure 1 is the overview of our game machine.

**Start Your Game**

**1. Plug in your game machine**

The game machine has two ports. Find a cable to connect your power bank to your game machine by the Micro-USB port and a 12v power adapter to the monitor. Then your game machine will start automatically.

**2. Initialize your game pad**

The first time you turn the machine on, you have to initialize the configuration of gamepad, so don’t forget to plug your gamepad into the USB port on the side of the machine. Follow the instructions on the screen and finish the key map.

**3. Start to play**

After the game machine is launched, you will see an interface which shows you many game platforms and you can use your game pad to choose one. Then you will enter into a game-selecting interface. Now choose a game and start to play!

**4. Play the original game made by our team**

We have designed a retro game called SNAKE on the Commodore 64 platform, which is popular in the 1980s. If you want to have a try, please do the following steps.

a) Download snake.prg (the snake program) and snake.prg.p2k.cfg (the key config) files from https://github.com/fumiama/c64-snake and move them into SHARE/roms/c64 directory on the EXTRA partition of the SD card.
b) Plug in the SD card and start the system by plug in power supply.
c) You will find the game named “SNAKE” in the Commodore 64 platform.
d) Select it like any other games and the initial screen will be shown.
e) Press any key to start the game.

f) Press up/down/left/right on the game pad or W/S/A/D on the keyboard (if you have one) to control the snake to eat the food.

g) If the snake rush into the wall surrounds or itself, the game will be over.

h) If you want to terminate directly, you should press start & select button at the same time.

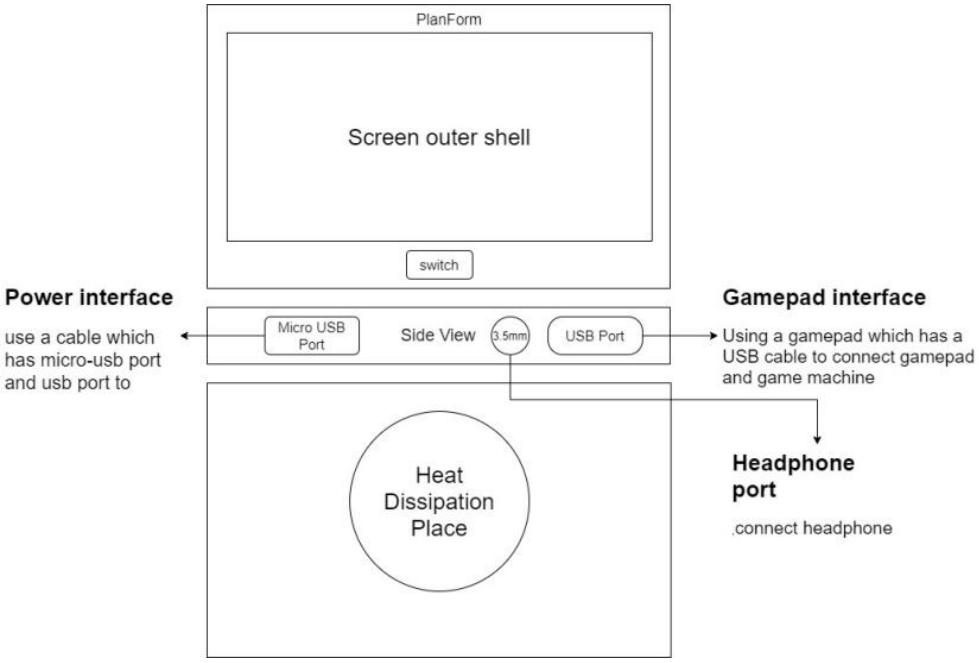


**Figure1 Overview of game machine**

# III. Design

## 3.1 Overall

In part3, the overview design of the retro game machine will be introduced, from the aspect of hardware (3.1.1) and software (3.1.2). The hardware design shows the components of the game machine while the software design illustrates the software SNAKE made by our team and the flowchart of it.

