UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Leonardo Nunes Parente

Controlador de estimulador neuromuscular utilizando Raspberry Pi

São Carlos

2017

Leonardo Nunes Parente

Controlador de estimulador neuromuscular utilizando Raspberry Pi

Monografia apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Área de concentração: Engenharia de Reabilitação e Sistemas Embarcados

Orientador: Prof. Dr. Alberto Cliquet Junior

São Carlos

2017

Nunes Parente, Leonardo
NP228c Controlador de estimulador neuromuscular utilizando Raspberry Pi / Leonardo Nunes Parente; orientador Alberto Cliquet Junior; coorientador Renato Varoto. São Carlos, 2017.

Monografia (Graduação em Engenharia Elétrica com ênfase em Eletrônica) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2017.

1. estimulação neuromuscular. 2. controlador neuromuscular. 3. linux embarcado. 4. raspberry pi . I. Título.

FOLHA DE APROVAÇÃO

Nome: Leonardo Nunes Parente

Título: "Controlador de estimulador neuromuscular utilizando Raspberry Pi"

Trabalho de Conclusão de Curso defendido e aprovado em 24/11/2017,

com NOTA 10,0 (dez , zero), pela Comissão Julgadora:

Prof. Titular Alberto Cliquet Júnior - Orientador - SEL/EESC/USP

Profa. Associada Liliane Ventura Schiabel - SEL/EESC/USP

Dr. Renato Varoto - Pós-doutorado/ UNICAMP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Rogério Andrade Flauzino

AGRADECIMENTOS

Agradeço primeiramente aos meus pais que me forneceram todo o suporte necessário para minha educação e crescimento pessoal e profissional.

Aos meus amigos e colegas de turma que me ajudaram durante toda essa jornada acadêmica.

Ao Prof. Dr. Henrik Gollee, orientador do meu *individual project* durante meu intercâmbio na Universidade de Glasgow, Escôcia.

À Frank Gouveia e Guilherme Rocha pelo suporte e auxílio.

À Adriano Barissa por auxíliar no projeto ao desenvolver a caixa 3D.

Ao Prof. Dr. Alberto Cliquet Junior, meu orientador, pela orientação e suporte durante o projeto.

Ao Dr. Renato Varoto por todo suporte, atenção e conselhos durante todo o desenvolvimento do trabalho.

RESUMO

No Brasil, há o aumento de casos de lesões medulares que geram incapacidade devida a perda parcial ou total da motricidade e sensibilidade. Esses indivíduos necessitam de auxílio constante para realizarem simples tarefas cotidianas. Deste modo, eles precisam cada vez mais de equipamentos para facilitar a realização dessas tarefas. Um desses equipamentos é o estimulador neuromuscular que utiliza uma técnica de ativação neural, objetivando a obtenção de contrações musculares, mediante a utilização de baixos níveis de corrente. O Laboratório de Biocibernética e Engenharia de Reabilitação (LABCIBER) da Escola de Engenharia de São Carlos - USP desenvolve projetos utilizando estimulação elétrica neuromuscular (EENM) para auxiliar a vida desses indivíduos. Dentro desse contexto, esse projeto propõe desenvolver um controlador para o estimulador neuromuscular desenvolvido pelo LABCIBER utilizando uma Raspberry Pi e uma tela *touchscreen*. O propósito foi obter um sistema mais compacto, autônomo e com uma interface amigável e simples. O controlador foi montado e testado no estimulador com diferentes ensaios e saída medida em um osciloscópio. Os objetivos iniciais foram cumpridos, validados e melhorias foram implementadas durante o projeto. O sistema final se mostrou pronto para ser utilizado em laboratório para estimulação em pacientes que sofreram lesões medulares.

Palavras-chave: Estimulação neuromuscular. Controlador neuromuscular. Engenharia de reabilitação. Sistema embarcado. Raspberry Pi.

ABSTRACT

In Brazil, there is an increase of cases of spinal cord injuries that generates incapacity due to partial or total loss of motor and sensitivity. These individuals need constant help to perform simple daily tasks. Thereby, they increasingly need equipments to facilitate these tasks. One of these devices is the neuromuscular stimulator that uses a neural activation technique, aiming to obtain muscular contractions, through the use of low current levels. The Laboratory of Biocibernetics and Rehabilitation Engineering (LABCIBER) of the School of Engineering of São Carlos - USP develops projects using neuromuscular electrical stimulation (NMES) to develops projects to assist the life of these individuals. Within this context of neuromuscular stimulation, this project proposes to create a controller for the stimulator developed by LABCIBER using a Raspberry Pi and a touchscreen display. The purpose was to achieve a more compact, standalone system with a friendly and simple interface. The controller was assembled and tested on the stimulator with different assays and measured output on an oscilloscope. The initial objectives were met, validated and improvements were implemented during the project. The final system was ready to be used in the laboratory for stimulation on patients who had suffered spinal cord injuries.

Keywords: Neuromusucular stimulation. Neuromuscular controller. Rehabilitation engineering. Embedded system. Raspberry Pi.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CI	Circuito Integrado
CPU	*Central Processing Unit*
CMOS	*Complementary metal-oxide-semiconductor*
CSI	*Camera Serial Interface*
DSI	*Display Serial Interface*
EENM	Estimulação Elétrica Neuromuscular
GPIO	*General Purpose Input/Output*
GPU	*Graphics Processing Unit*
GUI	*Graphical User Interface*
HDMI	*High-Definition Multimedia Interface*
LABCIBER	Laboratório de Biocibernética e Engenharia de Reabilitação
RAM	*Random Access Memory*
SSH	*Secure Shell*
SO	Sistema Operacional
SoC	System-on-a-chip
TTL	*Transistor-Transistor Logic*
USB	*Universal Serial Bus*
V	Volt(s)

SUMÁRIO

1 INTRODUÇÃO

Frequentemente, no Brasil, várias pessoas são vítimas da crescente violência urbana. Acidentes de trânsito e agressões por arma de fogo são exemplos dessa violência que consequentemente contribui para o aumento de casos de lesões medulares que geram incapacidade devida a perda parcial ou total da motricidade e sensibilidade.

A lesão medular é uma injúria às estruturas contidas no canal medular (medula, cone medular e cauda equina), podendo levar a alterações motoras, sensitivas, autonômicas e psico afetivas. Estas alterações se manifestam principalmente como paralisia ou paresia dos membros, alteração de tônus muscular, alteração dos reflexos superficiais e profundos, alteração ou perda das diferentes sensibilidades (tátil, dolorosa, de pressão, vibratória e proprioceptiva), perda de controle esfincteriano, disfunção sexual e alterações autonômicas como vasoplegia, alteração de sudorese, controle de temperatura corporal entre outras (MINISTÉRIO DA SAÚDE, 2013).

Com o avanço tecnológico dos últimos anos, a engenharia de reabilitação apresenta um papel fundamental no tratamento desses tipos de lesões, pois busca proporcionar uma melhoria na saúde e bem estar, tornando os pacientes menos dependentes e mais engajado socialmente. Umas das técnicas da engenharia de reabilitação é a estimulação elétrica neuromuscular (EENM) que utiliza impulsos elétricos para estimular, acelerar e evitar a perda das atividades motoras, proporcionando assim resultados promissores no restabelecimento de força (VAROTO, 2010).

Os sistemas de EENM podem ser divididos em duas categorias, superficial e implantada. Podem ser controladas por corrente ou tensão. Devida a uma menor variação na resistência e uma necessidade por uma consistente contração muscular e repetibilidade, aplicações com corrente são mais comuns em sistemas de EENM com eletrodos implantados. Já os sistemas controlados por tensão são mais comum em estimulação elétrica transcutâneo (superficial) (SHEFFLER; CHAE, 2007).

Os métodos invasivos não são muito interessantes do ponto de vista operacional, devido a uma série de inconveniências, tais como, possíveis quebras de eletrodos e infecções causadas pela abertura na qual é feita a introdução dos mesmos. Já a EENM feita com eletrodos de superfície, como é o caso do estimulador neuromuscular desenvolvido pelo LABCIBER, os eletrodos são colocados na superfície da pele e o sinal induz linhas de campo dentro do membro, de forma que os íons de sódio, localizados externamente à membrana do nervo motor, sofram um influxo súbito para dentro do nervo, gerando o potencial de ação (MANHÃES, 2004). Esta perturbação se propaga pelo axônio até a fenda sináptica e o músculo, então, é contraído. Sendo assim, executando a estimulação em

musculos/nervos específicos, de maneira controlada e cíclica e tomando certas precauções é possível realizar a marcha em pacientes com lesão medular em laboratório. Usando esta técnica, o paciente muda novamente sua condição e passa a realizar, de forma artificial, os movimentos que havia perdido com a lesão (CLIQUET, 1993).

1.1 Motivação

A maior motivação desse projeto é o intuito de aplicar a engenharia e conhecimentos desenvolvidos para o bem do próximo. Dentro da engenharia elétrica, a engenharia biomédica é a que mais tem contato direto com esse lado humano, pois os equipamentos médicos são desenvolvidos devido a uma necessidade dos pacientes.

Dentro do área de engenharia aplicada à medicina, o ramo de reabilitação exerce uma missão importante ao utilizar a tecnologia para auxiliar o paciente a recuperar sua independência.

1.2 Organização do trabalho

Este trabalho visa apresentar a construção de um controlador de um estimulador neuromuscular utilizando sistema embarcado. O mesmo está organizado em cinco capítulos. O primeiro capítulo é composto por esta introdução que situa o trabalho e sua motivação. No segundo capítulo será realizada a fundamentação teórica dos principais equipamentos e conceitos envolvidos no desenvolvimento do controlador. No terceiro capítulo será apresentado todos os materiais utilizados no projeto, a programação do dispositivo e as conexões dos componentes para formar o controlador. No quarto capítulo serão relatados e discutidos os testes realizados. Por fim, no quinto capítulo são feitas as considerações finais, bem como sugestões para trabalhos futuros.

2 EMBASAMENTO TEÓRICO

Para o entendimento completo do trabalho e de seu desenvolvimento, se faz necessário a introdução e contextualização de alguns temas que serão abordados.

2.1 Linux embarcado

Em termos gerais, Linux é o *kernel* de um sistema operacional elaborado sob o modelo de desenvolvimento e distribuição *open-source* e gratuito, o que contribuiu imensamente para sua popularização. O que define o sistema é o núcleo Linux, que foi desenvolvido por Linus Torvalds em 1991.

Kernel é o código responsável pelo núcleo do sistema operacional (SO) e é encarregado de fazer a interface direta com o *hardware*, gerenciar a comunicação de periféricos e também a memória do sistema, além de decidir a cada momento qual programa deverá ter acesso à Unidade Central de Processamento do sistema (MOTA, 2012).

Em sistemas embarcados, tipicamente refere-se ao sistema operacional como um Linux embarcado (ou eLinux), que nada mais é do que uma versão do programa adaptada para ser executada em um sistema específico com menor capacidade de processamento, já considerando pacotes e rotinas necessários para se fazer uso total de todos os periféricos e conexões presentes em uma placa de sistema embarcado.

2.2 Raspberry Pi

Os *single-board computers*, ou computadores de placa única em português, se popularizaram no ano de 2014 com o lançamento de computadores de baixo custo e com ótimo desempenho para pequenas aplicações. A Raspberry Pi está entre esses lançamentos.

Essa placa foi desenvolvida pela Fundação Raspberry Pi com o intuito de promover o ensinamento de computação básica para jovens. Esse objetivo foi alcançado e, mais ainda, esse computador está sendo usado para aplicações comerciais. A Figura 1 mostra a Raspberry Pi 3 Modelo B que foi utilizada nesse projeto. As especificações técnicas estão listadas na Tabela 1.

A fundação Raspberry Pi oferece diferentes sistemas operacionais para a placa em seu site oficial. Entretanto, o Raspian é o SO oficial apoiado e desenvolvido pela fundação. Raspian é um sistema operacional gratuito baseado no Debian e otimizado para o *hardware* da Raspberry Pi.

