VICTOR SANTOS UGRINOVICH

ADEQUAÇÃO DE ROBÔ CARTESIANO PARA TRABALHAR UTILIZANDO PNRD

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA MECÂNICA
2022

VICTOR SANTOS UGRINOVICH

ADEQUAÇÃO DE ROBÔ CARTESIANO PARA TRABALHAR UTILIZANDO PNRD

Trabalho apresentado ao Curso de Graduação em Engenharia Mecatrônica da Universidade Federal de Uberlândia, com parte dos requisitos para a obtenção do título de BACHAREL EM ENGENHARIA MECATRÔNICA.

Orientador: Prof. Dr. José Jean-Paul Zanlucchi de Souza Tavares

Uberlândia – MG

2022

Agradecimentos

À minha mãe Ana Roberta Silva Santos Ugrinovich, ao meu pai Reinaldo Ugrinovich e minha irmã Carolina Santos Ugrinovich por todo apoio e compreensão durante o curso e por sempre acreditarem em mim e me incentivarem.

À minha namorada Giulia Vanzo, pelo apoio, paciência e incentivo durante os momentos difíceis e por me fazer enxergar sempre as situações com mais otimismo.

A todos os meus amigos e familiares que direta ou indiretamente contribuíram para minha formação, com conselhos, palavras de conforto e apoio incondicional.

Ao Prof. Dr. José Jean-Paul Zanlucchi de Souza Tavares pela orientação e oportunidade de poder trabalhar em um projeto que me agregou bastante aprendizado e descobertas.

E, finalmente, à Universidade Federal de Uberlândia e à Faculdade de Engenharia Mecânica pela oportunidade de participar do curso de Engenharia Mecatrônica e por toda a infraestrutura oferecida durante estes anos como aluno.

Resumo

A rede de Petri PNRD possibilita a identificação de um produto e também permite realizar uma atualização do estado do item com relação ao processo esperado em tempo real. Para prova de conceito em uso industrial se pretende implementar tal sistema em um robô cartesiano, amplamente utilizado neste tipo de ambiente. Neste trabalho é realizado a adequação de um robô cartesiano para trabalhar com a rede PNRD. Utilizando uma garra robótica e um sensor RFID acoplado à garra, sendo o sensor lido por um Arduino Uno diretamente conectado ao computador principal em que o robô cartesiano está conectado. O robô cartesiano é controlado por um arduino Mega conectado à uma Shield Ramps 1.4, que converte os comandos do arduino Mega em sinal elétrico, assim realizando a movimentação do robô.

Palavras-chave: robô cartesiano, PNRD, IPNRD, RFID, Arduino Uno, Arduino Mega, Shield Ramps.

Abstract

The PNRD Petri net makes it possible to identify a product and also allows an update of the item's status in relation to the expected process in real time. For proof of concept in industrial use, it is intended to implement such a system in a Cartesian robot, widely used in this type of environment. In this project, the adequacy of a Cartesian robot to work with the PNRD network is carried out. Using a robotic gripper and an RFID sensor coupled to the grip, the sensor being controlled by an Arduino Uno directly connected to the main computer where the Cartesian robot is connected. The Cartesian robot is controlled by an Arduino Mega connected to a Shield Ramps 1.4, which converts the Arduino Mega commands into an electrical signal, thus making the robot move.

Keywords: Cartesian robot, PNRD, IPNRD, RFID, Arduino Uno, Arduino Mega, Shield Ramps.

Lista de Figuras

Glossário

RFID – Identificação por radiofrequência (*Radio-Frequency IDentification*)

RAMPS – Escudo RepRap Arduino Mega Pololu (*RepRap Arduino Mega Pololu Shield*)

PNRD – Redes de Petri Inseridas em Base de dados RFID (*elementary Petri Net inside a RFID distributed Database*)

Conteúdo

1 Introdução

A implementação de robôs nas industrias ampliou a quantidade deste tipo de maquinário no mundo todo (JM Rosario, 2010). Visando a ideia de fazer algo que seja tanto interessante para o meio acadêmico, quanto para a industria, o trabalho proposto visa adequar um robô cartesiano, caracterizado pela sua simplicidade e praticidade, para trabalhar com a Rede de Petri inserida em base de dados RFID (PNRD), utilizando um software de código aberto, adaptado para a realidade proposta, ou seja, utilizar um robô para detectar objetos e suas posições iniciais, e com isto, posicioná-los da maneira que for escolhida.

O robô cartesiano se caracteriza por permitir apenas o deslocamento linear do elemento terminal, nos eixos X, Y e Z. Esse tipo de configuração de maquinário é muito utilizado devido às seguintes características:

- Operação simples;
- Custo baixo de montagem;
- Facilidade de programar seus movimentos;
- Amplamente personalizável;

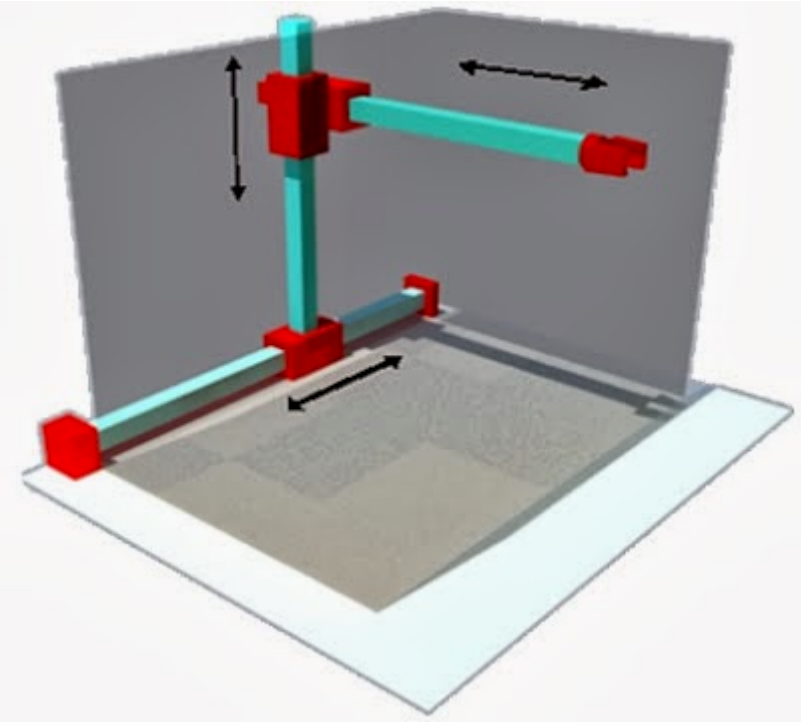

Figura 1: Esquema 3D de um Robô Cartesiano.

Além disso, o robô utilizado neste projeto não é novo, ele foi projetado e desenvolvido por Rodrigues (2017), e adaptado por Oliveira (2020) para suportar um sistema de corte laser. Isto reforça ainda mais as características apresentadas acima, mostrando ser amplamente personalizável, de fácil adaptação e é uma máquina que poderá ser utilizada posteriormente em uma ampla quantidade de projetos distintos.

O projeto a seguir contará com: objetivos, justificativa, fundamentação teórica, onde será realizada uma breve revisão bibliográfica acerca do tema, desenvolvimento, resultados e discussões, conclusão e trabalhos futuros e referências bibliográficas.

2 Objetivos

2.1 Objetivo Geral

Projetar a adaptação de um robô cartesiano previamente existente para trabalhar utilizando um sistema PNRD para solução do problema do Mundo de Blocos.

