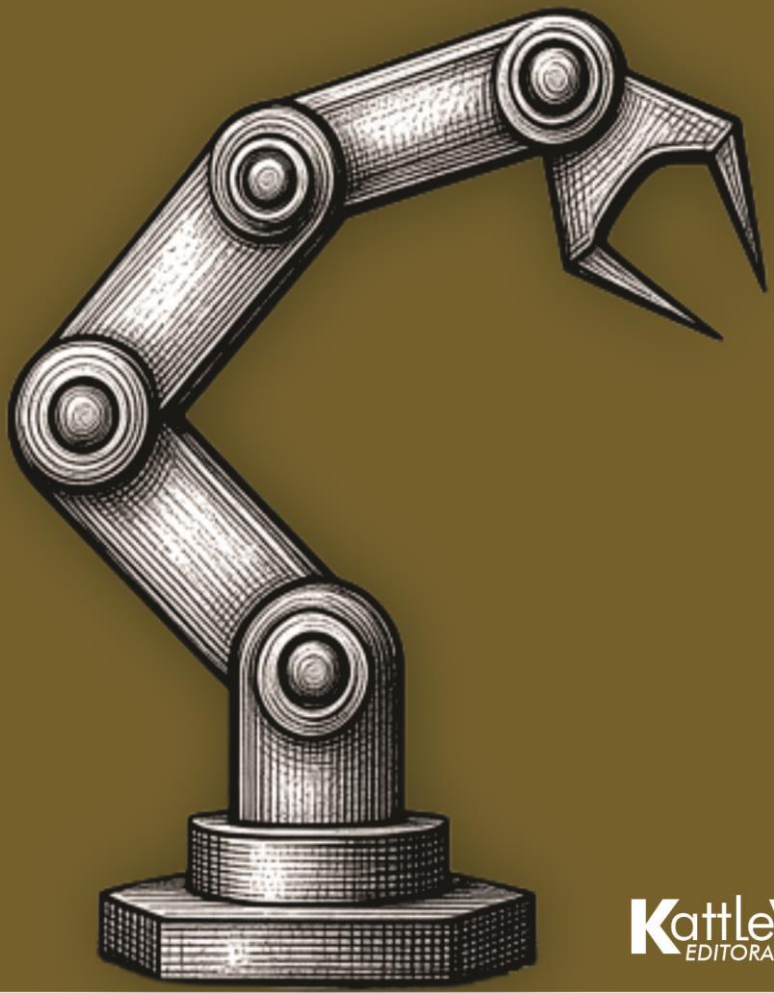
João Antonio da Silva Neto
Elves Sousa e Silva

CONTROLE DE MANIPULADOR ROBÓTICO COM FUNÇÃO DE APRENDIZAGEM USANDO HARDWARE E SOFTWARE DE CÓDIGO ABERTO



KattleYa EDITORA

CONTROLE DE MANIPULADOR ROBÓTICO COM FUNÇÃO DE APRENDIZAGEM USANDO HARDWARE E SOFTWARE DE CÓDIGO ABERTO

DIREÇÃO EDITORIAL: Luciele Vieira da Silva
DIAGRAMAÇÃO: Bruna Natalia de Freitas
REVISÃO ORTOGRÁFICA: Autores
DESIGNER DE CAPA: Editora Kattleya

O conteúdo da obra é de inteira e exclusiva responsabilidade dos autores, incluindo o padrão textual, o sistema de citação e referências bibliográficas.

2026 Editora Kattleya
Aldebaran | Tv. José Alfredo Marques, Loja 05
Antares, Maceió - AL, 57048-230
www.editorakattleya.com
editorakattleya@gmail.com

Catalogação na publicação
Elaborada por Bibliotecária Janaina Ramos – CRB-8/9166

S586c

Silva Neto, João Antonio da

Controle de manipulador robótico com função de aprendizagem usando Hardware e Software de código aberto / João Antonio da Silva Neto, Elves Sousa e Silva. – Maceió-AL: Kattleya, 2026.

Livro em PDF

ISBN 978-65-83366-27-6

1. Robótica. 2. Automação. 3. Engenharia de controle. I. Silva Neto, João Antonio da. II. Silva, Elves Sousa e. III. Título.

CDD 629.892

Índice para catálogo sistemático

I. Robótica

João Antonio da Silva Neto
Elves Sousa e Silva

CONTROLE DE MANIPULADOR ROBÓTICO COM FUNÇÃO DE APRENDIZAGEM USANDO HARDWARE E SOFTWARE DE CÓDIGO ABERTO

Maceió-AL
2026
KattleYa EDITORA

AGRADECIMENTOS

À comunidade de desenvolvedores engajados em contribuir para um conhecimento livre.

Aos professores que colaboraram para minha formação, em especial aos professores Alberdan e Marcelo pelo apoio durante o curso e em sua conclusão.

Aos servidores, amigos e colegas do curso de automação industrial do IFPB.

Aos familiares, amigos e colegas que participaram direta ou indiretamente deste trabalho.

(y)

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIAÇÕES E SIGLAS

AD Analógico para Digital
ARM Advanced RISC Machine
CA Código Aberto
CAD Computer Aided Design (Projeto Auxiliado por Computador)
CAM Computer Aided Manufacturing (Manufatura Auxiliada por Computador)
CAN Controller Area Network (Rede de área do controlador)
CC Corrente Contínua
CLP Controlador Lógico Programável
CNC Controle Numérico Computadorizado
CPU Central Process Unity (Unidade de Processamento Central)
DA Digital para Analógico
FMS Flexible Manufacturing System (Sistema Flexível de Manufatura)
FOSS Free Open Source Software (Software Livre e de Código Aberto)
FSF Free Software Foundation
GdL Grau de Liberdade
GUI Graphical User Interface (Interface Gráfica de Usuário)
HL Hardware Livre
HMI Humam Machine Interface
I^2C Inter-Integrated Circuit (Circuito Inter-integrado)

IDE Integrated Development Environment (Ambiente de Desenvolvimento Integrado)
IHM Interface Homem Máquina
IoT Internet of Things (Internet das Coisas)
ISO International Organization for Standardization (Organização Internacional para Padronização)
MCT Ministério da Ciência e Tecnologia
NC Controle Numérico
OPC OLE for Process Control
OSADL Open Source Automation Development Lab
OSI Open Source Initiative
PC Personal Computer (Computador Pessoal)
PCB Printed Circuit Board (Placa de Circuito Impresso)
PTP Point to Point (Ponto a ponto)
PWM Pulse Width modulation (Modulação por Largura de Pulso)
RAM Random Acess Memory (Memória de Acesso Ramdômico)
ROM Read Only Memory (Memória Apenas de Leitura)
ROS Robot Operating System (Sistema Operacional de Robôs)
RPi Raspberry Pi
RTAI Real Time Application Interface
SBC Single Board Computer (Computador em Placa Única)
SCADA Supervisory Controle And Data Acquisition (Controle Supervisório e Aquisição de Dados)

SL Software Livre
SO Sistema Operacional
SoC System on Chip
SPI Serial Peripheral Interface (Interface Periférica Serial)
TIC Tecnoclogias de Informação e Comunicação
TIC Tecnologias de Informação e Comunicação
UART Universal Asynchronous Receiver/Transmitter (Transmissor/Receptor Universal Assíncrono)
USART Universal Synchronous/Asynchronous Receiver/Transmitter (Transmissor/Receptor Universal Síncrono e Assíncrono)
USB Universal Serial Bus
USRT Universal Synchronous Receiver/Transmitter (Transmissor/Receptor Universal Síncrono)

SUMÁRIO

CAPÍTULO 1
CONTEXTUALIZAÇÃO E PROBLEMATIZAÇÃO

1.1.INTRODUÇÃO

Com a nova economia globalizada há a necessidade das plantas fabris de adaptarem, aprimorarem ou renovarem seus processos, pois para se manter no mercado é necessário ser produtivo. Para alcançar essa meta as empresas vêm investindo massivamente em novas tecnologias e se modernizando. Essas mudanças englobam a organização de novos *layouts* dos processos, modernização dos equipamentos e outros recursos da automação industrial. Para se manter ativo no mercado internacional é necessário ser competitivo e a inovação tecnológica faz parte dessa estratégia.

Em países como o Brasil, a modernização dos processos tornou-se um fator competitivo essencial devido a questões de custo, demanda e qualidade no produto. A necessidade de modernizar os processos produtivos, além de aumentar o volume de produção, tem de atender a requisitos fundamentais como: uso racional de matéria-prima e fontes energéticas, a redução de custos e fabricação de produtos de qualidade. A qualidade tornou-se um importante fator competitivo.

Ser sustentável também é um fator de produtividade. A indústria, como responsável por atividades de alto

impacto ambiental, deve seguir um modelo sustentável de produção. A geração de resíduos industriais é uma forma de desperdício e indício da ineficiência dos processos produtivos utilizados. Isso representa a perda de insumos do processo. Por esse motivo a indústria deve ter tecnologia de ponta (VALLE, 2004).

A automação possibilita a simplificação dos sistemas mecânicos, redução de custos e do tempo de desenvolvimento e a obtenção de produtos com elevado grau de flexibilidade e a capacidade de adaptação a diferentes condições de operação (ROSÁRIO, 2005).

A necessidade de se realizar tarefas com eficiência e precisão levou a criação de robôs. Eles foram responsáveis pela segunda revolução industrial, revolucionando a forma de produzir em série (ROSÁRIO, 2010). Geralmente os robôs atuam em áreas onde humanos não podem atuar. Nesses ambientes insalubres são realizadas tarefas desagradáveis ou de risco a vida como a presença de fontes de alto calor, ruído e gases tóxicos e/ou esforço físico extremo.

Os manipuladores robóticos são os mais utilizados na indústria, pois são bastante versáteis. Desenvolvidos desde o fim da segunda Guerra Mundial, criados para imitar os movimentos do braço e mão humana, eram usados para manipular tubos que continham substâncias radioativas. Depois disso eles foram aplicados na manufatura de peças de alta precisão na indústria aeronáutica (ANGELES, 2007). Isso levou às primeiras máquinas ferramentas numericamente controladas (NC). A diferença entre um manipulador dessa época e um atual, basicamente, é o

controle por computador. Enquanto antes era necessário um operador para dar os comandos, agora basta inserir a programação no controlador do braço robótico que ela será executada quantas vezes for ordenada.

Para programar um robô é necessário inserir as instruções em seu sistema de controle, que as transmitirá para os atuadores, que realizarão os movimentos do manipulador. O tempo de programação da máquina, independentemente do tipo, faz parte do tempo de *set up*. Esse é o período de interrupção da produção para ajuste das máquinas e afeta diretamente a produtividade. Mesmo que o robô venha a produzir artigos de qualidade, com redução de custos e insumos, se a sua produtividade não for aceitável ele pode ser substituído por outra máquina com características compatíveis com os padrões da empresa.

Outro fator fundamental na utilização de braços robóticos é o seu custo/benefício. Segundo Rosário (2010 apud TREMONTI, 2003) "a amortização do robô é fator decisório na sua aquisição. No mercado brasileiro estima-se que o '*pay back*' esteja em torno de 36 meses, em contraposto ao mercado norte americano, que está entre 24 e 12 meses com uma forte tendência de queda". Estima-se que uma mudança no mercado nacional venha ocorrer com o aumento do número de robôs instalados e sua consequente queda de preço. Mesmo sabendo-se que a taxa de amortização de um braço robótico pode variar de acordo com a empresa e a sua aplicação, esse ainda é um fator preponderante.

Por último, outro fator inconveniente com relação aos manipuladores robóticos é a sua integração ao sistema

corrente e a enorme dependência do fabricante para adaptações do equipamento. Apenas o fabricante, geralmente, tem total liberdade sobre o sistema de controle do robô e o seu *hardware*. Isso pode vir a dificultar a sua integração à algum periférico. A adoção de uma arquitetura facilmente encontrada no mercado e até aqueles de código aberto (*open source*), assim como seus protocolos de comunicação, propiciam maior flexibilidade e integração ao sistema. Isso facilita o interfaceamento com sistemas abertos como os sistemas baseado em Linux, por exemplo.

Nos últimos anos uma alternativa que vem crescendo na indústria brasileira, como parte de sua estratégia de inovação, é a adoção de software livre (SL) e hardware livre (HL), tecnologias de código aberto (CA). A adoção de software livre como paradigma do desenvolvimento pode ser resumido em cinco argumentos, segundo Silveira (2004, p. 34):

1. Argumento macroeconômico (custos);
2. Argumento de segurança;
3. Argumento da autonomia tecnológica;
4. Argumento da independência de fornecedores;
5. Argumento democrático.

Para pequena e microempresas que estão iniciando, o SL apresenta grandes vantagens como: custo social baixo, independência de um único fornecedor, desembolso inicial próximo de zero, suporte abundante e gratuito e sistemas e aplicativos muito configuráveis (SILVEIRA, 2004, p. 73). Com exceção do custo inicial o hardware possui as mesmas

características do software. Isso acarreta diretamente na criação de serviços e produtos com custo, qualidade e características atrativas ao consumidor.

Em uma pesquisa realizada pelo Instituto Sem Fronteiras verificou-se que 73% das grandes empresas brasileiras usam software livre. Já entre as menores empresas apenas 31% dão a preferência, segundo Sá et al (2011). Os motivos para isso são, basicamente, os já citados anteriormente. No início o custo foi o fator decisivo para a escolha do SL. Depois a qualidade passou a ser o destaque.

1.2. OBJETIVOS

1.2.1. Objetivo geral

Controlar um manipulador robótico de 3 graus de liberdade, implementar função de aprendizagem e criar uma interface de controle utilizando apenas software e hardware de código aberto.

1.2.2. Objetivos específicos

- Usar apenas software e hardware de código aberto em todas as etapas de desenvolvimento deste trabalho;
- Controlar trajetória de robô usando método de programação mestre/escravo e PTP;
- Desenvolver algoritmos de controle e armazenamento da trajetória do robô;
- Controlar ações do robô de acordo com a exigência da IHM;

- Embarcar o sistema de controle em hardware de código aberto e de baixo custo.

1.3. JUSTIFICATIVA

O presente trabalho se justifica pela necessidade de haver uma ferramenta alternativa para controlar braços robóticos, desenvolver sistemas de controle sem depender de uma solução proprietária e portá-los em dispositivos de arquitetura aberta baseados em Linux.

Este trabalho é importante para impulsionar esse segmento pouco explorado pela indústria nacional, visto que ainda está aquém do nível de outros países. Alguns segmentos do mercado, como o setor de Tecnologia da Informação (TI), já encaram as soluções de código aberto como importantes estratégias competitivas.

Apesar de reconhecida as suas vantagens, as soluções de código aberto são pouco exploradas devido, por exemplo, a falta de suporte, falta de mão de obra especializada, falta de conhecimento e os custos de migração do sistema vigente. Assim, o desenvolvimento de soluções nessa área é essencial, pois impulsiona a criação de projetos desse tipo e contribui no âmbito acadêmico, social e econômico.

CAPÍTULO 2
FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão abordados os conceitos fundamentais para desenvolvimento do projeto proposto.

2.1. SISTEMAS DE PRODUÇÃO

Segundo Groover (2011, p.3), "um sistema de produção é um conjunto de pessoas, equipamentos e procedimentos organizados pare realizar as operações de produção". Para fabricação de qualquer produto será necessário a presença desses elementos. São três os sistemas de produção: manual, mecanizado e automatizado.

2.1.1. Trabalho manual

No sistema de produção onde o trabalho é manual, a fabricação do produto ou realização do serviço é feita por operários que usam suas habilidades e ferramentas manuais. A ferramenta usada nesse sistema de produção deve requerer força e destreza do operador que a manuseia, isto é, ela não pode ser mecanizada. Todo o esforço para realização das tarefas do processo é provido pelos operadores.

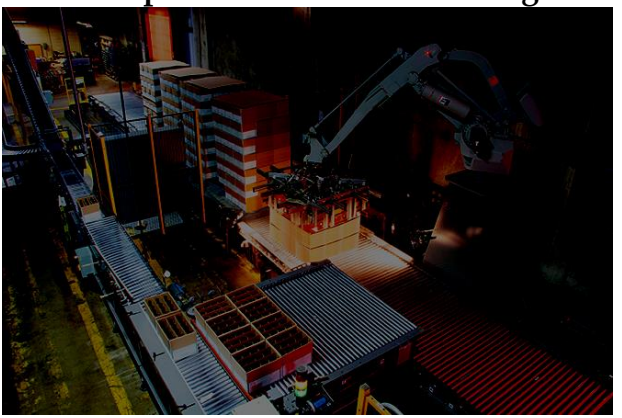
2.1.2. Trabalho mecanizado

No sistema de produção mecanizado ou sistema trabalhador-máquina, a fabricação do produto ou realização do serviço é feita por operadores que usam máquinas e equipamentos. As máquinas são usadas para diminuir o esforço feito pelo operador. Nesse sistema de produção são usados equipamentos de diversos portes, porém, sempre é necessário um ou mais trabalhadores para realizar o controle e operação. O equipamento não será capaz de realizar funções de forma autônoma.

2.1.3. Automatizado

No sistema de produção automatizado o trabalho é realizado por máquinas sem a participação direta de humanos. O sistema funciona de forma autônoma, sem que um operador precise controlá-lo para que as tarefas sejam executadas. Na figura 1 é mostrado o trabalho automatizado de transporte de carga. O robô organiza e empilha as caixas que chegam de uma esteira e deposita em outra.

Figura 1
Transporte automatizado de carga



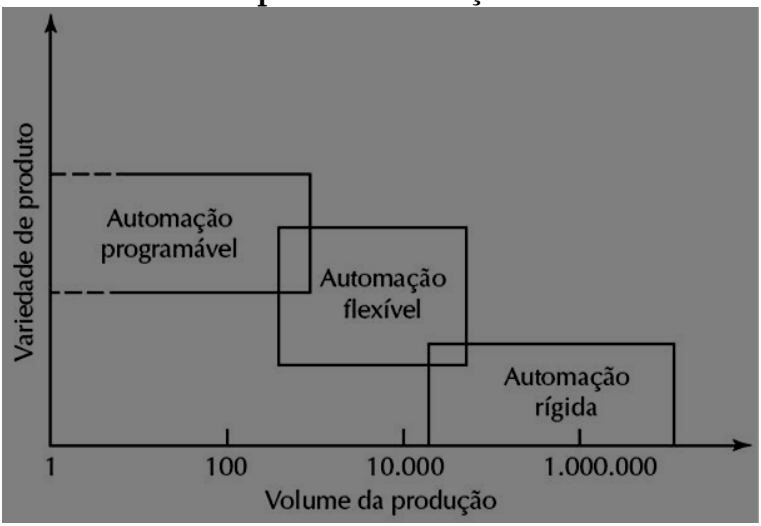
Fonte: Pixabay (2016).

Os sistemas de produção automatizados podem funcionar de duas formas: semiautomatizados e totalmente automatizados. Um processo semiautomatizado é capaz de realizar apenas um ciclo por vez, pois será necessária a intervenção de um humano para que o ciclo possa continuar ou ser iniciado novamente. O sistema de transporte da figura 1, por exemplo, será denominado semiautomatizado se for necessário a presença de um humano para retirar ou pôr as caixas na esteira a cada ciclo. Um sistema de produção totalmente automatizado é aquele capaz de executar vários ciclos sem qualquer intervenção humana.

Os sistemas de produção automatizados são classificados em três tipos: rígida, flexível e programável. Os três se distinguem, basicamente, pelo volume de produção e a variedade dos produtos dos processos que são

empregados. Na figura 2 é apresentado um diagrama que mostra a diferença entre os três tipos.

Figura 2
Variedade de produto x volume de produção nos tipos de automação.



Fonte: Groover (2011).

2.1.3.1. Automação rígida

A automação rígida é caracterizada pelo grande volume de produtos fabricados. Ela é utilizada em sistemas de produção contínua e possui programação e controle da produção simples, pois as máquinas são projetadas para um produto específico ou com pequenas variações. Esse tipo de automação está presente em linhas de engarrafamento de bebidas (figura 3), por exemplo.

Figura 3

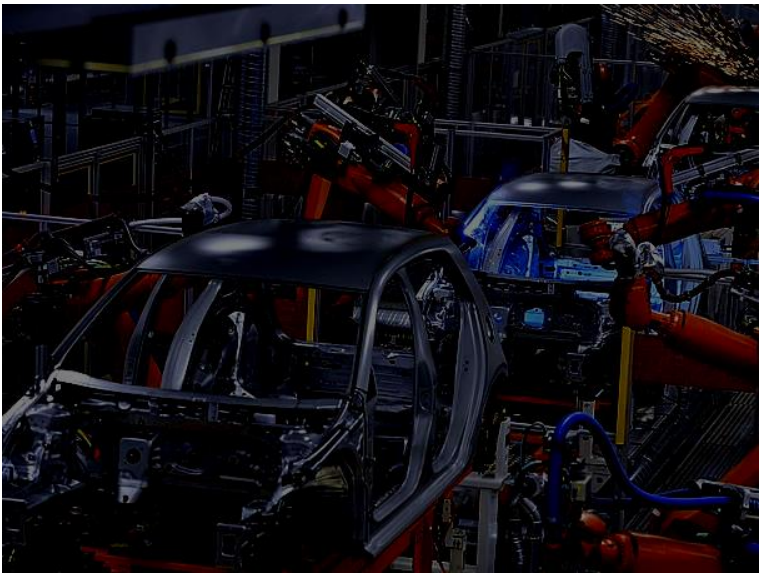
Linha de engarrafamento de bebidas.

Fonte: Wikimedia (2016).

2.1.3.2. Automação flexível

A automação flexível está presente em arranjos físicos onde o volume de produção é médio. Nela as máquinas produzem vários produtos, ainda que semelhantes. Dessa forma há a necessidade de reprogramação mesmo que seja simples. Esse tipo é empregado em linhas de montagens de automóveis, por exemplo. Na figura 4 é mostrada a linha de montagem robotizada de um chassi de carro. Quando um novo modelo de carro é fabricado os robôs, e CLPs do processo são reprogramados.

Figura 4
Linha robotizada de montagem.



Fonte: Wikimedia (2016).

2.1.3.3. Automação programável

A automação programável, caracterizada pelo baixo volume de produção e alta variedade de produtos. Isso requer frequente reprogramação das máquinas. É o tipo de automação onde mais são empregadas a máquinas CNC e os robôs industriais (CARRARA, 2008). Dessa forma os robôs demonstram suas principais características: extremamente adaptável a diferentes produtos e operações. Na figura 5 é mostrado um robô equipado com ferramentas de corte usinando uma peça. Para fabricar outra peça ele

será reprogramado e, provavelmente, a ferramenta de corte será trocada por outra adequada a próxima peça.

Figura 5
Robô equipado com fresadora usinando peça complexa

Fonte: Wikimedia (2016).

2.2. ROBÓTICA INDUSTRIAL

Um robô industrial, conforme Romano (2002, apud ISO 10218 1998, p.11) é uma "máquina manipuladora com vários graus de liberdade controlada automaticamente, reprogramável, multifuncional, que pode ter base fixa ou móvel para utilização em aplicações de automação industrial".

