Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Engenharia Elétrica

Desenvolvimento de um estimulador visual com LEDs para sistemas BCI-SSVEP

Breno Alves Navarro

Trabalho de Conclusão de Curso

João Monlevade, MG

2018

Breno Alves Navarro

Desenvolvimento de um estimulador visual com LEDs para sistemas BCI-SSVEP

Orientadora: Prof.ª Dr.ª Sarah Negreiros de Carvalho Leite

Coorientador: Prof. Dr. Marcelo Moreira Tiago

Trabalho de conclusão de curso apresentado à Universidade Federal de Ouro Preto como parte dos requisitos exigidos para obtenção do título de Bacharel em Engenharia Elétrica pelo Instituto de Ciências Exatas e Aplicadas.

Universidade Federal de Ouro Preto
João Monlevade, MG
2018

N322d Navarro, Breno Alves.
Desenvolvimento de um estimulador visual com LEDs para sistemas BCI-SSVEP [manuscrito] / Breno Alves Navarro. - 2018.

58f.: il.: color; grafs; tabs.

Orientadora: Profª. Drª. Sarah Negreiros de Carvalho Leite.
Coorientador: Prof. Dr. Marcelo Moreira Tiago.

Monografia (Graduação). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Departamento de Engenharia Elétrica.

1. Engenharia elétrica. 2. Interface cérebro-computador. 3. Potencial evocado (Eletrofisiologia) . I. Leite, Sarah Negreiros de Carvalho. II. Tiago, Marcelo Moreira. III. Universidade Federal de Ouro Preto. IV. Titulo.

CDU: 004.5

Catalogação: ficha.sisbin@ufop.edu.br

MINISTÉRIO DA EDUCAÇÃO
Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Aplicadas
Colegiado do Curso de Engenharia de Elétrica

UFOP

AUTORIZACAO DO COLEGIADO PARA LANCAMENTO DA NOTA ATV600 NO HISTÓRICO ESCOLAR EM SUBSTITUIÇÃO À ATA DE DEFESA

Autorização

O colegiado do curso de engenharia elétrica autoriza à orientadora Sarah Negreiros de Carvalho Leite o lançamento da nota do Trabalho de Conclusão de Curso - ATV 600 do aluno Breno Alves Navarro em virtude do aproveitamento do artigo publicado em evento cientifico ou periódico, conforme previsto no § 3º do Art 5º da resolução COEE nº 011/2018.

Nota atribuída pela orientadora: 9,0.

João Monlevade, 17 de dezembro de 2018.

Prof. Dr. Márcio Feliciano Braga
Presidente do Colegiado de Engenharia Elétrica

Agradecimentos

Primeiramente gostaria de agradecer a Prof.ª Dr.ª Sarah Negreiros de Carvalho Leite por ter aceitado me orientar e prover todo o suporte necessário para que esse trabalho pudesse ser realizado. Agradeço ao Prof. Dr. Marcelo Moreira Tiago por ter compartilhado seus conhecimentos e contribuído de forma significativa com o desenvolvimento do trabalho. Agradeço a minha mãe Maria Inêz Perpetua Alves por ter dado suporte emocional e ter sido um pilar de sustentação durante toda a minha graduação, a Naiara Gonçalves de Freitas por sua contribuição ao trabalho, agradeço aos familiares e amigos que estiveram ao meu lado. Por fim, agradeço a UFOP, o ICEA e o DEELT por terem me acolhido durante meu período de graduação e provido estrutura e suporte para que eu pudesse chegar até o final do curso e realizar este trabalho.

Resumo

Interfaces cérebro-computador baseadas no potencial evocado visualmente em regime estacionário (BCI-SSVEP) são tecnologias inovadoras que permitem a direta conversão dos sinais cerebrais em sinais de comando ou comunicação. Neste trabalho foi desenvolvido o módulo de estimulação visual de uma BCI empregando LEDs. O circuito permite configurar as frequências de cintilação de quatro LEDs no range de 1 a 100 Hz e escolher a forma de onda de alimentação dos LEDs entre quadrada e senoidal. O sistema desenvolvido foi testado e apresentou precisão de cintilação na frequência com variação máxima de 0.5 Hz em baixas frequências (1 a 20 Hz) e de 3 Hz para frequências mais elevadas. O estimulador visual também foi integrado a um sistema BCI-SSVEP e os sinais cerebrais de uma voluntária foram coletados por eletroencefalografia. Foi analisada a influência dos seguintes parâmetros: frequência de cintilação, onda de alimentação e cor dos LEDs. O sinal cerebral obtido foi ainda comparado com o originado quando a voluntária foi exposta a estimulação usando um monitor LCD. As melhores taxas de SNR foram obtidas usando LED verde com ambas as formas de onda de alimentação. Os sinais resultantes dos estímulos em alta frequência (70, 80 e 90 Hz) apresentaram uma SNR melhor que os em baixa (6, 10, 15 e 20 Hz). Para os dados coletados, o desempenho da estimulação por LEDs pode ser equiparado ao obtido com estimulação por monitor.

Palavras-chave: BCI, SSVEP, Estimulação visual, Potencial Evocado, Estimulação por LEDs.

Abstract

Brain-computer interfaces based on steady state visually evoked potential (BCI-SSVEP) are innovative technologies that allows the direct conversion of brain signals into command or communication signals. In this study the visual stimulation module by LEDs of a BCI was developed. The circuit allows you to set the flicker frequencies of four LEDs in the range of 1 to 100 Hz and choose the waveform of the LEDs between square and sine. The developed system was tested and showed frequency scintillation accuracy with a maximum variation of 0.5 Hz at low frequencies (1 to 20 Hz) and 3 Hz at higher frequencies. The visual stimulator was also integrated into a BCI-SSVEP system and the brain signals from a volunteer were collected by electroencephalography. The influence of the following parameters was analyzed: scintillation frequency, feed wave and color of the LEDs. The obtained brain signal was also compared to that obtained when the volunteer was exposed to stimulation using an LCD monitor. The best SNR rates were obtained using green LEDs with both feeding waveforms. The high frequencies (70, 80 and 90 Hz) presented a better SNR than the low frequencies (6, 10, 15 and 20 Hz). For the collected data the performance of LED stimulation can be equated to that obtained with monitor stimulation.
Keywords: BCI. SSVEP. Visual stimulation. Evoked potential. LED stimulation.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

O avanço da computação e da compreensão sobre o cérebro humano provê um grande potencial de controlar computadores, dispositivos eletrônicos e máquinas a partir da atividade elétrica cerebral. A conexão entre o cérebro e computadores pode ocorrer por meio de uma interface cérebro-computador (BCI).

BCIs são dispositivos que possuem aplicação em diversos campos, como o militar, da biomedicina assistiva e da indústria de games e entretenimento. As BCIs podem ser empregadas, por exemplo, para ampliar a capacidade motora, permitindo o controle de próteses, cadeiras de rodas automáticas e exoesqueletos, auxiliando pessoas com sistema motores danificados e/ou desmembramentos por acidentes ou por doenças congênitas, por exemplo, em casos de paralisia e esclerose múltipla. Também podem auxiliar na comunicação, tanto escrita como na geração de áudio (SANCHEZ et al., 2011) e no controle de jogos de realidade virtual ou imersivos (MARTIšIUS; DAMAšEVIčIUS, 2016).

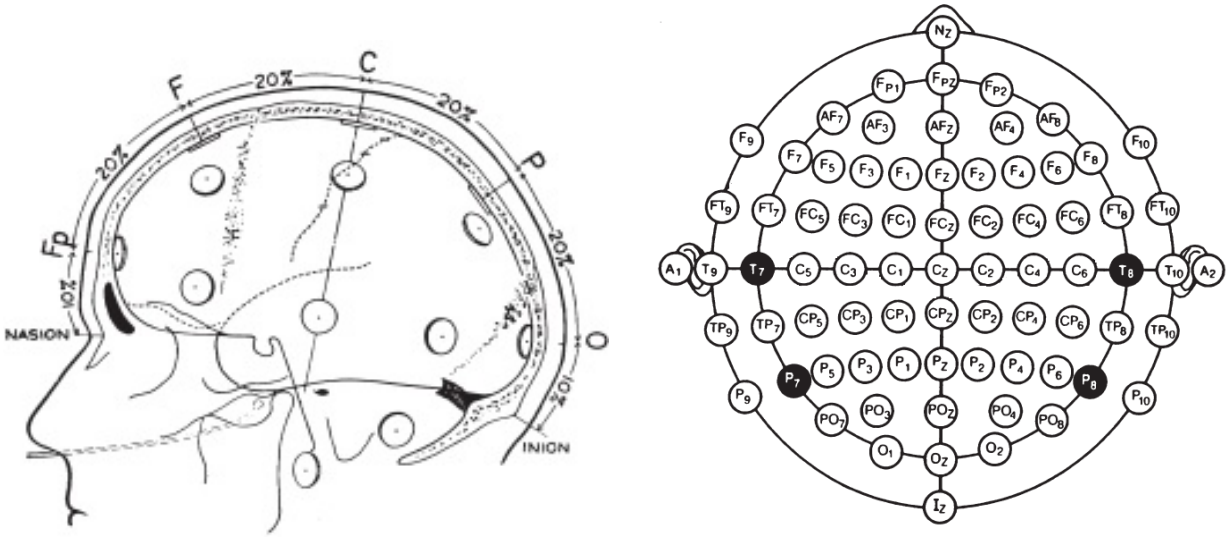
Os neurônios do cérebro se comunicam por meio de impulsos elétricos e a atividade elétrica resultante dessas interações pode ser monitorada por sensores usando técnicas invasivas ou não-invasivas.

Métodos invasivos necessitam de cirurgias para implantar eletrodos diretamente no córtex cerebral ou no tecido cortical, possibilitando a medição pontual de neurônios com alta relação sinal-ruído (SNR). Um dos processos para adquirir o sinal diretamente do córtex cerebral recebe o nome de eletrocorticografia (ECoG). O uso do ECoG pode se tornar impraticável devido ao elevado custo e às questões de ética, uma vez que é necessário um procedimento cirúrgico delicado para a implantação dos eletrodos. Além disso, os eletrodos podem desencadear uma reação de anticorpos, que pode vir a degenerar as células cerebrais e degradar o monitoramento da atividade cerebral (WALDERT, 2016).

O método não-invasivo mais usual para se adquirir sinais cerebrais é a eletroencefalografia (EEG), que consiste no posicionamento de eletrodos diretamente sobre o escalpo. Normalmente, os sensores são dispostos seguindo o sistema 10-20, conforme mostra a Figura 1. Apesar da praticidade e do baixo custo, o registro feito pelo EEG é mais suscetível a interferências e ruídos.

Os dois métodos mais comuns de conceber sistemas BCIs são empregando a imagética ou atenção seletiva. BCIs que atuam por imagética - simulação mental de um ato motor na ausência de qualquer ativação muscular - utilizam oscilações das atividades cerebrais geradas por meio da imaginação de movimentos musculares, por exemplo. As oscilações são tipicamente nomeadas e classificadas em função das bandas de frequência (delta < 4 Hz, teta 4-7 Hz, alfa 8-12 Hz, beta 12-30 Hz, gama > 30 Hz). BCIs baseadas em imagética

Figura 1 – Sistema 10-20 de posicionamento de eletrodos; (a)Vista lateral do esquema de distribuição dos eletrodos no escalpo; (b)Vista superior da distribuição dos eletrodos.



(a) (b)

Fonte: Adaptado de (KLEM et al., 1999)

não dependem de estímulos externos, porém necessitam de treinamento do indivíduo para padronizar as leituras cerebrais (GRAIMANN; ALLISON; PFURTSCHELLER, 2010).

Por outro lado, as BCIs baseadas em atenção seletiva necessitam de estímulo externo, podendo este ser auditivo (tons diferentes), somatossensorial (sensações nas partes distintas do corpo, como temperatura e pressão) ou estímulos visuais (luzes cintilantes). Uma vantagem de se usar BCIs de atenção seletiva é que nesta abordagem não é necessário realizar um treinamento prévio para o indivíduo aprender a estabelecer padrões mentais. Entretanto, o estímulo pode causar um certo desconforto ao usuário. As abordagens de BCI usando atenção seletiva mais empregadas atualmente são baseadas em P300 e SSVEP (*Steady State Visually Evoked Potential*).

As BCIs baseadas em P300 empregam o potencial que surge cerca de 300 ms após a ocorrência de um evento inesperado ou surpresa. A estimulação visual pode ser feita por LEDs, letras ou símbolos projetados em monitores (FAZEL-REZAI et al., 2012).

BCIs baseadas em SSVEP necessitam de uma fonte de luz que cintile constantemente em uma determinada frequência. Pois, quando um indivíduo é exposto a um estímulo visual, os disparos dos neurônios da região occipital entram em sincronismo com a frequência de cintilação da fonte luminosa, gerando um potencial que pode ser registrado. É possível evocar este potencial por meio de monitores ou LEDs. O uso de LEDs possibilita uma maior criatividade na distribuição espacial dos estímulos, controle analógico ou digital do brilho e possibilidade de se utilizar frequências maiores que a taxa de atualização da

tela de um monitor (60 Hz em monitores comuns). Além disso, no monitor a frequência para se gerar os estímulos deve ser sincronizada com a taxa de atualização da tela (ZHU JORDI BIEGER; AARTS, 2010). Por exemplo, para se gerar um estímulo de 30 Hz opera-se com um ciclo de dois frames, em um instante o frame está ligado e posteriormente o frame está desligado. Para 15 Hz, dois frames ligados e dois frames desligados intercaladamente, formando um ciclo de quatro frames. Por isso, monitores de 60 Hz permitem projetar estímulos nas frequências de 30, 20, 15, 12, 10, 8.57, 7.5, 6, 5, 3, 2, 1 Hz, ou seja, apenas frequências com divisores inteiros da taxa de atualização (ANDERSEN; MULLER, 2015).

Este trabalho tem como objetivo desenvolver um sistema embarcado de baixo custo e fácil manejo para estimular o potencial evocado visual no cérebro, por meio de LEDs. O sistema deve garantir exatidão na taxa de cintilação, ser estável e fornecer potência luminosa suficiente para que o sinal seja evocado. A cintilação do LED deve seguir dois padrões: onda senoidal e quadrada, de forma a permitir um estudo comparativo. Os LEDs devem ser capazes de cintilar na faixa de 1 a 100 Hz com intensidade luminosa controlável. O circuito deve ser eletromagneticamente isolado e deve permitir a disposição dos LEDs em várias posições.

O circuito projetado foi desenvolvido usando o microcontrolador Arduino Mega que permite alimentar os LEDs por diversas portas diferentes, além de permitir o manuseio das formas de onda, quadrada e senoidal, controlando a frequência e amplitude. Após gerar o estímulo, ainda é possível realimentar o Arduino Mega para realizar o controle da BCI, sem a necessidade de se empregar um outro controlador (MOULI; PALANIAPPAN; SILLITOE, 2014). O circuito de estimulação visual empregando o Arduino Mega é simples e permite a troca de cor e posicionamento dos LEDs com facilidade, a fim de se adequar melhor ao sistema desejado pelo usuário. Ele também fornece uma fonte de alimentação capaz de prover estabilidade e exatidão nas frequências de cintilação dos LEDs.

