UNIVERSIDADE LUTERANA DO BRASIL

PRÓ-REITORIA DE GRADUAÇÃO

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

DESENVOLVIMENTO DE UM ANALISADOR DE CO_2 PARA USO EM BIOFILTROS

ANDRÉ LUIS CASARA

Canoas, Julho de 2012

ANDRÉ LUIS CASARA

DESENVOLVIMENTO DE UM ANALISADOR DE CO_2 PARA USO EM BIOFILTROS

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia Elétrica da ULBRA como um dos requisitos obrigatórios para a obtenção do grau de Engenheiro Eletricista

Departamento:

Engenharia Elétrica

Área de Concentração

Instrumentação Eletroeletrônica

Professor Orientador:

MSc. Eng. Eletr. Luis Fernando Espinosa Cocian – CREA-RS: 88.866-D

Canoas Rio Grande do Sul

2012

FOLHA DE APROVAÇÃO

Nome do Autor: André Luis Casara

Matrícula: 002101409-4

Título: Desenvolvimento de um Analisador de CO_2 para uso em Biofitros.

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia Elétrica da ULBRA como um dos requisitos obrigatórios para a obtenção do grau de Engenheiro Eletricista

Professor Orientador:

MSc. Eng. Eletr. Luis Fernando Espinosa Cocian

CREA-RS: 88.866-D

Banca Avaliadora:

MSc. Eng. Eletr. Nolvi Franscisco Baggio Filho

CREA-RS: 139435

Conceito Atribuído (A-B-C-D):

MSc. Eng. Eletr. Paulo César Cardoso Godoy

CREA-RS: 011.6822-D

Conceito Atribuído (A-B-C-D):

Assinaturas:

Autor André Luis Casara	Orientador Luis Fernando Espinosa Cocian
Avaliador Paulo César Cardoso Godoy	Avaliador Nolvi Francisco Baggio Filho

Relatório Aprovado em:

DEDICATÓRIA

Dedico este trabalho à minha mãe Berenice Maria Casara, ao meu pai Sérgio José Casara, a minha esposa Luciana Regina Popdgaiski, meu vô Nestor Casara, minha vó Odete Colombo e a todos os meus amigos e profissionais que fortaleceram meu conhecimento e meu sucesso.

AGRADECIMENTOS

Todos os momentos foram compartilhados com várias pessoas e diversos profissionais no meu círculo de amigos, expresso meus sinceros e cordiais agradecimentos pela força e pelo sucesso.

Ao Professor Cocian e Miriam, pelas observações, dicas e apoio no desenvolvimento de todo o trabalho desenvolvido, com valiosas correções.

Aos outros professores Dalton e Godoy pelo apoio e troca de experiências com preciosas sujestões.

Ao Guilherme Eckert e Leonardo Casanova pelas sugestões e apoio.

Ao apoio e colaboração da Ivene do laboratório de microbiologia da Unisinos.

A minha esposa Luciana Regina Podgaiski pela paciência e colaboração valiosa.

RESUMO

CASARA, André Luis. Desenvolvimento de um Analisador de CO_2 para uso em Biofiltros. Trabalho de Conclusão de Curso em Engenharia Elétrica - Departamento de Engenharia Elétrica. Universidade Luterana do Brasil. Canoas, RS. 2012. Este trabalho mostra o desenvolvimento de um dispositivo que tem o objetivo principal adquirir dados da respiração microbiana de um biofiltro a partir da leitura indireta de CO_2. O sistema é composto de um *hardware* mecânico, placa eletrônica, amplificação do sensor, software na linguagem "C" para microcontrolador, na linguagem C# para interface com o computador e na linguagem SQL para salvar dados em um banco de dados. Estes dados são analisados através de uma interface gráfica e são filtrados entre intervalos de tempo escolhidos. O *software* possui telas onde fora possíveis realizar a calibração do sensor de CO_2 e salvar estes dados de calibração em tabelas no banco de dados. Foi realizada também a leitura, a calibração e aquisição dos dados de temperatura de um sensor para futuras análises. Todas as etapas foram concluídas e foi ainda possível testar na prática o crescimento de uma comunidade de microrganismos, comprovando as fases de crescimento microbiano com a literatura.

Palavras chave: Sensor CO_2. Biofiltro. Aquisição. Microrganismos Aeróbios.

ABSTRACT

CASARA, André Luis. Desenvolvimento de um Analisador de CO_2 para uso em Biofiltros (Development of CO_2 Analizer for use in Biofilters). Work of Conclusion of Course in Eletrical Engineering – ULBRA Eletrical Engineering Department. Lutheran University of Brazil. Canoas, RS. 2012. This work show the development of a device for aquire data of breathing micro-organisms of a biofilter through of indirect reading of CO_2. The system is one mechanical hardware, electronic board, amplifier sensor board, software in "C" language for microcontroller, software in "C#" language for interface with computer and software in "SQL" for save the data on database. Those data can be analized for a graphical interface and filtered for the one range of time. The software has windows where was possible the calibration of CO_2 sensor and save that data in a table of a database tables. It was made also the reading, calibration and acquisition of temperature data of a sensor for future analysis. All steps were finished, and it had the possibility, for practice testing of community micro-organisms, proving the phases of micro-organisms growth with literature.

Keywords: CO_2 sensor. Biofilter.Aquisition. Aerobic Microorganism

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT: Associação Brasileira de Normas Técnicas.

ADRESL:Registrador que contém a parte menos significativa do conversor analógico para digital.

ADRESH:Registrador que contém a parte mais significativa do conversor analógico para digital.

ADCON1: Registrador de controle do conversor analógico para digital

BRGEN: Registrador responsável pelo controle da taxa de transmissão.

LISTA DE SÍMBOLOS

GC- Cromatografia Gasosa

GC-MS- Cromatografia gasosa acoplada à massas

UV-VIS- Espectrometria do ultravioleta e do Visível

NIR- Espectrometria do infravermelho próximo

SUMÁRIO

1. INTRODUÇÃO

Com o crescimento populacional exponencial e o aumento da demanda de empregos, as indústrias em geral, hospitais estão produzindo cada vez mais resíduos perigosos que devem ser devidamente tratados e dispostos no meio ambiente. Assim, para o descarte destes resíduos são utilizados os aterros sanitários que consistem em valas cobertas que, quando cheias de resíduos são envelopadas com a colocação de respiradouros (chaminé que vem do centro da vala até a periferia) para retirada dos gases produzidos internamente. Segundo NBR 10004: 2004, estes aterros sanitários são considerados de classe 1 onde se tem a mais diversa composição de resíduos perigosos industriais. Em grande parte destes aterros não há fiscalização da poluição atmosférica emitida por estes respiradouros, os quais são nocivos aos ecossistemas e possíveis residências nas proximidades.

Existem muitos métodos de monitoramento da poluição atmosférica, dependendo do tipo de poluente, por exemplo, a cromatografia gasosa (GC), a espectrometria de massas (GC-MS), a espectrometria de espectro ultravioleta e visível (UV-VIS), espectro infravermelho (NIR), a eletroquímica, quimiluminescência e outras técnicas analíticas (Ohlweiler, 1984). Estas técnicas são consideradas de alto custo para realizar o monitoramento do poluente, possuem problemas metodológicos para extração e tratamento das amostras, problemas com descarte e limpeza de vidrarias pós-análises, interpretação de dados, calibrações regulares com padrões certificados, numerosas e extensivas amostragens no local e transporte adequado da amostra para o laboratório (Conti, 2008).

Os processos de remediação da poluição atmosférica são os mais variados, entre eles os processos de filtragens físicas e químicas, combustão (incineração térmica ou catalítica), adsorção, absorção, ozonização,

mascaramento e ou diluição dos gases poluidores (Macintyre, 2008). Através de processos caros, tal como a utilização do Biogás (Metano), é possível a recuperação energética, sendo muitas vezes inviável num aterro devido à composição dos gases serem diversificadas, necessidade de filtros químicos caros e o fluxo não permanecer constante ao longo do período de degradação das substâncias do aterro. A produção de biogás é viável quando temos uma grande quantidade de uma substância única, tal como excrementos do gado, porco, etc. (Schulz & Eder, 2001).

A biofiltragem vem ganhando espaço na remedição de gases poluentes, sendo um processo considerado de baixo investimento comparado com os métodos citados anteriormente (Jantschak et. al., 2004). O biofiltro consiste num reator no qual são criados microrganismos de diferentes espécies, que tem a função de degradar às substâncias poluidoras através de suas enzimas intra e extracelulares, catalisando assim as reações químicas de específicos compostos orgânicos nocivos a saúde e ao meio ambiente. Estas reações na presença de oxigênio realizam a oxi-redução das moléculas orgânicas formando moléculas mais simples e menos nocivas. A partir da degradação os microrganismos adquirem fonte de energia e matéria prima necessária para seu crescimento celular através do carbono disponibilizado pelas moléculas orgânicas. Como subproduto da respiração microbiana e quebra de ligações químicas, os microrganismos liberam dióxido de carbono (CO_2) e água. Os processos biológicos, segundo Cookson (1995), são confiáveis quando o processo de crescimento celular é controlado, e a bioquímica dos microrganismos e suas condições ambientais são bem entendidas.

Este projeto consiste no desenvolvimento de um analisador de CO_2 como ferramenta de auxílio na monitoração do crescimento microbiano em biofiltro aplicado a remoções de gases poluentes de aterros sanitários. O equipamento possibilita a medição dos níveis de CO_2 para verificar a etapa de crescimento microbiano no biofiltro.

Com as informações fornecidas pelo equipamento analisador de CO_2 é possível verificar a fase de crescimento microbiano e atuar nas variáveis de

controle ambientais dos microrganismos. Por exemplo, no controle do biofiltro em fluxo contínuo poderia-se saber a hora exata de retirada da biomassa microbiana e introdução de caldo nutritivo pela leitura do CO_2 oriundo da respiração celular de microrganismos aeróbios. Cabe salientar que normalmente para se avaliar o crescimento microbiano no biofiltro utilizam-se amostragens, as quais são levadas para laboratório onde é contado o número de colônias após três dias de incubação. Como conseqüência destas amostragens demoradas tem-se um atraso no tempo de resposta para atuar no sistema e melhorar sua eficiência.

Os dados de CO_2 obtidos pelo sistema desenvolvido são enviados ao computador, onde são armazenados e visualizados através de uma interface gráfica. Com base nestes dados podem-se tomar decisões para agir sobre o sistema na remoção de gases tóxicos ao meio ambiente. Com a interface gráfica pode-se avaliar tendências, ponto de operação, histórico dos experimentos que auxiliarão na otimização do desempenho do biofiltro. A interface com o usuário oferece a possibilidade de ajustes e calibrações do sensor de CO_2, sendo armazenados estes dados para aferir a leitura correta do analisador.

Como complemento para testes e calibrações, foi desenvolvido um sistema mecânico para simular o biofiltro em condições reais onde estão os microrganismos, válvulas para mistura dos gases, amostragem do gás de amostra e do gás de referência que foi usado na calibração do analisador.

1.1. Visão Geral do Problema

Promover uma alternativa econômica para tratamento de gases de aterros sanitários utilizando ferramentas microbiológicas através do uso de biofiltro. A leitura da concentração de dióxido de carbono (CO_2) permite controlar o crescimento de microrganismos dentro do biofiltro a fim de melhorar a eficiência na degradação de poluentes atmosféricos. É importante salientar que os microrganismos degradam os gases tóxicos convertendo em biomassa (aumento do número de células), CO_2 (respiração celular) e água.

1.2. Formulação do Problema de Engenharia

Remoção de poluentes gasosos tóxicos de aterros sanitários através do controle de crescimento de microrganismos.

1.3. Formulação do Problema Comercial

O tratamento dos gases de aterros sanitários através de ferramentas microbiológicas é viável economicamente (SABO 2003), pois necessita de baixo investimento inicial, requer basicamente controle de temperatura em temperaturas baixas (35 a 55 graus Celsius), controle de alimento para os microrganismos, controle do excesso de biomassa, e amostragens rotineiras para verificação de crescimento microbiano. Os materias empregados não precisam ser de custo elevado, podem-se usar plásticos, pois a temperatura é baixa e o meio microbiano é bastante ácido.

Cabe salientar que processos de utilização de energéticos, como o uso do biogás (metano), apresentam altos investimentos, pois requerem tubulações de aço inox para evitar corrosão, filtragem fisico-química com filtros que devem ser substituídos ou regenerados regularmente, vazão de gases adequada para a utilização energética, controle das emissões (CO, CO_2, H_2S, NO_x) provenientes da combustão na geração de energia e queima dos gases não utilizados, equipamentos a prova de explosão, controles automatizados de sincronismo e medições com o sistema energético.

Os ganhos sociais seriam os de evitar o aumento de gases de efeito estufa, tal como o metano (CH_4), onde o mesmo tem o potencial de aquecimento global vinte uma vezes maior que o CO2, reduzir os odores nas redondezas, podendo causar problemas pulmonares na população vizinha, evitar também o efeito da chuva ácida pela remoção do gás sulfídrico (H_2S).

Haveria a possibilidade de captação de capital estrangeiro com a venda de créditos de carbono para contribuir com os gastos operacionais e de manutenção do sistema, assim como a obtenção de recursos para melhorar o desenvolvimento do projeto.

1.4. Estudos de mercado

1.4.1 Identificação dos Interessados

No Rio Grande do Sul existem cerca de cinqüenta e cinco aterros sanitários, sessenta e sete aterros controlados e vinte e um lixões (Gomes, 2010), sendo que a grande maioria não tem tratamento dos gases tóxicos oriundos das valas sanitárias. Assim, há potencial de implementação do sistema nos diversos aterros sanitários do estado e possivelmente em outros estados da federação.

1.4.2 Oportunidade de Negócios

Há oportunidade de implementação na empresa Pró-ambiente na cidade de Gravataí, RS e na empresa Ultresa na cidade de Estância Velha, RS.

1.5. Definição do Escopo do Projeto

O sistema foi desenvolvido em etapas, onde realizou-se o desenvolvimento do sistema mecânico, eletrônico e software no computador.

1.5.1 Objetivos Gerais

Desenvolver um analisador de CO_2 como ferramenta de montoração e otimização de biofiltros aplicados à remoção de poluentes gasosos de aterros sanitários, e outras aplicações.

1.5.2 Objetivos Específicos

-Desenvolver um equipamento para medição de CO_2 na faixa de 350 à 10000 ppm.

-Desenvolver sistema mecânico para crescimento de microrganismos.

-Desenvolver software para aquisição e análise dos dados.

-Realizar a calibração dos sensores.

-Realizar a comunicação entre a placa eletrônica e o computador.

-Realizar o condicionamento de sinal adequado do sensor de CO_2 e temperatura.

-Armazenar todos os dados num banco de dados.

1.5.3 Metas

Foram alcançados 100% das metas propostas, desenvolvido o equipamento de medição de CO2, sistema mecânico, amostragem correta dos gases no biofiltro e a correção dos valores obtidos por meio de curva de calibração do analisador.

2. REFERENCIAL TEÓRICO

2.1. Conceitos Básicos e Estudos Preliminares

O projeto visa o desenvolvimento de um analisador de CO_2 que foi colocado na saída de um filtro biológico, como ferramenta para controlar o processo de filtragem biológica, e na remoção de gases tóxicos provenientes de aterros sanitários.

A seguir serão descritos alguns tópicos importantes para a compreensão do projeto.

2.1.1 Aterros Sanitários

Os aterros sanitários são locais onde são dispostos resíduos industriais perigosos que tem a função de minimizar os efeitos nocivos ao meio ambiente, com a impermeabilização do solo, tratamento e drenagem de efluentes líquidos e a cobertura dos resíduos. É importante lembrar que os aterros se diferem dos lixões, pois nestes o lixo é despejado no solo a céu aberto sem nenhum tratamento dos efluentes líquidos.

Segundo a norma NBR 10004:2004, os resíduos são classificados em:

a) resíduos classe I - Perigosos;

b) resíduos classe II – Não perigosos;

- resíduos classe II A – Não inertes.

- resíduos classe II B – Inertes.

Depois de dispostos estes resíduos no aterro, respeitando-se a classificação, os mesmos são coberto por uma manta impermeabilizante, e são instalados dutos de plástico ou concreto em toda a sua extensão para a eliminação dos gases tóxicos formados no interior da vala como subproduto da degradação.

2.1.2 Processos de Tratamento de Efluentes Gasosos

Existem vários tipos de processos para tratamento ou redução da concentração de gases tóxicos nocivos ao meio ambiente e a saúde das

pessoas. Segundo Janke (2002), pode ser classificado em em quatro processos diferentes:

a) Processos Térmicos: são aqueles que consistem na queima dos gases tóxicos. Existem métodos que ocorrem em altas temperaturas (cerca de 1500 ˚C), outros métodos misturam os gases com outros combustíveis fósseis tais como gás natural e óleo (750 a 900 ˚C) com aproveitamento energético, e também métodos com queima catalítica onde ocorre a oxidação dos gases por meio de um catalisador (300 a 500 ˚C) que acelera as reações químicas. Estes métodos tem desvantagens, muitas vezes não tem aproveitamento energético (calor), necessitam de combustíveis fósseis, possíveis emissões de monóxido de carbono (CO) e óxidos nitrosos (NO_x), gases contendo materiais tóxicos dos catalisadores (Si,P,As,S).

b) Processos Físicos: são os que utilizam processos físicos para remoção dos poluentes gasosos. Existem métodos que utilizam a absorção dos compostos gasosos por meio de um material absorvente, tal como o carvão ativado. Outros métodos utilizam a condensação onde o gás é resfriado até passar da fase gasosa para fase líquida. Estes métodos possuem diferentes capacidades de absorção para diferentes compostos, saturam sendo necessário a troca do elemento filtrante, problemas com descarte de material filtrante, a poeira é um problema nestes sistemas podendo ocorrer obstruções além de nem todos os poluentes serem condensáveis.

c) Processos Químicos: são os que utilizam reagentes para desencadear transformações químicas nos compostos de interesse. Há alguns processos onde a ozonização dos gases onde os mesmos passam por eletrodos com uma elevada tensão elétrica (6.000 a 24.000 V) formando ozônio (O_3) que é altamente reativo, realizando a degradação dos gases. Outros processos tais como os lavadores de gases, onde o gás é borbulhado em soluções altamente oxidativas e os compostos são adsorvidos no meio aquoso. Estes processos necessitam de grandes quantidades de reagentes, geração de muitos efluentes líquidos e necessitar de estação de tratamento dos dejetos.

d) Processos Biológicos: são os que utilizam os microrganismos para degradar as substâncias gasosas. Existem processos chamados de biolavadores onde os gases são inicialmente borbulhados no meio líquido e, após a troca gasosa com o meio, vão para outra câmara onde existe um material de suporte coberto por película biológica responsável pela degradação. Outro processo é o biofiltro de leito fixo, que consiste em uma torre onde o material filtrante pode ser sintético ou orgânico, que serve de suporte para os microrganismos e no caso do orgânico como fonte de nutrientes para o crescimento microbiano. O gás percorre esta torre em contra fluxo, sendo degradado pelo biofilme. Outro processo seria o biofiltro de leito escorrido, que é bem similar ao de leito fixo, sendo que a principal diferença é que há uma circulação de meio nutritivo líquido ao longo da torre. Esses processos anteriormente citados não podem ser utilizados em casos de gases com alta concentração de ácidos ou vazões muito elevadas, pois há necessidade de um tempo de permanência no sistema para realizar a degradação, além de administrar nutrientes para crescimento microbiano.

2.1.3 Parâmetros de Desenho de um Sistema de Biofiltração

Existem características que devem ser observadas para o desenvolvimento de um biofiltro, pois existe uma variedade muito grande dos contaminantes e de microorganismos (FRANKE, 2011). Assim é importante observar as seguintes características:

a) Características do gás contaminante: concentração dos gases, fluxos, presença de partículas e temperatura do gás para entrar no biofiltro.

b) Seleção do material filtrante: serve como suporte para o crescimento dos microrganismos, podendo ser utilizado serragem, plástico, rochas porosas, etc.

c) Nível de umidade dentro do biofiltro: mantém as condições ideais para o crescimento microbiano.

d) Microrganismos: podem-se empregar monoculturas de bactérias, fungos e mistura de espécies.

2.1.4 Técnicas para Determinação do CO_2

Existem várias técnicas para determinação dos níveis de dióxido de carbono, a seguir serão tratados os diferentes princípios utilizados.

a) Métodos potenciométricos: são células galvânicas onde são medidas as forças eletromotrizes pela diferença dos potênciais dos componentes de um par eletrólito como resposta previsível as atividades (ou concentrações) das espécies iônicas eletroativas presentes nas soluções em estudo (OHLWEILER, 1981). Muitas vezes é medido o ponto de equilíbrio entre uma solução referência com a solução de amostra, usando soluções colorimétricas indicadoras de potencial de hidrogênio (pH).

b) Métodos por radiação infravermelha: são detectores que utilizam a medição das radiações das ondas eletromagnéticas cujos comprimentos de onda variam de 0,7 a 1,0 μm. Estas radiações são divididas em três faixas espectrais: IV próximo (0,7 a 1,1 μm), IV médio (1,1 a 3,0 μm) e IV distante (3 a 1000 μm). Através de um filtro na faixa de absorção da molécula de dióxido de carbono é possível termos um sinal de saída proporcional a concentração (MOREIRA, 2011). Cabe salientar que a absorção da radiação infravermelha envolve transições vibracionais e rotacionais na ligação entre as moléculas, pois a radiação incidente excita as moléculas que respondem a diferentes comprimentos de onda (OHLWEILER, 1981).

c) Detectores por ionização de chama (FID): são detectores que se baseiam no princípio que a condutividade elétrica do gás é diretamente proporcional a concentração das partículas carregadas dentro do gás. Através de uma chama oxidante usando hidrogênio (H_2) e Ar sintético (79% N_2 + 21% O_2), o gás é ionizado por uma diferença de potencial de -170 V que proporciona uma resposta em milivolts proporcional a concentração quando diferentes gases passam pelo detector. O dióxido de carbono tem baixa resposta neste detector, sendo necessário convertê-lo através de reação catalítica com um material catalizador e hidrogênio a uma temperatura de 400 °C, para transformá-lo em metano (CH_4), que tem ótima resposta neste detector (MCNAIR, 1968).

d) Detectores de condutividade térmica (TCD): são detectores que se baseiam no princípio que corpos quentes perdem calor a uma taxa que depende da composição do gás ao redor de filamentos aquecidos. Existe um canal para o gás de arraste (He, N_2) que é chamado de referência e outro canal que contém o gás de arraste e a amostra, assim pelo desbalanço da ponte de Wheatstone obtem-se um sinal que é proporcional à concentração (MCNAIR, 1968).

e) Sensores catalíticos: são sensores que através de um filamento aquecido, junto a um elemento catalisador que acelera as reações químicas reage com o gás de interesse gerando uma diferença de potencial proporcional à concentração.

2.1.5 Aspectos Microbiológicos da Biofiltração

A capacidade de degradação depende das características dos contaminantes e da diversidade dos microrganismos envolvidos dentro do biofiltro, assim como o tempo de permanência de contato com o meio (Janke, 2002).

A reação bioquímica que ocorre dentro do biofiltro é mostrada abaixo:

Biomassa + Substrato = Biomassa + CO_2 + H_2O;

Em que:

-Biomassa: representa as células e o seu crescimento celular (aumento do número de células)

-Substrato: representa o meio de cultura dentro do biofiltro onde se fornecem os nutrientes necessários para a sobrevivência e reprodução dos microorganismos. Como parte do substrato, os gases contaminados levarão o carbono, que servirão como fonte de alimento secundária aos microrganismos.

-Dióxido de carbono (CO_2) e água (H_2O): são subprodutos gerados pela respiração celular.

A Ilustração 2-1 mostra um exemplo das fases de crescimento microbiano que serão descritas a seguir.

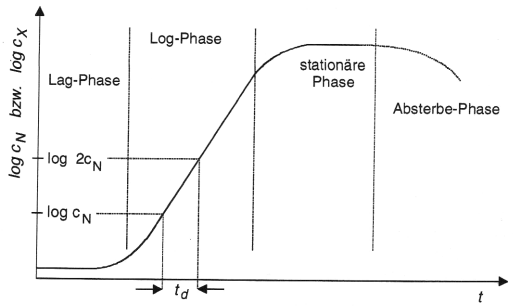


Ilustração 2-1. Fases do crescimento microbiano ao decorrer do tempo [Janke,2002]

Segundo Schwister (2003), cabe salientar que existem basicamente as seguintes etapas de crescimento:

- A fase LAG: início da divisão celular, adaptação ao meio de cultura e aumento lento da biomassa microbiana;

- A fase LOG: crescimento exponencial da biomassa microbiana e divisão celular, após adaptação do meio;

- A fase de TRANSIÇÃO: fase limite de crescimento microbiano e diminuição dos nutrientes no meio;

- A fase ESTACIONÁRIA: a massa de microrganismos alcança seu máximo valor, a taxa de crescimento e de morte alcança seu valor de igualdade;

- A fase de LETAL: os microrganismos não possuem mais espaço nem nutrientes para continuar o crescimento;

3. MATERIAIS E MÉTODOS

Este capítulo apresenta o desenvolvimento do analisador de CO_2 que passou por várias etapas e correções, e para isso foram utilizados vários materiais auxiliares tais como termômetros, compressor, fluxímetro, padrões gasosos e calibradores portáteis. Estes materiais possibilitaram os ajustes necessários em hardware e software, para o devido funcionamento do equipamento.

3.1. Descrição Geral do Sistema

O sistema completo compreende de sensores de temperatura e dióxido de carbono, placas eletrônicas para amplificação e interface com o computador, *software* para aquisição de dados e calibração dos sensores, montagem do sistema mecânico para simular um biofiltro.

A primeira etapa foi o desenvolvimento do *hardware* de leitura dos sensores de CO_2 e temperatura, que amplifica e converte os dados que são enviados ao computador pela porta serial. O *hardware* foi desenvolvido com duas saídas de relé eletromecânico, sendo uma delas utilizada para acionar uma mini bomba da água, com a função de colocar o substrato para dentro do biofiltro, e outra saída para retirar, através de um solenóide, o excesso de biomassa celular das bactérias cultivadas no biofiltro.

A segunda etapa compreende no desenvolvimento do *software* em linguagem C para o micro controlador PIC16F916 da Microchip Inc., com a função de ler as duas grandezas analógicas e convertê-las para digitais através de um conversor analógico para digital (ADC). Possui também a função de monitorar através de interrupções de recepção serial os comandos enviados pelo computador para efetuar a leituras dos sensores, e comandar o acionamento dos relés.

Com o decorrer do projeto, foi adicionado o sensor de temperatura no hardware, utilizando outro canal analógico do micro controlador, sendo necessários também sua amplificação e condicionamento adequado do sinal

deste sensor. Mostra-se abaixo na Ilustração 3-1 o diagrama em blocos final do hardware desenvolvido.

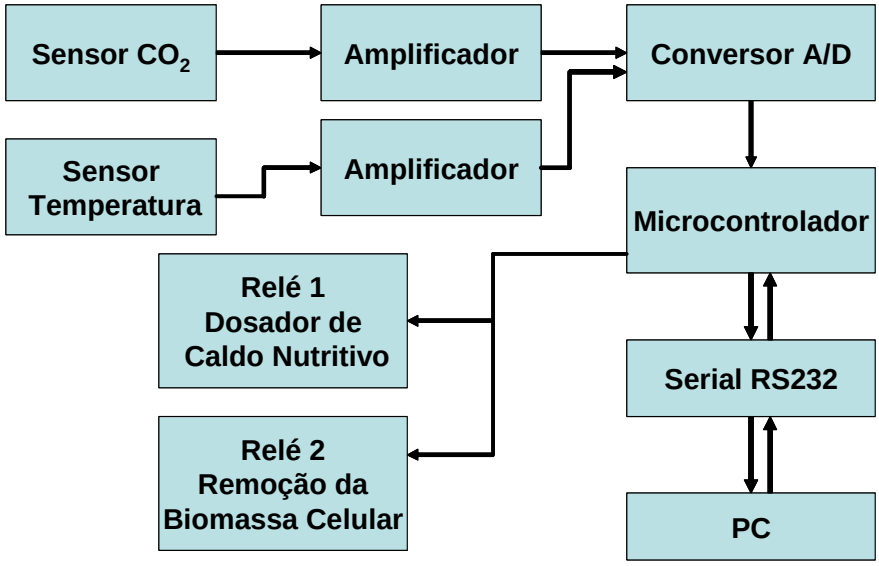


Ilustração 3-1 Diagrama final do hardware com o sensor de temperatura adicionado.

A terceira etapa compreendeu no desenvolvimento do software em linguagem C# da Microsoft Inc. Com ele foi implementada a comunicação usando a porta serial do computador, o ambiente gráfico para leitura dos dados, a calibração, os gráficos com as variáveis dos dois sensores utilizados no experimento, o acionamento manual do sistema e os parâmetros de conexão com banco de dados para armazenar os dados adquiridos. Na Ilustração 3-2 tem o diagrama em blocos simplificado de todo o *software* elaborado para o analisador.

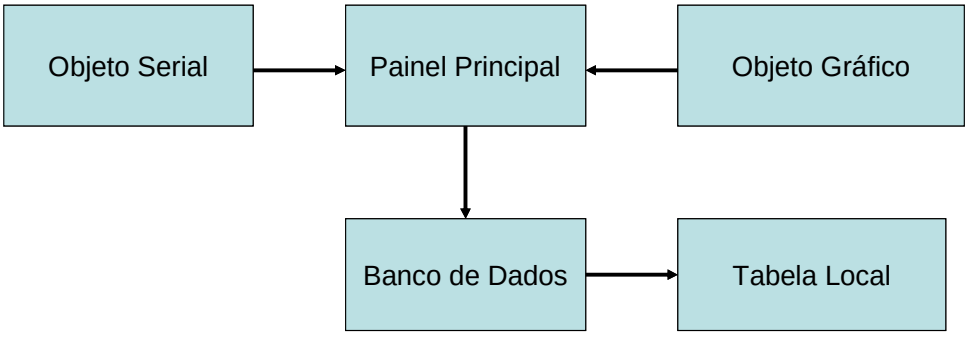


Ilustração 3-2 Diagrama em blocos do Software em C#.

A quarta etapa compreendeu o desenvolvimento das tabelas, comandos e conexão com o banco de dados utilizando a linguagem estruturada MYSQL da empresa Oracle Inc. A aquisição dos dados pode ser

guardada em tabelas devidamente planejadas e desenvolvidas para fornecer a possibilidade de pesquisa dos dados, para posterior análise dos experimentos.

A quinta etapa compreendeu o desenvolvimento do biofiltro e acessórios para o crescimento dos microorganismos, o controle de temperatura, o controle dos gases, o fornecimento de oxigênio para a sobrevivência dos microrganismos, assim como toda a estrutura mecânica que auxiliava o experimento. Foi utilizada uma fonte de tensão de 250 W para o acionamento das válvulas, ao ventilador para homogenização da temperatura e para alimentar os circuitos do *hardware*. A Ilustração 3-3 mostra uma visão geral do experimento realizado.

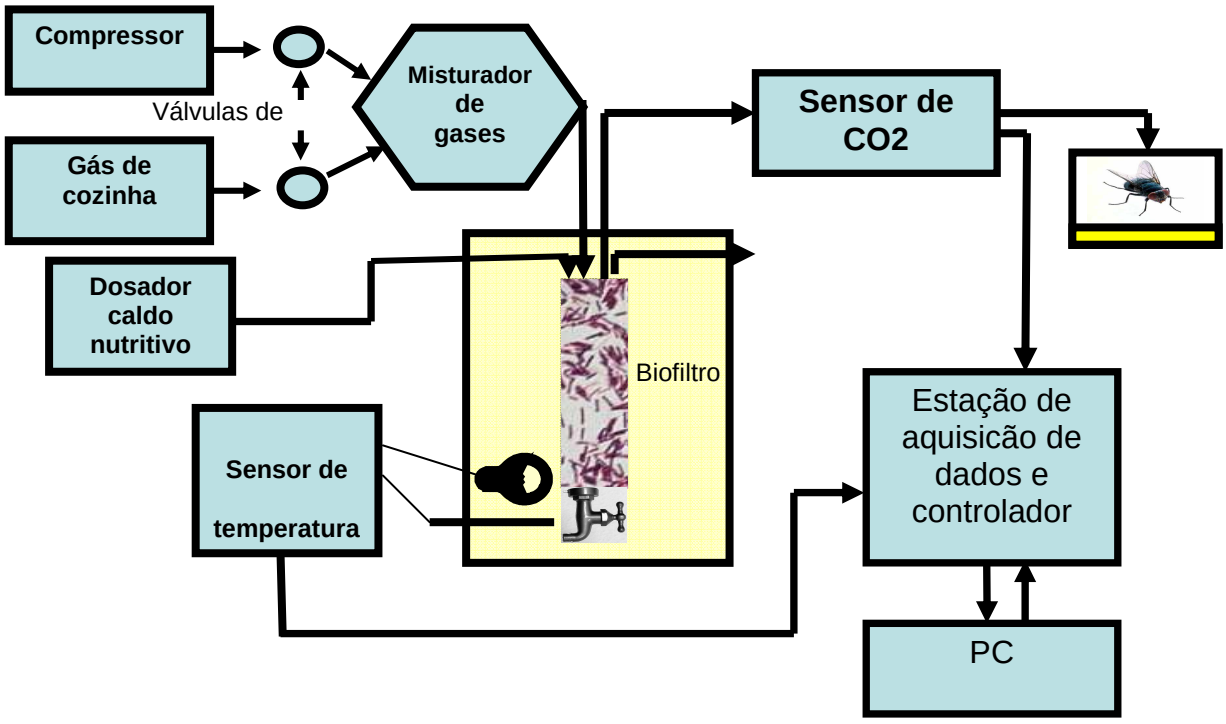


Ilustração 3-3 Diagrama geral do sistema

3.2. *Descrição dos Sistemas Mecânicos*

O sistema completo compreende de um compressor de ar, um butijão de gás de cozinha, válvulas para mistura gasosas, estufa, resistência, sensor de temperatura e CO_2, placa eletrônica para aquisição e comunicação com o computador.

A Ilustração 3-4 o fluxo dos gases é controlado por válvulas e reguladores de pressão, passando por um misturador de gases que tem a função de homogenizar a mistura, seguindo para o biofiltro com o controle da temperatura. Antes de entrar no biofiltro, estes gases passam por uma serpentina que condiciona termicamente o gás para que o mesmo não influencie no crescimento microbiano com o choque térmico, perturbando a temperatura ótima de crescimento. Segundo Ola et al. (2010) a temperatura ótima para o crescimento de microrganismos é de 35 °C, a qual foi escolhida para realização de todo experimento. Após este pré-aquecimento o gás chega ao reator onde é realizado o controle da temperatura através de um sensor e lâmpada de 100 W. Foi utilizado um dosador de água e caldo nutritivo para controlar a umidade e disponibilidade de meio nutritivo para suportar o crescimento microbiológico, pois na falta do carbono proveniente do gás contaminado este permite a sobrevivência dos microorganismos.

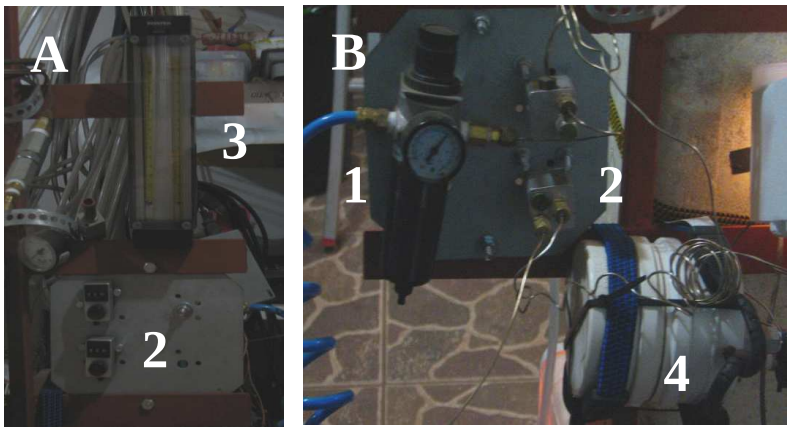


Ilustração 3-4 (A e B)– Válvula de pressão (1); Válvulas de fluxo (2); Rotâmetro (3); Misturador de gases (4).

O gás contaminado, que neste caso é para efeitos de simulação e testes, constitui-se uma mistura de gás de cozinha com ar comprimido (ver Ilustração 3-5, foi aplicado a um reator desenvolvido, utilizando válvulas de fluxo e válvulas de controle de pressão, para mantermos o controle da vazão e concentração do ar contaminado constante (Ilustração 3-5). Cabe salientar que o gás de cozinha GLP (gás liquefeito de petróleo) é uma mistura de 50% de propano e 50% de butano aproximadamente, sendo adicionados outros gases em menor concentração (mercaptanos). Esses gases sulfurados são

adicionados para promover segurança em caso de vazamento e ser detectado antes que algum acidente aconteça (MELO, 2011).

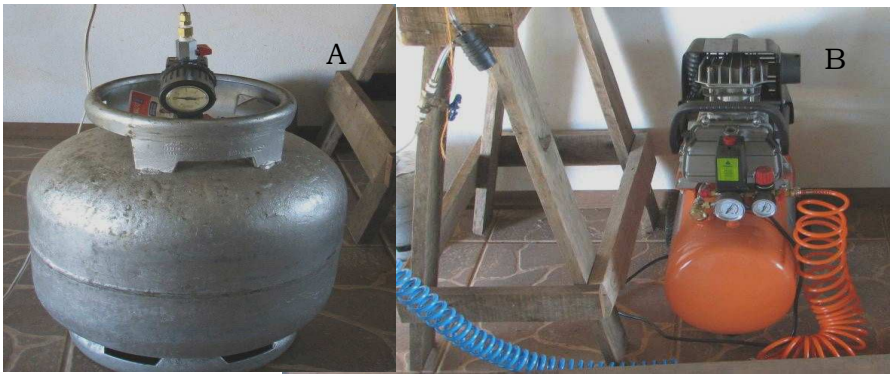


Ilustração 3-5 Gás de cozinha (A); Compressor (B)

A seguir na Ilustração 3-6 pode-se ver em detalhes o sistema de entrada dos gases no sistema, tanto a entrada do ar comprimido como a entrada do gás de cozinha, em conjunto com os reguladores de pressão.

Ilustração 3-6 Entrada do gás de cozinha e ar-comprimido

Com o auxílio de um fluxímetro digital e um rotâmetro (Ilustração 3-7) foi medido o fluxo dos gases, e construída a curva de calibração, para auxiliar os testes pilotos. Assim, com base nesse instrumento pode-se medir, testar e simular diferentes concentrações do gás contaminado que vai para o biofiltro.