2.2 Objetivos Específicos

Levando em consideração o objetivo geral apresentado e o desenvolvimento do projeto, os seguintes objetivos específicos podem ser destacados:

- Preparar o robô para estar totalmente funcional;
- Realizar a leitura das tags dos blocos;
- Controlar o robô independentemente (através do software);
- Criar um software que faça a integração entre a leitura das tags e a movimentação automática do robô;

3 Justificativa

Em períodos de tempo cada vez mais curtos, se faz necessário adaptar os métodos de produção para continuar sendo competitivo no mercado. Neste contexto é necessário apresentar soluções de baixo custo e boa performance, e é neste sentido que este trabalho opera. Aliando um robô cartesiano, que é um robô de arranjo mecânico simples e de fácil montagem, com um sistema PNRD, que possui uma técnica de planejamento, modelagem e controle de boa performance, é possível criar sistemas robóticos que serão modernos, eficientes e de custo reduzido.

4 Fundamentação Teórica

4.1 Mundo de Blocos

O primeiro passo para compreender este trabalho, é ter conhecimento do mundo de blocos. O mundo de blocos consiste em um número finito de blocos sobrepostos em uma plataforma, e o objetivo é mover os blocos de sua posição inicial para uma posição desejada. Para este problema sugerido, algumas regras devem ser estabelecidas:

- Somente um bloco pode ser movido por vez.
- A representação estabelecida é em duas dimensões, ou seja, um bloco não pode ser empilhado em cima do outro.

Neste projeto será utilizado um exemplo com três blocos, que podem ser distri- buídos em três colunas e três linhas. Ao se discretizar o espaço disponível em posições, temos uma representação do espaço de trabalho proposto em três linhas e três colunas, conforme a Figura 2.

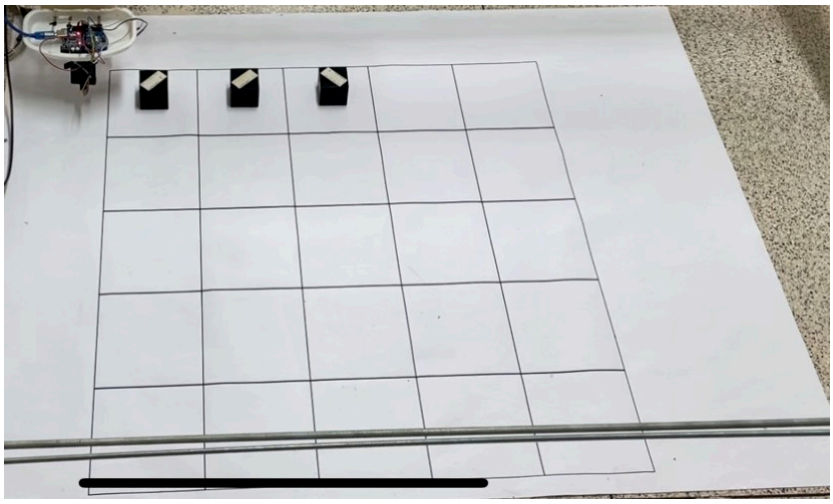

Figura 2: Fotografia do espaço de trabalho do robô.

4.2 PNRD

PNRD é a sigla para Rede de Petri inserida em base de dados RFID, e de acordo com Tavares e Saraiva (2010) e Aalst (1998), redes de Petri são excelentes ferramentas para modelagem e análise de processos. As redes de Petri podem ser usadas para especificar e exemplificar utilizando a linguagem gráfica, fluxos de trabalho complexos, aliado à isso, a teoria das redes de Petri fornece poderosas técnicas de analise que podem ser usadas para identificar as falhas dos procedimentos de fluxos de trabalho. Inseri-las em base de dados RFID é o primeiro passo para melhorar a integração de sistemas de controle com a tecnologia RFID.

Para Tavares e Saraiva (2010), o objeto é apenas uma ficha dentro da rede de Petri que representa o modelo de um processo no qual este objeto está inserido. Todo o planejamento de processo pode ser armazenado em uma etiqueta RFID na forma de uma matriz de incidência e ela representará o plano de processo, esta abordagem inclui até mesmo processos flexíveis, que é o fundamento da

abordagem PNRD. Além disto, a etiqueta também terá a informação de onde ela se encontra dentro do processo, para isto, é armazenado também o vetor de marcação Mk. Dependendo da memória da etiqueta, também é possível a adição de informações adicionais para atender as necessidades de cada processo.

Além disso também é necessário que existam leitores RFID nos pontos de transição, são eles carregam o vetor de disparos (uk) referente àquela transição. De forma que ao identificar uma etiqueta, automaticamente este leitor faz o cálculo do vetor Mk+1 e o salva na própria tag RFID.

A PNRD é um ótimo recurso para o controle de processos, pois é um metodo que possibilita identificar erros no processo. Quando ocorre um erro, o mesmo é identificado imediatamente pelo leitor RFID, pois haverá o surgimento de um valor negativo no vetor de marcações. É possível exemplificar graficamente utilizando a Figura 3:

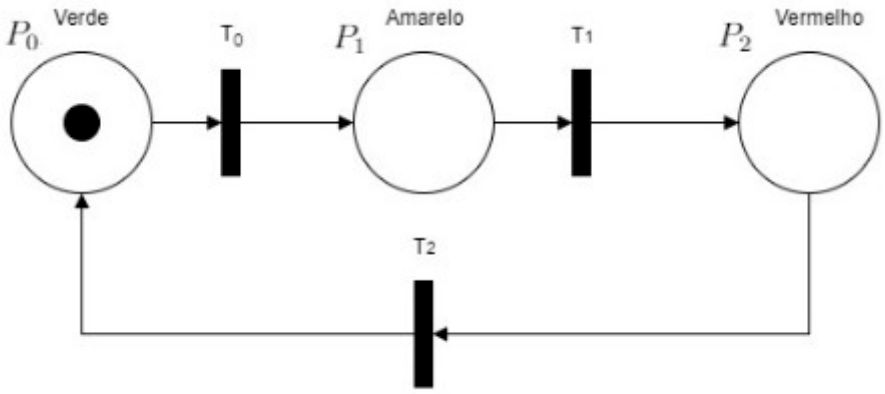

Figura 3: Exemplo de rede de Petri representando o funcionamento de um semáforo

Fonte: Silva et al. (2017)

4.3 IPNRD

Segundo Fonseca et al. (2018) a PNRD tem como objetivo o monitoramento de agentes passivos, como itens comerciais, peças e produtos. Informações acerca do modelo comportamental (matriz de incidência AT e vetor de marcação Mk) são armazenados em uma etiqueta passiva fixada ao objeto em questão, enquanto as informações referentes ao vetor de disparos uk estão associadas aos leitores e antenas distribuídos no ambiente.

Em situações de busca e salvamento, os agentes que requerem identificação são ativos, como usuários, vítimas e socorristas. Neste caso percebe-se que a estrutura clássica da PNRD é incapaz de controlar o comportamento dos mesmos, isso porque agentes ativos possuem comportamento autônomo e seus modelos comportamentais devem ir além do controle de movimentação ao longo dos pontos estratégicos.

Como os agentes ativos são normalmente microcrontrolados, torna-se inviável associar sua identificação a uma etiqueta, sendo mais razoável a utilização destas para a identificação de pontos de interesse no espaço, além do mais, em ambientes remotos existe a dificuldade de manter fontes de energias para os sensores, enquanto os sensores podem ser facilmente acoplados aos agentes ativos.

Esta é a base para a definição da iPNRD, nela, há uma inversão dos locais de armazenamento dos termos da Equação 2.1. Nesta abordagem, a matriz de incidência AT e o vetor de marcações Mk são armazenados no leitor, e na etiqueta são armazenadas informações do vetor de disparo uk, conforme apresentado na Figura 6.

