UNIVERSIDADE LUTERANA DO BRASIL

PRÓ-REITORIA DE GRADUAÇÃO

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

MARCOS HENRIQUE RIBACKI

CONTROLE MICROPROCESSADO DE TEMPERATURA EM MATRIZ PARA FORJAMENTO A QUENTE.

Canoas, dezembro de 2008.

MARCOS HENRIQUE RIBACKI

CONTROLE MICROPROCESSADO DE TEMPERATURA EM MATRIZ PARA FORJAMENTO A QUENTE.

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

Automação Industrial

Professor Orientador:

MSc. Eng. Eletr. Augusto de Mattos – CREA-RS: 88.003

Canoas

2008

FOLHA DE APROVAÇÃO

Nome do Autor: Marcos Henrique Ribacki

Matrícula: 012101089-9

Título: Controle microprocessado de temperatura em matriz para forjamento o quente.

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. Augusto de Mattos

CREA-RS: 88.003

Banca Avaliadora:

Dr. Eng. Eletr. Valner João Brusamarello
CREA-RS: 07.8158-D

Conceito Atribuído (A-B-C-D):

Dr. Eng. Eletr. João Carlos Vernetti dos Santos
CREA-RS: 04.5852-D

Conceito Atribuído (A-B-C-D):

Assinaturas:

Autor Marcos Henrique Ribacki	Orientador Augusto de Mattos
Avaliador Valner João Brusamarello	Avaliador João Carlos Vernetti dos Santos

Relatório Aprovado em:

DEDICATÓRIA

A meus pais Tarso e Rita, sem os quais não chegaria até aqui.

AGRADECIMENTOS

A empresa MAXIFORJA COMPONENTES AUTOMOTIVOS LTDA, pelo apoio financeiro, e por ter fornecido os materiais e equipamentos para a montagem do protótipo; Aos colegas que participaram direta ou indiretamente deste projeto e um agradecimento especial ao Eng. Tomaz Petracco, pelo incentivo e apoio profissional.

Agradeço também todos os professores que me possibilitaram ter um crescimento acadêmico, profissional e principalmente pessoal.

A todos que colaboraram direta ou indiretamente na elaboração deste trabalho, o meu reconhecimento.

Ao orientador, professor Augusto de Mattos, pelo apoio, dedicação e esforço pessoal proporcionado.

Aos meus pais, por sempre acreditarem e me incentivarem em tudo, para realização deste sonho.

RESUMO

Henrique Ribacki, Marcos. **Controle microprocessado de temperatura em matriz para forjamento quente**. Trabalho de Conclusão de Curso em Engenharia Elétrica - Departamento de Engenharia Elétrica. Universidade Luterana do Brasil. Canoas, RS. 2008.

Atualmente, na industria há uma necessidade contínua de automação e padronização dos processos produtivos, objetivando cada vez mais a redução de custos e aumento da produtividade. Este projeto tem como objetivo o controle contínuo da temperatura da matriz ferramental durante o processo de forjamento, através do aquecimento controlado de forma microprocessada. Foi desenvolvido e testado um sistema de controle digital, através do microcontrolador 8052, programado em linguagem C, atuando como um sistema PID digital. O sistema desenvolvido automatiza totalmente o aquecimento da matriz de forjamento. Também controla o correto funcionamento das resistências de aquecimento.

Palavras chave: Indústria. Padronização. Processos. Controle. Digital.

ABSTRACT

Henrique Ribacki, Marcos. Microprocessed control of temperature in tool the hot forging. Work of Conclusion of Course in Electrical Engineering - Electrical Engineering Department. Lutheran University of Brazil. Canoas, RS. 2008.

Currently in the industry there is a continuing need for automation and standardization of production processes, aiming cost reducing and productivity increasing. This project aims to continuous control the temperature of tooling during the forging process, through controlled heating in a microprocessor. Digital control system with the 8052 microcontroller, was developed and tested, programmed in C language, acting as a PID digital system. The system developed completely automates the heating die forging. It also controls the correct functioning of heating resistance.

Keywords: Industry. Standardization. Process. Control.Digital.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SÍMBOLOS

P – Ação Proporcional
I – Ação Integral
D – Ação Derivativa
Kp – Ganho proporcional
Ti – Constante de tempo integrativa
Td – Constante de tempo derivativa
τa – Tempo de retardo do método de ziegler-nichols (resposta ao salto)
Td – Constante de tempo do método de ziegler-nichols (resposta ao salto)
Mo – Overshoot
A/D – Conversor analógico para digital
D/A – Conversor digital para analógico
PWM – Pulse Width Modulation
e – Erro entre o valor desejado e o real
Vc – Sinal de controle
Ton – Tempo ligado
Toff – Tempo desligado

SUMÁRIO

1. INTRODUÇÃO

No processo de forjamento a quente, a estabilidade da temperatura da matriz ferramental é um fator significante na durabilidade do mesmo, bem como uma temperatura de trabalho onde possa ser observado o equilíbrio entre desgaste e tenacidade do material.

O sistema de controle microprocessado, busca um equilíbrio da temperatura próximo ao ponto de maior durabilidade da matriz. Com isso reduz-se as perdas com paradas de produção para aquecimento manual, ou para a troca do ferramental com desgaste prematuro ou trincas que mais tarde levarão à quebra do mesmo causando a inutilização precoce da ferramenta.

