UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS
DEPARTAMENTO DE ENGENHARIA DE MATERIAIS

Lincoln Burckhart Santos Silva

Criação de equipamento para otimização de processos de secagem de cerâmicas refratárias moldáveis.

São Carlos
2015

Lincoln Burckhart Santos Silva

Criação de equipamento para auxílio de otimização de processos de secagem de cerâmicas refratárias.

Trabalho de conclusão de curso apresentado à Escola de Engenharia de São Carlos da Universidade de São Paulo,

Orientador: Prof. Dr. Rafael Salomão

São Carlos
2015

Burckhart, Lincoln
B943c Criação de equipamento de otimização de processos de secagem de cerâmicas refratárias moldáveis. / Lincoln Burckhart; orientador Rafael Salomão. São Carlos, 2015.

Monografia (Graduação em Engenharia De Materiais e Manufatura) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2015.

1. Monitoramento. 2. Programação. 3. Massa. 4. Refratários. 5. Variação. I. Título.

Formulário para relatório de defesa de TCC

Relatório de defesa pública de Trabalho de Conclusão de Curso da Escola de Engenharia de São Carlos, da Universidade de São Paulo.

Aluno	LINCOLN BURCKHART SANTOS SILVA	No. USP: 7170820
Orientador ou resp. pela disciplina	PROF. DR. RAFAEL SALOMÃO	No. USP: 626500
Titulo do TCC	"CRIAÇÃO DE EQUIPAMENTO PARA OTIMIZAÇÃO DE PROCESSOS DE SECAGEM DE CERÂMICAS REFRATÁRIAS MOLDÁVEIS"	
Curso ou Ênfase	Engenharia de Materiais e Manufatura	
Disciplina	SMM0325 Trabalho de Conclusão de Curso	
Local da defesa:	SMM/EESC/USP	Data de defesa: 16/11/2015

Após declarada aberta a sessão, o(a) Sr(a) Presidente passou a palavra aos examinadores para as devidas arguições. Em seguida, a Comissão Julgadora proclamou o resultado:

Membros da Comissão Julgadora	Vínculo	Sigla Unidade	Nota
LIGIA M. M. COSTA	SMM/EESC	PESQUISADORA	8,0
EDUARDO B. FERREIRA	SMM/EESC	PROFESSOR	7,5
RAFAEL SALOMÃO	SMM/EESC	PROFESSOR	8,5

Média = 8,0			
Resultado final	(X) Aprovado	( ) Necessita de ajustes	( ) Reprovado

Observações da Comissão Julgadora

Eu, Priscila Gramato, Auxiliar Acadêmico, lavrei o presente relatório que assino com os(as) Senhores(as). São Carlos, 18/11/2015

Membro 1	Membro 2
Membro 3	Membro 4
Orientador(a) ou responsável pela disciplina	Auxiliar Acadêmico

RESUMO

BURCKHART, L. **Criação de equipamento de otimização de processos de secagem de cerâmicas refratárias moldáveis.** Número de folhas do trabalho 25f. Monografia (Enegnharia de Materiais e Manufatura) – Departamento de Engenharia de Materiais, Escola de Engenharia de São Carlos - Universidade de São Paulo, São Carlos, 2015.

Neste trabalho foi realizado o desenvolvimento de um software e a criação de um novo equipamento termogravimétrico para auxiliar o monitoramento de materiais cerâmicos. Tal equipamento é composto por uma balança analítica conectada a um forno tipo mufla (até 900°C) onde uma amostra de até 100 g pode ser inserida. O monitoramento dos dados de perda de massa é comandado por programa de computador utilizando a linguagem C# com interface gráfica utilizando-se do Microsoft Visual Studio 2015. Com isso, há detecção de variações de massa com o tempo, em função da temperatura da amostras. Dessa forma, espera-se obter informações que permitam que o material seja aquecido de forma mais eficiente e sem provocar danos que outrora ocorreria em um processo similar sem monitoramento.

Palavras-chave: Monitoramento. Programação. Massa. Refratários. Variação.

ABSTRACT

BURCKHART, L. **Design of equipment for the optimization of castable refractory drying.** Number of paper sheets: 25. Monograph (Materials and Manufacturing Engineering) - Department of Materials Engineering, São Carlos Engineering School, University of São Paulo, São Carlos, 2015.

This final paper addresses the development of a software and a thermogravimetric equipment for aiding of monitoring ceramic materials. One of them is the design of themogravimetric equipment for the monitoring of thermal treatments of cermics. Such equipment combines an analytical weighing scale connected to a muffler furnace in which a sample that weighs over 100g can be placed. The mass loss data are monitored by a computer program developed in Visual Studio 2015 enviroment and C# language with a graphic interface, so that the mass variations with time and also in the samples´ temperature can be detected. The material is expected to be heated more efficiently and without the possible damage risks caused by a similar process with no monitoring.

Keywords: Monitoring. Programming. Mass. Refractories. Variation.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Em muitos processos industriais, é necessário o aquecimento de um determinado material a temperaturas extremas, geralmente a ponto de fundi-las, como ocorre na siderurgia por exemplo. O receptáculo na qual este material é abrigado deve resistir a temperaturas muito superiores à do processamento, já que ele é uma ferramenta vital no ciclo de produção. Não pode ser perdido por fusão, rompimento por tensões internas geradas pela temperaturas, entre outras razões. Para isso, são utilizadas as chamadas cerâmicas refratárias, que por definição são uma classe de materiais que podem suportar temperaturas elevadas sem fundir ou deformar (SURENDRANATHAN, 2015 p.3).

Entretanto, essas cerâmicas exigem técnicas especiais no seu processamento, como, por exemplo, tipos específicos de pós em misturas cujas proporções devem ser bem avaliadas para otimizar seu empacotamento e resistência mecânica. Um dilema pode surgir nas etapas de secagem e aquecimento inicial dessas cerâmicas, necessárias para a remoção da água utilizada na mistura e hidratação dos ligantes: altas taxas de aquecimento, teoricamente, poupariam tempo de equipamento ocioso, porém ao fazer isso, há um grande risco de desintegração interna do material. Isso advém do fato da pressão do vapor d'água contido nos poros do interior do material em algum momento pode se tornar superior à sua resistência mecânica. Para compensar esse tipo de problema, pode-se simplesmente reduzir a taxa de aquecimento da cerâmica, porém isso pode tornar o processo oneroso pelo gasto energético envolvido. Por isso, uma solução foi proposta pelo grupo de pesquisa "Soluções Integradas em Manufatura e Materiais Cerâmicos". Desenvolver um equipamento termogravimétrico para monitorar esse processo. Daí foi criada a ideia de otimização dos processos de secagem de cerâmicas refratárias com a ajuda de um equipamento que consiste de uma balança acoplada a um forno com uma conexão a um computador via porta serial. Neste, existiria um programa que monitoraria a massa da cerâmica pelo tempo. Isso serviria para uma melhor análise de comportamento de secagem e por fim auxiliaria na criação num futuro próximo de um controle de taxa de aquecimento automatizado, otimizando o processo.

O objetivo deste trabalho foi abordar os conhecimentos existentes de fenômenos de secagem no interior das cerâmicas refratárias, e as técnicas de melhorias desse processo, detalhar o equipamento, coletar dados através do programa deste e analisar os resultados para avaliar como a otimização do processo seria relaizada no futuro.

1.1 FUNDAMENTAÇÃO TEÓRICA

Neste trabalho, dois conceitos serão fundamentais: o de termogravimetria e o de cerâmicas refratárias.

A termogravimetria consiste em um aparelho que mede a variação de massa pelo tempo. Isso é alcançado com um forno na qual dentro dele há um termopar para medir a temperatura, uma balança de precisão para medir a massa, e instrumentos auxiliares para coleta de dados obtidos pelo termopar e pela balança. Sua utilidade reside em estudar a cinética dos fenômenos de decomposição quantitativa dos materiais. (WUNDERLICH, B, 2005)

Cerâmicas refratárias são compostas por uma combinação de materiais naturais e sintéticos com alta resistência térmica. Geralmente eles são compostos de partículas refratárias de diferentes tamanhos misturadas com ligantes hidráulicos. A sua utilização mais comum é em altos fornos de indústrias siderúrgicas, petroquímica, cimento e cal.

Os refratários industriais são fornecidos em sua maioria na forma de tijolos em tamanhos e formatos padronizados. Porém outras formas comerciais incluem tubos, moldáveis, folhas, tecidos, fitas. Os refratários básicos são a base de MgO, cujas principais matérias-primas envolvidas são à base de argila, areias, magnesitas, cromo-magnesita, magnesita-cromo dolomitas e espinélios. (BENGISU, M., 2001.). Também existem os refratários ácidos, que são a base de SiO_2. Eles resistem, respectivamente, a escórias básicas e ácidas. Há também os refratários neutros, são relativamente inertes a ambas as escórias ácidas e neutras. Carbono, cromita e fosterita pertencem a essa classe.

As propriedades inerentes mais importantes das cerâmicas refratárias são o coeficiente de expansão térmica, a condutividade térmica, resistências ao choque térmico, resistência à escória, resistência mecância à frio e à quente, porosidade e densidade aparente. A expansão térmica baixa é desejável na maioria dos casos, já que a maior utilização é nos altos-fornos de indústrias, onde ocorre frequentes ciclos de aquecimento e resfriamento.

Os passos mais comuns em produzir cerâmicas refratárias convencionais são: moagem das matérias-primas, mistura, moldagem, secagem e aquecimento inicial, que será a abordagem principal do documento, e queima.

