Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Graduação em Engenharia Mecatrônica
Trabalho de Conclusão de Curso

Sistema de Controle e Supervisão da Estação de Processos MPS PA via Plataforma BeagleBone

Guilherme Pereira Marchioro Bertelli

Orientador: Prof. MSc. Heitor Medeiros Florencio

Natal/RN

Dezembro de 2015.

Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Graduação em Engenharia Mecatrônica

Sistema de Controle e Supervisão da Estação de Processos MPS PA via Plataforma BeagleBone

Guilherme Pereira Marchioro Bertelli

Orientador: Prof. MSc. Heitor Medeiros Florencio

Trabalho de Conclusão de Curso apresentada ao Programa de Graduação em Engenharia de Mecatrônica da Universidade Federal do Rio Grande do Norte como parte dos requisitos para a obtenção do título de Engenheiro Mecatrônico.

Natal/RN

Dezembro de 2015.

Universidade Federal do Rio Grande do Norte
Centro de Tecnologia
Graduação em Engenharia Mecatrônica

Trabalho de Conclusão de Curso apresentado à banca examinadora composta pelos seguintes membros:

Prof. MSc. Heitor Medeiros Florencio (Orientador) . . . . . DCA/UFRN

Prof. Dr. Luiz Affonso Guedes . . . . . .DCA/UFRN

Prof. Dr. Ivanovitch Medeiros Silva . . . . . .IMD/UFRN

Prof. Dr. Diogo Pinheiro Fernandes Pedrosa . . . . . DCA/UFRN

Agradecimentos

Primeiramente, gostaria de agradecer aos meus pais, Marcos e Alaísa, que sempre me providenciaram todo o suporte e serviram de modelo para minha educação e meu caráter pessoal; a minha namorada, Júlia, por toda paciência, apoio e inspiração neste período importante da minha formação; e aos meus amigos de infância Dário e Lorena, por estarem sempre ao meu lado.

Aos meus amigos e colegas de curso Ricardo, Harônio, João Victor, João Marcos, Márcio, Marié, Marilia, Tomaz, Matheus, que me mantiveram motivado durante todos esses anos, principalmente nas madrugadas que passamos estudando no Departamento de Engenharia de Computação e Automação; e aos meus amigos Victor, Sales, Andressa e Ivanízia pelo apoio pessoal e técnico na escrita deste trabalho de conclusão de curso.

Aos professores do curso, em especial a Dr. Pablo, Dr. Darlan, Dr. Diogo, Dr. Affonso e Dr. Márcio, Dr. Ortiz e Dr. Lauro, que acompanharam mais de perto minha graduação, por toda orientação dada durante o período que estive na universidade.

Ao meu orientador, MSc. Heitor, por toda disposição e orientação dada tanto no desenvolvimento do projeto quanto na escrita e revisão deste presente documento.

A todas as pessoas do laboratório de Robótica por sempre estarem me fazendo vivenciar novos aprendizados, sendo o local que mais pude absorver conhecimento durante minha jornada na UFRN.

A todos os professores dos quais tive o prazer de assistir aulas, teóricas ou práticas, durante meu intercâmbio na Escócia, na Glasgow Caledonian University, em Glasgow.

A todos os professores, bolsistas e funcionários do LAUT, pela companhia, apoio e aprendizados durante meu estágio curricular no semestre de 2015.2 e a equipe que me acompanhou na CLBI (Centro de Lançamento Barreira do Inferno) durante minha iniciação técnologica, em especial Dr. Glauberto, Dolvim e Gutemberg.

Para finalizar, gostaria de agradecer a todas as pessoas que contribuíram de uma forma direta ou indireta para a conclusão deste trabalho e do curso.

Resumo

Este trabalho de conclusão de curso apresentará o desenvolvimento de um sistema de automação para a estação de processos MPS PA, utilizando a plataforma de desenvolvimento BeagleBone Black. O sistema supervisório adotado foi o software livre ScadaBR, com a troca de dados feita via Modbus Serial. A programação do sistema embarcado, para realizar o controle da planta e aquisição e conversão de dados para o formato Modbus, foi feita na linguagem Python. Para o interfaceamento da BeagleBone com a planta, fez-se necessário o projeto de um circuito de conexão.

Palavras-chave: Automação; BeagleBone Black; Modbus Serial; Controle; Supervisão; Estação MPS PA.

Abstract

This end of course project presents the development of an automated system for the MPS PA process workstation, making use of the BeagleBone Black development platform. The supervisory system adopted was the open-source software ScadaBR, with the communication done via Modbus Serial. The programming in the embedded system, to accomplish the control of the plant and the acquisition and conversion of data to the Modbus format, was implemented in Python language. For the interfacing of the BeagleBone and the workstation, the design of a connection circuit was necessary.

Key-words: Automation; BeagleBone Black; Modbus Serial; Control; Supervision; MPS PA Workstation.

Sumário

Sumário i
Lista de Figuras iii
Lista de Tabelas e Algoritmos v
Lista de Abreviaturas vi
Capítulo 1 – Introdução 1
1.1 Motivações 2
1.2 Objetivos 4
1.3 Organização do Trabalho 5
Capítulo 2 – Automação Industrial 6
2.1 Introdução 6
2.2 Sensores e Atuadores 9
2.3 Sistemas de Controle 12
2.3.1 Controle PID 13
2.3.1.2 Controlador Digital 15
2.3.2 Dispositivos para Controle Automático 16
2.4 Sistemas Supervisórios 19
2.5 Protocolos de Comunicação 22
2.5.1 Modbus 23
2.5.1.1 Modo ASCII 24
2.5.1.2 Modo RTU 25
2.5.1.3 Principais Funções 26
Capítulo 3 – Desenvolvimento 27
3.1 Componentes 28
3.1.1 BeagleBone Black 28
3.1.2 ScadaBR 29
3.1.3 Estação MPS PA 30

3.2 Métodos 31
3.2.1 Comunicação Beagle – MPS PA 31
3.2.1.1 Implementação do Controlador 34
3.2.2 Comunicação Beagle – ScadaBR 36
3.2.2.1 Modbus Serial e Configuração dos *Data Points* 37
Capítulo 4 – Resultados 41
4.1 Desempenho do Sistema 42
4.2 Perda de Pacotes 44
Capítulo 5 – Considerações Finais 45
Referências Bibliográficas 47
Apêndice A – Detalhamento dos pinos da BeagleBone Black 49
Apêndice B – Diagrama de P&D da Estação MPS PA 50

Lista de Figuras

Figura 1.1	Modelo de Comunicação com a Estação MPS PA via Easyport	3
Figura 1.2	Modelo de Comunicação com a Estação MPS PA via CLP da Siemens	3
Figura 1.3	Modelo de Controle e Supervisão da Estação MPS PA via BeagleBone	4
Figura 2.1	Pirâmide Hierárquica da Automação Industrial	7
Figura 2.2	Diagrama Simplificado de um Controle Industrial	9
Figura 2.3	Sensores Industriais Típicos **(a)** PT100, da APCS **(b)** SMAR LD302 **(c)** FESTO BE.SI.0193	10
Figura 2.4	Modelo de um Sistema de Medição	10
Figura 2.5	Transmissores de Tecnologia sem Fio **(a)** Rosemount 848T **(b)** Siemens SITRANS P280 **(c)** Honeywell XYR 6000 **(d)** Yokogawa YTA510	11
Figura 2.6	Exemplos de Atuadores Industriais **(a)** Motor Trifásico WEG 22 PLUS **(b)** Cilíndro Pneumático Univer ISO 15552	12
Figura 2.7	Controle em Malha Aberta	12
Figura 2.8	Controle em Malha Fechada	13
Figura 2.9	Sistema de Controle Digital	15
Figura 2.10	CLP Modicon M340, da Schneider Electric	17
Figura 2.11	Centro de Usinagem Discovery 1250, da ROMI	18
Figura 2.12	PAC APAX-5000, da Advantech	19
Figura 2.13	Tela típica de supervisão (Software SuperView)	21
Figura 2.14	Localização do barramento de campo na hierarquia da planta	22
Figura 3.1	Esquemático do Projeto	27
Figura 3.2	BeagleBone Black	28
Figura 3.3	Tela de Supervisão do ScadaBR (Sistema de Ar Condicionado)	30

Figura 3.4 Estação de Processos MPS PA 31
Figura 3.5 Esquema de Interfaceamento BeagleBone – Placa I/O 32
Figura 3.6 Amplificador Operacional não-inversor 32
Figura 3.7 Circuito Integrado LM741 33
Figura 3.8 Divisor de Tensão 33
Figura 3.9 Comunicação BeagleBone – ScadaBR 36
Figura 3.10 Esquema do Circuito de conversão RS232-TTL 37
Figura 4.1 Interface desenvolvida no ScadaBR 41
Figura 4.2 Tela de supervisão para uma referência de 12 cm 42
Figura 4.3 Tela de supervisão para uma referência de 10 cm 42
Figura 4.4 Tela de supervisão para uma referência de 12 cm com tanque preenchido em 7 cm 43
Figura 4.5 Alertas de perdas de pacote 44

Lista de Tabelas e Algoritmos

Tabela 2.1	Cabeçalho do Modo ASCII	24
Tabela 2.2	Cabeçalho do Modo RTU	25
Tabela 3.1	Especificações da BeagleBone Black	28
Algoritmo 3.1	Implementação do Controlador	35
Algoritmo 3.2	Comunicação Modbus Serial BeagleBone – ScadaBR	39

Lista de Abreviaturas

A/D – Analógico/Digital

ASCII – *American Standard Code for Information Interchange*

ASI – *Actuator Sensor Interface*

BI – *Business Intelligence*

CAD – *Computer Aided Design*

CAE – *Computed Aided Engineering*

CAM – *Computer Aided Manufacturing*

CAN – *Controller Network Area*

CI – Circuito Integrado

CIM – *Computer Integrated Manufacturing*

CLP – Controlador Lógico Programável

CNC – *Computer Numeric Control*

CRC – *Cyclic Redundancy Check*

DDC – *Direct Digital Control*

I/O – *Inputs/Outputs*

JIT – *Just in Time*

LRC – *Longitudinal Redundancy Check*

ms - Milissegundos

OPC - *Object Linking and Embedding for Process Control*

P&ID – *Piping and Instrumentation Diagram*

PWM – *Pulse Width Modulation*

RTU – *Remote Terminal Unit*

s - Segundos

SCADA - *Supervisory Control and Data Acquisition*

SFC – *Sequential Fuction Chart*

TQM – *Total Quality Management*

TTL – *Transistor-Transistor Logic*

Capítulo 1

Introdução

As primeiras iniciativas do homem para mecanizar atividades manuais ocorreram desde a pré-história, com invenções como a roda, o moinho movido por vento ou força animal e as rodas d'água, demonstrando criatividade do homem para poupar esforço. Porém, a automação só ganhou destaque na sociedade quando o sistema de produção agrário e artesanal transformou-se em industrial, a partir da segunda metade do século XVIII, inicialmente na Inglaterra. [6]

No século XX, a automação passou a contar com computadores, servomecanismos e controladores programáveis, que são alicerce de basicamente toda tecnologia de automação contemporânea. [6]

Na década de 1980, as pesquisas visaram à integração e/ou automatização dos diversos elementos de projeto e manufatura com o objetivo de criar a fábrica do futuro. O foco das pesquisas foi expandir os sistemas CAD/CAM. Desenvolveu-se também o modelamento geométrico tridimensional com mais aplicações de engenharia (CAE). Alguns exemplos dessas aplicações são a análise e simulação de mecanismos, o projeto e análise de injeção de moldes e a aplicação do método dos elementos finitos. [6, 12, 13]

Hoje, os conceitos de integração total do ambiente produtivo com o uso dos sistemas de comunicação de dados e novas técnicas de gerenciamento estão se disseminando rapidamente. O CIM (*Computer Integrated Manufacturing*) já é uma realidade. [12]

A automação de sistemas vem se tornando um procedimento padronizado em ambientes industriais, principalmente dependendo da escala do processo, devido a suas inúmeras vantagens relacionadas a custo e segurança. Um processo devidamente automatizado reduz mão-de-obra, tempo de produção, aumenta escala de produção, anula falhas humanas e detecta erros com muito mais eficácia. [6]

Dada a importância de se ter um sistema automatizado na indústria, neste trabalho de conclusão de curso será apresentado o desenvolvimento de um sistema de automação de uma planta didática que abrange desde a aquisição dos dados dos sensores, controle do nível de um tanque até a supervisão das variáveis do processo. Sistemas de automação de processos industriais, incluindo o controle do nível de tanques, têm sido desenvolvidos, na sua grande maioria, utilizando controladores lógicos programáveis difundidos no meio industrial, que podem apresentar um menor poder computacional, se comparados com as plataformas de desenvolvimento de sistemas embarcados que vem surgindo nos últimos anos.

Com isso, neste presente trabalho, a estratégia de controle e a comunicação com o sistema de supervisão foram implementadas em um computador *single-board* de *hardware* livre.

1.1 Motivações

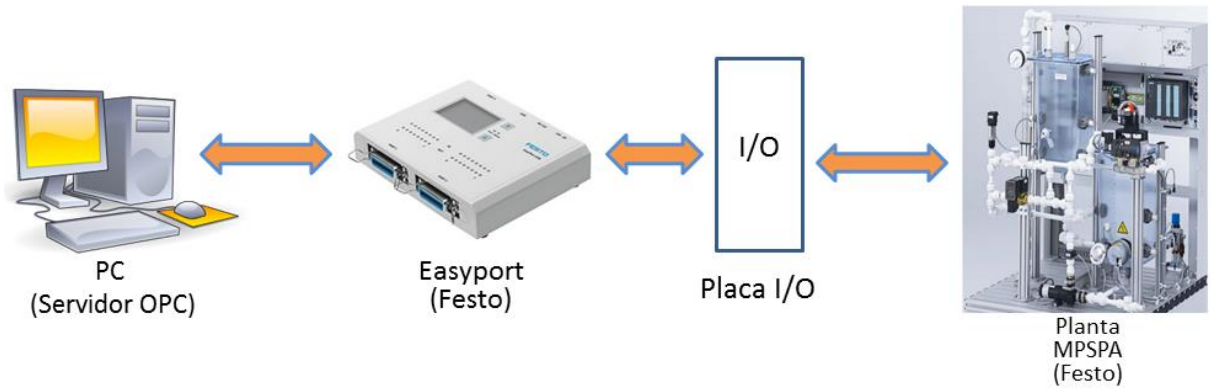
Atualmente, os elementos de um sistema de automação, sensores, atuadores e controladores, interagem por meio de sinais elétricos padrões da indústria, como sinais de corrente de 4 à 20 mA, sinais de tensão de 0 à 10 V ou até mesmo sinais digitais baseados em protocolos de comunicação. Diferentemente dos controladores lógicos programáveis, as plataformas de desenvolvimento de sistemas embarcados não possuem interfaces de comunicação padrões aos instrumentos de campo. Diante disso, a implementação de um sistema de controle e supervisão de uma planta de processos em um sistema embarcado depende da criação de interfaces de comunicação com os elementos de campo e com o sistema de supervisão.

