MARCO VINÍCIUS MUNIZ FERREIRA

AVALIAÇÃO DE PROJETO E IMPLEMENTAÇÃO DE REDES DE DISPOSITIVOS CENTRALIZADOS E DISTRIBUÍDOS APLICADOS EM SISTEMAS A EVENTO DISCRETO

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA MECÂNICA

2015

MARCO VINÍCIUS MUNIZ FERREIRA

AVALIAÇÃO DE PROJETO E IMPLEMENTAÇÃO DE REDES DE DISPOSITIVOS CENTRALIZADOS E DISTRIBUÍDOS APLICADOS EM SISTEMAS A EVENTO DISCRETO

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Mecânica da Universidade Federal de Uberlândia, como parte dos requisitos para a obtenção do título de **MESTRE EM ENGENHARIA MECÂNICA**.

Área de Concentração: Mecânica dos Sólidos e Vibrações.

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

UBERLÂNDIA - MG

2015

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

F383a
2015

Ferreira, Marco Vinícius Muniz, 1988-
Avaliação de projeto e implementação de redes de dispositivos centralizados e distribuídos aplicados em sistemas a evento discreto / Marco Vinícius Muniz Ferreira. - 2015.
169 f. : il.

Orientador: José Jean-Paul Zanlucchi de Souza Tavares.
Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Engenharia Mecânica.
Inclui bibliografia.

1. Engenharia mecânica - Teses. 2. Arquitetura de redes de computador - Teses. 3. Teoria do controle - Teses. I. Tavares, José Jean-Paul Zanlucchi de Souza, 1962- II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Engenharia Mecânica. III. Título.

CDU: 621

Dedico este trabalho a minha família e namorada, por sempre me incentivarem ao longo da minha carreira acadêmica.

Agradecimentos

Aos meus pais, as minhas irmãs e minha namorada, pelo carinho, amor, compreensão e incentivo nas horas mais difíceis.

À Universidade Federal de Uberlândia e à Faculdade de Engenharia Mecânica pela oportunidade de realizar esse curso.

À CAPES pelo apoio financeiro oferecido durante a realização deste trabalho.

Aos amigos que fazem parte da 6ª Turma de Engenharia Mecatrônica e aqueles que se juntaram a nós durante o curso, que apoiaram e ajudaram em todos os momentos.

Ao professor Dr. José Jean-Paul Z. de Souza Tavares pelas orientações, críticas e cobranças. Por ser um exemplo de amor à profissão, que conduz seus alunos e a ciência com excelência, garra, força, motivação, inspiração e muito conhecimento.

Aos companheiros do *MAPL* (*Manufacturing Automated Planning Lab*), agradeço pela ajuda na realização do projeto, pela nossa amizade e pelo exemplo a ser seguido.

E a todos os amigos da Universidade Federal de Uberlândia.

FERREIRA, M. V. M. Avaliação de projeto e implementação de redes de dispositivos centralizados e distribuídos aplicados em sistemas a evento discreto de baixa criticidade e baixo custo. 2015. 169 f. Dissertação de Mestrado, Universidade Federal de Uberlândia, Uberlândia.

Resumo

A automação moderna difere-se da clássica pela utilização de dispositivos microeletromecânicos e sistemas embarcados para garantir a lógica de controle. Tais dispositivos têm a capacidade de tomadas de decisões, seja de forma centralizada ou distribuída. Os sistemas a evento discreto possuem aplicação em várias áreas de engenharia, tais como automação da manufatura, robótica, supervisão de tráfego, logística e otimização de processos distribuídos. O objetivo deste trabalho é avaliar o impacto de projetos de redes de dispositivos centralizados e distribuídos aplicados em sistemas a evento discreto. Para isso é proposto um estudo de caso baseado em uma bancada de separação de peças e são avaliadas diversas arquiteturas de redes com relação ao desempenho, projeto e instalação.

Palavras Chave: arquitetura de controle de sistemas, sistema centralizado e distribuído, sistema a evento discreto de baixa criticidade, Arduino.

FERREIRA, M. V. M. Avaliação de projetos de redes de dispositivos centralizados e distribuídos aplicados em sistemas a evento discreto de baixa criticidade e baixo custo. 2015. 169 f. Dissertação de Mestrado, Universidade Federal de Uberlândia, Uberlândia.

Abstract

Modern automation approach differs from classical one by using embedded devices and microelectromechanical systems integrated with the control logic. Such devices have capability of decision making, either in centralized or distributed systems. The discrete event systems have application in various fields of engineering, such as manufacturing automation, robotics, traffic supervision, logistics and distributed process optimization. The aim of this work is to evaluate the device networks project impact in centralized and distributed systems applied to a discrete event. To achieve the goal a case study based on a parts separation bench is proposed and analyzed different network architectures of project, deployment and performance.

Keywords: system control architecture, centralized and distributed systems, discrete event system with low criticality, Arduino.

Lista de Figuras

Figura 2.1. Pirâmide da Automação Industrial. 6
Figura 2.2. Parte de um programa implementado por todas as linguagens do padrão IEC 61131-3. Adaptado de (PLCOpen, 2015). 9
Figura 2.3. Exemplo de SFC aplicado na Eq. 1. 9
Figura 2.4. Sistema Descentralizado. Adaptado de (Schuppen et al., 2011). 11
Figura 2.5. Sistema distribuído. Retirado de (Schuppen et al., 2011). 11
Figura 2.6. Exemplo de sistema distribuído. Adaptado de (Flammini et al., 2008). 12
Figura 2.7. Controle distribuído. Retirado de (Schuppen et al., 2011). 13
Figura 2.8. Controle distribuído com comunicação entre controladores. Retirado de (Schuppen et al., 2011). 14
Figura 2.9. Sistema de controle coordenado. Adaptado de (Schuppen et al., 2011). 14
Figura 2.10. Indústria de petróleo e gás. (FIELDBUS FOUNDATION, 2013). 16
Figura 2.11. Rede de sensores sem fio. Adaptado de (AKYILDIZ et al., 2002). 17
Figura 2.12. Rede para monitoramento de paciente em residências. Adaptado de (ALEMDAR & ERSOY, 2010). 19
Figura 2.13. Sistema de recuperação de quadril. Adaptado de (ISO-KETOLA et al., 2008). 19
Figura 2.14. Sistema de detecção de atividade vulcânica. Adaptado de (WERNER-ALLEN et al, 2005). 20
Figura 2.15. Aplicação residencial de RSSF. Retirado de http://revistahometheater.uol.com.br . 22
Figura 2.16. Camadas das RSSF. Adaptado de (AKYILDIZ et al., 2002). 23
Figura 2.17. Topologias das RSSF. 24
Figura 2.18. Problemas apresentados para agregação de dados. 26
Figura 2.19. Exemplo de um barramento I²C utilizando dois microcontroladores. 29
Figura 2.20. Esquema básico de um microcontrolador. 30
Figura 2.21. (a) *Arduino* Mega 2560; (b) *Arduino* Uno. 32
Figura 2.22. Exemplo de sistema eletropneumático. 35
Figura 2.23. (a) Grafcet do Sistema e (b) Grafcet modificado para transcrição em diagrama ladder. 36
Figura 2.24. Diagrama ladder do exemplo utilizado. 37
Figura 2.25. Alteração no software para transferência do diagrama ladder para *Arduino*. 38

Figura 4.1. Bancada de separação de peças XC243. 42
Figura 4.2. Peças metálicas e plásticas de três tamanhos distintos. 43
Figura 4.3. Fluxograma resumido do sistema. 44
Figura 4.4. Fluxograma da identificação das peças. 45
Figura 4.5. Fluxograma da separação das peças. 46
Figura 4.6. Esquema de conexão das entradas e saídas do sistema centralizado. 48
Figura 4.7. Sistema distribuído com comunicação serial entre dispositivos. 50
Figura 4.8. Sistema distribuído com comunicação I²C entre dispositivos. 51
Figura 4.9. Sistema distribuído com comunicação I²C entre dispositivos utilizando um mestre e dois escravos. 52
Figura 4.10. Sistema distribuído com comunicação I²C entre dispositivos com a lógica de separação de peças distribuída. 53
Figura 4.11. Fluxograma da separação de peças para cada um dos escravos. 54
Figura 4.12. Esquema de conexão das entradas e saídas do sistema distribuído com comunicação *Zigbee* entre dois dispositivos. 55
Figura 5.1. Histograma dos dados para o sistema centralizado. 59
Figura 5.2. Função de distribuição acumulada para peça metálica do sistema centralizado. 60
Figura 5.3. Função de distribuição acumulada para peça grande do sistema centralizado. . 60
Figura 5.4. Função de distribuição acumulada para peça média do sistema centralizado. .. 61
Figura 5.5. Função de distribuição acumulada para peça pequena do sistema centralizado. 61
Figura 5.6. Histograma dos dados para o sistema distribuído com comunicação serial entre os dispositivos a uma taxa de 115200 bps e tempo de solicitação de mensagem de 4 ms. 64
Figura 5.7. Função de distribuição acumulada para peça metálica do sistema distribuído com comunicação serial *RxTx* entre dispositivos do ensaio 1. 65
Figura 5.8. Função de distribuição acumulada para peça grande do sistema distribuído com comunicação serial *RxTx* entre dispositivos do ensaio 1. 65
Figura 5.9. Função de distribuição acumulada para peça média do sistema distribuído com comunicação serial *RxTx* entre dispositivos do ensaio 1. 66
Figura 5.10. Função de distribuição acumulada para peça pequena do sistema distribuído com comunicação serial *RxTx* entre dispositivos do ensaio 1. 66
Figura 5.11. Histograma dos dados para o sistema distribuído com comunicação serial entre os dispositivos a uma taxa de 115200 bps e tempo de solicitação de mensagem de 50 ms. 68
Figura 5.12. Histograma dos dados para o Sistema distribuído com comunicação serial entre os dispositivos a uma taxa de 1200 bps e tempo de solicitação de mensagem de 100 ms. . 69

Figura 5.13. Histograma dos dados para o Sistema distribuído com comunicação serial entre os dispositivos a uma taxa de 1200 bps e tempo de solicitação de mensagem de 800 ms.. 71
Figura 5.14. Histograma dos dados para o sistema distribuído com comunicação I²C entre dois dispositivos para o ensaio 1. 73
Figura 5.15. Função de distribuição acumulada para peça metálica do sistema distribuído com comunicação I²C entre dispositivos do ensaio 1. 74
Figura 5.16. Função de distribuição acumulada para peça grande do sistema distribuído com comunicação I²C entre dispositivos do ensaio 1. 75
Figura 5.17. Função de distribuição acumulada para peça média do sistema distribuído com comunicação I²C entre dispositivos do ensaio 1. 75
Figura 5.18. Função de distribuição acumulada para peça pequena do sistema distribuído com comunicação I²C entre dispositivos do ensaio 1. 76
Figura 5.19. Histograma dos dados para o Sistema distribuído com comunicação I²C entre os dispositivos, sendo dois escravos. 78
Figura 5.20. Função de distribuição acumulada para peça metálica do sistema distribuído com comunicação I²C entre dispositivos do ensaio 2. 79
Figura 5.21. Função de distribuição acumulada para peça grande do sistema distribuído com comunicação I²C entre dispositivos do ensaio 2. 79
Figura 5.22. Função de distribuição acumulada para peça média do sistema distribuído com comunicação I²C entre dispositivos do ensaio 2. 80
Figura 5.23. Função de distribuição acumulada para peça pequena do sistema distribuído com comunicação I²C entre dispositivos do ensaio 2. 80
Figura 5.24. Histograma dos dados para sistema distribuído com comunicação I²C entre dispositivos com a lógica de separação de peças distribuída. 82
Figura 5.25. Função de distribuição acumulada para peça metálica do sistema distribuído com comunicação I²C entre dispositivos do ensaio 3. 83
Figura 5.26. Função de distribuição acumulada para peça grande do sistema distribuído com comunicação I²C entre dispositivos do ensaio 3. 84
Figura 5.27. Função de distribuição acumulada para peça média do sistema distribuído com comunicação I²C entre dispositivos do ensaio 3. 84
Figura 5.28. Função de distribuição acumulada para peça pequena do sistema distribuído com comunicação I²C entre dispositivos do ensaio 3. 85
Figura 5.29. Histograma dos dados para sistema distribuído com comunicação *Zigbee* entre dispositivos. 87
Figura 5.30. Função de distribuição acumulada para peça metálica do sistema distribuído com comunicação *Zigbee* entre dispositivos..... 88

Figura 5.31. Função de distribuição acumulada para peça grande do sistema distribuído com comunicação *Zigbee* entre dispositivos..... 88
Figura 5.32. Função de distribuição acumulada para peça média do sistema distribuído com comunicação *Zigbee* entre dispositivos..... 89
Figura 5.33. Função de distribuição acumulada para peça pequena do sistema distribuído com comunicação *Zigbee* entre dispositivos..... 89
Figura 5.34. Intervalo de confiança para peça metálica no sistema distribuído com comunicação *RxTx* entre dispositivos. 91
Figura 5.35. Intervalo de confiança para peça grande no sistema distribuído com comunicação *RxTx* entre dispositivos. 91
Figura 5.36. Intervalo de confiança para peça média no sistema distribuído com comunicação *RxTx* entre dispositivos. 92
Figura 5.37. Intervalo de confiança para peça pequena no sistema distribuído com comunicação *RxTx* entre dispositivos. 92
Figura 5.38. Intervalo de confiança para peça metálica no sistema distribuído com comunicação I²C entre dispositivos. 94
Figura 5.39. Intervalo de confiança para peça grande no sistema distribuído com comunicação I²C entre dispositivos. 95
Figura 5.40. Intervalo de confiança para peça média no sistema distribuído com comunicação I²C entre dispositivos. 95
Figura 5.41. Intervalo de confiança para peça pequena no sistema distribuído com comunicação I²C entre dispositivos. 96
Figura 5.42. Intervalo de confiança para peça metálica nas arquiteturas de controle..... 98
Figura 5.43. Intervalo de confiança para peça grande nas arquiteturas de controle..... 98
Figura 5.44. Intervalo de confiança para peça média nas arquiteturas de controle. 99
Figura 5.45. Intervalo de confiança para peça pequena nas arquiteturas de controle. 99

Lista de Tabelas

Tabela 2.1. Divisões da família *IEC 61131* 8
Tabela 2.2. Detalhes dos projetos do *C4C Projet.* Disponível em http://www.c4c-project.eu 16
Tabela 2.3. Definição da termologia do barramento I²C. 29
Tabela 2.4. Trabalhos que utilizam microcontroladores em diversas áreas de atuação. 31
Tabela 2.5. Características dos *Arduino*s Uno e Mega 2560..... 32
Tabela 2.6. Tabelas de-para para o GRAFCET modificado. 36
Tabela 2.7. Lógica do sistema em português estruturado para o sistema eletropneumático. 38
Tabela 3.1. Experimentos a serem realizados. 40
Tabela 4.1. Lista de dispositivos utilizados na bancada. 42
Tabela 4.2. Massas das peças. 43
Tabela 4.3. Tabela verdade da identificação das peças..... 47
Tabela 4.4. Características dos quatro ensaios do sistema distribuído com comunicação serial *RxTx* entre dispositivos. 49
Tabela 4.5. Quadro de mensagens enviado pelo escravo ao mestre. 50
Tabela 4.6. Quadro da mensagem do escravo A. 52
Tabela 4.7. Quadro da mensagem do escravo B. 52
Tabela 5.1. Características do ensaio do sistema centralizado. 57
Tabela 5.2. Dados estatísticos para sistema centralizado. 58
Tabela 5.3. Dados estatísticos e teste de hipóteses para sistema centralizado. 59
Tabela 5.4. Características do Sistema distribuído com comunicação serial *RxTx* entre dispositivos. 63
Tabela 5.5. Dados estatísticos para o sistema distribuído com comunicação serial *RxTx* para o ensaio 1. 63
Tabela 5.6. Dados estatísticos e teste de hipóteses para sistema distribuído com comunicação serial *RxTx* entre dispositivos do ensaio 1. 64
Tabela 5.7. Dados estatísticos para Lógica Distribuída *RxTx* com 115200 bps de taxa de transmissão de dados e 50 ms de tempo de solicitação de mensagem. 67
Tabela 5.8. Dados estatísticos e teste de hipóteses para sistema distribuído com comunicação serial *RxTx* entre dispositivos do ensaio 2. 67

Tabela 5.9. Dados estatísticos para Lógica Distribuída *RxTx* com 1200 bps de taxa de transmissão de dados e 100 ms de tempo de solicitação de mensagem. 69
Tabela 5.10. Dados estatísticos e teste de hipóteses para sistema distribuído com comunicação serial *RxTx* entre dispositivos do ensaio 3. 70
Tabela 5.11. Dados estatísticos para Lógica Distribuída *RxTx* com 1200 bps de taxa de transmissão de dados e 800 ms de tempo de solicitação de mensagem. 70
Tabela 5.12. Dados estatísticos e teste de hipóteses para sistema distribuído com comunicação serial *RxTx* entre dispositivos do ensaio 4. 71
Tabela 5.13. Características do Sistema distribuído com comunicação serial *RxTx* entre dispositivos para o ensaio 1. 72
Tabela 5.14. Dados estatísticos para sistema distribuído com comunicação I²C entre os dispositivos para o ensaio 1. 73
Tabela 5.15. Dados estatísticos e teste de hipóteses para sistema distribuído com comunicação I²C entre dispositivos do ensaio 1. 74
Tabela 5.16. Características do Sistema distribuído com comunicação I²C entre três dispositivos para o ensaio 2. 77
Tabela 5.17. Dados estatísticos para Sistema Distribuído com comunicação I²C entre dispositivos, sendo dois escravos. 77
Tabela 5.18. Dados estatísticos e teste de hipóteses para sistema distribuído com comunicação I²C entre dispositivos do ensaio 2. 78
Tabela 5.19. Características do ensaio de sistema distribuído com comunicação I²C entre dispositivos e lógica distribuída. 81
Tabela 5.20. Dados estatísticos para Sistema distribuído com comunicação I²C entre dispositivos com a lógica de separação de peças distribuída. 82
Tabela 5.21. Dados estatísticos e teste de hipóteses para sistema distribuído com comunicação I²C entre dispositivos do ensaio 3. 83
Tabela 5.22. Características do Sistema distribuído com comunicação *Zigbee* entre dispositivos. 86
Tabela 5.23. Dados estatísticos para Sistema distribuído com comunicação *Zigbee* entre dispositivos. 86
Tabela 5.24. Dados estatísticos e teste de hipóteses para sistema distribuído com comunicação *Zigbee* entre dispositivos. 87
Tabela 5.25. Teste F para o Sistema distribuído com comunicação serial *RxTx* entre dispositivos. 93
Tabela 5.26. Inferência sobre a diferença entre as medias para Sistema distribuído com comunicação serial *RxTx* entre dispositivos. 93

Tabela 5.27. Característica dos ensaios para Sistema distribuído com comunicação I²C entre dispositivos 94
Tabela 5.28. Teste F para o Sistema distribuído com comunicação I²C entre dispositivos. . 96
Tabela 5.29. Inferência sobre a diferença entre as medias para Sistema distribuído com comunicação I²C entre dispositivos. 97
Tabela 5.30. Custo dos dispositivos as arquiteturas de controle. 97
Tabela 5.31. Teste F para as arquiteturas de controle. 100
Tabela 5.32. Inferência sobre a diferença entre as médias para as arquiteturas de controle. 100

Lista de Abreviações

AC – Alternating Current

bps – bits por segundo

CLP – Controlador Lógico Programável

CDF – cumulative distribution function

CNC – Commando Numérico Computadorizado

CMOS – Complementary Metal Oxide Semiconductor

DDE – Dynamic Data Exchange

DCOM – Distributed Component Object Model

EIA – Electronic Industries Alliance

EEPROM – Electrically-Erasable Programmable Read-Only Memory

FBD – Function Block Diagram

FIFO – First-in First-out

GSM – Global System for Mobile Communications

GRAFCET – Graphe Fonctionnel de Commande, Étapes Transitions

IEC – International Electrotechnical Commission

IEEE – Institute of Electrical and Electronics Engineers

IL – Instruction List

IHM – Interface Homem-Máquina

J-B – Jarque-Bera

K-S – Kolmogorov-Smirnov

LD – ladder

MAC – Media Access Control

ms – milissegundos

MEMS – Microelectromechanical Systems

MRP/ERP – Material Requirements Planning/ Enterprise Resource Planning

MMOS - Mixed Metal Oxide Semiconductor

MISO – Master in slave out

MOSI – Master out slave in

OPC – Object Linking and Embedding for Process Control

OSI – Open Systems Interconnection

PC – Personal Computer

PWM – Pulse Width Modulation

RSSF – Rede de Sensores sem Fio

ROM – Remote Operation Management

RxTx – Receiver/Transmitter

SDCD – Sistema digital de controle distribuído

ST – Structured Text

SFC – Sequential Function Chart

SCK – Serial Clock

SDA – Serial Data

SPI – Serial Peripherical Interface

SRAM – Static Random-Access Memory

SS – Slave Select

TCP/IP – Transmission Control Protocol/Internet Protocol

WSN – Wireless Sensor Network

UART – Universal Asynchronous Receiver/Transmitter

USART – Universal Synchronous Asynchronous Receiver Transmitter

UDP – User Datagram Protocol

Sumário

CAPÍTULO I - INTRODUÇÃO 1
1.1 Objetivos 3
1.2 Justificativa 3
CAPÍTULO II - REVISÃO BIBLIOGRÁFICA 5
2.1 Níveis da automação industrial 5
2.1.1 Controle convencional 6
2.1.2 Padrão IEC 61131 7
2.2 Controle distribuído 10
2.2.1 Arquiteturas de controle de sistemas distribuídos 10
2.2.2 Aplicação de controle de sistemas distribuídos 15
2.3 Redes de sensores sem fio 17
2.3.1 Aplicações das RSSF 18
2.3.2 Arquitetura das RSSF 22
2.4 COMUNICAÇÃO 26
2.4.1 IEEE 802.15 26
2.4.2 Serial 27
2.4.3 I²C 28
2.5 MICROCONTROLADORES 30
2.5.1 Arduino 31
2.5.2 Formas de comunicação com Arduino 33
2.5.3 Conversão padrão IEC 61131 em código de Arduino 34
CAPÍTULO III - METODOLOGIA 39
CAPÍTULO IV - ESTUDO DE CASO 41
4.1 Sistema centralizado 48

4.2 Sistema distribuído com comunicação serial *RxTx* entre dispositivos 49
4.3 Sistema distribuído com comunicação I²C entre dispositivos 50
4.4 Sistema distribuído com comunicação *Zigbee* entre dispositivos 54
CAPÍTULO V - RESULTADOS E DISCUSSÃO 56
5.1 Sistema centralizado 57
5.2 Sistema distribuído com comunicação serial *RxTx* entre dispositivos 62
5.2.1 Ensaio 1 62
5.2.2 Ensaio 2 67
5.2.3 Ensaio 3 68
5.2.4 Ensaio 4 70
5.3 Sistema distribuído com comunicação I²C entre dispositivos 71
5.3.1 Ensaio 1 72
5.3.2 Ensaio 2 76
5.3.3 Ensaio 3 81
5.4 Sistema distribuído com comunicação *Zigbee* entre dispositivos 85
5.5 Comparação entre os ensaios 90
5.5.1 Sistema distribuído com comunicação serial RxTx entre dispositivos. 90
5.5.2 Sistema distribuído com comunicação I²C entre dispositivos. 94
5.5.3 Comparação entre as arquiteturas de controle 97
CAPÍTULO VI - CONCLUSÃO 101
REFERÊNCIAS BIBLIOGRÁFICAS 105
APÊNDICE A 110
APÊNDICE B 115

CAPÍTULO I

INTRODUÇÃO

De acordo com Wehrmeister et al. (2014) a automação moderna e sistemas de controle incluem dispositivos eletromecânicos controlados por sistemas embarcados complexos e de tempo real, compreendendo componentes de *hardware* e *software*. Estes sistemas possibilitam a criação de dispositivos “inteligentes” de automação que são capazes de trabalhar autonomamente e tomar decisões decentralizadas. Isto muda drasticamente a abordagem (usualmente centrada em controladores lógicos programáveis - CLPs) adotada na automação industrial.

Na arquitetura centralizada os transdutores são lidos e escritos quando necessário e de forma direta, não havendo necessidade de troca de mensagens entre dispositivos, acarretando na rapidez de uma determinada ação, caso esta esteja condicionada apenas a leitura e atuação. Para esta arquitetura o maior atraso ocorre entre um evento na entrada e a sua respectiva atuação, que é mínima devido à alta performance dos controladores. Porém atualmente, cada fornecedor tem o seu próprio controlador lógico programável, que na maioria das vezes são caros e apresentam uma quantidade variável de entradas e saídas. Para uma indústria de grande porte a utilização de um controlador deste tipo encaixa perfeitamente nos requisitos desejados. Quando pensamos em empresas de pequeno e médio porte, a utilização deste equipamento já não é mais viável principalmente pelo custo, que na maioria das vezes é elevado, e/ou por ter mais entradas e saídas que o necessário.

Por outro lado, a utilização da arquitetura distribuída implica em atrasos na transmissão de informações que podem afetar ou não a performance do sistema. Sistemas a eventos discretos de baixa criticidade, normalmente, são sistemas possíveis de se operar com uma arquitetura distribuída. Sendo assim, é dependendo da aplicação industrial que se define

a utilização de uma arquitetura distribuída ou centralizada. Isto ocorre, pois, algumas aplicações requerem sistemas de alta performance para atender os requisitos do sistema. Neste caso, a troca de dados deve ser rápida, confiável e, geralmente, determinística.

As últimas décadas levaram a um grande avanço da tecnologia envolvendo sensores, circuitos integrados, e comunicação sem fio. Fato este que culminou na criação das redes de sensores sem fio (RSSF) ou *wireless sensor network* (WSN). Este tipo de rede é aplicado a diversas áreas, tais como militar, saúde, ambiente, residencial. E estas aplicações vão desde a monitoramento de ambiente, a rastreamento de produtos, coordenação e processamento em diferentes cenários.

Os sistemas micro eletromecânicos (MEMS – *Microelectromechanical Systems*), os novos materiais de sensoriamento, e o avanço dos microprocessadores junto as RSSF têm estimulado a criação de sensores “inteligentes”. Hoje em dia, pode-se encontrar diversos sensores encapsulado em um único *chip*, controlados pela lógica inserida no circuito integrado, com capacidade de comunicação sem fio.

A principal vantagem de uma RSSF em relação a uma rede de computadores está no fato desta rede ser autônoma e ter um alto grau de cooperação para execução de tarefas. Sendo assim, percebe-se que a junção entre as ideias em torno das RSSF e as arquiteturas de controle de sistemas podem levar a uma integração de novos microcontroladores com a automação industrial.

Um ponto crucial para qualquer empresa é a produtividade, sendo assim a utilização de controladores lógicos programáveis pode ser um empecilho, pois quando há a necessidade de troca um equipamento deste tipo, a reposição é demorada e, via de regra, o custo é alto, com isso a produtividade da empresa pode cair significantemente. Outro ponto importante é o fato de que um equipamento deste porte necessite de mão de obra especializada. Uma micro e pequena empresa, que possui, em geral, uma abordagem baseada em automação pontual e sem previsão de expansão, procura não ser refém deste tipo de equipamento.

Atualmente, existe a possibilidade de se utilizar microcontroladores em automação industrial. Tais dispositivos são relativamente baratos e difundidos comercialmente. Sendo assim, uma pergunta surge: por que não utilizar microcontroladores em empresas que não têm a capacidade de adotar um CLP?

A partir disso, como projetar, implementar e programar com uma significativa quantidade de possíveis soluções com relação à arquitetura centralizada e distribuída, bem como com cada vez mais possibilidade de embarcar microcontroladores no nível de sensores e atuadores?

Por fim, outra questão é o quanto uma abordagem mais descentralizada e sem fio pode aumentar na complexidade do software a ser implementado e impactar no tempo de resposta do sistema?

1.1 Objetivos

Esta dissertação tem como objetivo avaliar o impacto de projetos de redes de dispositivos centralizados e distribuídos aplicados em sistemas a evento discreto de baixa criticidade e baixo custo. Para atingir este objetivo é necessário definir os objetivos específicos do projeto. Portanto estes objetivos são divididos em:

- ✓ Estudar tipos de redes centralizadas e distribuídas;
- ✓ Definir parâmetros de referência para sistemas centralizados e distribuídos;
- ✓ Avaliar desempenho de redes distribuídas;
- ✓ Análise dos resultados encontrados quanto a *hardware/software* e desempenho.

1.2 Justificativa

O projeto justifica-se devido à miniaturização e redução de custo dos componentes eletrônicos dos sensores, atuadores e microcontroladores que favorecem a utilização destes circuitos de forma mais direta e próxima dos sistemas físicos nas aplicações industriais e cotidianas. A internet das coisas é uma revolução tecnológica em relação à computação e à comunicação e a proliferação do uso de microcontroladores nos mais diversos setores e aplicações fazem com o projeto esteja cada vez mais âmbito da engenharia.

Uma RSSF reduz o cabeamento para controle de processo, o que em uma indústria pode ser a solução de um problema devido a alguns ambientes serem impossibilitados da passagem de cabos. Outro ponto é o fato da redução do custo deste cabeamento, seja na aquisição, seja na montagem. Além disso as RSSF aumentam a colaboração entre equipamentos, descentralizando uma parte da lógica que atualmente é implantada em um CLP. Sistemas de controle têm ênfase centralizadora e sua descentralização ainda é um desafio.

Com este projeto procura-se fomentar a automação de micro e pequenas empresas que não possuem condições financeiras de aplicar as formas convencionais de automação.

O estudo das redes de dispositivos centralizados e distribuídos ainda é uma área pouco abordada pela mecânica, uma vez que tem sua origem na computação. Portanto, este

projeto visa avaliar o assunto de um ponto de vista mecatrônico, com o intuito de aprimorar os projetos mecânicos. É importante frisar que não é levada em conta a análise de segurança das transmissões de dados nas redes distribuídas, mas sim a análise de projeto e aplicação dessas redes.

A dissertação é dividida em seis capítulos sendo o segundo capítulo destinado à revisão bibliográfica. No capítulo III são apresentadas as atividades realizadas no projeto. Uma vez apresentada a metodologia, o quarto capítulo mostra o estudo de caso de uma bancada de separação de peças. No capítulo V os resultados e as discussões são apresentados para que o trabalho possa ser concluído no capítulo VI.

CAPÍTULO II

REVISÃO BIBLIOGRÁFICA

Para a realização deste trabalho foi necessário, em primeiro lugar, estudar a forma como é feito o controle convencional ou clássico, com vias de entender as diferenças em relação ao controle descentralizado ou distribuído. Além disso é necessário um estudo sobre as redes de sensores e atuadores, além dos padrões de comunicação em uso. Por fim, o estudo de microcontroladores é pertinente para o fechamento da pesquisa de forma a abranger todas as áreas aplicadas.

2.1 Níveis da automação industrial

A automação industrial bem como as redes de comunicações não fica restrita ao chão da fábrica. As crescentes demandas de informações e variáveis complexas dentro de um processo industrial aumentam a importância da automação de processos no setor fabril. A Fig. 2.1 apresenta a pirâmide da automação, dividida em cinco níveis desde os dispositivos de campo até o gerenciamento corporativo.

O primeiro nível é composto pelos sensores e atuadores e encontra-se no chão de fábrica. Há várias redes no primeiro nível, tais como, *Profibus* DP (não limitada apenas a este nível) e PA, *Fieldbus*, *CAN*, *HART*, entre outros. Este nível comunica-se com a camada de controle, composta pelos CLP's, *CNC'c, SDCD's* e *PC's*. Neste nível são executadas as ações de controle de uma planta industrial. As redes de comunicação desta camada são *Controlnet*, *Ethernet IP*, *OPC*, *Modbus* e *Profinet*.

A terceira camada ou nível é composta pelos sistemas de supervisão e otimização dos processos (*Workstation*, *PC* e IHM). As redes de comunicação deste nível são: *Ethernet TCP/IP, OPC, DDE* e *DCOM*. A partir do quarto nível é possível a programação e planejamento de produção e este nível utiliza os padrões *Ethernet TCP/IP*. O último nível é chamado de gerenciamento corporativo e integra uma centralização das informações. Os softwares de gestão de vendas e financeiro são utilizados nessa camada.

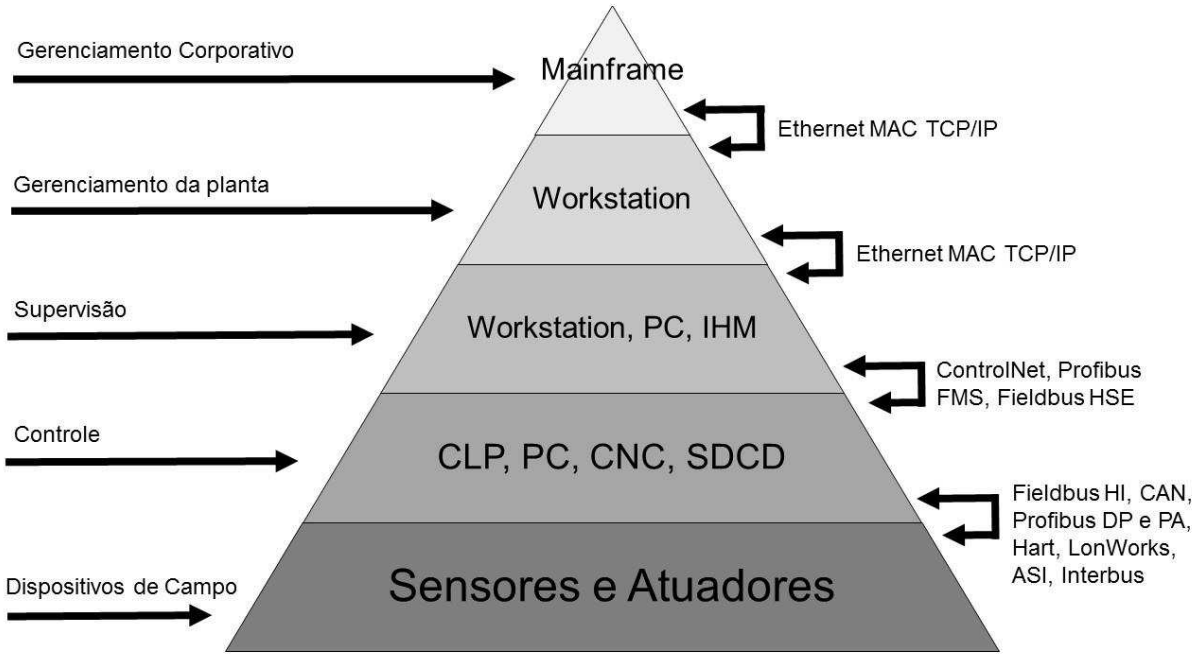


Figura 2.1. Pirâmide da Automação Industrial.

2.1.1 Controle convencional

De acordo com Petruzella (1996) um controlador lógico programável (CLP) é um dispositivo designado a realizar funções lógicas pré-definidas por meio de relés eletromecânicos. Este dispositivo surgiu em 1969 devido à grande dificuldade de modificar a lógica dos painéis de controle para cada mudança na linha de montagem. Já para Bolton (2015) um CLP é uma forma especial de um controlador microprocessado que utiliza memória programável para armazenar instruções e implementar funções tais como lógica, sequenciamento, temporização, contagem e operações aritméticas a fim de se controlar máquinas e processos.

Um CLP foi desenvolvido para ser operado por engenheiros ou pessoas com um certo nível limitado de conhecimento de computação e linguagens de programação, ou seja, um programador de CLP não precisa ser necessariamente um programador de computadores.

Sendo assim, o CLP foi programado de forma que o programa de controle fosse simples e em uma linguagem intuitiva.

Os CLPs trazem grandes vantagens uma vez que pode ser utilizado em uma diversidade de sistemas de controle. Para modificar um sistema de controle e as regras utilizadas neste sistema, é necessário apenas que o operador insira o novo conjunto de instruções, sem a necessidade de refazer as conexões físicas. Isto resulta em um sistema flexível capaz de ser utilizado em sistemas de controle que variam largamente em sua natureza e complexidade.

Quando comparados aos sistemas de relés, os CLPs implementam facilmente as mudanças, uma vez que essas são realizadas em *software*; podem ser expandidos adicionando novos módulos de CLP; são mais robustos e confiáveis que os sistemas de relés; são mais compactos; não requerem constante manutenção e podem operar em uma velocidade maior.

Atualmente, de acordo com Bakule e Papík (2012) a teoria de controle moderno e clássico consideram uma única planta com um único controlador. Todos processos computacionais podem ser baseados no conjunto de informações da planta. Sendo assim o controlador recebe todos os dados dos sensores e gera todos os sinais de saída para a planta.

Com isso os CLPs são considerados sistemas centralizados, uma vez que toda a lógica de um sistema é concentrada em um único controlador. Uma das desvantagens deste sistema é o fato de que todas as entradas e saídas estão acopladas a este controlador, com isso os custos relacionados ao cabeamento de toda a planta, quando há dispositivos espalhados longinquamente, é muito elevado.

2.1.2 Padrão IEC 61131

O *IEC 61131* é um padrão criado pela *International Eletromechanical Commission* (*IEC*) para controladores programáveis. O padrão é composto por nove partes, tal como mostra a Tab. 2.1. A primeira parte é composta pelas informações gerais do padrão, enquanto que a segunda parte apresenta os requisitos de *hardware.* A terceira parte do padrão *IEC 61131* é a primeira tentativa de padronizar as linguagens de programação para automação industrial (PLCopen, 2015). Foi criada com o intuito de evitar a proliferação de diferentes métodos de programação de dispositivos eletrônicos industriais.

Nesta parte do padrão são definidas quatro linguagens de programação, sendo duas delas textuais – lista de instruções (IL) e texto estruturado (ST); e duas versões gráficas – diagrama *ladder* (LD) e diagrama de bloco de funções (FBD). De acordo com PLCOpen (2015) a escolha da linguagem de programação depende do conhecimento do programador, do

problema em questão, do nível de detalhamento do problema, da estrutura de controle do sistema e da interface com outras pessoas ou departamentos.

A quarta parte do padrão é um guia de orientação ao usuário. A quinta e a sexta parte são responsáveis pela comunicação, sendo a última via *Fieldbus*. A sétima parte apresenta a programação utilizando lógica *Fuzzy*. A penúltima parte é um guia para implementação das linguagens e, por fim, a nona parte é apresenta uma interface de comunicação para pequenos sensores e atuadores.

Tabela 2.1. Divisões da família *IEC 61131*

Parte	**Assunto**
1	Visão Geral
2	Requisitos e testes de equipamentos
3	Linguagens de programação
4	Orientações a usuários
5	Comunicação
6	Segurança funcional
7	Programação da Lógica controle Fuzzy
8	Orientações de aplicação
9	Interface digital de comunicação para pequenos sensores e atuadores

A linguagem LD surgiu nos Estados Unidos e é baseada na representação gráfica de lógica *ladder* de relés. Por outro lado, a linguagem IL é de origem europeia e por se tratar de uma linguagem textual, assemelha-se à linguagem *assembly.* A linguagem de FBD é muito comum em processos industriais já que expressa o comportamento entre funções, blocos de funções e programas como um conjunto de blocos gráficos interconectados, tal como diagramas de circuitos eletrônicos. Já a linguagem ST é uma linguagem de alto nível derivada de Ada, Pascoal e “C”. Esta linguagem contém os elementos essenciais de uma linguagem de programação moderna, incluindo rotinas de iteração (*for*, *while* e *repeat*) e rotinas de seleção (*if* – *then* – *else* e *switch case*).

A Fig. 2.2 apresenta uma parte de um programa implementado por todas as quatro linguagens. Dada uma saída Q1 acionada por expressão lógica expressa na Eq. (1):

$$Q1 = (I0 + I1).\overline{I2} \tag{1}$$

Note na Fig. 2.2 que (a) e (b) correspondem as linguagens textuais IL e ST, respectivamente, enquanto que (c) e (d) correspondem as linguagens gráficas FBD e LD.

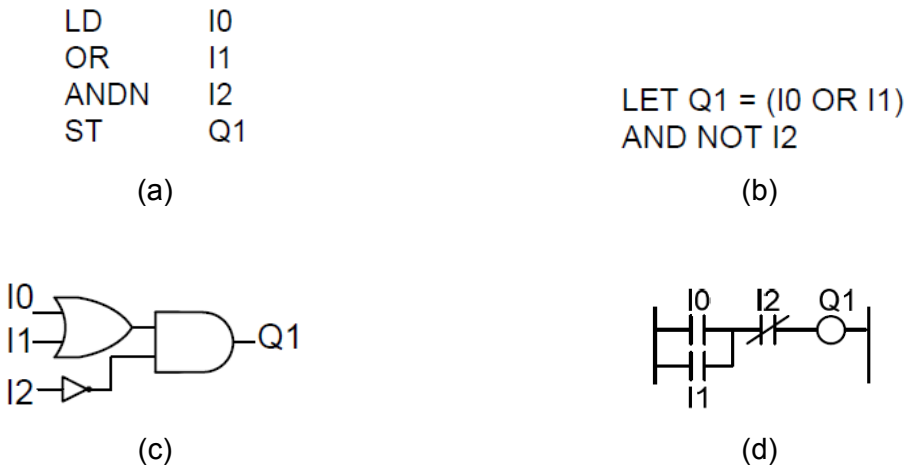


Figura 2.2. Parte de um programa implementado por todas as linguagens do padrão IEC 61131-3. Adaptado de (PLCOpen, 2015)

Em sequência, a norma *IEC 61131-3* também descreve um modelo de representação denominado diagrama de função sequencial (SFC). Esta representação descreve graficamente o comportamento do programa de controle. O SFC é derivado das redes de Petri e do padrão *IEC* 848 GRAFCET (*Graphe Fonctionnel de Commande, Étapes Transitions*) com as mudanças necessárias para converter a representação da documentação padrão em um conjunto de execuções dos elementos de controle. Um diagrama de função sequencial é composto de etapas, transições e ações. Cada etapa representa um estado particular do sistema a ser controlado. A transição é associada à condição que, quando verdadeira, desativa a etapa anterior a transição e ativa a próxima etapa. As etapas estão relacionadas as ações que realizam alguma ação de controle. A Fig. 2.3 mostra a mesma expressão lógica da Eq. (1) modelada utilizando um SFC composto de duas etapas (E0 e E1), uma ação (Q1) e uma transição ($(I0 + I1).\overline{I2}$).

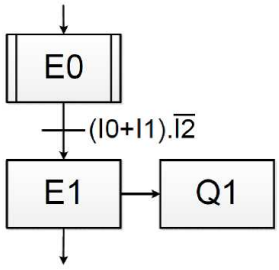


Figura 2.3. Exemplo de SFC aplicado na Eq. 1.

Um SFC não é restrito a uma sequência unilateral, admitindo sequencias alternativas e até mesmo sequencias paralelas. Qualquer sistema automático pode ser programado em qualquer uma das linguagens *IEC*, incluindo o próprio SFC. Entretanto alguns modelos de

CLPs não utilizam o SFC como uma linguagem de programação. Para este tipo de controlador, o SFC pode ser indiretamente implementado utilizando algumas metodologias de transcrição de SFC ou rede de Petri para algumas linhas de diagrama *ladder* (UZAM, JONES e AJLOUNI, 1996) (JIMENEZ, ERNESTO e LOPES, 2001) (THAPA, DANGOL e WANG, 2005), (SUESUT et al., 2004), (SILVEIRA e SANTOS, 2005).

2.2 Controle distribuído

Em um sistema de grande porte nem um modelo completo (informação a priori), nem um completo conjunto de dados mensurados (informação a posteriori) são disponíveis para um controlador centralizado (tomada de decisões) (Bakule e Papík, 2012). Para isso deve-se pensar em uma forma de distribuição das decisões tomadas referentes a um sistema.

Segundo Bakule (2008) a descentralização de um processo refere-se ao modelo e aos objetivos desejados. Ela permite uma implementação completamente independente das soluções de processos separados. A descentralização de um processo é dada pelo fraco acoplamento entre subsistemas, ou por subsistemas com objetivos contraditórios, ou por subsistemas atribuídos a diferentes autoridades, ou por sistemas de larga escala. As maiores dificuldades relacionadas aos sistemas descentralizados são:

- Dimensionalidade
- Restrições da estrutura da informação
- Incertezas do modelo
- Atrasos (*delays*)

Já em 2012, Bakule e Papík afirmaram que muitos problemas reais são considerados problemas complexos de larga escala pela natureza e não por escolha. Portanto, tais sistemas não podem ser tratados como uma caixa preta. Schuppen et al. (2011) considera que um sistema de grande porte pode ser decomposto em diversos sistemas menores e com menor complexidade. Para eles os sistemas distribuídos são definidos pela interconexão de dois ou mais subsistemas.

2.2.1 Arquiteturas de controle de sistemas distribuídos

Em um sistema descentralizado há dois ou mais controladores que são conectados ao sistema como mostra a Fig.2.4. Este tipo de arquitetura ocorre, por exemplo, em empresas com automação pontual ao longo do processo, mas sem a integração entre os controladores.

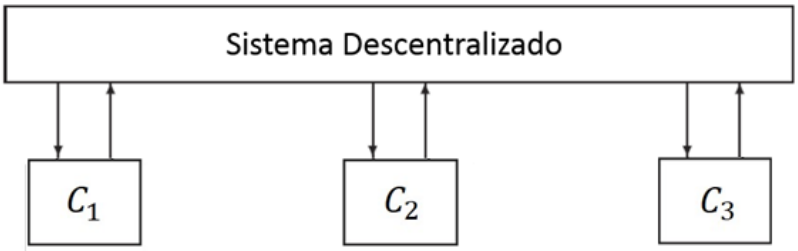


Figura 2.4. Sistema Descentralizado. Adaptado de (Schuppen et al., 2011)

A partir do momento em que definimos um sistema como sendo um conjunto de subsistemas interconectados passamos a classificá-lo como um sistema distribuído (Fig. 2.5). Este tipo de arquitetura é presente em sistemas como por exemplo, uma empresa que possui células flexíveis de manufatura conectadas a um controlador, mais um sistema de controle de estoque do tipo MRP/ERP conectado a outro controlador e um CLP utilizado em uma aplicação pontual.