The processor module can be found in the section 3.2. The display module can be found in the section 3.3. The power section and the control module lie in the section 3.4 and section 3.5 respectively.

### 3.1.1 Hardware Design

This section illustrates the outline of the game machine's hardware design and briefly states the function of each module.

The hardware design contains three parts, which are the memory part, the processor part and the I/O (input & output) part. The processor part is responsible for processing demand and executing computing tasks. The memory part is in charge of storing the operation systems and data. The I/O contains different kinds of external devices and is responsible for the transaction of input and output signals.

Each module mentioned in the overview can be classified into one of those three parts. The processor module belongs to the processor part and the memory module belongs to the memory part. The other modules all can be classified into the I/O part.

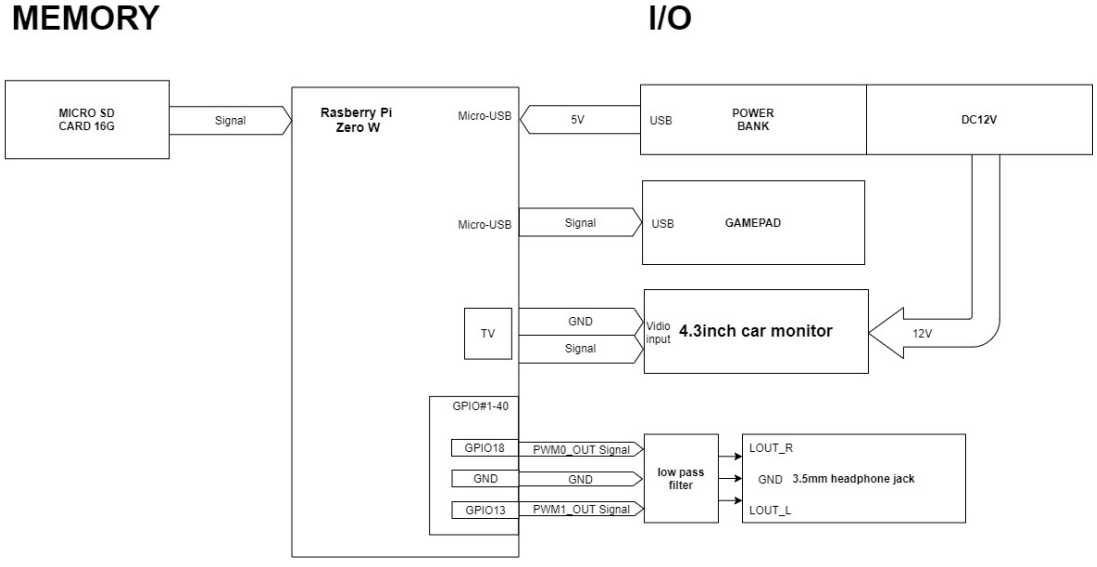


Figure 3-1 Hardware architecture

### 3.1.2 Software Design

This part introduces the operating system "Recalbox" and commodore64 emulator we used on the game machine. Besides, it also contains the introduction of the game SNAKE we developed in the game machine and the developing tools used during the software development.

In the section 3.1.2.1, a specific introduction of the Recalbox operating system is given. The section3.1.2.2 introduces the commodore64 and the section 3.1.2.2 contains an introduction to the game design.

#### 3.1.2.1 Recalbox Operating System

The Recalbox is an operating system based on Raspbian. It is an operating system which contains many retro computer's emulator, such as commodore64, windows x and so on. These emulators are capable of running old software that used to run on the corresponding real machines. The Recalbox is designed to allow people today to play retro games, hence the many games already included in the emulator above. For our project we completed the development of the snake game on windows/mac and chose the commodore64 emulator in Recalbox to load the game into.

The Recalbox operating system is first loaded into the Micro SD card and some configuration file has to be revised. Finally, the game can be loaded into the commodore64's emulator which has been installed in the operating system.

