U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Connect Four

Relatório do Projeto de LCOM
MIEIC

Turma 7, grupo 2
Adriano Filipe Ribeiro Soares up201904873@fe.up.pt
Vasco Marinho Rodrigues Gomes Alves up201808031@fe.up.pt

Índice

1. Instruções Para o Utilizador

Ao iniciar o programa, é apresentado com quatro opções que deverá selecionar com o rato de acordo com aquilo que pretende fazer: Local, Online, Instructions e Exit.

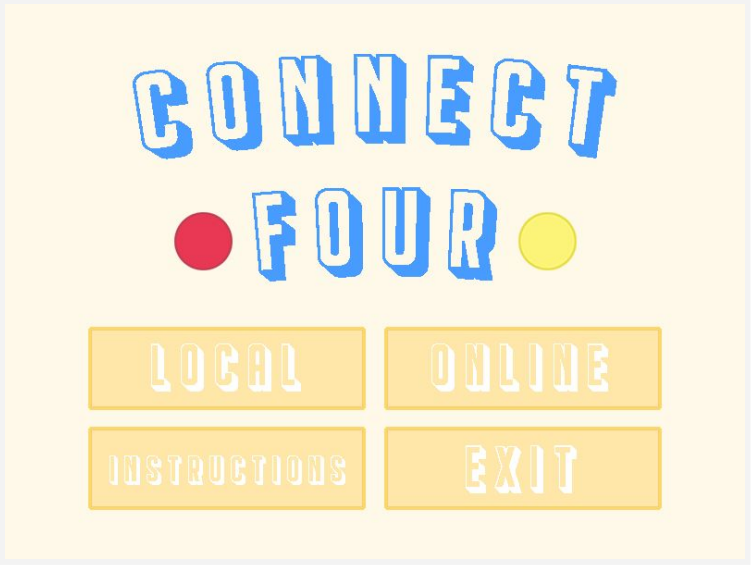


Figura 1 - Menu Inicial

Figura 2 - Botão de Jogo Multiplayer Local

Local: Como o nome indica, é iniciado um jogo multiplayer local.

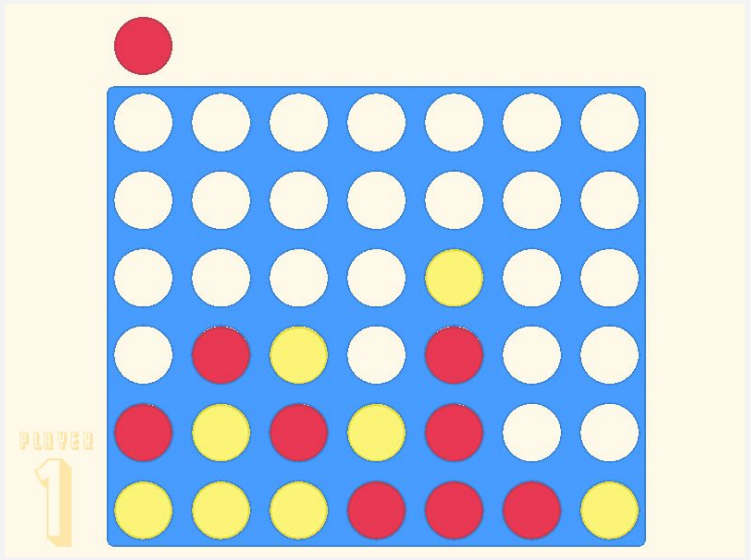


Figura 3 - Jogo Multiplayer Local/Porta-Série

O *jogador 1* deverá utilizar as setas do teclado para escolher a coluna e pressionar a tecla ENTER para jogar a peça. O *jogador 2* deve utilizar os botões esquerdo e direito do rato para escolher a coluna e o botão do meio para fazer a jogada. No caso de não existir botão do meio, pode realizar a jogada pressionando a tecla SPACE.

A qualquer momento, pode ser pressionada a tecla ESC para o jogo ser terminado e regressar ao menu inicial. Após 60 segundos, se o respectivo jogador não mover a coluna, é gerada uma jogada aleatória.

