INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA - CAMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE METAL MECÂNICA - DAMM
CURSO DE GRADUAÇÃO EM ENGENHARIA MECATRÔNICA

MAURICIO NEVES JUNIOR

DESENVOLVIMENTO DE CONTROLADOR CNC DIDÁTICO BASEADO EM *LINUXCNC* E SISTEMA EMBARCADO

FLORIANÓPOLIS, 2024

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA - CAMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE METAL MECÂNICA - DAMM
CURSO DE GRADUAÇÃO EM ENGENHARIA MECATRÔNICA

MAURICIO NEVES JUNIOR

DESENVOLVIMENTO DE CONTROLADOR CNC DIDÁTICO BASEADO EM *LINUXCNC* E SISTEMA EMBARCADO

Trabalho de Conclusão de Curso submetido ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos para obtenção do título de Engenheiro Mecatrônico em 2024.

Orientador: Prof. Me. Vitor Farias de Borba.

FLORIANÓPOLIS, 2024

Ficha de identificação da obra elaborada pelo autor.

Neves Junior, Mauricio
Desenvolvimento de Controlador CNC Didático Usando **LinuxCNC e Sistema Embarcado / Mauricio Neves Junior;** orientação **de Vitor Farias de Borba. - Florianópolis,** SC, 2024.
76p.

Trabalho de Conclusão de Curso (TCC) - Instituto Federal **de Santa Catarina, Câmpus Florianópolis.** Bacharelado **em Engenharia Mecatrônica.** Departamento **Acadêmico de Metal** Mecânica.
Inclui Referências.

1. CNC. 2. Didático. 3. Movimentação. 4. Sistema. **5. Embarcados. I. Farias de Borba, Vitor. II.** Instituto **Federal de Santa Catarina. III. Desenvolvimento de** Controlador **CNC Didático Usando LinuxCNC e** Sistema Embarcado.

DESENVOLVIMENTO DE CONTROLADOR CNC DIDÁTICO BASEADO EM *LINUXCNC* E SISTEMA EMBARCADO

MAURICIO NEVES JUNIOR

Este trabalho foi julgado adequado para obtenção do título de Engenheiro Mecatrônico e aprovado na sua forma final pela banca examinadora do Curso de Engenharia Mecatrônica do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 6 de setembro, 2024.

Banca Examinadora:

Documento assinado digitalmente
gov.br
VITOR FARIAS DE BORBA
Data: 27/09/2024 11:10:41-0300
Verifique em https://validar.iti.gov.br

Vitor Farias de Borba, Me.

Documento assinado digitalmente
gov.br
CASSIANO BONIN
Data: 27/09/2024 17:56:21-0300
Verifique em https://validar.iti.gov.br

Cassiano Bonin, Me.
IFSC

Documento assinado digitalmente
GOV.BR
GREGORY CHAGAS DA COSTA GOMES
Data: 27/09/2024 11:22:33-0300
CPF: ***.008.969-**
Verifique as assinaturas em https://v.ufsc.br

Gregory Chagas da Costa Gomes, Me.
IFSC

AGRADECIMENTOS

Esta monografia é o *gran finale* de uma batalha árdua e superação de desafios que, antes pareciam impossíveis, e agora são belas histórias e estarão para sempre na memória. Amizades formadas, conhecimentos adquiridos e momentos vividos marcaram os anos de estudos ao longo da graduação e foram a base preparatória para os futuros desafios que estão por vir.

E o primeiro grande pilar de sustentação vem da fé. Fé em Deus, fé em Nossa Senhora Aparecida, e a Eles, dedico meu agradecimento pois em orações foram ouvidos desabafos, tristezas e também agradecimentos por alegrias e graças alcançadas.

O segundo grande pilar, responsável por estarmos desde o primeiro dia de vida na Terra é a família. Sem o seu apoio, amor incondicional e grande motivação não seria possível quiçá iniciar esta jornada quanto mais terminá-la. Aos meus pais Maurício e Rogéria sou eternamente grato por me proporcionarem valores, conselhos, acolhimento e mostrar o caminho a ser trilhado. Aos meus irmãos Adriano, Fábio, Thayna, Alexander e Rafael, agradeço o companheirismo, orientações e por serem grandes exemplos de pessoas íntegras e honestas.

Amigos, palavra adjetiva que significa “Aquele que nutre admiração”, e para um estudante os amigos são os grandes alicerces da vida, pois são eles que estão ao nosso lado nos bons e maus momentos, nos estendem a mão nos momentos de dificuldade e comemoram nossas vitórias sem pedir nada em troca além de reciprocidade. Aos amigos formados ao longo da trajetória, deixo meu mais sincero agradecimento, em especial Beatriz B. Cardoso, Christian S. Feliciano, Gabriel V. da Motta, Jordana B. de Aguiar e Maria Eduarda C. da Silveira.

E, por fim, agradeço ao Instituto Federal de Santa Catarina pela qualidade de ensino e pelas oportunidades proporcionadas. Em especial agradeço aos professores Vitor Farias de Borba, Erwin Werner Teichmann, Eduardo Antônio Linck, Francisco Rafael Moreira da Mota, Adriano Régis e Roberto Alexandre Dias, pois a sua didática excepcional e trocas de experiências dentro e fora de sala de aula formaram e orientaram grande parte do futuro profissional que aqui vos escreve.

“A Persistência é o caminho do êxito”

-Charlie Chaplin

RESUMO

No ambiente industrial, máquinas operatrizes operadas por comando numérico computadorizado (CNC) por vezes são a diferença entre a economia de tempo produtivo e, consequentemente, dinheiro. Conforme os processos produtivos avançam, são exigidas pela indústria máquinas mais robustas e soluções mais integradas, porém, para o ambiente educacional, onde uma mesma máquina pode operar por décadas já que não são tão exigidas quanto em uma fábrica, o objetivo é ensinar aos alunos através de métodos teóricos e práticos, o funcionamento de tais sistemas, bem como a sua integração com outros sistemas independentes e até mesmo o desenvolvimento de um sistema próprio. Visando tal panorama, o projeto tem por objetivo desenvolver uma plataforma que permita o controle de movimentação de equipamentos robustos e/ou didáticos de forma compacta, e para atingir tal objetivo é proposta a utilização de sistemas embarcados como principal elemento deste controlador, substituindo o tradicional computador *desktop*. Ao longo deste trabalho são realizados breves apontamentos de tecnologias disponíveis para realizar esta tarefa e, após a avaliação de cada possibilidade, a solução proposta foi a utilização da placa de desenvolvimento *Raspberry PI* em conjunto com a placa controladora *BigTreeTech Octopus v1.1*. A solução escolhida foi desenvolvida e analisada com testes em bancada analisando a geração de pulsos, tanto a sua quantidade quanto a sua frequência, e em uma guia de movimentação linear para analisar o comportamento de tais pulsos, apresentando resultados satisfatórios para o controle de máquinas que possuam até 8 eixos, o que representa uma grande parte se não todos os equipamentos desejados.

Palavras Chave: CNC, didático, movimentação, sistema embarcado

ABSTRACT

In the industrial environment, machine tools operated by computer numerical control (CNC) are often the difference between saving productive time and, consequently, money. As production processes advance, the industry demands more robust machines and more integrated solutions. However, in the educational environment, where a single machine can operate for decades since they are not as heavily used as in a factory, the goal is to teach students, through theoretical and practical methods, the operation of such systems, as well as their integration with other independent systems, and even the development of their own system. With this scenario in mind, the project aims to develop a platform that allows for the compact control of robust and/or educational equipment. To achieve this goal, it proposes the use of embedded systems as the main element of this controller, replacing the traditional desktop computer. Throughout this work, brief references are made to available technologies for this task, and after evaluating each possibility, the proposed solution was the use of the Raspberry PI development board in conjunction with the BigTreeTech Octopus v1.1 controller board. The chosen solution was developed and analyzed through bench tests, evaluating the pulse generation, both in terms of quantity and frequency, as well as in a linear motion guide to analyze the behavior of these pulses, presenting satisfactory results for the control of machines with up to 8 axes, which represents a large portion, if not all, of the desired equipment.

Key Words: CNC, didactic, movement, embedded system

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

A automação e a robótica nos dias de hoje deixaram de ser apenas algo visto em roteiros de filmes ou em livros e passaram a fazer parte do cotidiano, seja aplicado em cenários industriais ou no dia a dia das pessoas, como por exemplo o uso de robôs autônomos para a fabricação de peças e transporte em armazéns e centros de distribuição, e máquinas autônomas para a limpeza residencial.

Em máquinas do tipo CNC (comando numérico computadorizado) geralmente utilizadas para operações que exigem posicionamento preciso com flexibilidade de reprogramação, existem algumas plataformas de comando. Cada fabricante possui a sua própria interface de comando, específica para interagir com o *hardware* próprio e com limitações significativas para se integrar com *hardwares* de outros fabricantes, como por exemplo, os controladores das fabricantes Fanuc, WEG e Siemens, que comercializam soluções em CNC e automação que interagem com os motores e *drivers* próprios. (FANUC, 2024) (SIEMENS, 2024) (WEG, 2024)

Contudo, independentemente de como a máquina será configurada, se faz necessário um sistema que gerencie e controle movimentos, sensoriamentos e outras funcionalidades, podendo ser um sistema autônomo, semiassistido ou totalmente assistido.

Sistemas CNC que possuem arquitetura fechada são excelentes para a indústria pois, embora o custo de instalação, montagem e manutenção sejam mais elevados, tais sistemas garantem a qualidade, a segurança e a repetibilidade de cada processo de fabricação que empregue o uso destas. Por outro lado, os sistemas de arquitetura livre ou, conforme o termo em inglês, *open source*, são sistemas que possuem a arquitetura aberta, permitindo um nível de integração e colaboração muito maior com *hardwares* disponíveis no mercado através de protocolos e interfaces padronizadas, além de possuírem uma maior flexibilidade quanto ao número e disposição dos eixos, permitindo o desenvolvimento de máquinas especiais. (LINUXCNC, 2024)

No âmbito educacional, existem as mais diversas formas e cenários de aplicação que envolvem o uso de máquinas autônomas, começando nas fases iniciais com robôs educativos com base em plataformas já desenvolvidas e prontas para a aplicação, e indo até dissertações e teses de mestrado e doutorado, com os alunos construindo suas próprias plataformas e maquinários.

Em cenários onde se busca desenvolver uma máquina CNC para aplicações de *hobby*, para estudos ou até mesmo para produção de equipamentos de baixo custo (como impressoras 3D), se busca utilizar de tecnologias que possuem arquiteturas *open source* pois além de reduzir os custos com o licenciamento de *software*, a utilização de tais arquiteturas dá a liberdade criativa para os projetos, podendo variar de simples movimentações a máquinas capazes de operar em cenários industriais.

1.1. Definição do Problema

Grandes companhias necessitam de sistemas confiáveis e sigilosos para proteger seus processos, métodos de fabricação, etc... e apesar de ocorrer atualmente uma forte tendência ao uso de arquiteturas abertas na indústria visando a redução de custos e facilidade de manutenção, muitas ainda recorrem a sistemas fechados, onde a interação de hardware e software ocorre apenas entre o fabricante do maquinário ou entre o fabricante e um único fornecedor, como por exemplo, a fabricante brasileira Romi, que possui centros de usinagem com comandos Fanuc e Siemens (ROMI, S/d)

Porém, para o âmbito educacional, seja em universidades ou centros de ensino técnico, onde o foco é o aprendizado sobre o funcionamento de sistemas mecatrônicos e de automação, além da possibilidade de desenvolvimento de novos mecanismos, existe a necessidade de se desenvolver e aplicar sistemas e plataformas capazes de serem modeladas conforme a necessidade ou conforme o tipo de sistema que se deseja aprender ou desenvolver, por exemplo, sistemas de usinagem CNC de bancada, impressoras 3D, entre outros.

1.2. Justificativa e Relevância

Máquinas CNC comerciais são poderosas ferramentas para o processo de usinagem, pois a sua capacidade de movimentação em vários eixos e versatilidade de ferramentas possibilita a usinagem de peças com geometria complexa com alta precisão e qualidade. Porém, tais vantagens trazem como contrapartida o alto custo

financeiro e o fato de serem sistemas fechados e de difícil integração. (CCV INDUSTRIAL, S.d)

Atualmente, existem sistemas controlados por computador em que todo o processamento é realizado pela CPU com uso de *softwares* dedicados, exigindo o uso de máquinas que aumentam o volume total necessário ao funcionamento do equipamento e por vezes superam o próprio tamanho do equipamento em desenvolvimento.

A utilização de sistemas embarcados para a execução destes *softwares* possibilita a inserção destes elementos de controle no interior dos gabinetes de comando e a redução do *footprint,* mantendo a flexibilidade e custos reduzidos destas máquinas.

Dentro do ambiente educacional, existem outros fatores envolvidos para a construção de sistemas CNC, como custos, a possibilidade de ser educacional, entre outros. Existem projetos comerciais voltados para o ensino em sala de aula, como o os tornos e centros de usinagem desktop da fabricante Haas, que além de compactas são de fácil utilização e manutenção, porém com um alto custo de implementação envolvido. Com esta constatação, diversos projetos de máquinas CNC compactas e de baixo custo surgiram nas universidades, porém com o foco sendo na construção eletromecânica da máquina, utilizando softwares e sistemas de comando mais generalistas. (HAAS, 2024)

1.3. Objetivos

Conforme o panorama descrito acima e a justificativa para a construção do projeto, o mesmo possui os seguintes objetivos:

1.3.1. Objetivo Geral

Desenvolver uma solução de controlador reprogramável e flexível baseado em sistema embarcado, para a utilização com máquinas CNC e Robôs didáticos.

1.3.2. Objetivos Específicos

- Levantamento e avaliação de possíveis arquiteturas de controladores CNC didáticos;
- Validação conceitual da arquitetura escolhida (Hardware e Software);
- Construção e implementação do *hardware*;
- Configuração do *software (LinuxCNC)*;
- Documentação da implementação de forma didática para alterações e implementações futuras.

1.4. Estrutura do Trabalho

O trabalho será estruturado conforme as seguintes etapas:

Fundamentação Teórica: serão apresentados os dados dos estudos das referências, bem como o comparativo técnico entre arquiteturas de comando.

Desenvolvimento: No desenvolvimento será descrito as atividades desenvolvidas ao longo do projeto, assim como a justificativa técnica pela arquitetura e sistema embarcado escolhidos e como foi realizada a sua implementação.

Resultados e avaliação de objetivos: Após uma bateria de testes, tanto do hardware quanto do software, foi realizada uma análise dos resultados obtidos sobre o desempenho do sistema como um todo e também do desempenho da arquitetura e sistema escolhidos.

Conclusão: Aqui foi exposta a conclusão do projeto, onde foi definido se o resultado que foi alcançado foi satisfatório ou não, assim como a revisão completa do projeto como um todo afim de apresentar alternativas e sugestões de melhorias para desenvolvimentos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo iremos abordar os princípios essenciais que sustentam a arquitetura de um controlador CNC e seus sistemas envolvidos, explorando desde os aspectos técnicos do LinuxCNC e suas aplicabilidades, até os conceitos e especificações dos sistemas embarcados.

2.1. Comando Numérico Computadorizado

O comando numérico é um método de controle para máquinas com a ideia de se utilizar a interpretação de comandos com números e letras para identificar instruções e, assim, gerar um impulso de saída que fará o controle dos movimentos e demais componentes da máquina. O primeiro protótipo teve sua construção em 1952 no *Massachusetts Institute of Technology* (MIT), e consistia primariamente de uma fresadora vertical copiadora, onde os dados eram transmitidos através de uma fita perfurada e “copiadas” pela ferramenta.

Com o avanço das tecnologias digitais, como circuitos eletrônicos, criação dos primeiros computadores pessoais, entre outras evoluções tecnológicas à época, o comando numérico acompanhou a evolução e assim surgiu o comando numérico computadorizado, em que a fita perfurada dava lugar ao microcomputador, substituindo o controle via *hardware* para o controle totalmente via *software*.

