UNIVERSIDADE FEDERAL DO CEARÁ – CAMPUS SOBRAL

ENGENHARIA DA COMPUTAÇÃO

MICROPROCESSADORES

PROFESSORES: MARCELO MARQUES SIMÕES DE SOUZA

TRABALHO 2

PIC18F4550 E DISPLAY LCD

Aluno: Emanuel Dêvid Paulino Felix **Matrícula: 469870**

Sobral – CE

2022.2

Sumário

1. INTRODUÇÃO

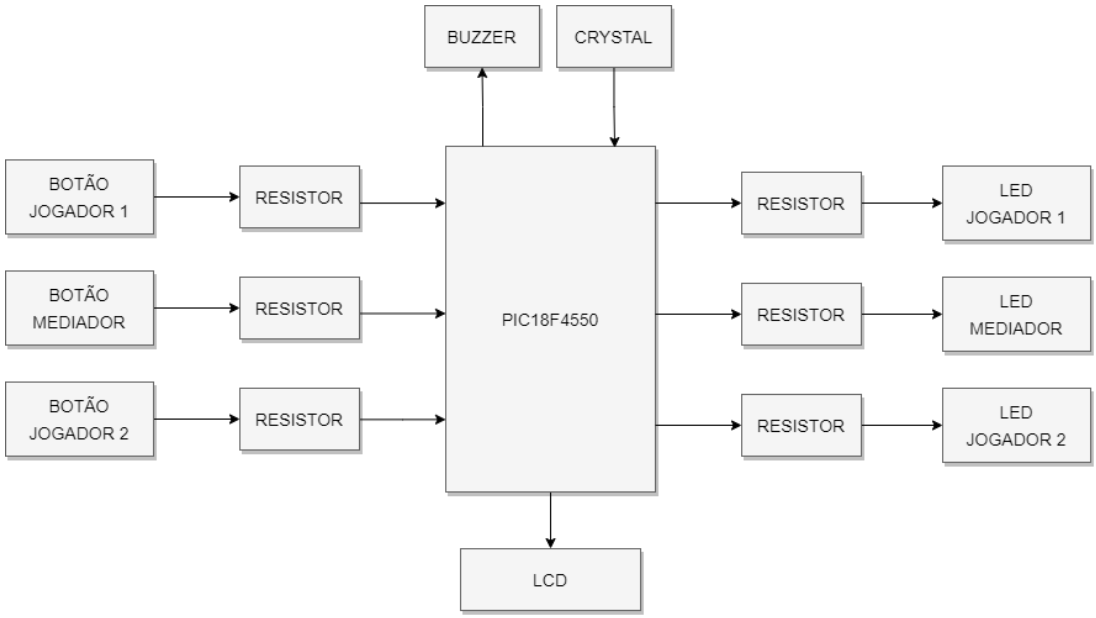
O presente relatório apresenta o segundo trabalho da disciplina de Microprocessadores, onde o objetivo é desenvolver um “jogo de reflexos” onde dois jogadores e um mediador participam. As tecnologias utilizadas foram o Proteus (para o projeto do hardware) e o MPLAB (para o desenvolvimento da lógica e do código).

Resumidamente, o jogo funciona da seguinte forma: existem três botões, um para o mediador e uma para cada um dos dois jogadores. Além disso, existem três LEDS, um buzzer e um display LCD. Quando o mediador pressiona o botão o LED correspondente ascende e o buzzer apita. Então, os jogadores devem apertar seus botões e ascenderá o LED referente ao jogador que apertou seu botão primeiro. Por fim, o display LCD mostrará o tempo de reação de cada jogador.