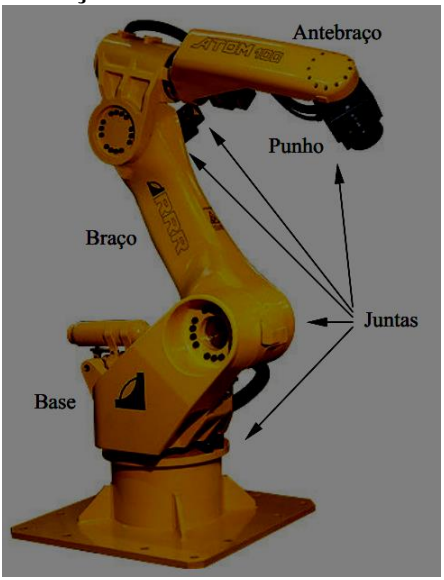
2.2.1. Manipuladores robóticos

Convencionalmente um robô industrial é formado por um manipulador, um efetuador (órgão terminal), um sistema de controle e um conjunto de sensores. A função primária do braço robótico é executar movimentos para deslocar ferramentas de acordo com a programação inserida no seu controlador e ser capaz de repeti-las.

> Um robô é um sistema mecânico, de geometria variada, formada por corpos rígidos, articulados entre si, destinados a sustentar e posicionar/orientar o órgão terminal, que, dotado de garra mecânica ou ferramenta especializada, fica em contato direto com o processo. A mobilidade do manipulador é o resultado de uma série de movimentos elementares, independentes entre si, denominados graus de liberdade do robô. (ROSÁRIO, 2005, p. 51 apud ARMADA, 1995)

Como sugerido pelo próprio nome, o braço robótico tem várias semelhanças com o membro humano e a nomenclatura de suas partes são dadas com base nisso, como pode ser visto na figura 6.

Figura 6
Braço robótico RRRobotica



Fonte: Carrara (2008).

O que diferenciará um braço do outro serão suas configurações e algumas características, como: volume de trabalho, sistema de acionamento, sistema de controle, desempenho e precisão, órgão terminal, sensores e programação.

O órgão terminal (efetuador) será a ferramenta utilizada pelo robô para executar tarefas, que pode ser uma garra, uma ventosa de sucção, ganchos, colheres, imãs ou eletroímãs.

Os sistemas de acionamentos de um robô geralmente são hidráulicos, elétricos ou pneumáticos. Apesar dos acionamentos elétricos apresentarem vantagens

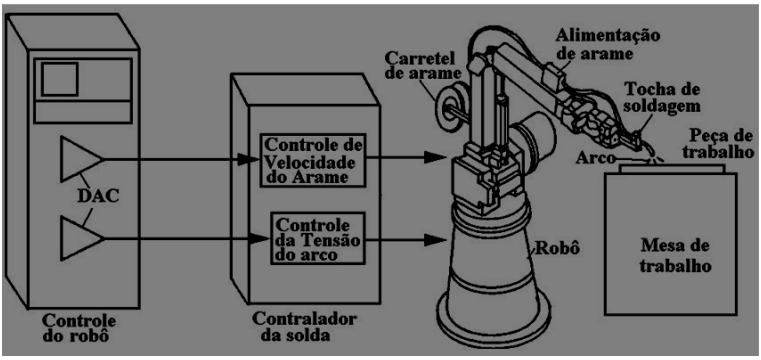
com relação ao custo, os hidráulicos são utilizados quando exigido grande capacidade de carga e os pneumáticos quando necessita-se de agilidade no processo. Para pequenas cargas geralmente são utilizados servomotores ou motores de passo, devido a razoável flexibilidade das suas formas de controle. A forma como o motor irá operar dependerá do que se deseja controlar.

O sistema de controle de um robô é realizado por meio de seu *software* e *hardware*. Basicamente, ele processa os sinais de entrada e fornece sinais de saída de acordo com a sua programação. Concomitante a esse processo, ele receberá estímulos de seus sensores externos que podem vir a mudar o curso da tarefa a depender do seu algoritmo.

Fundamentalmente robôs industriais possuem dois tipos de aplicação: transporte de materiais e fabricação. Em 1998, conforme Rosário (2005), 65% dos braços robóticos do Brasil eram empregados na indústria automobilística. Utilizados principalmente na soldagem por resistência por pontos. Isso demonstra que é um recurso não utilizado por toda indústria, seja por questões financeiras, necessidade ou pela alta taxa de amortização do país.

Na figura 7 é mostrado de forma simplificada a interação entre o robô, seu controlador, seu efetuador, o controlador do efetuador e o ambiente de trabalho.

Figura 7
Componentes essenciais de um sistema de robô com solda MIG



Fonte: Octans automação (2005).

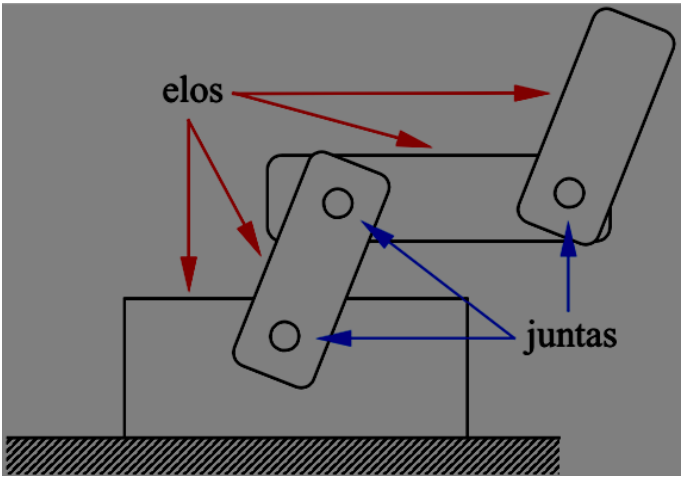
O controlador do robô possui o algoritmo do processo e é responsável pela movimentação do robô. Ele envia o sinal para os drivers de potência que alimentam os atuadores das juntas. Neste caso, o efetuador do robô é uma tocha para realizar operações de soldagem MIG. Há um controlador apenas para os parâmetros de soldagem (velocidade do arame, tensão do arco, corrente etc). Apesar dos dois controladores trabalharem em conjunto eles podem ser de fabricantes diferentes.

2.2.2. Juntas robóticas

O manipulador robótico é formado por elos acoplados por meio de juntas para realizarem movimentos individualmente. Na figura 8 é mostrada a relação dos elos e as juntas. O elo que estiver mais próximo da base é

denominado elo de entrada e o que estiver mais próximo do órgão terminal será o elo de saída.

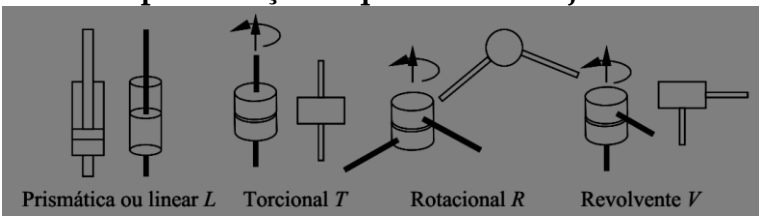
Figura 8
Esquema de notação de elos e juntas num braço mecânico ilustrativo



Fonte: Carrara (2008).

As juntas são as articulações entre as partes que se movem do robô. Há vários tipos para vários tipos de movimento que o robô execute. As juntas podem ser classificadas de acordo com as direções dos elos de entrada e saída em relação ao eixo de rotação, como visto na figura 9.

Figura 9
Representação esquemática das juntas



Fonte: Carrara (2008).

Junta linear (L): Não há eixo de rotação, a junta se desloca linearmente;

Junta torcional (T): Os elos de entrada e de saída têm a mesma direção do eixo de rotação da junta;

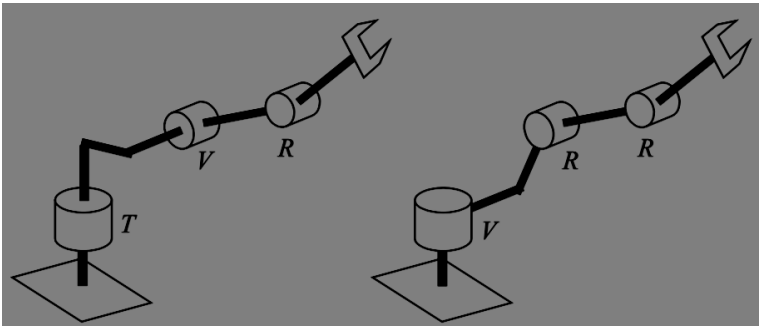
Junta rotacinonal (R): Os elos de entrada e saída são perpendiculares ao eixo de rotação da junta;

Junta revolvente (V): O elo de entrada possui a mesma direção do eixo de rotação. Já o elo de saída é perpendicular ao eixo de rotação.

Manipuladores robóticos podem apresentar diversas configurações. Cada configuração é determinada pelos movimentos relativos das três primeiras juntas que posicionam o efetuador (CRUZ, 2007).

Na figura 10 são mostradas duas configurações distintas de braço robótico, mas que executam os mesmo movimentos. A primeira configuração é do tipo TVR (torcional, revolvente e rotacional) já a segunda é VRR (revolvente, rotacional e rotacional)

Figura 10
Duas configurações distintas com movimentação idêntica



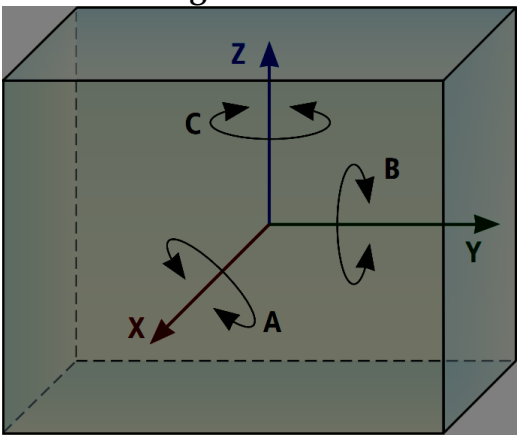
Fonte: Carrara (2008).

2.2.3. Graus de Liberdade

O grau de liberdade (GdL) representa a possibilidade de movimento. Se uma junta tem um grau de liberdade ela será capaz de se movimentar de uma única forma. A junta rotativa, por exemplo, pode apenas rotacionar sobre o eixo que está vinculada. Segundo Romano (2002), "o grau de liberdade é o número mínimo de variáveis independentes de posição que precisam ser especificadas para definir inequivocamente a localização de todas as partes de um mecanismo".

Na figura 11 pode-se observar três graus de liberdade para posicionamento das peças, em que A, B e C são graus de liberdade para a rotação da peça e X, Y e Z para translação.

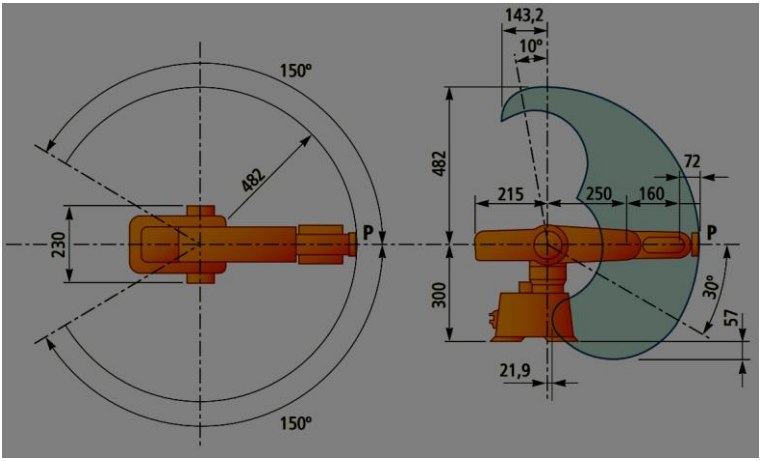
Figura 11
Eixo dos graus de liberdade



Fonte: Vianna, Angelo e Angelo (2011).

Os graus de liberdade definem se o movimento de um braço robótico será apenas em 2D ou em 3D também. Eles determinam o espaço operacional do robô, isto é, o espaço geométrico que o robô pode alcançar. Na figura 12 é mostrado o espaço operacional de um robô com cinco graus de liberdade.

Figura 12
Espaço operacional de robô com cinco graus de liberdade



Fonte: Vianna, Angelo e Angelo (2011).

2.2.4. Programação de robôs industriais

Para que o robô atue corretamente será necessário informar a trajetória pretendida, então um metódo de programação coerente com o processo deve ser escolhido. Sabendo-se da trajetória, o controlador deve interpretar estes dados e acionar os atuadores. Dessa forma o programador deve selecionar uma linguagem de programação de acordo com o *hardware* usado.

2.2.4.1. Métodos de programação

De forma geral existem métodos de programação *online* e *off-line*. Os métodos *online* são todos aqueles que

necessitam do robô funcionando para ser programado. Já os métodos *off-line* não precisam necessariamente do robô ou do sistema para que a programação seja desenvolvida, podem ser realizadas através de simulações computacionais.

Dos métodos de programar as mais comumente usadas são:

- Programação por aprendizagem;
- Programação por linguagem textual;
- Programação mecânica;
- Programação de célula de trabalho.

Todos os sistemas de controle de manipuladores robóticos contam com recursos de programação em linguagem textual, mesmo que não seja fornecida pelo fabricante. Esse tipo de programação consiste numa série de comandos sequenciais: um algoritmo. São definidos os pontos da trajetória e comandos específicos como abrir ou fechar a garra, por exemplo. Essa linguagem se assemelha a usada em máquinas CNC, pois o operador deve definir o trajeto que o robô deve percorrer (CARRARA, 2008, p.45).

Um exemplo de um programa para uma operação de deslocamento ponto-a-ponto, escrita em linguagem textual seria:

1. Move to P1 (mover garra para posição P1)
2. Move to P2 (mover garra para posição P2)
3. Move to P3 (mover garra para posição P3)
4. *Close gripper* (fechar a garra; prende objeto)
5. Move to P4 (mover garra para posição P4)
6. Move to P5 (mover garra para posição P5)

7. Open gripper (abrir a garra; liberar objeto)
8. Move to P1 and finish (voltar ao ponto inicial e finalizar programa)

A programação em linguagem textual é *off-line*. O programador pode criar todas aquelas linhas de código sem que seja necessário a utilização do robô. Utilizando esse método, o operador vai guiando o robô para que ele descreva o movimento desejado. Ao final do processo, ele registra os pontos percorridos e faz com que o software de controle do robô reproduza os movimentos "aprendidos".

A programação mecânica aplica-se a robôs de sequência fixa que são acionados por fim-de-curso, cames e chaves elétricas de contato, conforme Carrara (2008). É a forma mais simples e menos custosa de programação e é empregada junto com atuadores pneumáticos, que são acionados da mesma forma.

A programação de célula de trabalho é uma programação de controle inteligente, na qual o controlador pode tomar decisões autônomas com base na informação dos sensores. Essas decisões podem alterar a execução ou o ciclo de trabalho do robô.

A programação por aprendizagem, a mais comum dentre todas, consiste em posicionar o braço robótico nos pontos da trajetória desejada.

Podem ser:

- Aprendizagem direta (*walk through*);
- Apredizagem por simulação física;
- Aprendizagem por telecomando;
- Programação PTP (*point to point*)

- Programação Mestre-escravo (*Master-Slave*).

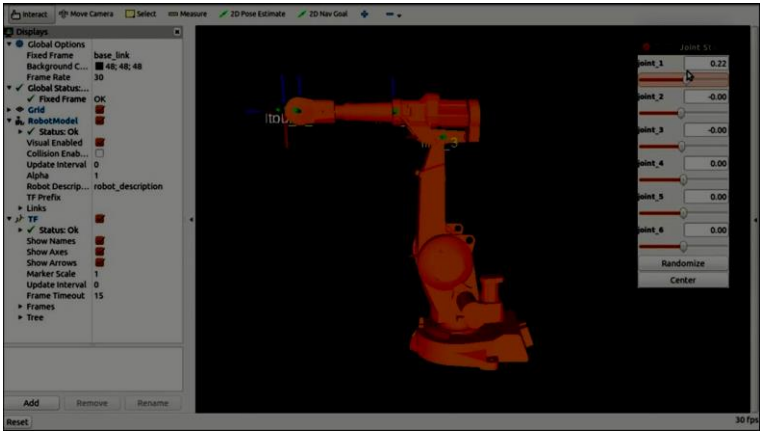
Na aprendizagem direta o operador guia fisicamente o robô pelo seu efetuador. Enquanto isso os sensores das juntas são utilizados para memorizar os pontos mais importantes. Esse método é indicado para robôs de estrutura leve e ambiente não hostis (ROSÁRIO, 2010, p.323). Na figura 13 pode ser visto um operador guiando um robô.

Figura 13
Operador guiando robô. Exemplo de aprendizagem direta

Fonte: Robotics Tomorrow (2017).

Na aprendizagem por simulação física é usado um modelo virtual do robô, que contém os dados de sua geometria e graus de liberdade; um modelo detalhado criado em um *software* CAD. Esse método é indicado para robôs de grande porte e ambiente hostis (ROSÁRIO, 2010, p.324). Para simulação, comumente são usados *softwares* CAM, como pode ser visto na figura 13.

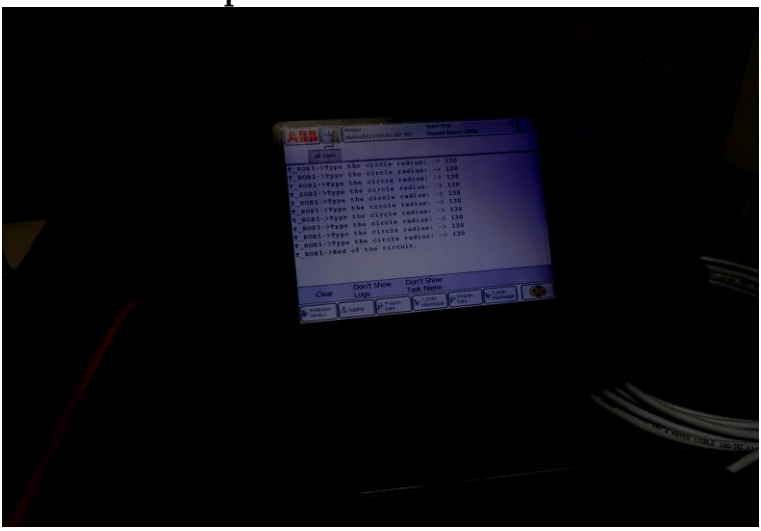
Figura 14
Simulação de manipulador da ABB no software CAM ROS



Fonte: Mastering ROS (2017).

Na aprendizagem por telecomando é utilziado um dispositivo de telecomando (*teach pedant* ou *teach-in-box*) para mover cada junta do robô isoladamente ou fornecer a posição e orientação para o efetuador (ROSÁRIO, 2010, p. 324). Este método é adequado a quaquer tipo de robô e o mais barato entre os já citados. Na figura 15 é mostrado um modelo de *Teach Pad* (TP).

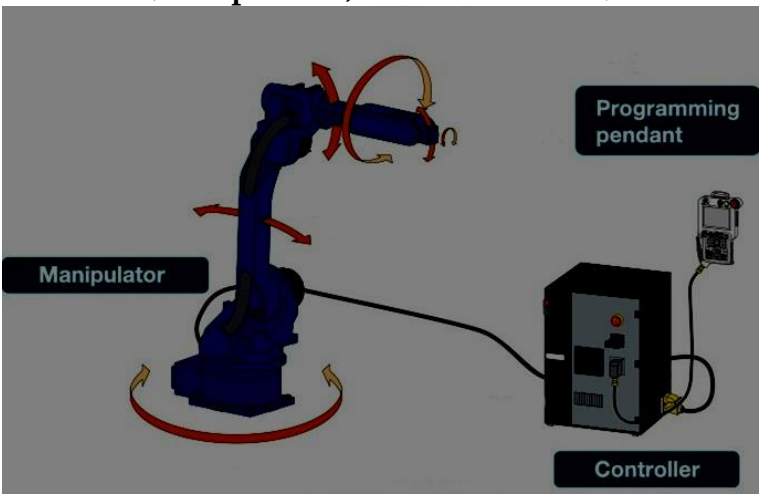
Figura 15
Teach pedant ABB modelo IRB140



Fonte: Auledas(2015).

Na figura 16 é mostrado como os componentes do robô são interligados. O *teach pedant* envia informações para o controlador, que, por sua vez, interpreta os comandos de posição e orientação e aciona os motores do manipulador. No manipulador, os sensores de posição de cada junta verificam se a posição desejada foi alcançada e envia esses dados para o controlador, que fará os devidos ajustes se necessário.

Figura 16
Interligação dos componentes do robô (manipulador, controlador e TP)

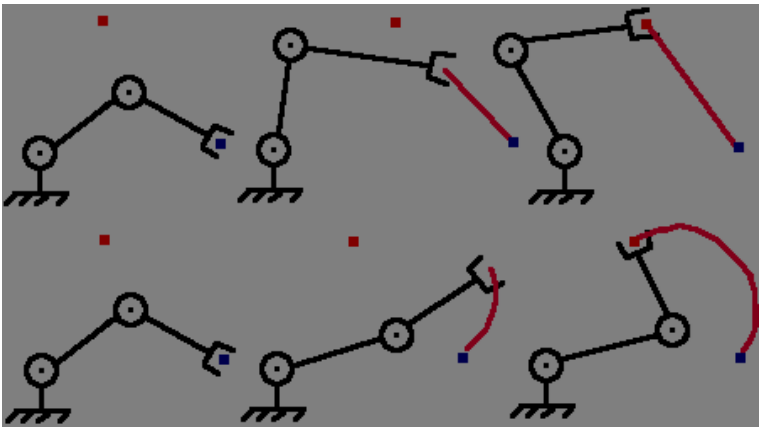


Fonte: Yaskawa (2017).

Na programação PTP é ensinado ao robô os seus pontos de passagem, que consiste na movimentação da juntas a partir de uma determinada posição até a seguinte. Neste método são gravados apenas os pontos essenciais da trajetória do efetuador. Este é um método indicado para aplicações que não requeiram um controle preciso da trajetória e velocidade intermediárias aos pontos essenciais de uma tarefa (ROSÁRIO, 2010, p.327). Robôs programados com esse método são conhecidos como *playback robots*, termo que caracteriza sua capacidade de executar trajetórias salvas.

De acordo com Rosário (2010, p.327), “para garantir a sincronização de movimentos das juntas, todos os graus de liberdade iniciarão e completarão os seus movimentos simultaneamente, através de velocidade máxima de junta indicada pelo operador”. O deslocamento de um ponto a outro ainda pode ser por interpolação linear (linha reta entre os pontos) ou circular (arco entre os pontos), como pode ser visto na figura 17.

Figura 17
Interpolação linear e circular de manipulador robótico



Fonte: Society of Robots (2014).

