UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

ENGENHARIA ELÉTICA

BRUNO KANASHIRO
PÂMELA DE SOUZA SCHIABER

IMPLEMENTAÇÃO DE UM SISTEMA DE COMUNICAÇÃO UTILIZANDO O PADRÃO OPC COM CONTROLE VIA DISPOSITIVO DE COMUNICAÇÃO MÓVEL

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO
2018

BRUNO KANASHIRO
PÂMELA DE SOUZA SCHIABER

IMPLEMENTAÇÃO DE UM SISTEMA DE COMUNICAÇÃO UTILIZANDO O PADRÃO OPC COM CONTROLE VIA DISPOSITIVO DE COMUNICAÇÃO MÓVEL

Trabalho de Conclusão de Curso de graduação, do curso de engenharia elétrica da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para a obtenção do título de Bacharel.

Orientador: Prof. Dr. Luiz Marcelo Chiesse da Silva

CORNÉLIO PROCÓPIO
2018

Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio
Departamento Acadêmico de Elétrica
Curso de Engenharia Elétrica

FOLHA DE APROVAÇÃO

Bruno Kanashiro

Implementação de um sistema de comunicação utilizando o padrão OPC com controle via dispositivo de comunicação móvel

Trabalho de conclusão de curso apresentado às 13:00hs do dia 11/06/2018 como requisito parcial para a obtenção do título de Engenheiro Eletricista no programa de Graduação em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná. O candidato foi arguido pela Banca Avaliadora composta pelos professores abaixo assinados. Após deliberação, a Banca Avaliadora considerou o trabalho aprovado.

_____
Prof(a). Dr(a). Luiz Marcelo Chiesse da Silva - Presidente (Orientador)

_____
Prof(a). Dr(a). Wagner Endo - (Membro)

_____
Prof(a). Me(a). Miguel Angel Chincaro Bernuy - (Membro)

A folha de aprovação assinada encontra-se na coordenação do curso.

AGRADECIMENTOS

Agradecemos à nossa família, nossos pais e irmãs por todo o apoio e compreensão.

Aos nossos amigos pela ajuda, por toda a paciência e apoio neste projeto.

Ao nosso orientador Prof. Dr. Luiz Marcelo Chiesse da Silva, pela sabedoria com que nos guiou nesta trajetória.

Ao professor Dr. Wagner Endo por ajuda e auxílio na reta final deste trabalho.

Enfim, a todos os que por algum motivo contribuíram para a realização deste projeto.

RESUMO

SCHIABER, Pâmela de Souza; KANASHIRO, Bruno. **Implementação de um sistema de comunicação utilizando o padrão OPC com controle via dispositivo de comunicação móvel**. 2018. 55 f. Trabalho de Conclusão de Curso (Graduação) – Engenharia Elétrica. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2018.

Este trabalho consiste em implementar um sistema de comunicação utilizando o padrão OPC e permitindo o controle do processo em um *smartphone*. O padrão OPC (*Open Platform Communications*) é um padrão desenvolvido para permitir a comunicação entre equipamentos de diferentes fabricantes. O controle do processo via dispositivo de comunicação móvel proporciona ao operador maior flexibilidade de trabalho, visto que o controle não precisa ser realizado no equipamento, o dispositivo só precisa estar conectado na mesma rede lógica. Para validar o sistema de comunicação, foram realizados testes em diferentes equipamentos como no arduino uno e em um sistema SCADA simulado que pode ser utilizados na indústria.

Palavras-chave: Padrão OPC. Sistema de comunicação. Controle via dispositivo de comunicação móvel.

ABSTRACT

SCHIABER, Pâmela de Souza; KANASHIRO, Bruno. **Implementation of a communication system using OPC standard via mobile communication device**. 2018 55 f. Trabalho de Conclusão de Curso (Graduação) – Engenharia Elétrica. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2018.

This work consists of implementing a communication system using OPC standard and allowing the process control on a smartphone. The OPC standard (Open Platform Communications) is a standard developed to allow the communication between equipment from different manufacturers. The process control via mobile communication device provides to the operator a bigger flexibility to work, since the control does not need be realised on the device, the device only needs to be connected on the same logic network. To validate the communication system, tests were carried out in different equipment such as Arduino UNO and a simulated SCADA system that can be used on industry.

Keywords: OPC standard. Communication system. Control via mobile communication device.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

OPC	*Open Platform Communications*
CLP	Controlador Lógico Programável
IHM	Interface Homem – Máquina
OLE	*Object Linking and Embedding*
DCOM	*Distributed Componen Object Model*
IEC	*International Electro technical Comission*
OPC UA	*Open Platform Communications Unified Architecture*
DCS	*Distribuited Control Systems*
SCADA	*Supervisory Control And Data Acquisition*
USB	*Universal Serial Bus*
PWM	*Pulse Widht Modulation*
IP	*Internet Protocol*
HTTP	*Hypertext Transfer Protocol*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
LED	*Light Emitting Diode*
ADO	*Active X Data Objects*

SUMÁRIO

1 INTRODUÇÃO

O avanço tecnológico se iniciou com a revolução industrial, por meio da mecanização da produção. Anteriormente, a produção ocorria em condições precárias nas fábricas, e a revolução industrial trouxe os conceitos iniciais futuramente se tornariam o conceito de automação industrial nas mesmas. Essa automação começou com células automatizadas nas indústrias, onde seus sistemas compartilhavam informações, o que não ocorria anteriormente (CAPELLI, 2006).

Relés e contatores começaram a ser utilizados, proporcionando o desenvolvimento de máquinas mais modernas. Na década de 60 surgiram a utilização de sistemas digitais, e os computadores comerciais começaram a ter aplicação na indústria como controladores em sistemas de grande porte. Porém, estes computadores possuíam algumas desvantagens como custo, fragilidade no funcionamento de acordo com o ambiente de utilização e dificuldade de programação. Então, na década de 70 foram desenvolvidos os primeiros controladores lógicos programáveis (CLP), que depois de melhorias tecnológicas, são utilizados até nos dias atuais (FRANCHI, 2008).

Esse controlador, ou até mesmo microcontroladores, permitem controlar processos e realizar a comunicação com outros controladores distantes que, de acordo com Balieiro (2008), possibilitam interligar o chão de fábrica com outros controladores por meio de redes de comunicação digital, denominadas de “barramento de campo” ou *Fieldbus*. Com estas redes é possível um intercâmbio de informações necessárias à coordenação do funcionamento da planta, coletar um número maior de informações de uma planta industrial para a melhoria da produção, e até possibilitar a interconectividade entre produtos de diferentes fabricantes.

Com a facilidade na comunicação de dados, foram desenvolvidos diferentes protocolos de comunicação, geralmente não compatíveis entre si. Dessa forma, só é possível acessar dados de determinado equipamento se este utiliza o protocolo adotado pelo fabricante do mesmo, dificultando ou até impossibilitando a integração entre equipamentos diferentes e fornecedores diferentes (BALIEIRO,2008).

Para solucionar este problema foram desenvolvidos protocolos de comunicação, sendo um deles o padrão OPC (*Open Platform Communications*), que estabelece regras para ser efetuada a comunicação de dispositivos de campo (como

um equipamento CLP e sistemas de aquisição de dados) com sistemas de monitoração, supervisão e gerenciamento (FONSECA, 2002).

Um dispositivo de campo que pode ser utilizado é o sistema arduino, que conforme McRoberts (2011) consiste em um *hardware* microcontrolado que possibilita o processamento de entradas e saídas entre um microcontrolador e os componentes externos conectados a ele.

Com a interconexão proporcionada pelas diversas redes de comunicação de dados, os supervisores das indústrias não precisam observar e controlar os processos de suas plantas pessoalmente. Com o uso de um dispositivo de comunicação móvel (como um *smartphone*) é possível observar e controlar os processos via *internet* ou rede local, facilitando e acelerando a tomada de decisões, i.e., por meio deste controle, mesmo à distância do processo, é possível adquirir dados e realizar mudanças com resposta em questão de segundos.

Desta forma, este trabalho irá mostrar a implementação de um sistema de comunicação utilizando o padrão OPC com o controle e supervisão efetuados via um dispositivo de comunicação móvel.

Esta implementação será feita por meio de 4 testes que podem ser utilizados como base para aplicações industriais, já que envolvem a utilização e armazenamento de dados digitais e analógicos. Os testes que serão apresentados neste trabalho são: Controle de Leds, armazenamento de dados analógicos, simulação de controle de esteiras industriais e armazenamento de dados digitais.

1.1 Objetivos

Este projeto de pesquisa possui os seguintes objetivos, subdivididos em objetivo geral e objetivos específicos.

1.1.1 Objetivo geral

Implementar subsistemas de comunicação utilizando o padrão OPC, com controle e supervisão via dispositivo de comunicação móvel de um processo com o controle implementado em uma plataforma baseada em microcontrolador.

1.1.2 Objetivos específicos

- Pesquisar os conceitos fundamentais relacionados à automação e redes de comunicação industrial.
- Determinar as especificações funcionais e características dos sistemas a serem utilizados inicialmente: padrão OPC, arduino e *software* IHM (Interface Homem-Máquina).
- Implementar o projeto de automação de modo a proporcionar a supervisão, incluindo a aquisição de dados OPC; e controle de um processo a partir de um dispositivo móvel.
- Efetuar testes para validar a efetivação do sistema para diferentes tipos de dados provenientes do processo automatizado.

1.2 Problema e justificativa

Atualmente, sistemas de automação industrial e controle de processos de diferentes fabricantes utilizam protocolos de comunicação distintos e muitas vezes proprietários, tornando uma desvantagem a sua utilização, impedindo a interoperabilidade entre os sistemas de supervisão, controle e automação de marcas diferentes (RIBEIRO et.al.,2008).

O sistema OPC é um padrão industrial projetado para a interconectividade de sistemas diferentes, provendo sua compatibilidade. O OPC pode criar e gerenciar especificações para padronizar a comunicação de dados *on-line*, resultando no acesso a estes dados por um ou mais sistemas, mesmo que as respectivas aplicações utilizem formatos de dados (arquivos) diferentes (RIBEIRO et.al.,2008). Estes dados *on-line* podem ser acessados diretamente por meio de um computador ou *smartphone*, facilitando o acesso à supervisão ou controle do processo.

Deste modo, a implementação do padrão OPC no projeto deste trabalho disponibilizando a parametrização e monitoramento de processos de coltrole em um *smartphone*, demonstra a viabilidade e vantagens ao incrementar as opções funcionais disponíveis dos mesmos, além de fornecer o conhecimento necessário a projetistas e futuros usuários para o uso desta tecnologia padrão, incentivando o conhecimento deste sistema no mercado e de seus respectivos operadores na área de automação, como uma importante ferramenta para aumentar a flexibilidade de um processo industrial (RIBEIRO et.al.,2008).

2 FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica para a elaboração deste trabalho está descrita a seguir, dividindo-se entre o surgimento da automação industrial até a utilização das redes de comunicação de dados utilizando o padrão de comunicação OPC e controle remoto do processo via dispositivo de comunicação móvel.

2.1 Automação

Para Silveira e Santos (1998) a automação é um conjunto de técnicas para a construção de sistemas ativos que dispõem da capacidade de atuar com uma eficiência otimizada, utilizando as informações recebidas do meio onde atuam. A automação pode ser identificada sempre que novas técnicas de controle são introduzidas em um processo.

De acordo com Moraes e Castrucci (2007), a palavra *automation* foi inicialmente utilizada pelo setor de *marketing* das indústrias de equipamentos na década de 1960 e buscava mostrar ao consumidor que seus processos industriais utilizavam computadores para realizar um controle automático.

Atualmente o conceito de automação expandiu-se e pode ser entendido como qualquer sistema apoiado em computadores, que visa a substituição do trabalho do ser humano em favor da segurança, qualidade do produto (ou serviço), rapidez da produção ou redução de custos. O grande desafio da automação é implementar com segurança e confiabilidade os sistemas de controle e comunicação necessários para o funcionamento eficiente do processo (MORAES; CASTRUCCI, 2007).