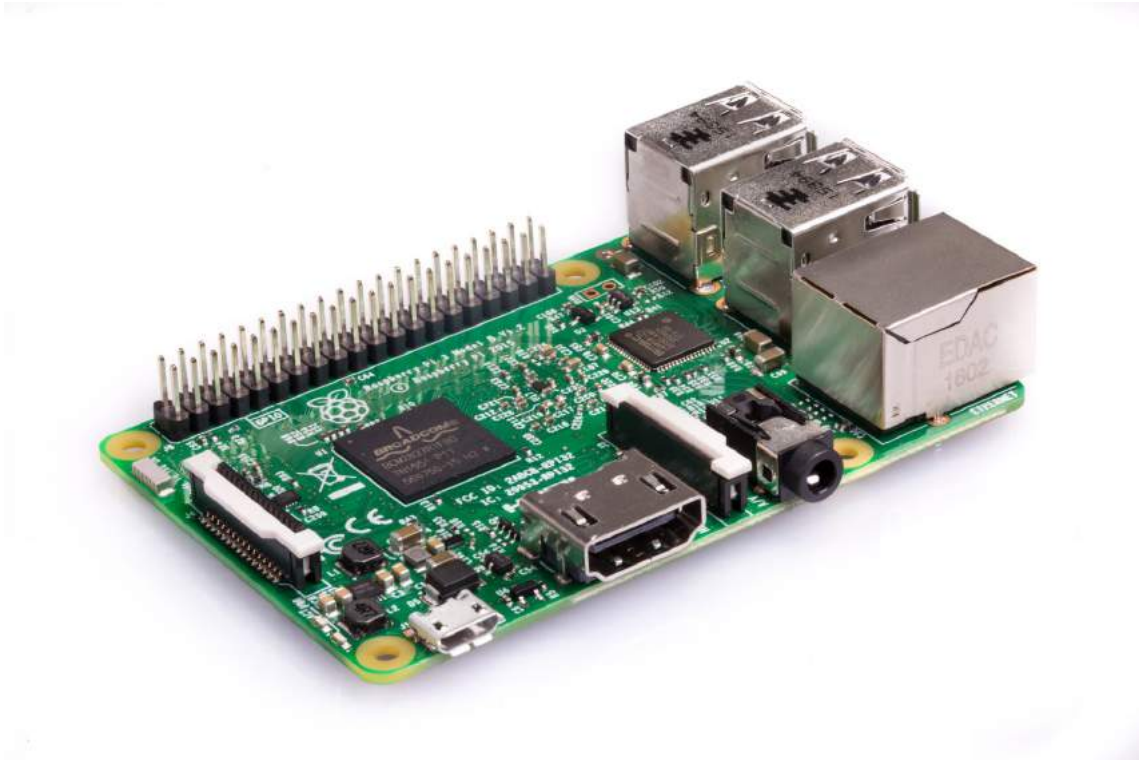

Figura 1: Raspberry Pi 3 modelo B

Fonte: Raspberry Pi Foundation (2017)

2.2.1 GPIO

General Purpose Input/Output (GPIO) são portas programáveis de entrada e saída de dados que são utilizadas para prover uma interface entre os periféricos e os microcontroladores. Na Raspberry Pi, eles estão localizados em uma das bordas da placa em um barramento de 40 pinos, onde 26 são de uso geral, 12 são alimentação ou referência (*ground*) e 2 de endereçamento da EEPROM (uso avançado) como é mostrado na Figura 2.

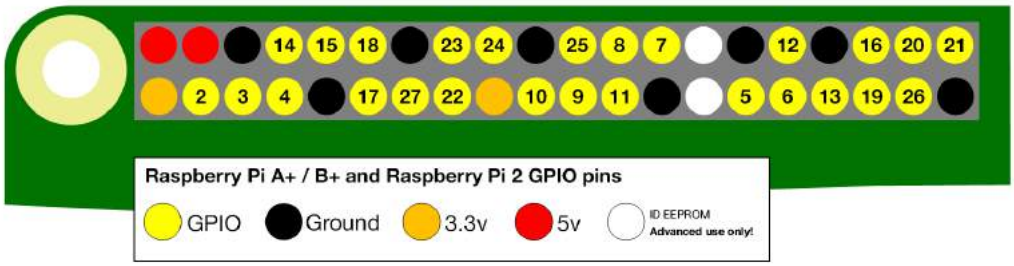

Figura 2: Raspberry Pi 3 - GPIO

Fonte: Raspberry Pi Foundation (2017)

Tabela 1: Especificações Raspberry Pi 3 modelo B

Categoria	Especificação
GPIO	40 pinos
Armazenamento	microSD
SoC	Broadcom BCM2837
CPU	4× ARM Cortex-A53, 1.2GHz
GPU	Broadcom VideoCore IV
RAM	1GB LPDDR2 (900 MHz)
Internet	10/100 Ethernet, 2.4GHz 802.11n wireless
Bluetooth	Bluetooth 4.1 Classic, Bluetooth Low Energy
Portas	HDMI, 3.5mm audio-video jack, 4× USB 2.0, Ethernet, CSI, DSI

Fonte: Raspberry Pi Foundation (2017)

2.2.2 Raspberry Pi touchscreen

A tela oficial da Raspberry Pi utiliza a porta DSI. DSI (Display serial interface) é uma interface serial de alta velocidade baseada em um número de 1 GBits de linhas de dados. A oscilação da tensão das linhas de dados é de apenas 200mV, o que faz com que o ruído eletromagnético criado e o consumo de energia sejam baixos (RASPBERRY PI FOUNDATION, 2017). A Figura 3 apresenta a tela e seus componentes.

Figura 3: Tela *touchscreen* oficial para Raspberry Pi

Fonte: Raspberry Pi Foundation (2015)

A última versão do Raspian já vem com os pacotes da tela *touchscreen*. Portanto, é necessário apenas conectar a tela na porta DSI da placa e alimenta-lá utilizando sua entrada micro USB 5V.

2.3 Qt

Qt é um *framework* multiplataforma para desenvolvimento de interfaces gráficas em C++ para computadores, sistemas embarcados e dispositivos móveis. É suportado por Linux, OS X, Windows, Android, iOS, BlackBerry e outras plataformas. O Qt é mantido pelo *Qt Project*, uma iniciativa de software livre envolvendo desenvolvedores individuais e provenientes de empresas como Nokia, Digia Plc e outras (THE QT COMPANY LTD., 2017a). Ele possui seu próprio ambiente de desenvolvimento integrado, o Qt Creator.

2.3.1 *Signals and slots*

Signals and slots, ou sinais e aberturas em português, são utilizados para comunicação entre objetos. Esse mecanismo é a característica central do Qt e provavelmente a razão que o diferencia dos demais *frameworks*. Um *slot* é uma função que é chamada em resposta a um sinal particular (THE QT COMPANY LTD., 2017a). Por exemplo, em uma aplicação, se um usuário clica no botão "fechar" (*signal*), a função "fechar janela" (*slot*) é chamada para fechar o programa. Deste modo, iterações do usuário (clicar, arrastar, soltar) com objetos (botões, barras, menus) são vistas como interrupções pelo sistema.

2.3.2 Qt para linux embarcado

Qt para linux embarcado é um *framework* C++ para interface gráfica de usuário e desenvolvimento de aplicações para sistemas embarcados. As aplicações são escritas diretamente ao *framebuffer*, eliminando a necessidade de se utilizar o modo janela (X11) e, assim, economizando memória de processamento. O Qt oferece diversos *plugins* que podem ser utilizados em sistemas com linux embarcado: EGLFS, LinuxFB, KMS, DirectFB, Wayland (THE QT COMPANY LTD., 2017b).

Para a Raspberry Pi, o Qt disponibiliza um guia para utilizar EGLFS junto com o Raspian. EGLFS é um *plugin* de plataforma que permite rodar uma aplicação do Qt sobre o EGL e OpenGL ES 2.0 sem necessitar de um sistema de janela, como X11 ou Wayland.

EGL é uma interface entre as APIs de renderização Khronos (como OpenGL, OpenGL ES ou OpenVG) e o sistema de janelas da plataforma nativa. A EGL lida com gerenciamento de contexto de gráficos, ligação de superfícies, sincronização de renderização e permite a renderização 2D e 3D com alto desempenho, aceleração e modo misto usando outras APIs Khronos (THE QT COMPANY LTD., 2017b).

2.4 Trabalhos anteriores

Esse projeto é baseado em dois trabalhos anteriores: *Raspberry Pi based system to control neuromuscular stimulation* (PARENTE, 2016) e Aplicação de um sistema de comando por voz e um *software* de controle na engenharia de reabilitação (BARBARINI, 2008).

O primeiro foi desenvolvido pelo autor desse projeto durante seu período de intercâmbio na Universidade de Glasgow, Escócia. O intuído do projeto era criar um controlador para um estimulador neuromuscular comercial utilizando uma Raspberry Pi 2 e sua tela *touchscreen* de 7". A diferença desse projeto para o já realizado é que o trabalho da Escócia utilizava parâmetros aleatórios para verificar a interferência dos mesmo na fadiga muscular e a forma de comunicar com o estimulador era via USB. Já o projeto atual é o contrário, o objetivo é usar o estimulador para tratar pacientes sem causar fadiga muscular e o estimulador utilizado foi desenvolvido no próprio LABCIBER. A Figura 4 mostra o sistema desenvolvido em Glasgow.

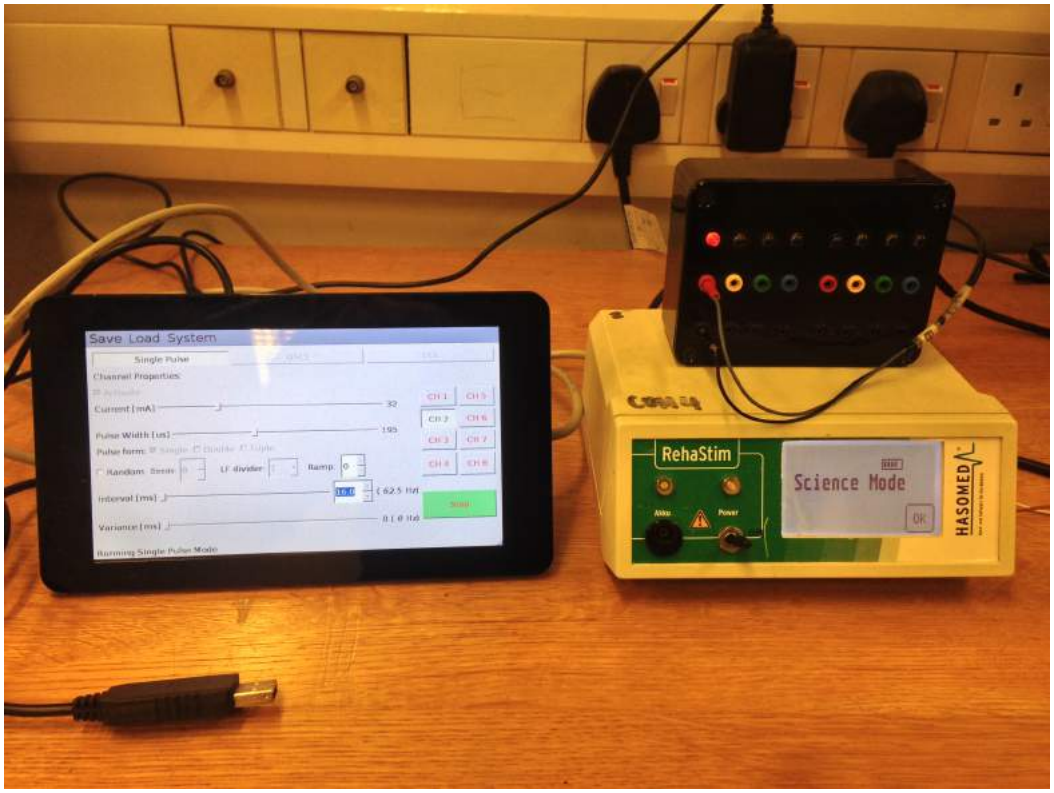

Figura 4: Controlador *touchscreen* e estimulador neuromuscular RehaStim

Fonte: Parente (2016)

Já o segundo trabalho foi desenvolvido sobre o estimulador criado pelo LABCIBER. Uma interface gráfica utilizando labVIEW foi criada para que um computador com sistema operacional Windows possa se comunicar com o estimulador. A Figura 5 apresenta a

interface em LabView. A interface e parâmetros do controlador do trabalho atual foram baseados nessa imagem.

