FUNDAÇÃO OSWALDO ARANHA
CENTRO UNIVERSITÁRIO DE VOLTA REDONDA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA
TRABALHO DE CONCLUSÃO DE CURSO

BRUNO MIRANDA SABENÇA
HENRIQUE GUIMARÃES PEREIRA

APLICAÇÃO DA INDUSTRIA 4.0 NA MANUTENÇÃO PREDITIVA DE EQUIPAMENTOS CRÍTICOS DE UMA PLANTA ATRAVÉS DE TECNOLOGIA IoT (*INTERNET OF THINGS*)

VOLTA REDONDA
2018

FUNDAÇÃO OSWALDO ARANHA
CENTRO UNIVERSITÁRIO DE VOLTA REDONDA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA
TRABALHO DE CONCLUSÃO DE CURSO

APLICAÇÃO DA INDÚSTRIA 4.0 NA MANUTENÇÃO PREDITIVA DE EQUIPAMENTOS CRÍTICOS DE UMA PLANTA ATRAVÉS DE TECNOLOGIA IoT (*INTERNET OF THINGS*)

Monografia apresentada ao curso de Engenharia Elétrica do UniFOA como requisito à obtenção do título de Engenheiro Eletricista.

Discentes:
Bruno Miranda Sabença
Henrique Guimarães Pereira

Orientador:
Prof. MSc. Edson de Paula Carvalho

Coorientador:
Prof. Aloano Régio de Almeida Pereira

VOLTA REDONDA
2018

FOA Fundação Oswaldo Aranha

UniFOA Centro Universitário de Volta Redonda

FOLHA DE APROVAÇÃO

Curso: Engenharia Elétrica

Acadêmico: Matrícula: Bruno Miranda Sabença 201410097; Henrique Guimarães Pereira 201410103

Título do TCC: Aplicação da indústria 4.0 na manutenção preditiva online de equipamentos críticos de uma planta através de tecnologia iOt (internet of things).

Apresentado publicamente perante a Banca Avaliadora, como parte dos requisitos para conclusão do Curso de Engenharia Elétrica.

Aprovada em 17 de novembro de 2018

Banca Avaliadora:

....

Professor Orientador
Edson de Paula Carvalho, Mestre, UniFOA

....

Professor Avaliador
Aloano Regio de Almeida Pereira, Especialista, UniFOA

....

Professor Avaliador
Orlando Moreira Guedes Júnior, Mestre, UniFOA

A Deus, onde depositamos a nossa fé nos momentos de dificuldades, e que nos deu força e saúde para podermos chegar até aqui e realizar mais essa conquista em nossas vidas. Aos nossos pais, que nos apoiaram e ajudaram com os nossos sonhos.

AGRADECIMENTOS

Agradecemos primeiramente à DEUS, que nos ajudou a chegar até aqui e nos deu força e paciência para a realização deste sonho.

Agradecemos aos nossos pais e familiares, pela a ajuda e apoio com palavras de incentivo, às suas experiências que contribuíram com o nosso conhecimento e formação como cidadãos de caráter e ética.

Agradecemos aos nossos amigos, que também de forma indireta foram importantes para a nossa formação, em especial ao nosso amigo Ronaldo Lee Tavares que, com todo seu ensinamento, tornou esse desafio possível.

Agradecemos aos nossos orientadores, Prof. MSc. Edson de Paula Carvalho e Prof. Aloano Régio de Almeida Pereira, que foram primordiais na conclusão deste trabalho.

Agradecemos ao UniFOA, pelo conhecimento que adquirimos durante o curso.

RESUMO

Com base na tecnologia IoT, será proposto neste trabalho, um método para monitorar via tela gráfica e gerenciar através de alarmes instantâneos pré configurados e históricos diários enviados via e-mail referente aos principais dados (temperatura, umidade relativa do ar e corrente elétrica) de um inversor de frequência utilizado para controlar o motor de uma correia transportadora, a fim de ter a possibilidade de se realizar a manutenção preditiva no mesmo. Para isso, será implementado, configurado e customizado o PI System (*Plant Information*), que consiste basicamente em um conjunto de servidor/cliente desenvolvido para automatizar a coleta de dados em determinado equipamento ou até mesmo em uma planta de processos, armazenando e apresentando estas informações de forma que os gestores e os responsáveis pelo equipamento possam acompanhar o desempenho e até mesmo identificar possíveis anormalidades. Além disso, será utilizado a plataforma Arduino como tecnologia para leitura das variáveis necessárias oferecendo desenvolvimentos interativos utilizando um microcontrolador, chamado também de computação física, devido a sua interação direta entre *software* e *hardware*, permitindo a fácil configuração de sensores, atuadores e outros dispositivos. O Arduíno Uno será configurado como um ponto de rede *ethernet* via TCP/IP de um segundo controlador, o CLP Siemens, que é um dos controladores mais utilizados nas indústrias atualmente. A comunicação entre o CLP e o PI System será realizada pela tecnologia OPC, considerada uma forma padrão para troca de informações de forma segura e rápida utilizado na automação, garantindo o fluxo de dados entre diferentes fabricantes.

Palavras chaves: Arduino. PI System. Manutenção preditiva. *Internet* das coisas. Indústria 4.0.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

CLP – CONTROLADOR LÓGICO PROGRAMÁVEL
CP – COMMUNICATION PROCESSORS
CPU – CENTRAL PROCESSING UNIT
EPROM – ERASABLE PROGRAMMABLE READ-ONLY
FBD – FUNCTION BLOCK DIAGRAM
GM – GENERAL MOTORS
IoT – INTERNET OF THINGS
LAD – LADDER DIAGRAM
LED – LIGHT EMITTING DIODE
MES – MANUFACTURING EXECUTION SYSTEMS
MPI – MULTIPOINT INTERFACE
OLE – OBJECT LINKING AND EMBEDDING
OPC – OPEN PLATFORM COMMUNICATIONS
PB – PROFIBUS
PI – PLANT INFORMATION
PI AF – PLANT INFORMATION ASSET FRAMEWORK
PI UDS – PLANT INFORMATION UNIVERSAL DATA SERVER
PID - CONTROL PARAMETER ASSIGNMENT
PI-PB – PLANT INFORMATION PROCESSBOOK
PLC – PROGRAMMABLE LOGIC CONTROLLER
PS – POWER SUPPLY
PWM – PULSE WIDTH MODULATION
RAM – RANDOM ACCESS MEMORY
S7 – STEP 7
SCADA – SUPERVISORY CONTROL AND DATA ACQUISITION
SCT – SPLIT-CORE CURRENT TRANSFORMER
STL – STATEMENT LIST
VBA – VISUAL BASIC APLICATIONS

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE APÊNDICES

1 INTRODUÇÃO

O processo de transformação digital está cada vez mais associado a aplicação da tecnologia na vida do ser humano, e este tema só tende a crescer nos próximos anos. Portanto, nada melhor que associar a vida no trabalho com a tecnologia do dia a dia, a fim de cada vez mais estarmos com o controle de toda nossa rotina em nossas mãos.

A "*Internet* das coisas" (IoT) está se tornando um tópico cada vez mais crescente de conversas no local de trabalho e também fora dele. É um tema que não só tem a capacidade de afetar a forma como vivemos, mas também como trabalhamos.

O acesso à *Internet* se tornou mais acessível com o passar dos anos, o valor a ser pago está cada vez menor, mais equipamentos estão sendo desenvolvidos com diversos recursos como *Wi-Fi*, *Bluetooth* e sensores incorporados. Cada vez mais o valor destas tecnologias está mais acessível e a utilização de *smartphones* em processos industriais tem crescido de forma exponencial. Todos estes avanços tecnológicos acarretam em grandes possibilidades para o IoT.

A realidade é que o IoT permite oportunidades e conexões praticamente infinitas, muitas das quais nem sequer pensamos ou compreendemos completamente o impacto de hoje. Não é difícil entender porque o IoT é um tema tão quente hoje, certamente abre a porta para muitas oportunidades, mas também para muitos desafios.

2 JUSTIFICATIVA

Juntamente à necessidade de se manter um equipamento em pleno funcionamento em processos industriais, há também o desejo de se desenvolver formas de controle a fim de melhorar a análise de falhas e até mesmo prever possíveis danos que possam ser diagnosticados com extrema rapidez e facilidade.

Sendo assim, o monitoramento remoto de equipamentos críticos em processos industriais é uma opção extremamente vantajosa para as empresas, visto que através de alarmes pré-definidos, por exemplo, e análises bem-feitas em tempo real, é possível realizar manutenção preditiva, evitando maiores transtornos em uma possível falha que teria como consequência, paradas de produção gerando prejuízos indesejados.

A partir deste ponto e sabendo do enorme crescimento do uso do celular no nosso cotidiano, pode-se chegar à conclusão que o uso da *internet* através do *smartphone* é a melhor opção para o resultado que será buscado neste trabalho.

3 OBJETIVOS

3.1 Objetivo Geral

Com a economia globalizada nos dias de hoje e a grande concorrência no mercado, as empresas buscam cada vez mais serem competitivas e entregar um produto ao cliente cada vez melhor.

A busca por resultados positivos está diretamente ligada à redução de custo e a manutenção é um dos que contribuem para um resultado positivo ou negativo, de acordo com o planejado.

Ter um bom gerenciamento na manutenção faz a diferença em seu resultado, e é um fator importante para a contribuição da produtividade, disponibilidade operacional, segurança de pessoas e ambiental.

Portanto, buscar alternativas eficazes e inovadoras será o foco deste trabalho, a fim de inovar a gestão da manutenção.

3.2 Objetivo Específico

O trabalho apresentado tem como finalidade demonstrar estudos que tenham o foco em atender as necessidades das indústrias, de forma que as manutenções sejam melhores gerenciadas. Com o objetivo de minimizar a manutenção corretiva será apresentado uma forma de modernização da manutenção preditiva, podendo ser realizada de forma remota através de mensagens via e-mail e telas gráficas acessíveis aos gestores no local de trabalho. Serão abordados neste trabalho, estudos que busquem viabilidade técnica para as indústrias, auxiliando a manutenção em sua gestão, onde a manutenção preditiva de forma online, fará com que as decisões sejam assertivas.

4 REVISÃO BIBLIOGRÁFICA

4.1 Manutenção

O conceito de manutenção com o passar dos anos vem evoluindo assim como a indústria. Estudos foram realizados de modo que a manutenção atendesse a necessidade das indústrias, garantindo o processo contínuo sem paradas imprevistas e consequentemente o custo para a realização da intervenção. Até pouco tempo, a manutenção era entendida como: “manter o equipamento em suas condições originais/fabricação”. Com base nos estudos e nas progressões das indústrias e da manutenção, o entendimento sobre a mesma também evoluiu, na qual visa atualmente: “Garantir a confiabilidade operacional, produtividade, evitar paradas imprevistas, redução de custo, ser assertivo evitando retrabalho e principalmente, segurança e meio ambiente” (GURSKI, 2002).

Segundo (OTANI e MACHADO, 2008), a manutenção surgiu antes da Segunda Guerra Mundial, as indústrias não tinham produções em grandes escalas, não se preocupavam com produtividade e nem tinham muitos equipamentos que necessitassem de intervenção, pois todo trabalho era “braçal”, o pouco de equipamento que fazia parte do processo produtivo, era todo mecânico e superdimensionado. Os serviços que ocorriam esporadicamente nos equipamentos, eram atividades simples, consistia basicamente em realizações de limpezas, lubrificações e reparos ou substituição de componentes quando quebravam. Nesse período, antes da Segunda Guerra Mundial, a manutenção se resume basicamente em “corretiva”.

Entre o período da Segunda Guerra Mundial e até os anos 60, onde as indústrias começaram a aumentar a sua produção em larga escala, ocorreu um aumento por diversos produtos nessa época. Isso fez com que as indústrias tivessem maior tempo de produção e disponibilidade operacional, aumentando o quantitativo de máquinas mecânicas e complexas. Com esse aumento de equipamentos, as fábricas se tornaram cada vez mais dependentes delas, a necessidade de intervenções também acompanhou essa evolução e devido à complexidade das máquinas e a necessidade de confiabilidade operacional, fez com que surgisse a manutenção preventiva (OTANI e MACHADO, 2008).

A manutenção preventiva, era realizada regularmente, com intervalos fixos, sem uma base de parâmetros ou dados, que direcionasse o período correto para a intervenção.

Os custos das indústrias aumentaram muito, comparando com outros custos operacionais. Foi criado um controle sistêmico de manutenção e planejamento, para que as preventivas fossem realizadas e o custo fosse reduzido. Atualmente esse conceito faz parte de diversas indústrias (MARCORIN e CAMELO LIMA, 2003).

Depois da década de 70 o processo produtivo disparou, as demandas de produção duplicaram, as indústrias se preocupavam em atender as solicitações do mercado, de forma eficaz e com qualidade. Porém, as paradas indesejadas, afetavam o andamento do ritmo produtivo, com aumento de custos e perda de clientes (MARCORIN e CAMELO LIMA, 2003).

A evolução da automação, junto com a modernização da mecanização, foram pontos chaves para indicar o quão a manutenção é importante para a indústria, em termos de confiabilidade e disponibilidade. Com isso passa a surgir à manutenção preditiva.

4.1.1 Manutenção Preditiva

Segundo (GURSKI, 2002), a Manutenção Preditiva atua sobre a variação de um parâmetro como base. O objetivo é predizer, qualquer anomalia antes que aconteça a falha, de forma a garantir a produção por um longo período, sem paradas imprevistas. A preditiva realiza análise dos parâmetros de diversos equipamentos, dos mais variados tipos de medições, com os equipamentos de uma planta de um processo produtivo em pleno funcionamento. Podemos citar alguns exemplos de medições, que podem ser realizadas:

- Corrente elétrica;
- Tensão elétrica;
- Potência elétrica (ativa, aparente e reativa);
- Vibração do motor elétrico;
- Temperatura dos motores e mancais;
- Temperatura de painéis elétricos (Termografia);
- Umidade do ar.

Com isso, relatórios para análise podem ser criados e direcionados aos responsáveis pelos equipamentos, auxiliando na gestão da manutenção. As manutenções podem ser realizadas de forma programada e o custo com aquisição de sobressalentes serão menores, reduzindo a necessidade de estoque de peças (GURSKI, 2002).

Esse tipo de manutenção é feito por acompanhamento, na qual traz-se à necessidade de mão-de-obra qualificada e maiores investimentos com equipamentos e softwares sofisticados, para esse tipo de atividade. Seu alto custo de investimento é rapidamente recompensado quando comparado com a quebra de equipamentos, paradas/interrupções de linhas de processo (perda de produção), segurança de pessoas e meio ambiente (GURSKI, 2002).

Empresas líderes do mercado ou de sucesso, vem adotando tais práticas em suas organizações, considerando a preditiva uma estratégia eficaz em seu processo produtivo. Conhecer a necessidade de cada organização, otimizando a sua manutenção é a nova tendência mundial, fazendo com que as empresas sejam assertivas em suas decisões, possibilidade de expansão no mercado e maior lucratividade, garantindo a sua sobrevivência (OTANI e MACHADO, 2008).

Segundo (SSPREDITIVA, 2018), a manutenção preditiva é aquela capaz de indicar as condições reais de funcionamento das máquinas com base em dados que informam o seu desgaste ou processo de degradação com o decorrer do tempo. É a manutenção que prediz o tempo de vida útil dos componentes das máquinas e equipamentos. A manutenção preditiva é reconhecida como uma técnica eficaz de gerenciamento de manutenção, sendo um processo que prediz o tempo de vida útil dos equipamentos bem como as condições para que esse tempo seja bem aproveitado, observe na figura 1.

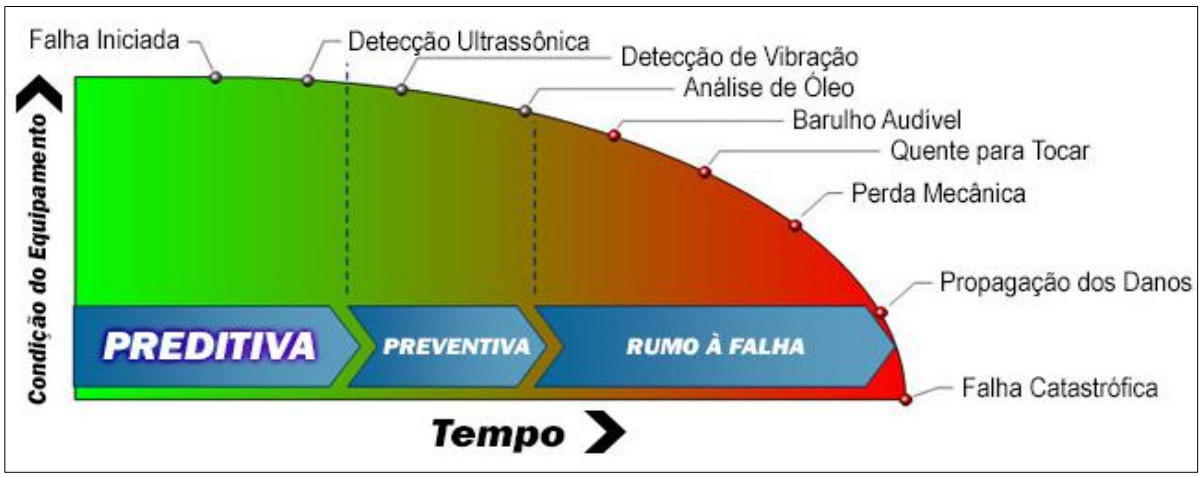


Figura 1 – Manutenção Preditiva
Fonte: SSPreditiva (2018)

Esse cenário precisa ser alterado de modo que a manutenção preditiva suba rapidamente. É necessário que diminuam as manutenções corretivas e preventivas, para que a preditiva seja um potencial de ganho para as indústrias, trazendo melhores resultados. A manutenção preditiva será o caminho para a excelência operacional (ABRAMAN, 2013).

4.1.1.1 Consequências devido à falta de manutenção preditiva

Segundo (MMTEC, 2018), não adotar as técnicas de análise de manutenção preditiva poderá resultar em:

- Baixa qualidade de produção: Inspeções frequentes em equipamentos, está diretamente relacionada a uma manutenção não eficiente, o que eleva o custo do controle de qualidade a médio prazo, resultando também numa queda da qualidade do processo;
- Queda da produtividade: Quando não se adota a manutenção preditiva, o ritmo de produção é afetado mesmo que o equipamento não sinalize algum defeito. Essa situação acontece porque um simples desgaste natural dos equipamentos pode elevar o tempo de produção e reduzir o desempenho deles, e quando surgir algum defeito mais evidente, haverá um custo com recursos, com sobressalentes, mão de obra e horas extras de colaboradores. Essa queda na produtividade se torna uma desvantagem

em um mercado cada vez mais competitivo e até mesmo a imagem da empresa pode ser afetada;

- Disponibilidade: Redução da capacidade de processo devido à ausência de manutenção preditiva. Está diretamente associado à confiabilidade e manutenibilidade dos equipamentos.

4.2 Industria 4.0

A evolução industrial começou com mecanização da produção, junto com a máquina a vapor, logo veio a eletricidade e a produção em larga escala de linhas de montagem de Henry Ford. Foram revoluções industriais impulsionadas por modernizações tecnológicas, onde as mudanças alcançaram as estruturas socioeconômicas e culturais a nível mundial e que se perpetuaram por séculos a primeira e por décadas a segunda (SIEMENS, 2017)

Nos anos 70 surge uma nova era industrial com o desenvolvimento de tecnologias, como: computadores, fibra óptica, microeletrônica, telecomunicações, nuclear e agricultura biogenética e biológica. Permitiu ao mundo avançar exponencialmente. Desde o início do século XXI, o mundo se deparou com uma evolução nos setores econômicos e da sociedade nunca antes vistas. Os estudos de especialistas, apontam que estas mudanças são apenas 1/10 daquilo que ainda está por surgir (SIEMENS, 2017).

Conforme os níveis de produção no mundo cresceram, os países desenvolvidos melhoraram o rendimento e a exigência das populações também aumentaram. O ritmo de vida da população está cada vez mais acelerado, graças as inovações tecnológicas, com isso, as empresas tiveram que acompanhar as exigências dos consumidores num ambiente cada vez mais competitivo (SIEMENS, 2017).

Com isso foram criadas as condições tecnológicas, visando as inovações nunca antes possíveis e que contribuíram para o surgimento dos sistemas ciberfísicos, a *internet* das coisas e a *internet* dos serviços. Assim começa a 4ª Revolução Industrial conhecida também como Indústria 4.0 (SIEMENS, 2017). Observe na figura 2 o resumo sobre a revolução industrial.

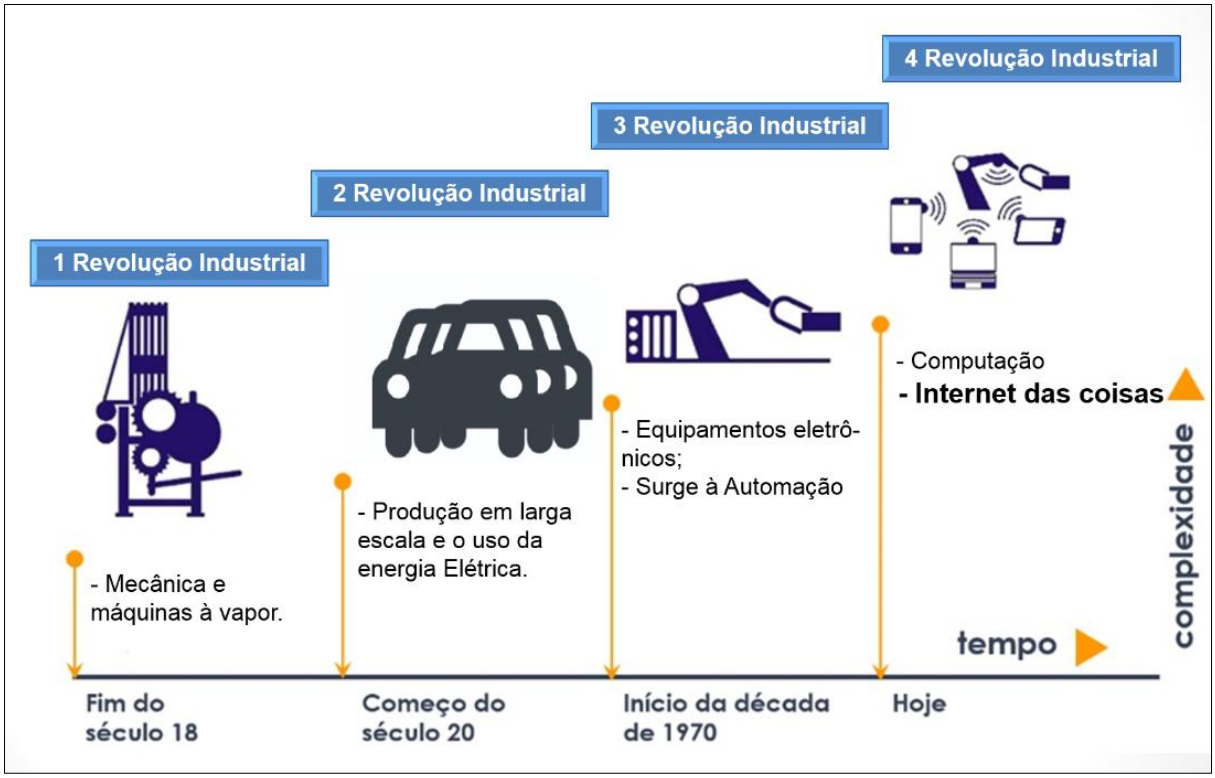


Figura 2 – Evolução Industrial
Fonte: Kagermann, Wahlster e Helbig (2013)

4.3 Internet das coisas (IoT)

Segundo (DINIZ, 2006), a *Internet* das Coisas é entendida como a conexão entre a vida rotineira do ser humano com a *internet*. Agindo de forma inteligente, conectando-se o mundo real com o mundo digital, fazendo com que as pessoas possam estar em constante comunicação com outras pessoas ou objetos. Com a evolução tecnológica no mundo, a necessidade de conectividade entre pessoas e máquinas aumentaram muito, os eletroeletrônicos evoluíram exponencialmente. Atualmente, diversos dispositivos podem se conectar à *internet*, surgindo assim o termo "*Internet* das coisas", devido a capacidade de um dispositivo se conectar à *internet*, permitindo a comunicação com outros dispositivos.

Hoje o IoT vem se tornando um assunto cada vez mais crescente, dentro das empresas em locais de trabalho, como no nosso dia a dia, tendo um enorme potencial de impactar a nossa vida, como também a forma como trabalhamos (DEIDMAR, SOBREIRA e LIMA, 2017).

O IoT realiza troca de dados de forma online, tem a função de fazer rastreamento, gerenciar e controlar qualquer dispositivo, tudo isso sendo feito a distância, através da tecnologia *mobile* (DEIDMAR, SOBREIRA e LIMA, 2017).

É quase impossível mensurar a sua capacidade de controle e monitoramento, mas podemos citar algumas, como: eletrodomésticos (televisão, máquina de lavar, liquidificador, ar condicionado e etc.), carros, câmeras de segurança, radares e entre outros, interligados na computação (DEIDMAR, SOBREIRA e LIMA, 2017).

Segundo (EVANS, 2011), na figura 3 é possível observar que estamos vivenciando uma nova revolução com a *internet*, onde as “coisas”, são controladas, monitoradas e distribuídas em tempo real e em qualquer ambiente.

Figura 3 – *Internet* das coisas
Fonte: Reis (2016)

4.3.1 Tecnologia *Mobile*

São conhecidos pela sua característica de portabilidade, podem ser transportados para qualquer lugar e ambiente, sua facilidade de manuseio é graças ao seu peso, que o torna cômodo no transporte. Os mais populares são: notebook,

tablets, agenda eletrônica, telefones celulares e smartphones (telefones com sistemas operacionais*).* Estes aparelhos permitem conexão e comunicação, através de redes sem fio, redes de telefonia celular, *bluetooth*, entre outras. A cada dia essa tecnologia vem se tornando cada vez mais útil e popular nos mais diversos tipos de usos e aplicações, devido a sua facilidade e simplicidade de utilização. As previsões de crescimento da base de objetos conectados contêm números astronômicos (EVANS, 2011).

A analista Gartner diz que, até 2020, haverá mais de 26 bilhões de dispositivos conectados, mas a projeção é de 50 bilhões. Isso significa muitas conexões - alguns até estimam que esse número seja muito maior, mais de 100 bilhões (TERRA, 2017).

O número elevado de smartphone e tablets conectados à *internet* era de 12,5 bilhões em 2010. Segundo estudos, estima-se que o IoT nasceu entre 2008 e 2009, conforme observado na figura 4 (EVANS, 2011).

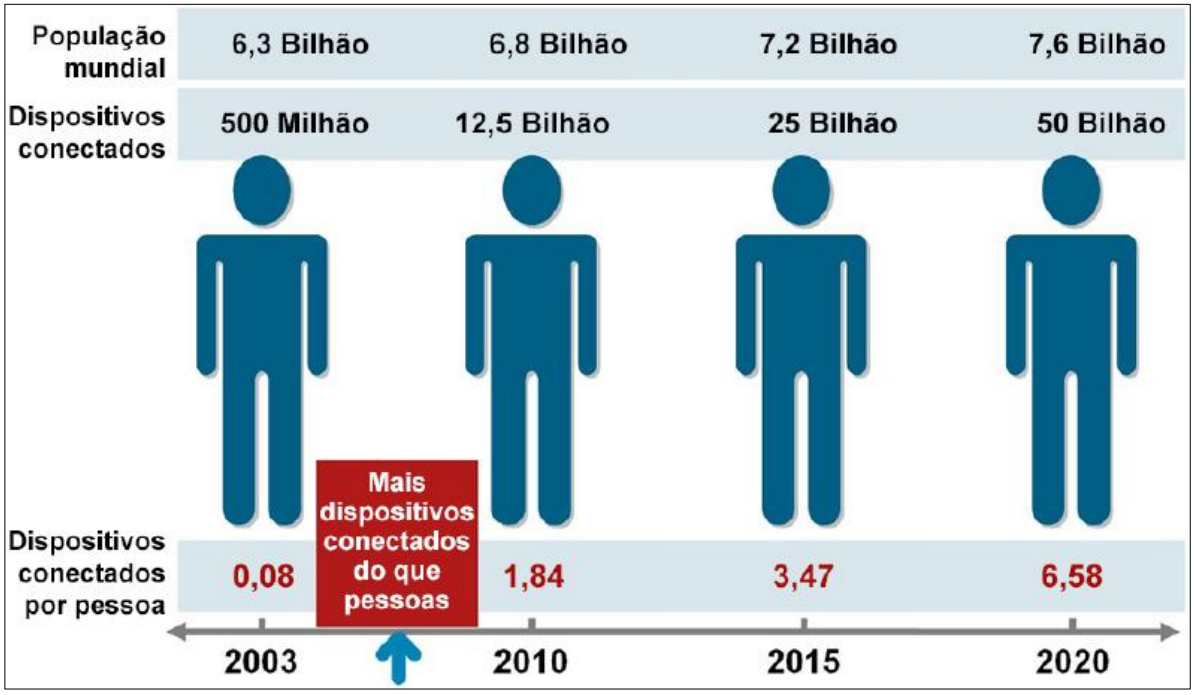


Figura 4 – A *internet* das coisas nasceu entre 2008 e 2009
Fonte: Evans (2011)

4.4 Microcontrolador

Em um projeto eletrônico que tenha o objetivo de controlar determinada função de forma autônoma, se faz necessário a utilização de um dispositivo que possua esta opção. Os microcontroladores, por possuírem ótimo custo benefício são bastante utilizados nessa situação (COSTA, SERMANN e SILVA, 2016).

Para que um microcontrolador seja capaz de executar todas as funções desejadas é necessário que o mesmo tenha alguns recursos disponíveis. Tais recursos são instalados de forma integrada, o que possibilita que o mesmo seja compacto. Algum destes recursos são: memória, processador, portas de entrada e saída e outros (COSTA, SERMANN e SILVA, 2016).

Atualmente existem vários tipos de microcontroladores com diversas funcionalidades. Cada um destes com especificações diferentes como: velocidade de processamento, tipo e quantidade de portas de entrada e saída, interfaces de comunicação, memória de armazenamento de dados e outras coisas (MARCHESAN, 2012).

Com base na aplicação deste projeto foi decidido utilizar a plataforma Arduino, pela facilidade de programação e pela gama de oportunidades que este dispositivo oferece, simplificando a utilização de microcontroladores (CARVALHO, 2011).

4.4.1 Arduino

O Arduino é uma plataforma eletrônica de código aberto *(open source)* que possibilita a facilidade na programação e interface de hardware em diversas funções de controles e acionamentos, seja na automação residencial ou industrial (CARVALHO, GOMES e SILVA, 2017).

Em 2005, o professor Massimo Banzi desenvolveu essa plataforma a fim de ensinar programação a seus alunos. Como os equipamentos necessários para uma aula prática eram caros e complexos, foi necessário desenvolver um dispositivo que fosse de fácil explicação e de baixo custo (CARVALHO, GOMES e SILVA, 2017).

Ainda segundo (CARVALHO, GOMES e SILVA, 2017), a partir do ponta pé inicial dado por Banzi e por se tratar de uma plataforma de código aberto, diversos desenvolvedores do mundo todo aperfeiçoaram o Arduino criando placas de

interfaces e até mesmo programações já pré configuradas, as chamadas bibliotecas, que possibilitam ao usuário utilizar programações complexas desenvolvidas por pessoas de alto grau de conhecimento.

Para que a programação tenha alguma função, além do *software* é necessário a utilização da placa Arduino que possui diferente tipos de entradas e saídas possibilitando a interface com o meio externo.

Neste projeto foi utilizado a placa Arduino Uno que possui todas as características necessárias para o protótipo a ser apresentado.

4.4.1.1 Placa Arduino UNO

A placa Arduino UNO possui como componente principal, o microcontrolador ATMEL ATMEGA328, um hardware de 8 bits da família AVR. É uma ferramenta simples e possui um hardware mínimo, com diversas características interessantes para projetos. Os componentes e especificações técnicas gerais dessa placa são dados a partir da figura 5 (SOUZA, 2013a).

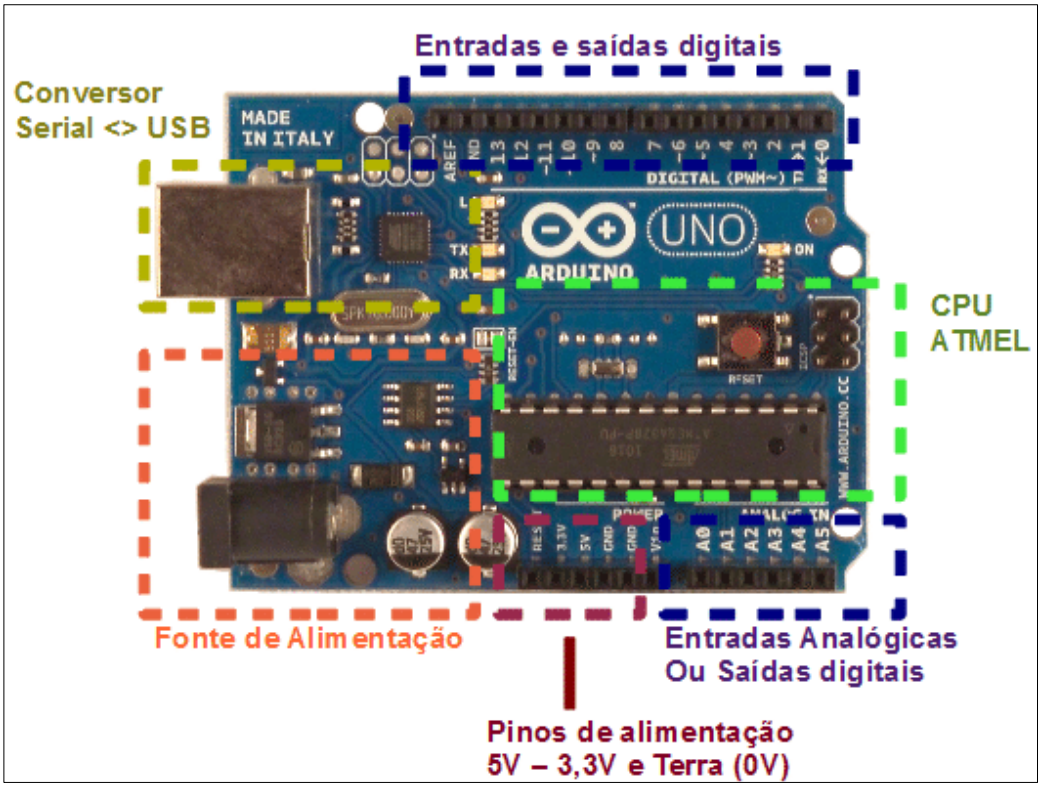


Figura 5 – Resumo da placa Arduino UNO
Fonte: Júnior, Barbosa e Júnior (2018)

Especificações técnicas da placa:

- Microcontrolador: ATmega 328
- Tensão de alimentação recomendada: 7-12V
- Pinos de entrada/saída digitais: 14 (6 podem ser usados como PWM)
- Pinos de entrada analógica: 6
- Tensão de entrada/saída: 5V

Segundo (SOUZA, 2013a), a alimentação da placa pode ser feita tanto através de conexão USB como também por meio de uma fonte externa através de um conector Jack, como pode-se ver na figura 6.

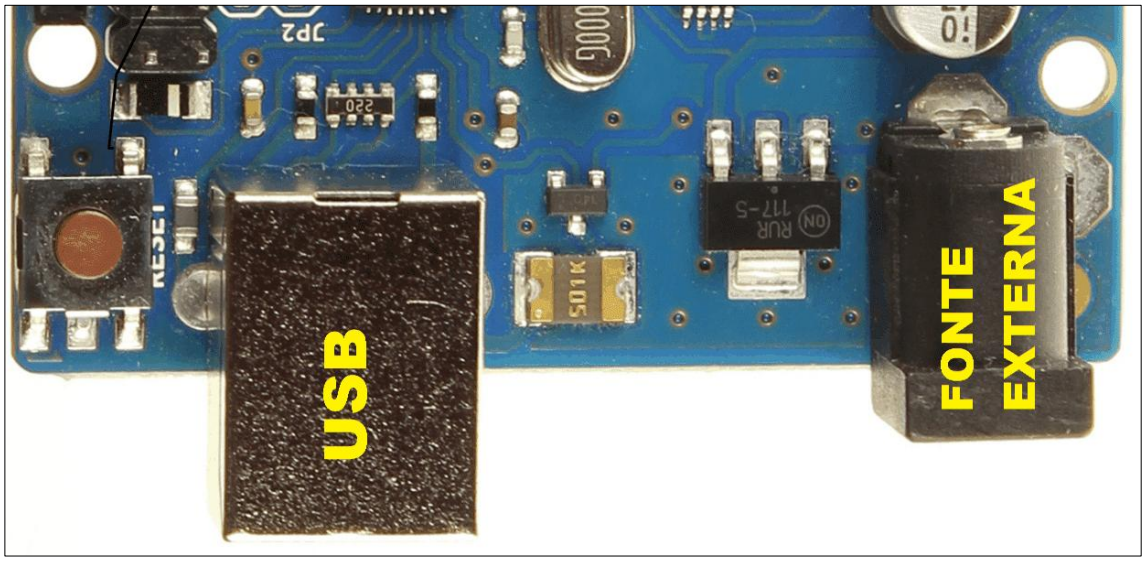


Figura 6 – Alimentação da placa Arduino UNO
Fonte: Souza (2013)

Os limites de tensão para alimentação externa, é de 6V a 20V, porém se a placa for alimentada com uma tensão abaixo de 7V, a tensão de funcionamento geral da placa (5V) pode ficar instável e se for alimentada com uma tensão acima de 12V, a placa pode sobreaquecer e danificar componentes internos. Portanto a tensão recomendada para alimentação é de 7-12V (SOUZA, 2013a)

A figura 7 mostra alguns dos conectores de alimentação utilizados para conexão de *shields*, módulos ou qualquer outro dispositivo externo à placa Arduino.

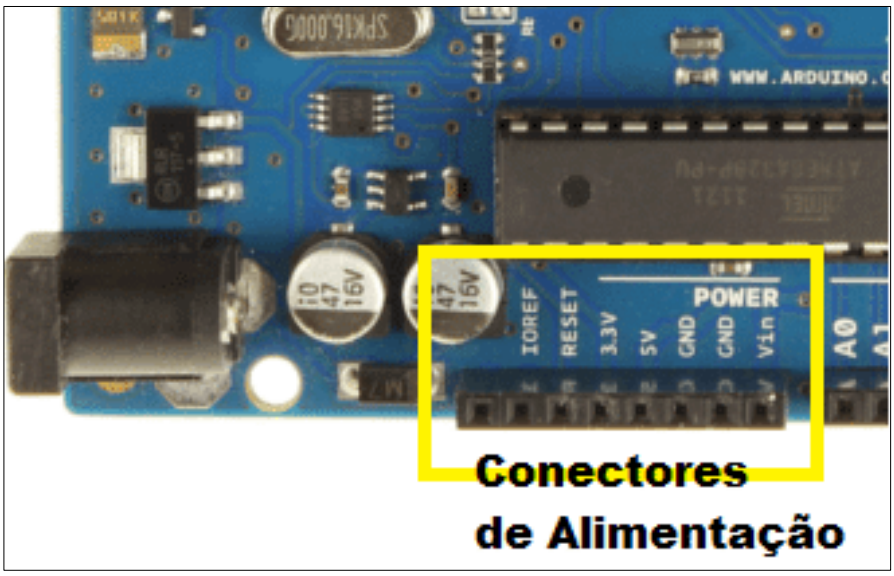


Figura 7 – Conectores de alimentação da placa Arduino
Fonte: Souza (2013)

- IOREF – Atravéz desta porta é fornecida uma tensão de referência para que determinados *shields* possam se adaptar para ter uma interface de tensão apropriada;
- RESET – Este pino está conectado ao *reset* do microcontrolador, portanto, através dele, é possível se obter um *reset* externo da placa, reiniciando o Arduino;
- 3,3V – Fornece uma tensão de 3,3V para dispositivos externos. Com uma corrente máxima de 50mA;
- 5V – Fornece uma tensão de 5V para dispositivos externos;
- GND – Pino de referência;
- VIN – Pino utilizado para alimentar a placa Arduino através de dispositivos externos. Caso o conector Jack esteja sendo usado para alimentar a placa, a tensão da fonte conectada estará no pino VIN.

A placa Arduino UNO possui um total de 14 pinos para entrada e saída digital, sendo que 6 destes podem ser usados como saída PWM (pinos 3, 5, 6, 9, 10 e 11). Estes pinos operam com uma tensão de 5V e podem fornecer uma corrente de até 40mA (CARVALHO, GOMES e SILVA, 2017).

Para a interface analógica, a placa dispõe de 6 pinos, sendo que cada um deste tem a resolução de 10 bits, operando em uma faixa de tensão de 0 a 5V. Na figura 8 pode-se visualizar os pinos de entrada e saída (CARVALHO, GOMES e SILVA, 2017).

