INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO NORTE CAMPUS NATAL ZONA NORTE

ANNE CAROLINE DE OLIVEIRA SILVA
LUANA DE MACEDO SILVA
RAIANA DA SILVA TORRES

SISTEMA AUTÔNOMO DE LOGÍSTICA INDUSTRIAL

NATAL - RN
2018

ANNE CAROLINE DE OLIVEIRA SILVA
LUANA DE MACEDO SILVA
RAIANA DA SILVA TORRES

SISTEMA AUTÔNOMO DE LOGÍSTICA INDUSTRIAL

Trabalho de Conclusão de Curso apresentado ao Curso Técnico de Nível Médio em Eletrônica, na Forma Integrado, do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, como parte das exigências para a obtenção do Diploma de Técnico em Eletrônica.

Orientador: Prof. Pedro Ivo de Araujo do Nascimento

NATAL - RN
2018

ANNE CAROLINE DE OLIVEIRA SILVA
LUANA DE MACEDO SILVA
RAIANA DA SILVA TORRES

SISTEMA AUTÔNOMO DE LOGÍSTICA INDUSTRIAL

Trabalho de Conclusão de Curso apresentado ao Curso Técnico de Nível Médio em Eletrônica, na Forma Integrado, do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, como parte das exigências para a obtenção do Diploma de Técnico em Eletrônica.

Trabalho de Conclusão de Curso apresentado e aprovado em ___/___/____, pela seguinte Banca Examinadora:

BANCA EXAMINADORA

_____

Prof Msc. Pedro Ivo de Araújo do Nascimento - Orientador
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

_____

Prof Msc. Marcus Vinicius Araujo Fernandes - Avaliador
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

_____

Prof Msc. Daniel Guerra Vale da Fonseca - Avaliador
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

AGRADECIMENTOS

Precisamos reconhecer que a elaboração do presente Trabalho, como também, a caminhada nesses 4 anos de curso não seria possível sem presença de algumas pessoas, essas sem dúvidas merecem nossos agradecimentos. Primeiramente, agradecemos a Deus por ter nos acompanhado em cada uma das experiências vividas ao longo do curso. Devemos nossos agradecimentos também, a cada um dos professores que passaram por nós e deixaram ensinamentos que levaremos para a vida, em especial ao nosso Orientador Pedro Ivo.

Não podemos esquecer de agradecer a nossa turma 4.4206.1V, 2018 (XOR), que nesses 4 anos se tornou uma segunda Família, vivemos momentos bons e ruins e somos gratas por cada um deles. Agradecemos especialmente ao nosso amigo Nicodemos Jonatas, por ter nos auxiliado em muitos momentos no desenvolver do trabalho. Somos muito gratas a empresa FESTO e a Universidade de Campinas que nos convidaram a participar do curso de capacitação em ROS, agradecemos também o apoio IFPB que contribuiu para o nosso desenvolvimento.

Por fim, gostaríamos de agradecer às nossas famílias. Primeiramente aos nossos pais que foram extremamente importantes na construção do nosso caráter e que se esforçam para garantir que nós tenhamos a melhor educação possível. Além deles, somos gratas a todos os familiares e amigos que mesmo indiretamente nos ajudaram a chegar onde chegamos.

RESUMO

Este projeto tem como objetivo o desenvolvimento de um sistema robótico autônomo, capaz de receber ordens de execução de tarefas em um sistema de logística industrial e traçar rotas que levem em consideração o menor caminho e o menor tempo de execução das atividades. Este sistema robótico será baseado na plataforma robótica Robotino 3.0 da Festo já disponível no Campus Natal Zona Norte e utilizará sistema de manufatura integrados para simulação do ambiente fabril. Para isso será utilizado o sistema operacional Robotino View, além de câmeras para identificação de padrões e de um manipulador robótico de 3 eixos.

Palavras-chave : Robotino; indústria 4.0; Arduino; manipuladores; servo motor; robótica; robôs móveis.

ABSTRACT

This project aims to develop an autonomous robotic system capable of receiving orders to perform tasks in an industrial logistics system and to draw routes that take into account the mallest path and the shortest execution time of the activities. This robotic system will be based on the Robotic Robotino 3.0 platform from Festo already available at Campus Natal Zona Norte and will use an integrated manufacturing system to simulate the manufacturing environment. Robotino View operating system will be used, as well as cameras for pattern identification and a 3-axis robotic manipulator.

Keywords: Robotino; industry 4.0; Arduino; manipulators; servo motor; robotics; mobile robots.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Desde a antiguidade o homem se questiona sobre os objetos inanimados e como seria o comportamento humano caso fosse possível dar vida e movimento a aqueles que nos auxiliam nas tarefas mais árduas. Até mesmo aristóteles pensava a respeito de como isso poderia afetar o comportamento da sociedade como ele cita em política, - "escravidão deixaria de ser necessária se as lançadeiras e os plectros pudessem mover-se por si mesmos" (koyré, 1991). e ao longo da história fazer com que as máquinas fossem nossa força de trabalho autônoma virou praticamente uma obsessão.

Para a civilização ocidental a forma de contar o seu respectivo avanço sempre se deu juntamente as formas de tecnologia empregada em sua época, tem sido assim desde a idade dos metais até a atualidade. Assim como também sempre foi da vontade do homem facilitar seu trabalho através do uso das máquinas, e assim é feito desde então.

Todas as épocas contam com suas atualizações de tecnologia, foi assim após a descoberta do fogo, invenção da roda, das máquinas a vapor, e pouco tempo depois com a invenção do transistor, com o passar dos anos vemos um crescimento exponencial das tecnologias que são desenvolvidas, o que nos trás aos dias atuais com o grande desafio de lidar com as máquinas autônomas, também conhecidas como robôs.

Os robôs são idealizados nas áreas da artes por muito tempo, O termo “robota” (servidão, esforço) foi usado pela primeira vez em 1921 pelo dramaturgo Tcheco Karen Capek mas foi de fato popularizado pelo escritor Isaac Asimov (CARRARA,2009), “como uma máquina com aparência humana não possuidora de sentimentos” (Romano e Dutra, 2002). O escritor também criou o que conhecemos hoje como as três leis fundamentais da robótica que são elas:

> *1ª-Um robô não pode fazer mal a um ser humano e nem consentir, permanecendo inoperante, que um ser humano se exponha a situação de perigo;*
> *2ª-Um robô deve obedecer sempre às ordens de seres humanos, exceto em circunstâncias em que estas ordens entrem em conflito com a 1ª lei;*
> *3ª. Um robô deve proteger a sua própria existência, exceto em circunstâncias que entrem em conflito com a 1ª e 2ªleis. (Romano e Dutra, 2002)*

Prosseguindo pelo ramo artístico, o primeiro robô encontrado nas telas de cinema se encontra no filme Metrópolis de 1926 e leva o nome de Maria, ele tinha aspecto de uma mulher, sendo esta, criada e programada para causar a discórdia entre o proletariado.

Atualmente há inúmeros desafios a serem enfrentados na área da tecnologia, estes ajudarão a formar os moldes de uma nova humanidade que irá mudar sua forma de ver o mundo, como vive, trabalha e se relaciona entre si, irá mudar não só o mundo artístico ou material mas transformará a forma de convívio social. Este advento já é conhecido como a quarta revolução industrial, tem um avanço tecnológico imenso que está tomando grandes proporções.

> *[...]Considere as possibilidades ilimitadas de ter bilhões de pessoas conectadas por dispositivos móveis, dando origem a um poder de processamento sem precedentes, capacidades de armazenamento e acesso ao conhecimento. Ou pense na impressionante confluência de inovações tecnológicas emergentes, abrangendo campos como inteligência artificial (IA), robótica, internet das coisas (IoT), veículos autônomos, impressão 3D, nanotecnologia, biotecnologia, ciência dos materiais, armazenamento de energia e computação quântica. [...](Schwab, 2016)*

Isto leva ao intuito deste projeto, que é a construção de um sistema robótico que seja capaz de se movimentar autonomamente fazendo o recolhimento, transporte e depósito de peças através de um manipulador móvel, em um ambiente que simula uma planta industrial, com o objetivo de criar uma solução cada vez mais rápida, eficiente e precisa desta tarefa, facilitando os processos e dando uma maior dinamicidade a eles.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 ROBÓTICA

Nas definições mais atuais, um robô industrial, de acordo com a norma ISO (International Organization for Standardization), é “uma máquina manipuladora com vários graus de liberdade controlada automaticamente, reprogramada, multifuncional, que pode ter base fixa ou móvel para utilização em aplicações de automação industrial”. As novas técnicas bases criadas para robôs que são utilizados atualmente foram desenvolvidas durante a Segunda Guerra Mundial, nesse momento houve a construção de um dispositivo conhecido como teleoperador ou “master-slave” sistema muito utilizado até hoje, era utilizado para manuseio de material radioativo. Separado em duas partes as quais uma ficava com o operador que tinha como tarefa realizar movimentos com o que é chamado de “master” e em seguida a outra parte chamada de “slave” que era capaz de realizar os movimentos produzidos pelo master a distância (Romano e Dutra, 2002).

Atualmente não há nenhuma dúvida que os robôs podem ser utilizados em todo lugar, para as mais diferentes tarefas, podem ter formato, tamanhos e pesos qualquer, isto depende muito do projeto, as necessidades do ambiente e do seu material (Pandey & Gelin, 2018). Todos estes aspectos são escolhidos minuciosamente para as mais variadas aplicações, desde ajudar pessoas em pequenas necessidades do dia a dia até a em grandes indústrias.

Usando de suas estruturas robustas, e forças sobre humanas ou precisões milimétricas aos robôs realizam as tarefas que humanos não podem (ou não querem) realizar, há também do tipo que têm capacidade de ser sociável. Um dos fatores que faz das máquinas robóticas tão importantes é sua acurácia, o que torna os objetos produzidos muito mais padronizados do que aqueles realizados por mãos humanas, aumentando a qualidade de peças e menor tempo e diminuindo o risco de falhas.

Há alguns dispositivos geralmente encontradas em robôs, principalmente industriais, são eles:

- Sensores;
- Unidade de controle;
- Atuadores;

- Manipuladores;

possuem a capacidade de se mover ao redor do ambiente, não estão fixados na sua própria base, também conhecido por veículo robótico ou rover. Podem ser classificados de acordo com o ambiente que se movem, tais como ambientes terrestres, aquáticos (AUVs), aéreos (UAVs) e híbridos.

2.1.1 Robótica industrial

Um progresso notório na automatização de produção se deu em 1769, quando James Watt desenvolveu a máquina a vapor e a forma de produzir das máquinas era voltada para fabricação do mesmo produto em série com o intuito de manter uma alta produtividade e volume, isso ocorreu até metade do século XX, então a tecnologia avançou e houve então uma mudança na organização das indústrias, que acabaram por optar por outros meios de produção, já que houve a necessidade de uma maior dinamicidade nos processos produtivos e por isso robôs industriais passaram a ser cada vez mais empregados pois: "[...]são essencialmente máquinas capazes de realizar os mais diversos movimentos programados, adaptando-se às necessidades operacionais de determinadas tarefas e empregando garras e/ou ferramentas oportunamente selecionadas.(Romano e Dutra, 2002)".

Atualmente há um constante investimento voltado para as áreas tecnológicas como ciência da computação, mecânica, eletrônica digital, materiais e logística da produção o que vêm contribuindo para um crescimento da segurança nos projetos de robôs também como uma significativa diminuição de gastos para efetivação destes nas indústrias, tudo isto aconteceu graças às características flexíveis de programação, e adaptação que são tão desejadas pela nova indústria e advém de uma necessidade do mercado consumidor que a cada dia se mostra mais dinâmico.

O uso de robôs industriais no em uma empresa está diretamente associado aos objetivos da produção, visando uma redução de custos com a redução de mão de obra humana, inclusive retirada dessa de atividades perigosas e insalubres, salto positivo na

produtividade, melhoras significativas nos padrões e qualidade dos produtos, e uma grande redução no desperdício de matéria-prima, sem contar o aumento na precisão de objetos em escala milimétrica. E por isso, Dispositivos robóticos estão sendo utilizados em larga escala na indústria, onde os robôs manipuladores operam nos mais diversos setores, desde da soldagem, pintura, até a montagem de materiais, tudo isso devido a sua velocidade, repetibilidade e precisão. Contudo, a falta de mobilidade dos robôs manipuladores restringe sua utilização, trazendo com isso a necessidade de outra classe de robôs, os robôs móveis.(SIEGWART e NOURBAKHSH, 2004).

Enquanto os robôs manipuladores são empregados para deslocar objetos definidos numa área de trabalho restrita através de movimentos conhecidos, os robôs móveis (robôs que possuem a capacidade de se movimentar em determinados ambientes, podendo ser esses ambientes aéreos, terrestres ou aquáticas.) se deslocam numa área de trabalho muitas vezes ilimitada e desconhecida, tendo que lidar com incertezas, como a diferença de ambientes e obstáculos desconhecidos e móveis.

Assim a classe de robôs móveis, devem ser constituídos por uma plataforma capaz de mover-se automaticamente num ambiente, percebendo e reagindo às mudanças do ambiente. Na qual, devido a capacidade de interagir com diversos ambientes, torna o estudo em robótica móvel de grande importância não apenas para a indústria, como também para aplicações de tecnologias no dia-a-dia.

Espera-se que a produção industrial mude consideravelmente no futuro próximo, uma mudança de paradigma chamada frequentemente de Indústria 4.0 (KAGERMANN, 2013). Parte dessa visão são as fábricas inteligentes, instalações sensíveis ao contexto que podem levar em conta informações como posições de objetos ou status das máquinas. Essas fábricas fornecem serviços de fabricação que podem ser combinados e modificados eficientemente de quase todas as formas(LUCKE, 2008).

Enquanto algumas fábricas serão projetadas de acordo com essa visão com redes de máquinas, ainda há instalações que serão incrementalmente atualizadas por razões econômicas, exigindo que os robôs se adaptem às máquinas existentes e trabalhem com segurança ao lado de humanos(ANGERER, 2012). Uma dessas adaptações é o reconhecimento de padrões de imagens, como por exemplo o padrão de luz industrial. Esses

sinais são frequentemente usados para indicar o status de uma máquina, por exemplo quando está prestes a ficar sem material, ou se é seguro para um humano realizar certas operações. Deste modo é necessário que os robôs móveis além de se deslocarem de maneira segura pelo ambiente industrial, também sejam capazes de identificar padrões de sinalização e para isso é necessária a visão computacional, por meio de câmeras e o seu processamento adequado.

2.2 INDÚSTRIA 4.0

A primeira Revolução Industrial (I1.0) teve início no século XVIII na Inglaterra e foi o marco inicial para uma série de mudanças que vieram acontecendo na indústria desde então. Um dos principais fatores que influenciou essa Revolução foi o surgimento da máquina a vapor de James Watt, desenvolvida entre os anos 1768 e 1782, ela foi utilizada principalmente na indústria de tecido. Neste momento já se via as transformações sofridas pela indústria: os trabalhadores artesanais, que antes eram responsáveis por todo o processo de produção, ganharam chefes, que passaram a monitorar a produção e gerenciar os lucros (ZARTE, 2016).

Aproximadamente dois séculos mais tarde, e com o fim da Segunda guerra mundial (1945) as técnicas de produção já existentes foram aprimoradas e, além disso, as áreas da indústria química, elétrica e do aço tiveram evoluções significativas. Neste período da Segunda Revolução Industrial(I2.0), surgiram os primeiros barcos de aço com motores a vapor, impulsionando assim o transporte de mercadorias.

Posteriormente, nas décadas de 1950 e 1970 a Terceira Revolução Industrial(I3.0) trouxe consigo o uso dos semicondutores, dos computadores, automação e robotização em linhas de produção. Inclusive, a robótica é uma das característica mais marcantes desta etapa.

Com o constante desenvolvimento da tecnologia de internet nessas últimas décadas, como também, a tecnologia desenvolvida na área dos projetos inteligentes, teve início uma outra transformação na indústria que desde o princípio, vem causando impactos que estão transformando o mundo que conhecemos hoje. Os professores Erilk Braynjolfsson e Andrew McAfee do Instituto de Tecnologia de Massachusetts apelidaram essa transformação como a “segunda idade da máquina”, porém, somente em 2011 na Feira Industrial de Hannover, na Alemanha ela foi chamada pela primeira vez de Indústria 4.0 (DREHER, 2016).

Essa nova era industrial é singularizada pelo uso de sistemas super inteligentes e capazes de tomar decisões autônomas. Ela se baseia na combinação de várias tecnologias. Aos poucos está forçando as empresas a refletir sobre a forma de administrar os seus negócios e processos, como se estabelecem na cadeia de valor, como pensam na evolução de novos produtos e os introduzem no mercado, ajustando as ações de marketing e de distribuição (COSTA, 2017). Até por que, a ideia de Indústria 4.0 é extremamente completa, o economista Klaus Schwab aponta 4 das principais mudanças que essa última evolução da indústria promete:

- Produtos mais inteligentes e mais produtivos;
- Elevação da expectativa dos clientes;
- Novas maneiras de contribuição e sociedade;
- A modificação do modelo operacional e conversão em modelo digital.

Atuando juntamente com essa ideia da Indústria 4.0, estão a Internet das Coisas (IoT)e os Sistemas Cyber-Physical (CPS).

2.2.1 Internet das coisas (iot)

Conectar vários dispositivos através da internet é algo que sem dúvidas possibilita muitos avanços na área da automação, e é exatamente isso que a junção entre a indústria 4.0 e a IoT viabiliza. A *Internet of things* já está bastante presente em nosso dia-a-dia, por exemplo, quando ligamos a nossa TV através de um smartphone, neste momento, os aparelhos se conectam por uma rede e passam a trabalhar juntos, ilustrado na Figura 1. Obviamente, esse é um simples exemplo no meio de uma infinidade de processos que podem ser transformados através dessa inovação.

É possível relacionar essa união com o funcionamento do nosso cérebro através da fala do inventor Nikola Tesla: “Quando a tecnologia sem fio for perfeitamente aplicável, a Terra inteira será convertida em um imenso cérebro, o que de fato é, com todas as coisas sendo partículas de um todo real e rítmico”(HUNT, 2010).

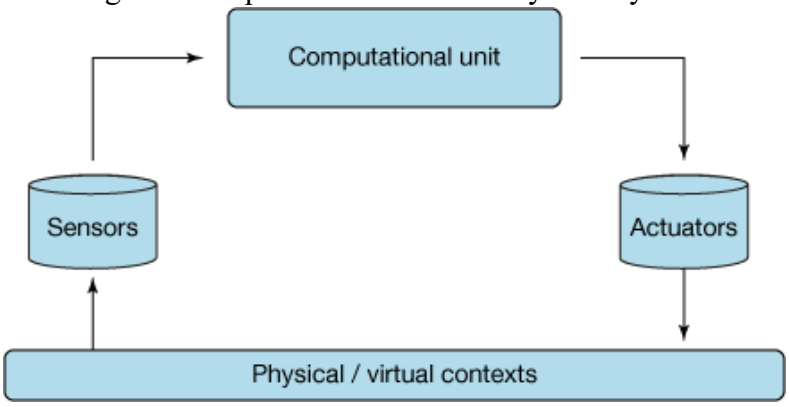
Figura 1: Imagem ilustrativa acerca de internet das coisas.

Fonte : Mobilis Computação Móvel

2.2.2 Sistemas cyber-physical (cps)

Os sistemas Cyber-Physical, também conhecido como CPS, são sistemas computacionais e colaborativos os quais elementos da computação se comunicam e se coordenam através de sensores e atuadores. Os sensores fiscalizam os indicadores tanto físicos quanto virtuais, em seguida os atuadores modificam o ambiente virtual e físico onde são executados, assim como é mostrado na Figura 2. Antes da aparição deste novo sistema, nós estávamos na geração dos embarcados, a diferença entre eles é que os sistemas cyber-physical, como seu próprio nome diz, enfatizam as ligações entre os elementos computacionais e os elementos físicos, enquanto os embarcados focam somente nos elementos computacionais (KHAITAN, 2014).

Figura 2: Arquitetura do Sistema Cyber-Fhysical.



Fonte: ibm, developerWorks.

Apesar do nome ainda não ser algo comum entre a maioria das pessoas, as possíveis aplicações do sistema em questão são muito mais úteis do que imaginam. Como por exemplo na agricultura, o uso de dispositivos com essa tecnologia resultará em um aumento da produtividade, já que o mecanismo buscará a eliminação das falhas e das possíveis perdas existentes. Contudo, o Sistema Cyber-Physical promete afetar positivamente diferentes áreas da indústria.

