Tiberiu-Alex-Irinel ȘOȘEA (78399) - Planes

Autorul poate fi contactat la adresa: **Login pentru adresa**

Introducere

Descriere generală

Joc avioane multiplayer, jocul se va juca pe 2 matrice led 8 x 8.

Scurta Descriere

Dupa cum stiti, exista o etapa de startup, unde cei doi jucatori isi vor pregati avioanele pe harta, urmata de o etapa turn-based in care are loc lupta efectiva. Jucatorii se pot plimba pe harta prin intermediul a doua butoane. Cand un jucator ataca, vede unde a lovit iar daca a atins un avion, partea lovita va licari. De asemenea, cand este randul jucatorului advers, iti poti vedea propriile avioane respectiv partile lovite din ele.

Motivatie

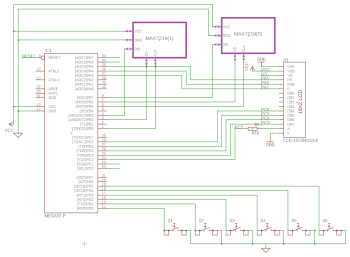
Joc micut si portabil, in lipsa unei ocupatii poate fi conectat la o baterie externa si jocul poate incepe.

Hardware Design

Piesele pe care le-am folosit au fost:

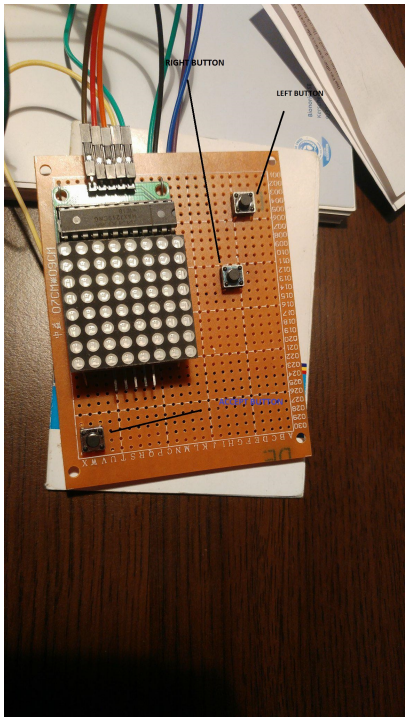
Număr piese	Nume piesă	Observații
1	Placă de bază cu ATMega324A-PU	Alimentare la 5V
2	Matrice led-uri 8 x 8 Max7219	
6	Butoane	
12	Distantieri	Evitarea miscarii pieselor
2	Placuta test	
3	Placuta sustinere	Baza pentru placuta prinicipala si placutele adversarilor
1	Modul LCD 1602 cu Backlight Galben-Verde de 5V	Announcer

Schema Electrica:



Matricea Max7219:

Placuta unui jucator fara sustinere:



Cele 2 placute conectate fara sustinere:

Intreg proiectul fara sustinere:

In timp ce un jucator alege unde sa atace, celalalt jucator isi vede avionul/avioanele:

Video in care se poate vedea lovirea unui avion:

https://drive.google.com/open?id=188_1Vjs_83NRT-Asc0V0f75AZ23Ih6Dl

Video in care se poate observa castigarea jocului:

https://drive.google.com/open?id=1fd-1rD6H6iXZ4o0x4xnekfTMdofTqG3W

Poze cu aspectul proiectului terminat:

Software Design

Mediu de dezvoltare: AtmelStudio

Librarii si surse 3rd-party: -

Structura proiect

Proiectul este organizat în jurul modulelor hardware, fiecare dintre cele 3 module principale. La fiecare modul hardware nou, testam functionalitatea inainte de a monta urmatorul modul.

LCD API este preluat din laborator. Am mai adus mici functionalitati : lumina fundal, text pe cele 2 linii.