#### 3.1.2.2 Commodore64 Emulator

Commodore64 emulator is a general term of the emulator of the commodore64. It can run the software which used to be run on the real commodore64. Software that can be run successfully on the commodore or on its emulator needs to be in the PRG format. The PRG file can be generated by assembly language. [1]

### 3.1.3 Game Design

We develop a game named SNAKE, which is programmed by the assembly language. It is loaded into the operating system stored in the Micro SD card. So, player can find the game in the commodore64 emulator mentioned above and enjoyed it.

The flow chart in Figure3-2 illustrate the flow of the game. When the game is executed, the information of the snake, score, direction and other parameters will be initialized as well as the game interface. This is the initial step. Then the game machine will wait for an input signal. Any inputs will start the game. Then the judge part will judge whether the snake hit itself or the wall. If it hit something, the game is over and shows fail's warning. If it doesn't hit anything, then judge whether the snake has eaten the food. Then the snake will move one grid depended on the parameter "d". If its head eats the food during its moving, its length will add. The snake moves once and the score increases by one point. Once the steps above have been completed, the program will check the value of the input signal again. If the input signal is the direction we have define, the program will return to the position where the first judgement was made, or the game will be paused or exited.

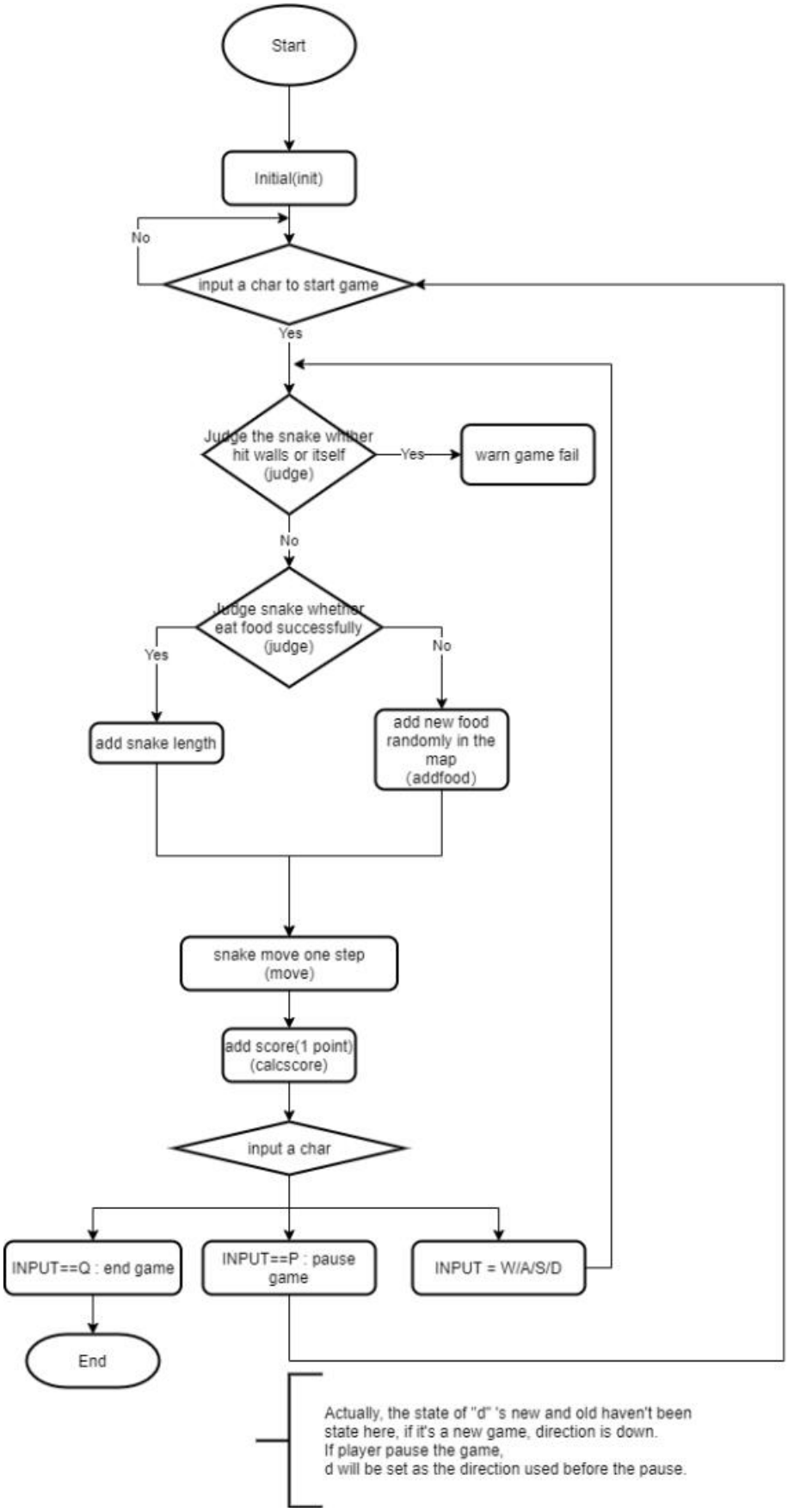