2. HARDWARE

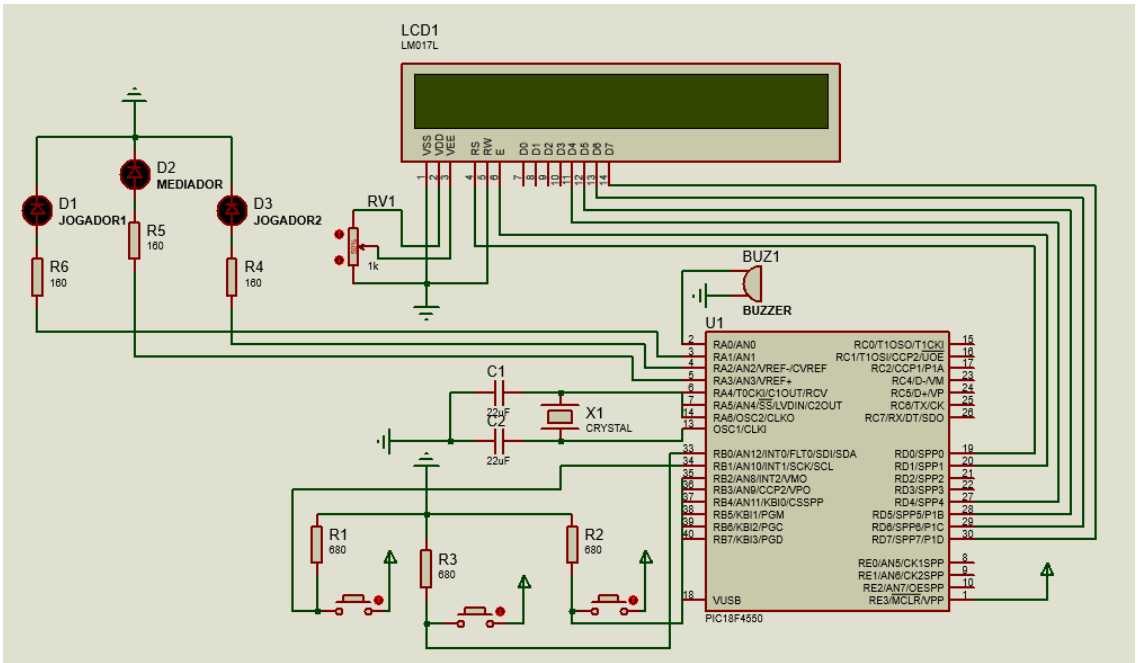
A seguir está mostrado o hardware desenvolvido tanto por meio de um diagrama de blocos, quanto em esquemático (feito no Proteus), respectivamente.

