UNIVERSIDADE FEDERAL DO PAMPA – UNIPAMPA

PAULO RICARDO FIUZA MARQUES

PROJETO DE UM BRAÇO ROBÓTICO UTILIZANDO ATUADORES PNEUMÁTICOS E ELÉTRICOS CONTROLADOS PELO SISTEMA EMBARCADO ARDUINO

Alegrete
2016

PAULO RICARDO FIUZA MARQUES

PROJETO DE UM BRAÇO ROBÓTICO UTILIZANDO ATUADORES PNEUMÁTICOS E ELÉTRICOS CONTROLADOS PELO SISTEMA EMBARCADO ARDUINO

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia Elétrica.

Orientador: Prof. Maurício França, Msc. Eng.

Alegrete
2016

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais).

F565p Fiuza, Paulo Ricardo
Projeto de um braço robótico utilizando atuadores pneumáticos e elétricos controlados pelo sistema embarcado Arduino / Paulo Ricardo Fiuza. – 2016.
108 p.

Trabalho de Conclusão de Curso (Graduação) Universidade Federal do Pampa, ENGENHARIA ELÉTRICA, 2016.
"Orientação: Maurício França".

1. Controle e automação. 2. Braço robótico. 3. Atuadores elétricos. 4. Atuadores pneumáticos. 5. Arduino. I. Título.

PAULO RICARDO FIUZA MARQUES

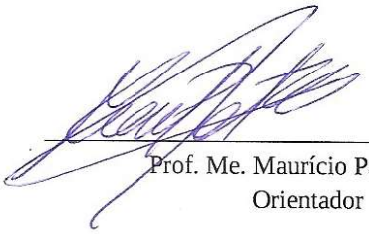
PROJETO DE UM BRAÇO ROBÓTICO UTILIZANDO ATUADORES PNEUMÁTICOS E ELÉTRICOS CONTROLADOS PELO SISTEMA EMBARCADO ARDUINO

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica da Universidade Federal do Pampa, como requisito parcial para obtenção do título de Bacharel em Engenharia Elétrica.

Área de Concentração: Robótica

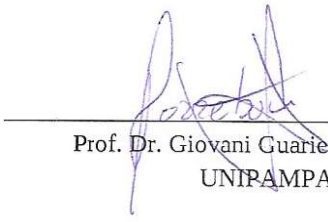
Trabalho de Conclusão de Curso defendido e aprovado em: 5 de dezembro de 2016.

Banca examinadora:



Prof. Me. Maurício Paz França
Orientador

Prof. Dr. Sidinei Ghissoni
UNIPAMPA



Prof. Dr. Giovani Guarienti Pozzebon
UNIPAMPA

Dedico este trabalho à minha família por todo esforço e incentivo para que fosse possível a conclusão da minha formação acadêmica.

AGRADECIMENTO

Agradeço aos meus familiares por todo o apoio e incentivo para que fosse possível a realização desse sonho. Aos amigos que estiveram junto durante a minha caminhada, que pude dividir momentos de lazer, estudos e profissional. E a todo o aporte material oferecido por todos para que fosse possível a realização desse trabalho.

Agradeço a Universidade Federal do Pampa pela formação, estrutura física fornecida e oportunidade de estudar em um curso de graduação de alto nível em minha cidade natal.

E ao professor Maurício França pela orientação e por compartilhar seus conhecimentos, e a todos os professores que contribuíram para minha formação acadêmica.

“Não creio que haja uma emoção mais intensa para um inventor do que ver suas criações funcionando”.

Nikola Tesla

RESUMO

A indústria brasileira encontra grande competição com produtos fabricados fora do país, que na maioria das vezes possuem qualidade de fabricação superior e com preços mais competitivos com o que se é produzido no Brasil. Uma das variáveis que causam essa perda são parques fabris pouco ou sem nenhuma forma de automatização, deixando-os com processos menos eficientes. Processos automatizados requerem um alto nível de investimento, sendo inviável para pequenas empresas que precisem produzir em larga escala. Considerando esse déficit na realidade da indústria nacional, deseja-se com este projeto estimular a pesquisa e projeto na área da automação industrial, assim como áreas afins, auxiliando como ponto de partida a construção de manipuladores robóticos de baixo custo. Portanto realizou-se o estudo, análise e desenvolvimento de um braço mecânico robótico eletropneumático com 4 graus de liberdade para movimentação de cargas de até 500 g. O projeto envolve a utilização de cilindros pneumáticos, motores de passo e servos-motores, como atuadores, sendo controlados por um *joystick* projetado. Utiliza o microcontrolador ATmega2560 embarcado no sistema Arduino MEGA R3, como módulo de processamento central do manipulador. Através de um *firmware* desenvolvido se realizam as operações de controle do braço, possuindo as funções de gravar posições e realizar movimentos no modo automático. O projeto possui um custo na faixa dos R$150,00, a maioria do seu material é procedente de sucata e/ou retirado dos laboratórios da UNIPAMPA, sendo a sua estrutura construída com chapas de madeira.

Palavras-chave: Controle e automação; Controle de processos; *Arduino;* Braço robótico; Mecatrônica; Atuadores pneumáticos; Atuadores elétricos; Processos automatizados.

ABSTRACT

Brazilian industry finds great competition with products manufactured outside the country, which in most cases have superior manufacturing quality and prices that are more competitive with what is produced in Brazil. One of the variables that causes this loss is little or no automated manufacturing parks, leaving them with less efficient processes. Automated processes require a high level of investment and are not feasible for small businesses that want to produce on a large scale. Considering this deficit in the reality of the national industry, the project is a research project and project in the area of industrial automation, as well as related areas, as a starting point for the construction of low cost robotic manipulators. Therefore, the study, analysis and development of an electro-pneumatic robotic mechanical arm with 4 degrees of freedom was carried out to handle loads up to 500 g. It combines the use of pneumatic cylinders, pitch motors and servo motors, such as actuators, controlled by a designed joystick. And it has the ATmega2560 microcontroller embedded in the Arduino MEGA R3 system, as the central processing module of the manipulator. Through a firmware developed the operations of control of the arm, having the functions of recording positions and perform movements in the automatic mode. The project has a cost in the range of R$ 150.00, most of its material comes from scrap and/or removed from the laboratories of UNIPAMPA, and its structure is built with wood place.

Keywords— Automation and Control; process control; Arduino; robotic arm; mechatronics; pneumatic actuator; electrical actuator; automation process.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

N_{in}	-	Número de dentes da engrenagem eixo de entrada
N_{out}	-	Número de dentes da engrenagem eixo de saída
p.	-	Página
T_{in}	-	Torque de entrada
V_{in}	-	Tensão de Entrada
V_{REF}	-	Tensão de Referência

LISTA DE SIGLAS

A	-	*Ampére*
ABNT	-	Associação Brasileira de Normas Técnicas
AC-DC	-	*Alternating Currente - Direct Current*
ADC	-	*Analog-to-Digital Converter*
AREF	-	Analog Reference
b	-	*Byte*
CAD	-	*Computer Aided Design*
CI	-	Circuito Integrado
EPROM	-	*Erasable Programmable Read-Only Memory*
EEPROM	-	*Electrically-Erasable Programmable Read-Only Memory*
F	-	Força
FIFO	-	*First In First Out*
FTDI	-	*Future Technology Devices International*
g	-	Gramas
ICSP	-	*In-Circuit Serial Programming*
IDE	-	*Integrated Development Environment*
I^2C	-	*Inter-Integrated Circuit*
k	-	*kilo*
mm	-	Milímetro
MISO	-	*Master In Slave Out*
MOSI	-	*Master Out Slave In*
n	-	Número de *bits* do conversor
N	-	Newton
PWM	-	*Pulse Width Modulation*
RX	-	*Receive Data*
SCK	-	*Serial Clock* (I^2C)
SCL	-	*Serial Clock* (SPI)
SDA	-	*Serial Data*
SPI	-	*Serial Peripheral Interface*
SRAM	-	*Static Random Access Memory*
SS	-	*Slave Select*

T	-	Torque
TX	-	*Transmit Data*
TTL	-	*Transistor-Transistor Logic*
TWI	-	*Two-Wire Interface*
UART	-	*Universal Asynchronous Receiver/Transmitter*
USB	-	*Universal Serial Bus*
V	-	*Volt*
W	-	*Watts*

LISTA DE SÍMBOLOS

β	-	Ângulo de passo
ω_{out}	-	Velocidade de saída
Ω	-	*Ohm*

SUMÁRIO

1. INTRODUÇÃO

Na última década o desenvolvimento da área de microeletrônica está se tornando cada vez mais rápido, os circuitos eletrônicos estão cada vez mais compactos e eficientes, com a redução significativa das suas dimensões e custos. Associados diretamente à microeletrônica, os sistemas microprocessados digitais e a ciência da computação também se desenvolveram rapidamente, desenvolvendo computadores mais poderosos, possuindo grande desempenho computacional. Com a concepção dos circuitos integrados possibilitou-se a produção em larga escala e com baixo custo de microprocessadores dedicados, conhecidos como microcontroladores. Atualmente esses dispositivos eletrônicos estão presentes não apenas em máquinas e equipamentos industriais, mas também em automóveis e outros produtos eletrônicos do nosso cotidiano. Os sistemas mecânicos também sofreram profundas modificações conceituais com a incorporação desses processadores e, com isso, tornaram-se mais rápidos, eficientes e confiáveis, com custos de implementação cada vez menores [1].

Segundo [2], em contrapartida com a crescente evolução desses componentes a indústria brasileira esta diante de problemas de competição internacional, obsolescência e inadequação de recursos humanos - este último se constitui no principal obstáculo para a modernização do parque industrial brasileiro. Muitas indústrias, por questão de sobrevivência, buscam modernizar métodos e equipamentos. Nesse contexto, já existe certo consenso de que a indústria nacional precisa reestruturar-se e capacitar-se para competir no mercado internacional, devido à facilidade de ingresso de produtos estrangeiros, estimulado pela globalização da economia. Então diretamente associado a esses avanços, está à utilização de robôs em diversos setores de produção. Um dos grandes motivos para o aumento do seu uso é que o custo vem declinando devido, principalmente, aos avanços na microeletrônica. Os robôs não estão ficando apenas mais baratos, mas estão se tornando eficientes, estão mais rápidos, precisos e flexíveis. À medida que os robôs se tornam mais econômicos na execução de suas funções e a mão de obra humana ficam cada vez mais cara, aumentam as aplicações industriais candidatas à automação robótica. Essa é a principal tendência que vem incentivando o crescimento do mercado de robôs industriais. Uma tendência secundária é que,

desconsiderando o aspecto econômico, à medida que os robôs passam a ser mais capacitados, tornam-se aptos a realizar cada vez mais tarefas perigosas ou impossíveis de serem executadas por trabalhadores humanos [2].

Um braço robótico pode ser comparado ao corpo humano, onde o sistema embarcado é o cérebro, realiza as operações de processamento. Nessa analogia o conhecimento e a capacidade de identificar as variáveis externas é o *firmware* existente dentro do microcontrolador, e os sensores representam os órgãos sensoriais identificando posição, pressão, torque e outros parâmetros no robô. Os membros inferiores e superiores do corpo humano são representados pelos atuadores pneumáticos, elétricos e/ou hidráulicos, que executam a movimentação física do braço robótico. O sistema nervoso central é a rede de comunicação entre os componentes e partes do robô, criando uma linha de troca de informações. O sistema sanguíneo é representado pelos cabos e condutores utilizados, assim como a estrutura mecânica lembra o esqueleto. E como o sistema biológico humano é necessário alimenta-lo, utilizando energia elétrica, ar, e/ou óleo [1].

No presente trabalho são apresentados tópicos do projeto de um braço robótico, no sentido de analisar a construção desse modelo proposto demonstrando as dificuldades de desenvolvimento físico, bem como o conhecimento obtido dentro das áreas abordadas pelo tema. O braço mecânico robótico desenvolvido possui como função básica a movimentação de cargas leves, até 500 g, com intuito de se desenvolver um produto de baixo custo e servindo futuramente para estudos dirigidos em aplicações mais específicas em processos industriais, que exigem maior aplicação de controle e estabilidade, empregando-se sensores.

Construiu-se um braço mecânico do tipo articulado com quatro graus de liberdade, que consiste em movimentos semelhantes a um braço humano. Possuindo cinco articulações, base, ombro, cotovelo, punho e garra. A estrutura física foi projetada com três membros/elos principais, utilizando chapas de madeira, com dois tipos de atuadores, motores de passo e servo-motores (elétricos) e cilindros pneumáticos. Com juntas realizando a conexão entre os membros, possibilitando a interligação dos movimentos do braço. E um manipulador do tipo garra para realizar o movimento de pinçar um objeto.

A base é movimentada por dois cilindros pneumáticos possibilitando uma lógica com três posições fixas. Assim como na base, no cotovelo também foi utilizado um cilindro pneumático, restringindo o movimento a elevado ou abaixado, duas posições

fixas. Em contrapartida no ombro foram utilizados dois motores de passo, possibilitando um movimento de até 270º no eixo vertical. No punho e garra foram utilizados servos-motores, no punho o movimento também se restringe no eixo vertical, possuindo liberdade de até 180º. E na garra o movimento se dá no eixo horizontal, onde a abertura máxima possui até 90º, e o mínimo até pressionar o objeto, verificado manualmente pelo operador. Esse sistema é todo operado através de um módulo de controle contendo chaves de impulso (*push button*) e potenciômetros, utilizando-se no processamento de dados o sistema embarcado Arduino.

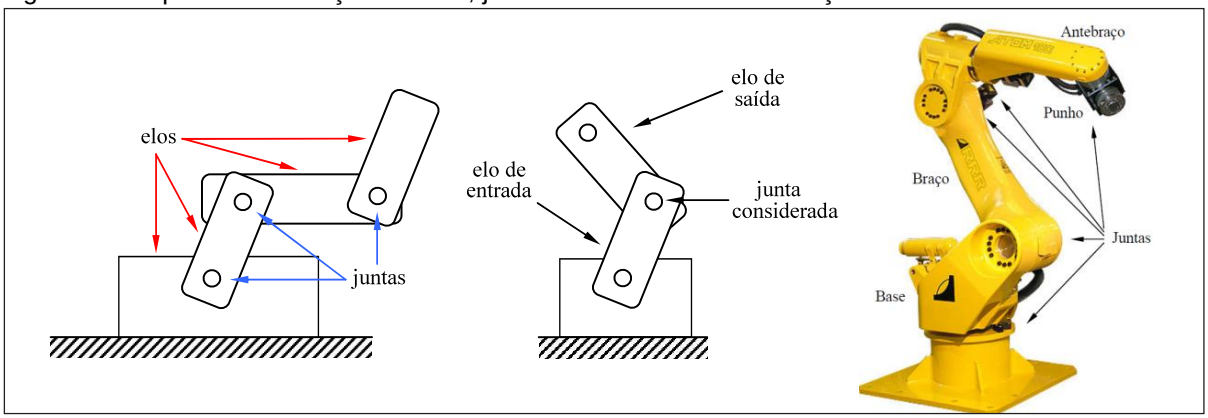
Portanto, nesse trabalho são apresentados temas pertinentes à efetivação do projeto, de forma a orientar o objeto de estudo e descrição das atividades. Os estudos realizados proporcionaram grande aprendizagem teórica e prática para os envolvidos, e também poderá ser útil no ambiente acadêmico para fins didáticos, assim como refinamento no seu funcionamento para fins industriais. Foi possível então, realizar o projeto de um braço robótico de baixo custo, utilizando principalmente componentes de sucatas de impressoras, e demais oriundos dos laboratórios da Universidade Federal do Pampa.

2. REVISÃO BIBLIOGRÁFICA

2.1 Manipuladores Robóticos

A estrutura dos braços robóticos ou manipuladores costumam ser divididos em duas partes principais, o braço e o punho (Figura 1). Onde o braço é identificado por elos conectados por juntas/eixos, onde são acoplados os atuadores que executam o movimento, é fixado por um lado pela base e na outra pelo punho. Já o punho é caracterizado conforme a aplicação do manipulador, consiste basicamente de elos compactos que interligam varias juntas próximas entre si [3].

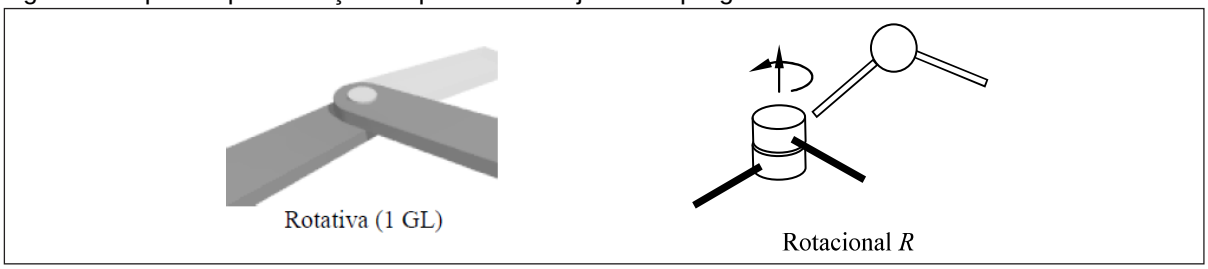
Figura 1 - Esquema de notação de elos, juntas e anatomia de um braço.



Fonte: [3] (p. 8 e 9).

As juntas do braço robótico podem ser rotativas, cilíndricas, prismáticas, de parafusos, esféricas e planares, podem se mover de uma a três direções, dependendo da quantia de graus de liberdade. As juntas rotativas (Figura 2) ainda possuem outras três divisões que são a rotativa de torção, rotativa rotacional e a rotativa revolvente [3].

Figura 2 - Tipo e representação esquemática da junta empregada.

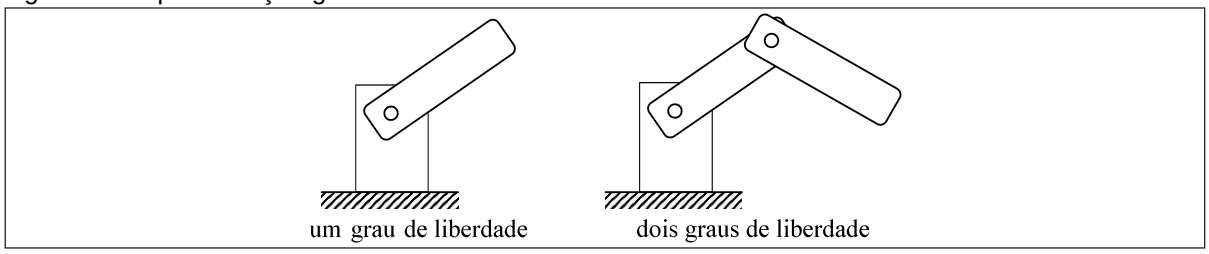


Fonte: [3] (p. 10).

Os graus de liberdade (GL) definem os movimentos do braço no espaço dimensional, cada junta determina o número de graus de liberdade, sendo a soma desses o número total do manipulador (Figura 3). Quanto maior o número de graus

de liberdade maior é o alcance de operação, em contrapartida maior são os problemas com cinemática, dinâmica e controle [3].

Figura 3 - Representação graus de liberdade.

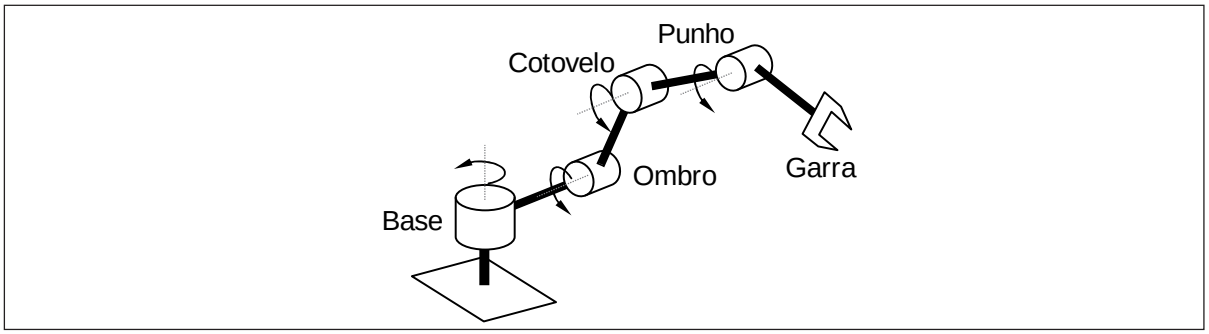


Fonte: [8] (p. 11).

Os robôs são configurados conforme o tipo de juntas que possui, recebendo as seguintes notações, cartesiano, cilíndrico, esférico, SCARA, articulado e paralelo. Possuem sistemas de acionamento responsáveis pelo movimento das articulações e desempenho dinâmico, esses acionadores podem ser hidráulicos, pneumáticos e/ou elétricos [3].

O presente projeto utiliza juntas rotativas rotacionais, que permitem o giro em torno de um eixo rotacional estacionário, apresentam elos de entrada e saída perpendiculares ao eixo de ligação. Com 4 graus de liberdade, apresenta movimento rotacional horizontal em torno de um eixo vertical na base, e rotacional vertical nas demais juntas movendo-se em torno de um eixo horizontal. O punho apresenta movimento em arfagem, que é a rotação de cima para baixo também em torno de um eixo horizontal, e uma garra de dois “dedos” com movimento de abertura e fechamento na horizontal com eixo vertical. Configurando-se assim como robô articulado ou revoluto, que possuem três juntas rotativas, assemelhando-se ao braço humano, com braço, antebraço e pulso (Figura 4). São os mais utilizados na indústria por possuir uma configuração muito versátil, assegurando uma grande gama de movimentos dentro de um espaço compacto. Apresentando elementos de acionamento ou atuadores com a integração de cilindros pneumáticos e elétricos, contando com motores de passo e servo-motores [3].

Figura 4 - Esquema resultante de juntas do braço robótico.



Fonte: Autor.

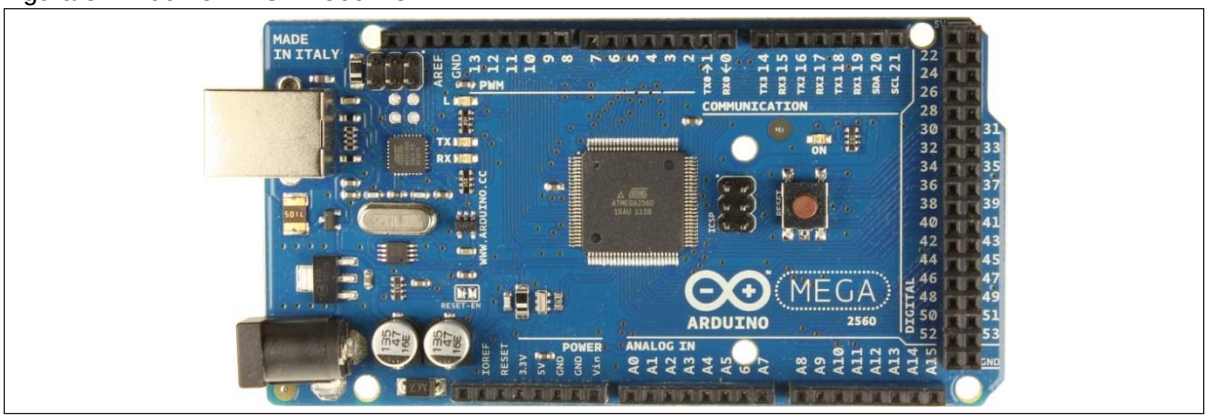
2.2 Arduino

É um pequeno computador programável que processa entradas e saídas entre um dispositivo e os componentes externos conectados a ele. Normalmente chamado de sistema embarcado, sistema que pode interagir com seu ambiente por meio de *hardware* e *firmware*. Para programá-lo é necessário utilizar o seu IDE, *software* livre no qual possibilita escrever o código na linguagem que o ele compreenda (C++, java). O IDE permite escrever um programa de computador (*firmware*), que é um conjunto de instruções passo a passo, das quais é feito o upload para o Arduino. Então esse executará essas instruções, interagindo com periféricos conectados [4].