lcd.h

```
#ifndef LCD_H_
#define LCD_H_

#include <avr/io.h>
#include <stdint.h>

/*************************************************************************
*******\
 * Define-uri pentru parametrizarea bibliotecii LCD
*
\*************************************************************************
*******/

// Portul pe care conectam pinii de date ai LCD-ului
#define LcdDATA_DDR             DDRC
#define LcdDATA_PORT            PORTC
#define LcdDATA_PIN             PINC

// Pinii de date ai LCD-ului
```

```
#define LcdD4                   PC6
#define LcdD5                   PC5
#define LcdD6                   PC4
#define LcdD7                   PC3

// Portul pe care conectam pinii de control ai LCD-ului
#define LcdCTRL_DDR             DDRA
#define LcdCTRL_PORT            PORTA
#define LcdCTRL_PIN             PINA

// Pinii de control ai LCD-ului
#define LcdRS                   PA5
#define LcdRW                   PA6
#define LcdE                    PA7

/*************************************************************************
*******\
 * Comenzi utile
*
\*************************************************************************
*******/

#define LCD_INSTR_4wire         0x28    // 4 biti de date, 2 linii,
font 8x5
#define LCD_INSTR_displayConfig 0x0f    // display pornit, cursor
afisat, blink activat
#define LCD_INSTR_incNoShift    0x06    // auto-increment, fara
shiftare
#define LCD_INSTR_clearDisplay  0x01    // sterge display
#define LCD_INSTR_returnHome    0x02    // reseteaza cursorul si
shiftarea
#define LCD_INSTR_nextLine      0xC0    // adresa celei de-a doua linii
#define LCD_INSTR_DDRAM         0x80    // prima adresa din DDRAM
#define LCD_INSTR_CGRAM         0x40    // prima adresa din CGRAM

/*************************************************************************
*******\
 * Macro-uri utile
*
\*************************************************************************
*******/

// Mask pentru bitii de date
#define LCD_DATA_MASK
(_BV(LcdD4)|_BV(LcdD5)|_BV(LcdD6)|_BV(LcdD7))

// Mask pentru bitii de control
#define LCD_CTRL_MASK           (_BV(LcdRS)|_BV(LcdRW)|_BV(LcdE))

// Controleaza pinul RS
#define LCD_RS_HIGH()           do { LcdCTRL_PORT |=  _BV(LcdRS); }
```

```
while(0)
#define LCD_RS_LOW()            do { LcdCTRL_PORT &= ~_BV(LcdRS); }
while(0)

// Controleaza pinul RW
#define LCD_RW_HIGH()           do { LcdCTRL_PORT |=  _BV(LcdRW); }
while(0)
#define LCD_RW_LOW()            do { LcdCTRL_PORT &= ~_BV(LcdRW); }
while(0)

// Controleaza pinul E
#define LCD_ENABLE()            do { LcdCTRL_PORT |=  _BV(LcdE); }
while(0)
#define LCD_DISABLE()           do { LcdCTRL_PORT &= ~_BV(LcdE); }
while(0)

// Delay-ul minim necesar pentru operatiile cu semnalul E -> 250ns
// Acopera toate celelalte delay-uri si este jumatate din Enable cycle
time (T_cycE)
#define LCD_DELAY()             __builtin_avr_delay_cycles(250 /
(1000000000/F_CPU))

/*************************************************************************
*******\
 * API LCD
*
\*************************************************************************
*******/

// Initializare LCD considerand o interfatare cu 4 pini de date.
// Trebuie apelata inainte de a face orice operatie cu LCD-ul.
void LCD_init(void);

// Executa secventa de citire a unui octet de date de la LCD.
// Pentru interfatarea cu 4 pini de date sunt necesare 2 transferuri.
uint8_t LCD_read(void);

// Citeste starea LCD-ului (contine busy flag).
uint8_t LCD_readStatus(void);

// Citeste un octet din ultima memorie folosita (DDRAM sau CGRAM).
uint8_t LCD_readData(void);

// Returneaza starea LCD-ului: 1 - busy, 0 - available
uint8_t LCD_isBusy(void);

// Asteapta pana cand LCD-ul devine disponibil pentru o noua comanda.
void LCD_waitNotBusy(void);

// Executa secventa de trimitere a unui octet de date catre LCD.
// Pentru interfatarea cu 4 pini de date sunt necesare 2 transferuri.
```

```
void LCD_write(uint8_t data);

// Trimite o instructiune de control catre LCD.
void LCD_writeInstr(uint8_t instr);

// Trimite o instructiune de scriere date catre LCD.
void LCD_writeData(uint8_t data);

void LCD_putChar(char c);                           // Afiseaza
caracterul pe LCD la adresa curenta.
void LCD_putCharAt(uint8_t addr, char c);           // Afiseaza
caracterul pe LCD la adresa primita.

void LCD_print(const char* msg);                    // Afiseaza
string-ul pe LCD incepand de la adresa curenta.
void LCD_printAt(uint8_t addr, const char* msg);    // Afiseaza
string-ul pe LCD incepand de la adresa primita.

void LCD_clear_top_line();                          // Sterge linia de
sus a LCD-ului
void LCD_clear_bottom_line();                       // Sterge linia de
jos a LCD-ului

#endif // LCD_H_
```

Datele (aprinderea matricii) trebuie transmise serial microcontrolerului MAX2179, Dupa cum am vazut, MAX2179 are urmatorii pini:

-VCC

-DND

-DIN

-CS

-CLK

Astfel, am setat urmatorii pini :

MAX_settings

```
#define CLK_HIGH1()  PORTB |= (1<<PB1)
#define CLK_LOW1()   PORTB &= ~(1<<PB1)
#define CS_HIGH1()   PORTB |= (1<<PB2)
#define CS_LOW1()    PORTB &= ~(1<<PB2)
#define DATA_HIGH1() PORTB |= (1<<PB3)
#define DATA_LOW1()  PORTB &= ~(1<<PB3)
#define INIT_PORT1() DDRB |= (1<<PB3) | (1<<PB1) | (1<<PB2)
```

```
#define CLK_HIGH2()  PORTB |= (1<<PB5)
#define CLK_LOW2()   PORTB &= ~(1<<PB5)
#define CS_HIGH2()   PORTB |= (1<<PB6)
#define CS_LOW2()    PORTB &= ~(1<<PB6)
#define DATA_HIGH2() PORTB |= (1<<PB7)
#define DATA_LOW2()  PORTB &= ~(1<<PB7)
#define INIT_PORT2() DDRB |= (1<<PB5) | (1<<PB6) | (1<<PB7)
```

