Claudinei Stiegelmaier

BRAÇO ROBÓTICO MÓVEL COM RECONHECIMENTO DE POSIÇÃO E RECOLHIMENTO DE OBJETOS

Horizontina - RS

2019

Claudinei Stiegelmaier

BRAÇO ROBÓTICO MÓVEL COM RECONHECIMENTO DE POSIÇÃO E RECOLHIMENTO DE OBJETOS

Trabalho Final de Curso apresentado como requisito parcial para a obtenção do título de Bacharel em Engenharia Controle e Automação na Faculdade Horizontina, sob a orientação do Prof. Dr. Geovane Webler.

Horizontina - RS

2019

FAHOR - FACULDADE HORIZONTINA

CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO

A Comissão Examinadora, abaixo assinada, aprova o trabalho final de curso

“Braço Robótico Móvel Com Reconhecimento De Posição E Recolhimento De Objetos”

Elaborada por:

Claudinei Stiegelmaier

Como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Controle e Automação

Aprovado em: 26/06/2019

Pela Comissão Examinadora

_____

Dr. Geovane Webler

Presidente da Comissão Examinadora - Orientador

_____

Me. Cristiano Rosa dos Santos

FAHOR – Faculdade Horizontina

_____

Me. Luis Carlos Wachholz

FAHOR – Faculdade Horizontina

Horizontina - RS

2019

À minha família, por sua capacidade de acreditar e investir em mim. Rose, seu cuidado e dedicação me deram em muitos momentos a esperança para prosseguir. Gabi, teu silêncio em minhas tantas horas de estudos significou segurança e certeza de que não estou sozinho nessa caminhada. Lucca, sua conquista acadêmica me incentivou ainda mais a não desistir. Bia, pelo incentivo em continuar.

AGRADECIMENTO

À minha mãe, ao meu pai (in memoriam), aos meus irmãos, cunhadas e cunhados, comadres e compadres, ao meu orientador Geovane Webler, quando tudo parecia estar de cabeça para baixo conseguiu me dar a direção necessária, a todos, que de uma forma ou de outra, aceitaram minha ausência e fizeram parte da minha formação, o meu muito obrigado.

"Escreva algo que valha a pena ler ou faça algo que valha a pena escrever".
(Benjamin Franklin)

RESUMO

O presente trabalho visa analisar a coleta de ovos de forma automática e contínua ao longo do período de produção de galinhas poedeiras confinadas em gaiolas. A proposta agrega valores ao coletar e organizar os ovos coletados diretamente em bandeja própria para o armazenamento de ovos. Para fins de testes, deteve-se em estabelecer deslocamentos somente para 12 posições, simulando assim uma bandeja de ovos muito comum de encontrar à venda em mercados e feiras. Para expressar os movimentos das coletas e estabelecer a precisão necessária, houve a necessidade de definir um braço robótico que conta com 3 servos motores para realizar os movimentos das juntas, sendo: 1 para movimento lateral (esquerda e direita), 1 para movimento de elevação e outro para aumentar e diminuir a extensão da posição da garra. Há também um motor de passo para abrir e fechar a garra. O movimento deste está atrelado ao pegar o ovo na posição reconhecida pelo sensor e soltar o ovo na caixa em posição pré-estabelecida. Possui também dois motores AC com ligação em série para se movimentar em deslocamento lateral do equipamento completo. Um sensor ultrassônico com a finalidade de identificar a posição do objeto e calcular a distância até o ovo que é o objeto em questão. Há também dois sensores ultrassônicos posicionados nas partes laterais do carro de deslocamento, com a função de reconhecer o fim de curso do movimento lateral. Todos os comandos de movimentação são pré-programados em controlador do tipo Arduino Mega. As fontes de alimentação necessárias são de 5 volts para os motores e sensores e 12 volts para o controlador. Para todos os testes em bancada, com protótipo feito em MDF de 3mm, a tensão necessária está sendo fornecida por uma fonte de computador com entrada de energia bivolt (110/220v) contando com conversão automática e saídas nas tensões necessárias para o projeto em questão. É possível considerar obtenção de grande êxito no projeto como um todo. Realizou-se a montagem do braço robótico em cima de um carro móvel. Sendo este capaz de realizar os movimentos de acordo com o programado e estipulado. A correta construção das partes mecânicas do braço robótico descreve o funcionamento dentro do esperado dos movimentos realizados com os componentes integrados ao projeto.

Palavras-chave: Braço robótico. Coleta de ovos. Galinhas poedeiras.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E/OU SIGLAS

CA - Corrente Alternada

CC ou DC - Corrente Contínua

CPU - Unidade Central de Processamento

EDVAC - *Electronic Discrete Variable Automatic Computer*

ENIAC - *Electronic Numerical Integrator and Computer*

IBGE - Instituto Brasileiro de Geografia e Estatística

ICSP - *In Circuit Serial Programming* (Programação em Circuito Serial)

IDE - *Integrated Development Environment*

ISO - *International Organization for Standardization* (Organização Internacional de Normalização)

LED - *Light Emitting Diode* (Diodos Emissores de Luz)

MDF - *Medium Density Fiberboard* (Placa de Fibra de Média Densidade)

POG - Produção de Ovos de Galinha

PWM - *Pulse-Width Modulation* (Modulação por Largura de Pulso)

TFC - Trabalho Final de Curso

UDESC - Universidade do Estado de Santa Catarina

USB - *Universal Serial Bus* (Porta Universal)

SUMÁRIO

1 INTRODUÇÃO

O setor avícola brasileiro é um dos principais componentes da economia brasileira. Ele se destaca no *ranking* da exportação da carne de frango e, também, como um dos maiores produtores deste tipo de carne. Através destas informações, verifica-se que a atividade da avicultura brasileira ganha destaque sendo uma das mais importantes produtoras de proteína animal considerando a produtividade e, também, a qualidade da carne e do ovo produzido.

Pesquisadores da UDESC, afirmam em publicação feita no Jornal Sul Brasil:

> "O alimento de origem animal mais completo em termos nutricionais é o ovo, estrutura reprodutiva das aves que quando não fecundada pode ser comercializada para fins de alimentação humana. Seu valor nutricional é inferior apenas ao do leite materno e é acessível a todas as classes sociais. Os ovos comerciais são produzidos em granjas que utilizam poedeiras comerciais leves (brancas) ou semipesadas (marrons ou pretas), que produzem ovos de cascas brancas ou marrons, respectivamente. As aves são criadas em gaiolas ou em piso, sem contato com machos. As coletas podem ser realizadas de forma manual ou mecânica, através de esteiras móveis localizadas nas calhas de coleta dos ovos". (SERAFINI, 2015).

Serafini (2015) diz, também, que após a coleta os ovos são transportados para a região onde serão classificados e embalados onde muitos produtores utilizam sistema de verificação de trincas através de luz e lavagem simples com água clorada para retirar as sujeiras da casca.

Diante do que já foi visto, pode ser percebido que o ovo é um alimento nobre e acessível e, como todo alimento, também necessita de cuidados para que possa ser consumido de maneira segura. Maior parte desta segurança se dá por uma produção consciente, coleta e armazenamento com cuidados especiais que não prejudiquem sua estrutura.

No Brasil, a maior parte da produção de ovos em granjas é realizado pelo sistema de criação de galinhas poedeiras em gaiolas. Este sistema garante uma produtividade com menor custo alimentar, pois as galinhas não realizam exercícios e o gasto de energia corporal se dá em produção de ovos. De acordo com Netto (2016) em sua pesquisa relacionada a produção de ovos por galinhas em pisos ou em gaiolas, a produção em gaiolas vem sofrendo críticas de campanhas a favor do bem-estar animal. Ele afirma também que estes protestos se baseiam no espaço reduzido, alta densidade e perda de comportamentos naturais das aves. Também indica que a produção em piso é maior e com melhor qualidade, mas o custo de produção também

aumenta consideravelmente. Como as galinhas estão livres para se locomover a queima de energia não se dá somente aos ovos, mas, também, ao equilíbrio do sistema corporal sendo necessário, inclusive, um espaço para a produção muito maior e com maiores gastos em limpeza do que a produção em gaiolas. Também é necessário dizer que a coleta somente pode ser feita de forma manual por necessitar de ninhos para a postura dos ovos. A recomendação é de que seja realizada no mínimo de 7 a 10 coletas diárias pois acredita-se que, quanto maior o número de coletas, menor o tempo que o ovo permanecerá na calha evitando assim o acúmulo de ovos e minimizando o tempo que as bactérias teriam para se proliferar. Recomenda-se, também, que os ovos durante a colheita sejam acondicionados em bandejas de plástico desinfetadas, pois são laváveis e de fácil desinfecção (ARAÚJO & ALBINO, 2011).

De acordo com as informações acima, este trabalho baseia-se somente na produção de ovos com galinhas que estejam em gaiolas, onde a proposta está basicamente no recolhimento dos ovos que estejam na calha. Esta coleta será feita por braço robótico móvel, com depósito direto em caixa específica, não sendo feita análise, limpeza e separação dos ovos recolhidos.

A proposta deste trabalho é a análise da possibilidade de realizar uma coleta de forma contínua e ininterrupta. Com isso, descartando a necessidade de acompanhamento de um supervisor e evitando também a circulação de pessoal nas proximidades das gaiolas. Neste aspecto há a diminuição do contato humano com os ovos, reduzindo a transmissão de possíveis contaminantes. Araújo & Albino (2011) dizem que a contaminação inicial do ovo apresenta apenas algumas colônias de microrganismos, os quais multiplicam-se dez vezes em apenas 60 minutos que o ovo permanece na calha esperando ser recolhido.

O deslocamento lateral do carro é constante, somente com a localização de um objeto na calha que acontecerá a parada do movimento. É nesta parada de movimento que será possível o recolhimento do objeto pela garra e posterior armazenamento.

1.1 TEMA

O tema deste estudo refere-se ao desenvolvimento de um sistema recolhedor de ovos automatizado para uso em confinamento de galinhas do tipo gaiola.

1.2 DELIMITAÇÃO DO TEMA

Este trabalho delimita-se ao desenvolvimento de um braço robótico que possa ser utilizado como recolhedor de ovos automatizado para ser usado por produtores com confinamento de galinhas do tipo gaiola.

1.3 PROBLEMA DE PESQUISA

Um dos grandes problemas encontrados por pequenos produtores de ovos é o custo que acresce nas despesas fixas quando há contratação de mão de obra terceirizada. Na pequena propriedade, geralmente é o produtor e seus familiares que realizam todas as atividades de gerenciamento e serviços.

Também é necessário levar em conta o que SEBRAE (2017) divulga através de pesquisa realizada pela empresa Meta Pesquisa de Opinião. A pesquisa, realizada com 6.587 pessoas, mostrou que 46% dos empresários de Pequenos Negócios afirmam ter dificuldade de encontrar mão de obra qualificada. Por outro lado, Egestor (2017) diz que os funcionários da empresa aumentam a qualidade da produção, mas o serviço contratado de terceiros nem sempre estará à altura. Neste sentido, diz também, que é fundamental fiscalizar os serviços para garantir que ele esteja de acordo com o que foi solicitado.

Não conseguindo a contratação do serviço terceirizado, o resultado de um eventual afastamento de sua propriedade pode comprometer sua produção, incorrendo em riscos de sustentabilidade. Para isto é necessário que se consiga uma coleta correta e de eficácia comprovada, de acordo com condições estipuladas pelos próprios produtores conforme a necessidade real de sua produção, evitando ao máximo o desperdício e perdas na produção por bactérias e outros contaminantes.

De acordo com o exposto acima, a pesquisa caracteriza-se com a seguinte pergunta: é possível construir um sistema automatizado capaz de realizar uma coleta segura e confiável de ovos para aves confinadas em gaiolas?

1.4 HIPÓTESES

Tendo a pesquisa analisado a dependência que o produtor necessita dispor e observando o que a utilização de um sistema funcional para o recolhimento de ovos

pode fazer para a melhoria do sistema de coleta, parte-se das hipóteses de que a melhor forma de auxiliar um pequeno produtor no recolhimento de ovos é:

1- O desenvolvimento de um braço robótico móvel para coleta automática de ovos auxilia o produtor, de forma eficaz e confiável nas tarefas de recolhimento e armazenamento de ovos;
2- A coleta de ovos através de uma calha com esteira móvel atende as necessidades de coleta e armazenamento de ovos sem necessidade de supervisão;

1.5 JUSTIFICATIVA

Ao observar a lacuna existente na exploração acadêmica na possibilidade de uma coleta automatizada em granjas produtoras de ovos, bem como a exploração limitada no que se refere à coleta e armazenamento dos ovos, notou-se a viabilidade de elaborar um projeto de pesquisa com ênfase na coleta automatizada e também possível armazenamento primário dos ovos coletados.

Além destas lacunas que geraram a oportunidade do projeto, existe a motivação pessoal do autor, onde em seu círculo familiar possui produtores que não possuem uma vida social como gostariam por necessidade de estar presente no setor de postura de sua propriedade.

O projeto tem por foco, portanto, desenvolver um sistema capaz de satisfazer as condições de coleta e armazenamento em caso de ausência do produtor sem a necessidade de contratação de pessoal para fazer o serviço de coleta no caso de ausência do referido produtor.

Com o intuito de realizar as duas funções, coleta e armazenamento, descartamos a possibilidade de utilizar somente a calha com esteira móvel. Como neste caso, somente seria realizado a coleta e o transporte para algum lugar específico, sendo necessário, após este transporte, a utilização de outro método para realizar o armazenamento dos ovos.

Como um incentivo a mais que exemplifica a importância do projeto, também pode ser observado que este é pioneiro ao abordar características de coleta e armazenamento em recipientes próprios para este fim, ou seja, diretamente na caixa de ovos. Assim há, ainda, a possibilidade de aplicar os resultados deste trabalho como

um benefício à sociedade, oferecendo-o à agricultores familiares como forma de modernizar sua produção.

Finalmente, como exercício da profissão engenheiro de controle e automação, é de suma importância o conhecimento, planejamento e controle dos equipamentos existentes no mercado. Isto contribui para o crescimento e desenvolvimento profissional e pessoal, exigindo conhecimento sobre processos e melhoria contínua de sistemas produtivos, que é uma das linhas de pesquisa adotadas pela Faculdade Horizontina.

1.6 OBJETIVOS

1.6.1 Objetivo Geral

O objetivo geral deste trabalho é reduzir a quantidade de interações humanas no ambiente de postura de ovos em uma propriedade com o sistema de confinamento de aves do tipo gaiola, utilizando um braço robótico móvel com reconhecimento de posição e recolhimento de objetos.

1.6.2 Objetivos Específicos

Para cumprir com o objetivo geral, definiu-se os seguintes objetivos específicos:

a) identificar a quantidade diária de ovos produzidos em uma propriedade modelo.

b) analisar a quantidade de interações humanas, por período do dia, necessárias para a coleta dos ovos.

c) desenvolver um protótipo de braço robótico móvel para realizar a coleta e armazenamento dos ovos.

d) analisar a eficácia do sistema de coleta projetado.

