JEFERSON AFONSO LOPES DE SOUZA

CONTROLE AUTOMÁTICO DE VELOCIDADE DE UM "COATER" MÁQUINA DE REVESTIMENTO DE PAPÉIS ESPECIAIS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de graduação em Engenharia

Área de Concentração:
Engenharia Mecatrônica

Orientador:

Prof. Dr. Lucas Moscato

SÃO PAULO

2011

FICHA CATALOGRÁFICA

Souza, Jeferson Afonso Lopes de
Controle automático da velocidade de um "coater": máquina de revestimento de papéis especiais / J.A.L. de Souza. –São Paulo, 2011. 103 p.

Trabalho de Formatura – Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1. Engenharia Mecânica 2. Mecânica aplicada 3. Meca Trônica 4. Indústria de celulose e papel I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II.t.

AGRADECIMENTOS

O autor deseja expressar seus mais sinceros agradecimentos:

Ao Prof. Lucas Moscato, pelo acolhimento à proposta do trabalho e orientação no decorrer do projeto.

Ao Prof Guenther Carlos Krieger Filho, pela orientação e ajuda no modelo e cálculos térmicos.

Aos Professores Tarcísio Antônio Hess Coelho e Fabio Gagliardi Cozman, pela oportunidade de continuidade do curso, fundamental e que nunca será esquecida.

Ao Prof Marcos Ribeiro Pereira Barretto, pela ajuda na escolha do tipo de controle.

Ao colega Felipe Langelotti, pela ajuda fundamental no desenvolvimento do software.

Aos meus pais, Daltro e Adelfina, e meu irmão Rafael, pela eterna ajuda.

À Carina, pela paciência ímpar e companheirismo.

ÍNDICE

ÍNDICE DAS FIGURAS

Resumo

Papéis Térmicos, Autocopiativos e couché são produzidos através da aplicação de uma tinta específica para cada tipo sobre um papel base – daí o nome de revestimento. Dependendo da gramatura final desejada aplica-se mais ou menos tinta, ou ainda diminui-se a sua viscosidade. A tinta geralmente é à base d´água, e teores de sólidos típicos estão na faixa de 30% a 46%. O papel em seguida sofre um processo de secagem por convecção forçada com jatos de ar a aproximadamente 250º C no interior de um forno, ocorrendo o processo de evaporação da água. A seguir o papel é rebobinado, reconstituindo as propriedades mecânicas da bobina base. A questão deste trabalho envolve a velocidade de revestimento, ou o controle da velocidade com que o papel percorre o túnel de secagem, na medida em que se sua velocidade for alta, e portanto a secagem não ocorrer completamente, o papel rebobina úmido, comprometendo sua qualidade. De forma oposta, se o papel percorrer o túnel de forma lenta, o papel fica muito seco, o que gera estática elétrica no rebobinador e também o fenômeno do "encanoamento", que é a dobra das laterais do papel nos cilindros de passagem, e comprometendo completamente a qualidade do produto na medida em que rebobina dobrado no sentido longitudinal, além de causar choques no operador da máquina. Geralmente o controle de velocidade é feito através da medição contínua da umidade do papel à saída da secagem por raios infravermelho. No entanto, devido aos altos custos deste equipamento, será apresentada neste trabalho uma solução de controle através da medida de temperatura do papel como medição indireta da umidade. O sistema será instalado em uma máquina de pequeno porte, e inicialmente será feito um levantamento térmico do processo, as características da planta e a escolha do sistema de controle mais adequado. Testes de controle serão realizados em um protótipo e, ao final do trabalho, serão apresentadas as conclusões e sugestões para a continuidade do projeto.

Abstract

Thermal, carbonless and couché papers are produced by applying a specific ink on a base paper – hence the name of coating. Depending on the desired weight applies more or less ink, or decreases the viscosity. The ink is generally water-based, and solids are typically in the range of 30% to 46%. The paper then undergoes a process of drying by forced convection using air jets to about 250ºC inside an oven, place the process of evaporation of water. The following paper is rewound, restoring the mechanical properties of the coil base. The point of this work involves the coating speed, or control the speed at which the paper travels through the drying tunnel, in that if your speed is high, and therefore does not completely dry, wet the paper rewind, compromising its quality. Conversely, if the paper go through the tunnel slowly, the paper is too dry, which generates static electricity in the winder and also the phenomenon of curling, which is the fold of the paper side of the cylinder passage, and completely compromising the quality of the product as it rewinds folded longitudinally and cause shocks to the machine operator. Usually the speed control is done by continuous measurement of moisture in the paper off the drying by infrared ray. However, due to the high costs of this equipment will be presented in this paper a control solution by measuring the temperature of the coated paper as indirect measurement of moisture. The system will be installed on a small factory and initially will be a survey of the thermal coating process, plant characteristics and choose the most appropriate control system. Control tests are performed on a prototype and, at the end of this work, we present the conclusions and suggestions for continuing the project.

SÍMBOLOS

D	Diâmetro, m
A_r	Área do bico, m^2
$A_{c,e}$	Área da seção transversal da saída do bico, m^2
A_{cel}	Área da superfície da célula, m^2
H	Altura do jato, m
ν	viscosidade cinemática, m^2/s
k	condutividade térmica, $W/m.K$
Pr	Número de Prandtl
ρ	massa específica, Kg/m^3
D_{AB}	coeficiente de difusão binária de massa, m^2/s
q''	fluxo de calor, W/m^2
h	coeficiente de transferência de calor por convecção, $W/m^2.K$
T_s	Temperatura da superfície, K
T_e	Temperatura ao longe, K
N_A''	fluxo de massa da espécie A, $Kg/s.m^2$
h_m	coeficiente de transferência de massa por convecção, m/s
$C_{A,E}$	concentração molar ao longe, $kmol/m^3$
$C_{A,s}$	concentração molar na superfície, $kmol/m^3$
Nu	número de Nusselt
Sh	número de Sherwood
Sc	número de Schmidt
Re	número de Reynolds
L	Comprimento característico, m
u_∞	velocidade do fluido, m/s

1 INTRODUÇÃO

Papéis especiais são fabricados por um processo de recobrimento, revestimento ou "coating" de uma tinta específica sobre um papel base. Como papel especial podemos citar o autocopiativo, popularmente conhecido como "carbonado" utilizado em grande escala nos anos 80 e 90 na confecção de talonários de notas fiscais, o papel couchê, utilizado em folhetos publicitários e rótulos adesivos e o papel térmico, utilizado nos aparelhos de fax e mais recentemente nos cupons de pedágio, passagens aéreas e cupons fiscais.

O processo de revestimento pode ser feito "on machine", em que o revestimento é feito na mesma linha em que o papel é fabricado a partir da celulose, ou "off machine" em que uma bobina jumbo de papel base, este proveniente da celulose, é colocado no "coater" para o recebimento do revestimento.

Máquinas de grande porte possuem velocidades de revestimento em torno de 1.000m/min, com uma produção mensal de 3.200 tons/mês. São equipamentos caros, da ordem de US$ 4.000.000,00, e os maiores fabricantes nacionais são a VCP (Votorantim Celulose e Papel), ou Fibria, com 80% de participação do mercado.

Outras aplicações desta máquina são na fabricação de auto-adesivos, destacando-se 3M e Avery-Denisson.

No nosso estudo iremos nos aprofundar no papel térmico, pois observou-se um grande incremento no consumo deste tipo de papel nos últimos anos devido ao avanço da automação comercial e bancária, que utiliza este tipo de papel na emissão de comprovantes, e também nos tíquetes de passagens aéreas. No entanto podemos aplicar os conceitos aqui apresentados para todos os tipos de papéis revestidos, fabricados "off machine".

Uma máquina de revestimento de papéis possui os seguintes principais elementos: desbobinador, cozinha para preparo da tinta, estação de aplicação e dosagem da tinta, túnel de secagem, secadores, ventiladores de circulação de ar, calandra e rebobinamento.

Uma idéia geral do processo pode ser resumida na aplicação de uma determinada

quantidade de tinta por m² de papel base, que determina a gramatura final do produto.

Esta tinta deve passar por um processo de secagem contínua num forno de modo a produzir a evaporação do veículo, geralmente água, ficando portanto somente as partículas sólidas da tinta adicionadas ao papel base. Preferencialmente o papel revestido deve apresentar uma umidade próxima ao papel base após a secagem, e este é justamente o parâmetro que iremos explorar neste texto.

Após a secagem o papel normalmente passa por um processo de laminação ou calandragem, de modo a normalizar sua superfície, restabelecer a espessura original e melhorar a propriedade de lisura e brilho.

Finalmente passa por um processo de rebobinamento em que devem ser observadas a tensão e alinhamento do papel, de forma a se ter um produto final com bom acabamento.

2- OBJETIVO

Este trabalho tem por objetivo automatizar a velocidade com que o papel atravessa o túnel de secagem para uma determinada gramatura de revestimento, de forma a se obter uma qualidade constante do produto acabado em termos de umidade do produto.

Um problema constante no processo é que se o papel passa lentamente sobre o túnel de secagem ocorre uma secagem excessiva, ocorrendo estática elétrica na bobina jumbo, ocasionando choques no operador, e também a ocorrência de “encanoamento” ou “curling” do papel, ou seja, as laterais do papel se dobram na passagem dos cilindros da máquina, comprometendo o rebobinamento final, obrigando a parada do processo, além de se observar a revelação precoce do papel térmico. A imagem formada num papel térmico é justamente a revelação por temperatura dos caracteres quando uma fonte térmica (cabeça térmica das impressoras) é acionada próxima do papel.

Caso o papel passe com uma velocidade muito alta, ocorre o descolamento ou delaminação do revestimento na calandra, além de se obter uma bobina jumbo úmida, que também compromete a qualidade final do produto.

Desta forma, o objetivo o trabalho é implementar um sistema automático que verifique se o papel após a saída do forno está em um set-point de umidade próximo ao do papel base original, independentemente do operador da máquina, e que este sensor alimente um controlador que irá comandar a velocidade com que o papel percorra o túnel de secagem.

3- FUNCIONAMENTO BÁSICO DE CADA ELEMENTO

Para o entendimento do objeto a ser controlado, será feita uma explicação básica da contribuição de cada componente da máquina e sua relação com a velocidade de secagem do papel. Verificaremos quais componentes se relacionam com a secagem e portanto dependentes da ação de controle e quais componentes têm seu funcionamento independente, ou seja, não precisam ser controlados pois qualquer que seja a velocidade do papel no túnel de secagem seu funcionamento não compromete o produto.

3.1 – DESBOBINADOR

O desbobinador é um cavalete em que um eixo suportado por mancais é ligado a um freio a disco pneumático cujo controle de intensidade é feito diretamente no painel. Este eixo suporta a bobina jumbo de papel base a ser revestida. Normalmente é utilizada nesta máquina um formato de 0,5m de largura por 1 metro de diâmetro, de gramatura 50 g/m2, com aproximadamente 300 Kg e 12.000 metros lineares.

Ela se movimenta solidária ao eixo através de placas que se “inflam” através do enchimento com ar comprimido de uma câmara de ar interna ao eixo, fazendo com que se pressionem contra o tubete interno da bobina jumbo.

Fig 3.1 – Desbobinador do papel base

O freio serve para se dar uma pré-tensão no papel para evitar que com o desenrolar da bobina o papel fique frouxo, o que pode resultar em dobras ao longo dos cilindros de passagem. Em outras palavras, o freio serve para manter o papel esticado durante o processo. Não depende da velocidade da máquina e a tensão do freio deve ser aliviada conforme ocorre o desbobinamento. A mesma tensão de freio para uma menor inércia provoca uma elevada tensão no papel, podendo romper-se no momento do revestimento, aquoso. Há sistemas automáticos que aliviam a tensão do freio conforme sua inércia diminui, porém não foi implementado nesta máquina devido ao reduzido número de regulagens feitas no processo, ao redor de três até o final do processo. A tensão do freio não depende da velocidade da máquina e sim do diâmetro (massa) da bobina, portanto é uma variável independente.

3.2 – APLICADOR E DOSADOR DE TINTA

No processo de aplicação e dosagem da tinta, um rolo pescador retira tinta em excesso da cuba, aplicando no papel, que passa a uma velocidade maior do que a do rolo aplicador. O excesso de tinta é retirado através de um dispositivo chamado de “air knife” ou faca de ar, juntamente com barras dosadoras “metering bars” em que o excesso de tinta é retirado e devolvido à cuba, ficando no papel somente a quantidade necessária de tinta que resulta numa gramatura final de papel desejada.

A cuba é alimentada por uma bomba radial, ligada a um grande reservatório de tinta ou “batch”. A circulação de tinta é tal que a cuba sempre fica cheia, mesmo que a velocidade da máquina seja máxima. O excesso de bombeamento é esgotado por um ladrão por gravidade, assim como o excesso retirado pela faca de ar, ambos retornando ao “batch”. Desta forma garantimos que o revestimento nunca terá falhas. A faca de ar é alimentada por um compressor radial que também possui potência suficiente para retirar o excesso mesmo com uma alta velocidade. E as barras dosadoras, por se tratarem de elementos ou obstáculos físicos - o excesso é retirado por raspagem, em que um “gap” determinado é mantido constante para a espessura ou gramatura de papel requisitada, sendo independentes da velocidade com que o papel passa por esses elementos. Podemos concluir que este sistema também é independente da velocidade da máquina.

Fig 3.2 – Estação de aplicação e dosagem da tinta de revestimento

3.3 – TÚNEL DE SECAGEM / SECADORES

Nesta parte da máquina é feita a secagem ou retirada de água da tinta aplicada no papel, permanecendo somente os sólidos que compõe a formulação específica para cada tipo de papel especial. Retira-se água através da convecção forçada de ar quente contra o papel, com o objetivo de restabelecer a umidade do papel base, porém agora revestido.

O ar quente pode ser produzido por inúmeros processos térmicos, sendo os mais comuns caldeira que aquece óleo térmico e este percorre serpentinas ao longo do forno. Ventiladores forçam a troca de calor entre o ar a temperatura ambiente e as serpentinas resultando em ar quente que é insuflado contra o papel ao longo do forno. Resistências elétricas possuem funcionamento análogo, porém não são muito utilizadas devido ao alto custo da energia elétrica. Queimadores a gás são normalmente utilizados e foram instalados nesta máquina.

São alimentados por GLP – Gás Liquefeito de Petróleo por caminhão bob-tail e são reabastecidos sem a necessidade de troca dos cilindros. É necessário o cálculo da potência

térmica dos queimadores para verificar a quantidade de cilindros necessária para que a taxa de evaporação do gás de cada cilindro seja suficiente para suprir a demanda dos queimadores. O princípio é queima do GLP com ar, que resulta num ar quente e seco, que é injetado no forno através de ventiladores que sopram o ar quente sobre o papel durante o processo de secagem.

Fig 3.3 – Queimador principal – capacidade 220.000 Kcal/h

Há sensores de temperatura ao longo da máquina e pode-se programar os queimadores para operarem numa determinada faixa de temperaturas, geralmente em torno de 200º C no início do forno e em torno de 120º ao final do forno. Geralmente os queimadores são de dois estágios de chama, em que ao se atingir o set de temperatura a chama alta é desligada, permanecendo somente a chama baixa. A circulação de ar é bastante importante, assim como a constante injeção de ar pelos queimadores, pois deve-se evitar a saturação do ar no interior do forno, o que seria muito prejudicial para o nosso processo.

O comprimento do forno também é importante pois tem relação direta com a velocidade de secagem/produção do papel. O cálculo da carga térmica será detalhado mais adiante, porém tamanhos típicos de fornos são de 12 a 25 metros de comprimento, variando em

função da gramatura final de revestimento, potência térmica instalada e velocidade de máquina requerida.

Fig 3.4 – Túnel de secagem – comprimento: 20 metros

Por tratar da secagem, possui relação direta com a velocidade da máquina, sendo portanto uma variável dependente do processo.

3.4 – CALANDRA

A calandra é responsável pelo acabamento superficial do papel para a melhoria das propriedades de brilho, lisura e compactação do revestimento no intuito de se preservar sua espessura original. É composta de três cilindros, portanto com dois “nips” de atuação, e os cilindros são pressionados através de quatro cilindros pneumáticos, dois para o cilindro superior e dois para o intermediário. O cilindro base recebe o pressão dos dois cilindros superiores e podem se movimentar através do acionamento de válvulas solenoides que pressionam/ elevam os cilindros. A elevação é necessária para a troca das bobinas ou quando ocorrem paradas de processo por por exemplo rompimento do papel.

A calandra é o elemento responsável pela "puxada" do papel na máquina. È ela quem deve vencer a tensão do freio do desbobinador e a inércia do papel ao longo da máquina, ditando portanto como o papel passa ao longo do forno de secagem.

Fig 3.5 – Calandra, motor de acionamento e cavalete rebobinador

Seu controle é feito através de um motor trifásico controlado por um inversor de frequência e sua velocidade atualmente é controlada pelo operador através de um potenciômetro. Este é justamente o objeto que desejamos controlar, ou seja, substituir o potenciômetro por um sistema que controle a velocidade da máquina através de um set-point de umidade ideal que desejamos atingir para o nosso produto. É portanto uma variável dependente.

Fig 3.6 – Detalhe da calandra com pistões pneumáticos

Fig 3.7 – Calandra com pistões elevados para passagem do papel

3.5 – REBOBINADOR

O sistema de rebobinamento instalado consiste em um cavalete onde um eixo semelhante ao do desbobinador é alimentado por um tubete de papelão vazio que serve de base para o rebobinamento de uma nova bobina de papel revestido. Este eixo é ligado a um semi-eixo (o eixo da bobina deve ser livre para a retirada do produto final) que por sua vez é solidário a uma polia que é ligada a um motor através de correias. Este motor é um motor de 8 pólos, portanto baixa rotação e elevado torque, e seu comando é realizado através de um inversor de frequência operando no modo torque, através de realimentação de sinal de posição por um encoder solidário ao eixo do motor.

Fig 3.8 – Detalhe do encoder do motor do rebobinador

Em um dos cilindros de passagem do rebobinador estão instaladas duas células de carga que servem para medir a tensão com que o papel está sendo rebobinado, independentemente da velocidade da calandra, ou seja, o rebobinador está sempre “esticando” o papel, com uma determinada tensão, e esta tensão é suportada pelo motor que aciona a calandra, não ocorrendo o escorregamento deste motor. Ou seja, a calandra

opera a velocidade constante determinada pelo operador, de forma que o rebobinador opere de forma independente.

Fig 3.9 – Motor do rebobinador sobre rolimãs e pistão hidráulico do alinhador lateral.

O sinal captado pela célula de carga é enviado a um controlador, em que definimos a tensão do papel desejada. Podemos determinar num potenciômetro deste controlador um valor numérico de unidade [kgf] para a tensão de nossa bobina revestida.

Este controlador envia como saída um sinal de comando ao inversor de frequência que controla o motor do rebobinador, realizando seu controle de forma que o papel revestido tenha sempre a mesma tensão de forma independente ao diâmetro da bobina e da velocidade da máquina.

Há ainda um segundo controle, este porém de alinhamento do papel, em que um sensor capta a posição do papel em relação ao tubete do rebobinador e envia este dado para uma

servo-válvula ligada a uma bomba hidráulica. Dependendo da posição em que o papel está, a bomba hidráulica aciona um pistão a óleo na base do cavalete que por sua vez movimenta todo o conjunto. Ou seja, todo o conjunto de motor, eixo, rebobinador está solidário ao eixo do pistão hidráulico e apoiado sobre mancais de rolamento que podem se movimentar no sentido transversal da máquina de forma a corrigir o rebobinamento e garantir uma boa qualidade no alinhamento lateral da bobina.

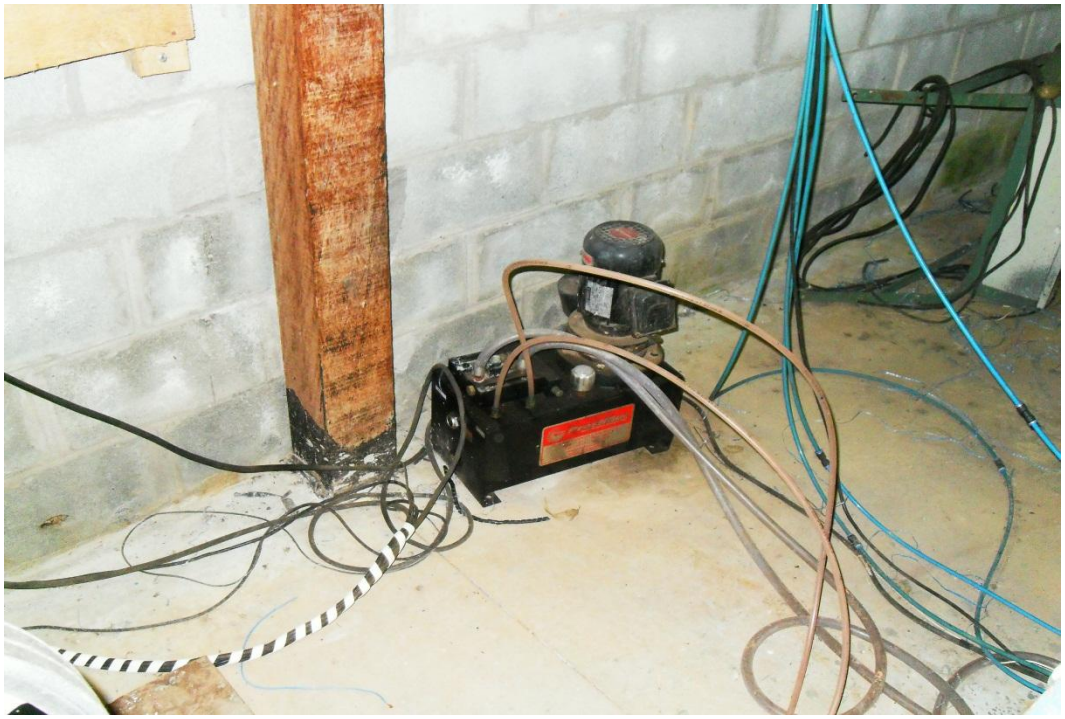


Fig 3.10 – Unidade hidráulica do alinhador lateral

Desta forma garantimos um produto com tensão constante de rebobinamento e alinhamento de sua lateral, de forma independente da velocidade com que o papel atravessa o túnel de secagem, ou seja, é uma variável independente.

