UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA MECATRÔNICA

JOÃO VICTOR FABRI

SISTEMA DE COMANDO ALTERNATIVO PARA MANIPULADOR CARTESIANO DE TOCHA DE SOLDAGEM

Joinville
2020

JOÃO VICTOR FABRI

SISTEMA DE COMANDO ALTERNATIVO PARA MANIPULADOR CARTESIANO DE TOCHA DE SOLDAGEM

Trabalho de conclusão de Curso apresentado como requisito parcial para obtenção do título de bacharel em Engenharia Mecatrônica no curso de Engenharia Mecatrônica da Universidade Federal de Santa Catarina, Centro Tecnológico de Joinville.

Orientador: Prof. Dr. Pablo Andretta Jaskowiak

Joinville
2020

AGRADECIMENTOS

Agradeço aos meus pais, que me ofereceram suporte financeiro e emocional durante toda minha jornada acadêmica.

Ao meu avô, que criou em mim a curiosidade e o espirito de sempre buscar conhecimento.

Às minhas avós, meus tios e meus irmãos, que estiveram em meus pensamentos durante todo o tempo que estive longe de casa.

Aos meus amigos, que me ajudaram nos momentos difíceis, compartilharam dos momentos felizes e se tornaram a minha segunda família.

Ao prof. Pablo Andretta Jaskowiak, que acreditou no meu potencial, me desafiou a superar os meus limites e me ensinou a ter confiança, coragem e determinação.

À prof.ª Tatiana Renata Garcia, que me fez mais humano, mais responsável e despertou minha paixão pelo ensino.

Por fim, agradeço ao Laboratório de Tecnologias da Soldagem, em especial o prof. Tiago Vieira da Cunha, que me dedicou seu tempo, ofereceu seus equipamentos e me ensinou muito sobre o quê é ser um engenheiro.

RESUMO

Procedimentos de soldagem são utilizados em diversas áreas da engenharia e assumem características peculiares à cada uma de suas aplicações. No que tange o processo de soldagem de objetos metálicos com a utilização de tochas de solda, o manuseio destes equipamentos pode ser nocivo aos seus operadores, visto que as tochas trabalham em altas temperaturas. Outro problema é, também, a repetitividade do processo, que gera problemas tanto à saúde dos operadores quanto à qualidade do produto. A automação é uma alternativa indispensável na execução destes procedimentos neste caso. Com a automação dos processos de soldagem é possível desenvolver estudos mais detalhados no comportamento da junção metálica e caracterizar uma série de variáveis de soldagem, como o tipo de tecimento, a velocidade e a energia de soldagem, em comparativo com a qualidade final da solda, uma vez que a automatização oferece maior reprodutibilidade dos processos e confiabilidade das características desejadas. Neste trabalho propõe-se a readaptação de um sistema de soldagem com dois eixos prismáticos ortogonais através do acoplamento de um sistema micro-controlado próprio, que possibilitará o desenvolvimento de uma interface de facilitação da execução dos programas de soldagem pelo usuário. Isso se faz necessário tendo em vista as limitações iniciais da interface de programação da máquina, que possui uma curva íngreme de aprendizado e uma interface de comando especializada. O sistema obtido como resultado deste trabalho permite que tanto pesquisadores quanto operários da área de soldagem que utilizam este equipamento tenham seu trabalho facilitado na programação de processos de soldagem. Há ainda possibilidades futuras de adaptação do sistema e equipamento para novas técnicas de tecimento de solda e novas funcionalidades.

Palavras-chave: Automação de processos. Readaptação. Manipulador cartesiano.

ABSTRACT

Welding procedures are used in several areas of engineering and assume peculiar characteristics to each of its applications. Regarding the process of welding metal plates with the use of welding torches, the handling of these equipments can be harmful to its operators, since the torches work at the melting temperature of the metal alloy. Another problem is the repetition of the process, which introduces problems both for the health of the operator and the quality of the product. Automation is an indispensable alternative in performing these procedures in this case. With the automation of the welding processes, it is possible to develop more detailed studies on the behavior of the metallic junction and to characterize a series of welding variables, such as the type of weave, the speed and the welding energy, in comparison with the final quality of the weld, since automation offers greater reproducibility of the processes and reliability of the desired characteristics. This work proposes the retrofitting of a welding system with two orthogonal prismatic axes through the coupling of its own micro-controlled system with a new one, which will enable the development of an interface to facilitate the execution of welding programs by the user. This is necessary in view of the initial limitations of the machine's programming controller, which has a steep learning curve and a specialized command interface. The system obtained as a result of this work allows both researchers and welding workers who use this equipment to have their work facilitated in the programming of welding processes. There are also future possibilities for adapting the system and equipment to new welding weaving techniques and new features.

Keywords: Process automation. Retrofitting. Cartesian Manipulator.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS

HTTP Hypertext Transfer Protocol

IHM Interface Humano-Máquina

LTS Laboratório de Tecnologias da Soldagem

PCB Printed Circuit Board

UART Universal Asynchronous Receiver/Transmitter

URL Uniform Resource Locator

SUMÁRIO

1 INTRODUÇÃO

Desde a primeira revolução industrial, o ser humano tem caminhado para a total automatização dos processos de produção de seus bens duráveis e não duráveis. Esses processos passaram a ser o grande foco do desenvolvimento tecnológico, tendo em vista a busca por máquinas que façam o trabalho do ser humano incessantemente.

Processos de soldagem rudimentares podem ser datados desde os primórdios das civilizações. Como discutido por Pires, Loureiro e Bolmsjö (2006), foram encontrados vestígios de práticas similares às que se tem hoje (como soldagem por pressão ou estado-sólido) nas civilizações Sumérias e Egípcias de 3000 AC. Os processos em questão utilizavam o aquecimento do metal e o impacto de martelos para unir peças metálicas em utensílios rudimentares.

Em se tratando da manufatura de grandes estruturas como embarcações, automóveis, aviões ou foguetes, existe a necessidade de unir uma quantidade considerável de partes metálicas para formar suas grandes estruturas. Soldagem é essencial nesses casos pois se trata do processo que visa a união entre dois objetos a partir da aplicação de uma força capaz de unir suas estruturas moleculares.

Pires et al. (2006) propõe que algum tipo de automação deve ser utilizada quando o número de conexões por meio da soldagem for grande o suficiente. Produzir um carro, por exemplo, exige a junção de dezenas de peças metálicas em formações muitas vezes complexas, o que só pode ser alcançado, na maioria das vezes, através da soldagem automatizada.

Outra preocupação constante em tais manufaturas é a repetibilidade do processo e a qualidade do produto final. Como mostrado por Norrish (2006), a mão-de-obra representa entre 70% e 80% do custo do processo de soldagens manuais, custo esse que abrange uma gama de fatores relacionados ao treinamento, segurança e retrabalho gerados pelo operador. Para processos que exigem alta confiabilidade, como o caso de embarcações e aeronaves, o soldador deve ter treinamento rigoroso, ao passo que falhas na execução da tarefa podem causar o retrabalho parcial ou até o descarte total do produto na linha de produção.

A automatização do processo, nesses casos, retira a responsabilidade de imperfeições das mãos do operador, visto que um equipamento dentro dos padrões de exigência do processo fabril é capaz de repetir centenas de vezes o mesmo procedimento com o mínimo esforço, enquanto distancia o operador da área de soldagem e garante níveis mais altos de segurança dentro dessas etapas de produção.

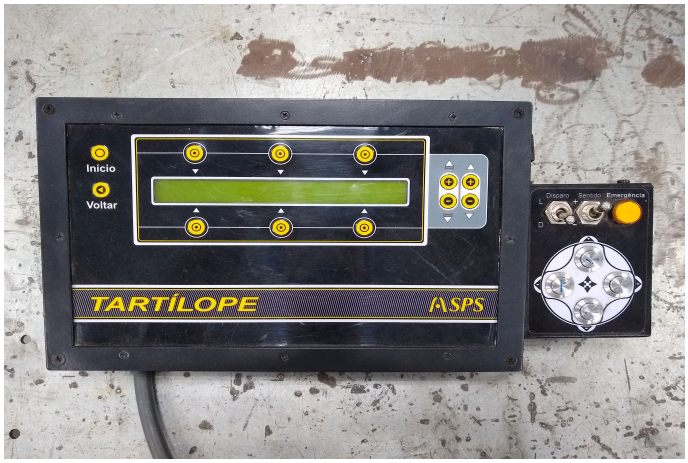
Em estudos relacionados à soldagem, como os conduzidos no Laboratório de Tecnologias da Soldagem (LTS) do Centro Tecnológico de Joinville da UFSC, busca-se

avaliar os impactos das variáveis de processo na qualidade da soldagem. Quando se faz necessário isolar a ação de um parâmetro na qualidade final do processo, deve-se minimizar a influência de fatores externos, como a manipulação da tocha de soldagem pelo soldador. Isso só pode ser atingido pela automatização do processo, uma vez que não utilizar esse artifício torna os resultados imprecisos e muitas vezes até descartáveis para este tipo de pesquisa.

É com isso em mente que escolheu-se para o laboratório a utilização de manipuladores robóticos nas conduções dos experimentos. Um destes equipamentos é o Tartílope V2F da SPS, que consiste em um sistema de movimentação cartesiano de dois eixos, com um trilho flexível que pode ser moldado à superfície de soldagem. Tal equipamento conta com uma interface de programação baseada na modificação de parâmetros escritos em texto em uma tela de LCD, com menus que podem ser navegados utilizando um conjunto de botões. Esse controle comanda um componente interno de processamento que determina a movimentação dos motores da máquina.

Como pode-se notar pela Figura 1, a interface de programação de procedimentos é um tanto quanto rústica, sendo necessário designar variáveis de processo (que muitas vezes não são apresentadas de forma clara) a partir dos parâmetros desejados para a trajetória utilizando os botões de navegação, enquanto as informações são dispostas em apenas duas linhas do display LCD, o que exige a navegação por vários menus até se atingir a configuração de soldagem desejada (vide manual de operação disponibilizado por LABSOLDA (2012)).

Figura 1 – Teclado de programação original.



Fonte: Autor (2019).

Essa característica da construção do teclado torna a utilização do equipamento complexa e contra-intuitiva, restringindo o manuseio a operadores devidamente capacitados. Com estas limitações em mente o presente trabalho busca desenvolver uma interface facilitadora deste manuseio, de modo a diminuir o tempo gasto em treinamento dos integrantes do LTS e a permitir a utilização mais efetiva do equipamento

nas pesquisas desenvolvidas neste ambiente.

1.1 JUSTIFICATIVA

Com o intuito de facilitar a utilização do Tartílope V2F, em vista das dificuldades inerentes à programação de processos que o manipulador oferece originalmente e com o entendimento da necessidade de repetitividade dos processos em estudo, este trabalho busca desenvolver um sistema alternativo de programação de procedimentos de soldagem. Busca-se possibilitar a utilização do novo sistema à partir de computadores comuns através de uma aplicação, que comandará uma placa micro-controlada, programada para acionar os motores da máquina sempre que solicitado pelo operador, possibilitando a troca entre o sistema de comando desenvolvido e o sistema antigo, sem comprometer o funcionamento do equipamento.

1.2 OBJETIVOS

O objetivo geral deste trabalho é tornar a interface de comando do manipulador do Tartílope V2F mais acessível e passível a modificações, tendo em vista as restrições do sistema original, de modo a simplificar a utilização da máquina nos procedimentos rotineiros do laboratório.

Os objetivos específicos são:

a. Examinar a dinâmica de funcionamento do Tartílope para elencar os requisitos de projeto do novo sistema de automatização.
b. Desenvolver um novo sistema de controle para a máquina a partir de uma placa de circuito impresso que permita o acoplamento de um sistema micro-controlado à máquina original;
c. Implementar uma interface usuário-máquina que permita a programação de tarefas de soldagem e o acionamento dos motores do Tartílope a partir do novo sistema desenvolvido, sempre que for solicitado pelo usuário.

1.3 ESTRUTURA DO TEXTO

Este trabalho busca explorar todo o processo de desenvolvimento do sistema proposto e apresentar os resultados obtidos ao fim das atividades propostas. No Capítulo 2 é feita a revisão de uma série de teorias sobre soldagem e automatização de processos. No Capítulo 3 é abordado o processo de desenvolvimento do novo sistema a ser acoplado na máquina, bem como os métodos utilizados para a programação do controlador e dos comandos para os motores. No Capítulo 4 do sistema discutem-se as novas ferramentas desenvolvidas e as limitações do sistema desenvolvido. Por fim, no

Capítulo 5, apresenta-se as considerações finais sobre o desenvolvimento do trabalho e as possíveis continuações do projeto.

2 FUNDAMENTAÇÃO TEÓRICA

A fim de contextualizar o projeto e criar uma fundamentação do trabalho com os conhecimentos necessários para o bom entendimento deste, neste capítulo será feita uma revisão teórica dos conteúdos pertinentes, passando por características básicas de procedimentos de soldagem, até os tipos de automatização dos processos utilizados neste ramo da produção.

