EIE3810 Microprocessor System Design Laboratory

Lab 6. Mini Project

姚子煊

Student ID: 115010267

2018/05/18

School of Science and Engineering

The Chinese University of Hong Kong, Shenzhen

Spring term, 2017-2018

**Experiment 1**

* **Design**

First of all, we set Page to “0”, where necessary initializations are done for the program. SYSTICK is initialized for the body part of the game, and TIM3 is initialized for JOYPAD use only. Then, TFTLCD and Buzzer are initialized, Keys and their interrupts are initialized, as well as USART1 and USART1 interrupt.

u8 Page **=** 0**;**

int main**(**void**)**

**{**

EIE3810\_clock\_tree\_init1**();**

EIE3810\_TFTLCD\_Init**();**

EIE3810\_KEY\_Init**();**

EIE3810\_Key1\_EXTIInit**();**

EIE3810\_Key2\_EXTIInit**();**

EIE3810\_KeyUp\_EXTIInit**();**

EIE3810\_Key0\_EXTIInit**();**

EIE3810\_USART1\_Init**(**72**,** 14400**);**

EIE3810\_USART1\_EXTIInit**();**

EIE3810\_SYSTICK\_Init**();**

EIE3810\_TIM3\_Init**(**99**,**7199**);** //100Hz 10ms

EIE3810\_Buzzer\_Init**();**

JOYPAD\_Init**();**

EIE3810\_NVIC\_SetPriorityGroup**(**5**);**//Set PRIGROUP

EIE3810\_TFTLCD\_DrawALL**(**WHITE**);**

Delay**(**5000000**);**

Page **=** 1**;**

Figure 1. Page 0: Initialization

Then, we enter welcome page, print the welcome sentences and enter the next stage.

**if** **(**Page **==** 1**){** //Welcome Page

printString2412**(**100**,** 100**,** "Welcome to mini Project!"**,** WHITE**,** BLUE**);**

Delay**(**5000000**);**

printString**(**120**,** 150**,** "This is the Final Lab."**,** WHITE**,** RED**);**

Delay**(**5000000**);**

printString**(**120**,** 200**,** "Are you ready?"**,** WHITE**,** RED**);**

Delay**(**5000000**);**

printString**(**120**,** 250**,** "OK! Let's start."**,** WHITE**,** RED**);**

Delay**(**5000000**);**

Page **=** 2**;**

**}**

Figure 2. Page 1: Welcome Page

We enter Difficulty Level Page, print the selecting sentences.

**if** **(**Page **==** 2**){** //Difficulty Level Page

EIE3810\_TFTLCD\_FillRectangle**(**0**,** 480**,** 0**,** 800**,** WHITE**);**

printString**(**120**,** 300**,** "Please select the difficulty level:"**,** WHITE**,** RED**);**

printString**(**120**,** 350**,** "Easy"**,** WHITE**,** BLUE**);**

printString**(**120**,** 400**,** "Hard"**,** BLUE**,** WHITE**);**

printString**(**120**,** 450**,** "Press KEY0 to enter."**,** WHITE**,** RED**);**

Delay**(**5000000**);**

**}**

Figure 3. Page 2: Difficulty Level Page

On this page, KEY\_UP and KEY1 interrupts are used to select easy or hard level. An extern universal variant “hard” is exported to the main program to indicate the difficulty level. “hard = 1” implies easy mode, “hard = 2” implies hard mode.

void EXTI0\_IRQHandler**(**void**)**

**{**

**if** **(**EIE3810\_read\_KEY\_UP**()** **&** **(**Page **==** 2**)){** //Easy

printString**(**120**,** 350**,** "Easy"**,** WHITE**,** BLUE**);**

printString**(**120**,** 400**,** "Hard"**,** BLUE**,** WHITE**);**

hard **=** 1**;**

**}**

Figure 4. KEY\_UP interrupt exception in Page 2 to select easy mode

void EXTI3\_IRQHandler**(**void**)**

**{**

**if** **(**EIE3810\_read\_KEY1**()** **&** **(**Page **==** 2**)){**

printString**(**120**,** 350**,** "Easy"**,** BLUE**,** WHITE**);**

printString**(**120**,** 400**,** "Hard"**,** WHITE**,** BLUE**);**

hard **=** 2**;**

**}**

Figure 5. KEY1 interrupt exception in Page 2 to select hard mode

We enter KEY0 interrupt. Inside KEY0 interrupt exception, we ask the user to send a random direction from USART1 and enter the next page: USART Page.

void EXTI4\_IRQHandler**(**void**)**

**{**

**if** **(**EIE3810\_read\_KEY0**()** **&** **(**Page **==** 2**)){**

EIE3810\_TFTLCD\_FillRectangle**(**0**,** 480**,** 0**,** 800**,** WHITE**);**

printString2412**(**50**,** 400**,** "Use USART for a random direction."**,** WHITE**,** RED**);**

Page **=** 3**;**

**}**

Figure 6. KEY0 interrupt exception in Page 2 and enter USART Page

Inside USART1 exception, the transmitted number is loaded to buffer, which is an extern universal variant exported to the main program to indicate the initial direction of the ball. A sentence indicating the number is printed to the LCD and we enter Game Page.

void USART1\_IRQHandler**(**void**)**

**{**

u32 CR1**;**

**if** **((**USART1**->**SR **&** **(**1**<<**5**))** **&&** **(**Page **==** 3**))** //check if received data is ready to be read

**{**

buffer**=**USART1**->**DR **+** 0x30**;** //read the data value to buffer

CR1**=**USART1**->**CR1**;**

**if** **(**buffer**==** '0'**{**

printString**(**120**,** 450**,** "The random number received is: 0"**,** WHITE**,** RED**);**

**}**

**if** **(**buffer**==**'1'**){**

printString**(**120**,** 450**,** "The random number received is: 1"**,** WHITE**,** RED**);**

**}**

**if** **(**buffer**==**'2'**){**

printString**(**120**,** 450**,** "The random number received is: 2"**,** WHITE**,** RED**);**

**}**

**if** **(**buffer**==**'3'**){**

printString**(**120**,** 450**,** "The random number received is: 3"**,** WHITE**,** RED**);**

**}**

**if** **(**buffer**==**'4'**){**

printString**(**120**,** 450**,** "The random number received is: 4"**,** WHITE**,** RED**);**

**}**

**if** **(**buffer**==**'5'**){**

printString**(**120**,** 450**,** "The random number received is: 5"**,** WHITE**,** RED**);**

**}**

**if** **(**buffer**==**'6'**){**

printString**(**120**,** 450**,** "The random number received is: 6"**,** WHITE**,** RED**);**

**}**

**if** **(**buffer**==**'7'**){**

printString**(**120**,** 450**,** "The random number received is: 7"**,** WHITE**,** RED**);**

**}**

Delay**(**20000000**);**

EIE3810\_TFTLCD\_FillRectangle**(**0**,** 480**,** 0**,** 800**,** WHITE**);**

Page **=** 4**;**

**}**

**}**

Figure 7. USART1 interrupt exception and Page 3: USART Page

Inside the game page, TIM4 is initialized to record the elapsed time of the game (that’s why it is initialized after we enter Game Page). If buffer is smaller and equal to 51 (ASCII Code of 3), the ball will bounce to the right, otherwise, the ball will bounce to the left, which is controlled by “xdir”. “xdir” refers to X-axis direction, “xdir = 1” ball moves to the right and x increases; “xdir = -1” ball moves to the left and x decreases. The angle of bounce is controlled by the increment of y. Basically, the larger the buffer number is, the bigger the y step is. In total, there are 8 different initial directions as shown in the Figure8.

图片包含 天空, 白板, 文字, 户外

已生成极高可信度的说明Figure 8. 8 different initial directions for the ball

The refreshment of the ball is put inside a while loop controlled by SYSTICK (task1HeartBeat count to 2 (20ms), Game Page will be refreshed). The idea is quite simple, the original ball is drawn white first and then x, y location of the ball will be updated according to the initial direction and difficulty level. Then, draw the ball red, we will see the ball moving. By the way, the difficulty level is controlled in a very clever way: the x and y step will be timed by “hard”, which is introduced before (“hard = 1” implies easy mode, “hard = 2” implies hard mode.) Thus, in hard mode, the ball will move 2 times faster than in easy mode.

**if** **(**Page **==** 4**){** //Game Page

count **=** 0**;**

EIE3810\_TIM4\_Init**(**9999**,**7199**);** // timer start

**if** **(!**xdir**){** // control initial direction

**if** **(**buffer **<=** 51**){**

xdir **=** 1**;**

**}**

**else{**

xdir **=** **-**1**;**

**}**

**}**

**while** **((**task1HeartBeat **>=** 2**)** **&&** **(**Page **==** 4**)){**

EIE3810\_TFTLCD\_DrawCircle**(**x**,**y**,** 10**,** 1**,** WHITE**);**

**if** **((**10 **<=** x**)** **&&** **(**x **<=** 470**)){** //in

**if** **(**xdir **==** 1**){** //right

x **=** x **+** 2 **\*** hard**;**

**}**

**if** **(**xdir **==** **-**1**){** //left

x **=** x **-** 2 **\*** hard**;**

**}**

**}**

**else** **if** **(**x **>** 470**){** //out

xdir **=** **-**1**;**

x **=** x **-** 5**;**

GPIOB**->**ODR **^=** 1**<<**8**;**

Delay**(**50000**);**

GPIOB**->**ODR **^=** 1**<<**8**;**

sprintf**(**bounces**,**"%d"**,**Bounces**);**

printString**(**9**\***18**,**700 **-** 36**,** bounces **,**WHITE**,**WHITE**);**