**Figure 3-1-2 Game Flow Chart**

### 3.1.4 Software Development Tool

The software needs to be done on our current computers, as computers like the commodore64 are no longer available. We use HoxsC64 64 Emulator (3.3.1.2) and OPHIS Assembler (3.1.2.1), two software that can be run on us computer, to develop and test the game.

#### 3.1.4.1 Simulation Platform

The software development has to be finish on our computers because of the elimination of commodore64. We use HoxsC64 64 Emulator (execution document) as our simulation platform, which can be used on Window and Mac. The emulator simulates the hardware environment of the Commodore 64. It can load the PRG file, which is our game's file type. So, the test of the game can be done on it.

#### 3.1.4.2 OPHIS Assembler

We choose OPHIS Assembler [2] as our assembler, which can assemble assembly language into machine code and generate PRG file. It's a cross-assembler for the 65xx series of chips. It supports the stock 6502 opcodes, the 65c02 extensions, experimental support for the 4502/4510 used in the Commodore 65 prototypes, and syntax for the "undocumented opcodes" in the 6510-chip used on the Commodore64. It's written in Python and can be used in Python environment. We can use the instructions of OPHIS to generate PRG file.

## 3.2 Processor Module

The processor module is used to process the computing task and complex orders. We use Raspberry Pi Zero W as our processor, which can support Recalbox system installed on it. The operating system "Recalbox" is stored in Micro-SD card and it can be read by the Raspberry Pi. The Raspberry Pi can output the video signal and transmit it to the screen through the TV contact. The detailed information of the GPIO and contact of the Raspberry Pi are displayed in the Figure3-2-1.

