UNIVERSIDADE FEDERAL DA BAHIA

ESCOLA POLITÉCNICA

PROGRAMA DE PÓS-GRADUAÇÃO EM MECATRÔNICA

LUCAS RAMALHO OLIVEIRA

Validação do Protótipo BIOX10 para a Análise Oxidativa de Biodiesel e Óleos Vegetais.

Salvador

2014

LUCAS RAMALHO OLIVEIRA

Validação do Protótipo BIOX10 para a Análise Oxidativa de Biodiesel e Óleos Vegetais.

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Mecatrônica, Escola Politécnica, Universidade Federal da Bahia, como requisito para obtenção do grau de Mestre em Mecatrônica.

Orientador: Prof. Dr. Iuri Muniz Pepe

Co-orientador: MSc. Luiz Carlos Simões Soares Junior

Salvador

2014

LUCAS RAMALHO OLIVEIRA

Validação do Protótipo BIOX10 para a Análise Oxidativa de Biodiesel e Óleos Vegetais.

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Mecatrônica, Escola Politécnica, Universidade Federal da Bahia, como requisito para obtenção do grau de Mestre em Mecatrônica.

Banca Examinadora

Prof. Dr. Iuri Muniz Pepe – Orientador _____

Doutor em Física Nuclear, Universite Catholique de Louvain, U.C.L., Bélgica
Universidade Federal da Bahia.

Prof. Dr. Antônio Cezar de Castro Lima _____

Doutor em Engenharia Eletrônica, University of Kent, U. Kent, Inglaterra
Universidade Federal da Bahia.

Prof. Dr. Daniel Maurice Grosjean _____

Doutor em Sciences, Chimie Organique Physique, França
Université Paris Diderot.

Dedicado a Deus, família e amigos

AGRADECIMENTOS

Agradeço a Deus por ter me proporcionado a chance de dar mais um passo adiante em minha carreira profissional, através do mestrado em Mecatrônica e ter direcionado as pessoas certas para me ajudar a vencer os desafios ao longo deste período. Agradeço a meu pai, minha mãe, irmã e minha tia Zulmira pelo apoio.

Agradeço ao meu orientador, Dr. Iuri M. Pepe, por ter aberto as portas do LaPO para mim, ter me instruído, disponibilizado todas as condições e recursos necessários para a realização do meu trabalho.

Ao meu co-orientador Luiz Carlos S. Soares Junior, que foi o construtor e projetista do BIOX10. E me forneceu valiosas informações para o meu trabalho.

Ao Msc. Angelo Oliveira dos Santos e Geydison Demetino, pelas muitas dicas de programação que foram essenciais para o desenvolvimento do software do BIOX10.

A Dr. Daniel M. Grosjean e Dr. Vitor F. Pinheiro, pelas informações valiosas e por terem fornecido óleos e biodiesel para serem analisados no BIOX10.

A ajuda de Anaildes, Érica, Leo e Carol, com o fornecimento de soluções de calibração, biodiesel, óleos e dicas na área de química.

Aos meus amigos do LaPO e aos membros da banca.

"Preparas uma mesa perante mim na presença dos meus inimigos, unges a minha cabeça com óleo, o meu cálice transborda."

Salmo 23:5

RESUMO

O Brasil tem um grande potencial para a produção de biodisel devido à diversidade de oleagenosas encontradas em seu vasto território. Por causa desta diversidade de matérias-primas o biodiesel produzido terá propriedades diferentes e a depender de suas condições de armazenamento, clima e transporte pode haver uma variação em sua qualidade. Com a finalidade de ter uma tecnologia nacional para atestar a qualidade do biodiesel através do seu estado de oxidação, foi desenvolvido no LaPO (Laboratório de Propriedades Óticas), situado no Instituto de Física da Universidade Federal da Bahia, um equipamento chamado de Biox10, que tem a finalidade de fazer a análise oxidativa, utilizando o método Rancimat. Neste trabalho foi possível mostrar o funcionamento do software e hardware Biox10, foram mostradas também as calibrações dos sistemas de instrumentação, térmico e de ar. Bem como a validação do equipamento como um todo, comparando o tempo de prateleira do biodiesel de matriz de sebo bovino, medido com o protótipo e o tempo de prateleira obtido com o Rancimat 873. Foi aplicada a metodologia ASLT (Accelerated Shelf-Life Testing) nas temperaturas de 80 a 120 °C. Sendo assim, foi considerado o Biox10 validado já que o tempo de prateleira determinado por dois métodos de análise pelo Rancimat 873 foi, respectivamente, 2684 e 2467 horas com erro de 4,1%, enquanto que o protótipo mediu 2345 com incerteza de 7,9%.

Palavras-Chave: Biodiesel, Estabilidade Oxidativa, Rancimat, Biox10.

ABSTRACT

Brazil has a great potential to produce biodiesel due to the diversity of oilseed found in its vast territory. The biodiesel produced in Brazil has different properties because of the diversity of raw material. Depending on the storage conditions, climate and transportation the biodiesel quality may change. With the purpose of having a national technology to certify the quality of biodiesel through its oxidation state, it has been developed in Lapo (Laboratory of Optical Properties) located in the Physics Institute from Universidade Federal da Bahia an prototype called Biox10 that is intended to oxidative analysis using the Rancimat method. This paper has shown how Biox10´s software and hardware works and also the calibrations of the instrumentation system, the heat system and the air system. And the validation of the equipment as a whole, comparing the shelf life of tallow biodiesel, measured with the prototype and shelf life obtained with the Rancimat 873. The ASLT (Accelerated Shelf-Life Testing) method was applied at temperatures from 80 to 120 °C. Therefore the Biox10 was considered validated since the shelf life determined by two methods of analysis by Rancimat 873 was, respectively, 2684 and 2467 hours with an error of 4.1%, while in 2345 the prototype measured with uncertainty of 7, 9%.

Keyword: Biodiesel, Oxidative Stability Index, Rancimat, Biox10.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADC	Conversor Analógico Digital
ASLT	Accelerated Shelf-Life Testing
ANP	Agência Nacional de Petróleo, Gás Natural e Biocombustíveis
CEI	Comissão Executiva Interministerial
GG	Grupo Gestor
IP	Induction Period
LaPO	Laboratório de Propriedades Óticas
PNPB	Programa Nacional de Produção e uso do Biodiesel

SUMÁRIO

1 INTRODUÇÃO

Bio-combustível é um tema que, atualmente, desperta o interesse da comunidade científica por ser uma fonte de energia alternativa e renovável. Com o objetivo de obter uma melhor eficiência destas fontes energéticas, muito trabalho vem sendo desenvolvido.

O biodiesel é um bio-combustível promissor, pois o Brasil é dotado de um vasto território e um clima favorável para o plantio de oleaginosas que ainda são o principal insumo para a produção deste biocombustível. Este potencial de produção de matéria-prima permite que haja um favorecimento do desenvolvimento e comercialização de um biodiesel de qualidade, além do desenvolvimento de tecnologias relacionadas com este biocombustível.

Com a finalidade de desenvolver uma tecnologia de avaliação da qualidade de óleos e biodiesel o Laboratório de Propriedades Óticas do Instituto de Física da Universidade Federal da Bahia, desenvolveu um protótipo chamado de BIOX10, capaz de determinar o estado oxidativo de óleos, gorduras e biodiesel. Este equipamento utiliza o método Rancimat, que consiste em borbulhar ar, com fluxo controlado, em uma amostra de óleo, de massa conhecida, aquecida a uma temperatura, também controlada, com a finalidade de determinar o período para a oxidação da amostra ensaiada. Para tanto, a variação da condutividade de uma porção de água deionizada, usada como armadilha para os voláteis transportados pelo sistema de ar do equipamento, é monitorada em função do tempo, de forma a evidenciar quando a indução da oxidação da amostra de óleo ocorre (Jain e Sharma, 2010).

Neste trabalho será mostrado o funcionamento do BIOX10, a calibração dos seus diferentes sistemas e a validação do mesmo, fazendo um comparativo entre as medidas obtidas no protótipo e as medidas feitas em um equipamento comercial (Rancimat 873 da Metrohm), para amostras submetidas às mesmas condições de temperatura, vazão e massa.

1.1 DELIMITAÇÃO DO TEMA

Neste ponto é importante frisar que uma primeira versão do protótipo de engenharia BIOX10, era chamada de Ranciquimis, quando este trabalho de pesquisa foi iniciado em fevereiro de 2012 no LaPO. Este protótipo contava com os principais sistemas e subsistema para a medição automatizada da oxidação, mas, por se tratar de um sistema mecatrônico de análise, envolvendo hardware mecânico, eletrônico, elétrico, sensores e sistema para medição de parâmetros físicos de interesse e software, porém, ainda haviam certas partes e pendências técnicas a serem acertadas e desenvolvidas. Assim o tema e objetivo dessa dissertação é validar o protótipo de equipamento BIOX10 enquanto sistemas de medição e análise da estabilidade oxidativa de biodiesel, além da criação de uma solução de software para controle e aquisição de dados com este equipamento.

1.2 PROBLEMAS DE PESQUISA

O BIOX10 é dotado de três sistemas que funcionam de forma integrada, o sistema térmico, a instrumentação e o sistema de controle do fluxo de ar.

Estes sistemas necessitavam de algumas intervenções mecânicas, calibrações e de um software para controle e aquisição de dados mais completo e possibilitando uma boa interface com o usuário, pois, sem essas intervenções não seria possível por o protótipo à prova e validar os sistemas que compõem o BIOX10.

Portanto era importante descobrir como solucionar os entraves do protótipo de engenharia, de forma a torná-lo mais próximo de um protótipo de teste de produção, etapa indispensável para chegar-se a um produto comercial.

1.3 HIPÓTESE

O Biox10 é um protótipo que foi desenvolvido com a finalidade de fazer a análise oxidativa em óleos vegetais e biodiesel, utilizando o método Rancimat de acordo com a norma européia EN14112.

No início do projeto Biox10, este protótipo era composto apenas do hardware, ou seja; o sistema de ar, térmico e instrumentação. Na execusão deste projeto, foi desenvolvido um software para controle e aquisição de dados, para ser possível fazer a análise da estabilidade oxidativa e as suas devidas calibrações, atendendo as expecificações da norma EN14112 e assim validar os sistemas isoladamente. A hipótese defendida é que seria possível validar o Biox10 em sua totalidade, buscando obter resultados próximos a partir da comparação das medidas feitas no protótipo e em um equipamento comercial quando submetido às mesmas condições, uma vez que ambos os equipamentos foram projetados para atender a norma EN14112.

1.4 OBJETIVOS

1.4.1 Objetivo Geral

Tornar o protótipo BIOX10 mais próximo de um equipamento comercial.

1.4.1.1Objetivos Específicos

a) Estudar o método Rancimat de análise da estabilidade oxidativa do biodiesel;
b) Estudar o equipamento BIOX10 e identificar todas as restrições associados à medição dos diferentes parâmetros de interesse;
c) Atuar na correção das restrições identificadas no Biox10 já existentes;
d) Calibrar os sistemas que compõem o equipamento e validar o protótipo desenvolvido.

1.5 JUSTIFICATIVA

O uso de biodiesel como combustível tem se tornado um atrativo, pois é uma fonte de energia renovável e traz benefícios ao meio ambiente, porém o biodiesel geralmente apresenta alterações em suas características ao logo da estocagem, o principal agente desta mudança é a oxidação. Desta maneira é de fundamental importância a determinação da estabilidade oxidativa para se poder determinar a qualidade do biodiesel.

Dentre os vários métodos de medição da oxidação de óleos vegetais e biodiesel, o método estudado neste trabalho é o Rancimat, que consiste em aquecer e injetar ar em uma amostra de óleo, para que haja a oxidação forçada da amostra. Durante o processo de oxidação da amostra, há uma reação química que resulta, principalmente, na formação de ácido fórmico e outros compostos voláteis de oxidação, que são transportados e dissolvidos em uma solução de água deionizada, onde é feito o monitoramento do estado de oxidação do biodiesel, pela medida da variação da condutividade da água (SOARES JUNIOR et al., 2012).

Portanto ter um equipamento nacional que possibilite fazer análise oxidativa em óleos vegetais e biodiesel é de muita relevância, pois se espera com o aprimoramento do Biox10 torná – lo um produto competitivo no mercado, gerando emprego e renda, além de gerar patentes e publicações para a comunidade acadêmica.

1.6 METODOLOGIA

Este trabalho pode ser dividido nas seguintes etapas:

a) Etapa 1. Estudos aprofundados na literatura sobre o método Rancimat;
b) Etapa 2. Análise aprofundada do estado de funcionamento do protótipo de equipamento baseado na metodologia RANCIMAT, já existente no LaPO;
c) Etapa 3. Estudo para identificar as restrições relacionadas à medição da estabilidade oxidativa com o equipamento BIOX10;
d) Etapa 4: Correções e solução das diversas restrições relacionadas às medições;
e) Etapa 5: Implementação de melhorias feitas no equipamento estudado.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 BIODIESEL

Nas últimas décadas, houve um forte apelo mundial no sentido do desenvolvimento sustentável. Isso se tornou mais evidente a partir de 1997 com a assinatura, por diversos países, do Protocolo de Kyoto. Neste acordo, diferentes países se comprometeram em reduzir a emissão de gazes poluentes causadores do efeito estufa. Neste contexto o Brasil decidiu na época adotar políticas energéticas limpas e renováveis, para contribuir com a redução dos impactos ambientais causados pelo seu desenvolvimento (SEBRAE, 2007).

Nos anos 70 e 80 do século passado a crise do petróleo, agravada pelos conflitos no oriente médio, resultou em um aumento significativo no preço do barril de petróleo. Impulsionado pela crise do petróleo o governo brasileiro decidiu investir em energias renováveis e foi criado, em 14 de novembro 1975, o Programa Nacional do Álcool – Proálcool. Este programa tinha como objetivo reduzir a dependência brasileira da gasolina importada, substituindo parte da frota nacional de veículos por automóveis movidos a álcool, além de misturar um percentual de álcool na gasolina, tornado menos poluente a sua combustão. Nos anos 80 foi criado o Programa Nacional de Óleos Vegetais para fins energéticos, este programa tinha como objetivo substituir cerca de 30% do óleo diesel por biocombustível produzido a partir de óleo de soja, canola, amendoim e girassol. Em 2003, o governo criou a Comissão Executiva Interministerial (CEI) e o Grupo Gestor (GG) responsáveis por implantar as ações da produção e uso do biodiesel. Essas duas entidades elaboraram em 2004 o Programa Nacional de Produção e uso do Biodiesel (PNPB) que tinha como principais diretrizes (SEBRAE, 2007):

a) Implantar um programa sustentável, promovendo inclusão social;
b) Garantir preços competitivos, qualidade e suprimento;

c) Produzir biodiesel de diferentes fontes oleaginosas em diferentes regiões.

A Lei 11.097, publicada em 13 de Janeiro de 2005, introduziu o biodiesel na matriz energética brasileira e ampliou as atribuições da ANP (Agência Nacional de Petróleo, Gás Natural e Biocombustíveis). Esta entidade passou a estabelecer normas regulatórias, autorizar e fiscalizar as atividades relacionadas à produção, transporte, transferência, armazenagem, estocagem, importação, exportação, distribuição, revenda e comercialização e avaliação de conformidade e certificação de biocombustíveis (ANP, 2012).

No mercado de biodiesel adotou-se uma nomenclatura para designar o biodiesel misturado no diesel mineral, em que BXX significava a porcentagem de biodiesel na mistura. Ou seja, quando se diz que o tipo do biodiesel é B5, isso significa que são adicionados 5% de biodiesel ao diesel mineral (SEBRAE, 2007).

O Brasil por ser um país que tem grande extensão territorial de clima variado, tem um grande potencial para a produção de biodiesel, pois favorece o cultivo de certa variedade de oleaginosas, mesmo as que não se destinam a alimentação humana, e ainda assim podem servir de matéria-prima para a produção deste biocombustível. Porém as oleaginosas mais comuns no Brasil usadas na produção de biodiesel são: soja, mamona, amendoim, girassol, algodão, dendê, pinhão manso. Além da gordura animal (sebo) que também serve como matéria-prima. Para ilustrar o potencial de distribuição agrícola das oleaginosas no território brasileiro, foi feito um mapa com base no clima e agricultura regional. Veja na Figura 1.

Figura 1. Potencial de distribuição agrícola das oleaginosas no Brasil.
Fonte: (SEBRAE, 2007).

2.1.1 Óleos e Gorduras

Os óleos e gorduras são substâncias insolúveis em água (hidrofóbica), que podem ser de origem vegetal ou animal, também são conhecidos como triacilgliceróis. Quando estão com consistência pastosa estes são vulgarmente chamados de gordura e quando estão na forma líquida são denominados óleos. Quando os óleos são oriundos de frutos, são denominados de azeites (REDA e CARNEIRO, 2007).

As substâncias presentes nos óleos podem ser divididas em dois grupos: a) glicerídeos e b) não- glicerídeos.

a) Glicerídeos: são denominados produtos da esterificação de uma molécula de glicerol com três de ácidos graxos (REDA e CARNEIRO, 2007);

b) Não-Glicerídeos: são componentes não glicerídeos encontrados no óleo, sua quantidade é da ordem de 5% em óleos no estado bruto, quando refinado ficam abaixo de 2%. Estas substâncias presentes nos óleos são responsáveis por algumas características do óleo como o sabor, coloração e ação oxidante (REDA e CARNEIRO, 2007).

Os ácidos graxos são compostos de triglicerídeos, que possuem em sua maioria 12, 14, 16 e 18 átomos de carbono e podem ser representados conforme mostrado na Figura 2. A Tabela 1 demonstra a distribuição dos ácidos graxos em alguns tipos de óleos e gorduras.

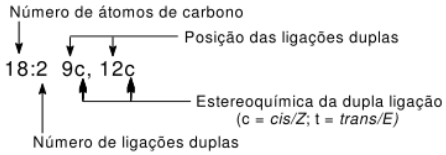


Figura 2. Representação dos ácidos graxos.
Fonte: (GARCIA; PROF; SCHUCHARDT, 2006).

Tabela 1. Distribuição dos ácidos graxos.

Óleo ou Gordura	Composição em ácidos graxos (% em massa)							Outros ácidos graxos (%)
	12:0	14:0	16:0	18:0	18:1	18:2	18:3	
babaçu	44 – 45	15 – 16,5	5,8 – 8,5	2,5–5,5	12 – 16	1,4 – 2,8	—	8:0 (4,1 – 4,8); 10:0 (6,6 – 7,8)
mamona	—	—	0,8-1,1	0,7-1,0	2,0 – 3,3	4,1 – 4,7	0,5 – 0,7	18:1-OH* (89); 20:1 (0,5), 18:0-2OH★ (0,6 – 1,1)
coco	44 – 51	13 – 18,5	7,5 – 11	1-3	5 – 8,2	1,0 – 2,6	—	8:0 (7,8 – 9,5); 10:0 (4,5 – 9,7)
milho	—	—	7	3	43	39	—	—
algodão	—	1,5	22	5	19	50	—	—
linhaça	—	-	6	4	13 – 37	5 – 23	26 – 58	—
oliva	—	1,3	7 – 16	1,4 – 3,3	64 – 84	4 – 15	—	—
dendê	—	0,6 – 2,4	32 – 45	4,0 – 6,3	38 – 53	6 – 12	—	8:0 (2,7 – 4,3); 10:0 (3 – 7)
amendoim	—	0,5	6 – 11,4	3 – 6	42,3 – 61	13 – 33,5	—	20:0 (1,5); 22:0 (3 – 3,5)
colza	—	1,5	1 – 4,7	1,0 – 3,5	13 – 38	9,5 – 22	1– 10	22:1 (40 – 60)
soja	—	—	2,3 – 11	2,4 – 6	23,5 – 31	49 – 51,5	2 – 10,5	—
girassol	—	—	3,6 – 6,5	1,3 – 3	14 – 43	44 – 68	—	—
sebo	—	3 – 6	25 – 37	14 – 29	26 – 50	1 – 2,5	—	—

*Ácido Ricinoléico: $CH_3(CH_2)_5CHOHCH_2CHCH(CH_2)_6CH_2COOH$
★Ácido Dihidroxiesteárico

Fonte: (GARCIA; PROF; SCHUCHARDT, 2006).

2.2 BIO COMBUSTÍVEIS

Os óleos vegetais foram utilizados pela primeira vez em motores a diesel, no início do século XIX, por Rudolf Diesel, o inventor do motor de mesmo nome, atendendo a uma solicitação do governo Francês, que tinha o intuito de promover a auto-suficiência energética em suas colônias africanas, minimizando o custo relativo às importações de combustíveis líquidos e carvão mineral.

Porém, a utilização de óleo vegetal em motores Diesel, pode apresentar uma série de problemas, já que a viscosidade do óleo vegetal é bem maior do que a do diesel mineral e pelo fato de sua volatilidade ser mais baixa. Assim, podem aparecer alguns problemas como: combustão incompleta e formação de fuligem no motor. Com a finalidade de compensar estas limitações são utilizados alguns métodos como: Pirólise, Microemulsão e Transesterificação (SHARMA, SINGH, e UPADHYAY, 2008).

2.2.1 Pirólise

A pirólise ou craqueamento térmico é a conversão de uma substância em outra utilizando calor, na ausência de oxigênio. A temperatura deste processo pode atingir em torno de 450ºC e a depender da substância que está sendo submetida a este processo pode ser adicionado um catalizador (GARCIA, PROF, e SCHUCHARDT, 2006).

O processo de pirólise é aplicado em óleos vegetais com a finalidade de quebrar as moléculas de triacilglicerois, para obter um produto que tenha viscosidade mais baixa, um maior número de cetano e quantidade satisfatória de enxofre, entretanto, em geral, produz quantidades relativamente grandes de resíduos de carbono (SHARMA et al., 2008).

Na figura 3 demonstra em mais detalhes da quebra da molécula de triacilglicerol (TG), quando a mesma é submetida a temperaturas entre 360 e 390°C,

dando origem aos ácidos graxos (FA) e quando a temperatura de processo está entre 390°C e 420°C, ocorre a descarboxilação do (FA), ou seja; uma quebra da molécula de (FA) resultando em gás carbônico e hidrocarboneto (ITO et al., 2012).

