U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

PONG

Trabalho prático - LC 2021/2022

Grupo T04-G07

Bruna Brasil Leão Marques - up202007191
Dinis Ribeiro dos Santos Bessa de Sousa - up202006303
Francisca Oliveira e Silva - up202005140
João António Maricato Malva - up202006605

Índice

Introdução

Para o nosso projeto, propusemo-nos criar um jogo de Pong. Este foi um dos primeiros jogos de computador desenvolvidos.

Tal como o original, este jogo de Pong tem dois modos de jogo: Jogador contra o computador e Jogador contra Jogador. A arena é dividida ao meio e há uma baliza de cada lado. O jogador controla um paddle de um dos lados da tela e compete contra o computador ou contra outro jogador que controla um paddle do lado oposto da arena. Ambos os jogadores usam os seus paddles para enviar a bola para o lado oposto, marcando pontos, enquanto evitam que o seu adversário faça o mesmo. Um ponto é marcado quando a bola passa pelos limites do campo do adversário. A partida termina quando um dos jogadores atingir uma pontuação de 5 pontos, tornando-se o vencedor.

No primeiro modo será possível utilizar poderes úteis no jogo, fugindo às regras do que seria um Pong normal.

Manual de Utilizador

Menu Inicial

Ao iniciar o jogo, o utilizador irá deparar-se com o menu inicial onde é possível visualizar as opções: “1 Player”, “2 Players” e “About", que correspondem respetivamente aos modo de jogo jogador contra computador, jogador contra jogador e informações. A opção selecionada estará a laranja e poderá ser alterada com as setas, para cima e para baixo, e executada ao premir a tecla ENTER.

Dependendo da altura do dia, o menu muda de aspeto. (Fig1. e Fig2.)

1 Player

Modo de jogo semelhante ao pong original, o objetivo é fazer com que a bola entre na baliza do ‘jogador’ adversário, que será controlado pelo computador. Para isso pode mover o seu paddle, usando as teclas ‘W’ para subir e ‘S’ para descer.

Quando a bola entra na baliza adversária, o jogador marca um ponto e a sua pontuação é atualizada.

