UM–SJTU Joint Institute

VE373 Design of Microprocessor Based Systems

Battle of Tanks based on Pic32 Microprocessor

Project Report

Name: Shen Tianhong ID: 515370910151

Name: Tang Qianyong ID: 515370910096

Name: Wu Yiming ID: 5143709243

Date: August 18, 2018

1 Abstract

In this project, we develop a device that can run a video game. The rule of the game is like the classic game, BATTLE of TANKS, which is detailed described in Section.4. To run the game, we use a PIC32 microprocessor. Each player can use two joystick to control the tank on the screen. And PIC32 controls the LCD display through an Arduino Maga board.

The module of the PIC32 we use in our project include PWM, Serial Communication and ADC.

2 Component Level Design

2.1 Overall Structure

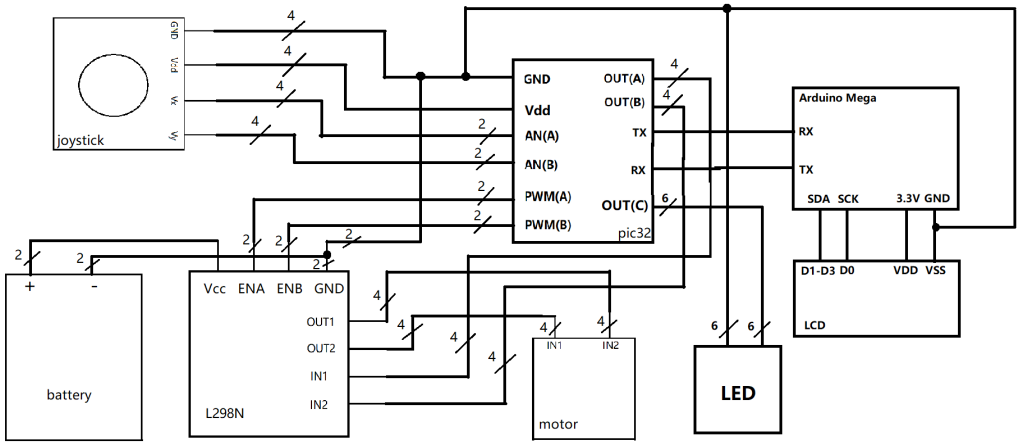

Figure 1: Component Level Overall Structure

For the component overall structure, we used 4 joysticks, 4 vibrating motors, an LCD displayer, 2 L297N driving board, one piece of battery, an Arduino Mega board and a PIC32 board. One joystick and one vibrating motor together forms one hand shank.

2.2 Hand Shank

The hand shank contains two parts, joysticks and motors. Joysticks of the hand shank are used to control directions, while motors are feedback system of the game which enhance the players' experience.

Figure 2: Hand Shank

Figure 3: Inside Structure

2.2.1 Joystick

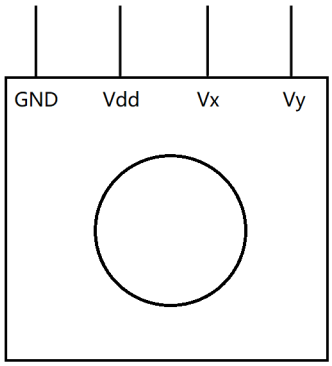

Figure 4: Component diagram of joystick

The joystick component have four pins. GND is the ground pin, Vdd is the input voltage pin, ranging from 0 to 5V(rated). Vx and Vy are two analog output pins, indicating the position of the joystick. Table below shows Vx and Vy with corresponding direction.

Direction	**Vx**	**Vy**
up	0	/
down	Vdd	/
left	/	Vdd
right	/	0
rest	Vdd/2	Vdd/2

2.2.2 Vibration Motor

The vibration motor has two inputs. When $V_{IN1} > V_{IN2}$, motor rotates positively. When $V_{IN1} < V_{IN2}$, motor rotates reversely. When $V_{IN1} = V_{IN2}$, motor stops. Motor is driven by L298N driving board.