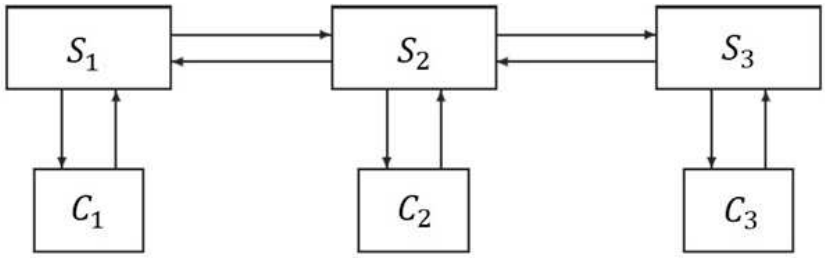


Figura 2.5. Sistema distribuído. Retirado de (Schuppen et al., 2011)

A Fig. 2.6 mostra o exemplo de um sistema distribuído, que trabalha perfeitamente em uma arquitetura centralizada. O controlador é responsável por receber os sinais dos sensores A e B, além de transmitir um sinal para o atuador C. A lógica é proposta de forma que se o sinal do sensor A for maior que o sinal do sensor B, o atuador C deve ser acionado. Caso este problema seja aplicado a uma arquitetura distribuída podem surgir alguns problemas. Flammini et al. (2008) realçou que os sensores A e B podem ser amostrados em instantes distintos, ou seja, $A = A(t_0)$ e $B = B(t_1)$ e só depois podem ser comparados. Mesmo supondo $t_0 \approx t_1$, pode ser difícil estimar exatamente os tempos de transmissão de A e B para o controlador. O quanto antes as mensagens dos sensores chegam no controlador deve transmitir o sinal para o atuador C. Sendo assim temos a Eq. (2):

$$t_d = \max(t_{dA} + t_{dB}) + t_{exec} + t_{dC} \quad (2)$$

Onde, t_d é o tempo desde o envio das mensagens pelos sensores até o recebimento da mensagem pelo atuador; t_{dA} e t_{dB} são os tempos de troca das mensagens dos sensores A e B, respectivamente, até o controlador; t_{exec} é o tempo de execução da lógica no controlador e t_{dC} é o tempo de troca da mensagem do controlador até o atuador C. Sendo assim, dependendo da velocidade de tráfego da rede e da quantidade de dispositivos utilizando a rede, t_d é significativo e variável.

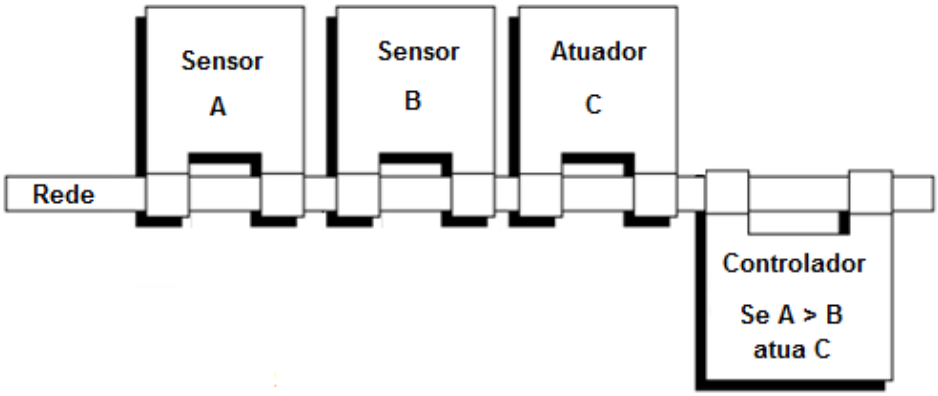


Figura 2.6. Exemplo de sistema distribuído. Adaptado de (Flammini et al., 2008)

Com este exemplo fica claro que a aplicação de um sistema distribuído é mais viável para problemas de baixa criticidade, ou seja, o tempo de resposta do sistema não deve ser da ordem de grandeza do tempo de troca de mensagens entre os dispositivos.

Outra colocação importante realçada por Flammini et al. (2008) é tal que o fator chave de uma aplicação industrial é a robustez do sistema devido às fortes interferências de sinais eletromagnéticos reduzirem a qualidade de transmissão ocasionando erros. Sendo assim a capacidade dos sensores em transferir informação em um tempo pequeno, fixo e conhecido é um ponto crucial.

Um sistema distribuído pode ser distinguido de acordo com a arquitetura de controle, sendo classificado de acordo com o grau de coordenação e o grau de hierarquia dos subsistemas. Segundo Schuppen et al. (2011) há quatro diferentes arquiteturas de controle para sistemas distribuídos: (1) controle distribuído; (2) controle distribuído com comunicação entre controladores; (3) controle coordenado e (4) controle hierárquico. As quatro arquiteturas estão em ordem sendo que (1) é a mais descentralizada e (4) a arquitetura que possui o maior grau de centralização. Tais arquiteturas são discutidas a seguir.

Controle distribuído

Nesta arquitetura cada controlador recebe as observações diretamente do sistema ou subsistema e produz uma entrada no sistema distribuído. A Fig. 2.7 mostra um esquema de sistema de controle distribuído. Note que não há comunicação entre os controladores C_1, C_2 e C_3, apenas os subsistemas S_1, S_2 e S_3 são interconectados.

Um exemplo deste tipo de controle é um sistema em que uma prensa hidráulica e uma máquina de corte trabalham cada uma com um controlador que não se comunicam entre si. Este tipo de problema é considerado um sistema semi automatizado uma vez que o setup dos dois sistemas é feito manualmente, sendo que a saída de um subsistema é a entrada do outro subsistema e vice-versa.

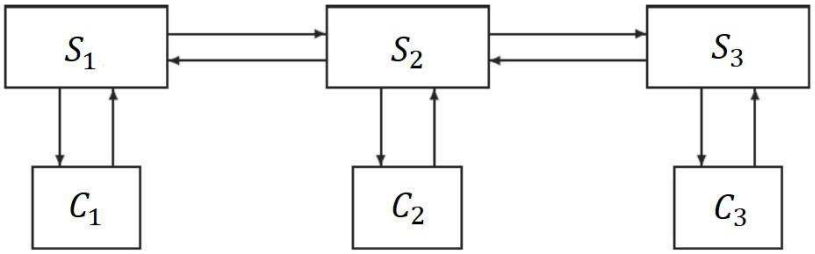


Figura 2.7. Controle distribuído. Retirado de (Schuppen et al., 2011)

Outra exemplo de sistema de controle distribuído são as cadeias de suprimento, em que cada subsistema é composto por uma empresa com um controlador que pode ser o mesmo das outras empresas ou não. Neste caso a troca de mensagens entre as empresas não é feita de forma automática.

Controle distribuído com comunicação entre controladores

Nesta arquitetura cada controlador recebe uma observação do sistema ou subsistema, mas também recebe uma ou mais observações de outro controlador. Note, pelo esquema da Fig. 2.8, que os controladores estão conectados entre si, possibilitando a troca de informações entre os próprios controladores. Neste caso, os controladores C_1 e C_3 possuem comunicação indireta, porém há casos em que esta comunicação é realizada de forma direta entre todos controladores. Um exemplo desta arquitetura de controle são as células flexíveis com comunicação entre si em uma fábrica automatizada.

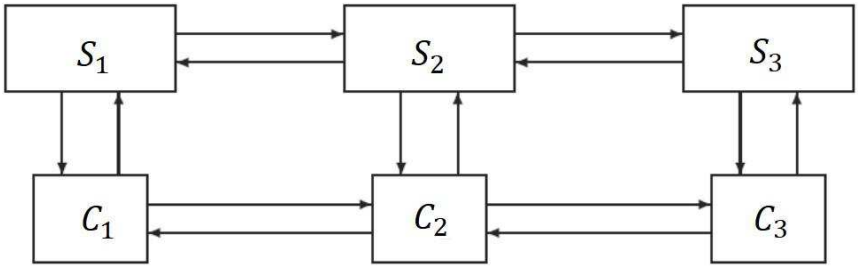


Figura 2.8. Controle distribuído com comunicação entre controladores. Retirado de (Schuppen et al., 2011)

As informações podem ser trocadas de três formas distintas: (a) o controlador envia um conjunto de suas observações; (b) o controlador envia a sua última observação quando presume que esta seja importante para outro controlador; e (c) o controlador requer informação de outro controlador se ele presumir que o controlador tenha uma informação importante que possa utilizar.

Controle coordenado

Na arquitetura de controle coordenado existe um controlador para o coordenador e um controlador para cada um dos subsistemas. A comunicação neste tipo de arquitetura é o que distingue das formulações do controle distribuído com ou sem comunicação. Este tipo de controle é utilizado em casos em que é necessário impor um certo grau de controle centralizado. A Fig. 2.9 mostra um sistema de controle coordenado.

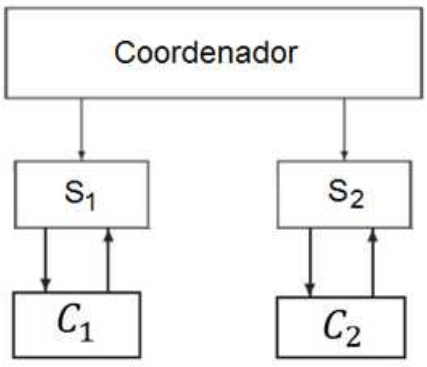


Figura 2.9. Sistema de controle coordenado. Adaptado de (Schuppen et al., 2011)

Há apenas um coordenador em arquitetura de controle coordenado, enquanto que quando há a necessidade de se utilizar mais de um coordenador o controle passa a ser um sistema de controle hierárquico.

Controle hierárquico

O controle hierárquico é uma generalização da arquitetura de controle coordenado e sua utilização depende da quantidade de coordenadores que serão necessários no sistema.

Um sistema de controle hierárquico consiste em uma ou mais camadas em que cada subsistema de uma camada é relacionado a um subsistema da próxima camada acima. Neste tipo de arquitetura há um controlador para cada subsistema que interage com outro controlador como em um sistema hierárquico. Sistemas de controle hierárquico a evento discreto, na maioria dos casos, concernem a um subsistema por nível.

2.2.2 Aplicação de controle de sistemas distribuídos

Atualmente, um exemplo de utilização das arquiteturas de sistemas distribuídos na indústria se dá pelo gerenciamento de ativos de automação e recursos geograficamente dispersos proposto pela FIELDBUS FOUNDATION (2013), o ROM (*Remote Operations Management*). Este segmento possui alto grau de personalização e soluções que não são facilmente configuradas.

O monitoramento de locais e equipamentos remotos é essencial para eficiência, saúde dos equipamentos e segurança. Uma capacidade de operação remota efetiva diminui tempo de viagem em campo e custos operacionais. (FIELDBUS FOUNDATION, 2013)

A utilização do ROM se dá, por exemplo, na indústria de óleo e gás onde a planta é geograficamente dispersa, os recursos são encontrados em locais de difícil acesso ou em áreas de risco, há unidades de produção subaquática e há a necessidade de transporte de recursos por grandes distâncias. A Fig. 2.10 ilustra um campo de extração de petróleo. Note que a utilização de um sistema centralizado para esta planta é inviável pois há plataformas de produção, campos com tanques, entre outros.

A Comunidade europeia financia um projeto denominado *Control for Coordination of Distributed Systems* (*C4C Project*) que tem como objetivo desenvolver o controle de sistemas distribuídos em cinco diferentes estudos de caso com teoria de controle, redes de comunicação e computação. Os estudos de caso são: coordenação de veículos subaquáticos, coordenação de veículos aéreos, coordenação de veículos guiados automatizados, coordenação de máquinas complexas distribuídas e sistema de controle e comando hierárquico em redes de rodovias. Os projetos estão disponíveis no site: http://www.c4c-project.eu e a Tab. 2.2 resume as universidades envolvidas, a área que cada uma desenvolve, as arquiteturas de controle e se há ou não incentivo privado nas pesquisas. Note que as duas pesquisas envolvidas pelas Universidade de Eindhoven, na Holanda, possuem incentivo de

empresas privadas. O projeto de veículos guiados automatizados é financiado por uma empresa que opera no porto de Antuérpia, na Bélgica, e o projeto de máquinas complexas distribuídas é financiado pela *Océ Technologies B.V*. que produz impressoras de alta velocidade em Venlo, na Holanda.

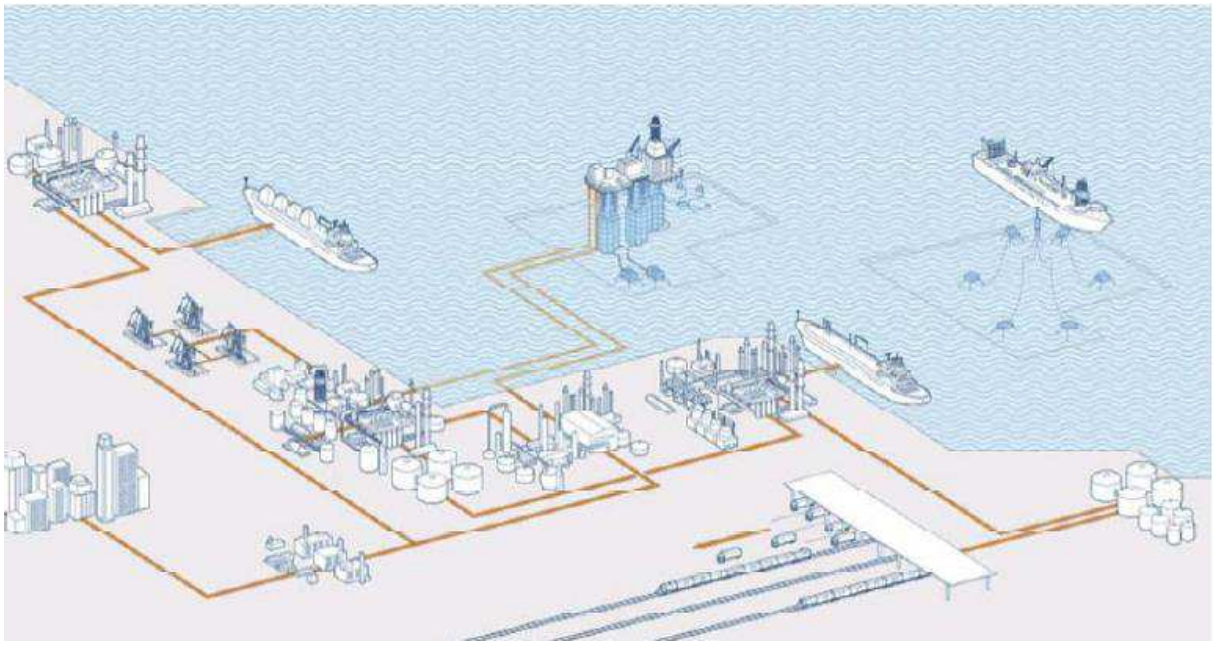


Figura 2.10. Indústria de petróleo e gás. (FIELDBUS FOUNDATION, 2013)

Tabela 2.2. Detalhes dos projetos do *C4C Projet.* Disponível em http://www.c4c-project.eu

País	**Universidade**	**Área**	**Arquitetura de controle**	**Incentivo Privado**
Portugal	University of Porto	Veículos subaquáticos	Distribuído com comunicação, coordenado e hierárquico	Não
Chipre	University of Cyprus	Veículo aéreo	Distribuído com comunicação	Não
Holanda	Delft University of Technology	Rede de rodovias	Hierárquico	Não
	Eindhoven University of Technology	Veículos guiados automatizados	Distribuído com comunicação e hierárquico	Sim
		Máquinas complexas	Coordenado e hierárquico	Sim

2.3 Redes de sensores sem fio

Segundo Akyildiz et al. (2002) as redes de sensores apresentam uma melhora em relação aos sensores tradicionais e são desenvolvidas de duas formas:

• Sensores posicionados longe do fenômeno através da utilização de técnicas complexas para distinguir o alvo do “barulho do ambiente”.

• Vários sensores apenas captando sinal através da transmissão de tempos em tempos para uma central de nós onde é feita a uma performance computacional para a fusão dos dados.

O desenvolvimento de redes de sensores requer tecnologia de três diferentes áreas: instrumentação, comunicação e computação (*hardware*, *software* e algoritmo). (CHONG; KUMAR, 2003)

As RSSF são normalmente compostas por um grande número de nós sensores inseridos ou bem próximo do fenômeno a ser estudado. Não necessariamente deve-se estudar o local certo de inserção dos nós sensores, porém os algoritmos e protocolos da rede devem possuir capacidade de se auto organizar. A Fig. 2.11 ilustra a ideia de uma RSSF, em que cada “nuvem” é formada por um conjunto de sensores. Quando um sensor necessita enviar um dado é criado um caminho até um *sink* que pode ser um sensor qualquer ou um concentrador de dados. Este *sink* pode estar conectado à Internet ou a um Satélite e faz uma ponte entre a rede e um usuário, ou gerenciador de tarefas.

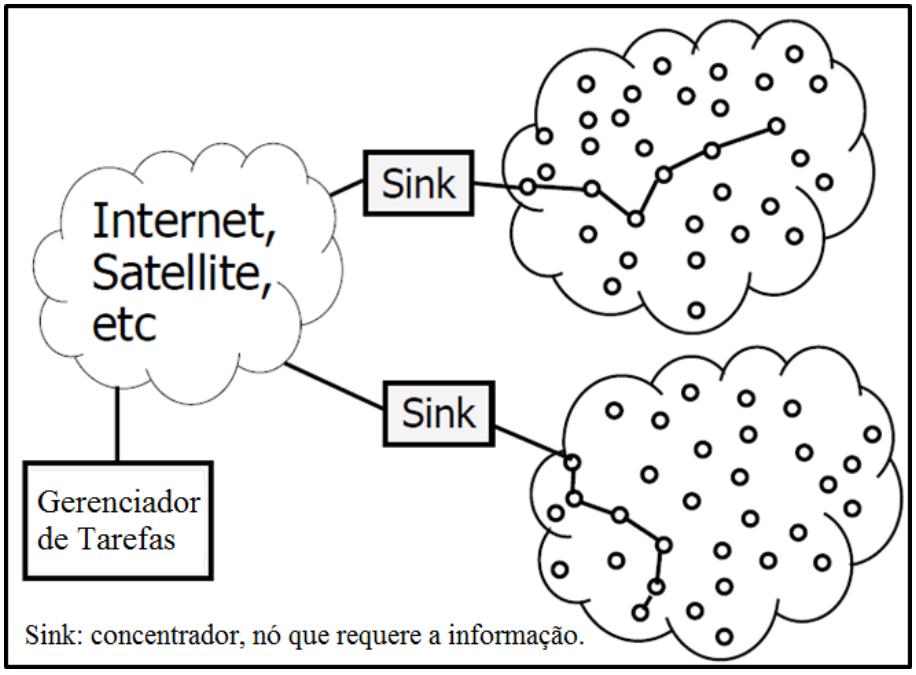


Figura 2.11. Rede de sensores sem fio. Adaptado de (AKYILDIZ et al., 2002)

2.3.1 Aplicações das RSSF

Militar

O principal exemplo de aplicação das RSSF na área militar é o sistema de defesa do Departamento de Defesa dos Estados Unidos conhecido até 2003 como *Command, Control, Communications, Computing, Intelligence, Surveillance, Reconnaissance and Targeting (C4ISRT) systems.*

Este sistema é capaz de monitorar forças amigas, equipamentos e munição; vigilância de campos de batalha; reconhecimento de forças e campos inimigos; avaliação de danos de batalha; detecção e reconhecimento de ataques nucleares, biológico ou químico.

Saúde

A área da saúde é uma área de aplicação das RSSF em que há grande investimento. Isto deve-se ao fato da busca de uma maior qualidade de vida pela sociedade. As aplicações vão desde rastreamento e monitoramento de médicos e pacientes dentro de um hospital a monitoramento de sinais vitais dos pacientes em ambiente externo.

Alemdar e Ersoy (2010) apresentam uma rede para auxiliar no monitoramento de pacientes em suas residências. A rede é dividida em três classes principais. A rede próxima ao corpo do paciente, composta por vários sensores, que avaliam frequência cardíaca, pressão sanguínea, entre outros sinais vitais. Essa rede é denominada “Rede do Corpo”. Há uma classe composta pelos cuidadores ou enfermeiros e os médicos ou psicólogos do paciente chamada de “Aplicação”. Outra classe engloba a rede pessoal do paciente que é composta pelos *smartphones* dos pacientes e câmeras espalhadas pela residência, a “Rede Pessoal”. As classes se comunicam via rede tanto ‘AdHoc’ quanto GSM ou Ethernet. A Fig. 2.12 ilustra essa rede apresentada.

Outro exemplo de RSSF aplicado à área da saúde é o trabalho apresentado por Iso-Ketola et al. (2008). O trabalho introduz um sistema para recuperação de operação de quadril. Este sistema é composto por sete nós de uma rede de sensores sem fio para medir uma postura inequívoca do quadril operado. Há também uma palmilha com um nó sensor de carga para medir a carga colocada sobre a perna operada. Se a carga ultrapassar um valor permitido para aquele paciente é emitido um sinal para a unidade de processamento e central de controle. Esta central é capaz de recolher e processar os dados dos nós sensores. A Fig. 2.13 apresenta este sistema do tipo *wearable computing*. Percebe-se que o sistema é uma bermuda equipada de sensores, sendo que a unidade de processamento encontra-se na parte superior frontal da bermuda. A rede é capaz de se comunicar com um telefone móvel via rede Bluetooth que se comunica a Internet via GSM.

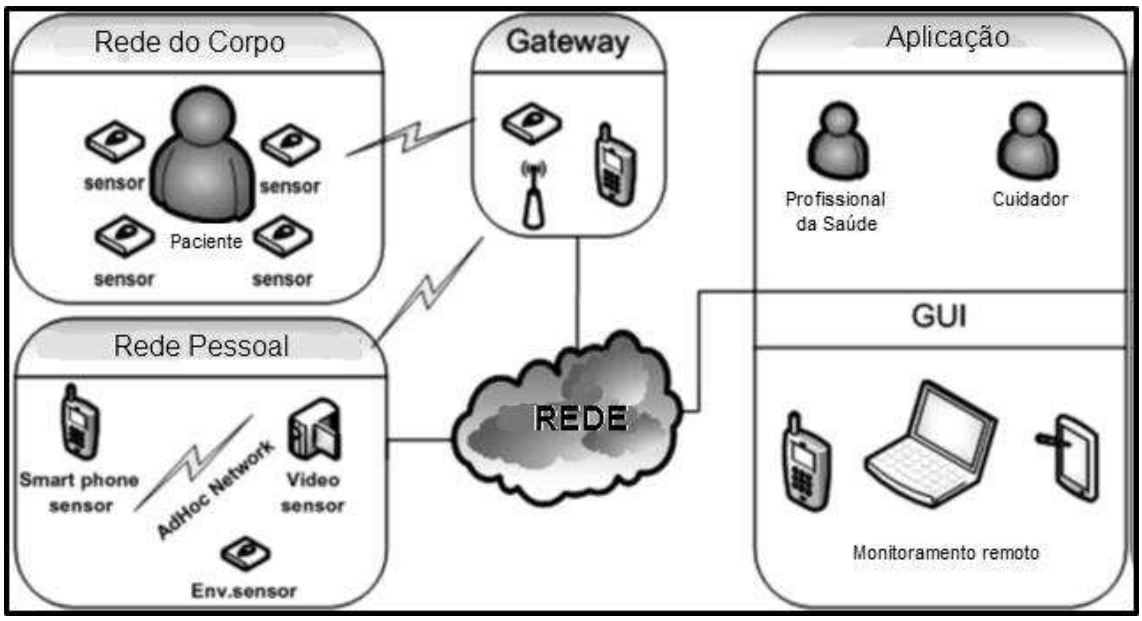


Figura 2.12. Rede para monitoramento de paciente em residências. Adaptado de (ALEMDAR & ERSOY, 2010)

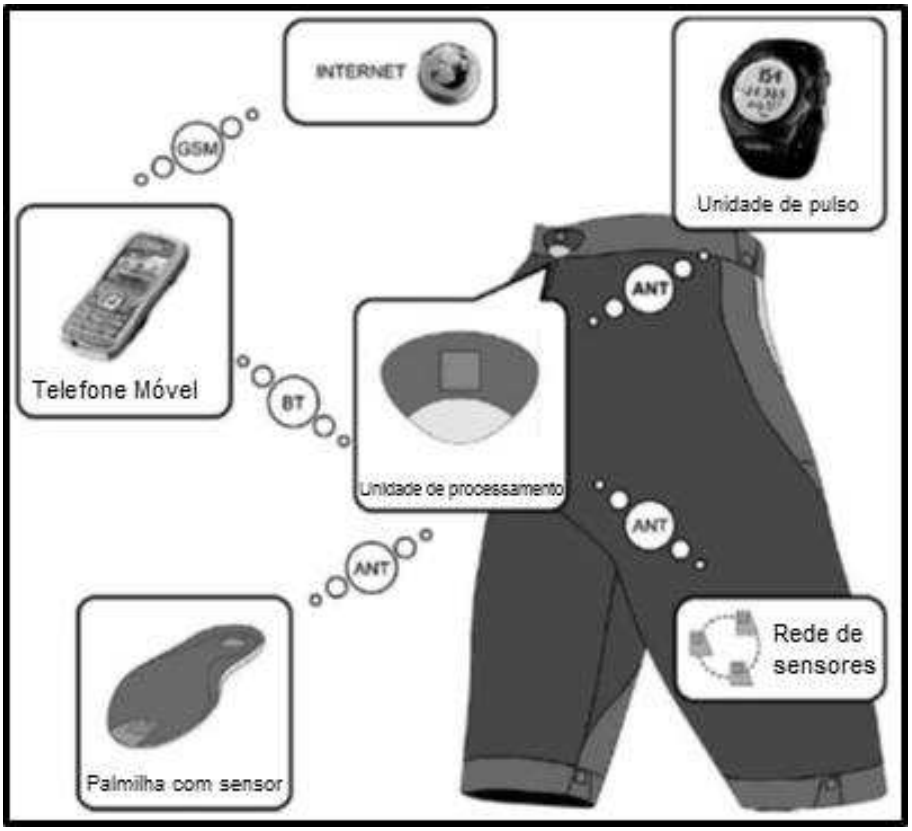


Figura 2.13. Sistema de recuperação de quadril. Adaptado de (ISO-KETOLA et al., 2008)

As redes de sensores não se aplicam necessariamente a sensoriamento de pacientes na área da saúde. Hande et al. (2007) desenvolveram um trabalho visando *harvesting* (extração) de energia para alimentação de roteadores que transmitem sinais dentro de um hospital. A rede é composta por vários roteadores que são alimentados pela energia luminosa das lâmpadas presentes nos corredores de hospitais. A utilização destas lâmpadas deve-se ao fato de que teoricamente as luzes do corredor nunca são apagadas. Com isso essa energia luminosa pode ser extraída para alimentar tais dispositivos.

Meio Ambiente

Um exemplo de aplicação das RSSF na área de meio ambiente é o trabalho proposto por Werner-Allen et al. (2005) na Universidade de Harvard, nos Estados Unidos. Este trabalho consiste em monitorar as atividades do vulcão Tungurahua no Equador. Diversos nós sensores são espalhados pela base do vulcão a fim de se ter um monitoramento das atividades vulcânicas da região. São responsáveis por essas observações sensores sismômetros capazes de traçar a velocidade sísmica na região, sensores de pressão, microfones, dentre outros. A Fig. 2.14 apresenta esse sistema.

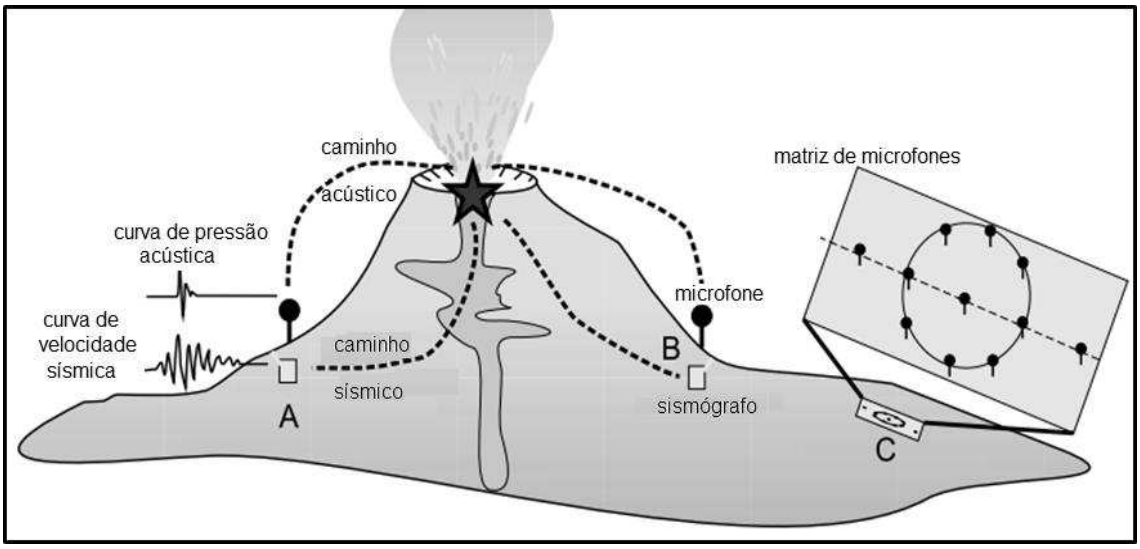


Figura 2.14. Sistema de detecção de atividade vulcânica. Adaptado de (WERNER-ALLEN et al, 2005)

Outra aplicação é apresentada por Akyildiz et al. (2002) na detecção de fogo em florestas. Neste tipo de rede, os sensores são responsáveis por monitorar regiões e se organizar para enviar um sinal dado alguma alteração no ambiente captado. Este tipo de rede necessita de um método de alimentação dos sensores cada vez mais eficiente, já que é inviável chegar a algumas regiões periodicamente para a troca de baterias. Portanto esta área

de aplicação está diretamente ligada à área de estudos sobre extração de energia do meio ambiente para consumo da rede (*harvesting*).

O *Emergency Alert System* (EAS) é um Sistema estadunidense de monitoramento de condições climáticas (disponível em http://alertsystems.org/). A rede é composta por vários sensores de chuva, nível de água, pressão capazes de captar o surgimento de um tornado ou inundação. Este sistema consegue alertar toda a população de uma região em qualquer ponto do país em poucos minutos.

Há também desenvolvimento na parte de agricultura de precisão como mostra Bo Sum e Wu (2013). Neste sistema uma rede de sensores foi instalada para captar a umidade do solo e enviar a uma central. Foi possível distinguir os sinais captados para três solos distintos: seco, úmido e encharcado de água.

Residêncial

Sensores e atuadores inteligentes podem ser embarcados em aparelhos com refrigeradores, micro-ondas, máquinas de lavar louça e roupa, ar condicionado entre outros. Esses nós sensores dentro dos dispositivos domésticos podem interagir um com o outro e com uma rede externa via *internet* ou satélite. Com isso os usuários podem manipular os dispositivos residenciais dentro da própria residência ou remotamente de forma rápida e fácil. Para ilustrar esse sistema é apresentado a Fig. 2.15 em que apresenta uma residência e alguns dos sensores (incêndio, chuva, segurança, infravermelho, entre outros) e dispositivos que podem ser interligados através da rede, tais como ar condicionado, telefone, televisores, fornos micro-ondas, aquecedor solar etc.

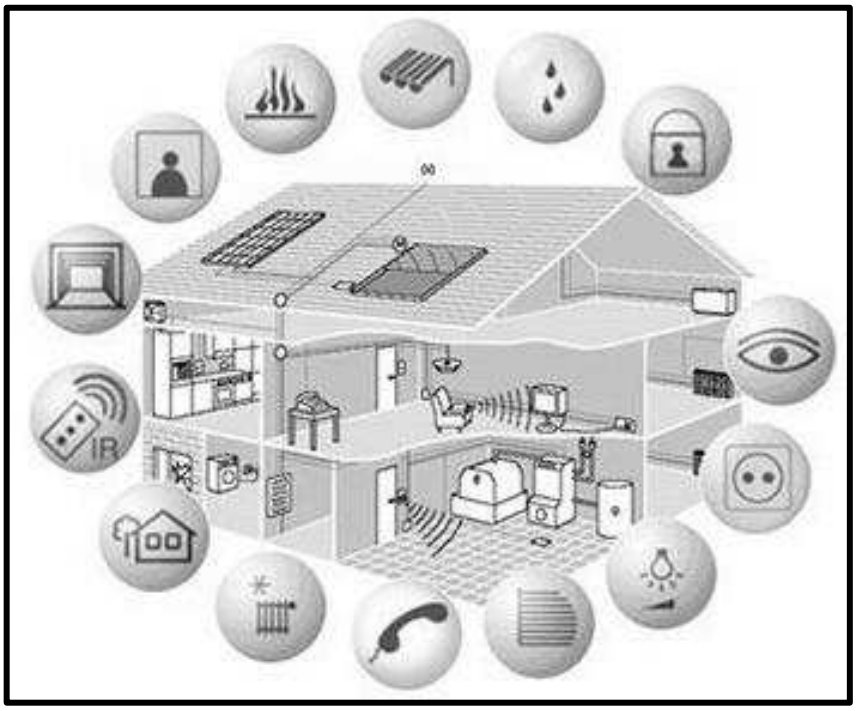


Figura 2.15. Aplicação residencial de RSSF. Retirado de http://revistahometheater.uol.com.br

Os Ambientes Inteligentes são classificados de duas formas segundo Essa (2000): os centrados no usuário e os centrados na tecnologia. No primeiro tipo o ambiente se adapta aos usuários em termos de capacidade de entrada e/ou saída. O segundo tipo diz que as tecnologias de hardware, soluções de rede e serviços de *middleware* devem ser desenvolvidos para acompanhar a evolução dos sistemas.

Outras aplicações

Outras aplicações de RSSF que não encaixam nas áreas mencionadas anteriormente são: controle de estoque e rastreamento e detecção de veículos. No controle de estoque cada item em um armazém pode ter um nó sensor anexado e os usuários finais podem descobrir a localização exata do item além da contagem do número de itens na mesma categoria. No rastreamento de veículos cada rodovia possui vários sensores espalhados de forma que um veículo que também possui um emissor de sinal pode ser encontrado pelo encontro da linha de rolamento (*Line of Bearing* - LOB) de três sensores. Esta LOB é uma linha que liga o veículo até o sensor que está o captando.

2.3.2 Arquitetura das RSSF

As arquiteturas das RSSF são capazes de integrar dados com protocolos de rede e comunicar de forma eficiente, consumindo a menor quantidade de energia possível. Estas arquiteturas promovem esforços cooperativos entre os nós sensores.

De acordo com Akyildiz et al. (2002) as redes de sensores possuem 5 camadas distintas: física, enlace, rede, transporte e aplicação, como mostra a Fig. 2.16. Dependendo das tarefas de detecção, diferentes tipos de software de aplicação podem ser construídos e utilizados na camada de aplicação. A camada de transporte ajuda a manter o fluxo de dados. A camada de rede tem cuidado de encaminhar os dados fornecidos pela camada de transporte.

Há também três planos de gerenciamento: plano de gerenciamento de energia, plano de gerenciamento de mobilidade e plano de gerenciamento de tarefas. O primeiro é capaz de desligar o receptor de mensagens de um nó imediatamente após receber uma mensagem de algum vizinho para evitar o recebimento de mensagens duplicadas. Quando a energia do sensor é baixa, também é função do plano de gerenciamento de energia enviar uma mensagem para os vizinhos informando a situação do sensor e que o mesmo não pode participar do roteamento de mensagens.

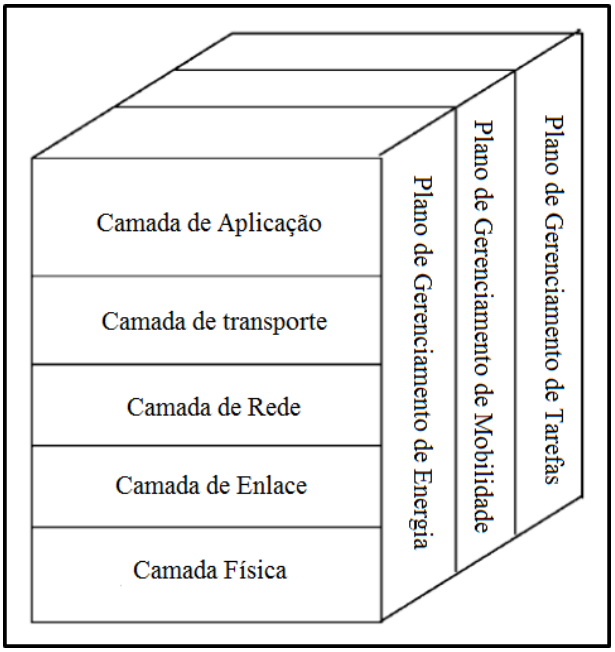


Figura 2.16. Camadas das RSSF. Adaptado de (AKYILDIZ et al., 2002)

O segundo plano é responsável por detectar e registrar a movimentação de transmissão de dados pelos nós sensores, fazendo com que uma rota de retorno de dados seja mantida até o usuário ou *sink*. Como os nós conseguem rastrear quais nós estão em sua vizinhança, este plano consegue balancear a energia e as tarefas executadas pelo nó.

Por fim o plano de gerenciamento de tarefas é responsável por balancear e agendar tarefas de sensoriamento de uma região específica. Um nó sensor com baixa energia não precisa executar uma tarefa que outro sensor na sua vizinhança está executando. Com isso

alguns nós podem realizar tarefas com mais frequência dependendo da sua capacidade de energia.

As redes de sensores podem ser desenvolvidas em diferentes topologias. Na topologia em linha cada nó comunica com apenas o nó anterior e posterior. Na topologia em barramento todos os nós estão ligados ao mesmo barramento, sendo assim apenas um nó escreve no barramento por vez.

A topologia em anel é semelhante a em linha, porém o primeiro e o último nó comunicam entre sim, fechando um anel. Nesta topologia a transmissão de dados é feita de forma unidirecional.

A topologia em estrela possui um concentrador como ponto central da rede e este retransmite todos os dados aos nós. Esta topologia torna mais fácil a localização de problemas. Na topologia em malha existe mais de um caminho para ir de um nó a outro na rede.

Na topologia em árvore há um nó central do qual saem ramos de nós. Nesta topologia a taxa de transmissão deve ser baixa. E por fim, a topologia em malha completa é uma topologia em que cada nó está conectado a todos os outros nós da rede. O tempo de espera é reduzido e eventuais problemas não interrompem o funcionamento da rede. A Fig. 2.17 apresenta estas sete topologias distintas.

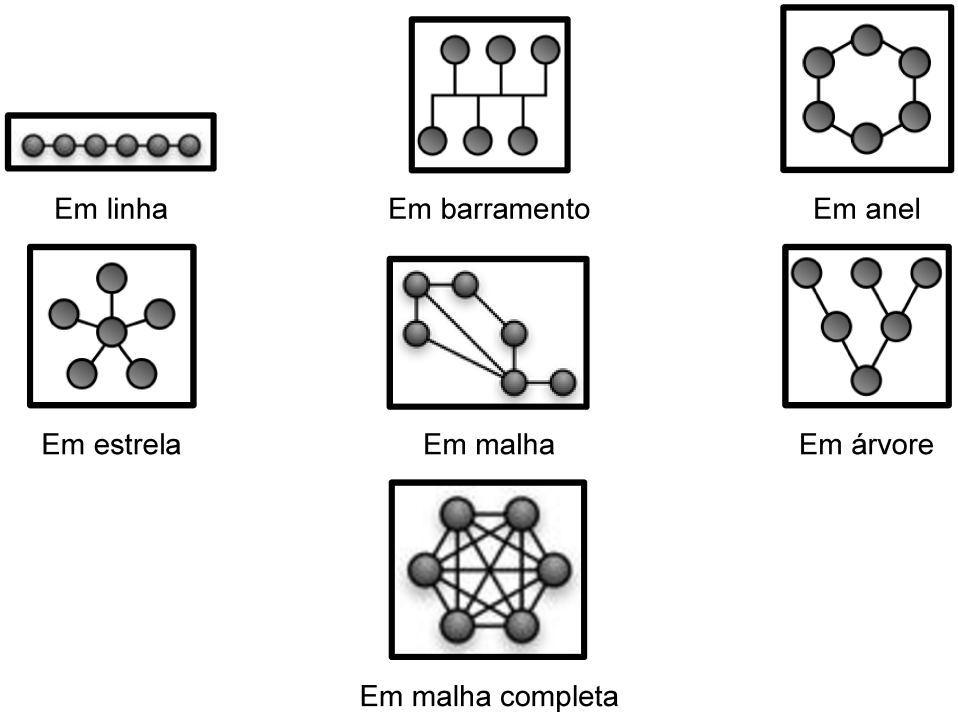


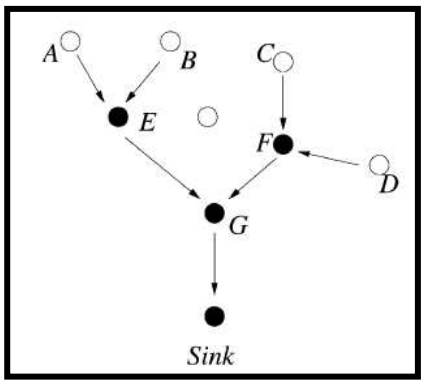
Figura 2.17. Topologias das RSSF.

Segundo Chong e Kumar (2003) apesar das redes de sensores sem fio possuírem várias aplicações diferentes, elas têm os mesmos problemas técnicos. Estes problemas podem ser referentes às condições ambientais adversas, pois os sensores podem estar sujeitos a interferências RF, ambientes altamente corrosivos, elevados níveis de humidade, vibrações, sujeira e poeira.

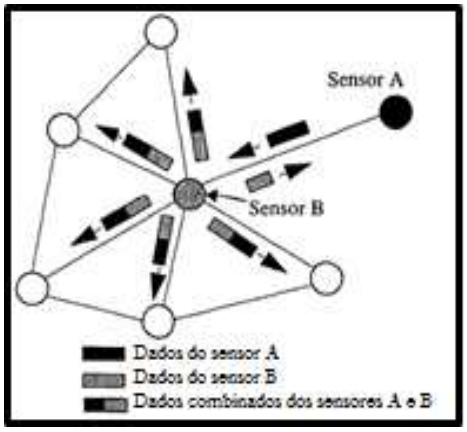
Os problemas técnicos podem estar limitados a recursos, seja por escassez de energia, falta de memória, ou falta de poder processamento.

Um destes problemas técnicos está relacionado a agregação de dados. Vários trabalhos entram nesta área. Akyildiz et al (2002) apresentaram um problema encontrado em redes de topologia do tipo árvore em que um nó sensor ao receber os dados de dois ramos deve ter a capacidade de agregar estes dados para passar a diante as informações importantes. Chandrakasan et al (1999) apresentaram um problema nas redes de topologia do tipo malha em que um sensor deve agregar um dado recebido à mensagem que deseja enviar para outros nós sensores. Akkaya e Younis (2005) apresentaram um problema encontrado na captação de sinal por meio de câmeras. Duas câmeras captam sinais que possuem uma região em comum. Portanto o nó que recebe esses sinais deve ser capaz de encontrar essa região em comum e eliminar um dos dados, já que estão duplicados. Outro problema apresentado é para uma topologia em rede em que dois caminhos podem ser seguidos e a mesma mensagem chega duas vezes a um certo nó desta rede. A Fig. 2.18 ilustra estes problemas que foram apresentados pelos autores.

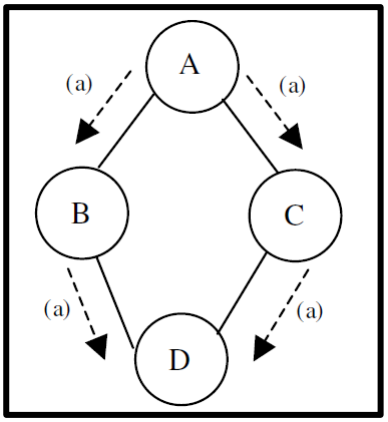
De acordo com Intanagonwiwat et al. (2002) a agregação de dados sempre é necessária quando as fontes de energia são limitadas. Sendo assim, o autor propõe uma busca gananciosa em árvore para a escolha do melhor caminho a ser seguido pelos dados durante a transmissão de um sinal de um nó a outro. Um nó na rede deve decidir (baseado em probabilidade e características do tráfego) de qual ou quais nós irá extrair dados. O *sink* tem informações empíricas de qual vizinho pode providenciar um dado com melhor qualidade, ou seja, menor perda de dados e menor *delay*.

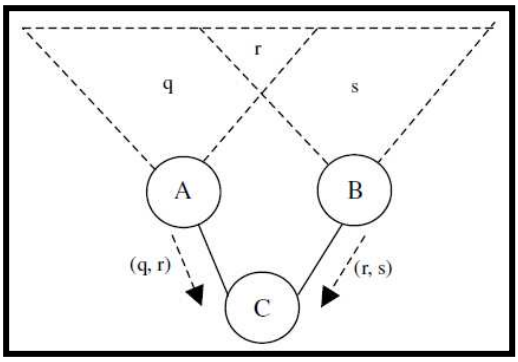


(AKYILDIZ et al., 2002)



Adaptado de (CHANDRAKASAN et al., 1999)





(AKKAYA & YOUNIS, 2005)

Figura 2.18. Problemas apresentados para agregação de dados.

2.4 COMUNICAÇÃO

Quando há mais de um dispositivo envolvido há a necessidade de comunicação entre eles. Para que uma rede funcione existem regras e formato de dados que devem ser seguidos, pois sem eles, elas simplesmente não funcionam. Este conjunto de regras e formatos de dados é denominado de protocolo. Nesta secção são apresentados três tipos de protocolos de comunicação. O primeiro deles é um protocolo de comunicação sem fio o padrão *IEEE* 802.15. O segundo é um protocolo de comunicação serial e por fim um protocolo denominado de barramento I²C.

2.4.1 IEEE 802.15

O *IEEE 802.15.4* é um padrão mantido pelo *Institute of Electrical and Eletronics Engineers* de especificação da camada física e efetua o controle de acesso para redes

pessoais sem fio de taxas de transmissão baixas. Este padrão é a base para as especificações *Zigbee*, ISA100.11a, *WirelessHART* e *MiWi*.

O enfoque deste padrão é em comunicação de baixo custo de dispositivos próximos com pequena ou nenhuma infraestrutura subjacente, com intuito de explorar ainda mais o baixo consumo de energia.

Os dispositivos podem utilizar uma de três possíveis bandas de frequência de operação (868/915/2450 MHz) e interagem através de uma rede sem fio simples. As definições das camadas de rede são baseadas no modelo OSI, apesar de o padrão definir apenas as camadas mais baixas: camada física e camada de acesso ao meio.

A camada física é a camada inicial do modelo OSI e fornece o serviço de transmissão de dados. A camada de controle de acesso ao meio (MAC) permite a transmissão de quadros pelo canal físico. Além do serviço de dados, ela controla a validação dos quadros, garante os *time slots* e manuseia as associações de nós. As camadas demais camadas não são definidas neste padrão.

2.4.2 Serial

A transmissão de dados de forma sequencial e individual dos bits é dada pelo dispositivo UART (*Universal Asynchronous Receiver/Transmitter*). A transmissão serial de informação digital (bits) através de um único cabo possui um custo menor que a transmissão paralela devido a necessidade de múltiplos cabos.

Os padrões RS-232, RS-422 e RS-485 são exemplos de padrões de comunicação serial propostos pela EIA (*Electronic Industries Alliance*) que utilizam a transmissão de sinal através de tensão. Há outras formas de transmissão de dados sem a utilização de cabos, como é o caso de infravermelho e Bluetooth.

A comunicação pode ser feita de forma *simplex* (em apenas uma direção, em que o dispositivo que recebe não envia informação de volta ao dispositivo que transmite), *full duplex* (em que ambos os dispositivos recebem e enviam ao mesmo tempo) ou *half duplex* (em que os dispositivos ora transmitem ora recebem dados.)

Todas as operações do *hardware UART* são controladas pelo sinal do *clock* que executa a um múltiplo da taxa de transmissão dos dados. Com isso o dispositivo receptor checa o estado do sinal em cada pulso do *clock.* A maioria dos dispositivos *UART* utilizam um pequeno buffer de memória *FIFO* (F*irst-In First-Out*) que auxilia na prevenção de perda de dados em taxas de transmissão alta.