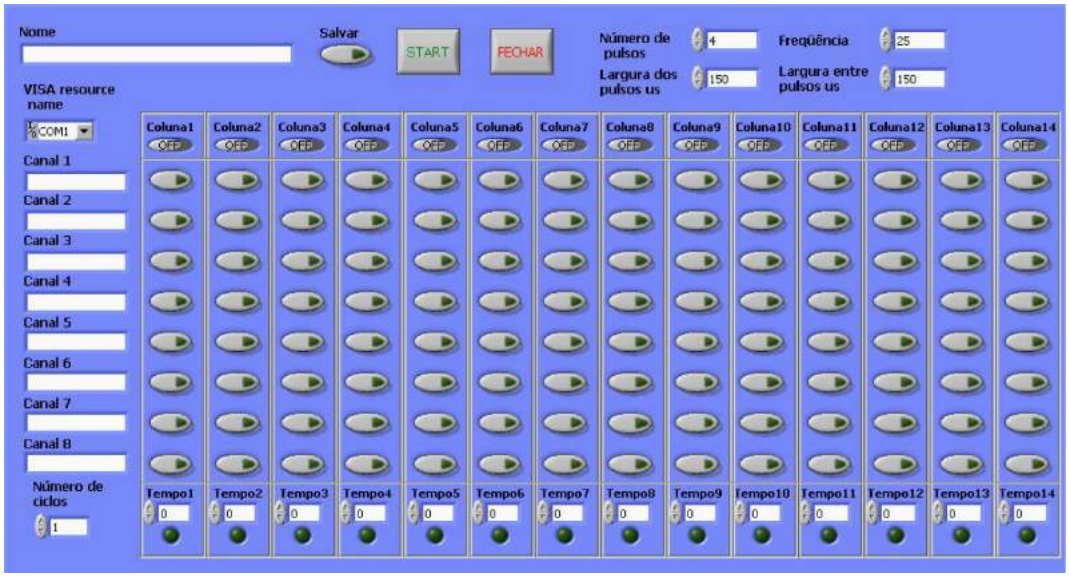

Figura 5: Tela do *software* de controle no labVIEW

Fonte: Barbarini (2008)

3 MATERIAIS E MÉTODOS

Este capítulo discorre sobre o desenvolvimento do projeto em si e os materiais utilizados no mesmo. É a parte mais importante, pois esse trabalho é o início de um novo modo de controlar o estimulador neuromuscular e sua interface e funcionalidades podem ser melhoradas. Portanto, esse capítulo fornece todos os conceitos para entendimento do controlador de estimulação neuromuscular e é escrito de forma a ser um tutorial para futuras modificações no projeto.

3.1 Componentes

Uma das principais vantagens de desenvolver um controlador utilizando um computador de placa única é a questão do custo do projeto total. Além de tornar o sistema mais portátil e com mais autonomia energética.

Todos os componentes utilizados são listados na Tabela 2. A coluna de preço contém o valor pago nos produtos sem considerar frete. Os preços são com base em vendedores que emitem nota fiscal e a pesquisa foi realizada em mais de um vendedor. O enfoque do valor apresentado é para entendimento da média de custo desse projeto, pois esses preços podem variar, e é por isso que esse trabalho não apresenta links nem os nomes de vendedores.

Tabela 2: Componentes utilizados

Componentes	Preço
Raspberry Pi 3 Modelo B	R$ 225,00
Display Raspberry Pi Touchscreen 7"	R$ 550,00
Carregador portátil Pineng PN-999	R$ 103,00
TXS0108E - conversor de nível lógico	R$ 5,00
Botão de 2 pinos com retorno	R$ 4,00
Barra de pinos fêmea 180° Dupla 1x20	R$ 3,00
Barra de pinos fêmea 180° Dupla 2x20	R$ 4,00
Cabo flat colorido 10 vias (metro)	R$ 4,00
2x Conector USB macho	R$ 3,50
2x Conector micro USB macho	R$ 3,50
Box Header 10 vias com travamento	R$ 2,00
Capacitor de 4,7uF	R$ 0,20
Resistor de 39k ohms	R$ 0,10
Total	R$914,30

Além dos componentes listados na Tabela 2, foi necessário um computador com GNU/Linux instalado e um cabo de rede para acessar a Raspberry Pi via *SSH*. Neste caso, foi utilizado a versão 16.04 x64 do sistema operacional Ubuntu.

3.2 Instalação do Sistema Operacional para Raspberry Pi

Antes de iniciar a programação da interface gráfica e do protocolo de comunicação entre a placa e o módulo de estimulação, é necessário a instalação de um sistema operacional, pois a Raspberry Pi é um computador em uma única placa. O sistema escolhido foi o Raspian, que é o SO oficial apoiado pela *Raspberry Pi Foundation*.

A versão utilizada do Raspian foi a *Stretch Lite*, pois não será necessária interface gráfica (*X Window System*) para o sistema operacional, já que o programa de controle do estimulador será embarcado e rodará entre o OpenGL e o modo janela (X11). Essa versão ocupa menos espaço de disco.

Para sua instalação, a Fundação Raspberry Pi fornece um guia de instalação simples que pode ser executado em um computador com Linux, Windows ou Mac. O Tutorial seguido por esse projeto foi utilizando um computador com Sistema Operacional Windows e é explicado a seguir.

a) Acesse o site oficial da Fundação Raspberry Pi: https://www.raspberrypi.org/ (acessado em 08 out. 2017).

b) Vá na aba *Downloads*, selecione o SO Raspian e baixe a versão Lite.

c) Ao término do passo anterior, deve-se obter um arquivo com extensão .img. Baixe o programa Win32DiskImager na página do SourceForge, não é necessário instalação.

d) Conecte um cartão micro SD (recomendado 8GB) ao computador e execute o programa Win32DiskImager. Seleciona a imagem baixada do Raspian e o local do disco SD, clique no botão *Write* e aguarde.

e) Pronto, o cartão SD está com o Sistema Operacional instalado e pronto para uso. Desconecte o cartão do computador e conecte-o à placa.

f) Alimente a Raspberry Pi e veja se o sistema inicializa. Caso haja erro, repita todos os item acima.

Após esse processo, a placa está pronta para uso e já pode ser configurada para receber o programa QSTIMBERRY Controller.

3.3 Configuração da Compilação Cruzada

Foi adotada a compilação cruzada, pois um PC atual possui uma grande capacidade de processamento, que é superior a placa embarcada e também um maior disco rígido para

alocação de dados. Deste modo, toda a programação da interface gráfica e comandos foi feita em um Notebook rodando Ubuntu, utilizando o programa Qt Creator - versão 5.7 e apenas o executável gerado foi enviado para a Raspberry Pi 3.

Primeiramente, foi necessário configurar a placa e o computador para que se comunicassem via *SSH* diretamente por um cabo de rede, essa configuração completa pode ser encontrada no Apêndice A.

Então, é exigido que as bibliotecas essenciais do Qt sejam instaladas na Raspberry Pi para que o executável gerado possa ser executado de modo embarcado (EGLFS). A própria comunidade do Qt fornece um tutorial para sincronizar as dependências da Raspberry Pi 2 com o computador e instalar os componentes necessários, esse tutorial foi traduzido, atualizado para a versão 3 da placa e está detalhado no Apêndice B.

3.4 QSTIMBERRY Controller: funcionalidades e algoritmos

QSTIMBERRY Controller é o programa criado utilizando Qt creator e rodado em uma Raspberry Pi 3 para controlar um estimulador neuromuscular, o que dá origem ao seu nome: Qt + STIMulation + raspBERRY.

A interface gráfica da versão 1.0.0 do programa de controle do estimulador neuromuscular é apresentada na Figura 6. A interface é dividida em três partes principais: barra de menu, *main widget* e barra de status. Cada funcionalidade do *software* será explicada por meio de fluxogramas e algoritmos e não por linhas de código para melhor entendimento. O código em C++ é desenvolvido sobre licença LGPL 3.0, é *open source* e para acesso deve ser requisitado ao LABCIBER.

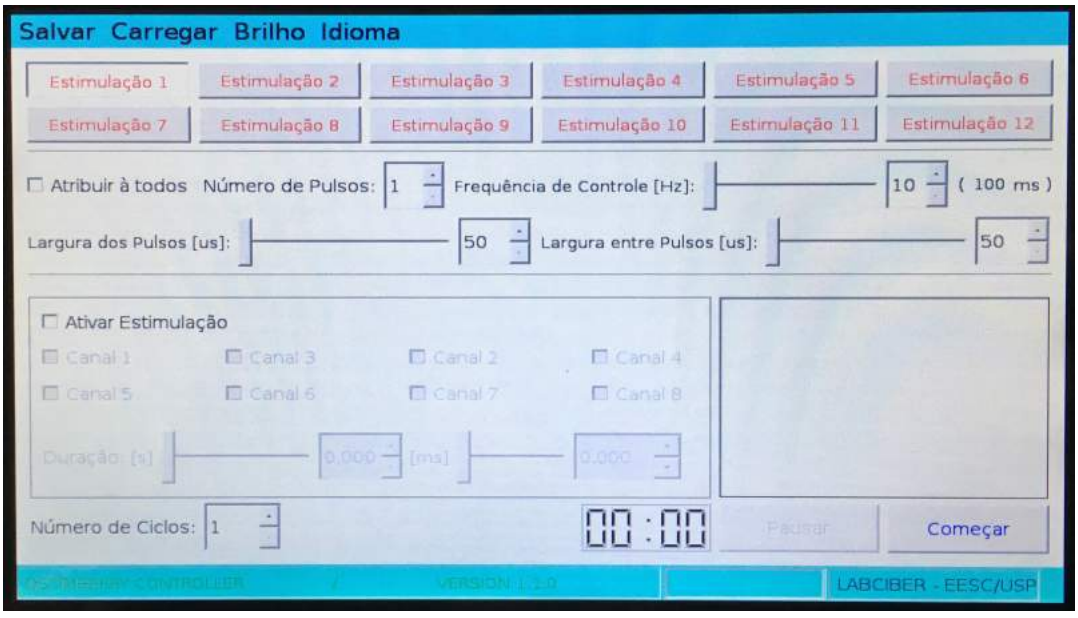

Figura 6: Interface do QSTIMBERRY Controller

O QSTIMBERRY Controller foi baseado no programa do LabVIEW que é utilizado atualmente pelo LABCIBER, adaptado para o modo *touchscreen* e foram adicionadas novas funcionalidades como por exemplo o modo "Atribuir à todos" e Cronômetro.

3.4.1 *Main Widget*

Um *widget* é um componente de uma interface gráfica do usuário (GUI) que pode incluir menus, botões, ícones, barras de rolagens, entre outros componentes. A tradução literal de *main widget* é ferramenta principal, e essa tradução pode ser utilizada nesse projeto, pois esse widget possui todos os componentes necessários para realizar a estimulação neuromuscular. É nesse objeto que o usuário define os parâmetros de estimulação, inicia, pausa e finaliza o processo. A programação dessa parte do software é a mais complexa e será explicada com mais detalhes.

A parte de cima do *widget* apresenta 12 botões para seleção de estimulação. Cada uma das doze estimulações possui 16 parâmetros que estão listados a seguir:

- Os 8 primeiros parâmetros: canal N ativado/desativado. Em que N vai de 1 a 8.

- 9°: Segundos de estimulação.

- 10°: Milissegundos de estimulação.

- 11°: índice de estimulação ativada.

- 12°: Número de pulsos.

- 13°: Largura dos pulsos.

- 14°: Largura entre pulsos.