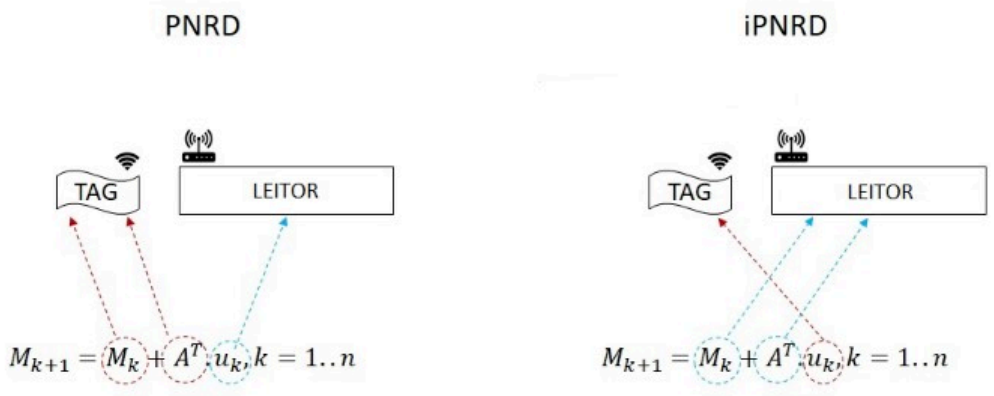

Figura 4: Associação dos termos da Equação 2.1 com os elementos onde serão armazenados nas abordagens PNRD e iPNRD.

Fonte: Fonseca et al. (2018)

4.4 Robô Cartesiano

Um robô cartesiano é uma máquina cujos três principais eixos de controle são lineares e perpendiculares entre si.

A localização da garra é feita em relação as coordenadas X, Y e Z configurado como referência.

Essa configuração estrutural é utilizada em equipamentos de diversas áreas trazendo segurança e qualidade. Cada eixo pode ser entendido como uma junta prismática, e portanto esses robôs são do tipo PPP pela classificação da robótica.(VMF Santos, 2004)

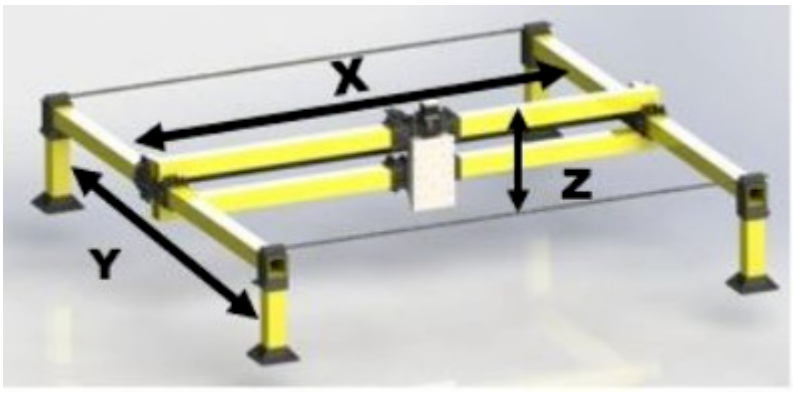

Figura 5: Fotografia do robô cartesiano utilizado.

Normalmente o controle desse tipo de robô é simples, uma vez que a movimentação em cada eixo representa uma relação direta no movimento do elemento terminal do robô.

4.5 Leitor RFID e tags

A tecnologia RFID baseia-se no princípio de transmissão/recepção de dados via ondas eletromagnéticas de rádio,o sistema é composto basicamente por três componentes conforme mostra a Figura 6, sendo o tag, um dispositivo que ao ser excitado por um campo eletromagnético provido pelo reader, terá seu circuito energizado e passará a enviar as informações previamente gravadas em sua memória, o reader, dispositivo que ao receber os dados de uma tag, decodifica suas informações, e uma interface de aplicação a qual será responsável por executar uma ação conforme os dados recebidos pelo reader (SILVEIRA;LEITE, 2016).

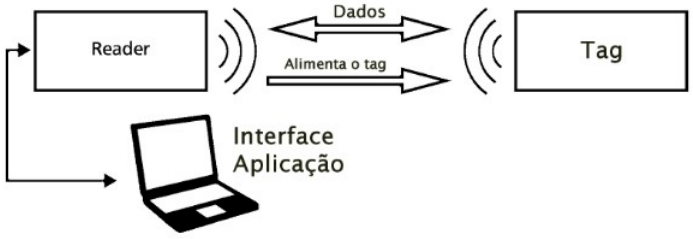

Figura 6: Diagrama de funcionamento de um sistema RFID.

Fonte: Hellermann Tyton(2017)

4.6 *Firmware* GRBL for RAMPS 1.4

O *firmware* GRBL for RAMPS 1.4 é um analisador de código G de código open-source, incorporado, de alto desempenho e faz a tradução do comando enviado pelo Arduino Mega para a *shield* Ramps conectada à ele.

O GRBL é um firmware desenvolvido para interpretar os códigos G, linguagem de máquina universal utilizada em equipamentos com base em robô cartesiano.

4.7 Arduino Mega

O Arduino Mega possui a finalidade de fazer a comunicação entre o computador e os motores acoplados ao robô, é ele que recebe o comando do software do computador e manda o sinal aos motores, os fazendo movimentar.

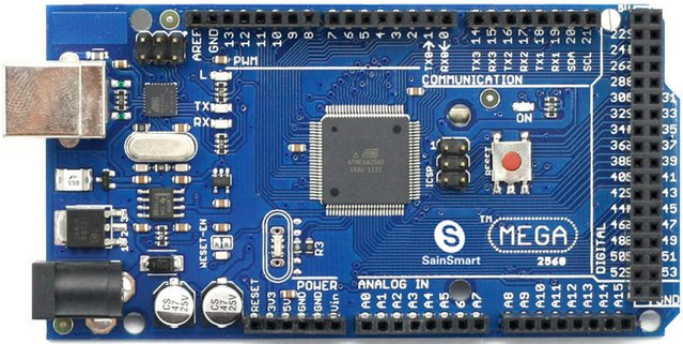

Figura 7: Arduino mega.

Fonte: https://www.arduino.cc/

4.8 Arduino Uno

O Arduino Uno possui a finalidade de fazer a comunicação entre o leitor RFID acoplado à garra robótica e o computador, é ele que envia ao software os dados contidos nas tags RFID, para que o software possa utilizar estes dados para calcular os movimentos que a garra robótica deverá fazer, a fim de movimentar cada bloco.

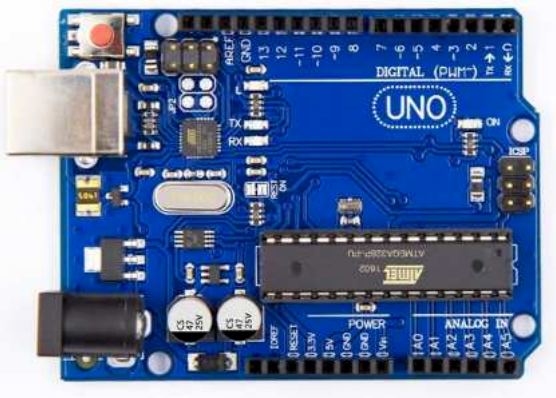

Figura 8: Arduino mega.

Fonte: https://www.arduino.cc/

4.9 RAMPS 1.4

A shield RAMPS 1.4 é quem faz o gerenciamento energético e das informações enviadas pelo Arduino Mega através dos pinos de contole: Se o Arduino recebe o comando para mover um eixo XYZ do robô, é a RAMPS quem vai acionar o driver do respectivo motor (e consequentemente o motor em si).

Figura 9: Módulo shield RAMPS 1.4.