O comando numérico computadorizado traz diversas vantagens em relação a usinagem convencional, como aumento na produtividade com alta taxa de repetibilidade e alta precisão dimensional da peça final, gerando retornos financeiros maiores ao longo prazo. Porém, em contrapartida, os sistemas CNC possuem um maior custo inicial, maior custo de manutenção e demandam um tempo maior para realizar o primeiro *setup* e calibração. (MARCICANO, S/d)

2.1.1. Controladores de Arquiteturas fechadas

Uma máquina CNC com arquitetura fechada tem em sua construção componentes produzidos por apenas um único fabricante, e seu software, na grande maioria das vezes é capaz de comunicar apenas com os componentes do respectivo fabricante.

Integrações com sensores, atuadores e demais componentes são limitadas e a integração entre outros sistemas CNC se dá apenas para um mesmo fabricante. A grande vantagem de tais sistemas é a segurança do processo, pois apenas a fabricante e a empresa na qual adquiriu o equipamento tem controle sobre os componentes internos, e a confiabilidade do sistema em garantir o processo no qual foi projetado para fazer. Porém, tal vantagem traz consigo a desvantagem de A figura abaixo mostra o painel de comando da fabricante Fanuc, modelo 0i-LF, que integram principalmente centros de usinagem e tornos CNC.

Figura 1 - Painel de Comando Fanuc Series 0i-LF

Fonte: Fanuc America (2024)

2.1.2. Controladores de Arquiteturas CNC Abertas

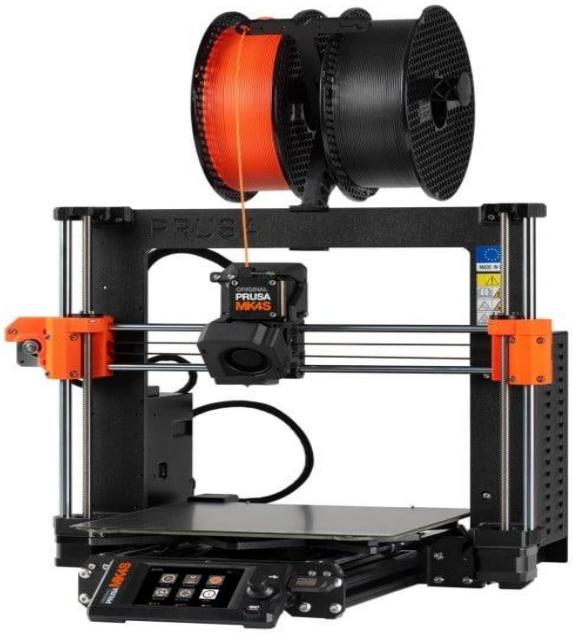
Segundo Pritschow et al. (2001), e segundo a definição do comitê técnico de sistemas abertos da IEEE através da norma IEEE 1003.0, as arquiteturas abertas são estruturas que mantém alta flexibilidade de hardware e diversos níveis de controle de software, devendo ser de fácil integração com outros projetos, algoritmos de

controle, entre outros. Os principais requisitos para um sistema de arquitetura aberta são:

- Extensibilidade: Os módulos da aplicação devem ser capazes de operar em uma plataforma sem conflitos ou interferências mútuas.
- Interoperabilidade: Segundo o conceito, os módulos devem ser capazes de trabalharem juntos, de maneira consistente e capazes de alternar dados entre eles de uma maneira definida.
- Portabilidade: A portabilidade aqui significa que um ou mais módulos da aplicação devem ser capazes de serem portados para diferentes plataformas sem modificações e mantendo suas capacidades.
- Escalabilidade: Seguindo os requerimentos de um determinado projeto, os módulos de aplicação devem ser capazes de serem expandidos ou reduzidos de maneira a manter a sua funcionalidade.

Exemplos práticos de arquiteturas CNC abertas são os sistemas de impressão 3D, pois além de possuírem um *hardware* customizável, o seu *software* é de código aberto, o que possibilita a aplicação do sistema não somente em impressão 3D, mas em outros sistemas que requerem movimentação cartesiana. A figura abaixo mostra uma impressora 3D da marca Prusa, modelo MK4S, que utiliza tais *hardwares* e tem seu software baseado no firmware Marlin. (PRUSA3D, 2024)

Figura 2 - Impressora 3D Prusa MK4S



Fonte: Prusa3D (2024)

2.1.3. Controladores Baseados em PC

Sistemas CNC com arquitetura baseada em PC são normalmente arquiteturas abertas onde um computador é o responsável pela leitura e processamento das informações dos sensores e encoders, bem como a geração de sinais para os sistemas de movimentação.

O principal meio de comunicação entre o computador e a máquina é realizado através da porta paralela, que é um *hardware* de custo reduzido sendo normalmente agregado ao computador, e que pode realizar de maneira eficiente a sincronização do controle do hardware externo no qual esteja conectado.

Porém, as portas paralelas apesar de serem utilizadas ainda hoje em alguns equipamentos específicos, são consideradas uma tecnologia obsoleta e por conta disto, enfrentam alguns desafios quando se trata de ambientes industriais devido a instabilidades e falta de segurança. (SILVEIRA, 2024)

Abaixo a figura mostra um exemplo de um controlador CNC baseado em PC com *Windows* 7 e o *software* Mach3.

Figura 3 - Controlador CNC Baseado em PC

Fonte: IESSA (2024)

2.1.4. Placas Controladoras Vs. Placas de Interface

A conexão entre computador e máquina CNC não é feita diretamente, sendo necessário um meio entre a porta paralela ou outra porta de comunicação presente e o equipamento que se deseja controlar. O circuito mais comum de ser utilizado são as placas adaptadoras e placas de interface, estes com o objetivo de realizar a adaptação entre o computador e a máquina na qual está comandando.

As placas adaptadoras fazem a simples adaptação dos pinos da porta paralela para os respectivos pinos de comando de um driver, sem contar com uma proteção adicional contra surtos ou amplificadores de sinal. Já as placas de interface contam com optoacopladores ou *buffers*, que funcionam como isoladores, separando o circuito da porta paralela do restante do sistema e, assim, garantindo a segurança tanto da porta paralela quanto do computador conectado a ela.

As placas controladoras, também conhecidas como placa de controle de movimento, surgiram com o avanço da tecnologia de computadores, que foi eliminando gradativamente as portas paralelas e abrindo espaço para comunicação serial, tendo como exemplo mais conhecido o *Universal Serial Bus* (USB). Isso fez com que fosse necessário o uso de microcontroladores capazes de transformar os sinais recebidos via comunicação serial em sinais de comando, realizando o cálculo de trajetórias de movimentação localmente. As placas controladoras, devido a atuar como um circuito independente e dedicado, não possuem a interferência de processos paralelos como ocorre no PC, o que melhora a geração de pulsos com ganhos em frequência e latência dos mesmos, além da possibilidade de se comandar mais I/Os dedicados. (SILVEIRA, 2024)

2.2. Sistemas embarcados

O conceito de sistemas embarcados surge do inglês *Embedded Systems*, que é um sistema capaz de operar independentemente de um computador ou até mesmo de uma fonte de energia fixa, como uma tomada ou gerador.

Um sistema embarcado por definição são componentes que interagem com o ambiente a sua volta através de sensores, possuem unidade de processamento e é integrado com o restante do hardware via um software dedicado conhecido como

firmware e estão aplicados em um outro sistema mais complexo, como por exemplo aparelhos celulares, televisores, micro-ondas, entre outros.

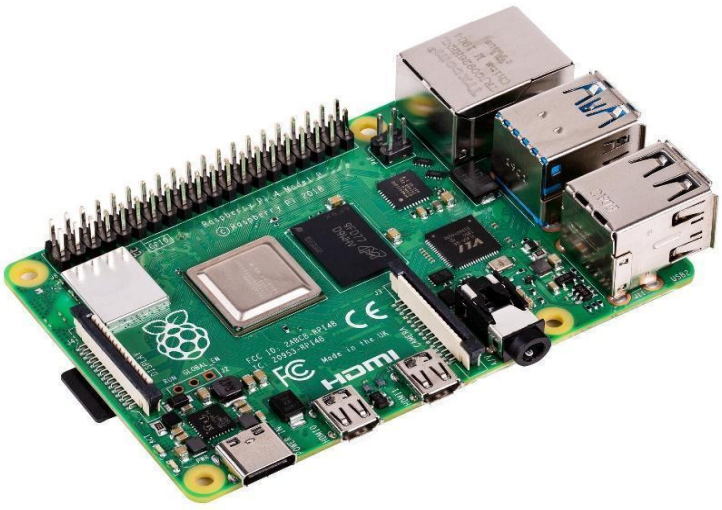
Um sistema embarcado possui características que os diferenciam de sistemas como um computador pessoal como, por exemplo, a funcionalidade única, maiores restrições de projeto, e reação a eventos de tempo real.

A vantagem de um sistema embarcado é a sua vasta aplicabilidade em uma ampla variedade de projetos, como sistemas eletrônicos de veículos, eletrodomésticos inteligentes, entre outros projetos. Porém, alguns tipos de sistemas embarcados possuem uma grande dificuldade em serem reprojetados, sendo necessário refazer todo o projeto em caso de atualizações de *hardware.* (BARROS; CAVALCANTE, 2024)

2.2.1. Raspberry PI

Desenvolvida pela *Raspberry Pi Foundation*, é uma placa de desenvolvimento que utiliza um microprocessador com arquitetura ARM e é popularmente conhecida principalmente por sua versatilidade e poder computacional, sendo por vezes suficiente para substituir um PC convencional, e pela capacidade de atuar como placa de monitoramento e controle de periféricos através de suas portas GPIO e através de outros meios de comunicação seriada como SPI, USB, I2C, entre outros. A figura 1 apresenta uma imagem da Raspberry 4 como referência.

Figura 4 - Placa de Desenvolvimento Raspberry Pi 4



Fonte: Newark (2024)

2.2.2. ESP32

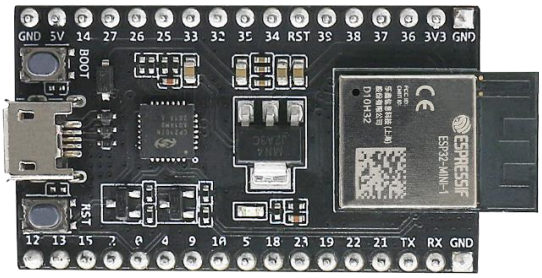
Desenvolvido pela Espressif, o ESP32 é um microcontrolador *System-On-Chip* com uma variedade de periféricos como I2C, SPI e UART. Com um processador de 2 núcleos de processamento e uma ampla capacidade de gerenciamento de I/Os, além de baixo consumo de energia, a ESP32 é amplamente aplicado em diversos cenários IoT, incluindo aplicação em projetos de/ automação residencial e industrial.

Sua grande vantagem é ser compacto ao ponto de poder ser adaptado em qualquer placa através de módulos, ou utilizado como uma placa de desenvolvimento, similar ao Arduino, além de possuir conectividades como WiFi e Bluetooth.

Porém a ESP enfrenta algumas limitações a depender da sua aplicação devido as baixas tensões das suas saídas, principalmente para aplicações industriais onde os equipamentos utilizam de maneira geral uma tensão igual ou superior a 5V e suas saídas podem fornecer apenas 3,3V, sendo necessário elevar ou diminuir a tensão para atingir compatibilidade com outros equipamentos. (ESPRESSIF, 2023.) (NABTO, 2024).

A figura 5 apresenta uma versão deste *hardware* para exemplo.

Figura 5 - Placa de Desenvolvimento ESP-32 Devkit



Fonte: Espressif (2024)

2.2.3. Arduino

Inicialmente construído pelos pesquisadores David Cuartielles, Massimo Banzi, Tom Igoe, David Mellis e Gianluca Martino no *Ivrea Interaction Design Institue* em 2005, o Arduino é uma plataforma *open source* de desenvolvimento de hardware e software baseado em microcontrolador AVR (ATMega328P) e linguagem de programação própria baseada em *Wiring*.

Foi projetada para ser uma plataforma de prototipagem de baixo custo com alta adaptabilidade, tendo como público-alvo estudantes sem um conhecimento prévio de eletrônica e/ou programação, porém devido às suas características técnicas e custo, rapidamente tomou proporções maiores e atualmente embarca desde projetos simples como controlar um LED, projetos de robótica como controle de movimentação e leitura de sensores até projetos científicos de alto nível. (ARDUINO LLC, 2023). A figura 3 exemplifica uma das versões do hardware mencionado o UNO R3.

Figura 6 - Placa de Desenvolvimento Arduino Uno R3

Fonte: Arduino LLC (2024)

2.3. Software

A seguir iremos trazer os conceitos dos principais softwares de controle CNC, especialmente aqueles envolvidos no desenvolvimento do projeto.

2.3.1. Mach3:

Desenvolvido pela *Artsoft,* empresa pertencente à *Newfangled Solutions,* o Mach3 é um software desenvolvido para *Windows* em que é possível comandar e supervisionar máquinas CNC, sendo necessário apenas um computador com o sistema operacional *Windows* e uma porta paralela conectada a placa mãe do computador. Posteriormente recebeu pacotes de atualização para incorporar a utilização de outros *hardwares* de comunicação não sendo limitada ao uso da porta paralela.

Atualmente, uma nova versão do software, o *Mach4*, está disponível para a comunidade. Embora receba a mesma nomenclatura do antecessor, o *Mach4* é voltado para controle e automação de máquinas industriais de alta precisão e confiabilidade, porém a sua aceitação não foi tão alta devido ao custo de licenciamento maior.

É uma solução robusta e confiável, e foi impulsionada por diversos entusiastas construtores de máquinas norte-americanos, porém é um *software* licenciado, e seu custo acaba sendo um dos fatores dominantes para *makers* e entusiastas com orçamento mais limitado. (MACH3, 2024)

2.3.2. LinuxCNC

Baseado em sistema operacional *Linux*, mais especificamente o Debian, o *LinuxCNC* é um sistema operacional completo, capaz de operar em equipamentos que possuem suporte a instalação do kernel *Linux*, variando entre computadores de mesa convencionais até placas de desenvolvimento, como Raspberry PI. Devido ao *LinuxCNC* ser *open source*, o custo de implementação do software é reduzido drasticamente quando comparado com o *Mach3* mesmo contendo características similares por conta do custo de licenciamento e manutenção.

Para que o *LinuxCNC* pudesse atingir o conceito de uma arquitetura aberta, era necessário que ele fosse capaz de ser integrado a qualquer hardware sem dificuldades ou sem a necessidade de alterar as características do mesmo, pensando nisso desenvolveu-se a camada HAL, da sigla *Hardware Abstraction Layer* ou Camada de Abstração do *Hardware*, que consiste em uma camada implementada em programação de alto nível capaz de distribuir os sinais emitidos pelo hardware, enviá-lo ao software e vice versa. A camada é formada principalmente por blocos de construção constituídos de drivers de dispositivos e o diferencial é a configuração para o usuário final que fica simplificada, bastando conectar os pinos HAL com os pinos físicos onde estão entrando os sinais.

O sistema também é capaz de ser utilizado em sistemas embarcados como *Beaglebone* e *Raspberry PI,* o que auxilia o desenvolvimento de sistemas mais compactos e em algumas aplicações podendo ser inclusive portáteis.