O esmagamento e trituração é realizado como na Figura 1 a seguir. Esses processos podem ser feitos a seco ou via úmida. Os materiais então são misturados com aditivos, como um ligante, que é um lubrificante que serve para soltar a cerâmica processada do molde após o seu processamento. Também pode ser utilizado um agente para melhorar a mistura, um plastificador para tornar a mistura mais plástica e uma defloculador para mudar a carga elétrica

das partículas, forçando-as a se repelirem, ao invés de se atraírem. Após esse processo, ocorre a prensagem – ou compactação - das partículas, na qual o pó formado pela trituração da matéria-prima é prensado dentro de uma matriz definida. Assim como na etapa anterior, ela pode ser a seco ou pode ser umidificada. Com isso, forma-se uma peça de formato definido com o material.

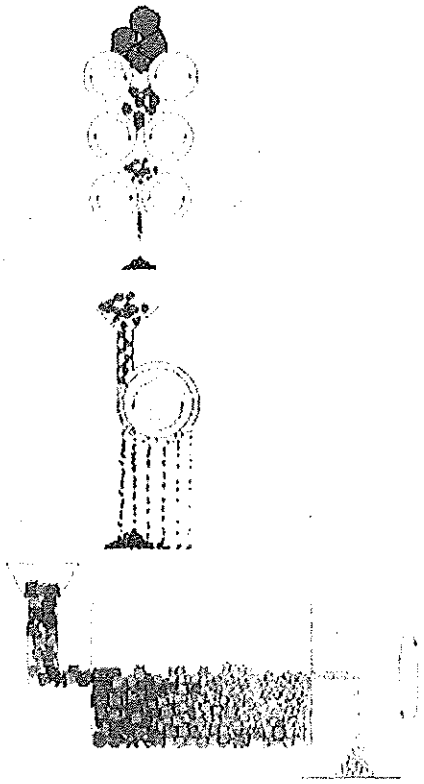


Figura 1 – Esquema de trituração de matérias-primas cerâmicas. Fonte: LESKO (2008, p 216.)

Em seguida, inicia-se a secagem, um dos processos mais importantes de todo o processamento. Na secagem, quaisquer resquícios de água que ainda existam são retirados do interior dos poros e canais. Porém, realizar esse processo sem danificar a estrutura do material pode ser um processo delicado e demorado. Para a melhor compreensão do que ocorre durante o fenômeno da secagem, deve-se abordar os chamados estágios da secagem. A secagem não ocorre apenas de forma contínua, mas comumente em vários estágios bem definidos ao longo do processo.

O primeiro estágio é a evaporação. A evaporação é comumente definida como um processo de mudança de fase que ocorre na interface entre uma fase líquida e uma fase gasosa, na qual a pressão do vapor líquida (Pv) é menor qie a pressão total do gás (Pgas) naquele ambiente. Por ser um processo interfacial, a evaporação pode ser melhorada aumentando-se a umidade e/ou os gradientes de temperatura entre as fases líquida e gasosa. Os métodos mais comuns para aumentar a taxa de evaporação consistem em aumentar a Pv (aumentando a

temperatura do sistema), ou diminuir P_A (diminuindo a umidade relativa do sistema circulando ar ou reduzindo a pressão ambiente (Pgas)). A força motriz da evaporação é o gradiente entre Pv e a pressão parcial de vapor d´água no existente no ambiente (P_A). Embora P_A dependa do quanto de vapor d´água esteja presente, Pv é dependente apenas da temperatura T, e pode ser estimada usando a equação de Antoine a seguir:

$$P_v = \exp\left(A - \frac{B}{T + C}\right) \qquad \text{Equação 1}$$

Onde Pv é dado em Pascal e T é dado em Kelvin e A, B e C são cosntantes empíricas adimensionais (para a água, A = 23,224 ,B = 3841,22 e C = -45,0). O alcance da validade dessa equação é entre 0°C e 374°C, quando o ponto crítico da água é alcançado e o equilíbrio entre líquido e vapor já não existe mais. Acima desta temperatura, ou apenas vapor existe no ambiente, a pressão na fase gasosa é linearmente dependente e pode ser estimada pela lei dos gases ideais. A secagem por evaporação em condições ambientes não causa um aumento de pressão que possa provocar uma ruptura de um material cerâmico poroso. Porém, em casos de materiais mais densos, pode tornar-se mais dispendioso em termos de tempo gasto, pois a remoção de água através da rede porosa é muito limitada por mecanismos de transferência de massa e calor.

Entretanto apenas mencionar os dois tipos de secagem e os mecanismos intrínsecos envolvidos não basta. Para uma possível otimização do processo de secagem, é necessário um maior aprofundamento dos conhecimentos existentes da secagem da cerâmica. Daí, é lançado mão do que é conhecido atualmente, os chamados estágios de secagem.

O primeiro estágio caracteriza-se pela evaporação da água de forma adiabática através da superfície do material. A taxa de evaporação é muito influenciada pelas condições externas, como a temperatura e a pressão parcial de vapor. Em um ambiente isotérmico, a taxa de evaporação permanece constante enquanto as forças de transporte existente mantenham o fluxo de água dos poros até a superfície do corpo. Porém, enquanto o processo de secagem prossegue, a frente de evaporação eventualmente alçanca até os vazios existentes entre as partículas. Isso ocorre pois o transporte de água, incialmente existente por causa do fluxo de fluidos, agora ocorre por causa da difusão de vapor por caminhos cada vez maiores. Então, esses dois efeitos entram em equilíbrio no primeiro pico da Figura 2, por volta de 50°-60°C. A Figura 2 mostra a evolução da taxa de secagem de concretos refratários acompanhada por termogravimetria.

Nota-se que a intensidade e a duração dos estágios de secagem são fortemente afetadas pela velocidade de aquecimento.

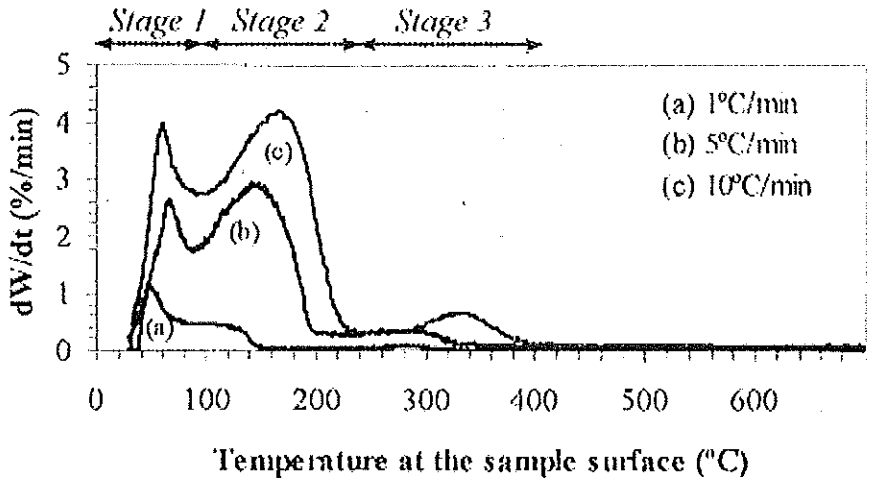


Figura 2 – Gráfico demonstrativo para os estágios de secagem. (INNOCENTINI, MURILO D. M.; CARDOSO, FÁBIO A.; AKYIOSHI, MARIO M.; PANDOLFELLI, VICTOR C., 2003)

Ao aquecer ainda mais o material, a temperatura na interface líquido-vapor acaba alcançando o ponto de ebulição, e o segundo estágio ocorre com a ebulição da água livre no sistema. O segundo estágio é a ebulição que ocorre quando a mudança de estado físico da água chega a tal ponto que a pressão de vapor d´água alcança a pressão do gás externo ($Pv \geq Pgas$). Neste regime, as bolhas de vapor são rapidamente formadas de dentro do líquido. Em sistemas aquosos abertos, $Pv = Pgas = 1$ atm, a pressão de equilíbrio ocorre a 100°C, conhecido como o ponto de ebulição da água. Mas há algumas situações na qual os sitemas são confinados, e o vapor d´água é gerado mais rapidamente do que é liberado do sistema, criando um acúmulo de pressão que acaba por gerar uma nova temperatura de ebulição e nova pressão de vapor em um novo estado de equilíbrio. Se isso ocorrer, esse acúmulo acaba por se tornar superior ao da resistência interna do material cerâmico. (INNOCENTINI, MURILO D. M.; CARDOSO, FÁBIO A.; AKYIOSHI, MARIO M.; PANDOLFELLI, VICTOR C., 2003). Neste caso, a geração de vapor é feita pelo próprio aumento da temperatura da peça, e a pressão do vapor torna-se a principal em movimentar o líquido para fora dela. Daí forma-se um segundo pico, onde há um equilíbrio entre a taxa de eliminação de vapor e a redução da frente de secagem até o centro do material.

Já no terceiro estágio, a água presente no sistema já não desempenha um fator decisivo no processo, mas sim a formação de compostos hidratados formados tanto no processamento

inicial de matéria-prima quanto no próprio processo de secagem. (SALOMÃO, R; PANDOLFELLI, VICTOR C, 2004). Este estágio pode se tornar significativo se quantidades elevadas de ligantes forem utilizadas.

Os estágios de aquecimento podem ser conhecidos e possuírem um explicação bem fundada, mas seu controle ainda não resolve totalmente o problema da otimização da secagem. Uma solução implementada com sucesso é aditivando a matéria-prima com fibras de polímeros como o polipropileno. Neste caso, as fibras acabam gerando canais entre os poros e os caminhos permeáveis já existentes na peça. Com isso, elas acabam por afetar a permeabilidade e o comportamento de secagem do material. Testes experimentais comprovam que de fato o acréscimo de fibras no material aumentam a permeabilidade da cerâmica após o aquecimento, mas não conseguem fornecer um dado mais preciso sobre o comportamento de secagem. Combinado com o acréscimo de fibras, a otimização do processo de secagem pode ser uma realidade se houver um controle computacional na qual há um monitoramento da perda de massa em função do tempo. Dessa forma, pode-se assim detectar a variação dela também de acordo com o tempo e temperatura, para que o sistema perceba o início e o fim do picos dos dois primeiros estágios de secagem. (INNOCENTINI, MURILO D. M.; CARDOSO, FÁBIO A.; AKYIOSHI, MARIO M.; PANDOLFELLI, VICTOR C., 2003).