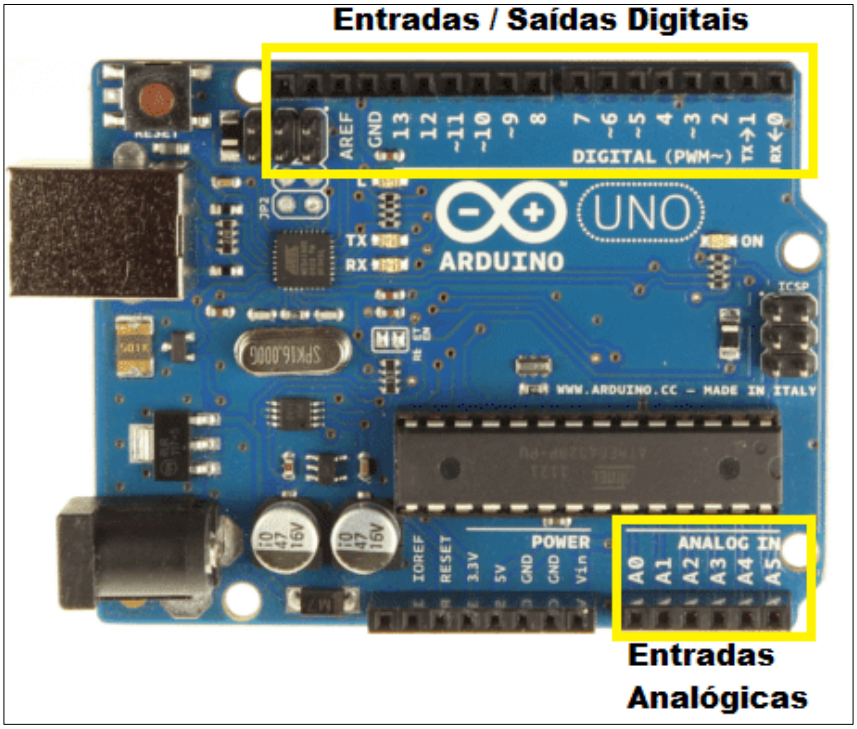


Figura 8 – Entradas e Saídas da placa Arduino UNO
Fonte: Souza (2013)

A comunicação da placa com o computador ocorre através de uma interface estabelecida por meio de outro microcontrolador, o ATMEGA16U2 representado na figura 9 (SOUZA, 2013a).

Figura 9 – Microcontrolador da porta USB da placa Arduino UNO
Fonte: Souza (2013)

4.4.1.2 *Software* de programação do Arduino

Toda a programação é feita através de um *software* gratuito obtido através do próprio site desta plataforma. Na figura 10 é possível visualizar a tela inicial do *software*.

O aplicativo utilizado é o Arduino Integrated Development Environment (*IDE*) ou Ambiente de Desenvolvimento Integrado. Este possibilita que o usuário crie a programação de diferentes projetos de forma simples e didática. Estas programações são chamadas *sketches* e são escritas na forma de texto.

A linguagem utilizada na programação do Arduino é C/C++, que possibilita certa facilidade no controle e tomadas de decisão das entradas e saídas.

A estrutura da programação consiste basicamente em duas etapas: *Setup* e *Loop* (CARVALHO, GOMES e SILVA, 2017).

- *Setup* – Esta etapa é executada somente uma vez, quando o programa é iniciado. Nela é executada toda a preparação da programação, determinando o comportamento e configuração dos pinos da placa e ativando a porta serial;
- *Loop* – Esta etapa é executada após o *setup.* As funções presentes no *loop* serão repetidas e executadas constantemente. É aqui que se concentra a parte principal da programação, interpretação das entradas e tomadas de decisão.

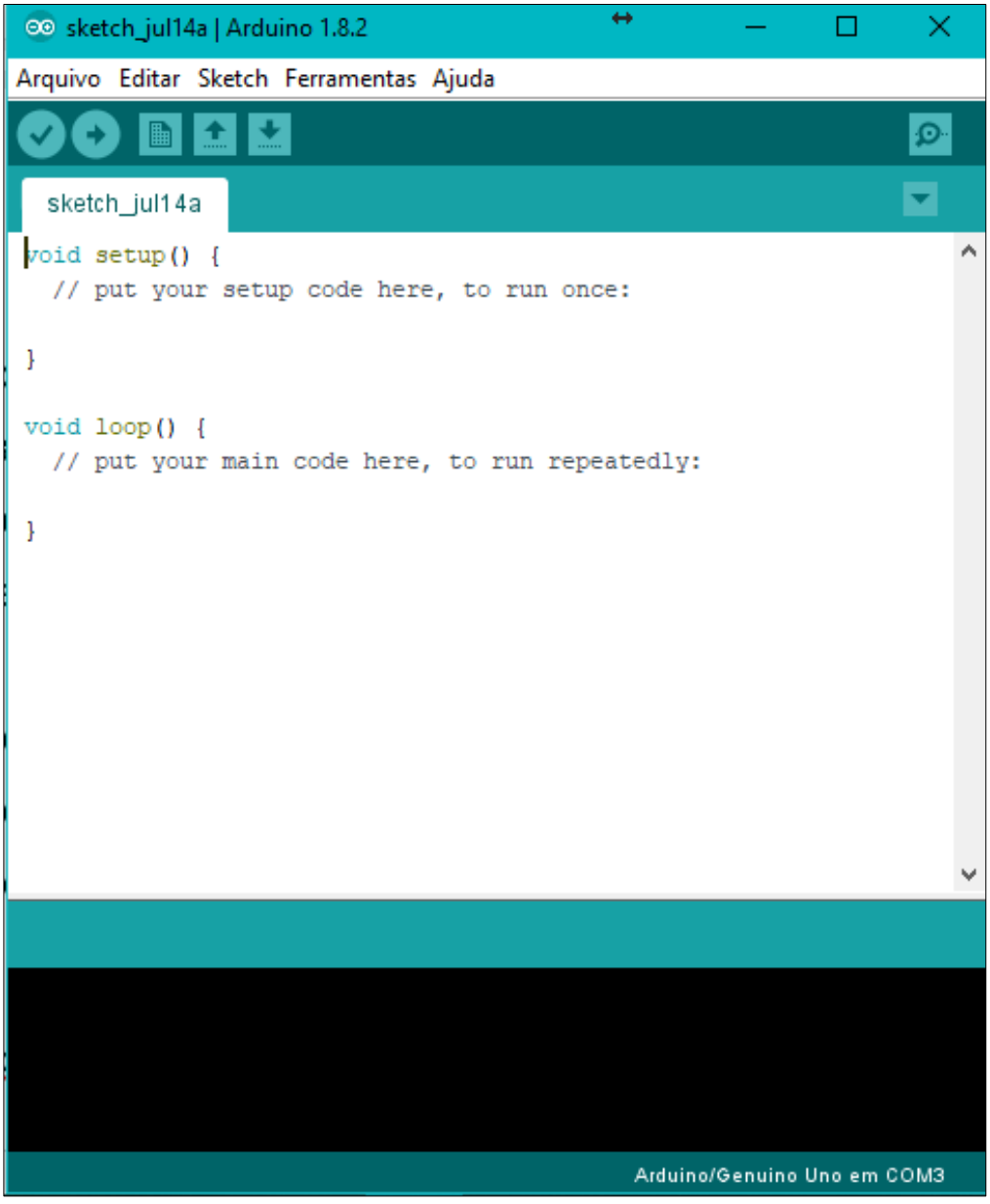


Figura 10 – Tela inicial do *software* de programação Arduino
Fonte: Autores (2018)

4.4.1.3 Bibliotecas de programação

Uma biblioteca pode ser definida como um conjunto de códigos pré configurados, que torna bem mais simples, uma programação que seria muito complexa, pois grande parte do código fica oculta e somente a parte necessária de configuração e entrada de dados está disponível ao usuário (MCROBERTS, 2011).

Neste projeto foram utilizadas as seguintes bibliotecas:

- Ethernet – Responsável pela configuração e controle do módulo Ethernet;

- Settimino – Responsável pela parte de comunicação do Arduino com o CLP Siemens;
- Dht – Responsável pelo controle do sensor DHT22;
- Emonlib – Responsável pelo controle do sensor de corrente SC013.

4.4.1.4 Módulo Ethernet Shield W5100

Segundo (LEMOS, 2014), através do módulo Arduino Ethernet Shield W5100, é possível estabelecer uma conexão de rede *ethernet* entre dispositivos. O chip W5100 permite a configuração de um IP definido pelo usuário, permitindo assim que se estabeleça uma configuração TCP/IP. Este *shield* se conecta a placa Arduino através de diversos pinos e soquetes permitindo uma conexão segura e isenta de cabos.

A conexão do módulo Ethernet com a rede é estabelecida através de um cabo de rede, onde uma de suas extremidades é ligada à porta RJ-45 do *shield* e a outra é ligada à um computador ou roteador (TÓFOLI, 2014).

Na figura 11 Podemos visualizar a placa utilizada neste projeto e que possui os seguintes *LEDs* informativos:

- PWR – Indica que a placa está sendo alimentada com energia;
- LINK – Pisca quando há troca de dados (recebe ou envia) e informa a presença de um link de rede;
- COLL – Quando há uma colisão de rede, este LED pisca;
- FULLD – Indica recepção e transmissão de dados nos dois sentidos simultaneamente;
- TX – Pisca quando há envio de dados;
- RX – Pisca quando há recebimento de dados;
- 100M – Pisca quando é detectado uma conexão de 100 Mb/s.

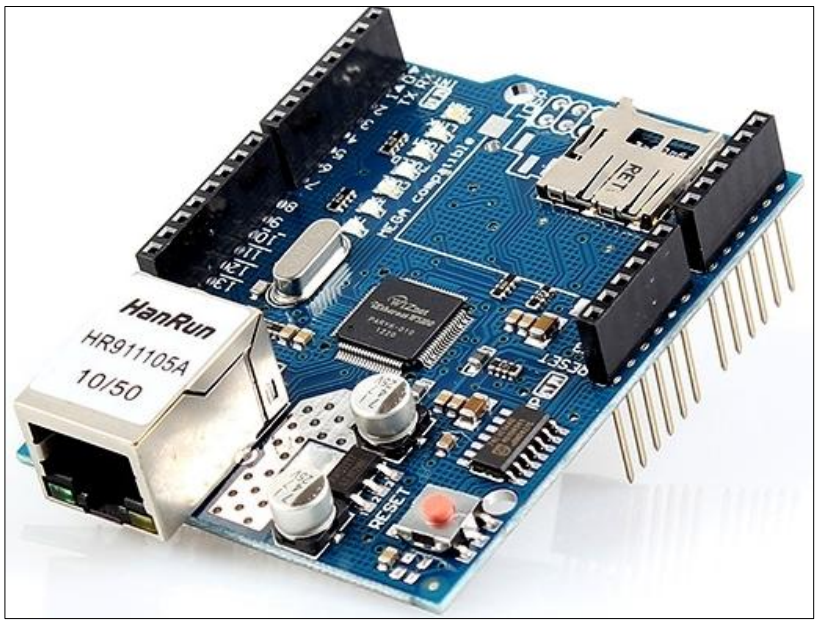


Figura 11 – Módulo Ethernet Shield W5100
Fonte: Flipflop (2014)

4.5 Sensor de corrente elétrica

Neste projeto será monitorado a corrente elétrica de saída de um inversor de frequência, sendo que o motor a ser controlado possui uma corrente nominal de 33A. Portando, será utilizado um sensor de corrente que atenda a essa faixa de operação e possibilite medições com níveis de sobrecarga, permitindo o gerenciamento de alarmes e uma manutenção preditiva eficaz.

Por questões de segurança devido ao nível elevado de corrente elétrica, não será possível abrir o circuito para medição, sendo necessário a utilização de um TC (transformador de corrente) que utiliza as propriedades magnéticas da corrente elétrica (DEMETRAS, 2017).

4.5.1 Lei de Ampere

Segundo (MOREIRA e PINTO, 2003), a Lei de Ampere afirma que uma corrente elétrica que percorre um condutor, induz um campo magnético proporcional à corrente permitindo que este campo magnético seja utilizado para medições e cálculos

indiretos, isto é demonstrado na figura 12. A equação que relaciona corrente elétrica (I) com o campo magnético (B) pode ser descrita da seguinte forma:

$$B = \frac{\mu * I}{2\pi * R}$$

Sendo que μ é a constante de permeabilidade magnética tendo o seguinte valor definido:

$$\mu = 4\pi * 10^{-7}\ T*m/A$$

E o valor de R é igual a distância do ponto a ser medido o campo magnético até o condutor por onde passa a corrente elétrica (I).

Figura 12 – Demonstração da Lei de Ampere
Fonte: Costa (2017)

4.5.2 Lei de Faraday

Segundo (SILVEIRA e VARRIALE, 2009), ao variar o campo elétrico ao longo do tempo no interior de uma espira, é gerado uma força eletromotriz acarretando em uma corrente elétrica induzida proporcional ao campo magnético, definindo assim a Lei de Faraday. Na figura 13 é demonstrado tal afirmação.

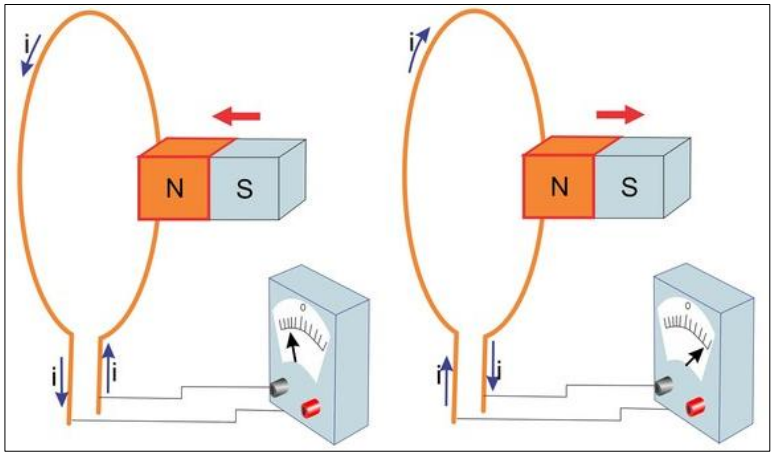


Figura 13 – Demonstração da Lei de Faraday
Fonte: Santos (2018)

4.5.3 Corrente alternada

Representada por uma função senoidal, figura 14, a corrente alternada varia de forma regular ao longo do tempo. Essa variação ocorre devido à oscilação da tensão entre valores negativos e positivos, alternando a polaridade do circuito fazendo com que o sentido da corrente mude. Essa frequência de oscilação é geralmente de 50 Hz ou 60 Hz, sendo que a segunda é adotada no Brasil. Esta frequência também pode ser controlada por inversores de frequência (SILVA, 2018).

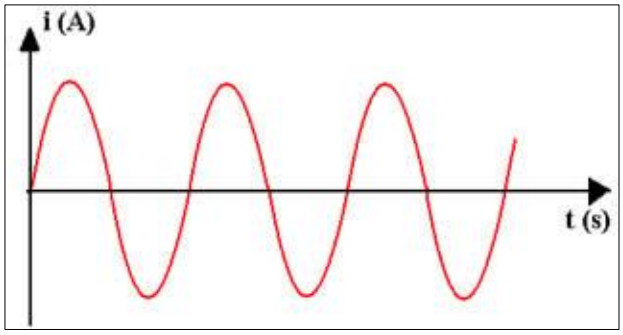


Figura 14 – Corrente alternada
Fonte: Silva (2018)

Segundo (HALLIDAY, RESNICK e KRANE, 2004), a corrente alternada pode ser representada pela seguinte fórmula:

$$i = i_m * sen(\omega t - \theta),$$

onde i_m é a máxima intensidade da corrente e θ é o ângulo de fase da relação entre a força eletromotriz e a corrente elétrica.

4.5.4 Transformador de corrente SCT-013

O sensor de corrente elétrica utilizado neste estudo é o SCT-013, figura 15, que baseado nas definições descritas anteriormente, permite a medição de níveis de corrente alternada de até 100A sem que haja contato elétrico com o circuito, necessitando apenas que o mesmo esteja em torno do cabo de uma das fases de saída do inversor de frequência.

Figura 15 – Sensor SCT-013-00
Fonte: Autores (2018)

Este TC é composto de um conjunto de espiras que estão posicionadas ao redor de um condutor chamado núcleo, ao qual passa um fluxo magnético induzido (Ib) pelo campo magnético gerado pela corrente elétrica do condutor (Ia) do equipamento principal a ser monitorado. Portanto, o TC terá em seus terminais de saída, uma corrente alternada induzida (Ic) proporcional à corrente (Ia) que passa pelo

condutor do equipamento. Na figura 16 podemos visualizar estas correntes (DEMETRAS, 2017).

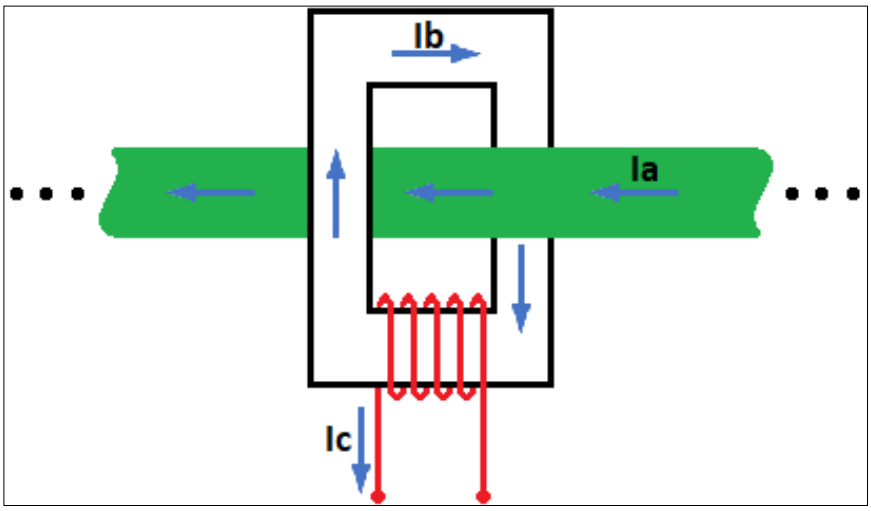


Figura 16 – Transformador de corrente
Fonte: Autores (2018)

Os transformadores de corrente podem ser classificados quanto ao modelo do núcleo, podendo ser de núcleo dividido (*split-core*) ou núcleo sólido (*solid-core*). O sensor SCT-013 utilizado neste projeto possui o núcleo divido, como visto nas figuras 17 e 18 podendo ser aberto para “abraçar” o cabo por onde passa a corrente a ser medida, sendo então um sensor não invasivo (DEMETRAS, 2017).

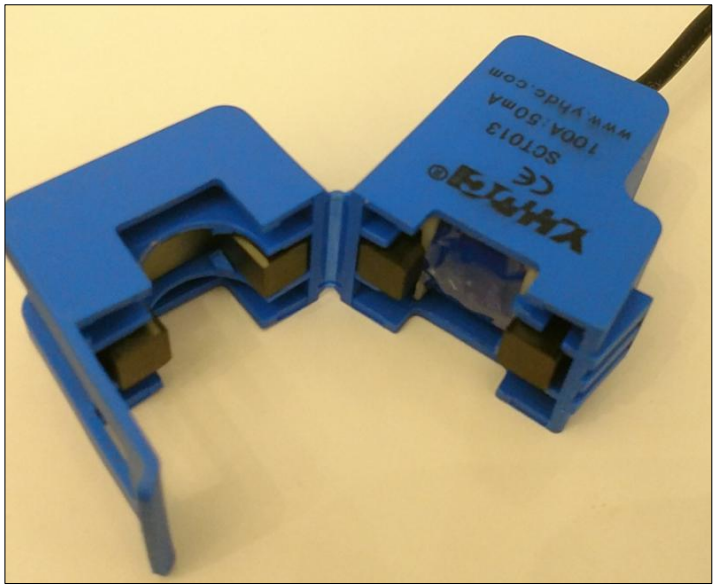


Figura 17 – Transformador de corrente com núcleo dividido
Fonte: Autores (2018)

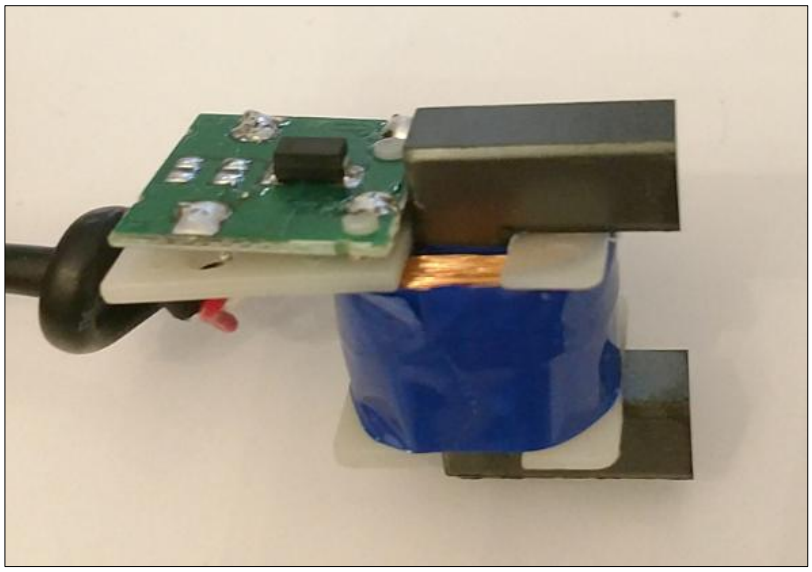


Figura 18 – Transformador de corrente visto por dentro
Fonte: Autores (2018)

Na tabela 1 é possível visualizar todos os modelos do sensor SCT-013. O modelo utilizado neste estudo foi o SCT-013-00 que possibilita a medição de corrente até 100A gerando um sinal de saída de 0 a 50mA.

Tabela 1 – Modelos do sensor SCT-013

Modelo	SCT-013-000	SCT-013-005	SCT-013-010	SCT-013-015	SCT-013-020
Corrente medida	0-100A	0-5A	0-10A	0-15A	0-20A
Sinal de saída	0-50mA	0-1V	0-1V	0-1V	0-1V
Modelo	SCT-013-025	SCT-013-030	SCT-013-050	SCT-013-060	SCT-013-000V
Corrente medida	0-25A	0-30A	0-50A	0-60A	0-100A
Sinal de saída	0-1V	0-1V	0-1V	0-1V	0-1V

Fonte: RHY (2018)

4.6 Sensor de temperatura e umidade DHT22

Neste estudo foi utilizado o sensor DHT22, figura 19, também conhecido como AM2302, que através de um termistor e um sensor capacitivo, é capaz de medir temperatura e umidade respectivamente em um único módulo. Os dados do sensor são enviados por meio de um controlador que envia um sinal serial através de um dos pinos do sensor. O DHT22 pode ser alimentado com uma tensão contínua entre 3.3V e 6V e possibilita uma medição de temperatura entre -40°C e 125°C com ± 0,5°C de

precisão e valores umidade entre 0% e 100% com precisão que varia de 2% a 5% (MOTA, 2018).

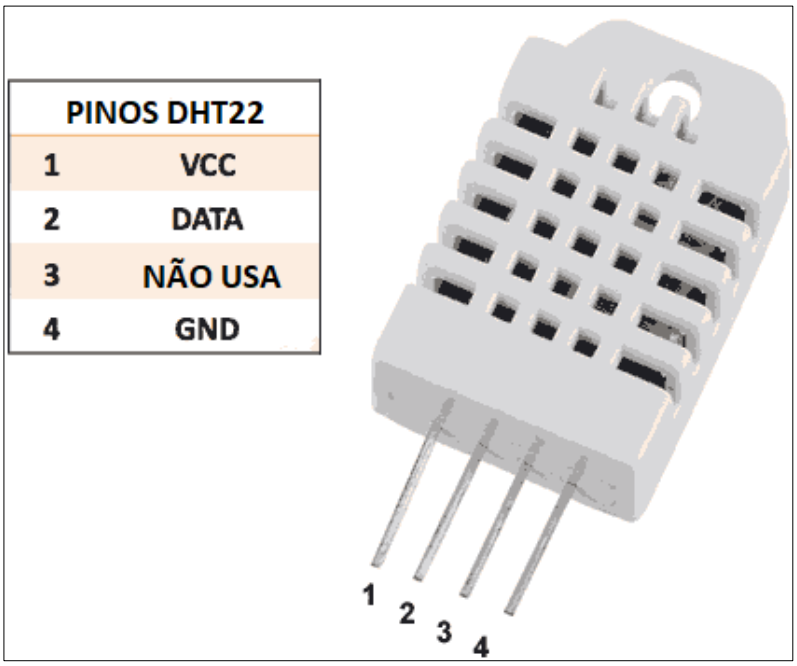


Figura 19 – Sensor DHT22
Fonte: Marian (2018)

4.7 Inversor de frequência ACS800

O protótipo descrito neste trabalho foi aplicado no painel de um inversor de frequência modelo ABB - ACS800-104-0030-5, responsável por controlar a velocidade do motor de uma correia transportadora.

Segundo (ABB, 2012), este inversor é utilizado em controle de motores de corrente alternada e é caracterizado pelos seguintes dados técnicos:

- Tensão de alimentação: 380 a 500Vdc;
- Potencia nominal: 22kW;
- Corrente nominal: 42A.

Este modelo é indicado para operar em condições ambientes onde a temperatura do ar esteja entre -15 a +50°C, e a umidade relativa do ar deve estar entre 5 a 95% (ABB, 2003). Portanto a manutenção preditiva deve ser realizada te tal forma que estes valores extremos não sejam alcançados.

4.8 Controlador lógico programável

No fim da década de 1960, os circuitos integrados foram muito explorados, onde foi desenvolvido minicomputadores, com aplicações em processos industriais e com controles online (MORAES e CASTRUCCI, 2007).

O CLP (Controlador Lógico Programável) nasceu basicamente na indústria automotiva, para ser mais específico, dentro da GM (General Motors) em 1969. Essa ideia surgiu, devido as montadoras necessitarem de adequar a lógica dos painéis de comando dos circuitos a relés, com a linha de produção. Essas mudanças geravam alto custo e tempo (MORAES e CASTRUCCI, 2007).

No ano de 1970 os controladores passaram a integrar na sua eletrônica, microprocessadores e com isso, foram denominados como Controladores Programáveis (CLP). Em 1980, os CLPs se aperfeiçoaram e as suas funções de comunicação, passam a ser utilizados em rede (MORAES e CASTRUCCI, 2007).

4.8.1 Arquitetura

Na figura 20, podemos observar a configuração de um CLP, no qual temos:

- Fonte de Alimentação – recebe da rede em alternada e converte para contínua. Caso ocorra a falta de energia, o programa estará seguro devido a uma bateria existente, para proteção. Tipos de fontes: *source* (parte interna do CLP) e *sink* (parte externa do CLP) (MORAES e CASTRUCCI, 2007);
- CPU – *Central Processing Unit* (unidade central de processamento), responsável pelo processamento do programa, memória de dados e memória-imagem das entradas e saídas (MORAES e CASTRUCCI, 2007);
- Memória:
    - EPROM – Responsável pela inicialização do CLP, toda programação vem de fábrica, onde não é possível acessar e realizar alterações (MORAES e CASTRUCCI, 2007);

  - RAM – Local onde se encontra a programação do usuário. O programa é processado e a memória de dados e memória-imagem, são atualizados (MORAES e CASTRUCCI, 2007);
  - Memória de Dados – Neste local, temos uma tabela de valores manipuláveis, referente ao processamento do programa do usuário (MORAES e CASTRUCCI, 2007);
  - Memória-Imagem – Reproduz a condição das entradas e saídas (MORAES e CASTRUCCI, 2007).
- Módulos de entrada e saída:
  - Entrada – Os módulos de entrada contêm optoisoladores em cada circuito. Quando um componente externo envia um sinal elétrico para à entrada do cartão, um diodo emissor de luz sensibiliza e faz com que circule uma corrente na entrada correspondente (MORAES e CASTRUCCI, 2007);
  - Saída – Os módulos de saída podem ser acionados de três formas: relé (capacidade de até 5A, maior robustez e vida útil baixa), triac (capacidade de até 1A, utilizado quando a fonte é VAC e vida útil longa) e transistor (capacidade de até 1A, utilizado quando a fonte é VCC e vida útil longa) (MORAES e CASTRUCCI, 2007).
- Terminal de programação – É o meio de comunicação que o desenvolvedor utiliza para implementação de programas através de um software do fabricante (MORAES e CASTRUCCI, 2007). Este meio de comunicação pode ser um computador, *notebook* ou *tablet*, na qual permite:
  - Autodiagnostico;
  - Revisão da programação;
  - Programação;
  - Monitoramento.

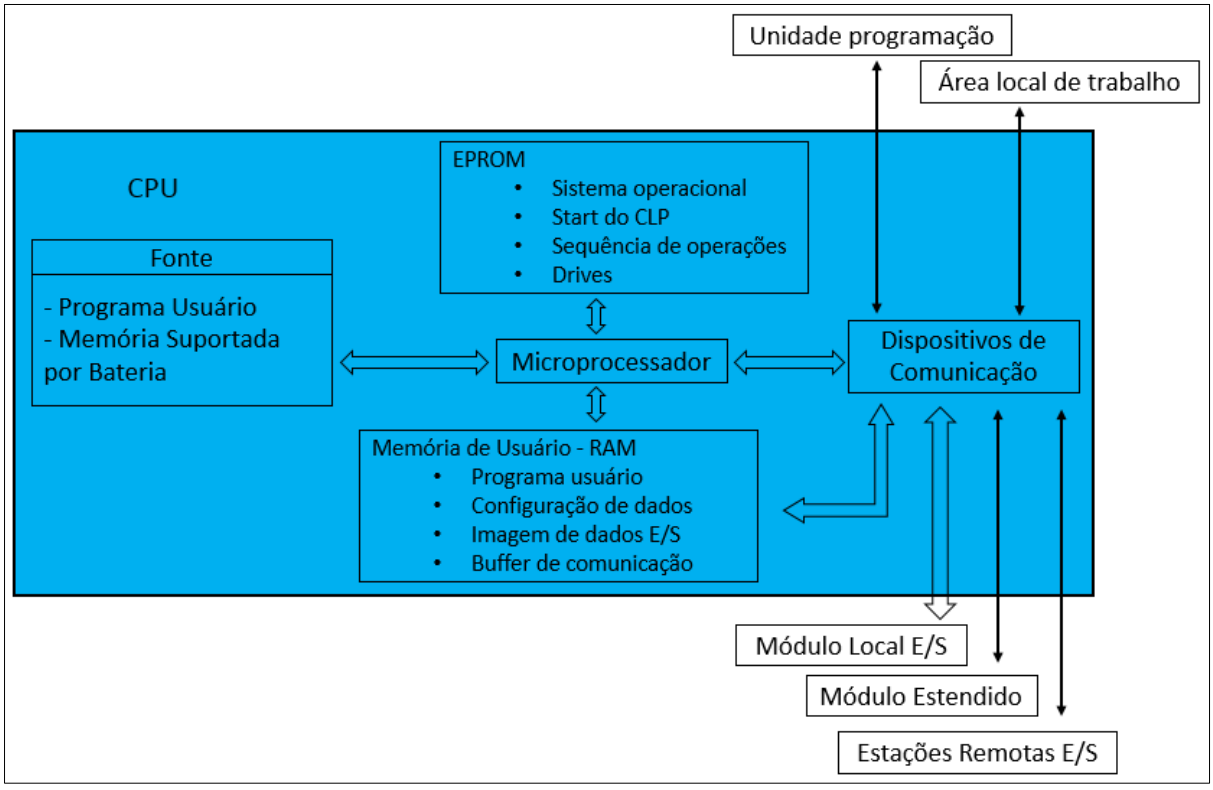


Figura 20 – Diagrama de blocos de um CLP
Fonte: Moraes e Castrucci (2007)

4.8.2 Ciclo de Execução (*Scan*)

O CLP executa algumas funções básica durante um ciclo, como:

- Atualização das entradas;
- Processamento da programação;
- Atualização das saídas.

Conforme podemos observar na figura 21, o *scan* ocorre em modo de ciclo fechado.

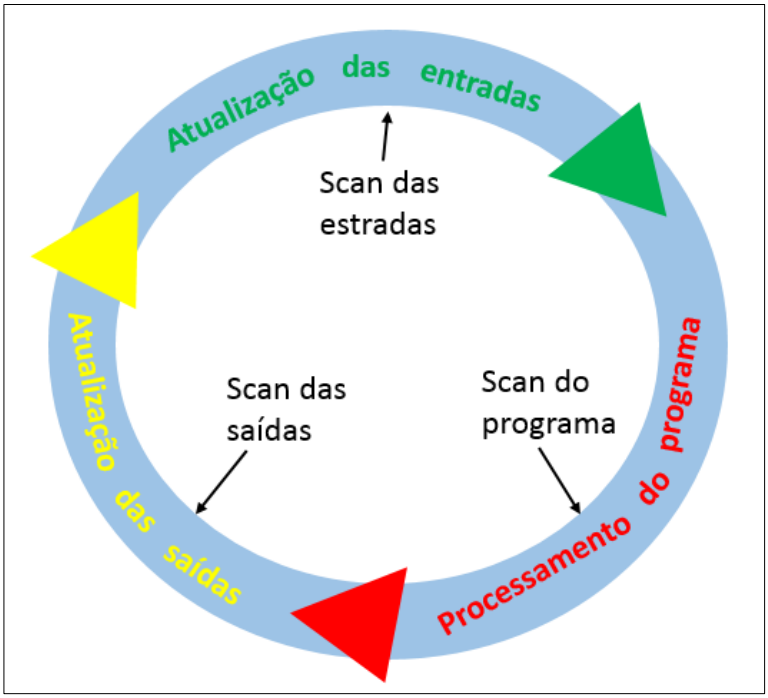


Figura 21 – Ciclo de varredura (*Scan*)
Fonte: Moraes e Castrucci (2007)

O CLP verifica a entrada e grava informações na memória-imagem de entrada. Depois realiza o *scan* e copia as informações da memória-imagem de saída, nos terminais de saída do módulo. Após o primeiro *scan*, a memória-imagem de entrada é resetada. O ciclo de varredura trabalha a partir dessa situação, atualizando a palavra e imagem de saída. Sempre que ocorre o *scan* da imagem de entrada, a palavra de entrada é atualizada (MORAES e CASTRUCCI, 2007).

4.8.3 CLP S7-400

O modelo de CLP utilizado neste estudo é o Step7-400 da Siemens, trata-se de um poderoso controlador com grande robustez, sua aplicação é infinita nos mais diversos processos produtivos (siderúrgica, farmacêutica, química, petróleo e entre outros). Vejamos na figura 22, um exemplo de S7-400.

Figura 22 – S7-400 Siemens
Fonte: Siemens (2017)

Segundo (UNICONTROL, 2007), temos as seguintes características desse modelo:

- CLP poderoso, modelos entre médio e grande porte;
- CPU's com diversos níveis de aplicações;
- Capacidade de expansão para 300 módulos;
- Pode se comunicar utilizando as seguintes redes:
  - MPI (Multipoint Interface);
  - PB (Profibus);
  - Profinet (Industrial *Ethernet*);
  - *TCP/IP*.
- Sem limitações de *slot*;
- Ferramenta integrada (HWConfig), para configuração e seleção de parâmetros;
- Multiprocessamento (permite utilizar até 4 CPU's no mesmo rack).

Na figura 23, podemos observar as variedades de cartões que compõem o S7-400, para a nossa aplicação, iremos comentar apenas sobre: PS (*power supply*), CPU e CP (processadores de comunicação). Para os outros cartões, como os de entrada e saída, estaremos substituindo-os pelo o Arduino.

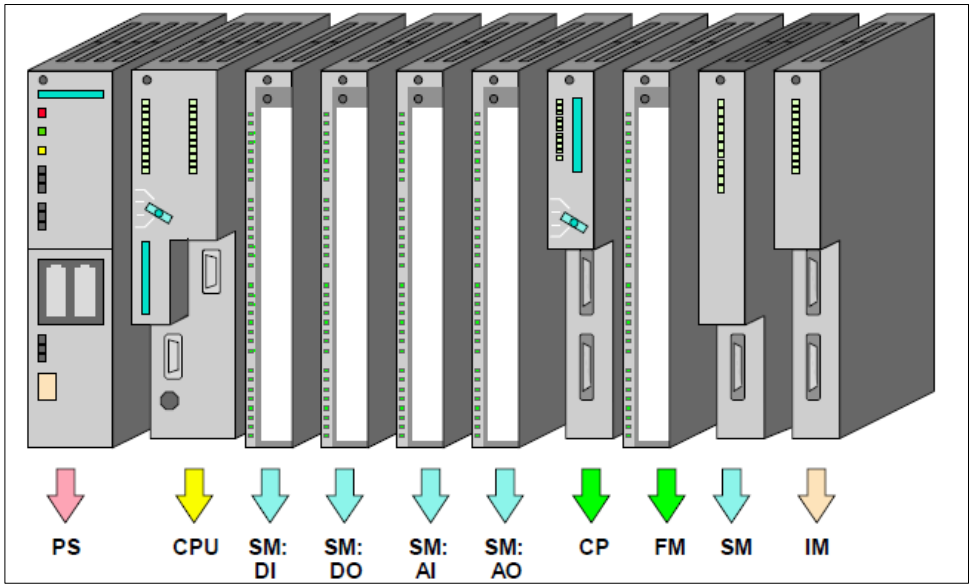


Figura 23 – Módulos S7-400
Fonte: Unicontrol (2007)

4.8.3.1 Módulos de Alimentação (*Power Supply Modules*)

Segundo a empresa (SIEMENS, 2016), os módulos de fonte de alimentação do S7-400, figura 24, alimentam os outros módulos no rack com suas tensões operacionais através do barramento de *backplane*. Eles não fornecem tensões de carga para os módulos de sinal.

Figura 24 – *Power Supply Module* (Módulo Fonte de Alimentação)
Fonte: Siemens (2007)

Conforme o manual da SIEMENS (2016), seguem algumas especificações comuns dos módulos de fonte de alimentação:

- Configuração encapsulado para uso em racks do CLP S7-400;
- Refrigeração via convecção natural;
- Conexão plug-in da tensão de alimentação AC-DC;
- Classe de proteção I (com condutor de proteção) de acordo com IEC 61140;
- Saídas com proteções de curto-circuito;
- Ambas as tensões de saída (5 VDC e 24 VCC) compartilham um terra comum;
- Análise de ambas as tensões, em caso de falha o mesmo envia um sinal de erro para a CPU;

- Bateria de *backup* opcional. Os parâmetros ajustados e o conteúdo da memória (RAM), são mantidos pelo barramento de *backplane* nas CPUs. É possível realizar uma reinicialização da CPU por causa do *backup* bateria, sem perder a programação realizada pelo usuário. O módulo da fonte de alimentação e os módulos de backup monitoram a voltagem da bateria;
- LEDs de falha/erro e status de operação na placa frontal.

4.8.3.2 CPU

Segundo (SIEMENS, 2005), a CPU tem como responsabilidade, o controle e regulação dos processos.

Dentro da família S7-400, existem variedades de CPU's. Estaremos ressaltando alguns dos recursos em comuns e exemplificando a compatibilidade em todo o sistema. Na figura 25, temos uma CPU de largura dupla com suas interfaces, controles do usuário e componentes de exibição (SIEMENS, 2005).

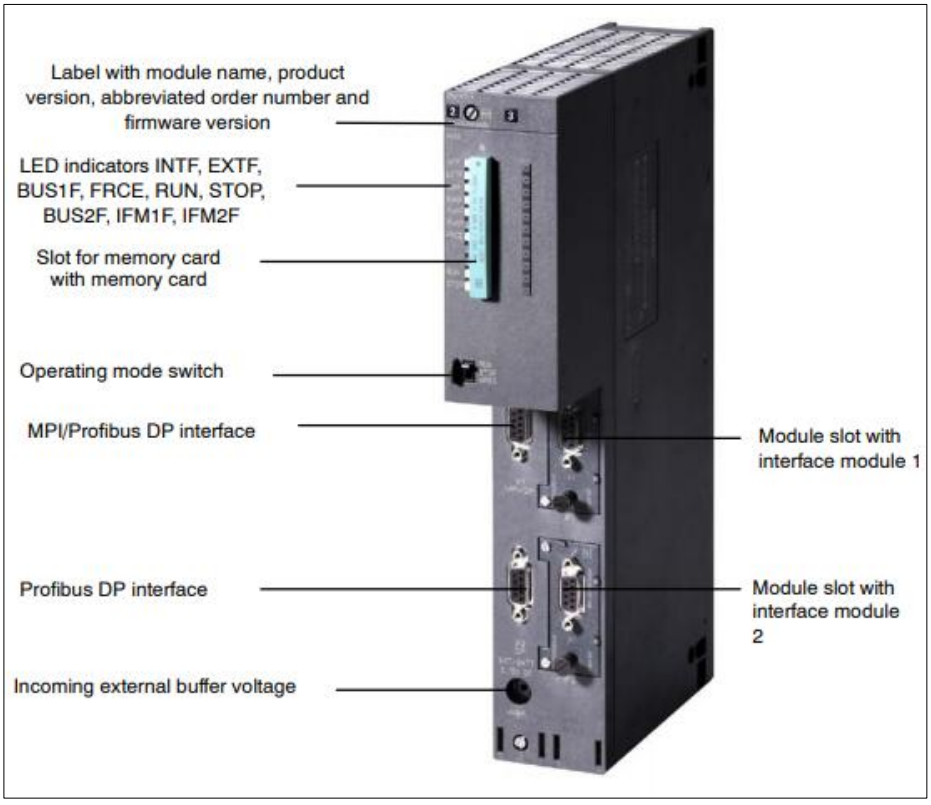


Figura 25 – CPU S7-400
Fonte: Siemens (2005)

- Interruptor do modo de operação do S7-400 – Seletor do modo de operação, RUN ou STOP. Também pode ser alterado pelo STEP 7 SIMATIC MANAGER;
- Interface MPI/DP – Os modelos de CPUs da família S7-400, possuem uma interface integrada de multiponto (MPI). A interface MPI é usada para as seguintes tarefas:
  - Programação e configuração de parâmetros;
  - Controle e monitoramento do usuário;
  - Configurando estruturas de rede.
- Interface Profibus DP – Exceto pela CPU 412-1, as demais CPUs possuem uma outra interface para comunicação via Profibus DP, além da interface MPI/DP. A interface Profibus DP é usada para:
  - A CPU atuar como o mestre DP para acessar todas as estações no Profibus DP;
  - A própria CPU é um escravo DP em uma interface DP.
- Funções Integradas – O sistema operacional de cada CPU, possui alguns blocos com funções específicas, que auxiliam na programação do usuário:
  - Controle operacional e funções de monitoramento;
  - Funções para comunicação;
  - Funções para diagnósticos;
  - Funções para transmissão de registros de dados;
  - Funções para controle de programa;
  - Funções para manipulação de interrupções;
  - Funções para gerar mensagens.
- Buffer de Diagnóstico – Cada CPU possui um buffer de diagnóstico, no qual mensagens de erro e diagnósticos são armazenados. As mensagens são inseridas no buffer de diagnósticos pela CPU ou por outros módulos. Mensagens de diagnósticos definidas pelo usuário também podem ser inseridas pelo usuário programa. Essas mensagens podem ser lidas pelo dispositivo de programação a qualquer momento. A data e hora da entrada também está incluída. As mensagens também podem ser enviadas para operadores de estações de controle e monitoramento. As entradas do buffer de diagnósticos não podem ser manipuladas (SIEMENS, 2005).