2.2.1 Arduino MEGA 2560

Utilizou-se no projeto o Arduino MEGA 2560 R3 (Figura 5), é a revisão três da placa que utiliza o microcontrolador Atmega2560. Possui 54 pinos digitais (entrada/saída) sendo que 15 podem ser usados como saídas PWM (2 a 13 e 44 a 46), 16 pinos analógicos, 4 USARTs (Portas Seriais de Hardware), um cristal oscilador de 16 MHz, entrada USB, entrada de alimentação, soquete de comunicação ICSP e um botão *reset*. A alimentação pode ser feita através do cabo USB, fonte de alimentação AC - DC ou bateria [5]. Enquadrando-se perfeitamente na necessidade de processamento e número de portas para o projeto.

Figura 5 - Arduino MEGA 2560 R3.



Fonte: Autor.

O Mega 2560 tem um polifusível que protege as portas USB do computador contra curto - circuitos e sobre - corrente. Apesar da maioria dos computadores fornecerem sua própria proteção interna, o fusível oferece uma camada extra de proteção. Se mais de 500 mA forem aplicados na porta USB do computador, o fusível romperá automaticamente a conexão até a curto ou a sobrecarga se extinguir [5].

Pode ser alimentado pela conexão USB ou por qualquer fonte de alimentação externa. A fonte de alimentação é selecionada automaticamente, a alimentação externa pode ser tanto de uma fonte ou de uma bateria. A fonte pode ser conectada com um *plug* P4 de 2,1 mm (centro positivo) no conector de alimentação. Cabos vindos de uma bateria podem ser inseridos nos pinos GND (terra) e V_{in} (entrada de tensão) do conector de alimentação. A placa pode operar com uma alimentação externa de 6 a 20 V. Entretanto, se a alimentação for inferior a 7 V, o pino 5 V pode fornecer menos de 5 V e a placa ficará instável. Se a alimentação for superior a 12 V o regulador de tensão pode superaquecer e avariar a placa. Portanto a alimentação recomendada é de 7 a 12 V [5].

Os pinos de alimentação são [5]:

V_{in}: entrada de alimentação para a placa Arduino quando uma fonte externa for utilizada. Pode-se fornecer alimentação por este pino ou, se usar o conector de alimentação, alimentar por este pino;

5 V: a fonte de alimentação utilizada para o microcontrolador e para outros componentes da placa. Pode ser proveniente do pino V_{in} através de um regulador *on-board* ou ser fornecida pela porta USB;

3V3: alimentação de 3,3 V fornecida pelo circuito integrado FTDI (controlador USB). A corrente máxima é de 50 mA;

GND: pino terra.

O ATmega2560 tem 256 kB de memória *flash* (onde são armazenados os programas) dos quais 8 kB são utilizados para *bootloader*, além de 8 kB de SRAM (onde ficam as variáveis) e 4 kB de EEPROM (guarda os dados permanentemente, mesmo que se desligue a placa). Já memória SRAM é apagada toda vez que se desliga o circuito [5].

Cada um dos 54 pinos digitais do MEGA pode ser usado como entrada ou saída, utilizando-se as funções de *pinMode()*, *digitalWrite()*, e *digitalRead()*. Eles operam com 5 V, cada pino pode fornecer ou receber um máximo de 20 mA e tem um resistor *pull-up* interno (desconectado por padrão) de 20 – 50 kΩ. Um máximo de 40 mA é o valor que não deve ser ultrapassado nas portas afim de evitar danos permanentes ao microcontrolador [5].

Além disso, alguns pinos têm funções especializadas [5]:

Serial: 0 (RX) e 1 (TX);

Serial 1: 19 (RX) e 18 (TX);

Serial 2: 17 (RX) e 16 (TX);

De Série 3: 15 (RX) e 14 (TX).

Usados para receber (RX) e transmitir (TX) dados seriais TTL. Estes pinos são conectados aos pinos correspondentes do chip série ATmega16U2 USB-TTL [5].

Interrupções externa: 2 (interromper 0), 3 (interromper 1), 18 (interromper 5), 19 (interromper 4), 20 (interromper 3), e 21 (interromper 2). Estes pinos podem ser configurados para disparar uma interrupção num nível baixo, um flanco ascendente ou descendente, ou uma mudança de nível, função *attachInterrupt*() [5].

PWM: 2 a 13 e 44 a 46. Fornecem uma saída analógica PWM de 8 *bits* com a função *analogWrite*() [5].

SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). Estes pinos suportam comunicação SPI, que embora compatível com o *hardware*, não está incluída na linguagem do MEGA 2560 [5].

LED: 13. Há um LED interno conectado ao pino digital 13, quando o pino está em nível alto o LED liga, e quando está em nível baixo o LED desliga [5].

O MEGA 2560 possui 16 entradas analógicas, cada uma delas está ligada a um conversor analógico - digital de 10 *bits*, ou seja, transformam a leitura analógica em

um valor dentre 1024 possibilidades (exemplo: de 0 a 1023). Por padrão, elas medem de 0 a 5 V, embora seja possível mudar o limite superior usando o pino AREF e a função *analogReference*() [5].

Adicionalmente alguns pinos têm funcionalidades especializadas [5]:

I2C: 20 (SDA) e 21 (SCL) suportam comunicação I2C (TWI) usando a biblioteca *wire*;

Reset: Envia o valor *low* para o pino 1 do microcontrolador o reiniciando [5].

A comunicação entre o Arduino MEGA 2560 com um computador, com outro Arduino, ou com outros microcontroladores, é muito simplificada. O ATmega2560 permite comunicação serial no padrão UART TTL (5 V), que está disponível nos pinos digitais 0 (RX) e 1 (TX). Um chip ATmega16U2 na placa encaminha esta comunicação serial através da USB e fornecem uma porta virtual para o software no computador. O *software* Arduino (IDE) inclui um monitor serial que permite que dados simples de texto sejam enviados e recebidos à placa Arduino. Os LEDs RX e TX da placa piscam quando os dados estão sendo transferidos ao computador pelo *chip* ATmega16U2 e há conexão USB (mas não quando há comunicação serial pelos pinos 0 e 1). A biblioteca *SoftwareSerial* permite comunicação serial por quaisquer dos pinos digitais do MEGA 2560 [5].

2.3 Linguagem de programação C++

Segundo [6], o C++ foi inicialmente desenvolvido por *Bjarne Stroustrup* dos *Bell Labs*, durante a década de 1980 com o objetivo de implementar uma versão distribuída do núcleo Unix. Pode-se dizer que C++ foi a única linguagem entre tantas outras que obteve sucesso como uma sucessora à linguagem C, inclusive servindo de inspiração para outras linguagens como Java, a IDL de CORBA e C#. A biblioteca padrão do C++ incorpora a biblioteca padrão do C com algumas pequenas modificações para trabalhar melhor com as novas funcionalidades criadas pela linguagem [6].

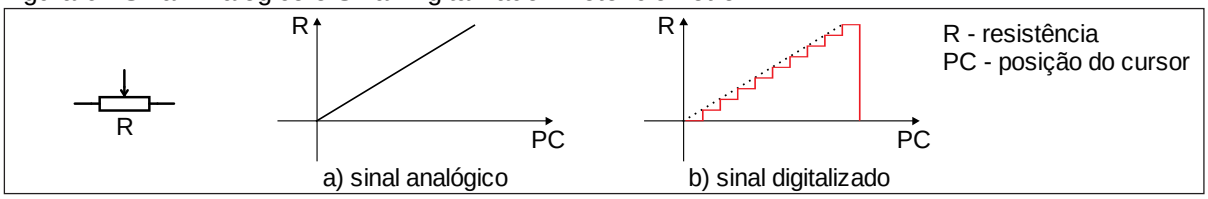
É compilado em três fases: pré-processamento, compilação propriamente dita (tradução para código objeto) e ligação. O C++ introduziu alguns conceitos de orientação a objetos ao C, como exemplificado pelas classes, que apresentam quatro características comumente presentes em linguagens de programação

orientadas a objeto: abstração, encapsulamento, herança e polimorfismo. Cada vez que uma classe é instanciada é criado um objeto na memória, que é basicamente um conjunto de atributos e operações reunidos. O tratamento de exceção é um mecanismo desenvolvido para lidar com a ocorrência de algumas condições (chamadas exceções) que alteram o funcionamento normal do fluxo de um programa de computador. O C++ suporta tal tratamento, de forma que o estado atual de um programa após uma exceção é alterado automaticamente para outro estado pré-definido para a recuperação do sistema [6] [7].

2.4 Conversão A/D

Os sinais utilizados no trabalho são oriundos de grandezas elétricas, em geral tensão em função do tempo. Os sinais podem ser analógicos ou digitais, sinais analógicos são aqueles que variam continuamente com o tempo, portanto, entre dois valores distintos do sinal existem infinitos valores. Em um sinal digital, a variação do valor do sinal com o tempo não é contínua, entre dois valores distintos do sinal, o total de valores no intervalo é finito, sendo dividido em nível alto ou baixo [8].

Figura 6 - Sinal Analógico e Sinal Digitalizado - Potenciômetro.



Fonte: Autor.

A conversão de um sinal analógico em digital (A/D) é de fundamental importância no processamento de sinais. Consiste em passar o valor de uma tensão analógica para um valor digital equivalente. Esse processo é basicamente um problema de amostragem do sinal, ou seja, medir periodicamente o sinal que se quer digitalizar e apresentar os valores medidos na forma digital (Figura 6). A taxa com que se repetem as medidas é chamada de frequência de amostragem. Percebe-se que, quanto maior for a frequência de amostragem, mais precisa será a reprodução do sinal em sua forma digital [8].

O Arduino realiza essa conversão A/D com um *clock* máximo de 200 kHz que fornece uma taxa de amostragem de aproximadamente 15 kHz. Os potenciômetros

conectados no mesmo realizam divisões de tensão que são interpretadas como mudanças de estados, logo esse valor analógico é transformado em um sinal digital. Como os potenciômetros são alimentados pelos 5 V do Arduino, e esse possui 10 *bits* de resolução, existem 1024 possibilidades digitais dentro do sinal analógico de 5 V, como já apresentado na seção 2.2.1.

$$ADC = \frac{V_{in} * 1024}{V_{REF}} \quad (1)$$

$$resolução = \frac{V_{REF}}{2^n} \quad (2)$$

2.5 Memória EEPROM

De acordo com [9], é um tipo de memória não volátil usada em computadores e outros dispositivos eletrônicos para armazenar pequenas quantidades de dados que precisam ser salvos quando a energia é removida. Permitem que o apagamento dos dados seja feito eletricamente e, ainda, isoladamente por palavra de dados, sem necessidade de reprogramação total. Atualmente já existe uma memória não volátil mais moderna, derivada da EEPROM, a memória *flash*. Quando grandes quantidades de dados estáticos devem ser armazenados, em unidades *flash* USB por exemplo, a memória *flash* é mais econômica do que os dispositivos tradicionais de EEPROM [9].

Enquanto uma EPROM é programada por um dispositivo eletrônico que dá voltagens maiores do que os usados normalmente em circuitos elétricos, e pode ser apagada apenas por exposição a uma forte luz ultravioleta, a EEPROM pode ser programada e apagada dentro do próprio circuito, eletricamente, pela aplicação de sinais de programação especiais. Embora uma EEPROM possa ser lida um número praticamente ilimitado de vezes, ela possui uma vida útil limitada, apresentando número finito de vezes que pode ser reprogramada (apagada e programada novamente), esse limite é delimitado pela deterioração interna do *chip* durante o processo de exclusão, que requer uma tensão elétrica mais elevada. Essa limitação foi estendida para um milhão de operações de gravação em EEPROM modernos. Entretanto essa vida útil da EEPROM deve ser considerada quando usadas em um computadores que tenham uma previsão de frequentes reprogramações. É por esta

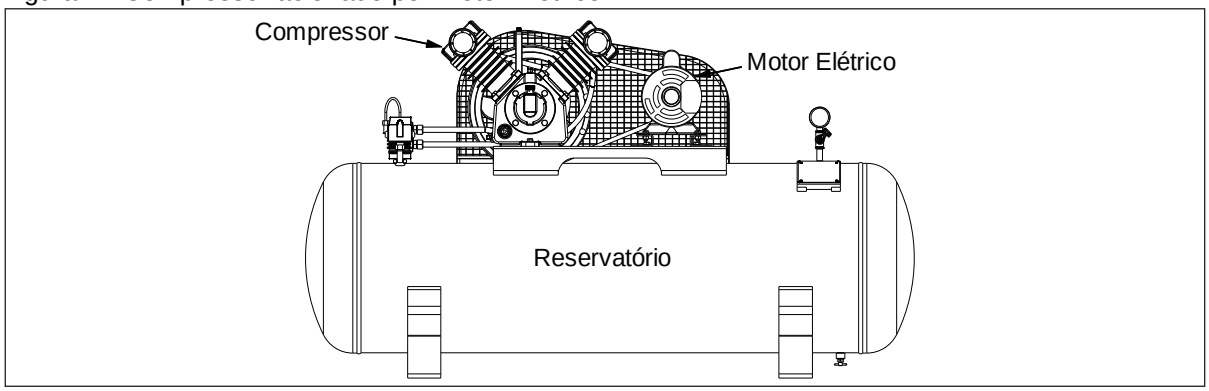
razão que EEPROM’s foram utilizados para informações de configuração, ao invés de memória de acesso aleatório [9].

Como cada novo dado gravado no *chip* requer o apagamento do anterior, considera-se apagamento e gravação como uma só operação, porém seria possível gravar o mesmo endereço de memória um *bit* de cada vez, fazendo então oito gravações com um só prévio apagamento. Entretanto a maioria das memórias EEPROM faz o apagamento do conteúdo do endereço automaticamente antes da gravação [9].

2.6 Sistemas Eletropneumáticos

Segundo [10], a energia pneumática provém da compressão do ar atmosférico em um reservatório, transformando-o em ar comprimido a uma dada pressão de trabalho, o equipamento que executa este processo é chamado de compressor. É uma velha forma de energia, porém só a partir de 1950 o ar comprimido foi aplicado industrialmente na automação e na racionalização da força humana, para trabalhos cíclicos e metódicos. Tornando-se indispensável nos mais variados ramos industriais [10]. É o elemento mais simples, de maior rendimento, e de menor custo que pode ser utilizado na solução de diversos problemas de automatização. Possui uma série de características próprias de seu fluído de utilização, o ar, que possibilitam favorecimento na sua utilização. Como quantidades ilimitadas de ar, fácil transporte por tubulação, armazenado em reservatório (Figura 7), insensível às oscilações de temperatura, não apresenta riscos de explosão, não há poluição ambiental, seus elementos de comando são materiais leves, alta velocidade de deslocamento, e podem ser solicitados em carga, até parar, sem sofrer danos. Entretanto é necessário tratamento do ar, pois requerem isenção de impurezas e umidade, não possibilita controle de velocidade suave com precisão e constante durante muitos ciclos, como não é um sistema robusto não se recomenda para trabalhos que exijam muita força, apresenta fuga de ar, e o seu custo de implantação é considerável, porém é rentável quanto ao custo de utilização [11].

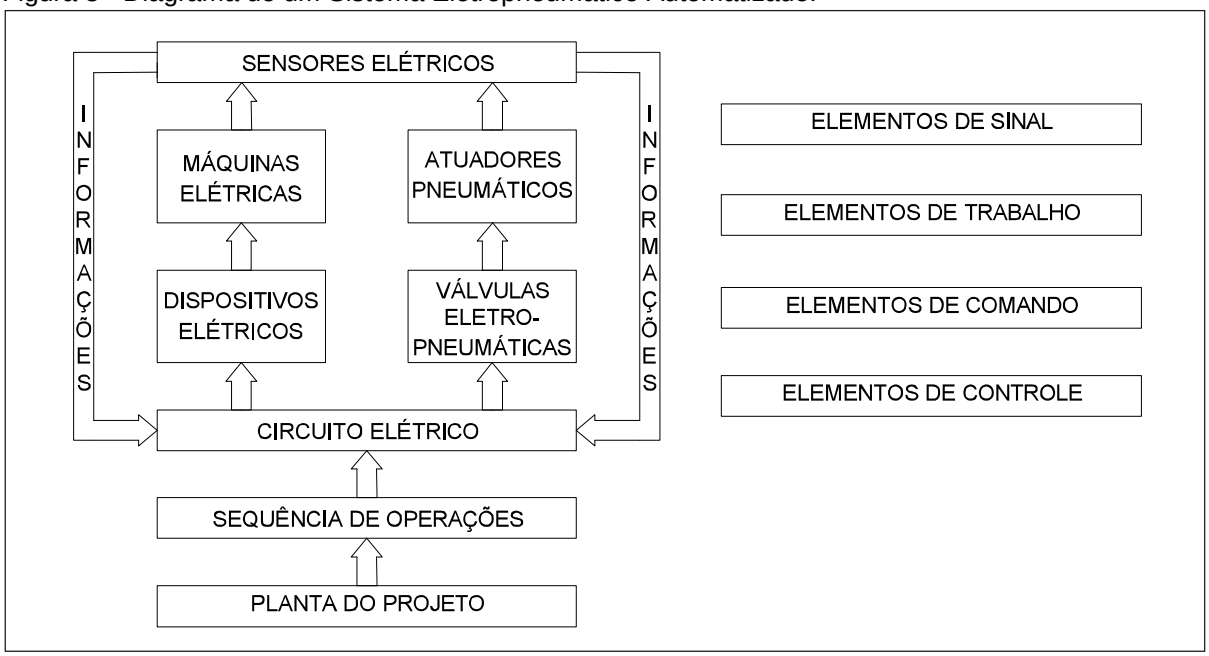
Figura 7 - Compressor acionado por Motor Elétrico.



Fonte: [12] (p. 16).

Os processos industriais que utilizam a combinação da energia pneumática com a elétrica são chamados de automação eletropneumática. Um sistema eletropneumático automatizado é composto pelas seguintes partes: elementos de sinal, elementos de trabalho, elementos de comando e elementos de controle (Figura 8). Onde a partir da planta do projeto de automação, retira-se a sequência de operações dos elementos de trabalho. Gerando-se um diagrama que informa a posição de cada elemento de trabalho nas etapas do processo automatizado, que é a programação. A partir dessa sequência de operações, constrói-se o circuito elétrico [10].

Figura 8 - Diagrama de um Sistema Eletropneumático Automatizado.



Fonte: [10] (p. 03).

Elementos de Controle - é um circuito elétrico que aciona os elementos de comando conforme as informações fornecidas pelos sensores juntamente com a sequência de operação [10].

Elementos de Comando - são válvulas pneumáticas, relês e contatores, que acionam os elementos de trabalho [10].

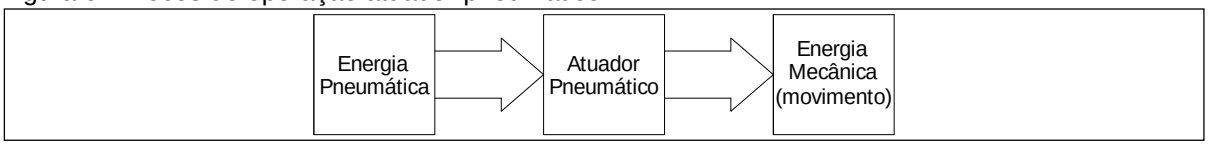
Elementos de Trabalho - transformam energia elétrica e pneumática em outras formas de energia. São os motores elétricos, cilindros e motores pneumáticos que executam uma determinada tarefa automaticamente, fazendo isso acionam os elementos de sinal [10].

Elementos de Sinal - são os sensores que informam continuamente ao elemento de controle sobre os estados do processo automatizado. Através da utilização desses o sistema se torna realimentado estabelecendo-se um processo automatizado [10].

2.7 Atuador Pneumático de Movimento Retilíneo

De acordo com [10], os atuadores pneumáticos são responsáveis pela transformação da energia pneumática em energia mecânica (movimentos retilíneo, angular e rotativo), na Figura 9 é apresentado esse processo.

Figura 9 - Blocos de operação atuador pneumático.

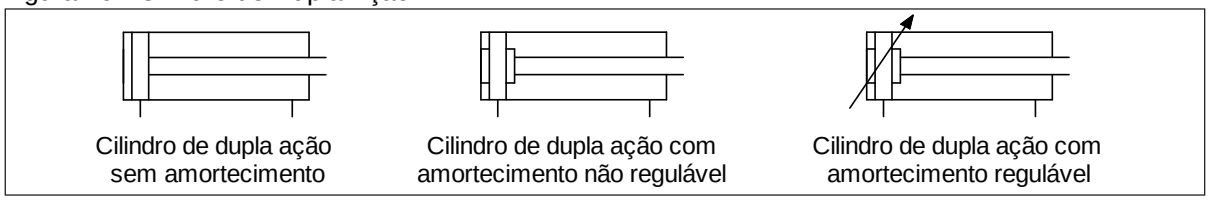


Fonte: [10] (p. 05).

2.7.1 Cilindro de Dupla Ação

Neste tipo de cilindro, o ar comprimido produz movimento nos dois sentidos, tem-se o avanço e retorno do cilindro através da energia pneumática (Figura 10). Quando grandes cargas são movimentadas por este tipo de cilindro, deve existir no mesmo um sistema de amortecimento pneumático que evite danificações devido aos fortes impactos nos fins de curso [10].

Figura 10 - Cilindro de Dupla Ação.



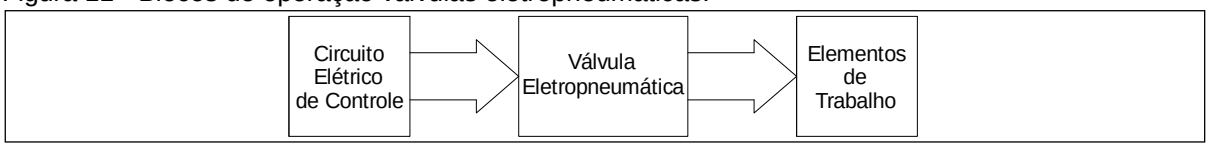
Fonte: [10] (p. 06).

Foi utilizado o cilindro de dupla ação da FESTO modelo DSNU-Q, que permite a aplicação de força tanto no avanço como no retorno.

2.8 Válvulas Eletropneumáticas

Segundo [10], as válvulas eletropneumáticas são os componentes do sistema automatizado que recebem o sinal elétrico do circuito de controle, acionando com isso, os elementos de trabalho, cilindros pneumáticos. Na Figura 11 é demonstrado o diagrama de blocos desse processo [10].

Figura 11 - Blocos de operação válvulas eletropneumáticas.



Fonte: [10] (p. 13).

Nas válvulas existem características básicas que definem o seu funcionamento, como as vias, posições e solenoides/bobinas [10].

Vias - são os orifícios que a válvula possui para a passagem do ar comprimido. Quanto à função, dividem-se em: conexão de entrada de ar comprimido (pressão), conexões para alimentação dos atuadores pneumáticos (utilização), orifícios de escape. As vias são identificadas através de letras maiúsculas ou números conforme simbologia ABNT, mostrado na Tabela 1 [10].

