Implementação do jogo PONG num sistema de tempo real

Bruno Moreira, Igor Carvalho, Leonel Peixoto, Liliana Ribeiro
Mestrado Integrado em Engenharia Informática e Computação
Faculdade de Engenharia da Universidade do Porto
ei12012@fe.up.pt; ei12178@fe.up.pt
ei12077@fe.up.pt; ei08161@fe.up.pt

***Abstract* — Este artigo descreve a implementação do jogo denominado Pong utilizando um comunicação via WIFI para a interação entre os utilizadores e o sistema. Tratando-se um projecto em tempo real, todas as medidas necessárias para uma correta especificação e operação do sistema irão ser discutidas, tais como, escalonamento de tarefas, tempos de execução e tempos de resposta. Para além dos principais componentes em tempo real, irá ser abordado a configuração do sistema Wi-Fi.**

I. INTRODUÇÃO

O objectivo deste trabalho é desenvolver um sistema em tempo real com escalonamento de tarefas. Para isto foi implementado o jogo do Pong, usando um Arduíno e cujos jogadores podem interagir através de uma aplicação Androíd.

II. ESPECIFICAÇÃO DO SISTEMA

A. Hardware utilizado

O sistema desenvolvido encontra-se constituído pelos seguintes elementos físicos:

• Um Arduíno Nano V3.0 como um microcontrolador ATmega328 da Atmel com arquitetura AVR de 16MHz (1 instrução/ciclo), 32 KB de *flash* e 2 KB de RAM.
• Um LCD monocromático de 84 por 48 pixéis com um controlador PCD8544 da Philips com *buffer* de 504 Bytes e interface SPI até 4 Mbit/s.
• Um módulo Wi-Fi versão ESP-01 com um microcontrolador ESP8266 da Espressif com interface RS232.

A comunicação entre o ATmega328 do Arduíno e o PCD8544 do LCD é feita através do protocolo SPI em que o ATmega328 se comporta como *master* e o PCD8544 como *slave* sendo que este último apenas recebe dados. O LCD possui dois modos de funcionamento importantes: modo de comandos e modo de dados, ambos selecionados pelo estado do pino D/C. O modo de comandos permite iniciar e configurar o LCD com determinadas características inerentes a este tipo de dispositivo como por exemplo o contraste e a frequência de resposta do ecrã LCD. O modo de dados permite enviar normalmente dados *byte* a *byte* para serem visualizados no LCD.

A interface com o Arduíno e o ESP8266 do módulo Wi-Fi é efetuada pela porta série no protocolo RS232 a 38400 *bauds* com 8 bits de dados, 1 *start* bit, 1 *stop* bit e sem paridade (8N1). Contudo, a configuração do módulo deve ser feita através de comandos do tipo "Heyes AT" que se encontram referenciados no final deste artigo. Como tal, no contexto deste
trabalho, na fase de arranque do sistema, o Arduíno envia um conjunto de comandos sucessivos para configurar o módulo sendo que este deve responder "OK" para indicar que o comando foi aceite:

• AT – verifica que o módulo está pronto a funcionar;
• AT+CWMODE=2 – ativa o módulo com sendo um *Access Point* (AP);
• AT+CWSAP="Pong"," ",1,0 – Configura o SSID do AP no canal 1 sem encriptação;
• AT+CIPMUX=0 – Permite aceitar uma ligação UDP simples;
• AT+CIPSTART="UDP","0",0,10000,2 – Abre um servidor UDP na porta 10000.

Depois do módulo estar devidamente configurado, este fica a escuta de pacotes UPD recebidos pela porta 10000 sendo que reencaminha os dados recebidos até ao Arduíno no formato "+IPD,n,d" em que n é o número de *bytes* recebidos e d os dados efetivamente recebidos. No contexto do jogo implementado, os dados recebidos serão sempre um único caracter que pode tomar 4 valores diferentes: 2 movimentos para cada um dos dois jogadores.

B. Esquema do circuito

No que respeita a alimentação do sistema, esta é fornecida pelo *host* ligado a porta USB do Arduíno. Com isso o Arduíno já se encontra alimentado com 5V, contudo, o LCD e o módulo Wi-Fi só podem serem alimentados com 3,3V. É então usada a saída 3,3V do Arduíno para alimentar o LCD, porem essa pode não fornecer intensidade suficiente para alimentar o módulo Wi-Fi que em modo AP requer cerca de 300mA. Como tal optou-se pelo uso de um transístor genérico NPN montado em regulador de tensão para alimentar o modulo Wi-Fi usando a linha de 5V que pode fornecer a intensidade requerida. Cada linha de alimentação é filtrada por um condensador de 47uF ligado ao GND como é possivel verificar na figura 1.

