《Embedded System and Microcomputer Principle》Project Report

Topic	**TBG Battle System**					
Group No.	**1**					
Member 1	**SID**	**12010820**	**NAME**	**黄日涵**	**Coefficient**	**1**
Member 2	**SID**	**12011215**	**NAME**	**何叶韬**	**Coefficient**	**1**
Member 3	**SID**	**12112328**	**NAME**	**郑祖彬**	**Coefficient**	**1**
Member 4	**SID**	**12110817**	**NAME**	**张展玮**	**Coefficient**	**1**

I. System Function

We're designed a TBG battle system. The TBG battle system allows players to select attack, defense, skills, or props at each turn in order to defeat their opponents. The system features a range of customizable game characters, each with their own unique attributes such as name, head portrait, attack power, defense power, and special skills. Players can select their characters using the touch screen interface, and the attributes of the selected characters are displayed on the LCD screen.

During gameplay, players can use the touch screen to attack, defend, and use special skills against their opponents. The system tracks the life points (HP) of each character and determines victories and defeats based on when a character's HP is equal to or less than 0. The information of victory and defeat is displayed on the LCD screen.

The TBG battle system also includes a wireless communication module that allows for close range multiplayer matches. Players can use their own miniSTM32 boards to battle against each other in real-time, with the screen of each side displaying the information of both players. The system uses a reasonable communication scheme to ensure the accuracy of the data transmitted during the online matchmaking process.

II. System Design

1. working principle

(a) The TBG battle system is initially in the character selection state, allowing players to choose their characters and display their attributes on the LCD screen.

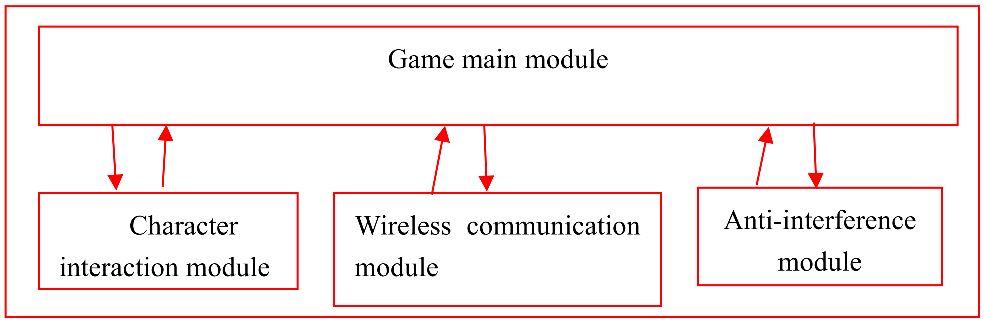
(b) Players use the touch screen to make their choices for each turn, including attacking, defending, and using special skills.

(c) The game uses a wireless communication module to enable players to battle each other on two miniSTM32 boards. The screen of each board displays the information of both players.

(d) During the battle, the system keeps track of the characters' HP and judges the winner when one character's HP is equal to or less than 0. The information of victory and defeat is displayed on the LCD screen.

(e) After the battle, the system returns to the character selection state, allowing players to choose new characters or play again with the same characters.

2. system frame diagram



3.Sub-module design

The system is divided into 4 sub-modules, they are the wireless communication module, the anti-interference module, the game main module and the character interaction module.

(a) Wireless communication module

To implement wireless communication, a NRF24L01 chip was used. It could transmit and receive messages from the MCU via SPI. Since on miniSTM32, the NRF module and W25Q64 share the same SPI, we needed to make use of time-sharing multiplexing. Each time the NRF24L01 can only work in one state, i.e. transmission mode or receiving mode. Given that we want two miniSTM32 to communication with each other, a proper configuration of NRF's mode is vital. Here, we implemented communication in non-interrupt method. In the TBG system, we defined a concept called "turn", during which players are prompted to take actions like attacking or using ultimate skills. In other words, if a player is not in his turn, he will wait for the action of the opponent and the miniSTM32 will wait to receive the message. Thus, the NRF24L01 module were configured to transmission mode during the "turn" and vice versa configured to receiving mode outside the "turn".

(b) Anti-interference module

Consider that all the students use the NRF24L01 to implement their project, interference could occur since the communication signal frequency will be around 2.4G Hz. Thus, we implemented an anti-interference module to analyze whether the message we received is legal or not. In this module, we set a prefix before each message. After our miniSTM32 received it, it will first examine whether the prefix is correct or not, it will only response to the message with exactly correct prefix. Thus, our module could avoid dummy command effectively.

(c) Game main module

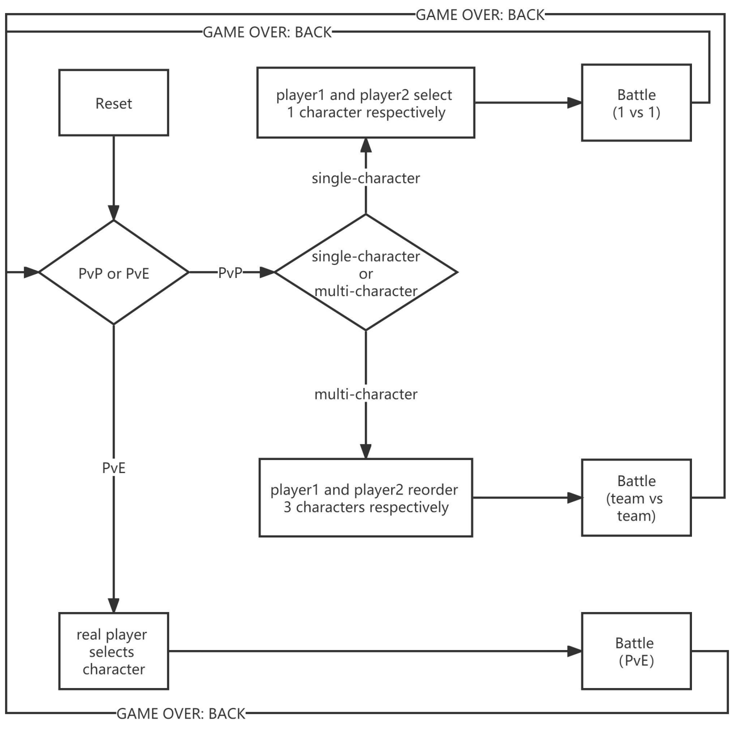

GAME OVER: BACK
GAME OVER: BACK
Reset
player1 and player2 select 1 character respectively
Battle (1 vs 1)
single-character
PvP or PvE
PvP
single-character or multi-character
multi-character
PvE
player1 and player2 reorder 3 characters respectively
Battle (team vs team)
real player selects character
Battle (PvE)
GAME OVER: BACK

Diagram: the structure of game main module

The structure of game main module is shown as the above picture.

In this module, it manages the different stages of the game. First of all, players should choose PvP or PvE. Without loss of generality, take PvP with single-character as example. Before the formal battle, each player can choose to be player1 or player2, but not the same. After confirming the identity of player, the player can only touch the half part of the screen during “Selecting character stage” and “Battle stage”. The player can not do operations on the opponent’s screen in the other half part. At the same time, the information that opponent does on the screen can be shown on the player’s screen simultaneously. In “Selecting character stage”, each character will blink until the player finally touches the corresponding head portrait to decide(More different characters to be selected). In case of touching by mistake or secondary selection, the player can change his character by touching another head portraits after the first selecting. After both players selecting their characters, it moves to the next stage, i.e. “Battle stage”. In one player’s turn , the corresponding head portrait will blink until the player finally touches the corresponding regions to decide which operations to do. The operation can be a normal attack, an aid attack(More attack or defense skills for one character), an active defend or a unique attack. Also, the four tools has the possibility of 40% to appear in each turn, where a weapon has the possibility of 10%, a shield has the possibility of 10%, an aid bag has the possibility of 15% and an Easter egg has the possibility of 5%. The character will evolve once every two rounds(Character development). When HP of the character belonging to one player is equal to or less than 0, the player will be judged to have lost, and the information of victory and defeat will display on LCD screen(Winning and losing judgment). The player can touch BACK to return the “PvP or PvE” Interface to play again.