Fig 3.11 – Quadro de comando do inversor do rebobinador e controle de tensão da célula de carga

4- FORMAS DE CONTROLE

Neste item serão apresentadas quatro soluções para o sistema de controle e sua viabilidade de aplicação, principalmente relacionado ao custo.

São três sistemas que trabalham com a medição de umidade do papel na saída do túnel de secagem e a comparação contínua deste valor com o set de umidade do fabricante do papel base, que consta em seu laudo técnico.

Dois sistemas utilizam um sensor infra-vermelho de reflexão que mede continuamente a umidade em uma das bordas do papel. Admitimos que a secagem seja uniforme no sentido transversal do papel, o que é bastante razoável, para permitir a adoção desta solução, pois os sensores são pontuais. O sistema coleta o dado e converte num sinal de 4mV a 5 V, possuindo também saídas digitais seriais do tipo RS232.

Uma outra solução utiliza micro-ondas para a medição de umidade e também possui as

saídas mencionadas na solução anterior.

Uma outra possível solução é a medição indireta da umidade através da medição da temperatura do papel na saída do forno. Para isso utilizamos um termômetro contínuo infravermelho com mira laser, em que os dados são enviados por um controlador que processa o sinal e resulta numa saída de 4mV a 5V para cada temperatura.

A ênfase para saídas analógicas tem o motivo pelo qual o controle do inversor de frequência da calandra é dado por sua entrada analógica, que possui o mesmo range. Desta forma podemos implementar diretamente o sinal de saída de nosso controlador na entrada do inversor, bastando obviamente uma calibração de escala para a variação de velocidade para cada valor de umidade ou temperatura.

A seguir temos o detalhamento de cada solução e a apresentação de suas vantagens e desvantagens para a escolha da solução final a ser implementada.

4.1 CONTROLE DE UMIDADE POR MICROONDAS

Neste sistema de controle a variável controlada é a umidade absoluta do material, no caso papel, e é feito de forma contínua por um sensor que emite micro-ondas e recebe um sinal por reflexão. Este valor é comparado ao set de umidade desejada e o controlador possui tanto saídas analógicas quanto seriais digitais. É utilizado para o monitoramento e controle na secagem de grãos, ração animal, papéis, madeira etc. É de origem alemã e possui um representante no país. Atende nossa aplicação, porém possui custo elevado de implementação, da ordem de R$ 50.000,00 (Cinquenta mil reais).

A seguir temos uma foto do aparelho que ilustra melhor sua aplicação:

Fig 4.1- Sensor de microondas monitorando a umidade de madeira. Extraído de [5].

Fig 4.2 – Detalhe do sensor e painel de comando. Exemplos de aplicação. Extraído de [5].

4.2 – CONTROLE DE UMIDADE POR INFRAVERMELHO

Nestes equipamentos a medição de umidade é feita através de um sensor infravermelho por reflexão, que de acordo com a intensidade de sinal refletido tem-se uma determinada umidade. Duas opções foram encontradas, uma de origem inglesa e outra de origem chinesa. Não foram encontradas opções nacionais, somente representantes destas empresas estrangeiras.

4.2.1 – INFRAVERMELHO NDC

Esta opção, de origem inglesa, é a mais completa e sofisticada, realizando a medição não somente da umidade mas também da gramatura final do produto.

Como aplicações, além do papel, também podemos citar a indústria de autoadesivos, em que a aplicação de “cola” deve ser muito precisa, com o objetivo do adesivo não se soltar com facilidade, mas também ser destacável de seu papel base pelo consumidor. Logo o monitoramento da quantidade de adesivo a ser aplicada por metro quadrado deve ser feito de forma precisa, sem falar na economia de custos que o sistema proporciona. Tal sistema foi implementado em grandes empresas instaladas no país, como por exemplo na 3M.

O custo novamente é um empecilho para a nossa aplicação, da ordem de R$ 60.000,00 fora impostos de importação.

Possui saídas analógicas, digitais, e também um software em plataforma Windows para o monitoramento e implementação do controle.

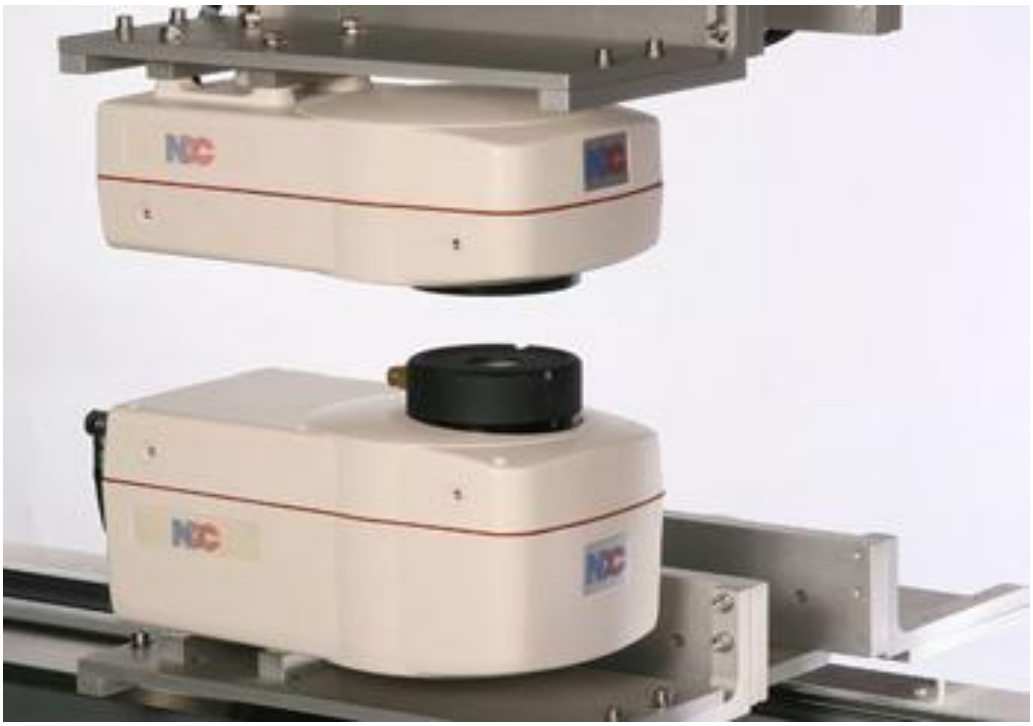


Fig 4.3 – Foto do sensor infravermelho por transmissão. Papel passa entre os sensores. Extraído de [6]

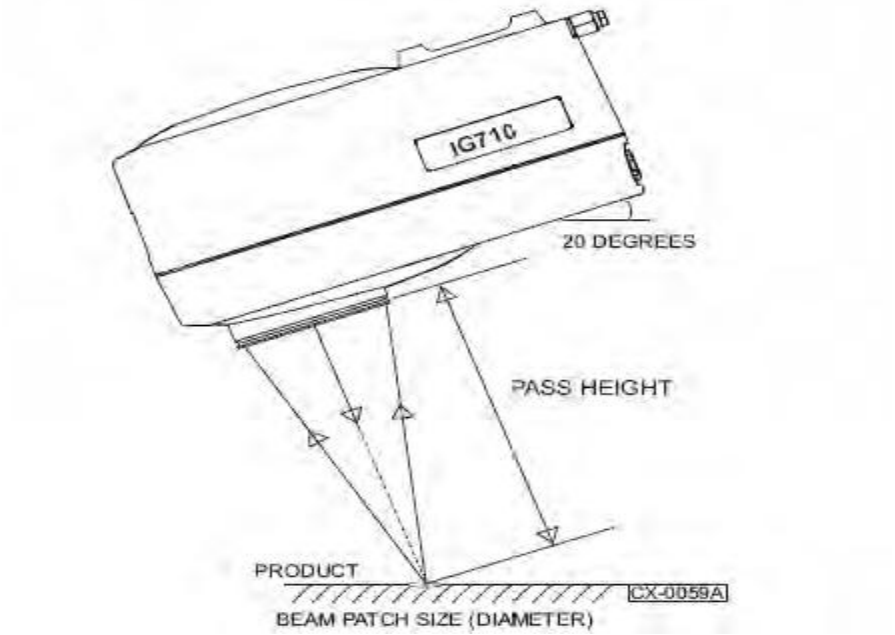


Fig 4.4 – Esquema do sensor infravermelho por reflexão. Extraído de [6].

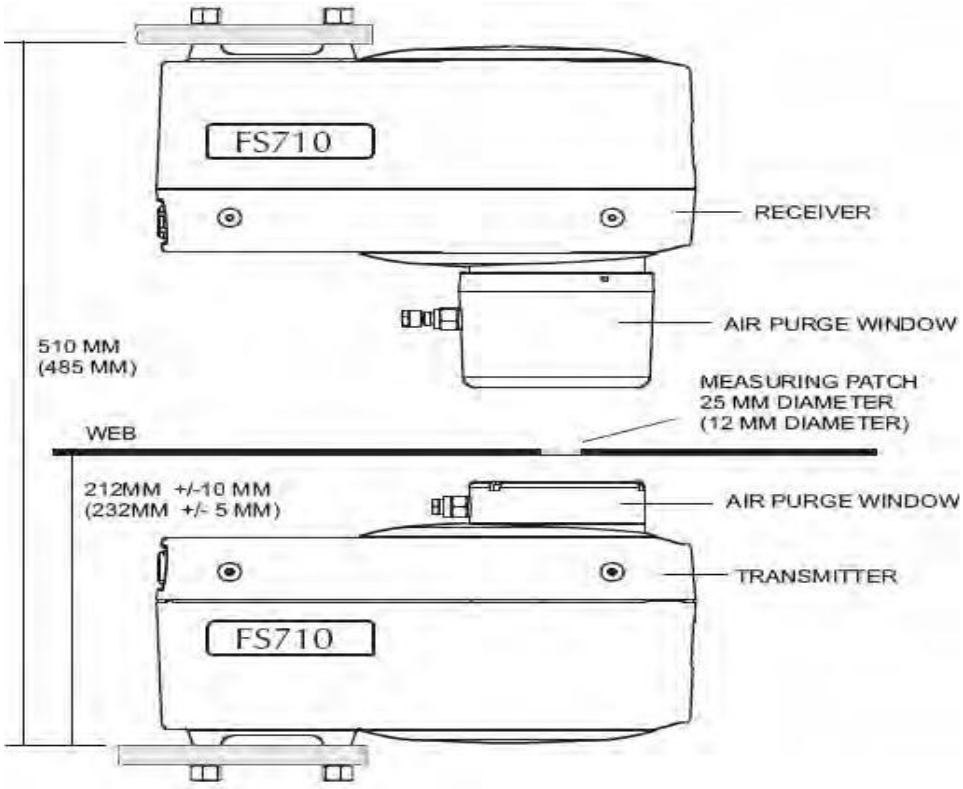


Fig 4.5 – Diagrama esquemático do sensor infravermelho por transmissão. Extraído de [6].

Além da aplicação por reflexão, o fabricante também disponibiliza uma aplicação por transmissão, em que tem-se um emissor e um receptor (vide foto e esquema acima). Os dois modelos podem ser aplicados ao nosso projeto.

4.2.2 – INFRAVERMELHO IST

Este aparelho é semelhante ao NDC, porém apresenta um conjunto mais modesto, além de seu preço estar na faixa de US$ 4.000,00, fora custos de importação e impostos. Possui saídas analógicas e seriais digitais (RS 232) e atende ao nosso projeto.

Fig 4.6 – Fotos do sensor infravermelho e da unidade de controle. Extraído de [7].

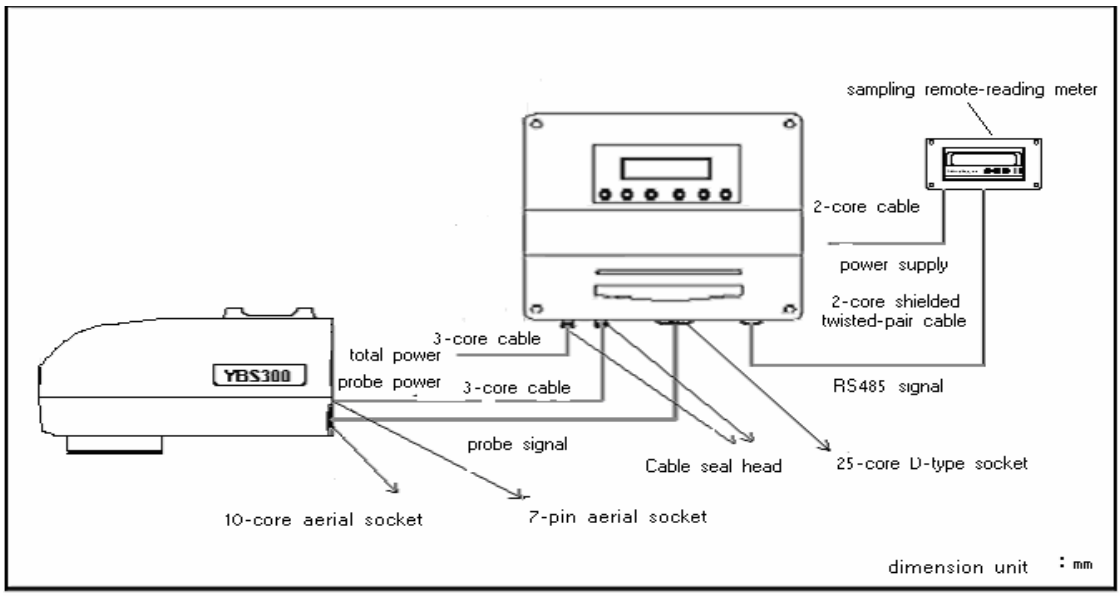


Fig 4.7 – Diagrama esquemático do sensor IST. Extraído de [7].

4.3 – CONTROLE DE UMIDADE POR TEMPERATURA

Nesta opção temos a medição da umidade do papel de forma indireta, através do monitoramento da temperatura do papel logo na saída do forno.

A motivação para esta solução foi o alto custo das soluções anteriores e também pela facilidade e acessibilidade de monitoramento da temperatura, além da facilidade de tratamento do sinal obtido para o controle de nosso inversor de frequência da calandra.

O conceito adotado para esta solução se baseia no fato de que num processo de mudança de fase a temperatura se mantém constante. A fase aquosa da tinta, ou simplesmente a água, ao entrar no forno de secagem, têm sua temperatura aumentada em virtude da ação dos jatos externos de ar até entrar em mudança de fase para vapor. Neste estágio sua temperatura se mantém constante até a total evaporação. Após esta fase, o papel tende a entrar em equilíbrio térmico com os jatos de ar, tendo sua temperatura aumentada. O gráfico abaixo ilustra melhor tal situação:

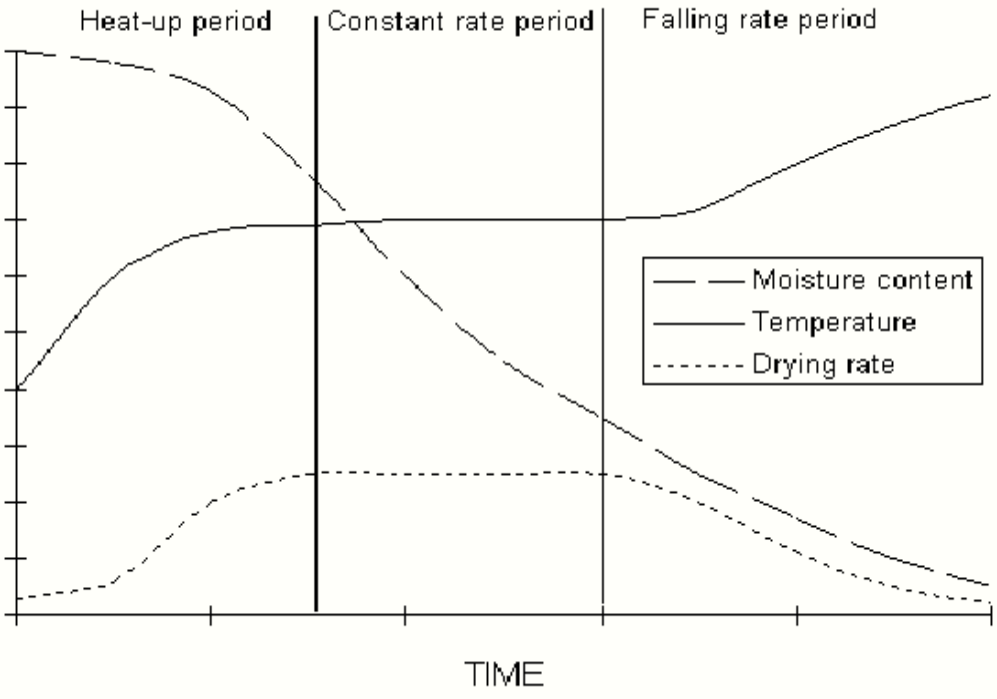


Fig 4.8 – Diagrama de secagem de papel – umidade, temperatura e taxa de secagem. Extraído de [2].

A idéia do controle é instalar o sensor de temperatura à saída do forno, de modo com que o papel, nesta posição, esteja na região 3 odo gráfico, ou na região “Falling rate period”.

Nesta região ocorreu a total evaporação da fase líquida, e o papel inicial sua etapa de aquecimento.

O set-point escolhido é um ligeiro aumento de temperatura, de modo com que não tenhamos um papel muito seco, de modo com que o controlador oscile a velocidade do motor da calandra em função deste valor.

Em outras palavras, se a temperatura de evaporação da água no forno for de 55ºC e o papel, na saída, estiver à 60ºC, temos certeza de que o papel não está úmido, e a velocidade está correta.

Caso o papel esteja à 80ºC, significa que podemos dar um incremento na velocidade da máquina.

Por outro lado, caso o papel esteja à 55º C na saída, ou seja, na temperatura de evaporação da água, não temos informação do percentual de umidade no papel, mas certamente não está seco; ou seja, deve-se diminuir a velocidade da máquina para a efetivação da secagem.

Hoje esta verificação é feita através do “feeling” do operador, e o controle de velocidade, conforme mencionado, é feito de forma manual através do potenciômetro instalado.

Abaixo temos um diagrama real de secagem de papel que traduz os conceitos apresentados:

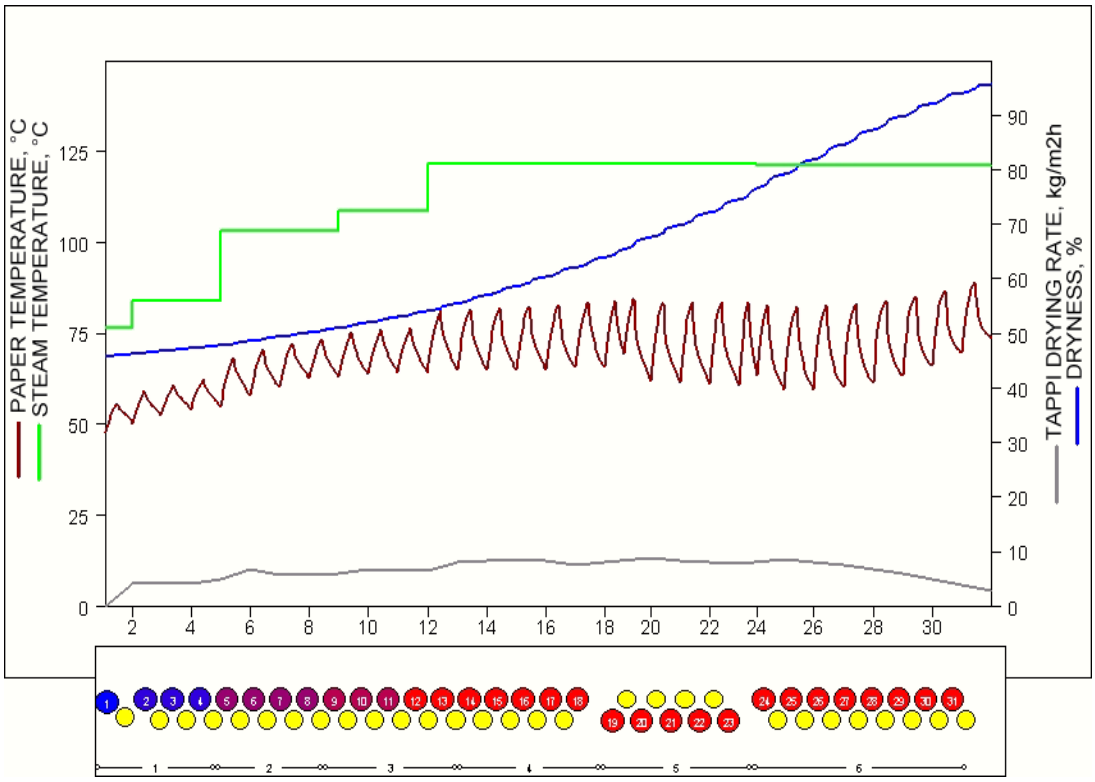


Fig 4.9 – Diagrama real de secagem de papel sob cilindros secadores. Extraído de [2].

