TRABALHO DE CONCLUSÃO DE CURSO 2

AUTOMAÇÃO DE AMPLIFICADORES PARA INSTRUMENTOS MUSICAIS

Wagner Mourthé Starling Pinheiro

Brasília, Julho de 2017

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

TRABALHO DE CONCLUSÃO DE CURSO 2

AUTOMAÇÃO DE AMPLIFICADORES PARA INSTRUMENTOS MUSICAIS

Wagner Mourthé Starling Pinheiro

Trabalho de Conclusão de Curso 2 submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Licenciado em Engenharia de Elétrica

Banca Examinadora

Prof. Ricardo Zelenovsky, Ph.D, FT/UnB _____
Orientador

Prof. Alexandre Ricardo Soares Romariz, Ph.D, FT/UnB _____
Examinador interno

Prof. Daniel Chaves Café, Ph.D, FT/UnB _____
Examinador interno

FICHA CATALOGRÁFICA

PINHEIRO, WAGNER
AUTOMAÇÃO DE AMPLIFICADORES PARA INSTRUMENTOS MUSICAIS [Distrito Federal] 2017.
xvi, 63 p., 210 x 297 mm (ENE/FT/UnB, Licenciado, Engenharia Elétrica, 2017).
Trabalho de Conclusão de Curso 2 - Universidade de Brasília, Faculdade de Tecnologia.
Departamento de Engenharia Elétrica

1. Amplificador musical analógico
2. Automação de equipamentos
3. Motor servo
4. Solução eletrônica em música

I. ENE/FT/UnB
II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

PINHEIRO, W.M.S. (2017). *AUTOMAÇÃO DE AMPLIFICADORES PARA INSTRUMENTOS MUSICAIS* . Trabalho de Conclusão de Curso 2, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 63 p.

CESSÃO DE DIREITOS

AUTOR: Wagner Mourthé Starling Pinheiro
TÍTULO: AUTOMAÇÃO DE AMPLIFICADORES PARA INSTRUMENTOS MUSICAIS .
GRAU: Licenciado em Engenharia de Elétrica ANO: 2017

Wagner Mourthé Starling Pinheiro
Depto. de Engenharia Elétrica (ENE) - FT
Universidade de Brasília (UnB)
Campus Darcy Ribeiro
CEP 70919-970 - Brasília - DF - Brasil

Dedicatória

A todos que me apoiaram durante o meu curso de graduação.

Wagner Mourthé Starling Pinheiro

Agradecimentos

Ao meu irmão e melhor amigo, Marcos, que, além de ser meu parceiro na tentativa de inserir o projeto no mercado, têm sido meu parceiro na música. A minha melhor amiga, Fernanda, que também é minha parceira nesta tentativa e, ao que tudo indica, será minha parceira na música em breve. A minha namorada, Bruna, que me apoiou durante todo o projeto, principalmente na fase final, sem me deixar desanimar nos momentos mais difíceis. Ao meu irmão, Lucas, que, apesar de estar do outro lado do mundo, se fez presente, sempre me apoiando e me motivando. A minha irmã, Dulce, por todo carinho e amor que recebo, mesmo à distância. Aos meus pais, Simone e Jânio, por terem me dado a estrutura e a educação que possibilitou o meu desenvolvimento intelectual e moral. A todos os meus amigos, que suportaram a minha ausência nos últimos 12 meses. Finalmente, agradeço ao meu orientador, Prof. Ricardo, que me deu a liberdade de desenvolver um projeto próprio.

Wagner Mourthé Starling Pinheiro

RESUMO

Os amplificadores de guitarra analógicos, apesar de tecnologicamente ultrapassados, ainda são preferidos pela maioria dos músicos. Os equipamentos digitais tentam emular o som destes amplificadores analógicos, mas poucos conseguem agradar os usuários, do ponto de vista do timbre. A criação do *RoadieBot* visa, por meio da automação das configurações de timbre, expandir a experiência do músico e facilitar o trabalho do engenheiro de gravação que utiliza amplificadores analógicos de instrumentos musicais. Esta obra descreve o desenvolvimento do primeiro protótipo deste produto, abordando todos os aspectos relevantes deste processo.

ABSTRACT

The analog guitar amplifiers, although technologically outdated, are still preferred by most musicians. The digital equipment tries to emulate the sound of these analog amplifiers, but few can please the user, from the timbre aspect. The creation of RoadieBot aims, through the automation of the amplifier configurations, to expand the musician's experience and to facilitate the work of the recording engineer when using analog musical instruments amplifiers. This work describes the development of this product's first prototype, addressing all relevant aspects of the process.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

3D	Três Dimensões
AD	Analógico Digital
CC	Corrente Contínua
E/S	Entrada/Saída
IDE	*Integrated Development Environment*
I^2C	*Inter-Integrated Circuit*
LCD	*Liquid Crystal Display*
MPP	Modulação por Posição de Pulso
NAMM	*National Association of Music Merchants*
RX	*Receiver*
SDA	*Serial Data*
SCI	*Serial Communication Interface*
SCL	*Serial Clock*
TX	*Transmitter*
UART	*Universal Asynchronous Receiver Transmitter*
USB	*Universal Serial Bus*
XLR	*Extra Long Run*

1 INTRODUÇÃO

1.1 AMBIENTAÇÃO

Existe uma grande discussão no âmbito da música em relação a que tipo de equipamento é melhor: o digital ou o analógico. O setor de equipamentos musicais é, provavelmente, um dos únicos setores que ainda usam válvulas nos circuitos eletrônicos, por exemplo. A característica sonora destes equipamentos de tecnologia ultrapassada acabou se tornando a referência de qualidade e até hoje a indústria lança novos equipamentos projetados com estas tecnologias

Em termos técnicos, os equipamentos digitais superam os equipamentos analógicos em vários quesitos como portabilidade, praticidade e capacidade de armazenar e processar grandes quantidades de informação.

Alguns dos defeitos associados ao equipamento analógico, como a distorção harmônica, no meio artístico musical, acabaram virando referência de qualidade. Na verdade, em alguns amplificadores de guitarra, o circuito é intencionalmente subdimensionado para que o sinal seja distorcido. O som de alta fidelidade oferecido pelo equipamento digital é tido como "sem graça" ou "sem vida" [1].

O equipamento musical digital oferece vantagens como: maior variedade de timbres, capacidade de armazenamento de configurações e praticidade, porém a característica sonora dos equipamentos analógicos ainda é o sonho de consumo da maioria dos músicos.

A tecnologia digital evolui constantemente e a indústria procura cada vez mais se aproximar da característica sonora analógica, porém ainda são os amplificadores analógicos os que dominam o mercado [2], [3].

A Figura 1.1 mostra uma configuração de amplificadores analógicos de guitarra em um palco. Apesar de extremamente caros, ainda dominam o mercado graças à característica sonora. Até artistas que necessitam viajar constantemente durante longas turnês optam pelo equipamento analógico, mesmo que sejam mais frágeis, pesados e grandes.

Um único amplificador analógico possui uma infinita variedade de timbres possíveis que dependem do ajuste feito pelo músico. Por não ter como alterar esse ajuste enquanto toca, o usuário acaba sendo limitado a uma única configuração durante toda sua performance.

Além disso, nos estúdios de gravação, os amplificadores de guitarra ficam em uma sala isolada, devido aos altos níveis de intensidade sonora destes equipamentos. Existe então, para o engenheiro de som, a necessidade de andar entre as salas para regular as configurações do amplificador até que esteja satisfeito com o timbre [4], [5].

A automatização dos ajustes de timbre do amplificador solucionaria este problema para ambos, os músicos e os engenheiros, já que uniria algumas das vantagens práticas do equipamento

Figura 1.1: Amplificadores de guitarra analógicos - **Fonte:** Pinterest

digital à característica sonora do equipamento analógico.

O músico poderia explorar ao máximo a capacidade do seu equipamento, trocando as configurações enquanto toca. Por outro lado, o engenheiro de som poderia controlar as configurações do equipamento remotamente da sala técnica.

O *MusicRadar*, um *website* especializado na área, fez uma lista com os 20 melhores lançamentos em amplificadores de baixo e guitarra da NAMM 2016, a maior feira de música do mundo [6]. Nesta lista, somente três empresas apresentaram equipamentos com alguma solução em automação de ajustes e estes eram os únicos digitais.

O foco da indústria de amplificadores ainda é a característica sonora e, por consequência, o equipamento analógico. Nenhum dos amplificadores analógicos apresentou alguma solução neste sentido, indicando que pode existir uma oportunidade nesta área.

1.2 DESCRIÇÃO DO PROJETO

Este trabalho descreve o desenvolvimento do protótipo de um produto que automatiza a configuração de timbre de amplificadores musicais, permitindo que estes sejam controlados remotamente. O produto combina parte da praticidade do amplificador digital com a característica sonora do analógico, sem que modificações no amplificador sejam necessárias.

Os circuitos eletrônicos destes amplificadores são verdadeiras obras de engenharia e arte, e

substituir um componente original por um automatizado direto no circuito poderia arruinar a característica sonora deste equipamento.

Com isso em mente, o protótipo proposto deve poder ser utilizado com qualquer formato de amplificador do mercado e interferir apenas na parte externa daquele, de maneira que a característica sonora do circuito original seja preservada.

O produto foi batizado de *RoadieBot*, pois *Roadie* é o termo em inglês que define o técnico responsável pelo equipamento do músico durante uma apresentação ou turnê, e *Bot* é uma abreviação de *Robot*, termo em inglês que significa robô.

O protótipo do *RoadieBot* será composto basicamente por uma pedaleira de controle e pelos módulos motorizados. A pedaleira de controle é o elemento de interface principal do produto, pelo qual o usuário poderá controlar os módulos, ajustando a configuração do amplificador como desejado. Os módulos são os elementos motorizados responsáveis por mover os potenciômetros do amplificador.

O controle por hardware poderá ser feito de três maneiras. A primeira forma possibilita ao usuário, por meio da pedaleira, controlar individualmente cada *knob* do amplificador no qual o módulo motorizado estiver instalado, armazenando o resultado final da configuração.

A segunda forma seria o ajuste manual de cada *knob*, ao invés de remoto, e então o armazenamento da configuração na pedaleira. Por último, o usuário tem a possibilidade de acionar, por meio da pedaleira, uma das configurações anteriormente armazenadas.

Um sistema contido na pedaleira de controle é apresentado, no qual o usuário pode personalizar o arranjo de suas configurações armazenadas, podendo acioná-las posteriormente por meio de um pedal ou editá-las.

A pedaleira de controle se comunica com os módulos por meio de cabeamento e é alimentada por uma fonte de alimentação independente.

Além destas formas de controle, o protótipo deverá poder ser acionado, também, por um *software* com interface no computador. Este tipo de controle é ideal para as aplicações em estúdio, onde o uso de computadores já é praticamente obrigatório atualmente.

O software em si não será abordado neste trabalho por estar sendo desenvolvido por outra pessoa, entretanto, a parte que torna possível a comunicação entre o software e o protótipo é parte integrante do mesmo.

Os módulos, por sua vez, são a parte chave do produto. Por apresentar característica modular, o cliente pode comprar quantos módulos desejar, adaptando o produto à configuração espacial do seu amplificador. Cada um deles é equipado com um motor elétrico que é responsável por girar um potenciômetro do amplificador.

Todo amplificador possui *knobs* acoplados ao eixo de cada um dos seus potenciômetros e, por possuírem os mais variados formatos, projetar uma peça capaz de se conectar a qualquer tipo de *knob* é de extrema dificuldade.

Então, para utilizar o protótipo do *RoadieBot*, o usuário deverá remover os *knobs* dos potenciômetros que deseja automatizar. Assim, uma simples junção de eixos fixada por um parafuso é suficiente para acoplar o eixo dos módulos ao eixo dos potenciômetros do amplificador.

O protótipo cujo desenvolvimento é retratado neste trabalho já foi testado em situações reais, em uma gravação profissional e com um músico em um ensaio.

Estes não foram testes controlados buscando resultados técnicos. Foram testes para averiguar a proposta de valor do projeto como produto.

1.3 OBJETIVOS DO PROJETO

Este projeto tem como objetivo principal desenvolver um protótipo funcional do produto *RoadieBot* que seja capaz de ser usado em testes com músicos e engenheiros de som para coletar informações quanto à validação do produto no mercado da música.

1.4 CONTRIBUIÇÕES DO PROJETO

Entre as principais contribuições deste trabalho, do ponto de vista do protótipo, destacam-se:

- Estimular o músico a explorar as capacidades de timbre do seu amplificador em apresentações ao vivo;
- Acelerar o processo de busca pelo timbre ideal durante a gravação.

1.5 DESCRIÇÃO DO TRABALHO

Este trabalho foi organizado de maneira que o leitor possa compreender todas as etapas do desenvolvimento e o funcionamento do protótipo. No Capítulo 2 são discutidos os aspectos técnicos e teóricos que foram necessários para as tomadas de decisão mais importantes em cada parte do protótipo. No Capítulo 3 é descrita a montagem de cada parte do protótipo. No Capítulo 4 é explanado o funcionamento dos códigos programados nos microcontroladores do projeto. Por fim, no Capítulo 6 são mostrados os resultados e são discutidas possíveis melhorias.

2 FUNDAMENTAÇÃO

Neste capítulo serão explanados aspectos técnicos e teóricos que levaram às escolhas necessárias para cada parte do protótipo. Serão abordados a comunicação entre os elementos do projeto, os módulos motorizados, a Pedaleira de Controle e a Ponte de Comando.

2.1 COMUNICAÇÃO E VISÃO GERAL

Esta seção trata da visão geral do protótipo e da escolha do protocolo de comunicação que será usado na conexão entre a pedaleira de controle e os módulos motorizados.

A Figura 2.1 mostra um diagrama de blocos que representa a visão geral do protótipo. Nota-se a presença de um novo elemento, a Ponte de Comando, que será explicado posteriormente. Todos os Módulos motorizados são conectados a este elemento individualmente, assim como a Pedaleira de Controle.

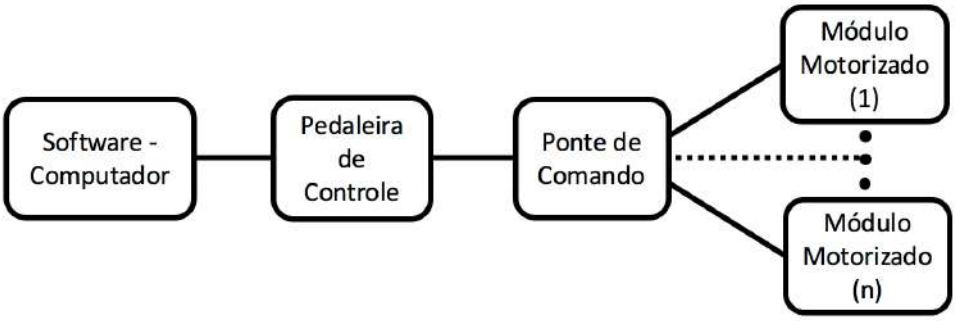

Figura 2.1: Visão geral do protótipo - **Fonte:** Autor

Este esquema geral de ligação foi uma consequência da escolha da comunicação entre os elementos. Inicialmente, foi considerado o uso do protocolo I^2C pela Philips. Este é um protocolo de comunicação serial que opera no princípio mestre-escravo e utiliza dois fios para realizar a transferência de dados: o barramento do relógio SCL, e a linha de dados SDA.

