DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Automação de uma Estação de Tratamento de Esgoto através de uma rede Zigbee.

Relatório submetido à Universidade Federal de Santa Catarina

como requisito para a aprovação na disciplina

DAS 5511: Projeto de Fim de Curso

Delano Romay Fiedler

Florianópolis, Julho de 2013.

Automação de uma Estação de Tratamento de Esgoto através de uma rede Zigbee.

Delano Romay Fiedler

Esta monografia foi julgada no contexto da disciplina
DAS5511: Projeto de Fim de Curso
e aprovada na sua forma final pelo
Curso de Engenharia de Controle e Automação

Prof. *Marcelo Ricardo Stemmer*

_____
Assinatura do Orientador

Banca Examinadora:

Ezequiel Medeiros
Orientador na Empresa

Prof. Marcelo Ricardo Stemmer
Orientador no Curso

Prof. Henrique Simas
Avaliador

Rafael Fazolin Wendling
Augusto Schneider Westphal
Debatedores

Agradecimentos

Agradeço primeiramente aos meus mestres, que durante meu caminho na Universidade propiciaram o ambiente e as condições necessárias para meu desenvolvimento intelectual de Engenharia de Controle e Automação.

Agradeço também aos meus colegas de empresa, em especial Edgar, Diego, Marcelo, Alcino, Ezequiel, Camila, Felipe e Clésio pela ajuda na elaboração deste trabalho e pela amizade.

Agradeço a CASAN, empresa que me acolheu desde a época de estagiário e me proveu de todas as ferramentas necessárias para meu desenvolvimento e elaboração do meu projeto.

Finalmente agradeço aos meus pais, que sempre foram o pilar de sustentação da minha vida e o apoio deles que me fez seguir em frente durante muitos momentos difíceis.

Resumo.

A CASAN, Companhia Catarinense de Águas e Saneamento, com matriz em Florianópolis, é uma empresa de capital misto, criada em 1970 cujo objetivo é fornecer água tratada, além de coletar e tratar esgoto sanitário. Presente em mais de 200 municípios catarinenses, atende atualmente uma população de mais de 2,5 milhões de habitantes.

Entre os processos desenvolvidos na empresa, podem-se citar: Captação de água bruta, tratamento de água bruta, transporte e reservação de água tratada, coleta, transporte e tratamento de esgoto sanitário.

Estes processos citados estão sempre sujeitos a estudos de melhorias, aumento de capacidade e modernização. Isto fica ainda mais evidente com o aumento da competição na área de atuação, onde as cidades podem promover disputas entre empresas de saneamento para analisar qual empresa é a mais capaz de realizar as atividades. Com isso, a CASAN vem investindo pesado em melhorias internas e de imagem da empresa, sendo que um dos maiores investimentos atualmente é a obra em Florianópolis de aumento da rede de esgoto sanitário, aumentando de 45 para 75% a população atendida, com um custo estimado de R$ 400 milhões.

A automação tem efeito sobre as áreas em questão, promovendo melhorias internas nos processos e alterando a imagem da empresa perante a população e prováveis clientes, gerando a modernização. Por este motivo, há cerca de três anos foi criado em Florianópolis o setor de Pesquisa e Desenvolvimento, setor ao qual eu faço parte cuja meta é desenvolver sistemas que auxiliem a modernização e automação da CASAN.

Em meu estágio obrigatório feito no mesmo local, ajudei a desenvolver um ambiente de monitoramento remoto de centenas de pontos de atuação, contemplando ETEs (Estação de Tratamento de Esgoto), ETAs (Estação de Tratamento de Água), EEEs (Estação Elevatória de Esgoto) e *boosters*. Isto gerou interesse meu em desenvolver o projeto de fim de curso na mesma área.

Com isso chega-se ao foco deste projeto: utilizando da premissa de automatizar para modernizar, decidi aplicar a pesquisa para desenvolvimento de um sistema de Automação *wireless*, baseado em redes industriais, com tecnologias até então novas e não utilizadas pela CASAN, para atingir o objetivo de um projeto piloto capaz de se adequar as mais diversas aplicações da empresa.

Neste projeto foram realizados: pesquisa de tecnologias, soluções de integração entre componentes, construção de redes de comunicação, concepção do sistema de Automação, implementação do projeto piloto e demonstração prática.

Como resultado atingido, será demonstrado o sistema totalmente funcional para a aplicação em uma EEE, onde a unidade sensora coletará a informação de nível do poço úmido, enviará esta informação através de uma rede *Zigbee* sem fio até a unidade coordenadora, que por sua vez se comunicará com um Arduino, fazendo o papel de controlador, que tomará decisão e enviará pacotes contendo a medida a ser tomada para o atuador remotamente. Além disso o Arduino ainda se comunicará com uma IHM (Interface Homem-Máquina) através de uma rede Modbus que mostrará dados do processo e como um extra ainda poderá receber comandos de operador através da IHM, alterando variáveis e sinais para o atuador.

Abstract.

CASAN, Companhia Catarinense de Água e Saneamento, headquartered in Florianópolis, is a mixed capital company, created in 1970 whose purpose is to provide treated water, besides collecting and treating sewage. Present in more than 200 cities in Santa Catarina, currently serves a population over 2.5 million inhabitants.

Among the processes developed within the company, it might be mentioned: Capture of raw water, raw water treatment, transportation and reservation of treated water, collection, transportation and treatment of sewage.

These processes cited are always subject to improvement and studies, capacity expansion and modernization. This is even more evident with increasing competition in the area, where cities can promote disputes among water utilities to analyze which company is better able to carry out the activities. Thus, CASAN is investing heavily in internal improvements and company image, and one of the biggest investments currently in Florianópolis is the work of increasing network of sewage, increasing from 45 to 75% of the population served, with a cost estimated R$ 400 million.

The automation has effects on both areas, promoting internal improvements to processes and changing the company's image before the public and prospective clients, generating modernization. For this reason, about three years ago was created in Florianópolis the Research and Development sector, the sector to which I belong, where the goal is to develop systems that assist the modernization and automation of CASAN.

In my compulsory internship done in the same place, I helped to develop an environment for remote monitoring of hundreds of points of action, contemplating ETEs (Sewage Treatment Plant), ETAs (Water Treatment Plant), EEEs (Sewage Pumping Station) and boosters. This led to my interest to develop this project in the same area.

With it comes to the focus of this project: using the premise to automate for modernization, I decided to apply a research to develop a system of automation wireless networks based on industrial network technologies with new and hitherto unused by CASAN to achieve the goal of a pilot project able to fit many different applications of the company.

This project were carried out: research technologies, solutions of integration between components, construction of communication networks, system design automation, implementation of a pilot and demonstration practice.

As a result achieved, it will demonstrated the fully functional system for use in an EEE, where the sensor unit collects the information level of the wet pit and send this information via a Zigbee wireless network to the coordinating unit, which in turn communicates with an Arduino, playing the role of controller, which will take a decision and send packets containing the measure to be taken to the actuator remotely. Also the Arduino will still communicate with an HMI (Human Machine Interface) via a Modbus network that shows the process data and as an extra can still receive commands from the operator via the HMI, changing variables and signals to the actuator.

Sumário.

Agradecimentos.....4
Resumo. 5
Abstract. 7
Sumário. 9
Capítulo 1: Introdução. 13
1.1: Descrição do problema. 14
1.2: Localização e escopo.....16
1.3: Motivação.....17
1.4: Justificativa.....17
1.5: Argumentação principal e metodologia empregada. 18
1.6: Plano de desenvolvimento do assunto.....20
Capítulo 2: Apresentação. 21
2.1: A empresa.....21
2.2: Processos operacionais. 22
2.2.1: Estação Elevatória de Esgoto.....22
2.2.2: Estação de Tratamento de Esgoto. 24
2.2.3: Reservatório de água tratada. 29
2.2.4: *Boosters.* 32
2.2.5: Estação de Tratamento de Água. 32
2.2.6: Captação de água bruta. 38
2.3: Desafios Operacionais. 41
2.4: Enquadramento do projeto.....43
Capítulo 3: Fundamentação teórica.....45

3.1: Formalização do problema. 45
3.1.1: Problema foco. 47
3.2: Técnicas para solução. 50
3.2.1: Interface 51
3.2.2: Controle. 57
3.2.3: Controle de malha aberta. 57
3.2.4: Envio de dados. 72
3.2.5: Atuação. 105
3.2.6: Medição. 108
3.3: Considerações Finais do capítulo. 111
Capítulo 4: Desenvolvimento. 113
4.1: Desenvolvendo aplicações Zigbee 113
4.1.1: Escolha de modelos de equipamentos. 114
4.1.2: Primeiro exemplo prático. 116
4.2: Desenvolvendo aplicações Arduino. 121
4.2.1: Escolha de modelo de equipamento. 122
4.2.2: Ambiente de Desenvolvimento Arduino. 123
4.2.3: Exemplos de aplicação. 124
4.3: Integrando Arduino e Xbee. 126
4.3.1: Exemplos de aplicação. 128
4.4: Aprofundando os conhecimentos sobre a IHM. 131
4.4.1: Beijer H-T80c 131
4.4.2: Programação da IHM. 133
4.4.3: Interligando a IHM ao controlador. 135
4.4.4: Exemplo de aplicação. 139
4.5: Apresentação conceitual da solução. 141

4.6: Considerações finais do capítulo. 142
Capítulo 5: Implementação. 143
5.1: Solucionando o problema de interface. 143
5.1.1: Programação visual da IHM. 143
5.1.2: Programação de comunicação da IHM. 153
5.2: Implantando o Arduino na solução final. 156
5.2.1: Biblioteca Modbus. 157
5.2.2: Biblioteca SoftwareSerial. 157
5.2.3: Recebendo e criando pacotes API com o Arduino. 158
5.2.4: Apresentação e discussão do código do Arduino para o problema da EEE. 162
5.2.5: Alterações do código do Arduino para uma possível implementação em aeradores. 172
5.3: Programando os módulos Xbee para a solução final. 173
5.3.1: Programação do Xbee Coordenador. 174
5.3.2: Programação do Xbee Roteador. 177
5.3.3: Programação dos Xbee Dispositivos Finais. 178
5.4: Construção física dos equipamentos. 179
5.4.1: Kit IHM + Arduino + Zigbee Coordenador. 180
5.4.2: Kit Roteador. 184
5.4.3: Kit Dispositivo Final. 186
5.5: Teste, avaliação e validação da solução. 189
5.6: Considerações finais do capítulo. 192
Capítulo 6: Resultados. 193
6.1: Implementação na planta de simulação CIOM. 193
6.1.1: Aplicando a solução na planta de simulação. 194
6.2: Casamento de especificações e solução final. 196

6.3: Ganhos obtidos. 197
6.4: Impactos do projeto. 198
6.5: Considerações finais do capítulo. 198
Capítulo 7: Conclusões e Perspectivas. 200
Bibliografia: 202

Capítulo 1: Introdução.

Este trabalho foi desenvolvido no contexto da disciplina de Projeto de Final de Curso, com caráter obrigatório na grade curricular do curso de Engenharia de Controle e Automação da Universidade Federal de Santa Catarina. Esta cadeira tem características práticas, voltada para a execução real de problemas de engenharia. Seguindo esta linha de trabalho, este projeto foi elaborado com a finalidade de agregar novas tecnologias e ferramentas de automação úteis para a empresa em que trabalhei, abrindo portas para aumentar tanto meu conhecimento quanto o da mesma.

O trabalho presente neste documento envolveu:

- Estudo da problemática geral da CASAN, onde as áreas de atuação foram analisadas para descobrir possíveis pontos de ataque para a solução de automação presente neste projeto.
- Análise do nível de automação atual e compatibilidade com o sistema proposto no trabalho para que haja uma simbiose entre ambos e não conflitos.
- Procura bibliográfica de prováveis soluções na área, entendendo o funcionamento de cada uma delas e de como é feita a integração entre soluções parciais em uma solução global.
- Análise de equipamentos dos fornecedores, comparando versões diferentes para diferentes utilidades.
- Compra de equipamentos para a fase de testes, buscando aprimorar o entendimento das funcionalidades e aprovar soluções.
- Montagem da solução final com os equipamentos e configuração estudada na fase anterior.
- Ajustes finais na solução encontrada e montagem final do conjunto.
- Implementação e testes para validação final da solução.

Este trabalho foi supervisionado na empresa pelo Engenheiro Eletricista Ezequiel Medeiros e na Universidade pelo Professor Marcelo Ricardo Stemmer.

1.1: Descrição do problema.

Atualmente, regida pela Superintendência Regional Metropolitana da CASAN, com sede em Florianópolis, Santa Catarina, encontram-se centenas de pontos de coleta e retenção de esgoto, além de pontos de coleta e tratamento de água. Com isso, é bastante dificultada a operação e o controle da situação individualmente em cada ponto, sendo o caso um candidato potencial para um projeto de automação.

Entre os processos desenvolvidos dentro da empresa, pode-se citar: Captação de água bruta, tratamento de água bruta, transporte e reservação de água tratada, coleta, transporte e tratamento de esgoto sanitário.

Para a captação de água bruta, é realizado um estudo geológico de potenciais regiões para se encontrar vertentes de água, esta podendo ser lagos, rios, mananciais, poços, entre outros. A qualidade inicial desta água é que determina o quanto de tratamento é necessário para transformá-la em água potável, sendo que a poluição é o fator predominante nestes custos.

O tratamento de água considerado simples, quando a água bruta se encontra sem poluentes, consiste em dosar cloro (um bactericida) e flúor (para prevenção de cárie dentária). O tratamento de água considerado completo, geralmente realizado em ETAs (Estação de Tratamento de Água), consiste primeiramente na aeração da água (facilitando as reações químicas), coagulação (adição de produtos químicos transformando as impurezas dispersas em estado coloidal), floculação (formação de aglomerados de impurezas), decantação (onde os flocos de impurezas são retirados pelo efeito da gravidade), correção da dureza (onde são controlados os sais de cálcio e magnésio, que conferem cheiro e gosto na água), correção de pH (para corrigir acidez ou alcalinidade da água), desinfecção (eliminação de elementos patogênicos), fluoretação e oxidação (controle de elementos como o ferro presentes na água).

O transporte de água tratada, realizado por tubulação atingindo todos os diversos pontos das comunidades, pode ser realizado por gravidade, onde pelos princípios físicos a água se movimenta de um lugar mais alto naturalmente para um lugar mais baixo, ou por recalque, onde *boosters* são responsáveis por aumentar a pressão da água dentro da tubulação, podendo atingir localidades mais altas.

A reservação da água tratada é feita em reservatórios, com o intuito de possibilitar a armazenagem de água tratada em diferentes localidades, atendendo as necessidades das mesmas.

A captação do esgoto sanitário é feito por tubulações que vão de encontro até as residências e empreendimentos. Após a coleta, o esgoto passa por EEEs (Estação Elevatória de Esgoto), onde o mesmo é armazenado em tanques até ser enviado para as ETEs (Estação de Tratamento de Esgoto).

A CASAN utiliza principalmente três tipos de tratamento de esgoto: Lodo ativado, lagoa de estabilização e filtro biológico.

Lodo ativado é um processo biológico que consiste em reservar o esgoto em tanques que na presença de oxigênio, agitação mecânica e micro-organismos formar flocos denominados lodo ativado. Este lodo pode ser retirado posteriormente por decantação e enviado ao aterro sanitário.

Lagoa de estabilização é um processo simples e natural, com o objetivo de remover matéria orgânica. O processo geralmente passa por mais de uma lagoa, sendo que a primeira, denominada anaeróbia, tem o objetivo de minimizar ao máximo o oxigênio, para que a estabilização da matéria orgânica se dê estritamente em condições anaeróbicas. Depois deste processo, geralmente o esgoto restante passa por uma série de lagoas do tipo facultativo, que são lagoas repletas de algas onde na região superficial ocorrem processos fotossintéticos aeróbicos e no fundo ocorrem os processos anaeróbicos.

Filtros biológicos são destinados à oxidação da matéria biológica restante de decantadores O efluente do decantador é disperso sobre um leito de rochas onde o ar possa circular. Com isso, este leito de rochas desenvolve uma colônia de micro-organismos aeróbios. Em locais onde o ar não circula, desenvolvem-se organismos

anaeróbicos e com isso se gera uma condição semelhante às lagoas de estabilização.

Com uma complexidade alta operacional, a CASAN necessita de profissionais experientes e competentes nos mais diversos setores e campos de atuação, sempre buscando ferramentas administrativas e operacionais para facilitar o trabalho destes indivíduos.

O problema em questão é então responder a pergunta fundamental: “Como auxiliar e facilitar a operação dentro da empresa?”.

Este projeto visa ser uma ferramenta útil para a empresa, no intuito de auxiliar a base operacional nas suas tarefas, modernizando os processos e melhorando as condições de trabalho envolvidas.

A intenção do trabalho é propor uma solução em automação flexível, que possa ser utilizada em diversos processos diferentes citados acima onde basicamente deve-se receber um sinal da variável medida com o objetivo de controla-la, através da atuação sobre as variáveis de processo.

1.2: Localização e escopo.

Localizando o tema dentro do curso de Engenharia de Controle e Automação, este é um trabalho que contempla ambas as áreas, sendo que a ênfase maior é com a Automação.

No Brasil, as empresas de água e saneamento estão em uma verdadeira corrida, devido à competição acirrada entre elas, para modernização e melhorias operacionais. Com isso, os projetos de automação florescem cada vez mais nestas áreas. Esta área em crescente expansão atrai estudantes e profissionais do setor, sendo de grande interesse para mim.

Observar a proximidade da automação com essas empresas também é bastante evidente. Com milhares de atuadores, válvulas, sensores de nível, conjuntos motor-bomba, entre outros, estas oferecem um leque muito grande para projetos na área do curso, entre eles o que será apresentado.

1.3: Motivação.

A principal motivação deste trabalho é ter observado que uma grande parte dos processos da CASAN ainda são realizados totalmente de forma manual, com dificuldade em se observar o que está acontecendo e sem nenhum *feedback* das variáveis operacionais.

Como a CASAN é uma empresa que investe e incentiva a pesquisa de novas tecnologias, me senti motivado para implementar um projeto que atendesse esses anseios e que fizesse a empresa abrir os olhos para algumas soluções que ainda não eram empregadas na mesma.

O uso de soluções completas de automação implementadas por terceirizados também era algo que me incomodava. Fazer um projeto de concepção totalmente dentro da CASAN era algo que eu sempre imaginei que poderia abrir os olhos dos funcionários para uma mentalidade de “sim, nós podemos”, jargão comum em épocas de eleição, mas que na sua virtuose tem uma mensagem muito forte de acreditar no potencial.

Como trabalho no setor de Pesquisa e Desenvolvimento, outro fator de motivação para mim foi estar sempre em contato com equipamentos e tecnologias novas, as quais testamos regularmente para observar o comportamento e uma delas foi pilar para concepção deste projeto. Desde o início fiquei encantado com seu potencial e sabia que gostaria de fazer um projeto com aquela tecnologia.

1.4: Justificativa.

Justificando a escolha do projeto, é importante citar que a CASAN trabalha com o bem mais precioso de todos: a água. Como parte do corpo de engenharia, devo encontrar soluções que melhorem a qualidade do serviço tanto de água tratada como de esgoto para as comunidades espalhadas pelas cidades.

Com a base de meus conhecimentos, a melhor maneira que posso encontrar para auxiliar na meta é através dos projetos de controle e automação. Fazer com que operadores de EEEs, ETAs, ETEs, entre outros, tenham informações precisas com facilidade e possam confortavelmente acionar ou desacionar um motor, por exemplo, baseado na informação de um sensor, facilita o trabalho destes

operadores e faz com que eles possam gastar seu tempo com monitoramento de diversos pontos, ao invés de arduamente controlar de maneira manual um único ponto.

Este é o ponto que precisa ser melhorado na CASAN. É necessário criar mais ferramentas para auxílio da base operacional, pois é nesta fase que erros ou defeitos podem causar prejuízos às comunidades inteiras.

Além disso, trabalhar com redes de sensoriamento e controle sem fio (*wireless*), sempre foi um objetivo meu, pois entendo que este é o futuro da automação, desprendendo fisicamente pontos do sistema e interligando-os através da tecnologia. Poder concatenar meus anseios com as necessidades da CASAN transformou este projeto em uma meta agradável de ser atingida e não custosa e morosa.

1.5: Argumentação principal e metodologia empregada.

Para chegar à solução presente neste documento, foi realizada uma pesquisa bibliográfica extensa sobre tecnologias sem fio aplicadas na automação, elementos de tomada de decisão, elementos sensores, além de uma extensa pesquisa na parte de eletrônica para criar componentes de ligação entre os diversos elementos presentes. Nesta primeira etapa, três livros foram de fundamental importância e serão utilizados como base para a fundamentação teórica deste documento.

Após chegar a conclusão de algumas prováveis soluções, as mesmas foram testadas no laboratório da CASAN para observar comportamentos, anomalias, e para familiarização com as tecnologias diferentes.

Com isso foram levantados prós e contras de cada provável candidato, eliminando os mais fracos e mantendo os melhores.

Chega-se então à seguinte hipótese para solução:

- Tomada de decisão: Para a tomada de decisão e interpretação dos sinais provenientes das outras partes, neste projeto será utilizado o Arduino, introduzido por Margolis[1] como sendo: “Um ambiente de programação que embora tenha sido desenhado para ser fácil de

utilizar, o seu potente *hardware* trabalha no mesmo nível de sofisticação que engenheiros empregam em dispositivos embarcados".

- Rede de comunicação sem fio: Para tornar o Arduino um controlador remoto, utilizou-se neste projeto uma rede do tipo Zigbee, definida por Farahani[2] como "O padrão global para implementação de redes sem fio de custo baixo, baixa taxa de dados e curto alcance com vida de bateria estendida".
- Módulos Zigbee: Como Zigbee é apenas uma topologia de rede, são necessários dispositivos que entendam este protocolo e que façam a interligação entre o Arduino e o atuador, sendo que no projeto foram utilizados módulos XbeePro série 2.
- Sensores: Como nos módulos Xbee existem entradas analógicas, o sensor que se adequa os projeto deve ter como saída tensão de 0-1 V ou corrente de 4-20 mA. Como o último é bem mais comum que o primeiro, foi decidido sensores com saída de corrente para o projeto.
- Interface: Para melhorar a interface com o operador, neste projeto será utilizada uma IHM, onde o operador conseguirá inserir comandos e ler dados do processo.
- Comunicação entre IHM e Arduino: Para que a IHM possa mostrar na tela as informações do processo, é necessário ela se comunicar com o Arduino, e esta troca de informações se dará neste projeto através de uma Rede Modbus RTU.
- Atuadores: Geralmente um conjunto bomba-motor, pode ser acionado diretamente através de uma contatora quando a potência for inferior a 5 cv ou por *Soft Starters* quando a potência for superior, segundo norma nt01 da CELESC (Centrais Elétricas de Santa Catarina)[3].

Os elementos citados acima irão compor o projeto de automação presente nesta tese, de maneira a atingir os objetivos propostos anteriormente.

1.6: Plano de desenvolvimento do assunto.

No segundo capítulo será apresentada a empresa em que este projeto foi realizado, mostrando os processos e áreas de atuação para situar melhor o trabalho dentro do escopo da CASAN e de sua complexidade operacional.

No terceiro capítulo será apresentado o problema em questão deste projeto em si, mostrando todos os detalhes que circularam a escolha do mesmo. Também é apresentada a fundamentação teórica por trás dos conceitos apresentados neste projeto bem como iniciação e primeiras análises das tecnologias empregadas.

No quarto capítulo serão apresentadas as soluções envolvendo as tecnologias do capítulo anterior, apresentando onde elas serão inseridas no problema para melhorar o resultado final.

No quinto capítulo será apresentada a implementação física do projeto, mostrando telas de captura do desenvolvimento de *software* bem como explicações para decisões de comando e de execução.

No sexto capítulo serão apresentados os resultados do projeto, mostrando em que pontos o mesmo melhorou o fluxo de operação e a maneira do processo interagir com os operadores.

No sétimo capítulo será finalmente apresentadas as conclusões deste projeto e as perspectivas de futuro para novas implementações além de expectativas de melhorias.

Capítulo 2: Apresentação.

Neste capítulo será apresentado o ambiente em que esse trabalho está imerso. Será apresentada a empresa em que o projeto se realizou, os processos que tem relevância para esta proposta, além de esquemáticos para auxiliar a compreensão da CASAN como um todo.

2.1: A empresa.

A CASAN, cuja missão é: “fornecer água tratada, coletar e tratar esgotos sanitários, promovendo saúde, conforto, qualidade de vida e desenvolvimento sustentável” [4], está presente em 201 municípios catarinenses e 01 paranaense, atuando diretamente nestes dois setores (fonte: CASAN, 2013).

A CASAN está dividida em quatro Superintendências Regionais de Negócios, sendo elas: Norte Vale do Itajaí, Oeste, Sul/Serra, e Metropolitana da grande Florianópolis.

Criada em 31 de Dezembro de 1970 através da Lei Estadual n.º 4547 e constituída em 02 de Julho de 1971 com o objetivo de coordenar o planejamento e executar, operar e explorar os serviços públicos de esgotos e abastecimento de água potável, bem como realizar obras de saneamento básico, em convênio com os municípios do Estado.

A primeira obra se deu no município de Urubici em 1972, com a implantação da rede de água tratada no município e desde lá vem crescendo tanto em tamanho quanto em investimentos, podendo-se citar o investimento de mais de R$ 404 milhões de reais em 2012 para execução de obras de água e esgoto nos municípios de Florianópolis, Criciúma, São José, Biguaçu, Concórdia e Rio do Sul (fonte: CASAN).

As ações desenvolvidas pela empresa neste período contribuíram para a evolução da qualidade de vida da população de Santa Catarina, apresentando um quadro que evolui de maneira favorável e consolidada. Sua atuação é fator preponderante no desenvolvimento econômico e social em sua área de concessão.

Segundo dados de 2013, 96.5% da população urbana dos municípios atendidos pela CASAN são contemplados com água tratada (fonte: CASAN, 2013), fazendo deste o índice mais alto do Brasil. Em Florianópolis o percentual chega a 99.5%[5].

Quando o assunto é esgoto sanitário, os índices são bem mais baixos, e Santa Catarina é o 11º pior Estado em tratamento de esgoto do Brasil (fonte: Ministério Público de Santa Catarina, 2008)[6], com 16% dos municípios com rede de esgoto sanitário. As médias dos municípios atendidos pela CASAN são de 19.8% sobre a população contemplada. Em Florianópolis, este número chega a 51,6% e Itá é o único município catarinense com 100% de índice de atendimento urbano de esgoto (fonte: CASAN, 2013).

Alguns dados brutos da CASAN ajudam a demonstrar o tamanho da empresa, como por exemplo, (fonte: CASAN, 2007):

- Produção de água tratada: 203.519.136.000 de litros por ano.
- Número de estações de tratamento: 299.
- Tubulação de adutoras: 921 km.
- Rede de distribuição: 10.975 km.

Este projeto foi desenvolvido no CIOM, Centro Integrado de Operação e Manutenção, na região da grande Florianópolis, no Departamento de Pesquisa e Desenvolvimento, parte integrante da SRM-Florianópolis.

2.2: Processos operacionais.

A CASAN atua sobre diversos processos desde a captação de água bruta até o tratamento do esgoto sanitário. Entre estes se podem citar: Estação Elevatória de Esgoto, Estação de tratamento de Esgoto, reservatórios de água tratada, *boosters*, Estação de Tratamento de Água e Captação de água bruta.

2.2.1: Estação Elevatória de Esgoto.

Todas as vezes que por algum motivo não seja possível, sob o ponto de vista técnico e econômico, o escoamento dos esgotos pela ação da gravidade, é

necessário o uso de elevatórias para elevar o esgoto de um ponto para outro de cota normalmente mais elevada.

Segundo a Associação Brasileira de Normas Técnicas (1992), na NBR 12.208[7], “é a instalação destinada ao transporte de esgoto do nível do poço de sucção das bombas ao nível de descarga da saída do recalque, acompanhando aproximadamente as variações de vazões afluente”.

Como partes componentes de uma EEE podemos citar: dispositivos de entrada, unidades de remoção de sólidos, medidor de nível, poço úmido, conjunto bomba-motor e poço seco.

O esgoto entra em uma EEE através de uma tubulação de entrada protegida contra resíduos sólidos (grade de proteção) e é reservada no poço úmido.

O sensor de nível, instalado no poço úmido, gera o sinal para que o conjunto bomba-motor, localizado no poço seco, atue, enviando o esgoto pela tubulação de saída com uma pressão de recalque maior que a de sucção. Com isso o esgoto pode ser enviado para outras EEEs ou para estações de tratamento.

A figura abaixo representa uma EEE genérica, para melhor entendimento:

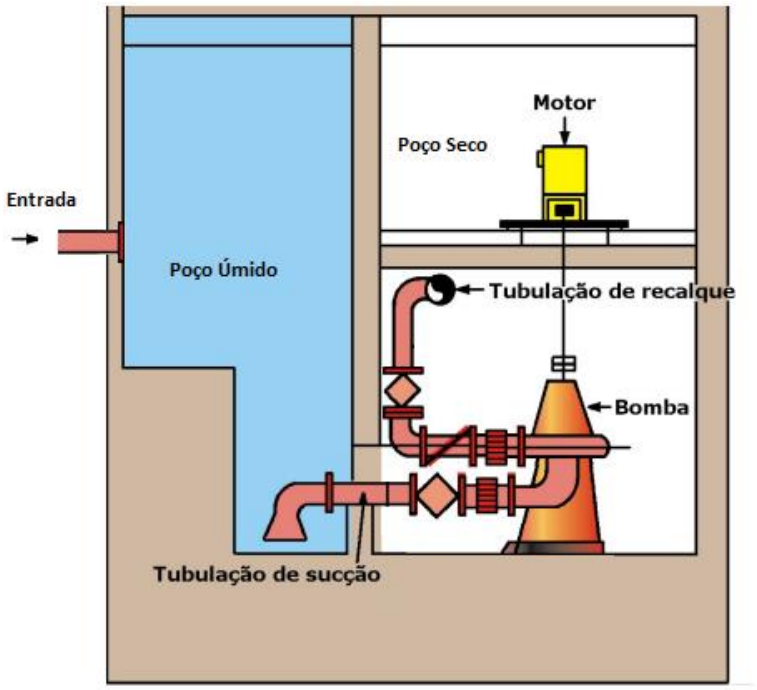

Figura 1- Estação Elevatória de Esgoto (Fonte: CASAN).

Geralmente o conjunto fica imerso em baixo da superfície, como na maioria dos casos da CASAN. O poço úmido tem uma limitação de capacidade de esgoto que pode conter e é especialmente importante que este poço nunca extravase, pois causaria uma contaminação do solo com esgoto. Por isso tem-se muito cuidado para dimensionar estes poços conforme o tamanho da comunidade que ele atende e há uma manutenção constante no conjunto bomba-motor para que os mesmos nunca fiquem estragados quando há a necessidade dos mesmos.

A CASAN ainda implanta mais uma medida de segurança nestas elevatórias, que é a presença de um conjunto bomba-motor reserva, para que possa assumir o funcionamento em caso de parada ou manutenção do outro conjunto.

Pelas elevatórias de esgoto da CASAN, em Santa Catarina, passam por ano, 20.377.791.000 de litros de esgoto (fonte: CASAN, 2013). Abaixo é apesentada a foto de uma das elevatórias de Florianópolis, localizada no bairro Bom Abrigo, região continental.

Figura 2- Elevatória de Esgoto J2A, Bom Abrigo (Fonte: CASAN).

2.2.2: Estação de Tratamento de Esgoto.

Também conhecido como ETAR (Estação de Tratamento de águas Residuais), a ETA é uma infraestrutura que trata os efluentes provenientes de residências, comércios e indústrias nas cidades. O tratamento é realizado até que

esta água esteja com um nível de poluentes aceitável pelo órgão vigente, neste caso a FATMA (Fundação de Amparo à Tecnologia e Meio Ambiente). Após este estágio a água tratada é retornada ao meio ambiente.

Como citado anteriormente, o tratamento de esgoto geralmente é realizado de três maneiras: Lodos ativados, lagoas de estabilização e filtro biológico. Em Florianópolis existem implementações de lodos ativados e lagoas de estabilização.

2.2.2.1: Lodo ativado.

Lodo ativado é o lodo resultante de um processo de tratamento de esgoto destinado à destruição de poluentes orgânicos biodegradáveis presentes em águas residuárias, efluentes e esgotos. O processo se baseia na oxidação da matéria orgânica, por bactérias aeróbias, controlada pelo excesso de oxigênio em tanques de aeração e posteriormente direcionado aos decantadores. O lodo decantado nos decantadores retorna ao tanque de aeração como forma de reativação da população de bactérias no tanque de aeração. Este retorno se dá na entrada do tanque onde o lodo em fase endógena se mistura ao efluente rico em poluente, aumentando assim a eficiência do processo.

É fundamental que a água a ser tratada não possua outros componentes que prejudiquem a vida de tais bactérias. As condições adequadas para o tratamento, tais como a concentração de oxigênio dissolvido, pH e a velocidade da água são essenciais ao perfeito funcionamento desse processo.

No Sistema ETE Insular da CASAN, ocorre um processo denominado Lodo Ativado de Aeração Prolongada. Este tratamento se divide em duas fases distintas: Pré-tratamento e tratamento secundário.

Na fase de pré-tratamento o esgoto é preparado para a próxima fase, e tem o objetivo de retirar partículas sólidas do mesmo. Primeiramente o esgoto passa por um sistema de gradeamento com duas grades mecanizadas do tipo cremalheira, evitando a passagem de objetos sólidos grandes. Uma imagem de uma grade cremalheira é apresentada na próxima página:

Figura 3- Grade com cremalheiras.

Estas grades tipo cremalheira podem subir até a superfície para a manutenção ou limpeza, e descerem novamente até a tubulação de esgoto para cumprir seu objetivo.

Após o esgoto passar pela grade, ocorre a desarenação. Este processo pretende retirar grãos de areia e outras partículas sólidas através de um processo mecânico do tipo gravimétrico, utilizando uma caçamba do tipo *clamshell* (concha dupla) e operada por uma talha elétrica.

A segunda fase, denominada tratamento secundário, é composto das seguintes unidades: Seletor biológico, câmara de desnitrificação, tanques de aeração, decantador secundário, adensadores de lodo, sistema de desidratação e prensa desaguadora.

O seletor biológico tem a função de misturar suavemente o esgoto pré-tratado com lodo ativado, evitando assim o desenvolvimento de micro-organismos indesejáveis ao tratamento e melhorando a sedimentação do lodo.

A câmara de desnitrificação recebe a mistura e tem função de reduzir o nitrato de iodo sob a ação de micro-organismos específicos.

Os tanques de aeração recebem da câmara e despejam o conteúdo em reservatórios onde é feita a aeração através de aeradores mecânicos. A quantidade de oxigênio inserida pelos aeradores propicia o desenvolvimento de bactérias

aeróbicas que irão digerir a matéria orgânica carbonácea e a nitrificação do nitrogênio orgânico total remanescente do efluente.

No decantador secundário, os flocos formados nos tanques de aeração se sedimentam e são encaminhados para um poço central onde o lodo se acumula (dai o nome lodo ativado).

Os adensadores de lodo servem para reduzir a quantidade de água presente no lodo sedimentado, e geralmente são do tipo gravimétrico.

O sistema de desidratação é um tanque de armazenamento do lodo adensado onde um sistema de pré-condicionamento químico à base de suspensão de cal e solução de polieletrólito prepara o lodo para seu descarte final.

Por último, a prensa desaguadora é responsável por comprimir o lodo resultante em blocos que serão despejados no aterro sanitário.

Uma foto da ETE Insular é mostrada abaixo:

Figura 4- ETE Insular (Fonte: Diário Catarinense).

Na foto acima, o tanque quadrado com divisões é onde ficam os aeradores, os tanques redondos maiores são os decantadores e os tanques redondos menores são o adensador e sistema de desidratação.

2.2.2.2: Lagoas de estabilização.

Os Sistemas de Lagoas de Estabilização constituem-se na forma mais simples para o tratamento dos esgotos. Há diversas variantes dos sistemas de lagoas de estabilização com diferentes níveis de simplicidade operacional e requisito de área.

Na CASAN, o Sistema de Potecas na grande Florianópolis utiliza uma lagoa anaeróbia seguida de três lagoas facultativas chicanadas (com desníveis).

A Lagoa Anaeróbia é uma das unidades de tratamento, onde a existência de condições estritamente anaeróbias é essencial. Tal é alcançando através do lançamento de uma grande carga de DBO (Demanda Bioquímica de Oxigênio) por unidade de volume da lagoa, fazendo com a taxa de consumo de oxigênio seja várias vezes a taxa de produção. No balanço de oxigênio, a produção pela fotossíntese e pela aeração atmosférica é neste caso, desprezível.

Nas lagoas facultativas o processo principal que ocorre é a fotossíntese. Para ocorrência desta, é necessária uma fonte luminosa, neste caso, representada pelo sol. Por esta razão na região de Florianópolis, provida de elevada radiação solar e baixa nebulosidade é propícia para a utilização desta unidade de tratamento.

A fotossíntese por depender da energia solar, é mais elevada próxima á superfície da lagoa. À medida que se aprofunda na lagoa, a penetração da luz é menor o que ocasiona a predominância do consumo de oxigênio dissolvido a partir de certa profundidade. Adicionado ao fato, a fotossíntese só ocorre durante o dia, fazendo com que durante a noite possa prevalecer à ausência de oxigênio.

Devido a estes fatos é essencial que haja diversos grupos de bactérias, responsáveis pela estabilidade da matéria orgânica, que possam sobreviver e proliferar, tanto na presença, quanto na ausência de oxigênio.

O processo de lagoas Facultativas é essencialmente natural, não necessitando de nenhum equipamento. Por esta razão, a estabilização da matéria orgânica se processa em taxas mais lentas, implicando na necessidade de um elevado período de detenção na lagoa (usualmente superior a 20 dias). A

fotossíntese, para que seja efetivada, necessita de uma elevada área de exposição para melhor aproveitamento da energia solar pelas algas.

As chicanas auxiliam a sedimentação dos dejetos sólidos nas regiões mais profundas ao propiciar barreiras para os mesmos na circulação.

Na figura baixo é apresentada uma foto do Sistema Potecas:

Figura 5- Sistema Potecas (Fonte: CASAN).

Nesta imagem a lagoa triangular maior é a lagoa anaeróbica e as outras são as lagoas facultativas com as chicanas aparentes.

2.2.3: Reservatório de água tratada.

Os reservatórios de água tratada são unidades hidráulicas de acumulação e passagem de água situados em pontos estratégicos das cidades com o intuito de atender as seguintes situações:

- Garantia de quantidade de água.
- Garantia de adução com vazão constante.
- Garantia de absorção dos picos de demanda.
- Reserva emergencial para suprir necessidades de outras localidades.

Quanto à sua configuração, os reservatórios podem ser: enterrados, semi-apoiado, apoiado, elevado e *stand pipe*.

Um reservatório enterrado é aquele que está completamente embutido no terreno, sem ser visível. O reservatório semi-apoiado, por sua vez, é aquele onde parte da construção se encontra abaixo do nível do terreno e parte acima. O apoiado tem a laje de fundo apoiada no terreno em que se encontra, sendo totalmente visível. O reservatório elevado apoia-se em estruturas de elevação e o *stand pipe* é semelhante ao elevado, porém contem uma estrutura de elevação embutida de forma a manter contínuo o perímetro de secção transversal da edificação.

Uma imagem dos tipos de reservatórios é apresentada abaixo, onde NT representa o nível do terreno:

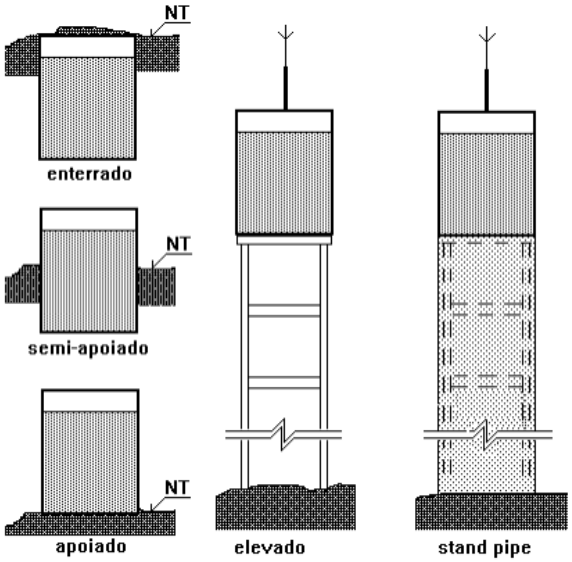

Figura 6- Configurações de reservatórios.

Os dois tipos de reservatórios mais comuns são os semi-apoiados e os elevados. Os elevados são utilizados quando se necessita uma cota piezométrica de montante superior a cota de apoio do reservatório no terreno local, ou seja, quando a rede de água necessita de uma pressão mínima e as cotas do terreno não oferecem tais condições, sendo necessário criar altura artificialmente.

Já os reservatórios semi-apoiados oferecem uma estabilidade estrutural elevada aliada a uma facilidade de construção e manutenção e são comuns quando se pode escolher um terreno de mais elevação natural. Na CASAN, encontram-se exemplos de reservatórios semi-apoiados, apoiados, elevados e *stand pipe.*

Em relação à localização no sistema, os reservatórios podem ser:

- Montantes.
- Jusantes.

Os reservatórios montantes são os que atuam antes da rede de distribuição de água, sendo que estes tem entrada por sobre o nível máximo de água e saída no nível mínimo e tem dimensionamento efetivo para manter a vazão do sistema de adução constante.

Os reservatórios jusantes são instalados após a rede de distribuição e servem para armazenar água nos períodos em que a capacidade da rede for superior à demanda. Quando a situação se inverte eles proporcionam a água extra necessária para abastecer as comunidades. Estes tem apenas uma tubulação que serve de entrada e saída simultaneamente, localizada no fundo do reservatório.

Como a CASAN tem uma rede de distribuição de recalque, onde a meta é manter a pressão constante em todas as localidades, o nível do reservatório varia durante o dia conforme o consumo das comunidades, aumentando com o baixo consumo e diminuindo com o aumento do consumo. Este comportamento pode ser visto no gráfico do nível do reservatório Caiobig, em Florianópolis:

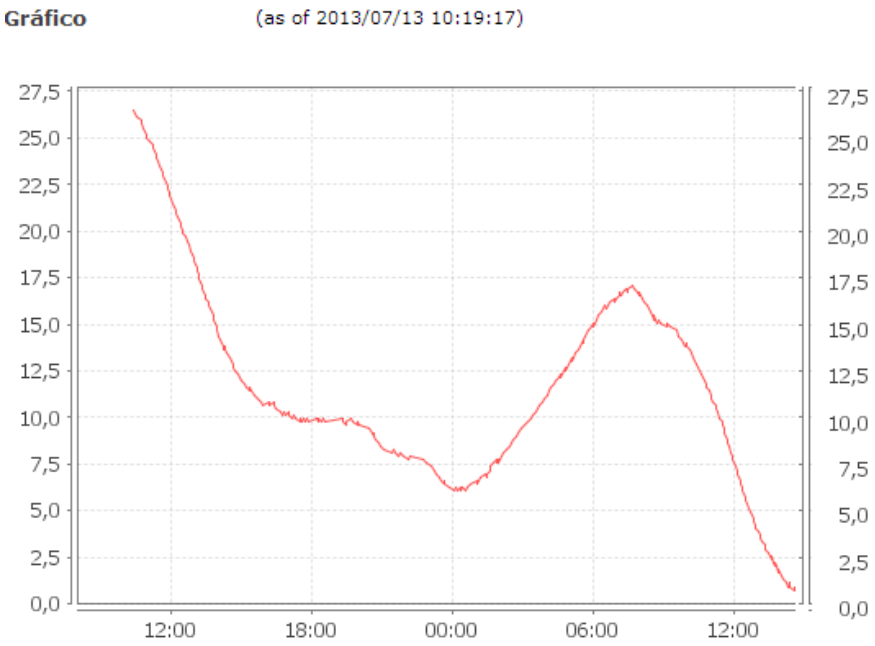

Figura 7- Nível do reservatório Caiobig (Fonte: Supervisório CASAN).

Este gráfico, retirado do dia 13/07/2013 do supervisório da CASAN, mostra em seu eixo x as horas e em seu eixo y a porcentagem de água total do reservatório

que está cheia no determinado horário. Pode-se perceber o comportamento variável citado acima.