Porém o sistema tem suas desvantagens, como possuir uma interface não intuitiva para iniciantes e por ser compatível apenas com portas paralelas e GPIOs, limitando o uso de placas controladoras com comandos através de USB e Ethernet devido ao alto custo de *hardware*. (LINUXCNC, 2024) (SILVEIRA, 2024)

2.4. Firmware

O *Firmware* é um tipo de software responsável por realizar a comunicação entre componentes e subcomponentes de um sistema. A principal diferença entre um firmware e um software convencional é que o firmware providencia apenas as instruções básicas para o correto funcionamento do hardware, diferente do software que pode ter uma ou mais funções e aplicações conforme o seu desenvolvimento. O firmware pode ser classificado em:

- ***Firmware bare-metal****:* É o firmware desenvolvido para rodar diretamente no hardware, sem a necessidade de um sistema operacional por trás. É conhecido por este nome por não possuir nenhuma camada intermediária entre o firmware e o hardware, sendo assim, aplicado em operações de baixo nível como roteadores, controladores de dispositivos e também aplicações embarcadas como sistemas náuticos e aeronáuticos, centrais veiculares.
- ***Firmware high-level***: Firmware desenhado para atuar em conjunto com um sistema operacional. Possui suporte a atualizações diretas e pode ser armazenado em uma memória flash. Amplamente utilizado em computadores e smartphones. (TECHTARGET, 2024)

2.4.1. Marlin

O *firmware* Marlin é derivado dos códigos fonte de outros dois *firmwares,* o *Sprinter* e o GRBL, que são códigos destinados a interpretação de comandos G-Code. Originalmente desenvolvido para máquinas de impressão 3D da *RepRap*, obteve uma rápida aceitação do público entusiasta e logo se popularizou entre a comunidade *maker,* o projeto se tornou *open source* em 2011 com uma licença

GPLv3, portanto, podendo ser utilizada em diversos projetos, sendo o *firmware* base de grandes marcas como *Ultimaker*, *Creality3D* e *Prusa*.

O seu grande diferencial é ter sido construído com base no framework do Arduino, sendo assim, extremamente versátil e compatível com diversos tipos de microcontroladores desde o *ATMega* AVR de 8-bits até a mais recente e robusta solução ARM 32-bits. Outros destaques do Marlin são a possibilidade de comandar até 10 eixos lineares ou rotativos, uma biblioteca G-Code completa com 150 comandos e suporte a diversas cinemáticas, como Delta, SCARA, cartesiana, entre outros. Além disso, ele é capaz de operar no modo *stand alone*, ou seja, sem a necessidade de um sistema operacional, podendo receber e interpretar comandos *G-Code* recebidos através de fontes de armazenamento como cartões SD, *pen drives*, entre outros, ou aliado a um *parser*, que consiste em código em baixo ou alto nível que traduz os comandos recebidos e também é capaz de enviar os comandos.

A grande desvantagem do Marlin é ter uma velocidade de processamento limitada quando comparado com outras alternativas como o Remora e o Klipper. Além disto, também não possui uma configuração intuitiva e requer uma constante regravação a cada atualização ou implementação de novas funcionalidades no *firmware.* (MARLIN, 2024) (P3D, 2024)

2.4.2. Klipper

O Klipper é um firmware para impressão 3D, desenvolvido inicialmente por Kevin O'Connor com o objetivo de aproveitar ao máximo o processamento do sistema embarcado no qual está instalado, sem a necessidade de placas adicionais.

O Klipper é capaz de atuar nas mais diversas arquiteturas de microcontroladores, como por exemplo AVR, ARM, PRU, entre outros, o que possibilita a portabilidade de código entre diferentes sistemas. Como grande parte do código é escrito em Python, a customização e adaptação do firmware ficam mais simplificadas comparado com a outros firmwares.

O *Klipper* tem uma grande vantagem de poder ser instalado diretamente em sistemas embarcados como *Raspberry PI,* aproveitando assim seu poder de processamento e tornando as impressões mais precisas e com melhores acabamentos. Em contrapartida, pode não ser intuitivo para iniciantes e conta com um

suporte um pouco mais limitado devido a comunidade reduzida em relação aos demais concorrentes. (KLIPPER, 2024)

2.4.3. Remora

O Remora foi desenvolvido inicialmente em 2017 pelo desenvolvedor Scott Alford com o intuito de ser uma solução em que fosse possível o usuário integrar com facilidade uma *Raspberry Pi* com placas controladoras encontradas comercialmente ou até mesmo desenvolvidas de maneira independente e utilizar o sistema LinuxCNC. O primeiro protótipo do projeto usou uma *Raspberry Pi 3* e um *Arduino Due* para comandar a placa de 32-Bit RADDS.

Os testes foram bem sucedidos, no entanto, conforme mais placas foram sendo introduzidas no mercado, o framework do Arduino se tornou cada vez mais limitado, sendo necessário uma mudança no framework e a migração para plataformas LPC17xx.

Atualmente o Remora é baseado na plataforma *bare-metal* Mbed OS6, conta com suporte para placas baseadas em Ethernet, desenhadas originalmente para sistemas *Mach3*. Conta também com o apoio da empresa Expatria Technologies, que além de desenvolver placas voltadas para o remora, ainda auxilia com o desenvolvimento do código-fonte. (Remora, 2023)

As tabelas abaixo sintetizam as informações de hardware e firmware descritas nos itens 2.2. e 2.4. trazendo as principais vantagens e desvantagens de cada um.

Tabela 1 - Vantagens e desvantagens das placas de desenvolvimento

Hardware	Vantagens	Desvantagens
Raspberry PI	• Capacidade de Rodar SO completo, substituindo o PC convencional; • Vasta quantidade de I/Os para controle; • Pode ser configurado para atuar como um sistema embarcado dedicado.	• Alto custo dependendo do modelo escolhido; • Necessária configuração a nível de SO para aplicações em Real Time;
ESP32	• Microcontrolador com dois núcleos de processamento e com frequências de operação mais altas • Capaz de realizar tarefas em Real Time • Possui canais ADC embarcados	• Pouco poder computacional quando comparado com uma Raspberry PI • Limite de memória RAM e ROM • Falta de suporte para protocolos industriais
Arduino	• Compatível com firmwares de movimentação CNC, como o GRBL • Compatibilidade e suporte a diversos *shields* e expansões • Suporte ativo da comunidade	• Limitação no processamento devido a arquitetura do controlador (8-bits, 16MHz) • Limitação na conectividade e interação com interfaces industriais como Profibus e Modbus

Fonte: Do Autor (2024)

Tabela 2 - Vantagens e desvantagens do firmware

Firmware	Vantagens	Desvantagens
Marlin	• Firmware popular, com amplo suporte e comunidade ativa • Compativel com microcontroladores mais simples como o ATMega328P • Recebe Melhorias e Atualizações com frequencia	• Complexo para realizar configurações manuais • Não possui suporte nativo para G-Code, sendo necessário uma adaptação. • Possui limitações no desempenho quando utilizado em microcontroladores simples.
Klipper	• Alto desempenho por conta de o processamento ser direto no computador. • Possibilita controle de movimento com alta precisão • Possui diversos recursos avançados como controle de entrada e compensação de vibração	• Necessita de um computador externo ao controlador • Caso haja instabilidade no sistema do computador, a comunicação com o resto do sistema pode ficar comprometida.
Remora	• Possui uma baixa latência de operação, aumentando a precisão do movimento. • Funcional em diversas placas controladoras, aumentando sua flexibilidade	• Documentação mais limitada quando comparado com o Marlin e o Klipper • Depende de hardware compatível para o correto funcionamento • Curva de aprendizado mais íngreme caso se utilize um FPGA.

Fonte: Do Autor (2024)

3. DESENVOLVIMENTO

Neste capitulo será determinado todo o escopo do projeto, abrangendo desde seus requisitos iniciais, comparativos e justificativas para a escolha dos componentes de *hardware* e *software*

3.1. Seleção de componentes

O projeto envolve o desenvolvimento de uma plataforma controladora CNC com os seguintes requisitos:

- **Baixo Custo**: O custo é um fator importante e decisivo para aplicações tanto industriais quanto independentes, portanto, o sistema deve ser desenvolvido utilizando materiais que possuam boa disponibilidade com o menor custo possível, viabilizando o desenvolvimento.
- **Fácil customização:** Por ser uma plataforma didática, o sistema deve ser montado de forma a se adaptar a quaisquer aplicações que seja submetido.
- **Robustez:** Por se tratar de uma plataforma didática que será projetada e reprojetada diversas vezes, o sistema tem como requisito ser robusto, de maneira que possa garantir o correto funcionamento em diversas circunstâncias.
- **Didático:** O projeto é direcionado para o mundo acadêmico e ao aprendizado aprofundado sobre sistemas e arquiteturas CNC, portanto deve ser construído de maneira didática e que facilite o aprendizado daqueles que irão manusear o equipamento.
- **Disponibilidade:** Os materiais requisitados para a construção do projeto devem estar disponíveis na instituição ou serem de fácil aquisição.

Em função do amplo uso do *LinuxCNC* na Engenharia Mecatrônica do IFSC e em razão dos benefícios de flexibilidade e custo citados na fundamentação teórica deste trabalho, optou-se por determinar este como padrão para o projeto. Esta opção permite que o sistema seja utilizado em qualquer *hardware* que suporte a distribuição do *Linux*, no entanto é preciso avaliar se este *hardware* é capaz de respeitar os requisitos de execução dos sistemas CNC.

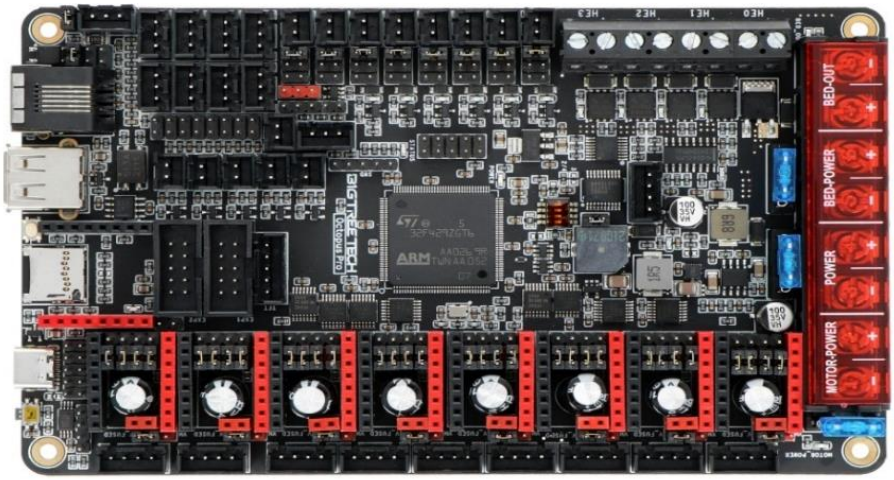
Observam-se, portanto duas principais opções de arquitetura de comunicação de sinais com periféricos. Na primeira opção são utilizadas as GPIO das placas para a construção de um meio de comunicação entre *hardwares* bastante semelhante a porta paralela dos computadores, podendo inclusive utilizar-se placas de interface para isolamento e proteção dos sistemas embarcados. A figura abaixo demonstra o modelo citado acima, com uma *Raspberry PI* conectada via GPIO a um *Shield* para comando CNC.

Figura 7 - Raspberry PI conectada a um *Shield* para comando CNC via GPIO

Fonte: Protoneer Wiki (2024)

A segunda opção seria a utilização de placas controladoras que sejam capazes de operar com o sistema *LinuxCNC* por meio de interfaces presentes em ambas as placas, como USB, Ethernet ou SPI. A figura abaixo é um modelo de placa controladora *BigTreeTech Octopus Pro* que é capaz de comandar sistemas de movimentação através de comandos recebidos via USB, GPIO e Ethernet.

Figura 8 - Placa Controladora BigTreeTech Octopus Pro



Fonte: Botland Store (2024)

A primeira opção envolveria o desenvolvimento de um *shield* capaz de transformar as GPIO da placa escolhida em uma porta paralela, no entanto esta opção limita o sistema exclusivamente a interface e suas limitações, principalmente em relação ao número máximo de entradas e saídas, o que permite um número máximo de 6 eixos controlados, ao custo do esgotamento de saídas.

A utilização de uma placa controladora permite maiores velocidades de execução do que aquelas esperadas por uma placa de interface, possibilita um expressivo aumento no número de entradas e saídas disponíveis, de acordo com o hardware escolhido. E principalmente, ao deliberar que a parte mais crítica da geração de pulsos ocorra em um *hardware* dedicado, aumenta a chance de sucesso do projeto. Entretanto um dos maiores problemas desta abordagem é o custo das placas controladoras e a escassez de placas compatíveis com o LinuxCNC.

Buscando se distanciar das limitações impostas pela porta paralela, optou-se pela segunda alternativa, dentre as possíveis implementações desta alternativa, foi encontrado o projeto Remora, o qual se trata de uma solução combinada, inicialmente entre a Raspberry como sistema de processamento em que o LinuxCNC será instalado e uma lista de placas controladoras compatíveis com o *firmware*. Como boa parte destas placas são placas controladoras de impressoras 3D, o custo e disponibilidade destas é mais atrativo do que outras soluções comerciais, tornando o projeto viável.

3.1.1. Sistemas Embarcados

Uma vez selecionada a arquitetura a ser validada, é preciso realizar a seleção dos componentes que a integram. O primeiro deles seria o sistema embarcado em que o sistema operacional irá rodar. Como dito anteriormente o Remora utiliza a *Raspberry Pi* como base para o seu desenvolvimento, embora a priori possam ser utilizados outros *hardwares* com as devidas alterações, opta-se pela manutenção desta opção em razão da documentação e validação prévia. Restando apenas a determinação de qual versão desta placa utilizar. Para isso, na tabela abaixo será demonstrado um comparativo entre as opções consideradas como mais atrativas, de modo a definir qual plataforma de desenvolvimento embarcada atende os critérios definidos no item 4.1.

Tabela 3 – Comparativo entre Placas Raspberry Pi

	Raspberry PI 3B	Raspberry PI 4	Raspberry PI 5
CPU	Broadcom BCM2837	Broadcom BCM2711	Broadcom BCM2712
Frequência de Operação	1.2GHz	1.8 GHz	2.4 GHz
GPU	Integrada ao SoC	Integrada ao SoC	VideoCore VII GPU
RAM	1GB LPDDR2 SDRAM	1GB, 2GB, 4GB or 8GB LPDDR4-3200	4GB and 8GB LPDDR4X-4267 SDRAM
Alimentação	5VDC 2.5A (Via USB. Suporta PoE)	5VDC 3A (Via USB e GPIO. Suporta PoE)	5VDC 5A (Via USB. Suporta PoE+)
Quantidade I/Os	40	40	40
Custo (R$)	419,90	664,90	673,45
Disponibilidade	Sim	Sim	Não

Fonte: Do Autor (2024)

Ao avaliar a tabela e os critérios de seleção, foi definido que a plataforma de desenvolvimento utilizada será a Raspberry PI 4. Apesar de ser tecnicamente superior, a Raspberry PI 5 não atendeu o critério de disponibilidade no campus e para a utilização da mesma seria necessário realizar a aquisição da placa, o que elevaria o custo do projeto. A Raspberry Pi 3B participou de testes iniciais, mas não apresentou resultados satisfatórios. A versão disponível no momento do projeto era a Raspberry Pi 4 Modelo de 4GB de memória RAM.

3.1.2. Placas Controladoras

A lista de placas controladoras atualmente compatíveis com o projeto Remora é limitada, mas está em expansão, existe ainda a possibilidade de compilar o firmware e realizar alterações para acomodar *hardwares* específicos que utilizem microcontroladores de placas já suportadas, o que poderia aumentar ainda mais esta lista, mas necessitaria de um tempo maior de desenvolvimento e validação. Tendo em mente a aplicação na mecatrônica em todos os projetos possíveis, optou-se pela validação de uma opção que possua ao menos 6 eixos de movimentação, o que permitiria inclusive o controle de robôs antropomórficos e outros equipamentos com maior quantidade de eixos. Após serem definidos os critérios, na tabela abaixo será demonstrado um comparativo entre placas controladoras compatíveis com o Remora e com número de eixos superior a 6, de modo a definir qual placa atende os critérios definidos no item 4.1

Tabela 4 - Comparativo entre Placas Controladoras

	BigTreeTech Octopus v1.1	**BigTreeTech Octopus PRO**	**MKS Monster 8 v2**	**Fysetec Spider**
CPU	STM32F446ZET6	STM32F429ZGT6	STM32F407VET6	STM32F446ZET6
Drivers	8	8	8	8
Interfaces	UART, SPI e STEP/DIR	UART, SPI e STEP/DIR	UART, SPI e STEP/DIR	UART, SPI e STEP/DIR
Fans	8	8	6	6
Energia	4 conectores	4 conectores	-	3 conectores
Protocolos	CAN BUS, I2C	CAN BUS, I2C	Built-in CAN, V-USB,	I2C
Uso com Raspberry PI	Sim	Sim	Sim	Sim
Preço*	R$ 249,91	R$ 301,20	R$ 261,58	R$ 213,27

*Valores de referência, obtidos a partir de sites de importação sem inclusão dos custos oriundos deste processo.