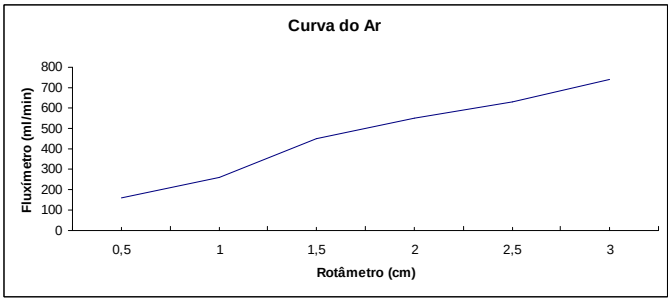
Ilustração 3-7 Na esquerda o Rotâmetro e direita o fluxímetro

Após testes e ajustes de conexões e correções de vazamento no sistema, levantou-se uma curva de calibração para o ar comprimido e gás de cozinha. Com os equipamentos de medição de fluxo, foram obtidos os dados demonstrados na Tabela 3.1 e Ilustração 3-8 para o ar-comprimido e na Tabela 3.2 e Ilustração 3-9 para o gás de cozinha. Cabe salientar que o rotâmetro é um instrumento de medição, muito utilizado em instrumentação de processo, composto de um tubo de vidro graduado em centímetros, que possui internamente uma esfera, que flutua e alcança uma altura proporcional ao fluxo utilizado e depende da densidade do gás utilizado. Por isso, foi utilizado um fluxímetro digital que funciona por um princípio que não depende da composição do gás, podendo assim realizar a relação de calibração com o rotâmetro e o fluxo real dos gases. O rotâmetro auxilia também a verificação visual se ocorreu algum desvio com o experimento, que, por exemplo, pode ter tido um vazamento ou rompimento de alguma mangueira.

Tabela 3.1 Dados de calibração para válvula de fluxo do ar-comprimido

Curva de Ar		
Pressão(PSI)	**Rotâmetro Pequeno(cm)**	**Fluxímetro(ml/min)**
80	0,5	162
80	1	261
80	1,5	450
80	2	550
80	2,5	630
80	3	742

Ilustração 3-8 Curva de calibração para válvula de fluxo do ar-comprimido

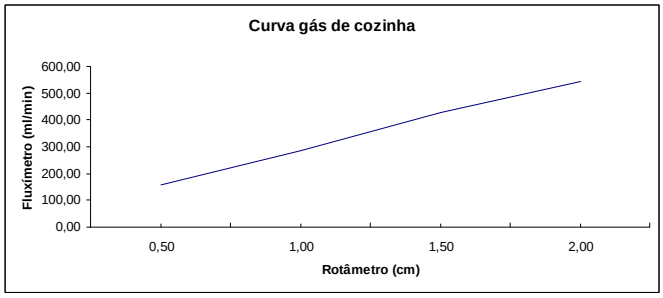


Foi utilizado em todos os testes o fluxo de 185 ml/min +-5% de ar comprimido, medindo na saída do biofiltro a vazão com o auxílio do fluxímetro digital. Não foi colocado o gás de cozinha no experimento, pois o sensor de CO_2 utilizado possui um filamento aquecido no seu interior que poderia trazer risco de explosão durante o experimento. O gás de cozinha seria uma fonte alternativa de energia (carbono) para os microorganismos, simulando os gases tóxicos de aterros sanitários.

Tabela 3.2 Dados de calibração para válvula de fluxo do gás de cozinha

Curva do Gás e Cozinha		
Pressão(PSI)	**Rotâmetro Pequeno(cm)**	**Fluxímetro(ml/min)**
80	0,5	157
80	1	285
80	1,5	427
80	2	543

Ilustração 3-9 Curva de calibração para válvula de fluxo do gás de cozinha



Para o controle da temperatura do biofiltro foi desenvolvida uma estufa, na qual foram utilizadas duas caixas de isopor fixadas por arames, recortadas e revestidas com papel alumínio para reflexão do calor. Foi utilizado um sensor de temperatura do tipo PT100, uma lâmpada de 100 W como elemento aquecedor do biofiltro, um controlador de temperatura da empresa NOVUS modelo N480 (Ilustração 3-10) para ajuste e controle da

temperatura para o crescimento dos microrganismos. Cabe salientar que o sensor de temperatura (LM35) usado, tem o intuito de registrar os dados para análise posterior ou a detecção de algum desvio, como por exemplo, a queima da lâmpada. Não foi realizado o controle de temperatura pelo projeto por não fazer parte do experimento proposto, tendo a função principal de ser um analisador de dados.

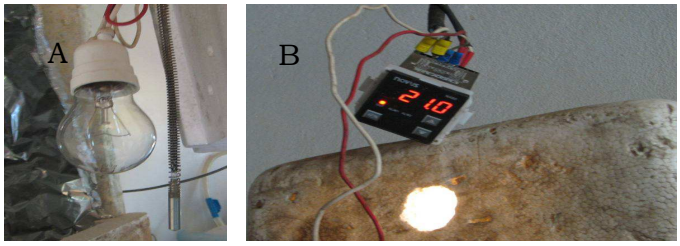


Ilustração 3-10 Elementos de controle de temperatura. A)sensor e lâmpada; B)controlador de temperatura

A estufa com o primeiro protótipo do biofiltro pode ser visualizada na Ilustração 3-11. O biofiltro consiste externamente de um tubo de polipropileno de 100 mm de diâmetro, com dois “caps.” (encaixe terminal para fechamento), dois anéis de vedação de borracha e conexões para encaixe da tubulação. A tubulação utilizada é de cobre com o diâmetro de 1/8”, sendo que antes de entrar no biofiltro foram dadas algumas voltas do tubo de cobre, para que o gás pré-aqueça, condicionando o gás na temperatura da estufa antes da entrada no biofiltro e não causar um choque térmico na colônia de microrganismos em estudo, que prejudique seu crescimento dentro do biofiltro.

Ilustração 3-11 Estufa (1) e primeiro protótipo do biofiltro(2).

Com o passar dos testes foram desenvolvidos alguns protótipos de materiais que ficam dentro do biofiltro, tais como utilizar rochas e serragem (madeira) como suporte ao crescimento dos microrganismos. No ultimo protótipo do biofiltro não foi usado nenhum material de suporte ao crescimento, pois o caldo nutritivo não era recirculado, permanecendo os microrganismos somente no meio líquido.

Um dos protótipos do biofiltro, representada pela Ilustração 3-12, utilizou-se um ventilador de fonte de computador (n° 1) para homogenizar o ar quente provindo do sistema de aquecimento para o controle preciso da temperatura dentro das caixas de isopor. Com referência a Ilustração 3-12, a entrada dos gases é indicada pelo n° 2, a saída do biofiltro pelo n° 4, a entrada do caldo nutritivo e água pela tubulação do n° 3.

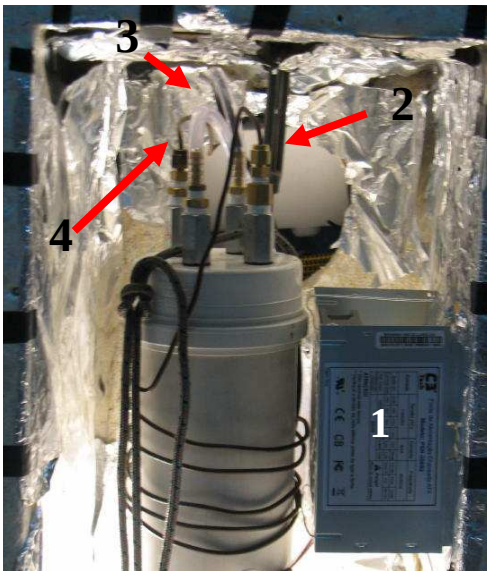


Ilustração 3-12 Biofiltro e ventilador para homogenizar temperatura. 1)Ventilador; 2)entrada dos gases após serpentina; 3)entrada do caldo nutritivo; 4)saída do biofiltro.

O biofiltro possui uma entrada para colocação de solução nutritiva, como colocado anteriormente, na qual é acoplado um reservatório com bomba e solenóide. O acionamento era feito através do sistema pelo relé e comandado manualmente pelo *software* no computador. Pode-se visualizar o reservatório na Ilustração 3-13. Referente a esta figura o número 1 é o reservatório, o número 2 é a fonte de alimentação, que fornece energia para

a placa de controle, controlador de temperatura e para o acionamento do solenóide do reservatório.

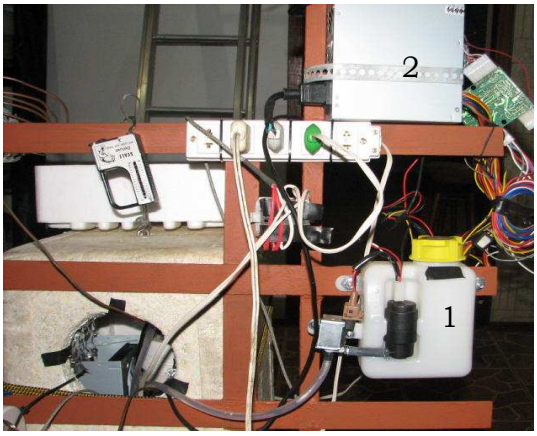


Ilustração 3-13 Reservatório para introdução de caldo nutritivo e água. 1) Reservatório; 2) Fonte de alimentação

Assim foi montado o sistema completo para controle do crescimento dos microrganismos, constituindo de um sistema misturador de gases, um sistema de pré-aquecimento, uma estufa com temperatura controlada e um biofiltro para crescimento dos microrganismos, com entrada de solução nutritiva para fornecimento de umidade e disponibilidade de alimento. Na saída do biofiltro foi colocada uma conexão para adaptar o sensor de CO_2 e no interior das caixas de isopor foi colocado o sensor de temperatura LM35. Pode-se visualizar a montagem do protótipo do sistema completo na Ilustração 3-14, mostrando o sistema com a estufa aberta e com a estufa fechada.

Ilustração 3-14 Sistema completo do experimento

Para que o sensor não sofresse obstrução pela saída de microrganismos das tubulações foi desenvolvido um sistema de desvio de fluxo de gases, onde uma das válvulas é aberta para direcionar o gás de dentro do sistema para o sensor de CO_2. Assim na Ilustração 3-15 podem-se visualizar as válvulas (1) e a posição do sensor de CO_2 (2) onde o fluxo de gases entra diretamente dentro do sensor.

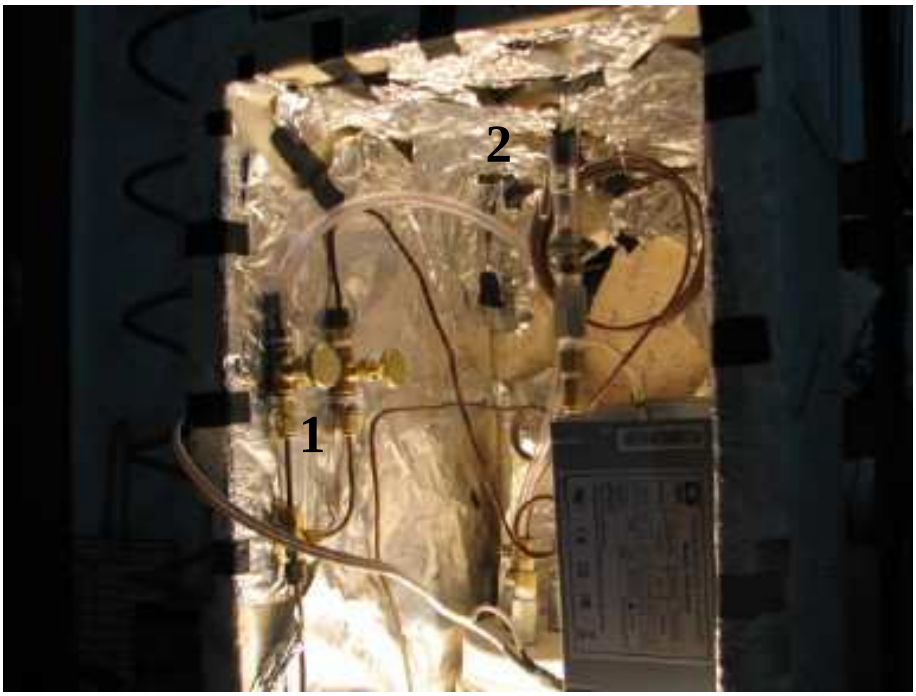


Ilustração 3-15 Válvulas para direcionar o fluxo de gases (1) e para o sensor de CO_2 (2).

Em todos os procedimentos no biofiltro foi utilizada água destilada e esterilizada, o caldo nutritivo foi realizado utilizando o reagente ACUMEDIA do laboratório (Nutrient Broth, marca NEOGEN), que é recomendado para o cultivo de uma grande variedade de microrganismos. Foi retirada do rótulo a receita que consistia em pesar numa balança analítica 8 g de meio em 1 L de água aquecendo com agitação manual até dissolver todo o pó do produto. Foram colocados dois litros desta solução e verificado ao longo do tempo a variação da concentração de CO_2, sendo realizado o experimento em batelada. Cabe salientar, segundo LEVENSPIEL (2000), os reatores descontínuos (batelada) são simples, necessitam poucos acessórios e são ideais em estudos de pequena escala.

3.3. Descrição dos Sistemas Eletroeletrônicos

O hardware desenvolvido utiliza um microcontrolador PIC16F916. Este microcontrolador de 8 bits, possui diversos periféricos, tal como portas digitais e analógicas, conversores A/D e contadores de tempo (TIMERs).

Foram utilizadas 5 portas digitais, sendo duas saídas para acionamento dos dois relés, uma saída para a transmissão serial (TX), uma saída para acionar o filamento do sensor de CO_2 e uma entrada para a recepção serial (RX).

Foi utilizado um conversor serial padrão RS232 para realizar a comunicação ponto a ponto. Este tipo de comunicação, segundo COCIAN, é considerado como desbalanceado e utiliza dois fios referenciados a terra, onde as tensões produzidas pelo driver aparecem através de um par de linhas de sinal que transmitem somente um sinal.

Foram utilizadas duas portas analógicas para realizar a conversão dos dois sensores empregados. Um dos sensores é o sensor de temperatura LM35 que após condicionamento de sinal é direcionada na entrada analógica RA1. O sensor de dióxido de carbono (CO_2) que após amplificação e filtragem de ruído é direcionado à entrada analógica RA0 da porta A (PORTA) do microcontrolador.

3.4. O Conversor A/D

O conversor analógico digital converte a grandeza analógica para uma representação de uma variável de 10 bits, e utiliza os registradores ADRESL e ADRESH para armazenar o valor da conversão. Sendo o valor mais significativo registrado no ADRESH e o valor menos significativo registrado no ADRESL.

Este conversor gera um resultado de 10 bits baseando-se no princípio de um conversor de aproximações sucessivas, guardando o resultado de conversão nos registradores utilizando um circuito "*sample and hold*". Pode-se visualizar melhor o circuito na Ilustração 3-16, onde mostra os registradores associados, assim como os bits de configuração para habilitação e controle do hardware.

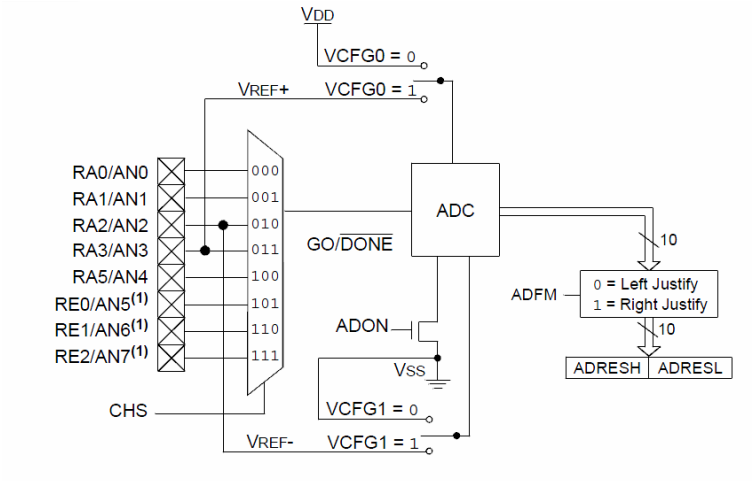


Ilustração 3-16 Diagrama em Blocos do Conversor A/D [datasheet PIC16F916]

Para a utilização do conversor é necessário configurar os registradores da porta A, selecionar os canais analógicos para o uso nos dois sensores, selecionar a tensão de referência, fonte do *clock* de conversão, controle das interrupções e leitura do resultado da conversão.

A tensão de referência do conversor analógico para digital foi definida em 5 V. Por ser um conversor com resolução de 10 bits calculam-se os passos da conversão da seguinte maneira:

2^{10bits}=1024 intervalos de conversão, sendo:

$V_{a/d}$ = 5 V / 1023;

$V_{a/d}$ = 0,00488 mV, resolução da conversão

O valor convertido (de analógico para digital), que se encontra nos registradores ADRESH e ADRESL segue o seguinte cálculo:

$V_{a/d}$ = 0,00488 mV * (ADRESH+ADRESL);

Como utilizamos um cristal de 20Mhz, e configuração do registrador ADCON1 para uma freqüência de conversão de 625Hz. O tempo para uma aquisição de 10 Bits é chamada de T_{ad} .Assim o tempo de aquisição total necessita de 11 periodos de T_{ad}.

A Ilustração 3-17 mostra o ciclo completo mínimo de uma convesão analógica para digital para cada entrada analógica que estão na Porta A (RA0 e RA1).

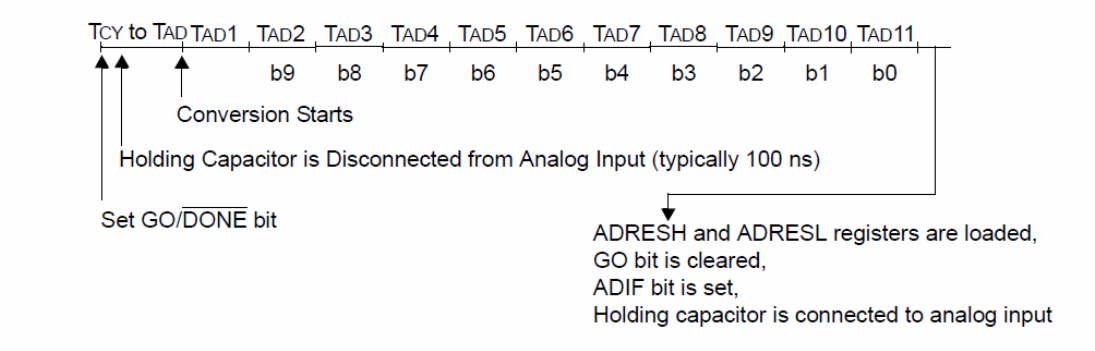


Ilustração 3-17 Ciclo Completo de Conversão A/D para cada Canal [datasheet PIC16F916.]

3.5. *As Saídas e Entradas Digitais*

Através da correta configurações dos registradores foram selecionadas quatro saídas digitais e uma entrada digital. Duas saídas compreendem o acionamento dos relés através do acionamento na base do transistor na placa eletrônica, uma saída para a de transmissão serial (TX) de dados, uma saída que comanda o filamento do sensor de CO_2. Foi utilizada apenas uma entrada para a recepção serial (RX) de dados. Deixou-se na reserva mais uma entrada para implementação futura de um alarme de limite de CO_2, para acionar, por exemplo, uma mensagem no computador ou um alarme visual.

As portas admitem tensões na faixa de 0 a 5 V, e cada porta tem seus registradores associados e seus bits de configuração.

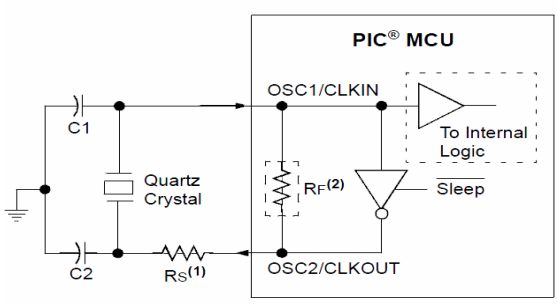
Para o acionamento do filamento foi utilizado o pino do microcontrolador RB2, utilizando a porta B. O Pino RB3 foi usado para um desenvolvimento futuro de limite de alarme.

Os Pinos de recepção RX (entrada digital) e os pinos de transmissão TX (saída digital) serão descritos na seção 3.7.

3.6. *O Cristal Ressonador*

Para o correto funcionamento do microcontrolador e precisão nos tempos de processamento, conversão e comunicação, foi escolhido o uso de um cristal ressonador na freqüência de 20 Mhz. A Ilustração 3-18 mostra o diagrama do circuito utilizado no hardware, no qual foram utilizados dois capacitores de 33 pF de cerâmica (C1 e C2), permitindo que o circuito

oscilasse a 20 Mhz com precisão. Não foi utilizado resistor em série (Rs) no circuito.



Note 1: A series resistor (RS) may be required for quartz crystals with low drive level.

2: The value of RF varies with the Oscillator mode selected (typically between 2 MΩ to 10 MΩ).

Ilustração 3-18 Circuito do Cristal utilizado no Microcontrolador [datasheet PIC16F916.]

3.7. *Serial AUSART (Adressable Universal Synchronous Asynchronous Receiver Transmitter)*

Foi utilizado o modo de comunicação serial do microprocessador pelos pinos RC6 e RC7 da porta C, sendo necessária a configuração dos pinos e registradores envolvidos para o correto funcionamento. Foi utilizado o modo assíncrono do receptor e transmissor serial. Este periférico possui todos os geradores de temporização, registradores de deslocamento e *buffers* de dados necessários à transferência dos dados seriais, independente da execução do programa no microcontrolador.

Para que a taxa de transmissão e recepção permanecessem sincronizadas com a do computador, foi necessária a configuração do registrador (SPBRG) gerador de taxa de transmissão e recepção (*Baud Rate*). Foi estipulada uma taxa de 9600 bits por segundo, comunicação a 8 bits, sem paridade e 1 bit de início e de parada. Seguindo dados do manual do fabricante os parâmetros são os seguintes:

F_{osc} = 20 Mhz;

$F_{baud\ rate}$ = 9600 bps;

Pelo manual do fabricante:

$F_{baud\ rate\ desejado} = F_{osc} / 64*(SPBRG + 1)$;

Isolando SPBRG:

SPBRG = 31,55;

Como pode-se apenas usar números inteiros dentro deste registrador, coloca-se o valor de 31, assim com este valor é possivel calcular o valor exato e o erro da precisão do baud rate, segue abaixo:

$F_{baud\ rate\ calculado} = F_{osc} / 64*(SPBRG + 1)$;

$F_{baud\ rate\ calculado}$ = 9765,62 bits por segundo ;

Como o erro é:

$Erro = (F_{baud\ rate\ calculado} - F_{baud\ rate\ desejado}) / F_{baud\ rate\ desejado}$

Erro= 1,7%

Assim o erro está dentro da faixa aceitável para um bom sincronismo entre nosso hardware e o computador.

A Ilustração 3-19 e a Ilustração 3-20 mostra o diagrama de bits e registradores de configuração envolvidos na transmissão e recepção serial da porta c.

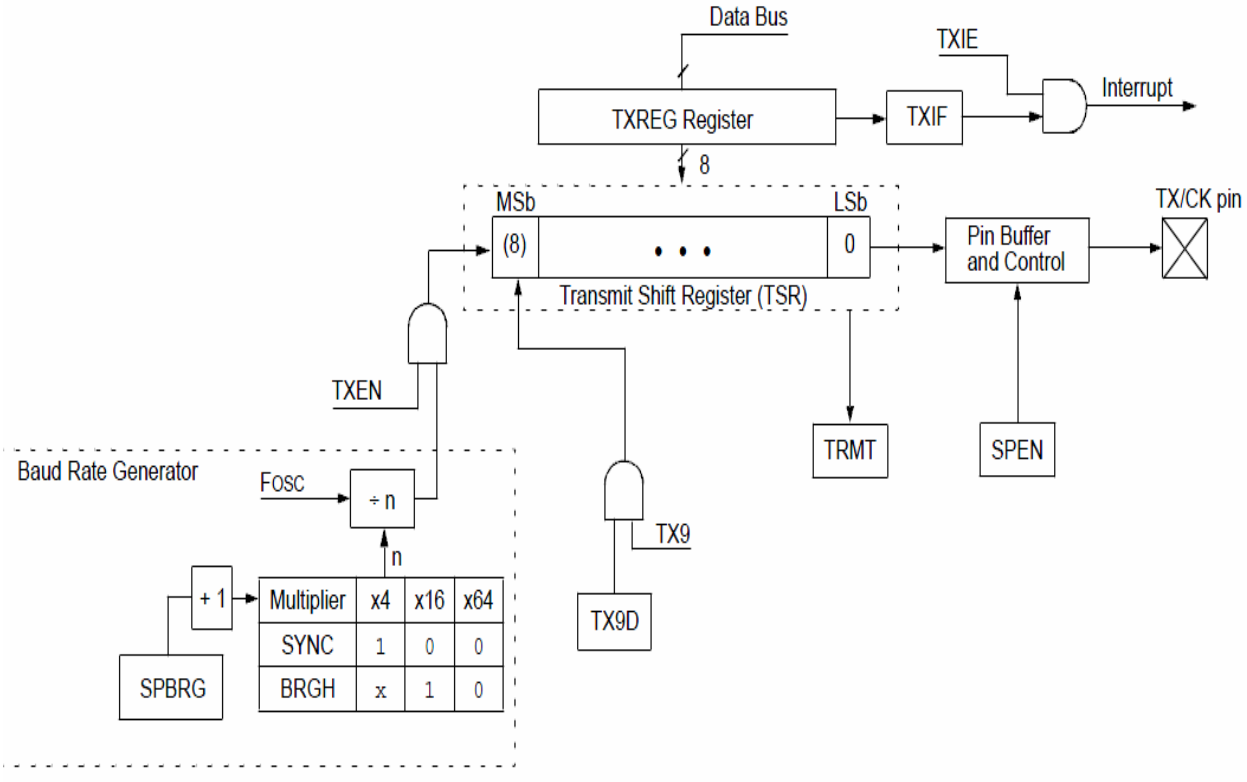


Ilustração 3-19 Bloco do Pino RC6 de Transmisão TX [datasheet PIC16F916]

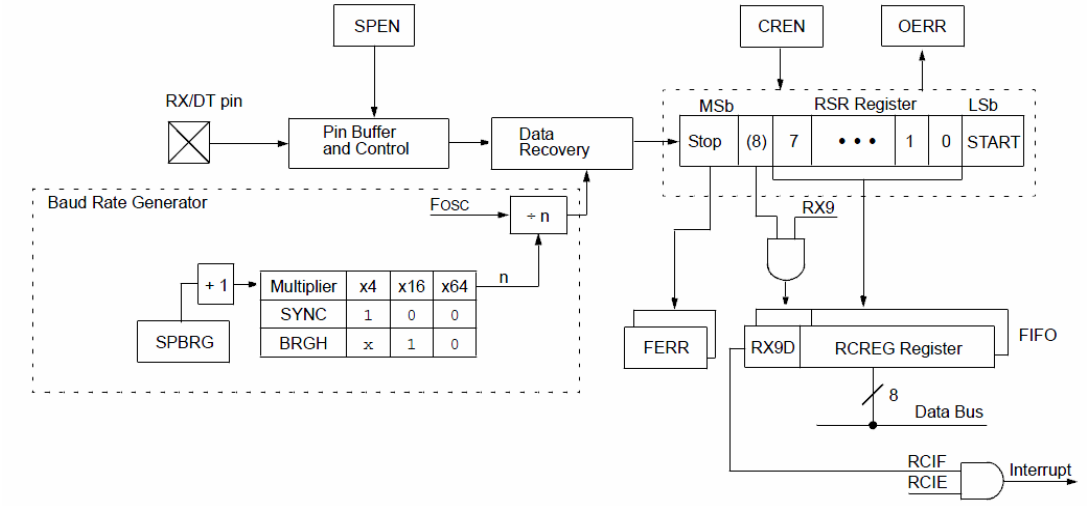


Ilustração 3-20 Bloco do Pino RC7 de Recepção RX [datasheet PIC16F916.]

Foi configurada a interrupção serial para verificar os dados recebidos, entre a porta serial do computador e a placa. Assim pela interface de software desenvolvida no computador foi possível realizar a comunicação de dados.

3.8. Sistema Eletrônico

Para o desenvolvimento do sistema eletrônico e a aquisição de dados para o experimento, foi planejado a comunicação serial através do componente da National Inc. modelo MAX232, que converte os níveis de tensão do microcontrolador para os níveis de tensão do padrão RS232 (Ilustração 3-21) que é ligado ao computador.

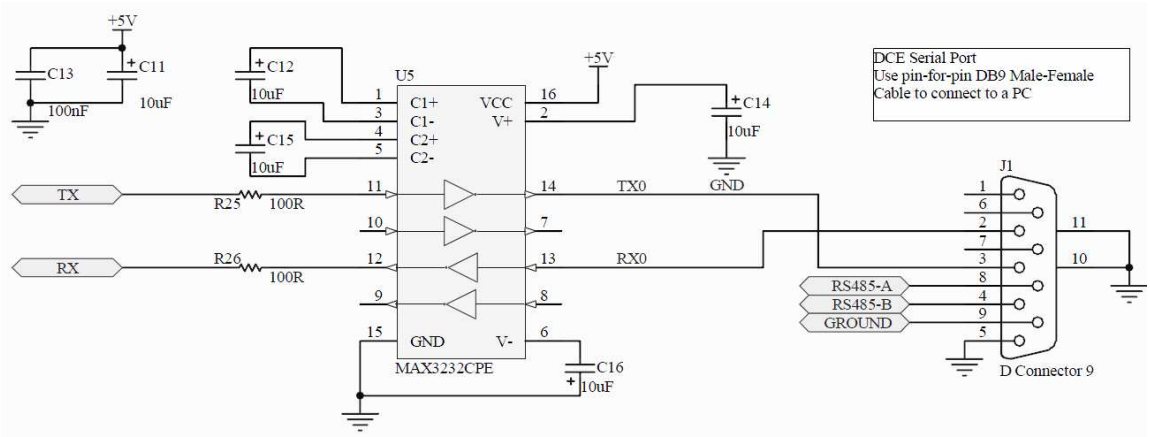


Ilustração 3-21 Circuito do Conversor Serial na Placa

Demonstra-se na Ilustração 3-22 o circuito principal planejado inicialmente para atender requisitos globais do projeto sem incluir as

devidas amplificações e condicionamento dos sinais dos sensores que serão posteriormente esclarecidas.

-Duas saídas utilizando relés eletromagnéticos

-Circuito de proteção contra tensão de pico reverso dos relés

-Entradas Digitais

-Duas entradas para o conversor analógico para digital.

-Alimentação de 12 V

-Fonte de 5 V

-duas entradas opto-acloplad as

-extensão futura para padrão RS485

-3 conectores para expansão de periféricos que foram usados para acoplar sensores e pino de controle do filamento do sensor de CO_2 e pino de alarme limite de CO_2 que será utilizado para desenvolvimento futuro.

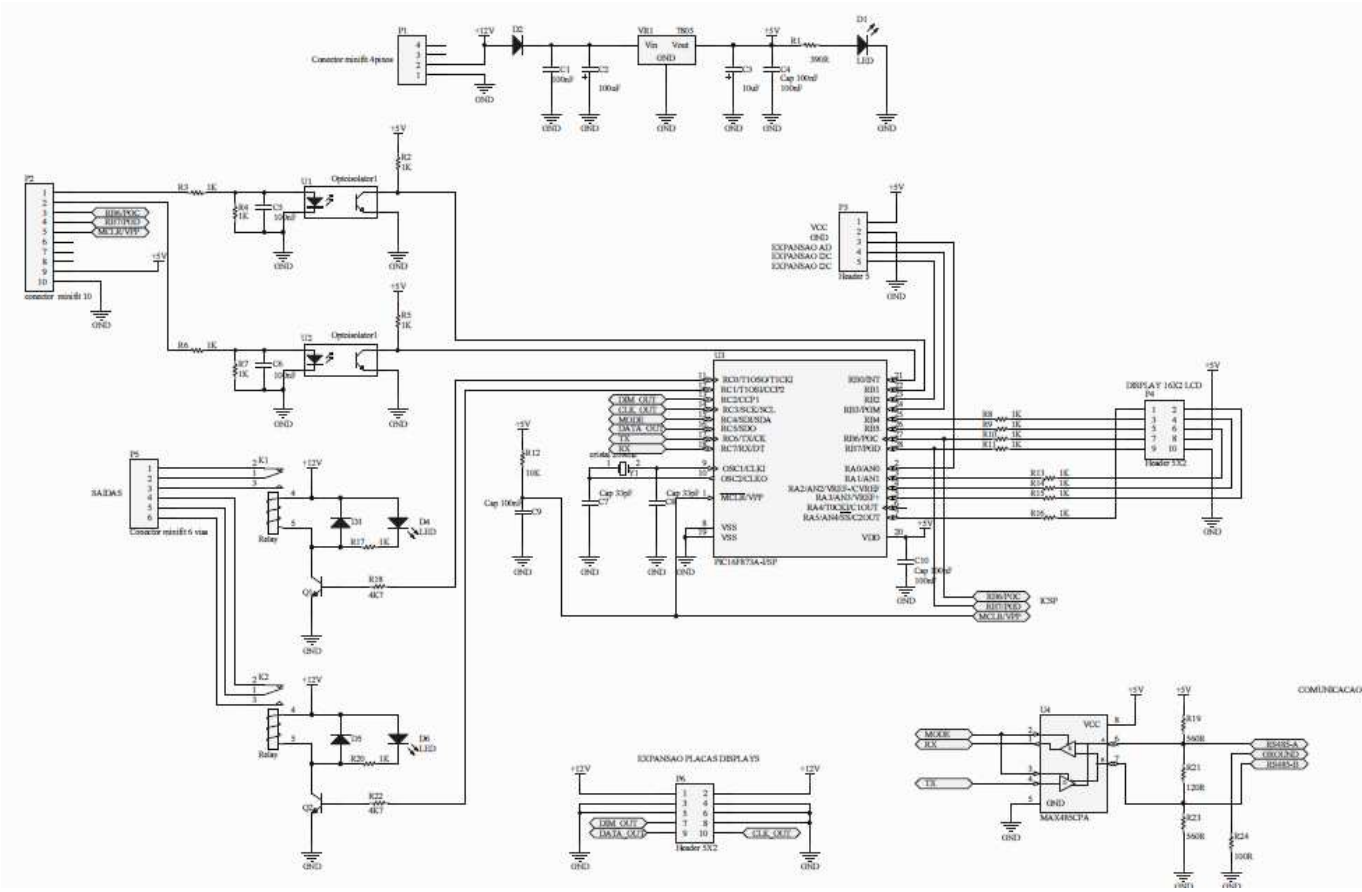


Ilustração 3-22 Circuito da Placa Completa Sem Conversor Serial

Após o uso de software Altium 2008, de desenvolvimento de placas de circuito impresso, foi realizado o roteamento do circuito a partir do esquemático mostrado anteriormente, gerando os arquivos de produção e enviado para empresa Micter S.A da cidade de Cachoeirinha. A Ilustração 3-23 mostra à esquerda placa produzida e a direita placa montada completa.

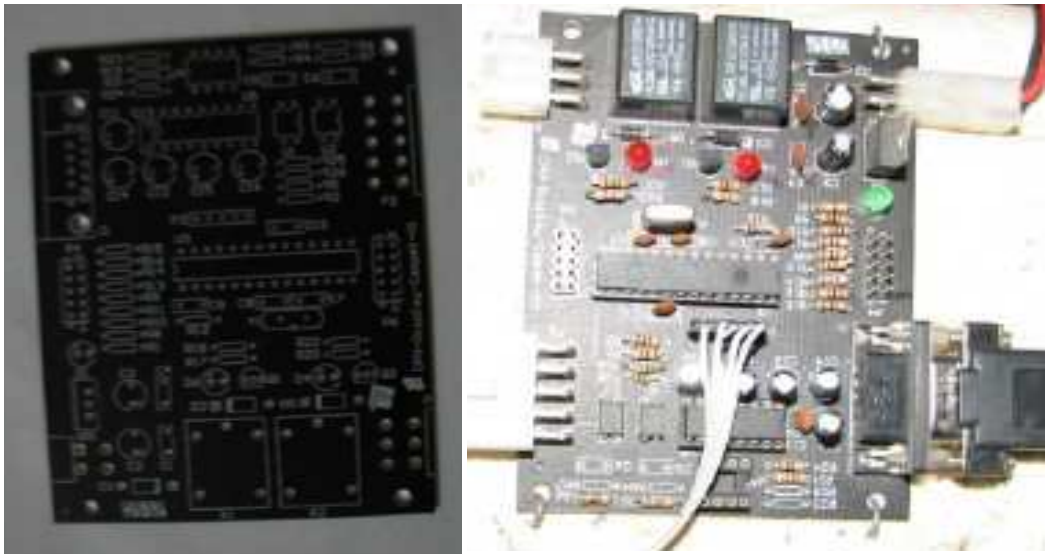


Ilustração 3-23 Fotos placa Completa

3.9. Sensor de temperatura LM35

Para a medição da temperatura foi utilizado o componente LM35, da National Instruments inc., que fornece uma saída linear proporcional à temperatura em graus Celsius. Este sensor trabalha com correntes da ordem de 60 pA, com baixo aquecimento próprio, fornecendo um boa precisão a temperatura ambiente (25^{0}C) em torno de +- 0,25^{0}C, e de 0,75^{0}C em toda sua escala de leitura que compreende de -55 ^{0}C à +150 ^{0}C.

No canal analógico, da porta RA1 do microprocessador foi colocado o circuito condicionador de sinal do sensor de temperatura. A Ilustração 3-24 mostra o sensor com o conector para o acoplamento adequado à placa condicionadora de sinal.

Ilustração 3-24 Sensor de Temperatura LM35 montado

O sensor é alimentado em 5 V, possui uma resposta amplificada fornecendo em sua saída uma tensão proporcional linear de 10 mV / ^{0}C como mostra a Ilustração 3-25.

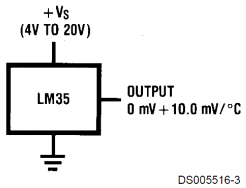


FIGURE 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)

Ilustração 3-25 Diagrama Sensor LM35 [datasheet]

3.10. Placa e Circuito Amplificador Sensor de Temperatura

Foi selecionada a faixa de 0 ^{0}C até 100 ^{0}C, por se tratar de um reator microbiológico e exigir temperaturas dessa ordem, sendo utilizado o valor de 35 ^{0}C para todo os experimentos. Cabe salientar que segundo Ola et al. (2010) a temperatura ótima para o crescimento de microrganismos é de 35 °C.

Assim definindo-se a faixa do sensor de tem-se:

$V_{a/d}$ = 5 V/ 1024;

$V_{a/d}$ = 0,00488 mV, por passo de conversão

Como se tem uma resposta do sensor de 10 mV / °C, em 100 °C tem-se uma saída de 1 V do sensor. Na **Erro! Fonte de referência não encontrada.**, utilizando um dos circuitos operacionais do circuito integrado MCP619, com ganho de 3,703 para ajustar a precisão do conversor A /D de 10 bits , que compreeende uma faixa de tensão de 0 à 5 V.

Pela equação do ganho Av da configuração usada de um amplificador não inversor tem-se:

Av = (1 + R2/R1);

Escolheu-se:

R2 = 10KΩ;

R1 = 2K7Ω;

Então:

Av = (1+2,703);

Av = 3,703;

Tem-se:

$V_{por\ ^0C}$ = 3,7Vdc / 100 ^{0}C =37mV / ^{0}C

Como a resolução de conversão A/D é de 4,88mV e o erro máximo é de +- 2,44mV, precisava-se de 15 passos do conversor para alterar um grau celsius na saída, ou seja, 0,06 ^{0}C por passo de conversão do conversor analógico para digital.