2.3 ROBOTINO 3

O robotino foi criado e desenvolvido pela empresa Festo Logistics, para ser didático e ser utilizado por qualquer pessoa, já que conta com um software de programação de fácil utilização chamado de Robotino View, que funciona a partir da conexão de blocos, mas também pode ser programado nas mais diversas linguagens, como java, C++, python e etc.

Este tem um chassi circular, em todas as suas versões, incluindo a mais atual versão, Robotino 3, o qual nos aprofundaremos na versão Basic, que está mostrada na Figura 3, ele contém 9 sensores infravermelhos posicionados na parte mais baixa de sua estrutura, duas baterias 12 volts, 3 rodas montadas em formato de estrela, cada uma separadas com 120 graus e distância uma da outras, o que dá a característica ominidirecional ao robô, uma unidade de controle com um processador entradas e saídas, digitais e analógicas, 2x Ethernet, 6x USB

2.0 (HighSpeed), 2 slots PCI Express, 1 interface VGA - 1x I / O para integração de componentes elétricos adicionais. Ele contém um processador Intel Core i5 com 2.4 GHz ou 1,8 GHz. O sistema operacional é armazenado em uma memória de 32 GB SSD. Um microcontrolador de 32 bits que gera diretamente os sinais PWM para o acionamento de até quatro motores CC elétricos é responsável pelo controle do motor. Um FPGA é usado para ler os valores do encoder dos motores. Isso permite, por exemplo, que os dados do odômetro e quaisquer dados adicionais de correção específicos do sensor sejam calculados diretamente no microcontrolador (Festo, 2013).

Figura 3: Robotino 3

Fonte:Fawkes Trac
(https://trac.fawkesrobotics.org/wiki/Robotino3)

Além do possível uso industrial este robô é costumeiramente utilizado em inúmeras competições espalhadas pelo mundo, a mais conhecida se leva o nome de Robocup, que foi criada para que houvesse um evento tão grande quanto a copa do mundo de futebol, para robôs, ela ocorre todos os anos e teve início no Japão, na cidade de tóquio, em junho de 1993, inicialmente era chamada de J-League, que era também o nome da recém criada liga japonesa de futebol profissional. Após um curto período de tempo, pesquisadores de todo o mundo entraram em contato solicitando a ampliação do evento, e só a partir daí o projeto passou a ser chamado de “Robot cup initiative”, ou como é mais conhecido, “Robocup”.

Na Robocup, atualmente há várias categorias, além do futebol, e uma delas leva o nome de “*Logistics*”, é utilizado o uso de robôs autônomos que tem como principal tarefa

mover peças enfrentando obstáculos até o alcance do objetivo proposto. Há as etapas Nacionais e a etapa mundial, ambas são “Open”, ou seja, não há necessidade de classificação para participar, basta ter os equipamentos necessários e se inscrever. Em ambas as fases a competição acontece todo ano em locais diferentes. No Brasil em 2018 a regional acontecerá em João Pessoa/PB e tem como principal objetivo a movimentação de discos pelo chão da arena para completar determinadas tarefas, como podemos ver na imagem 4. Já a etapa mundial aconteceu no Canadá na cidade de Montreal, e esta fase conta com uma estrutura mais elaborada, os robôs devem manipular os discos com uma garra na parte superior do robô, entre os chamados “MPS”, módulos que simulam plantas industriais, fabricados pela empresa Festo que simulam máquinas de manufatura, simulando uma fábrica base da teoria da Industria 4.0. O Robô tem a meta de se comunicar com os outros e levar as peças a cada etapa da fase de produção até que estejam prontas, podemos ver uma fase na Figura 4.

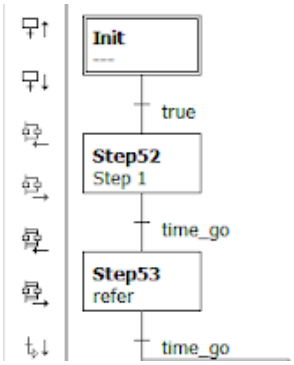
Figura 4**:** A esquerda um MPS e a direita Robotino 3.

Fonte: Blog da equipe Fawkes
(https://www.fawkesrobotics.org/blog/2015/04/26/)

2.3.1 Robotino view

O Robotino View é o ambiente de programação gráfica interativa próprio do Robotino, é onde os programas de controle podem ser criados e executados de forma mais simplificada. Nele, os programas de sequenciamento são exibidos na forma de GRAFCET (Gráfico Funcional de Comandos Etapa-Transição), é possível visualizar o gráfico na Figura 5.

Figura 5: Recorte do Gráfico de função sequencial (GRAFCET) montado no Robotino View.



Fonte: Autores.

Os componentes de hardware do Robô, nessa plataforma são representados por blocos funcionais, como por exemplo os sensores, motores, câmeras, garra, odometria e assim por diante. Dentre os blocos de função de navegação estão o position navigator, distance navigator e obstacle avoidance. Para o processamento de imagem encontra-se o colour range search, line detector e marker detection. Além dos blocos citados há um bloco em especial que é chamado LUA, que permite que as entradas e saídas sejam manipuladas através de linhas de códigos utilizando a linguagem denominada LUA script.

2.4 ROS

O ROS (Robot Operating System), é um framework[1] de software aberto, construído para contribuir para a geração de robôs atual. É um acervo de ferramentas, bibliotecas e convenções com o objetivo de tornar mais simples as criações de comportamentos complexos

do robô. O sistema pode ser utilizado em integração com várias plataformas, e está disponibilizado para vários sistemas operacionais como Windows, Linux e Mac OS. Ele consegue intermediar com qualidade os Sistemas operacionais com as bibliotecas abertas e suas aplicações sem perdas para o usuário (Vargas & Tavares, 2013).

O ROS foi criado, assim como muitos outros projetos, a partir da necessidade de uma estrutura de colaboração aberta. Em meados da década de 2000 vários grupos na Universidade de Stanford desenvolveram protótipos internos de sistemas de softwares flexíveis e dinâmicos designados ao uso robótico. Mas somente em 2007 a Willow Garage, conhecida por ser incubadora de tecnologia voltada a robótica visionária, forneceu recursos para que esses conceitos fossem ampliados. Desse momento em diante, esse sistema vem sendo bastante utilizado entres os pesquisadores da área da robótica.

2.5 MANIPULADORES

Segundo alguns estudos atuais, o uso de manipuladores robóticos estão se multiplicando em larga escala no meio industrial, cada vez mais empresas investem em tecnologia visando uma melhor automação dos processos laborais. Os manipuladores é um meio de trabalho utilizado desde do tempo da revolução industrial, como um método de substituição humana, pois a implantação dessas máquinas nas indústrias vem acompanhado com uma série de benefícios como um melhor aproveitamento de tempo na hora da criação do produto, agilidade na produção, capacidade de produzir peças mais complexas, pesadas e maiores, padronização, maior qualidade no produto final e etc.

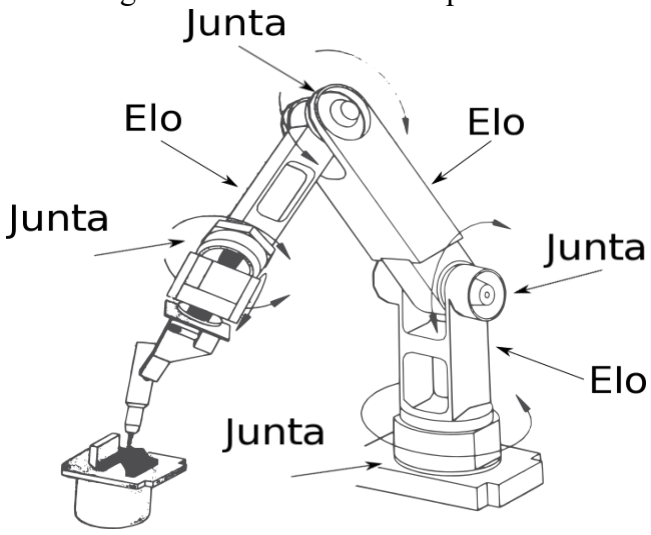
Os manipuladores podem ter diversos níveis de autonomia, a princípio todos tinham uma programação muito repetitiva, ou seja, faziam a mesma atividade repetidamente sem nenhuma alteração, hoje em dia já temos manipuladores mas flexíveis, que são capazes de realizar mais de uma função, em relação a orientação do utensílio que se trabalha e o trabalho aplicada sobre a peça. Todas as ações executadas pelo robô são programadas de acordo com as necessidades de articulação como velocidade, aceleração e direção, para um melhor rendimento.

2.5.1 Características de um manipulador

O manipulador robótico denominado como braço robótico é constituído por uma base, por um punho e os braços. A base é o ponto fixo que sustenta o braço. O braço é toda a parte física composto por elementos chamados de elos, onde os elos são conectados entre si por juntas que realizam os movimentos simples, conectadas aos atuadores que possuem um sistema de controle e propensão sensorial, através dele é realizado todo o movimento do braço. O punho é localizado na extremidade do braço, neste local as juntas são conectadas de forma que fiquem mais próximas entre si, sustentada por uma garra que está localizada na extremidade do braço programada para realizar as atividades desejadas. Na Figura 6 está ilustrada a estrutura do braço.

Para um melhor número de movimentos do braço é preciso definir os graus de liberdade, que está diretamente ligado ao movimento das juntas em função do eixo, caso a junta possua apenas um eixo é definido como um grau de liberdade, quando possui mais de um a junta possui três graus de liberdade. O número de graus de liberdade está associado com a os movimentos do eixo de forma única.

Figura 6: Estrutura do manipulador



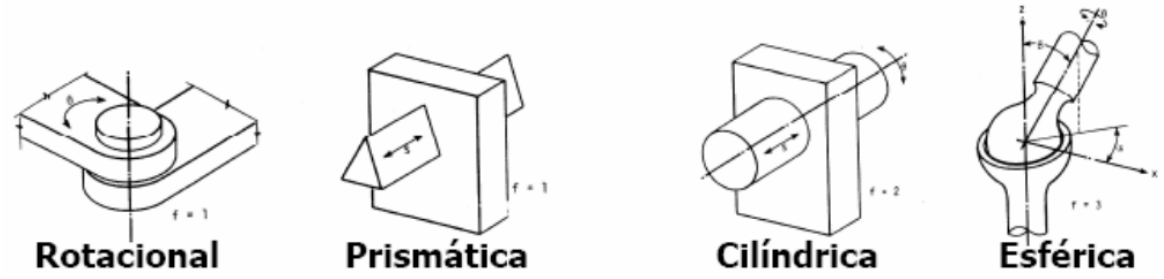
Fonte: Laboratório de eletrônica
http://labdeeletronica.com.br/robos-de-classe/

2.5.2 Juntas

Como explicado anteriormente as juntas são importantes nas junções de braços e elos e para definir os graus de liberdade, é através delas que se determina a quantidade de movimentos do manipulador em diferentes posições, porém, existem vários tipos de juntas, vejamos algumas a seguir, ilustradas também na Figura 7 (Pimenta, Thiago Tavares-2009):

- Juntas Esféricas: Possui três juntas que realiza a rotação em torno de três eixos;
- Juntas Rotacionais: Giram em torno de uma linha imaginária estacionária denominada eixo de rotação. Considerando como um exemplo, os eixo x,y e z, podemos dizer que uma junta é rotacional no eixo z, ou seja, ela terá a liberdade de realizar o movimento de rotação em torno do eixo z;
- Juntas Prismáticas: Possui duas hastes (elos) que se deslizam entre si em um movimento linear;
- Juntas Parafuso: Possui um parafuso e uma porca, seus movimentos são semelhantes os das juntas prismáticas, com uma diferença pois possui movimentos no eixo central. (Pimenta, Thiago Tavares -2009)

Figura 7: Juntas e suas formas físicas.



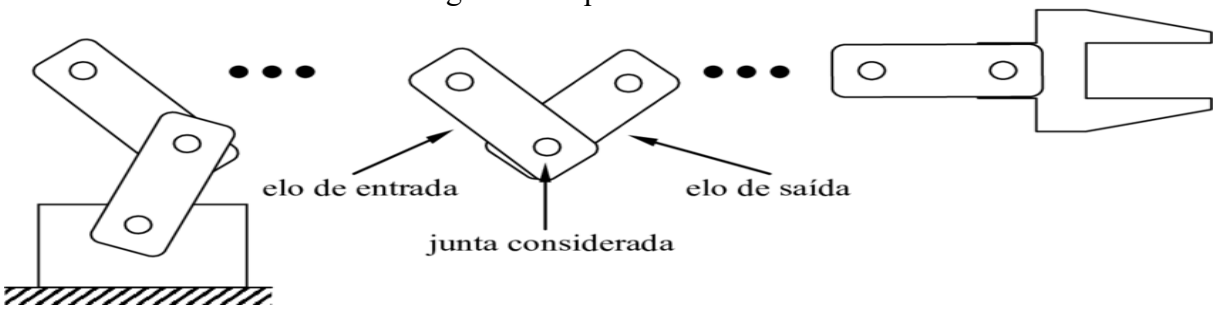
Fonte: Pimenta, Thiago 2009

2.5.3 Elos

O elo é o corpo do manipulador, que podem ter variados tamanhos e formas, e são conectados pelas juntas, uma em cada extremidade (Figura 8). A quantidade de elos que um braço robótico possui são numerados, sendo a base como o elo 0, o próximo elo que possui o

movimento é chamado de número 1 e assim consecutivamente, a quantidade de elos está relacionada com o tamanho do braço.

Figura 8: Sequências de Elos

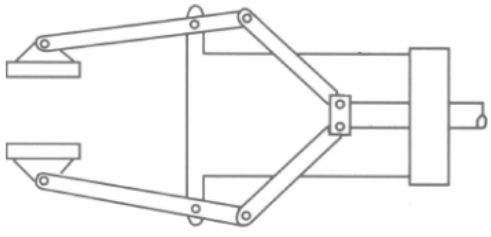


Fonte: Robótica industrial, IME-USP
https://www.ime.usp.br/~adao/ROBOTICAINDUSTRIAL1.pdf

2.5.4 Garras

Semelhante a uma mão humana a garra está sempre conectada no último elo do braço, assim como é mostrado na Figura 9, podendo realizar diversos tipos de tarefas determinadas pelo programador como por exemplo: Soldagem de determinadas peças, descarga de máquinas, pintura e outras funções.

Figura 9: Guarra semelhante uma mão humana



Fonte: Pimenta, Thiago-2009

2.5.5 Atuadores

Os atuadores são encarregados de converter a energia implementada no braço em movimento mecânico, com isso ele se torna o responsável pelos movimentos do manipulador.

Existem vários tipos de atuadores (também conhecido como motores) porém, o mais utilizado em equipamentos de pequeno porte são os eletromagnéticos, que possui alguns modelos como: Servomotor, motor de passo entre outros. Dentre esses motores citados o que é preferentemente utilizado em dispositivos de automação industrial em controle de precisão é o servo motor, pois ele possui várias vantagens como uma melhor precisão no controle, são mais velozes, potentes e possuem uma grande variedade do produto.

2.5.6 Servo motor

São motores de posições controladas, ou seja, atuadores, onde o programador escolhe a posição ou ângulo desejado e o motor corresponde ao ponto esperado. Apesar de se movimentar em diversos pontos alguns desses motores não possuem rotação contínua, trabalham dentro de um limite de ângulos específicos, que se diferenciam de acordo com o tipo de servo. Os ângulos são definidos através de pulsos, sendo cada um deles um ângulo em que o servo deve se posicionar, mantendo-a a mesma posição até ser enviado outro pulso, caso tenha qualquer tipo de pressão sobre o eixo o motor ainda mantém a última posição comandada, isso ocorre devido seu controle de feedback interno que é capaz de trabalhar independentemente. Para simplificar o modo de controle o Motor possui três fios de cores diferentes que correspondem a alimentação positivo e negativo (GND) e um último que funciona como linha de controle de sinal, ou seja, faz a conexão com o arduino.

Tem um funcionamento de acordo com o circuito de malha fechada, pois ele possui internamente dispositivos que envia para o sinal de controle uma informação da posição em que ele se encontra, com isso podendo ser ajustado conforme o sinal de entrada.

Internamente o motor é composto por um circuito de controle que faz a recepção do sinal de comando e ajusta a posição do servo, por um potenciômetro que está conectado ao eixo girando para a ponto desejada, comunicando-se com o circuito de controle para ter informações se o motor atingiu a posição esperada, tendo também um pequeno motor DC interno, e por último as engrenagens para fazer uma redução e ajuste para aumento de torque, como em uma garra por exemplo para mantê-la fechada segurando um objeto o torque do motor é quem mantém aquela posição.

Os servo motores de posição exigem como entrada de referência um sinal elétrico PPM (Pulse Position Modulation) ou PWM (Pulse-Width Modulation), onde a largura do pulso é proporcional à posição desejada.

Existem alguns servos mais avançados que possui controle digital, podendo ser controlado por comunicação serial por exemplo, eles consomem menos energia, tem um torque maior, sendo utilizados em sistemas de controles mais avançados.

O servo é um componente fundamental em projetos de robótica, fazendo-se presente em diversas áreas, sendo responsável por movimentar garras, pernas e braços do robô, responsável por comandar as asas dos aviões através dos flaps no aeromodelismo , responsável por girar as rodas dianteiras dos carros no automobilismo entre outras funcionalidades.

Figura 10: Servo motor.

Fonte: Como usar com Arduino – Servo Motor MG946R

Sendo um dispositivo de controle de posição angular, ou seja, ele gira para direita e esquerda com um ângulo máximo de 180ºgraus (dependendo do fabricante), partindo do eixo 0 ele segue até 90ºgraus para uma determinada direção, e seguindo para o lado oposto fazendo o mesmo mecanismo completando 180ºgraus, travando neste ângulo, quando não se tem nenhum tipo de alteração, pois ele não gira de forma contínua com 360º graus, justamente para fazer a posição angular que conjura em um ângulo de 180ºgraus.

2.6 PLATAFORMA MICROCONTROLADA ARDUINO

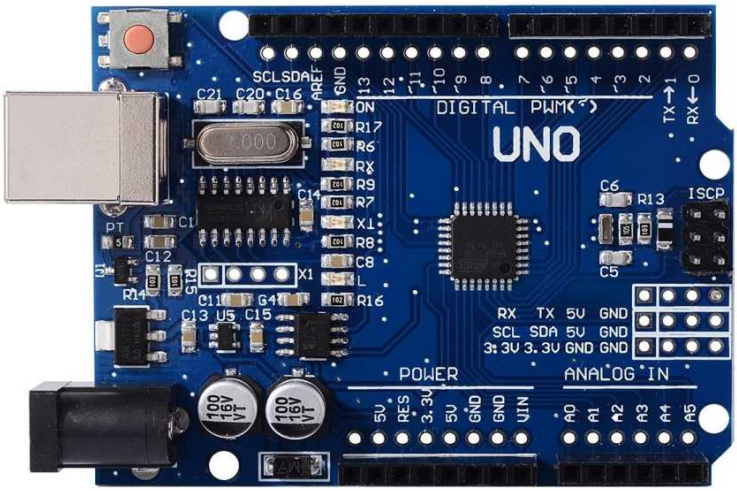
O Arduíno é uma plataforma programável constituída por dois componentes: O software que se configura na parte onde o usuário escreve o que deseja e o Hardware ou placa. O Hardware constitui-se por um microcontrolador Atmel AVR, com linguagem padrão C ou C++, e estruturas de entradas e saídas. Basicamente o arduino é um pequeno computador onde o microcontrolador é responsável por executar o código enviado pelo programador, onde serão processadas as entradas e saídas conectadas a componentes e dispositivos externos (FilipeFlop-2014).

Em termos práticos o arduino é composto por um pequeno chip chamado de microcontrolador onde estão conectadas às entradas e saídas mantendo-as expostas para uma melhor conexão com os dispositivos externos, um oscilador que em frequências determinadas enviam pulsos permitindo sua execução na velocidade exata, uma entrada USB onde possibilita uma conexão com um computador e um regulador de 5 volts.

Um exemplo simples de utilização do arduino é em um sinal de trânsito, no qual seria estabelecido um tempo de 6 segundos para os leds verde e vermelho permanecerem acesos e 4 segundos para o led amarelo, onde terá um intervalo de tempo de 2 segundos entre o acendimento de um led para o outro. Para que isso ocorra o arduino utilizar somente três leds de cores diferentes e três resistores, com isso ao se cumprir um ciclo de operação o arduino gravará esses valores e continuará em um loop infinito, acendendo e apagando os leds.

Existem vários modelos diferentes desta plataforma, podendo variar a quantidade de portas, o tamanho, o processador, a quantidade de bits, a tensão mínima e máxima suportada, o preço no mercado, entre outros. O arduino mega por exemplo possui 54 pinos digitais, uma memória flash de 256 KB, 4 KB de memória EEPROM, já o uno (Figura 11) possui 14 pinos digitais, uma memória flash de 32 KB, uma memória EEPROM de 1KB entres outras particularidades.