Note que ao final da secagem a temperatura do papel é de aproximadamente 80ºC, apresentando uma umidade de menos de 5%. Ao longo do processo de secagem a temperatura manteve-se praticamente constante, na média. Valores típicos de umidade para papel são de 2,57%, que pode ser aferido posteriormente com um medidor de umidade portátil para amostras, que são bem acessíveis.

Em nosso processo consideramos que o sistema de secagem possui potência térmica suficiente para atender à demanda, e a velocidade com que o papel atravessa o forno, e portanto a velocidade de produção da máquina, deve obedecer a um mínimo e estar limitada a um máximo, que pode ser a velocidade angular nominal do motor da calandra.

Devemos portanto inicialmente elaborar um estudo térmico do processo para determinar qual a taxa de evaporação de água desejada, e estimar a potência térmica necessária para tanto. Será verificado se os atuais queimadores podem atender a esta demanda, e se o

número de cilindros de gás instalados atendem à esta taxa de evaporação de GLP requerida. Por fim faremos uma estimativa do custo de gás consumido para cada quilo de papel revestido, realizando um comparativo com outras formas de energia disponíveis e se este custo está dentro do esperado para níveis competitivos de mercado.

5 – CÁLCULO DA POTÊNCIA TÉRMICA

Coaters para papéis especiais operam normalmente com velocidades de 100m/min a até 300 m/min. Há equipamentos que operam em velocidades superiores, porém fogem do escopo de nossa aplicação, tanto por fatores de escala quanto por fatores de capital investido. Para a nossa aplicação, o ideal seria uma produção de 1 bobina jumbo para cada 2 horas de trabalho, ou seja, 300 kg de papel base ou 12.600 metros lineares a cada 120 minutos, que resulta numa velocidade mínima de 100m/min. O papel base utilizado possui gramaturas de 48g/m^2a 50 g/m^2.

De acordo com as relações de polias e a velocidade máxima nominal do motor, a velocidade máxima que podemos atingir com esta máquina é de 240m/min. Por questões de segurança, adotaremos 210m/min como nossa velocidade limite, que resultaria numa taxa de revestimento de aproximadamente uma bobina jumbo por hora, que está ótimo para as instalações oferecidas.

Com o forno de secagem tendo um comprimento de 20 metros, temos, para 100m/min, um tempo de 12 segundos para um ponto do papel percorrer o forno. Já para a velocidade de 210m/min, temos um tempo de 5,7 segundos.

Formulações típicas possuem 40% de teor de sólidos em sua composição. Num revestimento razoável aplica-se 8 g/m^2 de sólidos. Realizando uma proporção, temos um total de 20g/m^2 de tinta aplicada, sendo portanto 12 g/m^2 a água a ser evaporada, ou uma taxa de 10g de H_2O/s na primeira velocidade e 21,05 g de H_2O/s na maior velocidade.

Em outras palavras, para cada passagem do papel pelo forno, devo retirar 120 g de água num comprimento de 20 metros por 0,5 metro de largura do papel, num tempo de 12

segundos para a primeira velocidade, e 5,7 segundos na segunda velocidade.

De acordo com [1] e [2], a melhor configuração de secagem para revestimento de papéis é através de jatos colidentes, que incidem sobre a superfície do papel de forma perpendicular.

Ainda segundo a literatura, a forma predominante de transferência de calor é a convecção forçada em regime turbulento, podendo-se desprezar a irradiação térmica, pois seus valores são pequenos comparados à convecção.

De fácil implementação, o arranjo pode ser facilmente implementado na máquina, e o modelo que iremos adotar para a realização de nossos cálculos está esquematizado abaixo, alimentado pelo jato de ar quente proveniente do queimador.

Com base na taxa de evaporação determinada acima, vamos estimar quantos g/s de H_2O cada jato consegue evaporar. Consideramos que o primeiro jato, além da transferência de calor e de massa, irá elevar a temperatura de filme de água da temperatura ambiente, considerada a 25º C, para a temperatura de 55º C, mantendo-se praticamente constante até o final do processo. Esta hipótese está de acordo com o diagrama real de secagem do papel representado pela fig 21. Iremos portanto verificar a quantidade de massa evaporada por um segundo arranjo, este avaliado na temperatura de 55º C, tendo-se a seguir a quantidade de arranjos necessária para a total evaporação de massa de água para a velocidade de máquina desejada.

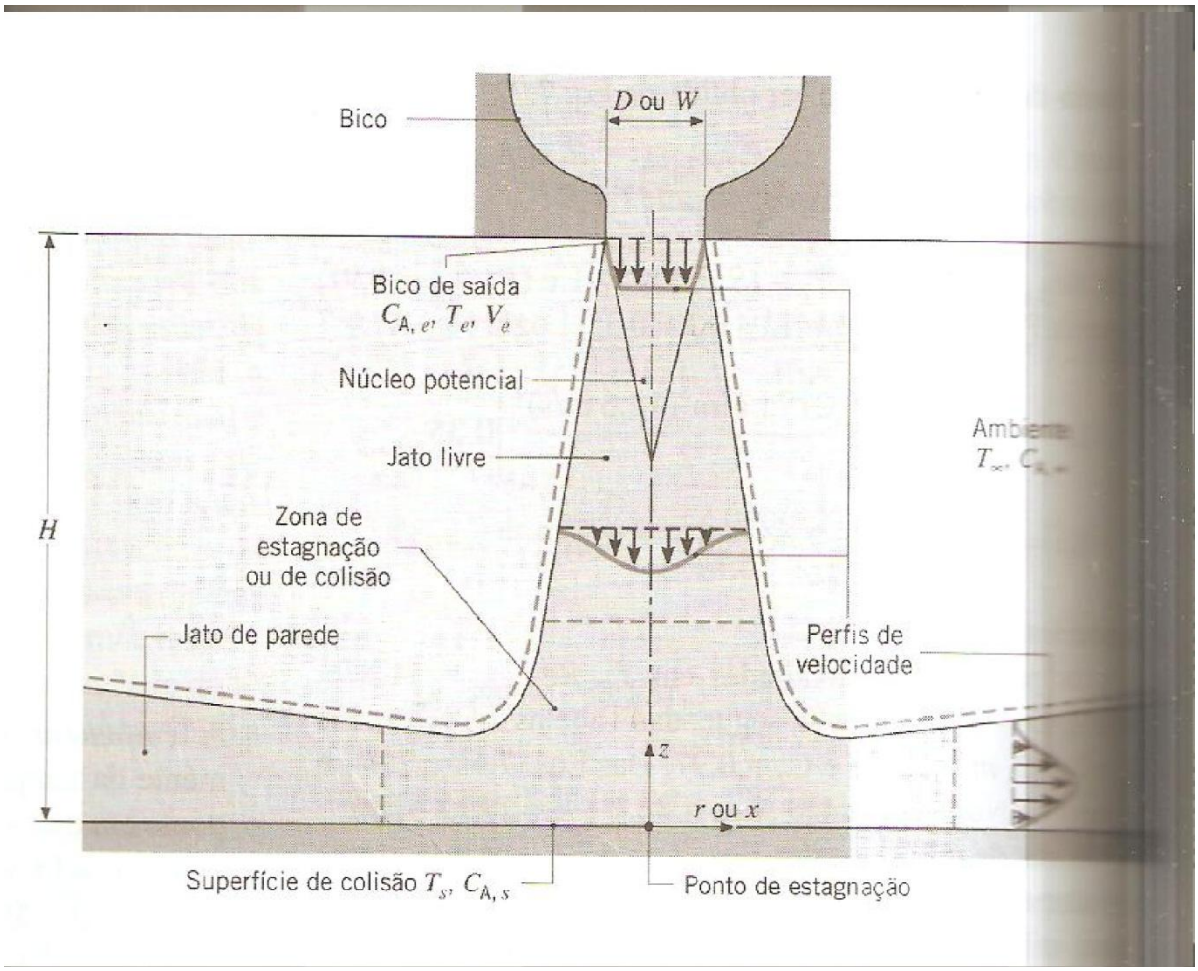


Fig 5.1 – Diagrama esquemático do arranjo do bico secador. Extraído de [1].

Um parâmetro geométrico pertinente é relativo à área do bico, que é definida como a razão entre a área da seção transversal da saída do bico e a área da superfície da célula. Para a nossa aplicação:

$A_r = D^2/4r^2$ ou $A_r = A_{c,e}/A_{cel}$

Este parâmetro geométrico deve ficar no intervalo 0,004$\leq A_r \leq$ 0,04.

Para o diâmetro do jato D=70mm, $A_{c,e} = 3{,}8 * 10^3\ mm^2$.

Para efeito de nossos cálculos, iremos adotar a célula como um círculo de diâmetro 0,5 metro, o que é bem razoável, pois o papel possui largura de 0,5 metro.

O valor de $A_{cel} = 196{,}3 * 10^3\ mm^2$. Com isto $A_r = 0{,}02$.

No sentido longitudinal, cada jato tem influência sobre 0,4 metros, aproximadamente.

De acordo com [4], temos uma relação otimizada entre D e H:

$$D = 0{,}2 * H$$

Logo a altura do jato deve ser H= 350mm.

Medindo-se a temperatura de saída do queimador, anotou-se o valor de 200ºC, que será adotado para os nossos cálculos, a uma velocidade de 50m/s, de acordo com as especificações do fabricante dos ventiladores. Esta também será a velocidade adotada na saída dos jatos.

Das tabelas de propriedades do ar a 277º C (550 K), temos:

$\nu = 45{,}57 * 10^{-6}\ m^2/s$
$k = 43{,}9 * 10^{-3}\ W/m * K$
$P_r = 0{,}683$

Das tabelas de vapor saturado a 27º C (300K):

$\rho_{A,sat} = 0{,}0255\ Kg/m^3$

Idem para 55º C (328 K):

$\rho_{A,sat} = 0{,}1045\ Kg/m^3$

Da tabela de Coeficientes de Difusão Binária em 1 atmosfera:

$D_{AB} = 0{,}26 * 10^{-4}\ m^2/s.$

Como a máquina está instalada no município de São Paulo, podemos estimar, com base em informações de empresas de previsão de tempo, que a umidade relativa média do ar é de 70%, que remos adotar para a elaboração de nossos cálculos.

Temos basicamente duas equações para quantificar a troca de calor, que será utilizada para o cálculo de potência térmica e consumo de gás, e a equação de transferência de massa, que irá quantificar quanto cada bico evapora de água.

$$(I)\ \ q'' = h * (T_s - T_e) \quad Lei\ de\ Newton\ do\ resfriamento.$$

E através de sua transferência de massa análoga:

$$(II) \quad N_A'' = h_m * \left(C_{A,s} - C_{A,e}\right)$$

Iremos inicialmente utilizar a equação de transferência de massa (II) para, depois de determinado o número de arranjos de bicos necessário, quantificar o calor trocado através de (I).

Utilizando a analogia entre as transferências de calor e massa na convecção:

$$\frac{\overline{Nu}}{Pr^{0,42}} = \frac{\overline{Sh}}{Sc^{0,42}} \quad (V)$$

E adotando as seguintes hipóteses:

-condições não influenciadas pelo nível de turbulência na saída do bico e a superfície;
-regime permanente;
-comportamento de gás ideal para o vapor d´água;

Podemos utilizar, para um único bico redondo, a correlação recomendada por [4].

$$\frac{\overline{Nu}}{P_r^{0,42}} = G\left(\frac{r}{D}, \frac{H}{D}\right) * F_1(R_e) \quad (VI)$$

$$G = 2 * A_r^{\frac{1}{2}} * \frac{1 - 2{,}2 * A_r^{\frac{1}{2}}}{1 + 0{,}2 * \left(\frac{H}{D} - 6\right) * A_r^{\frac{1}{2}}}$$

$$F_1 = 2 * Re^{\frac{1}{2}}(1 + 0{,}005 * Re^{0{,}55})^{\frac{1}{2}}$$

$$Re = \frac{u_\infty * L}{\nu}$$

$$S_c = \frac{\nu}{D_{AB}}$$

Validade da equação:

$$\begin{bmatrix} 2.000 \leq Re \leq 400.000 \\ 2 \leq \frac{H}{D} \leq 12 \\ 0{,}004 \leq A_r \leq 0{,}04 \end{bmatrix}$$

Considerando um comprimento característico L = 1,0m.

$Re = 1{,}09 * 10^6$

$G = 0{,}20$

$F_1 = 7.068{,}8$

$S_c = 1{,}24$

Obtemos portanto o valor de $\overline{Sh}$ através de (V) e (VI):

$$\overline{Sh} = 1.547{,}43$$

Obtemos o coeficiente $\overline{h_m}$ através das relações:

$$\overline{Sh} = \frac{\overline{h_m} * L}{D_{AB}} \ (III)$$

e

$$\overline{Nu} = \frac{\bar{h} * L}{k} \ (IV)$$

Determinando-se o valor de $\overline{h_m}$ conseguimos calcular o taxa de transferência de calor (I) e taxa de transferência de massa (II).

$\overline{h_m} = 0{,}040 \ m/s$ para a transferência de massa.

Para a transferência de calor, $\overline{Nu} = 1.204{,}57$

Que resulta em $\bar{h} = 52{,}88 \frac{W}{m^2} * K$, para a transferência de calor.

Iremos agora determinar a taxa de evaporação de água para o primeiro bico:

$$(II) \quad N_A^{''} = h_m * \left(C_{A,s} - C_{A,e}\right)$$

Iremos agora determinar as concentrações de água no ar, para as condições do problema, para realizar o cálculo de diferença de concentrações, adotando as seguintes hipóteses:

- a fase líquida da tinta não possui gases dissolvidos,
- a fase gasosa (ar + vapor d`água) pode ser tratada como uma mistura de gases perfeitos.

Inicialmente iremos determinar, com o auxílio de uma carta psicrométrica (anexo), a umidade absoluta de vapor d´água na mistura a 300 K, utilizando a curva de 70%:

Da carta obtemos o valor ω= 0,014 kg de água / kg de ar seco.

Esta quantidade de vapor permanece quando a mistura é aquecida bruscamente a 227º C (500 K). Porém o ar, de acordo com as tabelas, apresenta, para esta nova temperatura, $\rho = 0{,}6964\ kg/m^3$.

Temos portanto:

$$\rho_{mistura} = 0{,}014\,\frac{kg\ vapor}{kg\ ar\ seco} * 0{,}6964\,\frac{kg\ ar\ seco}{m^3} = 0{,}009749\ \frac{kg\ vapor}{m^3}$$

Das tabelas de vapor d´água, a 27º C (300 K) a água possui $\rho_{A,sat} = 0{,}0255\ Kg/m^3$.

Justamente esta diferença de concentrações é que provoca o fenômeno de transferência de massa:

Temos $N_A^{''} = 0{,}040 * (0{,}025500 - 0{,}009749) = 0{,}63\ g/s.m^2$
Considerando a área de atuação de cada arranjo com o sendo 0,5 m^2 (o papel possui largura de 0,5 metro).

$$N_A = 0{,}63 * 0{,}5 = 0{,}315\ g/s$$

Esta é a taxa de evaporação para o primeiro arranjo.

Iremos agora considerar a transferência de massa a partir do segundo arranjo, sob influência do primeiro; ou seja, a temperatura do filme d´água está agora a 55º C, utilizando o modelo da fig 21 de secagem de papel, e permanecendo constante nesta temperatura até o final da evaporação.

Nesta condição,

$$\rho_{A,sat} = 0{,}1045\ Kg/m^3$$

Logo

$$N_A^{''} = 0{,}040 * 0{,}5 * (0{,}104515 - 0{,}009749) = 1{,}9\ g/s$$

Como consideramos que as condições no interior do forno permaneçam constantes ao longo da evaporação, podemos concluir que, para a velocidade de 100 m/s, ou uma taxa de evaporação de 10 g /s, é necessário um arranjo inicial e cinco adicionais.

Já para a velocidade de 210 m/min, ou 21,05 g/s, é necessário um arranjo inicial e dez adicionais.

De posse destas quantidades, iremos calcular a potência térmica necessária para tal:

Como

$$\bar{h} = 52{,}88\ W/m^2.K$$

Substituindo em (I), temos:

$$q^{''} = 52{,}88 * (227 - 25)$$

$$q^{''} = 10.681{,}7\ W/m^2$$

ou, por hora:

$$q^{''} = 38.454\ kJ/h$$

Como a unidade mais utilizada para os fabricantes de queimadores é a caloria, temos:

1 cal = 4,184 J

ou

$q^{''} = 9.190{,}8\ Kcal/h$ para cada arranjo.

Utilizando 6 arranjos, temos

$$q^{''} = 55.144{,}8\frac{Kcal}{h}$$

Para 11 arranjos:

$$q^{''} = 101.098{,}8\frac{Kcal}{h}$$

A capacidade instalada dos queimadores é de 300.000 kcal/h, sendo o principal de 220.000 kcal/h e o secundário de 80.000 Kcal/h.

Sabendo que a capacidade calorífica do GLP é de 11.200 Kcal/kg, podemos estimar o consumo de gás para cada hora de trabalho:

$$consumo = \frac{55.144{,}8\ Kcal/h}{11.200\ Kcal} = 4{,}92\ Kg\ GLP/h$$

Para a configuração de 11 arranjos, temos:

$$consumo = \frac{101.098{,}8\ Kcal/h}{11.200\ Kcal} = 9{,}02\ Kg\ GLP/h$$

A um custo médio de R$ 2,85 / Kg, temos R$ 14,02 / h no primeiro caso, e R$ 25,70 / h no segundo caso.

No entanto, temos um custo de gás por kilo de papel revestido de R$ 0,093 / kg no caso 1 e R$ 0,085 / kg no caso 2.

O que nos leva a concluir que é mais vantajoso operar com 11 arranjos, pois além de aumentar a produção temos um menor custo de combustível por kilo de papel revestido.

Um outro dado que podemos determinar é a quantidade de cilindros que deve ser instalada para a potência requerida.

Um cilindro P-190 possui taxa de evaporação de 3,5 Kg/h de GLP a até 80% de seu volume.

Para volumes inferiores a num mínimo de 50%, que é o recomendado para um novo abastecimento, a taxa de evaporação é de 3,0 kg/h de GLP.

Como temos 9,02 Kg de GLP/ h para 11 arranjos, necessita-se portanto de 3 cilindros P-190 para o atendimento da potência requerida pelos queimadores.

5.1 SIMPLIFICAÇÕES E HIPÓTESES ADICIONAIS DO MODELO TÉRMICO

Algumas simplificações foram utilizadas na implantação do modelo, sendo a principal referente à área de atuação de cada bico, admitida como um círculo de raio 0,5m, e na não interferência de um arranjo para o seguinte. Outro fator foi o número de Reynolds calculado estar acima do intervalo de aplicação da equação proposta no modelo do jato. Porém, as outras condições foram satisfeitas.

Admitiu-se também uma umidade relativa do ar a 70%. No entanto, de acordo com empresas de previsão do tempo, esse valor varia muito durante o dia, e também com as estações do ano. Dados podem ser adquiridos nas páginas na Internet destas empresas.

Esta variação da umidade relativa altera muito a concentração de vapor no ar, tendo como consequência a variação na massa específica do ar quando aquecido pelo queimador, e finalmente na diferença de concentrações da água na equação de transferência de massa, alterando-se seu resultado de forma significante.

No entanto podemos afirmar que nosso modelo de controle não sofre com essa variação de umidade relativa do ar durante o turno de operação, justamente por medir o incremento de temperatura ao final do processo, indicativo da secagem do papel. O que pode ocorrer, neste caso, é uma redução da produtividade da máquina, que pode ser compensada por um acréscimo de potência térmica pelos queimadores, já que os mesmos, para a condição calculada neste trabalho, operam abaixo de suas capacidades nominais.

6 - CONTROLE

A seguir iremos expor algumas alternativas para o controle de nosso processo, a partir da medição da temperatura do papel por infravermelho ao final do túnel de secagem, e, a partir de uma matriz de decisão, implementar e detalhar a lógica de controle, o hardware escolhido e a apresentação do software elaborado. Para o controle listamos três opções, apresentadas a seguir:

A primeira proposta é a opção do controle por microprocessador PIC do fabricante MICROCHIP, em que há no mercado opções de placas montadas, com entradas e saídas analógicas e digitais, e o software utilizado para o controle pode ser implementado em linguagem C, e a gravação deste software é feita em um gravador de PIC, que geralmente acompanha a placa. O código é gerado em um PC e a comunicação é feita através de um cabo USB. Pode ser regravado e o controle é feito direto na máquina, coletando o valor analógico do termopar e, através de uma rotina, parametrizar a velocidade do inversor.

O monitoramento pode ser feito na tela do PC através de softwares desenvolvidos para tanto, em linguagens C++ ou DELHI 6.0.