Revisão bibliográfica

Em sistemas BCI-SSVEP é importante encontrar uma forma eficaz para evocar o potencial cerebral. Tipo de fonte luminosa, frequência de estimulação, posição dos estímulos, cor, método de aquisição e registro dos sinais são fatores que podem influenciar o desempenho do sistema.

As duas abordagens mais comuns para gerar a estimulação visual é por meio de LEDs ou monitores. O LED se mostra mais vantajoso que o monitor quando se pretende empregar um número de frequências maiores, uma vez que o monitor pode explorar um número de frequências limitado pelo seu *refresh rate* (XIE et al., 2016; REJER; ZAWISLAK, 2017). O LED também permite empregar estímulos cintilantes em frequências mais elevadas (40-100 Hz) com exatidão (WU et al., 2008; WANG X. GAO; GAO, 2008). As formas de onda mais usuais para se alimentar o circuito de estimulação

por LEDs são quadrada, senoidal e triangular, que são sinais periódicos de fácil manuseio (GRAIMANN; ALLISON; PFURTSCHELLER, 2010; TENG et al., 2011).

As cores evocam uma atividade cerebral particular para cada indivíduo, portanto é difícil chegar a uma conclusão geral de qual cor tem um melhor desempenho. Grande parte das BCIs utilizam estímulos nas cores verde, violeta, vermelho, cinza, branco e preto, que aliam o conforto visual do usuário a potência do sinal evocado (BAKARDJIAN; TANAKA; CICHOCKI, 2010). No estudo (R. KHOSLA A., 2014) são comparadas as taxas de acerto ao executar os comandos de controle de um protótipo de cadeira de rodas com 5 tipos de movimentação (frente, trás, direita, esquerda e parar), foram testadas as cores azul, verde, vermelho e violeta. Foi observado uma taxa de acerto maior para a cor violeta, quase idêntica a eficiência das cores verde e vermelha, e uma taxa de acerto um pouco menor para a cor azul. Embora outros estudos apontem que a cor verde apresenta melhor desempenho para sistemas BCI-SSVEP (GAO et al., 2003).

Geralmente cor, frequência e posicionamento dos estímulos são características que devem ser personalizadas para cada usuário, de maneira a se obter uma configuração mais confortável e com melhor acurácia. Outros fatores que podem aumentar a eficiência da BCI são a distância do estímulo visual, o tipo do eletrodo e o ângulo visual (BYCZUK MARCIN & PORYZALA, 2012; HWANG et al., 2012).

Estudos realizados em animais (GRAY et al., 1989; ECKHORN, 1994) mostraram que a evocação de potencial cerebral é observada predominantemente numa faixa de frequência de 30-80 Hz. Entretanto, em humanos as frequências puderam ser satisfatoriamente captadas entre 1-90 Hz, com forte ressonância em 10, 20, 40 e 80 Hz (HERRMANN, 2001).

Frequências baixas (1-20 Hz) tendem a apresentar potencial evocado com amplitude mais elevada, porém sofrem maior efeito de ruídos, além de serem mais incomodas para o indivíduo que utiliza a BCI, uma vez que a intermitência luminosa é intensamente percebida. Frequências médias (20-40 Hz) e altas (40-100 Hz) eliminam o problema da fadiga visual do usuário, porém a potência do sinal evocado tende a ser menor. Como estudado por Yijun et al. (2005), Materka e Byczuk (2006), uma solução para compensar o fraco potencial gerado por altas frequências e reduzir o cansaço visual causado pelas baixas frequências, é modular o sinal gerador do estímulo em amplitude. Uma portadora com alta frequência e um sinal modulador em baixa frequência são combinados para que a portadora diminua o cansaço visual e o modulador, geralmente na faixa de banda alfa, forneça o potencial pela manifestação de harmônicas (CHANG et al., 2013). Estudos sobre a faixa de frequência geralmente acabam resultando em frequências com forte SSVEP em 10, 13-25 e 40-60 Hz, porém a maioria dos teste não são feitos adquirindo o sinal pelo ponto de vista da BCI ou não utilizam um número adequado de amostras, seja de frequência ou de indivíduos. Uma pesquisa feita com modelo estatístico utilizando dez

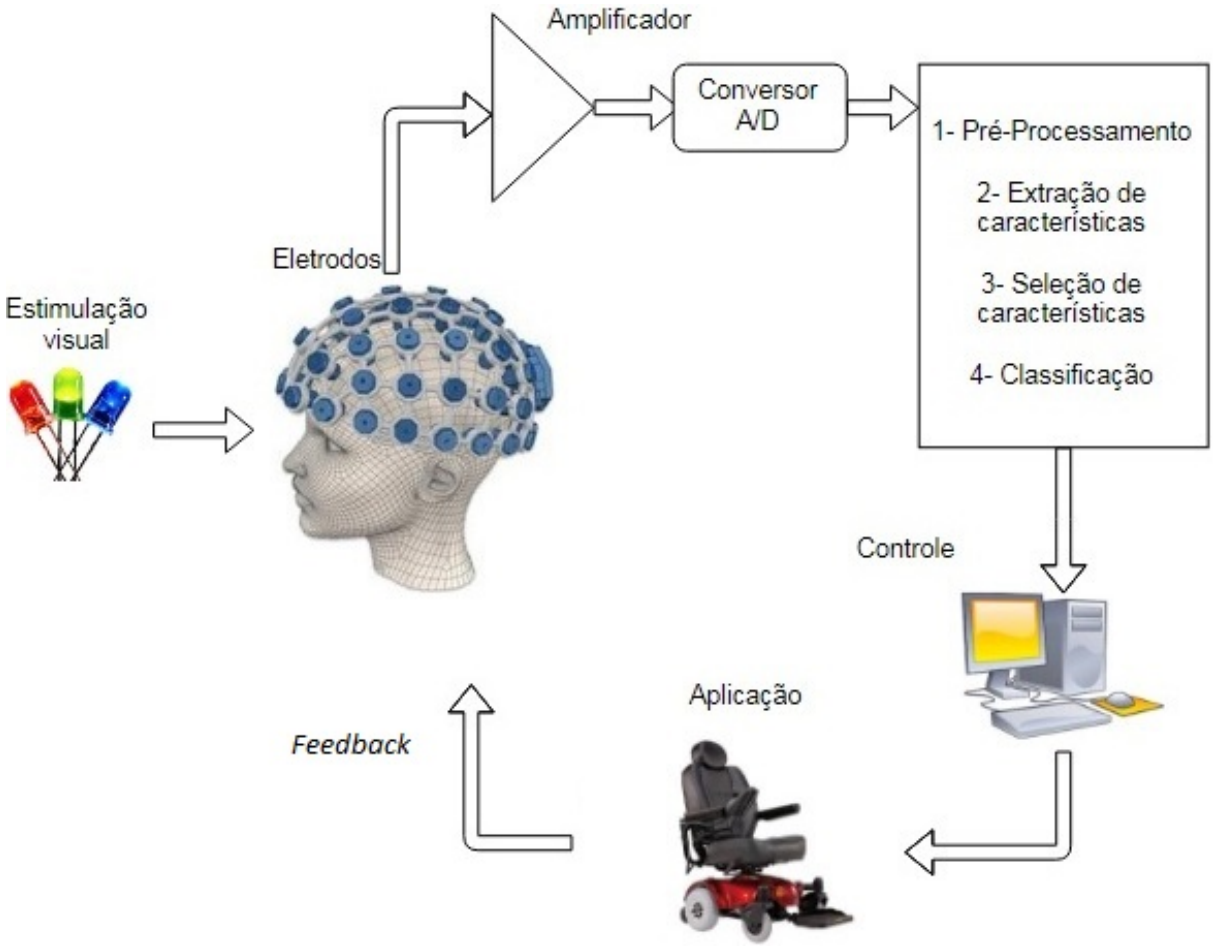
voluntários, captou sinais de 5 a 30 Hz num passo de 1 Hz por aquisição e apontou uma faixa de frequência ideal entre 12-18 Hz após calcular a média de todos os indivíduos (KUZ et al., 2013).

Outra abordagem para evocar o potencial visual é apresentada por Zhang et al. (2012) e implementada por Hwang et al. (2013). Eles propõem substituir o estímulo contínuo por uma codificação com diferentes frequências, de maneira a gerar uma sequência de sinais para cada comando. Os resultados deles se mostraram eficientes e comparáveis aos métodos mais usados. A principal vantagem desta abordagem é permitir utilizar um número maior de comandos com frequências limitadas, caso alguma faixa de frequência possua uma resposta consideravelmente melhor para determinado indivíduo.

2 Metodologia

O funcionamento de uma BCI-SSVEP pode ser esquematizado conforme mostra a Figura 2. Um dispositivo gera um estímulo visual numa determinada frequência, o usuário foca sua atenção nesse estímulo e um potencial evocado emerge no lobo occipital com a mesma frequência do estímulo gerado. Os eletrodos captam este potencial evocado, que é da ordem de milivolts (mV). Na sequência, o sinal é amplificado, digitalizado e processado para identificação do comando desejado a ser enviado à aplicação.

Figura 2 – Sistema genérico de uma BCI-SSVEP.



Neste trabalho, desenvolveu-se um estimulador visual de LEDs para ser integrado a um sistema BCI-SSVEP. O potencial evocado visualmente no cérebro pode ser captado com qualidade no range de frequências de 1 até 100 Hz, assim é interessante permitir que os LEDs trabalhem neste range, de maneira a alcançar um número maior de frequências utilizáveis e consequentemente, um maior número de comandos possíveis para a BCI.

Para controlar a intensidade e frequência de cintilação dos LEDs, optou-se por

utilizar o microcontrolador ATmega2560, que vem integrado na placa do sistema embarcado Arduino Mega 2560. A escolha por esse microcontrolador foi feita pelo fato dele possuir 15 saídas do tipo PWM (modulação por largura de pulso), as quais podem ser utilizadas para gerar a alimentação intermitente de até 15 LEDs seguindo o padrão de uma onda senoidal. Uma outra opção de microcontrolador é o Arduino Duo que possui duas saídas de sinal analógico. A desvantagem nesse caso é que seriam necessários dois microcontroladores para controlar 4 LEDs, por exemplo. Além disso, o Arduino Duo é mais custoso em comparação com o utilizado.

O Arduino Mega possui 5 temporizadores internos, que fazem a regulagem de funções de tempo e controlam a frequência de algumas saídas PWM alterando os registradores do microprocessador. Cada temporizador controla duas saídas moduladas. As saídas PWM vem originalmente com os temporizadores programados para frequência do PWM normal de 490 Hz e de 980 Hz para o PWM rápido (Pinos: 13 e 4), apesar de ser uma frequência pelo menos cinco vezes maior que a faixa de operação desejada, ainda assim não é rápido o suficiente para gerar um sinal com qualidade semelhante a um sinal puramente analógico. Alterando os registradores dos temporizadores é possível obter aumento da frequência do PWM. É desejado sempre uma frequência muito maior que a do sinal modulado, e o ATmega2560 permite aumentar até 31.5 kHz nos pinos de uso geral.

O Arduino Mega alimenta os LEDs com a corrente controlada por um resistor. Para o conforto do usuário, um potenciômetro pode ser utilizado para regular a intensidade do brilho do LED. O sistema de estimulação também exige que a frequência de cintilação dos LEDs seja bem definida, pois variações em torno da frequência são perceptíveis ao mecanismo neurofisiológico cerebral resultando em potenciais evocados menos intensos.

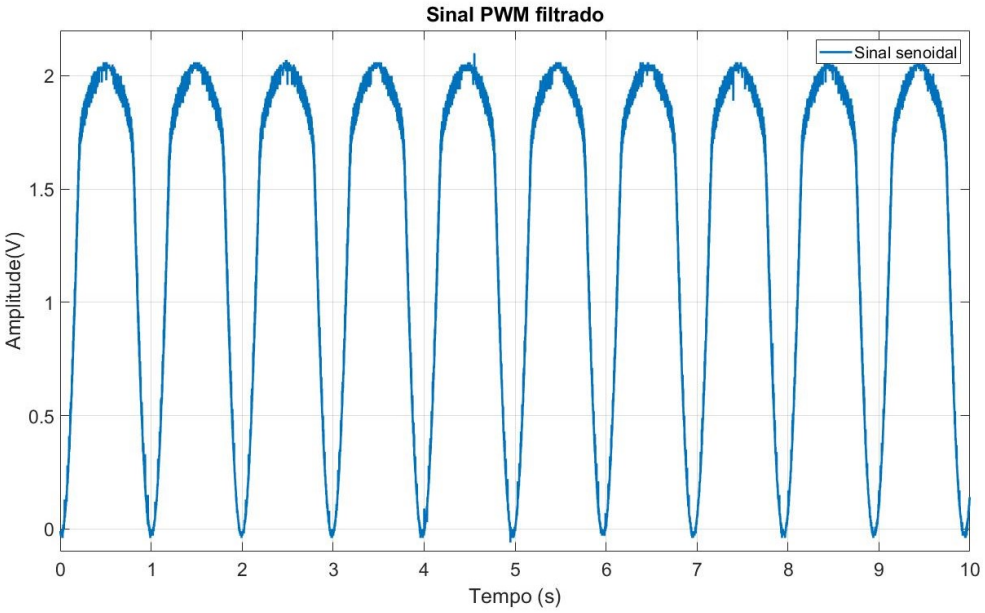
Outro fator importante a ser considerado é a interferência eletromagnética indesejada de cabos e componentes metálicos que conduzem corrente. Para minimizar este problema, foi realizado uma isolação de todo o circuito cobrindo cabos expostos, gerando a placa de circuito impresso e encapsulando o circuito em uma gaiola de Faraday.

2.1 Geração de estímulo por meio de onda senoidal

Uma onda senoidal é um sinal contínuo caracterizado por frequência, fase e amplitude. O sinal senoidal quando aplicado em um LED sofre uma deformação em sua amplitude devido a tensão de corte característica dos diodos. A Figura 3 apresenta o sinal de excitação de um LED com frequência de 1 Hz. Como é um sinal que não possui uma variação brusca entre os picos de máximo positivo e negativo, a transição entre os estados aceso e apagado é mais suave e imune a interferência.

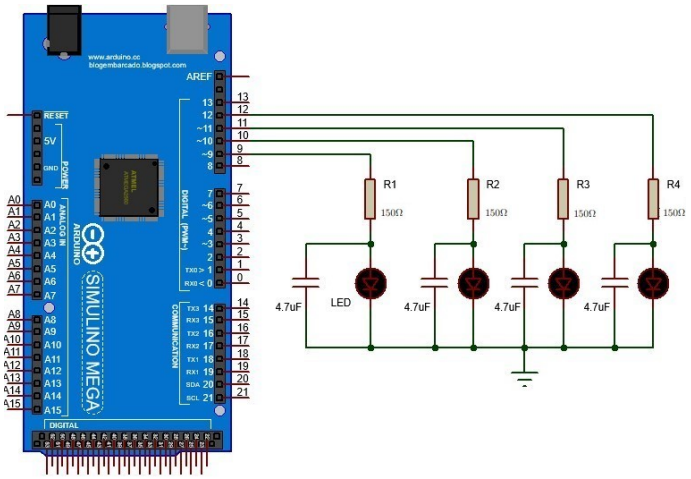
Como todas as saídas do Arduino Mega são digitais, não é possível conseguir diretamente uma saída analógica sem a utilização de algum conversor da saída digital para

Figura 3 – Sinal de excitação de um LED de cor verde a partir de uma onda senoidal.