Figura 11: Arduino Uno



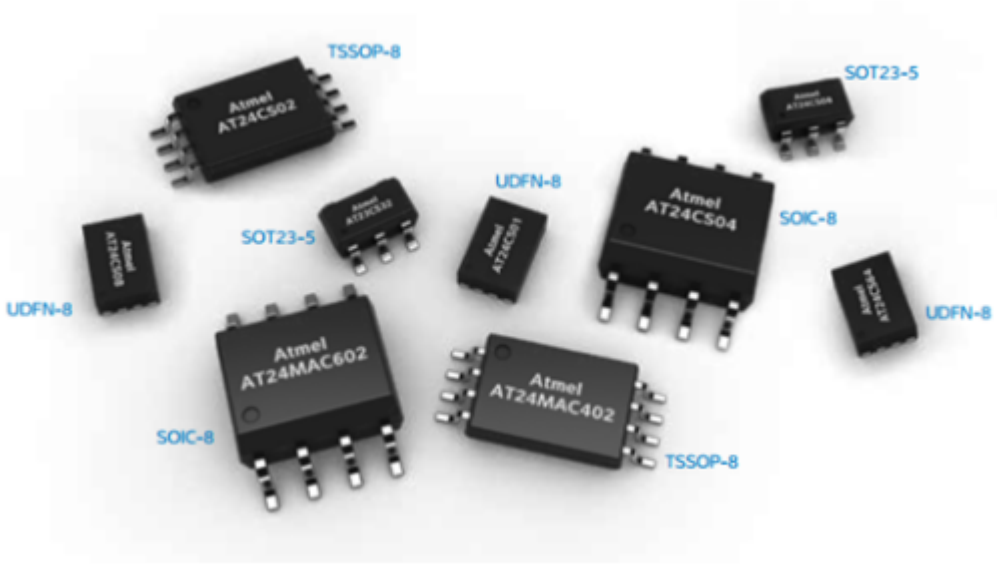
Fonte: https://portal.vidadesilicio.com.br/o-que-e-arduino-e-como-funciona/

2.7 EEPROM

A EEPROM (Electrically-Erasable Programmable Read-Only Memory) é um tipo de memória de leitura programável, na qual se caracteriza por não ser volátil, ou seja, as informações gravadas não se apagam após a suspensão da energia, armazenando continuamente os dados digitais (ver figura 12). As informações contidas nesse endereço de memória podem ser deletadas ou reescritas, porém não sendo possível fazer seleção de dados ou informações para uma nova escrita, no entanto sua exclusão é total, contendo um determinado número de vezes em que os dados podem ser reescritos e apagados.

Esse tipo de memória também pode ser utilizados para ativar dispositivos com outras funções, que não são exclusivamente endereços de memória, dispositivos como sensores de temperatura digital, relógios em tempos reais e outros. Produtos que possuem uma baixa quantidade de memória EEPROM, podendo conter dados de calibração. Utilizado também em jogos que necessite de artifícios que salvem a pontuação ou progresso, com o auxílio da memória flash. Os microcontroladores geralmente do tipo AVR em sua maioria contêm memória EEPROM para o auxílio no armazenamento de dados.

Figura 12: Memórias EEPROM



Fonte:http://ptcomputador.com/Ferragens/ram-cards-motherboards/59305.html

3 OBJETIVOS

3.1 GERAL

- Construção de um sistema robótico que seja capaz de se movimentar autonomamente fazendo o recolhimento, transporte e depósito de peças através de um manipulador

3.2 ESPECÍFICOS

- Desenvolver um sistema autônomo de logística industrial utilizando como base o Robotino 3.0 da empresa Festo. O robô será habilitado a tomar decisões que envolvam menor caminho e menor tempo de execução de tarefas, para isso utilizando o ambiente de programação gráfica interativa Robotino View.
- Desenvolver uma lógica de deslocamento reativa que faça com que o robô se desloque até as coordenadas exigidas desviando dos obstáculos a sua volta independente de mudança
- Apresentar um manipulador robótico independente de quatro eixos que receba a instruções básicas e execute as posições previamente armazenadas para recolhimento e depósitos de objetos.

4 METODOLOGIA

4.1 FUNDAMENTAÇÃO

Primeiramente fizemos todo um referencial teórico acerca do tema escolhido para ser trabalhado no projeto, iniciando sobre o que é robótica, onde se emprega, quando surgiu e etc. Após entender sobre estes conceitos iniciamos uma pesquisa acerca da robótica industrial e seus benefícios, o que inevitavelmente nos exigiu um conhecimento mais aprofundado sobre o conceito de indústria 4.0 e suas principais características, após isso iniciamos os estudos acerca do robô.

4.2 ESTUDO DO ROBÔ

Após a fundamentação, nós começamos os estudos acerca do Robotino 3, que é nosso objeto de estudo neste trabalho. Buscamos entender o funcionamento do robô, para assim, conseguirmos usá-lo para executar as missões estabelecidas. Iniciamos por sua estrutura física, como se dá o funcionamento dos seus motores, sensores, e câmera de forma mecânica e elétrica. Tendo em vista isto, iniciamos o processo de aprendizagem no ambiente de Programação Robotino View, todo o conhecimento que havíamos aprendido na disciplina de Controladores Lógicos programáveis, nos foi essencial, pois foi utilizada e aproveitada no desenvolvimento dos GRAFSET's da programação, já que o ambiente inicial é dividido em steps e em cada "step" é possível agregar uma ação, tomada de decisão, salvamento de valores em variáveis e Etc. Feito isso demos início ao estudo da movimentação do robô

4.2.1 Estudo da movimentação

Inicialmente nós exploramos a plataforma Robotino View para encontrar e compreender o funcionamento dos blocos relacionados aos motores e à movimentação dele. Este robô é omnidirecional, ou seja, a movimentação dele pode ser feita em vários ângulos o que o torna muito mais útil quando o objetivo é encontrar o melhor caminho para chegar em determinado ponto. Encontramos o bloco chamado Drive System, ele é a representação todos os motores do robô. Este bloco permite o controle tanto de um motor por vez, como também

dos 3 simultaneamente, o último é o que nós costumamos utilizar, controlado pelo bloco “Position Drive” que será comentado mais à frente.

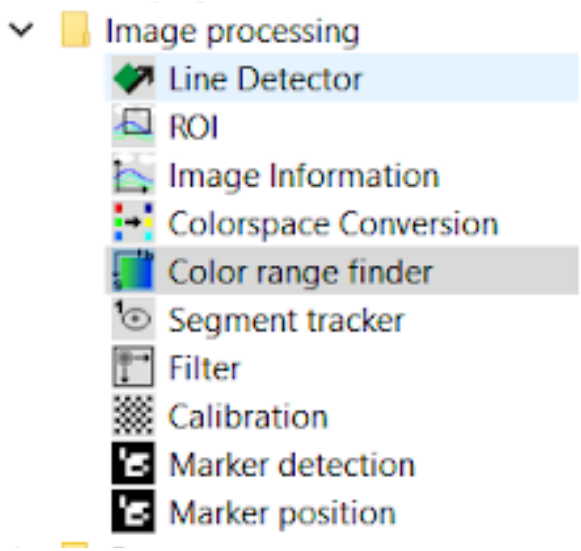
4.2.2 Estudo do position driver

O “Position Driver” do português “Navegador de posição” é um bloco que nos permite fazer com que o robô se desloque até determinado ponto controlando por sí, este controla o bloco chamado Drive System. Para isso ele faz o cálculo usando variáveis como a distância desejada, localização atual, rotações dos motores e velocidade. Estes fatores podem ser facilmente controlados, escolhemos o ponto o qual desejamos que o robô se desloque e conectamos ao bloco também, um outro bloco, chamado “Odometry”, que é basicamente um bloco do odômetro, porém, ao invés de mostrar a rotação de cada motor, este armazena as posições X e Y. Depois disso os motores se movem nos eixos X e Y de forma direta até o ponto antes escolhido. Valores de velocidade podem ser mudados nas propriedades do bloco, onde é permitida a alteração da velocidade máxima de acordo com a distância máxima, assim como para a mínima, também é possível preencher a tabela com mais pontos melhorando a precisão da velocidade. Em seguida é possível um gráfico que é calculado pelo próprio software que mostra a função que foi formada de acordo com as posições.

4.2.3 Estudo da câmera

A imagem é um importante indicador nas tomadas de decisão do robô, pensando nisso, começamos os estudos acerca da câmera buscando ferramentas para suprir as nossas exigências. O Robotino 3 já vem uma câmera que pode ser posicionada no local onde for conveniente, além dessa, a plataforma Robotino View permite que o programador utilize várias câmeras em seu código, que pode ser adicionada por conta própria, através das entradas USB’s existentes no robô. Os blocos que utilizam as câmeras estão dispostos dentro da pasta “Image Processing”, como pode ser observado na Figura 13.

Figura 13: Recorte que mostra os blocos presentes em “Image Processing”.



Fonte:Autores

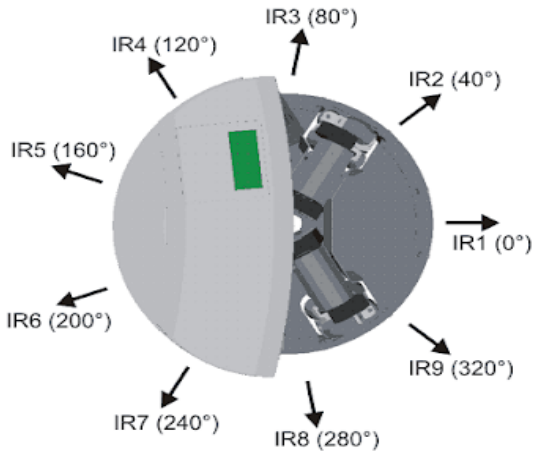
Os blocos possuem funções bem específicas e ao todo eles somam 10, cada um com uma forma diferente de tratar a imagem recebida. O primeiro é o “Line Detector”, ele utiliza a câmera para detectar linhas, como o seu nome já diz, sua saída é booleana e aponta se foi ou não encontrada a linha. O bloco “ROI” (Region Of Interest) é utilizado para recortar uma área importante para o código, ou seja, o programador escolhe uma área da imagem para ser ampliada na saída desse bloco. O “Image Information” é uma função que detecta os dados da imagem em questão, a saída deste bloco representa os valores em pixels de altura e largura da imagem. No “Colorspace conversion” acontece uma conversão no padrão de cores da imagem de entrada. O bloco “Color range finder” é um localizador de cores, como seu nome revela, nós selecionamos uma área da imagem e o bloco salvará em um vetor os valores máximo e mínimo de pixels naquela região, em sua saída estará uma imagem em tons de cinza com a parte selecionada em destaque. “Segmente Tracker” é um bloco que tem como função localizar um segmento específico da imagem, dando como saída os valores da cordenadas X e Y, a área, a quantidade de segmentos encontrados e a imagem em escala cinza com os segmentos destacados. No bloco “Filter” a imagem passará por uma espécie de edição, onde algumas de suas informações serão alteradas para enfatizar a área importante para o código. Por último estão os blocos “Maker detection” e ”Marker position” que estão relacionados a detecção e posicionamento com relação a QR codes, que mencionaremos mais a frente.

4.3 APLICAÇÕES

4.3.1 Canto

Para o início de todas as movimentações foi necessária uma referência inicial, para que o position driver tivesse uma localização e pudesse realizar os movimentos requeridos corretamente, verificamos que fazia-se necessário um alinhamento, para que a movimentação se tornasse mais precisa já que sem isso era possível que a cada saída do robô, houvesse um resultado com diferentes posições X, Y e de angulação. Então fizemos medições e constatamos um raio de 24,3 cm e uma angulação de 40 graus entre cada sensor como ilustra a Figura 14.

Figura 14: Distribuição dos Sensores infravermelhos.



Fonte: Sistema Olimpo.

Para centralizar a 30 centímetros da parede lateral Direita e igualmente distante da parede traseira, para isto fizemos cálculos com base na ilustração contida na Figura 15. Primeiramente, por serem simetricamente posicionados, os sensores 5 e 6 deveriam ficar a mesma distância para que o robô estivesse em paralelo em relação à barreira traseira, traçando uma bissetriz, temos 2 triângulos retângulos. Utilizamos das propriedades Trigonométricas para calcular a distância desejada:

$$\frac{CA}{HIP} = COS\ 0°$$

$$\frac{30\ cm}{\overline{AC}} = COS\ 20°$$

$$\frac{30\ cm}{\overline{AC}} = COS\ 20°$$

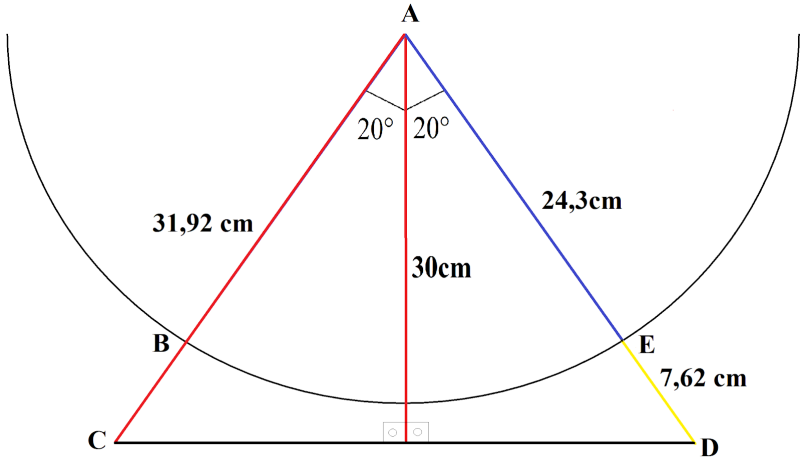
$$\frac{30\ cm}{COS\ 20°} = \overline{AC}$$

$$\overline{AC} = 31,92cm$$

$$\overline{AB} = \ Raio\ = 24,3cm$$

$$\overline{AC} - \overline{AB} = 7,62cm$$

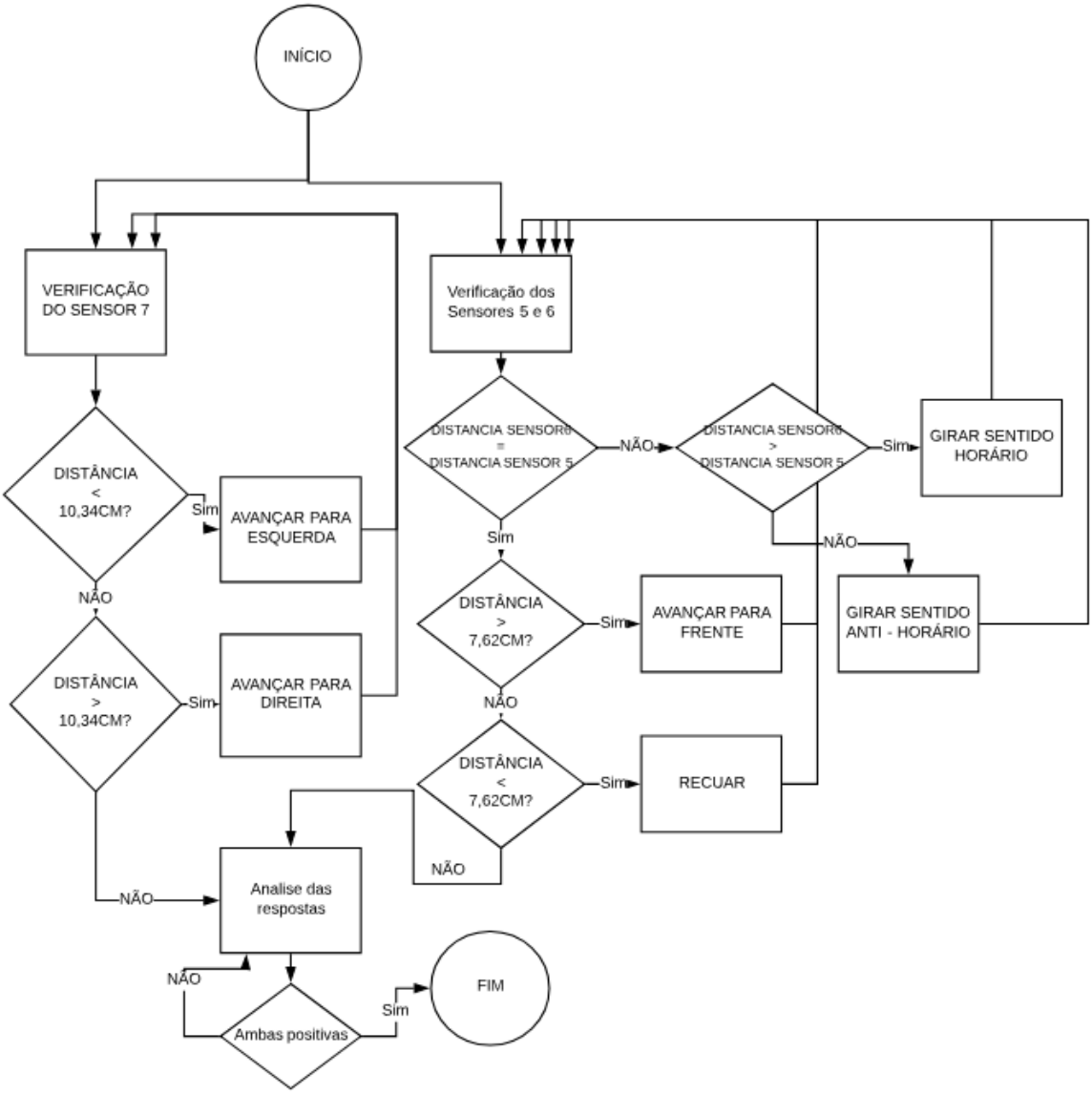
Figura 15: Imagem representativa dos dados métricos acerca do Robotino.



Fonte:Autores.

Então utilizamos a mesma propriedade para calcular a distância ideal para o sensor 7, e esta resultou em 10,34cm. Após isso, desenvolvemos um fluxograma para a escrita do código por meio de blocos, este está mostrado na Figura 16.

Figura 16:Fluxograma da lógica utilizada no Canto



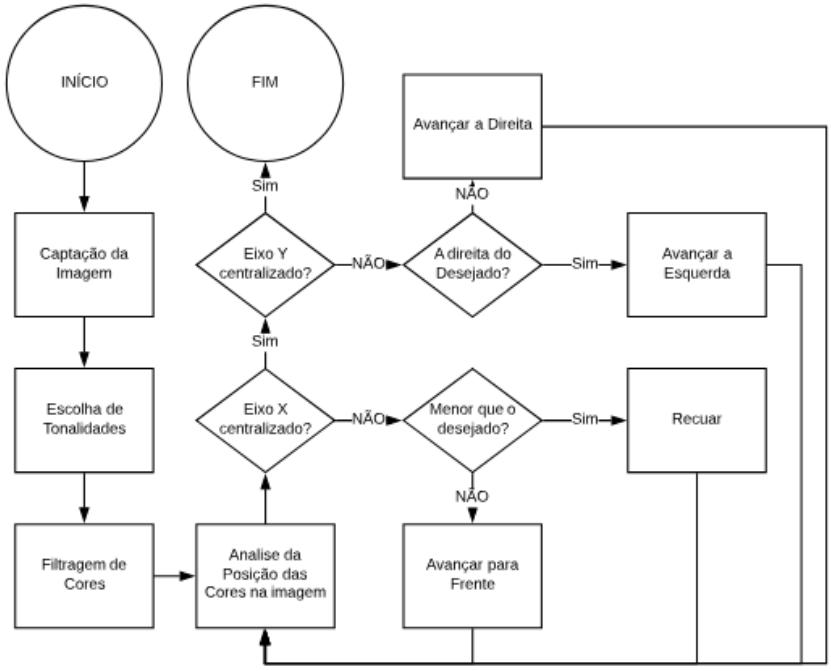
Fonte:Autoras

4.3.2 Pega puck

Para realizar a retirada e movimentação das peças fazia-se necessário que o robô tivesse capacidade de encontrá-la e encaixá-la em seu anexo e então arrastá-la para o local de destino. Para isso utilizamos um simples tratamento de imagem, que filtra uma determinada cor. Nós selecionamos as melhor disposição de cores encontrada da captação da imagem de forma o número de falhas e tempo de processamento seja reduzido, após a identificação da

cor utilizamos uma lógica de deslocamento descrita no fluxograma da Figura 17 que centraliza esta cor selecionada, de forma a encaixar o disco para passar à próxima etapa.