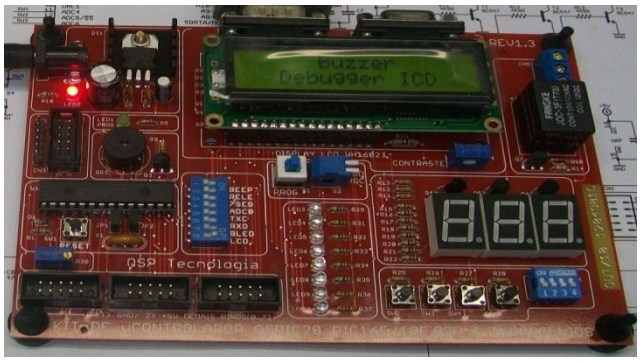


Fig 6.1 – Exemplo de placa PIC. Fonte: www.qsptec.com.

De baixo custo, ao redor de R$ 280,00, apesar de atender ao projeto, tem o problema da instabilidade e do fato de ser indicado para aprendizagem de controle de sistemas, e não necessariamente em uma aplicação industrial. È necessária a montagem de uma caixa protetora, para evitar danos à placa, pois os componentes eletrônicos ficam expostos.

Uma outra opção é através do controle por CLP, uma das mais utilizadas na indústria, e consiste basicamente na implementação de um software baseado em linguagem *Ladder* ou de relês, cujo software, livre, é de fácil programação, inclusive possuindo uma interface gráfica para sua elaboração.

O software é programado em ambiente Windows e pode ser alterado a qualquer momento. Basta conectar o CLP novamente e enviar um novo código, caso seja necessário.

Através da entrada para termopar, o CLP coleta os dados de temperatura e, com base nesse valor envia um sinal através de sua saída analógica, que por sua vez alimenta a entrada analógica do inversor de frequência, por sinal a mesma que hoje é utilizada pelo potenciômetro.

Tem a opção de uma IHM para a visualização dos parâmetros e para o início/ fim de processo. É projetado para suportar ambientes fabris e há várias opções no mercado, como WEG, Siemens, Rockwell etc. A faixa de preço é ao redor de R$ 1.800,00 e há a opção de uma IHM de 7 pol de tela, acréscimo de R$ 800,00.

Fig 6.2 – Exemplo de CLP – modelo PLC 300. Fabricante: WEG. Fonte: www.weg.com.br

Finalmente uma outra opção de controle do processo é através de redes de dados, assunto explorado durante a graduação do curso de mecatrônica, e também aplicado num projeto bastante interessante durante o curso, o projeto integrado Pi7.

Dentre os protocolos físicos existentes podemos citar o RS 232 (ponto a ponto), o RS 485, que utilizam como protocolos lógicos o MODBUS-RTU ou MODBUS-ASCII. No caso do inversor de frequência WEG o modo ASCII não é utilizado, e há ainda a opção de utilização de um protocolo próprio, que é o WEGBUS.

Há ainda a opção do protocolo físico FIELDBUS, tendo-se como opções de protocolos lógicos o Profibus DP, DeviceNet ou Ethernet IP, entre outros.

No presente trabalho iremos optar pela rede 485 devido à facilidade e familiaridade de trabalho e também pelo fator custo, pois para a transmissão dos sinais de controle, tanto dos sensores quanto dos atuadores (no caso o inversor de frequência), é necessária a instalação de uma interface ou placa de comunicação destes elementos com a rede, e a que possui menor custo para aquisição é a do 232. Apenas para ilustrar a diferença no custo, este modelo é cotado a R$ 220,00, enquanto que o FIELDBUS DeviceNet a R$ 2.350,00,

portanto claramente justificando nossa escolha.

Vale citar que a diferença entre a RS232 e RS 485 é que aquela é ponto a ponto. Já a causa a 485 nos dá a possibilidade de controlar até 255 pontos. A comunicação entre os dois modos é feita através da instalação de um chip conversor 232/485, ao custo de R$ 39,00.

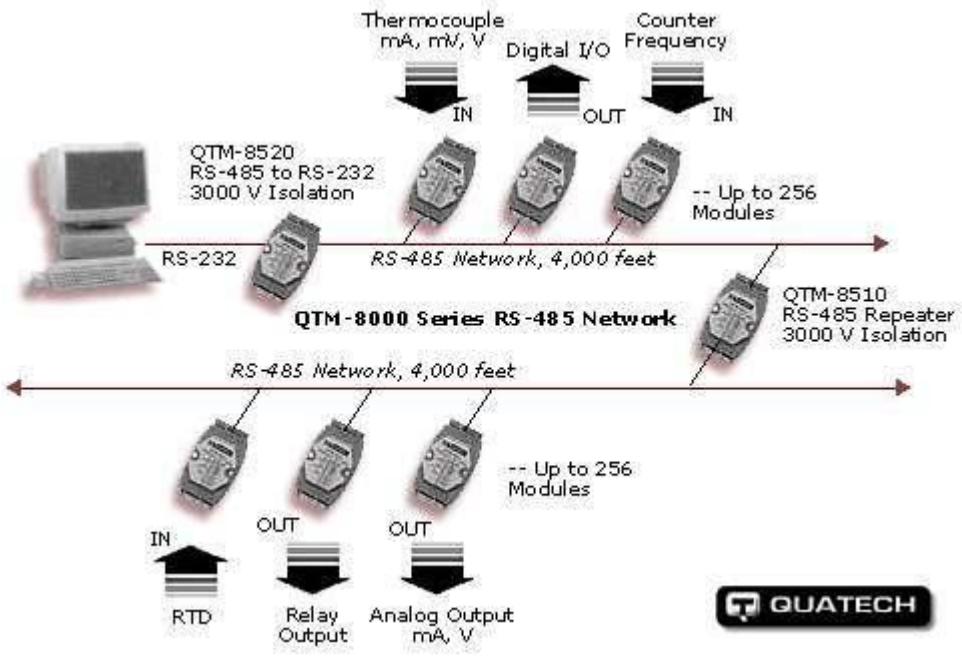


Fig 6.3 – Exemplo de rede 485. Fonte: www.quatech.com

A idéia do controle é comunicar o inversor de frequência à rede através de um endereço, e coletar o sinal do termopar e integrá-lo à rede. Isto é feito por um I/O remoto, que possui justamente essa função: converter o sinal do termopar em um sinal da rede.

Esta I/O remota foi orçada em R$ 620,00 do fabricante Advantech, modelo ADAM-4118, específica para termopar.

Fig 6.4 – Exemplo de I/O remoto. Fabricante: Advantech. Fonte: www.advantech.com

Através da comunicação dos componentes citados com a rede, o controle é feito através de um PC comum, podendo ser utilizado até um modelo 486, em que um software de comunicação e controle é implementado em JAVA, mais precisamente utilizando o software NETBEANS, gratuito que, através da construção de uma Interface Homem Máquina (IHM), pode-se realizar os comandos necessários e monitoramento das variáveis do processo em tempo real, substituindo por completo o quadro de comandos analógico.

Na implementação da IHM está o acionamento/parada do processo, o set de temperatura, a leitura de temperatura do processo, a velocidade do motor, a opção de imputar manualmente uma referência de velocidade e a opção de modo automático, em que se inicia o controle propriamente dito do processo, inabilitando a opção de referência de velocidade manual.
Um modelo de nossa IHM apresenta-se abaixo:

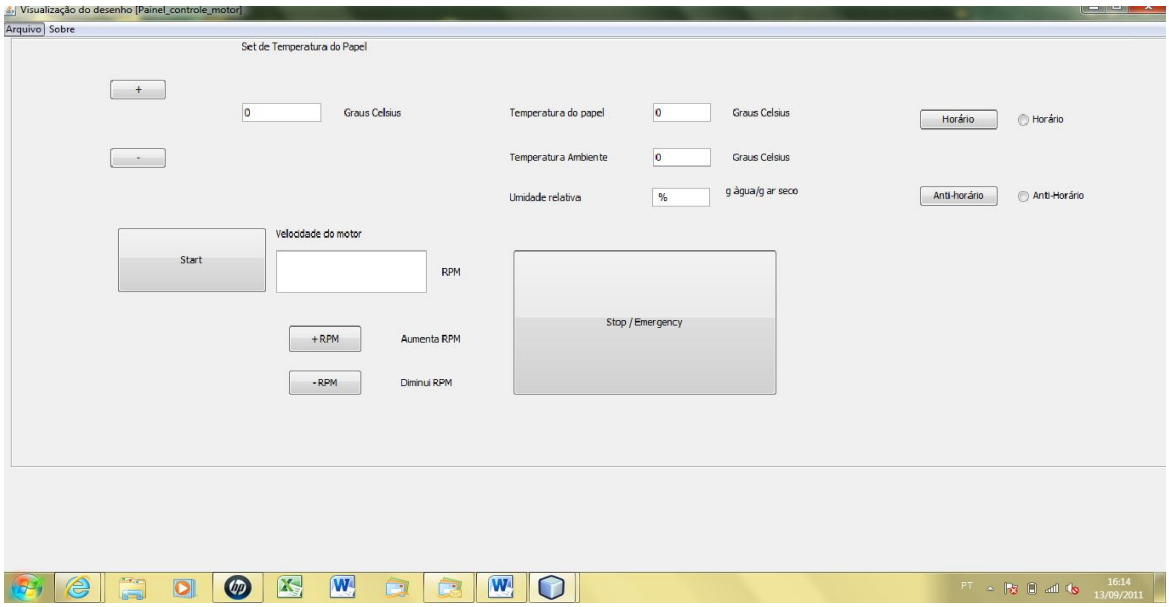


Fig 6.5 – Exemplo de IHM para controle, vista na tela do computador.

Como a rede nos dá a possibilidade de comandar até 255 pontos, esse sistema oferece a vantagem de ampliação de nosso sistema de controle para outros elementos da máquina, bastando a instalação de I/O's remotas e a continuidade de programação de nosso software. Como o processo ocorre a eventos discretos, fica a sugestão para trabalhos futuros da construção de redes de Petri e de SFC`s para a implementação do controle de todo o processo.

Em resumo oferece flexibilização no sistema, é facilmente implementado, devido à programação orientada a objetos, e conseguimos visualizar em tempo real o processo na máquina. Podemos implementar inclusive outras rotinas, para por exemplo revestir outros tipos de papéis que exijam outras configurações da máquina.

Uma desvantagem desta opção é a estabilidade, talvez relativa do sistema, que ficadependente do ambiente Windows. Comparando-se com o CLP possui menor estabilidade, porém futuramente poderemos implementar o código em um sistema operacional Linux Real Time, por exemplo.

Para a escolha de nosso sistema de controle utilizaremos uma matriz de custos como parâmetro de escolha para cada sistema de controle apresentado. Como o sensor utilizado será o mesmo para cada aplicação, o mesmo não entra em nossa matriz.

Além do custo, os outros parâmetros considerados para nossa escolha foram:(a) a flexibilização e facilidade na programação,(b) software gratuito,(c) implementação de IHM a baixo custo, (d) a possibilidade de expansão do controle para outros componentes da máquina, (e) estabilidade do sistema de controle, (f) familiaridade com o sistema.

A matriz com os requisitos encontra-se a seguir:

	PIC	CLP	485	OPÇÃO
CUSTO	R$ 280,00	R$ 1.800,00	R$ 720,00	PIC
PROGRAMAÇÃO	C/C++	Ladder	JAVA	485
SOFT GRATUITO	Sim	Sim	Sim	Qualquer
IHM	Não	Não	Sim	485
EXPANSÃO	Pequena	Média	Alta	485
ESTABILIDADE	Baixa	Alta	Média	CLP
FAMILIARIDADE	Não	Não	Sim	485
RESULTADO				**485**

Fig 6.6 – Matriz de decisão de controle.

De posse dos resultados apresentados, a solução por rede 485 foi eleita como a mais indicada para a nossa aplicação, apesar de que todas as soluções apresentadas podem ser implementadas para o cumprimento do objetivo de controle do processo.

Para a implementação de nosso controle é fundamental a correta escolha do sensor, na medida em que o mesmo não pode estar em contato com o papel.

Nesse caso, a utilização de um termopar já é descartada, e opções por infravermelho serão analisadas.

Há alguns modelos no mercado que realizam a leitura por laser e possuem uma saída RS 232, porém de custo elevado. São similares a termômetros infravermelho por gatilho, porém o laser fica acionado durante todo o processo de secagem do papel, e a informação é enviada à rede. Na verdade seria feita uma adaptação de um aparelho portátil ao nosso processo, que não foi recomendado pelo fabricante. O custo apresentado foi de R$ 1.800,00.

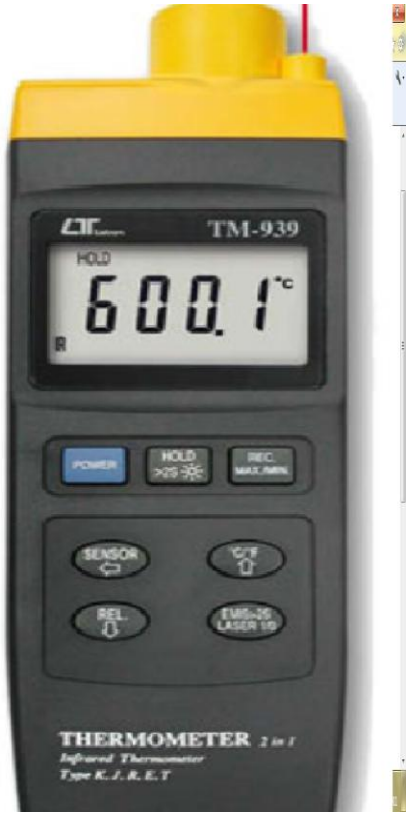


Fig 6.7 – Modelo de Termômetro IV Laser com saída RS 232. Fonte: www.impac.com.br

Após muita pesquisa foi encontada uma boa opção, que foi o pirômetro infravermelho, semelhante a um termopar, porém realizando a leitura da temperatura por infravermelho. Não possui o laser, e o sinal é convertido em uma voltagem. Não há portanto contato entre

o sensor e o papel, e sua comunicação com a rede é feita através da I/O remota, já discutida. O custo do sensor foi orçado em R$ 525,00, e exemplos são apresentados abaixo.

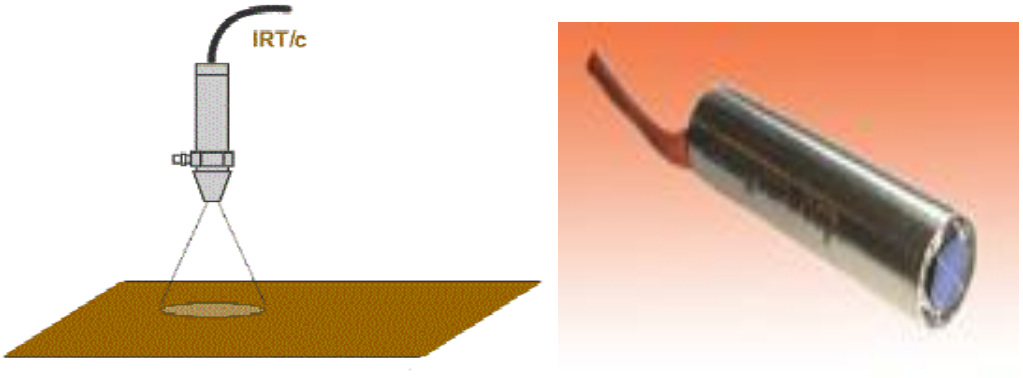


Fig 6.8 – Modelo de Pirômetro Infravermelho. Fonte: www.ztechsensores.com.br

A seguir será feita a implementação de nosso sistema de controle.

Inicialmente será feita uma modelagem de nosso sistema e a identificação dos parâmetros pertinentes ao controle e as características do sistema. Esses valores serão determinados de forma experimental durante o processo de operação da máquina.

Através destes dados iremos compor uma lógica de controle, implementada em nosso software.

Devido ao tamanho da máquina será construído um protótipo em bancada para simular a operação de secagem e para verificação da eficiência do controle. Busca-se também a correção de eventuais erros de modelagem e da lógica de controle adotada.

Para a modelagem de nosso sistema iremos considerar a diferença de temperatura lida pelo sensor e o valor de set-point, determinado pelo operador da máquina em nossa IHM.

A partir do valor desta diferença, é enviado um sinal para o inversor de frequência, que altera a referência de velocidade, e consequente alteração da temperatura do papel. Podemos afirmar que o sistema é *retroalimentado*, ou de *malha fechada:*

Iremos a seguir apresentar alguns resultados que foram verificados durante o processo de operação da máquina, e que servirão de base para a implementação de nosso algoritmo de controle.

Com a máquina operando em regime, verificou-se que a temperatura verificada no sensor tal que a qualidade do papel estava ideal era de 60º C. Por tratarmos de um papel termossensível, verificou-se também que à temperatura de 80º C iniciou-se o processo de revelação do mesmo, ou seja, como o corante utilizado para a imagem é da cor preta, a folha de papel ficou totalmente escurecida nesta temperatura, assemelhando-se a um filme fotográfico.

O valor de 60º C portanto será a nossa referência e nosso algoritmo de controle deve ser tal que durante o processo de revestimento o papel esteja na maior parte do tempo neste valor, independentemente das condições de temperatura e umidade relativa do ambiente e da temperatura da estufa.

Observou-se também que durante o início do processo, ou até o papel atingir esta temperatura, a máquina opera em velocidade constante de 300 rpm. Ou seja, não seria plausível a introdução de nosso controle até que seja atingido o primeiro valor de leitura da referência. Caso isso ocorra, o nosso controle tenderia a diminuir a velocidade da máquina, o que seria prejudicial ao nosso processo, dado que velocidades abaixo de 300 rpm prejudicam a continuidade do processo, pois verificou-se a instabilidade do rebobinador que, por operar por torque, gerou grandes oscilações devido à baixa velocidade, podendo provocar o rompimento da folha.

Sendo atingida a referência de temperatura, e devido à potência dos queimadores, a leitura do sensor de temperatura começa a aumentar, indicando uma secagem excessiva do papel. Como já colocado devemos aumentar a referência de velocidade para evitarmos a faixa de 80º C.

Considerando que a temperatura do forno tenha entrado em regime, ou seja, que a potência térmica entregue pelos queimadores não mais contribuirá para um aumento de temperatura do forno, foi construída uma tabela de valores que relaciona a diferença de temperatura do sensor com um acréscimo na referência de velocidade.

Através de um método iterativo e realizando interpolação linear, verificou-se que para cada 1º C de diferença de temperatura, aplicando-se um acréscimo de 50 rpm na referência de velocidade, o sistema restabelecia o set de temperatura após um período de 5 segundos. Adotando a hipótese de linearidade, adotaremos essa escala para os cálculos de variação de velocidade em função do erro de temperatura.

Adotando ainda a hipótese de nossa planta de comportar como um sistema de primeira ordem, podemos determinar sua constante de tempo. Considerando o critério de 2% para o tempo de estabilização do sistema, no qual temos um total de 4 constantes de tempo para 98% da resposta, determinamos τ=1,25 seg.

Portanto o período de amostragem de temperatura deve ser tal de forma que o sistema tenha alcançado a estabilidade da ação anterior. Como verificou-se que o sistema é corrigido após um período de 5 segundos, iremos adotar um tempo de amostragem de 7 segundos. Em outras palavras, o sistema realiza o processo de leitura do sensor a cada 7 segundos, e para cada leitura espera-se um restabelecimento da temperatura em 5 segundos.
É importante ressaltar que as hipóteses de linearidade do sistema e tempo de restabelecimento estão sendo admitidas para toda a faixa de operação de temperaturas do sistema.

Como não foi criado um modelo matemático do sistema e essas são observações empíricas, as verificações e posteriores ajustes podem ser realizados através do protótipo que será construído em bancada justamente para esta finalidade.

Pelo fato do sistema de controle ter sido implementado em software, as alterações necessárias podem ser facilmente implementadas para eventuais correções dos erros de modelagem.

De posse das características da planta iremos descrever o sistema de controle implementado e, para a realização de testes, será construído um protótipo que simula as condições de temperatura do papel e alteração da referência de velocidade de um motor.

A lógica de controle basicamente realiza a leitura do termômetro a cada 7 segundos e compara esse valor lido com a referência, escolhida pelo operador da máquina. Em função deste valor o sistema envia uma nova referência para o inversor.