Convém observar que existe uma área de estudo em robótica dedicada a movimentação dos robôs. Segundo Cabral (2004), “a cinemática de um robô manipulador é o estudo da posição e velocidade do seu efetuador e de seus ligamentos”. O objetivo da modelagem cinemática é a obtenção das esquações de posição e pode ser dividida em:

direta e inversa (ida e volta). Na cinemática direta o operador fornece os ângulos das juntas e o robô vai para onde foi indicado. Na cinemática inversa o operador fornece o ponto onde se quer chegar e a posição atual do robô, afim de que o sistema calcule os ângulos das juntas.

Na programação Mestre-Escravo, o robô (escravo) aprende os movimentos que deve realizar a partir da movimentação de um sistemas robótico mestre. Esse modo de aprendizagem é muito usado em operações médicas robotizadas a distância e intervenções submarinas automaizadas. Na figura 18 é mostrado um exemplo desse método de programação.

Figura 18
Modo de aprendizagem Mestre/Escravo

Fonte: RASC (2015).

Além dos métodos convencionais estão surgindo novos métodos de programação de robôs usando as

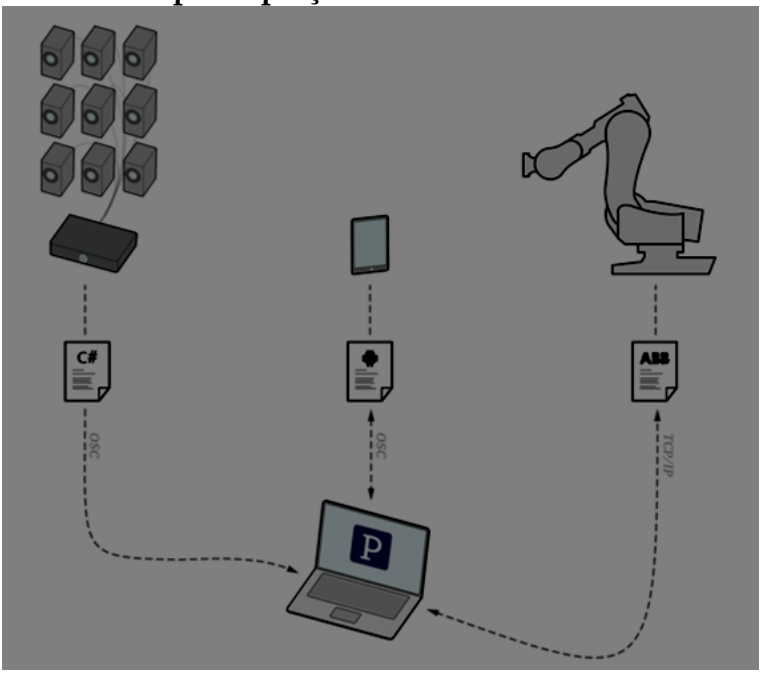
tecnologias emergentes. Uma dessas tecnologias é a captação de movimentos.

Pesquisadores do MADLAB, que trabalham no desenvolvimento de novas formas de comunicação com máquinas, criaram o Quipt, um *software* de controle para robôs industriais baseado em gestos. Os sistema de captura trabalha em conjunto com marcadores vestíveis (*wearables*) usados pelo programador. Dessa forma evita-se que o robô imite os movimentos de outra pessoa, tornando o método mais seguro e intuitivo.

A equipe do MADLAB[1] usou o manipulador ABB ARB 6700 em conjunto com os captadores inteligentes de movimento Vicon e o *software* de código aberto Robo.Op[2], usado para converter os movimentos captados em comandos para o robô. Esse sistema também possui um aplicativo para android, o qual o operador pode acompanhar o processamento dos dados durante o processo. Na figura 19 é mostrado a intereração entre esses elementos.

[1]http://www.madlab.cc/work
[2]http://www.madlab.cc/robo-op

Figura 19
Componentes usados no método de aprendizagem por captação de movimentos



Fonte: MADLAB (2017).

2.2.4.2. Linguagens de programação

A programação será a responsável por interfacear os comandos externos de posição e orientação com os atuadores do robô. As linguagens de programação podem ser classificadas, conforme Rosário (2010, p.332) de acordo com:

- **Nível de junta:** programação que requer a programação individual de cada junta para a posição seja alcançada;
- **Nível do manipulador:** nesse nível é necessário fornecer apenas a posição e a orientação do efetuador, pois o sistema obtém as posições cada de junta pelo modelo geométrico do robô;
- **Nível do objeto:** São necessárias apenas as especificações relativas ao posicionamento dos objetos no interior do volume de trabalho do robô. Para isso é necessário haver um modelo matemático do ambiente de trabalho do robô;
- **Nível do objetivo:** Nesse nível é apenas definida, como por exemplo, Montar peças A, B e C. Será necessário, além do modelo matemático do ambiente, um conjunto de dados relativos à tarefa.

2.3. SISTEMAS DE SUPERVISÃO E AQUISIÇÃO DE DADOS

2.3.1. Principais características

Um sistema de supervisão serve como interface para que um operador possa controlar variáveis do processo e acionar dispositivos remotamente. Um sistema supervisório também pode ser definido como uma interface de fácil leitura com o objetivo de converter dados do processo de produção em gráficos ou em “telas amigáveis” (LOPES, 2009, p.6).

Os sistemas supervisórios são usados em processos que precisam ser monitorados e/ou precisam ter seus dados analisados. Todas as informações são armazenadas em bancos de dados e os resultados são apresentados aos operadores e gestores do processo. Na figura 20 é mostrada a tela do supervisório de um sistema de geração de energia eólica.

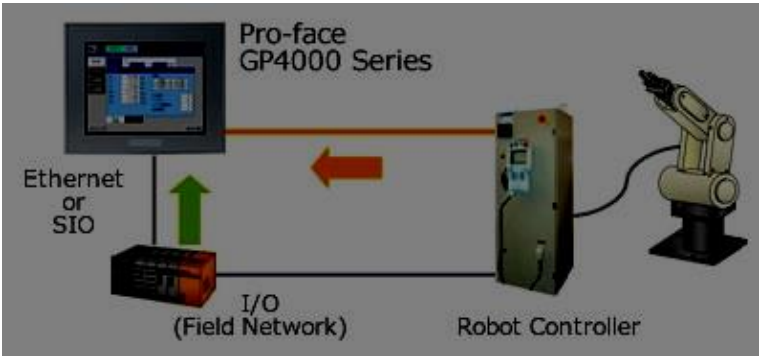
Figura 20
Tela do supervisório de um sistema de geração de energia eólica

Fonte: ScadaBR (2012).

A tela do supervisório apresentado na figura 20 mostra os valores de potência, tensão gerada, potência reativa, potência ativa, corrente e outras variáveis do processo de geração de energia. Observa-se na parte esquerda da tela as ações que o operador pode executar: Controle da potência máxima, controle de reativo, configuração de *restart* e liga/desliga do sistema.

Os sistemas supervisórios também são conhecidos como SCADA ou IHM. Usualmente o termo IHM (Interface Homem Máquina) é usado para descrever os painéis de acesso e acionamento dos equipamentos localizados no chão de fábrica. Eles são usados para monitoramento, controle de máquinas, processos e instalações industriais. Os seus principais componentes são o visor (*display*), teclas e botões (para navegação e inserção de dados), barramentos para placas de expansão e portas de comunicação (PAREDE, GOMES e HORTA, 2011). Na figura 21 é mostrado um IHM interfaceando com o controlador de um robô.

Figura 21
Interfaceamento entre IHM e controlador do robô



Fonte: Pro-face (2012).

Quando um sistema supervisório automaticamente lê os dados fornecidos pelo processo, interpreta-os e em seguida atua no processo, corringindo possíveis alterações de forma inteligente, ele é chamado de SCADA

(*Supervisory Control And Data Acquisition* - Controle Supervisório e Aquisição de Dados) (LOPES, 2009, p.7).

Um sistema supervisório pode ser instalado desde pequenos painéis junto ao processo até estações de controle localizadas em departamentos mais distantes. As redes e protocolos industriais são elementos chave na transmissão de informação da planta para o supervisório.

O mercado dispõe de várias soluções em sistemas supervisórios, desde *softwares* até os *hardwares*, cada um com seus recursos. É necessário escolher o sistema apropriado com base nas características do processo em que será implantado. Alguns dos critérios observados são:

- **Conectividade:** Refere-se a capacidade do sistema poder integrar componentes de diversos fabricantes. Para isso foram desenvolvidos os protocolos como serial, OPC, ODBC, XML e TCP/IP. Alguns fabricantes de CLP fornecem produtos que se integram apenas com os equipamentos da de sua marca. Já existem fabricantes que disponibilizam produtos compatíveis com muitos outros;
- **Arquitetura aberta:** Eventualmente o supervisório precisará trabalhar com outros sistemas já existentes. Em uma arquitetura aberta o usuário tem acesso direto à base de dados para desenvolver novas interfaces de comunicação. Uma arquitetura proprietária é aquela em que a base de dados é fechada e controlada pelo fabricante. Dessa forma o usuário deve recorrer ao fabricante para novas

implementações ou fazer uso apenas de seus produtos;

- **Facilidade de uso:** Sistemas com muitos recursos acabam por se tornar mais complexos, apesar dos fabricantes tentarem fornecer soluções mais intuitivas. Isso influencia diretamente no custo da mão de obra contratada. Sistemas de fácil utilização são melhores de assimilar e de realizar manutenção durante as fases de desenvolvimento;
- **Escalabilidade:** Refere-se a flexibilidade do sistema para receber futuras expansões e integrar-se com estações IHM e SCADA. Por isso é importante que o sistema seja compatível com diferentes plataformas (sistemas operacionais), isto é, por exemplo, consiga ser executados no Windows ou sistemas baseados em GNU/Linux;
- **Adequação ao processo:** Todo sistema de supervisão apresentará diferentes níveis de uso (administrador e usuário) e diferente limitações que influenciarão no seu custo. Com relação às adequações do processo, as principais características observadas são:

    - Quantidade de pontos (analógicos e digitais) a serem controlados/monitorados;
    - Número de estações de supervisão componentes do sistema;
    - Quantidade e tipo dos elementos de controle (CLPs) que devem se comunicar com o sistema de supervisão;

- Necessidade de implementar modificações na aplicação (licença de engenharia) ou apenas de executar aplicações previamente configuradas (licença de runtime).

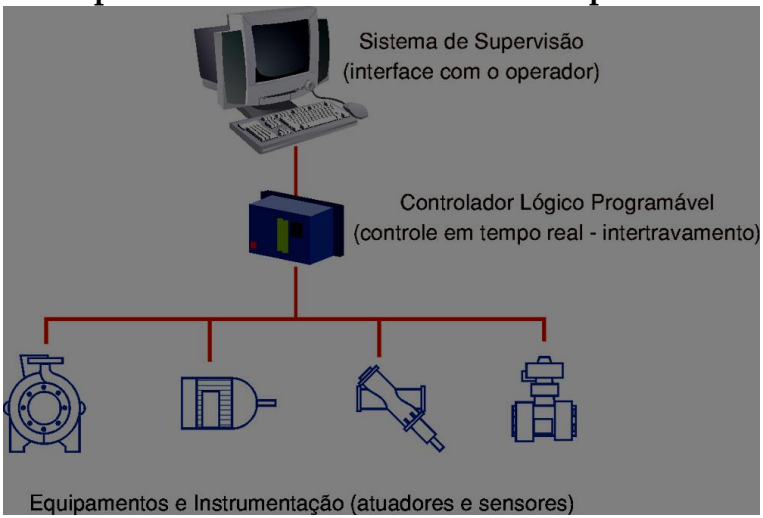
2.3.2.Arquitetura de um Sistema supervisório

A arquitetura de um sistema diz respeito como o ele é subdividido e organizado. Um supervisório possue vários componentes, os quais desempenham funções essenciais no sistema.

2.3.2.1. Componentes físicos

Os componentes físicos são os sensores e atuadores, rede comunicação, estações de controle e monitoramento central, como pode ser visto na figura 22.

Figura 22
Componentes físicos de um sistema supervisório



Fonte: Mecatrônica Atual (2004).

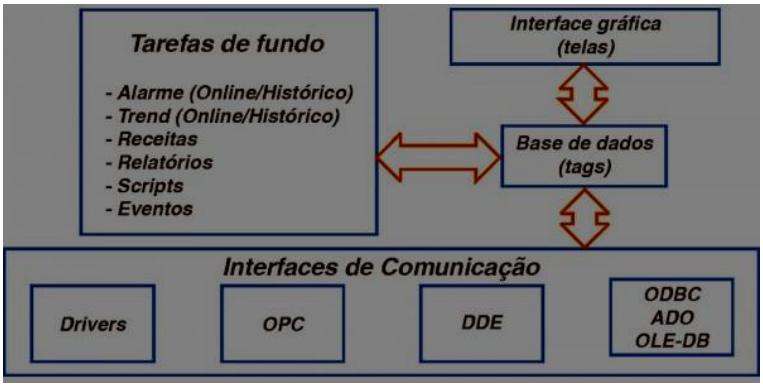
2.3.2.2. Componentes lógicos

Os componentes lógicos processam os dados captados e executam as tarefas programadas. Ele pode ser dividido em três módulos que são responsáveis pelas principais tarefas. O sistema de aquisição e dados, por exemplo, converte as informações dos sensores em informações que podem ser interpretadas pelo sistema.

Na figura 23 são destacados os principais componentes lógicos de um sistema supervisório.

Figura 23

Componentes lógicos de um sistema supervisório



Fonte: Mecatrônica Atual (2004).

As TAGs simbolizam as variáveis que serão manipuladas pelo sistema. Para facilitar a identificação dos dispositivos de campo e de controle são utilizadas simbologias e nomenclaturas para melhor identifica-los, como funciona a simbologia de instrumentação.

As telas representam cada processo ou unidade do processo. Ela permite ao operador visualizar as informações e acionar os equipamentos. A representação gráfica do processo pode ser feita com objetos estáticos, como a imagem de motores e bombas, ou por objetos dinâmicos, como as barras de deslizamento, *bargraph* e etc. Elas devem representar o processo de maneira intuitiva para o operador.

As tarefas de fundo são funções específicas que rodam em *background*, isto é, são executadas em paralelo com outras funções. Normalmente são executadas alarmes, receitas, relatórios, scripts e eventos.

As interfaces de comunicação são responsáveis por conectar o sistema de supervisão com elementos externos. Esses elementos podem ser: drivers e protocolos.

2.3.3. Protocolos de comunicação

Um protocolo de comunicação é um conjunto de regras que estabelecem o envio e o recebimento de dados entre equipamentos. Cada equipamento deve estar preparado para se comunicar utilizando determinado protocolo. Por esse motivo foram criados protocolos abertos. Isso possibilita que equipamentos de diferentes fabricantes possam se comunicar sem qualquer incompatibilidade, pois todos usam o mesmo conjunto de regras de envio e recebimento de dados.

Os protocolos industriais surgiram a partis dos protocolos usados nas redes de computadores, porém ambos possuem características diferentes. A principal delas é o fato das redes de computadores serem probabilísticas e das redes industriais serem determinísticas.

Uma rede probabilística não possui tempo exato para trafego de informações. Isso significa que uma informação enviada de um ponto a outro pode chegar em pouco tempo ou não; existe uma probabilidade dele chegar rapidamente.

Uma rede determinística possui tempos exatos para tráfego das informações. Nos sistemas industriais as informações são fundamentais, pois muitos processos são controlados a partir das variáveis medidas. A informação da variável medida deve chegar ao controlador em um tempo determinado, caso contrário o processo apresentará problemas.

2.3.3.1. Transmissão serial de informação

Esse tipo de comunicação é o mais comum, pois vários meios de comunicação têm adotado esse padrão, desde a comunicação entre máquinas, sensores e sistemas de supervisão de uma indústria, até a comunicação entre vários computadores pelo mundo. Sua principal vantagem é a quantidade de fios usados para o transporte de dados, o que possibilita a comunicação entre longas distâncias a baixo custo. Sua principal desvantagem é que dependendo do sistema e distância percorrida pelos dados, pode-se ter um atraso considerável entre os dispositivos do sistema.

Outro ponto importante é que a transmissão serial também pode ser efetuada através de sistemas sem fio, onde a comunicação ocorre por ondas de rádio ou radiação infravermelho. Um exemplo onde encontramos esse tipo meio físico para transmissão de dados são:

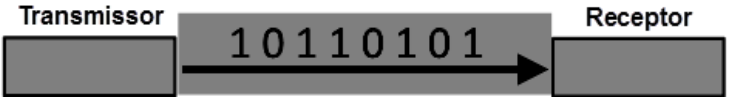
- Rádio frequência: Roteadores Wireless, Controle remoto de portão, Controle de aeromodelos, automodelo e nautimodelos, etc;

- Infravermelho: Controle de som, controle de luminosidade em sistemas inteligentes, controle de ar-condicionado, etc.

Durante a comunicação serial os dados precisam ser transferidos na maioria das vezes através de apenas dois fios, a essa característica dar-se o nome de transmissão serial.

Os bits que compõe o byte são serializados e transmitidos um a um, como mostrado na figura 24. As informações terão de ser transmitidas bit após bit com taxas de transmissão, indicação de início e de finalização bem estabelecidos, para que ambos os lados saibam quando o outro deseja se comunicar.

Figura 24
Modo de transmissão serial



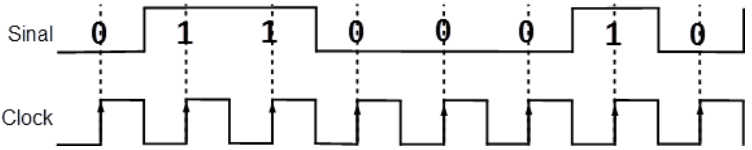
Fonte: Produzido pelo autor (2016).

Para garantir o envio e chegada dos dados corretamente, a transferência de bits entre emissor e receptor deve ser sincronizada.

Na forma de transmissão síncrona, um sinal de *clock* separado é associado aos dados no momento da transmissão. Esse *clock* é necessário para que ambos os lados (Transmissor e Receptor) possam reconhecer o início da transmissão, o momento em que os dados são enviados e o momento em que deve encerrar a comunicação

(NOERGAARD, 2005, p. 258). Na figura 25 é mostrado como o sinal da transmissão síncrona se comporta.

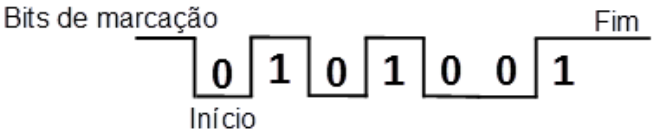
Figura 25
Transmissão síncrona de sinais



Fonte: Registrado pelo autor (2016).

Na transmissão assíncrona são necessários todos os requisitos acima, no entanto, para que não haja perda de dados devido a falta do clock, são ajustados o tempo para que ocorra o processo de envio e de recebimento durante a comunicação. Na figura 26 é mostrado como o sinal da transmissão assíncrona se comporta.

Figura 26
Transmissão assíncrona de sinais



Fonte: Registrado pelo autor (2016).

O tipo de transmissão de dados dependerá do *hardware* instalado no equipamento. O dispositivo instalado podem ser (FLORENCIO, 2016, p. 17):

- USRT (*Universal Synchronous Receiver/Transmitter* ou Transmissor/Receptor Universal Síncrono):

sincronização feita por *hardware*, o que acarreta em distâncias muito curtas;

- UART (*Universal Asynchronous Receiver/Transmitter* ou Transmissor/Receptor Universal Assíncrono): sincronização feita por *software*, o que acarreta em uma velocidade de transmissão limitada;
- USART (*Universal Synchronous/Asynchronous Receiver/Transmitter* ou Transmissor/Receptor Universal Síncrono e Assíncrono): é um dispositivo versãtil que pode ser usado dos dois modos.

2.3.3.2. Padrões de comunicação serial

Existem alguns tipos de comunicações em que apenas dois dispositivos estão presentes em uma rede, chamada de comunicação ponto-a-ponto (RS 232, por exemplo). Existem outras em que apesar de vários dispositivos estarem presentes apenas um pode iniciar a conversa e os outros devem apenas responder quando solicitados, a esse tipo de rede é chamada de conexão Mestre-Escravos (RS 485 / RS 422, por exemplo). Existem ainda outras onde todos os dispositivos da rede se comunicam de forma aleatória, sendo necessário para isso a implementação de um protocolo que dê suporte para a mesma, como por exemplo o Profibus.

Um padrão de comunicação serial muito usado é o *Universal Serial Bus* (USB). Esse é um padrão que permite a fácil conexão entre periféricos, além de fornecer alimentação elétrica, ampla compatibilidade, e conexão de vários aparelhos ao mesmo tempo (UNIVERSAL SERIAL BUS,2017). É usado numa enorme variedade de

dispositivos, desde pendrives até sistemas automotivos. Esse padrão foi criado para facilitar a conexão de dispositivos ao computador e acabou se tornando um padrão também adotados em outros dispositivos devidos a sua praticidade. Na figura 27 é mostrado alguns tipos de conectores USB.

Figura 27
Tipos de conecor USB. A partir da esquerda: micro, mini, tipo B, tipo A fêmea, tipo A macho

Fonte: Techtonic (2007).

2.4. SISTEMAS EMBARCADOS

2.4.1. Conceito e características

Segundo Raghavan, Lad e Neelakandan (2006, p.1), "um sistema embarcado é um computador com propósito específico, projetado para atuar em funções específicas das atividades designadas". Eles estão presentes no cotidiano das pessoas em dispositivos como: telefones residenciais, *smartphones*, *videogames*, TVs, carros (sistema de ignição, freio ABS), brinquedos, câmeras digitais, GPS, micro-ondas, geladeiras, roteadores de internet sem fio, aparelhos

de som, *tablets*, impressoras, *scanners*, cercas elétricas e muitos outros.

Devido as constantes inovações da informática, definir um sistema embarcado não é tão simples. Assim é mais apropriado citar suas características mais recorrentes (NOERGAARD, 2005, p. 5):

- **Sistemas embarcados são mais limitados em hardware e/ou software que os computadores pessoais:** este ponto é verdadeiro para uma parte significante dos sistemas embarcados existentes. Em termos de hardware a limitação pode ser de processamento, memória e consumo de energia, por exemplo. Para o software pode ser realizar apenas uma função, não ter muitas funções, não possuir sistema operacional (SO) ou possuir um SO limitado.
- **Um sistema embarcado é projetado para ser decicado a uma função:** a maioria é criada por esse motivo. Contudo, os *smartphones* atualmente podem executar muitas funções, diferente dos telefones móveis dos quais evoluiram. As TVs digitais (ou smartTVs) também possuem várias funções.
- **Um sistema embarcado possui maior requisito de confiabilidade do que outros sistemas computacionais:** Sistemas embarcados em equipamentos médicos e carros, de fato, possuem alta confiabilidade, caso contrário vidas correm perigo. Já *videogames, smartphones* e TVs não apresentam risco se não funcionarem corretamente.