As for the PvP with multi-character(Multi-character teams), players need to arrange the appearance order of 3 characters. The operation in “Battle stage” is the same as the PvP with single-character. In one battle, each player can use 3 character. When HP of the first character is equal to or less than 0, the next character is automatically summoned as the appearance order is arranged before. When HP of the last character is equal to or less than 0, the player will be judged to be lost.

As for the PvE, one side is a real player and the other side is a machine player. The machine player operates randomly during "Selecting character stage" and "Battle stage"(PvE).

In this module, it also manages specific regions which can be touched to implement the corresponding function(Touch screen operation). All operations during the game, including choosing game mode PvP or PvE , choosing game sub-mode single-character or multi-character, selecting or reordering characters, attacking, defending, releasing skills, taking tools in the battle, going back and so so. The game is completed by using touch screen.

(d) Character interaction module

The character interaction module is implemented in the "pkm_character.c" and its claim is in the "pkm_character.h". This module is a software module, without any interaction with hardware. It is used to provide and manage different characters.

In the module, there are three kinds of characters in total. They have different attributes and images. Their data are saved in a self defined structure type, the type of which is named "struct character". In order to minimize the memory consumption, the structure type only preserves int type data, including an important attribute named "int id". All the functions to be called are designed to identify the "id" of the structure firstly. By doing so, it is able to know what kind of character it is and then respond in the corresponding ways. Besides, in order to transfer the original structure to the functions rather than copying a new structure from the transferred structure, only the counter of the structure is allowed to be transferred in the functions.

In order to update the counterpart's parameters, while minimizing the data transferred between each player, there would be two characters in a game. The one is the player controlling the hardware, the other one is the counterpart. When receiving information, updating the self and counterpart's character by transferring them to the corresponding functions.

Every time the character need to change its state, just call the corresponding functions.

There are three attacking functions, including a normal attack, an aid attack and a unique attack. The unique attack would be different with different characters. The attack value of aid attack is only the half of the normal attack, but would increase the shortage attributes. The attack value of normal attack might cause a critical attack, which would increase 10 percent of the normal attack value. The possibility of that is 25%.

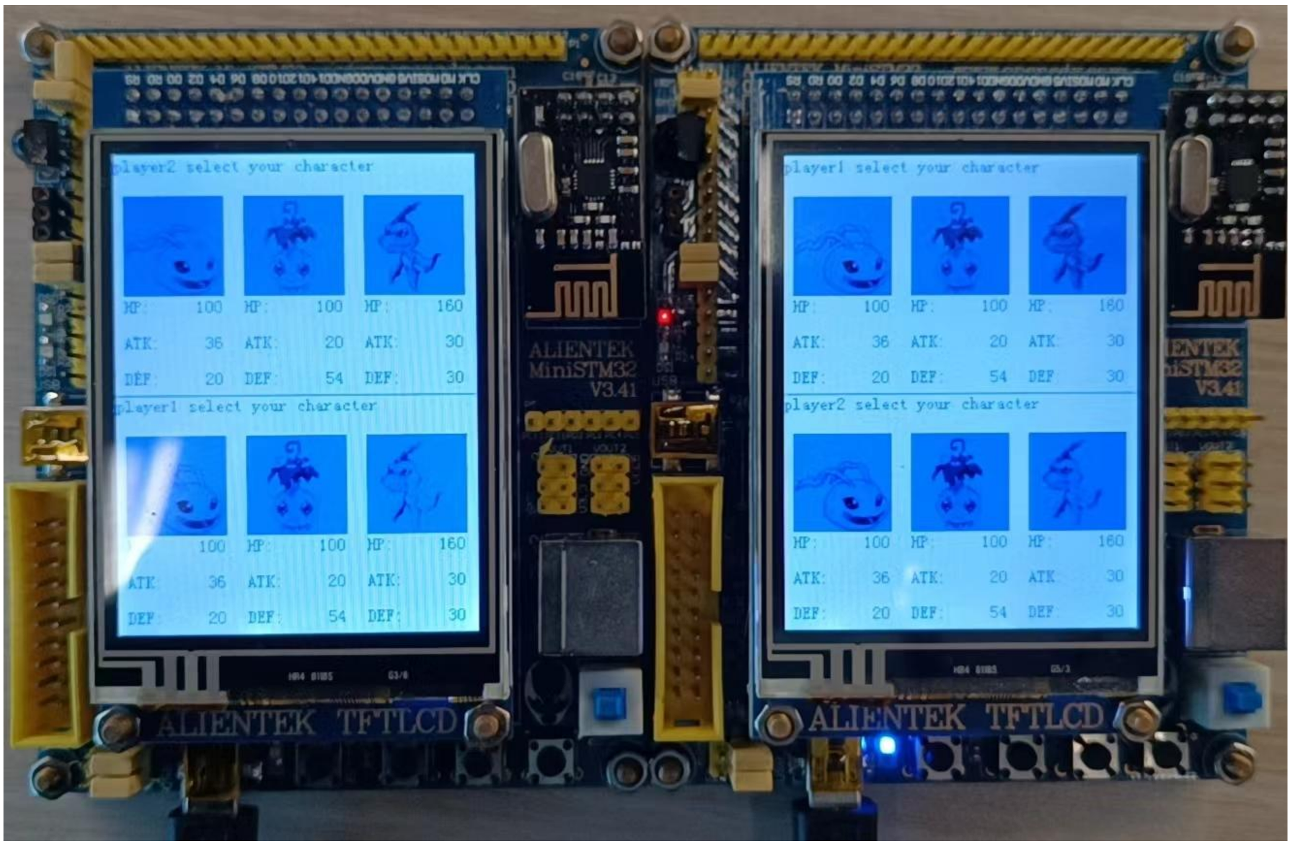
There are two defending functions, including a passive defend and an active defend. The passive defend would be called automatically, if the active defend is not used. The defend value of passive defend is the half of the active defend.

There are four tools to be used by characters, including a weapon, a shield, an aid bag and an Easter egg. The weapon would increase the attack value in the next turn. The shield would increase the defend value in the next turn. The aid bag would increase the current life. The Easter egg would provide a chance to revive.

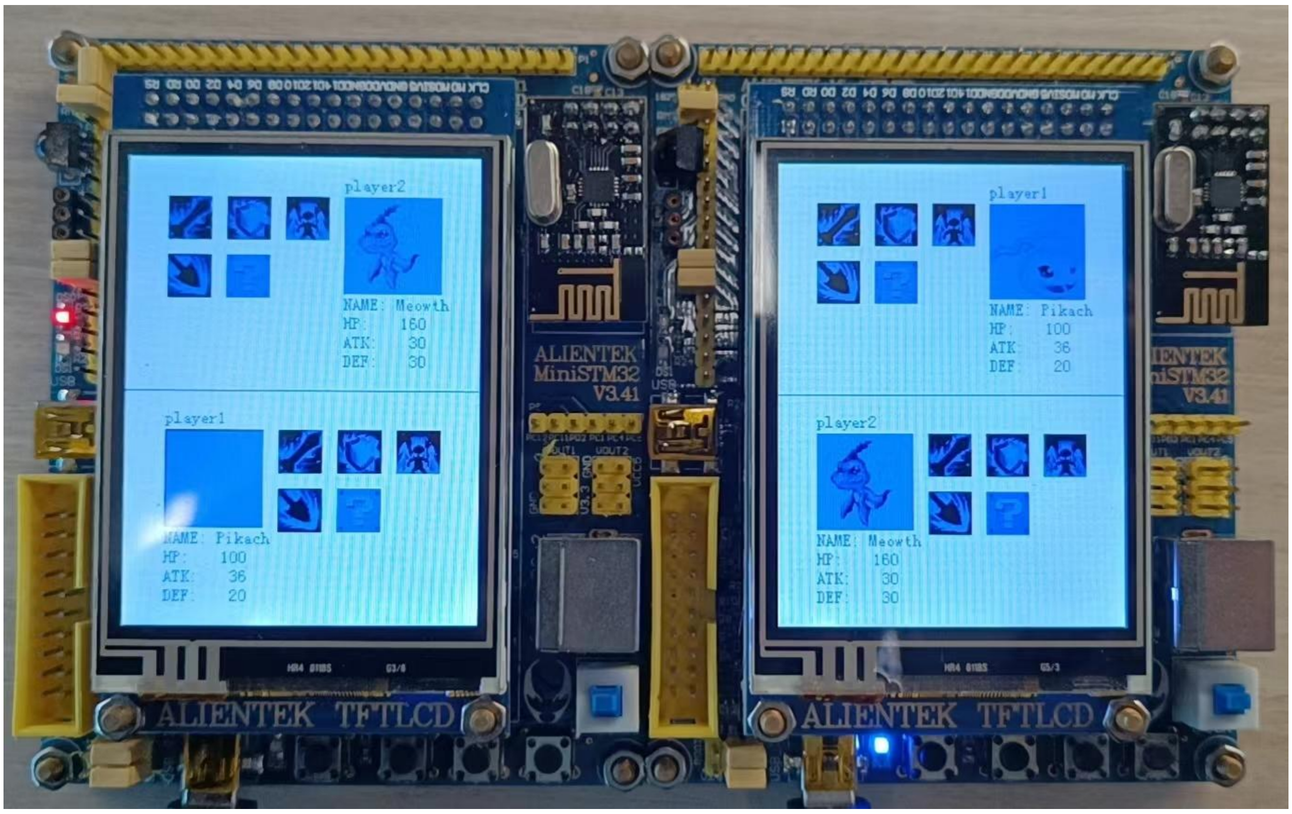
Eventually, the character is allowed to upgrade, increasing the basic attributes.