2.4.3 I²C

Todas as informações contidas nesta secção estão disponíveis em: http://www.nxp.com/documents/user_manual/UM10204.pdf e foram acessadas em 26/05/2015. Este manual define o protocolo de comunicação barramento I²C.

A Philips Semicondutores (atualmente NXP Semicondutores) desenvolveu um barramento bidirecional de dois cabos chamado de barramento I²C ou Inter IC. Este é um protocolo serial utilizado para conexão entre dispositivos de baixa velocidade, tais como microcontroladores, *EEPROMs*, conversores A/D e D/A, interfaces de entrada e saída e periféricos similares em sistemas embarcados.

As informações entre os dispositivos conectados ao barramento são carregadas por dois cabos, *Serial DAta (SDA)* e *Serial CLock (SCL)*. Cada dispositivo, não importando sua natureza, é reconhecido por um endereço único e é capaz de operar tanto como um transmissor ou receptor dependendo da função do dispositivo. Os dispositivos também podem ser considerados como mestres ou escravos quando transmitem dados. A Tab. 2.3 apresenta a termologia do barramento I²C. O mestre é um dispositivo que inicia a transferência de dados no barramento e gera o sinal do *clock* para permitir essa transferência. Neste momento, qualquer dispositivo endereçado é considerado um escravo.

A relação do tipo mestre-escravo pode ser ilustrada por um exemplo bem simples. Imagine uma sala de aula com um professor e alguns alunos. Nesta sala o meio de transmissão dos dados é o ar e se os alunos e o professor tentarem se falar ao mesmo tempo ocorrerá um congestionamento neste meio, gerando apenas um ruído. Ao denotarmos o professor como o mestre os alunos então, seriam os escravos. Espera-se que nesta sala de aula todos os alunos respeitem o professor, portanto todos ficam em silêncio enquanto o professor fala, ou seja, o mestre transmite uma quantidade de dados pelo ar até os escravos que apenas captam os sinais sonoros. Se o professor resolve fazer uma pergunta direcionada a um certo aluno ele, inicialmente, irá chamar pelo nome do aluno e fará sua pergunta. O aluno ao perceber que foi requisitado a responder uma pergunta, a responde imediatamente, enquanto que os outros alunos permanecem em silêncio. Esta situação é o que ocorre em uma relação mestre-escravo.

O barramento I²C é um barramento multi-mestre, ou seja, mais de um dispositivo é capaz de controlar o barramento. Como mestres são comumente microcontroladores, a Fig. 2.19 mostra um caso em que o dado é transferido entre dois microcontroladores pelo barramento I²C. Note que a figura evidencia as relações mestre-escravo e receptor-transmissor. Supondo que o microcontrolador A (MA) deseja enviar informações para o microcontrolador B (MB). Inicialmente o mestre MA endereça o escravo MB; então MA é o transmissor envia os dados para o receptor MB. Por fim, o mestre MA termina a transferência.

Tabela 2.3. Definição da termologia do barramento I²C.

Termo	**Descrição**
Transmissor	Dispositivo que envia os dados pelo barramento
Receptor	Dispositivo que recebe os dados pelo barramento
Mestre	Dispositivo que inicia a transferência, gera o sinal do *clock* e termina a transferência
Escravo	Dispositivo endereçado pelo mestre
Multi-mestre	Mais de um mestre podem tentar controlar o barramento ao mesmo tempo sem corromper a mensagem
Arbitração	Procedimento que garante que se mais de um mestre tentam simultaneamente controlar o barramento, apenas um é permitido realizar tal ação e a mensagem não é corrompida
Sincronização	Procedimento que sincroniza os sinais de *clock* de dois ou mais dispositivos

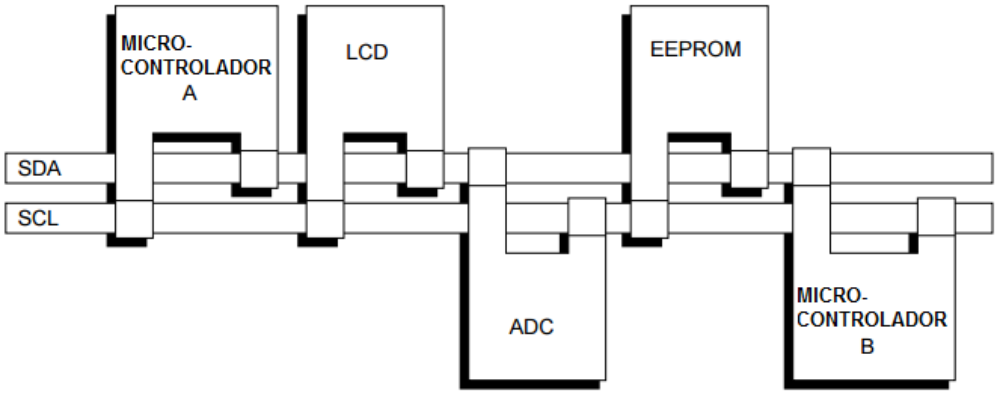


Figura 2.19. Exemplo de um barramento I²C utilizando dois microcontroladores.

Tanto SDA quanto SCL são linhas bidirecionais conectadas a uma tensão positiva por um resistor *pull-up.* Quando o barramento está livre, ambas as linhas estão em nível alto. Os dados podem ser transferidos pelo barramento I²C com uma taxa de 100 kbit/s no modo padrão, 400 kbit/s no modo rápido, 1 Mbit/s no modo mais rápido e 3,4 Mbit/s no modo de alta velocidade.

Devido à variedade de tecnologias de diferentes dispositivos, tais como CMOS, NMOS, bipolar, entre outros, que podem ser conectados ao barramento I²C, os níveis lógicos baixo (0) e alto (1) não são fixados e dependem do nível de tensão do barramento. Normalmente os níveis estão próximos de 30 % da tensão para o nível baixo e 70 % para o

nível alto. Alguns dispositivos utilizam 1,5 V para nível baixo e 3,0 V para nível alto, porém todos os dispositivos novos demandam a especificação 30 % / 70 %.

2.5 MICROCONTROLADORES

Um microcontrolador é um chip que pode ser programado para executar diversas funções específicas. Ele é composto por uma unidade de processamento conectado à memórias e periféricos de entrada e saída. O esquema básico de um microcontrolador é apresentado na Fig. 2.20. Existem duas arquiteturas para os microcontroladores: von Neumann e Harvard. Na primeira a memória de programa e a memória de dados estão em um mesmo barramento, enquanto que na segunda arquitetura os dois tipos de memória utilizam barramentos distintos.

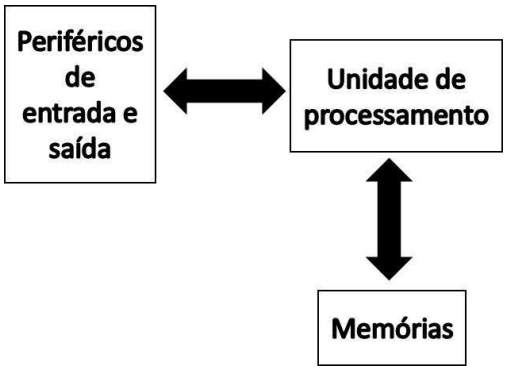


Figura 2.20. Esquema básico de um microcontrolador.

A utilização de microcontroladores como alternativas de baixo custo a equipamentos de automação tem sido bastante difundida no mundo. Há trabalhos que envolvem a utilização de microcontroladores do tipo PIC (Win e Htun, 2014) (Selvaraju et al., 2014), *Arduino* (Png et al., 2014) (Mowad et al., 2014) (Hanggoro et al., 2013) (Salinas et al., 2013), *Raspberry pi* (Ahmed et al., 2014) (Brock e Bruce, 2014) e *Beaglebone* (McPherson e Zappi, 2015). Podem ser encontrados trabalhos que utilizam mais de um microcontrolador como é o caso de Al-Sahib e Azeez (2015) que utilizam tanto *Arduino* quanto *Raspberry pi* e Travaglione et al. (2014) que utilizam *Arduino*, Rapsberry pi e *Beaglebone*. A Tab. 2.4 apresentam os trabalhos apresentados acima e suas áreas de aplicação. Observa-se que os microcontroladores podem ser utilizados em uma vasta variedade de sistemas, destacando-se processamento de áudio, energia solar e controle residencial.

2.5.1 Arduino

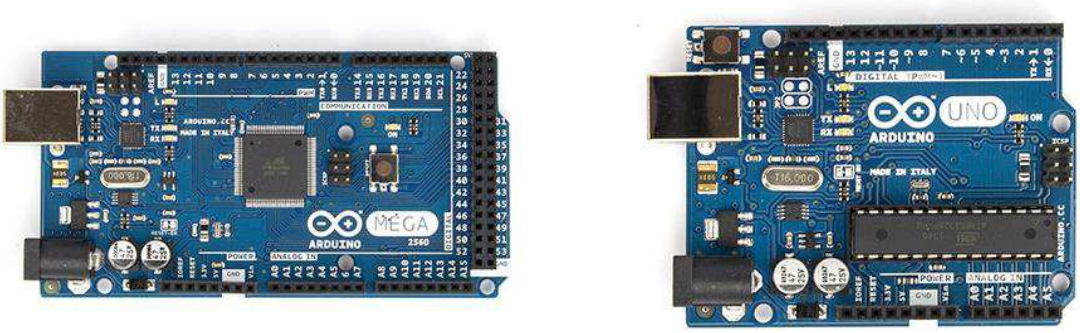
Arduino é uma plataforma eletrônica de código aberto baseada em *hardware* e *software* de fácil utilização. A plataforma tem origem em 2005 na cidade de Ivrea, na Itália e buscava interagir em projetos escolares de forma a ser mais barato que os outros sistemas de prototipagem da época. Atualmente é uma das plataformas mais difundida mundialmente e é utilizada por qualquer pessoa que se interesse em criar ambientes/projeto interativos.

Atualmente existem 21 diferentes placas de *Arduino* sendo que elas se diferenciam pela quantidade de memória, microcontrolador instalado, quantidade de entradas e saídas analógicas e digitais, quantidade de portas *PWM*, tipos de conectores de interface, entre outros. A Fig. 2.21 mostra dois *Arduino*s distintos. O da esquerda (a) é um *Arduino* Mega 2560 e o da direita (b) é um *Arduino* Uno. A Tab. 2.5 apresenta as características destas duas placas e suas principais diferenças estão no microcontrolador (ATmega328 para o Uno e ATmega2560 para o Mega), na quantidade entradas e saídas digitais (14 no Uno e 54 no Mega), na quantidade de entradas analógicas (6 no Uno e 16 no Mega) e nas memórias Flash SRAM e EEPROM (que são quatro vezes maior no Mega).

Tabela 2.4. Trabalhos que utilizam microcontroladores em diversas áreas de atuação.

Ano	**Autores**	**Área**	**Microcontrolador**
2015	McPherson e Zappi.	Ambiente de processamento de áudio e sensor	*Beaglebone*
2015	Al-Sahib e Azeez	Robô com interface móvel	*Arduino* *Raspberry pi*
2014	Travaglione et al.	Dados acústico subaquático	*Arduino* *Beaglebone* *Raspberry pi*
2014	Win e Htun	Rotacionamento de placa solar	PIC
2014	Selvaraju et al.	Sistema automático de monitoramento de carga com *Zigbee*	PIC
2014	Png et al.	Sistema de posicionamento interno	*Arduino*
2014	Mowad et al.	Sistema de controle de casa inteligente	*Arduino*
2014	Ahmed et al.	Monitoramento e controle de sistemas	*Raspberry pi*
2014	Brock e Bruce	Sensoriamento	*Raspberry pi*
2013	Salinas et al.	Captação de energia solar	*Arduino*

2013	Hanggoro et al.	Controle e monitoramento de casa verde	*Arduino*



(a) (b)

Figura 2.21. (a) *Arduino* Mega 2560; (b) *Arduino* Uno.

Tabela 2.5. Características dos *Arduino*s Uno e Mega 2560.

Arduino	**Uno**	**Mega 2560**
Microcontrolador	ATmega328	ATmega2560
Tensão de operação	5V	5V
Tensão de entrada (recomendado)	7-12V	7-12V
Tensão de entrada (limite)	6-20V	6-20V
Pinos de entrada e saída digital	14 (sendo 6 saídas PWM)	54 (sendo 15 saídas PWM)
Pinos de entrada analógica	6	16
Corrente DC por pino de entrada e saída	40 mA	40 mA
Corrente DC por pino 3.3V	50 mA	50 mA
Memória Flash	32 KB com 0.5 KB usada pelo *bootloader*	256 KB com 8 KB usada pelo *bootloader*
SRAM	2 KB	8 KB
EEPROM	1 KB	4 KB
Velocidade do *Clock*	16 MHz	16 MHz

Para a programação de um código *Arduino* utiliza-se o ambiente de desenvolvimento (IDE) *Arduino Software,* que também é de código aberto e roda nos sistemas operacionais *Windows*, *Mac OS X* e *Linux*. O ambiente é escrito em Java e baseado em *Processing* e outros *softwares* de código livre.

2.5.2 Formas de comunicação com Arduino

O *Arduino* possui diversos protocolos de comunicação integrados à placa ou que podem ser utilizados através de *Shields* comercialmente adquiridos. Dentre estes protocolos estão o padrão de comunicação serial, I²C, SPI, ethernet, Wifi, X-10, entre outros.

O módulo que processa a comunicação serial é a *UART* ou *USART* e todos os *Arduino*s possuem pelo menos uma porta serial. Esta porta serial padrão comunica-se tanto pelas portas digitais 0 (Rx) e 1 (Tx) quanto com um computador via cabo USB. A porta Rx é para recebimento de dados enquanto que a porta Tx é para transmissão de dados. Quando esse padrão é utilizado as portas digitais 0 e 1 não podem ser utilizadas nem como entrada nem como saída.

Alguns *Arduino*s possuem mais de uma porta de comunicação serial como é o caso do *Arduino* Mega que possui mais três portas seriais. Para a comunicação entre dois dispositivos seriais é necessário a ligação física da porta Rx do primeiro dispositivo com a porta Tx do outro e a porta Tx do primeiro com a porta Rx do outro dispositivo. Para que haja a troca de dados é necessário que os dispositivos estejam com o terra das placas interligados.

O padrão I²C no *Arduino* utiliza a biblioteca *Wire.h* e para a implementação do barramento são necessários dois pinos (SDA para os dados e SCL para o *clock*) que variam de acordo com a placa. No *Arduino* Uno usa-se os pinos analógicos A4 (SDA) e A5 (SCL), enquanto que no *Arduino* Mega 2560 os pinos utilizados são os digitais 20 (SDA) e 21 (SCL).

Há duas formas de se utilizar essa biblioteca no *Arduino* para comunicação entre os dispositivos. Na primeira forma o mestre escreve enquanto que o escravo recebe os dados. Já na segunda forma o mestre recebe os dados enviados pelo escravo.

Os primeiros dispositivos utilizados em automação residencial utilizavam o protocolo de comunicação X10, que utiliza a linha de energia elétrica da residência para medição de sinais e controle de acionamento de iluminação ou equipamentos eletrônicos. Neste protocolo todas as vezes que a tensão AC cruza o zero Volts um bit é enviado. No *Arduino* este protocolo é habilitado pela biblioteca *X10.h* e utiliza um módulo específico para o funcionamento do protocolo.

Outro protocolo de comunicação serial é o *Serial Peripheral Interface* (SPI) utilizado para comunicação de microcontroladores com dispositivos periféricos ou outros microcontroladores. Para este protocolo também há sempre um mestre (comumente o microcontrolador) que controla os dispositivos periféricos. Há três linhas comuns a todos os dispositivos:

- ✓ MISO (*Master In Slave Out*): a linha do escravo para enviar os dados ao mestre;
- ✓ MOSI (*Master Out Slave In*): a linha do mestre para enviar os dados aos periféricos;

- ✓ SCK (*Serial Clock*): os pulsos do *clock* gerados pelo mestre para sincronizar a transmissão de dados

  Há também uma linha específica para cada dispositivo:

- ✓ SS (*Slave Select*): o pino em cada dispositivo que o mestre pode utilizar para habilitar ou desabilitar o dispositivo.

Quando o pino SS está em nível lógico baixo, o dispositivo pode se comunicar com o mestre, porém quanto este nível é alto, o dispositivo ignora o mestre. Com isso pode haver a comunicação de vários dispositivos utilizando as mesmas linhas MISO, MOSI e SCK.

O *Arduino* também é capaz de se conectar à internet seja através de um módulo ethernet, ou da própria placa *Arduino* Ethernet ou Yún. O módulo é baseado no *chip* ethernet Wiznet W5100 que fornece uma rede IP capaz de utilizar os protocolos da camada de transporte TCP e UDP e suporta simultaneamente quatro conexões com *socket*. O módulo comunica-se com o microcontrolador pelo barramento SPI apresentado anteriormente e utiliza o conector padrão RJ-45.

Além da comunicação Ethernet via cabo, o *Arduino* pode se conectar à internet de forma wireless com o módulo Wifi ou com o *Arduino* Yún. O módulo permite a comunicação wireless através do padrão IEEE 802.11b/g. Assim como o módulo Ethernet o módulo Wifi fornece a rede IP com os protocolos da camada de transporte TCP e UDP.

O padrão IEEE 802.15 possui suporte para comunicação com o *Arduino* através do módulo Xbee.

2.5.3 Conversão padrão IEC 61131 em código de Arduino

Existem aplicativos que convertem um programa em alguma linguagem do padrão *IEC 61131* em linhas de código *Arduino*. Serão mostrados dois softwares que fazem esta conversão e para isso é apresentado um exemplo comum de uma aplicação industrial. Há um cilindro pneumático que deverá ser acionado por um botão. O cilindro só avançará se este estiver recuado e um botão for pressionado e ao atingir o seu curso máximo o cilindro deverá retornar.

O sistema é composto por um cilindro pneumático de ação simples e retorno por mola, uma válvula de 3 vias e 2 posições com acionamento por um solenoide (A) e retorno por mola, dois sensores fim de curso, 1S1 quando o pistão está recuado e 1S2 quando o pistão está acionado. Um esquema deste circuito é apresentado na Fig. 2.22.

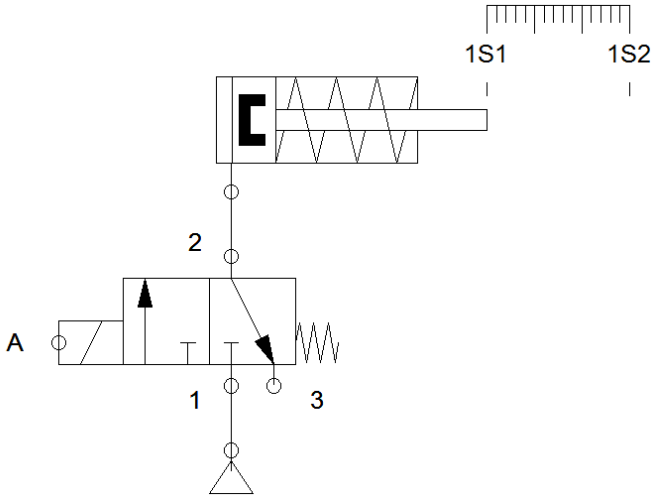


Figura 2.22. Exemplo de sistema eletropneumático.

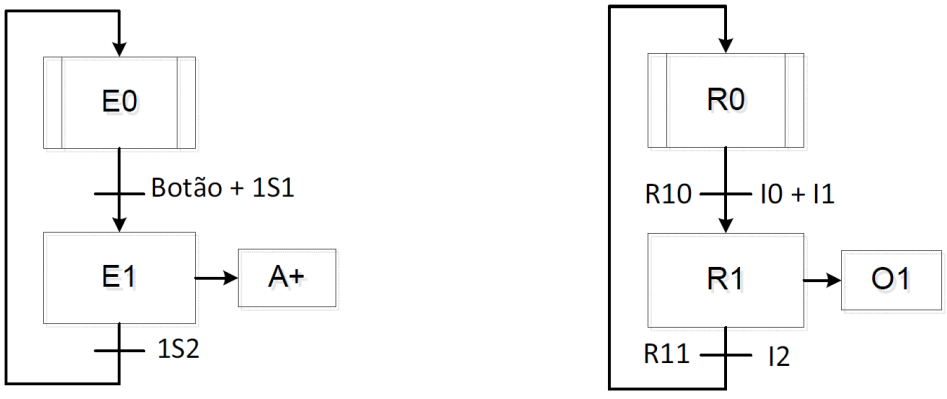
Este sistema foi programado em linguagem SFC sendo separado em duas etapas, E0 e E1. A etapa E0 corresponde ao estado inicial que é com o cilindro recuado, enquanto que a etapa E1 é o estado em que o cilindro avança. Neste estado a ação realizada é a de energizar o solenoide da válvula. Para sair de E0 e chegar em E1 é necessário que a transição seja ativada. Esta transição é a condição de sinal do sensor fim de curso 1S1 e o botão estarem em nível lógico alto ao mesmo tempo. A transição da etapa E1 para a E0 é a condição do sensor fim de curso 1S2 estar acionado.

Segundo Silveira e Santos (2005), é possível transformar um SFC/GRAFCET em linguagem *ladder*. Após a criação do GRAFCET criam-se tabelas de-para das entradas/saídas, etapas e transições para entradas digitais, saídas digitais e contatos lógicos do microcontrolador. Em seguida se desenha o GRAFCET modificado para posteriormente fazer o diagrama *ladder*. A Tab. 2.6 apresenta as tabelas de-para necessárias para modificar o GRAFCET. Na parte interna os contatos lógicos são substituídos pelos valores *Rx,* sendo R0 e R1 as etapas E0 e E1 e R10 e R11 as transições R10 e R11, respectivamente. Na parte externa as entradas são alteradas por entradas digitais do tipo *Ix*, sendo I0 para o botão, I1 para o sensor fim de curso 1S1 e I2 para o sensor fim de curso 1S2. Por fim, a saída A+ é substituída pela saída digital O1. O GRAFCET do sistema é apresentado na Fig. 2.23a, enquanto que a Fig. 2.23b apresenta o GRAFCET modificado. Por fim o diagrama *ladder* deste sistema é apresentado na Fig. 2.24.

Tabela 2.6. Tabelas de-para para o GRAFCET modificado.

GRAFCET	GRAFCET modificado
Interno	
E0	R0
E1	R1
Transição 0	R10
Transição 1	R11
Externo	
Botão	I0
1S1	I1
IS2	I2
A+	O1

O primeiro programa que permite a conversão para uma das linguagens padrões IEC61131 em *Arduino* é o Automgen. Criado há mais de duas décadas, o Automgen permite a criação de programas com as linguagens padronizadas, simulação em PC, geração de código e transferência para o CLP e outros dispositivos, tais como *Arduino*, PIC, entre outros. Este programa também permite a conversão de GRAFCET, *ladder*, diagrama de bloco de funções em código de *Arduino*.



(a) Grafcet (b) Grafcet modificado

Figura 2.23. (a) Grafcet do Sistema e (b) Grafcet modificado para transcrição em diagrama ladder.

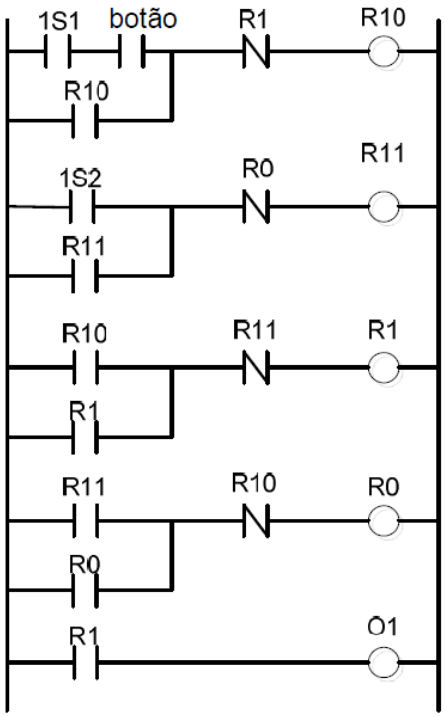


Figura 2.24. Diagrama ladder do exemplo utilizado.

O outro programa que permite a conversão de linguagem diagrama ladder em código de *Arduino* é o *Soapbox Snap*. É uma plataforma de automação livre e de código aberto para PC. A Fig. 2.25 apresenta o local no programa onde é feita a modificação para o tipo *SoapBox Snap Arduino Runtime* que permite a transferência do diagrama em linguagens de código de *Arduino*.

Esses programas fazem com que programas nas linguagens padrão IEC 61131 possam executar em *Arduino*. Caso se decida por utilizar um *Arduino* para o exemplo da Fig. 2.22, a programação da lógica do sistema fica resumida ao código em português estruturado apresentado na Tab. 2.7. Note que são necessárias apenas duas condições: um *if* e um *while*.

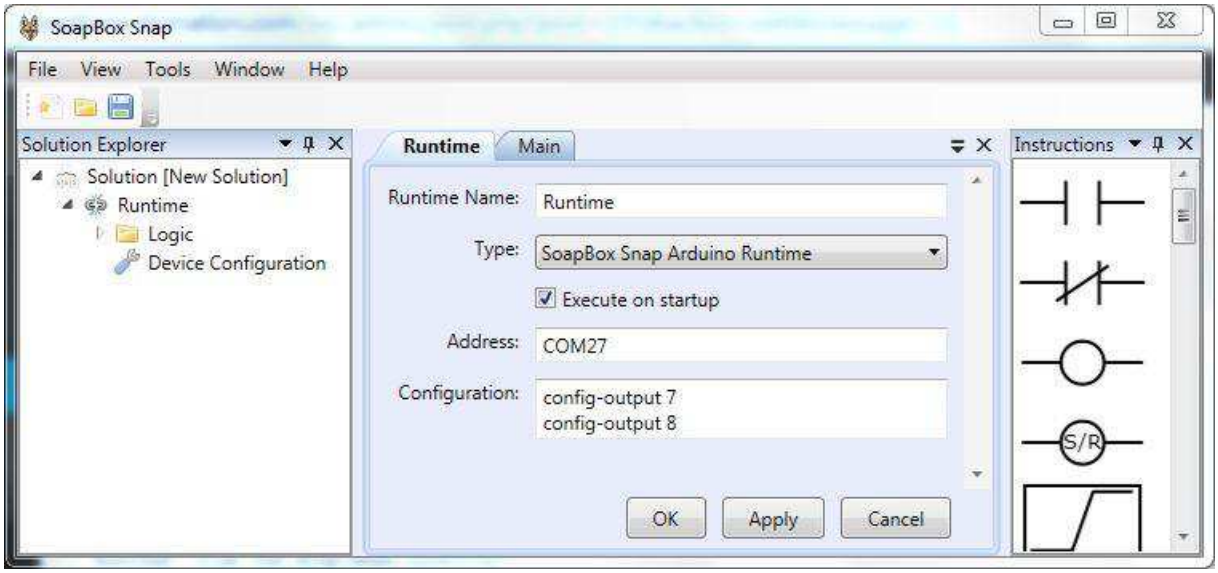


Figura 2.25. Alteração no software para transferência do diagrama ladder para *Arduino*.

Tabela 2.7. Lógica do sistema em português estruturado para o sistema eletropneumático.

```
início
se(1S1 e botão)
        enquanto(1S2 for falso)
                acione A+
        fim enquanto
fim se
fim
```

CAPÍTULO III

METODOLOGIA

Para atingir os objetivos apresentados no Capítulo I algumas atividades são necessárias. O primeiro objetivo específico era de estudar os tipos de redes centralizadas e distribuídas e requereu uma revisão bibliográfica apresentada no Capítulo II.

Para a definição de parâmetros de referência para sistema centralizados e distribuídos é preciso escolher um estudo de caso de sistemas a eventos discretos de baixa criticidade. Nesse caso um sistema centralizado bem como um grupo de equipamentos a serem aplicados para a distribuição do sistema são definidos. Os microcontroladores utilizados para o estudo de caso são os *Arduinos* Mega 2560 e Uno. Como há a necessidade de salvar os dados para uma posterior análise foi escolhido utilizar o *Arduino* Mega 2560 devido à quantidade de memória da placa. Com isso, pode-se utilizar uma biblioteca específica para guardar estes dados e não comprometer o desempenho do sistema. O *Arduino* Uno foi escolhido para o experimento de arquitetura de sistema distribuído com comunicação I²C entre três dispositivos. Como não precisa salvar dados neste microcontrolador, pode-se utilizar um dispositivo mais barato e que não influencia no desempenho do sistema. Para a arquitetura de sistema distribuído com comunicação sem fio entre dispositivos é utilizado módulo *Xbee* que suporta o padrão *IEEE 802.15.4*.

O terceiro objetivo específico é de avaliar o desempenho das redes distribuídas e para o tal são definidos os tipos de sistemas utilizados. O experimento com o sistema centralizado é a base de referência para o estudo de caso. Para os sistemas distribuídos optou-se pela arquitetura de controle de sistemas distribuídos com comunicação entre os dispositivos. Sendo assim, são propostas três distintas formas de comunicação entre estes dispositivos:

serial *RxTx,* padrão I²C e padrão *IEEE 802.15.4* utilizando *Xbee.* Para a comunicação serial *RxTx* são realizados quatro experimentos, pois a taxa de transmissão de dados e o tempo de solicitação de mensagens podem ser alterados cabendo a esses quatro experimentos avaliar situações distintas. Para a comunicação I²C são propostos dois experimentos distintos. No primeiro são utilizados apenas dois dispositivos, enquanto que no segundo é apresentado uma arquitetura contendo três dispositivos. Por fim, um experimento é realizado utilizando a comunicação sem fio com *Xbee*. Os tipos de experimentos são apresentados na Tab. 3.1. Para todos os ensaios são coletadas aproximadamente 100 amostras para garantir um ensaio estatisticamente coerente.

Tabela 3.1. Experimentos a serem realizados.

Sistema	Tipo de comunicação	Dispositivos
Centralizado	-	1
Distribuído com comunicação entre dispositivos	Serial *RxTx*	2
Distribuído com comunicação entre dispositivos	I²C	2
Distribuído com comunicação entre dispositivos	I²C	3
Distribuído com comunicação entre dispositivos	*Xbee*	2

Por fim, o quarto e último objetivo específico é de analisar os resultados encontrados quanto a desempenho e *hardware/software*. Para análise do desempenho são utilizadas ferramentas estatísticas (Apêndice A). Essas ferramentas são histograma dos dados, cálculo das médias, desvios-padrão amostrais, *kurtosis*, *skewness*, intervalo de confiança para a média e verificação do teste de normalidade para as distribuições. Será feita uma inferência dos dados com distribuição normal para verificar, com 95% de confiança, se as médias são iguais.

Em relação ao *software* é possível verificar a quantidade de linhas de código necessárias para execução dos programas para as diversas arquiteturas de controle. Essa quantidade de linhas não pode ser levada como uma comparação de qual sistema é melhor, uma vez que a programação depende de cada programador, que pode ter facilidade em um tipo de protocolo e dificuldade em outro, o que impactaria no total de linhas de código. Porém é uma análise válida quando se pretende observar o impacto de uma rede em um programa.

Para avaliação do *hardware* é necessário listar todos os dispositivos necessários para aplicação dos sistemas. Nesta análise é importante analisar a quantidade de microcontroladores e dispositivos extras utilizados, mantendo uma comparação de custo das arquiteturas de controle estudadas.

CAPÍTULO IV

ESTUDO DE CASO

A fim de realizar o estudo de caso, foi definido um problema de separação de peças utilizando uma bancada da Exsto (XC243) que representa um sistema a evento discreto de baixa criticidade. A Fig. 4.1 é uma foto da bancada. Note que há quatro compartimentos diferentes para as peças, sendo três quando válvulas são atuadas e um quando não há atuação.

A Tab. 4.1 mostra todos os dispositivos da bancada utilizados no estudo de caso. Os sensores óptico retroreflexivo, o sensor indutivo e o sensor capacitivo são utilizados na identificação da peça. Após a identificação o sistema atuará pelas válvulas avançando ou recuando os cilindros a fim de separar as peças. Os sensores de passagem de peça indicam o final da separação de uma peça.

As peças separadas são de dois tipos (metálica ou plástica) e três tamanhos (pequena, média ou grande). A Fig. 4.2 mostra as peças utilizadas na separação. As peças plásticas possuem duas cores (branco ou preto), porém não fará diferença para o estudo uma vez que não é levado em conta as cores das peças. Como as peças são de tamanhos e tipos distintos percebe-se que elas não possuem as mesmas massas e por isso a velocidade de movimentação da peça na esteira varia de acordo com o tipo peça a ser separada. A Tab. 4.2 apresenta a massa de cada uma das peças. Note que para o mesmo tamanho da peça, as metálicas possuem praticamente o dobro da massa das plásticas.

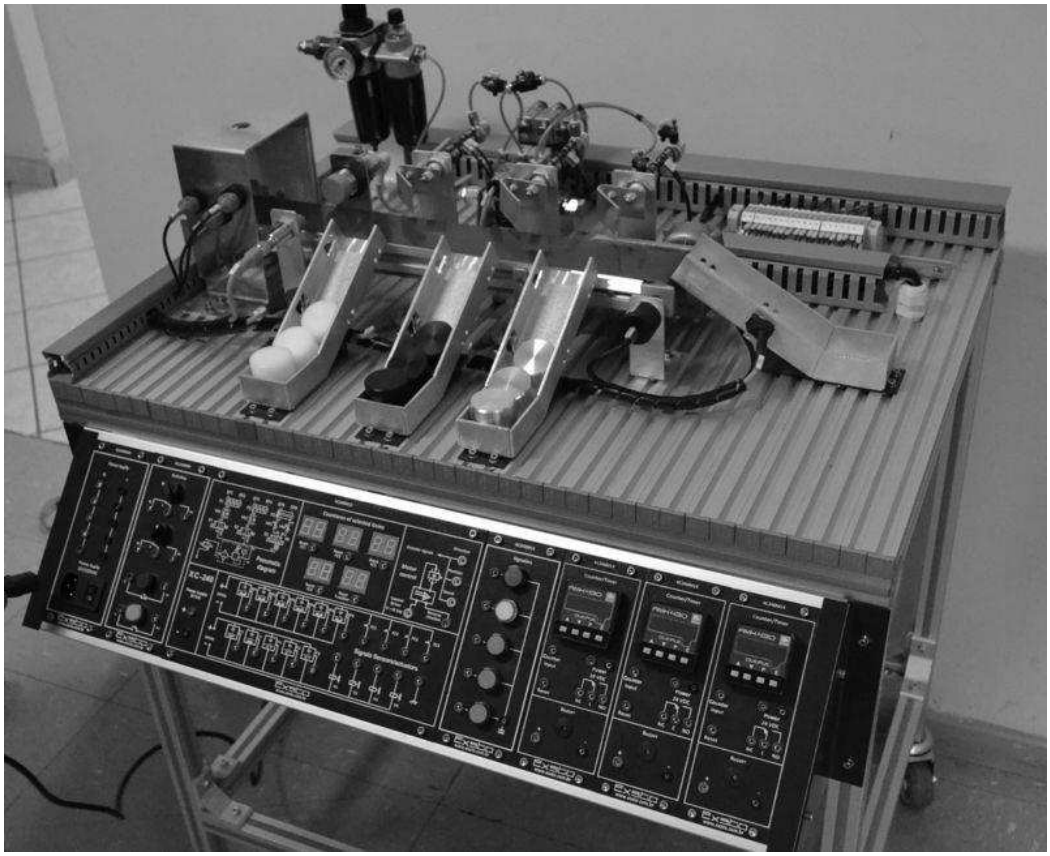


Figura 4.1. Bancada de separação de peças XC243.

Tabela 4.1. Lista de dispositivos utilizados na bancada.

Dispositivo	**Quantidade**
Sensor óptico retroreflexivo (identifica tamanho da peça)	3
Sensor indutivo digital (identifica peça metálica)	1
Sensor capacitivo digital (identifica fim da identificação de peça)	1
Sensor de passagem de peça	4
Cilindro dupla-ação com embolo magnético e regulador de velocidade	1
Cilindro simples-ação (retorno por mola) com embolo magnético e silenciador na via de escape	2
Válvula 5-2 vias com duplo acionamento elétrico e acionamento manual de emergência	1
Válvula 3-2 vias com acionamento simples elétrico, retorno por mola e acionamento manual de emergência	2

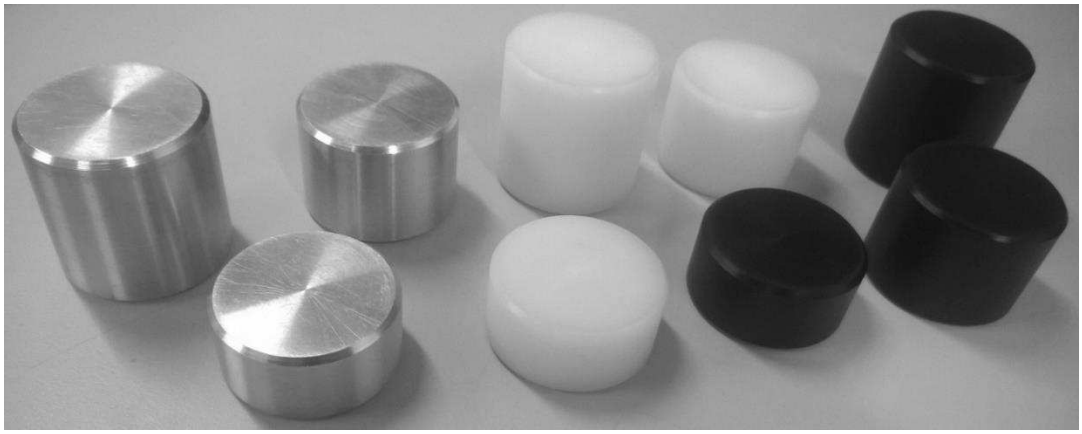


Figura 4.2. Peças metálicas e plásticas de três tamanhos distintos.

Tabela 4.2. Massas das peças.

Tipo	Tamanho	Massa (g)
Plástico	Pequena	31
	Média	47
	Grande	64
Metálica	Pequena	58
	Média	90
	Grande	121

O Capítulo II mostrou programas que transcrevem uma linguagem *ladder* ou SFC para linguagem C do *Arduino*. Como é necessário salvar os tempos de separação de cada peça para posterior análise é mais prático realizar a programação direta no ambiente de programação do *Arduino*, ao invés de utilizar algum destes programas.

O estudo de caso levou em conta a separação das peças de acordo com o tipo e o tamanho das peças, e foi definida a seguinte sequência:

- Peças metálicas, não importando o tamanho, são separadas no primeiro compartimento;
- Peças plásticas grandes são separadas no segundo compartimento;
- Peças plásticas médias são separadas no terceiro compartimento e
- Peças plásticas pequenas são separadas no quarto compartimento.

O fluxograma apresentado na Fig. 4.3 ilustra, de forma resumida, o modo como o sistema deve proceder. O processo é divido em duas etapas, sendo que a primeira consiste na identificação das peças e a segunda refere-se à atuação do sistema e, consequentemente,

na separação da peça identificada. O tempo de separação de uma peça é contado desde o instante em que a peça é detectada pelo sensor capacitivo (último sensor da identificação da peça) até que seja acionado o sensor fim de curso do determinado compartimento que a peça deve ser separada.

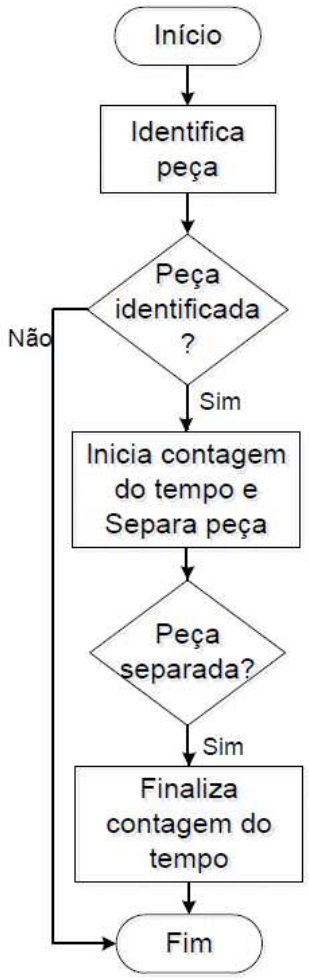


Figura 4.3. Fluxograma resumido do sistema.

A Fig. 4.4 apresenta o fluxograma da identificação da peça. Como a bancada já possui os sensores instalados em uma determinada ordem (porém pode ser alterado caso desejado), o processo de identificação manteve essa sequência. Para cada sensor foi definida uma variável denominada *flag"X"* em que *X* varia com o sensor. Os três primeiros sensores da bancada são os sensores óptico retroreflexivo, sendo o primeiro deles o sensor que está na altura média. A este sensor foi denominada a *flagf2.* Caso este sensor acuse nível lógico *alto* a *flag* booleana passa a ser *verdadeira*. O segundo dos sensores óptico retroreflexivo está na altura baixa, e a ele é associada a *flagf1.* Por fim o terceiro sensor está na altura alta, com a *flagf3*. Passados os três sensores a peça já pode ser identificada quanto à altura. Em seguida

o sensor indutivo captará um sinal nível lógico *alto* caso a peça seja metálica. O último sensor da identificação é o sensor capacitivo que este indica que a peça está no fim do processo de identificação e já pode ocorrer a separação. A Tab. 4.3 apresenta a tabela verdade da identificação das peças. Os valores '0' indicam que o sensor não foi acionado e os valores '1' indicam que o sensor foi acionado. Como são cinco variáveis a tabela é composta por 2^5 situações distintas. Para essas 32 possibilidades há apenas seis saídas válidas.

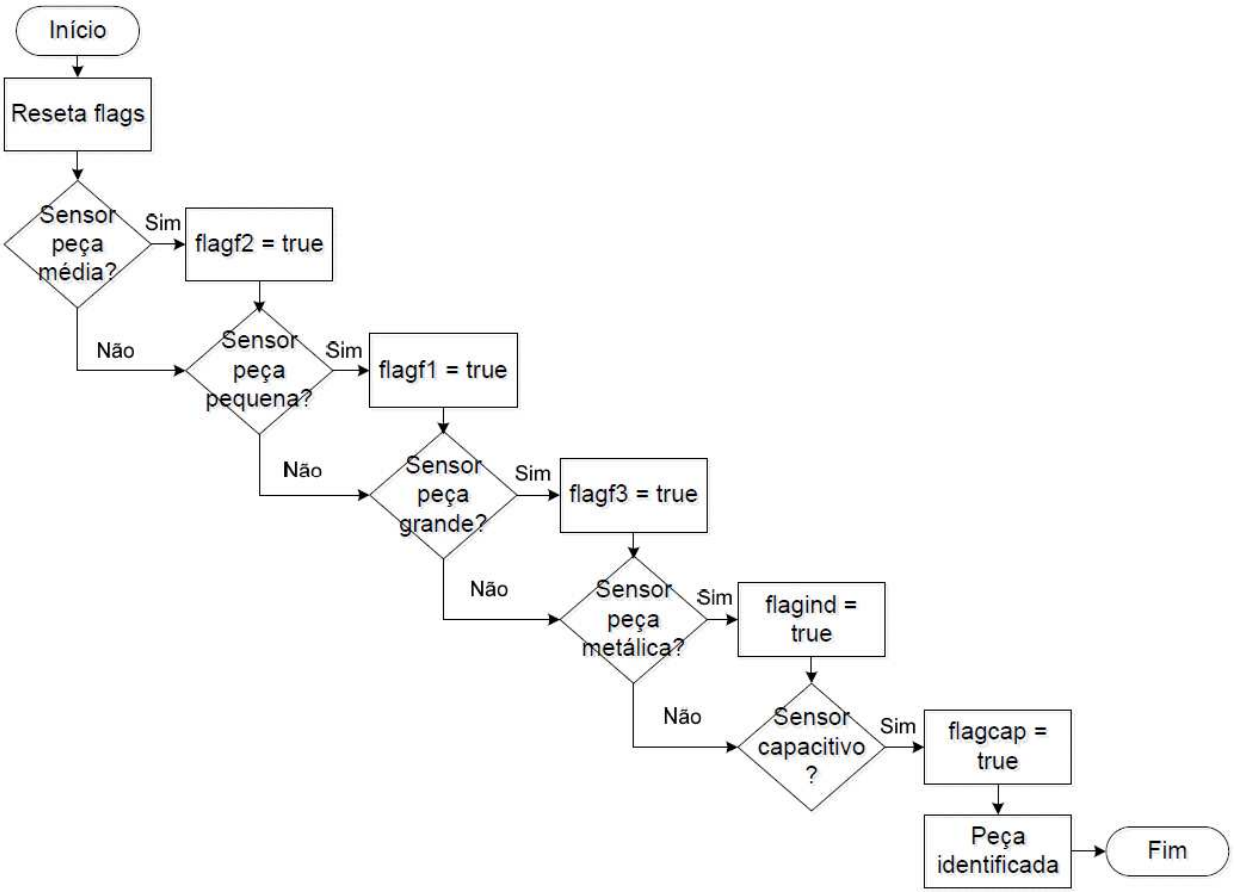


Figura 4.4. Fluxograma da identificação das peças.

O fluxograma da Fig. 4.5 apresenta a lógica de atuação do sistema quando uma peça é identificada. Conforme já descrito anteriormente, a separação de peças pode ocorrer em quatro compartimentos, portanto são utilizados três cilindros para que as peças caiam nos três primeiros compartimentos e a não atuação destes cilindros faz com que a peça caia no último compartimento que está no final da esteira. Quando a identificação percebe que uma peça é metálica, o sistema então avança o primeiro cilindro de simples ação até que o sensor de passagem de peça posicionado na entrada do compartimento seja acionado, que indica o fim da separação da peça metálica, logo a válvula responsável por este cilindro pode ser

desligada e o cilindro retorna devido à mola. O mesmo ocorre para as peças plásticas grandes que são separadas pelo segundo cilindro de simples ação, no segundo compartimento. Já as peças plásticas médias são separada por um cilindro de dupla ação, para isso é necessário acionar um solenoide da válvula para o avanço do cilindro quando a peça média é identificada. O retorno do cilindro só é realizado se o carretel da válvula trocar de posição, portanto deve ser alimentado o outro solenoide da válvula. Para este caso o avanço é chamado de válvula 3 e o retorno de válvula 4. Porém vale lembrar que se trata de uma única válvula duplo-solenoide. Quando uma peça plástica pequena é identificada nenhuma válvula é acionada, e a peça pequena percorre toda esteira até o quarto compartimento.