A pedaleira de controle exerceria o papel de mestre, enquanto cada módulo exerceria o papel de escravo, como ilustrado na Figura 2.2 abaixo. Para tal, cada módulo teria o seu próprio microcontrolador, que seria responsável pela comunicação (respondendo aos comandos do mestre), por controlar o motor servo e obter a posição angular do eixo.

A vantagem desta abordagem reside na simplicidade das conexões por cabo. Cada módulo apresentaria uma entrada e uma saída com quatro terminais, onde dois terminais estariam destinados à comunicação I^2C e os outros dois destinados à tensão de alimentação do módulo.

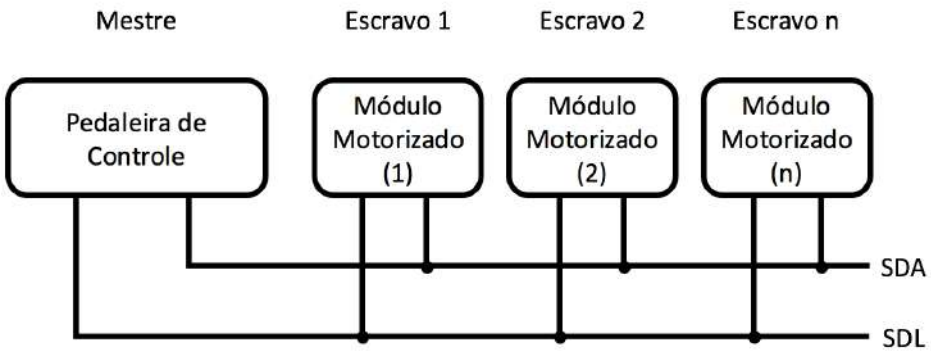

Figura 2.2: Visão geral do protótipo utilizando comunicação I^2C - **Fonte:** Autor

Os módulos poderiam, então, ser ligados em cascata, diminuindo o comprimento dos cabos e deixando o produto com um visual mais limpo.

A principal desvantagem desta implementação, que resultou no uso de outro tipo de comunicação, está no custo de fabricação, pois cada módulo seria um sistema embarcado por si só.

Além disso, cada módulo teria que possuir seu próprio endereço de escravo, diferente dos endereços de todos os outros módulos sendo usados simultaneamente. Caso dois módulos fossem endereçados igualmente, haveria um problema na comunicação.

Tendo em vista estes fatos, foi decidido inserir um elemento intermediário entre os módulos motorizados e a pedaleira de controle. Este elemento será chamado de Ponte de Comando e será composto pela fonte de tensão, que alimentará os módulos, e por um microcontrolador. O microcontrolador receberá os comandos da pedaleira de controle ou do software e controlará os motores.

Assim como os módulos motorizados, a Ponte de Comando terá característica modular e expansível. O número de Pontes de Comando utilizadas dependerá de quantos módulos o usuário necessita, pois cada Ponte de Comando poderá controlar um número máximo de módulos definido a partir das limitações do microcontrolador utilizado. A fonte de alimentação será dimensionada baseando-se neste número máximo de módulos que podem ser controlados por um elemento.

A Ponte de Comando irá se comunicar com a Pedaleira de Controle utilizando a SCI (Interface de Comunicação Serial) por meio do módulo UART (Transmissor/Receptor Universal Assíncrono). Este não é um protocolo de comunicação em si, mas um módulo que pode ser usado para comunicação serial ponto-a-ponto assíncrona [7] utilizando dois fios, o TX da linha de transmissão e o RX da linha de recepção [8].

A Ponte de Comando, então, recebe um comando e traduz em uma ação, seja mover módulos ou ler posições angulares. A Figura 2.3 abaixo ilustra a visão geral do projeto utilizando este tipo de comunicação.

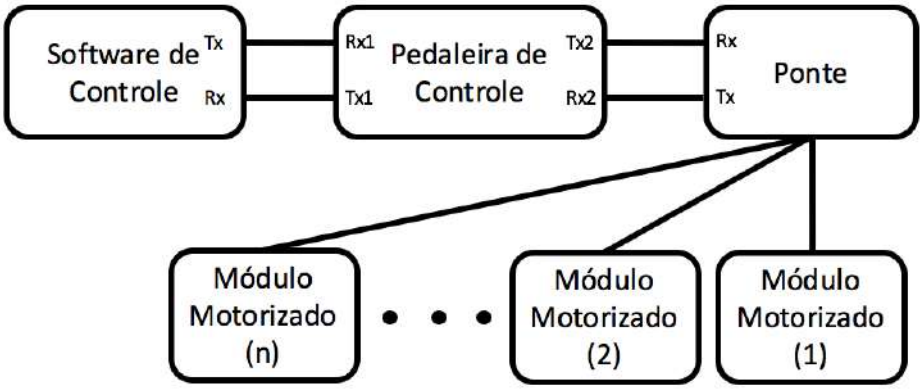

Figura 2.3: Visão geral do protótipo utilizando o módulo UART para comunicação - **Fonte:** Autor

O usuário, através da interface na pedaleira, define as ações a serem tomadas, seja mover um único módulo ou acessar uma configuração previamente definida. A pedaleira, por sua vez, enviará os comandos para a Ponte de Comando pela porta serial.

Além disso, a pedaleira funcionará também como a ligação entre o software de controle no computador e a Ponte de Comando. Por meio de uma porta USB, utilizando novamente a SCI e o módulo UART, o software envia o comando para a pedaleira que, por sua vez, envia o comando para a Ponte de Comando.

2.2 MÓDULOS MOTORIZADOS

Os módulos contendo os motores são a parte chave do produto. Cada módulo é responsável pelo ajuste de um único potenciômetro, devendo ser capaz de rodá-lo e obter a posição angular com precisão. Além disso, é importante que o usuário possa ajustar o amplificador manualmente, mesmo com os módulos conectados.

No protótipo, em uma extremidade do eixo estará o acoplamento com o potenciômetro do amplificador, e, na outra, um *knob* que pode ser ajustado pelo músico manualmente.

2.2.1 Motor

A posição angular do eixo é de extrema importância para este projeto. Os motores devem ser capazes de mover os potenciômetros com precisão no posicionamento angular.

Dois tipos de motores foram considerados: o motor de passo e o motor servo. Ambos trabalham com precisão em posição angular, porém cada um possui suas vantagens e desvantagens.

Motor de Passo

O motor de passo gira em passos angulares exatos devido ao arranjo de seus polos magnéticos e por isso é extremamente preciso em relação à posição angular [9], [10]. Nos testes, foi utilizado o motor de passo 28BYJ-48 fabricado pela Kiatronics [11], de tamanho reduzido e bastante usado em projetos com placas de prototipagem, como Arduino e Raspberry Pi.

Como esperado, o motor é bastante preciso em relação à posição angular, entretanto possui algumas desvantagens. Motores de passo, devido ao modo como são construídos, são bastante largos. Em contrapartida, os módulos devem ter a menor largura possível, pois alguns amplificadores possuem potenciômetros extremamente próximos um dos outros.

Um dos requisitos do projeto dos módulos é a capacidade de obter a posição angular quando requisitado. A maneira mais utilizada de obter tal informação com motores de passo é usando um sensor de fim de curso. O controlador manda o motor de passo se mover enquanto contabiliza o número de passos dados.

O motor para quando alcança o sensor de fim de curso e então a partir do número de passos é possível calcular a posição angular na qual se encontrava. A desvantagem deste processo é que o motor deve mover até o fim de curso para que a posição angular seja obtida, enquanto o ideal seria que o mesmo ficasse parado.

Outra desvantagem, também devido à construção do motor, é que mesmo com este desligado, o eixo oferece bastante dificuldade ao giro manual. Isto se deve ao fato deste modelo de motor de passo possuir uma caixa de redução na razão de 1 para 64.

Como o eixo do motor é acoplado mecanicamente ao do potenciômetro, o ajuste manual do amplificador seria muito difícil ou até impossibilitado.

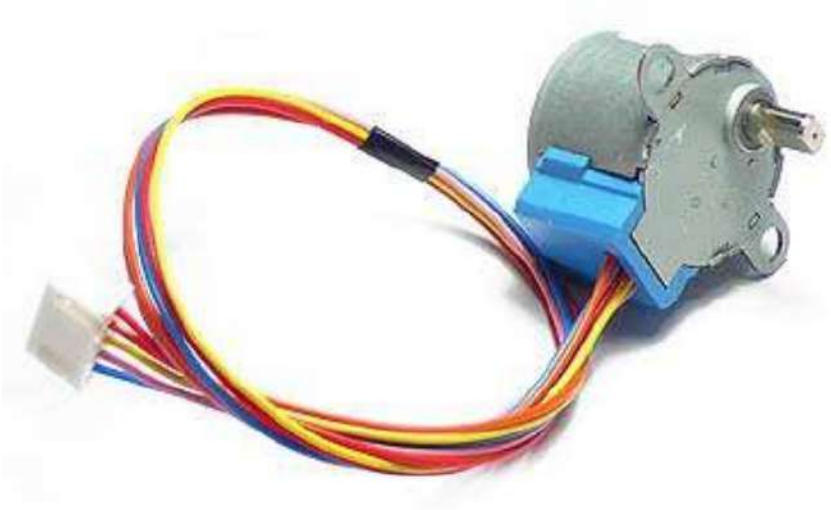

Figura 2.4: Motor de passo 28BYJ-48 - **Fonte:** Kiatronics

A Figura 2.4 mostra o motor de passo 28BYJ-48. Este motor foi descartado, pois não atendia,

principalmente, à especificação de tamanho do projeto.

Motor Servo

O motor servo é composto por um potenciômetro acoplado mecanicamente ao eixo de um motor CC comum com conjuntos de engrenagens de redução de velocidade [12], [10]. Um sistema de controle de malha fechada usa o potenciômetro para identificar a posição do eixo e aciona o motor até que o eixo esteja na posição correta. Este tipo de motor possui três terminais de conexão, dois são destinados à fonte de alimentação e um terceiro é o terminal de controle.

O comando com a posição angular é enviado para o motor pelo terminal de controle. A posição angular é determinada pela duração dos pulsos de um trem de pulsos aplicado neste terminal. Essa característica é chamada de MPP(modulação por posição do pulso).

Cada modelo de motor servo possui uma duração de pulso máxima, correspondendo ao máximo deslocamento em um sentido, e uma duração de pulso mínima, correspondendo ao máximo deslocamento no sentido contrário.

Em geral, os motores servo só giram cerca de 180 graus, enquanto os potenciômetros de um amplificador giram por volta de 320 graus, sendo essa a principal desvantagem deste motor para esta aplicação.

Contudo, como mencionado anteriormente, o tamanho dos módulos contendo os motores é de extrema relevância para este projeto e os motores servo são mais compactos que o motor de passo, e também mais baratos.

A solução para a limitação de alcance angular foi a inserção de um par de engrenagens. Em contrapartida, o torque e a precisão do motor foram reduzidos. Mesmo com o tamanho reduzido, o torque oferecido pelo motor servo é maior que o oferecido pelo motor de passo utilizado nos testes e a precisão, apesar de reduzida, ainda foi provada ser suficiente nos testes preliminares para esta aplicação.

De fábrica, o motor servo deixa de atender um dos requisitos de projeto do módulo, que é a capacidade de obter a posição angular do eixo e enviar esta informação para um controlador externo. É possível enviar comandos para o motor servo, mas não o contrário.

Para obter essa informação sem a adição de outros componentes ao módulo, uma modificação na parte interna do servo foi necessária.

Como explicado anteriormente, o motor servo possui um potenciômetro acoplado ao eixo, cuja função é justamente medir a posição angular. Entretanto, não se tem acesso, da parte externa, aos terminais do potenciômetro.

A modificação necessária visa possibilitar acesso externo ao terminal central do potenciômetro. Para isto, abre-se a carenagem do motor, um fio é soldado no terminal intermediário do potenciômetro e então a carenagem é fechada novamente.

Esta modificação é mostrada na Figura 2.5 abaixo. Na foto à esquerda está o esquema de ligação original e na foto à direita o esquema modificado com a adição de um fio no terminal central do potenciômetro.

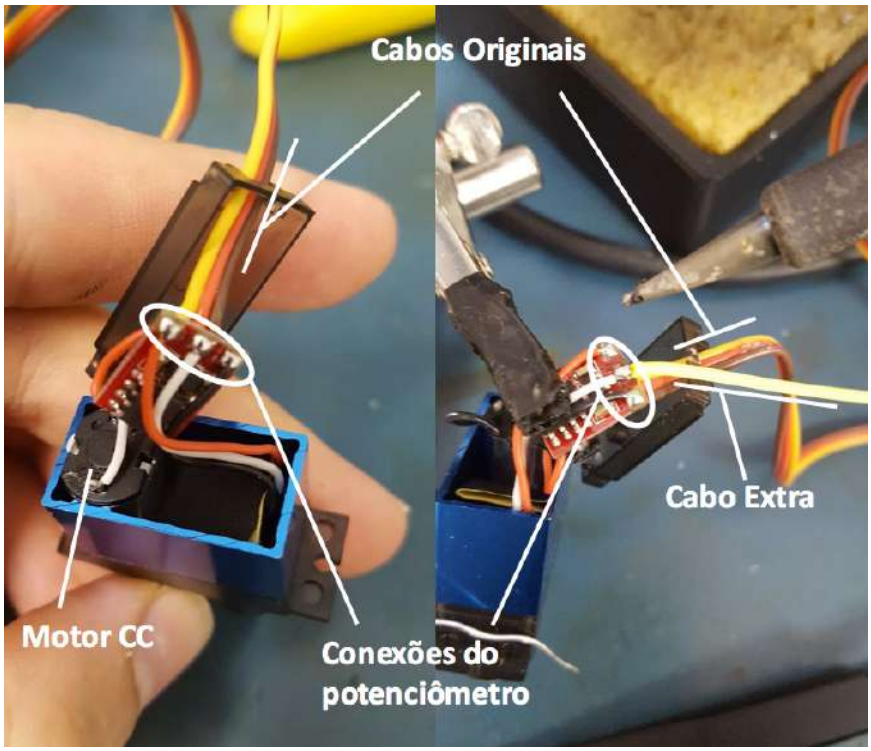

Figura 2.5: Modificação no motor servo para obter a posição angular - **Fonte:** Autor

Nos testes iniciais, foi utilizado o motor servo SG-90 da Tower Pro, mostrado na Figura 2.7 [13]. Este é um motor servo com circuito de controle analógico extremamente barato e facilmente encontrado em lojas especializadas em eletrônica.

Figura 2.6: Motor servo SG-90 - **Fonte:** Tower Pro

O motor SG-90 é capaz de rodar os potenciômetros dos amplificadores, trabalhando, porém, muito perto do seu torque máximo. O motor foi útil para a parte inicial do desenvolvimento, porém provavelmente apresentaria problemas de desgaste a curto ou médio prazo devido ao esforço excessivo, inviabilizando seu uso.

Além disso, em testes mais aprofundados, o motor apresentou precisão insatisfatória para a aplicação. Espera-se que, dentro do alcance angular do motor, tenha-se uma precisão de pelo menos 100 divisões, e ele apresenta cerca de 60.

O motor servo, teoricamente, deveria se mover não em passos discretos, mas continuamente, dependendo apenas da largura do pulso no terminal de controle. Entretanto, quando um comando para mover pequenas distâncias angulares (um grau, por exemplo) é enviado, o motor permanece parado.

