Integrando CLP's industriais com aplicações desenvolvidas na linguagem Java por meio do protocolo MODBUS.[1]

Robson Anversa[2] José Antônio O. de Figueiredo [3]

Junho 2018

Resumo

O presente trabalho apresenta um estudo sobre a integração de CLP's industriais com aplicações desenvolvidas em linguagem Java, por meio do protocolo *MODBUS* e com a biblioteca *EasyModbus*. O trabalho tem como principal contribuição o domínio desta técnica de integração, bem como o desenvolvimento de um estudo de caso prático, aplicado a um ambiente industrial onde a comunicação é feita com sucesso. O estudo de caso proposto implementa mecanismo para quantificação de perdas de frangos, em uma linha de abate de um frigorífico da cidade. O domínio desta tecnologia de comunicação possibilita o desenvolvimento de outras soluçãoes.

Palavras-chaves: CLP. Protocolo Modbus. Aplicações Java. EasyModbus.

Introdução

Integrar um Controlador Lógico Programável (CLP) utilizado na industria com uma aplicações desenvolvidas em Java, abre muitas possibilidades de trabalho e inovação, dentre elas a obtenção de dados destas máquinas e equipamentos industriais. Dados estes que podem ser armazenadas em bancos de dados e posteriormente processados com o objetivo de extração de informações úteis. Além disso, essa integração pode também oportunizar o controle remoto desses equipamentos.

O objetivo deste trabalho é a integração do CLP Industrial com aplicações desenvolvidas em Java. Essa integração pode ser feita por diversos meios, entre eles a comunicação serial e *socket*, entre outros. Neste trabalho, esta integração é feita por meio do protocolo *Modbus*, que

[1] Artigo apresentado ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-rio-grandense, Câmpus Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

[2] Graduando em Tecnologia em Sistemas para Internet do Instituto Federal de Educação, Ciência e Tecnologia Sul-rio-grandense, Câmpus Passo Fundo (IFSul – Passo Fundo/RS). e-mail: anversa.robson@gmail.com

[3] Orientador, professor do IFSUL. e-mail: jose.figueiredo@passofundo.ifsul.edu.br

roda sobre TCP/IP e que utiliza um mapa de endereços onde os equipamentos manipulam e compartilham as informações armazenada nesses endereços, diferente da integração por *socket* que demanda tempo de processamento e espera, além de que o modelo de CLP utilizado transmiti mensagens em binário que devem ser tratadas pela aplicação.

Como estudo de caso, foi desenvolvido um sistema para quantificar as perdas em um frigorífico de aves como estudo de caso. Essas quantificações permitem saber em qual ponto da linha de produção estão ocorrendo perdas, o que permite à empresa adotar medidas corretivas a fim de minimizá-las. Além disso, permite ainda verificar a eficiência das suas ações comparando os dados por meio de tabelas ou gráficos.

O restante do trabalho se divide conforme segue: na seção 1, é descrito o desenvolvimento do trabalho, começando pela apresentação dos sensores, CLP com suas características e linguagens de programação, protocolo *Modbus TCP/IP* e a biblioteca de integração *EasyModbus*; Na seção 2, é detalhado o estudo de caso, implementado por módulos com as tecnologias estudadas; Na seção 3 encontram-se os resultados obtidos e por fim, na seção 3 as considerações finais.

1 Desenvolvimento

Nesta seção são apresentadas as tecnologias, o protocolo e o método utilizado para a coleta dos dados dos CLP's, bem como a inserção no banco de dados e retorno em listagens para o usuário através do browser. A modelagem do sistema é descrita e na sequência é realizado uma análise dos resultados obtidos, baseado nesses resultados são traçados planos de melhorias e implantação futuras.

1.1 Sensores

O controle de processos industriais, instrumentação e mesmo monitoramento de processos distantes exige o uso de sensores especiais que devem converter as mais diversas grandezas físicas em sinais elétricos.

Rosário (2005) define sensor como sendo um transdutor que altera a sua característica física interna devido a um fenômeno físico externo, seja ele presença ou não de luz, som, gás, campo elétrico, campo magnético etc. Franchi e Camargo (2008) classificam os sensores quanto a sua saída em analógicos e digitais, onde sensores digitais fornecem um sinal lógico de saída ligado/desligado. Quanto aos sensores analógicos, esses oferecem um sinal analógico que pode ser tensão, corrente, resistência, entre outros. Os sensores mais utilizados são os de proximidade e podem ser mecânicos, ópticos (fotoelétricos), ultrassônico, magnéticos, indutivos e capacitivos. A Figura 1 mostra os tipos de sensores mais comuns.

Ainda, segundo Rosário (2005), um sensor óptico (fotoelétrico) é formado por um emissor e por um receptor de luz. O princípio de funcionamento de um sensor óptico baseia-se num circuito oscilador que gera uma onda convertida em luz pelo emissor. Um circuito eletrônico identifica essa variação de luz e emite um sinal elétrico.

Existem três formas de um sensor fotoelétrico operar. Por reflexão, nesta forma a luz é refletida no objeto e o sensor é acionado, não detecta objetos transparentes ou muito escuros. Por barreira, neste modo a luz é refletida por um espelho, quando o objeto rompe a barreira de luz entre o sensor e o espelho, a saída do sensor é comutada. A outra forma é por emissor-receptor, nesta modalidade o emissor e o receptor de luz infravermelho são montados separados e, quando o raio de luz é interrompido por um objeto colocado entre os dois, cessando a propagação de luz entre eles, o sinal de saída do sensor é comutado.

