Redução da Bradicinesia em Pacientes com Mal de Parkinson em Estado Inicial

Davi Antônio da Silva Santos
Graduando em Engenharia Eletrônica
Universidade de Brasília
Gama, Brasil
Email: wokep.ma.wavid@gmail.com

Victor Aguiar Coutinho
Graduando em Engenharia Eletrônica
Universidade de Brasília
Gama, Brasil
Email: victor.a.coutinho@gmail.com

***Resumo*—O uso repetitivo das mãos com auxílio audiovisual tem resultados positivos em pessoas diagnosticadas com Mal de Parkinson em estado inicial. Tem-se como objetivo apresentar um sistema a base de MSP430 fácil de utilizar que execute jogos interativos. Quem utilizá-lo terá redução da bradicinesia, principal sintoma da doença.**

***Keywords*—Mal de Parkinson; tratamento; exercício das mãos; MSP430.**

I. Introdução

A. Justificativa

O uso de exercícios simples com as mãos, repetição de sequências acompanhados de estímulos audiovisuais pode ser usado em conjunto com os tratamentos tradicionais para auxiliar na reabilitação dos pacientes acometidos do mal de Parkinson em estado inicial, atenuando a bradicinesia, um dos principais sintomas da doença [2], caracterizado pela extrema lentidão dos movimentos [1].

Assim, baseado no equipamento utilizado no artigo de Elisa Pelosin, propõe-se uma solução embarcada de baixo custo baseada no microcontrolador MSP430, utilizando-se a respectiva placa de desenvolvimento, a LaunchPad.

B. Objetivos

Exercício tem um forte impacto na recuperação de pacientes diagnosticados com Mal de Parkinson, porém um dos maiores problemas é a falta de interesses dos pacientes em exercícios físicos [3]. Como Diane Playford explicita em seu artigo Exercise and Parkinson's disease, o desafio não é definir programas de atividades saudáveis, mas incentivar pessoas a encontrarem atividades em que elas achem proveitosas.

Tem-se como objetivo apresentar um projeto de dispositivo que incentive aos pacientes diagnosticados com o Mal de Parkinson em estado inicial a exercitar as mãos como parte de tratamento para não perder o controle dos movimentos.

Com isso, almeja-se colocar jogos interativos de forma a interessar os pacientes a manterem o tratamento. Jogos que serão feitos a base de pesquisa em tratamentos de pacientes de Mal de Parkinson.

C. Requisitos

Com base no público alvo, pacientes diagnosticados com Parkinson em estado inicial, e o objetivo do sistema, melhorar a coordenação motora ao atenuar a bradicinesia, foram elencados os seguintes requisitos:

- O sistema proposto deve ser de fácil construção e baixo custo;
- O sistema deve ser fácil de operar;
- O sistema deve fornecer estímulos audiovisuais;
- A interface deve ser de fácil compreensão;
- Sequências geradas devem ser aleatórias, evitando que o paciente decore-as.

D. Benefícios

O sistema embarcado proposto, conforme exposto anteriormente, reduz a bradicinesia através da repetição de sequências reforçadas por estímulos audiovisuais, em conjunto com os tratamentos tradicionais, em pacientes diagnosticados com Parkinson em estado inicial.

II. Descrição do sistema

A proposta é apresentar um dispositivo que auxilia na redução da bradicinesia e que, de alguma forma, incentive quem a possui a exercitar as mãos. Para isso, o dispositivo possui um jogo que soa sinais com frequências diferentes e o usuário deva apertar os botões correspondentes. Importante ressaltar que o usuário deve seguir o protocolo de uso para melhores resultados.

Na inicialização do sistema, são emitidos quatro estímulos sonoros, correspondentes às frequências que serão utilizadas durante o jogo. Em seguida, é tocada sequência, a qual inicia em um elemento, e acende-se o LED verde enquanto o número de elementos da sequência não for digitado nos botões, indicando que o sistema está esperando uma entrada do usuário. Ao terminar a sequência, o LED verde desliga.