4.8.3.3 CP (Processadores de comunicação)

O processador de comunicação (CP), figura 26, foi desenvolvido para operação em conjunto com o S7-400, possibilitando a conexão do S7-400 à diferentes tipos de redes (SIEMENS, 2005).

O CP permite fornecer ou receber dados de outras CPU's ou máquinas por meio da interface padronizada OPC UA do controlador SIMATIC S7-400. O CP pode ser usado como um servidor OPC UA e/ou um cliente OPC UA (SIEMENS, 2017).

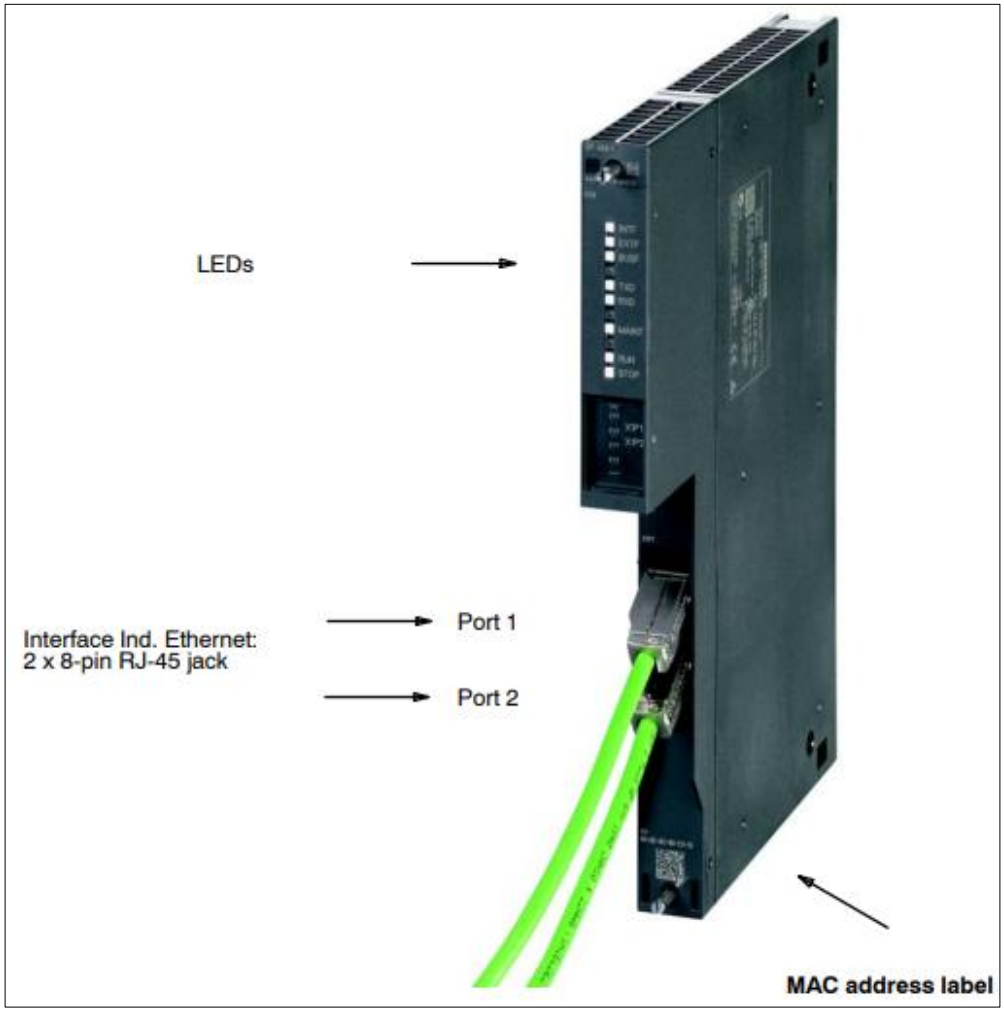


Figura 26 – CP (Processadores de comunicação)
Fonte: Siemens (2010)

O CP é exclusivo para automação com S7-400. Permite que o S7-400 seja conectado à Ethernet Industrial. Cada porta do switch foi desenvolvida para diagnósticos simples e é equipada com um LED duplo combinado RXD / TXD / LINK.

Para situações especiais, cada porta também pode ser definida para um modo fixo manualmente usando o STEP 7, por exemplo, 10 ou 100 Mbps (SIEMENS, 2010).

Conforme informado por (SIEMENS, 2010), na parte frontal do CP é composto por 9 LEDs, que sinalizam o modo de operação e status de comunicação. Note que na figura 27, podemos observar o significado de cada LED.

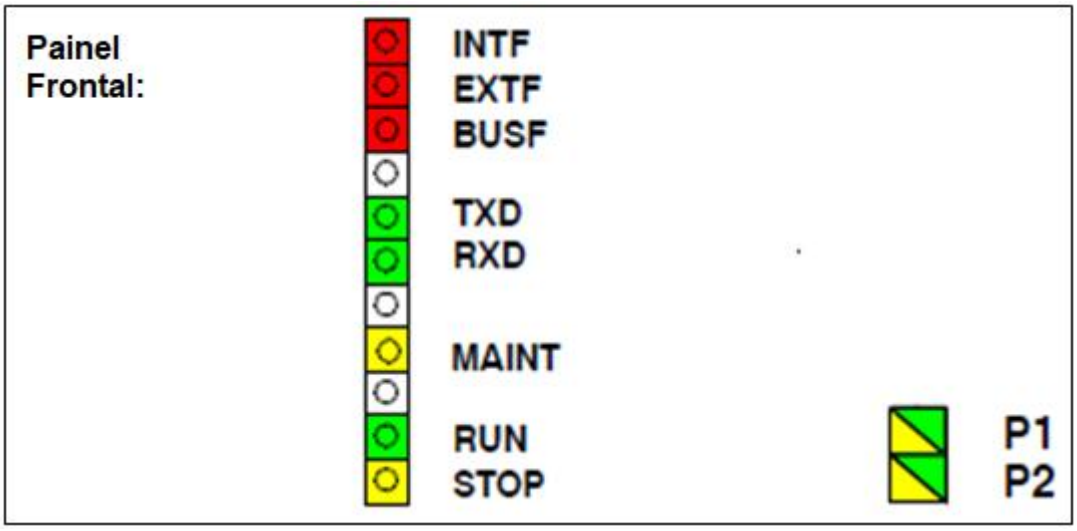
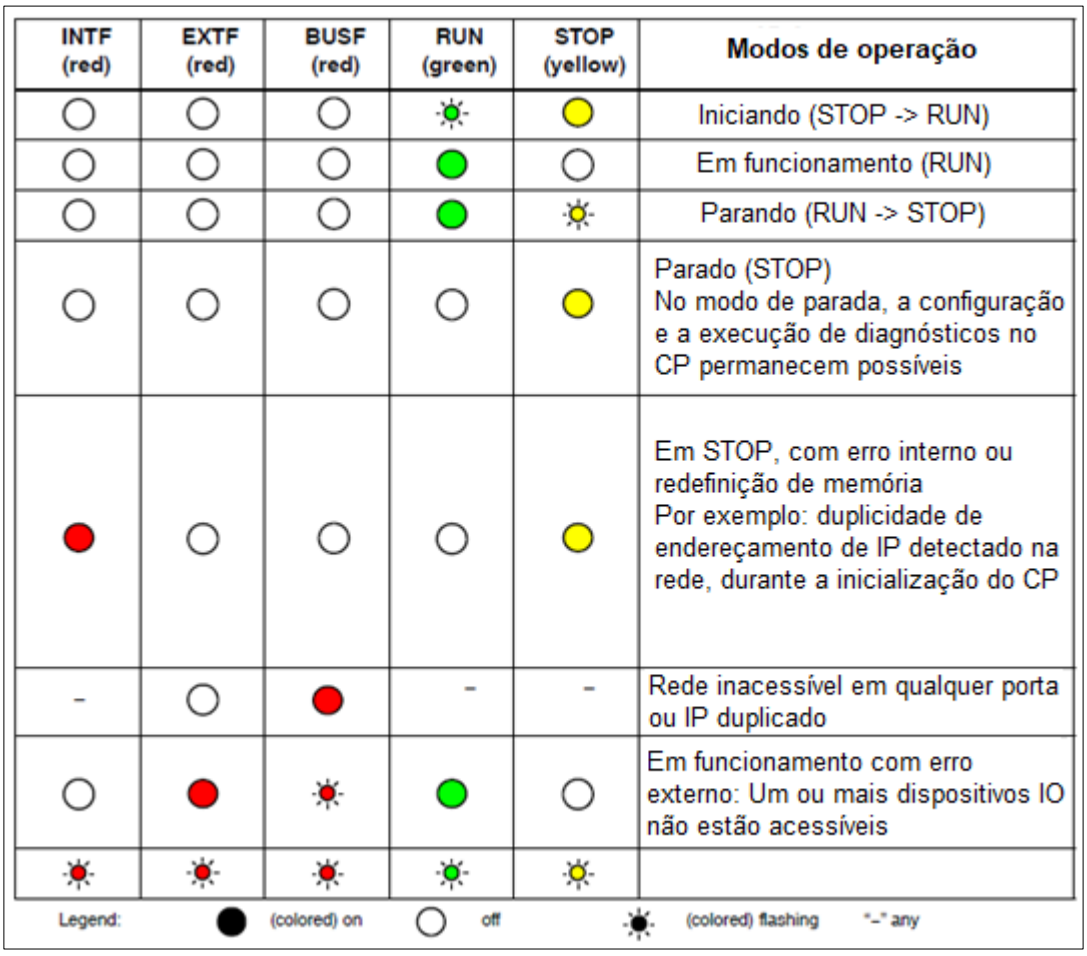


Figura 27 – LEDs frontais (Processadores de comunicação)
Fonte: Siemens (2010)

Os LEDs têm o seguinte significado:

- INTF: falha interna;
- EXTF: falha externa;
- BUSF: falha do bus;
- TXD: tráfego de quadros (envio) pela *Ethernet* (não relevante para dados PROFINET IO);
- RXD: tráfego de quadros (recebimento) pela *Ethernet* (não relevante para dados PROFINET IO);
- MAINT: Manutenção necessária (buffer de diagnóstico);
- RUN: Em funcionamento;
- STOP: Parado;
- P1 / P2: Status do link da porta *Ethernet* 1 / porta 2, atividade da porta *Ethernet* 1 / porta 2.

Nas figuras 28 e 29 temos o detalhamento das condições de indicação dos LED's.



INTF (red)	EXTF (red)	BUSF (red)	RUN (green)	STOP (yellow)	Modos de operação
○	○	○	☼	●	Iniciando (STOP -> RUN)
○	○	○	●	○	Em funcionamento (RUN)
○	○	○	●	☼	Parando (RUN -> STOP)
○	○	○	○	●	Parado (STOP) No modo de parada, a configuração e a execução de diagnósticos no CP permanecem possíveis
●	○	○	○	●	Em STOP, com erro interno ou redefinição de memória Por exemplo: duplicidade de endereçamento de IP detectado na rede, durante a inicialização do CP
-	○	●	-	-	Rede inacessível em qualquer porta ou IP duplicado
○	●	☼	●	○	Em funcionamento com erro externo: Um ou mais dispositivos IO não estão acessíveis
☼	☼	☼	☼	☼	

Legend: ● (colored) on ○ off ☼ (colored) flashing "–" any

Figura 28 – Status dos LEDs (Processadores de comunicação)
Fonte: Siemens (2010)

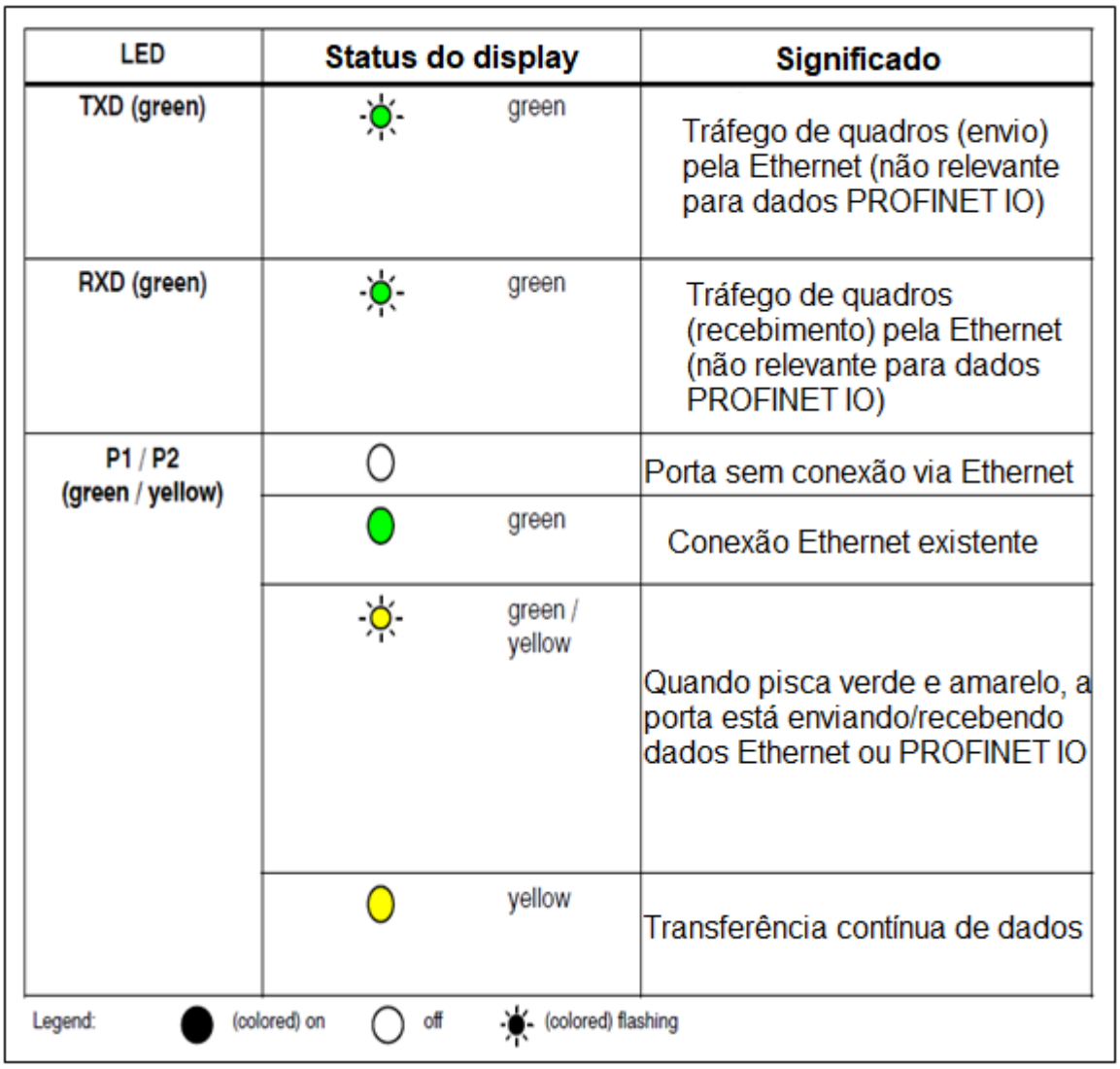


Figura 29 – Status dos LEDs (Processadores de comunicação)
Fonte: Siemens (2010)

4.8.4 Step 7 SIMATIC Manager

O SIMATIC Manager gerencia todos os projetos STEP 7. Conforme pode ser visto pela figura 30, após a sua instalação na estação de trabalho (estação de engenharia), a ferramenta pode ser acessada de duas formas (UNICONTROL, 2007):

1. Menu *iniciar* -> SIMATIC -> STEP 7 -> SIMATIC Manager*;*
2. Através do próprio ícone *"*SIMATIC Manager*"* na área de trabalho.

Figura 30 – Abrindo o SIMATIC Manager
Fonte: Unicontrol (2007)

O SIMATIC Manager é uma interface de gráfica com o usuário para a edição online/offline de objetos S7 (projetos, arquivos de programa do usuário, blocos, estações de hardware e ferramentas). Com o SIMATIC Manager é possível (UNICONTROL, 2007):

- Gerenciar projetos e bibliotecas;
- Habilitar as ferramentas STEP 7;
- Monitorar online o CLP;
- Configurar módulos de memória.

De acordo com (UNICONTROL, 2007), temos as seguintes ferramentas indicadas na figura 31, para desenvolvimento de projetos:

- LAD, STL, FBD - “Diagrama de Contatos”, “Lista de Instruções” ou “Diagrama de Blocos de Funções”, são os meios de programação para desenvolvimento de projetos;
- *Memory Card* – Reservado para armazenar os programas de usuário em módulos EPROM, tanto através da estação de engenharia, como de um gravador externo;

- *Configuring Networks* - Configuração de redes;
- *Setting the* PG-PC - Utilizado para selecionar o endereço local do nó e se conectar com a interface MPI;
- PID *Control* - São blocos “prontos” na biblioteca da Siemens, destinados a controle PID (malha fechada).

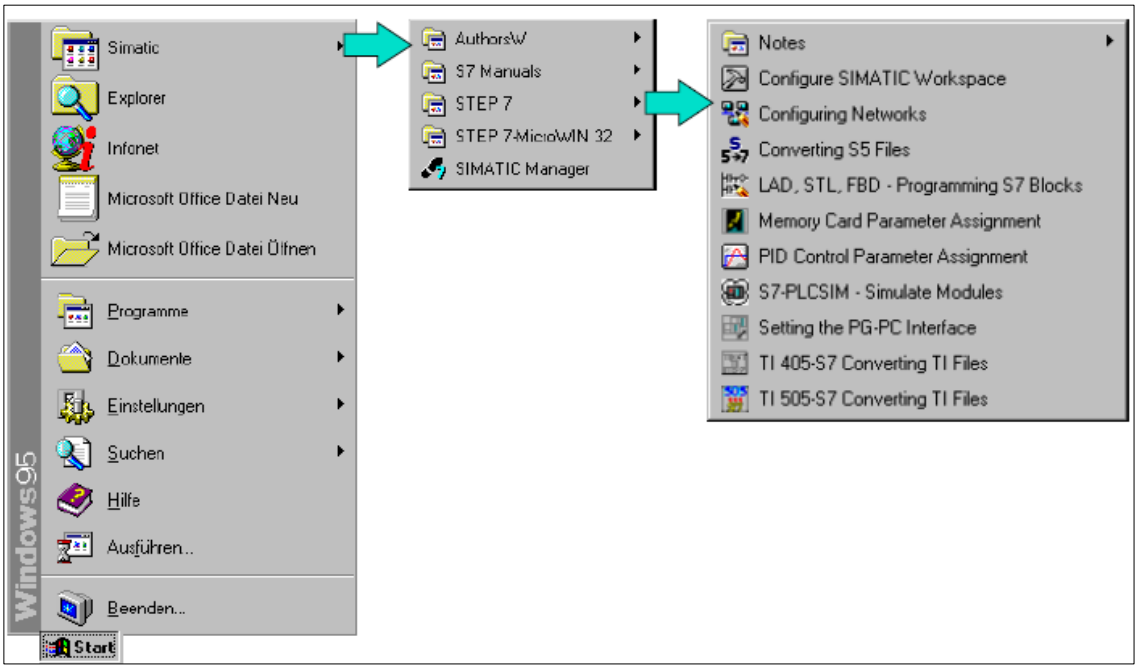


Figura 31 – Ferramentas do SIMATIC Manager
Fonte: Unicontrol (2007)

4.9 Redes de comunicação

Com o enorme avanço da tecnologia e crescente utilização de computadores, controladores e diferentes tipos de interface de processo, surgiu como consequência a necessidade de se estabelecer uma forma em que todos estes componentes estivessem conectados, possibilitando o compartilhamento de diferentes recursos como dados entre máquinas de forma simples tornando o processo até mesmo mais confortável para o operador (KUROSE e ROSS, 2013).

Portanto, as redes de comunicação industrial possibilitam esta interligação dos dispositivos de controle de processo em alta velocidade como por exemplo a interação entre controladores lógicos programáveis, sensores de detecção, motores, sistemas de supervisão, robôs, entre outros (SOUZA, 2010).

Existem hoje diversos tipos de redes e estas podem ser classificadas de acordo com a tecnologia de transmissão empregada, difusão ou ponto a ponto, ou de acordo com sua escala, podendo ser:

- PAN – Rede pessoal;
- LAN – Rede local;
- MAN – Rede metropolitana;
- WAN – Rede geograficamente distribuída.

Na figura 32 é possível observar a faixa de aplicação de diversas redes existentes de acordo com o nível de automação. Como podemos ver, a rede *Ethernet* pode ser aplicada em todos os níveis de automação.

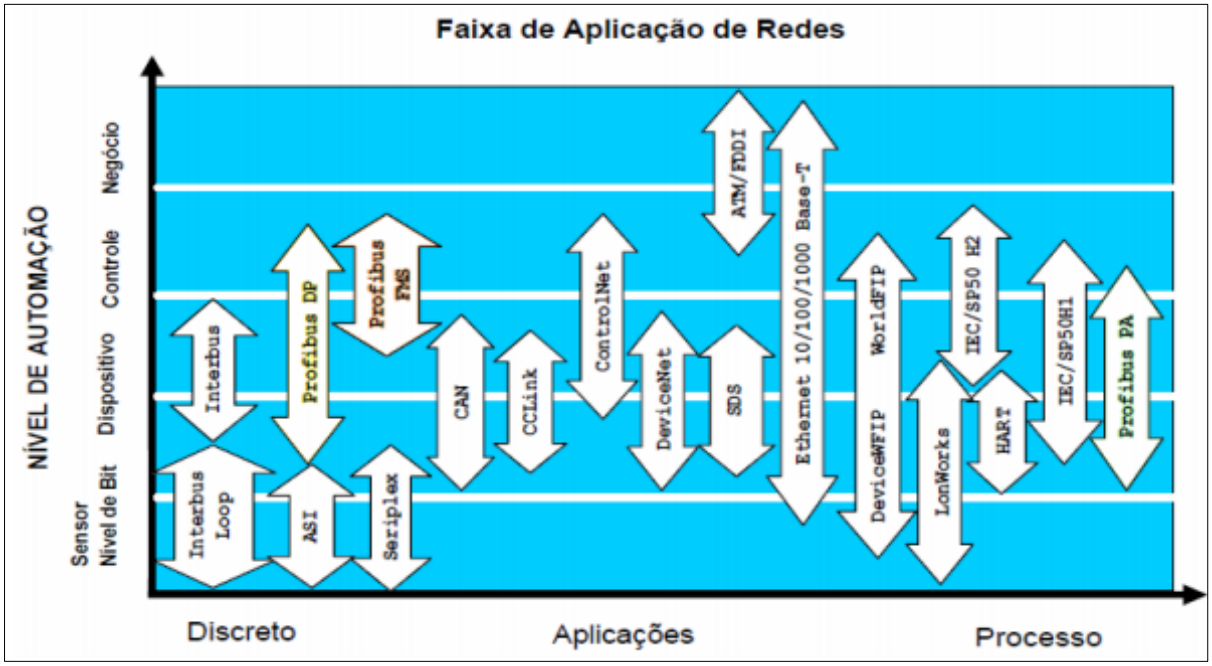


Figura 32 – Faixa de aplicação de redes
Fonte: Souza (2010)

4.9.1 Protocolos de comunicação

Para que dois equipamentos possam comunicar entre si é necessário que ambos utilizem o mesmo protocolo de comunicação para troca de dados, como uma forma de “idioma” que possa ser entendido pelos dois. Pode se dizer que o protocolo de comunicação é este “idioma”, que possibilita que diferentes sistemas que estejam

conectados a uma mesma rede, comuniquem-se e troquem dados entre si (LILA, 2012).

4.9.2 TCP/IP

O TCP/IP é hoje o principal protocolo ou conjunto de protocolos para envio e recebimento de dados. A sigla TCP significa *Trasmission Control Protocol* (Protocolo de controle de transmissão) e o IP, *Internet Protocol* (Protocolo de *Internet*). Portanto o TCP/IP é um conjunto de protocolos que permite que diversos tipos de aplicações troquem dados entre si.

Como dito no início, O TCP/IP na realidade é um conjunto de protocolos que pode ser dividido em quatro camadas, conforme mostrado na figura 33, responsáveis por tarefas distintas garantindo assim a integridade dos dados que trafegam pela rede (MARTINS, 2012).

Figura 33 – Camadas TCP/IP
Fonte: Martins (2012)

- Camada de aplicação – Quando um dispositivo tem a necessidade de requisitar algo que está na rede, esta requisição é feita nesta camada. Nela, pode ser encontrado os protocolos como SMTP (para e-mail), FTP (transferência de arquivos) e o HTTP (para navegar na *internet*) (MARTINS, 2012);
- Camada de transporte – Responsável por receber os dados da camada de aplicação realizar a verificação e separá-los em pacotes que serão enviados a próxima camada (MARTINS, 2012);
- Camada de rede – Nesta camada os pacotes são anexados aos endereços virtuais (IP), do remetente e destinatário podendo assim serem enviados através da camada de interface (MARTINS, 2012);

- Camada de Interface – A função desta camada é enviar ou receber os pacotes de dados da rede. Para isso é necessário um tipo de estrutura nesta camada. Uma estrutura bastante utilizada atualmente é a *Ethernet* (MARTINS, 2012).

4.9.3 Ethernet

Atualmente, o Ethernet é o padrão mais utilizado em redes locais (LAN) nas indústrias, devido sua simplicidade de aplicação e também por atender os requisitos mínimos para que se estabeleça uma conexão entre diferentes equipamentos. Pode ser definido como uma arquitetura que possibilita a interligação de redes locais (LAN) através de cabos de par trançado ou até mesmo fibra ótica.

Os dispositivos que estejam usando o protocolo Ethernet necessitam basicamente de um adaptador Ethernet (placas ou Shields dedicados) e cabos no padrão Ethernet como mostrado na figura 34 que interliguem os equipamentos. A interconexão de dispositivos pode ser feita de forma direta ou através de equipamentos adicionais como *hubs*, *switches* e roteadores, permitindo a expansão da rede.

A principal desvantagem desta configuração é a limitação de 100 metros de distância para transmissão, com o uso de par trançado, por conta disso, como não será utilizado repetidores neste projeto, esta é a distância máxima em que os dispositivos poderão estar entre si.

Figura 34 – Cabo de par trançado para rede Ethernet
Fonte: Digimer (2018)

4.9.4 Endereçamento IP

É por meio do endereço de IP de uma máquina que é realizado o envio dos dados através de uma rede TCP/IP. O IP consiste no endereço de identificação de uma determinada máquina na rede, permitindo que a troca de dados seja feita de forma referenciada sem que as informações sejam perdidas, garantindo o envio ao local específico. O Endereço IP é composto por quatro números, separados por pontos, onde uma parte representa a rede (NetID) e a outra, o equipamento (*host*) em específico (SOUZA, 2013b).

Ainda segundo (SOUZA, 2013b), as partes representadas no endereço IP podem variar de acordo com a classe de endereço IP. Na tabela 2 podemos ver um comparativo entre as quatro classes existentes.

Exemplo de endereço IP: 172.26.221.10

Tabela 2 – Classes de endereço IP

Classe	Bytes			
	1°	2°	3°	4°
A	Rede	Host	Host	Host
B	Rede	Rede	Host	Host
C	Rede	Rede	Rede	Host
D/E	Broadcasting	Broadcasting	Broadcasting	Broadcasting

Fonte: Souza (2013b)

Neste projeto foi utilizado a classe C onde os três primeiros *bytes* representam o número da rede e o último *byte* representa o número do *host*.

4.9.5 Máscara de sub-rede

Segundo (SOUZA, 2013b), assim como o endereço IP, a máscara de sub-rede é utilizada para definir a classe de endereçamento, portanto quando é configurado o TCP/IP, deve-se também especificar a máscara de sub-rede, especificando qual parte do endereço IP representa a rede e qual representa o host (computador ou equipamento).

Na configuração da máscara de sub-rede, também são utilizados quatro números separados por pontos e o número 255 é utilizado para identificar a parte que se refere ao endereço de rede.

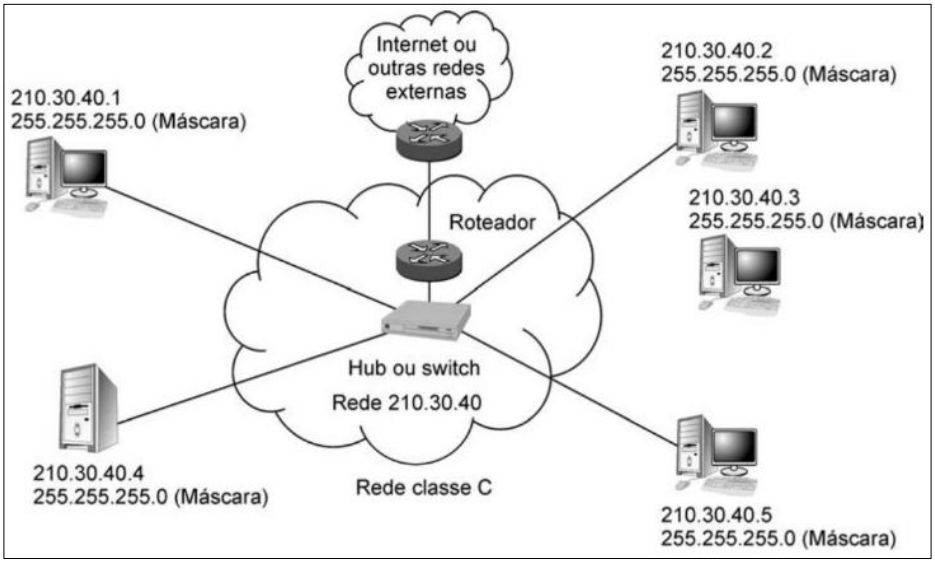


Figura 35 – Exemplo de configuração de rede classe C
Fonte: Souza (2013b)

4.9.6 Hub

Também conhecido como concentradores, os *hubs* são dispositivos capazes de transmitir pacotes de dados entre diferentes equipamentos conectados em uma mesma rede. Um *hub* possui diversas portas possibilitando a conexão de diversos dispositivos que necessitam estabelecer uma conexão entre si. Uma das principais características de um *hub* é que uma mesma informação é enviada para muitos receptores ao mesmo tempo (*broadcasting*) e somente aqueles pré configurados para receberem os dados, terão os dados recebidos.

O *hub* é mais indicado para pequenas redes, devido a ocorrência de colisões entre diferentes pacotes, sem contar que o tráfego de dados ocorre de forma mais lenta quando comparado à um *switch*, visto que a transmissão de dados ocorre uma de cada vez, onde caso o mesmo esteja transmitindo uma informação e um equipamento necessite enviar novos dados, isto só ocorrera após a finalização da operação anterior.

Portanto, devido a não necessidade de configuração e por nosso projeto se tratar de um protótipo com no máximo três dispositivos (Arduino, CLP e computador) conectados, decidimos utilizar o *hub* que possui uma fácil aplicação e atende a necessidade do projeto.

Na figura 36 podemos visualizar o modelo de hub de cinco portas que foi utilizado.

Figura 36 – Hub Multilaser RE10
Fonte: Nil Tec (2018)

4.9.7 Tecnologia OPC

Considerado uma forma padrão líder em conectividade de automação industrial, o sistema OPC (OLE *for Process Control*) foi desenvolvido com a finalidade de permitir que sistemas de controle (dispositivos atuantes no processo, sistemas de automação, entre outros) possam se comunicar de uma forma padrão com diversas tecnologias criadas pela Microsoft para plataforma WINTEL (computadores com sistema operacional Windows, da Microsoft e processadores da Intel), permitindo assim, o acesso aos dados de processo por diversos sistemas como SCADA, MES, *PI System* entre outros. Em 1999 a Microsoft desenvolveu a tecnologia OLE (*Object Linking and Embending*) com a finalidade de resolver o problema de ser necessário a integração de diversas aplicações dentro da plataforma Windows (FONSECA, 2002).

4.9.7.1 Cliente ou servidor OPC

Segundo (FONSECA, 2002), as aplicações que fazem uso da tecnologia OPC podem ser classificadas como clientes, servidor ou até mesmo possuir a integração dos dois. Geralmente, os sistemas de aquisição de dados (sistemas supervisórios entre outros) são clientes OPC, em contrapartida, as aplicações responsáveis pela interface direta com os equipamentos de campo, são os servidores.

Portanto, para a elaboração deste projeto foram configurados um servidor e um cliente OPC, sendo o servidor, responsável por transferir os dados de campos (umidade, temperatura e corrente) adquiridos pelo CLP através do Arduino, e o cliente OPC do PI System, responsável por receber os dados do CLP e transferi-los ao servidor do PI System.

4.9.7.2 *Software* KepServerEX

A configuração do servidor OPC do projeto em estudo foi feita por meio do *software* KEPServerEX, que pode ser definido como uma plataforma de conectividade do setor que fornece uma única fonte de dados de automação industrial para diversos aplicativos. O design desta plataforma, figura 37, permite que os usuários conectem, gerenciem, monitorem e controlem diversos dispositivos de automação e aplicativos de *software* por meio de uma interface de usuário intuitiva. O KEPServerEX utiliza o OPC (o padrão da indústria de automação para interoperabilidade) e os protocolos de comunicação centrados em TI (como SNMP, ODBC e serviços da Web) para fornecer aos usuários uma única fonte de dados industriais. Esta plataforma é desenvolvida e testada para atender aos requisitos de desempenho, confiabilidade e facilidade de uso.

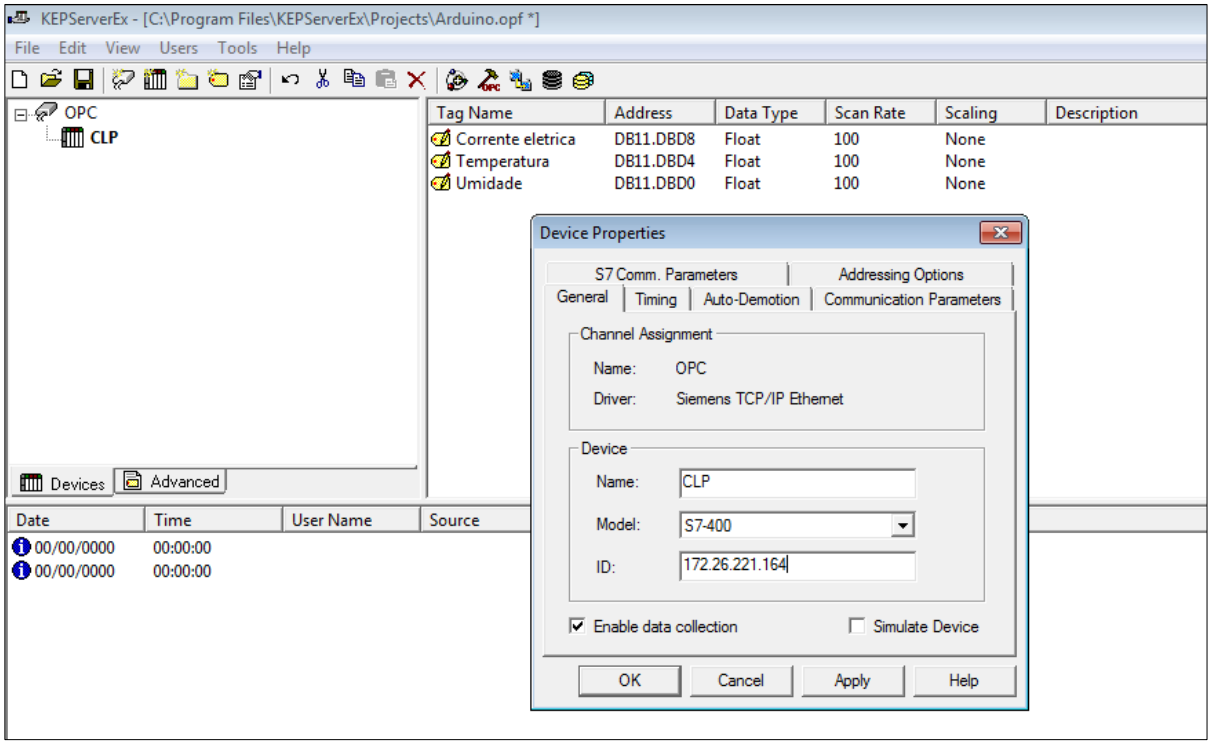


Figura 37 – Servidor OPC KepServerEX
Fonte: Autores (2018)

4.10 PI System

PI System é um *software* de gerenciamento de dados e armazenamento, sua configuração é montada em um servidor, na qual é aplicável para controle de dados com milhões de *tags*. É similar à um supervisório de controle de processo, permite desenvolvimento de telas gráficas para representação do processo, alarmes de falhas, banco de dados e visualização em tempo real da condição de operação da planta. Sua particularidade se destaca nas outras ferramentas disponíveis, onde um supervisório comum não tem, é possível enviar notificações (e-mail de falha ou alarme) instantâneo da variável monitorada para qualquer pessoa ou responsável de qualquer lugar, envio de mensagens de textos, interação com o Excel, cálculos estatísticos, acesso remoto através de dispositivos *mobile* e acesso de qualquer computador, possibilitando a visualização da planta.

O PI System utiliza algumas ferramentas, para análise e visualização de dados, que tem como objetivo, fazer com que esses dados armazenados em um servidor, cheguem até aos usuários, independentemente do local que se esteja. Essas ferramentas, trabalhando em conjunto, torna mais fácil para o usuário acessar e

alavancar os dados no trabalho do dia-a-dia. Com as ferramentas do PI System ou PI System Tools, os usuários podem visualizar os dados armazenados e coletados da forma que precisam, para entender e agir rapidamente (OSISOFT, 2017).

Segundo (GENENA, 2004), o PI (*Plant Information*) é composto por *software* servidor/cliente desenvolvidos com o objetivo de automatizar totalmente a coleta, armazenamento e apresentação das informações de uma planta de um processo, onde todas essas operações serão gerenciadas. Assim, o sistema *PI* é constantemente utilizado como uma integração e plataforma de desenvolvimento de largas aplicações, passando a ser um link entre área industrial e escritório. O *software* foi desenvolvido para executar a monitoração e análise de plantas de processo, trabalhando como servidor de dados para aplicações cliente. Usuários podem usar as aplicações clientes para visualizar os dados armazenados no servidor *PI*.

Segundo (SCHEUER, 2004), o sistema PI (*Plant Information*) se equipara basicamente em softwares e servidor/cliente, para registro, análise e monitoramento dos dados de uma planta ou processo. O PI Universal Data Server (PI UDS) é o centro do sistema, sendo a sua principal função, atuar como um servidor de dados, utilizando os mesmos princípios da Microsoft Windows. Diversos usuários dentro da empresa, sem restrições de nível hierárquico (operadores, engenheiros, gerentes e outros interessados no processo), podem se beneficiar com uma grande variedade de aplicações “clientes” para acessar ao PI UDS e analisar dados da planta de um processo, armazenados no sistema de arquivos (PI Archive), veja na figura 38.