2.2.4: ***Boosters.***

Boosters, ou em tradução livre: intensificadores, são bombas que intercaladas à tubulação aumentam a pressão, auxiliando o escoamento de água. Estas proporcionam a pressão necessária quando as condições topográficas ou as perdas de carga nas linhas assim o exigirem.

Em um *booster*, as duas variáveis em questão são a pressão de sucção e a pressão de recalque. O equipamento irá atuar quando a pressão de sucção estiver abaixo de um valor nominal estudado, fazendo com que a pressão de recalque (saída) atinja este valor através da atuação do motor.

Através do estudo piezométrico da rede de abastecimento de água, a CASAN pode decidir os locais em que há a necessidade de implantação de um *booster*, sendo que temos exemplos como: *booster* Graciliano Gomes, *booster* Tecnópolis, *booster* Praia Brava, todos localizados em Florianópolis.

2.2.5: Estação de Tratamento de Água.

Estações de tratamento de água são infraestruturas que recebem como entrada água bruta, proveniente da captação e tem como saída água tratada, potável para consumo conforme portaria de potabilidade, pronta para ser distribuída pelas redes de abastecimento.

Como finalidades do tratamento de água se pode citar:

- Higiênica: remoção de bactérias, elementos nocivos ou venenosos, minerais e compostos orgânicos em excesso, protozoários e outros microrganismos.
- Estética: correção da cor, turbidez, odor, sabor.
- Econômica: redução de corrosividade, dureza, turbidez, entre outros.

As ETAs no Brasil sofrem controle rigoroso por parte da Companhia atendendo exigência de legislação específica do Ministério da Saúde – Portaria nº 518/2004[8].

Esta Portaria contém parâmetros de qualidades física, química e bacteriológica que a água deve possuir para ser considerada potável.

Também consta na Portaria a frequência com que amostras devem ser coletadas para as respectivas análises, sendo que o quantitativo de coletas é função da população atendida pelo sistema.

No controle de qualidade da água distribuída é dada ênfase especial à qualidade bacteriológica, sendo que tanto a frequência como o número de coletas é significativa.

Seguindo as recomendações da Portaria, para tratamento de água existem diversas técnicas aplicadas conforme a qualidade da água bruta coletada, sendo que a CASAN se utiliza das seguintes: Casa de Química com desinfecção, Filtração lenta, Estação convencional, ETA compacta fechada, Filtração direta ascendente e Filtração direta descendente. Uma breve discussão sobre cada uma das estratégias será apresentada nos próximos tópicos.

2.2.5.1: Casa de Química com desinfecção.

Quando a água bruta proveniente da captação tem uma qualidade inicial muito boa, o único tratamento necessário é a adição de cloro antes de ser enviada para a rede de abastecimento.

Na casa de Química, uma bomba de cloro retira cloro líquido de um recipiente de armazenamento e envia para um dosador, que regula a quantidade de cloro que será inserida na água conforme índices pré-estabelecidos pois o excesso pode ser nocivo à saúde.

O cloro (Cl_2) é um elemento químico de alta reatividade, sendo que não é encontrado isolado na natureza, geralmente aparecendo sobre a forma de cloretos. Quando os cloretos sofrem processos de eletrólise, um dos resultantes da reação química é o cloro.

No tratamento de água, o cloro é utilizado pelo seu poder desinfetante, matando microrganismos patogênicos, sendo que os mais comuns são protozoários que podem causar doenças como a Giardíase e a Amebíase, comuns em lugares sem saneamento básico.

Geralmente águas de poços artesanais são filtradas pela própria terra e servem como um bom exemplo de água bruta que necessita apenas deste tratamento.

Na CASAN, existem diversos poços em Florianópolis, na região da Praia dos Ingleses e Praia do Santinho em que se utiliza esta técnica. As casas de Química, neste caso, se localizam em anexo aos poços de captação, evitando desperdícios por transporte. A ETA de Xavantina, mostrada na imagem abaixo também é outro exemplo:

Figura 8- Casa de Química (Fonte: CASAN).

NA imagem acima, a dosagem de cloro é feita nos tanques até completar o tratamento e depois enviado para as redes de abastecimento.

2.2.5.2: Filtração lenta.

Ambos os processos de filtração lenta e casa de Química trabalham com uma água bruta de boas qualidades, porém a água deste processo é proveniente de

captação superficial, como rios e lagos, logo não possuindo a fase de filtramento natural através do solo.

Devido a esta falta de filtro, a primeira etapa do processo é justamente a filtração. Este processo consiste em fazer a água bruta passar lentamente através de materiais porosos capazes de reter ou remover as impurezas. Em geral os componentes dos filtros são: seixos, areia e carvão antracitoso (carvão fóssil de alto teor de carbono).

Após a filtragem da água bruta, esta recebe a dosagem de cloro indicada como no processo anterior e é enviada a rede de abastecimento.

2.2.5.3: Estação convencional.

Tipo de tratamento mais comum na CASAN, este processo considera que a água bruta captada contém poluentes e agentes nocivos à população atendida. É necessário então, realizar todo o tratamento antes de enviar a água tratada ao sistema de abastecimento.

O tratamento convencional consiste dos seguintes passos:

- Aeração: Processo de tratamento pelo qual a área de contato entre água e o ar é aumentada, de modo a facilitar o intercâmbio ou troca de gases e substâncias voláteis entre a água e o ar.
- Coagulação: Tem por objetivo transformar as impurezas que se encontram em suspensões finas, em estado coloidal, e algumas que se encontram dissolvidas em partículas que possam ser removidas pela decantação ou flotação e filtração. Como principais agentes coagulantes podem ser citados o sulfato de alumínio, cloreto férrico e o policloreto de alumínio.
- Floculação: Processo que ocorre logo após ou simultaneamente com a coagulação e cuja característica fundamental é a formação de aglomerados gelatinosos chamados flocos, resultantes da reação entre o produto químico coagulante e as impurezas da água.

- Decantação ou Flotação: A decantação é o processo pelo qual se verifica a deposição dos flocos pela ação da gravidade. Na flotação, por injeção de ar, os flocos, ao invés de sedimentarem, vão à superfície onde são recolhidos.
- Tratamento por contato: Utilizado apenas nas ETAs que não necessitam de unidades de decantação e floculação, sendo que após coagulada a água é encaminhada diretamente aos filtros para retenção das impurezas.
- Correção de dureza: Neste processo são controlados excessos de sais de cálcio e magnésio presentes na água, que têm características incrustantes e conferem gosto.
- Correção de pH: Aplicação de produtos químicos visando corrigir a acidez ou alcalinidade excessivas da água. Esta providência visa principalmente proteger estruturas de distribuição e armazenamento de água. Os principais corretores são: hidróxido de cálcio, hidróxido de sódio e carbonato de sódio.
- Desinfecção: Destruição ou inativação de organismos patogênicos, capazes de produzir doenças, ou de outros organismos indesejáveis. Os principais desinfetantes são o cloro, o hipoclorito de sódio e o hipoclorito de cálcio.
- Controle de sabor e odor: Processos químicos e físicos que visam melhorar o paladar. As principais causas para o odor e sabor são as algas, decomposição de vegetais, bactérias e resíduos industriais.
- Fluoretação: Aplicação de fluossilicato de sódio ou ácido fluossilícico (mais conhecidos como flúor) para prevenção de cáries dentárias.
- Oxidação: Processo químico responsável pela retirada de metais da água através da oxidação. Utilizam-se ortopolífosfatos para gerar a reação.

Exemplos de estações da CASAN que utilizam este tratamento são: ETA Morro dos Quadros (Santo Amaro da Imperatriz), ETA Caçador e ETA Criciúma (mostrada abaixo):

Figura 9-. ETA Criciúma (Fonte: CASAN).

2.2.5.4: ETA Compacta fechada.

São estações que também realizam o tratamento convencional, sendo que estas unidades funcionam sob pressão, forçando a entrada de água bruta pela tubulação.

Estas ETAs são adquiridas prontas com capacidades de tratamento estabelecidas pelo fabricante. Em geral são utilizadas em cidades de pequeno porte.

A ETA da cidade de Meleiro, Santa Catarina, mostrada abaixo, é um exemplo de ETA compacta fechada:

Figura 10- ETA Meleiro (Fonte: CASAN).

2.2.5.5: Filtração direta ascendente.

As águas apropriadas para este tipo de tratamento possuem características de qualidade que exigem tratamento convencional, mas sem a necessidade de passagem por unidades específicas para floculação e decantação.

Neste tipo de ETA a filtração se dá em fluxo inverso, ou seja, de baixo para cima.

2.2.5.6: Filtração direta descendente.

A mesma descrição de filtração direta ascendente, com a diferença de que neste tipo de tratamento o sentido do fluxo é descendente.

2.2.6: Captação de água bruta.

A captação de água bruta é feita em mananciais. Mananciais são todas as fontes de água, superficiais ou subterrâneas, que podem ser usadas para o abastecimento público. Isso inclui, por exemplo, rios, lagos, represas e lençóis freáticos.

Falando mais especificamente de Florianópolis, os mananciais para abastecimento da cidade estão distribuídos em lençóis subterrâneos no norte da ilha e superficiais nas demais regiões, sendo sete insulares e um continental. São estes:

- Manancial da Lagoa da Conceição: Localizado no topo do morro da Lagoa da Conceição, é composto por um córrego com vazão de captação de 10 l/s. Este manancial abastece cerca de 3500 pessoas das regiões da Lagoa da Conceição e parte do bairro Itacorubi até proximidades do bairro Parque São Jorge. O tratamento da água bruta deste manancial é feito por casa de Química de onde segue diretamente para abastecimento.
- Manancial do Quilombo: Formado pelo córrego do Itacorubi, está localizado na comunidade hoje conhecida como Quilombo, abastecendo cerca de 1700 habitantes e tem vazão média de captação de 4 l/s. O sistema de abastecimento é composto pela captação, uma

casa de Química para tratamento da água e um reservatório para armazenamento e distribuição.

- Manancial Rio Tavares: Composto pelo Rio Tavares, o manancial abastece uma população de aproximadamente 8 mil pessoas nas áreas da Costeira do Pirajubaé e nas áreas médias e altas da Avenida Jorge Lacerda. Tem uma vazão de captação de 20 l/s e possui tratamento por casa de Química, de onde segue diretamente para abastecimento.
- Manancial Monte Verde: Composto pelo rio Pau do Barco abastece 4000 habitantes das comunidades de Monte Verde e Parque da Figueira, no bairro do Saco Grande. Tem uma vazão de captação de 7 l/s e possui um sistema de abastecimento composto de uma casa de Química e um reservatório.
- Manancial Cidade das abelhas (Rio do Mel): Formado pelo rio de mesmo nome, abastece uma população de 2300 habitantes dos bairros Saco Grande, Cacupé, Santo Antônio da Lisboa e Sambaqui. Tem uma vazão de captação de 4 l/s e o sistema de abastecimento é composto por uma adutora que conduz a água bruta até a casa Química do Cacupé, onde é feito o tratamento e armazenamento em reservatório.
- Manancial do Cacupé: Formado pelo córrego do Meimbipe, é integrado com o sistema do manancial Cidade das abelhas, onde ajuda a abastecer as mesmas regiões. Tem uma captação de 4 l/s. O tratamento é realizado na casa de Química do Cacupé.
- Manancial Córrego Grande: Composto pelo córrego de mesmo nome, este abastece uma população de 8000 habitantes dos bairros Jardim Anchieta e Córrego Grande. Tem uma vazão de captação de 14 l/s e como tratamento uma casa de Química.
- Manancial Ribeirão da Ilha: Composto pelo rio da ASCAN, quando necessário é utilizado para abastecer a população do norte do Ribeirão

da Ilha, junto com o sistema da Lagoa do Peri. Tem uma captação de 5 l/s e trata a água bruta através de uma casa de Química.

Somente estes mananciais não podem sozinhos abastecer todo o contingente de habitantes de Florianópolis, por isso o abastecimento é formado também pelos sistemas Costa Norte, Costa Leste-Sul e Cubatão-Pilões.

O sistema de abastecimento da Costa Norte tem água bruta captada de mananciais subterrâneos compostos por 22 poços de captação, localizados na região do Sítio de Capivari, no distrito de Ingleses e distrito do Rio Vermelho, localizados no norte da ilha. Além dos poços, o sistema é formado por uma Estação de Tratamento de Água (ETA Ingleses), um *booster* para recalque e reservatórios localizados nos distritos de Ingleses, Canasvieiras, Jurerê e Daniela.

Devido ao fato do norte da ilha ser uma região balneária, a vazão de produção dos poços é variável e sazonal. No inverno a vazão média é de aproximadamente 110 l/s e atende aproximadamente 64 mil habitantes. No verão a captação sobe para 300 l/s os quais atendem cerca de 130 mil habitantes.

Na ETA dos Ingleses, a água dos poços é tratada, através do tratamento convencional, passando posteriormente para os reservatórios. Este sistema é responsável atualmente pelo abastecimento dos distritos de: Rio Vermelho, Ingleses, Santinho, Ponta das Canas, Canasvieiras, Jurerê, Daniela e Ratones.

Já no sistema de abastecimento da Costa Leste-Sul a captação é realizada em uma barragem na Lagoa do Peri, onde uma vazão de aproximadamente 200 l/s é captada. Esta água bruta passa pela Estação de Tratamento de água da Lagoa do Peri, onde o tratamento convencional é realizado.

A água tratada é então bombeada para os diversos reservatórios localizados na região, para ser distribuída para a população. Atualmente, o sistema abastece cerca de 110 mil habitantes da região da Lagoa do Peri e sul da ilha.

Por sua vez, o sistema de abastecimento de Cubatão-Pilões é o maior dos três, sendo que a água bruta é captada dos rios Vargem do Braço e Cubatão, localizados no município de Santo Amaro da Imperatriz. Esta água é tratada na ETA Morro dos Quadros, também em Santo Amaro da Imperatriz e com uma vazão de

captação aproximada de 1400 l/s abastece cerca de 700 mil habitantes das cidades de Florianópolis, São José, Santo Amaro da Imperatriz e Palhoça.

2.3: Desafios Operacionais.

Nos tópicos anteriores foram demonstrados os processos operacionais da CASAN com o intuito de iniciar a base do trabalho, situando o mesmo no contexto em que ele se encontra.

Fica claro que a empresa atua em uma área muito abrangente, com dificuldades e complexidades operacionais muito amplas, aumentados ainda mais pelo porte da própria CASAN.

Com a divisão de superintendências, os esforços podem ser concentrados mais exclusivamente na região de atuação, focando a resolução dos problemas destas áreas.

Falando de Florianópolis, um grande desafio da CASAN é lidar com a população sazonal, onde no verão a cidade e visitada por centenas de milhares de turistas de todas as localidades do mundo. Este conflito de duas cidades totalmente diferentes, dependendo de qual época do ano está faz com que o dimensionamento de sistemas de abastecimento de água e captação de esgoto seja muito difícil.

É por esse motivo, principalmente, que eficiência operacional e de processos é um tema tão importante na CASAN. Por trabalhar no máximo da capacidade durante o verão, folgas são mínimas, e espaços para erros e desperdícios devem ser praticamente nulos, pois impactam o sistema como um todo.

Com isso, os departamentos sempre procuram maneiras e soluções que possam melhorar os processos, desde ferramentas que possam auxiliar os operadores a desenvolver seus trabalhos mais rapidamente e facilmente, até soluções de grande porte, onde toda uma cadeia de processos é modificada para atender a alguma especificação ou necessidade.

Voltado mais para a Automação, os grandes desafios nesta área dentro da empresa são relacionados mais a área operacional em si. Florianópolis possui centenas de pontos de operação da CASAN, onde algum tipo de equipamento realiza uma função citada nos capítulos anteriores. A falta de um padrão em alguns

casos e a falta de ferramentas de auxílio para os operadores causam o efeito de muitos destes pontos não estarem alinhados aos preceitos de eficiência necessários.

A base da cadeia operacional da CASAN é a informação. É ela que vai gerar as ações que os operadores devem realizar e é ela o alicerce das decisões. Por isso, se a informação é prolixa, sem sentido, ou não chega diretamente e rapidamente aos operadores, o que irá ocorrer é a tomada tardia de decisões ou, em casos extremos, a tomada incorreta de decisões.

Um dos problemas atuais da CASAN é justamente a falta da informação. Em muitos pontos, o operador só obtém a informação que ele precisa se for observar o processo por si mesmo. Além de ser antiprodutivo, este método sobrecarrega o funcionário, que por muitas vezes já tem atribuído a ele muitas localidades diferentes, podendo ocorrer falhas humanas.

A CASAN vem já há algum tempo buscando soluções em automação para gerar esse auxílio necessário, porém muitas vezes estas soluções prontas não atendem bem os anseios do nicho de operação da empresa. Um dos desafios aceitos pelo departamento que eu trabalho é de justamente oferecer soluções moldadas para nossos próprios problemas e de maneira que seja facilmente aceita e utilizada pela base operacional.

Outra dificuldade que a empresa passa no momento é a sempre difícil transição entre processos mecanizados para processos automatizados. Esta mudança deve primeiro transpor a barreira pessoal, onde funcionários que há muitos anos fazem o mesmo procedimento da mesma maneira devem se adaptar a uma nova realidade, onde eles terão que aprender novamente como realizar suas funções, ou até mesmo aprender novas funções, sendo que algumas antigas serão alteradas pela automação.

Este talvez seja o maior problema para um projeto de Automação, atender as necessidades de uma empresa moderna sem ferir o ego ou emocional dos funcionários, fazendo com que eles se sintam valorizados pelo trabalho e que ganharam uma ferramenta de auxílio para a realização das atividades e não uma ferramenta que irá substitui-los.

Encontrar ou até mesmo, como no caso deste projeto, criar soluções em automação que auxiliem os processos sem alienar os funcionários é o grande desafio em questão, e o proposto por este trabalho irá auxiliar a atingir este objetivo.

Resumindo, dentro dos nichos de problemas e desafios vividos pela empresa, um dos mais importantes é a Automação, pois é através dela que a CASAN irá caminhar para os objetivos gerais de eficiência, modernização e qualidade de serviços.

2.4: Enquadramento do projeto.

Diante do exposto anteriormente, sabendo da necessidade da CASAN de melhorias no setor operacional, e da complexidade do sistema como um todo, particionar o problema em uma pequena parte e promover uma solução para esta com moldes e flexibilidade de uma solução global, em que o problema de escalamento da solução seja facilmente resolvido, é um objetivo possível de ser alcançado por mim e que talvez motive a empresa a criar outros desenvolvimentos próprios com esta tecnologia empregada.

Nesta concepção de solução *bottom-up*, são os pequenos avanços em soluções para problemas específicos que vão se generalizando com o aumento da aceitação por parte da empresa até atingir processos maiores e sistemas mais complexos.

O objetivo deste trabalho então não é apenas promover um auxílio a uma operação singular do processo, mas também incentivar a pesquisa de outras soluções, construindo um círculo virtuoso de realizações.

Capítulo 3: Fundamentação teórica.

Neste capítulo será formalizado o problema em questão que gerou a solução para este projeto. Formalizando o mesmo através de esquemáticos e descrições que auxiliarão a responder uma pergunta fundamental durante a execução de um projeto de Automação: “Mas qual era o problema mesmo?”.

Um dos grandes erros na execução de projetos não só de Automação, bem como gerais, é conceber uma solução sem se basear nela como foco do projeto, sendo que a procura é pela adequação às necessidades, restringindo-se estritamente a isso. Um projeto subutilizado é tão ruim quanto um projeto equivocado em termos de eficiência.

Definido o problema, também serão apresentadas neste capítulo as técnicas para a solução propostas. Será realizado o embasamento teórico por detrás de cada tecnologia empregada, bem como suas vantagens, desvantagens e adequação ao projeto.

O objetivo proposto é que no final deste capítulo, a base do projeto esteja apresentada de maneira generalista, para que no próximo capítulo a solução específica possa ser estudada.

3.1: Formalização do problema.

Como apresentado anteriormente, Florianópolis tem 99.5% de sua população urbana contemplada com água tratada, enquanto que apenas 51.6% da população tem acesso ao esgoto sanitário. Isto demonstra que o problema sanitário na nossa cidade é mais urgente e requer mais cuidados que o problema de abastecimento. É por este motivo que o trabalho aqui apresentado será destinado à área de saneamento.

Dentro da área de saneamento, o sistema de esgoto de Florianópolis é de uma complexidade impar, sendo composto de dezenas de processos integrantes. O maior complexo de esgoto sanitário de Florianópolis é o complexo ETE Insular, localizado logo após a ponte Governador Ivo Campos, sentido ilha. É neste

complexo que 54 gramas de esgoto para cada cidadão da cidade são tratados por dia, totalizando aproximadamente 23 toneladas diárias (fonte: CASAN, 2013).

O processo de tratamento de esgoto que ocorre na ETE Insular já foi apresentado no capítulo 2, sendo que alguma parte deste já é feita de maneira automatizada.

Um esquemático básico do sistema ETE Insular é mostrado abaixo:

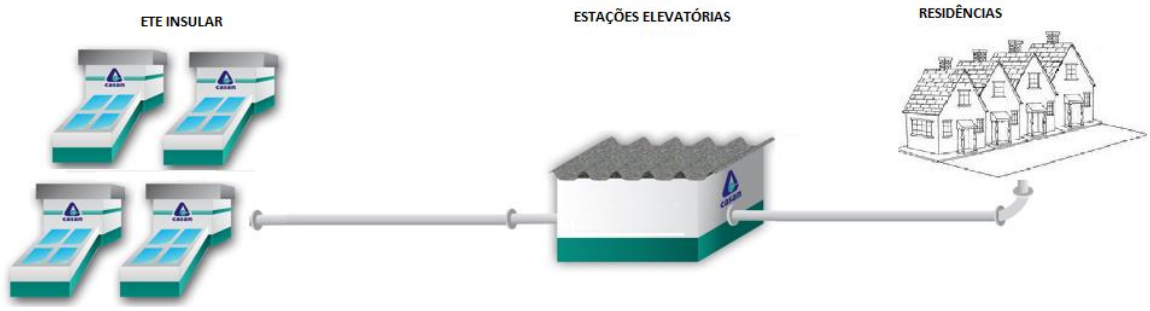

Figura 11- ETE Insular.

O esgoto proveniente de residências, comércios e indústrias espalhadas pelas cidades é enviado através de tubulação para as estações elevatórias que servem como espécie de *buffer* de esgoto, enviando o mesmo para a ETE Insular realizar o tratamento quando o poço úmido atingir um determinado nível.

Com isso, o problema proposto é o de realização de um projeto de automação para o sistema representado na figura 11 de maneira que o mesmo não gere conflito e nem atrapalhe sistemas de automação já presentes no complexo ETE Insular e que concilie os objetivos da CASAN de melhorias na eficiência operacional.

Com o problema geral exposto, é preciso apresentar o que eu denomino de problema foco. Este é a área de atuação deste projeto, e significa a porção do complexo problema da ETE Insular em que posso contribuir. Não serviria para nada neste momento propor um projeto utópico e heroico de concepção individual que resolvesse todos os problemas e automatizasse completamente a ETE Insular, mas seria de grande interesse da CASAN um projeto que auxiliasse o processo e ao

mesmo tempo trouxesse implementações reais de tecnologias até então não utilizadas dentro da empresa.

Como crescimento individual de conhecimentos e experiência, também me é muito mais interessante realizar uma solução de um problema que eu sou capaz e absorver essa experiência para implementações futuras semelhantes em minha carreira.

Observando pessoalmente o sistema como um todo, encontrei dois focos de atuação que precisavam de melhoras e estavam ao meu alcance. São eles: os aeradores da ETE Insular e as Estações Elevatórias de Esgoto que levam o esgoto sanitário até o tratamento.

3.1.1: Problema foco.

A porção do problema que quero atingir com este projeto é concentrada nos aeradores e nas elevatórias de esgoto. Dois problemas que aparentemente estão muito distantes um do outro, mas como pretendo provar neste tópico eles estão mais próximos do que a realidade aparenta inicialmente.

3.1.1.1: Aeradores.

Os aeradores são dispositivos mecânicos que promovem a aeração, ou melhor, promovem o aumento da quantidade de oxigênio em um tanque que armazena esgoto. A aeração é alcançada mecanicamente através de pás inseridas no tanque acionadas por motores. Este oxigênio é responsável pelo desenvolvimento das bactérias aeróbicas que consomem a matéria orgânica em uma reação química que gera calor.

O esquema de um aerador é apresentado na próxima página:

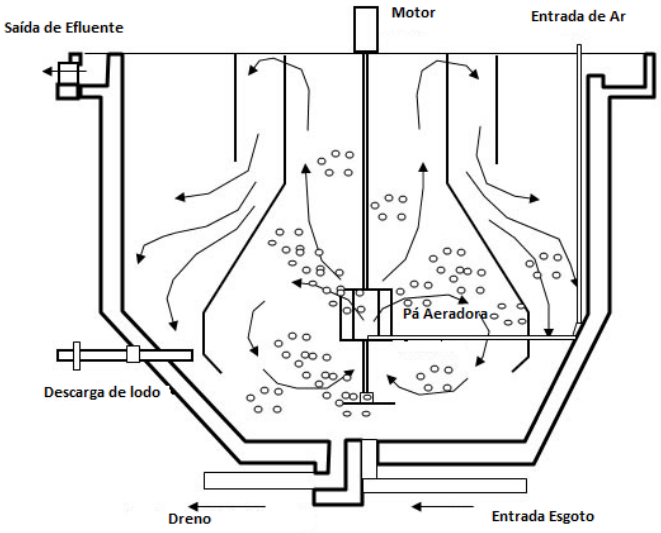

Figura 12 – Aerador.

O tanque possui uma entrada de ar para que o oxigênio possa aumentar devido à agitação mecânica proveniente da pá. Conectada a ela encontra-se o motor, provendo a energia cinética. O esgoto entra pela parte inferior do tanque e depois de completado o processo o efluente sai na parte superior para continuar o processo de tratamento. Ainda existe no tanque uma descarga de lodo que é enviado para o processamento e posterior envio ao aterro sanitário e um dreno de descarga para limpeza e manutenção.

O processo aeróbico feito pelos microrganismos é considerado ótimo quando o nível de oxigênio atinge níveis próximos a 10% do total constituinte do esgoto. Com isso o problema foco pode ser resumido em uma seguinte frase:

“Construção de um sistema automático que controle o nível de oxigênio dentro do tanque em valores da faixa de condição e que esta informação seja facilmente visualizada pelos operadores da ETE Insular. O operador ainda deve ser capaz de alterar o estado do motor bem quanto o funcionamento entre manual e automático quando assim desejar”.

Com isto fica formalizado o problema foco do aerador. As técnicas de sensoriamento, controle e automação para a realização deste objetivo serão discutidas posteriormente.

3.1.1.2: Estações Elevatórias de Esgoto.

As Estações Elevatórias de Esgoto são instalações que abrigam os conjuntos denominados moto-bombas, que podem ou não ter poços de armazenagem de esgoto, denominados poços úmidos. No caso da CASAN, as elevatórias de esgoto possuem poço úmido, responsável pelo armazenamento provisório do esgoto.

Por se tratar de armazenamento de um produto nocivo, é essencialmente importante que o poço úmido não sofra extravasamento, pois a ocorrência do fato gera prejuízos ambientais às comunidades circundantes e também gera multas ambientais contra a CASAN, desprestigiando a empresa perante a população.

O esquema de uma Estação Elevatória de Esgoto pode ser observado na página 23, figura 1.

O controle do nível no poço úmido é feito através do acionamento do conjunto moto-bomba presente na elevatória. Com isso, o problema foco pode ser resumido na seguinte frase:

“Construção de um sistema automático que controle o nível de esgoto no poço úmido para evitar que o mesmo extravase, sendo que estas informações sejam facilmente visualizadas pelos operadores das Elevatórias de Esgoto. O operador ainda deve ser capaz de alterar o estado do motor bem quanto o funcionamento entre manual e automático quando assim desejar.”.

Observando as duas frases propostas acima, chega-se a conclusão que são muito parecidas entre si, mudando apenas o processo envolvido e a variável a ser medida. Com o uso de sensores comerciais que geram saídas de corrente ou tensão conforme a variável medida, os dois problemas se transformam em um só com solução única aplicada em ambos os casos.

3.1.1.3: Diagrama do problema foco.

Depois de apresentados os dois problemas atacados por este projeto, os mesmo podem ser sintetizados em um único diagrama que serve de base para a concepção do projeto, apresentado na próxima página:

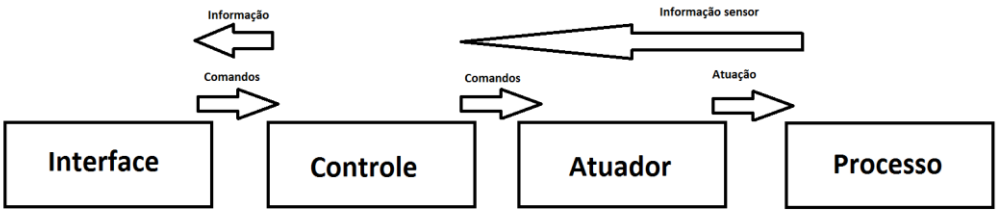

Figura 13- Diagrama do sistema.

Neste esquema, a interface é o canal de comunicação entre o operador e o controle sendo que esta recebe as informações do controlador para apresentar dados e envia comandos para alterar o controle. Este por sua vez, envia comandos para o atuador depois de decidir sua lógica com as informações do sensor proveniente do processo. O atuador altera fisicamente o processo, que reage a esta alteração definindo uma nova operação, que por sua vez é captada pelo sensor e começa um novo ciclo.

Este diagrama é a definição formal do problema foco que será utilizado para alcançar as soluções apresentadas nos próximos capítulos.

3.2: Técnicas para solução.

Definido o problema a ser resolvido, é necessário realizar o embasamento teórico das prováveis soluções, apresentando os conceitos e de que maneira estas soluções atingem os objetivos propostos.

Com isso, se têm cinco grandes áreas de ataque ao problema:

- Interface.
- Controle.
- Envio de dados.
- Atuação.
- Medição.

A área de interface irá se preocupar com a comunicação entre o controle e os operadores, tornando-se um canal de fácil acesso para observação de informações e atuação manual no processo.

A área de controle, por sua vez, irá por sua vez unir o resultado da leitura dos elementos sensores com a ação dos elementos atuadores. É no controle que a lógica de execução é proposta e seguida.

Para o envio de dados, o importante é como os elementos, citados acima, se interligam e trocam informações e comandos. O papel pode ser definido como o elemento de ligação entre dois ou mais blocos distintos que desejam se comunicar, alcançando o objetivo geral do controle.

Na atuação é onde ocorre fisicamente o desejo do controlador, é através dos comandos de controle que o atuador reage, transformando grandezas físicas do sistema.

Por último, a medição é o processo pelo qual se obtêm as informações das grandezas referentes do processo, e é de fundamental importância para que o controlador saiba qual decisão tomar. É através da medição que o controle sabe também se a ação gerada está levando ao resultado requerido.

Nos próximos tópicos será descrito mais extensamente cada uma destas áreas.

3.2.1: Interface.

Uma interface de controle, ou simplesmente interface, é um meio pelo qual uma pessoa operando um determinado equipamento pode instruir o dispositivo ou sistema a realizar algo que assim deseja, de maneira que o item sendo controlado possa responder apropriadamente. A interface tem o objetivo também de facilitar a inspeção visual de variáveis e estados do processo, bem como apresentar mensagens de atenção e emergência.

O desenho de uma interface afeta a quantidade de esforço que um usuário deve prover ao sistema e o esforço necessário para interpretar as informações, bem como a necessidade para aprender a manipulá-lo.

O termo usabilidade, segundo Nielsen[9], em tradução livre, “é o atributo pelo qual se pode medir a facilidade de utilização de interfaces, levando em consideração fatores psicológicos e fisiológicos humanos, transformando a utilização do sistema em algo efetivo e agradável de utilizar”.

Edward Rolf Tufte, um professor de Ciências da Computação da Universidade de Yale, Estados Unidos, foi um dos primeiros a perceber a necessidade da visualização de dados e criou o que provavelmente foi o primeiro sistema em que os usuários interagiam com o computador nos anos 1950, conhecido como IBM Sage System[10].

Este sistema de projeto militar, constituído de 60.000 tubos de vácuo e pesando aproximadamente 250 toneladas foi o maior sistema de computação já projetado e tinha como propósito a defesa do espaço aéreo americano. O projeto continuou seu funcionamento até 1983, quando foi finalmente desativado.

Abaixo se pode observar o que se tornou uma das primeiras interfaces da história:

Figura 14- Primeira Interface Homem-Maquina (Fonte: History of Computing, Lexikon Services).

Com o aumento da popularidade, hoje é comum encontrarmos interfaces nas mais diversas aplicações, como em geladeiras onde é possível alterar a temperatura ou o funcionamento do congelador e em automóveis onde se pode controlar diversas funções do veículo como, por exemplo, temperatura de ar condicionado e procurar rotas para destinos.

As interfaces podem ser funcionalmente e estruturalmente muito simples como teclados ou botões, sendo que sua complexidade pode aumentar conforme as aplicações, como por exemplo, um computador.

Quanto ao tipo, as interfaces possuem diversas classificações, porém três delas são mais importantes dentro do contexto da automação, sendo que uma será utilizada por este projeto. São elas:

- *Graphical User Interface* (GUI): são interfaces que aceitam entradas via teclados e *mouse* e geram saídas em telas de computadores.
- Interface baseada em web: podem prover controles em tempo real utilizando programas separados em localidades diferentes. A interface entre controlador e controle é feita baseada no envio de páginas tipo web.
- Telas de toque: São dispositivos que aceitam entradas via toque de dedos ou canetas especiais. Com o aumento da popularidade de dispositivos móveis utilizando esta tecnologia, hoje é comum encontrarmos interfaces deste tipo em nossos lares.

Analisando mais profundamente as telas de toque, uma interface de caráter industrial que chama a atenção é a interface comumente chamada de IHM (Interface Homem-Máquina). Esta interface tem uma gama de aplicação muito grande na área de automação e por se constituir de uma interface totalmente programável provém uma flexibilidade desejável neste projeto. Mais sobre a IHM é apresentado abaixo.

3.2.1.1: Interface Homem-Máquina.

Nos processos industriais a informação do nível de planejamento é levada para o nível de controle através da rede de comunicação de dados, possibilitando, assim, uma integração entre estes dois níveis e estabelecendo uma rede local.

As Interfaces Homem-Máquina, ou simplesmente IHM, representam uma forma de conferir mais produtividade e flexibilidade aos sistemas informatizados. As interfaces de tela gráfica e colorida dão ao sistema melhores condições para o controle e supervisão do número de variáveis existentes.

As IHMs são disponíveis em duas opções, a saber, *hardware* específico e *software* executado em computador.

O *hardware* é um dispositivo totalmente integrado ao controlador adotado e normalmente adapta-se bem ao ambiente industrial, porém só pode ser utilizado com os equipamentos disponíveis pelo fabricante eleito. Os *softwares* permitem que projetista tenha várias opções de comunicação com quaisquer marcas e modelos disponíveis no mercado. É a análise do custo X benefício e as exigências de cada sistema que determinarão o uso de uma ou de outra forma de IHM.

Interfaces são sistemas normalmente utilizados em automação de plantas industriais, as quais em geral caracterizam-se por ter um ambiente agressivo. A construção destes sistemas é extremamente robusta e resistente a jatos de água direto, umidade, temperatura e poeira de acordo com o IP (grau de proteção) necessário.

Além das suas características padrão, algumas IHMs modernas também apresentam capacidade para um gerenciamento de uma maior quantidade de variáveis.

As IHMs são aplicáveis em vários casos, desde simples máquinas de lavar pratos até cabines das aeronaves e helicópteros. Naturalmente, neste último caso as IHMs são extremamente especializadas para atender à função a que se destinam.

Ao contrário do que temos hoje, a integração do homem com o processo industrial no passado se dava através de um grande painel sinótico que utilizava anunciadores de alarmes, sinaleiros, chaves seletor e botoeiras que permitiam comandar ou visualizar apenas alguns estados pré-definidos, como ligado/ desligado, alto/baixo ou temperatura elevada/normal.

O desenvolvimento tecnológico que possibilitou o uso de visores alfanuméricos, teclados de funções e comunicação serial fez as IHMs como as conhecemos hoje, apresentando diversos benefícios, tais como a facilidade de programação e manutenção, a alteração de parâmetros do processo, maior flexibilidade frente às alterações necessárias no campo, além de uma operação totalmente amigável. Uma foto de um exemplo de uma IHM da marca SALFATIS é apresentada na próxima página:

Figura 15- IHM.

A IHM acima é composta por uma tela de cristal líquido de dimensões variáveis e uma série de teclas que podem ser utilizadas para navegação ou inserção de dados, programáveis via *software* proprietário (se escolhida a versão *software*).

Entre as diversas aplicações da IHM podemos citar:

- Visualização de alarmes em caso de condições anormais do sistema.
- Visualização de dados de equipamento de uma linha de produção.
- Visualização de dados de processo da máquina.
- Alteração de parâmetros do processo.
- Operação em modo manual de componentes da máquina.
- Alteração de configurações de equipamentos.

Através da visualização gráfica em cores e com alta definição o operador pode obter informações precisas a respeito do status do processo de forma muito mais prática e rápida.

Neste tipo de visualização, informações por cores e textos são muito usadas e é possível, inclusive, utilizar elementos graficamente animados. Desta forma, evidentemente, o operador terá uma melhor visualização quando for necessário obter informações do processo (o abrir de uma válvula ou o ligamento de um motor, por exemplo).

Na elaboração de uma interface gráfica, o projetista pode usar a representação de painéis sinóticos, ou que expressem um fluxo de produção, ou o *layout* dos equipamentos da planta, ou a sequência de controle, ou ainda outra

organização gráfico-lógica que expresse adequadamente a planta a ser supervisionada.

A comunicação entre a IHM e o controlador usualmente se dá por meio de um protocolo de comunicação específico que reproduz as variáveis do processo (*Tags*) na IHM. Uma *Tag* é, portanto, uma variável do tipo discreta, numérica ou alfanumérica, que, por ser bidirecional no sentido da comunicação entre o controlador e IHM, monitora o status do controlador e envia valores a ele.

As IHMs permitem armazenar conjuntos de *setpoints* gravados em arquivo que definem as diversas parametrizações do sistema, ou receitas de produção quando são enviados ao controlador. Assim, com um simples envio de comando, é possível, por exemplo, alterar rapidamente a produção de um determinado tipo de peça para outro tipo.

Quanto à alarmes, se desejado, as IHMs têm a capacidade de, através de sinais de alarme, isentar a CPU do controlador de monitorar situações anômalas do processo.

As *Tags* são constantemente monitoradas e a IHM pode alertar o operador sobre a troca de situação do processo e até inclusive apresentar uma sugestão de ação diante do defeito ocorrido.

Além destes recursos, as IHMs, em geral, também permitem o registro histórico (através do armazenamento dos dados) de eventos e alarmes, o que, posteriormente permite a análise de ocorrências para fins de controle estatístico, análise histórica para consultas, plotagens e outros tipos de relatório e gráficos de tendência.

Com todos estes pontos citados, é evidente que esta é uma ferramenta muito importante e poderosa em projetos de automação e um dos maiores engrandecimentos pessoais com a realização deste trabalho foi justamente ter contato com esta tecnologia e programá-las para atender as necessidades conforme as especificações do problema.

3.2.2: Controle.

Controle é um ramo interdisciplinar de matemática e engenharia que lida com comportamentos de sistemas dinâmicos com entradas e saídas. Quando uma ou mais variáveis de saída precisam seguir alguma referência o controle manipula as entradas do sistema para alcançar o efeito desejado na saída do sistema.

O objetivo genérico do controle é calcular soluções que gerem a ação corretiva apropriada que resultem em estabilidade do sistema, ou seja, que o sistema chegue ao seu ponto de equilíbrio. Os sistemas de controle podem ser analisados como tendo quatro funções: Medir, comparar, computar e agir.

Embora as técnicas de controle sejam utilizadas desde a antiguidade, uma análise mais formal do campo começou com a apresentação de um controle primitivo denominado *Centrifugal Governor*, apresentado por James Maxwell em 1868[11].

O *Centrifugal Governor* era uma espécie de dispositivo capaz de regular a velocidade de antigos motores à vapor controlando a quantidade de vapor que entrava nos mesmos, mantendo a mesma velocidade independentemente da carga a qual o mesmo era aplicado, utilizando os princípios básicos de um controlador proporcional.

Na mesma época, um dos colegas de Maxwell, Edward John Routh, foi capaz de abstrair os resultados do controlador *Governor* e com isso criou a classe de processos denominados sistemas lineares. Avanços na área foram feitos desde então, juntamente com o crescimento industrial e de tecnologias.

Quanto a sua classificação, os sistemas de controle podem ser divididos em controles de malha aberta e controles de malha fechada. Mais sobre estes controles será discutido abaixo.

3.2.3: Controle de malha aberta.

O sistema de controle de malha aberta é um tipo de controle contínuo no qual a saída não tem nenhuma influência ou efeito na ação de controle sobre o sinal de entrada, ou seja, em um controle de malha aberta, a saída não é nem medida e nem

utilizada para comparação com a entrada. Portanto, se espera que este tipo de controle siga fielmente os comandos de entrada ou *setpoint*, desconsiderando o resultado final.

Os sistemas de controle de malha aberta são utilizados quando os processos são relativamente simples pela sua facilidade e baixo custo de instalação e operação, especialmente em sistemas onde a realimentação não é crítica. Exemplos de sistemas de controle em malha aberta é a torneira de um chuveiro elétrico e a operação de um fogão residencial.

Uma desvantagem é que, pela falta de conhecimento das condições da saída, o sistema não tem capacidade para corrigir algum erro que ocorra quando o sistema flutua para outro ponto de operação, mesmo que este seja muito distante da referência.

Mais uma desvantagem deste tipo de controle é que os mesmos estão pobremente equipados para lidar com perturbações ou mudanças nas condições as quais podem reduzir sua habilidade em completar a tarefa desejada. Estes erros podem causar distúrbios nos processos e por isso necessitam supervisão extra dos operadores.

Um sistema de controle de malha aberta pode ser representado por uma cascata de blocos em série representada abaixo:

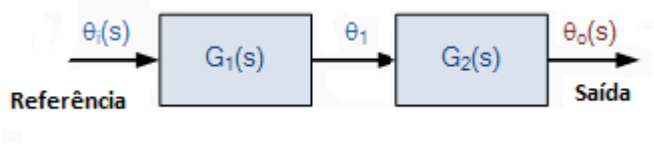

Figura 16- Diagrama de controle de malha aberta.

A função de transferência de cada bloco é:

$$G1 = \frac{\theta_1}{\theta_i} \ , G2 = \frac{\theta_O}{\theta_1}$$

A função de transferência global é:

$$FT = G1 \ x \ G2 = \frac{\theta_O}{\theta_i}$$

Com ganho:

$$Ganho = \frac{\theta_O}{\theta_i}$$

Com isso, pode-se resumir um controle de malha aberta como tendo as características abaixo:

1. Não há comparação entre valores atuais e desejados.
2. Não há autorregularão ou ação de controle sobre o valor da saída.
3. Cada entrada determina um ponto de operação fixo do controlador.
4. Mudanças ou perturbações não alteram o comportamento do controlador.

3.2.3.1: Controle de malha fechada.

O controle de malha fechada, também conhecido como controle de realimentação, é um sistema que utiliza os conceitos de malha aberta para seu caminho direto, mas contém uma ou mais voltas para comparar a saída atual com a saída desejada. A medição da saída é denominada sinal de realimentação e é ela que define a ação de controle.

Estes tipos de controladores são projetados para alcançar automaticamente uma condição de saída idêntica à desejada e mantê-la nesta condição através da realimentação. Para fazer isso, o controlador gera um sinal de erro o qual é a diferença entre a saída e a referência de entrada.

Devido ao fato do controle de malha fechada ter conhecimento das condições de saída, este está mais bem equipado para lidar quaisquer perturbações ou mudanças nas condições as quais possam alterar o sistema a ser controlado.

O termo malha fechada sempre implica no uso da realimentação para reduzir erros no sistema de controle, e é esta a grande diferença entre malha aberta e fechada. A precisão da saída depende da realimentação, sendo que atualmente pode-se realizar esta medição com uma precisão muito alta, sendo o controle de malha fechada a decisão de operação na maioria dos casos.