Caso o usuário acerte a sequência, o LED verde pisca, indicando o acerto, e o nível é incrementado, isto é, a próxima sequência aumentará em um elemento, colocado ao fim da sequência anterior. Também, em caso de acerto da sequência, o sistema esperará uma entrada do usuário indicando se ele quer

continuar o jogo, função atribuída ao botão S1, ou se ele quer desistir do jogo, função do botão S2. Esse momento é expresso ao usuário com ambos os LEDs ligados. Caso o usuário queira continuar, o sistema mostrará a próxima sequência, e caso ele queira desistir, o jogo desligará e exigirá um *reset* para reiniciar.

Caso o usuário erre, o LED vermelho piscará indicando um erro na digitação e, em seguida, os dois LEDs acenderão, esperando que o usuário informe que quer continuar jogando, intenção atribuída ao botão S1, ou que quer desistir do jogo, atribuída a S2. Caso o usuário deseje continuar, o jogo reiniciará, mas caso desista, o jogo desligará e será necessário um *reset* para reiniciar.

III. Descrição do Hardware

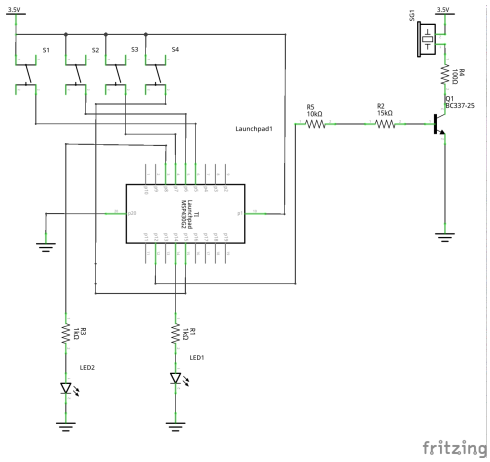

Figura 1. Esquemático do dispositivo proposto

O projeto é composto pela placa de desenvolvimento Launchpad, a qual contém um microcontrolador MSP430G2553; quatro *pushbuttons* para as entradas do usuário, *S1*, *S2*, *S3*, *S4*, conectados às entradas *P1.3*, *P1.4*, *P1.5* e *P1.7*, respectivamente; dois LEDs, sendo um verde, conectado à *P2.1*(para o código no Energia a porta utilizada havia sido o *P1.6*, porém para facilitar no código em liguagem C, optou-se deixar ambos os LEDs na porta 2) e outro vermelho, ligado à /emphP2.0; um *buzzer* eletromagnético; um transistor NPN BC337-25, conectado à *P2.4*; e resistores de 10 kΩ, 15 kΩ, 1 kΩ e 100 Ω para limitar a corrente fornecida pelas portas do MSP.

Os quatro *pushbuttons*, estão conectados diretamente ao Vcc = 3,5 V . Utilizou-se esta configuração pois no código foram habilitados os resistores internos de *pull-down* no microcontrolador.

Os resistores colocados sobre os LEDs limitam a corrente entregue pelas portas digitais do MSP, que não deve ser maior que 6 mA em módulo, segundo [6]. Já os resistores colocados antes do transistor limitam a corrente entregue pela porta digital do MSP e da fonte de 3,5 V ao *buzzer*.

Os resistores conectados ao transistor foram dimensionados segundo as especificações do MSP, do *buzzer* e do próprio transistor. Usando um multímetro digital, determinou-se que o *hfe* do transistor era de 308, o que estava dentro dos limites estabelecidos no *datasheet* do mesmo, entre 160 e 400 [4].

O *datasheet* do *buzzer* indicava uma corrente máxima de 40 mA [5], que no transistor corresponderia à corrente no coletor, e este dado foi usado para se calcular a corrente na base do transistor, 129,870 μA. A resistência na base foi dimensionada através da equação Rb = (Vcc - Vbe)/Ib. Considerou-se Vbe = 0,7 V e Vcc = 3,5 V, e obteve-se uma resistência de cerca de 21 kΩ, a qual foi substituída por uma de 25 kΩ.

IV. Descrição do Software

A. Software no Energia

A plataforma de prototipagem utilizada para codificar, compilar e enviar para o MSP430 foi o Energia 1.6.10E18. No código primeiramente, definiu variáveis para valores constantes e portas. A *TSMMS* é uma constante que representa o tempo que o *buzzer* fica ligado ou desligado pois a taxa entre sons que apresenta melhores resultados é de 3 Hz [1]. A *LEDverde* representa o pino em que o LED vermelho está conectado, no caso o P1.6 do MSP430. Já a constante *LEDvermelho* representa o pino P2.0 na qual o LED vermelho está conectado. A *PPWM* é o pino onde o *buzzer* está conectado.