Pinii folositi pentru butoane sunt urmatorii:

Button_Settings

```
        DDRD &= ~(1 << PD0);
        DDRD &= ~(1 << PD1);
        DDRD &= ~(1 << PD2);
        DDRB &= ~(1 << PB0);
        DDRD &= ~(1 << PD4);
        DDRD &= ~(1 << PD5);
        DDRD &= ~(1 << PD6);
        DDRD &= ~(1 << PD7);

        PORTD |= (1 << PD0);
        PORTD |= (1 << PD1);
        PORTD |= (1 << PD2);
        PORTB |= (1 << PB0);
        PORTD |= (1 << PD5);
        PORTD |= (1 << PD6);
        PORTD |= (1 << PD7);
```

Am in continuare functiile care care interfateaza ATMEGA meu cu MAX2179:

Interfacing

```
void spi_send(uint8_t data)
{
        uint8_t i;

        for (i = 0; i < 8; i++, data <<= 1)
        {
                CLK_LOW();
                if (data & 0x80)
                DATA_HIGH();
                else
                DATA_LOW();
                CLK_HIGH();
        }

}
```

```
void max7219_writec(uint8_t high_byte, uint8_t low_byte)
{
        CS_LOW();
        spi_send(high_byte);
        spi_send(low_byte);
        CS_HIGH();
}


void max7219_clear2(void)
{
        uint8_t i;
        for (i = 0; i < 8; i++)
        {
                max7219_writec(i+1, 0);
        }
}




void max7219_init(void)
{
        INIT_PORT();
        // Decode mode: none
        max7219_writec(0x09, 0);
        // Intensity: 3 (0-15)
        max7219_writec(0x0A, 1);
        // Scan limit: All "digits" (rows) on
        max7219_writec(0x0B, 7);
        // Shutdown register: Display on
        max7219_writec(0x0C, 1);
        // Display test: off
        max7219_writec(0x0F, 0);
        max7219_clear();
}

uint8_t display[8];

void update_display(void)
{
        uint8_t i;

        for (i = 0; i < 8; i++)
        {
                max7219_writec(i+1, display2[i]);
        }
}
```

```
void image(const __flash uint8_t im[8])
{
    uint8_t i;

    for (i = 0; i < 8; i++)
        display[i] = im[i];
}


void set_pixel(uint8_t r, uint8_t c, uint8_t value)
{
        switch (value)
        {
                case 0: // Clear bit
                display[r] &= (uint8_t) ~(0x80 >> c);
                break;
                case 1: // Set bit
                display[r] |= (0x80 >> c);
                break;
                default: // XOR bit
                display[r] ^= (0x80 >> c);
                break;
        }
}
```

Odata ce interfatarea a fost realizata, mai este doar o problema de programare. Astfel, intreg program-ul este impartit in 2 mari segmente→ partea de plasare a avioanelor, respectiv lupta propriu-zisa. Primul segment decurge astfel incat cei 2 oponenti isi pot alege locurile avioanelor in acelasi timp (pentru testare am limitat plasarea la un singur avion - astfel timpul jocului este mai mic, si este mai usoara urmarirea flow-ului programului. Momentan lcd-ul doar anunta ca trebuie “pregatite avioanele”. Odata ce aceasta etapa se incheie, jocul devine turn based. Cursorul licare si pozitia sa poate fi modificata prin intermediul butoanelor. Butonul de accept loveste o anumita casuta, iar turn-ul se schimba. Acum, jucatorul curent isi vede avioanele si jucatorul advers poate muta.

Rezultate Obţinute

Un joc compact si foarte practic care sa “omoare timpul”

Concluzii

A fost foarte interesanta aceasta conexiune hard-soft pe care la nicio alta materie nu am mai exerimentat-o. Partea tricky a fost realizarea hardware-ului

Download

Arhiva cu sursele, momentan am cod duplicat si imbarligat putin, insa codul este functional: planes.zip

Jurnal

Week 1: schema bloc

Week 2: schema electrica + terminare placa baza

Week 3: achizitionare toate piesele fara disnatieri si placi de suport

Week 4: terminarea software-ului

Week 5: montarea suportului pentru placa si cei 2 jucatori

Bibliografie/Resurse

Tutoriale arduino pt max:

8x8-led-matrix-max7219-tutorial-scrolling-text-android-control-via-bluetooth

how-to-control-max7219-led-matrix

Niste cod pt atmega max:

2352797

- Documentația în format PDF