2.1 SOLDAGEM

A soldagem é um conjunto de técnicas de manufatura que vêm sendo refinadas à séculos. Com processos rudimentares de união de partes metálicas presentes em diversas civilizações desde a Idade do Cobre, está em evolução constante como um processo de fabricação essencial no desenvolvimento de diversas tecnologias contemporâneas. Com o passar dos séculos, é evidente que uma gama de novos procedimentos foram introduzidos às técnicas de soldagem, acompanhando a evolução tecnológica da humanidade.

Consistindo da aplicação de uma alta energia, usualmente em forma de calor, a soldagem é dada pela fusão de uma ou mais partes que compõem o processo, causando a fusão dos componentes, unindo-os de forma permanente. Existe, no entanto, uma gama de processos de soldagem, cada qual com suas peculiaridades. Serão exploradas algumas dessas técnicas a seguir.

2.1.1 Processos de Soldagem

Como analisado por Norrish (2006), processos de soldagem podem ser divididos em duas grandes categorias: técnicas de soldagem por pressão ou fusão.

Processos de soldagem por pressão dependem diretamente da aplicação de força entre as partes, contando com estratégias peculiares para tornar a união definitiva, entre elas pode-se citar a união por fricção, carga explosiva, aplicação de corrente ou pressão a frio (NORRISH, 2006).

Processos de soldagem por fusão, por outro lado, têm como característica principal a fusão de um material intermediário, que é deposto sobre a junta de solda e forma o que é chamado de cordão de solda. A fusão do material é causada pela aplicação de altas correntes, arcos voltaicos ou campos elétricos de alta densidade e o tipo de fusão diferencia os métodos de soldagem nessa classe.

Grande parte dos processos de soldagem por fusão dependem da interação de um arco elétrico entre as partes a serem unidas. Para fins de simplificação das

teorias abordadas, serão analisadas algumas características construtivas de processos de soldagem por fusão com arco elétrico, tipo de soldagem comumente utilizado no equipamento em estudo (NORRISH, 2006).

Em procedimentos de soldagem com arco elétrico, segundo Wainer, Brandi e Mello (1992), costuma-se trabalhar com uma grandeza de abstração da energia disponível para a soldagem, o aporte líquido de energia H_l, descrito na Equação (1), em que V é a tensão em volts, I a corrente do arco em ampères, e_a é a eficiência do arco e v_s é a velocidade de soldagem em cm/min.

$$H_l = \frac{e_a 60 V I}{v_s} \tag{1}$$

Este parâmetro determina quanta energia estará realmente disponível para a soldagem dos materiais a serem ligados, para cada unidade de deslocamento do cordão de solda. Verifica-se, portanto, que manter uma velocidade de soldagem constante e conhecida é fundamental em processos de soldagem por eletrodo. A automação destes processos torna-se imprescindível em ambientes que exigem repetibilidade e prezam pela confiabilidade do processo.

2.1.2 Automatização da Soldagem

Tendo em vista a necessidade da automatização do processo de soldagem, é preciso avaliar as alternativas presentes no mercado que melhor se adéquem às especificações do processo estudado. Modenesi et al. (2005) dividem os tipos de operação de soldagem em quatro categorias: manual; semi-mecanizado; mecanizado e automático.

No Quadro 1, elaborado por Modenesi et al. (2005), pode-se observar quem é o responsável por executar as atividades relacionadas a cada tipo de operação associada ao procedimento de soldagem. Este quadro demonstra que processos implementados em operação automática são capazes de realizar todo o procedimento de soldagem sem intervenção humana (quando métodos de correção de trajetória são aplicados), enquanto processos mecanizados ou semi-mecanizados ainda exigem a ação ou supervisão de um operador durante o processo.

Apesar de apresentar grande vantagem em relação aos outros tipos de operação, em se tratando de qualidade da solda, tempo de produção e menor suscetibilidade a acidentes (não depende da supervisão do operador), o custo de aplicação deste tipo de automatização é alto e em muitos casos não é suficiente para eliminar a alternativa de solda mecanizada.

Dessa forma, é costumeira a utilização de equipamentos de soldagem mecanizada em linhas de produção ou, no caso em estudo, laboratórios, pois tais equipamentos permitem a programação e execução de tarefas de soldagem através da

Quadro 1 – Classificação dos processos de soldagem revisada

Atividades	**Tipos de operação**			
	Manual	Semi-mecanizado	Mecanizado	Automático
Abertura e manutenção do arco	Soldador	Máquina	Máquina	Máquina
Alimentação de material	Soldador	Máquina	Máquina	Máquina
Controle do calor e penetração	Soldador	Soldador	Máquina	Máquina
Deslocamento da tocha	Soldador	Soldador	Máquina	Máquina
Procura e seguimento de junta	Soldador	Soldador	Soldador	Máquina
Direcionamento da tocha e do arco	Soldador	Soldador	Soldador	Máquina
Correções e compensações	Soldador	Soldador	Soldador	Máquina (opcional)

Fonte: Modenesi, Marques e Bracarense (2005, p. 129)

movimentação e alimentação automática da tocha de solda, bem como o controle do arco e de outros fatores ligados ao processo.

Ogata (2010, p. 5) define sistemas de controle em malha aberta como sistemas "[...] em que o sinal de saída não exerce nenhuma ação de controle no sistema". Portanto, pode-se classificar operações de soldagem mecanizadas como sistemas em malha aberta, tendo em vista a ausência de sensoreamento e consequentemente a ausência de atuação ou controle sobre os sinais do equipamento. Isso implica em uma simplicidade maior no desenvolvimento de sistemas de soldagem mecanizados, mas reafirma a menor flexibilização das tarefas do equipamento em relação a sistemas automáticos.

2.2 SISTEMAS MECANIZADOS

Para suprir a falta do controle automático das ações realizadas por equipamentos de soldagem mecanizados, investe-se em componentes elétricos e mecânicos mais robustos, como motores de passo, que entregam maior precisão na execução de comandos como a movimentação da tocha, sem a necessidade de controle em malha fechada das variáveis do processo.

De acordo com Umans (2014), essa classe de motores oferece a possibilidade de determinar a posição do efetuador simplesmente salvando o número de passos digitais enviados ao motor. Isso implica na redução da complexidade das tarefas de processamento, fator essencial na aplicação de sistemas micro-controlados. Serão

exploradas algumas características deste tipo de motor, de modo a consolidar informações sobre seu funcionamento.

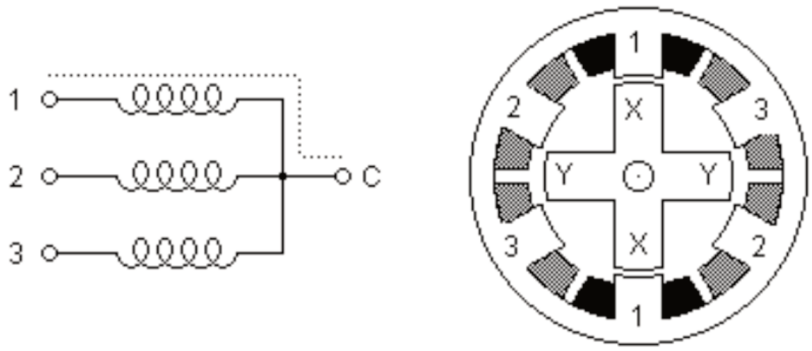
2.2.1 Motores de Passo

Motores de passo são máquinas elétricas cuja característica principal é a energização das bobinas do estator de modo sequencial, passo a passo. Segundo Condit (2004), podem ser divididos em três categorias básicas, de acordo com a construção do rotor, que são: máquinas de relutância variável, máquinas de ímã permanente e máquinas híbridas. Do ponto de vista elétrico, ainda pode-se dividir essas máquinas de acordo com o tipo de enrolamento do estator, sendo estes os motores unipolares, bipolares e bi-filares.

O acionamento dessas máquinas está diretamente ligado à sua construção, uma vez que a rotação do motor está atrelada à disposição dos polos do rotor e dos enrolamentos do estator. Serão exploradas as lógicas de acionamento dos motores de relutância variável e de ímã permanente com enrolamento bipolar, uma vez que os acionamentos da segunda categoria se assemelham.

Na Figura 2 pode-se observar um esquemático das ligações dos enrolamentos e uma vista de corte de um motor de relutância variável de 3 polos, em que os enrolamentos do estator estão numerados de 1 a 3 e os pares de dentes do rotor nomeados por X e Y.

Figura 2 – Exemplo de motor de relutância variável.



Fonte: Condit (2004).

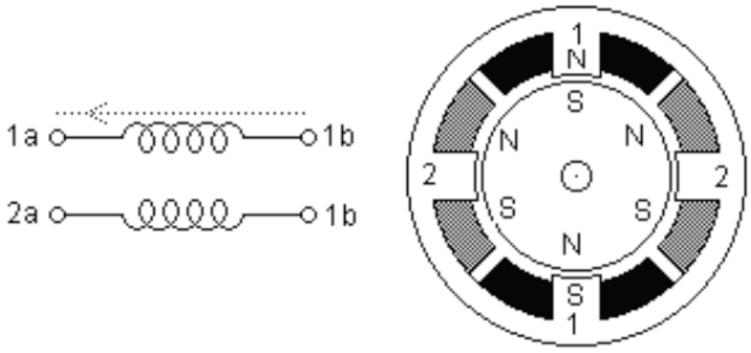
Condit (2004) explica que quando o enrolamento 1 é energizado, os dentes do rotor marcados por X são atraídos para os polos da bobina 1, de modo a minimizar o caminho percorrido pelo fluxo elétrico. Em seguida, energizando somente o enrolamento 2, os dentes marcados por Y são atraídos pelo polo 2, movendo o motor no sentido horário. Energizar somente a bobina 3 atrai os dentes marcados por X, dando sequência ao movimento, passo a passo. Acionar as bobinas sequencialmente na ordem reversa confere ao motor um movimento uniforme no sentido anti-horário.

Cada passo desse exemplo de motor rotaciona o eixo em 30 graus. Para obter

passos angulares menores, faz-se necessário aumentar o número de dentes do rotor e o número de polos no estator. O movimento do eixo deste tipo de motor se dá pelo alinhamento do elemento ferromagnético do rotor com o fluxo elétrico causado pela energização do polo do estator.

No caso de motores com rotor de ímã permanente, o alinhamento se dá pelo campo magnético gerado nos polos do estator. Na Figura 3 pode-se identificar um motor de passo de ímã permanente com enrolamentos bipolares.

Figura 3 – Exemplo de motor de passo bipolar.



Fonte: Condit (2004).

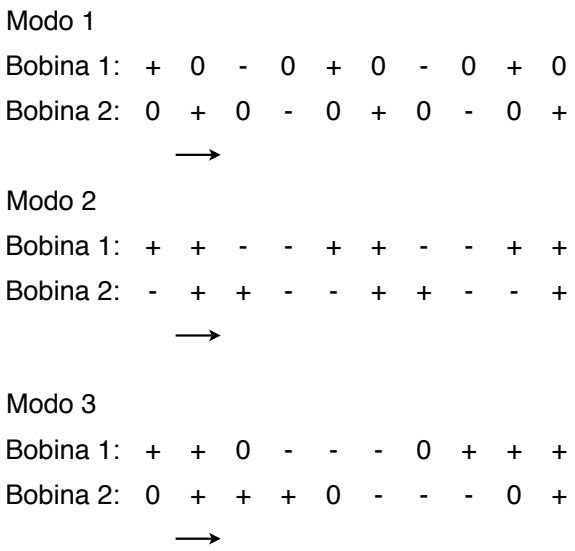
O acionamento deste tipo de motor se dá pela energização direta ou reversa das bobinas 1 e 2. O rotor é composto por 6 polos magnéticos, 3 nortes e 3 sul, arranjados de forma alternada ao redor do rotor. Para facilitar o entendimento, será adotada como corrente positiva aquela que percorre a bobina do terminal b para o terminal a, como ilustrado na Figura 3.

De acordo com Condit (2004), pode-se acionar esse motor de 3 modos distintos, alterando a ordem em que as bobinas são energizadas. A Figura 4 mostra as sequências descritas por CONDIT, com os símbolos “+”, “-” e “0” significando energização positiva, negativa e nula na bobina descrita, respectivamente.

No primeiro modo de funcionamento, somente uma bobina está ativada em cada instante de tempo. Quando a bobina 1 está energizada positivamente, a configuração se iguala à Figura 3, em que os polos da bobina 1 atraem os polos magnéticos do rotor mais próximos. Quando a bobina 2 é energizada positivamente, o polo esquerdo da imagem se torna o norte magnético e atrai o sul magnético mais próximo, criando torque no sentido horário. A sequência continua, invertendo as polaridades das bobinas.

No segundo modo de acionamento, as bobinas permanecem energizadas continuamente, seguindo a mesma lógica do acionamento 1. Esse modo de funcionamento produz torque mais elevado que o modo 1, enquanto demanda uma potência maior, tendo em vista que os enrolamentos estão constantemente energizados.

Figura 4 – Modos de acionamento do motor bipolar.



Fonte: Autor (2020).

Ambos modos de funcionamento produzem uma resolução de passo de 30 graus, ou 12 passos por rotação, para o motor da Figura 4.