Este projeto tem como objetivo a construção do protótipo de controle, baseado na família 8052, com o processador ATMEL AT89C52, atuando no sistema de aquecimento independentemente para duas matrizes de forma simultânea, através dos elementos de aquecimento inseridos diretamente na base de fixação da matriz superior e inferior. O sistema prevê a medição de duas temperaturas para cada sistema, sendo uma temperatura de controle do sistema de aquecimento e outra a monitoração da caixa de ligação.

Foi implementado um sistema de controle PID, cujos resultados são a análise das curvas de resposta do sistema, e alguns dados comparativos do aquecimento manual e automático do ferramental.

1.1. Visão Geral do Problema

O processo de forjamento exige do ferramental um pré-aquecimento, para que se consiga uma durabilidade mínima das matrizes. Com isso minimizam-se as quebras prematuras, que são causadas principalmente por trincas geradas por esforços mecânicos. A fadiga térmica entre outros fatores ocorre a partir da falta de tenacidade do material.

A figura 1-1 mostra o sistema aquecido de forma manual antes do forjamento, com o auxilio de chama gerada a gás, em contato direto com as matrizes.

Figura 1-1: Sistema de aquecimento manual

Este sistema de aquecimento gera paradas no processo produtivo, antes de cada turno e até mesmo durante o turno de trabalho. A temperatura da matriz reduz muito devido a dissipação térmica e não reposição do calor através do processo.

Outro fator relevante é a não uniformidade da temperatura da matriz, obtida apenas através do aquecimento superficial. Isto ocorre pelo contato direto da matriz com peças aquecidas elevando a temperatura de superfície, onde é sempre maior do que a temperatura da base ou de núcleo da matriz. Esta temperatura é influenciada pelo contato da parte inferior da matriz na mesa da prensa, que se torna um grande caminho de dissipação da temperatura do ferramental.

O aquecimento superficial, também pode gerar trincas na matriz pelo processo de fadiga térmica, ilustrado com a figura 1-2. Em A uma matriz sem aquecimento e em B com aquecimento superficial ressaltando a dilatação a superfície.

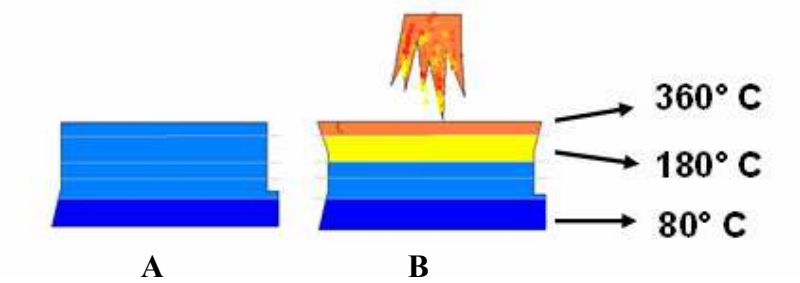


Figura 1-2: Ilustração da fadiga térmica

O que se busca é a uniformidade da temperatura com um sistema de aquecimento elétrico, controlado de forma precisa, através de elementos de aquecimento no interior da mesa. Onde era o principal ponto de dissipação térmica, agora será a ponte de reposição térmica.

O sistema precisa ser flexível ao ajuste de temperatura para cada processo, com controle estável, possibilitando assim a adaptação do sistema da melhor forma possível. Para isso será utilizado um controle PID.

1.2. Formulação do Problema de Engenharia

Variação térmica na matriz ferramental.

1.3. Formulação do Problema Comercial

A decisão por desenvolver este projeto é uma necessidade contínua de automação e padronização dos processos produtivos. Objetivando cada vez mais a redução de custos e aumento da produtividade, seja por otimização de processos ou por redução de paradas.

Acredita-se que é possível aumentar a vida útil da matriz ferramental, com a diminuição das variações térmicas do sistema.

1.4. Definição do Escopo do Projeto

O projeto contempla o desenvolvimento de um sistema de processamento e controle de aquecimento de matriz ferramental.

A partir de sensores de temperatura, posicionado na matriz ferramental será feito o processamento do controle PWM da atuação das resistências de aquecimento, com o processador AT89C52.

O sistema é independente, desenvolvido de forma completa, sendo um painel de controle contendo o módulo de processamento e interface com o operador, e o sistema de potência.

2. SISTEMAS DE CONTROLE

2.1. Sistema de controle proporcional

É o sistema mais simples de controle, constituído por um controlador com apenas o modo proporcional (P). Neste tipo de controlador, a variável de controle mantém uma relação linear entre o desvio ou erro (e) e o elemento final de controle, onde e representa o quanto a variável a ser controlada difere do valor desejado. O sinal de controle (v_c) é dado pela equação 2.1,

$$v_c = Kp.e$$

[Equação - 2.1]

Em que Kp = Constante de ganho proporcional.

Neste caso, a saída do controlador depende apenas da amplitude do erro no instante de tempo. A função de transferência $Gc(s)$ do controlador é dada pela equação 2.2.

$$Gc(s) = Kp$$

[equação - 2.2]

É comum que a saída do controlador seja expressa como uma porcentagem da saída total do controlador (eq. 2.3). Desta forma, uma variação de 100% em v_c corresponde a uma variação no erro, de um extremo a outro da banda proporcional (BP).

$$Kp = \frac{100}{BP}$$

[equação - 2.3]

O controle proporcional não consegue manter a variável controlada no valor desejado, quando a banda proporcional necessita de grande magnitude, ou quando há variação na carga, ou enquanto houver perturbação no processo. Logo,

terá uma alteração da variável controlada, a qual é denominada de desvio permanente. O valor deste desvio, é diretamente proporcional à largura da banda proporcional, que é ilustrada na fig 2-1.