Figura 4 - Botão de Jogo Multiplayer "Online"

Online: É iniciado um jogo multiplayer através da porta série. Este tipo de jogo é semelhante a um jogo multiplayer local, difere apenas no facto de não ser feita uma jogada aleatória após 60 segundos sem alterar a seleção de coluna. Caso o segundo computador não clique no botão Online, o primeiro computador entrará num ecrã de espera, do qual só poderá sair se pressionar a tecla ESC, ou caso o segundo computador se conecte.

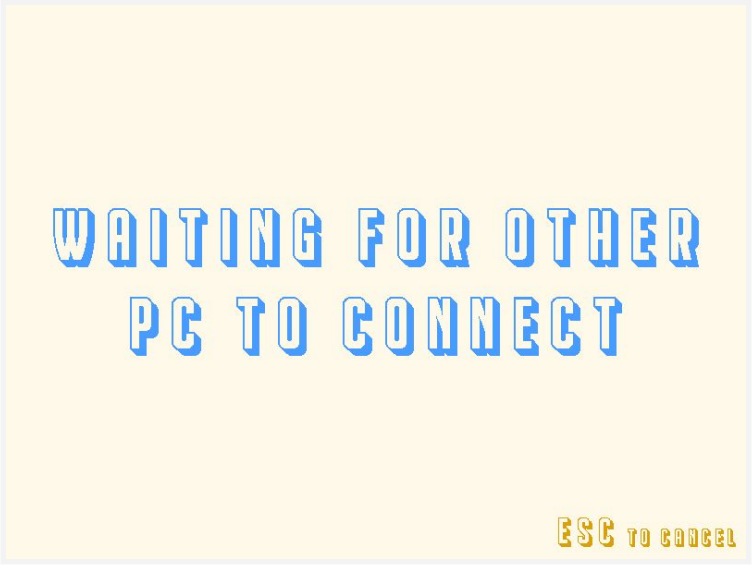


Figura 5 - Ecrã de Espera para Iniciar o Jogo através da Porta-Série

Figura 6 - Botão do menu de Instruções

Instructions: É apresentado um frame com as instruções necessárias e relevantes de acordo com o funcionamento do jogo.

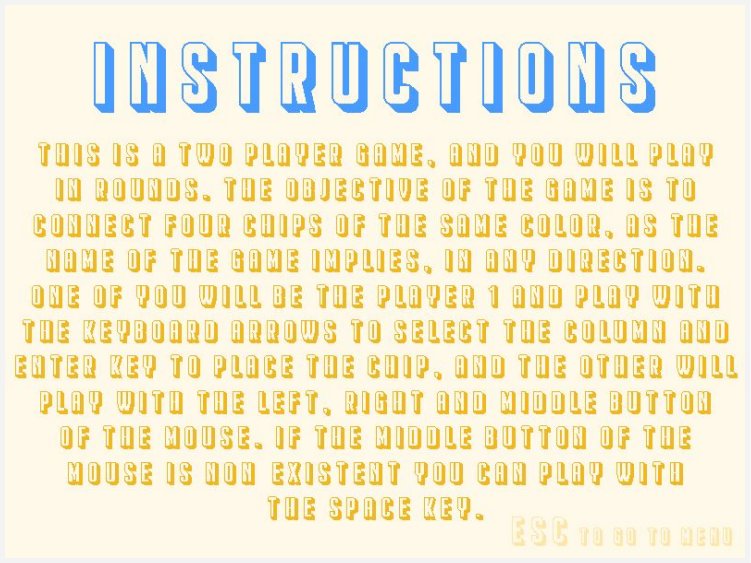


Figura 7 - Tela de Instruções

Figura 6 - Botão Exit

Exit: Sai do jogo, é possível também sair do jogo com o pressionar da tecla ESC.

---

2. Estado do Projeto

Dispositivo	Utilidade	Int.
Timer	Controlar FPS e animações	S
KBD	Jogar	S
Mouse	Navegar o menu e jogar	S
Video card	Mostrar e desenhar ecrãs	-
RTC	Jogada aleatória com alarmes	S
Serial port	Jogo multiplayer online	S

Tabela 1 - Funcionalidade dos dispositivos

Timer: O timer foi implementado para controlar os FPS do jogo, configurá-mo-lo a 60 frames/s. Foi imprescindível para realizarmos as animações de queda das fichas, e de seleção de coluna. As funções utilizadas foram as desenvolvidas no lab2.