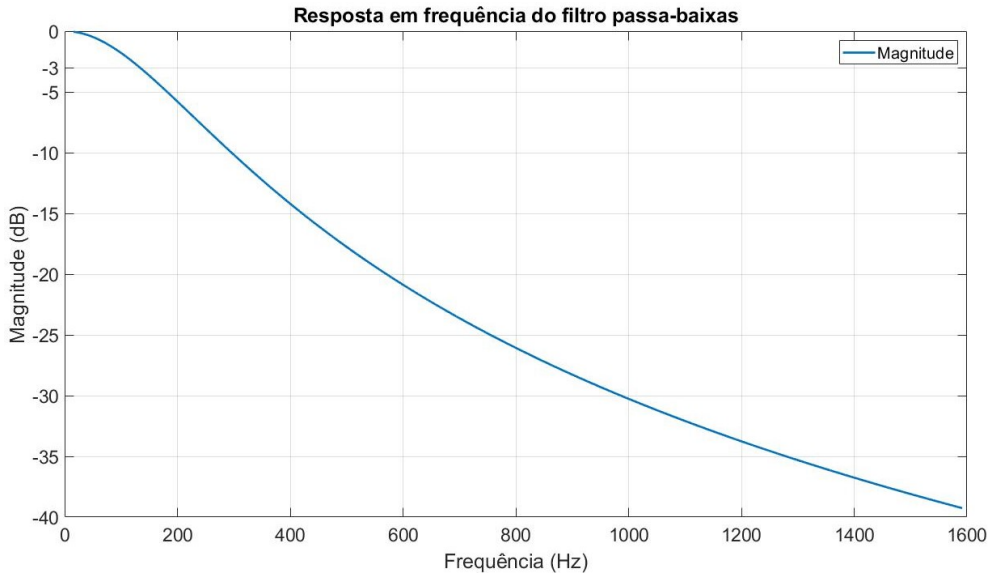
analógica. Um conversor D/A é um circuito que recebe na entrada uma palavra digital e fornece na saída uma tensão proporcional a um número binário fornecido na entrada. Este processo aumentaria a complexidade do circuito, além de encarecê-lo, uma vez que seria necessário inserir um conversor para cada LED.

Figura 4 – Diagrama do circuito RC de filtragem do sinal PWM.



Um método mais simples, e tão eficaz quanto, é a modulação por largura de pulso (PWM) que fornece essa conversão indiretamente com o emprego de um filtro passa-baixas para filtrar a frequência do PWM. Para garantir uma melhor filtragem, sem risco de perder o sinal, é necessário empregar uma frequência de PWM elevada em relação a frequência do sinal. Um simples filtro passivo RC pode ser usado para realizar a filtragem, já que a faixa de transição não precisa ser muito estreita. Para o projeto do filtro, foi utilizado um resistor de 150 Ω e um capacitor de 4.7 μF, como apresentado pela Figura 4, resultando em uma frequência de corte de aproximadamente 225 Hz. Como o filtro RC é um filtro passivo

Figura 5 – Resposta em frequência do filtro RC passa-baixas.



de ordem 1, a frequência de corte não é precisa em 225 Hz, como pode ser observado na Figura 5 que apresenta a sua resposta em frequência.

O algoritmo apresentado no Apêndice A faz uso da função *sin*() da biblioteca matemática do Arduino Mega para gerar uma referência senoidal que é mapeada em PWM, normalizando cada nível de tensão da função em uma largura de pulso que pode variar entre 0 e 100%.

O nível de corte do funcionamento dos LEDs depende de sua cor: os azuis possuem tensão de corte em torno de 2.5 V, enquanto que os verdes e vermelhos possuem tensão de corte em torno de 1.8 - 2 V. Essas são as tensões necessárias para que o LED ascenda. A tensão de saída do Arduino Mega varia entre 4 e 5 V, sendo muito superior a tensão de corte do LED. Desta forma, a senoide precisa ser calibrada para manter o tempo do LED aceso igual ao tempo do LED apagado. Uma desproporção entre os tempos aceso/apagado poderia prejudicar a frequência de cintilação desejada, acarretando num estímulo menos eficiente para gerar a atividade cerebral desejada.

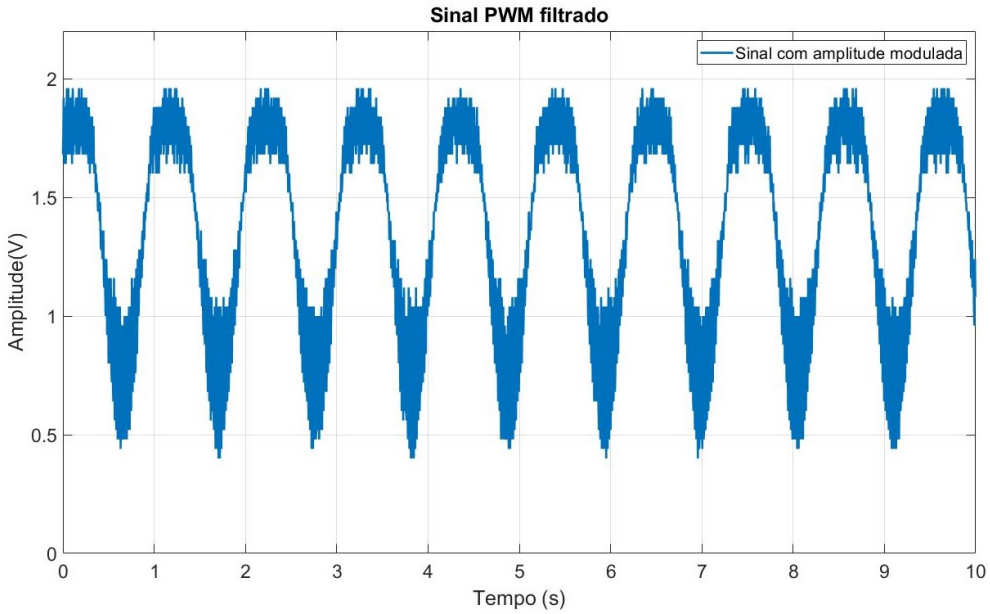
2.1.1 Sinal senoidal com amplitude modulada

Uma importante característica de um sinal senoidal é que ele pode ser modulado em amplitude por uma portadora de alta frequência (>40 Hz) (YIJUN et al., 2005; MATERKA; BYCZUK, 2006), a fim de tornar a estimulação visual menos fadigante e compensar a fraca potência do SSVEP em altas frequências pelo uso do sinal modulador em baixa frequência (<12 Hz) (CHANG et al., 2013). A modulação consiste em cintilar o LED na frequência desejada por meio de uma portadora em uma frequência elevada. Para realizar a modulação, um sinal modulador é gerado na frequência desejada para o estímulo, depois é multiplicado por uma portadora de frequência elevada, realizando a

modulação, e para finalizar o sinal modulador é novamente multiplicado pela portadora, ocorrendo a demodulação. Neste processo, é preciso respeitar o teorema de amostragem de Nyquist-Shannon, para que se possa recuperar o sinal sem perdas por *"aliasing"*. Para isso, a frequência da portadora deve ser pelo menos duas vezes maior que a frequência do sinal modulador.

A Figura 6 apresenta um sinal senoidal modulado em amplitude de 1 Hz. Para realizar essa modulação, foi gerado uma referência senoidal de 1 Hz e multiplicada por uma portadora de 100 Hz. Antes de se gerar o PWM, foi multiplicado novamente pela portadora para realizar a demodulação e assim ser gerado a referência em 1 Hz. É possível observar que o sinal apresenta uma deformação elevada, ou seja, se mostra bem mais ruidoso que o sinal senoidal puro. Esse ruído pode provocar inexatidão e instabilidade, causando o mal funcionamento da BCI.

Figura 6 – Sinal de excitação de 1 Hz modulado em amplitude.

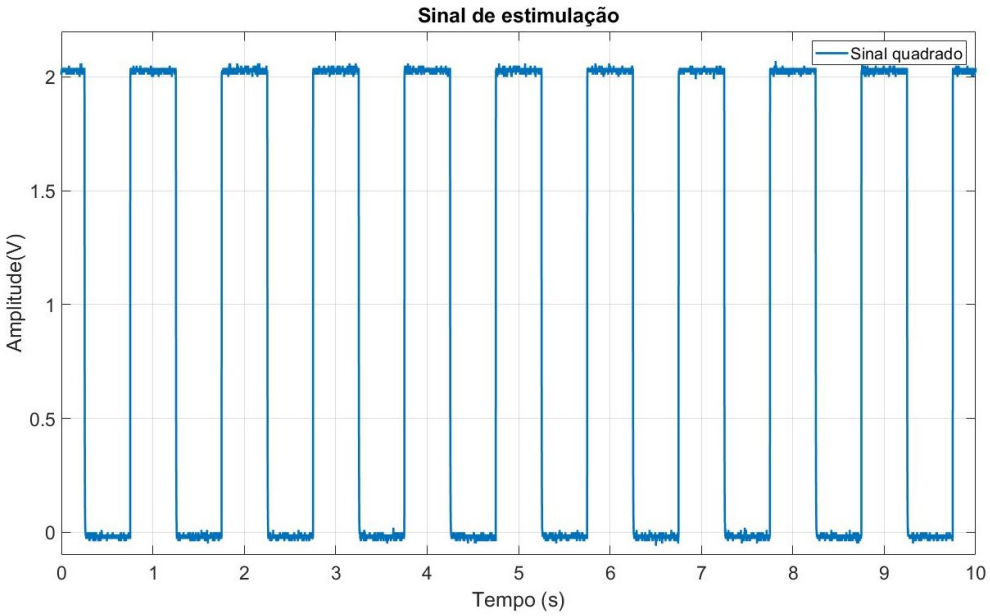


2.2 Geração de estímulo por meio de onda quadrada

Ondas quadradas são sinais periódicos que alternam de maneira abrupta entre dois níveis de amplitude específicos, conforme mostra a Figura 7. Elas podem ser escritas em termos de uma soma de infinitas senoides que oscilam nas harmônicas ímpares, considerando a decomposição em série de Fourier. A expressão 2.1 descreve uma onda quadrada de frequência ω.

$$f(\omega) = \frac{4}{\pi}\sum_{k=0}^{\infty}\frac{1}{(2k+1)}sen[(2k+1)\omega] = \frac{4}{\pi}sen(\omega) + \frac{4}{3\pi}sen(3\omega) + \frac{4}{5\pi}sen(5\omega) + ... \quad (2.1)$$

Figura 7 – Onda quadrada alimentando um LED de cor verde.

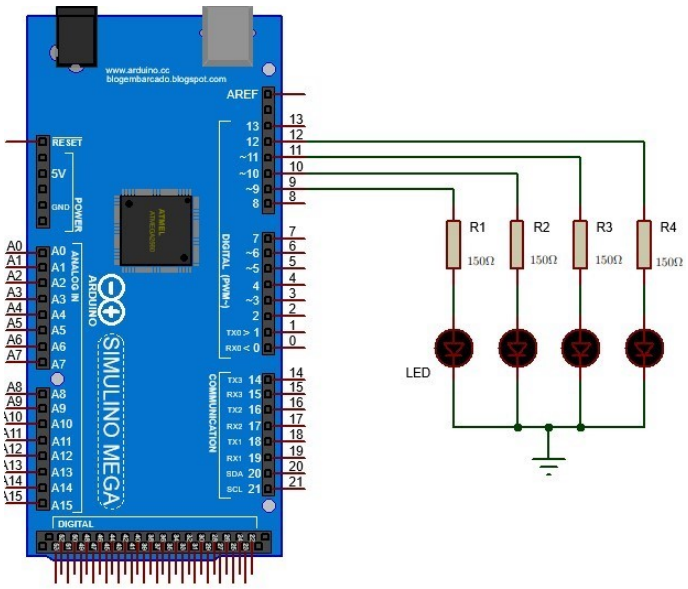


Ao empregar este padrão de controle de tensão dos LEDs, se está sujeito à interferência das componentes harmônicas, que podem gerar ruído no potencial evocado cerebral que se deseja mensurar. Este fato é explorado durante o tratamento e análise da atividade cerebral monitorada.

As ondas quadradas podem ser geradas diretamente a partir de interrupções dos temporizadores do microcontrolador, fazendo a troca de estado da saída: ligado/desligado. Para realizar as trocas de estado são utilizados contadores para medir o tempo relativo a metade da frequência desejada. Como são apenas dois estados, o tempo do LED aceso e apagado são iguais, independente da tensão de corte, embora seja possível alterar o *duty cycle*, para configurar mais tempo ligado ou mais tempo desligado. O sinal leva um intervalo de tempo estreito de aproximadamente 1 ms para realizar a troca de estado.

O algoritmo de geração do sinal de onda quadrada (Apêndice B) é bem simples e faz o uso de funções de um temporizador (Apêndice B.1 e Apêndice B.2) para gerar as interrupções com mais precisão. O circuito elétrico de alimentação por onda quadrada contem quatro resistores para limitar a corrente de excitação de cada LED em série com cada saída digital do Arduino, conforme ilustrado pela Figura 8.

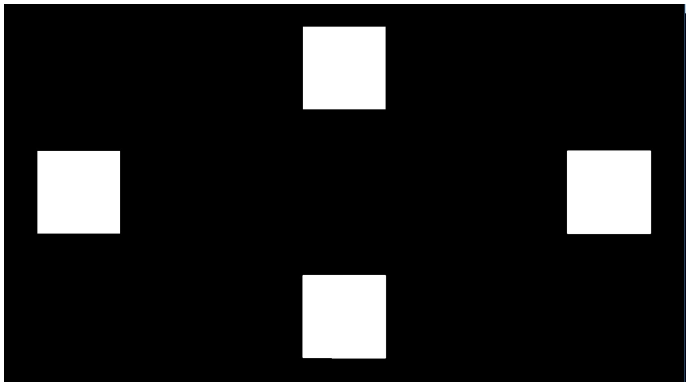
Figura 8 – Diagrama do circuito elétrico para onda quadrada.



2.3 Estímulo visual gerado por monitor

A título de comparação do sinal evocado por LEDs, foram realizados testes usando um sistema de estimulação visual usando um monitor LCD. A tela de estimulação consistiu em quatro estímulos posicionados à esquerda, direita, cima e baixo do monitor, conforme mostra a Figura 9, as figuras quadradas oscilavam entre as cores branco e preto em um fundo preto.

Figura 9 – Posicionamento dos estímulos visuais do monitor.