Figura 2.1 – Diagrama de blocos



Fonte: Autor

Figura 2.2 – Esquemático



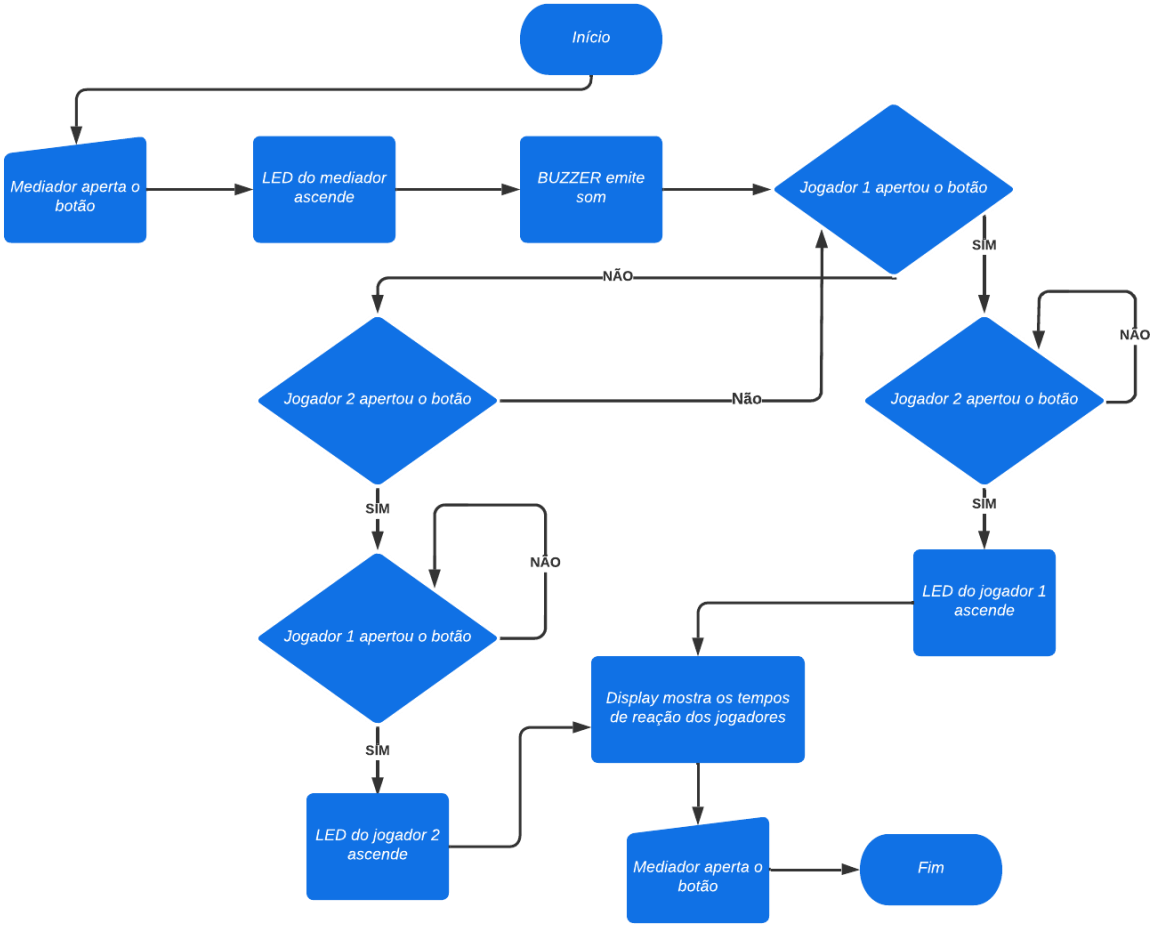
Fonte: Autor

Como pode ser visto, o hardware basicamente é constituído por um micro controlador (PIC18F4550), três botões, três LEDs, alguns GNDs, uma fonte, um cristal, quatro resistores, um buzzer e um display LCD. O funcionamento do hardware já foi anteriormente explicado de maneira breve, cabendo acrescentar apenas que o micro será responsável por todo o controle antes descrito.

3. SOFTWARE

Aqui será mostrado o software desenvolvido. Isso será feito, em primeiro lugar, por meio da Figura 3.1 que apresenta o fluxograma do software. Após isso, será mostrado todo o código desenvolvido no MPLAB.

Figura 3.1 – Fluxograma



Fonte: Autor

Figura 3.2 – Código lcd.h

```h
#ifndef XC_HEADER_TEMPLATE_H
#define XC_HEADER_TEMPLATE_H
#include <xc.h>

#define _XTAL_FREQ 4000000

#define lcd_rs     PORTDbits.RD0 // pino rs (register select)do LCD
#define lcd_enable PORTDbits.RD1 // pino enable do LCD
#define lcd_db4    PORTDbits.RD4 // pino de dados d4 do LCD
#define lcd_db5    PORTDbits.RD5 // pino de dados d5 do LCD
#define lcd_db6    PORTDbits.RD6 // pino de dados d6 do LCD
#define lcd_db7    PORTDbits.RD7 // pino de dados d7 do LCD

/*****************************************************************************/
/* Rotinas para o LCD */
/*****************************************************************************/
 void envia_nibble_lcd(char dado);
 void envia_byte_lcd(char endereco,char dado);
 inline void escreve_lcd(char c);
 void limpa_lcd(void);
 void inicializa_lcd(void);
 void caracter_inicio(char linha,char coluna);

#endif  /* XC_HEADER_TEMPLATE_H */
```

Fonte: Autor

Figura 3.3 – Código lcd.c

```c
#include "lcd.h"

// Escreve no LCD os 4 bits menos significativos de dado
void envia_nibble_lcd(char dado)
{
  lcd_enable = 1;
  lcd_db4 = dado & 0x01;
  lcd_db5 = (dado >> 1) & 0x01;
  lcd_db6 = (dado >> 2) & 0x01;
  lcd_db7 = (dado >> 3) & 0x01;
  //Gera um pulso de enable

  __delay_us(50); // Recomendado para estabilizar o LCD
  lcd_enable = 0;
  __delay_us(50);
}

// Esta rotina envia um dado ou um comando para o LCD
// endereco = 0 -> a variável dado será enviada como uma instrução
// endereco = 1 -> a variável dado será enviada como um caractere

void envia_byte_lcd(char endereco,char dado)
{
 lcd_rs = endereco; // Seta o bit RS para instrução ou caractere
 __delay_us(5);   // Aguarda 5 us para o LCD estabilizar
 envia_nibble_lcd(dado>>4); // Envia ao LCD os MSBs de dado
 envia_nibble_lcd(dado & 0x0f); // Envia os LSBs de dado
}
```