Isto depende de diversos fatores, como o circuito de controle do servo, a folga existente nas engrenagens internas e a qualidade do potenciômetro. O problema com o circuito de controle do servo é quantificado e é chamado de "Banda Morta"do motor. É um valor dado em microssegundos na largura do pulso, e idealmente é o menor possível. Esta falta de precisão pode ter sido, também, agravada pelo esforço exagerado ao qual o motor estava submetido nesta aplicação.

O número de 100 divisões não é arbitrário. Parte da proposta de valor do produto é a de adicionar características dos equipamentos digitais ao equipamento analógico. Nos equipamentos digitais que simulam amplificadores analógicos as configurações de timbre possuem geralmente 100 passos. Isto significa que 100 divisões são o suficiente para discretizar o alcance angular do potenciômetro dos amplificadores de modo satisfatório para o usuário.

Logo, para as interfaces, tanto a pedaleira quanto o software, as posições angulares de cada módulo irão variar de 0 a 100. Por isto um motor que só consiga dar 60 passos não atende aos requisitos do projeto e não pode ser usado.

Adquiriu-se então o motor servo com circuito de controle digital MG92B também fabricado pela Tower Pro, mostrado na FIGURAMG92B. Este motor é mais difícil de ser encontrado, porém com desempenho superior ao do SG-90.

A Tabela 2.1 faz um comparativo entre os dois modelos de motor servo utilizados durante o desenvolvimento. É importante notar que o torque do motor MG92B é quase duas vezes maior que o torque do motor SG90 utilizado anteriormente. Isto significa que o motor MG92B trabalhará em um nível de esforço ideal, sem prejudicar a sua precisão.

A precisão do MG92B é superior ao do SG-90, como é possível notar nos valores desta e da Banda Morta. Vale ressaltar que a precisão em número de passos foi obtida com o módulo acoplado ao potenciômetro do amplificador, ou seja, submetido à esforço.

Outra vantagem do novo motor em relação ao antigo está no material de fabricação das engrenagens internas e do eixo. Sendo fabricado em alumínio, ao invés de plástico, o motor MG92B apresentará uma durabilidade maior. Mesmo possuindo essa vantagens, o MG92B possui um tamanho semelhante ao SG90, fator de extrema relevância pro projeto, e também uma velocidade

Figura 2.7: Motor servo digital MG92B - **Fonte:** Tower Pro

	SG90	**MG92B**
Torque [kg/cm]	1,8	3,1
Dimensões [mm]	23 x 12,2 x 29	22,8 x 12 x 31
Precisão [passos]	60	>100
Velocidade [segundos/60 graus]	0,12	0,13
Banda Morta [microssegundos]	10	1
Material do Eixo e Engrenagens	Plástico	Alumínio
Circuito de Controle	Analógico	Digital

Tabela 2.1: Comparativo entre os dois motores servo utilizados durante o desenvolvimento - **Fonte:** Tower Pro

semelhante, de menor relevância no contexto geral.

O cabo original com três fios do MG92B foi trocado por um cabo com quatro fios seguindo a modificação descrita anteriormente neste capítulo. Além disso ele possui, também, um comprimento maior para que alcance a conexão com a Ponte de Comando.

No total foram compradas 10 unidades deste motor para o protótipo. As outra partes do protótipo foram então projetadas para conseguir lidar com os 10 motores.

2.2.2 Protótipo do Módulo

Uma vez que o motor a ser utilizado foi escolhido, o protótipo do módulo do motor foi desenhado usando o software SoliWorks. Esta subseção trata do desenvolvimento do protótipo do módulo motorizado, abordando todas partes que o compõem.

Primeiro Protótipo

A Figura 2.8 ilustra o primeiro protótipo do módulo, onde ficam evidentes a junção de acoplamento, o conjunto de engrenagens, o *knob* para ajuste manual e o motor servo, que naquele momento ainda era o modelo SG-90.

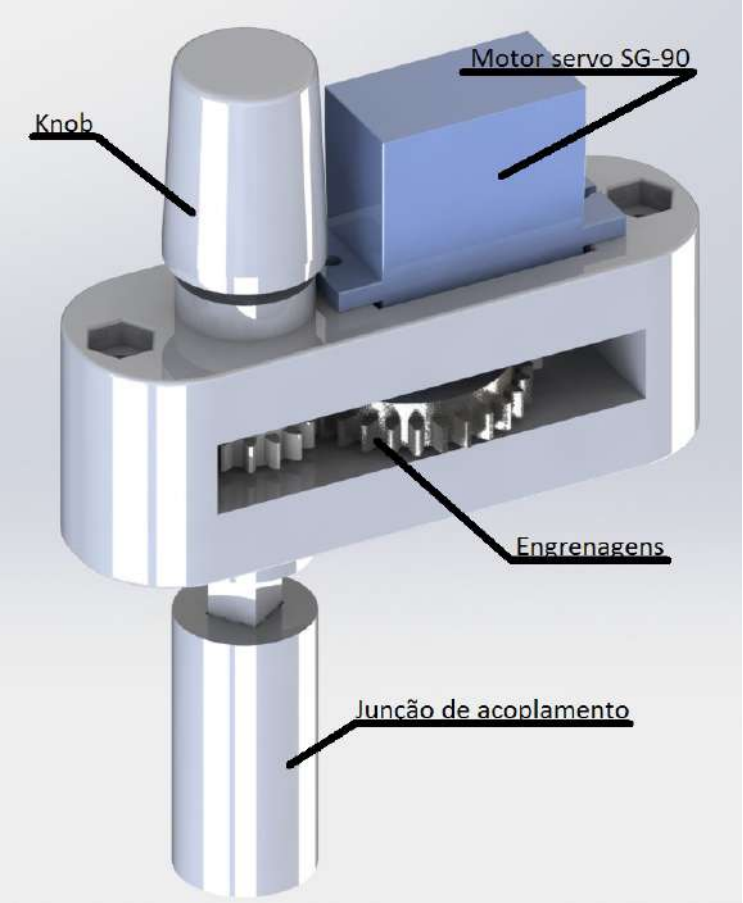

Figura 2.8: Desenho do primeiro protótipo do módulo motorizado - **Fonte:** Autor

Neste protótipo o eixo também foi projetado para ser impresso em três dimensões. Entretanto o conjunto montado se mostrou bastante instável e impreciso, principalmente devido à falta de um eixo e um mancal adequados.

O mancal é o dispositivo mecânico onde se apoia um eixo girante e, neste caso, era um buraco redondo na própria peça da estrutura. A resolução deste problema será abordada posteriormente nesta mesma subseção.

A maior parte das peças do protótipo do módulo motorizado foram fabricadas em plástico PLA biodegradável utilizando uma impressora 3D. São elas: o conjunto de engrenagens, as partes que compõe a estrutura do módulo e um *knob* que será utilizado no ajuste manual das configurações do amplificador.

O método de impressão em três dimensões por extrusão de plástico foi escolhido por oferecer enorme liberdade no projeto das peças a baixo custo, considerando possuir uma impressora deste

tipo em casa.

Entretanto, existem algumas limitações ao se trabalhar com impressões em três dimensões por extrusão de plástico. A principal delas é a resolução da impressora com a qual se está trabalhando que será um fator limitante para o tamanho dos detalhes de cada peça.

Alguns testes foram feitos para quantificar as limitações da impressora. A Figura 2.9 mostra um teste que consistia em imprimir uma peça com quatro buracos e quatro pinos cilíndricos de diâmetros distintos e conhecidos. Então, utilizando um paquímetro digital, mede-se o diâmetro dos buracos e pinos impressos.

Figura 2.9: Foto das peças impressas para o teste de precisão - **Fonte:** Autor

Analisando os resultados, pôde-se constatar que os diâmetros impressos eram em média 0,15 mm menores nos buracos e nos pinos em comparação com os diâmetros projetados.

Com esta informação, sabe-se que nos diâmetros do projeto deve ser considerada uma tolerância de 0,15 mm (Tabela 2.2). Isto diminui a possibilidade de imprimir peças com as dimensões erradas economizando tempo e recursos. Ainda sim, testes baseados em tentativa e erro não são dispensáveis.

Conjunto de Engrenagens

O conjunto de engrenagens foi a parte da fabricação que apresentou maior grau de dificuldade. Idealmente, as engrenagens teriam o menor tamanho possível, já que o tamanho dos módulos é um fator importante para o projeto, entretanto a resolução da impressora limita o tamanho mínimo deste conjunto devido à complexidade dos detalhes. Uma engrenagem maior em diâmetro pode ter dentes maiores que necessitam de menor resolução.

Diâmetro Projetado	Diâmetro Impresso
Buraco 8 mm	7.85 mm
Buraco 7 mm	6.83 mm
Buraco 6 mm	5.85 mm
Buraco 5 mm	4.86 mm
Pino 8 mm	7.87 mm
Pino 7 mm	6.86 mm
Pino 6 mm	5.83 mm
Pino 5 mm	4.84 mm

Tabela 2.2: Tabela com os resultados do teste de precisão - **Fonte:** Autor

A partir das limitações da impressora, chegou-se ao conjunto de engrenagens composto por uma engrenagem com 9 dentes e 12 mm de diâmetro externo e uma engrenagem com 24 dentes e 26 mm de diâmetro externo. A partir destas dimensões, projetou-se mais dois conjuntos de engrenagens: um com 0.2 mm e outro com 0.4 mm a mais nos diâmetros das engrenagens.

Os três conjuntos foram impressos e foram testados na montagem do módulo interpolando as diferentes combinações possíveis entre as engrenagens dos três conjuntos. O objetivo desta interpolação foi achar a combinação que apresentasse a menor folga possível sem oferecer resistência em excesso para o motor. O conjunto que apresentou o melhor resultado foi o composto pelas duas engrenagens com 0,2 mm a mais.

O tamanho final das engrenagens gera uma contrapartida: o diâmetro externo da engrenagem de 24 dentes é maior que a largura do módulo, logo se torna o componente limitante na proximidade mínima entre os módulos. Como já mencionado, a proximidade entre os módulos deve ser a menor possível, então esta contrapartida deve ser resolvida.

Projetaram-se então dois conjuntos de engrenagens com diferentes alturas no encaixe com o eixo e a altura do vão da estrutura do módulo foi aumentado. Alternar estes conjuntos nos módulos vizinhos resulta no desencontro das engrenagens e permite que os módulos encostem um no outro sem interferir nas partes móveis. A Figura 2.10 mostra o resultado desta modificação.

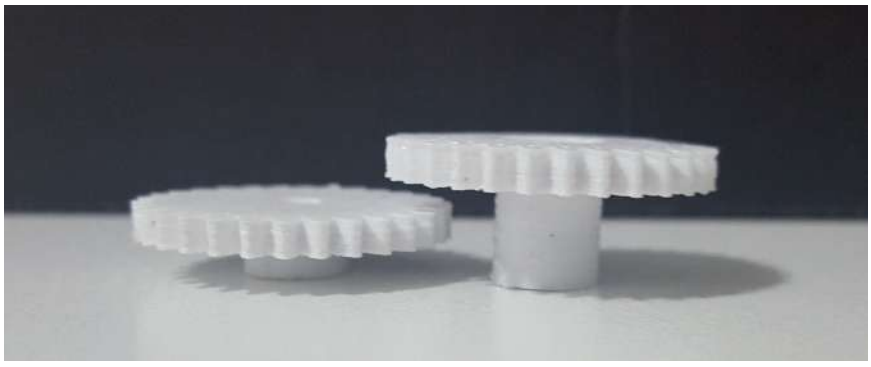

Figura 2.10: Engrenagens de 24 dentes de conjuntos com alturas diferentes - **Fonte:** Autor

A Figura 2.8 mostra o desenho deste protótipo. É possível identificar o motor servo SG-90, o sistema de engrenagens, a extremidade do eixo que se conecta ao amplificador com a junção de

acoplamento, e a extremidade do eixo com o *Knob* para o ajuste manual. A conexão com a barra metálica do suporte deverá ser parte deste módulo, porém ainda não foi desenhada.

Eixo e Mancal

A fabricação destes componentes utilizando impressão em três dimensões não forneceu bons resultados, como relatado anteriormente. O conjunto todo apresentou folgas e imprecisões que comprometeriam a precisão e o funcionamento do sistema.

Idealmente, estas peças seriam fabricados utilizando alguma liga metálica e processos mais precisos, porém, em baixa quantidade, representariam um custo muito elevado para este protótipo.

Visando então o baixo custo e a velocidade no processo de prototipagem deste produto, foi decidido utilizar eixos e mancais compatíveis da marca de brinquedos de montar Lego. As peças destes brinquedos possuem uma ótima precisão e são muito utilizadas em projetos de robótica no mundo inteiro.

Os eixos possuem uma seção transversal no formato do símbolo de adição "+", facilitando a transeferência de torque para a engrenagem, o *knob* e a junção de acoplamento de eixos.

A Figura 2.11 abaixo mostra os mancais e o eixo utilizados em cada módulo motorizado. Idealmente, todos os mancais seriam iguais ao mancal à esquerda na Figura 2.11, pois é o mais compacto. Entretanto, devido à indisponibilidade deste tipo de peça para as dez unidades de módulos, outro mancal, à direita na Figura 2.11, foi utilizado em conjunto com o primeiro.

Figura 2.11: Eixo e mancais utilizados nos módulos motorizados - **Fonte:** Lego

O mancal mais compacto é utilizado na parte da estrutura em que o espaço livre é menor, ou seja, onde o motor servo é preso. A parte oposta não possui tantas limitações em relação ao espaço, então o mancal menos compacto pôde ser usado sem problemas.

Suporte Mecânico

Um dos grandes desafios mecânicos do projeto do *RoadieBot* é o suporte que sustenta os motores acoplados ao amplificador. Abaixo, os dois principais requisitos do suporte são esclarecidos.

Primeiramente, o suporte deve ser capaz de sustentar todos os motores de modo que estes não exerçam nenhum esforço no eixo dos potenciômetros do amplificador. O usuário deve ter a garantia de que nenhum componente do seu amplificador será danificado, já que os equipamentos analógicos deste tipo possuem, em geral, um preço bastante elevado.

Em segundo lugar, o suporte deve ser adaptável aos diferentes formatos de amplificador presentes no mercado. A confecção de suportes personalizados tornaria o produto caro demais e inviabilizaria o projeto.

Pensando nestes requisitos, foi idealizado um suporte para os módulos motorizados. Uma barra metálica sustentará os módulos com os motores no painel frontal do amplificador. A barra utilizada é uma barra de aço inoxidável de 15,7 mm de diâmetro comprada em uma loja especializada. Os módulos podem deslizar pela barra até serem afixados nas posições corretas dependendo da disposição dos potenciômetros do amplificador a ser automatizado.

Para sustentar esta barra, inicialmente se pensou em utilizar uma cinta de amarração com tensionador, semelhante àquelas usadas para prender cargas. Duas peças fabricadas com impressão em três dimensões fariam a junção da barra com a cinta.

A cinta permite a adaptação do suporte a qualquer tipo de amplificador sem elevar excessivamente o preço de fabricação do produto, e a barra metálica oferece a sustentação para os módulos.

A Figura 2.12 mostra uma cinta de amarração de náilon da marca Vonder com o tensionador estilo catraca. Este tipo de cinta e tensionador permitiria a fixação do suporte nos mais variados formatos de amplificadores do mercado de modo rápido e prático.