As linhas que controlam o LCD encontram-se ligadas ao Arduíno através de resistências de 4,7KΩ que em conjunto com os díodos de proteção internos do LCD permitem fazer a conversão de tensão de 5V para 3,3V requerido pelo LCD respeitando a seguinte configuração:

• Pino 13 - SCK (LCD *serial clock*)
• Pino 11 - SDIN (LCD *serial data input*)
• Pino 5 - D/C (LCD *Data/Command select*)
• Pino 4 - CS (LCD *chip select*)
• Pino 3 - RST (LCD *reset*)

Por outro lado, o Arduíno comunica e controla o módulo Wi-Fi pela seguinte configuração:

• Pino 1 - RXD (WIFI RS232 *receive data*)
• Pino 2 - TXD (WIFI RS232 *transmit data*)
• Pino 8 - CD_PD (WIFI *chip enable*)
• Pino 9 - RST (WIFI *reset*)

De notar que a linha RXD do módulo WIFI está ligado ao GND

através de uma resistência de 1,5 KΩ que em conjunto com a resistência de 1KΩ interna da linha TXD do Arduíno, permite assegurar que o módulo recebe dados com uma tensão de 3,3V. Demais, o pino CD_PD também se encontra ligado ao GND através de uma resistência de 4,7KΩ de forma a garantir que o módulo esteja inativo enquanto o Arduíno não colocar o estado desse pino a alto durante a fase de arranque do sistema.
O pino 2 do Arduíno permite gerar um sinal sonoro através de um *buzzer* passivo protegido por uma resistência de 1KΩ.

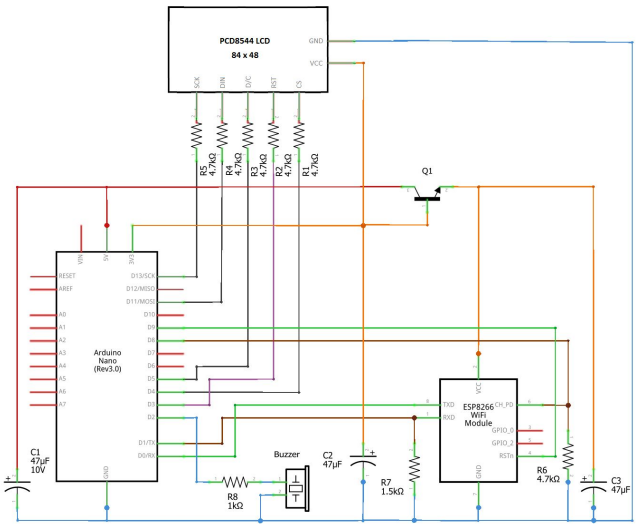

Fig. 1 Esquema do circuito.

C. Aplicação Android

Para controlar o seu raquete, cada um dos dois jogadores pode usar botões pulsadores que não estão representados no circuito anterior. Porem também pode ser usado qualquer dispositivo Android com interface Wi-Fi usando a aplicação desenvolvida em paralelo com este sistema. Essa aplicação permite controlar o raquete através do inclinômetro do dispositivo ou simples botões apresentados pela aplicação. A aplicação é composta por dois *threads* principais: um *thread* que trata de capturar os movimentos do utilizador e outro *thread* que trata de enviar os movimentos encapsulados em pacotes UDP.

III. MICRO-KERNEL

A. Inicialização

No arranque do sistema, numa primeira fase, o LCD e o modulo WIFI são inicializados de forma completamente sequencial. Numa segunda fase, inicia-se variáveis de estado do jogo e adiciona-se então à lista de tarefas do *kernel* as 4 principais tarefas que permitem tornar o sistema funcional em tempo real. Finalmente, são ativadas interrupções em função do timer1 do microcontrolador do Arduíno tornando o sistema "*tickbased"*.

B. Scheduler

A parte responsável por decidir qual a tarefa a ser executada de cada vez, o *scheduler*, é uma peça fundamental do *kernel*, visto que, não só assegura que todas as *deadlines* das tarefas são cumpridas, como também controla a ordem de execução. A implementação foi baseada no código disponibilizado na aula e é constituído por três partes fundamentais: o processamento das interrupções do sistema, o *sheduler* e o *dispatcher*. Quanto às interrupções estas têm como objectivo determinar quais as tarefas que precisam de ser ativadas. Visto que, o tempo de utilização de cada tarefa poderia ser melhorado foi decidido, neste caso, provocar uma interrupção a cada 512µs através do timer1 do Arduíno. Em relação ao scheduler este é responsável por decrementar uma variável chamada *delay* para o controlo de execução das tarefas, ou seja, define qual a tarefa pronta a ser executada. Por fim o *dispatcher* irá verificar qual a tarefa com a maior prioridade que está pronta a ser executada na lista de tarefas, após a verificação, chama a próxima tarefa. Concluindo, é utilizado o método de prioridades de tarefas, em que, a primeira tarefa a ser executada é aquela que foi adicionada em primeiro na lista de tarefas.

C. Tarefas

Para que o jogo consiga ser executado foram criadas quatro tarefas, todas elas periódicas com um período de 21 ticks.

A tarefa *gameLogic*, é a primeira tarefa a ser ativada. É esta tarefa que está responsável por implementar a lógica do jogo, ou seja verifica se a bola colide com algum dos jogadores ou com alguma das paredes do jogo e alterar as variáveis que guardam o estado do jogo de acordo com a situação atual. Esta tarefa possui um atraso de 1 tick em relação ao inicio da execução do *kernel*.