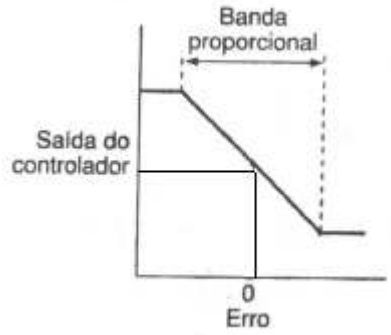


Figura 2-1: Largura de banda

2.2. Sistema de controle proporcional – Integral (PI)

É utilizado, quando o controle proporcional, não é suficiente para manter a variável controlada no valor desejado.

Um sistema de controle Proporcional – Integral PI, é um controle cuja velocidade de correção é proporcional a amplitude do desvio, enquanto existir desvio, a saída do controlador aumenta ou diminui.

A equação 2.4 mostra como fica a ação de controle.

$$\upsilon_c = BP.e + \frac{1}{Ti}\int e.dt$$

[equação - 2.4]

Ti = Tempo integral ou repetições por minuto, é o tempo que a ação integral leva para repetir o efeito da ação proporcional.

Quanto mais repetições por minuto, mais forte será a ação integral, ou seja, a "velocidade" de ação da parcela integral de controle depende da constante de tempo Ti (min) que deverá ser ajustada à melhorar o desempenho, minimizando as oscilações em torno do valor desejado.

Nota-se que há duas ações de controle: a primeira parcela $BP.Ve$, definirá a banda proporcional, enquanto a segunda dará o deslocamento necessário na banda proporcional.

Esta correção adicional atua como se estivesse deslocando a banda proporcional no sentido correto, para reduzir o desvio a zero. O deslocamento da banda proporcional cessará quando o desvio for reduzido a zero.

2.3. Sistema de controle proporcional – integral – derivativo (PID)

Um controlador PID possui as melhores características de cada controlador se utilizado de forma independente. Um controlador PID, consiste de uma porção do PI conectada em série com um PD. A função de transferência pode ser escrita como:

$$Gc(s) = Kp + \frac{Ki}{s} + Kd.s$$

2.4. PWM – Pulse Width Modulation

A modulação em largura de pulso permite variar a ação de controle de forma proporcional a partir de um sinal digital. Por simplicidade considerar um interruptor que liga e desliga a tensão sobre a carga, conforme mostrado na figura 2-2.

O interruptor fechado define uma largura de pulso pelo tempo em que ele fica nesta condição, e outro intervalo pelo tempo em que ele fica aberto. Os dois tempos juntos definem o período e, portanto, uma freqüência de controle.

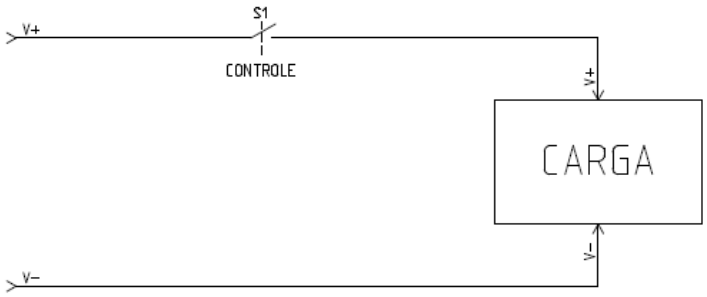


Figura 2-2: Sistema de controle PWM

A relação entre o tempo em que se tem o pulso e a duração de um ciclo completo de operação do interruptor define o ciclo ativo. Variando a largura do pulso e também o intervalo de modo a ter ciclos ativos diferentes, pode-se controlar a potência média aplicada a uma carga. Assim, quando a largura do pulso varia de zero até o máximo, a potência também varia na mesma proporção, conforme indicado na Figura 2.3.

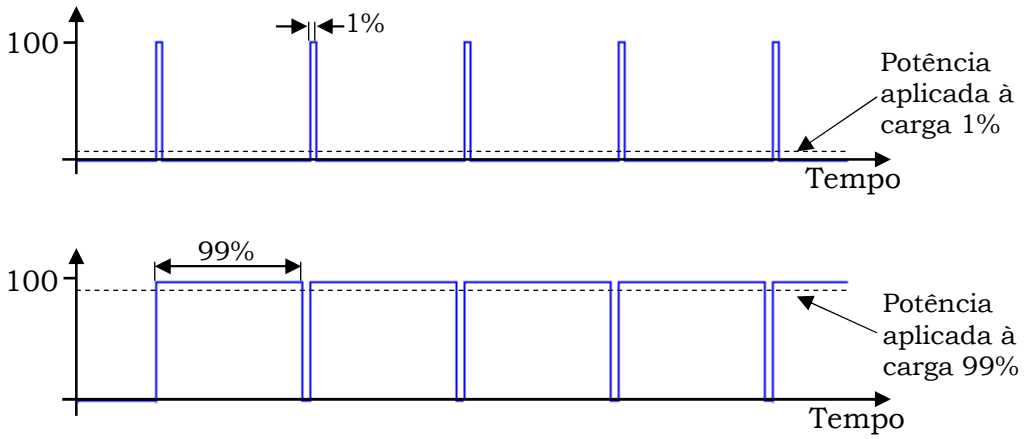


Figura 2-3:Largura de pulso controle PWM.