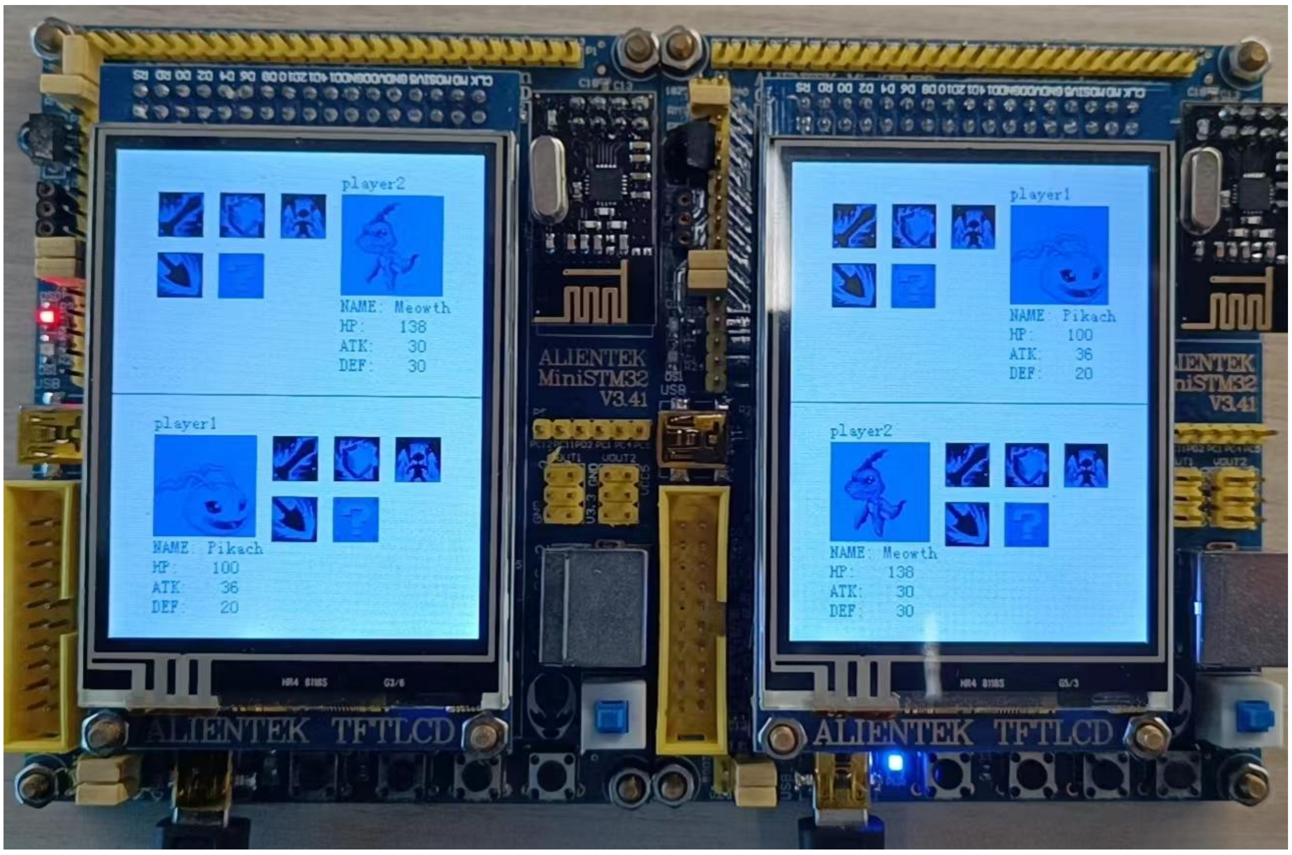
III. Results (screenshots and hardware photos)

(1) Game characters and attributes design (15%) + More different characters to be selected (5%)

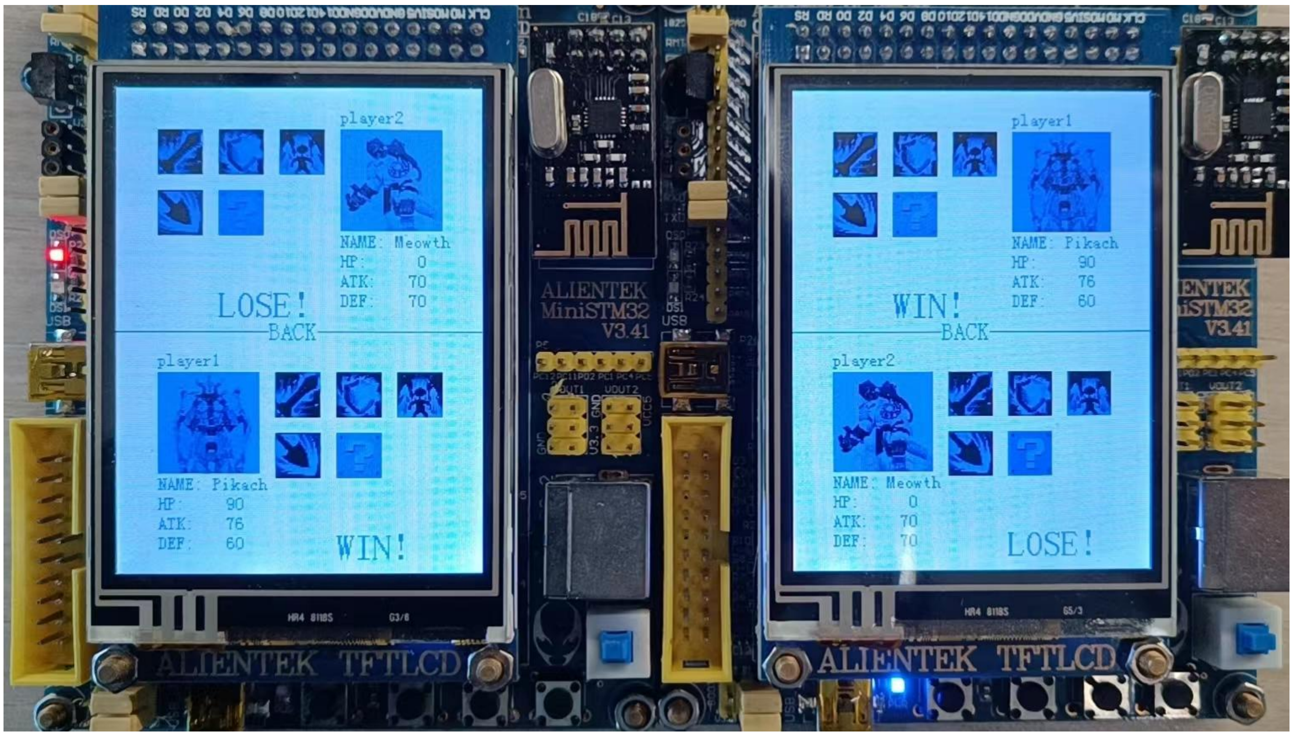


(2) Battle via wireless module (30%) + Touch screen operation (10%)