A tarefa *updateBall* é a responsável por mover a bola, isto é, altera a posição da bola de acordo com a direção atual da mesma. Esta tarefa tem um atraso de 2 *ticks*, assim esta desfasada da tarefa *gameLogic* por um *tick*.

A tarefa *playersInput* é responsável por ler os inputs dos jogadores. Se o módulo Wi-Fi estiver conectado então verifica se recebeu alguma mensagem do modulo. Se tiver recebido, verifica o conteúdo da mensagem e consoante o conteúdo move o jogador correspondente para a direção especificada. O jogador 1 pode enviar dois caracteres para indicar a direção para a qual se pretende movimentar, ou o 'U' que significa que o jogador 1 se quer mover para baixo, ou o 'O' que significa que o jogador 1 se quer mover para cima. Da mesma forma, o jogador 2 pode enviar um de dois caracteres dependendo da direção do movimento, um 'L' significa que o jogador 2 se quer mover para baixo e um 'R' significa que o jogador 2 se quer movimentar para cima. Assim basta analisar o caracter que se recebe para se saber logo qual o jogador que se esta a movimentar e para onde. Esta tarefa tem um atraso de 3 *ticks*.

A tarefa *updateGame* é a responsável pela parte gráfica do jogo e por mostrar mensagens informativas aos jogadores. É esta tarefa que desenha os limites do jogo, os dois jogadores, a bola e a pontuação atual. Se o jogo estiver num estado de golo, então desenha também uma mensagem a informar os jogadores de que ocorreu um golo. Se algum dos jogadores tiver ganho o jogo então desenha uma mensagem a informar qual dos jogadores ganhou. Sempre que existem colisões entre a bola e um dos jogadores é emitido um som. Esta tarefa possui um atraso de 4 *ticks*.

IV. MEDIÇÃO DE PERFORMANCES

A. Métodos

Para análise de desempenho, decidiu-se verificar o uso do CPU na sua totalidade e por cada tarefa durante a execução prolongada do jogo. Para tal, no inicio de cada tarefa iniciou-se um contador e no final verificou-se quanto tempo essa tarefa

demorou a ser executada com a ajuda da função *micros()*.

B. Resultados

Chegou-se aos seguintes resultados: a tarefa *gameLogic* demora 220 µs; a tarefa *updateBall* demora apenas 12 µs; a tarefa *playersInput* demora 424 µs; e a tarefa *updateGame* demora 9120 µs a executar.

C. Escalonamento do sistema

Depois de conhecido o tempo de execução, período e *deadline* de cada tarefa, é então possível construir o gráfico de Gantt do sistema que se pretende desenvolver. Este é apresentado na figura 2.

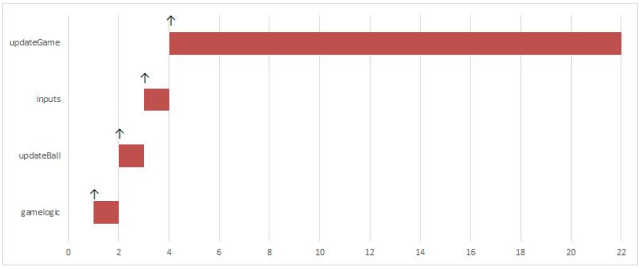

Fig. 2 Gráfico de Gantt do sistema.

Em média, o CPU do Arduíno é requerido 56.77% do seu tempo total; isto porque, as tarefas *gameLogic*, *updateBall*, *playersInput* e *updateGame* utilizam o CPU 42.92%, 2.35%, 82.81% e 98.66% respetivamente. Com isso pode-se afirmar que o sistema ainda pode ser acrescentado com mais tarefas sendo facilmente escalável pois o CPU encontra-se livre 43% do tempo.

V. MELHORIAS POSSÍVEIS

Uma possível melhoria que podia ser acrescentada ao sistema seria ao nível da lógica do jogo nomeadamente a adição de trigonometria com ângulos diferentes para a direção da bola quando esta toca na raquete de um jogador.

VI. CONCLUSÕES

Todos os objetivos propostos foram atingidos com sucesso. Ao longo do projeto o grupo foi ganhando não só conhecimentos em programação para Arduíno como em programação de sistemas em tempo real.

REFERENCIAS

[1] Atmel Corporation ATmega48A/PA/88A/PA/168A/PA/328/P Complete. (2013, Fevereiro) [Online]. Disponivel: http://www.atmel.com/Images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P datasheet Complete.pdf.

[2] Philips Semiconductors, PCD8544 - 48 × 84 pixels matrix LCD controller/driver. (1999, Abril) [Online]. Disponivel: http://eia.udg.edu/~forest/PCD8544_1.pdf.

[3] Espressif Systems, ESP8266EX Datasheet. (2015, Junho) [Online]. Disponivel: https://www.adafruit.com/images/product-files/2471/0A-ESP8266__Datasheet__EN_v4.3.pdf.

[4] Espressif Systems, ESP8266 AT Instruction Set. (2015, julho) [Online]. Disponivel: https://cdn.sparkfun.com/assets/learn_tutorials/4/0/3/4A-ESP8266__AT_Instruction_Set__EN_v0.30.pdf.