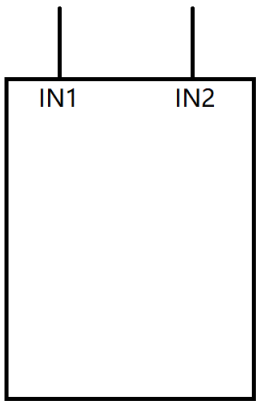

Figure 5: Component diagram of vibration motor

L298N Driving Board ENA and ENB are two input ports that control the voltage between OUT1 and OUT2. ENA and ENB are controlled by PWM signal. IN1 and IN2 control the rotation direction of the left motor, while IN3 and IN4 control the rotation direction of the right motor. In out project, since we are using a vibration motor, the rotation direction is not concerned, so we simply connect IN1 & IN2 and IN3 & IN4 to a 5V and GND.

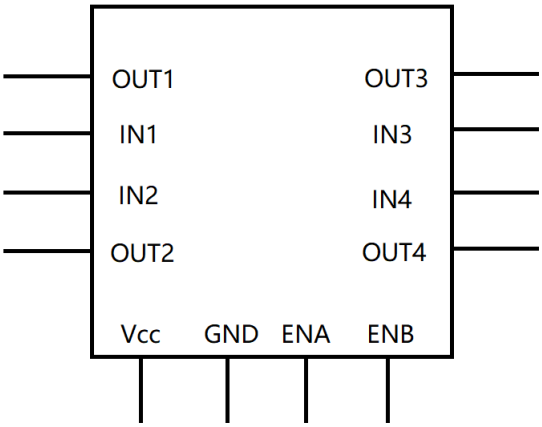

Figure 6: Component diagram of L298N driving board

2.3 LCD-Arduino

2.3.1 LCD

We use a 256 by 160 pixels LCD screen to display our game. The model of the LCD is JLX256160G-680, with a ST75256 controller inside. The diagram of the screen is shown below:

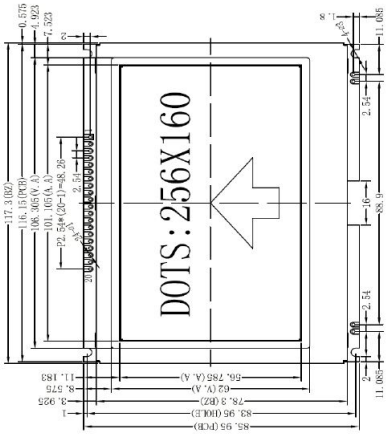

Figure 7: The JLX256160 LCD screen.

Inside the LCD there is a ST75256 controller. Its input pins includes Vdd, Vss and several I/O ports.

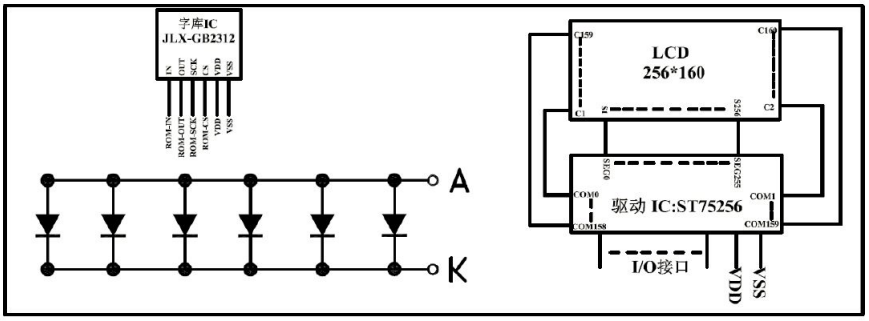

Figure 8: The JLX256160 LCD screen.

In this project, we use Inter-Integrated Circuit (I2C) to control the LCD screen. The following table shows the function of the LCD's pins at I2C mode.