A planta didática utilizada neste trabalho, a estação de processos MPS PA da empresa Festo, dispõe de uma placa de conexões com os elementos de todos os processos da estação, que pode ser utilizada para controlar e supervisionar tais processos. Atualmente, a empresa Festo disponibilizou duas formas de conexão com a planta: por meio de um CLP da Siemens ou por meio de uma interface entre os sinais da estação e um computador, o EasyPort.

A comunicação com a estação MPS PA por meio da interface EasyPort da Festo é apresentada no modelo mostrado na Figura 1.1. Os dados dos elementos da estação são recebidos e enviados pela interface EasyPort através de cabos com conectores do

padrão IEEE 488. Tais dados são recebidos e enviados para o computador através de uma conexão USB. O computador, por sua vez, pode acessar os dados através de softwares como FluidSIM ou ActiveX Control, que pode ser programado por Visual C++ ou LabVIEW. Além disso, um servidor OPC com os dados é disponibilizado com a instalação do EasyOPC, permitindo que a comunicação com a estação seja realizada por qualquer software que tenha um cliente OPC. Por exemplo, é possível realizar o controle de nível de um tanque da estação MPS PA através do ambiente Simulink do Matlab, que dispõe de um cliente OPC.

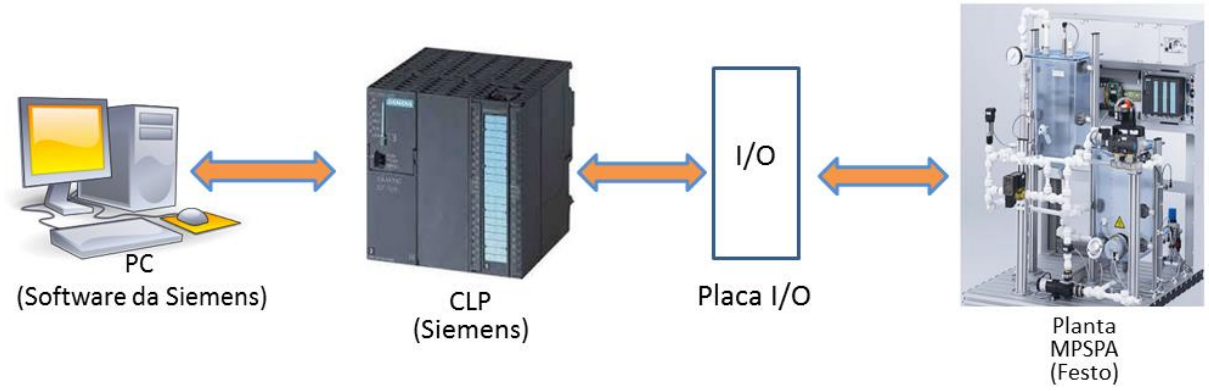
Figura 1.1 – Modelo de Controle e Supervisão da Estação MPS PA via Easyport



Fonte: Autor, 2015.

O segundo método para controlar e supervisionar a planta MPS PA consiste na comunicação por meio de um CLP da Siemens, o controlador S7-300, que pode ser programado por software proprietário da empresa, como mostrado na Figura 1.2.

Figura 1.2 – Modelo de comunicação com a estação MPS PA via CLP da Siemens



Fonte: Autor, 2015.

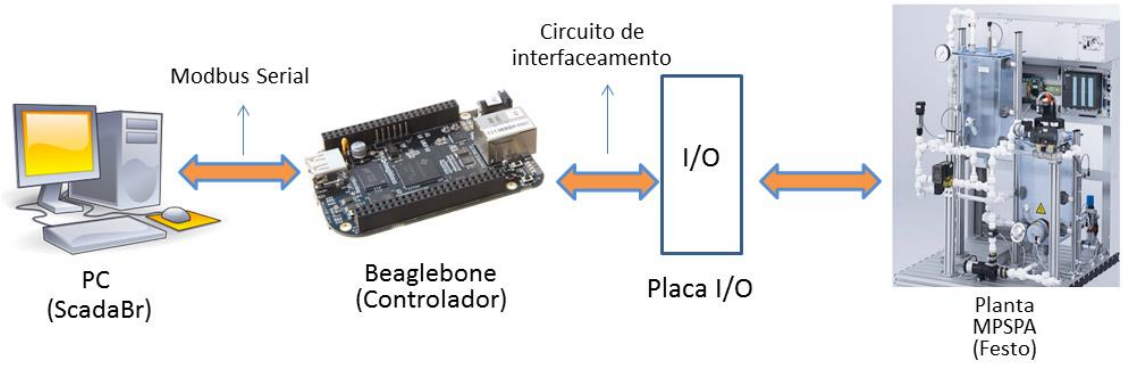
Este trabalho tem como motivação o desenvolvimento de um sistema de controle e monitoramento para a estação MPS PA com base em hardware e software de uso livre, dando ao usuário maior liberdade no projeto. O hardware será responsável pelo controle e aquisição dos dados do processo, enquanto que o computador será responsável pela supervisão do processo, utilizando software de supervisão de uso livre, como o ScadaBR. Porém, para a execução de tal procedimento, o hardware deve ter uma comunicação com o software de supervisão.

Nesse sentido, outro ponto que serve como motivação é a falta de documentação à respeito da implementação de protocolos de comunicação industrial, como o protocolo Modbus, em plataformas de sistemas embarcados para troca de informações com softwares SCADA.

1.2 Objetivos

As duas formas de comunicação com a estação MPS-PA descritas apresentam desvantagens em relação ao uso de hardware e software proprietários das empresas Festo e Siemens. Nesse sentido, o presente trabalho propõe um modelo de controle e supervisão da estação MPS-PA utilizando a plataforma BeagleBone Black (hardware livre), como mostrado na Figura 1.3.

Figura 1.3 – Modelo de Controle e Supervisão da Estação MPS PA via BeagleBone



Fonte: Autor, 2015.

A comunicação BeagleBone – PC será realizada através do protocolo Modbus, via serial. A estratégia de controle, a aquisição de dados a construção de mensagens no formato do Modbus foram implementadas na BeagleBone, utilizando a linguagem Python. Para que seja possível receber e enviar sinais para a placa de entradas e saídas (I/O) da planta, serão desenvolvidos circuitos de interface entre a Beaglebone e os sinais

da planta MPS PA, que convertem os sinais de tensão da placa I/O em sinais TTL compatíveis com portas digitais da plataforma, e vice-versa.

Essa proposta seria útil para livrar a necessidade do uso de softwares proprietários pagos. Com o controlador e todo o trafego de dados sendo trabalhado na BeagleBone, o usuário passa a ter muito mais liberdade de controle e manipulação da informação e, considerando o uso didático da planta, também abre espaço para futuros trabalhos. O mesmo pode ser dito da interface da supervisão.

1.3 Organização do Trabalho

Este trabalho está organizado da seguinte forma: este primeiro capítulo introdutório contextualiza, comenta a respeito do que se tem feito na planta utilizada e expõe a proposta do trabalho. No Capítulo 2 é feita uma revisão teórica dos conteúdos que servem como base para a execução do projeto: automação industrial, sistemas de controle, sistemas supervisórios e protocolos de comunicação. O Capítulo 3 mostra o desenvolvimento do trabalho, expondo os componentes, implementações, diagramas eletrônicos e execução da comunicação. O Capítulo 4 mostra os resultados e suas análises e, por fim, no Capítulo 5 se encontram as considerações finais do projeto.

Capítulo 2

Automação Industrial

Este capítulo irá abordar toda a fundamentação teórica que se faz necessário para a compreensão do trabalho, fazendo uma revisão sobre automação industrial e seus níveis hierárquicos (sensores e atuadores, sistemas de controle e sistemas supervisórios) e protocolos de comunicação.

2.1 Introdução

Desde tempos remotos, a automação vem a desenvolver estratégias e mecanismos que permitem libertar-se do trabalho de origem muscular e animal e das tarefas pesadas, rotineiras, perigosas e pouco precisas. Tem conseguido, simultaneamente, maiores velocidades na execução das tarefas, menores tempos de paragem, menor número de acidentes e a obtenção de produtos cada vez maiores e mais uniformes na qualidade. O objetivo, em processos produtivos, é otimizar três fatores: matéria prima, informação e energia. [8,13]

A automação, inicialmente, era caracterizada por pequenas ilhas com operações automatizadas, onde o fator humano era fundamental como elemento integrador e sincronizador de todas as operações. Este estágio caracterizava-se, entre outros fatores, por um elevado número de operários e layouts não otimizados. [6]

Caminhou-se depois para soluções de automatização centralizada. Nestas, todas as informações eram centralizadas em um único local, onde são tomadas todas as decisões e de onde partem todas as ordens. Com este nível, os layouts foram melhorados e um número de operários bastante reduzido. [6]

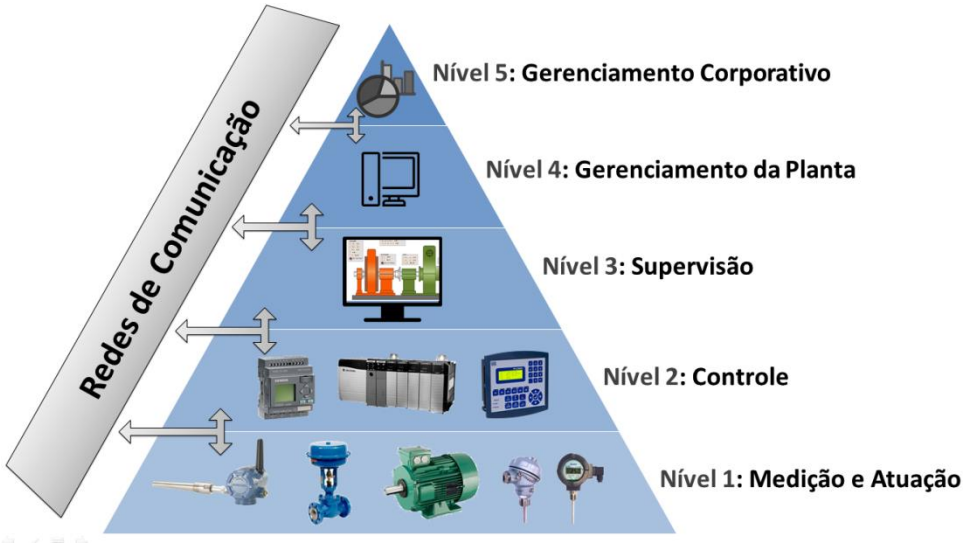
Com o surgimento de instrumentos pneumáticos na década de 1940, foi possível realizar o monitoramento e controle de forma automatizada. O operador já não precisava abrir ou fechar todas as válvulas manualmente, reduzindo o tempo de monitoramento do processo. Inicialmente, os controladores eram instalados próximos aos transmissores e às válvulas de controle, que pertenciam a sua malha de controle.

Com o passar dos anos, estes controladores de campo começaram a ser reunidos em uma sala que centralizada os elementos de controle, que passou a ser chamada de sala de controle e monitoramento de processos. [6, 13]

Após a década de 60, com o desenvolvimento e a utilização crescente de unidades de processamento de informação, as funções de condução dos processos foram sendo cada vez mais distribuídas pelo terreno e junto dos locais onde são necessárias, surgindo assim o que é atualmente designado de Sistema Digital de Controle Distribuído (SDCD). Este nível de automatização caracteriza-se por uma gestão global e integrada da informação, pela inserção de máquinas de controle numérico (CNC), de manipulação (Robôs), manuseamento automático de materiais, pela redução drástica do número de operários (nos setores de produção, principalmente), utilizando os conceitos de JIT (*Just-in-Time*) e TQM (*Total Quality Management*), e ainda por um uso mais intensivo dos equipamentos. [6]

Com a integração destes conceitos de tecnologia de produção, tem-se um sistema denominado de CIM, que é um sistema típico da automação industrial, presente na atualidade. O conjunto total das funções a serem implementados por um sistema de controle distribuído têm diferentes exigências ao nível da rapidez de atuação e da importância estratégica da mesma. Assim, as ações de um sistema de automação surgem agrupadas em vários níveis hierárquicos, havendo características funcionais e temporais bem específicas a cada nível. A pirâmide hierárquica da automação industrial pode ser vista na Figura 2.1.

Figura 2.1 – Pirâmide Hierárquica da Automação Industrial



Fonte: Autor, 2015.

O nível 1 refere-se a aquisição de dados e controle manual**,** majoritariamente composto por dispositivos de campo, como atuadores, sensores, transmissores e outros componentes presentes na planta.

O segundo nível compreende o controle individual: equipamentos que realizam o controle automatizado das atividades da planta. Nele se encontram CLPs, SDCDs (Sistema Digital de Controle Distribuído) e relés. Porém, existem tecnologias que permitem que o controle de determinado processo seja realizado por dispositivos de campo, o nível 1.

O terceiro nível engloba o controle de célula, supervisão e otimização do processo. Destina-se a supervisão dos processos executados por uma determinada célula de trabalho em uma planta. Na maioria dos casos, também obtém suporte de um banco de dados com todas as informações relativas ao processo.

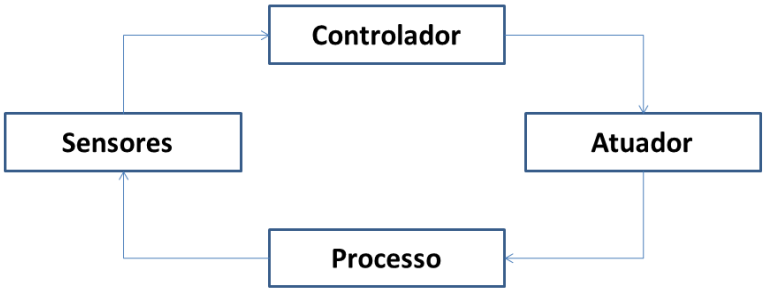
O nível 4, então, é responsável pela programação e planejamento da produção. Auxilia tanto no controle de processos industriais quanto na logística de suprimentos. Este nível geralmente denomina-se de Gerenciamento da Planta.

O quinto nível, por fim, se encarrega da administração dos recursos da empresa. Neste nível encontram-se softwares para gestão de venda, gestão financeira e BI (*Business Intelligence*) para ajudar na tomada de decisões que afetam a empresa como um todo.

Uma rede de comunicação interliga todos os níveis do sistema, como, por exemplo, a conexão entre os dispositivos de campo, controladores e sistemas supervisórios, bem como conecta os elementos dentro de um mesmo nível, como a comunicação entre sensores e atuadores. A interligação e comunicação, baseada em protocolos de comunicação industriais, entre todos os dispositivos em diferentes níveis hierárquicos, é essencial para o desempenho desejado do sistema de automação.

Tipicamente, na indústria, as principais variáveis à serem medidas e controladas são pressão, temperatura, vazão e nível. Para realizar o controle de tais variáveis em uma planta industrial, baseando-se no modelo padrão de um sistema de controle, dispõe-se de determinados componentes: processo, sensor, controlador e atuador, cuja relação pode ser vista no diagrama simplificado na Figura 2.2.

Figura 2.2 – Diagrama Simplificado de um Controle Industrial



Fonte: Autor, 2015.