2.1.1 Arquitetura da automação industrial

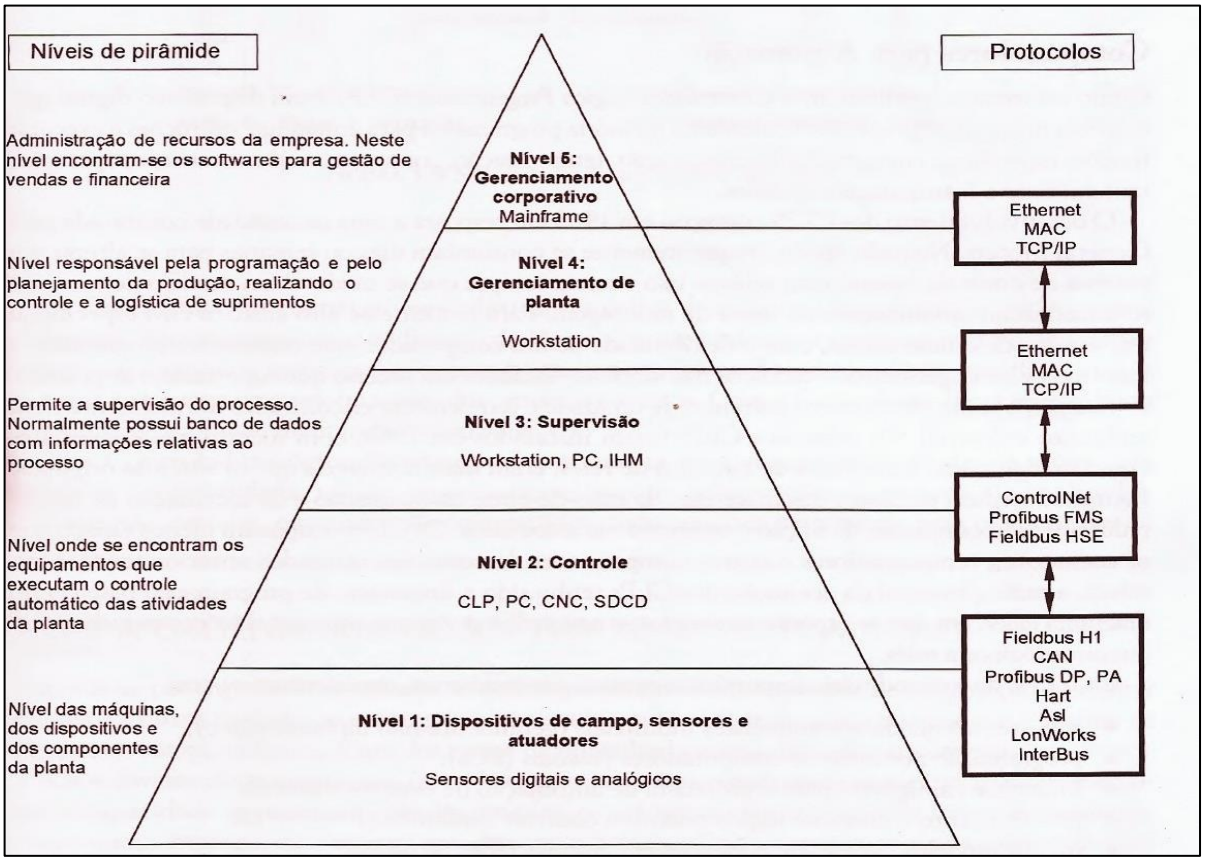
A automação industrial requer tarefas muito complexas e pode ser dividida em 5 níveis dentro de uma planta industrial, conforme apresentado a seguir com uma breve descrição:

- Nível 1: é o nível das máquinas, dispositivos e componentes (chão de fábrica).

- Nível 2: sistemas de controle e supervisão do processo.
- Nível 3: nível onde se realiza o controle do processo produtivo da planta, normalmente possui um banco de dados com informações de índices de qualidade.
- Nível 4: realiza a programação e o planejamento da produção, responsável pelo controle e logística dos suprimentos.
- Nível 5: responsável pela administração dos recursos da empresa, geralmente possui softwares para gestão de vendas e finanças, além de realizar a decisão e o gerenciamento de todo o sistema (MORAES; CASTRUCCI, 2007).

Na figura 1 é possível observar a ilustração desses níveis. As aplicações do padrão OPC estão normalmente situadas entre os níveis mais altos da pirâmide.

Figura 1 – Pirâmide de automação.



Fonte: Moraes e Castrucci (2007, P. 13).

2.2 Padrão OPC

Inicialmente o padrão OPC correspondia a sigla para OLE (*Object Linking and Embedding) for Process Control*, da tradução livre: vinculação e incorporação de objetos para controle de processo, ou seja, o OPC traduz o que um sistema supervisório manda para um controlador (MOREIRA e MAGALHÃES, 2010).

OLE (*Object Linking and Embedding)* é uma tecnologia desenvolvida pela *Microsoft* para possibilitar a integração de diferentes aplicações na plataforma *Windows*, visando melhoria no desempenho e confiabilidade do sistema (FONSECA,2002).

Segundo Fonseca (2002), esta tecnologia foi desenvolvida na década de 90 e sua continuação é definida como DCOM (*Distribuited Component Object Model*), que possibilita que um cliente acesse diferentes servidores ao mesmo tempo, da mesma forma que um servidor consegue disponibilizar suas funcionalidades a diferentes clientes.

Assim, com o padrão OPC, a comunicação e suas possíveis alterações podem ser feitas pelo usuário de forma simples. O OPC é independente de plataforma e foi desenvolvido por fabricantes de equipamentos e desenvolvedores de *software*, sendo utilizado pela primeira vez em 1996 (OPC FOUDATION, 2018).

Esse padrão é estudado e atualizado por um corpo técnico denominado “Fundação OPC”, responsável pelo desenvolvimento e manutenção do padrão (OPC FOUDATION, 2018).

O padrão OPC foi feito baseando-se na norma IEC 61131, elaborada pelo grupo IEC (*International Electro-technical Comission*), publicada em 1992 para estabelecer padrões de equipamentos CLPs (SILVA, 2011).

De acordo com a *OPC Foundation* (2018), existem dois tipos de OPCs: o OPC *Classic* e o OPC UA. O OPC *Classic* foi o primeiro a ser utilizado, restrito ao sistema operacional *Windows* para computadores pessoais, pois sua tecnologia é baseada no mesmo. Suas funções têm definições para acessar dados de processo, alarmes e dados de históricos, sendo um tipo de OPC utilizado em vários setores da indústria.

Por meio do OPC *Classic* foi atribuído o acrônimo *OLE for Process Control*, pois este tipo é restrito ao sistema *Windows*. Este tipo foi muito utilizado até 2008 quando foi desenvolvido o OPC UA (OPC Foundation, 2008).

O OPC UA (*Unified Architecture* – Arquitetura Unificada), foi lançado em 2008 pela OPC *Foundation*, e consiste em uma arquitetura orientada a serviços independentes, integrando todas as funcionalidades do OPC *Classic* em uma estrutura extensível (OPC Foundation, 2008).

Suas principais funções são, de acordo com OPC Foundation.b (2018):

- Equivalência funcional: atende e pode ser utilizado no lugar do OPC *Classic,* além de incluir outras funções, como a busca de servidores em computadores e redes locais, e a execução de programas definidos pelo servidor;
- Plataforma independente: pode ser utilizado em um computador, servidor na nuvem, microcontroladores, ou sistemas operacionais;
- Segurança: é adequado ao *firewall*, ao mesmo tempo que possui controles de segurança;
- Adição de novos recursos sem afetar os aplicativos já existentes.

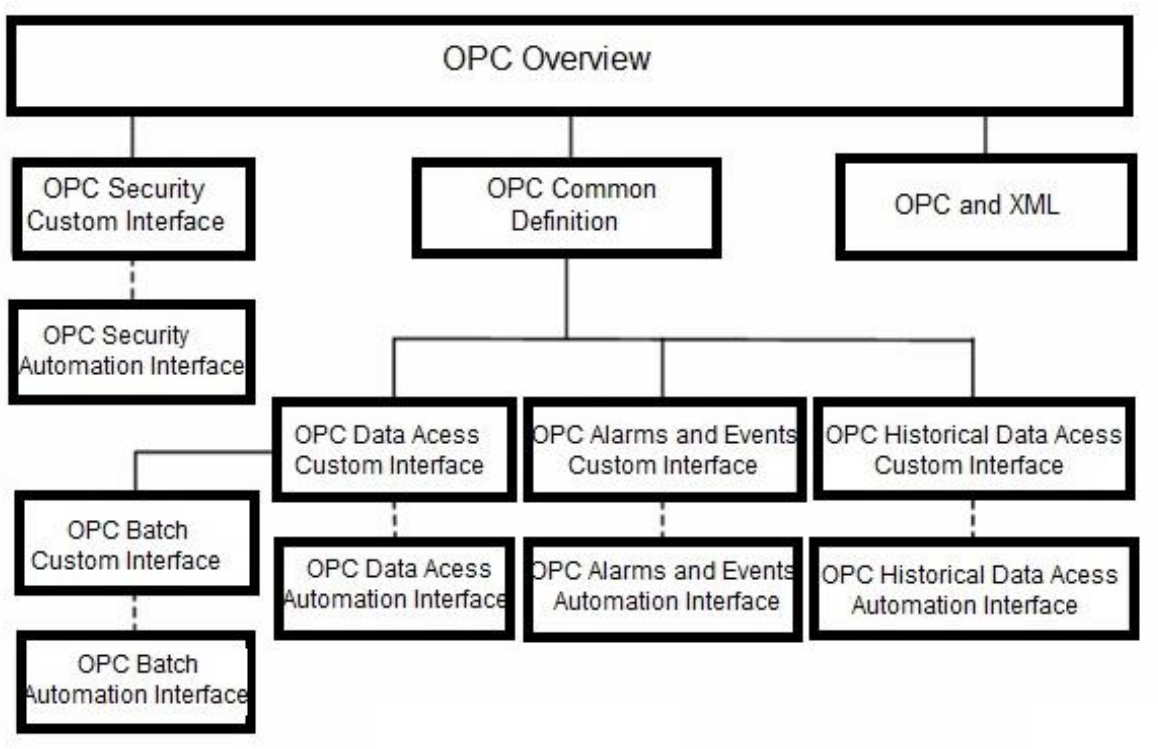
Com o surgimento deste segundo tipo de OPC, e a independência da utilização da plataforma *Windows*, a sigla do padrão OPC recebeu um novo significado: *Open Platform Communications* - Plataforma Aberta de Comunicação (OPC FOUNDATION,2018).

Como sua estrutura é extensível, o OPC possui diversas interfaces, que podem ser utilizadas separadas ou em combinação, sendo possível atender a várias aplicações em diversas áreas. A seguir são descritas as interfaces do padrão OPC, e a Figura 2 mostra sua composição e possíveis combinações (BALIEIRO,2008).

- OPC *Overview*: realiza a descrição geral dos campos de aplicação das especificações OPC.
- OPC *Common Definitions and Interfaces*: define as funcionalidades básicas para as demais especificações.
- OPC *Data Access Specification*: define a interface para leitura e escrita de dados em tempo real.
- OPC *Alarms and Events Specification*: define a interface para monitoração de eventos.
- OPC *Historical Data Access Specification*: define a interface para acesso a dados históricos.

- OPC *Batch Specification*: define a interface para acesso a dados de processos por batelada (batch), sendo uma extensão da OPC *Data Access Specification*.
- OPC *Security Specification*: define a interface para utilização de políticas de segurança.
- OPC *and XML*: realiza integração entre OPC e XML para aplicações via internet (BALIEIRO,2008).

Figura 2 – Interfaces do padrão OPC.



Fonte: Adaptado de Balieiro (2008, p. 42).

Assim, o padrão OPC permite a administração de três níveis diferentes de informação: coleta de dados de dispositivos inteligentes, processos de dados para DCS (sistemas de controle distribuído), e sistemas supervisório; sendo este último bastante utilizado na indústria (CUNHA, 2004).

2.3 Sistemas supervisórios

De acordo com Moraes e Castrucci (2007), sistemas supervisórios são sistemas digitais comumente implementados em plantas industriais que possuem

como finalidade a monitoração, operação e gerenciamento das variáveis do processo.

Estas variáveis são armazenadas em bancos de dados locais ou remotos e são utilizadas como registro histórico; podem servir de base para estudos no auxílio de tomada de decisões estratégicas de uma empresa (MORAES; CASTRUCCI, 2007).

Basicamente existem dois tipos básicos de variáveis em um processo industrial:

- Variáveis Digitais: possuem apenas 2 estados discretos, por exemplo: uma máquina ligada ou desligada; uma chave aberta ou fechada; um alarme que detecta alguma falha no processo (ativado ou desativado). Para Silveira e Santos (1998) uma variável é considerada digital quando varia bruscamente no tempo, tal variação discreta no tempo possui uma quantidade finita de possíveis valores dentro de um intervalo.
- Variáveis Analógicas: de acordo com Moraes e Castrucci (2007), uma variável é dita analógica quando assume valores em uma faixa determinada, por exemplo, a velocidade de um carro ou temperatura de um forno. Para Silveira e Santos (1998), uma variável é dita analógica quando varia continuamente no tempo e pode adquirir infinitos valores dentro de um intervalo qualquer.

Quanto à classificação dos sistemas supervisórios, de acordo com Moraes e Castrucci (2007), devem ser analisadas algumas características do processo, tais como complexidade, robustez e o número de entradas e saídas a serem monitoradas. Atualmente os dois grupos de sistemas supervisórios utilizados são:

- IHM / HMI (Interface Homem-Máquina / *Human Machine Interface*).
- SCADA (*Supervisory Control And Data Acquisition* – Aquisição de Dados e Controle Supervisório).

2.3.1 IHM/HMI

Uma IHM é um *hardware* composto normalmente por uma tela de cristal líquido e uma série de teclas para navegação e/ou inserção de dados em um

software destinado à sua programação (este *software* difere de acordo com o fabricante). Algumas das principais aplicações de uma IHM são:

- Visualização de alarmes em caso de condição anormal do processo.
- Visualização de dados dos equipamentos da linha de produção e dados do processo.
- Modificação de parâmetros do processo e/ou configurações de equipamentos.

As IHMs são sistemas supervisórios que tem suas características físicas de acordo com o ambiente a ser utilizado. Se utilizada no chão-de-fábrica, que normalmente caracterizam-se como ambientes agressivos, sua construção, de acordo com Moraes e Castrucci (2007), deve ser consideravelmente robusta e resistente a diversos fatores externos, como jatos de água diretos, variações bruscas de temperatura e umidade. Sua instalação geralmente é feita em locais próximos à linha de produção ou à estação de trabalho em supervisão.

Se o controle for feito à distância, pode ser utilizado um *smartphone* que, conforme PRIBERAM (2017), é um aparelho telefônico celular que possui conectividade e funcionalidades semelhantes a um computador pessoal de menor porte. Desta forma, pode ser considerado uma IHM quando instalados os *softwares* para esta finalidade, pois possui tela para visualização de dados e teclas de comando.