A Ilustração 3-26 mostra o circuito montado para realizar o condicionamento de sinal do sensor de temperatura.

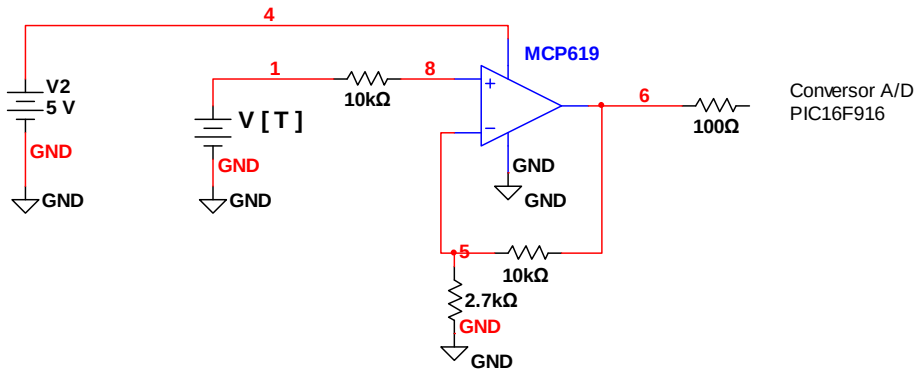


Ilustração 3-26 Amplificação de sinal do sensor LM35

Com base nas simulações feitas no programa Multisim, foi realizado o desenvovimento de uma placa e realizado testes. A Ilustração 3-27 mostra a placa final do *hardware*.

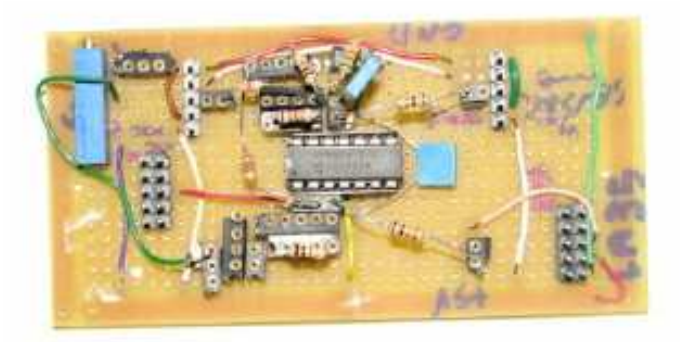


Ilustração 3-27 Foto placa amplificadora do sensor de temperatura e do sensor de CO_2

3.11. Sensor de CO_2

O sensor MG811 foi utilizado para detenção dos dados de CO_2. A Ilustração 3-28 mostra o sensor e a placa pré-amplificadora. Estes componentes foram adquiridos da empresa Paralax Inc. Foram utilizados dois conectores para o interfaceamento da fonte de alimentação, pino de controle do filamento e saída de sinal analógico.

Ilustração 3-28 Sensor e placa pré-amplificadora

O sensor de CO_2 modelo MG811 da empresa HANWEI, que possui uma elevada sensibilidade e seletividade ao CO_2 . Por seletividade entende-se que o sensor não varia sua diferença de potencial significativamente pela presença de outros gases tal como monóxido de carbono (CO), metano (CH_4) e etanol (C_2H_5OH). Sua faixa de operação compreende de 10 ppm a 10000 ppm de CO_2 . A Ilustração 3-29 mostra a resposta do sensor para diferentes gases.

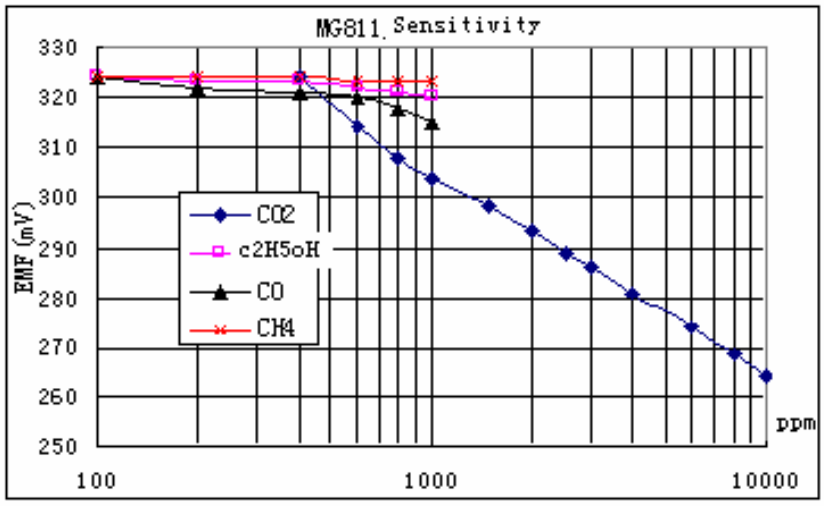


Ilustração 3-29 Resposta do sensor de CO_2 para diferentes gases testados (datasheet).

De acordo com o fabricante este sensor apresenta baixa dependência da temperatura e de diferença de umidade. Na Ilustração 3-30 pode-se observar esta característica.

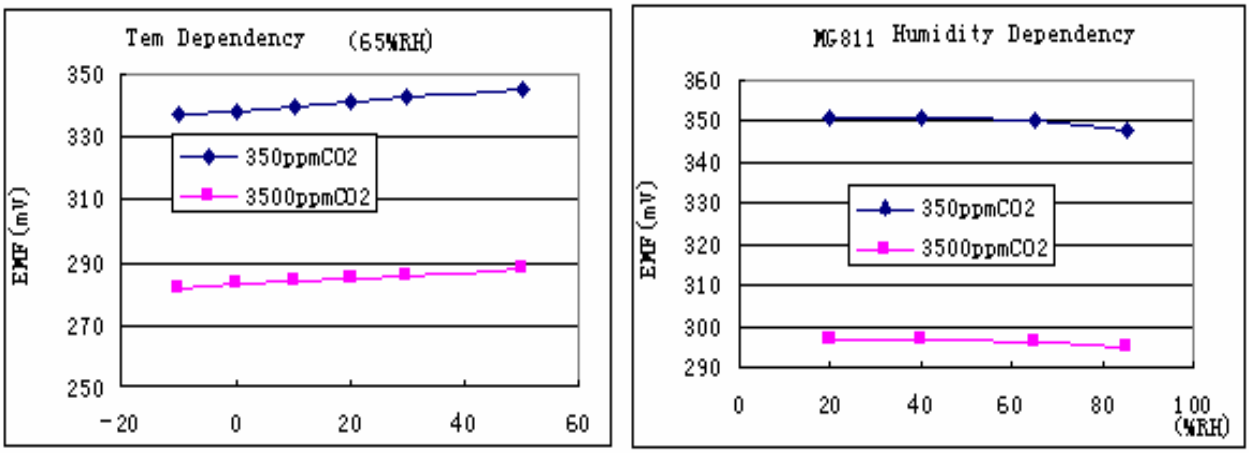


Ilustração 3-30 Resposta do sensor CO_2 para a variação de umidade (à esquerda) e temperatura (à direita) ;(datasheet).

O sensor MG811 adota o princípio da célula galvânica de eletrólito sólido (pilha), no qual é gerada uma diferença de potencial proporcional a concentração de CO_2. O sensor é composto pelas seguintes células sólidas:

-Ar

-Au/Carbonato/ Au

-CO_2

Quando o sensor é exposto ao CO_2 ocorrem às seguintes reações nos eletrodos ocorrem:

Reação Catódica : $2Li^+ + CO_2 + \frac{1}{2}O_2 + 2e^- = Li_2CO_3$

Reação Anódica : $2Na^+ + \frac{1}{2}O_2 + 2e^- = Na_2O$

Reação química total : $Li_2CO_3 + 2Na^+ = Na_2O + 2Li^+ + CO_2$

Como o sensor possui um filamento de aquecimento para elevar sua temperatura, depois de entrar em equilíbrio térmico, o sensor se aproxima a uma célula ideal, onde seus terminais produzem uma diferença de potencial (mV) proporcional à concentração de CO_2, correspondendo à equação de Nernst (RUSSEL, 1994).

$$emf = e_c - (\frac{RT}{2F}) \ln P_{co_2}$$ Equação (1);

Na qual:

P_{co_2} é a pressão parcial de CO_2

E_c é a diferença de potencial inicial a volume constante (variaçao da energia livre padrão).

R é a constante dos gases (8,315 $JK^{-1}mol^{-1}$)

T é a temperatura absoluta (298,2 K à 25^0C)

F é a constante de Faraday (96485 C mol^{-1})

A faixa de variação da diferença de potencial elétrica (emf) do sensor MG811 é de 30 mV a 50 mV, numa faixa de concentração de CO_2 que compreeende de 350 ppm à 10000 ppm de gás carbônico.

Isolando-se a equação (1) para acharmos o valor de dióxido de carbono, tem-se:

$$P_{co_2} = e^{-\frac{2F}{RT}(emf - E_c)}$$ Equação (2);

A equação (2) mostra o valor da concentração de dióxido de carbono, onde o valor da diferença de potencial convertida pelo sistema de aquisição do projeto é representado pelo valor de emf.

A Ilustração 3-31 mostra o diagrama do circuito do sensor e da placa de pré-amplificação. O sensor possui um filamento que á acionado pelo pino de controle do PIC16F916, onde o mesmo é acionado na porta de um transistor MOSFET. Assim este filamento proporciona a temperatura suficiente para que a reação ocorra e seja gerada uma diferença de potencial proporcional a concentração de CO_2. Este circuito possui uma saída regulada de 3,3V para a alimentação dos amplificadores operacionais da placa.

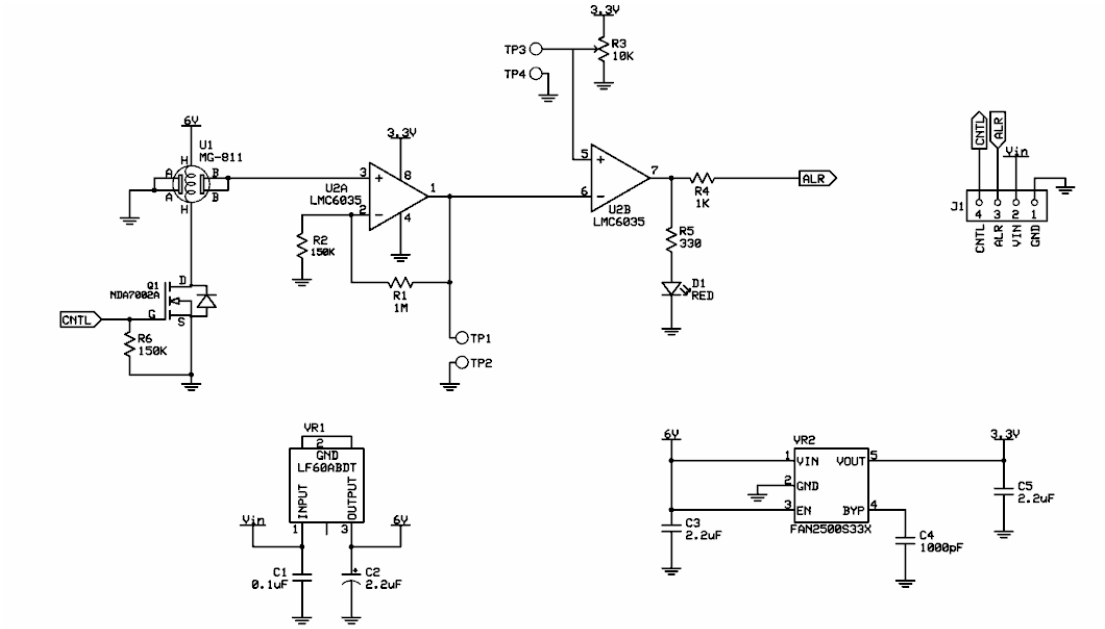


Ilustração 3-31 Circuito da placa pré-amplificadora e reguladores de tensão (datasheet,2012]

3.12. *Circuito com Filtro Passa Baixa Sensor CO_2*

A Ilustração 3-32 mostra o circuito completo de amplificação do sensor de CO_2. O amplificador operacional é responsável pela pré-amplificação, sendo necessária a colocação de um filtro passa baixa antes do segundo estágio de amplificação, para reduzir o ruído eletromagnético. O segundo estágio é representado pelo amplificador e pode ser visualizado na junto ao filtro passa baixa.

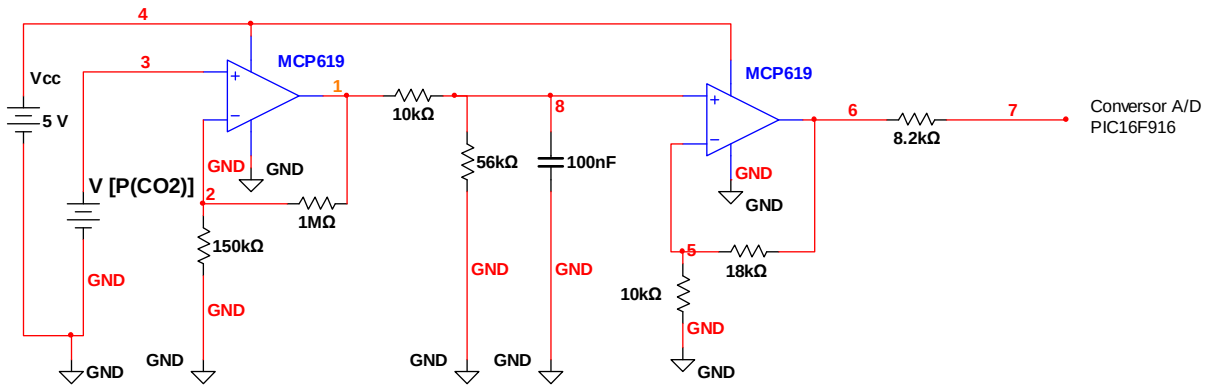


Ilustração 3-32 Circuito Acoplado à saída da placa pré-amplificadora.

Assim o ganho total no circuito é:

$Av_{total} = Av_1 * Av_{filtro} * Av_2$;

Na qual:

Av_1 =(1+1M/150K) = 7,666;

Av_{filtro} =(R2/R1.R2.C1 s + R1 + R2)=(56K/66K)= 0,8484;

Av_2 =(1+18K/10K)=2,8;

Av_{total}= 18,21;

A Ilustração 3-33 mostra o diagrama dos passos envolvidos no condicionamento de sinal.

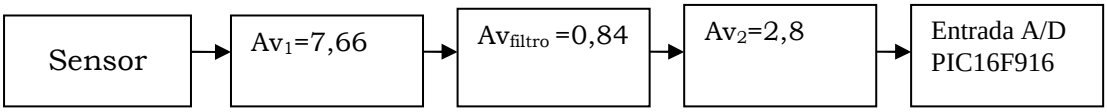


Ilustração 3-33 Diagrama em blocos do condicionador de sinal

3.13. Integração do Sistema

A Ilustração 3-34 mostra a montagem de todas as placas envolvidas no conjunto do analisador. É possível observar no centro onde se encontra o sensor de CO_2 e o cabo conversor de comunicação serial para USB, utilizado para aquisição de dados.

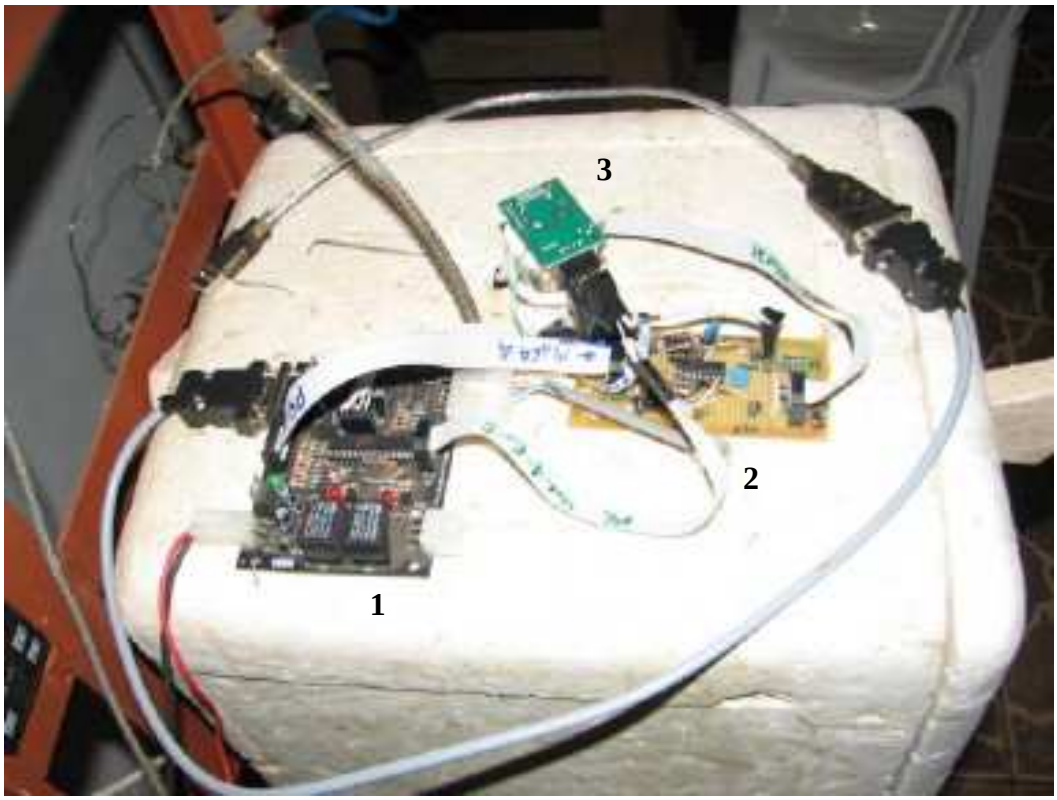


Ilustração 3-34 Circuito completo montado no sistema.1-Placa CPU; 2-Placa amplificadora sensores; 3-Placa pré amplificadora sensor de CO_2

3.14. *Firmware Microcontrolador*

O software desenvolvido para o microcontrolador PIC16F916 foi feito em linguagem C. Utilizou-se como ferramenta de desenvolvimento o software MPLAB versão 8.56. O compilador "C" utilizado foi da empresa HITECH Inc.

O *software* foi gravado pela serial do microcontrolador (ICSP) através do programador ICD3, equipamento adquirido da própria empresa Microchip.

Pela Ilustração 3-35 mostra o diagrama em blocos do *software* desenvolvido para microcontrolador. Inicialmente foram estipuladas todas as definições do compilador e do microcontrolador, utilizando seus registradores. Foram configuradas suas portas de entrada e saída, periférico de comunicação serial, taxa de transmissão, interrupções e conversor A/D, *timers* e *Wachtdog*. A função do Watchdog é a de reiniciar o microcontrolador no caso de o *software* trancasse devido a um erro de código.

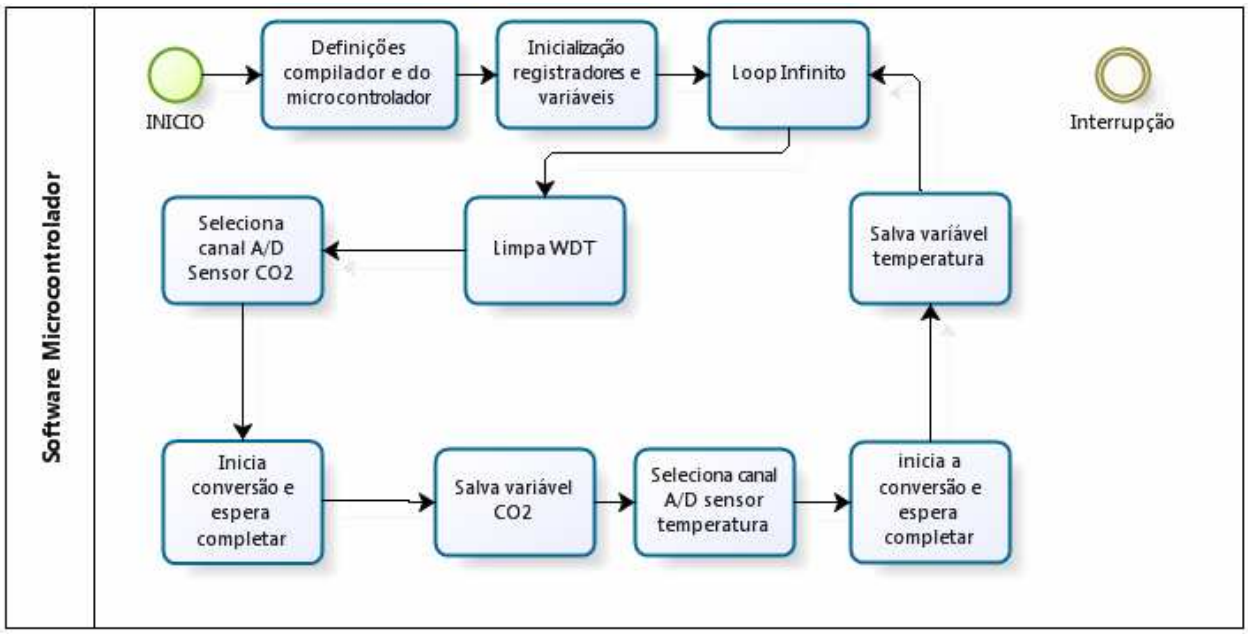


Ilustração 3-35 Diagrama em blocos do programa principal

Após isso, foram inicializadas as variáveis que conterão o valor convertido pelo A/D do sensor de CO_2 e do sensor de temperatura. Tem-se dois registradores que são lidos constantemente o ADRESH e ADRESL, onde são armazenados nas variáveis de 8 bits, que posteriormente são enviadas pela serial do microcontrolador.

Assim, com os valores menos significativos e mais significativos monta-se uma variável de 10 bits, que é a máxima resolução do conversor A/D. Temos uma faixa de valores que compreende de 0 a 1023, que representa a faixa de tensão entre 0 a 5 V. No computador, após transmissão dos dados, as variáveis dos sensores são convertidas para valores de 10 bits.

O programa funciona em um laço infinito, realiza as tarefas de limpar o temporizador do *Watchdog*, escolhe o canal analógico do sensor de CO_2, iniciar a conversão e espera o processo completar, salvando na variável do valor de CO_2. Depois é realizada a troca do canal analógico para o sensor de temperatura, inicia a conversão e espera o processo completar, salvando na variável do valor de temperatura. Este processo é repetido, mantendo sempre o valor da conversão atualizado para os dois sensores, parando apenas para atendimento da interrupção de recepção da serial por um curtíssimo espaço de tempo.

A Ilustração 3-36 mostra o fluxograma da rotina de interrupção de recepção de dados, que é chamada quando ocorre uma entrada de dados. O registrador de recepção contém o dado recebido do computador, é lido e de acordo com seu valor será executado o comando correspondente. A seguir tem-se a descrição dos nove comandos implementados:

Comando 0 = função para acionar relé 1 (Colocar caldo nutriente dentro do biofiltro).

Comando 1 = função para desligar relé 1 (Remover caldo nutriente dentro do biofiltro).

Comando 2 = função para acionar relé 2.

Comando 3 = função para desligar relé 2.

Comando 4 = função para enviar valor A/D do RA0 (leitura do potenciômetro de testes e do sensor CO_2)

Comando 5 = função para acionar o pino controle RB2, que liga filamento do sensor de CO_2.

Comando 6 = função para desligar pino controle RB2, que desliga filamento do sensor de CO_2.

Comando 7 = função para ler o estado do pino RB3 que é o alarme de limite de CO_2 (implementação futura).

Comando 8 = função para enviar valor A/D do RA1 (leitura do potenciômetro de testes e do sensor temperatura LM35).

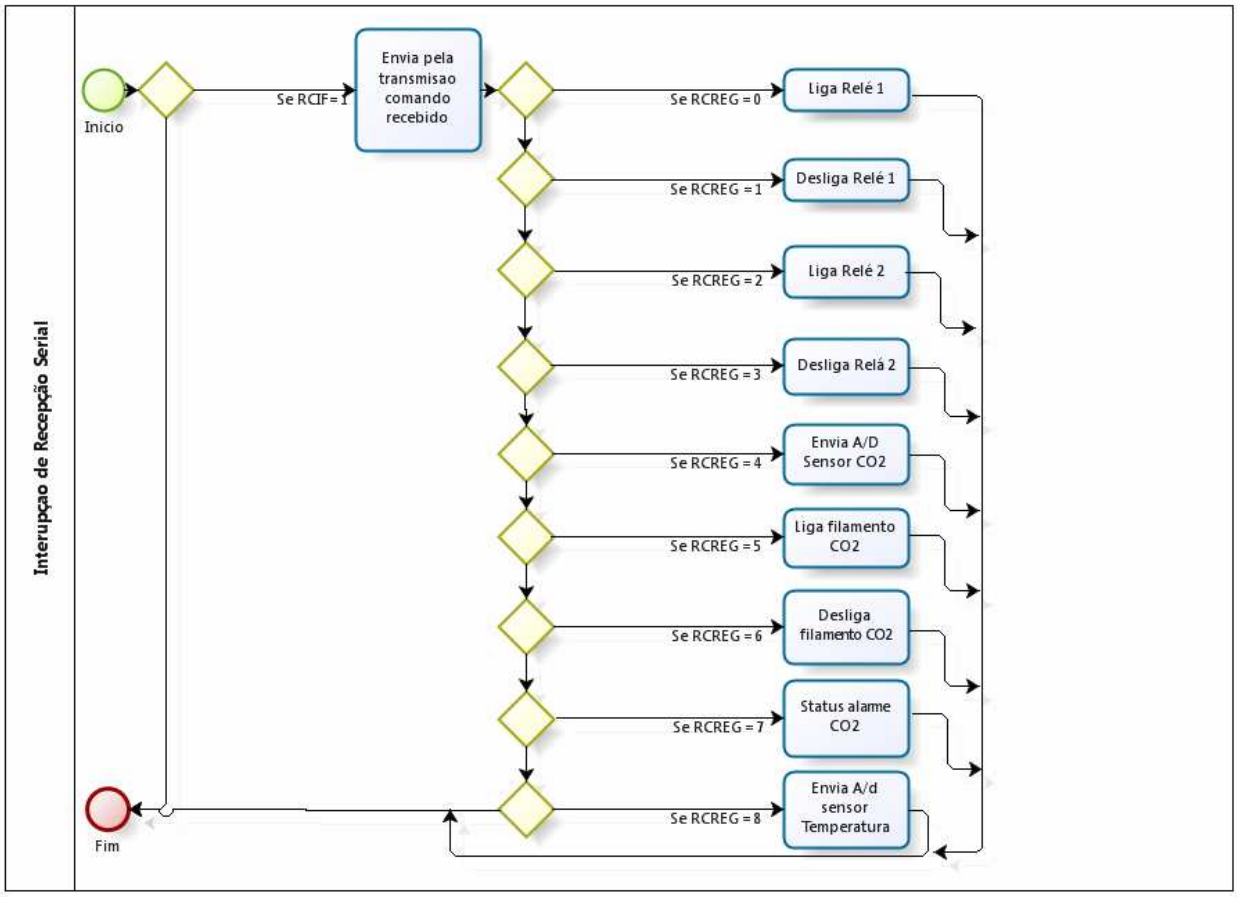


Ilustração 3-36 Fluxograma da interrupção de recepção serial

Para os testes foram utilizados dois potenciômetros com a alimentação em 5 V para testar os dois canais analógicos e verificar o funcionamento do conversor A/D e a troca de canais analógicos.

Os testes na serial foram realizados pelo programa Hiperterminal do Windows, assim como a utilização de um conector em *loop* , ligandos os pinos RX e TX do computador para testar o envio e recebimento de dados. Pelos comandos mandados serialmente tinham-se acesso através do computador ao monitoramento dos sensores, controle dos relés e pinos de controle do filamento.

3.15. Descrição dos Sistemas Informáticos e Computacionais

A interface de coleta de dados foi desenvolvida na linguagem C# e para o armazenamento dos dados foi utilizado o banco de dados. O objetivo foi elaborar uma interface gráfica onde pudessem ser medidos os valores dos sensores, realizadas as calibrações, configuração do sistema de comunicação e acionamento direto pelo computador do Hardware desenvolvido.

A ferramenta de desenvolvimento para a interface gráfica foi o programa Visual Studio 2010 da Microsoft e para gerenciamento do banco de dados e visualização das tabelas desenvolvidas foi utilizada a ferramenta da empresa Oracle Inc. cujo programa utilizado foi o Mysql Workbench 5.2.

O software do analisador de CO_2 foi dividido em 5 partes distintas, sendo que cada parte do software possui uma tela dedicada. A seguir descrevem-se cada uma delas:

Tela Gráfica: Busca dados de temperatura e CO_2, filtra os dados por data inicial e final, chama a última calibração da tabela, constrói o gráfico, e imprime para formato PDF.

Tela Configuração do Banco: Seleciona parâmetros de configuração de banco de dados e testa as conecxões.

Tela Comunicação: seleciona porta de comunicação e implementa comandos manuais para enviar a placa.

Tela Aquisição: adquire os dados dos sensores de temperatura e de CO_2 e os coloca em tabelas, chama última calibração, limpa os dados da tela, aciona e desaciona o filamento do sensor de CO2, coloca os valores médios, máximos e mínimos dos sensores, inicia, finaliza a aquisição e guarda dado no banco de dados.

Tela Calibração: adquire os dados para calibraçãos dos sensores, calcula os coeficientes, armazena os valores em tabelas, coloca os valores dos padrões e os guarda no banco de dados.

A seguir descrevem-se mais detalhamente as funções de cada tela, sendo que essas formam todo o aplicativo desenvolvido.

3.16. Tela Gráfica

A Ilustração 3-37 mostra a tela gráfica para analizar os dados obtidos no experimento dentro de uma faixa específica de datas. Se nenhuma data for selecionada o aplicativo usa a data do dia de acesso. A Ilustração 3-37 mostra a aquisição do aquecimento da estufa do biofiltro até os 55^0 Celsius durante um período de tempo, com leituras a cada segundo.

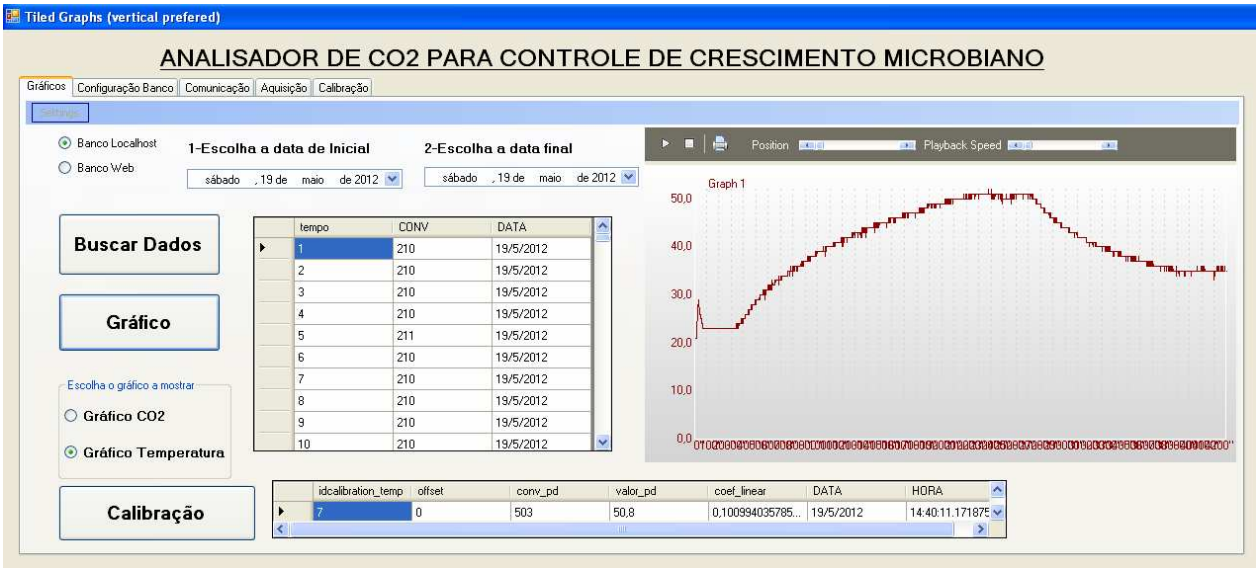


Ilustração 3-37 Tela Grafica

A pesquisa nos dados pode ser feita para o sensor de temperatura ou para o sensor de CO_2, e a mesma pode ser observada pressionando-se o botão "gráfico". Se nenhuma calibração for chamada, é possível visualizar os dados da conversão analógica independente para cada sensor em questão. Se for pressionado o botão de "calibração", o aplicativo verifica qual sensor está selecionado e busca no banco de dados Mysql os dados da última calibração salva. Usando o ícone de impressão é possível imprimir para um arquivo de formato PDF e salvo para futuras observações.

A opção de salvar os dados na internet é um melhoramento futuro, a ser implementado, sendo uma boa opção para compartilhar dados com outros pesquisadores.

A Ilustração 3-38 mostra o fluxograma em blocos do botão "busca dados". Quando o mesmo for pressionado, verifica-se de qual sensor devem-se buscar os dados, verifica-se quais as datas para realizar a busca pela tela do aplicativo. Monta-se o comando Mysql, abre a conexão com o banco,

envia os parâmetros da tela conexão e são colocados na tabela principal os dados solicitados e filtrados por período.

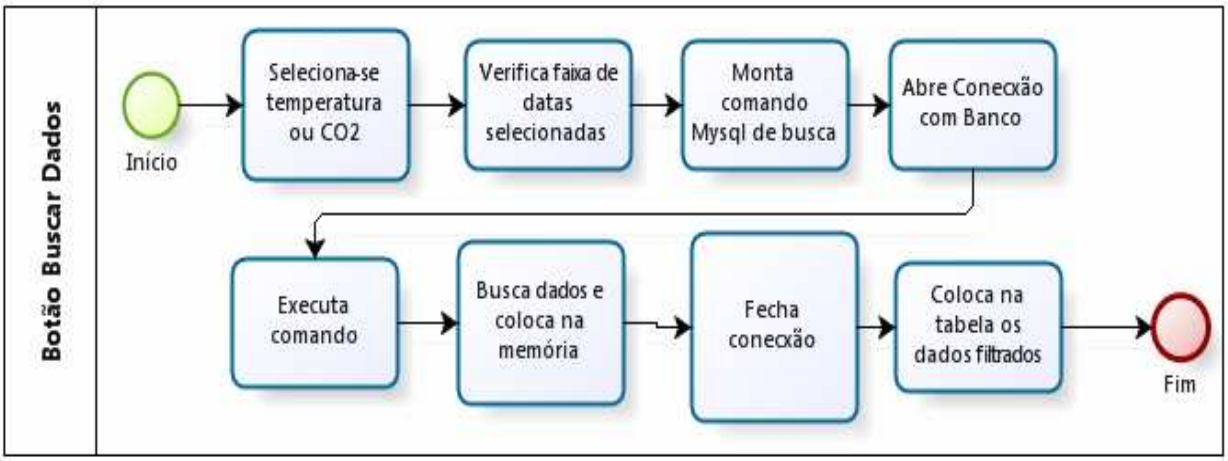


Ilustração 3-38 Diagrama Botão Busca Dados

Na Ilustração 3-39 indica a montagem do gráfico utilizando um objeto gráfico do aplicativo. Assim é verificada a necessidade do cálculo dos dados, antes de desenhar na tela, pois se existir algum valor na tabela de calibração, são calculados os dados na equação de calibração. No fim é realizado o uso de funções deste objeto para realizar a renderização dos dados que estavam na tabela principal.

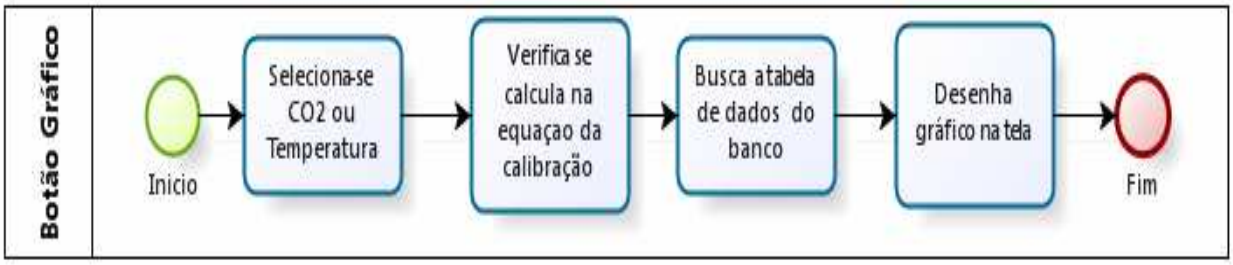


Ilustração 3-39 Diagrama Botão Gráfico

Se não for nunca presssionado o botão “Calibração” os dados serão desenhados no gráfico na resolução do conversor analógico para digital (Faixa de 0 a 1023). A Ilustração 3-40 permite visualizar os passos para chamar os dados da última calibração conforme sensor selecionado na tela principal. A busca no banco passa pelos mesmos processos esclarecidos anteriormente, com exceção que este faz a procura dos dados da data mais atual em que o sensor escolhido foi calibrado.

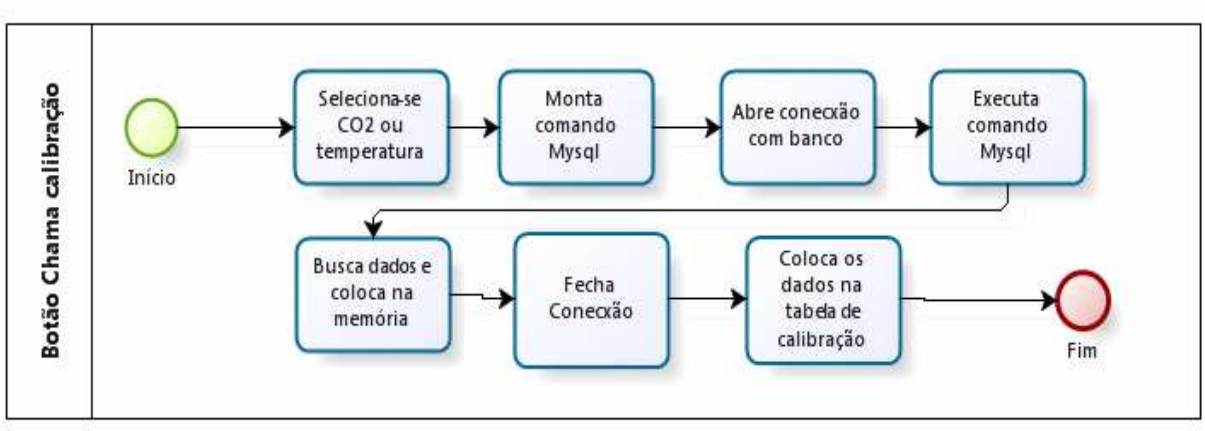


Ilustração 3-40 Diagrama Botão Busca Calibração

3.17. *Tela Configuração de Banco de Dados*

Esta tela tem o objetivo de obter os parâmetros necessários para abrir a conecxão, sendo que estes são passados, através de parâmetros, as permissões necessárias para se conectar com o banco de dados. Assim na Ilustração 3-41 podem-se visualizar os campos utilizados e os parâmetros passados ao banco.