Pin Number	Symbol	Name	Function
1	ROM-IN		
2	ROM-OUT		
3	ROM-SCK		
4	ROM-CS		
5	LEDA	Backlight	3.3V
6	VSS	Ground	0V
7	VDD	Power for circuit	5V or 3.3V
8	RS	Register Select	I2C port, conect to Vdd
9	RES	Reset	Reset at Low, high at work
10	CS	Chip Select	This pin to Vdd for I2C
11	D7	I/O	Slave address, connect to VSS
12	D6	I/O	Slave address, connect to VSS
13	D5	I/O	Not used, connect to VDD
14	D4	I/O	Not used, connect to VDD
15-17	D3-D1(SDA)	I/O	Serial input. (D1, D2, D3 together)
18	D0(SCK)	I/O	Serial Clock
19	RD(E)	Enable	Not used. Connect to VDD
20	WR	Read/Write	Not used. Connect to VDD

Table 1: Function of pins for I2C

2.3.2 Arduino

Instead of using PIC32 to control LCD directly, we use an Arduino board to drive the LCD, and let PIC32 to control the Arduino board by the UART module. The Arduino board we use is the Arduino Mega 2560. It has 8M memory which is power enough to drive our LCD screen.

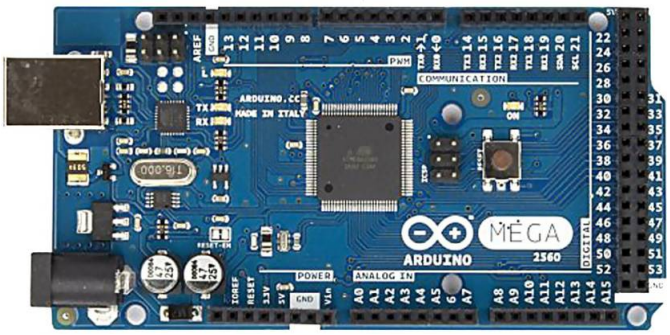

Figure 9: The Arduino Mega board.

2.4 LED

LED modules are used to indicate the hit point of each player. At beginning, each player has 3 hit points, which means each player has 3 LEDs on. If hit points reduces, LEDs will be turned of correspondingly. LEDs are controlled by PORT E of PIC32.

3 Functional Level Design

3.1 Overall Structure

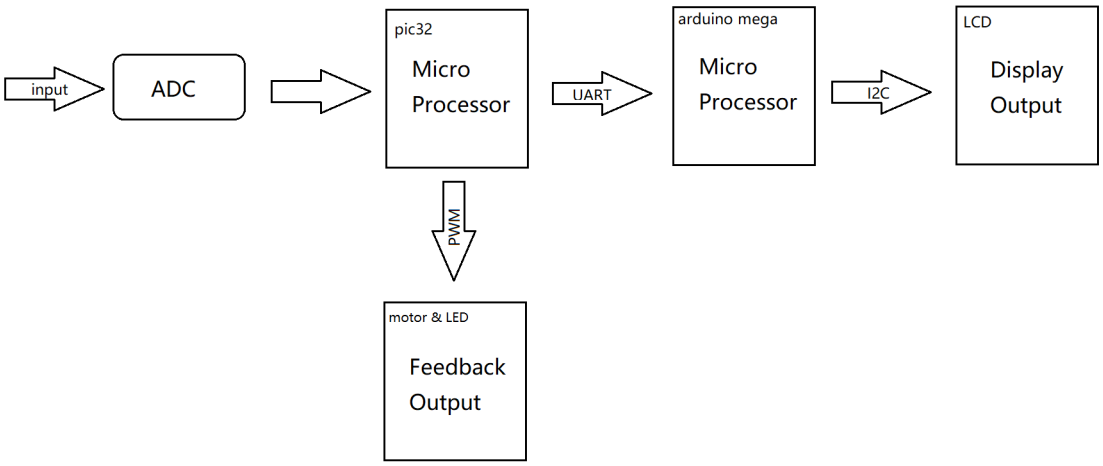

Figure 10: Overall Structure Diangram

3.2 ADC

Analog digital converter is used to convert the analog input from joysticks into digital so that we can process the digital number. Every joystick has two analog outputs, and we have four joysticks, so in total is eight analog inputs. We used "Scan Module" to read in the 8 inputs.

3.3 Serial Communication

Serial communicate is used in the communication between Arduino board and LCD screen, as well as Arduino board and PIC32. In specfic, we use I2C for the former, and UART for the latter.

3.3.1 Communication between PIC32 and Arduino

We use UART module of the PIC32 board to send message. Each time it send a serial of hex numbers to the Arduino, and according to these numbers, Arduino will send message to LCD to display the corresponding contents. The Specific meaning for these numbers will be detailed described in section 4.