O nível de proteção e a capacidade de gerenciamento de variáveis são modificados de acordo com o processo onde a IHM é utilizada. A IHM tem como função traduzir os sinais vindos do controlador para sinais gráficos que tornam o entendimento das informações do processo mais rápido e fácil para o operador (MORAES; CASTRUCCI, 2007).

2.3.2 SCADA

O sistema SCADA foi criado para supervisão e controle em casos onde o número de variáveis de entrada e saída são elevados, sejam estas variáveis analógicas ou digitais. Estes sistemas possuem aplicação na área industrial, e visam à segurança dos recursos humanos e equipamentos, além do controle e da aquisição de dados de plantas industriais.

A base de *hardware* deste sistema pode ser um computador comum, facilitando a operação e reduzindo os custos do sistema, e por estas razões é um sistema muito empregado nas indústrias (MORAES; CASTRUCCI, 2007).

Para os sistemas SCADA existem dois modos de comunicação: comunicação por consulta (*polling*) ou comunicação por interrupção (*report by exception*), que serão explicados a seguir (MORAES; CASTRUCCI, 2007).

2.3.2.1 Comunicação por consulta (*polling*)

Segundo Moraes e Castrucci (2007), este tipo de comunicação utiliza o modo mestre/escravo, onde a estação central (mestre) controla as comunicações, efetuando a leitura dos dados de cada estação remota (escravo) que respondem em caso de recepção de um pedido da estação central.

Cada estação remota possui um endereço de identificação única, e caso uma delas não responda durante um período prédeterminado de tempo, novas tentativas de *polling* são realizadas antes que a central declare *time-out* e avance para a próxima estação.

As principais vantagens deste modo são:

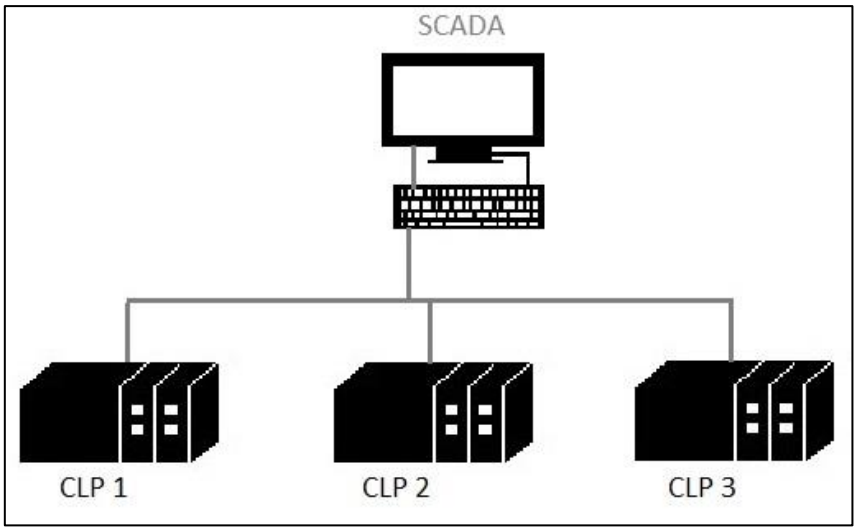
- Facilidade de aquisição de dados.
- Inexistência de colisões de tráfego de dados.
- Possibilidade de cálculo da largura de banda para comunicação e garantia dos tempos de resposta.
- Fácil detecção de falhas na ligação.

As desvantagens são:

- Estações remotas são incapazes de comunicar situações que necessitam de tratamento imediato à central.
- O aumento na quantidade de estações remotas aumenta o tempo de espera (maior o número de tentativas de polling).
- Comunicação entre estações remotas passam, obrigatoriamente, pela estação central (MORAES; CASTRUCCI, 2007).

Um exemplo de um sistema SCADA, utilizando comunicação por consulta, pode ser representado pelo diagrama da Figura 3:

Figura 3 – Sistema SCADA com comunicação por consulta.



Fonte: adaptado de Moraes e Castrucci (2007).

Como é possível observar na Figura 3, o sistema SCADA é o mestre e os CLPs são os escravos, ou seja; o sistema SCADA (a estação central), requisita e lê as informações contidas nos CLPs, que no caso são as estações remotas.

Dessa forma a leitura dos dados é feita em cada estação remota, que só responde em caso de recepção de um pedido da estação central. Cada estação remota possui um endereço de identificação única e, caso uma delas não responde, novas tentativas são realizadas antes que a central avance para a próxima estação.

2.3.2.2 Comunicação por interrupção (*report by exception*)

Neste modo de comunicação as estações remotas monitoram seus valores de entrada e saída e, caso encontre alguma alteração significativa ou extrapolação de valores predefinidos, inicia a transferência de dados com a central (MORAES; CASTRUCCI, 2007).

Antes de iniciar a transmissão dos dados, a estação remota verifica a disponibilidade do barramento de comunicação de dados e aguarda um período determinado antes de tentar enviar os dados novamente. Em caso de colisões de dados excessivas, a estação remota cancela a transmissão e aguarda a estação central enviar um pedido de leitura dos seus valores através de *polling*.

As vantagens deste modo de comunicação são:

- Redução no tráfego na rede.

- Detecção rápida de informação urgente.
- Comunicação entre as estações remotas (escravo / escravo).

As desvantagens são:

- Detecção de falhas de conexão mais lentas.
- Necessidade da intervenção do operador para obtenção de dados atualizados (MORAES; CASTRUCCI, 2007).

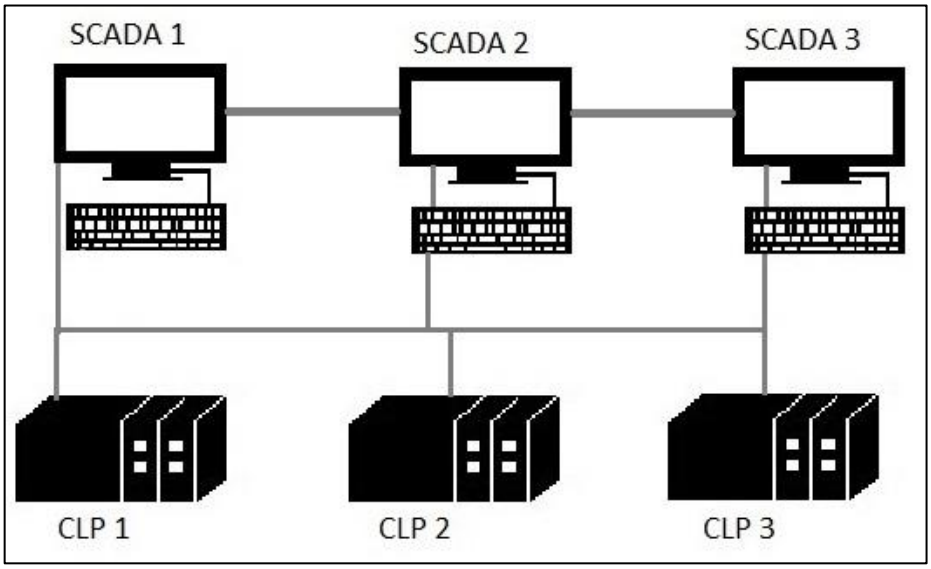
Este sistema é típico para controle de pequenos processos. Para o controle de processos industriais mais complexos é necessária uma arquitetura mais robusta, de forma que os dados disponibilizados pelo sistema sejam processados, formatados e manipulados para assegurar a qualidade da informação nos diversos sistemas de controle.

Assim, é necessário que os dados sejam recebidos de forma correta pelos destinatários. Portanto, é importante haver um critério para aquisição de dados das estações remotas no chão-de-fábrica (Moraes e Castrucci, 2007).

Uma solução simples para este problema consiste em efetuar o acesso de um sistema SCADA aos dados das estações remotas e disponibilizar os dados para outros sistemas através de uma rede de computadores totalmente independente das estações remotas.

Na Figura 4 é exibido o diagrama de um exemplo de utilização deste tipo de comunicação.

Figura 4 – Sistemas SCADA com comunicação por interrupção.



Fonte: adaptado de Moraes e Castrucci (2007).

Pela Figura 4, caso ocorra a perda do SCADA 1, o SCADA 2 pode ativar seu *driver* de comunicação e iniciar a leitura dos dados, anteriormente de responsabilidade do SCADA 1.

A grande vantagem desta arquitetura é a possibilidade do trabalho em redundância. Este trabalho em redundância é importante para a segurança do processo pois, caso o SCADA 1 perca informações, o SCADA 2 ainda possui as mesmas, tornando o sistema mais seguro.

2.4 Comunicação no OPC

O padrão OPC tem sua arquitetura baseada no modelo *client/server* (cliente/servidor), onde um aplicativo (definido como cliente OPC) realiza uma solicitação a um segundo aplicativo (definido como servidor OPC), que atende à solicitação. Este modelo pode ser utilizado em um único computador ou em rede, sendo que a última opção apresenta versatilidade, interoperabilidade e escalabilidade (MATRIKON, 2016).

A utilização de cliente e servidor OPC dependem do tipo de aplicação a ser feita, podendo haver necessidade de um cliente ou de um servidor, ou ambos. Geralmente, a comunicação com os dispositivos de campo são feitas por meio de servidores OPC e a monitoração de dados é feita por um cliente OPC (FONSECA,2002).

Cada servidor e cliente OPC dependem das especificações do controlador e sistema supervisório disponíveis. Resumidamente, um servidor OPC importa *tags* para o controlador e o cliente armazena os valores dessas *tags*, podendo haver modificação por meio do servidor ou cliente.

Essas *tags*, segundo Elipse (2005), são as variáveis envolvidas em um processo (podendo ser numéricas ou alfanuméricas), por exemplo a temperatura de uma caldeira ou a ativação de um sensor de presença. Ao importar *tags*, é criado automaticamente um grupo OPC, para maior organização do sistema.

2.4.1 Servidor OPC

O servidor OPC é um *software* para as interfaces OPC, e consegue responder à pedidos e fornecer dados para um ou mais clientes OPC em uma interface. O servidor OPC é disponível para diversos sistemas, sendo muito utilizado na automação industrial (MATRIKON, 2016).

Existem alguns desenvolvedores de servidores OPC, como Kepware, IOServer, Elipse, e Matrikon; que oferecem licenças de servidores OPC para disponibilizar os *drivers* dos protocolos necessários para a leitura dos dados dos equipamentos.

O servidor OPC consegue responder a requisições de um ou mais clientes, podendo ler, escrever e atualizar dados. Esses dados são gerados em forma de *tags*, ou seja, o servidor faz a conexão com as *tags* (variáveis do processo) e as disponibiliza ao cliente (RIBEIRO,2008).

A leitura de dados em um servidor OPC, conforme Covre (2011), pode ocorrer de 4 formas:

- Síncrona: depende de uma confirmação para execução antes de realizar nova leitura.
- Assíncrona: o servidor pode aceitar várias solicitações de dados, pois o cliente continua fazendo solicitações quando o servidor está buscando os dados requisitados anteriormente.
- Cíclica: há o envio de dados toda vez que o cliente requisitar.
- Por mudança de estado: o servidor envia ao cliente os dados que foram alterados.

A escrita pode ser síncrona ou assíncrona, e independe da leitura.

2.4.2 Grupo OPC

Ao importar as *tags*, o servidor OPC cria grupos OPC automaticamente, e um grupo tem a função de organizar as *tags* de acordo com as características de leitura dos itens, porque existem dados que precisam ser atualizados mais rapidamente que outros, ou dados que possuem características muito diferentes (COVRE, 2011).

Dessa forma, grupos OPC são utilizados para a organização do acesso de dados, separando em conjuntos as atividades de determinado processo. Por exemplo, é possível agrupar os valores de temperatura e outros de pressão em um processo (BALIEIRO,2008).

2.4.3 Cliente OPC

O cliente OPC é o responsável por receber as *tags* presentes no grupo OPC e geralmente é um sistema supervisório (RIBEIRO, 2008).

Qualquer cliente OPC consegue interagir ou requisitar dados de qualquer servidor OPC, independente do sistema ou fornecedor e, a partir do cliente OPC, os dados podem ser armazenados em um banco de dados para posteriormente serem atualizados e utilizados (COVRE, 2011).

2.5 Comunicação em rede com dispositivo de comunicação móvel

O uso de telefonia móvel no Brasil, conforme ANATEL (2017), é de mais de 240 milhões de linhas telefônicas no ano de 2017. Desta forma, o uso de um dispositivo de comunicação móvel para outras aplicações, como um *smartphone*, pode ser uma aplicação viável, pois é um equipamento amplamente utilizado.

A comunicação feita entre um dispositivo de comunicação móvel e um computador depende também das configurações do *firewall* do computador, que é, conforme *Microsoft* (2017), uma parte do modelo de segurança do *Windows*, filtrando o tráfego de rede no computador por questões de segurança, protegendo dados e reduzindo o risco de ameaças à rede.

Desta forma, é necessário incluir uma instrução *firewall* no computador que permita que um dispositivo externo, no caso o dispositivo de comunicação móvel, tenha acesso aos dados do servidor (o computador). Este acesso geralmente é realizado por meio da rede de comunicação em que ambos os aparelhos (dispositivo de comunicação móvel e computador) estão conectados.

Esta rede de comunicação normalmente é encontrada por meio do IP do computador então, após a configuração do *firewall* e com o servidor e cliente instalados, a troca de informações pode ser feita por meio do IP do computador e configurações que o aplicativo requerer (ELIPSE, 2014).