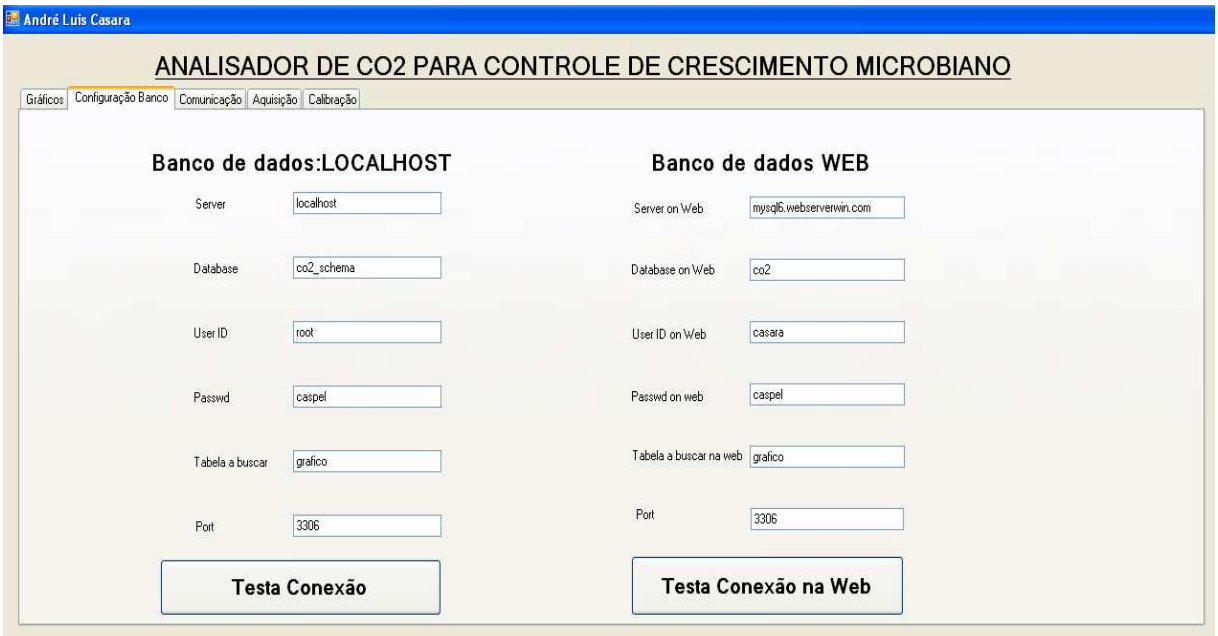


Ilustração 3-41 Tela configuração de parâmetros do banco de dados

Pela Ilustração 3-42 tem-se a idéia do processo para abrir a conexão no banco de dados, testar se os parâmetros passados são coerentes e mostrar uma mensagem de erro caso estejam equivocados.

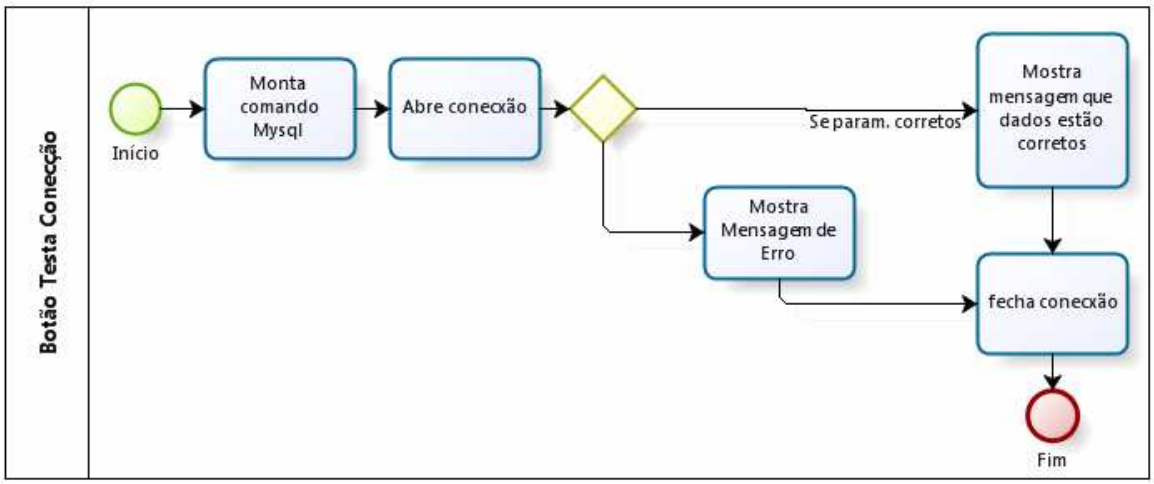


Ilustração 3-42 Diagrama Botão Testa Conexão

Foi colocado nesta tela um botão chamado “Testa Conexão”, para verificar se os parâmetros de conexão estão corretos, facilitando o diagnóstico com o banco, caso não seja possível gravar.

3.18. *Tela Comunicação e Controle Manual do Hardware*

A tela mostrada na Ilustração 3-43 tem como objetivo principal abrir e fechar a porta de comunicação serial do computador e testar o hardware manualmente. No lado esquerdo da tela temos um botão onde os comandos são enviados em código ASCII, e foram extensivamentes utilizados para testes e desenvolvimento do programa na linguagem C# e na Linguagem C, interfaceando o computador e a placa desenvolvida. Foi utilizado também cabo construído com os pinos 2 e 3 (TX e RX) do conector DB-9 ligados juntos, para testar o software na linguagem C#. O lado direito desta tela foi desenvolvido posteriormente para facilitar o uso do equipamento, tornando mais prática sua utilização.

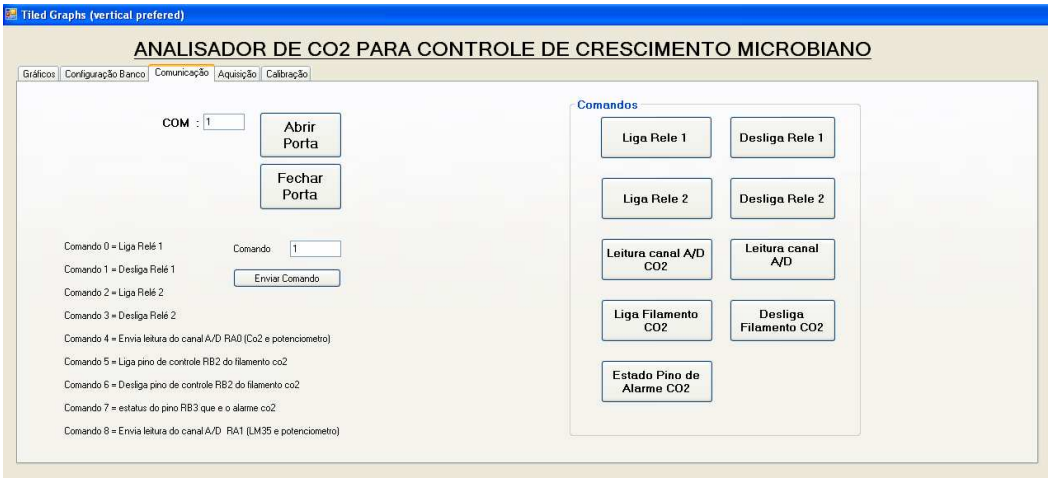


Ilustração 3-43 Tela Comunicação e acionamento manual do Hardware

O diagrama da Ilustração 3-44 mostra os passos para abrir e fechar a porta serial do computador. Como não existia saída padrão RS-232 no computador utilizado, pode ser usado um conversor USB/RS-232. O programa busca numa caixa de texto o numero da porta a ser aberta e depois executa o comando de abrir a porta. Para realizar o fechamento da porta serial é verificado se a mesma esta aberta, para depois executar o comando de fechar porta.

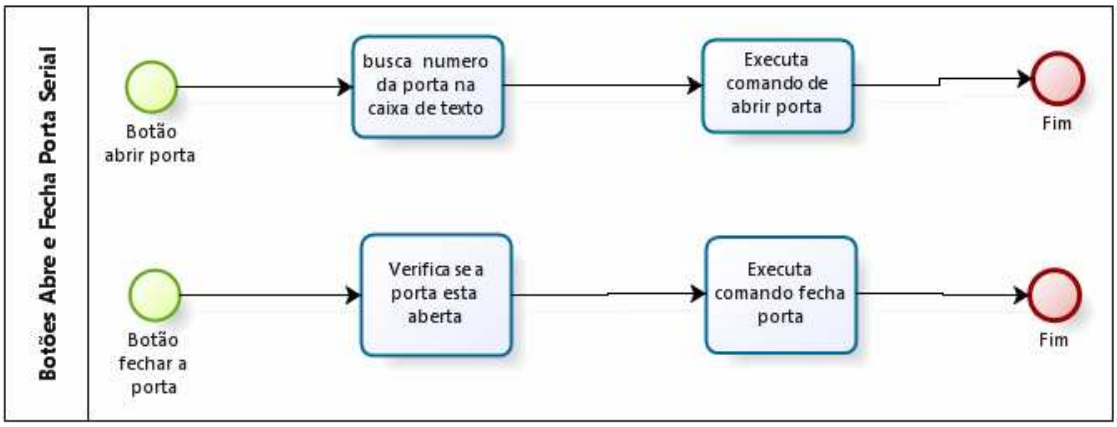


Ilustração 3-44 Diagrama dos botões abre e fecha porta

A Ilustração 3-45 foi dividida em blocos, mostrando a interrupção que ocorre quando algum dado for recebido. Todo o comando enviado ao microcontrolador, após ser processado e validado, o mesmo responde retransmitindo o comando recebido.

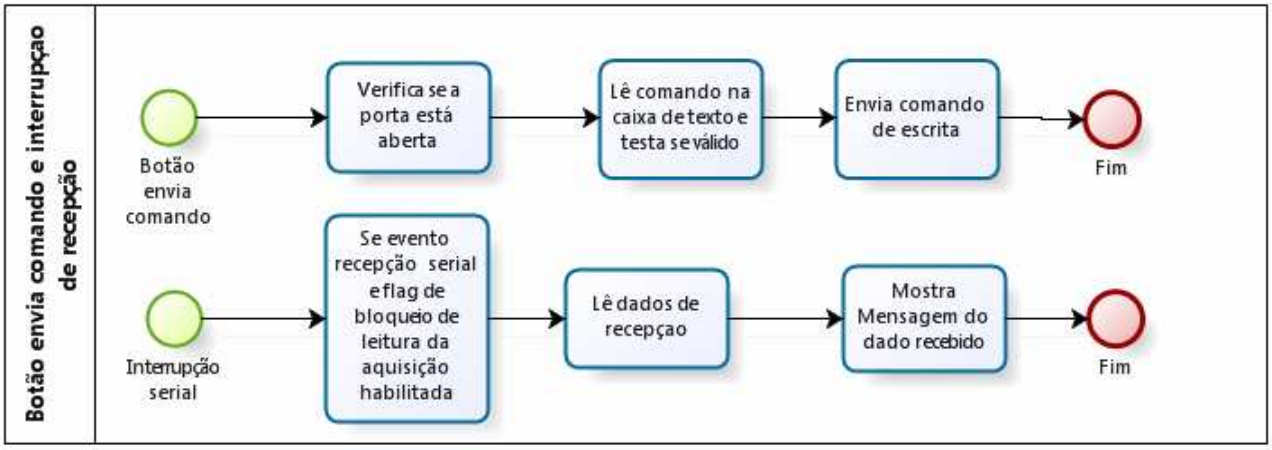


Ilustração 3-45 Diagrama Botão envia comando e interrupção de recepção

Quando se pressiona o botão “enviar comando”, a rotina verifica se a porta está aberta, efetua-se a leitura na caixa de texto, testa se o comando é

válido e envia-se o comando pela porta serial. Cabe salientar que esta tela usa a interrupção de recepção que é diferente da empregada na tela de aquisição e calibração. A diferença principal é a criação de uma interrupção de software (*Thread*) para as telas de calibração e aquisição. Assim foram criados variáveis indicadoras (*Flags*) para que ocorresse a leitura dos dados recebidos somente pela tela do software na qual foi enviado o comando.

Quando ocorrer a interrupção serial, uma rotina verfica de que tela o comando foi enviado, lê os dados e os mostra em uma caixa de mensagem, comprovando que o microcontrolador entendeu o comando.

Os botões do lado direito da tela (Ilustração 3-43) tem o mesmo funcionamento que o botão envia comando, salvo que cada botão já tem fixo o código do comando como parâmetro enviado.

3.19. Tela Aquisição de Dados

A Ilustração 3-46 mostra detalhes da etapa de aquisição de dados. A taxa de aquisição escolhida para registrar a leitura dos sensores de CO_2 e temperatura foi de 1 segundo. Há duas tabelas idênticas para cada sensor, onde são armazenados os dados da leitura menos significativa do conversor A/D, mais significativa do conversor A/D, valor de 10 bits do conversor A/D, data e hora da aquisição. As tabelas de calibração podem ser chamadas separadamente através do botão “calibração” que retorna os últimos dados calibrados. Se existe algum valor nas tabelas de calibração, o aplicativo coloca em duas caixas de texto os valores calibrados.

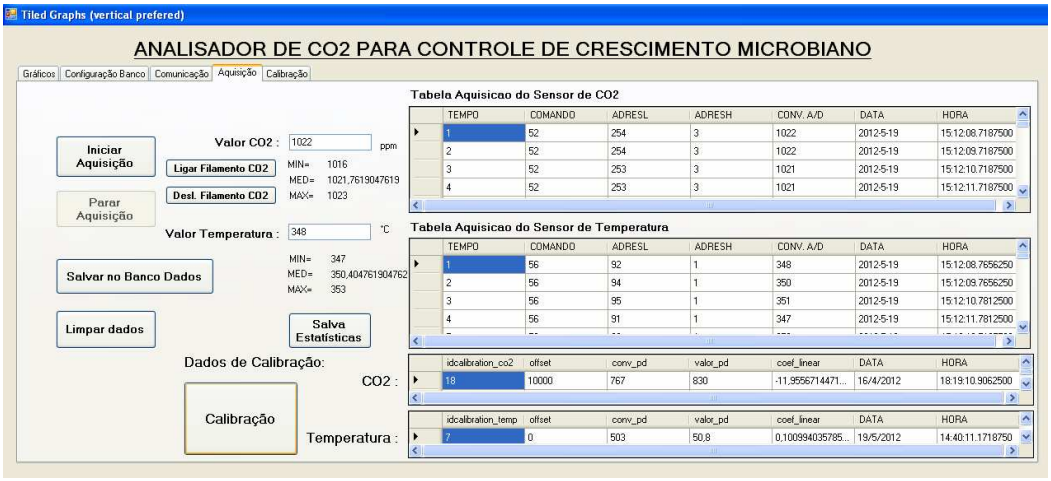


Ilustração 3-46 Tela aquisição de dados de temperatura e CO_2

As funções dos botões de “ligar o filamento” e “desligar o filamento” tem o mesmo funcionamento que os botões da tela de comunicão explicado anteriormente, onde envia-se os comandos conforme Ilustração 3-43. O Botão “limpa dados” limpa as tabelas dos dados aquiridos e os valores dos dados estatísticos utilizados, que eram mostrados nesta tela independente para cada sensor.

O botão “calibração”, mostrado no aplicativo tem a função de buscar os dados da tabela das calibrações no banco de dados conforme indicado o funcionamento na Ilustração 3-37.

A Ilustração 3-47, mostra o diagrama em blocos da rotina utilizada para enviar dois comandos de leituras dos dois sensores e o recebimento da resposta pelo canal serial, sendo os valores colocados em uma tabela independente para cada sensor. Nesta tela as leituras na recepção serial foram realizadas por interrupção de software.

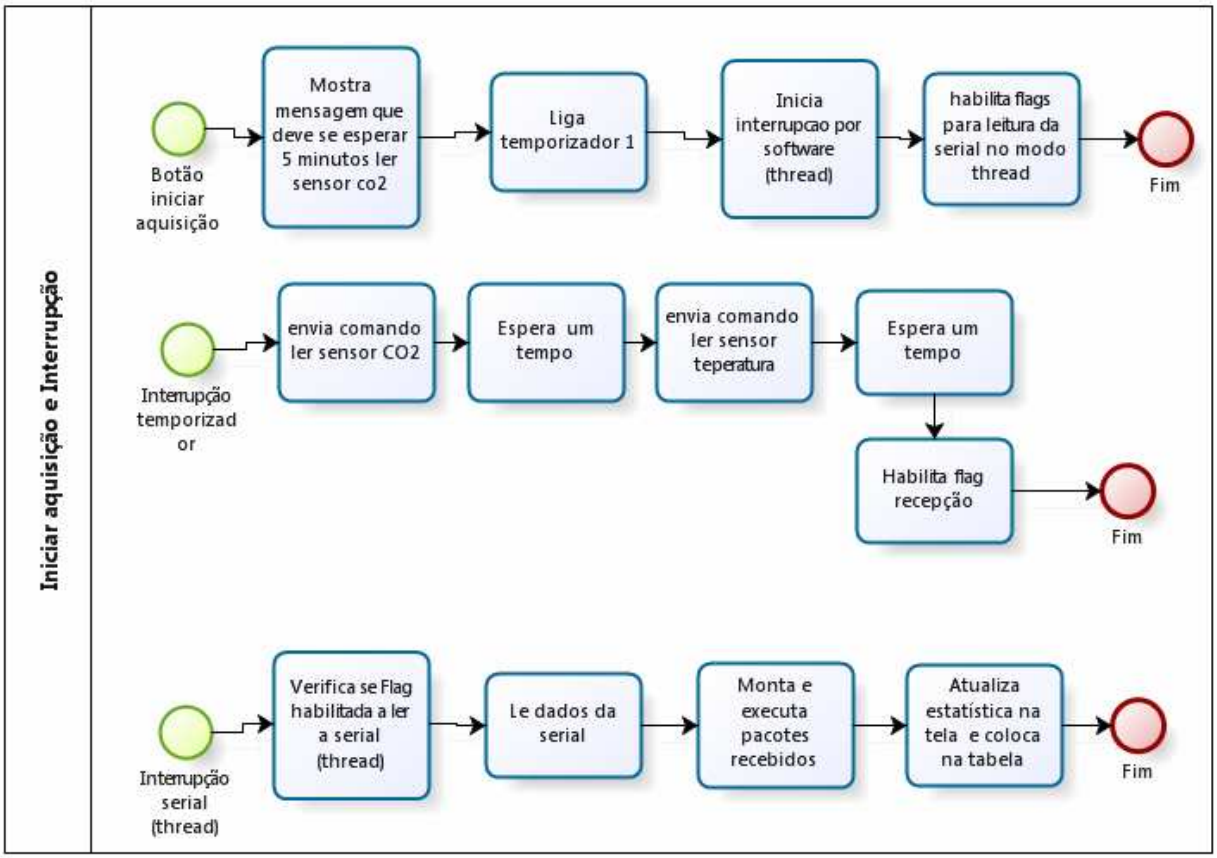


Ilustração 3-47 Botão iniciar aquisição e interrupção do temporizador e “Thread” serial

Todo o algoritmo da tela de aquisição inicia com o acionamento do botão “inicia aquisição”. O sistema mostra uma mensagem para aguardar os 5 minutos necessários para o filamento do sensor de CO_2 aqueca, então se habilita a interrupção de um objeto temporizador, e criam-se a interrupção por *software* (*Thread*). Após iniciar a interrupção da *Thread*, é criada uma variável global de controle para gerenciar o envio e recebimento dos comandos e dos dados. Assim a interrupção do temporizador envia os dois comandos de leitura, e a Thread processa os comandos recebidos pela placa do microcontrolador.

Para interromper o processo de aquisição dos dados do analizador, é seguido a rotina da Ilustração 3-48, que consiste em desligar o temporizador, desligar a interrupção por software Thread, resetar os flags de controle de recebimento de dados e avisar, através de uma mensagem ao usuário do equipamento, sobre o desligamento do filamento do sensor de CO_2.

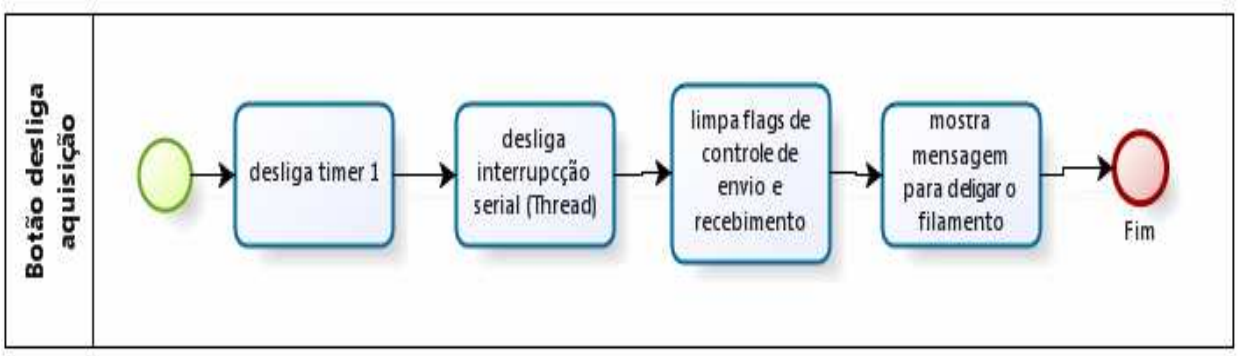


Ilustração 3-48 Botáo para desligar aquisição

Para futuras análises, foi desenvolvida uma tabela no banco de dados onde é possível salvar dados mínimos, máximos e médios das aquisições dos dois sensores. A Ilustração 3-49 observa-se o processo da rotina que copia os dados estatísticos da tela, acrescenta data e hora atual, monta o comando Mysql, abre a conexão executando o comando. Por fim encerra-se a conexão com os dados salvos na tabela do banco.

Há um botão, com a função de salvar todos os dados adquiridos nas duas tabelas, uma para o sensor de temperatura e outra para o sensor de CO_2. Nesta tabelas são também inseridas a data e a hora atual, para poder filtrar dos dados cronologicamente.

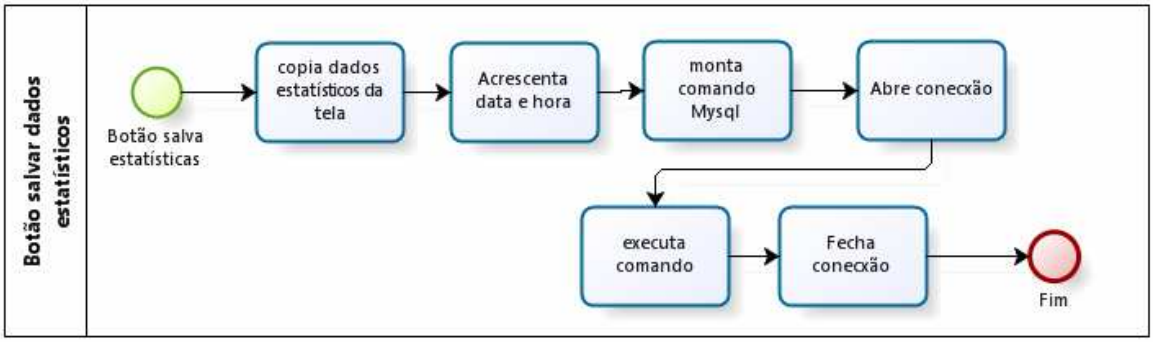


Ilustração 3-49 Botão para salvar estatísticas na tabela do banco

3.20. *Tela Calibração*

Devido à necessidade de aferir os valores fornecidos pelos sensores, a Ilustração 3-50 mostra a interface desenvolvida, para calibrar independentemente o sensor de temperatura e de CO_2. Há duas tabelas preenchidas conforme os valores dos padrões, dados adquiridos e cálculos realizados. Quando o botão "iniciar aquisição" é acionado, um comando é enviado para o hardware conforme sensor selecionado. Os botões "iniciar" e "parar aquisição" têm o mesmo funcionamento descrito anteriormente na tela de aquisição.

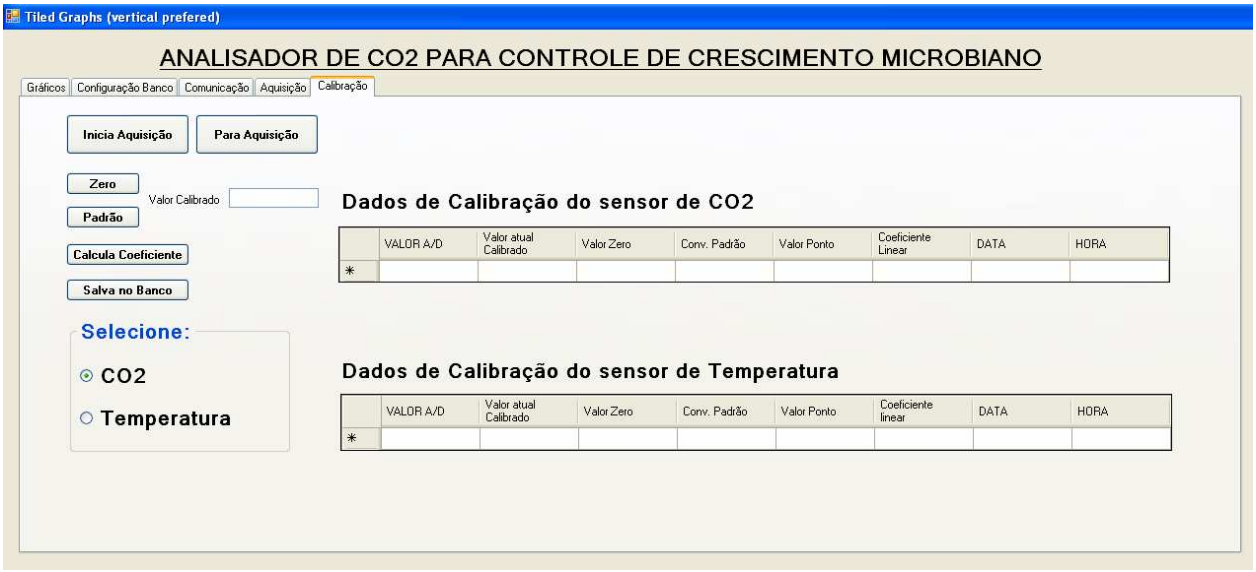


Ilustração 3-50 Tela de calibração dos dois sensores

A rotina verifica o recebimento de qualquer valor dos sensores e se existe um coeficiente da reta calculado entre os pontos de zero e padrão. Se existe um coeficiente calculado na célula da tabela, o aplicativo calcula o valor calibrado e o coloca nestas tabelas.

Quando já se tem colocado o valor do zero e do padrão (pelo pressionamento dos dois botões de “zero” e ”padrão”) é possível realizar o cálculo do coeficiente da reta. O diagrama na Ilustração 3-51 representa os passos do desenvolvimento do algoritmo do botão “calcula coeficiente”. Inicialmente é verificado qual o sensor selecionado, e em seguida os dados da tabela de calibração são buscados, o coeficiente linear é calculado e colocado na tabela.

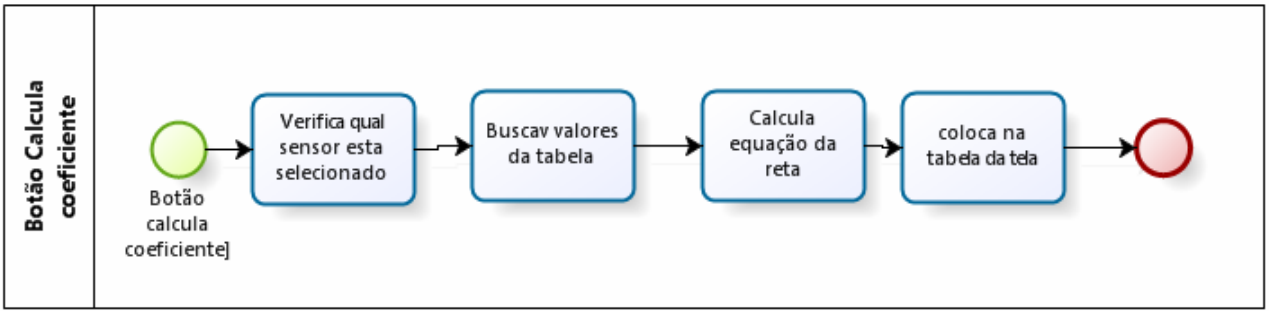


Ilustração 3-51 Diagrama do botão calcula equação

3.21. *Workbench MySQL*

Foi utilizado como ferramenta de testes e desenvolvimento o programa Workbench MYSQL da empresa Oracle inc. Neste aplicativo foram testados todos os comandos SQL (*Structered Query Language*) de inserir, apagar, criar tabelas e tipos de dados sendo que também é possível a visualização de todos os dados. A Ilustração 3-52 pode-se visualizar a tela principal do aplicativo.

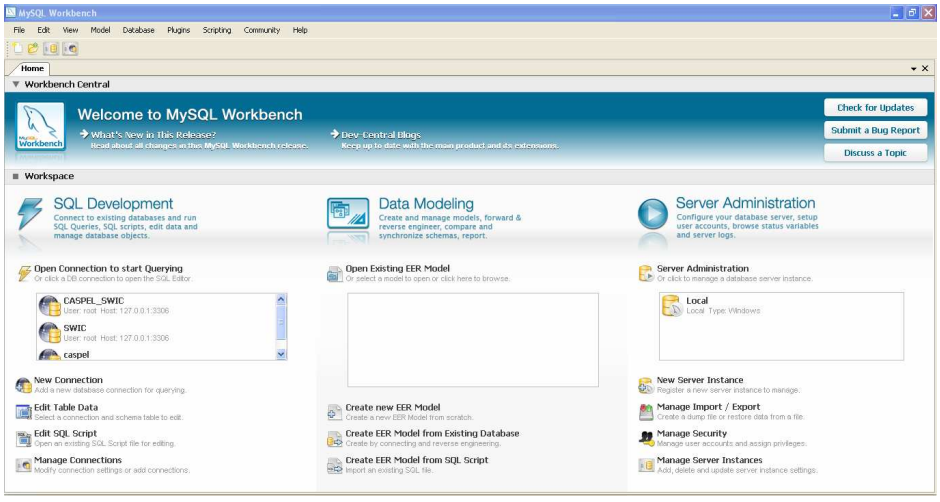


Ilustração 3-52 Tela do Programa de desenvolvimento e visualização do banco de dados

Neste aplicativo é possível criar e dar o nome da conexão, configurar senha e Login e porta de comunicação. Foram criados também as tabelas e tipo e dados de todas as colunas.

3.22. *Tabelas do Banco de Dados*

Foram criadas cinco tabelas nas quais os dados adquiridos foram gravados através de comandos SQL. Cada tabela possuiu suas colunas e tipo de dados, aos quais dependiam dos dados a serem salvos.

As cinco tabelas são acessadas pelo aplicativo desenvolvido e possuem os nomes abaixo:

Calibration_co2 = tabela onde os dados da calibração do sensor de CO_2 são salvos.

Calibration_temp = tabela onde os dados da calibração do sensor de temperatura são salvos.

Tabela_co2 = tabela onde os dados do sensor de CO_2 são salvos.

Tabela_temp = tabela onde os dados do sensor de temperatura são salvos.

Tabela_estatística = tabela onde os dados estatísticos dos dois sensores são salvos.

A Ilustração 3-53 mostra um exemplo da tabela e das colunas utilizadas na calibração do sensor de CO2. A tabela de calibração do sensor de temperatura é idêntica a esta.

Overview | Output | History | calibration_co2 (1)

Export

idcalibration_co2	offset	conv_pd	valor_pd	coef_linear	DATA	HORA
6	10000	556	530	-17.0323741007194	2012-04-10	12:12:35.0625000
7	5000	572	530	-7.81468531468532	2012-04-10	12:16:06.5625000
8	5000	585	540	-7.62393162393162	2012-04-10	12:19:56.5468750
9	10000	585	540	-16.1709401709402	2012-04-10	12:20:17.5468750
10	10000	598	540	-15.819397993311	2012-04-10	12:26:01.7812500
11	10000	740	540	-12.7837837837838	2012-04-13	15:19:40.1250000

Ilustração 3-53 Imagem da tabela de calibração sensor de CO_2

A Ilustração 3-54 mostra um exemplo dos tipos de dados das colunas utilizadas na tabela de calibração do sensor de CO2. Os tipos de dados da tabela de calibração do sensor de temperatura são idênticos a esta.

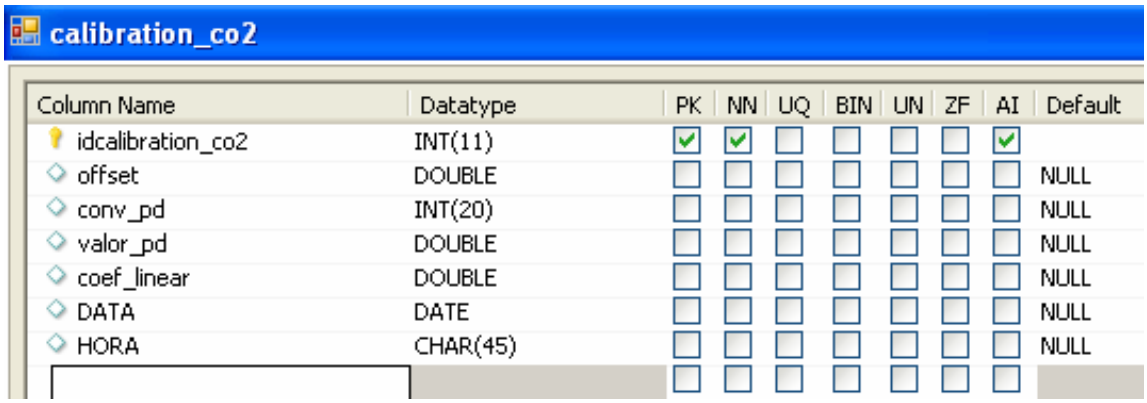
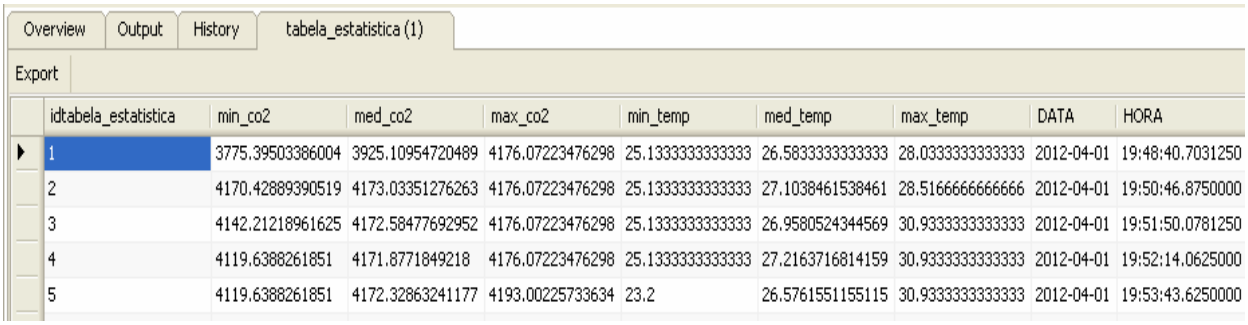


Ilustração 3-54 Tipo de dados da tabela de calibração sensor de CO_2

A Ilustração 3-55 mostra um exemplo da tabela utilizada nas estatísticas dos valores adquiridos e salvos manualmente.



idtabela_estatistica	min_co2	med_co2	max_co2	min_temp	med_temp	max_temp	DATA	HORA
1	3775.39503386004	3925.10954720489	4176.07223476298	25.1333333333333	26.5833333333333	28.0333333333333	2012-04-01	19:48:40.7031250
2	4170.42889390519	4173.03351276263	4176.07223476298	25.1333333333333	27.1038461538461	28.5166666666666	2012-04-01	19:50:46.8750000
3	4142.21218961625	4172.58477692952	4176.07223476298	25.1333333333333	26.9580524344569	30.9333333333333	2012-04-01	19:51:50.0781250
4	4119.6388261851	4171.8771849218	4176.07223476298	25.1333333333333	27.2163716814159	30.9333333333333	2012-04-01	19:52:14.0625000
5	4119.6388261851	4172.32863241177	4193.00225733634	23.2	26.5761551155115	30.9333333333333	2012-04-01	19:53:43.6250000

Ilustração 3-55 Imagem da tabela para salvar dados estatísticos dos dois sensores

A Ilustração 3-56 mostra os tipos de dados usados em cada coluna, onde se encontram informações de valores médios, valores mínimos e valores máximos dos dois sensores.

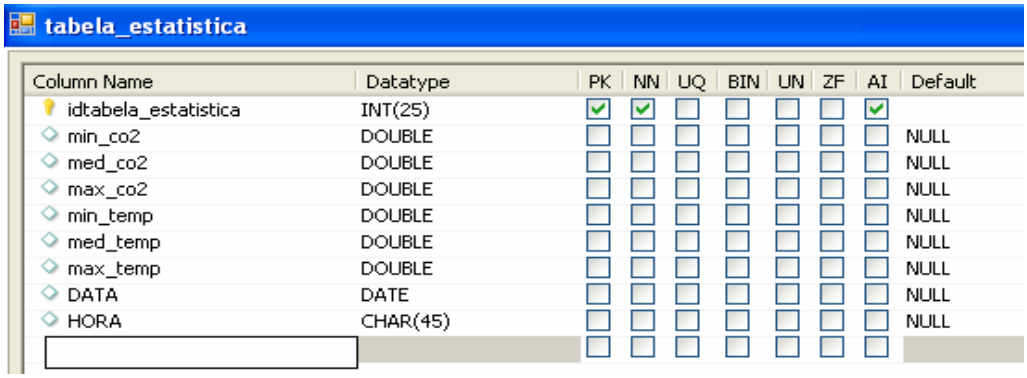


Ilustração 3-56 Tipo de dados da tabela estatística

A Ilustração 3-57 mostra um exemplo da tabela utilizada na aquisição do sensor de CO2.

Overview | Output | History | tabela_co2 (1)

Export

idtabela_co2	tempo	comando	ADRESL	ADRESH	CONV	DATA	HORA
1	1	52	135	3	903	2012-3-10	23:02:24.9843750
2	2	52	135	3	903	2012-3-10	23:02:25.9843750
3	3	52	135	3	903	2012-3-10	23:02:26.9843750
4	4	52	135	3	903	2012-3-10	23:02:28
5	5	52	135	3	903	2012-3-10	23:02:28.9843750
6	6	52	135	3	903	2012-3-10	23:02:29.9843750

Ilustração 3-57 Imagem da tabela dos dados adquiridos

A Ilustração 3-58 mostra um exemplo do tipo de dado utilizado na tabela de aquisição de dados do sensor de CO2.

tabela_co2

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
idtabela_co2	INT(11)	☑	☑	☐	☐	☐	☐	☑	
tempo	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
comando	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
ADRESL	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
ADRESH	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
CONV	INT(11)	☐	☐	☐	☐	☐	☐	☐	NULL
DATA	CHAR(45)	☐	☐	☐	☐	☐	☐	☐	NULL
HORA	CHAR(45)	☐	☐	☐	☐	☐	☐	☐	NULL

Ilustração 3-58 Imagem do tipo de dados da tabela dos dados adquiridos.