Combinar os modos 1 e 2 de funcionamento gera o que é chamado de meio-passo ou half-step, descrito no modo de operação 3, o que faz com que o motor tenha uma resolução de passos por rotação duas vezes maior que nos modos de operação anteriores, ou seja, cada passo gira o eixo do motor utilizado como exemplo em 15 graus, de modo que uma rotação completa é dada a cada 24 passos. Essa maior resolução vem em detrimento de um torque não constante, devido à forma como as bobinas são energizadas.

Combinar as duas técnicas construtivas apresentadas em uma mesma máquina resulta em um motor de passo híbrido. Motores com esse tipo de construção podem ser acionados seguindo a lógica de acionamento do motor de ímã permanente, enquanto adquire características de torque estático e dinâmico mais elevadas que das construções separadas.

Conhecidas as lógicas de acionamentos destes motores, é possível projetar equipamentos capazes de acioná-los com menor esforço. Drivers de motores de passo fazem este papel, transformando comandos lógicos de passo e direção no movimento desejado do eixo do motor. Isso reduz a complexidade do comando de atuação, o que torna essa classe de motores ainda mais compatível com sistemas eletrônicos digitais.

De interesse deste trabalho, serão estudadas as ferramentas necessárias para o desenvolvimento de um sistema de comando micro-controlado para os motores de passo do Tartílope V2F.

2.3 SISTEMAS MICRO-CONTROLADOS

Um microcontrolador consiste na associação de um microprocessador à memórias e um conjunto de periféricos de entrada e saída, de modo que o sistema final se assemelhe a um computador, capaz de receber dados, processar tarefas e enviar comandos.

Sua principal diferenciação, com relação a computadores e controladores mais robustos, é a especialização, uma vez que seus componentes de memória, variedade de periféricos e capacidade de processamento são projetados visando um conjunto de tarefas limitado e otimizado. Essa especialização reduz o tamanho destes equipamentos, diminui o custo de produção e consequentemente torna-os mais acessíveis.

Essa acessibilidade torna a tarefa de escolher um microcontrolador importante, tendo em vista a necessidade de adequá-lo satisfatoriamente à aplicação. Deve-se levar em consideração a capacidade de processamento exigida pela aplicação, bem como as necessidades de periféricos específicos, como por exemplo o número de portas lógicas, entradas e saídas analógicas, periféricos de comunicação, entre outros.

Em projetos de sistemas mais elaborados, que demandam maior confiabilidade na execução de tarefas ou exigem um hardware mais robusto, é possível distribuir as rotinas de processamento em mais microcontroladores. Essa decisão pode garantir um melhor funcionamento do sistema, ao passo que cada microcontrolador tem sua programação especializada em um conjunto reduzido de tarefas. Feito um bom projeto, as partes do sistema podem ser modificadas de forma independente, de modo que o sistema como um todo mantenha suas funcionalidades gerais.

Realizar um projeto com arquitetura distribuída, consequentemente, insere a necessidade de comunicação entre os microcontroladores do sistema, uma vez que deve existir conexão entre as decisões tomadas pelas partes. Desse modo, serão estudados os principais protocolos de comunicação utilizados neste projeto.

2.3.1 Protocolo de Comunicação: UART

De modo a satisfazer a necessidade do envio e recebimento de mensagens entre microcontroladores, computadores e componentes eletrônicos, uma série de periféricos e protocolos de comunicação foram implementados ao longo dos anos. Bell, Mudge e McNamara (2014) descrevem o processo de desenvolvimento de alguns de seus projetos na Digital Equipment Corporation. Um destes periféricos é o Universal Asynchronous Receiver/Transmitter (UART), ou Receptor/Transmissor Assíncrono Universal, em tradução livre.

Criado por Gordon Bell como parte de um dos primeiros computadores, o PDP-1, esse protocolo assume um formato serial de comunicação, em que os dados

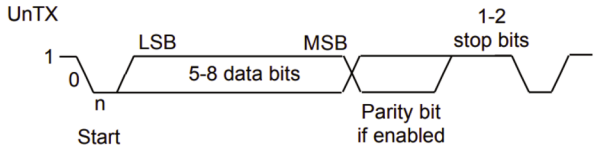
são enviados em um único meio de transmissão ou cabo, ordenados no tempo. Para isso, deve ser determinada uma velocidade de comunicação ou baud-rate entre os interlocutores, de modo que seja possível detectar as mensagens assíncronas.

A utilização desse protocolo é usualmente atrelada a um periférico de hardware nos interlocutores, que é responsável por receber e armazenar os bits em registradores de deslocamento. Isso implica em limitações de hardware ligadas ao número de portas que podem ser utilizadas na comunicação UART.

Uma vez que só é necessário um meio de comunicação para o envio de uma mensagem, dois interlocutores podem ser ligados de modo a estabelecer uma comunicação simplex (unidirecional), half-duplex (bidirecional, apenas um interlocutor envia mensagem por vez por um mesmo meio) ou full-duplex, utilizando dois cabos, permitindo comunicação bidirecional concomitante.

A Figura 5 mostra um quadro genérico para envio de um caractere via protocolo UART, no microcontrolador TM4C123GH6PM. O meio de transmissão permanece em nível lógico alto enquanto ocioso e o receptor identifica um inicio de mensagem quando recebe um bit de nível baixo.

Figura 5 – Quadro de mensagem UART.



Fonte: Texas Instruments (2014, p. 896)

Ao perceber a mudança de nível lógico do meio, o receptor inicia a amostragem, armazenando nos registradores de deslocamento os bits subsequentes. A mensagem padrão da comunicação pode ser configurada para carregar de 5 a 8 bits de dados, do bit menos significativo para o mais significativo. É possível também configurar a utilização de um bit de paridade (utilizado como detector de erro) e pelo menos um ou até dois bits de parada, que retornam o nível lógico do meio para alto.

Por se tratar de um protocolo assíncrono, é usual encontrar microcontroladores com buffers de recebimento e envio de mensagens UART. O microcontrolador TM4C123GH6PM, por exemplo, possui duas filas de armazenamento de 16 Bytes, uma para recebimento e outra pra envio. Isso garante uma redução de perdas de mensagem caso o sistema esteja em bloqueio ou executando outras rotinas.

Vale notar que caso uma fila fique cheia, mensagens subsequentes podem ser perdidas. Ruídos na transmissão também podem causar perda de mensagem, uma vez que bits de paridade e finalização da mensagem errados determinam erro no quadro de dados, que não é armazenado pelo microcontrolador em questão.

Uma vez consideradas as vantagens e desvantagens de utilizar esse protocolo,

um bom projeto pode criar estratégias para reduzir perdas de mensagem enquanto torna a comunicação eficiente. Boas estratégias são transmitir mensagens pequenas, reduzindo a chance de sobrecarga do buffer de recebimento e utilizar baud-rates menores, diminuindo as chances de ruído na transmissão.

Compreendidos os conceitos pertinentes à comunicação cabeada entre microcontroladores, é válido estudar como são enviadas e recebidas mensagens entre aplicações Web, uma vez que parte considerável deste trabalho envolve troca de mensagens via rede sem fio.

2.3.2 Protocolo de Aplicação: HTTP

O Hypertext Transfer Protocol ou Protocolo de Transferência de Hipertexto, em tradução livre, é um protocolo padrão de requisição-resposta entre clientes e servidores da Internet. Classificado como um protocolo da camada de aplicação do Modelo OSI, utilizado para trocar mensagens entre computadores diferentes ligados à Internet, Kurose e Ross (2010, p. 73) explicam que "o HTTP define a estrutura dessas mensagens e o modo como o cliente e o servidor as trocam".

As mensagens padrão do protocolo HTTP são compostas por uma linha de solicitação/status, um cabeçalho e um corpo de texto, variáveis de acordo com o tipo de solicitação feita, carregando informações importantes sobre a localização do documento, o status do servidor, o tipo do arquivo requerido, entre outros.

A primeira linha da requisição HTTP é a de solicitação. Composta pelo tipo de solicitação, o Uniform Resource Locator (URL) do documento e a versão do HTTP, essa linha informa ao servidor qual arquivo acessar e qual ação tomar em relação ao arquivo. O Quadro 2 relaciona o tipo de solicitação do HTTP 1.1 com a ação tomada pelo servidor, segundo Forouzan (2009).

Quadro 2 – Tipos de solicitação de recurso do HTTP 1.1.

Método (Comando)	Ação
GET	Solicita um documento ao servidor.
HEAD	Solicita informações sobre um documento, mas não o documento em si.
POST	Envia informações do cliente para o servidor.
PUT	Envia um documento do servidor para o cliente.
TRACE	Ecoa uma solicitação que chega.
CONNECT	Reservado.
OPTION	Solicita detalhamento sobre opções disponíveis.

Fonte: Forouzan (2009, p. 863).

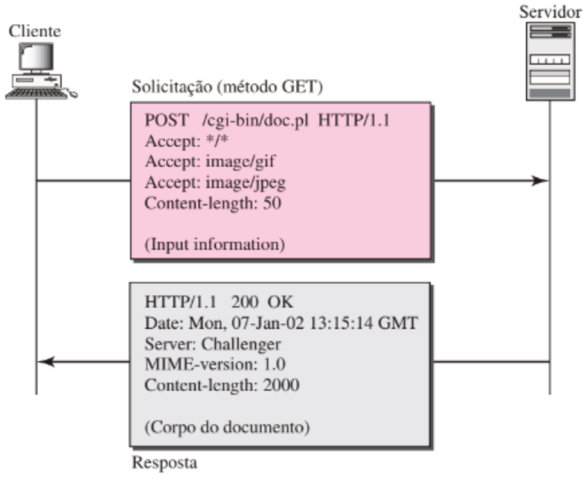
O tipo de solicitação é separado do URL por um espaço. O URL carrega a informação do endereço do arquivo, assumindo o formato de *protocolo:// hospedeiro:porta/caminho*. É possível omitir os três primeiros campos, desde que o documento a ser acessado esteja armazenado no destinatário final da mensagem e

seja acessado pela porta padrão 80. Nesse caso, o cabeçalho da requisição deverá conter as informações do destinatário.

Para finalizar a linha de solicitação, é informada a versão do HTTP utilizada na requisição, também separada por espaço. Em uma resposta, essa linha é substituída por uma linha de status. É informada, então, a versão do HTTP utilizada na resposta, o código de status da resposta e a frase de status.

A Figura 6 mostra um exemplo de mensagens HTTP trocadas entre um cliente e servidor para uma requisição do tipo POST. É possível observar a diferença entre os campos de solicitação e resposta de um arquivo para um servidor remoto.

Figura 6 – Exemplo de trocas de mensagem HTTP.



Fonte: Forouzan (2009, p. 867)

A mensagem de resposta da Figura 6 mostra uma resposta com código 200 e frase "OK". Forouzan (2009) divide os códigos de status em cinco grandes grupos, sendo eles os status informativos, de sucesso, de redirecionamento, de erros no cliente e de erros no servidor. A frase do status expande as informações do código, fornecendo maiores detalhes com relação aos erros ou ações tomadas pelo servidor.

Em seguida, são escritos os cabeçalhos da requisição/resposta. São nestes campos subsequentes que informações importantes com relação à mensagem são armazenadas. Forouzan (2009) divide esses campos em quatro categorias: cabeçalhos gerais, de entidade, de solicitação e de resposta.

Cabeçalhos gerais fornecem informações sobre a mensagem, como data, controle de cache e tipo de conexão. Enquanto isso, cabeçalhos de entidade carregam informações sobre o corpo do documento enviado na mensagem HTTP, como o tipo de codificação e o tamanho do corpo de dados.

Cabeçalhos de solicitação são exclusivos de requisições e carregam informações sobre o cliente e preferência quanto ao formato do documento, como a linguagem utilizada pelo cliente, tipos de arquivos suportados, credenciais de

autenticação, entre outros. Os cabeçalhos de resposta são exclusivos de respostas e carregam as configurações do servidor quanto aos métodos suportados, nome e versão do servidor, idade do arquivo, entre outros.

Uma vez escritos os cabeçalhos necessários para a comunicação, pode ser adicionado ao fim da mensagem HTTP um corpo com dados pertinentes. A mensagem está então completa e pode ser encapsulada em protocolos de camadas inferiores, para ser transmitida.

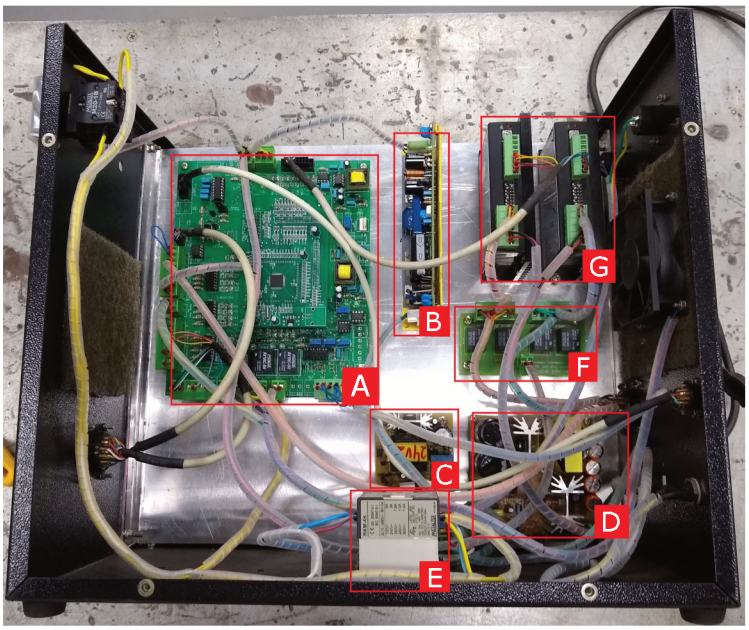
Com isso finalizam-se as principais teorias necessárias para o embasamento do projeto em questão. Em seguida serão exploradas as metodologias utilizadas para o desenvolvimento do trabalho.