Figura 1 – Seis principais tipos de sensores industriais

Fonte: Adaptado de Franchi e Camargo (2008)

1.2 CLP

Conforme Groover (2011), um controlador lógico programável (CLP) pode ser definido como um equipamento baseado em microcomputador que usa instruções baseadas em uma memória programável para implementar lógica, sequenciamento, temporização, contagem e funções aritméticas por meio de módulos de entrada/saída (E/S) digitais ou analógicos.

O CLP, ou sigla em inglês PLC (*Programmable Logic Controller*), surgiu em função das necessidades da indústria automobilística. Pois os painéis eletromecânicos para controle lógico utilizados anteriormente dificultavam as alterações e ajustes de sua lógica de funcionamento, fazendo com que as montadoras gastassem mais tempo e dinheiro a cada alteração na linha de produção.

Painéis eletromecânicos são constituídos de relés eletromecânicos, contatores, temporizadores, etc, os quais, associando seus contatos abertos e fechados, permitem implementar lógicas de funcionamento, chamada lógica de relés. A lógica de relés ainda é utilizada atualmente, porém em circuitos de baixíssima complexidade, em que não há necessidade de alterações na lógica de funcionamento.

Desta forma, em 1968, a General Motors desenvolveu o primeiro CLP, com grande versatilidade de programação e fácil utilização, o qual vem sendo aperfeiçoado constantemente, a fim de atender suas diversas aplicações atuais em automação de processos. Segundo Groover (2011), às características dos primeiros CLPs eram semelhantes às dos controles por relés que substituíram. Eram limitados a controles do tipo ligado/desligado. Em cinco anos, as melhorias no produto incluíram melhores interfaces de programação, capacidade aritméticas, manipulação de dados e comunicação com computadores.

Conforme Georgini (2000), nos anos 80, os aperfeiçoamentos atingidos fizeram do CLP um dos equipamentos mais atraentes na Automação Industrial. A possibilidade de comunicação em rede (criada em 1981) é hoje uma característica indispensável na indústria.

Atualmente os CLPs apresentam as seguintes características: módulos de I/O de alta densidade, ou seja, grande número de pontos de I/O por módulo; módulos remotos controlados por uma mesma CPU; módulos inteligentes (coprocessadores que permitem a realização de tarefas complexas como posicionamento de eixos, transmissão via rádio, leitura de código de barras, controle PID); softwares de programação em ambiente Windows; integração de

Aplicativos Windows (*Access*, *Excel*, *Visual Basic*) para comunicação com CLPs; recursos de monitoramento da execução do programa, diagnósticos e detecção de falhas; instruções avançadas que permitem operações complexas (ponto flutuante, funções trigonométricas); processamento paralelo, proporcionando confiabilidade na utilização em áreas de segurança; pequenos e micros CLPs que oferecem recursos de hardware e software dos CLPs maiores; e conexão em rede por meio de Rede Ethernet. (AZEVEDO et al., 2016)

Para Groover (2011) são cinco os componentes básicos do CLP: processador, unidade de memória, fonte de energia, módulos de I/O e dispositivo de programação. O processador é a unidade central de processamento (UCP) do CLP, ele executa várias funções lógicas e de sequenciamento por meio da manipulação das entradas do CLP. A UCP consiste de um ou mais microprocessadores semelhantes aos utilizados em PCs e em outros equipamentos de processamentos de dados. A diferença é que possuem sistema operacional executando em tempo real e são programados de modo a facilitar as transações de I/O e executar a função lógica de ladder. Além disso, os CLPs são robustos para que a UCP e outros componentes eletrônicos possam operar no ambiente eletricamente ruidoso da fábrica.

A unidade de memória está conectada à UCP e contêm os programas de lógica, sequenciamento e operações de I/O. Também mantém arquivos de dados associados a esses programas, inclusive bits de estado I/O, constantes de contadores e temporizadores, e valores de parâmetros e variáveis. Para Georgini (2000), o sistema de memória da UCP é composto pela memória do sistema de operação e memória de aplicação: a primeira é constituída pelo programa de execução (desenvolvido pelo fabricante) o qual determina como o sistema deve operar, e pelo rascunho do sistema, uma área de memória reservada para o armazenamento temporário; a segunda também chamada de memória do usuário, pois seu conteúdo é informado pelo usuário e nela estão os programas de lógica, sequenciamento e operações de I/O.

Os CLPs são programados por meio de dispositivos de programação. Diferentes fabricantes de CLP oferecem diferentes dispositivos, variando de painéis de controle simples a teclados e telas especiais de programação de CLPs.

As instruções de controle são repassadas ao CLP por meio de uma linguagem de programação que o usuário informa as instruções de controle ao CLP. Deste modo existe uma norma internacional definida pela *International Electromechanical Comission* (IEC), a IEC 611131-3, que aborda as linguagens de programação para CLP e define também a estrutura de um projeto, os tipos de dados e a organização interna de um programa.

A norma da IEC define cinco linguagens de programação para CLP, as quais são: *ladder*, lista de instruções, texto estruturado, diagrama de blocos de funções e diagrama funcional sequencial, as quais encontram-se descritas a seguir.

1.2.1 Ladder

Conforme Georgini (2000), a primeira linguagem de programação criada na programação de CLPs foi a Linguagem de Ladder. O fato de ser uma linguagem gráfica, baseada em símbolos semelhantes aos encontrados em esquemas elétricos, foi determinante para aceitação do CLP por técnicos e engenheiros acostumados com o sistema de controle a relés.

O nome Ladder (SILVEIRA, 2017) deve-se à representação da linguagem parecer com uma escada (ladder). A programação é feita pela inserção de componentes apropriados nos degraus do diagrama, os componentes são basicamente de dois tipos: contatos e bobinas.

1.2.2 Lista de instruções

É uma linguagem de baixo nível parecida com *Assembly*, onde é permitida apenas uma operação por linha. Também oferece um modo de inserir o diagrama de lógica ladder na memória do CLP, a Tabela 1 mostra um exemplo lista de instruções.

