**Relatório do Trabalho Prático 2**

Universidade de Coimbra

Departamento de Informática

Licenciatura de Engenharia Informática

**Diogo André Natário Fermino de Pina** | 2023212806

**Guilherme Teixeira Gonçalves Rosmaninho** | 2022257636

10 de Dezembro de 2023

Í N D I C E

1 – **Introdução**

2 – **Desenvolvimento**

3 – **Problemas**

3 – **Conclusão**

Uma imagem com eletrónica, Engenharia eletrónica, Fios elétricos, Componente de circuito

Descrição gerada automaticamenteUma imagem com cabo, Fios elétricos, eletrónica, Engenharia eletrónica

Descrição gerada automaticamenteUma imagem com eletrónica, Fios elétricos, cabo, Engenharia eletrónica

Descrição gerada automaticamente

**// Introdução**

O Trabalho Prático 2 tem como objetivo a criação de uma “mascote digital”, como foi o Tamagotchi, para gerações mais antigas, ou como foi o Furby para os que nasceram já no século XX. Esta mascote foi feita através do microcontrolador Arduino, e vamos dar o nome de Ardugotchi a ela. O objetivo é atender às necessidades da mascote, como dar comida, brincar e dormir, e não a deixar morrer.

**// Desenvolvimento**

Ao iniciar o Ardugotchi, ele é dado como acabado de nascer, logo, os valores de fome, comer e dormir são definidos a 0. A partir deste momento, quando passarem 4 minutos (mais ou menos 60 segundos), a luz de fome (Led Verde) acende, indicando ao utilizador que vai ter de clicar no botão para o Ardugotchi comer. A implementação do “4 minutos (mais ou menos 60 segundos)” foi feita através da criação de variável (exemplo: random\_num\_comer), dando o valor a esta de “random(0,121)/60.00”, que vai devolver um valor entre 0 e 120, que vai ser depois convertido em minutos, fazendo depois a soma desta variável com o valor predefinido menos 1 (valor da fome = 4, logo 4-1=3; este 3 vai ser somado ao valor da variável random\_num\_comer). Para devolver valores realmente aleatórios, tivemos de adicionar “randomSeed(analogRead(0))” no Setup() (algo que já tínhamos reparado que era necessário no último Trabalho Prático).

Também é utilizado um sensor de luz LDR, um componente cuja resistência varia de acordo com a intensidade da luz – quanta maior luminosidade, menor é o valor devolvido, o qual varia entre 0 e 1023 (gama ADC – conversor analógico para digital). A cada minuto de vida (enquanto acordado), é guardado o valor devolvido pelo LDR num vetor que, todos os valores medidos forem diferentes de zero (para ter a certeza que o programa so verifica a média quando todos os 6 valores forem medidos) e a média dos últimos 6 valores medidos for superior a dois terços da gama do ADC, o Ardugotchi vai ficar com sono, pedindo assim que o utilizador o deixe dormir.

Enquanto o Ardugotchi está a dormir, apenas o led de dormir está aceso, num fade, que foi realizado através de uma diferença de tempo, utilizando o micros(), assim evitando usar qualquer tipo de delay() que iria afetar o resto do código. Também, quando o Ardugotchi volta a acordar, os valores de fome e de brincar não foram afetados (adicionámos o tempo que o Ardugotchi esteve a dormir ao tempo que o Ardugotchi comeu e brincou da última vez).

Cada vez que passam 1 minuto desde que o Ardugotchi começou a sinalizar algum problema, e este problema não foi resolvido, são adicionados 5 pontos a um contador, que ao alcançar um valor superior a 25, vai causar a morte à mascote, algo que é sinalizado ao utilizador quando todos os leds piscarem de segundo em segundo. Mas por outro lado, se o problema for solucionado em menos de 15 segundos, e o contador tiver pelo menos 5 pontos, serão retirados 5 pontos a este. Quando o Ardugotchi morrer, o utilizador tem a opção de pressionar nos três botões ao mesmo tempo durante mais ou menos 2 segundos, realizando assim um tipo de Ressuscitação Cardiopulmonar (mais conhecido como CPR na sua sigla inglesa), que irá fazer uma “onda” nos leds e irá recomeçar o jogo.

O debouncing dos botões foi feito através do uso de millis(), onde foi definido o intervalo entre o último e o novo cliques como 200 milissegundos (valor que após testes pareceu mais adequado à finalidade).

**// Dificuldades**

Ao contrário do último trabalho prático, e pelo treino, já não tivemos tantos problemas e erros básicos, como foi esquecer o “;” no final de cada linha. Mas tivemos alguns outros problemas:

- No início tivémos problemas a tentar implementar uma forma para que os resultados do sensor LDR fossem guarados num array, e que estes valores fossem apagados quando não fossem mais precisos (quando houvesse mais que 6 valores guardados);

- Foi necessário muito “trial and error” para fazer uma forma que o millis() da fome e de brincar não alterassem enquanto o Ardugotchi dormir, acabando por no final adicionando simplesmente o tempo que o Ardugotchi esteve a dormir ao tempo que a mascote comeu/brincou pela última vez.

**// Conclusão**

Para concluir este relatório, achamos importante realçar que foi um projeto que nos cativou imenso a realizar, e que ao ver o resultado, estamos muito contentes e satisfeitos. Sempre é bom ver visualmente e fisicamente o produto do nosso trabalho em código.

**1 MINUTO DE VIDA NO VIDEO = 1/3 MINUTOS**