KBD: O teclado é utilizado para input do jogador 1, selecionando a coluna através das teclas seta-esquerda e seta-direita e fazendo a sua jogada com a tecla Enter. Também foi configurada a tecla Space para auxiliar o jogador 2. Para além da subscrição e cancelamento de interrupções, foram desenvolvidas funções para lidar com interrupções e escrever para o KBC, keyboard_ih() e kbc_write(), respetivamente. Estas encontram-se implementadas no ficheiro keyboard.c.

Mouse: O rato é utilizado para o jogador 2 navegar nas colunas através dos botões esquerdo e direito, e para efetuar a jogada através do botão do meio. Foi também utilizada a posição do rato para navegar o menu inicial e desta forma saber qual das opções o utilizador está a selecionar. As funções utilizadas foram as desenvolvidas no lab4.

Video Card: Utilizamos o modo gráfico 0x115, com uma resolução de 800x600, e cores de 24 bits, 8 bits por cada componente. Admitimos a cor 0x00b140 como transparente. Implementamos também double buffering e page flipping com o auxílio da função 07h. Para além das funções de iniciar o modo gráfico, de a alocação de memória gráfica (sendo esta o dobro dado que implementamos double buffering), e de coletar a informação acerca do modo gráfico (com chamada à função 01h), set_vbe_mode(), vg_init() e get_mode_info(), respectivamente, implementamos as funções draw_xpm() e draw_part_of_xpm(), que serão explicadas em detalhe mais à frente. As funções estão implementadas no ficheiro graph.c.

RTC: O RTC é utilizado para gerar um alarme após 60 segundos sem o utilizador mover a coluna. Após os 60 segundos é realizada uma jogada aleatória. Para isto foi necessário ler a data/hora atual e configurar um alarme. Para além de subscrever e cancelar interrupções, foi desenvolvido código para ativar e desativar o alarme, nas funções setAlarm() e deactivate_alarm(). Estas encontram-se implementadas no ficheiro rtc.c.

Serial Port: A porta-série está configurada em modo Half-Duplex, com interrupções, sem FIFOS, uma Baud Rate de 19200, transmissão de palavras com 8 bits, 1 stop bit, e sem controle de paridade. Para além da subscrição e cancelamento de interrupções, desenvolvemos as funções de configuração, que configura de acordo com o que foi acima referido, de enviar caracteres, e de receber caracteres/tratamento de interrupção, serial_config(), serial_send() e serial_ih(), respetivamente, presentes no ficheiro serial.c.

3. Estrutura/Organização do Código

Timer: O timer está implementado no ficheiro timer.c e inclui 4 funções: timer_set_frequency(), que, como o nome indica, configura a frequência do timer indicado pelo seu primeiro argumento; timer_subscribe_int() e timer_unsubscribe_int(), que subscrevem e cancelam as interrupções do timer; por fim, timer_get_conf() que solicita a configuração atual do timer.

KBD: O teclado está implementado no ficheiro keyboard.c e foi utilizada uma versão bastante semelhante àquela que foi desenvolvida no lab3. Foram implementadas quatro funções: kbc_susbscribe_int() e kbc_unsubscribe_int(), utilizadas para subscrever e cancelar interrupções, respetivamente; kbc_ih() que processa as interrupções; por último, a kbc_write(), utilizada para escrever no controlador.

Mouse: A implementação do rato é também bastante semelhante à do respectivo lab e encontra-se no ficheiro mouse.c. Este inclui cinco funções: mouse_subscribe_int() e mouse_unsubscribe_int(), utilizadas para subscrever e cancelar interrupções, respetivamente; as funções enable_data_reporting() e disable_data_reporting() para ativar/desativar o envio de informação; a função mouse_interrupt_handler() para processar as interrupções.

Video Card: A implementação da video card encontra-se no ficheiro graph.c. Este ficheiro inclui: set_vbe_mode() que altera o minix para o modo gráfico; vg_init() que aloca a memória de vídeo necessária para dar display a dois frames (double buffering); draw_xpm() que copia as cores armazenadas num array retornado pela função xpm_load() para a zona de memória de vídeo do buffer, na posição indicada pelos seus argumentos; draw_part_of_xpm() que é semelhante à função anterior, mas que desenha parte do XPM em questão, esta função é útil quando é necessário limpar um sprite (sem desenhar todo o background); finalmente a função display_frame() que chama a função 07h, trocando a video_mem pelo buffer (page flipping).