Bounces **++;**

printString**(**9**\***18**,**700 **-** 36**,** bounces **,**RED**,**WHITE**);**

**}**

**else** **if** **(**x **<** 10**){** //out

xdir **=** 1**;**

x **=** x **+** 5**;**

GPIOB**->**ODR **^=** 1**<<**8**;**

Delay**(**50000**);**

GPIOB**->**ODR **^=** 1**<<**8**;**

sprintf**(**bounces**,**"%d"**,**Bounces**);**

printString**(**9**\***18**,**700 **-** 36**,** bounces **,**WHITE**,**WHITE**);**

Bounces **++;**

printString**(**9**\***18**,**700 **-** 36**,** bounces **,**RED**,**WHITE**);**

**}**

**if** **((**20**<=** y**)** **&&** **(**y **<=** 780**)){**

**if** **(**ydir **==** **-**1**){**

**if** **(**buffer **<=** 51**){**

y **=** y **+** **((**buffer **-** 48**)\***2 **+**1**)** **\*** hard**;**

**}**

**else{**

y **=** y **+** **((**buffer **-** 48 **-** 4**)\***2 **+** 1**)** **\*** hard**;**

**}**

**}**

**if** **(**ydir **==** 1**){**

**if** **(**buffer **<=** 51**){**

y **=** y **-** **((**buffer **-** 48**)\***2 **+**1**)** **\*** hard**;**

**}**

**else{**

y **=** y **-** **((**buffer **-** 48 **-** 4**)\***2 **+** 1**)** **\*** hard**;**

**}**

**}**

**}**

**else** **if** **(**y **>** 780**){**

**if** **((**x **>=** x2**-**5**)** **&&** **(**x **<=** x2**+**85**)){**

y **=** y **-** 6**;**

ydir **=** 1**;**

GPIOB**->**ODR **^=** 1**<<**8**;**

Delay**(**50000**);**

GPIOB**->**ODR **^=** 1**<<**8**;**

sprintf**(**bounces**,**"%d"**,**Bounces**);**

printString**(**9**\***18**,**700 **-** 36**,** bounces **,**WHITE**,**WHITE**);**

Bounces **++;**

printString**(**9**\***18**,**700 **-** 36**,** bounces **,**RED**,**WHITE**);**

sprintf**(**lower**,**"%d"**,**Lower**);**

printString**(**9**\***18**,**700**+**36**,** lower **,**WHITE**,**WHITE**);**

Lower **++;**

printString**(**9**\***18**,**700**+**36**,** lower **,**RED**,**WHITE**);**

**}**

**else{**

Delay**(**5000000**);**

printString2412**(**0x64**+**125**,**0x64**,**"Game Over"**,**WHITE**,**BLACK**);**

printString2412**(**0x64**+**125**,**0x64**+**50**,**"Upper Player Wins!"**,**WHITE**,**BLACK**);**

**if** **((**x **>** 210**)** **&&** **(**x **<=** 336**)){**

printString2412**(**x**,**740**,**"Death review"**,**WHITE**,**BLACK**);**

**}**

**else** **if** **(**x **>** 336**){**

printString2412**(**336**,**740**,**"Death review"**,**WHITE**,**BLACK**);**

**}**

**else{**

printString2412**(**190**,**740**,**"Death review"**,**WHITE**,**BLACK**);**

**}**

Page **=** 5**;**

**}**

**}**

**else** **if** **(**y **<** 20**){**

**if** **((**x **>=** x1**-**5**)** **&&** **(**x **<=** x1**+**85**)){**

y **=** y **+** 6**;**

ydir **=** **-**1**;**

GPIOB**->**ODR **^=** 1**<<**8**;**

Delay**(**50000**);**

GPIOB**->**ODR **^=** 1**<<**8**;**

sprintf**(**bounces**,**"%d"**,**Bounces**);**

printString**(**9**\***18**,**700 **-** 36**,** bounces **,**WHITE**,**WHITE**);**

Bounces **++;**

printString**(**9**\***18**,**700 **-** 36**,** bounces **,**RED**,**WHITE**);**

sprintf**(**upper**,**"%d"**,**Upper**);**

printString**(**9**\***18**,**700**,** upper **,**WHITE**,**WHITE**);**

Upper **++;**

printString**(**9**\***18**,**700**,** upper **,**RED**,**WHITE**);**

**}**

**else{**

Delay**(**5000000**);**

printString2412**(**0x64**+**125**,**0x64**,**"Game Over"**,**WHITE**,**BLACK**);**

printString2412**(**0x64**+**125**,**0x64**+**50**,**"Lower Player Wins!"**,**WHITE**,**BLACK**);**

**if** **(**x **<=** 336**){**

printString2412**(**x**,**36**,**"Death review"**,**WHITE**,**BLACK**);**

**}**

**else** **if** **(**x **>** 336**){**

printString2412**(**336**,**36**,**"Death review"**,**WHITE**,**BLACK**);**

**}**

Page **=** 5**;**

**}**

**}**

EIE3810\_TFTLCD\_DrawCircle**(**x**,**y**,** 10**,** 1**,** RED**);**

EIE3810\_TFTLCD\_FillRectangle**(**x2**,** 80**,** 795**,** 5**,** BLACK**);**

task1HeartBeat **=** 0**;**

**}**

**}**

**}**

**}**

Figure 9. Page 4: Game Page

For checking the boundary conditions of the ball. If x is between 10 to 470, the ball will move to its next location according “xdir”; if x is out of the range of 10 to 470 (the ball is out of the screen), “xdir” will be changed: +1 to -1, -1 to +1, thus, x will be updated to the opposite direction. The buzzer is turned on and tuned off after a while, Bounces is incremented by one and updated to the screen. The same scenario happens for the Y-axis. If y is between 20 to 780, the ball will move to its next location according “ydir” and the number of buffer; if y is out of the range of 20 to 780 (the ball is out of the screen), we check the x value. If x value is between x2 and x2 + 80 (y > 780, checking lower bench) or x1 and x1 + 80 (y < 20, checking upper bench) “ydir” will be changed: +1 to -1, -1 to +1, thus, y will be updated to the opposite direction. The buzzer is turned on and tuned off after a while, Bounces is incremented by one and updated to the screen; otherwise, the user failed to catch the ball using the bench and game over, we enter Page 5: End Page to break the while loop of Game Page. Some text indicating the winner will show on the screen.

During the game, KEY0 and KEY2 interrupt are enabled to move the lower bench.

**if** **(**EIE3810\_read\_KEY0**()** **&** **(**Page **==** 4**)){**

EIE3810\_TFTLCD\_FillRectangle**(**x2**,** 80**,** 795**,** 5**,** WHITE**);**

**if** **(**x2 **<** 400**){**

x2 **=** x2 **+** 25**;**

**}**

EIE3810\_TFTLCD\_FillRectangle**(**x2**,** 80**,** 795**,** 5**,** BLACK**);**

**}**

Figure 10. KEY0 interrupt exception in Page 4: move lower bench to the right

void EXTI2\_IRQHandler**(**void**)**

**{**

**if** **((**EIE3810\_read\_KEY2**())** **&** **(**Page **==** 4**)){**

EIE3810\_TFTLCD\_FillRectangle**(**x2**,** 80**,** 795**,** 5**,** WHITE**);**

**if** **(**x2 **>** 0**){**

x2 **=** x2 **-** 25**;**

**}**

EIE3810\_TFTLCD\_FillRectangle**(**x2**,** 80**,** 795**,** 5**,** BLACK**);**

**}**

EXTI**->**PR **=** 1**<<**2**;**//pending register

**}**

Figure 11. KEY2 interrupt exception in Page 4: move lower bench to the left

During the game, KEY1 interrupt is enabled to pause and resume the game. If you press KEY1 for the first time (count = 0), Page is set to be 6. Thus, the processor will jump out of Game page and jump to Stop Page. Inside Stop Page, we print a sentence telling user that we are in Stop mode and set count to 1, then, enter an endless loop. Until KEY1 is pressed again (count = 1), the processor jumps to the interrupt exception and go back to Game Page and continue the game. Inside Game Page, count is set to 0 for the next pause.

void EXTI3\_IRQHandler**(**void**)**

**{**

**if** **(**EIE3810\_read\_KEY1**()** **&&** **(**Page **==** 4**)** **&&** **(**count **==** 0**)){**

Page **=** 6**;** // Go to Stop Page

**}**

**if** **(**EIE3810\_read\_KEY1**()** **&&** **(**Page **==** 6**)** **&&** **(**count **==** 1**)){**

Page **=** 4**;** // Go back to Game Page

printString**(**120**,** 500**,** "You are in STOP mode"**,** WHITE**,** WHITE**);** //Clear

**}**

EXTI**->**PR **=** 1**<<**3**;**//pending register

**}**

Figure 12. KEY1 interrupt exception in Page 4: pause and resume

**if** **(**Page **==** 6**){** //Stop Page

printString**(**120**,** 500**,** "You are in STOP mode"**,** WHITE**,** BLUE**);**

count **=** 1**;**

**while(**Page **==** 6**);**

**}**

Figure 13. Page 6: Stop Page

JOYPAD\_Read() is triggered by TIM3 to read the JOYPAD input every 10ms. All the functions of Keys can be realized on the JOYPAD. Start button can pause/continue the game, Left and Right button can move the upper bench in Game Page. Up and Down button can select difficulty level in Page 2.