Fonte: Do Autor (2024)

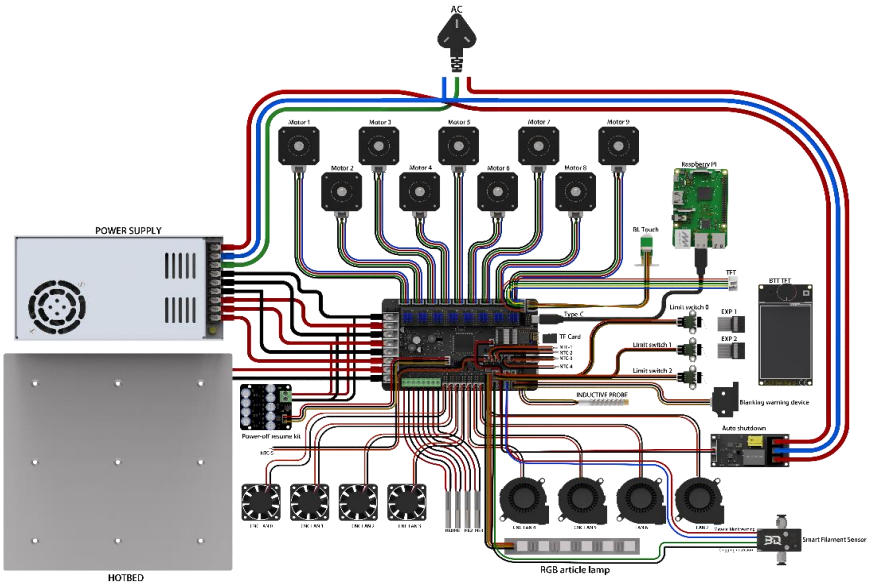
Ao avaliar a tabela acima, percebemos que as placas controladoras são muito similares no que se diz a tecnologias embarcadas e até mesmo o fabricante do

microcontrolador embarcado. As placas controladoras acima são compatíveis com o uso conjunto com sistemas embarcados e estão listadas como placas suportadas pelos principais softwares, como Marlin, Klipper e Remora. Porém, a placa que melhor atendeu os critérios de disponibilidade e compatibilidade com o *LinuxCNC* foi a BigTreeTech Octopus v1.1, embora não seja a placa mais em conta, foi a que apresentou maior disponibilidade de anúncios.

3.2. Montagem do sistema

O hardware escolhido foi montado seguindo o guia de montagem disponível no GitHub do projeto Remora. A imagem abaixo demonstra um exemplo de montagem do sistema voltado para impressão 3D

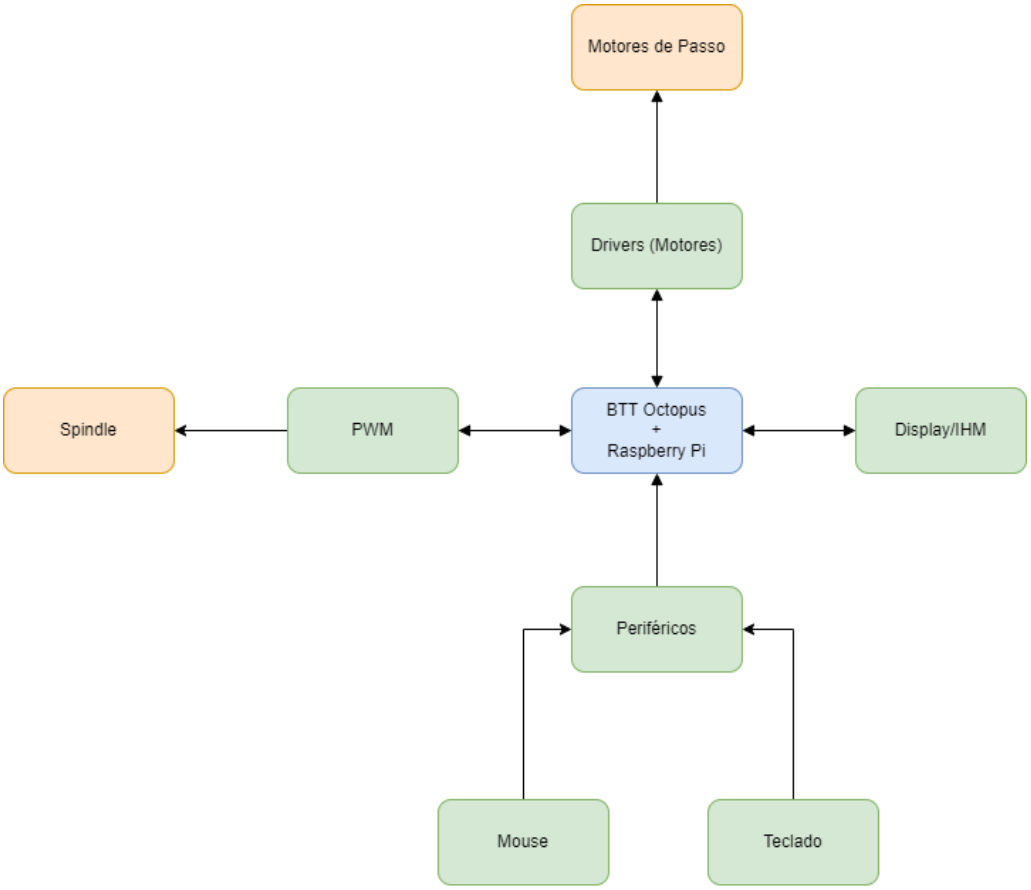
Figura 9 - Exemplo de conexão da BigTreeTech Octopus com a Raspberry Pi para controle de impressora 3D



Fonte: BigTreeTech GitHub (2021)

O fluxograma abaixo demonstra as conexões entre os componentes do sistema escolhido e indica como será o seu funcionamento principal

Figura 10 - Fluxograma de funcionamento do sistema



Fonte: Do Autor (2024)

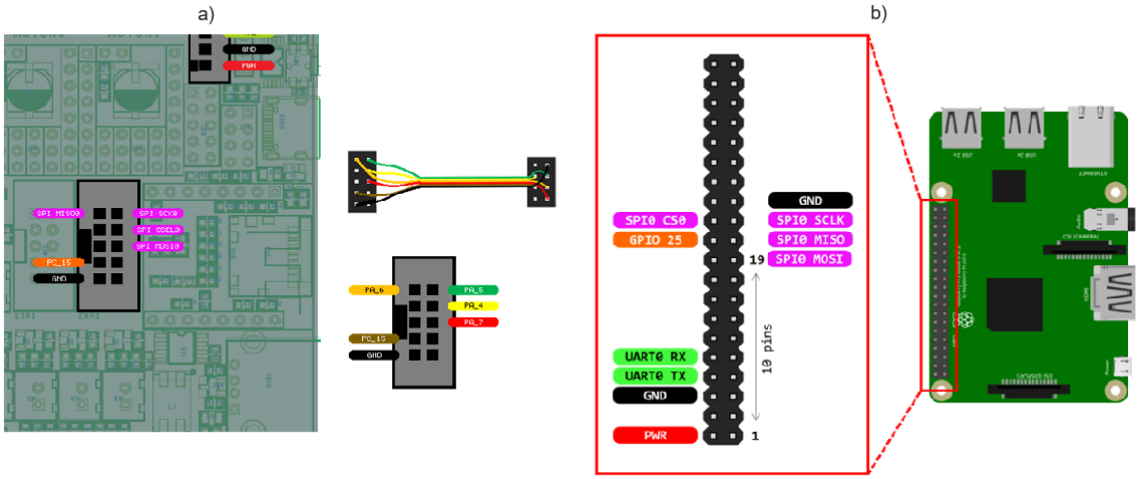
Conforme mostram os diagramas de conexão e a tabela 3, a conexão de comando entre a Raspberry PI e a placa controladora BTT Octopus foi feita utilizando os pinos GPIO de 19 a 24 nos pinos PA_4, PA_5, PA_6, PA_7 e PC_15 através de um cabo com conector de 10 pinos, assim, habilitando a comunicação via SPI entre as duas placas. A Tabela 3 apresenta as conexões realizadas que também são ilustradas graficamente na figura 11:

Tabela 5 - Pinagens de Conexão entre Raspberry PI – BTT Octopus

Pinos Raspberry Pi	Pinos BTT Octopus
GPIO19	PA_7
GPIO21	PA_6
GPIO22	PC_15
GPIO23	PA_5
GPIO24	PA_4

Fonte: Do Autor (2024)

Figura 11 - a) Diagrama de conexão BTT Octopus / b) Diagrama de conexão dos pinos Raspberry PI



Fonte: Adaptado de Remora Docs

3.2.1. Configuração do Hardware

A comunicação entre a placa BTT Octopus e os drivers de controle de motores pode ser configurado para atuar de três maneiras, sendo necessário configurar jumpers para cada modo de operação. Os modos de operação são:

- UART: do inglês Receptor-Transmissor Assíncrono Universal, é um hardware do tipo microchip que possibilita a comunicação entre um dispositivo de computação (PC, Sistemas Embarcados, etc...) e outros equipamentos. O funcionamento do microchip consiste basicamente na conversão de um sinal serial para um sinal paralelo e vice-versa. A comunicação pode ser unidirecional, bidirecional com espera, e bidirecional simultâneo.
- SPI: o SPI é um protocolo de comunicação que utiliza o sistema mestre-escravo para realizar comunicação serial entre qualquer dispositivo com envio de informações via clock com alta velocidade.
- Modo Step/Dir: É um protocolo de comunicação que realiza a leitura de dois sinais digitais, STEP e DIR, para determinar, respectivamente, a quantidade de movimento que o motor deve realizar e a direção deste movimento.

Os modos UART e SPI requerem drivers que atuem usando estes protocolos de comunicação os quais ainda são limitados em relação a disponibilidade, principalmente em drivers de maior potência, enquanto o modo Step/Dir é amplamente utilizado com os mais diversos tipos de drivers de motores de passo e servo motores. A figura 10 apresenta na placa como deve ser configurado cada modo de funcionamento.

Figura 12 - Jumpers de configuração da placa Big Tree Tech Octopus: a) Modo UART, b) Modo SPI, c) Modo Step/Dir

Fonte: Adaptado de Manual do Usuário BTT Octopus

3.2.2. Configuração do Software

O *LinuxCNC* conta com diversas versões para instalação em sistemas embarcados, sendo algumas customizadas para *hardwares* específicos. Os testes deste projeto foram realizados com duas distribuições diferentes, a primeira utilizando a versão 2.9.2 baseada em Debian *Bookworm*, e a segunda baseada na versão 2.8.x do *LinuxCNC*, feita utilizando como base o sistema *Raspbian* (Ou *Raspberry Pi* OS) e customizada pela *Expatria Technologies* para ser leve, simples e de fácil uso para as placas baseadas em Remora.

Para a instalação, tanto da versão oficial quanto da versão customizada, foi utilizado o próprio instalador da Raspberry PI, o *Raspberry Pi Imager*, pois ele conta com uma interface intuitiva, e é compatível com as versões testadas neste projeto.

O *firmware* utilizado foi a versão configurada para o chip STM32F446, que é o chip presente na placa escolhida para este projeto. A configuração de quais recursos da placa e como estes serão utilizados, assim como algumas configurações que a placa precisa para realizar os cálculos de trajetória são editadas em um arquivo "config.txt" de acordo com a configuração escolhida. O arquivo pode ser desenvolvido totalmente do zero em JSON, ou pode ser baseado em exemplos que acompanham a instalação do *firmware* ou do *LinuxCNC*:

- **Configuração XYZ**: Arquivo de configuração que habilita os drivers dos eixos X, Y e Z para movimentação em ambos os sentidos através da configuração Step/Dir.
- **Configuração Ender3**: Arquivo de configuração direcionado para as impressoras *Creality Ender 3,* onde são habilitados todos os recursos da impressora, como habilitação dos 3 eixos de movimentação cartesiana e o eixo tracionador correspondente ao bico injetor, habilitação de pinos PWM para controle de *fans*, fins de curso físicos e as saídas de aquecimento das mesas e do bico injetor.

Estas duas versões deste arquivo de configuração foram utilizadas para execução de testes iniciais no firmware neste projeto e seus códigos podem ser verificados na integra nos anexos 2 e 3, bem como o passo a passo para a configuração completa do software pode ser verificado no manual em apêndice.

4. TESTES E RESULTADOS

Este capítulo irá descrever os testes realizados com a plataforma escolhida durante o desenvolvimento com o objetivo de validar tanto a construção do sistema quanto a sua capacidade de atuar em sistemas de movimentação e os resultados obtidos serão discutidos assim como será feito um levantamento das limitações do sistema.

4.1. Metodologia de Testes

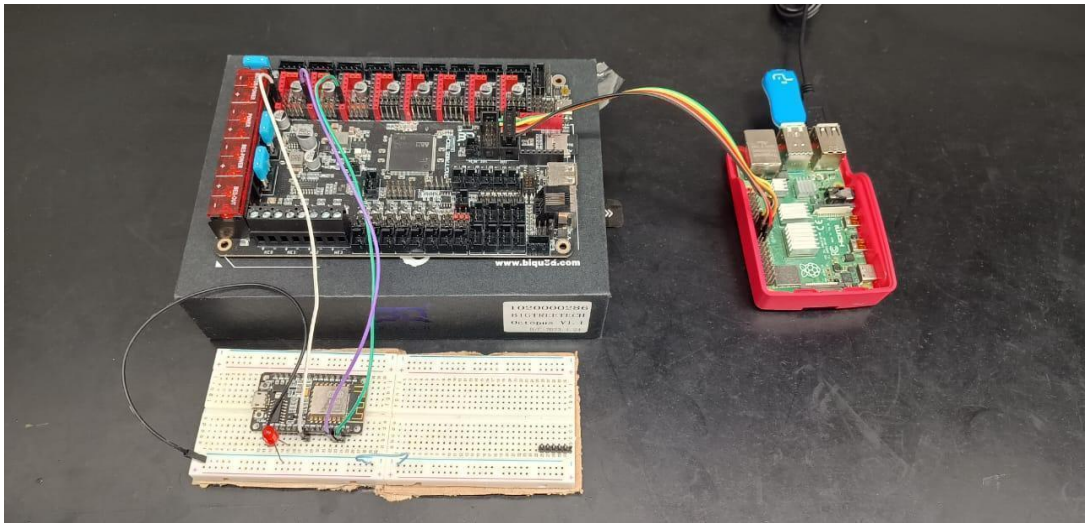
Após escolhida e montada a plataforma a ser avaliada, os testes foram realizados observando como os principais fatores a capacidade de geração de pulsos, a quantidade de pulsos gerados pela placa controladora e a estabilidade de comunicação com o sistema embarcado.

O sistema foi montado e configurado seguindo os passos descritos nos itens 3.6, 3.7 e 3.8 deste trabalho, bem como o manual de instruções em anexo ao mesmo. Para verificação da frequência dos pulsos gerados, utilizou-se um osciloscópio digital de quatro canais para garantir que fosse possível realizar as medições em mais de um eixo.

Porém, o osciloscópio garante apenas a medição da frequência, tensões de operação e exibe a forma de onda do pulso, para este trabalho se fez necessário fazer o controle também da quantidade de pulsos gerados pelo eixo para garantir a precisão na comunicação com o driver e, consequentemente, nos movimentos dos motores. Para fazer tal controle, foi desenvolvido um código em C/C++ aplicado em uma ESP8266 onde a GPIO da ESP se conectava aos pinos Step/Dir realizando a medição da quantidade de pulsos gerada. Para registrar os pulsos gerados em cada teste, foi desenvolvido um código em Python sendo que o computador se comunicava com a ESP via USB, recebendo os dados e registrando em uma planilha para controle posterior.