3.23. *Calibrações*

Foram realizadas as calibrações necessárias para os sensores de temperatura e para o sensor de CO_2. Foi realizada uma calibração com o uso de um ponto e zero para os dois sensores.

A Ilustração 3-59 mostra a calibração do ponto zero do sensor de CO_2, assim foi utilizado um cilindro padrão com o valor de 10000 ppm de CO_2, no qual através do software, o ponto foi salvo e colocado na tabela do banco de dados.

Ilustração 3-59 Calibração de Zero; 1-Cilindro Gás Padrão ; 2- Equipamento desenvolvido

Após a calibração de zero, foi realizada a calibração do ponto de 649 ppm de CO2. Após foi realizado o cálculo da equação da reta deste sensor. A Ilustração 3-60 pode-se visualizar o equipamento calibrado de medição e o analisador desenvolvido.

Ilustração 3-60 Calibração do ponto do sensor de $C0_2$; 1- Sensor CO_2 ; 2 – Placa Amplificação ; 3- Placa CPU ; 4 –Medidor calibrado de CO_2

A calibração do sensor de CO_2 foi realizada após 10 minutos de estabilização do filamento.

Para a calibração do sensor de temperatura foi utilizado um termômetro digital Fluke, com o qual se registrou o ponto da curva na faixa de 0 a 100 graus Celcius. A Ilustração 3-61 pode-se observar os testes realizados com padrão calibrado para verificar seu desempenho.

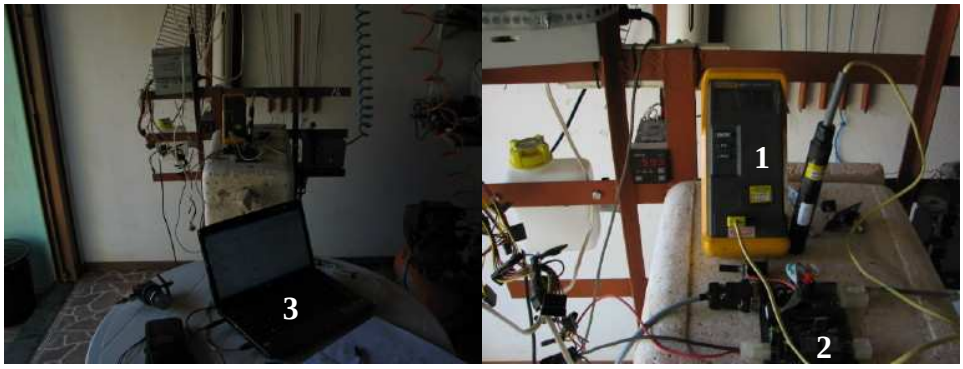


Ilustração 3-61 Conferência da calibração de temperatura no equipamento desenvolvido; 1- Medidor de temperatura Calibrado; 2- Placa Desenvolvida; 3- Local de Calibração

4. APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS

Foram realizadas várias etapas para o desenvolvimento do analisador de CO_2, desde a fabricação, a montagem das placas, dos circuitos de amplificação, calibrações, desenvolvimento do protótipo mecânico, circuito de amplificação do sensor de temperatura, calibrações do sensor de temperatura e testes biológicos práticos com uma população de microrganismos num período de 24h de aquisições.

Os dados do sensor de temperatura foram adquiridos para futuros trabalhos para corrigir o valor de CO_2 em diferentes temperaturas. Como todo o experimento foi fixado para 35^0 C, não foi realizada a correção de temperatura na leitura de CO_2.

Como a aquisição foi em intervalos de um segundo, obtiveram-se muitos dados que estão armazenados no banco de dados nas tabelas desenvolvidas para o experimento. Os dados são analisados utilizando o próprio aplicativo para fornecer as informações dos horários e datas da aquisição.

O experimento foi realizado em 24 horas, após a colocação do caldo nutritivo e de microrganismos de uma comunidade desconhecida inoculada, obtiveram-se interessantes respostas sobre o crescimento microbiano, correspondendo à literatura pesquisada (Janke, 2002). Com base nos dados adquiridos foi possível identificar as etapas de crescimento microbiano, correspondendo o analizador de CO_2 às expectativas de uma ferramenta de hardware e software para estudos de crescimento microbiano.

4.1. Aquisição Sensor de CO_2

Com base na Ilustração 4-1, podem-se observar os resultados do experimento prático, em que todos os dados adquiridos foram salvos e filtrados pelo analizador de CO_2. Foi obtido o número de horas de cada etapa do processo de crescimento microbiano, sendo a fase LAG com duração

aproximadamente de 12 horas, a fase LOG em torno de 5 horas, fase de TRANSIÇÃO em torno de 1 hora, fase de estabilização de 1 hora e no fim a fase letal que representa a morte dos microrganismos no final do experimento. È importante salientar que os resultados adquiridos são similares a Ilustração 2-1, correspondendo à literatura (Janke,2002) como uma comparação qualitativa.

Ilustração 4-1 Resultado prático do experimento de crescimento microbiano

Com uso do equipamento pode ser obtido um máximo de 4000 ppm de CO_2 no pico do gráfico onde os dados foram adquiridos. Houve uma discontinuidade no gráfico devido a uma queda de energia de 4h na aquisição do experimento, compreeendendo a parte que pode ser observada como uma linha reta antes do pico na Ilustração 4-1.

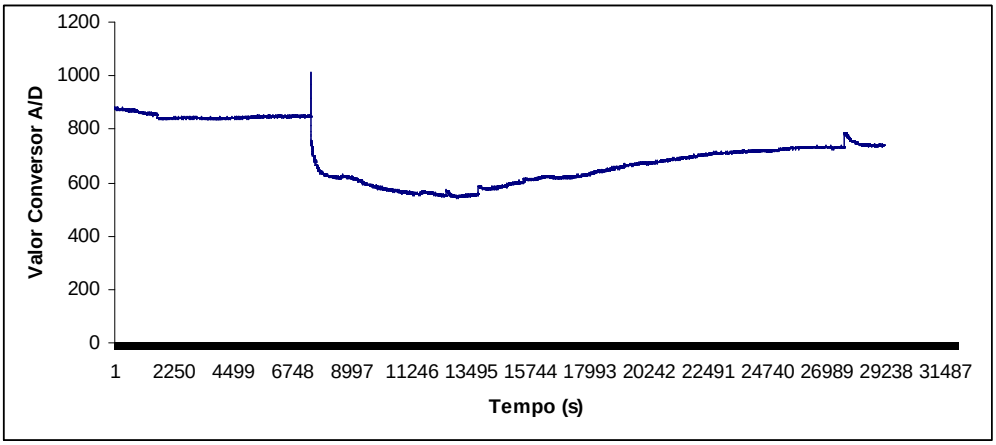


Ilustração 4-2 Dados adquiridos pelo conversor A/D do sensor de CO_2

A Ilustração 4-2 mostra os dados adquiridos pelo conversor A/D durante todo o experimento. Após aquisição foram tabulados os dados no programa EXEL da Microsoft.

Com os dados aquiridos foram realizados os seguintes cálculos e usando as seguintes constantes:

$R = 8{,}315\ J.\ K^{-1}.\ mol^{-1}$

$T = 308{,}2\ K\ (35^0\ C)$

$F = 96485\ C\ .\ mol^{-1}$

$P(CO_2) = e^{-(2F*(EMF - Ec))/(R*T)}$;

$P(CO_2) = e^{-75{,}29*(Vad*(Vcc/Resolucao\ Ad\ 10\ bits) - ((Vo\ data\ sheet\ sensor)*(Av\ amplificação)))}$;

$P(CO_2) = e^{-75{,}29((Vad*(5V/1023)) - (0{,}325mV*18{,}21))}$;

Na qual:

$P(CO_2)$ é a pressão parcial de dióxido de carbono.

Na Ilustração 4-3 mostra o resultado gráfico de todos dos dados adquiridos, sendo os dados em um valor numérico muito grande. Este gráfico é relevante, pois apresenta a transformação dos dados adquiridos através do condicionador de sinal implementado.

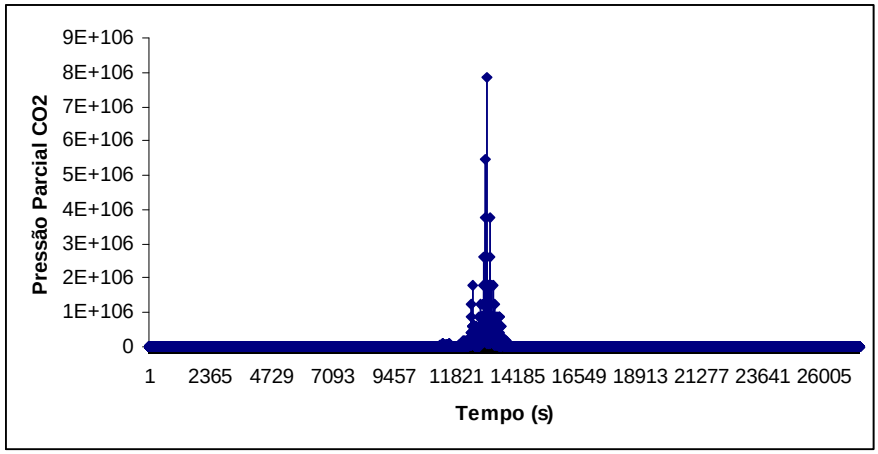


Ilustração 4-3 Grafico Pressão Parcial de CO_2 calculada com os dados

Devido ao resultado ser um valor muito elevado, foi realizado o cálculo do logaritmo deste valor e montado o gráfico na Ilustração 4-4. Assim foi obtido um resultado similar ao obtido pelo gráfico e calibração do software desenvolvido, só que em diferentes unidades de medida.

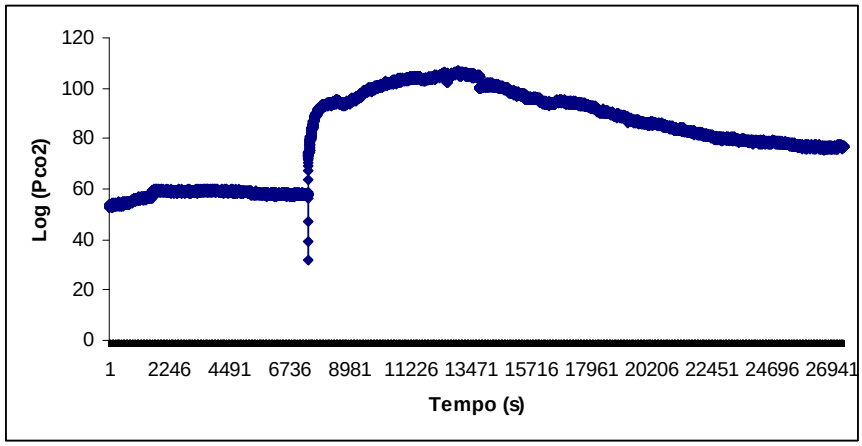


Ilustração 4-4 Gráfico Logaritmo da pressão parcial de CO_2

4.2. *Aquisição Sensor de Temperatura*

Pelas leituras de temperatura na Ilustração 4-5 pode-se visualizar que a mesma permaneceu estável durante todo o experimento.

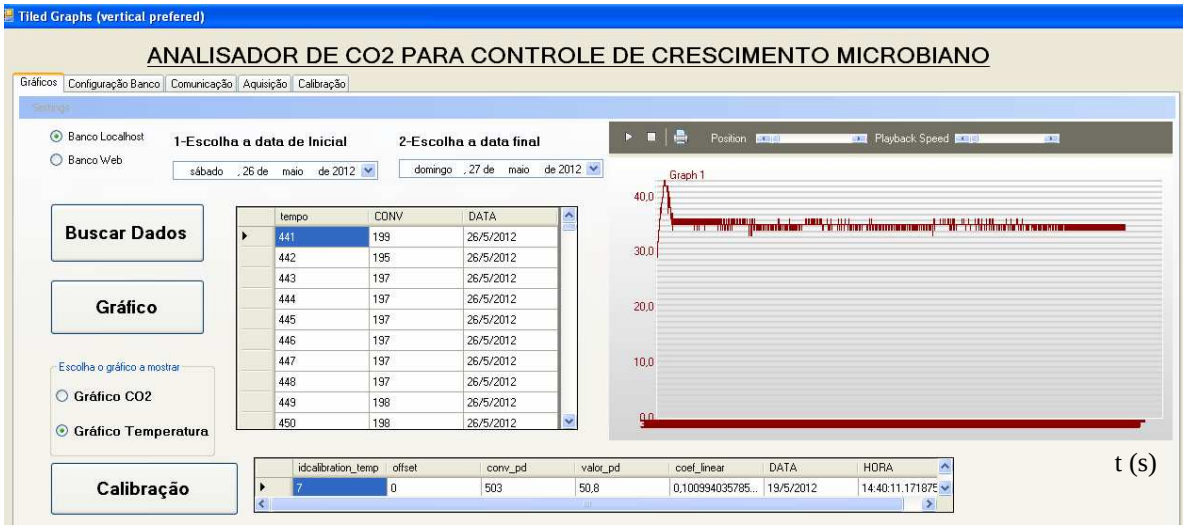


Ilustração 4-5 Grafico da temperatura monitorada durante todo o experimento.

Foram colocados todos os dados de temperatura no software EXEL da Microsoft e realizados cálculos de parâmetros estatísticos. Assim obtive-se os seguintes dados de todo o experimento:

Média = 34,95 ^{0}C;

Desvio padrão = 0,33 ^{0}C;

Erro padrão = 0,00149 ^{0}C;

$N_{amostral}$ = 49182;

Com base na Ilustração 4-6 podem-se observar todos os dados adquiridos pelo sensor de temperatura, sendo convertidos para temperatura pela equação de calibração usada no software. O coeficiente linear obtido foi de 0,1009, sendo convertido para temperatura no software quando necessário.

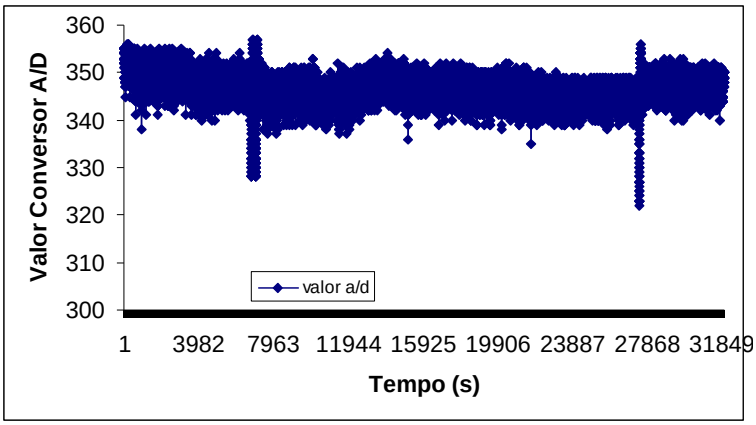


Ilustração 4-6 Gráfico dos dados do conversor A/D adquiridos pelo sensor de temperatura

4.3. Vazão de Ar comprimido

Durante o experimento foi medido com fluxímetro digital o fluxo de ar, o qual este fornecia oxigênio para os microrganismos aeróbios. Foi utilizada uma vazão de 185 ml/min, sendo realizadas verificações do fluxo a cada 6h.

Foi importante verificar o fluxo de ar na saída, pois se algum problema ocorresse com o ar comprimido, como por exemplo, um vazamento, os microrganismos morreriam.

Foram realizadas estas medições apenas como um indicador simples do fluxo de ar comprimido.

5. CONSIDERAÇÕES FINAIS

O projeto envolveu o desenvolvimento de hardware e software com o intuito de ser uma ferramenta de análise de projetos de biofiltragem e de auxiliar na remoção de poluentes atmosféricos. Para sua realização e desenvolvimento foram necessárias várias etapas, cada uma com a sua dificuldade e solução. Assim, descreveremos a seguir alguns pontos dos desafios enfrentados.

5.1. Problemas Mecânicos de Construção

Os testes experimentais do analisador de CO_2 necessitavam de um suporte físico para a fonte de alimentação, tomadas, controlador de temperatura, válvulas de fluxo, controladores de pressão, estufa feita de caixas de isopor, suporte para as placas de amplificação, sensores e placa de comunicação que controlava os relés.

Assim houve dificuldade na decisão do formato e disposição das partes, foi realizado solda em ferro e cortes de barras metálicas, assim como a pintura para não ocorrer corrosão. Foram muitos protótipos até se chegar ao protótipo final do biofiltro e nas suas conexões, pois muitas vezes ocorriam vazamentos dos gases e era necessário corrigi-los. Os sensores precisaram ser adaptados a estrutura, sendo necessários os ajustes nas conexões e suportes dos sensores como também nas placas desenvolvidas.

5.2. Problemas nos Circuitos Eletrônicos

Com o desenvolvimento foi necessário uma série de modificações no hardware eletrônico, para o bom funcionamento do equipamento.

Com o desenvolvimento da placa amplificadora do sensor de CO_2, inicialmente foi utilizado o amplificador operacional LM324 da National. Este amplificador com uma alimentação de 5 V produzia no máximo uma saída de 3,4V. Assim quando o sinal é convertido pelo conversor A/D e enviado pela serial, os dados pareciam não variar, sem ruído. Após testes o circuito integrado LM324 fora substituído por um amplificador operacional MCP619

da empresa Microchip que tem a característica de fornecer a saída quase 5 V (*Rail-to-rail*). Por consequência, pode-se amplificar mais o sinal do sensor de CO_2.

Outra dificuldade encontrada foi o ruído na saída da placa amplificadora do sensor de CO_2. Quando o filamento do sensor era ligado o ruído aumentava muito, chegando às vezes a 100 mV, alterando muito o sinal adquirido. Foi calculado e desenvolvido um filtro passa baixa passivo entre os dois estágios de amplificação melhorando os resultados. Mas isso gerou outro problema, pois o capacitor do filtro carregava ao longo do tempo e saturava a saída do amplificador operacional. Assim foi necessária a colocação de uma resistência em paralelo com o capacitor removendo este efeito.

Depois de todos estes ajustes ainda havia ruído quando se ligava o filamento do sensor de CO_2. Após muitos testes e isolando os circuitos percebeu-se que o ruído vinha da fonte de alimentação. Como o filamento consumia uma corrente alta em torno de 200 mA, esta provocava uma grande entrada de ruído, sendo solucionado com a colocação de um capacitor de 100uF/25V na alimentação do sensor.

5.3. Problemas na Calibração

Os padrões de gases são bem caros e deve ser solicitada ao fabricante a concentração desejada. Assim na maioria das aplicações onde são necessários padrões gasosos, é realizada somente a calibração com um único ponto.

Com base nestas dificuldades o equipamento desenvolvido teve somente uma verificação de calibração, sendo possibilitado o uso com a autorização da empresa ALAC que disponibilizasse um padrão de 10000 ppm.

Muitos dos testes iniciais foram realizados somente com alguns sopros, onde o CO_2 é oriundo da respiração. Foram realizados alguns testes com gelo seco, mas estes saturavam muito o sinal devido a alta

concentração. Verificou-se com o uso de equipamentos portáteis, que em média um ambiente arejado possuía em torno de 800 ppm de CO_2.

Para as calibrações de temperatura, foram mais acessíveis os padrões, tendo em vista que é mais comum sua utilização nas empresas.

5.4. Conclusões

Hardware e software foram desenvolvidos com sucesso, alcançando os objetivos propostos, ocorrendo também sua realização prática com o desenvolvimento da estrutura mecânica e montagem do biofiltro.

Houve comprovação qualitativa nos resultados dos dados práticos obtidos da curva de crescimento microbiano no software desenvolvido, em comparação com que era esperado pela pesquisa na literatura. Assim foi possível, através do equipamento, verificar em qual etapa de crescimento microbiano o biofiltro se encontrava, a partir da leitura do CO_2, que é produto da respiração microbiana.

Os dados obtidos em todos os experimentos foram guardados e podem ser facilmente filtrados por data e mostrado em forma gráfica dentro do aplicativo desenvolvido. Assim o equipamento é uma ótima ferramenta para análise de crescimento microbiano, servindo para estudos de bioprocessos.

Foi adicionado ao projeto a leitura e registro dos dados de temperatura nas proximidades do sensor de CO_2 para no futuro colaborar com qualquer otimização do processo de biofiltragem.

Ao decorrer do trabalho foi projetado, desenvolvido, modificado e calculado todas as etapas de construção e ajustes do equipamento. Sendo que a tela de calibração dos sensores foi adicionada para facilitar possíveis desvios nas leituras dos sensores e possibilitar ao usuário do equipamento calibrar sem ter de alterar software.

5.5. Sugestões para Trabalhos Futuros

No decorrer do desenvolvimento do projeto e ao longo das apresentações realizadas, foram sugeridas algumas melhorias que

contribuem para o aperfeiçoamento do equipamento. Algumas outras melhorias foram pensadas com o decorrer dos experimentos e testes.

Uma das melhorias seria a colocação de um recipiente com Hipoclorito a 10% na saída do sensor de CO_2 para matar os microrganismos que saem pelo fluxo de gases do experimento, melhorando assim a segurança das pessoas em volta do experimento.

Na tela de calibrações, seria importante a realização da média dos dados adquiridos antes de salvá-los na tabela de calibração. Assim se melhoraria a precisão nos resultados de calibração.

Na tela de aquisições poderia ser adicionada de uma caixa de texto na qual seria possível registrar observações a serem salvas junto com os dados, facilitando posterior análise.

Para salvar os dados no banco depois de coletado os dados da aquisição, poderia ser desenvolvido uma interrupção de software (Thread) para inserir os dados no banco, evitando assim laços de *software* que trancam o programa enquanto se está gravando os dados.

Poderia ser melhorado todo o código com mais funções que realizam a mesma coisa em telas diferentes, simplificando os passos do desenvolvimento e possibilitando o reuso de código.

Poderia ainda ser realizado pelo equipamento o controle de temperatura, para minimizar os custos do projeto, utilizando as saídas à relé da placa. Assim seria possível testar em outras temperaturas para verificar o desempenho.

O equipamento poderia ser também utilizado para fluxo contínuo de crescimento microbiando, sendo necessária outra válvula que removesse o excesso de microrganismos dentro do biofiltro. Assim poderia ser colocado na prática nos aterros sanitários, tendo a função da remoção de poluentes oriundos das chaminés das valas sanitárias.

Outro teste que poderia ser desenvolvido seria a colocação de bioindicadores para monitorar a qualidade do ar, sendo os poluentes removidos pelos microrganismos, o comportamento e a atividades biológicas poderiam ser avaliados por uma câmera de vídeo. Assim teríamos mais elementos de prova que a remoção dos poluentes seria eficiente.

Outra oportunidade de melhoria seria de enviar os dados em um banco de dados da “WEB”, para análise e monitoramento remoto dos dados e processos. Haveria ainda a possibilidade de implementar a comunicação com o equipamento sem fio (*wireless*), melhorando a mobilidade e menor número de cabos.

6. REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR13373**: **Ecotoxicologia aquática - Toxicidade crônica - Método de ensaio com Ceriodaphnia spp (Crustacea, Cladrocera).** Rio de Janeiro: ABNT, 2006.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS**. NBR12648**: **Ecotoxicologia aquática - Toxicidade crônica - Método de ensaio com algas (Chlorophyceae).** Rio de Janeiro: ABNT, 2005.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 12716**: **Água - Ensaio de toxicidade aguda com peixes - Parte III - Sistema de fluxo contínuo - Método de ensaio.** Rio de Janeiro:ABNT, 1993.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR12713: Water acute toxity test with Daphnia similes claus, 1876-Method of test.** Rio de Janeiro: ABNT, 1993

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS**. NBR10004**: **Classificação de resíduos sólidos, Solid waste classification.** Rio de Janeiro: ABNT, 2004.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR10157**: **Aterros de resíduos perigosos – Critérios para projeto, construção e operação.** Rio de Janeiro: ABNT, 1987.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR15088**: **Ecotoxicologia aquática – Toxicidade aguda – Método de ensaio com peixes.** Rio de Janeiro: ABNT, 2004.

AZEVEDO, F.A; CHASIN, A.M. **Metais: Gerenciamento da toxicidade.** São Paulo: Editora Atheneu, 2003.

COCIAN, L.F.E. **Manual da Linguagem C.** Canoas, ULBRA.

CONTI, M.E. **Biological Monitoring: Theory & Aplications.** Boston**:** WITpress, 2008.

COOKSON, John T. **Bioremediation engineering-desing and application**. New York: Mc Graw-Hill Inc.1995.

CRUZ, I.B.M. TAUFER, M. OLIVEIRA, A. K. **Moscas das Frutas de importância econômica no Brasil: Conhecimento básico e aplicado.** Ribeirão Preto: Editora Holos, 2000. Cap. 18, p.143 – 150.

FERNANDEZ ,J.A. REY,A. CARBALLEIRA, A. **An extended study of heavy metal deposition on Galicia (NW Spain) based on moss analysis**. Journal Science of the Total Environment, 2000. pp.31-44, 254 p.

FRIEDRICH, U. VAN LAGENHOVE, H. ALTENDORF, K. LIPSKI, A. **Microbial Community and phisicochemical analysis of an a industrial waste gas biofilter and design of 16S rRNA-targeting oligonucleotide probes**. Environmental Microbiology, 2003. 5(3), pg 183-201.

FUTUYMA, D.J. **Biologia evolutiva.** Segunda edição. Editora Funpec, 2003.

GONÇAVES, Cristina Maria da Silva. **Estudos de biodegradação de COV´s e aplicação na torre biológica de pratos.** 2004. Dissertação submetida a Universidade do Minho.

GOMES, Daiene da Silva. **Resíduos Sólidos Urbanos Gerados no Rio Grande do Sul.** FEPAM, Julho, 2010. Disponível em: <http://www.sfeditora.com.br/tecnologiaambiental/publicacoes.aspx?tipo=artigo&codigo=31> . Acesso em: 13 Maio 2012.

GRANUM, P.E. LUND, T. ***Bacillus cereus* and its food poisoning toxins.** FEMS Microbiology Letter 157 (1997), p.223-228.

JANKE, H. D. **Umweltbiotechnik**. Eugen Ulmer Gmb & Co,2002.

JANTSCHAK ,A. DANIELS, M. PASCHOLD, R. **Biofilter technology: An innovative and cost-effective system to remove VOC**. IEEE transactions on semiconductors manufacturing, 2004. vol 17, NO 3.

LEVENSPIEL, O. **Engenharia das reações químicas.** Tradução da 3 ª edição americana. Editora Edgard Blücher LTDA, 2000

MELO, D. **Bombeiros Alertam sobre acidentes com gás de cozinha.** Belém do Pará: Corpo de Bombeiros do Pará, Abril, 2011. Disponível em: <**http://www.bombeiros.pa.gov.br/index.php/noticias/ultimas-noticias/38-bombeiros-alertam-sobre-acidentes-com-gas-de-cozinha**> . Acesso em: 12 jun.2011.

OLA, I.O. AKINTOKUN,A.K. AKPAN, I, OMOMOWO, I.O. AREO, V.O. **Aerobic decoulourizaton of two reactive azo dyes under varying carbom and nitrogen source by *Bacillus cereus***. African Journal of Biotechnology, Vol. 9(5), p. 672-677, February, 2010. Disponível em :<www.academicjournals.org/AJB>.

PAST. **Paleontological Statistics program. Ver.1.9.** User Manual, Ryan,P.D. 2009. Disponível em: <http://folk.uio.no/ohammer/past/> . Acesso em: 15 maio 2011.

PERRY, H. R. GREEN, D. W**. Perry´s Chemical Engineer´s Handbook.** Seventh edition. Mc Graw Hill Inc,1997.

PROTOCOLO de fabricação de meio de cultura para *Drosophila melanogaster*. **LabDros,** Santa Maria, UFSM,2010. Disponível em**:** <http://w3.ufsm.br/labdros/index.php> . Acesso em: 12 Abr. 2011.

PRZYBULEWSKA,K. WIECZOREK, A. NOWAK, A. **Isolation of microorganisms capable of styrene degradation.** Polish Journal of Environmental Studyes, Vol.15, N.5 (2006), p.777-783

QUADROS, A.F. **Os isópodos terrestres são boas ferramentas para monitorar e restaurar áreas impactadas por metais pesados no Brasil?** Oecologia Australis 14(2),p. 569-583, 2010.

RUSSEL, J.B. **A equação de Nersnt-Quimica Geral.** 2. Ed. São Paulo: Makron books e McGraw Hill,1994.V. 2 cap. 18, p 905-908.

SCHWISTER, U.A. **Taschenbuch der Umwelt-Technik.** Düsseldorf: Fachbuchverlag Leipzig-Fachhochschule, 2003.

SABO F. **Verfahrenstechnische Übersicht und Zusammenhang zwischen Ökonomie und Ökologie.** VDI-Berichte Nr. 1777, Düsseldorf, 2003, ISBN: 3-18-09-1777-2

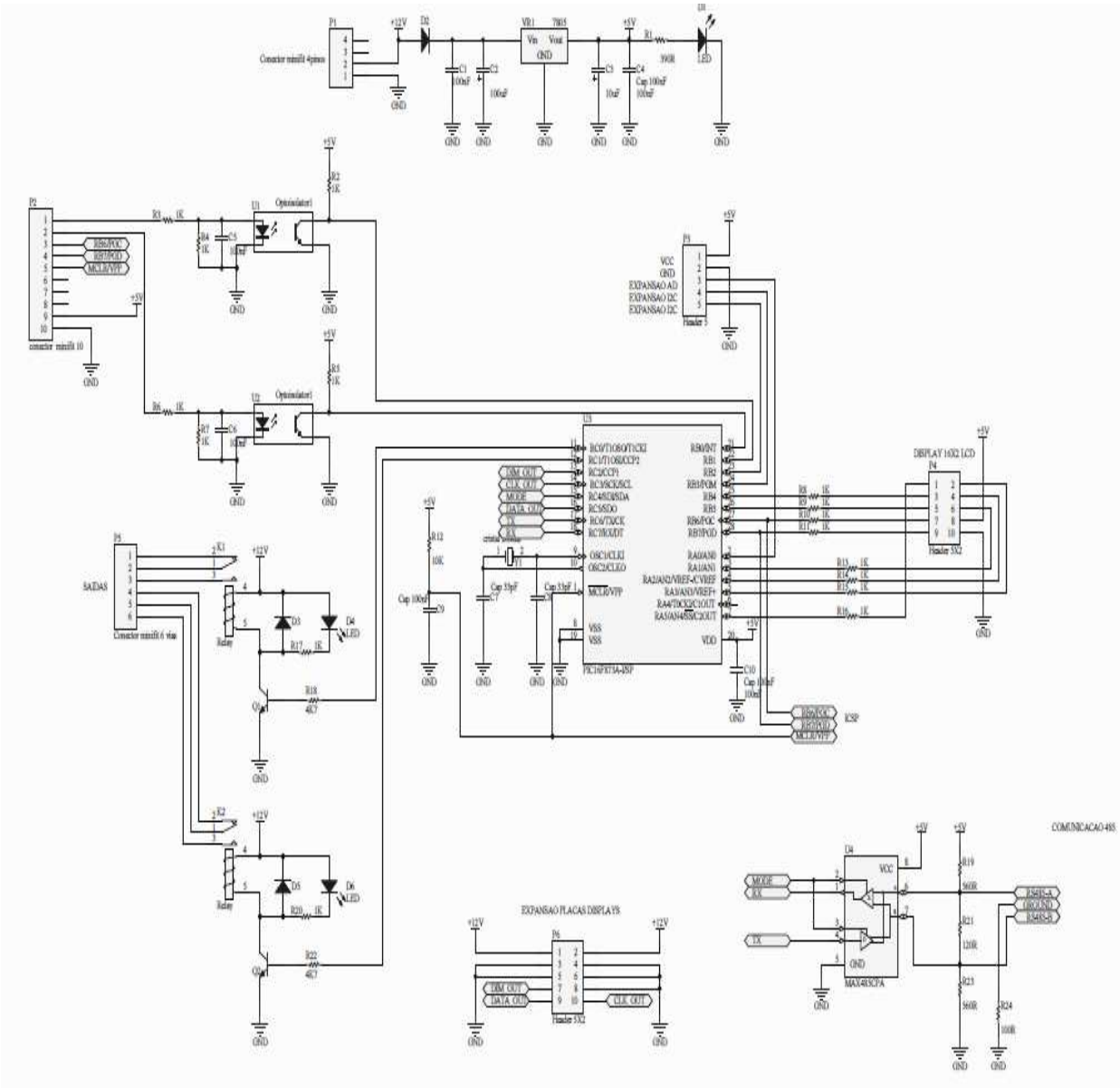
FRANKE W. **Biofilter als Bestandteil kombinierter Abluftbehandlungsverfahren in der Abwasserwirtschaft.** Band 28, KASSEL, 2011, ISBN 978-3-86219-170-3

MCNAIR H.M., BONELLI E.J. **BASIC GAS CHROMATOGRAPHY.** Varian, Berkeley, California, 1968.

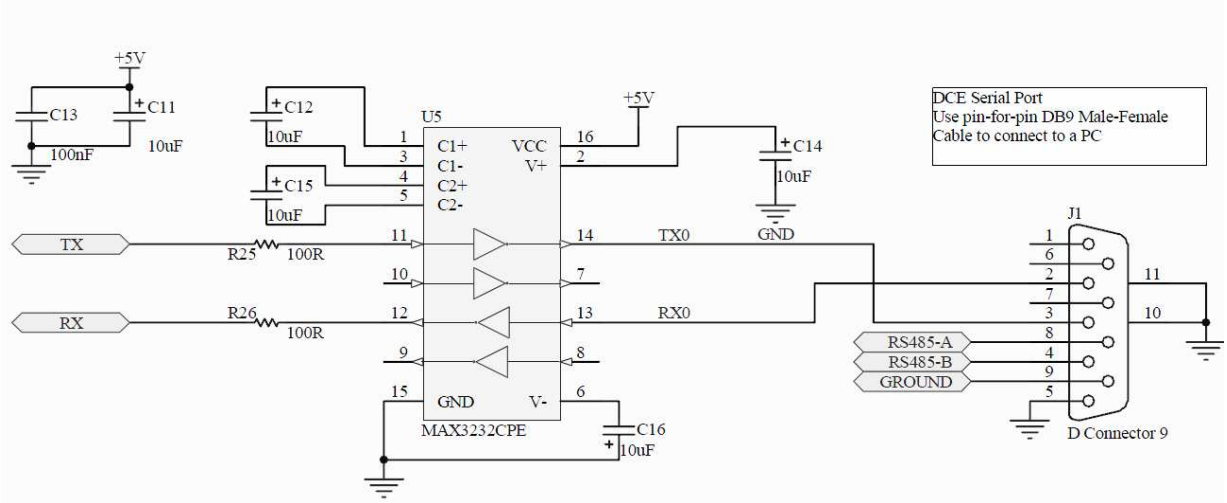
MOREIRA M.A. **Fundamentos do Sensoriamento Remoto e Metodologias de Aplicação.** Editora UFV, Universidade Federal de Viçosa, 2011, Viçosa.

OHLWEILER O.A. **Fundamentos de Análise Instrumental.** LTC Editora S.A., São Paulo, 1981.