2.2 Sensores e Atuadores

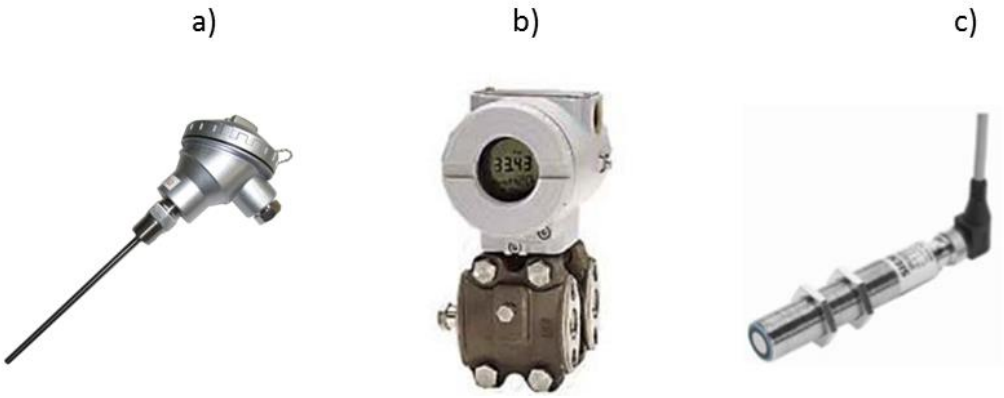
Sensores são os dispositivos utilizados para medição das variáveis do processo que, na maioria dos casos da indústria, são pressão, vazão, temperatura e nível. Usualmente, recebem um sinal elétrico e geram a grandeza a ser medida proporcional a este sinal. [12]

O princípio de funcionamento dos sensores são variados e os mais comumente utilizados são:

- Mecânicos: geram uma saída de grandeza mecânica (movimento, força, etc.) proporcional a grandeza medida;
- Resistivos: converte a variável de processo medida em uma variação de resistência elétrica. Um exemplo é o sensor de temperatura PT100 da APCS, mostrado na Figura 2.3(a), que mede a temperatura baseado na variação da resistência de um condutor metálico;
- Magnéticos: Converte a variável de processo medida em uma força eletromotriz induzida em um condutor pela variação no fluxo magnético, na ausência de excitação. A variação no fluxo feita é usualmente pelo movimento relativo entre um eletromagneto e um magneto ou porção de material magnético.
- Indutivos: converte a variável de processo em uma variação da auto-indutância elétrica de uma bobina;
- Capacitivos: converte a variável do processo em uma variação de capacitância. Um exemplo é o sensor de pressão mostrado na figura 2.3(b), que utiliza uma célula capacitiva para medir a pressão diferencial entre dois pontos;
- Ópticos: converte a variável de processo medida em uma variação de resistência elétrica (ou condutância) de um material semicondutor devido à variação da quantidade de luz incidente;

- Ultrassônicos: operam com a emissão e reflexão de um feixe de ondas acústicas. A saída comuta quando este feixe é refletido ou interrompido pelo material a ser detectado. Por exemplo, o sensor da FESTO, visto na Figura 2.3(c);

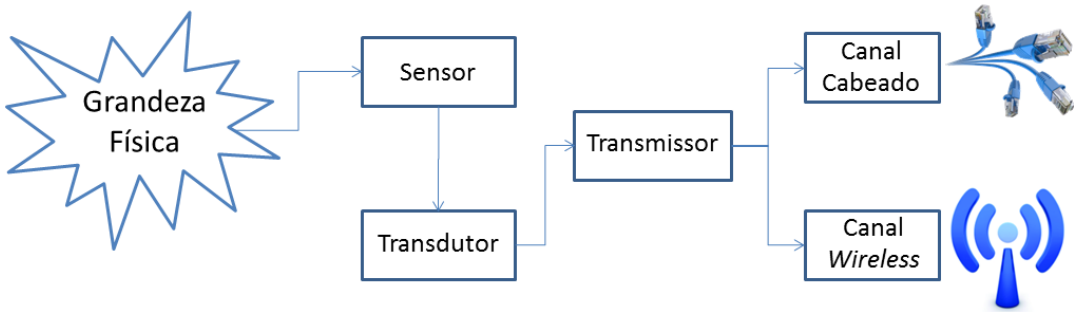
Figura 2.3 – Sensores Industriais Típicos **(a)** PT100, da APCS **(b)** SMAR LD302 **(c)** FESTO BE.SI.0193



Fonte: APCS. Disponível em: <www.apcs.net.au>. Acessado em 9 de dezembro de 2015; SMAR. Disponível em: <www.smar.com>. Acessado em 9 de dezembro de 2015; FESTO. Disponível em: www.festo.com. Acessado em 9 de dezembro de 2015.

Sabe-se que o sensor é o elemento responsável por medir a grandeza física no processo, porém o sinal gerado por ele não é o sinal que o controlador recebe. Os controladores compreendem apenas sinais elétricos de acordo com sua tecnologia de comunicação. Para que o valor de saída do sensor seja compreendido pelos controladores, existem mais dois elementos presentes no sistema de medição: o transdutor e o transmissor, formando o modelo de medição apresentado na Figura 2.4.

Figura 2.4 – Modelo de um Sistema de Medição



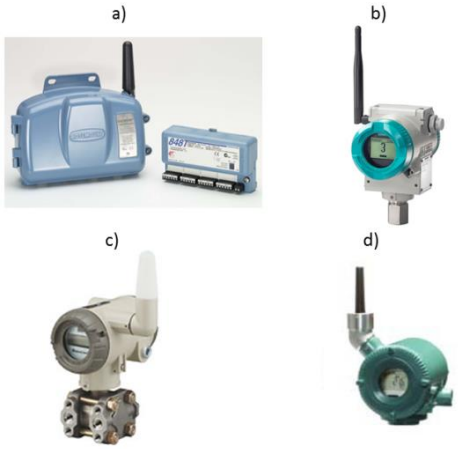
Fonte: Autor, 2015.

Após a medição da grandeza física via o sensor, o transdutor fica responsável por converte o sinal gerado pelo sensor em um sinal elétrico para o transmissor, que, por

sua vez, é responsável por enviar o sinal elétrico via canal com fio ou sem fio. A forma do sinal de medição enviado pelo transmissor depende da tecnologia de comunicação do equipamento, que pode ser através de sinais pneumáticos, sinais de corrente de 4 a 20 mA, sinais de tensão ou através de um sinal elétrico baseado em um protocolo de comunicação, como sinais da tecnologia HART, Foundation Fieldbus, Profibus, CAN, entre outros. Por exemplo, o dispositivo da figura 2.3(b), chamado anteriormente de sensor de pressão LD302, é um transmissor de pressão da tecnologia de comunicação Foundation Fieldbus, que engloba os três elementos: sensor, trandustor e transmissor. [12]

Além disso, desde 2009, existem grupos e empresas concentrados na produção de equipamentos de medição com tecnologia de comunicação sem fio. Por exemplo, a figura 2.5 apresenta quatro transmissores *wireless*: o transmissor de temperatura 848T da Emerson Rosemount e o transmissor de pressão SITRANS P280 da Siemens, que são baseados no protocolo *WirelessHART*, o transmissor de posição XYR 6000 da Honeywell e o transmissor de temperatura YTA510 da Yokogawa, que são baseados no protocolo ISA 100.11.a.

Figura 2.5 – Transmissores de Tecnologia sem Fio **(a)** Rosemount 848T **(b)** Siemens SITRANS P280 **(c)** Honeywell XYR 6000 **(d)** Yokogawa YTA510



Fonte: EMERSON. Disponível em: <www2.emersonprocess.com/>. Acessado em 9 de dezembro de 2015; SIEMENS. Disponível em: <www.siemens.com>. Acessado em 9 de dezembro de 2015; HONEYWELL. Disponível em: < honeywell.com/>. Acessado em 9 de dezembro de 2015; YOKOGAWA. Disponível em: www.yokogawa.com/>. Acessado em 9 de dezembro de 2015.

Os atuadores são os dispositivos que atuam alterando a saída do processo com base no valor de referência desejado, amplificando a energia ou transformando um sinal de energia elétrica em outras formas de energia. Cilindros pneumáticos e hidráulicos, motores e válvulas são exemplos comuns. Alguns exemplos de atuadores podem ser vistos na Figura 2.6. [8]

Figura 2.6 – Exemplos de Atuadores Industriais **(a)** Motor Trifásico WEG 22 PLUS **(b)** Cilíndro Pneumático Univer ISO 15552

(a) (b)

Fonte:WEG. Disponível em: <www.weg.net>. Acessado 18 de novembro de 2015; UNIVER. Disponível em: <http://www.univer-group.com/>. Acessado em 18 de novembro de 2015.

2.3 Sistemas de Controle

Um sistema de controle se refere a utilização de métodos para ajustar o fluxo de energia da entrada para a saída de um sistema ou processo de modo a atender os critérios de performance desejados. São divididos em sistemas de controle de malha aberta e sistemas de controle de malha fechada. [14]

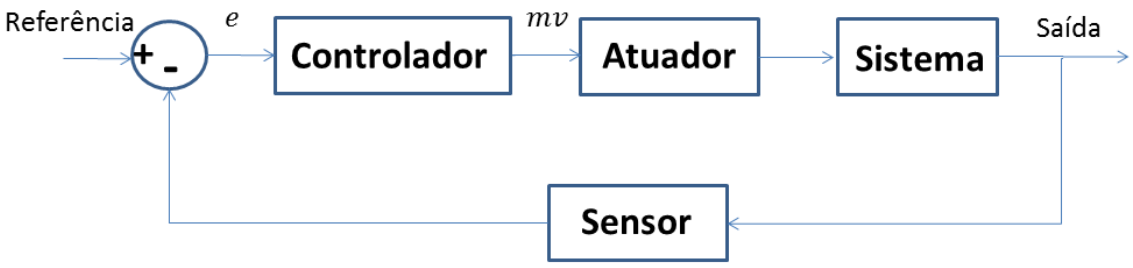
Um sistema de controle em malha aberta é aquele cuja resposta não possui nenhum influência sobre a entrada, como pode ser visto no esquemático da Figura 1.5. Tal controle possui as vantagens de ser mais barato, por exigir menos componentes, apresentar maior facilidade na construção e implementação e não apresentarem problemas de estabilidade. Porém, os erros causados por distúrbios e mudanças na calibração podem gerar uma resposta diferente da esperada, requerendo regulação periódica. [14]

Figura 2.7 – Controle em Malha Aberta

Fonte: Autor, 2015.

Os sistemas de controle de malha fechada tem sua saída continuamente monitorada por um sensor de forma a realimentá-la de volta a entrada. O valor real da saída é subtraído do valor de referência, resultando em um sinal de erro, que será processado pelo controlador para gerar um sinal de controle que tem por objetivo minimizá-lo. [9, 11]

Figura 2.8 – Controle em Malha Fechada



Fonte: Autor, 2015.

Em ambientes industriais, na grande maioria dos processos, utiliza-se o controle em malha fechada, por possuir uma menor sensibilidade a distúrbios, ser mais preciso na comparação dos valores desejados e reais, menos sensível na variação das características dos componentes, apesar da perda de ganho, e possibilidade de instabilidade. [8, 17]

Existem várias estratégias de controle disponíveis no mercado, porém a estratégia mais comumente utilizada na indústria é o controle PID (Proportional + Integrative + Derivative) ou suas derivações (PI, PI-D, etc) devido a seus resultados serem adequados para a grande maioria de sistemas industriais. Nessa estratégia, o sinal de controle mv, gerado pelo controlador, pode ser afetado por três ações de controle: proporcional, integral e derivativa

2.3.1 Controle PID

Considerando $r(t)$ como o sinal de referência, $e(t) = r(t) - y(t)$ é o erro medido, $y(t)$ o valor medido pelo sensor e $u(t)$ o sinal de controle, definem-se as ações de controle.

Ação Proporcional:

A ação proporcional é um amplificador, com ganho ajustável (K), cujo aumento diminui o erro de regime permanente e pode vir a tornar o sistema oscilatório. È um

controlador bastante limitado, melhorando o regime permanente mas piorando o transitório. [11]

$$u_P(t) = K_P e(t) \tag{2.1}$$

Ação Proporcional + Integral

O controle PI soma ao termo proporcional a integral no tempo do erro, zerando o erro de regime (pois aumenta a ordem do sistema). Pelo aumento da ordem do sistema, têm-se a possibilidade de novas instabilidades serem geradas e pode degradar o desempenho do controlador em malha fechada. É utilizado para corrigir problemas de regime permanente. [11]

$$u_{PI}(t) = K_P e(t) + \frac{1}{\tau_i}\int e(\tau)d\tau \tag{2.2}$$

Onde τ_i é o tempo de integração, especificado pelo operador. Usualmente, para o controle PI mostra-se suficientemente adequado para automação industrial.

Ação Proporcional + Derivativa

O controle derivativo leva em consideração a taxa de variação do erro e é utilizado quando têm-se uma resposta de regime transitório insatisfatória. Adiciona ao sistema um efeito de antecipação, fazendo com que o mesmo reaja não só a magnitude do sinal, como sua tendência, iniciando uma ação corretiva mais cedo. Têm a desvantagem de amplificar o ruído, podendo causar um efeito de saturação nos atuadores. [11]

$$u_{PD}(t) = K_P e(t) + \tau_d \frac{d}{dt} e(t) \tag{2.3}$$

Onde τ_dé a constante derivativa.

Ação Proporcional + Integral + Derivativa

O controle PID é utilizado quando tanto a resposta transitória quanto permanente são insatisfatórias, assumindo as vantagens de cada uma das ações de controle, tornando-o mais robusto. [11]

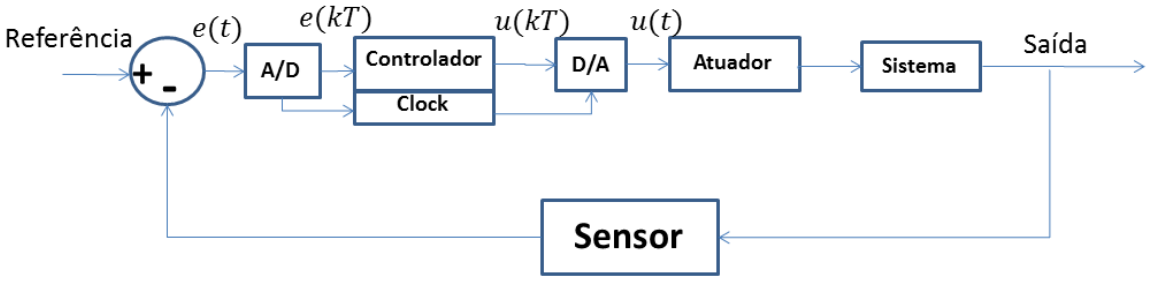
$$u_{PID}(t) = K_P e(t) + \frac{1}{\tau_i}\int e(\tau)d\tau + \tau_d \frac{d}{dt}e(t) \tag{2.4}$$

Contudo, estas equações são utilizadas para projeto de controladores analógicos. Para a implementação digital de um controlador PID, deve-se converter as equações para o domínio discreto.

2.3.1.1 Controlador Digital

A implementação de um controlador digital consiste na conversão das equações analógicas para o domínio discreto. A Figura 2.9 exemplifica o processo. O sinal analógico deve passar por uma A/D, com determinado tempo de amostragem, gerando um sinal de controle digital.