Um sistema de controle de malha fechada pode ser representado por o seguinte diagrama de blocos:

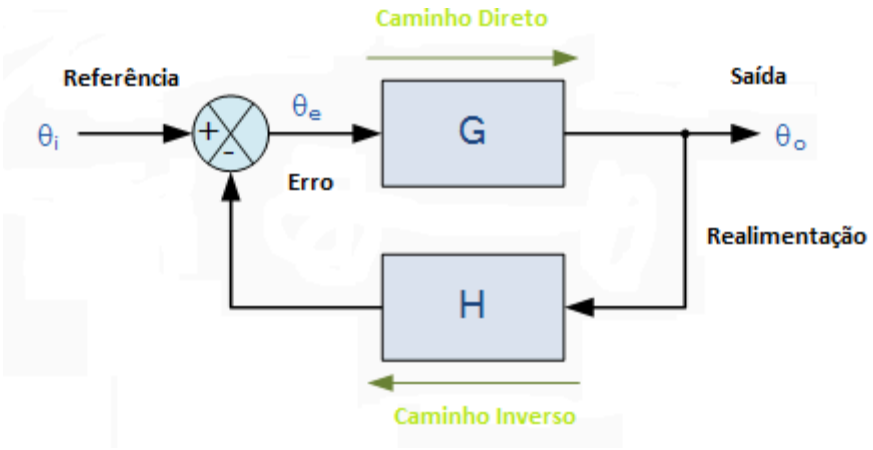

Figura 17- Diagrama de controle de malha fechada.

Onde G representa o ganho de malha aberta do controlador e é representado pelo caminho direto, e o bloco H representa o ganho da realimentação, representado pelo caminho inverso.

O bloco de diagramas acima tem a seguinte função de transferência global:

$$FT = \frac{G}{1 + GH}$$

A função acima apresenta o sinal positivo no denominador da fração quando a realimentação é do tipo negativa. Quando a realimentação for positiva, o denominador se altera para $1 - GH$.

Podemos observar que quando $H \to 1$ e o ganho G é grande, a expressão acima se aproxima para:

$$FT = \frac{G}{1 + GH} \to \frac{G}{1 + G} \to 1$$

Ou seja, a saída do sistema será idêntica a referência quando o sistema entrar em regime permanente.

Com isso, pode-se resumir um controle de malha aberta como tendo as seguintes características abaixo:

1. Reduz erros automaticamente ajustando a entrada do sistema.
2. Aumenta estabilidade de sistemas instáveis em malha aberta.

3. Reduz a sensibilidade do sistema a mudanças.
4. Aumenta a robustez do sistema perante perturbações.
5. Produz desempenhos confiáveis e previsíveis.

Ainda dentro dos controladores considerados de malha fechada, existem várias técnicas diferentes com as configurações de controladores mais diversas para atingir as metas do controle de malha fechada. Entre elas, as mais comuns são:

- Controle liga-desliga (on-off): Controlador que age de maneira binária através do atuador, ligando o estado ou desligando. Estes controladores são utilizados quando uma variação contínua da saída dentro de uma faixa de operação é aceita.
- Controle proporcional: Controlador que age sobre o ganho de malha fechada da planta, aumentando quando o erro é muito grande e reduzindo quando a saída vai se aproximando da referência.
- Controle proporcional integral: Semelhante ao controle proporcional, porém com o acréscimo de uma parcela integral, a qual força a saída a ter o mesmo valor da referência, chegando-se a erro zero quando em regime permanente.
- Controle proporcional integral derivativo: tem a mesma característica de um controle proporcional integral porém com a parcela derivativa pode-se alterar comportamentos no regime transitório do sistema, alterando características de curvas.

Olhando mais especificamente para o controle liga-desliga este, também denominado de controlador de histerese, é um controlador de malha fechada que altera entre dois estados abruptamente. Estes tipos de controladores são geralmente utilizados em plantas que aceitam entradas binárias, por exemplo, o abrir ou fechar de uma válvula. O controle liga-desliga geralmente é referenciado em problemas de tempo mínimo. Um exemplo é: a menor distância para um carro chegar a um ponto final é acelerar ao máximo até chegar à posição final e freia-lo imediatamente. Claro que este é um caso hipotético onde se desconsidera a inércia e o desconforto dos passageiros, mas ilustra o comportamento do controle.

Segundo Ballard[12], o controle do tipo liga-desliga apenas deve ser aplicado quando apenas as condições abaixo estão presentes:

- Controles precisos não são necessários devido ao fato que a variável medida fica constantemente varia ciclicamente;
- O tempo morto do processo é tão grande que não causa desgaste de peças pela troca constante de posição;
- A diferença entre o tempo morto e a constante de tempo do processo deve ser pequena para evitar que a amplitude da variável medida se altere muito durante o ciclo de medição.

Para implementar sistemas de controle, são utilizados dispositivos como CLPs (Controlador Lógico Programável), microprocessadores ou microcontroladores. Mais sobre estes dispositivos é apresentado no próximo tópico.

3.2.3.2: Controlador Lógico Programável.

Os Controladores Lógicos Programáveis são computadores digitais muito utilizados em projetos de controle e automação. Ao contrário de computadores domésticos, os CLPs são desenhados para múltiplas entradas e saídas, além de serem protegidos fisicamente contra altas temperaturas, ruídos elétricos, vibrações e impactos. Programas de controle são geralmente armazenados em memórias protegidas por baterias de emergência, evitando perda de programação com falta de energia elétrica.

Antes da concepção dos CLPs, a maioria das sequências lógicas e de travamento e controle eram basicamente constituídos por relés e temporizadores conectados a controladores dedicados para cada aplicação. Com o aumento da complexidade das atividades industriais, este método se provou cada vez mais ineficaz.

Com isso, em 1968, a divisão de transmissão automática da General Motors, Estados Unidos, fez a requisição de um dispositivo eletrônico que substituísse os sistemas à base de relés utilizados nas fábricas da GM. A proposta vencedora veio da Bedford Associates, da cidade de Bedford, Massachusetts, a qual instalou o primeiro modelo de CLP conhecido na época. Desde então, a indústria

automobilística continua sendo um dos maiores consumidores de CLPs do ramo industrial.

Um CLP é constituído basicamente de:

- Uma fonte de alimentação.
- Uma Unidade Central de Processamento (UCP).
- Memória do tipo fixa (memória de programa) e volátil (apagável, para dados).
- Dispositivos de entrada e saída.
- Terminal de programação.

Quanto à programação, a maioria dos programas para CLPs são escritos em aplicações especiais e então passados para o equipamento sendo que até hoje a maioria dos CLPs ainda são programados em *Ladder*, uma linguagem de programação com baseada em diagramas de circuitos.

Os programas então são rodados repetidamente automaticamente enquanto o sistema de controle estiver sendo utilizado. Os estados das entradas físicas são copiados para uma área de memória acessível pelo processador, no que é denominado de tabela de I/O. Este programa então acessa os dados e executa a leitura das instruções programadas, tomando as devidas decisões sobre as portas de saída. Estas decisões são então transformadas em entradas da tabela de I/O que por sua vez é acessada pelas saídas físicas do controlador que alteram seu estado para a nova configuração.

Geralmente, controladores deste tipo não possuem interfaces gráficas com os usuários, sendo necessária a interligação de periféricos como telas, botões, ou ate mesmo IHM para cobrir esta lacuna.

Entre os principais fabricantes de CLPs atualmente no mercado pode-se destacar a Samsung, Bradley e Siemens, o qual é apresentado uma imagem na próxima página do modelo S7-200:

Figura 18- CLP Siemens S7-200.

Na figura é visível o barramento de I/O do CLP, além de duas portas de comunicação com conectores db9 para conexão com outros equipamentos, inclusive IHMs.

Entre as principais aplicações de automação para CLPs pode-se citar:

- Indústria automobilística.
- Indústria siderúrgica.
- Indústria de papel e celulose.

A solução com uso de CLPs ainda é a mais aceita no ramo industrial quando se trata de controle e automação, mas em muitos casos a mesma é subutilizada, podendo fornecer muito mais do que é exigida, e nestes casos as soluções com microcontroladores podem ser mais efetivas.

3.2.3.3: Microprocessadores

Microprocessadores são unidades capazes de incorporar as funções de um UCP de um computador em um único circuito integrado. Os microprocessadores são capazes de receber dados digitais como entrada, os quais são processados de acordo com instruções pré-programadas em sua memória, provendo saídas em suas portas como resultados.

Durante a década de 1960, processadores de computadores eram construídos em circuitos integrados de média escala, cada um com centenas de transistores. Estes então eram soldados em placas de circuitos impressos e muitas dessas placas por sua vez eram interconectadas em um chassi. O grande número

destas placas causava sobreaquecimento com computador e limitavam o seu desempenho.

Foi então que na NASA, a agência espacial norte americana, apareceram as primeiras implementações de microprocessadores os quais em um único circuito integrado obtinham o mesmo desempenho que os grandes chassis citados anteriormente. Uma dos primeiros projetos deste tipo se chamava Apollo Guidance Computer, que auxiliou as missões Apollo á chegar ao espaço sideral.

Foi então que os adventos de computadores de baixo custo em circuitos integrados transformaram a sociedade moderna. Microprocessadores de propósitos gerais em computadores pessoais são utilizados para edição de textos, mostradores multimídias e comunicação através da internet.

Milhares de itens que até no passado não eram relacionados com computação agora estão embarcados com microprocessadores, entre eles: Chaves de carros, brinquedos, alarmes de incêndio, reprodutores de mídia e telefones.

Para se ter uma ideia de importância, em 2003 o mercado movimentado pela fabricação e venda de microprocessadores gerou uma renda bruta de US$ 44 bilhões em 2003 (fonte: IBM, 2003).

Quanto a sua estrutura, a disposição interna de um microprocessador varia dependendo da idade do mesmo e dos fins pretendidos do processador. A complexidade de um circuito integrado é limitada pelas limitações físicas do número de transistores que podem ser colocados em um único chip, o número de ligações entre o processador a outras partes do sistema, o número de interligações que é possível fazer sobre o chip e o calor que o chip pode dissipar. Avanços da tecnologia produzem chips cada vez mais complexos e poderosos.

Um microprocessador hipotético mínimo pode incluir apenas uma unidade aritmética e lógica (ALU), e uma secção lógica de controle. A ALU executa operações como adição, subtração e operações como AND ou OR. Cada operação da ALU define um ou mais *flags* em um registrador de status, que indicam os resultados da última operação (valor zero, número negativo, estouro ou outros). A secção lógica recupera códigos de operação de instrução de memória, e inicia a sequência de qualquer das operações que a ALU requere para executar a instrução.

Um código de uma única operação pode afetar muitos caminhos individuais de dados, registros e outros elementos do processador.

Com o avanço dos estudos na área, foi possível fabricar cada vez processadores mais complexos. O tamanho da estrutura tornou-se menor, permitindo que mais transistores sejam instalados em um chip e que por sua vez permitiu tamanhos de palavras de 4 bits até os modernos 64 bits. Vários recursos adicionais foram adicionados à arquitetura do processador, como maior número de registros, os quais aceleraram programas e instruções complexas.

Com a capacidade de colocar um grande número de transistores em um único chip, torna-se viável para integrar memória no mesmo chip que o processador. Esta cache tem a vantagem de um acesso mais rápido do que a memória fora do chip, e aumenta a velocidade de processamento do sistema para muitas aplicações. Geralmente, a velocidade do processador aumentou mais rapidamente do que a velocidade da memória externa, assim memória cache é necessária para acompanhar a velocidade de processamento.

Na automação o uso de microprocessadores é bastante difundida como unidade lógica em soluções completas. Como inerentemente é apenas uma unidade capaz de realizar processamento, sem interligação direta com o processo, é utilizado como agente tomador de decisão que se comunica com os periféricos do sistema recebendo e enviando comandos, emulando um computador pessoal.

Exemplos de microprocessadores são os famosos Intel Core i3, i5, e i7, além de outros de propósitos mais gerais como o P8085, mostrado na imagem abaixo:

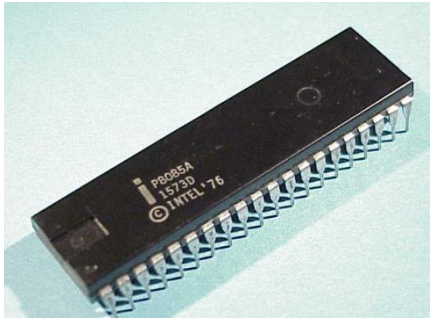

Figura 19- P8085 (Fonte: IBM)

3.2.3.4: Microcontroladores

Um microcontrolador é um computador de pequeno porte em um único circuito integrado contendo uma unidade de processamento, uma memória e periféricos programáveis de entrada e saída. Geralmente ainda são inclusos uma pequena memória RAM (Memória de Acesso Randômico) e uma memória EEPROM (memória não-volátil) ou Flash programável. Microcontroladores são desenhados para aplicações embarcadas, ao contrário de microprocessadores que geralmente são utilizados em aplicações de uso geral, como computadores. Na automação, microcontroladores são utilizados para o controle automático de processos, como controles de motores, maquinários, ferramentas, entre outros.

Alguns microcontroladores, desenhados para ter baixos consumos, podem trabalhar em frequências de relógio tão baixas quanto 4 kHz e utilizar palavras de apenas 4 bits, enquanto que outros podem atuar como processadores em tempo real, onde sua frequência pode chegar a frequências de mHz e 64 bits. Isso mostra a ampla gama de operações em que os microcontroladores estão envolvidos.

Quanto ao seu desenho, a maioria dos microcontroladores em uso nos dias atuais está embarcada em outros equipamentos, como automóveis, celulares e periféricos de computadores. Embora alguns sistemas embarcados sejam bastante sofisticados, a maioria dos sistemas tem limites mínimos de memória e desempenho, sem nenhum sistema operacional e baixa complexidade de *software*. Isso exige que os microcontroladores sejam flexíveis para se adaptarem a ambas as situações.

Microcontroladores devem prover respostas imediatas a eventos no sistema que estão controlando. Quando certo evento ocorre, um dispositivo de interrupção sinaliza ao processador para suspender as atividades e iniciar a Rotina de Serviço de Interrupção (RSI), comumente chamado de tratador de interrupção. O RSI vai realizar todo processamento necessário baseado na fonte da interrupção antes de retornar a sequência normal de programação. Fontes de interrupções podem incluir eventos como conversão analógica digital, mudanças em uma entrada ou dados recebidos em um canal de comunicação. Quando consumo de energia é um fator

importante, interrupções também podem causar a mudança de estado do controlador entre desativado e ativado.

Microcontroladores geralmente possuem alguns pinos de entradas/saídas de propósitos gerais. Esses pinos geralmente podem ser configurados para tanto entrada como saída. Quando configurados como entradas, geralmente estes pinos servem para leitura de dados de sensores ou sinais externos. Quando configurados como saídas, os pinos podem ser conectados a dispositivos de acionamento ou motores, por exemplo.

Muitos sistemas de automação necessitam ler dados de sensores que enviam sinais analógicos, e é para este propósito que existe nos microcontroladores o ADC(sigla em Inglês), Conversor Analógico Digital. Como estes processadores estão projetados para interpretar e processar dados digitais, os mesmos não teriam capacidade para interpretar os dados analógicos. Nestes casos, o conversor analógico digital é utilizado para converter os dados de entrada no formato possível de ser utilizado pelo controlador.

Ainda adicionado aos conversores, muitos microcontroladores possuem uma variedade de relógios. Dentre estes relógios, o tipo mais comum é o Relógio de Intervalo Programável (PIT em Inglês). O PIT pode contar regressivamente de um valor até zero, ou progressivamente até a capacidade de um registrador e indicando zero. Uma vez que alcança o zero, o relógio envia uma interrupção para o processador, indicando que o mesmo acabou de contar. Esta característica é importante quando se necessita registrar dados periodicamente, como em sensores de temperatura.

Um bloco de Modulação por Largura de Pulso, ou PWM em Inglês, torna possível controlar motores de passo ou cargas resistivas em tensão alternada sem estrangular a unidade central de processamento com esforço computacional.

Finalmente, um bloco de comunicação UART (*Universal Asynchronous receiver/transmitter)* torna possível receber e transmitir dados serialmente sem muito esforço computacional.

Com isso, se podem citar as características principais de um microcontrolador como sendo:

- CPU com processadores de arquitetura de 4 bits até 64 bits.
- Memória RAM para armazenamento de dados.
- Memória Flash para armazenamento de programação.
- Bits exclusivos para pinos de saída e entrada, permitindo controle e lógica de pinos individuais.
- Portas de entrada/saída seriais (UARTs).
- Periféricos como relógios, contadores, geradores de onda PWM e outros.
- Conversor analógico digital.
- Programáveis.

Dentre os diversos tipos de microcontroladores, podemos citar alguns como: Amtel AVR, Intel 8051, MIPS, Freescale Coldifre, entre outros.

Um em especial para este trabalho é o Amtel AVR. Este é um microcontrolador de arquitetura de 8 bits desenvolvido pela companhia Amtel e foi o primeiro microcontrolador de escala comercial a utilizar memória Flash para armazenamento de programas.

Dentre os diversos microcontroladores da família AVR, a série ATmega possui 32 KB de memória Flash e 28 a 100 pinos por microcontrolador. O Arduino, uma implementação em placa única de um controlador, utiliza o AVR ATmega em sua concepção e agrega diversas qualidades que podem ser úteis em um sistema de automação. Mais sobre o Arduino é apresentado a seguir.

3.2.3.5: Arduino.

O ambiente Arduino foi projetado para ser fácil de usar para iniciantes que não tem nenhuma ou pouca experiência com *software*. Com o Arduino, é possível construir objetos que podem responder e/ou controlar luzes, som, toque e movimento. O Arduino já foi usado para criar uma incrível variedade de coisas, incluindo instrumentos musicais, robôs de luz esculturas, móveis, jogos interativos e até mesmo roupas interativas.

O Arduino é usado em muitos programas educacionais em todo o mundo, especialmente por designers e artistas que desejam criar facilmente protótipos, mas

não precisam de uma compreensão profunda dos detalhes técnicos por trás de suas criações. Por ser projetado para ser usado por pessoas não técnicas, o *software* inclui uma abundância de exemplo de código para demonstrar como utilizar as diversas funcionalidades da placa Arduino.

Embora seja fácil de usar, o *hardware* do Arduino funciona no mesmo nível de sofisticação que os engenheiros empregam para a construção de dispositivos embarcados. Pessoas que já trabalham com microcontroladores também são atraídos para Arduino por causa de seu desenvolvimento ágil e sua facilidade de implementação rápida de ideias.

O Arduino é mais conhecido por seu *hardware*, mas ele também precisa de um *software* que programe o *hardware*. Tanto o *hardware* como o software são chamados de "Arduino". A combinação permite criar projetos nesse sentido e controlar o mundo físico. O *software* é gratuito, *open source* e multi-plataforma. As placas são baratas para comprar, ou podem inclusive ser construídas pelo projetista (os designs de *hardware* também são *open source*). Além disso, existe uma comunidade Arduino ativa e solidária que é mundial e acessível através dos fóruns Arduino e wiki (conhecido como Arduino *Playground*).

O projeto Arduino começou como um projeto de estudantes da Universidade de Ivrea, em Turim, Itália. Com o reconhecimento do projeto com uma menção honrosa no concurso Prixs Ars Electronica, realizado na Áustria, as primeiras versões foram comercializadas. O crescimento e a aceitação do projeto foi aumentando, sendo que em 2011 cerca de 300 mil unidades já haviam sido vendidas[13].

Atualmente existem 17 modelos de placas oficiais de Arduino, de dezenas de fabricantes, sendo que os mais comuns são o Arduino Uno (por ser o mais econômico), o Arduino Mega (por ter 100 pinos, sendo o maior deles) e o Arduino Leonardo (por utilizar um microcontrolador de 32 bits).

Por ser um projeto livre, muitas alterações e periféricos para conexão com o Arduino podem ser realizados, inclusive neste projeto será apresentado uma solução caseira para redes de comunicação utilizando o Arduino.

Uma imagem do Arduino Uno é apresentada na próxima página:

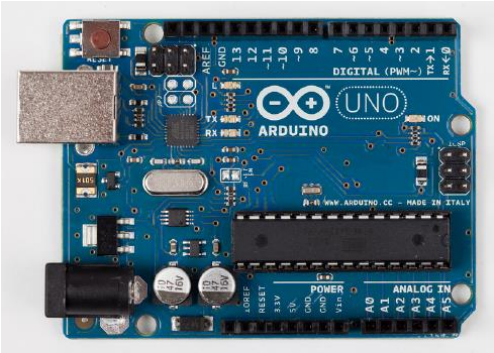

Figura 20- Arduino UNO.

Pode-se observar que o Arduino tem uma concepção que convida o usuário iniciante a experimentar. Cada pino do microcontrolador tem um auxiliar de conector para fios de rápida conexão, evitando a necessidade de soldas e contatos de fácil acesso para fonte de energia e cabo de dados.

Falando um pouco mais especificamente sobre o Arduino UNO, apresenta-se a tabela de características técnicas:

Microcontrolador	Atmega328
Tensão de operação	5V
Tensão de alimentação	7-12V
Limite tensão de alimentação	6-20V
Pinos I/O Digital	14 (6 com saídas PWM)
Pinos entrada analógica	6
Corrente máxima em pinos I/O	40 mA
Corrente máxima em pinos 3.3 V	50 mA
Memória Flash	32 KB
SRAM	2KB
EEPROM	1 KB
Velocidade de relógio	16 MHz

Tabela 1- Características técnicas do Arduino UNO.

Uma das características marcantes do Arduino Uno que fizeram com que algumas partes desse projeto fossem mais dificultosas é pelo fato de sua comunicação serial ser do tipo TTL (lógica de transistor transistor).

A maioria dos microcontroladores da atualidade transmite através de suas UARTs um bit de cada vez em uma taxa pré-determinada, sendo que os limites de tensão sempre ficarão entre 0V e Vcc, onde Vcc é a tensão de operação. Logo, nestes casos, a lógica 1 tem uma valor entre 5V e 3.3V para o Arduino e a lógica 0 tem valores entre 0V e 3.3V. O problema fica evidente quando existe a necessidade de se comunicar com um dispositivo através de redes RS-232 e redes RS-485, onde os sinais lógicos 0 tem tensões negativas. Mais sobre estes conceitos e a solução encontrada por este projeto será discutido nos próximos capítulos.

Outra limitação exclusiva do Arduino Uno é ter apenas um canal de comunicação serial, fato que aparentemente limita o número de dispositivos conectados a ele. Este projeto utilizou de algumas artimanhas para contornar esta situação, sendo que mais sobre este fato será decorrido posteriormente.

3.2.4: Envio de dados.

O envio ou transmissão de dados é o processo onde ocorre a transferência física de um dado entre componentes diferentes. Esta transmissão pode ser realizada ponto-a-ponto, onde a comunicação se faz unitariamente entre um par de elementos de cada vez, ou por difusão, onde um elemento pode transmitir seu conteúdo para diversos receptores.

Exemplos de canais de comunicação são fios trançados, fibras óticas, transmissão sem fio e dispositivos de armazenamentos, como CDs ou DVDs. Os dados são representados como sinais magnéticos tais quais: tensão elétrica, ondas de rádio, micro-ondas, ondas infravermelho, entre outros.

Enquanto que a transmissão analógica é a transferência de um sinal analógico contínuo variável (como um sensor, por exemplo), transmissão digital é a transferência de mensagens discretas. Estas mensagens são representadas ou por uma sequência de pulsos gerada por um método de codificação de linha ou por um

conjunto limitado de ondas variáveis através de uma modulação (transmissão de banda passante).

3.2.4.1: Codificação de linha.

Na codificação de linha, o objetivo é representar um sinal digital através de uma amplitude variável de corrente ou tensão. O formato da onda é utilizado para representar os 0 e 1 de um dado digital de maneira que a variação destas formas indicam a sequência de bits do dado.

Entre as diversas formas de formatação da onda podem ser citados os métodos:

- Método Unipolar.
- Método Polar.

Em um método unipolar, os dados são representados em sinais de apenas polo positivo ou negativo, por exemplo, um bit 1 será representado por uma tensão Vcc e um bit zero será representado pela tensão zero. Quando configurado desta maneira, o método é denominado NRZ (*Non-Return-to-Zero),* ou em português, Sem Retorno a Zero. O nome vem do fato de que o sinal configurado não altera seu estado no meio de um bit, ou seja, não retorna à tensão zero. A imagem abaixo mostra a codificação de um sinal digital com o método NRZ:

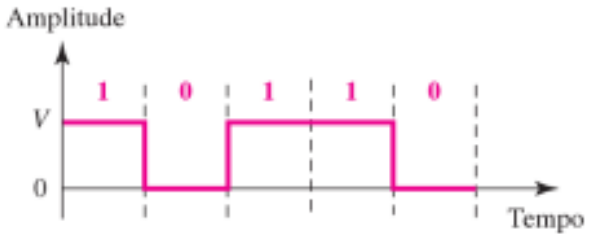

Figura 21- Método NRZ.

Na figura acima, o dado a ser transmitido é representado por uma sequência de bits 10110, que é codificada para uma onda pelo método NRZ para seu formato de transmissão. A leitura pelo receptor é feita através de períodos cíclicos de tempo, onde a cada estouro deste período, uma leitura de tensão é realizada para descobrir se foi transmitido um dado 1 ou 0.

Em um método polar, as tensões são tanto positivas como negativas na mesma onda de sinal, por exemplo, o nível de tensão para o sinal 1 pode ser +Vcc e para o sinal 0 pode ser –Vcc. Entre as variantes dos métodos polares se encontram o NRZ-L (*level)* e o NRZ-I (*Invert)*. No NRZ-L, o nível da tensão determina o valor do bit a ser transmitido, já no NRZ-I, a mudança ou falta de mudança na tensão determina o valor do bit a ser transmitido, sendo que este método não inverte a tensão do sinal de saída quando envia um bit 0 e modifica a tensão quando envia um bit 1. A imagem abaixo ajuda a entender melhor estes dois métodos:

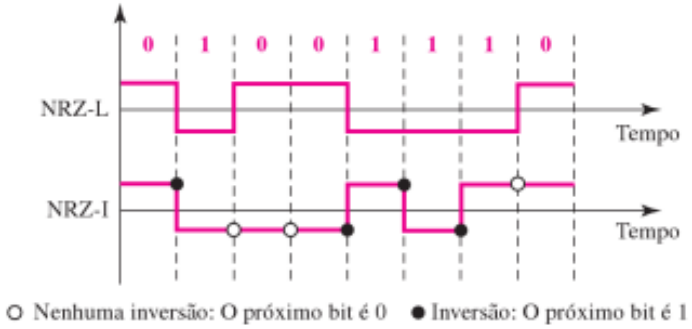

Figura 22- Métodos NRZ-L e NRZ-I.

Na figura acima, o dado a ser transmitido é representado pela sequência de bits 01001110, sendo que na configuração NZR-L, o sinal troca de polaridade sempre que o próximo bit for diferente do anterior, e no método NRZ-I o sinal troca de polaridade sempre que for transmitido um bit 1.

Um dos problemas que existem com estes três métodos apresentados acima é a sincronização dos relógios do emissor e receptor, pois ambos precisam estar exatamente no mesmo ciclo para não ocorrer pulos ou inversão de bits de leitura.

Outro problema existente nestes métodos é quando uma sequência grande de 0s ou 1s é transmitida. O que ocorre é uma distorção da potência média do sinal, sendo que o receptor pode ter dificuldade em reconhecer o valor do bit. O NRZ-I tem este problema um pouco aliviado, pois só ocorre em transmissões de sequências longas de 0s.

Para aliviar estes problemas, foi desenvolvido o método RZ (*Return-to-Zero)*, ou em português, Retorna a Zero, o qual utiliza três valores: positivo, negativo e zero. No método RZ, o sinal muda não entre bits, mas sim durante o bit. O que

ocorre é que o sinal vai à zero sempre na metade da transmissão de cada bit, e fica neste estado até o início da transmissão do próximo. A imagem abaixo ajuda a entender o processo:

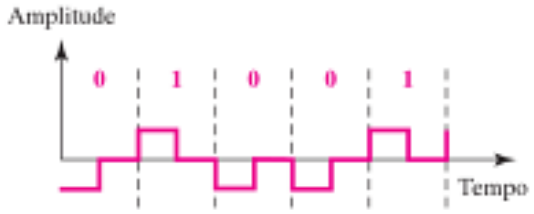

Figura 23- Método RZ.

Na figura acima, a sequência de dados a ser transmitida é representada pelos bits 01001, sendo que tensão negativa representa o nível lógico 0 e tensão positiva o nível lógico 1.

O problema deste método é a complexidade, pois utiliza três níveis de tensão, o qual é mais complexo de criar e distinguir. Como consequência, este método não é muito utilizado atualmente e ao invés disso utilizam-se métodos de Manchester, que melhoram o desempenho.

O método de Manchester combina os conceitos de RZ e NRZ-L em um único combo, sendo que a duração do bit é dividida em duas metades. A tensão permanece em um nível durante a primeira metade da transmissão e se desloca para outro nível na segunda metade. Esta transição no meio do bit é que justamente fornecerá o sincronismo. A figura abaixo apresenta o método de Manchester:

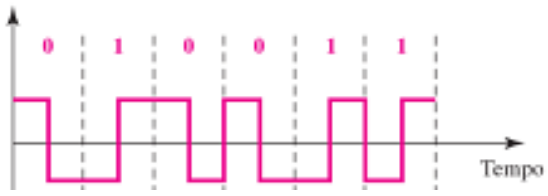

Figura 24- Codificação Manchester.

Neste método, um degrau ascendente significa um bit lógico 1 e um degrau descendente um nível lógico 0.

3.2.4.2: Modulação.

A modulação é o processo de modificação de um ou mais parâmetros de uma onda periódica, denominados sinais portadores, através de um sinal modulado o qual contém as informações a serem transmitidas.

Os três parâmetros fundamentais de uma onda são sua amplitude, frequência e fase. Qualquer um destes parâmetros pode ser modificado de maneira a se obter um sinal modulado.

Um dispositivo capaz de realizar a modulação é conhecido como modulador, e o dispositivo que transforma o sinal modulado em sinal original é denominado demodulador. Um modem é um dispositivo capaz de realizar as duas operações simultaneamente, enviando e recebendo sinais modulados.

O objetivo da modulação digital é a transferência de um sinal através de um canal de banda passante, onde os sinais são filtrados apenas para uma faixa de frequências pré-determinadas, evitando interferências e leituras incorretas.

Na modulação digital, o sinal portador analógico é modulado por um sinal discreto através de um método de modulação e transmitidos. Estes sinais ao serem recebidos, podem ser transformados novamente em sinais analógicos através do demodulador. Entre os métodos de modulação digitais podem ser citados:

- PSK (*Phase-shift keying)*.
- FSK (*Frequency-shift keying)*.
- ASK (*Amplitude-shift keying)*.

3.2.4.2.1: PSK.

O PSK, ou em português, Modulação por Deslocamento de Fase, converte sinais de portador modificando ou modulando a fase do mesmo. Existem duas maneiras fundamentais de utilizar a fase do sinal com este objetivo: identificando a fase por si mesma como portadora da informação, sendo que o demodulador necessita de um sinal de referência para comparar com o sinal recebido, ou observando as mudanças na fase como a informação, não precisando necessariamente de uma referência para o demodulador.

Basicamente, em uma modulação PSK, o bit 0 é representado por uma série de picos de onda em uma determinada frequência, e o bit 1 é representado por uma série de ondas de mesma frequência porém com um pico a mais ou a menos. Quando o modulador necessita alterar a característica de onda de um bit para outro, o que ocorre é a inversão de fase em 180^0. Com isso o demodulador pode identificar pelo número de picos a presença de um 0 ou 1 e pela alteração abrupta na fase a mudança do nível lógico. A figura abaixo tenta apresentar melhor este conceito:

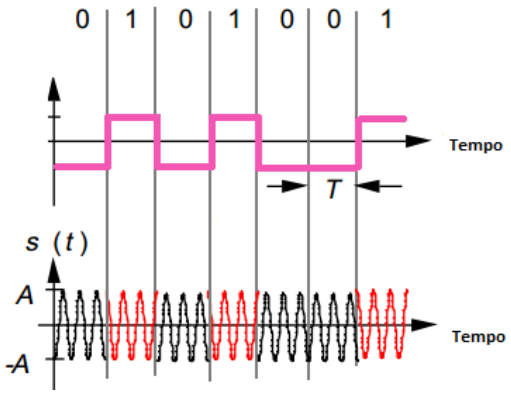

Figura 25- Modulação PSK.

Na figura acima, o nível lógico 0 possui três picos e o nível lógico 1 apenas dois picos, e a mudança de 180^0 se fase que caracteriza a modulação.

O padrão IEEE 802.11 de redes sem fio utiliza versões de PSK diferentes dependendo da necessidade de taxa de transmissão de dados.

3.2.4.2.2: FSK.

O FSK, ou em português, Modulação por Chaveamento de Frequência, converte sinais de portador modificando ou modulando sua frequência. O FSK utiliza um par de frequências discretas para transmitir dados binários. Nestes esquemas, o valor binário 1 é representado pela frequência alta, ou inteira, e a valor zero é representado pela frequência baixa, ou dividida. A imagem na próxima página representa melhor estes fatos:

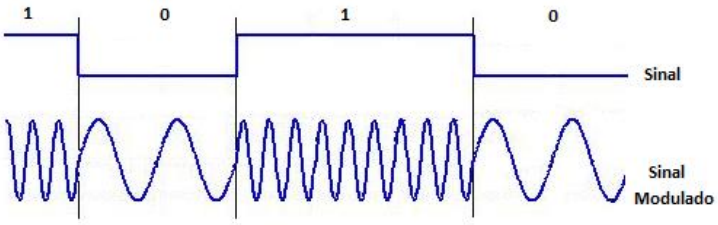

Figura 26- Modulação FSK.

Em alguns modelos de FSK, o demodulador pode reconhecer mais do que duas frequências, desta maneira são enviados mais que um bit por vez, aumentando a velocidade de transmissão.

3.2.4.2.3: ASK.

O ASK, ou em português, Modulação por Deslocamento de Amplitude, converte sinais do portador através da variação de amplitude. O princípio básico de funcionamento é como um interruptor, onde a presença do sinal do portador indica um nível lógico 1 enquanto que a falta de sinal indica um nível lógico 0. Estas modulações são denominadas de liga-desliga e seu funcionamento e apresentado abaixo:

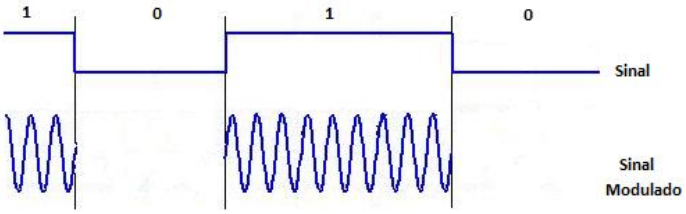

Figura 27- Modulação ASK liga-desliga.

O demodulador consegue identificar um nível lógico 0 ou 1 através da presença ou não de sinal. Um dos inconvenientes deste tipo muito básico de modulação por amplitude, é que é impossível para o demodulador reconstruir o sinal original, pois informação é perdida no nível lógico 0, e por isso versões mais recentes deste tipo de modulação não chaveiam o sinal e sim diminuem sua amplitude quando o nível lógico se torna zero. A imagem abaixo apresenta melhor esta diferença:

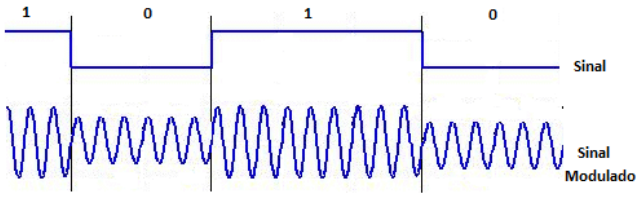

Figura 28- Modulação ASK.

Nesta configuração, o nível lógico 0 apenas sofre uma diminuição na amplitude do sinal, e não sua completa anulação. Com isso, o demodulador poderá reconstruir o sinal original se necessário.

Com isso foram demonstradas as principais técnicas para formatação de dados. Abaixo serão discutidos os canais de comunicação, responsáveis pelo envio físico dos dados e suas diferentes topologias.

3.2.4.3: Canais de comunicação.

Quando o assunto é redes de computação, um canal de comunicação refere-se ou à conexão física entre dois ou mais componentes para transmissão de dados (através de um fio, por exemplo) ou a conexão lógica entre estes mesmo elementos através de um canal sem fio de comunicação (ondas de rádio, por exemplo).

A comunicação entre dois elementos sempre necessitará de um caminho, um rumo a ser seguido, e este caminho é denominado canal de comunicação.

As topologias destes canais podem ser as mais diversas, sendo que algumas geram mais interesse industrial, com grandes aplicações e aceitação no meio, enquanto outras são mais recentes, recém-inseridas no meio.

Entre os tipos de canais de comunicação, podem ser citados: Comunicação Simplex, Half-Duplex, Full-Duplex, Ponto-a-Ponto e Multiponto. Mais sobre cada uma destas configurações é apresentado abaixo.

3.2.4.3.1: Comunicação Simplex.

Segundo Apurba Das[14], “a comunicação Simplex, definida pela ANSI” (*American National Standards Institute)* “é uma comunicação na qual todos os sinais só podem seguir em uma única direção”, ou seja, os papéis de transmissor e

receptor não se invertem durante o período de transmissão, sem qualquer tipo de retorno do receptor. A figura abaixo ajuda a entender melhor o conceito:

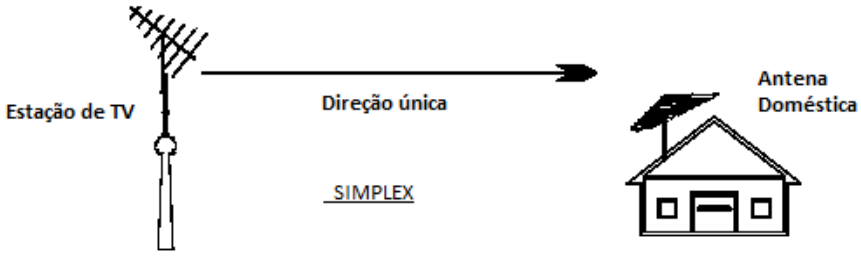

Figura 29- Comunicação Simplex.

Na figura acima, a estação de televisão transmite seu sinal através de antenas unidirecionalmente até o receptor, que transforma este sinal em imagens no aparelho de televisão. Em nenhum momento o aparelho retorna alguma mensagem ou sinal de volta para a estação.

Outros exemplos de comunicação Simplex são: portão de garagem, câmeras de vigilância e dispositivos de computação sem fio (como *mouses,* microfones e outros).

3.2.4.3.2: Comunicação Half-duplex.

Segundo Apurba Das (citado anteriormente), a comunicação Half-duplex é “a comunicação que provém troca de informações em ambos os sentidos, porém apenas uma direção de cada vez”, ou seja, transmissores e receptores trocam informações, porém geralmente o receptor deve esperar o término da transmissão do emissor para poder utilizar o canal de comunicação para o envio de sua resposta, como observado na figura abaixo:

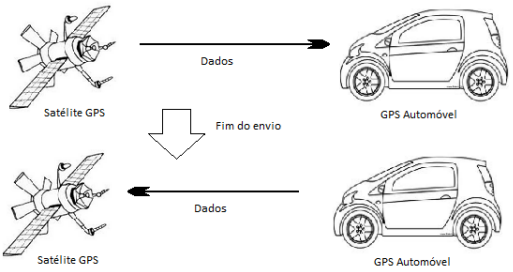

Figura 30- Comunicação Half-Duplex.

Nesta figura, o satélite envia informações para o GPS do carro que processa estes dados, espera o fim da transmissão e ai sim retorna outros dados para o satélite.

Outros exemplos de comunicação Half-Duplex são: *walkie talkies* e rádios amadores.

3.2.4.3.3: Comunicação Full-Duplex.

Segundo Apurba Das (citado anteriormente), a comunicação Full-Duplex "permite comunicação em ambos os sentidos e, ao contrário da Half-duplex, permite que isto ocorra simultaneamente", ou seja, o papel de transmissor e receptor não é mais bem definido já que qualquer componente pode realizar qualquer uma destas funções a qualquer hora. A imagem abaixo exemplifica a questão:

Figura 31- Comunicação Full-Duplex.

Geralmente a comunicação Full-Duplex é atingida ao se adicionar um canal a mais de comunicação, sendo um canal para cada sentido da informação. Nos cabos Ethernet, por exemplo, isto é obtido ao se adicionar mais um par de cabos trançados no conector, sendo que cada par realiza a comunicação em um sentido.

3.2.4.3.4: Comunicação Ponto-a-Ponto.

A comunicação Ponto-a-Ponto é definida por Apurba Das (citado anteriormente) como "Uma conexão direta entre dois nodos ou pontos terminais no sistema". Este tipo de topologia pode utilizar qualquer uma das configurações citadas acima (Simplex, Half-Duplex e Full-Duplex) desde que apenas dois elementos estejam envolvidos.

Nas comunicações Ponto-a-Ponto tradicionais, não existe um padrão de formatação de dados e pacotes, sendo que os dispositivos envolvidos em cada

ponta do canal de comunicação eram responsáveis por formatar os dados a serem transmitidos entre eles e interpretar os mesmos.

A implementação física do canal de comunicação pode ser as mais diversas, como par de fios trançado, RS-232, RS-485, *wireless*, infravermelho, entre outros. Mais sobre o RS-485 e uma solução *wireless* em particular serão discutidos em tópicos adiante.

3.2.4.3.5: Comunicação Multiponto.

Comunicação multiponto é aquela em que um número finito (maior ou igual a dois) de componentes participa do sistema. Neste tipo de configuração, cada elemento tem um papel especifico na rede, sendo que em muitos casos cada elemento deve responder a uma hierarquia de comando da própria rede, limitando quando e com quem os mesmos podem se comunicar.

A comunicação Multiponto pode tomar as mais diversas topologias de rede, sendo algumas livres e de uso mais geral e outras proprietárias e de uso mais específico. Entre elas podem-se citar:

- Redes Bus (barramento): Modbus, Fieldbus, etc.
- Redes Ring (anel): Ring, TokenRing, etc.
- Redes *Mesh* (malha): Zigbee, Meraki, etc.
- Redes Estrela: Zigbee, etc.

Algumas destas em especial chamam a atenção para este trabalho, sendo elas as redes Modbus e as topologias do padrão Zigbee, em especial a topologia *Mesh*.

A topologia bus gera interesse, pois é um dos padrões de rede presentes no software das IHMs em geral, e é bastante indicado para quando se deseja fazer a comunicação entre componentes externos e a IHM. Por ser um padrão aberto e livre, o Modbus é utilizado com muita frequência em projetos de automação industriais e por isso foi preferido neste projeto como o padrão bus a ser implementado.

O padrão Zigbee por sua vez é um padrão bastante recente que vem tomando os holofotes por se utilizar de redes sem fio, um tópico bastante atual e que

tem tomado à atenção de algumas empresas com necessidades próprias, onde redes cabeadas não são interessantes ou viáveis.

Por se tratar das duas redes presentes neste documento, um tópico exclusivo para o Modbus e um tópico exclusivo para o padrão Zigbee serão apresentados na sequência.

3.2.4.4: Padrão Modbus.

Modbus é um protocolo de rede criado em 1979 pela Schneider Eletric inicialmente com intuito de fazer a ligação entre sensores e CLPs. Devido a grande gama de aplicações encontradas nos últimos anos, se tornou um padrão *de facto* de comunicação.

Existem atualmente dezenas de versões do protocolo Modbus, como Modbus TCP/IP (utilizado para comunicação Modbus em redes TCP/IP) e Modbus ASCII (representação em ASCII para comunicação). Neste projeto é utilizado o Modbus RTU, um protocolo para comunicação serial através de cabos RS-232, RS-485, Ethernet, entre outros.

Derivado da arquitetura Mestre-Escravo, O Modbus é amplamente utilizado em Sistemas de Gerenciamento de Produção e em Sistemas de Automação Industrial sendo que um dos fatores para a ampla aceitação no meio é a facilidade de uso do protocolo.

O Modbus é considerado um protocolo de mensagem da camada de aplicação, a sétima camada no modelo OSI. Com isso, o protocolo deve trabalhar com pedidos e respostas para as camadas mais abaixo e entregar serviços especificados por funções de código. Estas funções de código são elementos da arquitetura do PDU (*Protocol Data Unit)* do protocolo. A tabela com estas principais funções é apresentada na próxima página:

Número do comando	Descrição
01	Lê as saídas digitais
02	Lê as entradas digitais
03	Lê saída analógica ou memória
04	Lê entrada analógica
05	Altera estado de uma saída digital
06	Altera estado de uma saída analógica
07	Lê registro de erros
08	Função de diagnóstico de erro
15	Altera várias saídas digitais
16	Altera várias saídas analógicas

Tabela 2- Funções Modbus.