2 PROCEDIMENTO EXPERIMENTAL

O equipamento de ensaios termogravimétricos desenvolvido é composto por um forno tipo mufla que está contectado com uma amostra ligada a uma balança analítica conectada a um computador por meio de um cabo de comunicação serial. Neste computador, o programa coleta as leituras de massa de acordo com o tempo. Por exemplo, no t = 0, a massa será de "x" g, no tempo t = 30, a massa será "x-0,005" g, e assim sucessivamente. Nos primeiros estágios de desenvolvimento, foram criados os suportes e apoios para o sistema, bem como o posicionamento do porta-amostras no forno.

Após este estágio inicial de desenvolvimento, testes foram realizados com o programa para determinar quais melhorias devem ser implementadas para que as condições necessárias para um bom ensaio termogravimétrico fossem alcançadas. Por exemplo, alguns controles adicionais poderiam ser implementados, ou alterações na interface em termos de visualização seriam mais bem vindas ao grupo.

Por fim, no estágio final foram realizados repetidos ensaios termogravimétricos com a utilização do programa já finalizado e preparado. Com isso, uma sequência de pontos com a coordenada "x" sendo o tempo em segundos, e "y", a massa da amostra em gramas, foi gerada na forma de um arquivo de texto, com a possibilidade de cópia para o Microsoft Excel, que por sua vez acaba produzindo os gráficos de massa e taxa de perda de massa.

2.1 MATERIAIS

Os materiais utilizados neste desenvolvimento e criação final do equipamento (Figura 3) foram os seguintes:

- Forno tipo mufla marca EDG modelo 3000 (Figura 4);
- Balança semi-analítica marca Gehaka modelo BK300 (Figura 5);
- Computador com cabo serial de protocolo RS232 (Figura 6);
- Microsoft Visual Studio 2015 (Figura 7);
- 3 amostras de cerâmicas refratárias compostas por alumina calcinada e cimento de aluminato de cálcio (CAC) de formato cilíndrico, com 15 mm de altura de 15 mm de diâmetro, sendo:
  - 1 amostra com 90% de Al_2O_3 calcinada e 10% de cimento de aluminato de cálcio (CAC) (EL70, Elfusa, Brasil)
  - 1 amostra com 80% de Al_2O_3 calcinada e 20% de cimento de aluminato de cálcio (CAC) (AlOOOSG, Almatis, EUA)
  - 1 amostra com 66% de Al_2O_3 calcinada e 34% de cimento de aluminato de cálcio (CAC)

Todas as amostras contiveram 22% de massa de água adicionada no momento da mistura.

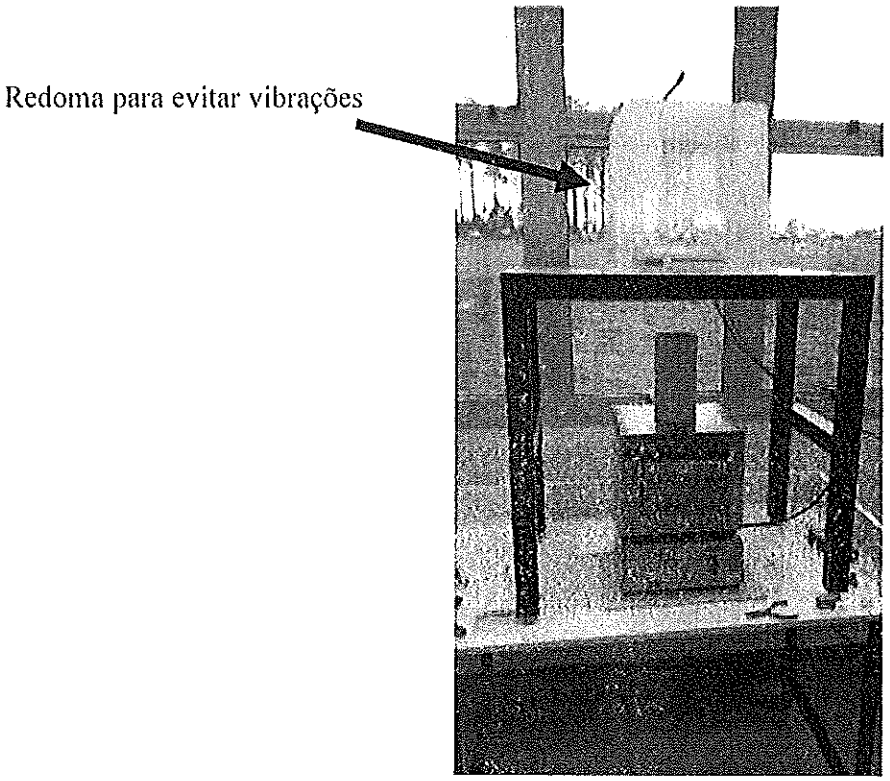


Figura 3 – Panorama do equipamento: No topo, a balança, conectada ao computador pelo cabo serial. Nela também é suspensa a amostra dentro de um cadinho que está dentro do forno, que está logo abaixo.

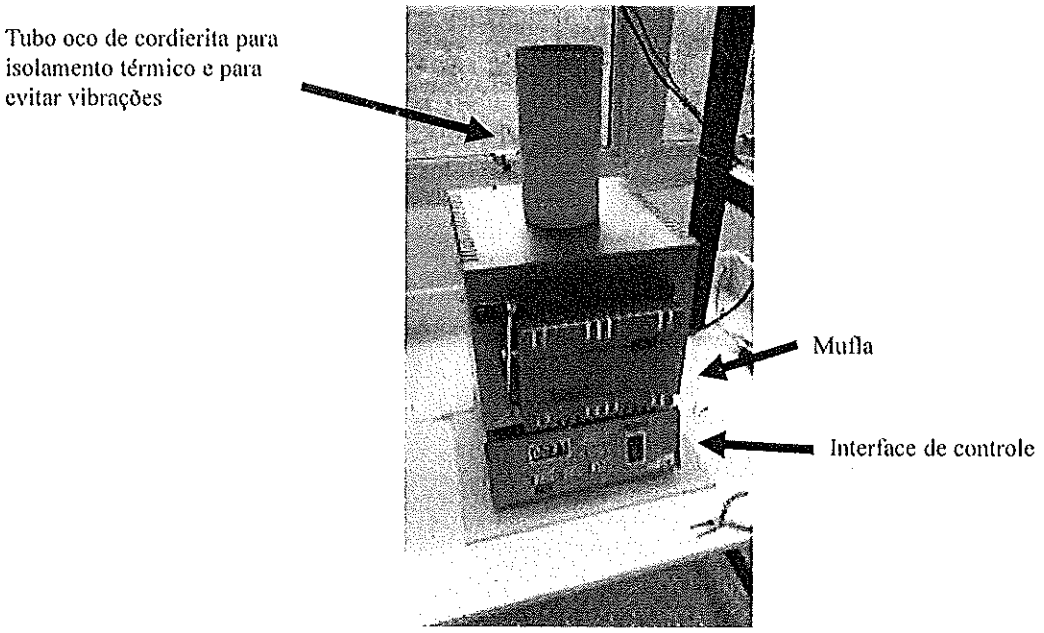


Figura 4 – Forno tipo mufla EDG 3000

Figura 5 – Balança semi-analítica Gehaka BK300.

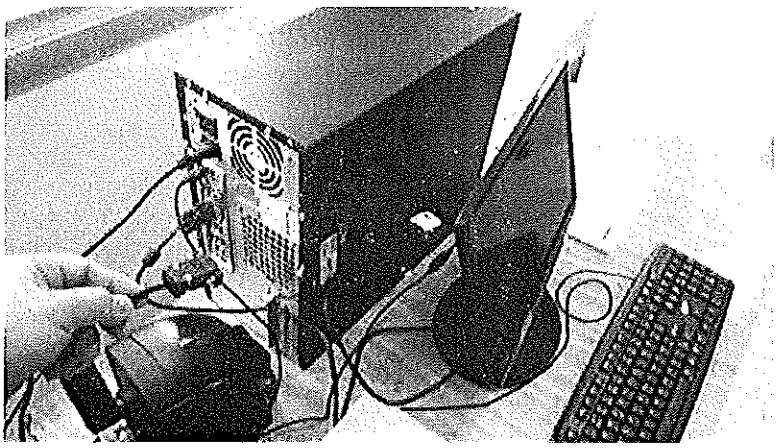


Figura 6 – Computador e cabo serial RS232, que conecta-se à balança.