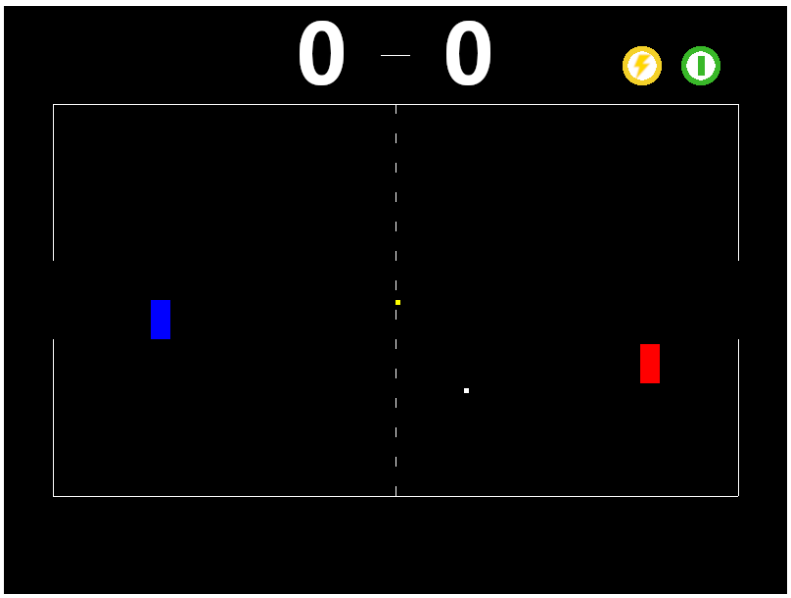

Fig.3 - powerups disponíveis para utilização

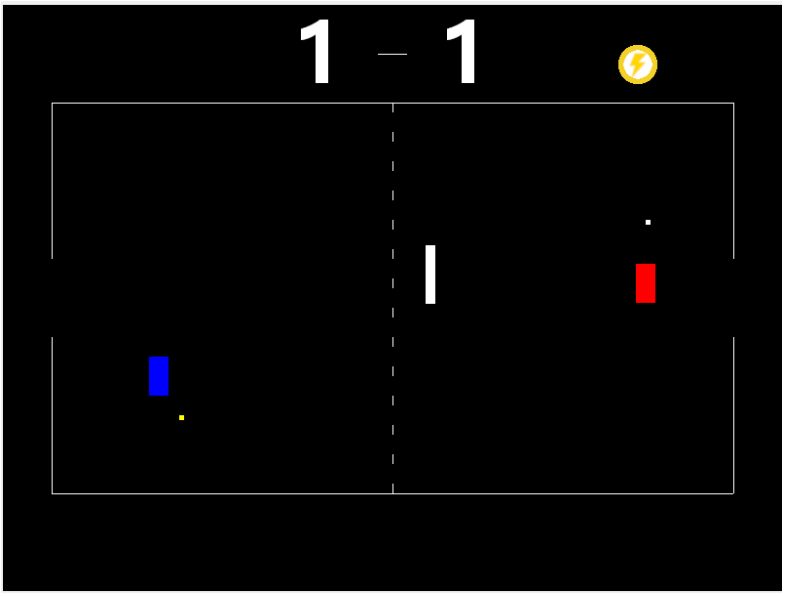

Fig.4 - powerup da parede ativo

Won/Lost

Menu final que mostra se o jogador ganhou ou perdeu o jogo.