Este princípio é usado no controle do PWM: é modulada (variada) a largura do pulso de modo a controlar o ciclo do sinal aplicado a uma carga e, com isso, a potência aplicada à mesma.

2.5. Métodos de Ziegler-Nichols/Resposta ao salto

Os métodos de Ziegler-Nichols foram introduzidos já em 1942 e hoje são considerados clássicos [1]. Estes métodos continuam a ser largamente aplicados até hoje, mesmo em sua forma original, mas normalmente são utilizados em alguma forma modificada. Os dois métodos básicos de ajuste de Ziegler-Nichols visam obter uma mesma resposta pré-especificada para o sistema em malha fechada, e diferem no que diz respeito à natureza da informação sobre a dinâmica do processo que é exigida por cada um deles.

O método da resposta ao salto, ou método do domínio do tempo, requer o conhecimento de duas grandezas que caracterizam a resposta ao salto de um processo. Já o método do período crítico, exige o conhecimento de duas grandezas características da resposta em freqüência do processo. Uma vez obtidas estas informações, basta recorrer às equações de cada método (mais detalhes destas equações estão apresentadas nos itens 2.3.1 e 2.3.2) para calcular os ganhos do controlador. Estas equações foram determinadas de maneira empírica por meio de ensaios de processos industriais típicos. As equações originalmente propostas por Ziegler e Nichols fornecem uma resposta que foi posteriormente considerada insatisfatória. Diferentes equações foram então propostas com base nos mesmos ensaios, obtendo-se melhor desempenho.

O método da resposta ao salto consiste de equações derivadas a partir dos parâmetros da curva de reação. Ela é obtida do sistema, quando o mesmo está em

malha aberta. Pode-se ver na figura 2.4, a excitação do tipo salto, a resposta do sistema e como são obtidos os parâmetros τa (tempo de retardo) e Tp (constante de tempo).

Ao aplicar um degrau ao processo e registrar o comportamento do sistema, deve-se obter uma curva "S" como mostrada na figura 2-4.

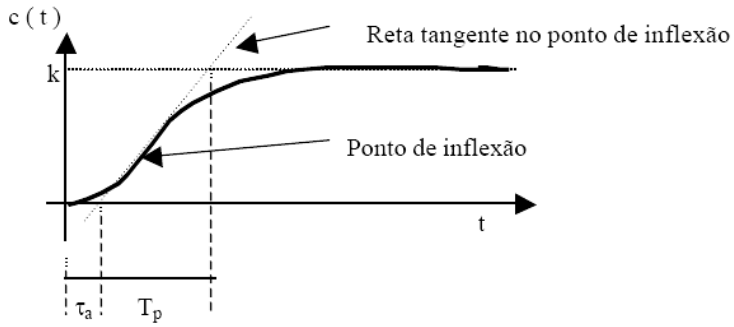


Figura 2-4: Curva em "s".

No entanto, sua aplicação é mais indicada para curva de reação na forma de um "S" e que atenda o seguinte critério: 0,1 < $\frac{\tau a}{Tp}$ < 1. Para esta curva, os parâmetros, tempo de retardo τa, e constante de tempo Tp, são determinados passando-se uma reta tangente no ponto de inflexão da curva, como ilustrado na figura 2.5.

Tendo-se obtido experimentalmente os valores de τa e Tp, pode-se recorrer à tabela 2-1para determinar os valores dos parâmetros do controlador PID.

Tipo de controlador	Kp	Ti	Td
P	$\frac{Tp}{K \cdot \tau a}$	∞	0
PI	$\frac{0{,}9 \cdot Tp}{K \cdot \tau a}$	$3{,}33 \cdot \tau a$	0
PID	$\frac{1{,}2.Tp}{K.\tau a}$	$2 \cdot \tau a$	$\frac{\tau a}{2}$

Tabela 2-1: Parâmetros de Ziegler Nichols pelo método de resposta ao salto

Os valores da tabela 2-1 foram determinados de forma empírica com objetivo de obter uma resposta com amortecimento de 1/4 na resposta à referência para processos industriais típicos (um amortecimento de 1/4 leva a um valor

máximo de overshoot de 25%). Enquanto a rejeição a perturbações muitas vezes apresenta um comportamento satisfatório, este amortecimento usualmente não é satisfatório na resposta à referência, causando em muitos casos um overshoot excessivo e baixa tolerância a variações na dinâmica do processo.

Este método limita-se a plantas de primeira ordem. Sistemas de segunda ordem, ou superior, por exemplo, não se enquadram nesta categoria. Por outro lado, este método baseia-se em identificação de formas de onda, o que pode ser problemático na prática, particularmente em aplicações com baixa relação sinal-ruído. Ainda assim, o método é adequado para um grande número de processos industriais.

3. DESENVOLVIMENTO DO PROJETO

Para a implementação de um sistema de controle, é necessário conhecer a planta que envolve o problema. Nesta seção, serão abordadas as variáveis envolvidas de forma conjunta, que caracterizam o processo a ser controlado.