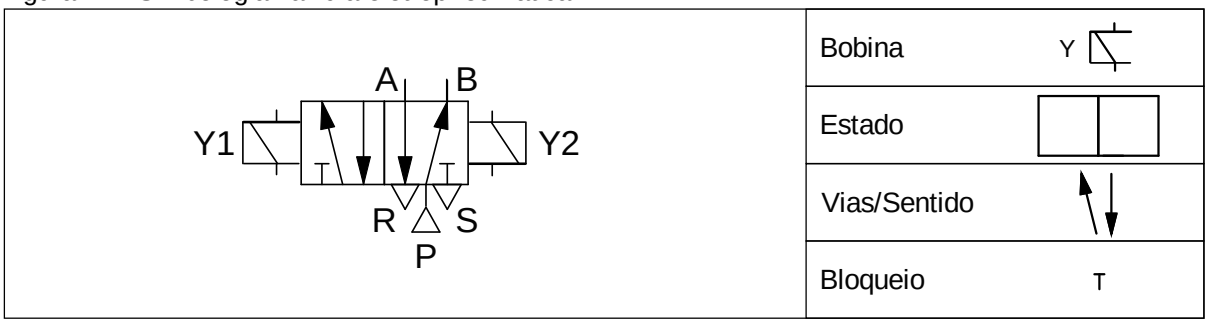
Tabela 1 - Válvulas Eletropneumáticas – Identificação vias.

Vias	Letras	Números
Pressão	P	1
Utilização	A, B, C	2, 4, 6
Escapes	R, S, T	3, 5, 7

Fonte: [10] (p. 14).

Posições - é o número de estados que a válvula pode ter ou permanecer. Cada posição que a válvula pode assumir é representada por meio de um quadrado. As linhas dentro destes quadrados indicam as vias de passagem de ar comprimido e as setas indicam o sentido. O bloqueio das vias é representado através de traços horizontais e o interligamento é identificado por um ponto. As conexões da válvula com o sistema são representadas por linhas externas no estado de repouso da válvula, juntamente com a identificação destas conexões. O estado de repouso é a posição que a válvula assume enquanto não é acionada eletricamente (Figura 12) [10].

Figura 12 - Simbologia válvula eletropneumática.



Fonte: [10] (p. 17).

O sistema de acionamento das válvulas é representado externamente por meio de solenoides. Podem-se ter válvulas acionadas por um solenoide, conhecidas como válvulas com comando unidirecional e válvulas acionadas por dois solenoides, identificadas como válvulas com comando bidirecional [10].

2.8.1 Válvula Eletropneumática Direcional e Bidirecional

De acordo com [10], o funcionamento deste tipo de válvula se baseia no deslocamento de um núcleo metálico mediante a ação de um campo magnético, determinando a trajetória do fluxo de ar. A força magnética, por sua vez, é criada

pela circulação da corrente elétrica no solenoide da válvula, a cada mudança de estado lógico a bobina também altera seu estado físico. Esse tipo de acionamento consome mais energia, pois é necessário que se mantenha circulação de corrente na bobina para que essa mantenha seu estado [10].

Na válvula com acionamento bidirecional, não há necessidade de se manter a corrente elétrica no solenoide para que ela permaneça em um determinado estado, consumindo menos energia. O acionamento e o desligamento são executados por pulsos de corrente elétrica de curta duração [10].

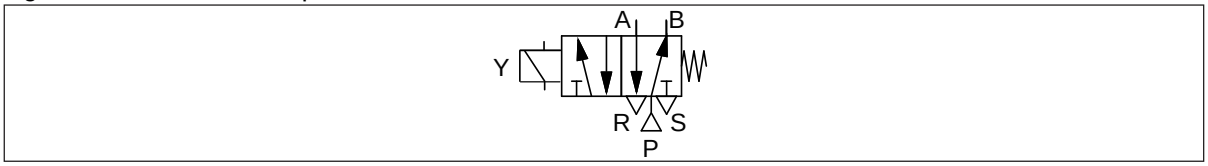
Como inicialmente não se está considerando eficiência energética do manipulador, foram utilizadas três válvulas eletropneumáticas de 5/2 vias com acionamento unidirecional da FESTO modelo MFH-5-1/8, com um solenoide de 24 V acoplado. E duas válvulas reguladoras de fluxo unidirecional, também da FESTO.

2.8.1.1 Válvula Eletropneumática de 5 Vias e 2 Estados (5/2) com Acionamento Unidirecional

Na posição de repouso desta válvula, o orifício P é direcionado ao B, e a via A é ligada ao escape R, não sendo utilizado o escape S [10].

Acionando-se o solenoide Y, a válvula troca de estado, ligando o orifício P ao A, a via B é ligada a S e o escape R não é usado. Enquanto o solenoide estiver acionado, a válvula permanece neste estado, caso contrário, retoma a posição de repouso. Na Figura 13 é apresentada a sua simbologia [10].

Figura 13 - Válvula Eletropneumática de 5/2 com Acionamento Unidirecional.

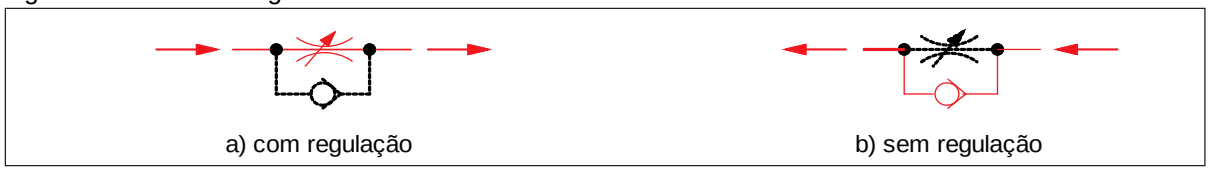


Fonte: [10] (p. 16).

2.8.1.3 Válvula Reguladora de Fluxo Unidirecional

Para realizar o ajuste de velocidade dos atuadores pneumáticos, utilizam-se as válvulas reguladoras de fluxo unidirecionais, que tornam o movimento mais suave. Nestes tipos de válvulas, a regulagem da vazão de ar é realizada somente em uma direção, mediante o ajuste manual do parafuso de estrangulamento (Figura 14a), e o fluxo de ar passa livremente no sentido contrário (Figura 14b) [10].

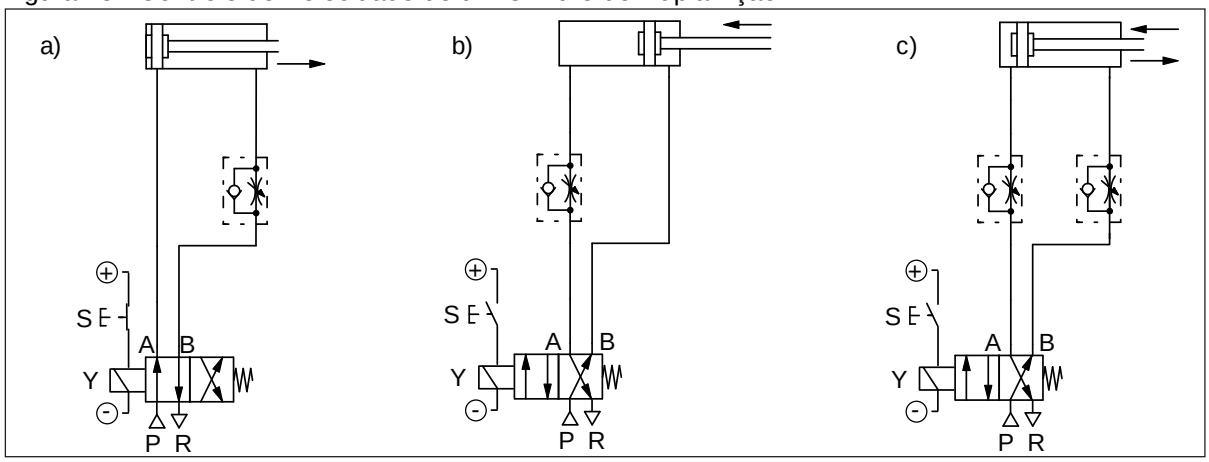
Figura 14 - Válvula Reguladora de Fluxo.



Fonte: [10] (p. 79).

Existem duas maneiras de controlar a velocidade nos atuadores de dupla ação. A primeira maneira é regular o fluxo na entrada do atuador, porém no acionamento de cargas pesadas, este método tem uma péssima regulação de velocidade devido à compressibilidade do ar. A segunda maneira de controlar a velocidade é regular o fluxo de ar na saída do atuador [10]. No projeto foram associadas às duas maneiras de controle, na entrada e saída. Recomendada sua utilização quando em cilindros de pequeno porte, de cursos e diâmetros reduzidos [10].

Figura 15 - Controle de Velocidade de um Cilindro de Dupla Ação.



Fonte: [10] (p. 81).

A Figura 15a mostra a posição correta da válvula reguladora de fluxo para controlar a velocidade de avanço. A Figura 15b apresenta o controle na velocidade

de retorno do cilindro. Unindo ambos os controles, tem-se o ajuste das velocidades de avanço e retorno, Figura 15c [10].

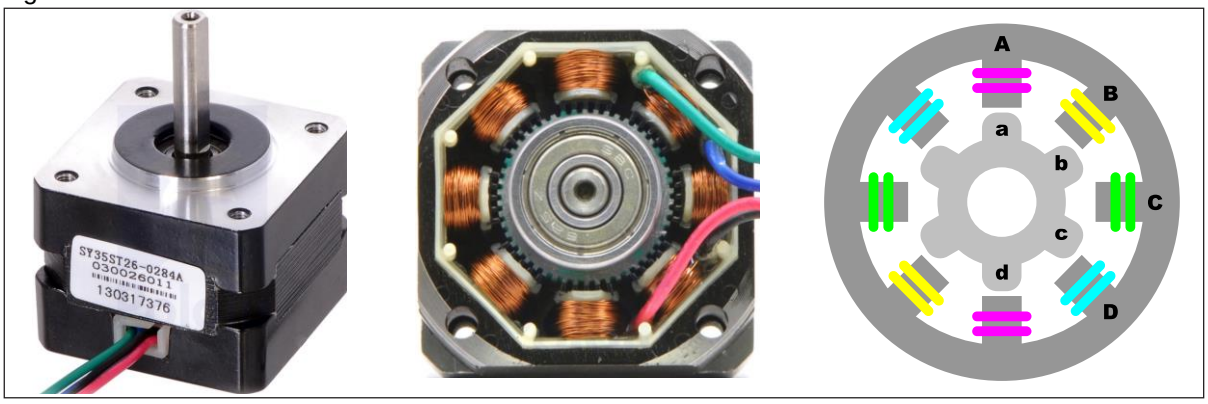
2.9 Atuadores Elétricos

De acordo com [3], são os motores de passo e servos-motores, geralmente utilizados em robôs de médio e pequeno porte, onde se demanda menor força e custos. Oferecem muita velocidade e baixo torque, mas possuem ótima precisão, normalmente utilizam redutores de velocidade para reduzir velocidade e aumentar a força torque [3].

2.10 Motor de Passo

Segundo [13], o motor de passo (Figura 16) foi inventado em 1936 por *Marius Lavet*, e é um tipo de motor elétrico muito utilizado quando necessário controle de posição ou movimento de forma precisa. Possuem esse nome porque giram em pequenos passos, e cada passo corresponde a um pulso que é fornecido aos enrolamentos do estator. Dependendo do seu modelo, pode avançar 90º, 45º, 18º, e inclusive frações de grau por pulso. Variando a velocidade dos pulsos pode-se girar lentamente, um passo por vez, ou rapidamente [13].

Figura 16 - Motor de Passo.



Fonte: [3] (p. 26).

O número de passos é dado pelo número de alinhamentos possíveis entre o rotor e as bobinas. Ou seja, para aumentar o número de passos de um motor se usa um número maior de bobinas, maior número de polos no rotor (para isso se coloca uma

roda dentada). Neste tipo de motor a rotação do eixo é controlado por uma série de campos eletromagnéticos que são ativados e desativados eletronicamente. Não utilizam escovas ou comutadores, e possuem um número fixo de polos magnéticos que determinam o número de passos por revolução (passos/rev). Os motores de passo mais comuns possuem de 3 a 72 passos/rev, significando que ele leva de 3 a 72 passos para completar uma volta. Controladores avançados de motores de passo podem utilizar PWM para realizarem micro passos, obtendo uma maior resolução de posição e operação mais suave, em detrimento de outras características [13].

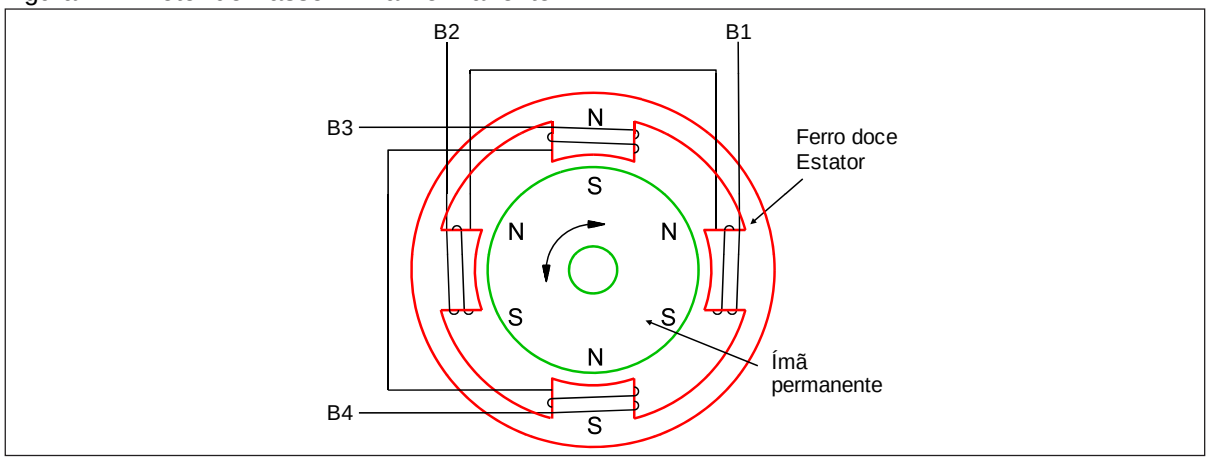
São classificados pelo torque que produzem, então para atingir todo o seu torque, suas bobinas devem receber toda a corrente marcada durante cada passo. Os seus controladores devem possuir circuitos reguladores de corrente para poderem fazer isto. A marcação de tensão (se houver) é praticamente sem utilidade. O controle computadorizado de motores de passo é uma das formas mais versáteis de sistemas de posicionamento, particularmente quando digitalmente controlado como parte de um servo sistema [13].

O número de polos do estator de um motor de passo nunca é igual ao número de polos do rotor. Essa característica é totalmente diferente de outros tipos de motores. Sendo essa diferença do número de polos o que confere a esse tipo de motor operar por passos. [13]

2.10.1 Tipos construtivos

De acordo com [13], os motores de passo são classificados em relação ao seu tipo construtivo, e podem ser de três tipos: relutância variável, ímã permanente e híbridos com escovas redundantes, explicar-se-á com ênfase o de ímã permanente (Figura 17), utilizado nesse projeto. Esse é similar ao motor de relutância variável, porém o rotor é construído com ímãs permanentes e não possui dentes, o que determina uma característica importante deste tipo, que é a de manter a última posição mesmo quando não energizado, conhecido como torque de residual. Capacidade máxima de carga estática que pode ser aplicada ao eixo sem que haja rotação contínua [13].

Figura 17 - Motor de Passo – Ímã Permanente.



Fonte: [14] (p. 09).

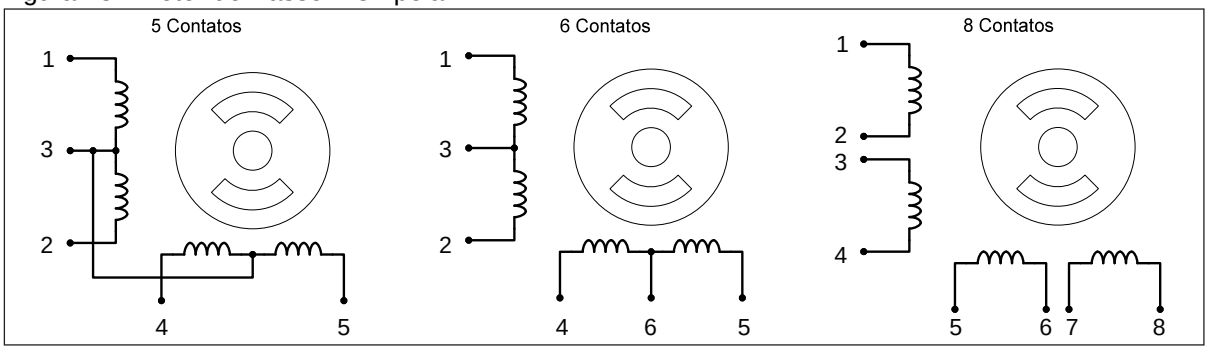
Segundo [15], a vantagem desse tipo de motor é o fato dele ter um campo magnético permanente que se soma ao campo magnético das bobinas, dando uma potência, ou torque, maior na partida. A desvantagem é o fato de possuírem um passo maior, com menor precisão. Entretanto pode ser aumentada através do aumento do número de polos no rotor ou aumento do número de fases [15].

2.10.2 Polos

Motores de passo geralmente têm duas fases, e podem ser unipolares ou bipolares [15].

Unipolares (Figura 18) - são utilizados dois enrolamentos por fase e costumam ter um contato em comum, resultando em cinco, seis ou oito conexões. Nos modelos onde a conexão comum dos dois polos é separada, são seis conexões externas e nos modelos onde a conexão comum é soldada internamente, são cinco conexões externas. Os de oito conexões externas contêm a conexão em comum dos dois polos separada e facilitam a ligação em série ou paralela das bobinas. Os modelos com cinco ou seis conexões têm as bobinas ligadas em série e necessitam da capacidade de reverter às ligações entre as bobinas [15].

Figura 18 - Motor de Passo – Unipolar.

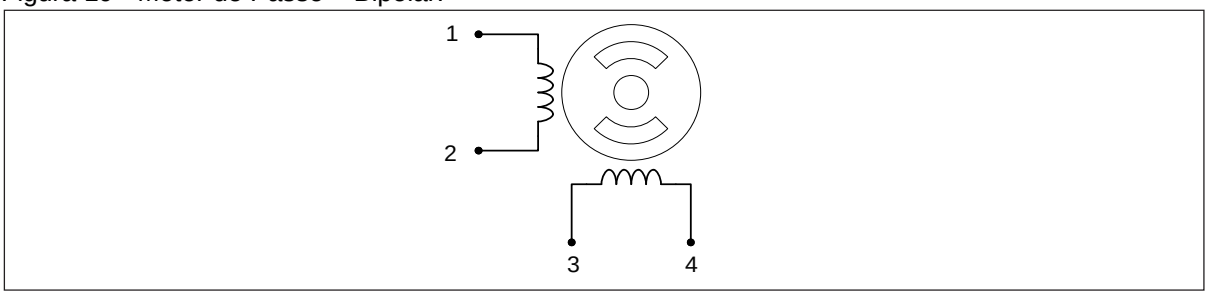


Fonte: [16] (p. 12).

Ligação reversa é um tipo de ligação muito comum entre motores onde os polos A e B das bobinas podem ser ligados ao positivo e negativo respectivamente, ou invertida, negativo e positivo respectivamente [15].

Bipolares (Figura 19) - utilizam uma ligação por polo e necessitam que o circuito de controle possa reverter o sentido da corrente para acionar as bobinas de forma correta. Chegam a apresentar 40% a mais de torque comparado a motores unipolares de mesmo tamanho, pois quando se energiza uma fase, magnetizam-se ambos os polos em que a fase está instalada, agindo sobre o rotor forças magnéticas de ambos os polos, sendo bastante considerada essa característica na aplicação nesse projeto [15].

Figura 19 - Motor de Passo – Bipolar.



Fonte: [16] (p. 13).

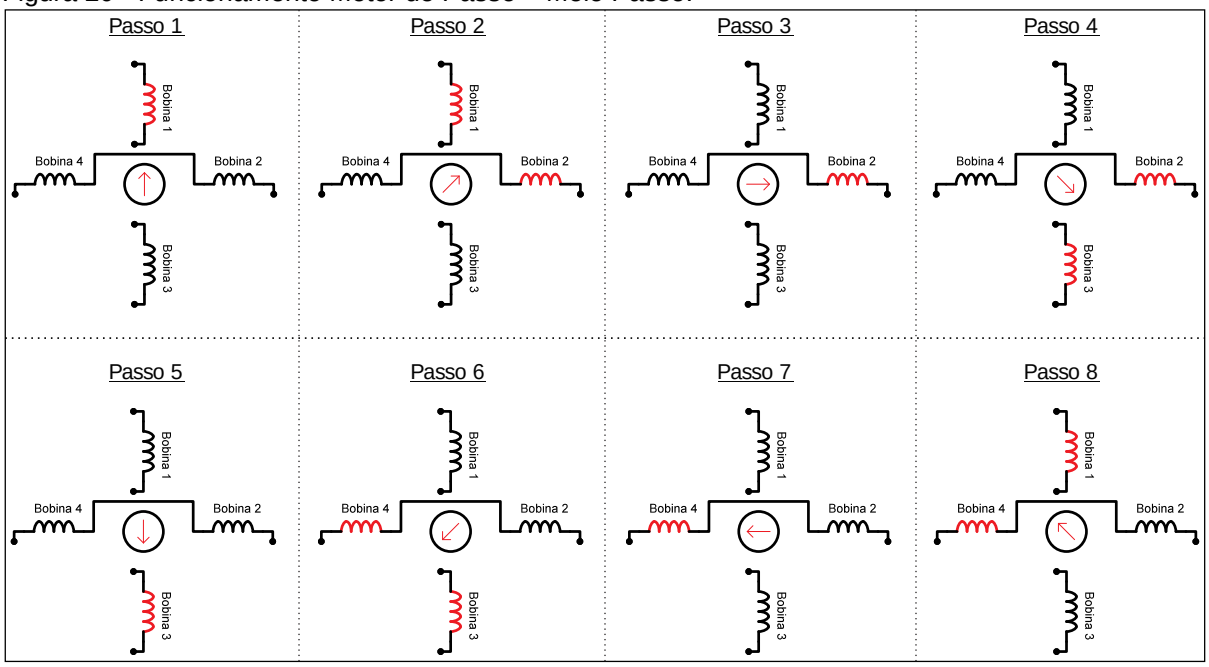
2.10.3 Tipos de ligações

Wave drive (passo completo 1) - liga uma bobina por vez, reduzindo o consumo de energia, porém perde torque [15].

Full drive (passo completo 2) – liga duas bobinas por vez, consome mais energia apresentando maior torque [15].

Half drive (meio passo - Figura 20) – alterna a ligação entre a *wave* e *full* (Tabela 2), dobrando a quantidade de passos necessários para o motor girar 360º, apresenta maior torque, mas com velocidade reduzida [15].

Figura 20 - Funcionamento Motor de Passo – Meio Passo.



Fonte: [16] (p. 15).

Tabela 2 - Mapa lógico – Meio Passo.

B - Bobina	UNIPOLAR					BIPOLAR		
Nº do passo	B1	B2	B3	B4	B1	B2	B3	B4
1	1	0	0	0	+1	-1	-1	+1
2	1	1	0	0	+1	-1	0	0
3	0	1	0	0	+1	-1	+1	-1
4	0	1	1	0	0	0	+1	-1
5	0	0	1	0	-1	+1	+1	-1
6	0	0	1	1	-1	+1	0	0
7	0	0	0	1	-1	+1	-1	+1
8	1	0	0	1	0	0	-1	+1

Fonte: [17] (p. 03).