void TIM3\_IRQHandler**(**void**)**

**{**

**if(**TIM3**->**SR**&**1**<<**0**)**

**{**

u8 JoyPadInput**;**

JoyPadInput**=**JOYPAD\_Read**();**

**if** **(**JoyPadInput**)**

**{**

u8 button1**=**JoyPadInput **&** 0x01**;**

u8 button2**=**JoyPadInput **&** 0x02**;**

u8 button3**=**JoyPadInput **&** 0x04**;**

u8 button4**=**JoyPadInput **&** 0x08**;**

u8 button5**=**JoyPadInput **&** 0x10**;**

u8 button6**=**JoyPadInput **&** 0x20**;**

u8 button7**=**JoyPadInput **&** 0x40**;**

u8 button8**=**JoyPadInput **&** 0x80**;**

**if** **(**button4**){**

button**++;**

**if** **((**button **>**2**)** **&&** **(**Page **==** 4**)** **&&** **(**count **==** 0**)){**

Page **=** 6**;** // Go to Stop Page

EIE3810\_turn\_off\_LED1**();**

button **=** 0**;**

**}**

**if** **((**button **>**2**)** **&&** **(**Page **==** 6**)** **&&** **(**count **==** 1**)){**

Page **=** 4**;** // Go back to Game Page

EIE3810\_turn\_on\_LED1**();**

printString**(**120**,** 500**,** "You are in STOP mode"**,** WHITE**,** WHITE**);** //Clear

button **=** 0**;**

**}**

**}**

**else** **if** **(**button5**>>**4**){**

button**++;**

**if** **((**button **>** 2**)** **&&** **(**Page **==** 2**)){**

printString**(**120**,** 350**,** "Easy"**,** WHITE**,** BLUE**);**

printString**(**120**,** 400**,** "Hard"**,** BLUE**,** WHITE**);**

hard **=** 1**;**

button **=** 0**;**

**}**

**}**

**else** **if** **(**button6**>>**5**){**

button**++;**

**if** **((**button **>** 2**)** **&&** **(**Page **==** 2**)){**

printString**(**120**,** 350**,** "Easy"**,** BLUE**,** WHITE**);**

printString**(**120**,** 400**,** "Hard"**,** WHITE**,** BLUE**);**

hard **=** 2**;**

button **=** 0**;**

**}**

**}**

**else** **if** **((**button7**>>**6**)** **&&** **(**Page **==** 4**)){**

EIE3810\_TFTLCD\_FillRectangle**(**x1**,** 80**,** 0**,** 5**,** WHITE**);**

**if** **(**x1 **>** 0**){**

x1**--;**

**}**

EIE3810\_TFTLCD\_FillRectangle**(**x1**,** 80**,** 0**,** 5**,** BLACK**);**

**}**

**else** **if** **((**button8**>>**7**)** **&&** **(**Page **==** 4**)){**

EIE3810\_TFTLCD\_FillRectangle**(**x1**,** 80**,** 0**,** 5**,** WHITE**);**

**if** **(**x1 **<**400**){**

x1**++;**

**}**

EIE3810\_TFTLCD\_FillRectangle**(**x1**,** 80**,** 0**,** 5**,** BLACK**);**

**}**

**else{**

count **=** 0**;**

**}**

**}**

**}**

TIM3**->**SR **&=~(**1**<<**0**);**

**}**

Figure 14. TIM3: JOYPAD

TIM4 is initialized to function as a real time clock, which updates the strings on the LCD every second, showing the Number of Bounces, Upper and Lower Player Score and the elapsed Time.

void TIM4\_IRQHandler**(**void**){**

**if** **((**TIM4**->**SR **&** 1**<<**0**)** **&&** **(**Page **==** 4**)){** //Update interrupt pending

printString**(**0**,**700 **-** 36**,**"Number of Bounces:"**,** RED**,** WHITE**);**

printString**(**9**\***18**,**700 **-** 36**,** bounces **,**RED**,**WHITE**);**

printString**(**0**,**700**,**"Upper Player Score:"**,** RED**,** WHITE**);**

printString**(**9**\***18**,**700**,** upper **,**RED**,**WHITE**);**

printString**(**0**,**700**+**36**,**"Lower Player Score:"**,** RED**,** WHITE**);**

printString**(**9**\***18**,**700**+**36**,** lower **,**RED**,**WHITE**);**

printString**(**0**,**700**+**72**,**"Time: "**,** RED**,** WHITE**);**

printString**(**9**\***18**,**700**+**72**,**t**,**WHITE**,**WHITE**);**

Timer **++;**

sprintf**(**t**,**"%d"**,**Timer**);**

printString**(**0**,**700**+**72**,**"Time: "**,** RED**,** WHITE**);**

printString**(**9**\***18**,**700**+**72**,**t**,**RED**,**WHITE**);**

**}**

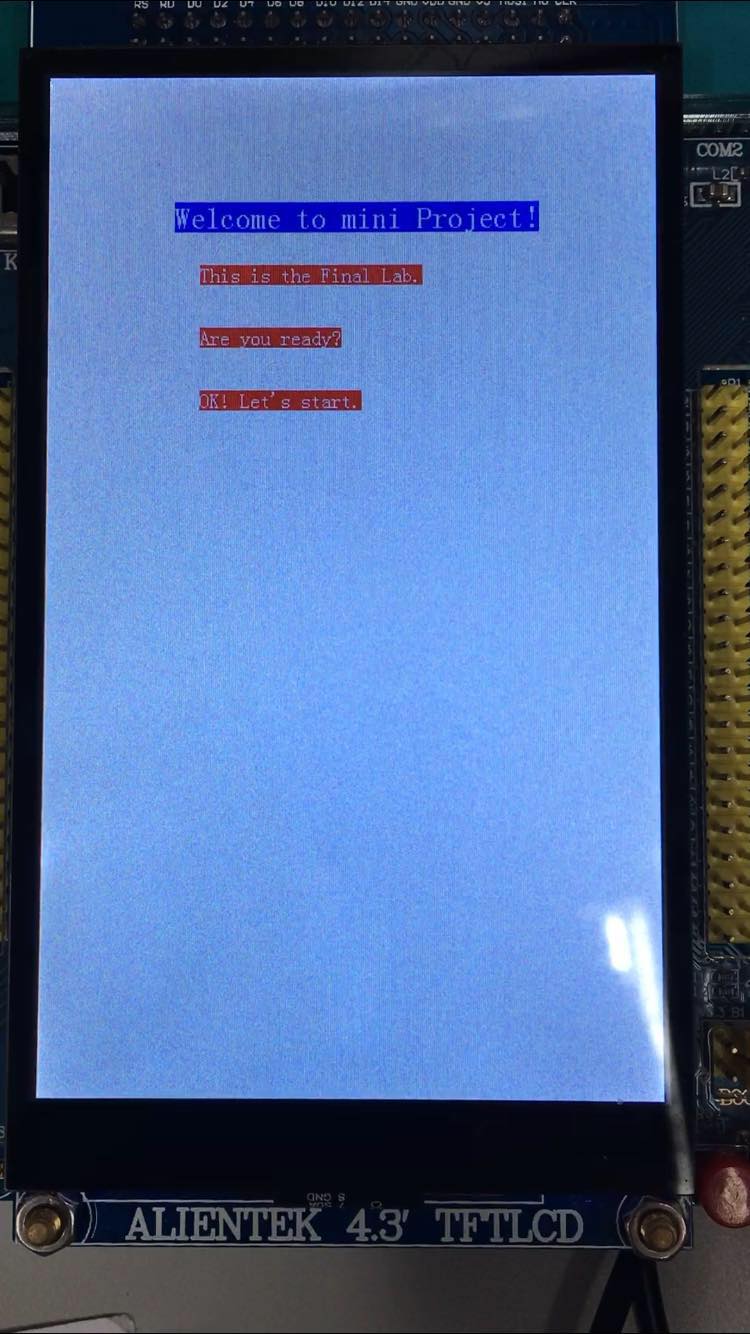
TIM4**->**SR **&=** **~(**1**<<**0**);** //Clear the update.

**}**

Figure 15. TIM4: Update strings of Number of Bounces,

Upper and Lower Player Score and the elapsed Time

By now, all the functions have already been achieved.