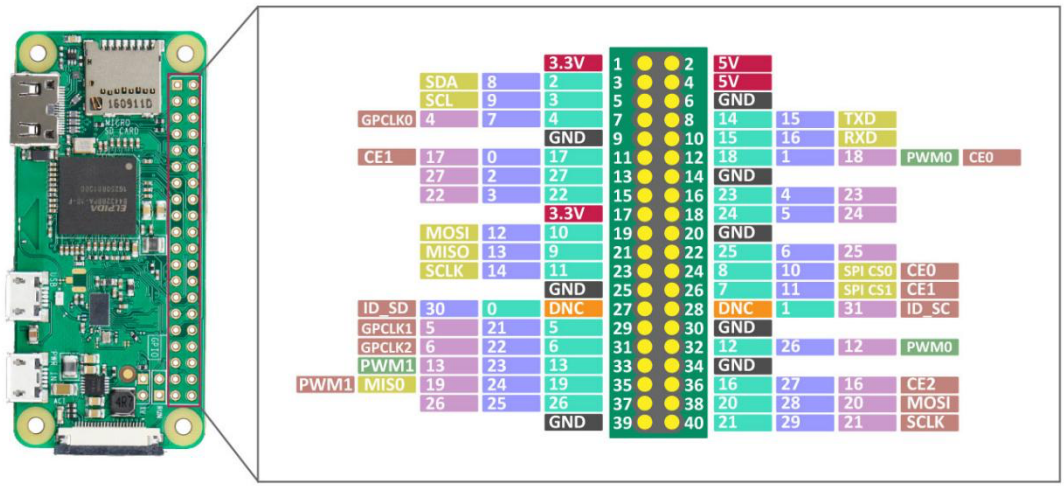


**Figure 3-2-1 GPIO and contact information**

## 3.3 Display Module

The display module is to display the image of the game machine. The display screen uses the car monitor, whose size is 4.3 inches. It can show colorful image. It has to be provided with 12V power adapter. The video signal outputs from the TV interface of Raspberry Pi Zero W and then inputs into the video interface of the car monitor.

## 3.4 Power Module

The power module is used to provide energy to the components in the game machine. We use 12V adapter to charge the car monitor. A power bank that can provide 5V DC power to the Raspberry Pi is used.

**Table 2 Power Supply Information**

| Name | Volt |
|---|---|
| Raspberry Pi Zero W | 5V |
| Car Monitor | 12V |
| Power Bank | 5V |

## 3.5 Control Module

Control module only has one component which is the gamepad. The gamepad is used to control the game machine. Figure3-5-1 shows the picture of the game pad. Figure 3-5-2 illustrates the function of the buttons on the gamepad.

# IV. Test Results

We test car monitor, the installation of the Recalbox system on the Raspberry Pi Zero W, the installation of the snake on the commodore 64 emulator, the initial screen of SNAKE, the end screen of the SNAKE, up, down, left, right and down buttons on the gamepad. The following section shows the results of our tests in detail.

1. The test results of the car Monitor are shown in Figure 4-1. The Figure 4-1 (a) is when the car monitor is unpowered, and the Figure 4-1 (b) is when it is powered up.


**4-1(a)** **4-1(b)**

**Figure 4-1 Test results of the car monitor**

2. The boot screen of a successful installation is shown in Figure 4-2.

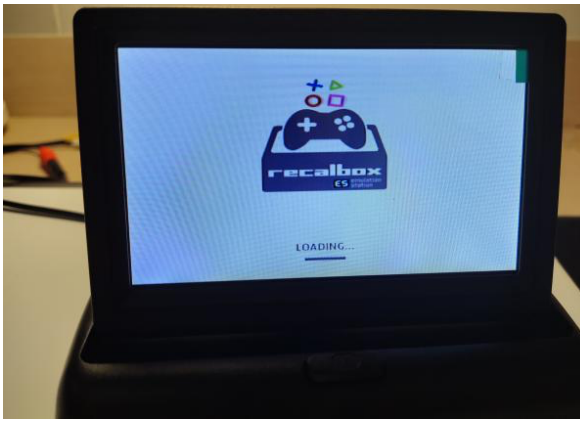


**Figure 4-2 The boot screen after successful installation of the OS**

3. As shown that Figure 4-3 is proves that the snake game has been successfully installed in the C64 emulator of the snake on the C64.


**Figure 4-3 Successful installation of the snake in C64 emulator**

4. The starting screen of Snake is shown in Figure 4-4. It displays completely score, wall, snake, food and a prompt that tells you how to start the game.