2 REVISÃO DA LITERATURA

2.1 PRODUÇÃO DE OVOS

De acordo com a divulgação do IBGE (2019) pode-se observar no quadro 1, que havia no Brasil, no final 2018, 1.898 produtores de galinhas poedeiras. Entretanto, o Instituto realiza este cadastrado somente pela internet e informa que as informações não correspondem às produções totais das Unidades da Federação, uma vez que são pesquisados apenas os estabelecimentos com 10.000 (dez mil) ou mais galinhas poedeiras.

Quadro 1 - Produção de Ovos de Galinha

POG - Número de informantes, número de galinhas poedeiras e quantidade produzida de ovos de galinha, segundo as Unidades de Federação			
4º trimestre 2018			
Região	**Número de informantes (Unidades)**	**Número de galinhas poedeiras (Cabeças)**	**Quantidade produzida de ovos de galinha (Mil dúzias)**
Norte	62	3.951.601	21.304
Nordeste	162	23.616.719	142.657
Sudeste	606	78.079.424	451.847
Sul	919	37.132.995	193.688
Centro-Oeste	149	21.397.470	117.745
Brasil	**1.898**	**165.677.922***	**936.315**
As informações não correspondem às produções totais das Unidades da Federação, uma vez que são pesquisados apenas os estabelecimentos com 10.000 ou mais galinhas poedeiras.			
*A partir de janeiro de 2006, não são divulgados os dados para menos de 3 (três) informantes dentro do Estado, mas são contabilizados na quantidade produzida.			

Fonte: IBGE - Produção de Ovos de Galinha - 4º trimestre 2018

A região sudeste do Brasil é a maior produtora de ovos do país. Apesar da informação estar ocultada, o estado de São Paulo é o maior produtor nacional de ovos com uma produção de 275.432 dúzias no trimestre, mais do que a soma dos estados dentro de qualquer outra região.

Há outros milhares de produtores de ovos que não entram nessa estatística. Cada produtor, em sua realidade, procura produzir sempre mais e com a melhor qualidade possível, objetivando ter o menor gasto para alcançar alguma rentabilidade do seu negócio podendo competir com os valores de mercado em relação aos grandes produtores.

2.2 ROBÔ INDUSTRIAL - BRAÇO ROBÓTICO

De acordo com a ISO 8373 (2012), um robô industrial é definido como sendo um dispositivo manipulador multiuso programável e reprogramável em três ou mais eixos para uso em tecnologia de automação ou em um local fixo ou dispostos de tal forma que ele possa se mover. Diz ainda que um robô industrial compreende o manipulador (braço do robô ou garra), incluindo acionamentos; o controlador, incluindo unidade de controle e qualquer interface de comunicação (hardware e software).

Santos (2015) classifica um robô através das tarefas realizadas por ele tendo três tipos de natureza básica: movimentação, medição e manipulação. Este trabalho está classificado como movimentação, pois está dentro dos parâmetros sugeridos de uso: em operações de embalagem; para classificação de peças; na colocação e retirada de peças em centros de usinagem ou máquinas/ferramenta; para carga e descarga de depósitos; quando realizando paletização.

2.3 SISTEMA EMBARCADO

Cunha (2007) afirma que o sistema embarcado é o mesmo que colocar capacidade computacional dentro de um circuito integrado, equipamento ou sistema. Neste caso precisa-se entender que um sistema embarcado deve ser mais do que um simples computador. É necessário que seja um sistema completo e independente, que esteja preparado e programado para realizar apenas uma determinada tarefa.

Oliveira (2016) define um sistema embarcado como sendo uma "caixa" com componentes eletrônicos. Este entendimento facilita muito o desenvolvimento das aplicações, uma vez que todos esses componentes devem ser corretamente configurados para um funcionamento apropriado do sistema embarcado.

Normalmente são compostos por microcontroladores e possuem periféricos embutidos dentro de seu encapsulamento facilitando o desenvolvimento de produtos.

2.3.1 Microcontroladores

De acordo com Filho (2014), os microcontroladores são dispositivos programáveis, na sua maioria, autocontidos (sistemas embarcados ou embutidos), possuindo um sistema dentro de um chip, estando praticamente em todo lugar: automóveis, aviões, brinquedos, TVs, etc.

Estes dispositivos compõem sistemas computacionais que controlam os mais diferentes equipamentos como, por exemplo: controle de estacionamentos, sistemas de automação, sistemas de segurança.

O grande ganho destes dispositivos é possuir o Hardware e software integrados em um único chip, memória (de dados e de programa), entrada/saída, temporizadores, relógio interno, entre outros hardwares específicos (FILHO, 2014).

Segundo Oliveira (2016) os microcontroladores, em geral, são muito pequenos, mesmo contendo vários periféricos, como memórias, barramentos, *timers*, portas de comunicação, conversores de sinal analógico em digital etc.

> Esse dispositivo é amplamente usado em automação industrial, residencial e predial, eletrodomésticos, brinquedos eletrônicos e em qualquer situação em que seja necessário o controle de um dispositivo de sinais eletrônicos. Por exemplo, num elevador, quando alguém aperta um botão para ir até um andar específico, o microcontrolador recebe essa informação como um dado de entrada, interpreta-o e aciona os motores do elevador até aquele andar, para e abre as portas. (OLIVEIRA, 2016).

Diante do exposto, é notória a utilização de controladores para que as sequências de operações pré-programadas sejam atendidas na ordem e velocidade desejada. Após a execução destas operações, o processador aguarda novas instruções a serem computadas, repetindo-se indefinidamente.

2.4 ARDUINO

O Arduino pode ser definido como sendo uma placa de desenvolvimento com código aberto. Foi desenvolvida pelo italiano Massimo Banzi, em 2005, tendo como base facilitar o aprendizado em eletrônica a estudantes, devendo ser uma plataforma de baixo custo atendendo várias aplicações. Houve uma adesão mundial à esta

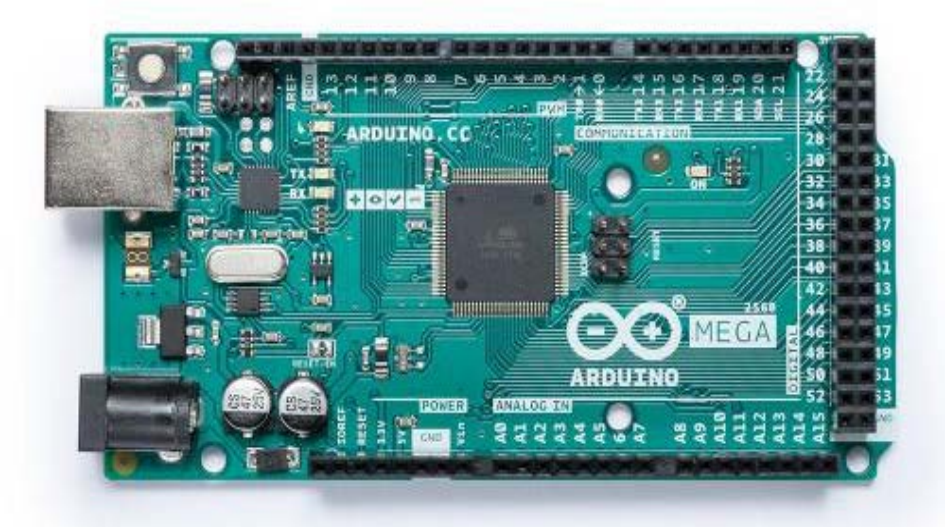
facilitada forma de trabalhar com eletrônica e diversos programadores trabalharam muito para desenvolver novas opções de produtos e códigos que atendessem as mais variadas aplicações.

O site, https://store.arduino.cc/usa/mega-2560-r3, da plataforma Arduino, dá a seguinte definição:

> "O Arduino é uma plataforma de prototipagem eletrônica *open-source* que se baseia em hardware e software flexíveis e fáceis de usar. É destinado a artistas, designers, hobbistas e qualquer pessoa interessada em criar objetos ou ambientes interativos. O Arduino pode sentir o estado do ambiente que o cerca por meio da recepção de sinais de sensores e pode interagir com os seus arredores, controlando luzes, motores e outros atuadores. O microcontrolador na placa é programado com a linguagem de programação Arduino, baseada na linguagem *Wiring*, e o ambiente de desenvolvimento Arduino, baseado no ambiente *Processing*. Os projetos desenvolvidos com o Arduino podem ser autônomos ou podem comunicar-se com um computador para a realização da tarefa, com uso de software específico". (ARDUINO, 2015).

Neste trabalho utilizar-se-á a placa de desenvolvimento Arduino Mega 2560 (Figura 1) que é uma placa microcontroladora baseada no ATmega2560.

Figura 1 - Placa de desenvolvimento Arduino Mega 2560



Fonte: Arduino (2015).

Como se pode observar, esta placa possui 54 pinos de entrada / saída digitais (dos quais 15 podem ser usados como saídas PWM), 16 entradas analógicas, 4 UARTs (portas seriais de hardware), um oscilador de cristal de 16MHz, uma conexão USB, conector de alimentação, um conector ICSP, e um botão de *reset*.

Na tabela 1, pode-se observar as especificações técnicas da placa Arduino Mega 2560 conforme consta em Arduino (2015).

Tabela 1 - Especificações Técnicas Arduino Mega

Microcontrolador	ATmega2560
Tensão operacional	5V
Tensão de entrada (recomendado)	7-12V
Tensão de entrada (limite)	6-20 V
Pinos Digitais I / O	54 (dos quais 15 oferecem saída PWM)
Pinos de entrada analógica	16
Corrente DC por pino de E / S	20 mA
Corrente DC por Pino 3.3V	50 mA
Memória *flash*	256 KB dos quais 8 KB usados pelo *bootloader*
SRAM	8 KB
EEPROM	4 KB
Velocidade do relógio	16 megahertz
LED_BUILTIN	13
comprimento	101,52 mm
Largura	53,3 mm
Peso	37 g

Fonte: Arduino (2015)

A placa Mega 2560 pode ser programada com o software Arduino (IDE) e o ATmega2560 já conta com a pré-programação de um *bootloader* que nos permite realizar o upload de um novo código sem o uso de um programador de hardware externo.

A programação é feita através da linguagem *Wiring*[1] e, ao enviar algum código ao IDE do Arduino, o mesmo converte o código escrito para a linguagem C e transmite ao compilador AVR-GCC[2], realizando a tradução dos comandos e informações para uma linguagem que o microcontrolador possa compreender.

[1] A *Wiring* é um projeto iniciado por Hernando Barragán e desenvolvido por uma pequena equipe de voluntários. É uma estrutura de programação de código aberto para microcontroladores e permite escrever software multi-plataforma para controlar dispositivos conectados a uma ampla gama de placas de microcontroladores e a possibilidade de criar todos os tipos de codificação criativa, objetos interativos, espaços ou experiências físicas.

[2] O *GNU Compiler Collection* (GCC) é um sistema compilador produzido pelo Projeto GNU que suporta várias linguagens de programação. AVR é uma família de microcontroladores desenvolvida desde 1996 pela Atmel.

A IDE do Arduino possui integrada uma linguagem própria baseada em C e C++ e apresenta alto grau de abstração, sendo assim, possibilita ao usuário usar o microcontrolador, mesmo sem conhecimento ou como é o funcionamento do mesmo.

De acordo com Souza (2013) o ciclo de programação do Arduino pode ser dividido em 4 etapas:

- A conexão da placa Arduino em uma porta USB do computador;
- Desenvolvimento do código ou *sketch* com os devidos comandos;
- Envio do código para a placa com a utilização da conexão USB;
- O Arduino irá receber os comandos e reiniciar, iniciando a execução do código desenvolvido.

Após o recebimento do *sketch* criado o Arduino passará a executar os códigos até que seja fornecido comandos de parada e não necessita mais a conexão do computador desde que esteja conectado em alguma fonte de energia.

2.5 PROGRAMAÇÃO

De acordo com o Priberam (2008-2013) a palavra programação indica o ato ou efeito de programar, mas, também pode ser visto como sendo a criação de um programa. O ato de programar é o mesmo que fornecer instruções a uma máquina ou a um mecanismo para um procedimento automático ou, também, submeter ou submeter-se a uma rotina ou aprendizagem para reagir de determinada maneira.

Carvalho (2017) faz uma abordagem sobre a história dos computadores relatando os primeiros sistemas físicos de contagem conhecidos citando, em sua pesquisa, que os seres humanos, desde 30.000a.C., já utilizavam algum sistema de contagem para definir colheitas e realizar festivais e cerimônias. Esses cálculos eram realizados com a utilização dos dedos ou ossos. Cita também que, em torno de 3.200a.C., onde acontece o início da escrita e a criação de vários sistemas numéricos e com isso permitindo um salto na forma de realizar diversos cálculos.

Com o passar do tempo é percebido que todos os métodos utilizados não eram suficientes para realizar cálculos rápidos e seria necessário a criação ou aperfeiçoamento de dispositivos e máquinas, onde estas deveriam, além de contagem e operações aritméticas, realizar de forma automática as mais variadas tarefas repetitivas e perigosas que necessitavam grande número de passos.

De acordo com Carvalho (2017) a primeira máquina que era considerada para realizar cálculos foi o ábaco, dispositivo criado que realizava operações aritméticas simples. Não se sabe ao certo de que época e onde foi desenvolvida. A mais antiga que se tem preservação é datada de 300 a.C. e é originada da Babilônia, não realizava cálculos automaticamente e assim como uma contagem manual o ábaco somente servia para armazenar os dados do cálculo que estava sendo realizado.

Mesmo apresentando essas limitações, o ábaco permite armazenar dados para operações mais complexas do que os sistemas anteriores, como as peças se movimentavam de um lado para outro e cada posição representa um valor. A complexidade do cálculo era limitada pela experiência e destreza do operador, ainda hoje há utilização do ábaco na China e no Japão.

Carvalho (2017) cita ainda, que a primeira calculadora mecânica foi construída em 1623 pelo inventor alemão Wilhelm Schickard e realizava as quatro operações básicas, os valores armazenados durante o cálculo eram representados pelo posicionamento de rodas dentadas. Tem-se, também, a calculadora mecânica Pascaline, desenvolvida em 1642 por Blaise Pascal, matemático francês, que ainda está preservada e é formada por uma caixa com muitas engrenagens, rodas dentadas e visores, a máquina possui oito rodas dentadas para a representação de oito dígitos.

Hoje em dia ainda são utilizados vários produtos com este princípio de funcionamento, podemos citar o hodômetro[3] mecânico dos veículos, que a cada 10 voltas da engrenagem de 100 metros a roda gira acrescentando mais 1 quilômetro.