(3) Winning and losing judgment (5%)



(4) More attack or defense skills for one character (5%)

a normal attack:

an aid attack:

an active defend:

a unique attack(special skills):

(5) Diversified skill props design and reasonable numerical design (10%)

a weapon:

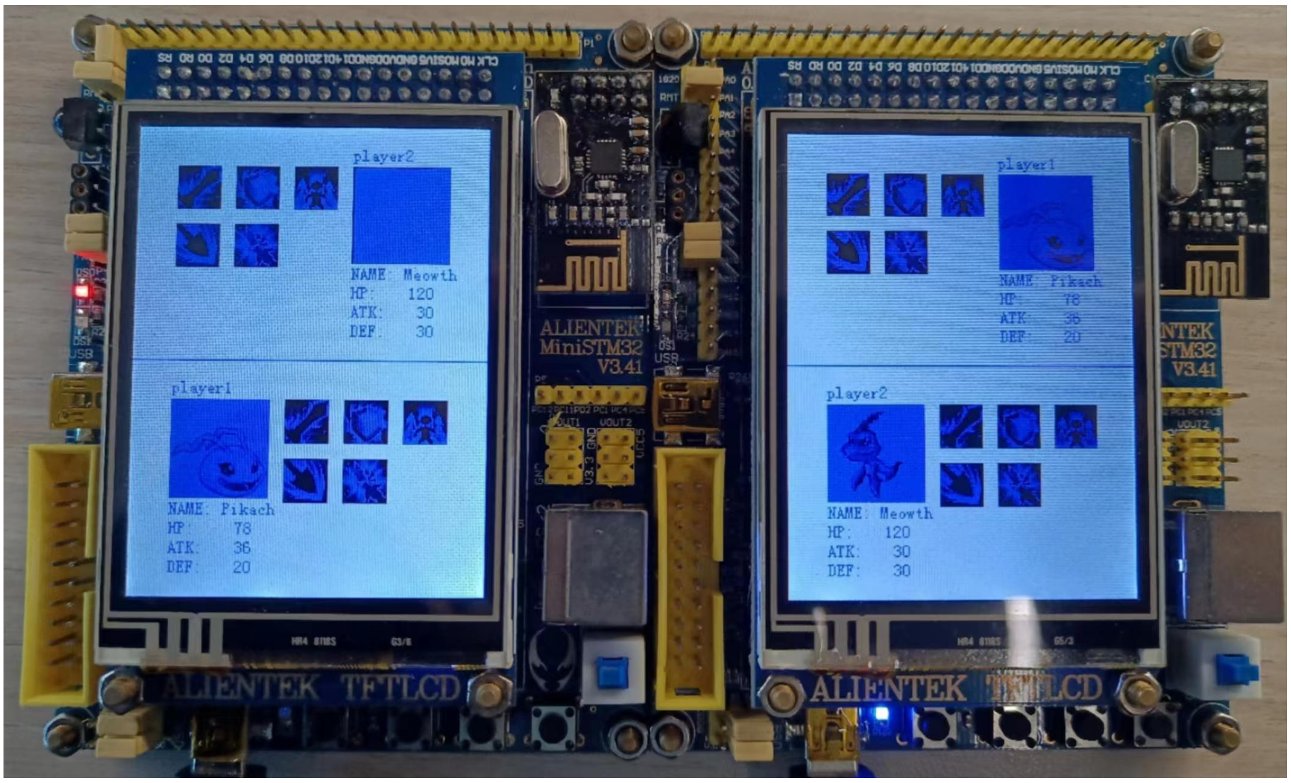
a shield:

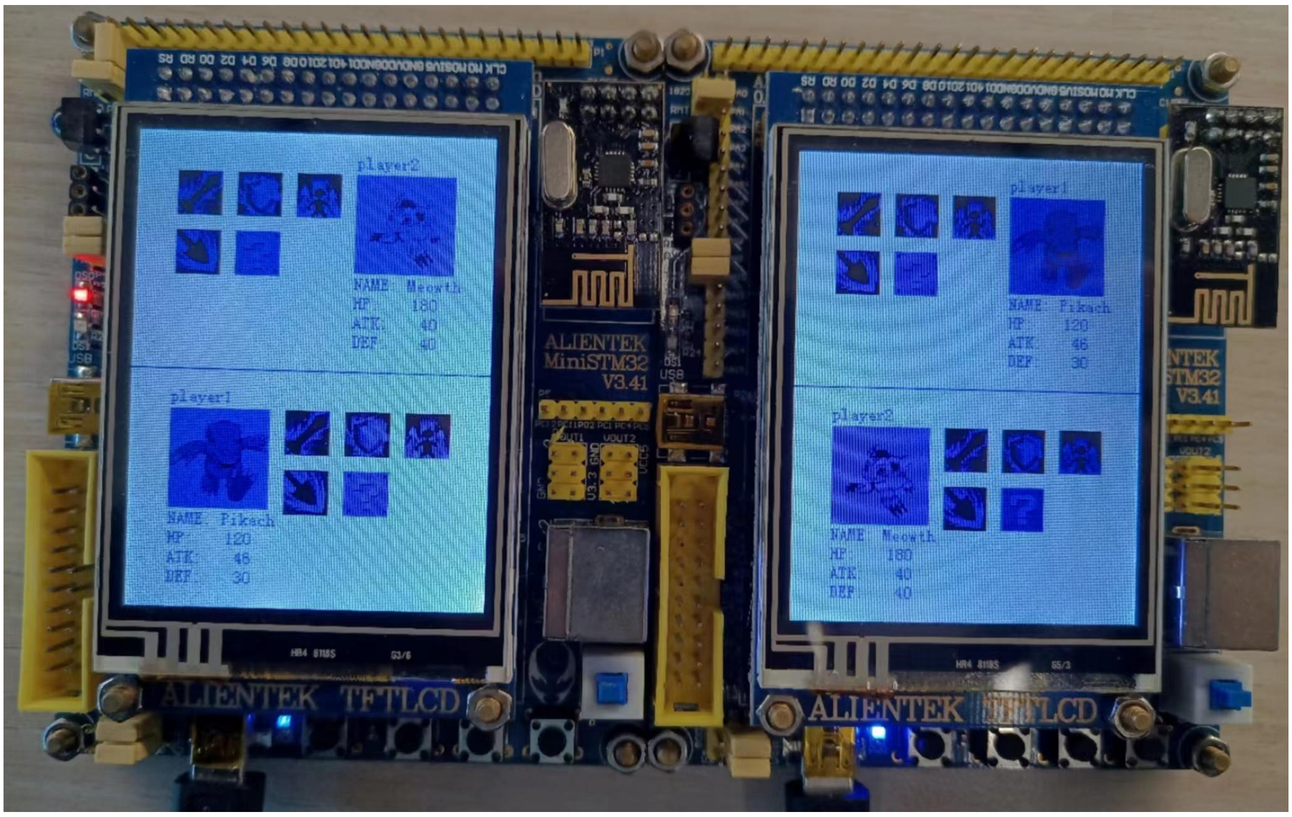
an aid bag:

an Easter egg:

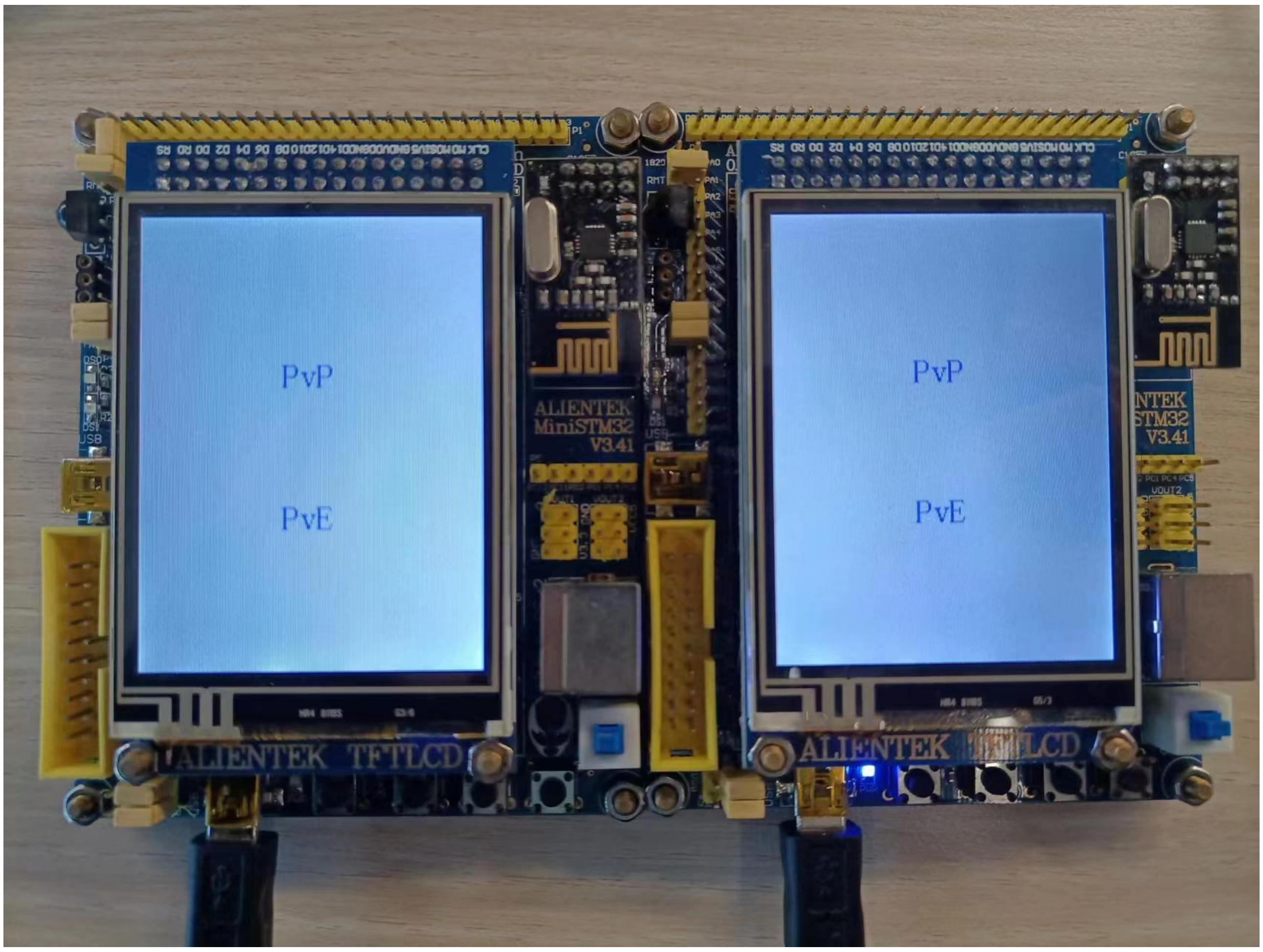
Default:

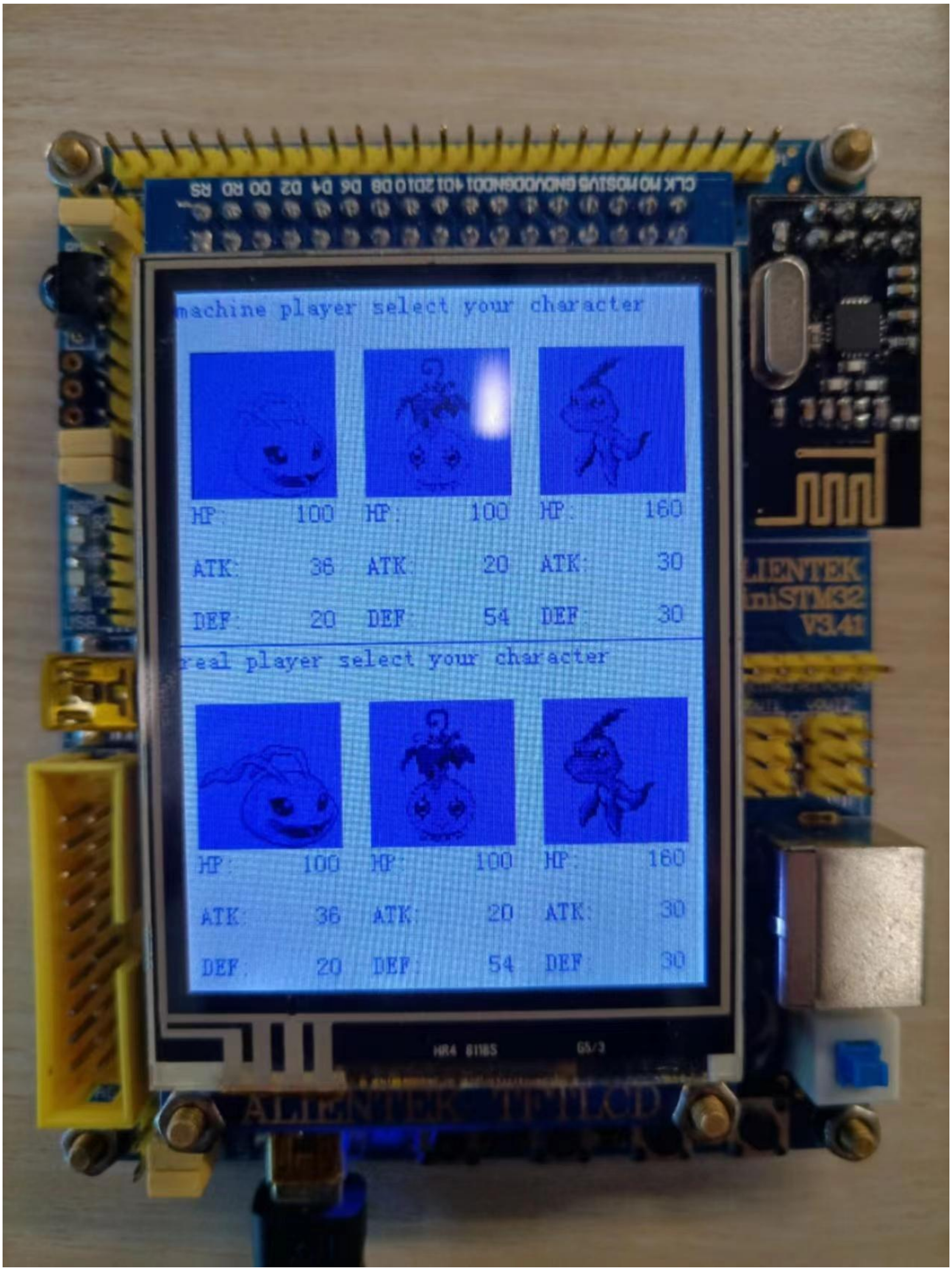
(6) Character development (5%)