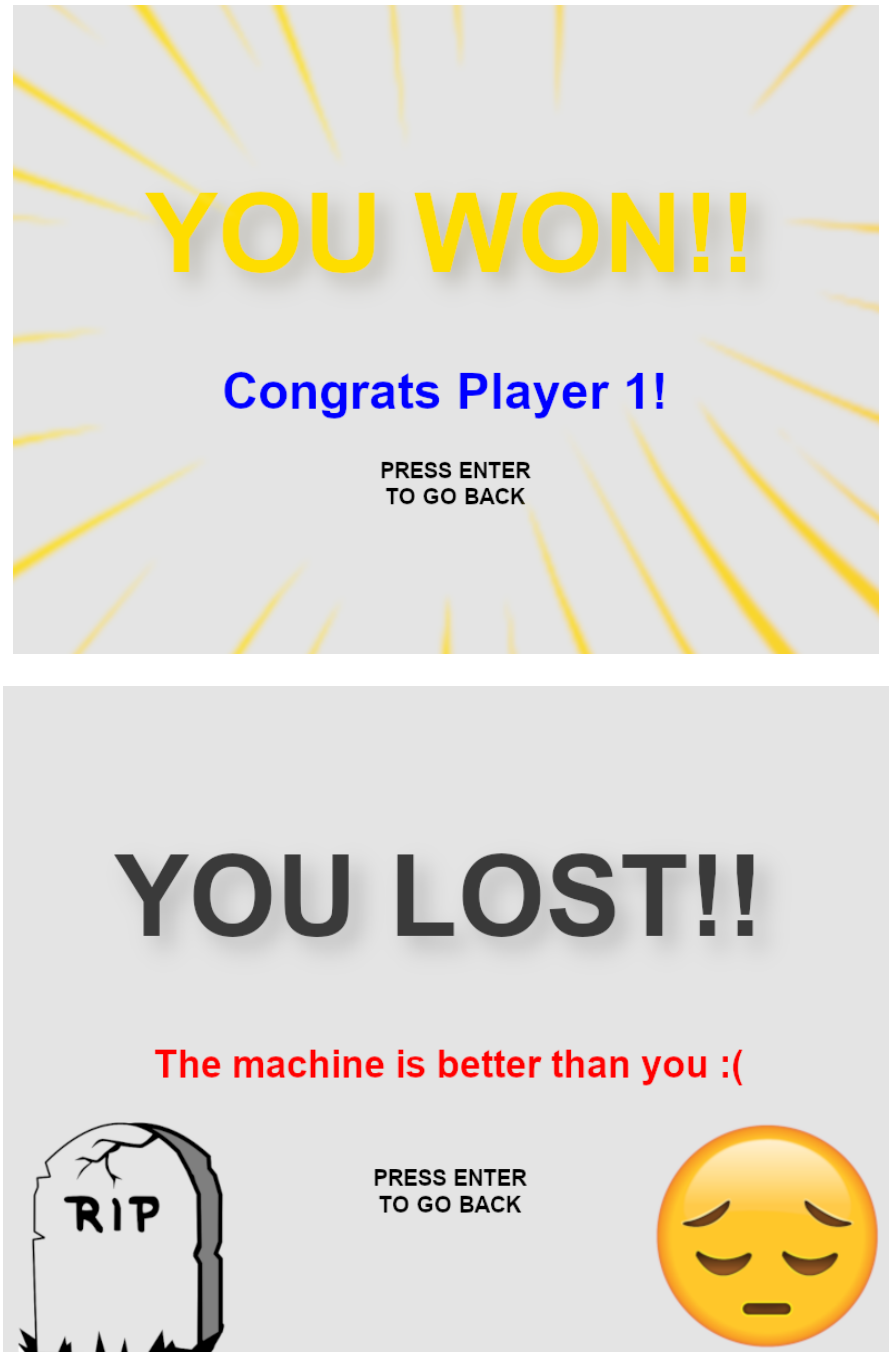

Fig.5 - won menu

Fig.6 - lost menu

2 Players

Modo de jogo igual a apenas 1 jogador, mas corre em duas vm's distintas. O azul é azul em ambos e o vermelho também.