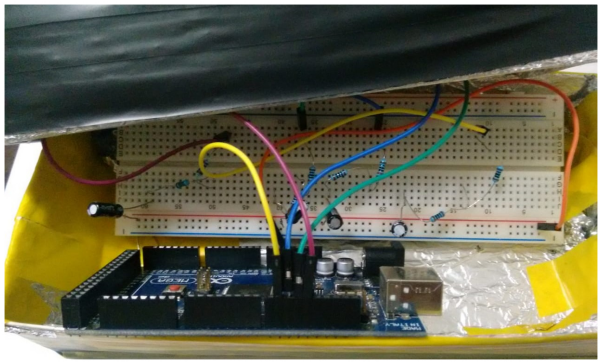
2.4 Procedimento experimental

O circuito elétrico foi encapsulado em uma caixa foleada com papel alumínio e fita isolante, conforme apresenta a Figura 10a, a fim de reduzir interferentes eletromagnéticos. Os LEDs foram posicionados em um monitor desenergizado (Figura 10b), com a frequência menor em cima e incrementando em sentido anti-horário. As pontas de metais dos LEDs

e a solda com o fio foram cobertas com fita isolante para minimizar as interferências eletromagnéticas.

Figura 10 – Procedimento de coleta de dados da BCI.

(a) Circuito dentro da caixa de isolação eletromagnética.



(b) Sistema da BCI com LEDs posicionados em frente a voluntária.

Foram realizadas 2 coletas de 12 segundos cada nos cenários especificados na Tabela 1.

Tabela 1 – Cenários de teste

Cenário	**Estímulo**	**Alimentação do estímulo**	**Frequência de estimulação**
1	LED vermelho	onda senoidal	6, 10, 15, 20, 40, 70, 80 e 90 Hz
2	LED vermelho	onda quadrada	6, 10, 15, 20, 40, 70, 80 e 90 Hz
3	LED verde	onda senoidal	6, 10, 15, 20, 40, 70, 80 e 90 Hz
4	LED verde	onda quadrada	6, 10, 15, 20, 40, 70, 80 e 90 Hz
5	LED azul	onda senoidal	6, 10, 15, 20, 40, 70, 80 e 90 Hz
6	LED azul	onda quadrada	6, 10, 15, 20, 40, 70, 80 e 90 Hz
7	Monitor LCD	onda senoidal	6, 10, 15 e 20 Hz

Participou do experimento uma voluntária. Os sinais EEG foram coletados usando 16 eletrodos secos posicionados na região occipital, parietal e central. O sinal foi filtrado analogicamente por um filtro Butterworth de oitava ordem na faixa de 5 a 100 Hz e por um filtro notch de quarta ordem em torno da frequência de alimentação da rede (60 Hz). Posteriormente, o sinal foi digitalizado a uma taxa de amostragem de 256 Hz e filtrado digitalmente pelo filtro espacial CAR (*Common Average Reference*) para retirada das interferências comuns aos eletrodos (TELLO et al., 2015).

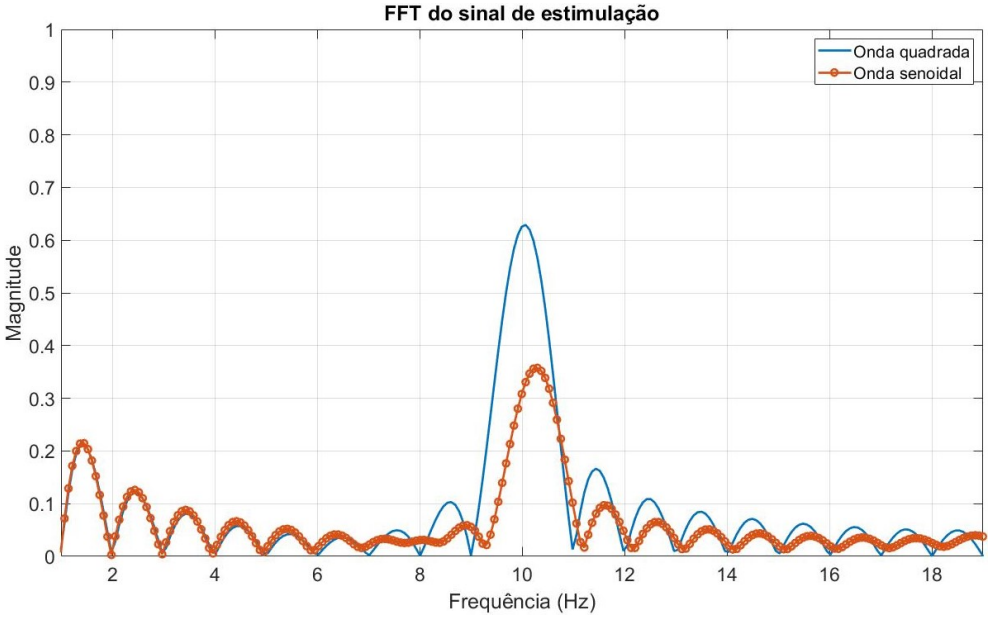
3 Resultados

Na primeira parte do experimento avaliou-se as ondas geradas pelo Arduino Mega para controlar o ritmo liga/desliga dos LEDs. Adotou-se como referência de sinal ideal a forma de onda gerada em um gerador de funções. Um osciloscópio foi utilizado para acompanhar a frequência do estímulo gerado pelo Arduino Mega e comparar as formas de onda dos sinais gerados nos quesitos estabilidade e precisão de oscilação na frequência desejada. Num segundo momento, coletou-se o sinal cerebral de uma voluntária enquanto esta era exposta aos estímulos visuais gerados pelo sistema de LEDs desenvolvido. Nesta etapa avaliou-se o sinal EEG coletado também por estimulação via monitor, o parâmetro de comparação considerado foi a SNR.

3.1 Estímulo gerado pelo Arduino Mega

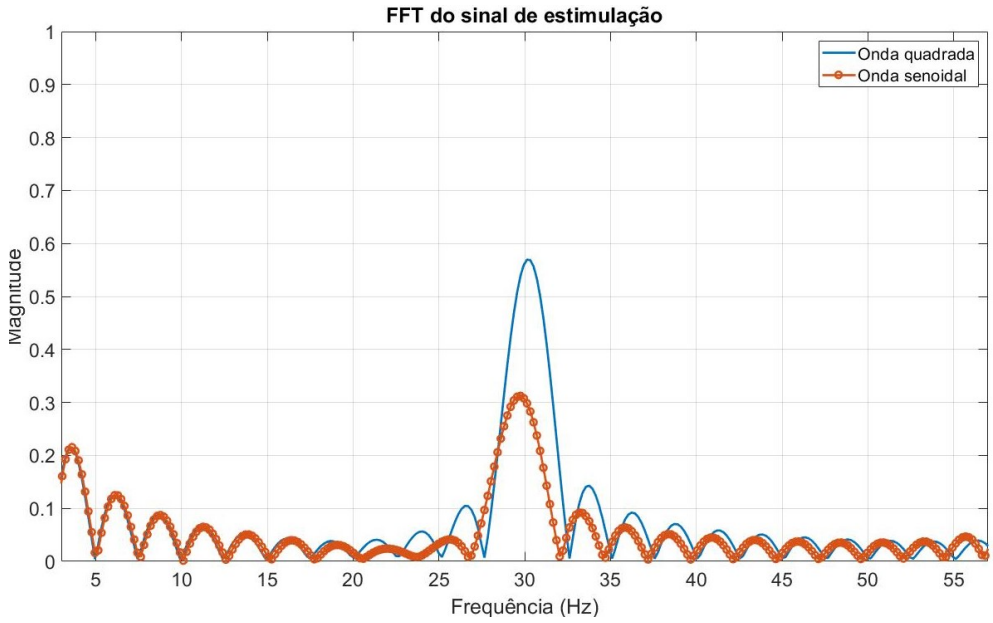
A Figura 11 apresenta a FFT do sinal de excitação do LED (obtida pelo código apresentado no Apêndice C) com uma frequência de 10 Hz, representando as baixas frequências. A linha azul representa a alimentação por onda quadrada e a linha laranja por onda senoidal, as formas de onda foram geradas diretamente pelo Arduino Mega. Espera-se que um sinal com exatidão apresente o pico do lóbulo principal na frequência de oscilação, já a estabilidade é indicada pela largura de banda do pico, quanto mais estável mais estreita.

Figura 11 – Ondas quadrada e senoidal de baixa frequência (10 Hz) gerada pelo Arduino Mega.



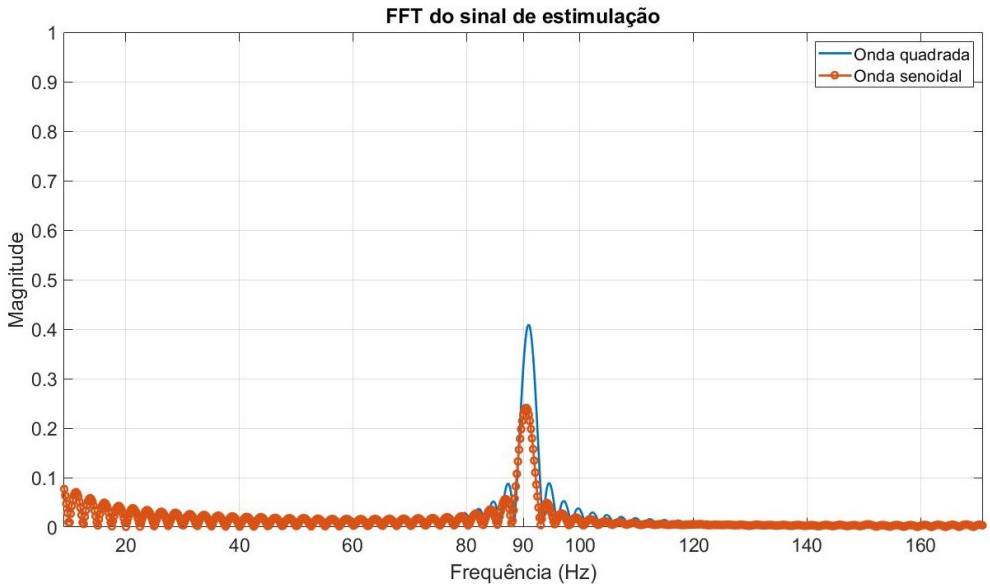
A Figura 12 contém a transformada de Fourier dos sinais de excitação de onda quadrada (azul) e senoidal (laranja) a uma frequência de 30 Hz, considerada como uma frequência média. Ambos os sinais apresentaram formas parecidas, contudo, o sinal de onda quadrada apresentou uma exatidão um pouco melhor, além de apresentar maior potência de excitação.

Figura 12 – Ondas quadrada e senoidal de média frequência (30 Hz) gerada pelo Arduino Mega.



As altas frequências apresentaram uma redução considerável de potência em relação as baixas e médias, conforme apresentado pela Figura 13. Pode-se observar que o desvio da exatidão é um pouco maior também.

Figura 13 – Ondas quadrada e senoidal de alta frequência (90 Hz) gerada pelo Arduino Mega.



As Tabelas 2 e 3 apresentam, de forma mais detalhada, os resultados de interesse. A frequência do pico é um parâmetro de exatidão, quanto mais próximo da frequência de estimulação mais preciso, a largura de banda representa a estabilidade do sinal, é desejado um valor menor para que não seja evocado potencial em frequências próximas à frequência estipulada e a amplitude do pico do sinal representa a potência do sinal de cintilação em relação a componente DC, essa tensão contínua aparece devido ao fato da corrente sempre fluir para fora do Arduino, ou seja, a tensão de alimentação possui *"offset"*.

Tabela 2 – Sinal da onda quadrada gerado pelo Arduino Mega.

Frequência desejada [Hz]	**Frequência obtida [Hz]**	**Largura de banda [Hz]**	**Amplitude do lóbulo principal [V]**
1	1.007	0.12	0.65
5	4.997	0.61	0.64
10	10.070	1.22	0.63
15	15.030	1.22	0.62
20	19.990	1.22	0.60
25	24.990	3.05	0.58
30	30.140	2.86	0.57
40	39.860	3.05	0.54
70	70.380	3.05	0.45
80	80.680	3.06	0.43
90	90.980	3.06	0.41
100	100.100	2.99	0.39

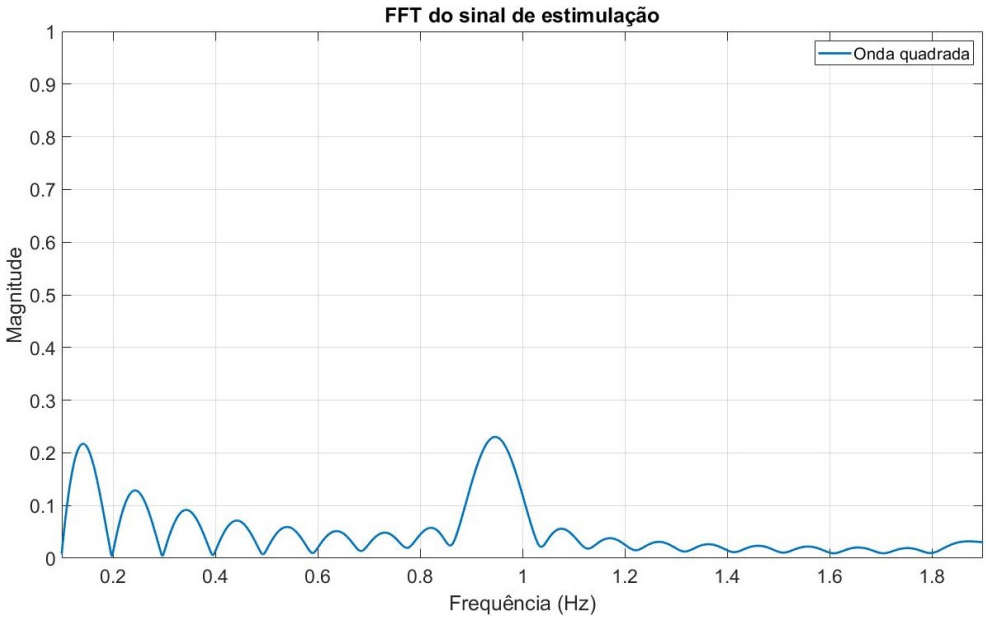
Tabela 3 – Sinal da onda senoidal gerado pelo Arduino Mega.

Frequência desejada [Hz]	**Frequência obtida [Hz]**	**Largura de banda [Hz]**	**Amplitude do lóbulo principal [V]**
1	0.999	0.12	0.35
5	5.074	0.61	0.34
10	10.300	1.14	0.36
15	15.030	1.14	0.35
20	20.220	3.05	0.33
25	25.180	3.05	0.33
30	29.750	3.24	0.31
40	38.910	3.06	0.30
70	69.810	2.86	0.27
80	80.950	1.30	0.26
90	90.410	3.24	0.24
100	100.600	1.20	0.23

3.1.1 Estímulo senoidal modulado em amplitude

A modulação em amplitude da onda senoidal se mostrou inviável de ser utilizada, porque o sinal apresentou grande influência de ruídos, como mostra a Figura 6. Além disso, analisando a transformada de Fourier do sinal, Figura 14 fica evidente a baixa relação sinal/ruído, o que prejudica a detecção da frequência de estimulação pela BCI.