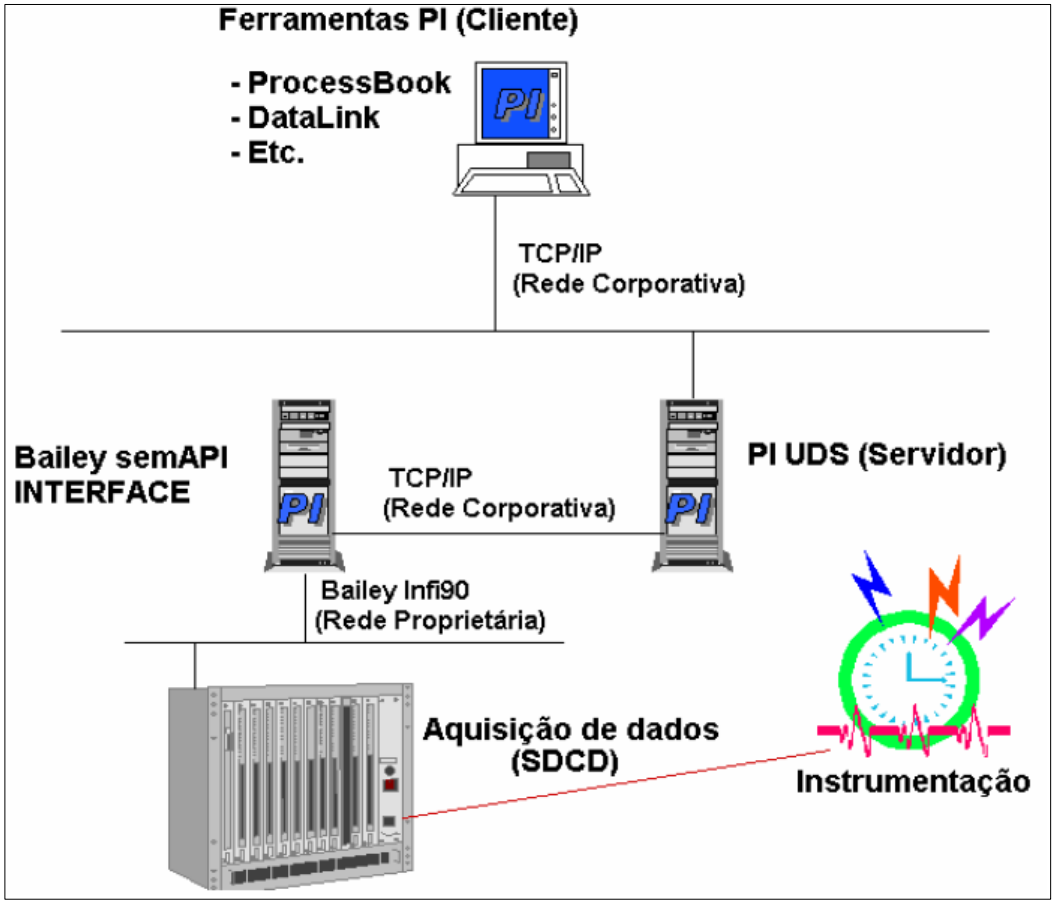


Figura 38 – Arquitetura do PI System
Fonte: Genema (2004)

Na figura 39, pode-se observar a configuração das partes envolvidas no PI System.

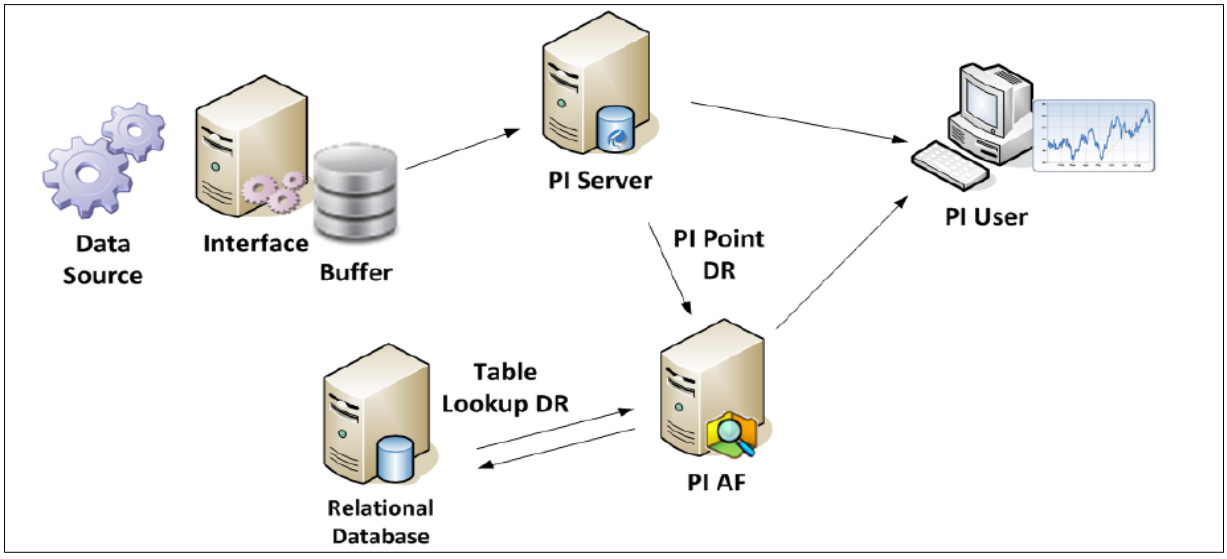


Figura 39 – Partes envolvidas em um PI System
Fonte: OSISoft (2012)

Os dados na *Data Source* (Fonte de Dados) são coletados pela interface *PI* da OSIsoft (empresa desenvolvedora do *software*) e enviados ao PI Server da OSIsoft, organizados pelo PI AF e Lidos pelo PI User (OSISOFT, 2012).

Segundo (OSISOFT, 2018), o PI System pode ser subdividido em 4 partes para se obter a inteligência operacional, veja na figura 40:

Figura 40 – Inteligência Operacional
Fonte: OSIsoft (2018)

Capturar, pesquisar, analisar, visualizar e compartilhar. Em cada uma das fases leva a uma compreensão detalhada, com base nos entendimentos permitirá que as organizações transformem a tomada de decisão de reativa para proativa. Podemos observar que, constantemente se descobre novas oportunidades para encontrar novos valores:

- Otimizar os processos;
- Elevar a qualidade;
- Aumentar a produtividade;
- Melhorar a eficiência energética;
- Gerenciar riscos;
- Avançar com o desempenho da segurança.

Com estas soluções para os principais negócios, as empresas serão competitivas e preparadas para chegar a excelência operacional (OSISOFT, 2018).

4.10.1 Capturar

Coletar, conectar e centralizar. Uma arquitetura escalável e aberta, e mais de 450 interfaces patenteadas e prontas para uso dão ao PI System sua capacidade única para coletar dados de séries cronológicas e de alta fidelidade a partir de uma diversidade de fontes - padrão, idioma, frequência, velocidade de entrega, formato ou dispositivo (OSISOFT, 2018).

Foi projetado para trabalhar com milhões de informações (pontos de dados), centraliza os dados com base em eventos e em tempo real. Diversas informações se juntam, as informações inconsistentes são eliminadas e todas as fontes de dados passam a ser únicas. Gerando assim uma infraestrutura de informação receptiva e sólida, apresentando ao usuário final um conhecimento incisivo e pronto para tomada de decisão imediata (OSISOFT, 2018).

4.10.2 Pesquisar e analisar

Compara de forma instantânea as informações históricas, com as de tempo real. Realiza a entrega de dados de forma rápida e eficiente, acessando décadas de importantes dados armazenados, comparando com os dados coletados em tempo real. Em apenas poucos segundos, é possível realizar análise de informação não antes acessível. Possibilitando investigar problemas, solucionar falhas de equipamentos, comparar o desempenho operacional do passado com o atual e medir plantas novas em relação as existentes (OSISOFT, 2018).

4.10.3 Visualizar

Possibilidade de acessar informações em qualquer dispositivo. Personalizar o conteúdo da planta para diferentes públicos. Apresentar a planta remotamente e online. Com a capacidade de visualização do PI é possível:

- Consolidar e visualizar dados operacionais e de negócios (OSISOFT, 2018);

- Exportar dados para Microsoft Excel para descobrir o verdadeiro custo e o valor das decisões de operação, e desenvolver relatórios personalizados para as autoridades reguladoras (OSISOFT, 2018);
- Traduzir para praticamente qualquer dispositivo - tablet, celular, laptop (OSISOFT, 2018)
- Ver dados carregados manualmente junto a dados de operações automatizados (OSISOFT, 2018);
- Visualizar tudo a partir de diagramas de processo detalhados e display de autoconfiguração para análise de batelada/completa e KPI/dashboards (OSISOFT, 2018).

4.10.4 Compartilhar

O PI System compartilha com toda a empresa, fornecedores, investidores e clientes informações operacionais da planta a qualquer hora, lugar e em qualquer dispositivo. Todas as informações são encaminhadas de forma segura, permitindo rastreabilidade, para onde estão indo e para quem está sendo enviado.

4.10.5 Visão Técnica

A arquitetura do PI System é projetada de forma que possibilite a sua expansão a medida que os negócios aumentem. Podemos observar na figura 41 e 42 dois exemplos de cenários com o PI. Na figura 43 é representado uma visão geral do PI System.

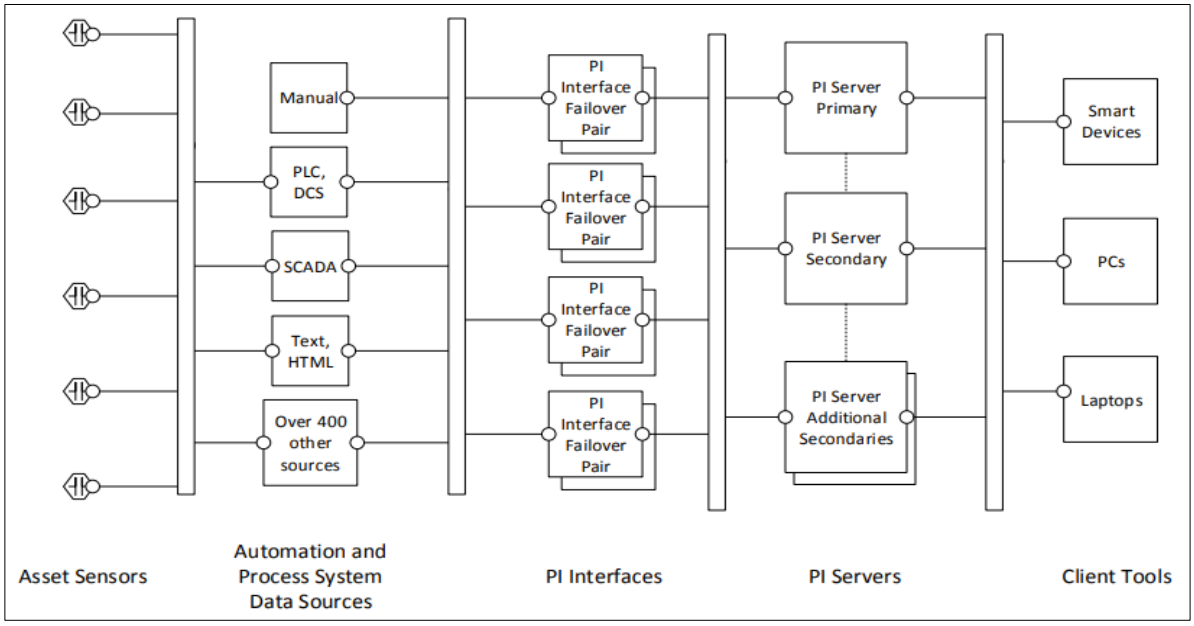


Figura 41 – Implementação em vários locais
Fonte: OSIsoft (2018)

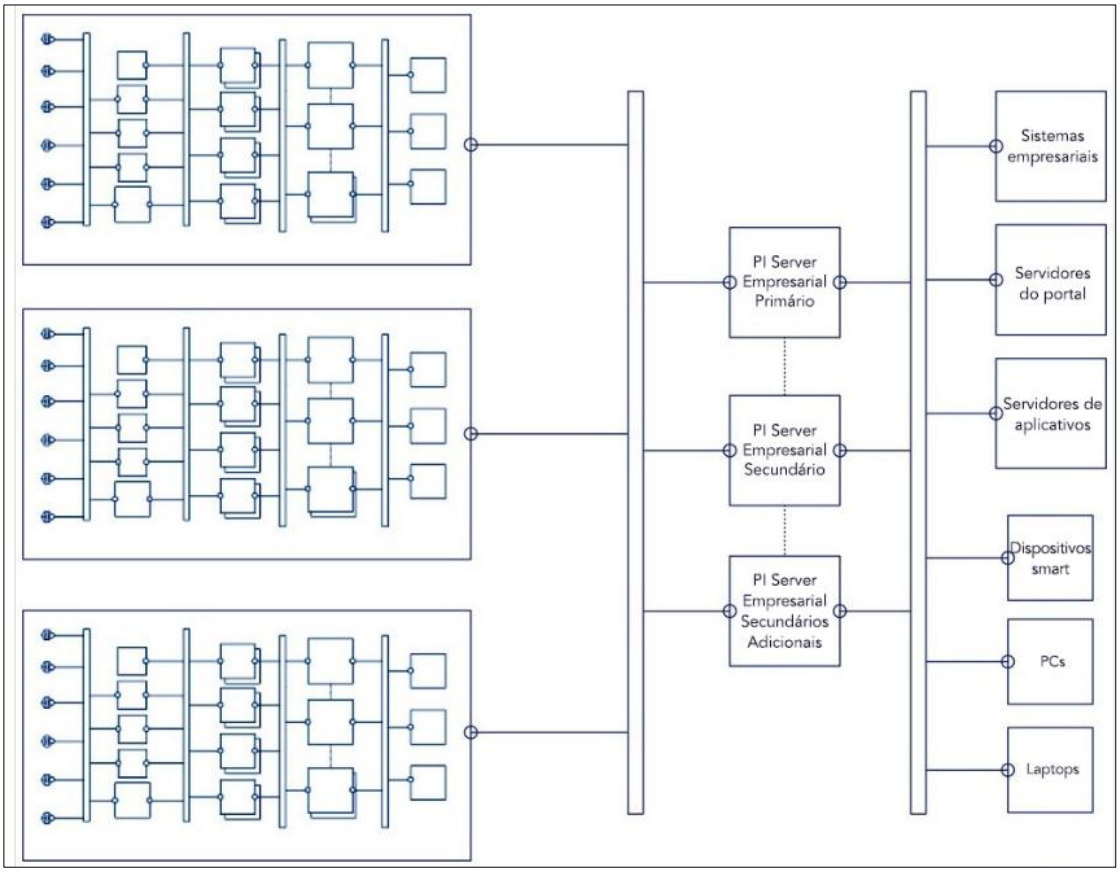


Figura 42 – Implementação em toda a empresa
Fonte: OSIsoft (2018)

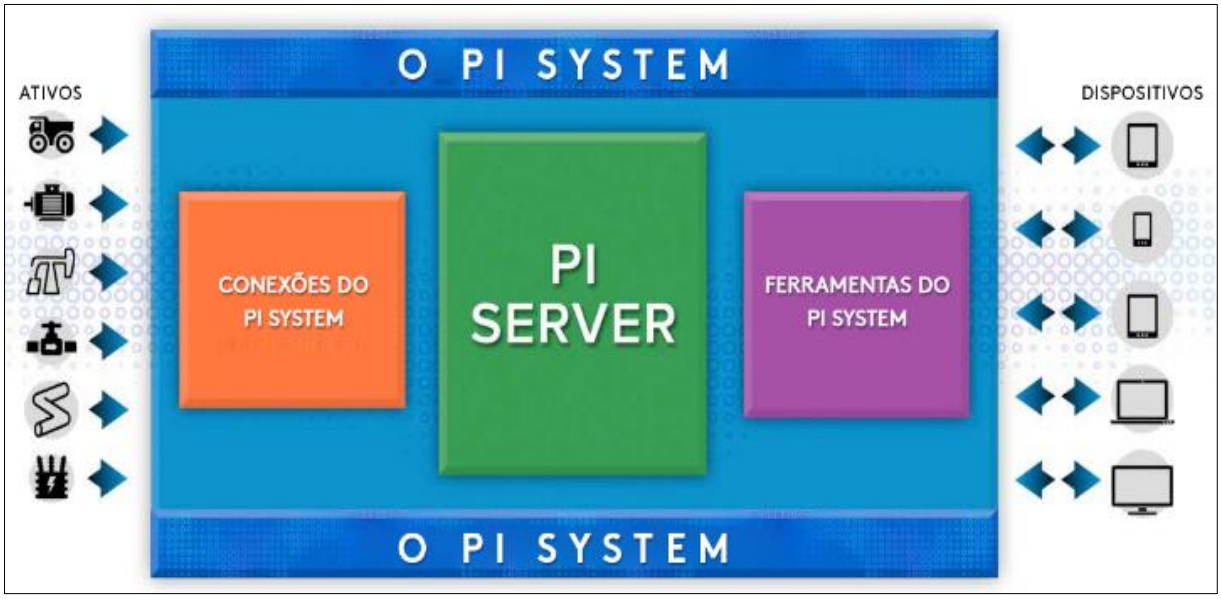


Figura 43 – Visão geral do PI System
Fonte: OSIsoft (2018)

4.10.6 Compreendendo o fluxo de dados no PI Server

Segundo (OSISOFT, 2012), o PI Data Archive Server (PI Server) tem a finalidade de armazenar os dados da planta em *tags* de forma individual, ela coleta informações de uma fonte de dados. A *tag* é um ponto único no PI System.

O fluxo de dados no PI Server é o primeiro mecanismo, conforme figura 44.

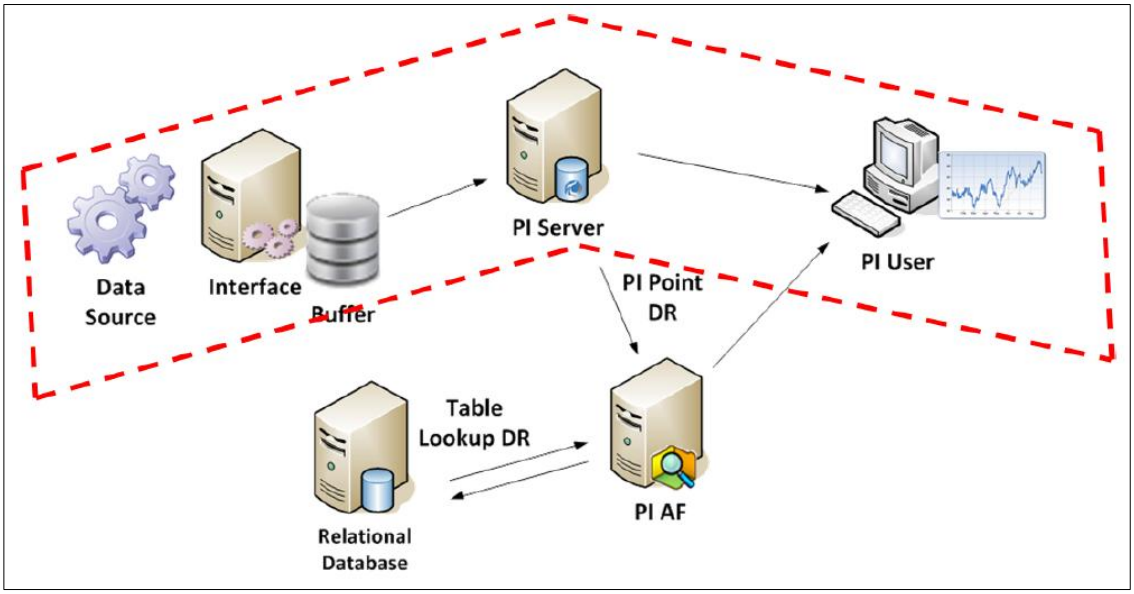


Figura 44 – Fluxo de dados no PI Server
Fonte: OSISoft (2012)

Veremos abaixo o passo a passo de cada etapa, da 1ª parte destacada na figura 44.

1. Fonte de dados: É o ponto inicial da coleta de dados, trata-se de um servidor da planta (qualquer interface realizando leitura do processo, temperatura, corrente, tensão e etc.);
2. Interface PI: Lê o valor bruto da fonte de dados;
3. Interface PI: Atribui um *timestamp* (registro de data e hora, é uma sequência de caracteres ou informações codificadas que identificam quando um determinado evento ocorreu, geralmente fornecendo data e hora do dia, às vezes com precisão de uma pequena fração de segundos), aplica exceções (realização de filtragens) e passa o valor para o *buffer* (Memória Principal, é uma região de memória física utilizada para armazenar temporariamente os dados enquanto eles estão sendo movidos de um lugar para outro);
4. Interface PI: Os valores que estão no *buffer*, são encaminhados para a tabela Snapshot (é uma cópia de dados que possui comportamento semelhante a um *backup*) no PI Server;
5. PI Server: Recebe as informações da Interface PI, filtra os dados e passa o valor da tabela Snapshot para a fila de eventos;
6. PI Server: Passa o valor da Fila de Eventos para os *archives* do PI;
7. PI User: Lê os dados da tabela de snapshots do PI Server e dos arquivos de *archive*.

4.10.7 Compreendendo o fluxo de dados no PI AF

Segundo (OSISOFT, 2012), a principal finalidade do PI AF (*PI Asset Framework*) é organizar os dados do ativo (equipamentos do campo, atuadores, sensores, plantas e etc), de acordo com a estrutura do equipamento monitorado ou das *tags* das fontes de dados. Facilita aos usuários não familiarizados com o processo/planta, para que consigam encontrar e usar os dados armazenados no *PI*. O PI AF trabalha de forma organizado de elementos e atributos. O PI User pode recuperar valores de atributos dos elementos. Um elemento é representado no PI AF como um equipamento, onde o atributo é uma propriedade deste equipamento. Temos

como exemplo um silo, sendo um elemento e seus atributos uma propriedade, como o conteúdo dentro do silo, temperatura e nível.

Um atributo é caracterizado pela fonte de seus dados. Os dados de um atributo podem ter origem em uma *tag* PI do PI Server, em uma pesquisa em tabela de um banco de dados relacional, em uma fórmula matemática ou ainda ser uma constante inserida no horário de criação do atributo. O PI User pode exibir valores de atributos de todas as diferentes fontes no mesmo cliente (OSISOFT, 2018).

O segundo mecanismo é o fluxo de dados no PI AF, vejamos na figura 45.

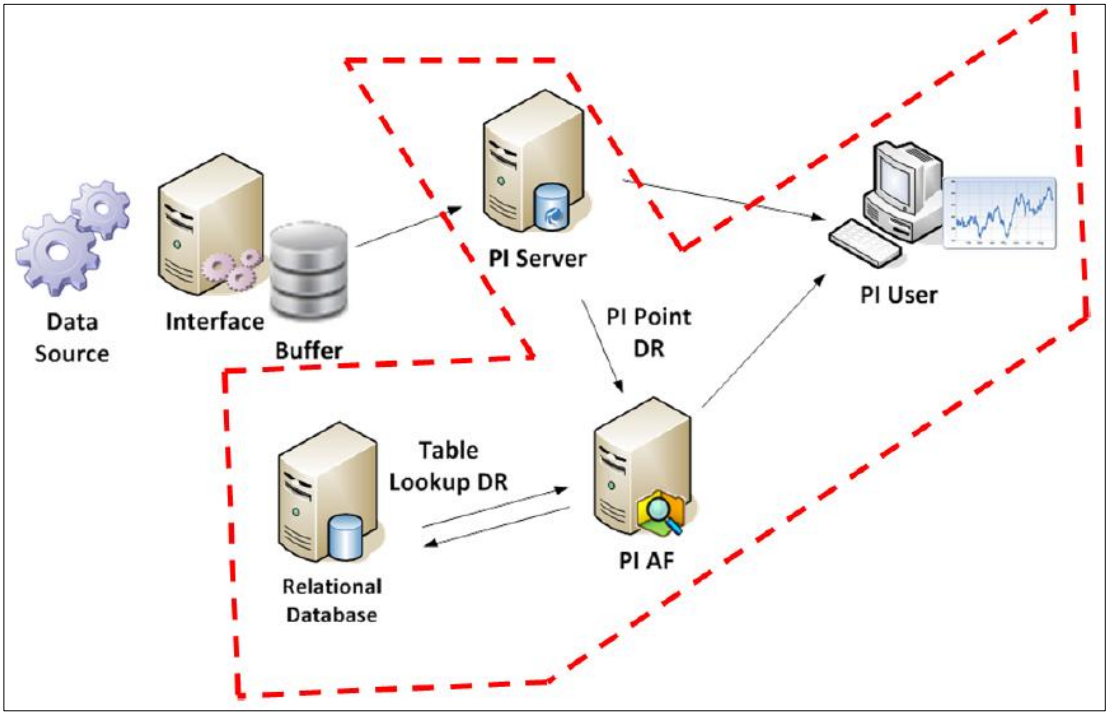


Figura 45 – Fluxo de dados no PI AF
Fonte: OSISoft (2012)

Segundo (OSISOFT, 2012), neste mecanismo o PI AF não precisa estar conectado diretamente a fonte de dados. Os dados das *tags* continuarão sendo armazenados no PI Server; o PI AF coleta os dados do PI Server e os disponibiliza ao PI User. O Servidor PI AF também coleta dados do Banco de Dados Relacionais, permitindo que o usuário visualize esses dados junto com os dados da *tag*.

4.10.8 Ferramentas do PI System

O PI System disponibiliza ferramentas que levam os dados do processo, para acesso, visualização e análise. É um conjunto de ferramentas completas e configuráveis de fácil utilização, podendo ser utilizado em qualquer horário e ambiente de trabalho.

Vejamos as ferramentas disponíveis para uso e suas finalidades:

- Acesso a dados a partir de qualquer local: tablet, celular, web (PI Vision);
- Visualização de processos e desenvolvimento de telas(PI ProcessBook);
- Visualizar e interagir com os displays do PI ProcessBook sem modificação (PI ActiveView);
- Exportar para o Excel (PI DataLink);
- Criar dashboards com SharePoint (PI WebParts);
- Registrar dados manualmente onde estiver (PI Manual Logger);
- Informações do quadro de bateladas (PI BatchView).

Mais especificamente, será abordado as seguintes ferramentas para o projeto em estudo: PI ProcessBook, PI Builder e PI ActiveView.

4.10.8.1 PI ProcessBook

PI ProcessBook é uma interface gráfica, de fácil utilização que permite:

- Exibir dados atuais e históricos do processo/planta;
- Criar animações gráficas;
- Criar diversos displays (telas), organizando por tipo de produção ou monitoramento;
- Escrever scripts usando o Microsoft Visual Basic, automatizando as telas;
- Dados de reprodução para analisar eventos segundo a segundo;
- Realização de relatórios.

O ProcessBook é um conjunto de displays ou telas individuais de dados e informações. Usado para organizar as informações/dados de um PI System e diversas fontes, de forma que se possa monitorar o processo ou as tarefas que realiza. Um

ProcessBook e seus displays são armazenados em um único arquivo (OSISOFT, 2012).

Segundo (GENENA, 2004), o PI ProcessBook (PI-PB) é uma ferramenta que permite ao desenvolvedor construir e visualizar telas de processos, gráficos e valores das variáveis envolvidas da empresa. Essa ferramenta é de simples utilização e de forma amigável, que permite a elaboração de vários displays/telas, informando os dados em tempo real.

Segundo (SCHEUER, 2004), o PI-PB é uma aplicação para exibição de dados armazenados no PI Archive de um processo de uma empresa ou de qualquer outra fonte de dados. Permite a elaboração de gráficos dinâmicos e desenvolvimento de displays de processo para monitoramento online. Incorpora o Microsoft Visual Basic for Aplications (VBA), possibilitando a automação e simplificação de rotinas e tarefas especiais.

O PI-PB permite que os usuários criem exibições gráficas enriquecendo-as com dados robustos e dinâmicos. Projetado para PC, suporta análise de alto nível e profundo de dados, ajuda os usuários a acessar e visualizar instantaneamente os dados do PI Server por meio de exibições gráficas interativas que podem ser simultaneamente preenchidas com dados ativos, anos de dados históricos e dados preditivos e de previsão. Uma vez criado os displays, tem-se a possibilidade de compartilhar e ser aproveitado em qualquer ambiente da empresa (OSISOFT, 2018).

O display do PI-PB pode conter dados/informações do PI System de uma ou todas as seguintes fontes:

- PI Server;
- PI Asset Framework;
- Cálculos realizados no PI;
- Banco de dados.

4.10.8.2 Display

O Display é a principal unidade para a realização de projetos gráficos dentro do PI-PB. A extensão do arquivo do display é o “**.**pdi”, na qual faz parte do PI-PB que tem a extensão “**.**piw”. O display contém uma biblioteca enriquecida de símbolos, que representa os mais diversos equipamentos (motores, válvulas, bombas, silos, tanques

e etc) em qualquer ambiente de uma empresa e permite também a criação de novos símbolos ou a importação de uma figura, permitindo ao desenvolvedor, criar telas o mais parecido com o físico da empresa e sem limites para inovações (OSISOFT, 2015).

4.10.8.3 PI Builder

Este software é um suplemento do PI System que permite usar o Excel para criar, visualizar e modificar objetos no banco de dados do PI Asset Framework (PI AF) ou no PI Data Archive, onde torna o trabalho de configuração de *tags* mais prático e dinâmico. Com o PI Builder, é possível realizar diversas alterações em massa com facilidade e fazer as seguintes tarefas (OSISOFT, 2015):

- Criar, editar e excluir *tags* no PI Data Archive ou editar os atributos desses pontos;
- Criar várias cópias de objetos existentes no PI AF (elementos e atributos, categorias, conjuntos de enumeração, modelos, tipos de referência, estruturas de eventos, transferências, portas, valores de enumeração e propriedades estendidas);
- Copiar objetos do PI AF. Em caso de necessidade de criar vários objetos semelhantes, existe a opção de criar apenas um e replicar essas informações utilizando o PI Builder;
- Alterar configurações de objetos existentes.

4.10.8.4 PI ActiveView

Permite acesso remoto às telas do PI-ProcessBook via *browser* do *Internet* Explorer. É uma ferramenta de visualização, sua principal função é compartilhar as telas ou displays criados no PI-PB, em qualquer unidade da empresa. Esse complemento do PI System, exibe os dados armazenados no PI Server e as variáveis coletadas de forma online.

Segundo (SCHEUER, 2004), o PI ActiveView é um complemento do PI System, uma ferramenta para visualização de dados instantâneos e históricos coletados de

diversas fontes. Permite a visualização das informações do processo armazenados no PI Archive. Os displays, uma vez desenvolvido, podem ser visualizados no PI-ActiveView através do *Internet explorer* ou rede corporativa.

Segundo (OSISOFT, 2018), o PI ActiveView, fornece meios de visualização ao se interagir com os displays do PI ProcessBook fora do PI ProcessBook. Ao utilizar o ActiveView em outros aplicativos, como o *Internet Explorer*, e instalar o executável local, todos os usuários com acesso a rede na empresa, poderão visualizar os arquivos de exibição "**.**pdi" do PI ProcessBook sem alteração.

4.10.9 Servidor do PI System

Devido à alta complexidade de se configurar e obter um servidor do PI System, foi decidido aplicar o presente projeto em uma empresa onde já existe um servidor pré configurado.

O servidor PI é o coração do PI System, que coleta, armazena e organiza dados de todas as fontes de dados, fornecendo uma infraestrutura de informações poderosa e flexível. O servidor também inclui ferramentas sofisticadas para análise, alerta, entre outros, que pode ser conectado a praticamente qualquer sistema de automação, laboratório ou informação existente. Operadores, engenheiros, gerentes e outros funcionários da empresa podem usar vários aplicativos para se conectarem ao servidor para exibir dados armazenados no PI (OSISOFT, 2015).

5 MATERIAIS E MÉTODOS

O protótipo do estudo realizado foi montado em uma determinada empresa com a finalidade de coletar os dados (umidade relativa do ar, temperatura ambiente e corrente elétrica) de um painel de inversor de frequência responsável por controlar a velocidade do motor de uma correia transportadora, a fim de se poder gerenciar sua manutenção. Dentro deste tópico serão apresentadas as configurações, montagens e desenvolvimento do mesmo.

5.1 Metodologia geral

A metodologia utilizada poderá ser aplicada em diversos equipamentos de diferentes empresas, de tal forma que o tema proposto possa ser o mais amplo possível. Portanto, para que isso seja possível, todos os materiais e equipamentos utilizados no projeto foram analisados para que a aplicação em diferentes ambientes seja possível e que a necessidade de manutenção de diferentes equipamentos seja atendida.

O método de análise de falha foi feito de forma preditiva, utilizando sensores, *hardwares* e *softwares*. Sendo assim, um programa que englobe as necessidades de interface de comunicação e as aplicações necessárias foi o principal foco de pesquisa.

Além do *software*, foi necessário a utilização de sensores com a finalidade de monitorar as informações mais relevantes de um determinado equipamento, no caso, foram monitoradas a umidade do ar, temperatura e corrente elétrica.

Para realização da interface entre os sensores e os *softwares utilizados*, foi necessário a utilização do controlador Arduino que possui entradas específicas para os diferentes tipos de variáveis que foram monitoradas.

Foram adquiridas informações em livros e outras monografias que utilizam alguma das tecnologias que estamos propondo. Por fim, foram realizados simulações e testes para obtenção do resultado final.

5.2 Desenvolvimento

A seguir serão apresentadas todas as etapas de configurações tanto na parte de *hardware* como também os *softwares* utilizados no projeto proposto anteriormente.

5.2.1 Configurações de *hardware* do Arduino

A placa Arduino UNO foi configurada de tal forma que fosse possível estabelecer a comunicação com o CLP. Para isso, foi utilizado o módulo de comunicação Ethernet Shield W5100. Na figura 46 podemos visualizar a montagem do módulo na placa Arduino UNO.

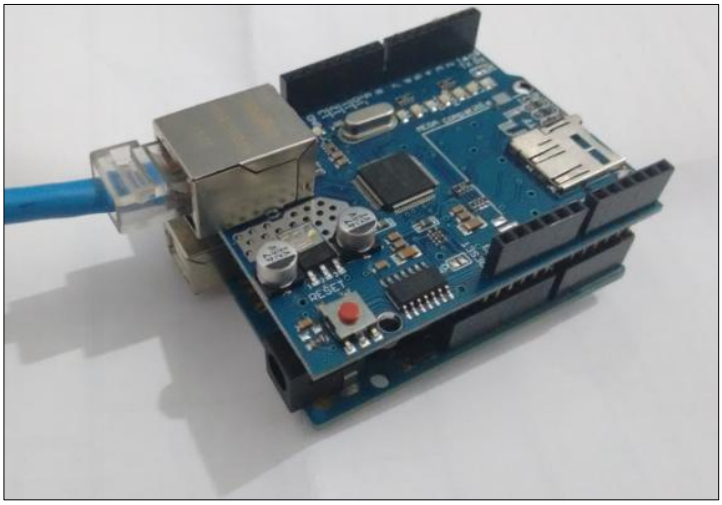


Figura 46 – Montagem do módulo Ethernet Shield W5100
Fonte: Autores (2018)

Para medição de temperatura e umidade do ar, foi configurado o sensor DTH22 na entrada analógica A1 da placa, conforme mostrado na figura 47.

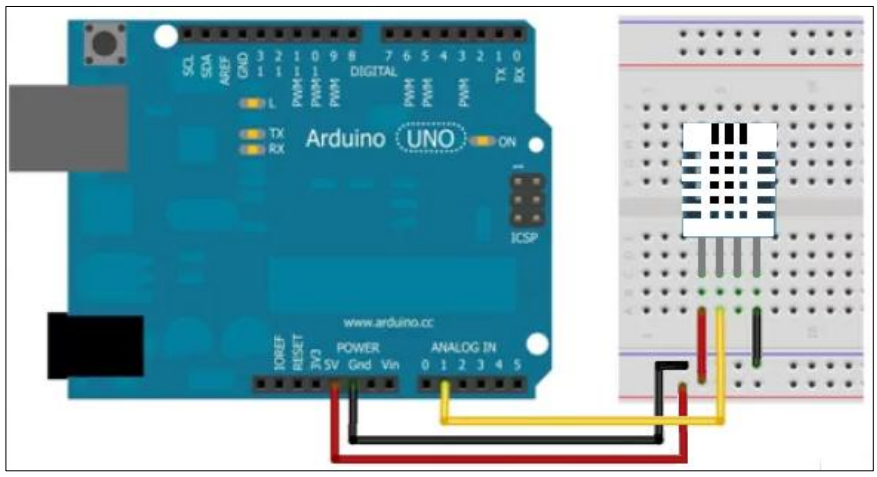


Figura 47 – Sensor DHT22 conectado ao Arduino
Fonte: Autores (2018)

Para obtenção dos valores de corrente elétrica foi realizado a ligação do sensor SCT-013-00 na porta analógica A0, conforme mostrado na figura 48.

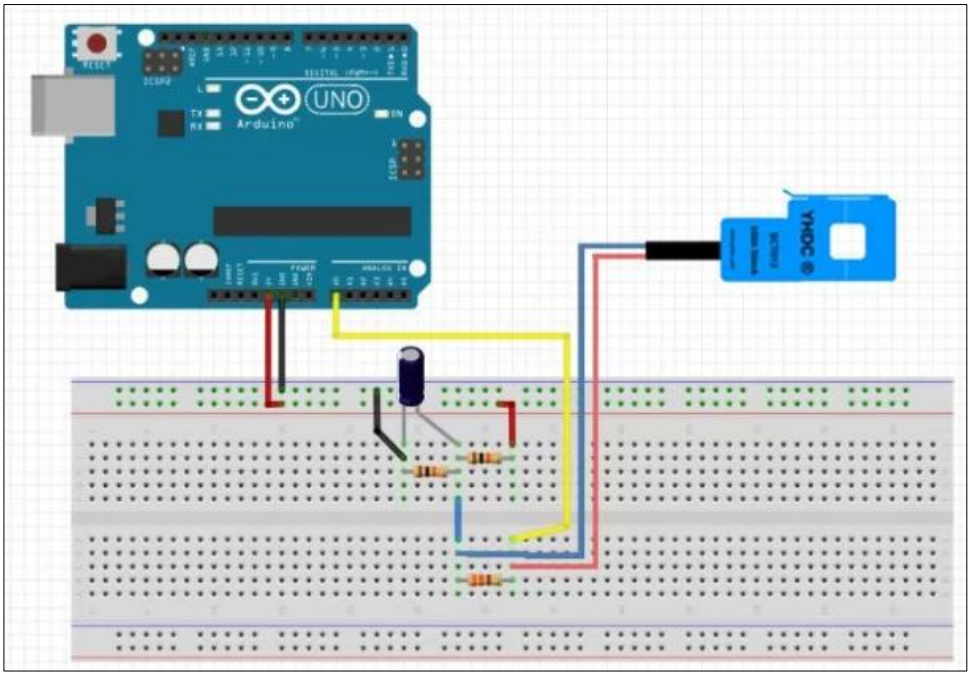


Figura 48 – Sensor SCT-013-00 conectado ao Arduino
Fonte: Vida de Silicio (2018)

Como o sensor de corrente emite sinal de saída com valores entre 0 a 50mA e o Arduino só realiza leitura de sinais de tensão de 0 a 5V, foi necessário instalar um resistor de carga (Rc) a fim de se obter um sinal de tensão (V) para leitura, conforme mostrado na figura 49.

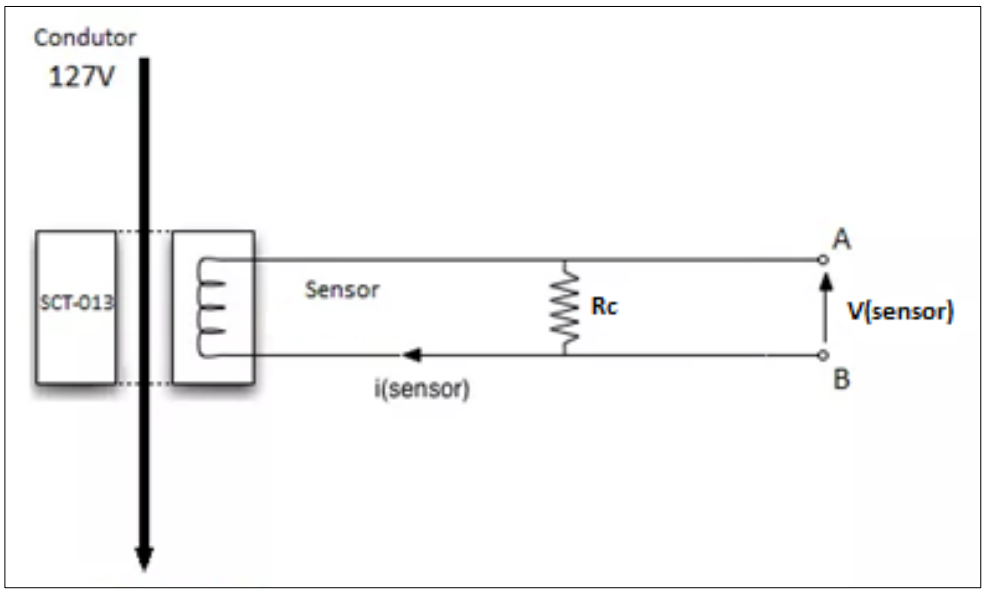


Figura 49 – Resistor de carga no sensor SCT-013-00
Fonte: Vida de Silicio (2018), adaptado pelos autores

Foi decidido utilizar um resistor que disponibilizasse uma forma de onda senoidal com variação de -2,5V a 2.5V, conforme mostrado na figura 50, sendo que esta senóide possui valor de pico a pico igual a 5V permitindo que a mesma seja ajustada posteriormente, para variar de 0 a 5V, possibilitando a leitura do Arduino. O resistor foi calculado através da seguinte equação:

$$R = \frac{V}{i}$$

Sendo, $\mathrm{V} = 2{,}5\mathrm{V}$ e o valor de i (corrente máxima medida) pode ser obtido através da corrente máxima de saída do sensor que é de 50mA. Este é o valor máximo eficaz (RMS) e o valor máximo medido é obtido através da seguinte equação:

$$i(máximo\ medido) = \sqrt{2} * i(RMS)$$

então,

$$i(máximo\ medido) = \sqrt{2} * 0{,}05 = 70{,}7mA$$

Sendo assim, o valor calculado do resistor de carga é apresentado a seguir:

$$R = \frac{2{,}5}{0{,}0707} = 35{,}4\Omega$$

Como o valor de 35,4Ω não é um valor comercial, foi decidido utilizar um resistor de 33Ω que pode ser encontrado facilmente no mercado.