Figura 2.9 – Sistema de Controle Digital



Fonte: Autor, 2015.

A conversão A/D é caracterizada pela discretização de um sinal contínuo através da definição de um tempo de amostragem e quantização do sinal. O tempo de amostragem refere-se ao tempo entre cada um dos pulsos digitais que irão compor o sinal analógico, e a quantização está relacionada a precisão do sinal digital, dependendo do número total de valores que o dispositivo de conversão pode gerar, com base na quantidade de bits do processador. [9] [11].

Esse sinal é, então, comumente, convertido de volta para analógico para ser enviado para os atuadores. O tempo de amostragem escolhido deve ser condizente com o tempo de operação do processo, de forma que atuador funcione corretamente.

Três termos compõem a equação 2.4. Transformando-as para o domínio discreto, têm-se:

$$K_p * e(t) \Leftrightarrow K_p * e[n] \tag{2.5}$$

$$\frac{1}{\tau_i}\int e(\tau)d\tau \Leftrightarrow \frac{t_{amostragem}}{\tau_i} * (e[n] + e[n-1]) \tag{2.6}$$

$$\tau_d \frac{d}{dt} e(t) \Leftrightarrow \frac{\tau_d}{t_{amostragem}} * (e[n] - e[n-1]) \tag{2.7}$$

A equação de um controlador PID digital é, portanto:

$$PID_{digital} = K_p e[n] + t_{amostragem} \frac{e[n] + e[n-1]}{\tau_i} + \frac{\tau_d}{t_{amostragem}} (e[n] - e[n-1]) \tag{2.8}$$

Os controladores PID são amplamente utilizados na indústria e diversos dispositivos já possuem funções de controle embutidas em sua programação, tornando desnecessária a implementação de ações de controle, sendo necessário apenas a sintonia dos parâmetros do controlador.

2.3.2 Dispositivos para Controle Automático

Os sistemas utilizados na automação industrial utilizam os mais diversos tipos de processadores, de microprocessadores, microcontroladores, como ARM Cortex A, i386, ou até mesmo FPGAs. A escolha do processador mais adequado depende de características como performance, segurança, e custo do projeto. O desenvolvedor de sistemas embarcados para automação deve estar atento às particularidades impostas pelo ambiente industrial e suas características, por existirem diferenças no controle de máquinas e processos. [19]

Por sua finalidades serem críticas, é desejado que estes dispositivos tenham uma programação flexível, alta confiabilidade, operem em tempo real, sejam pouco suscetíveis a ruídos e interferências, detectem falhas e operem em condições severas [19]. Os principais sistemas embarcados utilizados na automação industrial encontrados no mercado são:

- CLP (Controlador Lógico Programável);
- PAC (Controlador Programável de Automação);
- CNC (Comando Numérico Computadorizado);
- SDCD (Sistema de Controle Digital Distribuído);
- Instrumentos Inteligentes;
- Computadores *single-board*;

CLP:

Um Controlador Lógico Programável é um dispositivo que faz uso de uma memória programável para armazenar instruções que implementam funções como: lógica, sequenciamento, temporização, contagem e operações aritméticas, para controlar através de módulos de entrada e saída (digital e analógica) diversos tipos de máquinas e processos. Toda a lógica de acionamento pode ser desenvolvida através de software, que determina ao controlador a sequencia de acionamento a ser executada [19]. Um exemplo de CLP é mostrado na Figura 2.10.

Figura 2.10 – CLP Modicon M340, da Schneider Electric

Fonte: SCHNEIDER ELECTRIC. Disponível em: <http://www.schneider-electric.com/>. Acessado em 26 de novembro de 2015

A possibilidade de reprogramar a lógica do sistema caracteriza o CLP como um sistema flexível, proporcionando algumas vantagens em relação aos sistemas convencionais como menor ocupação de espaço, menor requerimento de potência, possibilidade de reutilização, programabilidade, confiabilidade, fácil manutenção, fácil interfaceamento e rapidez na elaboração de projetos pela simplicidade de sua linguagem de programação.

As lógicas implementadas recebem o sinais de diversos sensores e medidores, comparam com os valores de referência introduzidos pelo usuário e acionando motores e válvulas. Devido a sua simplicidade de programação, que, de acordo com a norma IEC 61131-3, pode ser realizada por diversos tipos de linguagens, como ladder, blocos funcionais ou SFC, e a possibilidade de aplicação em diversos processos industriais, é um dos controladores mais utilizados na indústria. [8, 13]

CNC:

Os CNCs são computadores dedicados ao controle de eixos de máquinas a fim de controlar o deslocamento de suas partes móveis e executar movimentos previamente programados. É utilizado na confecção de peças complexas, seriadas e/ou de grande precisão, especialmente quando usada em conjunto com os atuais programas CAD/CAM [19]. A Figura 2.11 mostra um exemplo de centro de usinagem, da ROMI.

Figura 2.11 – Centro de Usinagem Discovery 1250, da ROMI

Fonte: ROMI. Disponível em: <www.romi.com.br>. Acessado em 26 de novembro de 2015.

PAC:

A união de características como a programabilidade, simplicidade e robustez de um CLP com o desempenho, flexibilidade e facilidade de operação de um PC formam um PAC. Tal dispositivo de controle permite a implementação de algoritmos de controle mais complexos fazendo uso de lógica fuzzy, redes neurais e controle adaptativo, visando a máxima integração e minimizando os custos. Os PACs utilizam um sistema de protocolos de comunicação abertos e padrão, a nível de hardware e software. Protocolos como Foundation Fieldbus, Profibus, HART, seriais como RS 232C/485 e Ethernet ou mesmo protocolos standard da Internet, como TCP/IP, UDP, FTP e SMTP podem ser facilmente utilizados [19]. Como exemplo, têm-se o PAC APAX-5000, que pode ser visto na Figura 2.12.

Figura 2.12 – PAC APAX-5000, da Advantech

Fonte: ADVANTECH. Disponível em: <http://www2.advantech.com.br/apax/>. Acessado em 26 de novembro de 2015.

A junção dos instrumentos de medição e atuação com os dispositivos de controle permite a criação de sistemas de automação eficazes para diversos processos industriais, desde os mais simples até os mais complexos. Porém, mesmo com a união de toda essa tecnologia, o sistema de automação continua inseguro quando não se tem uma supervisão de todos os processos, incluindo o gerenciamento da planta, que engloba desde o gerenciamento de alarmes até o gerenciamento da produção.

2.4 Sistemas Supervisórios

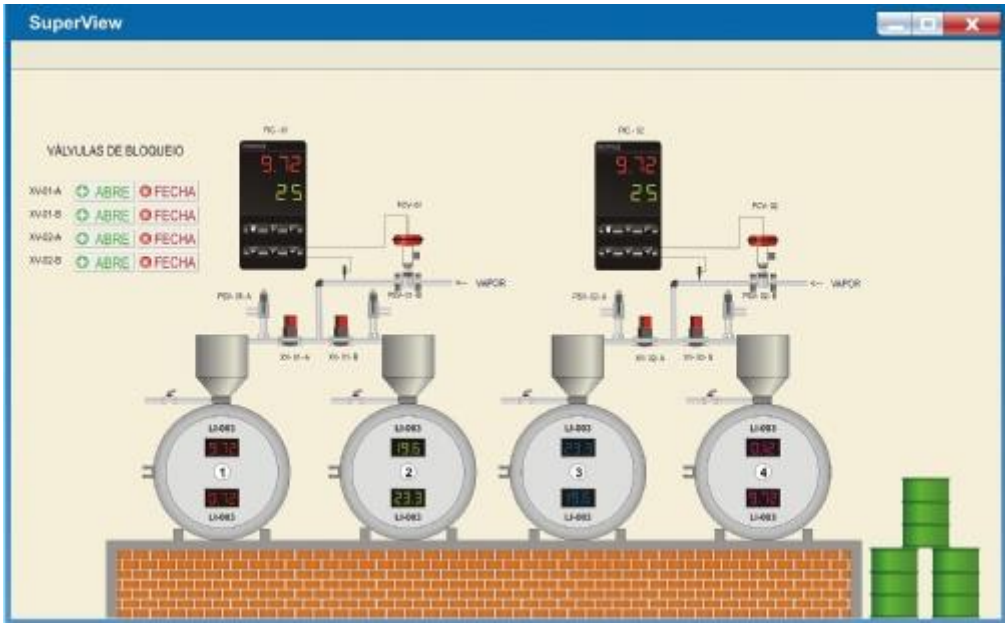
Sistemas supervisórios são responsáveis pelo monitoramento de processos industriais, através da coleta, formatação e apresentação dos dados dos dispositivos, apresentando-os ao operador em uma tela de supervisão de múltiplas formas (gráficos,

diagramas, sinalizadores, etc). Também chamados de SCADA (*Supervisory Control and data Acquisition*), tem como objetivo proporcionar ao usuário/operador uma interface de alto nível, informando-o, em tempo real, de todos os eventos relevantes da planta. Usualmente, os sistemas SCADA oferecem três funções. [20]

1. Função de supervisão: inclui todas as ações de monitoramento, como gráfico de tendências de variáveis digitais e analógicas, relatórios impressos ou em vídeo, sinótipos animados, etc.
2. Funções de operação: podem ligar e desligar equipamentos e sequência de equipamentos, operar malhas PID e mudar o modo de operação de equipamentos, direto da tela de supervisão.
3. Funções de controle: permitem definir diretamente ações de controle, sem depender de um dispositivo intermediário.
    a. Controle DDC (*Digital Direct Control*): as operações de entrada e saída são executadas diretamente dos cartões I/O ligados ao computador do SCADA. Os dados são amostrados e um algoritmo de controle é executado, e sua saída aplicada ao processo.
    b. Controle Supervisório: os algoritmos de controle são executados na UTR, mas os set-points das malhas de controle são calculados dinamicamente pelo sistema de supervisão de acordo com o comportamento do processo. Possui maior confiabilidade que o controle DDC e tem como vantagem poder atuar em múltiplas malhas simultaneamente, enquanto um operador, com um sistema convencional, só consegue trabalhar malha a malha.

As variáveis de entrada e saída são atribuídas a *tags* (que podem ser digitais ou analógicas) que são exibidas na interface de supervisão, seja em forma numérica ou elementos gráficos que são atualizados conforme os valores provindos do processo. Um exemplo de tela de supervisão é visto na Figura 2.13, do software SuperView.

Figura 2.13 – Tela típica de supervisão (Software SuperView)



Fonte: NOVUS. Disponível em: <http://www.novus.com.br/>. Acessado em 17 de novembro de 2015.

Agora com a conexão dos instrumentos de campos (sensores e atuadores), os controladores e os supervisórios, o sistema de automação torna-se eficaz no controle e monitoramento dos processos. Contudo, essa conexão de todos os sistemas, ou melhor, de todos os dispositivos, só é possível através dos sistemas de comunicação existente entre eles.

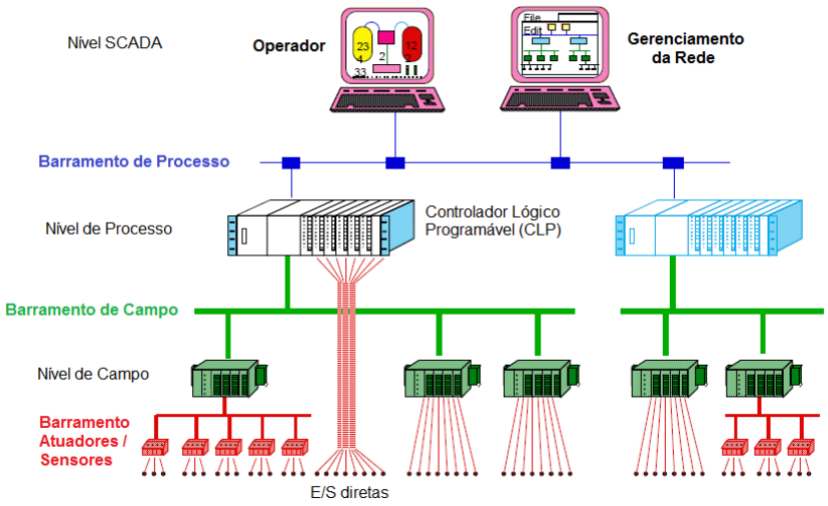
Com o avanço da tecnologia da informação, o ramo de sistemas de comunicação na automação ganhou um olhar atencioso. Os empresários, engenheiros e operadores perceberam que a falta de comunicação em qualquer ponto da pirâmide pode levar a falha na comunicão *down-top* do sistema, ou melhor, os fatores de produção, valores fundamentais para os empresários, podem não ser disponibilizados quando ocorre uma falha de comunicação no nível de dispositivos de campo.

Nesse sentido, além da utilização de protocolos de comunicação já difundidos nas redes de computadores convencionais, os sistemas de automação dependem dos protocolos de comunicação industriais que estão interligando todos os dispositivos de campo e de controle.

2.5 Protocolos de Comunicação

Redes industriais são de extrema importância em sistemas de automação, desde o nível mais baixo até o nível mais alto da pirâmide, permitindo que diversos elementos trabalhem de forma simultânea a fim de supervisionar e controlar um determinado processo. Tais elementos (sensores, atuadores, CLPs, máquinas CNC, computadores, etc.) necessitam estar interligados e trocando informações de forma rápida e precisa. Os sistemas de comunicação são constituídos por um arranjo topológico, interligando os vários módulos processadores através de enlaces físicos (meios de transmissão) e de um conjunto de regras com a finalidade de organizar a comunicação (protocolos), como mostrado na Figura 2.14. [16]

Figura 2.14 – Localização do barramento de campo na hierarquia da planta



Fonte: KONDRASOVAS, I.; Florianópolis, 2011.

O barramento de campo, visto na figura acima, consiste numa rede de dados interconectando um sistema de controle e caracteriza-se pela transmissão de inúmeros itens de dados pequenos (variáveis do processo) com um atraso limitado ($1ms \sim 1s$). Além disso, são robustos e de fácil instalação e manutenção, possuem taxa de transferência moderada ($50kbit/s \sim 5Mbits/s$) mas com uma larga faixa de distância ($10m \sim 4km$) que tem como objetivo economizar fiação mantendo o número de pontos finais. As redes de comunicação na pirâmide da automação são divididas em 3 grupos [16]:

- Redes de sensores ou Sensorbus: são redes apropriadas para interligar sensores e atuadores discretos tais como chaves limites (limit switches), contactores, desviadores, etc. São exemplos de rede Sensorbus: ASI, Seriplex, CAN e LonWorks.

- Redes de Dispositivos ou Devicebus: são redes capazes de interligar dispositivos mais genéricos como CLPs, outras remotas de aquisição de dados e controle, conversores AC/DC, relés de medição inteligentes, etc. Exemplos: Profibus-DP, DeviceNet, Interbus-S, SDS, LonWorks, CAN, ControlNet, Modbus e Foundation Fieldbus HSE.

- Redes de instrumentação ou Fieldbus: são redes concebidas para integrar instrumentos de campo no ambiente industrial, como transmissores de vazão, pressão, temperatura, válvulas de controle, etc. Exemplos: Foundation Fieldbus H1, HART, WorldFIP, Profibus-PA.