3 METODOLOGIA

Para desenvolver um dispositivo capaz de manter as funcionalidades do Tartílope V2F e ao mesmo tempo permitir a programação de tarefas utilizando métodos computacionais, é de extrema importância conhecer a dinâmica interna de funcionamento da máquina, para enfim determinar as ações a serem tomadas para o desenvolvimento do projeto. Com isso em mente, foi feita uma análise do interior da máquina, observando os principais componentes e as ligações entre eles. Com essas informações foi desenvolvido um esquemático de máquina para o Tartílope V2F, apresentado no Apêndice A.

Pode-se observar na Figura 7 os principais componentes da máquina destacados em vermelho, sendo estes: o sistema de comando da máquina (A), as fontes de tensão de 15V (simétrica), 24V e 30V (B, C e D, respectivamente), a contatora (E), o conjunto de relês de comando dos drivers (F) e os dois drivers de motor (G). O painel com os botões de liga-desliga e as interfaces de conexão ficam na face frontal e traseira da máquina, à esquerda e direita da figura, respectivamente.

Figura 7 – Interior do Tartílope V2F.

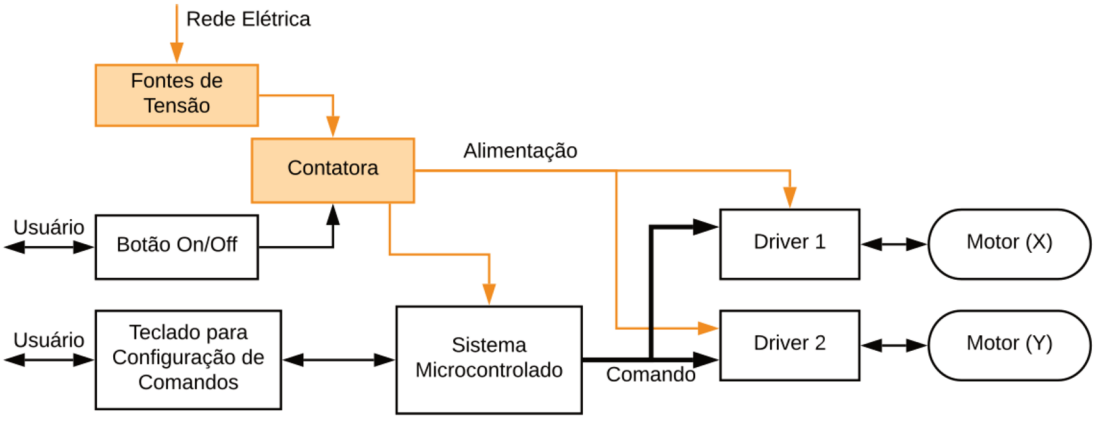


Fonte: Autor (2020).

Na Figura 8 é possível visualizar um resumo da dinâmica de comando do equipamento. Alimentado pela rede elétrica (em amarelo), três componentes principais atuam no funcionamento geral do Tartílope: o teclado (visto na Figura 1), o sistema de comando e os drivers dos motores.

Conhecendo a dinâmica do equipamento, foram determinados os requisitos de

Figura 8 – Fluxograma de funcionamento do Tartílope V2F.



Fonte: Autor (2020).

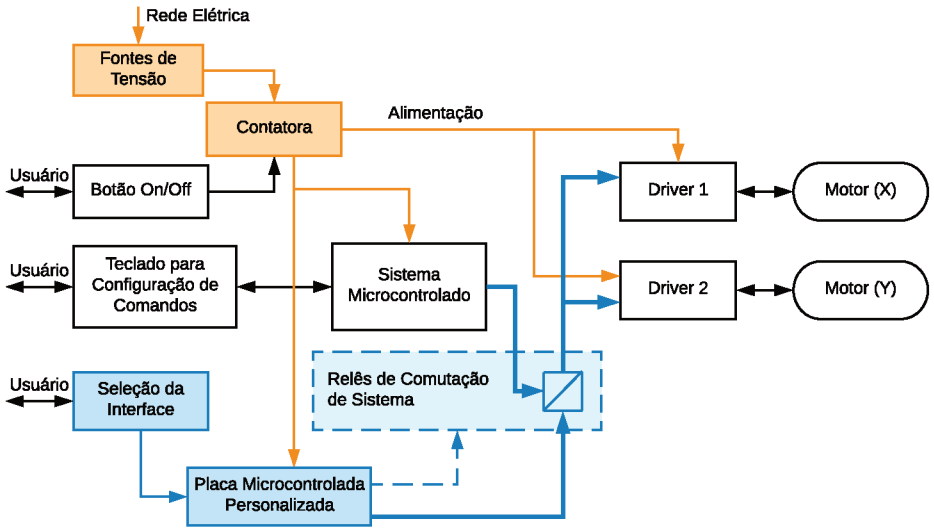
projeto, em conjunto com o Prof. Dr. Tiago Vieira da Cunha, coordenador do LTS da UFSC Joinville, que possui um grande conhecimento dos processos de soldagem e equipamentos utilizados no laboratório. Foi determinado que os seguintes requisitos deveriam ser atingidos:

- Manter um indicativo da posição atual do efetuador final, atualizada periodicamente;
- Permitir a configuração de variáveis de processo como velocidade de movimentação, relação passos por milímetro para os motores, entre outras;
- Possibilitar a realização de movimentações dos eixos de forma independente pelo chamado jogging;
- Permitir a movimentação da máquina até uma posição determinada no plano;
- Alternar entre o controle antigo e o novo;
- Tornar a posição atual do manipulador a referência do sistema de coordenadas;
- Ter apresentação clara e intuitiva das funções da máquina;
- Possibilitar a comunicação sem fio entre a IHM e o equipamento.

Com base nessas exigências, foi esquematizada uma nova dinâmica de funcionamento do sistema, ilustrada na Figura 9, em que o novo sistema a ser implementado está destado em azul.

Como já discutido anteriormente, a programação de tarefas na interface original da máquina demanda treinamento rigoroso e pleno conhecimento dos processos de soldagem para a determinação correta dos parâmetros utilizados nas configurações do equipamento. Com o intuito de criar um novo sistema de comando desse equipamento, portanto, decidiu-se utilizar uma interface de programação com uma placa micro-controlada capaz de receber os comandos de outro dispositivo por meio de comunicação remota.

Figura 9 – Dinâmica de funcionamento do sistema modificado.



Fonte: Autor (2020).

Prezando pela robustez do equipamento, optou-se por dividir os sistemas de acionamento dos motores e de envio e recebimento de tarefas em dois microcontroladores especializados. Essa estrutura confere segurança ao sistema de comando dos motores pois possibilita a redução da carga de processamento de tarefas, enquanto passa a responsabilidade das comunicações entre a Interface Humano-Máquina (IHM) e o sistema de comandos para um microcontrolador especializado.

Com a definição dos subsistemas necessários para o funcionamento do equipamento, pode-se passar para o desenvolvimento da Printed Circuit Board (PCB) - placa de circuito impresso, do inglês.

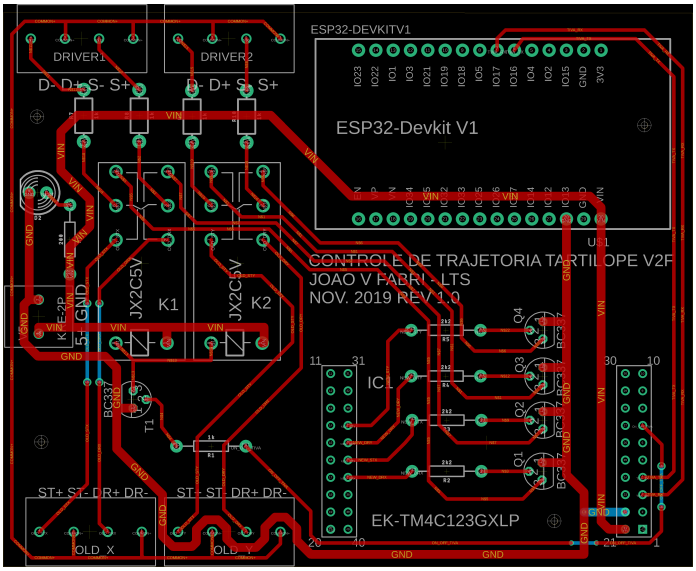
3.1 PLACA DE ACOPLAMENTO DO NOVO CONTROLADOR

Identificada a dinâmica interna do equipamento, foi possível inferir que uma forma de atuar sobre os motores é a de interceptar o sinal de comando enviado pelo sistema de controle da máquina para os drivers.

Para atuar sobre tal sinal, foram utilizados dois relês, acionados por um dos microcontroladores, como visto no esquemático da PCB, na Figura 10. Quando não energizado, o sinal de entrada dos drivers de motor é igual ao sinal enviado pelo sistema de comando original da máquina. Quando energizado, o sinal de entrada dos drivers é o vindo do microcontrolador do novo sistema de comando.

A comutação entre os sistemas pode ser feita por comandos diretos do microcontrolador, o que permite um alto grau de personalização da interface de comando por parte do programador, possibilitando cumprir com uma das requisições

Figura 10 – Footprint da placa desenvolvida.



Fonte: Autor (2020).

de projeto elencadas, de que o chaveamento possa ser feito através da interface de programação de tarefas.

Como pode-se observar pela Figura 10, a nova placa de comando consiste da ligação de dois microcontroladores de baixo custo e fácil acesso, o ESP32 Devkit e o EK-TM4C123GXL (doravante chamado apenas de Tiva C). O primeiro é encarregado da interface de comunicação sem fio (WiFi), enquanto o segundo é encarregado da execução das tarefas de movimentação e do chaveamento dos sinais de comando.

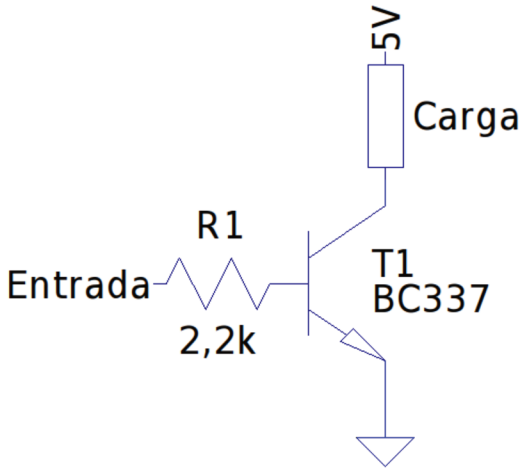
O modelo com dois microcontroladores foi adotado com o intuito de centralizar a lógica de funcionamento da máquina no Tiva C, possibilitando ao sistema funcionar de modo independente da comunicação sem fio dados pequenos ajustes na programação do microcontrolador. Desde que se respeite o protocolo de envio e recebimento de mensagens ao Tiva C, qualquer microcontrolador ou novo sistema implementado pode ser utilizado para comandar a máquina, visto que ambos os microcontroladores funcionam de forma independente.

Além dos microcontroladores e dos relês, foi adicionado um circuito de elevação de tensão em cada uma das portas de saída de sinal ligadas às entradas dos drivers de motor e às bobinas dos relês. Isso foi feito tendo em vista que o sinal de entrada dos drivers presentes na máquina (STR8 da Kalatec) requerem sinal lógico de 5V (como observado no manual de utilização do driver, no Anexo B), enquanto o Tiva C tem portas de saída com nível lógico de 3,3V.

A Figura 11 ilustra o circuito utilizado para adequar o sinal lógico do Tiva C à ativação dos drivers. Foram escolhidos transistores BC377 atuando como chave, selecionados de acordo com suas especificações de funcionamento. Quando a tensão

na base do transistor é 0V, este atua como chave aberta, o que torna a tensão de saída do circuito nula. Com nível lógico alto (tensão 3,3V), o transistor atua como chave fechada, aplicando na carga a tensão de 5V.

Figura 11 – Esquemático do circuito chaveador de sinal.



Fonte: Autor (2020).

Observou-se que o indexador de comando da placa original trabalha com os sinais STEP+, DIR+ e EN+ dos drivers ligados à fonte de tensão, enquanto o sinal de acionamento dos drivers é aplicado nas contrapartes negativas destes sinais. O comportamento foi mantido no projeto da placa de controle desenvolvida.