Figura 17: Fluxograma sobre Lógica do programa Pega Puck.



Fonte: Autores.

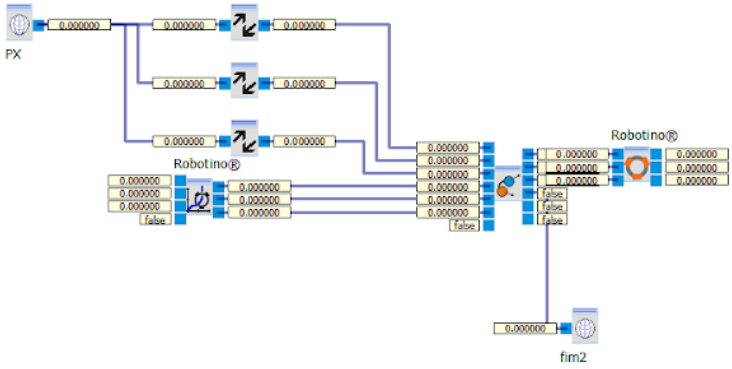
4.3.3 Posição por posição

Após concluirmos os estudos iniciais acerca das habilidades do Robotino 3 , nós começamos a implementação do nosso primeira programa. A lógica de movimentação utilizada por nós neste primeiro momento foi a de “Posição por posição”, ou seja, as posições por onde o robô passaria foram definidas anteriormente. Para que essa lógica desse certo nós traçamos uma espécie de mapa com com todas as posições necessárias para fazer o robô chegar ao seu destino. Fizemos a medição das coordenadas de cada um dos pontos e listamos os valores de x, y e do ângulos dos mesmos.

Na plataforma Robotino view, nós atribuímos por meio de Steps sequenciais cada uma dessas posições às variáveis com o nome PX. Em seguida, atribuímos em tabelas do bloco associadas ao “Position Driver” as coordenadas das posições medidas anteriormente, para

logo depois o “Drive System” receber os valores e direcionar os motores. O step principal dessa programação se chama “LIVRE”, e ele será mostrado na Figura 18.

Figura 18: Recorte do Step “LIVRE”.



Fonte: Autoras

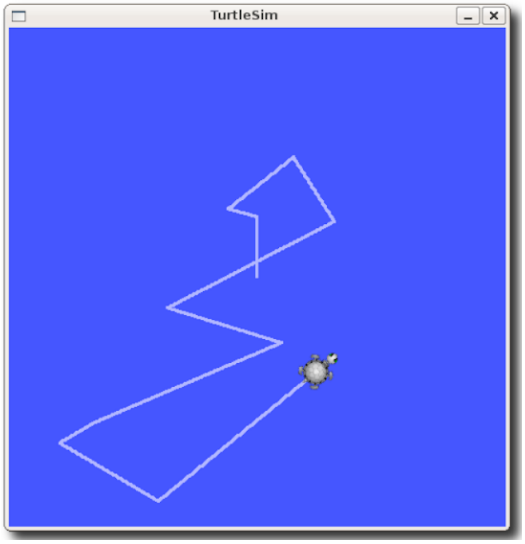
4.3.4 Ros

Inicialmente fizemos a instalação do software e ambientalização com o sistema do framework que é em si diferente da demais programações já utilizadas pelos componentes do trabalho, a seguir iniciamos os estudos acerca da estrutura deste sistema que consiste na comunicação de pacotes por meio de mensagens que são enviadas nos mais diferentes tipos de tópicos. Em seguida nos mantivemos a entender as especificidades destas estruturas voltadas para a plataforma robotino. Foi executado um estudo acerca de lógicas de deslocamento e testes na prática,

Tivemos a oportunidade de participar de um curso de capacitação ofertado pela Universidade de Campinas, (UNICAMP) em Parceria com a Empresa Festo realizada no sede da FESTO Brasil, onde foi ensinado conhecimentos básicos acerca do Framework, principalmente sobre a criação de pacotes como o exemplo apresentado na Figura 19 e implementação de pacotes já existentes específicos para Robotino, como também o entendimento sobre tópicos e suas funcionalidades, a partir disso iniciamos um estudo com o intuito de utilizá-lo para movimentação do robô.

Executamos alguns programas tutoriais encontrados no sítio do Framework disponivel no endereço http://wiki.ros.org/pt_BR/ROS/Tutorials, utilizando principalmente de simuladores como mostrado na Figura 15, fazendo a implementação de pacotes já existentes.

Figura 19: Imagem de simulador rodando programa de Tutorial.



Fonte: wiki.ros.org

Após estudos mais aprofundados verificamos que não era possível no momento utilizar tal método pois havia uma maior complexidade e carecia conhecimentos de programação abordando Orientação a Objetos de forma muito avançada que infelizmente não são ofertados pela grade do curso de eletrônica, e não havia tempo hábil para a implementação desta ferramenta. Então optamos por desenvolver nossa lógica no ambiente de programação Robotino View.

4.4 APLICAÇÃO EM COMPETIÇÃO

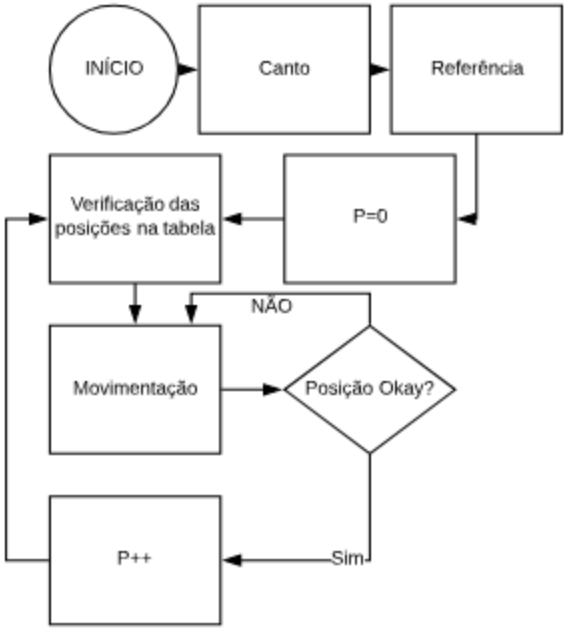
No decorrer da construção desse trabalho, nós participamos de algumas competições que nos permitiu somar bastante na nossa pesquisa e principalmente aplicar o que foi desenvolvido nos meses de estudo.

4.4.1 Larc/Cbr

A LARC (Latin American Robotics Competition) é uma competição focada no uso de robôs móveis autônomos aplicados a processos logísticos. O foco principal da nossa pesquisa está diretamente relacionado ao conceito dessa competição, o que tornou a nossa participação nela bastante proveitosa.

A lógica de movimentação escolhida por nós para realizar a prova foi a de deslocamento posição por posição (Figura 20) que foi explicada mais a cima. Optamos por essa lógica pois o acesso a arena de prova estava liberada para treinos, desse modo, se tornou possível o mapeamento dos pontos no qual o robô deveria se locomover. Além de se mover pelos obstáculos, para pontuar, o robô precisava encontrar o puck no estoque e com a ajuda de um suporte anexado na frente do robô arrastá-lo pelo chão até o lugar destinado para a entrega, ele deve fazer isso várias vezes dentro de 20 minutos, que é o tempo de produção.

Figura 20: Recorte de código utilizado para deslocamento ponto-a-ponto.

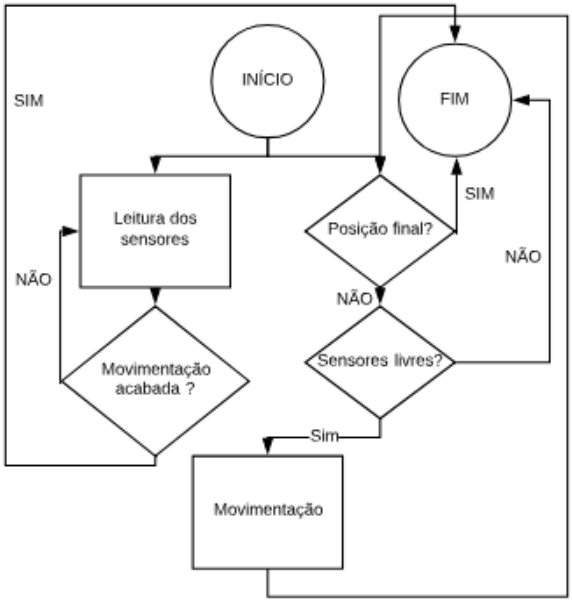


Fonte: Autoras.

4.5 MOVIMENTAÇÃO REATIVA

Em seguida, optamos por desenvolver uma lógica que tivesse característica reativa (ilustrada na Figura 21**)** como era o objetivo inicial do projeto para que não houvessem grandes problemas na mudança do cenário, garantindo a redução de acidentes com máquinas, outros robôs ou até mesmo seres humanos que pudessem trafegar dentro deste cenário, então a cada obstáculo detectado ou a chegada do ponto escolhido é feita a saída da rotina de movimentação para o desvio do obstáculo ou o próximo passo, respectivamente. Adicionamos também programas de reconhecimento de QR-Codes para que houvesse uma melhor distinção entre a as diferentes máquinas localizadas no ambiente.

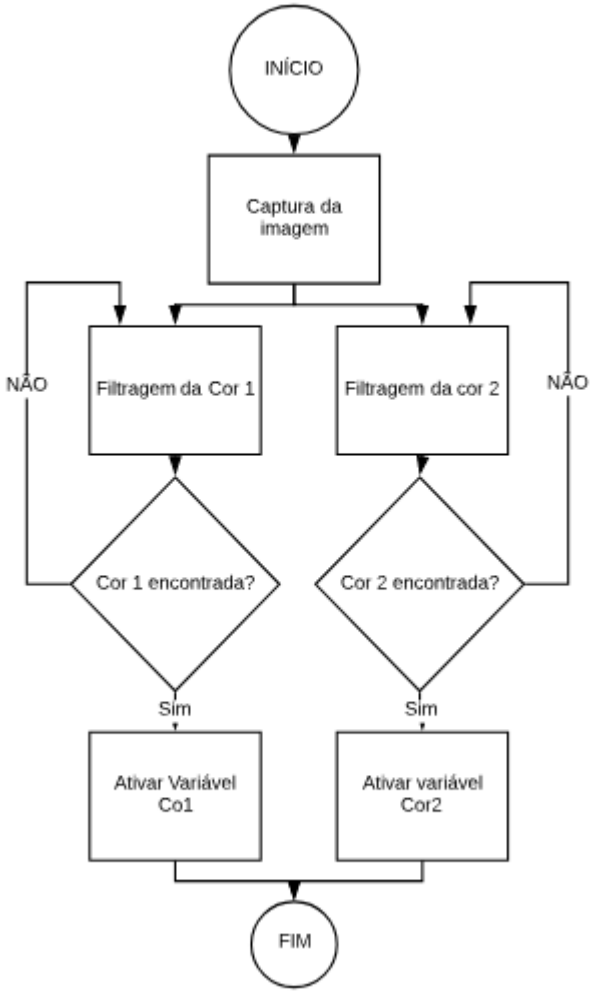
Figura 21**:** Lógica de movimentação Reativa.



Fonte: Autoras.

Além disso, desenvolvemos também um programa para a distinção entre as peças, que tem sua lógica mostrada na Figura 22, escolhemos utilizar como método a diferenciação entre as cores dos discos utilizados e posteriormente fizemos a integração de todo esse sistema com o braço robótico que foi projetado de forma independente para um melhor desempenho.

Figura 22: Lógica de distinção de peças.



Fonte: Autores.

4.6 METODOLOGIA DO BRAÇO ROBÓTICO

Uma das primeiras coisas a serem pensadas na hora da preparação do braço robótico foi a escolha do motor, pois era necessário ter um alto torque e resistência. Através de estudos foi possível selecionar o servo motor MG946R (que é uma versão mais atualizada do modelo MG945R), mostrado na Figura 23. Ele tem como característica o alto custo-benefício e uma rotação contínua, possuindo um torque de 10,5 Kg/cm para uma tensão de operação de 4,8 VDR e de 13 Kg/cm em uma tensão de 6,6VDC, possuindo engrenagens de metal, contendo uma modulação analogica, operando em uma velocidade de 0,20s/60ºgraus (4,8 VDR)/ 0,17s/60ºgraus (6,6VDC). Como visto o motor possui três terminais sendo o laranja o que recebe o pulso, ou seja, fica conectado ao arduino, o marrom conecta no GND, e o vermelho é

o positivo, sendo conectados no protoboard com um auxílio de uma carga ou fonte. (Oliveira, Euler)

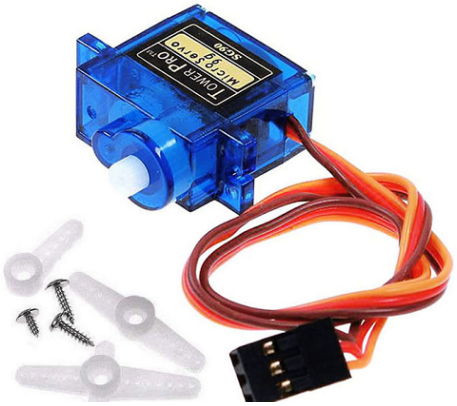
Figura 23: Servo motor MG946R

Fonte: https://portal.vidadesilicio.com.br/o-que-e-arduino-e-como-funciona/

Ao observarmos o servo escolhido podemos perceber que ele tem um tamanho que não se adequaria aos parâmetros da garra, pois como ela iria ficar na superfície dificilmente uma estrutura simples que sustentaria os motores e elos, suportaria o peso de mais um motor em um dos extremoz, com isso foi necessário a implementação de um micro servo diferenciado, tendo como modelo o Servo 9g Tower Pro SG90 (ver Figura 24).

Este tipo de servo possui algumas das mesmas aplicações do MG946R como por exemplo em aeromodelismo, em principal em projetos feitos com arduino ou outras plataformas microcontroladas, que são aplicações em que sua maioria não necessitam de alto torque e muita resistência. Como características principais que auxiliaram na escolha tem um torque de 1,2 Kg/cm até 1,6 Kg/cm que apesar de ser baixo comparado com o outro servo a tensão de operação é a mesma sendo igual a 4,8VDC e 6VDC, suas engrenagens são feitas de Nylon (fibras artificiais, com enorme capacidade de resistência e elasticidade), possuindo uma capacidade de rotação de 180ºgraus (Oliveira, Euler).

Figura 24: Servo motor 9g Tower Pro SG90.

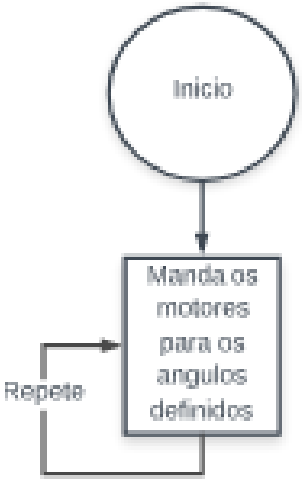


Fonte:http://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-micro-servo-motor-sg90-9g/

Como um primeiro contato com os servos foi necessário a utilização de um código exemplo chamado Knob fornecido pelo arduino, que através dele podemos testar o funcionamento dos motores em vários ângulos diferentes. O programa exemplo é bem simples, a biblioteca <servo.h> é declarada automaticamente pelo código, em seguida declara a variável do motor , roda o programa de acordo com a escolha do pino onde está o motor no arduino, em seguida ler o código e executa o movimento do motor de acordo com o ângulo escolhido.

Primeiramente foram feitas testes com apenas um servo (Figuras 25 e 26), de modo que ficasse mais fácil a compreensão do código, tendo bons resultados foram testados os três motores de uma vez obtendo posições diferentes, mesmo com o incremento de mais 2 motores(Figura 27) o código funcionará da mesma maneira pois possui a mesma lógica ilustrada na Figura 25.

Figura 25: Lógica de teste para os motores.



Fonte: Autoras

Figura 26: Teste no arduino com 1 servo.

Knob.ino

```
1  #include <Servo.h>
2
3  Servo myservo1;  // create servo object to control a servo
4
5  void setup()
6  {
7    myservo1.attach(47);  // attaches the servo on pin 9 to the servo object
8
9  }
10
11 void loop()
12 {
13   myservo1.write(60);
14
15 }
```

Fonte:Autoras.

Figura 27: Teste no arduino com os 3 servo.

```
#include <Servo.h>

Servo myservo1;  // create servo object to control a servo
Servo myservo2;  // create servo object to control a servo
Servo myservo3;  // create servo object to control a servo


void setup()
{
  myservo.attach(47);  // attaches the servo on pin 9 to the servo object

  myservo1.attach(14);  // attaches the servo on pin 9 to the servo object

myservo2.attach(18);  // attaches the servo on pin 9 to the servo object

}

void loop()
{
  myservo1.write(20);
  myservo2.write(70);
   myservo3.write(40);

}
```

Fonte:Autoras.

4.6.1 Desenvolvimento da garra

Como citado anteriormente a garra foi projetada de forma independente, ou seja, com o auxílio do orientador foi possível obter os compartimentos do braço robótico através da impressora 3D presente no IFRN. Foi necessário a utilização do Solidworks (uma ferramenta de CAD (Computer Aided Design, em português: desenho assistido por computador) que é um software avançado de criação de diagramas esquemáticos, onde os usuários além de criar diagramas podem gerar relatórios automáticos e listas de materiais. O programa é utilizado para desenvolver produtos que estão inseridos no dia a dia como braços robóticos, carros , aviões entre outros.

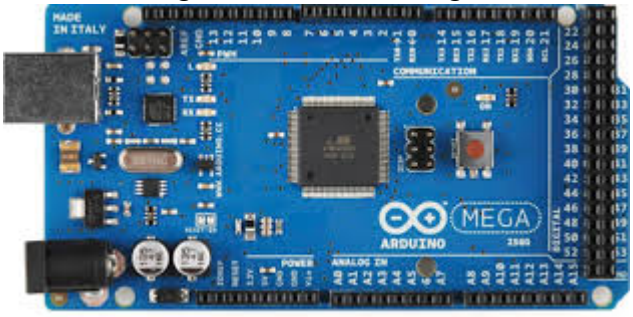
Através dele foi possível definir os padrões da garra como tamanho, espessura e diâmetro, para finalizar a impressão após ser feito todo um diagrama no Solidworks, o desenho foi enviado para outro programa chamado CURA que é um software open source, ou seja, ele é gratuito, e foi utilizado para deixar as figuras mais uniformes por tem um design mais sofisticado e facilitar no envio para a impressora.

4.6.2 Escolha do Arduino

Para o desenvolvimento do algoritmo no projeto, foi utilizado a linguagem de programação do arduino, onde foi realizado uma breve pesquisa sobre o tal, analisando seu funcionamento, conexões e componentes, tendo em vista que esta plataforma iria facilitar muito a progressão do projeto, pois ela possui uma grande variedade de aplicações e um excelente software e hardware, nos quais são fundamentais para a prototipagem do sistema.

O arduino possui uma ampla acessibilidade quando se trata na implementação de novas bibliotecas, já que existem algumas comunicações já prontas dentro do próprio sistema, tornando simples e rápido a continuidade do projeto. Após o breve conhecimento, foi feita a escolha do arduino Mega (Figura 28), onde foi realizado todo o programa dividido em etapas como o funcionamento dos motores e a gravação do código.

Figura 28: Arduino Mega.

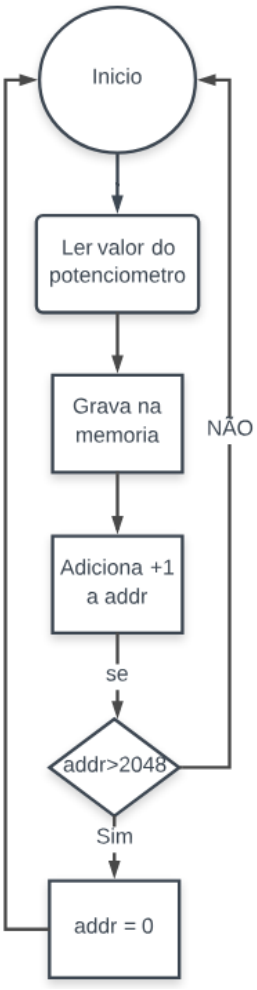


Fonte:https://www.arduino.blog.br/diferenca-arduino-uno-mega-nano-mini.html

4.6.3 Gravação

O desenvolvimento na aprendizagem da gravação foi feita da seguinte maneira, como dito antes o arduino possui comunicações e códigos exemplos, desta forma utilizamos como teste inicial dois exemplos da EEPROM para leitura e gravação (Figuras 29 e 30). Nesses códigos são explicados e demonstrados como se utiliza as funções EEPROWrite que está responsável pela gravação, onde utiliza a porta analogica para ler o valor de um potenciômetro e grava esse valor na memoria, e a EEPROMRead que responsável por ler o o que esta salvo dentro da memória eeprom.