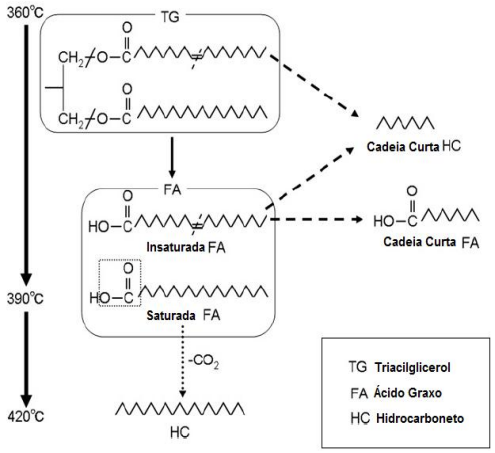


Figura 3. Reações do Processo de Pirólise.
Fonte: (ITO et al., 2012).

2.2.2 Microemulsão

As microemulsões são definidas como dispersões termodinamicamente estáveis e isotropicamente translúcidas. Estas podem ser realizadas entre líquidos imiscíveis, e são, geralmente, estabilizadas na presença de uma fina camada de tensoativo (SCIENCES, 2011).

As microemulsões são preparadas em óleos vegetais pela adição de álcoois como metanol ou etanol e um surfactante. Apesar de a microemulsão ser um dos métodos utilizados para reduzir a viscosidade do biodiesel, este processo apresenta algumas desvantagens para o motor de um veículo. São exemplos dessas desvantagens: grande quantidade de depósito de carbono, aumento da viscosidade do óleo lubrificante e combustão incompleta (GARCIA et al., 2006).

2.2.3 Transesterificação

Dentre os processos para reduzir a viscosidade o mais eficiente é a transesterificação, pois esta rota de produção resulta em um biodiesel com propriedades mais próximas as do diesel mineral.

A transesterificação é uma reação química entre um triglicerídeo e um álcool na presença de um catalisador. Este processo consiste em uma seqüência de três reações reversíveis consecutivas, em que os triglicerídeos são convertidos em diglicerídeos, que por sua vez são convertidos em monoglicerídeos e em seguida em glicerol. Para cada etapa do processo uma molécula de ester é produzida para cada molécula de triglicerídeo convertida. Na Figura 4 é apresentada a reação de transesterificação em óleos vegetais, onde se observa a reação de um triglicerídeo com um álcool, na presença de um catalisador, resultando em uma mistura de ésteres monoalquílicos de ácidos graxos e glicerol (SHARMA et al., 2008).

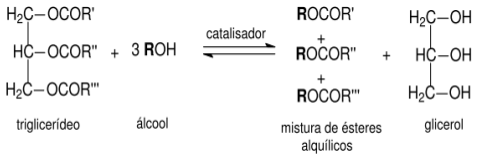


Figura 4. Reação de transesterificação.
Fonte: (GARCIA et al., 2006).

O biodiesel é produzido a partir de gorduras e óleos vegetais. Esta matéria-prima precisa ser preparada para em seguida entrar no processo de transesterificação. Durante esta etapa é adicionado o catalisador e o álcool (metílico ou etílico) ao óleo vegetal ou gordura animal. Após o término da reação de transesterificação a substância resultante é separada em duas fases, a fase pesada e a leve. Na fase pesada é feito a recuperação do álcool da glicerina, com a finalidade de obter a glicerina bruta. Em seguida esta glicerina é destilada, resultando em glicerina destilada e resíduo glicérico. Na fase leve os ésteres

resultantes da transesterificação, passam pela etapa de recuperação do álcool e em seguida são purificados, resultando assim em biodiesel, conforme o diagrama mostrado na Figura 5.

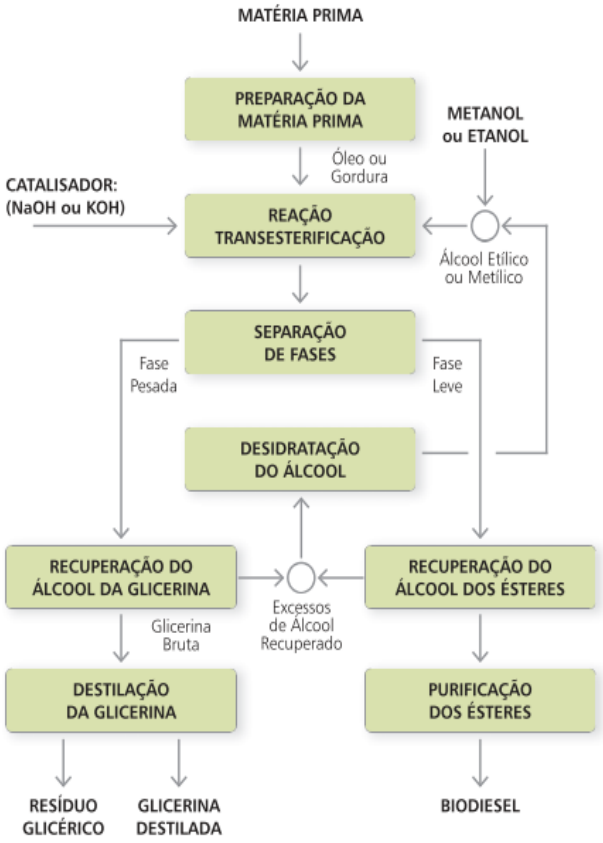


Figura 5. Diagrama do processo de transesterificação de biodiesel.

Fonte: (SEBRAE, 2007).

2.3 ESTABILIDADE OXIDATIVA

A estabilidade oxidativa é um importante fator para determinar a validade dos óleos e gorduras. Há diversos fatores que contribuem para a oxidação de óleos, dentre eles podem ser citados: o contato dos óleos com o ar do ambiente, temperatura inadequada, contaminação com bactérias e fungos, umidade, luz e contato com certos tipos de metais (DUNN, 2005).

A degradação do biodiesel ocorre principalmente de duas formas, a auto-oxidação e a foto-oxidação. A foto-oxidação ocorre quando o biodiesel é exposto à

radiação ultravioleta e a auto-oxidação ocorre quando há uma reação do radical da cadeia do ester monoalquílico de ácidos graxos, com o oxigênio da atmosfera. Esta reação resulta, inicialmente, em hidro-peróxidos e radicais livres. Em seguida, outras reações químicas acontecem e dão lugar a formação de ácidos, polímeros e outras substâncias. Quando essas substâncias resultantes da oxidação estão presentes no biodiesel, podem ser prejudiciais ao motor, pois podem ocasionar alterações em suas propriedades, como por exemplo: aumento da viscosidade, da acidez, dos níveis de peróxidos e ésteres. Tais mudanças no bicombustível podem resultar em acúmulo de polímeros nos bicos injetores do motor, prejudicando o sistema de injeção (LAPUERTA et al., 2012).

Na seção seguinte será discutido o método Rancimat, que é de fundamental importância para determinar o estado de oxidação de óleos, biodiesel e derivados.

2.3.1 Método Rancimat

O Rancimat é um método de oxidação forçada e acelerada, que consiste em transportar os voláteis oxidados de uma amostra 3 g de óleo, com fluxo de ar com vazão constante de 10 L/h. Esta amostra é aquecida em temperatura constante, que pode ser de 100, 110 ou 120 °C. Os vapores resultantes da oxidação misturados com o ar forçado são dissolvidos em 50 mL de água deionizada, onde está imerso um sensor de condutividade, conforme ilustra a Figura 6. Estes sensores, que por sua vez detectam a variação da condutividade na água deionizada, causada pela absorção do ácido carboxílico, os dados dessa medição dão lugar aos pontos que formam a curva de oxidação em função do tempo. A partir dessa curva é possível obter o período de indução (IP, *Indution Period*) da amostra. Isto é feito a partir da determinação do ponto de inflexão da curva de condutividade da água dionizada, que pode ser obtido pela interseção entre duas retas tangentes, traçadas a partir das duas extremidades desta curva, conforme ilustra a Figura 7 (JAIN; SHARMA, 2010).

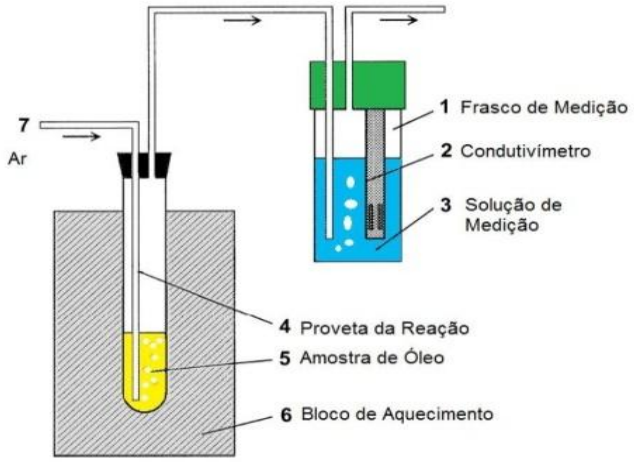


Figura 6. Diagrama do Método Rancimat.
Fonte: EN14112, 2003.

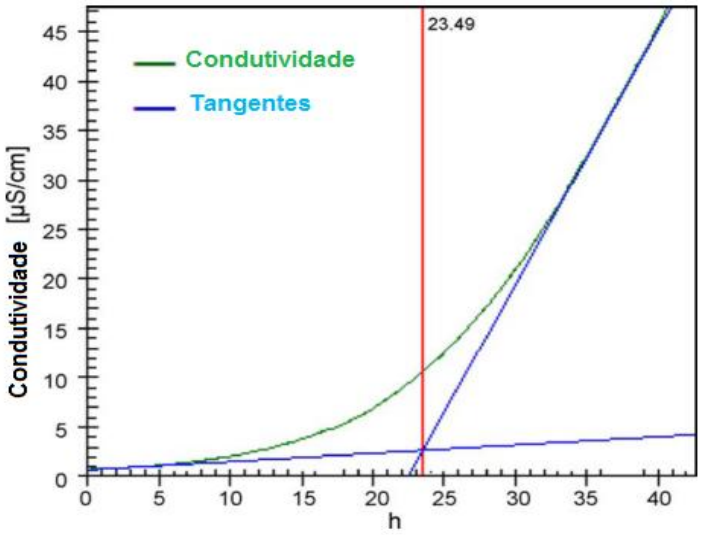


Figura 7. Gráfico da determinação do IP (período de indução) utilizando o método das tangentes.
Fonte: (JAIN; SHARMA, 2010)

A norma européia, EN14112, que regulamenta a estabilidade oxidativa em ésteres metílico de ácido graxo na temperatura de 110ºC, define o período de indução IP como um intervalo de tempo em que a medida da oxidação muda de concavidade, passando a crescer em uma taxa maior. A duração do IP determina a qualidade do biodiesel. Segundo a norma EN14112, na medição de um éster metílico não oxidado, o IP observado pode ser, em média, de 6h, para uma temperatura de oxidação à 110ºC.

3 MATERIAIS E MÉTODOS

3.1 O PROTÓTIPO BIOX10

O BIOX10 é o protótipo de engenharia de um equipamento que mede a estabilidade oxidativa de óleos vegetais e biodiesel a partir da determinação do IP (período de indução) a uma temperatura constante de oxidação forçada de 110°C. Este protótipo foi concebido com o intuito de nacionalizar esta tecnologia e reduzir custo, pois o método RANCIMAT Figura nas normas da ANP, para teste da qualidade deste biocombustível, entretanto, o equipamento comercial utilizado para fazer esta medição é importado e de custo elevado.

O BIOX10 é dotado de um sistema térmico, um sistema de instrumentação e um sistema de ar forçado. Na Figura 8 pode-se observar, em mais detalhes, o diagrama dos sistemas do BIOX10 e na Figura 9 é mostrado o protótipo BIOX10 em seu estado atual.

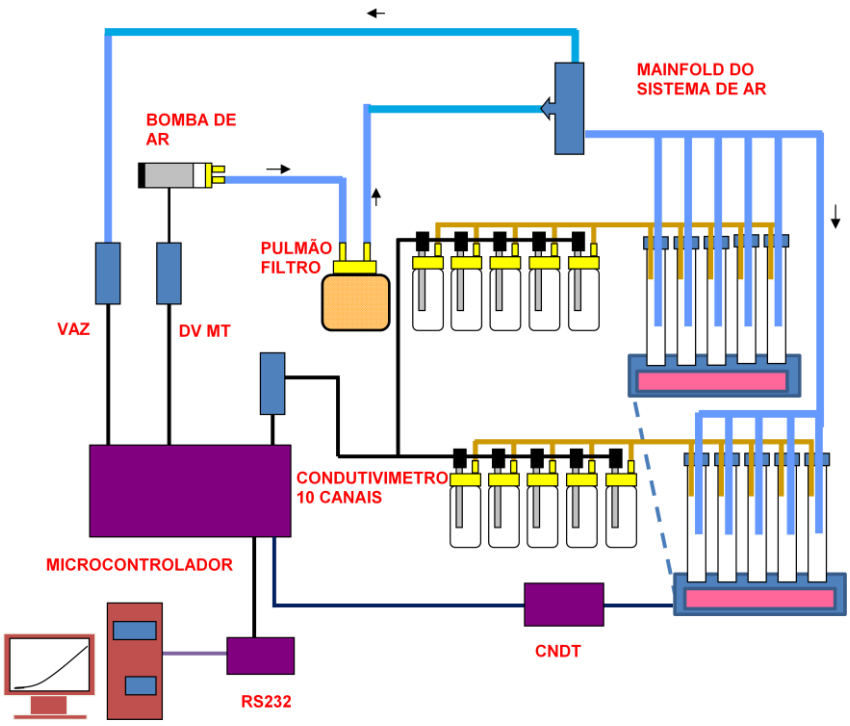


Figura 8. Diagrama dos sistemas do BIOX10.
Fonte: Relatório Técnico BIOX10.

Figura 9. Protótipo do BIOX10.

3.1.1 Sistema Térmico

O sistema de aquecimento é composto por um bloco de alumínio com furos para acomodar os tubos de ensaio que servem de suporte para as amostras. Acoplado a esse bloco, resistências elétricas são responsáveis por transferir calor para o bloco e consequentemente para as amostras. Para manter a temperatura constante, o protótipo é dotado de um senso térmico do tipo PT-100 e um controlador de temperatura com comunicação RS485, que permite o monitoramento dos parâmetros de temperatura por um computador pessoal (PC). Na Figura 10 é mostrado um bloco de alumínio semelhante ao utilizado no BIOX10 e na Figura 11 mostrada a placa controladora de temperatura.

Figura 10. Bloco de alumínio utilizado no aquecimento das amostras.

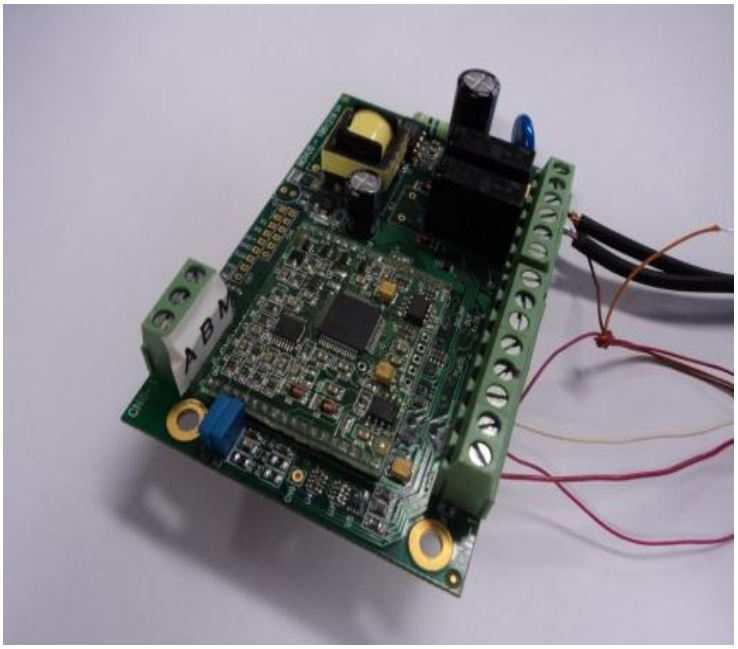


Figura 11. Placa Controladora da Temperatura do Bloco de alumínio

3.1.2 Sistema de Instrumentação

O sistema de instrumentação é composto de 10 canais com sensores de condutividade do modelo Q465M da QUIMIS, ilustrado na Figura 12, com uma célula

de medida de condutividade de constate 1 cm^{-1}. Este sistema possui também uma placa de condicionamento de sinal, que opera na faixa de leitura de 0 até 200 µS. Do ponto de vista da variável analógica medida 0 volt corresponde a 0 µS e 1 volt a 200 µS. Este sinal é convertido de analógico para digital pelo ADC (conversor analógico digital) de 10 bits do microcontrolador PIC 18F4550, fabricado pela Microchip, e estes dados são tratados, armazenado e apresentados pelo software dedicado, executado num computador pessoal. Mais detalhes do condicionamento de sinal podem ser visto no Anexo A.

Figura 12. Condutivimetro da QUIMIS.

3.1.3 Sistema de Ar

O sistema de fluxo de ar é composto de uma bomba de ar filtrado e seco, mangueiras e conexões, e tem a finalidade de borbulhar o ar na amostra, com uma vazão constante de 10 L/h, e assim transportar os produtos voláteis da oxidação para serem dissolvidos na água deionizada. No diagrama da Figura 13 são mostrados os seguintes elementos: 1 – filtro de partículas; 2 – bomba de gás; 3 – filtro molecular; 4 – sistema de distribuição de gás; 5 – tubo capilar; 6 – mangueira de distribuição; 7 – tubo de injeção. Este sistema de ar foi projetado para operar na faixa de vazão de 0 a 30 L/h (SOARES JUNIOR et al., 2012).

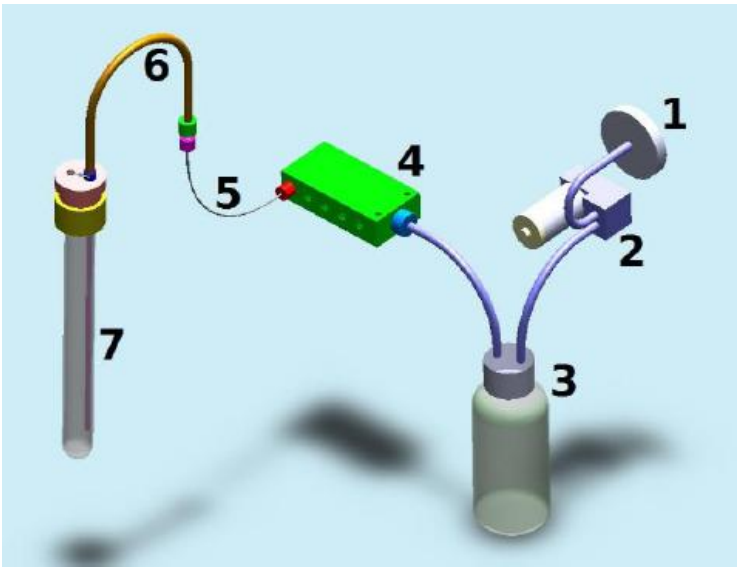


Figura 13. Diagrama do Sistema de Ar
Fonte: (Soares Junior et al., 2012)

Quando a bomba de ar, C103E-13, fabricada pela Parker, item 2 do diagrama mostrado na Figura 13, é acionada, o ar, inicialmente, passa pelo filtro de partículas (item 1) e em seguida passa pelo filtro molecular (item 3). Este processo de filtragem é de fundamental importância, pois além de reduzir significativamente a quantidade de partículas do ar, também reduz a umidade do mesmo, minimizando assim a quantidade de possíveis contaminantes das amostras, provenientes do ar.

O sistema de distribuição de gás (item 4) tem como finalidade distribuir ar para os dez capilares (item 5) e para o sensor de vazão. Nos capilares a vazão de ar é ajustada de forma a ter um fluxo laminar e constante ao longo das mangueiras de distribuição (item 6) e no tubo de injeção (item 7), que por sua vez injeta ar na amostra.

3.1.3.1 Ajuste da Vazão de Saída do Sistema de Ar

Com a finalidade de obter uma vazão uniforme ao longo das dez mangueiras de distribuição (item 7 da Figura 13) e diminuir as perturbações causadas quando há perda de carga devido ao uso das saídas de ar, em um ou mais pontos, foi necessário desenvolver um dispositivo que funcionasse como uma válvula de restrição da vazão ar.

Esta válvula, nada mais é do que um tubo capilar (item 6 da Figura 13) de diâmetro interno (D) de 0,8 mm e comprimento de 500 mm. Estas dimensões, juntamente com a pressão do ar fornecida pela bomba, criam uma grande resistência (alta impedância) à passagem do ar. Ou seja, um fluxo laminar de ar determinado pelo número de Reynolds, que pode ser calculado pela equação 1. Nesta equação, $\overline{\omega}$ é a velocidade de escoamento do gás, que vale 3 m/s e **v** é a viscosidade cinética do gás (Soares Junior et al., 2012).

$$Re = \frac{\overline{\omega} * D}{v} \qquad (1)$$

Para o escoamento ser laminar o número de Reynolds tem que ser menor que 2000 e se for maior do que 2400 o escoamento será turbulento. Como o número de Reynolds é de aproximadamente 139, pode ser obtido o coeficiente de atrito no tubo capilar λ, dada pela equação 2, além da perda de pressão no tubo capilar Δp. Aplicando o valor de λ na equação 3 e considerando que o escoamento é incompressivel, tem-se:

$$\lambda = \frac{64}{Re} \qquad (2)$$

$$\Delta p = \lambda \frac{L}{D} \frac{\rho}{2} \overline{\omega^2} \qquad (3)$$

Sendo assim, através da equação 3 obteve-se a perda de pressão no tubo capilar (Δp) igual a 1319 Pa (0,19 PSI). Desta maneira, a variações das pressões na saída nas mangueiras de distribuição, causada pelo acoplamento de um ou mais tubos de injeção é muito menor que a pressão no tubo capilar, que é inferior a 4% (Soares Junior et al., 2012).