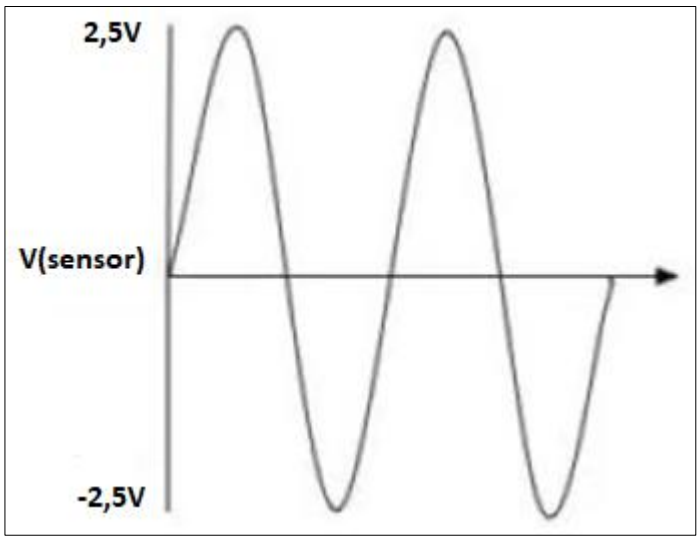


Figura 50 – Forma de onda 1 com o resistor de carga
Fonte: Autores (2018)

O Arduino só permite a leitura de valores analógicos entre 0 a 5V, portanto, além da necessidade de se calcular um resistor de carga adequado, o sinal senoidal de saída do sensor deve estar dentro desta faixa de leitura admissível, para isso foi configurado um circuito divisor de tensão com a finalidade de somar uma determinada tensão contínua à senóide de saída do resistor de carga. Como o resistor de carga calculado anteriormente irá permitir uma senóide com valor máximo de pico igual a 2,5V, será somado mais 2,5V de corrente contínua, deslocando toda a senóide para a parte positiva, obtendo a curva representada na figura 51, que possui variação de 0 a 5V.

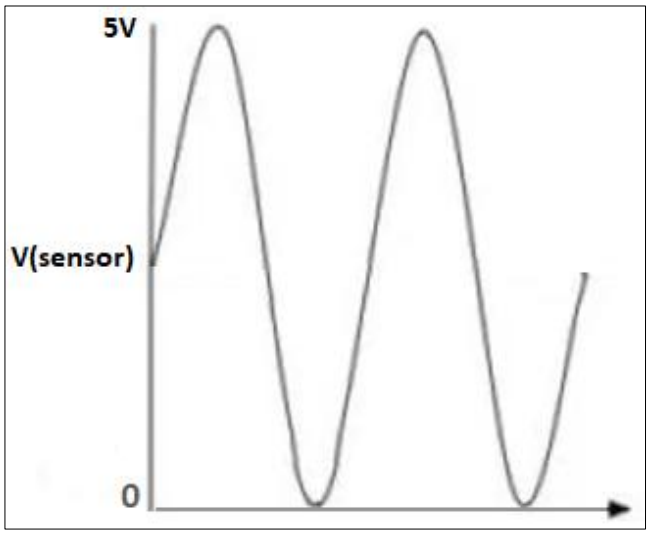


Figura 51 – Forma de onda 2 com o resistor de carga
Fonte: Autores (2018)

Para a obtenção do valor de 2,5V somados à senoide foi configurado o circuito apresentado na figura 52, que consta de um divisor de tensão com dois resistores iguais, permitindo utilizar a tensão de 5V proveniente do Arduíno e dividindo-a por dois obtendo os 2,5V necessários. Foram utilizados dois resistores (R1 e R2) de 10KΩ, por serem facilmente encontrados comercialmente e permitirem um baixo consumo de corrente devido à alta resistência. Além dos resistores, foi ligado um capacitor (C1) de 10µF a fim de garantir uma tensão contínua amenizando pequenas oscilações da tensão de alimentação.

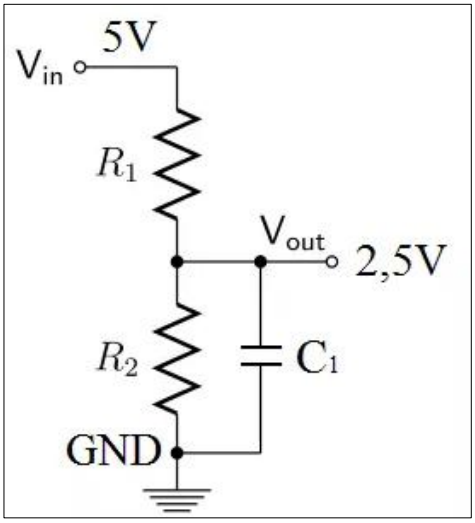


Figura 52 – Circuito divisor de tensão
Fonte: Autores (2018)

A figura 53 representa como ficou o circuito do sensor de corrente SCT-013-00, possibilitando que valores adequados de saída fossem enviados ao Arduino, permitindo a leitura.

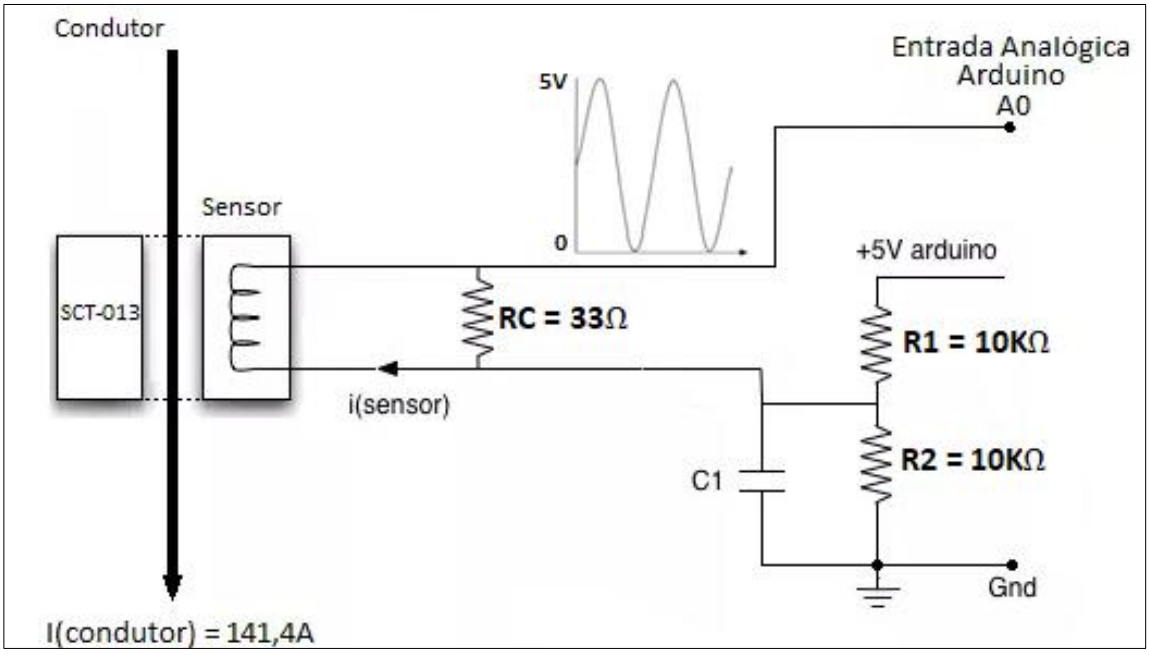


Figura 53 – Cicuito do sensor SCT-013-00
Fonte: Demetras (2018), adaptado pelos Autores

Sendo assim é possível visualizar na figura 54, o esquema com os dois sensores ligados ao Arduíno junto com o módulo Ethernet Shield W5100.

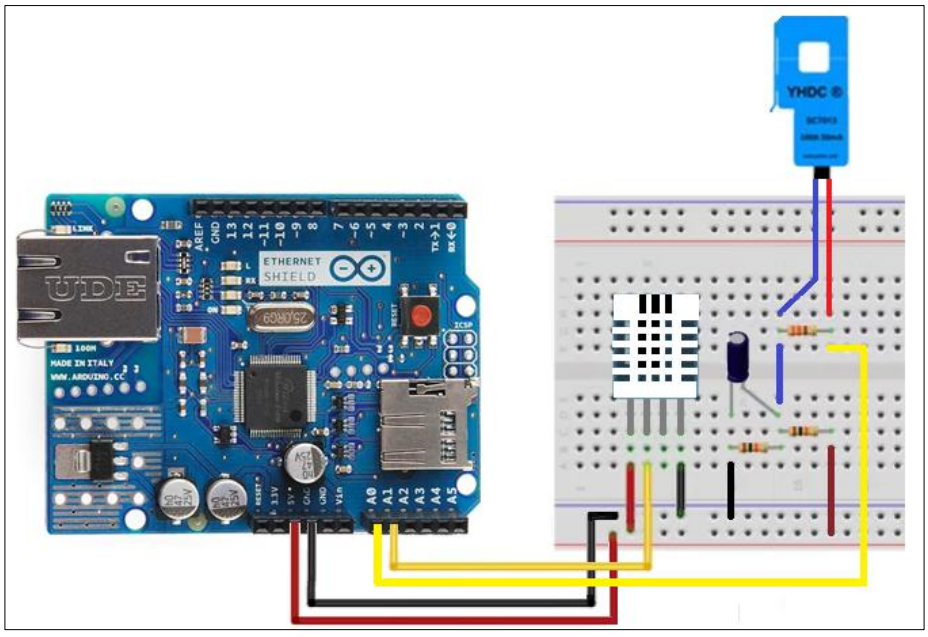


Figura 54 – Esquema dos sensores ligados ao Arduino
Fonte: Autores (2018)

5.2.2 Configurações de *software* do Arduino

No *software* IDE, foram configuradas todas as bibliotecas de programação e também foram definidas todas as entradas a serem lidas do processo. Toda a programação pode ser observada no apêndice 1 deste trabalho. Na figura 55, podemos observar as bibliotecas que foram utilizadas e parte da configuração inicial da programação. O endereço IP utilizado no Arduino foi o 172.26.222.211.

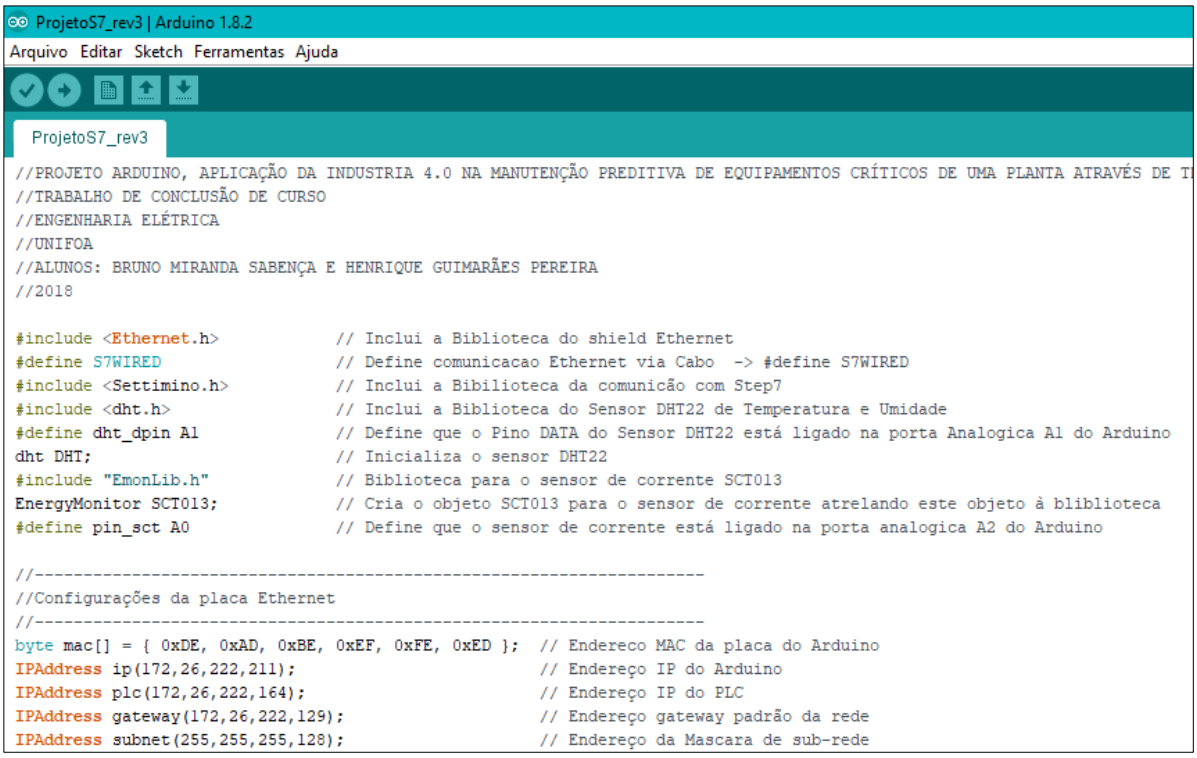


Figura 55 – Programação parcial do Arduino
Fonte: Autores (2018)

5.2.2.1 Sinal de teste de conexão

No Arduino foi configurado um sinal que denominamos como “teste de conexão”. Este sinal tem a finalidade de indicar falhas de comunicação entre o Arduino e toda a interface que está entre ele e o PI System. O “teste de conexão” é uma variável digital, ou seja, pode ter o valor lógico 0 ou 1. Sendo assim, este sinal foi configurado de tal forma que o mesmo fique variando continuamente entre 0 e 1 a partir do momento em que o Arduino é ligado. O tempo em que o sinal permanece em

0 ou 1 é de 500ms. Portanto, caso esta variável permaneça em 0 ou 1 por um tempo maior que 500 ms, podemos dizer que houve algum problema com a interface. Na figura 56 podemos entender melhor o funcionamento deste sinal.

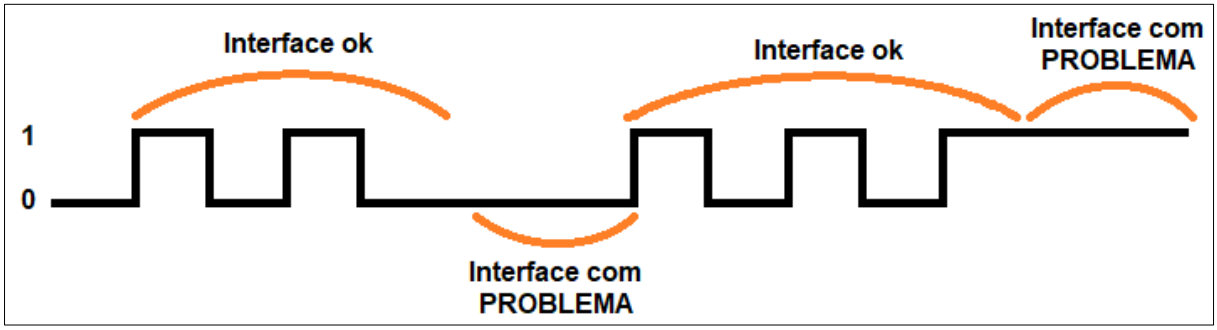


Figura 56 – Sinal de teste de conexão
Fonte: Autores (2018)

5.2.3 Configurações de *hardware* do CLP

A estrutura do controlador foi montada de uma forma simples, conforme mostrado na figura 57, com a finalidade de receber os dados enviados pelo Arduino. Nesta montagem foram utilizados um Rack, uma fonte, uma CPU e um CP de comunicação, todos da série S7-400 da Siemens. Um cabo padrão ethernet foi conectado à CP a fim de estabelecer a comunicação com o Arduino e com os *softwares* via OPC.

Figura 57 – Montagem do CLP
Fonte: Autores (2018)

5.2.4 Configurações de *software* do CLP

Com a necessidade de se estabelecer uma conexão entre o CLP, Arduino e demais *softwares,* foi realizado através do programa Simatic Manager (Step 7), a configuração necessária no controlador S7-400.

Na figura 58, podemos visualizar esta configuração, onde foram inseridos o rack, CPU e CP utilizados. Na parte de configuração da CP, foi inserido o IP utilizado no CLP, sendo este que o utilizado foi 172.26.222.164 e a máscara de rede 255.255.255.128.

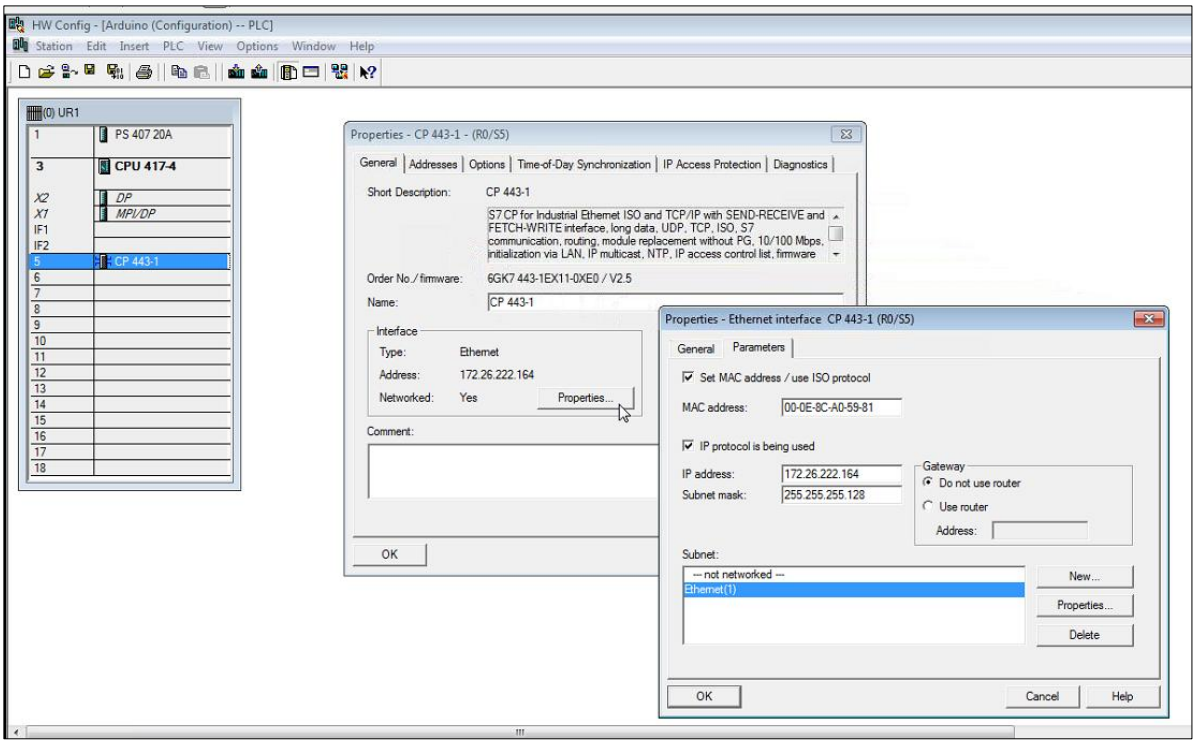
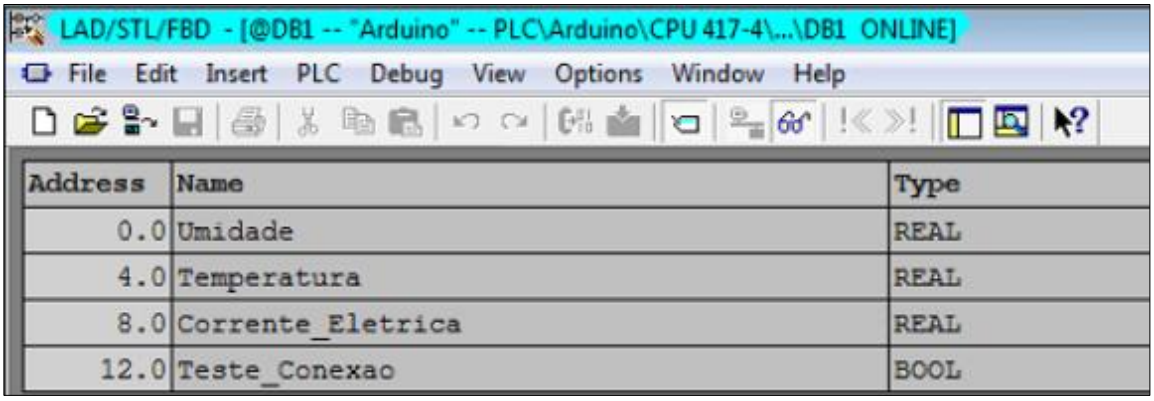


Figura 58 – Configuração do CLP via SIMATIC Manager
Fonte: Autores (2018)

Para armazenar os dados (corrente, temperatura, umidade e teste de conexão) enviados pelo Arduino, foi criado um DB (*Data Base*) que funciona como um banco de dados do projeto, armazenando as informações no CLP conforme mostrado na figura 59.



Address	Name	Type
0.0	Umidade	REAL
4.0	Temperatura	REAL
8.0	Corrente_Eletrica	REAL
12.0	Teste_Conexao	BOOL

Figura 59 – *Data Base* (DB) do projeto
Fonte: Autores (2018)

5.2.5 Configurações do PI System

Para gerenciamento das variáveis (umidade do ar, temperatura e corrente elétrica) via e-mail e tela gráfica, foram configuradas algumas ferramentas do *software* PI System. Tais configurações serão apresentadas dentro deste tópico

5.2.5.1 Liberação de acesso ao PI System

Antes de iniciar todas as devidas configurações, foi necessário criar um ponto de liberação denominado *trust,* no *software* de administração no PI System para permitir a leitura dos dados disponíveis no computador responsável por transmitir as variáveis do processo. O IP definido previamente no computador foi o 172.26.222.210.

Na figura 60 é possível observar a configuração de liberação do computador.

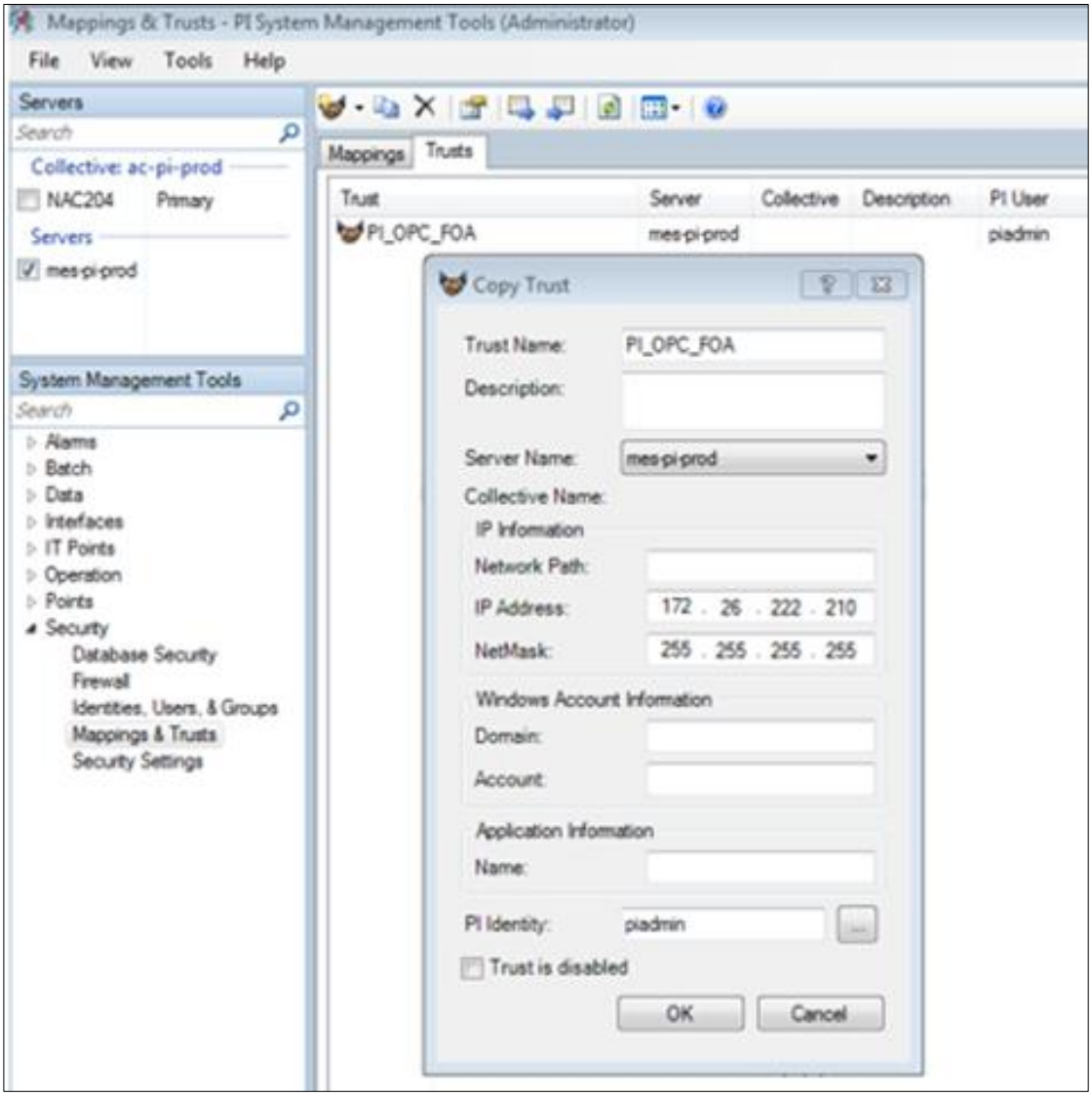


Figura 60 – Liberação de acesso ao PC
Fonte: Autores (2018)

5.2.5.2 Interfaces OPC do PI System

O envio de dados ao PI System é feito através da comunicação OPC, portanto, foram configurados um servidor OPC e um Cliente OPC.

O servidor OPC foi obtido através do *software* KepServerEX. Neste programa foi necessário configurar o endereço IP do CLP onde está sendo feita a leitura das variáveis, como podemos ver na figura 61, e também foi configurado o endereço (*tag*) das variáveis a serem lidas do CLP.

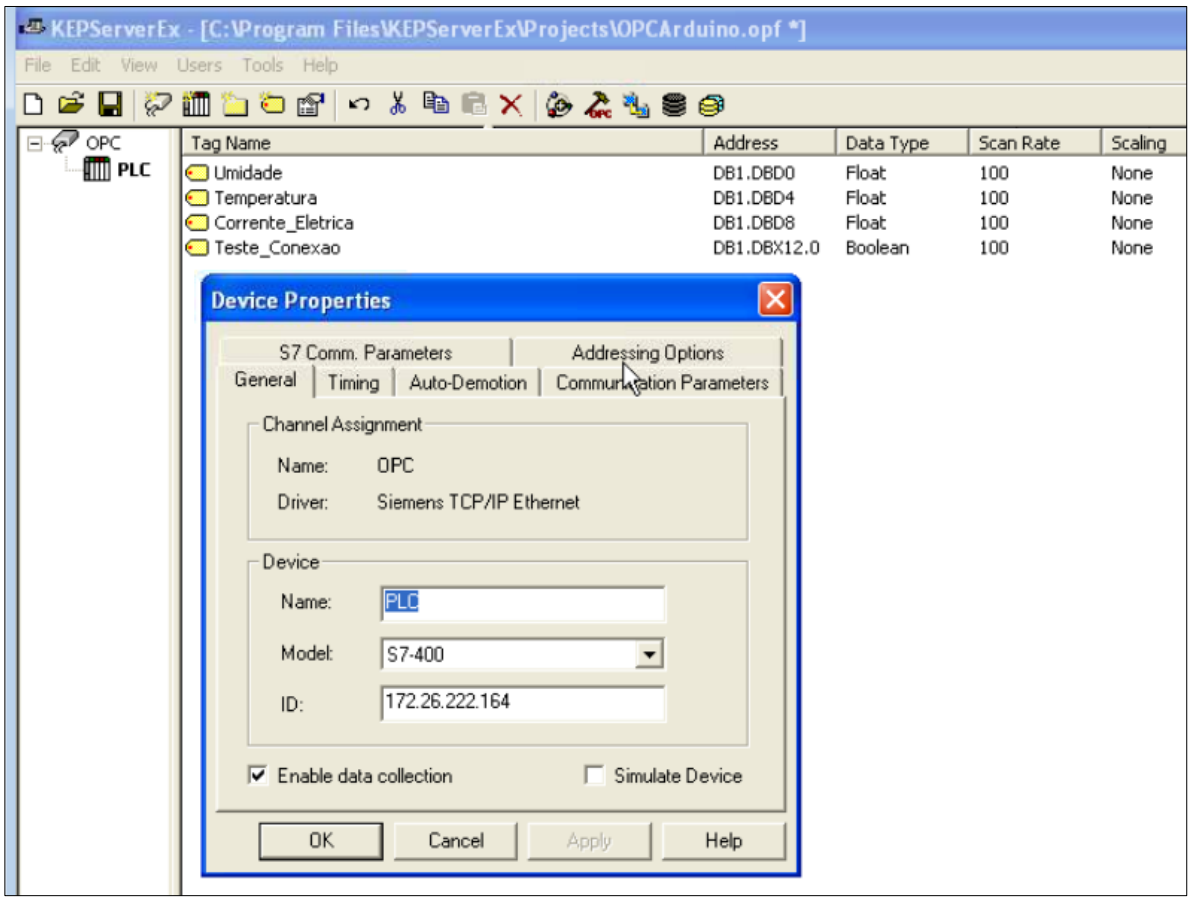


Figura 61 – Configuração do servidor OPC
Fonte: Autores (2018)

Na figura 62 podemos observar o endereço (DB1.DBD0) da variável “umidade” referente ao primeiro byte da DB1, como exemplo.

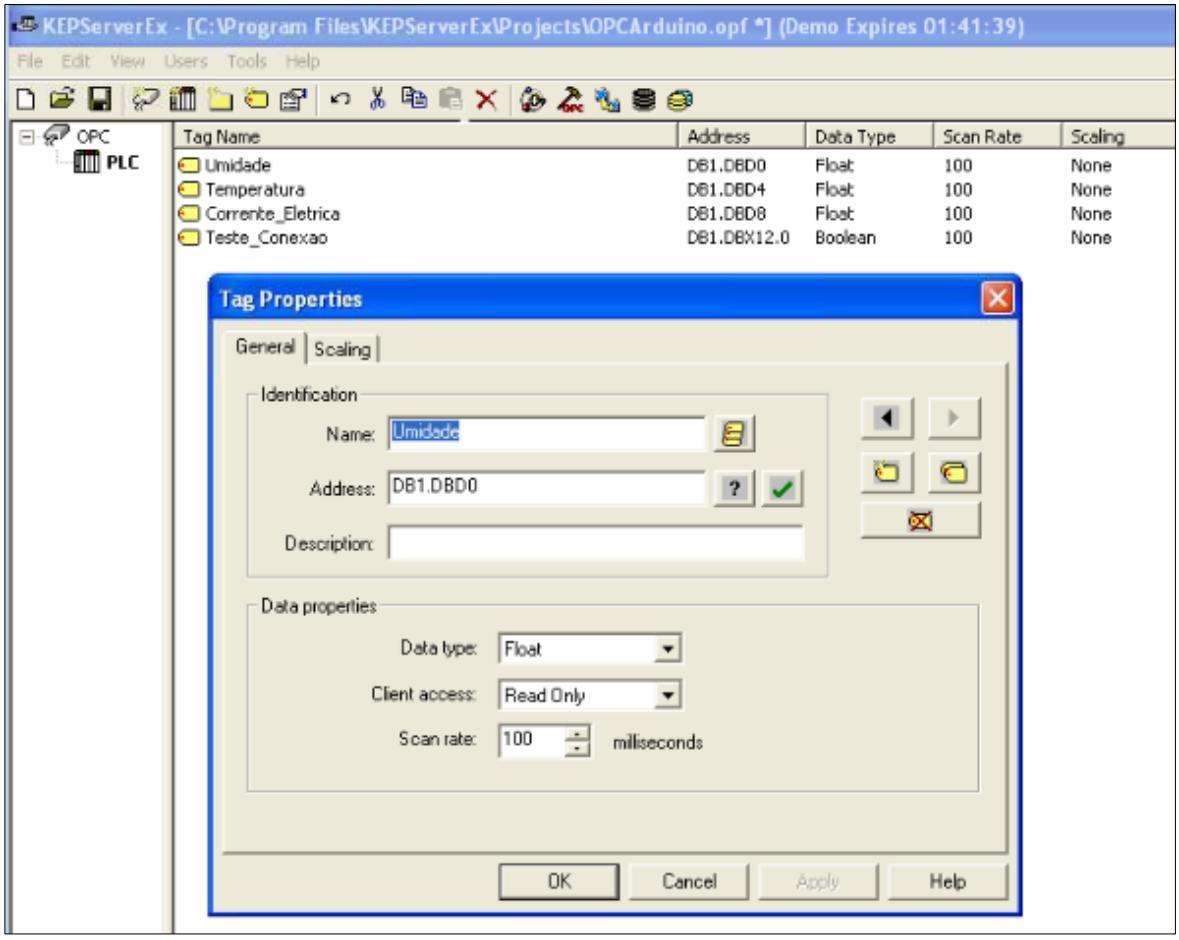


Figura 62 – Configuração de variável no servidor OPC
Fonte: Autores (20418)

Após configurado o servidor OPC, as variáveis se tornaram disponíveis ao cliente OPC.

Portando, em seguida foi instalado o *software* PI OPC, figura 63, a fim de se estabelecer a interface entre as informações enviadas pelo servidor e as ferramentas do PI System.

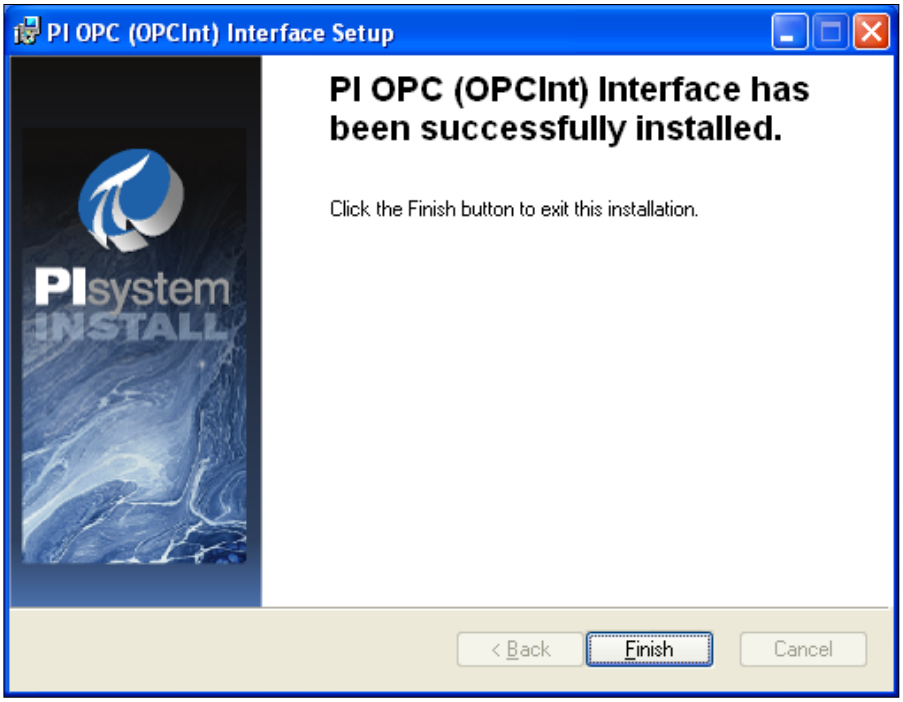


Figura 63 – *Software* PI OPC
Fonte: Autores (2018)

Após instalado o *software* PI OPC, foi necessário configurar um arquivo de texto denominado “OPCInt.bat”, disponível na pasta *“interfaces”* que é gerada automaticamente após a instalação do *software*. Na figura 64 é possível observar os dados que foram inseridos no arquivo de texto, onde:

a) Comentários;

b) *Point source* ou ponto de origem, para poder localizar as *tags* posteriormente;

c) Nome do servidor OPC;

d) Nome do servidor principal do PI Sytem;

e) Classes de *scan*, criadas para serem utilizadas posteriormente para informar de quanto em quanto tempo o *status* de uma variável deve ser atualizada. Podem ser criadas várias classes de *scan.*

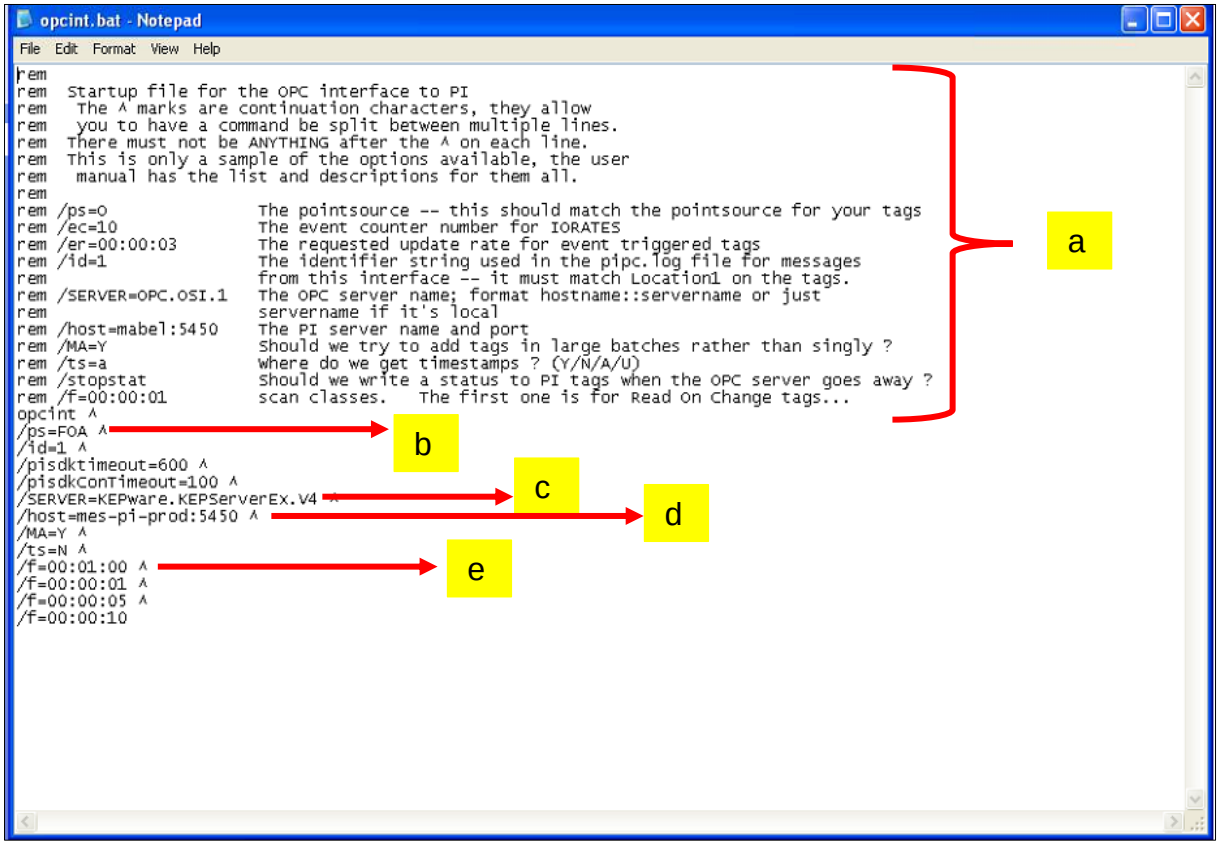


Figura 64 – Arquivo de texto do PI OPC
Fonte: Autores (2018)

5.2.5.3 Instalação do PI SDK

O PI SDK estabelece a interface entre os dados obtidos pelo OPC Client e a rede do servidor do PI System. Sendo assim, é necessário a configuração do mesmo, utilizando os dados do servidor do PI. Na figura 65, podemos observar a imagem ao fim da instalação do PI SDK.