**Figure 4-4 The start screen of the snake game**

5. The end screen of Snake is shown in Figure 4-5.

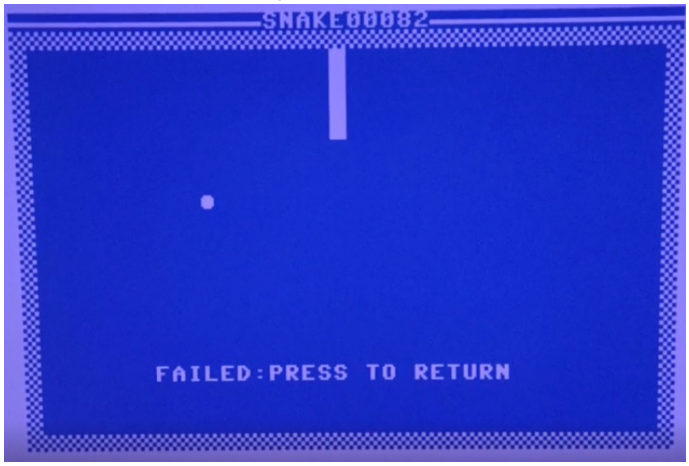


**Figure 4-5 The end screen of the snake game**

6. Test results for the down button on the gamepad are shown in Figure 4-6. Figure 4-6 (a) shows the result before pressing the down button, and Figure 4-6 (b) shows the result after pressing the down button.

**4-6(a)**

**4-6(b)**

**Figure 4-6 Test results for the down button on the gamepad**

7. Test results for the right button on the gamepad are shown in Figure 4-7. Figure 4-7 (a) shows the result before pressing the right button, and Figure 4-7 (b) shows the result after pressing the right button.

4-7(a)

4-7(b)

**Figure 4-7 Test results for the right button on the gamepad**

8. Test results for the up button on the gamepad are shown in Figure 4-8. Figure 4-8 (a) shows the result before pressing the up button, and Figure 4-8 (b) shows the result after pressing the up button.

4-8(a)

4-8(b)

**Figure 4-8 Test results for the up button on the gamepad**

9. Test results for the left button on the gamepad are shown in Figure 4-9. Figure 4-9 (a) shows the result before pressing the left button, and Figure 4-9 (b) shows the result after pressing the left button.

4-9(a)

4-9(b)

**Figure 4-9 Test results for the left button on the gamepad**

In addition, there are some software bugs. For instance, the food will not appear in some particular occasions. If user press a button that isn't be defined, the game will raise an error.

# V. Summary and Prospect

The objective of this project was to make the game achieve its original purpose that reinforcing intelligence and entertainment. The game machine was designed，constructed and commissioned to have an opportunity to play retro gaming. Through the development of the project, we successfully installed the Recalbox system on the Raspberry Pi. We finished the hardware connection between the TV and Raspberry Pi. Then we developed the game SNAKE which can be run on the Commodore 64 and loaded it into the game machine.

As a result of the test program, the following conclusions can be made :

1. Because of the long wait time in the game machine, it can temper people's patience.
2. The game machine is equipped with a Recalbox system, which makes it easy to play a variety of classical games. These games are simple to operate and educational.
3. It is a good choice to reduce software bugs to have a smooth gaming experience. Nevertheless, it can hardly be denied that these software bugs make snake more entertaining.

The following improvements can be made to the hardware and the software to have a smoother gaming experience.

The hardware can be improved in the following ways:

1. Add a 3.5 mm earphone port and a speaker.
2. Add a fan to cool down the machine.
3. Dig holes on the plastic shell to put fan, speaker and other devices.
4. Add a 12v to 5v module.

The software can be improved in the following ways:

1. Improve score calculating algorithm.
2. Add colorful UI.
3. Add difficulty, like speed-up mode or add walls.

# VI. Reference

[1] http://bbs.shumeipaiba.com/thread-59-1-1.html

[2] https://michaelcmartin.github.io/Ophis/book/book1.html

[3] MCS6500 Microcomputer Family, Programming Manual MOS Technology Inc., 1976.

# VII. Appendix

## 7.1 Platform & Price

Table 3 is shown that the platform and price for this design.