3 MATERIAIS

A seguir, encontra-se a especificação dos recursos utilizados para a realização deste trabalho.

3.1 Arduino UNO

O arduino pode ser entendido como um sistema microcontrolado que permite programar o processamento de entradas e saídas entre o dispositivo e os componentes externos conectados (MCROBERTS,2011).

Pode ser conectado a um computador, rede ou *internet*, para aquisitar e transmitir dados. O arduino possui um micro controlador Atmel AVR com suporte para entradas e saídas digitais e analógicas, e uma interface USB, que permite a comunicação de dados com um computador pessoal como um dispositivo, possibilitando um número maior de aplicações (MCROBERTS,2011).

Neste trabalho foi utilizado o modelo mais popular, o arduino UNO, como mostra a Figura 5.

Figura 5 – Arduino UNO.

Fonte: autoria própria.

Algumas especificações deste modelo:

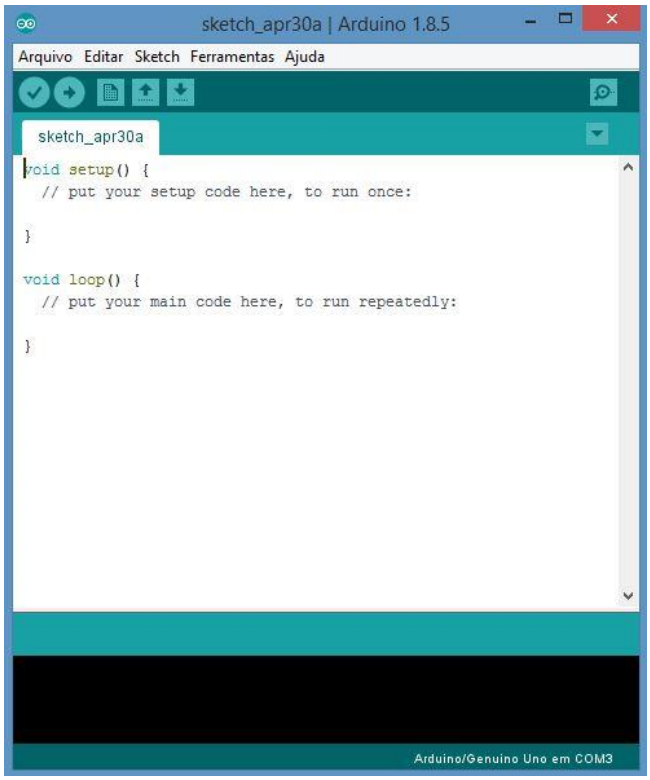
- 14 pinos de entrada e saída digital (6 podem ser utilizadas como saída PWM);
- 6 entradas analógicas;
- conexão USB;
- cristal oscilador de 16MHz;
- conexão ICSP;
- botão de reset;
- alimentação por fonte de alimentação externa ou por meio da conexão USB (Arduino, 2017).

3.1.1 Arduino *IDE*

Para a programação do arduino é utilizado o *software* livre “Arduino IDE”, que permite a programação em linguagem C. Este *software* verifica e permite realizar o *upload* do programa para o arduino via *USB* (MCROBERTS,2011).

Na figura 6 encontra-se a tela inicial deste *software*.

Figura 6 – Aduino IDE



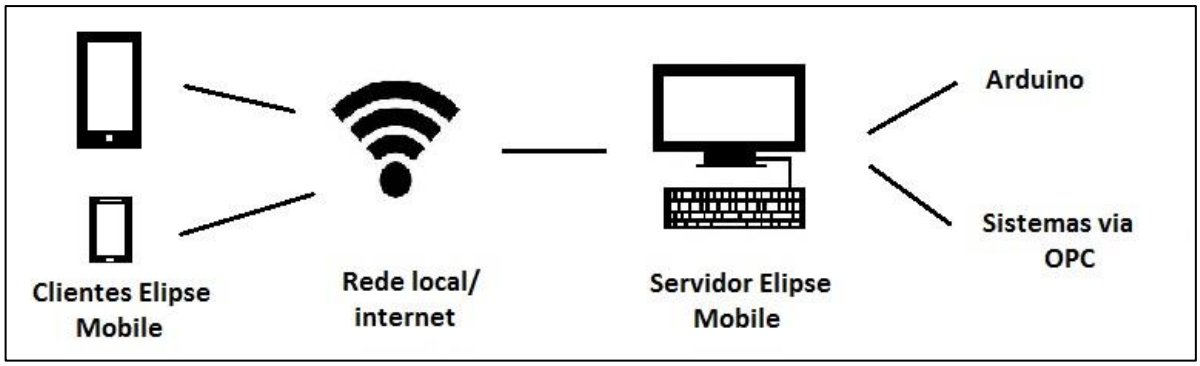
Fonte: Autoria própria

3.2 Elipse Mobile

O Elipse Mobile é um *software* que permite a integração com sistemas de automação, apresentando indicadores e possibilitando o comando de equipamentos do processo por meio de um dispositivo de comunicação móvel, como um *smartphone* ou *tablet*.

A utilização deste *software* está disponível em diferentes plataformas, e o acesso do processo pode ser feito também em ambiente *web*. O diagrama de sua conexão é mostrada na Figura 7 (ELIPSE,2017).

Figura 7 – Conexão Elipse Mobile



Fonte: adaptado de Elipse Mobile (2017)

Observando a Figura 7, os clientes do Elipse Mobile podem ser dispositivos como *tablets* e *smartphones*, e os usuários se conectam ao cliente por meio de programas instalados nestes dispositivos, sendo o servidor geralmente instalado em um computador, na mesma rede em que o cliente está conectado.

Desta forma os clientes podem se comunicar com o servidor por meio da *internet* ou rede local, que acessa os sistemas SCADA, efetuando a transferência de dados com o sistema ou equipamento do processo, ou chão de fábrica (sistema microcontrolado ou CLP). Esta comunicação pode ser feita com outros sistemas, já que utiliza o padrão OPC (ELIPSE MOBILE, 2017).

O Elipse Mobile funciona, resumidamente, por meio de um servidor instalado em um computador e um cliente, instalado em um dispositivo de comunicação móvel, como um *smartphone* ou *tablet*.

3.3 Elipse E3

O Elipse E3 é uma ferramenta SCADA, líder no mercado, utilizada para monitoramento e controle de processos. Possui possibilidade de comunicação com vários equipamentos, um simulador para verificação do processo, telas de controle, e comunicação via OPC Classic e UA.

Esta ferramenta permite diferentes aplicações, sendo flexível e confiável, com a comunicação via OPC e inclui algumas características como editor de telas, conexão com servidores remotos, fácil gerenciamento de aplicação, e possui interface de operação com o usuário, possibilitando simular a aplicação no servidor, validando as configurações no servidor e as alterações no cliente (ELIPSE E3, 2017).

3.4 ADO (*ActiveX Data Objects*)

O ADO (*Activex Data Objects*) é uma biblioteca da Microsoft que permite que dados sejam acessados e manipulados por clientes em um servidor ou banco de dados via comunicação OLE (MICROSOFT, 2013).

Esta biblioteca é de fácil utilização, alta velocidade, pouca sobrecarga de memória e permite a utilização em banco de dados simples, como o Excel (MICROSOFT, 2013).

Neste trabalho esta biblioteca foi utilizada no *software* Elipse E3, na realização do processo de armazenar dados digitais, e o registro dos dados foi feito em uma planilha Excel. Esta biblioteca foi escolhida por ser de livre acesso, fácil utilização e não precisar instalar programas, somente a programação no *software* SCADA e a configuração na planilha do banco de dados foi necessária.

3.5 *Software* "PLX-DAQ

Este *software* é utilizado para realizar o armazenamento de dados aferidos no arduino em um banco de dados. É bastante utilizado e é de livre acesso, por isso foi escolhido para ser usado neste trabalho.

O PLX-DAQ (*Parallax Data Acquisition tool*) é uma ferramenta de livre acesso da Parallax Inc, de aquisição de dados para o Excel. Esta adquire até 26

canais de dados e fornece fácil análise de planilhas, dados e monitoramento em tempo real (PARALLAX, 2018).

4 METODOLOGIA

O projeto deste trabalho foi implementado de modo a atender quatro aplicações exemplo, que envolveram o controle e armazenamento de dados analógicos e digitais, com as quais foram efetuados testes até a implementação final de cada aplicação.

As duas primeiras aplicações utilizaram um arduino UNO e consistiram em controlar (teste 1) e armazenar dados analógicos (teste 2). As duas últimas aplicações utilizaram o *software* Elipse E3, para o controle (teste 3) e armazenamento de dados digitais (teste 4).

Para a comunicação do servidor com os mesmos, também foi necessário adicionar as conexões deste e escolher qual o equipamento utilizado em cada aplicação/teste, sendo possível o reconhecimento e comunicação com o *software* e o sistema micro controlado.

A seguir são descritos os passos para a configuração em rede, a comunicação com os equipamentos utilizados em cada aplicação (no caso, duas realizadas em um arduino e duas no *software* SCADA com simulação “Elipse E3”), e os respectivos testes.

4.1. Configuração da rede

Para a comunicação e controle entre o cliente (dispositivo de comunicação móvel) e o servidor (computador), foi necessário configurar o *firewall* do computador onde estava instalado o servidor “*Elipse Mobile Server*”. Com o servidor instalado e configurado, os dados de usuário e senha da Elipse foram criados. No cliente OPC, o “*Elipse Client*”, a configuração consistiu na inserção do IP do servidor, do usuário e senha.

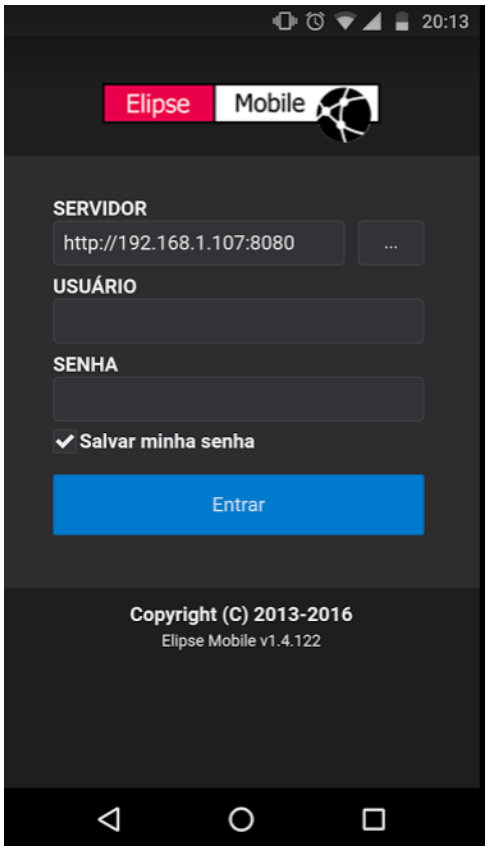
A configuração consistiu em acrescentar uma regra de *firewall*, para permitir a comunicação usando porta HTTP. Para isso ocorrer, foi necessário acessar as configurações avançadas do *firewall*, escolher a opção de inserir uma nova regra e especificar a mesma. As especificações encontram-se no apêndice A.

No servidor, além da instalação no computador, não foi necessário realizar outras configurações de rede, já que o mesmo foi instalado no computador que recebeu a nova regra de *firewall*.

Assim, no servidor somente foi necessário criar um usuário com senha para ser possível inserir e posteriormente acessar as telas de comando e visualização dos dados.

No cliente a configuração foi para acesso ao sistema a ser controlado, sendo necessário inserir o IP do servidor com a porta configurada na nova regra criada, no caso 8080, o usuário e a senha. Na figura 8 é possível observar a tela para entrada do Elipse *Client*.

Figura 8 – Tela para entrada do cliente.



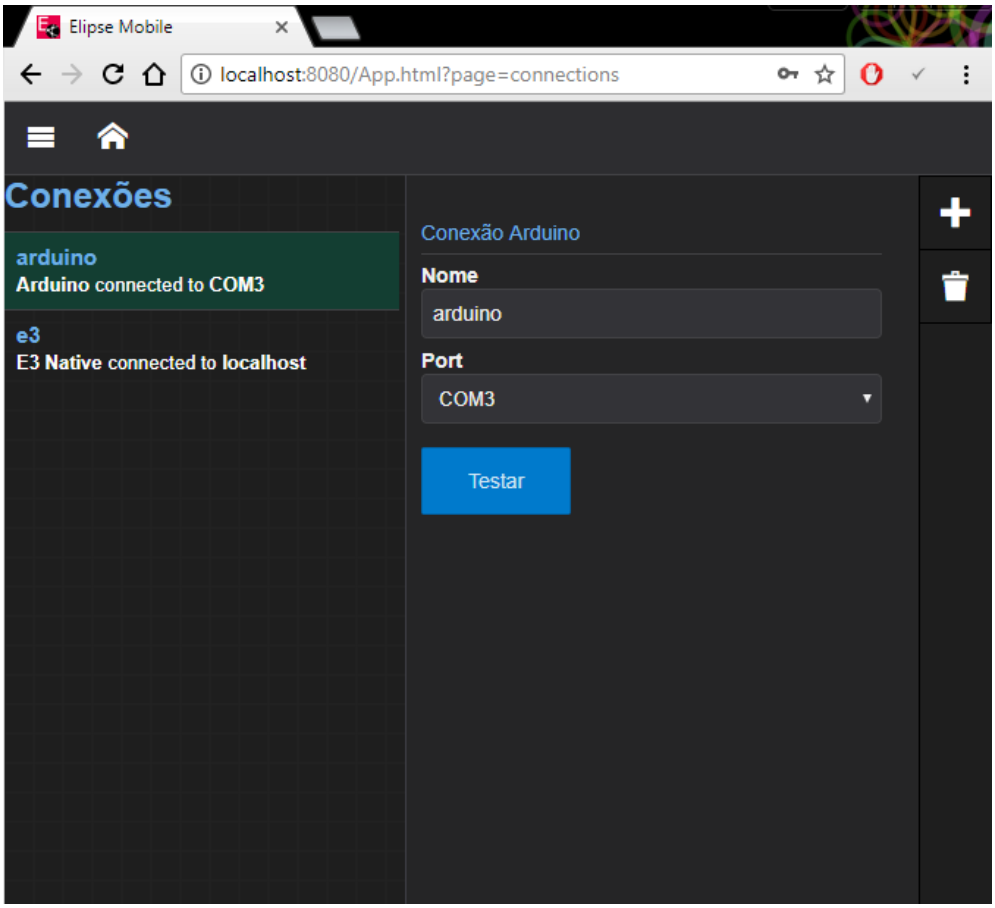
Fonte: autoria própria.