* **Test Result**

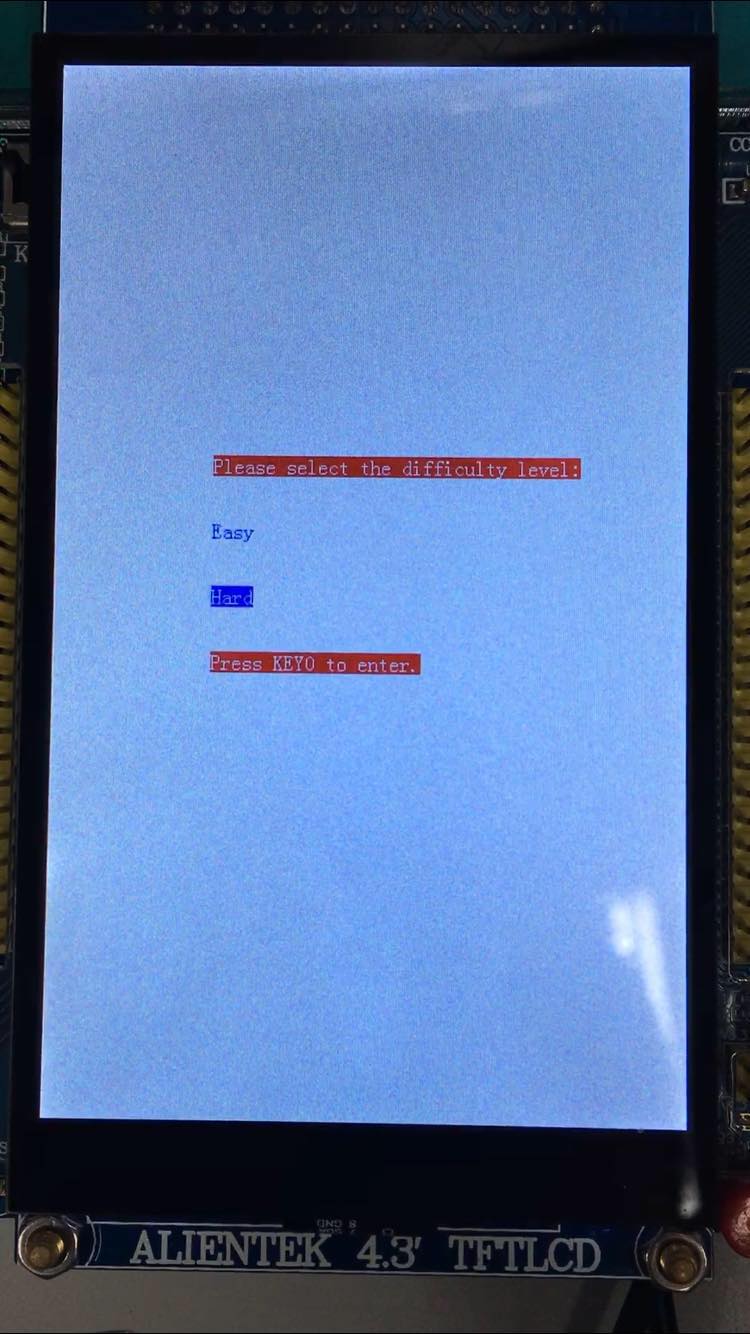
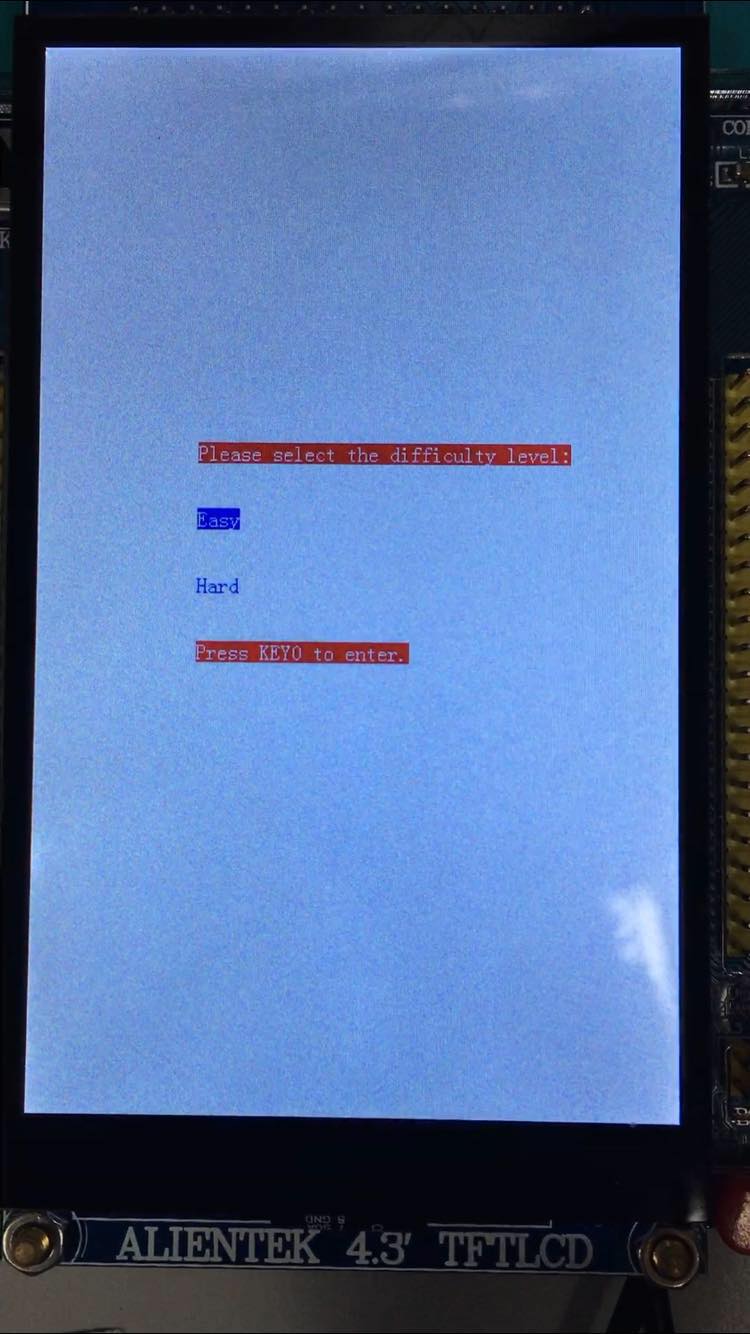
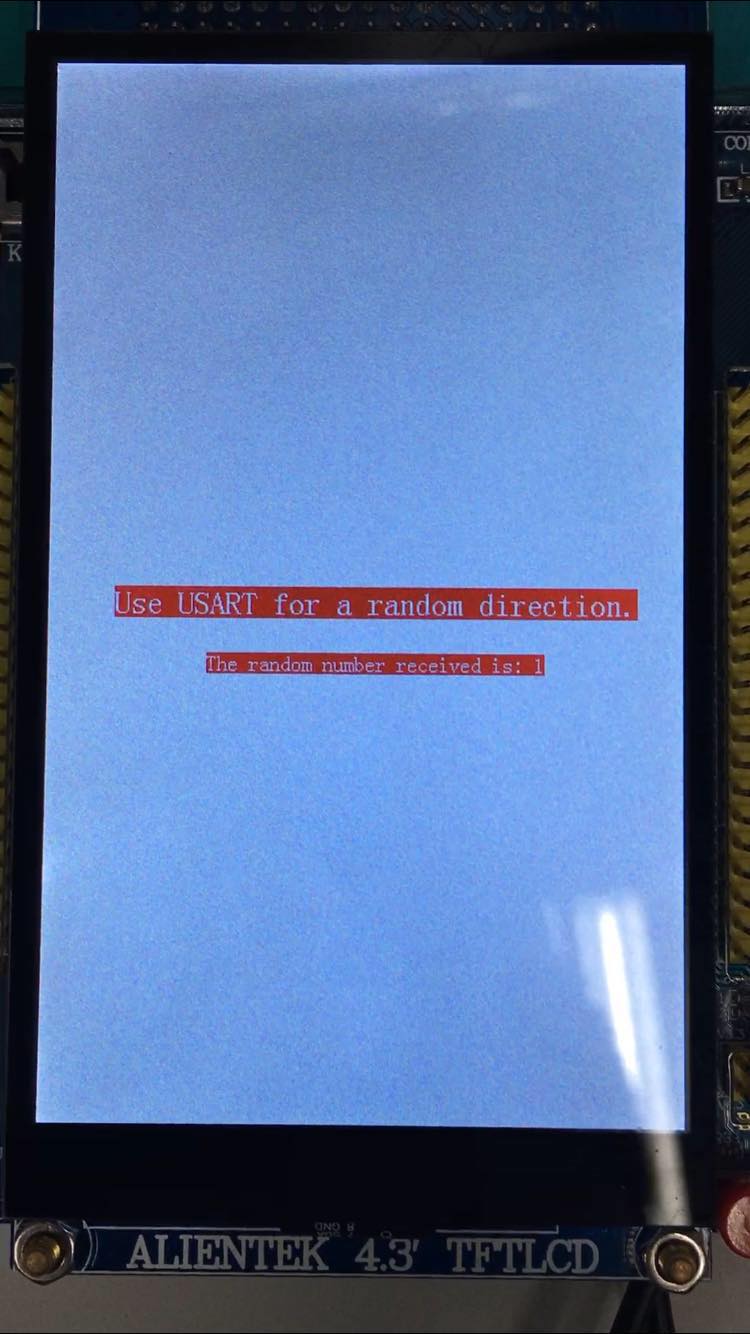
Figure 16. Welcome Page