Para que o motor gire no sentido anti-horário basta aplicar os pulsos de acionamento das bobinas de forma inversa, ou seja, de trás para frente. Também há

os micros passos, que envolvem uma interpolação entre as posições de passo completo e meio passo. Obtido através do controle linear das correntes de acionamento dos estatores, aplicando-se sinal PWM. Resultam em grande precisão e operação suave em baixas velocidades [15].

Para o projeto foram utilizados dois motores de passo retirados de sucatas de impressoras, ambos configurados para funcionarem como bipolar. Com os seguintes modelos PM55L-048 e 55SPM25D7ZA1 (fabricantes diferentes, mas mesmas especificações), que possuem 7,5º por passo completo ou 3,75º por meio-passo, operando então com 48 ou 96 passos por volta, respectivamente [18].

2.11 Servo-Motor

Segundo [3], são motores de corrente-contínua acoplados à redutores de velocidade junto com um sensor de posição, possuindo um sistema de controle realimentado (Figura 21). Geralmente são pequenos e possuem ampla variação de torque, o seu sistema de realimentação é independente do controlador. Possuindo então três fios, dois de alimentação e um para envio de sinal PWM, que pode efetuar o controle da sua posição e torque, enquanto esse sinal é enviado, o eixo do motor se mantém inalterado. Devido ao redutor existente no eixo apresentam ótima precisão [3]. Utilizou-se no projeto dois micro servos-motores do modelo MG90S, que apresentam alimentação nominal de 4 a 6 V, quanto mais próximo da faixa de tensão máxima maior é o torque, e esse é de até 2 kg/cm [19].

Figura 21 - Servo-motor MG90S.

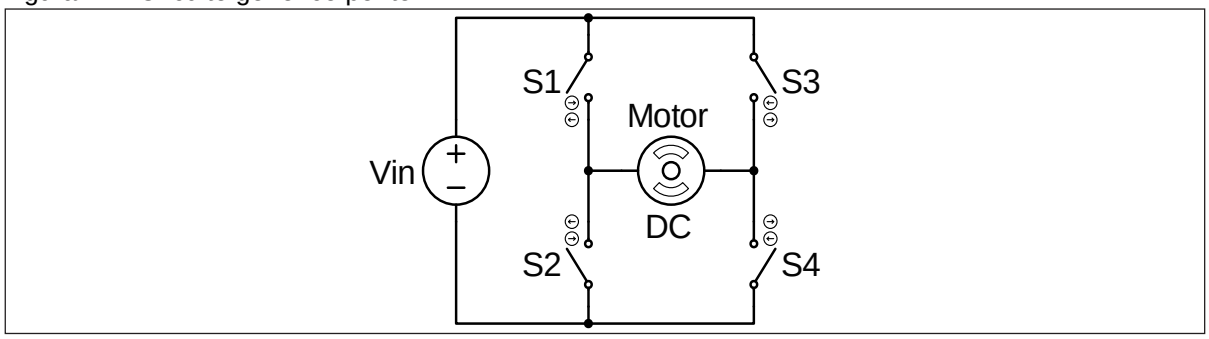
Fonte: [19] (p. 1).

2.12 Ponte H

De acordo com [20], é um circuito que converte uma fonte de corrente continua fixa, em uma tensão de corrente continua variável, abrindo e fechando diversas

vezes. Portanto, pode determinar o sentido da corrente, a polaridade da tensão e a tensão em um dado sistema ou componente. Seu funcionamento se dá pelo chaveamento de componentes eletrônicos usualmente utilizando do método de PWM para determinar além da polaridade, o módulo da tensão em um dado ponto de um circuito (Figura 22). Tem como principal função o controle de velocidade e sentido de motores DC escovados, podendo também ser usado para controle da saída de um gerador DC ou como inversor monofásico. O termo Ponte H é derivado da representação gráfica típica deste circuito, conforme mostrado na Figura 22 [20].

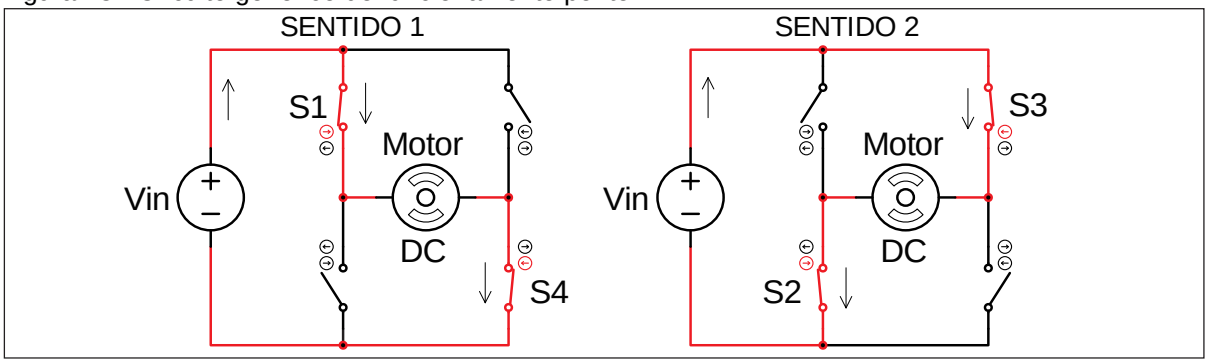
Figura 22 - Circuito genérico ponte H.



Fonte: Autor.

O circuito de ponte H é usado para determinar um sentido de corrente e valor de tensão no controle de um motor DC, o diagrama da Figura 23 ilustra de modo genérico o seu funcionamento. Acionando-se em conjunto, as chaves S1 e S4, o terminal esquerdo do motor fica com uma tensão mais positiva que o direito, fazendo a corrente fluir da esquerda para a direita. Deste modo, o motor adquire sentido de giro chamado Sentido 1 [20].

Figura 23 - Circuito genérico de funcionamento ponte H.



Fonte: Autor.

Acionando-se em conjunto as chaves S3 e S2, o terminal direito do motor fica com uma diferença de potencial maior que o terminal esquerdo, fazendo a corrente fluir da direita para a esquerda. Deste modo, o motor adquire sentido de giro chamado Sentido 2, que é inverso ao Sentido 1 [20].

Ao acionar em conjunto as chaves S1 e S3 ou S2 e S4 é provocado um curto nos terminais do motor. Isso é necessário quando se deseja frear um motor já em movimento ou aumentar a dificuldade de giro do eixo por um fator externo. Tal efeito é alcançado, pois a máquina DC passa a se comportar como um gerador quando tem seu eixo em movimento, tanto no caso da rotação, quanto no caso do giro do eixo por fator externo. Ao se gerar um curto circuito entre os terminais da máquina nesse estado, o torque necessário para manter ou colocar o motor em giro cresce, visto a necessidade de corrente exigida da máquina para seu movimento, o que causa o efeito chamado freio motor [20].

As chaves S1 e S2 não podem ser fechadas simultaneamente assim como as chaves S3 e S4. Pois o fechamento em conjunto de tais chaves causaria um curto na fonte de alimentação. Pode-se fazer o uso de PWM nas chaves para controlar a tensão média aplicada sobre o motor, e assim, controlar a velocidade da máquina DC [20].

No projeto a ponte H foi utilizada no controle do motor de passo bipolar, pois neste não é possível realizar a inversão do sentido da rotação de forma direta. Por meio do circuito ponte H é possível realizar a inversão do sentido da corrente nos motores bipolares.

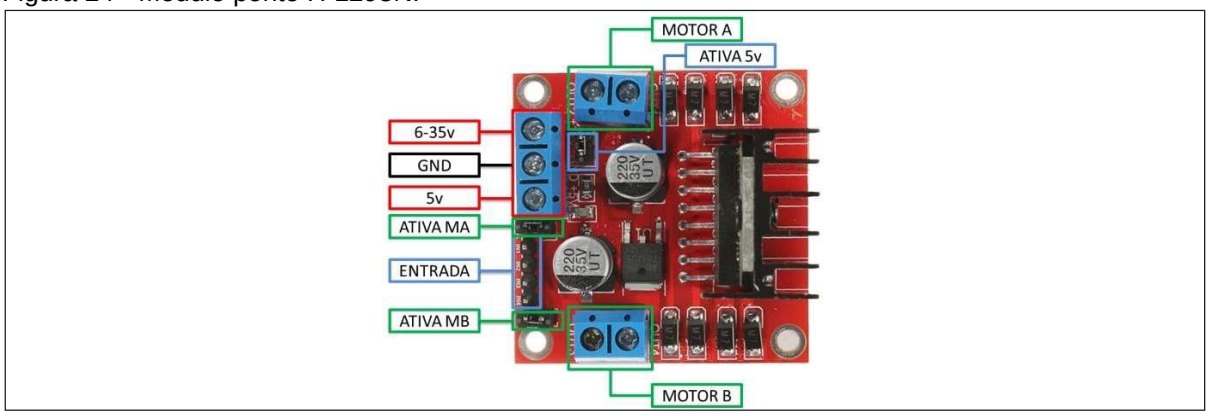
2.12.2 Módulo com L298N

No projeto foi utilizado o CI L298N, muito utilizado para o propósito de controle de motores, trata-se de uma ponte H em um componente integrado, e demais componentes de proteção, alimentação e controle. Possuindo as seguintes especificações [21]:

- tensão de operação: 4 ~ 35 V;
- controle de 2 motores DC ou 1 motor de passo;
- corrente de operação máxima: 2 A por canal ou 4 A máximo;
- potência máxima: 25 W;
- tensão lógica: 5 V;
- corrente lógica: 0 – 36 mA;
- dimensões: 43x43x27 mm;
- peso: 30 g.

Uma das vantagens do uso deste CI é o menor espaço ocupado, a baixa complexidade do circuito e o fato de ele já possuir dois circuitos H, podendo assim, controlar dois motores DC ou um motor de passo (Figura 24) [21].

Figura 24 - Módulo ponte H L298N.



Fonte: Autor.

Motor A e Motor B: conectores para os motores, 2 DC ou 1 de passo;

6 - 35 V: porta para alimentação da placa com tensão entre 6 a 35 V;

GND: porta conectada na referência/terra dos circuitos;

Ativa 5 V: quando *jumpeado* e sendo alimentado pela porta 6 – 35 V, a placa utilizará o regulador de tensão integrado para fornecer 5 V (na porta 5 V). Neste caso, não se deve alimentar a porta 5 V pois pode danificar os componentes;

5 V: em casos de não haver fonte de alimentação com mais de 6V se pode alimentar a placa com 5 V por esta porta;

Ativa MA: quando *jumpeado* aciona o motor A com velocidade máxima. Para controlar a velocidade do motor A basta remover o *jumper* e alimentar o pino com uma tensão entre 0 e 5 V, onde 0 V é a velocidade mínima (parado) e 5 V a velocidade máxima;

Ativa MB: quando *jumpeado* aciona o motor B com velocidade máxima. Para controlar a velocidade do motor B basta remover o *jumper* e alimentar o pino com uma tensão entre 0 e 5 V, onde 0 V é a velocidade mínima (parado) e 5 V a velocidade máxima;

Entrada: barramento composto por IN1 e IN2, utilizados para controlar o sentido do motor A. E, IN3 e IN4, utilizados para controlar o sentido do motor B, conectados com as portas do microcontrolador [21].

Verifica-se que agora no lugar das chaves S1 - S3 e S2 - S4, tem-se os pinos IN1 e IN2, respectivamente correspondentes. Para controlar o sentido, temos as seguintes combinações para o motor A (IN1 e IN2), conforme Tabela 3 [21].

Tabela 3 - Lógica combinacional - Ponte H L298N.

IN1 [V]	IN2 [V]	Estado
0	0	Desligado
0	1	Sentido 1
1	0	Sentido 2
1	1	Freio

Fonte: [21].

A lógica do motor B (*IN3* e *IN4*) segue a mesma da Tabela 3.

2.13 Equações

Na determinação do número de passos do motor, utiliza-se a Equação 3 [17].

$$Passos = \frac{360^{\circ}}{\beta} \quad (3)$$

Para transmissão de potência do eixo do motor para os seus respectivos elos são utilizadas engrenagens, onde a relação de transmissão é dada pela Equação 4 [3].

$$n = \frac{N_{out}}{N_{in}} \quad (4)$$

O cálculo da velocidade de saída é dado pela Equação 6 [3].

$$\omega_{in} = n * \omega_{out} \quad (5)$$

E o cálculo do torque de saída de transmissão é dado pela Equação 7 [3].

$$T_{out} = n * T_{in} \quad (6)$$

3. METODOLOGIA

Esse capítulo descreve o projeto e operação do braço robótico, apresenta uma descrição detalhada do trabalho, dos conceitos e definições de funcionamento. Visa explicar de forma clara para possíveis replicações dessa metodologia em outros projetos, e mostrar dificuldades encontradas. Assim como demonstrar os resultados obtidos, com as soluções encontradas para os problemas enfrentados, e possibilitar a aplicação em automação de processos industriais e auxílio didático para o ambiente acadêmico.

O capítulo está dividido em 4 grupos (etapas), onde cada um representa os processos de criação do projeto, existindo dentro subgrupos que objetivam melhorar o detalhamento desse.

Que são as seguintes:

- Etapa 1 – Estudo e viabilidade do braço robótico
- Etapa 2 – Desenvolvimento do *firmware*
- Etapa 3 – Simulação computacional
- Etapa 4 – Prototipação e testes em bancada

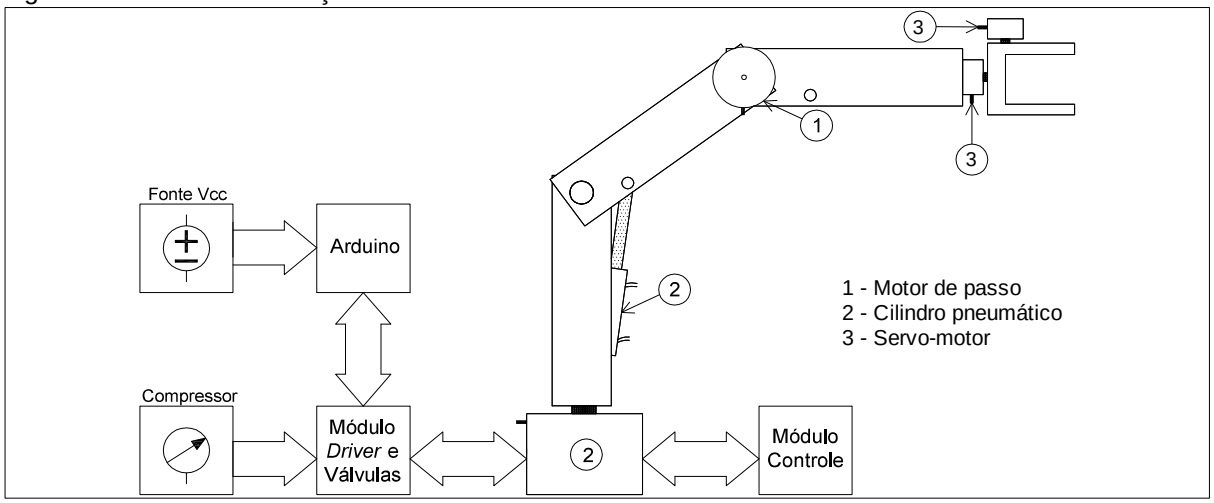
3.1 Etapa 1: Estudo e viabilidade do braço robótico

Com o tema do projeto definido se iniciaram as pesquisas de funcionamento e construção de braços robóticos, conforme apresentado no capítulo 2. De posse dessas informações foi possível idealizar o projeto, iniciando-se assim as primeiras definições de construção.

O braço robótico foi construído com quatro atuadores elétricos (2 motores de passo e 2 servos-motores) e dois pneumáticos (2 cilindros), possuindo quatro graus de liberdade, e com o microcontrolador ATmega2560 embarcado no sistema Arduino MEGA 2560 R3, como central de processamento. A ligação do sistema embarcado com os motores de passo é por intermédio de dois módulos ponte H, e com os cilindros por três válvulas solenoide, os servos são conectados diretamente. Entre os cilindros e as válvulas solenoide foram utilizadas seis válvulas de retenção para controle de vazão do ar comprimido. Para a comunicação do operador com o braço se utilizou um módulo de controle (*joystick*), contendo *knobs* e chaves impulso (*push button*). O braço possui alimentação externa, utilizando-se assim uma fonte de

tensão de 12 V e um compressor de ar, para os elementos elétricos e pneumáticos respectivamente. A estrutura/esqueleto possui peças de madeira projetadas especificamente para ele, na Figura 25 é apresentado o seu sistema completo.

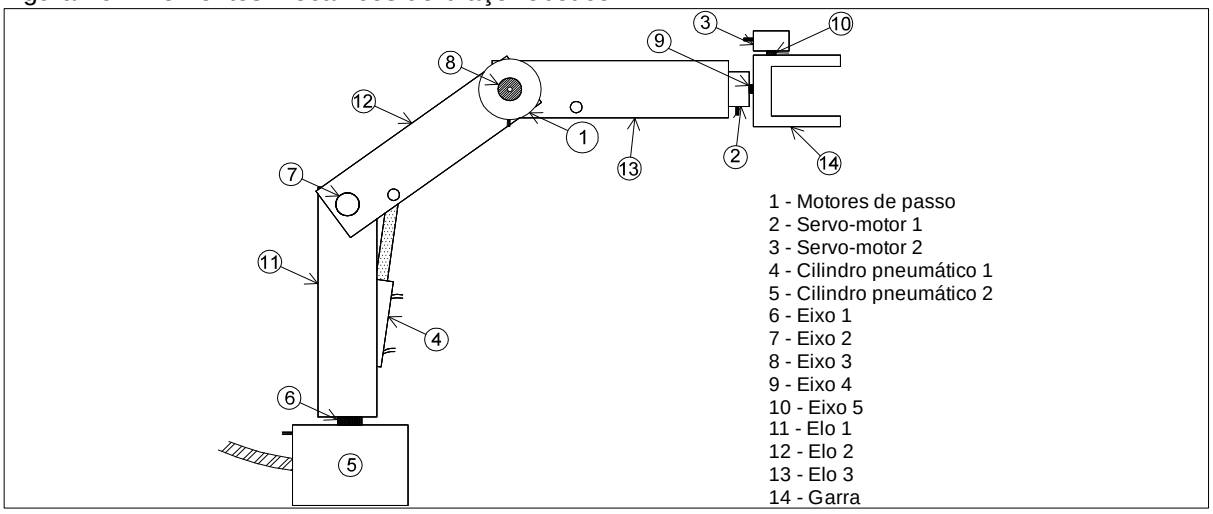
Figura 25 - Sistema do braço robótico.



Fonte: Autor.

Na base são utilizados dois cilindros, número 5 da Figura 26, permitindo através de uma lógica mecânica o movimento em três pontos fixos. A base é acoplada com a coluna através do eixo 1. A coluna é conectada com o elo 1 através do eixo 2, também chamado ombro, sendo o elo 1 movimentado pelo cilindro de elevação. Esse cilindro realiza a elevação ou abaixamento vertical do braço, alterando sua altura de alcance de operação em dois pontos fixos. O eixo 3, também chamado cotovelo, conecta o elo 1 com o elo 2, e o movimento do elo 2 é realizado por dois motores de passo, número 1, localizados no cotovelo e realizam o giro de até 270º do braço robótico, possibilitando um alcance de trabalho maior, compensando o pouco alcance dos cilindros. O elo 2 é conectado com a garra através do eixo 4, também chamado punho, que é comandado pelo servo 2. Esse realiza o giro 180º do punho possibilitando ajuste de posição conforme o controle. E a garra é controlada pelo servo 3, que realiza a abertura (até 90º) e fechamento (conforme carga), conectados através do eixo 5.

Figura 26 - Elementos mecânicos do braço robótico.



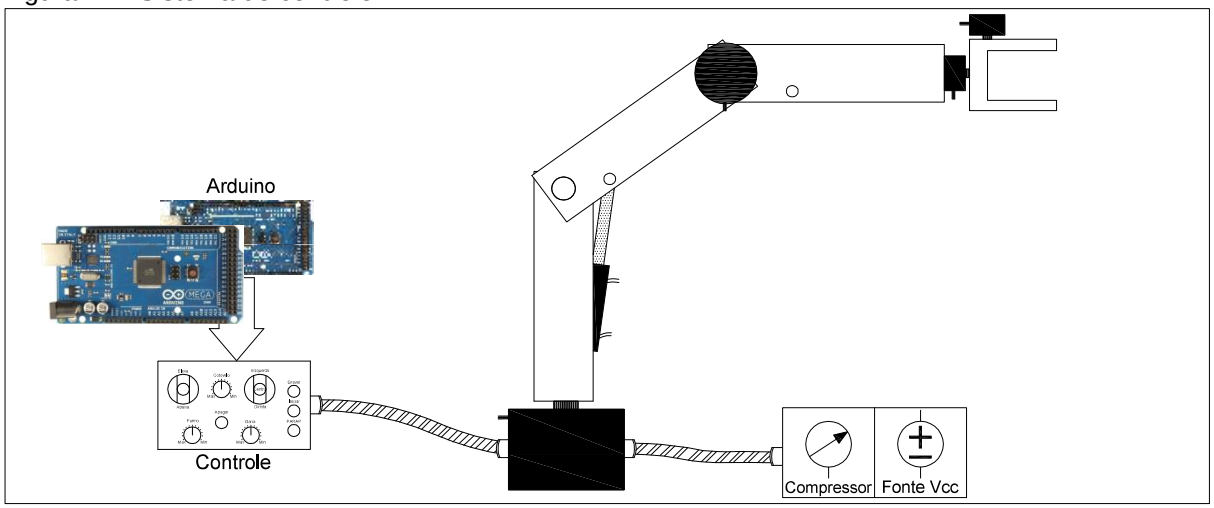
Fonte: Autor.

No cotovelo foram utilizados os motores de passo de ímã permanente PM55L-048 e 55SPM25D7ZA1, que possuem características equivalentes, proporcionando assim maior força nesse elo, no punho e garra se utilizou o servo-motor MG90S.

3.1.1 Princípio funcionamento

O braço robótico funciona da seguinte forma, o operador de posse do módulo de controle utilizará os *knobs* para movimentar os motores e as chaves os cilindros, conforme Figura 27. Inicialmente moverá um atuador por vez, posicionando conforme o desejado para pegar e alterar de posição a peça almejada. Após cada movimento é necessário que o operador pressione o botão Gravar para que a posição seja armazenada na memória EEPROM do sistema embarcado, criando então um ciclo de trabalho. Caso o operador não realize a gravação de cada movimento, o braço estará sendo executado no modo manual. Considerando que o objetivo do operador seja o modo automático, na qual o braço executa os movimentos de forma repetitiva, ao fim do processo quando o operador pressionar o botão Iniciar o braço executará os movimentos gravados, conforme ordem de gravação. Ficará nesse ciclo até o momento que for pressionado o botão Parar ou o Arduino seja desligado da energia. Como os movimentos são armazenados na memória EEPROM, quando o sistema for desligado e religado o ciclo de trabalho se repetirá assim que for pressionado o botão *Play*.

Figura 27 - Sistema de controle.

