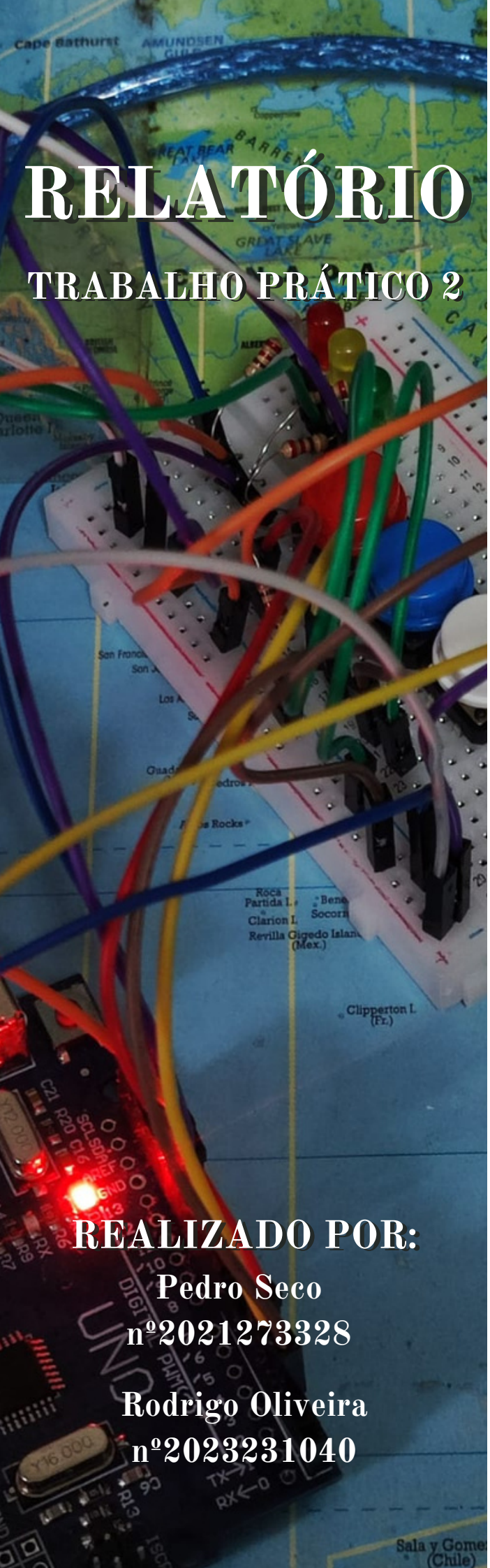

RELATÓRIO

TRABALHO PRÁTICO 2

REALIZADO POR:

Pedro Seco
nº2021273328

Rodrigo Oliveira
nº2023231040

1 2 9 0

UNIVERSIDADE D
COIMBRA

LICENCIATURA EM ENGENHARIA INFORMÁTICA

“Ardugotchi”

TECNOLOGIA DA INFORMÁTICA
PL4

DEZEMBRO DE 2023

Introdução

No âmbito da disciplina de Tecnologia da Informática, foi-nos desafiado a elaborar o segundo trabalho prático. Neste trabalho, o objetivo consistiu em implementar um Tamagotchi usando um arduino. Este tem 3 botões, que são usados para que o utilizador possa atender às necessidades do Tamagotchi (comer, dormir e brincar), assim como 3 LEDs que indicam quais as necessidades do mesmo, e por fim um LDR para a recolha de dados da luz ambiente.

Assim, este relatório serve para explicar de uma maneira mais clara e pormenorizada o código elaborado e referir os maiores obstáculos ultrapassados na elaboração do mesmo.

Desenvolvimento

Como foi referido acima, neste trabalho foi utilizado um arduino. No circuito do arduino utilizamos a *breadboard* para dispor os 3 LEDs, os 3 botões para corresponder às necessidades do Tamagotchi, resistências e fios de ligação. Iremos agora explicar a elaboração do código.

