Grupo: Bernardo Prina Righi, Douglas Somensi.

Requisitos do projeto

1. Projeto de interface gráfica para o envio de mensagens e leitura das mensagens recebidas e enviadas.

O projeto desenvolvimento será entregue juntamente com esta documentação.

2. Projeto de interface gráfica para configuração da porta de comunicação.

A configuração das portas seriais foi feita através do programa Socat e Minicom.

Socat é um utilitário baseado em linha de comando que estabelece dois fluxos de bytes bidirecionais e transfere dados entre eles. Já o Minicom é um programa de emulação de terminal e controle de modem baseado em texto para sistemas operacionais semelhantes a Unix.

3. Definição das teclas utilizadas para envio das mensagens e movimentação no histórico de mensagens recebidas e enviadas.

Foram mapeadas as teclas que o usuário poderia utilizar de acordo com padrão UTF-8, conforme função abaixo.

```
char mapeamento_teclado(int key)
{
   char key_map[150];

   for (int i = 0; i < 150; i++)
   {
      key_map[i] = '*';
   }

   key_map[0x02] = '1';
   key_map[0x03] = '2';
   key_map[0x04] = '3';
   key_map[0x05] = '4';
   key_map[0x06] = '5';
   key_map[0x07] = '6';
   key_map[0x08] = '7';
   key_map[0x09] = '8';
   key_map[0x0a] = '9';
   key_map[0x0b] = '0';
   key_map[0x1e] = 'A';
   key_map[0x30] = 'B';
   key_map[0x2e] = 'C';
   key_map[0x20] = 'D';
```

```
    key_map[0x12] = 'E';
    key_map[0x21] = 'F';
    key_map[0x22] = 'G';
    key_map[0x23] = 'H';
    key_map[0x17] = 'I';
    key_map[0x24] = 'J';
    key_map[0x25] = 'K';
    key_map[0x26] = 'L';
    key_map[0x32] = 'M';
    key_map[0x31] = 'N';
    key_map[0x18] = 'O';
    key_map[0x19] = 'P';
    key_map[0x10] = 'Q';
    key_map[0x13] = 'R';
    key_map[0x1f] = 'S';
    key_map[0x14] = 'T';
    key_map[0x16] = 'U';
    key_map[0x2f] = 'V';
    key_map[0x2d] = 'X';
    key_map[0x2c] = 'Z';
    key_map[0x15] = 'Y';
    key_map[0x11] = 'W';
    //barra de espaço
    key_map[0x39] = ' ';
    //enter
    key_map[0x1c] = 0x0A;
    //backspace
    key_map[0x0E] = ' ';

    return key_map[key];
}
```

4. Descrição do funcionamento do vídeo no modo texto colorido com 80 colunas e 25 linhas.

Foi definido um array com o total máximo de colunas possíveis(80) para que ao digitar uma frase, as letras digitadas fossem armazenadas para posteriormente ser mostrada no histórico da tela. No nosso caso, desenhamos uma borda azul para separar a parte da digitação do histórico, tanto nas beiradas da tela como no centro.

```
unsigned short *video = (unsigned short *)0xB8000;

void printc(int x, int y, int fcolor, int bcolor, char c)
{
    video[x + y * 80] = (fcolor << 8) | (bcolor << 12) | c;
}
```

```
void prints(int x, int y, int fcolor, int bcolor, char *str)
{

    int inicio = x + y * 80;

    while (*str)
    {
        video[inicio] = (fcolor << 8) | (bcolor << 12) | *str;
        write_serial(*str);
        inicio++;
        str++;
    }
}

void tela(void)
{
    int pos_x = 0;
    int pos_y = 0;
    int fcolor2 = Blue;
    int bcolor2 = Blue;

    for (pos_x = 0; pos_x < 80; pos_x++)
    {
        printc(pos_x, 0, fcolor2, bcolor2, ' ');
    }
    for (pos_x = 0; pos_x < 80; pos_x++)
    {
        printc(pos_x, 17, fcolor2, bcolor2, ' ');
    }
    for (pos_x = 0; pos_x < 80; pos_x++)
    {
        printc(pos_x, 24, fcolor2, bcolor2, ' ');
    }

    for (pos_y = 0; pos_y < 24; pos_y++)
    {
        printc(0, pos_y, fcolor2, bcolor2, ' ');
    }

    for (pos_y = 0; pos_y < 24; pos_y++)
    {
        printc(79, pos_y, fcolor2, bcolor2, ' ');
    }
}
```