Para a aplicação construir uma unidade de dados Modbus, o cliente deve iniciar uma transação Modbus. Realizada esta parte, o mestre da rede cria um pedido no formato do protocolo e envia ao escravo. O escravo recebe esta mensagem e seu servidor sabe exatamente qual ação deve tomar analisando o bit do pacote referente a função apresentada na tabela acima.

As mensagens no protocolo Modbus RTU são pacotes de tamanho variável com CRC (*Cyclic-Redundant Checksum)* de 16 bits. Estes CRCs são responsáveis pela validação do pacote, a através de uma soma dos valores da sequência de 16 bits, o receptor pode entender se o pacote recebido condiz com o pacote enviado através da comparação deste valor de CRC do emissor com o valor de CRC que o receptor próprio realiza. Segundo Drury [15], o formato do quadro (frame) Modbus RTU é descrito na tabela da próxima página:

Formato do quadro Modbus RTU	
Nome	Comprimento (bits)
Início	variável
Endereço	8
Função	8
Dados	n*8
CRC	16
Fim	variável

Tabela 3- Formato do quadro Modbus RTU.

Pacotes Modbus são desenhados apenas para enviar dados, não sendo possível enviar parâmetros como nome de pontos, resoluções, unidades, entre outros. Para estes casos o protocolo não é indicado e sim protocolos mais complexos como BACnet, Ethernet IP, entre outros.

Cada byte do campo dados do pacote Modbus RTU contém um cabeçalho extra denominado *start bit*, ou em português bit de partida. Este cabeçalho é o identificador de início do byte, necessário para que o receptor saiba quando o mesmo iniciou. De maneira análoga, no final do byte é acrescentado um bit denominado *stop bit*, ou bit de parada. Este bit é responsável por avisar o término do byte. Ainda se especificado explicitamente na chamada inicial do protocolo Modbus, mais um bit extra pode ser adicionado, chamado bit de paridade. Este bit é considerado um adicional ao CRC, fazendo uma checagem da recepção correta de cada byte de dados. O funcionamento deste bit é muito simples, ele recebe o valor da soma de todos os bits do byte a ser enviado e o receptor faz a soma dos bits do pacote recebido, conferindo com o campo bit de paridade do byte.

Cada nó em um sistema Modbus recebe um endereço entre 1 e 254, sendo o endereço 0 reservado apenas para mensagens tipo *broadcast* e de escrita. Além disso, o endereço 0 não é muito utilizado pois justamente como foi dito acima, não existe confirmação da mensagem recebida pelo escravo, limitando o uso.

Geralmente, na camada física, o protocolo Modbus conta com fiação RS-232 ou RS-485. Estes são dois padrões de comunicação serial e tem como diferença básica o número de fios e a sua configuração no terminal DB9 (presente em ambos).

Para comunicação Full-Duplex, o RS-232 geralmente necessita de 5 fios, enquanto o RS-485 necessita de 4 fios. Outra diferença é que o RS-232 necessita necessariamente de uma configuração ponto-a-ponto, enquanto que o RS-485 permite até 32 nodos no mesmo sistema. Expansões comerciais podem aumentar este valor para 64, 128 ou 256 nodos.

Apesar das limitações do Modbus RTU, ainda existem diversas razões para considera-lo um candidato para a maioria das aplicações industriais. Por exemplo, o protocolo é muito mais fácil de ser implementado que a maioria dos protocolos mais recentes, como os citados acima e possui um conhecimento já aceito dento do ramo. Além disso, falando mais especificamente deste projeto, a implementação deste protocolo completo em um microprocessador consumirá por volta de 2Kb de memória, sendo que um protocolo Ethernet ou BACnet utilizaria entre 30-100 Kb, mais do que a memória disponível em um Arduino Uno, por exemplo.

Como desvantagens do Modbus e de maneira mais geral, como desvantagem da topologia bus pode-se citar:

- Se o cabo principal falhar, todo o sistema falha.
- Reconfiguração de rede pode ser dificultosa.
- Quanto mais longe estão os equipamentos do mestre mais a potência do sinal é dissipada no meio.
- Os dados são removidos ao serem recebidos pelo escravo ou mestre, para evitar repetições na linha.
- Quesitos de segurança de pacotes contra acessos ilegais externos não são bem definidos.

Analisando os prós e contras do protocolo em questão e relembrando sempre a questão fundamental que circula este projeto, da busca pela solução adequada e que atende as especificações do problema, fica claro que utilizar o protocolo Modbus como forma de comunicação entre Arduino e IHM é a solução correta.

3.2.4.5: Padrão IEEE 802.15.4.

Antes de apresentar explicitamente o padrão Zigbee que é o núcleo deste trabalho, e necessário primeiramente apresentar o padrão IEE 802.15.4, o qual gere as regras e inspirou a construção do padrão Zigbee.

O IEE 802.15.4 é um padrão para comunicação sem eletrônica sem fio ciado pela IEEE (*Institute for Eletrical Electronics Engineer)*, uma associação de profissionais técnicos que vêm padronizando protocolos de comunicação, visando o crescimento e a interoperabilidade de tecnologias existentes e aquelas que poderão existir.

Este padrão define regras de comunicação da camada física e da camada de enlace (principalmente do MAC) para redes de comunicação de baixa taxa de transmissão.

Segundo Ramos, “é utilizado em aplicações nas quais é necessário baixa taxa de transmissão e baixo consumo de potência. É o padrão base para todos os protocolos de classe LR-WPAN (*Lower Rate Wireless Personal Area Network)*...dentre eles o protocolo Zigbee”.

Este padrão define especificamente duas camadas do modelo OSI, que é a camada de física e o Controle de Acesso ao Meio da camada de enlace, conforme figura abaixo:

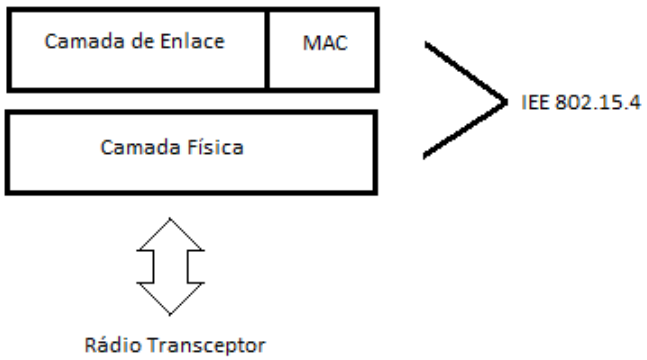

Figura 32- Camadas do modelo OSI envolvidas no padrão IEEE 802.15.4.

A camada física define características de hardware e comandos elétricos. Sua principal tarefa é transmitir e receber dados do meio externo e adequá-los para

enviar à camada de enlace. Nesta camada, técnicas de modulação de sinal apresentadas anteriormente são utilizadas para transmitir e demodular o canal.

Dentre as principais funcionalidades da camada física, segundo o padrão, podemos citar:

- Enviar dados para a camada de enlace.
- Recepção e transmissão de dados digitais.
- Ativação/desativação do transceptor de rádio.
- Detecção de Energia no canal a ser utilizado.
- Seleção do canal a ser utilizado.
- Indicação da qualidade das conexões através dos pacotes recebidos.
- Técnicas de acesso múltiplo ao canal.

A camada de enlace, e particularmente a porção do MAC desta camada, é responsável por toda operação que envolve o canal físico de comunicação. Entre as funcionalidades, segundo o padrão, pode-se citar:

- Fornecimento de uma conexão confiável entre duas entidades MAC.
- Manipulação do mecanismo da técnica de acesso ao canal.
- Associação/desassociação de PAN (*Personal Area Network,* que significa o número da rede).
- Sincronismo.
- Segurança do dispositivo.

O padrão IEE 802.15.4 opera na banda de frequência ISM (*Industrial Scientific Medical*), podendo trabalhar nas frequências de 869 MHz, 915 MHz e 2.4 GHz. A tabela na próxima página mostra as principais propriedades do padrão:

Descrição	869 / 915 MHz	2.4 GHz
Taxa de transmissão	40 kbps	250 kbps
Potência de saída na transmissão	1 mW (0 dB)	1 mW (0 dB)
Sensibilidade de recepção	-92 dBm	-85 dBm
Alcance da transmissão	interno 100 metros externo 200 metros	interno 100 metros externo 200 metros
Latência	15 ms	15 ms
Canais	10	16
Numeração do canal	1-10	11-26

Tabela 4- Propriedades do padrão 802.15.4.

A potência de saída do transmissor e a sensibilidade de recepção do receptor são fatores determinantes para analisar a energia de alcance do sinal.

Quanto maior a potência de saída, maior será o alcance do sinal. De outra forma, a sensibilidade do receptor determina a menor potência necessária para se interpretar o sinal.

Das três frequências estabelecidas pela ISM, somente a de 2.4 GHz pode operar em todo o mundo. A frequência de 869 MHz é utilizada na Europa e de 915 MHz é usada na América do Norte e América do Sul.

Dois tipos de dispositivos, segundo o padrão, participam do sistema, que são o FFD (*Full Feature Device*) que desempenham todos os papéis possíveis na rede como formar redes, rotear dados, e se comunicar com todos os outros dispositivos, e os RFD (*Reduce Feature Device*) que não possuem função de rotear dados e por isso só se comunicam com dispositivos FFD.

O padrão possui ainda três papéis na rede sem fio, o Coordenador PAN, responsável por criar as redes sem fio, o roteador e os *end devices*, dispositivos finais da rede.

A LR-WPAN opera em duas topologias de rede, que são estrela e *peer-to-peer*. A figura abaixo apresenta a topologia de estrela:

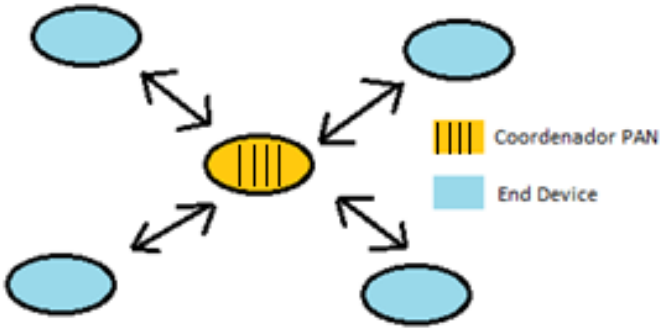

Figura 33- Topologia estrela.

Nesta topologia, o nó central é o coordenador PAN com o qual todos os outros *end devices* da rede se comunicam.

Na configuração *peer-to-peer*, apresentada na figura baixo, todos podem se comunicar diretamente entre si, respeitando o tipo de dispositivo em questão, sendo a base para a comunicação em rede *Mesh*.

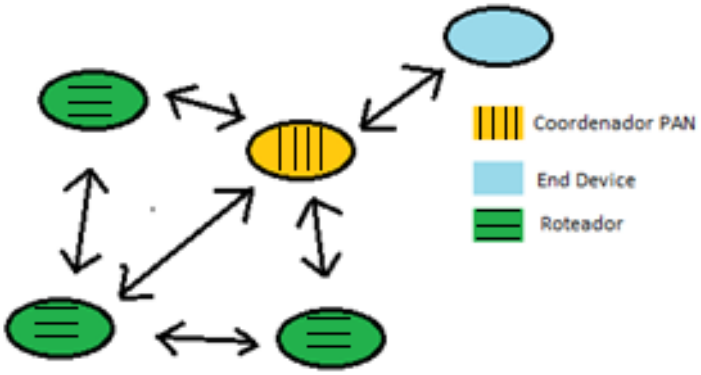

Figura 34- Topologia *peer-to-peer.*

Apresentado agora o padrão IEE 802.15.4 pode-se caminhar em direção ao núcleo do capítulo, a apresentação do padrão Zigbee, presente no próximo tópico.

3.2.4.6: Padrão Zigbee.

Em 2002, grandes empresas do setor de eletrônicos e áreas afins decidiram padronizar um sistema de comunicação eletrônica sem fio para o setor de automação. Eles juntaram forças com a finalidade promissora de tornar este sistema compatível independentemente da empresa que produzisse o circuito. Esta junção de forças ocasionou a formação de uma sociedade conhecida como Alliance Zigbee.

A Alliance Zigbee é responsável pela criação e manutenção de um padrão de rede de comunicação sem fio denominada padrão Zigbee, considerado aberto e livre para aquisição. Todos os dispositivos comerciais projetados com este padrão devem passar por testes de homologação antes de serem distribuídos no mercado.

O padrão Zigbee pode ser definido como uma convenção de comunicação de rede sem fio, a qual se utiliza de uma pilha de protocolo particular com a qual é capaz de manter uma comunicação com outros dispositivos do mesmo padrão. Geralmente um microcontrolador, detentor em sua memória do protocolo em questão, deve se comunicar com os outros microcontroladores pertencentes a uma rede comum, usando um transmissor de rádio para que os pacotes sejam enviados sem fio numa faixa de frequência específica pelo padrão.

O Zigbee pode assumir diversas topologias de rede, tendo a capacidade de grande economia de consumo de energia quando utilizados em modo sono (*sleep*). Baterias de lítio podem durar, nestes casos, anos para serem descarregadas por um dispositivo.

Analisando o perfil técnico, o padrão Zigbee utiliza quatro camadas de rede. As duas camadas de nível mais alto foram desenvolvidas pelo protocolo Zigbee e as duas camadas inferiores pelo protocolo IEEE 802.15.4, sendo que a camada física e a de enlace são as mesmas do padrão IEEE, enquanto que a camada de rede e aplicação desenvolvidas pela Alliance Zigbee.

3.2.3.6.1: Classificação.

Para melhor ficar situado nos conceitos da comunicação sem fio, podemos classificar, em termos de distância de comunicação, duas técnicas de comunicação de baixo alcance bastante utilizadas no mercado: WLAN e WPAN, conforme figura na próxima página:

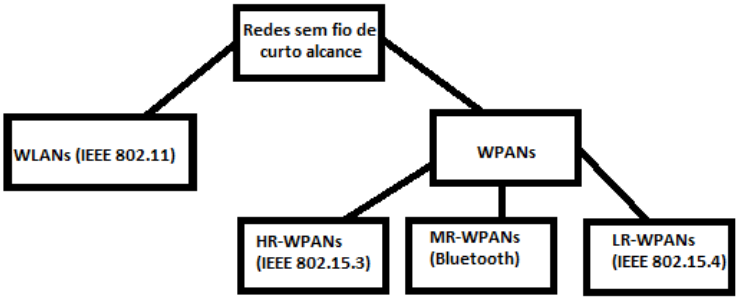

Figura 35- Diagrama de redes sem fio de curto alcance.

WLAN (IEEE 802.11) é uma classe de redes sem fio de curto alcance que propões a extensão, ou até mesmo a substituição, da rede de comunicação com fios (LANs) sendo que a WLAN pode ser integrada facilmente com a LAN.

O objetivo dos padrões utilizados pela WLAN é a maximização da taxa de transmissão e da faixa do raio de alcance na cobertura. Este padrão trabalha com frequência de operação em torno de 2.4GHz, assim como o padrão Zigbee e Bluetooth, e possui taxa de transmissão de até 11 Mbps.

As WPANs, em contraste com as WLANs, foram projetadas com o objetivo de substituir as redes de comunicação com fio. Essa classe pode se dividir em tr6es subclasses, conforme visto na figura acima.

O padrão IEEE 802.15.3, subclassificado como área de rede pessoal sem fio de alta transmissão (com o acrônimo de HR-WPANs), suporta uma taxa de transmissão de dados em torno de 11 a 55 Mbps, podendo, assim, cobrir um raio de 30 a 100 metros, trabalhando numa faixa de frequência de 2.4GHz.

Este padrão foi desenvolvido para tarefas que exijam alta taxa de transmissão, como o envio de um vídeo de uma câmera para uma televisão próxima. A grande desvantagem de usar tal tecnologia é sua alta complexidade de aplicação associada ao alto custo comercial, conforme figura na próxima página:

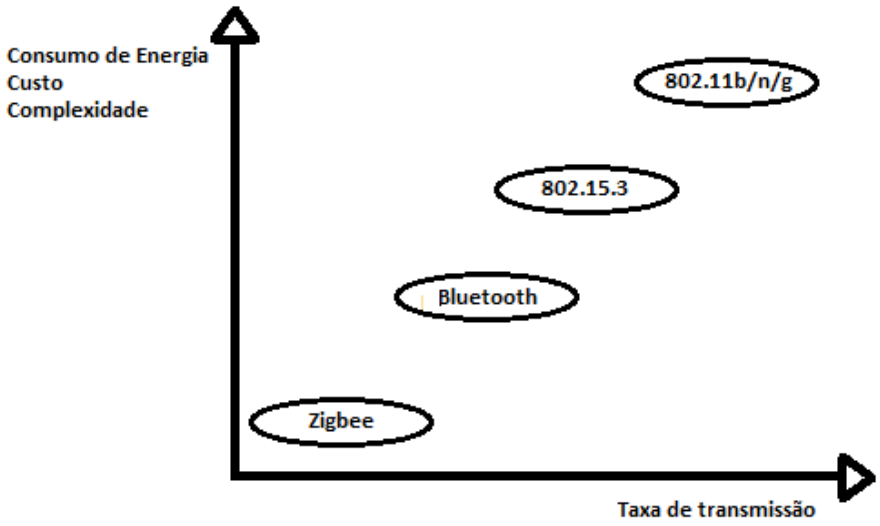

Figura 36- Comparativo de tecnologias.

O padrão Bluetooth, que se enquadra na subclasse como rede de área pessoal sem fio de média transmissão (com o acrônimo de MR-WPAN), atinge uma taxa de transmissão em torno de 1 a 3 Mbps e pode ser utilizado para substituir acessórios de computadores como fones de ouvido, *mouse*, etc. Este padrão exige menor complexidade e custo em relação ao IEEE 802.15.3, mas ainda assim consome uma significativa energia de alimentação.

Já o padrão Zigbee possui a menor complexidade e o menor custo da classe WPAN. Seus dispositivos consomem a menor energia de alimentação. Este baixo custo é refletido na minimização da taxa de transmissão, sendo que foi projetado para receber simples comandos o obter informações de sensores como, por exemplo, temperatura e umidade.

3.2.3.6.2: Tipos de dispositivos Zigbee.

No padrão Zigbee podem ser observados dois tipos de dispositivos que interagem entre si em uma rede. O tipo de dispositivo está intimamente ligado à quantidade de tarefas que uma pilha de instrução armazenada na memória do microcontrolador pode executar. São eles o FFD e o RFD, citados no tópico sobre o padrão 802.14.4.

Pode-se ainda, atribuir a cada um destes dispositivos um papel na rede, sendo que dispositivos FFD podem assumir o papel de coordenador e roteador enquanto que dispositivos RFD são os dispositivos finais.

O coordenador é responsável por formar a rede de trabalho, selecionando um canal de comunicação e um endereço específico. Ele permite a seleção de acesso aos roteadores e dispositivos finais que pretendem juntar-se à rede de trabalho preestabelecida.

O roteador tem o papel de oferecer rota de caminhos alternativos para pacotes de dados, quando um destino possui certas barreiras físicas, ou aumento do alcance. Ele pode transmitir e receber dados.

Dispositivos finais exercem tarefas reduzidas RFD, sendo seu papel relacionado com o envio e recebimento de informações advindas de dispositivos FFD.

3.2.3.6.3: Modos de operação dos dispositivos Zigbee.

Os modos de operação estão diretamente ligados ao consumo de potência elétrica que o dispositivo pode dispender em seus diferentes modos de trabalho. Os modos que o padrão Zigbee propõe são os seguintes:

- Modo Inativo.
- Modo de recepção.
- Modo de comando.
- Modo sono.
- Modo transmissão
- Modo ativo.

O modo inativo é o momento em que o dispositivo não está recebendo ou transmitindo dados. Neste modo o dispositivo Zigbee está em processo de verificação de dados vindos da antena. O dispositivo sai deste estado quando dados serias advindos do *buffer* estão prontos para serem empacotados (modo de transmissão) ou dados prontos da transmissão são coletados da antena do dispositivo (modo de recepção).

O dispositivo entra em modo de recepção quando um dado válido (ou informação, cujo formato pertence ao padrão Zigbee) é recebido, o dado é transferido para o buffer de transmissão serial.

O dispositivo entra em modo de comando quando é necessário ler ou modificar parâmetros do módulo de radiofrequência. Ao entrar neste modo, todos os dados recebidos são interpretados como linhas de comando.

Já o modo de sono é o modo de menor consumo de energia, chegando a consumir alguns microampères. É a solução encontrada para que o padrão Zigbee e seus módulos gastem menos energia da classe de rede de curto alcance.

Quando o dispositivo recebe os dados seriais e está pronto para empacota-los, o mesmo entra em modo de transmissão. O endereço do destino do dado determina qual módulo na rede que receberá este dado.

No modo de transmissão, o dispositivo assegura-se de que o endereço do módulo que receberá os dados e a rota dos dados são previamente conhecidos. Se todos os requisitos forem satisfeitos, os dados serão transmitidos.

Quando um dado é transmitido de um módulo para outro, uma mensagem de confirmação é enviada para o transmissor pela mesma rota em que o dado chegou. A função dessa mensagem é informar ao módulo transmissor que o dado foi recebido com sucesso. Caso esta mensagem não chegue, o dado será retransmitido.

Por último, o modo ativo é considerado quando o dispositivo ou está em modo de recepção, ou de comando, ou de transmissão.

3.2.3.6.4: Camadas do sistema Zigbee.

Como mencionado anteriormente, o Zigbee possui quatro camadas de rede, sendo elas: Aplicação, rede, enlace e física, sendo que as duas últimas seguem o padrão 802.15.4 e foram mencionadas anteriormente.

A camada de rede, por sua vez, oferece basicamente dois serviços: Dados e gerenciamento. O fluxo de dados que trafegam na rede, a distribuição de endereços de 16 bits na rede formada pelo coordenador, o controle da distância em que um

frame pode viajar na rede, a forma de comunicação do *frame* com os demais dispositivos (se em comunicação *multicast*, *broadcast* ou *unicast*) são todas funções da camada de rede.

Já a camada de aplicação é o nível mais alto do sistema de comunicação sem fio Zigbee. Podemos encontra nela três seções: APS (*Application Support Sublayer*), ZDO (*Zigbee Device Objects*) e *Application Framework*.

A APS oferece basicamente uma interface entre a camada de rede e a camada de aplicação.

A *Application Framework* é a subcamada em que estão os objetos de aplicação armazenados para o controle e gerenciamento dessa camada num dispositivo Zigbee. Estes objetos de aplicação são desenvolvidos para um específico perfil de aplicação e produzidos pelas empresas que fabricam os dispositivos Zigbee, sendo na maioria das vezes proprietário.

Em um dispositivo Zigbee podem ser armazenados 240 objetos de aplicação diferentes. Com um único rádio é possível ter o controle de 240 dispositivos conectados a ele.

Por último, o ZDO é responsável por gerenciamento geral do dispositivo, além de questões de segurança e política do protocolo.

3.2.3.6.5: Topologia de redes Zigbee.

A topologia de uma rede Zigbee é gerenciada pela camada de rede, podendo assumir a topologia estrela ou *peer-to-peer,* apresentadas anteriormente.

Uma topologia estrela, os dispositivos finais apenas se comunicam com o coordenador, não havendo a possibilidade de dados serem retransmitidos pelos roteadores. Esta topologia é análoga ao sistema de comunicação mestre-escravo.

Na topologia *peer-to-peer*, cada módulo FFD Zigbee pode se comunicar com o módulo FFD mais próximo, havendo, assim, a possibilidade de roteamento de dados. Nesta topologia, a rede é formada com os módulos exercendo todos os papéis estabelecidos pelo padrão.

Dependendo de como é organizada a inclusão de novos membros da rede, a topologia *peer-to-peer* pode assumir duas subdivisões, sendo topologia de rede *Mesh* e topologia de rede em árvore. Uma figura sobre a topologia *Mesh* é apresentada abaixo:

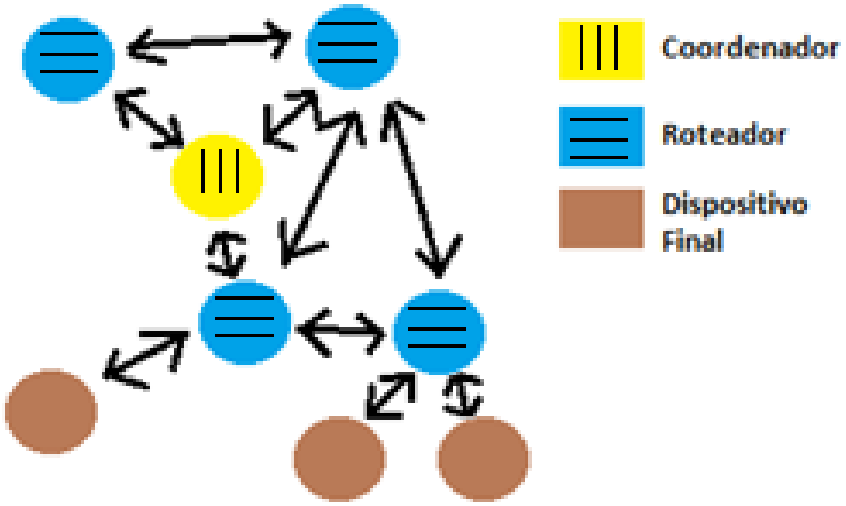

Figura 37- Topologia Mesh.

A topologia de rede *Mesh* pode ser diferenciada quando os novos membros FFD podem se comunicar com os módulos FFD da rede que estão dentro do seu alcance de radiação eletromagnética. O único impedimento para não haver comunicação com outros dispositivos é justamente o alcance, sendo que os dispositivos finais podem utilizar dos roteadores para se comunicar com o coordenador.

Uma imagem sobre a topologia em árvore é apresentada na próxima folha:

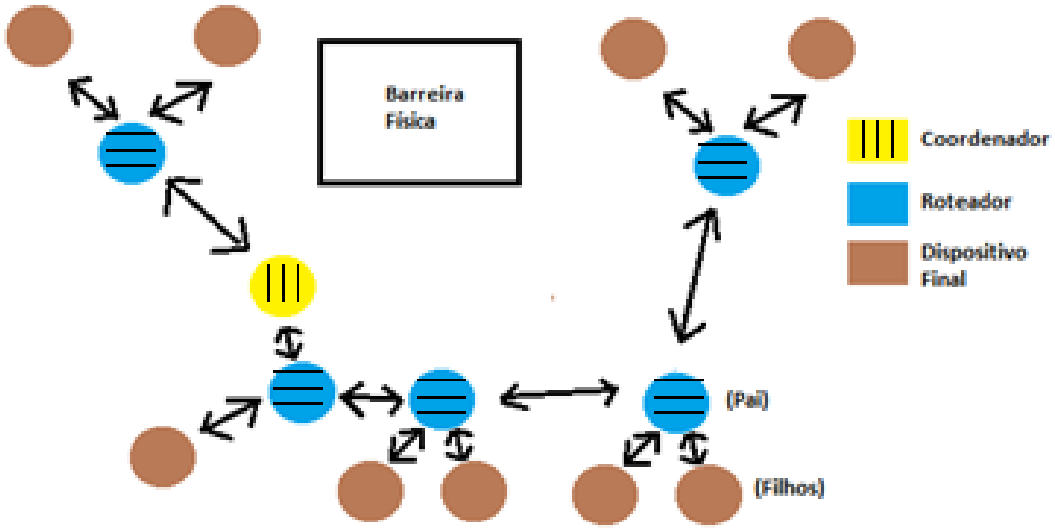

Figura 38- Topologia Árvore.

Na topologia em arvore, os membros só podem crescer de forma metódica e organizada. O nome dessa topologia é devido à forma como os novos dispositivos se juntam à rede, pois os roteadores funcionam como galhos e os dispositivos finais como folhas. Quando novos dispositivos finais precisam se junta à rede, eles precisam verificar o módulo pai (FDD) disponível mais próximo que pode recebê-los. Em outras palavras, a rede pode se expandir com a inclusão de novos dispositivos roteadores.

Note na figura acima, que se um dispositivo final do lado esquerdo da barreira física quiser enviar um dado para o dispositivo final do lado direito da barreira, os dados podem assumir uma rota que viabilize essa comunicação.

3.2.3.6.6: Formação de uma rede Zigbee.

A rede de trabalho dos dispositivos Zigbee é chamada de PAN. Cada rede formada possui um endereço específico chamado PAN ID, que é o meio para diferenciar as redes formadas por diversos dispositivos coordenadores. Esse endereço deve ser armazenado em cada dispositivo Zigbee ao se juntar à rede específica.

Para que haja uma PAN, o coordenador realiza uma série de leituras com intenção inicial de procurar um endereço e um canal de frequência disponível para a comunicação. Para isso, ele verifica o nível de energia de cada canal testado

(*Energy Scan*). Se o nível de energia de um canal for alto, esse canal não entrará na lista de memória de canal disponível no coordenador.

Ao verificar a energia dos canais testados, o coordenador verifica se os canais habilitados na sua memória estão sendo usados por outro coordenador. Para essa verificação, o coordenador envia um quadro em *broadcast*, solicitando os canais e endereços que estão sendo utilizados numa possível rede de trabalho.

Coordenadores ou roteadores respondem a essa solicitação enviando um *frame* com o canal e o endereço já usados na sua própria rede e se há ou não dispositivos para se juntar à rede em formação.

Depois de ter feito o teste de energia e a leitura dos canais e endereços utilizados, o coordenador analisa todos os *frames* recebidos, contendo informações de canais e endereços utilizados, e tenta conectar-se a um canal e endereço não utilizados, formando assim, uma rede de trabalho. O coordenador então se apodera deste canal e endereço. Após isso, roteadores e dispositivos finais podem ter acesso à rede formada.

É importante frisar que só há um dispositivo coordenador numa rede formada. Não podem existir na mesma rede dois dispositivos coordenadores com o mesmo PAN ID.

Todos os dispositivos da rede Zigbee recebem um endereço de 16 bits ao se juntar a PAN, sendo que o endereço do coordenador sempre é 00h.

3.2.3.6.7: Junção a uma PAN.

Após o coordenador iniciar uma PAN, os roteadores ou dispositivos finais devem se juntar a uma PAN próxima para formar uma rede de trabalho. Para isso eles fazem um processo de leitura de PAN para verificar as PANs mais próximas disponíveis. Após a leitura, o roteador ou dispositivo final analisa uma lista de PANs disponíveis para se juntar. No caso de equipamentos RFD, os mesmos só podem se juntar a um único módulo pai (FFD).

3.2.3.6.8: Segurança.

Se configurado como estado de segurança, o coordenador inicia a rede Zigbee usando uma chave de criptografia de 128 bits. Somente dispositivos que possuem a mesma chave podem ter acesso à rede formada. Esta chave pode ser adquirida por pré-configuração ou na aquisição da conexão de rede existente.

3.2.3.6.9: Coexistência.

Coexistência é a capacidade de dispositivos sem fio de diferentes tecnologias poderem operar perto um dos outros sem interferência na mesma faixa de frequência. O padrão Zigbee pode trabalhar na banda de frequência 2.4 GHz, assim como o IEEE 802.11b/n/g, Bluetooth e alguns telefones sem fio. Todas essas tecnologias podem trabalhar próximas sem que módulos Zigbee causem dano aos dados transmitidos, ou seja, coexistem.

Como o protocolo Zigbee é destinado a operar numa faixa frequencial que não necessita de licenciamento, seus dispositivos deveriam sofrer interferência de outros aparelhos que trabalham na mesma frequência, porém técnicas como espalhamento espectral, CSMA/CA, CCA, entre outras, permitem a coexistência sem interferência nos módulos Zigbee.

3.2.3.6.10: Licenciamento da frequência utilizada.

A grande maioria dos países do mundo possui órgãos responsáveis pela regulamentação da radiofrequência utilizada na comunicação sem fio. No Brasil, este órgão se chama Anatel (Agência Nacional de Telecomunicações).

Os dispositivos utilizados neste trabalho que possuem o protocolo Zigbee são homologados pela Anatel e não necessitam de licença para operar devido à baixa potência e à faixa de frequência de operação que estes módulos possuem.

3.2.3.6.11: Equipamentos com padrão Zigbee.

Como mencionado anteriormente, existem dezenas de fabricantes de módulos de comunicação baseados no protocolo Zigbee. O líder mundial na fabricação de módulos com protocolo Zigbee é a Digi International.

A Digi International, fundada em 1985 em Minnetonka, Estados Unidos, é uma empresa pioneira no ramo de soluções M2M (*Machine-to-Machine)* e atua em diversas áreas de automação, entre elas a de redes de sensoriamento sem fio.

O equipamento da Digi que possui o protocolo Zigbee é chamado de Xbee, e suas diversas configurações, opções e funcionamentos são apresentados no próximo tópico.

3.2.4.7: Módulos Xbee.

O módulo Xbee pode ser definido como uma plataforma embarcada de radiofrequência que utiliza a mesma configuração física de pinos para diferentes séries de arquiteturas de *hardware* e *software*. Se for necessário substituir um *hardware* da série 1 por um *hardware* da série 2, não haverá problema nas configurações dos pinos, pois eles terão os mesmos espaçamentos, numeração e, dependendo do *hardware*, os pinos terão as mesmas funções. No entanto, o protocolo destinado a uma série não funciona em outra.

Estes módulos devem conter um *firmware*, cuja seleção depende da aplicação desejada. O produto final Xbee depende da combinação *hardware* mais *firmware* e algumas variações são mostrados abaixo:

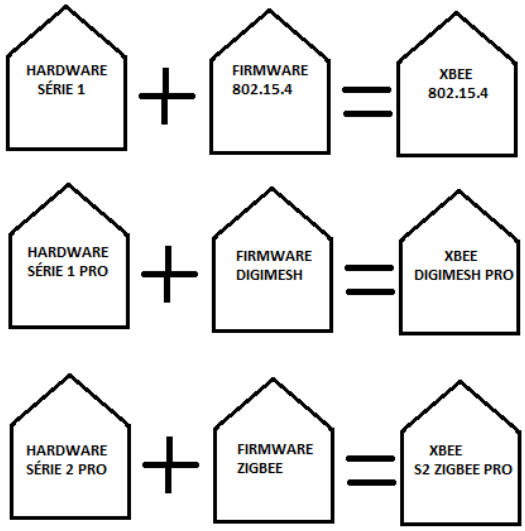

Figura 39- Algumas combinações de Xbee.

Como pode ser observado na figura acima, o *hardware* da série 1 acrescido do protocolo 802.15.4 forma o produto final Xbee 802.15.4 e assim sucessivamente.

3.2.3.7.1: XBee/Xbee-PRO.

Os *hardwares* podem ser classificados pela potência de transmissão em Xbee e Xbee-PRO. Os módulos Xbee-PRO possuem maior potência de transmissão tanto em ambientes externos (sem barreiras) como em ambientes internos (com barreiras), por esse motivo são usados quando se deseja um maior raio de alcance na comunicação entre os módulos.

Cada série de *hardware* pode possuir esses dois tipos de módulos ou apenas um deles. Para cada série de *hardware* pode haver ligeiras diferenças nas distâncias interno-externas de comunicação, topologia de rede, forma de comunicação, recursos entre outros, conforme tabela abaixo:

	Série 1 (802.15.4)		Série 2 (Zigbee)	
	Xbee	Xbee-PRO	Xbee	Xbee-PRO
Distância interna	30 m	90 m	40 m	90 m
Distância externa	90 m	1600 m	120 m	3200 m
Potência de transmissão	1 mW	63 mW	2 mW	50 mW
Sensibilidade de recepção	-92 dBm	-100 dBm	-96 dBm	-102 dBm
Corrente de pico na transmissão	45 mA	250 mA	40 mA	295 mA
Corrente na recepção	50 mA	55 mA	40 mA	45 mA
Taxa de transmissão	250 kbps	250 kbps	250 kbps	250 kbps

Tabela 5- Recursos das séries Xbee.

Observando a tabela acima, a primeira análise que pode ser feita é que realmente o alcance dos módulos PRO tem maior alcance que módulos que não são PRO tanto para a série 1 quanto para a série 2. Consequentemente, em módulos PRO a potência de transmissão é maior, gerando maiores correntes e exaurindo com mais rapidez uma bateria se a mesma fosse conectada aos mesmos. Ainda analisando a tabela acima, os módulos da série 2 conseguiram aumentar o alcance ainda mais em relação aos módulos da série 1 mas sem necessariamente aumentar

a potência de transmissão, ou seja, se a necessidade implica um alcance maior, módulos da série 2 PRO são mais eficientes energicamente do que módulos da série 1 PRO.

3.2.3.7.2: Características do Hardware Xbee/Xbee-PRO.

A figura abaixo apresenta o *hardware* de um módulo XBee-PRO da série 2:

Figura 40- Xbee-PRO série 2.

Esta versão de Xbee já vem com uma pequena antena integrada do tipo fio rígido, responsável pelo envio e recepção dos dados entre os módulos. Existem, ainda, outras versões com antenas estilo Chip ou com conectores RF para antenas externas.

O Xbee possui 20 pinos, cada um com uma determinada função. A tabela na próxima folha apresenta estas funções:

Número do pino	Nome	Direção	Estado padrão	Descrição
1	Vcc	----------	----------	Alimentação elétrica de tensão constante
2	DOUT	Saída	Saída	Saída de dado UART
3	DIN / CONFIG	Entrada	Entrada	Entrada de dado UART
4	DIO 12	Ambas	Desabilitado	Entrada / Saída digital 12
5	RESET	Ambas	Coletor Aberto	Reseta o módulo (pulso 20 ns)
6	RSSI PWM / DIO 10	Ambas	Saída	Indicação de potência do sinal Rx ou Entrada / Saída digital 10
7	DIO 11	Ambas	Entrada	Entrada / Saída digital 11
8	[reservado]	Ambas	Desabilitado	Não deve ser conectado
9	DTR / SLEEP_RQ / DIO8	Ambas	Entrada	Pino de controle do modo de sono ou Entrada / Saída digital 8
10	GND	----------	----------	Ligação ao terra
11	DIO 4	Ambas	Desabilitado	Entrada / Saída digital 4
12	CTS / DIO 7	Ambas	Saída	Controle de fluxo que limpa dados de envio ou Entrada / Saída digital 7
13	ON / SLEEP	Saída	Saída	Indicação do estado do módulo
14	Vref	Entrada	----------	Tensão de referência para medições analógicas
15	Associação / DIO 5	Ambas	Saída	Indica Associação ou Entrada / Saída digital 5
16	RTS / DIO 6	Ambas	Entrada	Controle de fluxo que requisita dados de envio ou Entrada / Saída digital 6
17	AD 3 / DIO 3	Ambas	Desabilitado	Entrada analógica 3
18	AD 2 / DIO 2	Ambas	Desabilitado	Entrada analógica 2
19	AD 1 / DIO 1	Ambas	Desabilitado	Entrada analógica 1
20	AD 0 / DIO 0	Ambas	Desabilitado	Entrada analógica 0

Tabela 6 – Tabela de pinos do Xbee.

Como pode ser visto na tabela acima, grande parte das portas podem ser configuradas via programação para realizar uma determinada função, sendo que quatro das portas são para entradas analógicas, como, por exemplo, sensores.

Com isso, pode-se ter uma noção básica do protocolo Zigbee e dos módulos Xbee que utilizam este protocolo. Ainda há muito para apresentar sobre as nuanças do protocolo, a criação de pacotes, modos de transmissão de dados e como interligar os módulos, entre outros, que serão apresentados nos capítulos que apresentarão a implementação deste projeto em si, sendo que o objetivo deste é apenas dar uma visão geral sobre as tecnologias envolvidas.

Completado este passo, termina aqui o tópico denominado “envio de dados” e pode-se dar sequência à próxima área de ataque do problema, a atuação.

3.2.5: Atuação.

A atuação é o processo pelo qual o controle tem acesso a alguma alteração física no sistema de automação com o objetivo de alterar uma saída conforme leis de controle pré-definidas.

As informações de sensores que chegam até o controle através do envio e recepção de dados são processadas e geram sinais para a atuação. O elemento capaz de produzir atuação e denominado atuador.

Os atuadores convertem alguma forma de energia de entrada em movimento ou acionamento de mecanismos, causando mudanças de comportamento no sistema. Existem quatro tipos principais de atuadores, sendo eles:

- Atuadores hidráulicos.
- Atuadores Pneumáticos.
- Atuadores Elétricos.
- Atuadores mecânicos.

Um atuador hidráulico consiste basicamente de um motor que utiliza de energia hidráulica para facilitar operações mecânicas. Esta operação mecânica gera movimentos que podem ser lineares, rotatórios ou oscilatórios. Devido ao fato de que líquidos não podem ser comprimidos, um atuador hidráulico pode atingir forças muito altas, porém é limitado em velocidade e aceleração.

Um atuador pneumático converte energia proveniente de ar comprimido em altas pressões em movimento linear ou rotatório. Energia pneumática é empregada quando se deseja respostas rápidas de inicialização e parada de movimento, e tem como vantagem a fonte de energia ser renovável.

Um atuador elétrico é alimentado por energia elétrica e a transforma em um torque mecânico. É um dos tipos de atuação mais utilizado nas indústrias pela sua facilidade de instalação e diversidade comercial muito grande de modelos de atuadores diferentes.

Um atuador mecânico funciona através da conversão de energia rotatória em energia linear para executar movimento. Isto é alcançado com o uso de polias, engrenagens, correntes e outros dispositivos que gerem este movimento.

Na CASAN, a maioria dos atuadores é elétrico, salvo algumas exceções quando a carga a ser movimentada é extremamente grande, onde são empregados atuadores pneumáticos e hidráulicos.

Como neste projeto, o atuador a ser utilizado é do tipo motor elétrico, um pouco mais sobre este dispositivo é comentado no tópico abaixo.

3.2.5.1: Motor Elétrico.

Um motor elétrico é uma máquina capaz de gerar energia mecânica através da utilização de energia elétrica. No funcionamento normal de um motor, estes operam através da interação de um campo magnético e um campo elétrico causado por uma corrente elétrica, gerando uma força no motor.

Motores elétricos podem ser alimentados com corrente contínua (motores DC), tais quais baterias ou retificadores de onda, ou alimentados por corrente alternada (AC), tais quais suprimento de energia da rua, inversores ou geradores.

Talvez um dos primeiros motores elétricos foram simples dispositivos eletrostáticos criados pro um monge escocês chamado Andrew Gorgon em meados de 1740[16]. Estes dispositivos eram capazes de atrair ou repelir placas de metal mecanicamente. Com a descoberta da Lei de Ampére, por André-Marie Ampére em 1820, Michael Faraday conseguiu provar, utilizando os conceitos de Ampére, a conversão da energia elétrica em energia mecânica através da interação entre campos magnéticos e elétricos. Esta foi a base para o desenvolvimento dos primeiros motores elétricos, fundamentais na segunda revolução industrial.

Como principais partes do motor podem-se citar:

- Rotor.
- Estator.
- Enrolamento.
- Comutador.

O rotor é a parte que se movimenta de um motor, sendo que este é responsável pelo giro do eixo, entregando a energia mecânica ao processo. Nos rotores ficam geralmente localizados condutores que carregam correntes elétricas capazes de interagir com o campo magnético do estator, gerando o movimento. Em

alguns casos os rotores carregam os magnetos e os estatores que possuem os campos elétricos.

O estator é a parte estática do motor, nele se encontra os enrolamentos que conduzem a corrente elétrica geradora do campo elétrico.

Enrolamentos são compostos geralmente por fios metálicos enrolados ao redor de um metal imanizado, capaz de gerar polos magnéticos quando energizado com corrente.

Finalmente, o comutador é um mecanismo utilizado para alterar a entrada de alimentação de alguns motores AC e DC, fazendo com que o rotor altere seus polos magnéticos através da inversão da corrente. Na ausência de tal inversão, o motor poderia aumentar sua velocidade indefinidamente até sua falha ou quebra.

3.2.4.1.1: Alimentação do motor elétrico.

Motores DC são geralmente alimentados por fontes de tensão contínua alimentada por baterias, de maneira que a bateria deve ser escolhida com capacidade para drenar a corrente gerada pelo motor. Geralmente motores DC são motores de pouca potência, como pequenos motores de brinquedos, e nestes casos pilhas fornecem uma boa fonte de alimentação.