Figure 17. Difficulty Level Page: Easy & Hard

****Figure 18. USART Page

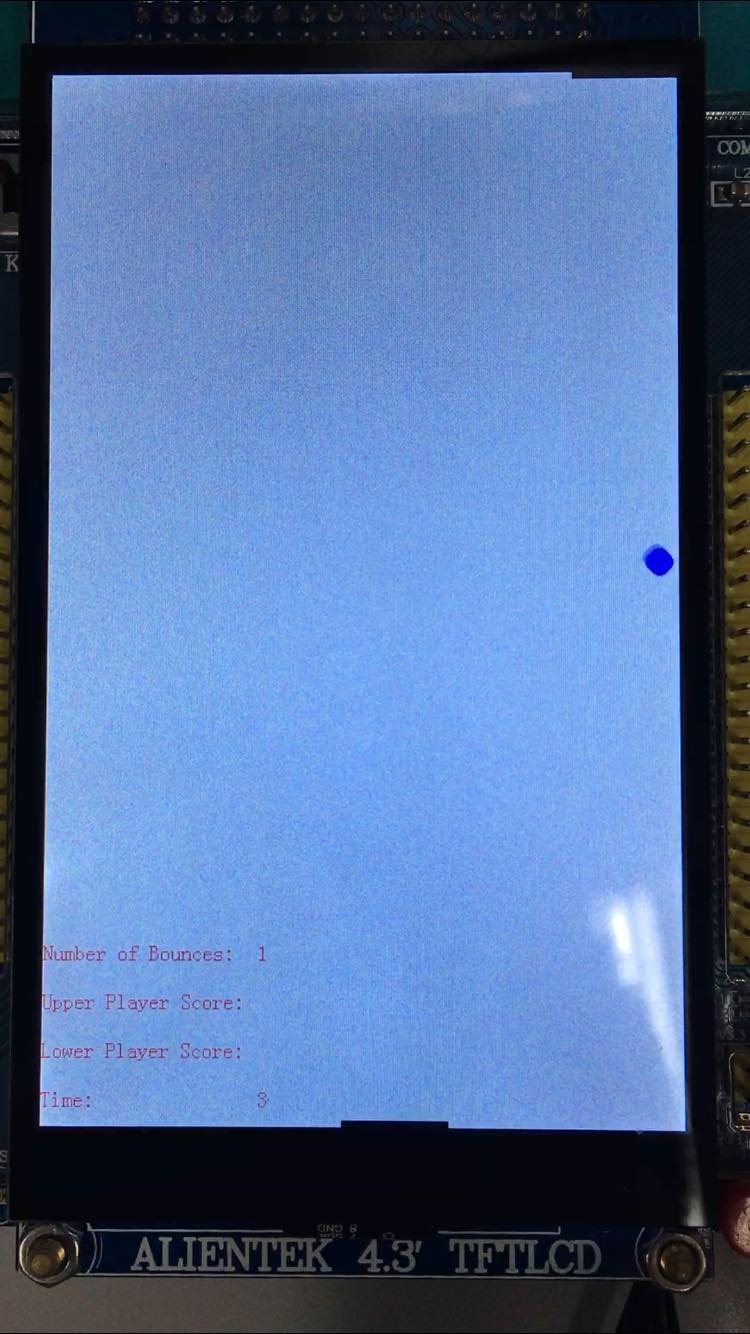
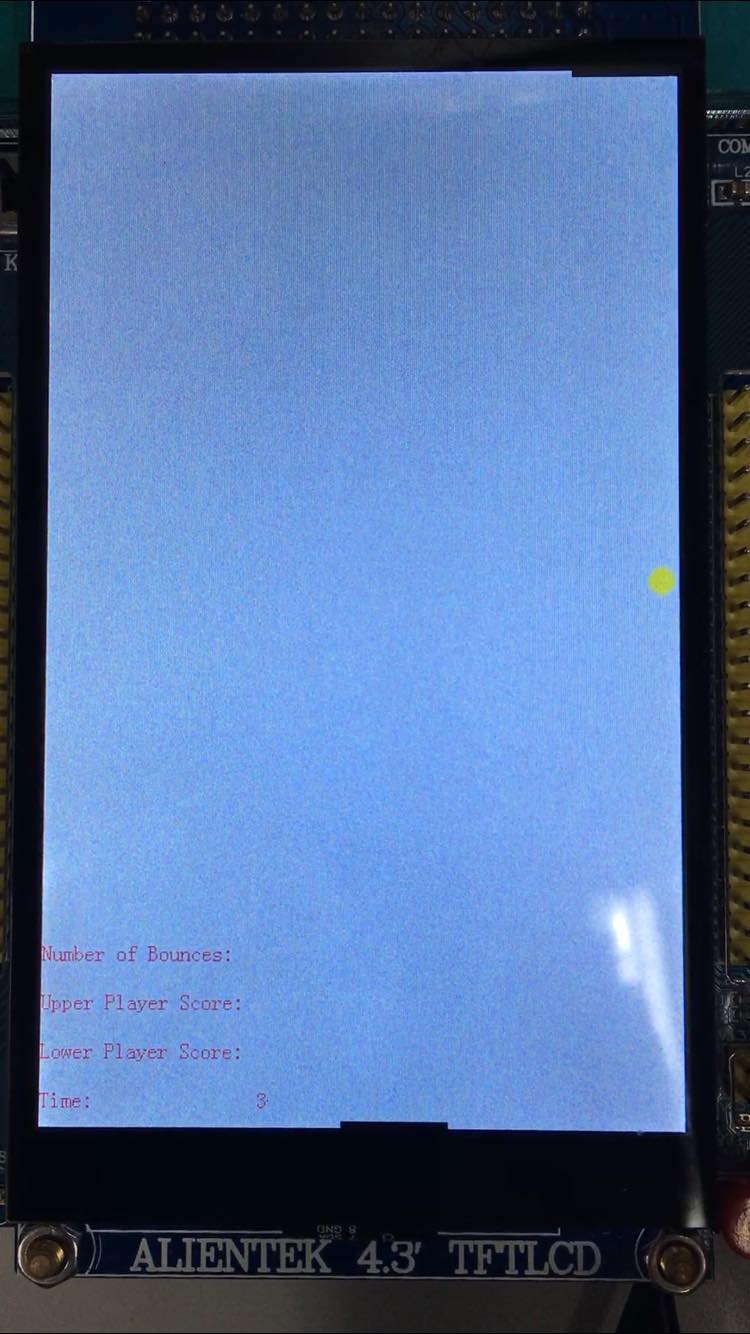


Figure 19. Game page: Bounce at the right side, Number of Bounces +1

The ball will change its color before and after each successful bounce, which is one of the improvements I made in Experiment 2. If the report is printed colorful, it can be easily seen. I didn’t record the video for the original version, but it is still able to illustrate the test result. In the original version of Experiment 1, the ball will not change its color it is always red.

Figure 20. Game page: Bounce at the upper side, Number of Bounces +1, Upper Player Score +1