Carvalho (2017) cita também o inglês Charles Babbage, que entre 1821 e 1849, após várias tentativas de projetar uma máquina de diferenças finitas, que fosse capaz de calcular uma série de valores numéricos usando somente a adição. Nesse meio tempo, entre fracassos da máquina de diferenças finitas, em 1834 ele projetou a máquina analítica que possuía muitas das principais estruturas lógicas dos computadores atuais, cuja operação era sequenciada e realizava vários cálculos com precisão de até 50 casas decimais. A máquina controla a execução de uma sequência de operações pela leitura de um cartão perfurado com uma sequência de instruções e através desta sequência de cartões a serem lidos. Os locais onde os cartões eram perfurados, seria definido o funcionamento da máquina e, formando assim, um programa de computador.

[3] Dicionário Priberam (2008-2013) define hodômetro com sendo um instrumento para medir as distâncias percorridas ou para contar o número de voltas de uma manivela.

A partir destas informações, a matemática Augusta Ada Byron, conhecida como Ada Lovelace, fez a publicação de um manual onde mostrava o funcionamento da máquina analítica e de como poderia ser programada para realizar as várias operações, sendo assim, conhecida como a primeira programadora de computadores.

No Brasil, estes cartões perfurados, foram utilizados até a década de 1980 para a programação de computadores digitais de grande porte.

Camargo (2017) afirma que em 1890, o americano Herman Hollerith inventou e construiu uma máquina para tabulação de cartões perfurados, baseado na máquina analítica proposta por Charles Babbage. A máquina detectava os locais das perfurações nos cartões e foi utilizada para processar os dados do censo americano. Com essa máquina, a censo da população foi concluído em seis meses, tendo assim uma grande economia de tempo e dinheiro. Para efeito de comparação, o censo da década anterior, ou seja, de 1880, foi todo processado manualmente e foi finalizado somente 12 anos depois de realizada a contagem.

Hollerith desenvolveu várias outras máquinas, onde o processo de alimentação de cartões era feito de maneira automática, da adição de números e para a ordenação dos cartões. Em 1896 Hollerith fundou a companhia de máquinas de tabulação, iniciando assim, a IBM (*International Business Machines*), uma das maiores empresas de computação até os dias atuais. Somente a partir dos avanços tecnológicos na área eletrônica do século XX, houve construções de computadores IBMs em escala comercial que utilizavam componentes mecânicos controlados eletronicamente.

No ano de 1938, foi finalizado pelo alemão Konrad Zuse a construção do computador Z1. Este é considerado o primeiro computador eletromecânico operacional e o primeiro computador programável do mundo. Utilizava lógica booleana e aritmética de ponto flutuante, já incluía várias características dos computadores atuais, como unidade de controle, unidade de memória e operações de ponto flutuante.

Em 1944 foi construído o Mark 1, para ser utilizado pela Universidade de Harvard. Ele era baseado em relés mecânicos e ocupava uma sala inteira. Este equipamento era revolucionário para a época, podendo realizar até três operações de soma ou subtração por segundo. A tecnologia de válvulas que foi desenvolvida na mesma época logo tornou esses computadores sem utilidade, pois com esse novo sistema era permitido a construção de computadores completamente eletrônicos.

O ENIAC, finalizado em 1945, foi o primeiro computador eletrônico digital de propósito geral, construído na Universidade da Pensilvânia, nos Estados Unidos. Utilizado para realizar cálculos de balística durante a Segunda Guerra Mundial, a programação era feita conectando e desconectando manualmente 6.000 cabos.

EDVAC, projetado por John Von Neumann, foi o primeiro computador que utilizou programas previamente armazenados em fitas magnéticas, hoje, este armazenamento pode ser feito com CDs, DVDs, cartões de memória, *pen drives*, etc. EDVAC trabalhava internamente com valores binários (0 e 1) e era dividido em três partes: CPU, unidade de memória e dispositivos de entrada e saída, a construção deste foi finalizada em 1951.

Hoje os computadores são conhecidos como microcomputadores ou computadores pessoais, com tamanho e custo muito reduzidos comparados com seus antecessores. Isso ocorre devido aos avanços que aconteceram na microeletrônica e consequente redução dos custos dos dispositivos eletrônicos.

Um dos primeiros microcomputadores a serem comercializados foi o Altair 8800, lançado em 1975 e permitia a execução de programas escritos na linguagem de programação Basic.

A partir deste ponto, várias empresas e produtos foram lançados no mercado. Produtos mais evoluídos no sentido de espaço de armazenamento, velocidade de processamento, efeitos visuais, menor tamanho e mais leves, tornando a vida e o dia-a-dia cada vez mais dependente do uso das máquinas. Não utilizados somente para cálculos matemáticos, mas, também, para o gerenciamento de tudo que está a nossa volta, organizando e liberando o ser humano de tarefas repetitivas e muitas vezes entediantes.

2.5.1 Linguagem de Programação

Oliveira (2015) diz que: o Arduino, assim como qualquer outro dispositivo programável, necessita de uma linguagem de programação, definindo como sendo um conjunto de comandos que permitirão o desenvolvimento de um programa para computador ou qualquer outro dispositivo programático.

Alves (2014) sugere que existem diversos tipos de linguagens de programação, mas que trabalham com formas, estrutura e diferentes comandos. Como no início da era da computação os programadores necessitavam programar em baixo nível, ou seja, diretamente em linguagem de máquina, também chamado de código binário,

onde eram utilizadas inúmeras combinações de 0s (zeros) e 1s (uns). Nesta época era praticamente impossível decorar a função que cada combinação exercia no processador. A partir deste pensamento, no início dos anos 50, quando os computadores ainda eram a válvulas, desenvolveu-se uma linguagem mais próxima do entendimento humano, esta linguagem foi denominada de *Assembly*. Nesta linguagem, as instruções binárias são representadas por palavras ou siglas genericamente denominadas de mnemônicos.

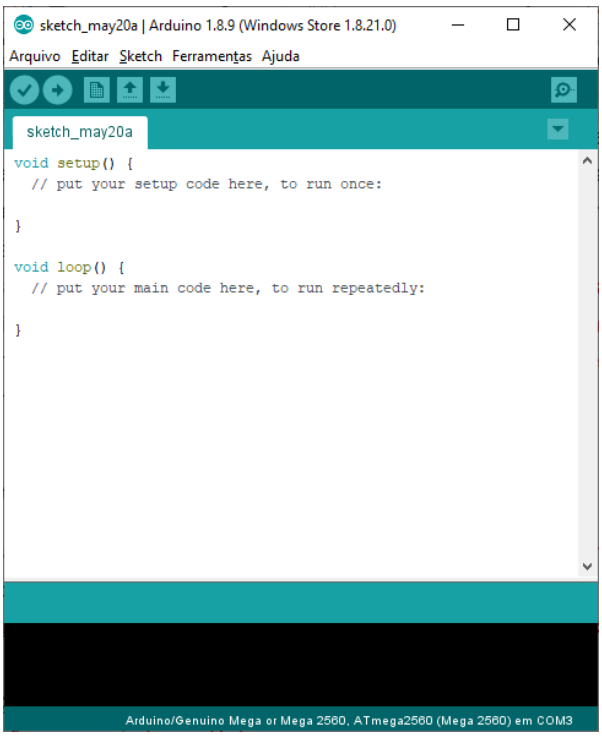
Diversas linguagens surgiram para facilitar a vida dos programadores, entre elas *Pascal, BASIC, C, C++, Java, Python*, entre outras. Uma das linguagens de programação mais usadas mundialmente é a Linguagem C que, de acordo com Manzano (2015), foi desenvolvida em 1972 por Dennis M. Ritchie e nasceu pela necessidade de escrever programas que utilizassem os recursos interno da máquina de uma forma mais fácil do que a linguagem de montagem *Assembly*. A linguagem C é uma linguagem de alto nível e obteve grande aceitação de programadores pelo seu alto grau de portabilidade. Um programa escrito nesta linguagem, em teoria, pode rodar em qualquer plataforma computacional, permite a integração direta entre alto nível e baixo nível que possibilita a escrita de código *Assembly* dentro do código em alto nível, a linguagem C foi desenvolvida para ser de propósito geral e estruturada.

É importante considerar o que é afirmado por Oliveira (2015) pois, segundo ele, a solução para um determinado problema computacional, pensada pelo programador e descrita em linguagem de programação, é chamada algoritmo, assim sendo e complementando a explicação, diz ainda:

> Por intermédio da linguagem, o programador consegue especificar como os dados serão tratados, armazenados e manipulados em um computador. Essa organização de comandos e regras é chamada de código-fonte. Uma vez criado esse código-fonte, temos toda a solução lógica (algoritmo) e comandos. Este é traduzido para uma linguagem de mais baixo nível, de máquina, para ser compreendida e executada pelo processador (OLIVEIRA, 2015).

Com base no que foi citado até aqui conclui-se que, para programar sistemas embarcados, será necessário um ambiente de desenvolvimento e compilação. Assim, se utilizará para este trabalho a programação direta no IDE do Arduino, conforme representado na figura 2. Esta linguagem é modelada a partir da linguagem *Wiring*, é uma linguagem própria do IDE baseada na linguagem C e C++.

Figura 2 - Interface do IDE do Arduino



Fonte: O Autor, 2019.

Através desta IDE é feita a programação e gerado o código-fonte que, no Arduino, é chamado de *sketch*. O compilador traduz este código e ele estará pronto para ser enviado e gravado na memória *flash* do microcontrolador do Arduino.

2.5.1.1 - Linguagem *Wiring* baseada em C /C++, utilizada no IDE do Arduino

Júnior (2015) enfatiza que toda linguagem de programação possui um conjunto básico de elementos (símbolos e palavras) de sintaxe para a escrita de comandos.

A estrutura do *sketch* é formada por 3 blocos, cada um com sua representatividade dentro do código. Estes blocos podem ser descritos da seguinte forma: Júnior (2015) afirma que o primeiro bloco é onde serão declaradas as variáveis globais do nosso código, Júnior (2015) e Oliveira (2015) concordam em dizer que o segundo bloco é da função *setup*() ou, se preferir *void setup*(). Este bloco é utilizado para a inicialização ou configurações iniciais, os comandos que estão nesta função serão executados somente uma única vez, o terceiro bloco é a função *loop*() ou *void loop*(). As funções dentro deste bloco serão executadas repetidamente, ou seja, quando a última linha for executada do código, começa a ser lido novamente todas as funções que estão dentro deste bloco, até que a placa seja desligada, botão *reset* seja

pressionado ou tenha encontrado alguma condição de parada dentro de alguma das funções escritas.

Com base no exposto acima, é necessária a apresentação de alguns símbolos que fazem parte desta linguagem de programação e que nos permitem construir cada um destes blocos com o código de forma correta e acertada:

- Todos os procedimentos de uma função devem estar limitados por chaves “{ }”;
- Cada procedimento deve ser finalizado por um ponto e vírgula ";";
- Usa-se "//" antes de um comentário em uma linha;
- Usa-se "/* */" para comentar um conjunto de linhas.

Além dos símbolos básicos apresentados acima, faz-se necessário também, o conhecimento das sintaxes e das funções, para isso é realizada uma separação quanto ao conhecimento da programação em tópicos, que serão: tipos de dados; operações e operadores lógicos e aritméticos.

2.5.1.1.1 Tipos de Dados

Os mais comuns tipos de dados (variáveis) utilizados no Arduino são:

- *boolean*: valor verdadeiro (*true*) ou falso (*false*);
- *chat*: um caractere;
- *byte*: um *byte*, ou sequência de 8 bits;
- *int*: número inteiro de 16 bits com sinal (-32768 a 32767);
- *usigned int*: número inteiro de 16 bits sem sinal (0 a 65535);
- *long*: número inteiro de 16 bits com sinal (-2147483648 a 2147483647);
- *unsigned long*: número inteiro de 16 bits sem sinal (0 a 4294967295);
- *float*: número real de precisão simples (ponto flutuante);
- *double*: número real de precisão dupla (ponto flutuante);
- *string*: sequência de caracteres;
- *void*: tipo vazio (não tem tipo).

Já os tipos padronizados de constantes são:

- *True*/*False*: constantes booleanas (Verdadeiro/Falso);
- *HIGH*/*LOW*: constantes de nível lógico (Alto/Baixo);
- *INPUT*/*OUTPUT*: constantes de direção (Entrada/Saída).

2.5.1.1.2 Operações e operadores lógicos e aritméticos

Abaixo segue relação dos operadores lógicos e aritméticos disponíveis:

Operadores aritméticos
% (restante)
* (multiplicação)
+ (adição)
- (subtração)
/ (divisão)
= (operador de atribuição)

Operadores de Comparação
! = (não é igual a)
< (menor que)
<= (menor ou igual a)
== (igual a)
> (maior que)
> = (maior que ou igual a)

Operadores Compostos
% = (resto composto)
& = (composto bit a bit e)
* = (multiplicação de compostos)
++ (incremento)
+ = (adição de compostos)
- (decremento)
- = (subtração composta)
/ = (divisão composta)
^ = (composto bit a bit xor)
| (composto bit a bit ou)

Operadores booleanos
! (não lógico)
&& (lógico e)
|| (lógico ou)

2.5.1.1.3 Estruturas de controle ou condicionais

Uma instrução condicional é uma forma de auxiliar na tomada de decisões. Pode ser exemplificado pela instrução condicional *if* que verifica uma condição e executa a instrução ou conjunto de instruções, se a condição for 'verdadeira', por exemplo, a instrução *digitalWrite*() que somente será executada caso a variável temperatura tenha o valor inferior a 15, o código para este exemplo seria:

```
4 if (temperatura < 15)           //quando a variável temperatura inferior a 15
5 {                               // início da condição verdadeira
6   digitalWrite(ledPort, HIGH);  // executa quando a condição for verdadeira
7 }                               // finaliza a condição verdadeira
```

Abaixo, segue a apresentação das principais estruturas de controle da programação em C/C++.

if()
if()...else
for()
switch()...case
while()
do()... while
break
continue
return
goto

2.5.1.1.4 Funções

De acordo com Monk (2017) "as funções são ferramentas chaves para criar *sketches* de fácil compreensão. Elas podem ser modificadas sem dificuldade e sem risco de a coisa toda se transformar em uma grande confusão". Ele afirma, também,

que uma função é um pouco parecida com um programa dentro de outro programa e que podem ser usadas para “empacotar” o código, podendo juntar as partes que podem ficar juntas. Desta forma as funções podem ser chamadas a partir de qualquer posição do *sketch* e pode conter variáveis próprias e seus próprios comandos como pode ser visto no exemplo abaixo:

```
6  void loop() {
7    temp();                          //chamada da função "temp"
8  }
9                                     // função "temp" criada pelo usuário
10 void temp()                        // nome dado à função
11 {                                  // início da função
12   int temperatura = 0;             // declarando variável "temperatura"
13   if (temperatura < 15)            // quando a variável temperatura inferior a 15
14   {                                // início da condição verdadeira
15     digitalWrite(ledPort, HIGH);   // executa quando a condição for verdadeira
16   }                                // finaliza a condição verdadeira
17 }                                  // fim da função
```

Monk (2017) enfatiza a parte da execução, afirmando que quando termina a execução de todos os comandos internos da função, a execução do *sketch* continua na próxima linha de onde a função foi chamada.