Motores AC possuem diversas alimentações distintas. Por exemplo, quando a tensão de alimentação proveniente do sistema elétrico ou de bateria é contínua, o motor AC é acionado através de inversores, que são equipamentos capazes de transformar ondas contínuas de corrente e tensão em uma onda alternada, necessária para o motor.

Quando o motor AC é de pequeno e médio porte, com poucos cavalos de potência, o acionamento pode ser realizado de forma direta, geralmente através de uma contatora bobinada, cortando a alimentação trifásica do motor quando sua bobina não está energizada e deixando passar a alimentação quando há energia em suas bobinas.

Para motores de maior parte, a corrente de partida pode causar danos ao motor, pois geralmente é muito maior que a corrente nominal, chegando a mais de cinco vezes este valor em uma partida seca. Nestes casos é necessária a utilização

de um equipamento denominado Soft-*Starter*, que não é nada mais do que um regulador de partida de motor, aumentando lentamente a alimentação do motor durante sua partida, desta maneira protegendo o mesmo contra as altas correntes.

Na CASAN, é muito comum ver implementações destes três tipos de partida AC, sendo difícil estipular qual é o mais utilizado.

Estipular atuadores não faz parte do projeto em si, pois o projeto deve abraçar os equipamentos já existentes nos pontos de possível implantação, com isso o mais importante é entender a alimentação destes motores, pois o controle é responsável por gerar estes comandos de acionamento. Mais sobre como acionar motores trifásicos AC com o Arduino ou o Xbee será apresentado no capítulo de implementação do projeto.

3.2.6: Medição.

Medição é o processo de aferir valores a objetos ou eventos. Qualquer medição pode ser julgada pelos três critérios básicos: Magnitude, dimensão (unidade de medida) e grau de incerteza. Com estes três critérios básicos pode-se comparar medidas realizadas por diferentes equipamentos e julgar a qualidade e a adequação dos mesmos ao projeto.

Apenas por curiosidade, antes da criação do SI (Sistema Internacional de Unidades), as primeiras medições eram realizadas utilizando padrões famosos como referência, como, por exemplo, o tamanho do antebraço de Faraós foi utilizado pelos primeiros engenheiros de pirâmides como unidades de medida. Mesmo com a adequação criada pelo SI, ainda existem muitos padrões diferentes aplicados em diversos países como Estados Unidos e Inglaterra.

Na automação, quando se fala em medição se fala indiretamente em sensores. Um sensor é um conversor capaz de medir uma quantidade física e convertê-la em um sinal que pode ser observado por um humano, ou na maioria dos casos por um controlador. As principais qualidades que se procura em um sensor para automação são:

- Sensível apenas as propriedades medidas.
- Insensível às variações do meio em que se encontra.

- Não influencia no processo ao qual está medindo.

É evidente que é impossível encontrar um sensor que contemple estas três qualidades totalmente, pois mudanças como, por exemplo, na temperatura, irão alterar o valor medido pelo sensor e não há como evitar. O que se busca então, ao selecionar sensores para um projeto, é o grau de incerteza dos mesmos, ou em outros termos, a precisão que o sensor atinge na medida quando dentro de uma faixa nominal de condições.

Geralmente sensores comerciais indicam a precisão para uma faixa de temperatura, pois esta é considerada um dos fatores externos mais influenciadores nas medias. Precisões de 5% parecem ser altas a princípio, mas às vezes em casos em que a qualidade do processo permite folgas pode ser uma opção muito mais robusta que sensores com precisão na casa dos décimos de por cento. Cabe ao projetista dos sistemas adequar o sensor ao seu projeto de maneira a atingir seus objetivos.

Outra característica importante dos sensores e que deve ser avaliada é a resolução dos mesmos. Resolução significa qual a menor variação no processo que será detectada pelo sensor e causará uma variação no sinal emitido pelo mesmo. Geralmente a resolução dos sensores é diretamente proporcional ao valor comercial dos mesmos, sendo que quanto maior a resolução mais caro é o equipamento. Mais uma vez a questão de adequação ao projeto é muito importante ao se dimensionar a resolução necessária para os sensores.

Para este projeto dois sensores são particularmente importantes, sendo eles o sensor de oxigênio dissolvido (OD) e o sensor de altura de coluna de água (nível). Mais sobre estes sensores é discutido nos próximos tópicos.

3.2.6.1: Sensor de Oxigênio dissolvido.

Um sensor de oxigênio dissolvido é um dispositivo elétrico ou eletrônico capaz de medir a proporção de Oxigênio (O_2) dissolvido em um líquido ou gás. A utilização mais comum destes tipos de sensores é para analisar a concentração de oxigênio nos gases de exaustão dos motores de combustão interna.

Para tratamento de efluentes, o sensor de oxigênio dissolvido geralmente é composto por uma sonda com uma membrana de PTFE, um polímero capaz de reagir com a presença do elemento químico. Esta reação gera um corrente na faixa de microampères, a qual é amplificada com a utilização de amplificadores operacionais para finalmente gerar o sinal de medição.

Na CASAN, os aeradores da ETE Insular utilizam estes sensores para mostrar aos operadores a quantidade de Oxigênio presente nos tanques de aeração, para que os mesmos possam tomar a decisão de ligar o motor que irá aerar o tanque ou não. Estes sensores já estão presentes no local e podem servir de entrada para o projeto presente neste trabalho.

3.2.6.2: Sensor de nível.

Os sensores utilizados na CASAN são do tipo transmissor de pressão submersível, capaz de ler o nível da coluna de água através da pressão exercida sobre uma sonda sendo que existem diversas opções de pressão nominal.

Segundo Gautshi[17], “estes sensores se baseiam na propriedade piezoelétrica do cristal de quartzo que, quando deformado elasticamente, gera um potencial elétrico em seus terminais por meio de certo plano cristalográfico”. A figura abaixo tenta explicar um pouco melhor o funcionamento:

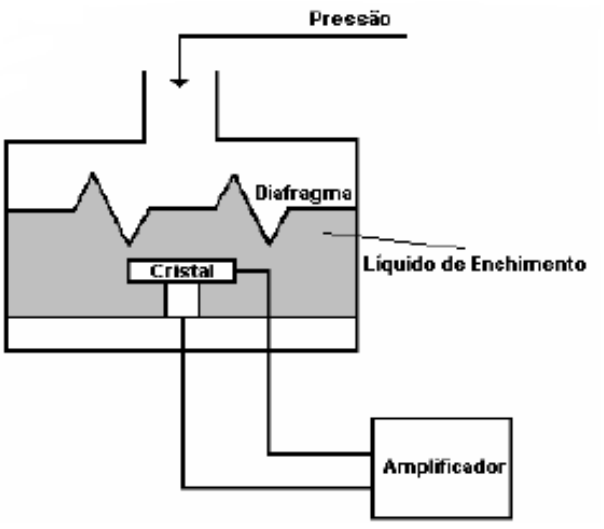

Figura 41- Funcionamento de um sensor transmissor de pressão.

Quando a pressão aumenta, o diafragma, mostrado acima, se distende, deformando o cristal de quartzo. Com este dispositivo, é possível operar num campo de frequência de 100kHz com linearidade de 1%, porém com uma tensão média na ordem de 1mV. Por isso sempre existe a associação com um amplificador operacional e dependendo se a leitura for em corrente, um circuito resistivo.

Para descobrir qual sensor utilizar, é necessário resolver a seguinte equação:

$$P = \mu\, g\, h$$

Onde P é a pressão em Pascal, μ a densidade do líquido e h a altura da coluna de água. Como as maiores Elevatórias de Esgoto da CASAN tem por volta de 7 metros de altura o h utilizado para escolha do sensor foi de 10 metros. Sabendo que a densidade do esgoto tem em média μ = 998,23 $\frac{kg}{m^3}$ podemos calcular a pressão como se segue:

$$P = 998{,}23 * 9{,}8 * 10$$

$$P = 97826{,}54\, pa = 0{,}98\, bar$$

Ou seja, um sensor de pressão nominal um bar é ideal para a situação das elevatórias de esgoto.

Vale lembrar que estes sensores geram sinais entre 4-20 mA em corrente, e os módulos Xbee operam com suas entradas analógicas em tensão de 0-1.2 V, ou seja, é necessário criar uma conversão de medidas antes de entrar com o valor nas portas analógicas do Xbee. Mais sobre as técnicas de solução deste problema serão apresentadas no capítulo de implementação.

3.3: Considerações Finais do capítulo.

Com isso foram apresentados neste capítulo toda a fundamentação e a base necessária para a construção deste projeto, iniciando com a apresentação do problema a ser resolvido, passando pelas teorias gerais de controle e se aprofundando um pouco nos elementos integrantes do sistema de automação deste trabalho.

Foram apresentados com um nível de aprofundamento os dois integrantes principais deste trabalho, o Arduino, com papel de controlador e o Xbee, baseado no protocolo Zigbee, que tem papel de transformar a solução em uma solução sem fio.

O objetivo principal era apresentar um pouco as tecnologias envolvidas no mesmo e motivar o trabalho em si, de maneira a fomentar o interesse pelas soluções presentes no documento.

No próximo capítulo será atacado o projeto de automação realizado, apresentando a metodologia e o caminho feito para se chegar na implementação que por fim será apresentada no capítulo 5.

Capítulo 4: Desenvolvimento.

Neste capítulo será apresentado como a solução final foi tomando sua forma, incluindo cada passo que foi dado em direção ao resultado, modelos preliminares, testes que levaram ao entendimento das tecnologias envolvidas, ferramentas auxiliares envolvidas e desenvolvimento de componentes necessários para a integração e funcionamento do sistema.

Além disso, na parte final do capítulo será apresentado o modelo proposto neste projeto, mostrando a nova maneira do processo problema funcionar, incluindo diagramas, modelos e arquiteturas envolvidas para que o projeto finalmente tome forma.

O objetivo deste capítulo é responder a pergunta: “como chegou ao objetivo?”, servindo de base para o capítulo seguinte que atacará a implementação da solução final. Este capítulo será dividido por tópicos, cada um deles focado em alguma parte específica da solução, em uma estratégia *bottom-up* de construção aonde cada parte menor vai se juntando com outras partes para formar o todo, mais complexo.

O primeiro elemento estudado na concepção do sistema de automação presente neste projeto foi o protocolo Zigbee e seus módulos Xbee, e é justamente neste ponto que este capítulo começa a seguir.

4.1: Desenvolvendo aplicações Zigbee.

Talvez por se tratar do diferencial deste trabalho, e por ser uma tecnologia que me era muito pouco familiar na época, a comunicação sem fio com redes Zigbee foi o que mais me chamou a atenção inicialmente. A metodologia utilizada para entender mais sobre o padrão se iniciou com a leitura de um livro teórico sobre o padrão, de Shalin Farahani[2].

Este livro ataca com muita profundeza o protocolo Zigbee, comparando com outras tecnologias sem fio, apresentando todos os detalhes do protocolo, mostrando exemplos de aplicações que já existem no mercado, além de comentar

extensamente sobre as necessidades de *hardware* que é preciso para a construção de um equipamento Zigbee.

Por se tratar de um livro teórico, o mesmo não apresenta informações sobre modelos de equipamentos existentes para criação de redes e nem apresenta exemplos de códigos e programação dos mesmos. Esta lacuna é suprida por outro livro de outro autor, explicados no próximo tópico.

4.1.1: Escolha de modelos de equipamentos.

Entendido o funcionamento do protocolo, deve-se observar que este em si é apenas a base teórica, fornecendo as regras para o funcionamento da rede Zigbee. A implementação real somente é possível de ser realizada com o uso de módulos que contém o protocolo Zigbee.

Como eu não tinha familiaridade alguma com equipamentos deste porte, procurei auxílio e embasamento em um livro recomendado por diversos fabricantes de módulos Zigbee sobre o assunto, de Robert Faludi[18].

Este livro, de caráter prático, apresenta inicialmente uma longa lista de *hardwares* comerciais do protocolo, incluindo módulos e periféricos com o padrão. Além de listar, cada equipamento é apresentado com parcimônia, mostrando detalhes de funcionamento e diferenças.

Foi através deste livro que eu conheci a Digi International, a maior fabricante do mundo de módulos deste tipo, e que, acabei por descobrir posteriormente, fornece praticamente exclusivamente todos os módulos Zigbee encontrados à venda no Brasil.

Encontrar comerciantes deste tipo de equipamentos no Brasil foi mais difícil do que o imaginado, sendo que praticamente todos estes se localizam no eixo Rio-São Paulo, distantes de onde foi implementado este projeto.

Escolhido um comerciante de São Paulo que trabalhava com os módulos Zigbee da Digi, denominados Xbee, a questão fundamental agora era a escolha do modelo. Literalmente existem dezenas de modelos diferentes, mas que podem ser filtrados seguindo o critério de perguntas na próxima página:

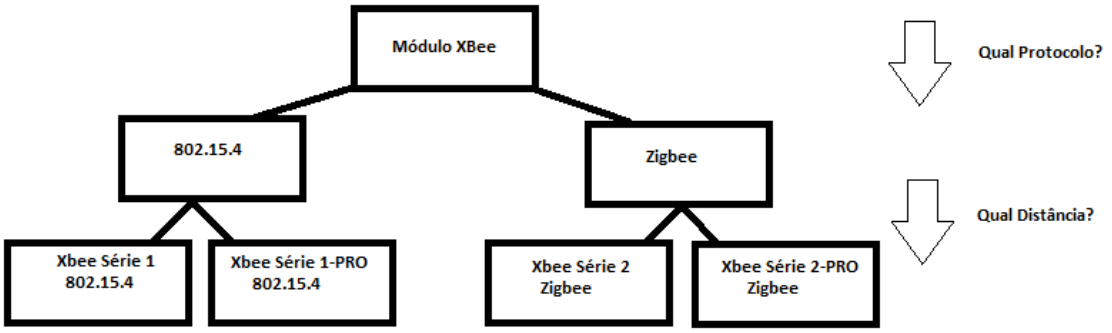

Figura 42- Critério de escolha de modelo Xbee.

A primeira pergunta ao se fazer é: “Qual o protocolo?”, pois módulos da Digi possuem versões puramente no protocolo 802.15.4 (módulos da série 1) e versões que utilizam o protocolo Zigbee (módulos da série 2). A segunda pergunta que se faz é: “Qual a distância?”, pois como visto em capítulos anteriores, módulos PRO conseguem um alcance maior em ambientes externos ao custo de gastarem mais energia na transmissão. Filtrado os resultados desta maneira, a única diferença entre os candidatos restantes é o tipo de antena.

Para este trabalho especificamente, foi escolhido os módulos da série 2-PRO Zigbee, primeiramente pelo fato de que nestes é possível configurar redes *Mesh* e trabalhar em modo API (*ApplicationProgramming Interface*). Mais sobre este modo será discutido nos próximos tópicos.

Segundo, a escolha do módulo PRO foi pelo fato de seu alcance ser muito maior em ambientes externos, ou seja, é mais adequado para plantas grandes, como no caso da ETE Insular, onde ficam os aeradores. No caso das EEEs, o módulo da série 2 convencional pode ser utilizado como alternativa mais econômica pois o módulo PRO é aproximadamente 60% mais caro.

Após a escolha dos módulos adequados, ainda é necessário obter um programador, ou seja, um *hardware* capaz de interligar o módulo Xbee com o *software* de programação do mesmo, que geralmente se encontra em um computador. Este programador é responsável por converter a interface serial do módulo Xbee para a interface USB, presente no computador.

Através do programador é possível realizar as seguintes funções:

- Gravar/atualizar o *firmware* dos módulos Xbee.
- Gravar parâmetros e configurar redes Xbee.
- Servir de interface entre aplicativos específicos no computador ou outros dispositivos e o Xbee.

Escolhidos e recebidos os equipamentos do comerciante, foi possível criar o primeiro exemplo de redes Zigbee utilizando os módulos Xbee, apresentado no próximo tópico.

4.1.2: Primeiro exemplo prático.

O primeiro exemplo prático de uma rede Zigbee é um projeto muito simples que consiste em um bate-papo entre dois dispositivos Xbee, sendo um coordenador e um dispositivo final. A figura abaixo apresenta o esquema proposto:

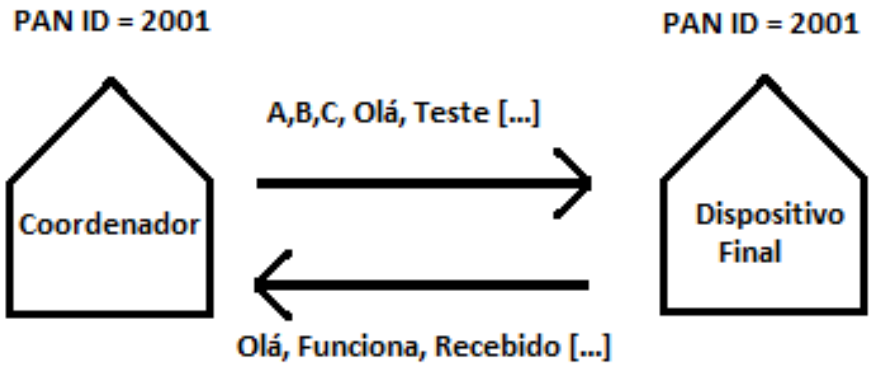

Figura 43- Bate-papo Zigbee.

Neste sistema muito simples, o coordenador (necessário em uma rede Zigbee), cria uma rede com o ID 2001, o dispositivo final entra nesta rede e após isso os dois dispositivos trocam mensagens entre si. Ligados a cada um dos dispositivos estará um computador o qual poderá imprimir na tela a mensagem que seu dispositivo Xbee conectado recebeu.

Para desenvolver este exemplo prático é necessário programar os dispositivos. Para os módulos Xbee, a Digi disponibiliza um *software* gratuito onde é possível realizar esta configuração. O *software* se chama X-CTU, que em Julho de 2013 se encontra na versão 5.2.8.6 (fonte: Digi International). Uma captura de tela do X-CTU é apresentada na próxima página:

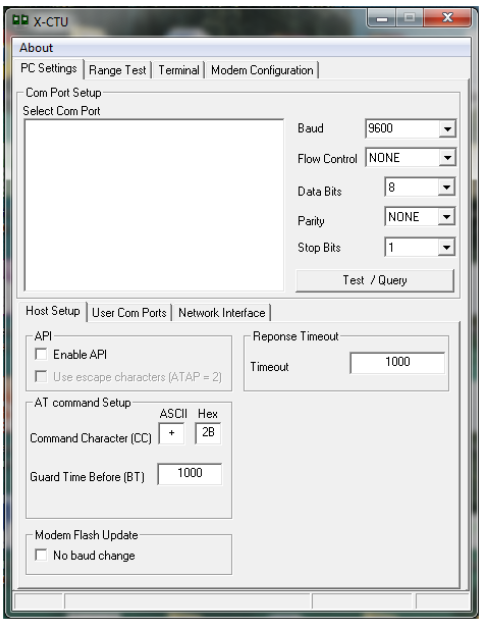

Figura 44- X-CTU.

Neste programa é possível alterar dados como taxa de transmissão, controle de fluxo, número de bits de dados por byte, paridade e bits de parada. Também é possível habilitar o modo API e tempo de *timeout* do próprio *software*. Para este simples exemplo, estes campos não precisam ser alterados e ficarão com seus valores padrão.

Ao se adaptar o módulo Xbee através do programador no computador, o X-CTU retornará uma mensagem do tipo *pop-up* abaixo:

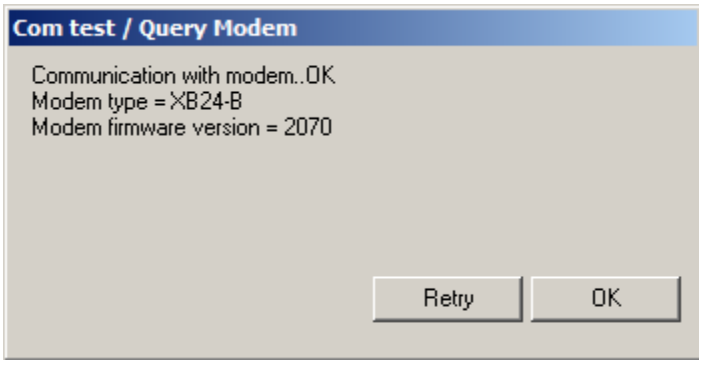

Figura 45- Conectando módulo ao X-CTU.

Esta caixa de diálogo avisa ao usuário que o X-CTU consegue se comunicar com o módulo e ainda identifica o tipo e a versão de *firmware* do mesmo. Os módulos da Digi já vêm de fábrica geralmente com o *firmware* mais atualizado para a versão do módulo, mas às vezes é necessário fazer o *update*, através do recebimento de arquivos do próprio sitio da Digi.

Com o módulo conectado ao X-CTU, já é possível finalmente configurar o mesmo para o exemplo deste tópico. Para isso deve-se clicar na aba *Modem Configuration*, visível no canto direito alto da figura 44. Ao clicar nesta tela, a seguinte tela é apresentada ao usuário:

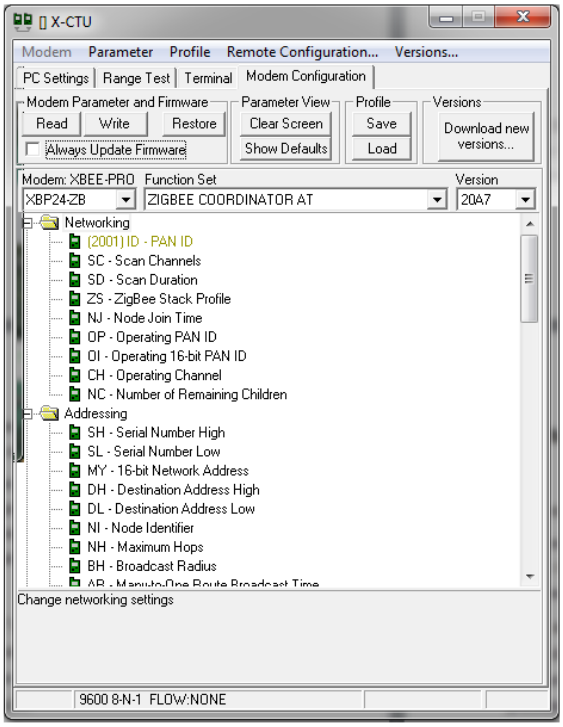

Figura 46- Configuração dos módulos.

Nesta tela é que a configuração dos dispositivos Xbee é feita. O botão *Read* é o responsável pela leitura das configurações atuais do módulo, enquanto que o botão *Write* é o responsável pela escrita dos parâmetros no módulo. O botão *Restore* é capaz de reverter as alterações no módulo e retornar ele à configuração padrão de fábrica.

No menu Function Set é onde o usuário escolhe a função que o módulo deve exercer na rede. Neste caso, o módulo configurado irá ser um coordenador com protocolo Zigbee AT e um PAN ID de 2001 (mesmo da figura 43).

Os dispositivos Xbee podem atuar em dois modos diferentes, AT e API e estes se referem a maneira em que o dispositivo usa sua conexão serial local. Módulos configurados no modo API utilizam dados empacotados no padrão Zigbee que são muito úteis quando dois dispositivos de mesmo tipo se comunicam. Já Xbees configurados em modo AT repetem na sua conexão serial exatamente os mesmos dados no mesmo formato que recebem. Como neste exemplo o objetivo é que seja impresso na tela exatamente o mesmo caractere que o dispositivo Xbee recebeu, o formato AT é mais interessante. Vale lembrar que este formato tem caráter exemplificativo e é praticamente inutilizado em projetos de automação e somente será apresentado em exemplos simples, sendo que toda a implementação final será baseada em modo API.

Neste exemplo só é necessário configurar três parâmetros presentes na tela da figura 46, que são o PAN ID, o Destination Adress High e Destination Adress Low. Estes dois últimos campos devem coincidir com o endereço físico de 64 bits presente em cada módulo Xbee. Este endereço está impresso na parte posterior de cada dispositivo Xbee e é único, conforme mostra a figura abaixo:

Figura 47- Endereço de 64 bits.

Na figura acima, é possível ver que o campo Destination Adress High neste caso deve receber o valor 0013A200 e o campo Destination Adress Low o valor 403B9E21. Resumindo, o coordenador neste caso deve receber as seguintes configurações:

- Function set: Zigbee coordinator AT.
- PAN ID: 2001.
- Destination Adress High: 0013A200.
- Desntination Adress Low: 403B9E21.

Após este passo é necessário apenas clicar no botão *Write* e esperar as configurações serem transmitidas ao módulo coordenador. Terminado este passo o módulo coordenador pode ser desconectado do programador e o módulo dispositivo final pode ser conectado, pois o mesmo também necessita de programação.

O dispositivo final, para este exemplo, deve receber as seguintes configurações:

- Function set: Zigbee end device AT.
- PAN ID: 2001.
- Destination Adress High: Endereço do módulo coordenador.
- Destination Adress Low: Endereço do módulo coordenador.

Com isso, pode-se então finalmente conectar cada um dos módulos em um computador diferente através do programador e abrir a aba Terminal, no *software* X-CTU. O terminal, como o próprio nome já diz, é uma aplicação que apresenta na tela dados recebidos serialmente e insere na sua serial dados digitados no mesmo. O resultado em ambos os computadores é o seguinte:

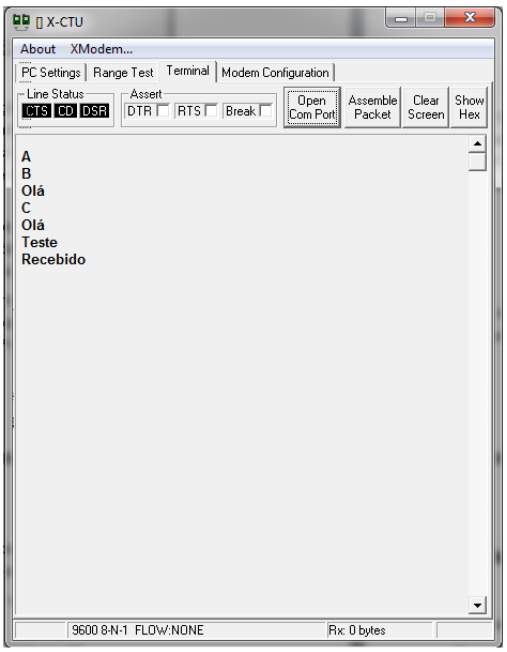

Figura 48- Resultado do exemplo.

Embora este exemplo seja muito simples, serviu para iniciar minha experiência na programação de dispositivos Zigbee e gerou um primeiro resultado parcial que pude apresentar para meus companheiros de empresa, gerando curiosidade e desejo por avançar para problemas mais complexos. Além disso, este exemplo serve neste documento para apresentar a programação de módulos Xbee, sendo que no capítulo de implementação final estes conceitos serão utilizados continuamente.

Com o primeiro exemplo realizado, aumentar a complexidade dos exemplos gerou um problema geral que apresenta as “limitações” do Xbee, que é a ausência de controle. Coloco limitações entre parênteses para enfatizar o fato de que os módulos Xbee não foram projetados para serem controladores, e não é seu papel em uma rede sem fio. Para tal é necessário inserir no sistema um dispositivo controlável, capaz de processar dados e gerar sinais. Este dispositivo pode ser um CLP ou um Arduino, por exemplo. A escolha deste projeto foi pelo Arduino, e um pouco mais sobre o desenvolvimento deste é apresentado no próximo tópico.

4.2: Desenvolvendo aplicações Arduino.

O Arduino é uma plataforma de controle integrada constituída por um microcontrolador, contatos de entrada e saída e uma fonte de alimentação, formando uma solução integrada para sistemas de automação.

Ao se iniciar este projeto, minha experiência com o Arduino era mediana, sendo que já havia desenvolvido algumas aplicações em Arduino na minha vida acadêmica. A metodologia utilizada para meu desenvolvimento no assunto foi semelhante à do Zigbee, sendo que fui buscar na literatura uma fonte de embasamento teórico para que eu pudesse criar exemplos práticos desenvolvendo os conceitos. Para tal utilizei o principal e mais vendido livro do mundo sobre o assunto, de Michael Margolis[1].

Este livro, extremamente popular entre desenvolvedores Arduino, apresenta o *hardware* Arduino e suas configurações bem como o ambiente de desenvolvimento de programas (IDE) também chamado de Arduino. Detalhes sobre sua

programação, tipos de variáveis aceitas, operadores matemáticos, comunicação serial, entradas e saídas analógicas e digitais, criação de bibliotecas e periféricos extras são apresentados. Um conteúdo muito extenso, mas que salva muito tempo de desenvolvimento das aplicações.

4.2.1: Escolha de modelo de equipamento.

Passada a fase de embasamento teórico, era hora de comprar o Arduino. Este, assim como o Xbee, também possui dezenas de versões distintas, cada uma com alguma funcionalidade distinta da outra.

A produção oficial do Arduino está nas mãos de duas empresas no mundo, a Italiana Smart Projects e a americana SparkFun Eletronics. No Brasil, historicamente a aceitação é maior das placas produzidas na Itália, por isso os revendedores geralmente apenas compram modelos da Smart Projects. Algumas lojas especializadas em Santa Catarina trabalham com venda de modelos de Arduino, mas, novamente, o eixo comercial principal se encontra no Rio de Janeiro e em São Paulo.

Escolhido um fornecedor paulista, a questão agora era qual modelo selecionar. Geralmente os modelos de Arduino se concentram em dois grandes grupos: Os Arduinos de 32 pinos e os Arduinos de 100 pinos. Neste projeto é necessária (como será visto mais adiante) a utilização de duas seriais para comunicação e teoricamente isto limitaria o modelo exclusivamente par os com 100 pinos, pois os de 32 pinos possuem apenas um canal de comunicação serial. Devido a um remanejo através de *software* foi capaz de utilizar dois canais seriais em um dispositivo de 32 pinos, e por isso este foi escolhido para o projeto, além de ter um custo de praticamente um terço do Arduino de 100 pinos.

Entre as versões de 32 pinos, a mais utilizada e mais econômica de todas é o Arduino Uno, atualmente na sua versão revisada 3 e estes motivos o levaram a ser escolhido para este projeto.

4.2.2: Ambiente de Desenvolvimento Arduino.

A programação do Arduino é criada no Ambiente de Desenvolvimento Arduino, um *software* gratuito disponível no sitio do Arduino. Em Julho de 2013 este *software* estava na versão 1.5.2 e uma captura de tela é apresentada abaixo:

Figura 49- IDE Arduino.

A IDE do Arduino trabalha em um dialeto C/C++ em que a maioria das funções destas linguagens está implementada, porém a maioria das bibliotecas destas não é compatível com o Arduino, principalmente pela sua pouca memória RAM.

No ambiente é possível criar e compilar códigos a serem escritos no Arduino, além de criar bibliotecas com chamadas de funções possíveis de ser realizadas pelo Arduino.

Para poder gravar o código no Arduino, o mesmo necessita estar conectado ao computador que contém o *software*. Para isso, basta retirar a alimentação de fonte do Arduino e conectar o mesmo através de um cabo USB-Mini USB no computador. O Arduino já vem com uma entrada Mini-USB, como pode ser visto na figura 20, facilitando a operação.

Para fazer o *upload* de códigos basta então selecionar o segundo botão da esquerda para a direita na imagem 49, com uma seta.

4.2.3: Exemplos de aplicação.

Para melhor entender o funcionamento do Arduino e como criar códigos para o mesmo, uma ferramenta da própria IDE foi fundamental neste projeto. Na aba $file \rightarrow examples$ existem por volta de 50 exemplos das mais diversas funcionalidades do Arduino, desde simples exemplos de teste de LEDs piscando até exemplos complexos como comunicação Ethernet TCP/IP. Os exemplos que foram utilizados para melhorar a compreensão e ajudar neste projeto foram os de comunicação serial, recebimento de dados e exemplos de controle baseado nestes mesmos.

Um exemplo que serve como inicialização a IDE do Arduino e possui alguns métodos utilizados na solução final é apresentado abaixo, denominado Evento Serial:

```
String inputString = "";         // string que armazena dados receb
boolean stringComplete = false;  // flag que indica se a string está completa

void setup() {
  // chamada de inicialização de serial.
  Serial.begin(9600);
  // reservando 200 bytes para a string
  inputString.reserve(200);
}

void loop() {
  // é o loop principal do programa
  if (stringComplete) {
    Serial.println(inputString); //imprime string
    // limpa a string
    inputString = "";
    stringComplete = false;
  }
}


void serialEvent() {
  while (Serial.available()) { //quando existe algum dado na serial
    // lê o caractere
    char inChar = (char)Serial.read();
    // adiciona o caractere na string
    inputString += inChar;
    // se o próximo caractere for uma nova linha então pode imprimir na tela a string
    if (inChar == '\n') {
      stringComplete = true;
    }
  }
}
```

Figura 50- Código do exemplo.

Primeiramente, é possível notar a semelhança da linguagem de programação com C/C++, como dito anteriormente. Em laranja ficam marcadas as funções definidas por bibliotecas padrão do Arduino, sendo que é possível utilizar estas ao invés de ter que criar cada uma das funções para cada aplicação.

A IDE do Arduino trabalha com definições globais, definições locais e rotinas. A primeira rotina é o *setup*, executado apenas na inicialização da aplicação e cada vez que a mesma for reiniciada. Tudo que for declarado antes desta parte é considerado como variável global e pode ser utilizado por qualquer função do código, como, neste exemplo, a $String\ inputString$ e o booleano $StringComplete$.

A segunda rotina é a rotina cíclica principal, denominada $loop$. É nela que se encontra o núcleo do código a ser executado e o Arduino fica lendo suas instruções sequencialmente até o fim da rotina, para então retornar ao início da mesma.

Existem duas maneiras de se romper esta sequência da rotina principal, uma sendo a chamada de interrupção do sistema, e outra sendo a chamada local de uma função.

Na interrupção do sistema, algum evento definido pelas bibliotecas do Arduino causa que este interrompa sua ordem natural de processamento e trate a interrupção. No exemplo acima, a função $SerialEvent$ é uma destas interrupções que avisam ao processador a chegada de um caractere na porta serial. O Arduino trata esta interrupção e retorna para o ponto em que parou na rotina principal.

Na chamada local de função, o programador faz com que a rotina principal deliberadamente chame outra função através da passagem ou não de parâmetros para atingir um objetivo ou requisitar alguma informação para seu processamento. O *software* da solução final deste projeto utilizará este conceito.

O programa da figura 50 atua da seguinte maneira: Primeiramente, este cria variáveis auxiliares necessárias e depois faz o *setup* da comunicação serial. O ciclo principal apenas fica testando se é para imprimir algum dado na tela e a chamada de interrupção trata o evento de digitação de um caractere, incluindo na *string* a ser impressa.

Os resultados podem ser observados no próprio IDE do Arduino na aba $Tools \rightarrow Serial\ Monitor$ depois de conectado e programado o equipamento ao computador, conforme figura abaixo:

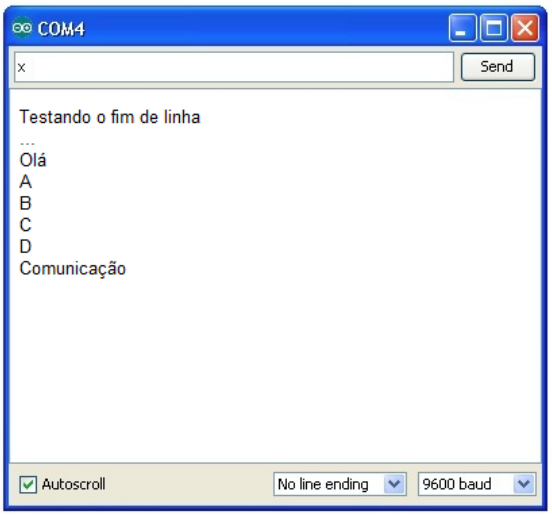

Figura 51- Resultado do exemplo.

A caixa contendo o caractere “x” é o local onde o usuário pode inserir sua mensagem que será impressa no terminal (abaixo) ao se quebrar a linha (pressionar *Send*).

O objetivo aqui não é estudar o código exemplo profundamente, isto será feito com o código principal deste trabalho, mas sim dar uma ideia de como funciona a IDE do Arduino e de como fui aplicando exemplos cada vez mais complexos para entender o processo de geração, compilação e execução de códigos Arduino.

Unindo a capacidade de controle e processamento do Arduino com a possibilidade de comunicação sem fio do Xbee, algumas aplicações bem amis interessantes podem ser realizadas e mais sobre este tema é discutido no próximo tópico.

4.3: Integrando Arduino e Xbee.

Passadas as criações de exemplos tanto com Xbee quanto com Arduino, o próximo passo é justamente interligar as duas tecnologias para criar aplicações mais complexas. Interligar um Arduino com um módulo Xbee requer a conexão de quatro pinos conforme abaixo:

Xbee	Arduino
Vcc (pino 1)	3v3
DOUT (pino 2)	Rx
DIN (pino 3)	Tx
GND (pino 8)	GND

Tabela 7- Ligação entre Xbee e Arduino.

Como pode ser visto, o Arduino pode servir de fonte de alimentação para o Xbee, pois este tem conectores de saída de 3.3 V, que é justamente a tensão de funcionamento nominal do Xbee. Os outros dois fios são a conexão direta entre as portas seriais de cada equipamento. Como ambos trabalham com serial TTL, a ligação é direta e utiliza apenas dois fios.

É extremamente recomendado não soldar diretamente fios nos pinos do Xbee, pois o sobreaquecimento do equipamento poderá causar danos e até perda do mesmo. Uma solução fácil para resolver este problema é a utilização de placas de conexão. Estes são dispositivos que além de criar um canal de acesso indireto aos pinos do Xbee, podem ainda transformar o espaçamento dos pinos do equipamento (de 2 mm) em um padrão mais comum, como o de 0.1 polegadas, utilizado na maioria das protoboards. A figura abaixo apresenta o Xbee posicionado sobre a placa de conexão:

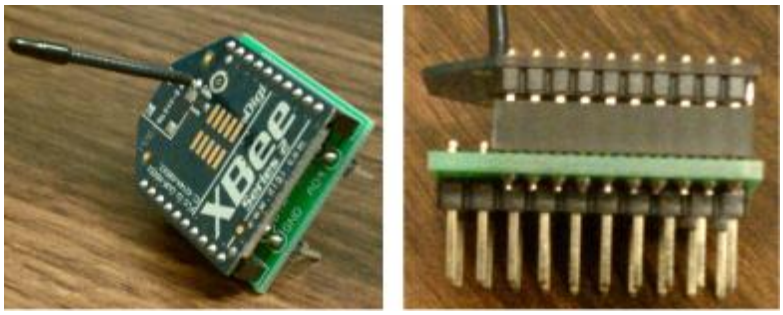

Figura 52- Placa de conexão Xbee.

É possível ver na figura a diferença entre o espaçamento de pinos do Xbee (parte superior da figura à direita) e o da placa de conexão (abaixo do mesmo). Com

a utilização desta placa de conexão, uma representação da ligação entre Xbee e Arduino é apresentada ABAIXO:

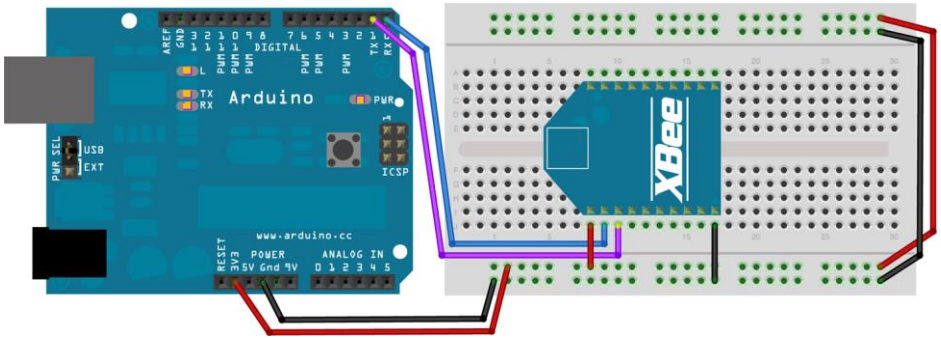

Figura 53- Conexão Arduino-Xbee.

A utilização da protoboard neste caso facilita a alteração de alguma ligação e a inserção de outros componentes no circuito, sendo geralmente utilizada até se encontrar a versão final do sistema, quando por fim é abandonada.

4.3.1: Exemplos de aplicação.

Interligados os dois equipamentos, foram realizados vários testes e exemplos de aplicações para verificar o funcionamento e entender as características técnicas de cada programação. O leque de possibilidade aumenta muito ao se adicionar a capacidade de processamento de um com a possibilidade de criação de redes sem fio do outro. Um exemplo que me auxiliou muito neste processo de aprendizado foi o clássico exemplo da Campainha do Arduino, o qual eu adaptei para uma versão mais completa, utilizando o Xbee. Neste exemplo, dois sistemas distintos sem qualquer ligação física por fiação irão se comunicar sendo que um faz papel de botão e outro de campainha. O esquema abaixo ajuda a entender melhor a situação:

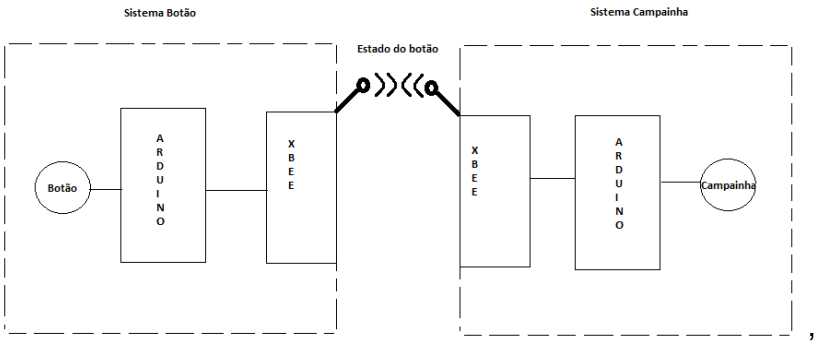

Figura 54- Esquema do exemplo de aplicação.

Na figura, o botão é ligado a uma entrada digital do Arduino do Sistema Botão, e este por sua vez é ligado com um Xbee conforme apresentado anteriormente. Este Xbee então forma uma rede Zigbee com outro Xbee do Sistema Campainha que recebe a informação e o Arduino deste sistema consegue ler a informação na sua porta serial, acionando ou não a campainha conforme decisão.

O primeiro passo deste sistema é configurar os módulos Xbee para formar a rede sem fio. A configuração do Xbee do Sistema botão é a seguinte:

- Function set: Zigbee coordinator AT.
- PAN ID: 2001.
- Destination Adress High: Endereço do Xbee Campainha.
- Desntination Adress Low: Endereço do Xbee Campainha.

A configuração do Xbee Campainha é a seguinte:

- Function set: Zigbee end device AT.
- PAN ID: 2001.
- Destination Adress High: Endereço do Xbee Botão.
- Destination Adress Low: Endereço do Xbee Botão.

Programados os Xbee, presta-se então atenção na programação dos Arduinos. O código do Arduino botão é apresentado abaixo:

```
int botao = 2;
void setup() {
pinMode(botao, INPUT);
Serial.begin(9600);
}
void loop() {
  // envia um caractere "D" se botao pressionado
  if (digitalRead(botao) == HIGH) {
Serial.print('D');
delay(10); // previne sobrefluxo de dados
}
}
```

Figura 55- Código do Arduino botão.

Este código, bastante simples, cria uma variável do tipo inteiro $int\ botao$ e utiliza esta variável na rotina de *setup* para habilitar o pino dois como entrada digital e inicia a comunicação serial com uma taxa de 9600 kbps. A função principal *loop*

examina esta entrada digital e se ela estiver ativa o mesmo escreve na serial o caractere “D”, escolhido arbitrariamente.

Já o código do Arduino Campainha é o seguinte:

```
int campainha = 5;
void setup() {
pinMode(campainha, OUTPUT);
Serial.begin(9600);
}
void loop() {
// procura por um caractere D na sua porta serial
if (Serial.available() > 0) {
if (Serial.read() == 'D'){
//toca a campainha rapidamente
digitalWrite(campainha, HIGH);
delay(10);
digitalWrite(campainha, LOW);
}
}
}
```

Figura 56- Código do Arduino Campainha.

Este código cria uma variável tipo inteiro $int\ campainha$ e utiliza esta variável na rotina de *setup* para habilitar o pino cinco do Arduino como saída digital e inicia a comunicação serial com a mesma taxa de 9600 kbps. A função principal *loop* é responsável pela leitura da porta serial e se ela encontrar o caractere “D”, o Arduino colocará sua saída digital em alto, tocando a campainha.