Fonte: https://proesi.com.br/modulo-shield-ramps-1-4.html

5 Desenvolvimento

A adaptação do robô cartesiano presente no Laboratório de Ensino à Mecatrônica, consiste na preparação da movimentação do robô cartesiano utilizando o *firmware* GRBL for RAMPS 1.4, adaptação de uma garra robótica ao eixo Z do robô cartesiano, adaptação de um leitor RFID à esta garra robótica, ambos conectados em um arduino UNO e na criação de um *software* em C que faça a integração do Arduino Mega conectado ao robô cartesiano ao Arduino Uno conectado ao leitor RFID, utilizando a biblitoca do *software* Universal Gcode Sender.

5.1 Robô Cartesiano

O robô cartesiano utilizado no projeto é um exemplar existente no Laboratório de Ensino a Mecatrônica que já fora utilizado em outros projetos passados (Rodrigues, 2017 e Oliveira, 2020). Este aparelho possui as seguintes especificações: volume de trabalho de 1100x1100x150mm (X-Y-Z), multifuncionalidade e baixo custo.

Nele está conectado o Arduino Mega, que por sua vez possui uma shiled Ramps 1.4 conectada no próprio Arduino, e o o Arduino por sua vez está conectado via USB à um computador, sendo o software aberto no computador responsável por ditar a movimentação do robô.

Figura 10: Fotografia do robô cartesiano utilizado.

Para a alimentação do robô cartesiano, foi utilizada uma fonte de 500W real, que foi conectada à shield RAMPS 1.4, utilizando o conector molex existente na placa, energizando assim os motores do aparelho.

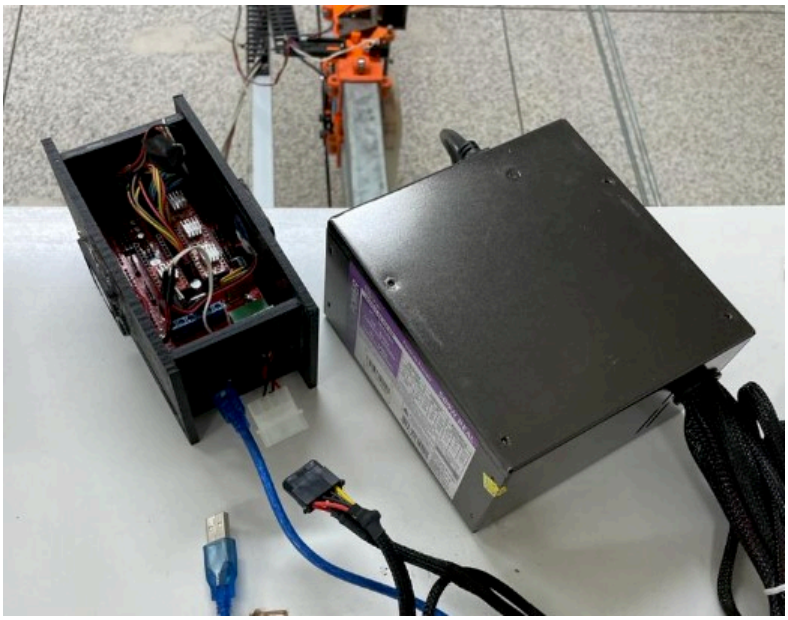

Figura 11: Fonte utilizada para alimentar o robô.

5.2 *Firmware* GRBL for RAMPS 1.4

O *firmware* GRBL for RAMPS 1.4 foi instalado no Arduino Mega, que possuia a placa RAMPS 1.4 previamente conectada à ele (Oliveira, 2020), seguindo as orientações e passo-a-passo de seu website oficial.

5.3 Arduino Mega

O Arduino Mega já possuia a placa RAMPS 1.4 previamente conetada à ele, bem como todas as conexões entre fios necessárias para o funcionamento do robô cartesiano (Oliveira, 2020). Com o Firmware GRBL for RAMPS 1.4 instalado nele e a shield RAMPS 1.4 também, não é necessário realizar mais nenhuma modificação.

5.4 Arduino Uno

O Arduino Uno utilizado foi encaixado no topo de uma cesta de plástico, para que ficasse firme e pudesse se conectar o leitor RFID, como é possível observar na Figura 12.

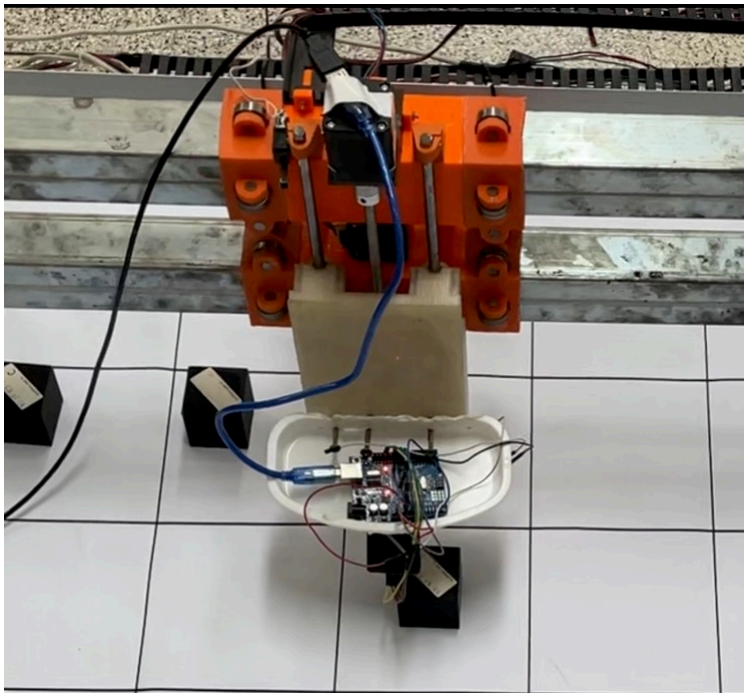

Figura 12: Arduino Uno conectado ao leitor RFID do robô.

Neste Arduino Uno foi inserido um código adaptado do website https://electropeak.com, que se encontra no Apêndice A.

5.5 RAMPS 1.4

A shield RAMPS 1.4 é quem faz o gerenciamento energético e das informações enviadas pelo Arduino Mega através dos pinos de contole: Se o Arduino recebe o comando para mover um eixo XYZ do robô, é a RAMPS quem vai acionar o driver do respectivo motor (e consequentemente o motor em si).

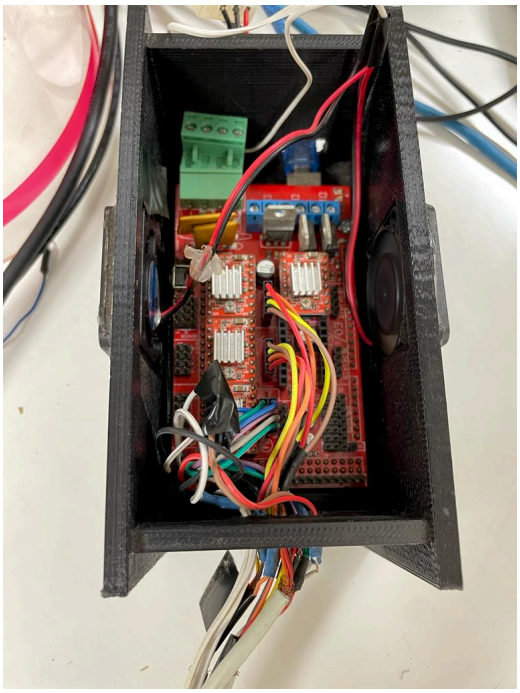

Figura 13: shield RAMPS 1.4 e suas conexões.