Abaixo temos um fluxograma que ilustra nossa lógica:

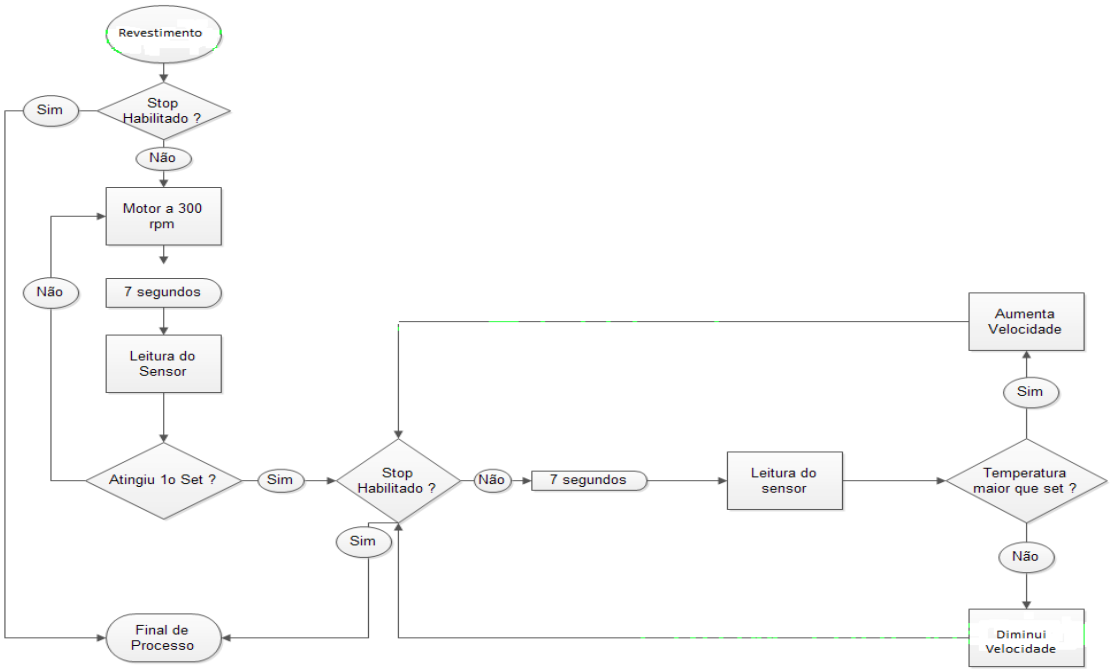


Fig 6.9 – Fluxograma de controle

A alteração da referência de velocidade é feita através de um incremento ou decremento de 50 rpm em relação à velocidade anterior.

Não foi implementada uma lógica proporcional (P) ou Proporcional-Integral-Derivativa (PID) pois, para diferenças de temperaturas elevadas o esforço de controle seria alto, resultando numa brusca alteração na referência de velocidade, que iria causar o rompimento da folha de papel, mesmo com um aumento dos tempos de aceleração e desaceleração do inversor de frequência.

Abaixo um diagrama de blocos de nosso sistema:

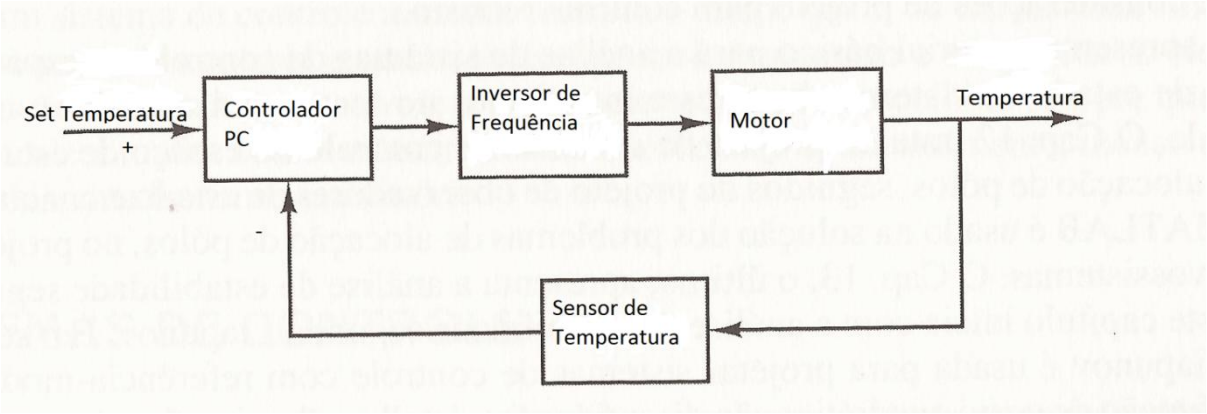


Fig 6.10 – Diagrama de blocos do sistema

Na fase inicial do processo, o operador pode controlar a velocidade do motor de forma manual, no sentido de se regular a passagem do papel na máquina e início do rebobinamento, enquanto o papel ainda não recebeu o revestimento, e o forno em início de aquecimento. Ao habilitar o processo, a velocidade mínima registrada é de 50 rpm, e o operador pode escolher o sentido de giro do motor.

As leituras estão sendo realizadas, e a temperatura visualizada em nossa IHM, porém o controle não é realizado. Esta é a opção "Manual".

A opção "Automático" implementa a ação de controle descrita, desabilita as opções de alteração de velocidade da IHM, e o controle é realizado para velocidades acima de 300 rpm, que, como visto, é a velocidade mínima para a continuidade.
Em qualquer momento o operador pode pressionar a tecla "Emergência", caso tenha a necessidade de parar completamente o processo por algum problema.

Abaixo temos uma representação de como o computador se comunica com a planta:

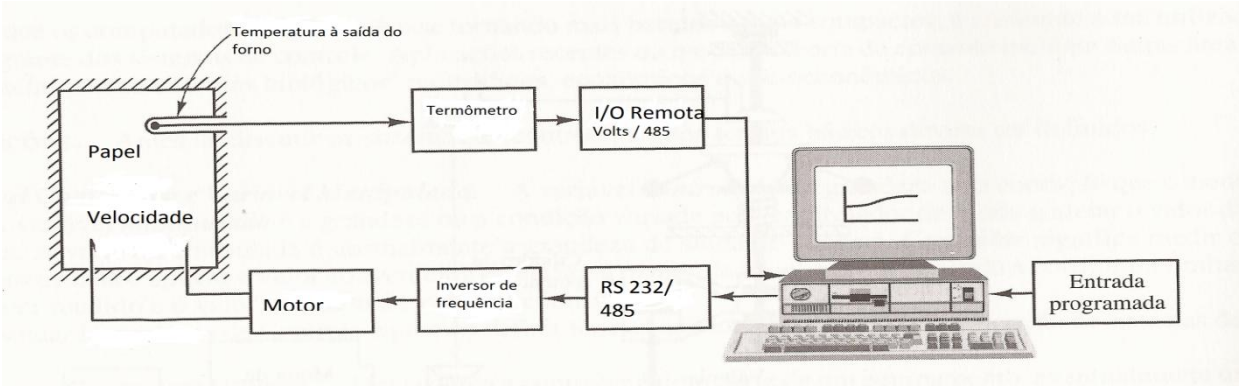


Fig 6.11– Diagrama esquemático do controle

Abaixo iremos detalhar cada componente de nosso sistema, utilizado tanto no protótipo quanto diretamente na aplicação.

O hardware utilizado será um Notebook HP G42 (Dual Core, 3GB RAM, 320 GB HDD,).

Entretanto vale lembrar que qualquer PC, mesmo obsoleto, cumpre com o objetivo do controle, não sendo necessária a utilização de uma configuração avançada.

Os softwares utilizados foram o Microsoft Windows 7 64 bits (www.microsoft.com.br) como sistema operacional e a IDE Netbeans 7.0.1 (www.oracle.com.br ou www..netbeans.org), de download gratuito, em linguagem de programação JAVA.

Para a conversão da temperatura do sensor (mV) para a rede, foi utilizado o modelo DVP04TC-S do fabricante DELTA ELETRONICS (www.deltaelectronics.com), que possui quatro entradas analógicas, específicas para utilização de termopares. A alimentação desta I/O é 24VCC, sendo necessária a compra de uma fonte auxiliar para sua energização.
Uma foto da I/O encontra-se a seguir:

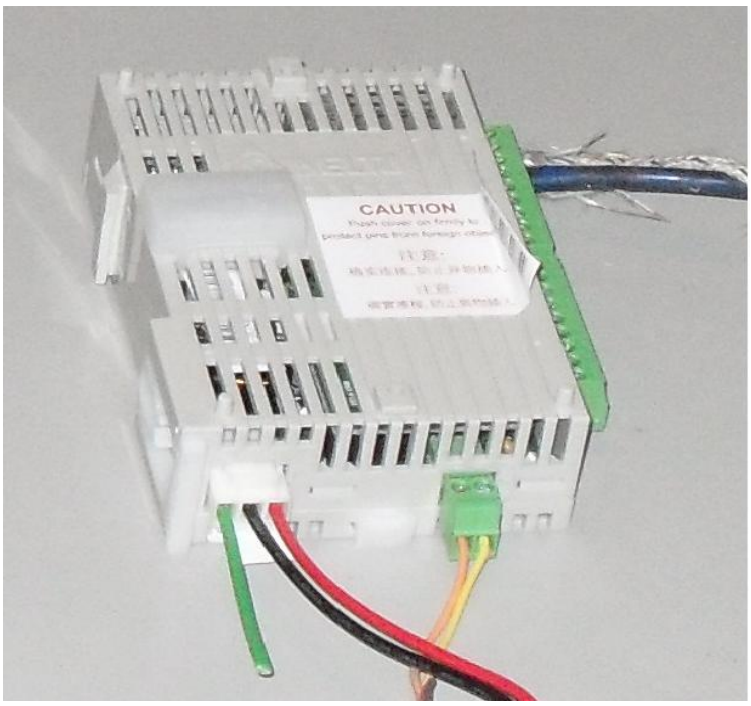


Fig 6.12 – I/O Remota com entrada termopar.

Uma sugestão para as outras três entradas seria o monitoramento da temperatura em diferentes pontos internos do túnel de secagem para verificar se há diferenças de temperatura nestes pontos.

Para a comunicação da rede foi adquirido um cabo USB/SERIAL do fabricante LEADERSHIP, modelo 8430, que fica responsável pela comunicação dos dados.

Foram adquiridos também dois conversores 232/485, para a realização da conversão 232 disponível na saída da serial para a 485 da I/O e também para a conversão 232 disponível no inversor para 485 da rede de dados.

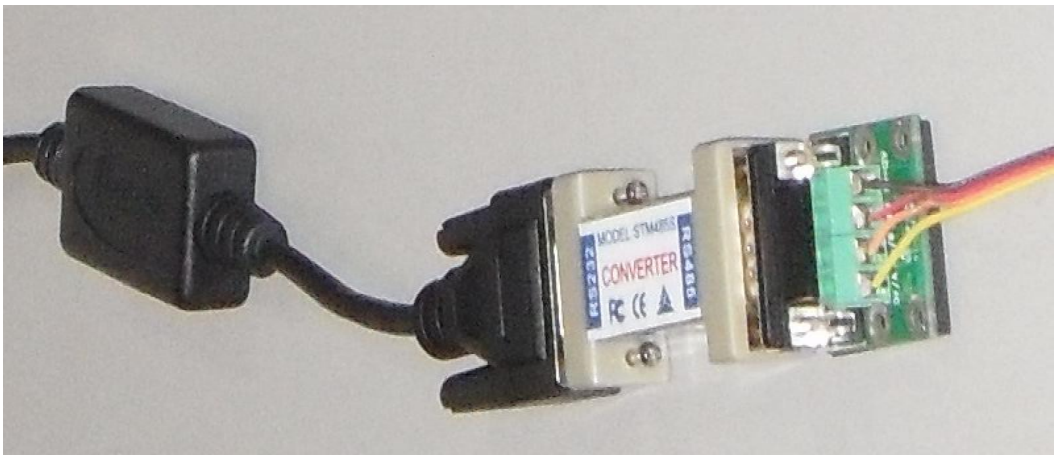


Fig 6.13 – Cabo USB / Serial e conversor 232/485.

Por tratar-se de uma comunicação de dados via porta serial, devemos configurar seus

parâmetros para não termos erros em nossa comunicação. Esses parâmetros devem ser os mesmos para todos os elementos da rede.

Em nosso caso, os parâmetros foram: DATA BITS 8, STOP BIT 1, PARIDADE EVEN (ou PAR), TAXA DE TRANSMISSÃO 9600kbps.

A paridade EVEN foi determinada pela I/O, que só possui esta forma, portanto determinando a escolha.

A porta de comunicação escolhida foi a COM3, porém, com a utilização do cabo USB/SERIAL podemos escolher tantas portas quantas entradas USB o nosso dispositivo possuir. Entretanto, devemos nos certificar que a porta tenha obrigatoriamente a configuração acima, e devemos também, no código de nosso software, indicar a porta de utilização.

O protocolo lógico utilizado foi o ModBus RTU. Apesar da I/O também trabalhar com o formato ASCII, o inversor apenas trabalha no modo RTU, com os caracteres no formato hexadecimal.

O inversor de frequência foi alocado no endereço 1 da rede, e a I/O no endereço 2.

Basicamente o que realizamos é uma função “WRITE REGISTER” para o inversor, em que as strings ou telegramas de escrita podem: Habilitar / Desabilitar, alterar o sentido de giro, e alterar a referência de velocidade.

Para cada envio de “WRITE REGISTER” há a verificação ou “ACK” para certificarmos que o inversor acatou o telegrama.

Um exemplo de string para habilitar o inversor é: "0106138B0303BD95".

Decodificação:

“ACK” do Inversor: a mesma string de envio.

Decodificação: 01h – endereço do inversor; 06h- Função WRITE REGISTER; 13h – Registrador(High), 8Bh – Registrador (low), 03h – Valor (High), 03h – Valor (low), BDh – CRC-, 95h – CRC +.

Já no caso da I/O o que é feito o envio de um “READ REGISTER”. A I/O retorna uma string de dados com a referência de temperatura.

Exemplo de “READ REGISTER”: "0203409C000151D7".

Decodificação: 02h – endereço do I/O, 03H – comando READ REGISTER, 409Ch - endereço da função, 0001h – Word Count, 51D7h- LRC.

Resposta da I/O: “02030200EC825215”
02h – Endereço da I/O, 03h, comando READ REGISTER, 02 – Registrador, 00ECh – valor da leitura, o restante da String não foi decodificado, mas assemelha-se a um byte de CRC- e CRC +.

Para o valor lido, 00ECh indica 236, equivalente a 23,6ºC.

Para a construção de nossas strings foram utilizados dois softwares adicionais, de propriedades dos fabricantes do inversor e da I/O, porém de download free.
No caso do inversor o software é o Multicom versão 1.2, disponível em www.weg.com.br, que realiza testes de comunicação do PC com o inversor.
Para cada comando de leitura ou escrita de um parâmetro, o mesmo escreve a string de envio/retorno.

Já no caso da I/O o software utilizado é o DTCOM, disponível em www.deltaelectronics.com, que é utilizado para alterar as configurações da I/O, como por exemplo o tipo de temopar utilizado, endereço que ocupará na rede, velocidade de transmissão etc.
De forma similar, indica o formato da string, incluindo o bit de checagem (algoritmo LRC). Desta forma, não foi necessário realizar o bit de checagem em nosso código-fonte, pois as strings já estavam prontas.

A utilização destes softwares foi vital, pois através dos mesmos é que foram implementadas praticamente todas as funções de escrita e leitura para nossos dispositivos.

Para a simulação de controle e testes de software foi construído um protótipo em bancada, que utilizou um PC Desktop defasado, um forno elétrico caseiro, um termopar tipo K e um motor de 0,75 cv trifásico, não mais utilizado na fábrica. O inversor de frequência é do mesmo modelo utilizado pela máquina.

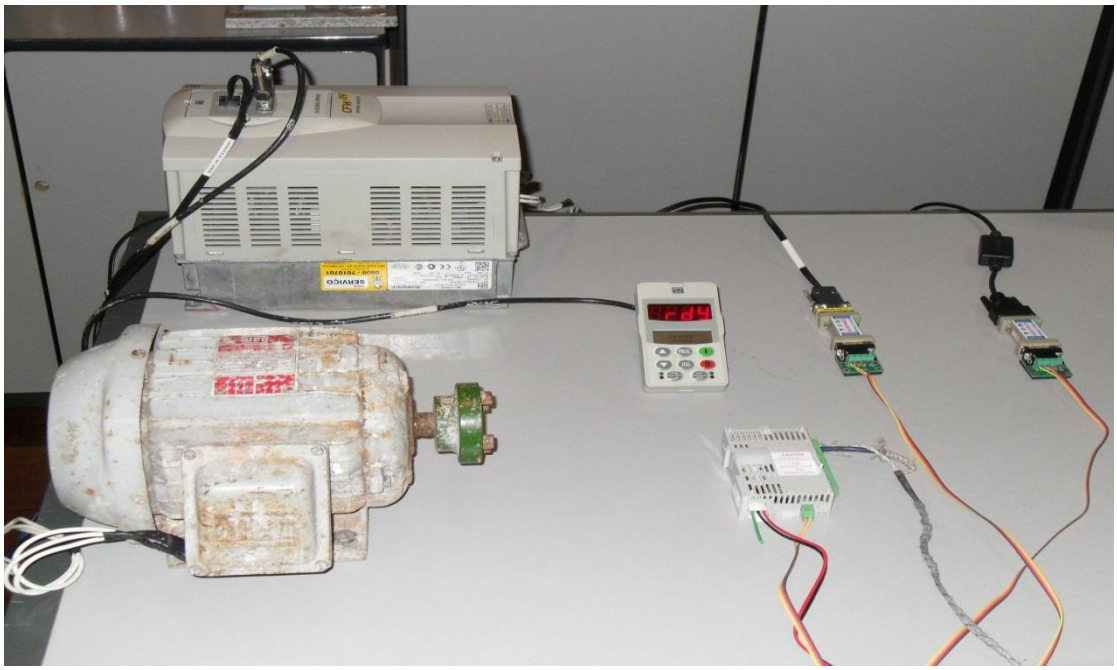


Fig 6.14 – Inversor, motor, IHM, conversores 232/485 e I/O Termopar/485.

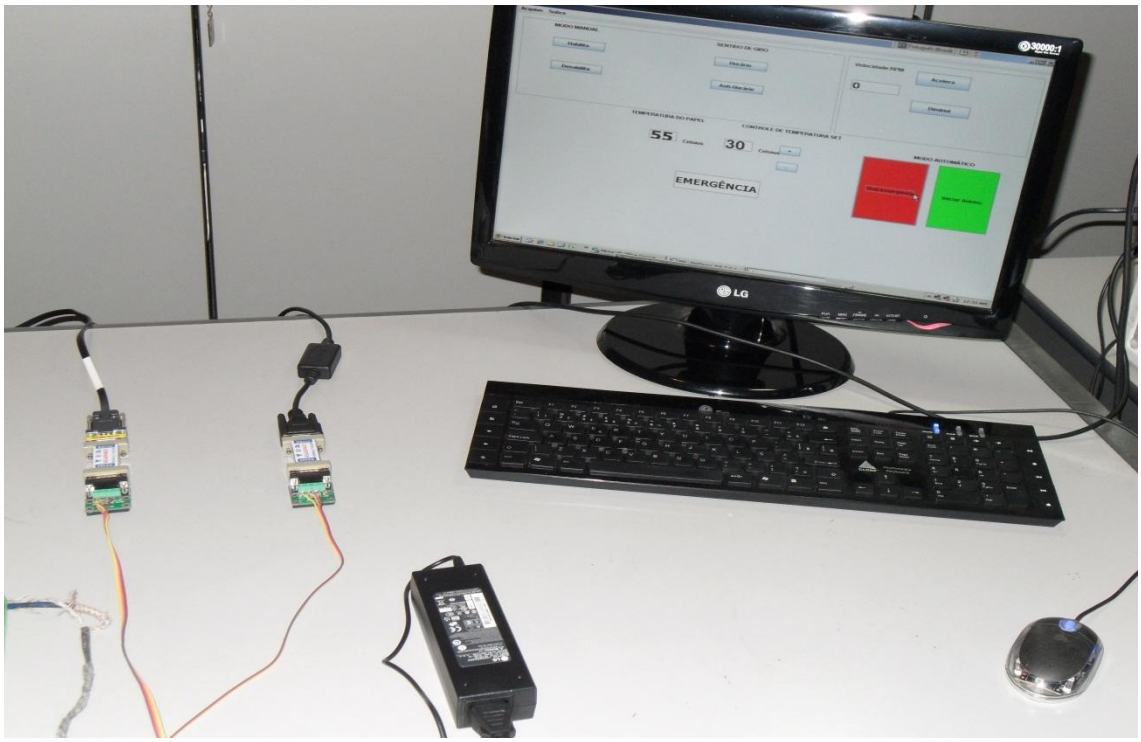


Fig 6.15 – IHM na tela do PC e fonte 24V D.C do I/O Termopar/485.

Fig 6.16 – PC e forno elétrico com termopar, simulando o forno do coater.

Coloca-se uma referência de temperatura para o papel, ligar o forno e acionar o motor. Verificar a sua movimentação, alteração de sentido de giro, e simular o modo automático através da leitura de temperaturas do forno caseiro e observar a variação de velocidade no eixo do motor. O registro de velocidades é feito na IHM, e também na IHM do inversor.

Infelizmente o sistema não é retroalimentado, ou seja, não há como verificar o resfriamento da temperatura com o aumento de velocidade, porém podemos ter uma boa idéia do sistema e sua funcionalidade.

A seguir iremos apresentar alguns aspectos relevantes de nosso código-fonte para o entendimento de sua implementação.

A listagem completa com os detalhes de comandos e funções encontra-se em CD anexo.

Basicamente o nosso software é dividido em cinco grandes classes. Faremos uma breve descrição das classes principais para o entendimento do código:

É nesta classe que é feita a construção gráfica de nossa IHM e o posicionamento de botões de comando e telas de visualização do processo.
O que for configurado nesta classe é o que o operador da máquina irá visualizar quando o sistema estiver controlando a máquina.