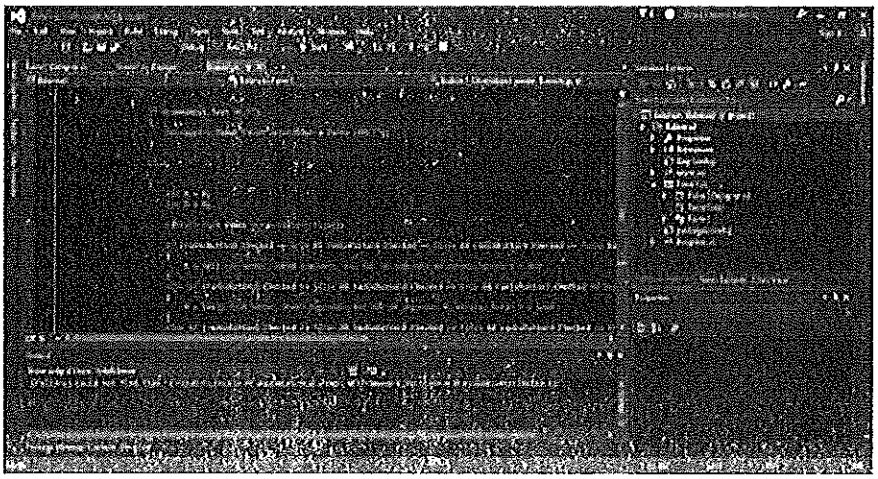


Figura 7 – Panorama da interface do Microsoft Visual Studio 2015

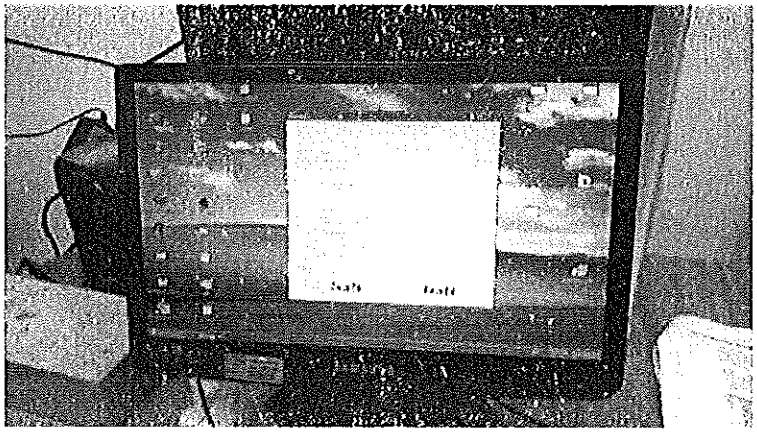


Figura 8 -- Primeira versão do programa de monitoramento em funcionamento.

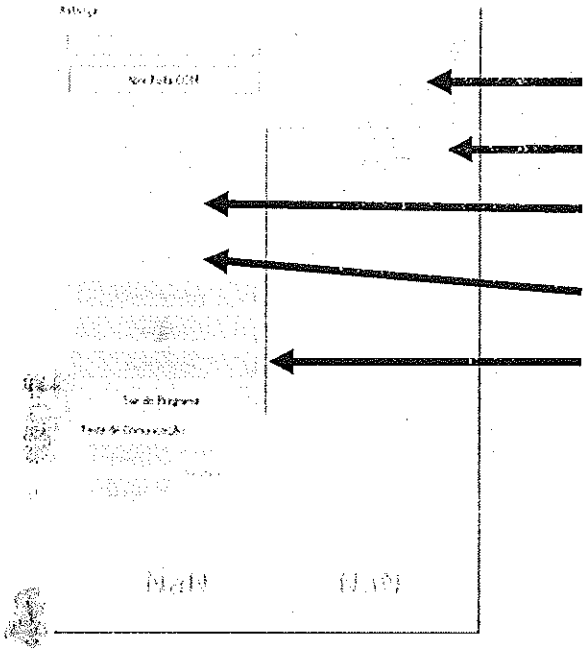


Figura 9 – Interface do programa de monitoramento acionado pelo Visual Studio.

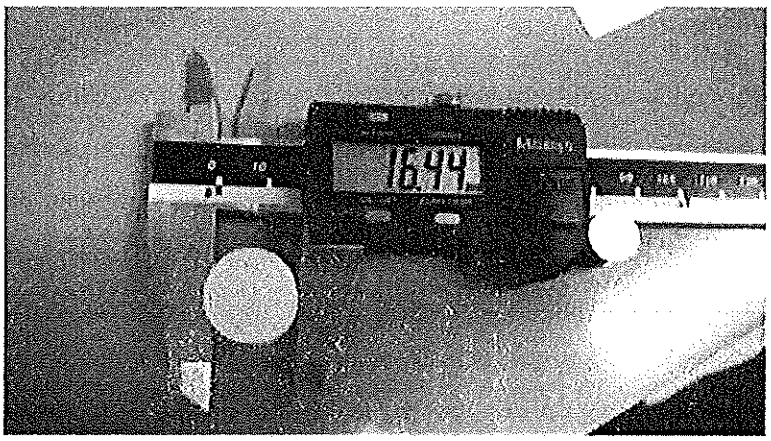


Figura 10 – Amostra de composto Alumina-CAC sob medição por um paquímetro.

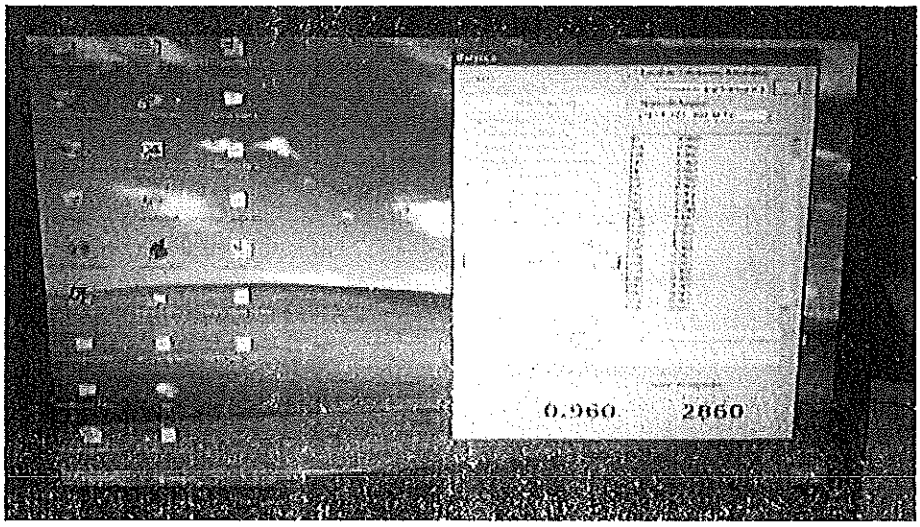


Figura 11 – Interface da última versão do programa de monitoramento em funcionamento.

2.2 MÉTODOS

Após o término do desenvolvimento do programa de medição de massa da balança, um ensaio preliminar foi realizado com o intuito de determinar as temperaturas em que há a perda água do sistema, para que assim o parâmetro de otimização preliminar fosse obtido.

Tal otimização foi realizada na forma de um programa de aquecimento em três etapas. Ao longo de 2 semanas foram realizados sucessivos ensaios com as três amostras. Os aquecimentos das amostras no forno tipo mufla, simultaneamente com a execução programa de coleta de dados (Figuras 8, 9 e 11), foram realizados e foram obtidos os resultados, com o regime de aquecimento adicionado ao resultado final. O aquecimento foi feits com base no seguinte programa:

- Etapa 1: Aquecimento a 2°C/min até 120°C.
- Etapa 2: Patamar em 120°C por 1035 min. (para remoção da água livre)
- Etapa 3: Aquecimento a 2°C/min até 800°C. (para remoção da água combinada)

Ao final do processo, os dados de massa por tempo foram exportados para que gráficos fossem criados no Microsoft Excel, utilizando-se a Equação 2, que calcula a percentagem de massa com o tempo em relação à massa inicial.

$$W(\%) = \frac{M_o - M}{M_o - M_f} \quad \text{Equação 2}$$

Na qual M_o (g) é a massa inicial da amostra, M (g) é a massa instantânea no tempo e M_f (g) é a massa final da amostra. Para calcular a massa de água livre, deve-se encontrar no gráfico de massa por tempo onde termina o primeiro patamar da perda de massa. Anota-se a massa inicial, em seguida a massa no final de todo o processo, e anota-se também a massa da amostra no final desse patamar.

Para calcular a massa de hidratados, repete-se o processo, mas anotando-se a massa onde termina o segundo patamar de perda de massa, e após o término da conta, descontar o que já foi perdido antes.

A velocidade de perda de massa (dW/dt) foi calculada para cada instante (i) de tempo (t) por meio da Equação 3.

$$\left(\frac{dW}{dt}\right)_i (\%) = \frac{W_{i+i0} - W_{i-i0}}{t_{i+i0} - t_{i-i0}} \quad \text{Equação 3}$$

Na qual W_{i+i0} é a massa num tempo determinado, W_{i-i0} é a massa num tempo determinado anterior a ele, t_{i+i0} é o tempo num intervalo determinado e t_{i-i0} é o tempo num intervalo determinado anterior a ele.

3 RESULTADOS E DISCUSSÃO

O programa foi criado utilizando-se a linguagem C#. Isso foi decidido pois ela é relativamente recente (lançada em 2001), e combina parte da facilidade de estruturação dos comandos observada na linguagem BASIC e com a robustez da linguagem C++, que apesar de ser bem mais antiga, ainda hoje é muito utilizada para desenvolvimento de programas e jogos de computador. Além disso, o C# possui a característica única de ser dotada de um parâmetro que é conhecido como programação assíncrona. A programação assíncrona é um tipo de processo que, quando executado, não ocupa toda a capacidade de processamento do sistema quando o código é executado. Isso é extremamente importante, pois quando há a programação no C++ com interface gráfica no modo CLS simples, há um travamento geral do sistema quando o programa é executado, particularmente quando há a execução de recorrências (chamados de

"loop" pelos programadores). Isso só pode ser remediado com a utilização do modo Win32, e criando as chamadas recorrências direcionadas a mensagens, e não a eventos.

Porém, mesmo utilizando-se de programação assíncrona, ela só existe com versões do .NET Framework 4.5 em diante. No caso deste trabalho, o computador em que o programa foi embarcado tolera no máximo até a versão 4.0. Daí foi necessária a busca de um protocolo especial que habilite o funcionamento do modo assíncrono. Ele foi conseguido em um site da internet cujo endereço está nas referências. Além disso, o código-fonte está descrito no Anexo 1.