4.2. Configuração dos equipamentos utilizados no projeto

A configuração com os equipamentos foi feita pela opção de “conexões” do servidor (Elipse Mobile Server). Esta configuração foi feita pela escolha de algumas opções, que estão no apêndice B:

As duas conexões, após o preenchimento das opções, foram testadas. Na Figura 9 é exibida a imagem da tela onde é possível observar as duas conexões.

Figura 9 – Conexões Elipse mobile



Fonte: autoria própria.

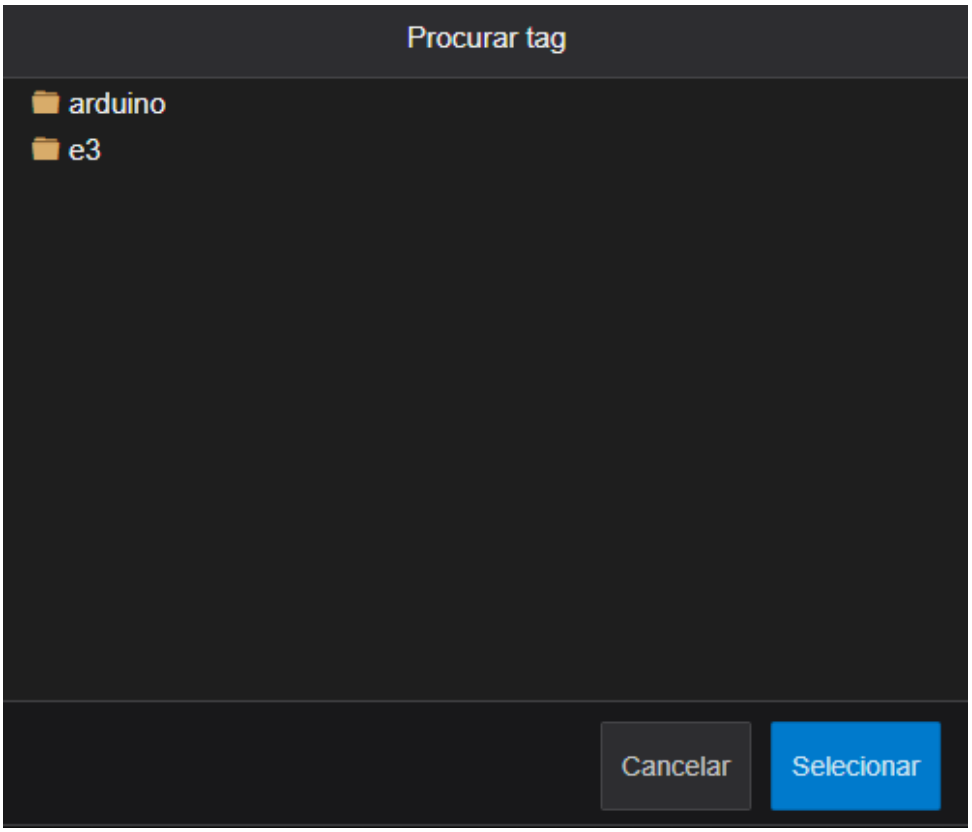
4.3. Configurações OPC

Para as aplicações o servidor OPC utilizado foi o *Elipse Mobile Server*, configurado em um computador, e o cliente foi o *Elipse Mobile Client*, utilizado em um dispositivo de comunicação móvel (*smartphone)*.

Em relação aos grupos e as *tags*, para cada aplicação foi criada uma tela do servidor e a organização das *tags* foi feita de acordo com o equipamento utilizado (arduino ou Elipse E3). Foram formados grupos de *tags*, e para as aplicações no Elipse E3 foi permitido utilizar somente duas *tags* em cada aplicação. No arduino foram utilizadas três *tags* para a primeira aplicação e uma para a segunda e terceira.

Na figura 10 é possível observar a imagem da tela dos grupos do arduino e Elipse E3. Como foram utilizadas poucas *tags* para as aplicações, a divisão das mesmas foi feita automaticamente de acordo com a ferramenta utilizada, no caso o arduino ou E3.

Figura 10 – Grupos OPC.



Fonte: autoria própria.

Assim, as *tags* das duas primeiras aplicações são localizadas na pasta “arduino” e as *tags* das duas últimas na pasta “e3”. Isso foi possível porque as *tags* da primeira aplicação são disponíveis quando o programa da aplicação 1 está aberto e conectado ao Elipse Mobile. Desta forma, as *tags* da aplicação 1 nunca estão disponíveis quando a aplicação 2 está conectada, e vice-versa, o que permitiu a construção de apenas uma pasta para estas *tags*. Observa-se que o mesmo ocorre entre as aplicações 3 e 4 feitas por meio do *software* Elipse E3.

4.4 Controle de LEDs

A primeira aplicação tem como objetivo apresentar visualmente um controle simples de liga/desliga de LEDs conectados ao arduino para o acionamento de três LEDs pelo cliente (*smartphone)* e servidor (computador pessoal).

No Elipse Mobile foi elaborada uma tela com botões liga/desliga e visualização do estado atual de cada LED, além do controle do acionamento dos mesmos. Para isto foi elaborado o respectivo código no arduino para permitir o

controle pelo cliente e servidor, através do *software* “Arduino IDE”. Na figura 11 encontra-se a descrição do código fonte para receber as entradas dos LEDs. Esta aplicação demonstrou claramente o controle realizado tanto pelo cliente quanto pelo servidor.

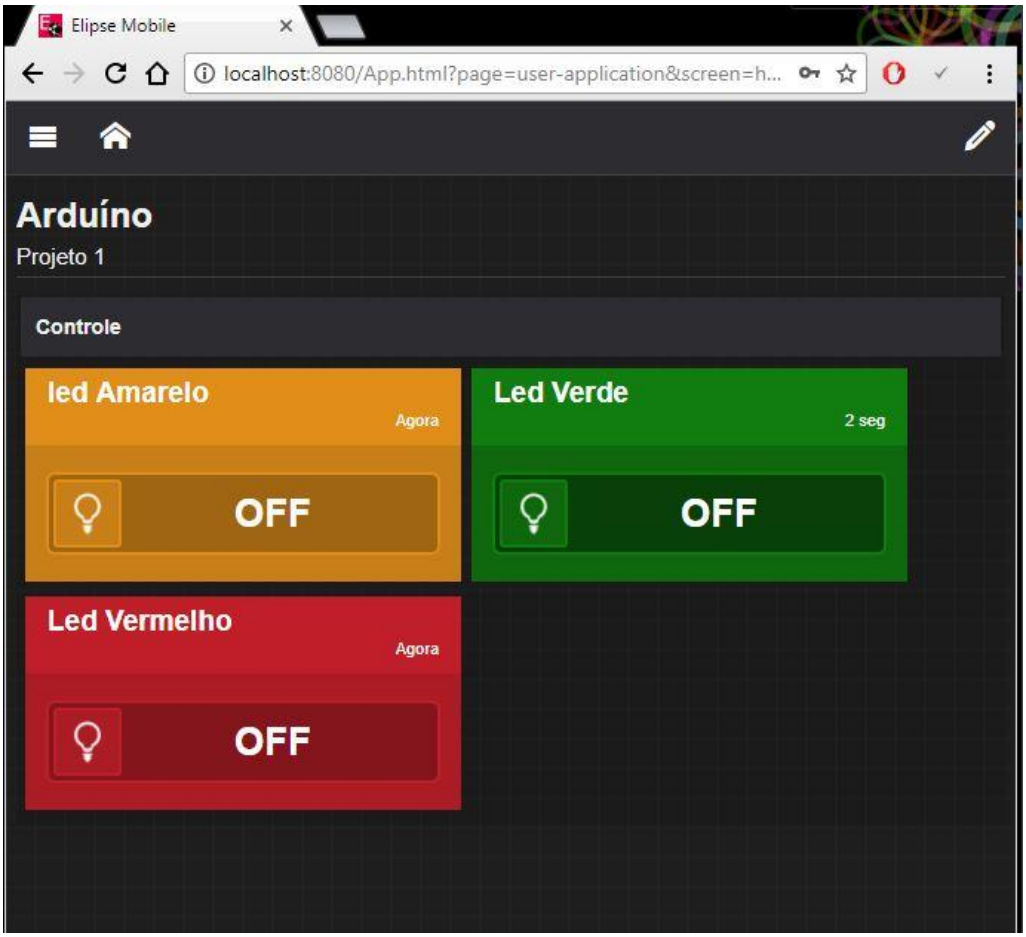
Figura 11- Código do arduino para o primeiro teste.

```
#include <modbus.h>
#include <modbusDevice.h>
#include <modbusRegBank.h>
#include <modbusSlave.h>
#include <ElipseMobile.h>

// recebendo as portas dos leds
ElipseMobile elipse;
int led_red = 12; // led vermelho
int led_green = 11; // led verde
int led_yellow = 9; // led amarelo

void setup()
{
  // permite utilizar as tags analógicas e digitais
  elipse.DigitalTags(14);  // 14 digital tags
  elipse.AnalogTags(6);  //6 analog tags

  // inicia a porta serial a 9600 bps:
  Serial.begin(9600);
  while (!Serial) {
    ;  // aguarda a porta serial conectar
  }
}
void loop()
{

elipse.ProcessCommands();  // Interface de comunicação com o Ellipse Mobile Server.
}
```

Fonte: autoria própria.

Para o arduino receber o Elipse Mobile Server, foi necessário também importar uma biblioteca do *software* na programação do arduino, disponibilizado pela Elipse. Posteriormente foi elaborada no servidor a tela de comando dos LEDs, exibida na figura 12.

Figura 12 – Tela de comando do primeiro teste.



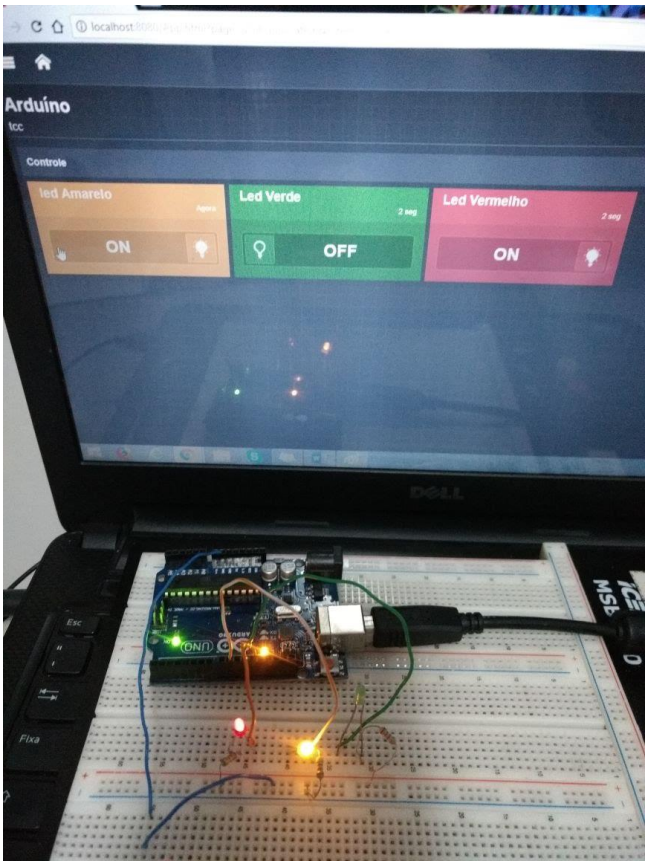
Fonte: autoria própria.

Como é possível observar, foram criados três botões de liga/desliga (do tipo *toggle*) e três sinalizadores: um vermelho, um verde e um amarelo, com os respectivos botões identificados com as cores e nome dos mesmos.

Em cada botão foi adicionada a *tag* do LED a ser controlado, por meio das configurações do botão, com espaço para acessar os grupos e selecionar o LED desejado. Nas figuras 13 é possível observar as imagens das ligações físicas entre o arduino e o servidor.

Neste teste foi possível observar o controle das variáveis de liga/desliga no cliente e servidor, e a atualização do estado na recepção do comando enviado pelo cliente ou pelo servidor. Assim foi possível validar a leitura e controle das variáveis.