3.1. Descrição Geral do Sistema

O sistema de automação realiza rotinas de controle e operação com base nas respostas das variáveis. Tais respostas são obtidas através de sensores [2], como PT100, sendo este o sensor que alimenta o sistema com a informação de temperatura, principal variável de controle.

A figura 3-1 mostra um diagrama de blocos do sistema, tornando mais objetivo o entendimento das variáveis.

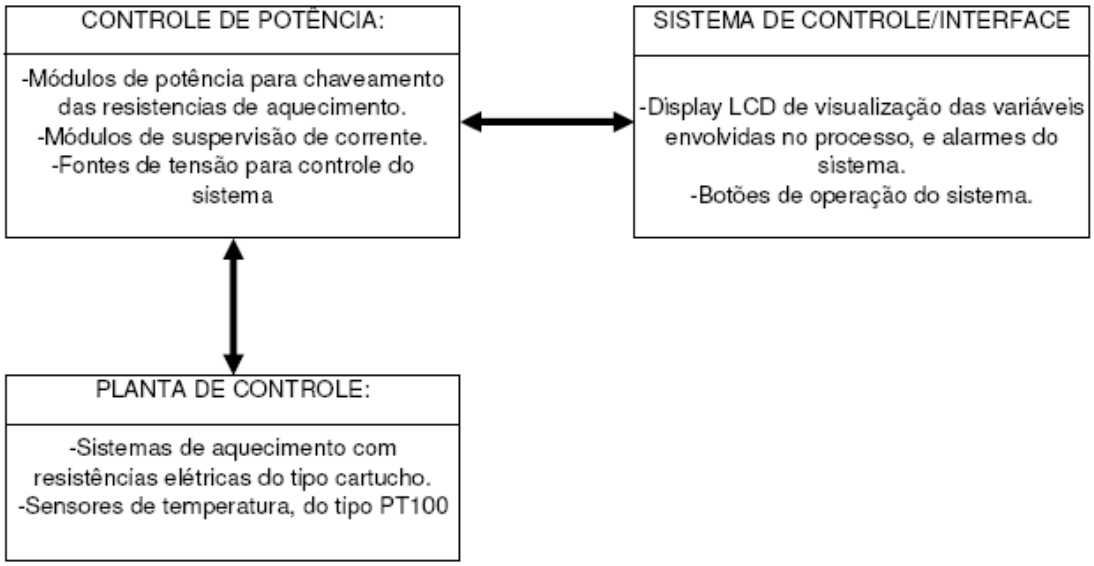


Figura 3-1:Diagrama de blocos simplificado do sistema.

3.2. Sistema de controle/Interface

A interface de operação do sistema, é composta de um display LCD do tipo 4 linhas x 20 colunas, onde serão visualizadas as variáveis do sistema. Botões para

a operação do sistema. Também está localizado na unidade de operação todo o sistema de controle da planta.

O sistema de controle/Interface pode ser subdividido em 4 partes distintas, apresentado na figura 3-2.

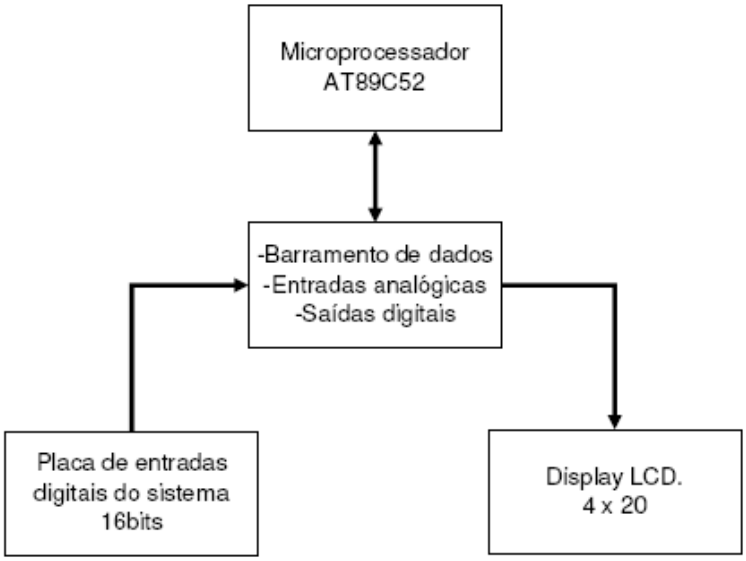


Figura 3-2: Partes do sistema de controle.

3.2.1. Microprocessador

No controle deste protótipo, é utilizado o kit de desenvolvimento 89C52 ULBEE51, que pode ser visto na figura 3-3, utilizado na disciplina de microprocessadores durante a graduação. Por estar montado de forma flexível, permite o desenvolvimento da maioria das aplicações facilitando assim uma alteração durante o desenvolvimento caso, seja necessário.