Os resultados podem ser observados pelos Gráficos 1 e 2. Neles há a massa das amostras coletadas pelo programa, sua variação e o comportamento de secagem das amostras, além da temperatura durante o processo. Nota-se que as três amostras possuem os picos e regiões dos estagios de secagem em períodos de tempo semelhantes. O que os diferencia são apenas as intensidades dos picos e uma diferença razoável na duração do pico no que diz respeito a amostra de com 66% de Al_2O_3 calcinada e 34% de cimento de aluminato de cálcio.

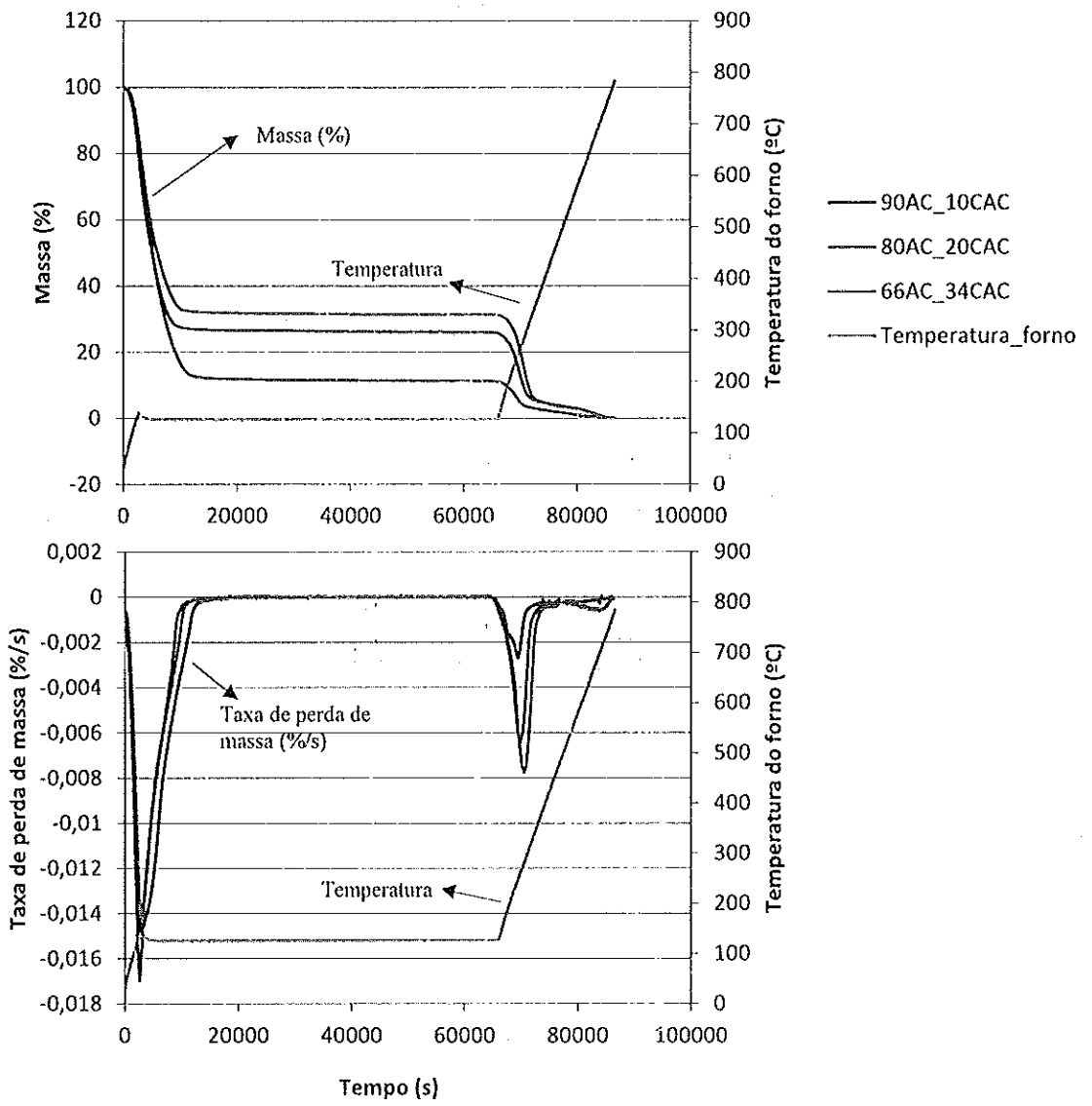


Gráfico 1 – Massa da amostra pelo tempo (superior) e taxa de perda de massa pelo tempo (inferior).

De maneira mais detalhada, pode-se também deduzir que o primeiro pico de todas as amostras denota que está ocorrendo a perda de água livre do sistema. Mesmo considerando as perdas por evaporação mencionadas no primeiro estágio de aquecimento, quando a temperatura alcança os 100°C, nota-se uma perda maior de massa e um acréscimo considerável da taxa de perda de massa com o tempo. Isso justifica a afirmação inicial, porém há um segundo pico. O que denota que provalvemente mais água está sendo removida do sistema. Isso somente é possivel quando há a decomposição de hidratos criados pela presença do ligante. Sabendo-se que as três amostras possuem a mesma quantidade total de água, pode-se determinar quais os percentuais de massa nas três amostras que apresentem essas fases (Gráfico 2):

Caso 1: Amostra com 90% de Al_2O_3 calcinada e 10% de cimento de aluminato de cálcio

$$W(\text{água livre em \%}) = \frac{90,418 - 74,448}{90,418 - 72,121} = 87,2\%$$

$$W(\text{compostos hidratados em \%}) = \frac{90,418 - 73,726}{90,418 - 72,121} - 87,2\% = 4,1\%$$

Caso 2: Amostra com 80% de Al_2O_3 calcinada e 20% de cimento de aluminato de cálcio

$$W(\text{água livre em \%}) = \frac{87,824 - 73,341}{87,824 - 69,517} = 82,7\%$$

$$W(\text{compostos hidratados em \%}) = \frac{87,824 - 70,117}{87,824 - 69,517} - 82,7\% = 14\%$$

Caso 3: Amostra com 66% de Al_2O_3 calcinada e 34% de cimento de aluminato de cálcio (CAC)

$$W(\text{água livre em \%}) = \frac{91,736 - 78,204}{91,736 - 71,819} = 67,7\%$$

$$W(\text{compostos hidratados em \%}) = \frac{91,736 - 72,653}{91,736 - 71,819} - 67,7\% = 28,1\%$$

Dessa forma, comparando-se os três casos, pode-se afirmar que houve uma alteração nos resultados finais dos teores de água livre e de hidratados liberados pelas amostras.

De posse desses dados, pode-se realizar a otimização do processo fazendo com que a etapa entre os picos observada no gráfico 1 seja reduzida o máximo possível, pois gera tempo de equipamento ocioso, tornado-se desnecessária. Com isso, o processo de secagem ocorreria de maneira mais rápida e com menor gasto energético.

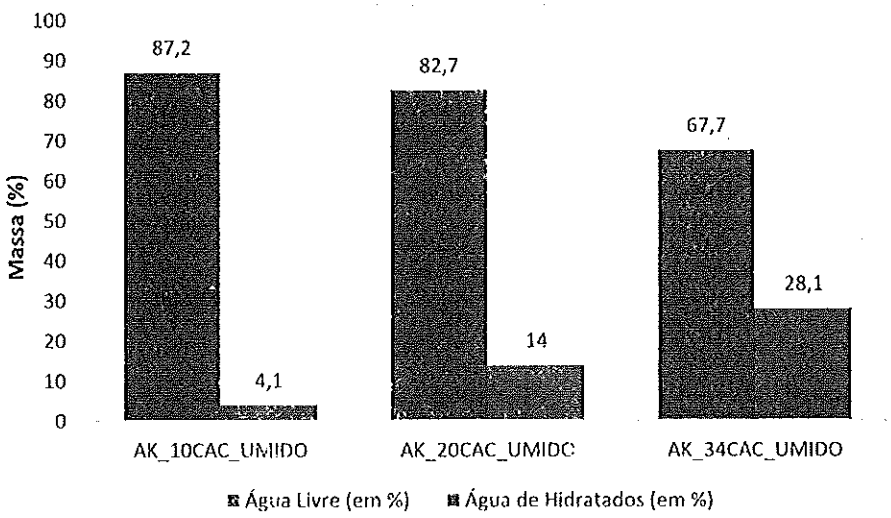


Gráfico 2 – Porcentagens de água presentes nas amostras.

Um fato interessante que pode ser observado é que, quanto maior a concentração de CAC no material refratário, maior a proporção de água liberada por decomposição em relação à liberação de água livre do sistema.

4 CONCLUSÕES

A criação deste equipamento permitiu que as secagens de cerâmicas sejam realizadas de maneira mais eficiente, pois com os monitoramentos tradicionais intermitentes, apenas pode-se saber quando há as secagens, mas não o comportamento delas. Não há como saber se está sendo liberada a água livre ou o dos compostos hidratados como na abordagem apresentada neste trabalho. Ele monitora em tempo real a massa com o tempo através de um ensaio termogravimétrico, gerando dados que, com a ferramenta adequada, fornece a perda de massa e a velocidade de perda de massa de maneira simultânea.

Na montagem do equipamento, cuidados especiais foram tomados para que houvesse o mínio de vibrações e outras interferências externas que poderiam alterar os resultados finais dos ensaios. Escolheu-se uma linguagem de programação que fosse a mais robusta e confiável possível, com bons resultados. Realizaram-se ensaios que comprovaram que, à medida de o teor de cimento de aluminato de cálcio aumentava, maior era a proporção de hidratados que era removida em relação à amostra inicial.