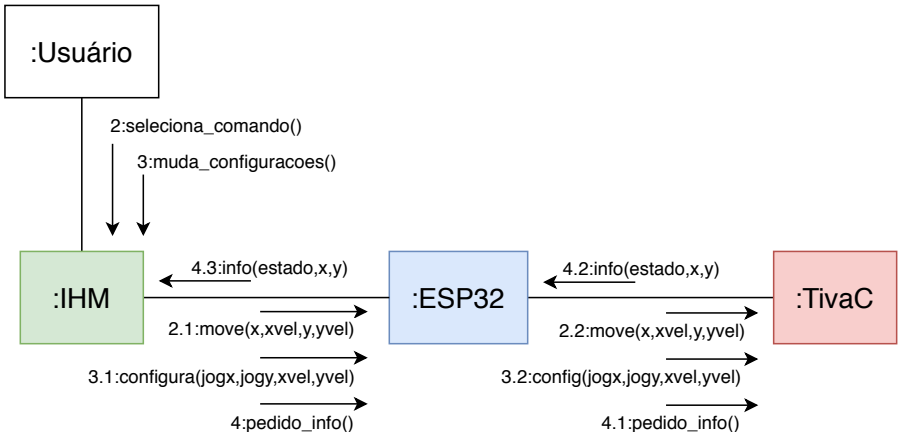
Por fim, foi adicionado um LED para identificar que a placa está alimentada e outro para a identificação visual de qual sistema de comando está ativo. O segundo LED foi ligado a uma porta lógica do Tiva C e acoplado ao painel da máquina. Com o projeto da placa pronto, passou-se para o desenvolvimento do firmware do sistema.

3.2 DINÂMICA DE COMUNICAÇÃO

Os microcontroladores foram programados prezando pela modularização do sistema, obedecendo um esquema de troca de mensagens que permitisse ao Tiva C a execução do comando dos motores e recebimento e envio de mensagens com um baixo tempo de bloqueio, de modo a evitar atrasos nas tarefas ou perdas de informação de comandos mais críticos. Na Figura 12 pode-se visualizar um diagrama resumido das trocas de mensagem internas ao equipamento.

O diagrama da Figura 12 apresenta as trocas de mensagem dos comandos de movimentação, configuração e informação. O comando de parada de emergência, requisitado pelo usuário, obedece a mesma ordem do pedido de informação, enquanto os demais comandos, como o de jogging, se assemelha ao de movimentação, sem carregar dados. As ligações entre a Interface Humano-Máquina e o ESP32 se dão pela conexão WiFi, com troca de mensagens por HTTP, enquanto as mensagens entre o

Figura 12 – Diagrama de Comunicação.

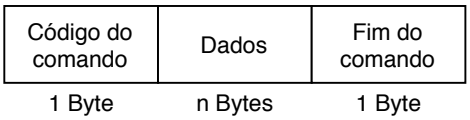


Fonte: Autor (2020).

ESP32 e o Tiva C são trocadas pelo protocolo UART.

Elaboradas de modo a cumprir com o levantamento de requisitos feito no início do projeto, as mensagens trocadas entre os dois microcontroladores do sistema obedecem uma estrutura que pode ser visualizada na Figura 13. Cada comando desenvolvido recebeu um código único de um Byte e exige um corpo de dados de 0 a 16 Bytes, que são enviados sequencialmente, um a um, utilizando o protocolo UART.

Figura 13 – Estrutura da mensagem entre microcontroladores.



Fonte: Autor (2020).

No Quadro 3 pode-se observar como são construídas cada uma das mensagens implementadas. Os dados que compõem as mensagens de movimentação e configuração são inteiros de quatro bytes, sendo que tanto o número de passos a serem percorridos quanto o número de passos por jogging são inteiros sem sinal, enquanto os valores de velocidades carregam inteiros com sinal.

Os códigos para comandos de Jogging na direção positiva e negativa do eixo X e na direção positiva e negativa do eixo Y são, respectivamente, 0xF9, 0xF7, 0xF4 e 0xF1, e foram omitidos do Quadro 3 por questões de clareza. Esses comandos não recebem dados como corpo, e têm comprimento de dois Bytes.

As mensagens de resposta do Tiva C para o ESP32 obedecem a mesma estrutura determinada anteriormente, no entanto, existe retorno somente em resposta aos comandos de parada imediata e informação, que carregam no corpo da mensagem o número de passos dados pelos motores dos eixos X e Y desde o recebimento do último comando de movimentação ou jogging. Isso permite à IHM manter um controle

Quadro 3 – Mensagens recebidas/enviadas pelo Tiva C

Comando	Código	Corpo	Comprimento
Mover	0xFE	passos e velocidade em cada eixo	18
Conecta sinal	0xEE	vazio	2
Parada Imediata	0xDD	vazio	2
Desconecta sinal	0xCC	vazio	2
Informação	0xBB	vazio	2
Finalizado	0xAA	vazio	2
Configurar	0x99	passos por jog, velocidades padrão	14
Mudar Origem	0x88	vazio	2

Fonte: Autor (2020).

da posição relativa da máquina.

Enquanto a comunicação entre o Tiva C e o ESP32 foi feita utilizando o protocolo UART para o envio e recebimento das mensagens, a comunicação do usuário, por intermédio da IHM, com o ESP32 foi desenvolvida à partir de uma estrutura cliente - servidor web, em que a IHM é responsável por estruturar as requisições HTTP e o microcontrolador é responsável por encaminhar os pedidos ao Tiva C.

As requisições obedecem à estrutura convencional, com header e corpo de dados. Cada comando é diferenciado pelo URL da requisição, uma lista com a correspondência entre os comandos implementados pode ser observada no Quadro 4.

Quadro 4 – Requisições HTTP padrão.

Comando	URL	Corpo (exemplo)
Mover	/COMMAND	x=100&y=100&xv=100&yv=100
Configurar	/CONFIG	jogx=100&jogy=100&xv=100&yv=100
Conecta sinal	/ON	vazio
Parada Imediata	/HALT	vazio
Desconecta sinal	/OFF	vazio
Informação	/INFO	vazio
Mudar origem	/ORIGIN	vazio
Jogging	/JOGX+	vazio
	/JOGX-	vazio
	/JOGY+	vazio
	/JOGY-	vazio

Fonte: Autor (2020).

Os comandos recebidos pelo ESP32 obedecem as mesmas determinações citadas anteriormente em relação aos campos presentes no corpo da mensagem, de modo que os dados são codificados em um esquema de chave-valor, como exemplificado no Quadro 4, em que o par chave-valor é separado por um “=” e cada par é separado por um “&”. Com exceção do pedido de informação, que é uma requisição do tipo GET, todas as outras requisições são do tipo POST.

As respostas enviadas pelo ESP32 carregam uma página HTML simples com as funções básicas da interface, que pode ser aberta por navegadores, um campo de código que identifica qual o estado de funcionamento da máquina (parada, executando tarefa, tarefa finalizada, etc) e os campos de dados com as posições, para as respostas de parada imediata e informação, que obedecem a mesma codificação de chave-valor da requisição.

Para programar a interface de recebimento de pacotes HTTP de forma mais eficiente, foram utilizadas as bibliotecas “WebServer.h” (GROKHOTKOV, 2014a) e “WiFi.h” (GROKHOTKOV, 2014b), utilizando a IDE da Arduino para compilar e carregar a programação no ESP32.

A biblioteca “WebServer.h” implementa um conjunto de ferramentas no melhor desenvolvimento de aplicações web. Foi instanciado um objeto da classe WebServer, que possui um conjunto de métodos que auxiliam no recebimento, leitura e envio de requisições HTTP.

Utilizando o método “on” de um objeto da classe WebServer, é possível configurar o servidor para identificar as requisições recebidas, passando como argumentos o URL para identificação do comando, o tipo de requisição (GET ou POST) e a função a ser chamada quando for recebida a requisição.

Em requisições do tipo POST com corpo de dados codificado por chave-valor, é possível acessar dados através do método ”arg”, que recebe uma string com a chave do dado a ser lido e retorna o valor em formato string. Isso permite uma interpretação dos dados recebidos na requisição de forma muito mais eficiente.

Para efetivamente receber requisições, o ESP32 deve preparar o módulo WiFi para atuar como ponto de acesso. É possível configurar o microcontrolador utilizando a função “softAP” da biblioteca WiFi. Uma vez iniciada a rotina de conexão e configurados os interpretadores das requisições, basta executar o método “handleClient” do objeto WebServer instanciado periodicamente.

Com a comunicação entre os microcontroladores e a Interface Humano-Máquina implementada, foi possível desenvolver a lógica de execução de comandos e o aperfeiçoamento da interface visual do equipamento.

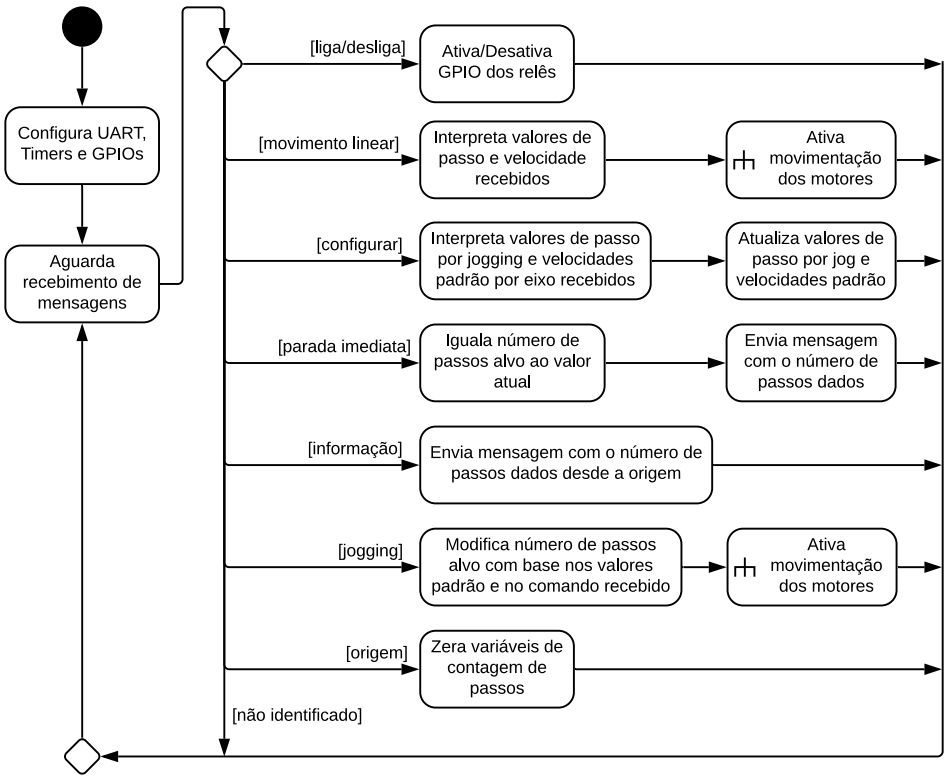
3.3 EXECUÇÃO DE TAREFAS

Com o intuito de criar um sistema de execução de tarefas robusto e introduzir uma proteção contra falhas de comunicação, ficou sob responsabilidade do Tiva C o recebimento de mensagens com comandos vindos do ESP32 e a execução das movimentações dos motores da máquina.

Recebendo as mensagens por protocolo UART, o microcontrolador permanece “escutando” o canal de comunicação até receber uma mensagem com comando, que é

então interpretada. A Figura 14 mostra um diagrama que representa o laço principal da programação do microcontrolador Tiva C.

Figura 14 – Diagrama de atividades do Tiva C.



Fonte: Autor (2020).

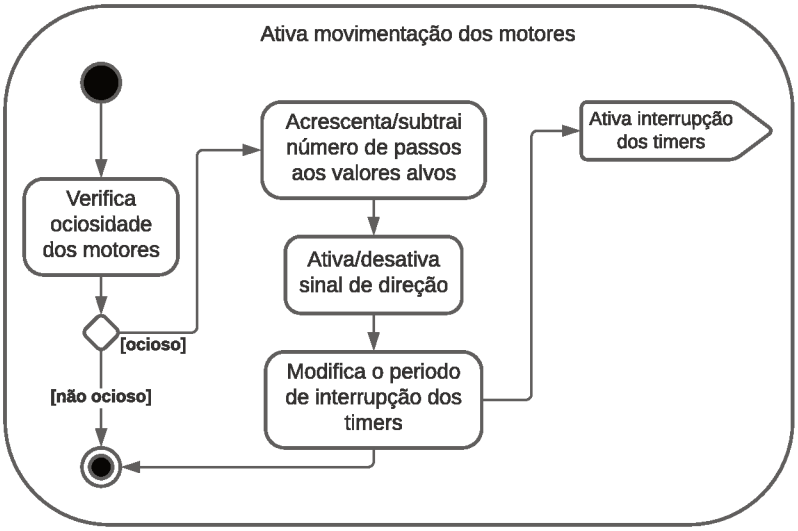
Quando ativados, os timers do Tiva C geram interrupções em um intervalo previamente definido na configuração do timer. Desse modo, comandos que modificam recursos com disputa (como as variáveis de contagem de passos e velocidades dos eixos, por exemplo) só tem suas funções executadas quando a máquina está parada. Os comandos de pedido de informação e de parada de emergência não disputam recursos com as demais funções, uma vez que não modificam os valores de variáveis críticas.