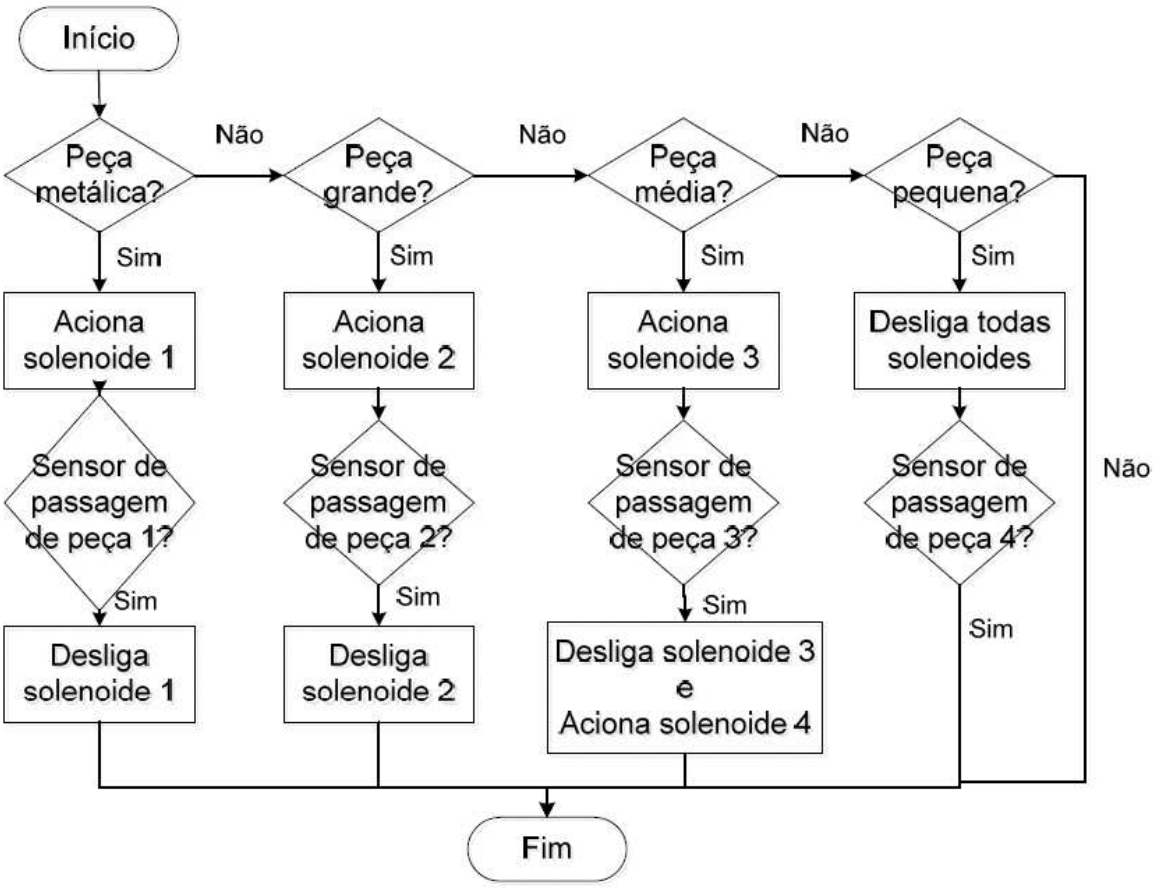


Figura 4.5. Fluxograma da separação das peças.

Tabela 4.3. Tabela verdade da identificação das peças.

flagf1	flagf2	flagf3	flagind	flagcap	Saída
0	0	0	0	0	X
0	0	0	0	1	X
0	0	0	1	0	X
0	0	0	1	1	X
0	0	1	0	0	X
0	0	1	0	1	X
0	0	1	1	0	X
0	0	1	1	1	X
0	1	0	0	0	X
0	1	0	0	1	X
0	1	0	1	0	X
0	1	0	1	1	X
0	1	1	0	0	X
0	1	1	0	1	X
0	1	1	1	0	X
0	1	1	1	1	X
1	0	0	0	0	X
1	0	0	0	1	pequena
1	0	0	1	0	X
1	0	0	1	1	metálica
1	0	1	0	0	X
1	0	1	0	1	X
1	0	1	1	0	X
1	0	1	1	1	X
1	1	0	0	0	X
1	1	0	0	1	média
1	1	0	1	0	X
1	1	0	1	1	metálica
1	1	1	0	0	X
1	1	1	0	1	grande
1	1	1	1	0	X
1	1	1	1	1	metálica

4.1 Sistema centralizado

Para o sistema centralizado todas as entradas e saídas estão conectadas ao mesmo dispositivo como mostra o esquema da Fig. 4.6. São necessárias nove entradas:

- sensor óptico de nível baixo (f1);
- sensor óptico de nível médio (f2);
- sensor óptico de nível alto (f3);
- sensor indutivo (find);
- sensor capacitivo (fcap);
- sensor de passagem da peça metálica (fc1);
- sensor de passagem da peça grande (fc2);
- sensor de passagem da peça média (fc3) e
- sensor de passagem da peça pequena (fc4)

Há também quatro saídas que atuarão os solenoides (V1, V2, V3 e V4).

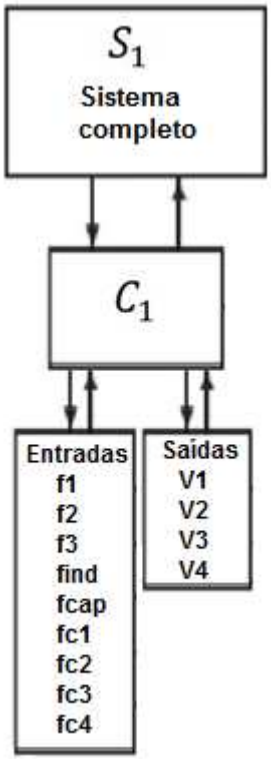


Figura 4.6. Esquema de conexão das entradas e saídas do sistema centralizado.

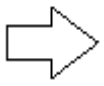
4.2 Sistema distribuído com comunicação serial *RxTx* entre dispositivos

Ao observarmos este estudo de caso percebe-se que é possível distribuir a lógica de controle do sistema a fim de torná-lo cada vez mais descentralizado. Para isso é necessário utilizar mais de um microcontrolador. Os próximos quatro ensaios são baseados na comunicação serial *RxTx* com par de cabos. O microcontrolador é capaz de trabalhar com várias taxas de transmissão de dados, desde 300 até 115200 bits por segundo. Quando inserimos uma comunicação entre dispositivos a taxa de transmissão limita a solicitação das mensagens já que não há possibilidade de haver comunicação quando um dispositivo solicita, repetidamente, uma mensagem em um tempo menor que o tempo de resposta da mensagem.

Portanto além de analisarmos a taxa de transmissão pode ser estudado o tempo de solicitação das mensagens. A Tab. 4.4 mostra os quatro ensaios realizados com a comunicação serial entre os dispositivos. As taxas de transmissão de dados escolhidas foram 1200 bps e 115200 bps, ou seja, uma baixa e uma alta, respectivamente. Inicialmente a taxa de 300 bps foi testada, e como o sistema apresentou muita incoerência esta taxa foi modificada para a taxa de 1200 bps. Observa-se que a tabela também apresenta o tempo teórico para que a maior mensagem seja enviada por um dispositivo. Esta mensagem é composta pela identificação da peça (sinal dos cinco sensores de identificação) e pelos sinais dos quatro sensores de passagem de peça conforme mostra a Tab. 4.5.

Tabela 4.4. Características dos quatro ensaios do sistema distribuído com comunicação serial *RxTx* entre dispositivos.

T. trans.	T. sol.
alto	alto
alto	baixo
baixo	alto
baixo	baixo



T. trans. (bps)	T. sol. (ms)	Tempo de troca de mensagens teórico [ms]
115200	4	1,25
115200	50	1,25
1200	100	120
1200	800	120

T. trans.: Taxa de transmissão de mensagens.
T. sol.: Tempo de solicitação de mensagens.

A forma com que a comunicação serial *RxTx* foi realizada é semelhante a um protocolo de comunicação mestre-escravo em que o mestre é responsável pela atuação das válvulas, e o escravo responsável pela identificação das peças e leitura dos sensores de passagem de peça. Com isso o mestre solicita o status de cada sensor. Os sinais são incorporados em uma

mensagem e enviados pelo escravo ao mestre. Neste tipo de sistema o mestre é conectado às saídas do sistema enquanto que o escravo é conectado às entradas. Esta arquitetura de controle e as ligações físicas dos controladores podem ser vistas na Fig. 4.7.

Tabela 4.5. Quadro de mensagens enviado pelo escravo ao mestre.

flagf1	*flagf2*	*flagf3*	*flagfind*	*flagfcap*	*flagfc1*	*flagfc2*	*flagfc3*	*flagfc4*

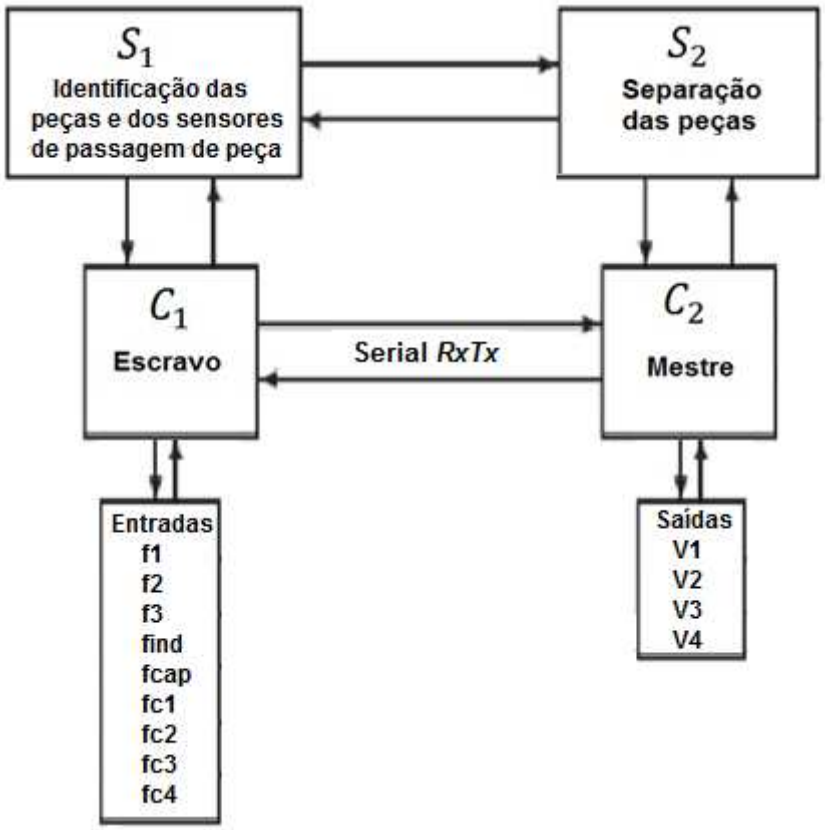


Figura 4.7. Sistema distribuído com comunicação serial entre dispositivos.

4.3 Sistema distribuído com comunicação I²C entre dispositivos

Outra forma de comunicação entre dispositivos é através do padrão barramento I²C. A velocidade padrão de comunicação deste padrão é 100 *kbps*, o que equivale a um tempo de 0,72 ms quando o quadro da mensagem é igual ao apresentado na Tab. 4.5. Como a forma de comunicação I²C é utilizando uma arquitetura mestre-escravo. O mestre é responsável pela separação das peças e o escravo é responsável pela identificação das peças e leitura

dos sinais dos sensores de passagem de peça (Fig. 4.8). A figura também mostra arquitetura de controle do sistema.

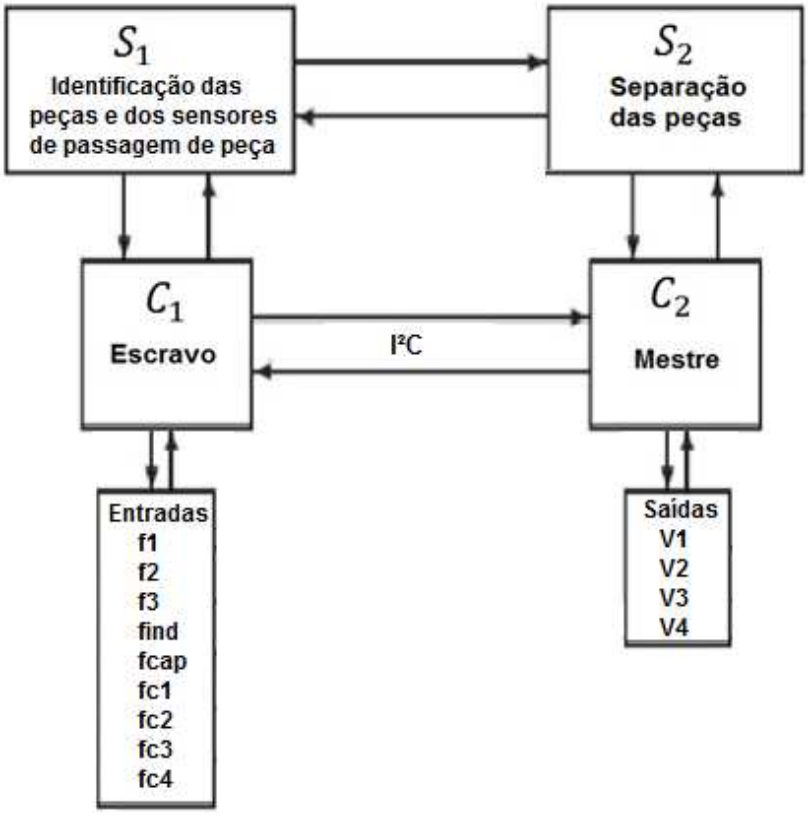


Figura 4.8. Sistema distribuído com comunicação I²C entre dispositivos.

Posteriormente o sistema foi distribuído em três dispositivos. Foram definidos dois escravos e um mestre. Nesse caso, a lógica de separação das peças é realizada pelo mestre, enquanto que a identificação das peças é realizada pelo escravo A e a identificação dos sensores de passagem de peça é realizada pelo escravo B. A Fig. 4.9 apresenta esta arquitetura de controle e as conexões das entradas e saídas aos respectivos controladores. Vale ressaltar que os três dispositivos estão conectados ao mesmo barramento, porém como a comunicação é endereçada os dois escravos não se comunicam. Note que há cinco entradas no escravo A, enquanto que o no escravo B há apenas quatro entradas. Portanto, o escravo A envia uma mensagem cujo quadro é apresentado na Tab. 4.6 e o escravo B envia uma mensagem cujo quadro é apresentado na Tab. 4.7. Esta diferença implica no tempo de troca de mensagens. Para o escravo A e o mestre, o tempo é de 0,40 ms e para o escravo B e o mestre o tempo é de 0,32 ms.

Tabela 4.6. Quadro da mensagem do escravo A.

flagf1	*flagf2*	*flagf3*	*flagfind*	*flagfcap*

Tabela 4.7. Quadro da mensagem do escravo B.

flagfc1	*flagfc2*	*flagfc3*	*flagfc4*

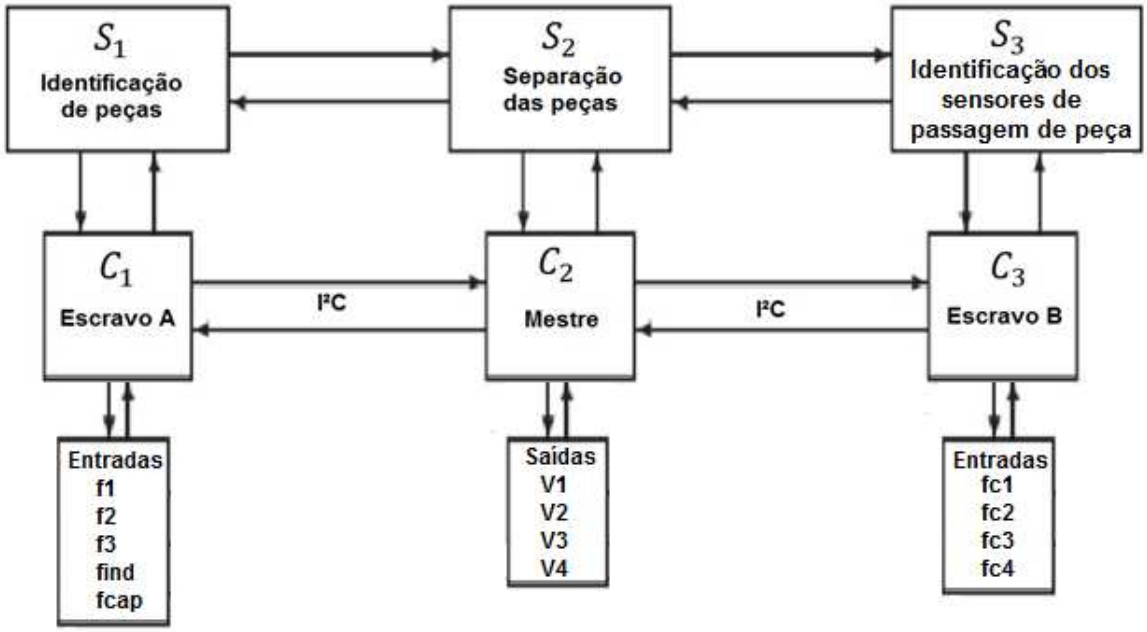


Figura 4.9. Sistema distribuído com comunicação I²C entre dispositivos utilizando um mestre e dois escravos.

Até o presente momento todos os ensaios foram realizados em um sistema que a lógica depende completamente do bom funcionamento dos dispositivos, isto é, quando um dos dispositivos por algum motivo para de trocar mensagens toda a separação das peças fica comprometida. A fim de evitar este problema foi proposta outra forma de distribuição da lógica. Nesta situação, também com três dispositivos, um é responsável pela identificação das peças, outro é responsável pela separação das peças metálicas e grandes e leitura dos sinais dos sensores de passagem de peça dos respectivos compartimentos e o último é responsável pela separação e leitura dos sensores de passagem de peça das peças média e pequena. A Fig. 4.10 apresenta esta arquitetura de controle e as conexões físicas das entradas e saídas. Note que neste caso, se o escravo A interromper a comunicação a separação das peças média e pequena não é comprometida, pois a comunicação entre a identificação das peças pelo mestre e a separação das peças pelo escravo B não foi interrompida. O mesmo ocorre

quando o escravo B interrompe a comunicação, mas a separação das peças metálica e grande não é comprometida. Nesta situação o tempo de troca de mensagens entre os escravos e o mestre é de apenas 0,40 ms uma vez que só é necessário enviar a mensagem de identificação da peça, cujo quadro é o mesmo da Tab. 4.6.

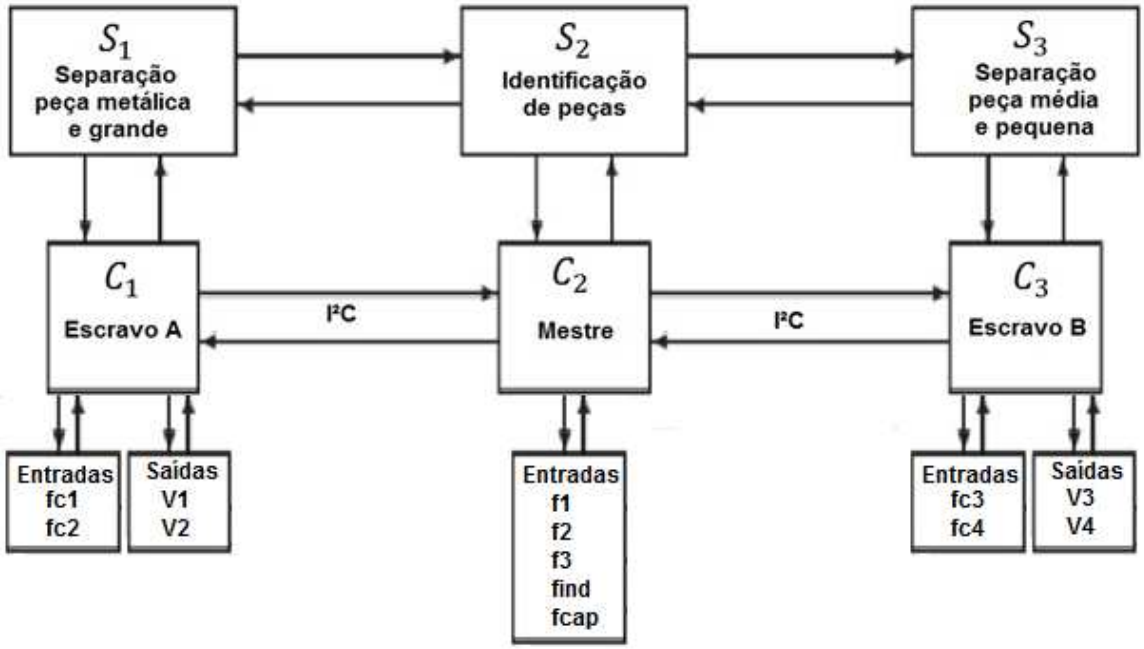
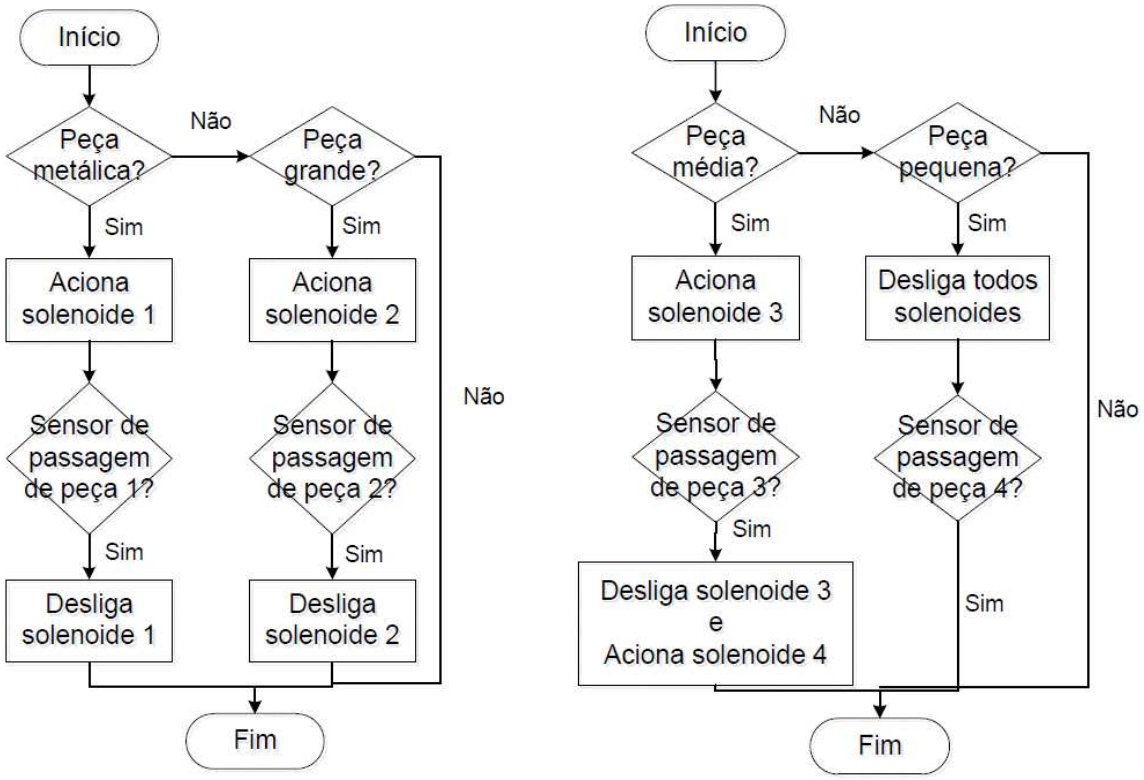


Figura 4.10. Sistema distribuído com comunicação I²C entre dispositivos com a lógica de separação de peças distribuída.

Note pela Fig. 4.11 que o fluxograma de separação das peças agora é dividido em dois. Isto acontece, pois, a lógica é distribuída nos dois escravos. Com isso o escravo A (Fig. 4.11a) não executa nenhuma influência na separação das peças médias e pequenas e o escravo B (Fig. 4.11b) não executa nenhuma influência na separação as peças metálicas e grandes.



(a) Escravo A (b) Escravo B

Figura 4.11. Fluxograma da separação de peças para cada um dos escravos.

4.4 Sistema distribuído com comunicação *Zigbee* entre dispositivos

A arquitetura de controle para este sistema é apresentada na Fig. 4.12, bem como as conexões físicas das entradas e saídas. Se observarmos as Fig. 4.7, Fig. 4.8 e Fig. 4.12 percebemos que as três figuras são praticamente idênticas, alterando apenas o tipo de comunicação entre os dispositivos. A utilização da mesma arquitetura de controle permite uma comparação entre os meios de comunicação. Este sistema com comunicação sem fio difere dos demais por não haver necessidade de manter todos os dispositivos no mesmo terra. Com isso há uma redução no custo de cabeamento e possibilidade de uma longa distância entre os dispositivos.

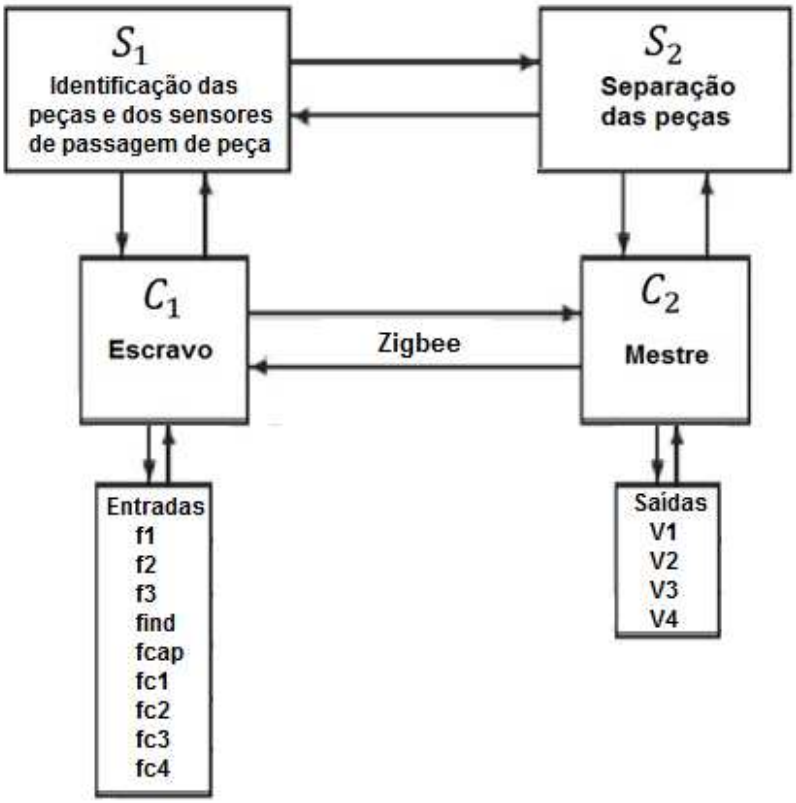


Figura 4.12. Esquema de conexão das entradas e saídas do sistema distribuído com comunicação *Zigbee* entre dois dispositivos.

CAPÍTULO V

RESULTADOS ENCONTRADOS

Neste capítulo são apresentados os resultados dos ensaios propostos no capítulo anterior. Afim de manter a sequência já utilizada inicialmente serão apresentados os resultados para o sistema centralizado, que servirá de referência. Os resultados dos ensaios dos sistemas distribuídos com comunicação entre dispositivos serão apresentados na seguinte ordem, a saber, comunicação serial *RxTx*, comunicação I²C e comunicação utilizando *Zigbee*.

Os resultados dos projetos levam em conta a quantidade de linhas de código utilizada em cada sistema. Para os cálculos dos tempos foram inseridas algumas linhas que não serão levadas em conta, pois são linhas que na aplicação do sistema não são necessárias. Outra forma de avaliar o desempenho do sistema é referente à quantidade de itens necessários para o funcionamento do sistema, ou seja, para cada sistema distribuído com comunicação entre dispositivos deve-se utilizar cabos para comunicação e aterramento dos dispositivos.

Os resultados apresentados consistem em uma tabela informando a quantidade de amostra para cada tipo de peça, a média aritmética do tempo de separação da peça e o desvio-padrão amostral dos dados, além do intervalo de confiança para média com nível de significância de 5%. Como já mencionado, o número de amostras coletadas para cada tipo de peças para cada sistema é de aproximadamente 100. Como os dados foram tratados e os *outliers* foram retirados, as tabelas apresentam uma quantidade menor de dados.

Além das tabelas, são apresentadas figuras contendo os histogramas dos dados para cada tipo de peça para cada sistema. Estes gráficos permitem uma análise visual de como se comporta a distribuição dos dados em cada sistema.

A partir dos histogramas pode-se observar uma distribuição aparentemente normal dos dados. Para a verificação desta hipótese são calculados os valores de *kurtosis* (curtose) e *skewness* (assimetria) dos dados, sendo que estes valores devem ser 3 e 0, respectivamente. Além destes cálculos podem ser realizados testes de hipótese. Foram listados quatro diferentes testes de hipótese para verificar a normalidade dos dados. Estes testes são: Kolmogorov-Smirnov, Lilliefors e Jarque-Bera. Os testes estão detalhados no Apêndice A. Para todos os testes o nível de significância definido por de 5%. Os testes foram feitos com o auxílio do software *Matlab*.

5.1 Sistema centralizado

O código utilizado para este tipo de sistema possui um total de 107 linhas (Apêndice B). As características deste ensaio estão apresentadas na Tab. 5.1. Como o sistema é centralizado há apenas um controlador (*Arduino* Mega 2560) e não há necessidade de cabos para comunicação. As quatros últimas linhas desta tabela apresentam a quantidade de cabos utilizados sem considerar a conversão dos sinais dos sensores de 24 V para 5 V; a quantidade de módulos relé para acionamento das válvulas; a quantidade de resistores de *pull-down* para os sensores e um *protoboard* para montagem do circuito. Como para todos os ensaios essas últimas linhas não são diferentes, elas são apresentadas apenas nesta tabela.

Como todos os sistemas possuem as mesmas características, diferenciando claramente na quantidade de dispositivos pode-se fazer um cálculo de custo sobre este ponto. O valor médio de um *Arduino* Mega 2560 encontrado na internet é de R$ 90,00. Portanto o custo do controle deste sistema será de R$ 90,00.

Tabela 5.1. Características do ensaio do sistema centralizado.

Característica	**Quantidade**
Linhas de código	107
Microcontrolador *Arduino* Mega 2560	1
Cabos de comunicação	0
Cabos de ligação	40
Módulo Relé 24 V	4
Resistor *pull-down*	9
protoboard	1

A Tab. 5.2 apresenta os dados para o sistema centralizado. Neste sistema toda a lógica de identificação e separação das peças concentra-se em um único microcontrolador. A Fig. 5.1 apresenta o histograma de cada tipo de peça para o sistema centralizado. Note que as distribuições se assemelham a uma distribuição normal e que os dados estão espalhados no tempo.

Os desvios-padrão amostrais são diferentes para os quatro tipos de peças. Isto ocorre pois, como foi dito anteriormente, as peças têm pesos distintos e quanto mais leve a peça menos esforço a esteira faz, conseguindo manter sua velocidade aproximadamente constante durante todo o percurso da peça. Como há três tamanhos distintos de peças metálicas, o desvio-padrão amostral para esse caso é o maior.

Tabela 5.2. Dados estatísticos para sistema centralizado.

Tipo de peça	**N**	$\bar{X}$ **[ms]**	σ **[ms]**	**IC de 95% para média [ms]**
Metálica	99	2428,0909	114,4384	$2203{,}7958 < \mu < 2652{,}386$
Grande	97	4418,9485	62,8141	$4295{,}8351 < \mu < 4542{,}0618$
Média	96	5908,7083	50,9718	$5808{,}8054 < \mu < 6008{,}6112$
pequena	94	6284,5638	33,7685	$6218{,}3787 < \mu < 6350{,}7489$

N.: Número de amostras
$\bar{X}$: média aritmética
σ : desvio-padrão amostral
IC.: intervalo de confiança

Para comprovar a normalidade da distribuição dos dados para este tipo de sistema a Tab. 5.3 apresenta os valores de *kurtosis* e *skewness* dos dados além dos resultados dos testes de hipóteses realizados. Observa-se que os valores estão próximos do esperado e os testes foram aceitos com exceção do teste de Kolmogorov-Smirnov (KS). Portanto, foram desenhadas as funções de distribuição acumulada (CDF – *cumulative distribution function*) dos dados e de uma curva com distribuição normal de média e desvio-padrão iguais ao encontrado para cada tipo de peça. Note que para as quatro figuras (Fig. 5.2 a Fig. 5.5) a CDF dos dados tende a ser a mesmo da distribuição normal. A *skewness* grande (0,43854) da peça pequena faz com que a Fig. 5.5 tenha uma curva um pouco fora da curva normal. Com esta análise apesar do teste de KS ter rejeitado a hipótese nula, podemos aceitar a distribuição dos dados como uma distribuição normal.

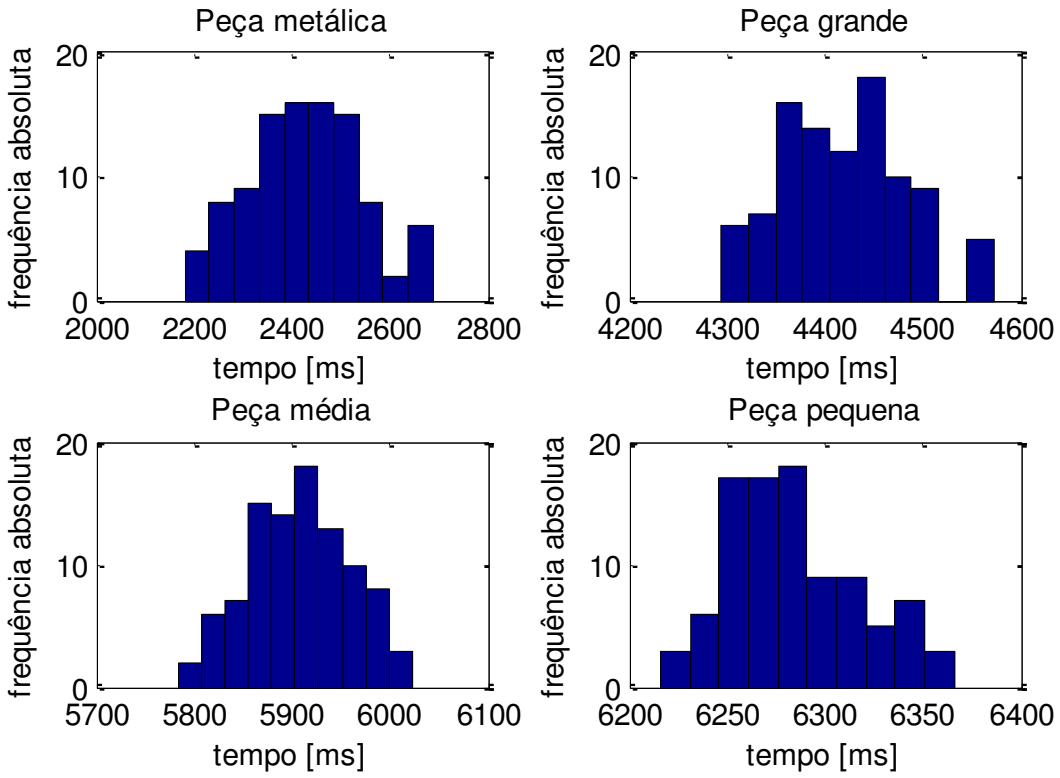


Figura 5.1. Histograma dos dados para o sistema centralizado.

Outro ponto importante para aceitar a hipótese de normalidade dos dados é o fato que em uma distribuição normal, a média aritmética, a mediana e a moda são o mesmo valor e para uma peça metálica a mediana é 2428 ms enquanto que a média é 2428,0909 ms, praticamente a mesma.

Tabela 5.3. Dados estatísticos e teste de hipóteses para sistema centralizado.

Tipo de peça	***kurtosis***	***skewness***	**Hipótese nula: Distribuição normal**		
			K-S	**Lilliefors**	**J-B**
Metálica	2,4593	0,12233	Rejeita	Aceita	Aceita
Grande	2,6484	0,26655	Rejeita	Aceita	Aceita
Média	2,5688	-0,06225	Rejeita	Aceita	Aceita
pequena	2,5301	0,43854	Rejeita	Aceita	Aceita

K-S: teste de Kolmogorov-Smirnov

J-B: teste de Jarque-Bera

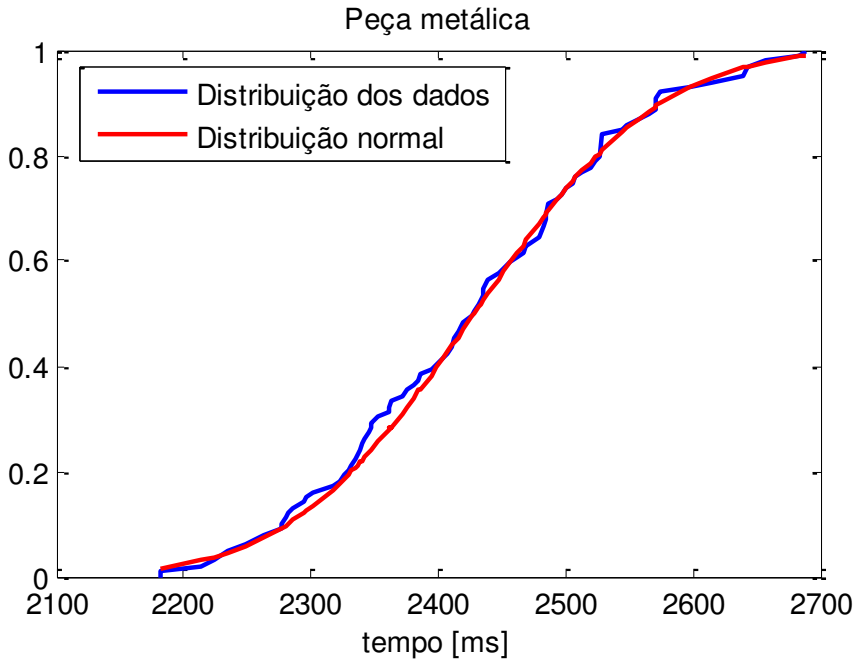


Figura 5.2. Função de distribuição acumulada para peça metálica do sistema centralizado.

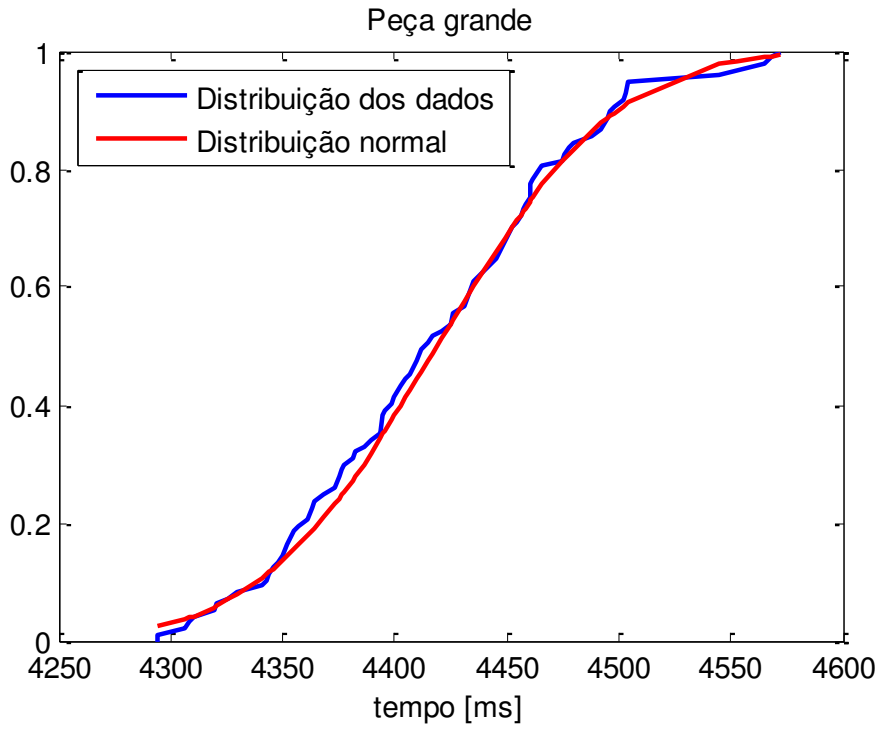


Figura 5.3. Função de distribuição acumulada para peça grande do sistema centralizado.

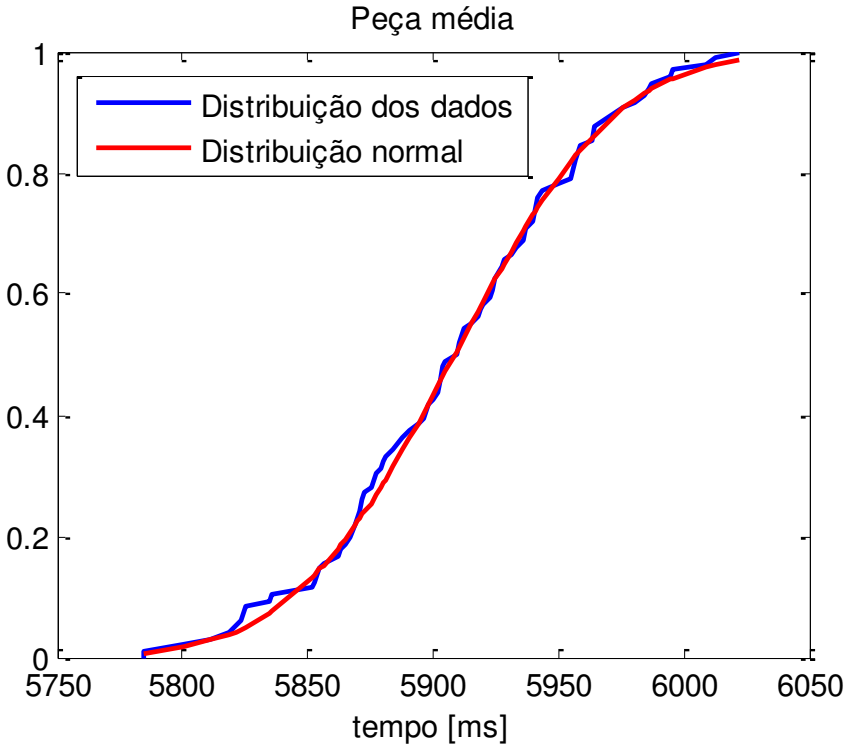


Figura 5.4. Função de distribuição acumulada para peça média do sistema centralizado.

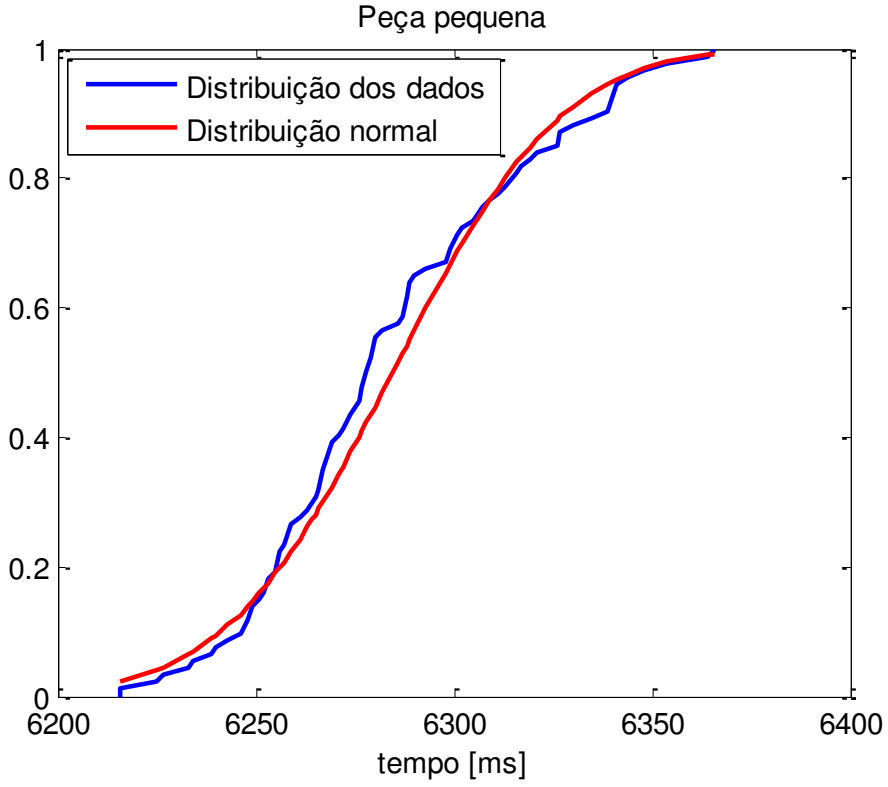


Figura 5.5. Função de distribuição acumulada para peça pequena do sistema centralizado.

5.2 Sistema distribuído com comunicação serial *RxTx* entre dispositivos

Como a arquitetura do sistema é distribuída há a necessidade de mais de um controlador, que neste caso, são dois *Arduino* Mega 2560. Como são utilizados dois microcontroladores o custo deste sistema é de R$ 180,00. A comunicação nesta arquitetura é feita de forma serial *RxTx*, então há a necessidade de dois cabos (transmissão de um dispositivo ligado a recepção do outro, e vice-versa) para comunicação e um cabo para conectar os terras dos dois dispositivos.

Para o sistema distribuído com comunicação serial *RxTx* entre dispositivos foram definidos quatro experimentos variando a taxa de transmissão de dados e o tempo de solicitação de mensagens. Os quatros experimentos são divididos em:

- Ensaio 1: taxa de transmissão de dados de 115200 bps e tempo de solicitação de mensagem de 4 ms;
- Ensaio 2: taxa de transmissão de dados de 115200 bps e tempo de solicitação de mensagens de 50 ms;
- Ensaio 3: taxa de transmissão de dados de 1200 bps e tempo de solicitação de mensagens de 100 ms e
- Ensaio 4: taxa de transmissão de dados de 1200 bps e tempo de solicitação e mensagens de 800 ms.

Para os quatro ensaios o código utilizado possui um total de 154 linhas, sendo 75 para o mestre e 79 para o escravo (Apêndice B). A única diferença nos códigos é o *setup* da taxa de transmissão e do tempo de solicitação das mensagens. A relação dos itens utilizados neste ensaio está apresentada na Tab. 5.4. Percebe-se que do total de linhas dos códigos 36 são próprias para a comunicação entre os dispositivos.

5.2.1 Ensaio 1

O primeiro ensaio deste tipo de sistema utilizou uma taxa de transmissão de mensagens de 115200 bps e um tempo de solicitação de mensagens de 4 ms. Este ensaio, com uma taxa de transmissão alta e um tempo de solicitação baixo, tenta minimizar ao máximo o tempo de resposta do sistema, portanto espera-se que seja o mais próximo do centralizado. A Tab. 5.5 apresenta a quantidade de amostras, a média amostral, o desvio-padrão amostral e o intervalo de confiança da média para este ensaio, enquanto que a Fig.

5.6 apresenta o histograma dos dados. Assim como no sistema centralizado os tempos estão espalhados no tempo tomando a forma de uma distribuição normal.

Tabela 5.4. Características do Sistema distribuído com comunicação serial *RxTx* entre dispositivos.

Característica	**Quantidade**
Linhas de código total	154
Linhas de código do mestre	75
Linhas de código de comunicação do mestre	16
Linhas de código do escravo	79
Linhas de código de comunicação do escravo	20
Microcontrolador *Arduino* Mega 2560	2
Cabos para comunicação	3

Tabela 5.5. Dados estatísticos para o sistema distribuído com comunicação serial *RxTx* para o ensaio 1.