- 15°: Frequência de controle.

- 16°: Número da Estimulação.

Exitem dois modos de estimulação e quem determina qual modo está acionado é o botão "Atribuir à todos". Caso o botão não esteja selecionado, os parâmetros de 12 a 15 podem ser diferentes para cada uma das 12 estimulações e quando o usuário clica no botão "Começar" toda os componentes são desativados com exceção dos botões "Pausar" e "Parar". Caso contrário, os parâmetros de 12 a 15 são setados iguais para todas as 12 estimulações, assim, o usuário apenas individualiza os canais e a duração de estimulação de cada uma das 12 estimulações. Porém, quando o botão "Começar" é pressionado, os componentes que determinam esses parâmetros ficam ativados durante as estimulações, podendo ser alterados pelo usuário. Isso pode auxilar o profissional que está utilizando o estimulador à encontrar parâmetros mais ideais dinamicamente.

Foi implementado a desativação dos objetos da tela para evitar qualquer erro do usuário enquanto a estimulação está ocorrendo e, desse modo, o sistema se dedica realizar

a estimulação sem interrupções.

Os parâmetros indicados na Figura 7 podem ser alterados tanto pela *slider* quanto pelo *spin box*, pois esses componentes estão conectados e ao alterar um, o outro é alterado para o mesmo valor automaticamente.

Figura 7: *Slider* e *spin boxes*

Há quatro componentes que são gerais para todas as estimulações: botão “Começar”, botão “Pausar”, número de ciclos e cronômetro. O número de ciclos determina quantas vezes o processo de todas as estimulações ativas serão repetidos em único início. Já o cronômetro conta todo o tempo decorrido até o termino total da estimulação. Esse componente só é reiniciado quando o usuário inicia uma nova estimulação.

Por exemplo, a “Estimulação 1” está ativa com duração de 5 segundos, a “Estimulação 6” está ativa com duração de 4 segundos e o número de ciclos é igual a 2. Após 9 segundos do início do processo, a “Estimulação 6” termina e a “Estimulação 1” inicia novamente. Assim, o total do processo de estimulação será de 18 segundos e esse valor ficará gravado no cronômetro até o usuário clicar no botão “Começar” novamente como é mostrado na Figura 8.

Ao clicar no botão “Começar”, novas variáveis de estimulação são criadas e recebem as Estimulações e canais ativos, sem alterar as variáveis já existentes. Isso é feito para que quando a estimulação realmente se inicie, o programa não precise varrer parâmetros não ativados, aumentando assim a precisão e eficiência do controlador. Esse processo é explicado no fluxograma da Figura 9.

Os sinais de controle são enviados pelo pinos de uso gerais (GPIO) da Raspberry Pi. Para transformar os dados inseridos pelo usuário em acionamentos dos pinos da placa, é necessário acesso à esses pinos dentro do programa. Para isso é utilizado a biblioteca pigpio. Sua documentação pode ser encontrada em http://abyz.me.uk/rpi/pigpio/ (acessado em

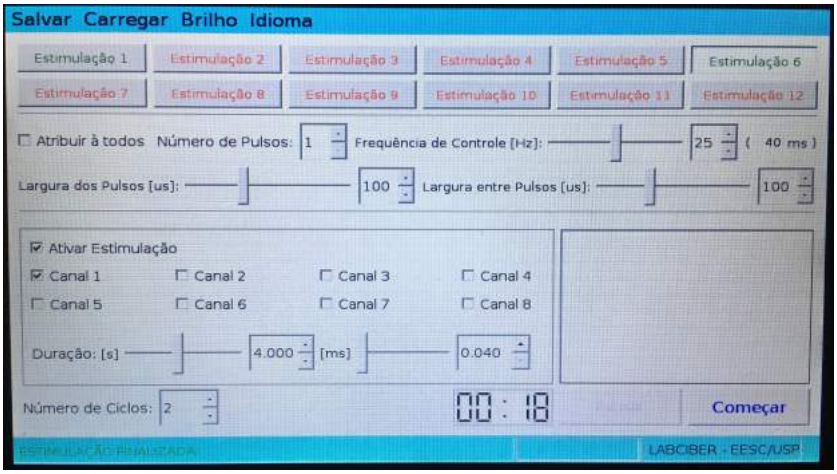

Figura 8: Exemplo de estimulação

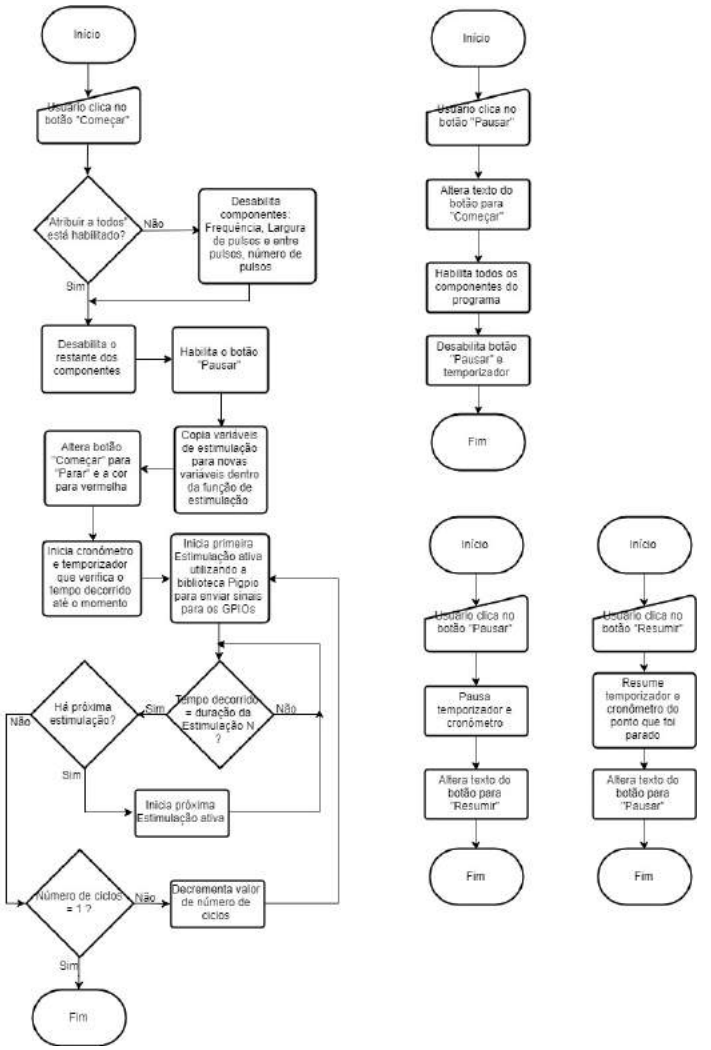

Figura 9: Fluxogramas - botões de estimulação

28 out. 2017). A Figura 10 indica os pinos correspondentes aos canais do estimulador e a numeração de acordo com a Raspberry Pi e a biblioteca pigpio.

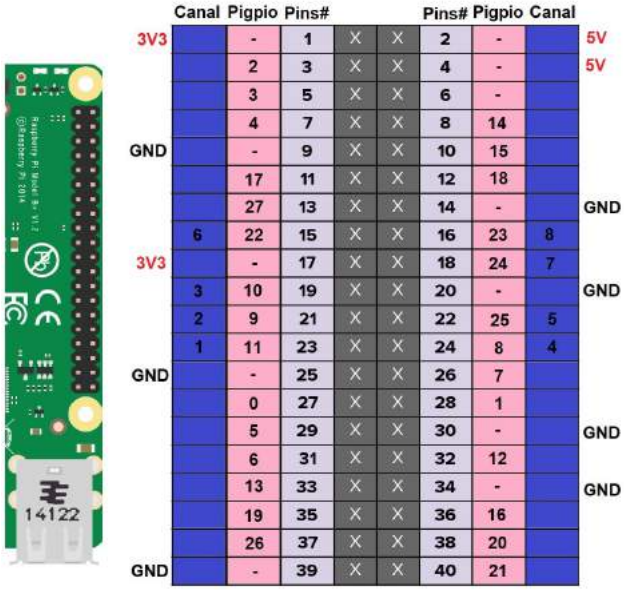

Figura 10: Pinos de controle dos canais de estimulação

3.4.2 Barra de menu

O segundo grupo de funcionalidades a ser apresentado é a barra de menu que é composta por quatro ferramentas: Salvar, Carregar, Brilho e Idioma. A Figura 11 mostra as ações presentes na aba Salvar. O usuário pode salvar os dados atuais de estimulação em um dos quatro Espaços disponíveis. Já a Figura 12 mostra o fluxograma da ação executada quando o usuário clica em um dos espaços.

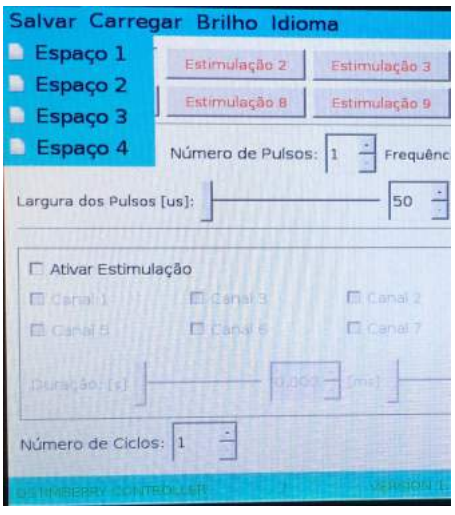

Figura 11: *Menu bar* - aba salvar

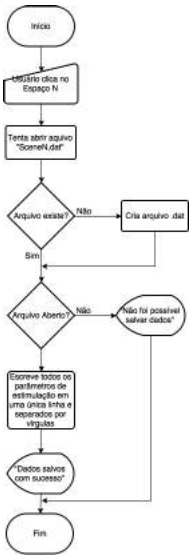

Figura 12: Fluxograma - aba salvar

A aba Carregar possui os mesmos quatro Espaços e uma funcionalidade a mais: o botão para resetar todas as variáveis. A Figura 13 e Figura 14 mostram o funcionamento dessas funções.

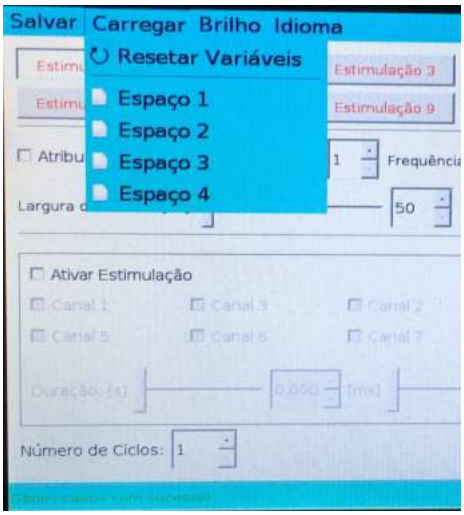

Figura 13: *Menu bar* - aba carregar

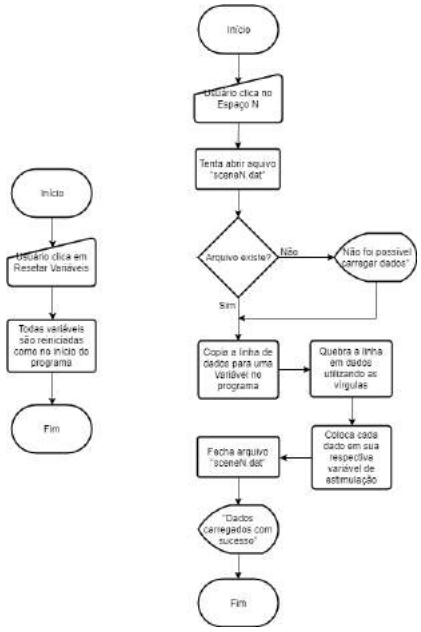

Figura 14: Fluxogramas - aba carregar

O Qt também permite que comandos linux sejam executados dentro do programa gerado. Desse modo, é possível alterar o arquivo responsável por ajustar o brilho da tela *touch*. Para que o programa não ficasse mais complexo, foram definidos 4 níveis de brilho: 25%, 50%, 75% e 100%, como é mostrado na Figura 15. O algoritmo é bem simples: quando o usuário clica em um dos níveis de brilho, o programa executa uma função do sistema que altera o valor do brilho da tela.