O software foi dividido em funções: *ler_portas*, *toca_buzzer*, *pisca2x*, *setup* e *loop*. A primeira função no código é a *ler_portas*. Essa tem como saída o valor do botão no qual foi apertado, sendo o botão S1 representando o sinal com mais baixa frequência e o S4 o sinal com mais alta frequência. A função começa com um *while(1)* pois ele só deve sair do laço quando algum botão for apertado e os demais não. Considerou os botões configurados como emphpull down. Para o usuário saber que que o botão foi apertado o LED verde, como pode ser visto na função principal *loop*, que anteriormente estava ligado desliga. O *delay* de 100 milissegundos para dar tempo do usuário soltar o botão.

A função *toca_buzzer* tem como entrada *nLED* e dentro da função tem o vetor com frequências já definidas. É tocado o *buzzer* na frequência que está na posição *nLED -1* do vetor *frequencias*. Após tocado por *TSOMMS* ms o buzzer desliga com a função *noTone* por *TSOMMS* ms.

Para indicar que uma pessoa acertou ou errou um LED pisca duas vezes, sendo que se o usuário acertou o LED verde que pisca, caso contrário o LED vermelho que pisca. Para isso, desenvolveu a função *pisca2x* tendo como entrada o LED desejado *LED_que_pisca*. Os *delays* de 100 ms foi definido para que o usuário perceba que piscou e não confunda com, no caso do LED verde, o momento para digitar. O *delay* de 1000 ms é para o usuário perceber a parte de piscar indicando acerto ou erro e a parte que liga os LEDs para o usuário decidir se quer continuar ou não, essas duas condições será explicada posteriormente.

É definido as portas de entrada e saídas na função *setup*, padrão no Energia. Definiu os pinos P1.3, P1.4, P1.5 e P1.7 como entrada e *pull down* para leitura dos botões S1, S2, S3 e S4, respectivamente. Definiu os pinos P2.4 (representando o buzzer), *LEDvermelho* e emphLEDverde como saída. Após essas definições colocou todas as saídas em nível lógico alto e gera um número aleatório a partir da leitura da temperatura do ambiente no *A10*. Por fim habilita a entrada e saída serial da placa.

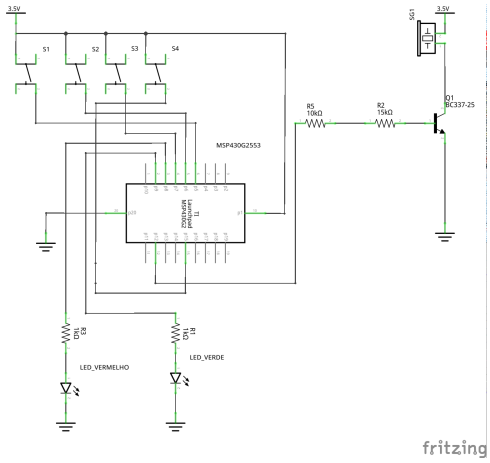

Figura 2. Novo esquemático do dispositivo proposto

A função principal é *loop*. Começa zerando todas as variáveis, exceto o *nivel* já que o primeiro nível considerado foi o zero. Ele apresenta todas as frequências no vetor na ordem dos botões que o representam. Dentro do *while(1)* está o jogo em si. A variavel*continuar_jogo* igual a zero significa que o jogo pode continuar, essa condição é avaliada no final do jogo. Considerou o nível máximo de 32, caso o nível seja esse então o jogo finaliza.

A posição do vetor de frequências é gerada por meio da função *random* e colocada na posição *i* da sequência de posições. Logo após é tocado as frequências desde primeira até a atual gerada com a função *toca_buzzer*. É feito a leitura dos botões com a função *ler_portas()* e decide se o usuário acertou a sequência comparando os vetores *seq_gen* (sequência gerada) e *seq_read* (sequência dos botões).