3.2 O SOFTWARE DO BIOX10

O Biox10 é dotado de um software que aciona e controla os seus atuadores, faz a leitura da condutividade na água deionizada nos canais selecionados pelo usuário, salva os dados em arquivo de extensão txt, calibra automaticamente os condutivímetros e mostra a curva experimental de análise oxidativa em tempo real na tela. Este software foi desenvolvido utilizando a linguagem orientada a objeto C#, que é uma linguagem multiplataforma desenvolvida pela Microsoft, que usa o *framework.Net* do mesmo fabricante. Na Figura 14, pode ser visto a tela principal do software Biox10.

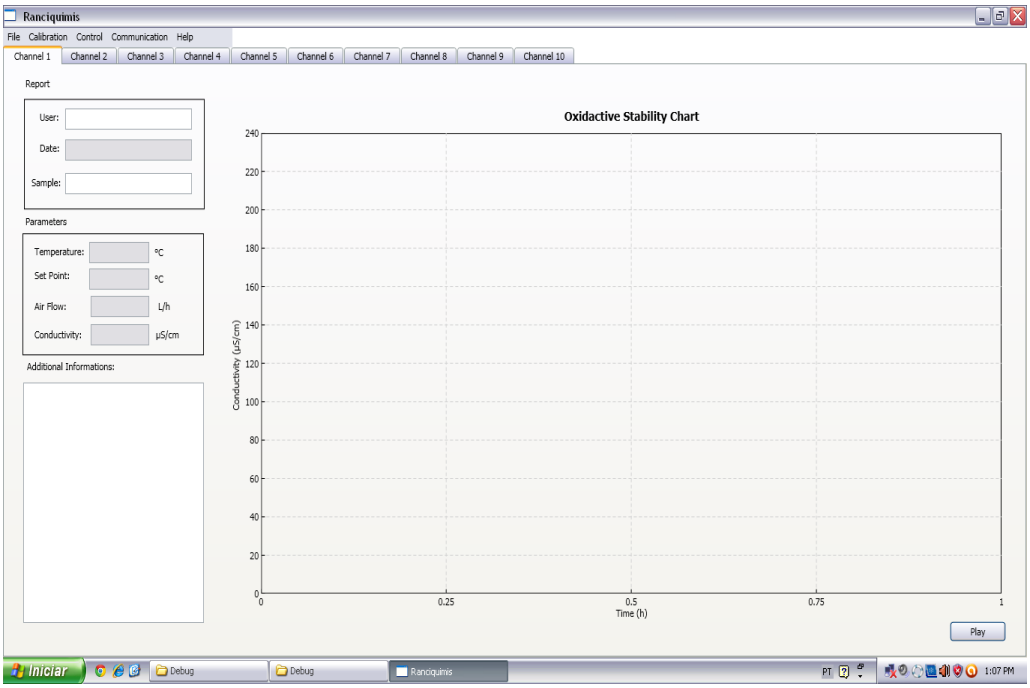


Figura 14. Tela Principal do Sofware Biox10.

Para fazer análise oxidativa é necessário informar a porta serial do PC de aquisição a qual o Biox10 está conectado e abrir a comunicação, isto pode ser feito na opção ***Communication*** da barra do menu. Feito isto, a próxima etapa é configurar os parâmetros de operação da aquisição de dados, como tempo de amostragem, temperatura de referência (*set-point*) e vazão, no item ***Parameters*** da opção ***Control*** do menu. Na Figura 15, pode-se notar a tela de configuração dos

parâmetros de operação da aquisição de dados do Biox10:

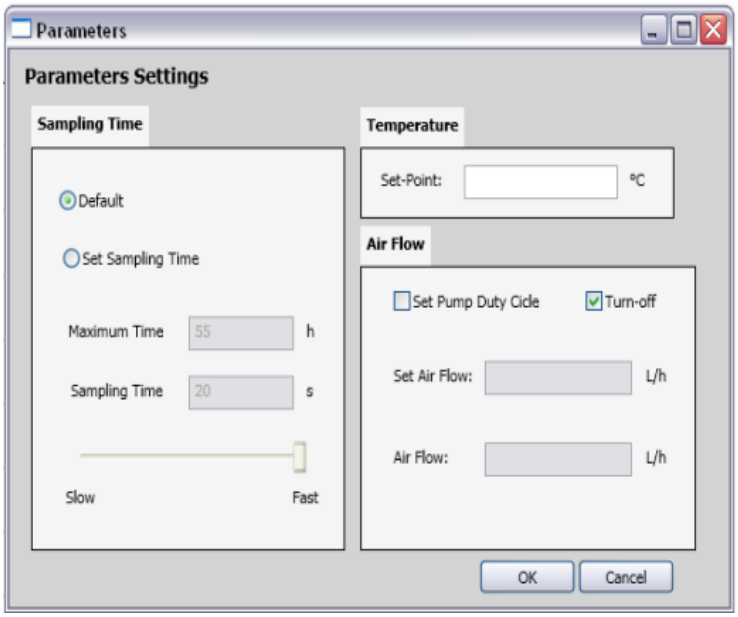


Figura 15. Tela de Configuração dos Parâmetros de Operação do Biox10.

Após configurar os parâmetros, é necessário aguardar a estabilização da temperatura do bloco de aquecimento. Quando isso ocorre, as amostras podem ser inseridas no equipamento. Enquanto o usuário espera o bloco atingir a temperatura de referência, ele pode informar ao software os dados que são pertinentes a cada amostra ou corrida de caracterização: seu nome, o nome da amostra, além de outras informações adicionais sobre as amostras que se pretende analisar. Em seguida, na tela principal, pressiona-se o botão de ***Play*** da aba correspondente ao canal da amostra, ou amostras, em que se pretende fazer a análise oxidativa. Na medida em que a leitura de condutividade de um dado canal chega em 200 µS, o software interrompe a leitura deste canal em particular e segue medindo os restantes, ou seja, cada canal tem um tempo próprio no qual a medição cessa.

3.2.1 Comandos de Comunicação

A comunicação do Biox10 é feita através da comunicação serial RS232. O protocolo de comunicação do Biox10 é bastante simples, pois consiste em enviar um comando de leitura ou escrita do tipo *string,* pela porta serial e aguardar uma

resposta do equipamento, que retorna outra *string, em caso de* leitura*,* ou acionamento de algum atuador dos sistemas que formam o equipamento. Para acionar a bomba é necessário enviar a *string* SD XXXX, em que XXXX é uma palavra binária de 10bits, ou seja, um valor entre 0 e 1023. De forma semelhante, para acionar o aquecedor envia-se ST XXXX. Ver Tabela 2.

Tabela 2. Tabela do Código de Acionamento da Bomba e do Aquecedor.

	Escrever
Acionar Bomba	SD XXXX
Acionar Aquecedor	ST XXXX

Para realizar a leitura da temperatura do *set-point*, temperatura do bloco, vazão e canais de condutividade é necessário enviar uma string solicitando ao equipamento a leitura da variável correspondente. Em seguida, o equipamento responde na forma de uma *string* composta de um identificador mais um número em contagens de um ADC de 10 bits. Na Tabela 3 é possível notar os códigos utilizados para a leitura dos sensores do Biox10:

Tabela 3. Tabela dos Códigos de Leitura dos Sensores.

	Escreve	Ler ADC (10bits)
Ler Temperatura_Bloco	LT	TT XXXX
Ler Set_point	LZ	TZ XXXX
Ler Vazão	LA6	A6 XXXX
Ler Canal1	LA0	A0 XXXX
Ler Canal2	LA1	A1 XXXX
Ler Canal3	LA2	A2 XXXX
Ler Canal4	LA4	A4 XXXX
Ler Canal5	LA5	A5 XXXX
Ler Canal6	LAB	AB XXXX
Ler Canal7	LA9	A9 XXXX
Ler Canal8	LA8	A8 XXXX
Ler Canal9	LAA	AA XXXX
Ler Canal10	LAC	AC XXXX

3.2.2 Fluxograma de Funcionamento do Software do Biox10

Quando o software do Biox10 é aberto, são criados os arquivos temporários em que são salvos os dados referentes à aquisição de dados de medida da condutividade dos diversos canais e são carregados os valores dos coeficientes de calibração que estão salvos em um arquivo de texto. Em seguida o contador é, repedidamente, disparado e interrompido após um determinado intervalo de tempo, configurado pelo usuário, este tempo pode variar entre 20 a 40 segundos. A cada evento de tempo, o software testa os canais que estão ativos em uma lista, cada canal ativo tem seu valor de condutividade e o tempo em que a medida foi feita lidos, essas informações são salvas em arquivos temporários. Posteriormente o software faz a leitura da temperatura do bloco de alumínio e da vazão de ar e são atualizados os conteúdos das caixas de texto da interface com o usuário. Essas grandezas podem então ser lidas, também é atualizado o gráfico da condutividade em função do tempo, dele é possível extrair o período de indução. Na Figura 16 é apresentado o fluxograma de funcionamento do software e no Apêndice A, pode ser visto o código fonte.

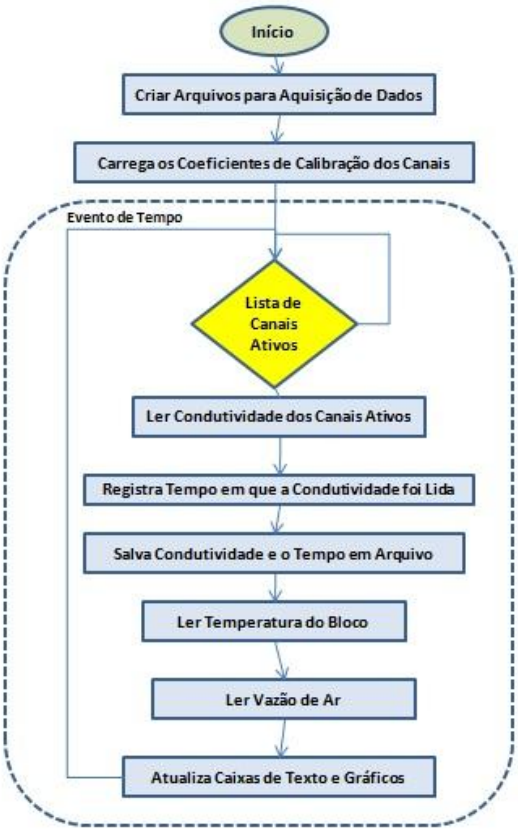


Figura 16. Fluxograma de Funcionamento do Software do Biox10.

3.3 PROCEDIMENTO DE CALIBRAÇÃO DOS SISTEMAS DO BIOX10

Como parte dos sistemas do BIOX10, os sensores estão presentes neste equipamento fornecendo a realimentação para a malha de controle da temperatura, monitorando a vazão de ar, controle de temperatura do bloco de aquecimento das amostras e a condutividade na água deionizada. As calibrações destes sensores garantem um bom funcionamento do equipamento, minimizando os erros nas medições do período de indução pelo BIOX10.

Como o Biox10 foi desenvolvido para atender a norma européia EN14112:2003, então busca – se que este equipamento atenda os seguintes requisitos previsto pela norma:

a) Vazão de ar no tubo de injeção de 10 L/h com erro de +/- 10%;
b) Temperatura de aquecimento do bloco 110°C com erro de +/- 0,1°C;
c) Massa da amostra 3g, com erro de +/- 0,01g;
d) 50 ml de água deionizada.

Neste capítulo serão discutidos os procedimentos de calibração dos instrumentos que compõem o protótipo desenvolvido, com a finalidade de atender os requisitos previstos pela norma EN14112.

3.3.1 Calibração do Fluxímetro BIOX10 do Sistema de Ar

Com um fluxímetro, fabricado pela Dwyer, modelo GFM – 1106, com precissão de +/- 7,5 mL/m, conectado na saída da mangueira de distribuição, conforme ilustra Figura 17, foi ajustando o ciclo ativo (*duty cycle)* da bomba, por

software, até que se obtivesse a vazão de 10 L/h. Em seguida foram medidas as vazões nas demais mangueiras de distribuição do equipamento correspondente aos demais canais, para averiguar e avaliar o erro entre eles, mais detalhes deste experimento podem ser vistos na Tabela 4.

Figura 17: Conexão do Fluxímetro na Mangueira de Distribuição

Tabela 4: Valores das vazões de ar nas mangueiras de distribuição.

Saidas de Ar	Vazão (ml/min)	Vazão (L/h)	DA	DR(%)
1	184	11,04	0,2459	2,23
2	180	10,8	0,006	0,1
3	180	10,8	0,006	0,1
4	179	10,74	0,054	0,5
5	182	10,92	0,1259	1,2
6	178	10,68	0,114	1,1
7	179	10,74	0,054	0,5
8	178	10,68	0,114	1,1
9	179	10,74	0,054	0,5
10	180	10,8	0,006	0,1
	media	10,794		0,5

DA	Desvio Absoluto
DR(%)	Desvio Relativo

Com base nos dados mostrados na Tabela 4, percebeu – se que a média do desvio relativo entre as saídas de distribuição de ar, para os 10 poços de medição, foi de +/-0,5%. Como a vazão média de saída foi de 10,794 L/h e o valor

ideal da vazão era de 10 L/h, foi constatado um erro de aproximadamente 8%, que é um resultado muito bom, pois a norma EN14112 tolera um erro de até 10%. Em seguida, foi escolhida a saída com o menor desvio relativo para a obtenção da curva de calibração do fluxímetro do BIOX10.

Para obter esta curva experimental foi necessário conectar o fluxímetro de referência, Dwyer modelo GFM – 1106 (ver Anexo B), a saída de ar escolhida e foi variado o *duty cycle* do controlador do tipo PWM que aciona a bomba, ao longo de sua região de funcionamento, entre 20% e 90% de sua potência nominal, ou ainda, entre 200 e 900 contagens do ADC de 10bits que digitaliza o sinal analógico do sensor de pressão MPX5010. Mais detalhes podem ser vistos na Tabela 5.

Tabela 5: Tabela de Dados para Curva de Calibração

Contagens	L/h
200	1.92
250	2.82
300	3.78
350	4.56
400	5.4
450	6
500	6.84
550	7.62
600	8.04
650	8.82
700	9
750	9.66
800	10.68
850	11.28
900	11.58

A partir dos dados da Tabela 5, foi obtida a curva de calibração do fluxímetro BIOX10, mostrada na Figura 18. A equação (4) é a regressão linear dos pontos experimentais, em que a vazão lida é *Q, Cont* corresponde às contagens do ADC e *off* é o valor do erro em relação ao zero (*off-set*) que vale 2,2.

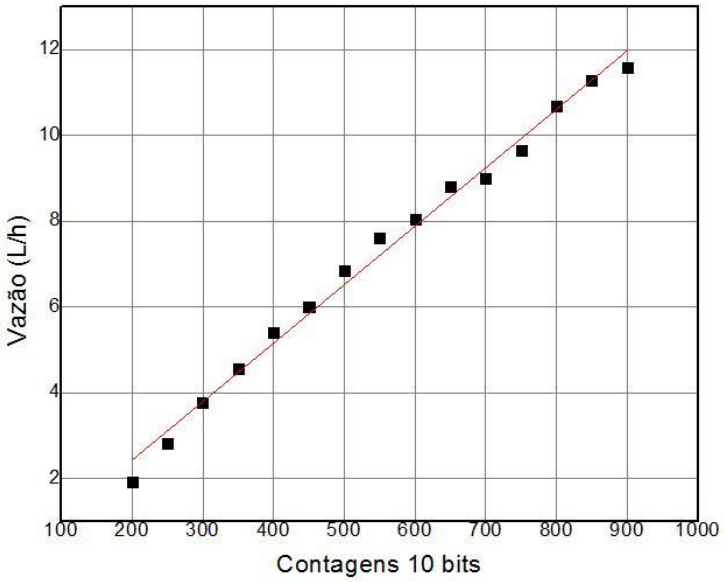


Figura 18. Curva de Calibração do Fluxímetro BIOX10

$$Q = 0{,}01364 * Cont - (0{,}30043 + off) \qquad (4)$$

Com a incorporação da equação 4, no sofware do BIOX10, foi possível monitorar a vazão de ar, que é fundamental para o processo de análise da qualidade oxidativa de biodiesel e óleos puros.

3.3.2 Calibração do Sistema Térmico

Para calibrar o sistema térmico foi necessário medir a temperatura nos pontos de aquecimento do bloco de alumínio e a temperatura das amostras nestes mesmos pontos de aquecimento. Assim, foi possível comparar o erro entre o valor alvo de temperatura (*set-point*) e a temperatura alcançada para cada amostra.

Inicialmente foi verificado o quão distante o *set-point* estava da temperatura média medida pelo PT-100 prisioneiro ao bloco de alumínio. Para fazer esta constatação, foi necessário ligar o BIOX10. Em seu software foi configurado o *set-point* de 110°C. Após aguardar a estabilização da temperatura do bloco, foram medidas as temperaturas nos poços de aquecimento, utilizando um termômetro

digital, modelo MT-455 da Minipa com a resolução de 0,1°C (ver Anexo C), que havia sido acoplado a uma peça de alumínio, com a mesma forma do fundo de um tubo de ensaio usado, para facilitar a troca de calor entre o termômetro e o bloco, conforme apresentado na Figura 19. Com estas medidas obteve-se a Tabela 6. Assim, foi possível avaliar o erro entre o *set-point* e a temperatura medida no PT-100 e o erro entre o PT-100 e a temperatura medida com o termômetro nos poços de medição.

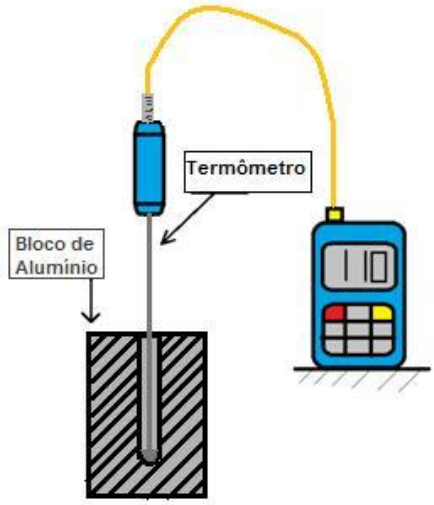


Figura 19. Termômetro MT-455 com um trocador de calor de alumínio na extremidade

Tabela 6: Tabela do erro nos pontos de distribuição de calor

	Temperatura nos Pontos de Aquecimento(°C)	Temperatura no PT-100(°C)	Temperatura Set-Point(°C)	Erro (°C)
Ponto 1	111,3	110 +/- 0,1	110	1,3
Ponto 2	110,9	110 +/- 0,1	110	0,9
Ponto 3	110,9	110 +/- 0,1	110	0,9
Ponto 4	111,5	110 +/- 0,1	110	1,5
Ponto 5	111,1	110 +/- 0,1	110	1,1
Ponto 6	111	110 +/- 0,1	110	1
Ponto 7	110,8	110 +/- 0,1	110	0,8
Ponto 8	110,9	110 +/- 0,1	110	0,9
Ponto 9	110,5	110 +/- 0,1	110	0,5
Ponto 10	110,9	110 +/- 0,1	110	0,9
Média	110,98	110+/- 0,1	110	0,98

Na Tabela 6, pode-se observar que o erro entre o *set-point* e a temperatura no PT-100 foi de +/- 0,1°C e a variação entre a temperatura média no

ponto de aquecimento nas amostras e a temperatura de referência foi de aproximadamente 1°C.

Para avaliar a necessidade de recalibrar o sistema térmico foi feito mais um teste, que consistiu em medir a temperatura em amostras de óleo. Novamente foi utilizada a temperatura de referência de 110°C e foi preparada uma amostra de 3g de óleo mineral em uma proveta. Em seguida foi inserida a amostra nos pontos de distribuição de calor e foram medidas as temperaturas, utilizando o mesmo termômetro citado anteriormente. Mais detalhes dos resultados deste experimento podem ser vistos na Tabela 7.

Tabela 7. Tabela da temperatura na amostra de óleo mineral

	Temperatura nos Pontos de Aquecimento(°C)	Temperatura Amostra(°C)
Ponto 1	111,3	109,5
Ponto 2	110,9	109,7
Ponto 3	110,9	109,9
Ponto 4	111,5	109,7
Ponto 5	111,1	110,4
Ponto 6	111,0	110,0
Ponto 7	110,8	109,5
Ponto 8	110,9	110,0
Ponto 9	110,5	109,8
Ponto 10	110,9	109,9
Média	110,98 +/- 0,18%	109,84 +/- 0,17%

Como observado na Tabela 7, a temperatura média nos pontos de aquecimento foi de aproximadamente 111°C e a temperatura na amostra de óleo foi de aproximadamente 110°C, não tendo assim a necessidade do sistema de controle de temperatura ser reaferido.

3.3.3 Calibração dos Condutivímetros do Sistema de Instrumentação

Para a calibração dos condutivímetros foi preparado uma solução padrão de 0,1Mol de cloreto de potássio (KCl) que corresponde a 7,4560 gramas de KCl diluídos em 1 litro de água deionizada, que resultou em uma condutividade de 14120

(µS/cm). Esta solução foi diluída em 50 vezes para ter uma solução padrão com a condutividade de 282,4 (µS/cm), que está na faixa próxima a das condutividades de trabalho dos equipamentos de Rancimat. A partir desta solução foram feitas novas diluições de 2x, 3x, 6x e 12x em água deionizada. Assim, foram obtidas novas soluções com condutividade de 141,2 (µS/cm), 94,1 (µS/cm), 47,1 (µS/cm) e 23,5 (µS/cm), respectivamente. Estas diferentes soluções padrão foram utilizadas para obter as curvas de calibração dos 10 condutivímetros que compõem o equipamento.

Para a obtenção destas curvas, o protótipo foi ligado e configurado para a vazão de 10 L/h, com o bloco na temperatura de 110°C. Em seguida, foram colocados 3g de óleo mineral nas provetas e com a temperatura ambiente de 25°C e foram introduzidas as soluções padrão no equipamento para serem medidas pelos condutivímetros. Para cada solução padrão foram feitas 3 medidas, a primeira medida foi feita após 5 minutos, a segunda foi feita após 10 minutos e a terceira após 15 minutos. Primeiramente foi medida a condutividade da água deionizada em todos os canais, em seguida foram medidas as condutividades, em contagens, das soluções padrão de 23,5 (uS/cm), 47,1 (uS/cm), 94,1 (uS/cm) e 141,2 (uS/cm). Através destas medidas foi possível obter os coeficientes da reta de calibração dos 10 condutivímetros. Na Tabela 8, é possível verificar os coeficientes angulares e lineares das retas (y=ax+b) de calibração.