Tabela 1 – Listra de instruções

Comentário	**Comando**
Armazena a entrada X	STR X
Entrada X1 em série com X	AND X1
Fornece Y como saída	OUT Y
Armazena o inverso da saída Y	STR NOT Y

Fonte: Adaptado de Groover (2011)

1.2.3 Texto extruturado

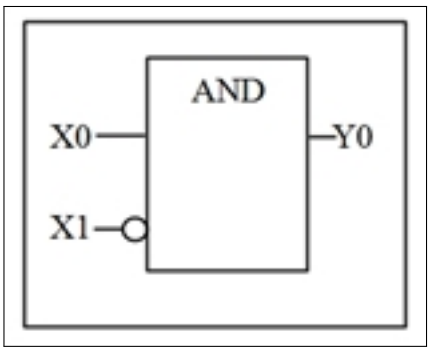
É uma linguagem de alto nível, estruturada em blocos semelhante a Pascal. Tal qual Groover (2011), a principal vantagem desta linguagem em relação às outras duas citadas anteriormente, é a capacidade de executar processos de dados e cálculos sobre valores que não sejam binários.

Nesta linguagem existem também tipos de dados específicos para gerenciamento de horas, datas e duração de tempo. Além disso, suporta interação de loops como o REPEAT UNTIL (repita até que), execução condicional IF-THEN-ELSE (se-então-senão) e funções como: SQR (raiz quadrada) e SIN (seno).

1.2.4 Diagrama de blocos de funções

É uma linguagem gráfica que permite aos elementos do programa (blocos) serem conectados entre si de forma semelhante a um circuito elétrico. Para Georgini (2000) essa linguagem é apropriada para aplicações que envolvam fluxo de informações, ou dados entre os componentes de controle. A Figura 2 apresenta um exemplo simples de programação em linguagem de diagrama de blocos de função.

Figura 2 – Diagrama de blocos para programação de CLP's.



Fonte: Adaptado de Georgini (2000)

1.2.5 Diagrama Funcional Sequencial

De acordo com Georgini (2000), o *Sequential Function Chart*(SFC), trata-se de uma linguagem gráfica que é utilizada para estruturar a organização interna de um programa, além de auxiliar a decomposição do problema de controle em partes menores. Cada elemento do SFC pode ser programado em qualquer uma das linguagens definidas pela própria norma da IEC.

1.3 Modbus TCP/IP

O protocolo *Modbus* é baseado no paradigma de comunicação mestre-escravo, onde o mestre é responsável por coordenar diversos escravos. O *Modbus* é utilizado no nível da camada de aplicação (KELLER, 2017) e foi inicialmente desenvolvido para ser utilizado na comunicação de CLPs. Foi criado pela Modicon Industrial Automation Systems (atual *Schneider Electric*[1]) nos anos 70, sendo posteriormente modificada para licença livre. Atualmente, o protocolo é mantido pela *Modbus-IDA*, que é formado por um grupo de usuários e fornecedores independentes.

O TCP/IP é uma suíte de protocolos para comunicação de dados na Internet. Esse conjunto de protocolos,(AUTOMATION, 2015a) e (AUTOMATION, 2015b) proporciona um mecanismo confiável de transporte de dados entre máquinas. O uso de *Ethernet* TCP/IP em fábricas permite uma verdadeira integração com a Intranet corporativa. Para mover o *Modbus* para o século XXI, uma especificação aberta do *Modbus* TCP/IP foi desenvolvida em 1999. A especificação do protocolo e o guia de implementação estão disponíveis para download no site da mantenedora do *Modbus*[2].

> Combinando uma rede física versátil, escalável e onipresente (Ethernet) com um padrão de rede universal (TCP/IP) e uma representação de dados neutra do fornecedor, o *Modbus* fornece uma rede verdadeiramente aberta e acessível para troca de dados de processo. É simples de implementar para qualquer dispositivo que suporte sockets TCP/IP. ” (MODBUS, 2018)

No protocolo *Modbus* apenas o mestre pode fazer requisições. Geralmente o mestre é um sistema supervisório e os escravos são CLPs, sendo que um dispositivo pode assumir ambos os papéis, mas não simultaneamente. O modelo de dados *Modbus* é baseado em um conjunto de tabelas, sendo que cada uma delas têm as suas características próprias, onde formam um mapa de memória que podem ser lidos e/ou gravados. A Tabela 2 representa as quatro principais tabelas do mapeamento *Modbus*.

Tabela 2 – Mapeamento Modbus

Tabelas	**Tipo**	**Permissão**	**Faixa Modbus (endereços)**
Discrete Inputs	1 bit	Somente Leitura	10001-19999
Coils	1 bit	Leitura/Escrita	1-1999
Inputs Registers	16 bits	Somente Leitura	30001-39999
Holding Registers	16 bit	Leitura/Escrita	40001-49999

Fonte: Adaptado de MODBUS (2018)

[1] https://www.schneider-electric.us/en/
[2] www.modbus.org/specs

1.4 API Java EasyModbus

A plataforma Java dispõe de diversas bibliotecas, que já estão bem desenvolvidas e testadas. Estas bibliotecas têm por função facilitar a utilização de algum recurso ou tecnologia.

A biblioteca *EasyModbus* (Rossmann-engineering (2017)), utilizada neste trabalho, abstrai a complexidade do protocolo *Modbus TCP/IP*, facilitando seu uso para o desenvolvedor, de forma que seja possível conectar-se ao CLP, apenas informando o endereço IP e a porta TCP deste.

A biblioteca *EasyModbus* (ROSSMANN-ENGINEERING, 2017), (FREITAS, 2014) dispõe de métodos que possibilitam realizar a leitura ou a escrita de vários endereços do mapa *Modbus* configurado no CLP, desde que pertençam a mesma tabela. Esta biblioteca é disponibilizada também em versões para Python e C#.