APÊNDICE A – PLACA CPU PIC16F916 PARTE 1



APÊNDICE B – PLACA CPU PIC16F916 PARTE 2



APÊNDICE C – PROGRAMA MICROCONTROLADOR EM LINGUAGEM C

```
////////////////////////////////////////////////////////////////////////////
//declacarao estrutura variáveis sensores
typedef struct SENSOR
{
BYTE CO2_AD_L;//valor canal ad ra0 parte menos significativa  variavel global para enviar pela transmissao
BYTE CO2_AD_H;
BYTE LM35_AD_L;//valor canao ra1
BYTE LM35_AD_H;//
};
////////////////////////////////////////////////////////////////////////////
//programa principal
#include "caspel.h"
#define   ENT0              RB0                     //pode ser usado pro nível
#define   ENT1              RB1                     //pode ser usado pro nivel do biofiltro
#define   ENT2              RB4                     //nao implementado
#define   ENT3              RB5                     //nao implementado
#define SAIDA_RELE1       RC0   //saida rele caldo nutritivo
#define SAIDA_RELE2       RC1   //saida controle da temperatura
#define ON                                  1
#define OFF                       0
#define HIGH                      1
#define LOW                       0
#define FALSE                     0
#define TRUE                      1
typedef bit           BOOL;
typedef unsigned char     BYTE;                           // 8-bit
typedef unsigned int      WORD;                           // 16-bit
typedef unsigned long     DWORD;                          // 32-bit
typedef signed char       SBYTE;                          // 8-bit signed
typedef signed int        SWORD;                          // 16-bit signed
typedef signed long       SDWORD;                                 // 32-bit signed
//Configuração dos fusiveis internos do PIC
__CONFIG(HS & WDTEN & PWRTEN & BOREN & DEBUGEN & UNPROTECT );//BITS CONFIGURAÇÃO COM icd2 para 16f916 para testar debug
//Variáveis globais
SWORD temp_prg;
//estrutura declarada com as variáveis do sensor
```

```
struct SENSOR sensor;

void main(void)

{

//inicialização do chip

TRISA = 0b00000011; //seite 46, ra0=entreda sensor co2, resto tudo saida, entrada LM35= RA1

TRISB = 0b00001011; //rb0=entrada/rb1=entrada e resto saida,rb2 liga filamento co2, rb3 le status alarme co2

TRISC = 0b10000000; //rc0=saida rele 1, rc1= saida rele 2, rc6=saida tx, rc7=entrada rx

//valores iniciais das portas acesso direto

PORTA = 0b00000000; //tudo desligado

PORTB = 0b00000000; //tudo desligado

PORTC = 0b00000000; //tudo desligado

OPTION = 0b10001111;//seite 35 o prescaler esta atribuído ao wdt com 1:128//bit7=1 disable pull-up//bit6=0
rbo/int on falling edge//bit5=0 internal clock//bit4=0 increment edge ra4/toclk//bit3=1 prescaler atributed to
Wdt//bit2-0=111 taxa de 1:128

//para o port A deve ser configurado

CMCON0=0b00000000;// colocado em off comparador e port a digital i/o

CMCON1=0b00000000;//sete 119 comparador do timer 1//CMCON1=0b;//nao usado ainda

ANSEL=0b00000011;//ra0=analigico,ra1 analogico ,pagina 181 analog select register  1= entrada analogica e 0
entrada digital

//para ajustar port c driver LCD

LCDCON=0b00000000;//desabilita lcd nos pinos

LCDPS=0b00000000;//desabilita driver lcd

LCDSE0=0b00000000;

LCDSE1=0b00000000;

//configura serial

TXSTA=0b00100010;//sete,132  configurar BRGH bit do sta ler pagina 134  deixado em low speed e usado 9600
de baud rate

RCSTA=0b10010000;//seite 133 , habilita serial,8bits,assincrono,habilirta recepcao

SPBRG=0b00011111;//configurar para 31  calculado pela equacao pagina 134

SSPCON=0b00000000;//tudo 0 desligado SPI

INTCON  = 0;

#if PRESCALER_T1 == 1 //selecionado no arquivo ISR.h  usei 4

T1CON   = 0b00000001;               // Habilita timer1 (TIMER MODE, 1:1)

#elif PRESCALER_T1 == 2

T1CON   = 0b00010001;               // Habilita timer1 (TIMER MODE, 1:2)

#elif PRESCALER_T1 == 4

T1CON   = 0b00100101;               // Habilita timer1 (TIMER MODE, 1:4)//20/11/11 alterei para nao
sincronizar pino clock externo

//T1CON   = 0b00100101;             // Habilita timer1 (TIMER MODE, 1:4)//para16f916//deu no mesmo

#elif PRESCALER_T1 == 8

T1CON   = 0b00110001;               // Habilita timer1 (TIMER MODE, 1:8)

#endif

CCP1IE  = ON;                                   // habilita interrupção do módulo capture/compare 1

CCP1CON = 0b00001011;               // habilita modulo compare CCP1

CCPR1H  = (INSTRUCTION_CYCLE/(TIMER1_FREQ*PRESCALER_T1)) / 256;

CCPR1L  = (INSTRUCTION_CYCLE/(TIMER1_FREQ*PRESCALER_T1)) % 256;

PIE1   = 0b01100100;        //cuidado se habilito interrup de transmissao vai para loop infinito na interrupcao
          //seite 37//o bit 0 habilita interrupt tmr1/habilita interrupção canal ad
```

```
PIR1   = 0b00001000;                    //seite 22//o ADIF-bit 6  seite 39 ler pois sao as flags das interrupcoes
PIE2   = 0b00000000;                    //seite 23//desabilitada interrupção de escrita da eeprom?????quando
usar???
PIR2   = 0b00000000;                    //seite 24//bit se escreveu certo na eeprom deve ser limpo por software
RBPU     = ON;                                          // Desabilita pull-ups da portb
PEIE     = ON;              //habilita interrupcao dos perifericos
GIE    = ON;                                            // habilita interrupcao global
////////////////////////////////////////////////////////
DelayMs(500);      // Delay inicialização das entradas
//inicializacao();//chama função de inicializacao dos registradores porta e interrupçoes no arquivo iniicialização.c
e .h
////////////////////////////////////////////////////////////////////////////////
          while(1)//loop infinito
          {
          CLRWDT();//limpa watchdog
////////////////////////////////////////////////////////////////////////////
//le canal ra0 sensor co2 pelo comando 4 leio valor do sensor co2
ADCON0=0b10000001;//seleciona canal ra0
DelayUs(100);      // Delay inicialização das entradas  quando trocar canais analógicos, obrigatório usar
GODONE = ON;
          while(GODONE);
sensor.CO2_AD_L=ADRESL;
sensor.CO2_AD_H=ADRESH;

////////////////////////////////////////////////////////////////////
//le canal ra1 sensor de temperatura LM35 pelo comando 8 leio o sensor de temperatura
ADCON0=0b10000101;//seleciona canal ra1
DelayUs(100);      // Delay inicialização das entradas  //teste
GODONE = ON;
          while(GODONE);
sensor.LM35_AD_L=ADRESL;
sensor.LM35_AD_H=ADRESH;

          }//fim do loop while(1) infinito
}//fim do arquivo main
////////////////////////////////////////////////////////////////////////
//Função de interrupção, quando ocorre entra aqui
void interrupt ISR(void)
{
          if(CCP1IF){
                                    CCP1IF=0;
                                    evento.timer = ON; //a cada interrupcao libera conversao
                                    }
          if(ADIF){
                                    ADIF=0;

                                    }
```

```
if(T0IF){
                    T0IF=0;
                    }
if(TMR1IF){
                    TMR1IF=0;
                    }
if(SSPIF){
                    SSPIF=0;
                    }
if(RCIF){
                    BYTE a;
          a=RCREG;//devemos ler para limpar a interrupcao
          TXREG=a;//envia de volta para testar e saber o que esta chegando
          if(a=='0')//funcionou vivas  mas se envio mais que 4 bites ele entra em crepe e nao
          funca mais ate resetar
          //para testar saídas
          SAIDA_RELE1=ON;
          if(a=='1')//funcionou vivas  mas se envio mais que 4 bites ele entra em crepe e nao
          funca mais ate resetar
          //para testar saídas
          SAIDA_RELE1=OFF;
          if(a=='2')//funcionou vivas  mas se envio mais que 4 bites ele entra em crepe e nao
     funca mais ate resetar
          //para testar saídas
          SAIDA_RELE2=ON;
          if(a=='3')//funcionou vivas  mas se envio mais que 4 bites ele entra em crepe e nao
funca mais ate resetar
          //para testar saídas
          SAIDA_RELE2=OFF;
     ///////testar canal A/d
          if(a=='4'){//funcionou vivas  mas se envio mais que 4 bites ele entra em crepe e nao
funca mais ate resetar
                    TXREG=sensor.CO2_AD_L;
                    do{
                    DelayUs(10);        // Delay
                    }
                    while(!TXIF);//consegui transmitir os dois bytes da conversao a/d
                    TXREG=sensor.CO2_AD_H;
          }
          if(a=='5'){
          RB2=ON;
          }
          if(a=='6'){//operando
          //desliga rb2
          RB2=OFF;
          }
          if(a=='7'){//tafuncionando 21/11/11
          static BOOL status_alarme;
```

```
                    status_alarme=RB3;
                    TXREG=status_alarme;//funcionando
                    }
                if(a=='8'){//funcionou vivas  mas se envio mais que 4 bites ele entra em crepe e nao funca
mais ate resetar
                        //para testar envio canal a/d
                        TXREG=sensor.LM35_AD_L;//funcionou....viva
                        do{
                        DelayUs(10);        // Delay
                        }
                        while(!TXIF);//consegui transmitir os dois bytes da conversao a/d
                        TXREG=sensor.LM35_AD_H;
                    }
                    }        //fim da função de interrupcao
```