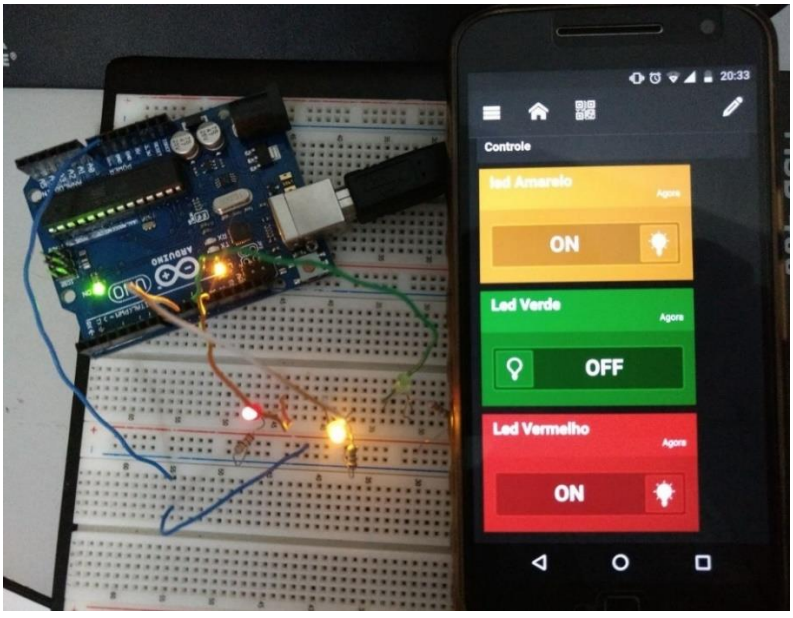
Figura 13 – Servidor e arduino no primeiro teste.



Fonte: autoria própria.

Na figura 14 é possível observar a tela de comando no celular cliente.

Figura 14 – Arduino e cliente no primeiro teste



Fonte: autoria própria.

4.5 Aquisição e armazenamento de dados analógicos

Utilizando o arduino UNO a segunda aplicação consistia em receber e armazenar dados analógicos de um potenciômetro conectado a uma entrada analógica do arduino. Foi necessário programar a configuração e o recebimento de dados da entrada analógica, e elaborar a tela com um botão de visualização do valor do potenciômetro. Para o armazenamento de dados foi necessário utilizar um *software* de uso livre, que efetua a comunicação entre o Excel e o arduino.

Este *software* é o “PLX-DAQ”, responsável por verificar a porta e a frequência de transmissão dos dados, configurados no código do arduino IDE. A programação para que os dados fossem formatados para uma tabela do Excel e permitir que fossem recebidos pelo servidor e cliente (configurações do Elipse Mobile) foi elaborada através deste *software*.

Para a visualização de um registro dos valores em relação a intervalos de tempo, foi gerado um gráfico com os valores adquiridos em intervalos de 4 segundos. Como apresentado, o teste para esta aplicação consistiu em adquirir valores de um potenciômetro por meio de um sistema arduino e armazenar estes valores em uma tabela do Excel. Na figura 15 é descrito este código fonte do arduino.

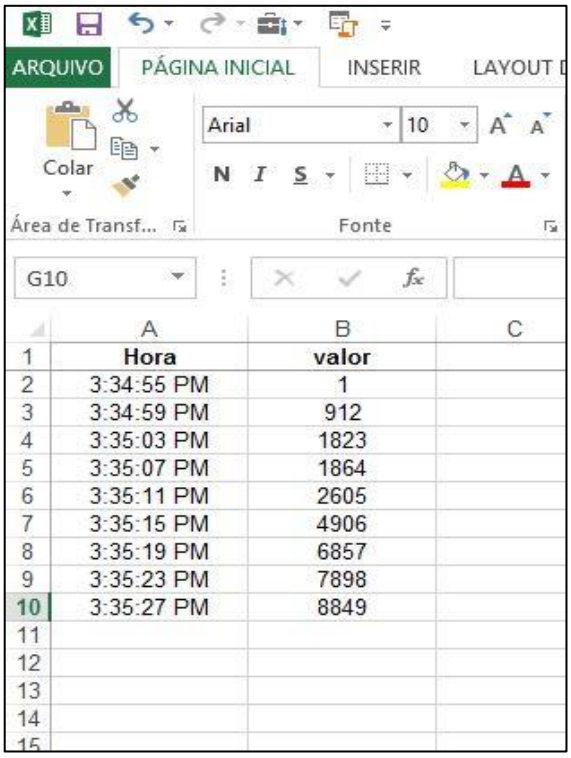
Figura 15 – Código do segundo teste.

```
#include <ElipseMobile.h>
ElipseMobile elipse;
const int pinoPotenciometro = 5;    //  variavel que define a porta do potenciometro.
int linha = 0;  // variavel que se refere as linhas do excel
int LABEL = 1;
int valor = 0;  // variavel que guarda o valor lido do potenciometro
void setup(){
 elipse.AnalogTags(6);  // habilita a utilização de entradas analógicas
Serial.begin(9600);  //  inicialização da comunicação serial
while (!Serial){
  ;                 // aguarda a porta serial conectar
}
Serial.println("CLEARDATA");            // Reset da comunicação serial
Serial.println("LABEL,Hora,valor,linha");   // Nomeia as colunas
}
void loop(){
elipse.ProcessCommands();
delay(2000);
valor = analogRead(pinoPotenciometro);   // faz a leitura do potenciometro
linha++;   // incrementa a linha do excel para que a leitura pule de linha em linha
Serial.print("DATA,TIME,"); //inicia a impressão de dados
Serial.print(",");Serial.print(valor);
Serial.println(linha);
if (linha > 10) //laço para limitar a quantidade de dados
{
linha = 0;
Serial.println("ROW,SET,2"); // alimentação das linhas com os dados
}
delay(2000);  // espera 200 milisegundos
}
```

Fonte: autoria própria.

No banco de dados, no registro de valores do potenciômetro foram adquiridos dados em intervalos de 4 segundos seguido do horário da aferição. A tabela de valores dos dados encontra-se na Figura 16, e na Figura 17 é possível observar o gráfico gerado com os valores em função do tempo.

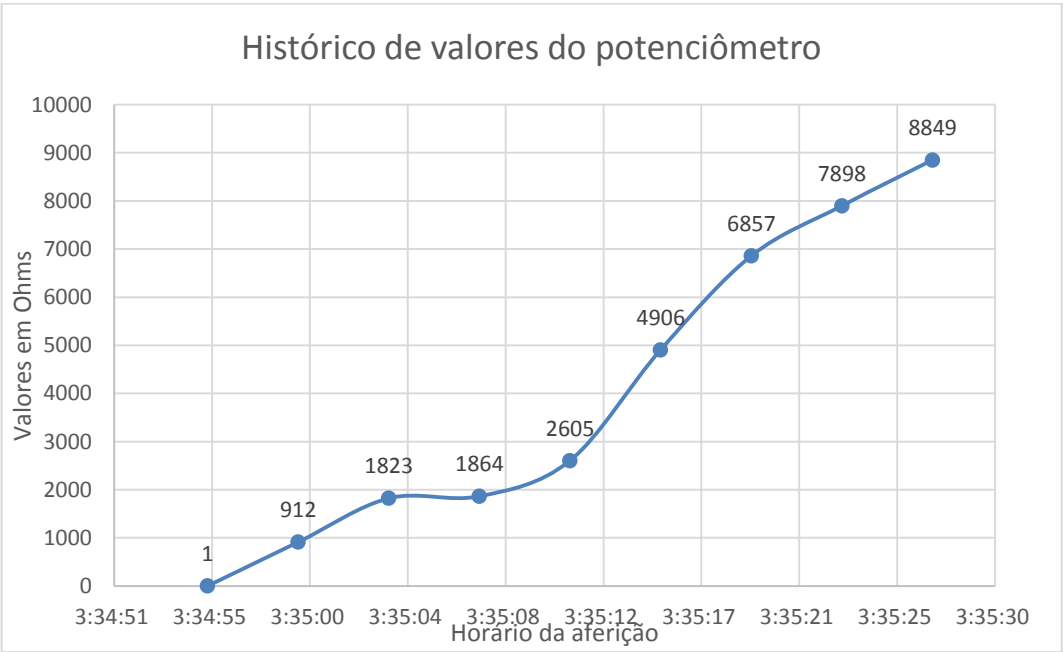
Figura 16 – Valores registrados no Excel.



	A	B	C
1	Hora	valor	
2	3:34:55 PM	1	
3	3:34:59 PM	912	
4	3:35:03 PM	1823	
5	3:35:07 PM	1864	
6	3:35:11 PM	2605	
7	3:35:15 PM	4906	
8	3:35:19 PM	6857	
9	3:35:23 PM	7898	
10	3:35:27 PM	8849	
11			
12			
13			
14			

Fonte: autoria própria.

Figura 17 – Gráfico de valores aferidos de 4 em 4 segundos.

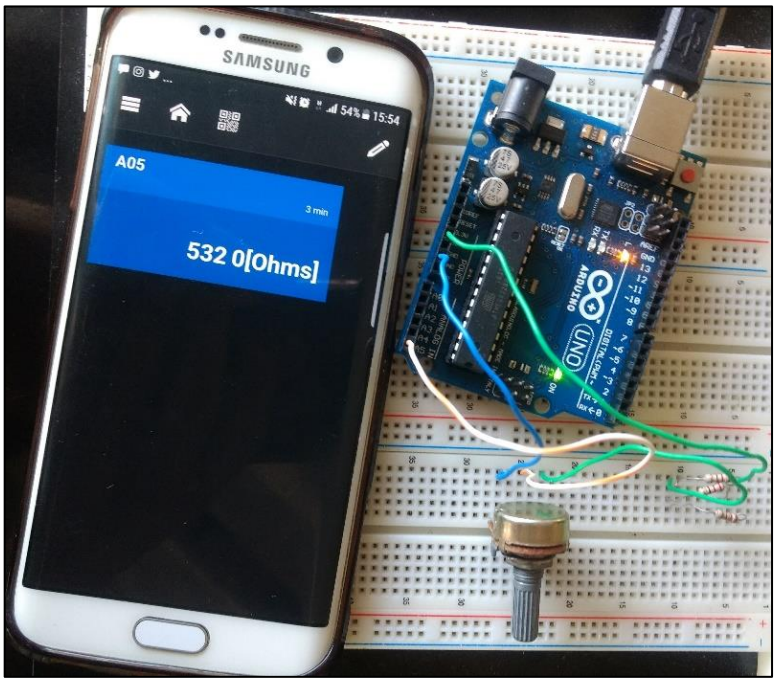


Fonte: autoria própria.

No servidor, foi elaborada uma tela para visualizar o valor aferido. Para o teste desta aplicação, o papel do servidor e cliente foi visualizar, apresentar e registrar os dados aferidos, já que a mudança de valores do potenciômetro foi feita de forma manual. A ligação física do arduino com o potenciômetro é exibida pela imagem da figura 18, onde é possível observar também a tela do cliente.

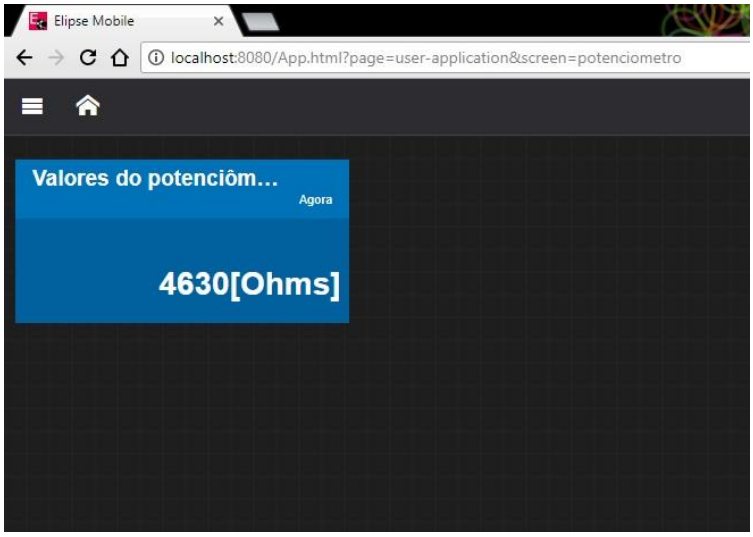
Na figura 19 é possível observar uma imagem da tela elaborada para o servidor.

Figura 18 – Arduino e cliente no segundo teste.



Fonte: autoria própria

Figura 19 – Tela do servidor da segunda aplicação.



Fonte: autoria própria.

Dessa forma, nesta aplicação foi possível realizar um processo muito utilizado na indústria, que é a realização do histórico de dados obtidos dos equipamentos, com a verificação e armazenamento de valores via dispositivo de comunicação móvel.

4.6 Controle de esteiras no Elipse E3

A terceira aplicação e teste consistiram em realizar a comunicação do cliente e servidor com o *software* SCADA Elipse E3, que possui um simulador para teste do funcionamento do programa. Este *software* com simulador devido à sua grande utilização na indústria e fácil visualização.

Neste *software* simulador foi elaborada a simulação de 3 esteiras industriais com controle separado, ou seja, o funcionamento de uma esteira não interfere nas outras duas. Esta versão do Elipse E3, por ser gratuita, não permite o controle de mais de duas *tags*, sendo realizado o controle de duas esteiras e a terceira utilizada para demonstrar a não interferência do controle das outras esteiras.

Assim como nas duas primeiras aplicações, foi elaborada uma tela específica para este projeto no *Elipse Mobile*, e o controle também foi realizado pelo servidor e cliente, assim como a visualização do estado (ligado ou desligado) de cada esteira, tanto no servidor quanto no cliente.