Suas principais características são, em resumo, conforme Molloy (2016, p. 56):

- Tendem a ter funções específicas;
- Frequentemente tem limite de processamento, memória e armazenamento disponível;
- Geralmente fazem parte de um sistema maior;
- Possuem aplicações em sistemas críticos, devido a sua confiabilidade
- Frequentemente usado em sistemas de tempo real, os quais suas saídas estão diretamente relacionadas com as suas entradas (sistemas de controle, por exemplo);
- Consomem menos energia e possuem menor custo que os computadores pessoais (PCs).

2.4.2. Aplicações e modos de funcionamento de sistemas embarcados

De acordo com Cunha (2010, p.3), as aplicações para sistemas embarcados podem ser:

Propósito geral: aplicações semelhantes ao computadores de mesa, mas em sistemas embarcados. Costuma ter maior intereção entre o usuário e o sistema, por isso é mais usado através de terminais de vídeo e monitores. Com exemplo: videogames, conversores de TV a cabo, caixas de bancos e IHMs. Para essa aplicação os computadores em placa única são bastante utilizados;

Sistema de controle: usado em controles em malha fechada com realimentação em tempo real. Geralmente são

as aplicações mais robustas, com placas dedicadas e múltiplas entradas e saídas;

Processamento de sinais: envolve um grande volume de informação a ser processada em curto espaço de tempo. Os sinais a serem tratados são digitalizados através de ADs, processados, e novamente convertidos em sinais analógicos por DAs

Comunicações e redes: usado para chaveamento e distribuição de informações em sistemas de telefonia, telecomunicações e internet.

O modo de funcionamento do sistema embarcado está relacionado a forma como ele vai se comportar. Em sistemas embarcados de **controle em tempo real** há limite de tempo para executar as tarefas programadas. Esse tipo não depende de uma entrada para executar uma tarefa, pois é capaz tomar decisões baseadas na ausência da mesma. Já um sistemas embarcados **reativo** irá executar a tarefa programada como resposta a um evento externo. Não haverá limite de tempo para execuar a tarefa, porém deverá fornecer uma saída assim que receber um sinal de entrada.

2.4.3. Microcontroladores

O microcontrolador é um chip que contém memória de armazenamento (memória ROM), memória de acesso randômico (memória RAM), unidade central de processamento de dados (CPU), pinos de entrada e saída. Ele é sempre usado para executar funções específicas. O

microcontrolador possui todos os seus componentes integrados em um único chip (Figura 28).

Figura 28
Microcontrolador de 8 bits ATmega32

Fonte: Wikimedia (2009).

Os microcontroladores são aplicados tanto em equipamentos domésticos quanto em processos industriais de automação, controle e instrumentação. Dentre os fabricantes de microcontroladores, tem-se:

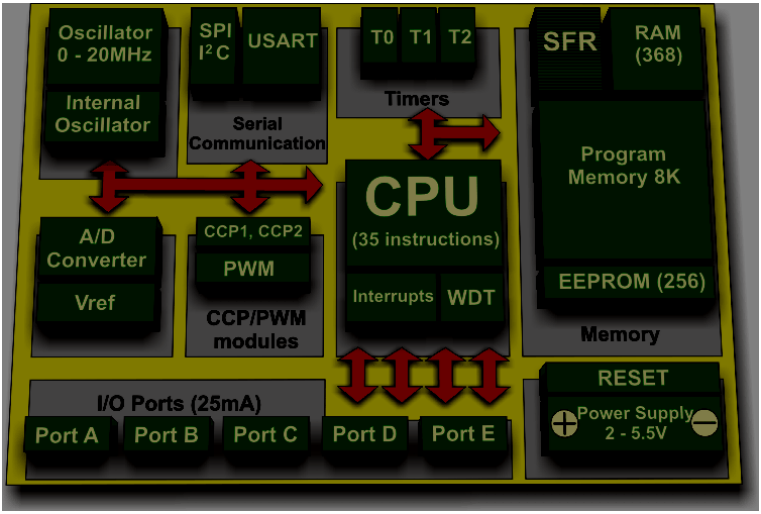
- Intel (8051);
- Motorola (HC908);
- Microchip (PIC);
- Texas Instruments (MSP);
- Mitsubishi;

- ATMEL (ATmega);
- Philips.

2.4.3.1. Arquitetura de um microcontrolador

Na figura 29 está representado o diagrama de blocos com a arquitetura do microcontrolador PIC16F877, fabricado pela Microchip®. Essa arquitetura terá variações dependendo do fabricante, pois terá mais ou menos blocos dependendo de sua finalidade, contudo possui os elementos básicos de todo microcontrolador. Todos esses recursos são essenciais para controle e comunicação com outros dispositivos.

Figura 29
Diagrama de blocos do microcontrolador PIC16F877



Fonte: Sena (2009).

Cada um desses blocos possui uma função específica:

- *Oscillator:* Pulso de clock que serve para dar sequência às atividades da CPU. Quanto maior seu valor, maoir seráa quantidade de comandos executados por intervalo de tempo;
- *PWM:* Bloco responsável pela Modulação por Largura Pulso. Dessa forma é possível ter saídas analógicas em um microcontrolador;
- *I/O Ports:* Portas de entrada e saída. Trabalham com até $5V_{CC}$;
- *Interrupts:* Bloco responsável por gerenciar as interrupções;
- WDT (*Watch Dog Time*): Bloco responsável por reiniciar o programa em caso de travamento;
- *Memory*: Blocos de memória de armazenamento permanente e temporário;
- *A/D Converter*: Bloco responsável pela conversão de sinal analógico em digital. A sua resolução vai depender do fabricante. Normalmente varia de ou a dez bits de resolução;
- USART, SPI e I^2C: Blocos responsáveis pela comunicação serial do microcontrolador. A USART é o componente reponsável por converter os dados originalmente disponíveis de forma pararela em serial. Ela realiza esse procedimento tanto na entrada quanto na saída de dados;
- *Timers*: Temporizadores.

2.4.3.2. Programação de microcontroladores

Para que o microcontrolador faça aquilo que é desejado pelo usuário ele deve possuir um programa que informe o que ele deve fazer, portanto o microcontrolador deve ser programado. Um programa é uma sequência de instruções a serem executadas. Para escrever um programa deve-se usar uma linguagem de programação. Por sua vez, uma linguagem de programação é um determinado conjunto de códigos, como se fosse um idioma, usado para instruir o microcontrolador das tarefas que ele deve executar.

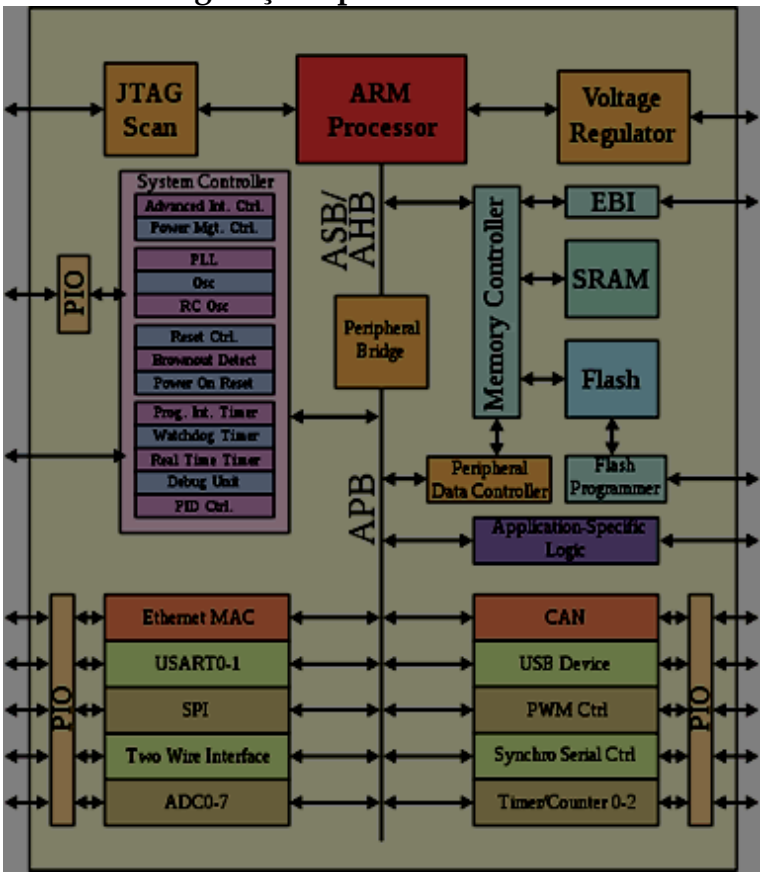
Normalmente os programadores usam linguagens que estão em um nível próximo da linguagem falada, chamada linguagem de alto nível. C ou C++ são exemplos de linguagens de alto nível usadas para programar microcontroladores. Contudo, o microcontrolador trabalha com linguagem de máquina, também chamada de linguagem de baixo-nível ou *Assembler*. Para converter um programa de alto nível para baixo nível usa-se um compilador. O texto com o programa numa linguagem de alto nível é chamado de código fonte.

Os compiladores estão integrados nos ambientes de desenvolvimento dos programas também chamado de IDE (*Integrated Development Environment*), do inglês: Ambiente de Desenvolvimento Integrado.

2.4.4. Computador em Placa Única

Computador em Placa Única ou *Single Board Computer* (SBC) são computadores que possuem todos os componentes necessários para o seu funcionamento em uma única placa eletrônica, cujo principal elemento é o SoC (System on a Chip – Sistema em um Chip). Segundo Maciel (2014), "SoC é um chip eletrônico que reúne todos os elementos básicos de um sistema computacional como CPU, memória principal (RAM), memória secundaria (ROM/FLASH) e alguns periféricos". Atualmente um SoC contém internamente diversos periféricos dos mais variados tipos como I^2C, SPI, UART, CAN, entre outros. Ele também pode possuir um processador com arquitetura ARM®, MIPS®, PowerPC™ ou qualquer arquitetura voltada para sistemas embarcados como as citadas. Na figura 30 é mostrado um típico SoC ARM.

Figura 30
Configuração típica de um SoC ARM



Fonte: Maciel (2014).

Dos SBC que se destacam atualmente no mercado tem-se: Raspberry Pi, A20-OlinuXino-micro, Arduino TRE, Banana Pi, Orange Pi, Cubieboard, BeagleBone Black, BeagleBone Green, C.H.I.P., Odroid-XU4, PineA64, HummingBoard, entre outros.

Convém destacar que diferença básica entre microcontroladores e os SBC é a capacidade de processamento de dados. Eles são constituídos basicamente pelos mesmos componentes de um computador, mas um SBC possui suporte para vídeo e pode executar sistemas operacionais. Enquanto um microcontrolador terá sua aplicação mais voltada aos sistemas embarcados com função específica, um SBC poderá ser usado como sistema embarcado de propósito geral.

2.4.5. Sistema Operacional embarcado

Conforme Noergaard (2005, p. 383), o sistema operacional (SO) é uma parte opcional de um sistema embarcado. Ele pode ser usado se o processador do sistema puder suportá-lo. Um SO é uma coletânea de bibliotecas e aplicações que visam facilitar o uso e gerenciamento do sistema embarcado.

Necessariamente todo SO possui um *kernel* (núcleo). Por sua vez, o *kernel* é responsável pela comunicação entre *software* e *hardware*. Ele também permite que os aplicativos sejam usados e gerencia os recursos do computador (memória e processamento, por exemplo).

Portanto um SO embarcado é otimizado para funcionar com uma determinada arquitetura de um determinado processador. Por esse motivo alguns fabricantes mantém versões de SOs para dar suporte ao seu hardware específico.

2.4.5.1. Linux embarcado

Linux embarcado é uma expressão usada para sistemas embarcados que utilizam uma versão otimizada do *kernel* Linux. Essa expressão pode ser substituída por Linux em um sistema embarcado (MOLLOY, 2016, p. 56).

O uso do Linux tem as seguintes vantagens, segundo Molloy (2016, p. 57):

- É eficiente e escalável, pois pode ser executado desde dispositivos de 8 bits e/ou com programação orientada a objetos até em servidores web, ou seja, ele funcionará em uma grande diversidade de arquiteturas e processadores;
- Possui excelente suporte de código aberto para o seus periféricos;
- É livre, ou seja, não necessita de permissão para usá-lo;
- Seu *kernel* é utilizado massivamente em todo o mundo, dessa forma erros são detectados tão rapidamente quanto são resolvidos.

BeagleBone Black e Raspberry Pi, por exemplo, mantém *kernels* Linux, otimizados para suas plataformas, disponíveis na internet. Assim qualquer um que queira desenvolver um SO ou aplicação já encontrará o *kernel* pronto.

Convém ressaltar que um sistema operacional que utiliza o kernel Linux é comumente chamado de GNU/Linux. A denominação 'GNU' vem do projeto homônimo que desenvolveu uma extensa série de ferramentas para ser usado em conjunto com o *kernel*. Além das ferramentas, esse projeto também é responsável pela

filosofia do SL e deu início a comunidade que da suporte ao sistema.

2.5. TECNOLOGIAS DE CÓDIGO ABERTO

Compreende tanto *softwares* como *hardwares* de código aberto. Vem gerando repercussão no meio acadêmico, empresarial e industrial.

2.5.1 Software Livre

"O movimento software livre é um movimento político para liberdade dos usuários de programas. Um programa livre pertence ao conhecimento humano, mas um privado não" diz Richard Stallman (Improprietários, 2008) fundador da *Free Software Foundation* (FSF) e do movimento no início da década de 1980. Frequentemente confundido com software gratuito, o SL se destaca por se preocupar com a comunidade e não com o mercado. Conforme Silveira (2004, p.13), a FSF se refere ao SL como a liberdade dos usuários executarem, copiarem, distribuírem, estudarem, modificarem e aperfeiçoarem o programa. Isso é definido nas quatro liberdades para os usuários:

- Nº 0: A liberdade de executar o programa para qualquer propósito;
- Nº 1: A liberdade de estudar como o programa funciona e adaptá-lo para suas necessidades. O acesso ao código-fonte é um pré-requisito para essa liberdade;

- Nº 2: A liberdade de redistribuir cópias de modo que você possa ajudar ao seu próximo;
- Nº 3: A liberdade de aperfeiçoar o programa e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie.

Software livre é de código aberto, mas nem todo código aberto é software livre, pois ele nem sempre assegura as quatro liberdades. Um fornecedor pode disponibilizar o código fonte do seu software, mas pode restringir a sua distribuição, por exemplo.

É importante distinguir algumas categorias de software como código aberto (CA), gratuito e livre. Há vários softwares proprietários gratuitos como o Adobe Reader®, mas ele não é CA nem livre. Uma característica marcante do Software livre e de Código aberto (do inglês FOSS – *Free and Open Source Software*) é a sua comunidade. Através dela os softwares são testados, avaliados e reparados.

Criada para incentivar a aproximação de entidades comerciais com o software livre, a Iniciativa *Open Source* (figura 31) é uma organizaçao dedica a promoção do software livre. Sua atuação principal é a de certificar quais licenças se enquadram como licenças de software livre e promover a divulgação do software livre e suas vantagens tecnológicas e econômicas. Em sua visão, o software livre no longo prazo é economicamente mais eficiente e de melhor qualidade e, por isso, deve ser incentivado.

Figura 31
Logo da Iniciativa

Fonte: The Open Source Initiative (2017).

No meio acadêmico SL não representa apenas redução de custos, mas também eficiência, confiabilidade e flexibilidade (SILVA e CUNHA, 2006). No curso de engenharia elétrica da UERJ o SL Scilab substituiu o proprietário Matlab® e constatou-se que ele pode ser usado no ensino e no projeto de sistemas eletrônicos e de controle, segundo Silva e Cunha(2006), professores da área.

Também foi realizada uma pesquisa sobre softwares CAD livre para aplicação em engenharia civil (BIZELLO e RUSCHEL, 2011). Os pesquisadores compararam vários programas com o líder de mercado AutoCAD®. Eles avaliaram três características básicas: recursos 2D, recursos 3D e extensibilidade (capacidade de invocar comandos do

programa dentro do código). De forma conjunta nenhum dos SL avaliados (QCAD, PythonCAD e BRL-CAD) corresponderam ao esperado. Todavia, individualmente, cada um se destacou em uma característica.

O *software* livre tem sido muito empregado em programas de inclusão digital no país. Seja nas estações digitais para o uso comunitário de computadores e acesso à internet ou no ensino de robótica. Este último vem sendo realizado em escolas públicas e frequentemente em conjunto com *hardware* livre.

O uso de software livre na indústria brasileira vem crescendo nos últimos anos. Em 2006, a SOFTEX, a UNICAMP em parceria com o Ministério da Ciência e Tecnologia (MCT) realizaram uma grande pesquisa sobre o impacto do software livre e de código aberto na indústria de software brasileira. Dentre as empresas entrevistadas encontram-se: Varig, Petrobrás, Carrefour, Grupo Pão de Açúcar, Wall-Mart, UOL, Telemar, Itaú, entre outras.

Em uma entrevista com 15 empresas identificou-se os motivos para adoção e uso do SL em seus ambientes de trabalho. Os dados são apresentados na tabela 1. Esta pesquisa utilizou a escala de Likert de 1 a 5, crescente em importância.

Tabela 1
Motivos para desenvolvimento e uso de FOSS

Motivos	Média	Desvio padrão
Redução de custos (hardware e software)	4,36	0,84
Maior flexibilidade/liberdade para adaptação	3,71	1,44
Maior qualidade (estabilidade, confiabilidade, disponibilidade)	3,64	1,34
Maior autonomia de fornecedor	3,64	1,69
Maior segurança/privacidade/transparência	3,57	1,34
Maior escalabilidade	3,50	1,29
Maior aderência a padrões/interoperabilidade	3,43	1,65
Filosofia/princípios	3,29	1,73
Inclusão digital/social	2,64	1,95
Maior legalidade (licenças)	2,57	2,28
Disponibilidade de recursos humanos qualificados	2,14	1,03
Menor tempo para o desenvolvimento	2,29	1,45

Fonte: SOFTEX, UNICAMP e MCT (2005).

Já entre os setores econômicos que usam FOSS, as Tecnologias de Informação e Comunicação (TIC) ficam em primeiro lugar. Esse setor compreende gerenciamento de

informações, publicidade, educação a distância, rede de computadores, aplicações para internet e pode envolver inclusive processos de automação dentre muitas outras áreas. Esse setor cresceu com a popularização da internet. Os setores são mostrados na tabela 2.

Tabela 2
Intensidade de uso de FOSS nos setores econômicos

Setor	Somatório das notas dadas pelos grupos
Comunicações e Informação	46
Governo	44
Comércio	43
Educação	41
Setor Financeiro	27
Serviços	20
Equipamento Eletro-eletrônico e de Comunicação	20
Saúde	15
Cultura e Entretenimento	11
Energia	10
Transporte, Logística e Armazenamento	3
Agricultura, Extração Vegetal, Silvicultura, Caça	2
Indústria Bélica	1

Fonte: SOFTEX, UNICAMP e MCT (2005).

Vê-se que a intensidade do uso de FOSS concentra-se em mercados de TIC, governo, comércio, educação e outros serviços. Destacando TIC, governo e serviços podemos caracterizá-los da seguinte forma, conforme SOFTEX, UNICAMP e MCT (2005):

- TIC: alvo sistemático de desenvolvimento tanto de sistemas operacionais, infraestrutura, middleware como de aplicativos;
- Governo: gera forte demanda provocada por razões filosóficas e por redução de custos. Faz uso de sistemas operacionais, aplicativos, *middleware*, entre outros;
- Serviços: forte presença de sistemas operacionais e secundariamente outros itens de infraestrutura. Tem pouca presença de aplicativos.

Alguns softwares livres muito populares são: Android (sistema operacional de smartphones), navegador Firefox, Blender (software de modelagem), Ubuntu, Debian, entre outros.

2.5.2 Hardware Livre

Com base nos mesmo princípios do SL, o *hardware* livre (HL), ou aberto, propõe que o projeto do produto físico seja distribuído sob uma licença aberta. Esse movimento começou no final da década de 1990 por engenheiros que buscavam aplicar os conceitos do SL ao *hardware* eletrônico de computador (OSIER-MIXON, 2010). Factualmente, a definição de HL ainda está sendo definida formalmente.

Assim como no software, exites a Associação do *Hardware* de Código Aberto. Essa organização tem por objetivo assegurar que esse tipo de tecnologia e conhecimento seja acessível a todos, encorajando o desenvolvimento colaborativo nos âmbitos educacional, ambiental, sustentável e bem estar humano. Para simbolizar o *hardware* CA foi criado a logo mostrada na figura 32.

Figura 32
Logo do open Source Hardware

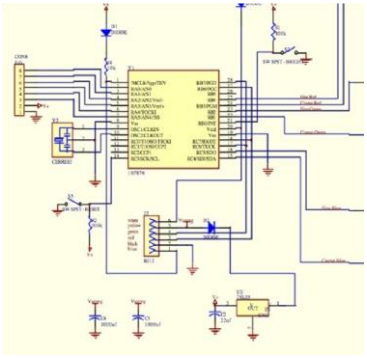
Fonte: Open Source Hardware Association. (2016).

As informações compartilhadas entre usuários e até empresas do segmentos são projetos descritos em desenho (CAD), diagramas de circuitos eletrônicos, esquemas elétricos, lista de componentes, *layout* de placas de circuito impresso, códigos de programação do *hardware* (*firmware*), seu *software* de controle, entre muitas outras formas. Alguns exemplos são destacado nas figuras 33 e 34.

Figura 33
Layout de uma PCB. À esquerda o projeto e à direita o produto finalizado

Fonte: Make (2007).