APÊNDICE D – PROGRAMA COMPUTADOR EM LINGUAGEM C#

```
#region classes declaradas de outros projetos  //29/8/11
using GraphLib;  //acrescentado biblioteca gráfica
#endregion
namespace grafico_com_banco
{
    public partial class Form1 : Form
    {
        #region inicializacao variaveis globais
        #region inicializaçao da classe grafica e objeto     //25-8-11 acrescentado
        private int NumGraphs = 1;//15/12/11
        private String CurExample = "TILED_VERTICAL_AUTO";
        private String CurColorSchema = "GRAY";
        private PrecisionTimer.Timer mTimer = null;
        private DateTime lastTimerTick = DateTime.Now;
        #endregion
        #region variaveis usada na tela aquisicao.
        //colocar aqui variavei globais
        public int temperatura=0;
        public int co2 = 0;//21/2/12
        public bool receive = false; //variavel de controle de laco de recepcao seria  se 0
segue comandos da tela de comunicacao se 1 segue tela de aquisicao
        public int valor_serial = 0;//valor para colocar no data grid 2
        int[]  Rx_dado= new int[10];//bufer de recepcao
        int count_rx = 0;//variavel que conta os bytres recebidos para montar o frame
        public bool read = false;//variavel de controle de laco da thread da aquisicao
        public bool aquisition = false;//variavel para nao enviar se nao leu tudo
        #endregion
        #region variaveis usada na tela calibracao
        /// ////////////////variaveis para serem usadas na calibracao
        /////variaveis globais
        public int temperatura_cal = 0;
        public int co2_cal = 0;//3/3/12
        public bool receive_cal = false; //variavel de controle de laco de recepcao seria  se
0 segue comandos da tela de comunicacao se 1 segue tela de calibracao
        public int valor_serial_cal = 0;//valor para colocar no data grid 2
        int[] Rx_dado_cal = new int[10];//bufer de recepcao
        int count_rx_cal = 0;//variavel que conta os bytres recebidos para montar o frame
        public bool read_cal = false;//variavel de controle de laco da thread da aquisicao
        public bool aquisition_cal = false;//variavel para nao enviar se nao leu tudo
        #endregion
        #region variaveis globais usadas para estatistica da tela aquisicao
        /// ////////////////////////////////////////estatisticas
        public int conv_ad = 0;//varialvel para mostrar datagrit valor total da conversao ad
        public int conv_adh = 0;//varialvel para mostrar valor mais significativo valor total
da conversao ad
        public double min_co2 = 0;//variavel estatistica minimo valor
        public double med_co2 = 0;//variavel estatistica med valor
        public double max_co2 = 0;//variavel estatistica maximo valor
        public double total_ad_co2 = 0;//variavel com valor total para calculos estatisticos
        public bool flag_min_co2 = false;//variavel para entrar no laco de valor minimo so uma
vez
        public double total_media_co2 = 0;//para calculo da media
        public bool flag_max_co2 = false;//
        public double variancia_co2 = 0;
```

```
        public double rsd_co2 = 0;//variavel estatistica percentagem desvio padrao
        public double rsd_co2_percent = 0;//variavel estatistica percentagem desvio padrao
        public int conv_ad_temp = 0;//varialvel para mostrar datagrit valor total da conversao
ad
        public int conv_adh_temp = 0;//varialvel para mostrar valor mais significativo valor
total da conversao ad
        public double min_temp = 0;//variavel estatistica minimo valor
        public double med_temp = 0;//variavel estatistica med valor
        public double max_temp= 0;//variavel estatistica maximo valor
        public double total_ad_temp = 0;//variavel com valor total para calculos estatisticos
        public bool flag_min_temp = false;//variavel para entrar no laco de valor minimo so
uma vez
        public double total_media_temp = 0;//para calculo da media
        public bool flag_max_temp = false;//
        public double variancia_temp = 0;
        public double rsd_temp = 0;//variavel estatistica percentagem desvio padrao
        public double rsd_temp_percent = 0;//variavel estatistica percentagem desvio padrao
        #endregion
        #region criando thread
        Thread Tr; //thread aquisition
        Thread Tr1;//thread calibration
        #endregion
        #region inicializando funções delegate
        internal delegate void Grid_Delegate();//passo nada no delegate exemplo de nehuma
passagem dce parametros.
        internal delegate void Grid_Delegate_cal();//para acessar as outras threads e mudar
parametros na calibracao
        #endregion
        #endregion
        #region inicializacao do form principal itens default
        public Form1()
        {
            InitializeComponent();
            #region inicializando regioes default sobre o banco de dados
            textBox1.Text = "localhost";//sempre string pro banco 22/8/11, se nao for
converter.
            textBox2.Text = "co2_schema";
            textBox3.Text = "root";
            textBox4.Text = "caspel";
            textBox5.Text = "grafico";
            textBox11.Text = "mysql6.webserverwin.com";
            textBox12.Text = "co2";
            textBox13.Text = "casara";
            textBox14.Text = "caspel";
            textBox15.Text = "grafico";
            textBox16.Text = "3306";//porta local
            textBox17.Text = "3306";//porta web
            textBox18.Text = "1";//PORTA SERIAL
            textBox19.Text = "1";//comandos 21/2/12
            radioButton1.Checked = true;//seleciona gravar no banco local
            #endregion
            #region grafico inicializando
            #endregion
        }
        #endregion

        #region Fechando o Form principal
        private void Form1_FormClosed(object sender, FormClosedEventArgs e)
        {
            Tr = null; //kill thread
            Tr1 = null;
            serialPort1.Close();//obrigatorio para deixar com fechada e poder abrir de novo
        }
        #endregion

        //banco de dados
        #region botoes aplicacao principal
        #region botao test conection on localhost
        private void button4_Click(object sender, EventArgs e)
        {
            string server, database, user, passwd , port;
            server = textBox1.Text;
            database = textBox2.Text;
            user = textBox3.Text;
            passwd = textBox4.Text;
            port = textBox16.Text;
```

```
            MySqlConnection conection = new MySqlConnection("Server=" + server + ";Database="
+ database + ";Uid=" + user + ";Pwd=" + passwd + "");//só pode variável string se numerico
            try
            {
                if (conection.State == ConnectionState.Closed)
                {
                    conection.Open();
                    MessageBox.Show("conecção realizada com sucesso");

                }
                //comando.ExecuteNonQuery();
                conection.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.StackTrace);
            }
        }

        #endregion
        #region botao test conection on web

        private void button6_Click(object sender, EventArgs e)
        {
            string server, database, user, passwd, port;
            server = textBox11.Text;
            database = textBox12.Text;
            user = textBox13.Text;
            passwd = textBox14.Text;
            port = textBox17.Text;
            MySqlConnection conection = new MySqlConnection("Server=" + server + ";Database="
+ database + ";Uid=" + user + ";Pwd=" + passwd + "");//só pode variável string se numerico
devo usar variavel.tostring()
            try
            {
                if (conection.State == ConnectionState.Closed)
                {
                    conection.Open();
                    MessageBox.Show("conecção web realizada com sucesso");

                }
                //comando.ExecuteNonQuery();
                conection.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.StackTrace);
            }


        }

        #endregion
        #region botao salvar no banco
        private void button2_Click(object sender, EventArgs e)
        {
            if (radioButton1.Checked)
            {
                //MessageBox.Show("radiobutton localhost habilitado");//funcionando 29/8
                string server, database, user, passwd, tabela;
                server = textBox1.Text;
                database = textBox2.Text;
                user = textBox3.Text;
                passwd = textBox4.Text;
                tabela = textBox5.Text;
                MySqlConnection conection = new MySqlConnection("Server=" + server +
";Database=" + database + ";Uid=" + user + ";Pwd=" + passwd + "");//só pode variável string se
numerico devo usar variavel.tostring()
                 try
                {
                    if (conection.State == ConnectionState.Closed)
                        conection.Open();

                    //comando.ExecuteNonQuery();
                    conection.Close();
                }
                catch (Exception ex)
```

```
                {
                    MessageBox.Show(ex.StackTrace);
                }
            }
            else
            {
                string server, database, user, passwd, tabela, port;
                server = textBox11.Text;
                database = textBox12.Text;
                user = textBox13.Text;
                passwd = textBox14.Text;
                tabela = textBox15.Text;
                port = textBox17.Text;
                MySqlConnection conection = new MySqlConnection("Server=" + server +
";Database=" + database + ";Uid=" + user + ";Pwd=" + passwd + "");//só pode variável string se
numerico devo usar variavel.tostring()
                 try
                {
                    if (conection.State == ConnectionState.Closed)
                        conection.Open();
                    DataTable tabela1 = new DataTable();
                     dataGridView1.Columns.Clear();
                    dataGridView1.DataSource = null;
                    dataGridView1.DataSource = tabela1;

                    conection.Close();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.StackTrace);
                }

            }
        }

        #endregion

        #region botao update no banco

        private void button5_Click(object sender, EventArgs e)
        {
            if(radioButton1.Checked)
            {
            string server, database, user, passwd, tabela;
            server = textBox1.Text;
            database = textBox2.Text;
            user = textBox3.Text;
            passwd = textBox4.Text;
            tabela = textBox5.Text;
             MySqlConnection conection = new MySqlConnection("Server=" + server + ";Database="
+ database + ";Uid=" + user + ";Pwd=" + passwd + "");//só pode variável string se numerico
devo usar variavel.tostring()
            try
            {
                if (conection.State == ConnectionState.Closed)
                    conection.Open();
               //comando.ExecuteNonQuery();
                conection.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.StackTrace);
            }
            }
            else
            {
                string server, database, user, passwd, tabela, port;
                server = textBox11.Text;
                database = textBox12.Text;
                user = textBox13.Text;
                passwd = textBox14.Text;
                tabela = textBox15.Text;
                port = textBox17.Text;
                MySqlConnection conection = new MySqlConnection("Server=" + server +
";Database=" + database + ";Uid=" + user + ";Pwd=" + passwd + "");//só pode variável string se
numerico devo usar variavel.tostring()
                try
```

```
                {
                    if (conection.State == ConnectionState.Closed)
                        conection.Open();
                    DataTable tabela1 = new DataTable();
                    dataGridView1.Columns.Clear();
                    dataGridView1.DataSource = null;
                    dataGridView1.DataSource = tabela1;

                    conection.Close();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.StackTrace);
                }
            }
        }

        #endregion

        #region botao buscar no banco

        private void button1_Click(object sender, EventArgs e)
        {
            if (radioButton1.Checked)//banco local host
            {
                #region tabela co2
                if (radioButton3.Checked)//tabela co2
                {

                    string server, database, user, passwd, tabela, port;
                    server = textBox1.Text;
                    database = textBox2.Text;
                    user = textBox3.Text;
                    passwd = textBox4.Text;
                    //tabela = textBox5.Text;
                    tabela = "tabela_co2";
                    port = textBox16.Text;
                    MySqlConnection conection = new MySqlConnection("Server=" + server +
";Database=" + database + ";Uid=" + user + ";Pwd=" + passwd + "");//só pode variável string se
numerico devo usar variavel.tostring()
                    ////////////////////captura faixa de datas para realizar a busca
                    string aba;
                    string baba;
                    aba = Convert.ToString(dateTimePicker1.Value);//data inicial
                    baba = Convert.ToString(dateTimePicker2.Value);
                    ////////////monta data inicial
                    DateTime busca_banco, busca_banco_1;
                    string _datum_bus, _day_bus, _month_bus, _year_bus, _monta_datum_bus,
_monta_datum_bus_final;
                    _datum_bus = aba;
                    busca_banco = Convert.ToDateTime(_datum_bus);
                    string _datum_bus_1;
                    _datum_bus_1 = baba;
                    busca_banco_1 = Convert.ToDateTime(_datum_bus_1);


                    _day_bus = Convert.ToString(busca_banco.Day);
                    _month_bus = Convert.ToString(busca_banco.Month);
                    _year_bus = Convert.ToString(busca_banco.Year);
                    _monta_datum_bus = _year_bus + "-" + _month_bus + "-" +
_day_bus;//montando para salvar no banco tabela tempo
                    _day_bus = Convert.ToString(busca_banco_1.Day);
                    _month_bus = Convert.ToString(busca_banco_1.Month);
                    _year_bus = Convert.ToString(busca_banco_1.Year);
                    _monta_datum_bus_final = _year_bus + "-" + _month_bus + "-" +
_day_bus;//montando para salvar no banco tabela tempo
                    MySqlCommand comando = new MySqlCommand("SELECT tempo,CONV,DATA FROM " +
tabela + " WHERE " + tabela + ".DATA BETWEEN CONVERT ('" + _monta_datum_bus + "',DATE ) AND
CONVERT ('" + _monta_datum_bus_final + "',DATE ) ", conection);                    //funcionou
                    MySqlDataAdapter leitura = new MySqlDataAdapter(comando);
                    try
                    {
                        if (conection.State == ConnectionState.Closed)
                            conection.Open();
                        DataTable tabela1 = new DataTable();
                        leitura.Fill(tabela1);
                        dataGridView1.Columns.Clear();
```

```
                        dataGridView1.DataSource = null;
                        dataGridView1.DataSource = tabela1;
                        conection.Close();
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show(ex.StackTrace);
                    }
                }
                #endregion
                #region tabela temperatura
                if (radioButton4.Checked)//tabela temperatura
                {
                    string server, database, user, passwd, tabela, port;
                    server = textBox1.Text;
                    database = textBox2.Text;
                    user = textBox3.Text;
                    passwd = textBox4.Text;
                    tabela = "tabela_temperatura";
                    port = textBox16.Text;
                    MySqlConnection conection = new MySqlConnection("Server=" + server +
";Database=" + database + ";Uid=" + user + ";Pwd=" + passwd + "");//só pode variável string se
numerico devo usar variavel.tostring()

                    ///////monta data para a busca
                    ////////////////////captura faixa de datas para realizar a busca
                    string aba;
                    string baba;
                    aba = Convert.ToString(dateTimePicker1.Value);//data inicial
                    baba = Convert.ToString(dateTimePicker2.Value);
                    ////////////monta data inicial
                    DateTime busca_banco;
                    string _datum_bus, _day_bus, _month_bus, _year_bus, _monta_datum_bus,
_monta_datum_bus_final;
                    _datum_bus = aba;
                    busca_banco = Convert.ToDateTime(_datum_bus);
                    _day_bus = Convert.ToString(busca_banco.Day);
                    _month_bus = Convert.ToString(busca_banco.Month);
                    _year_bus = Convert.ToString(busca_banco.Year);
                    _monta_datum_bus = _year_bus + "-" + _month_bus + "-" +
_day_bus;//montando para salvar no banco tabela tempo
                    ////////////////////////////////monta data final
                    _datum_bus = baba;
                    busca_banco = Convert.ToDateTime(_datum_bus);
                    _day_bus = Convert.ToString(busca_banco.Day);
                    _month_bus = Convert.ToString(busca_banco.Month);
                    _year_bus = Convert.ToString(busca_banco.Year);
                    _monta_datum_bus_final = _year_bus + "-" + _month_bus + "-" +
_day_bus;//montando para salvar no banco tabela tempo
                    /////monta comando para buscar no banco
                    MySqlCommand comando = new MySqlCommand("SELECT tempo,CONV,DATA FROM " +
tabela + " WHERE " + tabela + ".DATA BETWEEN CONVERT ('" + _monta_datum_bus + "',DATE ) AND
CONVERT ('" + _monta_datum_bus_final + "',DATE ) ", conection);                    //funcionou
                    MySqlDataAdapter leitura = new MySqlDataAdapter(comando);
                    try
                    {
                        if (conection.State == ConnectionState.Closed)
                            conection.Open();
                        DataTable tabela1 = new DataTable();
                        leitura.Fill(tabela1);
                        dataGridView1.Columns.Clear();
                        dataGridView1.DataSource = null;
                        dataGridView1.DataSource = tabela1;
                        conection.Close();
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show(ex.StackTrace);
                    }
                }
                #endregion
                #region tabela configuracao
        #endregion
            }
        }

        #endregion
```

```
        #region botao faz gráfico
        private void button3_Click(object sender, EventArgs e)
        {
            display.Smoothing = System.Drawing.Drawing2D.SmoothingMode.None;//troquei nome
objeto pra display 29-9-11
            //funcoes a serem chamadas
            Graphtabela_co2(); //funcao para ler da tabela e mostar na tela
            display.Refresh();//troquei nome do objeto
            UpdateGraphCountMenu();
            UpdateColorSchemaMenu();
            mTimer = new PrecisionTimer.Timer();
            mTimer.Period = 40;                         // 20 fps
            mTimer.Tick += new EventHandler(OnTimerTick);//evento que funciona com grafico
            lastTimerTick = DateTime.Now;
            mTimer.Start();
        }
        #endregion
        #endregion
        #region eventos menustrip
        #region update graphs count //funcao que atualiza contagem do menustrip da etapa num
graphs
        private void UpdateGraphCountMenu()
        {
            toolStripMenuItem2.Checked = false;
            toolStripMenuItem3.Checked = false;
            toolStripMenuItem4.Checked = false;
            toolStripMenuItem5.Checked = false;
            toolStripMenuItem6.Checked = false;
            toolStripMenuItem7.Checked = false;

            switch (NumGraphs)
            {
                case 1: toolStripMenuItem2.Checked = true; break;
                case 2: toolStripMenuItem3.Checked = true; break;
                case 3: toolStripMenuItem4.Checked = true; break;
                case 4: toolStripMenuItem5.Checked = true; break;
                case 5: toolStripMenuItem6.Checked = true; break;
                case 6: toolStripMenuItem7.Checked = true; break;

            }
        }
        #endregion
        #region menu e eventos num graphs
        private void toolStripMenuItem2_Click(object sender, EventArgs e)
        {
            NumGraphs = 1;
            CalcDataGraphs();
            UpdateGraphCountMenu();
        }

        private void toolStripMenuItem3_Click(object sender, EventArgs e)
        {
            NumGraphs = 2;
            CalcDataGraphs();
            UpdateGraphCountMenu();
        }
        private void toolStripMenuItem4_Click(object sender, EventArgs e)
        {
            NumGraphs = 3;
            CalcDataGraphs();
            UpdateGraphCountMenu();
        }

        private void toolStripMenuItem5_Click(object sender, EventArgs e)
        {
            NumGraphs = 4;
            CalcDataGraphs();
            UpdateGraphCountMenu();
        }

        private void toolStripMenuItem6_Click(object sender, EventArgs e)
        {
            NumGraphs = 5;
            CalcDataGraphs();
            UpdateGraphCountMenu();
        }
```

```
        private void toolStripMenuItem7_Click(object sender, EventArgs e)
        {
            NumGraphs = 6;
            CalcDataGraphs();
            UpdateGraphCountMenu();
        }
        #endregion

        #region update coplor schema menu  //funcao que atualiza schema

        private void UpdateColorSchemaMenu()
        {
            blueToolStripMenuItem.Checked = false;
            whiteToolStripMenuItem.Checked = false;
            grayToolStripMenuItem.Checked = false;
            lightBlueToolStripMenuItem.Checked = false;
            blackToolStripMenuItem.Checked = false;
            redToolStripMenuItem.Checked = false;

            if (CurColorSchema == "WHITE") whiteToolStripMenuItem.Checked = true;
            if (CurColorSchema == "BLUE") blueToolStripMenuItem.Checked = true;
            if (CurColorSchema == "GRAY") grayToolStripMenuItem.Checked = true;
            if (CurColorSchema == "LIGHT_BLUE") lightBlueToolStripMenuItem.Checked = true;
            if (CurColorSchema == "BLACK") blackToolStripMenuItem.Checked = true;
            if (CurColorSchema == "RED") redToolStripMenuItem.Checked = true;
            if (CurColorSchema == "DARK_GREEN") greenToolStripMenuItem.Checked = true;
        }
        #endregion
        #region menu e eventos color schemes
        private void blueToolStripMenuItem_Click(object sender, EventArgs e)
        {
            CurColorSchema = "BLUE";
            CalcDataGraphs();
            UpdateColorSchemaMenu();
        }
        private void whiteToolStripMenuItem_Click(object sender, EventArgs e)
        {
            CurColorSchema = "WHITE";
            CalcDataGraphs();
            UpdateColorSchemaMenu();
        }
        private void grayToolStripMenuItem_Click(object sender, EventArgs e)
        {
            CurColorSchema = "GRAY";
            CalcDataGraphs();
            UpdateColorSchemaMenu();
       }

        private void lightBlueToolStripMenuItem_Click(object sender, EventArgs e)
        {
            CurColorSchema = "LIGHT_BLUE";
            CalcDataGraphs();
            UpdateColorSchemaMenu();
        }

        private void blackToolStripMenuItem_Click(object sender, EventArgs e)
        {
            CurColorSchema = "BLACK";
            CalcDataGraphs();
            UpdateColorSchemaMenu();
        }

        private void redToolStripMenuItem_Click(object sender, EventArgs e)
        {
            CurColorSchema = "RED";
            CalcDataGraphs();
            UpdateColorSchemaMenu();
        }

        private void greenToolStripMenuItem_Click(object sender, EventArgs e)
        {
            CurColorSchema = "DARK_GREEN";
            CalcDataGraphs();
            UpdateColorSchemaMenu();
        }
        #endregion
```

```
        #endregion
        #region funcoes do graficos
        void Graphsensor_temperatura(){//será implementada
        //funcao para chamar grafico do sensor de temperatura
        }

       void Graphsensor_co2() {
            //inicio
            this.SuspendLayout();
            display.DataSources.Clear();
            display.SetDisplayRangeX(0, 10000);
            int j = 0; //contem o numero de graficos
            display.DataSources.Add(new DataSource());
            display.DataSources[j].Name = "Graph " + (j + 1);
            display.DataSources[j].OnRenderXAxisLabel += RenderXLabel;
           //arrumando
            this.Text = "Tiled Graphs (vertical prefered)";
            display.PanelLayout = PlotterGraphPaneEx.LayoutMode.TILES_VER;
            display.DataSources[j].Length = 15800;
            display.DataSources[j].AutoScaleY = true;//funcionou
            //display.DataSources[j].AutoScaleY = false;
            display.DataSources[j].SetDisplayRangeY(-600, 600);
            display.DataSources[j].SetGridDistanceY(1);
            //funcao a ser chamada
            retaFunction(display.DataSources[j], j);//funcao para achar equacao da reta
            //fim
            ApplyColorSchema();
            this.ResumeLayout();
            display.Refresh();
        }

        void Graphtabela_co2() {
            //funcao para ler da tabela e mostar na tela
            this.SuspendLayout();
            display.DataSources.Clear();
            //1-buscar dados no banco
            //2-ler numero de linhas na tabela
            int numero_linhas = 0;//variavel para colocar no grafico
            numero_linhas=dataGridView1.RowCount;//busca o numero de linhas buscado do banco
            //3-automatico faixa de dados
            //display.SetDisplayRangeX(0, 100);
            display.SetDisplayRangeX(0, numero_linhas);//devo buscar no banco antes para
mostar as linhas
            display.SetGridDistanceX(100);//14/2/12
            display.SetGridOriginX(0);
            //display.SetDisplayRangeX(0, 10000);
            int j = 0; //contem o numero de graficos
            display.DataSources.Add(new DataSource());
            display.DataSources[j].Name = "Graph " + (j + 1);
            display.DataSources[j].OnRenderXAxisLabel += RenderXLabel;
            //arrumando para pegar da tabela
            this.Text = "Tiled Graphs (vertical prefered)";
            display.PanelLayout = PlotterGraphPaneEx.LayoutMode.TILES_VER;
            //devo sempre definir o comprimento dos dados.
            //display.DataSources[j].Length = 15800;
            display.DataSources[j].Length = numero_linhas;//numero de linhas da tabela
            display.DataSources[j].AutoScaleY = true;//funcionou
            display.DataSources[j].AutoScaleX = false;//14/2/12
            //display.DataSources[j].AutoScaleY = false;
            //arrumando escala no Y
            display.DataSources[j].SetDisplayRangeY(0, 1600);//2/3/12
            display.DataSources[j].SetGridDistanceY(1000);//2/3/12
             tabelaFunction(display.DataSources[j], j);
            ApplyColorSchema();//2/3/12
        }
        protected void CalcDataGraphs()
        {
            this.SuspendLayout();
            display.DataSources.Clear();
            display.SetDisplayRangeX(0, 400);
            for (int j = 0; j < NumGraphs; j++)
            {
                display.DataSources.Add(new DataSource());
                display.DataSources[j].Name = "Graph " + (j + 1);
                display.DataSources[j].OnRenderXAxisLabel += RenderXLabel;
                switch (CurExample)
```

```
{
    case "NORMAL":
        this.Text = "Normal Graph";
        display.DataSources[j].Length = 5800;
        display.PanelLayout = PlotterGraphPaneEx.LayoutMode.NORMAL;
        display.DataSources[j].AutoScaleY = false;
        display.DataSources[j].SetDisplayRangeY(-300, 300);
        display.DataSources[j].SetGridDistanceY(100);
        CalcSinusFunction_0(display.DataSources[j], j);
        break;
    case "NORMAL_AUTO":
        this.Text = "Normal Graph Autoscaled";
        display.DataSources[j].Length = 5800;
        display.PanelLayout = PlotterGraphPaneEx.LayoutMode.NORMAL;
        display.DataSources[j].AutoScaleY = true;
        display.DataSources[j].SetDisplayRangeY(-300, 300);
        display.DataSources[j].SetGridDistanceY(100);
        CalcSinusFunction_0(display.DataSources[j], j);
        break;
    case "STACKED":
        this.Text = "Stacked Graph";
        display.PanelLayout = PlotterGraphPaneEx.LayoutMode.STACKED;
        display.DataSources[j].Length = 5800;
        display.DataSources[j].AutoScaleY = false;
        display.DataSources[j].SetDisplayRangeY(-250, 250);
        display.DataSources[j].SetGridDistanceY(100);
        CalcSinusFunction_1(display.DataSources[j], j);
        break;
    case "VERTICAL_ALIGNED":
        this.Text = "Vertical aligned Graph";
        display.PanelLayout = PlotterGraphPaneEx.LayoutMode.VERTICAL_ARRANGED;
        display.DataSources[j].Length = 5800;
        display.DataSources[j].AutoScaleY = false;
        display.DataSources[j].SetDisplayRangeY(-300, 300);
        display.DataSources[j].SetGridDistanceY(100);
        CalcSinusFunction_2(display.DataSources[j], j);
        break;
    case "VERTICAL_ALIGNED_AUTO":
        this.Text = "Vertical aligned Graph autoscaled";
        display.PanelLayout = PlotterGraphPaneEx.LayoutMode.VERTICAL_ARRANGED;
        display.DataSources[j].Length = 5800;
        display.DataSources[j].AutoScaleY = true;
        display.DataSources[j].SetDisplayRangeY(-300, 300);
        display.DataSources[j].SetGridDistanceY(100);
        CalcSinusFunction_2(display.DataSources[j], j);
        break;
    case "TILED_VERTICAL":
        this.Text = "Tiled Graphs (vertical prefered)";
        display.PanelLayout = PlotterGraphPaneEx.LayoutMode.TILES_VER;
        display.DataSources[j].Length = 5800;
        display.DataSources[j].AutoScaleY = false;
        display.DataSources[j].SetDisplayRangeY(-300, 600);
        display.DataSources[j].SetGridDistanceY(100);
        CalcSinusFunction_2(display.DataSources[j], j);
        break;
    case "TILED_VERTICAL_AUTO":
        this.Text = "Tiled Graphs (vertical prefered) autoscaled";
        display.PanelLayout = PlotterGraphPaneEx.LayoutMode.TILES_VER;
        display.DataSources[j].Length = 5800;
        display.DataSources[j].AutoScaleY = true;
        display.DataSources[j].SetDisplayRangeY(-300, 600);
        display.DataSources[j].SetGridDistanceY(100);
        CalcSinusFunction_2(display.DataSources[j], j);
        break;
    case "TILED_HORIZONTAL":
        this.Text = "Tiled Graphs (horizontal prefered)";
        display.PanelLayout = PlotterGraphPaneEx.LayoutMode.TILES_HOR;
        display.DataSources[j].Length = 5800;
        display.DataSources[j].AutoScaleY = false;
        display.DataSources[j].SetDisplayRangeY(-300, 600);
        display.DataSources[j].SetGridDistanceY(100);
        CalcSinusFunction_2(display.DataSources[j], j);
        break;
    case "TILED_HORIZONTAL_AUTO":
        this.Text = "Tiled Graphs (horizontal prefered) autoscaled";
        display.PanelLayout = PlotterGraphPaneEx.LayoutMode.TILES_HOR;
        display.DataSources[j].Length = 5800;
```

```
                        display.DataSources[j].AutoScaleY = true;
                        display.DataSources[j].SetDisplayRangeY(-300, 600);
                        display.DataSources[j].SetGridDistanceY(100);
                        CalcSinusFunction_2(display.DataSources[j], j);
                        break;
                    case "ANIMATED_AUTO":
                        this.Text = "Animated graphs fixed x range";
                        display.PanelLayout = PlotterGraphPaneEx.LayoutMode.TILES_HOR;
                        display.DataSources[j].Length = 402;
                        display.DataSources[j].AutoScaleY = false;
                        display.DataSources[j].AutoScaleX = true;
                        display.DataSources[j].SetDisplayRangeY(-300, 500);
                        display.DataSources[j].SetGridDistanceY(100);
                        display.DataSources[j].XAutoScaleOffset = 50;
                        CalcSinusFunction_3(display.DataSources[j], j, 0);
                        display.DataSources[j].OnRenderYAxisLabel = RenderYLabel;
                        break;
                }
            }
            ApplyColorSchema();
            this.ResumeLayout();
            display.Refresh();
        }

       ////////////////////////////////////////////////////////////////////////////////

        #region Funcoes para calcular graficos//17/12/11

////////////////////////////////////////////////////////////////////////////////////////////
///
        //13/2/12 desenvolvida para pegar dados da tabela e colocar no gráfico
        protected void tabelaFunction(DataSource src, int idx) {
            ///loop para carregar todos os dados da tebela
            for (int i = 0; i < src.Length; i++)
            {
                src.Samples[i].x = i;//carrega eixo x
                int dadosy = 0;//inicia variavel
                dadosy = 15;//teste
                string dadosy1="15";
                string dadosteste = "0";
                if (radioButton3.Checked)//para chamar dados co2
                {
                    double casa;
                    casa = Convert.ToDouble( dataGridView6.Rows[0].Cells[4].Value);
                    if (casa > 0 || casa < 0)//entra se tiver coeficiente linear>0
                    {
                        //fazendo calculo da calibracao
                        double home,haus;
                        haus= Convert.ToDouble(dataGridView6.Rows[0].Cells[1].Value);//valor
do b da equacao linear
                        casa =
Convert.ToDouble(dataGridView6.Rows[0].Cells[4].Value);//coeficiente linear
                        if (i < src.Length - 1)
                            dadosteste = dataGridView1.Rows[i].Cells[1].Value.ToString();
//funcionou depois de tanto sacrificio..
                        //casa =Convert.ToDouble(
dataGridView1.Rows[i].Cells[1].Value.ToString());
                        //home=(dadosteste*casa) + haus;
                        //home=casa*
                        dadosy = Convert.ToInt16(dadosteste);
                            home = (Convert.ToDouble(dadosy) * casa) + haus;
                        //home = casa*
                        //dadosy = Convert.ToInt16(dadosteste);//funcionando
                        dadosy = Convert.ToInt16(home);//funcionando
                        src.Samples[i].y = dadosy;//devo buscar dados da tabela;-carrego dados
de y
                    }
                    else
                    {
                        if (i < src.Length - 1)
                            dadosteste = dataGridView1.Rows[i].Cells[1].Value.ToString();
//funcionou depois de tanto sacrificio..
                        dadosy = Convert.ToInt16(dadosteste);//funcionando
                        src.Samples[i].y = dadosy;//devo buscar dados da tabela;-carrego dados
de y
                    }
                }
```

```
                if (radioButton4.Checked)//para chamar dados co2
                {
                    double casa;
                    casa = Convert.ToDouble(dataGridView6.Rows[0].Cells[4].Value);
                    if (casa > 0 || casa < 0)//entra se tiver coeficiente linear>0
                    {
                        //fazendo calculo da calibracao
                        double home, haus;
                        haus = Convert.ToDouble(dataGridView6.Rows[0].Cells[1].Value);//valor
do b da equacao linear
                        casa =
Convert.ToDouble(dataGridView6.Rows[0].Cells[4].Value);//coeficiente linear

                        if (i < src.Length - 1)
                            dadosteste = dataGridView1.Rows[i].Cells[1].Value.ToString();
//funcionou depois de tanto sacrificio..
                        //casa =Convert.ToDouble(
dataGridView1.Rows[i].Cells[1].Value.ToString());
                        //home=(dadosteste*casa) + haus;
                        //home=casa*
                        dadosy = Convert.ToInt16(dadosteste);
                        home = (Convert.ToDouble(dadosy) * casa) + haus;
                        //home = casa*
                        //dadosy = Convert.ToInt16(dadosteste);//funcionando
                        dadosy = Convert.ToInt16(home);//funcionando
                        src.Samples[i].y = dadosy;//devo buscar dados da tabela;-carrego dados
de y
                    }
                    else
                    {
                        if (i < src.Length - 1)
                            dadosteste = dataGridView1.Rows[i].Cells[1].Value.ToString();
//funcionou depois de tanto sacrificio..
                        dadosy = Convert.ToInt16(dadosteste);//funcionando
                        src.Samples[i].y = dadosy;//devo buscar dados da tabela;-carrego dados
de y
                    }
                }
            }
            src.OnRenderYAxisLabel = RenderYLabel;
            src.OnRenderXAxisLabel = RenderXLabel;//14/2/12
        }


        //17/12/11 desenvolvida para testar equacao da reta e aprendendo a usar objeto
        protected void retaFunction(DataSource src, int idx)
        {
            for (int i = 0; i < src.Length; i++)
            {
                src.Samples[i].x = i;
                src.Samples[i].y = (float)0.5 * i;
            }
            src.OnRenderYAxisLabel = RenderYLabel;
        }

        //29/9/11
        protected void CalcSinusFunction_0(DataSource src, int idx)
        {
            for (int i = 0; i < src.Length; i++)
            {
                src.Samples[i].x = i;
                src.Samples[i].y = (float)(((float)200 * Math.Sin((idx + 1) * (i + 1.0) * 48 /
src.Length)));
            }
            src.OnRenderYAxisLabel = RenderYLabel;
        }

        protected void CalcSinusFunction_1(DataSource src, int idx)
        {
            for (int i = 0; i < src.Length; i++)
            {
                src.Samples[i].x = i;

                src.Samples[i].y = (float)(((float)20 *
                                           Math.Sin(20 * (idx + 1) * (i + 1) * 3.141592 /
src.Length)) *
```

```
                                            Math.Sin(40 * (idx + 1) * (i + 1) * 3.141592 /
src.Length)) +
                                            (float)(((float)200 *
                                            Math.Sin(200 * (idx + 1) * (i + 1) * 3.141592 /
src.Length)));
            }
            src.OnRenderYAxisLabel = RenderYLabel;
        }

        protected void CalcSinusFunction_2(DataSource src, int idx)
        {
            for (int i = 0; i < src.Length; i++)
            {
                src.Samples[i].x = i;

                src.Samples[i].y = (float)(((float)20 *
                                            Math.Sin(40 * (idx + 1) * (i + 1) * 3.141592 /
src.Length)) *
                                            Math.Sin(160 * (idx + 1) * (i + 1) * 3.141592 /
src.Length)) +
                                            (float)(((float)200 *
                                            Math.Sin(4 * (idx + 1) * (i + 1) * 3.141592 /
src.Length)));
            }
            src.OnRenderYAxisLabel = RenderYLabel;
        }

        protected void CalcSinusFunction_3(DataSource ds, int idx, float time)
        {
            cPoint[] src = ds.Samples;
            for (int i = 0; i < src.Length; i++)
            {
                src[i].x = i;
                src[i].y = 200 + (float)((200 * Math.Sin((idx + 1) * (time + i * 100) /
8000.0))) +
                                +(float)((40 * Math.Sin((idx + 1) * (time + i * 200) /
2000.0)));
                /**
                            (float)( 4* Math.Sin( ((time + (i+8) * 100) / 900.0)))+
                            (float)(28 * Math.Sin(((time + (i + 8) * 100) / 290.0))); */
            }

        }
        #endregion

////////////////////////////////////////////////////////////////////////////////////////
//////
        //29/9/11
        private String RenderXLabel(DataSource s, int idx)
        {
            if (s.AutoScaleX)
            {
                //if (idx % 2 == 0)
                {
                    int Value = (int)(s.Samples[idx].x);
                    return "" + Value;
                }
                return "";
            }
            else
            {
                //int Value = (int)(s.Samples[idx].x / 200);//14/2/12
                int Value = (int)(s.Samples[idx].x);//14/2/12 funcionou viva...
                String Label = "" + Value + "\"";
                return Label;
            }
        }

        private String RenderYLabel(DataSource s, float value)
        {
            return String.Format("{0:0.0}", value);
        }

        private void ApplyColorSchema()
        {
            switch (CurColorSchema)
            {
```

```
case "DARK_GREEN":
    {
        Color[] cols = { Color.FromArgb(0,255,0),
                         Color.FromArgb(0,255,0),
                         Color.FromArgb(0,255,0),
                         Color.FromArgb(0,255,0),
                         Color.FromArgb(0,255,0) ,
                         Color.FromArgb(0,255,0),
                         Color.FromArgb(0,255,0) };

        for (int j = 0; j < NumGraphs; j++)
        {
            display.DataSources[j].GraphColor = cols[j % 7];
        }

        display.BackgroundColorTop = Color.FromArgb(0, 64, 0);
        display.BackgroundColorBot = Color.FromArgb(0, 64, 0);
        display.SolidGridColor = Color.FromArgb(0, 128, 0);
        display.DashedGridColor = Color.FromArgb(0, 128, 0);
    }
    break;
case "WHITE":
    {
        Color[] cols = { Color.DarkRed,
                         Color.DarkSlateGray,
                         Color.DarkCyan,
                         Color.DarkGreen,
                         Color.DarkBlue ,
                         Color.DarkMagenta,
                         Color.DeepPink };

        for (int j = 0; j < NumGraphs; j++)
        {
            display.DataSources[j].GraphColor = cols[j % 7];
        }

        display.BackgroundColorTop = Color.White;
        display.BackgroundColorBot = Color.White;
        display.SolidGridColor = Color.LightGray;
        display.DashedGridColor = Color.LightGray;
    }
    break;

case "BLUE":
    {
        Color[] cols = { Color.Red,
                         Color.Orange,
                         Color.Yellow,
                         Color.LightGreen,
                         Color.Blue ,
                         Color.DarkSalmon,
                         Color.LightPink };

        for (int j = 0; j < NumGraphs; j++)
        {
            display.DataSources[j].GraphColor = cols[j % 7];
        }

        display.BackgroundColorTop = Color.Navy;
        display.BackgroundColorBot = Color.FromArgb(0, 0, 64);
        display.SolidGridColor = Color.Blue;
        display.DashedGridColor = Color.Blue;
    }
    break;

case "GRAY":
    {
        Color[] cols = { Color.DarkRed,
                         Color.DarkSlateGray,
                         Color.DarkCyan,
                         Color.DarkGreen,
                         Color.DarkBlue ,
                         Color.DarkMagenta,
                         Color.DeepPink };

        for (int j = 0; j < NumGraphs; j++)
        {
```

```
                    display.DataSources[j].GraphColor = cols[j % 7];
                }

                display.BackgroundColorTop = Color.White;
                display.BackgroundColorBot = Color.LightGray;
                display.SolidGridColor = Color.LightGray;
                display.DashedGridColor = Color.LightGray;
            }
            break;

        case "RED":
            {
                Color[] cols = { Color.DarkCyan,
                                 Color.Yellow,
                                 Color.DarkCyan,
                                 Color.DarkGreen,
                                 Color.DarkBlue ,
                                 Color.DarkMagenta,
                                 Color.DeepPink };

                for (int j = 0; j < NumGraphs; j++)
                {
                    display.DataSources[j].GraphColor = cols[j % 7];
                }

                display.BackgroundColorTop = Color.DarkRed;
                display.BackgroundColorBot = Color.Black;
                display.SolidGridColor = Color.Red;
                display.DashedGridColor = Color.Red;
            }
            break;

        case "LIGHT_BLUE":
            {
                Color[] cols = { Color.DarkRed,
                                 Color.DarkSlateGray,
                                 Color.DarkCyan,
                                 Color.DarkGreen,
                                 Color.DarkBlue ,
                                 Color.DarkMagenta,
                                 Color.DeepPink };

                for (int j = 0; j < NumGraphs; j++)
                {
                    display.DataSources[j].GraphColor = cols[j % 7];
                }

                display.BackgroundColorTop = Color.White;
                display.BackgroundColorBot = Color.FromArgb(183, 183, 255);
                display.SolidGridColor = Color.Blue;
                display.DashedGridColor = Color.Blue;
            }
            break;

        case "BLACK":
            {
                Color[] cols = { Color.FromArgb(255,0,0),
                                 Color.FromArgb(0,255,0),
                                 Color.FromArgb(255,255,0),
                                 Color.FromArgb(64,64,255),
                                 Color.FromArgb(0,255,255) ,
                                 Color.FromArgb(255,0,255),
                                 Color.FromArgb(255,128,0) };

                for (int j = 0; j < NumGraphs; j++)
                {
                    display.DataSources[j].GraphColor = cols[j % 7];
                }

                display.BackgroundColorTop = Color.Black;
                display.BackgroundColorBot = Color.Black;
                display.SolidGridColor = Color.DarkGray;
                display.DashedGridColor = Color.DarkGray;
            }
            break;
    }
```

```
        }

        private void OnTimerTick(object sender, EventArgs e)
        {
            //função para ser chamada a cada interrupcao do timer
            if (CurExample == "ANIMATED_AUTO")
            {
                try
                {
                    TimeSpan dt = DateTime.Now - lastTimerTick;

                    for (int j = 0; j < NumGraphs; j++)
                    {

                        CalcSinusFunction_3(display.DataSources[j], j,
(float)dt.TotalMilliseconds);

                    }

                    this.Invoke(new MethodInvoker(RefreshGraph));
                }
                catch (ObjectDisposedException ex)
                {
                    // we get this on closing of form
                }
                catch (Exception ex)
                {
                    Console.Write("exception invoking refreshgraph(): " + ex.Message);
                }
            }
        }
        private void RefreshGraph()
        {
            display.Refresh();
        }
        #endregion
        #region comunicacao serial
        #region botao abre serial
        private void button7_Click(object sender, EventArgs e)
        {
            button7.Enabled = false;
            string porta;
            porta=textBox18.Text;
            int Port = Convert.ToInt16(porta);
            serialPort1.PortName ="COM"+ Port;
            serialPort1.Open();//abre porta serial
        }
        #endregion
        #region botao fecha serial
        private void button8_Click(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
            {
                serialPort1.Close();
                button7.Enabled = true;
            }
            else MessageBox.Show("A Porta esta fechada");
        }
        #endregion
        #region botao envia comando
        private void button9_Click(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
                {
                //serialPort1.Write("11");//envia em asci funcionou
                string comando;
                comando = textBox19.Text;
                serialPort1.Write(comando);
                }
            else {
                MessageBox.Show("A Porta serial está fechada, favor abrir");
            }
        }
        #endregion
        #region Botões de comandos manuais
        private void button25_Click(object sender, EventArgs e)
        {
```

```
        if (serialPort1.IsOpen)
        {
            //serialPort1.Write("11");//envia em asci funcionou
            string comando;
            comando = "0";
            serialPort1.Write(comando);
        }
        else
        {
            MessageBox.Show("A Porta serial está fechada, favor abrir");
        }
    }

    private void button30_Click(object sender, EventArgs e)
    {
        if (serialPort1.IsOpen)
        {
            //serialPort1.Write("11");//envia em asci funcionou
            string comando;
            comando = "1";
            serialPort1.Write(comando);
        }
        else
        {
            MessageBox.Show("A Porta serial está fechada, favor abrir");
        }
    }

    private void button26_Click(object sender, EventArgs e)
    {
        if (serialPort1.IsOpen)
        {
            //serialPort1.Write("11");//envia em asci funcionou
            string comando;
            comando = "2";
            serialPort1.Write(comando);
        }
        else
        {
            MessageBox.Show("A Porta serial está fechada, favor abrir");
        }
    }

    private void button31_Click(object sender, EventArgs e)
    {
        if (serialPort1.IsOpen)
        {
            //serialPort1.Write("11");//envia em asci funcionou
            string comando;
            comando = "3";
            serialPort1.Write(comando);
        }
        else
        {
            MessageBox.Show("A Porta serial está fechada, favor abrir");
        }
    }

    private void button27_Click(object sender, EventArgs e)
    {
        if (serialPort1.IsOpen)
        {
            //serialPort1.Write("11");//envia em asci funcionou
            string comando;
            comando = "4";
            serialPort1.Write(comando);
        }
        else
        {
            MessageBox.Show("A Porta serial está fechada, favor abrir");
        }
    }

    private void button32_Click(object sender, EventArgs e)
    {
        if (serialPort1.IsOpen)
        {
```

```
                //serialPort1.Write("11");//envia em asci funcionou
                string comando;
                comando = "8";
                serialPort1.Write(comando);
            }
            else
            {
                MessageBox.Show("A Porta serial está fechada, favor abrir");
            }
        }

        private void button28_Click(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
            {
                //serialPort1.Write("11");//envia em asci funcionou
                string comando;
                comando = "5";
                serialPort1.Write(comando);
            }
            else
            {
                MessageBox.Show("A Porta serial está fechada, favor abrir");
            }
        }

        private void button2_Click_1(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
            {
                //serialPort1.Write("11");//envia em asci funcionou
                string comando;
                comando = "6";
                serialPort1.Write(comando);
            }
            else
            {
                MessageBox.Show("A Porta serial está fechada, favor abrir");
            }
        }

        private void button29_Click(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
            {
                //serialPort1.Write("11");//envia em asci funcionou
                string comando;
                comando = "7";
                serialPort1.Write(comando);
            }
            else
            {
                MessageBox.Show("A Porta serial está fechada, favor abrir");
            }
        }
        #endregion
        #region interrupcao de recebimento serial
        private void serialPort1_DataReceived(object sender,
System.IO.Ports.SerialDataReceivedEventArgs e)
        {
            if (!receive)//usado somente em tela de comunicacao para testar comandos
            {
                //usado somente no form de comunicacao
                byte[] dados = new byte[1]; // declare numbers as an int array of any size
                serialPort1.Read(dados, 0, 1);
                MessageBox.Show(Convert.ToString(dados[0]));//recebo em asci2 se envio 1
recebo 49
            }
        }
        #endregion
        #endregion
        #region Aquisicao
        #region botao para iniciar recepcao
        private void button10_Click(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
            {
```

```
                MessageBox.Show("Não se esqueça de ligar o Filamento do sensor de CO2 para
obter leituras corretas\n\rFavor esperar 5 minutos");
                button10.Enabled = false;
                button11.Enabled = true;
                button12.Enabled = false;
                receive = true;//laco de controle para nao mexer no evento de recepcao
                aquisition = true;//habilita enviar na interrumcao do timer 1
                start_reading();//starta thread so deve ser startado uma unica vez senao dá a
merda toda de ler errado
                timer1.Start();//sera colocado na interrupcao para enviar comandos
automaticamente
            }
            else
            {
                MessageBox.Show("favor abrir porta serial");
                receive = false;
                aquisition = false;
            }
        }
        #endregion
        #region botao para parar recepcao
        private void button11_Click(object sender, EventArgs e)
        {
            limpa_buffer_rx();//25/2/12 foi necessario colocar aqui pois se parar de ler no
meio do buffer comeca a colocar errado informacoes no grid
            count_rx = 0;//25/2/12
            ///////falta ainda resetar coluna
            ////////////////////////////////////////////////////////////////////////
            timer1.Stop();
            receive = false;//variavel de Controle laco evento recepcao
            aquisition = false;//variavel para controlar envio de comandos
            for (int i = 0; i < 100000; i++) ;//26-2-12   delay para acabar de ler buffer
serial
            stop_reading();//mata treading de ler serial.
            button10.Enabled = true;
            button11.Enabled = false;
            button12.Enabled = true;
            MessageBox.Show("Não esqueça de desligar filamento após as leituras de CO2");
        }
        #endregion
        #region botao apagar
        private void button12_Click(object sender, EventArgs e)
        {
            timer1.Stop();
            receive = false;//variavel de Controlelaco recepcao
            aquisition = false;
            stop_reading();//mata treading de ler serial.
            dataGridView2.Rows.Clear();//limpa data grid
            dataGridView3.Rows.Clear();//limpa data grid
            textBox21.Text = "";
            textBox20.Text = "";
            ////////////////////estatistica CO2
            label37.Text = "";// = total_ad_co2;//variavel estatistica minimo valor
            label38.Text = "";// = total_ad_co2;//variavel estatistica med valor
            label39.Text = "";// = total_ad_co2;//variavel estatistica maximo valor
            ////////////////estatistica TEMPERATURA
            label46.Text = "";// = total_ad_co2;//variavel estatistica minimo valor
            label47.Text = "";// = total_ad_co2;//variavel estatistica med valor
            label48.Text = "";// = total_ad_co2;//variavel estatistica maximo valor
            total_media_co2 = 0;
            total_media_temp = 0;
            min_co2 = 0;
            max_co2 = 0;
            min_temp = 0;
            max_temp = 0;
            rsd_temp_percent = 0;
            rsd_temp = 0;
            rsd_co2 = 0;
            rsd_co2_percent = 0;
            flag_min_co2 = false;//para calculo estatistico
            total_ad_co2 = 0;//para calculos estatisticos precisa para nao dar erro de ir
incrementando
            flag_max_co2 = false;//
            ////////////////TEMPERATURA
            flag_min_temp = false;//para calculo estatistico
            total_ad_temp = 0;//para calculos estatisticos precisa para nao dar erro de ir
incrementando
```

```
            flag_max_temp = false;//
        }
        #endregion
        #region interrupcao timer1
        public void timer1_Tick(object sender, EventArgs e)
        {
            if (aquisition)
            {
                ////////////////LOOP back
                //comentado
                #region tudo funcionando na interrupcao do timer 1 por loopback envia tres
comandos
                ////////////////////////////////////////////////////////////////////////
                ////////envia comandos testes para testar data grid com loop back
                //string comando;
                //comando = "4";//valor a/d do sensor temperatura
                //serialPort1.Write(comando);
                //comando = "5";//valor a/d do sensor temperatura
                //serialPort1.Write(comando);
                //comando = "6";//valor a/d do sensor temperatura
                //serialPort1.Write(comando);
                #endregion

                //comentado
                #region tudo funcionando na interrupcao do timer 1 por loopback envia seis
comandos
                ////////////////////////////////////////////////////////////////////////
                ////////envia comandos testes para testar data grid com loop back
                //string comando;
                //comando = "4";//valor a/d do sensor temperatura
                //serialPort1.Write(comando);
                //comando = "5";//valor a/d do sensor temperatura
                //serialPort1.Write(comando);
                //comando = "6";//valor a/d do sensor temperatura
                //serialPort1.Write(comando);
                //comando = "8";//valor a/d do sensor temperatura
                //serialPort1.Write(comando);
                //comando = "5";//valor a/d do sensor temperatura
                //serialPort1.Write(comando);
                //comando = "6";//valor a/d do sensor temperatura
                //serialPort1.Write(comando);
                #endregion
                ///////////////PARA HARDWARE
                #region comando 4 para leitura sensor de co2 HARDWARE
                string comando;
                comando = "4";//valor a/d do sensor temperatura
                serialPort1.Write(comando);
                #endregion
                for (int i = 0; i < 1000000; i++);//dalay teste
                #region comando 8 para leitura sensor de co2 HARDWARE
                    //string comando;
                    comando = "8";//valor a/d do sensor temperatura
                    serialPort1.Write(comando);
                    #endregion
                for (int i = 0; i < 1000000; i++) ;//dalay teste 2
               aquisition = false;//25/2/12       deve estar no final?
            }
        }
        #endregion
        #region funcao limpa buffer recepcao
        public void limpa_buffer_rx()
        {
            for (int i = 0; i < Rx_dado.Length - 1; i++)
            {
                Rx_dado[i] = 0;
            }
        }
        #endregion
        #region Thread serial
        void read_serial()//EVENTO DE UMA THREAD
        {
            while (read)
            {
                if (!aquisition)
                {
                    if (serialPort1.IsOpen)//22/2/12  colocado pois dava erro quando fechava o
form
```

```
                    {
                        if (serialPort1.BytesToRead != 0)
                        {
                            int dado;
                            dado = serialPort1.ReadByte();
                            valor_serial = dado;
                            grid();//funcao delegate que coloca valores no data grid
                        }

                    }
                }
                System.Threading.Thread.Sleep(1);//obrigatorio senao come todo o processamento
tem que ser neste lugar senao estora memoria
            }

        }
        void start_reading()
        {
            //#region Treads
            Tr = new System.Threading.Thread(read_serial);//22/2/12
            Tr.Priority = ThreadPriority.Normal;
            read = true;//controle do laco da thread
            Tr.Start();//starta thread..

            //#endregion
        }
        void stop_reading()
        {
            read = false;
            Tr = null;

        }

        #region delegate aquisition

        private void grid() //
        {
            if (InvokeRequired)
            {
                Invoke((MethodInvoker)delegate
                {
                    datagrid_aquisition();
                    textbox_aquisition();
                });
            }
            else
            {
                datagrid_aquisition();
                textbox_aquisition();
            }
        }
        public void textbox_aquisition()
        {
            textBox20.Text = Convert.ToString(conv_ad + (256 * conv_adh));
            textBox21.Text = Convert.ToString(conv_ad_temp + (256 * conv_adh_temp));
            textBox20.Text = Convert.ToString(total_ad_co2);
            textBox21.Text = Convert.ToString(total_ad_temp);
            ////////////////estatistica CO2
            label37.Text = Convert.ToString( min_co2);// = total_ad_co2;//variavel estatistica
minimo valor
            label38.Text = Convert.ToString( med_co2);// = total_ad_co2;//variavel estatistica
med valor
            label39.Text= Convert.ToString( max_co2);// = total_ad_co2;//variavel estatistica
maximo valor
            ///////////////estatistica TEMPERATURA
            label46.Text = Convert.ToString(min_temp);// = total_ad_co2;//variavel estatistica
minimo valor
            label47.Text = Convert.ToString(med_temp);// = total_ad_co2;//variavel estatistica
med valor
            label48.Text = Convert.ToString(max_temp);// = total_ad_co2;//variavel estatistica
maximo valor
        }
        public void datagrid_aquisition()
        {
            int Row_contagem;
            int Row_contagem_temp;
            Row_contagem = dataGridView2.RowCount;
```

```
            Row_contagem_temp = dataGridView3.RowCount;
            Rx_dado[count_rx] = valor_serial;
            switch (count_rx++)
            {
                case 0:
                        ////////////////////////////////CO2
                        conv_ad = 0;//variavel para calcular valor total da conversao ad e
mostrar no data grid
                        conv_adh = 0;//variavel que tem a parte mais significativa do adresh
                        dataGridView2.Rows.Add(1);//adiciona linha no data grid
                        dataGridView2.Rows[Row_contagem - 1].Cells[0].Value =
Row_contagem;//TEMPO= valor do data grid
                        dataGridView2.Rows[Row_contagem - 1].Cells[1].Value = Rx_dado[count_rx
- 1];//COMANDO
                        //////////////teste 10-3-12
                        DateTime datum_banco;
                        string _datum, _day, _month, _year, _monta_datum;
                        _datum = DateTime.Now.Date.ToShortDateString();//DATA
                        datum_banco = Convert.ToDateTime(_datum);
                        _day = Convert.ToString(datum_banco.Day);
                        _month = Convert.ToString(datum_banco.Month);
                        _year = Convert.ToString(datum_banco.Year);
                        _monta_datum = _year + "-" + _month + "-" + _day;//montando para
salvar no banco tabela tempo
                        dataGridView2.Rows[Row_contagem - 1].Cells[5].Value =
_monta_datum;//DATA versao original
                        dataGridView2.Rows[Row_contagem - 1].Cells[6].Value =
DateTime.Now.TimeOfDay;//HORA
                    break;
                case 1:
                       /////////////////CO2
                        conv_ad = Rx_dado[count_rx - 1];//coloca valor menos significativo
                        dataGridView2.Rows[Row_contagem - 2].Cells[2].Value = Rx_dado[count_rx
- 1];//ADRESL
                    break;
                case 2:
                     ////////////////////////CO2
                        conv_adh = Rx_dado[count_rx - 1];
                        dataGridView2.Rows[Row_contagem - 2].Cells[3].Value = Rx_dado[count_rx
- 1];//ADRESH
                        //calcula conversao para colocar na coluna conversao a/d
                        dataGridView2.Rows[Row_contagem - 2].Cells[4].Value = conv_ad + (256 *
(Rx_dado[count_rx - 1]));//CONV. A/
                    break;

                case 3:
                    ////////////////////////////////TEMPERATURA
                    conv_ad_temp = 0;//variavel para calcular valor total da conversao ad e
mostrar no data grid
                    conv_adh_temp = 0;//variavel que tem a parte mais significativa do adresh
                    dataGridView3.Rows.Add(1);//adiciona linha no data grid
                    dataGridView3.Rows[Row_contagem_temp - 1].Cells[0].Value =
Row_contagem_temp;//TEMPO= valor do data grid
                    dataGridView3.Rows[Row_contagem_temp - 1].Cells[1].Value =
Rx_dado[count_rx - 1];//COMANDO
                    DateTime datum_banco_temp;
                    string _datum_temp, _day_temp, _month_temp, _year_temp, _monta_datum_temp;
                    _datum_temp = DateTime.Now.Date.ToShortDateString();//DATA
                    datum_banco_temp = Convert.ToDateTime(_datum_temp);
                    _day_temp = Convert.ToString(datum_banco_temp.Day);
                    _month_temp = Convert.ToString(datum_banco_temp.Month);
                    _year_temp = Convert.ToString(datum_banco_temp.Year);
                    _monta_datum_temp = _year_temp + "-" + _month_temp + "-" +
_day_temp;//montando para salvar no banco tabela tempo
                    dataGridView3.Rows[Row_contagem_temp - 1].Cells[5].Value =
_monta_datum_temp;//DATA versao original
                    ////////////////
                    dataGridView3.Rows[Row_contagem_temp - 1].Cells[6].Value =
DateTime.Now.TimeOfDay;//HORA
                    break;
                case 4:
                    //conv_ad_temp = Rx_dado[count_rx - 1];
                    ///////////////TEMPERATURA
                    conv_ad_temp = Rx_dado[count_rx - 1];//coloca valor menos significativo
                    dataGridView3.Rows[Row_contagem_temp - 2].Cells[2].Value =
Rx_dado[count_rx - 1];//ADRESL
                    break;
```

```
                case 5:
                    //BYTE6
                    ////////////////////TEMPERATURA
                    conv_adh_temp = Rx_dado[count_rx - 1];
                    dataGridView3.Rows[Row_contagem_temp - 2].Cells[3].Value =
Rx_dado[count_rx - 1];//ADRESH
                    //calcula conversao para colocar na coluna conversao a/d
                    dataGridView3.Rows[Row_contagem_temp - 2].Cells[4].Value = conv_ad_temp +
(256 * (Rx_dado[count_rx - 1]));//CONV. A/D
                    #region estatisticas
                    ////////////////estatisticas
                    total_ad_co2 = conv_ad + (256 * conv_adh);//CO2
                    total_ad_temp = conv_ad_temp + (256 * conv_adh_temp);//TEMP
                    ////////////////CO2
                    double coef_temp,coef_co2,b_temp,b_co2;//coeficientes e ofset
                    coef_co2 = Convert.ToDouble(dataGridView7.Rows[0].Cells[4].Value);
                    b_co2 = Convert.ToDouble(dataGridView7.Rows[0].Cells[1].Value);
                    if (coef_co2 > 0 || coef_co2 < 0 )
                    {
                        total_ad_co2=(total_ad_co2*coef_co2)+ b_co2 ;
                    }
                    /////////TEMPERATURA
                    coef_temp=Convert.ToDouble(dataGridView8.Rows[0].Cells[4].Value);
                    b_temp = Convert.ToDouble(dataGridView8.Rows[0].Cells[1].Value);
                    if (coef_temp > 0 || coef_temp < 0) //calcula coeficientes e offset
                    {
                        total_ad_temp = (total_ad_temp * coef_temp) + b_temp;
                    }
                    ////////////////////////////CO2
                    if ((min_co2 == 0) && (!flag_min_co2))
                    {
                        min_co2 = total_ad_co2;//variavel estatistica minimo valor
                        flag_min_co2 = true;
                    }
                    if (total_ad_co2 < min_co2)
                    {
                        min_co2 = total_ad_co2;//variavel estatistica minimo valor
                    }

                    ////////////////////////TEMPERATURA
                    if ((min_temp == 0) && (!flag_min_temp))
                    {
                        min_temp = total_ad_temp;//variavel estatistica minimo valor
                        flag_min_temp = true;
                    }
                    if (total_ad_temp < min_temp)
                    {
                        min_temp = total_ad_temp;//variavel estatistica minimo valor
                    }
                    ////////////////////////////////////////MEDIA/////////
                    ////////////////////////////////CO2
                    int media;
                    media = dataGridView2.RowCount - 1;
                    total_media_co2 += total_ad_co2;
                    if (total_media_co2 != 0) med_co2 = (total_media_co2 / media);
                    ////////////////////////TEMPERATURA
                    int media2;
                    media2 = dataGridView3.RowCount - 1;
                    total_media_temp += total_ad_temp;
                    if (total_media_temp != 0) med_temp = (total_media_temp / media2);
                    ////////////////////////////////////MAXIMO
                    ///////CO2
                    if ((max_co2 == 0) && (!flag_max_co2))
                    {
                        max_co2 = total_ad_co2;//variavel estatistica maximo valor
                        flag_max_co2 = true;
                    }
                    if (total_ad_co2 > max_co2)
                    {
                        max_co2 = total_ad_co2;//variavel estatistica minimo valor
                    }
                    ////////////////////TEMPERATURA
                    if ((max_temp == 0) && (!flag_max_temp))
                    {
                        max_temp = total_ad_temp;//variavel estatistica maximo valor
                        flag_max_temp = true;
                    }
```

```
                    if (total_ad_temp > max_temp)
                    {
                        max_temp = total_ad_temp;//variavel estatistica minimo valor
                    }

                    }
                    ////////////////////////////////////
                    ////////////////////////TEMPERATURE
                    if (total_ad_temp > med_temp)
                    {
                        rsd_temp += Math.Sqrt(total_ad_temp - med_temp);
                        if (rsd_temp > 0) variancia_temp = rsd_temp / dataGridView3.RowCount;
                        if (variancia_temp > 0) rsd_temp_percent = Math.Sqrt(variancia_temp);
                        rsd_temp_percent = rsd_temp_percent * 100;
                    }
                    else
                    {
                        if (total_ad_temp != med_temp)
                        {
                            rsd_temp += Math.Sqrt(med_temp - total_ad_temp);
                            if (rsd_temp > 0) variancia_temp = rsd_temp /
dataGridView3.RowCount;
                            if (variancia_temp > 0) rsd_temp_percent =
Math.Sqrt(variancia_temp);
                            rsd_temp_percent = rsd_temp_percent * 100;
                        }
                    }
                    #endregion
                    //deve sempre estar no ultinmo case
                    limpa_buffer_rx();
                    count_rx = 0;//laco controle case
                    aquisition = true;//habilita a enviar novamente
                    break;
                default:
                    MessageBox.Show("entrou no defaulf do switch recepcao thread");
                    //limpa_buffer_rx();
                    //count_rx = 0;
                    //talvez limpa buffer de recepcao rx_dado[]
                    break;
            }
        }
        #endregion
        #endregion
        #region salvar no banco aquisicao
        private void button13_Click(object sender, EventArgs e)
        {
            MessageBox.Show("SEJA PACIÊNTE - O tempo para salvar no banco depende do numero de
dados adquiridos");
            ////////////////////////////////////////////////////////////////////////
            string server, database, user, passwd, tabela_temp,tabela_co2;
            server = textBox1.Text;
            database = textBox2.Text;
            user = textBox3.Text;
            passwd = textBox4.Text;
            //tabela = textBox5.Text;
            //tabela = "tabela_co2";//26/2/12 so posso usar string nao consigo mais de 1000
            tabela_temp = "tabela_temperatura";//teste para mais de 1000 funcionou mas nao
mostra somente no dos
            tabela_co2 = "tabela_co2";//tabela dados co2
            string[,] co2 = new string[dataGridView2.RowCount + 100, dataGridView2.RowCount +
100];//tem que ser dinamico senao da crepe se chega a 1000
            string[,] temp = new string[dataGridView3.RowCount + 100, dataGridView3.RowCount +
100];//tem que ser dinamico senao da crepe se chega a 1000
            int countRow = dataGridView2.RowCount;
            int countCol = dataGridView2.ColumnCount;
            int countRow_temp = dataGridView3.RowCount;
            int countCol_temp = dataGridView3.ColumnCount;
            int j = 0;
            int k = 0;
            string data = "'"; //para uma linha funciona
            string data_temp = "'"; //para uma linha funciona
            string[] data_tudo = new string[countRow - 1];
            string[] data_tudo_temp = new string[countRow_temp - 1];
            ////////////////////////////////////////co2
            for (k = 0; k < (countRow-1); k++)
            {
                 for (j = 0; j < (countCol); j++)
```

```
                {
                    co2[j,k] = string.Format("" + dataGridView2[j, k].Value.ToString());
                     if( (j < (countCol)))//&&(k == 1))
                     {
                         data = data + co2[j, k] + "'";
                     }
                     if (j < (countCol-1))
                     {
                             //data = data + a[j, k] + "','";
                             data += ",'";
                     }
                }
                data_tudo[k] = data;
                data = "'";
            }
            ////////////////////TEMPERATURA
            //int l = 0;
            //int m = 0;

            for (k = 0; k < (countRow_temp - 1); k++)
            {
                for (j = 0; j < (countCol_temp); j++)
                {
                    temp[j, k] = string.Format("" + dataGridView3[j, k].Value.ToString());
                    if ((j < (countCol_temp)))//&&(k == 1))
                    {
                        data_temp = data_temp + temp[j, k] + "'";
                    }
                    if (j < (countCol_temp - 1))
                    {
                        //data = data + a[j, k] + "','";
                        data_temp += ",'";
                    }
                }
                data_tudo_temp[k] = data_temp;
                data_temp = "'";
            }
            ////////////////////////////////////////////////////////
            MySqlConnection conection = new MySqlConnection("Server=" + server + ";Database="
+ database + ";Uid=" + user + ";Pwd=" + passwd + "");//só pode variável string se numerico
devo usar variavel.tostring()
            try
            {
                if (conection.State == ConnectionState.Closed)
                    conection.Open();
                ///////////////grava tabela co2
                for (int i = 0; i < (countRow - 1); i++)
                {
                    MySqlCommand comando = new MySqlCommand("INSERT INTO " + tabela_co2 +
"(tempo,comando,ADRESL,ADRESH,CONV,DATA,HORA) VALUES (" + data_tudo[i] + ")",
conection);//funcionou para um unico dado uma unica linha
                    comando.ExecuteNonQuery();
                }
                ////////////grava tabela temperatura
                for (int i = 0; i < (countRow_temp - 1); i++)
                {
                    MySqlCommand comando = new MySqlCommand("INSERT INTO " + tabela_temp +
"(tempo,comando,ADRESL,ADRESH,CONV,DATA,HORA) VALUES (" + data_tudo_temp[i] + ")",
conection);//funcionou para um unico dado uma unica linha
                    comando.ExecuteNonQuery();
                }
                conection.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.StackTrace);
            }
        }
        #endregion
        #region botao liga e desliga filamento co2
        private void button14_Click(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
            {
                //serialPort1.Write("11");//envia em asci funcionou
                string comando;
                //comando = textBox19.Text;
```

```
                comando = "5";
                serialPort1.Write(comando);
            }
            else
            {
                MessageBox.Show("A Porta serial está fechada, favor abrir");
            }
        }
        private void button15_Click(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
            {
                //serialPort1.Write("11");//envia em asci funcionou
                string comando;
                //comando = textBox19.Text;
                comando = "6";
                serialPort1.Write(comando);
            }
            else
            {
                MessageBox.Show("A Porta serial está fechada, favor abrir");
            }
        }
        #endregion


        #region Botao salva estatisticas
        private void button24_Click(object sender, EventArgs e)
        {
            string server, database, user, passwd, tabela_temp, tabela_co2;
            server = textBox1.Text;
            database = textBox2.Text;
            user = textBox3.Text;
            passwd = textBox4.Text;

            tabela_co2 = "tabela_estatistica";//tabela dados estatisticos
            string min_co2, med_co2, max_co2, min_temp, med_temp, max_temp, datum, zeit;
            min_co2 = Convert.ToString(label37.Text).Replace(',','.');
            //ToString().Replace(',', '.')
            med_co2 = Convert.ToString(label38.Text).Replace(',', '.');
            max_co2 = Convert.ToString(label39.Text).Replace(',', '.');
            min_temp = Convert.ToString(label46.Text).Replace(',', '.');
            med_temp = Convert.ToString(label47.Text).Replace(',', '.');
            max_temp = Convert.ToString(label48.Text).Replace(',', '.');
            DateTime datum_banco;
            string _datum, _day, _month, _year, _monta_datum;
            _datum = DateTime.Now.Date.ToShortDateString();//DATA
            datum_banco = Convert.ToDateTime(_datum);
            _day = Convert.ToString(datum_banco.Day);
            _month = Convert.ToString(datum_banco.Month);
            _year = Convert.ToString(datum_banco.Year);
            _monta_datum = _year + "-" + _month + "-" + _day;//montando para salvar no banco
tabela tempo
            datum = Convert.ToString(_monta_datum);
            zeit = Convert.ToString(DateTime.Now.TimeOfDay);//teste
            MySqlConnection conection = new MySqlConnection("Server=" + server + ";Database="
+ database + ";Uid=" + user + ";Pwd=" + passwd + "");//só pode variável string se numerico
devo usar variavel.tostring()
            try
            {
                if (conection.State == ConnectionState.Closed)
                    conection.Open();
                MySqlCommand comando = new MySqlCommand("INSERT INTO " + tabela_co2 + "
(min_co2,med_co2,max_co2,min_temp,med_temp,max_temp,DATA,HORA) VALUES (" + min_co2 + "," +
med_co2 + "," + max_co2 + "," + min_temp + "," + med_temp + "," + max_temp + ",'" + datum +
"','" + zeit + "')", conection);
                comando.ExecuteNonQuery();
                conection.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.StackTrace);
            }
        }
        #endregion
        #endregion
        #region calibracao
```

```
        #region botao iniciar aquisicao calibracao
        private void button19_Click(object sender, EventArgs e)
        {
            if (serialPort1.IsOpen)
            {
                MessageBox.Show("Não se esqueça de ligar o Filamento do sensor de CO2 para
obter leituras corretas\n\rFavor esperar 5 minutos");
                button19.Enabled = false;//inicia aquisicao na calibracao
                button20.Enabled = true;
                receive = true;//laco de controle para nao mexer no evento de recepcao
                aquisition_cal = true;//habilita enviar na interrumcao do timer 1
                start_reading_cal();//starta thread so deve ser startado uma unica vez senao
dá a merda toda de ler errado
                timer2.Start();//sera colocado na interrupcao para enviar comandos
automaticamente
            }
            else
            {
                MessageBox.Show("favor abrir porta serial");
                receive= false;
                aquisition_cal = false;
            }
        }
        #endregion
        #region botao parar aquisicao calibracao
        private void button20_Click(object sender, EventArgs e)
        {
            ////////quando taxa de aquisicao ta muito alta
            limpa_buffer_rx_cal();//25/2/12 foi necessario colocar aqui pois se parar de ler
no meio do buffer comeca a colocar errado informacoes no grid
            count_rx_cal = 0;//25/2/12
            ///////falta ainda resetar coluna
            ////////////////////////////////////////////////////////////////////////
            timer2.Stop();
            receive = false;//variavel de Controle laco evento recepcao
            aquisition_cal = false;//variavel para controlar envio de comandos
            for (int i = 0; i < 100000; i++) ;//26-2-12   delay idiota para acabar de ler
buffer seria
            stop_reading_cal();//mata treading de ler serial.
            timer2.Stop();//desabilita interrupcao timer
            button19.Enabled = true;//habilita aquisicao calibracao
            button20.Enabled = false;
            button21.Enabled = true;
        }
        #endregion
        #region interrupcao timer 2 calibracao
        private void timer2_Tick(object sender, EventArgs e)
        {
            if (aquisition_cal)//flag para nao enviar enquanto processando os bytes recebidos
estao sendo processados
            {
                if (radioButton6.Checked)//se habilitado pede o valor do sensor de co2
                {
                    #region comando 4 para leitura sensor de co2 HARDWARE
                    string comando;
                    comando = "4";//valor a/d do sensor temperatura
                    serialPort1.Write(comando);
                    #endregion

                    #region delay besta para erros no data grid
                    for (int i = 0; i < 1000000; i++) ;//dalay teste entre comandos
                    #endregion
                }
                else if (radioButton7.Checked)//se esta selecionado enviar para leitura de
temperatura
                {
                    #region comando 8 para leitura sensor de co2 HARDWARE
                    string comando;
                    comando = "8";//valor a/d do sensor temperatura
                    serialPort1.Write(comando);
                    #endregion

                    #region delay besta para erros no data grid
                    for (int i = 0; i < 1000000; i++) ;//dalay teste entre comandos
                    #endregion
                }
```

```
                aquisition_cal = false;//25/2/12       deve estar no final? flag para thread
nao processar quando estiver enviando
            }
        }
        #endregion
        #region thread serial calibracao
        void read_serial_cal()//EVENTO DE UMA THREAD
        {
            while (read_cal)
            {
                if (!aquisition_cal)
                {
                    if (serialPort1.IsOpen)//22/2/12  colocado pois dava erro quando fechava o
form
                    {
                        if (serialPort1.BytesToRead != 0)
                        {
                            int dado;
                            dado = serialPort1.ReadByte();
                            valor_serial_cal = dado;
                            grid_cal();//funcao delegate que coloca valores no data grid
                        }

                    }
                }
                System.Threading.Thread.Sleep(1);//obrigatorio senao come todo o processamento
tem que ser neste lugar senao estora memoria
            }
        }
        void start_reading_cal()//starta thread so deve ser startado uma unica vez senao dá a
merda toda de ler errado
        {
            //#region Treads
            Tr1 = new System.Threading.Thread(read_serial_cal);//22/2/12
            Tr1.Priority = ThreadPriority.Normal;
            read_cal = true;//controle do laco da thread
            Tr1.Start();//starta thread..

            //#endregion
        }
        void stop_reading_cal()
        {
            read_cal = false;
            Tr1 = null;
        }
        #region delegate e evento serial thread TR1 da calibracao
        private void grid_cal() //
        {
            if (InvokeRequired)
            {
                Invoke((MethodInvoker)delegate
                {
                    datagrid_aquisition_cal();
                    textbox_aquisition_cal();
                });
            }
            else
            {
                datagrid_aquisition_cal();
                textbox_aquisition_cal();
            }


        }
        public void textbox_aquisition_cal()
        {
             }
        public void datagrid_aquisition_cal()
        {
            int Row_contagem_cal;
            int Row_contagem_temp_cal;
            Row_contagem_cal = dataGridView4.RowCount;
            Row_contagem_temp_cal = dataGridView5.RowCount;
            Rx_dado_cal[count_rx_cal] = valor_serial_cal;
            switch (count_rx_cal++)
            {
                case 0:
```

```
                    ////////////////////////////////CO2
                    conv_ad = 0;//variavel para calcular valor total da conversao ad e mostrar
no data grid
                    conv_adh = 0;//variavel que tem a parte mais significativa do adresh
                    DateTime datum_banco_cal;
                    string _datum_cal, _day_cal, _month_cal, _year_cal, _monta_datum_cal;
                    _datum_cal = DateTime.Now.Date.ToShortDateString();//DATA
                    datum_banco_cal = Convert.ToDateTime(_datum_cal);
                    _day_cal = Convert.ToString(datum_banco_cal.Day);
                    _month_cal = Convert.ToString(datum_banco_cal.Month);
                    _year_cal = Convert.ToString(datum_banco_cal.Year);
                    _monta_datum_cal = _year_cal + "-" + _month_cal + "-" +
_day_cal;//montando para salvar no banco tabela tempo
                    dataGridView4.Rows[Row_contagem_cal - 1].Cells[6].Value =
_monta_datum_cal;//DATA
                    ////////////////////////////////////////////////////
                    dataGridView4.Rows[Row_contagem_cal - 1].Cells[7].Value =
DateTime.Now.TimeOfDay;//HORA

                    ////////////////////////////////TEMPERATURA
                    conv_ad_temp = 0;//variavel para calcular valor total da conversao ad e
mostrar no data grid
                    conv_adh_temp = 0;//variavel que tem a parte mais significativa do adresh
                    dataGridView5.Rows[Row_contagem_temp_cal - 1].Cells[6].Value
=_monta_datum_cal;//DATA
                    dataGridView5.Rows[Row_contagem_temp_cal - 1].Cells[7].Value =
DateTime.Now.TimeOfDay;//HORA

                    break;
                case 1:
                    /////////////////CO2
                    conv_ad = Rx_dado_cal[count_rx_cal - 1];//coloca valor menos significativo
                    /////////////////TEMPERATURA
                    conv_ad_temp = Rx_dado_cal[count_rx_cal - 1];//coloca valor menos
significativo
                    break;
                case 2:
                    ////////////////////////CO2
                    conv_adh = Rx_dado_cal[count_rx_cal - 1];

                    ////////////////////////////////////////CO2
                    if (radioButton6.Checked)
                    {
                        dataGridView4.Rows[Row_contagem_cal - 1].Cells[0].Value = conv_ad +
(256 * (Rx_dado_cal[count_rx_cal - 1]));//CONV. A/D
                        double vaca;
                        vaca = Convert.ToDouble(dataGridView4.Rows[Row_contagem_cal -
1].Cells[5].Value);//carrega coeficiente se o mesmo foi calculado
                        if (vaca > 0 || vaca < 0)
                        {
                            ////////////////////////para colocar valor calibrado
                            double a, b, y;
                            a = Convert.ToDouble(dataGridView4.Rows[0].Cells[5].Value);
                            b = Convert.ToDouble(dataGridView4.Rows[0].Cells[2].Value);
                            y = (Convert.ToDouble(dataGridView4.Rows[0].Cells[0].Value) * a) +
b;//so funciona quando tudo e zero
                            dataGridView4.Rows[0].Cells[1].Value = Convert.ToString(y);
                        }
                    }

                    ////////////////////TEMPERATURA
                    conv_adh_temp = Rx_dado_cal[count_rx_cal - 1];

                    ////////////////////////////////////TEMPERATURA
                    if (radioButton7.Checked)
                    {
                        dataGridView5.Rows[Row_contagem_temp_cal - 1].Cells[0].Value =
conv_ad_temp + (256 * (Rx_dado_cal[count_rx_cal - 1]));//CONV. A/D
                        double boi;
                        boi = Convert.ToDouble(dataGridView5.Rows[Row_contagem_cal -
1].Cells[5].Value);
                        if (boi > 0 || boi < 0)
                        {
                            ////////////////////////para colocar valor calibrado
                            double a, b, y;
                            a = Convert.ToDouble(dataGridView5.Rows[0].Cells[5].Value);
                            b = Convert.ToDouble(dataGridView5.Rows[0].Cells[2].Value);
```

```
                            y = (Convert.ToDouble(dataGridView5.Rows[0].Cells[0].Value) * a) +
b;//so funciona quando tudo e zero
                            dataGridView5.Rows[0].Cells[1].Value = Convert.ToString(y);
                        }
                    }

                    ////////////////////////OBRIGATORIO deve sempre estar no ultinmo case

////////////////////////////////////////////////////////////////////////
                    //limpa buffer recepcao da calibracao
                    limpa_buffer_rx_cal();
                    count_rx_cal = 0;//laco controle case
                    aquisition_cal = true;//habilita a enviar novamente
                    break;
                default:
                    MessageBox.Show("entrou no defaulf do switch recepcao thread nao deve
entrar aqui");
                    break;
            }
        }
        #endregion
        #endregion
        #region funcao limpa buffer recepcao calibracao
        public void limpa_buffer_rx_cal()
        {
            for (int i = 0; i < Rx_dado_cal.Length - 1; i++)
            {
                Rx_dado_cal[i] = 0;
            }
        }
        #endregion
        #region botao de zero valor offset
        private void button16_Click(object sender, EventArgs e)
        {
            if (radioButton6.Checked)
            {
                string zero;
                zero = Convert.ToString(textBox22.Text);// valor do b na equacao da reta
                dataGridView4.Rows[0].Cells[2].Value = zero;//CONV. A/
                //dataGridView4.Rows[0].Cells[1].Value =
dataGridView4.Rows[0].Cells[0].Value;//coloca o valor da conversao a/d no zero
            }
            if (radioButton7.Checked)
            {
                string zero;
                zero = Convert.ToString(textBox22.Text);// valor do b na equacao da reta
                dataGridView5.Rows[0].Cells[2].Value = zero;//CONV. A/
                //dataGridView5.Rows[0].Cells[1].Value =
dataGridView5.Rows[0].Cells[0].Value;//coloca o valor da conversao a/d no zero
            }
        }
        #endregion
        #region botao para gravar valor do padrão
        private void button17_Click(object sender, EventArgs e)
        {
            if (radioButton6.Checked)
            {

                string padrao;
                padrao = Convert.ToString(textBox22.Text);// valor do b na equacao da reta
                dataGridView4.Rows[0].Cells[4].Value = padrao;//CONV. A/
                dataGridView4.Rows[0].Cells[3].Value =
dataGridView4.Rows[0].Cells[0].Value;//coloca o valor da conversao a/d no zero
            }
            if (radioButton7.Checked)
            {
                string padrao;
                padrao = Convert.ToString(textBox22.Text);// valor do b na equacao da reta
                dataGridView5.Rows[0].Cells[4].Value = padrao;//CONV. A/
                dataGridView5.Rows[0].Cells[3].Value =
dataGridView5.Rows[0].Cells[0].Value;//coloca o valor da conversao a/d no zero
            }
        }
        #endregion
        #region botao para calcular coeficiente linear
        private void button21_Click(object sender, EventArgs e)
        {
```

```
            if (radioButton6.Checked)//co2
            {
                ////////////////////calcula co2
                double x,y,a,b,b1,a1,v_ad_final,v_ad_inicial,temp_final,temp_inicial;
                v_ad_inicial = 0;// Convert.ToDouble(dataGridView4.Rows[0].Cells[1].Value);
                v_ad_final = Convert.ToDouble(dataGridView4.Rows[0].Cells[3].Value);
                temp_inicial = Convert.ToDouble(dataGridView4.Rows[0].Cells[2].Value);
                temp_final = Convert.ToDouble(dataGridView4.Rows[0].Cells[4].Value);
                //a = - (temp_final - temp_inicial)/ (v_ad_final-v_ad_inicial);//7/4/12
                a = (temp_final - temp_inicial) / (v_ad_final - v_ad_inicial);//original
                //y = a*x + b;
                dataGridView4.Rows[0].Cells[5].Value = Convert.ToString(a);
                //coloca no grid valor da temperatura calibrada
                }
            if (radioButton7.Checked)
            {
                double x, y, a, b, b1, a1, v_ad_final, v_ad_inicial, temp_final, temp_inicial;
                v_ad_inicial = 0;// Convert.ToDouble(dataGridView5.Rows[0].Cells[1].Value);
                v_ad_final = Convert.ToDouble(dataGridView5.Rows[0].Cells[3].Value);
                temp_inicial = Convert.ToDouble(dataGridView5.Rows[0].Cells[2].Value);
                temp_final = Convert.ToDouble(dataGridView5.Rows[0].Cells[4].Value);
                a = (temp_final - temp_inicial) / (v_ad_final - v_ad_inicial);
                dataGridView5.Rows[0].Cells[5].Value = Convert.ToString(a);
            }
        }
        #endregion
        #region botao salvar calibracao
        private void button18_Click(object sender, EventArgs e)
        {
            if (radioButton6.Checked)//grava na tabela co2
            {
                    double vaca;//variavel que identifica que tem um coeficiente de calibracao
, se nao tiver nao salva
                    vaca = Convert.ToDouble(dataGridView4.Rows[0].Cells[5].Value);
                    if (vaca > 0 || vaca < 0)
                    {
                        string server, database, user, passwd, tabela_temp, tabela_co2;
                        server = textBox1.Text;
                        database = textBox2.Text;
                        user = textBox3.Text;
                        passwd = textBox4.Text;
                        tabela_co2 = "calibration_co2";//tabela dados co2
                        string offset, conv_ad, valor_pad, coef_linear, datum, zeit;
                        double coef_linear1;
                        offset= Convert.ToString(dataGridView4.Rows[0].Cells[2].Value);
                        conv_ad = Convert.ToString(dataGridView4.Rows[0].Cells[3].Value);
                        valor_pad = Convert.ToString(dataGridView4.Rows[0].Cells[4].Value);
                        coef_linear1 = Convert.ToDouble(dataGridView4.Rows[0].Cells[5].Value);
                        datum = Convert.ToString(dataGridView4.Rows[0].Cells[6].Value);
                        zeit = Convert.ToString(dataGridView4.Rows[0].Cells[7].Value);
                        MySqlConnection conection = new MySqlConnection("Server=" + server +
";Database=" + database + ";Uid=" + user + ";Pwd=" + passwd + "");//só pode variável string se
numerico devo usar variavel.tostring()
                        try
                        {
                            if (conection.State == ConnectionState.Closed)
                                conection.Open();
                            MySqlCommand comando = new MySqlCommand("INSERT INTO " +
tabela_co2 + " (offset,conv_pd,valor_pd,coef_linear,DATA,HORA) VALUES (" +
offset.ToString().Replace(',','.') + "," + conv_ad + "," +
valor_pad.ToString().Replace(',','.') + ",'" + coef_linear1.ToString().Replace(',','.') +
"','"+datum+"','"+zeit+"')", conection);
                            comando.ExecuteNonQuery();
                            conection.Close();
                        }
                        catch (Exception ex)
                        {
                            MessageBox.Show(ex.StackTrace);
                        }
                    }
            }
            if (radioButton7.Checked)
            {
                    double boi;
                    boi = Convert.ToDouble(dataGridView5.Rows[0].Cells[5].Value);
                    if (boi > 0 || boi > 0)
                    {
```

```
                        string server, database, user, passwd, tabela_temp, tabela_co2;
                        server = textBox1.Text;
                        database = textBox2.Text;
                        user = textBox3.Text;
                        passwd = textBox4.Text;
                        tabela_temp = "calibration_temp";//teste para mais de 1000 funcionou
mas nao mostra somente no dos
                        string offset, conv_ad, valor_pad, coef_linear, datum, zeit;
                        double coef_linear1;
                        offset= Convert.ToString(dataGridView5.Rows[0].Cells[2].Value);
                        conv_ad = Convert.ToString(dataGridView5.Rows[0].Cells[3].Value);
                        valor_pad = Convert.ToString(dataGridView5.Rows[0].Cells[4].Value);
                        coef_linear1 = Convert.ToDouble(dataGridView5.Rows[0].Cells[5].Value);
                        datum = Convert.ToString(dataGridView5.Rows[0].Cells[6].Value);
                        zeit = Convert.ToString(dataGridView5.Rows[0].Cells[7].Value);
                        MySqlConnection conection = new MySqlConnection("Server=" + server +
";Database=" + database + ";Uid=" + user + ";Pwd=" + passwd + "");//só pode variável string se
numerico devo usar variavel.tostring()
                        try
                        {
                            if (conection.State == ConnectionState.Closed)
                                conection.Open();
                            MySqlCommand comando = new MySqlCommand("INSERT INTO " +
tabela_temp + " (offset,conv_pd,valor_pd,coef_linear,DATA,HORA) VALUES (" +
offset.ToString().Replace(',', '.') + "," + conv_ad + "," + valor_pad.ToString().Replace(',',
'.') + ",'" + coef_linear1.ToString().Replace(',', '.') + "','" + datum + "','" + zeit + "')",
conection);
                            comando.ExecuteNonQuery();
                            conection.Close();
                        }
                        catch (Exception ex)
                        {
                            MessageBox.Show(ex.StackTrace);
                        }
                    }
                }
            }
#endregion

        #endregion

        #region  botao chama calibracao tela graficos
        //chama calibracao
        private void button22_Click(object sender, EventArgs e)
          {
            if (radioButton3.Checked)//busca.....tabela calibration_co2
            {
                        string server, database, user, passwd, tabela_temp, tabela_co2;
                        server = textBox1.Text;
                        database = textBox2.Text;
                        user = textBox3.Text;
                        passwd = textBox4.Text;
                        tabela_co2 = "calibration_co2";//tabela dados co2
                        MySqlConnection conection = new MySqlConnection("Server=" + server +
";Database=" + database + ";Uid=" + user + ";Pwd=" + passwd + "");//só pode variável string se
numerico devo usar variavel.tostring()

                        MySqlCommand comando = new MySqlCommand("SELECT * FROM "+tabela_co2+"
WHERE idcalibration_co2 = (SELECT max(idcalibration_co2) from
calibration_co2)",conection);//comando funcionando
                        MySqlDataAdapter leitura = new MySqlDataAdapter(comando);
                        try
                        {
                            if (conection.State == ConnectionState.Closed)
                                conection.Open();
                            DataTable tabela1 = new DataTable();
                            leitura.Fill(tabela1);
                            dataGridView6.Columns.Clear();
                            dataGridView6.DataSource = null;
                            dataGridView6.DataSource = tabela1;

                            conection.Close();
                        }
                        catch (Exception ex)
                        {
                            MessageBox.Show(ex.StackTrace);
                        }
```

```
                    //}
                }
                if (radioButton4.Checked)//busca.....tabela calibration_temp
                {
                    string server, database, user, passwd, tabela_temp, tabela_co2;
                    server = textBox1.Text;
                    database = textBox2.Text;
                    user = textBox3.Text;
                    passwd = textBox4.Text;
                    tabela_temp = "calibration_temp";//tabela dados co2
                    MySqlConnection conection = new MySqlConnection("Server=" + server +
";Database=" + database + ";Uid=" + user + ";Pwd=" + passwd + "");//só pode variável string se
numerico devo usar variavel.tostring()
                    MySqlCommand comando = new MySqlCommand("SELECT * FROM " + tabela_temp + "
WHERE idcalibration_temp = (SELECT max(idcalibration_temp) from calibration_temp)",
conection);//comando funcionando
                    MySqlDataAdapter leitura = new MySqlDataAdapter(comando);
                    try
                    {
                        if (conection.State == ConnectionState.Closed)
                            conection.Open();
                        DataTable tabela1 = new DataTable();
                        leitura.Fill(tabela1);
                        dataGridView6.Columns.Clear();
                        dataGridView6.DataSource = null;
                        dataGridView6.DataSource = tabela1;

                        conection.Close();
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show(ex.StackTrace);
                    }
                }

            }
#endregion

            #region botao chama calibracao aquisição
            private void button23_Click(object sender, EventArgs e)
            {
                ////////////////////////////////////chama dados co2
                string server, database, user, passwd, tabela_temp, tabela_co2;
                server = textBox1.Text;
                database = textBox2.Text;
                user = textBox3.Text;
                passwd = textBox4.Text;
                tabela_co2 = "calibration_co2";//tabela dados co2
                MySqlConnection conection = new MySqlConnection("Server=" + server + ";Database="
+ database + ";Uid=" + user + ";Pwd=" + passwd + "");//só pode variável string se numerico
devo usar variavel.tostring()
                MySqlCommand comando = new MySqlCommand("SELECT * FROM " + tabela_co2 + " WHERE
idcalibration_co2 = (SELECT max(idcalibration_co2) from calibration_co2)",
conection);//comando funcionando
                MySqlDataAdapter leitura = new MySqlDataAdapter(comando);
                try
                {
                    if (conection.State == ConnectionState.Closed)
                        conection.Open();
                    DataTable tabela1 = new DataTable();
                    leitura.Fill(tabela1);
                    dataGridView7.Columns.Clear();
                    dataGridView7.DataSource = null;
                    dataGridView7.DataSource = tabela1;

                    conection.Close();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.StackTrace);
                }
                ////////////////////////////chama dados temperatura
                tabela_co2 = "calibration_temp";//tabela dados co2
                MySqlConnection conection1 = new MySqlConnection("Server=" + server + ";Database="
+ database + ";Uid=" + user + ";Pwd=" + passwd + "");//só pode variável string se numerico
devo usar variavel.tostring()
```

```
            MySqlCommand comando1 = new MySqlCommand("SELECT * FROM " + tabela_co2 + " WHERE
idcalibration_temp = (SELECT max(idcalibration_temp) from calibration_temp)",
conection1);//comando funcionando
            MySqlDataAdapter leitura1 = new MySqlDataAdapter(comando1);
            try
            {
                if (conection.State == ConnectionState.Closed)
                    conection.Open();
                DataTable tabela1 = new DataTable();
                leitura1.Fill(tabela1);
                dataGridView8.Columns.Clear();
                dataGridView8.DataSource = null;
                //dataGridView7.Rows.Add(1);
                dataGridView8.DataSource = tabela1;

                conection.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.StackTrace);
            }

        }
        #endregion

    }
}
```