3.3.2 Communication between Arduino and LCD

I2C is used in communication between Arduino board and LCD. In detail, two wires, SDA and SCK are used to transmit message. We use Arduino library "u8g2" to help us to drive the LCD.

3.4 PWM

PWM is like an analog output of the pic32 board by changing the duty cycle. In this part, PWM is used as an control signal to the L298N driving motor to drive the motor in the joystick. Also, it is used to adjust the rotation speed of the motor, so that when tank is being destroyed or attacking other tanks, players would get different vibration experience.

4 Game Design

4.1 Brief Introduction

The game will realize the game Battle of Tank. The game starts as reset button is pressed. Two players will use two joysticks per player to control their tank, for movement direction and shoot direction respectively. The shooting is automatically. Every tank has three Health Points. Game ends as one player loses all its HP.

4.2 Rules

- The game is consist of tank1, tank2, bullet, blocks.
- All components moves every turn. The interval between turns is 0.5 second.
- Each turn, tank1 moves first, then tank2 moves, tank1 shoots, tank2 shoots, and finally bullets act.
- Each turn, tanks can move to adjacent grid vertically or horizontally or keep static. Tanks can shoot vertically or horizontally. Bullet would move forward two girds in the shooting direction.
- Each tank has 3 HP as game starts. Each time a tank is shot, it will lose 1 HP. Each time a tank is struck by another tank, it would lose 3 HP, and striking tank would lose 1 HP.
- Damage is balanced at the end of each turn. Once a tank lose all HP, it loses the game. If both tank have 0 HP, then the game draw.

4.3 Utilities

4.3.1 Map

The LCD screen we use has 256 by 160 pixels. To simplify our game design, we devided the whole map into several "units". Each "unit" is a 20 by 20 square area. One of the two tanks (each can have four directions: right, left, up, down), blocks, or a bullet are draw in a unit. For a map, it is divided into 12*7 units.

In other word, an area of 240*140 on the 256*160 LCD is used to display our game.

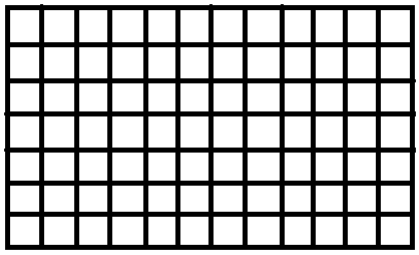

Figure 11: The screen is divided into 12*7 square units.

In each 20*20 unit, we draw a tank (with different directions), a block or a bullet. The following figure shows the detail of player1' s tank with direction of up.

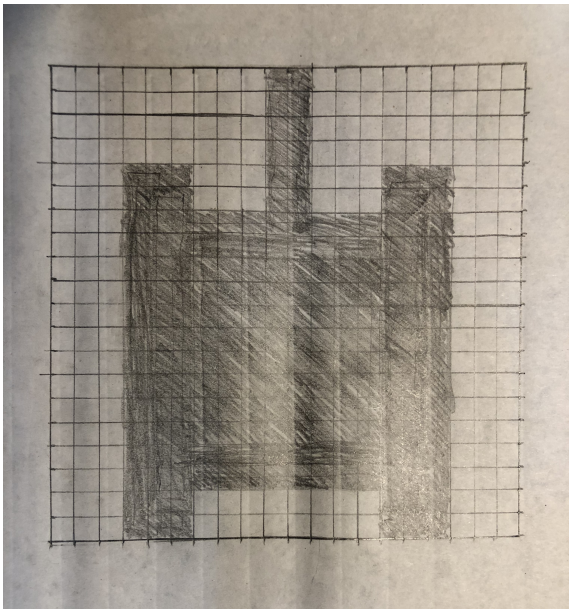

Figure 12: Draw a tank in a 20*20 pixels area.

The 12*7 map is described by an array with a size of 84. If i, m, n are integers, and i = 12 * m + n, so the i-th element in the array, map[i], means the m-th colounm, n-th row element of the 12*7 map.