A figura abaixo mostra como foi montado o sistema em bancada com a ESP conectada nos pinos step/dir do eixo 0:

Figura 13 - Montagem do hardware em bancada com sistema de contagem de pulsos



Fonte: Do Autor (2024)

Os pulsos foram gerados através de comandos *jog* e G-Code inseridos na interface do LinuxCNC, simulando uma movimentação linear em um ou mais eixos, simultaneamente ou não.

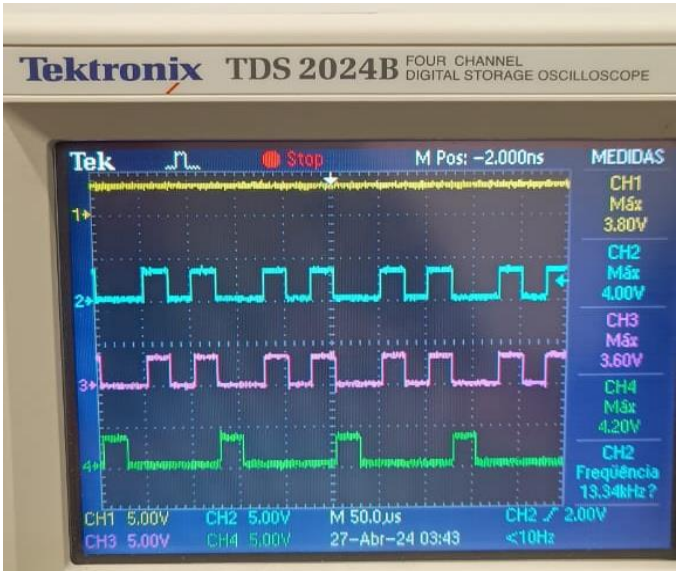
Um dado importante para a realização dos testes é a determinação da escala de movimentação. Esta variável descreve ao *software* qual a relação entre pulsos gerados e deslocamento linear ou angular por unidade de medida padrão. Por exemplo, quantos pulsos são necessários para que a máquina se movimente 1 milímetro. Com esta variável determinada é possível calcular rapidamente qual o número e frequência de pulsos gerados a partir de uma instrução.

4.2. Testes Preliminares

Com o controlador montado em bancada e com o sistema instalado nas configurações padrão sendo todo o sistema alimentado a partir da fonte USB da Raspberry. Foram gerados pulsos simultâneos nos eixos X, Y e Z, com a escala de 100pulsos/mm e utilizada uma instrução que ordenou o sistema a se deslocar por 100 mm em X e Y e 50 mm em Z gerando 10k pulsos em X e Y e 5k pulsos em Z. A velocidade de movimentação do eixo foi alterada para obter diferentes frequências de geração de pulsos com objetivo de obter qual a frequência máxima do sistema. Com este teste foi verificado que o sistema aparenta ter uma limitação de geração de pulsos em aproximadamente 13kHz, que pode ter sido causada por uma configuração no

software ou configuração e montagem de hardware. O resultado deste teste pode ser observado na figura 14.

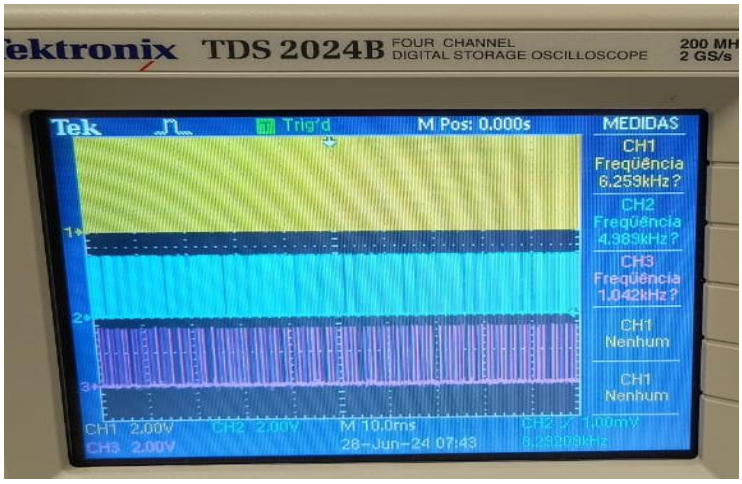
Figura 14 - Leitura de frequência via osciloscópio de frequências múltiplas



Fonte: Do Autor (2024)

Foram observados resultado semelhante quando utilizado apenas 1 eixo. Nesta condição em teoria o sistema teria a melhor performance, visto que se trata de pulsos com frequências iguais ou múltiplas, o que facilita o processamento. Optou-se por realizar um segundo teste com diferentes deslocamentos para que a geração de pulsos se torne mais aleatória. Os resultados em frequência máxima não sofreram alteração significativa. A figura 15 apresenta o resultado de um destes testes em que se pode observar diferentes frequências para cada eixo. Em ambos os casos após o teste com osciloscópio a ESP foi utilizada para realizar a contagem de pulsos, em todos os casos a contagem foi igual a quantidade esperada.

Figura 15 - Leitura de frequência via osciloscópio de frequências aleatórias



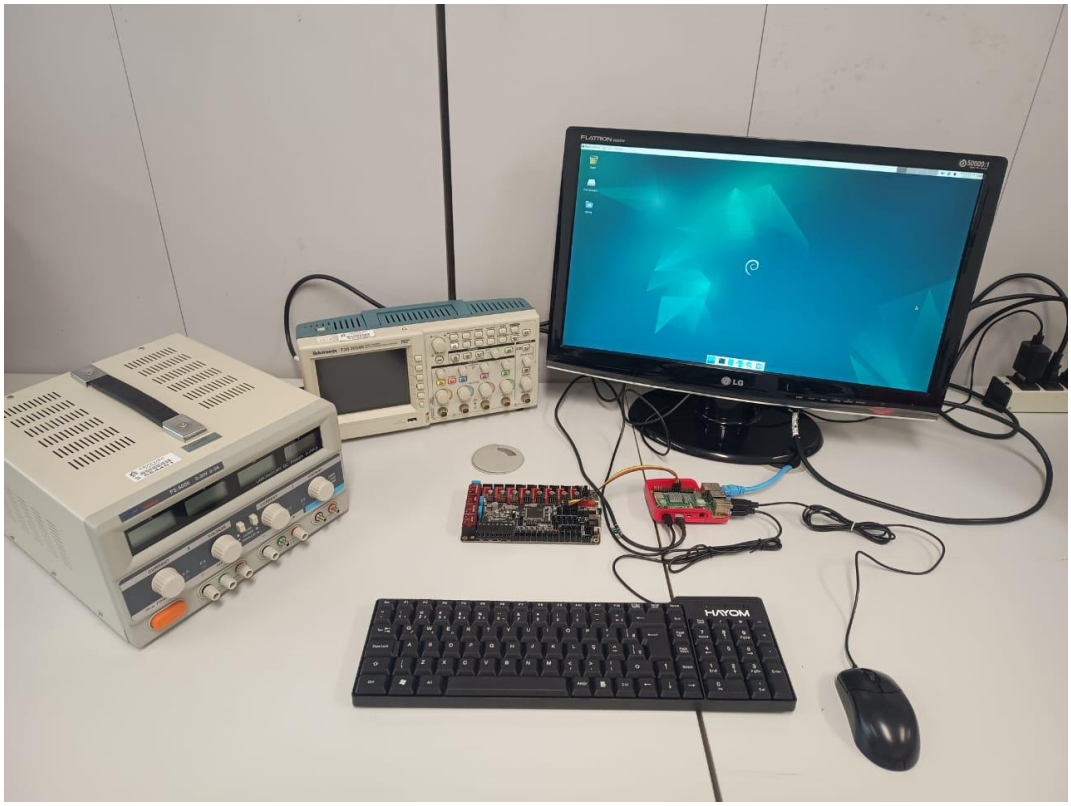
Fonte: Do Autor (2024)

4.3. Testes de Bancada

Após os testes preliminares feitos apenas com a placa controladora e a Raspberry PI, foram feitas algumas alterações a nível de hardware, como a utilização de uma fonte de bancada 24V para realizar a alimentação da placa controladora, e para a Raspberry Pi utilizou-se uma fonte de 5V/3A (15W) pois constatou-se que seu processamento se tornava limitado quando utilizado uma fonte inferior.

Com as melhorias e novos testes realizados seguindo os mesmos passos descritos em 4.2, o sistema registrou uma melhora na frequência dos pulsos gerados, saltando de 13kHz para 20kHz em movimentações com apenas um eixo e 18kHz em movimentações multi-eixo. Neste teste a ESP novamente confirmou a entrega de todos os pulsos esperados. A figura abaixo mostra o sistema montado em bancada acompanhado da fonte de bancada 24V e um osciloscópio para a aferição da frequência dos pulsos:

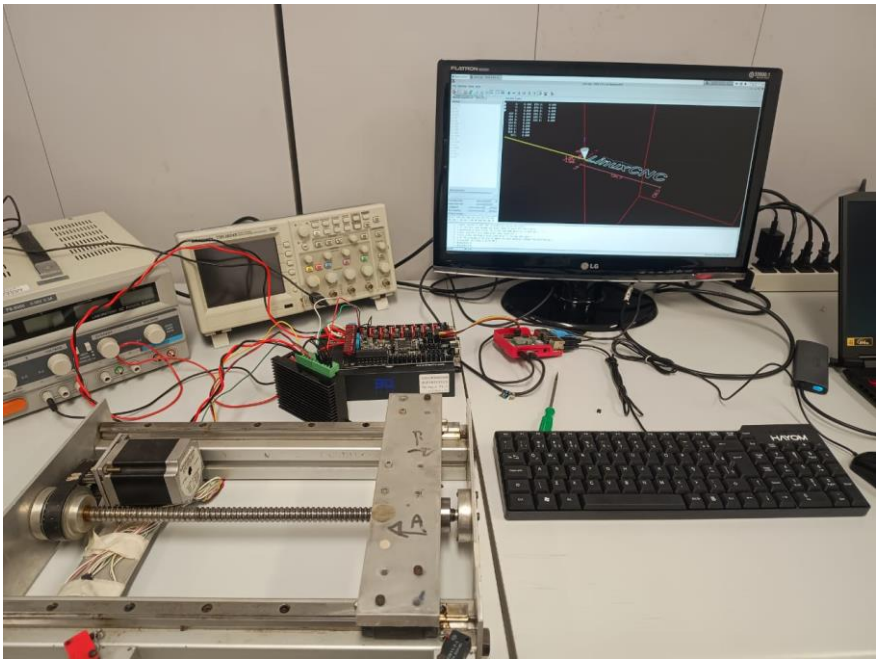
Figura 16 - Sistema completo em bancada



Fonte: Do Autor (2024)

Após a aferição dos pulsos e da quantidade que a placa controladora em conjunto com o remora poderia gerar, o sistema foi montado para validação de movimentação em motores de passo. O teste foi realizado utilizando uma guia de movimentação linear de apenas um eixo com o firmware configurado apenas para a movimentação linear, sem fazer a leitura de sensores de fim de curso. O driver utilizado para o teste foi o Driver Step/Dir modelo HY-DIV268N-5A comandando um motor de passo KTC-HT23-401 da Kalatec. A figura 17 mostra o sistema montado em bancada, conectado a guia linear:

Figura 17 - Sistema implementado em guia de movimentação linear



Fonte: Do Autor (2024)

Os testes foram feitos utilizando a movimentação em modo *jog*, que consiste na movimentação livre do eixo com apenas um comando, e utilizando comandos G-Code para enviar o eixo para diversas posições.

Constatou-se com o teste realizado acima que o sistema obteve uma movimentação satisfatória, com a velocidade de movimentação estando abaixo dos 2000 mm/min e com a escala de 80pulsos/mm escala corretamente calculada para o sistema mecânico em questão. Quando utilizado velocidades superiores a 2000 mm/min, o sistema é limitado automaticamente, porém sem o total desligamento ou travamento, por conta da limitação de geração de pulsos.

4.4. Limitações

Após os testes realizados em bancada, foram constatadas algumas limitações do sistema, sendo a principal a geração de pulsos, que ficou entre 13kHz e 20 kHz, se assemelhando a comunicação via porta paralela utilizando um PC desktop.

Também foram constatadas limitações com a alimentação via USB. A *Raspberry Pi* utiliza para a alimentação de energia a entrada USB-C 1.2, cujo a entrega de energia é limitada a 15W (5V/3A), enquanto a placa controladora *BigTreeTech Octopus* utiliza a entrada USB-C primariamente para a transferência de

dados e atualizações sem cartão SD. Ao utilizar a entrada USB aliado a uma fonte de baixa potência, o processador embarcado tanto na *Raspberry Pi* quanto na placa controladora não é capaz de atingir o seu potencial máximo de processamento, o que pode interferir na geração de pulsos para os drivers dos motores e causar gargalos e travamentos no *LinuxCNC*.

As limitações do próprio software também interferem no funcionamento do hardware pois o Remora, apesar de intuitivo, tem uma configuração que pode não ser tão trivial para iniciantes, sendo necessário o conhecimento prévio em programação, especialmente na linguagem *Python* e conhecimentos em modelos de arquivos JSON.

O *LinuxCNC* também requer um conhecimento básico em linguagem de programação pois algumas configurações do sistema como os arquivos HAL e INI por exemplo, pois seu código fonte tem como base a linguagem Python, enquanto outros módulos podem estar escritos em C/C++.

5. CONCLUSÃO

A validação proposta por este trabalho contribui para uma pesquisa de sistemas de movimentação com comando numérico computadorizado (CNC) de baixo custo a nível de laboratório/sala de aula, proporcionando mudanças na dinâmica de ensino para tais sistemas. Para atingir o objetivo proposto de desenvolver uma solução de controlador reprogramável e flexível baseado em sistema embarcado, para a utilização com máquinas CNC e Robôs didáticos foi proposta uma arquitetura composta por uma Raspberry Pi 4 em que o sistema *LinuxCNC* foi instalado e uma placa controladora de impressora 3D de 8 eixos rodando um *firmware* experimental denominado Remora.

Seguindo a proposta do desenvolvimento, foram encontradas limitações relacionadas a geração de pulsos, não sendo fator desqualificante do projeto uma vez que os resultados obtidos são semelhantes aqueles de um computador desktop com placa paralela. No entanto, era esperado um resultado superior ao apresentado, constatando-se que a alimentação de energia do sistema pode limitar o funcionamento a depender da forma que este é energizado.

O *firmware* empregado no projeto, o Remora, também possui suas limitações principalmente no que diz respeito a configurações e algumas funcionalidades que, devido a limitação de tempo por conta de fatores externos ao projeto, não foi possível estudá-las a fundo aplicando uma metodologia de testes mais direcionada a tais funcionalidades, sendo possível apenas validar o uso com três eixos através de um osciloscópio e apenas um eixo por vez com uso de motores reais.

Contudo, dado a proposta original do tema considera-se que o protótipo foi validado satisfatoriamente, atingindo os objetivos propostos e proporcionando uma alternativa de baixo custo a controladores CNC convencionais.

5.1. Melhorias Sugeridas

O tema desta monografia possui aberturas para melhorias e incrementos que podem deixar o protótipo ainda mais robusto e eficiente. Deixo como sugestão de melhorias futuras:

- Validação de movimentação com mais de 3 eixos (4º eixo, controle de spindle, entre outros)
- Construção de IHM usando tela própria, com interface para uma configuração mais rápida e precisa do sistema.
- Montagem mecânica do sistema em um gabinete portátil ou incorporado a uma máquina de bancada

6. REFERÊNCIAS BIBLIOGRÁFICAS

SOLUTIONS, Newfangled. *Mach3*. 2024. Disponível em: https://www.machsupport.com/software/mach3/. Acesso em: 20 jul. 2024.

REMORA. *Remora Documentation*. Disponível em: https://remora-docs.readthedocs.io/en/latest/index.html. Acesso em: 22 fev. 2024.

EMBARCADOS. *O que é firmware?* Disponível em: https://embarcados.com.br/o-que-e-firmware/. Acesso em: 28 jul. 2024.

TECHTARGET. *What is Firmware?* 2024. Escrito por Ben Lutkevich. Disponível em: https://www.techtarget.com/whatis/definition/firmware. Acesso em: 19 ago. 2024.