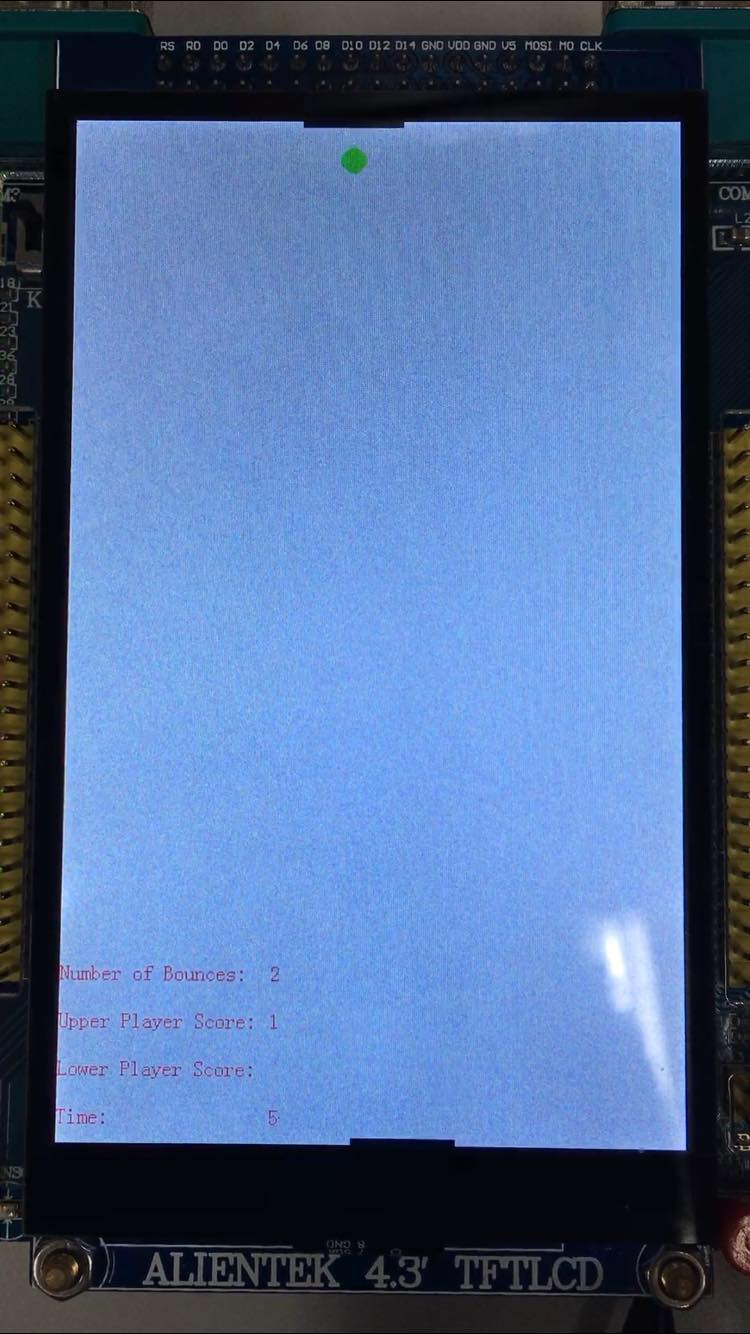
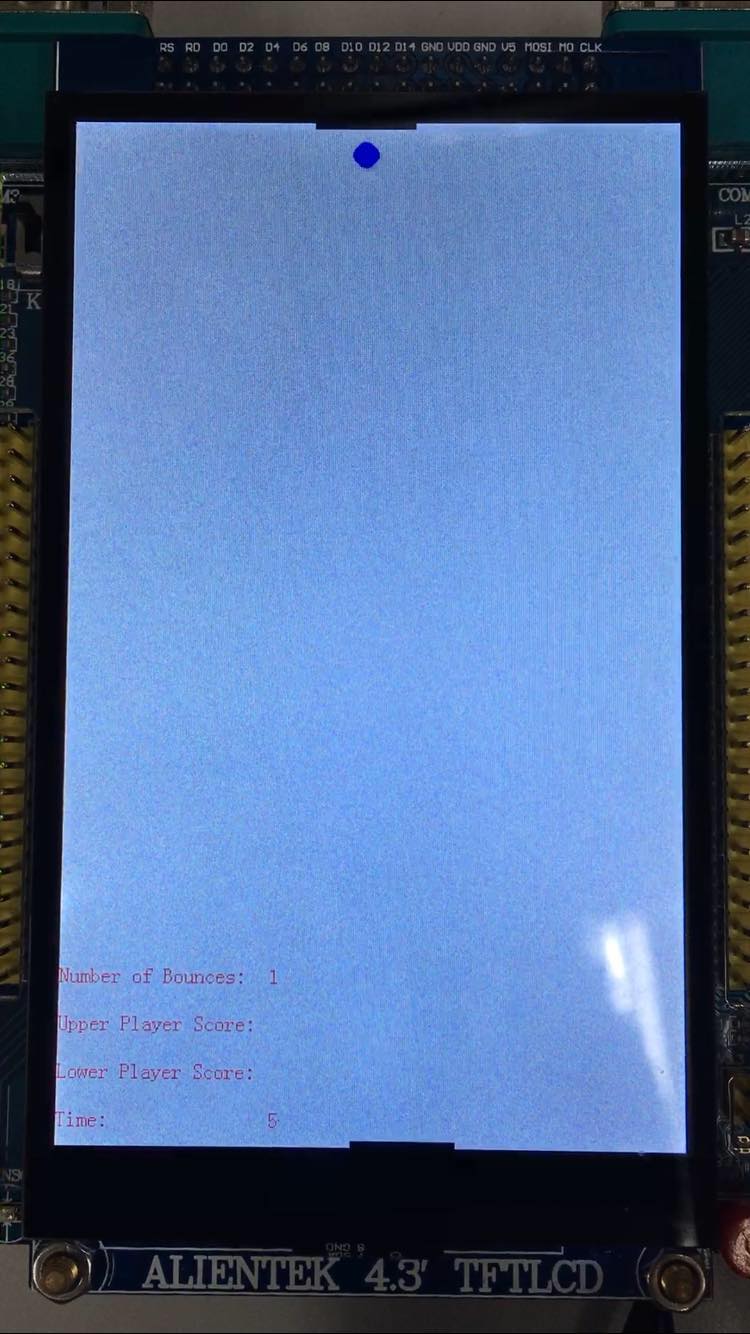
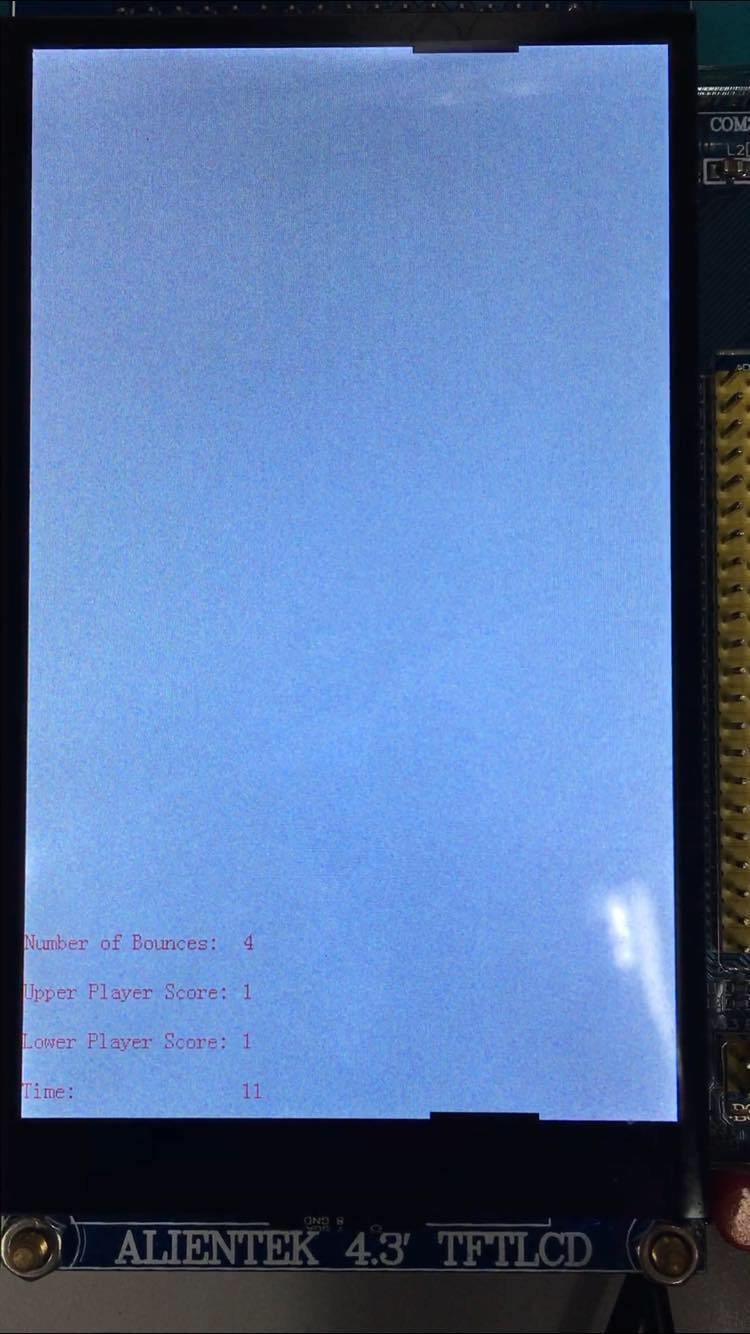
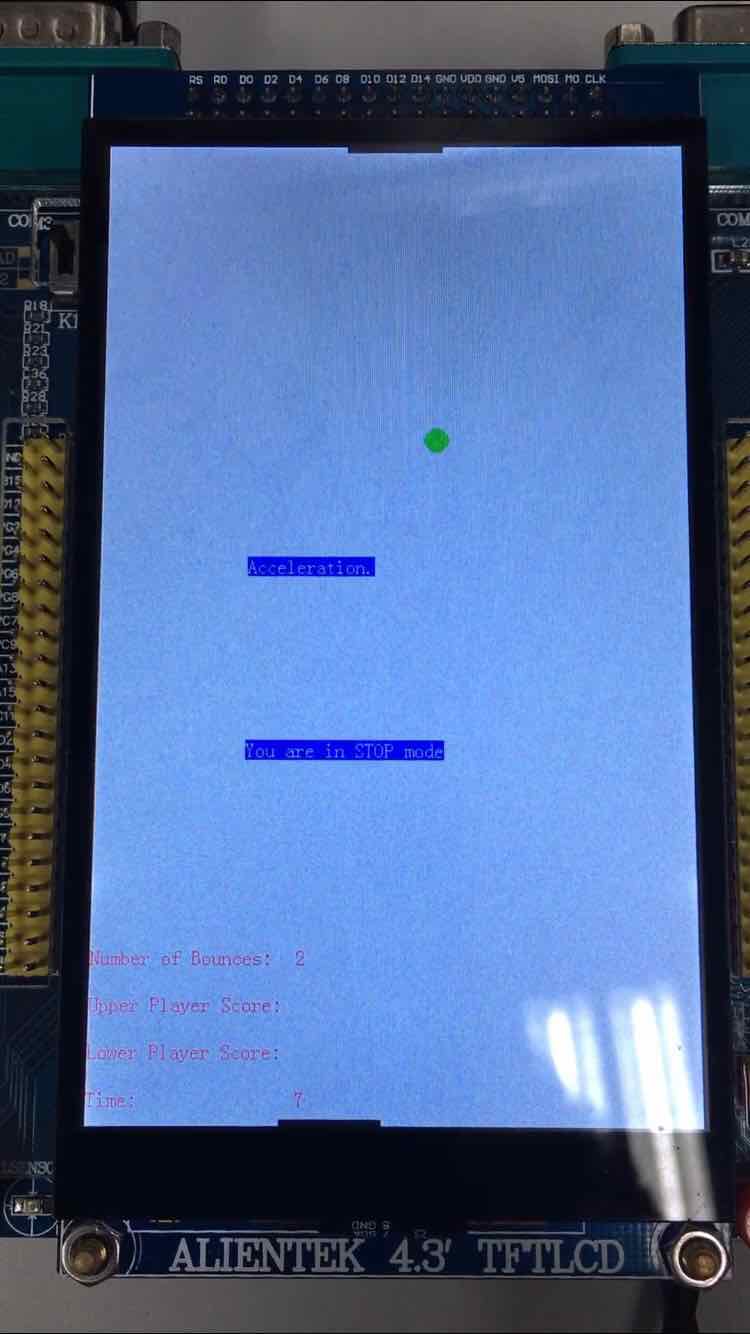
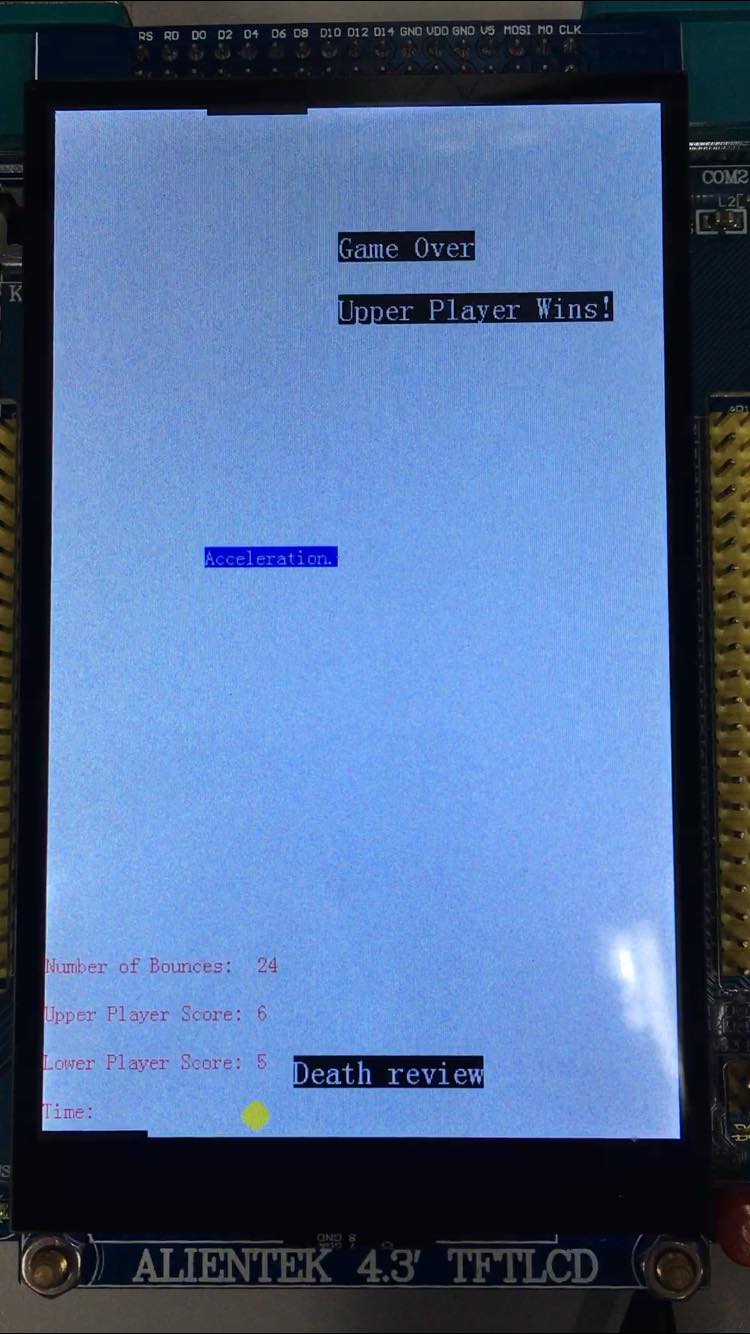


Figure 21. Game page: Bounce at the lower side, Number of Bounces +1, Lower Player Score +1

The right one of Figure 21, the ball is invisible for the Player as it changed to white because of the improvement as I mentioned before. The Player need to predict the next movement of the ball, which will add difficulty.