Os principais protocolos de comunicação utilizados na indústria, são [6, 16]:

- Modbus
- HART
- CAN
- Profibus
- Foundation Fieldbus
- ASI
- OPC

Dentre esses protocolos convencionais da indústria, o protocolo Modbus torna-se nótavel devido a grande quantidade de implementações em CLPs para comunicação com o nível de supervisão ou interconexões entre CLPs.

2.5.1 Modbus

O protocolo Modbus foi concebido pela empresa Modicon Industrial Automation Systems, atual Schneider, em 1979, para comunicar dispositivos mestres com seus escravos. A empresa publicou abertamente suas especificações permitindo seu

uso sem necessidade de *royalties*. Este fator, junto com a simplicidade do protocolo em si, permitiram que o Modbus se tornasse o primeiro padrão amplamente aceito em comunicações industriais. [5]

É um protocolo orientado a caractere, utilizado para conexões seriais padrão RS-232 e RS-485 bem como na camada de aplicação de redes industriais TCP/IP sobre Ethernet e MAP. Por sua simplicidade e facilidade de implementação, é um dos protocolos mais utilizados na automação industrial. [5]

O Modbus é baseado no modelo de comunicação mestre/escravo, e pode ser implementado com dois modos de transmissão: ASCII e RTU, que são selecionados durante a configuração dos parâmetros de comunicação. [4, 5]

2.5.1.1 Modo ASCII

Neste modo de transmissão, cada byte da mensagem é enviado como dois caracteres ASCII. A codificação ocorre da seguinte forma: cada byte é dividido em dois segmentos de 4 bits, onde cada segmento é codificado por sua representação hexadecimal do alfabeto ASCII. Diferentemente do modo RTU, permite transmissão assíncrona. Seu cabeçalho pode ser visto na Tabela 2.2. [4]

Tabela 2.2 – Cabeçalho do modo ASCII

Início	Endereço	Função	Dados	LRC	Fim
:	2 Chars	2 Chars	N Chars	2 Chars	CRLF

O final do frame é detectado pelos caracteres "CF" e "LF". O método de detecção de erros do modo ASCII é o LRC, cuja implementação é simples. Ocorre da seguinte forma [4]:

1. Soma dos campos:
   - $Soma = Endereço + Função + Dados;$
2. Cálculo do complemento de 2 para os 8 bits menos significativos:
   - $LRC = FFh - Soma(8\ bits) + 1;$
3. Primeiramente coloca-se o bit menos significativo na mensagem, depois o mais significativo.

O LRC gera um número hexadecimal de 2 algarismos. A probabilidade de um erro passar despercebido é de:

$$P_{falha} = \frac{1}{16 * 16} * 100 = 0.39\%$$

2.5.1.2 Modo RTU

Este modo usa uma abordagem síncrona para transmissão de dados. Cada byte é enviado usando um caractere de 11 bits: 1 start bit, usado para sincronização inicial; 8 bits de informação, codificado em binário com o bit menos significativo enviado primeiro; 1 bit de paridade, usado para detecção de erro; 1 stop bit, para garantir menor tempo ocioso entre transmissões consecutivas de caractere. O cabeçalho do moto RTU pode ser visto na Tabela 2.1. [4]

Tabela 2.1 – Cabeçalho do modo RTU

Início	**Endereço**	**Função**	**Dados**	**CRC**	**Fim**
Silêncio 3~5 chars	8 bits	8 bits	N x 8 bits	16 bits	Silêncio 3~5 chars

O CRC refere-se a detecção de erros do modo RTU. Sua implementação é mais extensa e é um método mais eficiente. Funciona da seguinte forma [4]:

1. Carrega-se o registrador CRC com um valor inicial $FFFFh$;
2. Submete-se o caractere da mensagem a uma lógica XOR com os 8 bits menos significativos do registrador CRC, retornando o resultado no mesmo.;
3. Desloca-se o conteúdo do CRC para a direita atribuído 0 a bit mais significativo.
4. Examina-se o bit menos significativo do CRC e:
    a. Se for igual a 0, repete-se o item 3;
    b. Se for igual a 1, submete-se o registrador CRC a uma lógica XOR com a constante $A001h$ retornando o resultado no mesmo, e repete-se o processo a partir do item 3.
5. Repetem-se os itens 3 e 4 até que tenham ocorrido 8 deslocamentos;
6. Repetem-se os itens 2 a 5 para os outros caracteres;
7. O valor final do CRC é o valor so campo Checksum;
8. O byte menos significativo vem antes do bit mais significativo, na mensagem.

O CRC gera um número binário de 16 bits e, dessa forma, a probabilidade de uma falha passar despercebido é de:

$$P_{erro} = \frac{1}{2^{16}} * 100 = 00015\%$$

2.5.1.3 Principais funções

O código de função varia de 1 a 255 ($0x01$ a $0xff$), mas apenas a faixa de 1 a 127 ($0x01$ a $0x7f$) é utilizada, já que o bit mais significativo é reservado para indicar respostas de execução. [4, 5]

A função *Read Coil Status*, cujo código é $0x01$, tem como finalidade a leitura das saídas discretas do dispositivo escravo, com endereços na forma $0x0nnn$, já *Read Input Status* lê as entradas discretas do mesmo, com endereços na forma $1nnn$. Seu código é $0x02$. [4, 5]

O código $0x03$ ativa a função *Read Holding Registers*, que lê os valores dos *holding registers* dos dispositivos escravos atrelados ao mestre, com endereços na forma $0x4nnn$, enquanto *Read Input Registers*, de código $0x04$, lê os valores dos *input registers* dos dispositivos mencionados. [4, 5]

Existe, também, a função *Force Single Coil*, de código $0x05$, que tem por finalidade a escrita de um valor de saída discreta de um dispositivo escravo, endereço $0x0nnn$. O valor é mantido constante enquanto uma nova operação de escrita ou a programação interna do dispositivo não o alterar. [4, 5]

Por fim, existe a função *Preset Single Register*, que é utilizada com o código $0x06$, que escreve um valor de *holding register* endereçado de $0x4nnn$. Assim como as saídas, o valor permanece constante até ser alterado pela dispositivo ou operações de escrita. [4, 5]

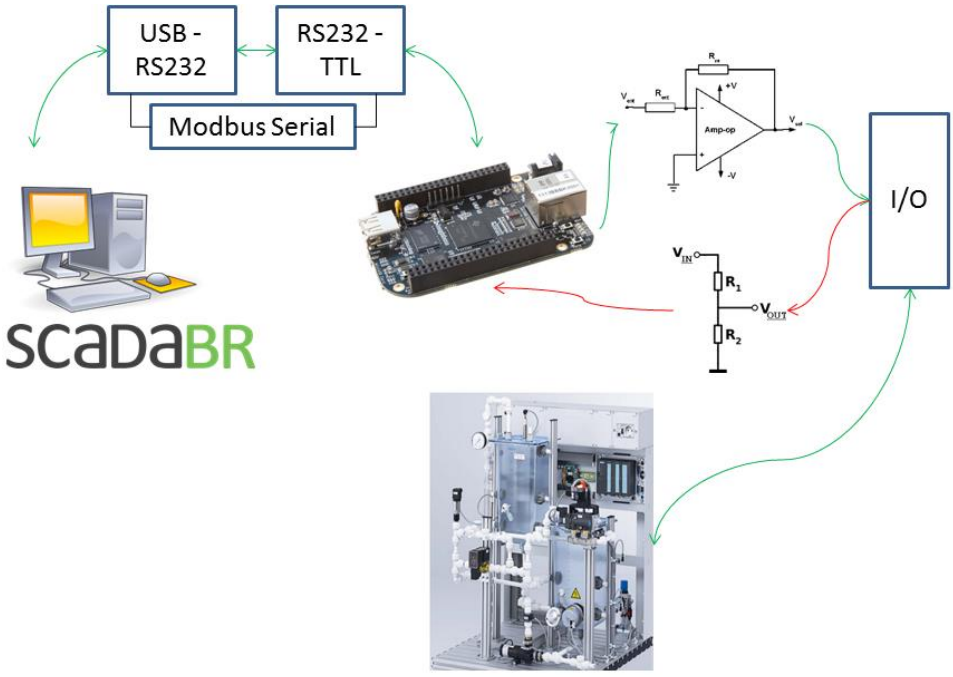
A revisão teórica apresentada neste capítulo de princípios de automação industrial, sistemas de controle, sistemas supervisórios e protocolos de comunicação servem de base para o desenvolvimento do projeto, que será apresentado no capítulo 3.

Capítulo 3

Desenvolvimento

Como explanado no Capítulo 1, este trabalho consiste no desenvolvimento de um sistema de controle, implementado na BeagleBone Black, e um sistema supervisório, no ScadaBR, realizando uma comunicação Modbus Serial com a BeagleBone, que, por sua vez, comunica-se com a planta didática MPS PA. A comunicação computador-BeagleBone é realizada por um cabo USB-RS232, que realiza a conversão de padrões USB-RS232, em conjunto com um circuito, baseado no CI MAX232, que converte RS232 em padrão TTL. O interfaceamento entre a plataforma Beagle a placa de entradas e saídas da planta envolve o projeto de divisores e amplificadores de tensão. A Figura 3.1 ilustra o projeto.

Figura 3.1 – Esquemático do projeto



Fonte: Autor, 2015.

Este capítulo irá abordar todo o procedimento necessário para realização dos objetivos deste trabalho. Nele serão introduzidos o computador *single-board*

Beaglebone Black, o software supervisório ScadaBR e a estação de processos MPS PA, para então explanar a elaboração do projeto, que consiste no o interfaceamento Beaglebone-MPSPA, na implementação do algoritmo de controle, na comunicação Beaglebone-ScadaBR através da comunicação Modbus Serial e no desenvolvimento da interface de monitoramento.

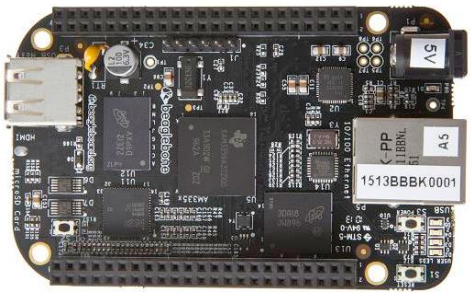
3.1 Componentes

Para realização do projeto, foram utilizados, essencialmente, a plataforma BeagleBone Black, para processamento, aquisição de dados e implementação do controlador e da comunicação Modbus, no software livre ScadaBr, para desenvolvimento do sistema supervisório, e a planta didática MPSPA, que provê o processo à ser trabalhado.

3.1.1 BeagleBone Black

A BeagleBone Black é uma plataforma de desenvolvimento de hardware livre, de baixo custo, baseada no processador Sitara XAM3359AZCZ100 Cortex A8 ARM, da Texas Instruments. [3]

Figura 3.2 – Beaglebone Black



Fonte: BEAGLEBOARD. Disponível em: <http://beagleboard.org/>. Acessado 25 de outubro de 2015.

A tabela à seguir mostra as principais características e especificações da plataforma [3]:

Tabela 3.1 – Especificações da BeagleBone Black

Processador	Sitara XAM3359AZCZ100, 1GHz, 2000MIPS

***Engine* Gráfica**	SGX530 3D, 20M Polígonos/s
Memória SDRAM	512MB, DDR3L, 800MHz
***Onboard* Flash**	4GB, MMC embarcado de 8 bits
Fontes	Mini USB, USB ou entrada DC (5V)
Ethernet	10/100 RJ45
Pinos	VCC 5V e 3.3V, 19 GPIO (entrada/saída digital, 3.3V), 7 entradas analógicas (1.8V), 5 PWM, 4 UART, 4 timers, entradas digitais configuráveis, etc
Conector SD/MMC	Micro SD, 3.3V

Fonte: BEAGLEBOARD. Disponível em: <http://beagleboard.org/>. Acessado 25 de outubro de 2015.

Diferentemente de outras placas de desenvolvimento de sistemas desenvolvimento com especificações semelhantes, a BeagleBone foi desenvolvida primariamente como uma ferramenta de prototipagem rápida e desenvolvimento de projetos eletrônicos devido ao seu hardware *open source* e grande quantidade de interfaces *I/O*. Seus 92 pinos conectores a tornam um computador adequado para coleta de dados e controle de dispositivos e equipamentos. A distribuição dos pinos pode ser vista no apêndice A. [3, 18]

A plataforma é distribuída com o sistema operacional Debian Linux e bibliotecas em C, C++ e Python, dando a BeagleBone versatilidade e performance para uma grande gama de aplicações industriais. [18]

Para este projeto, a BeagleBone Black serviu como dispositivo de controle da planta, enviando o sinal de PWM para a placa de *I/O*, adquirindo os dados e fazendo a comunicação via Modbus com o ScadaBR.

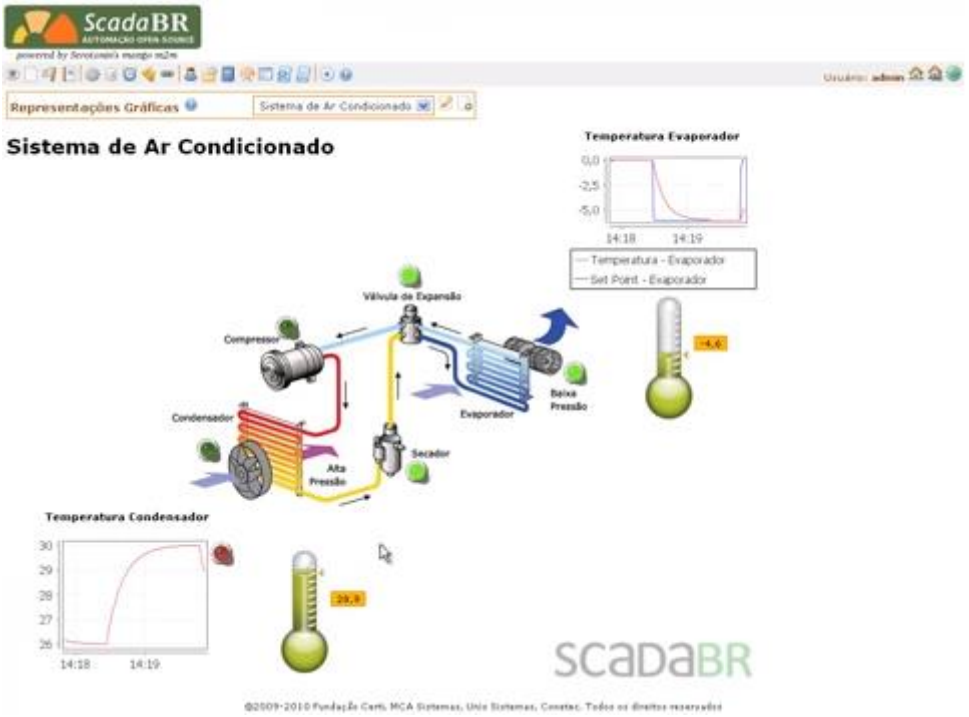
3.1.2 ScadaBR

O ScadaBR é um sistema para aquisição de dados e controle supervisório (SCADA) desenvolvido pela MCA sob a licença de software livre. Implantado em plantas industriais e prediais, o sistema possibilita uma solução aberta para desenvolvimento de sistemas supervisórios em várias áreas da indústria, como energia, saneamento, automação predial e industrial. [23]

Possui uma interface baseada em Web, para monitoramento remoto de sistemas

através de displays numéricos e gráficos, com registro histórico das variáveis medidas e geração de relatórios. Detecta ocorrência de alarmes, emitindo alertas visuais ou sonoros, podendo enviar mensagens por e-mail ou celular. Seus drivers de aquisição de dados são compatíveis com Windows e Linux, fazendo uso dos protocolos mais difundidos na indústria (Modbus, OPC, DNP3, etc.) [23]. A Figura 3.3 exibe uma tela de supervisão desenvolvida no software.