Entretanto, este tipo de tensionador pode não ser necessário e iria encarecer o produto. Uma alternativa seria utilizar um sistema de fivelas semelhante ao presente em mochilas para tensionar a cinta. Este sistema possui um custo bastante inferior ao custo do sistema do tipo catraca, entretanto, não se sabe se conseguirá tensionar a cinta o suficiente para ser capaz de sustentar o suporte com os motores.

A Figura 2.13 gerada no software de modelagem em três dimensões ilustra como seria o suporte sustentado pela cinta. A figura não mostra a cinta, pois ainda não se sabe qual tensionador será usado. Este trabalho não aborda a escolha deste tencionador por priorizar os outros aspectos do protótipo.

Já que o tencionador da cinta ainda não foi testado e escolhido, uma alternativa teve que ser implementada. Um pedestal de prato de uma bateria acústica foi utilizado para sustentar a barra, como ilustrado na Figura 2.14. Este tipo de pedestal foi escolhido por permitir a regulagem de altura necessária e por sua robustez.

Figura 2.12: À esquerda, cinta com tensionador do tipo catraca e, à direita, cinta com tensionador do tipo fivela - **Fonte:** Vonder e Tao Bao

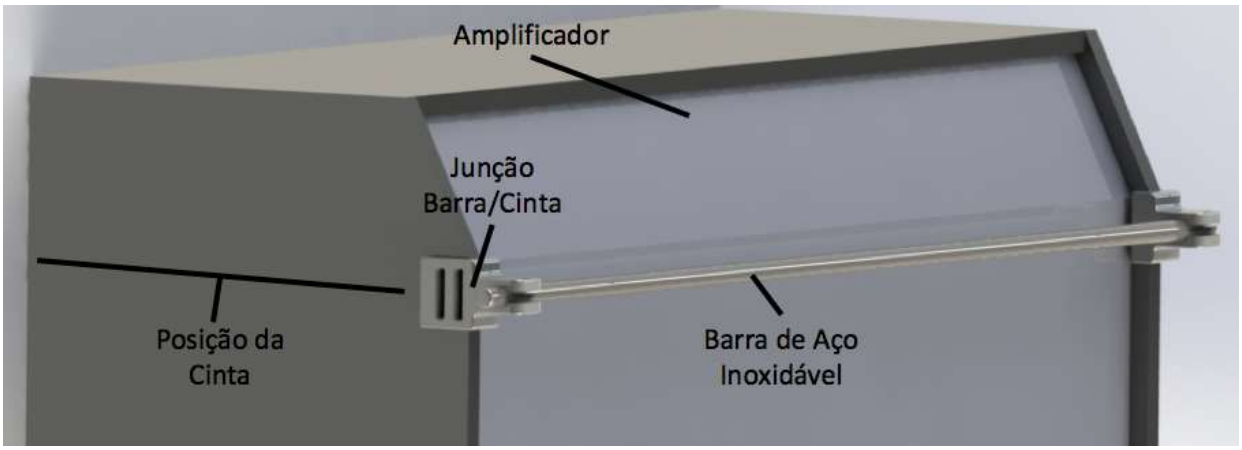

Figura 2.13: Suporte dos módulos motorizados no amplificador - **Fonte:** Autor

Como mencionado anteriormente, esta será a alternativa usada para a realização do protótipo. Entretanto, deve-se implementar o método que utiliza a cinta, e testar o melhor sistema tensionador para o desenvolvimento do produto, pois este método ofereceria maior praticidade para o usuário.

Knob e Junção de eixo

Outro ponto importante é a conexão do eixo do módulo com o eixo do potenciômetro do amplificador. Como descrito anteriormente, cada amplificador possui *Knobs* acoplados ao eixo de cada um dos seus potenciômetros. Estes *Knobs* possuem os mais variados formatos, e projetar uma peça capaz de se conectar a qualquer um destes formatos seria extremamente difícil.

Então, para utilizar o protótipo *RoadieBot*, o usuário deverá remover os *Knobs* dos potenciômetros que deseja automatizar. Assim uma simples junção de eixos apertada por um parafuso é suficiente para acoplar os módulos ao amplificador.

Figura 2.14: Suporte provisório dos módulos motorizados utilizando um pedestal de bateria - **Fonte:** Autor

No eixo, na ponta oposta à ponta da junção de acoplamento fica acoplado o *knob* para o ajuste manual das configurações do amplificador. Vale ressaltar que será necessário um torque maior do que o original para configurar o amplificador manualmente.

Isto se deve ao fato de o eixo estar acoplado ao motor servo. Além do conjunto de engrenagens responsável por este acoplamento, o próprio motor servo possui uma caixa de redução. Sendo assim, para que haja rotação, a inércia e, principalmente, o atrito do sistema devem ser vencidos.

Ainda sim, este esforço extra não prejudica a regulagem manual das configurações do amplificador.

Protótipo Final

Compilando todas as decisões descritas nas subseções anteriores, chegou-se ao protótipo de módulo motorizado ilustrado na Figura 2.15.

Em comparação ao protótipo anterior, este possui uma série de melhorias. Graças aos conjuntos de engrenagens em diferentes alturas e a remoção de um dos parafusos de fixação da estrutura, o tamanho do módulo pôde ser reduzido de modo que ainda não foi encontrado um amplificador com o qual os módulos não sejam compatíveis.

A remoção de um dos parafusos é importante para os usuários que possuem um amplificador com duas fileiras de potenciômetros. Nestes casos, uma fileira de módulos fica posicionada normalmente e a de cima fica posicionada de cabeça para baixo. O parafuso que foi retirado limitava a distância entre os módulos nesta disposição.

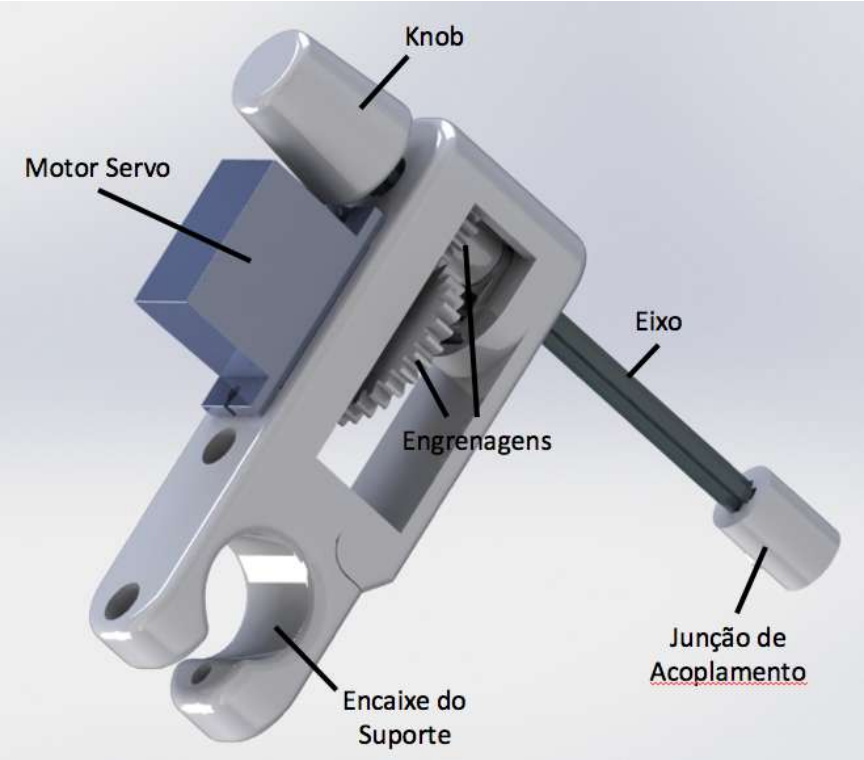

Figura 2.15: Desenho do protótipo final do módulo motorizado - **Fonte:** Autor

A Figura 2.16 mostra a diferença na distância entre os módulos para os dois protótipos. O parafuso aumentava a distância de um módulo até o outro em 16mm.

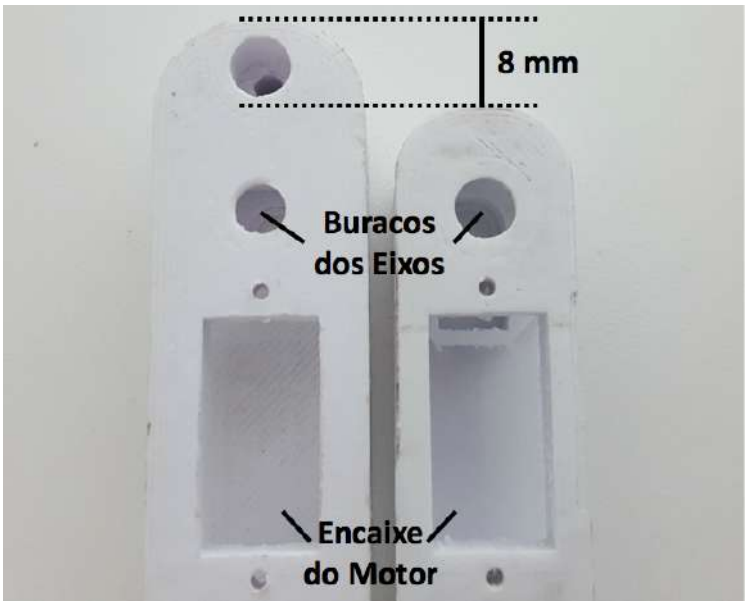

Figura 2.16: Diferença entre os protótipos na distância vertical mínima entre os módulos - **Fonte:** Autor

Além do tamanho reduzido, a utilização do eixo e mancais Lego diminuiu consideravelmente as folgas do sistema, resultando em um módulo mais robusto e confiável.

O primeiro protótipo ainda não continha a parte destinada ao acoplamento com o suporte em

contrapartida com o protótipo da Figura 2.15.

A barra de aço inoxidável entra pela cavidade da abraçadeira para o suporte, o módulo é posicionado de acordo com a disposição dos potenciômetros do amplificador e então o parafuso da braçadeira é tensionado até que o módulo esteja estável.

2.3 PEDALEIRA DE CONTROLE E PONTE DE COMANDO

Esta seção trata dos aspectos que envolvem o desenvolvimento do protótipo da Pedaleira de Controle e da Ponte de Comando. Serão abordados a escolha.

2.3.1 Microcontroladores

Por se tratar de um protótipo, serão usados como microcontroladores diferentes versões da plataforma de prototipagem Arduino, o qual é uma plataforma eletrônica de *hardware* livre focada na versatilidade e praticidade na realização de projetos com microcontroladores [14].

Existem diferentes versões de placas Arduino, variando em tamanho, quantidade de entradas/-saídas e disponibilidade de recursos. As placas são projetadas com microcontroladores AVR desenvolvidos pela Atmel e são programadas em uma linguagem específica, essencialmente C/C++, utilizando o software Arduino IDE e, na maior parte dos casos, conexão via cabo USB.

Inicialmente, tinha sido considerado o uso da comunicação I^2C entre a pedaleira e os módulos motorizados, como relatado na seção 2.1, mas esta decisão foi descartada por depender do uso de um microcontrolador por módulo motorizado e , consequentemente, aumentando o custo do projeto.

Nesta mesma seção, foi decidido inserir um novo elemento, chamado Ponte de Comando, responsável pela comunicação com a pedaleira e pelo controle dos módulos motorizados.

2.3.1.1 Ponte de Comando

A placa Arduino Mega 2560 fará o papel do microcontrolador no protótipo da Ponte de Comando, vista na Figura 2.17, se comunicando com a Pedaleira de Controle e controlando os módulos motorizados.

O grande fator limitante na escolha entre as versões disponíveis da placa Arduino para a Ponte de Comando é a quantidade de entradas com conversor AD que a placa possui. Estas entradas, geralmente em menor quantidade comparado às entradas digitais, são as responsáveis por medir a tensão no pino intermediário do potenciômetro do motor servo, logo cada módulo motorizado necessita de uma porta analógica.

Além da entrada analógica, é necessária também uma porta digital por módulo motorizado.

Figura 2.17: Arduino Mega 2560 - **Fonte:** Embarcados

Estas portas serão usadas para gerar o trem de pulsos responsável por controlar a posição do motor servo.

A tabela 2.3 evidencia algumas das especificações Arduino Mega 2560. Esta versão do Arduino foi escolhida por ser capaz de controlar todos os dez módulos motorizados sozinha, simplificando o protótipo.

	Arduino Mega 2560
Microcontrolador	ATmega2560
Pinos Digitais E/S	54
Entradas Analógicas	16
SRAM	8 kB
EEPROM	4 kB
Memória Flash	256kB

Tabela 2.3: Especificações da placa Arduino Mega 2560 - **Fonte:** Arduino

Na seção 2.1 deste trabalho foi descrito que a Ponte de Comando do produto deveria ser modular, ou seja, mais de uma Ponte poderia ser utilizada caso o usuário estivesse usando um número de Módulos Motorizados maior que o número máximo permitido por Ponte.

Entretanto, essa capacidade ainda não será implementada neste protótipo e por isso o Arduino Mega 2560 foi escolhido. Para o produto um microcontrolador menor e mais barato seria utilizado e, se nessário, o usuário adquiriria outra Ponte de Comando.

O Arduino Mega 2560 foi a solução mais rápida e barata para produzir o protótipo e validá-lo como produto.

2.3.1.2 Pedaleira de Controle

Para a pedaleira de controle, uma placa Arduino mais compacta pode ser usada, já que não existe a necessidade de tantas portas analógicas. A versão escolhida foi o Arduino Nano que pode ser visto na Figura 2.18.

Esta placa será responsável pela comunicação com a Ponte de Comando, pela conexão com o computador contendo o software da interface e pela interface da própria Pedaleira de Controle.

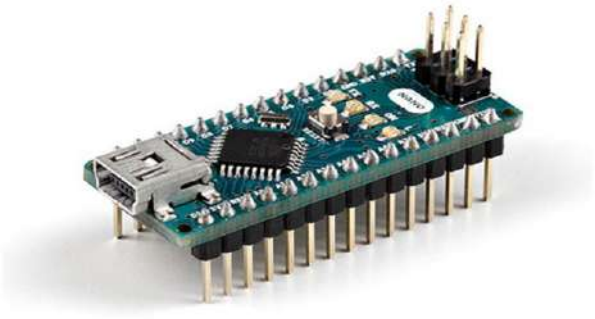

Figura 2.18: Arduino Nano - **Fonte:** Baú da Eletrônica

A tabela 2.4 lista as principais especificações desta placa que é, no geral, inferior à placa Arduino Mega 2560, mas suficiente para a aplicação na Pedaleira de Controle.

O Arduino Nano só não foi usado na Ponte de Comando por não possuir entradas analógicas em número suficiente para controlar os dez módulos motorizados que serão fabricados para o protótipo.

	Arduino Nano
Microcontrolador	ATmega328
Pinos Digitais E/S	22
Entradas Analógicas	8
SRAM	2 kB
EEPROM	1 kB
Memória Flash	32kB

Tabela 2.4: Especificações da placa Arduino Nano com o microcontrolador ATmega328 - **Fonte:** Arduino

2.3.2 Interface da Pedaleira de Controle

A interface na Pedaleira de Controle será composta por uma tela LCD, um *encoder* com botão e dois botões acionados com o pé, para que o músico possa controlar o protótipo enquanto toca. Esta interface permitirá que o usuário controle os Módulos Motorizados individualmente e acesse ou armazene configurações completas do seu amplificador.