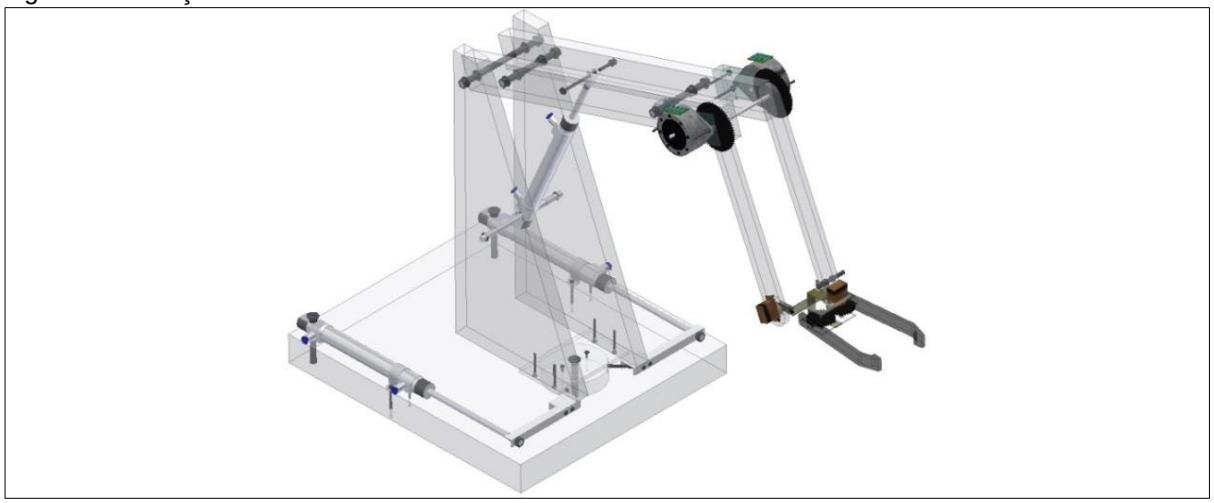


Fonte: Autor.

3.1.2 Estrutura mecânica

A estrutura do braço foi projetada seguindo como exemplo braços guindastes e robóticos existentes no mercado, guindaste devido ao atuador pneumático. Foi desenvolvido no CAD o desenho 2D e 3D do braço, para fim de analisar a sua mecânica e mostrar de forma detalhada a sua construção. No desenho computacional o projeto foi criado utilizando chapas em acrílico (Figura 28), material ideal para esse caso, entretanto no protótipo se utilizou chapas de madeira, obtidas de sucatas de uma madeireira local. Para fim de testes era o material mais acessível e de baixo custo disponível.

Figura 28 - Braço robótico CAD.



Fonte: Autor.

Efetuou-se a montagem conforme os atuadores utilizados, mantendo-se a escala proporcional à capacidade desses itens, na Figura 29 são apresentadas algumas fotos das etapas do projeto mecânico.

Figura 29 - Etapas processo de montagem.

Fonte: Autor.

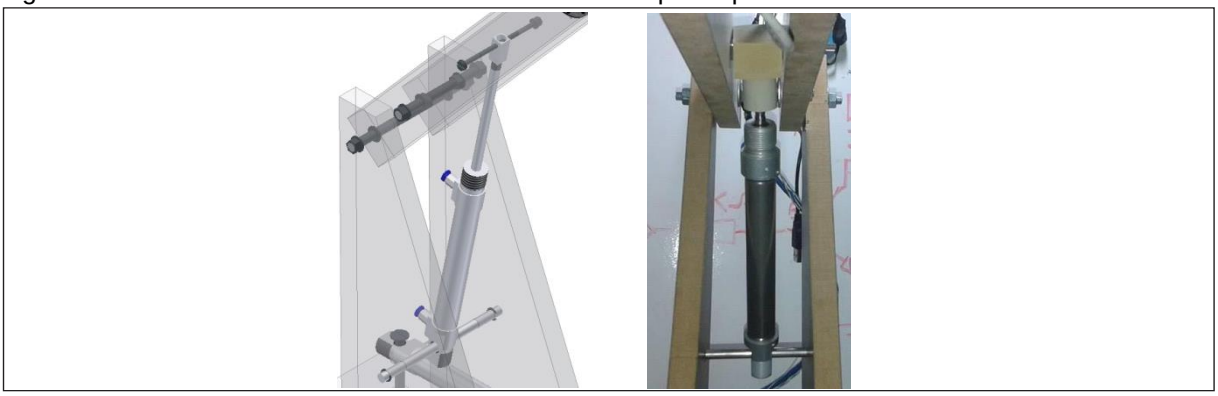
Na base para efetuar o movimento de giro foi projetada uma chapa circular de diâmetro de 80 mm e fixado dois pinos em sua lateral (Figura 30). Conforme a posição onde se fixam os pinos, é possível aumentar ou diminuir a amplitude do movimento de giro do braço, no projeto é possível atingir no máximo 120º (lado esquerdo e direito). Na haste dos cilindros é fixada uma chapa metálica projetada para efetuar a transferência de movimento do cilindro para a base circular.

Figura 30 - Sistema de giro da base no CAD e protótipo.

Fonte: Autor.

O cilindro do ombro é sustentado por um pino liso que permite o giro natural do atuador em seu eixo (Figura 31), para não restringir o movimento mecânico do cilindro.

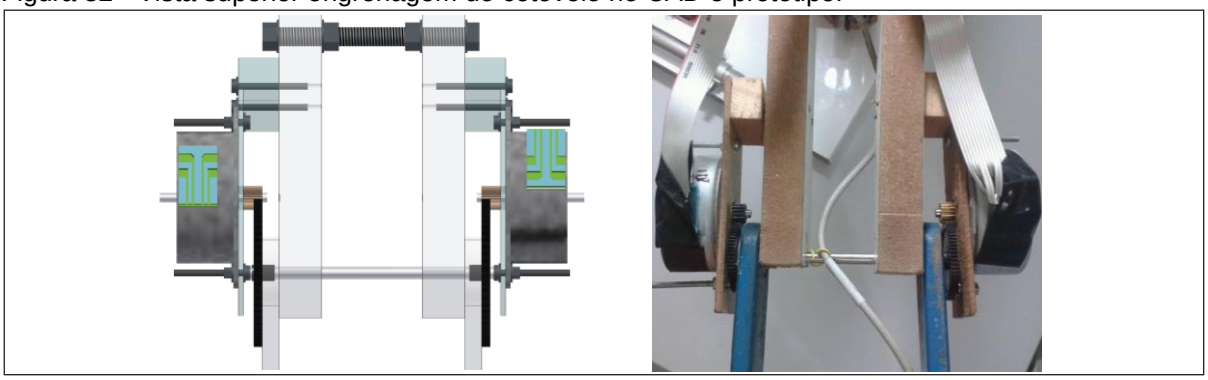
Figura 31 - Sistema de movimento do ombro no CAD e protótipo.



Fonte: Autor.

No cotovelo foram utilizadas engrenagens para transferir o movimento do eixo do motor de passo para o antebraço (Figura 32). Esse ajuste é necessário devido à necessidade de aumento do torque, obtendo-se através da transferência de movimento de uma engrenagem de diâmetro menor para uma de maior, onde se perde velocidade, mas aumenta o torque, conforme se pode observar com as Equações 4, 5 e 6 (seção 2.13).

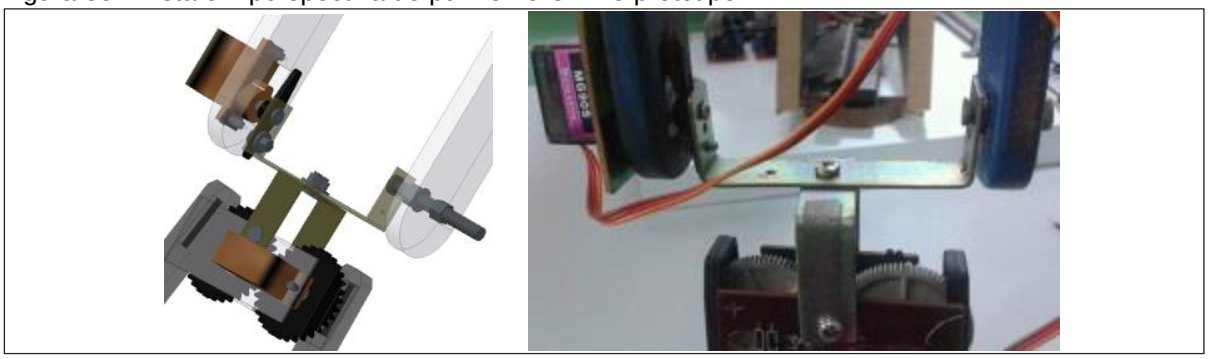
Figura 32 - Vista superior engrenagem do cotovelo no CAD e protótipo.



Fonte: Autor.

Na Figura 33 é mostrada montagem do punho, com o servo-motor colocado lateralmente à estrutura metálica.

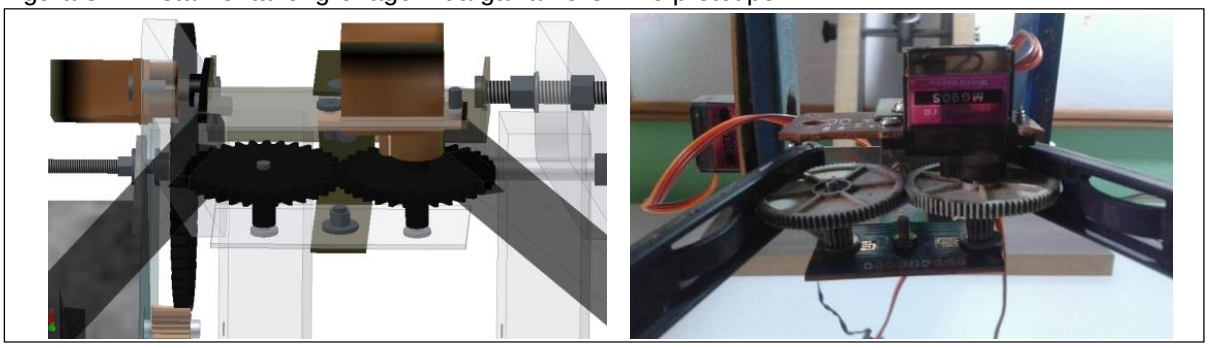
Figura 33 - Vista em perspectiva do punho no CAD e protótipo.



Fonte: Autor.

Na garra também foram utilizadas engrenagens, entretanto foi para o fim de facilitar o movimento de abrir e fechar da garra (Figura 34). Não há aumento de torque nem redução da velocidade, pois o eixo do servo-motor está acoplado diretamente à engrenagem, e essas possuem mesmo diâmetro.

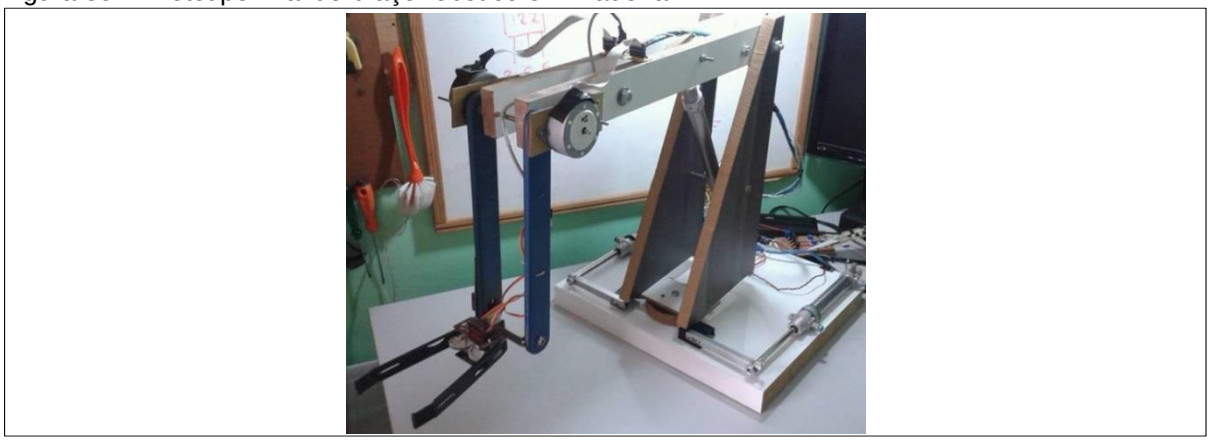
Figura 34 - Vista frontal engrenagem da garra no CAD e protótipo.



Fonte: Autor.

A Figura 35 apresenta a estrutura completa do protótipo do braço robótico, e no Apêndice E se encontram as medidas das peças.

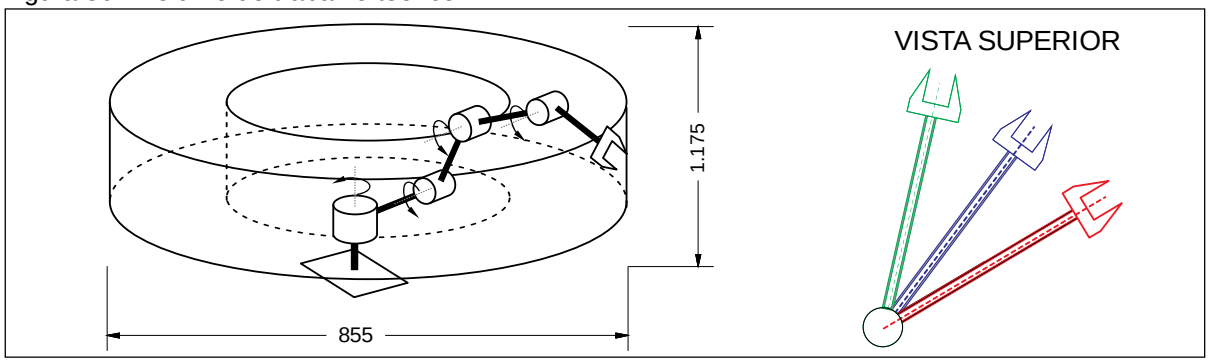
Figura 35 – Protótipo final do braço robótico em madeira.



Fonte: Autor.

Com essa estrutura o braço robótico atinge na vertical até 1.175 mm desde a sua base até a ponta da garra, e na horizontal 855 mm da sua coluna até a ponta da garra. Na Figura 36 é mostrado o volume de trabalho teórico do braço, que é o espaço onde o manipulador consegue posicionar a sua garra [16].

Figura 36 - Volume de trabalho teórico.



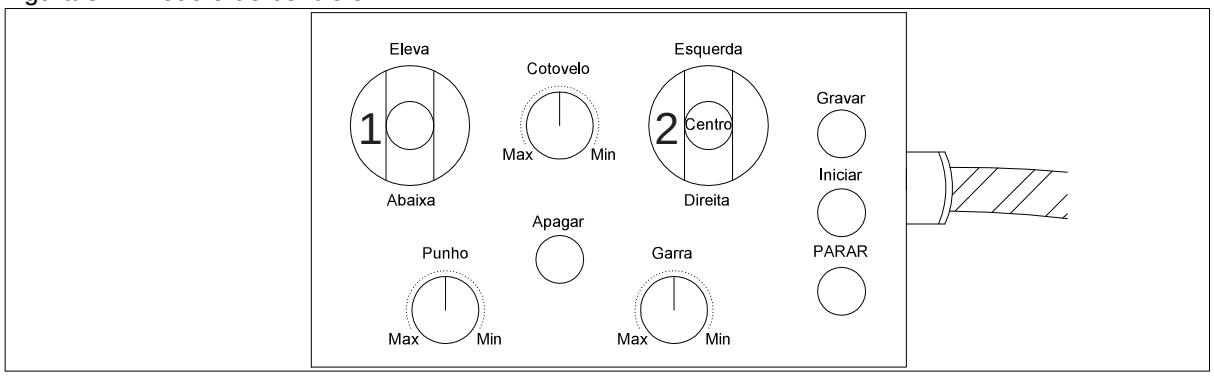
Fonte: Autor.

Entretanto por se tratar de volume teórico, é preciso atentar que como o seu movimento na base se restringe a três movimentos, esquerda, centro e direita, o braço atingirá somente essas zonas fixas de movimento, representadas pelas garras em verde, azul e vermelho, respectivamente. Nos Apêndices D, E e F são especificadas de forma detalhada as zonas de operação, desenhos da estrutura do braço e lista de peças utilizadas, respectivamente.

3.1.4 Módulo de controle

Visando a comunicação do meio externo com o sistema embarcado foi criado um módulo de controle (*joystick*), também chamado interface homem-máquina (Figura 37). Através desse módulo é possível que um operador controle o braço em tempo real, realizando os movimentos e controlando início e fim do ciclo de trabalho.

Figura 37 - Módulo de controle.



Fonte: Autor.

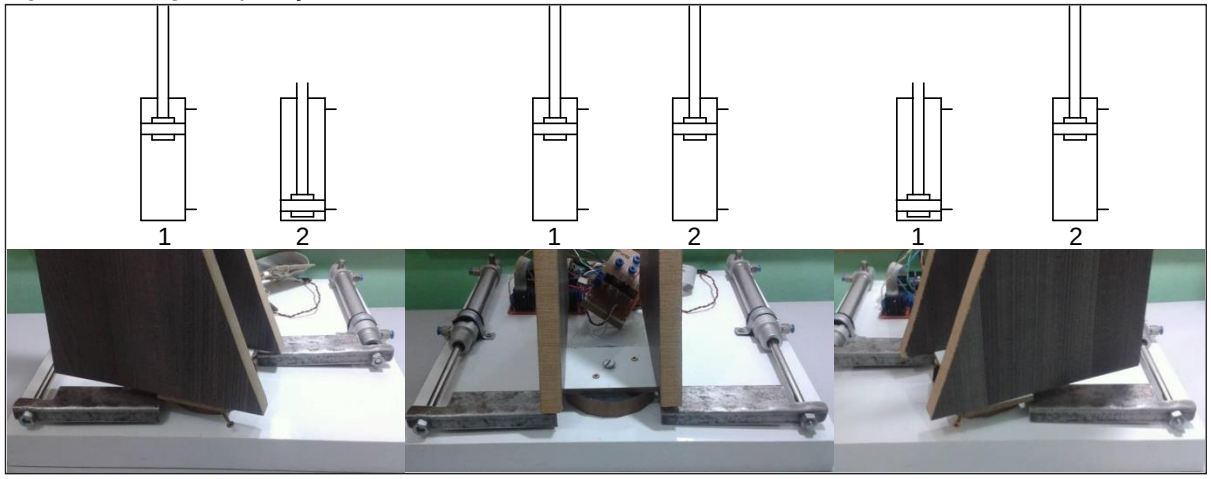
O módulo é composto por duas chaves de posição, três *knobs* e quatro botões, que controlam os cilindros, motores e modo de operação, respectivamente.

A chave 1 controla o movimento vertical do manipulador através do cilindro de elevação, se o operador mover a chave para cima será realizada a elevação, caso seja pressionada para baixo o movimento é de retração, abaixando. Ao realizar o movimento para elevar será colocado nível alto na válvula solenoide avançando o cilindro, e caso seja pressionado para abaixar será colocado nível baixo, recuando o cilindro. Então, se o pistão já estiver avançado e for pressionado para elevar, o atuador não muda seu estado, mantendo a posição elevada, o mesmo vale no caso de estar recolhido e for ajustado nível baixo. Esse comando controla a altura de trabalho vertical do sistema, assemelhando-se ao ombro humano.

A chave 2 controla o movimento horizontal do braço, os cilindros da base possuem estado inicial avançado quando o sistema do braço robótico é ligado. Se o operador pressionar direita o pistão 1 avança e o pistão 2 recua. Se pressionar para centro o pistão 1 e pistão 2 vão para o estado avançado, e pressionando esquerda o pistão 1 recua e o pistão 2 avança. Entretanto caso o braço esteja na posição direita e seja pressionado para esquerda, primeiro o braço irá para a posição centro, e após para a posição esquerda, utilizando mesma lógica da esquerda para direita

(Figura 38). Como foram utilizadas peças mecânicas, esse ajuste foi necessário para que não houvesse atrito no movimento e fosse mecanicamente realizável a operação. Os pistões restringem o alcance de trabalho do braço, mas com essa lógica mecânica foi possível atingir até 120º em três posições fixas.

Figura 38 - Lógica operação cilindros base.



Fonte: Autor.

O *knob* Cotovelo controla o movimento do cotovelo na vertical através da movimentação dos motores de passo, conforme o operador realiza o giro do *knob* o motor gira na mesma graduação. Essa graduação é configurada no *firmware*, onde se considera o número de passos do motor se adequando a operação como divisor de tensão realizada pelo potenciômetro. Então à medida que o operador gira o *knob* o motor realiza o movimento no mesmo sentido, possuindo 270º de liberdade, o giro vai de 45º a 315º e de 315º a 45º. Sendo necessária a limitação desses 45º para respeitar a estrutura física, do contrário ocorreria colisão dos motores com o elo 2.

O *knob* Punho realiza o movimento na vertical da garra através do giro do servo-motor 1, o funcionamento é o mesmo do *knob* do Cotovelo. Entretanto possui 180º de liberdade de operação, ou seja, o motor realiza o giro de 90º a 270º e de 270º a 90º.

O *knob* Garra controla a posição de abertura ou fechamento da garra, cabe ao servo-motor 2 essa graduação. O funcionamento é o mesmo dos anteriores, Cotovelo e Punho, possui abertura máxima de 90º e o fechamento verificado visualmente no manuseio com a carga.

O botão Gravar realiza a gravação do movimento realizado, que é necessário na criação de uma rotina repetitiva de trabalho. Ao pressionar esse botão, o dado atual, no caso o estado dos atuadores, é armazenado na memória EEPROM do sistema embarcado. Ou seja, ao realizar um movimento o operador deverá pressionar o botão Gravar para armazenar esse movimento. Enquanto não for pressionado, o movimento realizado não é gravado, podendo o operador ajustar livremente o braço e somente após chegar ao ponto desejado realizar a gravação. Caso sejam realizados vários movimentos sem que esse seja pressionado, o braço robótico estará operando no modo manual.

O botão Iniciar quando pressionado realiza os movimentos gravados de forma automática, repetindo o ciclo de trabalho criado pelo operador. Os movimentos são efetuados exatamente na ordem de gravação, a leitura da memória EEPROM é realizada no formato FIFO, ou seja, o primeiro movimento gravado será o primeiro a ser executado.

O botão Parar quando pressionado para imediatamente o movimento do braço robótico, utilizado em casos que o operador queira alterar o ciclo de trabalho ou de emergência. É um botão conectado diretamente na entrada de interrupção externa do Arduino, que é uma entrada que quando levada a nível alto interrompe todos os processamentos do micro controlador.

O botão Apagar quando pressionado limpa o *buffer* da memória EEPROM. Esse procedimento é necessário quando é criado novo ciclo de trabalho, pois como os dados são armazenados na memória não-volátil eles permanecem gravados enquanto não forem sobrepostos ou apagados, podendo repetir movimentos de um ciclo de trabalho anterior. Então, assim que pressionado é realizada a limpeza total da memória, porém é necessário que o Arduino esteja ligado.

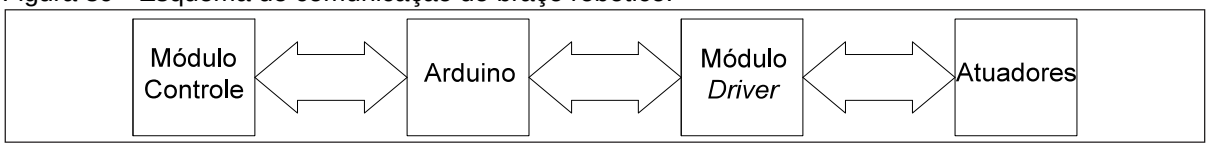
3.2 Etapa 2: Desenvolvimento do *firmware*

O *firmware* realiza os processos internos do sistema embarcado, verificando as portas e suas variações de sinal, interpretando e executando conforme a lógica criada (Figura 39). A linguagem de programação utilizada foi o C++, que possui total compatibilidade com o sistema embarcado, e também pela simplicidade de programar. O Arduino possui duas funções principais, a *main* e o *loop*. Na primeira

são ajustadas as condições iniciais de inicialização do sistema embarcado e na segunda o programa fica sendo executado repetidamente.