**Table 3 Platform and Price**

| components | Unit Price | Quantity | Total Price (With Tax) | Invoice |
|---|---|---|---|---|
| Raspberry Pi Zero W | ¥96 | 1 | ¥116 | Yes |
| Car Monitor | ¥55 | 1 | ¥58 | Yes |
| Concentrator | ¥18.5 | 1 | ¥18.5 | No (self-paying) |
| Speaker | ¥0 | 1 | ¥0 | No |
| Game Pad | ¥10 | 1 | ¥10 | No(self-paying) |
| Micro SD Card | ¥29.9 | 1 | ¥29.9 | Yes |
| total | | | ¥232.4 | |

## 7.2 The Key Code

1. Initialization function, as shown Figure 7-1. It initializes the variables that we need, and prints the initial interface of the game on the screen. Part of the initialization will call functions in subfunction modules.

```
.macro init          ;初始化函数
    lda #147         ; 清屏
    jsr chrout
    lda #$00
    sta s
    sta eat
    sta s + 1        ; 初始化分数为0
    jsr printscore   ; 打印分数
    jsr move_init    ; 移动参数初始化
    jsr printfield   ; 打印蛇，包括边框
    jsr printhint    ; 打印开始提示
    lda #go_d        ; 初始化方向为下
    sta d
    jsr srand        ; 初始化随机数种子
    jsr addfood      ; 增加食物
.macend
```

**Figure 7-1 Initialization function**

2. Food random generation function. It can be seen from Figure 7-2. It uses system time as the random seed, generates food in the reasonable place.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;
; addfood 随机生成食物
;;;;;;;;;;;;;;;;;;;;;;;;;;;
addfood:
.scope
.data
.space seed 6
.text
    lda #<field
    sta _ptr
    lda #>field
    sta _ptr + 1

    `propagate seed, $c3
    `propagate seed+1, $9e
    `propagate seed+2, $26
    `propagate seed, seed+1
    `propagate seed+1, seed+2
    `propagate seed+2, seed+3

    lda seed + 5
    ldx seed + 4
    sta seed + 4
    lda seed + 3
    stx seed + 3
    ldx seed + 2
    sta seed + 2
```

**Figure 7-2 Part of the food random generation function**

3. Judgment function. Functionally, it can be divided into eating judgment, bumping into the wall judgment and bumping into itself judgment. Eating judgment can affect the parameter passed to the movement function. Bumping into the wall judgment and bumping into itself judgment can decide whether the game is over. It is illustrated as Figure 7-3.

```
.macro judgeof
    lda d
    ldy #0
    ;;方向向上
    cmp #go_u
    beq _gup
    ;;方向向下
    cmp #go_d
    beq _gdown
    ;;方向向左
    cmp #go_l
    beq _gleft
    ;;方向向右
    cmp #go_r
    beq _gright
    jmp _End1

_gup:
        `_m_ptr_minus 40
        lda (_ptr),y
        jmp _compare

_gdown:
        ldy #40             ;地址+40
        lda (shead),y
        jmp _compare

_gleft:
        `_m_ptr_minus 1
        lda (_ptr),y
```

**Figure 7-3 Part of judgment function**

4. Movement function. It has two modes of movement. Depending on the parameter passed by the judgment function, it determines which mode the snack will move in. It is described as Figure 7-4.

```
.scope
.data zp
.space _stail 2      ; 蛇尾指针
.space _tmp 1
.text
move:
    lda eat                 ;eat判断
    bne _head_move          ;吃到食物直接进入模式一：head move，否则先执行tail move
    lda #csps
    ldy #0
    sta (_stail),y          ;tail move
    ldy #40
    lda (_stail),y
    cmp #csnk               ;down search
    bne +
    lda #40
    jsr _propergate_tail
    jsr _head_move
    rts
*   ldy #1
    lda (_stail),y
    cmp #csnk               ;right search
    bne +
    lda #1
    jsr _propergate_tail
    jsr _head_move
```

**Figure 7-4 Part of movement function**

5. Go to https://github.com/fumiama/c64-snake to check the whole code.