Figura 3.3 – Tela de Supervisão do ScadaBR (Sistema de Ar Condicionado)



Fonte:ScadaBR. Disponível em: <http://www.scadabr.org.br/>. Acessado em 23 de outubro de 2015.

No contexto do projeto, o software serviu para aquisição de dados da BeagleBone para mostra-los na forma de animações e gráficos.

3.1.3 Estação MPS PA

A *MPS PA Compact Workstation* é uma planta didática de processos da FESTO que tem o intuito de simular uma planta industrial real para realização de controle de diversos processos em malha fechada, requerendo compreensão de seus sensores, atuadores e placa de entradas/saídas. A planta possui 4 malhas de controle: pressão, temperatura, vazão e nível. O sistema pode ser visto na Figura 3.4. [1]

A estação possui 6 sensores (um sensor de nível ultrassônico, 2 sensores de nível capacitivos, um sensor de pressão opto-eletrônico, um sensor de vazão por turbina e um sensor de temperatura resistivo), uma bomba centrífuga, uma válvula proporcional para controle de vazão, uma válvula esférica e 8 válvulas manuais de regulagem. Sua placa de entradas e saídas recebe/envia sinais analógicos ($0{\sim}10V$) e digitais ($0{\sim}24V$). O Apêndice B deste trabalho mostra o diagrama de P&ID da estação. [1]

Figura 3.4 – Estação de Processos MPS PA

Fonte: FESTO Didactic. Disponível em: www.festo-didactic.com/. Acessado em 27 de outubro de 2015

3.2 Métodos

O projeto foi elaborado em duas etapas: comunicação BeagleBone – MPS PA, comunicação BeagleBone – ScadaBR e desenvolvimento da interface de supervisão.

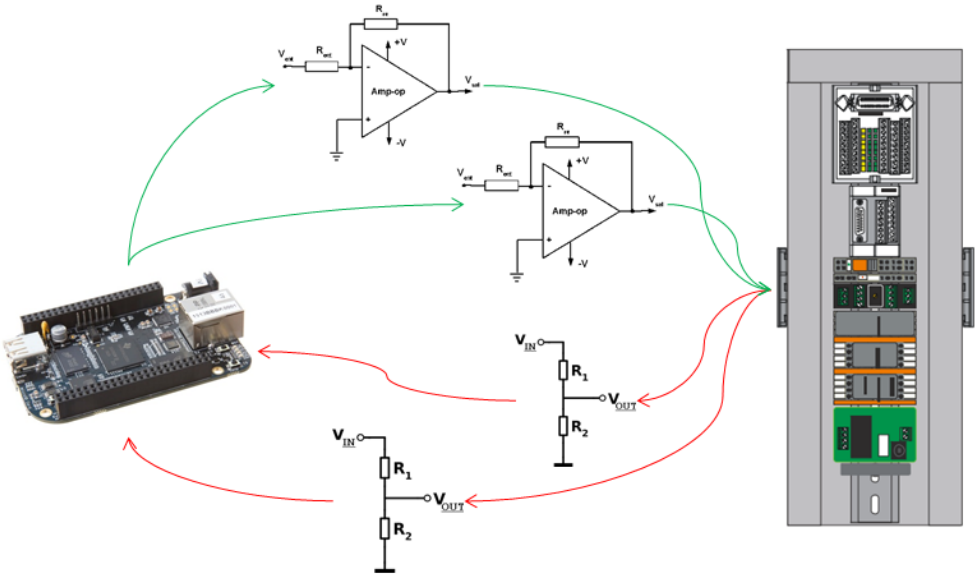
3.2.1 Comunicação BeagleBone – MPS PA

O interfaceamento da BeagleBone com a placa I/O da planta envolveu o projeto de pequenos circuitos para adequação do sinais transmitidos e recebidos. As entradas/saídas digitais plataforma BeagleBone são de 3.3V, enquanto suas entradas analógicas são de apenas 1.8V.

O sinal para controle da planta é feito pelo modo analógico, devendo ter uma tensão de 0~10V, precisando ter seu sinal amplificado em 3 vezes. A aquisição do sinal de nível, um sinal também analógico de 0~10V, deve ser feita por um pino analógico da Beagle, necessitando ser atenuado em 5.55 vezes. Para ativar o modo analógico do

motor, faz-se necessário enviar um sinal digital de 24V (bit 1) para o entrada I4 da placa I/O da planta, necessitando amplificação de 7.27 vezes do sinal da BeagleBone. Faz-se, também, a leitura de um sinal digital referente ao sensor capacitivo de nível alto/baixo, necessitando de outro divisor de tensão. Deve-se projetar, então, 2 amplificadores operacionais e dois divisores de tensão. A Figura 3.4 ilustra esse processo.

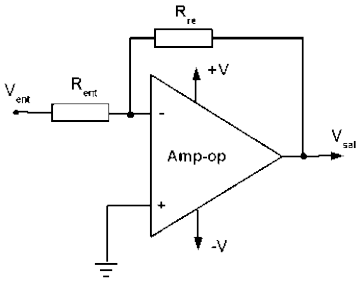
Figura 3.5 – Esquema de interfaceamento BeagleBone - Placa I/O



Fonte: Autor, 2015.

Pode-se ajustar o ganho de um amplificador operacional pela relação entre os resistores R_{ent} e R_{re}, sendo a tensão máxima de saída do amplificador limitada pela tensão de alimentação do CI. A Figura 3.5 mostra o esquema de um amplificador operacional não-inversor.

Figura 3.6 – Amplificador Operacional não-inversor



Fonte: Clube do Hardware. Disponível em: <clubedohardware.com>. Acessado em 7 de novembro de 2015

O ganho do amplificador é dado pela equação:

$$G = 1 + \frac{R_{re}}{R_{ent}}$$

Para um ganho de 3, utilizou-se uma resistência de realimentação $R_{re} = 20k\Omega$ com uma resistência de entrada $R_{ent} = 10k\Omega$:

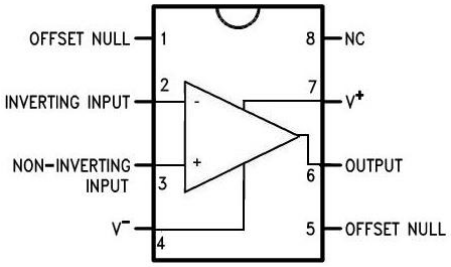
$$G_1 = 1 + \frac{20 * 10^3}{21 * 10^3} = 3$$

Para o sinal que necessita de uma tensão de 24V, projetou-se um amplificador operacional de ganho $G_2 = 11$, mas com sua tensão de saída saturada pela tensão de alimentação, de $24V$.

$$G_2 = 1 + \frac{220 * 10^3}{22 * 10^3} = 11$$

O CI utilizado foi o LM741, cujo esquema pode ser visto na Figura 3.7, à seguir.

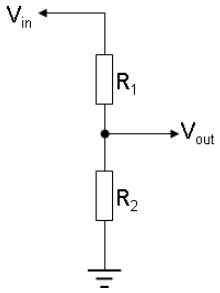
Figura 3.7 – Circuito Integrado LM741



Fonte: ALL DATA SHEET. Disponível em: <www.alldatasheet.com>. Acessado em 29 de outubro de 2015.

Os divisores de tensão foram projetados conforme o arranjo da Figura 3.8, abaixo:

Figura 3.8 – Divisor de tensão



Fonte: MECATRÔNICA SIMPLES. Disponível em: <mecatrônica-simples.blogspot.com>. Acessado em 7 de novembro de 2015.

Onde:

$$V_{out} = \frac{R_2}{R_1 + R_2} V_{in}$$

Dessa forma, para atenuar uma tensão de $10V$ para $1.8V$(entrada analógica da Beagle), foi utilizado $R_2 = 1.8k\Omega$ e $R_1 = 8.2k\Omega$. Têm-se:

$$V_{out1} = \frac{1.8}{8.2 + 1.8} V_{in1} => V_{out1} = 0.18 * 10 = 1.8V$$

Analogamente, para atenuar uma tensão de $24V$ para $3.3V$(entrada digital da Beagle), utilizou-se $R_2 = 2.7k\Omega$ e $R_1 = 18k\Omega$.

$$V_{out2} = \frac{2.7 * 10^3}{20.7 * 10^3} V_{in2} => V_{out2} = 0.13 * 24 = 3.13V$$

O sinal amplificado para 24V é enviado para a entrada digital *XMAX1-02*, ativando o modo analógico. O sinal de PWM é enviado para a entrada analógica *X2-8*, enquanto o sinal do sensor de nível, atenuado de 10 para 1.8 V, é obtivo da saída analógica *X2-1*. O sinal digital do sensor de nível alto é obtido da saída *I4*. O terra da planta é compartilhado tanto para as saídas analógicas quanto digitais.

3.2.1.1 Implementação do Controlador

O controlador PID foi implementado na BeagleBone em Python. Basicamente, foi efetuado em três etapas:

- Inicialização das entradas/saídas digitais e analógicas;
- Conversão AD do sinal de nível e ativação do modo analógico;
- Implementação do Controlador PID e envio do sinal via PWM.

O código em Python utilizou apenas bibliotecas nativas do Python: *time*, *Adafruit_BBIO.GPIO*, *Adafruit_BBIO.ADC* e *Adafruit_BBIO.PWM*. As entradas e saídas digitais devem ser configuradas (no caso, selecionadas como entrada ou saída) enquanto que a utilização das entradas analógicas depende da inicialização do modo de

conversão A/D para o pino selecionado. Deve-se, também, iniciar o pino de PWM (que também é uma saída digital).

Para inicializar as entradas/saídas digitais, faz-se:

GPIO.setup("pino", GPIO.IN) ou *GPIO.setup("pino", GPIO.OUT)*

O pino de PWM é iniciado como visto à seguir:
PWM.start("pino", dutycycle)

A função já deve ser inicializada com um valor de *duty cycle*, preferencialmente igual a zero.

O modo ADC é inicializado da seguinte forma:

ADC.setup()
ADC.read("pino")

A função *ADC.read()* lê o sinal analógico e o converte em um valor entre 0 e 1, fazendo-se necessário que se multiplique esse valor de modo a representar o nível do tanque. Considerando que o tanque da planta tem 35 cm de altura, multiplica-se o sinal lido por esse valor.

O controlador foi implementado com base no diagrama visto na Figura 2.9 e na equação (2.8). O pseudo-código encontra-se abaixo:

Algoritmo 3.1 – Implementação do Controlador PID

ENTRADA: ***dt*** (tempo de amostragem), ***kp*** (constante proporcional), ***ti*** (constante integrativa), ***td*** (constante derivativa), ***SP*** (referência)

Nível = Leitura(pino analógico)*35

Erro = ***SP*** – ***Nível***

Erro_anterior = 0

Integral = *0*

ENQUANTO abs(***Erro***) $\geq$ 0,05 **FAÇA**

    Integral = Integral + Erro*dt

    Derivativo = (Erro – Erro_anterior)/dt

    PID = ***kp*Erro + (Integral/ti) + (Derivativo*td)***

    Erro_anterior = Erro

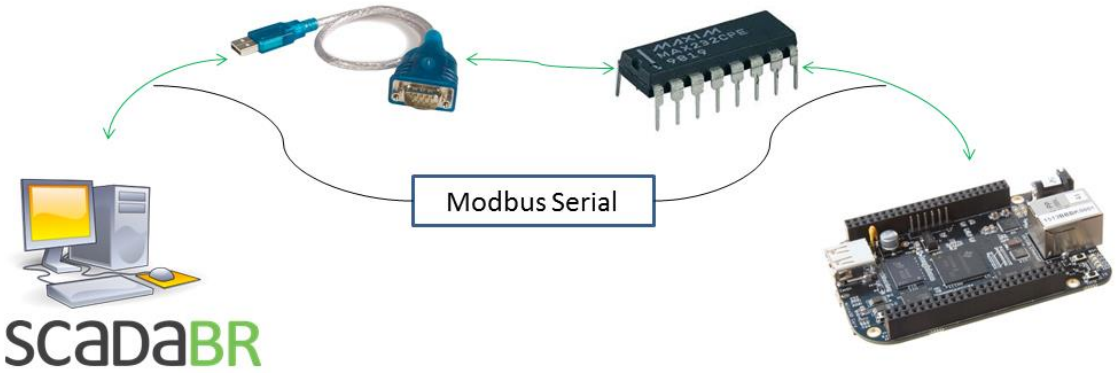
SE *PID* > 100
***PID* = 100**
FIM SE
SE *PID* < 0
***PID* = 0**
FIM SE
Iniciar.PWM(“pino”, ***PID***)
TempoRepetição(***dt***)
FIM ENQUANTO

Os valores adequados para Kp, ti e td foram obtidos experimentalmente e serão mostrados no capítulo seguinte, relacionado aos resultados obtidos. As condições do *if* servem para limitar o valor do controlador entre 0 e 100, para se adequar aos valores possíveis da função *PWM.set_duty_cycle(“pino”, dutycycle)*.

3.2.2 Comunicação BeagleBone – ScadaBR

A comunicação entre a BeagleBone e o software ScadaBR se fez via Modbus Serial. Dessa forma, utilizou-se um cabo USB-RS232 para o interfaceamento com o computador. Porém, como estamos trabalhando com sistemas digitais embarcados, é necessário converter o sinal para um sinal digital adequado à plataforma. No caso da BeagleBone, as portas digitais são baseadas no padrão TTL, assim sendo necessário utilizar um circuito conversor de lógica RS232-TTL. A Figura 3.9 provê uma visão geral do processo.

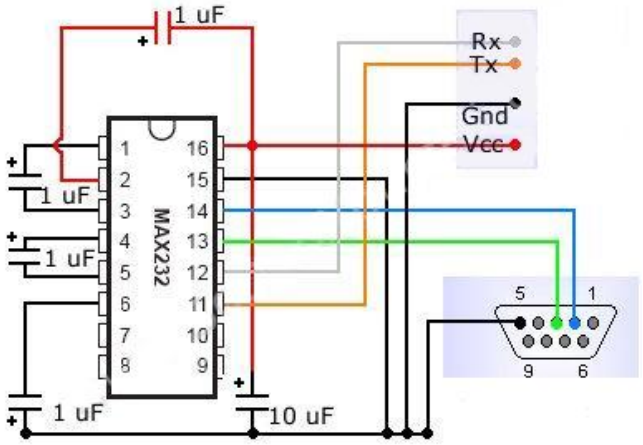
Figura 3.9 – Comunicação BeagleBone – ScadaBR



Fonte: Autor, 2015.