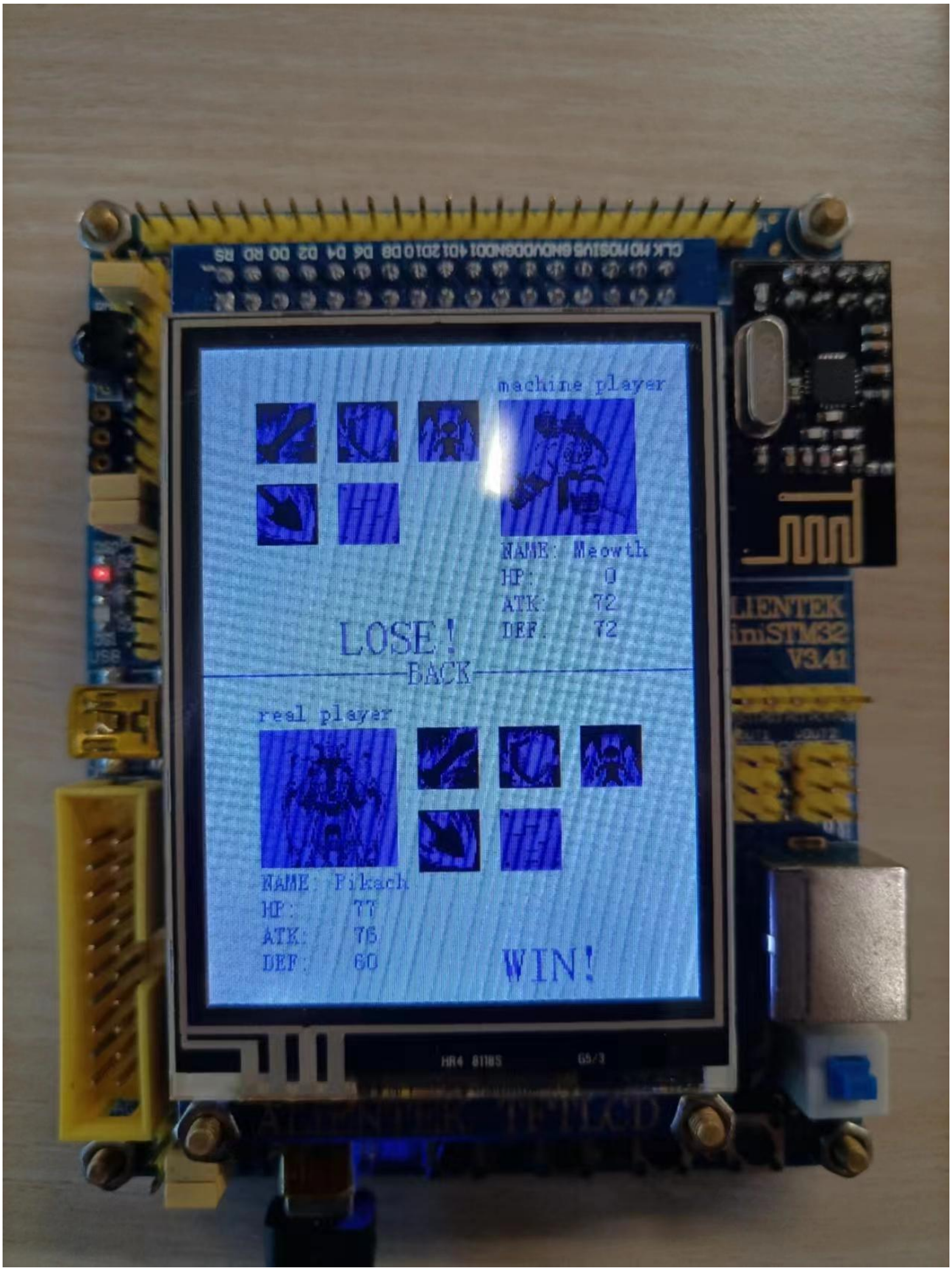

player2
NAME: Meowth
HP: 120
ATK: 30
DEF: 30
player1
NAME: Pikach
HP: 78
ATK: 36
DEF: 20
ALIENTEK MiniSTM32 V3.41
ALIENTEK TFTLCD


player2
NAME: Meowth
HP: 180
ATK: 40
DEF: 40
player1
NAME: Pikach
HP: 120
ATK: 46
DEF: 30
ALIENTEK MiniSTM32 V3.41
ALIENTEK TFTLCD

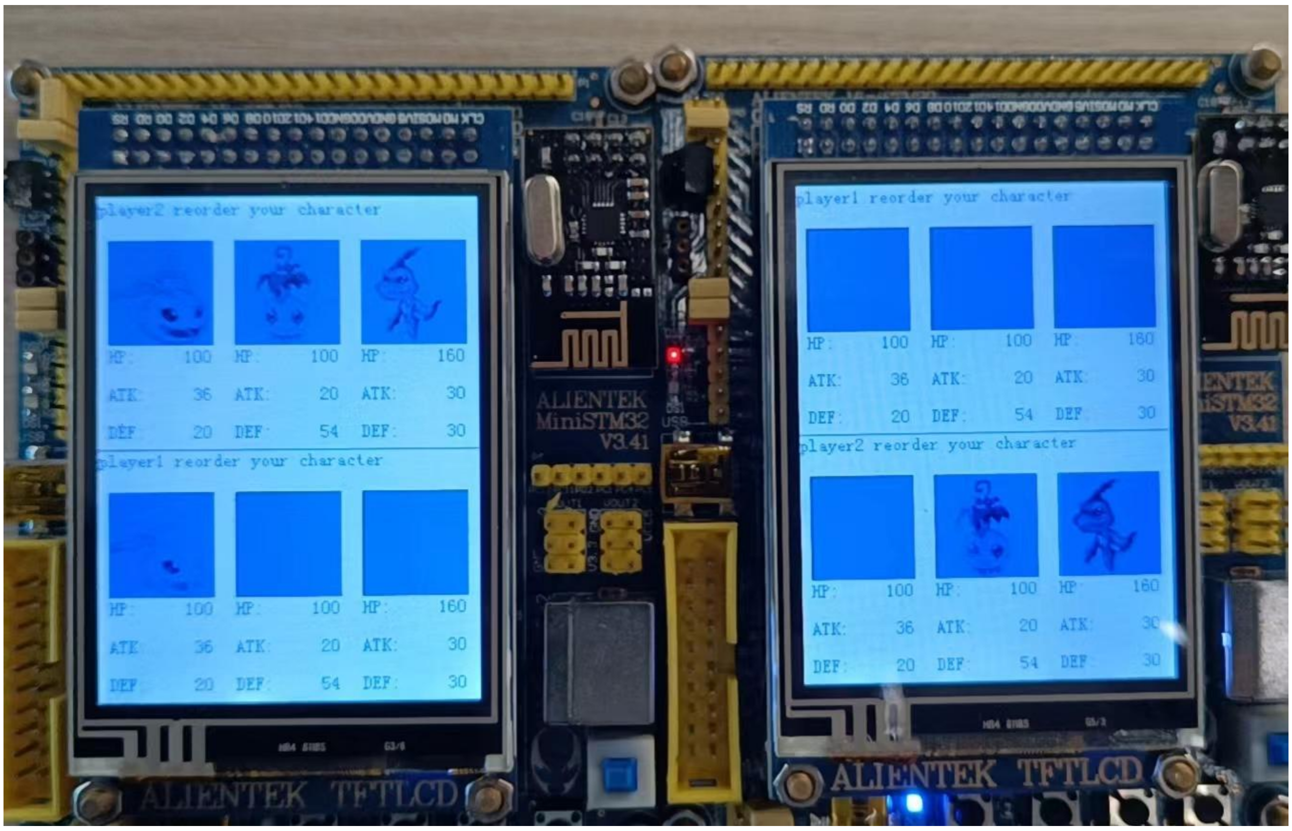
(7) PvE (5%)