Tabela 8. Coeficientes das Curvas de Calibração de Condutividade.

Canal	Coeficiente Angular	Coeficiente Linear
1	0,261	-0,6284
2	0,2713	0,39992
3	0,3058	-0,6247
4	0,3749	-2,8886
5	0,2925	2,416
6	0,2877	0,8813
7	0,3027	-0,6458
8	0,2844	-0,7396
9	0,2668	0,2482
10	0,2914	0,2866

Estes coeficientes de calibração serviram de base para desenvolver um método de calibração por *software*, semelhante ao método utilizado em um equipamento de Rancimat comercial, onde é possível calibrar os condutivímetros utilizando apenas uma solução padrão de 100 µS/cm a uma temperatura ambiente de

25ºC.

O método de calibração por *software* consiste em calcular o coeficiente angular da curva de calibração utilizando o próprio programa desenvolvido para fazer a análise oxidativa do biodiesel por método RANCIMAT. O método aqui reportado diferencia-se do anterior, pois não é necessário colocar óleo mineral no equipamento para em seguida aquecê-lo a uma temperatura de 110°C, e nem borbulhar ar a 10 L/h. É, entretanto, importante ter uma boa solução padrão de condutividade controlada e conhecida, e ajustar a temperatura ambiente do laboratório para 25°C.

Para calibrar os canais, foi necessário uma amostra padrão de condutividade de 100 µS/cm e ajustar a temperatura ambiente à 25°C, pois a leitura da condutividade varia com a temperatura, em seguida, na seção de calibração do *software* é informado o valor da temperatura e da solução padrão. Na sequência, o software faz a leitura dos dez condutivímetros e é possível perceber quais condutivímetros estão descalibrados. Após esta constatação, estes canais de medição foram selecionados e recalibrados, através do cálculo de um novo coeficiente angular. Para estes cálculos foram desprezados os coeficientes lineares, uma vez que se percebeu que estes coeficientes estavam próximos do zero (ver Tabela 6), restando assim calcular somente os coeficientes angulares utilizando a equação 5.

$$\alpha = \frac{SP}{Cont} \qquad (5)$$

Na equação 5, a constante *α* corresponde ao coeficiente angular, *SP* corresponde ao valor da condutividade da solução padrão e o *Cont* ao valor médio da condutividade lido em contagens de ADC. Após estes cálculos, foi possível obter uma Tabela com novos coeficientes angulares de calibração. Veja mais detalhes na Tabela 9.

Tabela 9. Novos Coeficientes de Calibração

Canal	Coeficiente Angular
1	0.27731
2	0.31766
3	0.31036
4	0.40617
5	0.30911
6	0.30129
7	0.31318
8	0.30293
9	0.27586
10	0.30921

3.4 MÉTODO NUMÉRICO PARA CALCULAR O PERÍODO DE INDUÇÃO

De acordo com a norma EN14112, o período de indução, em geral, pode ser obtido de duas maneiras. A primeira, através do método de interseção de duas retas tangentes ao longo da curva experimental, conforme apresentado na Figura 20. A segunda, através do ponto de máximo da segunda derivada da curva experimental no eixo das abicissas, conforme mostrado na Figura 21.

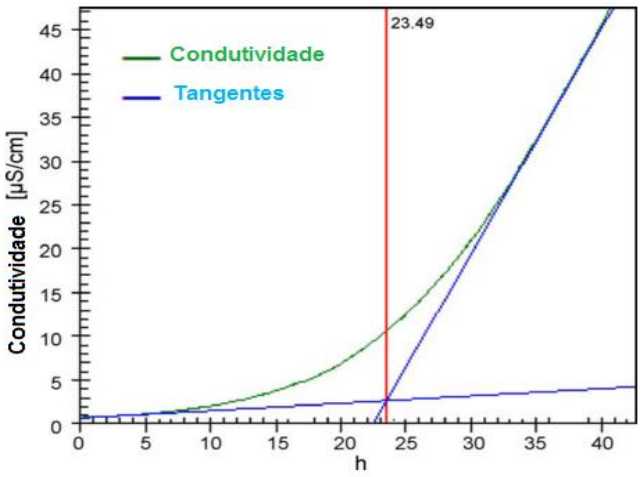


Figura 20. Período de Indução obtido pelo Método da Interseção das Retas Tangente
Fonte: JAIN; SHARMA, 2010

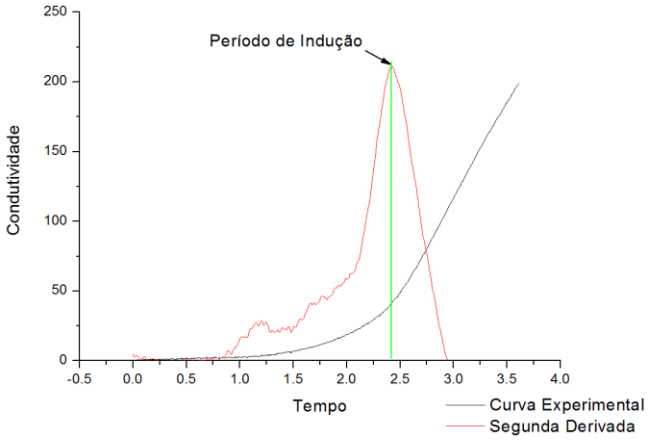


Figura 21. Método da Segunda Derivada

Com a finalidade de obter resultados mais precisos adotou-se o método da segunda derivada para determinar o período de indução das amostras estudadas.

Para exibir e analisar os resultados de monitoramento do estado de oxidação do biodiesel, e inclusive calcular a segunda derivada da curva experimental, utilizou-se o programa *Origin* versão 7. Este é um software especializado na preparação e apresentação de gráficos, que inclui um pacote de funções para cálculo numérico de parâmetros de interesse, ajuste de curvas e tratamento estatístico de grandezas, comercializado pela empresa *OriginLab Corporation*. Realizar este cálculo, utilizando esta plataforma, foi relativamente simples. Inicialmente os dados foram inseridos no software, em seguida tratados usando uma das ferramentas de alisamento (*smoothing*) pelo cálculo da média dos vizinhos ou *adjacent averaging*, depois, foi calculado, de forma numérica, a primeira derivada. Como a derivada de uma função (f(x)) dá conta da variação desta função a cada ponto (d/dx(f(x)), a nova função calculada (f´(x)) é mais susceptível a ruído. Para evitar que essa contaminação se propagasse nos resultados, foi necessário aplicar novo alisamento (*adjacent averaging*) nos dados calculados de f´(x), com a finalidade de minimizar o ruído, para então proceder o cálculo da segunda derivada (f´´(x)) e, assim, determinar seu ponto de máximo. Na maioria dos casos foi necessário aplicar duas vezes o alisamento antes de chegar na segunda derivada, para ter uma melhor identificação do período de indução, dado pelo ponto de máximo da curva f´´(x).

3.5 PROCEDIMENTO DE VALIDAÇÃO DO BIOX10

A validação do protótipo foi feita comparando a medida do tempo de prateleira do biodiesel, na temperatura de 25°C, utilizando o método ASLT (*Accelerated Shelf-Life Testing*) usando o protótipo desenvolvido e o Rancimat comercial modelo 873 da Metrohm.

O método ASLT consiste em extrapolar, por regressão linear, a reta do logaritmo do período de indução em função da temperatura (G. L. Hasenhuettl and P.J. Wan, 1992). A partir dessa extrapolação foi possível estimar o período de indução para temperaturas fora da faixa das medidas experimentais. Amostras similares foram ensaiadas, para a obtenção da reta do tempo de prateleira do biodiesel, no Rancimat 873 da Metrohm e no protótipo, ambos os equipamentos permitem a variação dos parâmetros temperatura do bloco de aquecimento e a vazão de ar.

Foram preparadas amostras com massa de aproximadamente 3g de biodiesel, em que se injetou ar a uma vazão constante de aproximadamente 10 L/h, em ambos os equipamentos. Em seguida foram medidos diferentes períodos de indução para as temperaturas de 80, 90, 100, 110 e 120°C, também nos dois aparelhos. Os diferentes períodos de indução do biodiesel derivado de uma matriz de sebo, para as diferentes temperaturas de ensaio, resultaram em retas de período de indução (IP) em função da temperatura. A partir destas retas foi possível estimar o período de indução para o biodiesel estudado, caso este biocombustível tivesse sido oxidado na temperatura ambiente de 25°C e assim determinado o IP à 25°C.

4 RESULTADOS

Após as medições dos períodos de indução de 3g de biodiesel no Rancimat 873 e no BIOX10, para as temperaturas de 80, 90, 100, 110 e 120°C, com a vazão de ar em 10 L/h, foram obtidos os resultados mostrados na Tabela 10.

Tabela 10. Períodos de indução obtidos no Rancimat 873 e no BIOX10 para as diferentes temperaturas.

Temperatura (°C)	Período de Indução Rancimat 873 (h)	Período de Indução BIOX10 (h)
80°C	39,68	35,65
90°C	16,93	14,79
100°C	8,18	7,15
110°C	3,61	3
120°C	1,8	1,7

A partir das informações contidas na Tabela 10, foi possível construir as retas experimentais e determinar suas equações por ajuste numérico, baseado em mínimos quadrados, de forma a determinar, indiretamente, os períodos de indução à temperatura de 25°C para o biodiesel estudado. As equações das retas ajustadas são apresentadas nas expressões (6), para o Rancimat Metrohm, e (7), reta obtida usando o BIOX10. As representações gráficas destes resultados podem ser vistas nas Figuras 22 e 23.

$$\log_{10} IP = -0{,}03358 * T + 4{,}2683 \quad (6)$$

$$\log_{10} IP = -0{,}03341 * T + 4{,}2054 \quad (7)$$

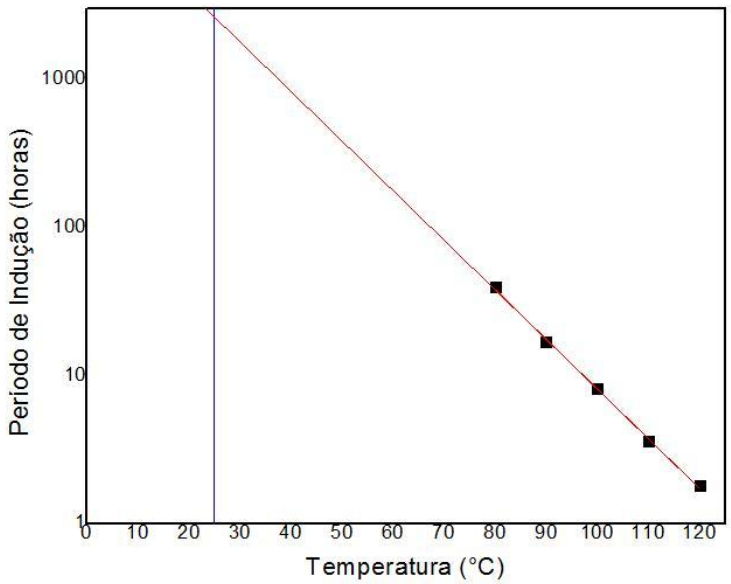


Figura 22. Reta ajustada e extrapolada para o período de indução em função da temperatura feita utilizando o Rancimat 873 comercial.

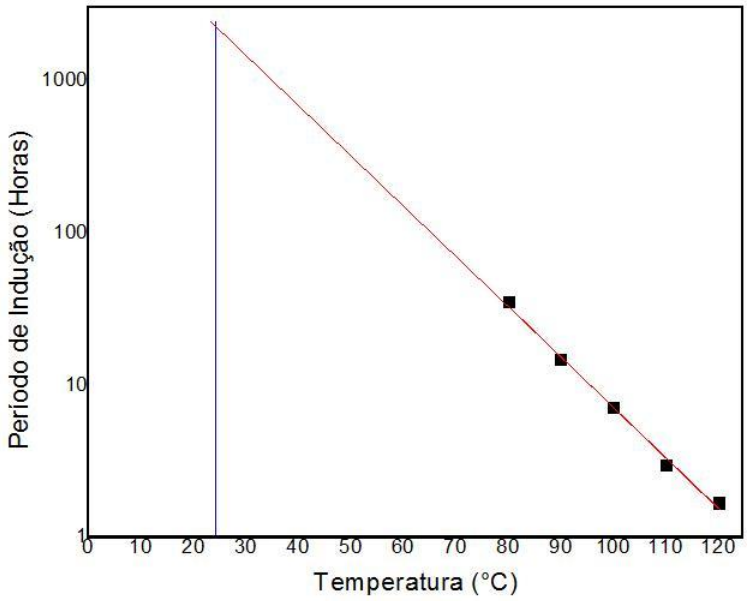


Figura 23. Reta ajustada e extrapolada para o período de indução em função da temperatura feita utilizando o BIOX10.

Para estimar os dois prazos de validade do biodiesel analisado foram usadas as equações (6) e (7) no cálculo do período de indução (IP) para a temperatura de 25°C. Com o Rancimat 873, usando a equação (6), o período de indução foi de 2684 horas, ou seja; 3 meses e 21 dias. O erro nesta estimativa é de 4,1%, o que pode ser considerado muito bom

para uma extrapolação, relativamente grande, já que as medidas se ativeram a temperaturas entre 80 e 120 °C. A equação (7) foi usada no cálculo do período de indução à 25°C para os resultados obtidos com o protótipo desenvolvido. O período de indução extrapolado foi de 2345 h ou 3 meses e 8 dias. A incerteza deste resultado, determinada a partir da curva extrapolada, é de 7,9%.

É importante dizer que o modelo de Rancimat usado para a validação do protótipo põe á disposição o operador uma rotina que permite avaliar o tempo de prateleira. Esta ferramenta de extrapolação do software Rancimat 873 não esclarece o método (tratamento dos dados, equações, etc.) usado. De qualquer forma, é importante reportar que empregando essa rotina foi encontrado um tempo de prateleira de 2467 h ou 3 meses e 12 dias. Mais detalhes podem ser vistos na Figura 24, onde é mostrado um *print screen* da tela com gráfico da extrapolação, o período de indução e os coeficientes da curva. Nota-se uma inversão nos eixos do gráfico, o que facilita o ajuste numérico e diminui o erro de extrapolação para a temperatura de 25°C. Este detalhe pode explicar o fato da incerteza no valor do tempo de prateleira, determinado com o Rancimat, ser cerca de duas vezes menor que a incerteza obtida com os dados do protótipo.

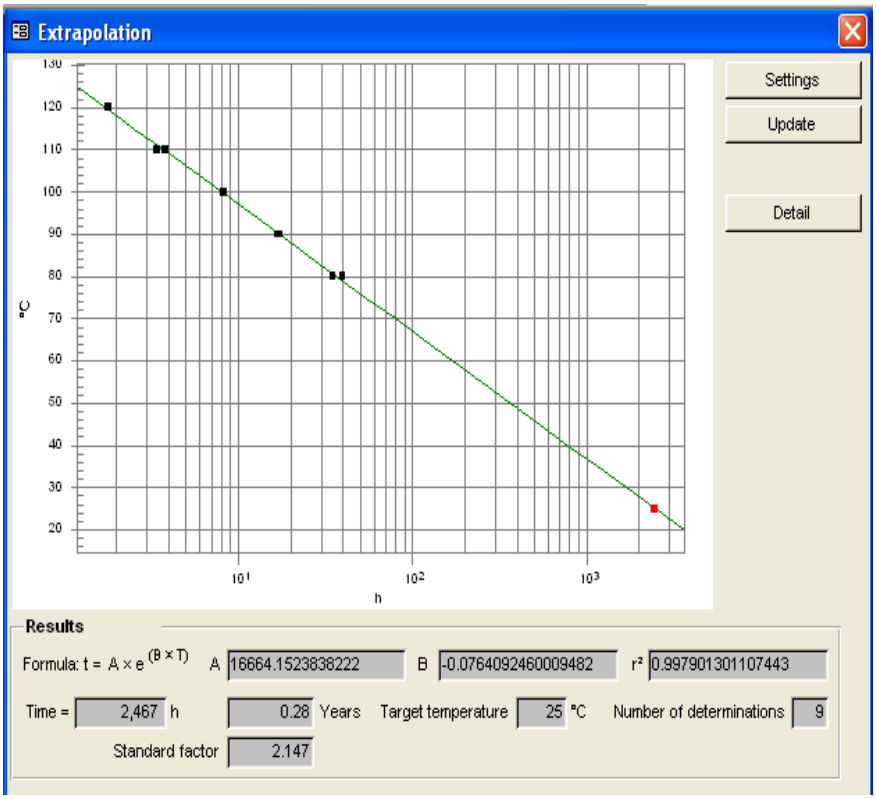


Figura 24. Gráfico da extrapolação da curva de predição do período de indução do biodiesel no software do Rancimat 873.

A Tabela 11 resume os resultados obtidos pelos dois diferentes equipamentos e pelos diferentes métodos de análise de dados, além do valor médio.

Tabela 11. Períodos de indução obtidos no Rancimat 873 e no BIOX10 para as diferentes temperaturas.

Método de análise	Rancimat 873 Metrohm	BIOX10
Extrapolação linear Origin	2684 +/- 107	2345 +/- 107
Extrapolação linear Metrohm	2467 +/- 99	-
Valor médio (h)	2699 +/- 185	

A Figura 25 ilustra os dados da Tabela 11, onde pode ser vista a boa concordância entre a medição feita pelos dois equipamentos, em especial, a absoluta compatibilidade (melhor que um desvio padrão) entre o tempo de prateleira determinado pelo BIOX10 em relação ao determinado pelo Rancimat 873, usando o tratamento de dados que o software deste fabricante traz (ponto 3). O ponto 4 é a média global determinada a partir dos três resultados (pontos 1, 2 e 3), este pode ser assumido como o valor mais provável para o tempo investigado. As barras de erro foram propositalmente computadas de forma a maximizar o intervalo de confiança, assumindo-se assim um desvio relativo de 6,9% na determinação desta grandeza.

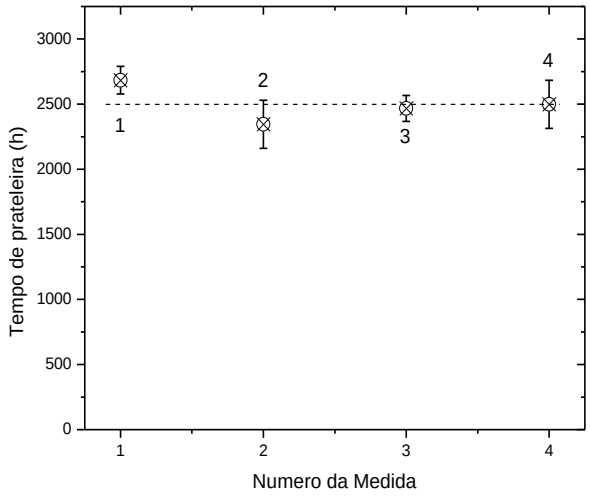


Figura 25. Resumo dos resultados e incertezas obtidos pelo Rancimat 873 e pelo BIOX10. Pontos 1 e 3 resultados do equipamento da Metrohm, pelo método de extrapolação linear e pelo método da extrapolação linear do software Metrohm, respectivamente. Ponto 2 resultado do BIOX10 por extrapolação linear. Ponto 4 média simples combinando todos resultados.

Observa-se também um pequeno afastamento entre o resultado do BIOX10 (ponto 2) e o resultado de Rancimat 873 (ponto 1), ambos determinados pela extrapolação linear, mesmo levando em conta o intervalo de confiança de cada medida. Este fato não chega a desabonar o acordo entre os equipamentos e pode ser explicado por uma subestimação da incerteza atribuída ao Rancimat, afinal, esta incerteza foi determinada a partir dos pontos medidos experimentalmente. Já que o fabricante Metrohm, em suas especificações técnicas, não apresenta um valor da incerteza global de medição e muito menos da incerteza na determinação do tempo de prateleira. Não obstante, ao presente erro característico de cada subsistema que compõe este equipamento.

5 OUTROS TESTES COM ÓLEOS E BIODIESEL

Com a finalidade de avaliar melhor o protótipo BIOX10, foi realizado mais alguns estudos com este equipamento, que possibilitaram observar melhor o comportamento do software e do hardware do BIOX10, quando usado para a análise oxidativa em diferentes óleos como soja, canola, dendê, coco, licuri e biodiesel de licuri, para diferentes temperaturas de forma a não extrapolar o limite de horas de trabalho do software, que era de 111 horas.

Para o óleo de soja, foram possíveis fazer as análises oxidativas nas temperaturas de 80, 90, 100, 110, 120 e 130 °C. Sendo que para a temperatura de 70°C não era possível, pois o período de indução iria praticamente dobrar, extrapolando o limite de horas de trabalho do software. Para as temperaturas entre 80 e 130°C foi obtido a Tabela 12.

Tabela 12. Tabela da Análise Oxidativa Realizada para Diferentes Temperaturas para o Óleo de Soja

Temperatura(°C)	IP Soja(h)
80	100,97
90	47,06
100	24,59
110	10,41
120	4,67
130	2,41

Para o óleo de canola, foram feitas as análises oxidativas para as temperaturas de 80, 90, 100, 110 e 120 °C. Apesar de ter feito estas medidas nestas temperaturas, poderia ser feito também nas temperaturas de 120°C e 70°C, uma vez que o período de indução praticamente dobra a cada 10°C que se reduz em relação ao período de indução obtido com a temperatura de aquecimento anterior. Os resultados podem ser vistos na Tabela 13.

Tabela 13. Tabela da Análise Oxidativa Realizada para Diferentes Temperaturas para o Óleo de Canola

Temperatura(°C)	IP Canola (h)
80	40,01
90	20,41
100	10,35
110	5,37
120	2,62

Para o óleo de dendê, foram feitas as análises oxidativas nas temperaturas de 90, 100, 110, 120 e 130 °C. Mais detalhes podem ser vistos na Tabela 14.