2.5.1.1.5 Bibliotecas

Bibliotecas são conjuntos de códigos e funções previamente armazenados. Normalmente são programados por outros usuários e pode ser reutilizado em outras aplicações, neste caso facilitando o desenvolvimento de novos códigos com o reaproveitamento dos já existentes.

Com o reaproveitamento do código já programado, não é necessário digitar todas as linhas de comando com as funções para que um motor seja ativado ou um sensor consiga transmitir e receber um sinal. Para o uso destas bibliotecas simplesmente as incluímos no código conforme mostrado abaixo:

```
1 #include <Servo.h>
2 #include <Ultrasonic.h>
```

2.6 MOTORES ELÉTRICOS

Para Petruzella (2013), os motores elétricos são utilizados para converter energia elétrica em energia mecânica. Representam uma das invenções mais úteis

na indústria elétrica. Funcionam com base em magnetismo e correntes elétricas. O magnetismo é a força que produz a rotação de um motor.

Braga (2014) define motor elétrico como sendo a principal forma de propulsão de muitos dispositivos que tenham partes mecânicas, como: robôs, braços mecânicos, automatismos etc., diz também que podem ser separados em dois tipos: os DC e os motores de passo.

2.6.1 Motor DC com caixa de Redução

Petruzella (2013) acredita que os motores de corrente contínua são usados em menor quantidade do que os motores do tipo CA. Faz apontamentos afirmando que é vantajoso transformar a corrente alternada em corrente contínua onde o torque preciso e o acionamento de velocidade são exigidos.

De acordo com Braga (2104) os motores DC à disposição dos projetistas de Robótica e Mecatrônica são motores de alta rotação e pequeno torque, não servindo para a maioria das aplicações. No entanto ao reduzir sua velocidade e, ao mesmo tempo, aumentar seu torque, podem ser usados para diversos tipos de projetos, mas para isso ser possível é necessário acoplar ao motor algum sistema mecânico que realiza essas alterações, este sistema é chamado de caixa de redução (figura 3) e normalmente são construídos com uma série de engrenagens.

Figura 3 - Motor DC com caixa de redução

Fonte: O Autor, 2019.

Braga (2014) conclui que a relação entre os tamanhos e o número de dentes das engrenagens nos dá a taxa de redução da velocidade e também o aumento da força obtida.

2.6.2 Servo Motor

Silveira (2017) ressalta que um servo motor (figura 4) nada mais é do que um motor comum com controladores e encoder[4] acoplados.

Figura 4 - Servo Motores

Micro servo SG90 Servo Motor MG995

Fonte: O Autor, 2019.

Pode-se dizer que um servo motor é um atuador que permite ter controle de velocidade e de posição.

O sítio virtual, www.vidadesilicio.com.br, da empresa Vida de Silício, sugere que o Micro Servo motor SG90 é um dos servos mais populares quando o assunto é Arduino ou robótica educacional. Isso porque ele possui um tamanho pequeno e torque adequado para maior parte das aplicações educacionais.

Os modelos de servo motores que estão em uso neste trabalho seguem a seguinte ligação (como em grande maioria dos servos motores que estão no mercado):

Fio Laranja: Sinal PWM

Fio Vermelho: +VCC

Fio Marrom: GND

[4] Encoder são dispositivos/sensores eletromecânicos cuja funcionalidade é transformar posição em sinal elétrico digital. (www.hitecnologia.com.br)

2.6.3 Motor de passo

De acordo com Agnihotri (2011) o motor de passo é definido como sendo um motor síncrono sem escovas que divide uma rotação completa em várias etapas.

A figura 5 apresenta o motor de passo 28BYJ-48 e o Driver ULN2003. Componentes utilizados em conjunto para a movimentação da garra que executará o movimento de abrir e fechar para o recolhimento do objeto.

Figura 5 - Motor de passo 28BYJ-48 + Driver ULN2003

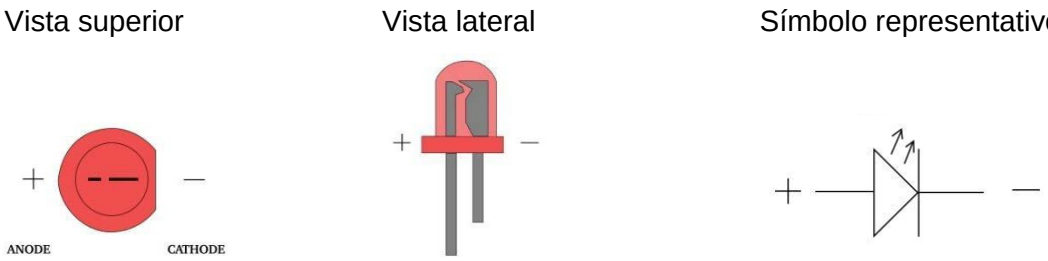
Fonte: O Autor, 2019

O driver ULN2003 foi desenvolvido especialmente para o uso com este tipo de motor facilitando, assim, a configuração e controle do mesmo. O motor de passo conta com caixa de redução interna em uma relação de redução de 1/64, seu torque máximo é de 2,2 Kgf.cm e o ângulo do passo é 5,625 que equivale a 64 por volta dada.

2.7 LED - DIODOS EMISSORES DE LUZ

Goeking (2009) afirma que os diodos emissores de luz, conhecidos como *Leds* (figura 6) por sua nomenclatura em inglês (*Light Emitting Diode*), são compostos por diodos semicondutores que convertem eletricidade em luz visível.

Figura 6 - Diodo emissor de luz



Fonte: Modificado de <https://www.filipeflop.com/?attachment_id=14162>.

A utilização de LED neste projeto é aplicada como componente para informação visual. Quando a caixa de armazenamento estiver completa o LED se acende informando este evento.

2.8 SENSOR ULTRASSÔNICO HC-SR04

Mota (2018) se refere ao sensor ultrassônico HC-SR04 (figura 7) afirmando que tudo começa pela emissão de um pequeno pulso sonoro de alta frequência que se propagam na velocidade do som no meio em questão. Quando este pulso atingir um objeto, um sinal de eco será refletido para o sensor.

Figura 7 - Sensor ultrassônico HC-SR04

Fonte: O Autor, 2019.

Através do eco captado pelo receptor do sensor ultrassônico, é possível medir a distância através da análise do tempo que o pulso sonoro leva desde o momento da emissão, até refletir em um obstáculo e retornar ao receptor, para tanto se usará a equação 1 para realizar os cálculos matemáticos para a conversão do tempo medido em distância.

$$D = \frac{t_m * V_s}{2} \quad (1)$$

A equação 1 está de acordo com o Manual do Usuário do HC-SR04, onde

D: distância do teste

t_m = tempo de alto nível

V_s = velocidade de propagação ultrassônica no ar (tabela 2)

O denominador 2 é utilizado para encontrar a metade do tempo, ou seja, o tempo de ida até o objeto.

Tabela 2 - Velocidade de Propagação do Som em Gases

Velocidade de Propagação do Som em Gases	
Substância	**Velocidade(m/s)**
Ar (0 ºC)	331
Ar (20 ºC)	**343**
Dióxido de carbono (0 ºC)	259
Oxigênio (0 ºC)	316
Hélio (0 ºC)	965

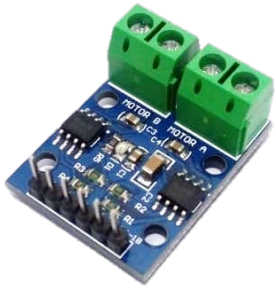
Fonte: Adaptado de Cutnell (2016 p.479).

Como visto, deve-se usar a velocidade de propagação do som no ar na equação 1 e, para tanto, é necessário usar a especificação de acordo com o que Cutnell (2016) mostra na tabela 2, substância: Ar, Temperatura: 20º C (temperatura ambiente), Velocidade do Som: 343m/s.

2.9 DRIVER DUPLO PONTE H L9110S

De acordo com a UsinaInfo, que é uma das maiores lojas virtual de Peças para Robótica e ferramentas para eletrônica do Brasil e revendedora do L9110s, o Driver Duplo Ponte H L9110s, conforme a figura 8, é um componente altamente tecnológico. Desenvolvido para aplicação em projetos eletrônicos e pode ser aplicado para controlar dois motores DC independentes ou um motor de passo bifásico quatro fios, controlando o sentido do giro e a velocidade dos motores.

Figura 8 - Driver Duplo Ponte H L9110s



Fonte: O Autor, 2019.

Bernardo (2016) auxilia informando também que a ponte H é basicamente um circuito eletrônico que permite o controle de motores DC e motores de passo. Dependendo dos componentes envolvidos, este controle permite direcionarmos o motor para um determinado sentido de rotação, horário ou anti-horário.

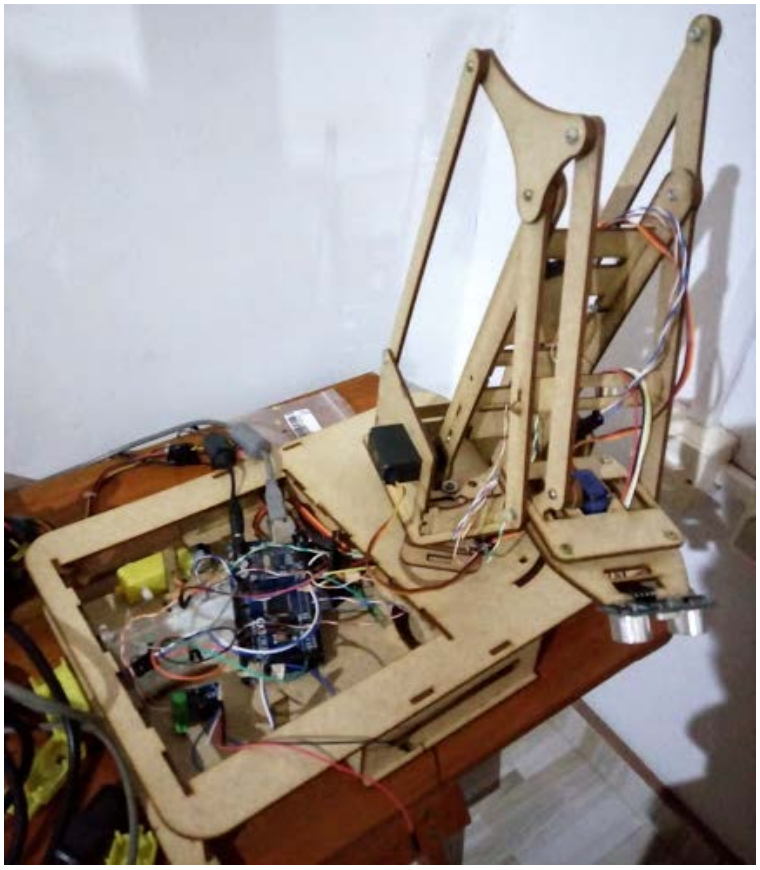
3 METODOLOGIA

Para a realização deste trabalho se fez uso da observação, análise e interpretação de resultados obtidos. A avaliação se dará por análise dos resultados gerados após o desenvolvimento do protótipo do braço robótico móvel para recolhimento dos ovos de galinhas poedeiras criadas em sistema tipo gaiolas.

Segundo Gil (2017), a pesquisa experimental constitui o delineamento mais prestigiado nos meios científicos consistindo, essencialmente, em determinar um objeto de estudo, onde se possa selecionar as variáveis que podem influenciar e definir formas para o controle e a possibilidade de observar os resultados produzidos no objeto. Gil (2017) define, ainda, que no modelo clássico de pesquisa experimental, o pesquisador precisa manipular pelo menos um dos fatores que se acredita ser responsável pela ocorrência do fenômeno que está sendo pesquisado.

Neste sentido será usado o braço robótico para recolhimento dos ovos (figura 9) como objeto de estudo, onde será identificada a posição, o deslocamento e a eficácia do projeto em relação ao seu uso em condições práticas.

Figura 9 - Braço robótico móvel



Fonte: O Autor, 2019.

3.1 INFORMAÇÕES DO PRODUTOR

Para que o projeto estivesse de acordo com as reais intenções, foi necessário o contato com um produtor de ovos da região. O fornecimento dos devidos parâmetros iniciais foi de suma importância para estruturar o projeto conforme a sua produção diária. O produtor também informou alguns gastos relativos à manutenção das galinhas poedeiras conforme dados apresentados no quadro 2, gerando assim, uma expectativa de lucros referente ao tempo médio de produção.

Quadro 2 - Dados apresentados pelo produtor

Custo no período de crescimento				
Custos	**Aves**	**Quant.**	**Unit.**	**Total**
Pintos de 1 dia	20	20	R$ 5,90	R$ 118,00
Perdas	5%	1	R$ 5,90	R$ 5,90
Ração Cria	19	0,1	R$ 1,32	R$ 2,51
Ração Recria	19	0,1	R$ 1,32	R$ 2,51
Mão de Obra	1	3	R$ 0,10	R$ 180,00
Sub total 1				R$ 303,02

Custo no período de produção				
Perdas	5%	1	R$ 5,90	R$ 5,90
Ração	18	1	R$ 1,32	R$ 23,76
Mão de Obra	1	14	R$ 0,15	R$ 1.134,00
Sub total 2				R$ 1.151,86

Total (1+2)				R$ 1.454,88

Depreciação		4%		R$ 58,20
Custo Financeiro		16%		R$ 232,78
Sub Total 3				R$ 290,98

Total (1+2+3)				R$ 1.745,85

Descarte (75%)	14	2	R$ 1,61	R$ 45,08
Sub Total 4				R$ 45,08

Total Geral (1+2+3-4)				R$ 1.700,77

Produção de Ovos (dz)	410			
Produção Diária (un.)	12			
Custo Dúzia				R$ 4,15
Valor de Venda				R$ 5,00
Lucro				21%

Fonte: O Autor, 2019, modificado de SAKOMURA (2014) com dados do produtor Ademir Gessinger de Santa Rosa, Rio Grande do Sul.

Todas as informações do quadro 2 foram fornecidas pelo produtor, exceto o valor do frango vivo (descarte) que está de acordo com o que a Embrapa (2019) sugere como preço de venda.

Pode ser visto no quadro 2 que a produção excede a expectativa com relação à média de produção tanto no Rio Grande do Sul que é de 53% como também a média nacional que está em 57%, como pode ser visto no quadro 1. O produtor está conseguindo uma média de 65% de produção comparando a quantidade de galinhas que possui com a produção diária de ovos.

O produtor informou também que realiza várias coletas durante o dia, como não cumpre horários específicos diz que geralmente realiza entre 5 e 7 coletas diárias.