2.3.2.1 Tela

A tela utilizada neste projeto é uma tela LCD de 16x2. Isto significa que a tela possui tamanho suficiente para mostrar dezesseis caracteres por linha em duas linhas. A Figura 2.19 mostra a tela utilizada neste protótipo.

Figura 2.19: Tela LCD 16x2 que será utilizada na interface da pedaleira de controle - **Fonte:** FilipeFlop

Esta tela possui 16 conexões, sendo que seis destas estarão ligadas direto ao microcontrolador. Além disso a tela necessita de um potenciômetro para realaizar a regulagem de contraste [8].

2.3.2.2 *Encoder*

O *encoder* é um dispositivo capaz de medir movimento ou posição angular (7). Possui o funcionamento semelhante ao de um potênciometro, porém pode rotacionar infinitamente e possui posições definidas dentro de uma volta completa.

O modelo utilizado neste protótipo, visto na Figura 2.2, possui vinte posições por volta e um botão embutido no seu eixo. Além disso, vem montado em um módulo que visa facilitar o seu uso em prototipagem, sendo o mais apropriado para este projeto.

Este componente irá permitir que o usuário navegue pelas opções de controle mostradas na tela LCD.

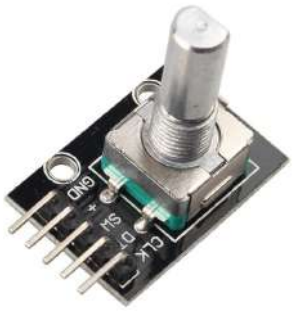

Figura 2.20: Módulo com *encoder* de vinte posições que será utilizado no protótipo da Pedaleira de Controle - **Fonte:** HU infinito

2.3.2.3 Botões

Muito comuns em equipamentos para músicos, botões próprios para serem acionados com o pé, chamados de *footswitch*, serão utilizados na Pedaleira de Controle para que o músico possa navegar enquanto toca entre as configurações previamente armazenadas.

Este tipo de botão possui uma construção robusta, feita especialmente para estas aplicações. A Figura 2.2 mostra o modelo de *footswitch* ideal para o produto.

Figura 2.21: *Footswitch* que será usado na interface da Pedaleira de Controle - **Fonte:** TiggerComp

2.3.3 Cabo de Comunicação

Um ponto que merece atenção é a escolha do cabo e terminais que conectam a Ponte de Comando à pedaleira de controle. A comunicação utilizando a SCI necessita de três condutores para funcionar: a linha Rx, a linha Tx e a tensão de referência.

Do ponto de vista do produto, é interessante que o usuário possa utilizar um cabo com o comprimento compatível com as suas necessidades. Um músico que tocar em palcos pequenos necessita de um cabo mais curto que um músico que toca em festivais ou um engenheiro de gravação. Por isto a necessidade de conectores, ao invés de uma instalação fixa.

Decidiu-se utilizar conectores compatíveis com os cabos já utilizados nestes ambientes para

facilitar a adaptação do produto nas mais diversas situações. O cabo perfeito para esta aplicação é aquele com conectores XLR utilizado em microfones. A Figura 2.22 mostra este tipo de cabo.

Figura 2.22: Cabo para microfones com conectores XLR - **Fonte:** PlayTech

Além de ser um cabo que possui três condutores, ele está presente na maioria, senão todas, as apresentações musicais ao vivo, sendo extremamente fácil encontrá-lo nos mais diversos comprimentos em qualquer loja, além de ser usado para conectar os sinais entre as salas nos estúdios de gravação.

3 HARDWARE

Neste capítulo será descrita a montagem do protótipo considerando os elementos que, no capítulo 2, foram discutidos individualmente.

Tanto a Pedaleira de Controle quanto a Ponte de Comando foram montados em *protoboards* para acelerar o processo de prototipagem e economizar recursos.

3.1 PEDALEIRA DE CONTROLE

O *hardware* da Pedaleira de Controle consiste basicamente nas ligações necessárias para a interface funcionar.

A Figura mostra uma foto da montagem da Pedaleira de Controle indicando cada elemento.

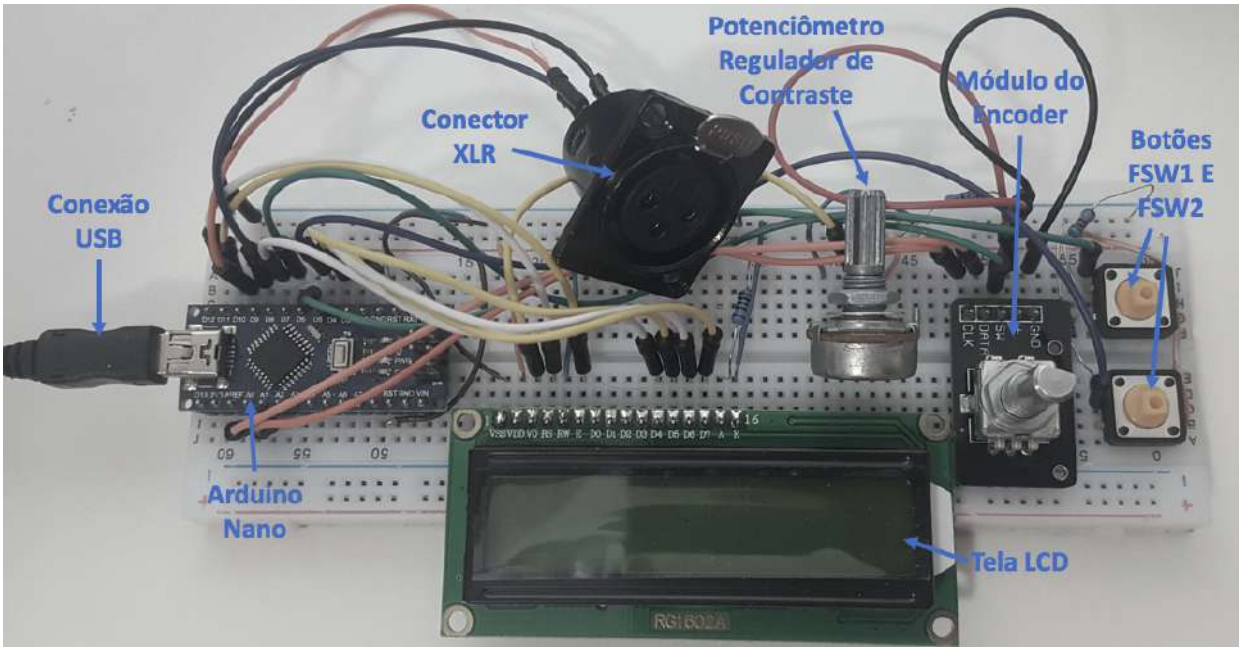

Figura 3.1: Foto do protótipo da Pedaleira de Controle montada em *protoboard* - **Fonte:** Autor

O Arduino Nano, além de controlar o sistema, também possibilita a comunicação serial com o Software no computador por meio da conexão USB. Além disso, esta conexão também é responsável por alimentar o conjunto.

O conector XLR de três pinos permite a comunicação serial com a Ponte de Comando utilizando um cabo para microfone.

A tela LCD ocupa boa parte das portas digitais do Arduino. São usadas seis portas do microcontrolador somente para esta função. Além disso a tela necessita de um potenciômetro utilizado

para regular o contraste.

O módulo do *Encoder* utiliza três portas digitais do Arduino Nano: duas para a posição angular(*Encoder* em si); e uma para o botão. O botão necessita de um resistor *pull-up* no seu terminal.

Por fim, os botões FSW1 e FSW2. Ambos necessitam de resistores *pull-up* e cada um ocupa uma porta digital. Estes botões comuns foram utilizados no lugar dos *footswitches*, para facilitar os testes e prototipagem. O *footswitch* necessita de uma embalagem rigída para ser montado.

3.2 PONTE DE COMANDO

A montagem da Ponte de Comando apresentou um grande problema com relação à tensão de alimentação.

Nos primeiros testes, o uso de vários motores gerava quedas de tensão grandes o suficiente para reiniciar o microcontrolador. Estas quedas ocorrem devido a picos de corrente na partida do motor ou em situações de alto esforço.

A primeira medida tomada foi a de adicionar capacitores de desacoplamento de 100nF nos terminais de alimentação de cada motor. Constatando-se a não suficiência desta medida, adicionou-se um banco de capacitores com capacidade de carga maior. Apesar da queda de tensão diminuir, o microcontrolador continuou reiniciando quando os motores eram ativados.

A solução temporária encontrada foi utilizar fontes de alimentação separadas: uma para o Arduino e outra para o resto do circuito. Esta medida foi tomada por falta de tempo para projetar um desacoplamento eficiente entre as duas partes do circuito, mas será o foco de desenvolvimentos futuros.

A Figura 3.2 mostra o esquemático do circuito para a conexão de um motor isoladamente. Os capacitores C1 e C2 são individuais para cada motor enquanto o capacitor Cb representa o banco de capacitores.

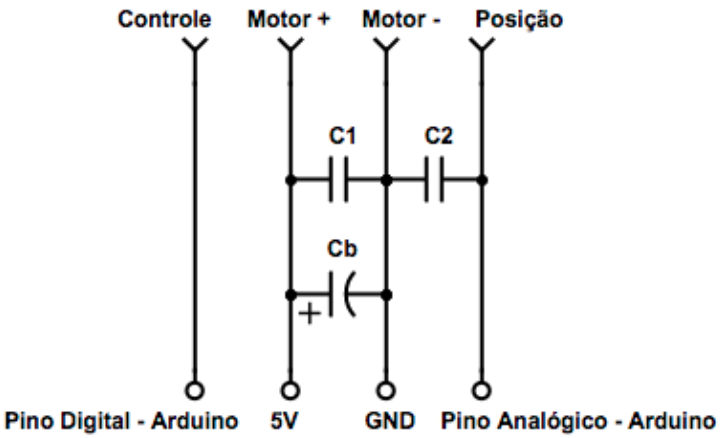

Figura 3.2: Esquemático do circuito de cada motor isoladamente - **Fonte:** Autor

Os capacitores C1 e Cb têm função de desacoplamento e foram mantidos, mesmo após a escolha por usar duas fontes. O capacitor C2 tem função de estabilização. Ele aumenta a precisão da leitura da tensão no terminal do potenciômetro do motor.

Esse circuito foi repetido dez vezes, uma para cada motor, e montado na *protoboard*.

A Figura 3.3 mostra uma foto da montagem do protótipo da Ponte de Comando com os seus principais elementos indicados.

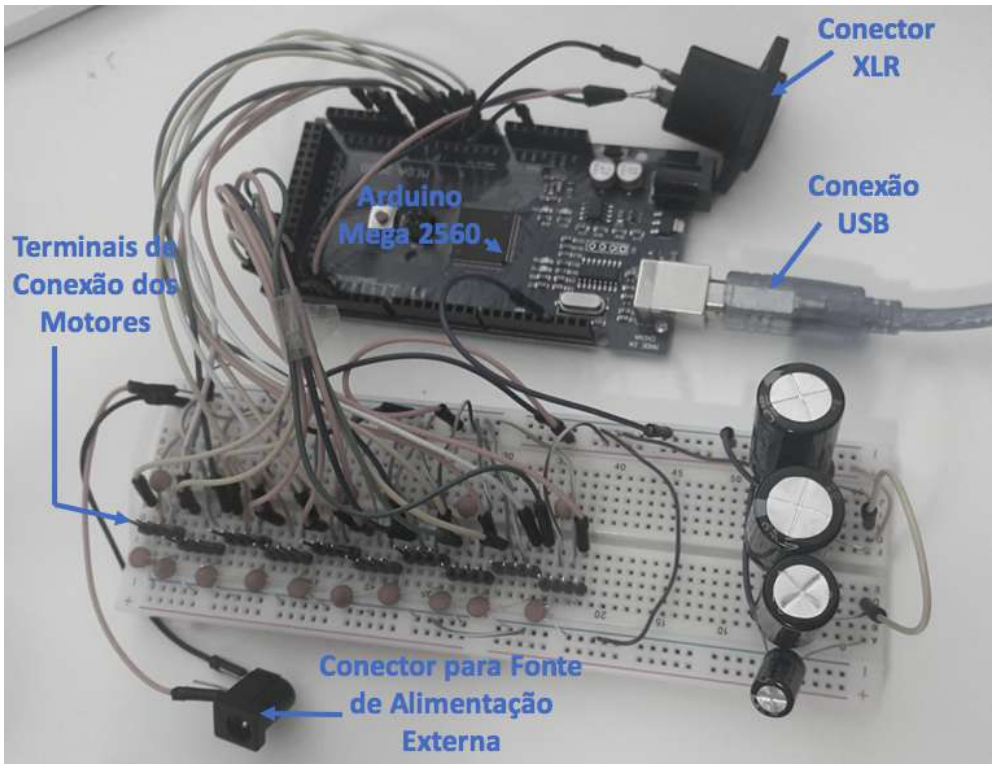

Figura 3.3: Foto do protótipo da Ponte de Comando montada em *protoboard* - **Fonte:** Autor

O Arduino Mega 2560 comanda todos os motores a partir dos comandos que recebe. É alimentado separadamente do resto do circuito pela conexão USB.

O conector XLR é usado na comunicação entre a Ponte de Comando e a Pedaleira de Controle utilizando um cabo de microfone.

Os terminais de conexão dos motores são os terminais do esquemático da Figura 3.2.

Por último, a fonte de tensão que alimenta os motores é acoplada ao conector para fonte de alimentação externa.

3.3 MÓDULO MOTORIZADO

Após a fabricação das peças, os módulos foram montados. A Figura 3.4 mostra um destes. Na foto, faltam a junção de acoplamento e o *knob*.

Figura 3.4: Foto do módulo motorizado fabricado e montado - **Fonte:** Autor

4 FIRMWARE

Este capítulo trata do *firmware* programado em cada microcontrolador presente no projeto.

Primeiramente, os comandos serão abordados, um por um, na seção 4.1.2. São estes os comandos que serão enviados, tanto da Pedaleira de Controle quanto do Software para a Ponte de Comando.

Antes de abordar o *firmware* de cada parte do protótipo, será abordado o formato do pacote de dados que será enviado pela comunicação serial. Este formato será o padrão para a comunicação de comandos entre o Software, a Pedaleira de Controle e a Ponte de Comando.

No pacote de dados existem dois tipos de dado: o comando e o dado secundário. Alguns comandos necessitam de uma informação além do comando em si. É o caso do comando `mover` por exemplo. Este comando necessita da informação da posição angular para a qual o motor deve se mover e esta informação é enviada no pacote de dados como dado secundário.

A Figura 4.1 ilustra como estes dados serão enviados. Para os comandos que não necessitam de informação secundária (Figura 4.1,letra A), o pacote de dados é formado por um prefixo "c"e um número inteiro respectivo ao comando. Para o caso de comandos que necessitam de um dado secundário (Figura 4.1, letra B), acrescido ao pacote do caso anterior, têm-se um prefixo "s"seguido de um número inteiro respectivo à informação secundária.