O equipamento desenvolvido abriu a possibilidade de uma secagem mais rápida quando, após o monitoramento inicial do comportamento de secagem, pode-se ajustar as taxas de aquecimento de maneira ótima, para redução de tempo e de custos com energia elétrica. Por exemplo, uma secagem de um tijolo refratário que poderia levar 5 dias poderia ser realizado em 12, 8 horas.

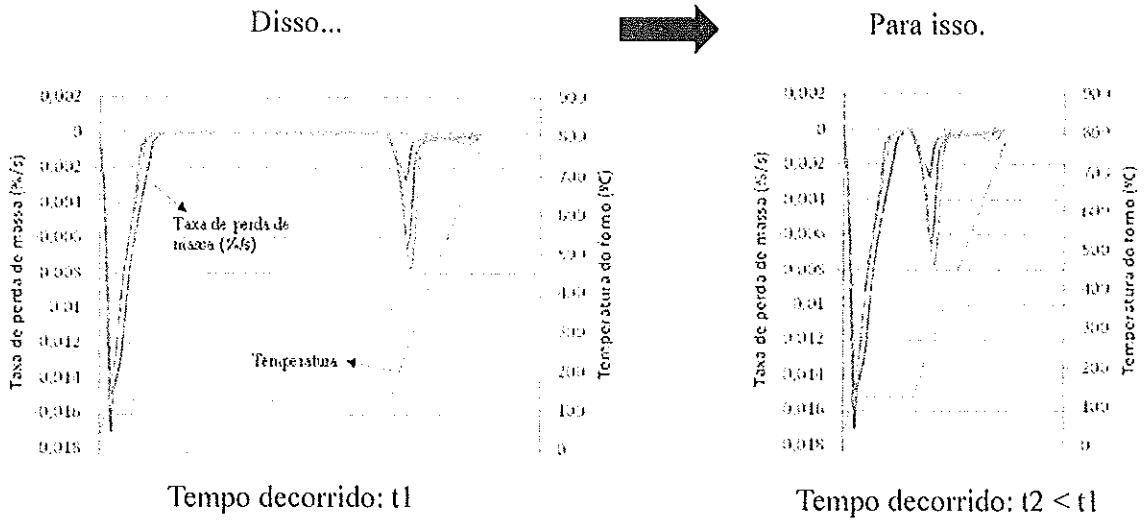


Figura 12 – Demonstração da possibilidade de otimização do processo de secagem com o equipamento e programa embarcado completamente desenvolvidos.

Além disso, pode-se colocar amostras maiores neste equipamento do que normalmente ocorre. Poderia colocar amostras na ordem de alguns centímetros, ao invés de alguns milímetros, o que simularia condições reais de secagem. Com um forno maior, um tijolo inteiro de cerâmica refratária poderia ser colocado e testado.

No caso deste trabalho, as taxas foram realizadas de maneira manual e com base em dados empíricos preliminares. Porém, abre-se a possibilidade de um controle automático de temperatura com a detecção de acordo com as variações de massa em um eventual aperfeiçoamento do programa.

REFERÊNCIAS

SURENDRANATHAN, A.O. **An introduction to ceramics and refractories.** CRC Press, 2015.

BENGISU, M. **Engineering Ceramics.** Springer Ed., p. 461-462, 2001.

LESKO, J. **Industrial design: materials and manufaturing guide.** Second edition, John Wiley & Sons, Inc. , 2008.

WUNDERLICH, B. **Thermal Analysis of Polymeric Materials.** Springer Ed., p. 428-436 ,2005.

<https://gist.github.com/dgrunwald/1961087> Acesso em 01/09/2015.

INNOCENTINI, MURILO D. M.; CARDOSO, FÁBIO A.; AKYIOSHI, MARIO M.; PANDOLFELLI, VICTOR C. **Drying Stages during the Heating of High-Alumina, Ultra-Low-Cement Refractory Castables.** J. Am. Ceram. Soc., 86 [7] 1146–48 (2003).

INNOCENTINI, MURILO D. M.; CARDOSO, FÁBIO A.; AKYIOSHI, MARIO M.; PANDOLFELLI, VICTOR C. **Vaporization Processes and Pressure Buildup during Dewatering of Dense Refractory Castables.** J. Am. Ceram. Soc., 86 [9] 1500–503 (2003).

SALOMÃO, R; PANDOLFELLI, VICTOR C. **Drying Behavior of Polymeric Fiber-Containing Refractory Castables.** Journal ot the Technical Assaciatian ot Refractaries, Japan, 24 [2J 83 - 87(2004).

ANEXO 1 – Código de adaptação de modo assíncrono para o NET.Framework 4.0 do C# versão 12.0

```
// Copyright (c) 2012 Daniel Grunwald
//
// Permission is hereby granted, free of charge, to any person obtaining a copy of
this
// software and associated documentation files (the "Software"), to deal in the
Software
// without restriction, including without limitation the rights to use, copy, modify,
merge,
// publish, distribute, sublicense, and/or sell copies of the Software, and to permit
persons
// to whom the Software is furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be included in all
copies or
// substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
// INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
PARTICULAR
// PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE
// FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT
OR
// OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER
// DEALINGS IN THE SOFTWARE.

using System;
using System.Diagnostics;
using System.Runtime.CompilerServices;
using System.Security;
using System.Threading.Tasks;

// This is a minimal implementation of the stuff needed by the C# 5 Beta
// compiler for 'async'/'await' feature.
// Simply add this file to a .NET 4.0 project, and 'async'/'await' will work
// in your code without requiring your users to install .NET 4.5.
// However, you still need the C# 5 compiler for compiling your code.
// Caveats:
//  - Stack trace is lost when rethrowing exceptions
//  - Unhandled Exceptions in 'async void' methods are re-thrown immediately
//    and not posted to the synchronization context
//  - Flowing ExecutionContext is not supported
//  --> do not use this in security critical code (APTCA)
//  - The code is not optimized. Tasks are always created, repeated boxing may occur.

namespace System.Threading.Tasks
{
    internal static class TaskEx
    {
        public static TaskAwaiter GetAwaiter(this Task task)
        {
            return new TaskAwaiter(task);
        }

        public static TaskAwaiter<T> GetAwaiter<T>(this Task<T> task)
        {
            return new TaskAwaiter<T>(task);
        }
    }
```

```
internal struct TaskAwaiter : INotifyCompletion
{
    readonly Task task;

    internal TaskAwaiter(Task task)
    {
        this.task = task;
    }

    internal static TaskScheduler TaskScheduler
    {
        get
        {
            if (SynchronizationContext.Current == null)
                return TaskScheduler.Default;
            else
                return TaskScheduler.FromCurrentSynchronizationContext();
        }
    }

    public bool IsCompleted
    {
        get { return task.IsCompleted; }
    }

    public void OnCompleted(Action continuation)
    {
        this.task.ContinueWith(
            delegate (Task task) {
                continuation();
            }, TaskAwaiter.TaskScheduler);
    }

    public void GetResult()
    {
        try
        {
            task.Wait();
        }
        catch (AggregateException ex)
        {
            throw ex.InnerExceptions[0];
        }
    }
}

internal struct TaskAwaiter<T> : INotifyCompletion
{
    readonly Task<T> task;

    internal TaskAwaiter(Task<T> task)
    {
        this.task = task;
    }

    public bool IsCompleted
    {
        get { return task.IsCompleted; }
    }

    public void OnCompleted(Action continuation)
```

```
        {
            this.task.ContinueWith(
                delegate (Task<T> task) {
                    continuation();
                }, TaskAwaiter.TaskScheduler);
        }

        public T GetResult()
        {
            try
            {
                return task.Result;
            }
            catch (AggregateException ex)
            {
                throw ex.InnerExceptions[0];
            }
        }
    }
}

namespace System.Runtime.CompilerServices
{
    internal interface INotifyCompletion
    {
        void OnCompleted(Action continuation);
    }

    internal interface ICriticalNotifyCompletion : INotifyCompletion
    {
        [SecurityCritical]
        void UnsafeOnCompleted(Action continuation);
    }

    internal interface IAsyncStateMachine
    {
        void MoveNext();
        void SetStateMachine(IAsyncStateMachine stateMachine);
    }

    internal struct AsyncVoidMethodBuilder
    {
        public static AsyncVoidMethodBuilder Create()
        {
            return new AsyncVoidMethodBuilder();
        }

        public void SetException(Exception exception)
        {
            throw exception;
        }

        public void SetResult()
        {
        }

        public void SetStateMachine(IAsyncStateMachine stateMachine)
        {
            // Should not get called as we don't implement the optimization that this
method is used for.
            throw new NotImplementedException();
        }
```