Fonte: Autor

Figura 3.4 – Código lcd.c

```
// Esta rotina foi implementada como uma forma conveniente
// de se escrever um caractere no display

inline void escreve_lcd(char c){envia_byte_lcd(1,c);};

// Função que envia instrução para limpar a tela do LCD
void limpa_lcd(void)
{
  envia_byte_lcd(0,0x01); // 0x01 -> instrução para limpar LCD
  __delay_ms(2);  // Aguarda 2ms para estabilizar o LCD
}

// Função de inicialização do LCD
void inicializa_lcd(void)
{
 lcd_enable = 0;
 lcd_db4 = 0;  // Garante que os pino DB4-DB7 estão em 0 (low)
 lcd_db5 = 0;
 lcd_db6 = 0;
 lcd_db7 = 0;

 __delay_ms(50); // Aguarda 5ms para estabilizar o LCD
 envia_nibble_lcd(0x03); // Envia comando de inicialização
 __delay_ms(4); // Aguarda 5ms para estabilizar o LCD
 envia_nibble_lcd(0x03); // Envia comando de inicialização
 __delay_us(100); // Aguarda 5ms para estabilizar o LCD
 envia_nibble_lcd(0x03); // Envia comando de inicialização

 envia_nibble_lcd(0x02); // Comando que configura o LCD para 4 bits

```

Fonte: Autor

Figura 3.5 – Código lcd.c

```
 envia_byte_lcd(0,0x01); // instrução para limpar LCD
 __delay_ms(2);  // Aguarda 2ms para estabilizar o LCD

 envia_byte_lcd(0,0x28); // Comando que configura o LCD para 4 bits, 2 linhas
 // e fonte 5X7.
 envia_byte_lcd(0,0x0c); // Display ligado, sem cursor

 envia_byte_lcd(0,0x06); // Posiciona o cursor à direita
}

// Esta função define a posição da tela do LCD (linha e coluna)
// a partir da qual serão realizadas futuras escritas de caracteres.

 void caracter_inicio(char linha,char coluna)
{
 char posicao=0;
 if(linha == 1)
{
 posicao=0x80; // Endereço inicial da linha 1 (0x80)
}
 if(linha == 2)
{
 posicao=0xc0; // Endereço inicial da linha 2 (0xc0)
}

posicao=posicao+coluna; //soma o número da coluna ao endereço inicial

envia_byte_lcd(0,posicao); // Envia commando
}

```

Fonte: Autor

Figura 3.6 – Código main

```
#include <xc.h>
#include <stdio.h>
#include <stdlib.h>
#include "lcd.h"
#define _XTAL_FREQ 4000000
#pragma config FOSC = XT_XT // Oscillator Selection bits (XT oscillator (XT))
#pragma config WDT = OFF    // Watchdog Timer Enable bit (WDT disabled (control is placed on the SWDTEN bit))
#pragma config PBADEN = OFF // PORTB A/D Enable bit (PORTB<4:0> pins are configured as digital I/O on Reset)
#pragma config MCLRE = ON   // MCLR Pin Enable bit (MCLR pin enabled; RE3 input pin disabled)
#pragma config LVP = OFF    // Single-Supply ICSP Enable bit (Single-Supply ICSP disabled)
unsigned int mediador = 0;  // variável para verificar se o botão do mediador foi pressionado
unsigned int button1 = 0;   // variável para verificar se o botão do jogador 1 foi pressionado
unsigned int button2 = 0;   // variável para verificar se o botão do jogador 2 foi pressionado

// função que escreve caracteres no lcd com a função "printf()"
void putch(char data)
{
    escreve_lcd(data);
}

int temp_player1 = 0; // variÃ¡vel que guarda o tempo do jogador1
int temp_player2 = 0; // variÃ¡vel que guarda o tempo do jogador2

void main(void)
{
    TRISBbits.TRISB0 = 1; // definindo RD0 como entrada do button mediador
    TRISBbits.TRISB1 = 1; // definindo RD1 como entrada do button jogador1
    TRISBbits.TRISB2 = 1; // definindo RD2 como entrada do button jogador2

    TRISAbits.RA3 = 0;    // definindo RA3 omo saida do led do mediador
    TRISAbits.RA0 = 0;    // definindo RA0 como saida do BUZZER
    TRISAbits.RA1 = 0;    // definindo RA1 como saida do led do jogador1
    TRISAbits.RA2 = 0;    // definindo RA2 como saida do led do jogador2
    TRISBbits.TRISB3 = 0; // definindo RD3 como saida ->buzzer

    PORTD = 0; // pinos a serem utilizados para o LCD
    TRISD = 0x00;
```