Figura 34
À esquerda o diagrama de um circuito eletrônico e à direita um firmware



```
6'b100111: begin // DAA

   // decimal adjust accumulator, or remove by carry any
   // results in nybbles greater than 9

   if (regfil[`reg_a][3:0] > 9 || auxcar)
      { auxcar, regfil[`reg_a] } <= regfil[`reg_a]+8'h06;
   state <= `cpus_daa; // finish DAA
   pc <= pc+16'h1; // Next instruction byte

end

6'b000100, 6'b001100, 6'b010100, 6'b011100, 6'b100100,
6'b101100, 6'b110100, 6'b111100, 6'b000101, 6'b001101,
6'b010101, 6'b011101, 6'b100101, 6'b101101, 6'b110101,
6'b111101: begin // INR/DCR

   regd <= opcode[5:3]; // get source/destination reg
   aluopra <= regfil[opcode[5:3]]; // load as alu a
   aluoprb <= 1; // load 1 as alu b
   if (opcode[0]) alusel <= `aluop_sub; // set subtract
   else alusel <= `aluop_add; // set add
   if (opcode[5:3] == `reg_m) begin

      raddrhold <= regfil[`reg_h]<<8|regfil[`reg_l];
      statesel <= `mac_indm; // inc/dec m
      state <= `cpus_read; // read byte

   end else state <= `cpus_indcb; // go inr/dcr cycleback
   pc <= pc+16'h1; // Next instruction byte

end
```

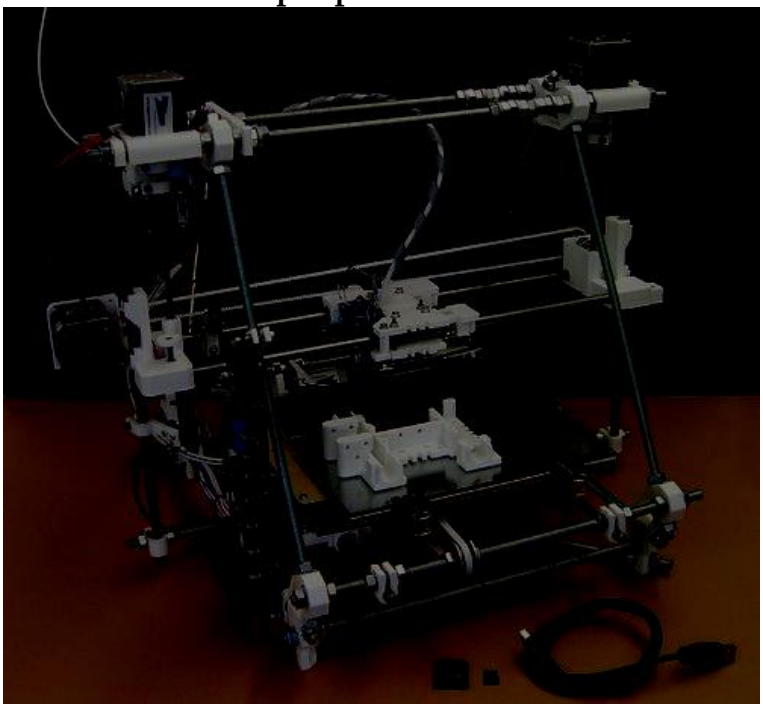
Fonte: Make (2007).

Existem projetos muito bem-sucedidos de HL. Eles são muito utilizados por hobbystas, estudantes e até empresas, tais como: Arduino, Bealge Bone, Oscar (projeto de carro), RepRap[3] e Raspberry Pi. Dentre eles o projeto Reprap talvez tenha sido a iniciativa que causou mais impacto na indústria.

A RepRap (figura 32) é a primeira impressora 3D de mesa e de baixo custo disponibilizada livremente. Iniciada em 2004, é uma máquina que manufatura objetos em plástico e pode criar as peças para construção de outra RepRap. Seu nome é abreviado de *Replicating Rapid-prototyper* (Prototipador de Replicação Rápida).

[3] O projeto Reprap está disponível em: http://reprap.org/

Figura 35
RepRapPro Mendel



Fonte: RepRap Wiki (2013).

A manufatura aditiva, popularmente conhecida como impressão 3D, é uma tecnologia disruptiva, uma vez que barateou todos os custos com esse tipo de tecnologia, se tornou muito acessível, criou novos mercados e descentralizou a produção. A facilidade de acesso a um equipamento do tipo possibilitou, em conjuntos com outros hardwares livre, a difusão da robótica, processos de manufatura, Internet da Coisas e o desenvolvimento de

tecnologias derivadas (“impressão 3D” de casas, por exemplo).

2.5.3. Tecnologias livres na indústria

Na indústria, o FOSS ainda está ganhando impulso. A implantação de novas tecnologias na indústria ocorre de maneira lenta, diferente do ambiente doméstico. As novas tecnologia têm de passar pelo aval de técnicos e engenheiros que atestem sua funcionalidade, caso contrário pode haver falhas (MECATRÔNICA ATUAL, 2011). No ambiente industrial insalubre e agressivo com altas temperaturas, vibrações e partículas em suspensão, por exemplo, falhas podem causar perdas irreparáveis e gerar grandes prejuízos.

No sul do país há casos em que Controladores Lógicos Programáveis (CLP), máquinas de Comando Numérico Computadorizado (CNC) e outros projetos são totalmente controlados a partir de sistemas com linux (MECATRÔNICA ATUAL, 2011). Além disso, as aplicações de linux embarcado são promissoras e estão crescendo dentro da indústria e diversas aplicações. Conforme a pesquisa desenvolvida pela UNICAMP, SOFTEX e MCT (p.6, 2005) são os modelos com maior oportunidade investimento.

O SL e HL apresentam uma série de modelos de negócios que podem ser implantados de acordo com a criatividade do empresário e a sua funcionalidade. Esses modelos podem ser aplicados tanto para serviços quanto para a criação de novos produtos.

Dentre as empresas que comercializam produtos com tecnologias livres destacou-se a estadunidense Adafruit Indutries, a italiana Arduino e a brasileira Metamáquina. A Adafruit, criada por uma aluna de engenharia eletrônica do MIT, desenvolve gadgets e aparelhos eletrônicos. É considerada a líder na indústria do HL (INOVAÇÃO TECNOLÓGICA, 2013). Arduino é o nome da empresa que desenvolve o produto homônimo citada anteriormente. A Metamáquina é primeira fabricante brasileira de impressoras 3D de baixo custo. Todas as máquinas são operadas por SL e baseadas em HL. Seu projeto é baseado em vários projetos disponíveis na internet, entre eles o RepRap.

Conforme a pesquisa mais recente da SOFTEX (2014, p.153), "É provável que Linux continue mantendo alguma participação no mercado de servidores/estações de trabalho, em virtude de este segmento ser menos contaminado por escolhas e preferências do usuário final sem perfil técnico."

Como a indústria é predominantemente guiada pelas preferências de professionais com perfil técnico, a não adoção de software livre se dá, segundo SOFTEX, UNICAMP e MCT (2005, p.45). Por:

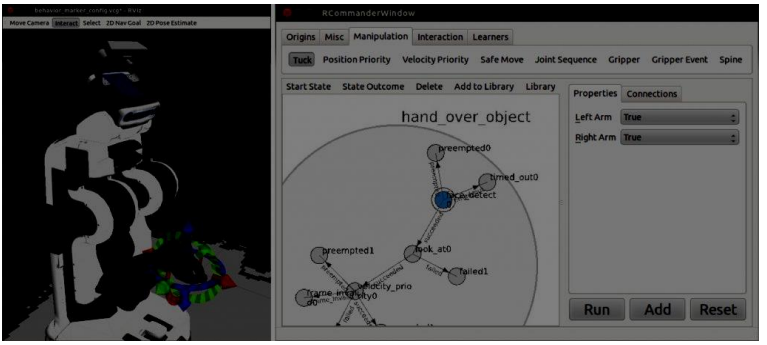
- Cultura estabelecida do software proprietário: as empresas ou clientes obrigam o uso de software proprietário e muitas vezes não há compatibilidade com os arquivos gerados em softwares comerciais;
- Problemas para instalação, como a falta de padronização dos sistemas a fim de facilitar a instalação e configuração de dispositivos e ferramentas, para utilização, pois as

interfaces com o usuário não são intuitivas e são mal elaboradas;
- Dificuldades em obter os drivers compatíveis com os fabricantes de hardware;
- Falta de versões em FOSS de ferramentas especializadas, como Autocad, Photoshop; para muitos, por enquanto, alguns softwares proprietários ainda são superiores;
- Falta de documentação, arquivos de ajuda e poucas bibliografias em português;
- Falta de suporte técnico ao usuário final (*help desk*).

Iniciativas como a *Open Source Automation Development Lab* (OSADL) também estão empenhadas a trazer as tecnologias livres para o ambiente industrial. OSADL é uma cooperativa que dá suporte para aqueles que querem implantar soluções livres na indústria. Eles são responsáveis por desenvolver, certificar e documentar os softwares usados, financiados pelos membros da cooperativa. É uma alternativa aos gastos com pesquisa e desenvolvimento de soluções para aprimoramentos nas plantas industriais (OSADL, 2017).

O SL *Robot Operating Sytem* (ROS) é uma plataforma usada para desenvolver programas para robôs. Ele possui uma coleção de ferramentas e bibliotecas que permitem a criação de sistemas robóticos complexos com uma grande variedade de comportamentos e funções. É um software fortemente usado na pesquisa e desenvolvimento que foi levado para ser usado na indústria, assim resultando em outra plataforma: ROS-Industrial. Na figura 36 é mostrado o ambiente de desenvolvimento do ROS na criação de comportamentos para o robô doméstico ROSCo.

Figura 36
Criação de comportamento para o robô doméstico ROSCo



Fonte: Geogia Tech (2016).

O ROS-Industrial é uma plataforma de código aberto criada para desenvolver projetos de manufatura automatizada e robótica para indústria. Ele possui uma coleção de plugins, bibliotecas e ferramentas voltados para a criação de interfaces para manipuladores robóticos, calibração de sensores, interfaceamento com redes industriais, simulação de processos (CAM), entre muitas outras funções. Ele é suportado por um consórcio internacinal de empresas influentes em seus setores, dedicadas a criar uma base de dados avançados e confiáveis para suas aplicações. As empresas do consórcio pode ser vista na figura 37.

Figura 37
Empresas americanas que fazem parte do ROS Industrial consortium

Fonte: ROS-Industrial (2015).

De acordo com Edwards (2013), dos motivos que o ROS-Industrial se tornou popular e influente, cita-se:

- Ele tem apoio de uma forte comunidade engajada em sempre aprimorá-lo;
- O seu repositório é bem organizado e atualizado;
- Enquanto outras iniciativas de código aberto são focadas na tecnologia, ele é focado no mercado;
- É bem documentado;
- É promovido por um consórcio de empresas influentes.

CAPÍTULO 3

METODOLOGIA

Neste capítulo serão apresentados os materiais, ferramentas e métodos necessários para desenvolvimento do projeto.

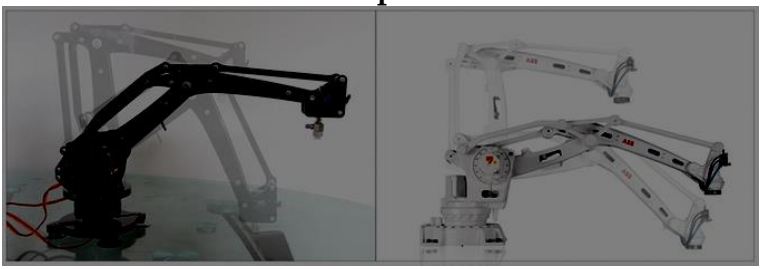
3.1. MATERIAIS

3.1.1. MeArm

Concebido inicialmente como uArm, pela UFactory[4], estreou na plataforma de finaciamento coletivo Kickstarter em 2014. Ele foi desenvolvido como uma miniatura de robô industrial, controlado por arduino, com quatro eixos e mecanismo paralelo baseado em um robô da fabricante ABB usado para empilhamento de paletes, como pode ser visto na figura 38. O mecanismo paralelo serve para que o efetuador fique sempre paralelo ao plano de trabalho.

[4]http://ufactory.cc

Figura 38
uArm e robô industrial paletizador ABB IRB460



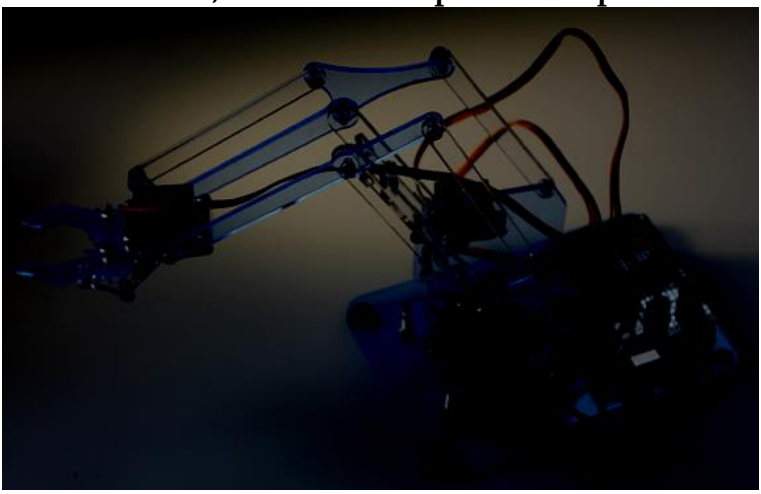
Fonte: Kickstarter (2012).

A ideia da Ufactory de construir miniaturas de robôs industriais adveio da necessidade de baretear os custos de robôs, uma vez que os modelos disponíveis comercialmente não eram acessíveis ao público em geral, e tornar a tecnologia mais popular. A uFactory também propôs tornar todo o projeto de código aberto, caso este fosse financiado. O projeto foi um sucesso, chegando a arrecadar $251.887,00.

Pouco tempo depois, o grupo Phenoptix, atualmente conhecido como Mime Industries, desenvolveu um manipulador robótico baseado no uArm de mais baixo custo ainda. Para isso eles eliminaram os mancais de rolamento usados nas juntas e um grau de liberdade no elo de saída para o efetuador. Os rolamentos foram substituídos por parafusos e no elo de saída o efetuador (garra) foi fixado. A Phenoptix nomeu esse manipulador de MeArm[5] (figura 39).

[5]O projeto MeArm pode ser baixado em: http://www.thingiverse.com/thing:298820

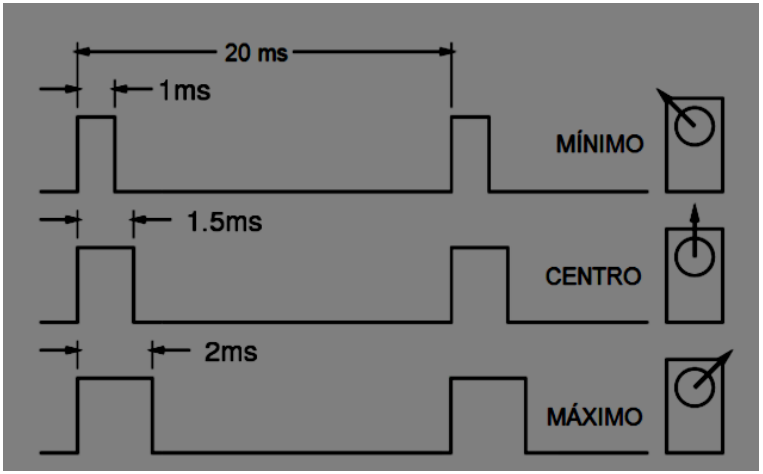
Figura 39
MeArm, desenvolvido pela Phenoptix.



Fonte: Phenoptix (2014).

Construído de partes em acrílico cortadas a *laser*, o MeArm utiliza micro servos para mover suas juntas e a garra. Os servos são alimentados com até 6V e o seu posicinamento é feito por modulação por lagura de pulso. Se o sinal PWM modulado for de 2ms o servo se posiciona em 180º. Já se o sinal modulado for de 1ms, o servo irá para 0º. Os outros ângulos são proporcionais a largura do pulso enviado, como pode ser visto na figura 40.

Figura 40
Sinais de controle de um servo motor

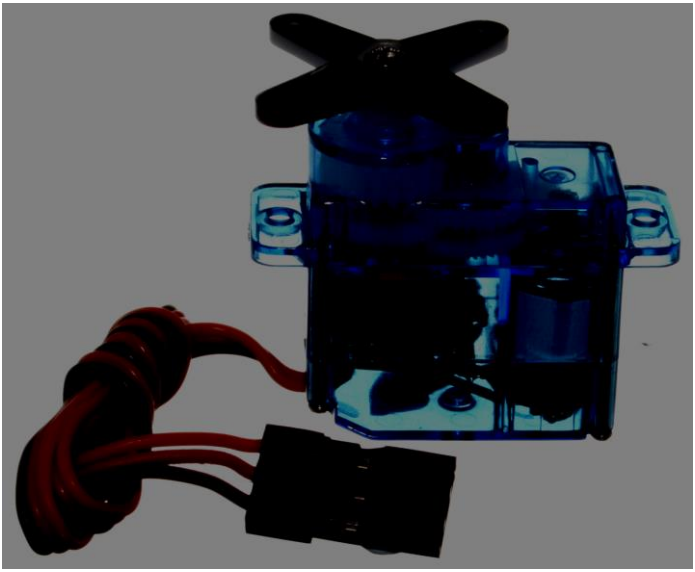


Fonte: EPUSP (2014).

Internamente o micro servo (figura 41) possui um motor CC acoplado em um conjunto de engrenagens e recebe sinal de uma placa que interpreta o pulso PWM enviado pelo controlador.

Figura 41

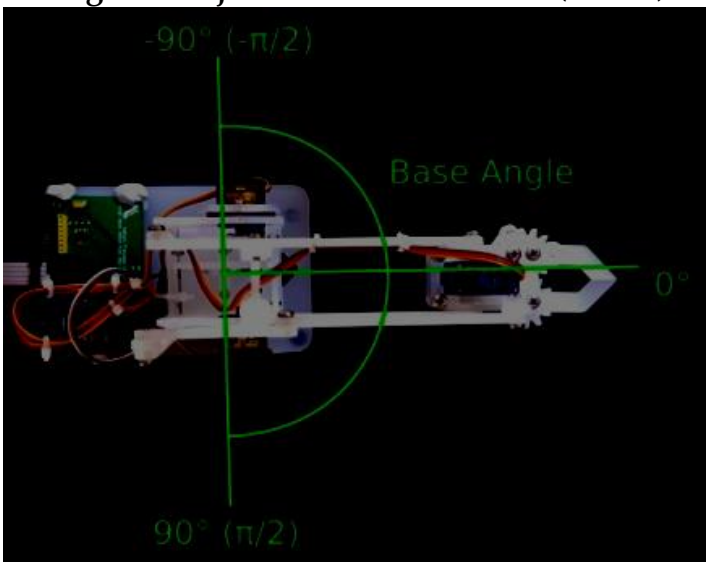
Micro servo Tower Pro SG90



Fonte: Oomlout (2009).

O MeArm possui uma junta torcional (T), responsável por girar a base no eixo Z e com ela todo o robô. Limitada pelo micro servo usado, esta varia de 0º a 180 º, como é mostrado na figura 42.

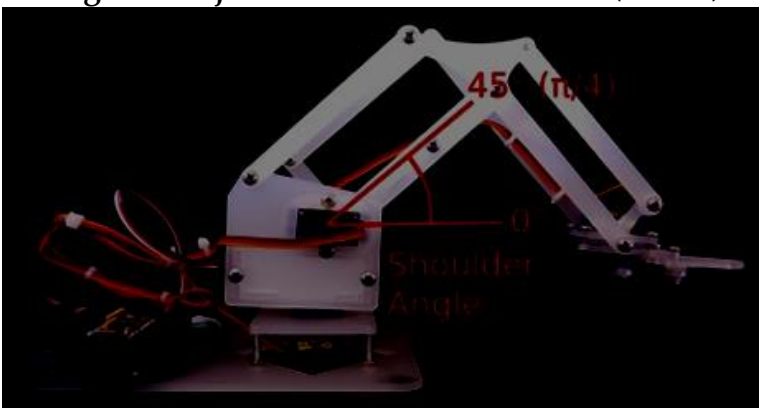
Figura 42
Ângulos da junta torcional da base (eixo Z)



Fonte: Bit of a hack (2014).

A junta responsável pelo deslocamento horizontal (no eixo X) é do tipo revolvente (V). Denominada de ombro, ela está localizada no lado direiro do robô. Podendo se deslocar no máximo 90º, seus limite variam de 45º até 135º, tomando o plano de apoio como referência, conforme a figura 43.

Figura 43
Ângulos da junta revolvente do ombro (eixo X)

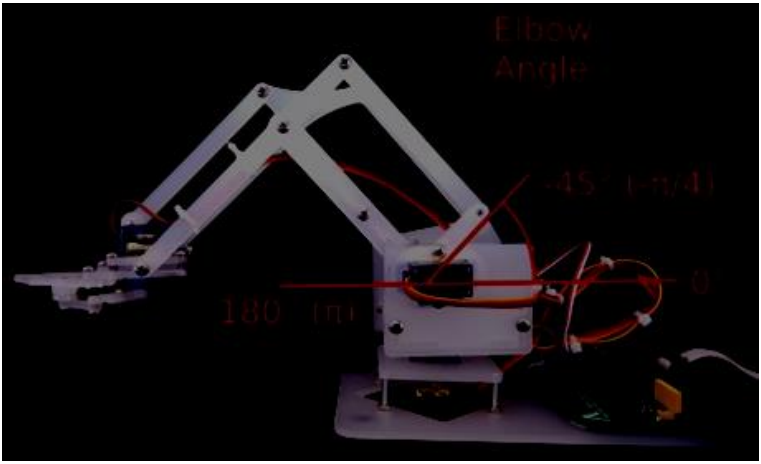


Fonte: Bit of a hack (2014).

Por fim, vê-se na figura 44 o "cotovelo" do robô, junta responsável pelo deslocamento vertical (eixo Y). Assim como o ombro, mas do lado esquerdo, esta é uma junta revolvente e se desloca no máximo 90º, com seus limites variando de 45º até 135º.

Figura 44

Ângulos da junta revolvente do cotovelo (eixo Y)



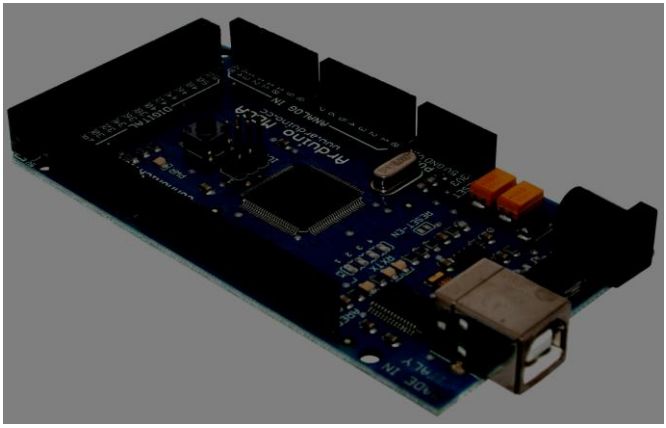
Fonte: Bit of a hack (2014).

Assim, essas três juntas conferem ao MeArm três graus de liberdade. Devido as juntas do ombro e do cotovelo estarem montadas de forma paralela, o efetuador sempre estará paralelo ao plano de trabalho. Esse tipo de configuração é indicado para trabalho em que o robô terá o seu efetuador numa posição fixa, apenas se deslocando em três dimensões. Pode ser usado para empilhar paletes, trabalhar sobre superfícies perpendiculares (pintura de um muro) ou paralela (usinagem).

3.1.2. Arduino MEGA 2560

O arduino (figura 45) é uma plataforma de código aberto para prototipagem eletrônica. O projeto do hardware, firmware, IDE, esquemático e PCB podem ser adquiridos em seu website[6]. Por esse motivo existem várias plataformas semelhantes, derivadas e réplicas.