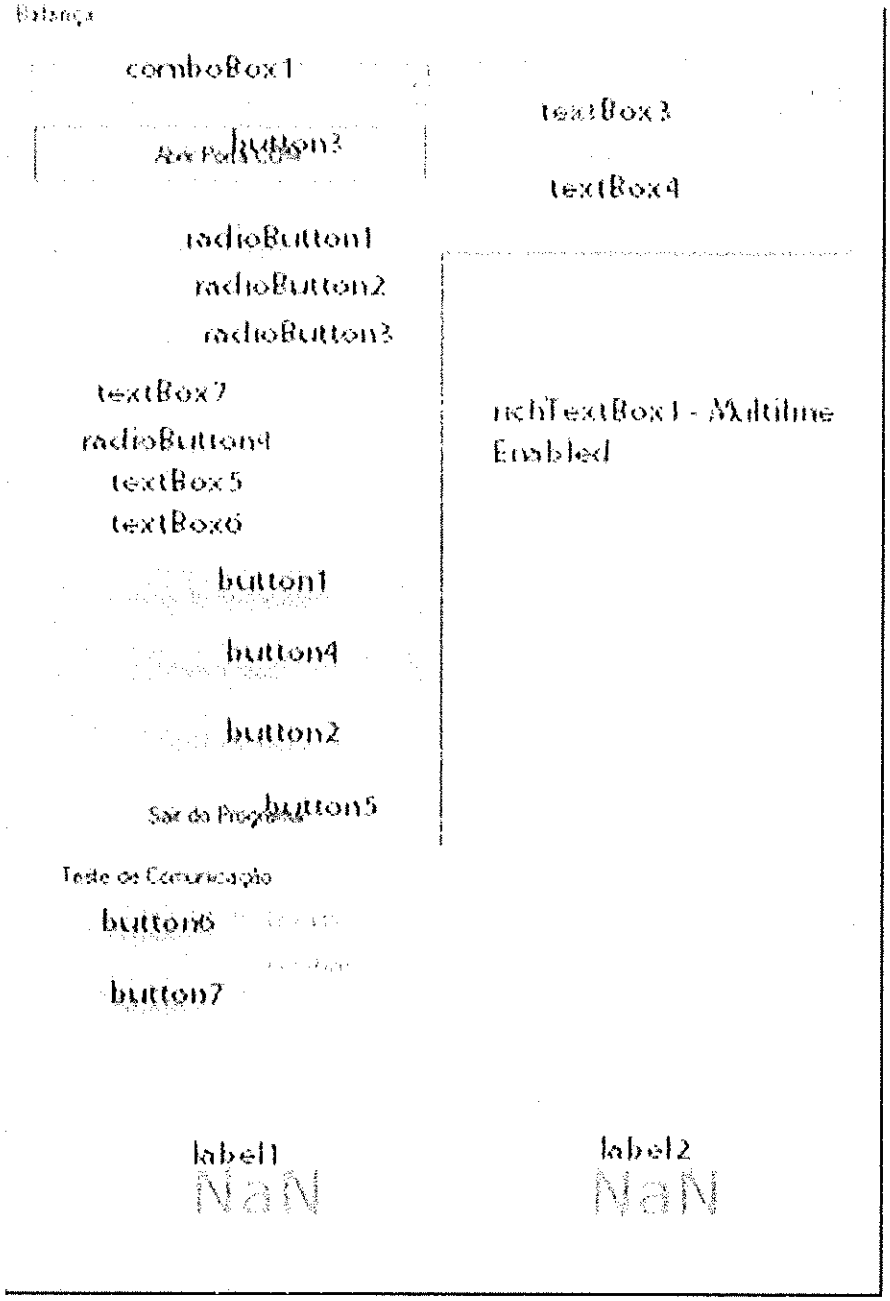
```
        public void Start<TStateMachine>(ref TStateMachine stateMachine) where
TStateMachine : IAsyncStateMachine
        {
            stateMachine.MoveNext();
        }

        public void AwaitOnCompleted<TAwaiter, TStateMachine>(ref TAwaiter awaiter,
ref TStateMachine stateMachine) where TAwaiter : INotifyCompletion where TStateMachine
: IAsyncStateMachine
        {
            awaiter.OnCompleted(stateMachine.MoveNext);
        }

        public void AwaitUnsafeOnCompleted<TAwaiter, TStateMachine>(ref TAwaiter
awaiter, ref TStateMachine stateMachine) where TAwaiter : ICriticalNotifyCompletion
where TStateMachine : IAsyncStateMachine
        {
            awaiter.OnCompleted(stateMachine.MoveNext);
        }
    }

    internal struct AsyncTaskMethodBuilder
    {
        TaskCompletionSource<object> tcs;

        public Task Task { get { return tcs.Task; } }

        public static AsyncTaskMethodBuilder Create()
        {
            AsyncTaskMethodBuilder b;
            b.tcs = new TaskCompletionSource<object>();
            return b;
        }

        public void Start<TStateMachine>(ref TStateMachine stateMachine) where
TStateMachine : IAsyncStateMachine
        {
            stateMachine.MoveNext();
        }

        public void SetStateMachine(IAsyncStateMachine stateMachine)
        {
            // Should not get called as we don't implement the optimization that this
method is used for.
            throw new NotImplementedException();
        }

        public void AwaitOnCompleted<TAwaiter, TStateMachine>(ref TAwaiter awaiter,
ref TStateMachine stateMachine) where TAwaiter : INotifyCompletion where TStateMachine
: IAsyncStateMachine
        {
            awaiter.OnCompleted(stateMachine.MoveNext);
        }

        public void AwaitUnsafeOnCompleted<TAwaiter, TStateMachine>(ref TAwaiter
awaiter, ref TStateMachine stateMachine) where TAwaiter : ICriticalNotifyCompletion
where TStateMachine : IAsyncStateMachine
        {
            awaiter.OnCompleted(stateMachine.MoveNext);
        }
```

```
        public void SetResult()
        {
            tcs.SetResult(null);
        }

        public void SetException(Exception exception)
        {
            tcs.SetException(exception);
        }
    }

    internal struct AsyncTaskMethodBuilder<T>
    {
        TaskCompletionSource<T> tcs;

        public Task<T> Task { get { return tcs.Task; } }

        public static AsyncTaskMethodBuilder<T> Create()
        {
            AsyncTaskMethodBuilder<T> b;
            b.tcs = new TaskCompletionSource<T>();
            return b;
        }

        public void Start<TStateMachine>(ref TStateMachine stateMachine) where
TStateMachine : IAsyncStateMachine
        {
            stateMachine.MoveNext();
        }

        public void SetStateMachine(IAsyncStateMachine stateMachine)
        {
            // Should not get called as we don't implement the optimization that this
method is used for.
            throw new NotImplementedException();
        }

        public void AwaitOnCompleted<TAwaiter, TStateMachine>(ref TAwaiter awaiter,
ref TStateMachine stateMachine) where TAwaiter : INotifyCompletion where TStateMachine
: IAsyncStateMachine
        {
            awaiter.OnCompleted(stateMachine.MoveNext);
        }

        public void AwaitUnsafeOnCompleted<TAwaiter, TStateMachine>(ref TAwaiter
awaiter, ref TStateMachine stateMachine) where TAwaiter : ICriticalNotifyCompletion
where TStateMachine : IAsyncStateMachine
        {
            AwaitOnCompleted(ref awaiter, ref stateMachine);
        }

        public void SetResult(T result)
        {
            tcs.SetResult(result);
        }

        public void SetException(Exception exception)
        {
            tcs.SetException(exception);
        }
    }
}
```

ANEXO 2 – Dados obtidos com o programa da balança em momentos cruciais

AK_10CAC_UMIDO			AK_20CAC_UMIDO			AK_34CAC_UMIDO		
Tempo (s)	Temperatura (°C)	Massa (g)	Tempo (s)	Temperatura (°C)	Massa (g)	Tempo (s)	Temperatura (°C)	Massa (g)
0	28,7	90,418	0	29,4	84,354	0	25,8	91,736
10	29,3	90,412	10	29,3	84,347	10	25,8	91,735
20	30,3	90,41	20	29,3	84,346	20	25,8	91,734
30	31,2	90,409	30	29,4	84,345	30	25,7	91,733
40	32,3	90,409	40	29,4	84,344	40	25,8	91,732
50	33,1	90,406	50	29,7	84,342	50	25,9	91,731
60	34,1	90,406	60	30,1	84,341	60	26,3	91,729
70	34,9	90,406	70	30,8	84,34	70	27	91,728
80	35,4	90,405	80	31,6	84,34	80	27,8	91,728
90	36,2	90,405	90	32,7	84,339	90	28,9	91,726
12230	125,1	74,45	12240	124,3	74,232	12210	124,6	78,242
12240	125,1	74,449	12250	124,3	74,232	12220	124,7	78,241
12250	125	74,448	12260	124,3	74,231	12230	124,9	78,241
12260	125	74,448	12270	124,2	74,231	12240	124,9	78,241
12270	125,1	74,447	12280	124,2	74,231	12250	125	78,241
12280	125,1	74,447	12290	124,2	74,23	12260	125	78,24
12290	125,1	74,446	12300	124,2	74,23	12280	125	78,239
70000	263,9	72,971	69800	262,3	72,472	70000	259,5	75,822
70010	264,4	72,967	69810	262,8	72,464	70010	259,9	75,808
70020	264,9	72,965	69820	262,8	72,455	70020	260,4	75,793
70030	265,3	72,961	69830	263,3	72,445	70030	260,9	75,777
70040	265,8	72,959	69840	263,7	72,436	70040	261,3	75,762
70050	265,3	72,954	69850	263,7	72,428	70050	261,3	75,749
70060	265,8	72,951	69860	264,2	72,419	70060	261,8	75,734
70070	266,7	72,948	69870	264,7	72,41	70070	261,8	75,72
70080	266,7	72,943	69880	264,7	72,401	70080	262,3	75,705
70090	266,7	72,942	69890	265,1	72,392	70090	262,7	75,689

APÊNDICE 1 – Quadro do programa da balança com a relação das variáveis.