5.6 Garra Robótica

A garra robótica acoplada ao eixo Z deste robô cartesiano, assim como o próprio robô, é uma peça aproveitada de um outro projeto desenvolvido no Laboratório.

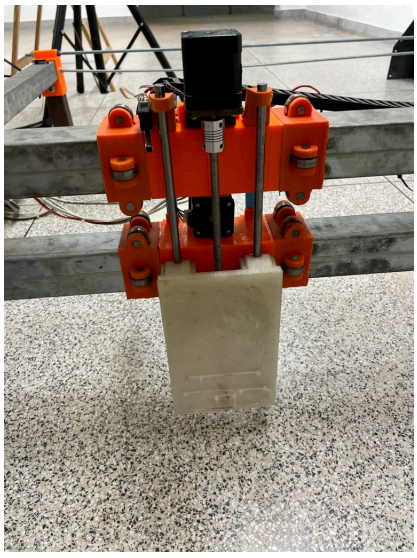

Figura 14: Eixo Z antes de acoplar a garra robótica.

Para acoplar a garra robótica ao eixo Z foram utilizados parafusos e uma cesta de plástico, para que o Arduino Uno que controla o leitor de RFID fique dentro da cesta, e a garra robótica foi presa com cola quente na parte inferior da cesta de plástico.

Figura 15: Eixo Z após acoplar a garra robótica.

A sua função é de realizar a movimentação dos blocos, a garra robótica irá até cada bloco e irá recolher o mesmo e levar para a posição desejada.

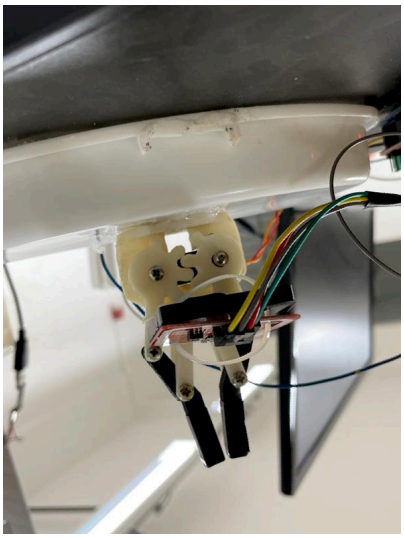

Figura 16: Garra robótica vista de frente.

5.7 Leitor RFID

O Leitor RFID (Identificação por Radiofrequência) também vem do mesmo projeto da garra robótica (Pires, 2019), portanto ele já estava acoplado à garra robótica desde o ínicio do desenvolvimento deste projeto. Ele é responsável por realizar a leitura das tags que estão fixadas em cada bloco e contém a informação de posição inicial e desejada de cada bloco. Através do software desenvolvido, é possível, atráves do leitor RFID, coletar a informação destas tags.

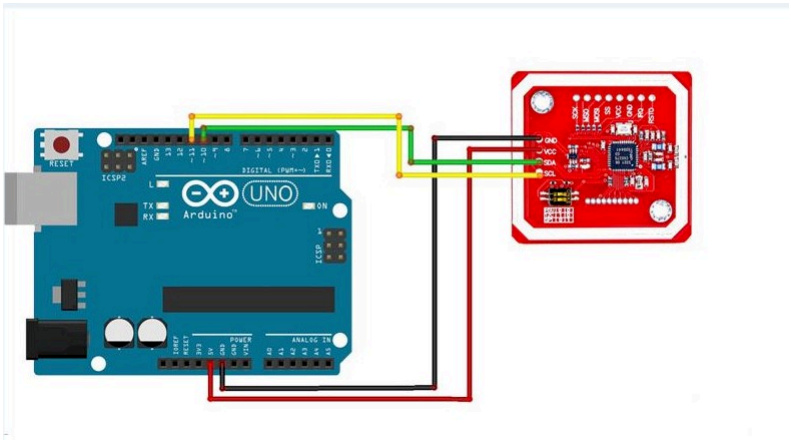

Figura 17: Conexões do Arduino Uno ao leitor RFID.

Fonte: ElectroPeak

5.8 Software desenvolvido

Para que o conjunto todo deste projeto funcionasse de uma maneira harmoniosa, foi necessário desenvolver um software próprio, que pudesse realizar todas as etapas do processo, que são:

- Controlar o robô independentemente;
- Realizar a leitura das tags dos blocos;
- Poder movimentar a garra robótica para coletar e soltar os blocos;
- Poder realizar o cálculo dos movimento através da PNRD;

No início do desenvolvimento, para testar primeiramente o movimento do robô em todos os eixos, o software utilizado foi o Universal Gcode Sender:

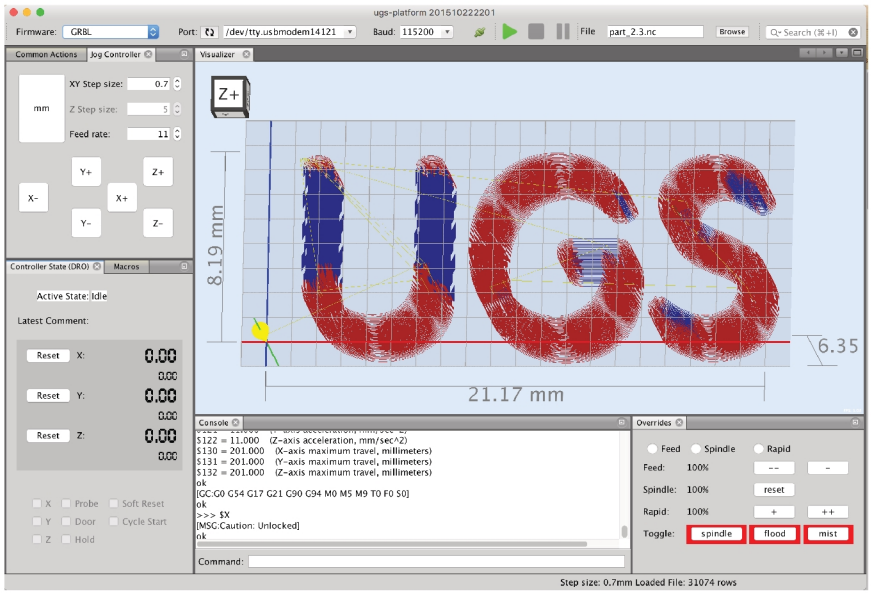

Figura 18: Interface do software open-source Universal Gcode Sender.

Após verificar o funcionamento dos motores de passo do robô cartesiano, o passo seguinte foi desenvolver o software específico que faria a integração entre todos os componentes; para tal, foi utilizado o software open-source GRBL-Plotter 1.6.6.3 como base do projeto, devido ao fato de possuir o código aberto, e também pelo fato de nele já estar inserido a biblioteca que realiza a comunicação entre Arduino Mega, shiled RAMPS 1.4 e o robô cartesiano, além de conseguir se comunicar com o Arduino Uno responsável por controlar o leitor RFID conectado à garra robótica, integrando assim todos os equipamentos do robô.

Devido ao fato de ser um programa bastante robusto e completo, ele se trata de um software com diversas classes em seu projeto principal, como é possível ver na Figura 19.