Tabela 14. Tabela da Análise Oxidativa Realizada para Diferentes Temperaturas para o Óleo de Dendê

Temperatura(°C)	IP Dendê (h)
90	93,71
100	41,35
110	17,93
120	8,61
130	4,14

Para o biodiesel de licuri, foram feitas as análises oxidativas nas temperaturas de 110, 120 e 130 °C. Os resultados são apresentados na Tabela 15.

Tabela 15. Tabela da Análise Oxidativa Realizada para Diferentes Temperaturas para o Biodiesel de Licuri

Temperatura (°C)	IP Biodiesel de Licuri (h)
110	9,5
120	6,29
130	2,96

Apesar de ter medido o período de indução em três temperaturas para um biodiesel de Licuri, também é possível fazer a análise oxidativa para as temperaturas de 100, 90 e 80°C, dentro dos limites de operação do software.

Para o óleo de licuri, foram feitas as análises oxidativas nas temperaturas

de 110, 120 e 130 °C. Detalhes dos resultados podem ser vistos na Tabela 16.

Tabela 16. Tabela da Análise Oxidativa Realizada para Diferentes Temperaturas para o Óleo de Licuri

Temperatura(°C)	IP Licuri (h)
110	36,27
120	20,18
130	10,6

Para o óleo de licuri, poderia obter também o período de indução para a temperatura de 100°C.

Para o óleo de coco, também foram feitas as análises oxidativas nas temperaturas de 110, 120 e 130 °C, porém, os resultados não foram conclusivos, pois o período de indução nas temperaturas de 110 e 120°C foram discordantes entre si. Este óleo, aparentemente, tem IP além do limite de trabalho do software do BIOX10, que é de 111 horas.

Portanto o BIOX10, nas condições atuais, obteve resultados satisfatórios para fazer a análise oxidativa a uma temperatura de 110°C em biodiesel de licuri e também em óleos de soja, canola, licuri e dendê. Entretanto, para óleos muito estáveis, como foi o caso do óleo de coco, as medidas não foram conclusivas, pois excediam os limites de operação do protótipo.

6 CONCLUSÃO

Neste trabalho foram discutidos alguns conceitos sobre biosiesel desde a sua produção ao método Rancimat, que foi utilizado para avaliar o biodiesel. Em seguida foi mostrado o funcionamento do Biox10, os métodos de calibração dos seus sistemas e a sua validação, comparando o Biox10 com o Rancimat 873, fabricado pela Metrohm.

A aferição do sistema de ar foi satisfatória, pois erro das medidas de vazão de ar ficou em torno de 8%, ou seja; abaixo de 10% previsto pela norma EN14112.

No sistema térmico, foi verificado que o erro entre a temperatura de referência e a temperatura média do bloco de alumínio, foi de +/- 0,1ºC e que a temperatura nos pontos de aquecimento são em média 1ºC acima da temperatura nas amostras. Na calibração dos condutivímetros foi mostrado o método para calcular os coeficientes de calibração dos dez canais.

Foi apresentada e discutida uma solução de software para o comando e o controle do protótipo desenvolvido, além dos procedimentos experimentais para sua validação.

A validação deste protótipo foi possível comparando o tempo de prateleira do biodiesel de matriz de sebo medido pelo Rancimat 873 da Metrohm, em relação à mesma medida feita pelo protótipo desenvolvido. Com Rancimat 873, foi encontrado um tempo de prateleira de 2684 horas com o erro aproximado de 4%. O tratamento de dados utilizou o software ORIGIN 7.0, da OriginLab Corp. para fazer a extrapolação linear do período de indução à 25 °C. Os dados medidos no Rancimat 873 também foram analisados no sofware fornecido pelo fabricante, foi obtido o tempo de prateleira de 2467 horas. O erro assumido para este resultado também foi de 4%, uma vez que o fabricante omite este dado.

Com o protótipo foi encontrado um tempo de prateleira de 2345 horas com erro de 7,9% para o mesmo biodiesel, pelo método da regressão linear no

software ORIGIN 7.0, da OriginLab Corp. Apesar de certo desacordo entre o resultado do Rancimat e do protótipo determinados por extrapolação linear, é inequívoca a compatibilidade entre os resultados do protótipo e do Rancimat 873, usando o software da Metrohm.

Portanto foi considerado validado o BIOX10, uma vez que os tempos de prateleira obtidos em ambos os equipamentos são compatíveis a menos de um desvio padrão, para o biodiesel de matriz de sebo. Porém com a finalidade de testar mais possibilidades de trabalho com este equipamento foram feitas medidas do período de indução em diversas temperaturas, para os óleos de soja, canola, dendê, licuri, coco e biodiesel de licuri. Durante estes testes foi possível fazer a análise oxidativa à 110°C destes óleos, porém para o óleo de coco, devido a sua alta estabilidade, não se obteve o mesmo sucesso, pois o experimento excedia o número máximo de horas de aquisição de dados do software, que era de 111 horas. Tal problema pode ser resolvido fazendo algumas alterações no software, para ampliar significativamente o número de horas de aquisição de dados do software do BIOX10, tornado assim possível, analisar óleos muito mais estáveis que o de coco a uma temperatura menor que 110°C.

SUGESTÕES PARA TRABALHOS FUTUROS

O projeto BIOX10 foi de grande valia para o estudo em questão, mas algumas melhorias podem ser implementadas tanto no software, quanto no hardware.

No software é necessário ampliar o tempo de aquisição de dados dos condutivímetros, para ser possível fazer a análise oxidativa de óleos com período de indução elevado e em temperaturas próximas a 25°C. É importante que no próprio software do BIOX10 tenha ferramentas de cálculo do período de indução e extrapolação linear para fazer a previsão do tempo de prateleira de óleos, gorduras e derivados, para que não haja necessidade do usuário comprar um software especialista em cálculo, para realizar estas operações.

O sistema de ar pode ser melhorado se for utilizado uma bomba de ar de motor *brushless* e de maior vazão. Pois a bomba *brushless*, do mesmo fabricante, proporciona, mais que o dobro da vida útil da bomba instalada.

Seria interessante acrescentar um sensor térmico ao sistema, para monitorar a temperatura ambiente, com a finalidade de reduzir o ruído térmico nas medidas de condutividade. Uma vez que a condutividade varia com a temperatura.

Reduzir o número de partes móveis, para reduzir o tempo gasto na preparação do experimento e na lavagem das peças após experimento.

Portanto o Biox10 é um protótipo que funciona bem para fazer análise oxidativa em óleos e biodiesel, porém este protótipo pode ser aperfeiçoado aplicando essas melhorias sugeridas nesta seção.

REFERÊNCIAS

Dunn, R. O. (2005). Oxidative stability of soybean oil fatty acid methyl esters by oil stability index (OSI). *Journal of the American Oil Chemists' Society*, *82*(5), 381–387. doi:10.1007/s11746-005-1081-6

G. L. Hasenhuettl and P.J. Wan. (1992). Temperature Effects on the Determination of Oxidative Stability with the Metrohm Rancimat. JAOCS.

Garcia, C. M., Prof, O., e Schuchardt, U. F. (2006). Transesterificação de óleos vegetais.

Ito, T., Sakurai, Y., Kakuta, Y., Sugano, M., e Hirano, K. (2012). Biodiesel production from waste animal fats using pyrolysis method. *Fuel Processing Technology*, *94*(1), 47–52. doi:10.1016/j.fuproc.2011.10.004

Jain, S., e Sharma, M. P. (2010). Review of different test methods for the evaluation of stability of biodiesel. *Renewable and Sustainable Energy Reviews*, *14*(7), 1937–1947. doi:10.1016/j.rser.2010.04.011

Lapuerta, M., Rodríguez-Fernández, J., Ramos, Á., e Álvarez, B. (2012). Effect of the test temperature and anti-oxidant addition on the oxidation stability of commercial biodiesel fuels. *Fuel*, *93*, 391–396. doi:10.1016/j.fuel.2011.09.011

Reda, S. Y., e Carneiro, P. I. B. (2007). óleos e gorduras : aplicações e implicações.

Sciences, A. P. (2011). Microemulsão : um promissor carreador para moléculas insolúveis, *32*(1), 9–18.

Sharma, Y. C., Singh, B., e Upadhyay, S. N. (2008). Advancements in development and characterization of biodiesel: A review. *Fuel*, *87*(12), 2355–2373. doi:10.1016/j.fuel.2008.01.014

Soares Junior, L. C. S., D'Erasmo, J. S., Costa Neto, P. R. da, e Pepe, I. M. (2012). Sistema de distribuição de gás para um equipamento de medida de estabilidade oxidativa de biocombustíveis. *VII Congresso Nacional de Engenharia Mecânica*, 8.

APÊNDICE A – Código Fonte do Software do Biox10

Main Window.Cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.IO.Ports;
using System.IO;
using System.Threading;
using System.Timers;
using System.Windows.Threading;

namespace Ranciquimis
{
   /// <summary>
   /// Interaction logic for MainWindow.xaml
   /// </summary>
   public partial class MainWindow : Window
   {
      #region variaveis
      Configurations Configura_Porta;
      SerialPort serial = new SerialPort();
      Manipula_Arquivo arquivo = new Manipula_Arquivo(10);
      Dados_Obj DAOB;
      Dados_operação DAOP;
      //SerialPort serial;
      String serialSelect;
      int id,i;
      Protocolo p=new Protocolo();
      //TimeSpan intervalo;
      long Dt_segundos;
      DateTime T1;
      int tempo_de_amostragem_ms = 20000, tempo_max_h, tamanho_timer = 200000;
      string tempo_sistema;
      string temperatura_bloco;
      string setpoint_temperatura;
      string vazao_tx;
      string update;
      string dados;
      double y1, y2, y3, y4, y5,y6,y7,y8,y9,y10;
      bool erro;
      bool ok_check;
```

```
    int horas_intervalo, minutos_intervalo, segundos_intervalo, segundos_total;
//    string setpoint, temperatura, dutycycle;


    string[] arrValores = new string[50];

    int c = 0;
    #endregion

    private static System.Timers.Timer aTimer;


   List<Canais> canaisativos = new List<Canais>();

    public MainWindow()
    {
        DAOB = new Dados_Obj();
        DAOP = Dados_Obj.getdados_operação();

        InitializeComponent();

        //Cria arquivos temporários para aquisição de dados

        arquivo.criar_todos_arquivos(10);
        carrega_variaveis_de_arquivo_calibracao_canais();

        for (int i = 0; i < 10; i++)//cria todos os canais na lista
        {

            Canais canal = new Canais(i);
            canaisativos.Add(canal);
        }



    }

   private delegate void UpdateUiTextDelegate(string text);


    private void OnTimedEvent(object source, ElapsedEventArgs e)
    {
        tempo_sistema = e.SignalTime.ToShortDateString();

        Dispatcher.Invoke(DispatcherPriority.Send, new UpdateUiTextDelegate(Atualizar),update);

        foreach (var canal in canaisativos)
        {
            if (canal.ativo)
            {
                T1 = DateTime.Now;
                Dt_segundos = (T1.Ticks - canal.H_Inicio.Ticks)/10000000;
                canal.Tempos_leitura.Add(Dt_segundos.ToString());
                canal.ler_condut_canais();

            }
        }

        ler_temperatura_setpoint_vazao();
    }
```

```
private void Atualizar(string texto)
{
    //Atualiza temperatura textbox
    temperatura_bloco = p.Read_temp.ToString();

    Canal1_control.temperatura_tb.Text = temperatura_bloco;
    Canal2_control.temperatura_tb.Text = temperatura_bloco;
    Canal3_control.temperatura_tb.Text = temperatura_bloco;
    Canal4_control.temperatura_tb.Text = temperatura_bloco;
    Canal5_control.temperatura_tb.Text = temperatura_bloco;
    Canal6_control.temperatura_tb.Text = temperatura_bloco;
    Canal7_control.temperatura_tb.Text = temperatura_bloco;
    Canal8_control.temperatura_tb.Text = temperatura_bloco;
    Canal9_control.temperatura_tb.Text = temperatura_bloco;
    Canal10_control.temperatura_tb.Text = temperatura_bloco;


    //Atualiza setpoint textbox
    setpoint_temperatura = p.Read_SP_TEMP.ToString();

    Canal1_control.setpoint_tb.Text = setpoint_temperatura;
    Canal2_control.setpoint_tb.Text = setpoint_temperatura;
    Canal3_control.setpoint_tb.Text = setpoint_temperatura;
    Canal4_control.setpoint_tb.Text = setpoint_temperatura;
    Canal5_control.setpoint_tb.Text = setpoint_temperatura;
    Canal6_control.setpoint_tb.Text = setpoint_temperatura;
    Canal7_control.setpoint_tb.Text = setpoint_temperatura;
    Canal8_control.setpoint_tb.Text = setpoint_temperatura;
    Canal9_control.setpoint_tb.Text = setpoint_temperatura;
    Canal10_control.setpoint_tb.Text = setpoint_temperatura;

    //Atualiza a vazao textbox
   // vazao_tx = p.Read_vazao.ToString();
    vazao_tx = String.Format("{0:f}", p.Read_vazao);

    Canal1_control.vazao_tb.Text = vazao_tx;
    Canal2_control.vazao_tb.Text = vazao_tx;
    Canal3_control.vazao_tb.Text = vazao_tx;
    Canal4_control.vazao_tb.Text = vazao_tx;
    Canal5_control.vazao_tb.Text = vazao_tx;
    Canal6_control.vazao_tb.Text = vazao_tx;
    Canal7_control.vazao_tb.Text = vazao_tx;
    Canal8_control.vazao_tb.Text = vazao_tx;
    Canal9_control.vazao_tb.Text = vazao_tx;
    Canal10_control.vazao_tb.Text = vazao_tx;

    //Atualiza caixa de texto da condutividade
    Canal1_control.condutividade_tb.Text = canaisativos.ElementAt<Canais>(0).Condut_text;
    Canal2_control.condutividade_tb.Text = canaisativos.ElementAt<Canais>(1).Condut_text;
    Canal3_control.condutividade_tb.Text = canaisativos.ElementAt<Canais>(2).Condut_text;
    Canal4_control.condutividade_tb.Text = canaisativos.ElementAt<Canais>(3).Condut_text;
    Canal5_control.condutividade_tb.Text = canaisativos.ElementAt<Canais>(4).Condut_text;
    Canal6_control.condutividade_tb.Text = canaisativos.ElementAt<Canais>(5).Condut_text;
    Canal7_control.condutividade_tb.Text = canaisativos.ElementAt<Canais>(6).Condut_text;
    Canal8_control.condutividade_tb.Text = canaisativos.ElementAt<Canais>(7).Condut_text;
    Canal9_control.condutividade_tb.Text = canaisativos.ElementAt<Canais>(8).Condut_text;
    Canal10_control.condutividade_tb.Text = canaisativos.ElementAt<Canais>(9).Condut_text;
```

```
    ////Atualização e evio das variáveis do gráfico: Y= condutividade e X= tempo
    y1 = canaisativos.ElementAt<Canais>(0).Read_condut;
    y2 = canaisativos.ElementAt<Canais>(1).Read_condut;
    y3 = canaisativos.ElementAt<Canais>(2).Read_condut;
    y4 = canaisativos.ElementAt<Canais>(3).Read_condut;
    y5 = canaisativos.ElementAt<Canais>(4).Read_condut;
    y6 = canaisativos.ElementAt<Canais>(5).Read_condut;
    y7 = canaisativos.ElementAt<Canais>(6).Read_condut;
    y8 = canaisativos.ElementAt<Canais>(7).Read_condut;
    y9 = canaisativos.ElementAt<Canais>(8).Read_condut;
    y10 = canaisativos.ElementAt<Canais>(9).Read_condut;

    double x1 = (double)(canaisativos.ElementAt<Canais>(0).condutividade.Count);
    double x2 = (double)(canaisativos.ElementAt<Canais>(1).condutividade.Count);
    double x3 = (double)(canaisativos.ElementAt<Canais>(2).condutividade.Count);
    double x4 = (double)(canaisativos.ElementAt<Canais>(3).condutividade.Count);
    double x5 = (double)(canaisativos.ElementAt<Canais>(4).condutividade.Count);
    double x6 = (double)(canaisativos.ElementAt<Canais>(5).condutividade.Count);
    double x7 = (double)(canaisativos.ElementAt<Canais>(6).condutividade.Count);
    double x8 = (double)(canaisativos.ElementAt<Canais>(7).condutividade.Count);
    double x9 = (double)(canaisativos.ElementAt<Canais>(8).condutividade.Count);
    double x10 = (double)(canaisativos.ElementAt<Canais>(9).condutividade.Count);

    ////envio das variáveis para o gráfico
    Canal1_control.AddData(x1, y1);
    Canal2_control.AddData(x2, y2);
    Canal3_control.AddData(x3, y3);
    Canal4_control.AddData(x4, y4);
    Canal5_control.AddData(x5, y5);
    Canal6_control.AddData(x6, y6);
    Canal7_control.AddData(x7, y7);
    Canal8_control.AddData(x8, y8);
    Canal9_control.AddData(x9, y9);
    Canal10_control.AddData(x10, y10);

}



private void Calibrar(object sender, RoutedEventArgs e)
{
    if ( (DAOP.Parametros_ok)&&(DAOP.Serial_configurada))
    {
        aTimer.Stop();
    }

    Calibra_Sensores Calibrar = new Calibra_Sensores();
    Calibrar.ShowDialog();

    if (DAOP.Parametros_ok)
    {
        aTimer.Start();
    }

}

private void Config(object sender, RoutedEventArgs e)
{
    Configura_Porta = new Configurations();
```

```
            Configura_Porta.ShowDialog();
            serialSelect = Configura_Porta.Comports_names.SelectionBoxItem.ToString();
        }

        private void Config_Parametros(object sender, RoutedEventArgs e)
        {
            Parameters Parametros = new Parameters();
            Parametros.ShowDialog();
            ok_check = DAOP.Parametros_ok;
            cria_timer();
        }

        private void Play1_bnt(object sender, RoutedEventArgs e)
        {
            id = 0;
            ativa_canal(id, Canal1_control.Sample_ct, Canal1_control.User_ct, tempo_sistema,
Canal1_control.Obs_ct, setpoint_temperatura, temperatura_bloco);
            Canal1_control.date_tb.Text = tempo_sistema;//imprime a data na tela
        }

        private void Play2_bnt(object sender, RoutedEventArgs e)
        {
            id = 1;
            ativa_canal(id, Canal2_control.Sample_ct, Canal2_control.User_ct, tempo_sistema,
Canal2_control.Obs_ct, setpoint_temperatura, temperatura_bloco);
            Canal2_control.date_tb.Text = tempo_sistema;//imprime a data na tela
        }

        private void Play3_bnt(object sender, RoutedEventArgs e)
        {
            id = 2;
            ativa_canal(id, Canal3_control.Sample_ct, Canal3_control.User_ct, tempo_sistema,
Canal3_control.Obs_ct, setpoint_temperatura, temperatura_bloco);
            Canal3_control.date_tb.Text = tempo_sistema;//imprime a data na tela
        }

        private void Play4_bnt(object sender, RoutedEventArgs e)
        {
            id = 3;
            ativa_canal(id, Canal4_control.Sample_ct, Canal4_control.User_ct, tempo_sistema,
Canal4_control.Obs_ct, setpoint_temperatura, temperatura_bloco);
            Canal4_control.date_tb.Text = tempo_sistema;//imprime a data na tela
        }

        private void Play5_bnt(object sender, RoutedEventArgs e)
        {
            id = 4;
            ativa_canal(id, Canal5_control.Sample_ct, Canal5_control.User_ct, tempo_sistema,
Canal5_control.Obs_ct, setpoint_temperatura, temperatura_bloco);
            Canal5_control.date_tb.Text = tempo_sistema;//imprime a data na tela

        }

        private void Play6_bnt(object sender, RoutedEventArgs e)
        {
            id = 5;
            ativa_canal(id, Canal6_control.Sample_ct, Canal6_control.User_ct, tempo_sistema,
Canal6_control.Obs_ct, setpoint_temperatura, temperatura_bloco);
            Canal6_control.date_tb.Text = tempo_sistema;//imprime a data na tela

        }
```

```
        private void Play7_bnt(object sender, RoutedEventArgs e)
        {
            id = 6;
            ativa_canal(id, Canal7_control.Sample_ct, Canal7_control.User_ct, tempo_sistema,
Canal7_control.Obs_ct, setpoint_temperatura, temperatura_bloco);
            Canal7_control.date_tb.Text = tempo_sistema;//imprime a data na tela

        }

        private void Play8_bnt(object sender, RoutedEventArgs e)
        {
            id = 7;
            ativa_canal(id, Canal8_control.Sample_ct, Canal8_control.User_ct, tempo_sistema,
Canal8_control.Obs_ct, setpoint_temperatura, temperatura_bloco);
            Canal8_control.date_tb.Text = tempo_sistema;//imprime a data na tela

        }

        private void Play9_bnt(object sender, RoutedEventArgs e)
        {
            id = 8;
            ativa_canal(id, Canal9_control.Sample_ct, Canal9_control.User_ct, tempo_sistema,
Canal9_control.Obs_ct, setpoint_temperatura, temperatura_bloco);
            Canal9_control.date_tb.Text = tempo_sistema;//imprime a data na tela

        }

        private void Play10_bnt(object sender, RoutedEventArgs e)
        {

            id = 9;
            ativa_canal(id, Canal10_control.Sample_ct, Canal10_control.User_ct, tempo_sistema,
Canal10_control.Obs_ct, setpoint_temperatura, temperatura_bloco);
            Canal10_control.date_tb.Text = tempo_sistema;//imprime a data na tela

        }

        public void ativa_canal(int id, string sample, string user, string tempo_sist, string obs,string setpoint, string
temperatura_bloco)
        {
            serial = Configurations.getconfigs();

            if (serial!=null)
            {

                Canais canal = new Canais(id, sample, user, tempo_sistema,
obs,setpoint_temperatura,temperatura_bloco);
                canaisativos.ElementAt<Canais>(id).Toggle();
                canaisativos.ElementAt<Canais>(id).H_Inicio = DateTime.Now;
                canaisativos.ElementAt<Canais>(id).setData(sample, user, tempo_sistema, obs,
setpoint_temperatura, temperatura_bloco);
            }
            else
            {
                MessageBox.Show("Erro:It not possible read channel! Set the serial configurations.");
                erro = true;
            }