RTC: O RTC foi implementado para a utilização de alarmes e a sua implementação encontra-se no ficheiro rtc.c. Foram criadas funções para subscrever e cancelar interrupções, rtc_subscribe_int() e rtc_unsubscribe_int(), e também para ativar e desativar as interrupções do alarme, activate_alarm() e deactivate_alarm(). Para a criação do alarme, foi necessário implementar a função readDate(), que lê a data e hora atual. Na implementação da função setAlarm(), é lida a data/hora e convertida em segundos, aos quais são adicionados 60 e o alarme é configurado. Foram utilizadas duas funções auxiliares, para passar de BCD para binário e vice-versa. Estas foram adaptadas a partir de código encontrado na web, cujo link encontra-se em no final desta secção.

Serial Port: A implementação da porta-série encontra-se no ficheiro serial.c, nele desenvolvemos as seguintes funções: serial_subscribe_int() e serial_unsubscribe_int() que subscrevem e cancelam as interrupções, respetivamente; serial_config() que configura a porta-série com os valores referidos na secção 3, através dos registos de controle deste dispositivo; serial_send() que, como o nome indica, envia um caractere à outra porta-série; e por fim, serial_ih() que processa as interrupções deste dispositivo quando uma mensagem é recebida, ocorre um erro, ou o THR está vazio.

Game: O jogo está completamente implementado no ficheiro game.c. Este ficheiro trata não só as mecânicas do jogo mas também as animações do mesmo. Este contém as seguintes funções: draw_background(), draw_board(), draw_chip(), draw_column(), draw_player_indicator(), draw_player_won_indicator(), todas estas funções desenham primeiramente no buffer utilizando draw_xpm() os devidos XPM's e chamam display_frame() para fazer display do frame sem ocorrer flashes e/ou artefactos; initiate_game() para além de fazer xpm_load() de todos os XPMS, column_select_animation(), como o nome indica, processa a animação de escolha de coluna; random_play_column_select_animation() processa a animação de escolha da coluna retornada pela função randomPlay(); chip_falling_animation() processa a animação de queda da ficha na respetiva coluna; validPlay() retorna um booleano, este é verdadeiro caso a jogada for válida e falso caso não seja; makePlay() atualiza o array Board mediante a coluna selecionada e o player que está a jogar; checkGameOver() verifica se o jogador colocou quatro peças em linha; game_over() que processa o final de jogo ao chamar a função draw_player_won_indicator() e espera 5 segundos antes de regressar ao menu; nextPlay() inicia a ronda seguinte ao chamar draw_player_indicator(), draw_chip(), e ao alterar o jogador.

Menu: O menu está implementado no ficheiro menu.c, este inclui as seguintes funções: startMenu() que carrega todos os XPMs e desenha o background do menu; showMenu() que desenha o cursor do rato e realça os botões caso o cursor esteja por cima de um dos botões; showOnlineWaitingMenu() que desenha o menu de espera quando um dos computadores quer conectar-se ao outro computador, mas o último ainda não selecionou o jogo através da porta-série; e por fim, showInstructionsMenu() que desenha o menu de instruções.

Proj: No ficheiro fornecido, proj.c, está um único loop que coordena todo o nosso jogo e menus, com o auxílio de máquinas de estado (que falaremos mais em detalhe na Secção 4), e todas as funções previamente referidas ao longo de todos estes ficheiros.