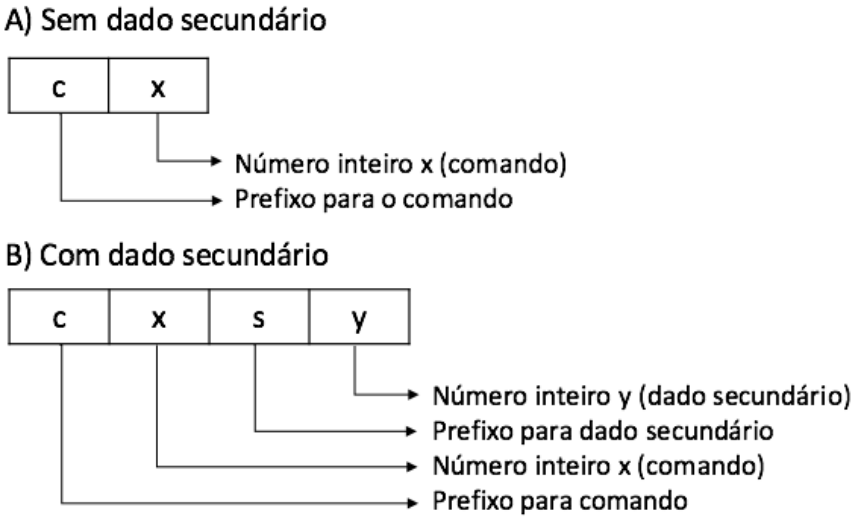

Figura 4.1: Formato do pacote de dados - **Fonte:** Autor

Cada comando é representado por um número inteiro pré-definido. Isto visa diminuir a quantidade de informação sendo transmitida pela porta serial. A Tabela 4.1 lista todos os comandos e seus respectivos números que serão transmitidos.

A função de cada comando e o seu funcionamento serão explicados individualmente nas seção 4.1.2.

Comando	Código
Mover	de 1 até 40
Ler	de 51 até 90
Attach	93
Detach	94
Attach_all	95
Detach_all	96
AutoCalibragem	99

Tabela 4.1: Lista de comandos e seus respectivos códigos - **Fonte:** Autor

4.1 PONTE DE COMANDO

O *Firmware* da Ponte de Comando (Anexo A) é responsável por interpretar e executar os comandos enviados pela Pedaleira de Controle e pelo Software.

4.1.1 Funcionamento Geral

Assim que o microcontrolador da Ponte de Comando é inicializado, ele se mantém à espera de um comando. Este comando pode vir tanto do Software quanto da Pedaleira de Controle, apesar de ambos virem pela mesma porta serial.

Assim que o prefixo indicando um comando é identificado na comunicação serial, o microcontrolador lê o próximo número inteiro, que corresponde ao código do comando. Então o comando correspondente ao código é acionado.

A Figura 4.2 esquematiza o funcionamento do *Firmware* da Ponte de Comando.

Após o término dos processos que cada comando envolve, a Ponte de Comando volta ao estado inicial à espera de um novo comando.

A seguir, cada comando é abordado individualmente.

4.1.2 Comandos

4.1.2.1 `Ativar`

Mesmo que submetido à tensão de alimentação, o motor servo permanece desativado até que receba um trem de pulsos no seu terminal de controle.

Este comando tem por finalidade dar início ao trem de pulsos que comanda o motor servo. O comando `Ativar` é necessário para que o motor se mova, logo, está inserido dentro do comando `Mover`.

Além do código do comando, o número correspondente ao motor que deve ser ativado é enviado como dado secundário.

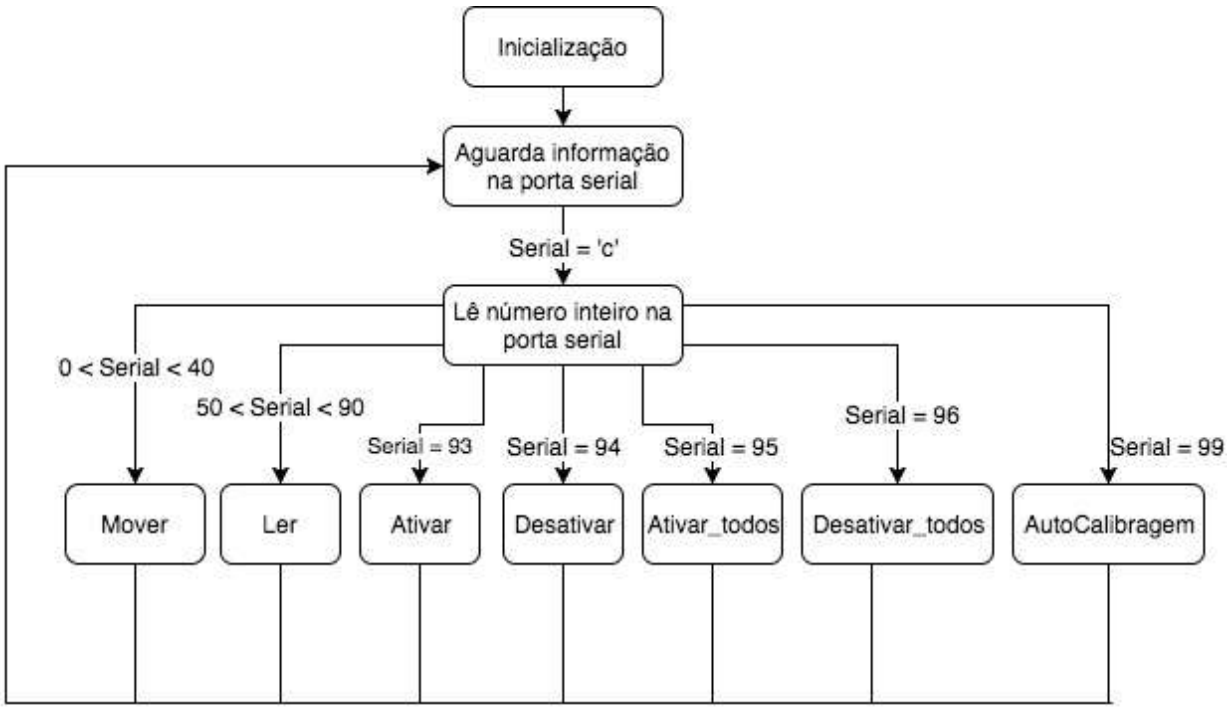

Figura 4.2: Diagrama de funcionamento do código da Ponte de Comando - **Fonte:** Autor

4.1.2.2 `Ativar_todos`

Este comando realiza a mesma função do comando `Ativar`, entretanto, ao invés ativar somente um servo, o comando `Ativar_todos` ativa todos os motores servos conectados à Ponte de Comando.

A principal vantagem deste comando é diminuir a quantidade de comandos enviados nas situações em que vários motores devem ser movidos. Caso não existisse, um comando `Ativar` teria que ser enviado para cada motor servo.

4.1.2.3 `Desativar`

Possui propósito oposto ao do comando `Ativar`. Desativa o trem de pulsos que é enviado para um motor servo específico.

O principal propósito de desativar um motor servo é o de permitir o ajuste manual das configurações do amplificador. Se o motor estiver ativado, tentará se manter na posição comandada pelo trem de pulsos, e, por isso, impede o ajuste manual.

A estrutura do comando é semelhante ao caso do comando `Ativar`. Possui código de comando e número do motor como dado secundário.

4.1.2.4 `Desativar_todos`

Este comando desativa o trem de pulsos de todos os motores servo conectados à Ponte de Comando, de maneira oposta ao comando `Ativar_todos`.

Não possui dado secundário, somente código de comando.

Após dois minutos de ociosidade(nenhum comando recebido), a Ponte de Comando, automaticamente, desativa todos os motores. Deste modo, o usuário não precisa se preocupar em desativar os motores antes de fazer um ajuste manual

4.1.2.5 `Mover`

O comando `Mover` é o responsável por controlar a posição dos motores servo. É um comando que necessita de dois dados: o número do motor servo e a posição para qual este deve se mover.

O número do servo a ser controlado corresponde ao próprio código do comando. Ou seja, para mover o servo "x", sendo "x"um número inteiro, deve-se enviar o comando de código "x".

Como pode ser visto na Tabela 4.1, o código para o comando `Mover` varia de 1 a 40. Sendo assim, considerando o código deste comando isoladamente, até 40 motores podem ser controlados.

A posição, necessária para o comando, é enviada como dado secundário, variando de 0 a 100, onde 0 equivale à posição angular máxima no sentido anti-horário (mínimo do potenciômetro) e 100 à posição angular máxima no sentido horário (máximo do potenciômetro).

Na Ponte de Comando, o valor da posição de 0 a 100 é convertido, então, para um valor de 180 a 0. A função que controla motores servo na linguagem de programação do software Arduino IDE necessita de uma entrada de 0 a 180 graus. Além disso, a inversão de 0 a 180 para 180 a 0 é necessária para compensar a inversão do sentido de rotação causado pelo conjunto de engrenagens.

A partir disto, a Ponte de Comando já possui os dados necessários para controlar o motor, mas antes, utiliza o comando `Ativar` para dar início ao trem de pulsos.

4.1.2.6 `Ler`

O comando `Ler` é responsável por obter a posição angular do eixo do motor. O único dado necessário para o comando é o número do motor, que está embutido no próprio número do comando.

O número do motor servo que deve ser lido é igual à subtração do número do comando por cinquenta. Por exemplo, o comando de número 65 equivale ao comando `Ler` para o motor número 15. Deste modo, nenhum dado secundário precisa ser enviado para a Ponte de Comando.

Entretanto, a Ponte de Comando deve responder, enviando a posição angular do motor requi-

sitado. Este é o único comando que recebe um dado de resposta da Ponte de Comando, o qual é enviado na seguinte forma: um prefixo "a"seguido de um número inteiro de 0 a 100 que corresponde à posição angular, onde 0 equivale à posição angular máxima no sentido anti-horário (mínimo do potenciômetro) e 100 à posição angular máxima no sentido horário (máximo do potenciômetro).

A Ponte de Comando realiza a leitura da tensão no terminal central do potenciômetro e converte o dado para um intervalo de 0 a 100. Em seguida, envia este dado pela porta serial.

4.1.2.7 `AutoCalibragem`

Por último, o comando `AutoCalibragem` é responsável por obter os dados necessários para a conversão realizada pelo comando `Ler`. Este comando deve ser acionado toda vez que a Ponte de Comando for ligada, caso contrário o comando `Ler` não funcionará corretamente.

O comando `AutoCalibragem` armazena as leituras do potenciômetro de cada motor servo nas posições máxima e mínima. Assim o comando `Ler` pode converter a leitura de tensão no pino central do potenciômetro para os valores utilizados nos comandos do protótipo.

Primeiro, todos motores são ativados. Em seguida, movem-se todos até a posição máxima(100). Armazena-se a leitura da tensão no terminal do potenciômetro do motor servo em uma variável para cada motor. Então, movem-se todos os motores até a posição mínima. A leitura do potenciômetro é, então, armazenada em outra variável para cada motor.

Desta maneira é obtido o intervalo da leitura de tensão do potenciômetro de cada motor. Com este intervalo é possível converter qualquer leitura de tensão para o intervalo de 0 a 100 usado no protótipo.

4.2 PEDALEIRA DE CONTROLE

A Pedaleira de Controle oferece uma interface em hardware para que o usuário possa controlar os módulos motorizados conectados à Ponte de Comando. A Figura 4.3 mostra uma foto da tela de inicialização da interface.

O *Firmware* da Pedaleira de Controle (Anexo B) funciona baseado nos menus da interface com a tela LCD. Cada menu é um *loop* de código infinito até que o botão do *Encoder* seja pressionado. Então, dependendo da posição do *Encoder*, o código avança até o próximo menu.

Um ponto importante é que além de transmitir os próprios comandos, a Pedaleira de Controle deve, também, retransmitir os comandos do Software e as respectivas respostas enviadas pela Ponte de Comando.

Assim, no *loop* de cada menu está inserida a função `Serial_repeat`. Esta função retransmite a mensagem que recebe do software para a Ponte de Comando e vice-versa.

Figura 4.3: Foto da tela de incialização - **Fonte:** Autor

A Figura 4.4 mostra o diagrama do funcionamento da função `Serial_repeat`.

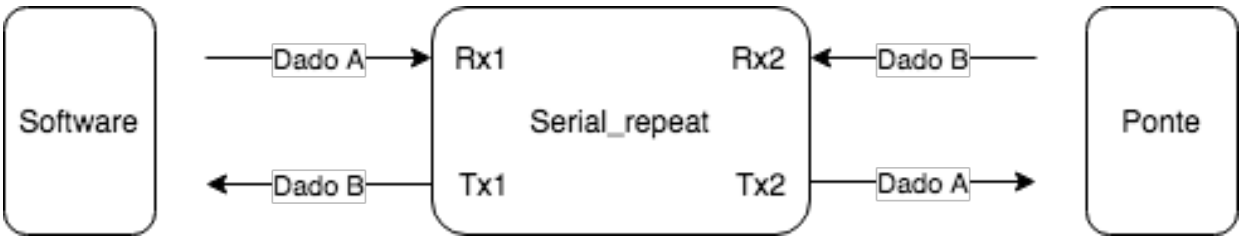

Figura 4.4: Diagrama de funcionamento da função `Serial_repeat` - **Fonte:** Autor

4.2.1 Menu Principal

A Figura 4.5 mostra um esquema que ilustra o funcionamento do *Firmware* da Pedaleira de Controle no *loop* do Menu Principal. A variável En_bt indica se o botão do *Encoder* foi apertado ou não, enquanto a variável En_pos indica a posição do *Encoder*.

O Menu Principal oferece três opções de escolha e cada uma delas será abordada a seguir. Na tela LCD, entretanto, só duas opções, uma em cada linha, aparecem por vez. Quais opções aparecem dependende da posição do *Encoder*.

A opção do menu que equivale à posição do *Encoder* é indicada por uma seta. Isto é válido para todos os outros menus também.

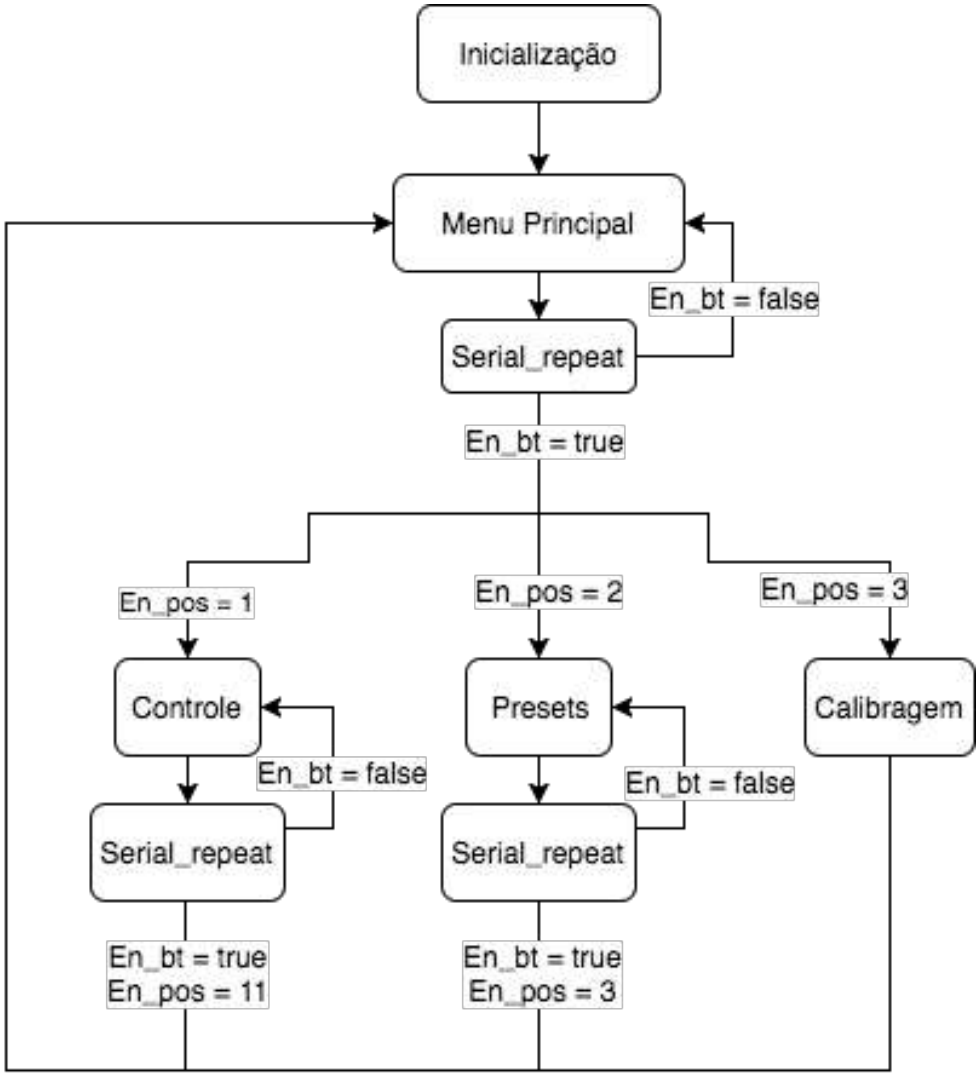

Figura 4.5: Diagrama do funcionamento do *firmware* da Pedaleira de Controle - **Fonte:** Autor

A Figura 4.6 mostra uma foto da tela LCD no menu principal com o *Encoder* na posição 2, ou seja, na opção "Presets". Se o botão do *Encoder* for pressionado nesta opção, a tela irá para o menu Presets.