3.2 MATERIAIS E EQUIPAMENTOS

Para montar o chassi do robô foi primeiramente feita uma lista de itens que são necessários estar presentes e, com isso, ter noção do espaço a ser utilizado. Como segue:

- *Protoboard*
  Utilização a *protoboard* para fazer as conexões da parte eletrônica, podendo assim realizar todos os testes antes das conexões definitivas serem feitas.
- Arduino
  Utilizado para controle do braço robótico.
- Fonte de alimentação
  Usado uma fonte de computador com saídas em diferentes tensões, 5V para alimentação dos motores e sensores, e saída 12V para alimentação do Arduino, saída 3.3V não foi utilizada.
- Dois Motores DCs e duas Caixas de redução
  Os motores em conjunto com as caixas de redução são utilizados para movimentar o robô lateralmente. As caixas de redução são utilizadas para transmitir torque ao eixo e para balancearmos a relação entre torque e RPM do motor.
- Um *driver* L9110
  Necessário para a correta distribuição de energia permitindo o controle da rotação e direção do giro das rodas.

- Três Servo motores

  Cada servo motor será responsável por um determinado movimento do braço robótico, um para subir e descer, um para girar esquerda e direita e um para alongar o braço.
- Um motor de passo e driver controlador ULN2003

  Utilizado para abrir e fechar a garra com excelente controle para abertura e fechamento dos dedos.
- Três Sensores Ultrassônicos

  Um sensor utilizado para detectar o objeto e os outros dois para detectar o final do percurso do movimento lateral.
- Quatro Rodas

  Utilizada para transformar o torque do eixo em movimento do robô.
- *Ball Caster*

  Esfera metálica deslizante utilizada na parte inferior do conjunto do braço robótico para que o braço tenha uma rotação equilibrada com o carro que fará o deslocamento lateral.
- MDF 3, 6 e 9mm

  MDF com 3mm de espessura utilizado para toda estrutura do carro de movimentação lateral e também do conjunto inteiro do braço robótico. MDF de 6 e 9mm utilizados para a construção do conjunto da garra.

Além do citado acima, também será necessário a utilização dos seguintes recursos:

- Alicate e Fiação

  Necessários para conexão dos componentes;
- Computador

  Realizar a programação do controlador;
- Papel e caneta

  Eventuais anotações pontuais;
- Chave de fenda, parafusos, cola e fita adesiva

  Fixação das peças em MDF e componentes eletrônicos;
- Ferro de solda e arame de estanho

  Fixação dos componentes eletrônicos;
- Multímetro

  Conferência de tensão e continuidade na fiação;

- Capacitor
  Regular a tensão para o sistema eletrônico;
- Resistor
  Resistência para evitar danos ao Led;
- Led vermelho
  Identificador visual quando a caixa estiver completa;
- Botão liga/desliga/emergência
  Ativa ou elimina a energia de todo o sistema.

3.2.1 Montagem da parte mecânica

Visando a necessidade dos materiais e componentes a serem colocados, mais o espaço necessário para que o ovo recolhido seja depositado, foi feito um esquema que utilizasse um espaço relativamente pequeno. Na Figura 10 tem-se uma ilustração da vista superior do chassi do carro que fará o deslocamento lateral com suas devidas medidas externas. Na figura 11 está a parte do teto do carro que também é a base do braço robótico e a base para o depósito dos objetos recolhidos.

Figura 10 - Chassi do carro de movimentação lateral

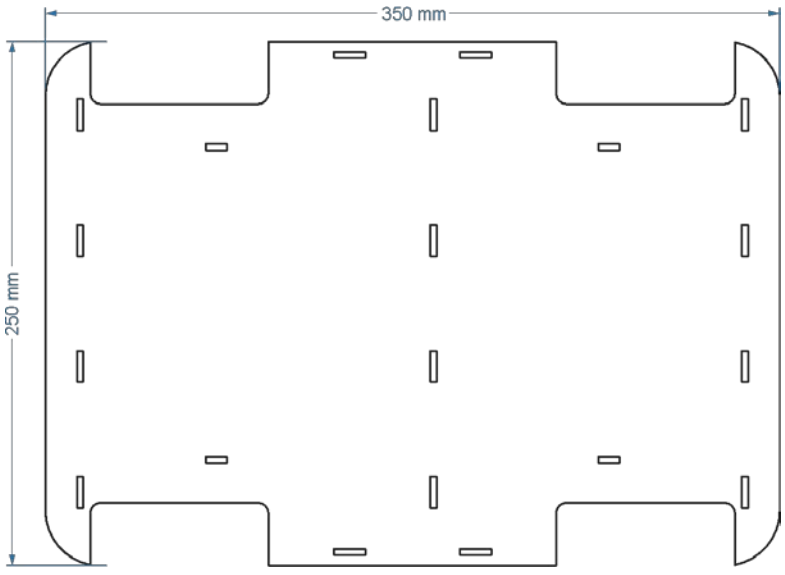
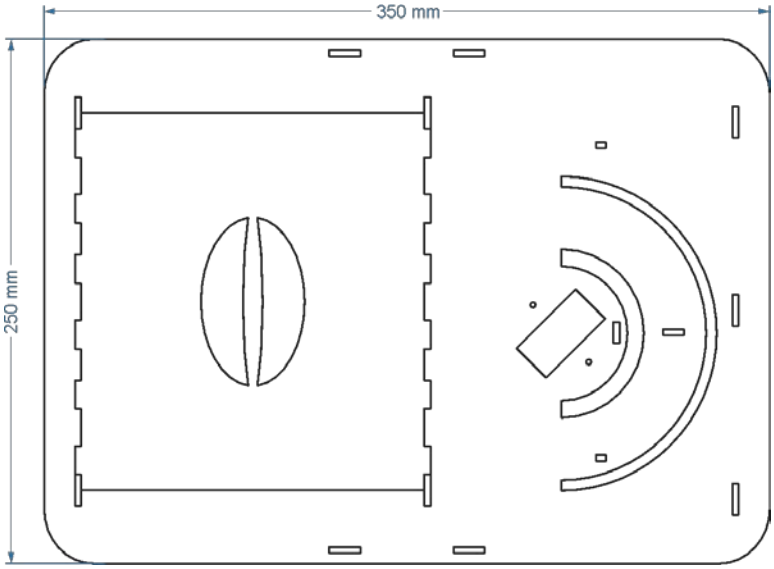


Figura 11 - Base do braço robótico e depósito dos objetos recolhidos



Fonte: O Autor, 2019.

Nas figuras 10 e 11 pode-se observar qual será o tamanho final do conjunto carro de deslocamento, tanto na largura como comprimento, a altura do conjunto se dará com a instalação do braço robótico. As imagens também mostram os pontos de encaixe para que a estrutura fique firme e estável.

Na figura 10 pode ser percebido cortes entrantes nas laterais da base, estes encaixes são para que as rodas possam ser colocadas diretamente no eixo do motor DC sem a necessitar de um prolongamento deste eixo. Na figura 11 pode-se observar o encaixe de uma “tampa”, necessidade verificada para o acesso facilitado após a montagem do Arduino e fiações de ligação aos componentes. Estes ficarão armazenados na parte interna do carro de deslocamento lateral, fazendo com isso que a fiação e componentes fiquem em lugar protegido contra eventuais danos ou esbarrões acidentais.

Para que a implementação esteja completa, fez-se necessário o desenvolvimento de funções de deslocamentos e também funções para o sistema de controle de cada junta do braço robótico. Estas funções são necessárias para que se consiga um melhor desempenho no controle da localização do objeto em questão e também da movimentação das juntas para pegar e soltar o objeto recolhido em seu devido lugar.

3.2.2 Montagem da parte eletrônica

Para o controle do braço robótico utilizou-se a placa Arduino Mega 2560. Sua função é realizar o controle dos motores DC, que farão o movimento de deslocamento lateral do carro que também é a base do braço robótico, controla também os servo motores responsáveis pelo movimento das articulações do braço robótico. Controla também os sensores de distância infravermelho, além de receber e processar os dados vindos dos três sensores de distância. Para os motores DC, o controle é realizado através de uma ponte H dupla que recebe o sinal digital e, assim, controla o sentido da corrente passando para os motores, alternando o sentido da corrente há alternância no sentido de movimentação do carro de movimentação lateral. No caso dos Servo motores, o próprio Arduino envia o sinal diretamente pois já possui uma ponte H embutida em seu sistema.

Estão sendo utilizados três servos motores da TowerPro e um motor de passo da Rohs. Um servo é modelo micro servo SG90 e dois do modelo MG995 Metálico. O servo SG90 está sendo utilizado para a movimentação do extensor do braço, já os MG995 fazem a função de abaixar/levantar o braço e de girar para a direita/esquerda o conjunto inteiro do braço robótico. O motor de passo é utilizado para a abertura e fechamento dos dedos da garra. Como os 3 servos juntos, mais o motor de passo, oferecem uma carga mecânica superior a corrente fornecida pelo Arduino é necessária uma fonte externa para a alimentação. Para tanto está sendo utilizando uma das saídas de 5V com capacidade de suportar até 18A (figura 12).

Figura 12 - Fonte de alimentação

Fonte: O Autor, 2019.

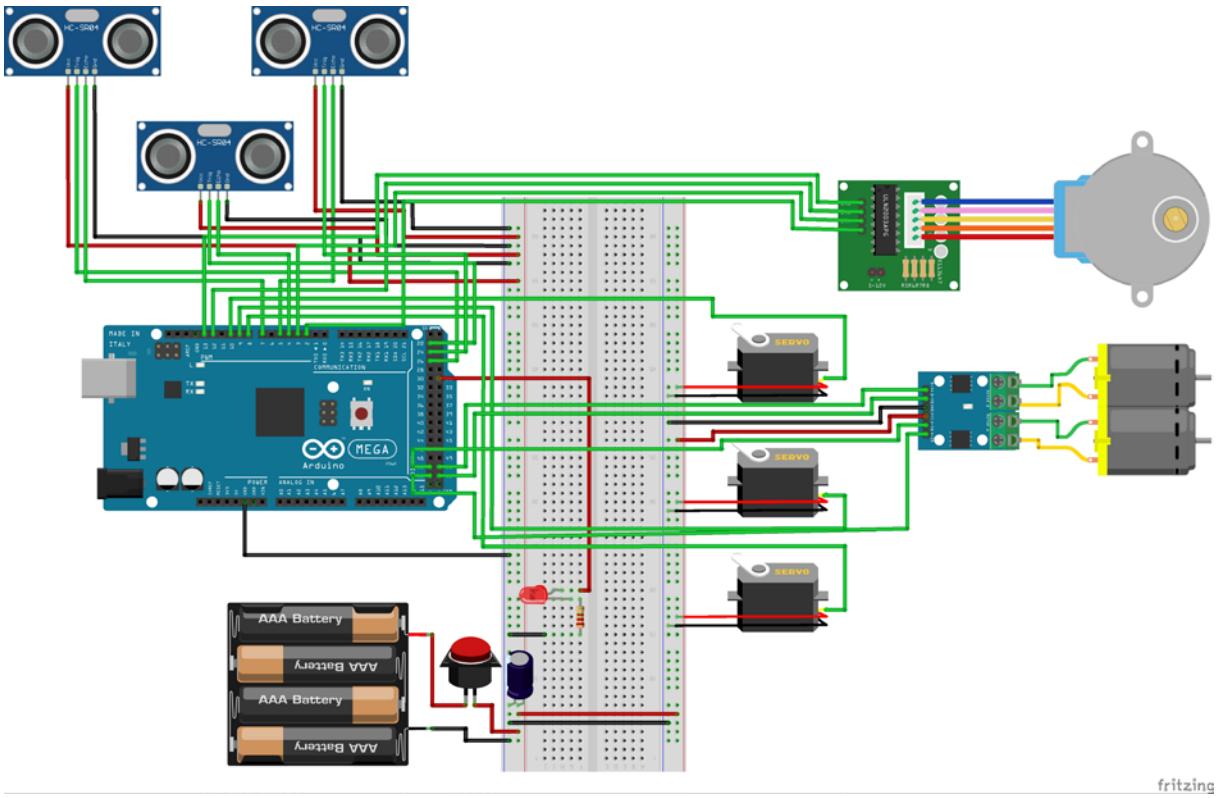
O Arduino será alimentado pela mesma alimentação externa, mas na saída de 12V com corrente podendo chegar à 10A.

Os servos foram ligados nas portas digitais PWM do Arduino 8, 9, 10 e 11, os sensores de distância estão nas portas PWM 4, 5 e 7 e também nas portas digitais 22, 24 e 26. Nas portas digitas estão os disparadores (gatilhos) e configuradas como saídas. As portas PWM são as receptoras e estão configuradas como portas de entrada. Os sensores também utilizam a corrente fornecida pela fonte externa de 5V.

O motor de passo necessita quatro portas para o gerenciamento e controle de sua rotação e precisão de movimentos angulares. Para este controle foram utilizadas as portas PWM 2, 3, 12 e 13 conectadas ao driver ULN2003 e deste controlador, conectados ao motor.

Na figura 13 se pode ver a montagem completa do sistema eletrônico do projeto, estando com todos os componentes. A fonte de alimentação é representativa pois o programa Fritzing não possui suporte para o tipo de fonte usado no projeto real.

Figura 13 - Sistema eletrônico do projeto



Fonte: O Autor, 2019.

Pode-se observar na figura 13 que a ligação dos motores DC estão em portas digitais do Arduino: 50, 52 e 51, 53 atuando com auxílio de uma ponte H dupla. Apesar

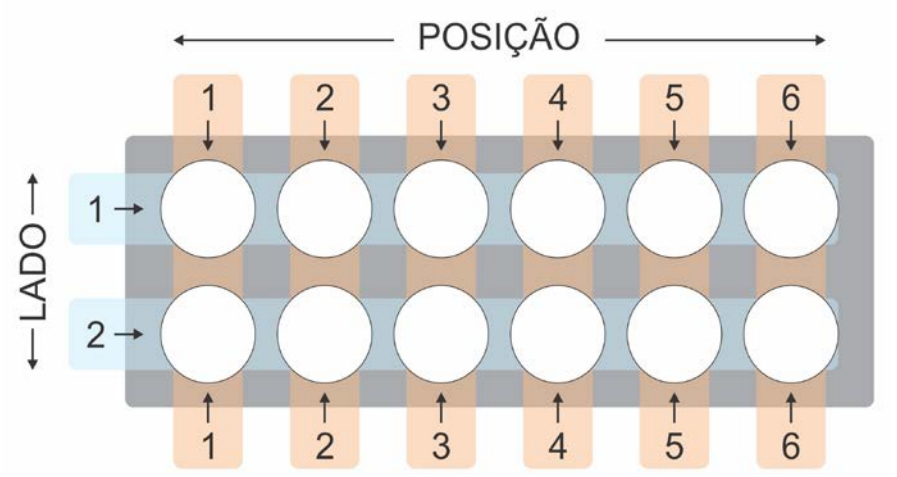
de estarem separadas estão com seu funcionamento síncrono, os dois motores trabalhando juntos para deslocar o carro para a mesma direção: esquerda ou direita.

A fixação dos servos motores está sendo feita por meio de parafusos e suportes em MDF. Os sensores fixados com fita adesiva e cola de alta aderência. A fiação e outros componentes fizeram-se uso de solda com estanho para evitar mau contato e assim diminuir riscos dos componentes se desconectarem quando o braço e o carro estiverem em movimento.