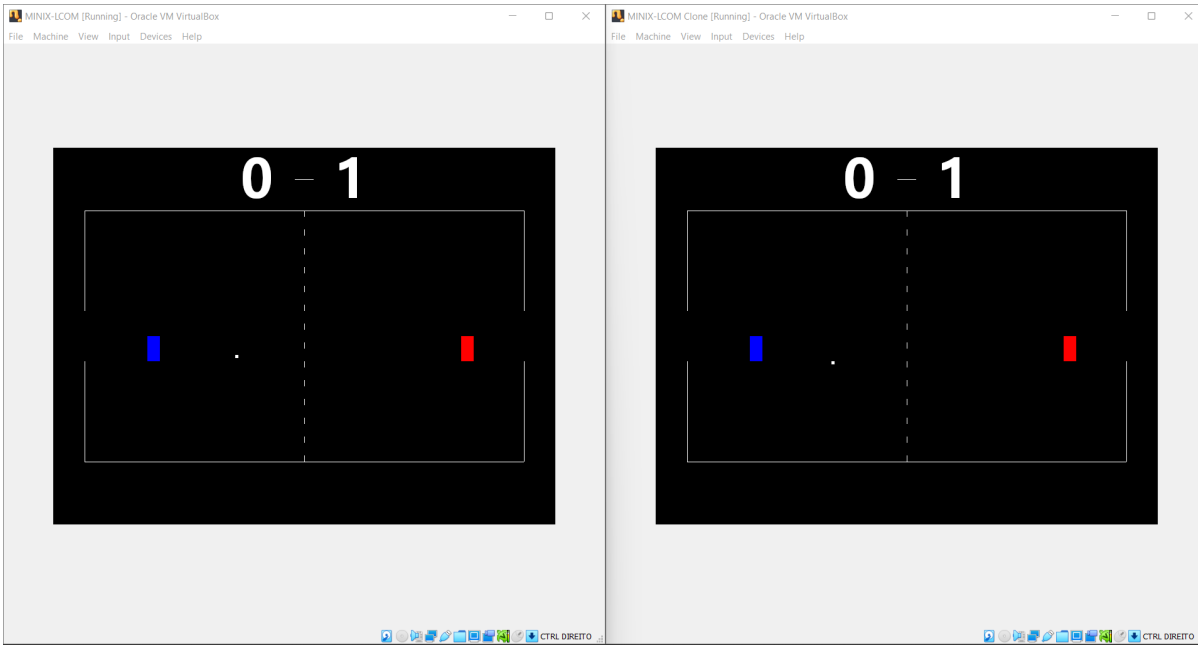

Fig.7 - jogo para dois jogadores

About

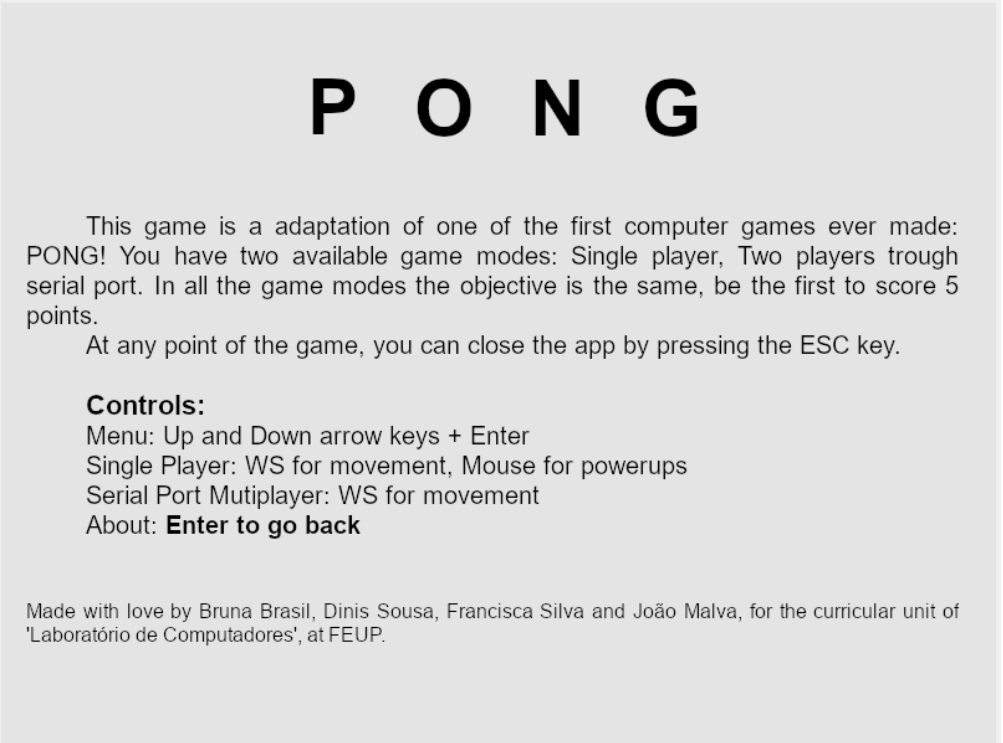

Fig. - menu about

Descrição e instruções de jogo para guiar o utilizador.

Estado do Projeto

Timer

O timer é usado para controlar o frame-rate do ecrã, que será atualizado 30 vezes por segundo.

O timer controla também o movimento da bola, vendo a cada frame se a bola colidiu com algum paddle ou entrou numa das balizas, verificando se é necessário atualizar o score ou acabar o jogo.

No modo de jogo de um jogador contra o computador, o timer controla o tempo de espera da recarga dos powerups (aumentar a velocidade da bola com o mouse esquerdo e colocar uma parede na posição do rato com o mouse direito). É também o timer que controla o

rate com que a parede vai decrescendo.

As funções relacionadas com o timer estão no ficheiro timer.c.

Keyboard

O keyboard é usado para ler os inputs do user no teclado. A qualquer momento do jogo, ao premir ESC o jogo acaba.

No menu, são usadas as setas para cima e para baixo para circular entre as opções, e o ENTER para escolher um modo de jogo.

O jogador controla o seu paddle com as teclas W e S, para ir para cima e para baixo respectivamente, em qualquer um dos modos de jogo.

No menu ‘About’ e no final de cada jogo, o jogador deverá premir ENTER para voltar ao menu.

As funções relacionadas com o keyboard estão no ficheiro kbc.c.

Mouse

O mouse é usado dentro do modo de jogo single player, para ativar os poderes.

Ao clicar no botão esquerdo do rato, quando o ícone do powerup estiver no ecrã, a bola aumenta de velocidade, num fator de 1.4x. A ação volta a estar ativa passados 15 segundos.

Ao clicar no botão direito do rato, quando o ícone do powerup estiver no ecrã, uma parede é colocada no local onde está o mouse, e decresce com o tempo até desaparecer. A ação volta a estar ativa passados 10 segundos, após desaparecer.

As funções relacionadas com o mouse encontram-se no ficheiro kbc.c.

Graphics Card

Neste projeto foi utilizado o modo de vídeo 0x115, 800x600.

A gráfica é usada para servir de interface ao utilizador, para ele poder ver o jogo e melhor interagir com o programa.

Foi usada uma técnica de double buffering, de modo a evitar problemas como 'flickering', em que a cada interrupção do timer eram desenhados os conteúdos do jogo para um buffer auxiliar e depois o conteúdo desse buffer era copiado para o buffer principal, buffer este que era um 'virtual adress space' correspondente a memória principal.

A gráfica é usada para desenhar ficheiros XPM, que contêm imagens para escrever no jogo, através da função draw_sprite() (Menu, Opções do Menu, Resultado, Ícones do poderes, About e Menu de Fim de Jogo) e também linhas horizontais e verticais, através das funções vg_draw_hline() e vg_draw_vline() (arena) e retângulos, através da função draw_rectangle() (paddles, bola, rato, parede).

A transparência é dada pela cor 'None', que corresponde a TRANSPARENCY_COLOR_8_8_8_8.