        }
```

```
private void Save_ck(object sender, RoutedEventArgs e)
{
   arquivo.salvar_arquivo();
}

private void Fechar(object sender, RoutedEventArgs e)
{
   arquivo.deletar_arquivos(10);
   zera_acionamentos();
   this.Close();

}

private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
{
   arquivo.deletar_arquivos(10);
   zera_acionamentos();

}

public void zera_acionamentos()
{
   try
   {
      serial = Configurations.getconfigs();

      if (serial.IsOpen)
      {

         p.enviaserial("SD0\r");
         Thread.Sleep(30);
         p.enviaserial("ST0\r");
         serial.Close();

      }
   }
   catch
   {}

}

public void ler_temperatura_setpoint_vazao()
{
   if (ok_check)
   {
      try
      {
         serial = Configurations.getconfigs();

         if (serial.IsOpen)
         {
            p.ler_temperatura_bloco();
            p.ler_setpoint();
            p.Ler_Vazão();
         }


      }
      catch
      { }
```

```
            }

        }

        public void cria_timer()
        {
            if (ok_check)
            {
                tempo_de_amostragem_ms = DAOP.Amostragem_ms;
                tamanho_timer = DAOP.Tamanho_timer;
                aTimer = new System.Timers.Timer(tamanho_timer);
                aTimer.Elapsed += new ElapsedEventHandler(OnTimedEvent);
                aTimer.Interval = tempo_de_amostragem_ms;
                aTimer.Start();
            }

        }

        public void carrega_variaveis_de_arquivo_calibracao_canais()
        {
            StreamReader arquivo_calib_ch = new System.IO.StreamReader(@"C:\Ranciquimis
Files\Calibration\Calibration Channel.txt");
            while (!arquivo_calib_ch.EndOfStream)
            {
                c++;
                arrValores[c] = arquivo_calib_ch.ReadLine();
            }
            arquivo_calib_ch.Close();

            DAOP.Coef_angular_ch1 = double.Parse(arrValores[6].Remove(0, 22));
            DAOP.Coef_angular_ch2 = double.Parse(arrValores[10].Remove(0, 22));
            DAOP.Coef_angular_ch3 = double.Parse(arrValores[14].Remove(0, 22));
            DAOP.Coef_angular_ch4 = double.Parse(arrValores[18].Remove(0, 22));
            DAOP.Coef_angular_ch5 = double.Parse(arrValores[22].Remove(0, 22));
            DAOP.Coef_angular_ch6 = double.Parse(arrValores[26].Remove(0, 22));
            DAOP.Coef_angular_ch7 = double.Parse(arrValores[30].Remove(0, 22));
            DAOP.Coef_angular_ch8 = double.Parse(arrValores[34].Remove(0, 22));
            DAOP.Coef_angular_ch9 = double.Parse(arrValores[38].Remove(0, 22));
            DAOP.Coef_angular_ch10 = double.Parse(arrValores[42].Remove(0, 23));
        }

    }
}
```

Configurations.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.IO.Ports;
```

```
namespace Ranciquimis
{
    /// <summary>
    /// Interaction logic for Configurations.xaml
    /// </summary>
    public partial class Configurations : Window
    {
        private static SerialPort serial= null;
        bool serial_config_ok = false;

        Dados_operação DOP_Config;

        public Configurations()
        {
            InitializeComponent();
            serial = new SerialPort();
            Comports_names.SelectedIndex = 0;
            Baud_Rates.SelectedIndex = 0;
            DOP_Config = Dados_Obj.getdados_operação();

            foreach (string s in SerialPort.GetPortNames())
            {
                Comports_names.Items.Add(s);
            }
        }
        public void configuraporta()
        {
          // serial = new SerialPort();
            try
            {
                serial.PortName = Comports_names.Text;
                serial.BaudRate = Convert.ToInt32(Baud_Rates.Text);
                serial.Handshake = System.IO.Ports.Handshake.None;
                serial.Parity = Parity.None;
                //serial.RtsEnable = false;
                //serial.DtrEnable = true;
                serial.DataBits = 8;
                serial.StopBits = StopBits.One;
                serial.ReadTimeout = 200;
                serial.WriteTimeout = 50;
            }
            catch(Exception e)
            {
                MessageBox.Show(""+e);
            }
        }

        public static SerialPort getconfigs()
        {
            return serial;
        }

        private void OK_config(object sender, RoutedEventArgs e)
        {
            if (!(serial.IsOpen))
            {
                configuraporta();
                try
                {
```

```
                serial.Open();
                serial_config_ok = true;
                DOP_Config.Serial_configurada = serial_config_ok;
                this.Close();
            }
            catch(Exception ex)
            {
                MessageBox.Show("Erro: Serial is Open."+ex);
            }
        }
        else
        {

        }
    }

    private void Cancel_config(object sender, RoutedEventArgs e)
    {
        this.Close();
    }

  }
}
```

Parameters.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.IO.Ports;
using System.Threading;
using System.Timers;
using System.Windows.Threading;

namespace Ranciquimis
{
  /// <summary>
  /// Interaction logic for Parameters.xaml
  /// </summary>
  public partial class Parameters : Window
  {
    int Set_Point, DutyCycle, Amostragem_s, Max_time_horas,timer_ms, Amostragem_ms;
    double vazao;

    string Set_DutyCycle,vazao_lh;
    string update;
    bool parametros_ok, Parametros_ok2;
    Protocolo prot = new Protocolo();
    Dados_operação DAOP2;
    //Configurations Configura_Porta;
```

```
SerialPort serial = new SerialPort();

private static System.Timers.Timer bTimer;

public Parameters()
{
   InitializeComponent();
   DAOP2 = Dados_Obj.getdados_operação();
   bTimer = new System.Timers.Timer(2000);
   bTimer.Elapsed += new ElapsedEventHandler(OnTimedEvent);
   bTimer.Interval = 200;

}

private delegate void UpdateUiTextDelegate(string text);


private void OnTimedEvent(object source, ElapsedEventArgs e)
{

   Dispatcher.Invoke(DispatcherPriority.Send, new UpdateUiTextDelegate(Atualizar_text), update);
   prot.Ler_Vazão();
   vazao = prot.Read_vazao;
}

private void Atualizar_text(string texto)
{
   Vazao_tb_h.Text = String.Format("{0:f}", prot.Read_vazao);


}

#region Set-Point_Temperatura


private void Set_Point_box(object sender, KeyEventArgs e)
{
   if (bTimer.Enabled)
   {
      bTimer.Stop();
   }

   if (e.Key == Key.Enter)
   {

      Setpoint_Temperatura();

      //if (Pump_slider.IsEnabled)
      //{
      //    bTimer.Start();
      //}

   }
}

public void Setpoint_Temperatura()
{
   Set_Point = Convert.ToInt16(float.Parse(textBox1.Text) * 10);
   string tex2 = "ST" + (Set_Point).ToString()+"\r";
```

```
    prot.enviaserial(tex2);

}
#endregion

#region Set_Point_Bomba

private void Pump_checked(object sender, RoutedEventArgs e)
{
    try
    {
        turn_off_btm.IsChecked = false;
        Set_AirFlow_tb.IsEnabled = true;

    }
    catch
    {

    }
}

private void Turnoff_checked(object sender, RoutedEventArgs e)
{
    try
    {
        if (bTimer.Enabled)
        {
            bTimer.Stop();
            Pump_ck_bt.IsChecked = false;
            Thread.Sleep(50);
            prot.enviaserial("SD0\r");

        }
    }
    catch
    {


    }



}

#endregion

private void Ok_Parametros(object sender, RoutedEventArgs e)
{
    parametros_ok = true;
    DAOP2.Parametros_ok = parametros_ok;
    DAOP2.Amostragem_ms = Amostragem_ms;
    DAOP2.Tamanho_timer = timer_ms;
    DAOP2.Tempo_max = Max_time_horas;
    DAOP2.Vazao_parametro = vazao;

    if (bTimer.Enabled)
    {
        bTimer.Stop();
    }
    this.Close();
```

```
}

private void Cancel_click(object sender, RoutedEventArgs e)
{

    if (bTimer.Enabled)
    {
        bTimer.Stop();
    }
    prot.enviaserial("SD0\r");
    Thread.Sleep(30);
    prot.enviaserial("ST0\r");

    this.Close();
}

public bool test_ok_parametros()
{
   return parametros_ok;
}

private void Default_check(object sender, RoutedEventArgs e)
{

   Amostragem_ms = 20000;
   timer_ms = 200000;
   Max_time_horas = 55;

   try
   {

      slider2.IsEnabled = false;
      slider2.Value = 20;
   }
   catch
   {

   }

}

private void Sampling_Time_check(object sender, RoutedEventArgs e)
{
   slider2.IsEnabled = true;

}

private void Amostragem_slider_change(object sender, RoutedPropertyChangedEventArgs<double> e)
{
   try
   {
      Amostragem_s = (int)slider2.Value;
      Sample_time_tb.Text = Amostragem_s.ToString();
      Max_time_horas = 10000 / ((3600 / Amostragem_s)); //estima tempo maximo de medida
      Max_time_tb.Text = Max_time_horas.ToString();
      Amostragem_ms = Amostragem_s * 1000;
      timer_ms = Amostragem_ms * 10;

      DAOP2.Amostragem_ms = Amostragem_ms;
```

```
            DAOP2.Tamanho_timer = timer_ms;
            DAOP2.Tempo_max = Max_time_horas;
        }
        catch
        {

        }

    }

    private void Set_Air_Flow_keydown(object sender, KeyEventArgs e)
    {
        if (bTimer.Enabled)
        {
            bTimer.Stop();
        }

        if (e.Key==Key.Enter)
        {

            //string ADC = Set_AirFlow_tb.Text;
            //Set_DutyCycle = "SD" + ADC + "\r";
            vazao_lh = Set_AirFlow_tb.Text;
            DutyCycle = Convert.ToInt32((double.Parse(vazao_lh)*72.71614) + 26.44381);
            Set_DutyCycle = "SD" + DutyCycle.ToString() + "\r";
            prot.enviaserial(Set_DutyCycle);
            bTimer.Start();

        }
    }

  }
}
```

InterfaceChanels.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Windows.Threading;

namespace Ranciquimis
{
  /// <summary>
  /// Interaction logic for InterfaceChanels.xaml
  /// </summary>
  public partial class InterfaceChanels : UserControl
  {
```

```
Dados_operação Daop3 = null;
const int tam_vetor_y = 10000;
double[] Vetor_y = new double[tam_vetor_y];

int cont200pt, contx, pt_base, cont_pt, pt_max;
double x_maximo, intervalo_x;


string atualiza;
// int tempo_amostragem_s = 20;
int tempo_amostragem_s;
bool parametro_ok_check;

//int tempo_amostragem_s;

private string user_ct;

public string User_ct
{
    get { return user_ct; }
    set { user_ct = value; }
}

private string sample_ct;

public string Sample_ct
{
    get { return sample_ct; }
    set { sample_ct = value; }
}

private string temperatura_ct;

public string Temperatura_ct
{
    get { return temperatura_ct; }
    set { temperatura_ct = value; }
}

private string setpoint_ct;

public string Setpoint_ct
{
    get { return setpoint_ct; }
    set { setpoint_ct = value; }
}

private string vazao_ct;

public string Vazao_ct
{
    get { return vazao_ct; }
    set { vazao_ct = value; }
}

private string obs_ct;

public string Obs_ct
{
    get { return obs_ct; }
```

```
            set { obs_ct = value; }
        }


        public InterfaceChanels()
        {
            InitializeComponent();
            DataContext = this;
            Daop3 = Dados_Obj.getdados_operação();

            //limpar Vetor_y
           // tempo_amostragem_s = Daop3.Amostragem_ms / 1000;

            for (int j = 0; j < Vetor_y.Length; j++)
            {
                Vetor_y[j] = 0.0;
            }



        }

        private void rootGrid_SizeChanged(object sender, SizeChangedEventArgs e)
        {
            myLineChart.Width = rootGrid.ActualWidth - 302;
            myLineChart.Height = rootGrid.ActualHeight - 60;
            // AddData();
            // redimenciona_grafico();

        }


        public void AddData(double x, double y)//X é o número de elementos da lista de condutividade e Y é o
valor da condutividade do canal correspondente
        {
            settempos();
            contx = (int)x;
            Vetor_y[(int)x] = y;
            myLineChart.chartGrid_update();
            myLineChart.DataCollection.DataList.Clear();

            LineCharts.DataSeries ds = new LineCharts.DataSeries();
            ds.LineColor = Brushes.Red;
            ds.LineThickness = 2;

            ds.LineSeries.Points.Clear();

            for (int i = 1; i <= (int)x; i++)
            {

                double xp = ((double)i*tempo_amostragem_s)/3600;
                double yp = Vetor_y[i];

                ds.LineSeries.Points.Add(new Point(xp, yp));

            }

            redimensiona_grafico();
            myLineChart.DataCollection.DataList.Add(ds);
```

```
        }

        public void redimensiona_grafico()
        {

            pt_base = 3600/tempo_amostragem_s; //calculo de quantos pontos em um determinado tempo de amostragem leva para completar uma hora

            cont_pt = contx / pt_base; //conta quantos grupos de pontos em uma hora de medida
            pt_max=10000/pt_base;//quantidade maxima de horas
           // pt_max = Daop3.Tempo_max; //quantidade maxima de horas

            if ((cont_pt>=1)||(cont_pt<=pt_max))
            {
                x_maximo = (((cont_pt*pt_base) + pt_base) * tempo_amostragem_s / 3600);
                intervalo_x = x_maximo / 4;

                myLineChart.Xmax = x_maximo;
                myLineChart.XTick = intervalo_x;
            }


        }

        public void settempos()
        {
            if (Daop3.Parametros_ok)
            {
                tempo_amostragem_s = (Daop3.Amostragem_ms) / 1000;
            }
            else
            {
                tempo_amostragem_s = 20;
            }
        }

        private void button1_Click(object sender, RoutedEventArgs e)
        {
            if ((string)button1.Content == "Play")
            {
                button1.Content = "Stop";
                //parametro_ok_check = Daop3.Parametros_ok2;

            }

            else
            {
                try
                {
                    button1.Content = "Play";
                }
                catch
                { }
            }

        }

        private void user_kd_ev(object sender, KeyEventArgs e)
        {
```

```
        if (e.Key==Key.Enter)
        {
            user_ct = user_tb.Text;
        }

    }

    private void sample_kd_ev(object sender, KeyEventArgs e)
    {

        if (e.Key == Key.Enter)
        {
            sample_ct = sample_tb.Text;
        }
    }

    private void temperatura_kd_ev(object sender, KeyEventArgs e)
    {
        if (e.Key == Key.Enter)
        {
            temperatura_ct = temperatura_tb.Text;
        }
    }

    private void setpoint_kd_ev(object sender, KeyEventArgs e)
    {
        if (e.Key == Key.Enter)
        {
            setpoint_ct = setpoint_tb.Text;
        }
    }

    private void vazao_kd_ev(object sender, KeyEventArgs e)
    {
        if (e.Key == Key.Enter)
        {
            vazao_ct = vazao_tb.Text;
        }
    }


    private void user_ev_lostfocus(object sender, RoutedEventArgs e)
    {
        user_ct = user_tb.Text;
    }

    private void sample_ev_lostfocus(object sender, RoutedEventArgs e)
    {
        sample_ct = sample_tb.Text;
    }

    private void Obs_lostfocus(object sender, RoutedEventArgs e)
    {

        Obs_rtb.SelectAll();
        obs_ct = Obs_rtb.Selection.Text;
    }


}
```

```
}
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
Canais.cs
-----------------------------------------------------------------------------------------------------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.IO.Ports;
using System.IO;
using System.Threading;
using System.Windows.Threading;
using System.Timers;


namespace Ranciquimis
{

    class Canais : Protocolo
    {

        Dados_operação DAOP_ch;
        int Id;
        public string Sample_Name="";
        string User_Name = "";
        string Data_Hora ="";
        public DateTime H_Inicio;   //Hora de inicio das medias
        public string OBS="";
        public bool ativo = false;
        string Vazão;
        string Temperatura;
        string Temperatura_setpoint;
        string IP="0";

        //string Tempo_Atual, Tempo_Anterior, Dt;
        int Horas;
        int Minutos;
        int Segundos;
        int Milisegundos;

        string index;
        string Dt;
        string condutividade_uni;
        string Tempo_Atual1;

         string conteudo;



        public Canais(int id)
        {
```

```
            // TODO: Complete member initialization
            this.Id = id;
        }

        public Canais(int id, string sample_name, string user_name, string data_hora, string obs, string
temperatura_setpoint, string temperatura_bloco) :this(id)
        {
            // TODO: Complete member initialization
            this.Sample_Name = sample_name;
            this.User_Name = user_name;
            this.Data_Hora = data_hora;
            this.OBS = obs;
            this.Temperatura_setpoint = temperatura_setpoint;
            this.Temperatura = temperatura_bloco;

        }

        public void setData(string sample_name, string user_name, string data_hora, string obs, string
temperatura_setpoint, string temperatura_bloco)
        {
            // TODO: Complete member initialization
            this.Sample_Name = sample_name;
            this.User_Name = user_name;
            this.Data_Hora = data_hora;
            this.OBS = obs;
            this.Temperatura_setpoint = temperatura_setpoint;
            this.Temperatura = temperatura_bloco;
        }

        //Configurations Configura_Porta;
        SerialPort serial = new SerialPort();
        public List<string> condutividade = new List<string>();
        public List<string> Tempos_leitura = new List<string>();

        #region Ler_Condutividade_Canais

        public double ler_condut(int id)
        {

            identifica_conteudo();
            base.enviaserial(conteudo);
            base.ler_serial();

            return base.Read_condut;

        }

        public int ler_condut_adc(int id)
        {

            identifica_conteudo();
            base.enviaserial(conteudo);
            base.ler_serial();

            return base.Read_contagens_adc;

        }

        public void ler_condut_canais()
        {
```

```
                    DAOP_ch = Dados_Obj.getdados_operação();
                    identifica_conteudo();
                    base.enviaserial(conteudo);
                    base.ler_serial();


                    if ((condutividade.Count >= 9999) || (base.Read_condut >= 200))//Dados no limite muda o estado do
Toggle para parar o canal lido p/ interromper leitura em 200 micro-siemmens
                    {
                        ativo = false;
                        MessageBox.Show("Measurement done.");
                    }

                    condutividade.Add(base.Condut_text);

                    index = condutividade.Count.ToString();
                    Dt = Tempos_leitura.Last();
                    condutividade_uni = base.Condut_text;
                    registra_tempo_medidas();
                   // Temperatura = base.Read_temp.ToString();

                    Manipula_Arquivo arquivo = new Manipula_Arquivo(Id, Sample_Name, User_Name, Data_Hora, OBS,
DAOP_ch.Vazao_parametro.ToString(), Temperatura, IP, index, Dt, condutividade_uni, Tempo_Atual1, Temperatura_setpoint);
                    arquivo.grava_no_arquivo();


                }
                #endregion


                public void Toggle()
                {
                    if (this.ativo)
                        this.ativo = false;
                    else
                        this.ativo = true;
                }

                public void registra_tempo_medidas()
                {

                    Horas = DateTime.Now.Hour;
                    Minutos = DateTime.Now.Minute;
                    Segundos = DateTime.Now.Second;
                    Milisegundos = DateTime.Now.Millisecond;

                    //Tempo_Atual = new TimeSpan(Horas, Minutos, Segundos);

                    Tempo_Atual1 = Horas.ToString() + ":" + Minutos.ToString() + ":" + Segundos.ToString() + ":" +
Milisegundos.ToString();

                }
                public void identifica_conteudo()
                {
                    switch (Id)
                    {
                        case 0:
                            conteudo = "LA0\r";
                            break;
                        case 1:
                            conteudo = "LA1\r";
```

```
                break;
            case 2:
                conteudo = "LA2\r";
                break;
            case 3:
                conteudo = "LA4\r";
                break;
            case 4:
                conteudo = "LA5\r";
                break;
            case 5:
                conteudo = "LAB\r";
                break;
            case 6:
                conteudo = "LA9\r";
                break;
            case 7:
                conteudo = "LA8\r";
                break;
            case 8:
                conteudo = "LAA\r";
                break;
            case 9:
                conteudo = "LAC\r";
                break;
        }
    }
  }
}
```

Protocolo.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO.Ports;
using System.Threading;
using System.Windows;
using System.Windows.Threading;

namespace Ranciquimis
{
  class Protocolo
  {
    SerialPort serial = new SerialPort();
    string dados;
    char[] pacote;
    double vazao_lh;
    int condut, condut_ADC;
    Dados_operação DOP_PROT;

   // public List<double> condutividade = new List<double>();

    private double read_temp;
    private double read_SP_TEMP;
    private double read_condut;
    private string condut_text;
    private double read_vazao;
```

```
private int read_contagens_adc;

public int Read_contagens_adc
{
   get { return read_contagens_adc; }
   set { read_contagens_adc = value; }
}

public double Read_vazao
{
   get { return read_vazao; }
   //set { read_vazao = value; }
}

public string Condut_text
{
   get { return condut_text; }
   //set { condut_text = value; }
}

public double Read_condut
{
   get { return read_condut; }
   //set { read_condut = value; }
}

public double Read_SP_TEMP
{
   get { return read_SP_TEMP; }
   //set { read_SP_TEMP = value; }
}

public double Read_temp
{
   get { return read_temp; }
   //set { read_temp = value; }
}

#region escrita

public void enviaserial(string data)
{
   serial = Configurations.getconfigs();


      try
      {
         if (serial.IsOpen)
         {
            byte[] hexstring = Encoding.ASCII.GetBytes(data);
            foreach (byte hexval in hexstring)
            {
               byte[] _hexval = new byte[] { hexval }; // need to convert byte to byte[] to write
               serial.Write(_hexval, 0, 1);
               Thread.Sleep(10);
            }
         }
      }
      catch(Exception e)
      {
```

```
                MessageBox.Show("Erro: Serial Port is not open."+e);
                //erro_leitura = true;
            }

        }

        #endregion

        public void ler_setpoint()
        {
            enviaserial("LZ\r");
            ler_serial();