Figura 45
Arduino MEGA 2560



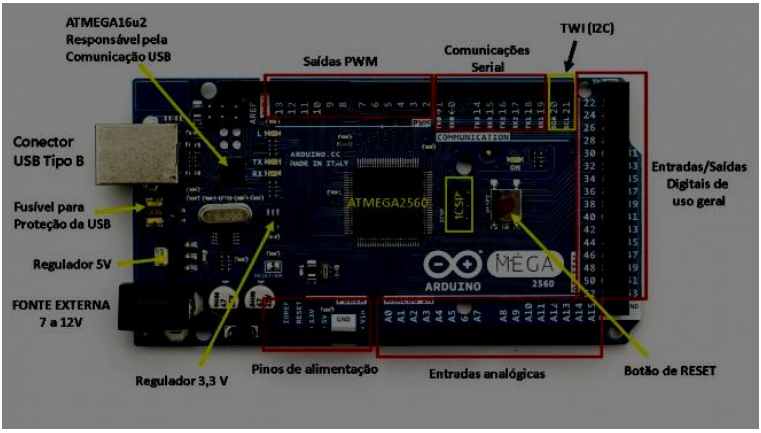
Fonte: Oomlout (2009).

Corriqueiramente o arduino é confundido com um microcontrolador, mas ele é uma plataforma de desenvolvimento. O microcontrolador é de fato o chip que processa as entradas e fornece saídas, neste caso o Atmega2560, fabricado pela ATMEL. O termo plataforma

[6] O projeto pode ser baixado em: https://www.arduino.cc/en/Main/ArduinoBoardMega2560

é devido ao fato do chip possuir todos os seus pinos já direcionados a portas em uma placa de circuito impresso, ou seja, ele já possui USB, fonte de alimentação, cristal oscilador, encaixe para pino nas portas e outros recursos, como mostrado na figura 46.

Figura 46
Recursos do Arduino MEGA 2560



Fonte: Souza (2014).

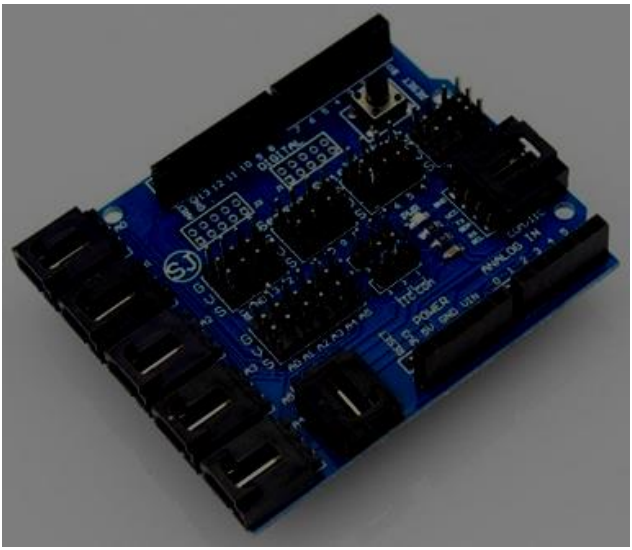
O arduino pode ser alimentado tanto pela porta USB quanto por uma fonte externa. Em caso de erro no projeto ou falha do circuito, há um fusível resetável para proteger a porta USB do computador em que for conectado. Seus pinos operam com tensão de 5V e podem fornecer ou drenar até 40 mA (ATMEL, 2014).

A saída PWM possui 8 bits de resolução, variando de 0 a 255. Já a conversão de analógico para digital é feita com uma resolução de 10 bits, isto é, pode variar de 0 a 1023

Para desenvolvimento deste projeto as saídas PWM, as entradas analógicas e o conector USB foram essenciais. Porém, devido a quantidade de cabos usados, foi usado em conjunto com o arduino um *Sensor Shield*, como mostrado na figura 47. Esta foi a solução usada para mater os conectores e seus cabos organizados. S*hields* são placas que são conectadas ao arduino para expandir suas capacidades.

Figura 47

Sensor Shield V4.0 para arduino



Fonte: Filipe Flop (2016).

A IDE usada para programar o arduino foi a versão 1.8.1. Ela possui uma ferramenta que permite monitorar a comunicação serial entre o arduino e o computador, muito útil para desenvolvimento deste projeto.

3.1.3. Scilab

Scilab (*Scientific Laboratory*) é um software livre e de código aberto de cálculo numérico, que prover um ambiente computacional para aplicações científicas e de engenharia (SCILAB, 2015). Ele está disponível[7] para sistemas baseados em Linux, Mac OS X e Windows.

É desenvolvido desde 1990 por pesquisadores do "*Institut Nationale de Recherche en Informatique et en Automatique* (INRIA)" e pela "*Ecole Nationale des Ponts et Chaussée*" (ENPC) na França. Atualmente é mantido pelo Scilab Enterprises.

O Scilab tem uma sofisticada estrutura de dados que inclui objetos como funções racionais, polinômios e sistemas lineares, por exemplo. Possui um interpretador e uma linguagem de programação estruturada própria, além de poder trabalhar com as linguagens C e Fortran. Dentre as suas muitas biblotecas e funções, o Scilab também trabalha com:

- Operações matemáticas e análise de dados;
- Visualização de dados em 2D e 3D;
- Estatística;
- Projeto e análise de sistemas de controle;
- Processamento de sinais contínuos e discretos;
- Otimização de sistemas.

[7]Scilab pode ser baixado em: http://www.scilab.org/download/latest

Possui ferramentas para troca de dados com dispositivos externos. Ainda possui um modelador e simulador de sistemas dinâmicos híbridos, o Xcos, comparado constantemente com o Simulink do Matlab. Sua logo é mostrada na figura 48.

Figura 48
Logo do Scilab

Fonte: Scilab (2015).

Por ser usado internacionalmente em ambientes acadêmicos e industriais, o Scilab é uma plataforma em constante atualização e aperfeiçoamento. É usado por empresas como o Grupo Airbus, a Arcelor Mittal e a Peugeot. Ainda possui vários estudos de caso documentados para acesso[8] ao público nas seguintes áreas: aeroespacial, automotiva, energia, metereologia, minieração, metalurgia, médica, farmacêutica, educacional e de pesquisa.

Em automação, o scilab é usado para modelagem e simulação (Scicos), controle de processos, controle

[8] Estudos de caso disponíveis em: http://scilab.io/use-cases-menu/

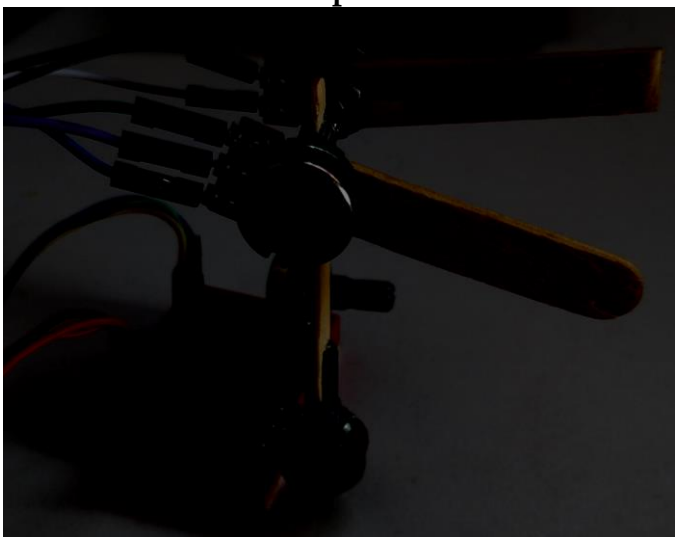
clássico, controle robusto, controle discreto (digital), controle inteligente, robótica, criação de supervisório, RTAI (*Real Time Application Interface*) etc. Além disso ele possui ferramentas para uso do protocolo de comunicação digital OPC (OLE for Process Control), protocolo modbus, serial, HART, entre outros.

Para este projeto foram usando as suas ferramentas para criação de uma interface gráfica de usuário (GUI) e transmissão de dados pela porta serial.

3.1.4. Manípulo mestre

A programação do MeArm é feita com o método mestre e escravo, usado para movimentar o manipulador, e o método ponto a ponto, usado para guiar o robô por pontos salvos. Para o primeiro método, foi construído um manípulo para servir de mestre para o manipulador robótico. Ele pode ser visto na figura 49.

Figura 49
Mestre do manipulador robótico



Fonte: Produzido pelo autor (2017).

O manípulo possui a mesma quantidade de graus de liberdade do robô. Em suas juntas foram usados potenciômetros: um de 100kΩ (base) e dois de 10kΩ (ombro e cotovelo). Inclusive os potenciômetro do ombro e do cotovelo foram dispostos paralelos, como no MeArm.

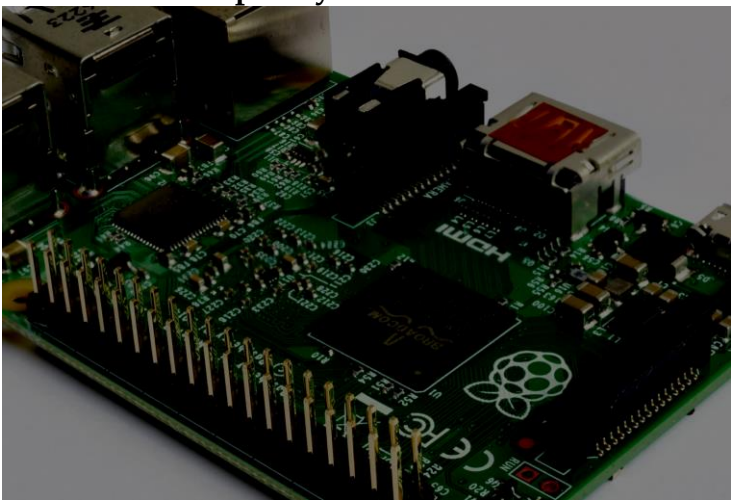
Os potenciômetros para controle da base e da garra foram montados em uma *protoboard*, que também serve de apoio do conjunto. Já os demais potenciômetros foram fixados no elo de entrada, feito com palitos em madeira.

A conexão dos potenciômetros ao arduino pode ser vista nos apêndices A e B.

3.1.5. Raspberry Pi 2 modelo B

O Raspberry Pi (RPi) é um SBC desenvolvido pela inglesa *Raspberry Pi Foundation E*le foi criado para servir como plataforma educacional com o objetivo de criar uma forma das pessoas entenderem mais sobre o mundo digital, aprenderem a resolver problemas desse tipo e estarem preparadas para os empregos do futuro. É um projeto de hardware livre[9], além de ser uma placa de baixo custo para que o maior número de pessoas tenha acesso (RASPBERRY PI, 2016). Na figura 50 é mostrado o modelo B de sua versão 2.

Figura 50
Raspberry Pi 2 modelo B.



Fonte: Multicherry (2015).

[9]Projeto Raspberry Pi disponível em: https://www.raspberrypi.org/documentation/

Esse modelo vem com um SoC Broadcom BCM 2836 que inclui:

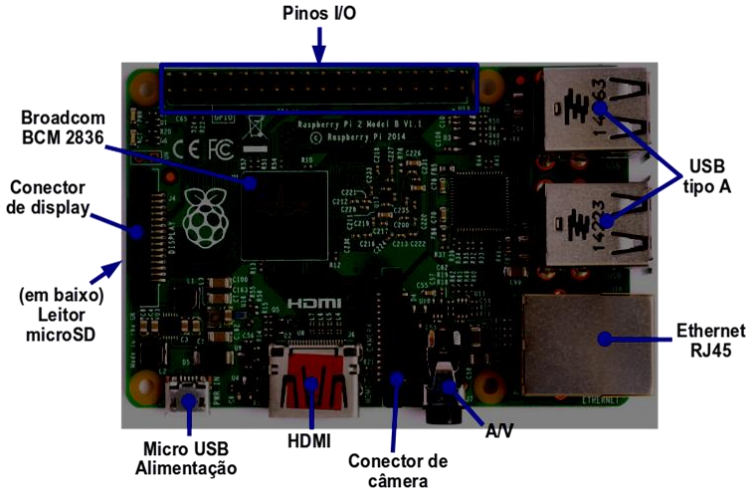
- Processador quad-core ARM Cortex-A7 com 900MHz, com ponto flutuante por hardware;
- 1 GB de memória RAM;
- VideoCore IV 3D graphics core.

Na placa também encontram-se:

- 4 portas USB tipo A;
- Porta HDMI;
- Porta Ethernet 10/100 (RJ45);
- Conector combinado 3.5mm de áudio e vídeo composto;
- Conector para câmera (CSI);
- Conector para Display (DSI);
- Compartilmento para cartão Micro SD;
- Micro USB para alimentação;
- 40 pinos de entrada e saída, os quais 26 são de propósito geral.

Na figura 51 é possível observar como esses componentes estão distribuídos na placa.

Figura 51
Indicação dos componentes do Raspberry Pi 2 modelo B.



Fonte: Registrado pelo autor (2017).

O RPi foi escolhido para esse projeto por ser livre, por suas configurações e pela necessidade de processamento gráfico para a IHM. Como ele pode ser configurado para ser um sistema embarcado de propósito específico ou geral, ainda será possível implementar novas funcionalidades futuramente.

Devido ao seu processador ARMv7, o Raspberry Pi pode executar uma gama de distribuições GNU/Linux otimizadas para processadores de arquitetura ARM. Para

este projeto foi usado o sistema operacional Ubuntu com interface MATE, otimizado para Raspberry Pi.

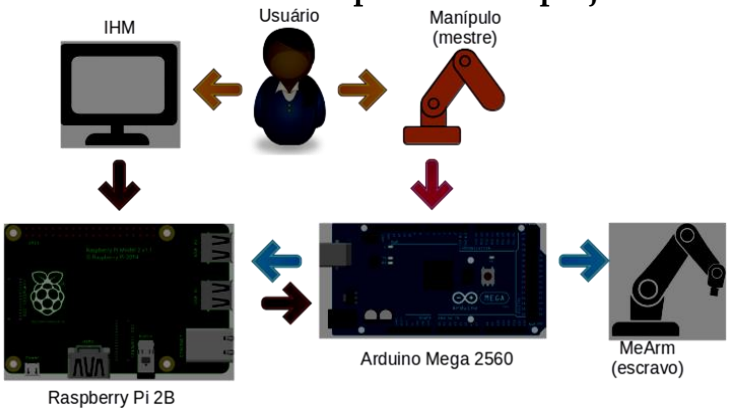
O Ubuntu MATE[10] foi escolhido por ser livre, flexível, compatível com os *softwares* de controle necessários (Scilab e Arduino), possuir boa documentação, receber atualizações frequentemente e oferecer um bom suporte por parte da comunidade.

3.2. MÉTODOS

3.2.1. Montagem dos componentes

Na figura 52 é mostrado como os componentes do projeto interagem entre si.

Figura 52
Conexão dos componentes do projeto



Fonte: Registrado pelo autor (2017).

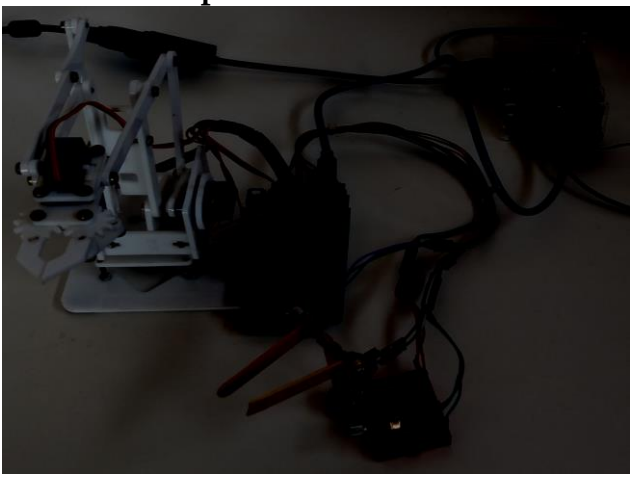
[10]Ubuntu MATE para Raspberry Pi disponível em: https://ubuntu-mate.org/raspberry-pi/

O usuário será responsável por controlar o manípulo e interagir com o IHM. A movimentação do manípulo é interpretado pelo Arduino, que recebe os sinais dos potenciômetros, e enviado para o MeArm, que imita os seus movimentos. Os comandos feitos pelo IHM são processados no Raspberry Pi e enviados ao arduino. Quando requisitado, o arduino também envia dados para o RPi.

O RPi se comunica com o arduino através da porta USB. O manípulo e o MeArm se conectam ao arduino diretamente através de fios. Como o consumo é baixo o arduino fornece a alimetação dos componentes. A tela do IHM se conecta ao RPi pela porta HDMI.

Na figura 53 são mostrados os componente físicos conectados entre si.

Figura 53
Componentes conectados



Fonte: Registrado pelo autor (2017).

O usuário interage com o IHM usando mouse e teclado e visualiza a interface de controle em um monitor LCD. Todo o conjunto foi montado na bacada de testes mostrada na figura 54.

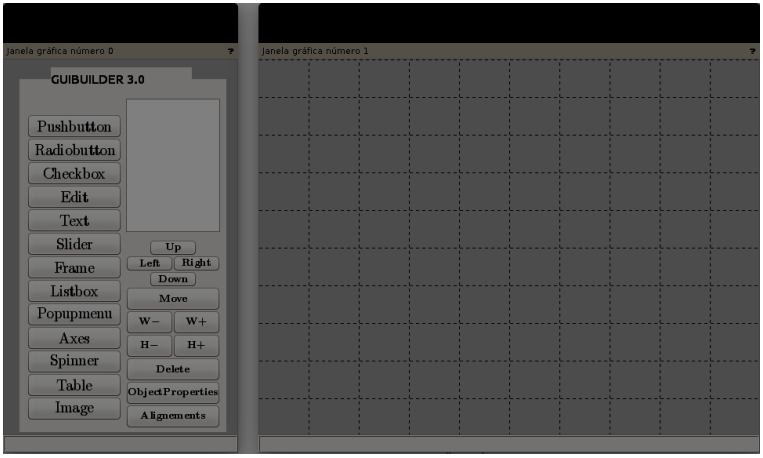
Figura 54
Bancada de testes do projeto.

Fonte: Registrado pelo autor (2017).

3.2.2. Criação da GUI no Scilab

A Interface Gráfica de Usuário (no inglês, GUI) foi desenvolvida no Scilab usando a *toolbox* GUI Builder 3.0. Na figura 55 pode-se observar a sua interface.

Figura 55
Interface da toolbox GUI Builder 3.0 do Scilab



Fonte: Registrado pelo autor (2017).

A *toolbox* dispõe dos elementos básicos para contrução de IHMs, tais como: botões, caixas de marcação, campo de edição, tabelas, imagens, caixa de lista, *sliders*, barras gráficas, botões giratório, entre outros.

A interface da suporte para que o usuário insira os elementos nos tamanho, forma e quantidade que deseja. Quando a GUI está pronta é gerado um *script* para compilá-la e executá-la pelo Scilab. No código gerado, o usuário deve inserir todas as funções dos elementos criados graficamente.

A GUI criada para este trabalho pode ser vista integralmente no apêndice C. Para compilá-la, basta copiar o código e criar um script no Scilab usando o SciNotes.

3.2.3. Comunicação serial no Scilab

Para comunicar o Scilab com o arduino pela porta serial foi usada a *toolbox Serial Communication* 0.4.1. Trata-se de uma ferramenta simples e objetiva. Para execução deste projeto utilizou-se apenas quatro comandos:

- closeserial: encerra a comunicação serial na porta definida;
- openserial: inicia a comunicação serial na porta definida. Configura-se também os bits de paridade e a velocidade de transmissão;
- readserial: lê a informação do buffer da porta definida;
- writeserial: insere uma informação no buffer da porta definida.

Afim de facilitar a conexão serial, foi inserido no código uma lógica de verificação que detecta o sistema operacional e em qual entrada serial o arduino foi conectado. O usuário deve apenas selecionar as opções que surgirem nas janelas.

Para os elementos inseridos na GUI foram atribuídas as seguintes fuções:

- **Salva posição (Botão):** quando pressionado, registra na memória do arduino a posição que o MeArm se encontra no momento e também as envia para o Scilab;
- **Executar (Botão):** quando pressionado, executa a sequência de posições salvas na memória indefinidamente;
- **Velocidade (slider):** a medida que varia, envia um valor de 3000 a 4980 pela serial;

- **Armazenar (Botão):** quando pressionado, armazena a sequência de posições da memória em um arquivo;
- **Executar arquivo (Botão):** quando pressionado, carrega as posições salvas em arquivo na memória do arduino e as executa;
- **Pausar (Botão):** quando pressionado, pausa a execução dos movimentos do robô;
- **Continuar (Botão):** quando pressionado, retoma a execução dos movimentos do robô;
- **Parar (Botão):** quando pressionado, interrompe a execução de movimentos do robô e o faz retornar para o modo de aprendizagem (volta a se mover de acordo com o manípulo);
- **Reiniciar (Botão):** quando pressionado, apaga as posições da memória;
- **Iniciar comunicação serial (Botão):** quando pressionado, inicia a comunicação serial com o arduino;
- **Encerrar comunicação serial (Botão):** quando pressionado, encerra a comunicação serial com o arduino;
- **Inserir número de repetições (campo de edição):** campo para que o usuário insira a quantidade de vezes que as posições da memória serão executadas;
- **Enviar número de repetições (Botão):** Envia o número de repetições e inicia a execução dos movimentos;
- **Fechar (Botão):** fecha a interface de controle.

O códido usado no Scilab para cada função pode ser visto no apêndice C. Todos os comandos enviados pelo Scilab correspondem a números que variam de ‘0002’ a ‘4980’. Esses números são interpretados pelo arduino, que executa a função desejada (figura 57). Convém ressaltar

que na comunicação serial são transmitidos apenas dados do tipo caractere. Portanto, tanto na chegada quanto na saída do Scilab e do Arduino os dados são convertidos para os tipos adequados.

O RPi dispões de um dispositivo UART integrado na placa, portanto a transmissão serial de dados foi assíncrona.

3.2.4. Algoritmo de controle do arduino

No Arduino está o código mais complexo, uma vez que o código do Scilab é focado na GUI e comandos pela porta serial. O código do arduino é responsável, resumidamente, por:

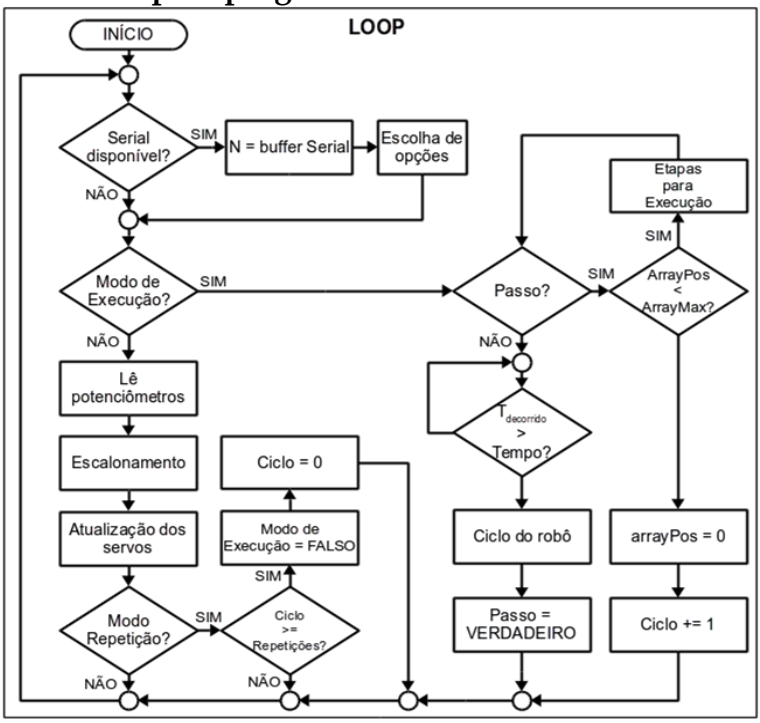
- Interpretar os sinais analógicos enviados pelo manípulo e movimentar as juntas do MeArm (função de aprendizagem);
- Salvar as posições de todas as juntas na memória;
- Executar a sequência de posições salvas de todas as juntas (*playback*);
- Calcular a variação de deslocamento, de forma que todas juntas se movam em sincronia entre as posições salvas;
- Variar a velocidade de execução em tempo real, ou seja, enquanto a execução do movimento está acontecendo;
- Receber os comandos pela serial e executá-los imediatamente.