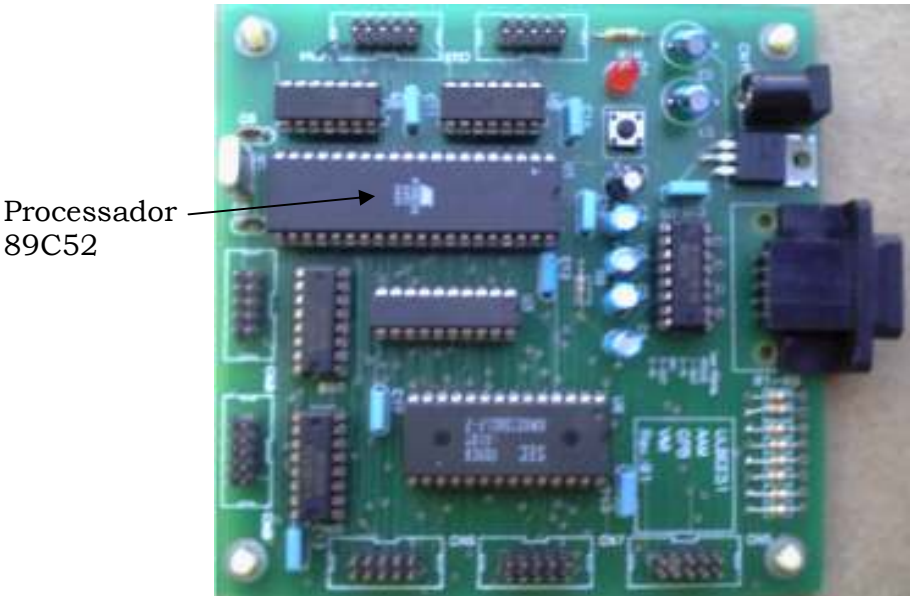


Figura 3-3: Kit de desenvolvimento AT89C52 ULBEE51.

3.2.2. Barramento de dados

O barramento de dados é a principal placa de integração do sistema, faz a integração das entradas e saídas do sistema com o microprocessador. Contém uma extensão do barramento de dados do microprocessador, e faz o controle da interface das entradas digitais, analógicas através de um conversor AD, saídas para acionamento do sistema de potência e o envio de dados para o display.

As entradas analógicas são convertidas com o ADC7824, um conversor AD com 4 canais de 12bits, utilizado em função da leitura de 4 temperaturas, com range de 0 – 400°C, com resolução de 1°C. No apêndice C, encontra-se o diagrama eletrônico desta placa.

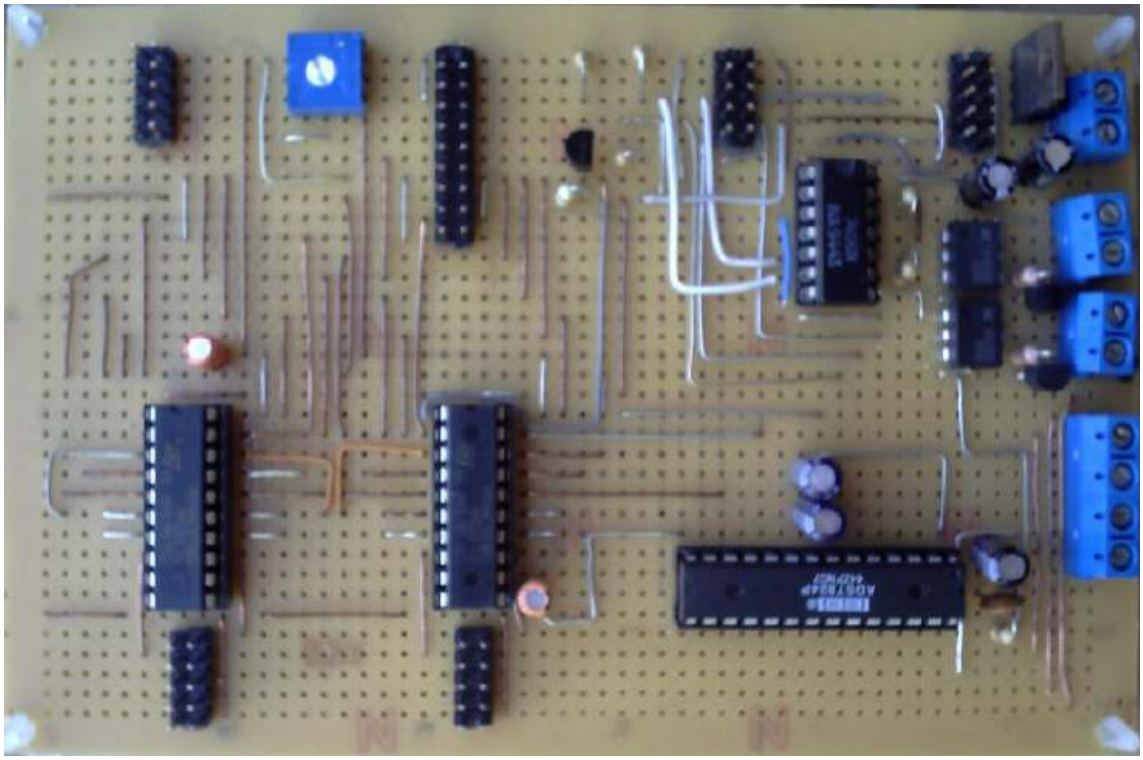


Figura 3-4: Placa do barramento de dados (A1), entradas analógicas e saídas digitais

3.2.3. Entradas digitais

A placa de entradas digitais do sistema, é a interface dos sinais de alarme e dos botões de operação. Ela faz a leitura destes sinais vindos do controle de potência (alarmes), e os botões de operação. Todos estes são ligados em nível de tensão 24Vcc, convertidos para 5Vcc através de optoacopladores 4N26, e disponibilizados ao barramento de dados em 2 bytes. O diagrama eletrônico encontra-se no apêndice C.