As funções relacionadas com a gráfica encontram-se no ficheiro video.c.

Real Time Clock

O RTC é usado para mudar o aspeto do menu inicial dependendo da hora do dia. Este chama a função rtc_update_darkmode() para aceder ao registo das horas(apenas depois de verificar se não está a decorrer nenhum update aos valores), converter o valor lido em BCD para hexadecimal, e determinar se é de noite ou de dia. Inicialmente esta função era chamada todos os minutos, no entanto como so o menu é afetado pela hora do dia, esta função é chamada apenas no início.

Assim, não há necessidade de lidar com as interrupções.

Todas as funções relacionadas com o RTC estão implementadas em rtc.c.

Serial Port

A serial port é utilizada no modo multiplayer do jogo, para transmitir informação sobre a posição do player e o estado atual do jogo. Para isso foram definidas várias constantes auxiliares, ou seja, números de 8 bits com um determinado significado.

Para iniciar a ligação, ambas as vm's enviam SER_INIT para sinalizar que estão prontas para começar o jogo. Aquela que tiver selecionado primeiro a opção multiplayer do menu será a única a conseguir receber o código da outra, logo deve ser essa a começar o seu jogo e a mandar SER_START para a outra vm para esta começar também ao mesmo tempo. Essa primeira faz as trocas necessárias para tornar o seu player no player2 (vermelho). A função que determina quem deve fazer as trocas é ser_check_connection().

Ao longo do jogo, há 3 tipos de mensagens:

- do 0 ao 20(width da tela / velocidade da bola), que manda informação sobre a posição do player após um movimento. Em vez de enviar apenas um código a sinalizar movimento para cima ou para baixo, é transmitida a posição para evitar possíveis perdas de informação no envio. É de salientar que ao ler a informação do Reciever Buffer Register após uma interrupção, é lida toda a informação que tenha sido enviada até ao momento para manter a posição o mais sincronizada

possível. Apenas quando é recebido um dos outros tipos de mensagem é que não é esvaziado o buffer.

- SER_GOAL_1 e SER_GOAL_2, que informa a outra vm de que ocorreu um golo e que, se a segunda ainda não o conseguiu identificar, deve aumentar os scores e fazer refresh a jogada
- SER_WINNER, um número para ser somado a variável winner utilizada para identificar quem ganhou a jogada. A necessidade deste offset é para não se confundir com os dados da posição do jogador

São utilizadas interrupções apenas para a Recieved Data, que são permitidos na função ser_init() (também utilizada para limpar o buffer de qualquer lixo que possa ter sido deixado por processos anteriores). Ao receber uma interrupção, é chamada a função ser_read_data() que lê o Line Status Register para saber se realmente há informação disponível antes de proceder à execução explicada anteriormente. Esta apenas atualiza a variável read_data para ser usada no resto da execução.

Para transmitir dados, é necessário primeiro ler a LSR para saber se o THR está vazio e pronto para guardar a informação no buffer. Quando isto se verificar a funçao ser_transmit_data() pode enviar os dados para o THR.

Todas as funções relacionadas com a Serial Port estão implementadas em serialport.c.

Estrutura de código

Timer Module (7%)

Neste ficheiro estão presentes as funções desenvolvidas no Lab2, relacionadas com as interrupções do timer. Realizado pelos Dinis Sousa e Francisca Silva.

Keyboard and Mouse (KBC) Module (16%)

Neste ficheiro estão presentes as funções desenvolvidas no Lab3 e Lab4, relacionadas com as interrupções do keyboard e do mouse. Realizado igualmente pelos Dinis Sousa e Francisca Silva.

Video Module (7%)

Neste ficheiro estão presentes as funções desenvolvidas no Lab5, relacionadas com a placa de vídeo. Realizado pelo Dinis Sousa.

RTC Module (7%)

Onde encontram-se as funções de comunicação com o RTC, quer para subscrever as interrupções, quer para ler os registos, para conversão de BCD para hexadecimal e assim, atualizar a variável bool 'darkmode' conforme se está de dia ou de noite. Realizado pela Francisca Silva

Serial Port Module (12%)

Neste ficheiro encontram-se as funções da serial port, nomeadamente para subscrever, configurar, enviar e ler informação. Realizado pela Francisca Silva.

Images Module (5%)

Neste ficheiro são carregadas todas as sprites para um array de imagens. Realizado igualmente pelos Bruna Brasil e Dinis Sousa.