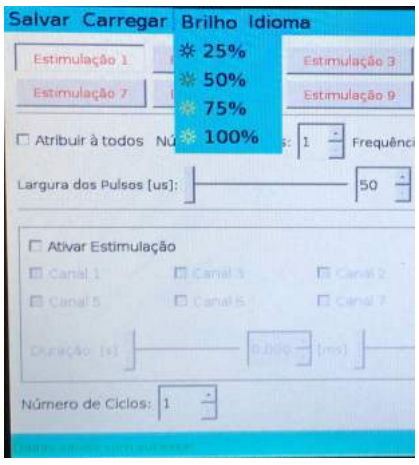

Figura 15: *Menu bar* - aba brilho

A última funcionalidade a barra de menu é a aba de seleção de idioma. O *software* possui dois idiomas: português e inglês. O usuário pode alterar o idioma padrão a qualquer momento e essa escolha fica salva mesmo ao reiniciar o sistema, pois a opção do usuário fica salva em um arquivo .dat. A Figura 16 e Figura 17 apresentam a aba idioma.

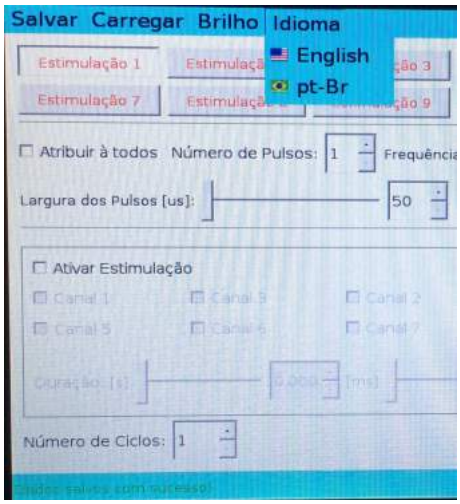

Figura 16: *Menu bar* - aba idioma

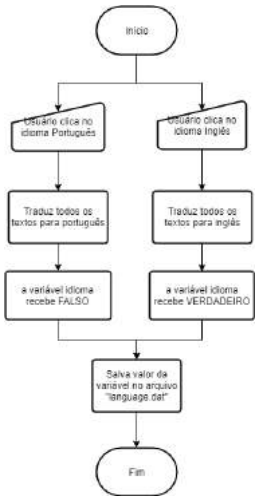

Figura 17: Fluxograma - aba idioma

3.4.3 Barra de status

Por último, a barra de status tem função apenas de auxiliar o usuário com mensagens e indicativos visuais. Quando uma estimulação é iniciada, uma mensagem é mostrada indicando qual das 12 estimulações foi iniciada e seus parâmetros: canais ativos, duração (segundos + milisegundos), frequência, largura de pulso em nível alto, largura de pulso em nível baixo e quantidade de pulsos. Também possui uma barra de progresso que indica a duração percorrida da estimulação atual até o momento.

A Figura 18 exemplifica as informações apresentadas acima. Nota-se que a estimulação tem duração de 6 segundos e decorreram 2 segundos no momento da foto, portanto a barra de progresso é menor que 50%.

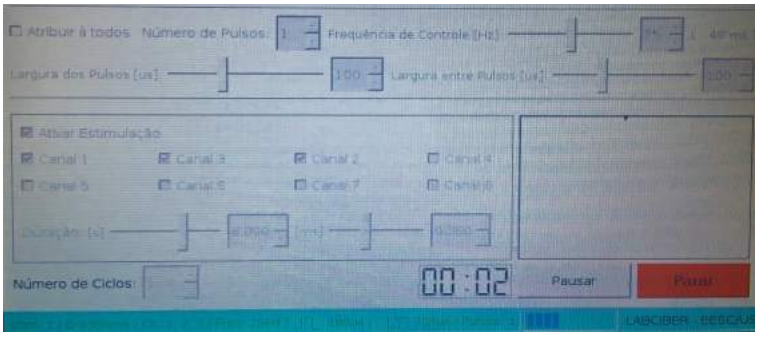

Figura 18: *Status bar* - exemplo de estimulação

A Barra de status também apresenta mensagens quando um arquivo de estimulação é salvo ou carregado. As mensagens são mostradas no idioma selecionado pelo usuário (português ou inglês).

3.5 Hardware: componentes e conexões

O que foi apresentado até esse momento foi a configuração e programação do computador de placa única e do *software* de controle. Porém, para que o controlador se comunique com o estimulador neuromuscular de 8 canais, é necessário o uso de periféricos. Um dos motivos desse uso é porque a entrada do estimulador utiliza TTL (nível lógico 5V) e os GPIOs da Raspberry Pi utilizam CMOS (nível lógico 3,3V). Portanto, é necessário o uso de um conversor de nível lógico.

O circuito integrado TXS0108E foi utilizado, pois possui 8 canais de conversão, o que é exatamente o necessário para a aplicação do projeto (8 canais de estimulação). A Figura 19 mostra o CI já soldado em uma placa para acesso via barra de pinos.

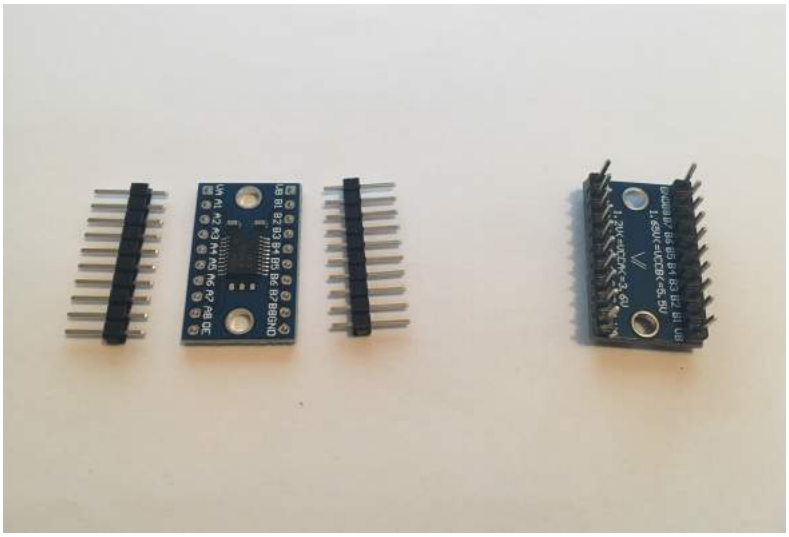

Figura 19: TXS0108E - Conversor de nível lógico

Foi necessário fazer um conector próprio para conectar a entrada do conversor (3,3V) aos pinos de uso geral da Raspberry Pi. A Figura 20 retrata o conector soldado à mão, já a Figura 21 demonstra o esquemático das conexões entre a placa e o módulo conversor.

Na saída do TXS0108E também foi feito outro conector próprio com um *box header* com travamento na saída para que um cabo flat proveniente do estimulador seja conectado e receba sinais de controle com nível de 5V. A Figura 22 retrata o conector de saída do controlador e a Figura 23 apresenta o esquemático do conector.

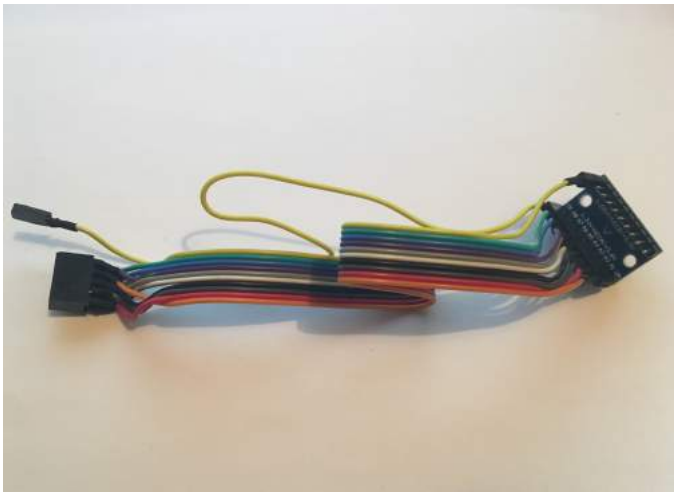

Figura 20: Conector GPIOs - TXS0108E

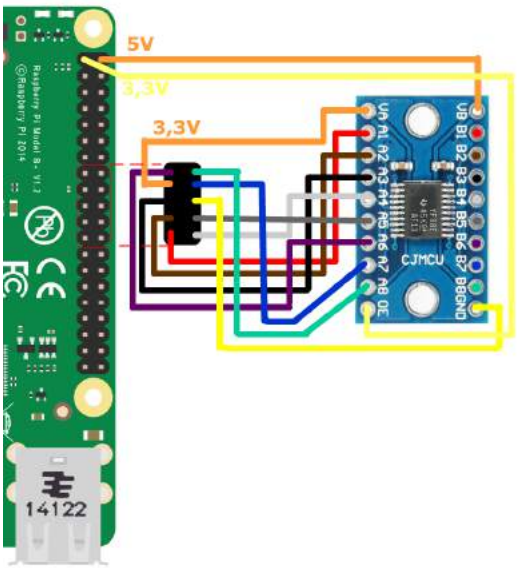

Figura 21: Esquemático conector GPIOs - TXS0108E

Para alimentar o sistema, foi escolhido uma bateria portátil para carregar dispositivos móveis, mais conhecido como *power bank*. A marca escolhida foi o da Pineng e o modelo é o PN-999, como é apresentado na Figura 24. Esse dispositivo possui capacidade de 20000mAh (para 3,7V), duas saídas USB: uma 5V/2,1A e outra 5V/1A. Também possui um display que mostra o nível de bateria atual, um botão para ligar e desligar o *power bank* e uma entrada micro USB 5V/2A para carregar a bateria.

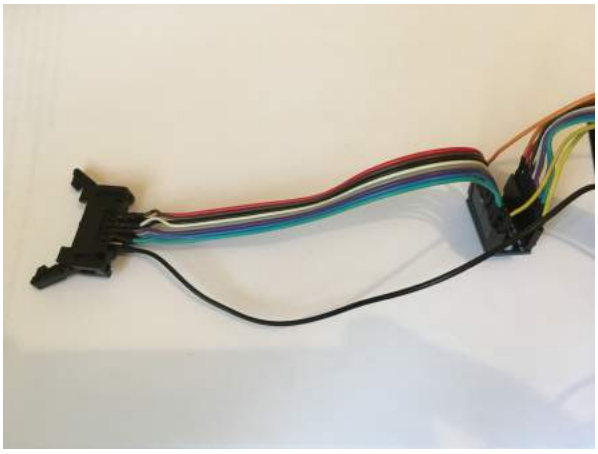

Figura 22: Conector TXS0108E - saída para estimulador

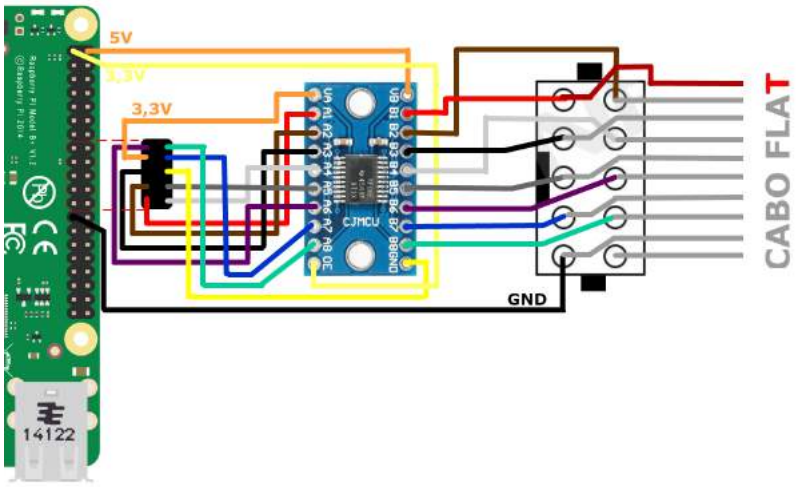

Figura 23: Esquemático conector TXS0108E - saída para estimulador

Figura 24: *Power bank* Pineng PN-999

Como a alimentação da tela e da placa são via micro USB, foi necessário montar dois conectores USB macho para micro USB macho, que são apresentados na Figura 25. Já a Figura 26 mostra os cabos atrelados ao carregador portátil. Fita isolante líquida foi aplicada nos conectores micro USB para evitar curto.