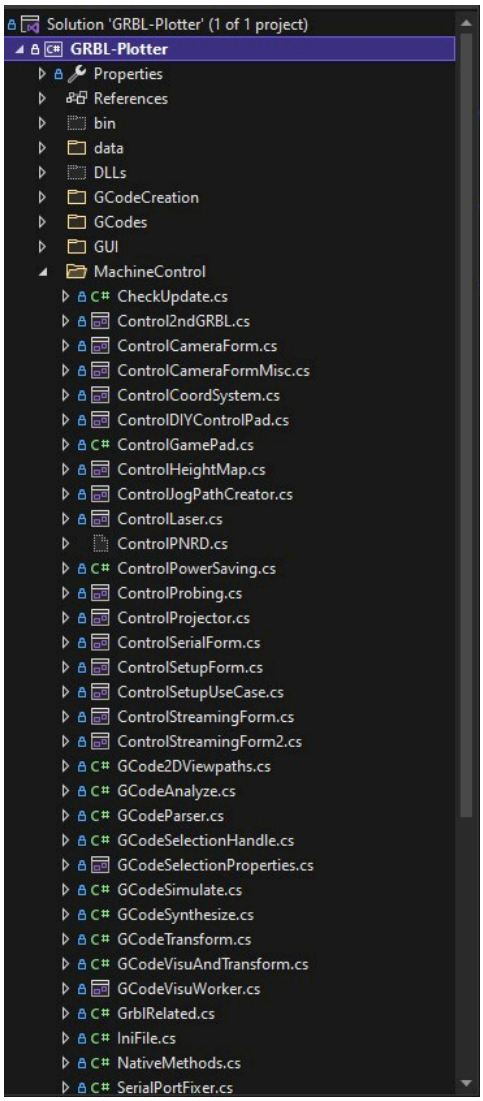

Figura 19: Projeto do software de código aberto GRBL-Plotter 1.6.6.3.

As alteração feita no software de código aberto GRBL-Plotter 1.6.6.3 foi adicionar um botão que iniciará o movimento do robô, que sai sempre da mesma posição inicial e percorre o caminho mostrado na Figura 2, varrendo o espaço de todos os blocos, se movimentando através do código G que se encontra no Apêndice B.

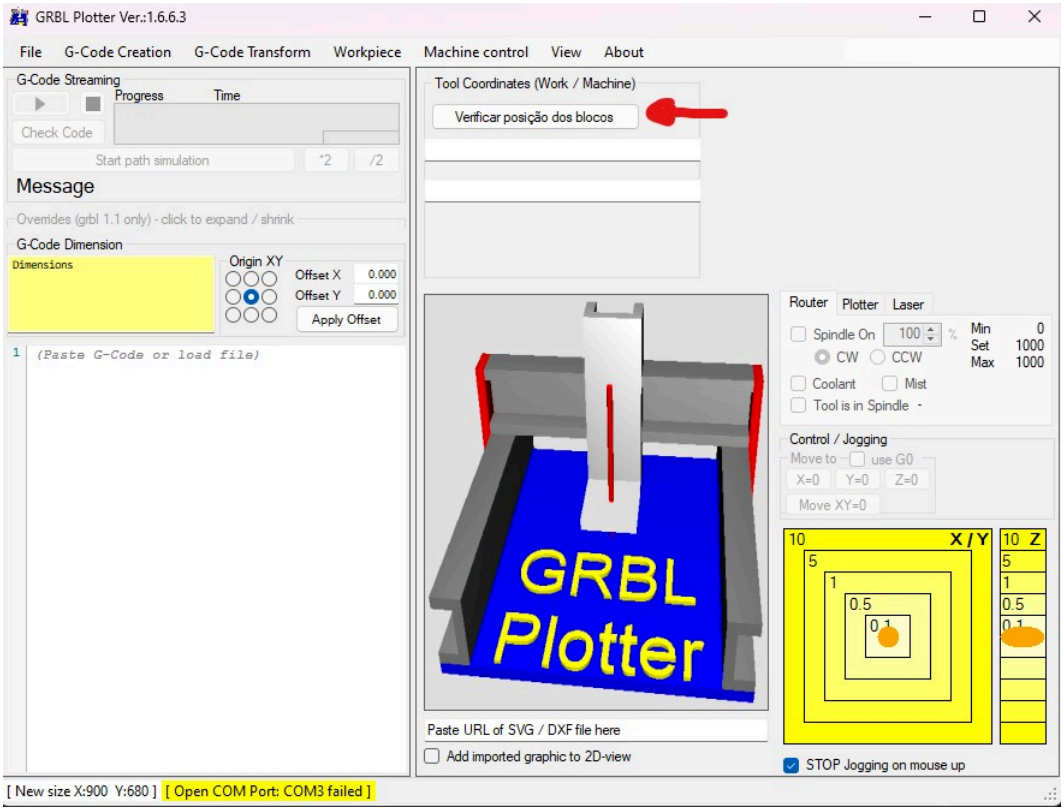

Figura 20: Tela inicial contendo o botão adicionado ao software de código aberto GRBL-Plotter 1.6.6.3.

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    TimeSpan[] testeLeiturasUnoTempo = new TimeSpan[3];
    string[] testeLeiturasUnoPosicao = new string[3];

    string diretorioParente = Directory.GetParent(Environment.CurrentDirectory).Parent.FullName;
    string diretorioGCodeInicial = string.Format("{0}\\GCodes\\gcode.gcode", diretorioParente);
    LoadFile(diretorioGCodeInicial);
    BtnStreamStart_Click(this, null);

    testeLeiturasUnoTempo = _serial_form.connectToArduinoUno();

    testeLeiturasUnoPosicao  = _serial_form.converterTempoEmPosicao();

    for (int i = 0; i < 3; i++)
    {
        textBox1.Text = string.Format("{0} | {1}", textBox1.Text, testeLeiturasUnoTempo[i].Seconds);
        textBox2.Text = string.Format("{0} | {1}", textBox2.Text, testeLeiturasUnoPosicao[i]);
    }
}
```

Figura 21: Função do software que dispara o robô e recebe as leituras realizadas pelo Arduino Uno

Conforme a garra robótica se mexe de acordo com o código G pré estabelecido, caso o leitor RFID acoplado à garra encontra um bloco (devido ao fato do bloco possuir uma TAG RFID acoplada à ele), é imediatamente disparado um sinal ao software que consegue captar este dado graças ao código implementado relatado na Figura

```
1 reference
public TimeSpan[] connectToArduinoUno()
{
    horarioInicial = DateTime.Now;
    string selectedPort = "COM4";
    serialPort[1] = new SerialPort(selectedPort, 115200, Parity.None, 8, StopBits.One);
    serialPort[1].Open();
    //textBox1.Text = "CONECTADO";
    //button1.Text = "Disconnect";
    //enableControls();

    var leituraBloco1 = serialPort[1].ReadLine();
    vetorTempoBloco[0] = (DateTime.Now - horarioInicial);
    //textBox1.Text = vetorTempoBloco[0].ToString();
    var leituraBloco2 = serialPort[1].ReadLine();
    vetorTempoBloco[1] = (DateTime.Now - horarioInicial);
    //textBox2.Text = vetorTempoBloco[1].ToString();
    var leituraBloco3 = serialPort[1].ReadLine();
    vetorTempoBloco[2] = (DateTime.Now - horarioInicial);
    //textBox3.Text = vetorTempoBloco[2].ToString();

    return vetorTempoBloco;
}
```

Figura 22: Função do software que recebe as leituras realizadas pelo Arduino Uno.

É importante salientar que além de receber a informação de cada Tag RFID, (armazenadas nas variáveis leituraBloco1, leituraBloco2 e leituraBloco3) também é iniciado um contador de tempo para verificar o tempo em que cada bloco foi lido e ele é armazenado no vetorTempoBloco[]. Esta etapa é realizada para garantir que cada informação trazida na Tag dos blocos (caso a Tag possua a informação da posição inicial do bloco) é verdadeira, visto que o robô sempre leva o mesmo tempo para percorrer o caminho inicial, portanto toda vez o robô encontrar um bloco na posição 1x3 por exemplo, sempre será no mesmo tempo pré determinado. Este tempo foi determinado empiracamente, fazendo uma varredura inicial e anotando o tempo em que cada bloco era lido em cada posição, com isso foi possível estabelecer a relação entre tempo para realizar a leitura e a posição em que cada bloco se encontra, assim montando a seguinte função no código:

```
1 reference
public string[] converterTempoEmPosicao()
{
    for (int i = 0; i < 3; i++)
    {
        if (22 <= vetorTempoBloco[i].Seconds && vetorTempoBloco[i].Seconds <= 24)
            vetorPosicaoBloco[i] = "1x1";
        //posicao 1x2
        else if (25 <= vetorTempoBloco[i].Seconds && vetorTempoBloco[i].Seconds <= 27)
            vetorPosicaoBloco[i] = "1x2";
        //posicao 1x3
        else if (28 <= vetorTempoBloco[i].Seconds && vetorTempoBloco[i].Seconds <= 30)
            vetorPosicaoBloco[i] = "1x3";
        //posicao 2x1
        else if (40 <= vetorTempoBloco[i].Seconds && vetorTempoBloco[i].Seconds <= 42)
            vetorPosicaoBloco[i] = "2x3";
        //posicao 2x2
        else if (43 <= vetorTempoBloco[i].Seconds && vetorTempoBloco[i].Seconds <= 45)
            vetorPosicaoBloco[i] = "2x2";
        //posicao 2x3
        else if (46 <= vetorTempoBloco[i].Seconds && vetorTempoBloco[i].Seconds <= 48)
            vetorPosicaoBloco[i] = "2x1";
        //posicao 3x1
        else if (58 <= vetorTempoBloco[i].Seconds && vetorTempoBloco[i].Seconds <= 60)
            vetorPosicaoBloco[i] = "3x1";
        //posicao 3x2
        else if (61 <= vetorTempoBloco[i].Seconds && vetorTempoBloco[i].Seconds <= 63)
            vetorPosicaoBloco[i] = "3x2";
        //posicao 2x3
        else if (64 <= vetorTempoBloco[i].Seconds && vetorTempoBloco[i].Seconds <= 66)
            vetorPosicaoBloco[i] = "3x3";
    }

    return vetorPosicaoBloco;

}
```

Figura 23: Função do software que converte o tempo em que cada bloco foi lido, para a sua posição.

Além disso, enquanto a varredura do robô é feita, é disponibilizado no software o caminho total que o robô irá percorrer na cor preta, bem como o caminho já percorrido na cor verde, a fim de facilitar a vizualição e o entedimento. Esta parte já é original do software.

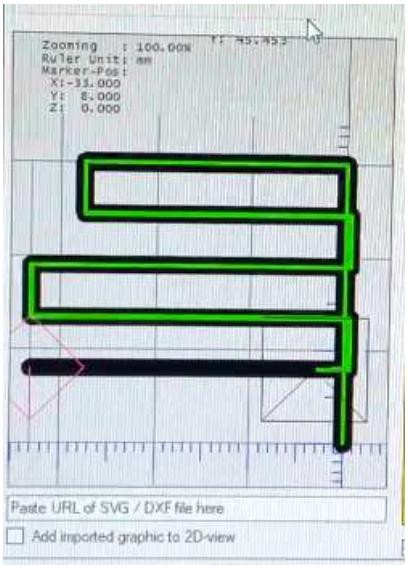

Figura 24: Trajetória do robô vista pelo software.

6 Resultados e Discussões

6.1 Movimentação do robô cartesiano

O primeiro teste realizado no inicio do projeto, foi o de movimentação do robô cartesiano. Sua movimentação é divida em três partes:

- Movimentação no eixo X.
- Movimentação no eixo Y.
- Movimentação no eixo Z.

Após instalar o firmware GRBL for RAMPS 1.4 no Arduino Mega, foi possível controlar o robô através do software Universal G Code Sender (Figura 18). Para os testes inicias foram enviados comandos simples para ele se movimentar em cada posição, com isso foi possível verificar o seu funcionamento e movimentação em cada eixo, como é possível verificar nas Figuras 25 e 26.

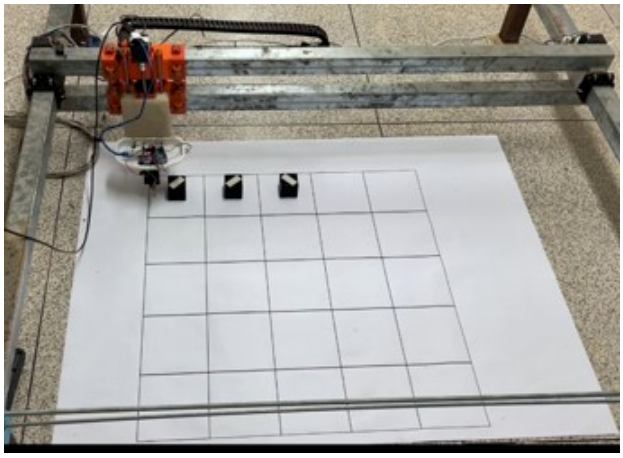

Figura 25: Posição inicial do robô cartesiano.

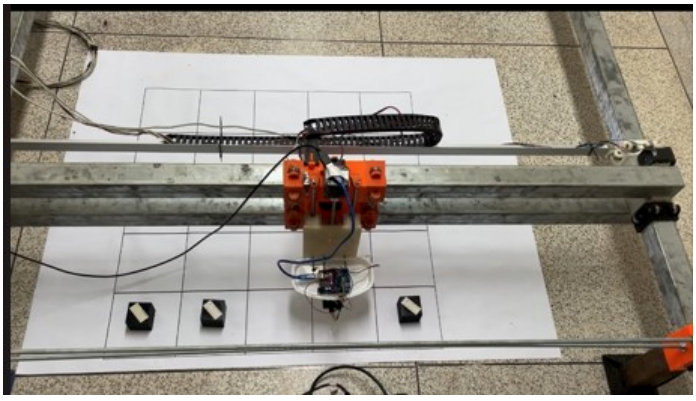

Figura 26: Posição final do robô cartesiano.

6.2 Leitura da Tag RFID

Antes do leitor RFID ser acoplado à garra robótica, foi necessário verificar o seu funcionamento, a fim de evitar quaisquer problemas com a leituras das tags após o acoplamento ser realizado. Com o upload do código C++ feito ao Arduino Uno que controlorá a leitura (visto no Apêndice A), foi possível constatar de que a leitura das Tags RFID estavam funcionando.

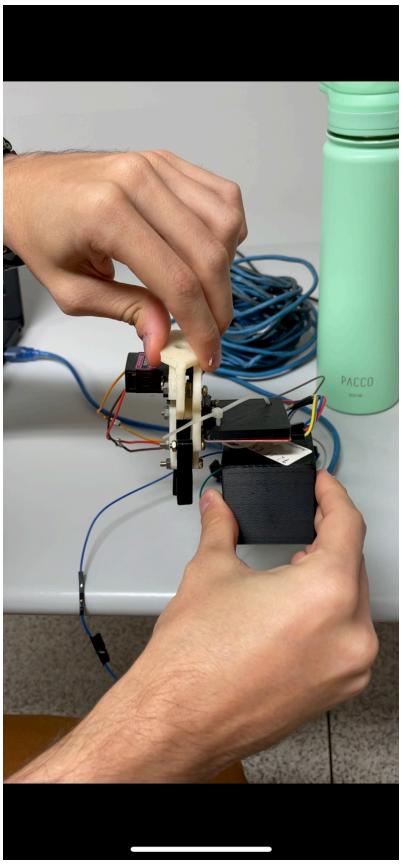

Figura 27: Leitura da Tag sendo feita pelo leitor RFID.

6.3 Software desenlvolvido

Os resultados obtidos do Software desenvolvidos são bem satisfatórios, visto que englobam exatamente a proposta inicial do projeto, que é:

- Preparar o robô para estar totalmente funcional;
- Realizar a leitura das tags dos blocos;
- Controlar o robô independentemente (através do software);
- Criar um software que faça a integração entre a leitura das tags e a movimentação automática do robô;

É possível ver o total funcionamento do software desenvolvido e adaptado no vídeo cujo link se encontra no Apêndice C.

7 Conclusão

O trabalho realizado propiciou um grandioso aprendizado sobre um projeto de engenharia. A partir do projeto de adaptação de um robô cartesiano já existente para trabalhar com a tecnologia PNRD e iPNRD para solução do problema do mundo de blocos, também realizando o processo de melhoria, padronização e documentação, possibilitando o correto funcionamento do sistema e a utilização em projetos futuros.

De forma a cumprir o objetivo do trabalho, foi realizado o estudo e análise da estrutura inicial, componentes e parâmetros do robô cartesiano existente, definindo as alterações necessárias e também realizando manutenção na máquina. Para a adaptação de peças foi preciso o estudo sobre o uso da garra robótica, bem como a melhor maneira de acoplar o leitor RFID e seu arduino dedicado, as soluções de mercado e o projeto das peças para acoplamento e suporte dos componentes.

A utilização de peças improvisadas,foi uma etapa bastante desafiadora, visto que o sucesso de suas adaptações foram por tentativa e erro. A montagem do sistema é complexa e objetivava o correto funcionamento da máquina, o que foi validado com os testes realizados.

Levando-se em consideração esses aspectos, analisando aos requisitos adotados para o projeto inicialmente e comparando com os resultados obtidos, é notável que o desenvolvimento deste trabalho atinge os requisitos propostos, e é possível concluir que tais objetivos foram atingidos em total integridade.

8 Trabalhos Futuros

Para trabalhos futuros utilizando este projeto, recomenda-se:

- Aplicação da PNRD e iPNRD utilizando a adaptação feita ao robô;
- Sistema de otimização mecânica para melhorar a estrutura do sistema em relação as características mecânica da máquina e identificando pontos de falha;
- Sistema que possibilite o robô a trabalhar remotamente utilizando um servidor em nuvem;

9 Referências Bibliográficas

(Rodrigues, 2017) - RODRIGUES, Bernardo Henrique de Morais. Robô Cartesiano Cnc: Projeto e construção de um robô cartesiano multifuncional. 2017. Trabalho de Conclusão de Curso (Graduação em Engenharia Mecatrônica) – Universidade Federal de Uberlândia, Uberlândia, 2017.

(Fonseca, 2018) - FONSECA, João Paulo da Silva. Redes de petri de alto nível e pnrd invertida associadas ao controle de robôs móveis: uma abordagem para operações de busca e salvamento em trilhas e travessias. 2018. Tese apresentada ao Programa de Pós- graduação em Engenharia Mecânica da Universidade Federal de Uberlândia (Doutorado em Engenharia Mecânica) – Universidade Federal de Uberlândia, Uberlândia, 2018.

(Pires, 2019) - PIRES, Alan Rogério Barreto. Sistema ciber-físico baseado na integração PNRD e IPNRD utilizando cloud computing. 2019. Trabalho de Conclusão de Curso (Graduação em Engenharia Mecatrônica) – Universidade Federal de Uberlândia, Uberlândia, 2019.

(Oliveira, 2020) - OLIVEIRA, Nina Cervilha. Adaptação de robô cartesiano para máquina de corte a laser. 2020. Trabalho de Conclusão de Curso (Graduação em Engenharia Mecatrônica) – Universidade Federal de Uberlândia, Uberlândia, 2020.

(Tavares, Souza) - TAVARES, José Jean Paul Zanlucchi de Souza e SOUZA, Gabriel de Almeida. PNRD and iPNRD Integration Assisting Adaptive Control in Block World Domain. Disponível em: https://ceur-ws.org/Vol-2424/paper6.pdf

(JM Rosário,2010) - ROSÁRIO, João Mauricio. Robótica Industrial I: Modelagem, Utilização e Programação. São Paulo, João Maurício Rosário, 2010.

10 Apêndice A - Programa do Arduino Uno

Programa na linguagem C++ que se encontra no Arduino Uno utilizado.