Figure 22. Pause and Resume Figure 23. End Page



* **Conclusion**

We successfully completed step 2.2) to 2.8) and all the functions of game is achieved.

**Experiment 2**

* **Design**

**1.** As mentioned above, the first improvement is to change the ball color every time of successful bounces. The detailed code is shown below.

**if** **(**Bounces**%**6 **==** 0**){**

EIE3810\_TFTLCD\_DrawCircle**(**x**,**y**,** 10**,** 1**,** RED**);**

**}**

**else** **if** **(**Bounces**%**6 **==** 1**){**

EIE3810\_TFTLCD\_DrawCircle**(**x**,**y**,** 10**,** 1**,** YELLOW**);**

**}**

**else** **if** **(**Bounces**%**6 **==** 2**){**

EIE3810\_TFTLCD\_DrawCircle**(**x**,**y**,** 10**,** 1**,** BLUE**);**

**}**

**else** **if** **(**Bounces**%**6 **==** 3**){**

EIE3810\_TFTLCD\_DrawCircle**(**x**,**y**,** 10**,** 1**,** GREEN**);**

**}**

**else** **if** **(**Bounces**%**6 **==** 4**){**

EIE3810\_TFTLCD\_DrawCircle**(**x**,**y**,** 10**,** 1**,** BLACK**);**

**}**

**else** **if** **(**Bounces**%**6 **==** 5**){**

EIE3810\_TFTLCD\_DrawCircle**(**x**,**y**,** 10**,** 1**,** WHITE**);**

**}**

Figure 24. Change the ball color

We utilize Number of Bounces, the ball is invisible for the Player when it changed to white every 6 times of bounces. The Player need to predict the next movement of the ball, which will add difficulty and fun.

**2.** The second improvement is that another feature of playing will the CPU is added. We will enter Page 10 for one more selection before the game (CPU or another player with JOYPAD).

**if** **((**Page **==** 10**)** **&** **(!**CPU**)){** //CPU Page

printString**(**100**,** 300**,** "Please select who you would like to play with:"**,** WHITE**,** RED**);**

printString**(**120**,** 350**,** "Play with another user with JoyPad."**,** WHITE**,** BLUE**);**

printString**(**120**,** 400**,** "Play with CPU."**,** BLUE**,** WHITE**);**

printString**(**120**,** 450**,** "Press KEY0 to enter."**,** WHITE**,** RED**);**

Delay**(**5000000**);**

**}**

Figure 25: Page 10: CPU or JOYPAD selection Page

On this page, KEY\_UP and KEY1 interrupts are used to select JOYPAD or CPU mode. An extern universal variant “CPU” is exported to the main program to indicate the mode. “CPU = 1” implies CPU mode, “hard = 0” implies JOYPAD mode.

**if** **(**EIE3810\_read\_KEY1**()** **&** **(**Page **==** 10**)){** //CPU

printString**(**120**,** 350**,** "Play with another user with JoyPad."**,** BLUE**,** WHITE**);**

printString**(**120**,** 400**,** "Play with CPU."**,** WHITE**,** BLUE**);**

EIE3810\_turn\_on\_LED1**();**

CPU **=** 1**;**

**}**

Figure 26. KEY1 interrupt exception in Page 10 to select CPU mode

**if** **(**EIE3810\_read\_KEY\_UP**()** **&** **(**Page **==** 10**)){** //JoyPad

printString**(**120**,** 350**,** "Play with another user with JoyPad."**,** WHITE**,** BLUE**);**

printString**(**120**,** 400**,** "Play with CPU."**,** BLUE**,** WHITE**);**

CPU **=** 0**;**

**}**

Figure 27. KEY\_UP interrupt exception in Page 10 to select JOYPAD mode

During the game, if CPU mode is on, x1 (upper bench) will be automatically updated with respect to x (ball), thus, the Upper Player (CPU) will never lose. The JOYPAD will be disabled if CPU mode is on.

**if** **(**CPU**){** // Play with CPU

EIE3810\_TFTLCD\_FillRectangle**(**x1**,** 80**,** 0**,** 5**,** WHITE**);**

**if** **((**x **>=** 40**)** **&** **(**x **<=** 440**)){**

x1 **=** x **-** 40**;**

**}**

**else** **if** **(**x **<** 40**){**

x1 **=** 0**;**

**}**

**else{**

x1 **=** 400**;**

**}**

EIE3810\_TFTLCD\_FillRectangle**(**x1**,** 80**,** 0**,** 5**,** BLACK**);**

**}**

Figure 28. CPU mode

**3.** The third improvement is that both player had a chance to speed up the movement of the ball. If the ball is speeded up, “hard = 2\*hard”, which will double the updating step of both x and y. Lower Player presses KEY\_UP, Upper Player presses JOYPAD UP.

**if** **(**EIE3810\_read\_KEY\_UP**()** **&** **(**Page **==** 4**)){** //JoyPad

**if** **(**acce **==** 0**){**

printString**(**120**,** 350**,** "Acceleration."**,** WHITE**,** BLUE**);**

hard **=** 2**\***hard**;**

acce **=** 1**;**

**}**

**}**

Figure 29. KEY\_UP in Game Page to speed up the movement of the ball

**if** **(**button3**){**

button**++;**

**if** **((**button **>**2**)** **&&** **(**Page **==** 4**)){** //JoyPad

**if** **(**acce **==** 0**){**

printString**(**120**,** 350**,** "Acceleration."**,** WHITE**,** BLUE**);**

hard **=** 2**\***hard**;**

acce **=** 1**;**

**}**

**}**

**}**

Figure 30. JOYPAD Button Up in Game Page to speed up the movement of the ball

* **Test Result:**

Color changing feature has been shown in Test Result of Experiment 1.