A mesma é desenvolvida pela Rossmann Engineering Systems[3], uma companhia Indiana que produz software de código aberto para tecnologia de automação.

2 Estudo de caso: sistema para quantificação de perdas em frigorífico

Para o desenvolvimento desta integração entre CLP e aplicações Java com banco de dados, optou-se pelo desenvolvimento de um estudo de caso que tem como objetivo quantificar as perdas em uma linha de abate de frangos em um frigorífico da região.

Frigoríficos são ambientes hostis com muita umidade, por utilizarem água em seus processos automatizados e constantemente passar por processos de limpeza. Além disso existe bastante interferência eletromagnética gerada pela grande quantidade de motores indutivos, o que torna necessário que os equipamentos sejam robustos e suportem essas adversidades, diante deste cenário faz-se necessário o uso de CLP's ao invés de alguma placa embarcada que teria a mesma finalidade, isto porque o tempo de sua vida útil seria muito inferior pelo fato de não serem projetadas para resistirem a tais ambientes.

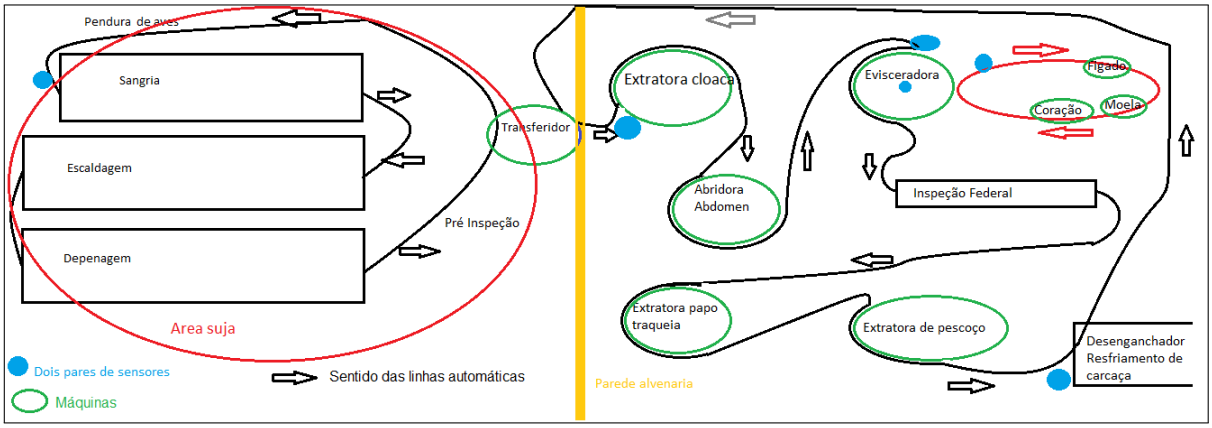
O abate de aves é automatizado, as mesmas são penduradas pelas pernas em uma linha de abate chamada *noria*, passam por uma cuba de choque para desmaiar e não sofrer durante o processo, em seguida são sangradas e depenadas, após passam pelo processo de evisceração e inspeção, acabando no *chiller* que é um tanque cheio água gelada a fim de resfriar as carcaças antes de seguirem o processamento. A evisceradora, máquina que extrai as vísceras possui 24 módulos e é em formato de carrossel.

O sistema idealizado para quantificação de perdas, funciona da seguinte forma: três pontos de contagens das aves na linha para identificar a quantidade perdida durante o processo de depenagem e a quantidade perdida durante o processo de evisceração. Dessa forma serão instalados sensores na entrada e saída da evisceradora para identificar a qual módulo a ave será direcionada e se a extração das vísceras foi realizada com sucesso tendo assim a porcentagem de eficiência de cada módulo. A Figura 3 mostra a localização destes sensores na planta de abate.

As contagens obtidas serão armazenadas em um banco de dados para controle e consultas futuras. Isto permitirá à empresa, por exemplo, comparar os dados entre turnos ou após implantação de medidas para redução de perdas. A contagem da eficiência dos módulos da evisceradora irá auxiliar também na manutenção e na regulagem dos mesmos, sendo que torna-se necessário

[3] http://rossmann-engineering.easymodbustcp.net/de/

Figura 3 – Posicionamento dos sensores em uma planta de abate, para o sistema de quantificação de perdas.



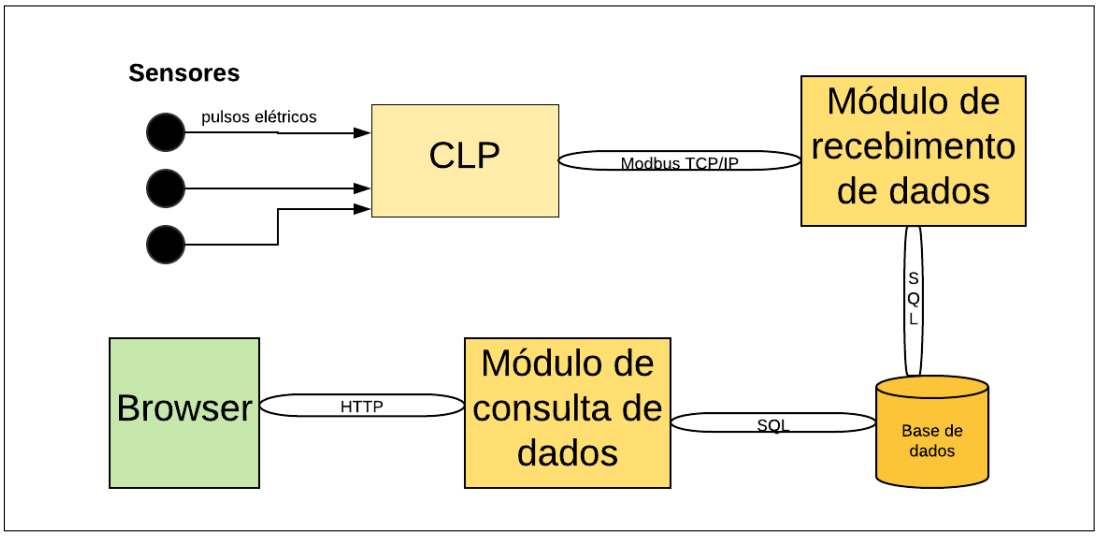
Fonte: Do autor.

intervir na regulagem apenas dos que não estiverem extraindo uma porcentagem satisfatória de vísceras.