Figura 25: Cabos para alimentação

Figura 26: Carregador portátil com cabos conectados

Um botão foi adicionado ao sistema para desligar/ligar o *backlight* da tela durante a execução do programa, e assim, dar uma maior autonomia ao sistema. O botão utilizado é de 2 terminais, duas posições e com retorno. Como qualquer outra chave mecânica, esse componente apresenta o efeito *bouncing*, que é a oscilação do sinal de acionamento quando a tecla é pressionada. Um dos meios de evitar esse efeito é utilizar um circuito com resistor e capacitor como é mostrado no esquemático da Figura 27. Foi utilizada uma placa perfurada para soldar os componentes e uma barra de pinos para tornar o circuito modular. A Figura 28 mostra o botão conectado ao circuito *debouncing*.

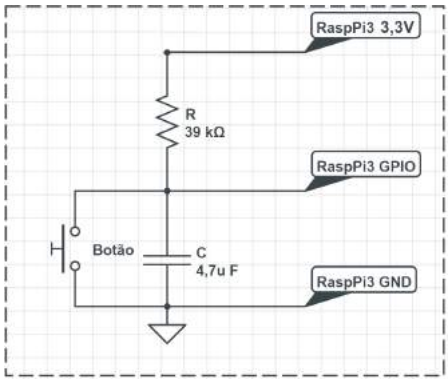

Figura 27: Esquemático circuito *de-bouncing*

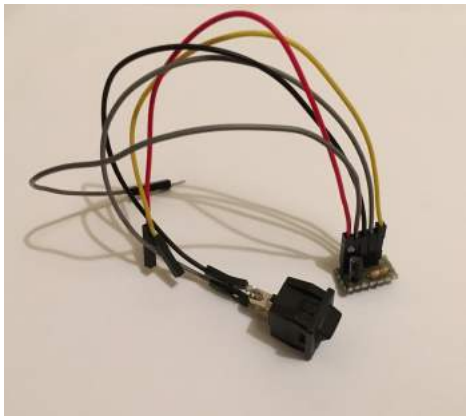

Figura 28: Botão para ligar/desligar o *backlight* da tela

Por fim, todos os componentes foram conectados para formar o controlador de estimulação completo e funcional. A Figura 29 apresenta as conexões do controlador em funcionamento.

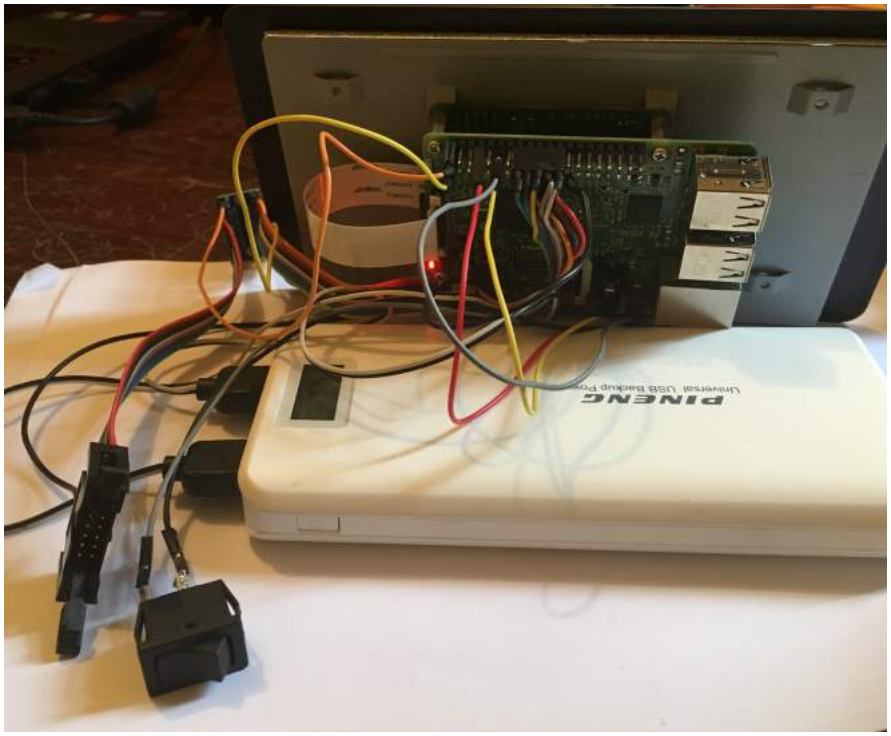

Figura 29: Controlador e suas conexões

3.5.1 Impressão 3D

Após definir todos os componentes eletro-eletrônicos do controlador de estimulação neuromuscular, é necessário projetar a parte mecânica a qual comportará o sistema. A caixa do QSTIMBERRY foi desenvolvida pelo projetista e designer Adriano Ordoz Barissa

com base no hardware da Figura 29 utilizando o *software SolidWorks*. Em seguida, o desenho 3D foi enviado para impressão na impressora 3D Lulzbot TAZ 6. O material utilizado foi a acrilonitrila butadieno estireno, cuja sigla ABS deriva da forma inglesa *acrylonitrile butadiene styrene*.

A impressão da caixa foi divida em três partes: a parte principal, em que a tela *touchscreen* é acoplada, que demorou 26 horas de impressão, a tampa inferior que demorou 8 horas e 30 minutos, e a tampa traseira, cuja impressão durou 3 horas e 30 minutos. A Figura 30 apresenta o caixa montada com os componentes já inseridos.

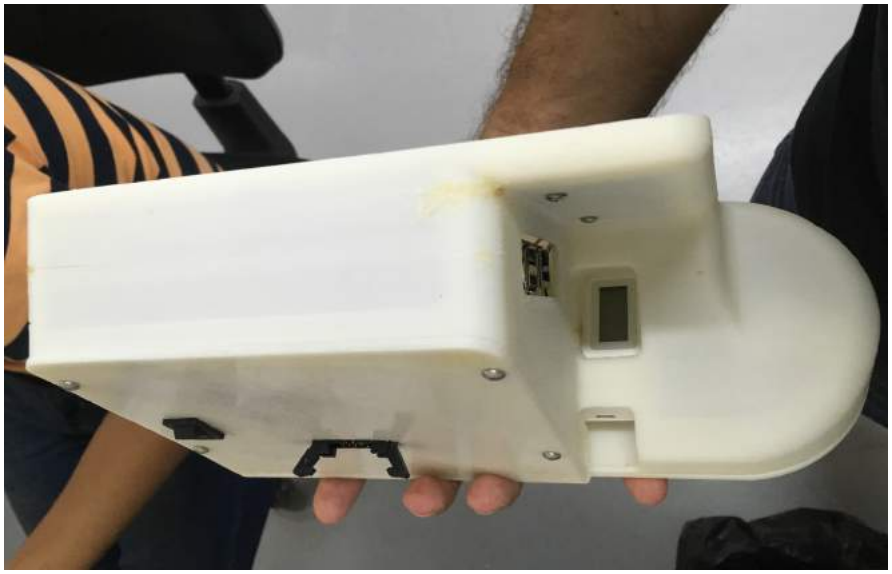

Figura 30: Caixa montada após impressão

Ao termino da impressão, todo o excesso de material foi removido e as partes impressas foram lixadas. Como a parte principal possui detalhes e é grande, surgiram diversas rachaduras. Portanto, foi necessário utilizar um ferro de solda e ABS proveniente do excesso de material para tapar essas rachaduras.

Mesmo após esse processo, a superfície da caixa não era lisa e não se mostrou pronta para ser pintada. Deste modo, foi necessário aplicar massa poliéster que é utilizada no setor automotivo para corrigir amassados e ranhuras em peças plásticas. O resultado após aplicar a massa e lixar a superfície é mostrado na Figura 31.

Para finalizar o processo da caixa, foi aplicada tinta automotiva, que é impermeável, à superfície da parte mecânica impressa 3D do controlador de estimulador neuromuscular. O resultado final do sistema pronto para uso é apresentado na Figura 32 e Figura 33.

Figura 31: Caixa após aplicação da massa poliéster

Figura 32: Vista frontal do controlador de estimulador neuromuscular

Figura 33: Vista traseira do controlador de estimulador neuromuscula

4 RESULTADOS E DISCUSSÃO

O penúltimo capítulo apresenta os testes do sistema e de estimulação e avalia o desempenho do projeto desenvolvido. Cada resultado obtido é estudado e discutido para que melhorias possam surgir e falhas sejam corrigidas.

4.1 Teste da interface gráfica

O teste da interface do programa QSTIMBERRY ocorreu durante todo o desenvolvimento de sua programação, para que *bugs* fossem corrigidos e funcionalidades fossem implementadas e melhoradas. Por fim, foi pedido para que uma pessoa leiga no assunto testasse a interface para avaliar se é amigável ou não. Testes em *softwares* por pessoas que não são aquelas que desenvolveram o programa são importantes, pois o desenvolvedor pode ter criado um vício durante a programação e não notar todos os *bugs* do programa.

O resultado da interface foi positivo e aparentemente todos os erros foram corrigidos. Porém, para operar o controlador e o estimulador, é necessário conhecimento dos equipamentos e do protocolo de estimulação. Portanto, é necessário orientação de pessoas que já entendem do processo para que um novo indivíduo possa operar o sistema completo.

4.2 Teste dos sinais de controle

Antes de testar o controlador no estimulador neuromuscular, foi feito o teste com osciloscópio de bancada para verificar se os parâmetros inseridos no QSTIMBERRY Controller estão sendo convertidos realmente em sinais de controle para acionar o estimulador.

Vários testes foram realizados com diferentes parâmetros e todos foram satisfatórios. Um desses testes é exemplificado a seguir:

- A Figura 34 mostra os parâmetros setados para estimulação: canal 8 selecionado, frequência de 25Hz, 100μs de largura de pulso, 100μs de largura entre pulsos e número de pulsos igual a 5.

- O osciloscópio foi conectado na saída do box header. A Figura 35 mostra a frequência de estimulação. Pode-se observar que a medida Delta dada entre cursor 1 e 2 do osciloscópio é igual a 25Hz, validando assim, a frequência de estimulação.

- A escala do osciloscópio foi ajustada para poder medir o sinal de controle emitido durante um ciclo (25Hz). A Figura 36 apresenta os pulsos de controle para o canal 8. Nota-se que há 5 pulsos na imagem e que a largura entre pulsos é de 100μs como estipulado no *software* de controle.

- Também pode ser observado a amplitude de 5V do pulso, demonstrando assim, a eficiência do CI conversor de nível lógico.