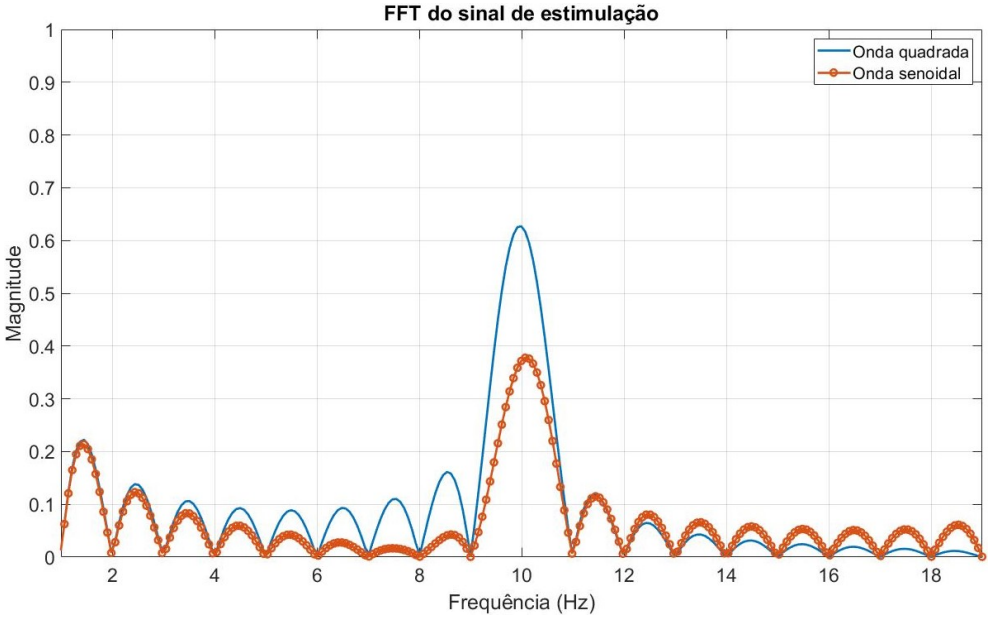
Figura 14 – Transformada de Fourier do sinal senoidal modulado em amplitude.



3.2 Ensaio utilizando gerador de funções

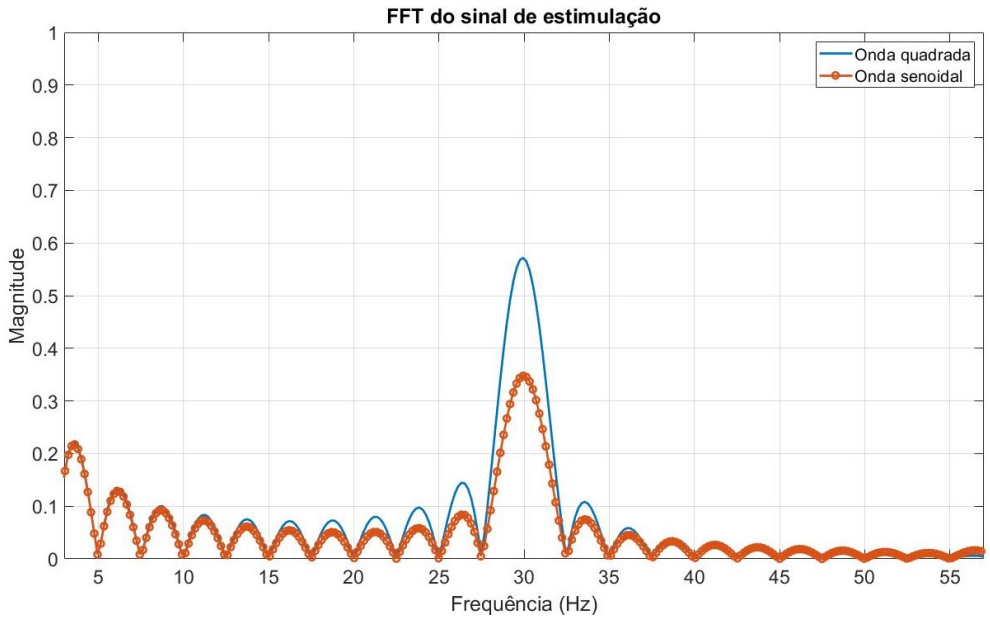
O sinal gerado pelo Arduino Mega foi comparado com o do gerador de sinais. A Figura 15 mostra a frequência de 10 Hz. A análise gráfica demonstra que para o gerador de funções, ambas as formas de onda possuem boa exatidão e boa estabilidade.

Figura 15 – Ondas quadrada e senoidal de baixa frequência (10 Hz) do gerador de funções.



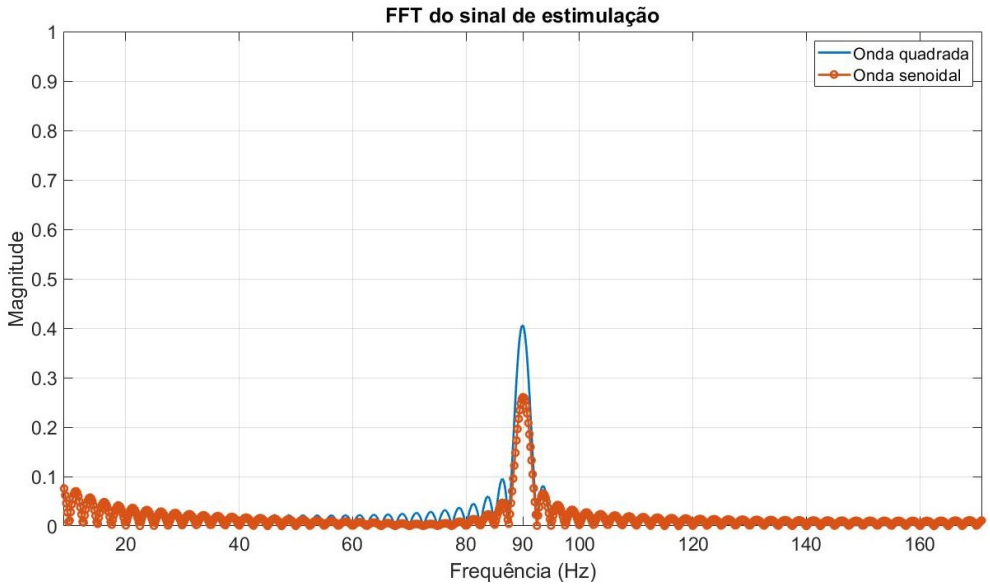
A Figura 16 mostra uma média frequência gerada em 30 Hz. A exatidão e estabilidade do sinal ainda se manteve com boa qualidade. A potência do sinal de excitação apresentou ser um pouco mais fraca que para o caso de baixas frequências.

Figura 16 – Ondas quadrada e senoidal de média frequência (30 Hz) do gerador de funções.



A Figura 17 mostra um exemplo de alta frequência gerada em 90 Hz. A análise gráfica demonstra que para o gerador de funções ambas as formas de onda possuem boa exatidão e boa estabilidade, porém a potência do sinal fica gradativamente mais fraca ao se aumentar a frequência. A potência do sinal senoidal continuou sendo mais fraca que a do sinal quadrado e apresentou menos ruído.

Figura 17 – Ondas quadrada e senoidal de alta frequência (90 Hz) do gerador de funções.



Os resultados numéricos obtidos pelo gerador de funções são apresentados pelas Tabelas 4 e 5.

Tabela 4 – Sinal de onda quadrada gerado pelo gerador de funções.

Frequência desejada [Hz]	**Frequência obtida [Hz]**	**Largura de banda [Hz]**	**Amplitude do lóbulo principal [V]**
1	0.999	0.12	0.65
5	4.997	0.57	0.64
10	9.995	1.23	0.63
15	15.030	1.14	0.61
20	20.220	2.86	0.61
25	24.990	3.05	0.59
30	29.950	3.05	0.57
40	40.050	3.05	0.54
70	70.000	3.06	0.45
80	79.920	3.05	0.43
90	90.030	2.86	0.40
100	99.950	3.08	0.39

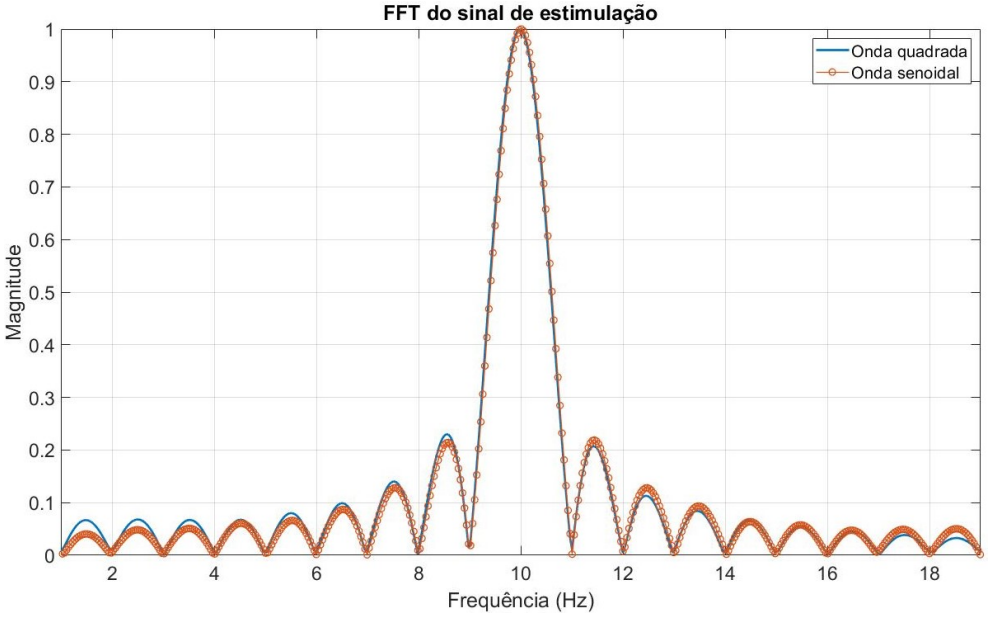
Tabela 5 – Sinal de onda senoidal gerado pelo gerador de funções.

Frequência desejada [Hz]	**Frequência obtida [Hz]**	**Largura de banda [Hz]**	**Amplitude do lóbulo principal [V]**
1	0.991	0.11	0.38
5	4.997	0.61	0.38
10	10.070	1.22	0.38
15	14.950	1.22	0.37
20	19.840	2.86	0.36
25	24.990	3.05	0.36
30	29.950	3.05	0.35
40	40.050	3.05	0.33
70	70.000	3.06	0.29
80	79.920	3.05	0.27
90	90.030	3.05	0.26
100	100.010	3.28	0.25

3.3 Estímulo do gerador de funções captado pelo fotodiodo

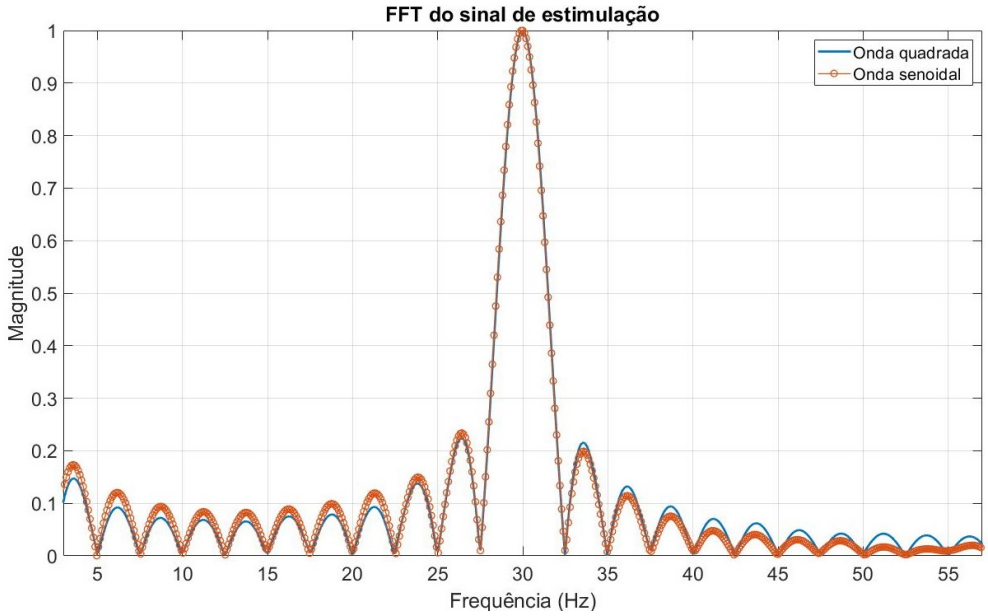
Foram realizados testes para avaliar o efeito da intensidade de cintilação do LED, por meio de um fotodiodo encostado, sem distância entre as capsulas, e fixado com uma fita isolante preta para que não houvesse interferência de alguma fonte luminosa externa. A Figura 18 mostra a FFT dos sinais gerados por onda quadrada e senoidal oscilantes em 10 Hz. Para ambas as formas de onda o resultado foi semelhante, sinal com exatidão adequada e largura de banda estreita.

Figura 18 – Ondas quadrada e senoidal de baixa frequências (10 Hz) do gerador de funções captadas por um fotodiodo.



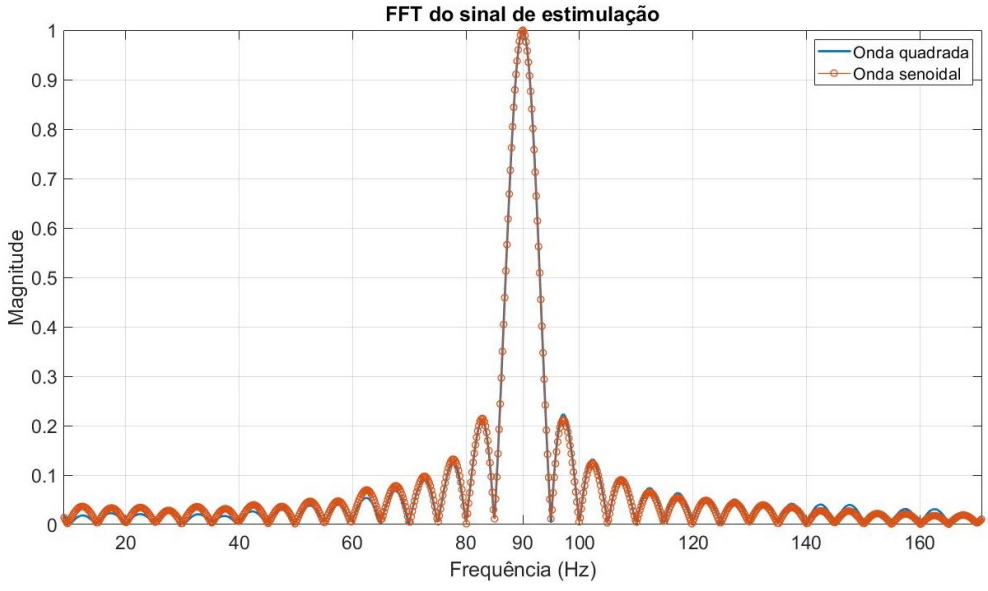
A média frequência de 30 Hz, representada pela Figura 19, mostrou resultados semelhantes a frequência de 10 Hz, os sinais de onda quadrada e senoidal não apresentaram muita diferença.

Figura 19 – Ondas quadrada e senoidal de média frequências (30 Hz) do gerador de funções captadas por um fotodiodo.



A Figura 20 apresenta o sinal de alta frequência em 90 Hz. Novamente, o resultado foi semelhante entre as ondas quadrada e senoidal. A ideia desse teste era captar a perspectiva de um sensor simulando um olho humano, usando um circuito totalmente desacoplado.