Começámos por declarar as variáveis utilizadas. 'LDR' define o pino onde o LDR está conectado; 'ultimaVezComer', 'ultimaVezAtencao', 'ultimaVezDormir', 'ultimaMedicaoLuz', 'ultimaVezLEDcomer', 'ultimaVezLEDdormir', 'ultimaVezLEDatencao' e 'ultimaImpressaoPenalizacoes' guardam o tempo em milissegundos da última vez que o Tamagotchi comeu, brincou, dormiu, registou a luminosidade ambiente, guardou o instante em que cada LED acendeu e a última vez em que foi guardado a impressão para a visualização das penalizações, respetivamente. As constantes criadas foram as seguintes: 'tempoParaComer', 'tempoParaAtencao', 'tempoParaIrDormir', 'tempoAdormir', 'tempo15sPenalizacao', 'tempo60sPenalizacao', intervaloImpressaoPenalizacoes', que correspondem respetivamente ao tempo necessário para o Tamagotchi ter fome, para precisar de atenção, ter sono, para o tempo que o Tamagotchi passa a dormir, tempo para o utilizador ser bonificado, tempo para o utilizador ser penalizado e o tempo para imprimir as penalizações do utilizador. Para se tratar do debounce foram criadas as variáveis 'estadoBotaoComer', 'estadoBotaoAtencao' e 'estadoBotaoDormir' que guardam o estado atual dos botões; 'ultimoEstadoBotaoComer', 'ultimoEstadoBotaoAtencao', 'ultimoEstadoBotaoDormir' que guardam os últimos estados dos botões; e 'debounceDelay' que é um valor predefinido (50 milissegundos) para evitar leituras falsas no botão. O bool 'acordado' indica quando o Tamagotchi está acordado (acordado = true) ou a dormir (acordado = falso); 'tempoAdormirInicio' guarda o instante em que o Tamagotchi começou a dormir. As variáveis "comerAceso", "atencaoAceso" e "dormirAceso" controlam se os LEDs para comer, para atenção e para dormir estão acesos ou não, de modo a poder guardar o instante em que os mesmos acenderam. "JogoON" é uma variável criada para controlar se o jogo está a decorrer ou não. "TempoAdormirInicio" guarda o instante de tempo em que o Tamagotchi começou a dormir. "penalizacoes" é uma variável que foi criada para ir acumulando os pontos das penalizações. "numAmostra" corresponde ao número de amostras para calcular a média da leitor de sensor de luz. "amostrasLuz[6]" corresponde a um vetor (array) que vai armazenando as amostras do sensor de luz (LDR). "indiceAmostras" corresponde ao índice para saber a posição atual no vetor de amostras (começa em 0 e vai incrementando). Por fim, "media" guarda a média das amostras de luz e "gamaADC" corresponde ao valor máximo do conversor analógico-digital (ADC).

Passado agora ao funcionamento do código, no "void setup()", é iniciada a comunicação *Serial* através do "Serial.begin(9600)". Através do primeiro ciclo for são definidas as configurações dos pinos 11 a 13 como outputs, aos quais estão conectados os LEDs que representam as necessidades do Tamagotchi; No segundo ciclo for, definimos os pinos 2 a 4 como entradas pull-up, sendo que estes pinos estão conectados aos botões usados para poder atender as necessidades do Tamagotchi. Além disso, também definimos o LDR como input. Por fim, utilizamos o randomSeed(analogRead(1)) para inicializar a geração de números aleatórios que serão utilizados como margens de tempo aleatórias no decorrer do código.

Iremos agora explicar o resto do código. Começando pelas funções, a função "gerirPenalizacoes()" é responsável por controlar as penalizações do "Tamagotchi", ajustando-as com base em certos critérios de tempo e ações do jogador. Esta função recebe como parâmetros o tempo atual (definido no void loop), o instante de tempo da última ação (quando o LED fica aceso) e diferentes tipos de penalização (15 segundos - "tempo15sPenalizacao" e 60 segundos - "tempo60sPenalizacao"). Se o intervalo de tempo entre o tempo atual e o instante em que o LED acendeu for menor que 15s, as penalizações são reduzidas em 5 pontos, com um mínimo de 0. Se o jogador demorar mais de 60s, aumenta as penalizações em 5 pontos por cada minuto demorado. Esta função irá ser chamada posteriormente em cada uma das funcionalidades do Tamagotchi