ANEXO A – DATASHEET MG811 SENSOR DE CO_2

Hanwei Electronics MG-811 http://www.hwsensor.com

MG811 CO2 Sensor

Features

Good sensitivity and selectivity to CO2
Low humidity and temperature dependency
Long stability and reproducibility

Application

Air Quality Control
Ferment Process Control
Room Temperature CO2 concentration Detection

Structure and Testing Circuit

Sensor Structure and Testing Circuit as Figure, It composed by solid electrolyte layer (1), Gold electrodes (2), Platinum Lead (3), Heater (4), Porcelain Tube (5), 100m double-layer steeless net (6), Nickel and copper plated ring (7), Bakelite (8), Nickel and copper plated pin (9).

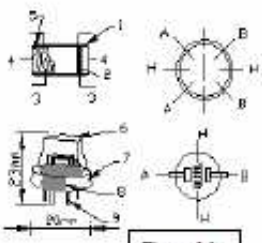


Figure 1A

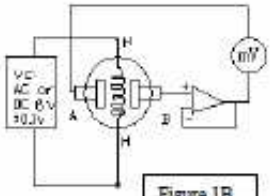
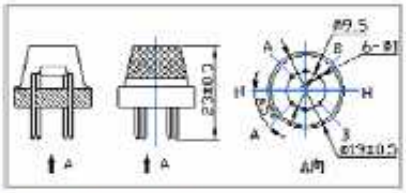
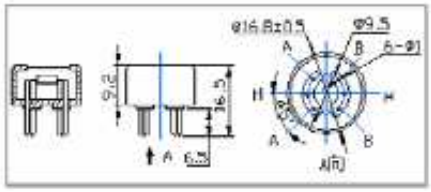


Figure 1B





Working Principle

Sensor adopt solid electrolyte cell Principle. It is composed by the following solid cells:
Air, Au|NASICON|| carbonate|Au, air, CO2

When the sensor exposed to CO2, the following electrodes reaction occurs:
Cathodic reaction: $2Li^+ + CO_2 + 1/2O_2 + 2e^- = Li_2CO_3$
Anodic reaction: $2Na^+ + 1/2O_2 + 2e^- = Na_2O$
Overall chemical reaction: $Li_2CO_3 + 2Na^+ = Na_2O + 2Li^+ + CO_2$

The Electromotive force (EMF) result from the above electrode reaction, accord with according to Nernst's equation::

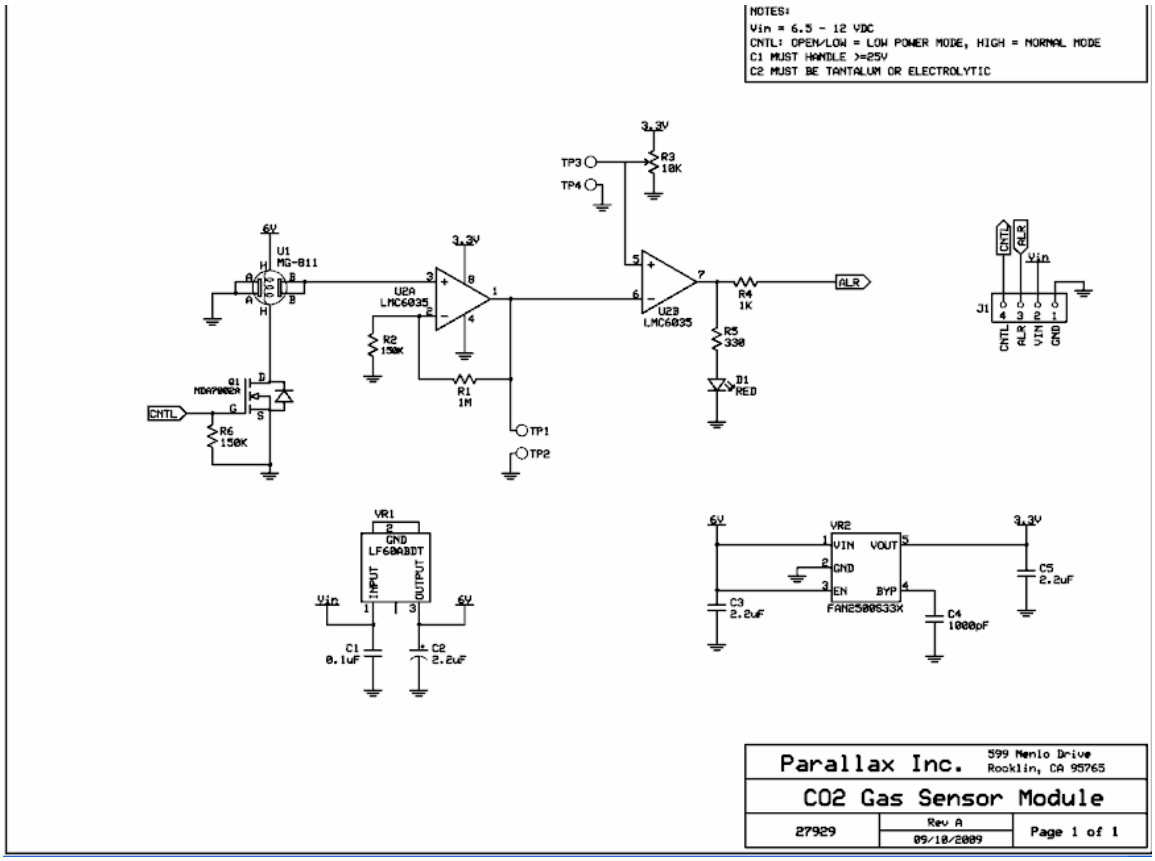
$$EMF = Ec - (R \times T) / (2F) \ln (P(CO_2))$$

$P(CO_2)$—CO2--- partial Pressure Ec—Constant Volume R—Gas Constant volume
T— Absolute Temperature (K) F—Faraday constant

From Figure 1B, Sensor Heating voltage supplied from other circuit , When its surface temperature is high enough , the sensor equals to a cell, its two sides would output voltage signal ,and its result accord with Nernst's equation. In sensor testing, the impedance of amplifier should be within 100—1000GΩ. Its testing

Tel: 86-371-67169080 Fax: 86 371 67169090 E-mail: sales@hwsensor.com

ANEXO B – DATASHEET PRÉ AMPLIFICADOR MG811



ANEXO C – DATASHEET LM35 SENSOR DE TEMPERATURA

National Semiconductor

November 2000

LM35
Precision Centigrade Temperature Sensors

General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55 to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a −55° to +150°C temperature range, while the LM35C is rated for a −40° to +110°C range (−10° with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

Features

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guaranteeable (at +25°C)
- Rated for full −55° to +150°C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 µA current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only ±¼°C typical
- Low impedance output, 0.1 Ω for 1 mA load

Typical Applications

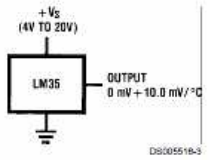
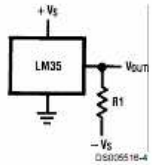


FIGURE 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)



Choose $R_1 = -V_S/50\ \mu A$
V_{OUT} = +1,500 mV at +150°C
= +250 mV at +25°C
= −550 mV at −55°C

FIGURE 2. Full-Range Centigrade Temperature Sensor

LM35 Precision Centigrade Temperature Sensors

ANEXO D – DATASHEET PIC16F916

MICROCONTROLADOR

MICROCHIP **PIC16F913/914/916/917/946**

28/40/44/64-Pin Flash-Based, 8-Bit CMOS Microcontrollers with LCD Driver and nanoWatt Technology

High-Performance RISC CPU:

- Only 35 Instructions to learn:
  - All single-cycle instructions except branches
- Operating speed:
  - DC – 20 MHz oscillator/clock input
  - DC – 200 ns instruction cycle
- Program Memory Read (PMR) capability
- Interrupt capability
- 8-level deep hardware stack
- Direct, Indirect and Relative Addressing modes

Special Microcontroller Features:

- Precision Internal Oscillator:
  - Factory calibrated to ±1%, typical
  - Software selectable frequency range of 8 MHz to 125 kHz
  - Software tunable
  - Two-Speed Start-up mode
  - External Oscillator fail detect for critical applications
  - Clock mode switching during operation for power savings
- Software selectable 31 kHz internal oscillator
- Power-Saving Sleep mode
- Wide operating voltage range (2.0V-5.5V)
- Industrial and Extended temperature range
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Brown-out Reset (BOR) with software control option
- Enhanced Low-Current Watchdog Timer (WDT) with on-chip oscillator (software selectable nominal 268 seconds with full prescaler) with software enable
- Multiplexed Master Clear with pull-up/input pin
- Programmable code protection
- High-Endurance Flash/EEPROM cell:
  - 100,000 write Flash endurance
  - 1,000,000 write EEPROM endurance
  - Flash/Data EEPROM retention: > 40 years

Low-Power Features:

- Standby Current:
  - <100 nA @ 2.0V, typical
- Operating Current:
  - 11 µA @ 32 kHz, 2.0V, typical
  - 220 µA @ 4 MHz, 2.0V, typical
- Watchdog Timer Current:
  - 1 µA @ 2.0V, typical

Peripheral Features:

- Liquid Crystal Display module:
  - Up to 60/96/168 pixel drive capability on 28/40/64-pin devices, respectively
  - Four commons
- Up to 24/35/53 I/O pins and 1 input-only pin:
  - High-current source/sink for direct LED drive
  - Interrupt-on-change pin
  - Individually programmable weak pull-ups
- In-Circuit Serial Programming™ (ICSP™) via two pins
- Analog comparator module with:
  - Two analog comparators
  - Programmable on-chip voltage reference (CV_{REF}) module (% of V_{DD})
  - Comparator inputs and outputs externally accessible
- A/D Converter:
  - 10-bit resolution and up to 8 channels
- Timer0: 8-bit timer/counter with 8-bit programmable prescaler
- Enhanced Timer1:
  - 16-bit timer/counter with prescaler
  - External Timer1 Gate (count enable)
  - Option to use OSC1 and OSC2 as Timer1 oscillator if INTOSCIO or LP mode is selected
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Addressable Universal Synchronous Asynchronous Receiver Transmitter (AUSART)
- Up to 2 Capture, Compare, PWM modules:
  - 16-bit Capture, max. resolution 12.5 ns
  - 16-bit Compare, max. resolution 200 ns
  - 10-bit PWM, max. frequency 20 kHz
- Synchronous Serial Port (SSP) with I^2C™

ANEXO E – DATASHEET MCP619 AMPLIFICADOR OPERACIONAL

MICROCHIP **MCP616/617/618/619**

2.3V to 5.5V Micropower Bi-CMOS Op Amps

FEATURES

- Low Power: I_{DD} = 25 µA, max
- Low Offset Voltage: 150 µV, max
- Rail-to-Rail Swing at Output
- Low Input Offset Current: 0.3 nA, typical
- Specifications rated for 2.3V to 5.5V Supplies
- Unity Gain Stable
- Chip Select ($\overline{CS}$) Capability with MCP618
- Industrial Temperature range supported
- No Phase Reversal
- Available in Single, Dual, and Quad

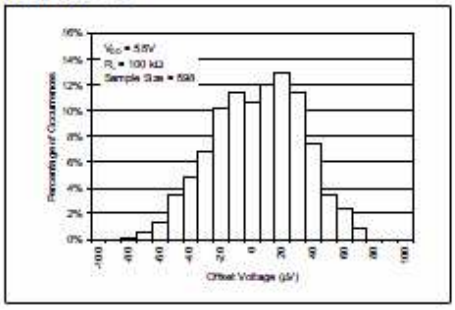
APPLICATIONS

- Battery Powered Instruments
- Strain Gauges
- Medical Instruments
- Test Equipment

AVAILABLE TOOLS

- Spice Macromodels (at www.microchip.com)
- FilterLab™ Software (at www.microchip.com)

HISTOGRAM OF INPUT OFFSET VOLTAGE

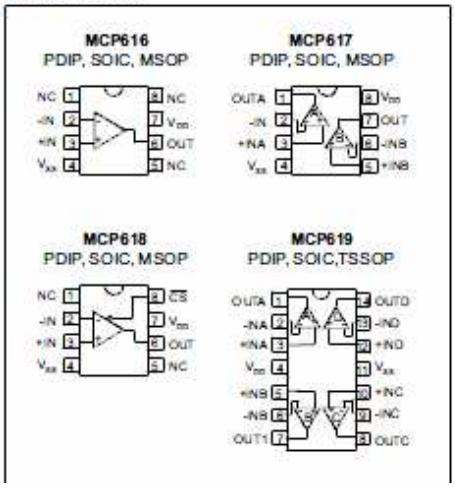


DESCRIPTION

The MCP616, MCP617, MCP618 and MCP619 from Microchip Technology, Inc. are unity gain stable, low offset voltage operational amplifiers capable of precision low power single supply operation. Performance characteristics include ultra low offset voltage (150 µV, max.), rail-to-rail output swing capability, and low input offset current (0.3 nA typ.). These features make this family of amplifiers well suited for single supply precision, high impedance, battery powered applications.

The single MCP616 is available in standard 8-lead PDIP, SOIC, and MSOP packages. Another version of the single op amp, MCP618 is offered with a Chip Select ($\overline{CS}$) option in standard 8-lead PDIP, SOIC, and MSOP packages. The dual MCP617 is offered in standard 8-lead PDIP, SOIC, as well as the MSOP package. Finally, the quad MCP619 is offered in 14-lead PDIP, SOIC and TSSOP packages. All devices are fully specified from -40°C to +85°C with power supplies from 2.3V to 5.5V.

PACKAGES



]

ANEXO F – DATASHEET RS232 CONVERSOR SERIAL

TEXAS INSTRUMENTS

TRS232

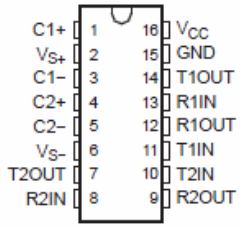
www.ti.com SLLS861A–AUGUST 2007–REVISED SEPTEMBER 2008

DUAL RS-232 DRIVER/RECEIVER WITH IEC61000-4-2 PROTECTION

FEATURES

- Meets or Exceeds TIA/RS-232-F and ITU Recommendation V.28
- Operates From a Single 5-V Power Supply With 1.0-μF Charge-Pump Capacitors
- Operates up to 120 kbit/s
- Two Drivers and Two Receivers
- ±30-V Input Levels
- Low Supply Current . . . 8 mA Typical
- ESD Protection Exceeds JESD22
  - 2000-V Human-Body Model (HBM) (A114-A)
- Upgrade With Improved ESD (15-kV HBM) and 0.1-μF Charge-Pump Capacitors Is Available With the TRS202

D, DW, N, NS, OR PW PACKAGE
(TOP VIEW)



APPLICATIONS

- TIA/RS-232-F
- Battery-Powered Systems
- Terminals
- Modems
- Computers

DESCRIPTION/ORDERING INFORMATION

The TRS232 is a dual driver/receiver that includes a capacitive voltage generator to supply TIA/RS-232-F voltage levels from a single 5-V supply. Each receiver converts TIA/RS-232-F inputs to 5-V TTL/CMOS levels. This receiver has a typical threshold of 1.3 V, a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into TIA/RS-232-F levels. The driver, receiver, and voltage-generator functions are available as cells in the Texas Instruments LinASIC™ library.

ANEXO G– CERTIFICADO CALIBRAÇÃO CILINDRO CO_2

THE LINDE GROUP

Linde

1/1

Data / *Date*
23/03/2010

CERTIFICADO DE ENSAIO

Nº 682/10

Nº do cilindro / *Cylinder No* 6066D
Nº do pedido / *Order No* MOli248

Cliente / *Customer*
LABORATÓRIO ALAC

Cilindro / *Cylinder*

Tipo de cilindro *Cylinder type*	Conexão da válvula *Valve connection*	Pressão de enchimento *Filling pressure* 21 °C	Volume nominal de gás *Gas volume* 20 °C, 101,3 Kpa (a)
A- 10	ABNT-218-2	10.000 kPa	1,0 m³

Componente *Component*		Encomenda *Order*	Resultado da analise *Analysis result*	Unidade *Unit*	Incerteza abs. *Uncertainty*
Monóxido de Carbono	CO	1	1,02	% mol/mol	± 0,02
Dióxido de Carbono	CO_2	1	1,00	% mol/mol	± 0,02
Metano	CH_4	1	1,03	% mol/mol	± 0,02
Etano	C_2H_6	1	1,00	% mol/mol	± 0,02
Propano	C_3H_8	1	0,99	% mol/mol	± 0,02
Isobutano	C_4H_{10}	1	0,99	% mol/mol	± 0,02
Nitrogênio	N_2	94	94,0	% mol/mol	± 0,1

Tolerância de preparação / *Blend tolerance* : 6 % relativa / *% relative*
Estabilidade / *Shelf life* : 24 meses / *months*
Temperatura recomendada para uso/ armazenagem : 0 a / *to* 40 °C
Recommended storage and usage temperature
Pressão mínima de uso / *Minimum pressure of use* : 100 kPa

Comentários / *Comments:*
Mistura Padrão: Calibração
Mistura: Tóxico
Método Analítico: µGC – TCD
Nº método / instrução: 222 / BR-INS-0052
Padrão de Referência: Multimix (cils OI 7184, 265682, 3473356)
Rastreado a padrões de referência através do certificado: RBC/INMETRO 54.408; Nmi 32.20.324-03, 32.20.324-02
A incerteza de medição declarada é baseada em uma incerteza padronizada combinada, multiplicada por um fator de abrangência K=2,0, fornecendo um nível de confiança de aproximadamente 95%
Observações: Tranvase do cilindro BA 9177 cert. 681/10

Data de fabricação: 03/03/2010
Data de recebimento cil.: 03/03/2010
Data do ensaio: 15/03/2010

DNV
OCS 0010

Responsável : Katia Luk. Alvares.
Katia Lukjanenko Alvares
Química

Os resultados declarados referem-se apenas ao item especificado. A reprodução deste documento só poderá ser feita integralmente, sem alterações

Linde Gases Ltda.

Endereço *Mailing Address* / Local *Site*
Laboratório de Gases Especiais - LGE
Rod. D. Gabriel Paulino Bueno Couto, Km 65
CEP 13212-240 Bairro Japí - Jundiaí S.P.

Telefone / *Telephone*
0800 725 4633

Telefax / *Telefax*
(011) 4582 - 4669

BR-PRO-0014A 1
http://hiq.linde-gas.com.br