Fonte: Autor

Figura 3.7 – Código main

```
    ADCON1 = 0x0F;

    T2CONbits.T2CKPS1 = 0; // Pre scaler 1:4
    T2CONbits.T2CKPS0 = 1; // Pre scaler 1:4
    PR2 = 250;             // 250 Para estourar a cada 1 ms

    // T2OUTPS3 a T2OUTPS0 controlam quantos estouros de timer2 precisam pra levar TMR2IF pra 1
    T2CONbits.T2OUTPS3 = 0;
    T2CONbits.T2OUTPS2 = 0;
    T2CONbits.T2OUTPS1 = 0;
    T2CONbits.T2OUTPS0 = 1;
    INT0IE = 1; // habilita interrupï¿½ï¿½o int0

    PEIE = 1; // habilita interrupicoes dos perifericos
    GIE = 1;  // habilita interrupicoes globalmente

    TMR2 = 0;

    T2CONbits.TMR2ON = 1; // iniciaa a contagem do timer
    int print_tela = 0;   // variavel que controla a impressao (print) na tela
```

Fonte: Autor

Figura 3.8 – Código main

```
66      while (1)
67      {
68          while (mediador == 1)
69          {
                if (TMR2IF){
                    TMR2IF = 0;
72                  if (button1 == 0)
73                      temp_player1++;
74                  if (button2 == 0)
75                      temp_player2++;
76              }
77
78              if ((button1 == 1) && (button2 == 1)){
79                  mediador = 0;
80
81                  if(temp_player1 < temp_player2)
                        PORTAbits.RA1 = 1;
83
84                  else if(temp_player1 > temp_player2)
                        PORTAbits.RA2 = 1;
86
87                  print_tela = 1;
88              }
89          }
```

Fonte: Autor

Figura 3.9 – Código main

```
91          while (print_tela){
92              inicializa_lcd(); //inicializando o lcd
93              limpa_lcd(); //limpando lcd
94              caracter_inicio(1, 1); // define o ponto de inicio da frase na primeira linha
95              printf("TEMPO JOGADOR 1: = %d ms", temp_player1); //printando na tela
96              __delay_ms(250);        // escreve string no LCD
97              __delay_ms(250);        // escreve string no LCD
98              caracter_inicio(2, 1); // define o ponto de inicio da frase na segunda linha
99              printf("TEMPO JOGADOR 2: = %d ms", temp_player2); //printando na tela
100             __delay_ms(250); // escreve string no LCD
101             __delay_ms(250); // escreve string no LCD
102             print_tela = 0;  // renicia a variavel pra que o print ocorra apenas uma vez
103         }
104     }
105 }
```

Fonte: Autor

Figura 3.10 – Código main

```
108 //interrupção que é chamada quando o botao do mediador é pressionado
109 void __interrupt(high_priority) isr()
110 {
    if (INT0IF){
        if ((button1 == 0) && (button2 == 0)){
            mediador = 1;
            PORTAbits.RA3 = 1; //ascende o LED do mediador
            INT1IE = 1; // habilitando interrupção que trata do botao 1
            INT2IE = 1; // habilitando interrupção que trata do botao 2
117         temp_player1 = 0;
118         temp_player2 = 0;
            PORTAbits.RA0 = 1;
120         __delay_ms(500);
            PORTAbits.RA0 = 0;
            PORTAbits.RA3 = 0;
123     }
124     else if ((button1 == 1) && (button2 == 1)){
125         // condicao para voltar ao estagio inicial
126         mediador = 0;
127         button1 = 0;
128         button2 = 0;
            PORTAbits.RA1 = 0;
            PORTAbits.RA2 = 0;
            PORTAbits.RA3 = 0;
132         limpa_lcd();
133     }
134
        INT0IF = 0; //zerando variavel para que o processo ocorra apenas uma vez quando a interrupção é chamada
136
137     return;
138 }
```