3.3 CÓDIGO DE PROGRAMAÇÃO

O código gerado para localização do objeto necessita de apenas um sensor HC-SR04. O princípio de funcionamento é bastante simples: quando um objeto estiver a uma distância previamente estabelecida ele faz com que os motores DCs parem de funcionar e que o restante do código passe a ser executado, uma linha de cada vez. Quando finalizar todas as linhas de comando é novamente ativado o sensor de localização até que o sensor reconheça os 12 objetos, conforme figura 14, a serem recolhidos, também é feita a liberação dos motores DCs para continuarem girando para o mesmo lado que estavam no momento da parada.

Figura 14 - Posições de armazenamento em uma caixa de ovos



Fonte: O Autor, 2019.

Conforme visto na figura 14, este número de recolhimentos está relacionado ao número de posições que temos em uma caixa de armazenamento de ovos.

Os dois sensores que estão programados para reconhecer os limites de deslocamento lateral, funcionam da mesma forma. Quando encontram a final do

deslocamento acionam o sentido contrário da corrente que está sendo enviada aos motores DC. Desta forma, o carro de deslocamento estará sempre em movimento, para um ou outro lado até que o sensor do objeto reconheça o próximo objeto a ser recolhido.

No momento em que o sensor localiza um objeto a ser recolhido, o corte da corrente para os motores DCs é imediato, mas a parada do carro de deslocamento lateral não. Isto se dá devido a inércia do movimento que é essencial para que o deslocamento continue por mais 2 a 2,5cm. Este deslocamento faz com que o objeto fique centralizado embaixo da garra, não sendo necessário ajustes extras na programação para que tenha um *delay* de parada.

Após o reconhecimento do objeto, há um *delay* para o início do movimento do braço robótico, como comentado anteriormente, este espaço de tempo é necessário para que se tenha a parada completa do carro de deslocamento lateral.

Abaixo descreve-se o pseudocódigo, que conforme Carvalho (2017):

> Para facilitar a comunicação entre as pessoas e os computadores, os programas são inicialmente escritos em uma linguagem intermediária, mais próxima de uma linguagem natural, como o português, mas já seguindo o formato adotado pela maioria das linguagens de programação. A sequência de passos escrita nessa linguagem é denominada pseudocódigo ou algoritmo. Um algoritmo torna mais fácil ao ser humano conferir se a sequência de passos especificada é capaz de resolver o problema. Conferido o algoritmo, ele pode ser traduzido para uma linguagem de programação, formando assim um programa (CARVALHO, 2017).

Nestes termos, usa-se então a lógica que descreve o deslocamento do carro, a detecção do objeto e o que será feito após encontrar o objeto. Assim, os sensores de deslocamento lateral estando sempre ativos e enquanto não reconhecem o final da trajetória continuam sempre na mesma direção de rotação. Após o encontro do final da trajetória é acionado a corrente inversa ao sentido de rotação daquele momento. Com isso, o carro faz o deslocamento para o sentido oposto até encontrar o final da trajetória no outro sentido, continuadamente até algum objeto ser reconhecido para ser recolhido.

Para facilitar o entendimento de todo processo gerado pelo código e lembrando que o armazenamento será feito em caixas de ovos com 12 espaços, sendo 2 lados de 6 posições conforme mostrado na figura 14, pode-se identificar cada passo do código na forma representativa abaixo mostrada:

```
Início do Loop
     Deslocamento lateral (direita ou esquerda)
```

```
Se o lado da caixa for menor que 3
     Se reconhece objeto entre 12 a 15cm de distância
          Para o deslocamento lateral
          Baixa o braço
          Fecha a garra
          Sobe o braço
          Gira o braço para a direita
          Acertar a posição na caixa
          Baixa o braço
          Abre a garra
          Sobe o braço
          Gira o braço para a esquerda
          Retorna o braço para a posição de leitura
          Aumenta 1 na contagem de posição da caixa
     Se a posição na caixa for igual a 6
          Aumenta 1 no lado da caixa
          Contagem de posição da caixa retorna para 1
     Se lado da caixa for igual a 2
          Contagem do lado da caixa retorna para 1
          Aumenta 1 na contagem de caixas cheias
          Acende o LED representando a parada do
          sistema para a troca da caixa de coleta.
Fim do loop
Retorna ao início para novo loop
```

Além da descrição acima feita por pseudocódigo, disponibiliza-se a versão final do código completo gerado para este braço robótico, o mesmo encontra-se junto ao APÊNDICE A ao final deste trabalho.

Como se pode ver, os quatro condicionais apresentados no código dizem respeito ao comportamento dos movimentos perante a detecção do objeto. O controle do movimento do braço robótico se baseia na próxima posição livre da caixa de armazenamento dos objetos. O local onde será recolhido será sempre o mesmo perante a posição do carro de deslocamento, mas será guardado cada vez em uma posição diferente da anterior até satisfazer todas as condições de posição sem uso.

A cada iteração é testado se o objeto está dentro de uma faixa pré-estabelecida, assim escrito no código:

```
79      if (distanciaovo > 15 && distanciaovo < 18) {
```

Por esta análise, o objeto terá que estar entre uma distância máxima e uma mínima (estabelecida empiricamente), se não estiver nestas condições o carro segue no seu deslocamento lateral.

4 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Como visto nos textos anteriores, houve uma apresentação inicial de cada parte deste projeto. Desta forma, cria-se uma facilidade a mais no entendimento com relação ao conteúdo completo do trabalho. A partir deste ponto, a apresentação e a análise dos resultados estão focadas na parte final da montagem do braço robótico e sua funcionalidade. Mostrando com maior detalhes os eventuais problemas encontrados e possibilitando o leitor seguir uma linha contínua de raciocínio. Baseando-se nisso, houve necessidade desta etapa ser separada em: montagem da parte mecânica, montagem da parte eletrônica e programação do controlador. Assim, cada etapa da construção e montagem do braço robótico pode ser descrita, apresentada e analisada com maior quantidade de informações como pode ser visto nos itens abaixo descritos.

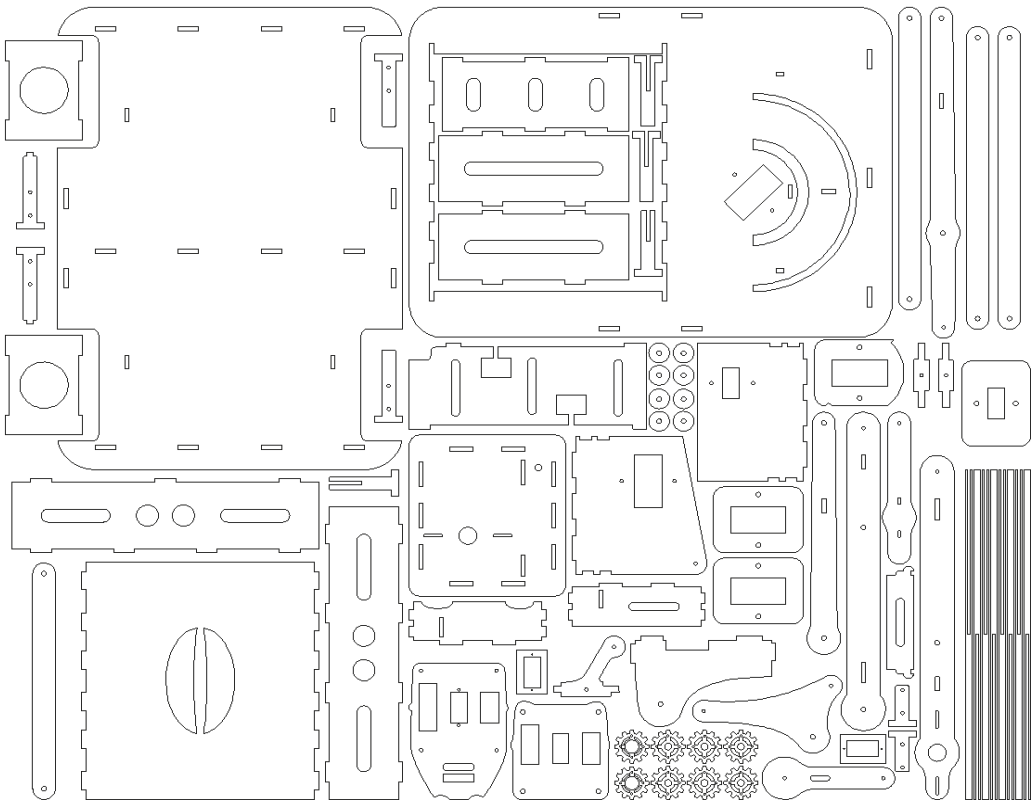
4.1 MONTAGEM MECÂNICA

A construção da parte mecânica, feita em MDF 3mm, teve que ser feita por 2 vezes, sendo que na primeira tentativa de construção houve pequenas folgas que atrapalhavam o desempenho do movimento. Após esta análise houve, também, a necessidade de ter um alcance maior em relação ao objeto para ser recolhido. Desta forma foram aumentadas algumas medidas e feitos alguns ajustes nas furações. Com isso o braço consegue trabalhar com menor quantidade de folgas e também com baixa quantidade de atritos entre as partes em MDF conectadas uma as outras, gerando assim maior rigidez sem afetar a mobilidade.

Após a segunda tentativa de construção conseguiu-se gerar todas as funções de movimento de cada junta sendo aplicado pelos motores utilizados neste projeto, todos realizando o movimento como esperado.

A montagem da estrutura física, com MDF 3mm, escolhido por ter o custo relativamente menor e que concebia grande rigidez quando bem estruturado, na figura 15 é ilustrada a montagem das peças que foi enviado para corte em máquina laser. Nem todas as peças foram envolvidas na estrutura do braço robótico e no carro de deslocamento lateral. Algumas foram cortadas para testes e talvez possível uso para aumentar a rigidez das peças longas, uso que acabou não sendo necessário.

Figura 15 - Peças da estrutura montada para corte a laser

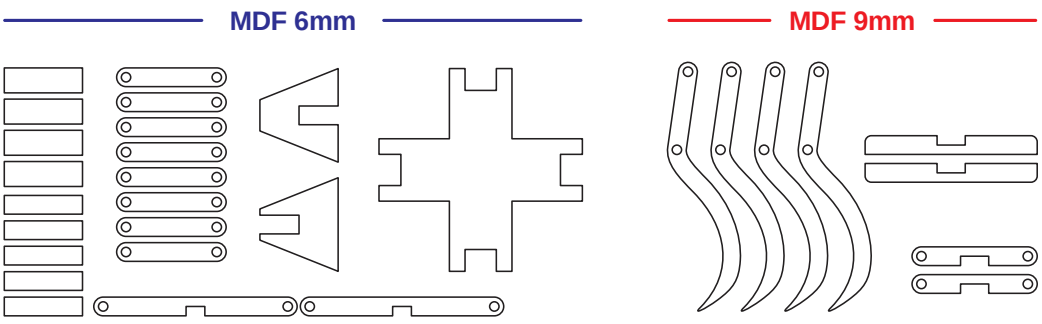


Fonte: O Autor (Adaptado de Pirt, 2017).

Pode ser visto que é uma estrutura com muitas peças, mas de fácil montagem, houve a intenção de ocupar o espaço da melhor forma possível evitando gastos desnecessários e desperdício de material.

Na figura 16 são apresentadas as peças necessárias para a construção do conjunto da garra e seu suporte. Houve necessidade de estudo em separado somente para este componente. Como havia necessidade de não danificar o objeto e a construção ser feita sem folgas ou espaços em que o objeto possa cair ou se soltar da garra durante o transporte até o armazenamento.

Figura 16 - Peças em MDF necessárias para a garra



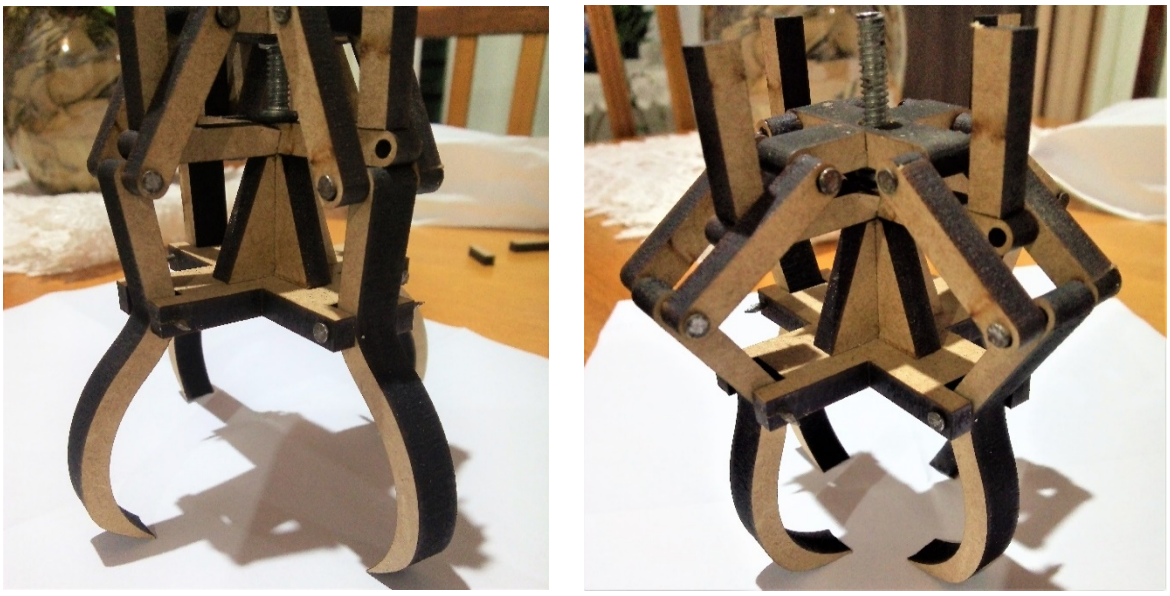
Fonte: O Autor, 2019.

Diferente do restante da estrutura, a garra foi desenvolvida com peças em MDF de 6 e 9mm de espessura. Ganhando rigidez e fornecendo maior segurança para o transporte do objeto até a caixa de armazenamento.

Após frustradas tentativas na intenção de uso do motor DC e após com servo motor houve a necessidade da utilização de um motor de passo para a movimentação dos dedos da garra. Com o motor DC não se conseguiu o movimento correto no controle do fechamento da garra. Com o servo motor, limitado ao giro de 180º, não fornecia movimento suficiente para a abertura de fechamento dos dedos da garra. Etapa solucionada com o uso de um motor de passo podendo ter o controle absoluto sobre a velocidade de giro e posição exata para que os dedos da garra não prejudicassem o objeto quando este for recolhido.

Na figura 17 pode-se ver o conjunto dos componentes da garra quando fechada e também como se apresenta aberta. As medidas de abertura e fechamento dos “dedos” da garra foram baseadas em ovos comuns produzidas em propriedade rural parceira. O tamanho médio dos ovos são de 5,5cm de comprimento e 4,3cm em sua altura. O tamanho máximo dos ovos encontrados é de 6 cm de comprimento e 4,7cm de altura, o tamanho mínimo ficou em 4,8cm de comprimento e 3,7 de altura. A garra, quando fechada, ainda terá uma abertura de 3cm, e quando aberta chegará a 7cm, nestas condições consegue atender todas as opções de ovos produzidas na propriedade.