Como já descrito anteriormente, os comandos de movimentação linear e configuração recebem dados no corpo da mensagem. Durante o ciclo de interpretação do comando de movimentação, o microcontrolador lê os valores de passos a serem dados pelos motores e a velocidade com que os motores devem executar a movimentação. Esses dados, recebidos em um conjunto de 4 bytes, são convertidos em inteiros de 32 bits com e sem sinal e salvos.

Para manter um controle de posição relativa interno ao Tiva C, cada eixo possui duas variáveis que armazenam o número de passos atual e o número de passos alvo. Receber um comando de movimentação linear ou jogging acrescenta ou subtrai

uma quantidade de passos do número de passos alvo. Em seguida, são ativadas as interrupções dos timers. A Figura 15 mostra o diagrama de atividades da movimentação.

Figura 15 – Diagrama da movimentação dos motores.



Fonte: Autor (2020).

Deve-se notar que o comando de movimentação linear requer a leitura dos dados de passos e velocidade recebidos, enquanto os comandos de jogging têm número de passos a serem incrementados e velocidade de movimento fixos, que podem ser alterados pelo comando de configuração. Feito isso, o microcontrolador obedece as ações descritas no diagrama da Figura 15.

Cada ciclo dos timers invoca uma interrupção, invertendo o estado lógico da porta de saída para o sinal do driver de motor correspondente ao eixo. A posição atual é atualizada a cada dois ciclos, uma vez que um passo de motor é dado na borda de descida do nível lógico (nível alto para nível baixo). Se o valor da posição atual for igual à posição alvo, a interrupção do timer é desativada.

Para fazer a conversão do vetor de bytes recebido na mensagem para inteiros de 4 bytes, foi declarado um novo tipo, BitConverter, uma união entre uint32_t e um vetor de uint8_t. A Figura 16 mostra um trecho do código da função de leitura dos dados recebidos em uma mensagem de movimentação.

Figura 16 – Fragmento da função de leitura da mensagem de movimentação.

```
1     for(i=0;i<4;i++)  bits.u8a[i] = words [16-i];
2     yvel_temp = (int) bits.u32i;
```

Fonte: Autor (2020).

Para realizar a conversão de um valor, é declarada a variável “bits”, do tipo BitConverter. Como é desejado converter um conjunto de 4 Bytes para o valor

correspondente do tipo inteiro, são atribuídos os Bytes recebidos (contidos nas posições do vetor “words”) aos membros do tipo inteiro de 8 bits do conversor. Acessar o atributo complementar retorna o valor convertido para o tipo inteiro de 32 bits, que é convertido no tipo inteiro.

Uma vez que o Tiva C recebe os dados dos valores na ordenação big-endian e sua memória trabalha em little-endian, é necessário inverter a ordem com a qual os bytes são inseridos no vetor de uint8_t do conversor, o que é feito no laço “for” de associação dos valores lidos às posições de “bits”.

É pertinente também comentar sobre as funções de gerenciamento das interrupções dos timers (“Timer0IntHandler” e “Timer1IntHandler”). Como já descrito, essas funções são chamadas a cada fim de ciclo dos timers. A Figura 17 mostra a função “Timer0IntHandler”, que gerencia a interrupção do timer do motor do eixo X. As variáveis globais “cur_x_pos”, “target_x_pos” e “x_positve” guardam, respectivamente, a posição corrente, a posição final desejada e a direção do movimento do eixo X. A variável booleana “stepx” alterna entre o nível lógico alto e baixo a cada chamada da função “setXMovers”, correspondendo ao nível lógico do sinal enviado ao driver de motor.

Figura 17 – Trecho da função de gerenciamento da interrupção do Timer 0.

```
1     TimerIntClear(TIMER0_BASE , TIMER_TIMA_TIMEOUT);
2     if(cur_x_pos  < target_x_pos  &&  x_positive){
3         setXMovers ();
4         if(! stepx) cur_x_pos  ++;
5     } else if (cur_x_pos  > target_x_pos  && !x_positive) {
6         setXMovers ();
7         if(! stepx) cur_x_pos  --;
8     } else {
9         TimerDisable(TIMER0_BASE ,TIMER_A);
10    }
```

Fonte: Autor (2020).

A função “TimerIntClear” limpa as flags de interrupção, impedindo que outra interrupção seja chamada pelo mesmo timer antes de finalizar a execução. Em seguida, são checados os valores das variáveis de controle de posição e acionados os motores (pelas funções “setXMovers” ou “setYMovers”). Caso a posição atual chegue ao valor da posição alvo ou seja incompatível com o movimento (posição atual maior que posição final enquanto a máquina se move no sentido positivo, por exemplo), é bloqueada a ativação de novas interrupções pela função “TimerDisable”.

Uma vez implementadas as funções que habilitam o sistema a executar os pedidos de movimentação, passou-se para o desenvolvimento da interface gráfica de comando da máquina e a aplicação para gerenciamento dos comandos apresentados graficamente.

3.4 INTERFACE HUMANO-MÁQUINA

Sob a perspectiva de proporcionar uma interface intuitiva de utilização das funções implementadas no novo sistema de comando do manipulador, desenvolveu-se uma aplicação utilizando o framework de desenvolvimento de interfaces gráficas QT (The QT Company, 2014). Utilizando a linguagem de programação C++ (ISO, 2014), atrelada a um conjunto de bibliotecas e ferramentas que permitem um desenvolvimento mais facilitado de softwares, a ferramenta foi escolhida por permitir um projeto de interface intuitivo e simplificado, além de já ser da familiaridade do programador.

Uma vez escolhida a ferramenta de desenvolvimento, foi implementada a interface gráfica utilizando a ferramenta QT Creator (The QT Company, 2014). Na Figura 18 é possível observar a janela de envio de comandos antes (a) e depois (b) de se conectar aos motores.

Figura 18 – Interface de comando do manipulador.

(a) (b)

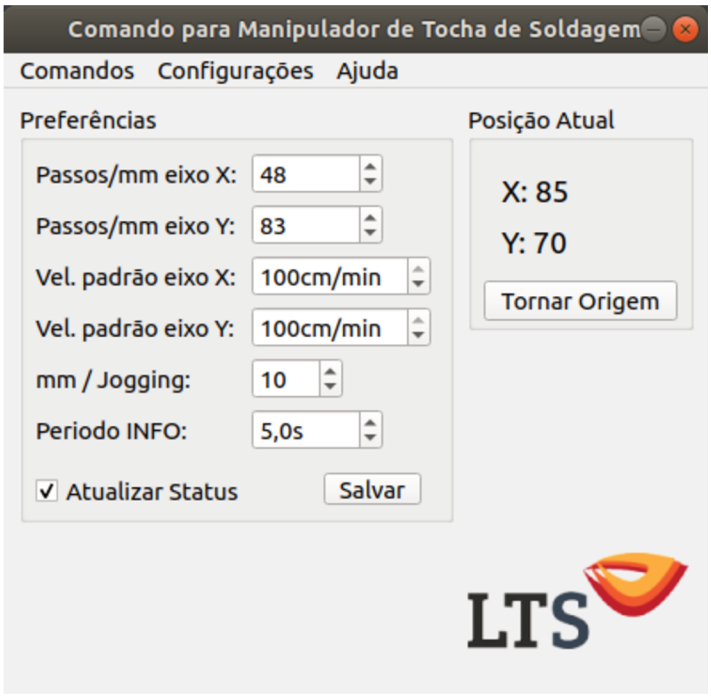
Fonte: Autor (2020).

Essa janela contém as principais funções para a utilização da máquina, separadas em caixas com seus respectivos títulos. A informação de status do motor e das posições do manipulador são atualizadas periodicamente, como determinado nos requisitos de projeto. Além das apresentações de informação, essa tela fornece quatro funcionalidades básicas ao sistema, entre elas a parada de emergência da máquina e as movimentações no estilo jogging, independente por eixo e o movimento linear no plano. O usuário pode acessar as configurações da interface ou a ajuda pela barra de menu superior.

A Figura 19 mostra a tela de configurações da IHM. No campo Preferências, o usuário pode modificar algumas variáveis de processo utilizadas para enviar comandos

à máquina e também configurar algumas características de funcionamento da interface.

Figura 19 – Interface de configurações.



Fonte: Autor (2020).

Seguindo a estrutura de comandos definida para a comunicação e execução de tarefas, as funções de movimentação linear e movimentação por eixo são apenas abstrações da mesma mensagem de movimentação definida entre o ESP32 e o Tiva C. Desse modo, para proporcionar uma interface que utiliza medidas de distância na apresentação dos comandos, é necessário determinar relações de equivalência entre números de passo e milímetros percorridos. As configurações de passos por milímetro são utilizadas para tornar a interface mais intuitiva com relação a isso. Outra configuração utilizada para simplificar a experiência de uso do operador da máquina é a velocidade padrão dos eixos, utilizada para configurar tanto a movimentação por eixo, quanto o jogging.

É possível configurar também a frequência com que as informações sobre o posicionamento do manipulador são atualizadas através da configuração do período e da caixa de seleção para atualização periódica do status.

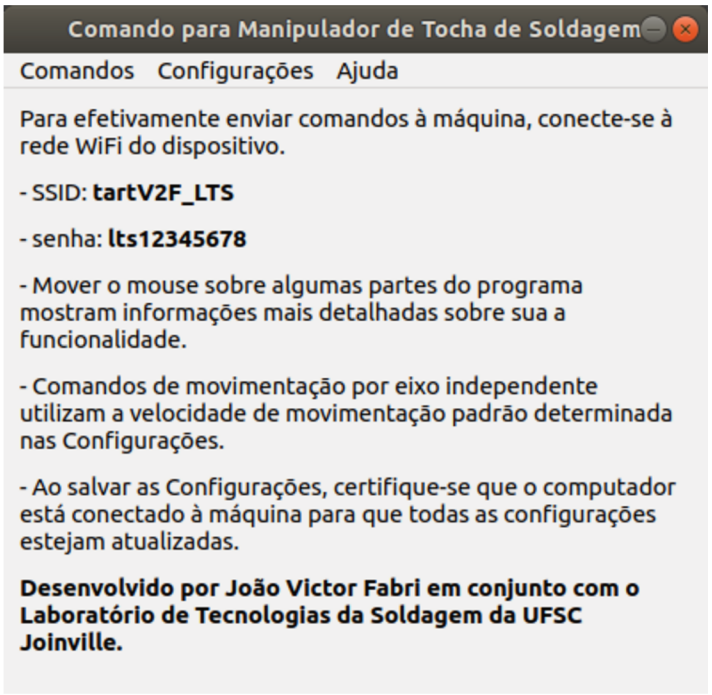
Apertar o botão “Salvar” abre um pop-up de aviso, pedindo a confirmação de que o usuário quer sobrescrever o arquivo de configurações atual ou criar um novo, caso não exista. Confirmar o salvamento das configurações envia um comando de configuração pra máquina com os números de passos por jogging para os eixos X e Y (utiliza a relação passo/mm e mm/jogging das configurações) e as velocidades padrão dos eixos.

Além das preferências, é possível visualizar a posição atual da máquina no

plano. Pressionar o botão "Tornar Origem"envia um comando de mudança da origem para a máquina e atualiza a posição na interface para (0,0).

Para finalizar o desenvolvimento da IHM, foi adicionado ao software uma aba de ajuda, com informações básicas da utilização do equipamento. Na Figura 20 é possível visualizar essa janela. A ajuda conta com algumas informações sobre a interface, já descritas nesse trabalho, bem como as informações sobre a rede WiFi do ponto de acesso do ESP32, e informações sobre o autor do projeto.

Figura 20 – Janela com informações gerais do software.



Fonte: Autor (2020).

Finalizado o projeto da interface gráfica da aplicação, foram adicionadas as funcionalidades aos elementos da IHM. Para tal, foram utilizadas as bibliotecas do API de gerenciamento de redes do QT, como a QNetwork, que permite gerenciar comunicações mais facilmente.

Para enviar requisições, o programador instancia um objeto do tipo QNetworkAccessManager, que conta com diversos métodos uteis para comunicação. É possível obter a resposta recebida conectando o sinal de finalizado a um slot de interpretação da resposta. No caso de requisições do tipo POST, é possível enviar um corpo com dados passando como argumento do método “post” um vetor de caracteres, tomando a precaução de modificar o cabeçalho do tipo de conteúdo enviado.

Um objeto do tipo QNetworkRequest pode ser utilizado para estruturar a requisição, recebendo o URL da mensagem no seu construtor e modificando o cabeçalho de tipo de conteúdo pelo método “setHeader”. A Figura 21 apresenta um fragmento de código (linhas 158 a 163) da função de envio de configuração para a

máquina.

Figura 21 – Fragmento da função de envio de configuração.

```
1    QNetworkAccessManager * mgr = new QNetworkAccessManager(this);
2    connect(mgr,SIGNAL(finished(QNetworkReply*)),this,SLOT(onfinish(
        QNetworkReply*)));
3    connect(mgr,SIGNAL(finished(QNetworkReply*)),mgr,SLOT(deleteLater()))
        ;
4    QNetworkRequest request(QUrl("http://192.168.4.1/CONFIG"));
5    request.setHeader(QNetworkRequest::ContentTypeHeader,"application/x-
        www-form-urlencoded");
6    mgr->post(request,data.toLocal8Bit());
```

Fonte: Autor (2020).