Se a sequência digitada não for igual ao apresentada então a variável *fail* fica igual a um. Com o fail igual a um, o LED vermelho pisca duas vezes indicando o erro e o nível retorna para um. Caso contrário, ou seja acertou, o LED verde pisca duas vez para indicar o acerto o nível aumenta e a variável *i* também aumenta para quando retornar o *while* a nova posição do vetor das frequências ser guardada na próxima posição do vetor *seq_gen*.

Como a decisão de se o usuário acertou ou não, chega na etapa de decidir se o mesmo deseja continuar a jogar. Para isso, ambos os LEDs ficam ligados e é feito a leitura dos botões, essa leitura é armazenada na variável *continuar_jogo*. Se apertou S1 significa que deseja continuar e o S2 para finalizar o jogo. Se for continuar, os LEDs desligam e o LED verde pisca duas vezes indicando o desejo do usuário. Se não deseja continuar, o LED vermelho pisca duas vezes indicando o desejo de não continuar e o programa entra em um laço infinito (a linha do *while(1);*).

B. Portando o Código do Energia para C e Assembly

Notou-se que no esquemático original do projeto os LEDs ficavam muito distantes no mapeamento dos pinos, então o LED verde foi movido para o pino perto do LED vermelho, resultando em LED verde em P2.1 e LED vermelho em P2.0, de acordo com o novo esquemático 2.

Para converter o código usado no Energia para a linguagem C teve que substituir algumas funções e criar outras. Como, por exemplo, no lugar de *digitalRead()* usou *P1IN* ou *P2IN* para a leituras das portas P1 e P2, respectivamente. Uma função criada foi a função *delay*, pois no energia já tinha ela na biblioteca, mas em C houve a necessidade de criar uma utilizando os *timers*.

Uma das maiores dificuldades foi a elaboração de um código em C para o MSP430 que substituísse a função *Tone* e *noTone*. Para elas, usou o canal de comparação 1. Ao fazer no modo comum, ou seja, para ativar o som colocar o TA1CCR2 igual a metade de TA1CCR0 e para desativar colocar TA1CCR0 e TA1CCR1 igual a 0. Porém houve vários problemas como: um sinal com frequência alta no momento em que deveria estar desligado, frequências diferentes das desejadas e algumas distorções no sinal de som.

Para resolver esse problema, colocou no início do código da função *toca_buzzer* comandos para ativar o canal de comparação e no final dessa função colocou comandos para desativar o canal de comparação. No *switch* ele altera o período do sinal no *buzzer*. No fim são zerados os registradores TA1CTL, TA1CCR0 e TA1CCR2. Foi adicionado uma interrupção na parte em que o usuário deve decidir se vai continuar o jogo.

A função que gerava números aleatórios disponibilizada pelo Energia teve de ser substituída por um equivalente em C. Para isso, foram elaboradas duas funções que trabalham para gerar os números aleatórios, a *configura_lcg*, que inicializa a semente do gerador a partir da leitura do sensor de temperatura interno do MSP430G2553 feita através do ADC10, o conversor analógico digital de 10 bits presente no mesmo; e *lcg*, o próprio gerador, que retorna um número aleatório baseado na semente toda vez que é chamada.

Para que o código não ficasse poluído, foi criada uma função responsável por piscar os LEDs duas vezes, *pisca2x*, ação que é repetida diversas vezes durante o código. Esta função apenas utiliza a função *delay* implementada para controlar o tempo de nível alto e nível baixo do LED que for passado para a função.

O código convertido para a linguagem C acrescenta interrupções na porta P1 e modos de baixo consumo para substituir os momentos em que o *polling*, momentos em que a CPU fica ocupada checando repetidamente se houve uma mudança em uma variável, era desnecessário; e os laços infinitos que encerravam o programa caso o nível fosse ultrapassado ou caso o usuário desistisse de continuar jogando.

O processador é desligado, isto é, entra em modo de baixo consumo zero (LPM0), sempre que o programa checa se o usuário gostaria de continuar jogando. O processador é acordado por uma interrupção nos botões responsáveis por confirmar ou negar a opção de continuar o jogo. Já nas situações em que o jogador desiste de continuar o jogo ou o nível máximo (32) é atingido, todos os sinais de *clock* do microcontrolador são desligados, pois este é colocado no modo de baixo consumo 4 (LPM4) e as interrupções são desabilitadas, fazendo com que a única maneira de reiniciar o sistema seja um *reset*.