4.3.2 Bullet

The bullet map keeps the status of each bullet.

```
extern int bullet[MAP_RANGE_X][MAP_RANGE_Y];
static int bullet_old[MAP_RANGE_X][MAP_RANGE_Y];
void InitBulletStat();
void UpdateBulletStat();
enum BULLET_STAT{
    BULLET_NONE, BULLET_UP, BULLET_DOWN,
    BULLET_LEFT, BULLET_RIGHT
};
```

4.3.3 Tank

The tank utility realizes the function of tank movement and shoot, as shown as following structure, the tank can UP, DOWN, LEFT, RIGHT, or STATIC, can shoot UP, DOWN, LEFT, RIGHT.

```
typedef struct{
    int x;
    int y;
    int stat;
    int move_dir;
    int shoot_dir;
    int HP;
} tank;
enum MOVE_DIR{
    MOVE_UP,
    MOVE_DOWN,
    MOVE_LEFT,
    MOVE_RIGHT,
    MOVE_STATIC
};
enum SHOOT_DIR{
    SHOOT_UP,
    SHOOT_DOWN,
        SHOOT_LEFT,
    SHOOT_RIGHT
};
```

4.3.4 Game

```
enum GAME_STAT
{
    RUNNING,
    TANK1_WIN,
    TANK2_WIN,
    DRAW
};
void GameStart();
int CheckGameStat();
void GameRound();
void SendGameStat(int stat);
```

5 Schematics Design

5.1 pic32mx795m512l

Element	Function	Port	Expasion Port
JoyStick1	AN0	RB0	J11 34
	AN1	RB1	J11 33
JoyStick2	AN2	RB3	J11 32
	AN3	RB4	J11 31
JoyStick3	AN8	RB8	J10 33
	AN9	RB9	J10 34
JoyStick4	AN14	RB14	J10 59
	AN15	RB15	J10 60
motor1	OC1	RD0	J11 19
motor2	OC2	RD1	J11 20
motor3	OC3	RD2	J11 17
motor4	OC4	RD3	J11 18
Arduino Mega	U1TX	RF3	J11 43
	U1RX	RF2	J11 41

Table 2: Schematics of pic32mx795m512l

5.2 Arduino Mega

In our project, the Arduino Mega board serves as the driver of the LCD screen. For the Arduino board, it receive message from UART module of the LCD. In our project, we set pin 10 and 11 as RX and TX.

On the other hand, it use I2C to send message to LCD. We use SDA and SCK of the Mega board as the output for I2C, and connect "3.3V" and "GND" to LCD's VDD and VSS. Other pins on LCD that needs to connect to VDD and VSS are also connected to these two pins.

Element	Port
TX	11
RX	10
SDA	SDA
SCK	SCK

Table 3: Schematics of Arduino Mega

5.3 LCD

In Section 2.3.1 we introduce the basic structure of the JLX256160 LCD screen, as well as its pins for I2C mode. Here, the VDD is provided by Arduion's "3.3V", and VSS is provided by "GND".

Arduino	LCD
SDA	D0
SCK	D1-D3
VDD	LEDA
	VDD
	RS
	RES
	D4
	D5
	RD(E)
	WR
VSS	VSS
	CS
	D6
	D7

Table 4: Conection between Arduino and LCD

6 Result

After we assemble all the components, we have our project finished.

Figure 13: The outlook for our project

Figure 14: The overall circuit.

7 Discussion

In this project we have meet tons of difficulties. Several of them are worthy to be mentioned in this section.

7.1 LCD display

In this project we use a 256*160 LCD, which is far larger and more complex than what we used in lab. We tried to use I2C to let it communicate with PIC32 directly.We tried this for two weeks but at last failed. So we turned to use an Arduino board as the driver of the LCD. PIC32 send message and command to Arduino, and Arduino control the LCD to display. Actually there are a great many libraries with powerful function on github.com. We choose the library u8g2, which greatly release our workload in dealing with the LCD display.

8 Reference

1. https://www.mouser.com/catalog/specsheets/ArduinoBoardMega2560.pdf

9 Appendix

1. https://github.com/onethree-13/ve373proj
2. https://www.bilibili.com/video/av28982794/