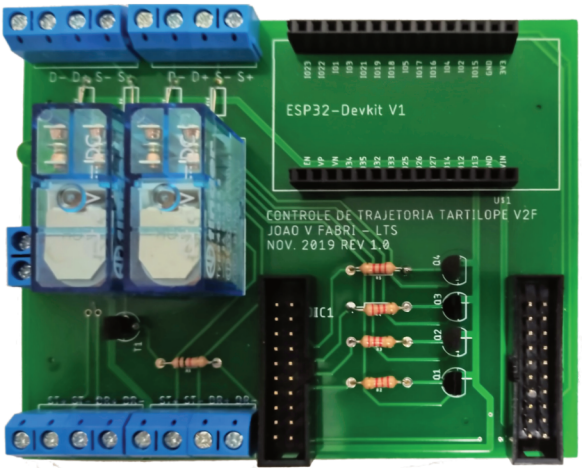
A mesma dinâmica foi utilizada nas funções de movimentação do manipulador, com as devidas modificações no URL da requisição e no campo de dados enviado. Para outros comandos que utilizam a requisição POST, mas não necessitam enviar dados, é possível omitir o cabeçalho do tipo de conteúdo e passar uma string vazia como argumento.

Para as requisições de informação, que são periodicamente enviadas pela interface, utiliza-se o método “get” do gerenciador de rede, que só recebe como argumento o objeto de requisição.

4 AVALIAÇÃO DO SISTEMA

Uma vez projetada a placa de acoplamento do sistema de comando, foi fabricada a PCB e soldados os componentes necessários. A Figura 22 mostra o resultado final, com os componentes soldados. Foram adicionados headers para o posterior encaixe das placas de desenvolvimento dos microcontroladores utilizados.

Figura 22 – Placa com componentes soldados.



Fonte: Autor (2020).

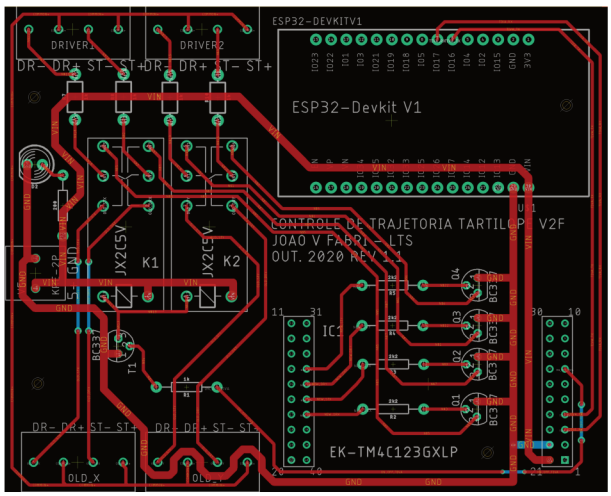
Com os componentes soldados, foram realizados testes de continuidade das trilhas da PCB, que não apresentaram nenhum defeito. No entanto, dois erros no design da placa foram identificados: o texto que identifica os sinais de entrada na placa (headers azuis da parte inferior da Figura 22) deveria ser igual ao texto das portas superiores e a trilha que deveria estar ligada à porta 29 do ESP32, está ligada à porta 28. A Figura 23 mostra uma nova versão do design da placa, com as devidas correções feitas.

De modo a corrigir o erro sem que fosse necessário produzir uma nova placa, as portas 28 e 29 do ESP32 foram ligadas por um fio. Isso implica que a porta 28 deve permanecer em estado lógico baixo durante a utilização da placa desenvolvida. Foram apagados os textos com os nomes das portas de entrada do sinal da máquina, de modo a evitar equívocos na instalação.

Antes de introduzir a placa de acoplamento no interior do equipamento, foram realizados testes para verificar o funcionamento dos sinais de controle dos drivers de motor. Utilizando um osciloscópio, foram medidos os sinais de tensão de saída, que apresentaram valores de tensão adequados ao funcionamento dos drivers. Foram também verificados os funcionamentos dos relês e a capacidade de enviar e receber os comandos WiFi programados.

Uma vez testada em bancada, a placa foi introduzida e fixada no equipamento,

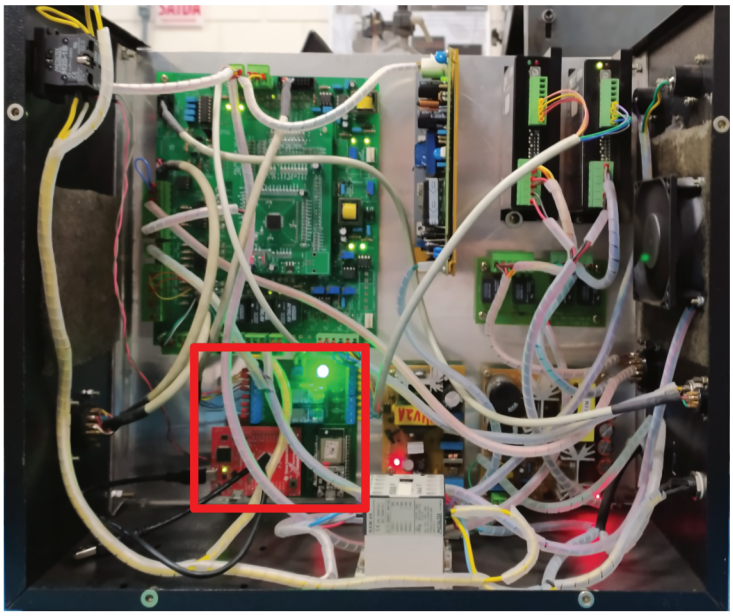
Figura 23 – Footprint da placa com correções.



Fonte: Autor (2020).

com suas conexões devidamente realizadas. A Figura 24 mostra o interior do Tartílope V2F, com a placa de controle desenvolvida destacada em vermelho. Foram acoplados os fios de sinal de saída do controlador antigo às portas de entrada da placa. Um novo cabo de 8 vias foi utilizado para ligar as portas de saída de sinal às entradas de comando dos drivers. Por fim, foi adicionado um cabo de alimentação ligado à entrada de 5V do controlador antigo e um LED soldado à porta PD2 do Tiva C.

Figura 24 – Interior do Tartílope V2F com a placa instalada em destaque.



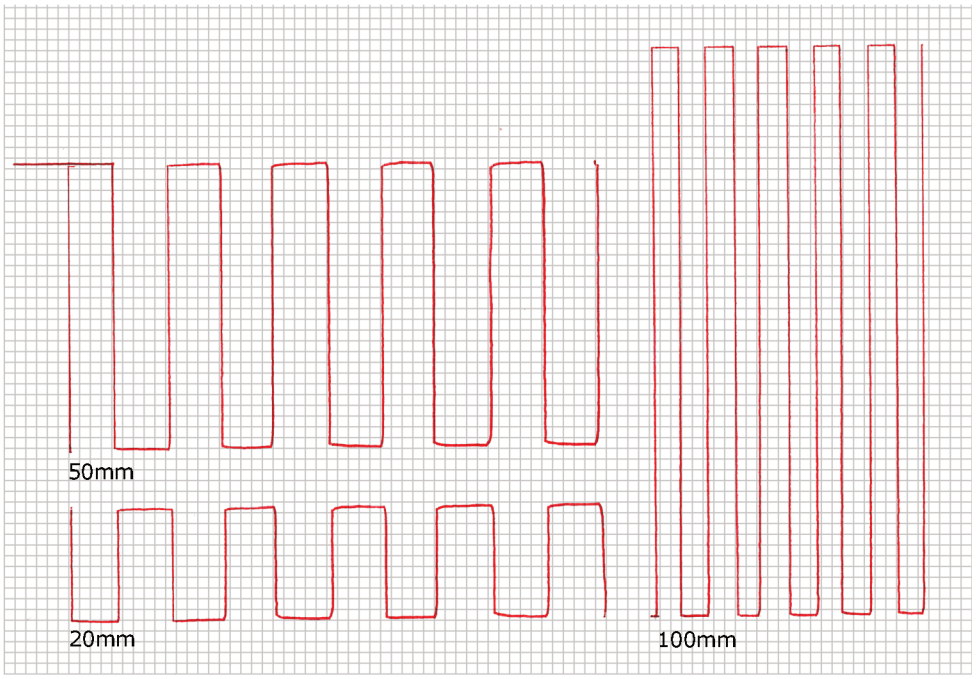
Fonte: Autor (2020).

Assim que foi verificado o funcionamento do chaveamento dos sistemas, iniciou-se a preparação dos testes de calibração. Para realizar as medições do deslocamento do manipulador foi fixada uma caneta esferográfica no punho do efetuador final, que

percorre uma folha de papel. Ao realizar as movimentações de comando, a caneta descreve o movimento realizado marcando o papel, de modo que a medição das linhas pode ser utilizada como uma descrição do percurso do manipulador.

Com uma velocidade fixa de 60cm/min foram realizados testes dos comando de movimentação para quatro distâncias diferentes (10, 20, 50 e 100mm), para ambos os eixos. Foram utilizadas as relações de passos por milímetro observadas nas configurações do equipamento, nos valores de 48 para o eixo X e 83 para o eixo Y. A Figura 25 mostra a folha de teste com os movimentos realizados para caracterizar o eixo Y da máquina em vermelho, com uma grade de 2mmx2mm e indicações da distância percorrida esperada adicionados digitalmente.

Figura 25 – Caracterização da movimentação do Eixo Y.



Fonte: Autor (2020).

Para cada uma das distâncias de controle foi realizada uma movimentação análoga à de uma onda quadrada, com a amplitude igual à distância de medição pertinente ao teste. Esse padrão pode ser observado na Figura 25. Após realizar o movimento repetidamente, foram medidas, utilizando-se um paquímetro, dez amostras para cada uma das distâncias.

Na Tabela 1 estão contidas as médias e desvios padrão para os movimentos realizados por cada eixo, em milímetros. É possível notar que o eixo X apresenta um erro consistente nas movimentações, o que pode ser ligado à utilização de uma relação de passos por milímetro inadequada ou às próprias características construtivas da máquina. O eixo Y, no entanto, não apresentou divergências significativas do valor esperado.

É válido notar também que a forma como foram medidas as distâncias

Tabela 1 – Caracterização da movimentação do Tartílope V2F (em mm).

Valor Esperado	Eixo X		Eixo Y	
	Média	Desvio-Padrão	Média	Desvio-Padrão
10mm	9.467	0.330	9.995	0.174
20mm	18.078	0.167	19.973	0.147
50mm	47.109	0.339	50.859	0.460
100mm	96.450	0.226	102.7	0.293

Fonte: Autor (2020).

percorridas pode inserir erros aleatórios e consistentes. Para minimizá-los, faz-se necessário adequar o ambiente de testes e utilizar formas de medição mais robustas. Devido à pandemia do COVID-19, em curso durante o desenvolvimento dos testes do equipamento, não foi possível realizar uma caracterização aprofundada da movimentação do equipamento, devido à possibilidade de acesso aos laboratórios estar reduzida.

Caracterizadas as funções de movimentação, o próximo passo foi verificar se o restante dos comandos do sistema funcionam satisfatoriamente. Considerando a limitação de que alguns destes só podem ser requisitados com a máquina parada, testes foram executados manualmente para determinar o funcionamento das funções implementadas.

Foram verificados os funcionamentos do botão de parada de emergência enquanto houvesse movimento, dos botões de jogging sendo pressionados continuamente, do envio de configuração durante movimento e da mudança de origem durante o movimento. Nenhum problema foi detectado durante os testes.

4.1 LIMITAÇÕES

Com a avaliação do funcionamento do sistema concluída, faz-se necessário entender algumas limitações do projeto. Dois pontos podem ser destacados: a atualização da informação de posição dos motores e o tempo de espera entre o envio do comando e o início da execução de movimentação.

Uma vez que a aplicação utiliza o protocolo HTTP para o envio e recebimento de mensagens, a máquina não tem autonomia para enviar mensagens à IHM sem antes receber uma requisição. Isso implica na necessidade de pedidos constantes de informação do sistema computadcional para a máquina para que seja mantida uma apresentação contínua da posição do manipulador no plano. No entanto, tendo em vista que esse controle de posição não precisa ser instantâneo e pode ter intervalos de poucos segundos de incerteza, como estabelecido na análise de requisitos do sistema, a utilização da comunicação estilo cliente-servidor não é empecilho para o bom funcionamento do sistema. Uma alternativa, caso seja necessário tornar essa

apresentação da posição mais próxima do tempo real, é utilizar outro protocolo de aplicação da comunicação sem fio, como o Bluetooth, observadas as restrições do ESP32, ou até mesmo substituir o sistema de comunicação.

A segunda limitação a ser destacada é o tempo de espera entre o envio do comando de movimentação e o início da execução. Ao contrário dos comandos que não carregam informação entre a interface e o Tiva C, comandos de movimentação e configuração tem um tempo de espera de 1 a 2 segundos entre o momento do envio do comando pela IHM até o início do movimento. Apesar de curta, a espera é uma limitação a ser considerada. É válido, portanto, entender o processo envolvido no envio, transporte e leitura dos comandos da máquina, uma vez que esses processos estão diretamente ligados com o tempo de espera do equipamento.