A lógica do programa é analisar repetidamente no *loop* a alteração do estado das variáveis atuadores. Esse processo é feito se verificando as portas utilizadas para comunicação com o módulo de controle. A cada alteração dos *knobs*, chaves com retorno e botões, o código interpreta esses dados, e envia sinais para o módulo driver dos atuadores, por sua vez esses *drivers* irão alterar o estado dos atuadores conforme o sinal recebido.

Figura 39 - Esquema de comunicação do braço robótico.



Fonte: Autor.

Após a energização do sistema as variáveis do *firmware* são inicializadas, e então entra na função do *loop* de verificação. Nessa função o programa analisa as entradas ajustadas do Arduino, mencionadas anteriormente. Então, por exemplo, caso o *knob* Cotovelo seja girado o código fará o tratamento desse sinal, e enviará para o driver do motor, ponte H, e esse acionará o motor de passo.

O botão Parar interrompe por completo o ciclo do programa, sendo utilizado como botão de segurança.

No Apêndice A é apresentado o fluxograma do *firmware* criado. Com o fluxograma se pôde iniciar a programação por partes do *firmware*, realizando os testes em separado de cada atuador e função.

No código abaixo está exemplificado uma rotina de inicialização dos parâmetros do programa.

```
#include <biblioteca.h>     // adiciona biblioteca necessária ao código
#define valor1              // seta uma variável
boolean variavel;           // seta uma variável booleana
int valor2;                 // seta uma variável inteira
char valor3                 // seta um valor sem retorno
void setup() {              // laço inicial do Arduino
```

```
}
void loop() {                    // laço de operação do programa
}
```

Para controlar os motores de passo é utilizada a seguinte rotina existente na IDE do *Arduino*.

```
#include <Stepper.h>        // adiciona a biblioteca do motor de passo
#define STEPS 100           // seta o número de passos do motor
Stepper stepper(STEPS, 8, 9, 10, 11);   // pinos de ligação com motor
int previous = 0;                       // variável analógica inicial,
void setup()
{
  stepper.setSpeed(30);     // velocidade do motor em rpm
}
void loop()
{
  int val = analogRead(0);            // sinal da porta analógica é lido
  stepper.step(val - previous);       // motor gira conforme valor subtraído
                                      // do valor atual lido menos o anterior
  previous = val;       // o valor atual é armazenado na variável anterior
}
```

Na rotina de controle dos cilindros se utiliza a seguinte lógica:

```
#define sensorVal1      2               // define a porta de ligação física
#define sensorVal2      3               // dos sensores de posição
void setup() {
  pinMode(2, INPUT_PULLDOWN);  // entrada em nível baixo sensor 1
  pinMode(3, INPUT_PULLDOWN);  // entrada em nível baixo sensor 2
  pinMode(13, OUTPUT);              // saída para abertura do pistão
  pinMode(14, OUTPUT);              // saída para retração do pistão
}
void loop() {
```

```
  int sensorVal1 = digitalRead(2); // sensor da posição estendido é lido
  int sensorVal2 = digitalRead(3); // sensor da posição recolhido é lido
    if (sensorVal1 == LOW) {         // se o sensor 1 foi ativado
      digitalWrite(13, HIGH);        // estende o pistão
    }
    else if (sensorVal2 == LOW)  { // sensor 2 foi ativado
      digitalWrite(14, HIGH);         // retrai o pistão
    }
}
```

Os botões Gravar, Iniciar e Apagar seguem a mesma ideia dos sensores dos atuadores pneumáticos. A suas respectivas portas no Arduino estarão em nível alto, quando um dos botões for pressionado a sua porta passará para nível baixo. Dessa forma acionará o caso verdadeiro, efetuando a operação lógica criada no programa, executando o trabalho desejado.

A gravação dos valores recebidos nas portas do sistema embarcado segue a seguinte rotina:

```
#include <EEPROM.h>     // a biblioteca da memória é carregada
int addr = 0;  // a gravação iniciará na posição 0 da memória
void setup() {
                  // nenhuma variável a ser setada
}
void loop() {
  int val = analogRead(0) / 4;       // é preciso dividir a variável lida por 4
                                     // porque entradas analógicas variam
                                     // de 0 a 1023 bits e cada byte da
                                     // EEPROM só pode realizar de 0 a 255
EEPROM.write(addr, val);             // grava valor lido no endereço atual
addr = addr + 1;                     // vai para o próximo endereço
if(addr == EEPROM.length())          // se atingir o limite da memória
  addr = 0;                          // retorna para o endereço zero
```

```
        delay(100);                          // tempo para realizar processo
        }
```

E então o processo de leitura é realizado da seguinte forma, seguindo a mesma lógica da gravação, exceto pelo comando da leitura:

```
        #include <EEPROM.h>
        int address = 0;
        void setup() {
                        // ...
         }
        void loop() {
          value = EEPROM.read(address);  // realiza leitura do endereço atual
          address = address + 1;
          if(address == EEPROM.length())
            address = 0;
            delay(500);
        }
```

A rotina para apagar a memória é a seguinte:

```
        #include <EEPROM.h>
        void setup()  {
        for ( int i = 0 ; i < EEPROM.length() ; i++ )        // apaga do endereço zero
          EEPROM.write(i, 0);                                // ao endereço máximo
          digitalWrite(13, HIGH);    // um led sinaliza o processo de cada limpeza
        }
        void loop()  {
                        // não há repetição do processo, sendo acionado somente
                        // quando solicitado
        }
```

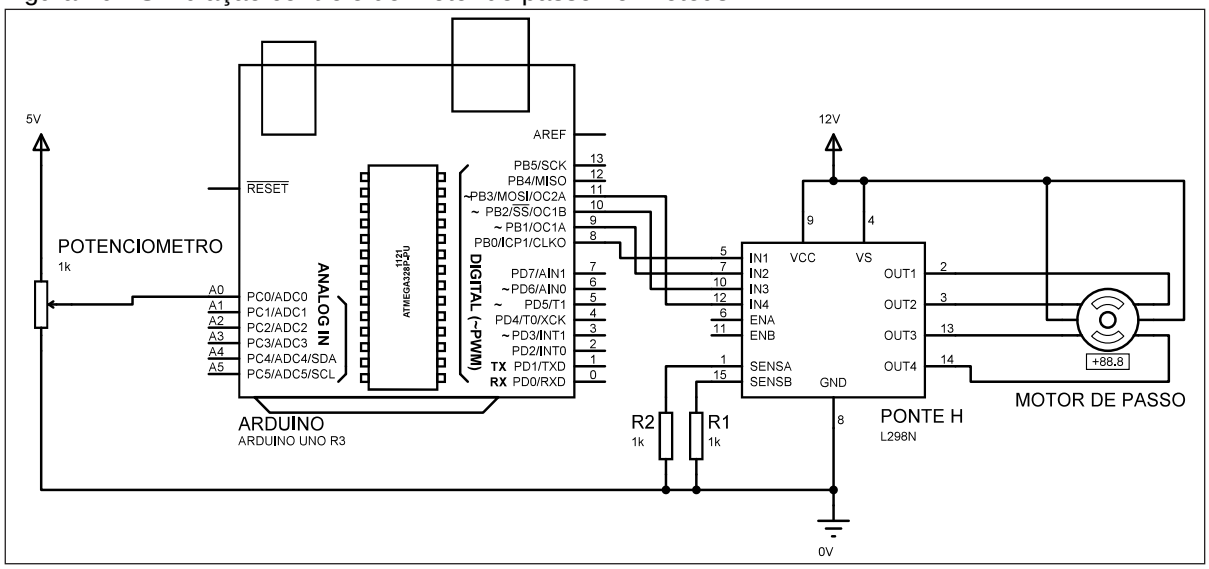
No Apêndice B é apresentado o *firmware* completo utilizado no projeto.

3.3 Etapa 3: Simulação computacional

Com a criação do *firmware* foram iniciadas as simulações computacionais, mediante a utilização do *software* Proteus®. Como na biblioteca desse não há a placa do Arduino MEGA, foi utilizada a placa do UNO, e também devido às limitações dessa placa as conexões das portas do microcontrolador foram alteradas.

Primeiramente foi criado o circuito de controle dos motores de passo, efetuando o teste de controle de um motor de passo por vez, Figura 40.

Figura 40 - Simulação controle do motor de passo no Proteus.

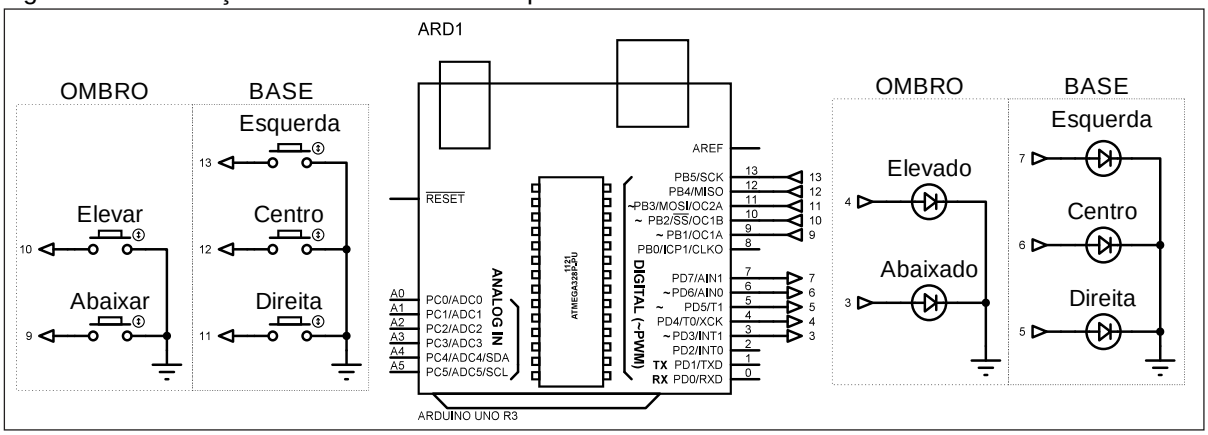


Fonte: Autor.

O motor de passo do Proteus® possui um display abaixo do seu símbolo que mostra o ângulo de rotação do motor. A partir desse é possível controlar e verificar a posição do eixo do motor. Então após o início da simulação foi possível efetuar o controle do motor, sendo satisfatória a resposta do *firmware* para essa etapa.

Após os testes do controle dos motores serem positivos, iniciaram-se as simulações da etapa de controle dos atuadores pneumáticos. Entretanto como não é possível simular os cilindros pneumáticos no Proteus® foram utilizados LEDs nos seus lugares, que sinalizaram o estado atual do cilindro. Na Figura 41 é apresentado o circuito desenhado para efetuar essa etapa de teste.

Figura 41 - Simulação controle dos cilindros pneumáticos no Proteus®.



Fonte: Autor.

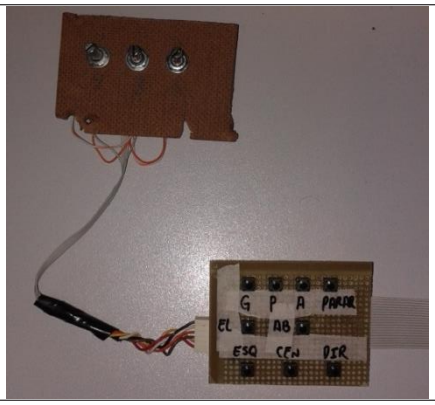
Quando o botão Elevar é pressionado o LED Elevado acende e permanece nesse estado, e quando o botão Abaixar é pressionado o LED Elevado apaga e o Abaixado acende. Essa lógica de operação do ombro serve também para o movimento dos LEDs da base, quando uma posição está acesa as outras permanecem apagadas. Conforme se realizou os testes de cada botão, obteve-se também resposta positiva na simulação do controle dos cilindros pneumáticos.

A partir dos resultados positivos das etapas de teste, montou-se o circuito completo no Proteus®, conforme se verifica no Apêndice C. Entretanto devido a problemas de compatibilidade da placa do Arduino MEGA adicionada à biblioteca do Proteus® não foi possível realizar a simulação do circuito completo. Entretanto como se verificou que tanto o *firmware* como as ligações elétricas estavam corretos, partiu-se diretamente para os testes em bancada, montando o circuito em *protoboard*.

3.4 Etapa 4: Testes em bancada

Com as simulações realizadas e com os resultados parcialmente satisfatórios, e devido aos problemas citados anteriormente, efetuou-se os testes em bancada. Confeccionou-se para o controle do braço uma placa com botões e potenciômetros, conforme se pode ver na Figura 42.

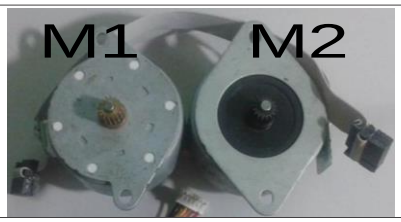
Figura 42 - Placa do controle.



Fonte: Autor.

Onde foram utilizados botões conhecidos como *push button*, e potenciômetros de 5 kΩ e 10 kΩ. Com a placa do controle finalizada se iniciou a procura em sucatas de impressoras por motores de passo que apresentassem bom funcionamento, obtendo-se os modelos PM55L-048-HPG9 (M1) e 55SPM25D7ZA1 (M2), Figura 43.

Figura 43 - Motores de passo.



Fonte: Autor.

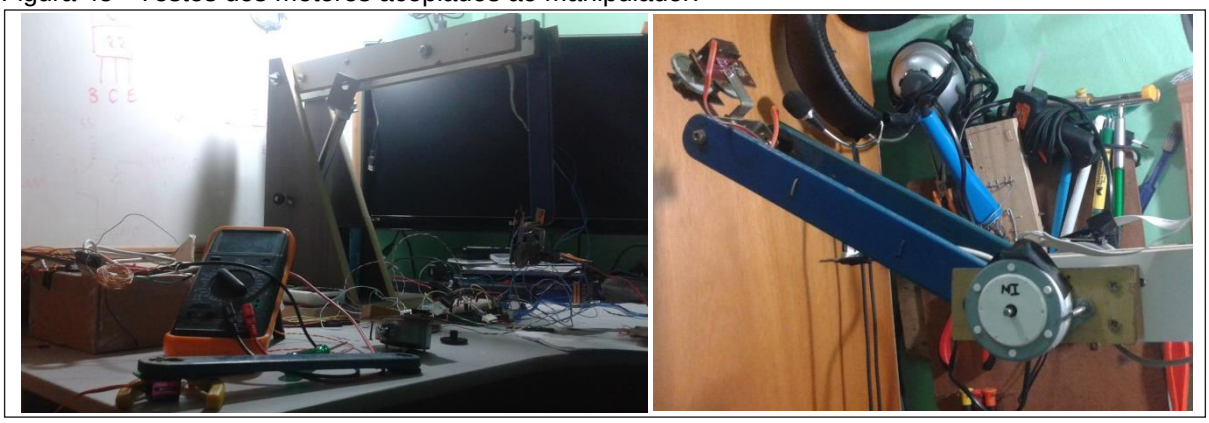
De posse de todos os componentes necessários, módulo controle, motor de passo, driver L298N, Arduino, *protoboard*, fonte 12 V, LEDs e resistores, pode-se então montar o circuito na bancada (Figura 44).

Figura 44 - Circuito completo em bancada, motores a vazio.

Fonte: Autor.

Assim como na simulação, os movimentos dos motores de passo e servos-motores controlados pelos potenciômetros funcionaram de acordo. Entretanto diferentemente da simulação onde se encontram sinais ideias, no teste em bancada ocorre trepidação no eixo dos motores em ±1º. Problema oriundo do atrito entre as partes metálicas internas do resistor variável. A Figura 45 apresenta os testes realizados com os motores montados no braço robótico.

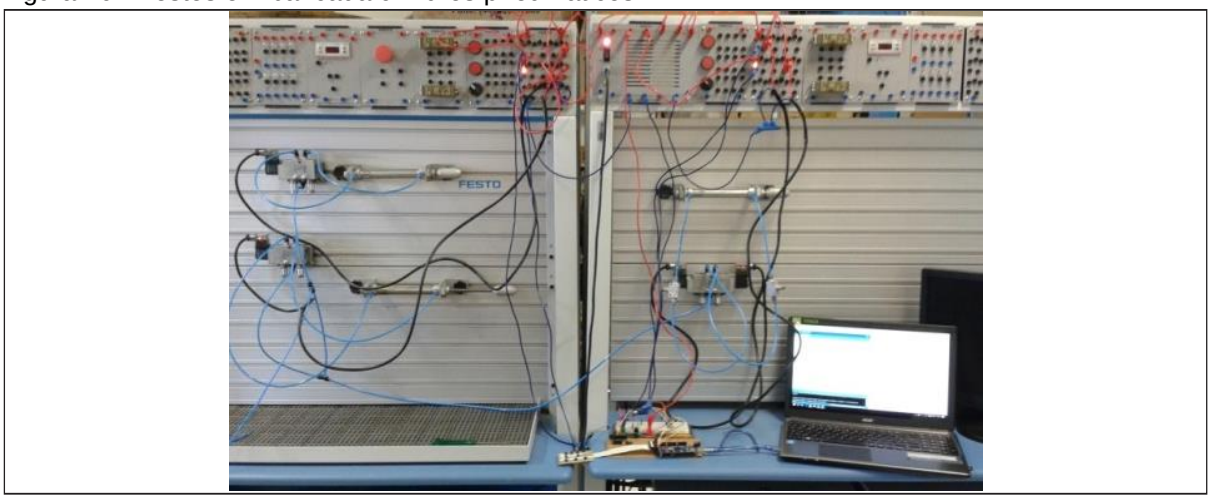
Figura 45 - Testes dos motores acoplados ao manipulador.



Fonte: Autor.

O controle dos cilindros pneumáticos, exemplificados por LEDs, funcionaram perfeitamente, acionando-os de forma correta. E então se efetuou o teste com os cilindros no laboratório de pneumática, apresentando operação de acordo com o esperado (Figura 46).

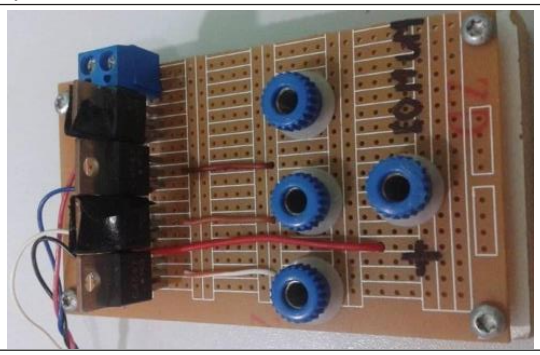
Figura 46 - Testes em bancada cilindros pneumáticos.



Fonte: Autor.

Para o acionamento das válvulas solenoide dos cilindros foi desenvolvida uma placa com transistores (Figura 47) para fazer a comunicação entre o Arduino e as válvulas. Como apresentado na Seção 2.2.1, a corrente máxima nas portas do Arduino é de 40 mA e 5 V, e como a válvula trabalha com 24 V, se conectado diretamente ao sistema embarcado esse queimaria suas portas. Portanto os transistores operam como chaves que possibilitam trabalhar em tensões e correntes maiores que o suportado pelo Arduino.

Figura 47 - *Drive* de corrente para válvulas solenoide.



Fonte: Autor.

E também o teste das opções de gravação e execução dos movimentos, assim como de parar e apagar, funcionaram de forma satisfatória, executando corretamente o *firmware* desenvolvido. Na Tabela 4 são apresentadas as correntes que circulam pelos motores, *drives* e válvulas, possibilitando a análise do consumo do braço robótico, bem como potência do projeto.

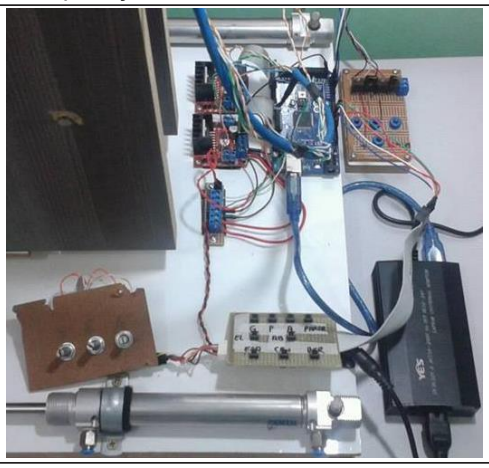
Tabela 4 - Correntes de operação do braço robótico.

Componente	Ibobina	Itotal
Motor de passo 1	300 mA	600 mA
Motor de passo 1	300 mA	600 mA
Servo-motor punho	-	140 mA
Servo-motor garra	-	110 mA
Ponte H 1	-	750 mA
Ponte H 2	-	750 mA
Válvula solenoide	110 mA	450 mA

Fonte: Autor.

Na Figura 48 são mostradas as placas de comando e operação do manipulador robótico.

Figura 48 - Placas de comando e operação.



Fonte: Autor.

E na Figura 49 se pode ver uma sequência de operação do braço, realizada no laboratório de hidráulica e pneumática da UNIPAMPA. O diagrama eletropneumático completo do braço se encontra no APÊNDICE C.

Figura 49 - Sequênca de movimentos manipulador robótico.

Fonte: Autor.

4. ANÁLISE DOS RESULTADOS

Na construção da estrutura da base foi necessário projetar duas chapas metálicas chamadas no projeto de haste direita e haste esquerda (APÊNDICE E). Essas realizam a transferência de movimento dos cilindros da base para o pino de giro da base circular, pois em um primeiro momento colocou-se os cilindros diretamente acoplados para realizarem esse movimento, ocorrendo atrito com a coluna do braço. Com o uso dessas chapas foi possível solucionar esse problema no movimento de giro da base do braço robótico.

Inicialmente a ideia era fixar o antebraço diretamente ao eixo dos motores de passo, entretanto esses não tiveram torque suficiente para manter posição, gerando movimento próprio no motor. Perdendo completamente a posição referência com o seu respectivo potenciômetro, e após a utilização das engrenagens foi incrementado força de torque, propiciando a utilização e controle correto desses motores.

E no segundo, no punho, idealizou-se a utilização de motor de passo para efetuar o giro da garra. Contudo como anteriormente os outros motores de passo já haviam apresentado problemas no torque, trocou-se por servo-motor que possui menor peso e torque suficiente para executar movimentos demonstrativos. Pelo mesmo fator também se substituiu o motor de passo da garra por outro servo-motor.