O padrão RS232 implica em um sinal baixo de -3 a -15 volts, e um sinal alto de 3 a 15 volts. Um sinal no padrão USB, nesse caso, convertido pra RS232, fornece um sinal baixo/alto de $-12/12$ volts. Já o padrão TTL, presente na maioria dos sistemas digitais, tem uma lógica de sinal baixo/alto de $0{\sim}0.8/2{\sim}5$ volts. A conversão foi feita utilizando um CI MAX232 e um pequeno circuito associado, composto de cinco capacitores eletrolíticos. O esquema do circuito pode ser visto na Figura 3.10, à seguir.

Figura 3.10 – Esquema do Circuito de conversão RS232-TTL



Fonte: NBGlim. Disponível em: <http://www.nbglin.com/rs232.htm>. Acessado em 14 de novembro de 2015.

3.2.2.1 Modbus Serial e Configuração dos *Data Points*

Foi utilizado, para a implementação da comunicação, o Modbus Serial tipo ASCII, cujo cabeçalho pode ser visto na Tabela 2.2. O ScadaBR envia a mensagem em hexadecimal, confirme a tabela ASC, enviando os caracteres CRLF após pular uma linha, em ASC. O ScadaBR atua como mestre, enquanto a BeagleBone como escravo.

Deve-se, inicialmente, criar um *data source*, que é atrelado a fonte de aquisição de dados, no caso, a BeagleBone Black. Dentro do *data source*, criam-se os *data points*, que se referem ao tipo de informação recebida/enviada e seu endereço no registrador.

Ao se criar e habitar um *data point* no ScadaBR, o programa envia um *request* para o registrador especificado, esperando um retorno do tipo de informação dada ao *data point*. Dessa forma, quando se cria um *data point* do tipo *holding register* (função 03) configurado como inteiro de dois bytes sem sinal, e especifica-se o endereço do escravo como 01, o programa envia a seguinte requisição:

:	01	03	00000001	LRC	“CR”+”LF”

E espera uma resposta no seguinte formato:

:	01	03	02 + 2 bytes de informação	LRC	“CR”+”LF”

O código *02* no início da informação da mensagem indica o tamanho total da mensagem, em *bytes*. Se, por exemplo, um *float* fosse enviado, ao invés de um *inteiro*, esse valor seria *04*. Caso a requisição fosse de um valor binário, a resposta também deverá incluir *02* no espaço dos dados, pois o menos tamanho para o cabeçalho é *2 bytes*. Nesse caso, espaço da função seria *01*.

Um mesmo escravo não pode ter *data points* de tipos de dados iguais, de forma a não haver choque de dados em uma mesma variável. Os valores lidos no programa e enviados para o ScadaBR são: Set-Point (inteiro, fornecido pelo usuário), nível (*float*, leitura em tempo real do tanque) e um sinal binário enviado pelo sensor de nível alto (lido em tempo real). São necessários, então, 2 escravos para 3 *data points*: um escravo para receber a informação binária do sensor alto e a informação *float* do nível, e um escravo para receber o dado tipo inteiro da referência. Para que a comunicação ocorra de forma síncrona e sem perda de pacotes, o tempo de amostragem do código na BeagleBone deve ser o mesmo que configurado no *data point* (período de atualização).

O código, feito em Python, consiste basicamente na conversão dos sinais recebidos da planta pela BeagleBone no formato ASCII e envio do cabeçalho Modbus para a porta serial em resposta as requisições lidas do ScadaBR, com o tempo da comunicação de acordo com as configurações dos parâmetros do *data source*. Fez-se o uso da biblioteca *serial*, que já está inclusiva no pacote Python. Para utilizar os pinos UART da Beagle, usados para comunicação serial, deve-se habilita-los pelo seguinte comando, no prompt do Linux/debian:

echo BB-UART1 > /sys/devices/bone_capemgr./slots*

Para iniciar a comunicação serial, é necessário “abrir” a porta através da seguinte linha de código:

ser.open()

E ao final da execução, deve-se fechá-la:

ser.close()

Os comandos para escrita e leitura na porta serial são, respectivamente:

ser.read()
ser.write()

A função de aquisição e conversão de dados é executada dentro de um *loop* de leitura da porta serial, enquanto a leitura não for nula e a partir do momento que a porta é habilitada.

Basicamente, lê-se a porta serial e com base em trechos analisados da requisição, cria-se um *switch-case* para envio das respostas esperadas para cada tipo de requisição. O pseudo-código o processo de leitura e envio dos dados via Modbus Serial:

Algoritmo 3.2 – Comunicação Modbus Serial Beagle - ScadaBR

```
FUNÇÃO leitura:
    Resposta = " "
    Escravo = leitura[1]+leitura[2]
    Função = leitura[3]+leitura[4]
    CRLF = leitura[15]+leitura[16]
    ESCOLHA Escravo
    CASO '01':
        SE Função = 03
            Resposta =':'+Escravo+'030200'
            Nível = Leitura.pino("pinonível")
            Resposta += str(hex(int(Nível))*350
            Resposta += LRC(Resposta)+CRLF
        SENÃO SE Função = 01
            Resposta =':'+Escravo+'010200'
            Sensor = Leitura.pino("pinosensor")
            SE Sensor = 0
```

```
                    Resposta += '0000'
                SENÃO
                    Resposta += 'FF00'
                FIM SE
                Resposta += LRC(Resposta)+CRLF
            FIM SE
        CASO '02':
            SE Função = 03
                Resposta =':'+Escravo+'030200'
                Resposta += str(hex((SP))
                Resposta += LRC(Resposta)+CRLF
            FIM SE
        FIM ESCOLHA
RETORNE Resposta
```

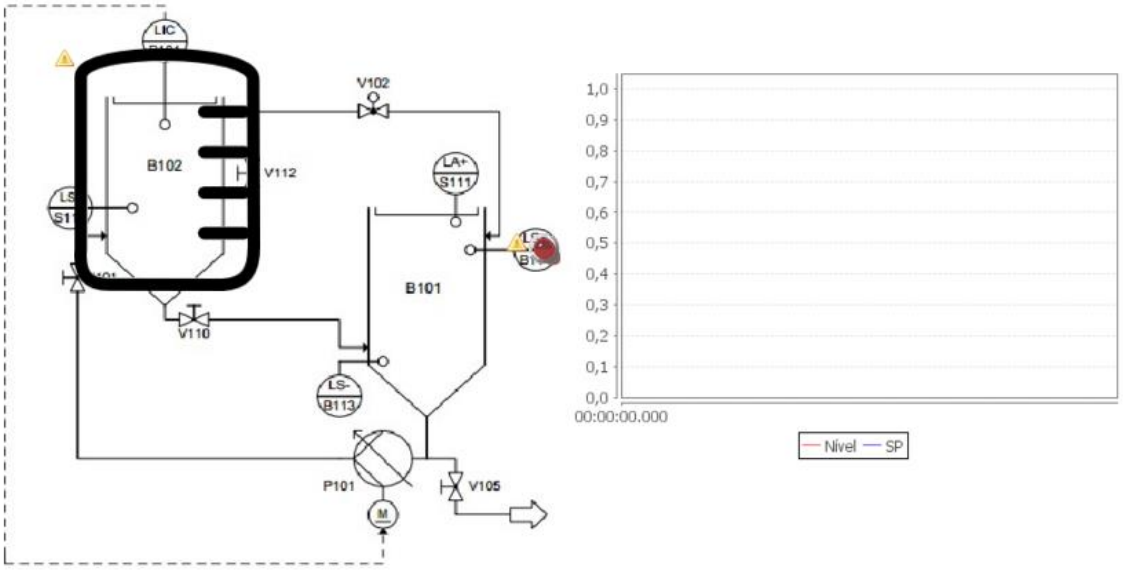
Após a integração das duas etapas do projeto (comunicação BeagleBone – MPS PA e comunicação BeagleBone – ScadaBR), pode-se colocar o sistema em execução para analisar os resultados, que são mostrados no capítulo 4.

Capítulo 4

Resultados

A interface gráfica construída no ScadaBR pode ser vista na Figura 4.1, onde a figura do sistema é o diagrama de tubulação e instrumentação (P&ID) do sistema de nível da planta. O tanque está atrelado ao *data point* do nível do processo, e o gráfico recebe o nível e o set-point. O LED indicativo próximo ao tanque de armazenamento recebe um valor binário do sensor de nível alto da planta.

Figura 4.1 – Interface desenvolvida no ScadaBR



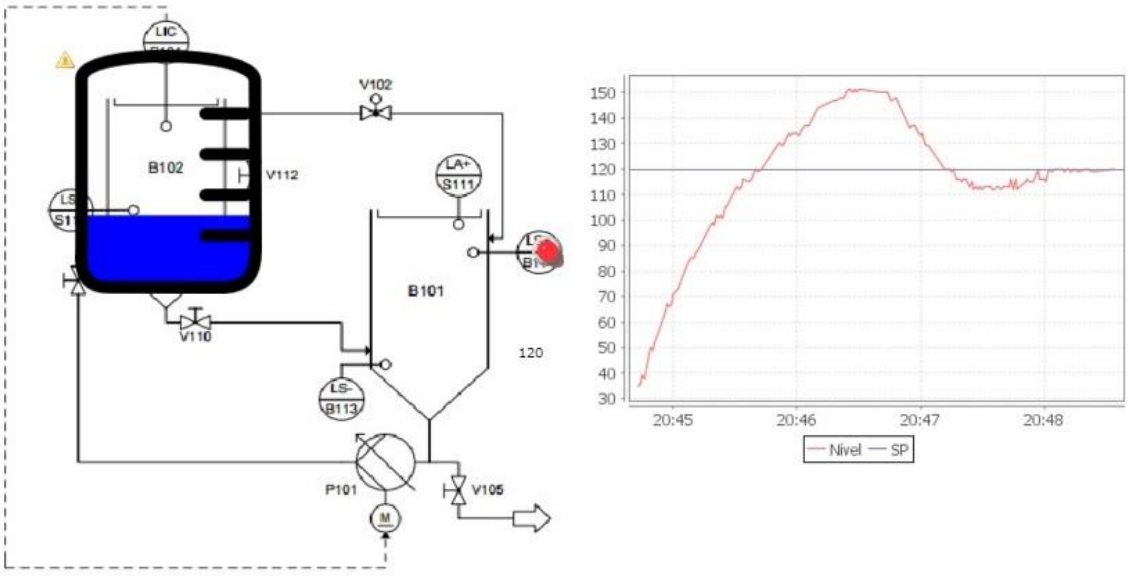
Fonte: Autor, 2015.

Com todos os *data points* devidamente configurados e habilitados, inicia-se a aquisição de dados e controle do processo pela BeagleBone, enquanto em paralelo as informações são formatadas no padrão Modbus e enviadas pelas saídas UART da BeagleBone, para realizar a comunicação via serial com o supervisório. Para todos os testes mostrados neste capítulo, utilizou-se como parâmetros do controlador $K_p = 10, \tau_i = 1$ e $\tau_d = 0.2$, obtidos experimentalmente, e um tempo de amostragem $t = 100ms$. Deve-se salientar que o nível mínimo do tanque é de 3,5 cm, devido ao nível de água no tanque de armazenamento, e que os valores mostrados no gráfico se encontram em milímetros.

4.1 Desempenho do Sistema

Inicialmente, aplicou-se ao sistema um *set-point* de 12 cm, e a resposta dinâmica foi mostrada na tela de supervisão conforme pode ser visto na Figura 4.2.

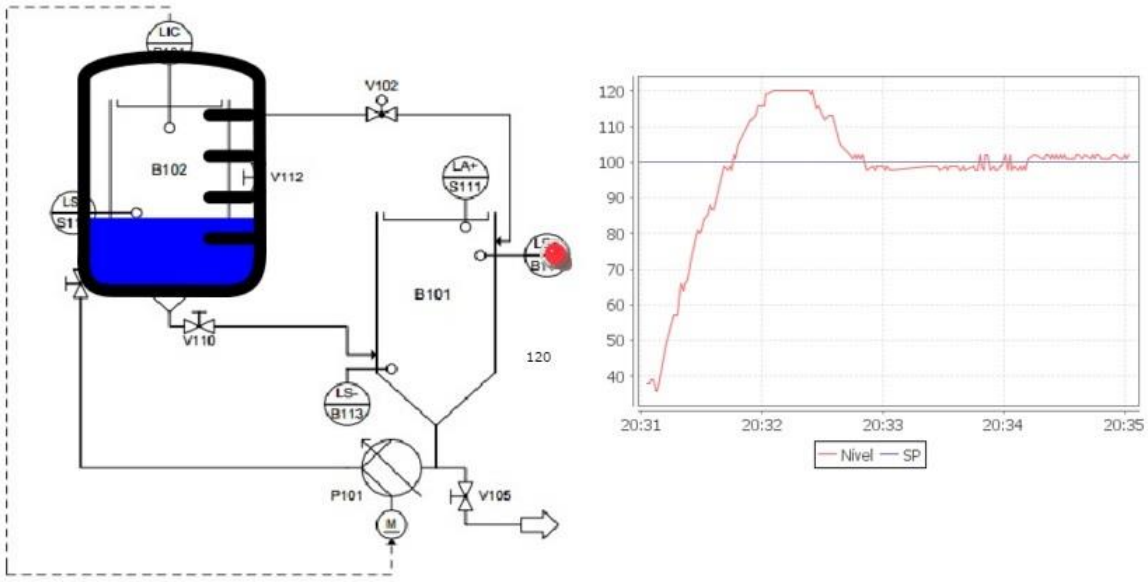
Figura 4.2 – Tela de supervisão para uma referência de 12 cm



Fonte: Autor, 2015.

O resposta do sistema teve um *overshoot* de 25,8% e um tempo de acomodação de 198s. Pode-se observar o nível do tanque no supervisório, em azul, correspondente ao nível do tanque na planta. O segundo teste foi realizado diminuindo-se o set-point para 10 cm, e a Figura 4.3 mostra a tela de supervisão para essa situação.

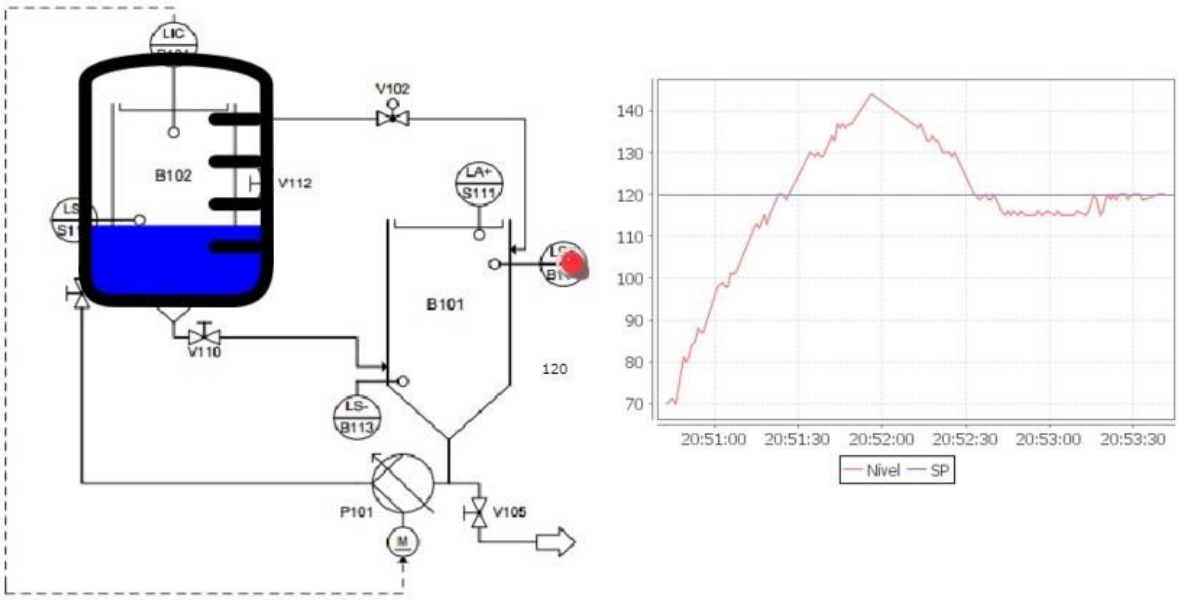
Figura 4.3 – Tela de Supervisão para uma referência de 10 cm



Fonte: Autor, 2015.