2.1 Arquitetura do sistema proposto

O sistema proposto está dividido em módulos funcionais interdependentes, o que facilita a implementação. Esta separação em módulos integrados, conforme demonstrado na Figura 4, facilita o desenvolvimento e a manutenção de cada módulo de forma independente. Os módulos representados na figura serão detalhados a seguir.

Figura 4 – Módulos funcionais interdependentes do sistema de integração do CLP com Aplicação Java.



Fonte: Do autor.

2.1.1 CLP e Sensores

Os sensores utilizados no projeto são do tipo fotoelétricos emissor-receptor, escolhidos por possuírem maior precisão em aplicações que exigem maior frequência de *clock*. Os sensores detectam a interrupção entre o emissor e receptor, acionando um pulso elétrico em suas saídas que estão diretamente conectadas às entradas do CLP.

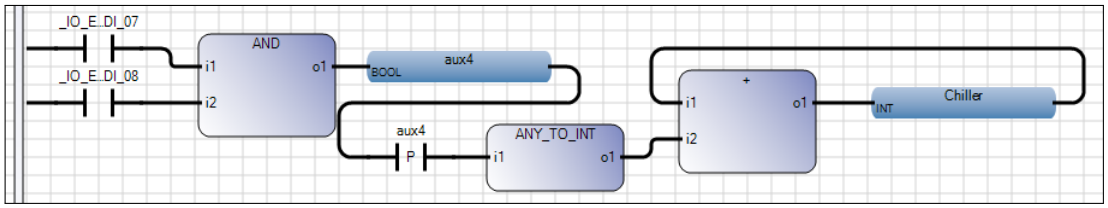
Para realizar a contagem de objetos que se encontram em movimento foi necessário desenvolver um suporte para a instalação de dois sensores: um deles detecta um ponto onde a linha está tensionada e a oscilação é a mínima possível; enquanto isto, o outro sensor detecta o frango transportado em um ponto que possui mais oscilação, sendo que o primeiro fica acionado por um curto período de tempo e detecta todos os ganchos até mesmo os vazios, e o segundo detecta as aves porém em movimento, correndo risco de contar várias vezes a mesma ave, por isso os dois tem que estarem acionados ao mesmo tempo para incrementar mais um na contagem.

O CLP utilizado no sistema, da marca Allen-Bradley[4], possui uma porta Ethernet/IP, pela qual é realizado a programação do mesmo e também a comunicação com a aplicação Java desenvolvida. Para programá-lo optou-se pela linguagem de bloco de funções, uma vez que a curva de aprendizado é menor, facilitando a implementação de contadores. As entradas digitais do CLP estão configuradas como variáveis booleanas, desta forma cada vez que as duas entradas interligadas aos sensores de um mesmo ponto de contagem estiverem em nível lógico alto um bloco de função contador é acionado e incrementa mais um em uma variável de contagem

Estas variáveis estão apontadas no mapeamento *Modbus*, desta forma podem ser acessadas para leitura e escrita conforme a tabela que foi apontada (descrição do mapeamento *Modbus* na seção 2). A Figura 5 demonstra um contador desenvolvido com blocos de funções. Por padrão o que estiver no lado esquerdo dos blocos são entradas e o que estiver no lado direito é saída, por isso que o lado direito da variável está conectado novamente ao bloco de soma.

Da esquerda para a direita estão as entradas digitais sete e oito onde os sensores são ligados, as quais estão conectadas a um bloco de funções AND. A saída da função AND é armazenada em uma variável auxiliar, para a seguir ser utilizada em uma função de pulso representada pela letra P. Esta função P, impede que que a operação de soma fique incrementando na velocidade do *clock* indefinidamente, pois a cada *clock* a função soma executa, como a função P fica em nível lógico alto apenas durante o período de um *clock*, para sua saída ser *true* novamente a entrada deve retornar a *false* e *true* mais uma vez, desta forma a função de soma continua executando a cada *clock*, mas soma mais um apenas uma vez a cada ocorrência de *true* na saída da função AND, durante os outros *clocks* está somando zero. O resultado da função P é convertido para inteiro e é colocado na função soma.

Figura 5 – Diagrama de blocos de funções para implementar um contador no CLP.



Fonte: Do autor.