APÊNDICE 2 – Linha de código do programa da balança.

```
//----Programa de medição de massa pela balança analítica------
//    Autor: Lincoln Burckhart Santos Silva
//    Criado em Setembro de 2015
//    Linguagem Utilizada: C# versão 12.0
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;
using System.Diagnostics;

namespace Balanca2
{
    public partial class Form1 : Form
    {

        public Form1()
        {
            InitializeComponent();
            VerificarPortas();
        }



        void VerificarPortas()
        {
            String[] ports = SerialPort.GetPortNames();
            comboBox1.Items.AddRange(ports);

        }

        private void button5_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            try
            {
                if (comboBox1.Text == "")
                {
                    MessageBox.Show("Favor Selecionar a Porta COM!!");
                }
                else
                {
                    serialPort1.PortName = comboBox1.Text;
                    serialPort1.BaudRate = Convert.ToInt32(4800);
                    serialPort1.DataBits = Convert.ToInt32(7);
                    serialPort1.Open();
                    comboBox1.Enabled = false;
                    groupBox1.Enabled = true;
                    groupBox2.Enabled = true;
```

```
                groupBox3.Enabled = true;
                groupBox4.Enabled = true;
                textBox1.Enabled = true;
                textBox2.Enabled = true;
                button1.Enabled = true;
                button2.Enabled = true;
                button3.Enabled = false;
                button4.Enabled = false;
                button6.Enabled = true;
                button7.Enabled = true;
                radioButton1.Enabled = true;
                radioButton2.Enabled = true;


            }
        }
        catch(UnauthorizedAccessException)
        {
            MessageBox.Show("Acesso Não Autorizado!!!");
        }
    }

//--------------------------Botão ENVIAR-----------------------------------
    private void button6_Click(object sender, EventArgs e)
    {
        try
        {
           if (comboBox1.Text == "")
          {
                MessageBox.Show("Favor Selecionar a Porta COM!!");
            }
            else
            {
                    char[] _bytes = { '\x0050', '\x000D', '\x000A' };
                byte[] _enviarbytes = {    0x50 ,    0x0D ,    0x0A };
                string mensagem = "";
                    for (int i = 0; i<3; i++)
                    {
                      mensagem = mensagem + _bytes[i];
                    }
                textBox1.Text = mensagem;
                serialPort1.Write(_enviarbytes, 0, 3);
            }
        }
        catch (UnauthorizedAccessException)
        {
            MessageBox.Show("Acesso Não Autorizado!!!");
        }
    }


//--------------------------Botão LER-----------------------------------
    private void button7_Click(object sender, EventArgs e)
    {
        try
        {
            if (comboBox1.Text == "")
            {
                MessageBox.Show("Favor Selecionar a Porta COM!!");
            }
           else
           {
```

```
                    textBox2.Text = serialPort1.ReadExisting();
                    label1.Text = (textBox2.Text).Substring(5,9);
                }
            }
            catch (UnauthorizedAccessException)
            {
                MessageBox.Show("Acesso Não Autorizado!!!");
            }
        }


        //---------------Parada-----------------------------------
        public static Task Delay(double milliseconds)
        {
            var tcs = new TaskCompletionSource<bool>();
            System.Timers.Timer timer = new System.Timers.Timer();
            timer.Elapsed += (obj, args) =>
            {
                tcs.TrySetResult(true);
            };
            timer.Interval = milliseconds;
            timer.AutoReset = false;
            timer.Start();
            return tcs.Task;
        }

        //---------------------Botão Iniciar Monitoramento-------------------------
        int parada = 0;

        private async void button1_Click(object sender, EventArgs e)
        {
            if (textBox3.Text == "" ||  textBox4.Text =="")
            {
                MessageBox.Show("Favor informar o Local de Salvamento e o Nome de
Arquivo!!!");
            }
            else
            {
                var stopwatch = new Stopwatch();

                groupBox3.Enabled = false;
                button4.Enabled = true;
                button1.Enabled = false;
                button6.Enabled = false;
                button7.Enabled = false;
                button2.Enabled = false;
                button5.Enabled = false;

                parada = 0;

                try
                {
                    if (comboBox1.Text == "")
                    {
                        MessageBox.Show("Favor Selecionar a Porta COM!!");
                    }

                    else

                    {
                        int t = 0;
                        int l = 0;
```

```
                        List<String> dados = new List<String>();

                        if (radioButton1.Checked == true && radioButton2.Checked ==
false && radioButton3.Checked == false && radioButton4.Checked == false)
                        {
                            t = 362; //3 horas, 10800 segundos, mas são 30 segundos de
atraso, então t = 360
                        }
                        else if (radioButton1.Checked == false && radioButton2.Checked
== true && radioButton3.Checked == false && radioButton4.Checked == false)
                        {
                            t = 1442; //24 horas, 86400 segundos, mas são 60 segundos
de atraso, então t = 1440
                        }
                        else if (radioButton1.Checked == false && radioButton2.Checked
== false && radioButton3.Checked == true && radioButton4.Checked == false)
                        {
                            t = 2; //loop infinito...
                        }
                        else if (radioButton1.Checked == false && radioButton2.Checked
== false && radioButton3.Checked == false && radioButton4.Checked == true)
                        {
                            t = (((Convert.ToInt32(textBox5.Text))*3600) / (60 /
(Convert.ToInt32(textBox6.Text))) + 2); //Converte o que estiver na caixa de texto.
                        }
                        for (int k = 0; k < t; l++, k++)
                        {
                            textBox2.Text = (serialPort1.ReadExisting()).Substring(0,
14);

                            if (parada == 1)
                            {
                                break;
                            }
                            else
                            {
                                richTextBox1.Text = String.Join(Environment.NewLine,
dados);
                                serialPort1.DiscardInBuffer();

                                //Ensaio Curto
                                if (radioButton1.Checked == true &&
radioButton2.Checked == false && radioButton3.Checked == false && radioButton4.Checked
== false) //atraso pra ensaio curto
                                {
                                    //120p/h , 2p/min, 1p/30s   //30000
                                    //para o relogio
                                    label1.Text = (textBox2.Text).Substring(5, 7);
                                    label2.Text = (30 * k).ToString();
                                    if ((textBox2.Text).Substring(0, 1) != "S")
                                    {
                                        await Delay(30000);

                                    }
                                    else
                                    {
                                        dados.Add(label2.Text + "\t" + label1.Text);
                                        await Delay(30000);
                                    }
                                }
```

```
                                //Ensaio Longo
                                else if (radioButton1.Checked == false &&
radioButton2.Checked == true && radioButton3.Checked == false && radioButton4.Checked
== false) //atraso pra ensaio longo
                                {
                                    label1.Text = (textBox2.Text).Substring(5, 7);
                                    label2.Text = (60 * k).ToString();
                                    if ((textBox2.Text).Substring(0, 1) != "S")
                                    {
                                        await Delay(60000);
                                    }
                                    else
                                    {
                                        dados.Add(label2.Text + "\t" + label1.Text);
                                        await Delay(60000);
                                    }
                                }

                                //Loop infinito
                                else if (radioButton1.Checked == false &&
radioButton2.Checked == false && radioButton3.Checked == true && radioButton4.Checked
== false)
                                {
                                    label1.Text = (textBox2.Text).Substring(5, 7);
                                    label2.Text = ((60 /
(Convert.ToInt32(textBox7.Text))) * 1).ToString();
                                    if ((textBox2.Text).Substring(0, 1) != "S")
                                    {
                                        await Delay((60 /
(Convert.ToInt32(textBox7.Text))) * 1000);
                                    }
                                    else
                                    {
                                        dados.Add(label2.Text + "\t" + label1.Text);
                                        await Delay((60 /
(Convert.ToInt32(textBox7.Text))) * 1000);
                                    }
                                    k--;
                                }

                                //Tempo e pontos/min determinados.
                                else if (radioButton1.Checked == false &&
radioButton2.Checked == false && radioButton3.Checked == false && radioButton4.Checked
== true)
                                {
                                    label1.Text = (textBox2.Text).Substring(5, 7);
                                    label2.Text = ((60 /
(Convert.ToInt32(textBox6.Text))) * k).ToString();
                                    if ((textBox2.Text).Substring(0, 1) != "S")
                                    {
                                        await Delay((60 /
(Convert.ToInt32(textBox6.Text))) * 1000);
                                    }
                                    else
                                    {
                                        dados.Add(label2.Text + "\t" + label1.Text);
                                        await
Delay((60/(Convert.ToInt32(textBox6.Text)))*1000);
                                    }
                                }
                            }
                        }
```

```
                }
            }
            catch (UnauthorizedAccessException)
            {
                MessageBox.Show("Acesso Não Autorizado!!!");
            }
            richTextBox1.SaveFile(textBox3.Text + "\\" + textBox4.Text + ".txt",
RichTextBoxStreamType.PlainText);
        }
    }
    //--------------------Botão Parar---------------------------
    private void button4_Click(object sender, EventArgs e)
    {
        button1.Enabled = true;
        button4.Enabled = false;
        button6.Enabled = true;
        button7.Enabled = true;
        button2.Enabled = true;
        button5.Enabled = true;
        groupBox3.Enabled = true;
        parada = 1;
    }

    //--------------------Botão Exportar Dados----------------------------------
---------
    private void button2_Click(object sender, EventArgs e)
    {
        // Create a SaveFileDialog to request a path and file name to save to.
        SaveFileDialog saveFile1 = new SaveFileDialog();

        // Initialize the SaveFileDialog to specify the RTF extension for the
file.
        saveFile1.DefaultExt = "*.txt";
        saveFile1.Filter = "Arquivo de Texto|*.txt";

        // Determine if the user selected a file name from the saveFileDialog.
        if (saveFile1.ShowDialog() == System.Windows.Forms.DialogResult.OK &&
           saveFile1.FileName.Length > 0)
        {
            // Save the contents of the RichTextBox into the file.
            richTextBox1.SaveFile(saveFile1.FileName,
RichTextBoxStreamType.PlainText);
        }
    }

    //-----Variável global para selecionar o local de salvamento.
    FolderBrowserDialog folderBrowserDialog1 = new FolderBrowserDialog();
    //------

    private void button8_Click(object sender, EventArgs e)
    {
     string folderPath = "";
        if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)
        {
            folderPath = folderBrowserDialog1.SelectedPath;
        }
        textBox3.Text = folderPath;
    }
}
}
```