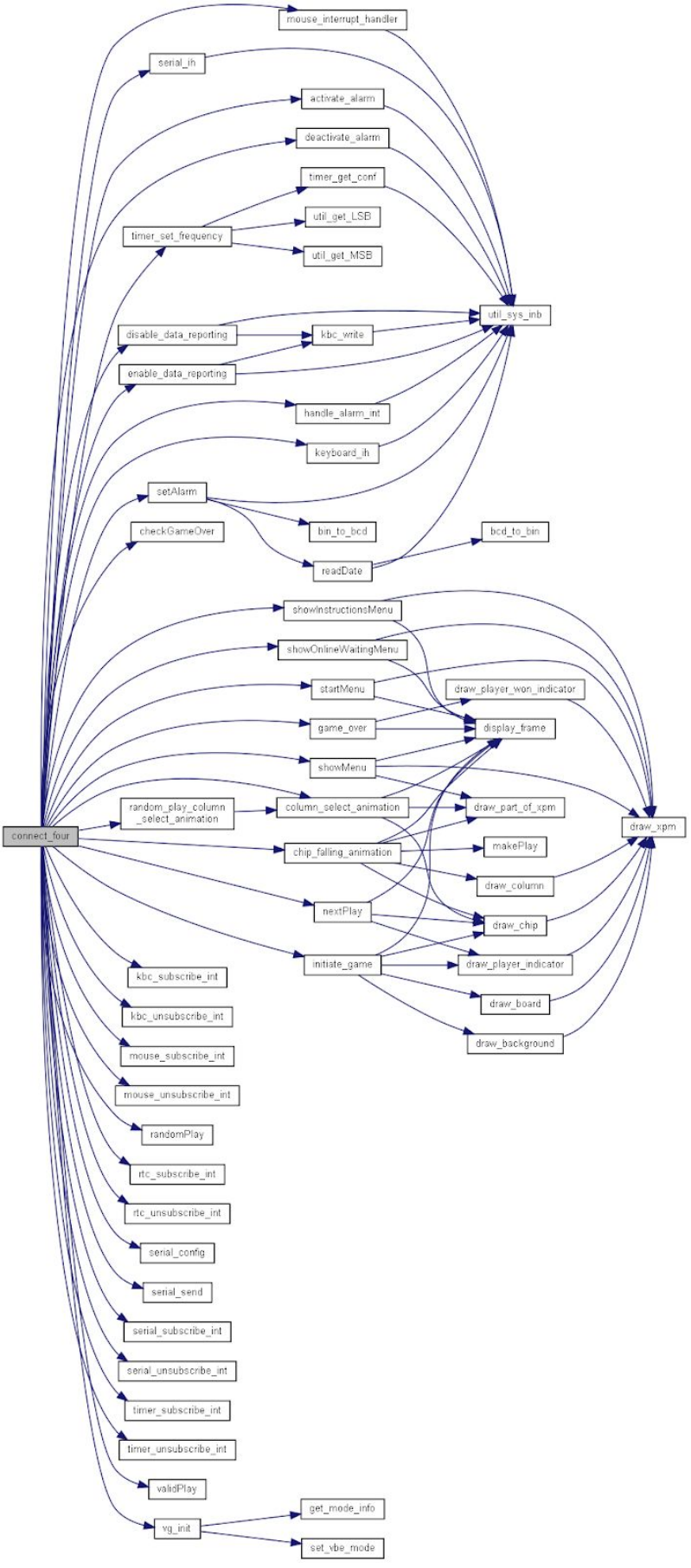
Peso dos módulos

Módulo	Peso	Adriano	Vasco
timer.c	5%	50%	50%
keyboard.c	5%	50%	50%
mouse.c	5%	50%	50%
graph.c	15%	60%	40%
rtc.c	10%	10%	90%
serial.c	15%	100%	0%
game.c	15%	50%	50%
menu.c	15%	60%	40%
proj.c	15%	60%	40%

Tabela 2 - Peso relativo dos módulos

Código Retirado da Web: Foi retirado e adaptado código da web que permite transformar um número de BCD para binário e vice-versa. Link: https://stackoverflow.com/questions/13247647/convert-integer-from-pure-binary-to-bcd

Gráfico das chamadas de Função:



4. Detalhes da Implementação

Máquinas de Estado:

Para o funcionamento do jogo, foram implementadas três máquinas de estado no loop que o coordena, em proj.c. Estas foram criadas para controlar tanto o próprio programa, como o menu e também o jogo. O programa em si é uma máquina com 4 estados, o MENU, o MENU DE ESPERA DA SERIAL, o JOGO e o MENU DE INSTRUÇÕES, estados 0, 1, 2, e 3 respetivamente.