A manipulação dos dados das mensagens, tanto no ESP32 quanto no Tiva C, pode ser apontada como um dos gargalos na execução da tarefa. Sendo necessário enviar quatro valores inteiros de 4 bytes cada por dois sistemas diferentes, o processo de leitura e conversão desses dados introduz um atraso curto, porém considerável, na execução das rotinas dos comandos.

Para ilustrar, podemos comparar os códigos das funções de manuseio dos comandos de jogging e de movimentação do ESP32, presentes nas Figuras 26 e 27. A primeira função tem quatro linhas de código, sendo a segunda e a terceira responsáveis pelo envio da mensagem. O tempo de espera entre o recebimento do pacote HTTP e o envio da mensagem UART para o Tiva C é quase nulo.

Figura 26 – Função de envio do comando Jogging em X positivo.

```
1 void handle_jogxp(){
2     Serial.println("JogX+");
3     Serial2.write(JOGXp);
4     Serial2.write(0xFF);
5     server.send(200,"text/html", sendHTTPresponse("JOGX+",JOGXp));
6 }
```

Fonte: Autor (2020).

A segunda função, no entanto, conta com 6 linhas de código a mais, que ocupam pelo menos 2 ciclos de processamento cada uma (acesso e comparação de variáveis). Ainda existe o tempo de envio da mensagem completa, que tem 16 bytes a mais que a da primeira função. Para diminuir o tempo de latência entre o envio do comando e o início da movimentação, pode-se otimizar as funções da troca de mensagem para que menos transformações sejam feitas e menores sejam as mensagens trocadas. É possível considerar também a redução do tamanho dos dados trocados, considerados os limites da máquina em números de passos que podem ser dados ou velocidades que são comumente utilizadas.

Outro empecilho para a redução do tempo de resposta dos comandos de

Figura 27 – Fragmento da função de comando de movimentação.

```
1 void handle_command(){
2   Serial.println("COMMAND");
3   union u32tou8 x,y,xv,yv;
4   String order ="x=" + server.arg("x") +"&y=" + server.arg("y")+"&xv=" +
        server.arg("xv")+"&yv=" + server.arg("yv");
5   if(server.hasArg("x"))
6     x.u32integer = (uint32_t) server.arg("x").toInt();
7   if(server.hasArg("y"))
8     y.u32integer = (uint32_t) server.arg("y").toInt();
9   if(server.hasArg("xv"))
10    xv.u32integer =  server.arg("xv").toInt();
11  if(server.hasArg("yv"))
12    yv.u32integer =  server.arg("yv").toInt();
13  Serial2.write(COMMAND);
14  send_value(x);
15  send_value(xv);
16  send_value(y);
17  send_value(yv);
18  Serial2.write(0xFF);
19  server.send(200,"text/html", sendHTTPresponse(order, COMMAND, x.
       u32integer, y.u32integer));
20 }
```

Fonte: Autor (2020).

movimentação é o recebimento das mensagens pelo Tiva C, que são lidas byte a byte no laço principal da programação apresentado na Figura 28, com cada leitura seguida da chamada da função “SysCtlDelay”, que bloqueia a execução do código por um número de ciclos, nesse caso gerando um atraso de $0,0333$ segundos por byte lido. Uma mensagem de 16 bytes, portanto, atrasa o início da movimentação em aproximadamente $0,53$ segundos.

Figura 28 – Laço de leitura do buffer da UART.

```
1 while (UARTCharsAvail(UART1_BASE) ) {
2     words[i] = UARTCharGet(UART1_BASE) ;
3     i++ ;
4     SysCtlDelay(SysCtlClockGet( ) / 30) ;
5 }
```

Fonte: Autor (2020).

É possível identificar que sem a espera entre as leituras, existe a perda de informação do comando recebido, uma vez que o ciclo de leitura e escrita do byte recebido acaba sendo mais rápido do que o ciclo de envio por parte do ESP32. Quando isso ocorre, o Tiva C identifica que o buffer de leitura do canal UART está vazio e segue para a identificação do comando recebido. Fica evidente, portanto, que aplicar

a utilização de um byte de fim de mensagem pode ser suficiente para eliminar a necessidade do atraso na leitura dos dados recebidos, reduzindo o tempo de espera entre o comando e a execução da tarefa.

5 CONCLUSÕES

O objetivo principal desse projeto foi o de desenvolver uma interface de programação de tarefas de soldagem alternativa à presente na máquina em estudo, o Tartílope V2F (SPS, 2012). Contando com uma interface de programação de tarefas difícil de manusear, foi determinada a necessidade de tornar a utilização do manipulador cartesiano mais simples e intuitiva.

Para isso, foram elencados os requisitos do projeto com base nas necessidades levantadas pelo coordenador do Laboratório de Tecnologias da Soldagem do campus Joinville. Para isso, foram desenvolvidos uma placa micro-controlada capaz de alternar entre os sistemas de comando e uma interface de programação de tarefas.

Uma vez projetados, construídos e testados, foi possível atingir os requisitos estipulados no início do trabalho. Oferecendo uma interface intuitiva e passível de modificações, o novo sistema de comandos do Tartílope V2F atingiu as expectativas depositadas sobre o projeto.

O novo sistema de comando conta com uma aplicação com funções apresentadas em uma interface intuitiva e capaz de comunicar ordens ao equipamento por conexão WiFi. Permite a movimentação do manipulador e o posicionamento no plano com um sistema de localização interno. Além disso, o desenvolvimento deste trabalho resultou em um sistema que permite a adição de novas funcionalidades à máquina, ao passo que é desenvolvido em linguagens de programação e microcontroladores acessíveis e conhecidos pela comunidade acadêmica.

É evidente, no entanto, que existem melhorias a serem feitas e novas funcionalidades que podem ser implementadas futuramente no equipamento. Além de corrigir as limitações já discutidas no Capítulo 4, algumas sugestões para trabalhos futuros são:

- Criar um aplicativo móvel para controle da máquina;
- Analisar e otimizar os tempos de execução e resposta das tarefas;
- Otimizar o processamento das tarefas pelos microcontroladores;
- Implementar comandos complexos de tecimento;
- Melhorar a interface visual e a experiência de usuário da IHM;
- Generalizar o sistema para qualquer máquina controlada por motores de passo;
- Utilizar codificações conhecidas como o código G (ISO, 2009) nas comunicações e na execução de comandos;
- Ampliar as funções de comunicação em rede da máquina, implantando sistema de comando remoto, por exemplo, conectando a máquina à Internet.

REFERÊNCIAS

BELL, C.; MUDGE, J.; MCNAMARA, J. **Computer Engineering**: A dec view of hardware systems design. [S.l.]: Digital Press, 2014.

CONDIT, R. **Stepping Motors Fundamentals**. [S.l.]: Microchip Technology Inc., 2004. https://www.microchip.com/wwwAppNotes/AppNotes.aspx?appnote=en012151.

FOROUZAN, B. **Comunicação de Dados e Redes de Computadores**. 4. ed. Porto Alegre: McGraw Hill Brasil, 2009.

GROKHOTKOV, I. **WebServer.h**. 2014. https://github.com/espressif/arduino-esp32/blob/master/libraries/WebServer/src/WebServer.h.

GROKHOTKOV, I. **WiFi.h**. 2014. https://github.com/espressif/arduino-esp32/blob/master/libraries/WiFi/src/WiFi.h.

ISO. **ISO 6983 - Numeric control of machines-Program format and definition of address words-Part 1**: Data format for positioning, line motion and contouring control systems. Geneva, Suíça: International Organization for Standardization, 2009.

ISO. **ISO/IEC 14882:2014 Information technology — Programming languages — C++**. 4. ed. Geneva, Suíça: International Organization for Standardization, 2014.

KUROSE, J. F.; ROSS, K. W. **Redes de computadores e a Internet**: uma abordagem top-down. 5. ed. São Paulo: Addison Wesley, 2010.

LABSOLDA. **Manual do Equipamento Tartílope V2-V2F**. 2012. http://www.labsolda.ufsc.br/publicacoes/manuais-equipamentos/tartilope_v2-v2f_manual_usuario_(2012).pdf.

MODENESI, P.; MARQUES, P.; BRACARENSE, A. **Soldagem**: fundamentos e tecnologia. Belo Horizonte: UFMG, 2005.

NORRISH, J. **Advanced welding process**. Londres: Elsevier Science, 2006.

OGATA, K. **Engenharia de controle moderno**. São Paulo: Pearson Prentice Hall, 2010.

PIRES, J.; LOUREIRO, A.; BOLMSJÖ, G. **Welding robots**: technology, system issues and application. Londres: Springer London, 2006.

SPS. **Tartílope V2F**. 2012. https://www.sps-soldagem.com.br/tartilope_v2f.php.

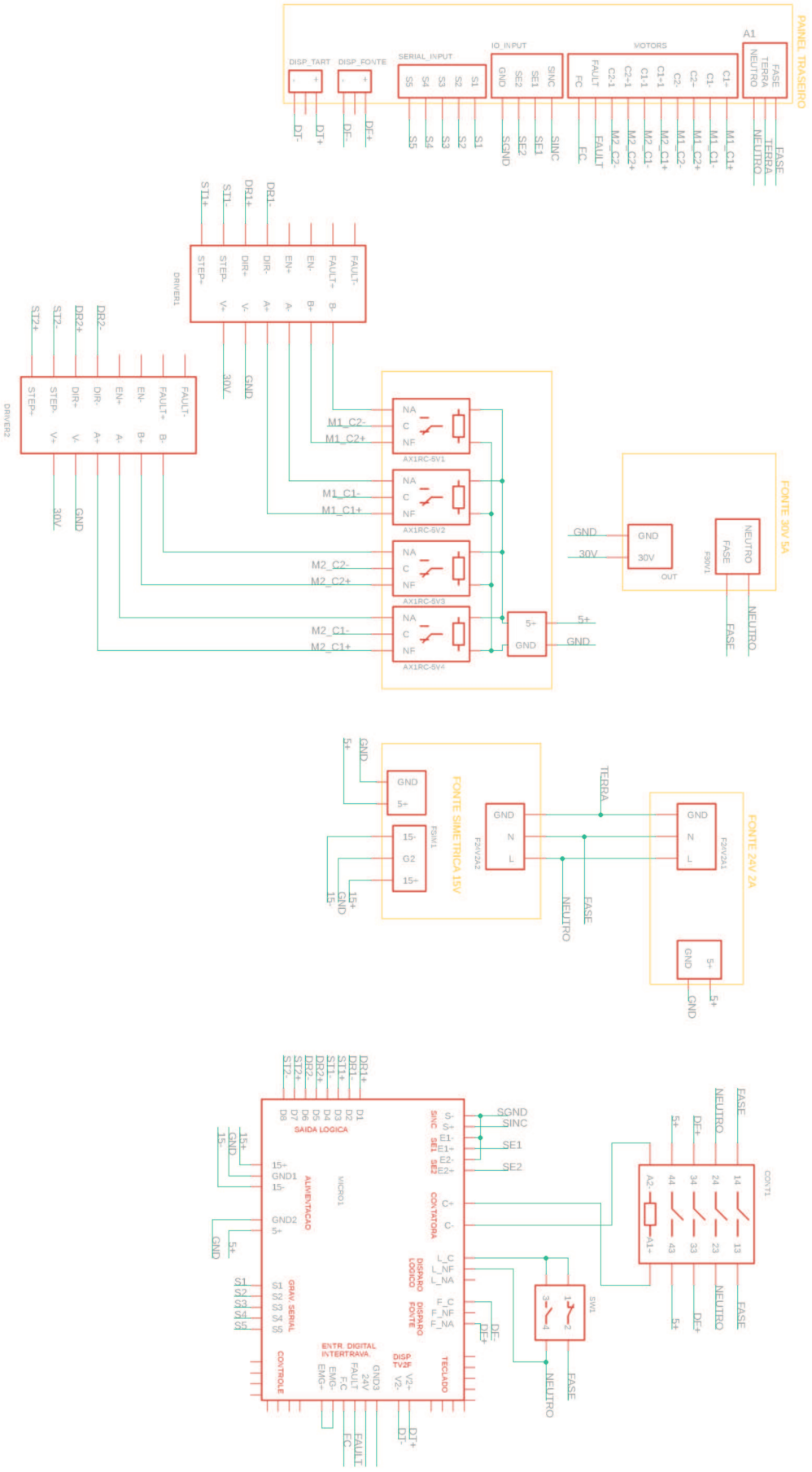
Texas Instruments. **Tiva TM4C123GH6PM Microcontroller Datasheet**. 2014. https://www.ti.com/lit/pdf/SPMS376E.

The QT Company. **QT Framework**. 2014. https://www.qt.io/.

UMANS, S. **Máquinas Elétricas de Fitzgerald e Kingsley**. 7. ed. Porto Alegre: AMGH, 2014.

WAINER, E.; BRANDI, S.; MELLO, F. de. **Soldagem**: processos e metalurgia. São Paulo: Edgard Blücher, 1992.

A - ESQUEMÁTICO TARTÍLOPE V2F



B - ESQUEMÁTICO PLACA DE COMANDO