        }
        public void Ler_Vazão()
        {


            enviaserial("LA6\r");
            ler_serial();

        }

        public void ler_temperatura_bloco()
        {
            enviaserial("LT\r");
            ler_serial();

        }


        #region Ler_serial
        public void ler_serial()
        {
            DOP_PROT = Dados_Obj.getdados_operação();
            pacote = new Char[8];
            Thread.Sleep(100);
            serial.Read(pacote, 0, 7);
            //dados = "" + pacote[0] + pacote[1] + pacote[2] + pacote[3] + pacote[4] + pacote[5] + pacote[6];
            //dados = pacote.ToString();
            dados = new string(pacote);
            //TRATAMENTO DA LEITURA: SET-POINT, TEMPERATURA E CONDUTIVIDADE

            switch (dados.Substring(0, 2))
            {
                case "TT":
                    int temp = int.Parse(dados.Substring(dados.IndexOf("TT") + 2).Remove(4));//Leitura de
Temperatura
                    read_temp = (temp/10.0);
                    break;
                case "TZ":
                    int temp2 = int.Parse(dados.Substring(dados.IndexOf("TZ") + 2).Remove(4));//Leitura de
Temperatura de Set Point
                    read_SP_TEMP = (temp2 / 10.0);
                    break;
                case "A6":
                    int vazao = int.Parse(dados.Substring(dados.IndexOf("A6") + 2).Remove(4));//Leitura da Vazão
                    //vazao_lh = 0.01312 *vazao - 2.26916;//calibração antiga
                    vazao_lh = 0.01364 * vazao - 2.50043;
```

```
    read_vazao = vazao_lh;
    break;
case "A0":
    condut = int.Parse(dados.Substring(dados.IndexOf("A0") + 2).Remove(4));//Leitura Canal 1
    read_condut = (condut)*DOP_PROT.Coef_angular_ch1;
    Read_contagens_adc = condut;
    //read_condut = (condut);
    condut_text = read_condut.ToString();
    break;
case "A1":
    condut = int.Parse(dados.Substring(dados.IndexOf("A1") + 2).Remove(4));//Leitura Canal 2
    read_condut = (condut) * DOP_PROT.Coef_angular_ch2;
    Read_contagens_adc = condut;
    //read_condut = (condut);
    condut_text = read_condut.ToString();
    break;
case "A2":
    condut = int.Parse(dados.Substring(dados.IndexOf("A2") + 2).Remove(4));//Leitura Canal 3
    read_condut = (condut) * DOP_PROT.Coef_angular_ch3;
    Read_contagens_adc = condut;
    //read_condut = (condut);
    condut_text = read_condut.ToString();
    break;
case "A4":
    condut = int.Parse(dados.Substring(dados.IndexOf("A4") + 2).Remove(4));//Leitura Canal 4
    read_condut = (condut) * DOP_PROT.Coef_angular_ch4;
    Read_contagens_adc = condut;
    //read_condut = (condut);
    condut_text = read_condut.ToString();
    break;
case "A5":
    condut = int.Parse(dados.Substring(dados.IndexOf("A5") + 2).Remove(4));//Leitura Canal 5
    read_condut = (condut) * DOP_PROT.Coef_angular_ch5;
    Read_contagens_adc = condut;
    //read_condut = (condut);
    condut_text = read_condut.ToString();
    break;
case "AB":
    condut = int.Parse(dados.Substring(dados.IndexOf("AB") + 2).Remove(4));//Leitura Canal 6
    read_condut = (condut) * DOP_PROT.Coef_angular_ch6;
    Read_contagens_adc = condut;
    //read_condut = (condut);
    condut_text = read_condut.ToString();
    break;
case "A9":
    condut = int.Parse(dados.Substring(dados.IndexOf("A9") + 2).Remove(4));//Leitura Canal 7
    read_condut = (condut) * DOP_PROT.Coef_angular_ch7;
    Read_contagens_adc = condut;
    //read_condut = (condut);
    condut_text = read_condut.ToString();
    break;
case "A8":
    condut = int.Parse(dados.Substring(dados.IndexOf("A8") + 2).Remove(4));//Leitura Canal 8
    read_condut = (condut) * DOP_PROT.Coef_angular_ch8;
    Read_contagens_adc = condut;
    //read_condut = (condut);
    condut_text = read_condut.ToString();
    break;
case "AA":
    condut = int.Parse(dados.Substring(dados.IndexOf("AA") + 2).Remove(4));//Leitura Canal 9
```

```
                read_condut = (condut) * DOP_PROT.Coef_angular_ch9;
                Read_contagens_adc = condut;
                //read_condut = (condut);
                condut_text = read_condut.ToString();
                break;
            case "AC":
                condut = int.Parse(dados.Substring(dados.IndexOf("AC") + 2).Remove(4));//Leitura Canal 10
                read_condut = (condut) * DOP_PROT.Coef_angular_ch10;
                Read_contagens_adc = condut;
                //read_condut = (condut);
                condut_text = read_condut.ToString();
                break;
            }

        }

            #endregion
    }
}
```

Dados_operação.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Ranciquimis
{
    class Dados_operação
    {
        private int amostragem_ms;
        private int tamanho_timer;
        private int tempo_max;
        private bool parametros_ok;
        private bool calibra_sensores_ok;
        private int temperatura_standard_solution;
        private double vazao_parametro;

        //variável do valor da condutividade da solução de calibração
        private double standard_solution;
        private double standard_solution_media_adc;

        private bool serial_configurada;



        //condutividade lida pelos canais que serão calibrados

        private double condutividade_calib_ch1;
        private double condutividade_calib_ch2;
        private double condutividade_calib_ch3;
        private double condutividade_calib_ch4;
        private double condutividade_calib_ch5;
        private double condutividade_calib_ch6;
        private double condutividade_calib_ch7;
        private double condutividade_calib_ch8;
        private double condutividade_calib_ch9;
        private double condutividade_calib_ch10;
```

```
  //Coeficientes angulares de calibração dos canais de condutividade
  private double coef_angular_ch1;
  private double coef_angular_ch2;
  private double coef_angular_ch3;
  private double coef_angular_ch4;
  private double coef_angular_ch5;
  private double coef_angular_ch6;
  private double coef_angular_ch7;
  private double coef_angular_ch8;
  private double coef_angular_ch9;
  private double coef_angular_ch10;

 //parametros
  public bool Serial_configurada
  {
     get { return serial_configurada; }
     set { serial_configurada = value; }
  }

  public double Vazao_parametro
  {
     get { return vazao_parametro; }
     set { vazao_parametro = value; }
  }


  public int Amostragem_ms
  {
     get { return amostragem_ms; }
     set { amostragem_ms = value; }
  }

  public int Tamanho_timer
  {
     get { return tamanho_timer; }
     set { tamanho_timer = value; }
  }

  public int Tempo_max
  {
     get { return tempo_max; }
     set { tempo_max = value; }
  }

  public bool Parametros_ok
  {
     get { return parametros_ok; }
     set { parametros_ok = value; }
  }

  public int Temperatura_standard_solution
  {
     get { return temperatura_standard_solution; }
     set { temperatura_standard_solution = value; }
  }

  public double Standard_solution
  {
     get { return standard_solution; }
```

```
    set { standard_solution = value; }
}

public double Standard_solution_media_adc
{
    get { return standard_solution_media_adc; }
    set { standard_solution_media_adc = value; }
}

public bool Calibra_sensores_ok
{
    get { return calibra_sensores_ok; }
    set { calibra_sensores_ok = value; }
}

//encapsulamentos dos coeficientes angulares dos canais
#region coeficientes_de_calibração
public double Coef_angular_ch1
{
    get { return coef_angular_ch1; }
    set { coef_angular_ch1 = value; }
}

public double Coef_angular_ch2
{
    get { return coef_angular_ch2; }
    set { coef_angular_ch2 = value; }
}

public double Coef_angular_ch3
{
    get { return coef_angular_ch3; }
    set { coef_angular_ch3 = value; }
}

public double Coef_angular_ch4
{
    get { return coef_angular_ch4; }
    set { coef_angular_ch4 = value; }
}

public double Coef_angular_ch5
{
    get { return coef_angular_ch5; }
    set { coef_angular_ch5 = value; }
}

public double Coef_angular_ch6
{
    get { return coef_angular_ch6; }
    set { coef_angular_ch6 = value; }
}

public double Coef_angular_ch7
{
    get { return coef_angular_ch7; }
    set { coef_angular_ch7 = value; }
}

public double Coef_angular_ch8
```

```
{
    get { return coef_angular_ch8; }
    set { coef_angular_ch8 = value; }
}

public double Coef_angular_ch9
{
    get { return coef_angular_ch9; }
    set { coef_angular_ch9 = value; }
}

public double Coef_angular_ch10
{
    get { return coef_angular_ch10; }
    set { coef_angular_ch10 = value; }
}
#endregion

#region condutividade_lida_canais
public double Condutividade_calib_ch1
{
    get { return condutividade_calib_ch1; }
    set { condutividade_calib_ch1 = value; }
}
public double Condutividade_calib_ch2
{
    get { return condutividade_calib_ch2; }
    set { condutividade_calib_ch2 = value; }
}
public double Condutividade_calib_ch3
{
    get { return condutividade_calib_ch3; }
    set { condutividade_calib_ch3 = value; }
}
public double Condutividade_calib_ch4
{
    get { return condutividade_calib_ch4; }
    set { condutividade_calib_ch4 = value; }
}
public double Condutividade_calib_ch5
{
    get { return condutividade_calib_ch5; }
    set { condutividade_calib_ch5 = value; }
}
public double Condutividade_calib_ch6
{
    get { return condutividade_calib_ch6; }
    set { condutividade_calib_ch6 = value; }
}
public double Condutividade_calib_ch7
{
    get { return condutividade_calib_ch7; }
    set { condutividade_calib_ch7 = value; }
}
public double Condutividade_calib_ch8
{
    get { return condutividade_calib_ch8; }
    set { condutividade_calib_ch8 = value; }
}
public double Condutividade_calib_ch9
```

```
        {
            get { return condutividade_calib_ch9; }
            set { condutividade_calib_ch9 = value; }
        }
        public double Condutividade_calib_ch10
        {
            get { return condutividade_calib_ch10; }
            set { condutividade_calib_ch10 = value; }
        }
        #endregion
    }
}
```

Dados_Obj.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Ranciquimis
{
    class Dados_Obj
    {
        private static Dados_operação dados_obj = null;

        public Dados_Obj()
        {
            dados_obj = new Dados_operação();
        }

        public static Dados_operação getdados_operação()
        {
            return dados_obj;
        }
    }
}
```

Manipula_Arquivo.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace Ranciquimis
{
    class Manipula_Arquivo
    {
        int Numero_Canais;
        int Id;
        public string Sample_Name = "";
        string User_Name = "";
        string Data_Hora = "";
        public string OBS = "";
        string Vazão;
        string Temperatura;
```

```
        string SetPoint;
        string IP = "0";
        string Index;
        string Dt;
        string Condutividade_uni;
        string Tempo_Atual1;


        public Manipula_Arquivo(int numero_canais)
        {
            this.Numero_Canais = numero_canais;
        }

        public Manipula_Arquivo(int id, string sample_name, string user_name, string data_hora, string obs, string
vazão, string temperatura, string ip, string index, string dt, string condutividade_uni, string tempo_atual1, string setpoint)
            : this(id)
        {

            this.Id = id;
            this.Sample_Name = sample_name;
            this.User_Name = user_name;
            this.Data_Hora = data_hora;
            this.OBS = obs;
            this.IP = ip;
            this.Temperatura = temperatura;
            this.Vazão = vazão;
            this.Index = index;
            this.Dt = dt;
            this.Condutividade_uni = condutividade_uni;
            this.Tempo_Atual1 = tempo_atual1;
            this.SetPoint = setpoint;
        }

        public void criar_todos_arquivos(int numero_canais)
        {
            if (!Directory.Exists(@"C:\Ranciquimis Files"))
            {
                Directory.CreateDirectory(@"C:\Ranciquimis Files\Channels");
                for (int i = 0; i < numero_canais; i++)
                {
                    FileInfo arquivo = new FileInfo(@"C:\Ranciquimis Files\Channels\Channel" + (i + 1).ToString() +
".txt");
                    FileStream fs = arquivo.Create();

                }
            }
            else
            {
                for (int i = 0; i < numero_canais; i++)
                {
                    FileInfo arquivo = new FileInfo(@"C:\Ranciquimis Files\Channels\Channel" + (i + 1).ToString() +
".txt");
                    FileStream fs = arquivo.Create();

                }
            }

        }

        public void deletar_arquivos(int numero_canais)
        {
```

```
            for (int i = 0; i < numero_canais; i++)
            {
                File.Delete(@"C:\Ranciquimis Files\Channels\Channel" + (i + 1).ToString() + ".txt");
            }
        }

        public void grava_no_arquivo()
        {


            //Vazão = base.Read_vazao;
            //Temperatura = base.Read_temp.ToString();

            FileStream fs0 = new FileStream(@"C:\Ranciquimis Files\Channels\Channel" + (Id + 1).ToString() +
".txt", FileMode.OpenOrCreate, FileAccess.Write);
            StreamWriter sw0 = new StreamWriter(fs0);

            sw0.WriteLine("Channel " + (Id + 1).ToString());
            sw0.WriteLine("Sample: " + Sample_Name);
            sw0.WriteLine("User: " + User_Name);
            sw0.WriteLine("Date: " + Data_Hora);
            sw0.WriteLine("Vazão: " + Vazão+" l/h");
            sw0.WriteLine("Block Heat: " + Temperatura + " °C");
            sw0.WriteLine("Temperature Set-Point: " + SetPoint + " °C");
            sw0.WriteLine("Indution Period (IP): " + IP + " h");
            sw0.WriteLine("Obs.: " + OBS);
            sw0.WriteLine("\n\n");
            sw0.WriteLine("Index;Time(H:M:S:ms);Measurement Time(Seconds);Conductivity(µS/cm)");
            sw0.Close();
            fs0.Close();

            StreamWriter sw1 = File.AppendText(@"C:\Ranciquimis Files\Channels\Channel" + (Id + 1).ToString() +
".txt");

            sw1.WriteLine(Index + ";" + Tempo_Atual1 + ";" + Dt + ";" + Condutividade_uni);
            sw1.Close();

        }


        public void salvar_arquivo()
        {


            Microsoft.Win32.SaveFileDialog dlg = new Microsoft.Win32.SaveFileDialog();
            // dlg.DefaultExt = ".txt";
            // dlg.Filter = "Text documents (.txt)|*.txt";
            if (dlg.ShowDialog() == true)
            {
                string destino = dlg.FileName;

                string origem = @"C:\Ranciquimis Files\Channels";
                // File.Copy(@"C:\Ranciquimis Files\Channels\Channel1.txt", filename, true);
                DirectoryCopy(origem, destino, true);

            }


        }
```

```
        private static void DirectoryCopy(string sourceDirName, string destDirName, bool copySubDirs)
        {
            DirectoryInfo dir = new DirectoryInfo(sourceDirName);
            DirectoryInfo[] dirs = dir.GetDirectories();

            if (!dir.Exists)
            {
                throw new DirectoryNotFoundException(
                    "Source directory does not exist or could not be found: "
                    + sourceDirName);
            }

            if (!Directory.Exists(destDirName))
            {
                Directory.CreateDirectory(destDirName);
            }

            FileInfo[] files = dir.GetFiles();
            foreach (FileInfo file in files)
            {
                string temppath = Path.Combine(destDirName, file.Name);
                file.CopyTo(temppath, false);
            }

            if (copySubDirs)
            {
                foreach (DirectoryInfo subdir in dirs)
                {
                    string temppath = Path.Combine(destDirName, subdir.Name);
                    DirectoryCopy(subdir.FullName, temppath, copySubDirs);
                }
            }
        }
    }
}
```

Manipula_arvivos_calibracao.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;

namespace Ranciquimis
{
    class Manipula_arqivos_calibracao
    {
        DateTime Data = DateTime.Now;
        Dados_operação DAOP_MAC;

        public void criar_arquivo()
        {
            if (!Directory.Exists(@"C:\Ranciquimis Files\Calibration"))
            {
                Directory.CreateDirectory(@"C:\Ranciquimis Files\Calibration");
                FileInfo arquivo = new FileInfo(@"C:\Ranciquimis Files\Calibration\Calibration Channel.txt");
                FileStream fs = arquivo.Create();
            }
```

```
        }
        public void deletar_arquivo()
        {
            File.Delete(@"C:\Ranciquimis Files\Calibration\Calibration Channel.txt");
        }

        public void grava_no_arquivo()
        {
            DAOP_MAC = Dados_Obj.getdados_operação();
            FileStream fs0 = new FileStream(@"C:\Ranciquimis Files\Calibration\Calibration Channel.txt",
FileMode.OpenOrCreate, FileAccess.Write);
            StreamWriter sw1 = new StreamWriter(fs0);

            sw1.WriteLine("Conductivity Calibration:");
            sw1.WriteLine("Standard Solution:\t" + DAOP_MAC.Standard_solution.ToString() + " µS/cm" + "\t
Standard Solution Temperature(°C): " + DAOP_MAC.Temperatura_standard_solution.ToString());
            sw1.WriteLine("-------------------------------------------------------------------------------");
            sw1.WriteLine("Channel 1:");
            sw1.WriteLine("Conductivity:\t" + DAOP_MAC.Condutividade_calib_ch1.ToString() + " µS/cm" );
            sw1.WriteLine("Angular Coefficient 1:\t" + DAOP_MAC.Coef_angular_ch1.ToString());
            sw1.WriteLine("-------------------------------------------------------------------------------");
            sw1.WriteLine("Channel 2:");
            sw1.WriteLine("Conductivity:\t" + DAOP_MAC.Condutividade_calib_ch2.ToString() + " µS/cm" );
            sw1.WriteLine("Angular Coefficient 2:\t" + DAOP_MAC.Coef_angular_ch2.ToString());
            sw1.WriteLine("-------------------------------------------------------------------------------");
            sw1.WriteLine("Channel 3:");
            sw1.WriteLine("Conductivity:\t" + DAOP_MAC.Condutividade_calib_ch3.ToString() + " µS/cm");
            sw1.WriteLine("Angular Coefficient 3:\t" + DAOP_MAC.Coef_angular_ch3.ToString());
            sw1.WriteLine("-------------------------------------------------------------------------------");
            sw1.WriteLine("Channel 4:");
            sw1.WriteLine("Conductivity:\t" + DAOP_MAC.Condutividade_calib_ch4.ToString() + " µS/cm");
            sw1.WriteLine("Angular Coefficient 4:\t" + DAOP_MAC.Coef_angular_ch4.ToString());
            sw1.WriteLine("-------------------------------------------------------------------------------");
            sw1.WriteLine("Channel 5:");
            sw1.WriteLine("Conductivity:\t" + DAOP_MAC.Condutividade_calib_ch5.ToString() + " µS/cm");
            sw1.WriteLine("Angular Coefficient 5:\t" + DAOP_MAC.Coef_angular_ch5.ToString());
            sw1.WriteLine("-------------------------------------------------------------------------------");
            sw1.WriteLine("Channel 6:");
            sw1.WriteLine("Conductivity:\t" + DAOP_MAC.Condutividade_calib_ch6.ToString() + " µS/cm");
            sw1.WriteLine("Angular Coefficient 6:\t" + DAOP_MAC.Coef_angular_ch6.ToString());
            sw1.WriteLine("-------------------------------------------------------------------------------");
            sw1.WriteLine("Channel 7:");
            sw1.WriteLine("Conductivity:\t" + DAOP_MAC.Condutividade_calib_ch7.ToString() + " µS/cm");
            sw1.WriteLine("Angular Coefficient 7:\t" + DAOP_MAC.Coef_angular_ch7.ToString());
            sw1.WriteLine("-------------------------------------------------------------------------------");
            sw1.WriteLine("Channel 8:");
            sw1.WriteLine("Conductivity:\t" + DAOP_MAC.Condutividade_calib_ch8.ToString() + " µS/cm");
            sw1.WriteLine("Angular Coefficient 8:\t" + DAOP_MAC.Coef_angular_ch8.ToString());
            sw1.WriteLine("-------------------------------------------------------------------------------");
            sw1.WriteLine("Channel 9:");
            sw1.WriteLine("Conductivity:\t" + DAOP_MAC.Condutividade_calib_ch9.ToString() + " µS/cm");
            sw1.WriteLine("Angular Coefficient 9:\t" + DAOP_MAC.Coef_angular_ch9.ToString());
            sw1.WriteLine("-------------------------------------------------------------------------------");
            sw1.WriteLine("Channel 10:");
            sw1.WriteLine("Conductivity:\t" + DAOP_MAC.Condutividade_calib_ch10.ToString() + " µS/cm");
            sw1.WriteLine("Angular Coefficient 10:\t" + DAOP_MAC.Coef_angular_ch10.ToString());
            sw1.WriteLine("-------------------------------------------------------------------------------");

            sw1.Close();
```

```
        }

    }
}
-----------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------
Calibra Sensores.cs
-----------------------------------------------------------------------------------------------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.Threading;
using System.Timers;
using System.Windows.Threading;

namespace Ranciquimis
{
    /// <summary>
    /// Interaction logic for Calibra_Sensores.xaml
    /// </summary>
    public partial class Calibra_Sensores : Window
    {
        int temperatura_calibração;
        private static System.Timers.Timer cTimer;
        string update, comboxitem;
        double condutividade;
        bool ok_calibra_sensores;
        string ch1_condut, ch2_condut, ch3_condut, ch4_condut, ch5_condut, ch6_condut, ch7_condut,
ch8_condut, ch9_condut, ch10_condut;
        Dados_operação DOP_CS;
        Manipula_arqivos_calibracao mac = new Manipula_arqivos_calibracao();
        double standard_solution_value;

        public Calibra_Sensores()
        {
            InitializeComponent();
            DOP_CS = Dados_Obj.getdados_operação();
            cTimer = new System.Timers.Timer(12000);
            cTimer.Elapsed += new ElapsedEventHandler(OnTimedEvent);
            cTimer.Interval = 3000;
            mac.criar_arquivo();
        }

        private delegate void UpdateUiTextDelegate(string text);
        private void OnTimedEvent(object source, ElapsedEventArgs e)
        {