Para evitar interrupções espúrias, a interrupção é desativada

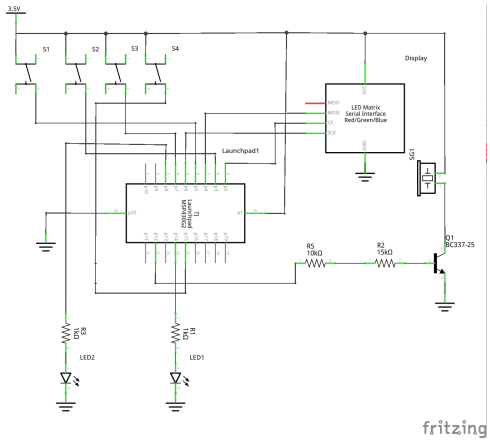

Figura 3. Esquemático do dispositivo proposto no quarto ponto de controle

e ativada nos momentos oportunos, isto é, desativada no momento em que o novo número aleatório é gerado e é ativada somente após o sistema validar a entrada do usuário e solicitar se ele gostaria de continuar ou não o jogo, entrando em modo de baixo consumo, na função que substitui o laço principal do jogo, *jogo01*.

C. *Mudanças realizadas no Quarto Ponto de Controle*

Após testes realizados com o software e o hardware desenvolvidos no terceiro ponto de controle, notou-se que a experiência do usuário seria melhorada com uma adição de uma interfaxe gráfica. Com base nisso, adicionou-se uma matriz 8x8 de LEDs vermelhos conectada a um driver, o CI MAX7219, o qual se comunica pelo protocolo SPI.

A adição desta matriz tornou necessária a elaboração de uma biblioteca específica, responsável por implementar a inicialização do protocolo SPI, no qual o MSP430 seria o mestre e o MAX7219 o escravo; a inicialização dos registros de controle do CI utilizado, conforme as especificações do *datasheet*, [7]; e as funções para mostrar frases e caracteres de diferentes formas na matriz. Também foi necessário implementar a tabela ASCII para que fosse possível a fácil representação dos caracteres no display, sendo esta implementada a partir de um vetor de 2048 elementos do tipo *char*, os quais de oito em oito formam a fonte necessária à esta implementação.

O protocolo SPI usa dois pinos fixos no MSP, P1.2 para o SIMO (entrada do escravo e saídaa do mestre) e o P1.4, responsável pelo sinal de *clock*. O CI MAX7219 ainda necessita de um pino de *LOAD* para configurar quando o conteúdo armazenado no seu registrador de deslocamento deve ser executado. Esse pino foi configurado como P1.0, um pino de entrada e saída comum do MSP430. Devido a essas mudanças, as entradas dos botões foram alteradas, sendo os botões S1, S2, S3 e S4 mapeados para os respectivos pinos: P1.3, P1.1, P1.5 e P1.7, conforme a figura 3.

REFERÊNCIAS

[1] PELOSIN, E. et al., Reduction of Bradykinesia of Finger Movements by a Single Session of Action Observation in Parkinson Disease. *Neurorehabilitation and Neural Repair*. p.552-560, 2013.

[2] Medtronic Brasil. *Sobre a Doença de Parkinson*. Disponível em: <http://www.medtronicbrasil.com.br/your-health/parkinsons-disease/index.htm>. Acesso em 03 de setembro de 2017.

[3] PLAYFORD, Diane. Exercise and Parkinson's disease. *Neurol Neurosurg Psychiatry*, 2011;82:1185.

[4] FAIRCHILD SEMICONDUCTOR CORPORATION. *BC337-25 NPN General Purpose Amplifier*. 1997.

[5] *KX-1200 Series Magnetic Transducer*

[6] TEXAS INSTRUMENTS. *MSP430G2x53, MSP430G2x13 Mixed Signal Microcontroller*. p.70, 2011.

[7] MAXIM INTEGRATED. *MAX7219/MAX7221 Serially Interfaced, 8-digit LED Display Drivers*. p.17, 2003.