No controle do braço robótico o problema principal está na leitura dos sinais analógicos dos potenciômetros e comunicação com os motores. Em testes iniciais foram utilizados potenciômetros de 100 kΩ, porém, o motor apresentava grande trepidação oriunda da variação na posição do eixo, mesmo em repouso, com variações de ±4º no eixo. Com a utilização do *monitor serial* da IDE do Arduino se verificou que o sinal enviado pelo potenciômetro não era exato, gerando mudança nos pulsos enviados ao *drive*, movimentando a posição do motor. Essa imprecisão se dá devido aos atritos mecânicos sofridos internamente no potenciômetro, somada a resistência dos cabos de conexão, mas principalmente ao alto valor resistivo do potenciômetro. Então como solução mais simples se efetuou a troca por potenciômetros de valores de 5 kΩ e 10 kΩ, diminuindo consideravelmente a imprecisão, encontrando-se variação de ±1º no eixo dos motores. Essa trepidação além de prejudicar a suavidade dos movimentos do braço, também ocasiona maior aquecimento dos motores, diminuindo sua vida útil. Para contornar esse problema se

pode utilizar resistores de potência e controladores de tensão para limitar a corrente do comum e bobinas dos motores. Porém se optou pela troca do tipo de ligação do motor para bipolar que suporta corrente total de 800 mA em suas bobinas, o que reduziu o aquecimento e melhorou o torque. Entretanto, outra solução muito mais plausível seria o uso de *drives* específicos para motores de passo, como o A4988, que permite o ajuste de corrente máxima fornecida para as bobinas do motor e a configuração do tipo de passo a ser utilizado, passo completo, meio passo ou micro passo. Melhorando consideravelmente a suavidade de controle, movimento e consumo dos motores de passo. Para os servos motores um simples regulador de tensão, como o 7806, é suficiente para controlar sua alimentação e correta operação. Onde esse regulador idealmente forneceria 6 V, que é a tensão de máximo torque dos servos motores utilizados, porém na prática fornece um valor um pouco menor que esse, mantendo o torque nominal e valor de corrente aceitável do servo. No projeto alimentou-se os servos-motores com a fonte de tensão de 5 V existente no próprio módulo da ponte H.

Na programação do *firmware* a grande dificuldade foi à execução da leitura das posições dos atuadores, pois como o uso de sensores de posição foram limitados, surgiram erros de execução na leitura dos dados gravados. A posição a ser executada se perdia da posição atual, confundindo a ordem dos movimentos. A solução mais factível encontrada foi à utilização de um vetor posição fixo, onde cada atuador possui endereço fixo, sendo sempre executado nessa ordem. Ou seja, por exemplo, mesmo que o atuador do cotovelo não precise mudar de posição, o seu endereço é verificado e executado, no caso ainda permanece parado porque não há nova posição, seu estado atual não precisa ser incrementado ou decrementado conforme o estado futuro.

Tabela 5 - Custo total do projeto.

1	Arduino MEGA 2560 R3	R$70,00
2	Ponte H L298N	R$30,00
2	Servo-motor MG90S	R$40,00
2	Potenciômetro 5 kΩ	R$4,00
-	Parafusos, porcas e arruelas	R$10,00
TOTAL		R$154,00

*Todos os valores são aproximados.

5. CONSIDERAÇÕES FINAIS

O projeto atingiu de forma satisfatória os resultados esperados, contendo alguns problemas que podem ser resolvidos em trabalhos futuros, ou melhorados. Na estrutura mecânica a principal dificuldade foi o projeto e execução, tendo em vista que se partiu muito mais para área mecânica que o da elétrica. Porém foi possível solucionar a maioria desses, como a questão dos eixos do cotovelo e punho.

Com esse projeto foi possível trabalhar nas áreas da mecânica, computação e elétrica, que são os pilares da robótica e mecatrônica. Possibilitando grande aprendizado e, principalmente, capacidade de resolução de problemas, aliando os conhecimentos teóricos obtidos durante o curso com a execução prática do projeto. Espera-se que esse projeto auxilie na disseminação da robótica e mecatrônica dentro dessa universidade, e que possibilite estudos futuros com pesquisas nesses ramos. Portanto, foi criado no presente trabalho um braço mecânico robótico utilizando atuadores pneumáticos e elétricos controlados por Arduino, de baixo custo, propiciando replicações e utilização para estudo no ambiente acadêmico da Universidade Federal do Pampa.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] ROSÁRIO, João Maurício. **Princípios de mecatrônica** / João Maurício Rosário. – São Paulo: Prentice Hall, 2005.

[2] CRAIG, John J. **Robótica** / John J. Craig; tradução Heloísa Coimbra de Souza; revisão técnica Reinaldo Augusto da Costa Bianchi. – 3. Ed. – São Paulo: Pearson Education do Brasil, 2012.

[3] CARRARA, Valdemir. **Introdução à Robótica Industrial**. INPE. Disponível em: <http://mtc-m21b.sid.inpe.br/col/sid.inpe.br/mtc-m21b/2015/08.25.14.16/doc/publicacao.pdf>. Acesso em 05/2016.

[4] MCROBERTS, Michael. **Arduino básico** / Michael McRoberts; [tradução Rafael Zanolli]. -- São Paulo: Novatec Editora, 2011.

[5] ____. **Datasheet ATmega 640/V-1280/V-1281/V-2560/V-2561**. Disponível em: <http://www.atmel.com/images/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf>
Acesso em: 05/2016.

[6] STROUSTRUP, Bjarne. **The C++ programming language** / Bjarne Stroustrup. -- Fourth edition.

[7] KERNIGHAN, Brian W. ; RITCHIE, Dennis M. **C: A Linguagem de Programação**. 11ª Edição. Porto Alegre: EDISA, 1998.

[8] DIAGO, Ronaldo. **Eletrônica: eletrônica digital** / Ronaldo Diago, Valder Moreira Amaral (autores); Edson Horta (coautor); Marcos Vagner Zamboni (revisor); Jun Suzuki (coordenador). -- São Paulo: Fundação Padre Anchieta, 2011. (Coleção Técnica Interativa. Série Eletrônica, v. 4).

[9] IDOETA, Ivan V. ; Capuano, Francisco G. **Elementos de eletrônica digital**. 40º ed. São. Paulo: Érica, 2008.

[10] BONACORSO, Nelso Gauze. **Automação Eletropneumática** / Nelso Gauze Bonacoreso, Valdir Noll. - São Paulo: Erica, 1997.

[11] FIALHO, Arivelto Bustamante. **Automação Pneumática: Projetos, Dimensionamento e Análise de Circuitos** / Arivelto Bustamante Fialho. – São Paulo: Érica, 2003.

[12] ____. **Manual de Instrução: Compressores de Pistão**. Disponível em: <http://www.pressure.com.br/download/manual_compressores_de_pistao.pdf>. Acesso em 05/2016.

[13] WILDI, Theodore. **Máquinas eléctricas y sistemas de potencia**. Sexta edición. Pearson Educación, Mexico, 2007.

[14] GRIMBLEBY, J. B. **Stepping Motors**. Disponível em: <http://homepage.divms.uiowa.edu/~jones/step/an907a.pdf>. Acesso em 09/2016.

[15] ACARNLEY, P. P. (Paul P). **Stepping motors**. – 4th ed. – (Control engineering series; no. 63). IEE, 2002.

[16] MANTOVANI, S. C. A; OKI, N. **Aula 3 – Motor de Passo**. Disponível em: <http://www.feis.unesp.br/Home/departamentos/engenhariaeletrica/aula3-motor-de-passo-2013-1-13-03-2013-final.pdf>. Acesso em 09/2016.

[17] ____. **Datasheet AN907 – Stepping Motors Fundamentals**. Disponível em: <http://homepage.divms.uiowa.edu/~jones/step/an907a.pdf>. Acesso em 09/2016.

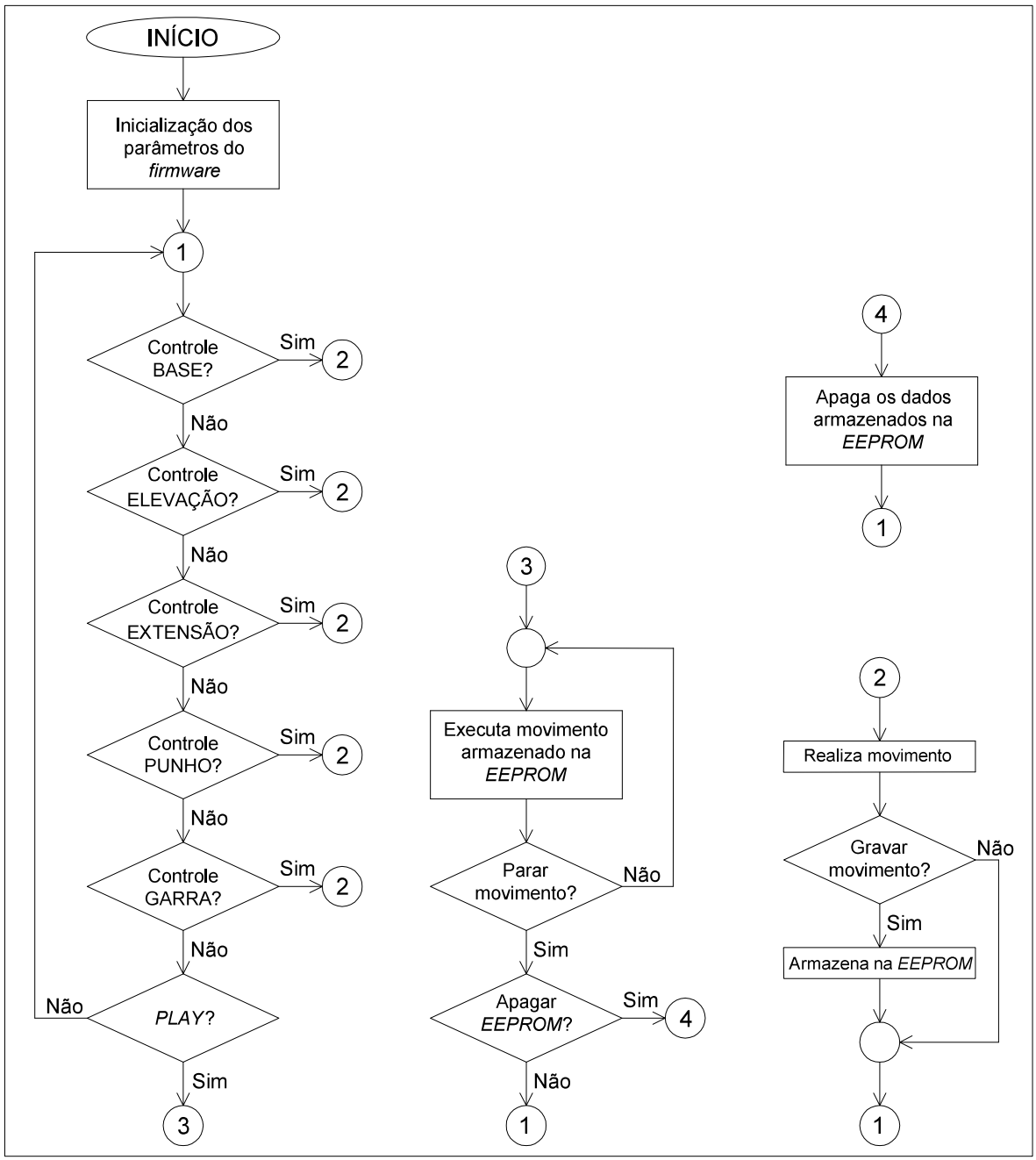
[18] ____. **Datasheet PM55L-048**. Disponível em: <https://www.jameco.com/Jameco/Products/ProdDS/2209254.pdf>. Acesso em 10/2016.

[19] ____. **Datasheet MG90S**. Disponível em: <https://engineering.tamu.edu/media/4247823/ds-servo-mg90s.pdf>. Acesso em 10/2016.

[20] MAIMON, Felipe. **Projeto de um Sistema Eletrônico para o Controle de Motores de Alta Potência por PWM**. PUC Rio. 2004.

[21] ____. **Datasheet L298N – Dual Full – Bridge Driver**. Disponível em: <https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf>. Acesso em: 05/2016.

APÊNDICE A – Fluxograma



APÊNDICE B - Firmware do Braço Robótico

```
/*
Observação:
O valor da variável "n" define faixa de operação do servo.
Para n = 1, uma volta do potenciômetro representa 1 volta do servo.
Para n = 2, uma volta do potenciômetro representa 2 voltas do servo.
Para n = 0.25, uma volta do potenciômetro representa um quarto de volta do servo.
Para uma operação mais precisa, é aconselhável que o produto n*passos_por_rev
resulte num valor inteiro. A utilização dos chamados "micropassos" pode
flexibilizar esse conselho.
*/

//Utilização da bibliotecas
#include <EEPROM.h>
#include <Stepper.h>
#include <Servo.h>

//entradas
#define bGravar     47   // grava posição     - 16
#define bPlay       49   // iniciar gravaões  - 17
#define bApagar     51   // apagar memória    - 18
#define bParar      53   // parar operação    - 19
#define bDireita    37   //gira para direita  - 11
#define bCentro     39   //gira para centro   - 12
#define bEsquerda   41   //gira para esquerda - 13
#define bElevar     43   //eleva o braço      - 14
#define bAbaixar    45   //abaixa o braço     - 15

//saídas
#define Direita     22  //fio preto
#define Esquerda    23  //fio branco
```

```
#define Ombro       24  //fio azul

//variável botão não pressionado
#define bGravar0    80
#define bPlay0      81
#define bParar0     82
#define bApagar0    83
#define bDireita0   84   //gira para direita
#define bCentro0    85   //gira para centro
#define bEsquerda0  86   //gira para esquerda
#define bElevar0    87   //eleva o braço
#define bAbaixar0   88   //abaixa o braço

const int passos_por_rev = 48; //Quantidade de passos por rotação do motor
boolean aGravar, aPlay, aParar, aApagar, aDireita, aCentro, aEsquerda, aElevar, aAbaixar; //Grava o útimo valor lido nos botões

//Pinos utilizados para o controle dos motores
Stepper motor1(passos_por_rev, 13, 12, 11, 10);
Stepper motor2(passos_por_rev, 17, 18, 19, 20);
Servo myservoP; //Punho
Servo myservoG; //Garra

//EEPROM
int k = 0;
int addr = 0;
int address = 0;
byte value;
//MOTOR DE PASSO
int posicao;      //Posição atual motor de passo
float n = 3;      //Faixa de operação do motor, altera quantia de passos
int valC;         //Recebe valor potenciometro do cotovelo
//SERVO-MOTOR
int valP;         //Punho
```

```
int valG;        //Garra
//INTERUPÇÃO
int p = 0;
//CILINDROS
int c;           //centro
int d;           //direita
int e;           //esquerda
int Dir;
int Esq;
int Cen;
int ab;
int el;
int Abai;
int Elev;

void setup() {

  //AQUI VAI O CONTROLADOR PID PARA SINCRONIZAR O MOTOR DE PASSO COM O POTENCIOMETRO DO CONTROLE AO LIGAR
  //posicao = map(analogRead(2), 0, 1023, 0, n * passos_por_rev); //Esse comando sincroniza a posição inicial do motor com a posição inicial do potenciômetro.
  posicao = 117; //valor em graus da posição em repouso ao ligar o braço
  myservoG.attach(7);       //saidas dos servos
  myservoP.attach(5);
  motor1.setSpeed(50);
  motor2.setSpeed(50);

  //define entradas
  pinMode(bGravar,   INPUT);
  pinMode(bPlay,     INPUT);
  pinMode(bParar,    INPUT);
  pinMode(bApagar,   INPUT);
```

```
  pinMode(bDireita,  INPUT);
  pinMode(bCentro,   INPUT);
  pinMode(bEsquerda, INPUT);
  pinMode(bElevar,   INPUT);
  pinMode(bAbaixar,  INPUT);

  //define saídas
  pinMode(Direita,  OUTPUT);
  pinMode(Esquerda, OUTPUT);
  pinMode(Ombro,    OUTPUT);

  //define estado inicial das portas
  digitalWrite(bGravar,   HIGH);
  digitalWrite(bPlay,     HIGH);
  digitalWrite(bParar,    HIGH);
  digitalWrite(bApagar,   HIGH);
  digitalWrite(bDireita,  HIGH);
  digitalWrite(bCentro,   HIGH);
  digitalWrite(bEsquerda, HIGH);
  digitalWrite(bElevar,   HIGH);
  digitalWrite(bAbaixar,  HIGH);
  digitalWrite(Direita,   LOW);
  digitalWrite(Esquerda,  LOW);
  digitalWrite(Ombro,     LOW);

  //interrupção externa
  attachInterrupt(0,Parar,RISING); //Rising - quando nível alto na porta

  Serial.begin(9600);       //para mostrar na tela "monitor serial"
  Serial.println(posicao);  //mostra na tela posição inicial
}

void loop() {
  //Controle PUNHO
```

```
valP = analogRead(0);                //Lê valor do potenciometro
valP = map(valP, 0, 1023, 0, 180);    //Converte para graus de 0 a 180
myservoP.write(valP);                //Executa movimento servo
//CONTROLE GARRA
valG = analogRead(1);
valG = map(valG, 0, 1023, 90, 175);
myservoG.write(valG);
//CONTROLE COTOVELO
valC = analogRead(2);                          //Lê potenciometro
valC = map(valC, 0, 1023, 0, n * passos_por_rev);
//Sequência para movimento do motor de passo
if (valC != posicao) {
  if (valC > posicao) {
    motor1.step(1);
    motor2.step(1);
    posicao ++;
  }

  if (valC < posicao) {
    motor1.step(-1);
    motor2.step(-1);
    posicao --;
  }
}
//mostra valores na tela
Serial.print("Posição: ");
Serial.print(posicao);
Serial.print("\t Cotovelo: ");
Serial.print(valC);
Serial.print("\t Punho: ");
Serial.print(valP);
Serial.print("\t Garra: ");
```

```
  Serial.println(valG);
  delay(15);

  //Verifica botões
  switch (CheckButton()) {
    case bDireita:
     {
       Serial.print("direita pressionado\n");
       d = 1;      //para não abrir válvula quando já estiver na direita
       c = 0;
       if (e == 1) {
         digitalWrite(Esquerda, LOW);
         delay(1000);
         digitalWrite(Direita, HIGH);
         e = 0;
         delay(100);
       }
       else {
         digitalWrite(Esquerda, LOW);
         digitalWrite(Direita, HIGH);
       }
       break;
     }

    case bCentro:
     {
       Serial.print("centro pressionado\n");
       c = 1;
       digitalWrite(Direita, LOW);
       digitalWrite(Esquerda, LOW);
       delay(100);
       break;
     }
```

```
case bEsquerda:
  {
    Serial.print("esquerda pressionado\n");
    e = 1;
    c = 0;
    if (d == 1) {
      digitalWrite(Direita, LOW);
      delay(1000);
      digitalWrite(Esquerda, HIGH);
      d = 0;
      delay(100);
    }
    else {
      digitalWrite(Direita, LOW);
      digitalWrite(Esquerda, HIGH);
    }
    break;
  }

case bElevar:
  {
    el = 1;
    ab = 0;
    Serial.print("elevar pressionado\n");
    digitalWrite(Ombro, HIGH);
    delay(100);
    break;
  }

case bAbaixar:
  {
    el = 0;
```

```
      ab = 1;
      Serial.print("abaixar pressionado\n");
      digitalWrite(Ombro, LOW);
      delay(100);
      break;
    }

  case bParar:
    {
      delay(100);
      Serial.print("parar pressionado\n");
      Parar();
      break;
    }

  case bGravar:
    {
      delay(100);
      Serial.print("gravar pressionado\n");
      Gravar();
      break;
    }

  case bPlay:
    {
      delay(100);
      Serial.print("play  pressionado\n");
      Play();
      break;
    }

  case bApagar:
    {
      delay(100);
```

```
        Serial.print("apagar  pressionado\n");
        Apagar();
        break;
      }
    default: ;
  }
}
//Verifica botões
char CheckButton() {                        // funÃ§Ã£o que verifica estado dos botÃµes / sensores, eliminando necessidade de tratamento por "debounce"
  if (aGravar != digitalRead(bGravar)) {  // verifica se houve alteraÃ§Ã£o no estado do bMais
    aGravar = !aGravar;                 // se houve, aMais diferente de aMais anterior
    if (aGravar)                      // se verdadeiro
      return bGravar0;                  // botÃ£o solto
    else                              // senÃ£o
      return bGravar;                   // botÃ£o pressionado
  }
  else if (aPlay != digitalRead(bPlay)) {
    aPlay = !aPlay;
    if (aPlay)
      return bPlay0;
    else
      return bPlay;
  }
  else if (aParar != digitalRead(bParar)) {
    aParar = !aParar;
    if (aParar)
      return bParar0;
    else
      return bParar;
  }
```

```
  else if (aApagar != digitalRead(bApagar)) {
    aApagar = !aApagar;
    if (aApagar)
      return bApagar0;
    else
      return bApagar;
  }
  else if (aDireita != digitalRead(bDireita)) {
    aDireita = !aDireita;
    if (aDireita)
      return bDireita0;
    else
      return bDireita;
  }
  else if (aCentro != digitalRead(bCentro)) {
    aCentro = !aCentro;
    if (aCentro)
      return bCentro0;
    else
      return bCentro;
  }
  else if (aEsquerda != digitalRead(bEsquerda)) {
    aEsquerda = !aEsquerda;
    if (aEsquerda)
      return bEsquerda0;
    else
      return bEsquerda;
  }
  else if (aElevar != digitalRead(bElevar)) {
    aElevar = !aElevar;
    if (aElevar)
      return bElevar0;
    else
      return bElevar;
```

```
  }
  else if (aAbaixar != digitalRead(bAbaixar)) {
    aAbaixar = !aAbaixar;
    if (aAbaixar)
      return bAbaixar0;
    else
      return bAbaixar;
  }
  else
    return 0;
}

void Gravar() {
  valC = analogRead(2);
  valC = map(valC, 0, 1023, 0, n * passos_por_rev);
  EEPROM.write(addr, valC);      //realiza escrita do valor em val
  addr = addr + 1;
  delay(100);

  valP = analogRead(0);
  valP = map(valP, 0, 1023, 0, 180);
  EEPROM.write(addr, valP);      //realiza escrita do valor em val
  addr = addr + 1;
  delay(100);

  valG = analogRead(1);
  valG = map(valG, 0, 1023, 0, 180);
  EEPROM.write(addr, valG);      //realiza escrita do valor em val
  addr = addr + 1;
  delay(100);

  Esq = e;
```

```
  EEPROM.write(addr, Esq);
  addr = addr + 1;
  delay(100);

  Dir = d;
  EEPROM.write(addr, Dir);
  addr = addr + 1;
  delay(100);

  Cen = c;
  EEPROM.write(addr, Cen);
  addr = addr + 1;
  delay(100);

  Elev = el;
  EEPROM.write(addr, Elev);
  addr = addr + 1;
  delay(100);

  Abai = ab;
  EEPROM.write(addr, Abai);
  addr = addr + 1;
  delay(100);

  if (addr == EEPROM.length())  //se atingir o limite de gravações retorna posição 0
    addr = 0;
  delay(100);
}

void Play() {
  address = 0;
  addr = addr - 1;
  for (int i = 0; i < addr; i += 3) {
    if (p == 1) {
```

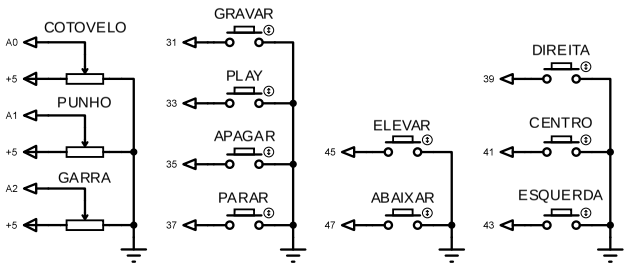
```
  break;
}
else {
  value = EEPROM.read(address);
  while (value != posicao) {
    if (value > posicao) {
      motor1.step(1);
      motor2.step(1);
      posicao ++;
    }

    if (value < posicao) {
      motor1.step(-1);
      motor2.step(-1);
      posicao --;
    }
  }
  posicao = value;
  address = address + 1;
  delay(100);

  value = EEPROM.read(address); // Leitura da EEPROM
  movServo(myservoP, value, 15);
  delay(100);
  address = address + 1;
  delay(100);

  value = EEPROM.read(address); // Leitura da EEPROM
  movServo(myservoG, value, 15);
  delay(100);
  address = address + 1;
  delay(100);
```