Figura 20 – Ondas quadrada e senoidal de alta frequências (90 Hz) do gerador de funções captadas por um fotodiodo.



Os resultados numéricos do sinal captado pelo fotodiodo estão apresentados na Tabela 6.

Tabela 6 – Ondas quadrada e senoidal do gerador de funções captadas pelo fotodiodo.

Frequência desejada [Hz]	**Frequência captada [Hz]**	**Largura de banda [Hz]**	**Amplitude do lóbulo principal [V]**
2	2.003	0.12	1
10	10.010	1.19	1
15	14.980	1.24	1
20	20.030	1.23	1
25	24.990	1.19	1
30	30.000	2.98	1
40	40.030	2.98	1
70	69.930	5.96	1
80	79.870	6.16	1
90	90.000	5.96	1
100	99.930	6.15	1

3.4 Dados coletados da BCI

O critério de avaliação de desempenho da estimulação visual empregado baseou-se na relação sinal ruído calculada em torno do estímulo, utilizando a função do Apêndice D. Para frequências até 20 Hz, considerou-se o sinal em torno da frequência de cintilação com uma margem de +-0.5Hz, e o cálculo do ruído foi feito numa faixa de +-2Hz, de maneira a evitar sobreposição com faixas de estímulos em frequências próximas.

Para os estímulos em frequências superiores a 20 Hz, foi calculada a potência do sinal considerando uma faixa de +- 3 Hz em torno da frequência de estimulação e de +- 6 Hz para a estima da potência do ruído local, já que os sinais de excitação se mostraram menos estáveis a medida que a frequência de cintilação aumentava e a distância entre as frequências empregadas era maior.

A análise foi feita utilizando os sinais registrados pelo eletrodo Oz posicionado na região occipital (ver Fig. 1). A Tabela 7 apresenta os valores em decibéis (dB) da SNR calculados em cada um dos cenários de teste (Tabela 1) nas duas coletas de 12 s.

Tabela 7 – Valores da SNR em dB.

Cenário	**Coleta**	**6 Hz**	**10 Hz**	**15 Hz**	**20 Hz**	**40 Hz**	**70 Hz**	**80 Hz**	**90 Hz**
1	**1**	3.43	-0.96	-0.96	-1.92	3.81	14.75	9.93	10.50
	2	1.76	-1.44	1.79	0.18	-1.37	9.95	5.04	6.55
2	**1**	3.76	-2.16	2.18	1.50	3.98	14.84	11.33	12.79
	2	2.77	-0.76	2.62	-0.24	0.10	10.56	7.19	8.86
3	**1**	3.64	1.00	5.32	4.95	8.12	14.70	11.28	7.69
	2	3.28	2.42	6.30	2.32	3.42	8.35	6.34	3.69
4	**1**	2.78	4.41	9.09	2.65	10.85	16.82	14.63	10.39
	2	2.07	3.42	6.11	1.80	6.20	12.80	10.31	6.23
5	**1**	3.80	1.80	2.39	3.74	5.72	13.51	8.33	16.01
	2	2.93	-1.35	0.34	2.02	0.70	7.39	3.41	11.83
6	**1**	1.67	0.01	1.47	4.49	6.50	15.33	10.76	19.13
	2	1.13	-0.58	2.70	4.24	2.33	10.76	7.38	13.26
7	**1**	6.65	-5.66	-2.73	2.11	-	-	-	-
	2	5.55	-6.64	-0.73	1.04	-	-	-	-

4 Discussão

Com o desenvolvimento deste trabalho foi possível compreender melhor as interfaces cérebro-computador com o intuito de se desenvolver um sistema apropriado de estimulação visual por LEDs capaz de evocar o potencial cerebral. Para o melhor funcionamento da BCI é necessário garantir um sinal com potência adequada, exatidão da frequência de estimulação e estabilidade para que o sinal não perca potência ou seja evocado em frequências diferentes das desejadas.

Para os sinais gerados por meio do gerador de funções (Tabelas 4 e 5), não se obteve uma diferença muito relevante na exatidão e estabilidade entre os dois tipos de ondas de alimentação dos LEDs (quadrada e senoidal), indicando uma exatidão próxima a desejada em todas as faixas de frequência testadas e uma estabilidade maior para frequências mais baixas, com largura de banda crescente para altas frequências.

Os testes com fotodiodo (Tabela 6) seguiram a mesma linha, apresentando uma precisão adequada para todas as frequências e maior estabilidade nas baixas frequências, alcançando uma variação de até 6.15 Hz na frequência de estimulação de 100 Hz. A alta exatidão obtida pelo fotodiodo provavelmente se deve ao fato de que o circuito não possui um elemento que pudesse causar uma distorção no sinal, como o filtro capacitivo e o LED. O fotodiodo atuou apenas como uma chave de liga/desliga.

O circuito estimulador desenvolvido mostrou ser uma plataforma capaz de gerar o estímulo visual com intensidade e exatidão adequados, além de ser de fácil implementação (REJER; ZAWISLAK, 2017) e barato. Pelas Figuras 11, 15 e 18, que mostram a FFT dos sinais, é possível observar que o pico do potencial ocorre bem próximo da frequência estipulada em todos os casos, porém as ondas quadradas são mais precisas que as senoidais. Se verificou uma maior precisão e estabilidade para as baixas frequências (<20 Hz), tanto no caso de alimentação por onda quadrada quanto senoidal. Para frequências entre 20 e 40 Hz os resultados também foram satisfatórios em termos de estabilidade e exatidão, apesar de ocorrer um aumento da largura de banda, o que faz com que o sinal fique variando em frequências próximas. Já para frequências maiores (>40 Hz), o Arduino Mega conseguiu garantir melhor estabilidade para sinais gerados por onda senoidal.

O gerador de função resultou em uma exatidão um pouco melhor para as altas frequências, porém a comparação mostrou uma resposta semelhante ao Arduino, o que mostra que o sistema mais simples teve um resultado semelhante em comparação a um sistema mais sofisticado.

Para todos os casos, exceto os realizados com fotodiodo, a potência do sinal senoidal foi menor em relação ao sinal de onda quadrada. Pelo sinal captado pelo fotodiodo, não

ocorreu diferença na potência dos sinais, porque a amplitude dos sinais representa a potência da alimentação do fotodiodo e não a potência dos sinais de estimulação visual, sendo assim não é possível realizar uma comparação direta entre a potência das ondas quadrada e senoidal por esta abordagem. Entretanto os resultados de estabilidade e exatidão se mostraram semelhantes.

A Tabela 7 foi construída com os valores evocados e captados pelo eletrodo do lóbulo occipital posicionado no ponto Oz [1]. Um sinal detectável pela BCI é um sinal que apresenta potência SNR positiva, quanto maior o valor melhor é a diferença entre o sinal evocado e ruídos locais.

No primeiro cenário de teste, a voluntária recebeu estímulos de uma LED vermelho excitado por uma onda senoidal. Na primeira sessão, foi possível observar que para 6 Hz a BCI teve uma resposta boa, já para as frequências de 10, 15 e 20 Hz, o sinal de estimulação ficou imerso em ruídos. Para as altas frequências testadas, o sinal foi evocado de forma excelente, em especial o estímulo em 70 Hz apresentou as melhores taxas de SNR. A sessão 2 apresentou resultados semelhantes para as frequências de 6 e 10 Hz, entretanto houve uma melhora para as frequências de 15 e 20 Hz. Nas altas frequências, o sinal de 40 Hz não evocou muito bem e o sinal de 70 Hz continuou sendo o mais potente.

No segundo cenário, o LED vermelho foi excitado por uma sinal de onda quadrada. Na primeira coleta, o sinal apresentou bons resultados para 6, 15 e 20 Hz, porém em 10 Hz o sinal ficou imerso em ruído. Para as altas frequências, o sinal apresentou bons resultados para todas as frequências, o SNR foi melhor para 70, 80 e 90 Hz. A segunda coleta apresentou resultados semelhantes em relação a primeira. Porém o sinal de 20 Hz ficou indetectável e o sinal de 40 Hz apresentou uma potência muito baixa.

No cenário 3 foi trocado a cor de estimulação para verde. Os estímulos gerados pela excitação senoidal foram capazes de gerar um sinal com SNR detectável em todas as frequências na primeira coleta. O sinal em 70 Hz novamente apresentou melhor potência. Na segunda coleta, todas as frequências também tiveram bons valores de SNR, porém com menor potência que a primeira coleta, exceto as frequências de 10 e 15 Hz, que ficaram melhores.

No quarto cenário, o LED verde foi excitado por um sinal de onda quadrada. A BCI foi capaz de detectar todas as frequências nas duas coletas, porém a primeira teve um melhor resultado. A frequência de 10 Hz teve uma potência muito boa nesse cenário, se comparada com os demais cenários. Novamente as altas frequências apresentaram melhores taxas de SNR.

No cenário 5 foram utilizados LEDs de cor azul excitados por onda senoidal. Na primeira coleta, os resultados foram satisfatórios para todas as frequências. Já na segunda

[1] A Figura 1 mostra o posicionamento dos eletrodos.

coleta, a potência ficou bem mais fraca, e para a frequência de 10 Hz o sinal ficou imerso em ruído.

Ao ser excitado por uma onda quadrada, o LED azul não evocou o sinal tão bem quanto a onda senoidal nas frequências de 6, 10 e 15 Hz. Porém em 20 Hz e nas altas frequências o desempenho foi melhor, chegando a ser até 3 dB mais potente, como no caso da frequência de 90 Hz. A única ocorrência em que o sinal não teve potência suficiente para se destacar dos ruídos foi em 10 Hz na segunda coleta.

Os testes realizados com o estimulação via monitor não permitiam estimulação acima de 30 Hz, desta forma só foi possível comparar os sinais em altas frequências. O sinal de estimulação em 6 Hz teve uma resposta muito boa, sendo melhor que qualquer estimulação gerada pelos LEDs, tanto na coleta 1 quanto na 2. O sinal de 20 Hz foi detectado, porém no mesmo nível que o obtido por LED. Já para os sinais de 10 e 15 Hz, o sinal evocado foi fraco, sendo que em 10 Hz o resultado foi o pior obtido considerando todos os cenários.

Em particular, a frequência de 10 Hz não apresentou ser adequada para estimulação da voluntária, isso pode variar de pessoa para pessoa. O LED vermelho apesar de ter conseguido evocar o potencial nas altas frequências, ficou com o SNR muito baixo em frequências menores. Era esperado ter um melhor resultado com o LED azul no cenário de excitação por onda senoidal, dado que a tensão de corte de um LED azul é mais próxima da metade da tensão de alimentação, assim o tempo em que o LED fica acesso seria o mesmo que o tempo que fica apagado, e apesar de se ter obtido bons resultados, eles não foram os que geraram a melhor estimulação. Além disso, a voluntária destacou uma maior fadiga visual para a cor azul, o que pode ter feito com que ela piscasse mais os olhos, interferindo na qualidade do sinal.

O LED verde foi o que apresentou melhor resultado, evocando o potencial com excelente SNR para todas as frequências utilizadas no experimento.

Em relação ao incômodo causado pelas estimulações, a voluntária destacou que a cor verde e o monitor foram os que menos causaram irritação aos olhos.

As duas formas de onda empregadas apresentaram resultados semelhantes, havendo um desempenho ligeiramente superior para as ondas quadradas.

É importante observar também que o procedimento experimental durou cerca de 3 horas (com intervalos de repouso) e os cenários foram realizados na ordem apresentada pela Tabela 1. Os resultados finais podem ter sofrido um prejuízo causado pelo cansaço da voluntária.

5 Conclusão

As BCI-SSVEP necessitam de um sistema de estimulação visual preciso e versátil. Neste trabalho foi desenvolvido um circuito com 4 LEDs que cintilam em frequências ajustáveis no range de 1 a 100 Hz. Este circuito também permite controlar a intensidade do brilho, o posicionamento espacial e as cores dos LEDs. A alimentação dos LEDs pode ser realizada por meio de ondas quadradas ou senoidais, isto impacta na forma como ocorre a cintilação e estimulação visual do indivíduo, flexibilizando o emprego deste sistema para estudos e aplicações diversas.

O controle do circuito foi realizado por meio de um Arduino Mega, que forneceu resultados satisfatórios quando comparados aos padrões gerados pelas ondas de um gerador de funções. Os sinais apresentaram potência, estabilidade e precisão semelhantes para todos os casos. O sinal quadrado apresentou um potencial de estímulo elevado para todas as frequências, porém ocupou uma largura de banda maior. A onda senoidal apresentou uma intensidade menor, quando comparada à onda quadrada, porém com uma largura de banda mais estreita, principalmente nas altas frequências. Ambos os tipos de onda se mostraram aplicáveis para estimular visualmente os indivíduos que controlam uma BCI-SSVEP.

As estimulações por meio de LEDs apresentaram resultados favoráveis ao ser analisada a SNR de cada cenário. Apenas na frequências de 6 Hz o monitor mostrou um desempenho superior, nas demais frequências os LEDs se equipararam ao monitor ou mostraram resultados melhores. Uma das vantagens previamente abordada que o LED possui, era a possibilidade de se usar frequências acima da taxa de atualização do monitor, e os LEDs tiveram excelentes resultados referentes às altas frequências.

No experimento realizado, todas as cores testadas (vermelho, verde e azul) deram bons resultados, sendo que o LED de cor verde, excitado por um sinal de onda quadrada, foi o que apresentou o melhor resultado.

Referências

ANDERSEN, S. K.; MULLER, M. M. Driving steady-state visual evoked potentials at arbitrary frequencies using temporal interpolation of stimulus presentation. *BMC Neuroscience*, v. 16, n. 1, p. 95, Dec 2015. ISSN 1471-2202.

BAKARDJIAN, H.; TANAKA, T.; CICHOCKI, A. Optimization of SSVEP brain responses with application to eight-command Brain-Computer Interface. *Neuroscience Letters*, v. 469, n. 1, p. 34 – 38, 2010. ISSN 0304-3940.

BYCZUK MARCIN & PORYZALA, P. . M. A. On diversity within operators EEG responses to LED-produced alternate stimulus in SSVEP BCI. Bulletin of the Polish Academy of Sciences: Technical Sciences, p. 447–453, 2012.

CHANG, M. et al. An amplitude-modulated visual stimulation for reducing eye fatigue in SSVEP-based brain-computer interfaces. v. 125, 12 2013.

ECKHORN, R. Oscillatory and Non-Oscillatory Synchronizations in the Visual Cortex of Cat and Monkey. In: _____. *Oscillatory Event-Related Brain Dynamics*. Boston, MA: Springer US, 1994. p. 115–134. ISBN 978-1-4899-1307-4.

FAZEL-REZAI, R. et al. P300 brain computer interface: Current challenges and emerging trends. *Frontiers in neuroengineering*, v. 5, p. 14, 07 2012.

GAO, X. et al. A BCI-based environmental controller for the motion-disabled. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, v. 11, n. 2, p. 137–140, June 2003. ISSN 1534-4320.

GRAIMANN, B.; ALLISON, B.; PFURTSCHELLER, G. Brain–Computer Interfaces: A Gentle Introduction. In: _____. *Brain-Computer Interfaces: Revolutionizing Human-Computer Interaction*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 1–27. ISBN 978-3-642-02091-9.