Figura 4.6: Foto da tela no menu principal - **Fonte:** Autor

4.2.2 Controle

O menu Controle se destina a controlar os módulos motorizados individualmente em tempo real utilizando a interface.

No menu, existem 10 opções que correspondem ao número do motor escrito por extenso (um, dois, três, ..., dez) e a opção "Retorno".

A opção "Retorno"(Figura 4.7) volta para o Menu Principal, enquanto as outras opções levam para a tela de controle em tempo real do respectivo motor.

Figura 4.7: Foto da tela com a opção "Retorno- **Fonte:** Autor

Ao selecionar um motor, a Pedaleira de Controle envia o comando de leitura para identificar a posição angular. Então, utiliza este valor como posição inicial do *Encoder* enquanto é mostrado na tela LCD.

Para voltar à tela de seleção do motor, basta pressionar o botão do *Encoder*.

A Figura mostra fotos da tela nas duas situações. Em "A"o usuário escolhe o motor que vai ser controlado e em "B"o usuário pode controlá-lo em tempo real enquanto acompanha pela tela.

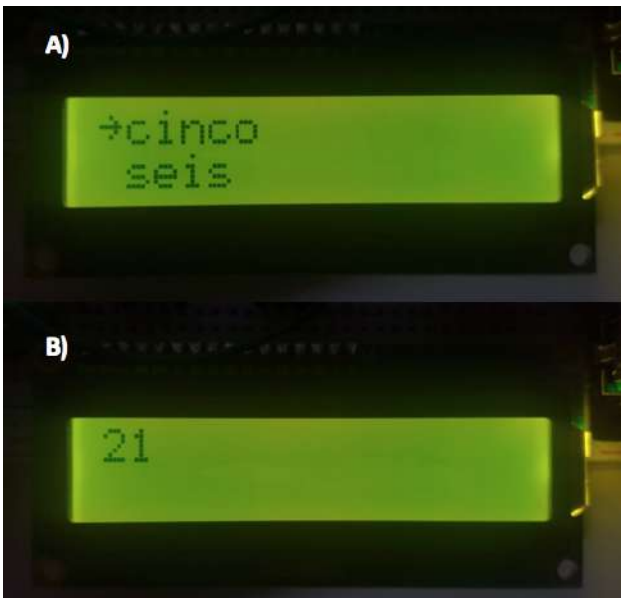

Figura 4.8: Foto da tela no menu de controle. Em "A"são listados os motores e em "B"a posição do motor escolhido - **Fonte:** Autor

4.2.3 Presets

Este é o menu utilizado para administrar as configurações armazenadas e para tocar ao vivo. A Figura 4.9 mostra um diagrama de como este menu funciona.

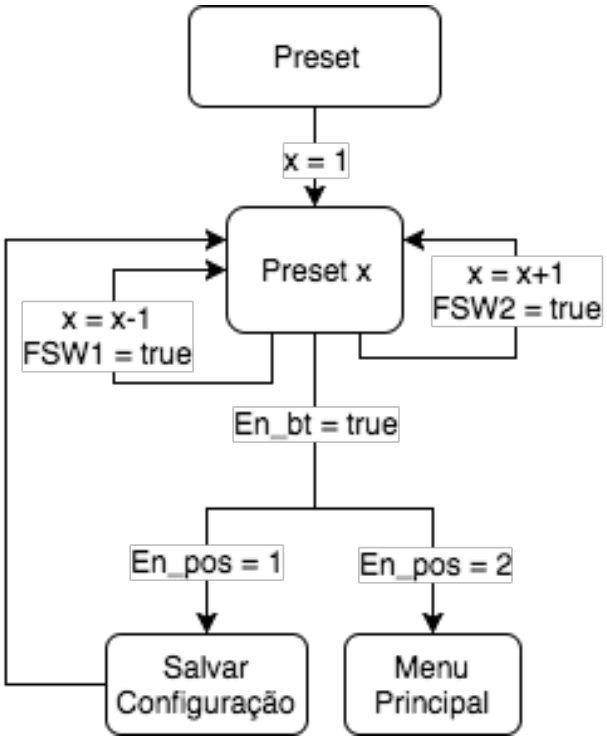

Figura 4.9: Diagrama de funcionamento do menu Presets - **Fonte:** Autor

As variáveis `FSW1` e `FSW2` representam os dois botões *Footswitch* responsáveis por alternar entre os *presets*. O botão `FSW1` decresce o índice do *preset* enquanto o botão `FSW2` acresce o

mesmo índice. No *preset* 1, o botão `FSW1` não tem função. Já no *preset* 5, o botão `FSW2` não tem função, pois o protótipo irá armazenar no máximo cinco *presets*.

Toda vez que o índice do *preset* é alterado, a Pedaleira de Controle envia os comandos para mover todos os motores de acordo com as posições armazenadas no novo índice.

O *Encoder* é usado para selecionar uma de duas opções: salvar as posições angulares dos módulos motorizados no *preset* atual ou retornar ao Menu Principal.

Para configurar um novo *preset*, o usuário possui duas opções: regular as configurações do amplificador manualmente e apertar salvar; ou utilizar o menu de controle para regular os motores e apertar salvar.

A Figura 4.10 mostra a tela LCD no menu Presets. Nota-se que o *Encoder* está na posição 1, pois a seta indica a opção "salvar".

Figura 4.10: Foto da tela no menu Presets - **Fonte:** Autor

4.2.4 Calibragem

A opção Calibragem não é, de fato, um menu ainda. Por enquanto, no protótipo, esta opção somente envia o comando de AutoCalibragem para a Ponte de Comando.

Futuramente, no produto, pretende-se adicionar outras funções de calibragem, justificando a existência desta opção no Menu Principal.

5 TESTES

Após a fabricação do protótipo, foram realizados alguns testes considerando os dois principais segmentos de cliente. Estes testes têm como objetivo averiguar a validade do projeto como um produto no mercado e coletar informações para possíveis melhorias no *design*.

Foram feitos testes em um estúdio de gravação e em um estúdio de ensaio. No estúdio de gravação o engenheiro utilizava o software para controlar o protótipo, enquanto nos ensaios o músico utilizou a Pedaleira de Controle para alternar entre configurações de timbres armazenadas.

A Figura 5.1 mostra o protótipo sendo utilizado na gravação de uma banda de renome nacional em um estúdio profissional.

Figura 5.1: Protótipo sendo utilizado em uma gravação profissional - **Fonte:** Autor

O produto demonstrou confiabilidade e não apresentou nenhum problema técnico durante os testes nesta gravação. O engenheiro de gravação responsável, não teve dificuldade em utilizá-lo e gostou bastante da solução.

No teste em ensaio também não houve nenhum problema técnico e o músico ficou bastante satisfeito de poder acessar os diferentes timbres do seu amplificador. A interface da Pedaleira de Controle foi suficientemente satisfatória e intuitiva para o músico.

Entretanto, o músico ficou bastante insatisfeito com o suporte dos módulos. O suporte é muito grande e necessita de muito tempo para ser instalado. Músicos, em geral devido às dificuldades logísticas das apresentações ao vivo, necessitam de equipamentos práticos e rápidos para serem montados e este suporte representa exatamente o oposto.

No caso do estúdio de gravação, a praticidade do suporte não é tão relevante. Estúdios geral-

mente já possuem equipamentos grande e mais um equipamento não faz tanta diferença. Porém, a facilidade e rapidez de instalação é também um fator importante para este segmento de cliente.

Sendo assim, a principal conclusão a ser retirada dos testes é de que o produto fornece uma proposta de valor que resolve um dor dos clientes, tanto do músico como do engenheiro de gravação, entretanto, o protótipo ainda não é prático o suficiente para ser aceito no mercado como um produto.

Os desenvolvimentos futuros devem, do ponto de vista do produto, abordar a questão da praticidade e da facilidade de instalação do suporte.

6 CONCLUSÃO

Os amplificadores de guitarra analógicos, apesar de tecnologicamente ultrapassados, ainda são os favoritos dos músicos. Isto ocorre devido à sua característica sonora, que se tornou a referência em timbres.

Os equipamentos digitais tentam emular o som deste amplificadores, mas poucos conseguem agradar aos músicos. Ainda sim, estão cada vez mais difundidos, pois são mais práticos e oferecem mais recursos. Existe, então, uma grande oportunidade em aliar o som analógico aos recursos de equipamentos digitais.

A criação do *RoadieBot* visa, por meio da automação das configurações de timbre, expandir a experiência do músico e facilitar o trabalho do engenheiro de gravação que utiliza amplificadores analógicos de instrumentos musicais.

O músico poderá armazenar diferentes configurações de timbre do seu amplificador, explorando o equipamento ao máximo enquanto o engenheiro de som poderá controlar o amplificador remotamente, facilitando a procura do timbre ideal quando o equipamento se encontra em outra sala.

O objetivo do trabalho é o desenvolvimento de um protótipo funcional que consiga representar minimamente a proposta de valor do produto. Assim, poderá coletar informações quanto à validação do produto no mercado musical.

Tratou-se, neste trabalho do desenvolvimento do protótipo desde o estágio inicial até o estado com os recursos apresentados, abordando todos os seus aspectos: incluindo escolha de componentes, materiais, fabricação e programação.

Pode-se dizer que o objetivo do trabalho foi alcançado com sucesso. O estado de desenvolvimento resultante já permite a realização de testes em situações reais, tanto para o músico quanto para o engenheiro.

Foram realizados testes bem sucedidos com o protótipo, que não apresentou nenhum problema até então. Já foi possível coletar informações quanto à validação no mercado e possíveis melhorias de *design* de produto, principalmente no quesito praticidade.

O suporte utilizado é muito grande e trabalhoso para ser instalado. Os clientes, principalmente os músicos, buscam sempre os equipamentos mais práticos. Com isso em mente, um novo suporte mecânico é a principal melhoria do ponto de vista de produto.

A principal deficiência técnica do protótipo está no circuito de desacoplamento da Ponte de Comando. As tentativas de desacoplar eletronicamente os motores do microcontrolador falharam e uma solução totalmente não-ideal foi utilizada.

Apesar de esta solução tornar possível o uso do equipamento em teste, é totalmente inade-

quada para o produto final.

Além disso, a escolha de outro material para a fabricação das engrenagens, os mancais e o eixo poderia aumentar a precisão do sistema e , também, a durabilidade.

O plástico impresso em três dimensões foi essencial na prototipagem destas peças, entretanto, é inviável para um produto. Impressões em três dimensões possuem baixa repetibilidade, além de possuir elevado custo na produção em larga escala

As peças de Lego foram uma ótima solução para a fase de prototipagem, mas devem ser utilizados peças mais resistentes, precisas e baratas em larga escala.

Na parte do *firmware*, existe muito espaço para desenvolvimento futuro, como: capacidade maior de armazenamento de *presets*; melhoria na interface; e inserir dados de confirmação na comunicação serial.

A melhoria na interface deve se basear na experiência do usuário, sendo atualizada constantemente até a versão final.

Por fim, após sanadas as dificuldades listadas acima, deve-se realizar o projeto de placas de circuito impresso para substituir as *protoboards* utilizadas no protótipo. *Protoboards* podem apresentar mau contato e as conexões podem ser tiradas do lugar acidentalmente sem grande dificuldade.

Em resumo, por se tratar do primeiro protótipo deste produto, cada parte do projeto pode ser melhor desenvolvida. Este protótipo apresenta as soluções mais práticas e viáveis para o tempo de desenvolvimento e recursos disponíveis.

Este trabalho descreveu, então, os passos iniciais no caminho para se chegar a uma versão do produto *RoadieBot* capaz de ingressar no mercado de equipamentos voltados para a música.

REFERÊNCIAS BIBLIOGRÁFICAS

1 MALECZEK, S. Testing the sound quality of acoustic vacuum tube amplifier. *Archives of Acoustics*, v. 33, Issue 4, p. 135–140, 2008.

2 PAKARINEN, J.; YEH, D. A review of digital techniques for modeling vacuum-tube guitar amplifiers. *Computer Music Journal*, v. 33, n. 2, p. 85–100, 2009.

3 LOFFT, A. *10 Things about Audio Amplifiers You've Always Wantede to Know*. 2008. Disponível em: <http://www.audioholics.com/audio-amplifier/10-things-about-audio-amplifiers>.

4 VALLE, S. d. *Manual Prático de Acústica*. [S.l.]: Editora Música e Tecnologia, 2009.

5 HUBER, D.; RUNSTEIN, R. *Modern Recording Techniques*. [S.l.]: Focal Press, 2010.

6 ASTLEY-BROWN, M. *The best new guitar and bass amps of NAMM 2016*. 2016. Disponível em: <http://www.musicradar.com/news/guitars/the-best-new-guitar-and-bass-amps-of-namm-2016-633959>.

7 GRIDLING, G.; WEISS, B. *Introduction to Microcontrollers*. [S.l.]: Vienna University of Technology, 2007.

8 ZELENOVSKY, R.; MENDONCA, A. *Microcontroladores: Programação e Projeto com a Família 8051*. [S.l.]: MZ Editora Ltda., 2005.

9 VIS, P. J. *Stepper Motor Basics*. Disponível em: <http://www.petervis.com/electronics/stepper-motor/stepper-motor-basics.html>.

10 HUGHES, A. *Electric Motors and Drives*. [S.l.]: Newnes, 2006.

11 KIATRONICS. *28BYJ-48 – 5V Stepper Motor*. [S.l.].

12 DIGEST, C. *Servo Motor: Basics, Theory and Working Principle*. 2016. Disponível em: <http://circuitdigest.com/article/servo-motor-basics>.