Por exemplo, para o botão “Habilita”:

```
private void jButtonZposActionPerformed(java.awt.event.ActionEvent evt) {
      int incr = 1;
      try {
         controller.habilita(incr);
        jTextField1.setText(""+rpm);
        String temp = Integer.toString(temperatura);
        jTextField2.setText(""+temperatura);
        jTextField4.setText("MANUAL");
    //    controller.leitura_termometro(incr);
```

```
        jTextField3.setText("25");
        } catch (Exception ex) {
      }
//    Habilita
      }
```

Abaixo temos um exemplo de IHM gerada para o modo “MANUAL”.

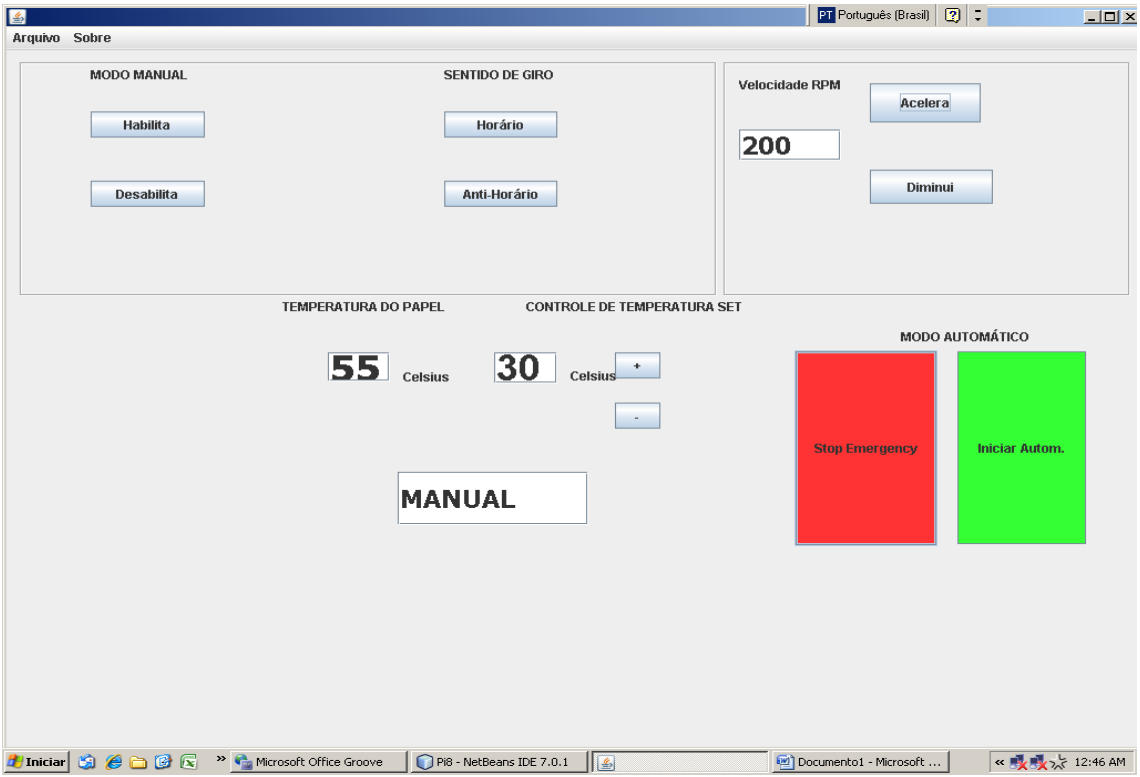


Fig 6.17 – IHM gerada pelo software – modo “MANUAL”.

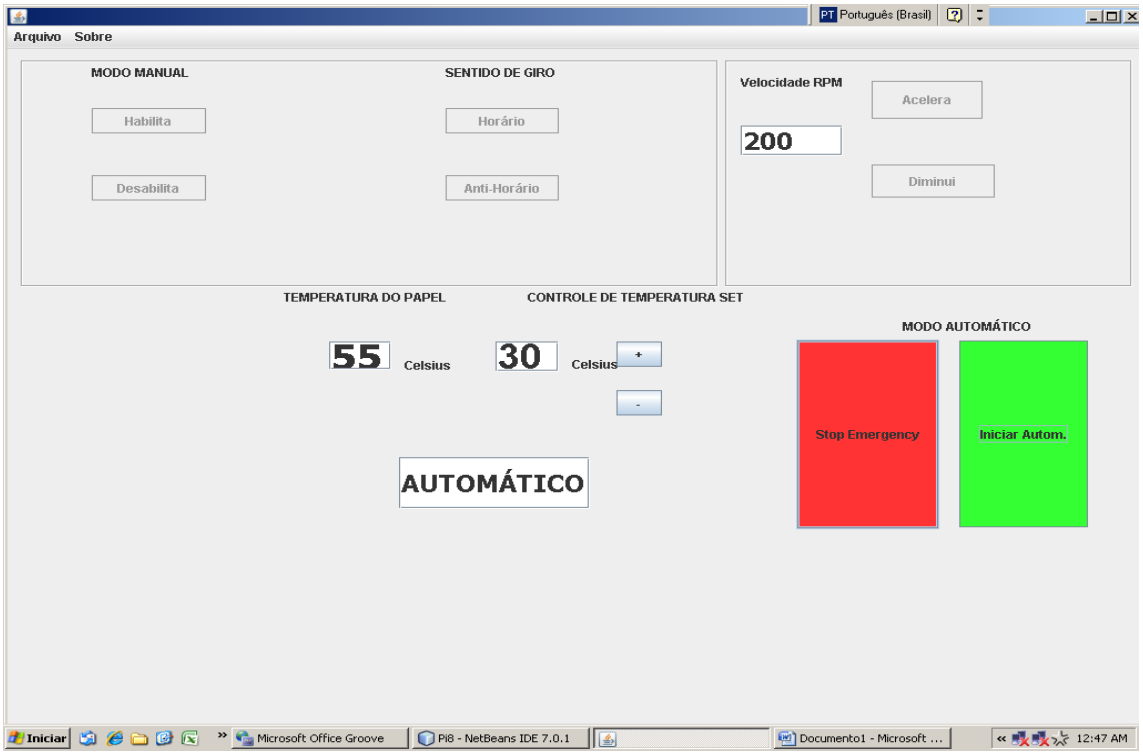


Fig 6.18 – IHM gerada para o modo "AUTOMÁTICO".

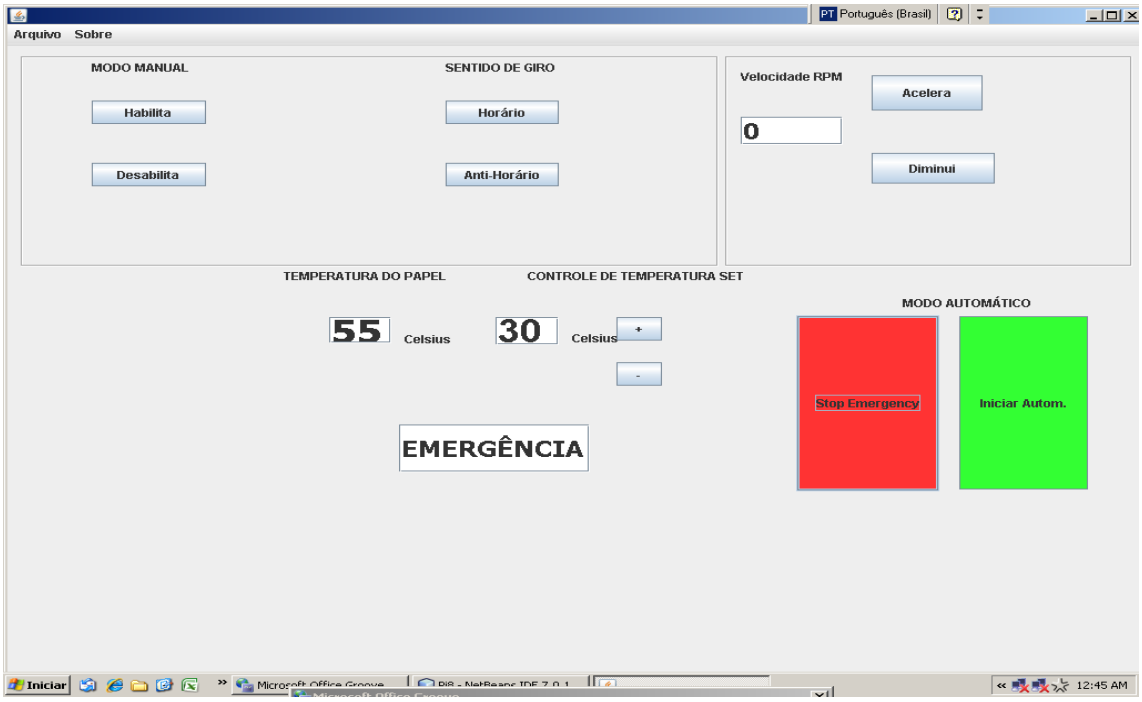


Fig 6.19 – IHM gerada para o modo "EMERGÊNCIA".

Nesta classe estão inseridas as funções de comando atreladas aos botões nas telas. Ou seja,

se o operador clicar em “Habilitar”, há uma função específica dentro da classe Controller que irá ativar uma determinada função. No exemplo, irá enviar uma string específica para habilitar o inversor.

Exemplo da função “Habilita”:

```
public void habilita (int inc) throws Exception {
        String frame="0106138C019ACD5E";
         byte[] data1 = hexStringToByteArray(frame);
         ihm.rpm="50";
         commController.sendManual(data1);
         Thread.sleep (50);
        // Ao habilitar parte com 50 rpm para evitar acidentes.
         String frame1="0106138B0303BD95";
         byte[] data2 = hexStringToByteArray(frame1);
         commController.sendManual(data2);
    } // Habilita
```

Nesta classe é que são definidos os parâmetros da comunicação serial, e as funções de envio e leitura das strings.

Percebe-se que há um efeito cascata entre as classes:

$$IHM \rightarrow CONTROLLER \rightarrow COMMCONTROLLER$$

Exemplo de rotina para enviar dados pela serial:

```
public void sendManual(byte[] data1) throws Exception {
        outputSerialStream.write(data1);
        outputSerialStream.flush();
    } // send
```

Exemplo de abertura de porta serial:

```
public void connect1 ( String portName ) throws Exception {
      CommPortIdentifier portIdentifier = CommPortIdentifier.getPortIdentifier(portName);
      if ( portIdentifier.isCurrentlyOwned() ) {
         System.out.println("Error: Port is currently in use");
      }  else {
         CommPort commPort = portIdentifier.open(this.getClass().getName(),2000);
         if ( commPort instanceof SerialPort ) {
            serialPort = (SerialPort) commPort;

serialPort.setSerialPortParams(9600,SerialPort.DATABITS_8,SerialPort.STOPBITS_1,SerialPort.P
ARITY_EVEN);
            inputSerialStream = serialPort.getInputStream();
            outputSerialStream = serialPort.getOutputStream();
         } else {
            System.out.println("Error: Only serial ports are handled.");
          }
      }
   } // connect
```

Operando no modo manual, há em nossa IHM o botão “TEMP”, que realiza a leitura de temperatura em nosso sensor.

Operando no modo automático, devemos ter uma rotina que realiza a aquisição ou leitura de temperatura de forma cíclica, com intervalos de tempo definidos.

Esta implementação foi feita através de um “aperto” do botão “TEMP” de forma automática, ou seja, é como se o operador estivesse aquisitando a temperatura transcorridos intervalos fixos de tempo.

Isto é feito através da inserção da função Thread, listada abaixo:

```
Thread activator = new Thread("Activator") {
      @Override
      public void run() {
            while(true) {
            try{
```

```
                jButtonIniciarAutomaticoActionPerformed(null);
                Thread.sleep(10000);
                } catch(InterruptedException ie){}
              }
          }
        };
    activator.setDaemon(true);
    activator.start();
        }
```

O intervalo de tempo é determinado em ms. Ou seja, para o exemplo para cada 10000ms é realizado o aperto do botão “TEMP” e o valor lido é mostrado na IHM.

Após a construção do código-fonte e sua devida compilação, iremos realizar o teste de nosso sistema.

Após verificar todas as conexões e fontes de energia, rodamos o código fonte, a IHM ocupou a tela do PC e iniciou-se pela aquisição das temperaturas.

Para efeitos de calibração do termopar, foi feita a comparação com um termômetro digital portátil e realmente a temperatura mostrada na tela do PC era a mesma do termômetro auxiliar.

Um problema ocorreu na comunicação entre o PC e o inversor, que não estava acatando aos comandos.
Após muito trabalho, verificou-se que o inversor não estava reconhecendo a conversão 485/232.
Após consultar a assistência técnica do fabricante, foi afirmado que essa conversão pode apresentar falhas, e que o correto seria instalar uma placa de comunicação 485, ao custo de R$ 620,00.
A instalação dessa placa resolveria o problema, e não seria necessário o conversor, porém a um custo adicional.

Percebeu-se que o inversor desempenhava sua função somente na comunicação 232, ponto a ponto.
Uma opção realizada foi uma "ponte" entre os cabos RX, TX e GND da serial que alimentavam o inversor e que agora alimentariam também o conversor 232/485 da I/O.
Esta solução funcionou para o inversor, porém houve uma distorção e instabilidade dos valores lidos de temperatura, invalidando a opção.

Diante desse quadro, optou-se pela comunicação através de duas portas seriais, a saber COM3, já utilizada, e a COM4.
Foi necessária a aquisição de mais um cabo USB/Serial, e, após a configuração dos parâmetros da porta, foi feita a implementação de abertura das portas no software.
Novamente a solução não funcionou da forma com que foi implementada. O JAVA estava acatando somente a abertura da segunda porta solicitada. Exemplo: abertura da COM3 e em seguida COM4, o resultado era a abertura somente da COM4.
Este problema não foi resolvido em JAVA, e a solução foi portanto uma **alternância** na abertura das portas seriais, sendo a COM3 default.
Logo, para cada solicitação de leitura de temperatura, fecha-se a COM3, abre-se a COM4, realiza-se a aquisição de temperatura, fecha-se a COM4 e novamente abre-se a COM3, preferencial aos comandos do inversor.

Este procedimento foi feito de forma auxiliar, porém o correto seria a aquisição da placa de comunicação 485 para o inversor.

Após a solução dos problemas de comunicação, o sistema funcionou de forma perfeita, tanto no modo manual quanto no automático, em que as leituras eram realizadas e, de acordo com o set de temperatura da tela, ocorria a alteração automática da referência de velocidade do motor.
Fez-se a simulação por um tempo prolongado e em datas subsequentes, e problemas não foram verificados.

7 – CONCLUSÕES

Através do presente trabalho pode-se ter uma idéia geral de como os papéis especiais são fabricados através do detalhamento das partes constituintes básicas de um coater.

A modelagem térmica do sistema auxiliou na compreensão do processo de secagem e serviu para quantificar a potência térmica necessária para um aumento de produtividade do equipamento.

O modelo de jato de ar, que é a melhor configuração para este tipo de processo de secagem, será implementado na máquina, e certamente será percebido um aumento de performance.

A seguir foram observados aspectos relevantes ao processo e as características da planta, para enfim gerarmos um modelo a ser controlado.

Foram apresentadas soluções de controle, e uma matriz de decisão acabou por escolher a melhor solução, ou a mais conveniente para a aplicação.

Os resultados apresentados no protótipo foram bastante satisfatórios e, como já comentado, o sistema será instalado na máquina nas festas de fim de ano.

Desta forma podemos afirmar que os objetivos do trabalho foram atingidos, e estabeleceu-se uma solução de baixíssimo custo quando comparadas às soluções tradicionais. Em números, representa praticamente 1% do que se encontrou no mercado. É portanto uma solução ideal para empresas de pequeno e médio porte, não sendo restrita à secagem de papel. Todo processo de secagem contínua pode ter esse sistema instalado, tais como madeira, grãos, alimentos, ração animal, tecidos etc.

Algumas simplificações foram utilizadas na implantação do modelo, sendo a principal referente à área de atuação de cada bico, admitida como um círculo de raio 0,5m, e na não interferência de um arranjo para o seguinte. Outro fator foi o número de Reynolds calculado estar acima do intervalo de aplicação da equação proposta no modelo do jato.

Porém, as outras condições foram satisfeitas.

Admitiu-se também uma umidade relativa do ar a 70%. No entanto, de acordo com empresas de previsão do tempo, esse valor varia muito durante o dia, e também com as estações do ano. Dados podem ser adquiridos nas páginas na Internet destas empresas.

Esta variação da umidade relativa altera muito a concentração de vapor no ar, tendo como consequência a variação na massa específica do ar quando aquecido pelo queimador, e finalmente na diferença de concentrações da água na equação de transferência de massa, alterando-se seu resultado de forma significante.

No entanto podemos afirmar que nosso modelo de controle não sofre com essa variação de umidade relativa do ar durante o turno de operação, justamente por medir o incremento de temperatura ao final do processo, indicativo da secagem do papel. O que pode ocorrer, neste caso, é uma redução da produtividade da máquina, que pode ser compensada por um acréscimo de potência térmica pelos queimadores, já que os mesmos, para a condição calculada neste trabalho, operam abaixo de suas capacidades nominais.

Como sugestão de continuidade ao presente trabalho, podemos citar a automação de acionamento e controle dos demais itens que compõe a máquina, como rebobinador, bomba de tinta, alinhador de banda lateral, ventoinhas, secadores, faca de ar etc.

Ou seja, podemos inserir em nosso software todos os componentes que fazem parte do processo e controlar seu acionamento por exemplo a eventos discretos.

Para tanto basta realizar um diagrama SFC ou mesmo uma rede de Petri a eventos discretos.

Vale lembrar que uma rede 485 suporta até 255 componentes, e sua inserção na rede pode ser feita através das I/O´s.
Caso esse conjunto seja implementado, podemos, para não sobrecarregarmos o software, deixar nossa IHM apenas para acionamento e monitoramento dos elementos da máquina, e o eventual controle destes componentes ser realizado através de PID´s que se comunicam com a rede.

Ou seja, a rotina de controle fica fora do software, e somente o acionamento, seguindo a lógica da rede de Petri é que será implementada, em conjunto com o monitoramento das variáveis do processo.

8- BIBLIOGRAFIA

[1] Frank P. Incropera, David P. DeWitt – *Fundamentos de Transferência de Calor e de Massa* 5º edição– 2003 – LTC editora

[2] Johan Gullischsen, Hannu Paulapuro, Esa Lehtinen, Brian Attwood – *Papermaking Part 2, Drying – book 9 of Papermaking Science and Technology- Fapet* – 2000 – Helsinki, Finland

[3] Sonntag, Richard E, Borgnakke, Claus, Wylen, Gordon J. Van – *Fundamentos da Termodinâmica* – 5o edição – 1998 – Editora Edgard Blucher Ltda

[4] Spalding, D.B., *Convective Mass Transfer,* McGraw-Hill, New York, 1963.

[5]www.lenox.ind.br – Catálogo Mutec instruments GmbH. www.muetec.de

[6]www.ndcinfrared.com. UK. NDC Infrared Engineering Ltd. NDC Brasil, www.ndcinfrared.com.br

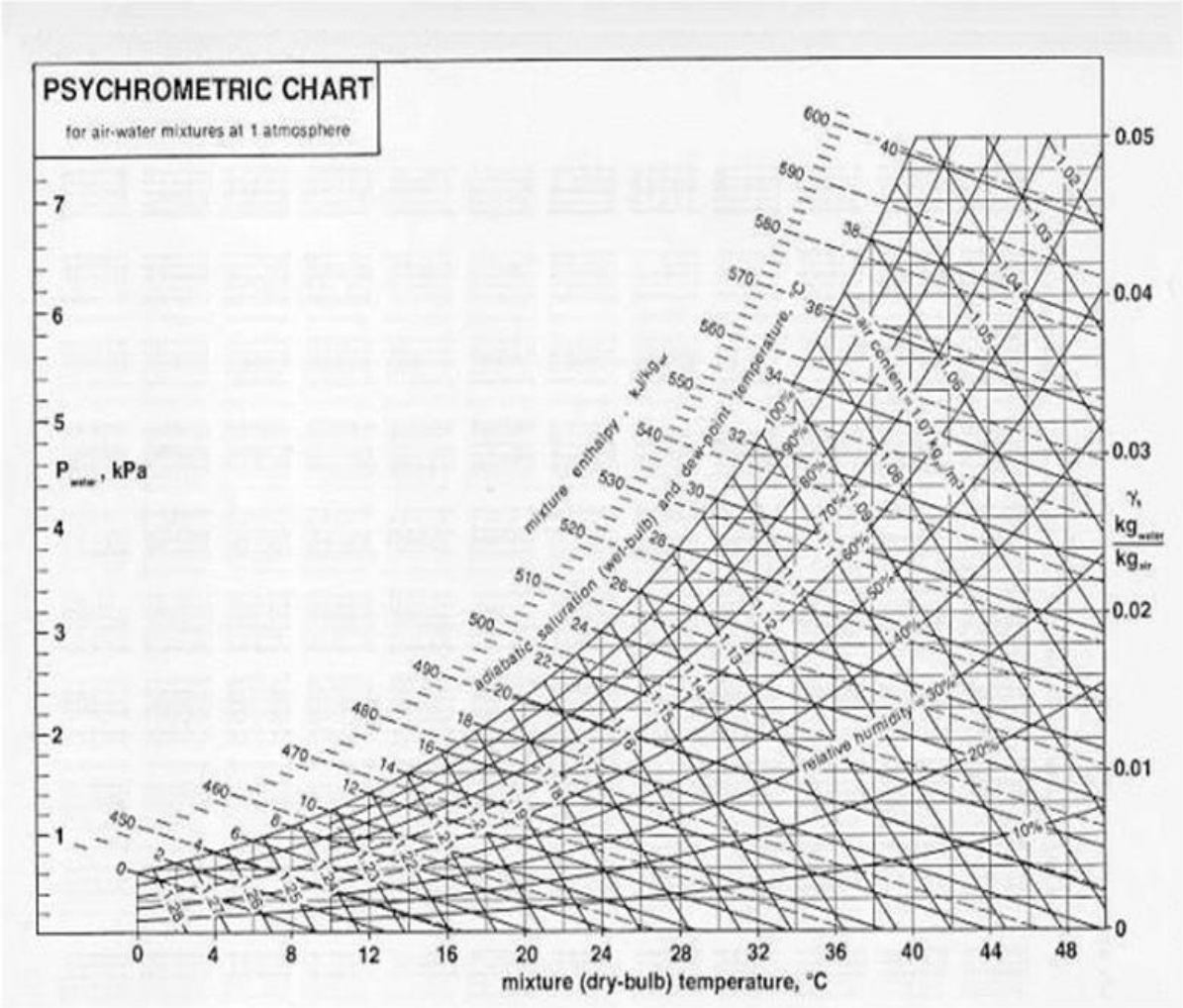
[7] Zhangzhou Yujie Electronics Co., Ltd, yujie-sales@yuj.com.cn

[8] Ogata, Katsuhiko – *Engenharia de Controle Moderno* 3ºEdição- 2000 – LTC Editora.