O esquema da ligação física do sistema é apresentado abaixo:

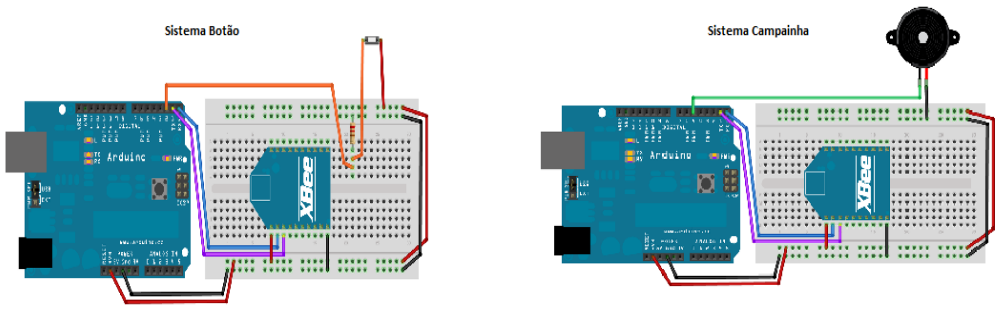

Figura 57- Ligação física do exemplo.

Este exemplo funcional é muito simples e serviu para poder demonstrar um pouco mais das possibilidades de implementação que estes equipamentos fornecem, colocando na forma prática e visível um pouco do potencial apresentado

teoricamente até agora. Para aplicações mais complexas, como a solução final proposta por este trabalho, é necessário entender o funcionamento do modo API do Xbee, pois este modo permite criar uma rede verdadeiramente Zigbee e aumentar muito as possibilidades de ação. O capítulo 5 apresentará estes conceitos na solução final, onde o tema será retomado.

Com a implementação deste exemplo e de outros de mesmo teor técnico, foi decidido então que o conhecimento básico dos componentes e de sua programação foi alcançado, podendo-se prosseguir para o próximo passo, estudar e entender a IHM.

4.4: Aprofundando os conhecimentos sobre a IHM.

A IHM, Interface Homem-Máquina tem como principal característica fornecer um ambiente de fácil entendimento e acesso a informação de processos pelo operador humano, sendo a ferramenta pela qual o mesmo interage com o sistema.

Em relação a este projeto, o objetivo para esta parte era se apoderar dos equipamentos que já eram utilizados pela CASAN, pois ao se concretizar, o período de aprendizagem por parte dos operadores seria menor devido ao fato de já estarem acostumados com o equipamento, seus botões e características técnicas.

Na CASAN, uma das IHMs utilizadas é a H-T80c da fabricante Beijer, localizada em Utah, Estados Unidos. Como existia um modelo deste prontamente disponível para uso em meu projeto, a escolha foi exatamente esta. No próximo tópico estarão descritas algumas de suas características técnicas.

4.4.1: Beijer H-T80c.

A Beijer H-T80c é uma IHM da série H, de características adequadas para a maioria das aplicações, com grande flexibilidade de uso. A figura abaixa apresenta o modelo utilizado neste projeto

:

Figura 58- IHM Beijer H-T80c.

Entre as características da IHM H-T80c pode-se citar:

- Tela colorida de oito polegadas com painel de toque.
- Botões laterais de acesso.
- Interface de comunicação RS-232, RS-485 e RS-422.
- Interface Ethernet.
- *Drivers* para os principais controladores do mercado.
- Programação gráfica.
- Biblioteca de símbolos.
- Suporte a Modbus RTU.

Quanto à alimentação, a IHM deve receber uma tensão de 24V DC e consome uma potência de 20 W.

Ainda presentes na parte posterior da IHM estão 10 chaves de ajuste que devem ser reguladas conforme a aplicação para seu melhor funcionamento. São elas (tabela na próxima página):

Chave	Significado
SW 1	Reservado
SW 2	Reservado
SW 3 + SW 4	Ligado + Ligado = Roda aplicação do usuário Ligado + Desligado = Roda teste interno Desligado + Ligado = Autaliza a BIOS Desligado + Desligado = Roda teste de bancada
SW 5	Ligado = parâmetros de comunicação ajustados na tela do terminal Desligado = parâmetros de comunicação ajustados via software
SW 6	Ligado = senha ativada Desligado = senha desativada
SW 7	Ligado = menu do sistema habilitado Desligado = menu do sistema desabilitado
SW 8	Ligado = Nível minimo de usuário Desligado = Nível máximo de usuário
SW 9	Reservado
SW 10	Ligado = Porta COM2 em RS-485 Desligado = Porta COM2 em RS-232/RS-422

Tabela 8- Chaves da IHM.

Estas chaves são as responsáveis por algumas das configurações vitais para o projeto, sendo que no capítulo de implementação elas serão comentadas novamente.

4.4.2: Programação da IHM.

A programação das IHMs da Beijer é feita através de um *software* proprietário e de custo levemente elevado chamado H-Designer, exclusivo para ambiente Windows. A figura na próxima página apresenta uma captura de tela deste programa:

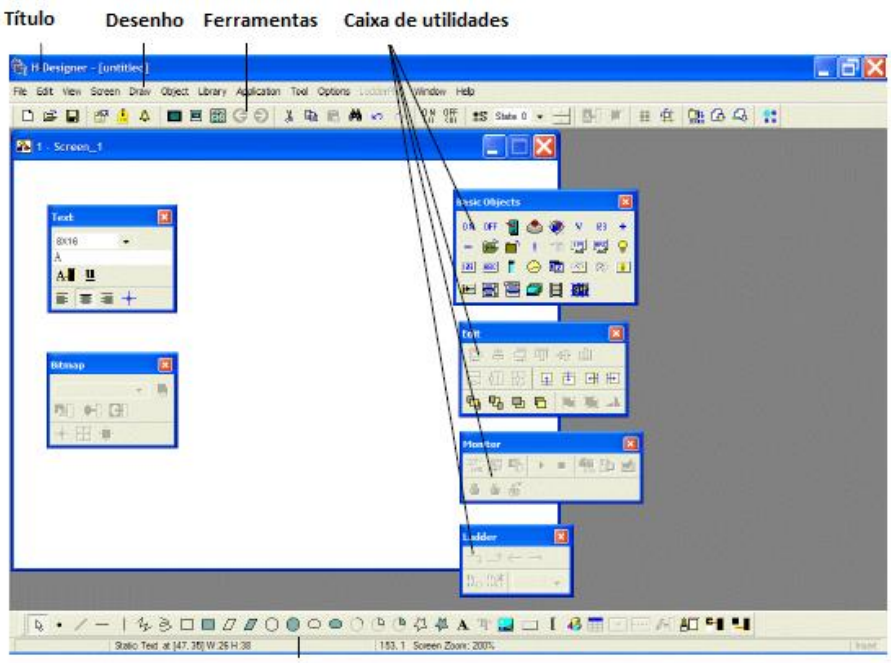

Figura 59- H-Designer.

O propósito aqui não e entrar em detalhes do *software* em si, para isto existem manuais no sítio da Beijer contendo todas as informações necessárias, sendo que este é apenas uma ilustração do procedimento.

Vale citar que é possível criar representações gráficas das mais diversas, telas diferentes para cada processo, menus animados, escrita e leitura de parâmetros do controlador, entre outros. Além disso, é possível compilar a aplicação para procurar erros e fazer a simulação do projeto, em um terminal criado virtualmente pela própria ferramenta.

Depois de criada a aplicação gráfica no H-Designer, o procedimento para realizar o *download* da aplicação para a IHM é bastante simples, apresentado abaixo:

1. Deixar o SW7 na posição ligada.
2. Ligar a IHM.
3. Conectar a IHM ao computador através de um cabo RS-232 macho-fêmea.
4. No H-Designer, ir à aba $Application \rightarrow Download\ Firmware \rightarrow Application$ e clicar.
5. Esperar a IHM avisar o fim de recebimento de configuração.

Embora o procedimento seja simples, um elemento fundamental para a realização do processo não vem junto com a IHM em sua embalagem, que é o cabo RS-232 macho-fêmea. Em meu projeto tive que criar e fabricar um cabo deste tipo que atendesse a especificação da IHM e foi surpreendente o tempo que o mesmo tomou devido à falta de informação sobre a configuração do mesmo pelo próprio fabricante, mais interessado em vender o mesmo produto pronto como um periférico extra de valor exageradamente alto.

4.4.3: Interligando a IHM ao controlador.

Como dito em suas especificações técnicas, a IHM utilizada neste projeto possui *drivers* para a maioria dos controladores industriais presentes no mercado, entre eles:

- CLPs da série 5, SLC, IQ da Allen Bradley (bastante comum nos Estados Unidos).
- Delta DVP.
- ERO TFS.
- Festo FPC.
- Fuji NB.
- GE-Fanuc.
- Hitachi EC.
- LG série K.
- Matsushita.
- Mitsubishi.
- Siemens Simatic.

Analisando a lista acima, é evidente que o usual, em projetos em que a IHM é utilizada, é interliga-la com algum CLP industrial e por isso que grande parte dos fabricantes líderes de mercado está presente na lista.

Este trabalho tenta mudar um pouco esta prática comum de soluções em automação, utilizando como controlador o Arduino, um dispositivo mais econômico que os controladores lógicos programáveis disponíveis comercialmente.

Outra vantagem do Arduino é sua flexibilidade de programação, muito maior que em CLPs industriais, pois se trata de uma plataforma livre, em que praticamente todo seu núcleo de rotinas e funções pode ser alterado para se adaptar ao projeto.

A decisão entre utilizar um CLP ou um controlador como o Arduino em um projeto depende de muitos fatores, sendo um deles a capacidade. Em processos em que é exigida uma velocidade de processamento muito rápida com dezenas ou até centenas de variáveis envolvidas (entradas e saídas analógicas e digitais) o CLP é a melhor solução pelas limitações técnicas do Arduino. Em processos em que apenas algumas entradas e saídas estão envolvidas e o custo é fator decisivo, o Arduino é uma melhor escolha.

Infelizmente, por não ser uma escolha muito comum, a IHM utilizada no projeto, bem como praticamente todas as outras disponíveis no mercado, não possuem suporte direto ao Arduino, sendo necessárias algumas manobras para implementar a comunicação, discutido abaixo.

4.4.3.1: Configurando a IHM.

Para a IHM se comunicar com o Arduino, uma das soluções é a utilização de um *driver* de configuração padrão da IHM, denominado “Modbus Master/Slave”, ou traduzindo, Modbus Mestre/Escravo.

Nesta configuração a IHM irá formar uma rede Modbus RTU *half-duplex* (discutida no tópico 3.2.3.4) onde ela é o mestre e o Arduino, ou qualquer outro dispositivo, seu escravo.

Na rede acima, os dados repassados de um dispositivo a outro são chamados de registradores, e tem um formato de *word* (16 bits). Cada registrador recebe um número de identificação, sendo que este pode assumir um valor entre 0 e 65535.

O canal de comunicação desta rede pode ser via cabo RS-232, RS-485 ou RS-422, sendo necessário alterar as chaves da IHM para o formato de comunicação correto.

A taxa de transmissão desta rede pode ser entre 9600 e 115200 kbps, sem paridade, um bit de início e um bit de parada por byte de dados enviado. Através da

chave SW5 estes parâmetros podem ser definidos na própria IHM ou através da utilização do H-Designer.

O Arduino possui suporte a redes Modbus através da implantação de bibliotecas específicas para o caso, e o estudo destas bibliotecas será apresentado no capítulo de implementação. Porém, o componente faltante para a interligação dos dois dispositivos é a falta de um canal de comunicação RS-232, RS-485 ou RS-422 no Arduino.

Como foi apresentado anteriormente, o Arduino possui canal de comunicação serial TTL e não há como interligar diretamente uma interface deste tipo com uma interface RS-485, por exemplo. Para tal é necessário um conversor destas duas interfaces, servindo de elemento de mediação e tradução entre os dois dispositivos.

4.4.3.2: Conversor RS-485/Serial TTL.

Como Arduino e IHM estarão configurados em uma topologia Mestre/Escravo, a comunicação *half-duplex* supre as necessidades do sistema, onde o mestre requisita ou envia alguma informação ao escravo e depois de realizada esta comunicação o escravo responde diretamente ao mestre. A escolha do RS-485 em relação ao RS-232 e o RS-422 foi pelo fato de se reduzir um fio em relação aos outros, além de alcançar distâncias muito maiores que o RS-232.

Por ser livre, existem diversos entusiastas do Arduino que criam e distribuem projetos para a plataforma. Existe no Brasil, uma empresa especialista nestas modificações, o Laboratório de Garagem, com sede na cidade de São Paulo. No sitio da empresa existe uma versão comercial que realiza a conversão de Serial TTL para RS-485, com o objetivo de comunicar o Arduino aos computadores que não possuem entradas USB ou que possuem todas elas já utilizadas por outros periféricos. Embora não tenha sido concebido especificamente para conexão com uma IHM, o princípio de funcionamento do dispositivo garante o funcionamento com este objetivo. A figura na próxima página apresenta o modelo do conversor disponível para compra no sítio:

Figura 60- Conversor RS-485/Serial TTL.

Neste trabalho, porém, não foi utilizado este dispositivo em questão. O criador desta placa, propagando os princípios da liberdade proposta pelos fundadores do Arduino, distribuiu para todos os usuários o esquema elétrico e de ligação dos componentes presentes na placa, para que cada um pudesse implementar sua própria versão caseira com as mudanças necessárias para aplicações específicas, dando a possibilidade de escolha em comprar a versão montada ou fazer a sua própria. O esquema abaixo é o esquema elétrico apresentado pelo criador e disponível para consulta:

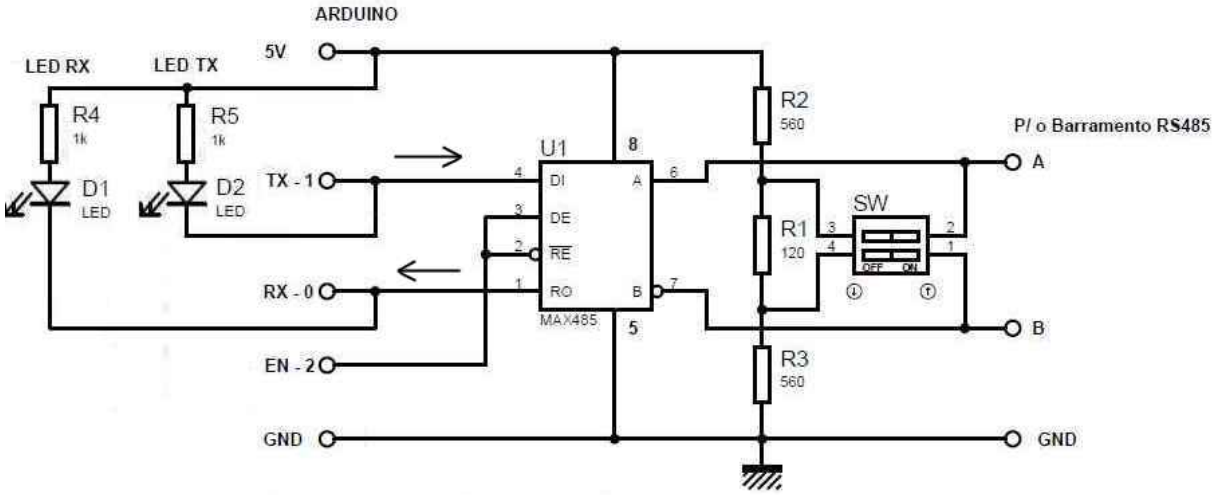

Figura 61- Ligação do conversor.

O circuito acima utiliza um circuito integrado MAX 485, especializado para conversão de serial para RS-485. Ele recebe do Arduino cinco entradas. Uma de alimentação, utilizando a fonte de 5V do Arduino, uma de TX, uma de RX (provenientes da comunicação serial), uma entrada de *enable*, que é a habilitação

para enviar dados, necessário em uma rede Modbus *half-duplex* para evitar que dois componentes coloquem dados na linha de comunicação ao mesmo tempo, e uma entrada de GND ou terra. A saída do MAX 485 (A e B na figura 61) são os dois fios necessários do barramento 485 *half-duplex.* Ainda presente no circuito, dois LEDs de identificação que acendem quando há algum dado no TX ou no RX do Arduino.

Com isso, decidi por fazer a minha própria versão do conversor, pois a maioria dos componentes eletrônicos é de baixo custo podem ser encontrados facilmente em lojas especializadas da região. O resultado final é apresentado na próxima página:

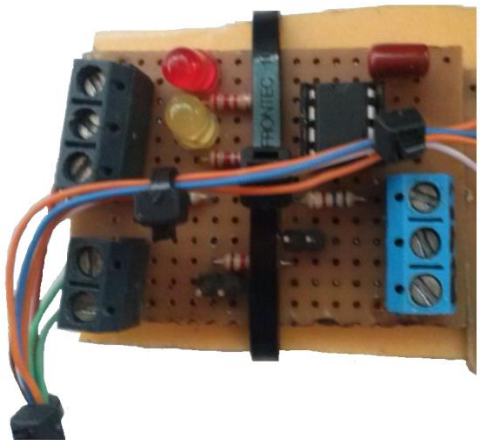

Figura 62- Conversor caseiro RS-485/Serial TTL.

Este conversor foi testado com o exemplo básico que será apresentado abaixo e se comportou conforme o esperado, realizando as mesmas funções do seu homólogo comercial, mas com um custo dez vezes menor aproximadamente.

4.4.4: Exemplo de aplicação.

Para aprimorar a familiaridade com o H-Designer bem como testar o funcionamento da rede Modbus Mestre/Escravo entre Arduino e IHM através da utilização do conversor acima, um exemplo muito simples foi fundamental.

A proposta deste exemplo é apresentada na figura 63, na próxima página:

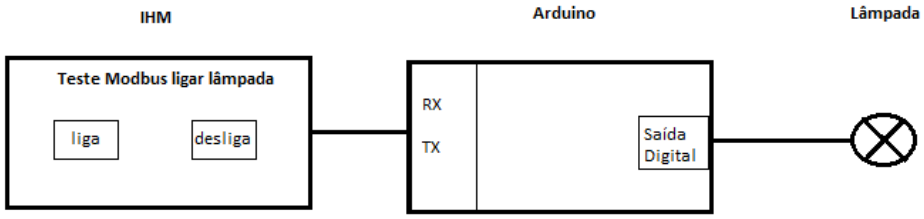

Figura 63- Exemplo de aplicação.

Neste exemplo, um usuário é capaz de ligar e desligar uma lâmpada remotamente selecionando a opção no painel da IHM. Esta por sua ver irá se comunicar com o Arduino através da rede Modbus RTU que processará o comando de ligar ou desligar, alterando o estado de uma saída digital conectada à lâmpada.

A foto abaixo apresenta a montagem que realizei em bancada para construção deste exemplo:

Figura 64- Implementação prática do exemplo.

Para utilização do Arduino como dispositivo escravo de rede Modbus é necessária o emprego de uma biblioteca que suporta as funções descritas no tópico 3.2.3.4 deste documento. A análise desta biblioteca será descrita com detalhes no capítulo que foca a implementação da solução final. Outra artimanha que foi utilizada neste exemplo e que será importante para a solução final é o acionamento de dispositivos 220V de corrente alternada (neste caso a lâmpada fria mas que poderia ser um motor 380V trifásico) através de uma porta digital do Arduino, sendo que esta conexão não é tão trivial pelo motivo do Arduino possuir saídas digitais de tensão 3.3V de corrente contínua.

Este exercício foi fundamental para depurar possíveis erros no conversor e na programação de IHM e controlador e seu sucesso, adicionado ao fato do sucesso de todos os outros exemplos presentes neste capítulo, gerou a confiança necessária para a idealização e implementação da solução final, descrita brevemente a seguir.

4.5: Apresentação conceitual da solução.

Com o aprofundamento do projeto foi finalmente possível apresentar uma solução final, concretizando a proposta de solução apresentada no início do documento. Testados todos os elementos e suas interligações foram observados através de exemplos que a ideia seria capaz de ser realizada e que a mesma trazia os efeitos almejados. O diagrama genérico representado pela figura 13 se especializa no diagrama abaixo, que será a solução final:

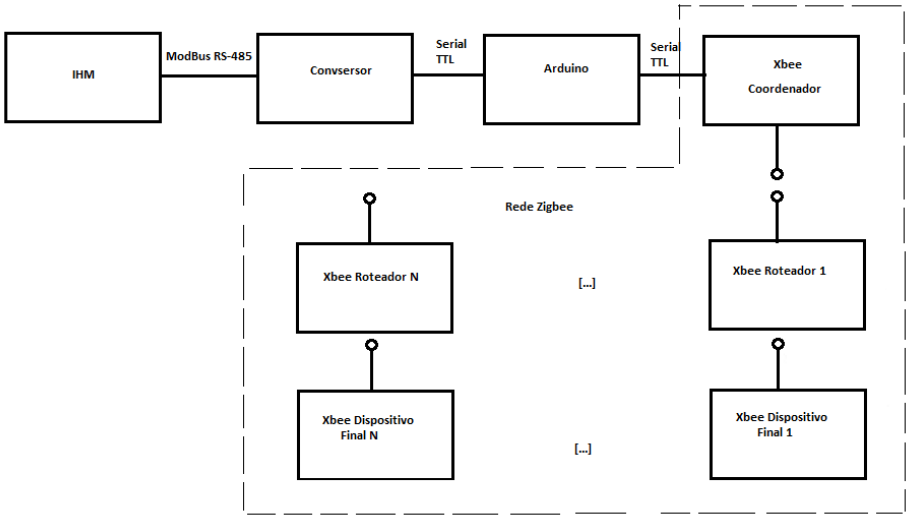

Figura 65- Solução do projeto.

Será formada uma rede Zigbee com a presença de Roteadores para aumentar o alcance quando necessário e Dispositivos Finais, responsáveis pela leitura de sensores e conexão com os atuadores do processo. Estes dispositivos se comunicarão com o Coordenador, conectado ao Arduino que processará as informações provenientes dos Dispositivos Finais e apresentará estas na tela da IHM, bem como gerará sinais de atuação comparando estes sinais com os provindos da IHM ou através de leis de controle pré-definidas (modo manual e modo automático).

Para o problema das EEEs, a distância entre Xbee Coordenador e Dispositivo Final, conectado ao conjunto bomba-motor e sensor de nível, não é grande o suficiente para exigir a presença do roteador. Além disso, EEEs são geralmente unidades autônomas, necessitando apenas um Dispositivo Final por elevatória.

Já para o problema dos aeradores, por se tratar de uma planta muito mais extensa, a presença de Roteadores é necessária, pois a distância entre sala de controle onde se localizará a IHM e processo é maior do que o recomendado. Além disso, por se tratar de diversos aeradores que trabalham na mesma estação, a rede Zigbee será composta de diversos Dispositivos Finais, um para cada aerador.

Este projeto foi concebido de maneira à facilmente adaptar uma configuração à outra, auxiliado pela facilidade proposta pelo próprio protocolo Zigbee em inserção de nós na rede e utilização de Roteadores. Como será visto no próximo capítulo, as alterações necessárias na programação e desenvolvimento do projeto para as duas localidades é mínima e fácil de ser entendida, auxiliando programadores a alterar as configurações conforme as novas necessidades da CASAN.

4.6: Considerações finais do capítulo.

Este capítulo tinha como objetivo apresentar a metodologia de como a solução final para este projeto foi sendo criada, mostrando um pouco de cada passo que foi dado através dos exemplos utilizados.

Também era objetivo apresentar um pouco mais da tecnologia empregada, principalmente a parte de configuração e uma breve apresentação dos *softwares* de auxílio.

Aumentando a complexidade dos exemplos, a solução final foi tomando forma, juntando cada um dos componentes conforme proposto pela aproximação *bottom-up* e finalmente esta foi apresentada de maneira conceitual no final do capítulo, servindo de iniciação para o próximo.

No próximo capítulo, será atacada a solução final, apresentando todos os detalhes do projeto e como os problemas foram solucionados.

Capítulo 5: Implementação.

Neste capítulo será apresentada a implantação da solução do problema. Todos os detalhes construtivos e de programação serão apresentados e discutidos. Conceitos do projeto serão demonstrados em sua finalidade real além de artimanhas necessárias para interligação e bom funcionamento do mesmo.

O método passo-a-passo de configuração de cada um dos elementos da figura 65 será descrito, com o objetivo de responder a pergunta “Qual a solução?”, sendo que ao final do capítulo o projeto finalmente estará apresentado devidamente em sua concepção e funcionamento final.

Este capítulo estará dividido por elemento do sistema de automação para facilitar o entendimento e compreensão, sendo que ao primeiro escolhido é a interface.

5.1: Solucionando o problema de interface.

Neste projeto o operador humano deveria ser capaz de facilmente ler variáveis importantes nos processos e atuar manualmente sobre o sistema de maneira segura e eficaz. A IHM Beijer H-TC80 foi capaz de realizar esta funcionalidade através da programação correta e eficaz. A programação da IHM para a solução do projeto se dividiu em duas partes:

- Programação visual da IHM.
- Programação de comunicação da IHM.

A programação visual da IHM trata de questões ergonômicas e funcionais do dispositivo em relação ao humano que a irá utilizar. Mais sobre estes dois quesitos é apresentado a seguir.

5.1.1: Programação visual da IHM.

A programação visual da IHM é realizada exclusivamente através do *software* H-Designer. Felizmente, esta ferramenta possui uma ampla gama de funções que facilitam a criação das telas.

Como este projeto tem duas aplicações distintas, aeradores e EEEs, dois arquivos diferentes do H-Designer foram criados, um para ser compilado em cada caso.

5.1.1.1: IHM para EEE.

No caso das EEEs, a tela da IHM deve conter as seguintes informações: Tipo de controle, estado da bomba e nível do poço úmido. A primeira tela, ou seja, a tela que o operador se deparará cada vez que o sistema for inicializado é a seguinte:

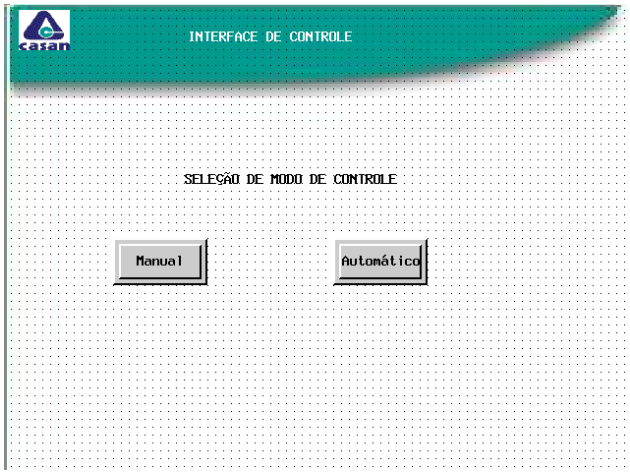

Figura 66- Tela inicial da IHM

Este primeiro quadro possui como fundo de tela o padrão da CASAN para qualquer grafia pertencente à empresa (documentos, placas, *softwares*, pinturas, entre outros), constituído da cor verde-água e o símbolo da empresa. Este fundo foi escolhido para que o projeto siga os padrões em prática no meio, associando o mesmo à CASAN.

Além disso, no alto da tela encontra-se o nome “Interface de Controle”, atribuindo uma identificação à mesma. Na interface de controle o operador tem a possibilidade da seleção do modo de controle entre manual e automático.

A representação gráfica dos botões é uma simples sobreposição de caixas de cor cinza dando efeitos de profundidade sensíveis ao toque. O primeiro modelo deste botão era bem mais complexo, imitando um botão real com duas imagens: uma dele normal e outra sendo pressionado. Porém, recebendo um *feedback* de alguns operadores de EEEs da CASAN, logo percebi que a vontade deles era por

um sistema mais simples, sendo que alguns tiveram dificuldade de acionar corretamente este modelo de botão mais complexo. A solução mais simples provou ser melhor neste caso.

O botão em si possui uma rotina que é ativada cada vez que o mesmo é pressionado. Os botões manual e automático são do tipo “*Goto screen button*”, uma rotina do H-Designer que altera a tela atualmente apresentada para o operador. Os atributos desta rotina foram os seguintes:

- Open/Go to: Screen 2 (botão Manual), Screen 3 (botão Automático).
- Enabled By: None.
- Execution: On press.
- Security Level: 9.

O atributo “Open/Go to” é marcado com o parâmetro “Screen 2”, ou “Screen 3”, ou seja, quando a rotina manual ou automática for ativada a IHM irá trocar a tela atual pela tela com este nome. O atributo “Enabled By” serve para permitir a ação da rotina apenas se alguma condição pré-estabelecida for atingida, sendo que neste caso não é necessário nenhuma restrição. O “Execution”, por sua vez, concede inicialização da rotina ao apertar ou soltar o botão. O movimento natural de um ser humano ao pressionar um botão consiste em duas etapas, a primeira sendo a de exercer pressão sobre o mesmo e a segunda de recuar, retirando a pressão exercida. Este atributo trabalha justamente com estas duas lógicas.

O primeiro raciocínio lógico de um programador seria marcar o “Execution” com o atributo “On release” e não dar muita importância para o parâmetro, como aconteceu inicialmente comigo. Logo percebi, porém, que é bastante comum os operadores da CASAN pressionarem o botão por um tempo aparentemente maior que o necessário, e quando perguntado sobre o motivo para tal ação a resposta era: “Estou esperando a tela mudar”. Quando o terceiro caso seguido ocorreu eu percebi a importância do atributo e prontamente alterei para “On press”, ou seja, ao ser pressionado, gerando um desempenho melhor.

Finalmente, o atributo “Security Level” garante que a rotina será inicializada no nível de segurança máximo, ou seja, ela pode alterar qualquer parâmetro da IHM sem restrições. Este atributo é mais utilizado quando diversos usuários tem acessos

a níveis diferentes dentro da mesma IHM, sendo que alguns devem ter restrição sobre o que e onde podem agir. No caso da IHM para EEEs isto não é importante, pois todos os funcionários que precisam alterar estados ou ler algum valor nas estações gozam do mesmo privilégio. Este atributo será omitido dos próximos botões por ser sempre idêntico.

Pressionado o botão Manual, o operador se defronta com a seguinte tela:

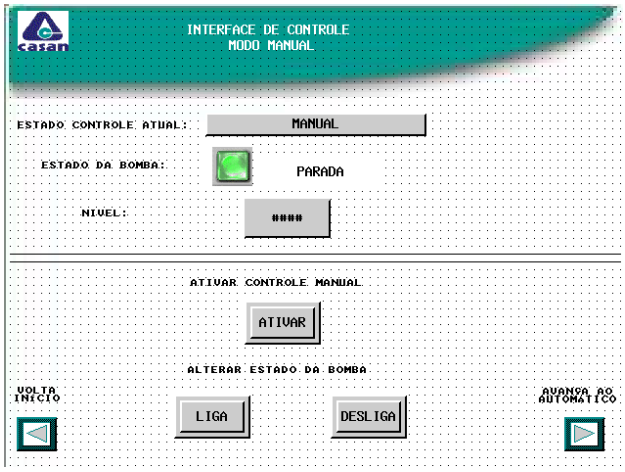

Figura 67- Tela de modo manual da IHM.

Esta tela tem o nome de “Interface de Controle Modo Manual”, e como o nome já diz, trata da parte manual do sistema, onde o operador atua como desejar e achar necessário. A tela é dividida em duas partes, seccionadas por dois traços escuros no centro da imagem. A parte superior é a parte de informação, onde o usuário pode observar os seguintes dados:

- Estado controle atual.
- Estado da bomba.
- Nível.

O “Estado controle atual” é responsável por informar se o sistema está em operação manual ou automática. Esta caixa de informação é do tipo “Multistate Indicator”, uma rotina do H-Designer que lê um determinado endereço e aplica uma indicação gráfica para cada valor desta leitura. Os atributos desta rotina para a IHM da EEE são os seguintes:

- Read: W0.
- Type: Value.

- Format: Unsigned Binary.
- Text: State 1 = text MANUAL, State 2 = text AUTOMATICO.

O atributo “Read” indica de que parte da memória a IHM deve buscar a informação. No caso deste projeto, todos os registradores ficam armazenados no Arduino, ou seja, a unidade controladora. Portanto, quando o operador ativa o controle manual na IHM, ele está fazendo uma operação de escrita em um registrador no Arduino através da rede Modbus RTU e quando a IHM quer ler o estado de algum registrador esta requisita uma operação de leitura de registrador para o Arduino através da mesma rede. Mais sobre os registradores será discutido na parte de comunicação da IHM, mas por momento basta citar que o registrador do Arduino que guarda a informação de estado de controle é o registrador W0, inserido como parâmetro neste atributo.

O atributo “Type” define o tipo de dado que será lido do registrador, podendo ser uma string, um valor haxadecimal, ou no caso deste projeto um “Value”, que representa os números decimais inteiros positivos.

O atributo “format” indica para a IHM qual a forma esperada ao ler estes dados do Arduino, sendo que o parâmetro “Unsigned Binary” indica que será números binários sem sinal, o qual é o padrão de escrita do Arduino assim como da maioria dos equipamentos que fazem transmissão de dados. Outras opções poderiam ser “BCD” ou “Signed Binary”.

Por último, o atributo “Text” confere a representação gráfica dependendo do valor lido no registrador. Neste botão, quando a IHM lê o valor 1 no registrador W0 ela imprime na tela o texto “MANUAL” e quando o valor lido é 2, é impresso “AUTOMATICO”, servindo para a interpretação do operador.

A segunda informação presente na IHM para o operador no modo manual é o “Estado da bomba”, que serve para indicar se a mesma está parada ou em operação. Esta informação é apresentada na tela através de um botão tipo “Multistate Indicator”, semelhante ao “Estado controle atual”, configurado da seguinte maneira:

- Read: W1.
- Type: Value.

- Format: Unsigned Binary.
- Image: State 1 = file BombaDesligada.jpg, State 2 = file BombaLigada.jpg.
- Text: State 1 = text DESLIGADA. State 2 = text LIGADA.

A diferença para o botão anterior é a leitura de outro registrador do Arduino, neste caso o registrador W1, que guarda a informação de bomba ligada ou desligada. Como obter a informação de bomba ligada será apresentado mais para frente no tópico destinado para tal.

Outra diferença é a presença do atributo “Image”, onde é possível atribuir uma imagem de arquivo associada a cada estado obtido pela leitura do registrador. Neste caso, quando a IHM ler informação de bomba parada esta irá atribuir a imagem “BombaDesligada.jpg”, uma forma retangular verde possível de ser vista na figura 67, e quando ler a informação de bomba ligada irá atribuir a imagem “BombaLigada.jpg” à mesma. O atributo “Text” confere um texto de “LIGADA” ou “DESLIGADA” à imagem conforme leitura do registrador.

A terceira informação presente na tela da IHM é o “Nivel”, responsável por apresentar ao operador a altura de esgoto contido no poço úmido da EEE. Esta informação é mostrada através de um botão do tipo “Numeric Display”, uma rotina do H-Designer para apresentação de dados numéricos na tela da IHM. Os atributos desta rotina são os seguintes:

- Read: W3.
- Type: Value.
- Format: Unsigned Binary.
- Integral Digits: 1.
- Fractional Digits: 2.
- Gain: 1.

Como obter as leituras de nível não será discutido neste tópico ainda, apenas sendo preciso comentar que estas leituras são armazenadas no registrador W3 do Arduino, configurado como parâmetro do atributo “Read”. O formato e tipo de dados são idênticos aos casos anteriores.

O atributo “Integral Digits” serve para informar quantos dígitos do valor recebido do registrados são inteiros e o parâmetro “Fractional Digits” quantos são fracionados. Neste caso uma possível leitura na tela seria de 9,22 metros ou 3,70 metros. O atributo “Gain” permite alterar valor de ganho da leitura do registrador, sendo neste caso não necessário (configurado em 1).

Estes são as três informações de leitura da tela da figura 67, valendo citar ainda que estes botões estão protegidos contra toque, sendo impossível o operador apertá-los por engano.

Já na parte de baixo da figura 67, estão presentes os parâmetros que o operador pode alterar no sistema, sendo eles: Ativação do controle manual e alteração do estado da bomba.

A ativação do controle manual é feita através de um botão do tipo “Set Constant Button”, uma rotina do H-Designer capaz de atribuir um valor constante a um registrador específico quando este for pressionado. A configuração deste botão segue abaixo:

- Write: W0.
- Set Value: Word.
- Value: 1.
- Execution: On press.
- Notification: None.

Com estas configurações, ao se pressionar o botão “ATIVAR”, a IHM irá escrever no registrador W0 do Arduino o valor 1, do tipo “word”. Os registradores de uma rede Modbus são sempre deste tipo, ou seja, são palavras de 16 bits e para garantir a escrita correta o tipo “word” deve ser selecionado.

É possível observar ainda que o registrador W0 é o mesmo utilizado para a leitura do modo de controle, presente acima na mesma tela, fechando assim o ciclo de escrita/leitura do parâmetro.

Com o atributo “Notification” é possível avisar sonoramente ou visualmente a escrita do parâmetro, mas como neste projeto já há botões específicos para esta notificação o mesmo receberá o parâmetro “None”, ou seja, nenhum.

O outro parâmetro possível de se alterar no sistema é o estado da bomba. Para realizar tal tarefa existem dois botões do tipo “Set Constant Button”, com a seguinte parametrização:

- Write: W2.
- Set Value: Word.
- Value: 1 (botão DESLIGA), 2 (botão LIGA).
- Execution: On press.
- Notification: None.

Primeiramente vale observar que estes botões escrevem em um registrador do Arduino diferente do registrador utilizado para leitura do estado da bomba, e isto não é um erro. O que ocorre é que se for utilizado o mesmo registrador, a IHM irá entender que a bomba esta ligada quando for enviado o comando para ela ligar, mas isso pode não ser verdade se ocorrer uma falha ou até mesmo quebra da bomba. É necessário então um registrador exclusivo que guardará a interpretação de um sinal físico proveniente da bomba quando a mesma está operando, ou como conceituamos na CASAN, um registrador de liga bomba e um registrador de bomba ligada.

Os valores do tipo “word” podem assumir os valores 1 e 2 para desliga e liga respectivamente e serão escritos ao se pressionar o botão sem nenhuma notificação sonora ou visual.

Ainda presente na figura 67 está dois botões, no canto direito inferior e no canto esquerdo inferior que servem para voltar para a tela da figura 66 (tela inicial) ou avançar para a tela de controle automático ainda a ser apresentada.

Com isso fica explicada toda a configuração da tela de controle manual, restando apenas agora a tela da controle automático, quando o botão “Automatico” da figura 66 é pressionado.

A tela da IHM para controle automático está presente na próxima página:

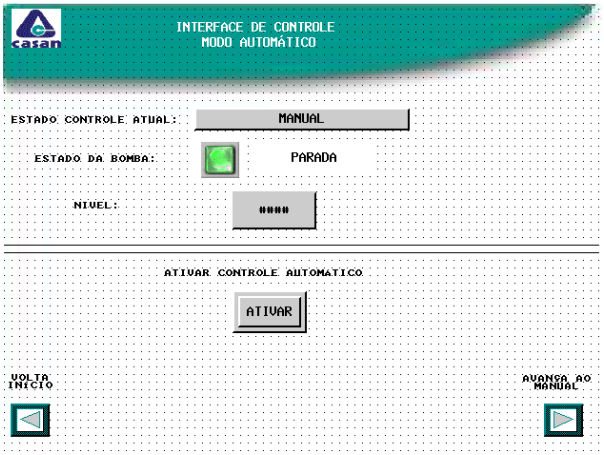

Figura 68- Tela modo automático da IHM.

Esta tela, denominada “Interface de controle modo automático” possui as mesmas informações do modo manual na parte superior da tela, sendo elas: Estado do controle, estado da bomba e nível. A configuração destes botões é idêntica ao caso anterior.

A diferença está na parte inferior da tabela, onde o operador humano não tem mais possibilidade de controlar o estado da bomba, isto está sendo feito de forma automática pelo controlador. A única atuação do operador humano neste processo é a ativação do controle automático quando este se encontra em estado manual. Este é atingido através de um botão “Set Constant Button” com os seguintes parâmetros:

- Write: W0.
- Set Value: Word.
- Value: 2.
- Execution: On press.
- Notification: None.

Com isto a parte gráfica da IHM presente na solução final para a EEE fica apresentada faltando agora explicar a parte gráfica para a solução dos Aeradores.

5.1.1.2: IHM para aeradores.

Nesta implementação, o operador deve conseguir observar dados de oito aeradores, numerados de A1 a A8, recebendo informação de funcionamento. Esta informação é capturada na tela inicial da IHM, apresentada abaixo:

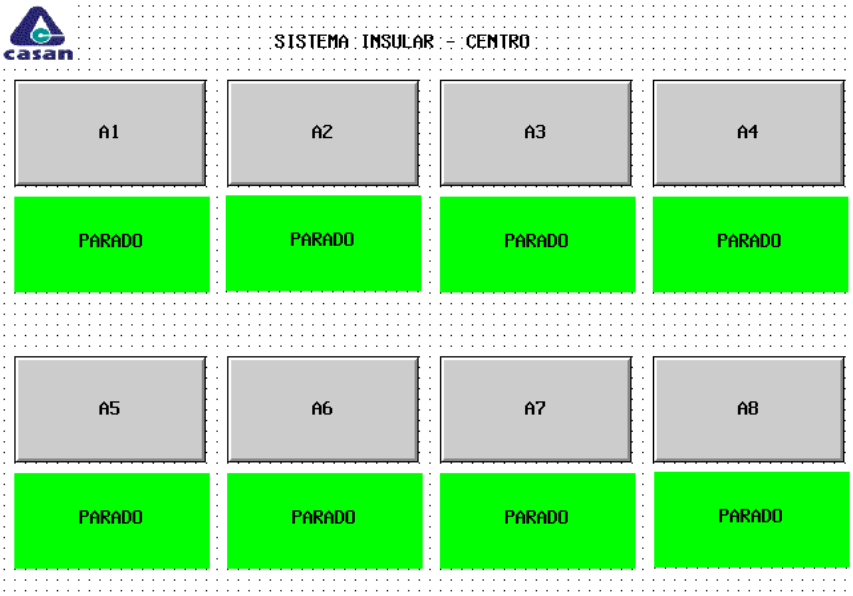

Figura 69- Tela inicial da IHM com aeradores.

Nesta mesma tela o operador pode observar o estado de todos os aeradores de maneira rápida e direta. Os botões denominados “A1, A2, A3, A4, A5, A6, A7 e A8” são do tipo “Goto Screen”, onde cada um abre a tela individual de cada aerador, sendo que a tela do A1 é apresentada abaixo:

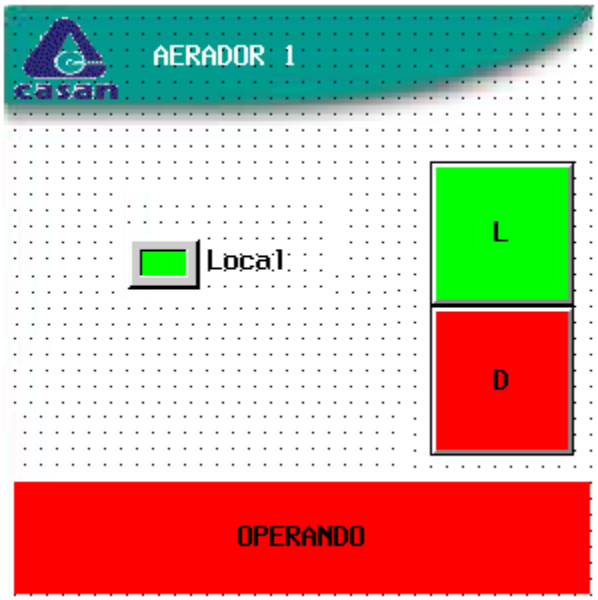

Figura 70- Tela de modificação de parâmetros do aerador 1.

Nesta tela o operador tem disponível a informação de “Local”, ou seja, o modo de controle atual do sistema. O termo local em si é bastante utilizado na

CASAN com significado de manual, e por isso é utilizado nesta tela. Além disso o operador tem acesso a um botão para ligar o motor (botão verde com a letra “L”) e um botão para desligar o motor (botão vermelho com a letra “D”). No canto inferior é possível observar o estado atual do motor, entre operando (em vermelho) e parado (em verde). Esta tela é repetida para cada um dos oito aeradores.

A programação dos botões é idêntica ao caso para a EEE, e por isso não são entrados em detalhe neste tópico.

Apresentadas as diferenças nas telas das duas aplicações, o problema se torna novamente em um só: a parte de programação da IHM, que por sua vez é independente do *layout* das telas.

5.1.2: Programação de comunicação da IHM.