Figura 29: EEPROM.Write.



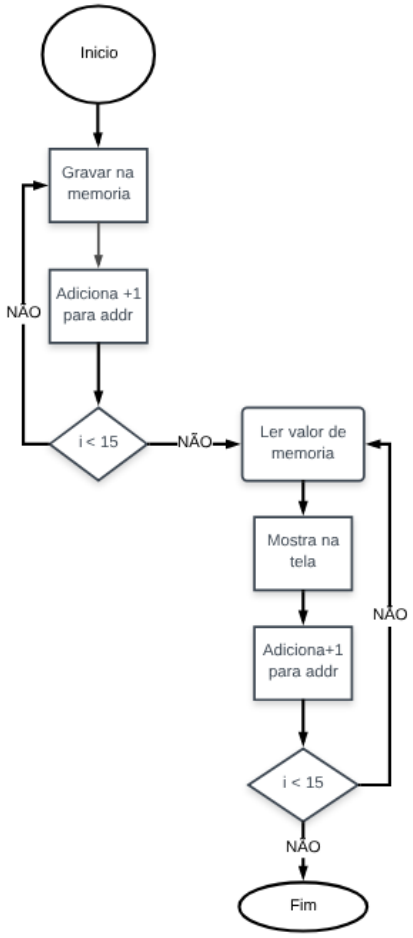
Fonte: Autoras

Figura 30: EEPROM.Read

Fonte: Autoras

Foram feitas algumas modificações para que o código exemplo melhor se adapte-se aos critérios do projeto. O fluxograma abaixo (Figura 31) explica de maneira sucinta a metodologia usada para a gravação dos ângulos no endereço de memória, de forma que após definidos, são gravados na memória 15 valores que equivale ao número de posições possíveis, em seguida esses valores são lidos e mostrados no monitor serial.

Figura 31: Fluxograma de gravação dos ângulos.

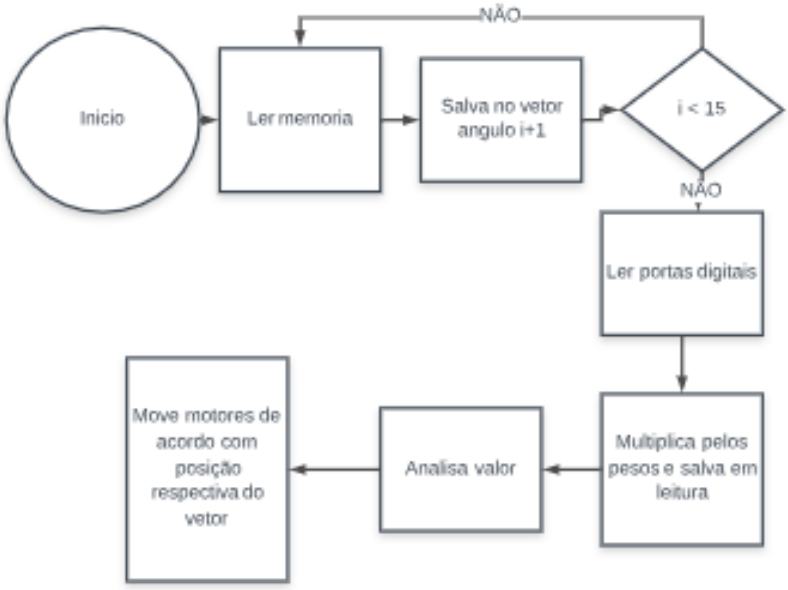


Fonte: Autoras

Como código principal responsável pelo acionamento dos motores temos um fluxograma a seguir onde demonstra os principais pontos do código. De início podemos observar que os valores gravados no código anterior são lidas e salvas em um vetor, caso esses valores sejam menores que 15, o código volta para ler o proximo endereço de memoria, mas se o valor lido for igual a 15, segue para o próximo bloco que está responsável por ler as portas digitais do arduino, em seguida multiplica esse valor pelos pesos que são 1, 2 e 4 que correspondem a quantidade de posições do manipulador, logo após salvam os resultados em leitura, em seguida o valor de leitura é analisado, sendo enviado para os servos os angulos

correspondentes á posição específica de forma a mover-los para a posição esperada de acordo com os ângulos salvos no vetor.

Figura 32: Fluxograma do código manipulador



Fonte: Autoras

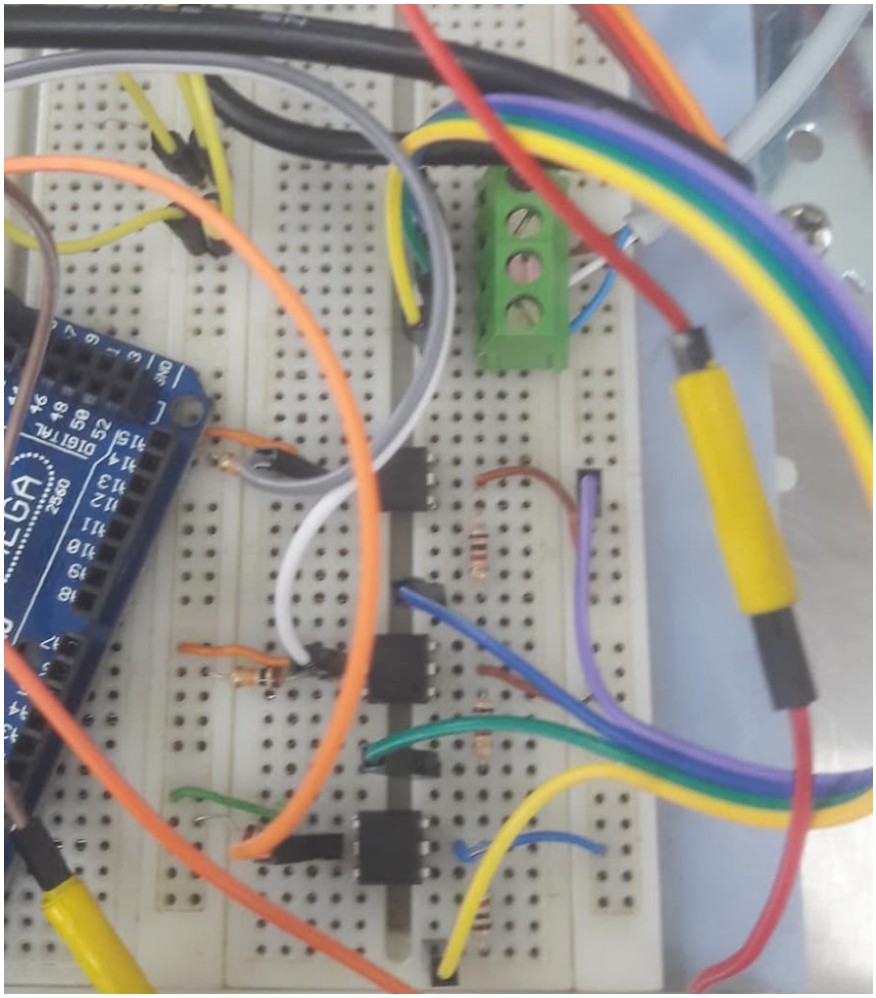
4.6.4 Circuito optoacoplador

A saída digital do robotino deveria estar ligada nas entradas digitais do arduino, porém a saída digital do robotino é de 24 volts, caso esse valor de tensão seja enviado para o arduino através das portas ele será danificado, sabendo disto, foi necessário a implementação de um circuito para unir ambas as partes, escolhemos utilizar um circuito optoacoplador (Figura 33), que tem a função de isolar sinais do circuito quando possuem tensões diferentes. Internamente o optoacoplador possui um diodo emissor de luz (LED) de um lado e um fototransistor bipolar do outro, tendo um isolamento entre eles, responsável por separar as tensões de 5 e 24 volts.

O circuito funcionará da seguinte forma, sempre que o robotino for conectado e mandar uma mensagem com o valor 1 para um dos optoacopladores, o sinal de saída que chegará no arduino é negativo, caso a mensagem enviada seja 0 a saída (sendo a entrada do arduino) converte-se em positiva, pois o fototransistor interno é do tipo PNP. Para ajudar a manter todos os componentes em perfeitas condições foi necessário implementar alguns

resistores para auxílio na limitação da corrente junto com uma fonte de 5 Volts para alimentar os motores e parte do circuito Optoacoplador.

Figura 33: Circuito optoacoplador.



Fonte: Autoras

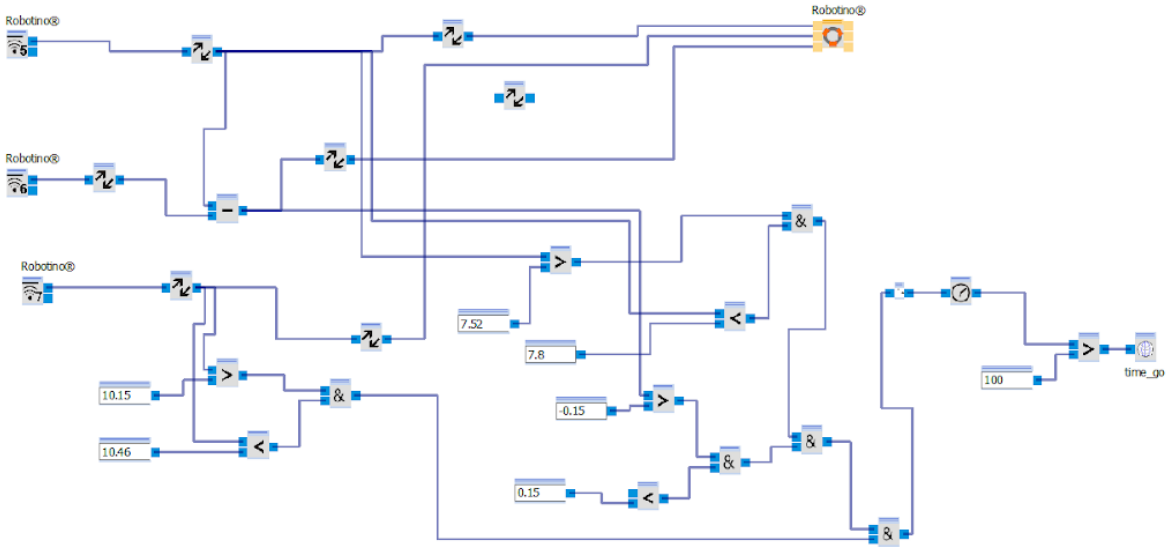
5 RESULTADOS E DISCUSSÕES

5.1 PARTE 1: ROBOTINO

5.1.1 Canto

Após ter feito os cálculos necessários, obtivemos com precisão a distância que cada sensor infravermelho deveria ficar da base para que o robô estivesse devidamente alinhado. Aplicando essa lógica de referenciação na plataforma Robotino View (Figura 34), nós conseguimos tornar a movimentação do Robotino muito mais precisa. Além disso, a referenciação pode ser utilizada em qualquer canto da arena, com angulação de 90º.

Figura 34: Recorte do código de alinhamento com o canto.



Fonte: Autoras.

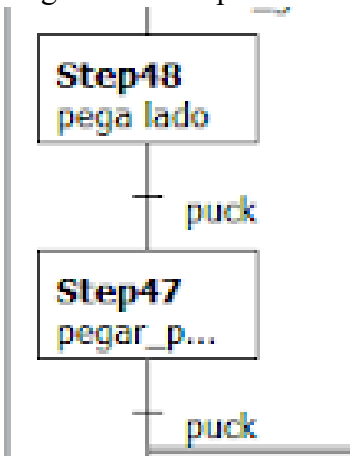
5.1.2 Posição por posição

Ao desenvolvermos a lógica de deslocamento de posição por posição decidimos colocar em prática e então aplicamos os códigos nas tarefas apresentadas na competição a qual participamos, denominada LARC/CBR 2018 (Latin America Robotics Competition/Competição Brasileira de Robótica) na categoria Logistics, neste ano a prova continha duas fases, uma delas compartilhada, onde o robô necessitava recolher a peça arrastando-a no chão e deixando-a em um lugar demarcado.

5.1.3 Pega Puck

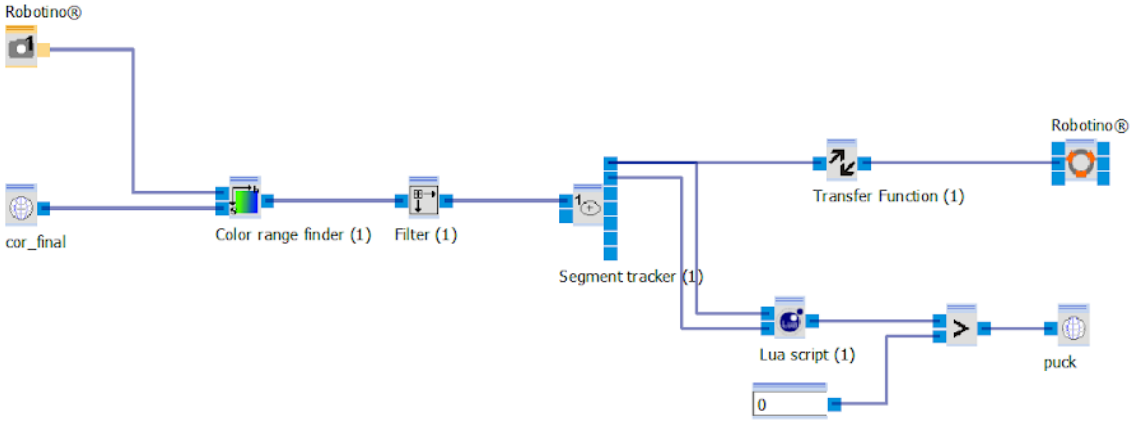
Para o recolhimento da peça preferimos utilizar duas etapas, dividindo-as em steps diferentes (Figura 35) no primeiro ilustrado pela Figura 36, utiliza os blocos para fazer o alinhamento do robô de forma somente horizontal em relação a imagem captada pela da câmera (step 48, pega_lado), e na Figura 37 um avanço vertical como também um pequeno ajuste horizontal é feito pelos blocos do step 47, nomeado de pega_puck.

Figura 35: Recorte de código utilizando para fazer o encaixe do disco.



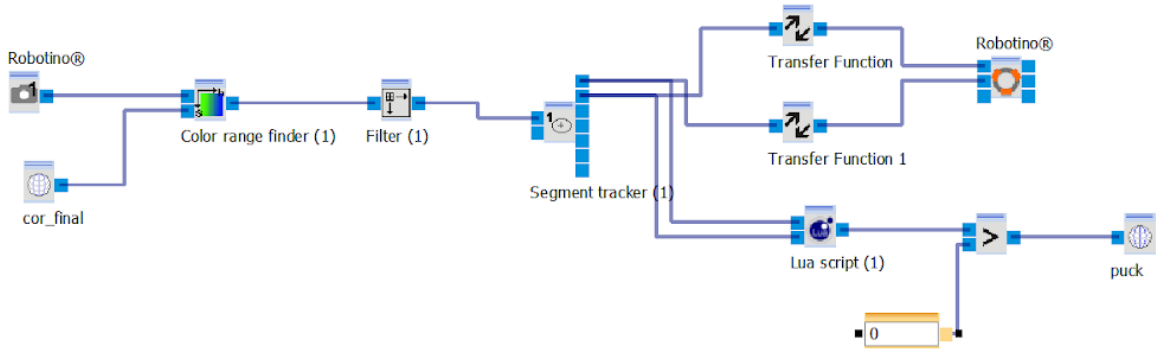
Fonte: Autoras.

Figura 36: Recorte de código retirado de dentro do Step 48 “pega_lado”.



Fonte: Autoras.

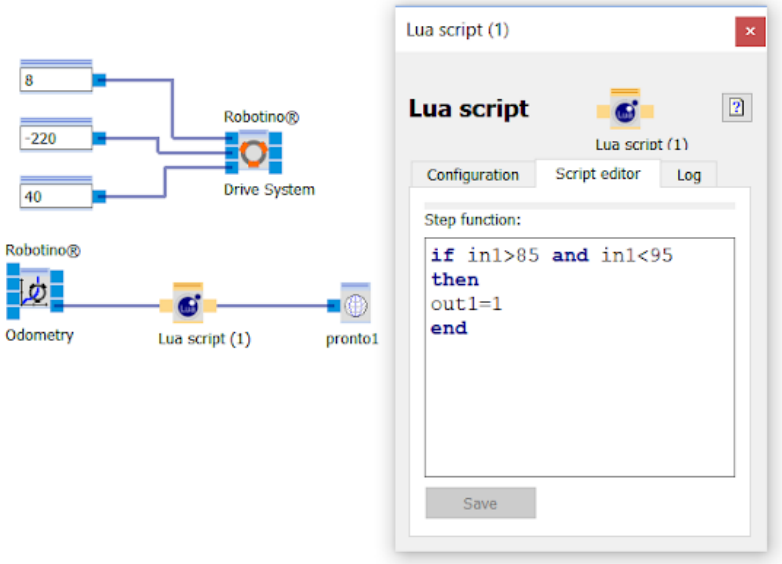
Figura 37: Recorte de código retirado de dentro do Step 47, “pega_puck”.



Fonte: Autoras.

Após encaixe da peça no anexo utilizamos o programa nomeado como “Translacional1” mostrado na Figura 38, para fazer com que o robô fizesse o movimento de translação ao redor do Disco sem necessitar do uso de câmeras, diminuindo assim as possíveis falhas que poderiam ocorrer por influência da Luz. Utilizamos a lógica de deslocamento ponto-a-ponto e de recolhimento de peças em em ambas as provas da competição, com isso alcançamos o 3° lugar na categoria.

Figura 38**:** Recorte de código utilizado.

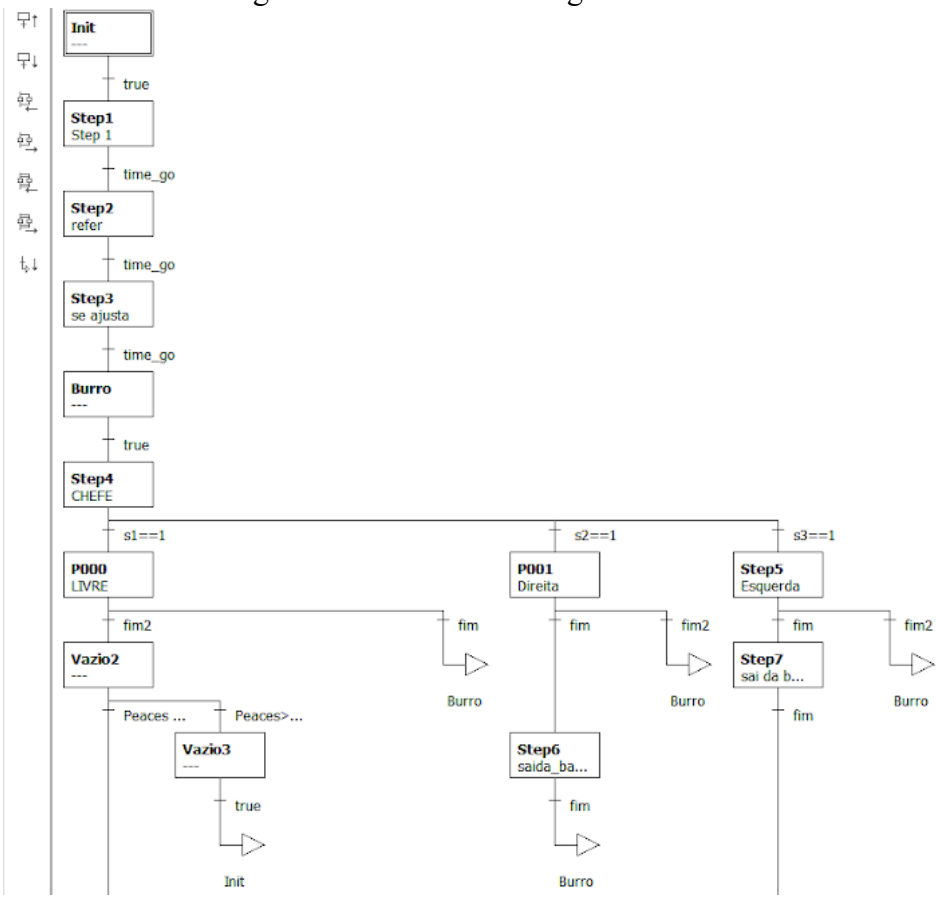


Fonte: Autoras.

Após isso partimos para lógica reativa que resultou em um GRAFSET com o formato mostrado na Figura 39. No step1 é realizado um alinhamento do robô em relação a parede, no seguinte

são salvas as suas referências, no step3 fizemos com que o robô se afastasse da parede para que não influenciasse no início na lógica de desvio.