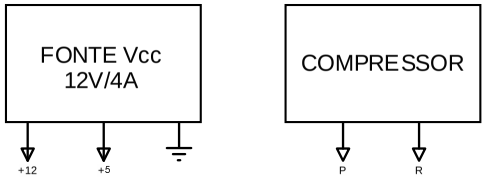
```
value = EEPROM.read(address); //Esquerda
e = value;
if ((e == 1) && (d == 1)) {
  digitalWrite(Direita, LOW);
  delay(3000);
  digitalWrite(Esquerda, HIGH);
  d = 0;
  delay(2000);
}
else if (e == 1) {
  digitalWrite(Direita, LOW);
  digitalWrite(Esquerda, HIGH);
  delay(2000);
}
address = address + 1;
delay(100);

value = EEPROM.read(address); //Direita
d = value;
if ((d == 1) && (e == 1)) {
  digitalWrite(Esquerda, LOW);
  delay(3000);
  digitalWrite(Direita, HIGH);
  e = 0;
  delay(2000);
}
else if (d == 1) {
  digitalWrite(Esquerda, LOW);
  digitalWrite(Direita, HIGH);
  delay(2000);
}
address = address + 1;
delay(100);
```

```
    value = EEPROM.read(address); //Centro
    c = value;
    if (c == 1) {
      digitalWrite(Direita, LOW);
      digitalWrite(Esquerda, LOW);
      delay(2000);
    }
    address = address + 1;
    delay(100);

    value = EEPROM.read(address); //Elevar
    el = value;
    if (el == 1) {
      digitalWrite(Ombro, HIGH);
      delay(3000);
    }
    address = address + 1;
    delay(100);  //para dar tempo do cilindro se manter ou mover

    value = EEPROM.read(address); //Abaixar
    ab = value;
    if (ab == 1) {
      digitalWrite(Ombro, LOW);
      delay(10000);
    }
    address = address + 1;
    delay(100);

    if (address == EEPROM.length()) { //se atingir o limite de gravaÃ§Ãµes retorna posiÃ§Ã£o 0
      address = 0;
```

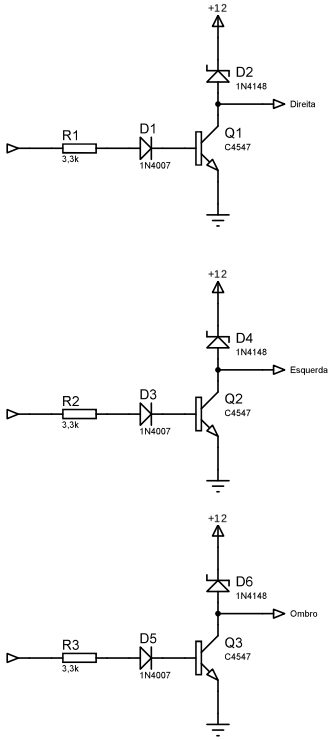
```
        delay(500);
      }
    }
  }
}

void Apagar() {
  for ( int i = 0 ; i < EEPROM.length() ; i++ ) {
    EEPROM.write(i, 0);
  }
}

void movServo(Servo &s, int ANGfinal, int veloc) {
  int ANGinicial = s.read();                    // lê a posição atual do servo e o coloca
como o angulo inicial de movimento
  if (ANGinicial > ANGfinal) {
    for (int i = ANGinicial; i > ANGfinal; i--) { //decremanta 1º grau
      s.write(i);                    // 'escreve' o valor 'i' no servo selecionado
      delay(veloc);                     // tempo de espera.
    }
  }
  else {
    for (int i = ANGinicial; i <= ANGfinal; i++) { //incremanta 1º grau
      s.write(i);                    // 'escreve' o valor 'i' no servo selecionado
      delay(veloc);                     // tempo de espera.
    }
  }
}

void Parar(){
  delay(10000);
}
```

MÓDULO CONTROLE

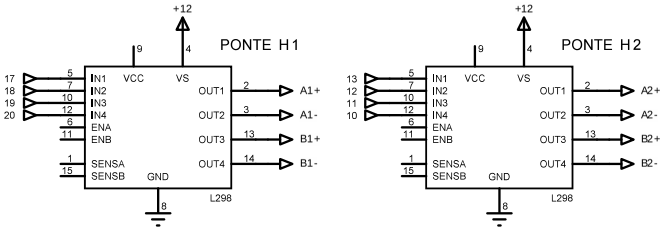


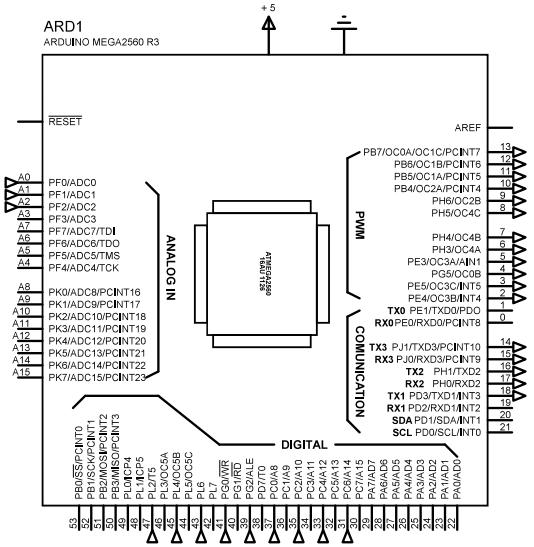


DRIVE VÁLVULAS SOLENOIDE

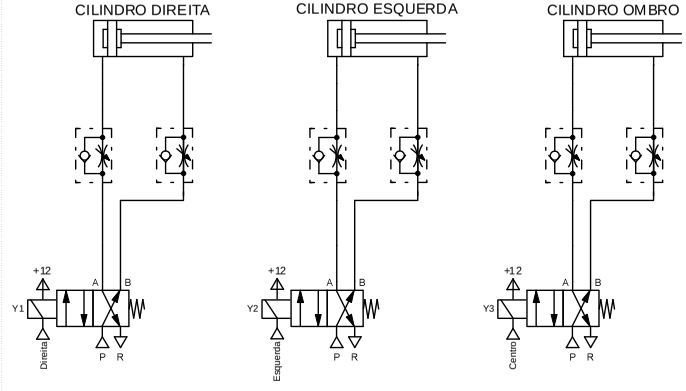


DRIVE MOTORES

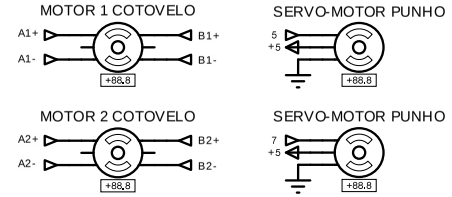




VÁLVULAS SOLENOIDE

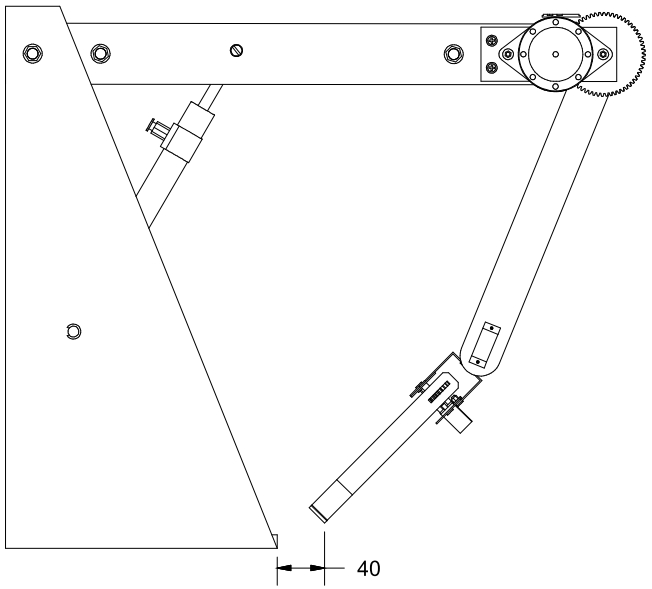


MOTORES

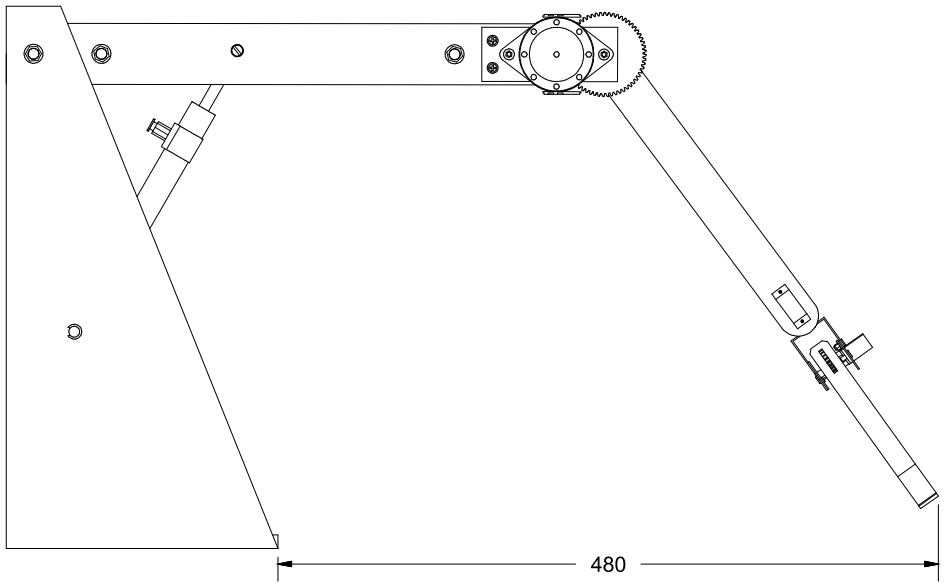


DRAWN Paulo Ricardo	08/11/2016	TITLE	APÊNDICE C - Esquema Eletropneumático	103
		SIZE A3	DWG NO	REV
		SCALE 1:1		SHEET 1 OF 1

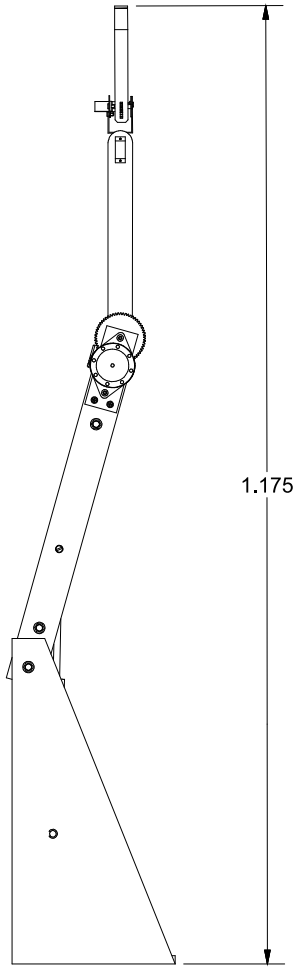
ALCANCE MÍNIMO
EM RELAÇÃO A BASE



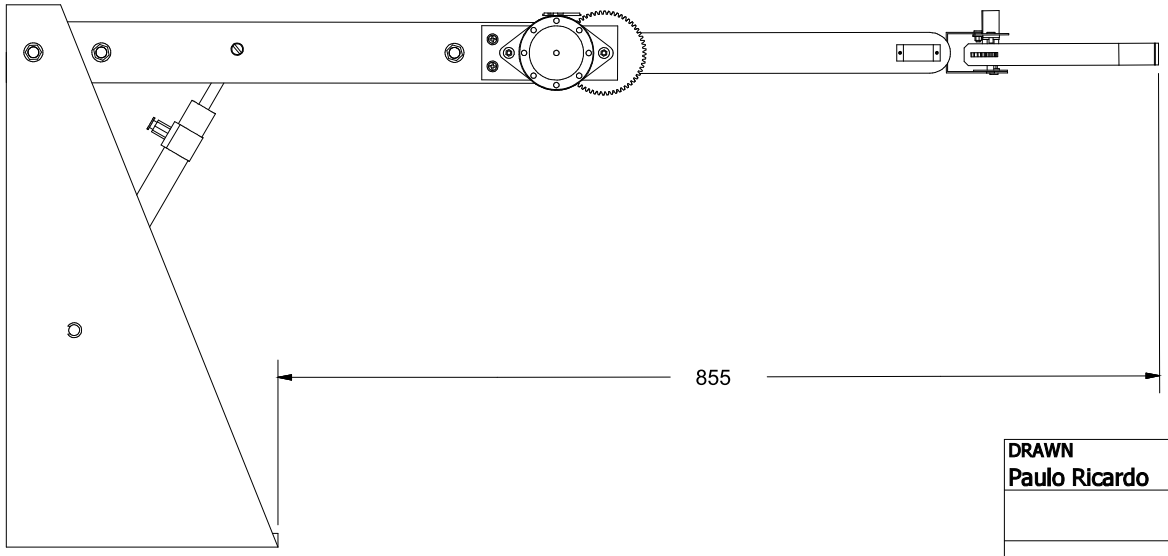
ALCANCE MÁXIMO
EM RELAÇÃO A BASE



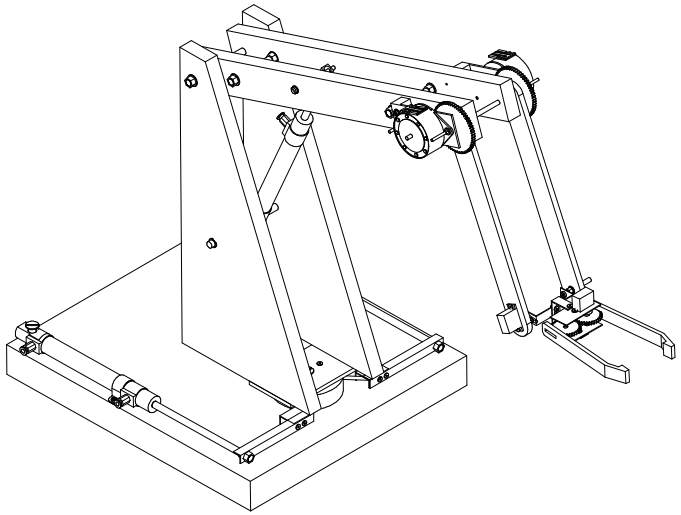
ALCANCE MÁXIMO
VERTICAL



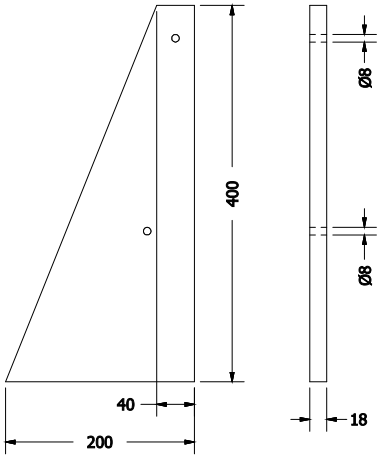
ALCANCE MÁXIMO
HORIZONTAL



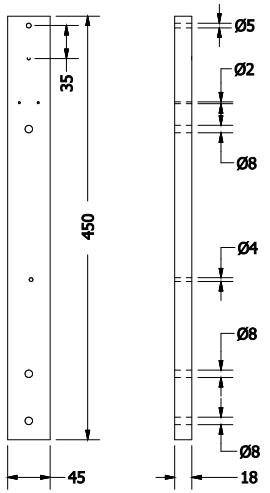
DRAWN Paulo Ricardo	08/11/2016	TITLE APÊNDICE D - Área de Alcance		105
		SIZE A3	DWG NO	REV
		SCALE 1:4		SHEET 1 OF 3



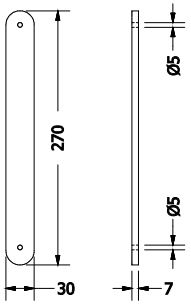
COLUNA



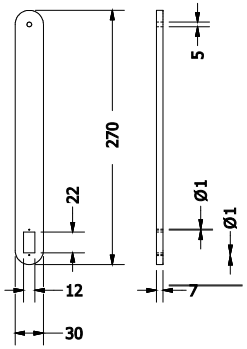
BRAÇO



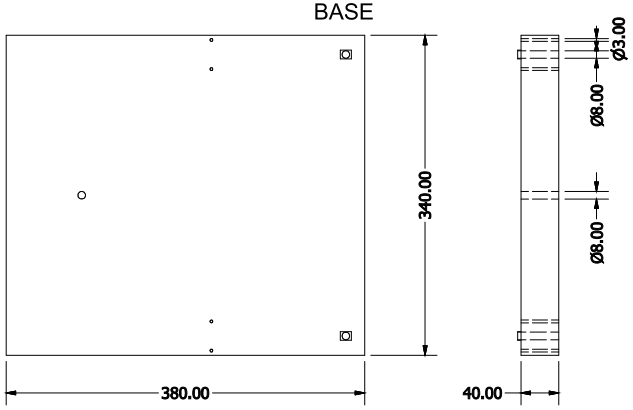
ANTEBRAÇO 1



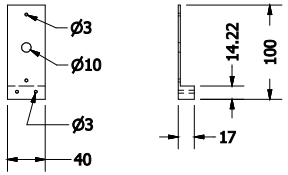
ANTEBRAÇO 2



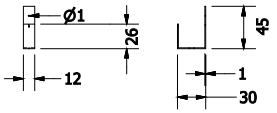
BASE



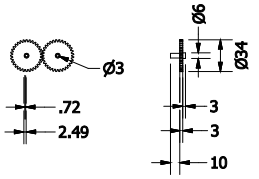
SUPORTE MOTOR DE PASSO



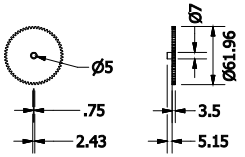
CHAPA ENTRE GARRA E PUNHO



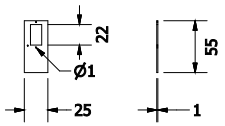
ENGRENAGEM GARRA



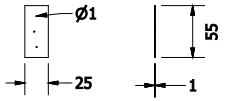
ENGRENAGEM COTOVELO



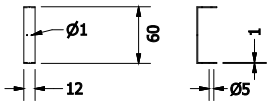
PLACA SUPERIOR GARRA



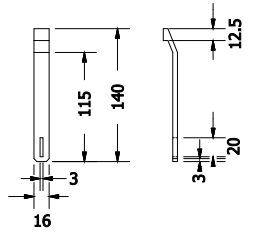
PLACA INFERIOR GARRA



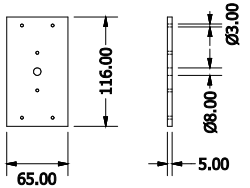
PUNHO



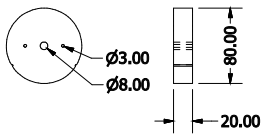
DEDO GARRA



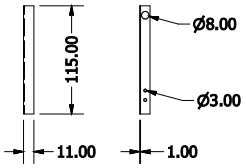
ADAPTADOR BASE CIRCULAR



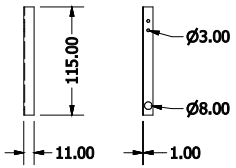
BASE CIRCULAR



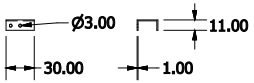
HASTE DIREITA



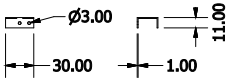
HASTE ESQUERDA



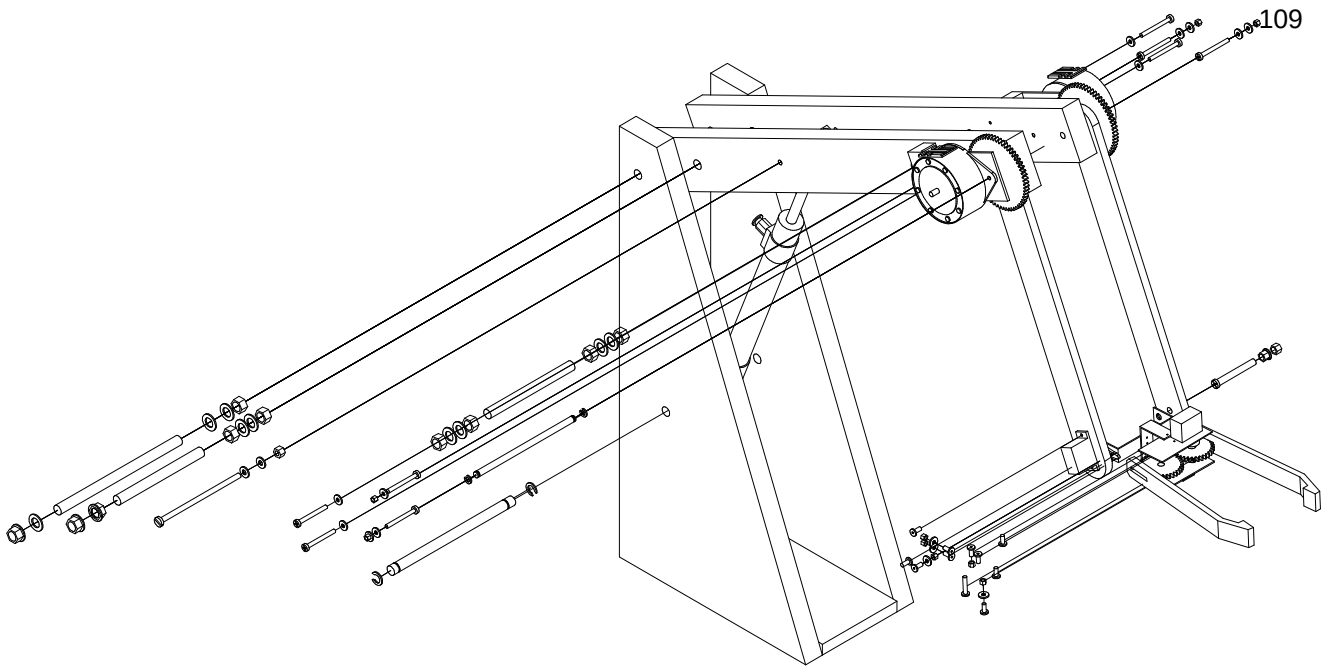
CAIXA HASTE DIREITA



CAIXA HASTE ESQUERDA



DRAWN Paulo Ricardo	08/11/2016	TITLE APÊNDICE E - Vistas das Peças da Estrutura	107
		SIZE A3 / DWG NO	REV
		SCALE 1:4	SHEET 2 OF 3



COLUNA
1 - PINO Ø8 RETIRADO SUCATA IMPRESSORA
2 - ARRUELA DE PRESSÃO Ø7
1 - CILINDRO DA FESTO
1 - ADAPTADOR PARA CONECTAR O CILINDRO AO BRAÇO

BRAÇO
3 - PARAFUSO ROSCA SEM FIM Ø7
10 - PORCA Ø 7
12 - ARRUELA Ø7
1 - PARAFUSO Ø3
1 - PORCA Ø3
9 - ARRUELA Ø3
4 - PARAFUSO PARA MADEIRA Ø2
4 - PARAFUSO Ø2
1 - PINO Ø5 RETIRADO SUCATA IMPRESSORA
2 - ARRUELA DE PRESSÃO Ø5

ANTEBRAÇO
4 - PARAFUSO Ø2
1 - ARRUELA Ø2
1 - PORCA Ø2
1 - PARAFUSO Ø3
1 - PORCA Ø3
1 - PORCA DE PRESSÃO Ø3

GARRA
8 - PARAFUSO Ø2
4 - ARRUELA Ø2
4 - PORCA Ø2

DRAWN Paulo Ricardo	08/11/2016	TITLE	APÊNDICE F - LISTA PEÇAS		
		SIZE A3		DWG NO	REV
		SCALE 1:4			SHEET 3 OF 3