13 TOWER PRO. *SG90 9g Micro Servo*. [S.l.].

14 ARDUINO. *What is Arduino?* 2017. Disponível em: <https://www.arduino.cc/en/Guide/Introduction>.

APÊNDICES

A. CÓDIGO FONTE - PONTE DE COMANDO

```
//Biblioteca usada para controlar os motores servo
#include <Servo.h>

//Declara os servos
Servo s[11];

////////////
//VARIÁVEIS//
////////////

int angle, servo, i, cmd = 0;
int servo_pin[11] = {0, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31}; // Pinos
int total = 10; //número de servos
float Max[11], Min[11], a, c;
unsigned long ultimo_cmd_tempo = 0, cmd_tempo;

////////
//SETUP//
////////


void setup() {
  Serial1.begin(57600);

  // Impede o parseint() de deixar o loop mais devagar esperando por um ti
  Serial1.setTimeout(50);
}


////////////////
//AUTOCALIBRAGEM//
////////////////

//Função para calibrar a leitura da posição dos motores

void AutoCalibragem() {
```

```
  //Ativar todos os servos
  Ativar_todos();

  //Mover todos para posição máxima
  for (i = 1; i <= total; i++) {
    s[i].write(0);
  }
  delay(1500);

  //MMover todos para posição mínima
  for (i = 1; i <= total; i++) {
    s[i].write(180);
  }
  delay(1500);

  //Mover todos para posição máxima
  for (i = 1; i <= total; i++) {
    s[i].write(0);
  }
  delay(1500);

  //Ler Max
  for (i = 1; i <= total; i++) {
    Max[i] = float (analogRead(i));
  }

  //Mover todos para posição mínima
  for (i = 1; i <= total; i++) {
    s[i].write(180);
  }
  delay(1500);

  //Ler min
  for (i = 1; i <= total; i++) {
    Min[i] = float (analogRead(i));
  }

  //Desativar todos os servos
  Desativar_todos();
}
```

```
///////
//LER//
///////

//Função para obter a posição angular dos motores

void ler(int ser) {
  a = analogRead(ser);
  c = map(a, Max[ser], Min[ser], 180, 0);
}

/////////
//MOVER//
/////////

//Função para mover os motores

void mover(int ser, int pos) {
  //o set de engrenagens inverte o sentido de rotação do motor, então o co
  int pos2 = map(pos, 0, 180, 180, 0);
  s[ser].write(pos2);

}

///////////////
//ATIVAR_TODOS//
///////////////

//Função para ativara todos os motores

void Ativar_todos() {
  for (i = 1; i <= total; i++) {
    ler(i);
    mover(i, c);
    s[i].attach(servo_pin[i], 710, 2240);
  }
}

/////////
```

```
//ATIVAR//
/////////

//Função para ativar um motor específico

void Ativar(int i) {
  ler(i);
  mover(i, c);
  s[i].attach(servo_pin[i], 710, 2240);
}

//////////////////
//DESATIVAR_TODOS//
//////////////////

//Função para desativar todos os motores

void Desativar_todos() {
  for (i = 1; i <= total; i++) {
    s[i].detach();
  }
}

/////////////
//DESATIVAR//
/////////////

//Função para desativar um motor específico

void Desativar(int i) {
  s[i].detach();
}

/////////
//OCIOSO//
/////////

//Função para desativar os motores em caso de ociosidade

void Ocioso(){
  cmd_tempo = millis();
```

```
  if (cmd_tempo - ultimo_cmd_tempo > 2000){
    Desativar_todos();
    ultimo_cmd_tempo = cmd_tempo;
  }
}

////***************////
////LOOP PRINCIPAL////
////***************////

void loop() {
  Ocioso();
  if (Serial1.available()) {

    if (Serial1.read() == 'c') {
      ultimo_cmd_tempo = millis(); // Zera o timer de ociosidade
      cmd = Serial1.parseInt(); //Lê o número até um "." e declara como um
    }

    //Neste caso o comando é para mover o motor de número "cmd"
    if (cmd < 40 && cmd != 0) {
      if (Serial1.read() == 's') {
        angle = Serial1.parseInt();
        Ativar(cmd); //É necessário ativar o motor antes de movê-lo
        mover(cmd, angle);
      }
    }

    // Neste caso o comando é para ler o motor de número cmd-50. Ex: cmd=5
    else if (cmd > 50 && cmd < 90) {
      cmd = cmd - 50;
      ler(cmd);
      Serial1.println(c);
    }

    //Ativa um servo específico
    else if (cmd == 93) {
      if (Serial1.read() == 's') {
        servo = Serial1.parseInt();
        Ativar(servo);
      }
```

```
    }

    //Desativa um servo específico
    else if (cmd == 94) {
      if (Serial1.read() == 's') {
        servo = Serial1.parseInt();
        Desativar(servo);
      }
    }

    // Aciona todos os servos
    else if (cmd == 95) {
      Ativar_todos();
    }

    // Aciona todos os servos
    else if (cmd == 96) {
      Desativar_todos();
    }

    // 99 é o comando para a calibragem automática
    else if (cmd == 99) {
      AutoCalibragem();
    }
    cmd = 0; //zera a variável "cmd"

  }
}
```

B. CÓDIGO FONTE - PEDALEIRA DE CONTROLE

```
//Biblioteca que gera uma segunda porta serial que fará a comunicação com
#include <SoftwareSerial.h>

//Biblioteca da tela LCD
#include <LiquidCrystal.h>

//Biblioteca para o Encoder
#include <RotaryEncoder.h>

//Biblioteca para utilizar a memória não-volátil to microcontrolador
#include <EEPROM.h>

//Declaração dos pinos utilizados nas bibliotecas
SoftwareSerial mySerial(9, 10); //Serial
LiquidCrystal lcd(12, 11, 8, 4, 3, 2);  //LCD
RotaryEncoder encoder(14, 15);  //Encoder

//Pinos dos botões
#define En_bt_pin 5
#define FSW1 6
#define FSW2 7

////////////
//VARIÁVEIS//
////////////

//Variáveis usadas ao longo do código
int servo, i, Endereco, Posicao;
int total = 10; //número de servos
int En_pos = 0, P = 1;
float  a, b, c;
char seta = (char)126, blank = (char)16;
boolean En_bt = false;

//Opções do menu principal e do menu de controle
String menu_controle_opcoes[10] = {"um             ", "dois           ", "
String menu_principal_opcoes[3] = {"Controle       ", "Presets        ", "
```

```
//Variáveis para a rotina de debouncing dos botões
unsigned long lastDebounceTime = 0;
unsigned long debounceDelay = 50;



///////////////
//INTERRUPÇÕES//
///////////////

//Rotina de interrupção do Encoder

ISR(PCINT1_vect) {
  //Função da biblioteca do Encoder
  encoder.tick();
}


//Rotina da interrupção dos botões

ISR (PCINT2_vect) {
  //Rotina de debouncing e identificação do botão pressionado
  static unsigned long last_interrupt_time = 0;
  unsigned long interrupt_time = millis();
  if (interrupt_time - last_interrupt_time > 250) {
    //Checa se o botão do Encoder foi o botão pressionado
    if (digitalRead(En_bt_pin) == LOW) {
      En_bt = true;
    }
    //Checa se FSW1 foi o botão pressionado
    if (digitalRead(FSW1) == HIGH) {
      if (P >= 1 & P < 5 ) {
        P = P + 1;
        preset_acessar(P);
      }
    }
    //Checa se FSW1 foi o botão pressionado
    if (digitalRead(FSW2) == HIGH) {
      if (P > 1 & P <= 5) {
        P = P - 1;
```

```
        preset_acessar(P);
      }
    }
    last_interrupt_time = interrupt_time;
  }

}


/////////
//SETUP//
/////////

void setup() {
  // Serial que conecta com o computador pelo cabo USB
  Serial.begin(9600);

  // Serial que conecta a pedaleira de controle com a ponte de comando pel
  mySerial.begin(57600);

  //Define pino do botão do encoder como entrada
  pinMode(En_bt_pin, INPUT);

  //Ativa interrupções para os registradores PCMSK1 e PCMSK2
  PCICR |= (1 << PCIE2) | (1 << PCIE1);
  // Ativa interrupções dos três botões da interface
  PCMSK2 |= (1 << PCINT21) | (1 << PCINT22) | (1 << PCINT23);
  // Ativa interrupções para os dois terminais do Encoder
  PCMSK1 |= (1 << PCINT8) | (1 << PCINT9);

  //Inicia a biblioteca da tela LCD. Linhas e colunas
  lcd.begin(16, 2);

  //Escreve tela inicial
  lcd.setCursor(2, 0);
  lcd.print("ROADIEBOT :)");
  lcd.setCursor(4, 1);
  lcd.write("Prot. v1");
  delay(1500);
  lcd.clear();
```

```
}

//////////
//ENCODER//
//////////

//Função que retorna a posição do Encoder limitada por posição mínima e má

void Encoder(int ROTARYMIN, int ROTARYMAX) {
  encoder.tick();
  int newPos = encoder.getPosition();
  if (newPos < ROTARYMIN) {
    encoder.setPosition(ROTARYMIN);
    newPos = ROTARYMIN;
  } else if (newPos > ROTARYMAX) {
    encoder.setPosition(ROTARYMAX);
    newPos = ROTARYMAX;
  }
  if (En_pos != newPos) {
    En_pos = newPos;
  }
}

/////////
//ATIVAR//
/////////

//Manda o comando para ativar um servo específico

void Ativar(int servo) {
  mySerial.print("c93s");
  mySerial.print(servo);
}

////////////
//DESATIVAR//
////////////

//Manda o comando para desativar um servo específico

void Desativar(int servo) {
```

```
  mySerial.print("c94s");
  mySerial.print(servo);
}

//////////////////
//AUTOCALIBRAGEM//
//////////////////

//Manda o comando para realizar a calibragem automática

void AutoCalibragem() {
  mySerial.print("c99");
}

/////////
//MOVER//
/////////

//Manda o comando para mover um servo

void Mover (int servo, int angle) {
  mySerial.print("c");
  mySerial.print(servo);
  mySerial.print("s");
  mySerial.print(angle);
}

///////
//LER//
///////

//Manda o comando para obter a posição de um servo

void Ler(int servo) {
  servo = servo + 50;
  mySerial.print("c");
  mySerial.print(servo);
  delay(200);
  c = mySerial.parseInt();
}
```

```
////////////////
//CONTROLE_SERVO//
////////////////

//Função do menu de controle para controlar um servo em tempo real

void Controle_Servo(int ser) {
  Ativar(ser);
  Encoder(0, 100);
  Ler(ser);
  c = c / 1.8; //c é o resultado de Ler
  En_pos = c;
  encoder.setPosition(c);
  while (true) {
    Encoder(0, 100);
    lcd.clear();
    lcd.print(En_pos);
    En_pos = En_pos * 1.8;
    Mover(ser, En_pos);
    delay(100);
    if (En_bt == true) {
      En_bt = false;
      Desativar(ser);
      menu_controle(ser);
    }

  }
}

////////////////
//SERIAL_REPEAT//
////////////////

//Função para transmitir o comando do computador para a Ponte de Comando

void Serial_repeat() {
  while (mySerial.available()) {
    Serial.write(mySerial.read());
  }
  while (Serial.available()) {
    mySerial.write(Serial.read());
```

```
  }
}

//////////////////
//PRESET_ACESSAR//
//////////////////

//Função para acessar um preset armazenado

void preset_acessar(int p) {
  //Enderecos da memória. Banco 1: 1 a 50; banco 2: 51 a 100; banco 3: 101
  Endereco = (p - 1) * 50 + 1;
  for (int servo = 1; servo < 11; servo++) {
    Posicao = EEPROM.read(Endereco);
    Mover(servo, Posicao);
    Endereco++;
  }
}

//////////////////
//PRESET_SALVAR//
//////////////////

//Função para salvar um preset

void preset_salvar(int p) {
  lcd.clear();
  lcd.print("Salvando...");
  //Enderecos da memória. Banco 1: 0 a 49; banco 2: 50 a 99; banco 3: 100
  Endereco = (p - 1) * 50 + 1;
  for (int servo = 1; servo < 11; servo++) {
    Ler(servo);
    Posicao = c;
    EEPROM.put(Endereco, Posicao);
    Endereco++;
  }
  delay(500);
}

//************************//
////////MENU PRINCIPAL///////
```

```
//************************//

//Função do Menu Principal

void menu_principal(int encod) {
  encoder.setPosition(encod); // Posicao inicial do encoder
  while (true) {
    Serial_repeat();
    Encoder(1, 3);
    if (En_pos < 3) {
      lcd.home();
      lcd.print(seta);
      lcd.print(menu_principal_opcoes[En_pos - 1]);
      lcd.setCursor(0, 2);
      lcd.print(blank);
      lcd.print(menu_principal_opcoes[(En_pos)]);
    }
    else if (En_pos == 3) {
      lcd.home();
      lcd.print(blank);
      lcd.print(menu_principal_opcoes[En_pos - 2]);
      lcd.setCursor(0, 2);
      lcd.print(seta);
      lcd.print(menu_principal_opcoes[(En_pos - 1)]);
    }

    if (En_bt == true) {
      En_bt = false;
      switch (En_pos) {
        case 1 : {
            menu_controle(1);
          }
        case 2 : {
            menu_presets();

          }
        case 3 : {
            AutoCalibragem();
          }
      }
    }
```

```
  }
}

//////////////////
//MENU_CONTROLE//
//////////////////

//Função do Menu Controle

void menu_controle(int encod) {
  encoder.setPosition(encod); // Posicao inicial do encoder
  while (true) {
    Serial_repeat();
    Encoder(1, (total + 1)); // +1 posição por causa do "Retornar" que ocu
    if (En_pos < total) {
      lcd.home();
      lcd.print(seta);
      lcd.print(menu_controle_opcoes[En_pos - 1]);
      lcd.setCursor(0, 2);
      lcd.print(blank);
      lcd.print(menu_controle_opcoes[(En_pos)]);
    }
    else if (En_pos == total) {
      lcd.home();
      lcd.print(seta);
      lcd.print(menu_controle_opcoes[En_pos - 1]);
      lcd.setCursor(0, 2);
      lcd.print(blank);
      lcd.print("Retorno         ");
    }
    else {
      lcd.home();
      lcd.print(blank);
      lcd.print(menu_controle_opcoes[En_pos - 2]);
      lcd.setCursor(0, 2);
      lcd.print(seta);
      lcd.print("Retorno         ");
    }
    if (En_bt == true) {
      En_bt = false;
      if (En_pos == total + 1)
```

```
        menu_principal(1);

      Controle_Servo(En_pos);
    }
  }
}

////////////////
//MENU_PRESETS//
////////////////

//Função do Menu Presets

void menu_presets() {
  encoder.setPosition(P);
  lcd.clear();
  Serial_repeat();
  while (true) {
    Encoder(1, 2);
    if (En_pos == 1) {
      lcd.home();
      lcd.print("    PRESET ");
      lcd.print(P);
      lcd.setCursor(0, 2);
      lcd.print(seta);
      lcd.print("salvar  ");
      lcd.print(blank);
      lcd.print("voltar");
    }
    else {
      lcd.home();
      lcd.print("    PRESET ");
      lcd.print(P);
      lcd.setCursor(0, 2);
      lcd.print(blank);
      lcd.print("salvar  ");
      lcd.print(seta);
      lcd.print("voltar");
    }
    if (En_bt == true) {
      En_bt = false;
```

```
      switch (En_pos) {
        case 1 : {
            preset_salvar(P);
          }
        case 2 : {
            menu_principal(2);
          }
      }
    }
  }
}


////////
//Loop//
////////

void loop() {
  menu_principal(1);
}
```