Ambas as soluções de IHM para EEE e aeradores são programadas identicamente em relação à sua comunicação e por isso este tópico não necessita de divisão. Como programar esta parte é descrita a seguir.

Inicialmente, no H-Designer é possível configurar todos os parâmetros de comunicação entre IHM e dispositivos periféricos comunicantes. Para garantir que os dados inseridos no *software* irão reger a rede de comunicação a chave SW5 deve estar desligada, conforme tópico 4.4.1.

A configuração destes parâmetros é feita na aba “Application Properties” que abre automaticamente quando um novo arquivo no H-Designer é criado. Esta aba é mostrada abaixo:

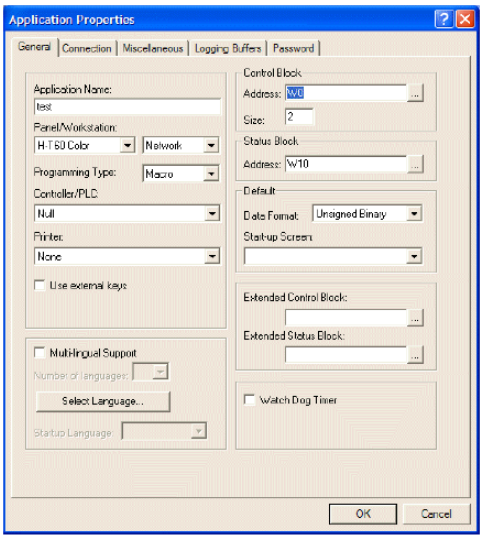

Figura 69- Application Properties.

Em $Application\ Properties \rightarrow general$, no canto esquerdo alto, o campo “Application name” serve para inserção de um nome para a aplicação a ser criada. No caso deste projeto a aplicação foi denominada “Projeto Zigbee”.

No campo “Panel/Workstation” é inserido o modelo da IHM a ser utilizada, sendo que neste caso é a H-T80c. No campo “Controller/PLC” é onde o *driver* do controlador pode ser selecionado, conforme citado no tópico 4.4.3. Para o Arduino a escolha neste campo é por “Modbus máster/Slave”, ou seja, rede Modbus do tipo Mestre/Escravo.

Ainda nesta aba devem ser alterados os campos “Control Block”, “Status Block” e “Data Format”. O campo “Control Block” serve para informar onde iniciam os registradores de controlador, sendo que no caso deste projeto é o W0, o primeiro registrador programado no Arduino. Abaixo deste campo é ainda necessário informar qual o tamanho deste bloco (campo “Size”), sendo que para o projeto aqui apresentado um tamanho de 10 registradores é o suficiente.

O “Status Block” são registradores que podem informar diversos tipos de erros desde comunicação até pacotes mal formados ou corrompidos. Como esta funcionalidade não está presente no Arduino, este campo deve ser deixado vazio (sem registradores) para evitar erros.

No campo “Data Format” é especificado qual o formato de dados que o controlador conectado a IHM espera receber da mesma, sendo que pro caso do Arduino este formato é “Usigned Binary”.

Depois de configurada esta aba apenas mais uma é necessária para o bom funcionamento, está é a $Application\ Properties \rightarrow Connection$, apresentada abaixo.

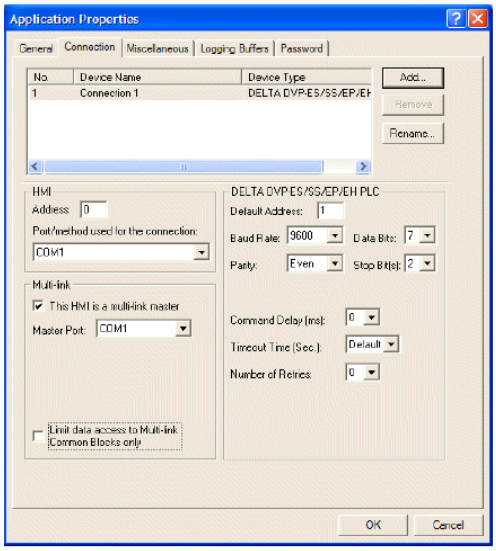

Figura 70- Aba Connection.

Nesta Aba deve ser definido em que porta de comunicação da IHM o controlador estará conectado e qual o método utilizado para a comunicação entre os dois (campo “Port/Method used for the connection”). No caso deste projeto o campo será preenchido com o dado “COM1 – RS 485”, ou seja, porta de comunicação 1 RS-485.

Ainda nesta aba devem ser configurados os parâmetros de “Default Adress”, “Baud Rate”, “Data bits”, “Stop bits”, e “Parity” do mestre da rede Modbus, que neste caso é a IHM. Para este projeto foram utilizadas as seguintes configurações:

- Default Adress: 0.
- Baud Rate: 9600.
- Data bits: 8.
- Stop bits: 1.
- Parity: None.

Lembrando que estes dados devem bater com o da programação do Arduino para criação da rede Modbus para a comunicação entre os dois nós.

Feito estes passos, a última etapa necessária é adicionar um dispositivo no sistema. Isto é necessário para que a IHM entenda que tipo de dispositivo estará conectado e seu comportamento e endereço. Isto é atingido pressionando o botão “Add” na figura 70, onde é possível atribuir um nome ao dispositivo, um endereço de rede e seu comportamento. Para este projeto, o dispositivo adicionado se chamará Arduino, com um endereço de nó Modbus dois e com comportamento de “Slave” (escravo).

Terminada esta configuração, basta salvar o arquivo e fazer seu *upload* para a IHM utilizando o cabo RS-232 criado para tal funcionalidade (vide tópico 4.4.2).

Com isso, toda a configuração e apresentação de telas da IHM presentes na solução final é apresentada, podendo-se continuar a sequência para o Arduino.

5.2: Implantando o Arduino na solução final.

Definida a apresentação final das telas da IHM e conhecida a necessidade da utilização do conversor para ligação entre IHM e Arduino, agora é chegada a hora de apresentar a programação do controlador deste projeto.

Como foi comentado anteriormente, uma das premissas em que este projeto foi baseado é a fácil alteração entre as configurações necessárias para uma aplicação em uma EEE e uma aplicação nos aeradores e isto será provado neste tópico, onde as alterações entre a programação do Arduino de uma em relação à outra será mínima.

O Arduino terá papel de controlador neste projeto, ou seja, ele é o dispositivo centralizador que tomará as decisões e interligará todos os componentes integrantes.

Para realizar tal feito, o Arduino necessita ser programado conforme as necessidades apresentadas anteriormente, e esta programação é a chave de todo o projeto. A criação e utilização de bibliotecas foi fundamental nesta caminhada, sendo suas em especial: A biblioteca Modbus e a biblioteca SoftwareSerial, descritas abaixo:

5.2.1: Biblioteca Modbus.

Esta biblioteca é uma implementação para placas integradas Arduino e através desta é possível transformar o controlador em um escravo de rede Modbus. Lembrando as funções do Modbus da tabela 2, tópico 3.2.3.4, a biblioteca deve ser capaz de implementar ao menos três pares de funções, sendo elas:

- Ler entradas analógicas e digitais.
- Alterar saídas analógicas e digitais.
- Alterar várias saídas analógicas e digitais.

A biblioteca utilizada neste projeto é uma biblioteca denominada ModbusSlave que costumava ser suportada oficialmente pelo Arduino nas versões beta da IDE Arduino. Porém com o lançamento da primeira versão completa, esta biblioteca desapareceu da lista de suporte e ficou totalmente desatualizada para a nova linguagem de programação do Arduino.

Como as mudanças das versões beta para a versão atual da IDE são, na sua maioria, mudanças na forma de declaração de variáveis e mudanças nas chamadas de funções, resolvi por conta própria atualizar a biblioteca ModbusSlave para a versão atual da IDE (1.5.2). Este “sequestro” de biblioteca para atualização somente é permitido pelo fato de todas as bibliotecas do Arduino são de domínio publico e não podem ser resgatadas por direitos autorais. Isso me deu base legal para alterar a mesma conforme minhas necessidades e publicar os resultados na comunidade para que outros possam desfrutar das mesmas funcionalidades.

As funcionalidades utilizadas para o projeto serão discutidas especificamente quando o código do Arduino for apresentado.

5.2.2: Biblioteca SoftwareSerial.

Esta biblioteca, por sua vez, é totalmente suportada na versão atual da IDE Arduino sendo necessário apenas utiliza-la de forma correta e sem necessidade de modificações.

O princípio de funcionamento é através da virtualização de uma porta serial do Arduino, emulando todas as funcionalidades da biblioteca serial padrão. Com

isso, é possível criar novos canais de comunicação em dispositivos que só possuem um, como é o caso do Arduino UNO neste trabalho. O contraposto, cada criação de uma porta serial utiliza dois pinos livres do equipamento, reduzindo o número de entras e saídas disponíveis para aplicação.

Neste projeto, toda a comunicação provinda do Arduino e para o Arduino é realizada através das duas portas seriais, uma conectada no conversor que por sua vez estará conectado na IHM e outra no dispositivo Xbee Coordenador, ou seja, nenhuma informação é passada diretamente através de uma entrada ou saída digital/analógica. Por isso, não há problema em se utilizar dois pinos (no caso deste trabalho o pino 2 e o pino 3) para a criação da segunda porta serial necessária.

As suas funcionalidades utilizadas neste projeto serão devidamente apresentadas quando o código final for demonstrado.

5.2.3: Recebendo e criando pacotes API com o Arduino.

Como mencionado em capítulos anteriores, os equipamentos Xbee presentes nesta solução trabalharão em modo API e o dispositivo responsável pela criação dos pacotes e interpretação dos mesmos é o Arduino. Uma breve introdução ao assunto é apresentada no próximo tópico.

5.2.3.1: Modo API.

Um API, *Application Programing Interface*, é simplesmente um conjunto de interfaces padrão criados para permitir que dois dispositivos interajam entre si. O API permite que uma aplicação requisite alguma informação de outra aplicação de uma maneira formal padrão. No escopo deste trabalho o API é utilizado na interface entre Xbees.

No exemplo do tópico 4.1.2 a interação entre homem e dispositivo era feita diretamente através do teclado e para isto o modo AT era recomendado. O problema deste modo é sua baixa robustez e eficiência na troca de mensagens, fazendo com que o lado computacional tenha um pior desempenho. O modo API tenta equilibrar a balança entre facilidade de interação com humanos e facilidade de interação entre

máquinas, sendo um pouco mais difícil de entender e programar, mas trazendo uma série de vantagens que melhoram a eficiência técnica da comunicação.

O objetivo do modo API é transmitir dados altamente estruturados rapidamente, de forma previsível e confiável.

5.2.3.2: Estrutura dos dados API.

Os dados do modo API são organizados conforme a figura abaixo:

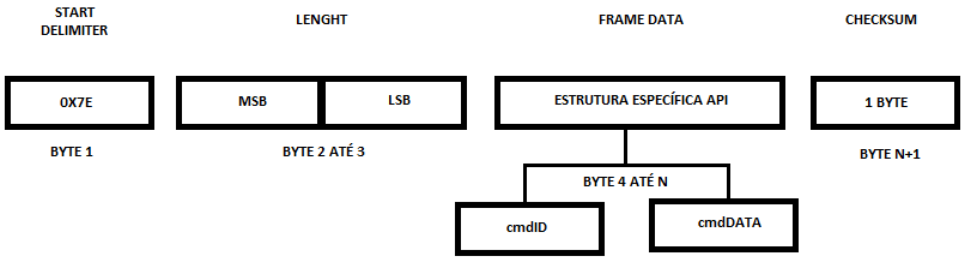

Figura 71- Modo API.

O “Start Delimiter” informa a iniciação da comunicação em modo API. Quando receptores Xbee recebem o byte 0x7E, perceberão que esta comunicação se iniciou e passarão a decifrar os bytes seguintes como um pacote de dado.

O “Lenght”, representado pelos bytes 2 e 3, indica o comprimento do quadro de dados, denominado “Frame Data”. Esta divisão ocorre primeiramente pelo byte mais significante (MSB) e depois pelo menos significante (LSB).

Já o “Frame Data” é subdividido em duas partes, sendo “cmdID” e “cmdDATA”. O byte “cmdID” indica qual comando a ser realizado segundo uma tabela do modo API, apresentada na próxima página:

Comando	Função	Valor Hexa
AT Command	Configura salvando ou lê um determinado parâmetro local no módulo	0x08
AT Command - Queue	Configura sem salvar ou lê o valor de um parâmetro local no módulo	0x09
Zigbee Transmit Request	Comando Zigbee que solicita uma transmissão para um determinado endereço	0x10
Explicit Adressing Zigbee Command frame	Permite que os Dispositivos Finais e os ID sejam especificados em um pacote de transmissão	0x11
Remote AT Command Request	Utilizado para solicitar ou configurar parâmetros de um módulo remoto	0x17
Create Source Route	Cria uma rota de comunicação no módulo	0x21
AT Command Response	Responde a solicitação de um comando AT	0x88
Modem Status	Retorna o estado do módulo para um determinado evento	0x8A
Zigbee Transmit Status	Indica se a transmissão do último pacote foi transmitida com sucesso	0x8B
Zigbee Rceiver Packet	Recebe um pacote pelas portas UART do módulo	0x90
Zigbee Explicit Rx Indicator	Quando o módulo recebe um pacote Zigbee Transmit Packet é enviado este pacote pela UART	0x91
Zigbee IO Data Sample Rx Indicator	São enviados pela UART o estado das portas digitais e os valores da conversão analógica	0x92
Xbee Sensor Read Indication	Recebe leitura dos sensores	0x94
Node Identification Indicator	Recebe identificação do módulo remoto	0x95
Remote Command Response	Recebe este pacote em resposta ao comando Remote Command Request	0x97
Over-the-air Firmware Update Status	Fornece a indicação do estado da atualização do Firmware	0xA0
Route Record Indicator	É recebido quando o módulo envia um Zigbee Route Record Command	0xA1
Many-to-one Route Request Indicator	É enviado pela UART quando o Many-to-one Route Request Command é recebido	0xA3

Tabela 9- Tabela dos comandos do modo API.

Como pode ser visto, o modo API possui diversas funcionalidades que auxiliam na criação de projetos com o intuito de obter alguma informação de módulos remotos ou alterar alguma configuração dos mesmos. Para a programação do Arduino neste trabalho foram utilizados dois comandos, o “Remote AT Command Request” e o “Remmote Command Response”, que serão apresentados com mais detalhes abaixo.

5.2.3.3: Remote AT Command Request (0x17).

Através desse pacote o Arduino pode solicitar ao módulo Coordenador que envie um comando AT sem fio para um módulo remoto roteador ou dispositivo final.

O campo 0x17 pode assumir vários comandos AT, como por exemplo, o comando “IS” que solicita ao módulo remoto informações sobre as portas digitais e analógicas e o comando “D1” que modifica o estado de uma saída digital. Mais sobre este assunto será discutido no código do programa.

A tabela abaixo descreve a funcionalidade de cada byte do pacote 0x17:

	Remote AT Command Request (0x17)				
	Campo do pacote		Byte	Exemplo	Descrição
Pacote API	Start Delimiter		0	0x7E	Byte de Início do pacote
	Lenght		MSB 1	0x00	Número de byte emtre o Lenght e o Checksum, ou seja, o número de bytes o Frame Data
			LSB 2	0x0F	
	Frame Data	Frame Type	3	0x17	Tipo de comando do pacote
		Frame ID	4	0x01	Identifica o pacote UART do host relacionado co sua confirmação. Se configurado com 0x00 não gera resposta.
		Endereço de 64 bits do dispositivo final	5	0x00	Configura o endereço de 64 bits do destino do dispositivo final. O endereço de 64 bits 0x0000000000000000 é resevado para o Coordenador. O endereço de 64 bits 0x000000000000FF é reservado quando se deseja transmitir em Broadcast.
			6	0x00	
			7	0x13	
			8	0xA2	
			9	0x00	
			10	0x40	
			11	0x4B	
			12	0x7C	
			13	0xA8	
		Endereço de 16 bits do dispositivo final	14	0xFF	Configura o endereço do destino do dispositivo final, caso ele seja conhecido. Quando configurado com o endereço 0xFE o endereço é desconhecido ou é um envio Broadcast.
			15	0xFE	
		Remote Command Options	16	0x02	Campo para habilitar as seguintes opções: 0x01: Desabilitar ACK, 0x02: Aplica mudanças idediatamente, 0x40: Usa tempo limite estendido para transmissão de destino, 0x00: sem função
		Comando AT	17	0x49 (I)	Nome do camando AT
			18	0x53 (S)	
		Parâmetro do comando	19		Caso presente, indica o valor do parâmetro solicitado para configurar um dado registro. Se não houver nenhum caractere presente, o registro é apenas consultado.
	Checksum		20	0xE8	É o resultado do checksum.

Tabela 10- Pacote 0x17

O Arduino será responsável pela montagem deste pacote conforme padronização observada acima e ao enviar o pacote Remote AT Command Request, o pacote recebido é o Remote Command Response (0x97) apresentado no próximo tópico.

5.2.3.4: Remote Command Response (0x97).

A resposta ao comando 0x17 é um pacote com a seguinte formatação, presente na tabela da próxima página:

	Remote AT Command Responde (0x97)				
	Campo do pacote		Byte	Exemplo	Descrição
Pacote API	Start Delimiter		0	0x7E	Byte de Início do pacote
	Lenght		MSB 1	0x00	Número de byte emtre o Lenght e o Checksum, ou seja, o número de bytes o Frame Data
			LSB 2	0x17	
	Frame Data	Frame Type	3	0x97	Tipo de comando do pacote
		Frame ID	4	0x01	Identifica o pacote UART do host relacionado co sua confirmação. Se configurado com 0x00 não gera resposta.
		Endereço de 64 bits do dispositivo final	5	0x00	Configura o endereço de 64 bits do destino do dispositivo final. O endereço de 64 bits 0x0000000000000000 é resevado para o Coordenador. O endereço de 64 bits 0x000000000000FFF é reservado quando se deseja transmitir em Broadcast.
			6	0x00	
			7	0x13	
			8	0xA2	
			9	0x00	
			10	0x00	
			11	0x4B	
			12	0xA2	
			13	0xA8	
		Endereço de 16 bits do dispositivo final	14	0xFF	Configura o endereço do destino do dispositivo final, caso ele seja conhecido. Quando configurado com o endereço 0xFE o endereço é desconhecido ou é um envio Broadcast.
			15	0xFE	
		.	16		Não Utilizado
		Comando AT	17	0x49 (I)	Nome do camando AT
			18	0x53 (S)	
		Estado do comando	19	0x00	0x00: Dado recebido com sucesso, 0x01: Houve erro, 0x02: Comando Inválido, 0x03: Parâmetro Inválido, 0x04: O comando de transmissão remota falhou.
		Dados de comando	20	0x01	Informa que existem portas digitais e analógicas habilitadas
			21	0x00	Indica quais as portas digitais que estão habilitadas
			22	0x02	
			23	0x04	Indica quais as portas analógicas que estão habilitadas
			24	0x00	Indica o estado das portas digitais
			25	0x02	
			26	0x02	Valor convertido da porta analógica
			27	0x33	
	Checksum		28	0xB1	É o valor do checksum

Tabela 11- Pacote 0x97.

Esta tabela é muito importante para o Arduino poder reconhecer algum estado do sistema e receber informação de sensores para tomar as devidas ações de controle.

Com o conhecimento do modo API e das bibliotecas utilizadas agora já é possível apresentar o código de programação do Arduino para o problema da EEE.

5.2.4: Apresentação e discussão do código do Arduino para o problema da EEE.

Abaixo é apresentado o código do Arduino utilizado na solução deste problema:

```
#include <ModbusSlave.h>
   #include <SoftwareSerial.h>

    /* Instanciando um objeto mbs do tipo ModbusSlave */
    ModbusSlave mbs;
    SoftwareSerial mySerial(2,3); // RX TX

    /* registradores */
    enum {
            MB_W0,          /* Registrador Modbus para modo de controle*/
            MB_W1,        /* Registrador Modbus que indica estado da bomba*/
            MB_W2,        /* Registrador Modbus para ligar bomba */
            MB_W3,        /* Registrador Modbus que guarda o nível*/
            MB_REGS     /* guarda o número total de registradores, necessário para biblioteca */
    };

    int regs[MB_REGS];
    int status_digital = 0;
    int analogValue = 0;
    int last_indicator=0;
    int erro = 1; /*configuracao inicial do erro */

    void setup()
    {

    /* Configuraçao da rede Modbus */
    const unsigned char SLAVE = 2;
    const long BAUD = 9600;
    const char PARITY = 'n';
    pinMode(6, OUTPUT);
    const char TXENPIN = 6;

    /* chamada da funçao configure com a passagem dos parametros */
    mbs.configure(SLAVE,BAUD,PARITY,TXENPIN);
    mySerial.begin(9600);

    }

    void loop()
    {
      if (mySerial.available() >= 23) {
    // procura o Start delimiter
    if (mySerial.read() == 0x7E) {
      for (int i = 0; i<19; i++) {
        byte discard = mySerial.read();
      }
```

Continua na próxima página.

```
    erro = mySerial.read();
    for (int i = 0; i<5; i++) {
      byte discard = mySerial.read();
    }
    status_digital = mySerial.read();
    int analogHigh = mySerial.read();
    int analogLow = mySerial.read();
    analogValue =  (analogLow + (analogHigh * 256))*1200/1024;
  }
}

if (erro == 0)
  {
    if (status_digital == 4){
           regs[MB_W1]=2; /*bomba ligada*/ }
    else { regs[MB_W1]=1; /*bomba desligada*/ }
regs[MB_W3]=analogValue;
  }

  if (analogValue>4 && regs[MB_W0] == 2) /*Modo automatico*/
  {
    regs[MB_W2]=1; /*liga bomba*/
  }
  if (analogValue<2 && regs[MB_W0] == 2)
  {
    regs[MB_W2]=0; /*desliga bomba*/
  }


    /* update dos registradores com as informacoes da IHM*/
   mbs.update(regs, MB_REGS);

    if(regs[MB_W2]==1 && last_indicator==0) {
       setRemoteState(0x5);
       last_indicator=1;}
       else{
         if(regs[MB_W2]==0 && last_indicator==1){
       setRemoteState(0x4);
       last_indicator=0;

         }
       }

  }
```

Continua na próxima página.

```
  void setRemoteState(int value) {  // passa ou um 0x4 ou um 0x5 para ligar ou desligar um pino
   // start delimiter
  mySerial.write(byte(0x7E));
   // MSB
  mySerial.write(byte(0x0));
   // LSB(o numero de bytes que restam no pacote
   // sem contar o checksum
  mySerial.write(byte(0x10));
  // comando 0x17
  mySerial.write(byte(0x17));
  //frame ID com valor 0 para não gerar resposta de identificacao
  mySerial.write(byte(0x00));
  // ID do recipiente
  mySerial.write(byte(00));
  mySerial.write(byte(0x13));
  mySerial.write(byte(0xA2));
  mySerial.write(byte(00));
  mySerial.write(byte(0x40));
  mySerial.write(byte(0x9E));
  mySerial.write(byte(0x86));
  mySerial.write(byte(0x21));
  // endereço de 16 bits do recipiente, 0xFFFE se desconhecido
  mySerial.write(byte(0xFF));
  mySerial.write(byte(0xFE));
  // 0x02 aplica mudanças imediatamente, é o Remote comand option
  mySerial.write(byte(0x02));
  // nome do comando em ASCII
  mySerial.write(byte('D'));
  mySerial.write(byte('1'));
  // parâmetro do comando
  mySerial.write(byte(value));
  // checksum
  long sum = 0x17 + 0x13 + 0xA2 + 0x40 + 0x9E + 0x86 + 0x21 + 0xFF + 0xFE +0x02 + 'D' + '1' + value;
  // calcula o valor do checksum
  mySerial.write(byte(0xFF - ( sum & 0xFF)));
  delay(10); // evita sobrefluxo da porta serial
}
```

É evidente que embora não seja muito extenso, analisar este código inteiro de uma única vez seria confuso e extenuante, por isso o mesmo será analisado por partes, começando com a parte do código inicial, presente na próxima página:

```
#include <ModbusSlave.h>
   #include <SoftwareSerial.h>

    /* Instanciando um objeto mbs do tipo ModbusSlave */
    ModbusSlave mbs;
    SoftwareSerial mySerial(2,3); // RX TX

    /* registradores */
    enum {
            MB_W0,          /* Registrador Modbus para modo de controle*/
            MB_W1,        /* Registrador Modbus que indica estado da bomba*/
            MB_W2,        /* Registrador Modbus para ligar bomba */
            MB_W3,        /* Registrador Modbus que guarda o nível*/
            MB_REGS     /* guarda o número total de registradores, necessário para biblioteca */
    };

    int regs[MB_REGS];
    int status_digital = 0;
    int analogValue = 0;
    int last_indicator=0;
    int erro = 1; /*configuracao inicial do erro */

    void setup()
    {

    /* Configuraçao da rede Modbus */
    const unsigned char SLAVE = 2;
    const long BAUD = 9600;
    const char PARITY = 'n';
    pinMode(6, OUTPUT);
    const char TXENPIN = 6;

    /* chamada da funçao configure com a passagem dos parametros */
    mbs.configure(SLAVE,BAUD,PARITY,TXENPIN);
    mySerial.begin(9600);

    }
```

As primeiras duas linhas do código definem quais bibliotecas devem ser inclusas na aplicação, sendo elas a “ModbusSlave” e a “SoftwareSerial”, citadas anteriormente neste documento.

O próximo passo é criar um objeto do tipo Modbus, neste caso o “mbs”. Ao ser instanciado, este herda todas as características e funcionalidades da classe ModbusSlave, sendo possível a partir deste momento utilizar os métodos da classe para inferir no processo.

Seguindo a mesma ideia, e criado na próxima linha um objeto do tipo SoftwareSerial, aproveitando das funcionalidades da biblioteca. Nos parâmetros passados nesta inicialização estão os pinos que o programador deseja sacrificar para a criação de uma porta serial virtual, neste caso o pino 2 para o Rx e o pino 3 para o Tx.

A lista “enum” é responsável por concatenar todos os registradores necessários para este projeto. São eles:

- Registrador W0: Registra o modo de controle (automático/manual).
- Registrador W1: Registra o estado da bomba.
- Registrador W2: Registrador que comanda o estado da bomba (ligado/desligado).
- Registrador W3: Registrador que armazena o nível do poço úmido.
- Registrador REGS: Registrador que armazena o numero de registradores na aplicação.

Vale ressaltar que estes registradores obrigatoriamente devem coincidir com os registradores programados na IHM para não haver conflitos ou troca de informações, inclusive no seu nome. O número de registradores também não pode passar do número de registradores programados no “Control Block” da IHM, o qual foi explicado no tópico 5.1.2.

Seguindo no código, algumas variáveis globais auxiliares são criadas e serão discutidas quando forem utilizadas no código. Terminada as definições globais, a próximo passo de uma programação de Arduino é a rotina de *setup*, ou inicialização.

Na inicialização, é utilizado o método da classe ModbusSlave denominado *configure*. Este método tem como parâmetros o endereço do nó Modbus, a sua *baud rate* (taxa), a sua paridade e qual pino do Arduino será responsável pela habilitação da comunicação serial (*enable pin)*. Este método sempre considera apenas a opção de 1 bit de parada, sendo que não é necessário informar este parâmetro.

Para este projeto foi utilizado o Arduino como endereço de escravo 2, taxa de 9600 bps, sem paridade (‘n’) e o pino 6 como pino de habilitação e por isso o mesmo é inicializado como “OUTPUT”, ou seja, saída.

A última linha desta parte do código inicia a porta serial do Arduino também com taxa de 9600 bps, a mesma do objeto Modbus.

Uma segunda parte do código é apresentada na próxima página:

```
void loop()
{
  if (mySerial.available() >= 23) {
// procura o Start delimiter
if (mySerial.read() == 0x7E) {
  for (int i = 0; i<19; i++) {
    byte discard = mySerial.read();
  }

  erro = mySerial.read();
  for (int i = 0; i<5; i++) {
    byte discard = mySerial.read();
  }
  status_digital = mySerial.read();
  int analogHigh = mySerial.read();
  int analogLow = mySerial.read();
  analogValue =  (analogLow + (analogHigh * 256))*1200/1024;
 }
}
```

A rotina *loop*, que ficará executando ciclicamente realiza primeiramente a checagem de pacotes na porta serial virtual. Lembrando que esta porta estará conectada diretamente ao Xbee Coordenador. A condição *if* testa, então, a existência de um pacote do tipo Remote Command Response proveniente do Xbee dispositivo final através do método *available* que retorna o número de bytes presentes nesta porta serial virtual. Se o numero de bytes é maior ou igual a 23 (um número escolhido ao acaso, poderia ser qualquer valor entre 5 e 25 aproximadamente dependendo do tamanho do pacote formado), um pacote completo está disponível.

O que ocorre em seguida é o descarte dos bytes que não interessam para esta aplicação, e seguindo a arquitetura da tabela 11, segue a leitura até o “Estado do comando”, que guarda esta informação na variável erro.

Os próximos 5 bytes lidos são descartados até se chegar à informação de estado das portas digitais. É neste byte que a informação real do motor ligado ou desligado chega através de uma entrada digital do Xbee Dispositivo Final conectado diretamente a um relé do motor, concretizando a diferença entre o registrador liga bomba e o registrador bomba ligada, explicados em tópicos anteriores.

Na sequência, o byte a seguir é a parte alta da conversão analógica/digital do sensor de nível do Xbee Dispositivo Final, conectado a uma entrada analógica do

mesmo. A parte baixa é então lida desta conversão e uma equação é realizada para converter este valor (lido em mV) para metros de coluna de água.

A próxima parte do código é apresentada abaixo:

```
if (erro == 0)
  {
    if (status_digital == 4){
           regs[MB_W1]=2; /*bomba ligada*/ }
    else { regs[MB_W1]=1; /*bomba desligada*/ }
regs[MB_W3]=analogValue;
  }

  if (analogValue>4 && regs[MB_W0] == 2) /*Modo automatico*/
  {
    regs[MB_W2]=1; /*liga bomba*/
  }
  if (analogValue<2 && regs[MB_W0] == 2)
  {
    regs[MB_W2]=0; /*desliga bomba*/
  }


    /* update dos registradores com as informacoes da IHM*/
   mbs.update(regs, MB_REGS);

    if(regs[MB_W2]==1 && last_indicator==0) {
       setRemoteState(0x5);
       last_indicator=1;}
       else{
         if(regs[MB_W2]==0 && last_indicator==1){
       setRemoteState(0x4);
       last_indicator=0;

         }
       }

  }
```

Lembrando primeiramente que a variável “erro” recebeu o parâmetro presente no campo “Estado do comando” do pacote 0x97. Somente quando este campo é zero a comunicação ocorreu sem erros, e a condição testada é justamente esta, atualizando os registradores somente em casos positivos.

O próximo teste de condição é para o registrador da bomba ligada. O relé proveniente do motor estará conectado na porta digital 3 do Xbee Dispositivo Final, e o byte recebido pela variável “status_digital” conserva o estado de todas as portas digitais do módulo, ou seja, ao receber o valor 100 em binário que é o equivalente ao

numeral 4 em decimal, significa que a entrada digital 3 está ativada, ou seja, o motor está operando e o registrador referente a ele é atualizado com o valor 2 (programado na IHM como sendo bomba ligada).

O registrador de nível também recebe o valor da leitura analógica convertida para metros testada a condição de ausência de erros.

Por último, a lei de controle simples é redigida, sendo que se o modo de controle for automático então, o controlador passará a avaliar a altura em metros do poço úmido e alterará o registrador W2 (liga bomba) conforme a altura. Neste caso o comportamento é de ligar quando a altura estiver maior que quatro metros e desligar quando a altura for menor que dois metros. Nesta fase o envio do pacote para o Dispositivo Final ainda não foi realizado, sendo que apenas o registrador foi atualizado.

O próximo passo é então utilizar o método *update* da classe ModbusSlave para sobrescrever os registradores com alguma possível alteração proveniente da IHM. Como na própria IHM o controle de qual alteração o operador pode ou não realizar, não é necessário realizar esta tarefa no Arduino, sendo que o operador sempre terá o direito de alterar algum registrador quando achar necessário.

O ultimo passo do *loop* é testar a condição de necessidade de ligar a bomba através da leitura do registrador W2, e nesta varredura cíclica assim que ativado o registrador, uma função denominada “setRemoteState” é ativada, com o envio do parâmetro 0x4 para desligar e 0x5 para ligar. Estes parâmetros são empíricos e definidos pelo próprio padrão API Zigbee e não podem ser alterados.

Vale ainda citar que para evitar sempre enviar o mesmo comando para o Dispositivo Remoto, um teste para analisar qual foi o último comando enviado é realizado através da variável “last_indicator”, sendo que o comando só é passado adiante se o último enviado não tenha sido idêntico.

A terceira e última parte do programa é apresentada na próxima página:

```
  void setRemoteState(int value) {  // passa ou um 0x4 ou um 0x5 para ligar ou desligar um pino
 // start delimiter
mySerial.write(byte(0x7E));
 // MSB
mySerial.write(byte(0x0));
 // LSB(o numero de bytes que restam no pacote
 // sem contar o checksum
mySerial.write(byte(0x10));
// comando 0x17
mySerial.write(byte(0x17));
//frame ID com valor 0 para não gerar resposta de identificacao
mySerial.write(byte(0x00));
// ID do recipiente
mySerial.write(byte(00));
mySerial.write(byte(0x13));
mySerial.write(byte(0xA2));
mySerial.write(byte(00));
mySerial.write(byte(0x40));
mySerial.write(byte(0x9E));
mySerial.write(byte(0x86));
mySerial.write(byte(0x21));
// endereço de 16 bits do recipiente, 0xFFFE se desconhecido
mySerial.write(byte(0xFF));
mySerial.write(byte(0xFE));
// 0x02 aplica mudanças imediatamente, é o Remote comand option
mySerial.write(byte(0x02));
// nome do comando em ASCII
mySerial.write(byte('D'));
mySerial.write(byte('1'));
// parâmetro do comando
mySerial.write(byte(value));
// checksum
long sum = 0x17 + 0x13 + 0xA2 + 0x40 + 0x9E + 0x86 + 0x21 + 0xFF + 0xFE +0x02 + 'D' + '1' + value;
// calcula o valor do checksum
mySerial.write(byte(0xFF - ( sum & 0xFF)));
delay(10); // evita sobrefluxo da porta serial
}
```

Esta é a função do código do Arduino que irá criar o pacote Remote AT Command Request (0x17) e enviá-lo através da porta serial virtual “mySerial” até o Xbee Coordenador, que por sua vez repassará para o Dispositivo Final da rede Zigbee.

Cada byte deve ser inserido um por um através do método “write”, que escreve o mesmo na porta selecionada. O primeiro byte escrito é o 0x7E, que é o *Start Delimiter* da tabela 10. Depois é escrito o *Lenght* com seus dois bytes, o mais e menos significativo em hexadecimal, neste caso 0x0 para o MSB e 0x10 para o LSB. O próximo byte é o *Frame Type*, ou seja, o comando 0x17 referente ao pacote Remote AT Command Request.

O próximo byte, de *Frame ID* é setado em 0x00 para não gerar resposta de configuração de *hosts*. Os próximos oito bytes são o endereço de 64 bits do

Dispositivo Final (figura 47). Como o *Frame ID* foi setado para não gerar resposta, o endereço de 16 bits do Dispositivo Final não é conhecido, e por isso estes próximos dois bytes recebem o valor 0xFFFE, para quando não se conhece este endereço.

O próximo byte, o *Remote Command Options*, é setado com o valor 0x02 para que as alterações tenham efeito imediatamente. Com isso resta ainda montar os bytes de *Comando AT* com os caracteres ASCII ‘D’e ‘1’, os quais identificam que as mudanças requisitadas por este pacote devem ocorrer na saída digital 1 do Dispositivo Final, a qual estará ligada no circuito de ativação do motor, este sendo explicado em tópicos ainda por vir.

O ultimo byte montado no pacote é o *Checksum*. Segundo o protocolo Zigbee, para calcular este byte é preciso adicionar todos os bytes presentes no *Frame Data*, ou seja, todos os bytes menos o *Start Delimiter* e *Lenght*, mantendo apenas os 8 bits menos significativos e subtrair este resultado do valor 0xFF. Manter os 8 bits menos significativo no código acima é atingido ao realizar o operador “&” com o valor 0xFF e o resultado subtraído do valor 0xFF. Esta etapa é necessária para as rotinas de checagem automática dos módulos Xbee, garantindo que nenhum pacote seja descartado indevidamente.

Com isso fica apresentado o código do Arduino utilizado para a solução de automação da EEE. Como prometido anteriormente, o próximo tópico irá tratar das alterações necessárias no código para uma possível implementação para os aeradores. A ênfase maior deste capítulo está sendo e continuará sendo na implementação para uma EEE, pois a construção física do projeto piloto (ainda a ser apresentada) deu-se em uma planta de EEE devida a necessidade de menos equipamentos e conclusão de projeto com um orçamento menor.

5.2.5: Alterações do código do Arduino para uma possível implementação em aeradores.

Uma das premissas que foi prometida neste projeto é a facilidade de alterações para a implementação do projeto em uma EEE ou um aerador. Neste tópico ficará provada esta facilidade na parte do código de programação. No tópico que apresentará o projeto piloto na sua concepção estrutural ficará provado a facilidade de alteração estrutural.

A primeira alteração necessária de ser realizada é a respeito do número de dispositivos finais. Na programação do Arduino para a EEE ocorre a suposição de apenas um Dispositivo Final integrava a rede, com a possível presença ou não de roteadores, os quais não alteram a programação do Arduino.

Em uma rede de aeradores, informações a respeito de diversos dispositivos finais chegam ao Arduino, e o mesmo necessita identificar de qual dispositivo final é cada mensagem. E isto, felizmente, é muito fácil de realizar. Cada pacote 0x97 vem com o campo *Endereço de 64 bits do Dispositivo Final*, um identificador único para cada módulo Xbee.

Na função *loop* do código apresentado anteriormente, vários bytes que não interessavam para o programa eram descartados, a diferença agora é que alguns destes bytes importam, especialmente os bytes de número 5 à 13, os quais contém esta informação que deve ser armazenada em uma variável interna.

Outra alteração necessária é o acréscimo no número de registradores no código do Arduino, sendo que cada aerador presente na rede adicionará cinco registradores extras, sendo um de enumeração, assim como o caso da EEE.

Quando for chamada a função *setRemoteState*, presente no código da EEE, além de passar como parâmetro o valor que a saída digital deve assumir, o endereço de 64 bits colhido anteriormente também deve ser passado, para que o pacote seja montado com este endereço.

Estas são basicamente todas as alterações necessárias no código do Arduino, sendo que seu núcleo é basicamente o mesmo.

5.3: Programando os módulos Xbee para a solução final.

Realizada a programação do Arduino na solução final, foi necessário programar os dispositivos Xbee para formação da rede Xbee. Esta programação se divide basicamente em três partes:

- Programação do Xbee Coordenador.
- Programação dos Xbee Roteadores.
- Programação do Xbee Dispositivos Finais.

Vale citar aqui que a programação destes dispositivos não se altera em absolutamente nada para uma aplicação em EEE ou aerador, pois o que estes dispositivos formarão é uma rede Zigbee de comunicação sem fio, sem se importar com quais periféricos estão conectados.

Nos próximos tópicos será apresentada esta programação individual de cada elemento.

5.3.1: Programação do Xbee Coordenador.

Sempre existe apenas um dispositivo Coordenador em uma rede Zigbee, e o utilizado na solução final presente neste projeto teve que ser parametrizado para o funcionamento correto do sistema de automação. Os parâmetros abaixo foram os alterados:

Parâmetro	Valor
PAN ID	2001
SC	1FFE
SD	3
ZS	2
NJ	FF
DH	0
DL	FFFF
NI	COORDENADOR
NH	1E
BH	0
DD	30000
NT	3C
NO	2
PL	4
PM	1
EE	0
EO	0
BD	3
NB	0
SB	0

Tabela 12- Programação do Xbee Coordenador.

Justificando um pouco os parâmetros, o PAN ID (identificador da rede) utilizado foi o 2001 por arbitrariedade, pois todos os exemplos que criei anteriormente à solução final tinham este mesmo valor. O parâmetro SC (canais escaneados) fica setado com o maior valor possível, 1FFE (valor hexadecimal) que significa que todos os canais são escaneados à procura de um canal livre para inicialização da rede. O parâmetro SD é responsável por parametrizar o tempo de escaneamento comentado acima, e o valor 3 significa trinta segundos.

O parâmetro ZS indica qual perfil de protocolo Zigbee o dispositivo representa, e o numeral dois indica um perfil Zigbee-PRO, modelo de todos os módulos presentes neste trabalho. O parâmetro NJ indica por quanto tempo o Coordenador deve aceitar conexões de novos módulos querendo participar da rede e o valor FF indica que o dispositivo sempre permitirá que novos componentes sejam adicionados. Os parâmetros DH e DL servem para limitar a conexão do Coordenador com um único outro dispositivo através de seu endereço de 64 bits inseridos nestes parâmetros. Ao se utilizar os valores 0 e FFF, o programador indica que o Xbee Coordenador deve se comunicar com todos os dispositivos da rede, sem nenhuma restrição. O pacote criado pelo Arduino que identificará o destinatário, conforme visto no tópico referente a programação do Arduino.

O parâmetro NH indica quantas vezes a mesma mensagem pode passar por diferentes componentes (geralmente por causa de erros e eventuais retransmissões) até que a mesma seja descartada. Inserindo o valor 1E neste parâmetro para o projeto, fica explicitado que este número máximo é de 30 saltos. Como os tempos de transmissão dos módulos são extremamente baixos, 30 saltos levariam cerca de 2 segundos para serem realizados na distância máxima entre roteadores.

O parâmetro BH indica o alcance de transmissão que o aparelho deve atingir. Ao ser programado com o valor 0, o mesmo tem significado de sempre ter o alcance máximo. Este parâmetro é interessante para aplicações que necessitam de economia de energia, o que não é o caso deste trabalho.

Já o DD é o identificador de tipo de dispositivo, identificando versões diferentes de equipamentos. Este parâmetro é importante quando se deseja que alguns dispositivos da rede não se comuniquem entre si por possuírem versões diferentes de *firmware*, o que poderia causar conflitos. No caso deste projeto, como

todos os dispositivos possuem o mesmo *firmware* e são do mesmo modelo, todos os módulos receberão o valor 30000, um valor sem um propósito específico.

O parâmetro NT é responsável pela emissão de uma confirmação de entrada na rede quando dispositivos solicitam a mesma. O valor do parâmetro indica o tempo que o Coordenador tem para responder este pedido, sendo o valor 3C um padrão do fabricante que não foi alterado.

O NO por sua vez é responsável pelo gerenciamento de modificação do parâmetro DD. Programado com o valor 2, se o Coordenador descobre um dispositivo que deseja se juntar a rede e possui o mesmo *firmware* mas com o parâmetro DD diferente, o Coordenador tem a capacidade de inferir sobre o DD do dispositivo conectado, alterando o mesmo para o valor de seu DD.

O PL é o nível de energia do sistema, sendo que o valor 4 significa que o mesmo irá gastar toda a energia disponível, gerando o máximo de potência.

PM é uma espécie de força adicional, aumentando o ganho do equipamento em torno de 1dB com o sacrifício de alguns miliwatts de potência e é ativado quando o parâmetro se encontra em 1.

EE cuida da parte de criptografia das mensagens, porém neste projeto a mensagem em si não contém dados sigilosos e por isso não é utilizada, recebendo o valor 0.

EO é um parâmetro auxiliar do EE que deve também receber o valor 0 se não for utilizada a criptografia.

BD é a taxa de transmissão de dados e quando apresenta o valor 3 significa uma taxa de 9600 bps, a utilizada por todo este projeto. NB por sua vez indica a paridade e o parâmetro zero infere uma condição de sem paridade, a mesma de todos os outros dispositivos deste projeto. Por fim, SB é o bit de parada, sendo que o valor zero indica 1 bit de parada.

O *firmware* utilizado neste projeto foi o XBP-24 ZB, sendo que a *Function Set* do Coordenador foi a de Zigbee API Coordinator.

5.3.2: Programação do Xbee Roteador.

O Roteador é o elemento capaz de estender o alcance da rede Zigbee, sendo responsável por criar novas rotas de comunicação entre Coordenador e Dispositivos Finais. Neste projeto a configuração dos Roteadores foi a seguinte:

Parâmetro	Valor
PAN ID	2001
SC	1FFE
SD	3
ZS	2
NJ	FF
DH	0
DL	FFFF
NI	ROTEADOR
NH	1E
BH	0
DD	30000
AR	FF
NT	3C
NO	2
PL	4
PM	1
EE	0
EO	0
BD	3
NB	0
SB	0

Tabela 13- Programação do Xbee Roteador.