2.1.2 Módulo de recebimento de dados

A aplicação foi desenvolvida com o intuito de consultar as informações registrada nos contadores programados no CLP e armazenar em um banco de dados local. Para o armazenamento de dados utilizou-se SGBD PostgreSQL, no qual foi apenas criado uma nova base de dados e definido um usuário e uma senha. A utilização da biblioteca *Hibernate* facilita as operações com a base de dados utilizando apenas notações nas classes Java do sistema, onde uma classe

[4] https://ab.rockwellautomation.com/

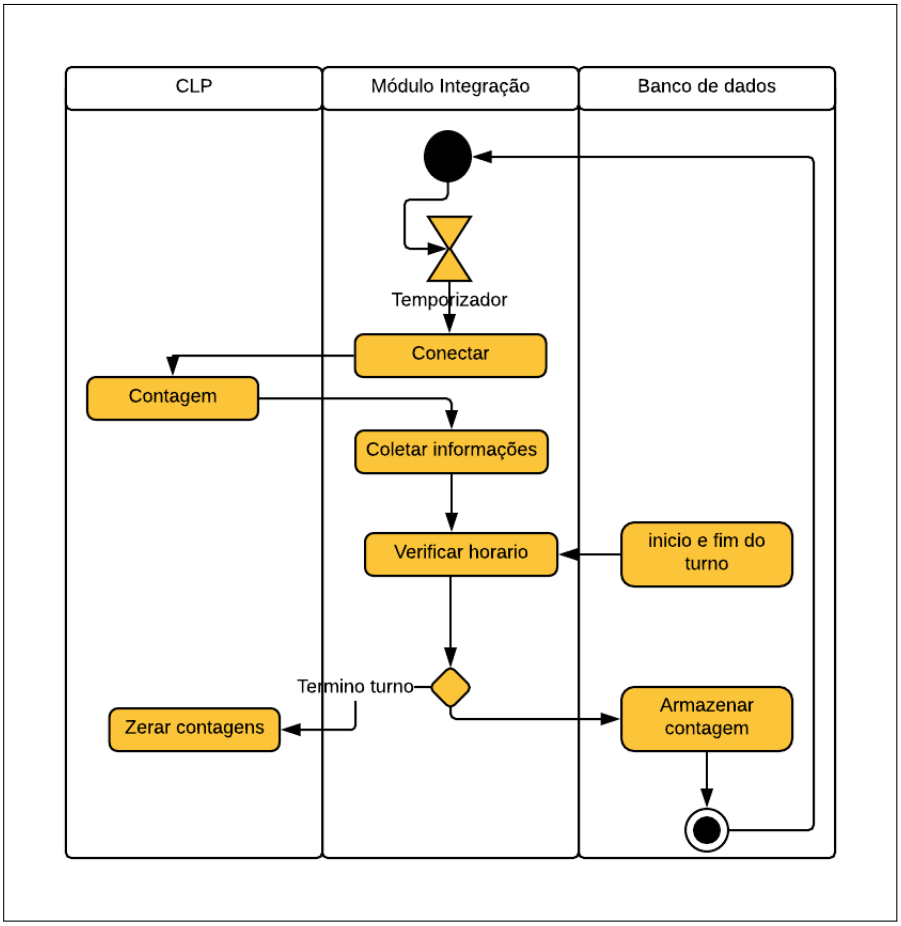
torna-se uma tabela e cada atributo da classe uma coluna, desta forma a *Hibernate* armazena os objetos das classes na base de dados e transforma as consultas em objetos para ser utilizados pelo sistema.

Cada tipo de contagem tornou-se uma classe dentro da aplicação, além disso existe uma classe principal que estabelece a conexão, realiza a consulta das variáveis a cada dez minutos e fecha a conexão. Para este controle, utilizou-se uma função de controle, que cria uma nova *thread* a cada determinado intervalo de tempo, executando as coletas de dados e as inserções no banco de dados. A aplicação conta com uma classe responsável pela consulta de informações, na base de dados, sobre o início e fim de cada turno; isto com o objetivo de definir dentro de qual turno cada consulta encontra-se, ou se está no período de intervalo dos mesmos.

Caso os minutos da hora forem menores que dez faz a inserção em uma tabela que possui as coletas de hora em hora. A primeira vez que estiver no horário de intervalo entre turnos, armazena em uma tabela que estão as contagens do dia de produção de cada turno e zera as variáveis no CLP, fazendo a próxima leitura apenas quando o horário estiver dentro do próximo turno de produção. Faz a verificação se é o último dia do mês, caso for, realiza a soma das contagens diárias e armazena em uma tabela de leituras mensais.

A Figura 6 demonstra o diagrama de atividades do módulo de recebimento de dados que limita-se a conectar-se, coletar as informações, armazenar no banco de dados e zerar os contadores.

Figura 6 – Diagrama de atividades representando o módulo de recebimento de dados



Fonte: Do autor.

Para conectar-se ao CLP, ler e escrever nas suas variáveis, a aplicação utiliza a biblioteca *EasyModbus*, que abstrai toda a complexidade do protocolo *Modbus*, pela mesma também aciona-se saídas do CLP, que acionam lâmpadas para identificar visualmente que está conectado, ou zerando as variáveis.

Os principais métodos da biblioteca *EasyModbus* utilizados neste trabalho são:

- Connect(): Método responsável por conectar ao CLP. Os parâmetros são IP e porta;
- ReadingHoldingRegister(): Método responsável por fazer a leitura dos registros que o CLP está mantendo. Como parâmetro deve ser informado a posição da informação no mapeamento de endereços;
- ConvertRegistrerToDouble(): Método que converte os valores lidos nos registradores do CLP;
- Disconect(): Método responsável por desconectar a comunicação estabelecida entre o CLP e a aplicação Java.

O banco de dados é manipulado apenas pelo módulo de recebimento de dados, sendo que uma das aplicações faz a inserção dos dados das contagens e a outra realiza as consultas das informações para disponibilizar ao usuário. A base de dados utilizada é PostgreSQL, sendo que foi apenas criado uma nova base de dados, as tabelas e suas colunas foram definidas todas pela aplicação Java intermédio da biblioteca *Hibernate JPA* passando o nome do banco, usuário, senha e a porta de conexão em um arquivo de configuração.

2.1.3 Módulo de consulta de dados

Desenvolvida com o Framework Primefaces[5], que é uma biblioteca de componentes de interface gráfica para as aplicações Web baseadas em JSF e é muito flexível e personalizável, com uma grande opção de componentes para os mais diversos fins, por exemplo as tabelas e os gráficos desenvolvidos neste projeto.