[9] Horton, Ivor- *Beginning Java 2* – 1999 – Wrox Press, Birmingham, UK

[10] Sekkuar – *Implementação de rotina thread* - Fórum on line sobre programação java - 18/11/2011. http://javafree.uol.com.br/viewprofile.jbb?u=38647

ANEXO A: CARTA PSICROMÉTRICA



Extraído de *www.fem.unicamp.br/~em672/**psicrometria**.ppt*

ANEXO B: CÓDIGO-FONTE

O código fonte está dividido em seus pacotes ou classes principais.

CLASSE MAIN

```
package Pi8;

public class Main {

   public static void main(String[] args) {
     try {
       // Cria os componentes
       CalculadoraIHM ihm = new CalculadoraIHM();
       Controller controller = new Controller();
       CommunicationController commController = new CommunicationController();
       int i = 0;
       Monitor monitor = new Monitor();

       // Estabelece a relacao entre os componentes.
       // Seria mais correto criar Interfaces para Controller e IHM
       // ao invés de passar diretamente a referência,
       // como feito abaixo, por simplicidade.
       ihm.setController(controller);

       controller.setCommController(commController);
       controller.setIHM(ihm);

       monitor.setCommController(commController);
       monitor.setIHM(ihm);

         commController.init1("COM4");


       // Abre a IHM
       ihm.setVisible(true);

       // inicia a thread de monitoracao; não pode ser executada ao mesmo
       // tempo que controller: executada apenas quando o botão executar do
       // modo automático é acionado
       while(i == 0){
             if(controller.enableMonitor == true){
                monitor.run();
                i=1;
           }
       }


     } catch (Exception e) {
        System.out.println("Erro durante inicializacao");
        e.printStackTrace();
     }
   } // main

} // Main
```

CLASSE MONITOR

```
package Pi8;

public class Monitor extends Thread {

   private CalculadoraIHM ihm;
   private CommunicationController comm;

   public void setIHM(CalculadoraIHM ihm) {
      this.ihm = ihm;
   } // setIHM

   public void setCommController(CommunicationController comm) {
      this.comm = comm;
   } // setCommController


       // while true
   } // run
 // Monitor
```

CLASSE TIMER

```
package Pi8;

public class Timer extends Thread {

   private long timeout;
   private boolean timedOut =false;

   public Timer (long timeout) {
      this.timeout = timeout;
   } // constructor

   public void run() {
      try {
         timedOut = false;
         sleep(timeout);
         timedOut = true;
      } catch(Exception e) {
         e.printStackTrace();
      }
   } // run

   public boolean timedOut() {
      return timedOut;
   }

} // Timer
```

CLASSE CONTROLLER

```
package Pi8;

public class Controller {

   public enum State  { IDLE, EXECUTING_RIDE, SUSPENDED };
```

```
    private State state = State.IDLE;
    private CalculadoraIHM ihm = new CalculadoraIHM();
    private CommunicationController commController = null;
    private CommReturn ret = null;
    public boolean enableMonitor = false;
    public int rpm=0;
//    int data4, data5;

    public void setIHM(CalculadoraIHM ihm) {
        this.ihm = ihm;
    } // setIHM


    public void setCommController(CommunicationController commController) {
        this.commController = commController;
    }// setCommController

    public void executeRide()  {

      try {

          enableMonitor=true;
      }  catch(Exception e)  {
          e.printStackTrace();

      }
    }

    public void suspend() throws Exception {

       if (state.equals(State.EXECUTING_RIDE))  {


          state=State.SUSPENDED;
          //ihm.enterSuspendMode();
          enableMonitor=false;
       }
       else  if (state.equals(State.SUSPENDED))  {
          state=State.EXECUTING_RIDE;
          //ihm.enterExecuteMode();
          enableMonitor=true;
       }


    }
   public State getState() {
      return state;
   }
   public static int byteArrayToInt(byte[] b, int offset) {
       int value = 0;
       for (int i = 0; i < 4; i++) {
          int shift = (4 - 1 - i) * 8;
          value += (b[i + offset] & 0x000000FF) << shift;
       }
       return value;
    }
```

```
// i==2, quer só o valor da temp em hexa do I/O remoto.
 public static byte[] hexStringToByteArray(String s){

 int len = s.length();
 byte[] data = new byte[len / 2];
 for (int i=0; i < len; i+=2) {
    data[i / 2]=(byte)  ((Character.digit(s.charAt(i),16)<<4)
                        + Character.digit(s.charAt(i+1),16));

 }
 return data;
 }
 public void habilita (int inc) throws Exception {


    String frame="0106138C019ACD5E";

    byte[] data1 = hexStringToByteArray(frame);

    ihm.rpm="50";
    commController.sendManual(data1);

    Thread.sleep (50);
    // Ao habilitar parte com 50 rpm para evitar acidentes.

    String frame1="0106138B0303BD95";

    byte[] data2 = hexStringToByteArray(frame1);


    commController.sendManual(data2);
 } // Habilita

 public void desabilita(int inc) throws Exception {

    String frame="0106138B0300FD94";

    byte[] data1 = hexStringToByteArray(frame);



    commController.sendManual(data1);
 } // Desabilita

 public void leitura_termometro(int inc) throws Exception {
   // commController.disconnect();
   // Thread.sleep(1000);
 //   commController.init2("COM5");
  //  Thread.sleep(3000);

    byte [] data3 = new byte [16];

    String frame="0203409C000151D7";
    // string de leitura
    byte[] data1 = hexStringToByteArray(frame);

    commController.readManual(data1);
```

```
}

public void trocaporta1(int incr) throws Exception {
   commController.serialPort.close();
   Thread.sleep(1500);

   commController.init2("COM3");



}

 public void trocaporta2(int incr) throws Exception {
   commController.serialPort.close();
                //  Thread.sleep(50);

   commController.init2("COM4");



}
public void horario(int inc) throws Exception {

   String frame = "0106138B0404FE67";


   byte[] data1 = hexStringToByteArray(frame);

   commController.sendManual(data1);


} // Horário

public void antihorario(int inc) throws Exception {

   String frame = "0106138B0400FFA4";


   byte[] data1 = hexStringToByteArray(frame);
   commController.sendManual(data1);
} // Anti-Horário


public void acelera(double velocidade) throws Exception {

   if  (velocidade == 1)  {

   String frame = "0106138C019ACD5E";

   byte[] data1 = hexStringToByteArray(frame);

   commController.sendManual(data1);
   ihm.rpm="50";
  }

else
   if  (velocidade == 2)  {

   String frame = "0106138C03344D82";
```

```
    byte[] data1 = hexStringToByteArray(frame);

    commController.sendManual(data1);
    ihm.rpm = "100";
    }

else
    if  (velocidade == 3)   {

    String frame = "0106138C04EC4FE8";

    byte[] data1 = hexStringToByteArray(frame);

    commController.sendManual(data1);
    ihm.rpm="150";
    }

else
    if  (velocidade == 4)    {

    String frame = "0106138C06684EEB";

    byte[] data1 = hexStringToByteArray(frame);

    commController.sendManual(data1);
    ihm.rpm="200";
    }

else
    if  (velocidade == 5)     {

    String frame = "0106138C08344AB2";

    byte[] data1 = hexStringToByteArray(frame);

    commController.sendManual(data1);
    ihm.rpm="250";
    }
else
    if  (velocidade == 6)     {

    String frame = "0106138C0992CB58";

    byte[] data1 = hexStringToByteArray(frame);

    commController.sendManual(data1);
    ihm.rpm="300";
    }
else

    if  (velocidade == 7)      {

    String frame = "0106138C0B22CB8C";

    byte[] data1 = hexStringToByteArray(frame);

    commController.sendManual(data1);
    ihm.rpm="350";
    }
else
```

```
    if  (velocidade == 8)       {

    String frame = "0106138C0CD04839";

    byte[] data1 = hexStringToByteArray(frame);

    commController.sendManual(data1);
    ihm.rpm="400";
    }
  else
    if  (velocidade == 9)       {

    String frame = "0106138C0E4C4930";

    byte[] data1 = hexStringToByteArray(frame);

    commController.sendManual(data1);
    ihm.rpm="450";
    }
  else
    if  (velocidade == 10)      {

    String frame = "0106138C0FDC48CC";

    byte[] data1 = hexStringToByteArray(frame);

    commController.sendManual(data1);
    ihm.rpm="500";
    }

  else
    if  (velocidade == 11)      {

    String frame = "0106138C1176C113";

    byte[] data1 = hexStringToByteArray(frame);

    commController.sendManual(data1);
    ihm.rpm="550";
    }
  else
    if  (velocidade == 12)      {

    String frame = "0106138C13104059";

    byte[] data1 = hexStringToByteArray(frame);

    commController.sendManual(data1);
    ihm.rpm="600";
    }

    if (velocidade == 13)       {

    String frame = "0106138C14AAC3DA";

    byte[] data1 = hexStringToByteArray(frame);

    commController.sendManual(data1);
    ihm.rpm="650";
    }
```

```
else
   if  (velocidade == 14)     {

   String frame = "0106138C164442F6";

   byte[] data1 = hexStringToByteArray(frame);

   commController.sendManual(data1);
   ihm.rpm="700";
   }
else
   if  (velocidade == 15)     {

   String frame = "0106138C17DEC30D";

   byte[] data1 = hexStringToByteArray(frame);

   commController.sendManual(data1);
   ihm.rpm="750";
   }
else
   if  (velocidade == 16)     {

   String frame = "0106138C19784717";

   byte[] data1 = hexStringToByteArray(frame);

   commController.sendManual(data1);
   ihm.rpm="800";
   }
else
   if (velocidade == 17)      {

   String frame = "0106138C1B12C658";

   byte[] data1 = hexStringToByteArray(frame);

   commController.sendManual(data1);
   ihm.rpm="850";
   }
 else

   if  (velocidade == 18)     {

   String frame = "0106138C1CAC4418";

   byte[] data1 = hexStringToByteArray(frame);

   commController.sendManual(data1);
   ihm.rpm="900";
   }
 else
   if  (velocidade == 19)     {

   String frame = "0106138C1E46C4F7";

   byte[] data1 = hexStringToByteArray(frame);

   commController.sendManual(data1);
   ihm.rpm="950";
```

```
      }
    else
      if  (velocidade == 20)      {

      String frame = "0106138C1FE0451D";

      byte[] data1 = hexStringToByteArray(frame);

      commController.sendManual(data1);
      ihm.rpm="1000";
      }
   } // Define a velocidade;


}
```

CLASSE COMMCONTROLLER

```
package Pi8;

import java.util.Map;
import java.util.HashMap;
import gnu.io.CommPort;
import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;

import java.io.FileDescriptor;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.lang.Thread;



public class CommunicationController {

   public enum Register { INICIAR, CONTINUAR, SUSPENDER, X, Y, PROGRAM_LINE,
VELOCIDADE, HOME, JOGX, JOGY };
   public static int SUCCESS = 0;
   public static int LRC_ERROR = -1;
   public static int TIMEOUT_ERROR = -2;

   private Map<Register, Integer> addressMap = new HashMap<Register, Integer>();
   private String commPortName = null;
   public SerialPort serialPort = null;
   private InputStream inputSerialStream = null;
   private OutputStream outputSerialStream = null;


   private static int NUMBER_OF_RETRIES = 15;
   private static long FRAME_TIMEOUT = 60000;
   private String sync = new String();

   public String temp_sensor3, temp_sensor4, temp_sensor5="", temp_sensor7, temp_sensor6;
   public String temp_sensor0, temp_sensor2, temp_sensor8, temp_sensor9;
   public boolean loop=true, loop1=true;
```

```
public int temp_sensor10, temp_sensor1, temp3, temp4, temp5, temp6, temp7;
public byte[] temp = new byte[2];
private enum FunctionCode { ReadRegister, WriteRegister, WriteFile };

public CommunicationController() {
  //Implementa Mapa de memória conforme sugerido em aula
  addressMap.put(Register.INICIAR, 0);
  addressMap.put(Register.SUSPENDER, 1);
  addressMap.put(Register.CONTINUAR, 2);
  addressMap.put(Register.JOGX, 4);
  addressMap.put(Register.JOGY, 5);
  addressMap.put(Register.VELOCIDADE, 6);
  addressMap.put(Register.HOME, 7);
  addressMap.put(Register.X, 8);
  addressMap.put(Register.Y, 9);
  addressMap.put(Register.PROGRAM_LINE, 10);
} // constructor

public static int byteArrayToInt(byte[] b, int offset) {
   int value = 0;
   for (int i = 3; i < 5; i++) {
      int shift = (4 - 1 - i) * 8;
      value += (b[i + offset] & 0x000000FF) << shift;
   }
   return value;
}

public void init1(String commPort) throws Exception {
  this.commPortName = commPort;
  connect1(commPort);
} // setCommParameters]

 public void init2(String commPort) throws Exception {
  this.commPortName = commPort;
  connect2(commPort);
} // setCommParameters



private void sendRequest(byte[] frame) throws Exception {
   outputSerialStream.write(frame);
   outputSerialStream.flush();
} // sendRequest

public void sendRequestDirect(int frame) throws Exception {
   outputSerialStream.write(frame);
   outputSerialStream.flush();
} // sendRequestDirect

public void sendManual(byte[] data1) throws Exception {
    outputSerialStream.write(data1);
    outputSerialStream.flush();
} // sendGCode

public void readManual(byte[] data1) throws Exception {
   loop=true;
   loop1=true;

   outputSerialStream.write(data1);
   outputSerialStream.flush();
```

```
byte[] buffer = new byte[16];
int idx = 0;


while (  (idx < 9) && (loop==true) ) {
   if ( inputSerialStream.available() > 0 ) {
      int chInt = inputSerialStream.read();

            buffer[idx] = (byte)chInt;

            if (idx==0) {
            temp_sensor0=String.valueOf(chInt);
            temp_sensor1=Integer.valueOf(temp_sensor0, 16).intValue();


            }

            if (idx==1) {
            temp_sensor0=String.valueOf(chInt);
            temp_sensor1=Integer.valueOf(temp_sensor0, 16).intValue();



            }

            if (idx==2) {
            temp_sensor0=String.valueOf(chInt);


            }

            if (idx==3) {
            temp_sensor3=String.valueOf(chInt);

            temp3 = chInt*100;
            System.out.println("got " + chInt*100 );


            }

            if (idx==4) {
            temp_sensor4=String.valueOf(chInt);
            temp4=chInt;

            System.out.println("got " + chInt);

            }

             if (idx==5) {

            temp_sensor0=String.valueOf(chInt);



            }


             if (idx==6) {
```

```
                temp_sensor0=String.valueOf(chInt);


                }


                if (idx==7) {
                temp_sensor0=String.valueOf(chInt);


                loop=false;


                }


        }

                idx++;
        }


                if(temp3>1){
                temp5=temp3+temp4;
                temp_sensor6=String.valueOf(temp5);
                temp6 = Integer.parseInt(temp_sensor6, 16);
                temp7=temp6/10;
                }
                else
                      {

               //   temp_sensor6=String.valueOf(temp4);
                  temp5=temp4/10;
                  temp_sensor6=String.valueOf(temp5);
                  temp6 = Integer.parseInt(temp_sensor6, 16);
                  temp7=temp6;




                }



                temp_sensor9=String.valueOf(temp7);
                System.out.println("got " + temp_sensor9);

                CalculadoraIHM.temp_sensor=temp_sensor9;
                loop1=false;

}



public void connect1 ( String portName ) throws Exception {
   CommPortIdentifier portIdentifier = CommPortIdentifier.getPortIdentifier(portName);
   if ( portIdentifier.isCurrentlyOwned() ) {
      System.out.println("Error: Port is currently in use");
```

```
        }  else {
           CommPort commPort = portIdentifier.open(this.getClass().getName(),2000);
           if ( commPort instanceof SerialPort ) {
              serialPort = (SerialPort) commPort;

serialPort.setSerialPortParams(9600,SerialPort.DATABITS_8,SerialPort.STOPBITS_1,SerialPort.PARITY_
EVEN);
              inputSerialStream = serialPort.getInputStream();
              outputSerialStream = serialPort.getOutputStream();
           } else {
              System.out.println("Error: Only serial ports are handled.");
           }
        }
     } // connect

     public void connect2 ( String portName ) throws Exception {
        CommPortIdentifier portIdentifier = CommPortIdentifier.getPortIdentifier(portName);
        if ( portIdentifier.isCurrentlyOwned() ) {
           System.out.println("Error: Port is currently in use");
        }  else {
           CommPort commPort = portIdentifier.open(this.getClass().getName(),2000);
           if ( commPort instanceof SerialPort ) {
              serialPort = (SerialPort) commPort;

serialPort.setSerialPortParams(9600,SerialPort.DATABITS_8,SerialPort.STOPBITS_1,SerialPort.PARITY_
EVEN);
              inputSerialStream = serialPort.getInputStream();
              outputSerialStream = serialPort.getOutputStream();
           } else {
              System.out.println("Error: Only serial ports are handled.");
           }
        }
     } // connect


     public void disconnect() {
        try {
           serialPort.close();
        } catch (Exception e) {
           e.printStackTrace();
        }
     } // disconnect

} // CommunicationController
```

CLASSE IHM

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

/*
 * CalculadoraIHM.java
 *
 * Created on 05/09/2011, 11:29:53
 */

package Pi8;
```

```
import java.util.logging.Level;
import java.util.logging.Logger;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.Scanner;

/**
 *
 * @author Jeferson Afonso Lopes de Souza
 */
public class CalculadoraIHM extends javax.swing.JFrame {

    public int modo, velocidade=0, temperatura=70, automatico=0, temp_sensor1, incr1;
    private double SpindleSpeed, incPosX, incPosZ;
    public String rpm, temp_sensor2;
    public static String temp_sensor;
    public static boolean loop=false, loop2;
    private String temp_sensor9;

    //private Calculadora calc = new Calculadora();
    /** Creates new form CalculadoraIHM */
    public CalculadoraIHM() {
        initComponents();
        modo=0;


        Thread activator = new Thread("Activator") {

        @Override
        public void run() {


            while(true) {
                try{
                    jButtonIniciarAutomaticoActionPerformed(null);
                    Thread.sleep(10000);

                } catch(InterruptedException ie){}
            }
        }




    };
    activator.setDaemon(true);
    activator.start();


    }
    private Controller controller = null;

 /************* INÍCIO DA PARTE DE CÓDIGOS COMPLEMENTARES*****************/

    public void setController(Controller controller)  {
```

```
    this.controller=controller;

   }     // setController


 /************** FIM DA PARTE DE CÓDIGOS COMPLEMENTARES*******************/
/*=========================================================================*/
/*=========================================================================*/
/*=========================================================================*/
/********************** INÍCIO DA PARTE GRÁFICA ************************/


   /** This method is called from within the constructor to
    * initialize the form.
    * WARNING: Do NOT modify this code. The content of this method is
    * always regenerated by the Form Editor.
    */
   @SuppressWarnings("unchecked")
   // <editor-fold defaultstate="collapsed" desc="Generated Code">
   private void initComponents() {

      buttonGroup1 = new javax.swing.ButtonGroup();
      jPanel1 = new javax.swing.JPanel();
      jButtonXpos = new javax.swing.JButton();
      jButtonZneg = new javax.swing.JButton();
      jButtonZpos = new javax.swing.JButton();
      jButtonXneg = new javax.swing.JButton();
      jLabel1 = new javax.swing.JLabel();
      jLabel7 = new javax.swing.JLabel();
      jPanel2 = new javax.swing.JPanel();
      jButton1 = new javax.swing.JButton();
      jButton2 = new javax.swing.JButton();
      jTextField2 = new javax.swing.JTextField();
      jTextField3 = new javax.swing.JTextField();
      jLabel4 = new javax.swing.JLabel();
      jLabel5 = new javax.swing.JLabel();
      jButton4 = new javax.swing.JButton();
      jTextField4 = new javax.swing.JTextField();
      jLabel3 = new javax.swing.JLabel();
      jLabel2 = new javax.swing.JLabel();
      jButton5 = new javax.swing.JButton();
      jButton3 = new javax.swing.JButton();
      jPanel3 = new javax.swing.JPanel();
      jPanel4 = new javax.swing.JPanel();
      jButtonZerarX = new javax.swing.JButton();
      jButtonZerarZ = new javax.swing.JButton();
      jLabel9 = new javax.swing.JLabel();
      jTextField1 = new javax.swing.JTextField();
      jButtonAbrir = new javax.swing.JButton();
      jButtonIniciarAutomatico = new javax.swing.JButton();
      jLabel6 = new javax.swing.JLabel();
      jButton6 = new javax.swing.JButton();
      jButton7 = new javax.swing.JButton();
      jButton8 = new javax.swing.JButton();
      jButton9 = new javax.swing.JButton();
      jButton10 = new javax.swing.JButton();
      jTextField5 = new javax.swing.JTextField();
```