Figura 65 – PI SDK
Fonte: Autores (2018)

5.2.5.4 Configuração das variáveis no PI System

Através da ferramenta PI Builder, foram configuradas via Excel, as *tags* das variáveis a serem utilizadas no PI System. A tabela 3 mostra como ficou esta configuração, sendo que a função de cada coluna é descrita abaixo:

- Name – *Tag* criada para ser usada nas ferramentas do PI System;
- Description – Descrição breve da variável a ser lida;
- Eng units – Unidade de engenharia da variável;
- Point source – Nome do ponto de origem das *tags* configuradas no Cliente OPC;
- Point type – Tipo do valor lido da variável (Float32 = Variável com valor real, INT32 = Variável com valor inteiro);
- Location4 – Número da classe de scan a ser utilizada na leitura.

Tabela 3 – Configuração das *tags* no PI System

Name	Description	engunits	pointsource	pointtype	instrumenttag	location4
PROJ.ARDUINO_TEMPERATURA	PROJETO ARDUINO- TEMPERATURA	ºC	FOA	Float32	OPC.PLC.Temperatura	3
PROJ.ARDUINO_UMIDADE	PROJETO ARDUINO- UMIDADE	%	FOA	Float32	OPC.PLC.Umidade	3
PROJ.ARDUINO_CORRENTE_ELETRICA	PROJETO ARDUINO- CORRENTE_ELETRICA	A	FOA	Float32	OPC.PLC.Corrente_Eletrica	3
PROJ.ARDUINO_TESTE_CONEXAO	PROJETO ARDUINO- TESTE_CONEXAO		FOA	int32	OPC.PLC.Teste_Conexao	3

Fonte: Autores (2018)

Além das *tags* relacionadas às variáveis do Arduino, foi necessário criar *tags* internas no PI System para geração do relatório diário. As informações disponíveis no relatório são: Valor máximo e média de cada variável ao longo do dia e tempo de operação do equipamento durante o dia. Portanto, na tabela 4, é possível observar as respectivas *tags* com os scripts (conjunto de instruções para que uma função seja executada) desenvolvidas no PI Buider.

A coluna “exdesc” corresponde ao local onde é escrito o script de cada *tag*. Os scripts que correspondem as funções de valor máximo e média de uma *tag* devem ser escritos da seguinte forma:

Função(‘tag’, ‘x’, ‘x2’)

Sendo que x equivale ao horário final do período que foi analisado para gerar o relatório e x2 é o horário inicial. Nos scripts da tabela 4, foram utilizados ‘t-1h’ e ‘y-1h’ em todas as *tags*. Isso significa que a função será executa no período de 23:00h de dois dias atrás até às 23:00h do dia anterior do envio do relatório (t = 00:00h do dia atual, y = 00:00h do dia anterior. Ambos estão sendo subtraídos menos 1 hora o que corresponde a 23:00h).

Cada script possui uma função específica sendo elas descritas abaixo:

TAGMAX: Valor máximo lido de uma determinada *tag* em determinado período de tempo;

TAGAVG: Valor médio obtido de uma determinada *tag* em determinado período de tempo;

TIMEGE: Tempo decorrido a partir do momento em que o valor de uma determinada *tag* ultrapassa um valor de referência.

No script que contabiliza o tempo de operação do equipamento (função TIMEGE) há uma diferença em relação ao demais, pois é necessário indicar o valor de referência ou comparação. No caso em questão, foi utilizado o valor igual a 5 como referência, ou seja, caso o valor da *tag* de corrente elétrica seja maior que 5, a função

irá começar a contabilizar o tempo, indicando quanto tempo o equipamento está ligado. Esta função retorna o valor em segundos, portanto há uma divisão por 60 no final do script a fim de transformar para minutos.

Tabela 4 – *Tags* internas do PI System

Name	Description	engunits	exdesc	pointtype
PROJ.ARDUINO_CORRENTE_ELETRICA_MAX	PROJ - ARDUINO CORRENTE ELETRICA MAX	A	TAGMAX ('PROJ.ARDUINO_CORRENTE_ELETRICA','t-1h','y-1h')	Float32
PROJ.ARDUINO_CORRENTE_ELETRICA_MED	PROJ - ARDUINO CORRENTE ELETRICA	A	TAGAVG ('PROJ.ARDUINO_CORRENTE_ELETRICA','t-1h','y-1h')	Float32
PROJ.ARDUINO_TEMPERATURA_MAX	PROJ - ARDUINO TEMPERATURA MAX	ºC	TAGMAX ('PROJ.ARDUINO_TEMPERATURA','t-1h','y-1h')	Float32
PROJ.ARDUINO_TEMPERATURA_MED	PROJ - ARDUINO TEMPERATURA MED	ºC	TAGAVG ('PROJ.ARDUINO_TEMPERATURA','t-1h','y-1h')	Float32
PROJ.ARDUINO_UMIDADE_MAX	PROJ - ARDUINO UMIDADE MAX	%	TAGMAX ('PROJ.ARDUINO_UMIDADE','t-1h','y-1h')	Float32
PROJ.ARDUINO_UMIDADE_MED	PROJ - ARDUINO UMIDADE MED	%	TAGAVG ('PROJ.ARDUINO_UMIDADE','t-1h','y-1h')	Float32
PROJ.ARDUINO_TEMPO_OPERACAO	PROJ. ARDUINO - TEMPO OPERACAO	min	TIMEGE('PROJ.ARDUINO_CORRENTE_ELETRICA','t-1h','y-1h',5)/60	Float32

Fonte: Autores (2018)

Após serem criadas, as *tags* foram inseridas no banco de dados do PI AF através do *software* de desenvolvimento do PI System, o PI System Explorer. Vemos na figura 66, que as *tags* foram inseridas na parte denominada *elements* podendo ser utilizadas posteriormente para realizar a notificação via e-mail.

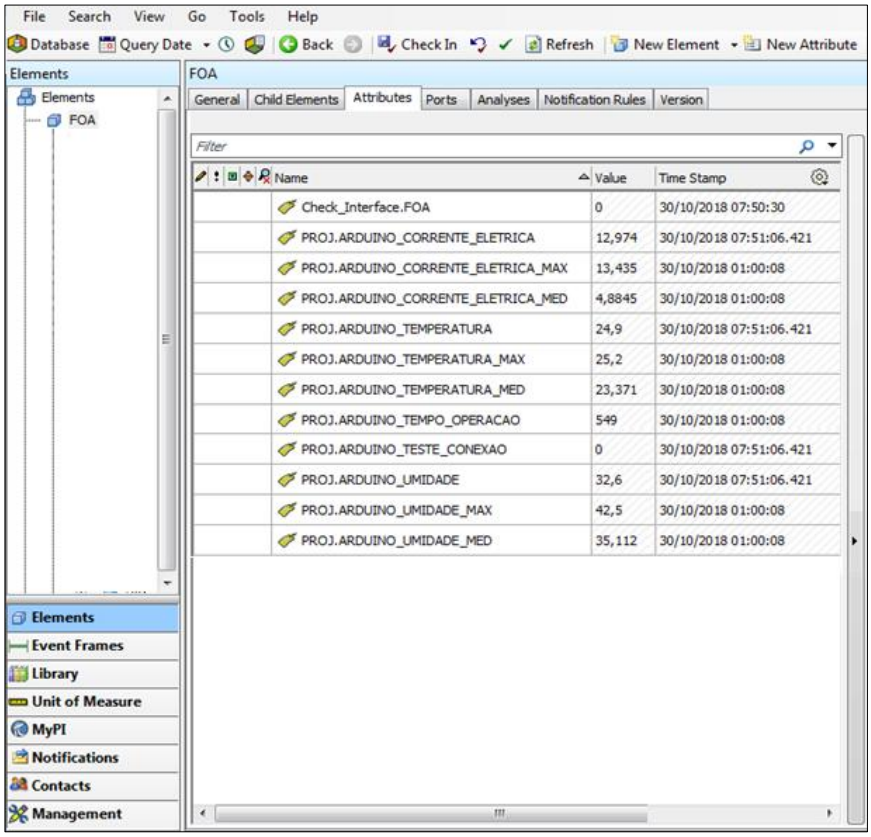


Figura 66 – *Tags* do PI System
Fonte: Autores (2018)

5.2.5.5 Configuração dos e-mails de alerta de falha e histórico diário

No *software* PI System Explorer, na parte denominada *Notifications* foi realizado a configuração para envio instantâneo de e-mails de alertas e também um relatório diário com o histórico dos valores máximo e média das variáveis nas últimas 24 horas.

Caso o valor atual de alguma das variáveis (temperatura, corrente ou umidade) ultrapasse um respectivo valor de referência, um e-mail de alerta será enviado instantaneamente. Na figura 67 vemos a configuração da condição para disparo do e-mail de alerta de temperatura onde o valor de referência utilizado foi de 30°C. O mesmo foi feito para as demais variáveis, corrente e umidade e o valor de referência de cada um foi 15A e 55% respectivamente, equivalente a 15% a mais que o valor máximo medido em condições normais durante 1 mês aproximadamente, assim como o valor de referência de temperatura.

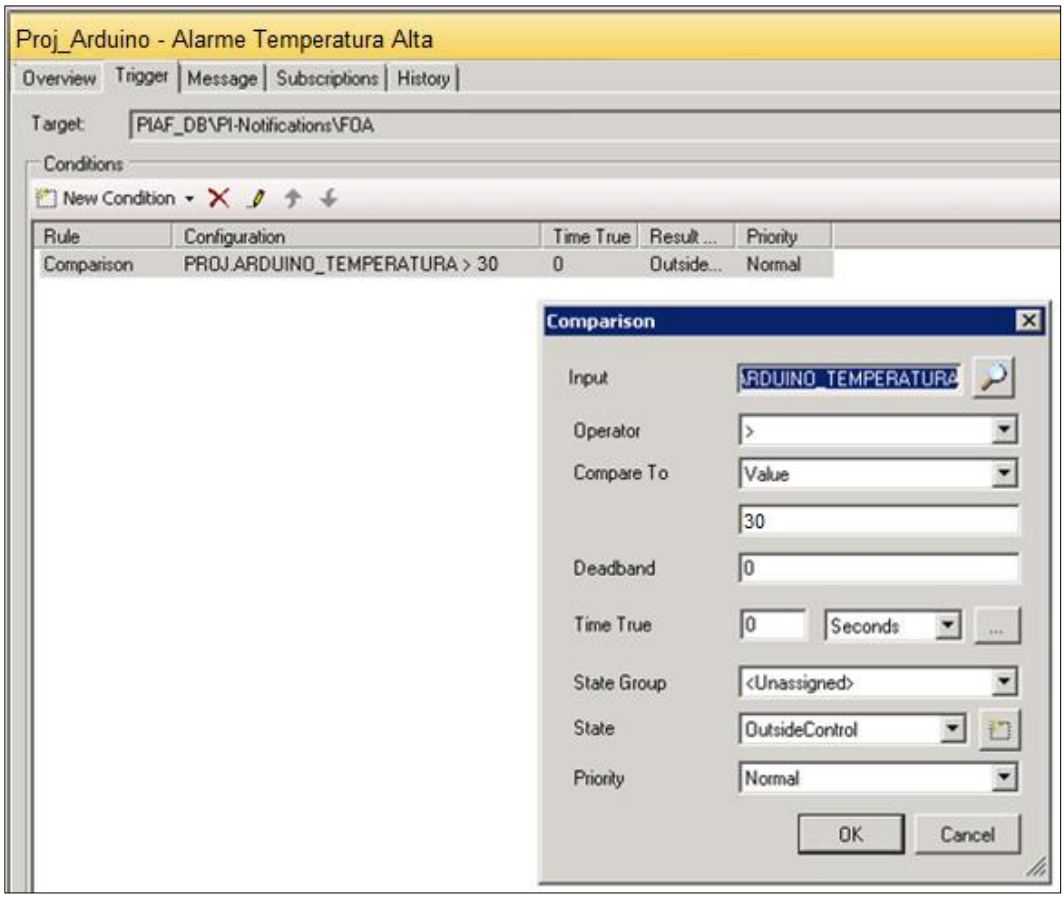


Figura 67 – Configuração da condição de disparo do e-mail de alerta de temperatura
Fonte: Autores (2018)

Após configurado a condição de envio do e-mail para cada variável, a mensagem do e-mail foi escrita conforme mostra a figura 68, com o título do e-mail na parte denominada *subject* e o corpo do e-mail na parte denominada *body.* O mesmo foi feito para as demais variáveis.

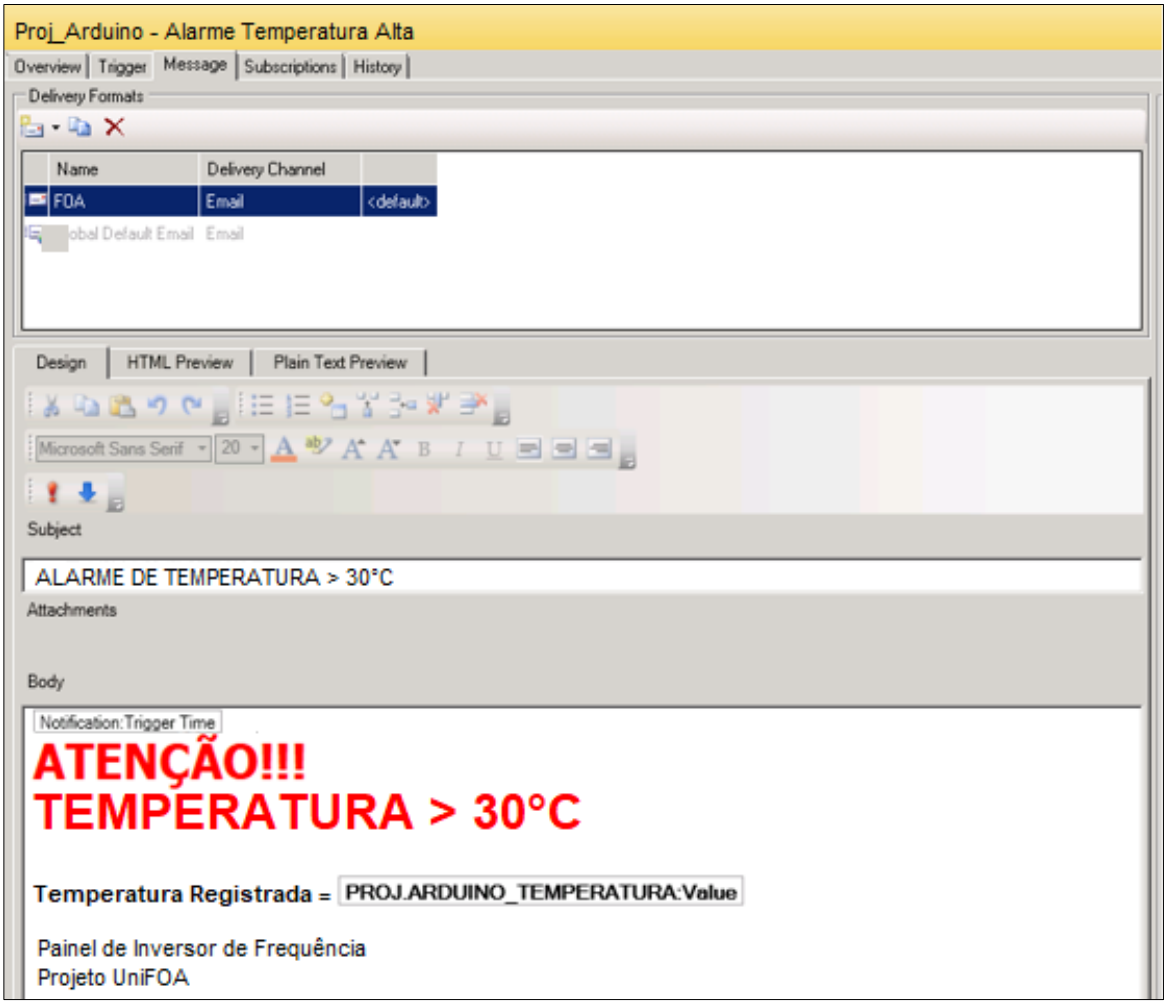


Figura 68 – Corpo do e-mail de alerta de temperatura
Fonte: Autores (2018)

Conforme mostra a figura 69, na aba denominada *subscripitions* foram inseridos os e-mails dos destinatários que no caso em questão foram os autores deste trabalho. O mesmo foi feito em todos os e-mails de alerta e relatório diário.

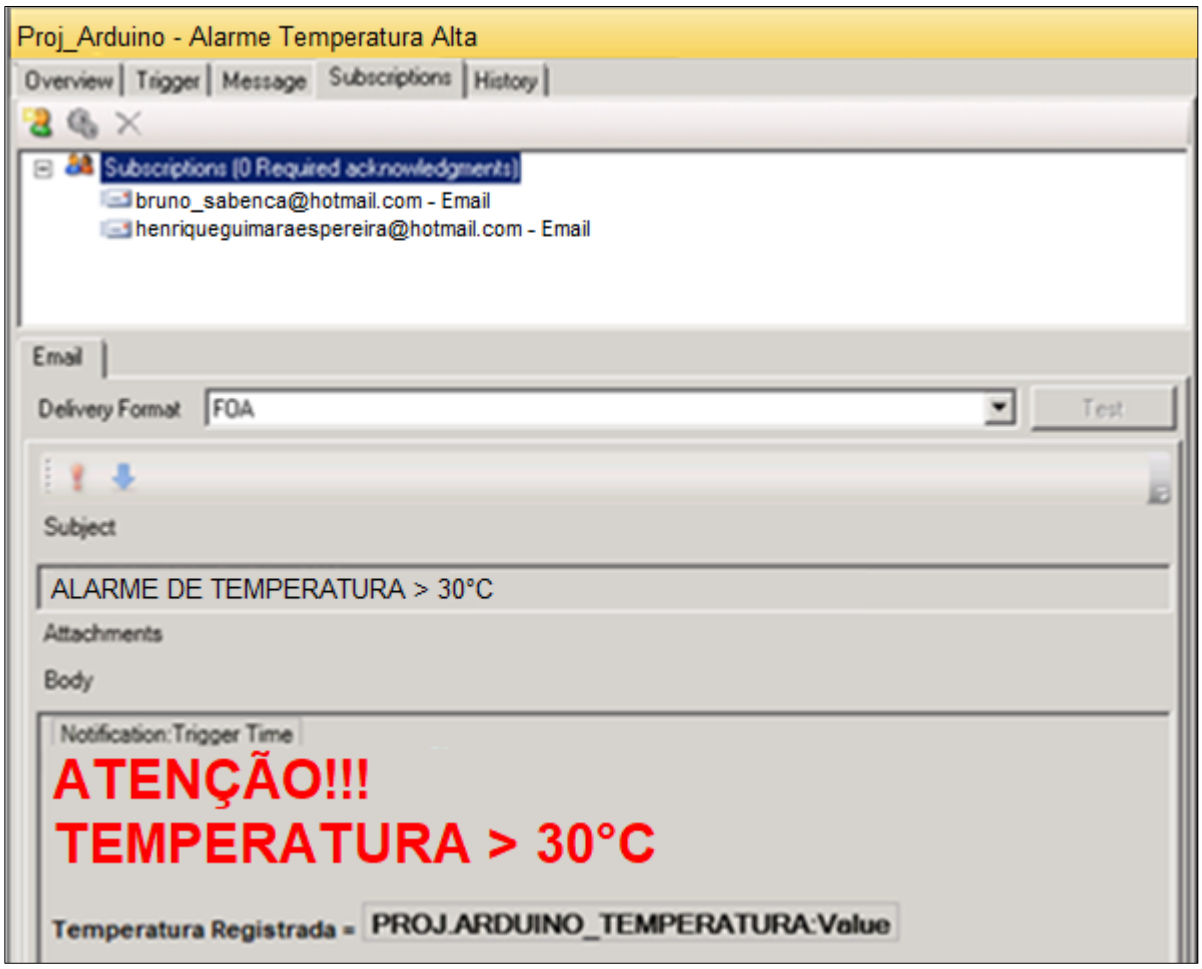


Figura 69 – Destinatários do e-mail de alerta de temperatura
Fonte: Autores (2018)

Outro e-mail que foi configurado, foi o que indica que houve perda de comunicação e os dados não podem mais ser recebidos pelo PI System. Este e-mail tem a finalidade de indicar que há algum problema na interface entre os sinais lidos pelo Arduino e o PI System, sendo necessário a verificação de todo o sistema.

Para disparo deste e-mail foi utilizado o sinal digital de “teste de conexão” enviado pelo Arduino, que normalmente fica variando entre sinal 0 e 1. Quando esta oscilação de sinal para de acontecer por 1 minuto, um e-mail conforme mostra a configuração da figura 70, será enviado instantaneamente.

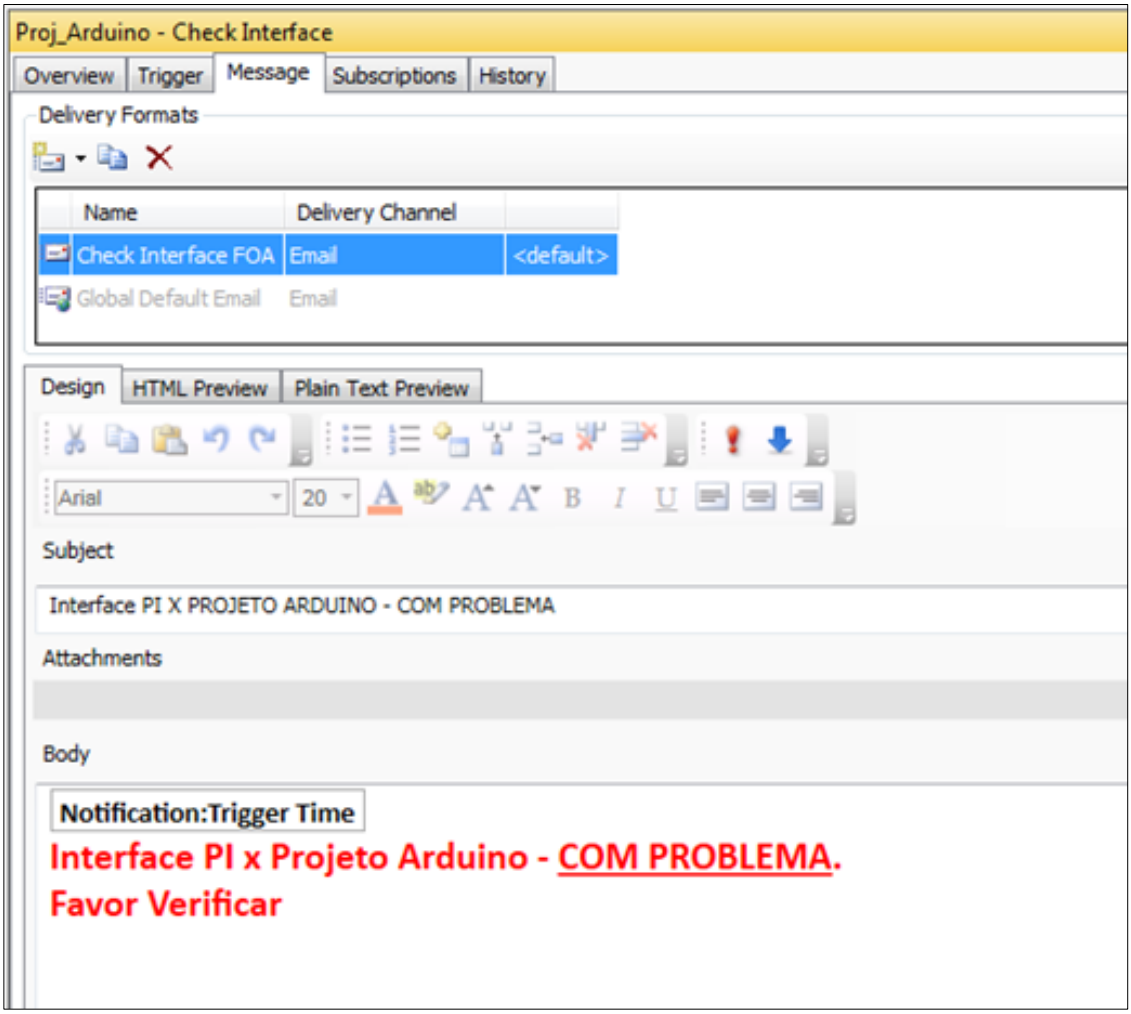


Figura 70 – Configuração do e-mail de falha de comunicação
Fonte: Autores (2018)

Caso o sinal de teste de comunicação volte a oscilar, significa dizer que a interface foi restabelecida. Sendo assim, foi configurado o e-mail conforme figura 71, para ser enviado no momento em que a falha seja corrigida.

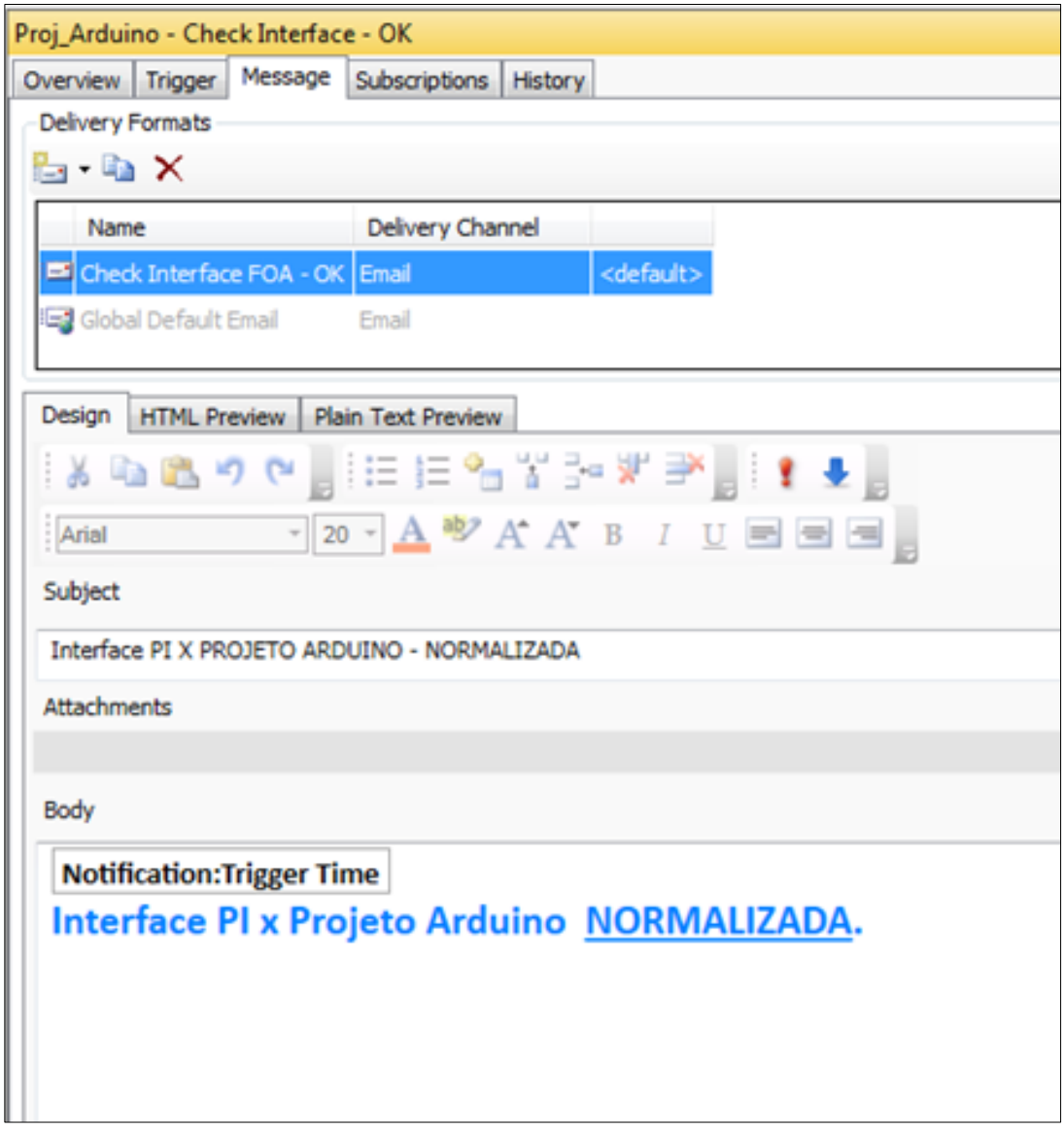


Figura 71 – Configuração do e-mail de alerta de comunicação restabelecida
Fonte: Autores (2018)

O e-mail de relatório diário foi configurado conforme figura 72, utilizando as *tags* internas criadas no PI System. Este relatório contemplará o valor máximo e média de cada variável, assim como o tempo de operação do equipamento durante 24 horas do dia anterior. O e-mail foi configurado para ser enviado todo dia às 07:30h com o resultado do histórico do período de 23:00h às 23:00h dos dias anteriores.

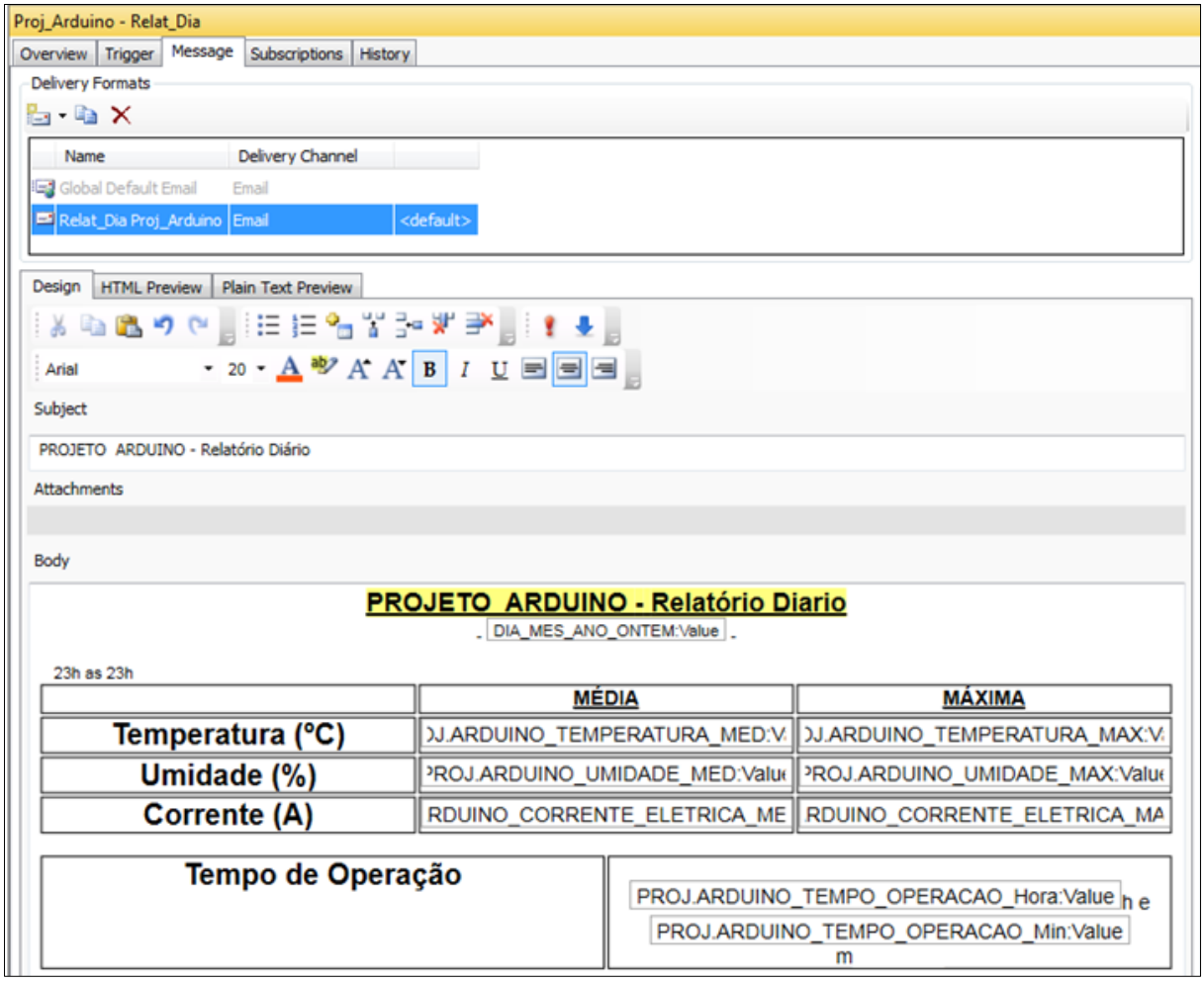


Figura 72 – Configuração do e-mail de relatório diário
Fonte: Autores (2018)

5.2.5.6 Tela gráfica

A ferramenta utilizada para criação da tela gráfica foi o PI ProcessBook. Na figura 73, podemos observar a tela inicial para configuração e desenvolvimento de telas gráficas, através desta ferramenta.

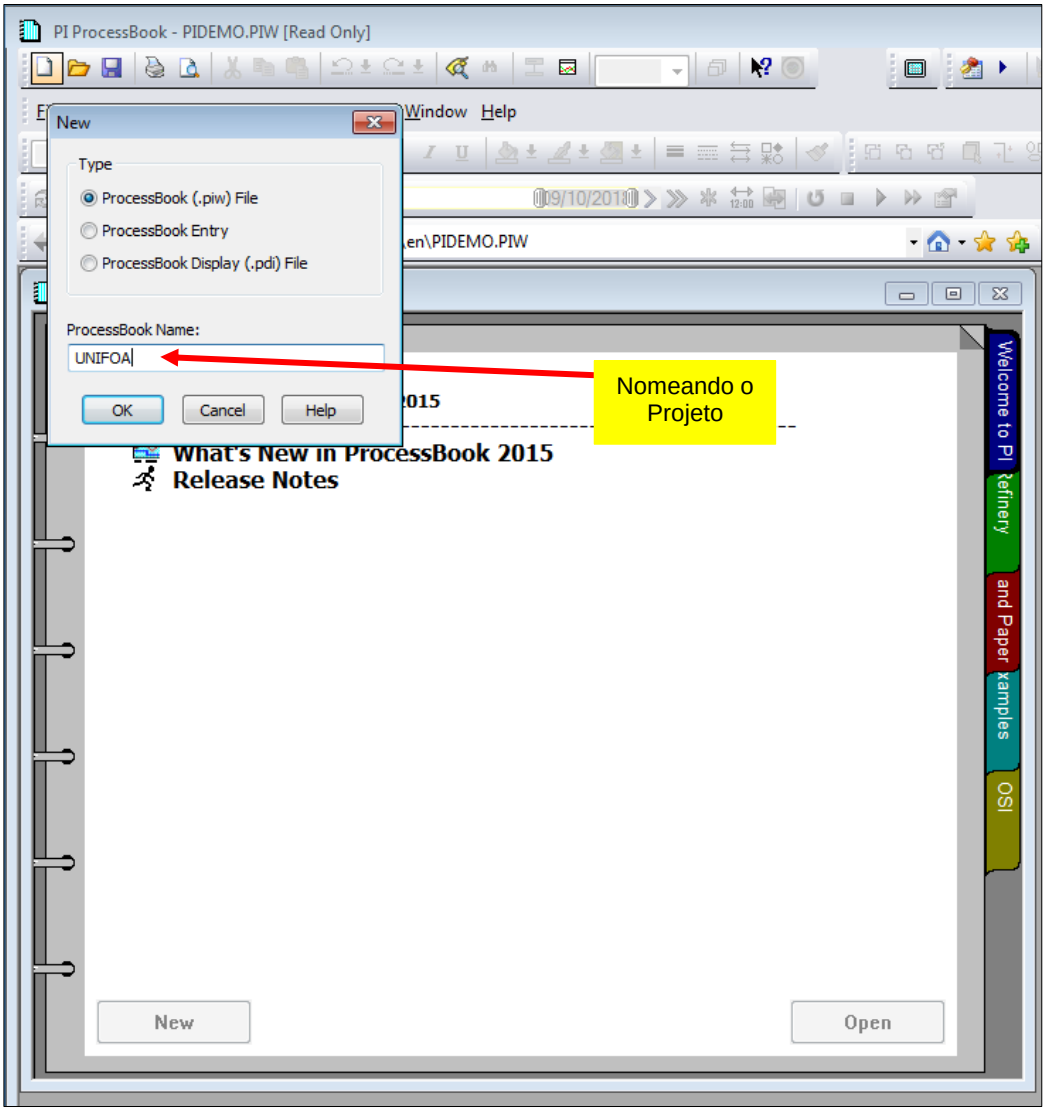


Figura 73 – Tela inicial do PI ProcessBook
Fonte: Autores (2018)

Após ser configurado o nome do projeto, foi criado o nome das telas que serão utilizados para desenvolvimento de animações, observe na figura 74.

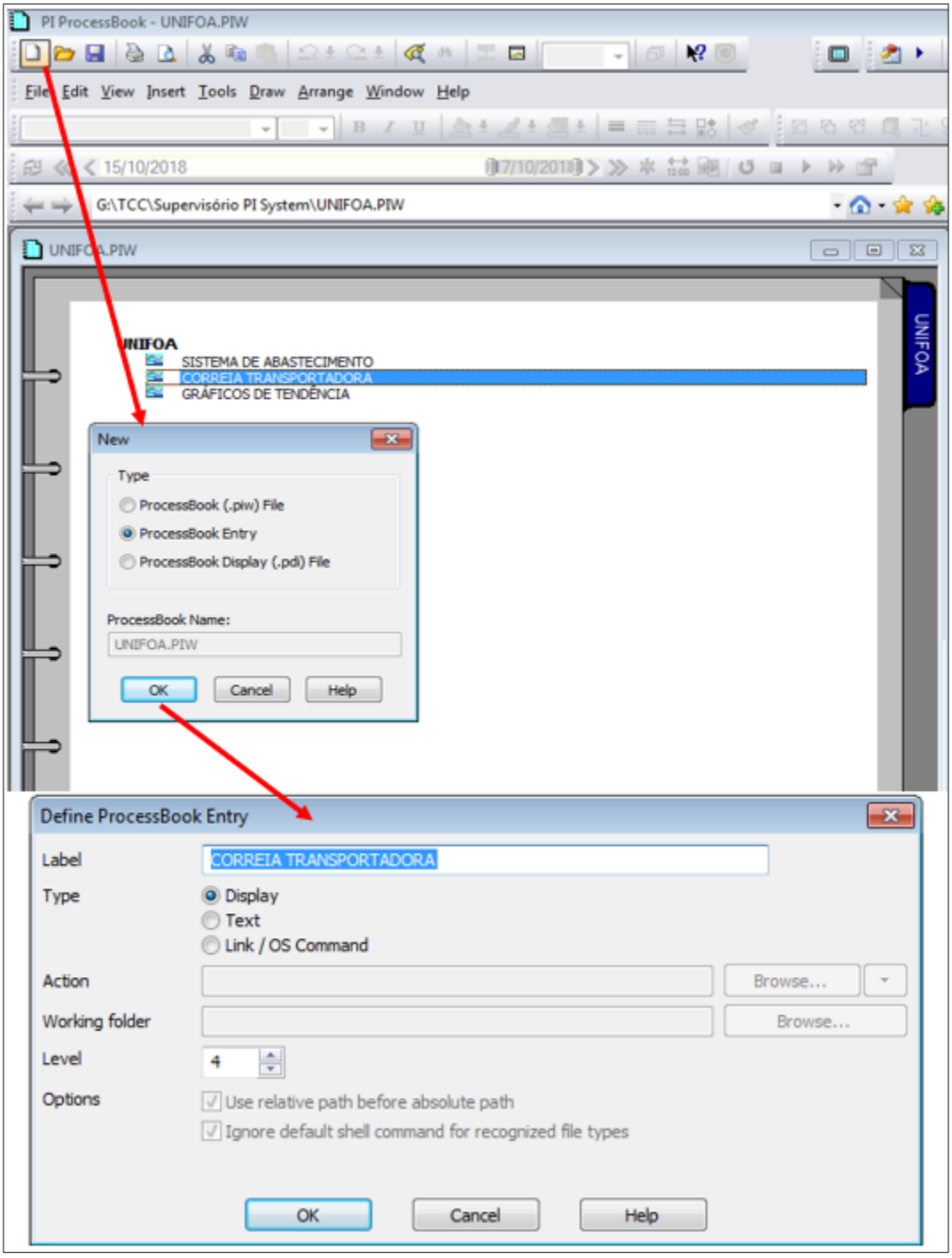


Figura 74 – Configurando novas telas via PI ProcessBook
Fonte: Autores (2018)

Nas figuras 73 e 74, nota-se o porquê do nome da aplicação ser PI ProcessBook, pois gera-se um layout parecido com a de um livro, onde pode-se organizar os projetos de formar amigável.

Ao clicar no nome da tela configurada anteriormente, será iniciado a área de trabalho do ProcessBook, para desenvolvimento. Veja o exemplo na figura 75.