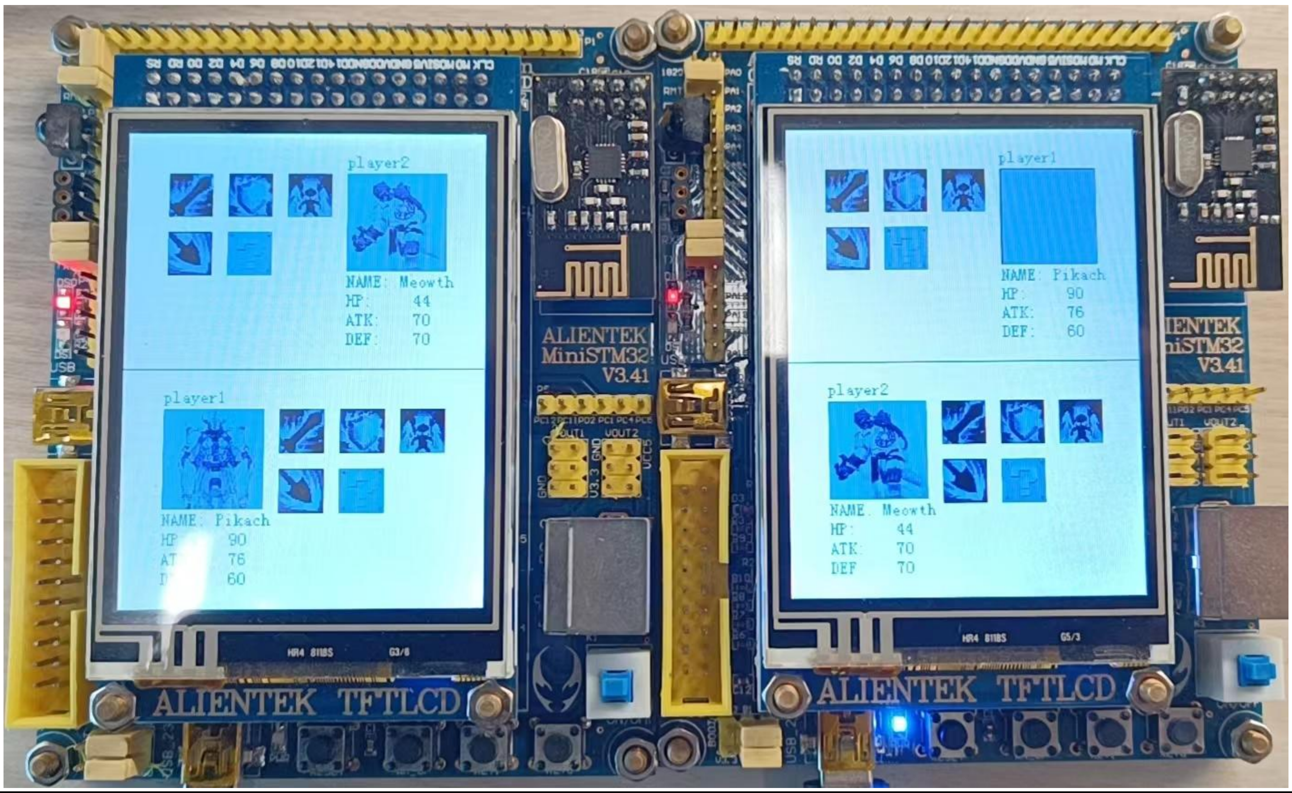

PvP
PvE
PvP
PvE
ALIENTEK TFTLCD
ALIENTEK MiniSTM32 V3.41
ALIENTEK TFTLCD

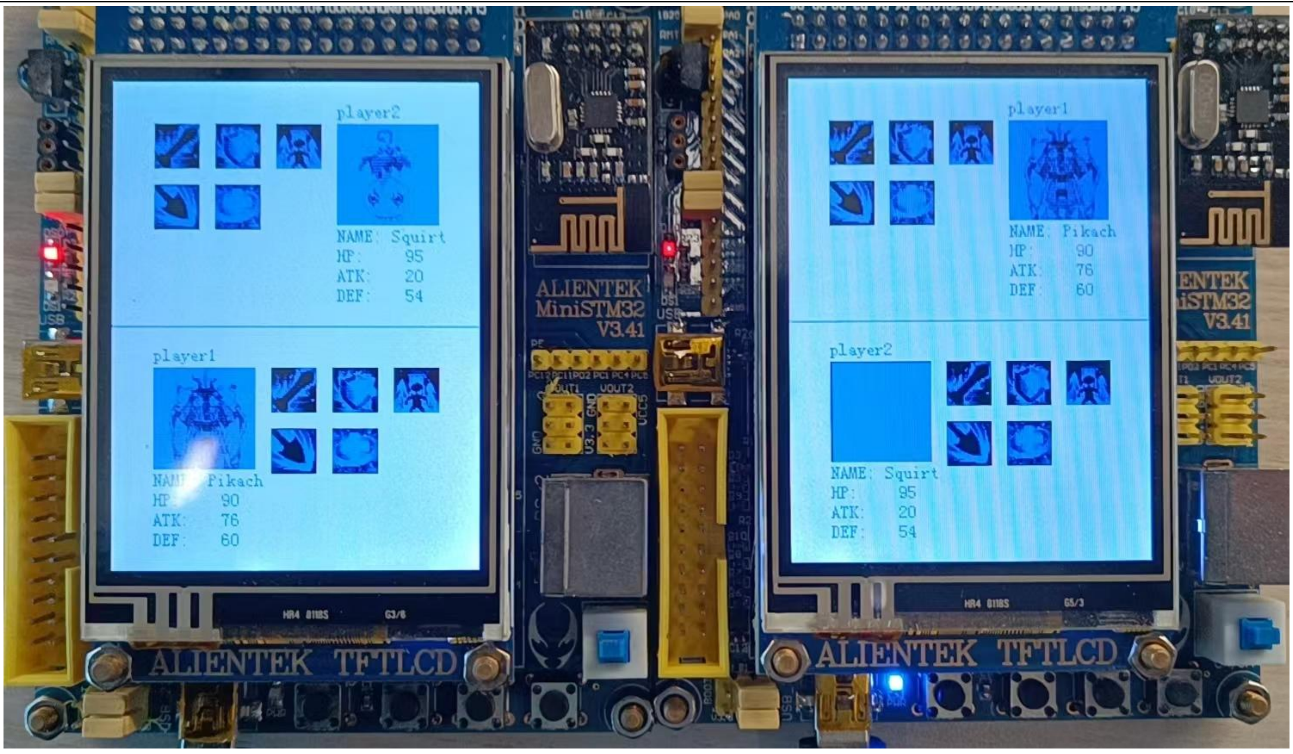

machine player select your character
HP: 100
ATK: 36
DEF: 20
HP: 100
ATK: 20
DEF: 54
HP: 160
ATK: 30
DEF: 30
real player select your character
HP: 100
ATK: 36
DEF: 20
HP: 100
ATK: 20
DEF: 54
HP: 160
ATK: 30
DEF: 30
ALIENTEK MiniSTM32 V3.41
ALIENTEK TFTLCD

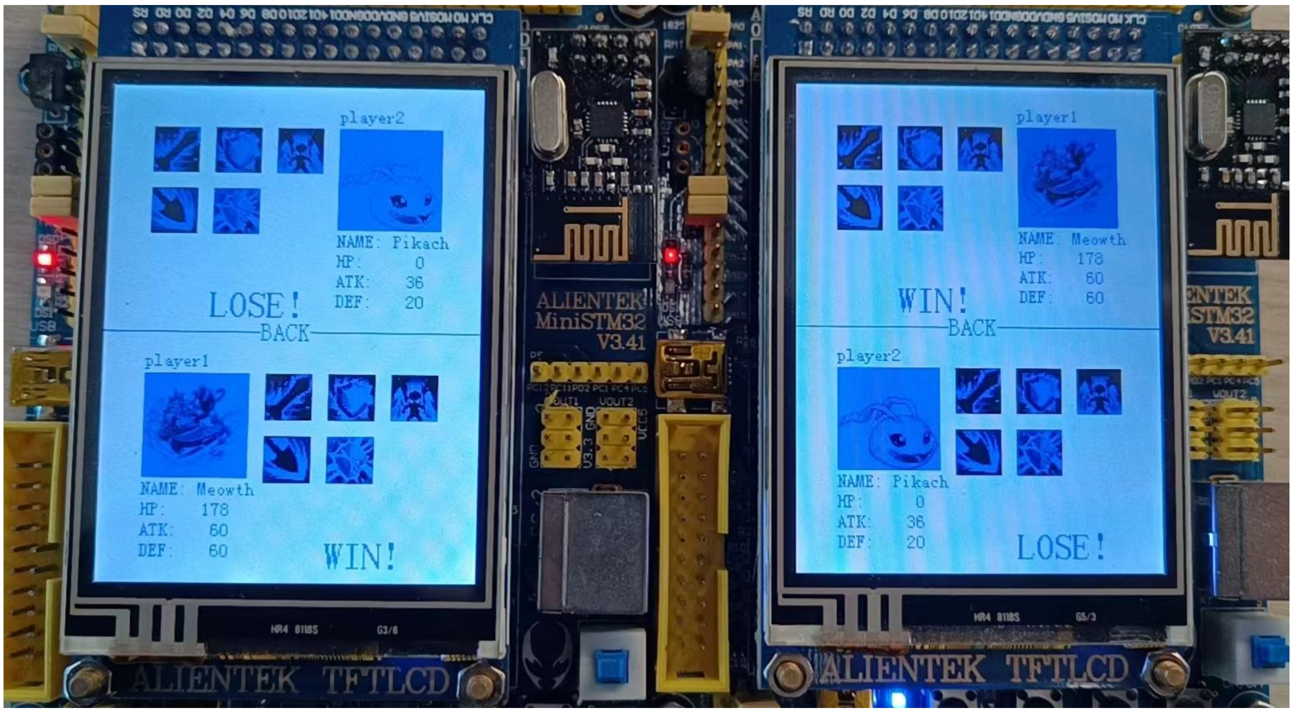


(8) Multi-character teams (5%)