GITHUB. *Remora*. Disponível em: https://github.com/scottalford75/Remora. Acesso em: 21 mar. 2024.

ROBODK. *Difference between Robots and CNC Machines*. Disponível em: https://robodk.com/blog/difference-robots-cnc-machines/. Acesso em: 19 ago. 2024.

ASME. *Explore the Basics of USB-C and USB Power Delivery*. Disponível em: https://www.asme.org/topics-resources/content/explore-the-basics-of-usb-c-and-usb-power-delivery. Acesso em: 28 jul. 2024.

RASPBERRY PI. *Raspberry Pi 4 Model B Specifications*. Disponível em: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/. Acesso em: 19 ago. 2024.

3D LAB. *O que é Klipper e o que tem a oferecer para sua impressora 3D?* Disponível em: https://3dlab.com.br/o-que-e-klipper-e-o-que-tem-a-oferecer-para-sua-impressora-3d/. Acesso em: 19 ago. 2024.

KLIPPER 3D. *Klipper*. Disponível em: https://www.klipper3d.org/. Acesso em: 19 ago. 2024.

P3D. *Klipper vs Marlin, Which One to Choose?* 2024. Disponível em: https://p3d.mx/blogs/3d-printer-review/klipper-vs-marlin?srsltid=AfmBOoooIvYiiRmFMdjhCOhYha6018Yx5C5W0yBFPIiisKfUN-l7XjzT. Acesso em: 22 ago. 2024.

STALLINGS, William. *Arquitetura e organização de computadores*. 9. ed. São Paulo: Pearson, 2018. 840 p.

ASATO, Osvaldo Luís. *CNCs de arquitetura aberta na manufatura: análise e síntese*. 2000. 107 f. Dissertação (Mestrado em Engenharia Mecânica) – Universidade de São Paulo, São Carlos, 2000.

PRITSCHOW, Gunter; et al. *Open Controller Architecture - Past, Present and Future.* 2001, 50 f. Projeto de Pesquisa.

MARCICANO, João Paulo P. *Introdução ao Comando Numérico*. São Paulo: USP, S/d. Disponível em: http://sites.poli.usp.br/d/pmr2202/arquivos/aulas/cnc.pdf

MARLIN FIRMWARE. *What is Marlin?* 2024. Disponível em: https://marlinfw.org/docs/basics/introduction.html. Acesso em: 20 ago. 2024.

SILVEIRA, Cláudio Abilio da. *LinuxCNC: componentes, latência, arquivo INI e arquivo HAL*. Florianópolis, Sc: IFSC, 2024. 40 slides, color.

WEG. *Automação e Controle Industrial*. 2024. Disponível em: https://www.weg.net/catalog/weg/BR/pt/Automa%C3%A7%C3%A3o-e-Controle-Industrial/c/BR_WDC_IA. Acesso em: 21 ago. 2024.

SIEMENS. *Máquinas-Ferramenta*. 2024. Disponível em: https://www.siemens.com/br/pt/produtos/automacao/sistemas-automacao/cnc-sinumerik/automation-systems.html. Acesso em: 22 ago. 2024.

FANUC. *Sistemas CNC e Soluções*. 2024. Disponível em: https://www.fanucamerica.com/bra/fanuc-brazil/cnc-products. Acesso em: 22 ago. 2024.

BARROS, Edna; CAVALCANTE, Sérgio. *Introdução a Sistemas Embarcados*. Disponível em: https://cin.ufpe.br/~vba/periodos/8th/s.e/aulas/STP%20-%20Intro%20Sist%20Embarcados.pdf. Acesso em: 23 ago. 2024.

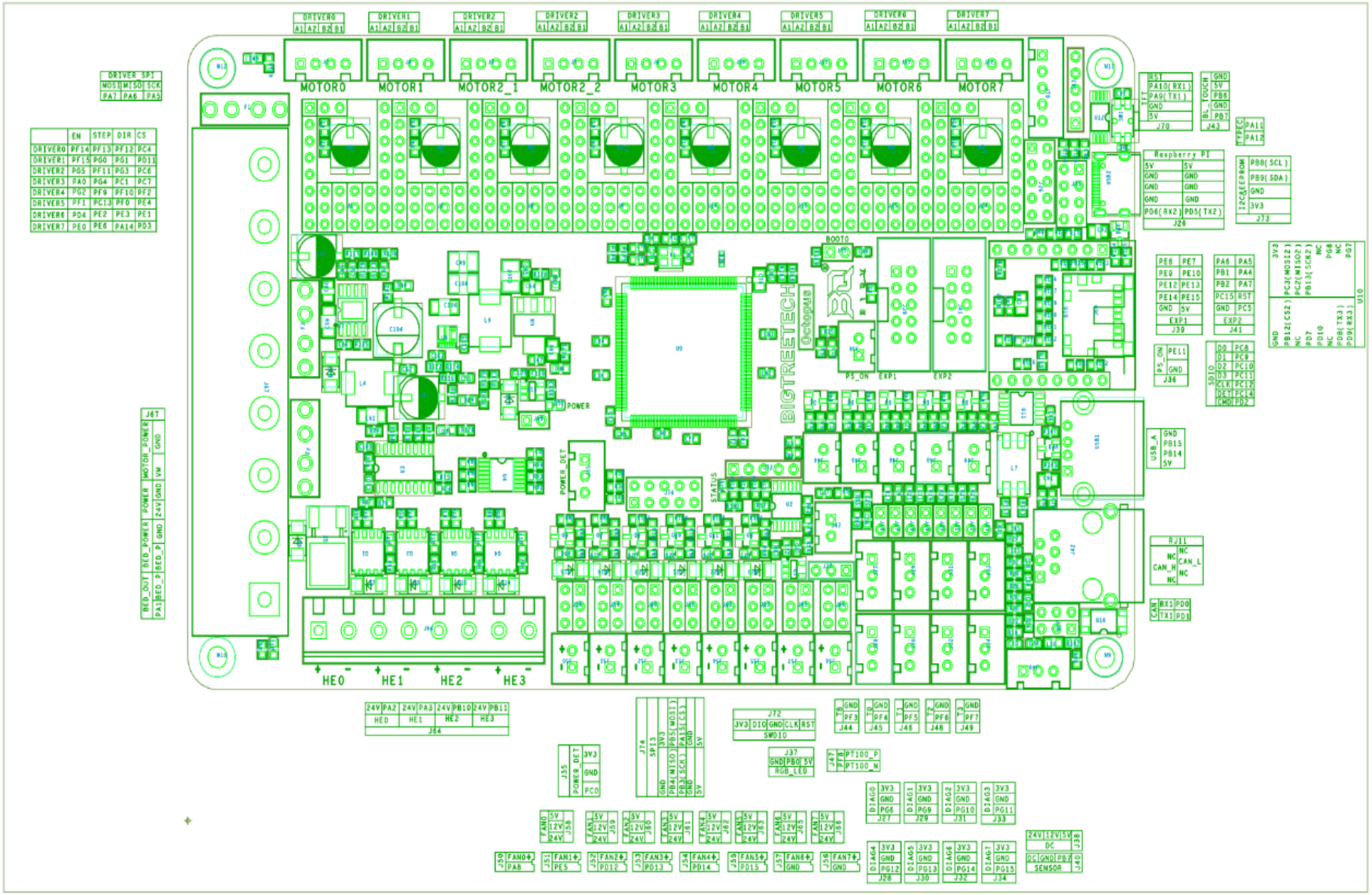
SILVEIRA, Cláudio Abilio da. *Controladores Baseados em PC: MACH3 e LinuxCNC*. Florianopolis, Sc: IFSC, 2024. 38 slides, color.

NABTO. *ESP32 for IoT: A Complete Guide*. 2024. Disponível em: https://www.nabto.com/guide-to-iot-esp-32. Acesso em: 24 ago. 2024.

HAAS AUTOMATION. *Desktop Machines*. Disponível em: https://www.haascnc.com/pt/machines/desktop-machines.html. Acesso em: 20 set. 2024.

ANEXOS

Anexo 1 – Diagrama de pinos da placa controladora BigTreeTech Octopus v1.1


DRIVER0
DRIVER1
DRIVER2
DRIVER2
DRIVER3
DRIVER4
DRIVER5
DRIVER6
DRIVER7
MOTOR0
MOTOR1
MOTOR2_1
MOTOR2_2
MOTOR3
MOTOR4
MOTOR5
MOTOR6
MOTOR7
DRIVER SPI
MOSI MISO SCK
PA7 PA6 PA5
EN STEP DIR CS
DRIVER0 PF14 PF13 PF12 PC4
DRIVER1 PF15 PG0 PG1 PD11
DRIVER2 PG5 PF11 PG3 PC6
DRIVER3 PA0 PG4 PC1 PC7
DRIVER4 PG2 PF9 PF10 PF2
DRIVER5 PF1 PC13 PF0 PE4
DRIVER6 PD4 PE2 PE3 PE1
DRIVER7 PE0 PE6 PA14 PD3
BIGTREETECH
Octopus
BOOT0
POWER
POWER_DET
STATUS
PS_ON
EXP1
EXP2
HE0
HE1
HE2
HE3
J67
BED_OUT BED_POWER POWER MOTOR_POWER
PA1 BED_P GND 24V GND VM GND
J64
24V PA2 24V PA3 24V PB10 24V PB11
HE0 HE1 HE2 HE3
J74
SPI3 3V3 PB5(MOSI) PB4(MISO) PB3(SCK) PA15(CS) GND 5V
J72
3V3 DIO GND CLK RST
SWDIO
J37
GND PB0 5V
RGB_LED
J35
POWER_DET 3V3 GND PC0
J47
PT100_P PT100_N
J44 PF3
J45 PF4
J46 PF5
J48 PF6
J49 PF7
FAN0 5V 12V 24V J58
FAN1 5V 12V 24V J59
FAN2 5V 12V 24V J60
FAN3 5V 12V 24V J61
FAN4 5V 12V 24V J62
FAN5 5V 12V 24V J63
FAN6 5V 12V 24V J65
FAN7 5V 12V 24V J66
J50 FAN0 PA8
J51 FAN1 PE5
J52 FAN2 PD12
J53 FAN3 PD13
J54 FAN4 PD14
J55 FAN5 PD15
J57 FAN6 GND
J56 FAN7 GND
DIAG0 3V3 GND PG6 J27
DIAG1 3V3 GND PG9 J29
DIAG2 3V3 GND PG10 J31
DIAG3 3V3 GND PG11 J33
DIAG4 3V3 GND PG12 J28
DIAG5 3V3 GND PG13 J30
DIAG6 3V3 GND PG14 J32
DIAG7 3V3 GND PG15 J34
24V 12V 5V DC
DC GND PB2 SENSOR J40
J70
TFT
5V PA10(RX1) PA9(TX1) GND 5V
J43
BL_TOUCH
GND 5V PB6 GND PB7
TYPEC
PA11 PA12
Raspberry PI
5V 5V
GND GND
GND GND
GND GND
PD6(RX2) PD5(TX2)
J26
I2C&EEPROM
PB8(SCL) PB9(SDA) GND 3V3
J73
PE0 PE7
PE0 PE10
PE12 PE13
PE14 PE15
GND 5V
EXP1
J39
PA6 PA5
PB1 PA4
PB2 PA7
PC15 RST
GND PC5
EXP2
J41
3V3
PC3(MOSI2)
PC2(MISO2)
PB13(SCK2)
GND
PB12(CS2)
NC
PD7
PD10
PD8(TX3)
PD9(RX3)
NC
PG8
NC
PG7
U10
PS_ON PE1
GND
J36
SDIO
D0 PC8
D1 PC9
D2 PC10
D3 PC11
CLK PC12
CMD PD2
USB_A
GND
PB15
PB14
5V
RJ11
NC NC
CAN_H CAN_L
NC NC
CAN
RX1 PD0
TX1 PD1

Anexo 2 - Código de Configuração da Placa Controladora - Movimentação X, Y e Z:

```
{
	"Board": "BIGTREETECH OCTOPUS",
	"Modules":[
	{
	"Thread": "Servo",
	"Type": "Reset Pin",
		"Comment":			"Reset pin",
		"Pin":				"PC_15"
	},
	{
	"Thread": "Base",
	"Type": "Stepgen",
		"Comment":			"X - Joint 0 step generator",
		"Joint Number":			0,
		"Step Pin":		"PF_13",
		"Direction Pin":	"PF_12",
		"Enable Pin":		"PF_14"
	},
	{
	"Thread": "Base",
	"Type": "Stepgen",
		"Comment":			"Y - Joint 1 step generator",
		"Joint Number":			1,
		"Step Pin":		"PG_0",
		"Direction Pin":	"PG_1",
		"Enable Pin":		"PF_15"
```

```
        },
        {
        "Thread": "Base",
        "Type": "Stepgen",
                "Comment":                  "Z - Joint 2 step generator",
                "Joint Number":             2,
                "Step Pin":           "PF_11",
                "Direction Pin":      "PG_3",
                "Enable Pin":         "PG_5"
        }
        ]
}
```

Anexo 3 - Código de Configuração da Placa Controladora - Sistema Ender 3:

```
{
	"Board": "BIGTREETECH OCTOPUS",
	"Modules":[
	{
	"Thread": "Servo",
	"Type": "Reset Pin",
		"Comment":			"Reset pin",
		"Pin":				"PC_15"
	},
	{
	"Thread": "Base",
	"Type": "Stepgen",
		"Comment":			"X DRIVER0 - Joint 0 step generator",
		"Joint Number":			0,
		"Step Pin":		"PF_13",
		"Direction Pin":	"PF_12",
		"Enable Pin":		"PF_14"
	},
	{
	"Thread": "Base",
	"Type": "Stepgen",
		"Comment":			"Y DRIVER1 - Joint 1 step generator",
		"Joint Number":			1,
		"Step Pin":		"PG_0",
		"Direction Pin":	"PG_1",
		"Enable Pin":		"PF_15"
	},
	{
	"Thread": "Base",
	"Type": "Stepgen",
		"Comment":			"Z DRIVER2 - Joint 2 step generator",
		"Joint Number":			2,
		"Step Pin":		"PF_11",
		"Direction Pin":	"PG_3",
		"Enable Pin":		"PG_5"
	},
	{
	"Thread": "Base",
	"Type": "Stepgen",
		"Comment":			"E0 DRIVER3 - Joint 3 step generator",
```

```
        "Joint Number":             3,
        "Step Pin":          "PG_4",
        "Direction Pin":     "PC_1",
        "Enable Pin":        "PA_0"
},
{
"Thread": "Servo",
"Type": "Digital Pin",
        "Comment":                  "X min DIAG0",
        "Pin":                      "PG_6",
        "Mode":                     "Input",
        "Data Bit":                 0
},
{
"Thread": "Servo",
"Type": "Digital Pin",
        "Comment":                  "X max DIAG4",
        "Pin":                      "PG_12",
        "Mode":                     "Input",
        "Data Bit":                 1
},
{
"Thread": "Servo",
"Type": "Digital Pin",
        "Comment":                  "Y min DIAG1",
        "Pin":                      "PG_9",
        "Mode":                     "Input",
        "Data Bit":                 2
},
{
"Thread": "Servo",
"Type": "Digital Pin",
        "Comment":                  "Y max DIAG5",
        "Pin":                      "PG_13",
        "Mode":                     "Input",
        "Data Bit":                 3
},
{
"Thread": "Servo",
"Type": "Digital Pin",
        "Comment":                  "Z min DIAG2",
```

```
        "Pin":                      "PG_10",
        "Mode":                     "Input",
        "Data Bit":                 4
},
{
"Thread": "Servo",
"Type": "Digital Pin",
        "Comment":                  "Z max DIAG6",
        "Pin":                      "PG_14",
        "Mode":                     "Input",
        "Data Bit":                 5
},
{
"Thread": "Servo",
"Type": "Temperature",
        "Comment":                  "Heated Bed tenperature sensor",
        "PV[i]":                    0,
        "Sensor":                   "Thermistor",
                "Thermistor":
                {
                        "Pin":      "PF_3",
                        "beta":     3990,
                        "r0":       100000,
                        "t0":       25
                }
},
{
"Thread": "Servo",
"Type": "PWM",
        "Comment":                  "Bed heater PWM",
        "SP[i]":                    0,
        "PWM Pin":                  "PA_1"
},
{
"Thread": "Servo",
"Type": "Temperature",
        "Comment":                  "Ext 0 temperature sensor",
        "PV[i]":                    1,
        "Sensor":                   "Thermistor",
                "Thermistor":
                {
```

```
                        "Pin":          "PF_4",
                        "beta":         3990,
                        "r0":           100000,
                        "t0":           25
                }
        },
        {
        "Thread": "Servo",
        "Type": "PWM",
                "Comment":                  "Ext0 heater PWM",
                "SP[i]":                    1,
                "PWM Pin":                  "PA_2"
        },
        {
        "Thread": "Servo",
        "Type": "PWM",
                "Comment":                  "Ext0 part cooling fan PWM FAN0",
                "SP[i]":                    2,
                "PWM Max":                  128,
                "PWM Pin":                  "PA_8"
        },
        {
        "Thread": "Base",
        "Type": "RCServo",
                "Comment":                  "RC servo for probe bltouch according to marlin",
                "SP[i]":                    3,
                "Servo Pin":          "PB_6"
        }
        ]
}
```