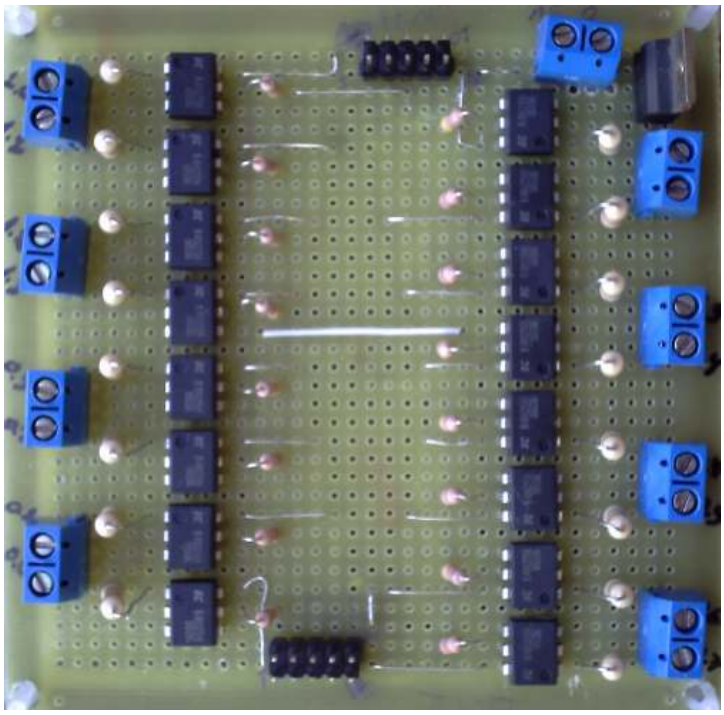


Figura 3-5: Placa de entradas digitais (A2).

3.2.4. Display LCD

O Display é a interface de visualização dos dados e parâmetros do sistema; é através dele que se faz os ajustes de set-points no microprocessador, visualização dos alarmes e status do sistema. Está ligado ao microprocessador através do barramento de dados. O datasheet encontra-se no anexo C.

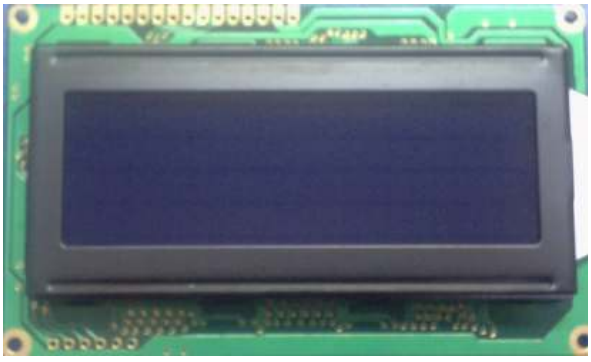


Figura 3-6: Display LCD.

3.2.5. Protótipo de controle

O protótipo de controle é uma giga de testes montada para o desenvolvimento do projeto. Todas as variáveis da planta real podem ser simuladas através do protótipo de controle. A figura 3-7 mostra o protótipo de controle.

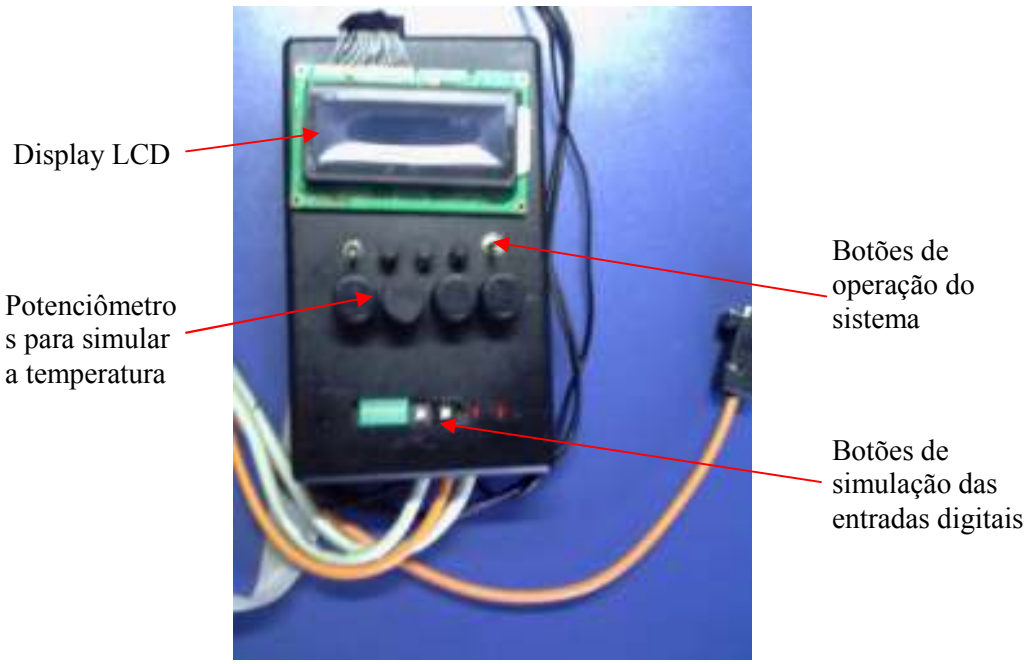


Figura 3-7: Protótipo de controle

3.2.6. Controle final implementado

Após todos os testes do sistema de controle e implementação do software, o sistema de controle foi montado em uma caixa de montagem padrão para sistemas industriais.