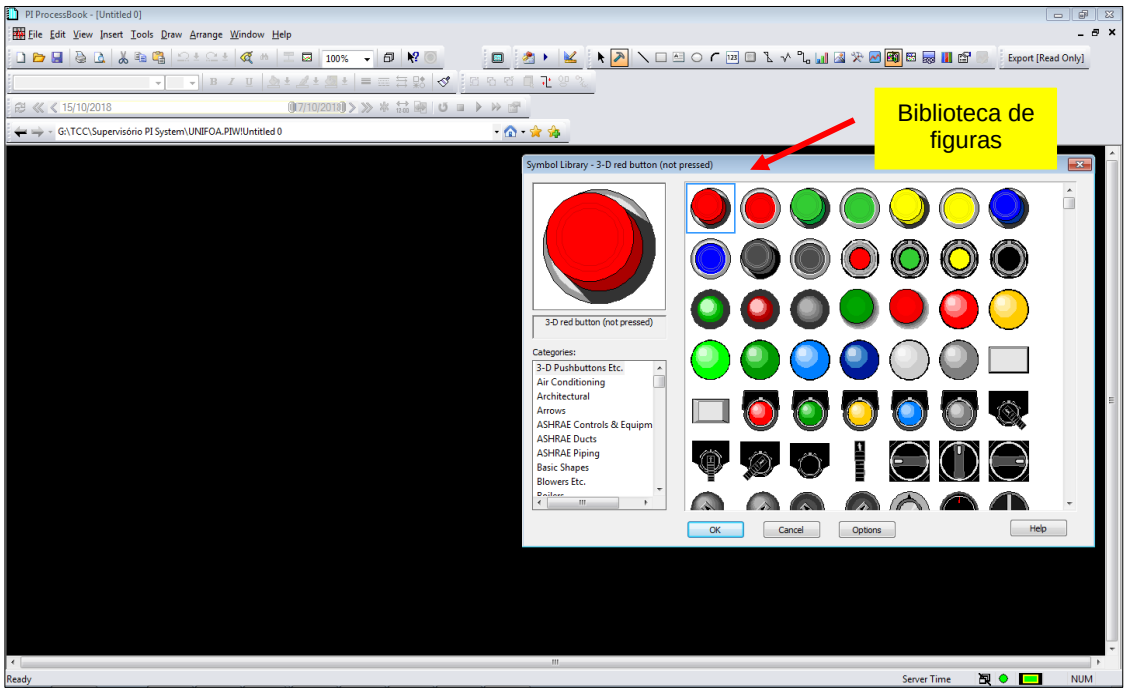


Figura 75 – Área de trabalho do ProcessBook para desenvolvimento de animações
Fonte: Autores (2018)

Podemos observar o resultado final da tela, com a representação de um processo na figura 76, com a utilização de um motor, redutor, correia transportadora e um painel com inversor de frequência.

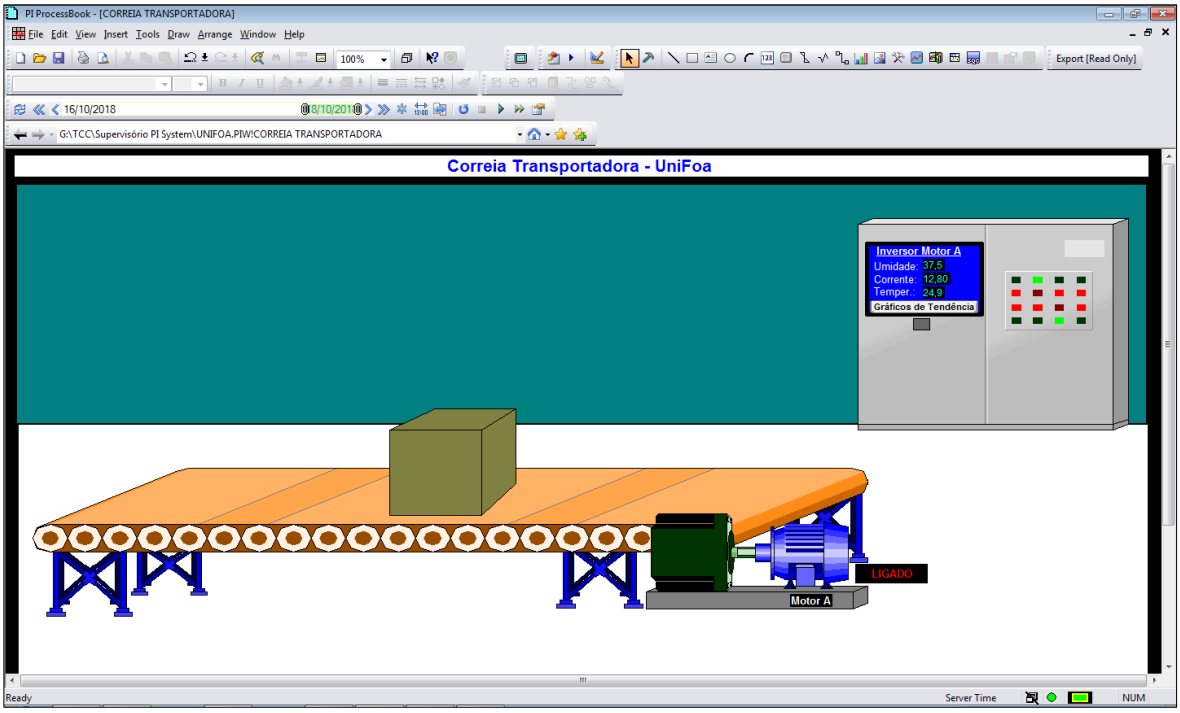


Figura 76 – Visão geral do processo
Fonte: Autores (2018)

Observe na figura 77, as medições das variáveis sendo indicadas na IHM do inversor, onde as mesmas estão sendo monitoradas e entregue pelo Arduino, e na mesma tela, há a opção de abrir os gráficos de tendência, figura 78.

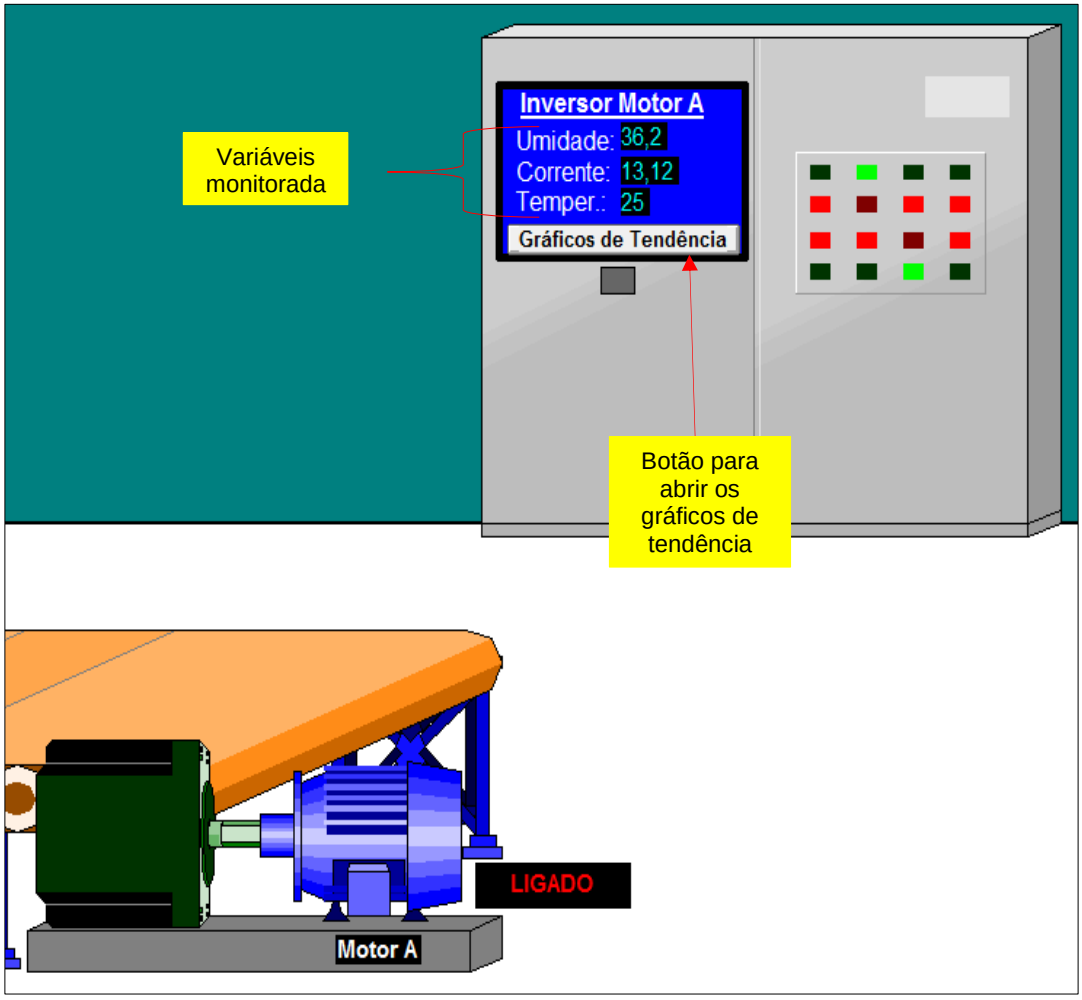


Figura 77 – Variáveis visualizadas na tela gráfica
Fonte: Autores (2018)

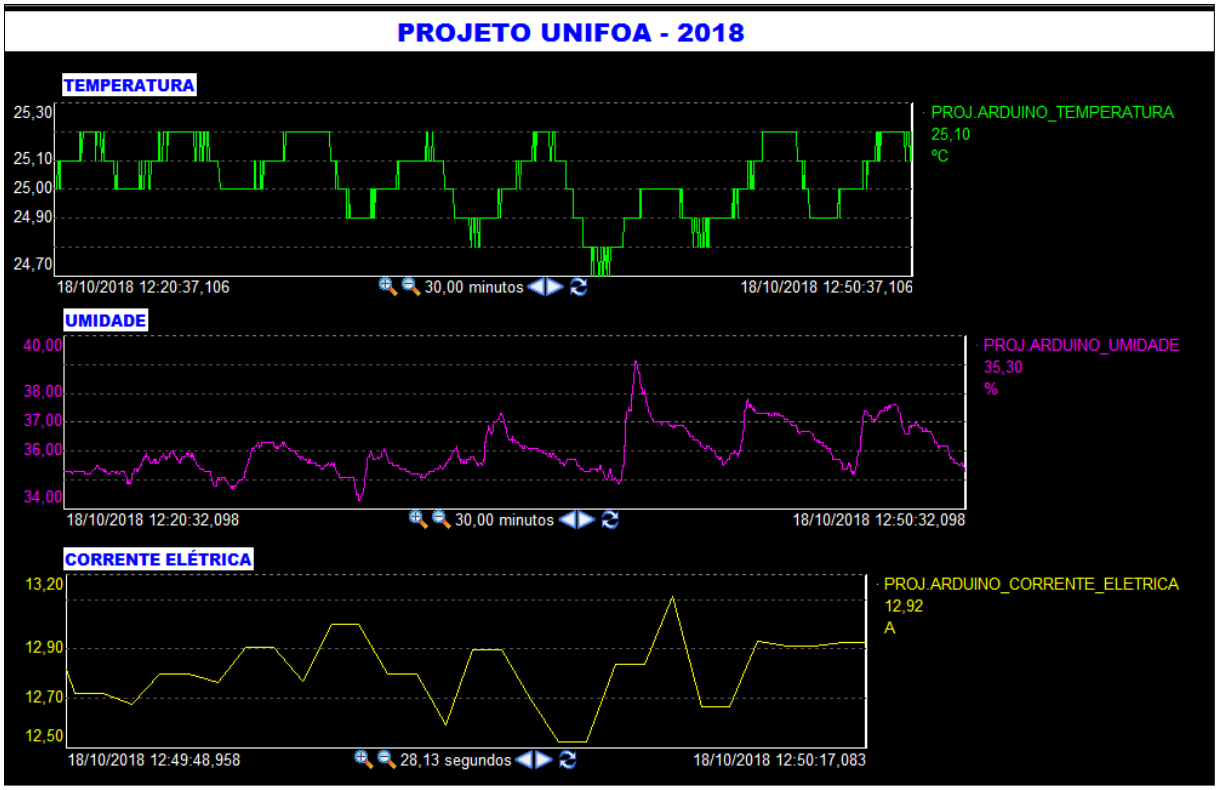


Figura 78 – Gráficos de tendência das variáveis
Fonte: Autores (2018)

Após a realização da representação do processo, foi configurado as *tags* das variáveis (temperatura, umidade e corrente) do PI System, com as figuras, para que seja realizado o monitoramento online, figuras 79 e 80.

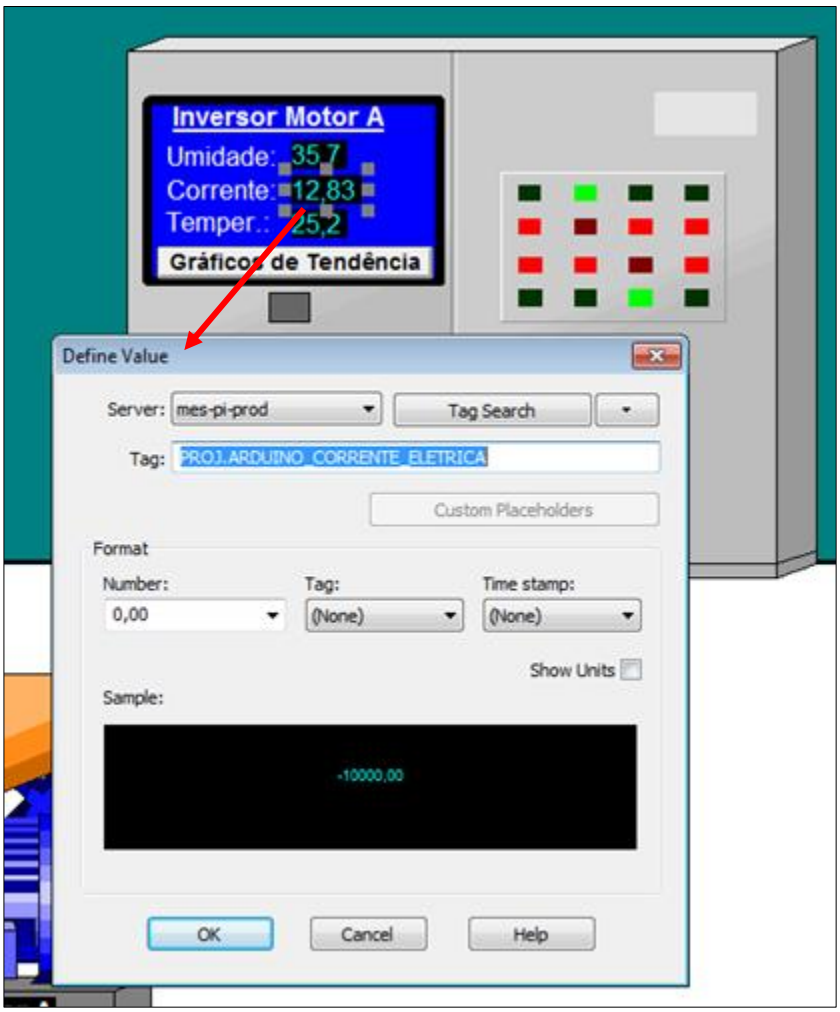


Figura 79 – Configuração da tag de corrente
Fonte: Autores (2018)

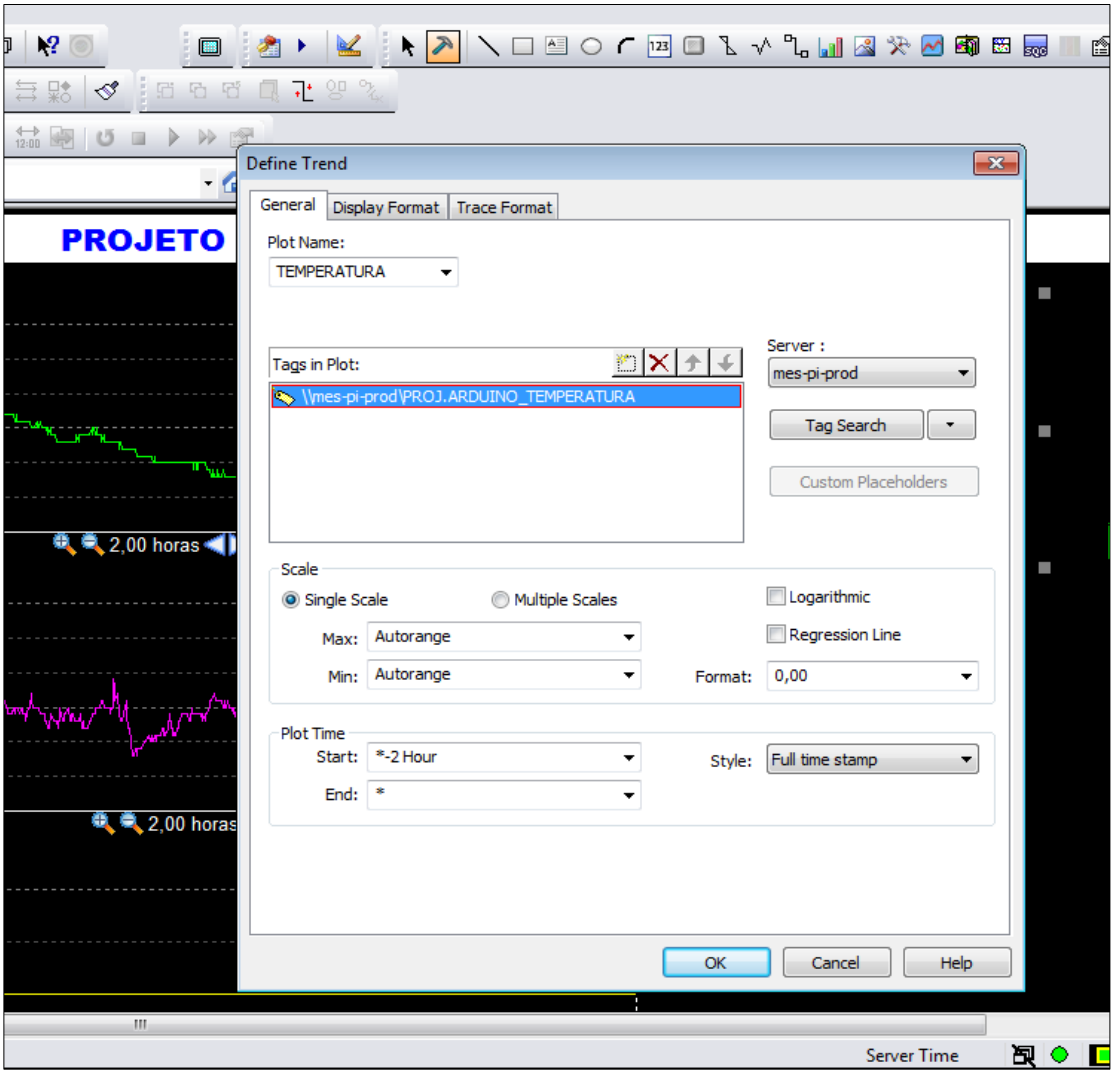


Figura 80 – Configuração do gráfico de temperatura
Fonte: Autores (2018)

O projeto desenvolvido, teve de ser convertido, para que fosse gerado um arquivo “.html” (essa opção é encontrada no próprio menu do ProcessBook), figura 81. Onde o mesmo pode ser visualizado através da ferramenta ActiveView e o arquivo pode ser compartilhado com outras pessoas que tenham acesso à um computador com as ferramentas instaladas.

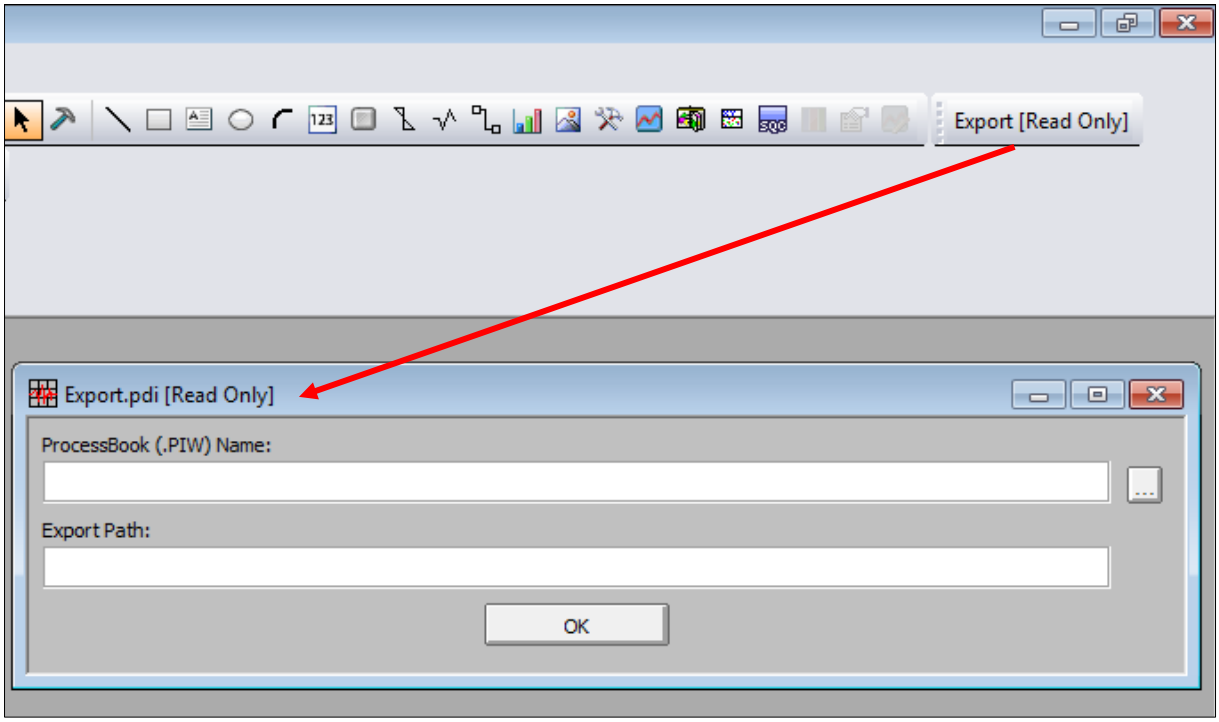


Figura 81 – Convertendo o arquivo da tela gráfica
Fonte: Autores (2018)

Ao abrir a janela, foi inserido o projeto que se desejava realizar a conversão e onde iria ficar o arquivo gerado, figura 82. Vale ressaltar, que caso o projeto seja disponibilizado para visualização na empresa com outras pessoas, esse arquivo gerado, deverá ficar em uma pasta na rede onde todos tenham acesso (essa pasta de rede deverá ser indicada no *"Export Path"*).

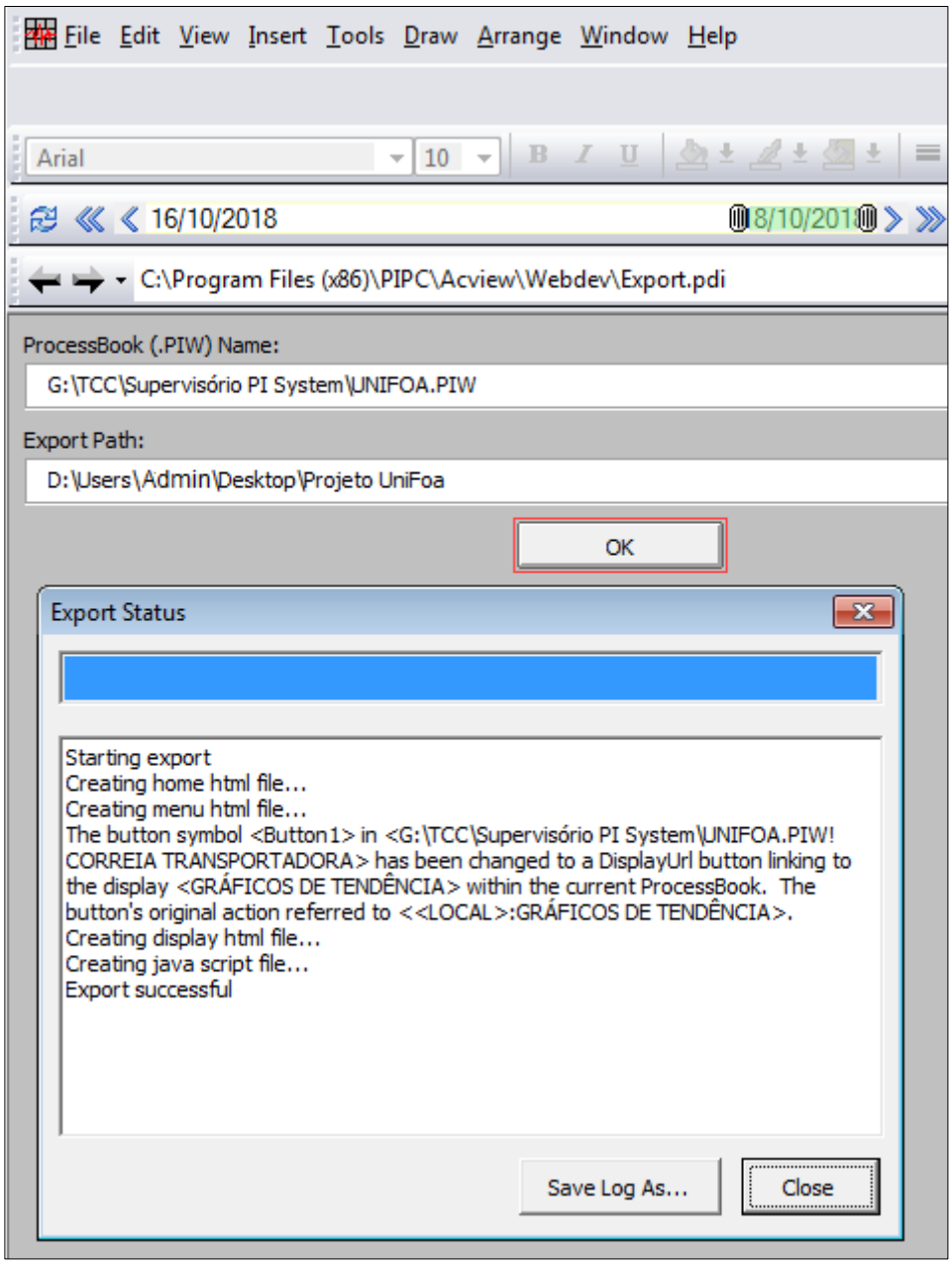


Figura 82 – Convertendo o arquivo
Fonte: Autores (2018)

Após a conversão do projeto, na figura 83 temos os arquivos gerados pelo ProcessBook. O arquivo “PBdsply.htm” será visualizado pelo ActiveView (outra ferramenta do PI System, para visualização). Podemos renomear o arquivo de acordo com a unidade da empresa ou projeto.

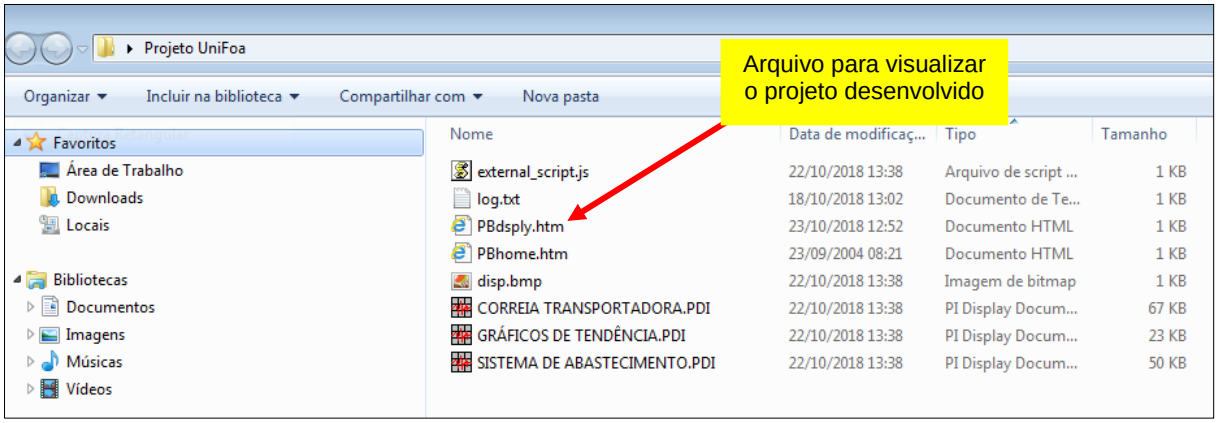


Figura 83 – Pasta com os arquivos gerados após conversão
Fonte: Autores (2018)

Este arquivo poderá ser compartilhado com outras pessoas ou até mesmo, carregá-lo em um *pendrive*, para que possa ser visualizado em qualquer local da empresa (desde que se tenha acesso ao projeto pela rede e o PI ActiveView instalado), permitindo que as reuniões sobre o processo da empresa ou manutenção, sejam realizadas de forma que toda a planta, possa ser observada remotamente e online. Na figura 84, temos a visão geral do processo, sendo visualizado através do ActiveView.

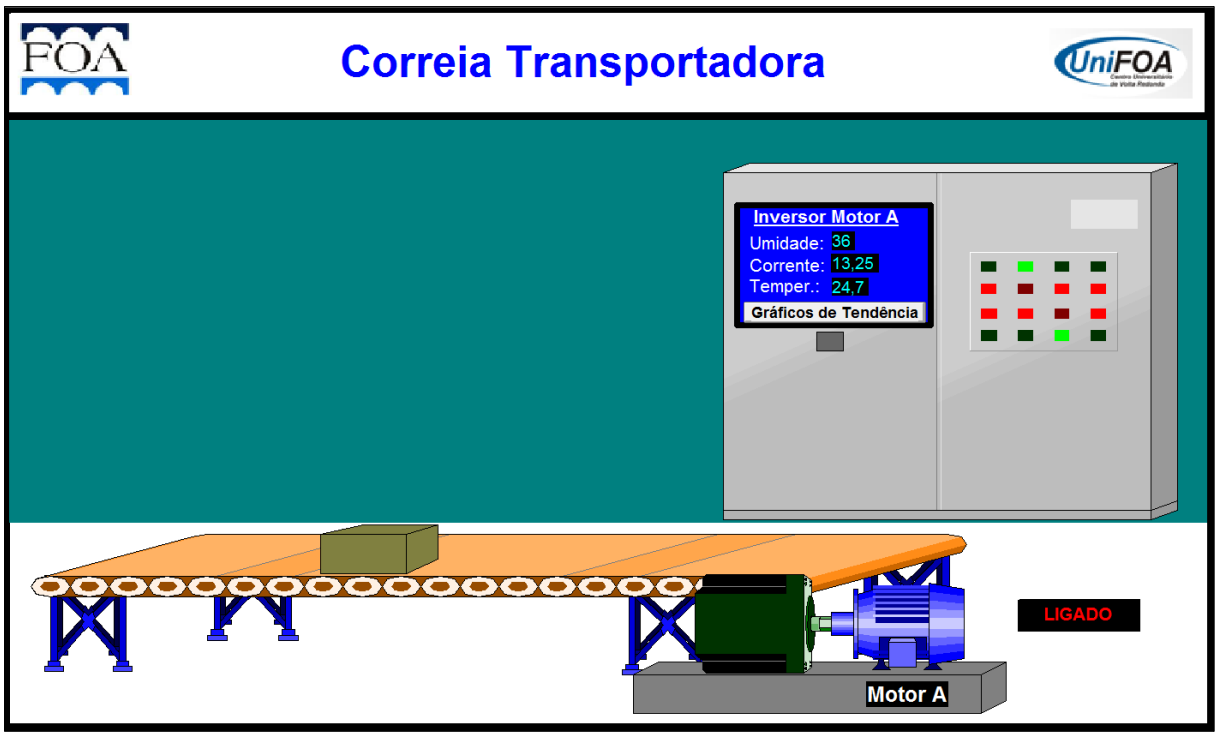


Figura 84 – Visão geral pelo ActiveView
Fonte: Autores (2018)

6 TESTES E RESULTADOS

Após finalizar as devidas configurações, inicialmente, foi realizado o teste de conexão via *software* IDE do Arduino, através do monitor serial. Na figura 85, pode-se visualizar que a conexão foi bem estabelecida e os dados enviados para o CLP.

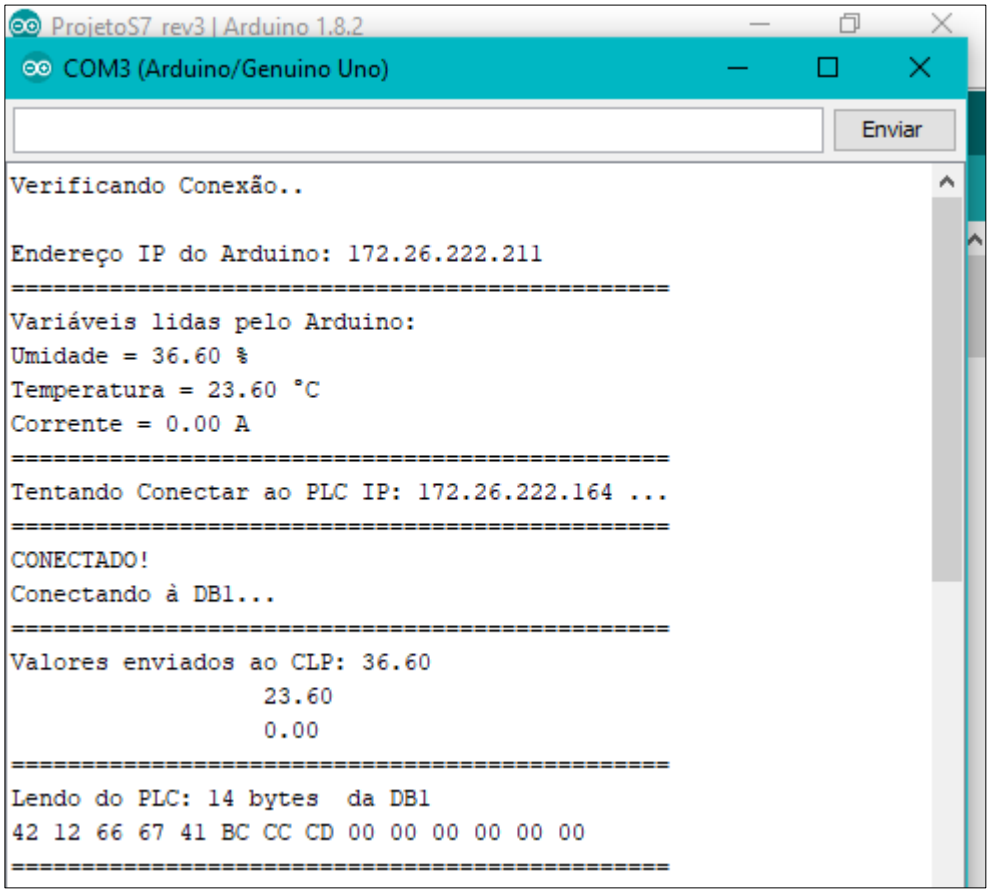


Figura 85 – Teste inicial (Conexão estabelecida)
Fonte: Autores (2018)

Após certificar que a conexão estava estabelecida pelo Arduino, foi verificado se os sinais (umidade, temperatura e corrente elétrica) estavam sendo devidamente recebidos pelo CLP. Na figura 86 podemos visualizar através do Simatic Manager que os dados foram recebidos corretamente na DB1, responsável por armazenar os dados, podendo então ser processados pelo PI System via OPC.

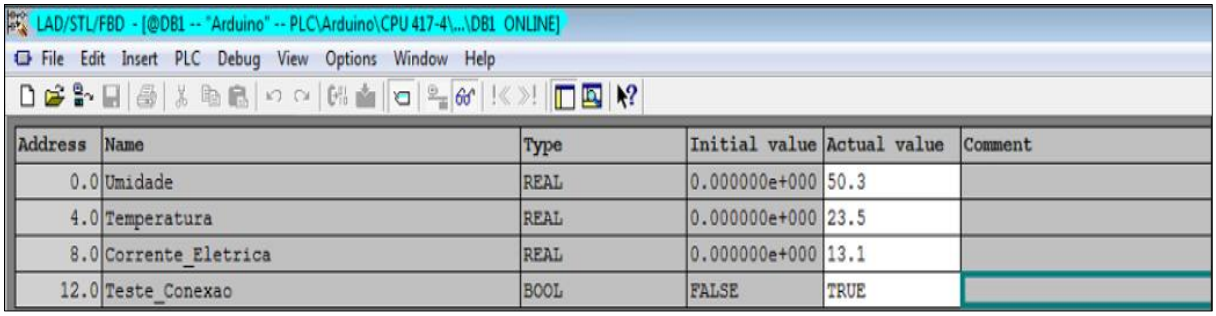


Figura 86 – Informações recebidas no CLP
Fonte: Autores (2018)

Com todos os dados sendo disponibilizados corretamente na rede, foi verificado se o *software* KepServer estava realizando a leitura dos dados da DB1 do CLP corretamente. Na figura 87, podemos observar na parte circulada de vermelho que os dados das variáveis estavam sendo processados corretamente pelo servidor OPC.

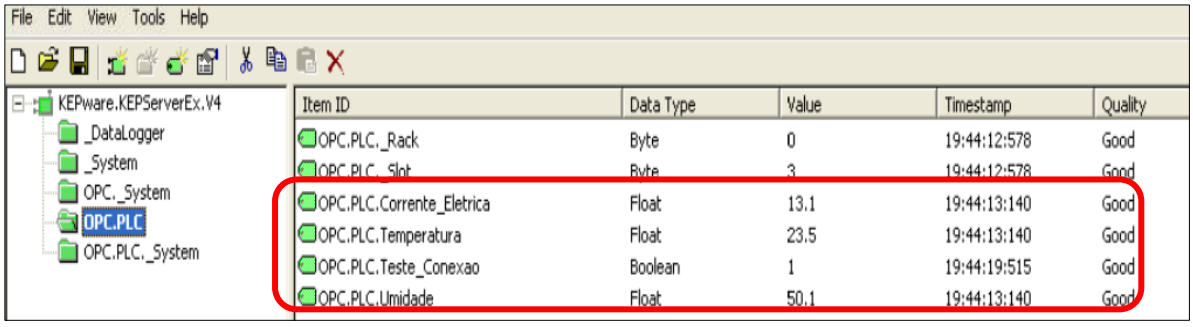


Figura 87 – Servidor OPC online
Fonte: Autores (2018)

Para teste do cliente OPC, foi utilizado a ferramenta nomeada OPC Tool que, possibilita realizar teste de conexão. Portanto, após a instalação do PI OPC, foi realizado o teste a partir da pasta criada automaticamente no computador, normalmente no caminho "C:\Arquivos de programas\PIPC\PI-OPC Tools\PI_OPCTool", conforme mostra a figura 88.

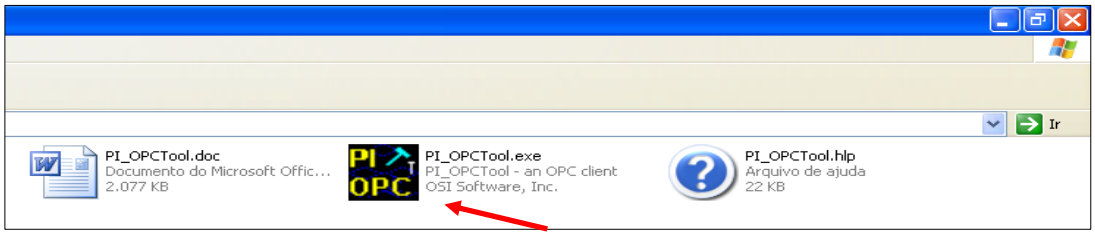


Figura 88 – OPC Tool
Fonte: Autores (2018)

Após abrir o programa, foram executados os seguintes passos conforme indicado na figura 89:

1 – Selecionar o servidor OPC que deverá aparecer automaticamente na lista;

2 – Conectar com o servidor;

3 – Escolher um nome qualquer para o grupo;

4 – Clicar em "*create"* para criar o novo grupo;

5 – Listar todas as *tags* disponíveis no servidor;

6 – Selecionar alguma *tag* com duplo clique;

7 – Conferir e anotar o nome, pois essa será a *tag* utilizada nas configurações do PI System;

8 – Adicionar ao grupo e verificar o *status* da variável na área denominada "*Points".* Deve estar indicando "*running"* conforme mostra a figura 89.