Tipo de peça	**N**	$\overline{X}$ **[ms]**	σ **[ms]**	**IC de 95% para média [ms]**
Metálica	94	2482,7872	117,0794	$2253{,}3158 < \mu < 2712{,}2587$
Grande	103	4435,5534	66,7449	$4304{,}7359 < \mu < 4566{,}3709$
Média	105	6002,7905	71,9783	$5861{,}7156 < \mu < 6143{,}8654$
pequena	102	6359,8725	50,8295	$6260{,}2486 < \mu < 6459{,}4965$

N.: Número de amostras
$\overline{X}$: média amostral
σ : desvio-padrão amostral
IC.: intervalo de confiança

Para este ensaio a Tab. 5.6 apresenta os valores de *kurtosis* e *skewness* dos dados além dos resultados dos testes de hipóteses realizados. Observa-se que os valores estão próximos do esperado e os testes foram aceitos com exceção do teste de Kolmogorov-Smirnov (KS). Portanto, foram desenhadas as funções de distribuição acumulada dos dados e de uma curva com distribuição normal de média e desvio-padrão iguais ao encontrado para cada tipo de peça. Note que para as quatro figuras (Fig. 5.7 a Fig. 5.10) a CDF dos dados tende a ser a mesmo da distribuição normal. Sendo assim, aceitaremos a hipótese nula, ou seja, as distribuições para um nível de significância de 5% podem ser adotadas como distribuições normais.

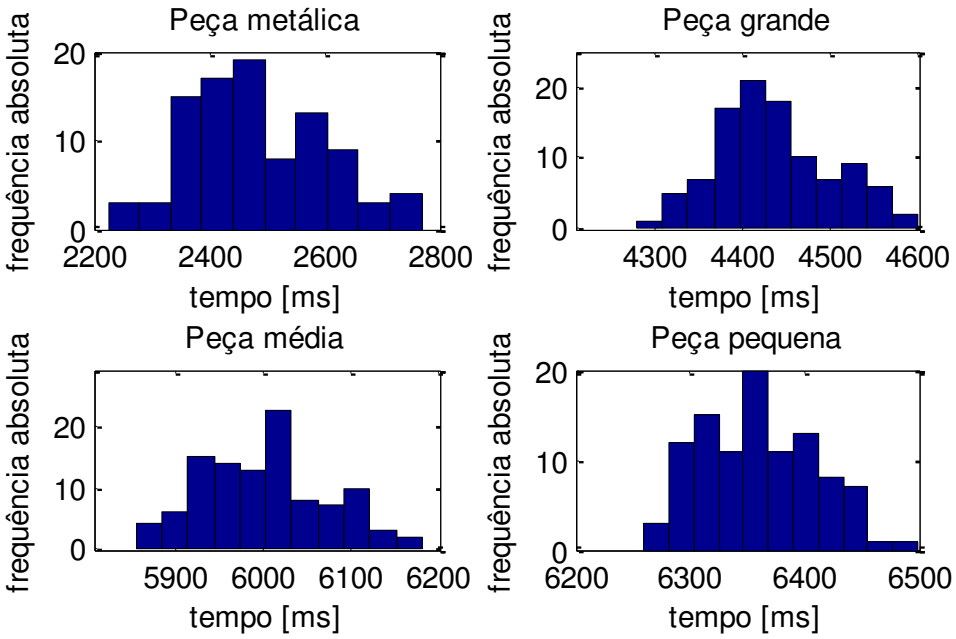


Figura 5.6. Histograma dos dados para o sistema distribuído com comunicação serial entre os dispositivos a uma taxa de 115200 bps e tempo de solicitação de mensagem de 4 ms.

Tabela 5.6. Dados estatísticos e teste de hipóteses para sistema distribuído com comunicação serial *RxTx* entre dispositivos do ensaio 1.

Tipo de peça	***kurtosis***	***skewness***	**Hipótese nula: Distribuição normal**		
			K-S	**Lilliefors**	**J-B**
Metálica	2,5122	0,24785	Rejeita	Aceita	Aceita
Grande	2,6805	0,31825	Rejeita	Aceita	Aceita
Média	2,5266	0,21835	Rejeita	Aceita	Aceita
pequena	2,3917	0,27267	Rejeita	Aceita	Aceita

K-S: teste de Kolmogorov-Smirnov

J-B: teste de Jarque-Bera

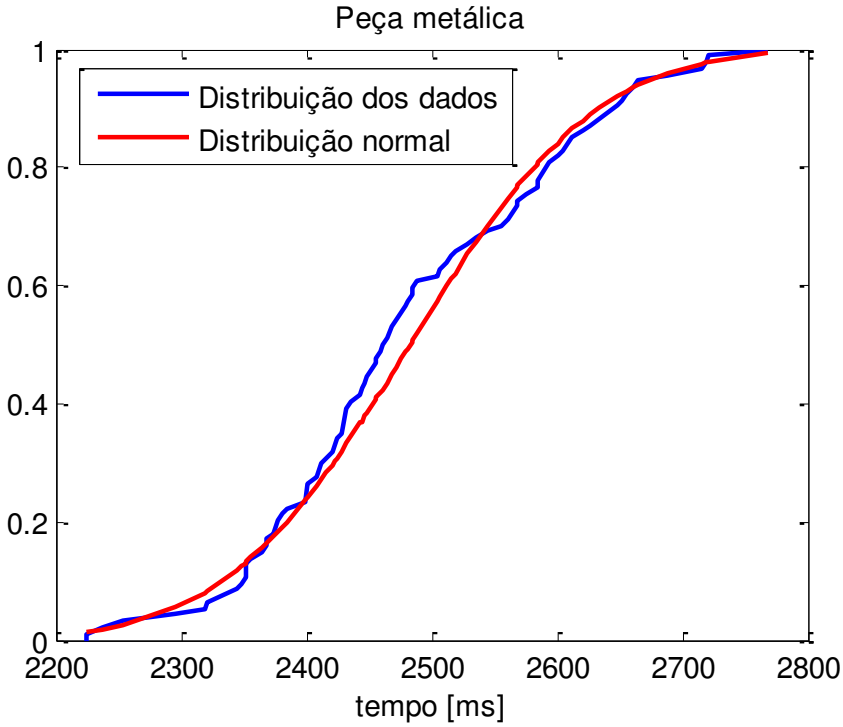


Figura 5.7. Função de distribuição acumulada para peça metálica do sistema distribuído com comunicação serial *RxTx* entre dispositivos do ensaio 1.

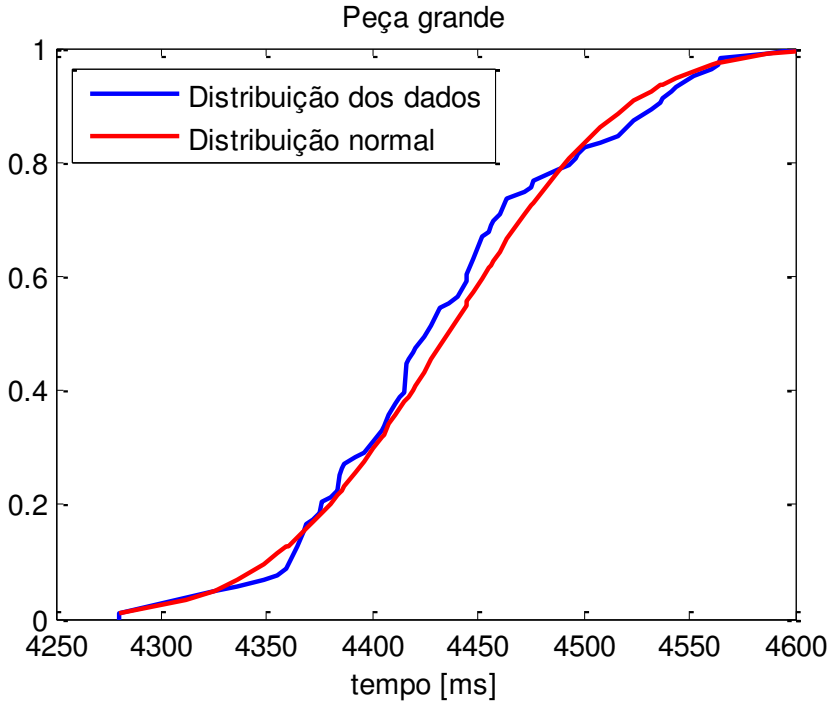


Figura 5.8. Função de distribuição acumulada para peça grande do sistema distribuído com comunicação serial *RxTx* entre dispositivos do ensaio 1.

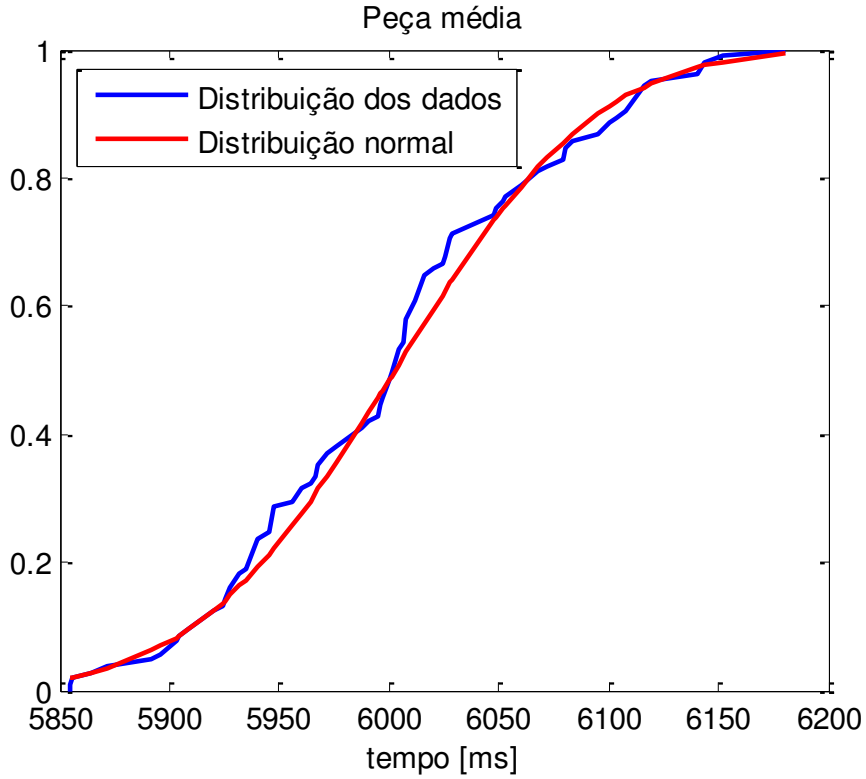


Figura 5.9. Função de distribuição acumulada para peça média do sistema distribuído com comunicação serial *RxTx* entre dispositivos do ensaio 1.

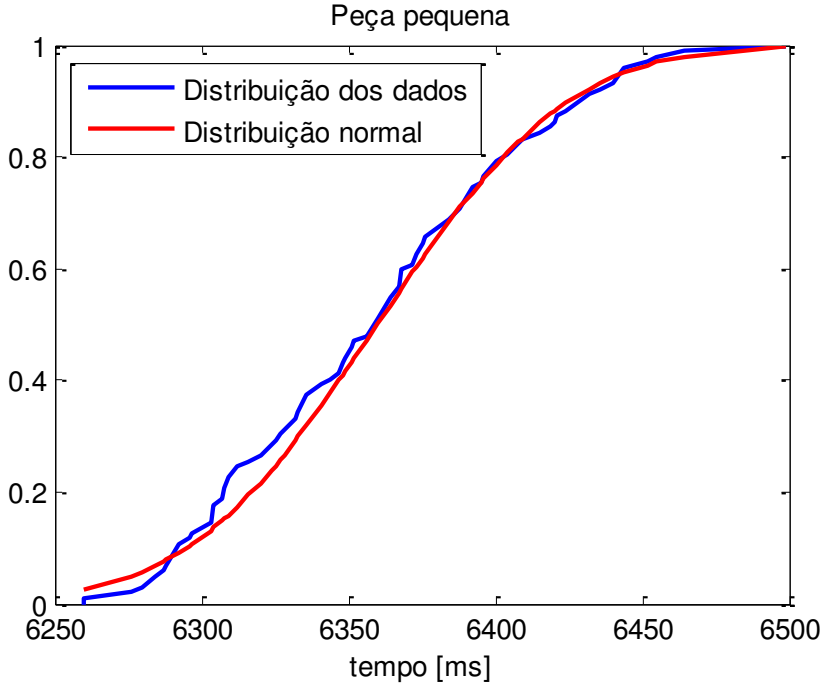


Figura 5.10. Função de distribuição acumulada para peça pequena do sistema distribuído com comunicação serial *RxTx* entre dispositivos do ensaio 1.

5.2.2 Ensaio 2

O segundo ensaio do sistema distribuído com comunicação serial *RxTx* manteve a taxa de transmissão de dados em 115200 bps porém o tempo de solicitação de mensagens foi de 50 ms. Os dados estatísticos do ensaio estão apresentados na Tab. 5.7 e aparentemente são próximos aos dados do sistema com tempo de solicitação de 4 ms, porém ao observarmos a Fig. 5.11 percebem-se algumas lacunas no eixo do tempo. Isto ocorre pois há um atraso inserido no sistema e em algumas das vezes este atraso faz com que algumas medidas concentrem em algum valor.

O aparecimento dessas lacunas nos histogramas reflete nos testes de hipóteses referente a este ensaio. A Tab. 5.8 mostra que tanto o teste de Kolmogorov-Smirnov quanto o teste de Lilliefors rejeitaram a hipótese nula, portanto os dados não são, com 5% de significância, de uma distribuição normal. Para este ensaio não são apresentados os CDFs dos dados pois não há necessidade de confrontar os resultados dos testes.

Tabela 5.7. Dados estatísticos para Lógica Distribuída *RxTx* com 115200 bps de taxa de transmissão de dados e 50 ms de tempo de solicitação de mensagem.

Tipo de peça	**N**	$\overline{X}$ **[ms]**	σ **[ms]**	**IC de 95% para média [ms]**
Metálica	102	2531,7353	114,4074	$2307{,}501 < \mu < 2755{,}9696$
Grande	96	4437,8542	75,7668	$4289{,}3539 < \mu < 4586{,}3544$
Média	95	5974,0737	82,3912	$5812{,}59 < \mu < 6135{,}5574$
pequena	93	6308,7312	41,1643	$6226{,}0907 < \mu < 6391{,}3717$

N.: Número de amostras
$\overline{X}$: média amostral
σ : desvio-padrão amostral
IC.: intervalo de confiança

Tabela 5.8. Dados estatísticos e teste de hipóteses para sistema distribuído com comunicação serial *RxTx* entre dispositivos do ensaio 2.

Tipo de peça	***kurtosis***	***skewness***	**Hipótese nula: Distribuição normal**		
			K-S	**Lilliefors**	**J-B**
Metálica	2,8199	0,055811	Rejeita	Rejeita	Aceita
Grande	2,7677	0,10133	Rejeita	Rejeita	Aceita
Média	3,2817	0,39729	Rejeita	Rejeita	Aceita
pequena	2,3252	0,25514	Rejeita	Rejeita	Aceita

K-S: teste de Kolmogorov-Smirnov; J-B: teste de Jarque-Bera

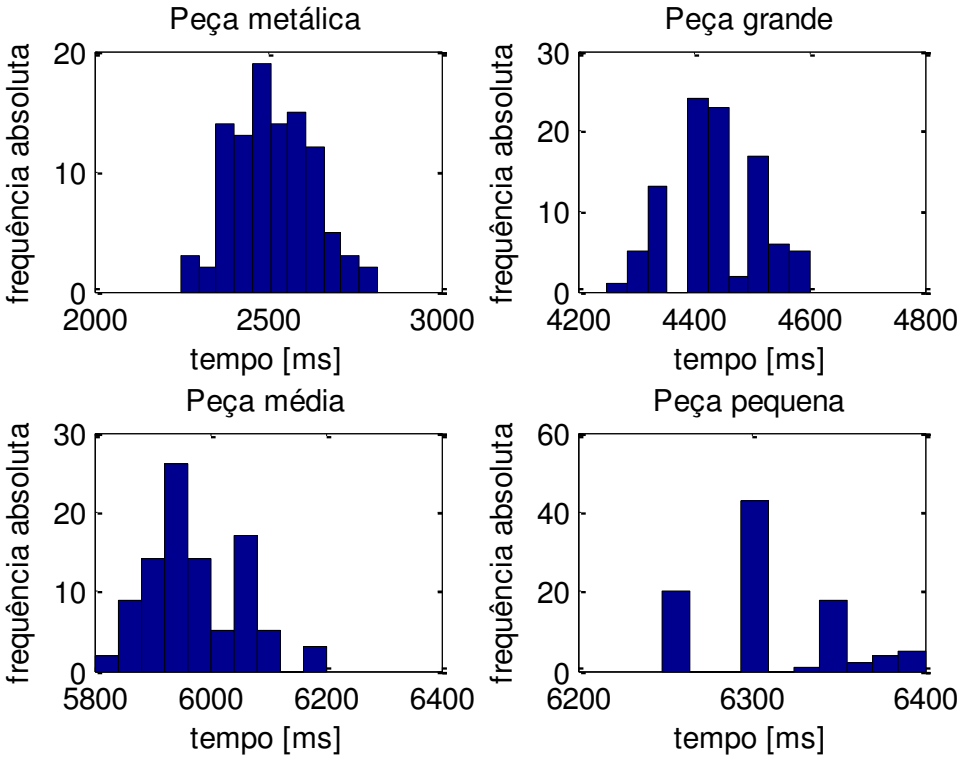


Figura 5.11. Histograma dos dados para o sistema distribuído com comunicação serial entre os dispositivos a uma taxa de 115200 bps e tempo de solicitação de mensagem de 50 ms.

5.2.3 Ensaio 3

Visto os dois ensaios para uma taxa de transmissão de dados bastante elevada, os dois próximos ensaios são para uma taxa baixa (1200 bps). Como apresentado, o tempo para da troca de mensagem nessa taxa de transmissão leva 120 ms, então o terceiro ensaio para este tipo de sistema trabalha com um tempo de solicitação de mensagens de 100 ms. A Tab. 5.9 apresenta os dados estatísticos do ensaio enquanto que a Fig. 5.12 apresenta o histograma. Para este sistema é notória a presença de um atraso na comunicação uma vez que os dados estão concentrados em tempos com um intervalo de aproximadamente 100 ms.

Novamente os testes de hipótese de K-S e Lilliefors rejeitaram a hipótese nula. O teste de J-B também rejeitou esta hipótese para a peça pequena. Isto é comprovado pelo baixíssimo valor da *kurtosis* para este tipo de peça. Os resultados são apresentados na Tab. 5.10. Portanto, conclui-se com 5% de significância que os dados para este ensaio não são podem ser considerados como uma distribuição normal.

Tabela 5.9. Dados estatísticos para Lógica Distribuída *RxTx* com 1200 bps de taxa de transmissão de dados e 100 ms de tempo de solicitação de mensagem.

Tipo de peça	**N**	$\bar{X}$ **[ms]**	σ **[ms]**	**IC de 95% para média [ms]**
Metálica	82	2473,1463	90,2724	$2296{,}2156 < \mu < 2650{,}0771$
Grande	98	4414,2449	74,5661	$4268{,}098 < \mu < 4560{,}3918$
Média	95	5989,4421	76,4936	$5839{,}5174 < \mu < 6139{,}3668$
pequena	99	6334,3333	47,7145	$6240{,}8146 < \mu < 6427{,}852$

N.: Número de amostras

$\bar{X}$: média amostral

σ : desvio-padrão amostral

IC.: intervalo de confiança

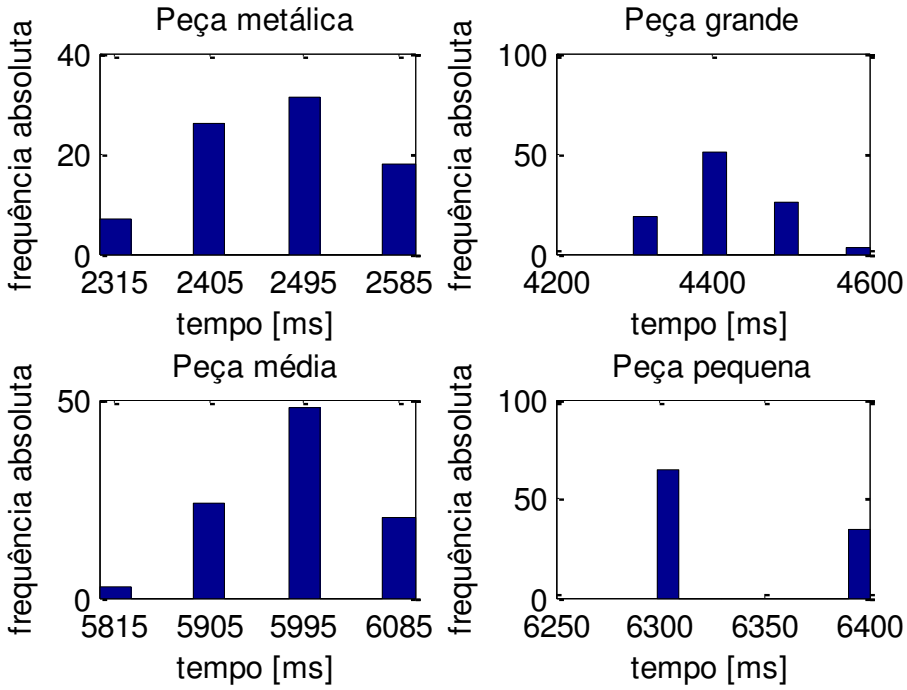


Figura 5.12. Histograma dos dados para o Sistema distribuído com comunicação serial entre os dispositivos a uma taxa de 1200 bps e tempo de solicitação de mensagem de 100 ms.

Tabela 5.10. Dados estatísticos e teste de hipóteses para sistema distribuído com comunicação serial *RxTx* entre dispositivos do ensaio 3.

Tipo de peça	*kurtosis*	*skewness*	Hipótese nula: Distribuição normal		
			K-S	Lilliefors	J-B
Metálica	2,1987	-0,15395	Rejeita	Rejeita	Aceita
Grande	2,7076	0,21369	Rejeita	Rejeita	Aceita
Média	2,6362	-0,25371	Rejeita	Rejeita	Aceita
Pequena	1,4349	0,65927	Rejeita	Rejeita	Rejeita

K-S: teste de Kolmogorov-Smirnov
J-B: teste de Jarque-Bera

5.2.4 Ensaio 4

Por fim, para o quarto ensaio foi utilizado um tempo de solicitação de mensagens de 800 ms. Observa-se pela Fig. 5.13 que os dados estão concentrados em apenas dois valores para as peças metálica, grande e média, e em três valores para as peças pequena. Porém verifica-se que os três valores da peça pequena são 6399, 6400 e 6401 ms que podem ser considerados o mesmo valor. Para as outras peças os dois valores de cada tipo estão espaçados de aproximadamente 800 ms e o desvio-padrão para estes casos é de aproximadamente 400 ms como mostra a Tab. 5.11.

Tabela 5.11. Dados estatísticos para Lógica Distribuída *RxTx* com 1200 bps de taxa de transmissão de dados e 800 ms de tempo de solicitação de mensagem.

Tipo de peça	N	$\overline{X}$ [ms]	σ [ms]	IC de 95% para média [ms]
Metálica	115	2900,8783	388,7557	$2138{,}9311 < \mu < 3662{,}8254$
Grande	114	4421,0175	401,1234	$3634{,}8301 < \mu < 5207{,}205$
Média	117	6017,1026	401,3792	$5230{,}4138 < \mu < 6803{,}7913$
pequena	107	6400,0093	0,21698	$6399{,}5841 < \mu < 6400{,}4346$

N.: Número de amostras
$\overline{X}$: média amostral
σ : desvio-padrão amostral
IC.: intervalo de confiança

Graficamente é observado que os dados não seguem de uma distribuição normal, e a Tab. 5.12 confirma esta afirmação mostrando que todos os testes de hipóteses foram rejeitados para os quatro tipos de peças.

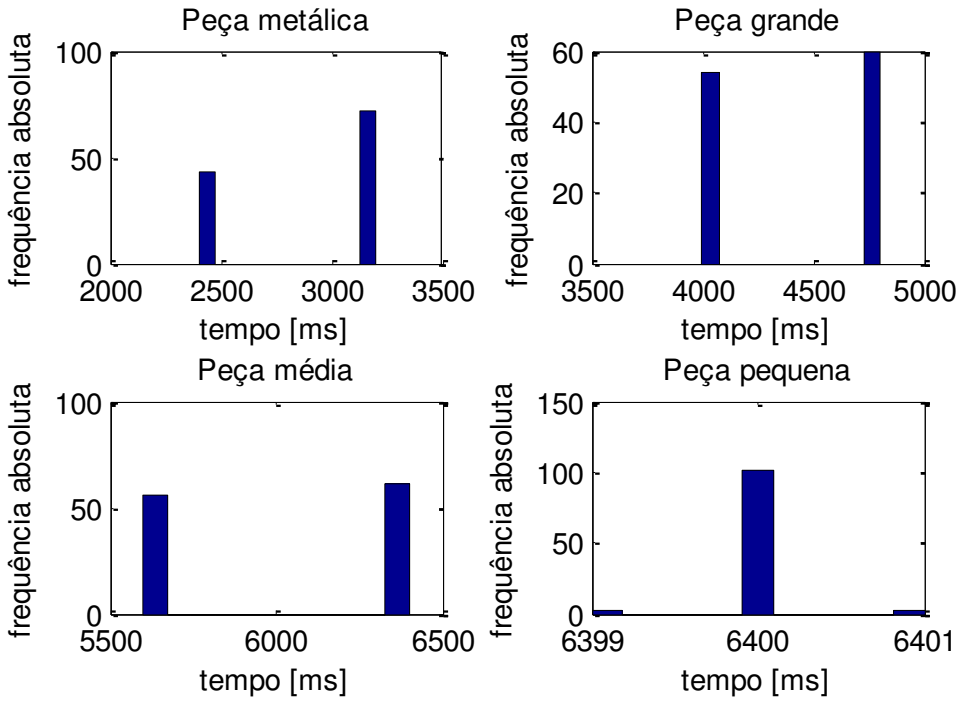


Figura 5.13. Histograma dos dados para o Sistema distribuído com comunicação serial entre os dispositivos a uma taxa de 1200 bps e tempo de solicitação de mensagem de 800 ms.

Tabela 5.12. Dados estatísticos e teste de hipóteses para sistema distribuído com comunicação serial *RxTx* entre dispositivos do ensaio 4.

Tipo de peça	***kurtosis***	***skewness***	**Hipótese nula: Distribuição normal**		
			K-S	**Lilliefors**	**J-B**
Metálica	1,2716	-0,5212	Rejeita	Rejeita	Rejeita
Grande	1,0111	-0,10541	Rejeita	Rejeita	Rejeita
Média	1,0073	-0,085552	Rejeita	Rejeita	Rejeita
pequena	21,3309	0,7979	Rejeita	Rejeita	Rejeita

K-S: teste de Kolmogorov-Smirnov
J-B: teste de Jarque-Bera

5.3 Sistema distribuído com comunicação I²C entre dispositivos

Para o sistema distribuído com comunicação I²C entre dispositivos são definidos três ensaios, sendo o primeiro deles com comunicação entre dois dispositivos, o segundo com

comunicação entre três dispositivos e o terceiro com comunicação entre três dispositivos com a lógica distribuída.

5.3.1 Ensaio 1

O primeiro ensaio é semelhante ao ensaio 1 do sistema distribuído com comunicação serial *RxTx* entre dispositivos, pois são utilizados 2 microcontroladores e são necessários 3 cabos para comunicação, sendo dois de troca de mensagens e um de ligação dos terras. Portanto, o custo deste sistema também é de R$ 180,00. A Tab. 5.13 apresenta as características deste ensaio. O total de linhas de código é de 150, sendo 32 exclusivas para comunicação (Apêndice B).

Tabela 5.13. Características do Sistema distribuído com comunicação serial *RxTx* entre dispositivos para o ensaio 1.

Item	**Quantidade**
Linhas de código total	150
Linhas de código do mestre	72
Linhas de código de comunicação do mestre	12
Linhas de código do escravo	78
Linhas de código de comunicação do escravo	20
Microcontrolador *Arduino* Mega 2560	2
Cabos para comunicação	3

Para este ensaio a quantidade de amostras, a média amostral, o desvio-padrão amostral e o intervalo de confiança para a média são apresentados na Tab. 5.14. Como o tempo de troca de mensagens é muito pequeno o histograma dos dados volta a ser espalhado no tempo como mostra a Fig. 5.14.

A Tab. 5.15 mostra que o teste de hipóteses de K-S foi rejeitado para os quatro tipos de peças, enquanto que os testes de Lilliefors e Jarque-Bera aceitaram a hipótese nula. Portando os CDFs dos dados são apresentados nas figuras seguintes (Fig. 5.15 a Fig. 5.18). Pelas curvas apresentadas será aceita a hipótese nula, portanto os dados se assemelham a uma distribuição normal para um nível de significância de 5%.

Tabela 5.14. Dados estatísticos para sistema distribuído com comunicação I²C entre os dispositivos para o ensaio 1.

Tipo de peça	**N**	$\overline{X}$ **[ms]**	σ **[ms]**	**IC de 95% para média [ms]**
Metálica	100	2462,75	92,9808	$2280{,}5111 < \mu < 2644{,}9889$
Grande	99	4408,697	63,6049	$4284{,}0337 < \mu < 4533{,}3602$
Média	102	5990,5098	72,4165	$5848{,}576 < \mu < 6132{,}4436$
pequena	101	6333,6535	47,7764	$6240{,}0134 < \mu < 6427{,}2936$

N.: Número de amostras

$\overline{X}$: média amostral

σ : desvio-padrão amostral

IC.: intervalo de confiança

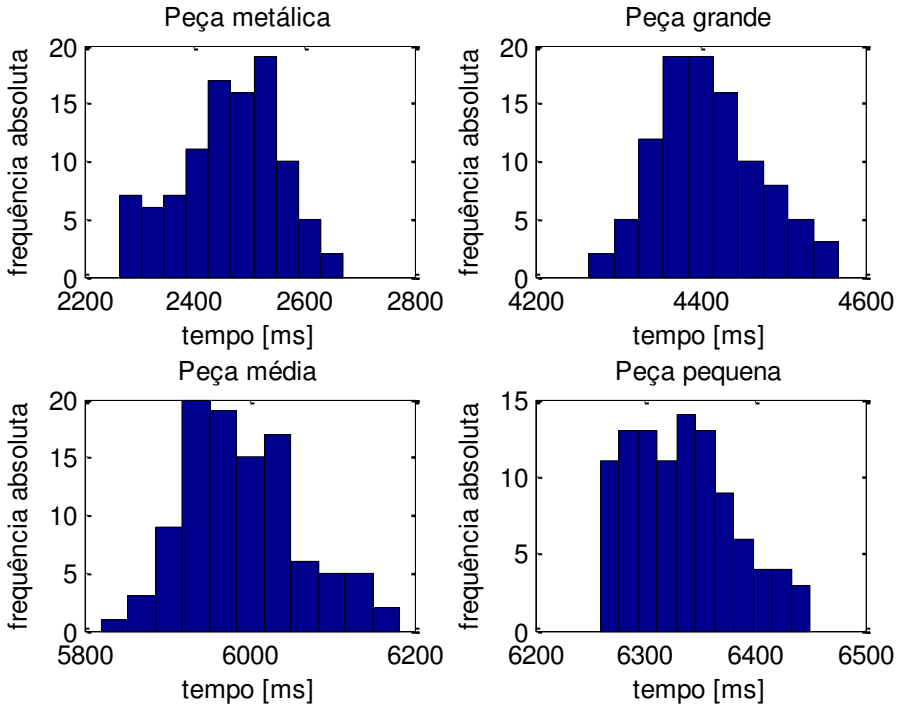


Figura 5.14. Histograma dos dados para o sistema distribuído com comunicação I²C entre dois dispositivos para o ensaio 1.

Tabela 5.15. Dados estatísticos e teste de hipóteses para sistema distribuído com comunicação I²C entre dispositivos do ensaio 1.

Tipo de peça	***kurtosis***	***skewness***	**Hipótese nula: Distribuição normal**		
			K-S	**Lilliefors**	**J-B**
Metálica	2,592	-0,255	Rejeita	Aceita	Aceita
Grande	2,774	0,32105	Rejeita	Aceita	Aceita
Média	2,9094	0,42388	Rejeita	Aceita	Aceita
pequena	2,4902	0,46865	Rejeita	Aceita	Aceita

K-S: teste de Kolmogorov-Smirnov
J-B: teste de Jarque-Bera

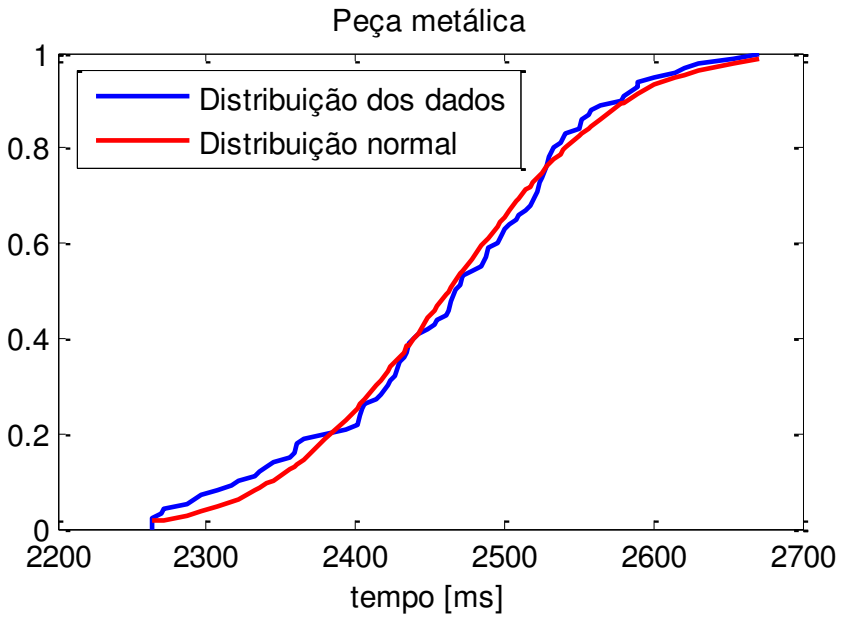


Figura 5.15. Função de distribuição acumulada para peça metálica do sistema distribuído com comunicação I²C entre dispositivos do ensaio 1.

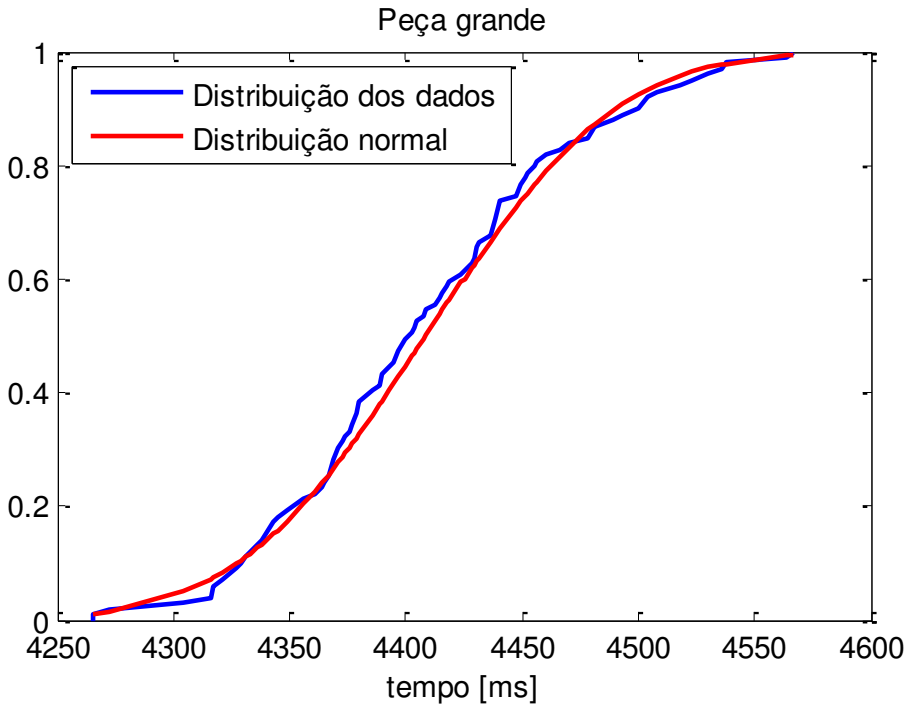


Figura 5.16. Função de distribuição acumulada para peça grande do sistema distribuído com comunicação I²C entre dispositivos do ensaio 1.

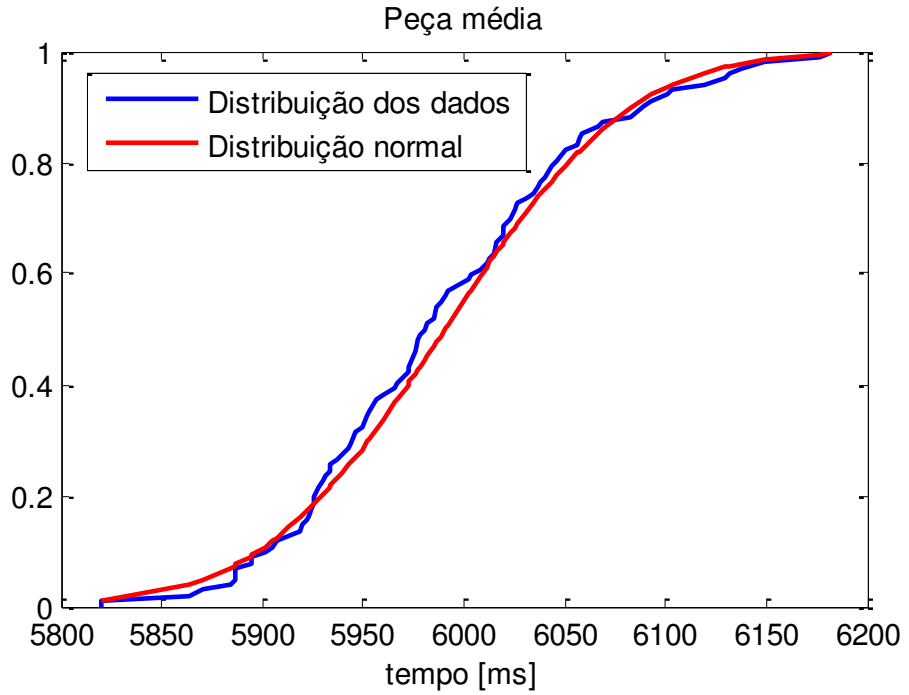


Figura 5.17. Função de distribuição acumulada para peça média do sistema distribuído com comunicação I²C entre dispositivos do ensaio 1.

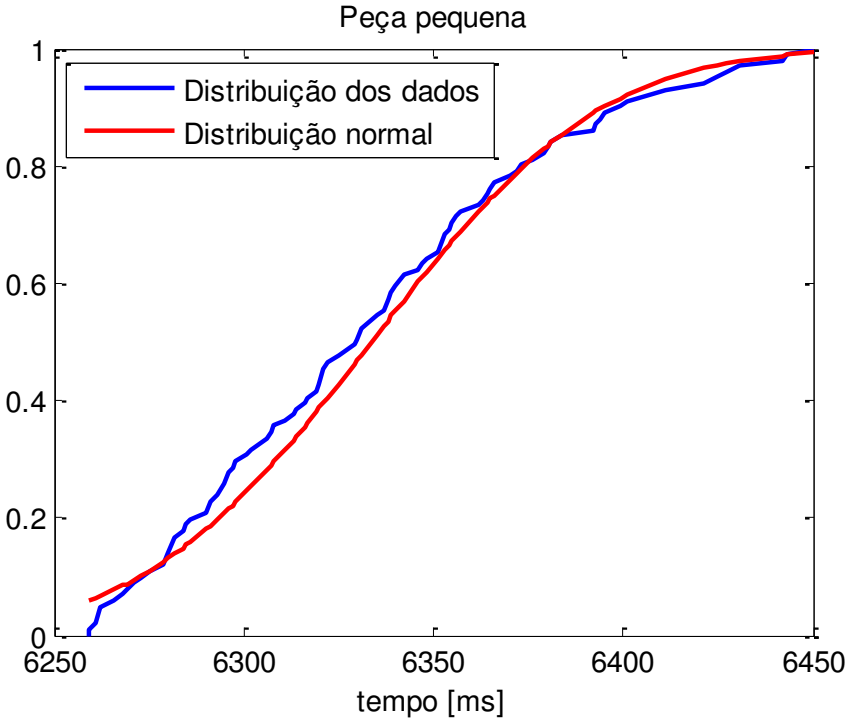


Figura 5.18. Função de distribuição acumulada para peça pequena do sistema distribuído com comunicação I²C entre dispositivos do ensaio 1.

5.3.2 Ensaio 2

Para o ensaio 2 são utilizados três microcontroladores, sendo dois *Arduino* Mega 2560 (R$ 90,00 cada) e um *Arduino* Uno (R$ 60,00 cada). Para este ensaio o custo considerado é de R$ 240,00. São necessárias 236 linhas de código (Apêndice B) distribuídas entre o mestre e os escravos sendo que destas 73 são exclusivas para utilização da comunicação via barramento I²C. Como as duas linhas de comunicação (dados e *clock*) devem ser conectadas e os terras também, são necessários oito cabos para a comunicação. Esses dados são apresentados na Tab. 5.16.

Os dados estatísticos são mostrados na Tab. 5.17 e a Fig. 5.19 apresenta o histograma dos dados. Novamente os dados estão espalhados no tempo. A Tab. 5.18 apresenta os testes de hipóteses e como era de se esperar apenas o teste K-S rejeitou a hipótese nula. Por isso são apresentadas as curvas CDFs nas Fig. 5.20 a Fig. 5.23. Para este ensaio pode-se dizer com um nível de significância de 5% que os dados se assemelham a uma distribuição normal.

Tabela 5.16. Características do Sistema distribuído com comunicação I²C entre três dispositivos para o ensaio 2.

Item	**Quantidade**
Linhas de código total	236
Linhas de código do mestre	142
Linhas de código de comunicação do mestre	50
Linhas de código do escravo 1	51
Linhas de código de comunicação do escravo 1	12
Linhas de código do escravo 2	43
Linhas de código de comunicação do escravo 2	11
Microcontrolador *Arduino* Mega 2560	2
Microcontrolador *Arduino* Uno	1
Cabos para comunicação	8

Tabela 5.17. Dados estatísticos para Sistema Distribuído com comunicação I²C entre dispositivos, sendo dois escravos.

Tipo de peça	**N**	$\overline{X}$ **[ms]**	σ **[ms]**	**IC de 95% para média [ms]**
Metálica	102	2465,2059	98,9287	$2271{,}3093 < \mu < 2659{,}1025$
Grande	100	4359,88	53,9018	$4254{,}2345 < \mu < 4465{,}5255$
Média	101	5927,099	53,1318	$5822{,}9626 < \mu < 6031{,}2355$
pequena	102	6292,3725	41,6537	$6210{,}7329 < \mu < 6374{,}0122$

N.: Número de amostras

$\overline{X}$: média amostral

σ : desvio-padrão amostral

IC.: intervalo de confiança

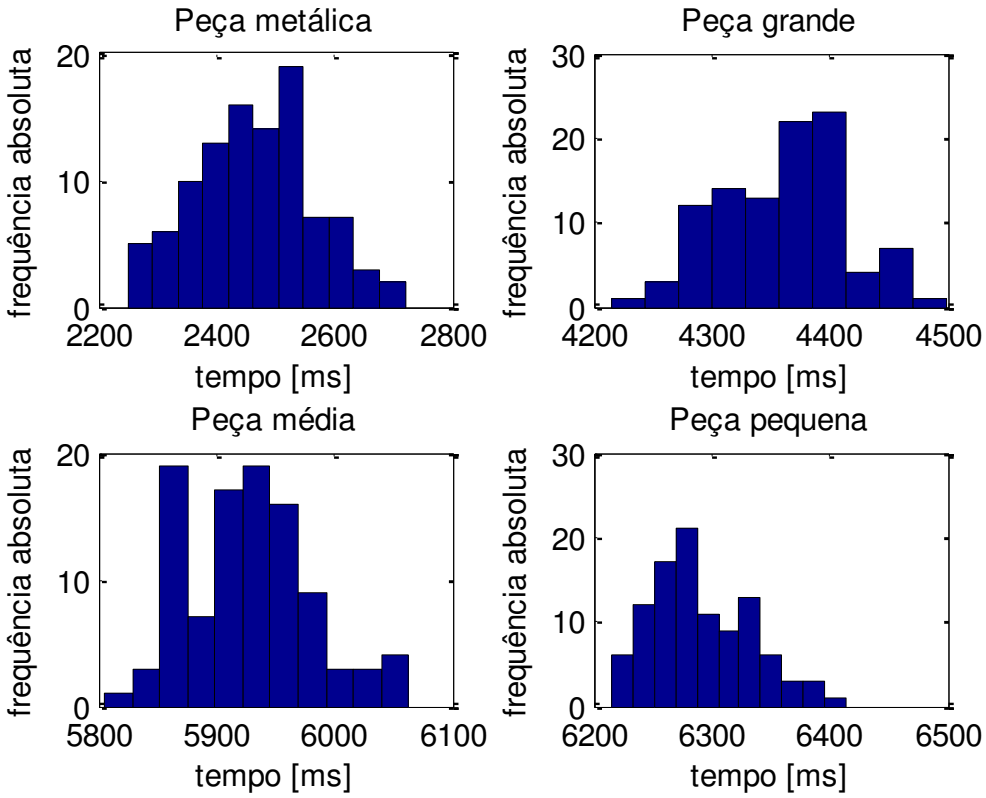


Figura 5.19. Histograma dos dados para o Sistema distribuído com comunicação I²C entre os dispositivos, sendo dois escravos.

Tabela 5.18. Dados estatísticos e teste de hipóteses para sistema distribuído com comunicação I²C entre dispositivos do ensaio 2.

Tipo de peça	***kurtosis***	***skewness***	**Hipótese nula: Distribuição normal**		
			K-S	**Lilliefors**	**J-B**
Metálica	2,6227	0,053764	Rejeita	Aceita	Aceita
Grande	2,7635	-0,04903	Rejeita	Aceita	Aceita
Média	2,711	0,33525	Rejeita	Aceita	Aceita
pequena	2,8888	0,54622	Rejeita	Aceita	Aceita

K-S: teste de Kolmogorov-Smirnov

J-B: teste de Jarque-Bera

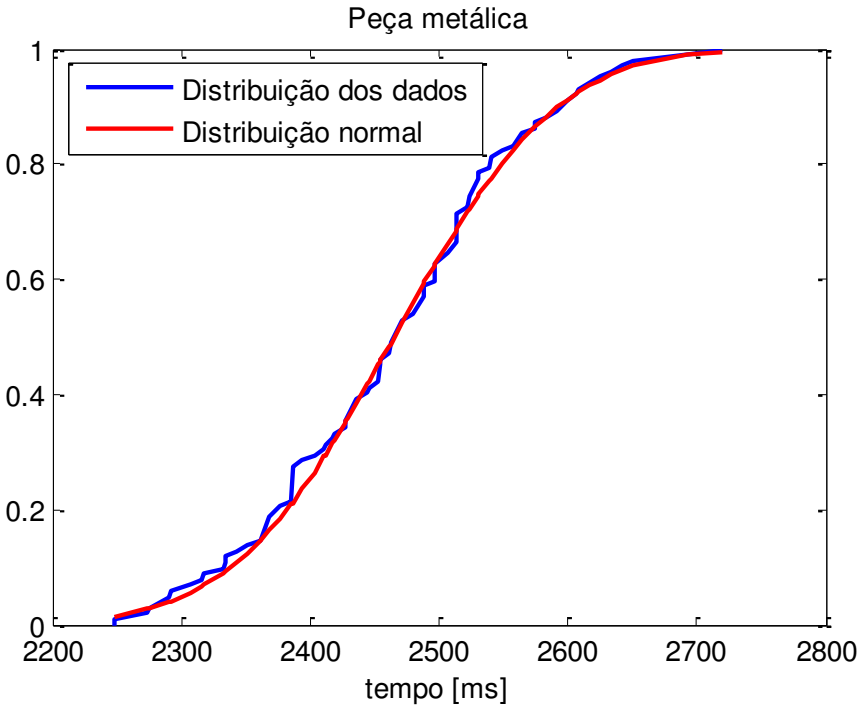


Figura 5.20. Função de distribuição acumulada para peça metálica do sistema distribuído com comunicação I²C entre dispositivos do ensaio 2.