Como servidor foi utilizado o Glassfish[6], que é um servidor de aplicação Open Source. Ele implementa as especificações Java EE, além de suportar Enterprise JavaBeans, JPA, JavaServer Faces, entre outras.

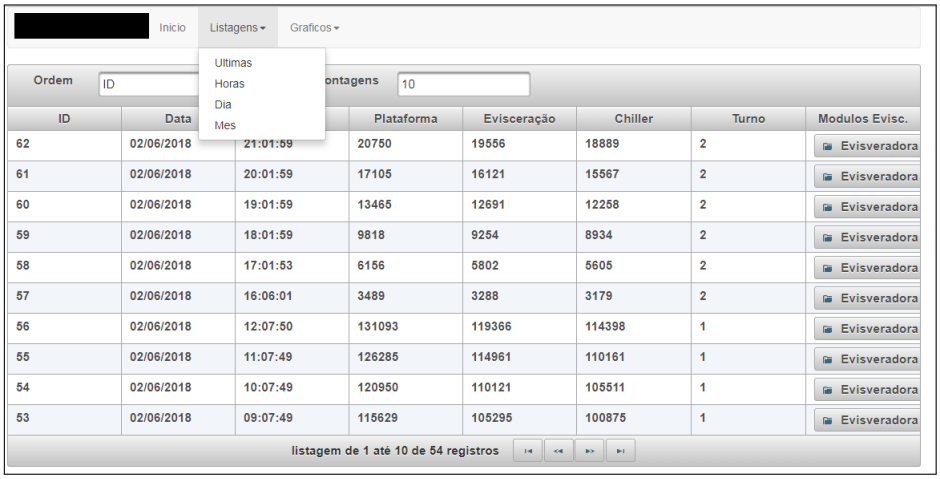
A aplicação de Consulta realiza conexão ao banco de dados e o usuário pode visualizar no browser as listagens ou gráficos como preferir, que podem ser a cada dez minutos, de hora em hora, por dia de produção ou por mês. A Figura 7 demonstra a listagem por hora.

Na Figura 8 é apresentado uma tela de gráficos, das coletas de dez em dez minutos, os gráficos possuem recurso de *zoom* que altera a escala conforme a faixa selecionada, nesta imagem as barras indicam as coletas e as linhas indicam as perdas. Com o recurso de *zoom* é possível visualizar melhor as linhas, com um duplo clique o gráfico retorna ao *zoom* normal.

Na Figura 9 é demonstrado a mesma tela da Figura 8, porem com o recurso de *zoom* que permite visualizar um gráfico de linhas mostrando as perdas identificadas a cada dez minutos nos pontos de contagem.

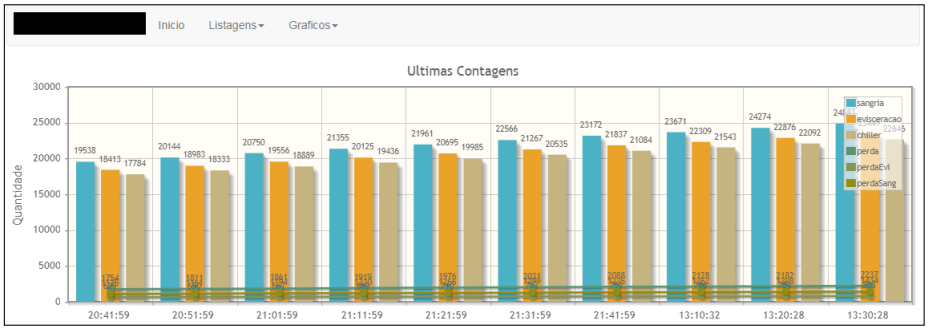
[5] https://www.primefaces.org/
[6] http://www.oracle.com/technetwork/java/javaee/downloads/index.html

Figura 7 – Tela com listagens das coletas a cada hora apresentado no Módulo de Consulta de Dados.



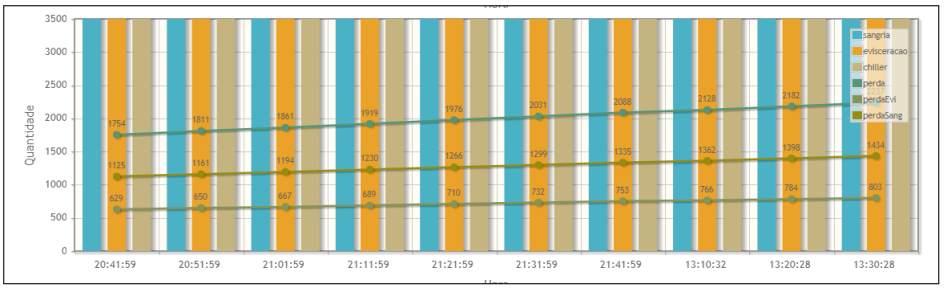
Fonte: Do autor.

Figura 8 – Tela mostrando gráfico das coletas de dados a cada 10 minutos, visualizado no Módulo de Consulta de Dados.



Fonte: Do autor.

Figura 9 – Tela mostrando gráfico das coletas de dados a cada 10 minutos, visualizado no Módulo de Consulta de Dados com o recurso de *zoom*.