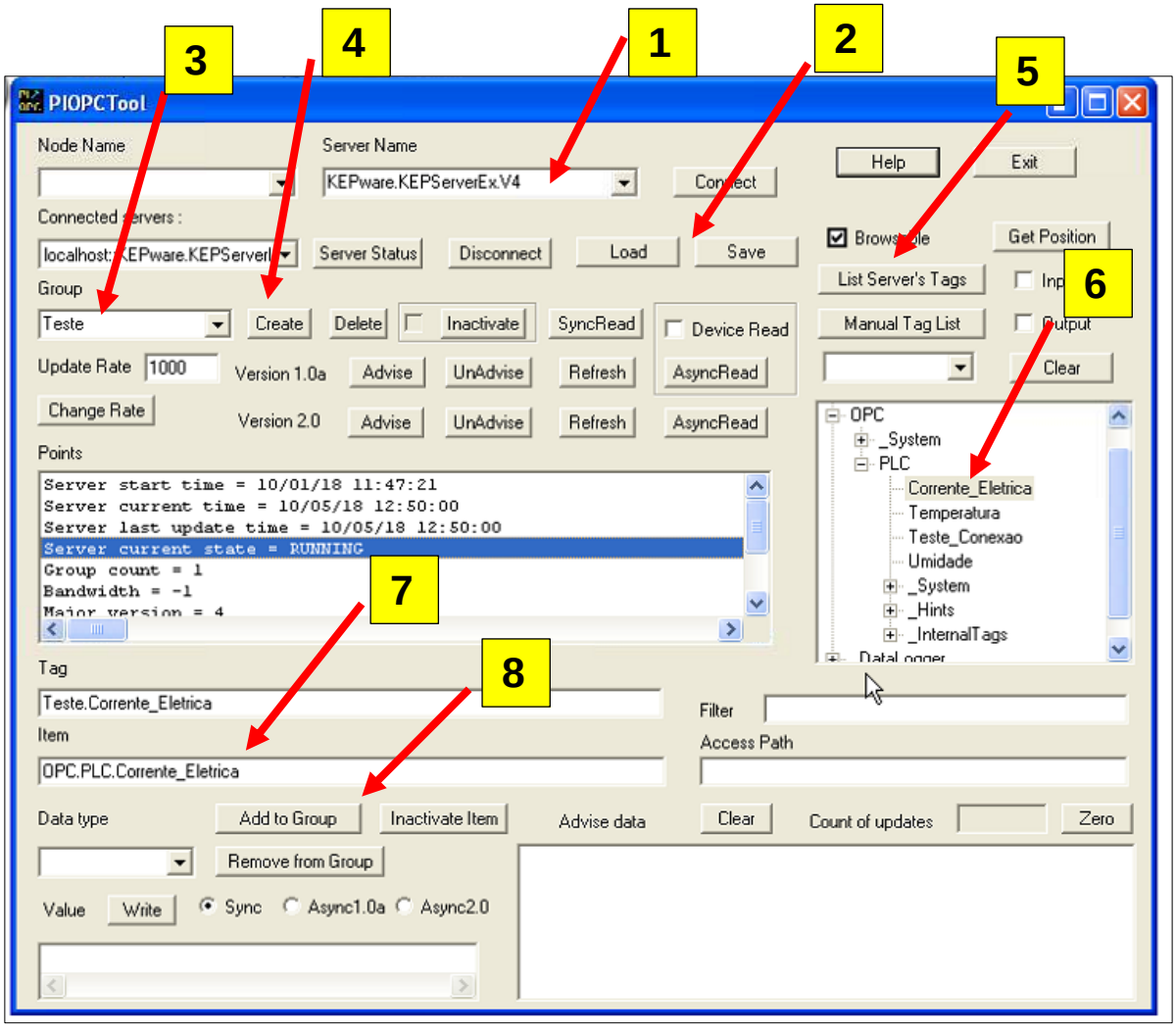


Figura 89 – Teste de conexão via OPC Tool
Fonte: Autores (2018)

Com todas as interfaces funcionando corretamente, foram realizados os testes no PI system.

Para validação das configurações de notificação via e-mail, foram simulados via Arduino os valores de 35°C, 20A e 65% correspondente a temperatura, corrente e umidade respectivamente e o sistema funcionou normalmente conforme mostrado nas figuras 90, 91 e 92 que representa o e-mail recebido no momento da mudança do sinal, ultrapassando os valores de referência de 30°C, 15A e 55% para disparo de cada e-mail.

Figura 90 – Recebimento do e-mail de alerta de temperatura alta
Fonte: Autores (2018)

Figura 91 – Recebimento do e-mail de alerta de corrente alta
Fonte: Autores (2018)

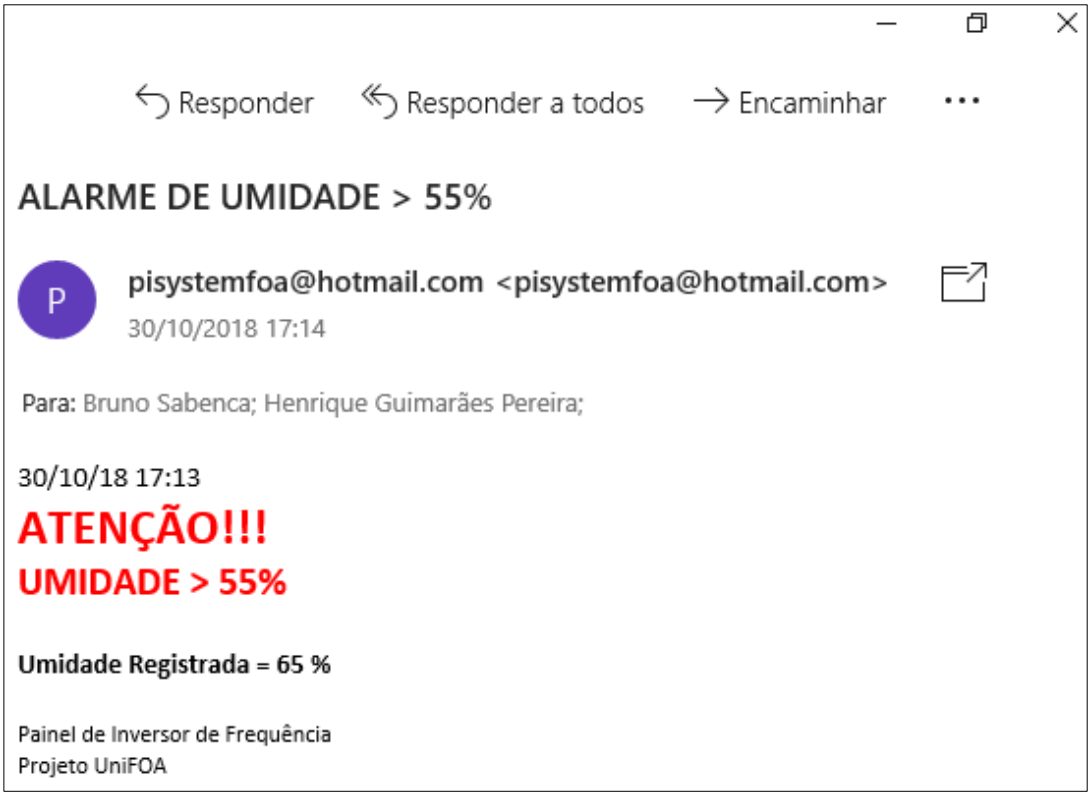


Figura 92 – Recebimento do e-mail de alerta de umidade alta
Fonte: Autores (2018)

Na figura 93, é possível observar através do gráfico de tendência criado no PI ProcessBook, o momento em que foram simulados os valores de temperatura, umidade e corrente. Podemos notar a variação do sinal no final do gráfico.

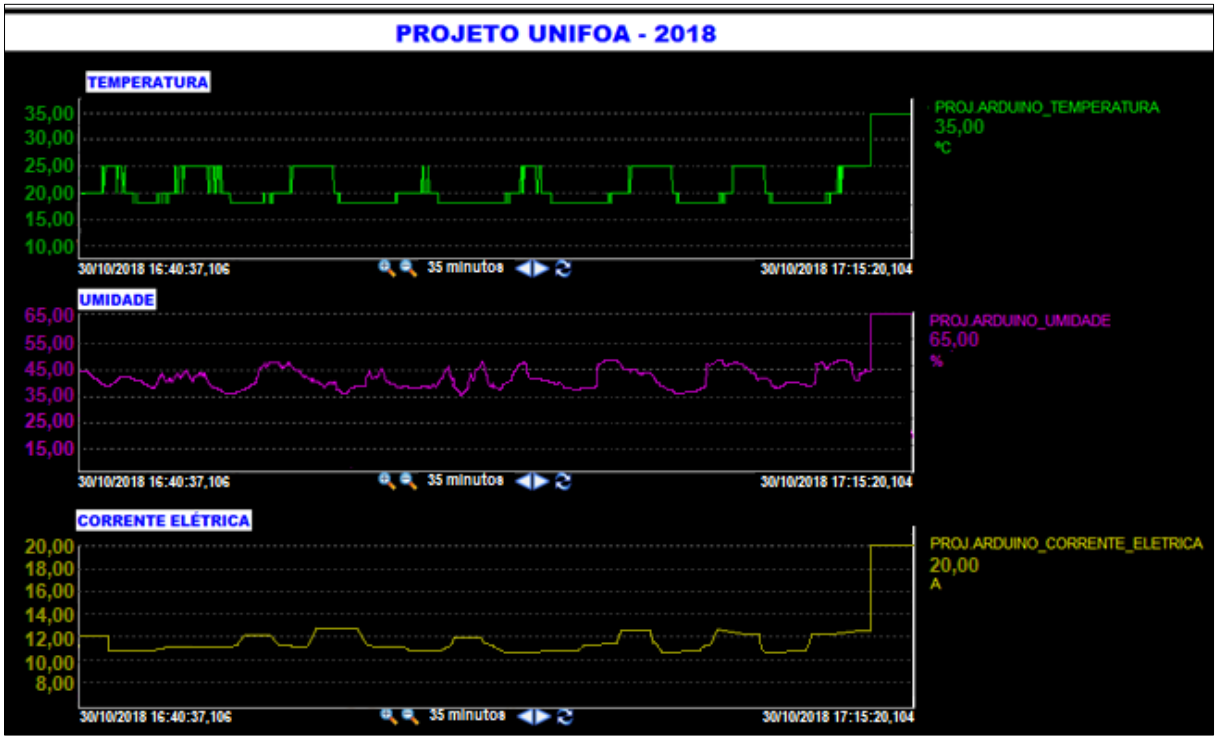
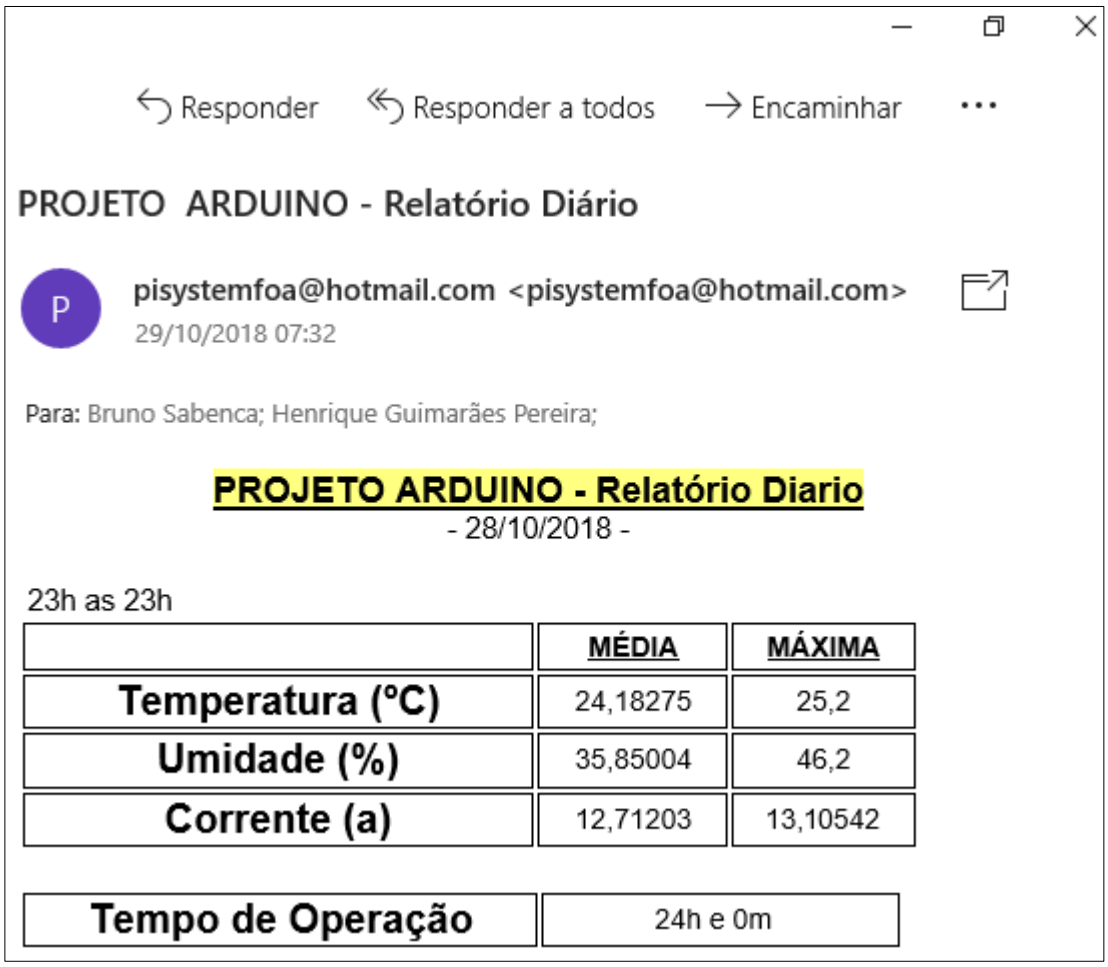


Figura 93 – Gráfico de tendência das variáveis após simulação dos sinais
Fonte: Autores (2018)

O teste do relatório ocorreu no dia seguinte ao término das configurações. Na figura 94 é possível observar que o e-mail foi recebido corretamente e os dados coletados durante o dia foram transmitidos normalmente, podendo assim ser analisados e auxiliar na manutenção preditiva do equipamento através das informações recebidas.



Responder Responder a todos Encaminhar ...

PROJETO ARDUINO - Relatório Diário

P

pisystemfoa@hotmail.com <pisystemfoa@hotmail.com>

29/10/2018 07:32

Para: Bruno Sabenca; Henrique Guimarães Pereira;

PROJETO ARDUINO - Relatório Diario

- 28/10/2018 -

23h as 23h

	MÉDIA	MÁXIMA
Temperatura (°C)	24,18275	25,2
Umidade (%)	35,85004	46,2
Corrente (a)	12,71203	13,10542

Tempo de Operação	24h e 0m

Figura 94 – Recebimento do e-mail de relatório diário

Fonte: Autores (2018)

7 CONCLUSÃO

Durante o desenvolvimento deste projeto e das pesquisas realizadas sobre o tema, pode-se observar que desde o uso da energia elétrica nas indústrias no século 20, os equipamentos ligados ao processo produtivo, vêm evoluindo em tecnologias de forma exponencial. O avanço da eletrônica junto com a informatização, estão cada vez mais presentes dentro da indústria. A tecnologia contribui de diversas formas, como, na segurança, produtividade, conforto, precisão, conectividade entre pessoas e equipamentos e modernização da planta.

Nas indústrias, a automação domina o processo produtivo, segurança e meio ambiente e avança extraordinariamente em residências, prédios e comércios. Em paralelo, foi visto que a informatização ou IoT, vem ocupando o seu espaço em ambos os setores, auxiliando a automação.

Hoje as empresas buscam ser competitivas, entregar produtos com qualidade e com menor prazo. Buscam novas ideias que possam proporcionar melhores resultados, reduzindo o seu custo, com baixo investimento e retornos significativos.

A manutenção é um fator importante dentro das empresas, onde o seu papel pode ser impactado quando mau gerenciada. As manutenções corretivas são as mais significativas e a preditiva ainda é pouco explorada.

O desenvolvimento deste projeto, trouxe como uma solução, análise preditiva para as indústrias, com o monitoramento das variáveis de um inversor de frequência, com o envio de e-mails instantâneos, no momento de um desvio significativo das variáveis monitoradas. Foram feitas as coletas de dados de temperatura, corrente elétrica e umidade. Utilizou-se um CLP da Siemens (CPU, Placa de rede e Fonte), trabalhando com o Arduino como ponto de rede. Observou-se um grande desafio, fazer com que o Arduino fosse uma remota de um CLP trocando dados com a CPU. Foram realizados estudos em automação (configuração e programação de CLP Siemens), programação em Arduino (linguagem C/C++) e revisão dos conteúdos de eletrônica, física e elétrica do curso de Engenharia Elétrica. A alternativa abordada, contribui com o avanço do uso do Arduino nas automações industriais. Após estudos e pesquisas, foram obtidos os resultados esperados entre Arduino e CLP, onde a configuração de troca de dados ocorreu perfeitamente sem perda de informações ou falhas de comunicação.

Outro desafio encontrado, foi a configuração do *software* para comunicação com o CLP, armazenamento de dados, desenvolvimento de telas gráficas e o envio de e-mail instantâneo e após estudos e pesquisas, optou-se em utilizar o PI System. Adquiriu-se grande conhecimento em automação (configuração de supervisório), parametrização, desenvolvimento de telas gráficas, geração de relatórios (configuração via *software*) e envio de e-mails automáticos (configuração via *software*).

O objetivo foi alcançado com êxito, pois foi possível monitorar um inversor de frequência a distância, gerar um banco de dados e enviar relatórios automáticos com horário pré-definido, indicando o comportamento durante um determinado período. O e-mail pode ser acessado em qualquer local que se esteja, desde que se tenha acesso à um dispositivo (computador, notebook, tablet ou celular), com acesso à *internet*. As informações foram enviadas no horário exato da falha, qual o tipo de falha e o valor lido pelo sensor. A partir destes dados conseguimos chegar à conclusão que o motor da correia transportadora que possui uma corrente nominal de 33A, está trabalhando em média a 40% de sua carga. Esta informação foi obtida através dos dados de máximo valor lido pelo sensor de corrente enviado através do relatório diário. Analisando os demais dados, concluímos que o painel do inversor de frequência está operando em boas condições ambientes conforme orientação do manual do fabricante.

O *software* utilizado para desenvolvimento de tela gráfica, permitiu gerar um arquivo, onde o mesmo pode ser compartilhado dentro da empresa e ser visualizado em qualquer ambiente com acesso ao computador, dessa forma todos passam a ter a real situação do ambiente produtivo, auxiliando nas tomadas de decisões.

A gestão da manutenção, passou a se tornar precisa e eficaz, onde agora é possível identificar o momento exato da falha ou a tendência de falha, podendo se antever o problema e diminuir as paradas corretivas de emergência.

Portanto, com o desenvolvimento desse projeto, foi realizado a manutenção preditiva online, sem a necessidade de interferência humana para coleta de informações e aviso sobre falhas, fazendo com que os responsáveis estejam conectados com os equipamentos 24 horas por dia.

8 REFERÊNCIAS

ABB. **Hardware Manual - ACS800-104 Inverter Modules**. B. ed. : ABB, v. I, 2003.

ABB. **ABB industrial drives, ACS800, drive modules 0.55 to 2900 kW catalog**. ABB. [S.l.], p. 61. 2012.

ABRAMAN. **A Situação da Manutenção no Brasil**. Associação brasileira de Manutenção de Ativos. Salvador, p. 23. 2013.

AZEVEDO, M. T. D. **Transformação Digital na Industria 4.0 e a Rede de Água no Brasil**. Escola Politécnica da Universidade de São Paulo. São Paulo, p. 177. 2017.

CARVALHO, F. A. D.; GOMES, H. D. S.; SILVA, T. R. C. D. **Pulseira de Localização via Módulo GPS/GSM**. Centro UNiversitário de Volta Redonda. Volta redonda, p. 79. 2017.

CARVALHO, M. F. P. R. D. Automação e Controle Residencial via Internet Utilizando Arduino. **As Mudanças Climáticas, Desastres Naturais e Prevenção de Riscos: Estamos Preparados?**, Rio de Janeiro, n. 1, p. 34, 2011.

COSTA, A. P. D.; SERMANN, F. C.; SILVA, G. G. D. **Desenvolvimento de um Protótipo para Medição de Energia Elétrica**. Universidade Tecnológica Federal do Paraná. Curitiba, p. 71. 2016.

COSTA, M. D. L. Campos magnéticos gerados por bobinas e a lei de Ampére. **Anel de Thompson - Projeto de Fisica C Prática**, 2017. Disponivel em: <http://fisicacproj.blogspot.com/2017/07/campos-magneticos-gerados-por-bobinas-e.html>. Acesso em: Agosto 2018.

DEIDMAR, G. L. C.; SOBREIRA, D. D. S.; LIMA, W. D. D. Internet das coisas na Educação. **Tecnologias em Projeção**, v. 8, n. 2, p. 67-78, Dezembro 2017.

DEMETRAS, E. SCT-013 – Sensor de Corrente Alternada com Arduino. **Vida de Silicio**, 2017. Disponivel em: <https://portal.vidadesilicio.com.br/sct-013-sensor-de-corrente-alternada/>. Acesso em: Julho 2018.

DIGIMER. CABO REDE / PATCH CORD RJ45 CATEGORIA 5, 15M, PAR TRANÇADO AMP MD9 5625. **Digimer**, 2018. Disponivel em:

<http://www.digimer.com.br/cabo-rede-rj45-rj45-categoria-5-15m-montado-par-trancado-amp-patch-cord.html>. Acesso em: Agosto 2018.

DINIZ, E. H. Internet das Coisas. **GV-executivo**, v. 5, n. 1, p. 59, 2006.

EVANS, D. A Internet das Coisas, Como a próxima evolução da Internet está mudando tudo. **Cisco Internet Business Solutions Group (IBSG)**, p. 4-11, Abril 2011.

FFILIPEFLOP. Ethernet Shield W5100 para Arduino. **FILIPEFLOP**, 2018. Disponivel em: <https://www.filipeflop.com/produto/ethernet-shield-w5100-para-arduino/>. Acesso em: Agosto 2018.

FONSECA, M. D. O. **Comunicação OPC – Uma abordagem prática**. VI Seminário de Automação de Processos, Associação Brasileira de Metalurgia e. Vitória, p. 12. 2002.

GENENA, S. K. **Simplementação, Configuração e Customização do Sistema PI na Unidade Multipropósito de FCC**. Universidade Federal de Santa Catarina. Florianópolis, p. 91. 2004.

GRUBER, V. **Sistema de Monitoramento Remoto Baseado em Rrede de Celular GSM/GPRS para Gerenciamento de Desgaste de Pastilha de Freio e Vibração da Torre em Aerogeradores**. Universidade Federal do Rio Grande do Sul. Porto Alegre, p. 78. 2007.

GURSKI, C. A. **Curso de Formação de Operadores de Refinaria:** Noções de Confiabilidade e Manutencção Industrial. Curitiba: Equipe Petrobrás, 2002.

HALLIDAY, D.; RESNICK, R.; KRANE, K. S. **Física 3**. 5. ed. Rio de Janeiro: LTC, v. 2, 2004.

JÚNIOR, A. J. D. S.; BARBOSA, D. F.; JÚNIOR, A. G. D. C. Controle Automático de Luminosidade de Ambientes e Alarme com Trava Eletrônica, Aplicados a Sistemas Residenciais, Utilizando Rede ZigBee e Arduino. **XLII CONGRESSO BRASILEIRO DE EDUCAÇÃO EM ENGENHARIA**, Mauá, 2015. 10.

KAGERMANN, H.; WAHLSTER, W.; HELBIG, J. **Recommendations for implementing the strategic initiative Industrie 4.0**. [S.l.]: Acatech - National Academy of Science and Engineering, 2013.

KUROSE, J.; ROSS, K. **Computer Networking:** A Top-Down Approach Featuring the Internet. 6. ed. [S.l.]: Pearson Education, v. 1, 2013.

LEMOS, H. D. Aplicação da Computação Ubíqua na Educação a Distancia para Elucidação da Fotossíntese no Ensino de Biologia. **ERI-GO 2014**, Goiânia, n. 2, p. 12-23, Novembro 2014.

LILA, F. **Protocolos de Comuniação Aplicado na Automação de Usinas hidrelétricas**. Universidade do Sul de Santa Catarina. Palhoça, p. 69. 2012.
MARCHESAN, M. **Sistema de Monitoramento Residencial Utilizando a Plataforma Arduino**. Colégio Técnico Industrial de Santa Catarina. Santa Catarina, p. 62. 2012.

MARCORIN, W. R.; CAMELO LIMA, C. R. Análise dos Custos de Manutenção. **Revista de Ciência & Tecnologia**, Piracicaba, v. 11, n. 22, p. 35-42, dezembro 2003.

MARIAN, P. AM2302 / DHT22 Datasheet. **Electro Shematics**, 2018. Disponivel em: <https://www.electroschematics.com/11293/am2302-dht22-datasheet/>. Acesso em: Julho 2018.

MARTINS, E. O que é TCP/IP? **Tecmundo**, 2012. Disponivel em: <https://www.tecmundo.com.br/o-que-e/780-o-que-e-tcp-ip-.htm>. Acesso em: Julho 2018.
MCROBERTS, M. **Arduino Básico**. 1. ed. São Paulo: Novatech Ltda, v. 1, 2011.

MMTEC. O que a falta de manutenção preditiva provoca? **MMtec**, 2018. Disponivel em: <http://www.mmtec.com.br/o-que-a-falta-de-manutencao-preditiva-provoca/>. Acesso em: 2018.

MORAES, C. C. D.; CASTRUCCI, P. D. L. **Engenharia de Automação Industrial**. 2. ed. Rio de Janerio: LTC, 2007.

MOREIRA, M. A.; PINTO, A. D. O. Dificuldades dos Alunos na Aprendizagem da Lei de Ampère, à Luz da Teoria dos Modelos Mentais de Johnson-Laird. **Revista Brasileira de Ensino de Física**, Porto Alegre, v. 25, n. 3, p. 317-325, Setembro 2003.

MOTA, A. DHT11 e DHT22, Sensor de umidade e Temperatura com Arduino. **Vida de Silício**, 2018. Disponivel em: <https://portal.vidadesilicio.com.br/dht11-dht22-sensor-de-umidade-e-temperatura/>. Acesso em: Julho 2018.

NILTEC. Switch mini 5 portas Multilaser RE105. **Niltec**, 2018. Disponivel em: <http://www.niltec.com.br/detalhes.php?p=28766&l=32>. Acesso em: Agosto 2018.

OLIVEIRA, I. R. H.; SANTOS, C. R. B.; RODRIGUES, M. A. L. **Desenvolvimento de um aplicativo Android para monitoramento Microcontrolado do nível de um reservatório de água residencial em tempo real**. Universidade Federal de Uberlândia. Uberlândia, p. 6. 2014.

OSISOFT. PI Builder Overview. **OSIsoft**, 2015. Disponivel em: <https://livelibrary.osisoft.com/LiveLibrary/content/en/server-v5/GUID-146F64E4-FBDA-48A4-977C-16178581316D#addHistory=true&filename=GUID-BC326432-90B2-441E-9E83-04B83495DB0E.xml&docid=GUID-E79B7E93-7A62-4D93-A062-E96E0C4D4294&inner_id=&tid=&query=&scope=&>. Acesso em: out. 2018.

OSISOFT. **PI Server Managing and Delivering Sensor-Based Data**. San Leandro: OSIsoft, 2015.

OSISOFT. PI System. **OSISoft**, 2018. Disponivel em: <https://www.osisoft.pt/pi-system/>. Acesso em: Setembro 2018.

OSISOFT, L. **Administração do PI System para profissionais de TI**. San Leandro: OSIsoft, 2012. 255 p.

OSISOFT, L. **PI Processbook Guia de Usuário**. San Leandro: [s.n.], 2015.

OSISOFT, L. **PI System Tools:** Data for Every User. San Leandro: OSISoft, 2017.

OTANI, M.; MACHADO, W. V. A proposta de desenvolvimento de gestão da Manutenção Industrial na busca da excelência ou classe mundial. **Gestão Industrial**, Ponta Grossa, v. 4, n. 2, p. 1-16, 2008.

REIS, T. Internet das Coisas: o que você precisa saber. **Cedrotech**, 2016. Disponivel em: <http://blog.cedrotech.com/internet-das-coisas-o-que-voce-precisa-saber/>. Acesso em: Setembro 2018.

RHY. Split core current transformer. **rhydolabz**, 2018. Disponivel em: <http://www.rhydolabz.com/documents/Specification%20of%20%20SCT013.pdf>. Acesso em: 2018.

SANTOS, J. C. F. D. Indução Eletromagnética. **Globo**. Disponivel em: <http://educacao.globo.com/fisica/assunto/eletromagnetismo/inducao.html>. Acesso em: Agosto 2018.

SCHEUER, A. **Instalação e Administração do Sistema PI na Unidade Multipropósito de FCC**. Universidade Federal de Santa Catarina. Florianópolis, p. 65. 2004.

SIEMENS. **Automation System S7-400 Configuration and Use**. Nurnberg: [s.n.], 2005.

SIEMENS. **S7-CPs for Industrial Ethernet**. Nurnberg: [s.n.], 2010.
SIEMENS. **Automation System S7-400:** Hardware and Installation. Nurnberg: Siemens, 2016.

SIEMENS. **S7-400 Automation System Module Data**. Nurnberg: [s.n.], 2016.

SIEMENS. **Conceito de Industria 4.0**. Portugal: Siemens SA, 2017.

SIEMENS. **SIMATIC S7-400 Adavenced Controller**. [S.l.]: Siemens, 2017.

SILVA, D. C. M. D. Corrente alternada. **Mundo Educação**, 2018. Disponivel em: <https://mundoeducacao.bol.uol.com.br/fisica/corrente-alternada.htm>. Acesso em: Agosto 2018.

SILVEIRA, F. L. D.; VARRIALE, M. C. O rolamento freado do magneto na rampa: uma interessante aplicação da lei de Faraday-Lenz. **Revista Brasileira de Ensino de Física**, Porto Alegre, v. 31, n. 4, p. 4303, 2009.

SOUZA, E. L. D. S. **Redes de Comunicação Industrial**. Universidade São Francisco. Campinas, p. 40. 2010.

SOUZA, F. Arduino UNO. **Embarcados**, 2013a. Disponivel em: <https://www.embarcados.com.br/arduino-uno/>. Acesso em: Julho 2018.

SOUZA, L. B. D. **Redes de Computadores:** Guia Total. 1. ed. São Paulo: Érica Ltda, 2013b.

SSPREDITIVA. **SS Preditiva**, 2018. Disponivel em: <http://sspreditiva.com.br/>. Acesso em: 2018.

TERRA. Segundo projeção, até 2020 cerca de 50 bilhões de dispositivos estarão conectados à internet. **Terra**, 2017. Disponivel em: <https://www.terra.com.br/noticias/dino/segundo-projecao-ate-2020-cerca-de-50-bilhoes-de-dispositivos-estarao-conectados-a-internet,3c5324c3a9226e95fb14590df0973620fyajrwwb.html>. Acesso em: Julho 2018.

TÓFOLI, R. J. **Casa Inteligente – Sistema De Automação Residencial**. Fundação Educacional do Município de Assis. Assis, p. 74. 2014.

UNICONTROL. **Curso PLC Siemens Módulo Básico Usando o Software STEP 7**. São Paulo: [s.n.], 2007.

APÊNDICE 1

PROGRAMAÇÃO ARDUINO

Neste apêndice será apresentada toda a programação, com comentários, utilizada no Arduino. Para utilização da mesma, esta deve ser digitada no software IDE e efetuado download para a placa Arduino.

```
//PROJETO ARDUINO, APLICAÇÃO DA INDUSTRIA 4.0 NA MANUTENÇÃO PREDITIVA DE EQUIPAMENTOS CRÍTICOS DE UMA PLANTA ATRAVÉS DE TECNOLOGIA IoT (INTERNET OF THINGS)
//TRABALHO DE CONCLUSÃO DE CURSO
//ENGENHARIA ELÉTRICA
//UniFOA
//ALUNOS: BRUNO MIRANDA SABENÇA E HENRIQUE GUIMARÃES PEREIRA
//ANO: 2018

#include <Ethernet.h>                 // Inclui a Biblioteca do shield Ethernet
#define S7WIRED                       // Define comunicação Ethernet via Cabo
#include <Settimino.h>                // Inclui a Biblioteca da comunicação com Step7
#include <dht.h>                      // Inclui a Biblioteca do Sensor DHT22 de Temperatura e Umidade
#define dht_dpin A1                   // Define que o Pino DATA do Sensor DHT22 está ligado na porta analógica A1 do Arduino
dht DHT;                              // Inicializa o sensor DHT22
#include "EmonLib.h"                  // Inclui a biblioteca para o sensor de corrente SCT013
EnergyMonitor SCT013                  // Cria o objeto SCT013 para o sensor de corrente atrelando este objeto à biblioteca
#define pin_sct A0                    // Define que o sensor de corrente está ligado na porta analógica A2 do Arduino
//--------------------------------------------------------------------
//Configurações da placa Ethernet
//--------------------------------------------------------------------
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };     // Endereço MAC da placa do Arduino
IPAddress ip(172,26,222,211);                            // Endereço IP do Arduino
IPAddress plc(172,26,222,164);                           // Endereço IP do PLC
IPAddress gateway(172,26,222,129);                       // Endereço gateway padrão da rede
IPAddress subnet(255,255,255,128);                       // Endereço da Máscara de sub-rede
//--------------------------------------------------------------------
//Configuração do pacote a ser trocado com o PLC
//--------------------------------------------------------------------
int DBNum = 1;                        // Número da DB a ser Lida/Escrita no PLC
byte Buffer[14];                      // Número de Bytes a serem trocados com o PLC
S7Client Client(_S7WIRED);            // S7Client cria um Cliente Ethernet via Cabo TCP/IP
```

```
//---------------------------------------------------------------------
// Setup: Inicialização das portas Serial e Ethernet
//---------------------------------------------------------------------
void setup() {
   Serial.begin(9600);                          // Inicializa a Porta Serial para utilização do monitor serial
   Serial.println("Verificando Conexão..");     // Escreve no monitor serial. Iniciando programa
   Ethernet.begin(mac,ip,gateway,subnet);       // Inicializa a Biblioteca Ethernet
   delay(2000);                                 // Aguarda 2 Segundos
   Serial.println("");                          // Pula uma linha em branco no monitor serial
   Serial.print("Endereço IP do Arduino: "); // informa no monitor serial o endereço IP atribuído ao arduino
   Serial.println(Ethernet.localIP());
   SCT013.current(pin_sct, 60);                 //calibração do sensor de corrente = Relação de espiras no secundário/resistor de carga --> 2000/33 = 60
}
//---------------------------------------------------------------------
// Configuração para conexão com o PLC
//---------------------------------------------------------------------
bool Connect()
{
   int Result=Client.ConnectTo(plc,
                               0,               // Rack da CPU do CLP a ser conectado
                               3);              // Slot da CPU do CLP a ser conectado
   Serial.print("Tentando Conectar ao PLC IP: ");   // Escreve no monitor serial o endereço IP do PLC
   Serial.print(plc);
   Serial.println(" ...");
   if (Result==0)                                  // Para uma conexão bem-sucedida do resultado deve ser igual a 0.
   {
     delay(2000);                                  // Após ”result” igual a 0, aguarda 2 segundos
     Serial.println("=============================================");   //Escrevendo no monitor serial
     Serial.println("CONECTADO!");                 // Informa no monitor serial que a conexão com o PLC foi bem-sucedida
   }
   else
     Serial.println("Erro de Conexão");            // Caso o resultado da conexão seja diferente de 0, informa no monitor serial que houve erro de conexão
   return Result==0;
}
```

```
//---------------------------------------------------------------------
// Configurações da Leitura da DB do PLC
//---------------------------------------------------------------------
void Dump(void *Buffer, int Length)
{
  int i, cnt=0;
  pbyte buf;

  if (Buffer!=NULL)
    buf = pbyte(Buffer);
  else
    buf = pbyte(&PDU.DATA[0]);

  Serial.print("Lendo do PLC: ");         // Informa no monitor serial, o tamanho e bytes da DB lida do PLC
  Serial.print(Length);
  Serial.print(" bytes ");
  Serial.print(" da DB");
  Serial.println(DBNum);

  for (i=0; i<Length; i++)
  {
    cnt++;
    if (buf[i]<0x10)
    Serial.print("0");
    Serial.print(buf[i], HEX);
    Serial.print(" ");
    if (cnt==16)
    {
      cnt=0;
      Serial.println();
    }
  }
}
//---------------------------------------------------------------------
// Check de ERRO
//---------------------------------------------------------------------
void CheckError(int ErrNo)
{
  Serial.print("Error No. 0x");
  Serial.println(ErrNo, HEX);
```

```
  if (ErrNo & 0x00FF)
  {
    Serial.println("ERRO GRAVE, desconectando...");
    Client.Disconnect();
  }
}
void TesteConex()                    // Rotina para enviar sinal pulsante ao PLC e poder verificar a
conexão
{
  int ResultWrite;                   // Cria a variável inteira "ResultWrite"
  bool pulso = HIGH;                 // Cria a variável booleana "pulso" e atribui nível lógico igual a 1
  ResultWrite = Client.WriteArea(S7AreaDB,      // Escrevendo no PLC
                        DBNum,                  // Número da DB definida no início da programação
                        12,                     // Byte a ser escrito na DB
                        1,                      // Tamanho do pacote enviado
                        &pulso);                // Variável enviada
  delay(500);                                   // Aguarda 500 ms
  pulso = LOW;                                  // Atribui nível logico igual a 0 à variável "pulse"
  ResultWrite = Client.WriteArea(S7AreaDB,      // Escrevendo no PLC
                        DBNum,                  // Número da DB definida no início da programação
                        12,                     // Byte a ser escrito na DB
                        1,                      // Tamanho do pacote enviado
                        &pulso);                // Variável enviada
  delay(100);                                   // Aguarda 100 ms
}
//--------------------------------------------------------------------
// Rotina de Loop Principal (Programa Principal)
//--------------------------------------------------------------------
void loop()
{
  DHT.read22(dht_dpin);                 // Lê as informações do sensor DHT22
  float Corrente = SCT013.calcIrms(1480);   // Obtém o valor de corrente do sensor SCT013 a partir
de 1480 amostras como variável real (float)
  float Umidade = DHT.humidity;         // Obtém o valor de umidade do sensor DHT22 como
variável real (float)
  float Temperatura = DHT.temperature;  // Obtém o valor de temperatura do sensor DHT22 como
variável real (float)
  int Result;                           // Define a variável inteira
  int ResultWrite;                      // Define a variável inteira
```

```
  float Size = sizeof(Buffer);                    // Define a variável real (float) e atribui seu tamanho ao
mesmo tamanho da variável buffer
  if (Corrente<0.4)
  Corrente = 0;
  //---------------------------------------------------------------------
  // Programação para mostrar no monitor serial, o valor atual das variáveis lidas pelo Arduino
  //---------------------------------------------------------------------
  Serial.println("============================================");
  Serial.println("Variáveis lidas pelo Arduino:");   // Escrevendo no monitor serial
  Serial.print("Umidade = ");
  Serial.print(DHT.humidity);
  Serial.println(" %  ");
  Serial.print("Temperatura = ");
  Serial.print(DHT.temperature);
  Serial.println(" °C  ");
  Serial.print("Corrente = ");
  Serial.print(Corrente);
  Serial.println(" A ");
  Serial.println("============================================");

  //---------------------------------------------------------------------
  // Execução da rotina de escrita e leitura no PLC somente se estiver conectado
  //---------------------------------------------------------------------
  while (!Client.Connected)
  {
    if (!Connect())
       delay(100);
  }

  Serial.print("Conectando à DB");              //Informa no monitor serial qual DB está sendo conectada
  Serial.print(DBNum);
  Serial.println("...");

  TesteConex();                                  // Executa a função para enviar sinal de teste de conexão

  Serial.println("============================================");
  Serial.print("Valores enviados ao CLP: ");
  Serial.println(Umidade);                       // Informa no monitor serial o valor da variável de umidade
que o Arduino está enviando
  Serial.print("                ");
```

```
  Serial.println(Temperatura);                // Informa no monitor serial o valor da variável de temperatura
que o arduino esta enviando
  Serial.print("                ");
  Serial.println(Corrente);                   // Informa no monitor serial o valor da variável de corrente que
o arduino esta enviando
  Serial.println("=============================================");

  //---------------------------------------------------------------
  //Converte as variáveis de Litle Endian(Arduino) para Big Endian(PLC) para serem compreendidas
pelo PLC
  //---------------------------------------------------------------
  Umidade = S7.FloatAt(&Umidade,0);           // Conversão da variável Umidade
  Temperatura = S7.FloatAt(&Temperatura,0);   // Conversão da variável Temperatura
  Corrente = S7.FloatAt(&Corrente,0);         // Conversão da variável Corrente

  //---------------------------------------------------------------
  //Escrevendo as variáveis no CLP
  //---------------------------------------------------------------
  //variável 1
  ResultWrite = Client.WriteArea(S7AreaDB,    // Solicitação de acesso a DB
                        DBNum,                // Número da DB
                        0,                    // Byte a ser Escrito na DB
                        sizeof(float),//       // Tamanho do Pacote Enviado
                        &Umidade);            // variável Enviada
  //Variavel 2
  ResultWrite = Client.WriteArea(S7AreaDB,    // Solicitação de acesso a DB
                        DBNum,                // Número da DB
                        4,                    // Byte a ser Escrito na DB
                        sizeof(float),        // Tamanho do Pacote Enviado
                        &Temperatura);        // Variável Enviada
   //Variavel 3
  ResultWrite = Client.WriteArea(S7AreaDB,    // Solicitação de acesso a DB
                        DBNum,                // Número da DB
                        8,                    // Byte a ser Escrito na DB
                        sizeof(float),        // Tamanho do Pacote Enviado
                        &Corrente);           // Variável Enviada
  //---------------------------------------------------------------
  //Lendo a DB do PLC
  //---------------------------------------------------------------
  Result=Client.ReadArea(S7AreaDB,            // Solicitação de acesso a DB
```

```
                    DBNum,                  // Número da DB
                    0,                      // Inicia a leitura no Byte Nº?
                    Size,                   // Bytes a serem lidos
                    &Buffer);               // Escreve no Buffer

  if ((Result==0) && (ResultWrite==0))
  {
    Dump(&Buffer, Size);
  }
  else
    CheckError(Result);
  Serial.println(" ");
  Serial.println("================================================");

  delay(200);
}
```