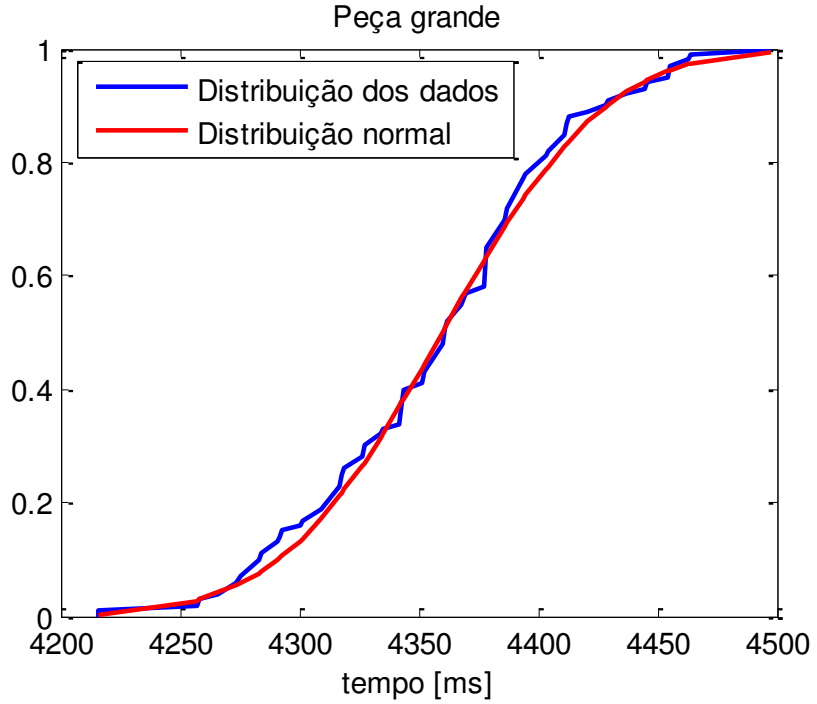


Figura 5.21. Função de distribuição acumulada para peça grande do sistema distribuído com comunicação I²C entre dispositivos do ensaio 2.

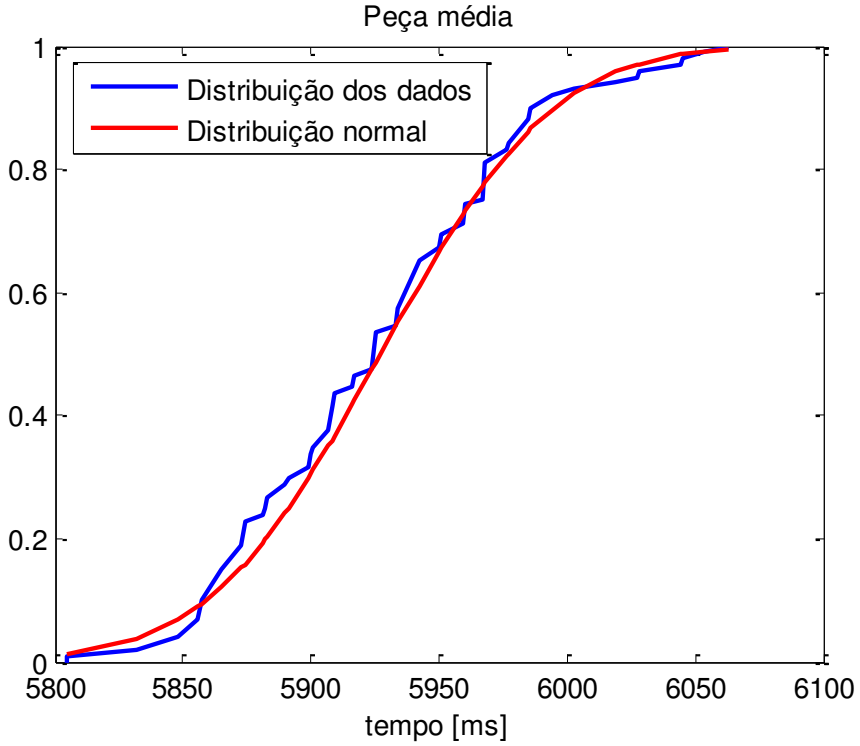


Figura 5.22. Função de distribuição acumulada para peça média do sistema distribuído com comunicação I²C entre dispositivos do ensaio 2.

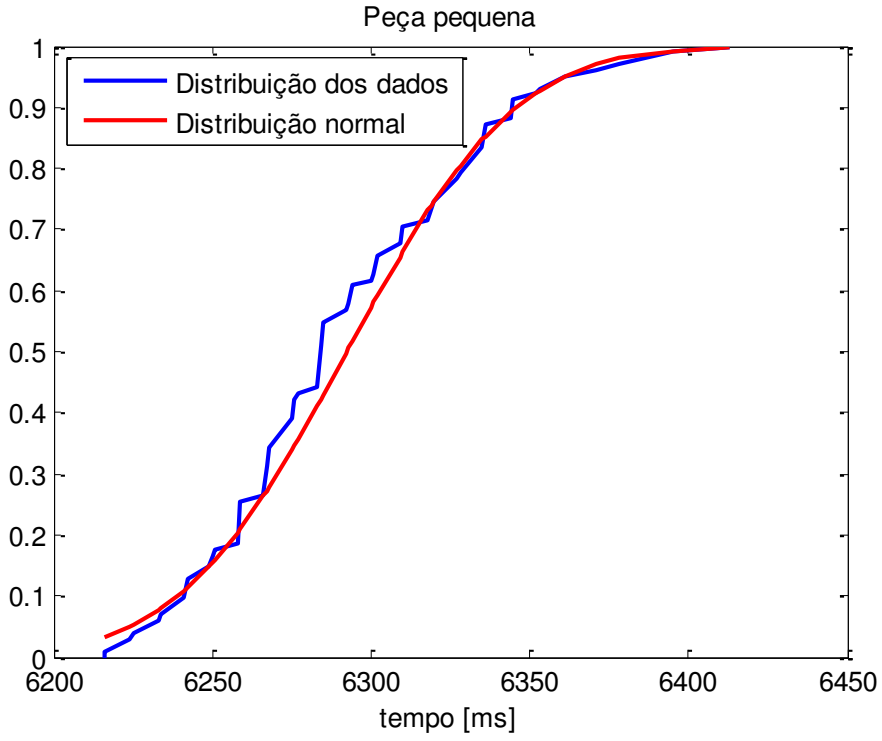


Figura 5.23. Função de distribuição acumulada para peça pequena do sistema distribuído com comunicação I²C entre dispositivos do ensaio 2.

5.3.3 Ensaio 3

Por fim o último ensaio deste protocolo de comunicação possui o mesmo custo do ensaio 2 (R$ 240,00), pois são utilizados dois *Arduino* Mega 2560 e um *Arduino* Uno. O sistema tem um total de 211 linhas de código (Apêndice B), sendo 74 dessas linhas exclusivas para a comunicação I²C. As características do ensaio estão apresentadas na Tab. 5.19.

Tabela 5.19. Características do ensaio de sistema distribuído com comunicação I²C entre dispositivos e lógica distribuída.

Item	**Quantidade**
Linhas de código total	211
Linhas de código do mestre	62
Linhas de código de comunicação do mestre	20
Linhas de código do escravo 1	73
Linhas de código de comunicação do escravo 1	27
Linhas de código do escravo 2	76
Linhas de código de comunicação do escravo 2	27
Microcontrolador *Arduino* Mega 2560	2
Microcontrolador *Arduino* Uno	1
Cabos para comunicação	8

Para este ensaio os dados estatísticos são apresentados na Tab. 5.20, enquanto que a Fig. 5.24 mostra o histograma destes dados. A Tab. 5.21 mostra que apenas o teste K-S não rejeita a hipótese nula, com isso as curvas CDFs apresentadas (Fig. 5.25 a Fig. 2.28) comprovam que os dados podem ser considerados como uma distribuição normal para uma significância de 5%.

Tabela 5.20. Dados estatísticos para Sistema distribuído com comunicação I²C entre dispositivos com a lógica de separação de peças distribuída.

Tipo de peça	**N**	$\overline{X}$ **[ms]**	σ **[ms]**	**IC de 95% para média [ms]**
Metálica	98	2466,3367	109,0485	$2252{,}6057 < \mu < 2680{,}0678$
Grande	102	4330,9608	71,4859	$4190{,}851 < \mu < 4471{,}0706$
Média	99	5965,9091	59,4226	$5849{,}443 < \mu < 6082{,}3752$
pequena	93	6274,5376	38,6806	$6198{,}7251 < \mu < 6350{,}3501$

N.: Número de amostras
$\overline{X}$: média amostral
σ : desvio-padrão amostral
IC.: intervalo de confiança

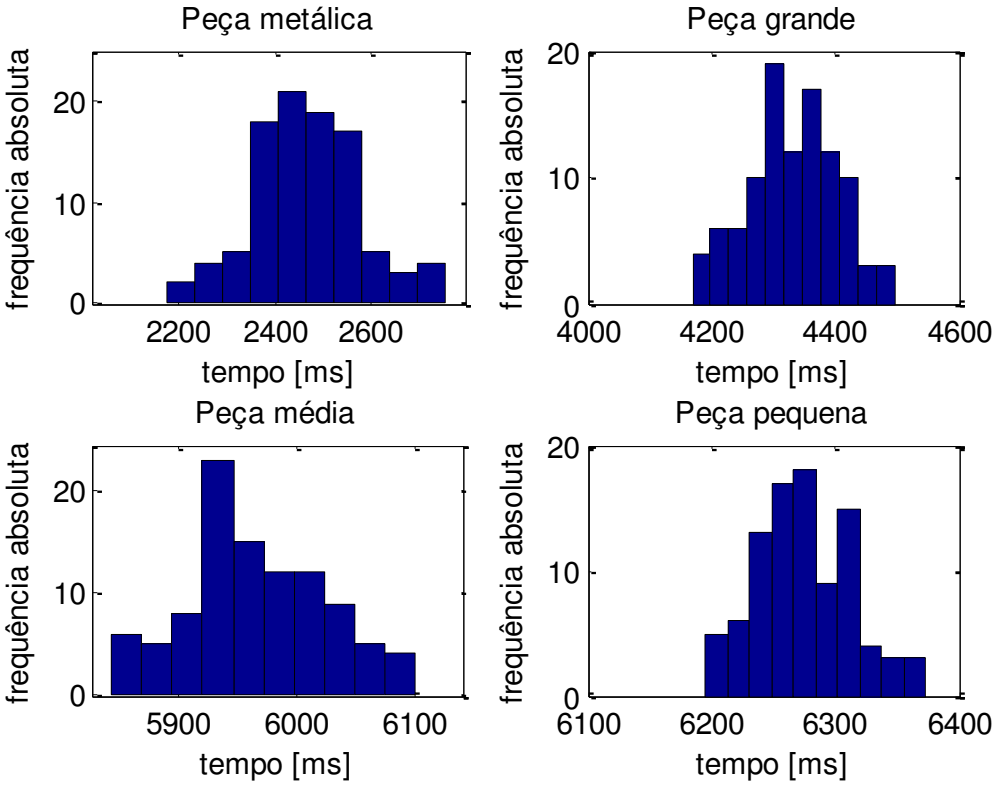


Figura 5.24. Histograma dos dados para sistema distribuído com comunicação I²C entre dispositivos com a lógica de separação de peças distribuída.

Tabela 5.21. Dados estatísticos e teste de hipóteses para sistema distribuído com comunicação I²C entre dispositivos do ensaio 3.

Tipo de peça	***kurtosis***	***skewness***	**Hipótese nula: Distribuição normal**		
			K-S	**Lilliefors**	**J-B**
Metálica	3,4025	0,18631	Rejeita	Aceita	Aceita
Grande	2,5691	-0,041079	Rejeita	Aceita	Aceita
Média	2,5895	0,1134	Rejeita	Aceita	Aceita
pequena	2,7783	0,28234	Rejeita	Aceita	Aceita

K-S: teste de Kolmogorov-Smirnov
J-B: teste de Jarque-Bera

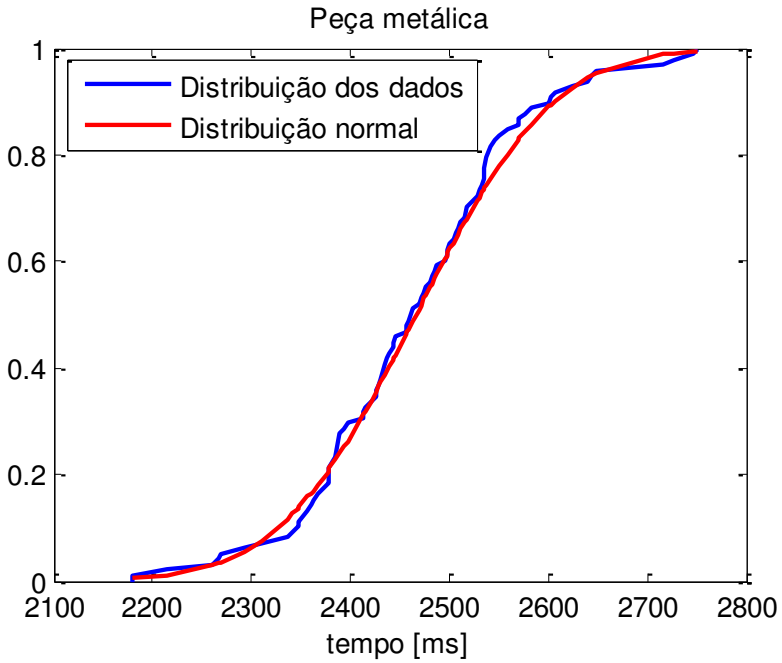


Figura 5.25. Função de distribuição acumulada para peça metálica do sistema distribuído com comunicação I²C entre dispositivos do ensaio 3.

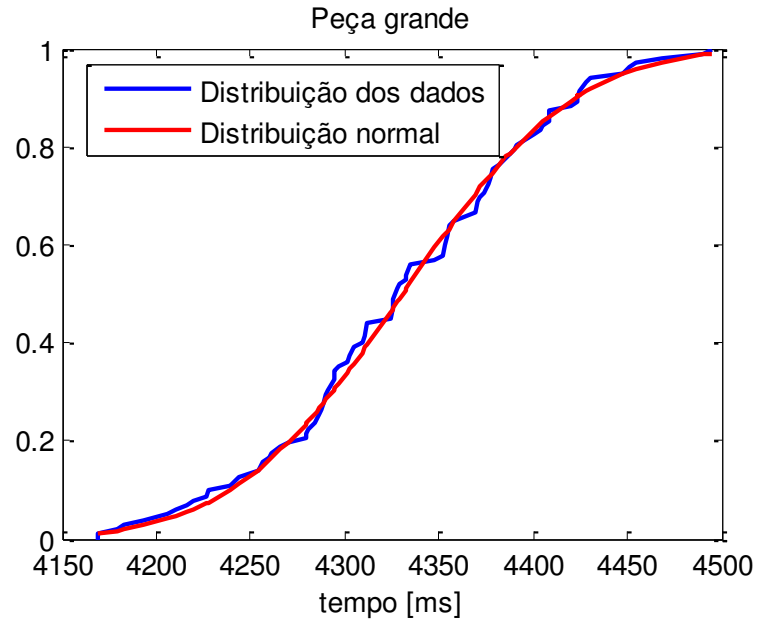


Figura 5.26. Função de distribuição acumulada para peça grande do sistema distribuído com comunicação I²C entre dispositivos do ensaio 3.

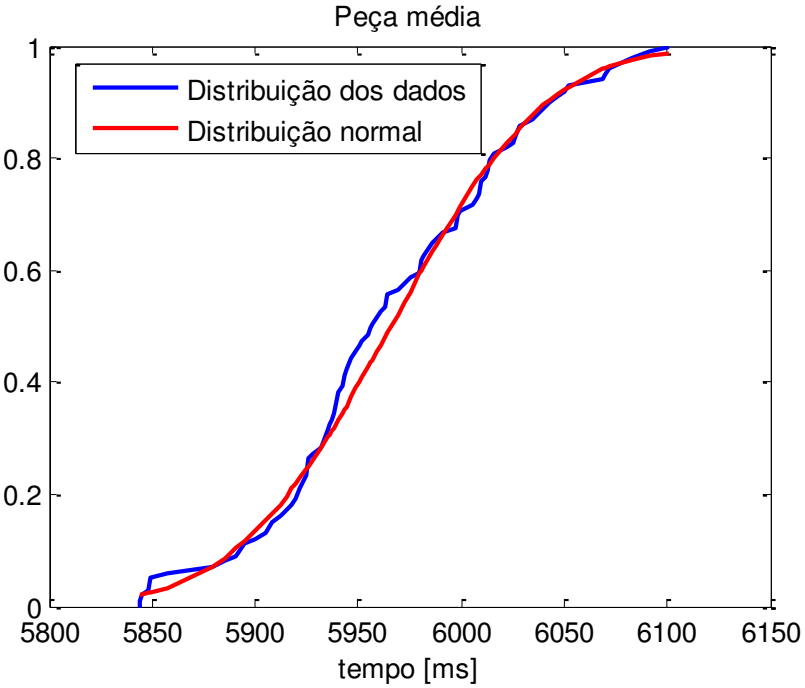


Figura 5.27. Função de distribuição acumulada para peça média do sistema distribuído com comunicação I²C entre dispositivos do ensaio 3.

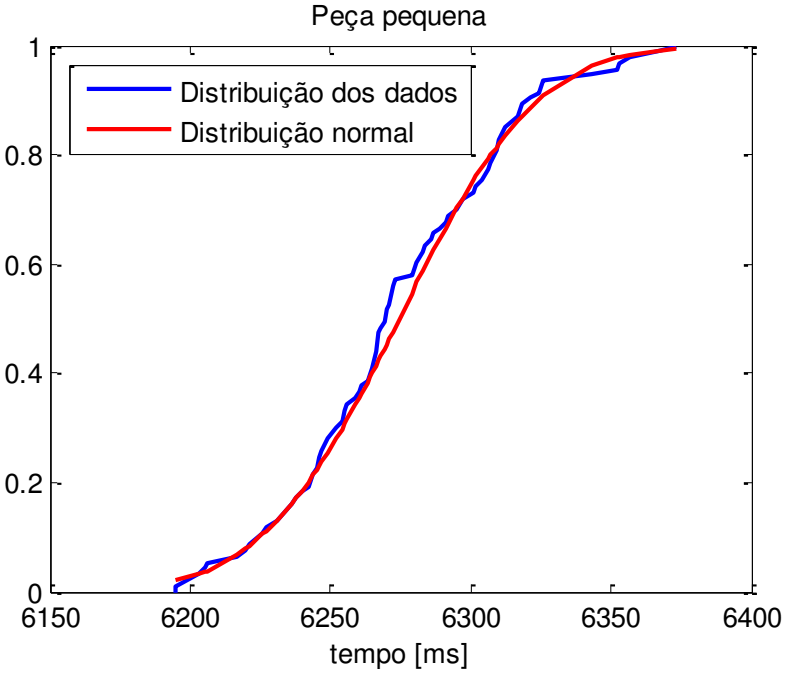


Figura 5.28. Função de distribuição acumulada para peça pequena do sistema distribuído com comunicação I²C entre dispositivos do ensaio 3.

5.4 Sistema distribuído com comunicação *Zigbee* entre dispositivos

Para utilizar o módulo de comunicação *Zigbee* no *Arduino* é necessário um módulo *Xbee* (R$ 160,00 cada), além de um módulo de encaixe (R$ 45,00 cada). Portanto o custo desta arquitetura de controle é elevado e atinge a quantia de R$ 590,00. Uma das vantagens da utilização desta arquitetura é o fato de não se utilizar cabos para comunicação e não haver necessidade de ligação dos terras. Para um sistema em que há uma longa distância entre os dispositivos ou que cabeamento entre esses dispositivos é inviável, esta arquitetura é indicada. O total de linhas de código utilizada é de 151 (Apêndice B), sendo 40 dessas linhas exclusivas para a comunicação entre os dispositivos. A Tab. 5.22 traz as características deste ensaio.

Os dados estatísticos são apresentados pela Tab. 5.23 enquanto que o histograma dos dados é mostrado pela Fig. 5.29. Note que para a peça média há dois picos distintos no histograma. Isto será refletido no teste de hipóteses, uma vez que claramente esses dados nãos seguem uma distribuição normal. A Tab. 5.24 traz os resultados dos testes de hipóteses que é rejeitado em todos os testes para a peça média e aceito para as demais peças apenas no teste de Jarque-Bera.

Tabela 5.22. Características do Sistema distribuído com comunicação *Zigbee* entre dispositivos.

Item	**Quantidade**
Linhas de código total	151
Linhas de código do mestre	70
Linhas de código de comunicação do mestre	18
Linhas de código do escravo	81
Linhas de código de comunicação do escravo	22
Microcontrolador *Arduino* Mega 2560	2
Módulo Xbee	2
Módulo de encaixe	2
Cabos para comunicação	-

Para validar esses resultados as curvas CDFs estão apresentadas nas Fig. 5.30 a Fig. 5.33. Note que para as três primeiras figuras as curvas funções de distribuição cumulativa são distintas, principalmente para a peça média, da curva função de distribuição cumulativa de uma distribuição normal. Para a peça pequena, como foram aceitos dois testes de hipótese e a CDF aproxima de uma distribuição normal a hipótese nula será aceita com um nível de significância de 5%.

Tabela 5.23. Dados estatísticos para Sistema distribuído com comunicação *Zigbee* entre dispositivos.

Tipo de peça	**N**	$\overline{X}$ **[ms]**	σ **[ms]**	**IC de 95% para média [ms]**
Metálica	105	2423,0667	136,1743	$2156{,}17 < \mu < 2689{,}9633$
Grande	105	4449,4762	123,3772	$4207{,}6614 < \mu < 4691{,}291$
Média	105	5810,9619	144,5396	$5527{,}6695 < \mu < 6094{,}2543$
pequena	119	6274,9832	47,3388	$6182{,}2008 < \mu < 6367{,}7656$

N.: Número de amostras
$\overline{X}$: média amostral
σ : desvio-padrão amostral
IC.: intervalo de confiança

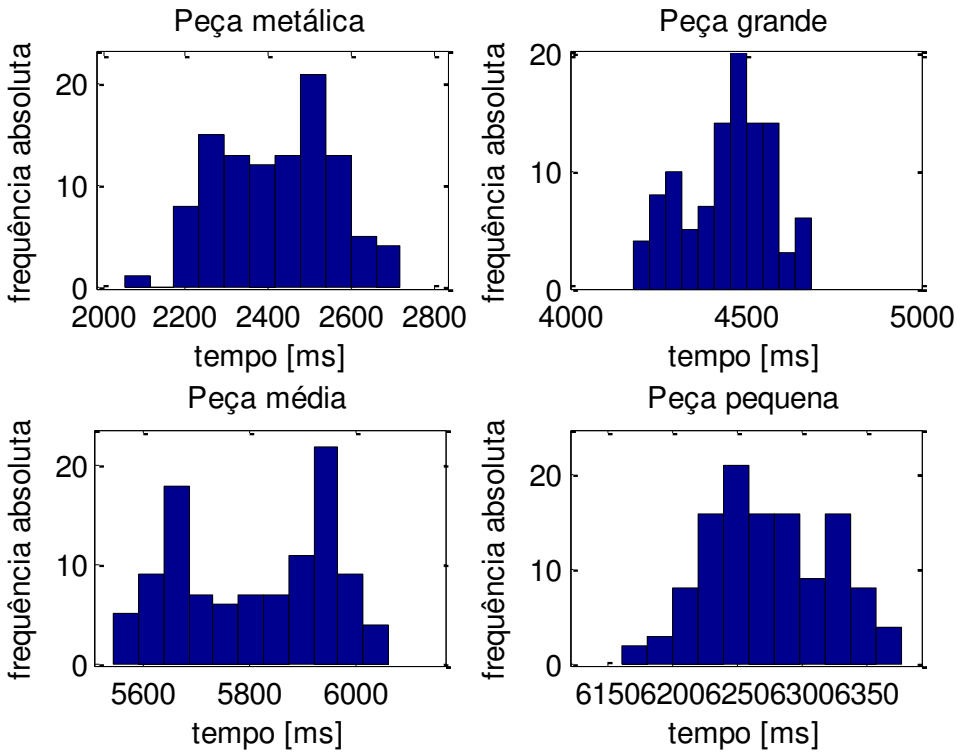


Figura 5.29. Histograma dos dados para sistema distribuído com comunicação *Zigbee* entre dispositivos.

Tabela 5.24. Dados estatísticos e teste de hipóteses para sistema distribuído com comunicação *Zigbee* entre dispositivos.

Tipo de peça	***kurtosis***	***skewness***	**Hipótese nula: Distribuição normal**		
			K-S	**Lilliefors**	**J-B**
Metálica	2,4049	-0,07311	Rejeita	Rejeita	Aceita
Grande	2,3417	-0,30244	Rejeita	Rejeita	Aceita
Média	1,5991	-0,14897	Rejeita	Rejeita	Rejeita
pequena	2,4088	0,1077	Rejeita	Aceita	Aceita

K-S: teste de Kolmogorov-Smirnov

J-B: teste de Jarque-Bera

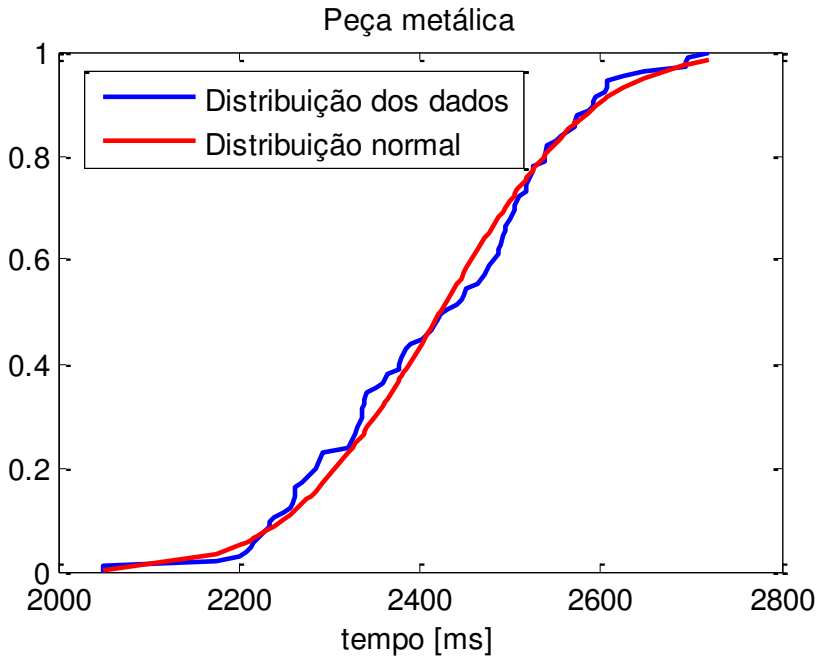


Figura 5.30. Função de distribuição acumulada para peça metálica do sistema distribuído com comunicação *Zigbee* entre dispositivos.

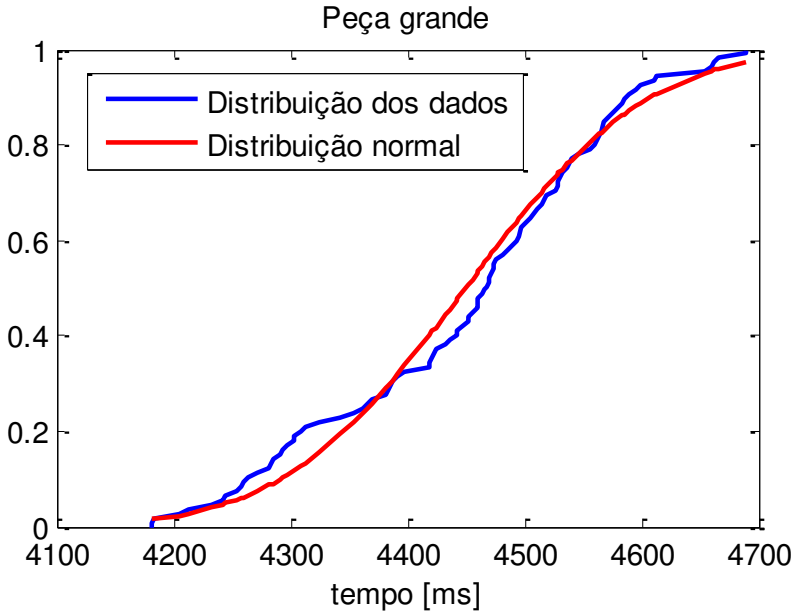


Figura 5.31. Função de distribuição acumulada para peça grande do sistema distribuído com comunicação *Zigbee* entre dispositivos.

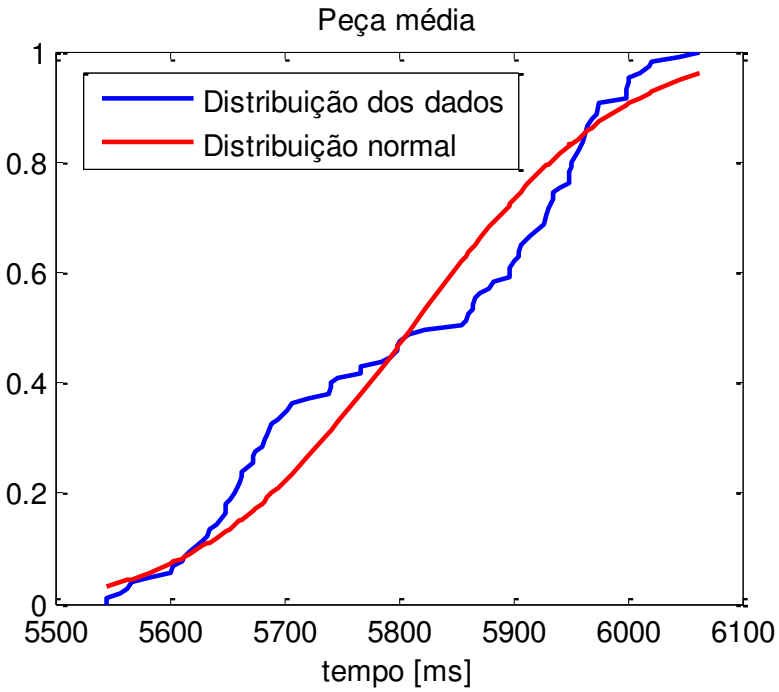


Figura 5.32. Função de distribuição acumulada para peça média do sistema distribuído com comunicação *Zigbee* entre dispositivos.

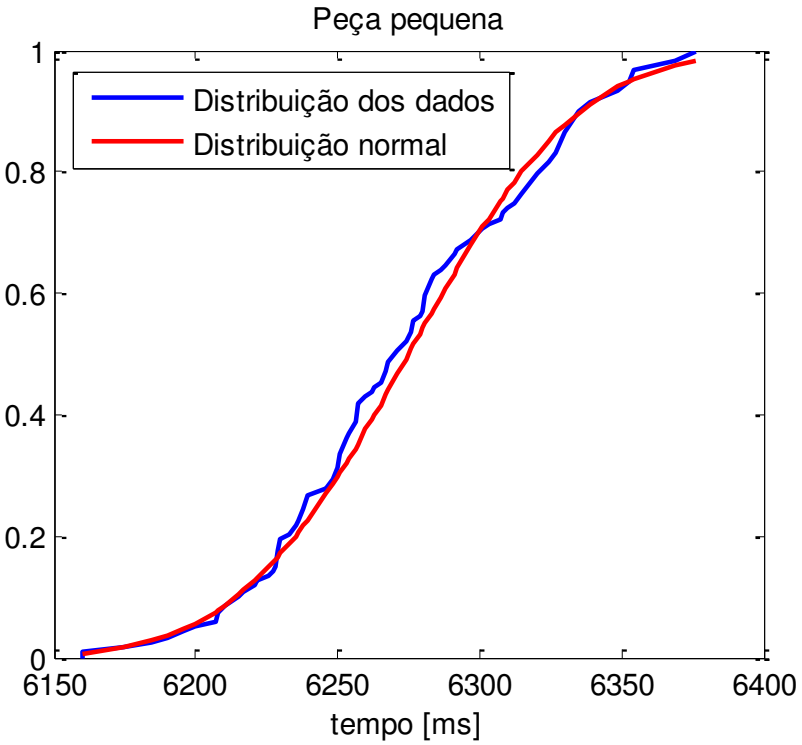


Figura 5.33. Função de distribuição acumulada para peça pequena do sistema distribuído com comunicação *Zigbee* entre dispositivos.

5.5 Comparação entre os ensaios

A comparação entre os ensaios se dá pela apresentação dos intervalos de confiança para os quatro tipos de peças entre algumas topologias. Inicialmente será comparado o sistema distribuído com comunicação serial *RxTx* entre dispositivos para os quatro ensaios desta arquitetura. Em seguida a mesma comparação é feita entre o sistema distribuído com comunicação I²C entre dispositivos para os três ensaios desta arquitetura.

5.5.1 Sistema distribuído com comunicação serial RxTx entre dispositivos.

Os quatro ensaios desta arquitetura de controle foram apresentados na Tab. 4.4. A Fig. 5.34 apresenta os intervalos de confiança para a peça metálica. Note que para os três primeiros ensaios esses valores estão próximos, quanto que para o ensaio 4 o intervalo é consideravelmente elevado. Isto ocorre uma vez que neste ensaio o tempo de solicitação de mensagens é de 800 ms, e, como foi mostrado no histograma do mesmo, há apenas dois valores para o tempo de separação das peças e estes valores distam um do outro de aproximadamente 800 ms. Com isso o desvio-padrão para este ensaio é grande, o que torna o intervalo de confiança grande. O mesmo ocorre para as peças grandes (Fig. 5.35) e médias (Fig. 5.36). Já para as peças pequenas, como mostra a Fig. 5.37, o intervalo de confiança do quarto ensaio é apenas um ponto, já que o desvio-padrão é próximo de 0, uma vez que os valores do tempo de separação das peças foram de 6399, 6400 e 6401 ms.

Para fazer uma inferência sobre a diferença das médias tomando dois ensaios, deve-se inicialmente verificar se as populações são homocedásticas ou heterocedásticas. Portanto o teste F é realizado, conforme apresentado no Apêndice A. A Tab. 5.25 apresenta os resultados do teste comparando os ensaios 1, 2 e 3 tomados dois a dois. O ensaio 4 não foi considerado por não se tratar de uma distribuição normal. Já a Tab. 5.26 mostra o intervalo de confiança para a diferença entre as médias. Para este caso a hipótese nula é de que as médias são iguais com 95% de confiança. Note que a maioria das comparações recusaram a hipótese nula.

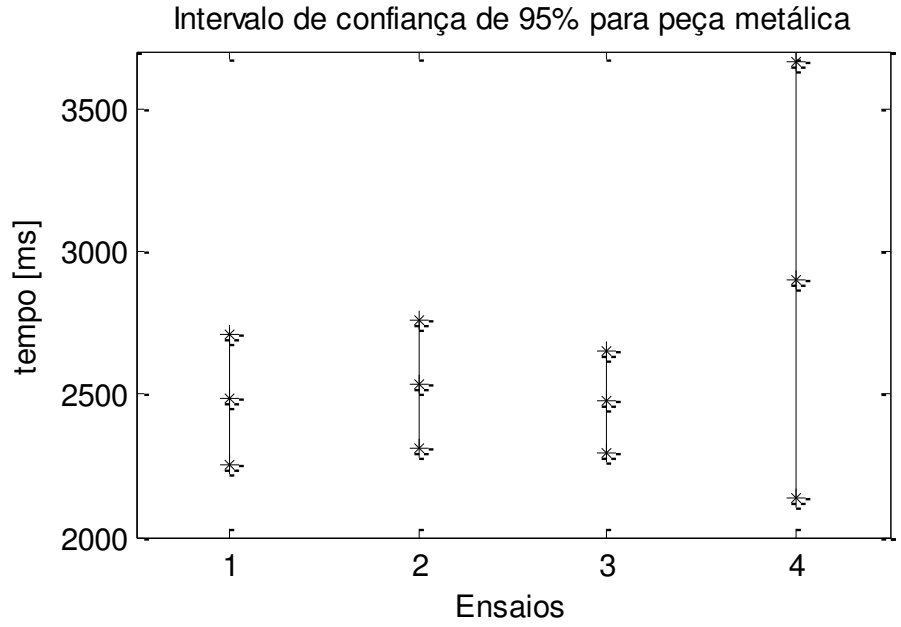


Figura 5.34. Intervalo de confiança para peça metálica no sistema distribuído com comunicação *RxTx* entre dispositivos.

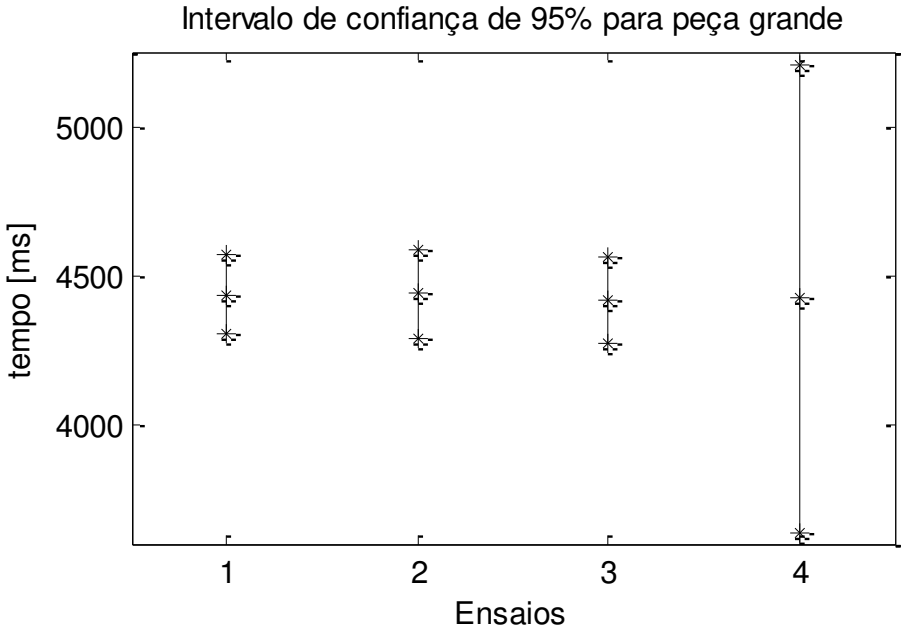


Figura 5.35. Intervalo de confiança para peça grande no sistema distribuído com comunicação *RxTx* entre dispositivos.

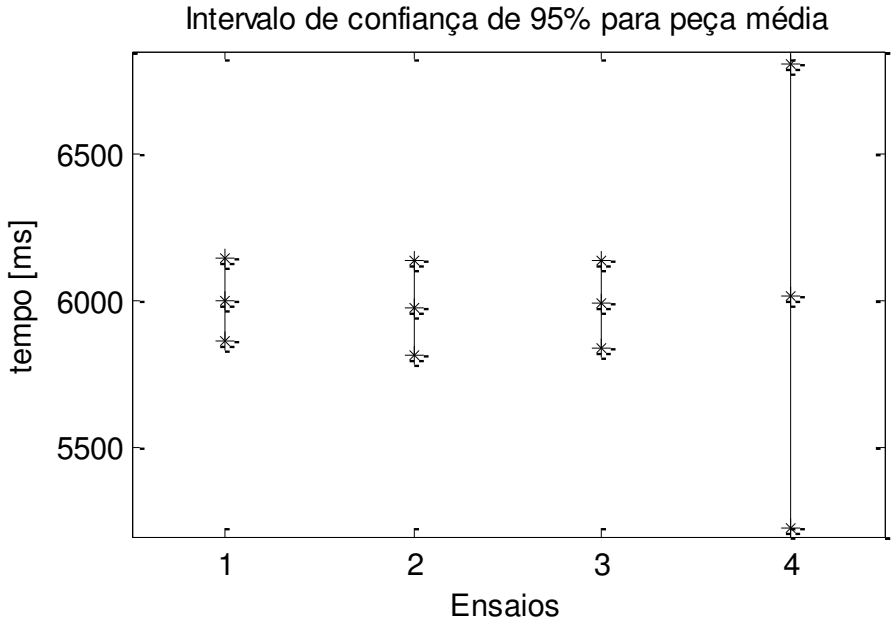


Figura 5.36. Intervalo de confiança para peça média no sistema distribuído com comunicação *RxTx* entre dispositivos.

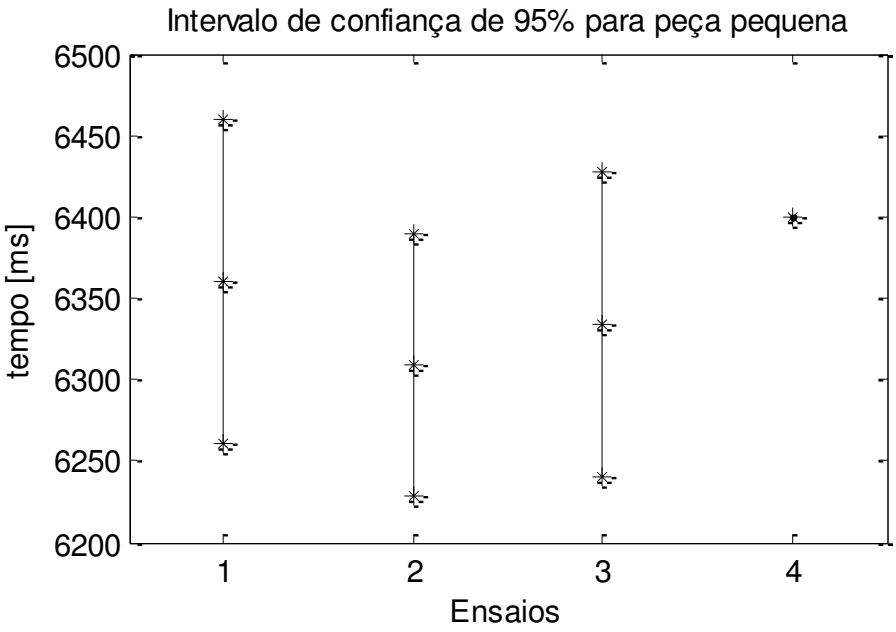


Figura 5.37. Intervalo de confiança para peça pequena no sistema distribuído com comunicação *RxTx* entre dispositivos.

Tabela 5.25. Teste F para o Sistema distribuído com comunicação serial *RxTx* entre dispositivos.

Tipo de peça	Ensaios		Hipótese nula: homocedasticidade
metálica	1	2	Aceita
	1	3	Rejeita
	2	3	Rejeita
grande	1	2	Aceita
	1	3	Aceita
	2	3	Aceita
média	1	2	Aceita
	1	3	Aceita
	2	3	Aceita
pequena	1	2	Aceita
	1	3	Aceita
	2	3	Aceita

Tabela 5.26. Inferência sobre a diferença entre as medias para Sistema distribuído com comunicação serial *RxTx* entre dispositivos.

Tipo	Ensaio		IC de 95% para a diferença entre as médias [ms]	Hipótese nula: $\mu_a = \mu_b$
metálica	1	2	- 52,6335 $< \mu_a - \mu_b <$ - 45,2627	Rejeita
	1	3	- 16,2558 $< \mu_a - \mu_b <$ 35,5376	Aceita
	2	3	33,6414 $< \mu_a - \mu_b <$ 83,5366	Rejeita
grande	1	2	- 7,3386 $< \mu_a - \mu_b <$ 2,737	Aceita
	1	3	16,4909 $< \mu_a - \mu_b <$ 26,1261	Rejeita
	2	3	22,2794 $< \mu_a - \mu_b <$ 24,9392	Rejeita
média	1	2	23,4767 $< \mu_a - \mu_b <$ 33,9569	Rejeita
	1	3	11,5878 $< \mu_a - \mu_b <$ 15,109	Rejeita
	2	3	- 20,5599 $< \mu_a - \mu_b <$ - 10,1769	Rejeita
pequena	1	2	45,6126 $< \mu_a - \mu_b <$ 56,67	Rejeita
	1	3	22,3245 $< \mu_a - \mu_b <$ 28,7539	Rejeita
	2	3	- 30,0918 $< \mu_a - \mu_b <$ - 21,1124	Rejeita

5.5.2 Sistema distribuído com comunicação I²C entre dispositivos.

Os três ensaios desta arquitetura de controle são apresentados na Tab. 5.27. As quatro figuras seguintes (Fig. 5.38 a Fig. 5.41) apresentam os intervalos de confiança da média para as peças metálica, grande, média e pequena, respectivamente.

A inferência sobre a diferença das médias é apresentada na Tab. 5.28. Para os ensaios 1 e 2 apenas para as peças médias as populações não são homocedásticas. Para os ensaios 2 e 3 o mesmo ocorre, porém, agora para as peças grandes. Os resultados apresentados na Tab. 5.29 mostram que para as peças metálicas pode-se afirmar com 95% de confiança que as médias para os três ensaios são as mesmas. Para os outros tipos de peças não se pode fazer a mesma afirmação.

Tabela 5.27. Característica dos ensaios para Sistema distribuído com comunicação I²C entre dispositivos

Ensaio	**Característica**
1	2 dispositivos
2	3 dispositivos
3	3 dispositivos com lógica distribuída

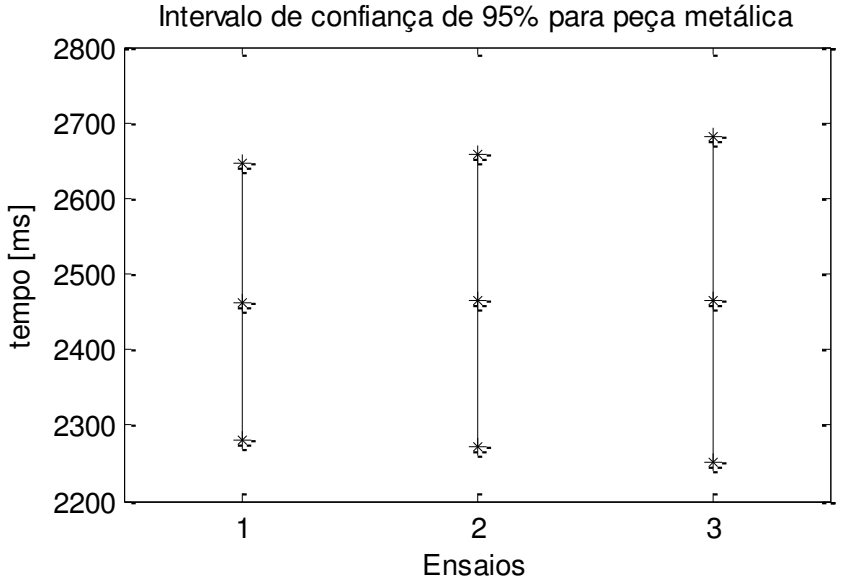


Figura 5.38. Intervalo de confiança para peça metálica no sistema distribuído com comunicação I²C entre dispositivos.