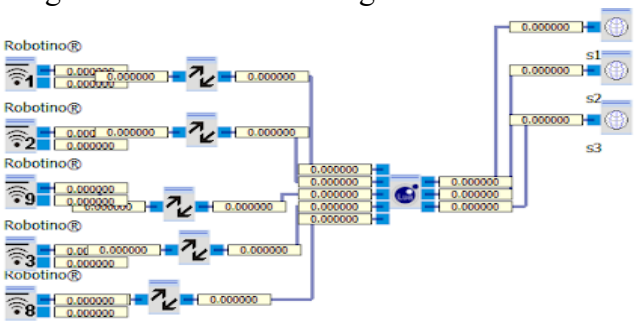
Figura 39: Recorte de código utilizado.



Fonte: Autoras.

Então no Step4 que tem subprograma denominado “CHEFE” (Figura40) que analisa os sensores e faz a decisão para qual das 3 seguintes opções o programa deve seguir, caso seja detectado que nenhum dos sensores infravermelhos estão próximos a obstáculos então terá sua saída S1 ativa, o que levará o próximo Step.

Figura 40: Recorte de código utilizado.

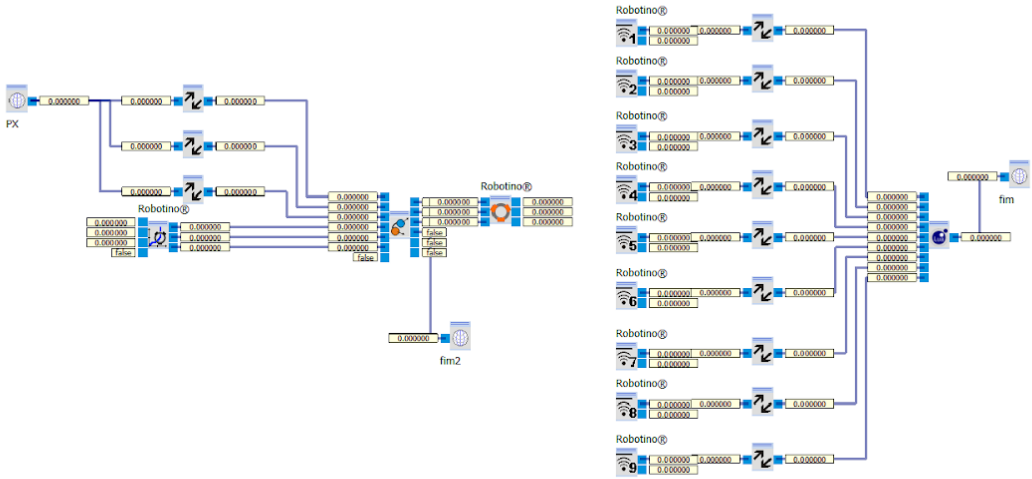


Fonte: Autoras.

No step P000 com subprograma denominado ” LIVRE” (Figura 41, o robô se desloca diretamente até o ponto programado até que um de seus sensores detecte algum obstáculo, ou chegue até o ponto de destino, no primeiro caso o programa direciona o processo para o Step chamado “Burro”, que logo então chega ao “CHEFE” novamente para uma nova tomada de decisão. Caso o obstáculo se localize a Esquerda, a saída será S2 levando o processo para o Step P0001 com subprograma denominado Direita, e o robô desviará para direita até que seus sensores frontais estiverem totalmente livres, então, no step seguinte, o robô se direciona para frente cerca de 40 cm no subprograma “Sai da Barreira”, para evitar que o deslocamento se inicie com pontos cegos entre os sensores frontais.

Assim que completar o ciclo de desvio o programa será redirecionado para se tomar uma decisão novamente. Caso o obstáculo se localize a direita a saída será S3 e haverá uma forma de desvio muito semelhante à descrita anteriormente para o lado inverso.

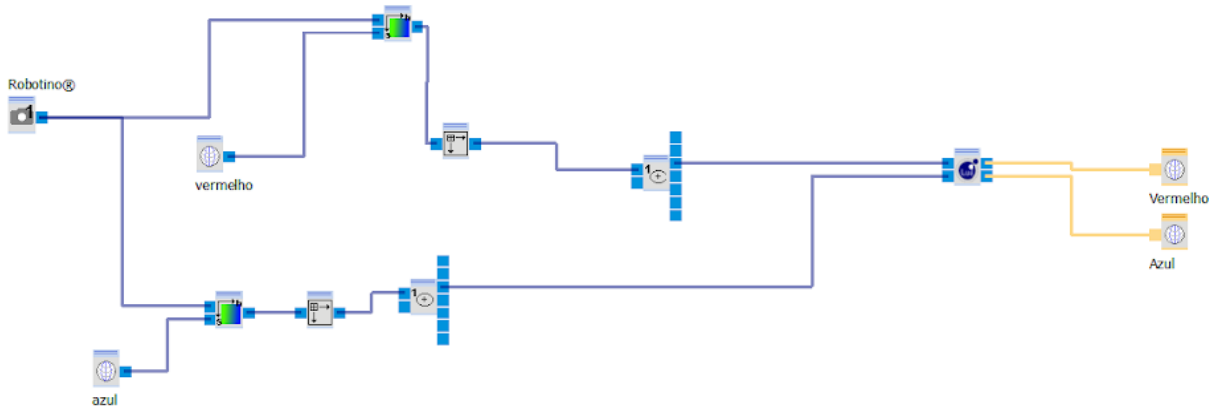
Figura 41: Recorte de código utilizado.



Fonte: Autoras.

Há o deslocamento em direção a posição 0 que será localizada na mesa de recolhimento de peças. é necessária a distinção entre as peças antes do recolhimento das mesmas que é feita pelo subprograma identifica_cor ilustrado abaixo na Figura 42.

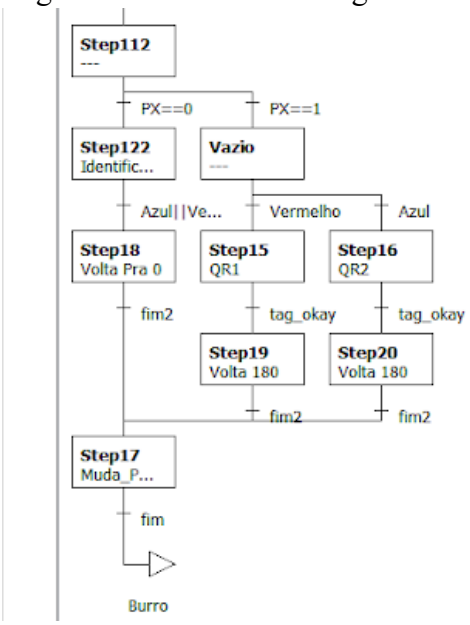
Figura 42: Recorte de código utilizado.



Fonte: Autoras.

Após essa distinção, são enviadas saídas digitais para o Arduíno que processa-as e estabelece as posições dos servos motores para o recolhimento da peça, em seguida no Step 17 há a mudança de ponto de localização, como a localização atual se encontra no ponto 0, está comutará para 1 e então na etapa de depósito o robô se locomove para a posição 1. Caso a cor encontrada for a cor Vermelho o robô faz a distinção das mesas de depósito através dos QR-Codes e a depositará naquela que tiver o QR-Code 1 Caso a cor identificada for azul depositará naquela que tiver o QR-Code 2. Como é ilustrado no GRAFSET na Figura 43. Novamente o Step 17 comutará o valor da posição, agora que está 1 irá para 0 fazendo assim um reinício do ciclo de recolhimento e entrega.

Figura 43: Recorte de código utilizado.



Fonte: Autoras.

5.2 PARTE 2 : MANIPULADOR

Com os resultados obtidos foi possível a visualização da garra operando de maneira esperada, atendendo a todos os comandos enviados pelo código. Ela funciona da seguinte maneira, ao usar o código de gravação, serão armazenados e guardados todos os ângulos desejados em que ele deve operar, em seguida com o auxílio do robotino que enviará um número em binário, ou seja, digital onde será convertido em decimal, pois esse número corresponderá a uma posição que foi escolhido para ser executada no momento.

Podendo cada motor movimentar-se em um ângulo de até 180º graus, para obter os 3 eixos de movimentação foram necessários a utilização de 3 servos. Para melhor compreensão em seguida será explicado os códigos desenvolvidos para a composição do braço robótico .

De início foi necessário declarar algumas bibliotecas que são <EEPROM.h> e <SERVO.h>, variáveis para nomear os motores: servo.motor1, servo.motor2 e servo.motor3; e botões: p1, p2 e p3; e variáveis de controle b1, b2, b3; int leitura = 0 e o vetor int angulo[15].

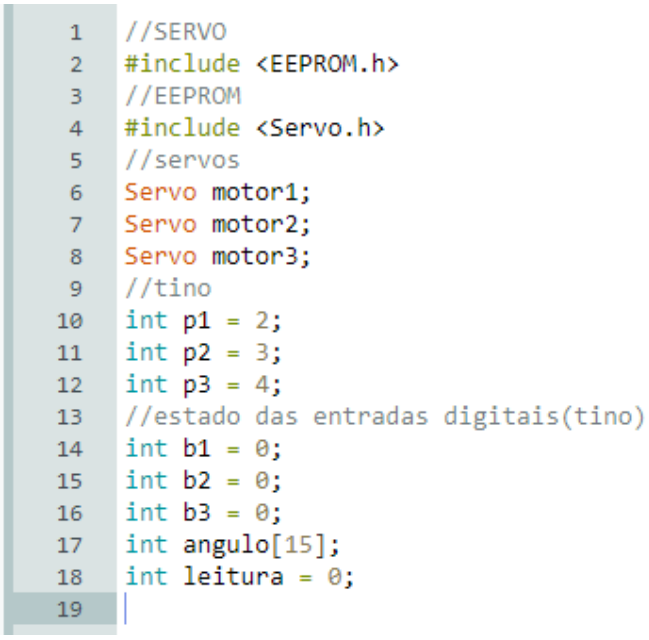
A biblioteca <EEPROM.h> é responsável por habilitar o uso da memória EEPROM no arduino, estando presente no código exemplo. Já a biblioteca <SERVO.h> facilita a

utilização do servo, visto que geralmente em códigos que exija posição, ângulos e outras funções são necessários um cálculo para chegar a determinada estado, utilizando essa biblioteca o programa nomeia os motores e ângulos sem que precise de cálculos.

As variáveis de controle são variáveis escolhidas pelo programador para nomear funções em que precise de outros comandos para serem compiladas e facilitar o seu chamado durante o decorrer do codigo. Primeiramente foi declarado os motores chamando-os de Servo motor1, Servo motor2 e Servo motor3, deste modo podemos referenciá-lo durante o codigo. Dando sequência, com as variáveis p1, p2 e p3 que são os botões, foi necessário adicionar um valor a cada uma delas (sendo eles 2, 3 e 4) que corresponde a os pinos que estão ligados no robotino, ou seja, deste modo mantenho o robô conectado ao arduino, fazendo leitura da saída digital do robô. Responsável por salvar o estado da entrada digital dos pinos p1 até p3 foram declaradas b1, b2 e b3 como responsável por essa ação.

Utilizamos a variável leitura para salvar o valor numérico que está sendo representado em binário pelas portas digitais do robotino e o vetor ângulo que vai salvar os valores gravados na EEPROM (ver Figura 44) .

Figura 44: Recorte inicial do código completo.



```
1  //SERVO
2  #include <EEPROM.h>
3  //EEPROM
4  #include <Servo.h>
5  //servos
6  Servo motor1;
7  Servo motor2;
8  Servo motor3;
9  //tino
10 int p1 = 2;
11 int p2 = 3;
12 int p3 = 4;
13 //estado das entradas digitais(tino)
14 int b1 = 0;
15 int b2 = 0;
16 int b3 = 0;
17 int angulo[15];
18 int leitura = 0;
19
```

Fonte: Autoras.

Como configuração inicial foi declarado o void setup que é a parte de configuração do arduino, onde ele roda o programa e volta para o início, configurando os pinos p1, p2 e p3 com entradas 2 3 4. Como uma característica própria da biblioteca, são declaradas as variáveis motor1.attach, motor2.attach e motor3.attach que configura o pino em que o motor está conectado. Em Serial.begin(9600) é habilitada a conexão serial, onde o número é a velocidade de comunicação (em bits por segundo, baund), também foi usado a memoria EEPROM.read em que o endereço de memória lido corresponde a repetição atual e mostra esse valor no monitor serial(ver Figura 45).

Figura 45: Fragmento do código do manipulador.

```
void setup() {
  pinMode(p1, INPUT);
  pinMode(p2, INPUT);
  pinMode(p3, INPUT);
  //pinos do servo
  motor1.attach(14);
  motor2.attach(47);
  motor3.attach(18);
  //inicia comunicação serial
  Serial.begin(9600);
  for (int i = 0; i <= 14; i++) {
    angulo[i] =  EEPROM.read(i);
    Serial.println(EEPROM.read(i));
    delay(15);
  }

}
```

Fonte: Autoras

Em seguida o código se repete através do Void Loop (ver Figura 46), salva o pino utilizando Digitalread, pois isso faz-se uma conversão de binário em digital, salvando o valor da conta na variável Leitura e mostrando o valor na tela (mostra na tela Leitura, Serial.println e imprime o valor da variável Leitura).

Figura 46: Fragmento do código completo.

```
void loop() {
ponto:
  b1 = digitalRead(p1);
  b2 = digitalRead(p2);
  b3 = digitalRead(p3);
  leitura = b1 * 1 + b2 * 2 + b3 * 4;

  Serial.println("leitura");
  Serial.println(leitura);

```

Fonte: Autoras.

Se o valor da leitura for igual a zero entra na primeira posição, movendo os motores utilizando os valores salvos nas primeiras 3 posições do vetor ângulo(0, 1 e 2) e coloca cada um desses valores de forma a representar o ângulo no comando referente ao motor específico (motor1.write(angulo[0])) agindo de forma semelhante para os outros valores possíveis de

leitura (0 a 4) mudando somente a posição do vetor que está sendo usado como ângulo (Figura 47).

Figura 47: Recorte final do código do completo.

```
1
2   if (leitura == 0)
3   {
4     motor1.write(angulo[0]);
5     motor2.write(angulo[1]);
6     motor3.write(angulo[2]);
7     goto ponto;
8   }
9   else if (leitura == 1)
10  {
11    motor1.write(angulo[3]);
12    motor2.write(angulo[4]);
13    motor3.write(angulo[5]);
14    goto ponto;
15  }
16  else if (leitura == 2)
17  {
18    motor1.write(angulo[6]);
19    motor2.write(angulo[7]);
20    motor3.write(angulo[8]);
21    goto ponto;
22  }
23  else if (leitura == 3)
24  {
25    motor1.write(angulo[9]);
26    motor2.write(angulo[10]);
27    motor3.write(angulo[11]);
28    goto ponto;
29  }
30  if (leitura == 4)
31  {
32    motor1.write(angulo[12]);
33    motor2.write(angulo[13]);
34    motor3.write(angulo[14]);
35    goto ponto;
36  }
```

Fonte:Autoras

Em seguida para gravação da sequência de ângulos utilizados, foi aplicado o uso de um código , em que são declarados os ângulos dentro de um vetor que em seguida são gravados na memória .

O código abaixo (Figura 48) funciona da seguinte maneira, primeiramente é declarada a biblioteca <EEPROM.h> que como dita anteriormente é usada para habilitar a memória EEPROM, em seguida declaramos o vetor em que serão salvos os ângulos desejados, onde em

cada 3 em 3 são salvas posições diferentes,sendo cada posição um conjunto de 3 ângulos, 1 para cada motor da garra.

Em seguida inicia-se a comunicação serial no void setup, com o valor de 9600 como a velocidade de comunicação. Após isso no void loop abre-se um laço de repetição, gravando um número específico, cujo o valor é o mesmo que esta dentro do vetor.

A posição do vetor que está sendo gravada será correspondente ao número da repetição atual, e a memória que está sendo gravada também é o número da repetição atual, pois é um laço onde será feito o mesmo processo para todas as números desejados. Após terminar as 15 repetições, o programa lê os valores salvos e mostra na tela, repetido-se 15 vezes, em seguida usa-se um while para parar o código.

Figura 48: Código de gravação.

```
1
2
3   #include <EEPROM.h>
4     // Angulos/valores a serem salvos
5   int Numero [12] = {20,10,0,95,70,0,95,70,40,20,10,40,20,70,0};
6
7   void setup() {
8       Serial.begin(9600);
9   }
10
11  void loop()
12  {
13    // Escreve
14    for  (int i =; i<=11; i++){
15        Serial.print( "gravando numero na memoria");
16         Serial.println(numero [i]);
17         EEPROM.write(i, numero [i]);
18         delay(1000);
19    }
20    // Ler
21    for (int i=0; i<=11; i++){
22    Serial.print( "ler numero na memoria");
23     Serial.println(EEPROM.read([i]));
24
25  }
26   While (i)
27
28   {}
29
30  }
```

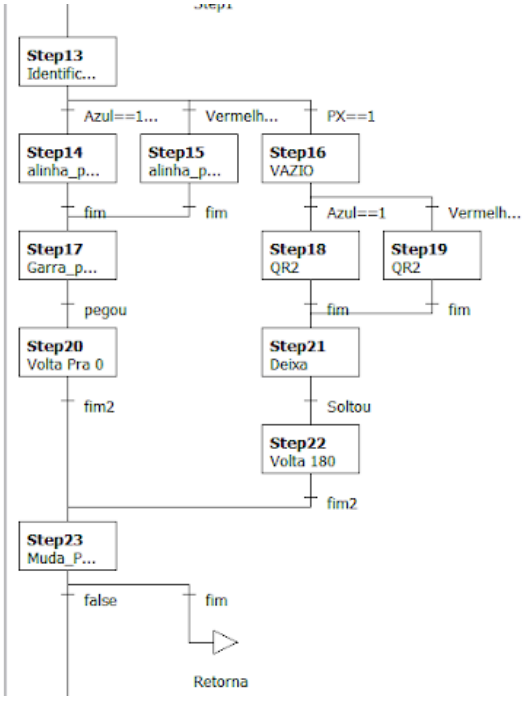
Fonte: Autoras.

5.3 MOVIMENTAÇÃO BRAÇO ROBÓTICO COM O ROBOTINO

Tendo como objetivo principal um sistema capaz de movimentar-se fazendo o reconhecimento do trajeto, e transporte de objetos através do manipulador, podermos alcançar e constatar o propósito com os seguintes resultados fazendo a junção do robotino com o braço robótico.

A cada chegada do robotino nas estações de recolhimento é iniciado um programa de reconhecimento da cor da peça, nomeado de “Identifica_cor”, após esse reconhecimento, há o alinhamento do robô com a peça e logo em seguida é acionado uma espécie de contador, que a cada 2 segundos informa uma nova posição para realizar o processo de retirada da peça sobre a mesa que leva o nome de “Garra_pega”, o robô então retorna sua angulação para 0 graus, atualiza qual será o próximo ponto e então se encaminha para ele. A partir daí o robô realiza a movimentação até a área de depósito, ao identificar o QR-Code correspondente a cor encontrada anteriormente, inicia o processo de depósito que é análogo ao de retirada, em seguida é atualizado novamente o próximo ponto de destino através do deslocamento reativo do robô. Este processo é verificado na Figura 49.

Figura 49**:** Recorte do código final.

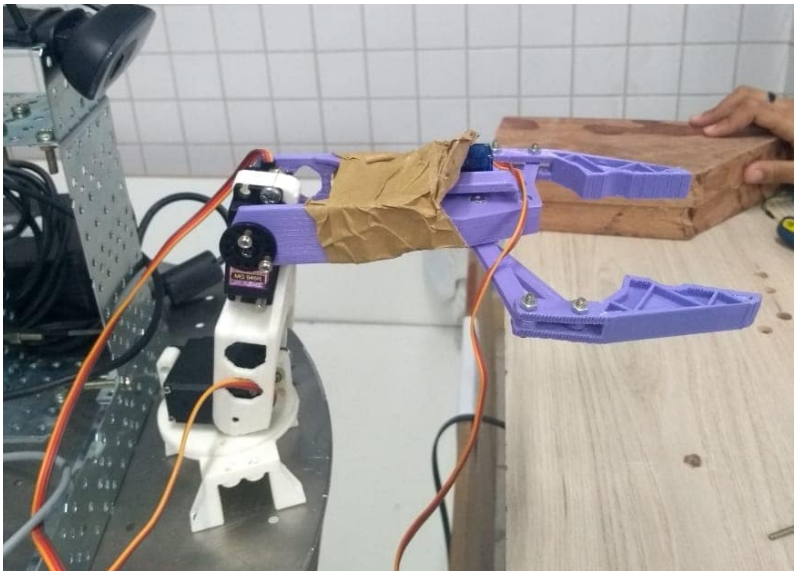


Fonte: Autoras.