Após os testes de funcionamento das esteiras, foi feita a conexão com o servidor e cliente, com a validação dos resultados efetuada por meio da função de simulação desde *software*.

A simulação, como já citado, foi feita no *software* Elipse E3, e funcionou da seguinte forma: ao clicar no botão liga, a esteira inicia seu funcionamento e o respectivo botão torna-se verde. Ao clicar novamente no botão, retorna à cor vermelha e a esteira para de funcionar.

Assim, inicialmente foram elaboradas três esteiras com o efeito animado de translação, responsável por representar a esteira em funcionamento. Foram inseridos botões de controle para cada uma das esteiras, e o controle nesta etapa foi elaborado com o comando de liga/desliga conectado à animação de translação da imagem, com o efeito da esteira em movimento (em funcionamento).

Estas conexões são estabelecidas por meio de associações entre as figuras e comandos por meio de *tags* internas, ao clicar nas propriedades da figura e botão. Para a mudança de cor dos botões (vermelho quando não funcionando e verde em funcionamento), também foi realizada esta associação de *tags*.

Finalizando, a tela foi simulada e o controle das esteiras foi verificado para a próxima etapa: o controle das esteiras pelo Elipse Mobile. Na figura 20 é exibida a imagem da tela das esteiras elaborada no Elipse E3.

Figura 20 – Esteiras no Elipse E3.

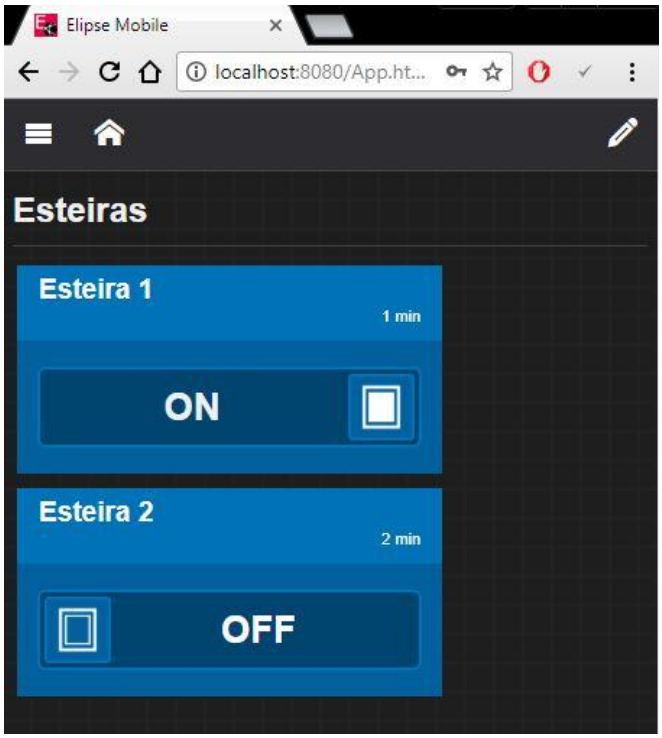
Fonte: autoria própria.

Dessa forma, foi realizada a conexão do Elipse E3 no Elipse Mobile, e foi elaborada a tela de controle do mesmo. Como no primeiro teste, também foram utilizados botões *toggle* que possuem a função de ligar e desligar as esteiras. Nesses botões, foram associadas as tags do Elipse E3 para o respectivo controle.

Dessa forma, na tela do servidor foram criados três botões, um para cada esteira, de modo que ao clicar nos mesmos era alterado o estado de funcionamento ou não da esteira do respectivo botão.

Observa-se que um botão não altera o estado do funcionamento de outra esteira. A Figura 21 exibe a imagem da tela de controle elaborada para o Elipse Mobile Server, a Figura 22 para a tela do Elipse E3 e tela de controle no cliente.

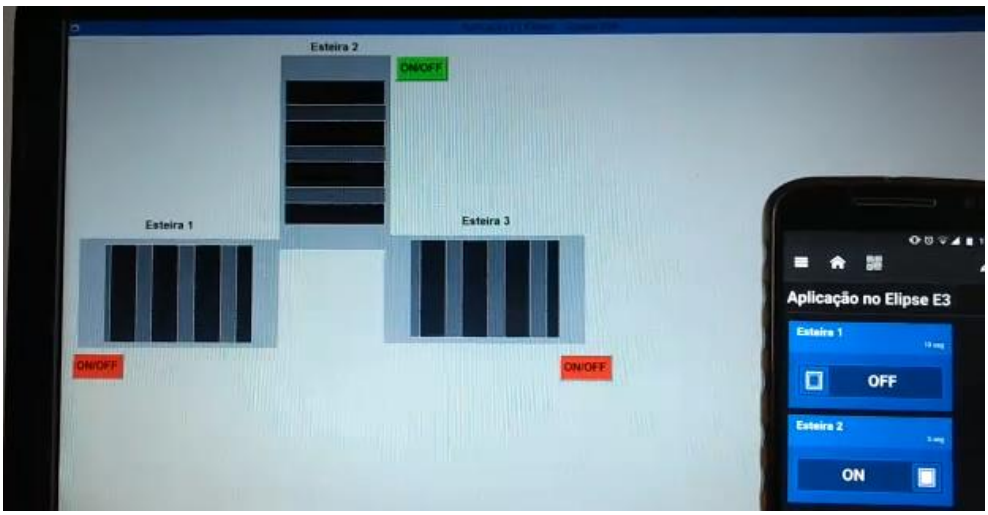
Figura 21 – Tela de comando do terceiro teste.



Fonte: autoria própria.

Como o Elipse E3 possibilita a conexão com outros sistemas e dispositivos, como um CLP, o controle deste por meio de um dispositivo de comunicação móvel proporciona mais uma opção de automação para aplicações industriais.

Figura 22 – Elipse E3 e cliente.



Fonte: autoria própria.

4.7 Aquisição e armazenamento de dados digitais

Na quarta aplicação e teste, foi verificado o armazenamento de dados utilizando o *software* Elipse E3 e o Excel. Para isto, foi usado o ADO (*ActiveX Data Objects*) no Excel, que é uma biblioteca da *Microsoft* que permite o acesso a banco de dados e utiliza a linguagem OLE. A configuração desde componente é realizada por meio de programação no Elipse E3, através do qual foi elaborado um programa para inserir comandos para utilizar o Excel e o controle no servidor e cliente.

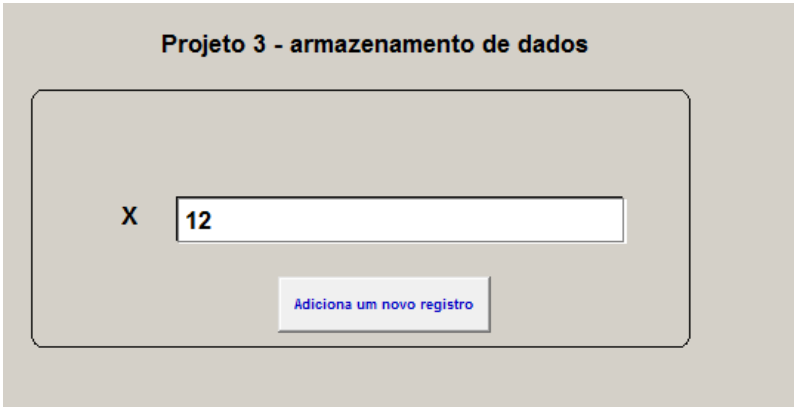
Inicialmente foi criada uma tela no Elipse E3 para receber valores e armazená-los no Excel e a criação de uma *tag* para realizar esta conexão via ADO, que recebeu o nome de DBconn e, na configuração de inicialização "*OnStartRuning*", foi inserido o código para a conexão com o arquivo do Excel. Este código fonte é exibido na Figura 23, e na Figura 24 é possível observar a imagem final da tela.

Figura 23 – Código para conectar o Excel ao Elipse E3.

```
Sub DBConn_OnStartRunning()

    'Cria conexão com o banco de dados
    Set Value = CreateObject("ADODB.Connection") 'ADODB = objeto de conexão ADO

    'String de conexão

    Value.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=ExcelE3.xls;Extended Properties=Excel 8.0"

End Sub
```

Fonte: autoria própria

Figura 24 – Tela no Elipse E3.



Fonte: autoria própria.

Como é possível observar, esta tela é utilizada somente para receber valores para x. Posteriormente foi selecionada no Excel a coluna da tabela que seria

utilizada, por meio da configuração “ExcelData”, e foi criada mais uma *tag*, que em sua configuração de inicialização “*OnStartRunning*” permitiu inserir valores na coluna selecionada. O respectivo código fonte está exibido na figura 25.

Figura 25 – Código para permitir a utilização da coluna selecionada no Excel.

```
Sub RS_OnStartRunning()

    'Cria o recordset
    Set Value = CreateObject("ADODB.Recordset") ' permite adicionar dados
    Set DBConn = Application.GetObject("Dados.DBConn").Value ' permite utilizar a tag DBconn

    Value.Source = "ExcelData" ' seleção da coluna realizada no Excel
    Value.ActiveConnection = DBConn ' indica qual conexão irá utilizar, no caso a tag DBconn
    Value.CursorType = 1 'adOpenKeyset 'tipo de visualização
    Value.LockType = 3 'adLockOptimistic ' tipo de acesso
    Value.Open

End Sub
```

Fonte: autoria própria.

Para receber os valores pela tela foi criada mais uma *tag* e o código de programação para enviar os dados ao Excel, vinculado ao comando de clicar o botão, em que a *tag* foi associada (as *tags* anteriores foram associadas na inicialização). Este código fonte está descrito na Figura 26.

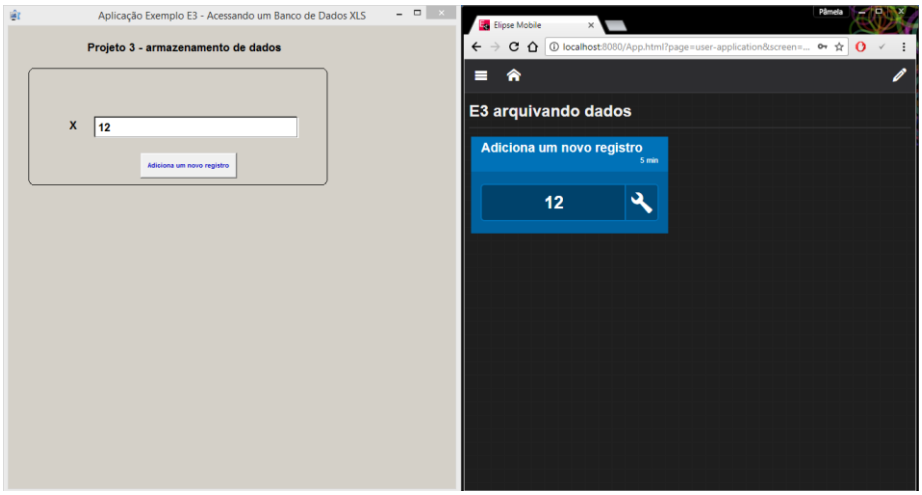
Figura 26 – Código para envio dos dados.

```
Sub CommandButton1_Click()
    dim i
    Set RS = Application.GetObject("Dados.RS").Value
    set RPos = Application.GetObject("Dados.RPos")

    RS.AddNew       'adiciona um novo registro
        RS.Fields("X") = Application.GetObject("Dados.X").Value


    RS.Update       'salva quaisquer mudanças feitas na linha corrente do objeto Recordset
    RS.Requery      'atualiza os dados em um objeto Recordset

    RS.MoveLast
    Application.GetObject("Dados.NRecords").Value = RS.RecordCount
    RPos.Value = RS.RecordCount

End Sub
```

Fonte: autoria própria.

Após estas configurações, foi realizada a conexão com o Elipse Mobile, da mesma forma que a terceira aplicação, e o controle foi executado. Na figura 27 é possível observar as imagens das telas do Elipse E3 e do servidor, e na Figura 28 os valores armazenados no Excel.

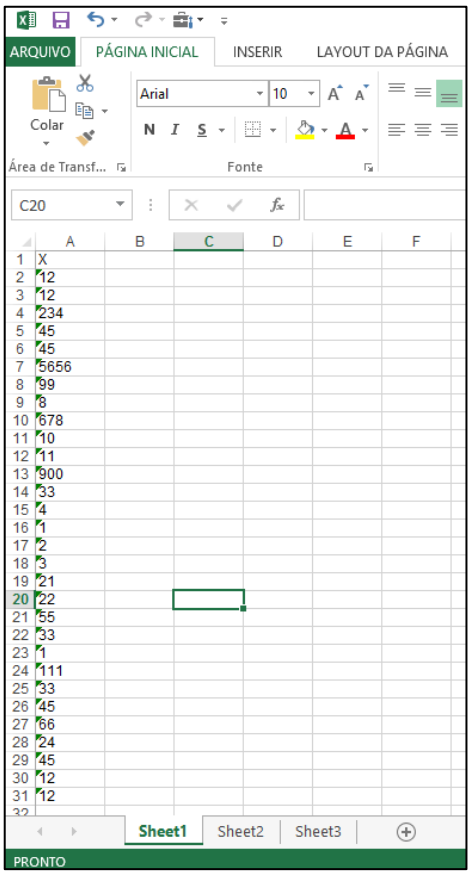
Figura 27 – Elipse E3 e servidor.