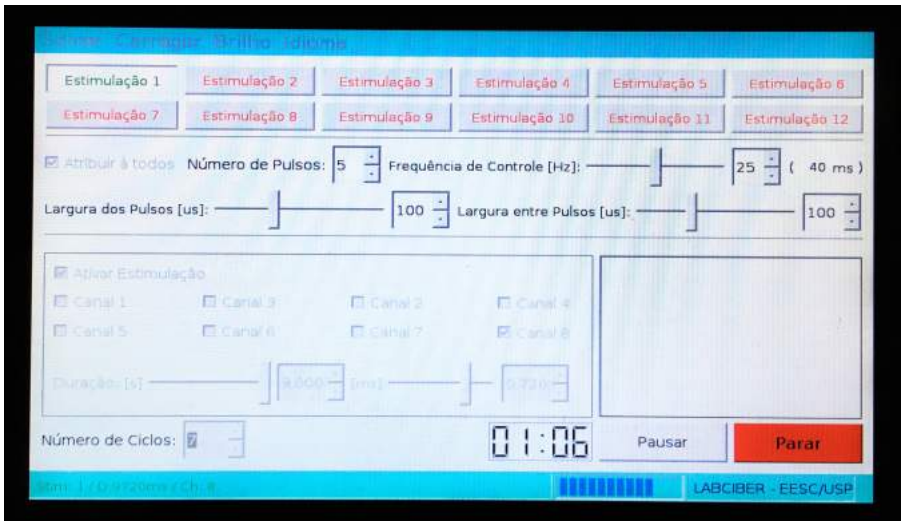

Figura 34: Parâmetros para realizar teste do sinal de controle

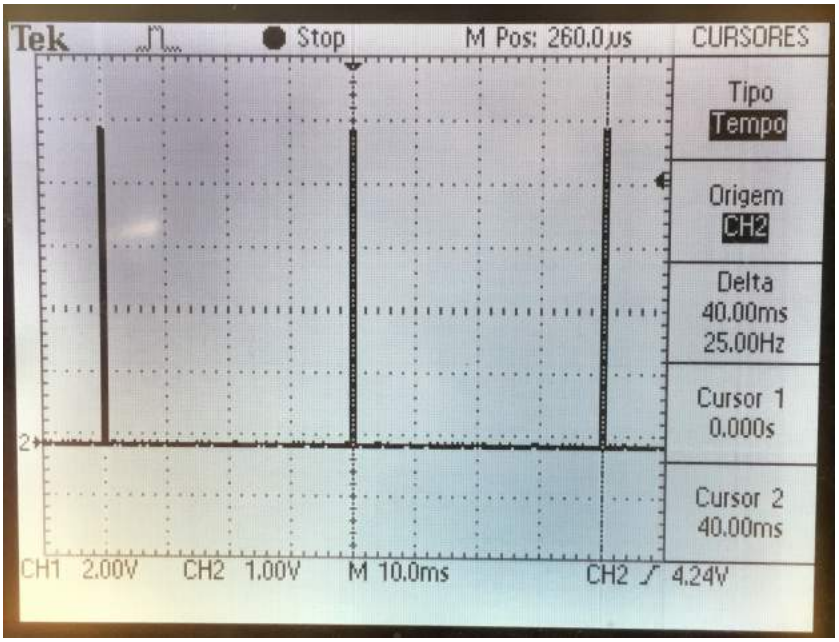

Figura 35: Frequência do sinal de saída do controlador

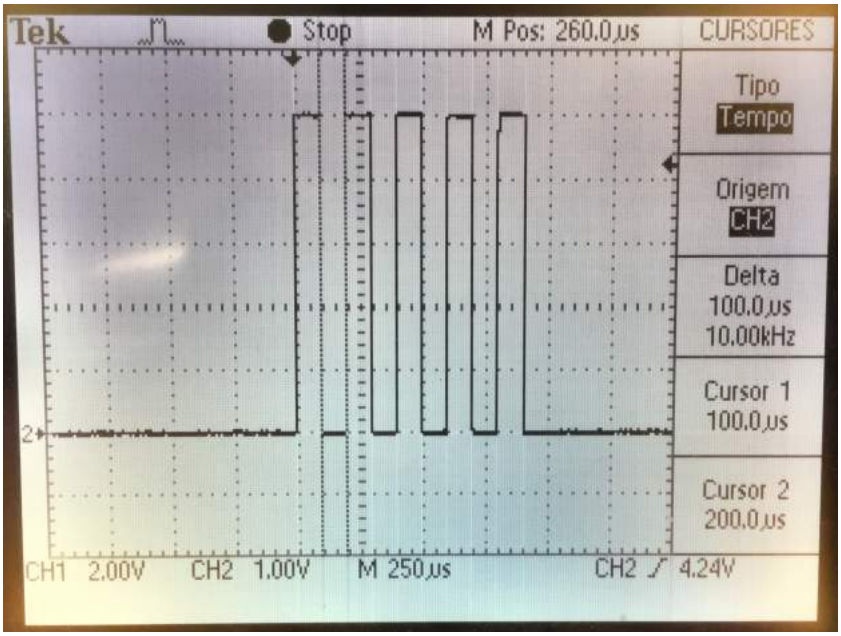

Figura 36: Sequência de pulsos de controle do controlador

4.3 Teste no estimulador

Após a verificação do sinal de controle, o controlador foi conectado ao estimulador para poder validar o sistema completo.A probe do osciloscópio foi ligada a uma resistência de 1000 ohms colocada na saída do estimulador, essa resistência é utilizada para simular a impedância da pele humana. A Figura 37 indica a largura dos pulsos, enquanto a Figura 38 apresenta a frequência entre pulsos. Nota-se também a amplitude do sinal de saída que é maior que 90V.

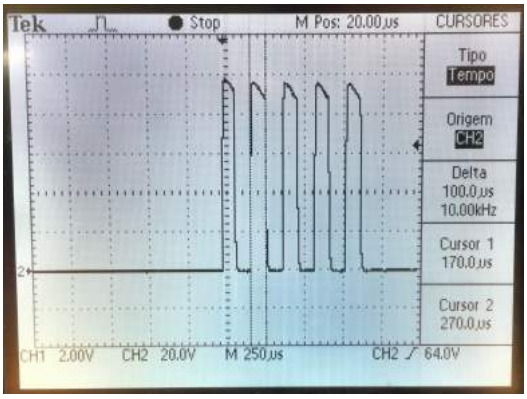

Figura 37: Saída do estimulador - frequência dos pulsos

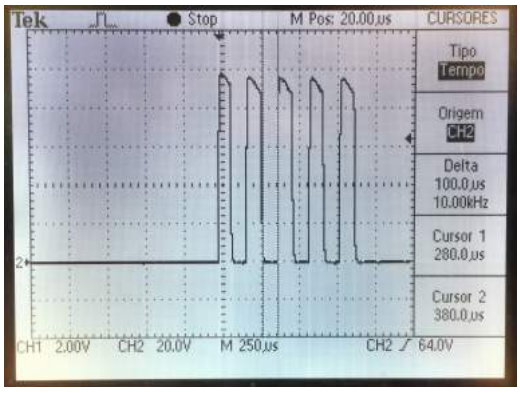

Figura 38: Saída do estimulador - frequência entre pulsos

A frequência de estimulação também foi medida e é mostrada na Figura 39.

Portanto, foi constatada a eficiência do controlador e também uma possível melhoria: o estimulador necessitava de um estágio de *buffer* devido ao controlador usado na versão

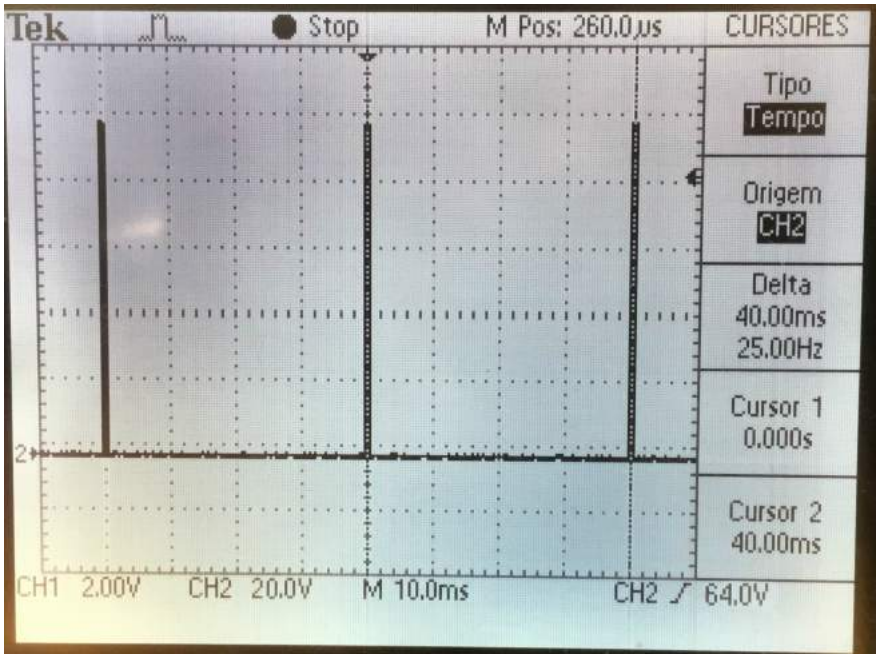

Figura 39: Saída do estimulador - frequência de estimulação

anterior. Porém, com o novo sistema de controle, esse estágio pode ser removido e, assim, diminuir o custo e tamanho do estimulador neuromuscular.

4.4 Teste de autonomia de bateria

Um dos fatores importantes de se utilizar uma placa embarcada para realizar o processo de controle é que o sistema seja portátil e sua autonomia seja suficiente para realizar estimulações durante um dia inteiro de trabalho, isto é, 8 horas.

O teste ideal a ser realizado é utilizar o equipamento na prática, direto com os pacientes durante algumas jornadas de trabalho e realizar a média de duração da bateria. Porém, uma estimativa foi estabelecida de acordo com o seguinte teste:

- Carregar a bateria portátil até a tela da mesma indicar 100%.

- Alimentar o *touchscreen* com a saída de 5V/1A e a placa com a saída de 5V/2,1A.

- Setar parâmetros de estimulação e estimular em intervalos aleatórios.

- Deixar o *backlight* da tela sempre ligado.

- Esperar o *power bank* descarregar por completo.

Os parâmetros de teste foram os seguintes:

- Estimulação de 1 a 6

- Todos os parâmetros iguais: canais de 1 a 4 selecionados, frequência de 25Hz, 100ms de largura de pulso, 100ms de largura entre pulsos, número de pulsos igual a 5,

duração de 9960ms.

- Número de ciclos igual a 10.

Logo, o ciclo total de estimulação do teste foi de 10 minutos e o intervalo entre esses ciclos foram feitos de forma aleatória. O teste se iniciou 12:05 com a bateria a 100% e se estendeu até 17:50 do dia seguinte, totalizando 29 horas e 45 minutos de funcionamento, e foi considerado que o *power bank* tem capacidade de 12580mAh para uma carga de 5V. O consumo do sistema teve uma média de 425mA.

Isso demonstra que o controlador tem autonomia de pelo 3 dias de trabalho sem necessitar de recarga. Outro ponto é que um carregador portátil com menos capacidade pode ser utilizado sem problemas.

É valido introduzir que a curva de descarregamento da bateria não é linear, por exemplo, os primeiros 15% de carga (de 100% a 85%) duram mais tempo do que os últimos 15% (de 15% a 0%) .Então o teste completo de descarga se faz necessário.

5 CONCLUSÃO

O projeto reúne diferentes conhecimentos sobre engenharia eletrônica e computação: C++ e Linux embarcado, circuito *debouncing*, solda e conexões e operação de equipamentos de medições. O objetivo principal do projeto foi alcançado: um controlador embarcado com interface gráfica amigável. Além disso, melhorias foram implementadas durante o desenvolvimento, como por exemplo, as abas de idiomas e de brilho da tela, o botão para desligar/ligar o *backlight* e o botão de "Pausar"/"Resumir". O resultado do trabalho foi um sistema eletrônico funcional e validado, com alta autonomia de bateria e com um design mecânico que fornece robustez e ergonomia ao sistema e facilita seu transporte.

O texto foi escrito em forma de tutorial para que modificações e melhorias sejam feitas por outros interessados em aplicar essa nova tecnologia de computadores de placa única e sistemas embarcados na área de estimulação neuromuscular e talvez em outras áreas médicas.

Deve-se enfatizar também a contribuição desse trabalho para demonstrar a importância de melhorias contínuas na área da saúde com o intuito de melhorar a vida e bem estar do paciente. E para que num futuro próximo, pessoas que sofreram de lesão medular possam ter uma vida normal e sem limitações como qualquer ser humano deve ter.