            Dispatcher.Invoke(DispatcherPriority.Send, new UpdateUiTextDelegate(Atualizar_text), update);
            lertodos_canais();

        }
```

```
private void Atualizar_text(string texto)
{
   Ch1_tb.Text = ch1_condut;
   Ch2_tb.Text = ch2_condut;
   Ch3_tb.Text = ch3_condut;
   Ch4_tb.Text = ch4_condut;
   Ch5_tb.Text = ch5_condut;
   Ch6_tb.Text = ch6_condut;
   Ch7_tb.Text = ch7_condut;
   Ch8_tb.Text = ch8_condut;
   Ch9_tb.Text = ch9_condut;
   Ch10_tb.Text = ch10_condut;

}
private void Temperatura_selection(object sender, SelectionChangedEventArgs e)
{
   try
   {
            comboxitem = Temperatura_cbox.SelectedValue.ToString().Remove(0, 37);
            temperatura_calibração = int.Parse(comboxitem);
            DOP_CS.Temperatura_standard_solution = temperatura_calibração;
   }
   catch
   {

   }
}
private void Standard_solbox_keydown(object sender, KeyEventArgs e)
{
   if (e.Key == Key.Enter)
   {
     standard_solution_value = double.Parse(standard_sol_box.Text);
     DOP_CS.Standard_solution = standard_solution_value;
     Channels_groupBox.IsEnabled = true;
     cTimer.Enabled = true;

   }
}
private void Standard_solbox_lostfocus(object sender, RoutedEventArgs e)
{
   standard_solution_value = double.Parse(standard_sol_box.Text);
   DOP_CS.Standard_solution = standard_solution_value;
   Channels_groupBox.IsEnabled = true;
   cTimer.Enabled = true;
}
private void Ch1_check(object sender, RoutedEventArgs e)
{

   Calibra_Canais CC1 = new Calibra_Canais(0);
   cTimer.Stop();
   CC1.ShowDialog();

   if (CC1.calibra_status_test())
     {
      cTimer.Start();
      DOP_CS.Condutividade_calib_ch1 = CC1.condutividade_media;
      DOP_CS.Coef_angular_ch1 = CC1.coef_angular;
     }
```

```
}

private void Ch2_check(object sender, RoutedEventArgs e)
{
   Calibra_Canais CC2 = new Calibra_Canais(1);
   cTimer.Stop();
   CC2.ShowDialog();

   if (CC2.calibra_status_test())
   {
      cTimer.Start();

      DOP_CS.Condutividade_calib_ch2 = CC2.condutividade_media;
      DOP_CS.Coef_angular_ch2 = CC2.coef_angular;
   }
}

private void Ch3_check(object sender, RoutedEventArgs e)
{
   Calibra_Canais CC3 = new Calibra_Canais(2);
   cTimer.Stop();
   CC3.ShowDialog();

   if (CC3.calibra_status_test())
   {
      cTimer.Start();

      DOP_CS.Condutividade_calib_ch3 = CC3.condutividade_media;
      DOP_CS.Coef_angular_ch3 = CC3.coef_angular;
   }
}

private void Ch4_check(object sender, RoutedEventArgs e)
{
   Calibra_Canais CC4 = new Calibra_Canais(3);
   cTimer.Stop();
   CC4.ShowDialog();

   if (CC4.calibra_status_test())
   {
      cTimer.Start();

      DOP_CS.Condutividade_calib_ch4 = CC4.condutividade_media;
      DOP_CS.Coef_angular_ch4 = CC4.coef_angular;
   }
}

private void Ch5_check(object sender, RoutedEventArgs e)
{
   Calibra_Canais CC5 = new Calibra_Canais(4);
   cTimer.Stop();
   CC5.ShowDialog();

   if (CC5.calibra_status_test())
   {
      cTimer.Start();

      DOP_CS.Condutividade_calib_ch5 = CC5.condutividade_media;
      DOP_CS.Coef_angular_ch5 = CC5.coef_angular;
```

```
    }
}

private void Ch6_check(object sender, RoutedEventArgs e)
{
    Calibra_Canais CC6 = new Calibra_Canais(5);
    cTimer.Stop();
    CC6.ShowDialog();

    if (CC6.calibra_status_test())
    {
        cTimer.Start();

        DOP_CS.Condutividade_calib_ch6 = CC6.condutividade_media;
        DOP_CS.Coef_angular_ch6 = CC6.coef_angular;
    }
}

private void Ch7_check(object sender, RoutedEventArgs e)
{
    Calibra_Canais CC7 = new Calibra_Canais(6);
    cTimer.Stop();
    CC7.ShowDialog();

    if (CC7.calibra_status_test())
    {
        cTimer.Start();

        DOP_CS.Condutividade_calib_ch7 = CC7.condutividade_media;
        DOP_CS.Coef_angular_ch7 = CC7.coef_angular;
    }

}

private void Ch8_check(object sender, RoutedEventArgs e)
{
    Calibra_Canais CC8 = new Calibra_Canais(7);
    cTimer.Stop();
    CC8.ShowDialog();

    if (CC8.calibra_status_test())
    {
        cTimer.Start();

        DOP_CS.Condutividade_calib_ch8 = CC8.condutividade_media;
        DOP_CS.Coef_angular_ch8 = CC8.coef_angular;
    }

}

private void Ch9_check(object sender, RoutedEventArgs e)
{
    Calibra_Canais CC9 = new Calibra_Canais(8);
    cTimer.Stop();
    CC9.ShowDialog();

    if (CC9.calibra_status_test())
    {
        cTimer.Start();
```

```
        DOP_CS.Condutividade_calib_ch9 = CC9.condutividade_media;
        DOP_CS.Coef_angular_ch9 = CC9.coef_angular;
    }
}

private void Ch10_check(object sender, RoutedEventArgs e)
{
    Calibra_Canais CC10 = new Calibra_Canais(9);
    cTimer.Stop();
    CC10.ShowDialog();

    if (CC10.calibra_status_test())
    {
        cTimer.Start();

        DOP_CS.Condutividade_calib_ch10 = CC10.condutividade_media;
        DOP_CS.Coef_angular_ch10 = CC10.coef_angular;
    }
}

private void Ok_calibra_ss(object sender, RoutedEventArgs e)
{
    ok_calibra_sensores = true;
    DOP_CS.Calibra_sensores_ok = ok_calibra_sensores;
    cTimer.Stop();
    mac.grava_no_arquivo();
    this.Close();
}

private void Cancel_ss(object sender, RoutedEventArgs e)
{
    try
    {
        if (cTimer.Enabled)
        {
            cTimer.Stop();
            this.Close();
        }

    }
    catch (Exception ex)
    {
        MessageBox.Show("Erro"+ex);
    }

}

private void lertodos_canais()
{
        for (int i = 0; i < 10; i++)//cria todos os canais na lista
        {
            Canais canal = new Canais(i);
            condutividade = canal.ler_condut(i);

            switch (i)
            {
                case 0:
                    ch1_condut = condutividade.ToString();
                    break;
                case 1:
```

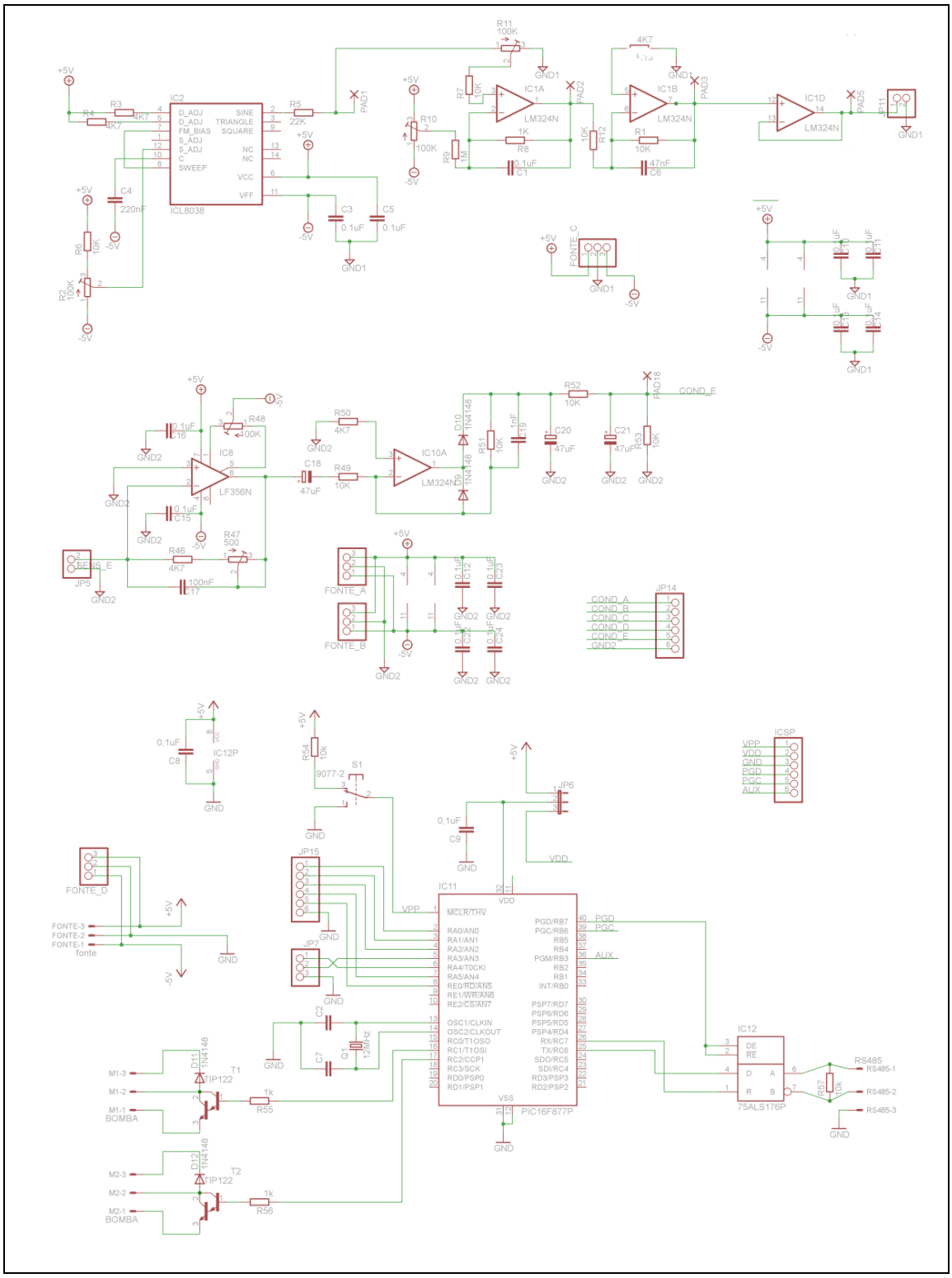
```
                        ch2_condut = condutividade.ToString();
                        break;
                    case 2:
                        ch3_condut = condutividade.ToString();
                        break;
                    case 3:
                        ch4_condut = condutividade.ToString();
                        break;
                    case 4:
                        ch5_condut = condutividade.ToString();
                        break;
                    case 5:
                        ch6_condut = condutividade.ToString();
                        break;
                    case 6:
                        ch7_condut = condutividade.ToString();
                        break;
                    case 7:
                        ch8_condut = condutividade.ToString();
                        break;
                    case 8:
                        ch9_condut = condutividade.ToString();
                        break;
                    case 9:
                        ch10_condut = condutividade.ToString();
                        break;
                }
                Thread.Sleep(250);
            }
        }

        private void Conductivity_Calibration_Closing(object sender, System.ComponentModel.CancelEventArgs
e)
        {
            cTimer.Stop();
        }


    }
}
-----------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------
Calibra_Canais.cs
-----------------------------------------------------------------------------------------------------------
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Shapes;
using System.Timers;
using System.Threading;

namespace Ranciquimis
```

```
{
    /// <summary>
    /// Interaction logic for Calibra_Canais.xaml
    /// </summary>
    public partial class Calibra_Canais : Window
    {
        System.Windows.Threading.DispatcherTimer dispatcherTimer = new
System.Windows.Threading.DispatcherTimer();

        public int Id;
        double soma = 0, soma_adc = 0;
        public double condutividade_media, refval, condutividade_media_adc;
        bool ref_ok, abilita_ctimer;
        // int standard_solution_temperature;
        double condutividade_elementos, condutividade_elementos_adc;

        public double coef_angular;
        const int tam_vetor = 10;
        double[] Vetor = new double[tam_vetor];
        double[] Vetor_adc = new double[tam_vetor];
        int i = 0;

        Dados_operação DOP_CC;

        public Calibra_Canais(int id)
        {
            InitializeComponent();
            this.Id = id;

            DOP_CC = Dados_Obj.getdados_operação();
            dispatcherTimer.Tick += new EventHandler(dispatcherTimer_Tick);
            dispatcherTimer.Interval = new TimeSpan(0, 0, 1);
            Ref_condut_tb.Text = DOP_CC.Standard_solution.ToString();
            //mac.criar_arquivo(Id);

        }


        private void Play_calib_clk(object sender, RoutedEventArgs e)
        {

            //if (ref_ok == true)
            //{
                dispatcherTimer.Start();

            //}

        }

        private void Cancel_calib_clk(object sender, RoutedEventArgs e)
        {
            dispatcherTimer.Stop();
            abilita_ctimer = true;
            this.Close();

        }

        private void OK_calib_clk(object sender, RoutedEventArgs e)
        {
```

```
            dispatcherTimer.Stop();

            // mac.grava_no_arquivo(Id, DOP_CC.Temperatura_standard_solution, condutividade_elementos,
condutividade_media, refval, coef_angular);

            abilita_ctimer = true;
            this.Close();
        }

        private void dispatcherTimer_Tick(object sender, EventArgs e)
        {
            // code goes here

            loading_bar.Value++;
            if (loading_bar.Value<=loading_bar.Maximum)
            {
                Play_calib_btn.Content = "Stop";
                leitura_condutividade_calib();

            }
            if (loading_bar.Value == loading_bar.Maximum)
            {
                dispatcherTimer.Stop();
                Play_calib_btn.Content = "Play";
                ch_read_tb.Text = condutividade_media.ToString();
                coefficient_tb.Text = calula_coeficiente().ToString();
                loading_bar.Value = 0;
            }

        }

        private double leitura_condutividade_calib()
        {
            Canais ch = new Canais(Id);


            if (i<10)
            {
                condutividade_elementos = ch.ler_condut(Id);
                condutividade_elementos_adc = ch.ler_condut_adc(Id);

                Vetor[i] = condutividade_elementos;
                Vetor_adc[i] = condutividade_elementos_adc;

                soma += condutividade_elementos;
                soma_adc += condutividade_elementos_adc;
            }

            if (i==10)
            {
                i = 0;
            }

            condutividade_media = soma / loading_bar.Maximum;
            DOP_CC.Standard_solution_media_adc = soma_adc / loading_bar.Maximum;

            return condutividade_media;
        }

        public double calula_coeficiente()
```

```
        {
          // coef_angular = condutividade_media / refval;
            coef_angular = DOP_CC.Standard_solution / DOP_CC.Standard_solution_media_adc;
            return coef_angular;
        }

        public bool calibra_status_test()
        {

            if (abilita_ctimer)
            {
                abilita_ctimer = true;
            }
            else
            {
                abilita_ctimer = false;
            }

            return abilita_ctimer;
        }

        private void Calibration_Closing(object sender, System.ComponentModel.CancelEventArgs e)
        {
            dispatcherTimer.Stop();
            abilita_ctimer = true;
        }

    }
}
```

Anexo A – Diagrama do Circuito de Condicionamento de Sinal



Anexo B – Especificações do Fluxímetro Dwyer

Series GFM Gas Mass Flow Meters combine a straight tube sensor with a restrictor flow element to provide high accuracy and repeatability. Flow rates are virtually unaffected by temperature and pressure variations. Actual gas flow is displayed in engineering units on a 3-digit, 90° tiltable LCD readout. Units can be used with Series GFT Flow Totalizer for applications requiring totalization. Series GFM includes a NIST traceable certificate.

	Model		
Flow Range	Aluminum	SS	Process Connector Compression Fitting
0-10 mL/m	**GFM-1101*** **$648.00Ⓝ**	**GFM-2101*** **1077.00Ⓝ**	1/4″
0-20 mL/m	**GFM-1102*** **648.00Ⓝ**	**GFM-2102*** **1077.00Ⓝ**	1/4″
0-50 mL/m	**GFM-1103*** **648.00Ⓝ**	**GFM-2103*** **1077.00Ⓝ**	1/4″
0-100 mL/m	**GFM-1104*** **648.00Ⓝ**	**GFM-2104*** **1077.00Ⓝ**	1/4″
0-200 mL/m	**GFM-1105*** **648.00Ⓝ**	**GFM-2105*** **1077.00Ⓝ**	1/4″
0-500 mL/m	**GFM-1106*** **648.00Ⓝ**	**GFM-2106*** **1077.00Ⓝ**	1/4″
0-1000 mL/m	**GFM-1107*** **648.00Ⓝ**	**GFM-2107*** **1077.00Ⓝ**	1/4″
0-2 L/min	**GFM-1108*** **648.00Ⓝ**	**GFM-2108*** **1077.00Ⓝ**	1/4″
0-5 L/min	**GFM-1109*** **648.00Ⓝ**	**GFM-2109*** **1077.00Ⓝ**	1/4″
0-15 L/min	**GFM-1111*** **829.00Ⓝ**	**GFM-2111*** **1367.00Ⓝ**	1/4″
0-30 L/min	**GFM-1131*** **829.00Ⓝ**	**GFM-2131*** **1367.00Ⓝ**	1/4″
0-50 L/min	**GFM-1133*** **829.00Ⓝ**	**GFM-2133*** **1367.00Ⓝ**	1/4″
0-100 L/min	**GFM-1142*** **1018.00Ⓝ**	**GFM-2142*** **1562.00Ⓝ**	3/8″
0-200 L/min	**GFM-1143*** **1232.00Ⓝ**	**GFM-2143*** **1895.00Ⓝ**	3/8″
0-500 L/min	**GFM-1144*** **1314.00Ⓝ**	**GFM-2144*** **2021.00Ⓝ**	1/2″
0-1000 L/min	**GFM-1145*** **1477.00Ⓝ**	**GFM-2145*** **2272.00Ⓝ**	3/4″

SPECIFICATIONS
Service: Clean gases compatible with wetted parts.
Wetted Materials:
GFM-1XXX: Anodized aluminum, brass, 316 SS and fluoroelastomer O-rings;
GFM-2XXX: 316 SS and fluoroelastomer O-rings.
Accuracy: ±1.5% FS including linearity over 59 to 77°F (5 to 25°C) and 5 to 60 psia (0.35 to 4 bar).
Repeatability: ±0.5% of full-scale.
Response Time: 2 s to within ±2% of actual flow.
Output: Linear 0 to 5 VDC and 4 to 20 mA.
Max. Particulate Size: 5 microns.
Temperature Limits: 32 to 122°F (0 to 50°C).
Power Supply: ±12 VDC.
Process Connections: 1/4″ compression fitting for flow rates ≤50 L/m; 3/8″ for 100 and 200 L/m; 1/2″ for 500 L/min; 3/4″ for 1000 L/min.
Pressure Limits: 500 psig (34.5 bar).
Leak Integrity: 1 x 10^{-7} sccs of helium.
Display: 90° tiltable, 3-1/2 digit.
Agency Approvals: CE.

ACCESSORIES
For Series GFM Gas Mass Flowmeters
Model GFM-110P, 110V Power Supply . . . . . **$52.50Ⓑ**
Model GFM-220PE, 220V Power Supply . . . . . **77.00Ⓑ**
Model GFM-CBL4, 3´ cable for 4 to 20 mA output . . . . . **37.25Ⓑ**
Model GFM-CBL5, 3´ cable for 0 to 5 VDC output . . . . . **37.25Ⓑ**
Ⓑ Items are subject to Schedule B discounts.

Anexo C – Especificações do Termômetro MT-455 Minípa

TERMÔMETRO DIGITAL 2 CANAIS

MODELO: MT-455

CARACTERÍSTICAS

- Display: 3 1/2 dígitos, 2000 Contagens (com iluminação)
- Taxa de Amostragem: 2.5 vezes/s
- Indicação de Polaridade: Automática
- Indicação de Sobrefaixa: OL
- Indicação de Bateria Fraca:
- Mudança de Faixa: Automática
- Temperatura em °C ou °F
- Tipo de Termopar: K
- Desligamento Automático: Aprox. 30 minutos ou desabilitado
- Data Hold
- Modo Relativo
- Registro MAX / MIN
- Ajuste de Offset
- Função T1 - T2
- Ambiente de Operação: 0°C a 30°C (Umidade Relativa RH ≤ 80%), 30°C a 40°C (RH ≤ 75%) e 40°C a 50°C (RH ≤ 45%)
- Ambiente de Armazenamento: -20°C a 60°C, RH ≤ 80%
- Altitude de Operação: Até 2000m
- Uso Interno
- Alimentação: Quatro baterias de 1.5V (AAA)
- Duração da Bateria: Aprox. 200h (baterias de carbono-zinco)
- Dimensões: 160(A) x 83(L) x 38(P)mm
- Peso: Aprox. 240g (incluindo bateria)

TEMPERATURA

- Faixas: -200°C ~ 1372°C, -328°F ~ 1999°F
- Precisão: -60°C ~ 1372°C ± (0.1%+1.0°C)
  -200°C ~ -60°C ± (0.1%+2.0°C)
  -76°F ~ 1999°F ± (0.1%+2.0°F)
  -328°F ~ -76°F ± (0.1%+4.0°F)
- Resolução: 0.1°C (-200°C~200°C)
  1°C (200°C~1372°C)
  0.1°F (-200°F~200°F)
  1°F(-328°F~-200°F ou 200°F~1999°F)

* Nota: A precisão especificada não inclui o erro do termopar.

PONTA TERMOPAR

- Tipo de Transdutor: Termopar Tipo K (NiCr-NiAl)
- Faixa: -40°C ~ 204°C (-40°F ~ 399°F)
- Precisão: ±0.75% ou ±2.2°C (±1.4°F)
- Comprimento do Cabo: Aprox. 1m