Mantendo-se grudada na superfície do robotino foi definida como posição inicial (chamada de posição 0 no código) do braço robótico como os servos nos seguintes ângulos:

Servo1(servo inferior) estará em um ângulo de 20ºgraus (para cima), de modo que o servo esteja em pé, o servo2 (servo do meio) estará em um ângulo de 10ºgraus posicionado de forma deitada para que a garra possa pegar o objeto desejado, já a garra posto no servo3 permanece parada e aberta com um ângulo de 0ºgraus (ver figura 50).

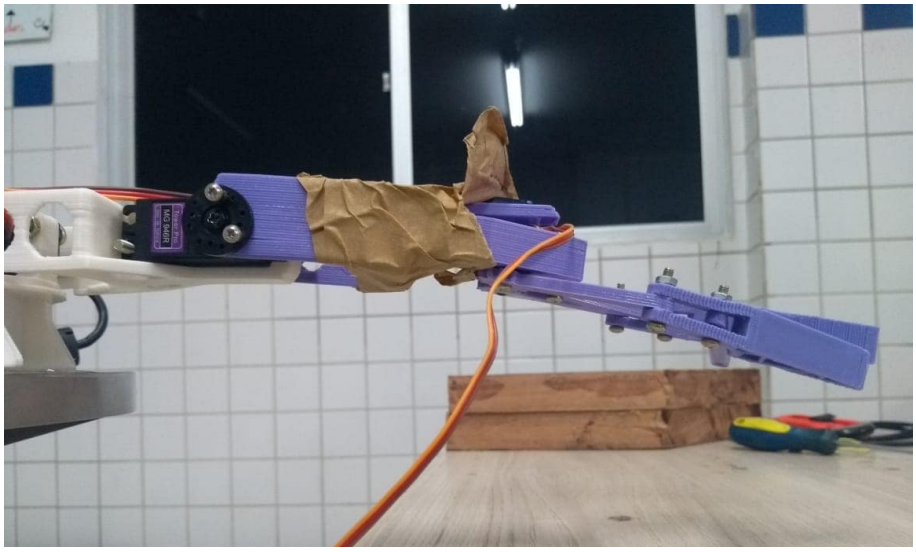
Figura 50: Manipulador executando a posição 0.



Fonte: Autoras.

Em seguida o robotino enviará o comando onde o braço irá se aproximar da bancada onde estará a peça a ser pega, executando a posição 1 onde o servo1 passará para um ângulo de 95ºgraus de forma que ele fique deitado, já o servo2 estará em um ângulo de 70ºgraus posto em pé, o servo3 permanecerá no ângulo de 0ºgraus mantendo-se a garra aberta, com isso o braço estará de forma totalmente deitado(ver Figura 51).

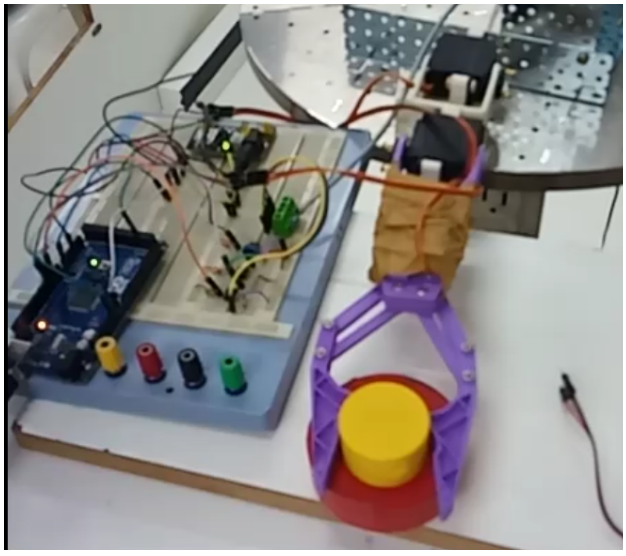
Figura 51: Execução da posição 1.



Fonte: Autoras.

Na posição 2 ambos os servos 1 e 2 permanecem na mesma posição que anteriormente, não ocorrendo variação, apenas na posição da garra que passará para um ângulo de 40ºgraus que é o ângulo onde ela estará fechada com a peça capturada (Figura 52). Este movimento estará responsável pela captura da peça.

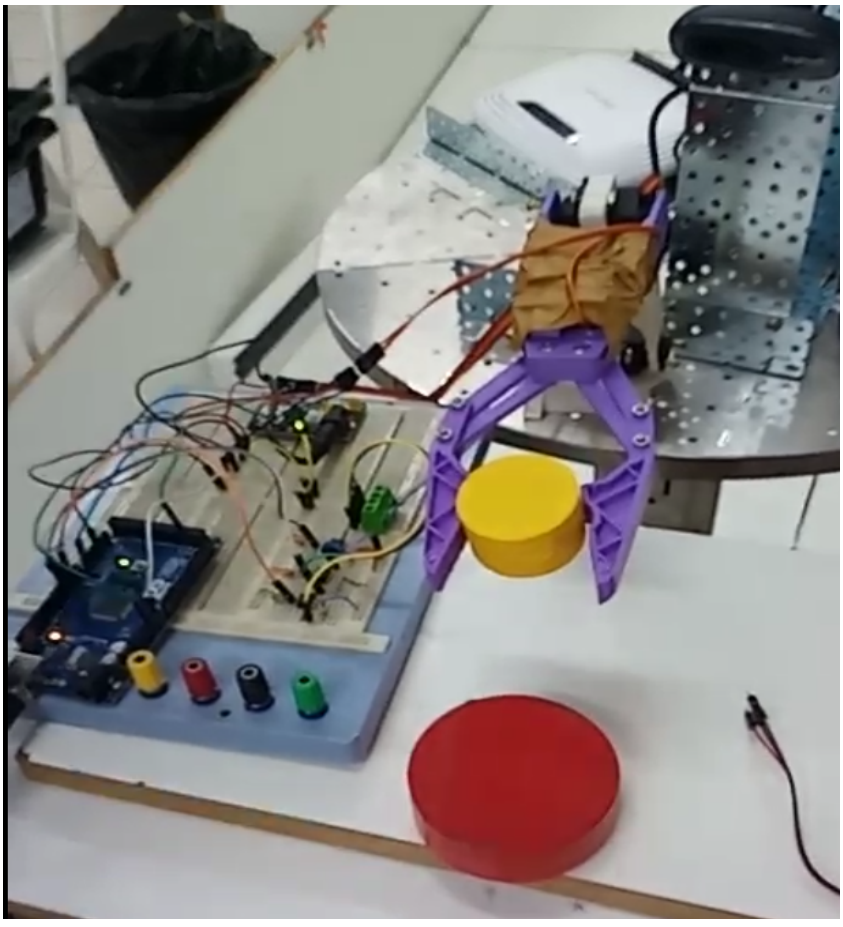
Figura 52: Aplicação da posição 3.



Fonte:Autoras

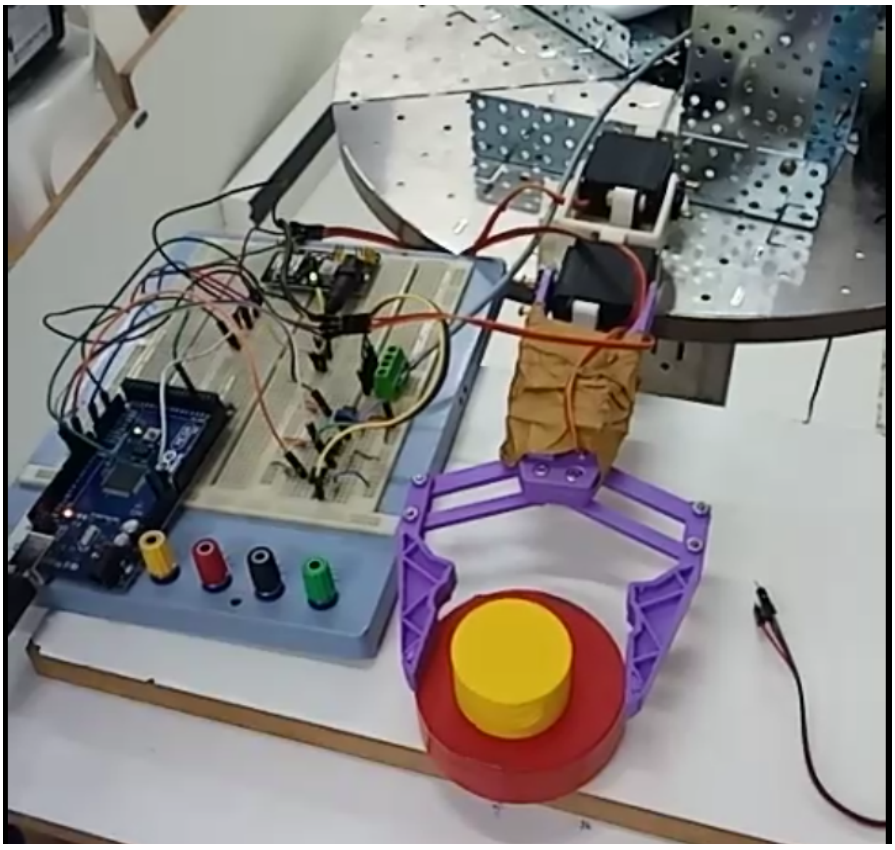
Após ter pego o objeto desejado será enviado um novo comando para a garra onde na posição 3 o braço voltará para a posição inicial (posição 0), mantendo os ângulos iniciais e (servo1 em 20ºgraus e servo2 em 10ºgraus) mantendo a peça presa na garra com um ângulo de 40ºgraus (ver Figura 53). Após a peça ser capturada é necessário a movimentação do robotino para chegar ao lugar de entrega, após isso o robotino enviará um comando onde o braço executará a posição 2 novamente, no qual agarra se aproximará da bancada e fará o movimento de abertura deixando a peça no local desejado (Figura 54).

Figura 53: Realização da posição 4.



Fonte: Autoras

Figura 54: Manipulador executando a posição 2 novamente.



Fonte: Autoras

Por último, executando a última posição (posição 4) do manipulador os servos serão postos nos ângulos de 20ºgraus para o servo1, 70ºgraus para o servo2 e 0ºgraus para o servo3, mantendo o braço em uma posição totalmente em pé, com a garra aberta, estando pronto para o próximo comando do robotino (ver Figura 55).

Figura 55: Execução da última posição gravada

Fonte: Autoras

6 CONCLUSÃO

Ao fim deste Trabalho de Conclusão de Curso, que teve como ideia principal a implementação de um sistema autônomo de logística industrial, obtivemos bons resultados. Além disso, durante as pesquisas nós tivemos a oportunidade de entender um pouco mais sobre o universo da robótica móvel, que vem crescendo e se tornando cada dia mais presente nas indústrias.

Para o grupo, desenvolver lógicas de movimentação para um robô de alto nível como é o caso do Robotino 3 foi algo muito gratificante, tendo em vista que isso provavelmente não seria possível, se não tivéssemos o apoio do Campus IFRN-ZN. As experiências adquiridas no desenvolver do presente trabalho são inúmeras.

Esperamos que as pesquisas nessa área se tornem cada vez mais frequentes. Em alguns anos, nós veremos uma indústria bem mais eficiente e automatizada, com robôs autônomos altamente desenvolvidos. Como metas futuras temos a intenção de realizar as movimentações e controle do manipulador utilizando ROS para um melhor desempenho.

7 REFERÊNCIAS

About ROS. Disponivel em: <http://www.ros.org/about-ros/>. Acesso em: 02 dez. 2018

Angerer, S., Strassmair, C., Staehr, M., Roettenbacher, M., Robertson, N.M.: Give me a hand – The potential of mobile assistive robots in automotive logistics and assembly applications. In: IEEE International Conference on Technologies for Practical Robot Applications (TePRA). (2012)

CARRARA, V. Apostila de Robótica. Universidade Braz Cubas, Área de Ciências Exatas Engenharia Mecânica, Engenharia de Controle e Automação, 2009. p. 13-27.

DREHER, A. The Smart Factory of the Future – Part 1. Belden News. Available:http://www.belden.com/blog/industrialethernet/The-Smart-Factory-of-theFuture-Part1.cfm. Access: 2 jun.2016.

Festo Didactic, "Robotino® Mobile robot platform for research and training," Denkendorf, 56940, 2013.

Hoje na História: 1926 - Metropolis, de Fritz Lang, estreia em Berlim. Opera Mundi. Dísponível em: <https://operamundi.uol.com.br/historia/26506/hoje-na-historia-1926-metropolis-de-fritz-lang-estreia-em-berlim> Acesso em: 2 ago. 2018.

HUNT, I. Lightning in His Hand: The Life Story of Nikola Tesla. 1.ed. Peak District: Pikes Peak Library District, 2010.

Lucke, D., Constantinescu, C., Westk ̈amper, E.: Smart Factory – A Step towards the Next Generation of Manufacturing. In: Manufacturing Systems and Technologies for the New Frontier, 41st CIRP Conf. on Manufacturing Systems. (2008)

Kagermann, H., Wahlster, W., Helbig, J.: Recommendations for implementing the strategic initiative INDUSTRIE 4.0. Final Report, Platform Industrie 4.0 (2013)

KOYRÉ, A. Os Filósofos e a Máquina. In KOYRÉ, A. Estudos de História do Pensamento Filosófico. Rio de Janeiro: Forense Universitária, 1991.

Latin American and Brazilian Robotics Competition 2018. Cbrobotica. Dísponível em : <http://www.cbrobotica.org/> Acesso em: 29 ago. 2018>.

M. Zarte, A. Pechmann, J. Wermann, F. Gosewehr and A. W. Colombo, "Building an Industry 4.0-compliant lab environment to demonstrate connectivity between shop floor and IT levels of an enterprise," *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, Florence, 2016, pp. 6590-6595.

SIEGWART, Roland; NOURBAKHSH, Illah R. Introduction to Autonomous Mobile Robots. Cambridge: MIT Press, 2004. 321 p.

Simplício, P. V. G. e Lima, B. R., "Manipuladores robóticos industriais," Caderno de Graduação-Ciências Exatas e Tecnológicas-UNIT, 2016.

Pandey & Gelin, A Mass-Produced Sociable Humanoid Robot, IEEE ROBOTICS & AUTOMATION MAGAZINE: Amit Kumar Pandey e Rodolphe Gelin, Set 2018

Romano, V. F.; Dutra, M. S., **Introdução à Robótica Industrial.** In: Vitor Romano. (Org.). Robótica Industrial: Aplicação na Indústria de Manufatura e de Processos, 1 ed. São Paulo: Edgard Blücher, pp. 1-19, 2002.

Vargas, I. G., Tavares, D. M. "Estudo de Caso Usando o Framework Robot Operating System (ROS)". X Encontro Anual de Computação – EnAComp, 2013.

Pimenta, T. Tavares. Controle de Manipuladores Robóticos.PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO DEPARTAMENTO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO-2009

Gomes, Sinésio. Controle e automação industrial III. Ministradas nos Cursos Tecnicos em Eletronica-2014

Motta, Allan. Vida de silicio. o que e arduino e como funciona. 2 de maio de 2017 <https://portal.vidadesilicio.com.br/o-que-e-arduino-e-como-funciona/>

O que é memória EEPROM?, Dísponível em : http://ptcomputador.com/Ferragens/ram-cards-motherboards/59305.html> Acesso em: 28 nov. 2018>.

8 ANEXO I - CÓDIGO DO MANIPULADOR COMPLETO

```
//SERVO
#include <EEPROM.h>
//EEPROM
#include <Servo.h>
//servos
Servo motor1;
Servo motor2;
Servo motor3;
//tino
int p1 = 2;
int p2 = 3;
int p3 = 4;
//estado das entradas digitais(tino)
int b1 = 0;
int b2 = 0;
int b3 = 0;
int angulo[15];
int leitura = 0;
void setup() {
  // put your setup code here, to run once:
  pinMode(p1, INPUT);
  pinMode(p2, INPUT);
  pinMode(p3, INPUT);
  //pinos do servo
  motor1.attach(14);
  motor2.attach(47);
  motor3.attach(18);
  //inicia comunicação serial
  Serial.begin(9600);
```

```
  for (int i = 0; i <= 14; i++) {
    angulo[i] =  EEPROM.read(i);
    Serial.println(EEPROM.read(i));
    delay(15);
  }

}

void loop() {
ponto:
  b1 = digitalRead(p1);
  b2 = digitalRead(p2);
  b3 = digitalRead(p3);
  leitura = b1 * 1 + b2 * 2 + b3 * 4;

  Serial.println("leitura");
  Serial.println(leitura);

  if (leitura == 0)
  {
    motor1.write(angulo[0]);
    motor2.write(angulo[1]);
    motor3.write(angulo[2]);
    goto ponto;
  }

  else if (leitura == 1)
  {
    motor1.write(angulo[3]);
    motor2.write(angulo[4]);
```

```
  motor3.write(angulo[5]);
  goto ponto;
 }
 else if (leitura == 2)
 {
  motor1.write(angulo[6]);
  motor2.write(angulo[7]);
  motor3.write(angulo[8]);
  goto ponto;
 }
 else if (leitura == 3)
 {
  motor1.write(angulo[9]);
  motor2.write(angulo[10]);
  motor3.write(angulo[11]);
  goto ponto;
 }

 if (leitura == 4)
 {
  motor1.write(angulo[12]);
  motor2.write(angulo[13]);
  motor3.write(angulo[14]);
  goto ponto;
 }}
```

9 ANEXO II - PROGRAMAÇÃO DO ROBOTINO

Linhas de código do bloco Lua - Chefe

```
if in1>15
and in2>15
```

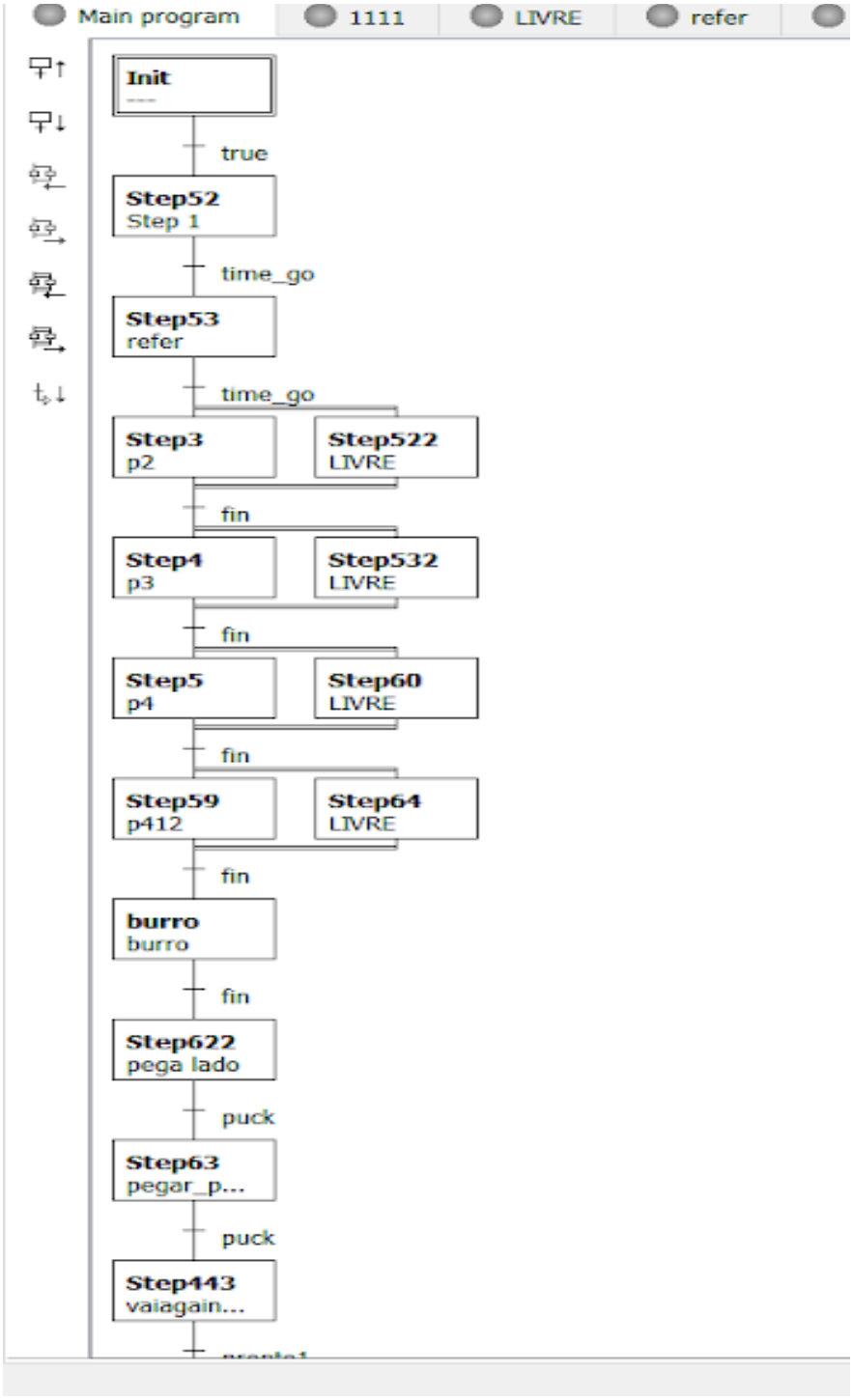
```
and in3>15
and in4>15
and in5>15 then

out1=1
out2=0
out3=0

end

if in1<15
and in2<15
and in3<15
and in4>15
and in5>15 then

out1=0
out2=1
out3=0

end

if in1<15
and in2<15
and in3<15
and in4<15
and in5>15 then

out1=0
out2=1
out3=0

end

if in1<15
and in2>15
and in3>15
and in4>15
and in5>15 then
```