Anexo 4 – Código para contagem de pulsos

```
/*

    Step counter for BTT Octopus


    Developer: Mauricio Neves Junior

    Date: 2024/05/02

    Version: v1.15


*/


/* LIBRARIES AND DEFINITIONS */

#include <Arduino.h>


/* VARIABLES */

// Defina os pinos conectados aos pinos "step" dos drivers dos motores de passo

const int pulsePinX = D1; // Pino para o eixo X

const int pulsePinY = D2; // Pino para o eixo Y

const int pulsePinZ = D5; // Pino para o eixo Z

const int dirX = D6;

const int dirY = D7;

const int dirZ = D8;


const int led = D4;


// Variáveis para contar os pulsos

volatile unsigned long pulseCountX = 0;

volatile unsigned long pulseCountY = 0;
```

```
volatile unsigned long pulseCountZ = 0;
volatile unsigned long lastPulseCountX = 0;
volatile unsigned long lastPulseCountY = 0;
volatile unsigned long lastPulseCountZ = 0;
float frequencyX = 0.0;
float frequencyY = 0.0;
float frequencyZ = 0.0;

// Variáveis para armazenar os estados dos pinos de direção
bool currentDirX = LOW;
bool currentDirY = LOW;
bool currentDirZ = LOW;
bool lastDirX = LOW;
bool lastDirY = LOW;
bool lastDirZ = LOW;

/* FUNCTIONS */
// Funções de interrupção na borda de subida
void IRAM_ATTR handlePulseRisingX() {
  pulseCountX++;
}

void IRAM_ATTR handlePulseRisingY() {
  pulseCountY++;
}

void IRAM_ATTR handlePulseRisingZ() {
```

```
  pulseCountZ++;
}

// Funções de interrupção na borda de descida
void IRAM_ATTR handlePulseFallingX() {
  pulseCountX = 0;
}

void IRAM_ATTR handlePulseFallingY() {
  pulseCountY = 0;
}

void IRAM_ATTR handlePulseFallingZ() {
  pulseCountZ = 0;
}

void setup() {
  // Inicializa a serial para monitoramento
  Serial.begin(115200);

  // Configura os pinos como entrada
  pinMode(pulsePinX, INPUT);
  pinMode(pulsePinY, INPUT);
  pinMode(pulsePinZ, INPUT);
  pinMode(dirX, INPUT);
  pinMode(dirY, INPUT);
  pinMode(dirZ, INPUT);
```

```
  pinMode(led, OUTPUT);

  // Anexa as funções de interrupção aos pinos
  attachInterrupt(digitalPinToInterrupt(pulsePinX), handlePulseRisingX, RISING);
  attachInterrupt(digitalPinToInterrupt(pulsePinY), handlePulseRisingY, RISING);
  attachInterrupt(digitalPinToInterrupt(pulsePinZ), handlePulseRisingZ, RISING);
  // Imprime mensagem inicial
  Serial.println("Contador de Pulsos Iniciado");
}

void loop() {
  static unsigned long lastPrintTime = 0;
  unsigned long currentTime = millis();

  // Calcular a frequência a cada segundo
  if (currentTime - lastPrintTime >= 1000) {
    // Calcular a frequência para o eixo X
    frequencyX = (pulseCountX - lastPulseCountX) / 1.0; // Pulsos por segundo
    lastPulseCountX = pulseCountX;

    // Calcular a frequência para o eixo Y
    frequencyY = (pulseCountY - lastPulseCountY) / 1.0; // Pulsos por segundo
    lastPulseCountY = pulseCountY;

    // Calcular a frequência para o eixo Z
    frequencyZ = (pulseCountZ - lastPulseCountZ) / 1.0; // Pulsos por segundo
    lastPulseCountZ = pulseCountZ;
```

```
// Imprimir os valores dos contadores de pulsos e as frequências
Serial.print("Eixo X: ");
Serial.println(pulseCountX);
Serial.print("Frequencia X: ");
Serial.print(frequencyX);
Serial.println(" Hz");

Serial.print("Direcao X: ");
if (pulseCountX == 0) {
  Serial.println("Parado");
} else if (currentDirX != lastDirX) {
  Serial.println(currentDirX ? "Direita" : "Esquerda");
  lastDirX = currentDirX;
}

Serial.print("\nEixo Y: ");
Serial.println(pulseCountY);
Serial.print("Frequencia Y: ");
Serial.print(frequencyY);
Serial.println(" Hz");

Serial.print("Direcao Y: ");
if (pulseCountY == 0) {
  Serial.println("Parado");
} else if (currentDirY != lastDirY) {
  Serial.println(currentDirY ? "Direita" : "Esquerda");
```

```
      lastDirY = currentDirY;
    }

    Serial.print("\nEixo Z: ");
    Serial.println(pulseCountZ);
    Serial.print("Frequencia Z: ");
    Serial.print(frequencyZ);
    Serial.println(" Hz");

    Serial.print("Direcao Z: ");
    if (pulseCountZ == 0) {
      Serial.println("Parado");
    } else if (currentDirZ != lastDirZ) {
      Serial.println(currentDirZ ? "Direita" : "Esquerda");
      lastDirZ = currentDirZ;
    }

    Serial.println("\n");
    lastPrintTime = currentTime;
  }
}
```

Anexo 5 – Código para leitura e gravação de quantidade de pulsos em planilha

```
import serial
import csv

# Configurar a porta serial (ajuste a porta e a taxa de transmissão conforme necessário)
ser = serial.Serial('COM3', 115200)

# Abrir um arquivo CSV para escrita
with open('dados.csv', mode='w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(['Contagem de Pulsos por Eixo'])  # Cabeçalho da coluna

    try:
        while True:
            if ser.in_waiting > 0:
                line = ser.readline().decode('utf-8').rstrip()
                print(line)
                writer.writerow([line])
    except KeyboardInterrupt:
        print("Interrompido pelo usuário")
    finally:
        ser.close()
```

APÊNDICES

Apêndice A – Manual de Montagem e Configuração

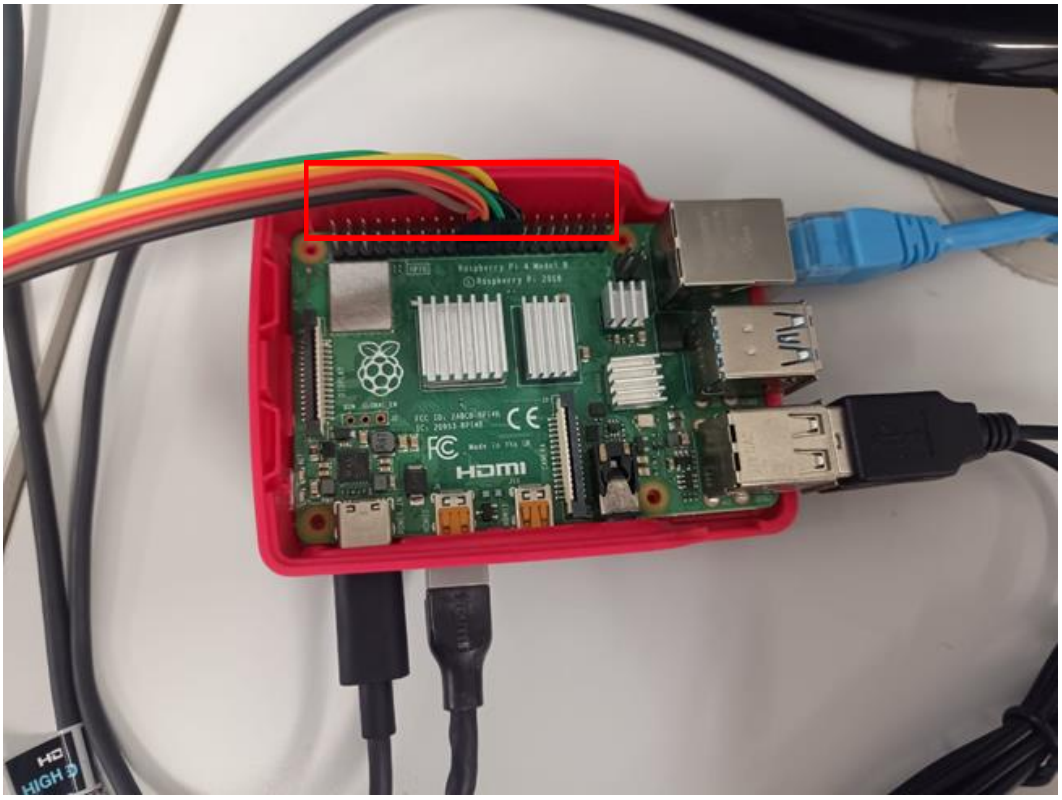
OBS: Caso deseje utilizar os mesmos arquivos apresentados nesta monografia, acesse https://github.com/mauricionj10/CNC_Controller_RPi_Remora

1. HARDWARE

1.1. Montagem do Hardware

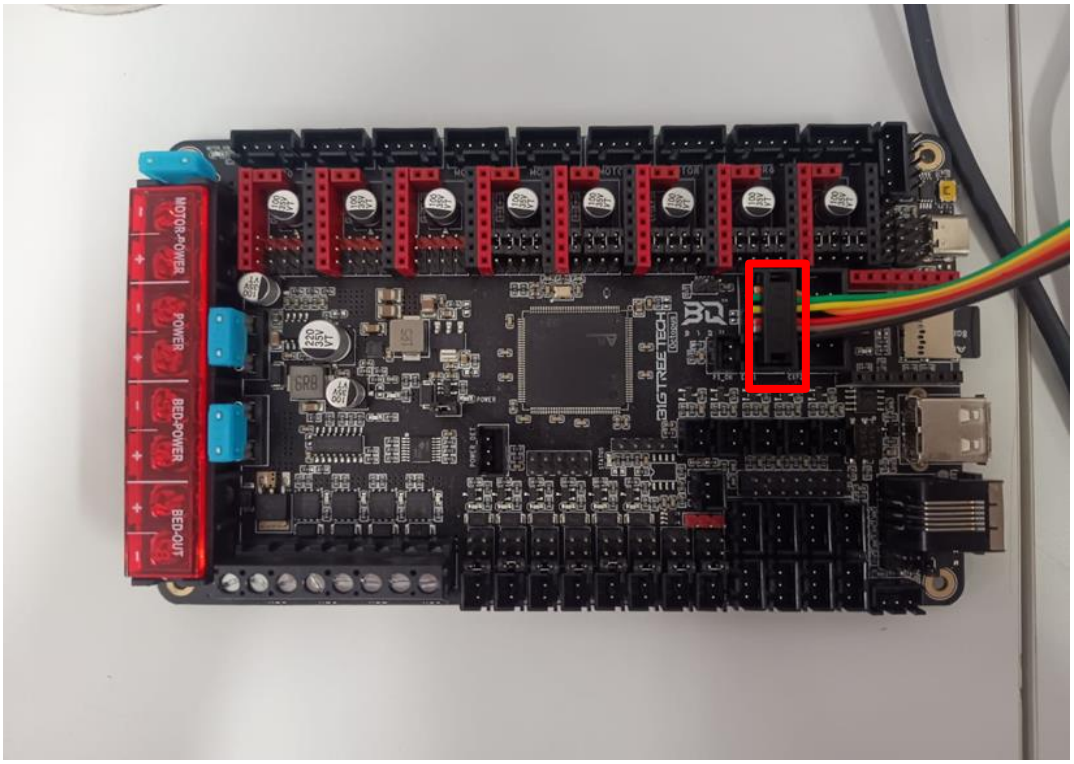
1.1.1. Retire a placa controladora e a Raspberry Pi da embalagem, caso haja.

1.1.2. Caso utilize uma Raspberry PI com case, remova a parte superior do case, expondo os pinos GPIO, conforme mostra a imagem abaixo:



1.1.3. Para realizar a conexão do GPIO com a placa controladora corretamente, é necessário realizar a montagem de um cabo de 10 pinos, que será conectado ao pino EXT1 da placa

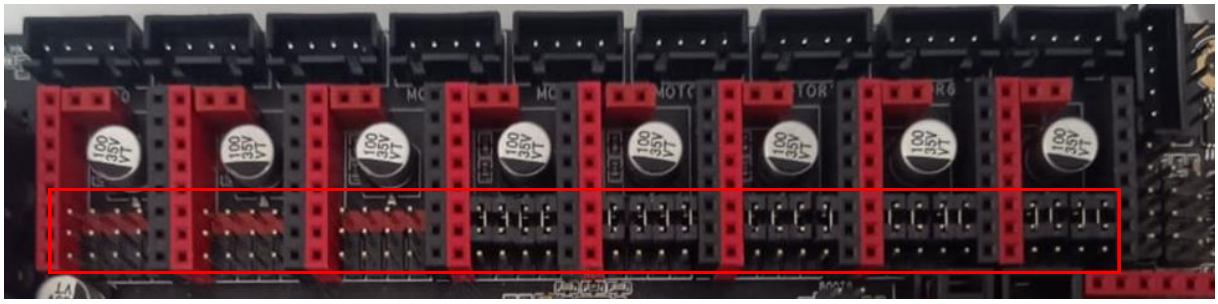
controladora, enquanto o outro lado do cabo irá nos pinos GPIO 19, 21, 22, 23 e 24 da Raspberry. Como alternativa, podem ser montados cabos convencionais para conexão entre pinos GPIO, porém deve-se atentar a correta ligação dos pinos, bem como estar ciente que podem ocorrer instabilidades por conta de uma má conexão ou por conta da qualidade dos fios.



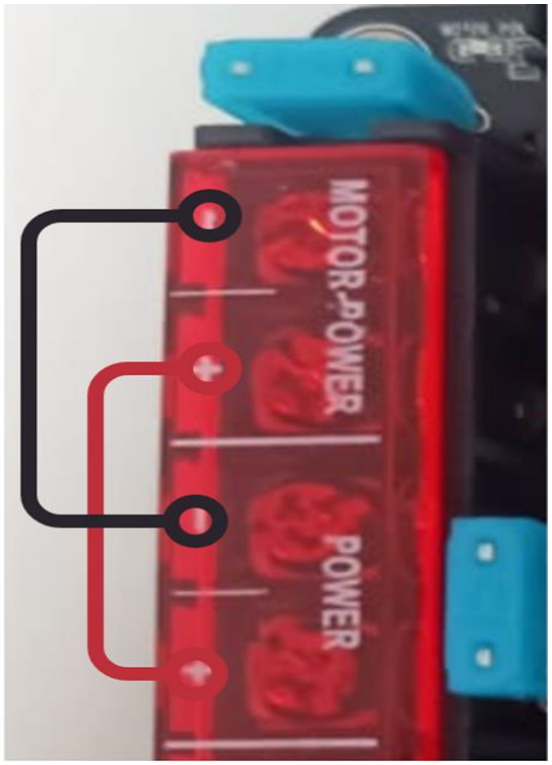
1.2. Configuração do Hardware

1.2.1.