Figura 17 - Conjunto com dedos da garra abertos e fechados.



Fonte: O Autor, 2019.

O conjunto das peças e o desenho dos dedos da garra foram pensados com o intuito de não danificar o objeto, desde o momento do recolhimento até o depósito em lugar proposto. Evitando danificar também possíveis objetos que estejam nas proximidades do objeto a ser coletado. Com seu formato interno circular é possível coletar o objeto em qualquer posição que esteja depositado na calha.

Pode-se observar na figura 18, com o ovo já coletado, que há espaços entre o ovo e os dedos da garra, mas, devido ao formato das peças, o objeto não se solta do interior da garra.

Figura 18 - Ovo coletado pela garra

Fonte: O Autor (2019).

O desenho dos dedos e da garra é um fator importante para que se possa coletar os ovos em seus vários tamanhos e em posicionamentos diversos. O ovo não é pressionado por nenhuma das partes da garra, mas sim, devidamente apoiado com o auxílio da gravidade sobre o mesmo.

4.2 MONTAGEM ELETRÔNICA

A montagem da parte eletrônica foi elaborada em etapas diversas, onde após cada componente ser inserido no sistema houve a necessidade de testes de funcionamento. A maior parte dos componentes já tinham sido usados por outros

acadêmicos em projetos diversos e não havia conhecimento se todos encontravam-se em funcionamento, gerando tempo em excesso para a montagem. Na figura 19, pode ser observado quando parte dos componentes estavam sendo montados para teste de funcionamento. Houve necessidade de troca de um micro servo SG90 e um servo motor MG995. Além disso, foi necessária a troca de um sensor HC-SR04 e um dos motores DC por ter os conectores danificados.

Figura 19 - Montagem parcial para teste dos componentes

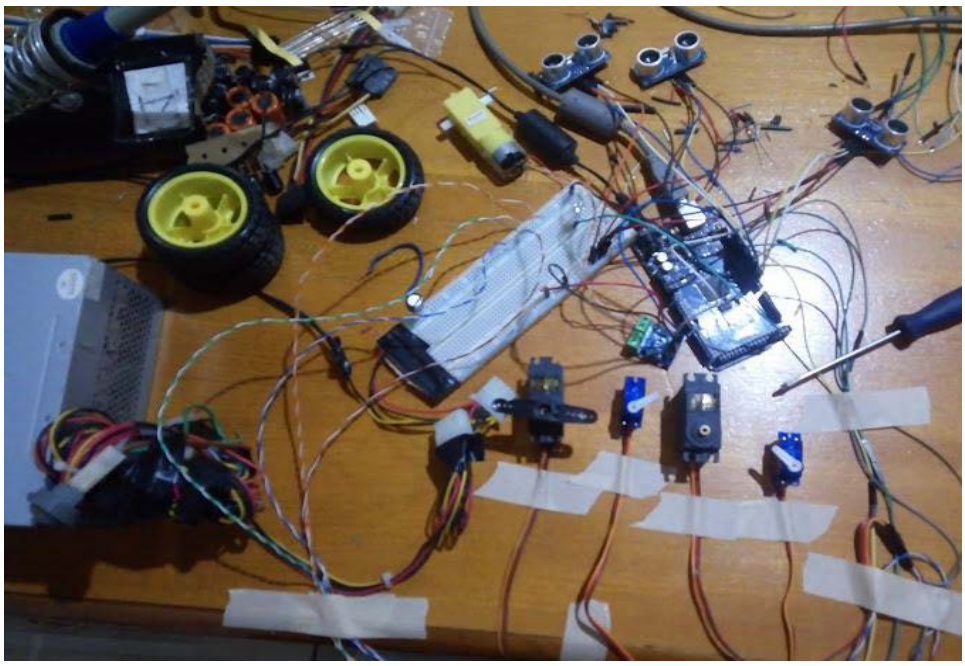
Fonte: O Autor, 2019.

Nos testes realizados para o controle dos dedos da garra, houve tentativa de uso de um motor de passo retirado de um aparelho leitor de DVD de computador. Nestes testes, houve a queima de um drive ponte H duplo L9110. Após inspeção visual foi percebido que o motor havia se danificado e consumiu corrente acima do suportado pelo drive L9110 consequentemente ocasionando sua avaria.

Após a análise inicial e troca dos componentes com defeito, pode ser montado o conjunto completo e, desta forma, feitos testes e pequenos ajustes em relação ao comprimento da fiação usada. Houve uma tentativa de padronização das cores da fiação para melhorar o entendimento de cada conexão feita.

Na figura 20 já é possível observar o sistema eletrônico completo, com todos os componentes montados e em funcionamento.

Figura 20 - Montagem de todos componentes para testes



Fonte: O Autor, 2019.

Pode-se observar que há muitos cabos que podem ser eliminados caso não usar a *protoboard* para a montagem do sistema, exemplo: ligando diretamente ao cabo de energia, tanto positivo quanto negativo, unificando toda a fiação que sai da fonte de alimentação. Com o uso da *protoboard* há ganho na facilidade e agilidade na troca de fiação e possibilidade de montagem e desmontagem de todo ou em parte do sistema podendo realizar a análise com melhor eficácia e com menor tempo necessário.

4.3 PROGRAMAÇÃO

Com base nos resultados obtidos na construção das funções para o movimento dos motores utilizados neste projeto, conseguiu-se executar de forma precisa e realizados de acordo com o esperado. Todas as funções projetadas funcionaram como planejado, conseguindo posicionar o braço robótico para realizar a coleta e posterior depósito nas posições estipuladas na caixa de armazenamento.

Entretanto é válido acrescentar que em todas as funções de movimentação dos motores houve a necessidade de acrescer um tempo de atraso para o início da próxima execução. Como este atraso precisava ser do tipo bloqueante, foi usado a

função *delay* em todos os pontos que foi julgado necessário. Após o uso desta parada no sistema, cada movimento está sendo executado com precisão, tornando assim, necessário o uso do *delay*.

A maior dificuldade encontrada nesta parte do projeto está relacionada ao posicionamento do braço robótico e também sobre o objeto que tem necessidade de ser localizado. Há necessidade de correção do programa de reconhecimento do objeto e correção dos ângulos dos servos motores a cada mudança de estrutura para o deslocamento do carro. Neste sentido, se faz necessário a construção precisa e alinhada de um sistema de trilhos que sirva de base sólida, sem solavancos e sem desníveis para que o carro possa se deslocar de um lado para outro sem possíveis travamento das rodas.

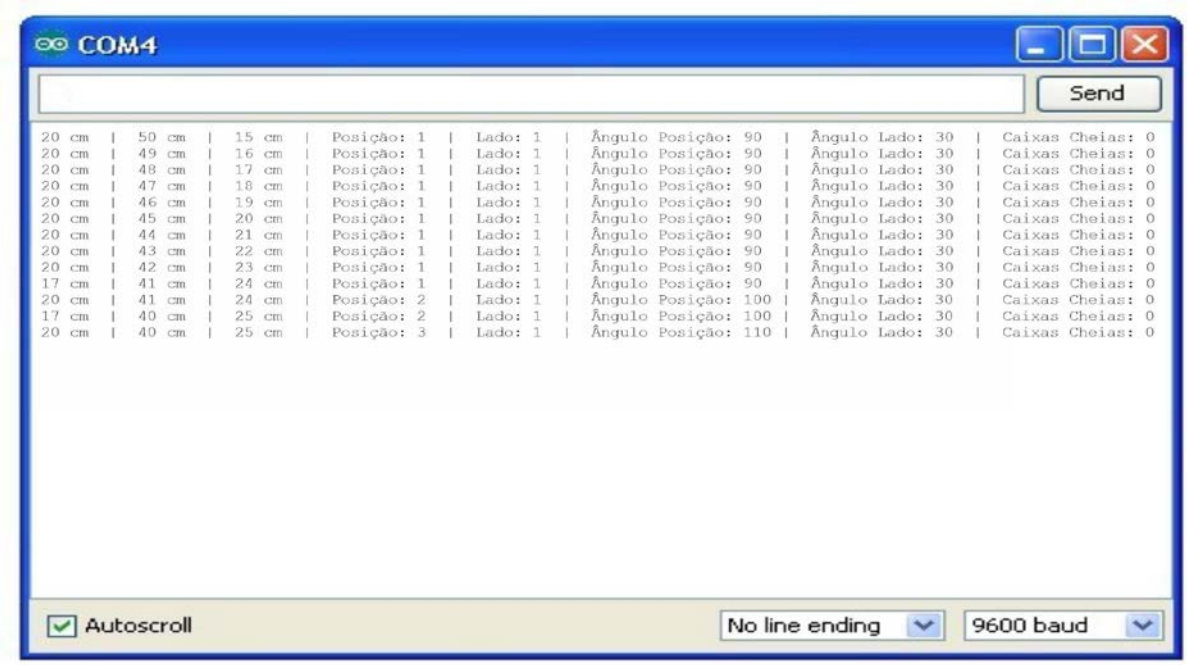
Outra dificuldade foi estipular o ângulo inicial de cada servo motor. Tendo necessidade de ser realizados testes com todo o protótipo montado, mas sem conectar o eixo dos servos motores em sua devida haste de movimentação. Após a verificação visual aproximada do ângulo de cada servo motor foi realizada a fixação de todos os componentes na estrutura, realizando o ajuste fino de cada ângulo e tornando assim o código totalmente eficaz.

CONCLUSÃO

Com base no propósito deste trabalho foi verificado que se conseguiu concluir grande parte da proposta inicial do projeto. A construção das partes mecânicas do braço robótico e o funcionamento correto de todos os movimentos realizados com todos os componentes integrados ao projeto. No entanto, não houve possibilidade de colocar em funcionamento em local próprio para a coleta dos ovos. O produtor que aceitou realizar os testes práticos entrou em contato informando problemas externos e resolveu adiar a implantação do sistema. Nestas condições não foi possível determinar um padrão estável no recolhimento dos ovos e testar a efetiva melhoria para que o produtor pudesse se ausentar por um período prolongado durante o dia. Foi possível a realização de todos os testes do algoritmo simulando o recolhimento de ovos em bancada de testes.

Para a simulação foi desenvolvida uma função a mais e associada ao código. O uso desta função é auxiliar possibilitando visualizar os resultados em tempo real através do monitor serial da própria IDE do Arduino (figura 21). Assim pode ser realizada a análise mesmo não estando com o protótipo em seu lugar ideal.

Figura 21 - Monitor serial para acompanhamento em tempo real



Fonte: O Autor, 2019.

Na figura 21, é apresentado na sequência, as leituras da distância do objeto (ovo), distância da margem direita, distância da margem esquerda, posição da

próxima posição livre da caixa, qual lado da caixa está em uso, qual o próximo ângulo de movimentação lateral para a posição da caixa, qual o ângulo de movimentação para o lado da caixa e por último quantas caixas já foram completadas.

Também não foi construído o espaço necessário para o deslocamento lateral do carro, esta etapa também estava projetada para ser feita diretamente na propriedade do produtor, já anexada à gaiola das galinhas poedeiras.

Mesmo apresentando os problemas acima é possível considerar que foi obtido grande êxito no projeto como um todo. Foi realizada a montagem do braço robótico sobre uma plataforma móvel, que se apresentou capaz de realizar os movimentos de acordo com o programado e estipulado. A instalação em unidades produtoras diversas pode ser feita somente com pequenos ajustes na programação e com a necessidade da instalação de uma base firme e alinhada para a movimentação livre do carro que fará o deslocamento lateral da plataforma.

Para trabalhos futuros é possível a configuração de maiores quantidades de ovos a serem recolhidos. Esta alteração é de fácil implementação sendo necessária somente mudança na programação do sistema para que cada posição seja devidamente acertada. Há também necessidade de aumentar o tamanho do carro de deslocamento para que a nova caixa de armazenamento tenha uma base de sustentação firme e segura.

Outro acréscimo que pode ser feito ao adicionar novas funções é uma função que substitua as caixas de ovos preenchidas por outra vazia, de forma autônoma. Neste caso é possível que haja a necessidade de novo desenho do carro de movimentação e, possivelmente, a utilização de outro controlador e talvez um segundo braço robótico para a troca ou reposição da caixa de armazenamento.

REFERÊNCIAS

AGNIHOTRI, Nikhil. **Stepper Motor: Basics, Types and Working**. 2011. Disponível em: <https://www.engineersgarage.com/articles/stepper-motors?page=1>. Acesso em 09 jun. 2019.

ALVES, William Pereira. **Linguagem e Lógica de Programação**. 1ª ed. São Paulo. Érica. 2014.

ARAÚJO, W.A.G. & ALBINO, L.F.T. **Incubação Comercial**. 1ª. ed. Viçosa - MG: Transworld Research Network, 2011. 171p

ARDUINO. Disponível em <https://store.arduino.cc/usa/mega-2560-r3>. Acesso em 18 maio 2019. 2015.

BERNARDO, Jader. **6 alternativas de como controlar motores com a Ponte H**. 6 out.2016. Disponível em: <http://eletronworld.com.br/eletronica/6-alternativas-de-como-controlar-motores-com-a-ponte-h/#ponte-h-com-shield>. Acesso em 20 maio 2019.

BERTECHINI, A. G. **Mitos e verdades sobre o ovo de consumo**. Conferência APINCO. Campinas – SP, p. 19, 2003.

BRAGA, C. Newton. **Motores DC e caixas de redução**. 2014. Disponível em: <http://www.newtoncbraga.com.br/index.php/robotica/5168-mec070a>. Acesso em 20 maio 2019.

CARVALHO, André C. P. L. F. de. LORENA, Ana Carolina. **Introdução à computação: hardware, software e dados**. 1ª ed. Rio de Janeiro. LTC. 2017.

CUNHA, Alessandro. **Sistemas Embarcados**. Revista Saber Eletrônica - 414. Editora: Saber, BRASIL, 2007.

CUTNELL, João D., JOHNSON, Kenneth W. **Física**. 9ª ed. Rio de Janeiro. LTC. 2016.

EMBRAPA Suínos e Aves - Central de Inteligência de Aves e Suínos. **Preços**. 2019. Disponível em: <https://www.embrapa.br/suinos-e-aves/cias/precos>. Acesso em 19 maio 2019.

EGESTOR. **Terceirização: O que é, suas vantagens e desvantagens**. 02 jan. 2017. Disponível em: <https://blog.egestor.com.br/terceirizacao-o-que-e-suas-vantagens-e-desvantagens/>. Acesso em: 11 jun. 2019.

FERREIRA, Aurélio B. de H. **Dicionário Aurélio**. 5ª ed. Curitiba: Editora Positivo, 2014.