GRAY, C. M. et al. Oscillatory Responses in Cat Visual Cortex Exhibit Inter-Columnar Synchronization Which Reflects Global Stimulus Properties. *Nature*, v. 338, p. 334–337, 1989.

HERRMANN, C. S. Human EEG responses to 1-100 Hz flicker: resonance phenomena in visual cortex and their potential correlation to cognitive phenomena. v. 137, 04 2001.

HWANG, H.-J. et al. A new dual-frequency stimulation method to increase the number of visual stimuli for multi-class SSVEP-based brain-computer interface BCI. *Brain Research*, v. 1515, p. 66 – 77, 2013. ISSN 0006-8993.

HWANG, H.-J. et al. Development of an SSVEP-based BCI spelling system adopting a QWERTY-style LED keyboard. *Journal of Neuroscience Methods*, v. 208, n. 1, p. 59 – 65, 2012. ISSN 0165-0270.

KLEM, G. H. et al. The ten-twenty electrode system of the International Federation. *Electroencephalogr Clin Neurophysiol*, v. 52, n. 3, 1999.

KUZ, R. et al. On the Quantification of SSVEP Frequency Responses in Human EEG in Realistic BCI Conditions. *PLOS ONE*, Public Library of Science, v. 8, n. 10, p. 1–9, 10 2013.

MARTIšIUS, I.; DAMAšEVIčIUS, R. A Prototype SSVEP Based Real Time BCI Gaming System. *Computational Intelligence and Neuroscience*, p. 15, 2016.

MATERKA, A.; BYCZUK, M. Alternate half-field stimulation technique for SSVEP-based brain-computer interfaces. *Electronics Letters*, v. 42, n. 6, p. 321–322, March 2006. ISSN 0013-5194.

MOULI, S.; PALANIAPPAN, R.; SILLITOE, I. Arduino based configurable LED stimulus design for multi-frequency SSVEP-BCI. 07 2014.

R. KHOSLA A., J. R. S. Influence of stimuli colour in SSVEP-based BCI wheelchair control using support vector machines. J Med Eng Technol, p. 12, 2014.

REJER, I.; ZAWISLAK, T. A low-budget stimulation system for evoking SSVEP built on Arduino/Genuino platform. *Przeglad Elektrotechniczny*, R. 93, nr 1, p. 112–116, 2017.

SANCHEZ, G. et al. Simple communication using a SSVEP-based BCI. *Journal of Physics: Conference Series*, v. 332, p. 012017, 12 2011.

TELLO, R. G. et al. Comparison of the influence of stimuli color on steady-state visual evoked potentials. *Research on Biomedical Engineering*, v. 1, 07 2015.

TENG, F. et al. Square or Sine: Finding a Waveform with High Success Rate of Eliciting SSVEP. v. 2011, p. 364385, 09 2011.

WALDERT, S. Invasive vs. Non-Invasive Neuronal Signals for Brain-Machine Interfaces: Will One Prevail? *Frontiers in Neuroscience*, v. 10, p. 295, 2016. ISSN 1662-453X.

WANG X. GAO, B. H. C. J. Y.; GAO, S. Brain-Computer Interfaces Based on Visual Evoked Potentials. IEEE Eng. Med. Biol., p. 64–71, 2008.

WU, Z. et al. Stimulator selection in SSVEP-based BCI. *Medical Engineering & Physics*, v. 30, n. 8, p. 1079 – 1088, 2008. ISSN 1350-4533. Special Issue (part): Bioengineering in Taiwan.

XIE, S. et al. Stimulator Selection in SSVEP-Based Spatial Selective Attention Study. v. 2016, p. 1–9, 01 2016.

YIJUN, W. et al. Brain-computer interface based on the high-frequency steady-state visual evoked potential. In: *Proceedings. 2005 First International Conference on Neural Interface and Control, 2005.* [S.l.: s.n.], 2005. p. 37–39.

ZHANG, Y. et al. Multiple frequencies sequential coding for SSVEP-Based Brain-Computer Interface. *PLOS ONE*, Public Library of Science, v. 7, n. 3, p. 1–9, 03 2012.

ZHU JORDI BIEGER, G. G. M. D.; AARTS, R. M. A survey of stimulation methods used in SSVEP-based BCIs. Computational Intelligence and Neuroscience, p. 12, 2010.

Apêndices

A Onda senoidal

```
/*  Autor: Breno Alves Navarro
 *
 *   Função de geração de um sinal analógico senoidal
 *  através de uma normalização de uma senóide
 *  interna gerada pela função math.h do arduino.
 *
 *  Necessário usar filtro passa-baixas externo na
 *  saída do PWM.
*/
int pin = 9,pin2 = 10, pin3 = 11,pin4 = 12;    // Alguns dos Pinos com PWM no arduino mega 2560, 11 e 12 são pares que utilizam o mesmo timer assim como 9 e 10
// Alguns dos Pinos com PWM no arduino uno, 11 e 3 são pares que utilizam o mesmo timer assim como 9 e 10

float t=0;
float rad = 0.00084097; // tempo de incrementação calibrado por pontos em radianos;
float led,led2,led3,led4;
int duty,duty2,duty3,duty4;

// frequência do sinais senoidal
float f1=6;  // LED 1
float f2=10;  // LED 2
float f3=15;  // LED 3
float f4=20;  // LED 4

void setup() {

    // Configura a frequência do PWM
        TCCR2B = TCCR2B & 0b11111000 | 0X01;                //
          Altera a frequência do PWM para 31,5 KHz (0x01 =
```

```
            31,5 KHz | 0X02 = 4 KHz) nas portas 11 e 12
28          TCCR1B = TCCR1B & 0b11111000 | 0X01;              //
            Altera a frequência do PWM para 31,5 KHz (0x01 =
            31,5 KHz | 0X02 = 4 KHz) nas portas 9 e 10
29
30      // Seta os pinos como saída
31      pinMode(pin,OUTPUT);
32      pinMode(pin2,OUTPUT);
33      pinMode(pin3,OUTPUT);
34      pinMode(pin4,OUTPUT);
35 }
36
37 void loop() {
38   while(t<200000*PI){
39      float aux = 2*PI*t;                    // Amostragem do
          sinal
40      int a= 128;                            // Auxiliar pra
          realizar a normalização do duty cycle
41      float b=1;                            // Amplitude
          calibrada para corrigir o efeito de corte do LED
42
43      led = a*sin(aux*f1);                   //  Sinal senoidal
          que será modulado do LED 1
44      duty = b*map(led,-a,a,0,255);          //  Normalização do
          duty cycle do PWM 1
45
46      led2 = a*sin(aux*f2);                  //  Sinal senoidal
          que será modulado do LED 2
47      duty2 = b*map(led2,-a,a,0,255);        //  Normalização do
          duty cycle do PWM 2
48
49      led3 = a*sin(aux*f3);                  //  Sinal senoidal
          que será modulado do LED 3
50      duty3 = b*map(led3,-a,a,0,255);        //  Normalização do
          duty cycle do PWM 3
51
52      led4 = a*sin(aux*f4);                  //  Sinal senoidal
          que será modulado do LED 4
```

```
53        duty4 = b*map(led4,-a,a,0,255);       //  Normalização do
              duty cycle do PWM 4
54
55        // Saída de cada pino
56        analogWrite(pin,duty);
57        analogWrite(pin2,duty2);
58        analogWrite(pin3,duty3);
59        analogWrite(pin4,duty4);
60        t = t+rad;  // tempo de incrementação calibrado pra
              amostrar a senoide na frequência mais proxima do
              desejado
61    }
62    t=0; // Zera o tempo instantâneo de amostragem
63 }
```

B Onda quadarada

```
1 /*  Autor Breno Alves Navarro
2  *
3  *   Script de geração de onda quadrada
4  *   a base de interrupção de um temporizador
5  *   do arduino
6  *
7  *   Necessário uma função para gerar a interrupção
8  *   do temporizador. "TimerOne.h"
9  */
10 #include "TimerOne.h" // Função que causa a interrupção
11
12 // Pinos
13 int a = 9,b = 10,c = 11,d = 12;
14
15 // Frequências
16 float f1 = 1; // Frequência em Hz
17 float f2 = 15; // Frequência em Hz
18 float f3 = 30; // Frequência em Hz
19 float f4 = 100; // Frequência em Hz
20
21 // Contadores
22 float c1=0,c2=0,c3=0,c4=0;
23
24 // Tempo de interrupção
25 float t1 = 5000/f1;
26 float t2 = 5000/f2;
27 float t3 = 5000/f3;
28 float t4 = 5000/f4;
29
30 void setup()
31 {
32 pinMode(a, OUTPUT);pinMode(b, OUTPUT);pinMode(c,
     OUTPUT);pinMode(d, OUTPUT);
33 Timer1.initialize(100); // Inicializa o Timer1 e configura
     para um período de 100 us
34 Timer1.attachInterrupt(callback); // Configura a função
     callback() como a função para ser chamada a cada
```

```
    interrupção do Timer1
35
36 }
37 void callback()
38 {
39   c1=c1+1; c2=c2+1; c3=c3+1; c4=c4+1; // Contadores auxiliares
40
41   if(c1 >= t1){
42       digitalWrite(a, !digitalRead(a)); // Troca o estado da
           saída (Alto para baixo ou baixo para alto)
43       c1=0;}                            // Zera o contador
44
45   if(c2 >= t2){
46       digitalWrite(b, !digitalRead(b)); // Troca o estado da
           saída (Alto para baixo ou baixo para alto)
47       c2=0;}                            // Zera o contador
48
49   if(c3 >= t3){
50       digitalWrite(c, !digitalRead(c)); // Troca o estado da
           saída (Alto para baixo ou baixo para alto)
51       c3=0;}                            // Zera o contador
52
53   if(c4 >= t4){
54       digitalWrite(d, !digitalRead(d)); // Troca o estado da
           saída (Alto para baixo ou baixo para alto)
55       c4=0;}                            // Zera o contador
56   }
57
58 void loop()
59 {
60 }
```