Para desenvolver algoritmo de controle do robô, inicialmente foi estudado o código desenvolvivo por

Pinaut[11]. Deste código advieram as ideias de sincronização das juntas, registro das posições em arrays e funcionamento do modo de execução.

Para melhor compreensão do código na figura 56 está a função loop do código de forma simplificada. Alguns detalhes foram ignorados afim de facilitar o entendimento de como o algoritmo funciona. O código fonte completo pode ser visto no apêndice D.

Figura 56
Loop do programa inserido no arduino

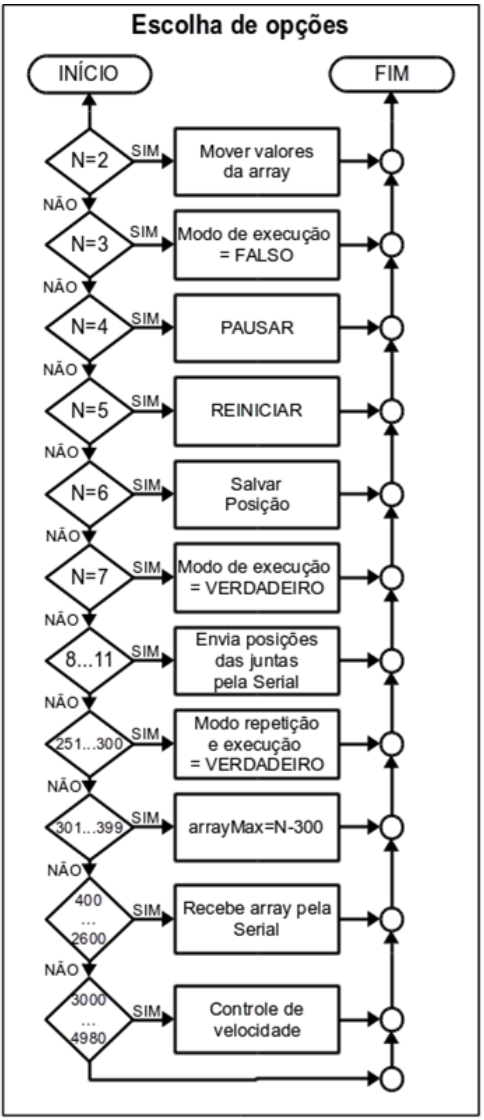


Fonte: Registrado pelo autor (2017).

[11]http://letsmakerobots.com/node/42704

Primeiramente é verificado se há dados na porta serial a serem lidos. Se sim, os dados atribuídos a uma variável que será comparada em uma lista de opções, como mostrada na figura 57.

Figura 57
Parte do código para escolha de opções de acordo com a variável recebida



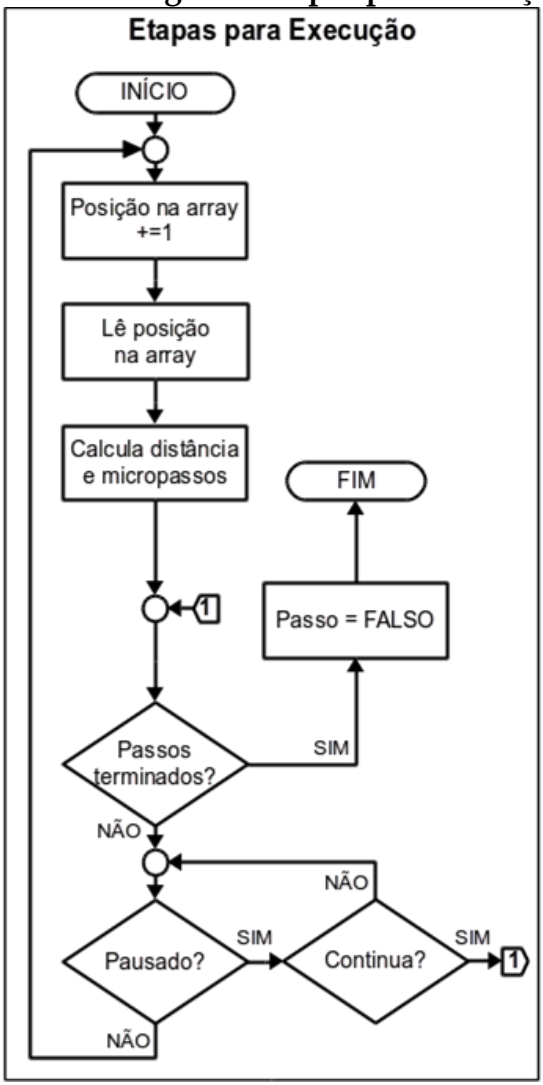
Fonte: Registrado pelo autor (2017).

Todos os valores que podem vir pela serial são previstos nas opções disponíveis. Independente da opção escolhida, o programa segue o fluxo e prossegue com o *loop*. Depois da escolha de opções verifica-se se o modo de execução é verdadeiro. Se não for, lê-se o valor dos potenciômetros, escalona-se esses valores para as juntas e atualiza-se a posição dos servos. Se o modo de repetição for FALSO, esse procedimento é realizado novamente.

Caso o modo de execução seja VERDADEIRO, e o modo de repetição for FALSO, as posições salvas na memória serão executadas indefinidamente. A execução dos movimentos pode ser resumida em três etapas essenciais: leitura da posição da array, cálculo da distância e verificação do tempo decorrido para atualização dos servos.

Sempre que a posição das juntas é salva, esses dados são armazenados em arrays (matrizes de uma linha e ‘n’ colunas, o qual ‘n’ é o número de posições salvas). O código aponta a posição da array que deve ser lida e, faz a leitura, como visto na figura 58.

Figura 58
Parte do código das etapas para execução



Fonte: Registrado pelo autor (2017).

Uma vez feita a leitura é feito o cálculo da distância entre a posição atual e a posição que será alcançada. A função deslocamento () é responsável pelo cálculo e pode ser vista no apêndice D. Essa função calcula a variação individual de cada junta, tomando como base a maior variação. Isso garante que todas as juntas alcancem a posição final simultaneamente.

Depois dessa etapa, antes que os servos atualizem suas posições com a variação calculada é feita a verificação do tempo decorrido em relação ao tempo estabelecido. Se o tempo decorrido até aquele momento for maior que o tempo estabelecido os servos são atualizados, caso contrário não são. Quanto menor for o tempo estabelecido, maior será a velocidade de execução; quanto maior for o tempo estabelecido, menor será a velocidade de execução. Dessa forma é feito o controle de velocidade do robô. O tempo estabelecido é definido pelo usuário através da IHM.

A verificação do tempo decorrido foi usado no lugar do comando *delay* (tempo de espera para avançar a próxima linha do código) para que os comandos vindos da serial sejam executados imediatamente. Enquanto o programa está em espera (*delay*), ele não executa nada.

Se o modo de repetições for VERDADEIRO, o modo de execução também será enquanto o número de ciclos executados for menor que o número estabelecido pelo usuário.

Convém ressaltar que os pulsos enviados aos servos são em microsegundos (us), devido a maior precisão. Anteriormente foi dito que se o sinal PWM modulado for de 2ms o servo se posiciona em 180º. Já se o sinal modulado

for de 1ms, o servo irá para 0º. Nesse caso 2ms equivalem a 2000us. Contudo esse padrão de 2000us equivaler a 180º varia para cada fabricante. O microservo usado na base, por exemplo vai para posição de 0º quando o pulso é de 600us, e 180º quando o pulso é 2400us.

3.2.5. Geração de arquivos .csv pelo Scilab

Para armazenar uma sequência de posições salvas do robô de forma que sejam acessadas posteriormente, cria-se um arquivo no formato '.csv'. Arquivos desse tipo armazenam dados tabelados e os valores são separados por um delimitador (vírgula ou quebra de linha). O csv tornou-se um padrão adotado inicialmente por mainframes e mais tarde por banco de dados.

Neste projeto, esse formato foi adotado pois a manipulação das posições salvas são feitas através de matrizes de quatro linhas por 'n' colunas, o qual 'n' é o número de posições salvas.

Sempre que o usuário requisita que uma posição seja salva, o arduino registra a posição de cada junta (base, ombro, cotovelo e garra; quatro no total) e envias essas informações também para o Scilab. Caso o usuário deseje salvar a sequência de posições em arquivo é usado o seguinte comando:

```
write_csv(%M, home + '/pos_robo.csv');
```

Esse comando salva a matriz '%M' no arquivo 'pos_robo.csv', que fica no diretório home.

Para ler os dados do arquivo e armazená-los em uma nova matriz (%G) são usados os seguintes comandos:

```
%G=[];
%G=read_csv(home + '/pos_robo.csv');
```

Para enviar as posições pela serial estabeleceu-se o seguinte protocolo:

- Verifica-se o número máximo de colunas (que equivalem ao número de posições salvas) e armazena-se esse valor na variável %arrayMax;
- Todos os valores enviados deve possuir quatro caracteres, portanto valores menores que 1000 tem um '0' adicionado no começo;
- %arrayMax é somada a 300 e enviada pela serial. O algoritmo do arduino reconhece todos os valores de 301 a 399 como valores de arrayMax de 1 a 99. Se for enviado 0301, o arduino reconhece arrayMax = 1, por exemplo;
- A matriz %G é lida e seus dados são enviados pela serial. Individualmente são enviadas todas as linhas de cada coluna, ou seja: posição da base, posição do ombro, posição do cotovelo e posição da garra.

Abaixo pode-se observar o recho do código que realiza essas funções. O código completo se encontra no apêndice C.

```
[n1,%arrayMax]=size(%G);
if %arrayMax<10 then
    am=string(%arrayMax);
```

```
    am='030'+am;
    disp('array enviada: ');
    disp(am);
    writeserial(%porta_serial,am);
else
    am=string(%arrayMax);
    am='03'+am;
    disp('array enviada: ');
    disp(am);
    writeserial(%porta_serial,am);
end
for  i=1:%arrayMax
   for j=1:4
     var=%G(j,i);
     num=strtod(var);

     if num<1000 then
        var='0'+var;
        writeserial(%porta_serial,var);
        disp(var);
        sleep(50);
     else
        writeserial(%porta_serial,var);
        disp(var);
        sleep(50);
     end
   end
end
```

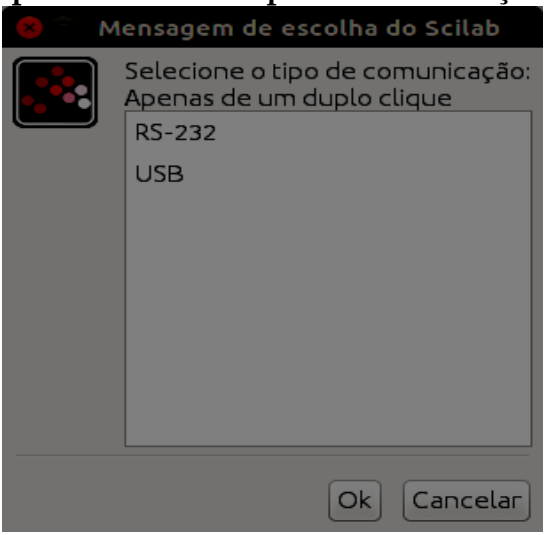
CAPÍTULO 4
RESULTADOS E ANÁLISES

Neste capítulo serão apresentados os resultados e análises decorrente dos testes realizados no sistema.

4.1. EXECUÇÃO DA INTERFACE DE CONTROLE

A complição do *script* no Scilab funcionou conforme o esperado. Quando compilado, a primeira janela que surge solicita que o usuário escolha o tipo de comunicação (figura 59). A opção RS-232 é mostrada apenas como um exemplo dos tipos de comunicação que podem ser implementados.

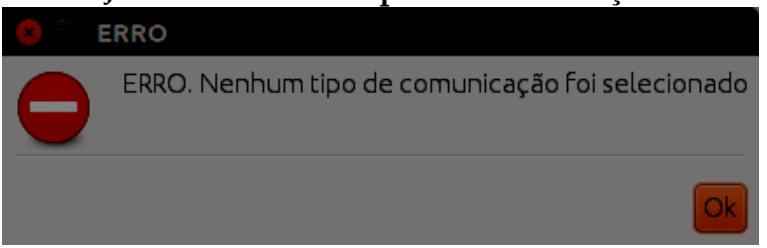
Figura 59
Janela para escolha do tipo de comunicação serial



Fonte: Registrado pelo autor (2017).

Caso nenhum tipo seja selecionado surgirá uma janela informando o erro, como mostrada na figura 60.

Figura 60
Janela de erro do tipo de comunicação

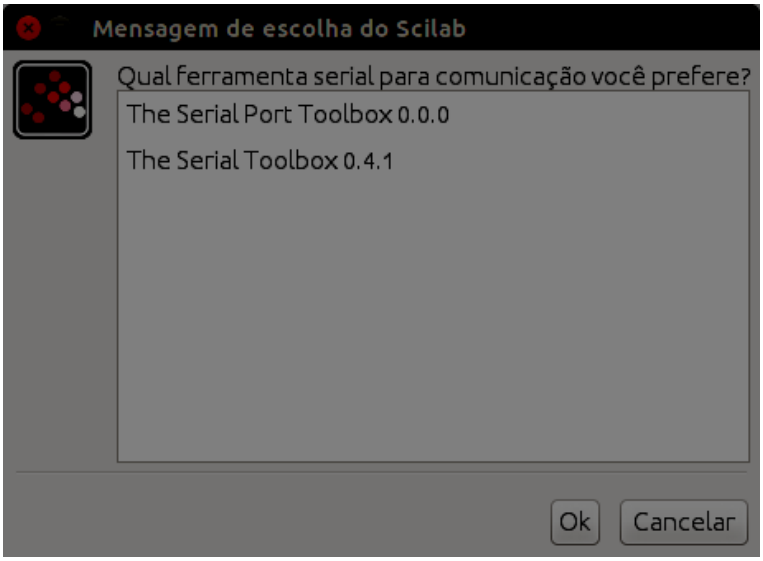


Fonte: Registrado pelo autor (2017).

Se não houver erros, a janela seguinte solicita que o usuário selecione a *toolbox* que prefere usar. Em sistemas GNU/Linux há duas *toolboxes* que podem ser usadas, mas Windows e MAC OS dispões apenas de uma. Para evitar problemas, caso a GUI seja executada em outros sistemas, foi criada a janela de opções mostrada na figura 61.

Figura 61

Janela para seleção de toolbox

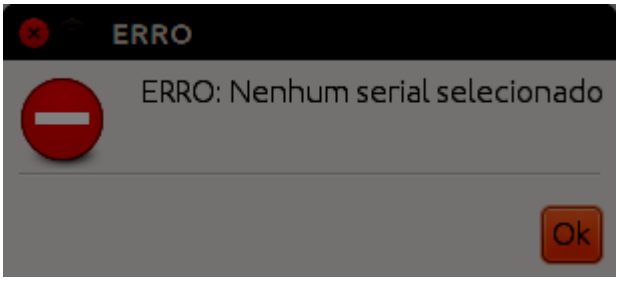


Fonte: Registrado pelo autor (2017).

Caso nenhuma *toolbox* seja selecionada surgirá uma janela informando do erro, como mostrada na figura 62.

Figura 62

Janela de erro da toolbox

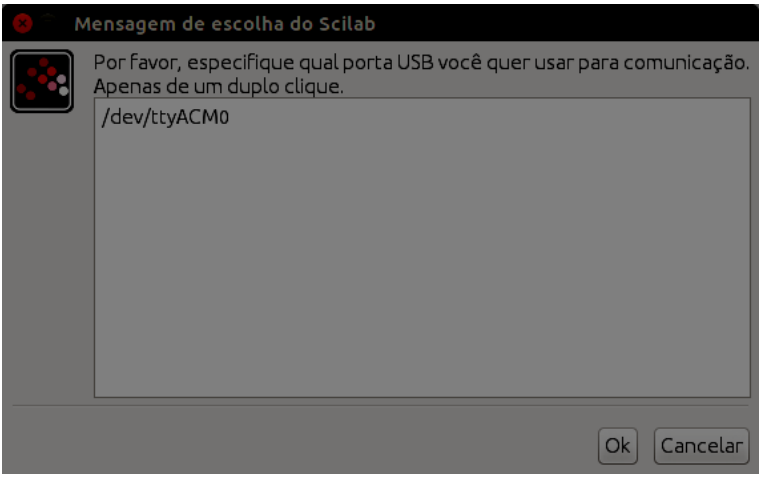


Fonte: Registrado pelo autor (2017).

Uma vez que o tipo de comunicação e a toolbox foram selecionados, o software verifica em que porta serial o arduino está conectado. No Linux serão usadas as portas ttyACM* enquanto no windows são as portas COM*. Na figura 63 é mostrada a janela de seleção da porta USB em que o arduino está conectado.

Figura 63

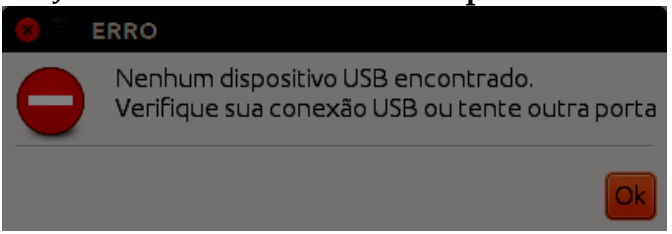
Janela de escolha da porta USB



Fonte: Registrado pelo autor (2017).

Caso ocorra um erro nessa etapa, a janela da figura 64 surgirá.

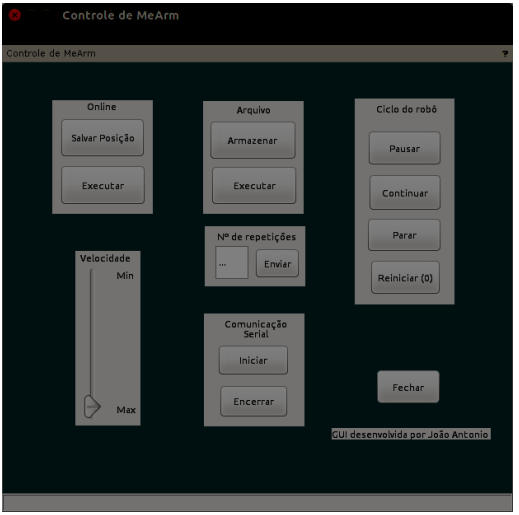
Figura 64
Janela de erro de escolha da porta USB



Fonte: Registrado pelo autor (2017).

Enfim, selecionadas as opções de configuração da comunicação serial, a interface de controle do robô estará disponível, como mostrada na figura 65.

Figura 65
Interface de controle do MeArm



Fonte: Registrado pelo autor (2017).

Os procedimentos realizados para conexão serial requerem maior atenção por dois motivos: ser essencial para comunicação dos dispositivose e o tempo gasto pelo sistema para estabelecer a conexão. Dependendo das configurações de *hardware* do sistema embarcado, o tempo para estabelecer conexão pode chegar a mais de 650ms (SCILAB, 2015). A princípio pode parecer irrisório, mas se conexão for feita durante a operação pode congestionar o *buffer* da transmissão serial com os outros comandos. Ainda pode ocorrer do operador esquecer de realizar a conexão manualmente, acarretando no não funcionamento do sistema.

Mesmo usuários sem perfil técnico terão facilidade em usar a IHM de controle do equipamento, pois é intuitiva e possui funções simples.

4.2. GERAÇÃO DE TRAJETÓRIA DO MEARM

A combinação dos métodos de aprendizagem mestre/escravo e PTP foi realizada com sucesso, visto que é possível controlar o robô e criar pontos de passagem facilmente usando a interface.

A programação do robô a nível de junta foi a mais apropriada, dado o método de aprendizagem mestre/escravo e a ausência de comandos pela interface para posicionamento e orientação.

O método usado para sincronização do movimento das juntas é excelente, pois garante a estabilidade dos movimentos, dado que sem atrasos o robô avança para o ponto seguinte sem nenhum impecilho.

Os comandos da interface de controle para salvar as posições da memória em arquivo, geram dados tabulados no formato .csv. Na figura 66 é mostrado um arquivo .csv gerado com duas posições salvas.

Figura 66
Pontos de passagem da trajetória do robô gerados através da interface de controle

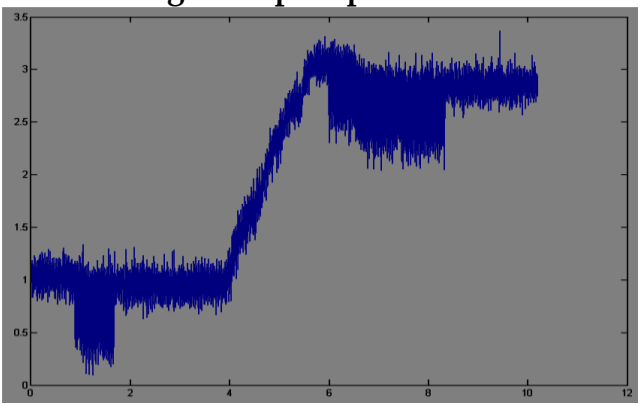
Fonte: Registrado pelo autor (2017).

Ainda é possível gerar os pontos de passgem do robô editando o arquivo gerado.

4.3. AQUISIÇÃO DE DADOS

Afim de verificar a precisão dos potenciômetros usados, foram feitos testes para verificar os sinais enviados ao arduino. Com base no gráfico gerado (figura 67), pode-se observar que nos períodos de rotação em que o potenciômetro sai da inércia há um transiente de tensão maior que em outros períodos.