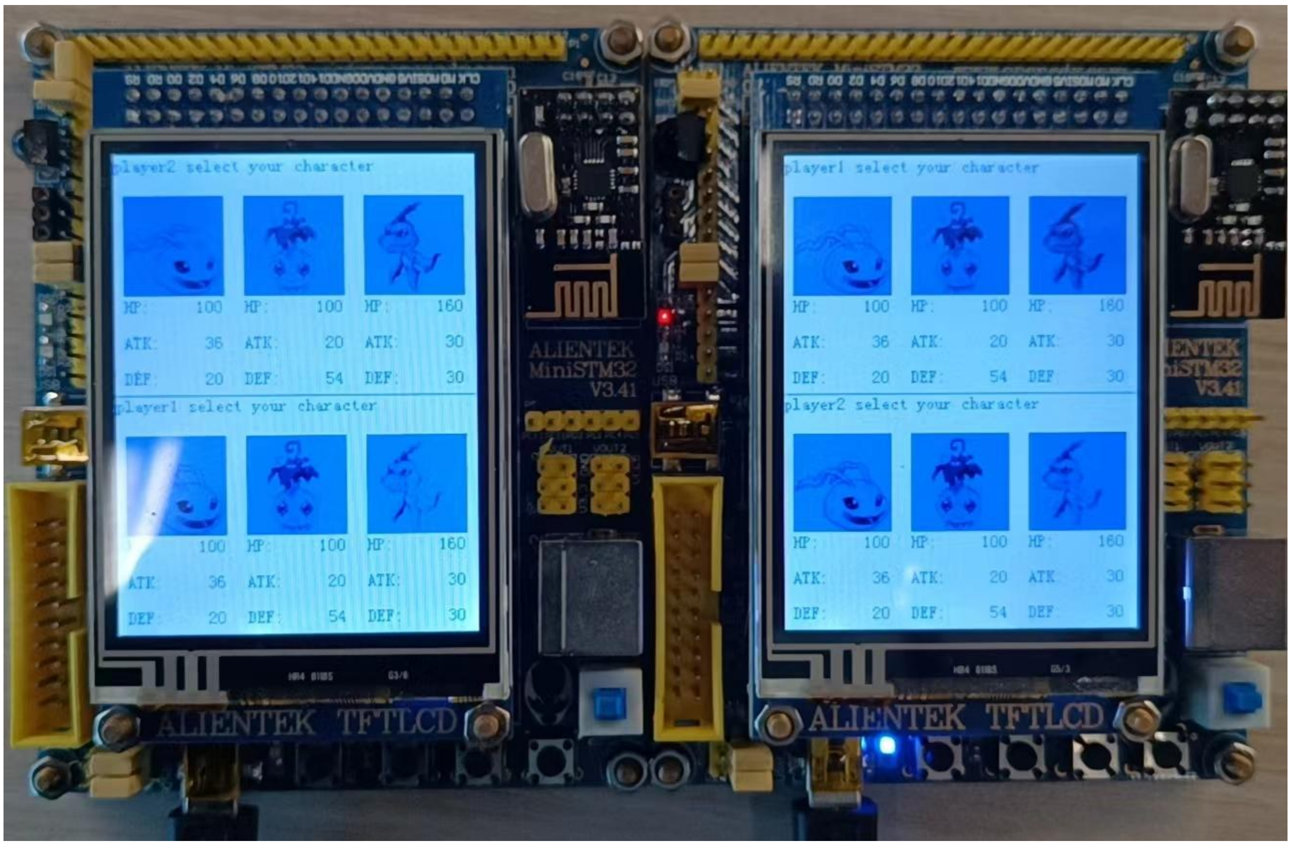

player2 reorder your character
HP: 100 HP: 100 HP: 160
ATK: 36 ATK: 20 ATK: 30
DEF: 20 DEF: 54 DEF: 30
player1 reorder your character
HP: 100 HP: 100 HP: 160
ATK: 36 ATK: 20 ATK: 30
DEF: 20 DEF: 54 DEF: 30
ALIENTEK MiniSTM32 V3.41
ALIENTEK TFTLCD
player1 reorder your character
HP: 100 HP: 100 HP: 160
ATK: 36 ATK: 20 ATK: 30
DEF: 20 DEF: 54 DEF: 30
player2 reorder your character
HP: 100 HP: 100 HP: 160
ATK: 36 ATK: 20 ATK: 30
DEF: 20 DEF: 54 DEF: 30
ALIENTEK TFTLCD


player2
NAME: Meowth
HP: 44
ATK: 70
DEF: 70
player1
NAME: Pikach
HP: 90
ATK: 76
DEF: 60
ALIENTEK MiniSTM32 V3.41
ALIENTEK TFTLCD
player1
NAME: Pikach
HP: 90
ATK: 76
DEF: 60
player2
NAME: Meowth
HP: 44
ATK: 70
DEF: 70
ALIENTEK TFTLCD





(9) Some transition animation or delicate UI (5%)


player2 select your character
HP: 100 HP: 100 HP: 160
ATK: 36 ATK: 20 ATK: 30
DEF: 20 DEF: 54 DEF: 30
player1 select your character
player1 select your character
player2 select your character
ALIENTEK MiniSTM32 V3.41
ALIENTEK TFTLCD

IV. Work allocation description

(1) Member 1: Huang Rihan implemented the wireless communication module and help modified the main code so that two devices can transmit message and users can play the battle game in real time.
(2) Member 2: He yetao, finished the character interaction module.
(3) Member 3: Zheng Zubin, finished the Game main module.
(4) Member 4: Zhang Zhanwei completed the player selection function and part of the report.

V. Problems encountered and solutions

Problem 1: When one party sends an attack, how do the two parties change the state synchronously?
Solution: Stm32 can only transmit strings. If all our attributes (HP, ATK, DEF, etc.) and props are transmitted, it will increase the processing difficulty. We take advantage of the feature that both sides store their own and the other's skills. When the attacker sends an attack, he only needs to transmit the number represented by the attack type to the target side to apply the attack type.

Problem 2: Difficulty implementing the wireless communication module.
Solution: Research and experiment with different wireless communication technologies (such as 2.4G, Bluetooth, or wi-fi) to determine the best solution for your system. Implement error-checking and data validation techniques to ensure reliable data transmission between the miniSTM32 boards.

Problem 3: Difficulty designing and implementing the game logic.
Solution: Break the game down into smaller, more manageable tasks and focus on implementing each one individually. Utilize debugging tools and techniques to identify and fix any issues in the game's logic.

VI. Summary & experience

Summary:

In this project, we designed and implemented a turn-based game using miniSTM32 boards. The game featured customizable characters with various attributes, touch screen operation, winning and losing judgment, and wireless communication for close range multiplayer matches. We chose to use the 2.4G wireless communication module and designed a reasonable communication scheme to ensure real-time and accurate data transmission during online matchmaking.

Experience:

Working on this project was a valuable learning experience for us. We gained hands-on experience in designing and programming an embedded system, as well as implementing touch screen operation and wireless communication. We also learned about the challenges and limitations of embedded systems, such as power constraints and the need for reliable performance.

One of the most challenging parts of the project was designing the communication scheme for the wireless multiplayer matches. We had to consider factors such as transmission range, data rate, and interference in order to ensure that the matches ran smoothly.

What’s more, we know how teamwork can be beneficial in a project. Working as a team allows you to divide the work among team members, which can be more efficient and allow each person to focus on their strengths. Team members can share their knowledge and ideas with each other, which can lead to more creative solutions and a better overall result. Effective teamwork requires good communication, which can improve overall project performance by ensuring that everyone is on the same page and knows what is expected of them.

Overall, we enjoyed working on this project and feel that we have gained valuable skills and knowledge that will be useful in future projects and endeavors.