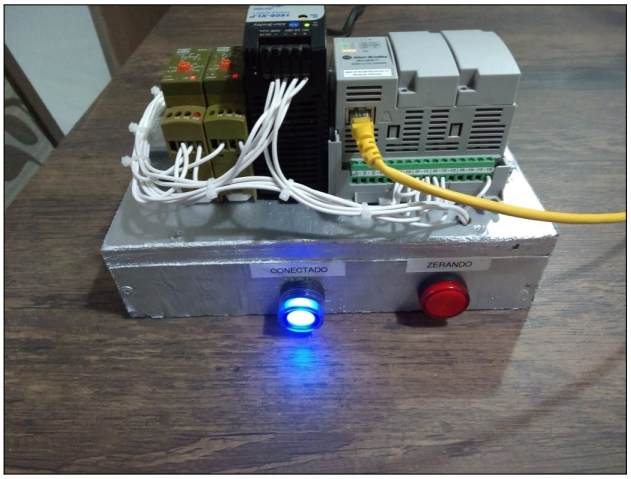
Fonte: Do autor.

2.2 Validação dos resultados

Para validar os resultados, foi desenvolvido um circuito protótipo com relés temporizadores com a finalidade de popular o banco de dados e verificar se a cada hora o sistema realizava a inserção de dados na tabela correta, do mesmo modo a averiguar se no final de cada turno armazenava os dados na tabela de contagens diárias, além de zerar as variáveis no controlador lógico programável.

A Figura 10 apresenta o protótipo desenvolvido para automatizar os testes e popular o banco de dados.

Figura 10 – Protótipo desenvolvido para popular o banco de dados e demais testes da integração de CLP com aplicação Java utilizando protocolo *Modbus*.



Fonte: Do autor.

3 Resultados obtidos

O principal resultado obtido está na implementação de uma comunicação bidirecional, sobre TCP/IP, entre um CLP industrial e a aplicações desenvolvidas em Java, utilizando o protocolo *Modbus* e a biblioteca *EasyModbus*.

Além disso, destaca-se que estudo de caso proposto teve ótima aceitação na empresa por parte dos setores responsáveis (Programação e Controle de Manutenção),(Controle de Qualidade) e também pelo pessoal responsável pela análise de rendimento. Dessa forma uma implantação do sistema no ambiente de produção está em discussão para ser realizada em breve.

Considerações finais

O trabalho desenvolvido atingiu o objetivo proposto de integrar a tecnologia industrial, neste trabalho especificamente os CLP's, com as tecnologias de desenvolvimento Java na plataforma Java EE. Este tipo de integração vem ganhando destaque na área conhecida por quarta revolução industrial, especificamente no assunto integração de máquinas e sistemas. A integração foi possível pela possibilidade de uso do protocolo *Modbus*, que é implementada com a biblioteca *EasyModbus*.

No que tange a aplicação desenvolvida, destaca-se que a opção pela arquitetura dividida em módulos facilitou a implementação do estudo de caso, pois o mesmo foi desenvolvido por etapas, que integram-se de forma transparente para o usuário do sistema.

Como trabalhos futuros, pretende-se implementar o controle de usuários com login e senha, também implantar o sistema no frigorífico de aves, instalando o CLP com os sensores e as aplicações java no servidor da empresa, para que seja acessado de qualquer computador da rede. Com o domínio da tecnologia de comunicação pode-se também trabalhar no desenvolvimento de

outras soluções que venham a surgir. Além disso, busca-se fazer uma publicação especificamente sobre o método de integração Java com *Modbus*.

Abstract

This work present a studty about integration process of industrial PLC's with Java applications, usign MODBUS protocol over EasyModbus library. The main contribution is compreension of integration technique applied in a practial study case on industrial environment. The proposed study case implements a mechanism to quantify looses on chicken slaughter line. The mastery of this communication technology allows the development of other solutions.

Key-words: PLC. Modbus Protcol, Java Applications. EasyModBus.

Referências

AUTOMATION, R. *Configuração de rede EtherNet/IP*. 2015. Online. Disponível em: <http://literature.rockwellautomation.com/idc/groups/literature/documents/um/enet-um001_-pt-p.pdf>. Acesso em: 07.04.2018.

AUTOMATION, R. *Fundamentos da Rede Ethernet/IP*. 2015. Online. Disponível em: <http://www.rockwellautomation.com/resources/downloads/rockwellautomation/bra/pdf/t02-fundamentos-da-rede-ethernetip.pdf>. Acesso em: 07.04.2018.

AZEVEDO, P. K. et al. Desenvolvimento de um sistema supervisório e lógicas de clp no ambiente de geração de energia. Florianópolis, SC., 2016.

FRANCHI, C. M.; CAMARGO, V. L. A. *Controladores lógicos programáveis: sistemas discretos*. [S.l.: s.n.], 2008.

FREITAS, C. Protocolo modbus: Fundamentos e aplicações. *Artigo técnico O Embarcados*, 2014.

GEORGINI, M. *Automação aplicada: descrição e implementação de sistemas seqüenciais com PLCs*. [S.l.]: Ed. Érica, 2000.

GROOVER, M. P. *Automação e tecnologias de controle. In: GROOVER, Mikell P. Automação Industrial e Sistemas de Manufaturas.* [S.l.]: Ed. Pearson, 2011.

KELLER, A. L. Internet das coisas aplicada à indústria: dispositivo para interoperabilidade de redes industriais. Universidade do Vale do Rio dos Sinos, 2017.

MODBUS. *Modbus Technical Resources*. 2018. Online. Disponível em: <http://www.modbus-.org/tech.php>. Acesso em: 07.05.2018.

ROSÁRIO, J. M. *Princípios de mecatrônica*. [S.l.]: Pearson Educación, 2005.

ROSSMANN-ENGINEERING. *EasyModbusTCP JAVA Implementation*. 2017. Online. Disponível em: <http://easymodbustcp.net/java-modbusclient-methods>. Acesso em: 07.05.2018.

SILVEIRA, C. B. *Como Funciona a Linguagem LADDER*. 2017. Online. Disponível em: <https://www.citisystems.com.br/linguagem-ladder/>. Acesso em: 07.05.2018.