Fonte: autoria própria

Desta forma foi possível validar o armazenamento de dados no controle, o que pode ser aplicado na indústria como uma mudança de valores de segurança para um processo, por exemplo. O armazenamento dos dados pode ser muito importante, pois permite a realização de um histórico de funcionamento do sistema.

Figura 28 – Dados armazenados no Excel.



Fonte: autoria própria

5 RESULTADOS E DISCUSSÕES

A seguir encontram-se um resumo dos resultados obtidos nos quatro testes para as quatro aplicações descritas anteriormente, assim como comentários acerca dos resultados obtidos para o a implementação executada.

5.1 Teste de controle liga/desliga LEDs

Na primeira aplicação, onde foi utilizado um dispositivo de comunicação móvel para realizar o comando de liga/desliga das saídas digitais do arduino, foi observado um resultado satisfatório.

Assim foi possível a alteração do estado dessas variáveis sem a necessidade de uma conexão física entre smartphone e o arduino, o que pode facilitar e agilizar todo um processo, já que as alterações podem ser feitas longe do equipamento a ser controlado.

5.2 Teste de aquisição e armazenamento de dados analógicos

Na segunda aplicação, armazenamento de dados analógicos, foi utilizado um *software* que permitiu a comunicação do arduino com o servidor e o banco de dados. Com este *software* foi possível não só armazenar estes dados mas também a criação de um gráfico com o histórico dos dados adquiridos do potenciômetro, registrando também em qual tempo foi feito o armazenamento dos dados.

A escolha do *software* foi feita analisando sua aplicação, se é bastante utilizado e se é de plataforma livre. O software utilizado é de plataforma livre, bastante utilizado e foi feito por uma empresa voltada a possibilitar e facilitar diversas aplicações com arduino, por isso este foi o escolhido.

Desta forma, o cliente e o servidor conseguiram ler os dados do potenciômetro (variável analógica) conectado ao arduino de forma remota. Nesta aplicação apresentaram-se alguns ruídos na leitura dos dados analógicos devido o armazenamento dos dados. Estes ruídos apareciam da seguinte maneira. Dentro do intervalo de medição (4 segundos) apareciam valores muito elevados, e em seguida os valores reais do potenciômetro.

Porém, como esses ruídos foram perceptíveis (valores discrepantes), foram considerados aceitáveis ao processo, já que não interferiram no resultado final, pois tanto a leitura quanto o armazenamento dos dados na planilha foram realizados de forma eficiente, gerando um resultado satisfatório.

5.3 Teste de controle de esteiras no Elipse E3

A terceira aplicação utilizou um sistema simulado de 3 esteiras industriais com controle independente e que também devem ser ativadas e desativadas pelo dispositivo de comunicação móvel. O resultado obtido pode ser considerado satisfatório, pois foi possível controlar as esteiras separadamente.

Foi utilizada uma versão do Elipse Mobile (servidor e cliente) em que é possível importar somente 2 *tags* quando a conexão é realizada com o *software* SCADA Elipse E3.

Tendo em vista essa limitação, só foi possível controlar 2 esteiras simultaneamente mas, mesmo com essa limitação de *tags,* foi possível observar que as esteiras podem ser controladas separadamente, e o controle de uma não interfere no funcionamento das outras.

5.4 Teste de aquisição e armazenamento de dados digitais

Na quarta aplicação foi realizada a verificação do armazenamento de dados digitais, nesta foi utilizado o *software* SCADA Elipse E3 e, para haver o armazenamento dos dados, foi usado novamente o Excel, por meio da biblioteca da *Microsoft* ADO no servidor.

A utilização desta biblioteca permitiu o armazenamento de dados e foi escolhida por utilizar a comunicação OLE, que é a base da comunicação OPC, facilitando então a utilização e entendimento desta.

Como foi possível modificar o valor da variável no servidor e cliente, sendo no último de maneira remota, e verificar após isso os valores no banco de dados, o resultado encontrado foi satisfatório.

Diferente da outra aplicação que houve armazenamento de dados, nesta não houve ruídos. Isso se deve ao fato de serem alterações digitais e não analógicas, que foi o caso anterior.

6 CONSIDERAÇÕES FINAIS

Este trabalho consistiu em implementar um sistema de comunicação com controle via dispositivo de comunicação móvel. O sistema utilizou *softwares* que possibilitassem a utilização do padrão OPC, um padrão que possibilita a comunicação entre diferentes equipamentos e controladores.

Para este trabalho, foi utilizado um servidor e um cliente para quatro situações diferentes. O servidor foi instalado em um computador e toda a configuração do controle dos processos foi realizada no mesmo.

Após isso, foi configurado o cliente, que teve seu *software* instalado em um *smartphone*. Dessa forma, o cliente conseguia realizar a função de enviar comandos de liga/desliga ou alteração de dados, sem estar conectado fisicamente ao computador.

A comunicação do OPC utilizou o tipo por mudança de estado, e o sistema SCADA teve o modo de comunicação por consulta.

Foram realizados então quatro testes para validar o controle e a comunicação do trabalho. O primeiro teste utilizou um controlador simples, no caso um arduino, enviando sinais para ligar e desligar LEDs e realizar a leitura de status dos mesmos. O segundo, utilizando o mesmo controlador, para receber os dados de um potenciômetro e armazená-los.

Já o terceiro projeto consistiu em utilizar um *software* mais industrial, o Elipse E3, que simulou o funcionamento de esteiras rolantes, realizando comando de ligar ou desligar e também verificar o status de cada esteira.

O último teste consistiu em testar a capacidade de armazenar dados digitais, o que o padrão OPC também permite, e neste caso os mesmos foram armazenados com sucesso.

Como foram realizados testes com variáveis analógicas e digitais, foi possível verificar que, ao trabalhar com variáveis analógicas é necessário um cuidado maior em relação ao controle e configuração. Isto foi observado no teste em que era necessário armazenar os valores analógicos em uma planilha, nesta etapa foram observados ruídos quando os valores eram apresentados.

Já no outro teste que também armazenava dados, não ouve a observação desses ruídos, já que as variáveis eram digitais. Como o ruído observado foi

pequeno e não prejudicou a observação no cliente e servidor, nem no registro dos dados, foi considerado que o mesmo não interferiu no processo.

A utilização do padrão OPC foi importante porque possibilita a interação de outros equipamentos que podem ser utilizados futuramente em continuações deste trabalho, como a inclusão de um CPL industrial nos projetos que utilizam o *software* Elipse E3, pois o mesmo, além de realizar as simulações também permite a utilização de controladores como o CLP.

Além da utilização de outros equipamentos, o controle via dispositivo de comunicação móvel facilita o trabalho na indústria, podendo aumentar a eficiência e a flexibilidade em um processo industrial, já que o que foi feito nos testes é muito utilizado na indústria, como ligar ou desligar motores via sistema supervisório e armazenar dados para histórico.

REFERÊNCIAS

AGÊNCIA NACIONAL DE TELECOMUNICAÇÕES. Disponível em: < http://www.anatel.gov.br >. Acesso em: 20 dez. 2017.

ARDUINO, **Arduino Uno** – Overview. Disponível em: < https://store.arduino.cc/usa/arduino-uno-rev3 >. Acesso em: 10 dez. 2017.

BALIEIRO, Ricardo Luis. **Desenvolvimento de uma ferramenta computacional para aquisição via internet de dados de dispositivo de campo em ambientes fieldbus**. Dissertação de mestrado – Universidade de São Paulo, São Carlos, 2008.

CAPELLI, Alexandre. **Automação industrial**: controle do movimento e processos contínuos. 2. ed. São Paulo: Érica,2006.

COVRE, Helber Peixoto. **Integração de dados dos sistemas de proteção de subestações distribuidoras.** Dissertação de mestrado – Universidade de São Paulo, São Paulo, 2011.

CUNHA, Márcio José da. **Implementação de uma ferramenta computacional aplicada no processo de identificação de sistemas em ambientes Fieldbus Foundation.** Dissertação de mestrado – Universidade de São Paulo, São Carlos, 2004.

ELIPSE E3. Disponível em: < https://www.elipse.com.br/produto/elipse-e3/ >. Acesso em: 20 dez. 2017.

ELIPSE Software. **Tutorial-** Elipse Scada: HMI/Scada Software. Versão 2.27. Elipse Software Ltda,2005.

ELIPSE Sofware. **How to connect to the Elipse Mobile Server using your phone.** 27 ago. 2014. Disponível em: < http://blog.elipse.com.br/blog/2014/8/27/how-connect-to-the-elipse-mobile-server-using-your-phone >. Acesso em: 20 dez. 2017.

ELIPSE MOBILE. **Elipse Mobile**. Disponível em: https://www.elipse.com.br/produto/elipse-mobile/ >. Acesso em 30 nov. 2017.

FONSECA, Marcos de Oliveira. **Comunicação OPC –** Uma abordagem prática. In: VI Seminário de Automação de Processos, Associação brasileira de metalurgia e materiais. Vitória, 2002.

FRANCHI, Claiton Moro. **Controladores lógicos programáveis**: sistemas discretos. São Paulo: Érica, 2008.

MATRIKON. **OPC Tutorials**. 2016. Disponível em: < http://www.matrikonopc.com/resources/opc-tutorials.aspx > Acesso em: 10 maio 2016.

MCROBERTS, Michael. **Arduino básico**. São Paulo:Novatec, 2011.

MICROFSOFT. **Visão geral do Firewall do Windows com Segurança Avançada**. Disponível em: < https://msdn.microsoft.com/pt-br/library/hh831365%28v=ws.11%29.aspx?f=255&MSPPError=-2147217396 >. Acesso em: 10 dez. 2017.

MICROSOFT. **Microsoft ActiveX Data Objects Reference.** 01 jul. 2013**.** Disponível em: < https://msdn.microsoft.com/pt-br/library/office/jj249010.aspx >. Acesso em: 21 maio 2018.

MORAES, Cicero Couto de; CASTRUCCI, Plínio de Lauro. **Engenharia de automação industrial**. 2. ed. Rio de Janeiro: LTC, 2007.

MOREIRA, Raquel Mendes; MAGALHÃES, Jaqueline Barbosa. **XC103 – Automação e Controle:** Teoria, Caderno de Experiências e Manual. Santa Rita do Sapucaí: Exto Tecnologia, 2010. 176 p.

OPC Foundation. **What is OPC?** 2018. Disponível em: < https://opcfoundation.org/about/what-is-opc/ >. Acesso em: 30 abril 2018.

OPC Foundation.b. **Unified Architecture**. 2018. Disponível em: < https://opcfoundation.org/about/opc-technologies/opc-ua/ >. Acesso em: 30 abr. 2018.

PARALLAX Inc. **PLX-DAQ.** 2018. Disponível em: https://www.parallax.com/downloads/plx-daq >. Acesso em: 21 maio 2018.

PRIBERAM Dicionário. **Smartphone.** Disponível em:< https://www.priberam.pt/dlpo/smartphone >. Acesso em: 20 dez. 2017.

PRUDENTE, Francesco**. Automação industrial pneumática:** teoria e aplicações. Rio de Janeiro, RJ: LTC, 2013.

PUPO, Maurício Santos. **Interface homem-máquina para supervisão de um CLP em controle de processos através da www.** Dissertação de mestrado – Universidade de São Paulo, São Carlos, 2002.

RIBEIRO, Denise Ferreira, et al. **Utilização do drive OPC na integração de sistemas de automação industrial**. III Seminário Nacional de Sistemas Industriais da Automação (SSIA). Belo Horizonte, out. 2008.

SILVA, Clodoaldo. **Normalização IEC 61131** – Parte 3: Linguagens para programação de CLP (Norma IEC 61131-3). Clube da Eletrônica – Automação e Controle, jun. 2011.

SILVEIRA, Paulo Rogério; SANTOS, Winderson E. **Automação e controle discreto**. 4. ed. São Paulo: Érica, 1998.

APÊNDICE A – Configuração da rede

Inicialmente é necessário entrar no painel de controle > Firewall do Windows > Configurações avançadas > Regras de entrada > Ações > Nova Regra. Ao entrar nesta opção é necessário escolher as seguintes especificações:

- Tipo de regra: foram dadas quatro opções: programa, porta (TCP ou UPD), pré-definida ou customizada. Foi escolhida a opção “porta”, já que esta controla a conexão TCP.
- Protocolo TCP ou UPD: foi escolhida a opção TCP, já que a comunicação entre o servidor e cliente é feita via internet.
- Aplicação da porta: haviam duas opções: todas as portas locais ou uma porta específica. A última opção foi escolhida conforme sugestão da Elipse e nesta foi inserido o endereço padrão, que no caso é o 8080.
- Tipo de permissão: permitir a conexão, permitir se é seguro ou bloquear. A opção escolhida foi a de permitir a conexão.
- Perfil de aplicação da regra: a regra poderá ser aplicada quando o computador está conectado ao seu domínio coorporativo (opção domínio), quando está conectado a uma rede privada (opção particular), e quando o computador está conectado a um local de rede pública (opção público). Estas três regras foram selecionadas.
- Nome: nome para a nova regra, como “Elipse Mobile”.

APÊNDICE B – Configuração dos equipamentos utilizados no projeto.

Essa configuração foi feita na área conexões no Elipse Mobile Server, ao clicar em nova conexão foi necessário escolher as seguintes opções. Para o arduino:

- Tipo de conexão: Arduino.
- Nome.
- Porta: nesta opção foi necessário colocar qual a porta utilizada pela conexão do arduino, verificada no software “Arduino IDE”

.

Para a conexão do Elipse E3 com o servidor, foram escolhidas as seguintes opções:

- Tipo de conexão: E3.
- Nome.
- Servidor: local.