O estado inicial é o MENU, cuja implementação é também uma máquina de estados, com 4 estados, isto porque dispõe de quatro botões. O 1º estado é relativo ao botão LOCAL, o 2º ao botão ONLINE, o 3º ao botão INSTRUCTIONS e o 4º ao botão EXIT. Sempre que o rato se move o estado pode ser alterado, isto se o cursor estiver por cima de um dos botões, de modo a alterar o destaque dos mesmos. Caso o utilizador selecione um dos botões o estado do programa alterar-se-á de acordo. Por exemplo: caso o utilizador selecione o botão de jogo ONLINE, o estado do programa passará para 1, de acordo com o que está acima referido.

Tanto o estado MENU DE ESPERA DA SERIAL como o MENU DE INSTRUÇÕES apenas fazem display dos backgrounds correspondentes e alteram o estado do programa caso a tecla ESC seja pressionada, voltando para o menu.

Relativamente ao estado de JOGO, este também é uma máquina de estados, com 6 estados: 0, quando não está a fazer nada; 1, caso o jogador queira mover a coluna de seleção para a esquerda; 2, caso o jogador queira mover a coluna de seleção para a direita; 3, caso o jogador peça para jogar uma ficha numa determinada coluna; 4, quando uma jogada random é efetuada; e por fim, 5, caso o jogo tenha acabado. Quando está no estado 5, após 5 segundos de mostrar o vencedor o utilizador regressa ao MENU (estado 0 do programa).

RTC:

Implementamos o RTC, utilizando este apenas para contar 1 minuto sem que nenhuma ação seja feita no modo de jogo multiplayer local. Sendo assim, apenas o configuramos no início do programa, e caso o jogador pressione o botão LOCAL, este será ativo. Após o final do jogo, seja porque alguém pressionou a tecla ESC ou não, este dispositivo é desativado.

Placa Gráfica:

Inicialmente estava previsto utilizarmos o modo 0x14C, com resolução 1152x864, e 32 bits por cor, 8 bits por componente, mais 8 bits para a transparência, no entanto, este modo demonstrou-se ser demasiado intensivo, resultando em animações e frame-rates muito lentas. Isto porque desenhamos 1152x864 x 3 bytes = 3.981.312 bytes por cada vez que fazemos display de algo. Deste modo alteramos para o modo 0x115, muito menos intensivo (1.440.000 bytes). No entanto, só após a implementação de page-flipping, double-buffering e da função draw_part_of_xpm() presente em graph.c, é que obtivemos resultados aceitáveis.

A função draw_part_of_xpm() é utilizada para desenhar o background por cima do sprite de modo a não desenharmos o frame todo de raiz. O único entrave que encontramos foi o facto de termos de eliminar o sprite atrasado por dois frames e não por um, isto porque não eliminamos o conteúdo da video-mem após fazer page-flipping. Visto que as animações do JOGO são feitas através de uma função posição-tempo, foi relativamente fácil a sua implementação em game.c. No entanto, no MENU tivemos que utilizar um array que é atualizado com as duas posições do rato anteriores.

Porta-Série:

Relativamente à porta série, está configurada em modo Half-Duplex pois este jogo tem a característica de ser jogado por rondas, deste modo apenas temos que ter atenção em qual dos computadores foi pressionado primeiro o botão ONLINE, e definimo-lo como o HOST, passando este a ser o que joga primeiro, ou seja, o Player 1. Caso o utilizador prima o botão ONLINE, um pacote SER_REQUEST é enviado para o outro computador, o segundo computador atualiza o booleano HOST para false quando a mensagem é recebida, e entra no MENU DE ESPERA DA SERIAL (estado 1 do programa). Quando o segundo computador prime no mesmo botão, este envia um pacote SER_CONFIRM, para o jogo ser iniciado, e o booleano established_connection passa a true.

Quando são pressionados os botões de seleção de coluna correspondentes apenas enviamos o respetivo estado da máquina de estados relativa ao JOGO. Caso um dos jogadores pressione a tecla ESC, é enviado um pacote SER_EXIT, que faz com que o jogo termine em ambos os lados, o booleano established_connection passa a false, e o HOST a true.

5. Conclusão

Consideramos que a realização deste projeto contribuiu para consolidarmos e melhorarmos o nosso conhecimento sobre e programação de periféricos. Embora trabalhoso, com o projeto sentimos um sentimento de recompensa, o que não aconteceu nas aulas laboratoriais, onde sentimos algumas vezes que a carga de trabalho era desproporcional ao tempo de aula. Entraremos em mais detalhes nos formulários de auto-avaliação.