A programação é muito semelhante ao Coordenador, sendo que no Roteador são indispensáveis os parâmetros DH e DL serem 0 e FFFF respectivamente, pois o Roteador deve servir para aumentar o alcance de uma mensagem, transmitindo a mesma para todos os dispositivos da rede, sendo que para tal estes parâmetros necessitam estes valores.

O parâmetro adicional na programação é o AR, parâmetro que assim que descobre uma rota específica para um determinado equipamento, este utilizará a mesma rota sempre. Ao se especificar o valor FF, esta descoberta é desativada. Isto é importante, pois o ambiente em que os roteadores estão instalados não é inerte,

podendo sofre alterações. Os milissegundos que possa vir a ser ganho com a rota fixa não vale o prejuízo possível de um pacote nunca chegar a seu destino.

O firmware utilizado para o Roteador foi o XBP-24 ZB, sendo que a Function Set foi a de Zigbee API Router.

5.3.3: Programação dos Xbee Dispositivos Finais.

Os Xbee Dispositivos Finais são os que ficarão na ponta final do sistema de automação, sendo responsáveis pela leitura dos sensores e conexão com os atuadores do sistema. A configuração do Xbee Dispositivo Final é a seguinte:

Parâmetro	Valor
PAN ID	2001
SC	1FFE
SD	3
ZS	2
NJ	FF
JN	0
DH	0
DL	0
NI	DISPOSITIVO FINAL
NH	1E
BH	0
DD	30000
NT	3C
NO	2
PL	4
PM	1
EE	0
EO	0
BD	3
NB	0
SB	0
D0	2
D1	4
D3	3
IR	100

Tabela 14- Programação do Xbee Dispositivo Final.

Alguns parâmetros de programação variam em relação aos anteriores e estes serão discutidos com mais detalhes. Por exemplo, o parâmetro DH e DL de um Dispositivo Final sempre serão 0, pois o objetivo final deste equipamento é a comunicação com o Coordenador, que sempre assume este endereço.

O parâmetro JN permite que o Dispositivo Final envie um *frame* com suas informações iniciais assim que entra na rede. Para este projeto este parâmetro foi desativado (valor 0), pois é redundante em relação ao parâmetro IR, como será demonstrado nos próximos parágrafos.

O parâmetro D0 altera o pino D0 do módulo Xbee para entrada analógica, o qual será conectado o sensor (valor 2), o parâmetro D1 altera o pino D1 do Xbee para saída digital, onde será inserido o circuito que irá alimentar o motor (valor 4) e o parâmetro D3 irá alterar o pino D3 do Xbee para entrada digital, a qual receberá informação de bomba ligada (valor 3).

Por fim, o parâmetro IR serve para enviar uma amostra das portas analógicas e digitais do Dispositivo Final a cada período de tempo especificado. No caso deste projeto, este período é de 100 milissegundos. É por este motivo que o parâmetro JN é supérfluo, pois o dispositivo sempre envia informações a cada período de tempo.

Explicadas as programações de IHM, Arduino e Xbee utilizados neste projeto, a parte lógica da solução está completa, faltando apenas apresentar a parte física, de construção do sistema e todas as soluções encontradas para tal feito. O próximo tópico começará a descascar este assunto.

5.4: Construção física dos equipamentos.

Este projeto incluiu, além da programação de toda lógica do sistema de automação proposto, a construção de um modelo físico, baseado no problema da automação da EEE, para que o mesmo pudesse ser testado na planta de simulação de Elevatória de Esgoto presente no CIOM, em Florianópolis, Santa Catarina.

A construção deste modelo do sistema de automação pode ser dividida em três partes, como mesmo defini:

- Construção do kit IHM + Arduino + Zigbee Coordenador.
- Construção do kit Roteador.

- Construção do kit Dispositivo final.

Cada um destes kits tem sua própria configuração e dificuldades de construção e são aprofundados nos próximos tópicos.

5.4.1: Kit IHM + Arduino + Zigbee Coordenador.

Este conjunto de equipamentos, comunicantes entre si, são representados por quatro elementos individuais. São eles:

1. IHM Beijer H-T80c.
2. Conversor RS-485/Serial TTL.
3. Arduino Uno Rev3.
4. Xbee S2-PRO.

Para interligar a IHM e o conversor foi criado um cabo RS-485 o qual poderia ser conectado à porta COM1 da IHM e a outra ponta no barramento A e B do conversor. A figura abaixo exemplifica melhor a situação.

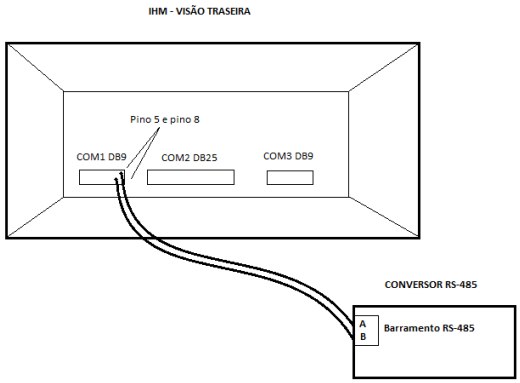

Figura 72- Esquema da ligação entre IHM e conversor.

O cabo é conectado entre barramento A e B do conversor e a porta de comunicação COM1 da IHM, sendo esta do tipo DB9 (nove pinos).

Para interligar conversor e Arduino, cinco fios são necessários. São eles:

1. Um fio entre o pino 5V do Arduino e o borne Vcc do conversor.
2. Um fio entre o pino Rx do Arduino e o borne Rx do conversor.
3. Um fio entre o pino Tx do Arduino e o borne Tx do conversor.

4. Um fio entre o pino 6 do Arduino (habilitação) e o borne EN do conversor.
5. Um fio entre o pino GND do Arduino e o borne GND do conversor.

A figura abaixo exemplifica melhor a situação:

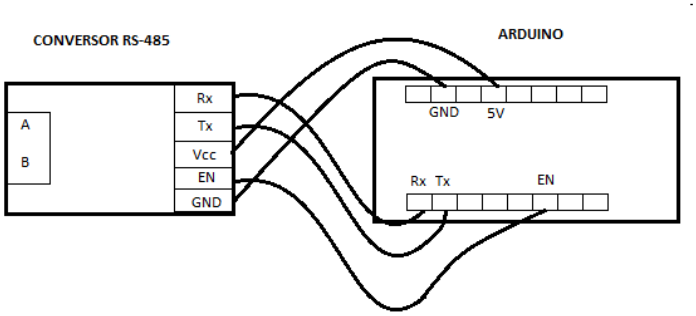

Figura 73- Esquema de ligação entre conversor e Arduino.

Uma das vantagens de se utilizar o Arduino é que o componente serve de fonte de alimentação tanto para conversor como para Xbee Coordenador, como será apresentado mais adiante.

Para interligar Xbee Coordenador e Arduino são necessários quatro fios, sendo eles:

1. Um fio entre o pino 3.3V do Arduino e o pino Vcc do Xbee.
2. Um fio entre o pino GND do Arduino e o pino GND do Xbee.
3. Um fio entre o pino 2 do Arduino (Rx virtual) e o pino DOUT do Xbee.
4. Um fio entre o pino 3 do Arduino (Tx virtual) e o pino DIN do Xbee.

A figura abaixo ajuda a entender melhor esta ligação:

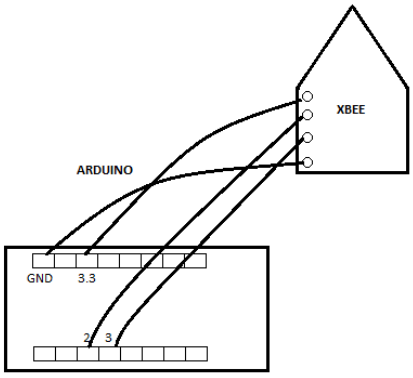

Figura 74- Esquema de ligação entre Arduino e Xbee Coordenador.

O Arduino neste caso também serve de fonte para o Xbee Coordenador, sendo que o kit em questão necessita apenas de uma fonte com duas saídas: Uma 12V DC para o Arduino e outro 24V DC para a IHM.

O sistema como um todo é representado abaixo:

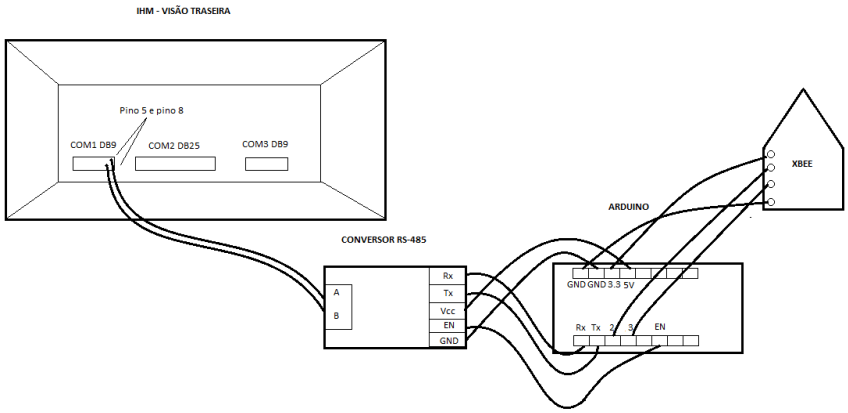

Figura 75- Esquema de ligação do kit.

Definido o modelo de construção do primeiro kit, diversas versões preliminares do quadro com os componentes foram sendo idealizadas e aprimoradas, sempre tentando atender o pressuposto de simplicidade, acesso fácil às partes para manutenção e organização geral. Com isso, a versão final do kit é apresentada abaixo:

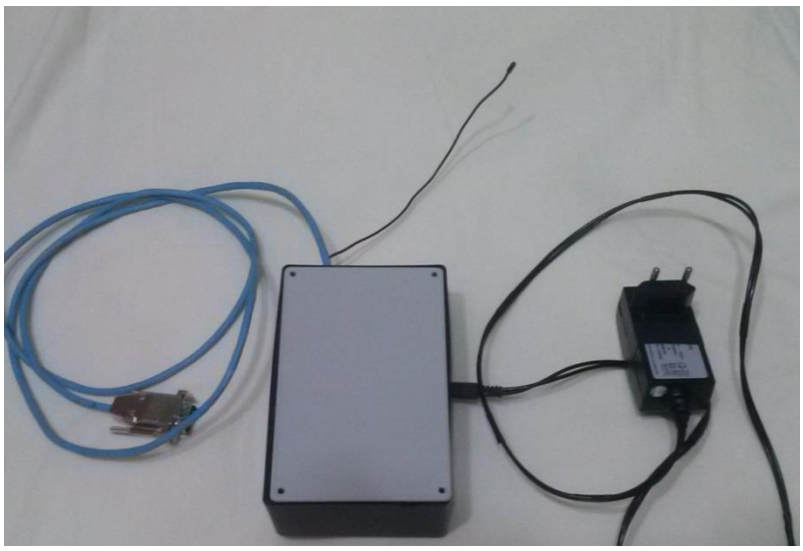

Figura 76- Kit implementado.

Nesta visão externa do kit, pode-se observar em azul com ponta prateada o cabo de conexão com a IHM e no lado direito da foto a fonte de alimentação do sistema. Ainda no canto superior da imagem é possível ver a antena do Xbee Coordenador, a qual foi trocada por uma maior (vide imagem) para ficar posicionada no exterior da caixa plástica. A caixa em si é uma caixa padrão para instalações elétricas com medidas 20x10x10 cm, podendo ser aparafusada em paredes ou mesmo escorada em uma plataforma ou mesa.

A imagem abaixo apresenta o interior do kit:

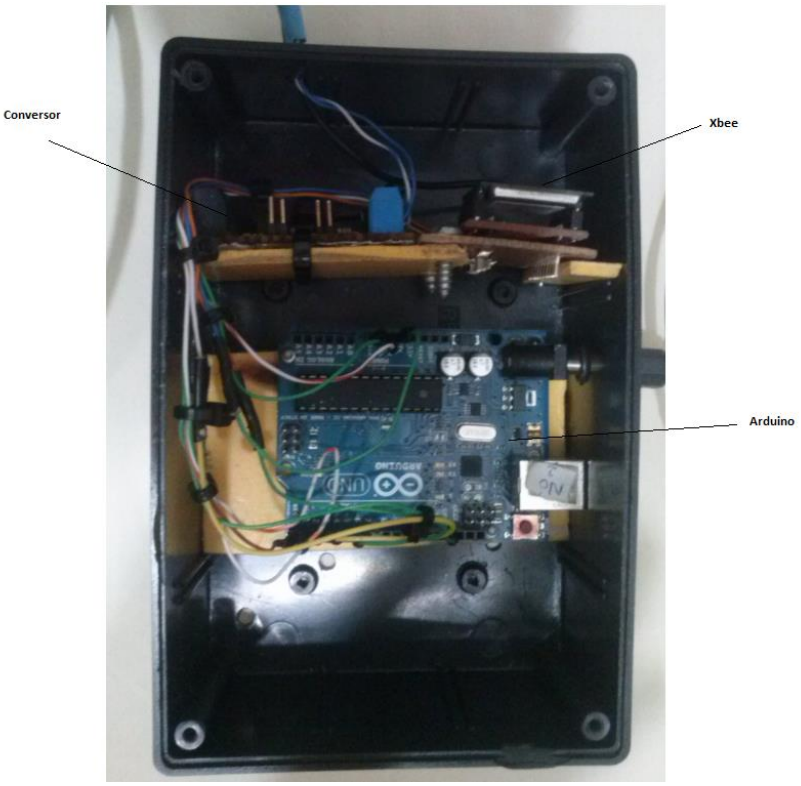

Figura 77- Visão interna do kit.

No centro da imagem é possível ver o Arduino Uno com a ligação conforme figura 75. No canto superior esquerdo da caixa (de lado) encontra-se o conversor caseiro feito para converter RS-485 em serial e no canto esquerdo alto (de lado) encontra-se o módulo Xbee que criará a rede Zigbee da solução.

Todas as ligações elétricas no kit são ou soldadas, ou conectadas por bornes para evitar desconexão acidental, ou quando ligadas diretamente no pino (caso do Arduino), os fios foram cobertos com conectores termo retráteis, aumentando a área de contato para evitar as desconexões citadas.

5.4.2: Kit Roteador.

O kit roteador por sua vez é composto dos seguintes componentes:

1. Módulo Xbee Roteador.
2. Fonte de Alimentação.

O objetivo do kit Roteador é prover o aumento do alcance da rede Zigbee quando necessário, e por isso não apresenta nenhuma entrada ou saída analógica/digital ou qualquer outro periférico.

Como a potência consumida pelo módulo Xbee é bastante baixa, neste kit a inovação está por conta da fonte de alimentação. Seria bastante interessante a utilização deste kit se o mesmo não tivesse que ficar atrelado sempre a uma tomada elétrica para fornecer sua energia. O kit Roteador, idealmente, deveria ser posicionado em áreas onde existia deficiência ou ausência total do sinal da rede Zigbee, e sua fonte deveria se adaptar a esta necessidade.

Com isso, uma das fontes mais interessantes e fáceis de serem concebidas é através da utilização de pilhas. É bastante comum em lojas especializadas em eletrônica a venda de conectores para pilhas, geralmente com o propósito de trocar conectores antigos em brinquedos em houve corrosão da pilha, mas neste projeto pude adaptar um destes mesmos aparatos para criar uma fonte para Xbee.

Como a alimentação do Xbee recomendada é de 3.3V, são necessárias duas pilhas AA em série para fornecer esta energia. A figura abaixo apresenta um conector genérico deste tipo, exemplificando a situação:

Figura 78- Conector de pilhas.

Com isso, basta conectar o terminal de cor vermelha no pino Vcc do Xbee Roteador e o terminal de cor preta no pino GND do Xbee.

O modelo final do kit utilizado neste projeto é apresentado nas próximas duas figuras, uma da visão externa e outra da interna:

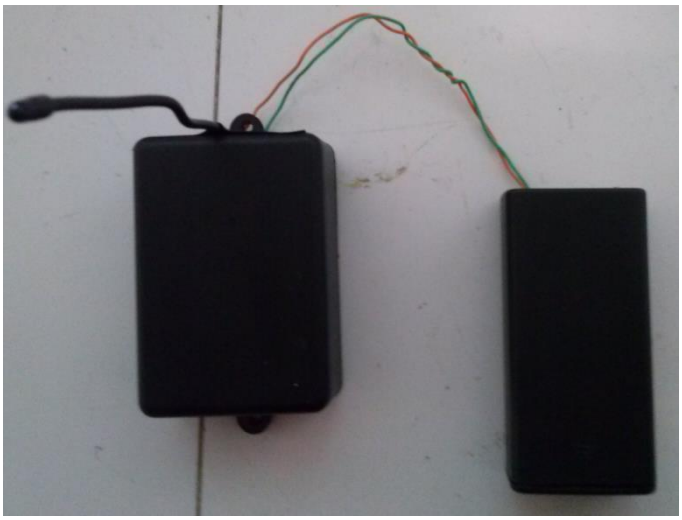

Figura 79- Visão externa do kit.

Figura 80- Visão interna do kit.

Na visão externa, é possível observar no lado direito a fonte à pilha utilizada como alimentação do Xbee Roteador, e na visão interna é observado a simplicidade do kit, com apenas o Xbee Roteador alimentado pronto para rotear mensagens provenientes da rede Zigbee.

Este modelo se mostrou bastante versátil e o consumo de energia das pilhas foi muito baixo, sendo que no tópico específico de validação da solução este tema retornará.

5.4.3: Kit Dispositivo Final.

O kit Dispositivo Final tem a função de receber o sinal da sonda de nível do poço úmido e enviar o comando ao atuador (conjunto bomba-motor) para o mesmo alterar seu estado conforme comando do Arduino. Nenhuma destas ligações é trivial e necessita circuitos adicionais para o funcionamento.

Primeiramente, para acionar o motor é utilizada uma saída digital do Xbee Dispositivo Final. Estas saídas possuem tensão máxima de 1V DC. Para acionar o motor presente na planta da EEE do CIOM, é necessário uma tensão de 220V AC na bobina de uma contatora, a qual por sua vez fornecerá a tensão trifásica 380V AC necessária para o motor. Converter esta tensão do Xbee para a tensão da bobina da contatora necessita algumas manobras e um conhecimento de circuitos eletrônicos.

É então necessário uma fonte de 12V DC para alimentar um relé, cuja função será logo apresentada. Esta fonte não pode alimentar diretamente o Xbee Dispositivo Final, pois é um valor de tensão muito alto. Para evitar o uso de duas fontes, foi inserido um circuito integrado regulador de tensão, no caso um LM78L33. Quando submetido a uma tensão de entrada superior a 6V DC, como no caso deste projeto, com a fonte de 12V DC, este CI regula a tensão para uma saída de 3.3V DC, exatamente o necessário para o Xbee.

O relé comentado anteriormente deve ser um relé de acionamento 12V DC com contatos secos de 220V AC, ou seja, quando excitado com uma tensão de 12V DC na sua bobina, o mesmo fecha contato liberando a tensão de 220V AC disponível em seu contato seco.

Com este contato seco, é possível ligar a saída do relé diretamente em um terminal da bobina da contatora, fechando um circuito 220V.

O problema está então em como atingir a excitação da bobina do relé com a saída da porta digital do Xbee. Este problema é resolvido ao se soldar em uma das

pernas da bobina do relé o Vcc de 12V DC da fonte de alimentação, e fechando o circuito com o negativo de um transistor NPN.

Este transistor é então conectado na saída digital do Xbee, com isso ao receber o sinal do Xbee, o transistor se polarizará, deixando corrente circular da região N para a região P. Este terminal N é então conectado na outra bobina do relé, ativando o mesmo quando neste caso.

Embora este circuito seja um pouco complexo, a figura abaixo visa ajudar na compreensão:

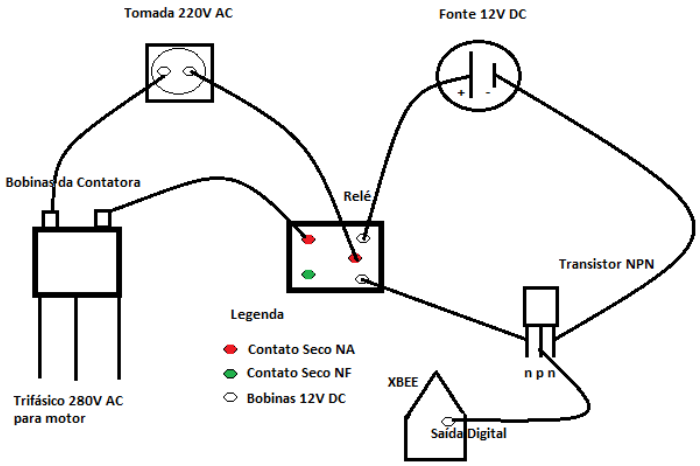

Figura 81- Esquema de ligação do circuito de acionamento do motor.

Assim se resolve o problema de acionamento do motor. Para leitura do sinal do sensor de nível, é necessária uma conversão de medidas, pois o sensor tem um padrão de corrente entre 4-20 mA e o Xbee lê em suas entradas analógicas tensão de 0-1V DC. Esta conversão é feita utilizando a lei de Ohm que diz:

$$V = R.i$$

Ou seja, tensão é o produto de resistência e corrente. Como tenho uma medida de corrente máxima de 20 mA e minha tensão máxima é de 1V, substituindo na equação acima:

$$1 = R * 20/1000$$

$$R = 500\ \Omega$$

Com isso, ao se adicionar um resistor de 500 ohms no sistema, quando a corrente for 20 mA a leitura na porta analógica do Xbee será de 1V. A figura abaixo mostra a ligação elétrica realizada neste trabalho para tal feito:

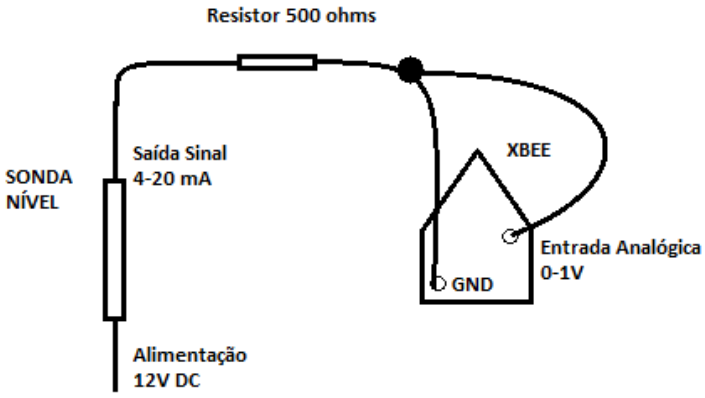

Figura 82- Conversão corrente-tensão.

Este circuito gera a tensão de 0-1V DC através da ligação entre sinal de sonda de nível e GND do Xbee passando por um resistor de 500Ω. Esta tensão é então ligada até a entrada analógica do equipamento, o qual realizará a leitura.

O kit Dispositivo Final montado para a instalação na planta de teste EEE CIOM possui todos estes circuitos citados acima. A imagem da versão final do kit é apresentada abaixo:

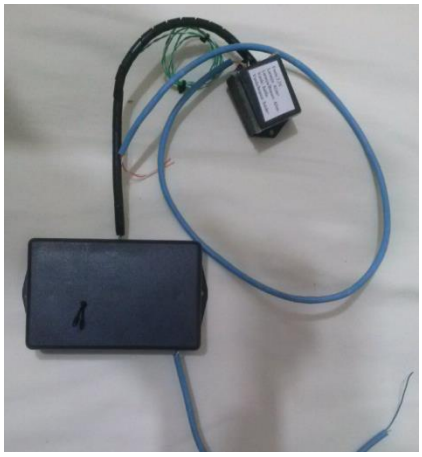

Figura 83- Visão externa.

Nesta figura é possível ver a antena do módulo Xbee Dispositivo Final (na caixa plástica maior), bem como o regulador de tensão de 3.3V (caixa plástica

menor). Os dois cabos azuis são onde são conectados o sensor de nível, o terminal da bobina da contatora e a fonte de alimentação de 12V.

O interior do kit é mostrado na próxima imagem:

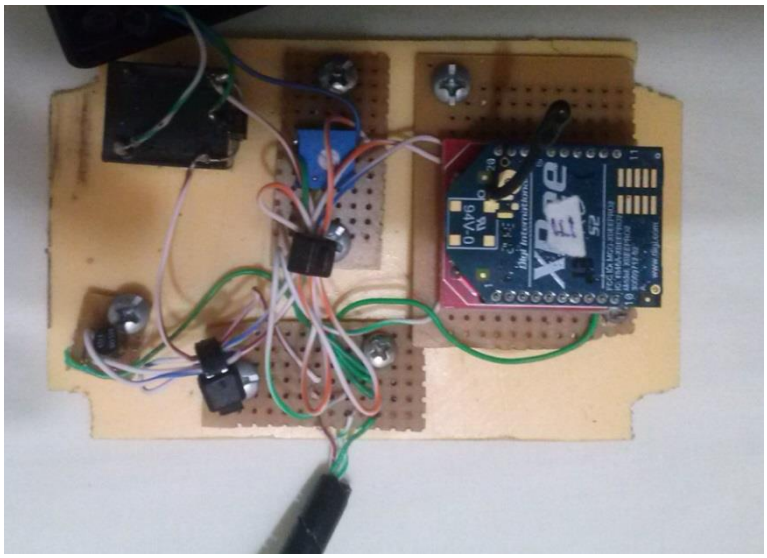

Figura 84- Visão interna.

Nele é possível ver o módulo Xbee Dispositivo Final (marcado com um ‘E’), além do relé (canto superior esquerdo), o transistor (canto inferior esquerdo) e o resistor, neste caso um potenciômetro ajustado para 500Ω (em azul).

Com isso foram apresentados todos os kits que fizeram parte da construção física deste projeto e que foram implementados na planta teste de EEE.

Resta ainda, neste capítulo, analisar como a solução foi testada, avaliada e validada, servindo de iniciação para o capítulo 6, onde serão discutidos os resultados.

5.5: Teste, avaliação e validação da solução.

Cada um dos elementos presentes na solução em automação apresentada neste documento foi devidamente testado para verificar seu devido funcionamento. Um dos elementos que mais gerou intriga e dúvidas foi o conversor RS-485 caseiro que resolvi montar. Embora os componentes e arquitetura do circuito sejam fáceis

de reproduzir, nos primeiros testes realizados o componente parecia nunca funcionar como esperado.

Encontrar o foco do problema nesta situação às vezes era muito complicado, pois todas as tecnologias envolvidas eram praticamente desconhecidas por mim na época, e nunca conseguia descobrir se o erro estava na programação da IHM, do Arduino, do Xbee ou alguma ligação equivocada entre os componentes.

Então, para descobrir se meu conversor realmente funcionava como esperado, resolvi retirar os elementos que ainda tinha dúvida da equação. Utilizando o supervisório da CASAN, um aplicativo em SCADA que ajudei a implementar durante meu estágio obrigatório, pude conectar um Arduino remotamente através de uma rede RS-485 com o servidor utilizando meu conversor. Este teste provou que o componente funcionava, bastando agora avaliar o mesmo.

Para a avaliação do componente, resolvi comprar o conversor comercial RS-485 apresentado na figura 60. Tendo com o que comparar, descobri que, assim como esperado, o desempenho do conversor caseiro era exatamente o mesmo do conversor comercial, validando o componente.

Este foi apenas um pequeno teste para avaliar uma parte muito específica da solução. De maneira mais generalista, um teste que foi realizado com bastante insistência foi o teste da IHM, onde um dos requisitos da interface era a fácil adaptação dos operadores.

Como próximo ao meu posto de trabalho geralmente era fácil encontrar funcionários da CASAN da área de operação, sempre conseguia que algum deles testasse as telas da IHM que ia criando, recebendo o *feedback* dos mesmos sobre prováveis melhorias. A versão final, apresentada neste trabalho, contou com um teste onde os avaliadores poderiam dar uma nota de zero a dez para três quesitos selecionados: Entendimento da tela, facilidade de atuação e clareza dos resultados.

Desta maneira eu pude especificar mais meu projeto para o gosto dos operadores, pois desde o princípio eu já sabia que seria muito difícil implantar a ideia da automação nestes.

A versão final das telas da IHM ficou menos polida do que eu gostaria, mas o resultado agradou muito mais a quem realmente interessava: os utilizadores.

Outro teste que repeti por diversas vezes durante a elaboração deste trabalho foi o de distância de comunicação entre os módulos Xbee. Era necessário provar com meus próprios olhos as especificações técnicas de distância em espaços fechados e espaços abertos. E para minha surpresa, em muitos destes testes o desempenho de sinal que atingi foi maior do que o especificado pelo fabricante. Muitos de meus colegas também ficaram bastante impressionados com este comportamento, ainda mais comparado com outra tecnologia de transmissão sem fio que estávamos testando na época.

Um teste interessante que foi realizado também foi o de consumo de energia dos módulos Xbee, principalmente no dispositivo Roteador que funcionava à pilha. O teste realizado foi o de se medir a tensão em aberto das pilhas no início e após duas semanas de continuo funcionamento do aparelho repetir esta medição.

O consumo de energia neste caso foi tão diminuto que foi menor que a escala do multímetro utilizado para a leitura. Como os resultados foram excelentes, a alimentação com pilha ficou presente na implementação final.

O sistema final, apresentado neste capítulo, também foi testado extensivamente na planta emuladora de EEE do CIOM, pois era uma verdadeira oportunidade poder testar o equipamento em um ambiente muito próximo do real sem precisar sair do próprio laboratório de testes, e esta planta foi fundamental para o desenvolvimento da solução final. Até o fechamento deste documento, a planta estava sendo controlada por duas semanas com o sistema de automação apresentado neste capítulo (modo automático) sem nenhum evento de extravasamento da mesma.

O objetivo de todos estes testes era conseguir simular o máximo de situações diferentes possíveis de serem encontradas em prática, pois é impossível prever tudo que possa acontecer, e observar como o sistema reage quando situações adversas acontecem faz parte de um bom desenvolvimento de projeto.

Uma destas fatalidades ocorreu em um dia que ocorreu uma queda de energia no local em que o sistema estava instalado. Quando a energia retornou o sistema não se comportou como desejado e ficou em um estado travado onde não

conseguia mais tomar decisão alguma de controle. O erro neste caso foi percebido e pode ser consertado na versão final apresentada no documento.

É por esses acontecimentos que a fase de teste foi considerada a mais importante deste projeto, sem ela não há como avaliar e observar o comportamento do sistema e muitas mudanças de projeto veio desta fase.

5.6: Considerações finais do capítulo.

Neste capítulo foi apresentada a implementação do sistema de automação pretendida por este projeto. Foi mostrado como programar e interligar cada componente e as diferenças da solução para uma implantação em EEE ou aeradores.

Ainda foi apresentado o projeto físico instalado na planta de emulação de EEE do CIOM, apresentando os três kits concebidos e suas nuanças. No final do capítulo foi falado um pouco dos testes realizados para validação da solução.

O objetivo era além de apresentar o projeto em questão, fomentar o leitor para o leque de possibilidades que estas tecnologias abrem para projetos de automação, com a intenção de propagar o conhecimento e as alternativas que estão adentrando no mundo da automação para os velhos estigmas praticados por muitas empresas.

No próximo capítulo serão apresentados e discutidos os resultados obtidos com este projeto, analisando se o mesmo atingiu os objetivos propostos no inicio do documento, com o intuito de fornecer um encerramento ao documento e ao projeto.

Capítulo 6: Resultados.

Neste capítulo serão apresentados os resultados atingidos com a implementação da solução final. Será analisado o casamento ou não das especificações com o produto final, analisando em que áreas o sistema se mostrou mais eficiente que o esperado e em que áreas o sistema não atingiu os objetivos.

A análise destes resultados poderá então, finalmente, encerrar a última fase da elaboração deste documento e consequentemente do projeto, restando então apenas as considerações finais.

Primeiramente será apresentada a implementação que foi utilizada como referência para estas análises, realizada na planta de simulação de Estação Elevatória de Esgoto do CIOM.

6.1: Implementação na planta de simulação CIOM.

A planta de simulação do CIOM, utilizada para os testes, simula a entrada de esgoto no poço úmido através da entrada variável de água em um tubo de 10 metros de altura. Inserido neste tubo se encontra uma sonda capaz de medir o nível da água. A saída do esgoto é simulada por um motor, que assim como nas EEEs reais, é acionado quando o nível atinge uma faixa máxima. A figura na próxima página apresenta o tubo da planta de simulação:

Figura 85 – Tubo para simulação de poço úmido.

O sistema de saída do tubo constituído pelo motor é apresentado abaixo:

Figura 86 – Motor da planta de simulação.

Com este sistema é então possível simular de maneira fácil o comportamento de uma EEE da CASAN.

6.1.1: Aplicando a solução na planta de simulação.

Finalmente, foram integrados os três kits desenvolvidos neste projeto na planta do CIOM para analisar o comportamento do sistema completo, apresentando os resultados.

O primeiro kit, constituído da IHM, conversor, Arduino e Xbee foi montado em uma bancada distante do motor. A imagem abaixo apresenta o kit montado:

Figura 87- Kit IHM implementado.

Em outra localidade, próxima ao motor, foi instalado o kit Dispositivo Final, ligado diretamente a uma contatora para acionamento do motor. A figura abaixo apresenta esta configuração:

Figura 88 - kit Dispositivo Final implementado.

A contatora (no canto esquerdo da tela) é a responsável pelo acionamento do motor trifásico mostrado anteriormente.

Ainda nesta aplicação foi utilizado o kit Roteador, instalado em um edifício entre os outros dois kits, estendendo a cobertura da rede. Com esta montagem foram avaliados os resultados, apresentados no próximo tópico.

6.2: Casamento de especificações e solução final.

Uma das especificações apresentadas neste projeto era a não interferência com outros sistemas já instalados. Isto se verifica na planta de simulação que pôde continuar a ser utilizada manualmente por outros funcionários para testes mesmo com a aplicação da solução deste documento, facilitando inclusive o acionamento da mesma através da IHM.

De maneira mais generalista, esta conciliação entre tecnologias existentes e implementadas nesta solução também se verifica. A criação da rede Zigbee só ocorre em canais de comunicação livres e não causa interferência em outras possíveis aplicações de radiofrequência.

Outra especificação era a de melhoria de desempenho do processo. Na planta de simulação isto ficou bastante evidente, pois no funcionamento antigo quando um operador desejava simular alguma altura específica da coluna de água eram necessários dois funcionários, um próximo à coluna de água observando a variação do nível e outro próximo ao acionamento do motor controlando manualmente este aumento. Com a implementação desta solução, o operador que controla o acionamento do motor consegue obter a informação de nível na tela da IHM, alterando-a conforme desejado no modo manual.

Esta situação, embora muito simples, mostra um dos conceitos básicos das vantagens de um projeto de automação que este projeto alcançou, o de facilitar a operação e atuação sobre processos, neste caso auxiliando o operador.

Quanto à especificação de controle automático, embora o controle utilizado nesta solução não seja muito complexo, este atingiu exatamente as expectativas esperadas na introdução, principalmente na procura da “solução mais adequada”. É evidente que no controle de nível de uma Elevatória de Esgoto não é necessário um algoritmo complexo de controle de processos, e o desempenho deste seria muito semelhante ao controlador implementado neste projeto.

Um resultado muito interessante atingido nesta área foi a facilidade que se observou em explicar o conceito do controle automático implementado para pessoas que não tinham conhecimento algum de leis de controle e que iriam utilizar o sistema.

Quanto a facilidade de instalação física da solução, observou-se que a divisão da solução em kits gerou uma maior facilidade de instalação principalmente na parte do roteador, que poderia ser adicionado na aplicação facilmente e sem modificar nenhum aspecto físico e de programação dos outros componentes.

Já a instalação física dos outros kits se mostrou um pouco mais dificultosa quando não realizada por mim. Pedi para alguns colegas instalarem por si mesmos os kits na planta de simulação e muitos tiveram dificuldade em entender quais fios se interconectavam, e o que descobri com isto é que as vezes o que parece ser fácil para o projetista, como neste caso a instalação elétrica, pode ser muito difícil para alguém não familiarizado com a situação.

Quanto à facilidade de alteração no projeto entre uma aplicação em EEE e aeradores, a programação da IHM se mostrou eficaz, sendo necessário apenas alterar entre os dois programas previamente compilados. Neste quesito o vencedor foi o Xbee, em que não era necessária nenhuma alteração de programação entre as duas aplicações, com isso, os kits Roteador e Dispositivo Final não necessitam de nenhuma alteração física e de programação para as duas aplicações. O Arduino por sua vez, era o que mais necessitava alterações na programação, e isto, embora facilmente alterado (como demonstrado neste documento), gerou certa insatisfação tanto minha quanto da empresa.

6.3: Ganhos obtidos.

Entre os ganhos obtidos com a realização deste trabalho podem-se citar:

- Inserção da empresa em tecnologias modernas antes não aplicadas.
- Automação de processos anteriormente totalmente manuais.
- Transformação sistemas interligados por cabeamento por sistemas integrados por redes sem fio.
- Possibilidade de obtenção de informação de diversos equipamentos remotos, todos concatenados em uma única tela.
- Otimização do tempo de trabalho de operadores.
- Redução do tempo para se realizar tarefas.
- Inserção de controle automático para auxiliar os operadores.

Observando a lista acima, é possível perceber que o projeto traz fatores impactantes positivos para a empresa e tem potencial para se tornar uma das soluções em automação que irão impulsionar a CASAN para a modernização dos processos, uma das metas da mesma.

6.4: Impactos do projeto.

Embora tenha sido implementado apenas na planta de simulação, o projeto concluído neste trabalho é totalmente funcional e apresentou uma série de avanços na área de automação dentro da empresa, sendo que esta está apenas começando a ser explorada. Apresentar uma solução baseada em redes de comunicação sem fio gerou muito interesse por parte dos funcionários e inclusive novas ideias baseadas nesta tecnologia foram planejadas no fim do mesmo.

Uma mensagem que o projeto e o projetista desejavam passar era sobre a automação de maneira mais generalista através de uma aplicação específica. A mensagem era a de que a automação não esta presente no mundo atual para desocupar cargos e trazer desemprego, e sim para alavancar o crescimento da empresa e aumentar o potencial dos funcionários, melhorando tanto suas condições de trabalho quando a eficiência que os mesmos pudessem atingir.

Este projeto tentou plantar uma semente neste sentido, sendo um dos projetos pioneiros de solução em automação criados dentro da própria empresa, e espera ter abrido os olhos não só para os potenciais projetos na área, mas também para a criação de vagas de emprego na área, atualmente sem nenhum profissional em toda a CASAN.

6.5: Considerações finais do capítulo.

Este capítulo apresentou a implementação na planta de simulação do projeto realizado neste trabalho. A ligação física foi brevemente apresentada e os ganhos reais naquela aplicação foram apresentados.

No final do capítulo foram apresentados ainda os ganhos de maneira mais geral do trabalho, com as vantagens que o mesmo poderia trazer em

implementações na área. Finalmente os impactos do projeto foram analisados, apresentando um lado mais humano pro trás da automação.

Com isso se encerra o projeto e o documento que o formaliza, restando apenas as conclusões gerais, presentes no próximo capítulo.

Capítulo 7: Conclusões e Perspectivas.

Neste trabalho foi apresentada uma solução em automação para o tratamento de efluentes. Foi citado como, com a utilização de uma rede Zigbee, é possível ter a informação e controle de diversos sistemas remotos em um único centro integrado de monitoramento.

Foram apresentadas extensivamente as tecnologias de interface, controle, envio de dados, medição e atuação utilizados para se atingir os objetivos provenientes das especificações do problema.

A empresa em que se tomou vida o sistema foi descrita, mostrando e discutindo todos seus processos operacionais com o intuito de enquadrar o problema atacado no escopo da instituição.

Foi então mostrado como a solução foi implementada em uma planta de simulação e os resultados desta atividade apresentados e discutidos.

O sentimento do projetista é de dever cumprido, sendo que o sistema de automação apresentado atinge as especificações para as quais o mesmo foi projetado e pode auxiliar em muito as atividades que a empresa realiza. A implementação em uma planta real não foi possível por alguns motivos que fogem do poder do projetista, porém isso não invalida todo o esforço e os resultados apresentados neste documento.

Pensando nas perspectivas sobre o futuro do projeto, algumas incertezas surgem. Não há dúvidas que a automação no início gera um impacto organizacional muito grande, e durante a realização deste trabalho sempre tentei minimizar este impacto, convencendo as pessoas ao redor da importância da modernização e chamando-as para compartilhar suas ideias, apresentando um lado interessante da automação que auxilia a todos.

Porém, é fácil observar ainda na empresa muita restrição nesta área, principalmente por um pensamento equivocado onde “máquinas geram demissões de funcionários” e é realmente muito difícil mudar este pensamento.

Observando o lado humano, eu entendo a preocupação de alguns, pois na sociedade capitalista o que manda é o dinheiro, e um dos maiores medos da classe trabalhadora é perder seu emprego. O que é preciso fazer é um trabalho de conscientização dentro da empresa sobre a verdade por trás da automação, um trabalho que eu iniciei durante este projeto, mas, sendo o único Engenheiro de Controle e Automação dentro da empresa, não foi possível atingir todos os objetivos nesta área.

Como pensamento final, deixo aqui minha certeza do bom caráter de todos os funcionários da CASAN, e tenho certeza de que quando a empresa estiver pronta para as mudanças, os projetos de automação irão florescer no meio, se não com este aqui apresentado, mas com outros baseados ou não nele.

Bibliografia:

[1] M. Margolis, “Arduino Cookbook”, segunda edição, O’Reilly Media, 2011.

[2] S. Farahani, “Zigbee wireless networks and tranceivers”, Newnes, 2008.

[3] Norma Técnica 02 – CELESC disponível em <www.celesc.com.br>, acessado em Maio de 2013.

[4] Missão da CASAN, disponível em <www.casan.com.br>, acessado em Junho de 2013.

[5] Diário Catarinense edição de 12 de Dezembro de 2012.

[6] Ministério Público de Santa Catarina, disponível em <www.mpsc.mp.br>, acessado em Junho de 2013.

[7] Norma 12208 da ABNT, disponivel em <www.abntcatalogo.com.br>, acessado em Junho de 2013.

[8] Portaria 518 do Ministério da Saúde, disponível em <www.saude.gov.br>, acessado em Maio de 2013.

[9] J. Nielsen “Designing web usability: the practice of simplicity.” USA: New Riders Publishing, 2000.

[10] P. E. Ceruzzi, “A History of Modern Computing”, MIT, 2003.

[11] J. C. Maxwell “On Governors”, Clark Editing, original 1868, relançado em 1976.

[12] G. Ballard, “PID versus On-off Control, why use a valve positioner?”, disponível em < http://ir.library.oregonstate.edu/xmlui/bitstream/handle/1957/5231/PID_Vs_On-Off_ocr.pdf?sequence=1>, acessado em Maio de 2013.

[13] “How many Arduinos were sold?” disponível em <www.quora.com>, acessado em Maio de 2013.

[14] A. Das, “Line Communication System”, Harvard, 2006.

[15] B. Drury, “Control Techniques Drives and Control Handbook”, Segunda edição, IET, 2009.

[16] S. P. Parker, “Engine”, Mcgraw-Hill, 1994.

[17] G. Gautschi, “Piezoeletric Sensorics”, Springer, 2002.

[18] R. Faludi, “Building Wireless Sensor Networks”, O’Reilly, 2011.

[19] “”Zigbee Alliance”, ALLIANCE, disponível em <www.zigbee.org>, acessado em Maio de 2011.

[20] Xbee/Xbee-PRO ZB RF Modules, manual, Minnetonka Digi International, 2011.

[21] Wireless LAN MAC and PHY Specification, disponível em <http://standards.ieee.org>, acessado em Julho de 2013.

[22] H.M. Oliveira “Engenharia de Telecomunicações”, Editora UFPE, 2008.

[23] T.S. Rappaport, “Comunicação sem fio”, Editora 408, 2008.

[24] A.S. Tannenbaum, “Redes de Computadores”, Editora Elsevier, 2003.

[25] M. Stemmer, “Redes locais industriais: A integração da produção através das redes de comunicação”, Editora UFSC, 2010.

[26] W. Boyes, “Instrumentation Reference Book”, quarta edição, Butterworth-Heinemann,2009.