```
        jLabel8 = new javax.swing.JLabel();
        jMenuBar1 = new javax.swing.JMenuBar();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jPanel1.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        jButtonXpos.setText("Horário");
        jButtonXpos.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonXposActionPerformed(evt);
            }
        });

        jButtonZneg.setText("Desabilita");
        jButtonZneg.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jButtonZnegMouseClicked(evt);
            }
        });
        jButtonZneg.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonZnegActionPerformed(evt);
            }
        });

        jButtonZpos.setText("Habilita");
        jButtonZpos.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonZposActionPerformed(evt);
            }
        });

        jButtonXneg.setText("Anti-Horário");
        jButtonXneg.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonXnegActionPerformed(evt);
            }
        });

        jLabel1.setText("MODO MANUAL");

        jLabel7.setText("SENTIDO DE GIRO");

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(62, 62, 62)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 275,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                        .addComponent(jButtonZpos, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                        .addComponent(jButtonZneg, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 111, Short.MAX_VALUE)))
```

```
            .addGap(44, 44, 44)
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
               .addGroup(jPanel1Layout.createSequentialGroup()
                  .addComponent(jLabel7, javax.swing.GroupLayout.DEFAULT_SIZE, 93,
Short.MAX_VALUE)
                  .addGap(142, 142, 142))
               .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                     .addComponent(jButtonXpos, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                     .addComponent(jButtonXneg, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 129, Short.MAX_VALUE))
                  .addContainerGap())))
      );
      jPanel1Layout.setVerticalGroup(
         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
         .addGroup(jPanel1Layout.createSequentialGroup()
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                  .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)
                  .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE, 17,
javax.swing.GroupLayout.PREFERRED_SIZE))
               .addGroup(jPanel1Layout.createSequentialGroup()
                  .addGap(47, 47, 47)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                     .addComponent(jButtonZpos)
                     .addComponent(jButtonXpos))))
            .addGap(73, 73, 73)
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
               .addComponent(jButtonZneg)
               .addComponent(jButtonXneg))
            .addContainerGap(46, Short.MAX_VALUE))
      );

      jButton1.setText("+");
      jButton1.addActionListener(new java.awt.event.ActionListener() {
         public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton1ActionPerformed(evt);
         }
      });

      jButton2.setText("-");
      jButton2.addActionListener(new java.awt.event.ActionListener() {
         public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton2ActionPerformed(evt);
         }
      });

      jTextField2.setFont(new java.awt.Font("Arial", 1, 24));
      jTextField2.addActionListener(new java.awt.event.ActionListener() {
         public void actionPerformed(java.awt.event.ActionEvent evt) {
            jTextField2ActionPerformed(evt);
         }
      });
```

```
        jTextField3.setFont(new java.awt.Font("Tahoma", 1, 24));
        jTextField3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jTextField3ActionPerformed(evt);
            }
        });

        jLabel4.setText("Celsius");


        jLabel5.setText("Celsius");

        jButton4.setText("TEMP");
        jButton4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });

        jTextField4.setFont(new java.awt.Font("Tahoma", 1, 24));

        jLabel3.setText("TEMPERATURA DO PAPEL");

        jLabel2.setText("CONTROLE DE TEMPERATURA SET");

        jButton5.setBackground(new java.awt.Color(255, 204, 51));
        jButton5.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton5ActionPerformed(evt);
            }
        });

        jButton3.setBackground(new java.awt.Color(51, 255, 51));
        jButton3.setForeground(new java.awt.Color(255, 102, 102));
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });
        jButton3.addPropertyChangeListener(new java.beans.PropertyChangeListener() {
            public void propertyChange(java.beans.PropertyChangeEvent evt) {
                jButton3PropertyChange(evt);
            }
        });

        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addGap(161, 161, 161)
                        .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 8,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addComponent(jButton5, javax.swing.GroupLayout.PREFERRED_SIZE, 8,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(18, 18, 18)
                        .addComponent(jTextField4, javax.swing.GroupLayout.PREFERRED_SIZE, 172,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
                .addGroup(jPanel2Layout.createSequentialGroup()
                    .addGap(37, 37, 37)
                    .addComponent(jButton4)
                    .addGap(102, 102, 102)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(jPanel2Layout.createSequentialGroup()
                            .addComponent(jLabel3)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                            .addComponent(jLabel2))
                        .addGroup(jPanel2Layout.createSequentialGroup()
                            .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE, 56,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                            .addComponent(jLabel4)
                            .addGap(40, 40, 40)
                            .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE, 57,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel2Layout.createSequentialGroup()
                                .addGap(40, 40, 40)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                                    .addComponent(jButton2, 0, 0, Short.MAX_VALUE)
                                    .addComponent(jButton1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
                            .addComponent(jLabel5))))))
                .addContainerGap(96, Short.MAX_VALUE))
        );
        jPanel2Layout.setVerticalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, jPanel2Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jButton3)
                    .addComponent(jButton5)
                    .addGroup(jPanel2Layout.createSequentialGroup()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                        .addComponent(jLabel3)
                        .addComponent(jLabel2))
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                        .addGroup(jPanel2Layout.createSequentialGroup()
                            .addGap(36, 36, 36)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                .addGroup(jPanel2Layout.createSequentialGroup()
                                    .addComponent(jButton1)
                                    .addGap(24, 24, 24)
                                    .addComponent(jButton2))
                                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
```

```
                                    .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
31, javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(jLabel5))
                                .addGap(61, 61, 61)))
                            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup()

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                                    .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE, 29,
javax.swing.GroupLayout.PREFERRED_SIZE)
                                    .addComponent(jLabel4))
                                .addGap(61, 61, 61))))
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addGap(41, 41, 41)
                        .addComponent(jButton4)))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 38,
Short.MAX_VALUE)
                .addComponent(jTextField4, javax.swing.GroupLayout.PREFERRED_SIZE, 53,
javax.swing.GroupLayout.PREFERRED_SIZE)))
            .addContainerGap())
    );

    javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
    jPanel3.setLayout(jPanel3Layout);
    jPanel3Layout.setHorizontalGroup(
        jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 74, Short.MAX_VALUE)
    );
    jPanel3Layout.setVerticalGroup(
        jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGap(0, 476, Short.MAX_VALUE)
    );

    jPanel4.setBorder(javax.swing.BorderFactory.createEtchedBorder());

    jButtonZerarX.setText("Acelera");
    jButtonZerarX.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButtonZerarXActionPerformed(evt);
        }
    });

    jButtonZerarZ.setText("Diminui");
    jButtonZerarZ.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButtonZerarZActionPerformed(evt);
        }
    });

    jLabel9.setText("Velocidade RPM");

    jTextField1.setFont(new java.awt.Font("Tahoma", 1, 24));
    jTextField1.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jTextField1ActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout jPanel4Layout = new javax.swing.GroupLayout(jPanel4);
    jPanel4.setLayout(jPanel4Layout);
```

```
        jPanel4Layout.setHorizontalGroup(
            jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel4Layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
false)
                    .addComponent(jTextField1, javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel9, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addGap(26, 26, 26)
                .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jButtonZerarZ, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 100, Short.MAX_VALUE)
                    .addComponent(jButtonZerarX, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 100, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(161, Short.MAX_VALUE))
        );
        jPanel4Layout.setVerticalGroup(
            jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel4Layout.createSequentialGroup()
                .addGroup(jPanel4Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel4Layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jLabel9))
                    .addGroup(jPanel4Layout.createSequentialGroup()
                        .addGap(19, 19, 19)
                        .addComponent(jButtonZerarX, javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGap(7, 7, 7)
                .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE, 31,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jButtonZerarZ, javax.swing.GroupLayout.PREFERRED_SIZE, 38,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(72, 72, 72))
        );

        jButtonAbrir.setBackground(new java.awt.Color(255, 51, 51));
        jButtonAbrir.setFont(new java.awt.Font("Tahoma", 0, 24));
        jButtonAbrir.setText("Stop");
        jButtonAbrir.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonAbrirActionPerformed(evt);
            }
        });

        jButtonIniciarAutomatico.setBackground(new java.awt.Color(51, 255, 51));
        jButtonIniciarAutomatico.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonIniciarAutomaticoActionPerformed(evt);
            }
        });

        jLabel6.setText("MODO AUTOMÁTICO");

        jButton6.setBackground(new java.awt.Color(102, 255, 51));
        jButton6.setFont(new java.awt.Font("Tahoma", 0, 16));
        jButton6.setText("Automatico");
```

```
        jButton6.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton6ActionPerformed(evt);
            }
        });

        jButton7.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        jButton7.setForeground(new java.awt.Color(51, 51, 255));
        jButton7.setText("MÁQUINA");

        jButton8.setFont(new java.awt.Font("Tahoma", 0, 14)); // NOI18N
        jButton8.setForeground(new java.awt.Color(51, 51, 255));
        jButton8.setText("MOINHO");

        jButton9.setBackground(new java.awt.Color(51, 255, 51));
        jButton9.setFont(new java.awt.Font("Arial Black", 0, 14)); // NOI18N
        jButton9.setForeground(new java.awt.Color(51, 51, 51));
        jButton9.setText("LIGA BOMBA");

        jButton10.setBackground(new java.awt.Color(255, 51, 51));
        jButton10.setFont(new java.awt.Font("Arial Black", 0, 14)); // NOI18N
        jButton10.setText("DESLIGA BOMBA");

        jTextField5.setFont(new java.awt.Font("Arial Black", 0, 24)); // NOI18N
        jTextField5.setText("MOINHO");

        jLabel8.setText("                BOMBEAMENTO DE TINTA");
        setJMenuBar(jMenuBar1);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(1446, 1446, 1446)
                .addComponent(jPanel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(24, 24, 24)
                        .addComponent(jPanel2, 0, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(1, 1, 1)
                        .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(18, 18, 18)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(layout.createSequentialGroup()
                                .addComponent(jPanel4, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                                .addGap(505, 505, 505))
                            .addGroup(layout.createSequentialGroup()
                                .addGap(4, 4, 4)
                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                                    .addComponent(jButtonIniciarAutomatico,
javax.swing.GroupLayout.PREFERRED_SIZE, 8, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
                        .addGroup(layout.createSequentialGroup()
                            .addComponent(jButtonAbrir, javax.swing.GroupLayout.PREFERRED_SIZE, 129,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGap(30, 30, 30)
                            .addComponent(jButton6, javax.swing.GroupLayout.PREFERRED_SIZE, 122,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                        .addGap(580, 580, 580))))
                .addGroup(layout.createSequentialGroup()
                    .addGap(95, 95, 95)
                    .addComponent(jLabel6)
                    .addContainerGap())))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
                .addGap(134, 134, 134)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                    .addComponent(jLabel8, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 276, Short.MAX_VALUE)
                    .addGroup(layout.createSequentialGroup()
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jButton8, javax.swing.GroupLayout.DEFAULT_SIZE, 93,
Short.MAX_VALUE)
                            .addComponent(jButton7, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                        .addGap(18, 18, 18)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                            .addComponent(jButton10, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                            .addComponent(jButton9, javax.swing.GroupLayout.DEFAULT_SIZE, 165,
Short.MAX_VALUE))))
                .addGap(51, 51, 51)
                .addComponent(jTextField5, javax.swing.GroupLayout.PREFERRED_SIZE, 141,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(918, 918, 918))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addComponent(jPanel1, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                    .addComponent(jPanel4, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(34, 34, 34)
                        .addComponent(jLabel6)
                        .addGap(35, 35, 35)
                        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jButtonAbrir, javax.swing.GroupLayout.PREFERRED_SIZE, 195,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jButton6, javax.swing.GroupLayout.PREFERRED_SIZE, 190,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(jButtonIniciarAutomatico))
                    .addGroup(layout.createSequentialGroup()
                        .addGap(13, 13, 13)
                        .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(36, 36, 36)
```

```
                .addComponent(jLabel8, javax.swing.GroupLayout.PREFERRED_SIZE, 21,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                  .addComponent(jTextField5, javax.swing.GroupLayout.DEFAULT_SIZE, 57,
Short.MAX_VALUE)
                  .addComponent(jButton9, javax.swing.GroupLayout.DEFAULT_SIZE, 57,
Short.MAX_VALUE)
                  .addComponent(jButton7, javax.swing.GroupLayout.DEFAULT_SIZE, 57,
Short.MAX_VALUE))
                .addGap(13, 13, 13)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                  .addGroup(layout.createSequentialGroup()
                    .addGap(221, 221, 221)
                    .addComponent(jPanel3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                  .addGroup(layout.createSequentialGroup()
                    .addGap(47, 47, 47)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                      .addComponent(jButton8, javax.swing.GroupLayout.PREFERRED_SIZE, 58,
javax.swing.GroupLayout.PREFERRED_SIZE)
                      .addComponent(jButton10, javax.swing.GroupLayout.PREFERRED_SIZE, 61,
javax.swing.GroupLayout.PREFERRED_SIZE))))))
            .addContainerGap())
      );

      pack();
   }// </editor-fold>

   private void jButtonXposActionPerformed(java.awt.event.ActionEvent evt) {
     int incr=1;
      try {
        controller.horario(incr);
      } catch (Exception ex) {
      }


   }
/**/ // horário
   private void jButtonZposActionPerformed(java.awt.event.ActionEvent evt) {
      int incr = 1;
      try {

        controller.habilita(incr);
        jTextField1.setText(""+rpm);
        String temp = Integer.toString(temperatura);
        jTextField2.setText(""+temperatura);
        jTextField4.setText("MANUAL");
    //    controller.leitura_termometro(incr);
        jTextField3.setText("25");

      } catch (Exception ex) {
      }
  //    Habilita

   }
      // habilita
   private void jButtonXnegActionPerformed(java.awt.event.ActionEvent evt) {
      int incr = 1;
```

```
        try {
          controller.antihorario(incr);
        } catch (Exception ex) {
        }
//      incrementaSubtraiX(incr, modo);

    }
          // anti-horário
    private void jButtonZerarXActionPerformed(java.awt.event.ActionEvent evt) {
         velocidade = velocidade +1;

        try {
           controller.acelera(velocidade);
           jTextField1.setText(""+rpm);
        } catch (Exception ex) {

        }

    }
        // aumenta velocidade;
    private void jButtonZnegMouseClicked(java.awt.event.MouseEvent evt) {
        int incr = 1;
        velocidade = 0;
        try {
          controller.desabilita(incr);
          jTextField1.setText("0");

        } catch (Exception ex) {
        }


    }
        // desabilita
    private void jButtonAbrirActionPerformed(java.awt.event.ActionEvent evt) {

        int incr = 1;
        velocidade = 0;
        try {
          automatico=0;
          controller.desabilita(incr);
          jTextField1.setText("0");
          jTextField4.setText("EMERGÊNCIA");
          jButtonXpos.setEnabled(true);
          jButtonZpos.setEnabled(true);
          jButtonXneg.setEnabled(true);
          jButtonZerarX.setEnabled(true);
          jButtonZneg.setEnabled(true);
          jButtonZerarZ.setEnabled(true);

        } catch (Exception ex) {
        }

    }

    private void jButtonZerarZActionPerformed(java.awt.event.ActionEvent evt) {

        velocidade = velocidade -1;
        try {
           controller.acelera(velocidade);
           jTextField1.setText(""+rpm);
```

```
    } catch (Exception ex) {

    }


}
       // desacelera
private void jButtonIniciarAutomaticoActionPerformed(java.awt.event.ActionEvent evt) {

//   jButtonXpos.setEnabled(false);
//   jButtonZpos.setEnabled(false);
//   jButtonXneg.setEnabled(false);
//   jButtonZerarX.setEnabled(false);
//   jButtonZneg.setEnabled(false);
//   jButtonZerarZ.setEnabled(false);
//   jTextField4.setText("AUTOMÁTICO");

//   automatico=1;
   try {


       if (automatico==1)   {
      // algoritmo de controle do sistema, deveria ser implementado na
      // controller, porém obtive problemas para chamar a função.
         jButton4.doClick();
         Thread.sleep(10);
         jTextField3.setText(""+temp_sensor);
         Thread.sleep(10);
          // aguarda a cte de tempo do sistema, de 3 segundos
       if (temperatura < temp_sensor1)  {
          jTextField3.setText(""+temp_sensor);
          velocidade = velocidade +1;

          controller.acelera(velocidade);
          jTextField1.setText(""+rpm);
       }
       else

       if (temperatura > temp_sensor1)  {


          if (velocidade <= 5)   {
           jTextField3.setText(""+temp_sensor);
           controller.acelera(velocidade);
           jTextField1.setText(""+rpm);
          }
        // ate a velocidade de 250 rpm mantém a velocidade da maquina, por
        // questoes de continuidade no papel.
          else {
          velocidade = velocidade -1;
          jTextField3.setText(""+temp_sensor);
          controller.acelera(velocidade);
          jTextField1.setText(""+rpm);
          }
       }
       else

       if (temperatura == temp_sensor1) {
          jTextField3.setText(""+temp_sensor);
          controller.acelera(velocidade);
```

```
            jTextField1.setText(""+rpm);
          }

          }

      } catch (Exception ex) {

      }
   }

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:

   temperatura=temperatura+1;

   try {
        String temp = Integer.toString(temperatura);
         jTextField2.setText(""+temp);
      } catch (Exception ex) {
}
}
private void jButtonZnegActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
}

private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {

   // TODO add your handling code here:
//   jTextField1.setText("Jeferson");
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:

   temperatura = temperatura-1;
   try {
        String temp = Integer.toString(temperatura);
        jTextField2.setText(""+temperatura);
      } catch (Exception ex){
      }

}

private void jTextField3ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
}

private void jTextField2ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:


}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
   int incr=1;
   try {
    // jButton4.doClick();
   //controller.trocaporta1(incr);
  //  Thread.sleep(100);
```

```
    controller.leitura_termometro(incr);
    jTextField3.setText(""+temp_sensor);
    temp_sensor1=Integer.parseInt(temp_sensor, 10);
    Thread.sleep(30);
 //   controller.trocaporta2(incr);
   // jButton5.doClick();
    } catch (Exception ex){

    }
}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
    int incr=1;
    try {

        controller.trocaporta1(incr);


         Thread.sleep(10);
         jButton3.doClick();
         Thread.sleep(10);
         jButton3.doClick();
         Thread.sleep(10);
         jButton5.doClick();
//         controller.leitura_termometro(incr);
//         jTextField3.setText(""+temp_sensor);
     //     if(automatico==1)  {
     //     Thread.sleep(100);
     //     jButtonIniciarAutomatico.doClick();
     //     }
    } catch (Exception ex){
    }

}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
    int incr=1;
    try {

        controller.trocaporta2(incr);

    } catch (Exception ex){

    }
}

private void jButton3PropertyChange(java.beans.PropertyChangeEvent evt) {
// TODO add your handling code here:
}

private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:

    try {

        automatico=1;
        jTextField4.setText("AUTOMÁTICO");
        jButtonXpos.setEnabled(false);
```

```
        jButtonZpos.setEnabled(false);
        jButtonXneg.setEnabled(false);
        jButtonZerarX.setEnabled(false);
        jButtonZneg.setEnabled(false);
        jButtonZerarZ.setEnabled(false);

    } catch (Exception ex){

    }


}

    /**
    * @param args the command line arguments
    */


    // Variables declaration - do not modify
    private javax.swing.ButtonGroup buttonGroup1;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton10;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JButton jButton5;
    private javax.swing.JButton jButton6;
    private javax.swing.JButton jButton7;
    private javax.swing.JButton jButton8;
    private javax.swing.JButton jButton9;
    private javax.swing.JButton jButtonAbrir;
    private javax.swing.JButton jButtonIniciarAutomatico;
    private javax.swing.JButton jButtonXneg;
    private javax.swing.JButton jButtonXpos;
    private javax.swing.JButton jButtonZerarX;
    private javax.swing.JButton jButtonZerarZ;
    private javax.swing.JButton jButtonZneg;
    private javax.swing.JButton jButtonZpos;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    private javax.swing.JLabel jLabel9;
    private javax.swing.JMenuBar jMenuBar1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    private javax.swing.JPanel jPanel4;
    private javax.swing.JTextField jTextField1;
    private javax.swing.JTextField jTextField2;
    private javax.swing.JTextField jTextField3;
    private javax.swing.JTextField jTextField4;
    private javax.swing.JTextField jTextField5;
    // End of variables declaration

}
```