(comer, dormir e brincar) quando o jogador clicar no botão, para verificar o tempo que o jogador demorou a satisfazer as necessidades do Tamagotchi. A função “verificaLuz()” aceita como argumento o tempo atual e esta função analisa a luminosidade do ambiente através de um sensor de luz(LDR) conectado ao pino A0. Primeiro, verificamos se passou um minuto desde a última medição de luz, e se sim, a variável “ultimaMedicaoLuz” é atualizada para o tempo atual de modo a controlar o intervalo entre as medições. É realizada a leitura do LDR e o valor é armazenado no vetor “amostrasLuz” na posição determinada pelo “indiceAmostra”. Depois, o índice da amostra é incrementado para a próxima posição no vetor de modo a permitir que a próxima leitura seja guardada. Quando o índice da amostra for 6, ou seja, quando passarem 6 minutos será feita a média dos resultados dos últimos 6 minutos, ou seja, dos 6 resultados captados pelo LDR. Depois de calcular a média, o índice da amostra é igualado a 0, de modo a guardar novas leituras para os próximos 6 minutos. Passando agora às funções “verificaComer()” e “verificaAtencao()”. Iremos apenas explicar a função “verificaComer()” , pois a função “verificaAtencao()” funciona de forma idêntica. Tanto uma como a outra aceitam como argumento o tempo atual. Na função “verificaComer()” começamos por verificar se o tempo estabelecido para alimentar o Tamagotchi passou, acrescentando um tempo aleatório entre -60000 e 60000 milissegundos. Se a condição for verdadeira, o LED verde irá acender. E se tal acontecer, o tempo da última vez que o LED foi aceso (“ultimaVezLEDcomer”) é atualizado para o tempo atual, de forma a poder guardar esse instante. Depois, garantimos que passamos a variável “comerAceso” para false, de forma a guardar apenas o instante exato em que o LED acendeu. De seguida, utilizamos o Debounce, e quando o botão de comer for apertado, ou seja, quando o jogador satisfizer a necessidade do Tamagotchi, o LED irá ser apagado e atualizamos o tempo da última vez em que o botão foi pressionado (“ultimaVezComer = millis()”). De seguida, chamamos a função que falamos anteriormente, “gerirPenalizacoes()” para contarmos as penalizações com base no tempo atual, no último instante em que o LED acendeu e nos tempos estabelecidos para as penalizações. Por fim, passamos a variável “comerAceso” para verdadeiro, de modo a podermos guardar o próximo exato instante em que o LED irá acender. A função “verificaAtencao()” funciona de modo igual, apenas com outros nomes de variáveis. Para terminar as funções, falta referir a função “verificaDormir()”. A parte inicial desta função é idêntica à das funções “verificaComer()” e “verificaAtencao()”, com exceção do facto de o LED poder acender se o tempo predefinido passar ou se a média dos valores recolhidos no LDR for superior a ⅔ da gama do ADC. Além disso, o que difere também esta função é quando o botão para satisfazer a necessidade de dormir é pressionado. Quando este é pressionado, é guardado e atualizado o instante em que o botão foi pressionado (“ultimaVezDormir = millis()”), os LEDs ficam todos apagados, a variável “acordado” fica false (já iremos explicar o papel importante desta variável), a variável “dormirAceso” fica verdadeira de modo a podermos guardar o próximo instante em que o LED irá acender e chamamos a função “gerirPenalizacoes()” para verificar quais foram as penalizações do jogador.

No void loop(), guardamos o tempo atual numa variável (“tempoAtual”) e verificamos as leituras do LDR, chamando a função “verificaLuz()” independentemente do jogo estar ON ou não. Se o jogo estiver ON, ou seja, se a variável “JogoON” estiver verdadeira, e se o Tamagotchi estiver acordado, iremos chamar as funções “verificaComer()”, “verificaAtencao()” e “verificaDormir()”. Caso não esteja acordado, verificamos o intervalo de tempo entre o tempo atual e o instante em que começou a dormir. Se esse intervalo de tempo for superior ou igual ao tempo configurado para estar a dormir (“tempoAdormir”) então as variáveis que guardam os instantes em que os botões foram pressionados, são atualizadas para o tempo atual de modo a “resetar” as variáveis. Além disso, as variáveis “comerAceso”, “atencaoAceso” e “dormirAceso” passam a ser verdadeiras de modo a que depois possa ser guardado o instante em que o LED irá acender e a variável “acordado” passa a ser verdadeira, pois já passou o tempo de dormir do Tamagotchi. De seguida, como queremos que as penalizações sejam impressas a cada minuto, se o intervalo de tempo menos o instante da última impressão for igual ou superior ao tempo anteriormente definido (1 minuto), então é impresso no Serial Monitor as penalizações. Se as penalizações forem superiores a 25, então será impresso no Serial Monitor que o Tamagotchi morreu e a variável “JogoON” passa a ser falsa. Como esta variável passa a ser falsa, então o Tamagotchi irá apenas desligar todos os LEDs e não irá fazer mais nada, dando como terminado o jogo.

Para concluir, este trabalho foi essencial para a melhor compreensão de toda a matéria que temos vindo a dar nas aulas desde o início do ano. Ao longo do trabalho, fomos tendo alguns obstáculos, como por exemplo, as penalizações, mas com o esforço e com alguns conselhos do professor, foi possível ultrapassá-los. Sentimos que já estamos bastante confortáveis a trabalhar com o Arduíno.

NOTAS: Vídeo foi acelerado em 2x, e todas as variáveis do código relativas a tempo foram divididas por 4.