5. Descrição do funcionamento da leitura do teclado e montagem da tabela de valores que indicam as teclas mapeadas.

```
while (1)
   {
      dataehora(19, 2);
      key = inb(0x60);

      if (serial_received()) {
         printc(visualizar_linha, 1, White, Black, read_serial());
      }

      if (mapeamento_teclado(key) != '*' && key < 150)
      {
         if (key_temp != key)
         {
            key_temp = key;

            if (mapeamento_teclado(key) != 0x0A) {

               if (key == 0x0E) {
                  chat_linha--;
                  auxcol--;
                       printc(chat_linha, 22, White, Black, mapeamento_teclado(key)); //imprime
linha do chat
                  vetor_buffer[auxrow][auxcol] = mapeamento_teclado(key);
               }

                 printc(chat_linha, 19, White, Black, mapeamento_teclado(key)); //imprime linha
do chat
               vetor_buffer[auxrow][auxcol] = mapeamento_teclado(key);
               chat_linha++;
               auxcol++;
            }
            else // if (mapeamento_teclado(key) == 0x0A)
            {
               vetor_buffer[auxrow][auxcol] = '\0';
               dataehora(auxrow, 2);
               visualizar_linha = 20;
               prints(visualizar_linha, auxrow, White, Black, vetor_buffer[auxrow]);
               auxcol = 0;
               limpa_chat();
               tela();
               chat_linha = 20;
```

```
                if(auxrow == 16){
                    limpa_tela();
                    tela();
                    prints(visualizar_linha, 1, White, Black, vetor_buffer[auxrow]);
                    dataehora(1, 2);
                    auxrow = 2;
                } else {
                    auxrow++;
                }
            }
        }
    }
}
```

6. Descrição da configuração da porta serial utilizada para comunicação.

Para fazer esse teste do serial será necessário o QEMU (emulador de hardware), do minicom (leitor visual de Serial) e o Socat (Programa virtualizador de portas seriais).

- Geramos duas portas seriais utilizando o Socat, executando o comando socat -d -d pty,raw,echo=0 pty,raw,echo=0
- Alteramos o Makefile disponibilizado pelo professor, editando a parte do run, informando a porta gerada.

```
run: $(PROJECT).elf
$(QEMU) -soundhw pcspk -serial /dev/pts/2 -kernel $<
```

Obs.: Na parte do /dev/pts/2, foi utilizada a primeira porta gerada no output do Socat.

- Após isso, executamos o programa e depois o minicom para ler a segunda porta gerada pelo Socat, através do comando: minicom -D /dev/pts/3 -b 38400 -8.

Obs.: Na parte do "/dev/pts/3", foi utilizada a segunda porta gerada no output do Socat e no "38400" definimos o boud rate, mesmo que usamos no arquivo main.c para leitura e gravação na porta.

O minicom gera duas portas seriais virtuais, por exemplo, /dev/pts/2 e /dev/pts/3. Editando o makefile, utilizamos o qemu para mapear para um dispositivo a serial que no nosso caso seria /dev/pts/2.

Utilizamos o minicom para ler a serial /dev/pts/3 usando o boud rate 38400, ou seja, cada vez que escrevemos na porta /dev/pts/2 podemos ler o valor na /dev/pts/3, pois é gerado um túnel entre essas duas seriais.

7. Descrição da configuração para leitura da data e hora do RTC.

A função outb é uma instrução de montagem com o mesmo nome, usado para falar com dispositivos mapeados por porta.

```
void dataehora(int linha, int coluna)
{
   //Dia
   outb(0x70, 0x07);
   int dia = inb(0x71);
   printc(0 + coluna, linha, White, Black, (dia >> 4) + '0');
   printc(1 + coluna, linha, White, Black, (0x0F & dia) + '0');
   printc(2 + coluna, linha, White, Black, '/');

   //Mes
   outb(0x70, 0x08);
   int mes = inb(0x71);
   printc(3 + coluna, linha, White, Black, (mes >> 4) + '0');
   printc(4 + coluna, linha, White, Black, (0x0F & mes) + '0');
   printc(5 + coluna, linha, White, Black, '/');

   //Ano
   outb(0x70, 0x09);
   int ano = inb(0x71);
   printc(6 + coluna, linha, White, Black, (ano >> 4) + '0');
   printc(7 + coluna, linha, White, Black, (0x0F & ano) + '0');

   //Horas
   outb(0x70, 0x04);
   int hora = inb(0x71);
   printc(8 + coluna, linha, White, Black, '-');
   printc(9 + coluna, linha, White, Black, (hora >> 4) + '0');
   printc(10 + coluna, linha, White, Black, (0x0F & hora) - 3 + '0');
   printc(11 + coluna, linha, White, Black, ':');

   //Minutos
   outb(0x70, 0x02);
   int min = inb(0x71);
   printc(12 + coluna, linha, White, Black, (min >> 4) + '0');
   printc(13 + coluna, linha, White, Black, (0x0F & min) + '0');
   printc(14 + coluna, linha, White, Black, ':');

   //Segundos
   outb(0x70, 0x00);
   int seg = inb(0x71);
   printc(15 + coluna, linha, White, Black, (seg >> 4) + '0');
```

```
    printc(16 + coluna, linha, White, Black, (0x0F & seg) + '0');
    printc(17 + coluna, linha, White, Black, ':');
}
```