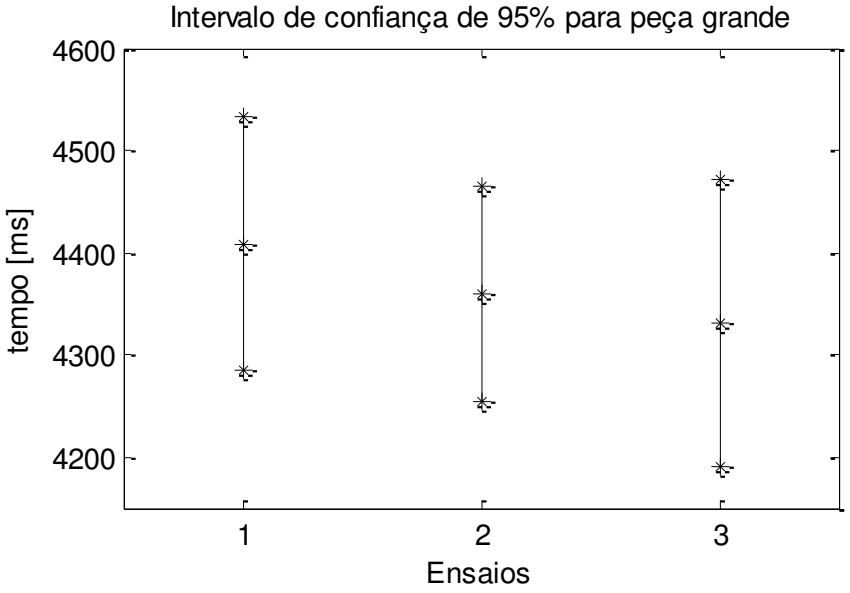


Figura 5.39. Intervalo de confiança para peça grande no sistema distribuído com comunicação I²C entre dispositivos.

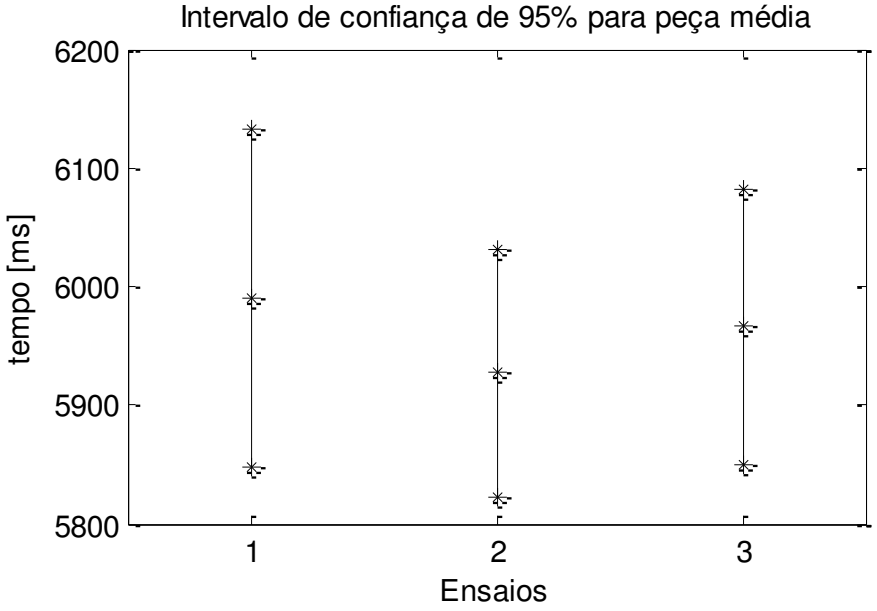


Figura 5.40. Intervalo de confiança para peça média no sistema distribuído com comunicação I²C entre dispositivos.

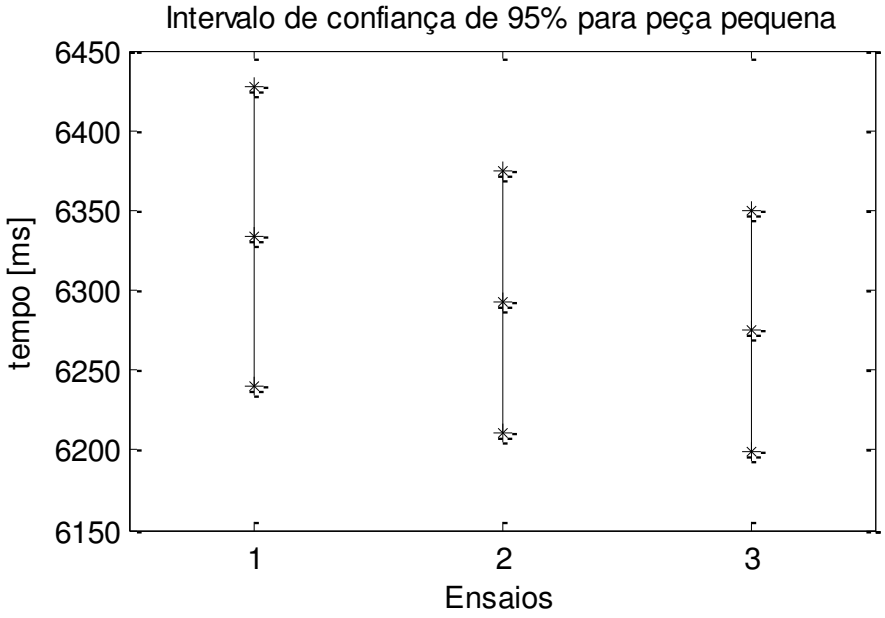


Figura 5.41. Intervalo de confiança para peça pequena no sistema distribuído com comunicação I²C entre dispositivos.

Tabela 5.28. Teste F para o Sistema distribuído com comunicação I²C entre dispositivos.

Tipo de peça	**Ensaios**		**Hipótese nula: homocedasticidade**
metálica	1	2	Aceita
	1	3	Aceita
	2	3	Aceita
grande	1	2	Aceita
	1	3	Aceita
	2	3	Rejeita
média	1	2	Rejeita
	1	3	Aceita
	2	3	Aceita
pequena	1	2	Aceita
	1	3	Rejeita
	2	3	Aceita

Tabela 5.29. Inferência sobre a diferença entre as medias para Sistema distribuído com comunicação I²C entre dispositivos.

Tipo	**Ensaio**		**IC de 95% para a diferença entre as médias**	**Hipótese nula: $\mu_a = \mu_b$**
metálica	1	2	- 8,4433 $< \mu_a - \mu_b <$ 3,5315	Aceita
	1	3	- 12,7401 $< \mu_a - \mu_b <$ 5,5667	Aceita
	2	3	- 7,8789 $< \mu_a - \mu_b <$ 5,6173	Aceita
grande	1	2	43,3098 $< \mu_a - \mu_b <$ 54,3242	Rejeita
	1	3	72,0189 $< \mu_a - \mu_b <$ 83,4535	Rejeita
	2	3	14,2126 $< \mu_a - \mu_b <$ 43,6258	Rejeita
média	1	2	48,6838 $< \mu_a - \mu_b <$ 78,1378	Rejeita
	1	3	17,5186 $< \mu_a - \mu_b <$ 31,6828	Rejeita
	2	3	- 43,0037 $< \mu_a - \mu_b <$ - 34,6165	Rejeita
pequena	1	2	37,5136$< \mu_a - \mu_b <$ 45,0484	Rejeita
	1	3	48,8346 $< \mu_a - \mu_b <$ 69,3972	Rejeita
	2	3	14,5278 $< \mu_a - \mu_b <$ 21,142	Rejeita

5.5.3 Comparação entre as arquiteturas de controle

Quando analisamos os custos relativos aos equipamentos (microcontroladores e módulos externos) das arquiteturas de controle, percebemos um aumento à medida que o sistema se torna mais distribuído, conforme mostra a Tab. 5.30. Note que para implementar um sistema distribuído com comunicação *Zigbee* entre dispositivos o valor é cerca de 550% maior que de um sistema centralizado. Porém, nem sempre há a possibilidade de se utilizar essa arquitetura mais barata.

Tabela 5.30. Custo dos dispositivos as arquiteturas de controle.

Arquitetura	Custo (R$)
Centralizado	90,00
Distribuído com comunicação serial *RxTx* entre dispositivos	180,00
Distribuído com comunicação I²C entre dois dispositivos	180,00
Distribuído com comunicação I²C entre três dispositivos	240,00
Distribuído com comunicação I²C entre dispositivos e lógica distribuída	240,00
Distribuído com comunicação *Zigbee* entre dispositivos	590,00

Agora, comparando o sistema centralizado a outras duas arquiteturas de controle (com comunicação serial *RxTx* – ensaio 1 e com comunicação I²C – ensaio 1) obtemos os intervalos de confiança para a média como mostram as figuras Fig. 5.42 a Fig. 5.45. A arquitetura distribuída com comunicação *Zigbee* não foi inserida nesta comparação por não se tratar de uma distribuição normal. Ao compararmos os ensaios pelo teste F (Tab. 5.31) nota-se que em alguns casos as populações podem ser consideradas homocedásticas, e em outros são consideradas heterocedásticas. Com isso a Tab. 5.32 mostra os intervalos de confiança para a diferença entre as médias e o resultado para a hipótese de que as médias são as mesmas.

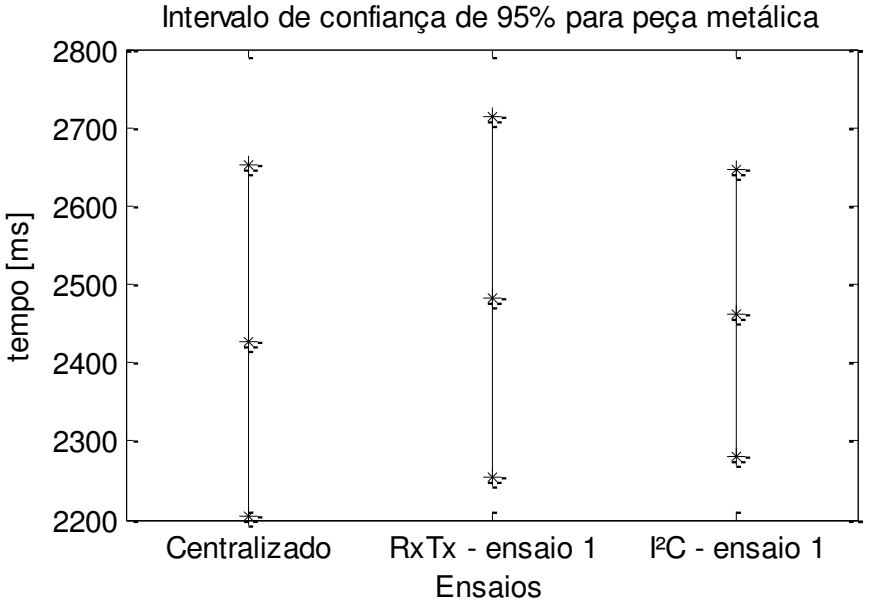


Figura 5.42. Intervalo de confiança para peça metálica nas arquiteturas de controle.

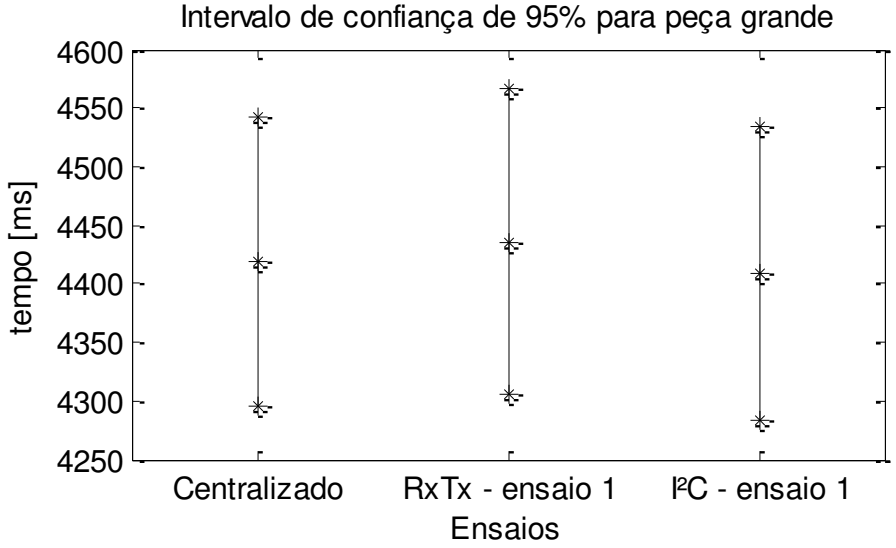


Figura 5.43. Intervalo de confiança para peça grande nas arquiteturas de controle.

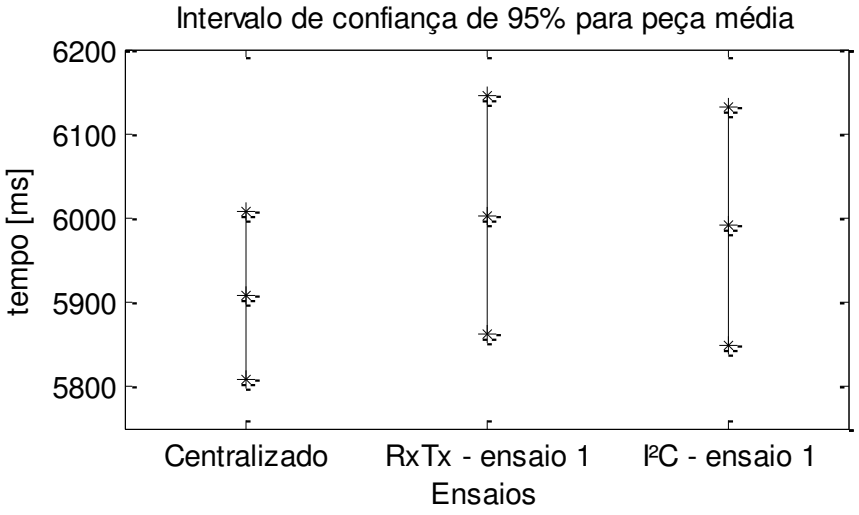


Figura 5.44. Intervalo de confiança para peça média nas arquiteturas de controle.

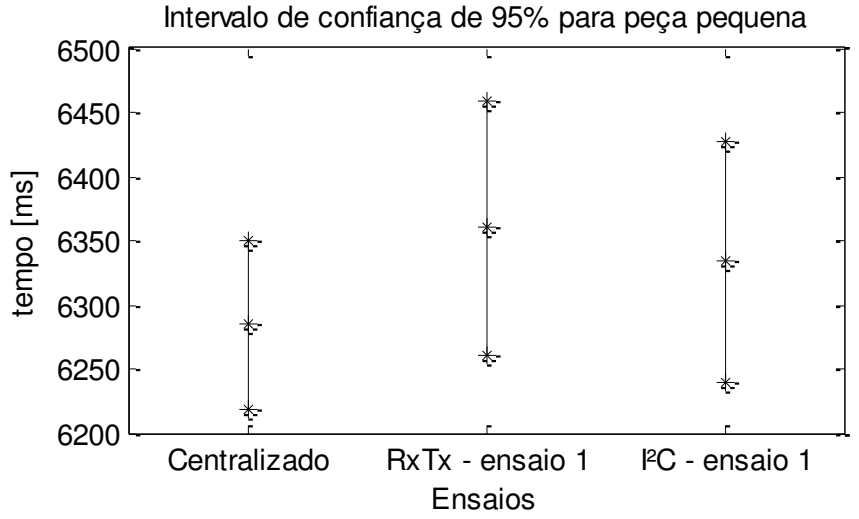


Figura 5.45. Intervalo de confiança para peça pequena nas arquiteturas de controle.

Tabela 5.31. Teste F para as arquiteturas de controle.

Tipo de peça	Ensaios		Hipótese nula: homocedasticidade
metálica	centralizado	*RxTx*	Aceita
	centralizado	I²C	Rejeita
	RxTx	I²C	Rejeita
grande	centralizado	*RxTx*	Aceita
	centralizado	I²C	Aceita
	RxTx	I²C	Aceita
média	centralizado	*RxTx*	Rejeita
	centralizado	I²C	Rejeita
	RxTx	I²C	Aceita
pequena	centralizado	*RxTx*	Rejeita
	centralizado	I²C	Rejeita
	RxTx	I²C	Aceita

Tabela 5.32. Inferência sobre a diferença entre as médias para as arquiteturas de controle.

Tipo	Ensaio		IC de 95% para a diferença entre as médias	Hipótese nula: $\mu_a = \mu_b$
metálica	centralizado	*RxTx*	- 56,2954 $< \mu_a - \mu_b <$ - 53,0978	Rejeita
	centralizado	I²C	- 59,1064 $< \mu_a - \mu_b <$ - 10,2118	Rejeita
	RxTx	I²C	- 5,1632 $< \mu_a - \mu_b <$ 45,2376	Aceita
grande	centralizado	*RxTx*	- 21,1743 $< \mu_a - \mu_b <$ - 12,0355	Rejeita
	centralizado	I²C	7,9969 $< \mu_a - \mu_b <$ 12,5061	Rejeita
	RxTx	I²C	22,898 $< \mu_a - \mu_b <$ 30,8148	Rejeita
média	centralizado	*RxTx*	- 107,5313 $< \mu_a - \mu_b <$ - 78,6331	Rejeita
	centralizado	I²C	- 95,4475 $< \mu_a - \mu_b <$ - 66,1555	Rejeita
	RxTx	I²C	10,7459 $< \mu_a - \mu_b <$ 13,8155	Rejeita
pequena	centralizado	*RxTx*	- 85,4291 $< \mu_a - \mu_b <$ - 65,1883	Rejeita
	centralizado	I²C	- 58,8334 $< \mu_a - \mu_b <$ - 39,346	Rejeita
	RxTx	I²C	23,2608 $< \mu_a - \mu_b <$ 29,1772	Rejeita

CAPÍTULO VI

DISCUSSÃO E CONCLUSÃO

A proposta desta dissertação era de avaliar o impacto de projetos de redes de dispositivos centralizados e distribuídos aplicados em sistemas a evento discreto de baixa criticidade e baixo custo. Para tal, como apresentado no capítulo de resultados e discussões, há alguns pontos importantes para a tomada de decisão de qual arquitetura deve ser escolhida para um determinado sistema. Porém, é importante ressaltar que, com apenas um estudo de caso não há possibilidades de se afirmar qual o tipo adequado da arquitetura de controle.

A arquitetura de controle centralizada pode ser adota quando não há a necessidade clara da utilização de mais de um dispositivo. Com isso o projeto tem um custo final consideravelmente menor que nas outras arquiteturas, o que faz com que a empresa possa adquirir mais de um equipamento, mantendo um deles como reserva, caso haja algum problema no dispositivo. O resultado disso é um baixo tempo de parada para manutenção ou troca de equipamento na empresa, o que gera uma maior produtividade, porém terá custos adicionais se precisar expandir o sistema.

Uma arquitetura de controle distribuída pode ser adotada quando apenas um dispositivo possui um número de entradas ou saídas menor que a solicitada em um sistema, ou quando há a necessidade de manter apenas algumas funções separadas do restante do sistema. Vale ressaltar que caso uma empresa adote por este tipo de arquitetura ela deve estar disposta a pagar um valor acima do centralizado, uma vez que são utilizados mais dispositivos.

Uma vez optada pelo sistema distribuído várias formas de comunicação podem ser implementadas, portanto há casos que é melhor optar por um tipo de comunicação ou por

outro tipo. Um exemplo disso é quando se decide utilizar dispositivos que possuem apenas um canal de comunicação serial *RxTx*. Caso haja três dispositivos, sendo que dois deles decidem comunicar com um dispositivo, o próprio programador deve realizar o controle das mensagens recebidas de forma que o programa consiga diferenciar os remetentes. Quando se opta pela comunicação por barramento I²C, o próprio protocolo realiza essa verificação, uma vez que cada dispositivo tem um endereço fixo.

Um exemplo da utilização da comunicação com *Zigbee* está relacionada à distância entre os dispositivos e a possibilidade de cabeamento entre os dois. Imagine que a identificação da peça, como no estudo de caso, ocorresse em um setor da empresa e a separação da peça em outro setor. Se o caminho entre os dois setores possuir algum motivo que impossibilita a passagem de um cabo não há como os dois dispositivos se comunicarem de forma serial ou via barramento I²C. Portanto a empresa deve optar pela utilização da comunicação com sem fio. De fato, isto implicará em um alto custo de instalação dos dispositivos, uma vez que, esta tecnologia ainda é recente e cara. Outro exemplo da utilização do *Zigbee* é quando há a comunicação entre dispositivos de níveis de tensão diferentes. Imagine um subsistema trabalhando a 220 V e um subsistema trabalhando a 12 V. Como este tipo de comunicação não necessita a equiparação do aterramento dos dispositivos, não há necessidade de aterramento dos subsistemas, o que economizaria significativamente para a empresa.

A comparação entre as arquiteturas de controle se deu a partir do intervalo de confiança para a diferença entre as médias dos tempos tomando dois experimentos por vez. A hipótese nula (H_0) é de que as duas médias são iguais, portanto o intervalo de confiança serve para aceitar ou rejeitar essa hipótese. Quando o valor zero está compreendido neste intervalo aceitamos a hipótese nula e dizemos com um certo nível de significância que as médias são iguais, ou seja, a arquitetura escolhida não influencia no tempo de separação da peça. Caso a hipótese seja rejeitada, duas situações podem ser analisadas. Pelas equações Eq. 6 e Eq. 8 apresentadas no Apêndice A a diferença entre as médias é feita como uma diferencia entre as médias $\mu_a - \mu_b$, ou seja, quando este intervalo é inteiramente positivo podemos afirmar que a média μ_a é maior que a média μ_b. Quando o intervalo é completamente negativos afirmamos que $\mu_b > \mu_a$.

Sendo assim, ao compararmos os ensaios 1, 2 e 3 da arquitetura de controle distribuído com comunicação serial *RxTx* entre os dispositivos podemos afirmar com 5% de significância que a taxa de transferência de dados e o tempo de solicitação de mensagens influenciam no tempo de separação das peças.

Quando comparamos os ensaios 1, 2 e 3 da arquitetura de controle distribuído com comunicação I²C entre dispositivos podemos afirmar com 95% de confiança que para as

peças metálicas não há influência da arquitetura utilizada. Porém para os outros tipos de peça há diferença entre as médias. Ao compararmos o ensaio 1 com os ensaios 2 e 3, podemos dizer que o primeiro ensaio possui um tempo médio de separação das peças grande, média e pequena maior que o tempo de separação das peças para os outros dois ensaios. Um dos fatores que levam a essa afirmação é o fato de o quadro da mensagem trocada no primeiro ensaio é maior que o quadro das mensagens trocas dos ensaios restantes. Ao compararmos os dois ensaios com três dispositivos (ensaios 2 e 3) vemos que, para as peças médias, o ensaio 2 possui um menor tempo médio de separação das peças, enquanto que para as peças grandes e pequenas o ensaio três é o que possui um menor tempo de separação das peças.

Ao compararmos o sistema centralizado com o ensaio 1 da arquitetura de controle distribuída com comunicação serial *RxTx* observamos que a hipótese nula foi rejeitada para os quatro tipos de peça. Como este teste foi realizado com a diferença entre a média do sistema centralizado e a média do distribuído e todos os intervalos de confiança deram estritamente negativos, pode-se afirmar com 95% de confiança que a média do tempo de separação em um sistema centralizado é menor que a média do tempo de separação em um sistema distribuído com comunicação serial *RxTx* quando a taxa de transmissão de dados é de 115200 bps e o tempo de solicitação de mensagens é de 4 ms.

Ao compararmos o sistema centralizado e o ensaio 1 da arquitetura de controle distribuída com comunicação I²C observamos com 95% de confiança que para as peças metálicas, médias e pequenas a média do tempo de separação do sistema centralizado é menor que a média de separação do sistema distribuído. Já para a peça grande a média do sistema distribuído é menor que a do sistema centralizado. Isto ocorre, pois, a média do sistema centralizado é maior que a média do sistema distribuído com comunicação I²C e o desvio-padrão amostral para os dois casos são próximos (aproximadamente 62 para o sistema centralizado e 63 para o I²C).

Ao compararmos os dois sistemas distribuídos para a peça metálica pode-se afirmar com 95% de confiança que os tempos médios de separação das peças são os mesmos. Porém para os outros tipos de peça, o sistema distribuído com comunicação I²C possui um tempo médio inferior ao tempo do sistema distribuído com comunicação serial *RxTx*.

Por fim, apesar de ser comprovado a influência entre as distintas arquiteturas de controle para grande parte das comparações, o valor máximo do intervalo de confiança foi de aproximadamente 50 ms. Ou seja, em sistemas de baixa criticidade, como o estudo de caso, não há obrigatoriedade de se optar por uma arquitetura em relação a outra e esta escolha fica a critério do projetista.

O trabalho apresentou uma análise sobre as arquiteturas de controle distribuído utilizando três distintos tipos de comunicação. Vale ressaltar que a heterogeneidade de

comunicação em um projeto de automação é comum, portanto há possibilidade de utilização de diversas formas de comunicação em um mesmo sistema. Por exemplo, um microcontrolador *Arduino*, pode se comunicar utilizando as três formas de comunicação apresentadas (serial *RxTx*, I²C e *Zigbee*). Porém, sabe-se que o *Zigbee* utiliza um canal serial para transmissão de dados, isto significa que, para se comunicar utilizando ao mesmo tempo as três formas de comunicação, a comunicação serial *RxTx* e o *Zigbee* devem utilizar canais distintos.

REFERÊNCIAS BIBLIOGRÁFICAS

AHMED, N.; AMIR, S.; SIDDIQUI, A. A. *Raspberry pi* based online parameters monitoring and control system implemented using four sensor nodes. NED University Journal of Research. p. 49-56. Dec. 2014.

ALEMDAR, H.; ERSOY, C. 2010. Wireless sensor networks for healthcare: A survey. Computer Networks. p. 2688-2710.

AL-SAHIB, N. K. A.; AZEEZ, M. Z. Build and Interface Internet Mobile Robot using *Raspberry pi* and *Arduino*. Innovative Systems Design and Engineering. 6(1). 2015.

AKKAYA, K.; YOUNIS, M. A survey on routing protocols for wireless sensor networks. Ad Hoc Networks. p. 325-349. Nov. 2005.

AKYILDIZ, I. F., SU, W., SANKARASUBRAMANIAM, Y., CAYIRCI, E. Wireless sensor networks: a survey. Computer Networks. p. 393-422. Dec. 2002.

BAKULE, L. Decentralized control: An overview. Annual Reviews in Control. 32: 87–98. 2008.

BAKULE, L.; PAPÍK, M. Decentralized control and communication. Annual Reviews in Control. 36 (1): p. 1–10. Apr. 2012.

BO SUM, J. J.; WU, K. A Prototype Wireless Sensor Network for Precision Agriculture. Proceedings of the IEEE 33rd International Conference on Distributed Computing Systems Workshops. 2013.

BOLTON, W. Programmable Logic Controllers. 6th edition. Waltham: Newnes. 2015.

BROCK, J. D.; BRUCE, R. F. Sensing the world with a *Raspberry pi*. Journal of Computing Sciences in Colleges. 30 (2): p. 174-175. Dec. 2014.

CHANDRAKASAN, A.; AMIRTHARAJAH, R., CHO, S.; GOODMAN, J.; KONDURI, G.; KULIK, J.; RABINER, W.; WANG, A. Design considerations for distributed micro-sensor systems. Proceedings of the IEEE 1999 Custom Integrated Circuits Conference, San Diego, CA, May 1999.

CHONG, C.Y.; KUMAR, S. P. Sensor Networks: Evolution, Opportunities, and Challenges. Proceedings of the IEEE. Aug. 2003.

ESSA, I.A. Ubiquitous sensing for smart and aware environments, IEEE Personal Communications. Oct, 2000.

FIELDBUS FOUNDATION. FOUNDATION for ROM: Petrobras Live Field Demonstration. 2013.

FLAMMINI, A.; FERRARI, P.; MARIOLI, D.; SISINNI, E.; TARONI, A. Wired and Wireless sensor networks for industrial applications. Microeletronics Journal. 40(9): p. 1322-1336. Oct. 2008.

HANDE, A.; POLK, T.; WALKER, W.; BHATIA, D. Indoor solar energy harvesting for sensor network router nodes. Microprocessors and Microsystems. 2007.

HANGGORO, A.; PUTRA, M.A. ; REYNALDO, R. ; SARI, R.F. Green house monitoring and controlling using Android mobile application. In Proceedings of International Conference on QiR (Quality in Research). 25-28 June, Yogyakarta. 2013.

INTANAGONWIWAT, C.; ESTRIN, D.; GOVINDAN, R.; HEIDEMANN, J. Impact of Network Density on Data Aggregation in Wireless Sensor Networks. Proceedings of the IEEE 22nd International Conference on Distributed Computing Systems. 2002.

ISO-KETOLA, P.; KARINSALO, T.; VANHALA, J. HipGuard: a wearable measurement system for patients recovering from a hip operation. In: Second International Conference on Pervasive Computing Technologies for Healthcare, 2008.

JARQUE, C. M., and A. K. BERA. "A Test for Normality of Observations and Regression Residuals." International Statistical Review. Vol. 55, No. 2, 1987, pp. 163–172.

JIMENEZ, I.; LOPES, E.; RAMIREZ, A. “Synthesis of Ladder Diagrams from Petri Nets Controller Models”. In Proceedings of the IEEE International Symposium on Intelligent Control, Mexico City, September 5- 7. 2001.

LILLIEFORS, H. W. "On the Kolmogorov-Smirnov test for normality with mean and variance unknown." Journal of the American Statistical Association. Vol. 62, 1967, pp. 399–402.

MASSEY, F. J. "The Kolmogorov-Smirnov Test for Goodness of Fit." Journal of the American Statistical Association. Vol. 46, No. 253, 1951, pp. 68–78.

MCPHERSON, A.; ZAPPI, V. An Environment for Submillisecond-Latency Audio and Sensor Processing on *Beaglebone* Black. Audio Engineering Society. May. 2015.

MILLER, L. H. "Table of Percentage Points of Kolmogorov Statistics." Journal of the American Statistical Association. Vol. 51, No. 273, 1956, pp. 111–121.

MOWAD, M. A. E. L.; FATHY, A.; HAFEZ, A. Smart Home Automated Control System Using Android Application and Microcontroller. International Journal of Scientific & Engineering Research. 5 (5). May. 2014.

PETRUZELLA, F. D. Programmable logic controllers. New York: Glencoe/McGraw-Hill. 1996.

PLCOpen, 2015. IEC 61131-3: a standard programming resource. Disponível em: http://www.plcopen.org/pages/tc1_standards/downloads/intro_iec.pdf. Acessado em 02/04/2015.

PNG, L. C.; CHEN, L.; LIU, S.; PEH, W. K. An *Arduino*-based indoor positioning system (IPS) using visible light communication and ultrasound. In Proceedings of IEEE International Conference on Consumer Electronics. 26-28 May. Tapei. 2014.

SALINAS, O. H.; SEBASTIAN, P. J.; ESTRADA, A.; ORTIZ, M. E. L. Low Cost Solarimetric Station with Solar Resource Calculation Based on *Arduino* Microcontroller and Web Platform. Energy and Environment Focus. 2 (4): p. 326-330. Dec. 2013.

SCHUPPEN, J. H.; BOUTIN, O.; KEMPKER, P. L.; KOMENDA, J.; MASOPUST, T.; PAMBAKIAN, N.; RAN, A. C. M. Control of Distributed Systems: Tutorial and Overview. European Journal of Control. 17: (5-6). p. 579–602. June. 2011.

SELVARAJU, S. RUBANANTH, R.; RATHINAKUMAR, R. *ZIGBEE* based Automatic Load Monitoring System. International Journal of Computer Applications. 90(7). 2014.

SILVEIRA, P. R.; SANTOS, W. E. Automação e Controle Discreto. São Paulo: Erica. 2005.

SUESUT, T.; INBAN, P.; NILAS, P.; RERNGREUN, P.; GULPHANICH, S. “Interpretation Petri net model to IEC 1131-3: LD for programmable logic controller”. In Proceedings of the IEEE International Conference on Robotics, Automation and Mechatronics, Singapore, December 1-3. 2004.

THAPA, D.; DANGOL, S.; WANG, G. N. “Transformation from Petri Nets Model to Programmable Logic Controller using One-to-One Mapping Technique”. In Proceedings of the IEEE International Conference on Computational Intelligence for Modelling, Control and Automation and the IEEE International Conference on Intelligent Agents, Web Technologies and Internet Commerce, Vienna, November 28-30. 2005.

TRAVAGLIONE, B.; MUNYARD, A.; MATTHEWS, D. Using low cost single-board microcontrollers to record underwater acoustical data. In Proceedings of 43rd International Congress on Noise Control Engineering. 16-19 Nov. Melbourne. 2014.

UZAM, M.; JONES, A. H.; AJLOUNI, N. “Conversion of Petri Net Controllers for Manufacturing Systems into Ladder Logic Diagrams”. In Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation, Kauai, November 18-21. 1996.

WEHRMEISTER, M. A; FREITAS, E. P.; BINOTTO, A. P. D.; PEREIRA, C. E. Combining aspects and object-orientation in model-driven engineering for distributed industrial mechatronics systems. IEEE Mechatronics. 24 (7). p. 844-865. Oct. 2014.

WERNER-ALLEN, G.; JOHNSON, J.; RUIZ, M.; LEES, J.; WELSH, M. Monitoring Volcanic Eruptions with a Wireless Sensor Network. Second European Workshop on Wireless Sensor Networks (EWSN'05). 2005.

WIN, E. T.; HTUN, Z. L. Design and Implementation of Solar Tracking Control System using Microcontroller. International Journal of Scientific Engineering and Technology Research. 3(10): p. 2038-2041. May. 2014.

APÊNDICE A

Análise Estatística

Para a análise dos dados é necessário calculas algumas medidas, seja de tendência ou dispersão. Além destas medidas são realizados testes de hipóteses para verificar a normalidade dos dados. Para inferir a hipótese de que duas médias são iguais deve-se, primeiramente, verificar se as amostras são homocedásticas ou heterocedásticas. O teste F faz essa verificação e é apresentado em sequência. Após o teste pode-se calcular o intervalo de confiança para a diferença entre as médias, que também está apresentado no apêndice. A seguir são apresentados os parâmetros e os testes de hipóteses calculados, além das funções utilizadas no *software Matlab R2012a.*

Média aritmética

A média aritmética é uma medida de tendência central e é calculada pela Eq. 3:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} A_i \tag{3}$$

Onde *A* é o vetor de dados, que se deseja calcular a média e *N* é o número de dados. A função utilizada para o cálculo da média é dada por $h = mean(dados)$.

Desvio-padrão amostral

O desvio-padrão amostral é uma medida de dispersão relativa à média, e se obtém fazendo a raiz quadrada da variância conforme mostra a Eq. 4. O desvio padrão é uma medida

que só pode assumir valores positivos e quanto maior o seu valor, maior é a dispersão dos dados.

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(A_i - \mu)^2} \quad (4)$$

Onde *A* é o vetor de dados, μ é a média aritmética desses dados e *N* a quantidade de dados. A função utilizada para o cálculo do desvio-padrão amostral é dada por $h = std(dados)$.

***Kurtosis* (curtose)**

A *kurtosis* é uma medida de dispersão referente a curva da distribuição de probabilidade de um conjunto de dados. Ela está relacionada ao achatamento da curva. Uma *kurtosis* de valor 3 indica que a curva possui o mesmo achatamento de uma distribuição normal. Um valor acima que 3 indica que a curva possui um achatamento maior, ou seja, o pico desta curva é mais alto que o de uma distribuição normal, e o valor abaixo de 3 indica um pico mais baixo.

A função utilizada para o cálculo da *kurtosis* é dada por $h = kurtosis(dados)$.

***Skewness* (assimetria)**

A *skewness* é uma medida de dispersão referente a curva da distribuição de probabilidade de um conjunto de dados. Ela está relacionada a assimetria da curva. Uma *skewness* de valor 0 indica que a curva possui uma distribuição simétrica. Um valor acima de 0 indica que a distribuição tem uma cauda direita (valores acima da média), enquanto que um valor abaixo de 0 indica uma distribuição com uma cauda esquerda, ou seja, valores abaixo da média.

A função utilizada para o cálculo da *skewness* é dada por $h = skewness(dados)$.

Testes de normalidade

Para verificar a hipótese de que os dados foram retirados de uma população com distribuição de probabilidade normal devem ser realizados alguns testes. São apresentados

os testes realizados e suas devidas funções. Estão em sequência os testes de Kolmogorov-Smirnov, de Lilliefors e de Jarque-Bera.

Teste de Kolmogorov-Smirnov

O teste para verificação de normalidade de Kolmogorov-Smirnov é um teste não paramétrico que considera como hipótese nula que os dados são retirados de uma função distribuição de probabilidade normal. A base da formulação do teste no Matlab é de Massey (1951) enquanto que o algoritmo é semelhante no proposto por Marsaglia et al. (2003).

A função pode ser utilizada da seguinte forma: $h = kstest(dados)$. Quando a função retorna o valor 0 ela aceita a hipótese nula, enquanto que quando retorna o valor 1 a função rejeita a hipótese nula. O nível de significância alfa padrão da função é de 5%. A função rejeita a hipótese nula com a comparação do *p-value* com o nível de significância alfa.

Teste de Lilliefors

Este teste de normalidade foi proposto por Lilliefors (1967) e é baseado no teste de Kolmogorov-Smirnov. Da mesma forma que o teste anterior este define a hipótese nula de que os dados são de uma distribuição normal, enquanto que a hipótese alternativa afirma que não é uma distribuição normal. Para calcular o valor crítico (*cv*) o teste interpola uma tabela de valores críticos calculados utilizando simulação de Monte Carlo para amostras menores que 1000 e significância entre 0,001 e 0,5.

Quando o valor calculado pelo teste estatístico é maior que o *cv*, o teste rejeita a hipótese nula ao nível de significância alfa. A função é definida por: $h = lillietest(dados)$ e o nível de significância alfa padrão é de 5%.

Teste de Jarque-Bera

O teste de normalidade proposto por Jarque e Bera (1987) recebe o nome dos dois autores e utiliza como parâmetros os coeficientes de *kurtosis* e *skewness* (que em uma distribuição normal equivalem a 3 e 0, respectivamente).

O teste de Jarque-Bera utiliza a distribuição qui-quadrado para estimar o *cv* para amostras grandes, deferindo para o teste de Lilliefors para amostras pequenas. Para amostras grandes é realizado uma aproximação do qui-quadrado analítico.

A função é definida por $h = jbtest(dados)$ com um nível de significância alfa padrão de 5%.

Teste F

O teste F verifica se as variâncias de duas amostras são iguais. Estatisticamente o teste é realizado pela Eq. (5):

$$F = \frac{s_1^2}{s_2^2} \tag{5}$$

Onde s_1 e s_2 são os desvios-padrão amostral. Quanto mais essa razão desvia de 1 é mais provável que a hipótese nula seja rejeitada. Sobre a hipótese nula, o teste F tem uma distribuição do tipo F com $N_1 - 1$ graus de liberdade no numerador e $N_2 - 1$ graus de liberdade no denominador, sendo N_1 e N_2 o número de amostras dos dois conjuntos de dados.

A função que realiza este teste no Matlab é $h = vartest2(dados_1, dados_2)$. Caso a função retorne 0, o teste não tem condições de rejeitar a hipótese nula de que os dois conjuntos de dados possuem a mesma variância. Caso retorne 1, a hipótese nula é rejeitada. O nível de significância padrão do teste é de 5%.

Quando a hipótese nula é aceita dizemos que as populações são homocedásticas e quando é rejeitada dizemos que as populações são heterocedásticas.

Intervalo de confiança para a diferença entre duas médias ($\mu_a - \mu_b$)

Quando as variâncias populacionais são desconhecidas pode-se calcular este intervalo de confiança para dois casos: homocedástico e heterocedástico. Para o primeiro caso, o intervalo de confiança é dado pela Eq. (6) sendo $t_{\frac{\alpha}{2}}$ com $n_a + n_b - 2$ graus de liberdade e s_p calculado conforme a Eq. (7).

$$IC(\mu_a - \mu_b)_{1-\alpha} = \overline{x_a} - \overline{x_b} \pm t_{\frac{\alpha}{2}} s_p \sqrt{\frac{1}{n_a} + \frac{1}{n_b}} \tag{6}$$

$$s_p = \sqrt{\frac{(n_a - 1)s_a^2 - (n_b - 1)s_b^2}{n_a + n_b - 2}} \tag{7}$$

Quando as populações são heterocedásticas o intervalo de confiança é dado pela Eq. (8) sendo $t_{\frac{\alpha}{2}}$ com ϑ graus de liberdade, calculado pela Eq. (9).

$$IC(\mu_a - \mu_b)_{1-\alpha} = \overline{x_a} - \overline{x_b} \pm t_{\frac{\alpha}{2}} \sqrt{\frac{s_a^2}{n_a} + \frac{s_b^2}{n_b}} \tag{8}$$

$$\vartheta = \frac{(\frac{s_a^2}{n_a} + \frac{s_b^2}{n_b})}{\frac{\left(\frac{s_a^2}{n_a}\right)^2}{n_a - 1} + \frac{\left(\frac{s_b^2}{n_b}\right)^2}{n_b - 1}} \tag{9}$$

Quando o intervalo de confiança para a diferença entre as médias compreende o valor 0, pode-se afirmar com o nível de significância considerado que os valores médios em questão são iguais. Então, se o intervalo não compreende o valor 0, essa afirmação deve ser negada.