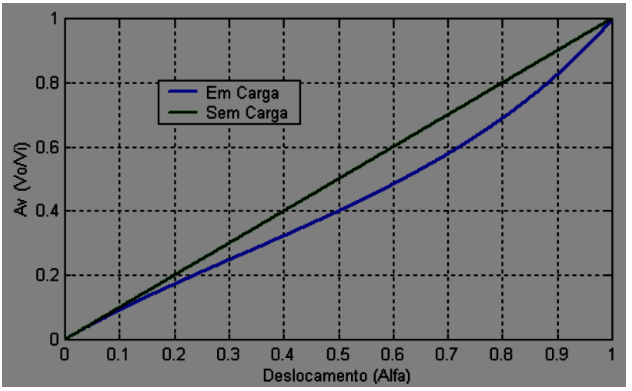
Figura 67
Sinal gerado pelo potenciômetro



Fonte: Registrado pelo autor (2017).

Isso acontece é devido às cargas presentes na saída de sinal do potenciômetro. Segundo Coelho (2004), este fenômeno é conhecido por efeito de carga. Na figura 68 pode-se observar graficamente o Efeito de Carga em um potenciômetro:

Figura 68
Efeito da resistência de entrada na linearidade



Fonte: Coelho (2004).

Esse efeito é constatado quando o manípulo é movido lentamente, pois o robô tremula. Se o manípulo é movido rapidamente o robô se mantém firme, exatamente como apresentado na figura 67.

Apesar de servir para o propósito pretendido, o potenciômetro não oferece grande precisão. Assim, o uso de encoders no manípulo mestre pode resolver essa questão.

4.4. SISTEMA EMBARCADO E TECNOLOGIAS LIVRE

A interface de usuário poderia ser executada a partir de um PC ou *notebook*, porém o uso de um sistema embarcado dedicado e de código aberto diminui os custo com hardware, facilita a troca em caso de quebra e facilita a manutenção em caso de falha. O RPi ainda suporta várias

melhorias que podem ser implementadas na interface de controle. Transmissão de dados via ethernet, controle por captura de movimento usando a entrada de câmera e conexão pelo seu GPIO são alguns exemplos. Pelo RPi ser popular, há muita informação disponível na internet e novos experimentos sendo divulgados livremente com constância.

O uso do linux embarcado garantiu a portabildiade e escalabildiade do sistema, pois foi possível executar um aplicativo como o Scilab em um processador ARM. Ainda é possível agregar novas funções à interface de usuário, visto que outros softwares podem ser embarcados.

A comunicação serial via USB pode ser substituída por outros meios e protocolos, como RS-232, RS-485, Ethernet, Modbus, OPC, SPI, I^2C, entre outras. Como são todos protocolos abertos, podem ser estudados e adaptados para o algoritmo de controle do MeArm.

5. CONSIDERAÇÕES FINAIS

Baseando-se nos resultados obtidos ao longo do trabalho podemos concluir que a implantação do sistema de controle para o MeArm é operacional. O método de aprendizagem mestre/escravo, que orienta e posiciona o braço robótico, usado em conjunto com a interface gráfica de usuário embarcada no Raspberry Pi, que gera os pontos de passagem da trajetória, formam um sistema prático, intuitivo e funcional.

O experimento demonstra que a integração dos componentes funcionam muito bem sem qualquer uso de recursos proprietários. O uso da comunicação serial para

comunicar o arduino (controlador do MeArm) com a IHM é apenas um exemplo do que pode ser implementado dentre as opções abertas disponíveis.

O uso do RPi, com o GNU/Linux embarcado Ubuntu MATE, se mostrou eficiente, em razão de ser compacto, de baixo custo, fácil de manter, de baixo consumo de eletricidade. Além de embarcar um SO customizável, escalável e fácil de operar.

Este trabalho comprova que o uso de tecnologias livres no projeto e desenvolvimento de sistemas de automação são possíveis e representam importante estratégia competiva, podendo estimular a criação de um novo modelo de negócio. A medida que protótipos mais robustos, de maior porte e com mais funções forem criados, as soluções abertas ganharão força no mercado. Se esse tipo de serviço vier acompanhado de um bom suporte, atualizações constantes, boa documentação e mão de obra especializada, se tornará tão popular e influente quanto o ROS-Industrial.

Dada a constante evolução da tecnologia, a implantação de robôs colaborativos e o uso de Sistemas Flexíveis de Manufatura (FMS) na fabricação de produtos altamente customizáveis, surge a necessidade por sistemas embarcados para controlar esses equipamentos. O uso de SBCs com Linux embarcado representam uma vantagem por serem de baixo custo, confiáveis e altamente customizáveis.

A utilização de soluções de código aberto são uma forte tendência no mercado, uma vez que temas como Internet das Coisas (do inglês: IoT, *Internet of Things*) e a

manufatura aditiva (impressão 3D) se tornaram populares. Com a evolução da IoT, por exemplo, surge a necessidade por conectividade e suas aplicações para diversos dispositivos. Dessa forma é muito caro para que a empresa fique criando soluções proprietárias para todas as formas de conexão que surgirão. Já a popularidade da impressão 3D, por sua vez, permite que existam vários fabricantes de peças, o que diminui o custo de aquisição e descentraliza a fabricação de artigo similares das grandes indústrias. A popularização dessas tecnologias tornam os seus produtos mais seguros e acessíveis à todos, criando tendências de consumo e novos patamares de desenvolvimento humano.

Este trabalho e todos os seus resultados são disponibilizados livremente na internet, tais quais os softwares e projetos de hardwares usados em seu desenvolvimento, no link: https://github.com/Joaotoin/Controle-MeArm.

Sugere-se para trabalhos futuros:

- Uso de encoders no lugar de potenciômetros para captação dos movimentos dados pelo operador;
- Usar os dados tabulados em arquivos .csv para gerar gráficos e mapear as trajetórias;
- Controlar robô através de gestos usando a biblioteca Open CV do Scilab;
- Implementar protocolo de comunicação serial com *checksum*;
- Utilizar biblioteca já existente de cinemática inversa do Mearm (MeArm IK) e implementá-la na interface de controle;

- Substituir o MeArm por um braço robótico com sensores de posição das juntas, assim possibilitando o controle em malha fechada;
- Implementar controle em malha fechada para posição, velocidade e aceleração;
- Realizar testes de precisão e repetibilidade;
- Realizar modelagem geométrica do MeArm e criar modelo de cinemática direta e inversa;
- Simular modelagem no ROS;
- Impletar novas formas de controle e operação através da Internet das Coisas.

REFERÊNCIAS

ANGELES, Jorge. **Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms**. 3rd ed. New York: Springer, 2007.

APOCALYPSE, Priscila P; CHAIB, Ramon P. S. **Device drivers em linux embarcad**o. Disponível em: <https://github.com/priscila225/DeviceDrivers>. Acesso em 20 fev 2017.

ARDUINO. **Language Reference**. 2017. Disponível em: <https://www.arduino.cc/en/Reference/HomePage>. Acesso em: 2 fev 2017.

ARDUINO. **Language Reference**. Disponível em: <https://www.arduino.cc/en/Reference/HomePage>. Acesso em: 2 fev 2017.

ATMEL. **Datasheet: Atmel Atmega640/V-1280/V-1281/V-2560/V-2561/V**. 2014. Disponível em: <http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-Atmega640-1280-1281-2560-2561_datasheet.pdf>. Acessoem 20 dez 2014.

AULEDAS. **Teach Pedant ABB model IRB140**. Disponível em: <https://commons.wikimedia.org/wiki/File%3ATeach_Pendant_ABB.JPG>. Acesso em: 28 jan 2017.

BIT OF A HACK. **MeArm IK**. Disponível em: <http://bitofahack.com/post/1433701488>. Acesso em 20 jan 2017.

BIZELLO, S., RUSCHEL, R. **Estudo de CAD livre para implementação de ferramenta de projeto**. Gestão & Tecnologia de Projetos, São Carlos, v. 6, n. 1, maio. 2011. Disponível em: <http://www.iau.usp.br/posgrad/gestaodeprojetos/index.php/gestaodeprojetos/article/view/171>. Acesso em: 02 Jan. 2014.

CABRAL. Eduardo L. L. **Cinemática direta de robôs manipuladores**. 2004. Disponível em: <http://sites.poli.usp.br/p/eduardo.cabral/Cinem%C3%A1tica%20Direta.pdf>. Acesso em 25 fev 2017.

CARRARA, Valdemir. **Robótica**. São Paulo, 2008, 95 p. Apostila do curso de Engenharia de Controle e Automação e Engenharia Mecânica – Universidade Braz Cubas.

CHRISTIAN, Yhan. **Controle PID de temperatura com Arduino e Scilab**. Disponível em: <http://engenheirocaicara.com/projeto-do-mes-parte-1-controle-pid-de-temperatura-com-arduino-e-scilab/>. Acesso em 20 jan 2017.

COCOTA JÚNIOR, A. N. J.; FUJITA, H. S.; SILVA, I. J. **Um Robô Manipulador de Baixo Custo para a**

Educação. X Congreso de Tecnologías Aplicadas a la Enseñanza de la Electrónica. 2012:164-169.

COCOTA JÚNIOR, A. N. J.; MOREIRA, A. D.; BARBOSA, R. C.; LAGE, V. **Desenvolvimento de um Robô Antropomórfico com Punho Esférico para Práticas de Robótica com Alunos de Graduação**. SBAI [Internet]. 2013;1(1):1-6.

CUNHA, Alessandro F. **O que são sistemas embarcados**. 2010. Disponível em: <http://files.comunidades.net/mutcom/ARTIGO_SIST_EMB.pdf>. Acesso em 20 fev 2017.

DEBRAY. Yann; SCILAB ENTERPRISES. **Temperature monitoring tutorial with Scilab/Xcos and arduino**. Disponível em: <https://www.scilab.org/community/news/Scilab-Arduino-low-cost-data-acquisition>. Acesso em: 22 dez. 2016.

EDWARDS, Shaun. **ROS-Industrial: a open source case study**. 2013. Disponível em: <http://roscon.ros.org/2013/wp-content/uploads/2013/05/ROSCon-ROS-Industrial-Open-Source-Case-Study-Rev-0.04.pdf>. Acesso em 3 fev. 2017.

EPUSP.DControle de um servo motor. 2014. Disponível em:

<http://www2.pcs.usp.br/~labdig/pdffiles_2014/controle-servo-semestral.pdf>. Acesso em 20 jan 2017.

FILIPE FLOP. **Sensor Shield V4.0 para Arduino**. 2016. Disponível em <http://www.filipeflop.com/pd-6b5e7-sensor-shield-v4-0-para-arduino.html?ct=3d60f&p=2&s=1>. Acesso em 20 jan 2017.

FLORENCIO. Heitor M. **Sistemas Embarcados: Protocolos de Comunicação - Serial**. Disponível em: <http://www.dca.ufrn.br/~heitorm/aulasDCA/dca0119/DCA0119-11-SistemasEmbarcados-Comunicacao-Serial.pdf>. Acesso em 25 fev 2017.

FRITIZING. **Learning**. Disponível em: <http://fritzing.org/learning/>. Acesso em 3 fev. 2017.

GEORGIA TECH. **ROS Commander (ROSCo): Behavior Creation for Home Robots**. Disponível em:<http://pwp.gatech.edu/hrl/ros-commander-rosco-behavior-creation-for-home-robots/>. Acesso em 3 fev. 2017.

GROOVER, Mikell P. **Automação Industrial e Sistemas de Manufatura**. Tradução: Jorge Ritter, Luciana Amaral Teixeira, Marcos Vieira; revisão técnica José Hamilton Chaves Gorgulho Júnior. 3 ed. São Paulo: Pearson Prentice Hall, 2011.

Improprietário: o mundo do software livre. Direção: Jota Rodrigo. Roteiro, pesquisa, entrevistas e imagens: Daniel Pereira Bianchi e Jota Rodrigo. Osasco-SP, 2008. Documentário, 32'04". Disponível em: <http://www.youtube.com/watch?v=MKDn9quw5sc>. Acesso em: 25 fev. 2014.

INOVAÇÃO TECNOLÓGICA. **Jovem engenheira vira ícone do Hardware Livre**. Disponível em: <http://www.inovacaotecnologica.com.br/noticias/noticia.php?artigo=jovem-engenheira-vira-icone-hardware-livre#.Uy7yZfldVpQ>. Acesso em 16 mar. 2014.

LOPES, M. A. M. **A importância dos sistemas supervisórios no controle de processos industriais**. 49f. Monografia (Curso de Controle e Automação) – UFOP, Ouro Preto, 2009.

MA, L.; FENG, X.; PENG, Z. **Integrated Design and Implementation of Embedded Control Systems with Scilab**. Sensors 8, 5501-5515 (2008). Disponível em: <http://www.mdpi.com/1424-8220/8/9/5501/pdf>. Acesso em 20 jan 2017.

MACIEL, Francisco. **O que é Linux Embarcado**. 2014. Disponível em: <http://softwarelivre.blog.br/2014/05/24/o-que-e-linux-embarcado/>. Acesso em 24 fev 2017.

MADLAB. **QUIPT: taming industrial robots**. Disponível em: <http://www.madlab.cc/quipt/>. Acesso em 4 fev. 2017.

MECATRÔNICA ATUAL. **O linux na indústria**. 2011. Disponível em: <http://www.mecatronicaatual.com.br/educacao/903-o-linux-na-indstria>. Aceso em: 10 mar. 2014.

MEDINA, Rene M.; CRISPIM, Sérgio F. **Fatores determinantes no processo de decisão de investimentos em robotização na indústria brasileira de autopeças**. Gestão & Produção, São Carlos, v. 17, n. 3, p. 567-578, 2010. Disponível em: <http://www.scielo.br/pdf/gp/v17n3/10.pdf>. Acesso em: 14 nov. 2013.

MIME INDUSTRIES. **MeArm - Pocket Sized Robot Arm**. Disponível em: <https://github.com/mimeindustries/MeArm>. Acesso em 3 fev. 2017.

MOLLOY, Derek. **Exploring Raspberry Pi(R) - Interfacing to the real world with embedded linux**. Indianapolis: Wiley, 2016.

MULTICHERRY. **Top half of Raspberry Pi 2 Model B v1.1 seen from rear at angle**. 2015. Disponível em: <https://commons.wikimedia.org/wiki/File%3ARaspberry

_Pi_2_Model_B_v1.1_rear_angle_new.jpg>. Acesso em 20 jan 2017.

MUÑOZ, Juan J. P. **Control de temperatura y humedad de uma maqueta de invernadero mediante tecnologia Open Source**. 2014. Disponível em: <http://repositorio.upct.es/handle/10317/4215>. Acesso em 12 dez 2016.

NOERGAARD, Tammy. **Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers**. Newnes: Burlington, 2005.

OCTANS AUTOMAÇÃO. **Automação na soldagem**. Disponível em: <http://www.soldaautomatica.com.br/index_arquivos/Arquivos/Automacao%20na%20Soldagem%20-%20UNB.pdf>. Acesso em: 12 dez 2016.

OOMLOUT. **Arduino Mega**. 2009. Disponível em <https://commons.wikimedia.org/wiki/File%3AArduino_Mega_2.jpg>. Acesso em: 20 jan. 2017.

OOMLOUT. **Micro servo. 2009**. Disponível em: <https://commons.wikimedia.org/wiki/File%3AMicro_servo.jpg>. Acesso em 20 jan 2017.

OPEN SOURCE INITIATIVE. **The Open Source Initiative Keyhole Logo**. Disponível em: <https://opensource.org/>. Acesso em 3 fev. 2017.

OSADL. **Our services at a glance**. Disponível em: <https://www.osadl.org/Benefits.why-osadl.0.html>. Acesso em 3 fev. 2017.

OSIER-MIXON, Jeffrey M. **Hardware Aberto: Como e Quando Funciona: aplicando conceitos de software livre à objetos físicos**. 2010 Disponível em: < http://www.ibm.com/developerworks/br/library/os-openhardware/> Acesso em: 17 mar. 2014.

PAREDE, Ismael M.; GOMES, Luiz E. L; HORTA, Edson. **Eletrônica: Automação Industrial**. São Paulo: Fundação Padre Anchieta, 2011.

PHENOPTIX. **MeArm V0.4 - Pocket Sized Robot Arm**. Disponível em: <https://commons.wikimedia.org/wiki/File%3ABlueblack_preview_featured.jpg>. Acesso em: 28 jan. 2017.

PRO-FACE. **Connection diagram of programmable display and robot controller** Disponível em: <http://www.proface.eu/press/2012/0725.html>. Acesso em: 2 fev. 2017.

RAGHAVAN, P.; LAD, Amol; NEELAKANDAN, Sriram. **Embedded Linux System Design and Development**. Boca Raton: Auerbach Publications, 2006.

RASC. **Master/Slave Arm**. Disponível em: <http://rasc.usc.edu/masterslave-arm.html>. Acesso em: 28 jan. 2017.

RASPBERRY PI. **About us**. Disponível em: <https://www.raspberrypi.org/about/>. Acesso em 20 fev 2017.

RAYSARO, Márcio Coiado. **Sistema open-source de supervisão, controle e aquisição de dados**. Cuiabá. 2012. Universidade de Cuiabá. Curso superior em Sistema de Informação.

ROBOTICS TOMORROW. **Getting Started with Collaborative Robots? Part 1 - What can collaborative robots do?** Disponível em: <http://www.roboticstomorrow.com/article/2015/12/getting-started-with-collaborative-robots-part-1—what-can-collaborative-robots-do/7298/>. Acesso em: 28 jan. 2017.

ROS-Industrial. **RIC-Americas Annual Meeting – 2015**. Disponível em: <http://rosindustrial.org/news/2015/3/4/ric-americas-annual-meeting>. Acesso em 3 fev. 2017.
ROSÁRIO, João Maurício. **Automação Industrial**. São Paulo: Baraúna, 2009.

ROSÁRIO, João Maurício. **Princípio de Mecatrônica**. São Paulo: Prentice Hall, 2005.

ROSÁRIO, João Maurício. **Robótica Industrial I: Modelagem, Utilização e Programação**. São Paulo: Baraúna, 2010.

SÁ, A. O.; PERDIGÃO, G. O.; SANTOS, R. O. S.; CARVALHO, R. S. **Impacto dos softwares livre na indústria de software brasileira**. 2011. Disponível em: <http://www.periodicos.letras.ufmg.br/index.php/ueadsl/article/viewFile/2925/2884>. Acesso em 23 fev. 2014.

SCADABR. **Centro de Controle e Operação**. Disponível em: <http://www.scadabr.com.br/?q=case_cco>. Acesso em: 1 fev. 2017.

SCILAB ENTERPRISES. **Scilab for very beginnners**. Disponível em: <https://scilab.io/scilab-for-beginners-tutorial/>. Acesso em: 22 dez 2016.

SCILAB. **About Scilab**. 2015. Disponível em: <http://www.scilab.org/scilab/about>. Acesso em 22 jan 2017.

SCILAB. **Ajuda do Scilab**. 2015. Disponível em: <https://help.scilab.org/docs/5.5.2/pt_BR/index.html>. Acesso em: 2 fev 2017.

SCILAB. **Ajuda do Scilab**. https://help.scilab.org/docs/5.5.2/pt_BR/index.html>. Acesso em: 2 fev 2017.

SCILAB. **ATOMS : Serial Communication Toolbox details**. Disponível em: <https://atoms.scilab.org/toolboxes/serial>. Acesso em 25 fev 2017.

SENA, António Sérgio. Microcontroladores PIC. Disponível em: <http://www.portugal-a-programar.pt/forums/topic/54728-microcontroladores-pic/>. Acesso em: 28 jan. 2017.

SILVA, E. M., CUNHA, J. P. V. S. **SCILAB, SCICOS e RLTOOL: Softwares Livres no Ensino de Engenharia Elétrica**. Anais do Congresso Brasileiro de Automática, XVI, 2006. Salvador, Brasil, pp. 1620-1625. Disponível em: <http://www.lee.eng.uerj.br/~elaine/501.pdf>. Acesso em: 31 Dez 2013.

SILVEIRA. Sérgio Amadeu. **Software Livre: a luta pela liberdade do conhecimento**. São Paulo: Editora Fundação Perseu Abramo, 2004.

SOCIETY OF ROBOTS. **Motion planning**. Dinponível em: <http://www.societyofrobots.com/robot_arm_tutorial.shtml>. Acesso em: 28 jan. 2017.

SOFTEX; UNICAMP; Ministério da Ciência e Tecnologia. **O Impacto do Software Livre e de Código Aberto na Indústria de Software do Brasil**. Campinas: Softex, 2005. Disponível em:

<http://www.mct.gov.br/upd_blob/0008/8690.pdf>. Acesso em: 11 mar. 2014.

SOFTEX. **Software livre: tendências, oportunidades e desafios**. Campinas: Softex, 2014. Disponível em: <http://www.softex.br/download/71029>. Acesso em: 20 dez. 2016.

SOUZA, Fábio. **Arduino MEGA 2560**. 2014. Disponível em: <https://www.embarcados.com.br/arduino-mega-2560/>. Acesso em: 20 jan 2017.

TECHTONIC. **USB Types**. Disponível em: <https://commons.wikimedia.org/wiki/File%3AUSB_types_2.jpg>. Acesso em: 2 fev. 2017.

THE OPEN SOURCE HARDWARE. **Open source hardware-Logo**. Disponível em: <http://www.oshwa.org>. Acesso em 3 fev. 2017.

TOWER PRO. **Datasheet. SG90 Digital**. Disponível em: <http://www.micropik.com/PDF/SG90Servo.pdf>. Acesso em 20 jan 2017.

UNIVERSAL SERIAL BUS. **USB Specifitaions**. Disponível em: <http://www.usb.org/developers/docs/>. Acesso em 4 fev. 2017.

VALLE, Cyro E. **Qualidade Ambiental: ISO 14000**. 5. ed. São Paulo: editora Senac São Paulo, 2004.

VIANNA, Maria L. R.; ANGELO, Edvaldo; ANGELO, Gabriel. **Mecânica: automação**. São Paulo: Fundação Padre Anchieta, 2011 (Coleção Técnica Interativa. Série Mecânica, v. 4)

YASKAWA. **What is a Robot?** Disponível em: <https://www.yaskawa.co.jp/en/product/robotics/about>. Acesso em: 2 fev. 2017.

Desde o fim da segunda guerra mundial, quando a produção industrial sofreu grande aumento, novos modelos de gestão e tecnologias vem sendo desenvolvidas. Para atender as diversas exigências foi criada uma tecnologia que faça uso de sistemas mecânicos, elétricos, eletrônicos e de informática para efetuar controle de sistemas produtivos: a automação. Consequentemente, a necessidade de se realizar certas tarefas com mais eficiência e precisão levou a criação de manipuladores robóticos. Para programar um robô é necessário inserir as instruções em seu sistema de controle, que as transmitirá para os atuadores, basicamente. Dependendo da forma como ele é programado, a sua produtividade é afetada, pois despende-se muito tempo configurando-o. Outro fator inconveniente é a sua integração a outros sistemas e a enorme dependência do fabricante para adaptações do equipamento, visto que são usadas soluções fechadas. O objetivo deste trabalho é criar uma interface de controle para um manipulador robótico de 3 graus de liberdade (MeArm), implementar função de aprendizagem e embarcar o sistema utilizando apenas software e hardware de código aberto.

ISBN 978-65-83366-27-6
978-65-83366-27-6
KattleYa EDITORA