Para esta referência, o sistema teve um *overshoot* de 20% e tempo de estabilização de $179s$. O terceiro teste enviou uma referência de 12 cm, mas com o tanque já preenchido em 7 cm, e a resposta do sistema pode ser vista na tela de supervisão mostrada na Figura 4.4. Observa-se, pelo gráfico, que o sistema teve um overshoot de 19.8% e um tempo de resposta de $163s$.

Figura 4.4 – Tela de supervisão para uma referência de 12 cm com tanque preenchido em 7 cm



Fonte: Autor, 2015.

Para todos os teste de controle, pôde-se observar ruído na resposta dinâmica, este atrelado a fatores externos como vibração do motor e da água e interferência magnética nos fios do interfaceamento (amplificados pelos amplificadores operacionais).

Nota-se, também, que a luz indicativa do sensor de nível alto permaneceu na cor vermelha em todas as simulações demonstradas. Isto se deve ao fato de o tanque de armazenamento não se encontrar mais em nível alto devido a quantidade de água abastecida ao tanque controlado e, portando, o sensor envia zero para a BeagleBone.

O tempo de resposta é consideravelmente lento devido ao motor dos tanques não ser adequado para o tamanho dos mesmos, independe da corrente enviada pelo sistema embarcado.

Em alguns pontos dos gráficos, pode-se notar uma disparidade no valor do nível (na Figura 4.4, no pico do *overshoot*, por exemplo). Estes pequenos saltos no valor lido ocorrem devido a perda de pacotes.

4.2 Perda de Pacotes

Durante o procedimento de envio dos dados via serial no formato Modbus para o ScadaBR, ocorreram alertas de perda de pacote (acompanhadas de um alarme sonoro). Como pode ser visto na Figura 4.5, ocorrem 7 mensagens de erro em um minuto.

Figura 4.5 – Alertas de perda de pacote

Id	Nível de alarme	Tempo	Mensagens
37550		Dez 03 20:16	'Beagle': Exceção do modbus master: LRC mismatch: given=155, calc=170
37547		Dez 03 20:16	'Beagle': Exceção do modbus master: LRC mismatch: given=159, calc=170
37544		Dez 03 20:16	'Beagle': Exceção do modbus master: LRC mismatch: given=159, calc=170
37481		Dez 03 20:16	'Beagle': Exceção do modbus master: LRC mismatch: given=142, calc=154
37448		Dez 03 20:16	'Beagle': Exceção do modbus master: LRC mismatch: given=128, calc=138
37451		Dez 03 20:16	'Beagle': Exceção do modbus master: LRC mismatch: given=127, calc=138
37448		Dez 03 20:16	'Beagle': Exceção do modbus master: LRC mismatch: given=128, calc=138
37391		Dez 03 20:15	'Beagle': Exceção do modbus master: LRC mismatch: given=128, calc=138

Fonte: Autor, 2015.

Considerando que o tempo amostragem do algoritmo de envio e recebimento de $t = 100ms$, são transmitidos 600 pacotes por minuto. Dessa forma, a probabilidade de se ter uma perda de pacote é:

$$P_{erro} = \frac{7}{600} = 0.0116\%$$

Os erros de perda de pacote, ou *LRC mismatch*, são geralmente atrelados a sincronização da transferência/recebimento dos dados relacionados a configuração do tempo de atualização e *timeout* dos *data points* Modbus. A porcentagem de perdas foi muito pequena e não afeta o correto funcionamento da supervisão.

Capítulo 5

Considerações Finais

O desenvolvimento do sistema de automação para o processo desejado mostrou-se bastante proveitoso por requerer a integração de vários conhecimentos da engenharia, como sistemas de controle, sistemas supervisórios, protocolos de comunicação bem como conhecimentos técnicos da planta à ser trabalhada, do sistema embarcado utilizado para aquisição e processamento dos dados, e do software utilizado para o monitoramento.

Os parâmetros do controlador obtidos experimentalmente mostraram-se adequados para o processo, realizando um controle satisfatório do nível do tanque com *overshoot* entre 20~25%. O tempo de resposta, substancialmente lento, esta mais atrelado ao fato do motor ser inapropriado para o sistema de tanques.

A BeagleBone Black demonstrou ser uma ferramenta eficiente para o controle de nível do sistema, com uma grande quantidade de entradas e saídas digitais configuráveis, saídas disponíveis para PWM, bibliotecas já inclusas no pacote Python para conversão AD, comunicação serial e ativação de PWM. A linguagem Python, porém, revelou-se pouco conveniente para programação paralela.

Com software baseado em web ScadaBR foi possível criar uma interface para representação dos dados, adquiridos via Modbus, condizente e eficiente para o monitoramento do processo. A quantidade de erros foi pouco significante e o monitoramento funcionou de forma válida. Deve-se atentar ao período de atualização configurado no *data source*, devendo este ser menor ou igual ao tempo de repetição do algoritmo de comunicação implementado na BeagleBone, para evitar uma perda de pacotes acima do desejado. O envio de dados pelo interface, porém, demonstrou ser pouco intuitivo, havendo a necessidade de se criar um código em Javascript para desempenhar a função de envio de um valor inteiro/*float*.

O desenvolvimento e documentação deste projeto serão utéis para trabalhos futuros, tanto para a comunicação da BeagleBone com softwares SCADA, via serial,

quanto para o controle de outras malhas da estação MPS PA. Como sugestões de trabalhos futuros, pode-se:

- Controlar as malhas de vazão, pressão e temperatura da planta, fazendo um estudo da placa de entradas e saídas;
- Propor um controlador mais eficiente, de forma a reduzir o *overshoot* e tempo de acomodação do sistema;
- Utilizar outra linguagem de programação mais efetiva para lidar com programação paralela, ou pesquisar mais à fundo e realizar o paralelismo eficientemente com a própria linguagem Python;
- Solucionar o problema de envio de informação pela interface do ScadaBR, pela configuração de um data point por código em Javascript.

Por fim, este trabalho foi produtivo por requerer o interfaceamento de ferramentas utilizadas na indústria, o controle de um dos processos mais comumente tratado em fábricas e ambientes de manufatura, e a implementação de um dos protocolos mais utilizados para transmissão de dados entre equipamentos, dispositivos de controle e sistemas de monitoramento, em ambientes industriais.

Referências Bibliográficas

[1] **FESTO.** *MPS PA Datasheets*. Esslingen, 2006. 194 p.

[2] **MCA SISTEMAS.** *ScadaBR: Manual do Software*. Florianópolis, 2010. 70 p.

[3] **COLEY, G.** *BeagleBone Black System Reference Manual*. Dallas: Adafruit, 2005. 108 p.

[4] **MODICON, INC**. *Modicon Modbus Reference Guide*. Massachusetts, 1996. 121 p.

[5] **WILAMOWSKI, B. M.; IRWIN, J. D.** *Industrial Communications Systems*. 2ª Edição. California: CRC Press, 2011. 909 p.

[6] **CASTRUCCI, P.; MORAES, C. C.** *Engenharia de Automação Industrial*. 2ª Edição. São Paulo: LTC, 2007. 358 p.

[7] **LUTZ, M.** *Learning Python*. 5ª Edição. Massachusetts: O' Reilly Media, 2013. 1600 p.

[8] **NETO, J. T. C.** *Controladores Lógicos Programáveis*. Natal, 2011. Apostila. 84 p.

[9] **OGATA, K.** *Engenharia de Controle Moderno*. 5ª Edição. São Paulo: Pearson, 2011. 809 p.

[10] **VALENTI, C.** *Implementing a PID Controller Using a PIC18 MCU*. Microchip Technology Inc, Application Note DS00937A, 2004.

[11] **MENEGHETTI, F**. *Sistemas de Controle*. Natal, 2007. Apostila. 101 p.

[12] **THOMAZINI, D.; ALBUQUERQUE, P. U. B.** *Sensores Industriais: Fundamentos e Aplicações*. 5ª Edição. São Paulo: Editora Érica, 2012. 244 p.

[13] **COELHO, M. S.** *Introdução a Automação Industrial*. São Paulo, 2007. Apostila. 173 p.

[14] **CULLEN, G.** *Control Engineering 3: Module 1*. Glasgow, 2014. Notas de Aula. 192 p.

[15] **GUEDES, L. A.** *Redes para Automação Industrial*. Natal, 2015. Notas de Aula. 205 slides.

[16] **KONDRASOVAS, I.** *Sistemas Supervisórios e Protocolos de Comunicação Industrial*. Florianópolis, 2010. Notas de Aula. 37 slides.

[17] **AFONSO, A. P.**; *Introdução aos Sistemas de Controle*. Disponível em: < matematiques.com.br/arquivos>. Acessado em 4 de dezembro de 2015.

[18] **LOGIC SUPPLY**. *BeagleBone – Getting Ready for Industrial Applications*. Disponível em: <http://inspire.logicsupply.com/2014/07/beaglebone-getting-ready-for-industrial.html>. Acessado em 8 de novembro de 2015.

[19] **FREITAS, C. M.**; *Sistemas Embarcados na Automação Industrial*. Disponível em: <http://www.embarcados.com.br/sistemas-embarcados-na-automacao-industrial/>. Acessado em 6 de novembro de 2015.

[20] **FILHO, C. S.**; *Capítulo 3: SCADA*. Belo Horizonte, 2006. Notas de Aula. 48 p.

[21] **DEVLINUXBR**. *Vamos falar de Python? – Theading*. Disponível em: <http://devlinuxbr.blogspot.com.br/2015/03/vamos-falar-de-python-threading.html>. Acessado em 13 de novembro de 2015.

[22] **PRADO, S**. *Primeiras Impressões da BeagleBone Black*. Disponível em: <http://sergioprado.org/primeiras-impressoes-da-beaglebone-black/>. Acessado em 21 de novembro de 2015.

[23] **MCA SISTEMAS**. Disponível em: <http://www.mcasistemas.com.br/>. Acessado em 13 de novembro de 2015.

[24] **ALL DATA SHEET**. Disponível em: <www.alldatasheet.com>. Acessado em 29 de outubro de 2015.

[25] **MECATRÔNICA ATUAL**. Disponível em: <http://www.mecatronicaatual.com.br/>. Acessado em 3 de novembro de 2015.

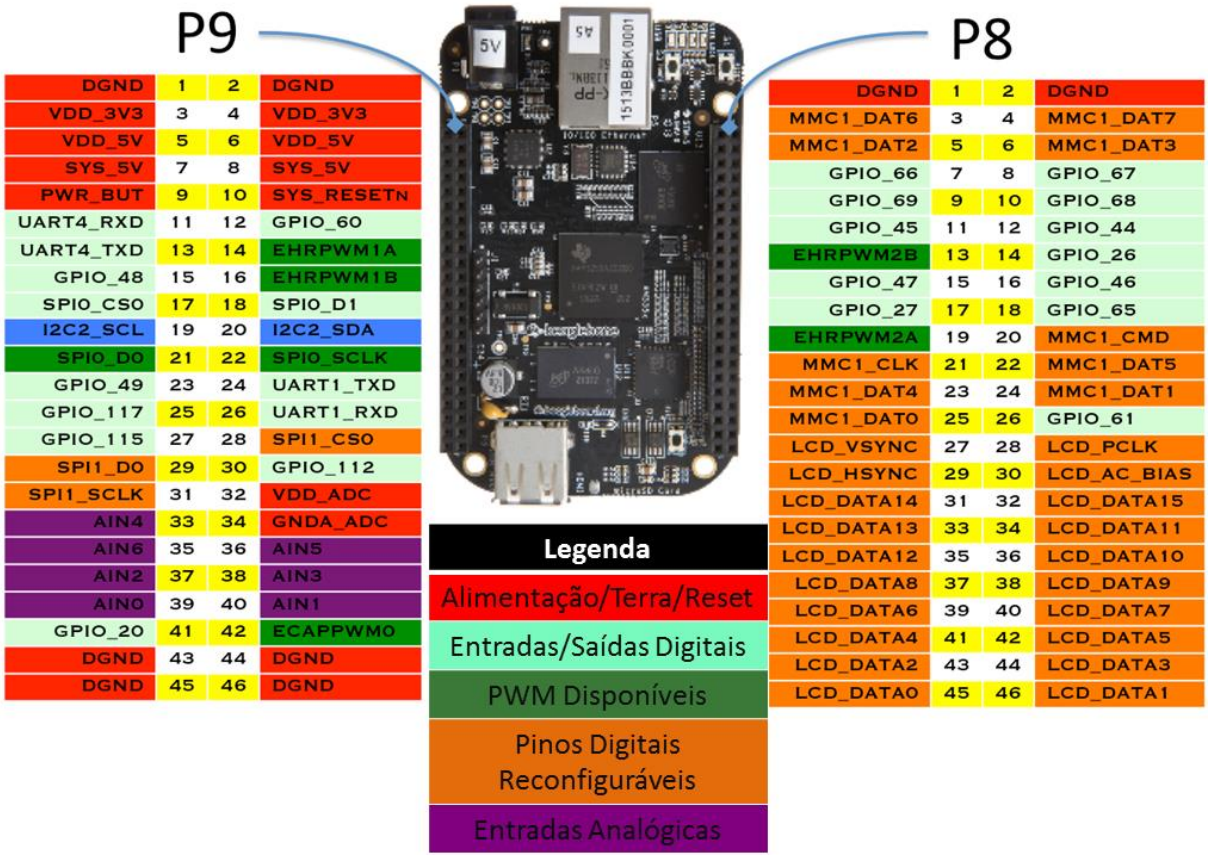
[26] **SCADABR, Forum**. Disponível em: <http://www.scadabr.com.br/?q=forum>. Acessado em 4 de dezembro de 2015.

[27] **CLUBE DO HARDWARE, Forum**. Disponível em: <http://forum.clubedohardware.com.br/>. Acessado em 25 de novembro de 2015.

[28] **MECATRÔNICA SIMPLES**. Disponível em: <mecatrônica-simples.blogspot.com>. Acessado em 7 de novembro de 2015.

Apêndice A

Detalhamento dos Pinos da BeagleBone Black



Apêndice B

Diagrama P&ID da MPS PA