Fonte: Autor

Figura 3.11 – Código main

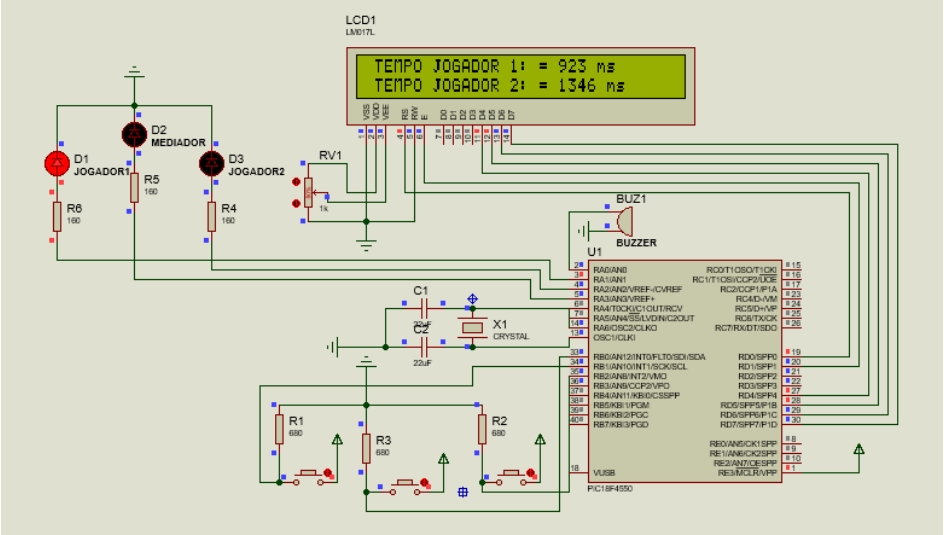
```
140     //interrupção que trata do botao do jogador 1 (RB1/INT1)
        if (INT1IF){
142         button1 = 1; //se o botao for pressionado button1 recebe 1 e sua incrementação é parada na main
            INT1IF = 0; //zerando variavel para que o processo ocorra apenas uma vez quando a interrupção é chamada
144         return;
145     }
146
147     //interrupção que trata do botÃ£o do jogador 2 (RB2/INT2)
        if (INT2IF){
149         button2 = 1; // se o botao for pressionado button2 recebe 1 e sua incrementação é parada na main
            INT2IF = 0; //zerando variÃ¡vel para que o processo ocorra apenas uma vez quando a interrupção é chamada
151         return;
152     }
153     return;
154 }
```

Fonte: Autor

4. SIMULAÇÃO

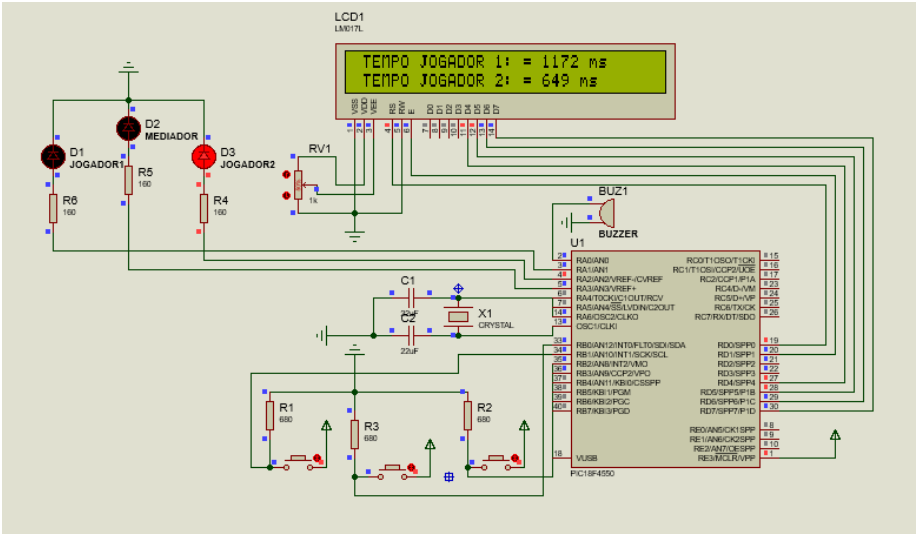
Finalmente, as figuras abaixo mostram alguns resultado obtidos na simulação, sendo um exemplo em que o jogador 1 venceu e outro em que o jogador 2 venceu. Ademais, está disponível, ao fim deste documento, o link de um vídeo em que é possível ver um melhor funcionamento do software.

Figura 3.9 – Vitória do jogador 1



Fonte: Autor

Figura 3.9 – Vitória do jogador 2



Fonte: Autor

LINK DO VÍDEO DE SIMULAÇÃO NO PROTEUS:
https://drive.google.com/file/d/1cuytAKhnDt0NCE7MOypDeLWGvSqzH7HJ/view?usp=share_link