```
    /*
  PN532-NFC-RFID-Module-Library
  modified on 18 Nov 2020
  by Amir Mohammad Shojaee @ Electropeak
  https://electropeak.com/learn/
  based on www.electroschematics.com Arduino Examples
*/
#include <SoftwareSerial.h>
#include <PN532_SWHSU.h>
#include <PN532.h>
SoftwareSerial SWSerial( 10, 11 ); // RX, TX
PN532_SWHSU pn532swhsu( SWSerial );
PN532 nfc( pn532swhsu );
void setup(void) {
  Serial.begin(115200);
  Serial.println("Começando a Leitura!");
  nfc.begin();
  uint32_t versiondata = nfc.getFirmwareVersion();
  if (! versiondata) {
    Serial.print("Não foi possível encontrar um modulo PN53x");
    while (1); // Halt
  }
  // Caso tenha encontrado um valor, esse valor é printado

  Serial.print("Encontrado chip PN5"); Serial.println((versiondata>>24) & 0xFF, HEX);

  Serial.print("Versao do Firmware. "); Serial.print((versiondata>>16) & 0xFF, DEC);

  Serial.print('.'); Serial.println((versiondata>>8) & 0xFF, DEC);

  // Configurar o Arduino para ler tags RFID
  nfc.SAMConfig();
  Serial.println("Esperando uma TAG ...");
}
void loop(void) {
  boolean success;
  uint8_t uid[] = { 0, 0, 0, 0, 0, 0, 0 };  // Buffer para armazenar o UID retornado
  uint8_t uidLength;                        // Tamanho do retornado UID
success = nfc.readPassiveTargetID(PN532_MIFARE_ISO14443A, &uid[0], &uidLength);
  if (success) {
    Serial.println("Tag Encontrada!");
    Serial.print("UID Length: ");Serial.print(uidLength, DEC);Serial.println(" bytes");
    Serial.print("UID Value: ");
    for (uint8_t i=0; i < uidLength; i++)
    {
```

```
      Serial.print(" 0x");Serial.print(uid[i], HEX);
    }
    Serial.println("");
    // Delay de 2 segundos para não bagunçar a leitura
    delay(2000);
  }
  else
  {
    // Tratamento de erro de leitura
    Serial.println("Erro na leitura...");
  }
}
```

11 Apêndice B - Código G

Código G que se encontra no software desenvolvido, com o intuito realizar o movimento inicial do robô cartesiano.

```
O0001
N10 G00 X0.00 Y0.00
N20 G01 Y30.00 F1000.00
n25 G01 X-0.05
N30 X-27.5
N40 Y24.50
N50 X0.50
N60 Y19.00
N70 X-33.00
N80 Y13.50
N90 X0.50
N110 Y8.0
N120 X-33.00
N500 M30
```

12 Apêndice C - Links Úteis

Links dos softwares utilizados, bem como vídeos mostrando o desenvolvimento do projeto ao longo do tempo.

Universal G-code sender
Versão: 1.1
Link : https://winder.github.io/ugs$_website$/

Firmware GBRL
Versão: GBRL for Ramps 1.4
Link: https://sourceforge.net/projects/grblforramps14

GRBL Plotter
Versão: GRBL-Plotter Vers. 1.6.6
Link: https://github.com/svenhb/GRBL-Plotter

Vídeos contendo o desenvolvimento do projeto
Link: https://drive.google.com/drive/u/0/folders/1-44OzhmzM_zYHVpRFsPuBdymD6kflWCK