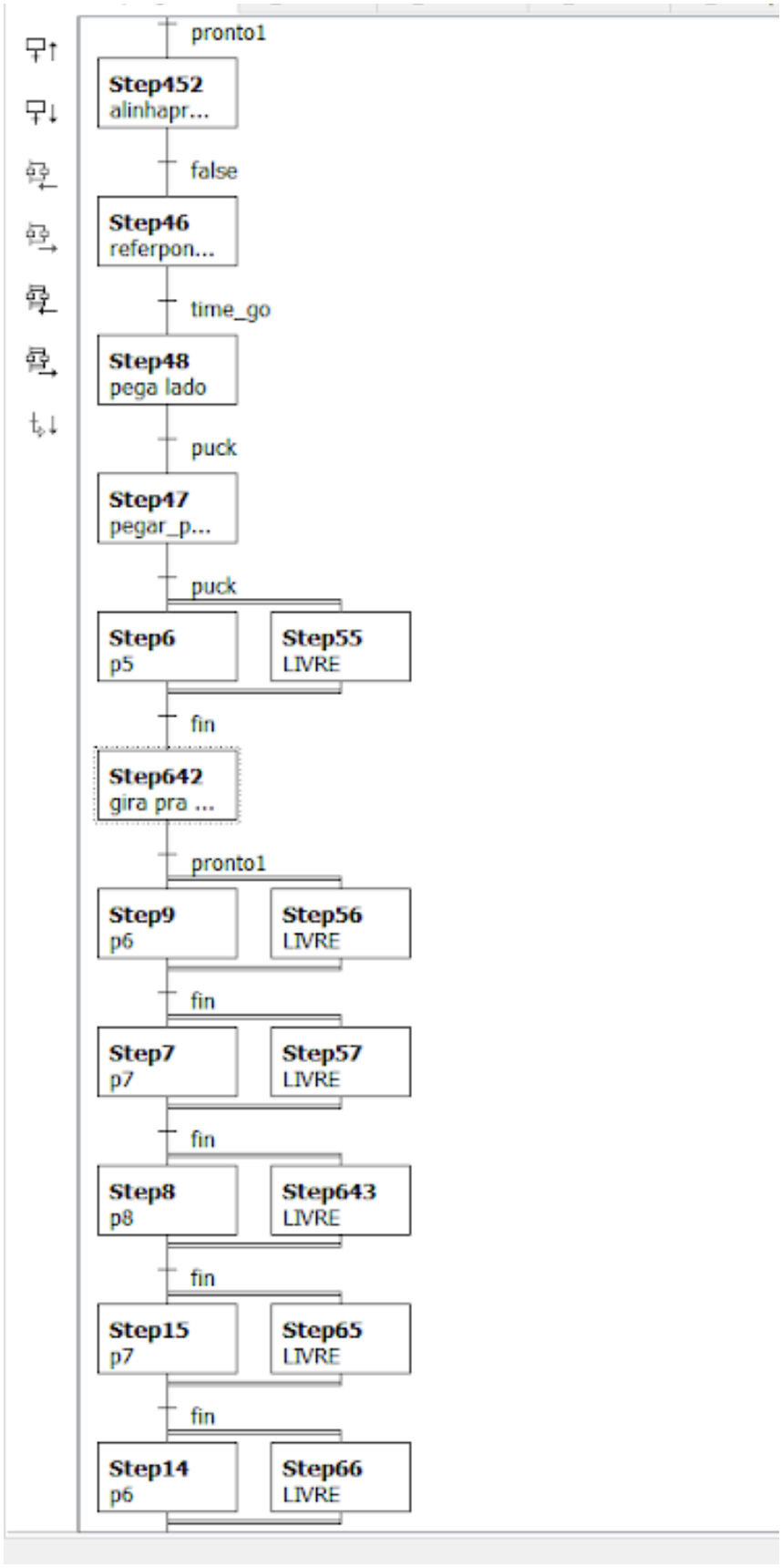
```
out1=0
out2=1
out3=0

end

if in1<15
and in2<15
and in3<15
and in4>15
and in5<15 then

out1=0
out2=0
out3=1

end

if in1<15
and in2<15
and in3>15
and in4>15
and in5>15 then

out1=0
out2=1
out3=0

end

if in1>15
and in2<15
and in3>15
and in4>15
and in5>15 then

out1=0
out2=1
```

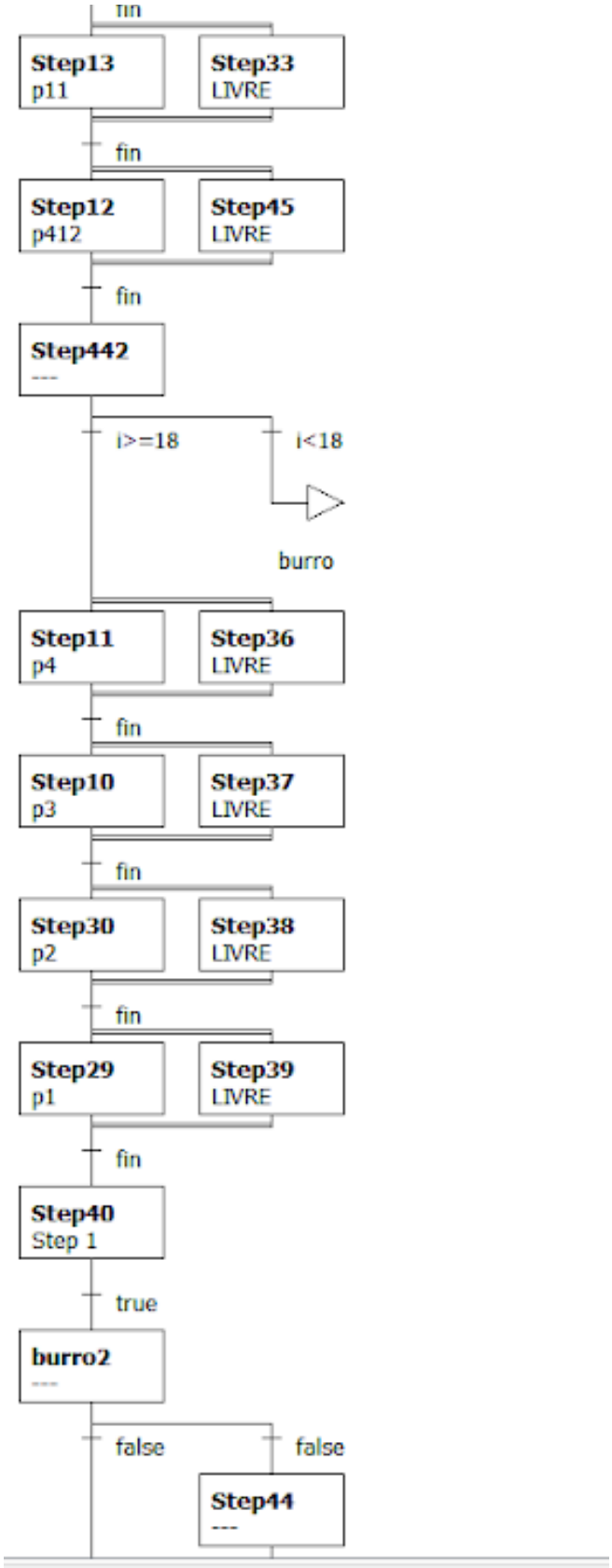
```
out3=0

end

if in1<15
and in2>15
and in3<15
and in4>15
and in5>15 then

out1=0
out2=0
out3=1

end

if in1>15
and in2>15
and in3<15
and in4>15
and in5>15 then

out1=0
out2=0
out3=1

end

if in1<15
and in2>15
and in3<15
and in4>15
and in5<15 then

out1=0
out2=0
out3=1

end
```

```
if in1<15
and in2>15
and in3>15
and in4>15
and in5<15 then

out1=0
out2=0
out3=1

end

if in1<15
and in2>15
and in3>15
and in4<15
and in5>15 then

out1=0
out2=1
out3=0

end
```

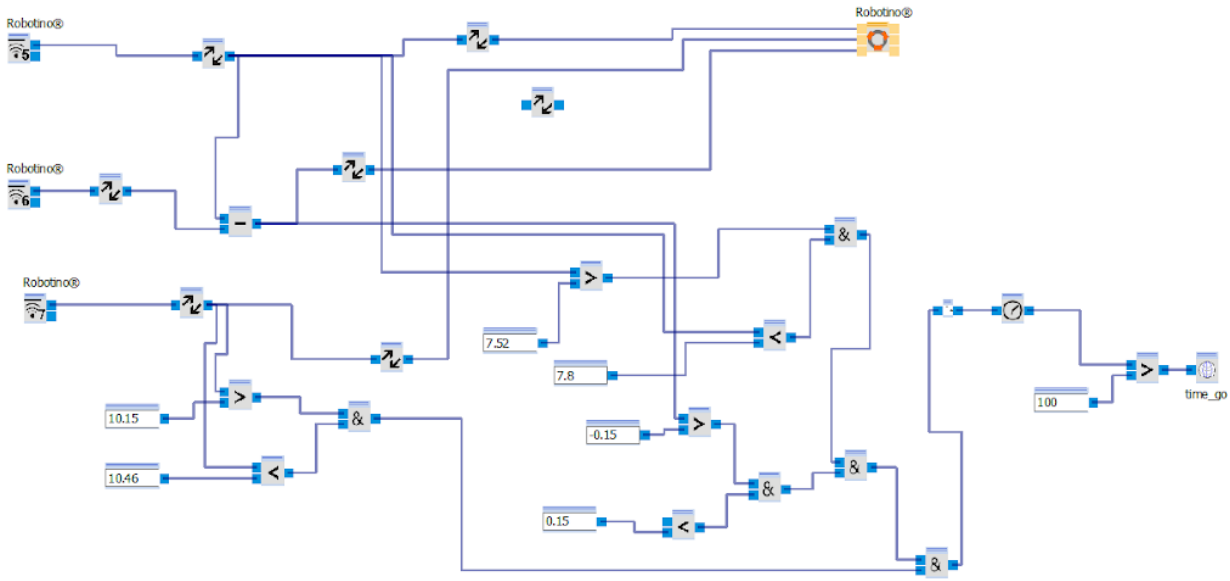
10 ANEXO III - PRIMEIRA PROVA - LÓGICA DE DESLOCAMENTO



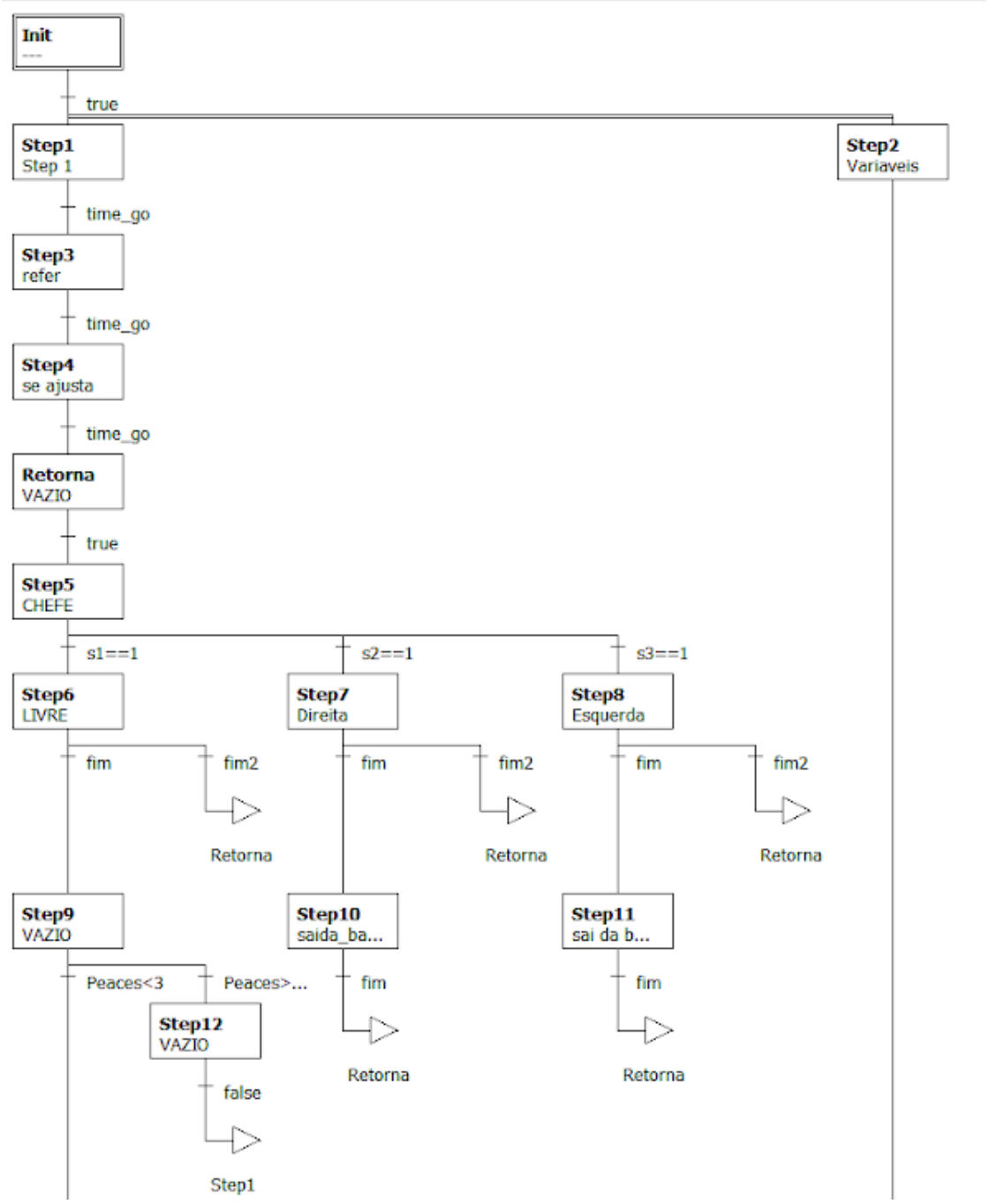

pronto1
Step452
alinhapr...
false
Step46
referpon...
time_go
Step48
pega lado
puck
Step47
pegar_p...
puck
Step6
p5
Step55
LIVRE
fin
Step642
gira pra ...
pronto1
Step9
p6
Step56
LIVRE
fin
Step7
p7
Step57
LIVRE
fin
Step8
p8
Step643
LIVRE
fin
Step15
p7
Step65
LIVRE
fin
Step14
p6
Step66
LIVRE

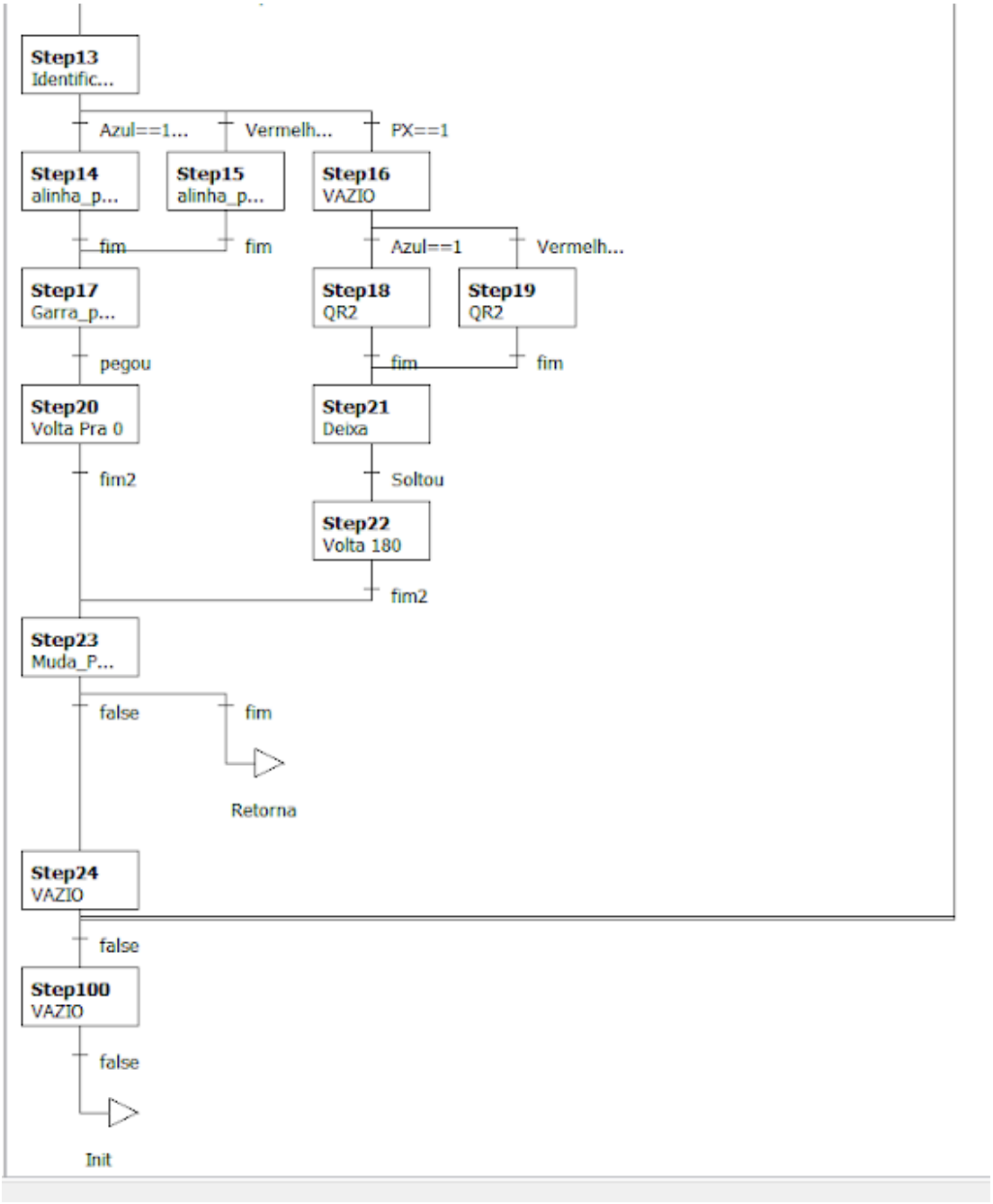

fin
Step13
p11
Step33
LIVRE
fin
Step12
p412
Step45
LIVRE
fin
Step442
---
i>=18
i<18
burro
Step11
p4
Step36
LIVRE
fin
Step10
p3
Step37
LIVRE
fin
Step30
p2
Step38
LIVRE
fin
Step29
p1
Step39
LIVRE
fin
Step40
Step 1
true
burro2
---
false
false
Step44

Alinhamento com a parede traseira.



Lógica de Deslocamento Reativa.




Step13
Identific...
Azul==1...
Vermelh...
PX==1
Step14
alinha_p...
Step15
alinha_p...
Step16
VAZIO
fim
fim
Azul==1
Vermelh...
Step17
Garra_p...
Step18
QR2
Step19
QR2
pegou
fim
fim
Step20
Volta Pra 0
Step21
Deixa
fim2
Soltou
Step22
Volta 180
fim2
Step23
Muda_P...
false
fim
Retorna
Step24
VAZIO
false
Step100
VAZIO
false
Init

11 ANEXO III - GRAVAÇÃO NA EEPROM.WRITE

12 ANEXO III - LEITURA NA EEPROM.READ