5.1 Implementações futuras

Esse texto também sugere futuras melhorias tanto para o controlador quanto para o estimulador neuromuscular. Essas melhorias são listadas a seguir.

- Implementar medição em tempo real do sinal do canal de saída do estimulador. Isso permite ao usuário constatar que os parâmetros inseridos estão sendo realmente entregues ao estimulador e também detectar problemas na parte de potência do equipamento. Para isso, é necessário outro controlador com entrada analógica que trata esse dados e comunique-se com a Raspberry Pi via UART ou I2C.

- Com o controlador atual, o estimulador não necessita mais de dois estágios: *buffer* e acoplamento óptico. Isso pode reduzir o tamanho do equipamento ou um novo estimulador com controlador integrado pode ser construído.

- A Raspberry Pi 3 possui módulo WiFi integrado, o que permite que o programa QSTIMBERRY se comunique com um banco de dados em nuvem e/ou receba comandos por meio da internet.

REFERÊNCIAS

BARBARINI, E. S. **Aplicação de um sistema de comando por voz e um software de controle na engenharia de reabilitação**. 2008. 68 f. Trabalho de Conclusão do Curso (Engenharia Elétrica com ênfase em Eletrônica) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2008.

CLIQUET, A. Rehabilitation engineering at the state university of campinas. **IEE Engineering in Medicine and Biology Magazine**, Estados Unidos, v. 1(2), p. 8–11, 1993.

MANHÃES, R. B. **A Engenharia de Reabilitação e as Características Psicossociais de Pessoas com Lesão Medular Submetidas a um Programa de Estimulação Elétrica Neuromuscular**. 2004. 248 f. Dissertação (Mestrado) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2004.

MINISTÉRIO DA SAÚDE - Secretaria de Atenção à Saúde. **Diretrizes de Atenção à Pessoa com Lesão Medular**. Brasilia, DF: Ministério da Saúde, 2013. 70 p. ISBN: 978-85-334-2025-0.

MOTA, J. E. **Descobrindo o Linux**. 3. ed. Brasília: Novatec, 2012. 928 p.

PARENTE, L. N. **Raspberry Pi based system to control neuromuscular stimulation**. 2016. 41 f. Individual project (Electronics & Electrical Engineering) — School of Engineering, University of Glasgow, Glasgow, UK, 2016.

RASPBERRY PI FOUNDATION. **Official Raspberry Pi DSI Display**. Caldecote, UK, 2015. Disponível em: <https://www.raspberrypi.org/blog/the-eagerly-awaited-raspberry-pi-display/>. Acesso em: 29 out. 2017.

_____. **Official website**. Caldecote, UK, 2017. Disponível em: <https://www.raspberrypi.org/>. Acesso em: 29 out. 2017.

SHEFFLER, R. L.; CHAE, J. Neuromuscular electrical stimulation in neurorehabilitation. **Muscle & Nerve**, Cleverland, v. 35(5), p. 562–590, 2007.

THE QT COMPANY LTD. **About Qt**. California, USA, 2017. Disponível em: <https://wiki.qt.io/About_Qt>. Acesso em: 29 out. 2017.

_____. **Qt for embedded linux**. California, USA, 2017. Disponível em: <http://doc.qt.io/qt-5/embedded-linux.html>. Acesso em: 29 out. 2017.

VAROTO, R. **Desenvolvimento e avaliação de um protótipo de sistema híbrido para membro superior tetraplégico**. 2010. 250 f. Tese (Doutorado em Engenharia Elétrica) — Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2010.

Apêndices

APÊNDICE A – CONFIGURAÇÃO SSH RASPBERRY PI

Esse tutorial apresenta como acessar a Raspberry Pi 2 ou 3 via cabo Ethernet utilizando um Notebook, sem necessitar de nenhum periférico (monitor, teclado e/ou mouse) para o computador de placa única.

Requisitos

- Cabo Ethernet.

- Raspberry Pi 2 ou 3.

- Cartão micro SD com Raspian Jessie SO.

- Notebook com GNU/Linux (nesse tutorial foi utilizado o Ubuntu 16.04).

Download

- Baixar a imagem do Raspian Jessie LITE OS e queimar em um micro SD Card.

Permitir SSH na Raspberry Pi

- Inserir o micro SD Card no Notebook com Ubuntu. Dois diretórios serão abertos.

- Criar um arquivo nomeado “ssh” dentro da partição **boot** com o comando:

```
$ sudo touch ssh
```

Configurar como auto-login

- Criar o arquivo /etc/systemd/system/getty@tty1.service.d/autologin.conf com o comando touch:

```
$ sudo touch /etc/systemd/system/getty@tty1.service.d/autologin.conf
```

- Abrir o arquivo com o editor nano:

```
$ sudo nano /etc/systemd/system/getty@tty1.service.d/autologin.conf
```

- Adicionar o código a seguir:

```
[Service]
ExecStart=
ExecStart=-/sbin/agetty --autologin pi --noclear %I 38400 linux
```

- Sair do arquivo, salvar e executar o comando:

```
$ sudo systemctl enable getty@tty1.service
```

- Inserir o micro SD Card na Raspberry Pi.

Compartilhar acesso à internet (Wifi) do notebook à ethernet

- Instalar network-manager:

```
$ sudo apt-get install network-manager
```

- Instalar nmap:

```
$ sudo apt-get install nmap
```

- Acessar e habilitar compartilhamento de rede no Ubuntu:

```
System Settings > Network > Wired > Options.. > Ipv4 Settings >
Method > Shared to other computers
```

- Salvar e reiniciar o notebook.

Descobrir IP da Raspberry

- Alimentar e conectar a Raspberry Pi ao Notebook através do cabo ethernet.

- Execute o comando para descobrir qual o IP da Raspberry:

```
$ nmap -n -sP 10.42.0.255/24
```

- Para o meu caso, resultado do comando acima foi:

```
Starting Nmap 7.12 ( https://nmap.org ) at 2017-02-23 21:16 BRT
Nmap scan report for 10.42.0.1
Host is up (0.00016s latency).
Nmap scan report for 10.42.0.25
Host is up (0.0024s latency).
Nmap done: 256 IP addresses (2 hosts up) scanned in 2.73 seconds
```

- O primeiro IP (10.42.0.1) encontrado é do *host* (Notebook) e o segundo corresponde à Raspberry Pi, que no meu caso é 10.42.0.25.

Acesso via SSH

- Por fim, é só executar o comando ssh para acessar a Raspberry Pi pelo Notebook:

```
$ ssh -Y pi@10.42.0.25
```

- A senha padrão do para o usuário ”pi” é ”raspberry”.

APÊNDICE B – COMPILAÇÃO CRUZADA EFGLS

O tutorial de compilação cruzada utilizando Qt Creator e Raspberry Pi 3 é fornecido pela própria comunidade do Qt no link: http://wiki.qt.io/RaspberryPi2EGLFS (acessado em 29 nov. 2017). Porém, como o texto está em inglês e modificações tiveram que ser feitas, o conteúdo foi traduzido e adaptado para esse trabalho.

Forçar 'apt-get' a usar Ipv4

- Como o protocolo ethernet entre Raspberry e o Notebook é Ipv4, é necessário alterar a configuração atual do comando apt-get para setar a placa para receber o qt5:

```
$ sudo apt-get -o Acquire::ForceIPv4=true update
```

Preparando o sistema da Raspberry Pi

- Primeiramente, atualize o SO:

```
$ sudo rpi-update
$ reboot
```

- Caso vá usar a biblioteca pigpio (utilizada nesse trabalho), instale-a antes:

```
$ sudo apt-get pigpio
```

- Abra o arquivo */etc/apt/sources.list* e descomente a linha **deb-src**:

```
$ sudo nano /etc/apt/sources.list
```

- Atualize e instale as bibliotecas necessárias:

```
$ sudo apt-get update
$ sudo apt-get build-dep qt4-x11
$ sudo apt-get build-dep libqt5gui5
$ sudo apt-get install libudev-dev libinput-dev libts-dev
 libxcb-xinerama0-dev libxcb-xinerama0
```

- Prepare o diretório qt5 na Raspberry:

```
$ sudo mkdir /usr/local/qt5pi
sudo chown pi:pi /usr/local/qt5pi
```

Preparando o sistema do Notebook Ubuntu

- Crie o diretório para sincronizar os dados com a Raspberry Pi:

```
$ mkdir ~/raspi
```

- Acesse o diretório e clone o *toolchain* da Raspberry para dentro dele:

```
$ cd ~/raspi3
$ git clone https://github.com/raspberrypi/tools
```

- Sincronize os dados da Raspberry Pi com a pasta /raspi3 em seu computador via rsync. Substitua *IPraspberrypi* pelo IP encontrado no Apêndice anterior:

```
$ mkdir sysroot sysroot/usr sysroot/opt
$ rsync -avz pi@IPraspberypi:/lib sysroot
$ rsync -avz pi@IPraspberypi:/usr/include sysroot/usr
$ rsync -avz pi@IPraspberypi:/usr/lib sysroot/usr
$ rsync -avz pi@IPraspberypi:/opt/vc sysroot/opt
```

- Ajuste o *symlink* para ser relativo utilizando o script de python abaixo:

```
$ wget https://raw.githubusercontent.com/riscv/riscv-poky/master
/scripts/sysroot-relativelinks.py
$ chmod +x sysroot-relativelinks.py
$ ./sysroot-relativelinks.py sysroot
```

- Baixe qtbase e configure o Qt:

```
$ git clone git://code.qt.io/qt/qtbase.git -b 5.8
$ cd qtbase
$ ./configure -release -opengl es2 -device linux-rpi3-g++ -device-option
CROSS_COMPILE=~/raspi3/tools/arm-bcm2708/gcc-linaro-arm-linux-gnueabihf
-raspbian-x64/bin/arm-linux-gnueabihf-  -sysroot ~/raspi3/sysroot -prefix
 /usr/local/qt5pi -extprefix ~/raspi3/qt5pi -hostprefix ~/raspi3/qt5 -v
 -nomake examples -nomake tests -no-use-gold-linker

$ make
$ make install
```

Isso faz com que as ferramentas do host (notebook) como *qmake* vão para /raspi/qt5, enquanto *make install* vai para /raspi/qt5pi. Para executar o mesmo processo na Raspberry Pi 2 e/ou outra versão do Qt, acesse a páginal oficial.

- Agora, transfira o Qt para a Raspberry Pi ressincronizando as pastas:

```
$ rsync -avz qt5pi pi@IPraspberypi:/usr/local
```

- Construa um exemplo no Notebook e transfira o executável para a placa para testar:

```
$ cd qtbase/examples/opengl/qopenglwidget
$ ~/raspi/qt5/bin/qmake
$ make

$ scp qopenglwidget pi@IPraspberypi:/home/pi
```

Correções de bibliotecas na Raspberry Pi

- Permita que o *linker* encontre as bibliotecas do Qt:

```
$ echo /usr/local/qt5pi/lib | sudo tee /etc/ld.so.conf.d/qt5pi.conf
$ sudo ldconfig
```

- Corrija os *bugs* das bibliotecas EGL/GLES:

```
$ sudo mv /usr/lib/arm-linux-gnueabihf/libEGL.so.1.0.0 /usr/lib/
arm-linux-gnueabihf/libEGL.so.1.0.0_backup
$ sudo mv /usr/lib/arm-linux-gnueabihf/libGLESv2.so.2.0.0 /usr/lib/
arm-linux-gnueabihf/libGLESv2.so.2.0.0_backup
$ sudo ln -s /opt/vc/lib/libEGL.so /usr/lib/arm-linux-gnueabihf/
libEGL.so.1.0.0
$ sudo ln -s /opt/vc/lib/libGLESv2.so /usr/lib/arm-linux-gnueabihf0/
libGLESv2.so.2.0.0

$ sudo ln -s /opt/vc/lib/libEGL.so /usr/lib/arm-linux-gnueabihf/
libEGL.so.1.0.0$
$ sudo ln -s /opt/vc/lib/libGLESv2.so /usr/lib/arm-linux-gnueabihf/
libGLESv2.so.2.0.0
```