APÊNDICE B

Códigos das arquiteturas de controle

Arquitetura centralizado

Código da arquitetura de controle centralizado

```
// Biblioteca do cartão SD
#include <SD.h>
File dataFile;

// bancada
// Sensores de identificação da peça
int f1 = 38;          // sensor optico tamanho pequeno
int f2 = 39;          // sensor optico tamanho médio
int f3 = 40;          // sensor optico tamanho grande
int ind = 41;          // sensor indutivo
int cap = 42;          // sensor capacitivo

// Sensores fim de curso
int fc1 = 43;          // fim de curso separação peça pequena
int fc2 = 44;          // fim de curso separação peça média
int fc3 = 45;          // fim de curso separação peça grande
int fc4 = 46;          // fim de curso separação peça metálica

// Atuadores
int v1 = 30;          // válvula 3-2 retorno por mola para primeira separação
int v2 = 31;          // válvula 3-2 retorno por mola para segunda separação
int v3 = 32;          // válvula 5-2 duplo solenoide para terceira separação - AVANÇO
int v4 = 33;          // válvula 5-2 duplo solenoide para terceira separação - RETORNO


// FLAGS
boolean flagf1 = false;
boolean flagf2 = false;
boolean flagf3 = false;
boolean flagind = false;
boolean flagcap = false;
boolean flagfc1 = false;
boolean flagfc2 = false;
boolean flagfc3 = false;
```

```
boolean flagfc4 = false;

// TIME
long t_ant = 0;
long t_atual = 0;
long time = 0;
//int taxa = 300;  // taxa de comunicação

void setup(){
  Serial.begin(9600);

  // Definições para utilização do Shield Ethernet e cartão SD no Arduino MEGA
  // Não utilizar as portas 4 e 10
  pinMode(10,OUTPUT);
  digitalWrite(10,HIGH);
    if(!SD.begin(4)){
      Serial.println("Card failed, or not inserted");
  }
  if(SD.exists("datalog.txt")){
    SD.remove("datalog.txt");
  }

  dataFile = SD.open("datalog.txt",FILE_WRITE);
    if(!dataFile){
      Serial.println("erro na abertura do arquivo");
  }
  dataFile.println("type;time");
  dataFile.flush();

  // Definição das PORTAS
  pinMode(fc1,INPUT);
  pinMode(fc2,INPUT);
  pinMode(fc3,INPUT);
  pinMode(fc4,INPUT);
  pinMode(ind,INPUT);
  pinMode(cap,INPUT);
  pinMode(f1,INPUT);
  pinMode(f2,INPUT);
  pinMode(f3,INPUT);
  pinMode(v1,OUTPUT);
  pinMode(v2,OUTPUT);
  pinMode(v3,OUTPUT);
  pinMode(v4,OUTPUT);
  digitalWrite(v1,LOW);
  digitalWrite(v2,LOW);
  digitalWrite(v3,LOW);
  digitalWrite(v4,HIGH);
  delay(20);
  digitalWrite(v4,LOW);
}

void loop(){

  if(digitalRead(f1)==HIGH){
    flagf1 = true;
  }
  if(digitalRead(f2)==HIGH){
    flagf2 = true;
  }
```

```
  if(digitalRead(f3)==HIGH){
    flagf3 = true;
  }
  if(digitalRead(ind)==HIGH){
    flagind = true;
  }
  if(digitalRead(cap)==HIGH){
    flagcap = true;
  }
  */
  if((flagf1&&!flagind)&&flagcap&&(!flagf2&&!flagf3)){      // Peça pequena nao metálica
    //Serial.println("Peça pequena");
    t_ant = millis();
    flagf1 = false;
    flagf2 = false;
    flagf3 = false;
    flagind = false;
    flagcap = false;
    //delay(200);
    digitalWrite(v1,LOW);
    digitalWrite(v2,LOW);
    digitalWrite(v3,LOW);
    digitalWrite(v4,LOW);
    //Serial.println("Pistão 1 acionado");
    while(!digitalRead(fc4)){
      // espera o fim de curso ser acionado
    }
    t_atual = millis();
    time = t_atual - t_ant;
    dataFile.print("pequena");
    dataFile.flush();
    dataFile.print(";");
    dataFile.flush();
    dataFile.println(time);
    dataFile.flush();
   // Serial.println("Salvo no cartão");
  }

  else if((flagf1&&flagf2)&&!flagf3&&(!flagind&&flagcap)){      // Peça média nao metálica
    //Serial.println("Peça média");
    t_ant = millis();
    //delay(200);
    flagf1 = false;
    flagf2 = false;
    flagf3 = false;
    flagind = false;
    flagcap = false;
    digitalWrite(v4,LOW);
    digitalWrite(v3,HIGH);
    //Serial.println("Pistão 2 acionado");
    while(!digitalRead(fc3)){
      // espera o fim de curso ser acionado
    }
    t_atual = millis();
    digitalWrite(v3,LOW);
    digitalWrite(v4,HIGH);
    time = t_atual - t_ant;
    dataFile.print("media");
     dataFile.flush();
```

```
  dataFile.print(";");
   dataFile.flush();
  dataFile.println(time);
   dataFile.flush();
   digitalWrite(v4,LOW);
  //Serial.println("Salvo no cartão");
}

else if((flagf1&&flagf2)&&flagf3&&(!flagind&&flagcap)){      // Peça grande nao metálica
  //Serial.println("Peça grande");
  t_ant = millis();
  //delay(200);
  flagf1 = false;
  flagf2 = false;
  flagf3 = false;
  flagind = false;
  flagcap = false;
  digitalWrite(v2,HIGH);
  //Serial.println("Pistão 3 acionado");
  //delay(10);
  //digitalWrite(v3,LOW);
  while(!digitalRead(fc2)){
    // espera o fim de curso ser acionado
  }
  t_atual = millis();
  digitalWrite(v2,LOW);
  //Serial.println("Pistão 3 recuado");
  //delay(10);
  //digitalWrite(v4,LOW);
  time = t_atual - t_ant;
  dataFile.print("grande");
   dataFile.flush();
  dataFile.print(";");
   dataFile.flush();
  dataFile.println(time);
   dataFile.flush();
  //Serial.println("Salvo no cartão");
}

  else if((flagf1||flagf2||flagf3)&&(flagind&&flagcap)){      // Peça metálica
  //Serial.println("Peça metálica");
  t_ant = millis();
  //delay(200);
  flagind = false;
  flagf1 = false;
  flagf2 = false;
  flagf3 = false;
  flagcap = false;
  digitalWrite(v1,HIGH);
  while(!digitalRead(fc1)){
    // espera o fim de curso ser acionado
  }
  t_atual = millis();
  time = t_atual - t_ant;
  digitalWrite(v1,LOW);
  dataFile.print("metalica");
   dataFile.flush();
  dataFile.print(";");
   dataFile.flush();
```

```
    dataFile.println(time);
     dataFile.flush();
    //Serial.println("Salvo no cartão");
  }
}
```

Arquitetura de controle distribuído com comunicação serial *RxTx* entre dispositivos

Código do mestre

```
// MESTRE-ESCRAVO
// CÓDIGO MESTRE

// Biblioteca do cartão SD
#include <SD.h>
File dataFile;

// Timer de interrupção
#include <TimerOne.h>

// Atuadores
int v1 = 30;        // válvula 3-2 retorno por mola para primeira separação
int v2 = 31;        // válvula 3-2 retorno por mola para segunda separação
int v3 = 32;        // válvula 5-2 duplo solenoide para terceira separação - AVANÇO
int v4 = 33;        // válvula 5-2 duplo solenoide para terceira separação - RETORNO

int msg[9] = { };
//int tempodelay = 10;

// FLAGS
boolean flagf1 = false;
boolean flagf2 = false;
boolean flagf3 = false;
boolean flagind = false;
boolean flagcap = false;
boolean flagfc1 = false;
boolean flagfc2 = false;
boolean flagfc3 = false;
boolean flagfc4 = false;

// TIME
long t_ant = 0;
long t_atual = 0;
long time = 0;

//int taxa = 115200; // Taxa de comunicação
//int delaysolicitacao = 100000;

void setup(){
  Serial.begin(9600);
  Serial1.begin(115200);
  Timer1.initialize(4000); // microsegundos
  Timer1.attachInterrupt(solicita);

  // Definições para utilização do Shield Ethernet e cartão SD no Arduino MEGA
  // Não utilizar as portas 4 e 10
  pinMode(10,OUTPUT);
  digitalWrite(10,HIGH);
  if(!SD.begin(4)){
    Serial.println("Card failed, or not inserted");
  }
  if(SD.exists("datalog.txt")){
    SD.remove("datalog.txt");
  }
```

```
  dataFile = SD.open("datalog.txt",FILE_WRITE);
  if(!dataFile){
    Serial.println("erro na abertura do arquivo");
  }
  dataFile.println("type;time");
  dataFile.flush();

  // Definição das PORTAS
  pinMode(v1,OUTPUT);
  pinMode(v2,OUTPUT);
  pinMode(v3,OUTPUT);
  pinMode(v4,OUTPUT);
  digitalWrite(v1,LOW);
  digitalWrite(v2,LOW);
  digitalWrite(v3,LOW);
  digitalWrite(v4,HIGH);
  delay(20);
  digitalWrite(v4,LOW);
}

void loop(){

  lemsg();

  if((msg[0]&&!msg[1]&&!msg[2]&&!msg[3])&&msg[4]){      // Peça pequena nao metálica
    //Serial.println("Peça pequena");
    t_ant = millis();
    digitalWrite(v1,LOW);
    digitalWrite(v2,LOW);
    digitalWrite(v3,LOW);
    digitalWrite(v4,LOW);
    //Serial.println("Pistão 1 acionado");
    do{
      lemsg();
      // espera o fim de curso ser acionado
    }while(!msg[8]);
    t_atual = millis();
    time = t_atual - t_ant;
    dataFile.print("pequena");
    dataFile.flush();
    dataFile.print(";");
    dataFile.flush();
    dataFile.println(time);
    dataFile.flush();
   // Serial.println("Salvo no cartão");
  }

  else if(msg[2]&&msg[0]&&msg[1]&&!msg[3]&&msg[4]){      // Peça grande nao metálica
    //Serial.println("Peça grande");
    t_ant = millis();
    //delay(200);
    digitalWrite(v2,HIGH);
    //Serial.println("Pistão 3 acionado");
    //delay(10);
    //digitalWrite(v3,LOW);
    do{
      lemsg();
      // espera o fim de curso ser acionado
```

```
  }while(!msg[6]);
  t_atual = millis();
  digitalWrite(v2,LOW);
  //Serial.println("Pistão 3 recuado");
  time = t_atual - t_ant;
  dataFile.print("grande");
  dataFile.flush();
  dataFile.print(";");
  dataFile.flush();
  dataFile.println(time);
  dataFile.flush();
  //Serial.println("Salvo no cartão");
}

else if(msg[1]&&msg[0]&&!msg[2]&&!msg[3]&&msg[4]){      // Peça média nao metálica
  //Serial.println("Peça média");
  t_ant = millis();
  //delay(200);
  digitalWrite(v4,LOW);
  digitalWrite(v3,HIGH);
  //Serial.println("Pistão 2 acionado");
  do{
    lemsg();
    // espera o fim de curso ser acionado
  }while(!msg[7]);
  t_atual = millis();
  digitalWrite(v3,LOW);
  digitalWrite(v4,HIGH);
  //delay(tempodelay);
  time = t_atual - t_ant;
  dataFile.print("media");
  dataFile.flush();
  dataFile.print(";");
  dataFile.flush();
  dataFile.println(time);
  dataFile.flush();
  //digitalWrite(v4,LOW);
  //Serial.println("Salvo no cartão");
}



  else if(msg[3]&&msg[4]&&(msg[0]||msg[1]||msg[2])){      // Peça metálica
  //Serial.println("Peça metálica");
  t_ant = millis();
  //delay(200);
  digitalWrite(v1,HIGH);
  do{
    lemsg();
    // espera o fim de curso ser acionado
  }while(!msg[5]);
  t_atual = millis();
  time = t_atual - t_ant;
  digitalWrite(v1,LOW);
  dataFile.print("metalica");
  dataFile.flush();
  dataFile.print(";");
  dataFile.flush();
  dataFile.println(time);
```

```
    dataFile.flush();
    //Serial.println("Salvo no cartão");
  }

}

void lemsg(){
  if(Serial1.available()>8){  // Caso receba uma mensagem
    for(int i=0;i<9;i++){  // faz a leitura dos dados e salva na variável msg de tamanho 9
        int data = Serial1.read();
        msg[i] = data;
     // Serial.print(msg[i]);
    }
    //Serial.flush();
    //Serial.println(" ");
  }
}

void solicita(){  // Solicita status do Escravo
  Serial1.write(1);
  //Serial.println("msg enviada");
}
```

Código do escravo

```
                        // MESTRE ESCRAVO
                         // ESCRAVO

// bancada
// Sensores de identificação da peça
int f1 = 2;          // sensor optico tamanho pequeno
int f2 = 3;          // sensor optico tamanho médio
int f3 = 4;          // sensor optico tamanho grande
int ind = 5;         // sensor indutivo
int cap = 6;         // sensor capacitivo

// Sensores fim de curso
int fc1 = 7;         // fim de curso separação peça pequena
int fc2 = 8;         // fim de curso separação peça média
int fc3 = 9;         // fim de curso separação peça grande
int fc4 = 10;        // fim de curso separação peça metálica

// FLAGS
boolean flagf1 = false;
boolean flagf2 = false;
boolean flagf3 = false;
boolean flagind = false;
boolean flagcap = false;
boolean flagfc1 = false;
boolean flagfc2 = false;
boolean flagfc3 = false;
boolean flagfc4 = false;

void setup(){
  //Serial.begin(9600);
  Serial1.begin(115200);
  pinMode(fc1,INPUT);
  pinMode(fc2,INPUT);
  pinMode(fc3,INPUT);
  pinMode(fc4,INPUT);
  pinMode(ind,INPUT);
  pinMode(cap,INPUT);
  pinMode(f1,INPUT);
  pinMode(f2,INPUT);
  pinMode(f3,INPUT);
}

void loop(){

    if(digitalRead(f1)==HIGH){
      flagf1 = true;
    }
    if(digitalRead(f2)==HIGH){
      flagf2 = true;
    }
    if(digitalRead(f3)==HIGH){
      flagf3 = true;
    }
    if(digitalRead(ind)==HIGH){
      flagind = true;
    }
    if(digitalRead(cap)==HIGH){
```

```
      flagcap = true;
    }
    if(digitalRead(fc1)==HIGH){
      flagfc1 = true;
    }
    if(digitalRead(fc2)==HIGH){
      flagfc2 = true;
    }
    if(digitalRead(fc3)==HIGH){
      flagfc3 = true;
    }
    if(digitalRead(fc4)==HIGH){
      flagfc4 = true;
    }


    int msg[ ] = {flagf1, flagf2, flagf3, flagind, flagcap, flagfc1, flagfc2, flagfc3, flagfc4};

    if(Serial1.available()>0){
      int data = Serial1.read();
      Serial.print(data);
      if(data == 1){
        for(int i=0;i<9;i++){
          Serial1.write(msg[i]);
          //Serial.print(msg[i]);
        }
        //Serial.println("msg enviada");
        if(flagfc1||flagfc2||flagfc3||flagfc4){
          //Serial.println("vai resetar as flags");
          flagf1 = false;
          flagf2 = false;
          flagf3 = false;
          flagind = false;
          flagcap = false;
          flagfc1 = false;
          flagfc2 = false;
          flagfc3 = false;
          flagfc4 = false;
        }
      }
    }
}
```

Arquitetura de controle distribuído com comunicação I²C entre dispositivos

Dois dispositivos

Código do mestre

```
                                        // MESTRE-ESCRAVO
                                        // CÓDIGO MESTRE

// Biblioteca do cartão SD
#include <SD.h>
File dataFile;

// Timer de interrupção
//#include <TimerOne.h>

#include <Wire.h>

// Atuadores
int v1 = 30;        // válvula 3-2 retorno por mola para primeira separação
int v2 = 31;        // válvula 3-2 retorno por mola para segunda separação
int v3 = 32;        // válvula 5-2 duplo solenoide para terceira separação - AVANÇO
int v4 = 33;        // válvula 5-2 duplo solenoide para terceira separação - RETORNO

int msg[9] = { };
int tempodelay = 10;

// FLAGS
boolean flagf1 = false;
boolean flagf2 = false;
boolean flagf3 = false;
boolean flagind = false;
boolean flagcap = false;
boolean flagfc1 = false;
boolean flagfc2 = false;
boolean flagfc3 = false;
boolean flagfc4 = false;

// TIME
long t_ant = 0;
long t_atual = 0;
long time = 0;

void setup(){
  Wire.begin();
  Serial.begin(9600);

  // Definições para utilização do Shield Ethernet e cartão SD no Arduino MEGA
  // Não utilizar as portas 4 e 10
  pinMode(10,OUTPUT);
  digitalWrite(10,HIGH);
  if(!SD.begin(4)){
    Serial.println("Card failed, or not inserted");
  }
  if(SD.exists("datalog.txt")){
    SD.remove("datalog.txt");
  }
```

```
  dataFile = SD.open("datalog.txt",FILE_WRITE);
  if(!dataFile){
    Serial.println("erro na abertura do arquivo");
  }
  dataFile.println("type;time");
  dataFile.flush();

  // Definição das PORTAS
  pinMode(v1,OUTPUT);
  pinMode(v2,OUTPUT);
  pinMode(v3,OUTPUT);
  pinMode(v4,OUTPUT);
  digitalWrite(v1,LOW);
  digitalWrite(v2,LOW);
  digitalWrite(v3,LOW);
  digitalWrite(v4,HIGH);
  delay(10);
  digitalWrite(v4,LOW);
}

void loop(){
  //delay(tempodelay);
  lemsg();

  if((msg[0]&&!msg[3])&&msg[4]&&(!msg[1]&&!msg[2])){      // Peça pequena nao metálica
    //Serial.println("Peça pequena");
    t_ant = millis();
    //delay(200);
    digitalWrite(v1,LOW);
    digitalWrite(v2,LOW);
    digitalWrite(v3,LOW);
    digitalWrite(v4,LOW);
    //Serial.println("Pistão 1 acionado");
    do{
      lemsg();
      // espera o fim de curso ser acionado
    }while(!msg[8]);
    t_atual = millis();
    time = t_atual - t_ant;
    dataFile.print("pequena");
    dataFile.flush();
    dataFile.print(";");
    dataFile.flush();
    dataFile.println(time);
    dataFile.flush();
   // Serial.println("Salvo no cartão");
  }
  else if(msg[1]&&!msg[3]&&msg[4]&&!msg[2]){      // Peça média nao metálica
    //Serial.println("Peça média");
    t_ant = millis();
    //delay(200);
    digitalWrite(v4,LOW);
    digitalWrite(v3,HIGH);
    //Serial.println("Pistão 2 acionado");
    do{
      lemsg();
      // espera o fim de curso ser acionado
    }while(!msg[7]);
```

```
  t_atual = millis();
  digitalWrite(v3,LOW);
  digitalWrite(v4,HIGH);
  time = t_atual - t_ant;
  dataFile.print("media");
  dataFile.flush();
  dataFile.print(";");
  dataFile.flush();
  dataFile.println(time);
  dataFile.flush();
  //Serial.println("Salvo no cartão");
}

else if(msg[0]&&msg[1]&&msg[2]&&!msg[3]&&msg[4]){      // Peça grande nao metálica
  //Serial.println("Peça grande");
  t_ant = millis();
  //delay(200);
  digitalWrite(v2,HIGH);
  //Serial.println("Pistão 3 acionado");
  //delay(10);
  //digitalWrite(v3,LOW);
  do{
    lemsg();
    // espera o fim de curso ser acionado
  }while(!msg[6]);
  t_atual = millis();
  //Serial.println("Pistão 3 recuado");
  digitalWrite(v2,LOW);
  time = t_atual - t_ant;
  dataFile.print("grande");
  dataFile.flush();
  dataFile.print(";");
  dataFile.flush();
  dataFile.println(time);
  dataFile.flush();
  //Serial.println("Salvo no cartão");
}

  else if(msg[3]&&msg[4]&&(msg[0]||msg[1]||msg[2])&&!msg[8]){      // Peça metálica
  //Serial.println("Peça metálica");
  t_ant = millis();
  //delay(200);
  digitalWrite(v1,HIGH);
  do{
    lemsg();
    // espera o fim de curso ser acionado
  }while(!msg[5]);
  t_atual = millis();
  digitalWrite(v1,LOW);
  time = t_atual - t_ant;
  dataFile.print("metalica");
  dataFile.flush();
  dataFile.print(";");
  dataFile.flush();
  dataFile.println(time);
  dataFile.flush();
  //Serial.println("Salvo no cartão");
}
```

```
}

void lemsg(){
  Wire.requestFrom(2,9);
  if(Wire.available()>0){  // Caso receba uma mensagem
    for(int i=0;i<9;i++){  // faz a leitura dos dados e salva na variável msg de tamanho 9
        int data = Wire.read();
        msg[i] = data;
        //Serial.print(msg[i]);
    }
    //Serial.flush();
    //Serial.println(" ");
  }
}
```

Código do escravo

```
                        // MESTRE ESCRAVO
                        // ESCRAVO

#include <Wire.h>

// bancada
// Sensores de identificação da peça
int f1 = 2;          // sensor optico tamanho pequeno
int f2 = 3;          // sensor optico tamanho médio
int f3 = 4;          // sensor optico tamanho grande
int ind = 5;         // sensor indutivo
int cap = 6;         // sensor capacitivo

// Sensores fim de curso
int fc1 = 7;         // fim de curso separação peça pequena
int fc2 = 8;         // fim de curso separação peça média
int fc3 = 9;         // fim de curso separação peça grande
int fc4 = 10;        // fim de curso separação peça metálica

// FLAGS
boolean flagf1 = false;
boolean flagf2 = false;
boolean flagf3 = false;
boolean flagind = false;
boolean flagcap = false;
boolean flagfc1 = false;
boolean flagfc2 = false;
boolean flagfc3 = false;
boolean flagfc4 = false;

unsigned char msg[9] = { };

void setup(){
  Serial.begin(9600);
  Wire.begin(2);
  Wire.onRequest(requestEvent);
 // Serial1.begin(9600);
  pinMode(fc1,INPUT);
  pinMode(fc2,INPUT);
  pinMode(fc3,INPUT);
  pinMode(fc4,INPUT);
  pinMode(ind,INPUT);
  pinMode(cap,INPUT);
  pinMode(f1,INPUT);
  pinMode(f2,INPUT);
  pinMode(f3,INPUT);
}

void loop(){

    if(digitalRead(f1)==HIGH){
      flagf1 = true;
    }
    if(digitalRead(f2)==HIGH){
      flagf2 = true;
    }
    if(digitalRead(f3)==HIGH){
```

```
      flagf3 = true;
    }
    if(digitalRead(ind)==HIGH){
      flagind = true;
    }
    if(digitalRead(cap)==HIGH){
      flagcap = true;
    }
    if(digitalRead(fc1)==HIGH){
      flagfc1 = true;
    }
    if(digitalRead(fc2)==HIGH){
      flagfc2 = true;
    }
    if(digitalRead(fc3)==HIGH){
      flagfc3 = true;
    }
    if(digitalRead(fc4)==HIGH){
      flagfc4 = true;
    }


    unsigned char msg[ ] = {flagf1, flagf2, flagf3, flagind, flagcap, flagfc1, flagfc2, flagfc3, flagfc4};

}

void requestEvent(){
  unsigned char msg[ ] = {flagf1, flagf2, flagf3, flagind, flagcap, flagfc1, flagfc2, flagfc3, flagfc4};
  Wire.write(msg,9);
  if(flagfc1||flagfc2||flagfc3||flagfc4){
     //Serial.println("vai resetar as flags");
     flagf1 = false;
     flagf2 = false;
     flagf3 = false;
     flagind = false;
     flagcap = false;
     flagfc1 = false;
     flagfc2 = false;
     flagfc3 = false;
     flagfc4 = false;
  }
}
```

Três dispositivos

Código do mestre

```
                                                    // MESTRE-ESCRAVO
                                                    // CÓDIGO MESTRE

// Biblioteca do cartão SD
#include <SD.h>
File dataFile;

// Timer de interrupção
//#include <TimerOne.h>

#include <Wire.h>

// Atuadores
int v1 = 30;          // válvula 3-2 retorno por mola para primeira separação
int v2 = 31;          // válvula 3-2 retorno por mola para segunda separação
int v3 = 32;          // válvula 5-2 duplo solenoide para terceira separação - AVANÇO
int v4 = 33;          // válvula 5-2 duplo solenoide para terceira separação - RETORNO

byte msg[5] = { };
byte msg2[4] = { };

boolean flagf1 = false;
boolean flagf2 = false;
boolean flagf3 = false;
boolean flagind = false;
boolean flagcap = false;
boolean flagfc1 = false;
boolean flagfc2 = false;
boolean flagfc3 = false;
boolean flagfc4 = false;

boolean vaizerar = false;

// TIME
long t_ant = 0;
long t_atual = 0;
long time = 0;

void setup(){
  Wire.begin();
  Serial.begin(9600);

  // Definições para utilização do Shield Ethernet e cartão SD no Arduino MEGA
  // Não utilizar as portas 4 e 10
  pinMode(10,OUTPUT);
  digitalWrite(10,HIGH);
  if(!SD.begin(4)){
    Serial.println("Card failed, or not inserted");
  }
  if(SD.exists("datalog.txt")){
    SD.remove("datalog.txt");
  }

  dataFile = SD.open("datalog.txt",FILE_WRITE);
```

```
  if(!dataFile){
    Serial.println("erro na abertura do arquivo");
  }
  dataFile.println("type;time");
  dataFile.flush();

  // Definição das PORTAS
  pinMode(v1,OUTPUT);
  pinMode(v2,OUTPUT);
  pinMode(v3,OUTPUT);
  pinMode(v4,OUTPUT);
  digitalWrite(v1,LOW);
  digitalWrite(v2,LOW);
  digitalWrite(v3,LOW);
  digitalWrite(v4,HIGH);
  delay(10);
  digitalWrite(v4,LOW);
}

void loop(){
  //Serial.println("Entrou no loop");
  lemsg_ident();
  lemsg_fc();

  if(flagfc1||flagfc2||flagfc3||flagfc4||vaizerar){
    //Serial.println("ZEROU TUDO");
    zerar_fc();
    zerar_ident();
    vaizerar = false;
  }

  if(flagf1&&!flagf2&&!flagf3&&!flagind&&flagcap){      // Peça pequena nao metálica
    //Serial.println("Peça pequena");
    t_ant = millis();
    //delay(200);
    digitalWrite(v1,LOW);
    digitalWrite(v2,LOW);
    digitalWrite(v3,LOW);
    digitalWrite(v4,LOW);
    zerar_ident();
    //Serial.println("Pistão 1 acionado");
    do{
      lemsg_fc();
      // espera o fim de curso ser acionado
    }while(flagfc4== false);
    t_atual = millis();
    vaizerar = true;
    zerar_fc();
    time = t_atual - t_ant;
    dataFile.print("pequena");
    dataFile.flush();
    dataFile.print(";");
    dataFile.flush();
    dataFile.println(time);
    dataFile.flush();
   // Serial.println("Salvo no cartão");
  }
  else if(flagf1&&flagf2&&!flagf3&&!flagind&&flagcap){      // Peça média nao metálica
    //Serial.println("Peça média");
```

```
  t_ant = millis();
  //delay(200);
  digitalWrite(v4,LOW);
  digitalWrite(v3,HIGH);
     zerar_ident();
  //Serial.println("Pistão 2 acionado");
  do{
    lemsg_fc();
    // espera o fim de curso ser acionado
  }while(flagfc3 == false);
  t_atual = millis();
  digitalWrite(v3,LOW);
  digitalWrite(v4,HIGH);
  zerar_fc();
  vaizerar = true;
  time = t_atual - t_ant;
  dataFile.print("media");
  dataFile.flush();
  dataFile.print(";");
  dataFile.flush();
  dataFile.println(time);
  dataFile.flush();
  //Serial.println("Salvo no cartão");
}

else if(flagf1&&flagf2&&flagf3&&!flagind&&flagcap){      // Peça grande nao metálica
  //Serial.println("Peça grande");
  t_ant = millis();
  //delay(200);
  digitalWrite(v2,HIGH);
  zerar_ident();
  vaizerar = true;
  //Serial.println("Pistão 3 acionado");
  //delay(10);
  //digitalWrite(v3,LOW);
  do{
    lemsg_fc();
    // espera o fim de curso ser acionado
  }while(flagfc2 == false);
  t_atual = millis();
  zerar_fc();
  //Serial.println("Pistão 3 recuado");
  digitalWrite(v2,LOW);
  time = t_atual - t_ant;
  dataFile.print("grande");
  dataFile.flush();
  dataFile.print(";");
  dataFile.flush();
  dataFile.println(time);
  dataFile.flush();
  //Serial.println("Salvo no cartão");
}

  else if((flagf1||flagf2||flagf3)&&flagind&&flagcap){      // Peça metálica
  //Serial.println("Peça metálica");
  t_ant = millis();
  //delay(200);
  digitalWrite(v1,HIGH);
  zerar_ident();
```

```
    do{
      lemsg_fc();
      // espera o fim de curso ser acionado
    }while(flagfc1 == false);
    t_atual = millis();
    zerar_fc();
    vaizerar = true;
    digitalWrite(v1,LOW);
    time = t_atual - t_ant;
    dataFile.print("metalica");
    dataFile.flush();
    dataFile.print(";");
    dataFile.flush();
    dataFile.println(time);
    dataFile.flush();
    //Serial.println("Salvo no cartão");
  }

}

void lemsg_ident(){
  //Serial.println("entrou na função lemsg()");
  Wire.requestFrom(2,5);
  //Serial.println("solicitou do escravo 1");
  if(Wire.available()>0){  // Caso receba uma mensagem
    for(int i=0;i<5;i++){  // faz a leitura dos dados e salva na variável msg de tamanho 9
        int data = Wire.read();
        msg[i] = data;
    }

  //Serial.println("recebeu resposta do escravo1");
  }
  if(msg[0]==HIGH){
    flagf1 = true;
  }
    if(msg[1]==HIGH){
    flagf2 = true;
  }
    if(msg[2]==HIGH){
    flagf3 = true;
  }
    if(msg[3]==HIGH){
    flagind = true;
  }
    if(msg[4]==HIGH){
    flagcap = true;
  }
}

void lemsg_fc(){
  //Serial.println(1);
Wire.requestFrom(4,4);
//Serial.println(2);
  //Serial.println("solicitou do escravo 2");
  if(Wire.available()>0){
    //Serial.println(3);
    for(int i=0;i<4;i++){
      delay(2);
      //Serial.println(4);
```

```
      int data = Wire.read();
      //Serial.println(5);
      msg2[i] = data;
      //Serial.println(6);
    }
    //Serial.println(7);
    //Serial.println("recebeu resposta do escravo2");
  }
  //Serial.println(8);

  if(msg2[0]==HIGH){
    flagfc1 = true;
  }
  //Serial.println(9);
    if(msg2[1]==HIGH){
    flagfc2 = true;
  }
  //Serial.println(10);
    if(msg2[2]==HIGH){
    flagfc3 = true;
  }
  //Serial.println(11);
    if(msg2[3]==HIGH){
    flagfc4 = true;
  }
}

void zerar_ident(){
  flagcap = false;
  flagf1 = false;
  flagf2 = false;
  flagf3 = false;
  flagind = false;
}

void zerar_fc(){
  flagfc1 = false;
  flagfc2 = false;
  flagfc3 = false;
  flagfc4 = false;
}
```

Código do escravo A

```
                        // MESTRE ESCRAVO
                        // ESCRAVO

#include <Wire.h>

// bancada
// Sensores de identificação da peça
int f1 = 2;          // sensor optico tamanho pequeno
int f2 = 3;          // sensor optico tamanho médio
int f3 = 4;          // sensor optico tamanho grande
int ind = 5;         // sensor indutivo
int cap = 6;         // sensor capacitivo

// FLAGS
boolean flagf1 = false;
boolean flagf2 = false;
boolean flagf3 = false;
boolean flagind = false;
boolean flagcap = false;

byte msg[ ] = {flagf1, flagf2, flagf3, flagind, flagcap};

void setup(){
  Serial.begin(9600);
  Wire.begin(2);
  Wire.onRequest(requestEvent);
  pinMode(f1,INPUT);
  pinMode(f2,INPUT);
  pinMode(f3,INPUT);
  pinMode(ind,INPUT);
  pinMode(cap,INPUT);
}

void loop(){
     if(digitalRead(f1)==HIGH){
       flagf1 = true;
     }
     if(digitalRead(f2)==HIGH){
       flagf2 = true;
     }
     if(digitalRead(f3)==HIGH){
       flagf3 = true;
     }
     if(digitalRead(ind)==HIGH){
       flagind = true;
     }
     if(digitalRead(cap)==HIGH){
       flagcap = true;
     }
     msg[0] = flagf1;
     msg[1] = flagf2;
     msg[2] = flagf3;
     msg[3] = flagind;
     msg[4] = flagcap;
}

void requestEvent(){
```

```
  Wire.write(msg,5);
     */
     flagf1 = false;
     flagf2 = false;
     flagf3 = false;
     flagind = false;
     flagcap = false;
}
```

Código do escravo B

```
                        // MESTRE ESCRAVO
                        // ESCRAVO
#include <Wire.h>

// bancada
// Sensores fim de curso
int fc1 = 7;         // fim de curso separação peça pequena
int fc2 = 8;         // fim de curso separação peça média
int fc3 = 9;         // fim de curso separação peça grande
int fc4 = 10;          // fim de curso separação peça metálica

// FLAGS
boolean flagfc1 = false;
boolean flagfc2 = false;
boolean flagfc3 = false;
boolean flagfc4 = false;

byte msg[4] = { };

void setup(){
  Serial.begin(9600);
  Wire.begin(4);
  Wire.onRequest(requestEvent);
  pinMode(fc1,INPUT);
  pinMode(fc2,INPUT);
  pinMode(fc3,INPUT);
  pinMode(fc4,INPUT);
  }

void loop(){

     if(digitalRead(fc1)==HIGH){
       flagfc1 = true;
     }
     if(digitalRead(fc2)==HIGH){
       flagfc2 = true;
     }
     if(digitalRead(fc3)==HIGH){
       flagfc3 = true;
     }
     if(digitalRead(fc4)==HIGH){
       flagfc4 = true;
     }
       msg[0] = flagfc1;
       msg[1] = flagfc2;
       msg[2] = flagfc3;
       msg[3] = flagfc4;
}

void requestEvent(){
  Wire.write(msg,4);
     flagfc1 = false;
     flagfc2 = false;
     flagfc3 = false;
     flagfc4 = false;
}
```

Três dispositivos com lógica distribuída

Código do mestre

```
#include <Wire.h>

int f1 = 2;
int f2 = 3;
int f3 = 4;
int ind = 5;
int cap = 6;

unsigned long t_ant = 0;
unsigned long t_atual = 0;
unsigned long intervalo = 20;

byte msg[5] = { };

boolean flagf1 = false;
boolean flagf2 = false;
boolean flagf3 = false;
boolean flagind = false;
boolean flagcap = false;


void setup(){
  Serial.begin(9600);
  Wire.begin();
  pinMode(f1,INPUT);
  pinMode(f2,INPUT);
  pinMode(f3,INPUT);
  pinMode(ind,INPUT);
  pinMode(cap,INPUT);
}

void loop(){
  t_atual = millis();
  if(digitalRead(f1)==HIGH){
    flagf1 = true;
  }
  if(digitalRead(f2)==HIGH){
    flagf2 = true;
  }
  if(digitalRead(f3)==HIGH){
    flagf3 = true;
  }
  if(digitalRead(ind)==HIGH){
    flagind = true;
  }
  if(digitalRead(cap)==HIGH){
    flagcap = true;
  }

   msg[0] = flagf1;
   msg[1] = flagf2;
   msg[2] = flagf3;
   msg[3] = flagind;
   msg[4] = flagcap;
```

```
  for(int i=0;i<5;i++){
    Serial.println(msg[i]);
  }

  if(t_atual-t_ant > intervalo){
    t_ant = t_atual;
    Serial.println("entrou");
    Wire.beginTransmission(0);
    Wire.write(msg,5);
    Wire.endTransmission();
    Serial.println("msg enviada");
    if(flagcap){
      flagf1 = false;
      flagf2 = false;
      flagf3 = false;
      flagind = false;
      flagcap = false;
    }

  }
}
```

Código do escravo A

```
// Biblioteca do cartão SD
#include <SD.h>
File dataFile;

// Timer de interrupção
//#include <TimerOne.h>

#include <Wire.h>

// Atuadores
int v1 = 30;          // válvula 3-2 retorno por mola para primeira separação
int v2 = 31;          // válvula 3-2 retorno por mola para segunda separação

int fc1 = 41;
int fc2 = 42;

byte msg[5] = { };


boolean flagf1 = false;
boolean flagf2 = false;
boolean flagf3 = false;
boolean flagind = false;
boolean flagcap = false;

// TIME
long t_ant = 0;
long t_atual = 0;
long time = 0;

void setup(){
  Wire.begin(0);
  Wire.onReceive(receiveEvent);
  Serial.begin(9600);

  // Definições para utilização do Shield Ethernet e cartão SD no Arduino MEGA
  // Não utilizar as portas 4 e 10
  pinMode(10,OUTPUT);
  digitalWrite(10,HIGH);
  if(!SD.begin(4)){
    Serial.println("Card failed, or not inserted");
  }
  if(SD.exists("datalog.txt")){
    SD.remove("datalog.txt");
  }

  dataFile = SD.open("datalog.txt",FILE_WRITE);
  if(!dataFile){
    Serial.println("erro na abertura do arquivo");
  }
  dataFile.println("type;time");
  dataFile.flush();

  // Definição das PORTAS
  pinMode(v1,OUTPUT);
  pinMode(v2,OUTPUT);
```

```
  digitalWrite(v1,LOW);
  digitalWrite(v2,LOW);

  pinMode(fc1,INPUT);
  pinMode(fc2,INPUT);
}

void loop(){
  //delay(100);
  //Serial.println("to no loop");
  if(flagf1&&flagf2&&flagf3&&!flagind&&flagcap){      // Peça grande nao metálica
  //Serial.println("entrou 1");
    //Serial.println("Peça grande");
    t_ant = millis();
    //delay(200);
    digitalWrite(v2,HIGH);
    zera();
    while(!digitalRead(fc2)){
      // nada acontece
    }
    t_atual = millis();
    //Serial.println("Pistão 3 recuado");
    digitalWrite(v2,LOW);
    time = t_atual - t_ant;
    dataFile.print("grande");
    dataFile.flush();
    dataFile.print(";");
    dataFile.flush();
    dataFile.println(time);
    dataFile.flush();
    //Serial.println("Salvo no cartão");
    zera();
  }

    else if((flagf1||flagf2||flagf3)&&flagind&&flagcap){      // Peça metálica
    //Serial.println("entrou 2");
    //Serial.println("Peça metálica");
    t_ant = millis();
    //delay(200);
    digitalWrite(v1,HIGH);
    zera();
    while(!digitalRead(fc1)){
      // nada acontece
    }
    t_atual = millis();
    digitalWrite(v1,LOW);
    time = t_atual - t_ant;
    dataFile.print("metalica");
    dataFile.flush();
    dataFile.print(";");
    dataFile.flush();
    dataFile.println(time);
    dataFile.flush();
    //Serial.println("Salvo no cartão");
    zera();
  }

  else if((flagf1||flagf2)&&flagcap&&!flagf3&&!flagind){
    //Serial.println("entrou 3");
```

```
    zera();
  }

  delay(10);

}

void receiveEvent(int howMany){
  //Serial.println("chegou");
  if(Wire.available()>0){  // Caso receba uma mensagem
    for(int i=0;i<5;i++){  // faz a leitura dos dados e salva na variável msg de tamanho 9
        int data = Wire.read();
        msg[i] = data;
        //Serial.println(msg[i]);
        //delay(10);
    }

  //Serial.println("recebeu resposta do escravo1");
  }
  if(msg[0]==HIGH){
    flagf1 = true;
  }
    if(msg[1]==HIGH){
    flagf2 = true;
  }
    if(msg[2]==HIGH){
    flagf3 = true;
  }
    if(msg[3]==HIGH){
    flagind = true;
  }
    if(msg[4]==HIGH){
    flagcap = true;
  }

}

void zera(){
  flagcap = false;
  flagf1 = false;
  flagf2 = false;
  flagf3 = false;
  flagind = false;
}
```

Código do escravo B

```
// Biblioteca do cartão SD
#include <SD.h>
File dataFile;

// Timer de interrupção
//#include <TimerOne.h>

#include <Wire.h>

// Atuadores
int v3 = 32;          // válvula 3-2 retorno por mola para primeira separação
int v4 = 33;          // válvula 3-2 retorno por mola para segunda separação

int fc3 = 43;
int fc4 = 44;

byte msg[5] = { };

boolean flagf1 = false;
boolean flagf2 = false;
boolean flagf3 = false;
boolean flagind = false;
boolean flagcap = false;

// TIME
long t_ant = 0;
long t_atual = 0;
long time = 0;

void setup(){
  Wire.begin(0);
  Wire.onReceive(receiveEvent);
  Serial.begin(9600);

  // Definições para utilização do Shield Ethernet e cartão SD no Arduino MEGA
  // Não utilizar as portas 4 e 10
  pinMode(10,OUTPUT);
  digitalWrite(10,HIGH);
  if(!SD.begin(4)){
    Serial.println("Card failed, or not inserted");
  }
  if(SD.exists("datalog.txt")){
    SD.remove("datalog.txt");
  }

  dataFile = SD.open("datalog.txt",FILE_WRITE);
  if(!dataFile){
    Serial.println("erro na abertura do arquivo");
  }
  dataFile.println("type;time");
  dataFile.flush();

  // Definição das PORTAS
  pinMode(v3,OUTPUT);
  pinMode(v4,OUTPUT);
```

```
  digitalWrite(v3,LOW);
  digitalWrite(v4,HIGH);
  delay(20);
  digitalWrite(v4,LOW);

  pinMode(fc3,INPUT);
  pinMode(fc4,INPUT);
}

void loop(){
  //delay(100);
  if(flagf1&&flagf2&&!flagf3&&!flagind&&flagcap){      // Peça média nao metálica
    //Serial.println("Peça media");
    t_ant = millis();
    //delay(200);
    digitalWrite(v3,HIGH);
    digitalWrite(v4,LOW);
    zera();
    while(!digitalRead(fc3)){
      // nada acontece
    }
    t_atual = millis();
    //Serial.println("Pistão 3 recuado");
    digitalWrite(v3,LOW);
    digitalWrite(v4,HIGH);
    time = t_atual - t_ant;
    dataFile.print("media");
    dataFile.flush();
    dataFile.print(";");
    dataFile.flush();
    dataFile.println(time);
    dataFile.flush();
    //Serial.println("Salvo no cartão");
    zera();
  }

    else if((flagf1||flagf2||flagf3)&&!flagind&&flagcap){      // Peça pequena não metálica
    //Serial.println("Peça pequena");
    t_ant = millis();
    //delay(200);
    digitalWrite(v3,LOW);
    digitalWrite(v4,LOW);
    zera();
    while(!digitalRead(fc4)){
      // nada acontece
    }
    t_atual = millis();
    time = t_atual - t_ant;
    dataFile.print("pequena");
    dataFile.flush();
    dataFile.print(";");
    dataFile.flush();
    dataFile.println(time);
    dataFile.flush();
    //Serial.println("Salvo no cartão");
    zera();
  }

  else if(((flagf1&&flagf2&&flagf3)&&flagcap&&!flagf3)||(flagind&&(flagf1||flagf2||flagf3))){
```

```
    zera();
  }

}

void receiveEvent(int howMany){
  if(Wire.available()>0){  // Caso receba uma mensagem
    for(int i=0;i<5;i++){  // faz a leitura dos dados e salva na variável msg de tamanho 9
        int data = Wire.read();
        msg[i] = data;
    }

  //Serial.println("recebeu resposta do escravo1");
  }
  if(msg[0]==HIGH){
    flagf1 = true;
  }
    if(msg[1]==HIGH){
    flagf2 = true;
  }
    if(msg[2]==HIGH){
    flagf3 = true;
  }
    if(msg[3]==HIGH){
    flagind = true;
  }
    if(msg[4]==HIGH){
    flagcap = true;
  }
}

void zera(){
  flagcap = false;
  flagf1 = false;
  flagf2 = false;
  flagf3 = false;
  flagind = false;
}
```

Arquitetura de controle distribuída com comunicação *Zigbee* entre dispositivos

Código do mestre

```
                                        // MESTRE-ESCRAVO
                                        // CÓDIGO MESTRE

// Biblioteca do cartão SD
#include <SD.h>
File dataFile;

// Timer de interrupção
#include <TimerOne.h>

// Atuadores
int v1 = 30;        // válvula 3-2 retorno por mola para primeira separação
int v2 = 31;        // válvula 3-2 retorno por mola para segunda separação
int v3 = 32;        // válvula 5-2 duplo solenoide para terceira separação - AVANÇO
int v4 = 33;        // válvula 5-2 duplo solenoide para terceira separação - RETORNO

byte msg[9] = { };

// FLAGS
boolean flagf1 = false;
boolean flagf2 = false;
boolean flagf3 = false;
boolean flagind = false;
boolean flagcap = false;
boolean flagfc1 = false;
boolean flagfc2 = false;
boolean flagfc3 = false;
boolean flagfc4 = false;

// TIME
long t_ant = 0;
long t_atual = 0;
long time = 0;

void setup(){
  Serial.begin(9600);
  Serial1.begin(9600);
  Timer1.initialize(15000); // microsegundos
  Timer1.attachInterrupt(solicita);

  // Definições para utilização do Shield Ethernet e cartão SD no Arduino MEGA
  // Não utilizar as portas 4 e 10
  pinMode(10,OUTPUT);
  digitalWrite(10,HIGH);
  if(!SD.begin(4)){
    Serial.println("Card failed, or not inserted");
  }
  if(SD.exists("datalog.txt")){
    SD.remove("datalog.txt");
  }

  dataFile = SD.open("datalog.txt",FILE_WRITE);
  if(!dataFile){
    Serial.println("erro na abertura do arquivo");
```

```
  }
  dataFile.println("type;time");
  dataFile.flush();

  // Definição das PORTAS
  pinMode(v1,OUTPUT);
  pinMode(v2,OUTPUT);
  pinMode(v3,OUTPUT);
  pinMode(v4,OUTPUT);
  digitalWrite(v1,LOW);
  digitalWrite(v2,LOW);
  digitalWrite(v3,LOW);
  digitalWrite(v4,LOW);
}

void loop(){

  lemsg();

  if((msg[0]&&!msg[1]&&!msg[2]&&!msg[3])&&msg[4]){      // Peça pequena nao metálica
    t_ant = millis();
    digitalWrite(v4,LOW);
    do{
      lemsg();
      // espera o fim de curso ser acionado
    }while(!msg[8]);
    t_atual = millis();
    time = t_atual - t_ant;
    dataFile.print("pequena");
    dataFile.flush();
    dataFile.print(";");
    dataFile.flush();
    dataFile.println(time);
    dataFile.flush();
   // Serial.println("Salvo no cartão");
  }

  else if(msg[2]&&msg[0]&&msg[1]&&!msg[3]&&msg[4]){      // Peça grande nao metálica
    //Serial.println("Peça grande");
    t_ant = millis();
    digitalWrite(v2,HIGH);do{
      lemsg();
      // espera o fim de curso ser acionado
    }while(!msg[6]);
    t_atual = millis();
    digitalWrite(v2,LOW);
    time = t_atual - t_ant;
    dataFile.print("grande");
    dataFile.flush();
    dataFile.print(";");
    dataFile.flush();
    dataFile.println(time);
    dataFile.flush();
    //Serial.println("Salvo no cartão");
  }

  else if(msg[1]&&msg[0]&&!msg[2]&&!msg[3]&&msg[4]){      // Peça média nao metálica
    t_ant = millis();
    digitalWrite(v4,LOW);
```

```
    digitalWrite(v3,HIGH);
    //Serial.println("Pistão 2 acionado");
    do{
      lemsg();
      // espera o fim de curso ser acionado
    }while(!msg[7]);
    t_atual = millis();
    digitalWrite(v3,LOW);
    digitalWrite(v4,HIGH);
    time = t_atual - t_ant;
    dataFile.print("media");
    dataFile.flush();
    dataFile.print(";");
    dataFile.flush();
    dataFile.println(time);
    dataFile.flush();
  }


    else if(msg[3]&&msg[4]&&(msg[0]||msg[1]||msg[2])){      // Peça metálica
    //Serial.println("Peça metálica");
    t_ant = millis();
    //delay(200);
    digitalWrite(v1,HIGH);
    do{
      lemsg();
      // espera o fim de curso ser acionado
    }while(!msg[5]);
    t_atual = millis();
    digitalWrite(v1,LOW);
    time = t_atual - t_ant;
    dataFile.print("metalica");
    dataFile.flush();
    dataFile.print(";");
    dataFile.flush();
    dataFile.println(time);
    dataFile.flush();
  }

}

void lemsg(){
  if(Serial1.available()>8){  // Caso receba uma mensagem
  //Serial.println("recebeu");
    for(int i=0;i<9;i++){  // faz a leitura dos dados e salva na variável msg de tamanho 9
        int data = Serial1.read();
        msg[i] = data;
      //Serial.print(msg[i]);
    }
    //Serial.flush();
    //Serial.println(" ");
  }
}

void solicita(){  // Solicita status do Escravo
  Serial1.write(1);
 //Serial.println("enviou");
}
```

Código do escravo

```
                    // MESTRE ESCRAVO
                              // ESCRAVO

// bancada
// Sensores de identificação da peça
int f1 = 42;          // sensor optico tamanho pequeno
int f2 = 43;          // sensor optico tamanho médio
int f3 = 44;          // sensor optico tamanho grande
int ind = 45;         // sensor indutivo
int cap = 46;         // sensor capacitivo

// Sensores fim de curso
int fc1 = 47;         // fim de curso separação peça pequena
int fc2 = 48;         // fim de curso separação peça média
int fc3 = 49;         // fim de curso separação peça grande
int fc4 = 50;         // fim de curso separação peça metálica

// FLAGS
boolean flagf1 = false;
boolean flagf2 = false;
boolean flagf3 = false;
boolean flagind = false;
boolean flagcap = false;
boolean flagfc1 = false;
boolean flagfc2 = false;
boolean flagfc3 = false;
boolean flagfc4 = false;

void setup(){
  Serial.begin(9600);
  pinMode(fc1,INPUT);
  pinMode(fc2,INPUT);
  pinMode(fc3,INPUT);
  pinMode(fc4,INPUT);
  pinMode(ind,INPUT);
  pinMode(cap,INPUT);
  pinMode(f1,INPUT);
  pinMode(f2,INPUT);
  pinMode(f3,INPUT);
}

void loop(){
     //Serial.println("loop");
     if(digitalRead(f1)==HIGH){
       flagf1 = true;
     }
     if(digitalRead(f2)==HIGH){
       flagf2 = true;
     }
     if(digitalRead(f3)==HIGH){
       flagf3 = true;
     }
     if(digitalRead(ind)==HIGH){
       flagind = true;
     }
     if(digitalRead(cap)==HIGH){
       flagcap = true;
```

```
    }
    if(digitalRead(fc1)==HIGH){
      flagfc1 = true;
    }
    if(digitalRead(fc2)==HIGH){
      flagfc2 = true;
    }
    if(digitalRead(fc3)==HIGH){
      flagfc3 = true;
    }
    if(digitalRead(fc4)==HIGH){
      flagfc4 = true;
    }


    byte msg[ ] = {flagf1, flagf2, flagf3, flagind, flagcap, flagfc1, flagfc2, flagfc3, flagfc4};

    if(Serial.available()>0){
      //Serial.println("recebeu");
      int data = 0;
      data = Serial.read();
      //Serial.println(data);
      if(data == 1){
        for(int i=0;i<9;i++){
          Serial.write(msg[i]);
          //Serial.print(msg[i]);
        }
        //Serial.println("enviou");
        //Serial.println("msg enviada");
        if(flagfc1||flagfc2||flagfc3||flagfc4){
          //Serial.println("vai resetar as flags");
          flagf1 = false;
          flagf2 = false;
          flagf3 = false;
          flagind = false;
          flagcap = false;
          flagfc1 = false;
          flagfc2 = false;
          flagfc3 = false;
          flagfc4 = false;
        }
      }
    }
    delay(10);
}
```