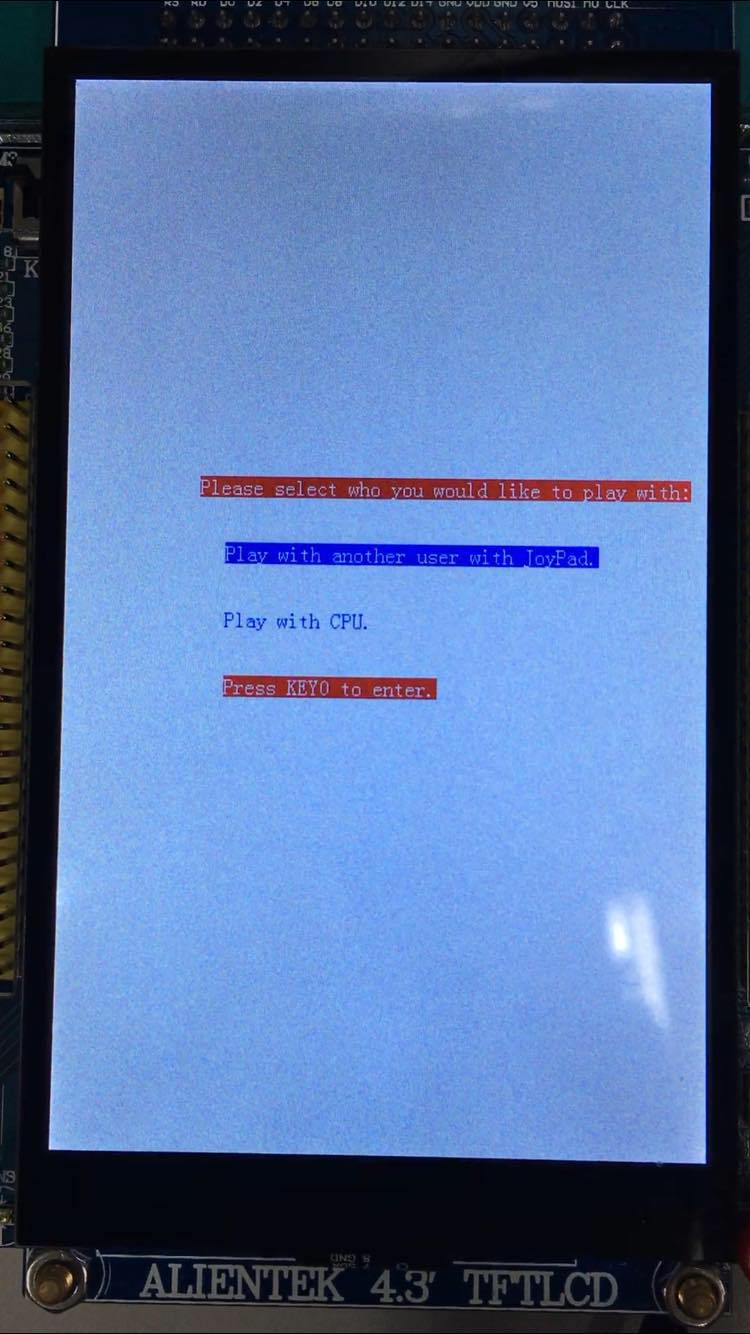
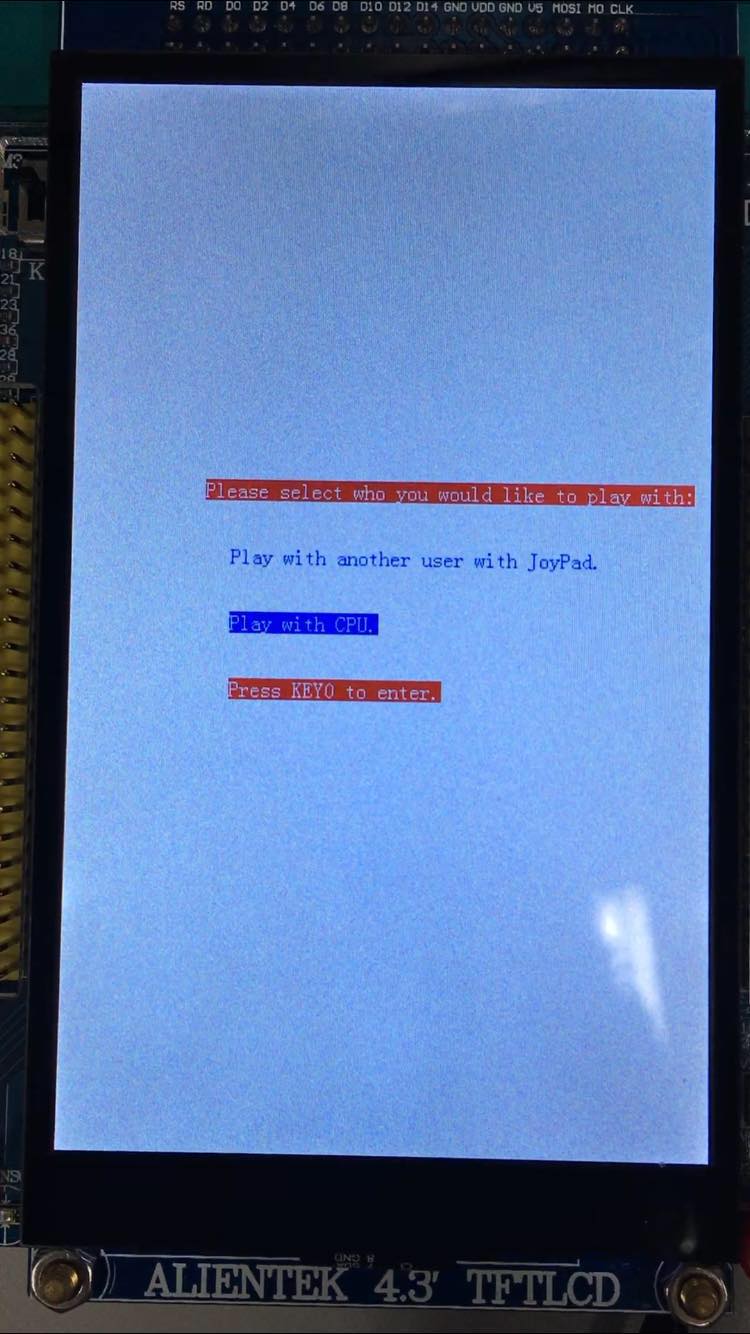
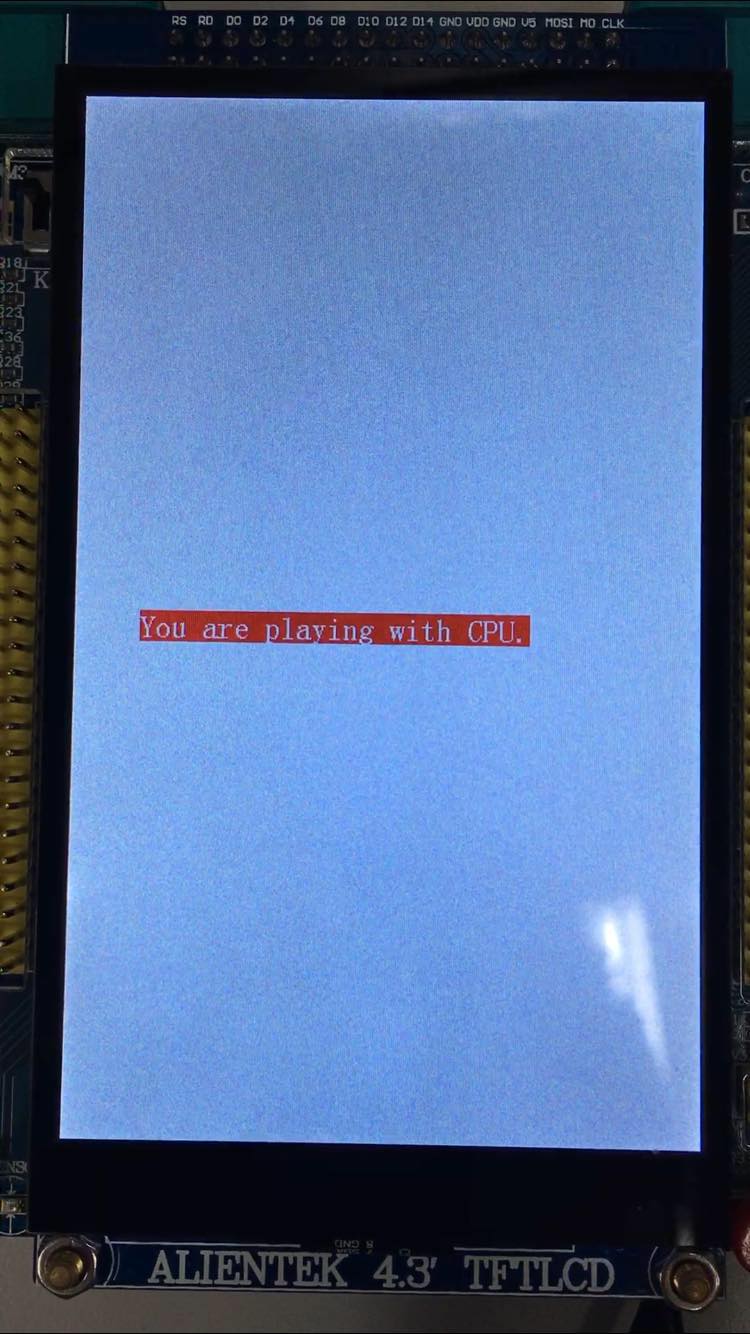
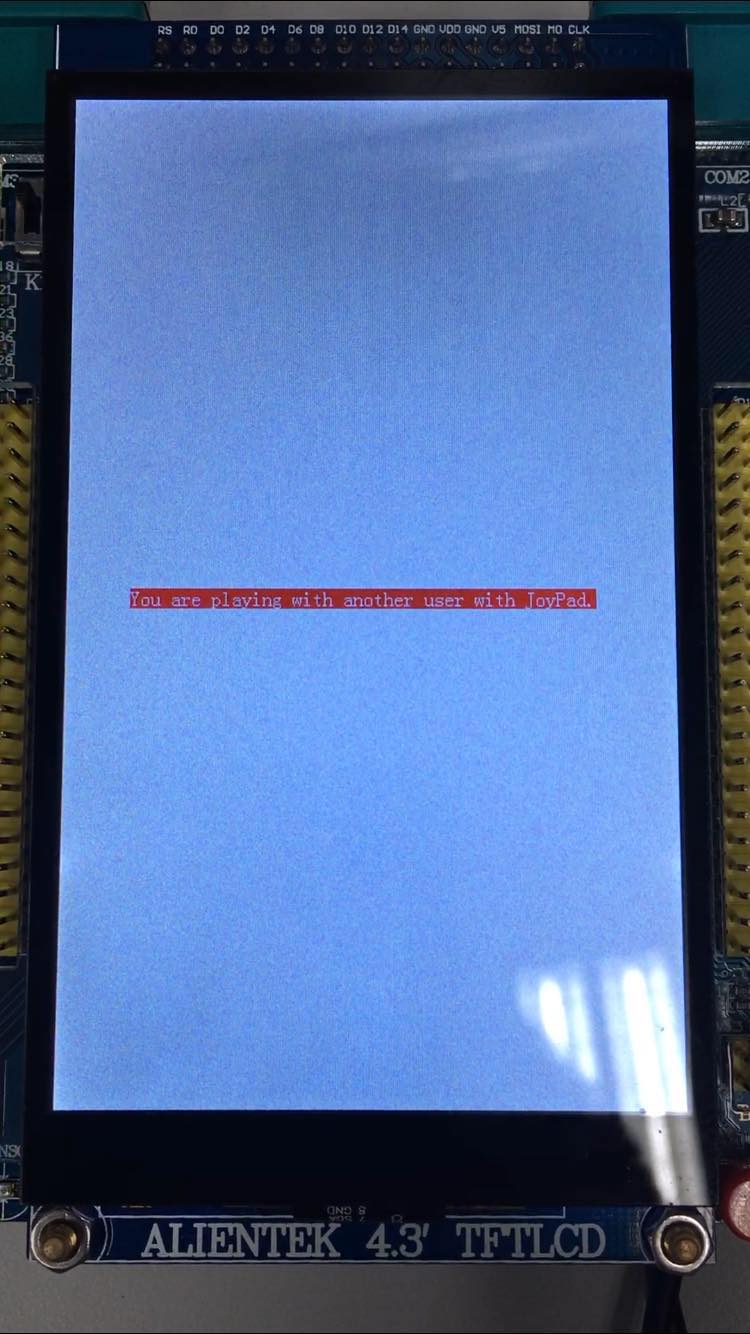


Figure 31. CPU or JOYPAD selection Page



图片包含 电子产品

已生成极高可信度的说明Figure 32. Indicating CPU or JOYPAD mode

Figure 33. Acceleration

* **Conclusion:**

We successfully achieved 3 improvements as shown above.

**Appendix:**

Because some of the features, such as Acceleration, invisible white and Pause, are hard to see from the snapshots, so I upload the video