Após conectar as placas, faça a configuração do modo de operação dos drivers conectados à placa, alterando os jumpers próximo aos conectores, conforme mostra a figura abaixo. Cada configuração de pinos corresponde a um modo de operação do driver, que são UART, SPI e Step/Dir, portanto para configurar corretamente, basta posicionar os jumpers seguindo a imagem exibida no item 3.7 da monografia.



1.2.2. Após definir o modo de operação e conexão da placa controladora, conecte a alimentação 24V aos bornes de alimentação posicionados ao extremo esquerdo da placa. Para acionar apenas a placa conecte a alimentação nos pinos correspondentes ao POWER. Para a alimentação dos motores ou dos drivers, faça uma ponte entre o pino POWER e o MOTOR-POWER, conforme mostra a imagem abaixo.



1.2.3. Conecte a Raspberry PI a uma fonte de alimentação através de um cabo USB-C. A fonte ideal para operação da raspberry é de 5V/3A (15W)

2. Firmware

2.1. **Configuração do Firmware**

2.1.1. No site do Remora, clique em *Firmware.* (https://github.com/scottalford75/Remora/tree/main)

2.1.2. Na tela seguinte, clique em *FirmwareBin*

2.1.3. Na tela seguinte, selecione a pasta correspondente ao modelo do microcontrolador presente na placa controladora. Para este projeto, foi utilizado o modelo STM32F446.

2.1.4. Faça o download do arquivo *Firmware.bin*

2.1.5. Para configurar o firmware corretamente, é necessário a criação de um arquivo em .txt chamado *config.txt.* No site citado no item 2.1.1, é possível encontrar configurações modelo que podem ser adaptadas para a sua necessidade, basta entrar em Firmware -> ConfigSamples e escolher a pasta correspondente a sua placa controladora.

2.1.6. No arquivo **Config.txt**, monte a configuração do firmware conforme os requisitos para o seu projeto, determinando quantidade de eixos e suas respectivas pinagens, ativação de controles PWM, entre outros.

2.2. **Instalação do Firmware**

2.2.1. Formate um cartão de memória microSD em formato FAT32. Após isso, copie e cole os arquivos firmware.bin e config.txt diretamente no diretório raiz do cartão de memória.

2.2.2. Insira o cartão de memória no slot de cartão da placa controladora e energize a placa. O controlador fará a leitura dos dados, indicando o status através de um led embutido na placa.

2.2.3. Após o led estabilizar aceso, desligue a placa da energia e remova o cartão SD da placa. Insira o cartão no computador e verifique se o arquivo firmware.bin foi convertido para

firmware.cue, indicando que o firmware foi corretamente instalado.

2.2.4. Para fazer customizações no firmware, basta editar o arquivo config.txt conforme descrito no passo 1 deste manual.

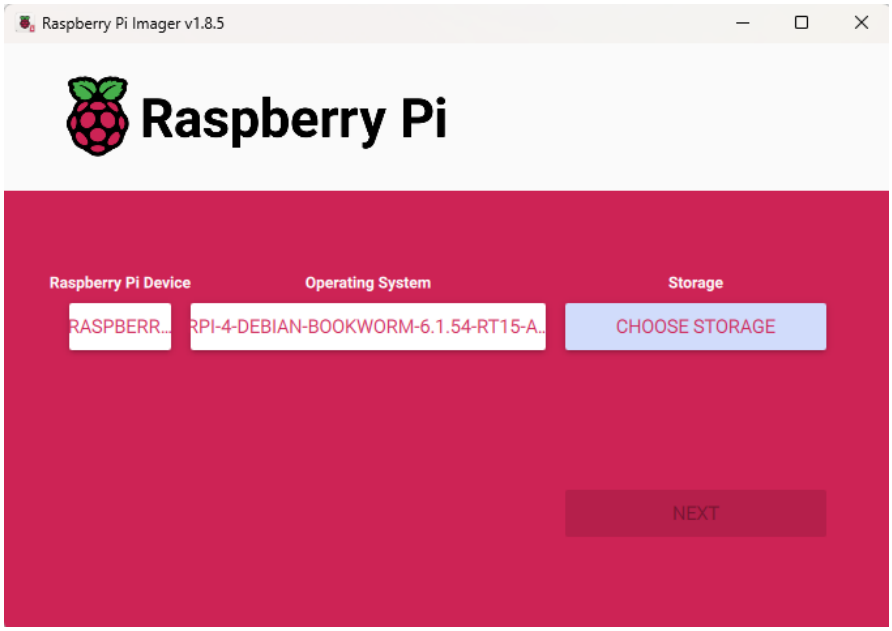
3. SOFTWARE (LINUXCNC)

3.1. Instalação do LinuxCNC

3.1.1. No site do linuxcnc (https://linuxcnc.org/downloads/), faça o download da versão específica para Raspberry Pi na qual está utilizando. Sempre faça o download da versão mais recente do sistema.

- LinuxCNC 2.9.2 Raspberry Pi 4 OS based on Debian Bookworm Raspberry Pi 4 Uspace compatible with Mesa Ethernet and SPI interface boards.
- LinuxCNC 2.9.2 Raspberry Pi 5 OS based on Debian Bookworm Raspberry Pi 5 Uspace compatible with Mesa Ethernet boards. Note that SPI is not currently supported with the Pi5.

3.1.2. Utilize o software de sua preferência para criar uma imagem do linuxcnc em um cartão de memória microSD (Tamanho mínimo de 8GB). Para este projeto, foi utilizado o Raspberry PI Imager por ser o instalador próprio para Raspberry Pi, eliminando possíveis problemas de compatibilidade. Caso deseje utilizar, o download pode ser feito em https://www.raspberrypi.com/software/



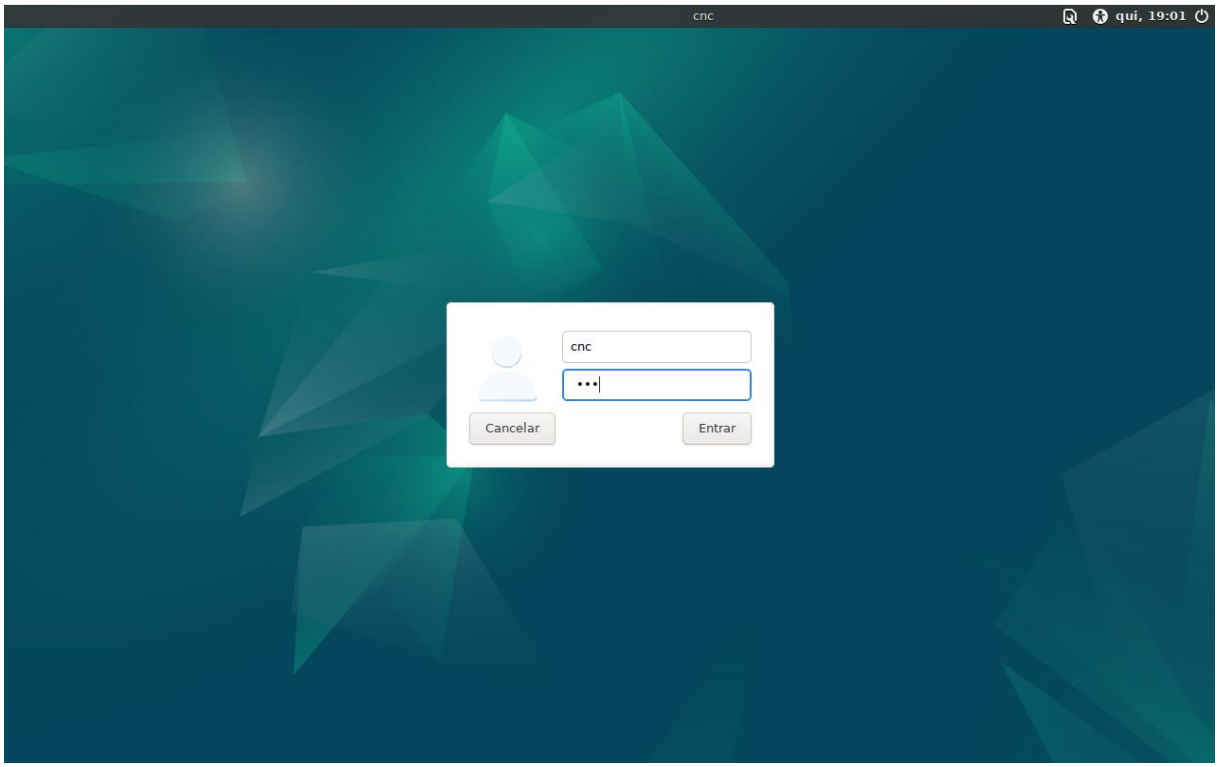
3.1.3. Após criado a imagem, insira o cartão microSD no slot da Raspberry PI e conecte-a na energia. Conecte a raspberry pi em um monitor ou IHM de sua preferência e aguarde o sistema realizar o primeiro boot. A placa é compatível com periféricos como mouse e teclado USB.

3.2. **Configuração do LinuxCNC**

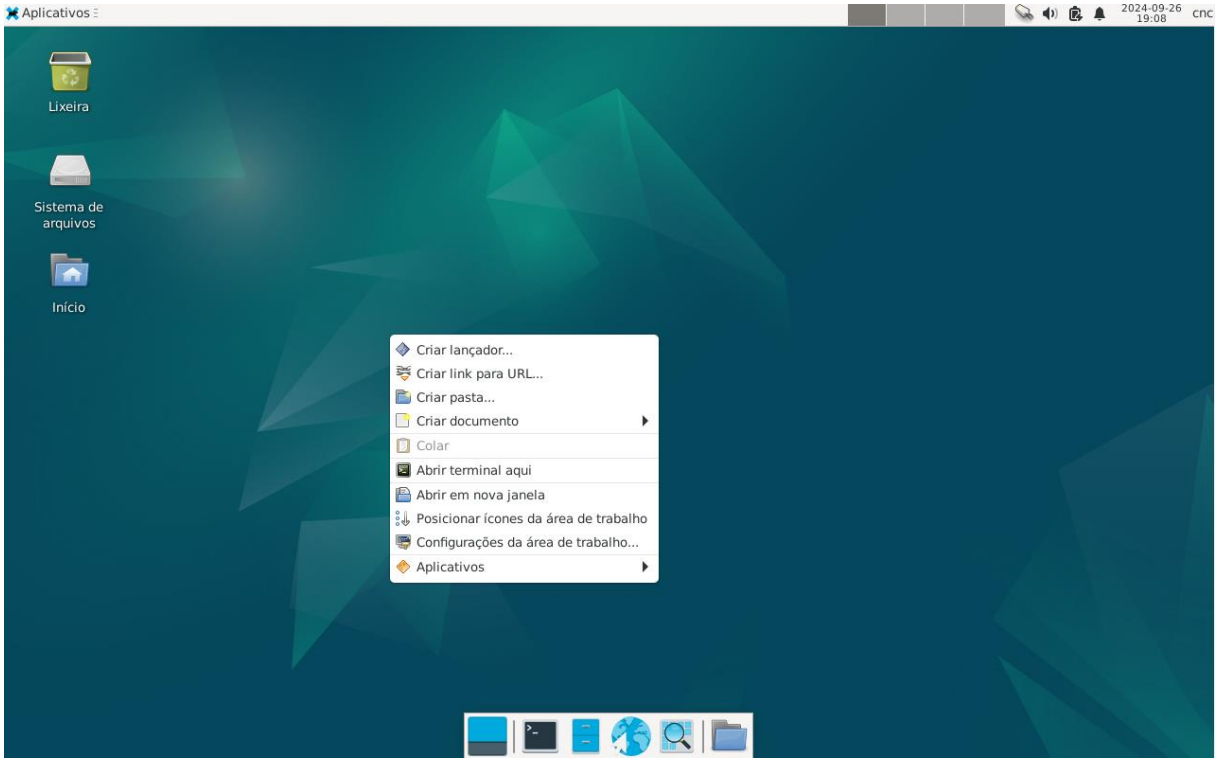
3.2.1. Após a instalação do sistema, na tela de início, basta inserir o usuário e senha padrão, descritos abaixo. Poderá ser modificado após a instalação.

usuário: cnc

senha: cnc



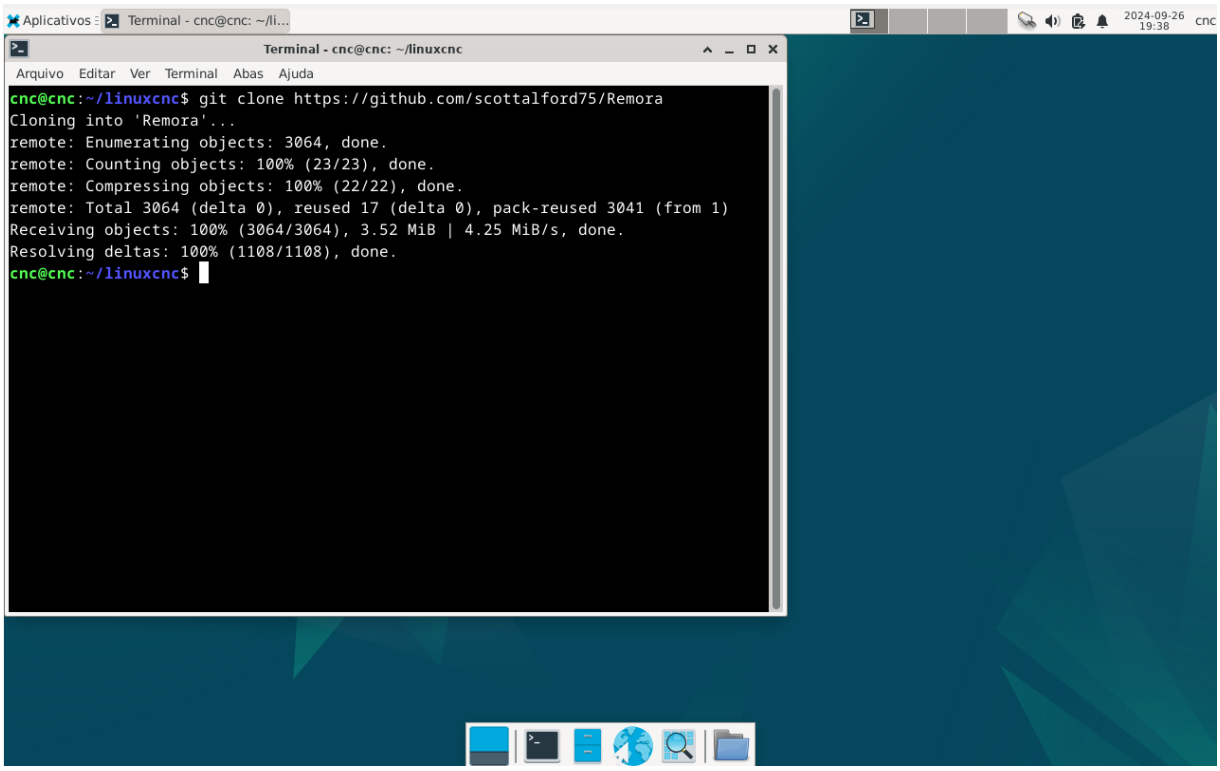
3.2.2. Após a abertura da tela inicial, abra um terminal apertando o atalho *Ctrl* + *Alt* + *T* ou clicando com o botão direito em qualquer lugar da tela e selecionando a opção "*Open Terminal Here"*



3.2.3. No terminal digite os comandos a seguir para instalar o componente do Remora no LinuxCNC:

Obs: Caso o comando git não esteja instalado, digite no terminal sudo apt-get install git. Necessário estar conectado a internet.

- Mkdir ~/linuxcnc
- cd ~/linuxcnc
- git clone https://github.com/scottalford75/Remora
- cd Remora/LinuxCNC/Components/Remora
- sudo halcompile --install remora.c



3.2.4. Para uso com a configuração Ender 3, é necessário a instalação dos componentes PRUencoder e PIDcontroller. Para isso basta digitar os seguintes comandos:

- cd Remora/LinuxCNC/Components/PRUencoder
- sudo halcompile --install PRUencoder.c
- cd Remora/LinuxCNC/Components/PRUencoder
- sudo halcompile --install PIDcontroller.c