View Module (5%)

Neste ficheiro estão presentes as funções que permitem renderizar a informação presente no modelo do jogo, de forma interativa para o utilizador. Realizado igualmente pelos Bruna Brasil e Dinis Sousa.

Model Module (20%)

Neste ficheiro estão presentes as funções que são responsáveis pela lógica do jogo, entre elas: seleção da entrada no menu, movimento dos paddles do jogo, ativar os poderes do jogo, colisão da bola com as paredes e paddles. É responsável ainda por guardar a informação, da posição, cor, tamanho e outras características pertinentes sobre os diferentes objetos do jogo. Realizado por Bruna Brasil, Dinis Sousa e João Malva.

Handlers Module (20%)

Neste ficheiro está uma máquina de estados que controla os diferentes estados do jogo, sendo constituída por um mainHandler, que chama um handler que seja correspondente ao estado atual. Os handlers de cada estado podem alterar o estado e chamar funções que mudam o model e atualizam a view do jogo. Realizado por todos os elementos do grupo.

Detalhes da Implementação

Foi implementada uma máquina de estados, controlada pelo mainHandler(), e em que o estado é guardado na variável ‘state’ o estado atual do jogo. Cada estado tem a sua forma de lidar com as interrupções geradas pelos periféricos, que será naturalmente diferente se o utilizador estiver no jogo ou no menu, por exemplo.

O jogo foi implementado numa arquitectura MVC (model-view-controller), em que o modelo guarda a lógica do jogo, o view renderiza a informação do jogo e o controller (neste caso representado pelo ‘handler’) chama funções do model e da view aquando de interrupções do utilizador.

No menu, para mudar entre entradas aquando do input da tecla do user são chamadas as funções addMenuEntry() e backMenuEntry(). Quando o utilizador prime ENTER para selecionar uma entrada, a função getCurrentEntry() é chamada, e o estado é alterado para o estado escolhido.

No jogo, para detetar colisões entre a bola e os outros elementos do jogo, são utilizadas as funções:

- ballCollidesPlayer(): recebe cada player por referência e verifica as posições da bola e do player e retorna true em caso de colisão. Conforme este retorno, a bola inverte o seu sentido;

- ballCollidesWall(): verifica se há colisão entre uma parede temporária (um dos ‘power ups’), se estiver ativa, e a bola;

- dentro da função moveBall(), compara-se a posição da bola com as extremidades da arena, com exceção do espaço que a cada baliza ocupa, e muda o sentido do movimento em caso de colisão.

A cada jogador está associado uma pontuação. A função goal() verifica se a bola ultrapassou uma das extremidades laterais da arena, uma vez que dentro das balizas não ocorre colisão. Em caso de golo, incrementa a pontuação do jogador que marcou e reseta as posições dos jogadores e da bola.

O jogo chega ao fim quando a função gameWinner() define um jogador como vencedor, isto é, quando marca cinco golos.

No entanto, no modo multiplayer ocorre um erro quando o adversário ganha o jogo. Por algum motivo que não conseguimos identificar, o próprio jogador percebe sempre que ganha mas não que perde.

Conclusão

Após o desenvolvimento do projeto, podemos dizer que este foi sem dúvida um trabalho difícil, mas muito gratificante e que valeu a pena o trabalho investido. Todos os objetivos a que nos propusemos foram alcançados, e ainda implementamos o RTC.

Sendo esta a primeira experiência do grupo com drivers de periféricos, foi difícil, principalmente no início, saber como e por onde começar. Algo que se aplicou à maior parte dos labs, é que apesar de ser difícil começar, depois de perceber a utilidade e a forma de comunicar com o periférico, o trabalho tornava-se mais eficiente. No entanto, até chegar a esse ponto demorava muito tempo, sendo a maior parte do tempo investido num lab não gasto a programar mas sim a ler documentação e andar para trás e para a frente nos guiões dos labs.

A informação dos labs, ainda que desorganizada, estava bastante completa e foi uma ajuda no desenvolvimento dos mesmos.

Achamos ser muito bom poder chegar ao final do semestre e poder criar um projeto que tenha implementado todo o trabalho do resto do semestre, isso foi recompensador do trabalho realizado ao longo do semestre das aulas e é uma boa sensação poder ver todos os pedaçinhos daquilo que foi um trabalho contínuo se juntar em algo mais complexo e concreto.

Houve dificuldades em fazer o RTC e a porta série, devido à falta de informação. Seria benéfico disponibilizar os labs no moodle de forma semelhante aos restantes labs.