B.1 TimerOne.h

```
1 /*
2 *  Interrupt and PWM utilities for 16 bit Timer1 on
    ATmega168/328
3 *  Original code by Jesse Tane for http://labs.ideo.com August
    2008
4 *  Modified March 2009 by Jérôme Despatis and Jesse Tane for
    ATmega328 support
5 *  Modified June 2009 by Michael Polli and Jesse Tane to fix a
    bug in setPeriod() which caused the timer to stop
6 *  Modified April 2012 by Paul Stoffregen - portable to other
    AVR chips, use inline functions
7 *  Modified again, June 2014 by Paul Stoffregen - support
    Teensy 3.x & even more AVR chips
8 *
9 *
10 *  This is free software. You can redistribute it and/or
    modify it under
11 *  the terms of Creative Commons Attribution 3.0 United States
    License.
12 *  To view a copy of this license, visit
    http://creativecommons.org/licenses/by/3.0/us/
13 *  or send a letter to Creative Commons, 171 Second Street,
    Suite 300, San Francisco, California, 94105, USA.
14 *
15 */
16
17 #ifndef TimerOne_h_
18 #define TimerOne_h_
19
20 #if defined(ARDUINO) && ARDUINO >= 100
21 #include "Arduino.h"
22 #else
23 #include "WProgram.h"
24 #endif
25
26 #include "config/known_16bit_timers.h"
27
```

```
28 #define TIMER1_RESOLUTION 65536UL  // Timer1 is 16 bit
29
30
31 // Placing nearly all the code in this .h file allows the
     functions to be
32 // inlined by the compiler.  In the very common case with
     constant values
33 // the compiler will perform all calculations and simply write
     constants
34 // to the hardware registers (for example, setPeriod).
35
36
37 class TimerOne
38 {
39
40
41 #if defined(__AVR__)
42 public:
43 //**************************
44 //  Configuration
45 //**************************
46 void initialize(unsigned long microseconds=1000000)
     __attribute__((always_inline)) {
47 TCCR1B = _BV(WGM13);        // set mode as phase and frequency
     correct pwm, stop the timer
48 TCCR1A = 0;                 // clear control register A
49 setPeriod(microseconds);
50 }
51 void setPeriod(unsigned long microseconds)
     __attribute__((always_inline)) {
52 const unsigned long cycles = (F_CPU / 2000000) * microseconds;
53 if (cycles < TIMER1_RESOLUTION) {
54 clockSelectBits = _BV(CS10);
55 pwmPeriod = cycles;
56 } else
57 if (cycles < TIMER1_RESOLUTION * 8) {
58 clockSelectBits = _BV(CS11);
59 pwmPeriod = cycles / 8;
60 } else
```

```
61 if (cycles < TIMER1_RESOLUTION * 64) {
62 clockSelectBits = _BV(CS11) | _BV(CS10);
63 pwmPeriod = cycles / 64;
64 } else
65 if (cycles < TIMER1_RESOLUTION * 256) {
66 clockSelectBits = _BV(CS12);
67 pwmPeriod = cycles / 256;
68 } else
69 if (cycles < TIMER1_RESOLUTION * 1024) {
70 clockSelectBits = _BV(CS12) | _BV(CS10);
71 pwmPeriod = cycles / 1024;
72 } else {
73 clockSelectBits = _BV(CS12) | _BV(CS10);
74 pwmPeriod = TIMER1_RESOLUTION - 1;
75 }
76 ICR1 = pwmPeriod;
77 TCCR1B = _BV(WGM13) | clockSelectBits;
78 }
79
80 //**************************
81 //  Run Control
82 //**************************
83 void start() __attribute__((always_inline)) {
84 TCCR1B = 0;
85 TCNT1 = 0;     // TODO: does this cause an undesired interrupt?
86 resume();
87 }
88 void stop() __attribute__((always_inline)) {
89 TCCR1B = _BV(WGM13);
90 }
91 void restart() __attribute__((always_inline)) {
92 start();
93 }
94 void resume() __attribute__((always_inline)) {
95 TCCR1B = _BV(WGM13) | clockSelectBits;
96 }
97
98 //**************************
99 //  PWM outputs
```

```
100 //****************************
101 void setPwmDuty(char pin, unsigned int duty)
      __attribute__((always_inline)) {
102 unsigned long dutyCycle = pwmPeriod;
103 dutyCycle *= duty;
104 dutyCycle >>= 10;
105 if (pin == TIMER1_A_PIN) OCR1A = dutyCycle;
106 #ifdef TIMER1_B_PIN
107 else if (pin == TIMER1_B_PIN) OCR1B = dutyCycle;
108 #endif
109 #ifdef TIMER1_C_PIN
110 else if (pin == TIMER1_C_PIN) OCR1C = dutyCycle;
111 #endif
112 }
113 void pwm(char pin, unsigned int duty)
      __attribute__((always_inline)) {
114 if (pin == TIMER1_A_PIN) { pinMode(TIMER1_A_PIN, OUTPUT);
      TCCR1A |= _BV(COM1A1); }
115 #ifdef TIMER1_B_PIN
116 else if (pin == TIMER1_B_PIN) { pinMode(TIMER1_B_PIN, OUTPUT);
      TCCR1A |= _BV(COM1B1); }
117 #endif
118 #ifdef TIMER1_C_PIN
119 else if (pin == TIMER1_C_PIN) { pinMode(TIMER1_C_PIN, OUTPUT);
      TCCR1A |= _BV(COM1C1); }
120 #endif
121 setPwmDuty(pin, duty);
122 TCCR1B = _BV(WGM13) | clockSelectBits;
123 }
124 void pwm(char pin, unsigned int duty, unsigned long
      microseconds) __attribute__((always_inline)) {
125 if (microseconds > 0) setPeriod(microseconds);
126 pwm(pin, duty);
127 }
128 void disablePwm(char pin) __attribute__((always_inline)) {
129 if (pin == TIMER1_A_PIN) TCCR1A &= ~_BV(COM1A1);
130 #ifdef TIMER1_B_PIN
131 else if (pin == TIMER1_B_PIN) TCCR1A &= ~_BV(COM1B1);
132 #endif
```

```
133 #ifdef TIMER1_C_PIN
134 else if (pin == TIMER1_C_PIN) TCCR1A &= ~_BV(COM1C1);
135 #endif
136 }
137
138 //*****************************
139 //  Interrupt Function
140 //*****************************
141 void attachInterrupt(void (*isr)())
      __attribute__((always_inline)) {
142 isrCallback = isr;
143 TIMSK1 = _BV(TOIE1);
144 }
145 void attachInterrupt(void (*isr)(), unsigned long
      microseconds) __attribute__((always_inline)) {
146 if(microseconds > 0) setPeriod(microseconds);
147 attachInterrupt(isr);
148 }
149 void detachInterrupt() __attribute__((always_inline)) {
150 TIMSK1 = 0;
151 }
152 static void (*isrCallback)();
153
154 private:
155 // properties
156 static unsigned short pwmPeriod;
157 static unsigned char clockSelectBits;
158
159
160
161
162
163
164 #elif defined(__arm__) && defined(CORE_TEENSY)
165
166 #if defined(KINETISK)
167 #define F_TIMER F_BUS
168 #elif defined(KINETISL)
169 #define F_TIMER (F_PLL/2)
```

```
170 #endif
171
172 public:
173 //****************************
174 //  Configuration
175 //****************************
176 void initialize(unsigned long microseconds=1000000)
      __attribute__((always_inline)) {
177 setPeriod(microseconds);
178 }
179 void setPeriod(unsigned long microseconds)
      __attribute__((always_inline)) {
180 const unsigned long cycles = (F_TIMER / 2000000) *
      microseconds;
181 if (cycles < TIMER1_RESOLUTION) {
182 clockSelectBits = 0;
183 pwmPeriod = cycles;
184 } else
185 if (cycles < TIMER1_RESOLUTION * 2) {
186 clockSelectBits = 1;
187 pwmPeriod = cycles >> 1;
188 } else
189 if (cycles < TIMER1_RESOLUTION * 4) {
190 clockSelectBits = 2;
191 pwmPeriod = cycles >> 2;
192 } else
193 if (cycles < TIMER1_RESOLUTION * 8) {
194 clockSelectBits = 3;
195 pwmPeriod = cycles >> 3;
196 } else
197 if (cycles < TIMER1_RESOLUTION * 16) {
198 clockSelectBits = 4;
199 pwmPeriod = cycles >> 4;
200 } else
201 if (cycles < TIMER1_RESOLUTION * 32) {
202 clockSelectBits = 5;
203 pwmPeriod = cycles >> 5;
204 } else
205 if (cycles < TIMER1_RESOLUTION * 64) {
```

```
206 clockSelectBits = 6;
207 pwmPeriod = cycles >> 6;
208 } else
209 if (cycles < TIMER1_RESOLUTION * 128) {
210 clockSelectBits = 7;
211 pwmPeriod = cycles >> 7;
212 } else {
213 clockSelectBits = 7;
214 pwmPeriod = TIMER1_RESOLUTION - 1;
215 }
216 uint32_t sc = FTM1_SC;
217 FTM1_SC = 0;
218 FTM1_MOD = pwmPeriod;
219 FTM1_SC = FTM_SC_CLKS(1) | FTM_SC_CPWMS | clockSelectBits |
      (sc & FTM_SC_TOIE);
220 }
221
222 //****************************
223 //  Run Control
224 //****************************
225 void start() __attribute__((always_inline)) {
226 stop();
227 FTM1_CNT = 0;
228 resume();
229 }
230 void stop() __attribute__((always_inline)) {
231 FTM1_SC = FTM1_SC & (FTM_SC_TOIE | FTM_SC_CPWMS |
      FTM_SC_PS(7));
232 }
233 void restart() __attribute__((always_inline)) {
234 start();
235 }
236 void resume() __attribute__((always_inline)) {
237 FTM1_SC = (FTM1_SC & (FTM_SC_TOIE | FTM_SC_PS(7))) |
      FTM_SC_CPWMS | FTM_SC_CLKS(1);
238 }
239
240 //****************************
241 //  PWM outputs
```

```
242 //****************************
243 void setPwmDuty(char pin, unsigned int duty)
      __attribute__((always_inline)) {
244 unsigned long dutyCycle = pwmPeriod;
245 dutyCycle *= duty;
246 dutyCycle >>= 10;
247 if (pin == TIMER1_A_PIN) {
248 FTM1_C0V = dutyCycle;
249 } else if (pin == TIMER1_B_PIN) {
250 FTM1_C1V = dutyCycle;
251 }
252 }
253 void pwm(char pin, unsigned int duty)
      __attribute__((always_inline)) {
254 setPwmDuty(pin, duty);
255 if (pin == TIMER1_A_PIN) {
256 *portConfigRegister(TIMER1_A_PIN) = PORT_PCR_MUX(3) |
      PORT_PCR_DSE | PORT_PCR_SRE;
257 } else if (pin == TIMER1_B_PIN) {
258 *portConfigRegister(TIMER1_B_PIN) = PORT_PCR_MUX(3) |
      PORT_PCR_DSE | PORT_PCR_SRE;
259 }
260 }
261 void pwm(char pin, unsigned int duty, unsigned long
      microseconds) __attribute__((always_inline)) {
262 if (microseconds > 0) setPeriod(microseconds);
263 pwm(pin, duty);
264 }
265 void disablePwm(char pin) __attribute__((always_inline)) {
266 if (pin == TIMER1_A_PIN) {
267 *portConfigRegister(TIMER1_A_PIN) = 0;
268 } else if (pin == TIMER1_B_PIN) {
269 *portConfigRegister(TIMER1_B_PIN) = 0;
270 }
271 }
272
273 //****************************
274 //  Interrupt Function
275 //****************************
```

```
276 void attachInterrupt(void (*isr)())
      __attribute__((always_inline)) {
277 isrCallback = isr;
278 FTM1_SC |= FTM_SC_TOIE;
279 NVIC_ENABLE_IRQ(IRQ_FTM1);
280 }
281 void attachInterrupt(void (*isr)(), unsigned long
      microseconds) __attribute__((always_inline)) {
282 if(microseconds > 0) setPeriod(microseconds);
283 attachInterrupt(isr);
284 }
285 void detachInterrupt() __attribute__((always_inline)) {
286 FTM1_SC &= ~FTM_SC_TOIE;
287 NVIC_DISABLE_IRQ(IRQ_FTM1);
288 }
289 static void (*isrCallback)();
290
291 private:
292 // properties
293 static unsigned short pwmPeriod;
294 static unsigned char clockSelectBits;
295
296 #undef F_TIMER
297
298 #endif
299 };
300
301 extern TimerOne Timer1;
302
303 #endif
```

B.2 TimerOne.cpp

```
1  /*
2  *  Interrupt and PWM utilities for 16 bit Timer1 on ATmega168/328
3  *  Original code by Jesse Tane for http://labs.ideo.com August 2008
4  *  Modified March 2009 by Jérôme Despatis and Jesse Tane for
       ATmega328 support
5  *  Modified June 2009 by Michael Polli and Jesse Tane to fix a bug
       in setPeriod() which caused the timer to stop
6  *  Modified Oct 2009 by Dan Clemens to work with timer1 of the
       ATMega1280 or Arduino Mega
7  *  Modified April 2012 by Paul Stoffregen
8  *  Modified again, June 2014 by Paul Stoffregen
9  *
10 *  This is free software. You can redistribute it and/or modify it
       under
11 *  the terms of Creative Commons Attribution 3.0 United States
       License.
12 *  To view a copy of this license, visit
       http://creativecommons.org/licenses/by/3.0/us/
13 *  or send a letter to Creative Commons, 171 Second Street, Suite
       300, San Francisco, California, 94105, USA.
14 *
15 */
16
17 #include "TimerOne.h"
18
19 TimerOne Timer1;              // preinstatiate
20
21 unsigned short TimerOne::pwmPeriod = 0;
22 unsigned char TimerOne::clockSelectBits = 0;
23 void (*TimerOne::isrCallback)() = NULL;
24
25 // interrupt service routine that wraps a user defined function
       supplied by attachInterrupt
26 #if defined(__AVR__)
27 ISR(TIMER1_OVF_vect)
28 {
29 Timer1.isrCallback();
30 }
31
```

```
32 #elif defined(__arm__) && defined(CORE_TEENSY)
33 void ftm1_isr(void)
34 {
35 uint32_t sc = FTM1_SC;
36 #ifdef KINETISL
37 if (sc & 0x80) FTM1_SC = sc;
38 #else
39 if (sc & 0x80) FTM1_SC = sc & 0x7F;
40 #endif
41 Timer1.isrCallback();
42 }
43
44 #endif
```

C Análise de frequência

```
%% Autor: Marcelo Moreira Tiago
%  Modificado em Outubro de 2018 por Breno Alves Navarro
%
%  Função que realiza a transformada de fourier
%  para a análise do sinal no espectro de frequência
%
clc;clear all;close all;
%% INÍCIO

Dados = csvread('Dados.CSV');    % Arquivo contendo os dados do sinal
    captados por um oscilosópio
fr = 1;                          % Frequência do sinal

X = Dados(:,1);                  % Vetor contendo o tempo
Y = Dados(:,2);                  % Vetor contendo a amplitude do sinal
dt = X(2)-X(1);                  % Diferencial do tempo
plot(X,Y)
xlabel('Tempo (s)');
ylabel('Amplitude(V)');
title('Onda quadrada');
axis([0 4/fr 0 2.5])

% Gera o vetor de frequencias
Nfft=32*4096;
f=((0:Nfft/2-1)/dt/Nfft)';

% Realiza a fft
fftpwm=fft(Y,Nfft);
fftpwm=fftpwm(1:Nfft/2,:);

figure
plot(f,abs(fftpwm)/max(abs(fftpwm)))
xlabel('Frequencia (Hz)');
ylabel('Amplitude(V)');
title('FFT Sinal PWM');
axis ([0 2*fr 0 1])
```

D Análise da relação sinal-ruído

```
%{
Autor: Breno Alves Navarro
Modificado em Dezembro 2018

Função que realiza o cálculo local da potência
da relação sinal-ruído da frequência evocada

Parâmetros de entrada:
ff = frequência a ser medida o SNR
canais = número de canais a se gerar os resultados
máx 16 canais
%}

function snr_bci2(ff,canais)

close all
dir1 = 'C:\Users\Breno\Dropbox\Breno (solo)\TCC\Monografia\TCC
   2\Coleta da Naiara\Dados da BCI\LED_Naiara_Verd_alta_quad\';
dir2 = [dir1,'\CAR\'];

if(ff<=20)
freqs = [6, 10, 15, 20]; % Vetor de baixas frequências es estimulação
elseif(ff>20)
freqs = [40, 70, 80, 90]; % Vetor de altas frequências es estimulação
end

n_freqs = length(freqs); % Número de frequências
channels = [1:16]; % Canais que coletaram os estímulos dos eletrodos
nchannels = length(channels); % Número de canais
trials = [1 2]; % Sessões que foram realizadas
n_trials = length(trials); % Número de sessões

% Carrega dados
for f=1:n_freqs
freq=int2str(freqs(f));
for t=1:n_trials
trial=int2str(trials(t));

load(strcat(dir1,'SSVEP_',freq,'_Hz_training_subject_LED_Naiara_Verd_alta_quad
```

```
fulldata = rawData';

media = sum(fulldata)/nchannels;
if trials(t) == 1
for gg=1:nchannels
Xc(gg,:) = fulldata(gg,:) - media;
end
end

if trials(t) == 2
for gg=1:nchannels
Xc2(gg,:) = fulldata(gg,:) - media;
end
end

save(strcat(dir2,'SSVEP_',freq,'_Hz_training_subject_LED_Naiara_monitor_sessio
end

end



%WELCH

if(ff > 20)
for ii = 1:canais
[pxx,f] = pwelch(Xc(ii,:),[],[],[ff-6:0.1:ff+6],256);
ps = sum(pxx(31:91));
pn = sum(pxx)-ps;
SNR = (30*ps)/(11*pn);
SNRdb(ii) = pow2db(SNR);
end

for ii = 1:canais
[pxx2,f] = pwelch(Xc2(ii,:),[],[],[ff-6:0.1:ff+6],256);
ps2 = sum(pxx2(31:91));
pn2 = sum(pxx2)-ps2;
SNR2 = (60*ps2)/(61*pn2);
SNRdb2(ii) = pow2db(SNR2);
end

elseif(ff<=20)
```

```
for ii = 1:canais
[pxx,f] = pwelch(Xc(ii,:),[],[],[ff-2:0.1:ff+2],256);
ps = sum(pxx(16:26));
pn = sum(pxx)-ps;
SNR = (30*ps)/(11*pn);
SNRdb(ii) = pow2db(SNR);
end

for ii = 1:canais
[pxx2,f] = pwelch(Xc2(ii,:),[],[],[ff-2:0.1:ff+2],256);
ps2 = sum(pxx2(16:26));
pn2 = sum(pxx2)-ps2;
SNR2 = (60*ps2)/(61*pn2);
SNRdb2(ii) = pow2db(SNR2);
end
end

disp('Potência da sessão 1') % Print do valor da SNR
SNRdb'

disp('Potência da sessão 2') % Print do valor da SNR
SNRdb2'
end
```

TERMO DE RESPONSABILIDADE

O texto do trabalho de conclusão de curso intitulado ***"Desenvolvimento de um estimulador visual com LEDs para sistemas BCI-SSVEP"*** é de minha inteira responsabilidade. Declaro que não há utilização indevida de texto, material fotográfico ou qualquer outro material pertencente a terceiros sem a devida citação ou consentimento dos referidos autores.

João Monlevade, 17 de dezembro de 2018.

Breno Alves Navarro

MINISTÉRIO DA EDUCAÇÃO
Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Aplicadas
Colegiado do Curso de Engenharia de Elétrica

UFOP

DECLARAÇÃO DE CONFERÊNCIA DA VERSÃO FINAL

Declaro que conferi a versão final a ser entregue pelo aluno **Breno Alves Navarro**, autor do trabalho de conclusão de curso intitulado **Desenvolvimento de um estimulador visual com LEDs para sistemas BCI-SSVEP** quanto à conformidade nos seguintes itens:

1. A monografia corresponde a versão final, estando de acordo com as sugestões e correções sugeridas pela banca e seguindo as normas ABNT;
2. A versão final da monografia inclui a ata de defesa (Anexo IV - apenas verso), a ficha catalográfica e o termo de responsabilidade (ANEXO X -) devidamente assinado.

João Monlevade, 17 de dezembro de 2018.

Prof.ª Dr.ª Sarah Negreiros de Carvalho Leite