FILHO, P. P. R. **Microcontroladores PIC - Linguagem C utilizando CCS para leigos**. Ceará. IFCE. 2014.

GIL, Antonio Carlos. **Como Elaborar Projetos de Pesquisa**. 6ª ed. Atlas. 07/2017.

GOEKING, Weruska. **Lâmpadas e Leds**. 46ª ed. Nov. 2009. Disponível em: <https://www.osetoreletrico.com.br/lampadas-e-leds/>. Acesso em: 22 maio 2019

HC-SR04 - Manual do Usuário. Disponível em: <https://elecfreaks.com/estore/download/EF03085-HC-SR04_Ultrasonic_Module_User_Guide.pdf>. Acesso em 20 maio 2019.

IBGE - INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. **Produção de Ovos de Galinha** - 4º trimestre 2018. Rio de Janeiro: IBGE, 2019. Disponível em: <https://sidra.ibge.gov.br/home/pog/brasil>. Acesso em: 10 maio 2019.

ISO 8373:2012. **Robots and robotic devices - Vocabulary**. Disponível em: <https://www.iso.org/obp/ui/es/#iso:std:iso:8373:ed-2:v1:en>. Acesso em 18 maio 2019.

JUNIOR, Lorival, F. **Vamos aprender a utilizar Sensor URM37 V 32 para medir distâncias entre objetos.** 8 jun. 2017. Disponível em: <https://loriroboticaeducacional.blogspot.com/2017/06/vamos-aprender-utilizar-sensor-urm37-v.html>. Acesso em: 03 abr 2019.

JUNIOR, Sergio Luiz Stevan. SILVA, Rodrigo Adamshuk. **Automação e instrumentação industrial com Arduino: teoria e projetos**. 1ª ed. São Paulo. Érica. 2015.

MANZANO, José Augusto N. G. **Linguagem C: acompanhada de uma xícara de café**. 1ª ed. São Paulo. Érica. 2015.

Micro Servo Motor SG90. Disponível em: <https://www.vidadesilicio.com.br/micro-servo-motor-sg90>. Acesso em: 22 maio 2019

MONK, Simon; tradução: LASCHUK, Anatólio. **Programação com Arduino: começando com sketches**. 2ª ed. Porto Alegre. Bookman. 2017.

MOTA, Allan. **HC-SR04 – Sensor Ultrassônico de distância com Arduino**. 2018. Disponível em: <https://portal.vidadesilicio.com.br/hc-sr04-sensor-ultrassonico/>. Acesso em: 22 maio 2019

NETTO, Daniel A. **Avaliação da produção de galinhas poedeiras criadas sob condições de clima quente em piso e em gaiola**. Disponível em: <https://www.ufmt.br/zoocba/arquivos/126421cc8df291c5e2c8b6176b40503f.pdf>. Acesso em: 10 maio 2019.

OLIVEIRA, André S. de, ANDRADE, Fernando S. de. **Sistemas Embarcados: Hardware e Firmware na Prática**. 2ª ed. São Paulo. Érica. 2010. 6ª tiragem. 2016.

OLIVEIRA, Cláudio V., ZANETTI, Humberto P. **Arduino Descomplicado - Como Elaborar Projetos de Eletrônica**. 1ª ed. São Paulo. Érica. 2015.

PEREIRA, F. **PIC programação em C**. São Paulo. Érica. 2003.

PETRUZELLA, Frank D. **Motores elétricos e acionamentos**. Porto Alegre. AMGH. 2013.

PIRT, Ben. **Laser cut files for MeArm**. 2017. Disponível em: <https://github.com/mimeindustries/MeArm>. Acesso em 26 mar 2019.

PRIBERIAM, **Dicionário Priberam da Língua Portuguesa**. 2008-2013. Disponível em: <https://dicionario.priberam.org/. Acesso em: 18 maio 2019.

SAKOMURA, Nilva Kazue. 2014. **Manejo e Processamento dos Ovos - Aula 9**. Disponível em: <http://javali.fcav.unesp.br/Home/departamentos/zootecnia/NILVAKAZUESAKOMURA/aula_9_proc_ovos_muda_descarte_e_custo.pdf>. Acesso em 09 jun 2019.

SANTOS dos, W.JR., G. e Chaves, J. **Robótica Industrial - Fundamentos, Tecnologias, Programação e Simulação**. 1ª ed. São Paulo. Érica. 2015.

SEBRAE, Nacional. **Terceirização da mão de obra**. 08 maio 2017. Disponível em: <http://www.sebrae.com.br/sites/PortalSebrae/bis/terceirizacao-da-mao-de-obra,345baf08868eb510VgnVCM1000004c00210aRCRD>. Acesso em: 11 jun. 2019.

SERAFINI, Suélen. Soares, J. G.. da Silva, K. C. C.. Manenteboiago, M.. **Produção, Estrutura e Processamento de Ovos.** 149 ed. ano 7 - 12 mar 2015. Disponível em: <https://www.udesc.br/arquivos/ceo/id_cpmenu/1043/caderno_udesc_149_15198236923071_1043.pdf>. Acesso em: 10 maio 2019.

SILVEIRA, Cristiano Bertulucci. **Servo Motor: Veja como Funciona e Quais os Tipos**. 2017. Disponível em: <https://www.citisystems.com.br/servo-motor/>. Acesso em: 22 maio 2019

SOUZA, Fábio. **Arduino MEGA 2560**. 28 abr. 2014. Disponível em: <https://www.embarcados.com.br/arduino-mega-2560/>. Acesso em 15 abr 2019.

SOUZA, Fábio. **Introdução ao Arduino - Primeiros passos na plataforma**. 06/11/2013. Disponível em: <https://www.embarcados.com.br/arduino-primeiros-passos/>. Acesso em: 18 maio 2019.

UsinaInfo - Disponível em: <https://www.usinainfo.com.br/4-nossa-empresa>. Acesso em: 20 maio 2019.

APÊNDICE A

Código completo gerado para este braço robótico.

```
#include <Servo.h>
#include <Ultrasonic.h>
#include <Stepper.h>

Stepper motor(2048,2,3,12,13);

#define SINAL_ovo 22 // Pino do gatilho PWM
#define PWM_ovo 4 // Saída PWM 0-25000US, cada 50US representam um
centímetro
#define SINAL_dir 24
#define PWM_dir 5
#define SINAL_esq 26
#define PWM_esq 7

int EnPwmCmd[4] = {0x44, 0x02, 0xbb, 0x01}; // Comando de medida de
distância por PWM
int pos = 90;
int posD = 45;
int posLado = 0;
int vai = 0;
int Angulo_BaixaSobe = 75; //baixa e sobe
int Angulo_AbreFecha = 90; //abre e fecha
int Angulo_DireitraEsquerda = 110; //direita e esquerda
int Angulo_AcertaPosicao = 30; //acerta posição na caixa
int posicaocaixa = 1;
int ladocaixa = 1;
int quantcaixas = 0;
int const ledVermelho = 31;
const int Rodas_A1A = 50;
const int Rodas_A1B = 52;
const int Rodas_B1A = 51;
const int Rodas_B2A = 53;

int distanciaovo; // Variável que armazenará o valor para impressão
int distanciadir;
int distanciaesq;
byte speed = 255;  // Mude este valor (0-255) para controlar a velocidade
dos motores

Servo Servo_BaixaSobe; // baixa e sobe, pino 8
Servo Servo_AbreFecha; // abre e fecha, pino 9
Servo Servo_DireitaEsquerda; // direita e esquerda, pino 10
Servo Servo_AcertaPosicao; // acerta posição na caixa, pino 11

void setup() {
  Serial.begin(9600); // Inicia a configuração serial com o monitor em uma
taxa de 9600 bits
  Serial.println("Lendo dados dos sensores...");
  pinMode(SINAL_ovo, OUTPUT); // Define o pino COMP / SINAL como saída
  pinMode(SINAL_dir, OUTPUT);
  pinMode(SINAL_esq, OUTPUT);
  pinMode(ledVermelho, OUTPUT);
  digitalWrite(SINAL_ovo, LOW); // Ativa o Pino COMP/SINAL
  digitalWrite(SINAL_dir, LOW);
  digitalWrite(SINAL_esq, LOW);
  pinMode(PWM_ovo, INPUT); // Define o pino PWM como entrada
  pinMode(PWM_dir, INPUT);
```

```
  pinMode(PWM_esq, INPUT);
  pinMode(Rodas_A1A, OUTPUT); // Colocando os pinos como saída
  pinMode(Rodas_A1B, OUTPUT);
  pinMode(Rodas_B1A, OUTPUT); // Colocando os pinos como saída
  pinMode(Rodas_B2A, OUTPUT);
  motor.setSpeed(100);

  // Escreve os dados dentro da EEPROM do URM37
  for (int i = 0; i < 4; i++) {
    Serial.write(EnPwmCmd[i]);
    Servo_BaixaSobe.attach(8);
    Servo_AbreFecha.attach(9);
    Servo_DireitaEsquerda.attach(10);
    Servo_AcertaPosicao.attach(11);
  }
  //Inicia Servos na posição
  Servo_BaixaSobe.write(0); //baixa e sobe
  Servo_AbreFecha.write(90); //abre e fecha
  Servo_DireitaEsquerda.write(110); //direita e esquerda
  Servo_AcertaPosicao.write(0); //acerta posição na caixa
}

//Inicia o loop
void loop() {
  lado();
  anda(vai);
  distancia_obj();
  distancia_dir();
  distancia_esq();
  mostrar();

  if (ladocaixa < 3) {
    if (distanciaovo > 3 && distanciaovo < 20) {
      para();
      baixa();
      fechar();
      tempo2();
      sobe();
      direita();
      acertaposicao();
      baixa();
      abrir();
      tempo2();
      sobe();
      zeraposicao();
      esquerda();
      posD -= 5;
      posicaocaixa += 1;
    }

    if (posicaocaixa == 7) {
      posD = 45;
      posLado += 20;
      ladocaixa += 1;
      posicaocaixa = 1;
    }
}
  if (ladocaixa == 3) {
    posD = 45;
    posLado = 0;
    ladocaixa = 1;
```

```
    quantcaixas += 1;
    led();
}
  tempo2();
} //FIM DO VOID LOOP.

void direita() {
  for (pos = Angulo_DireitraEsquerda; pos >= posD; pos -= 1) {
    Servo_DireitaEsquerda.write(pos);
    delay(10);
  }
}

void esquerda() {
  for (pos = posD; pos <= Angulo_DireitraEsquerda; pos += 1) {
    Servo_DireitaEsquerda.write(pos);
    delay(10);
  }
  tempo();
}

void sobe() {
  for (pos = Angulo_BaixaSobe; pos >= 0; pos -= 1) {
    Servo_BaixaSobe.write(pos);
    delay(20);
  }
  tempo();
}

void baixa() {
  for (pos = 0; pos <= Angulo_BaixaSobe; pos += 1) {
    Servo_BaixaSobe.write(pos);
    delay(20);
  }
  tempo();
}

void fechar(){
  motor.step(512); //8 voltas
  delay(10);
}

void abrir(){
  motor.step(-512); //8 voltas
  delay(10);
}

void acertaposicao() {
  for (pos = posLado; pos <= Angulo_AcertaPosicao; pos += 1) {
    Servo_AcertaPosicao.write(pos);
    delay(10);
  }
  tempo();
}

void zeraposicao() {
  for (pos = Angulo_AcertaPosicao; pos >= posLado; pos -= 1) {
    Servo_AcertaPosicao.write(pos);
    delay(10);
  }
}
```

```
void anda(int vai) {
  if (vai == 1) {
    vai_esquerda();
  }
  else {
    vai_direita();
  }
}

void lado() {
  if (distanciaesq <= 10 && distanciaesq >= 2) {
    para();
    vai = 1;
  }
  if (distanciadir <= 10 && distanciadir >= 2) {
    para();
    vai = 0;
  }
}

void para() {
  analogWrite(Rodas_A1A, 0);
  analogWrite(Rodas_A1B, 0);
  analogWrite(Rodas_B1A, 0);
  analogWrite(Rodas_B2A, 0);
  tempo2();
}

void distancia_obj() {
  digitalWrite(SINAL_ovo, HIGH); // Desliga o pino COMP/SINAL
  delayMicroseconds(11);
  digitalWrite(SINAL_ovo, LOW); // leitura - Pin PWM pulsos de saída
  distanciaovo = pulseIn(PWM_ovo, HIGH); // Leitura do valor de pulso de
entrada do PWM
  distanciaovo = distanciaovo / 2; // Define a medate da distância de
leitura
  distanciaovo = distanciaovo * 0.0343; // Converte a distância para a
distância real através da velocidade do som
}

void distancia_dir() {
  digitalWrite(SINAL_dir, HIGH); // Desliga o pino COMP/SINAL
  delayMicroseconds(11);
  digitalWrite(SINAL_dir, LOW); // leitura - Pin PWM pulsos de saída
  distanciadir = pulseIn(PWM_dir, HIGH); // Leitura do valor de pulso de
entrada do PWM
  distanciadir = distanciadir / 2; // Define a medate da distância de
leitura
  distanciadir = distanciadir * 0.0343; // Converte a distância para a
distância real através da velocidade do som
}

void distancia_esq() {
  digitalWrite(SINAL_esq, HIGH); // Desliga o pino COMP/SINAL
  delayMicroseconds(11);
  digitalWrite(SINAL_esq, LOW); // leitura - Pin PWM pulsos de saída
  distanciaesq = pulseIn(PWM_esq, HIGH); // Leitura do valor de pulso de
entrada do PWM
  distanciaesq = distanciaesq / 2; // Define a medate da distância de
leitura
```

```
  distanciaesq = distanciaesq * 0.0343; // Converte a distância para a
distância real através da velocidade do som
}

void led() {
  for (int led = 0; led < 10; led++) {
    digitalWrite(ledVermelho, HIGH);
    tempo2();
    digitalWrite(ledVermelho, LOW);
    tempo2();
  }
}

void tempo() {
  delay(100);
}

void tempo2() {
  delay(250);
}

void vai_esquerda()
{
  analogWrite(Rodas_A1A, 0);
  analogWrite(Rodas_A1B, speed);
  analogWrite(Rodas_B1A, 0);
  analogWrite(Rodas_B2A, speed);
}

void vai_direita()
{
  analogWrite(Rodas_A1A, speed);
  analogWrite(Rodas_A1B, 0);
  analogWrite(Rodas_B1A, speed);
  analogWrite(Rodas_B2A, 0);
}

void mostrar() {
  Serial.print(distanciaovo); Serial.print(" cm  |  ");
Serial.print(distanciadir); Serial.print(" cm  |  ");
Serial.print(distanciaesq); Serial.print(" cm");
  Serial.print("  |  Posição: "); Serial.print(posicaocaixa);
Serial.print("  |  Lado: "); Serial.print(ladocaixa);
Serial.print("  |  Posição: ");
  Serial.print(posD); Serial.print("  |  Lado: "); Serial.print(posLado);
Serial.print("  |  Caixas Cheias: "); Serial.println(quantcaixas);
}
```