As figuras 3-8 e 3-9 mostram a montagem final do sistema de controle.

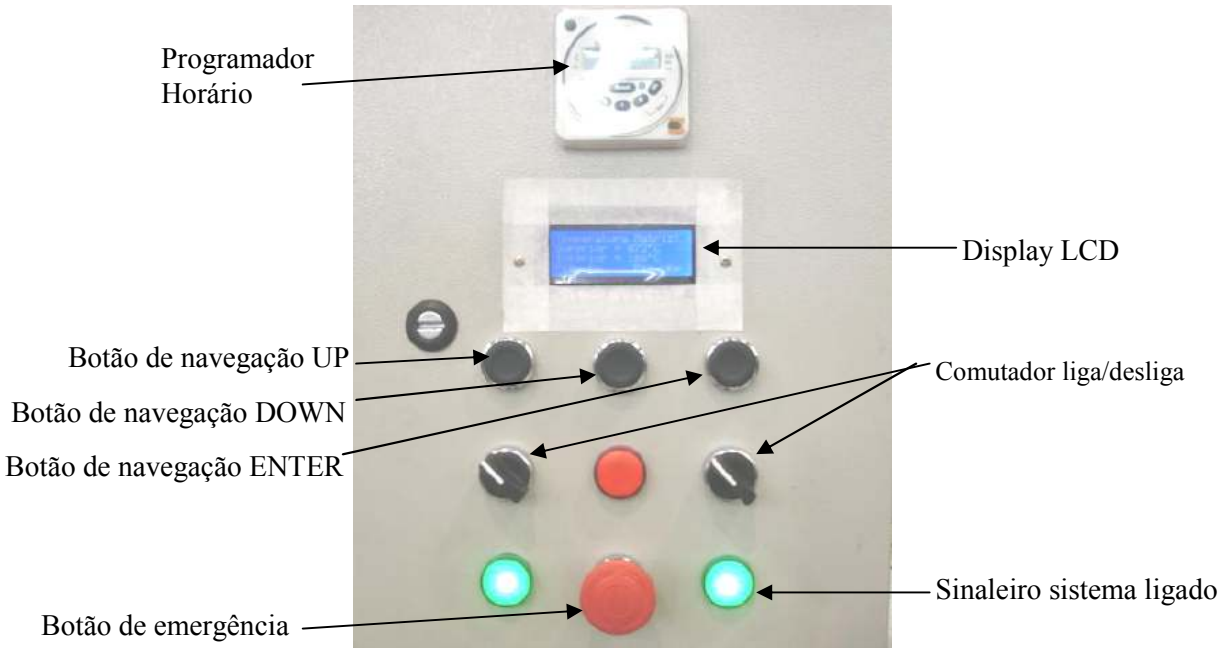


Figura 3-8: Vista do painel de operação

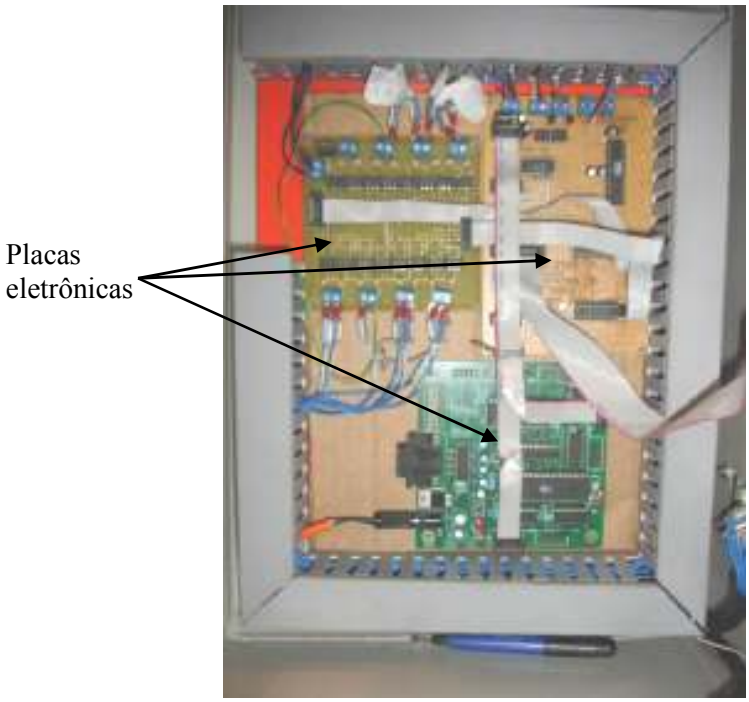


Figura 3-9: Vista da placa de montagem do painel de operação

3.3. *Controle de potência*

O controle de potência, é a unidade de chaveamento do sistema de aquecimento, através do sistema PWM, com contatores de estado sólido, do tipo 3RF2420 (Siemens). Todo projeto elétrico desta etapa foi desenvolvido com o software CADdy++Eletrical, e está no apêndice A. As figuras 3-10 e 3-11 mostram a etapa de potência.

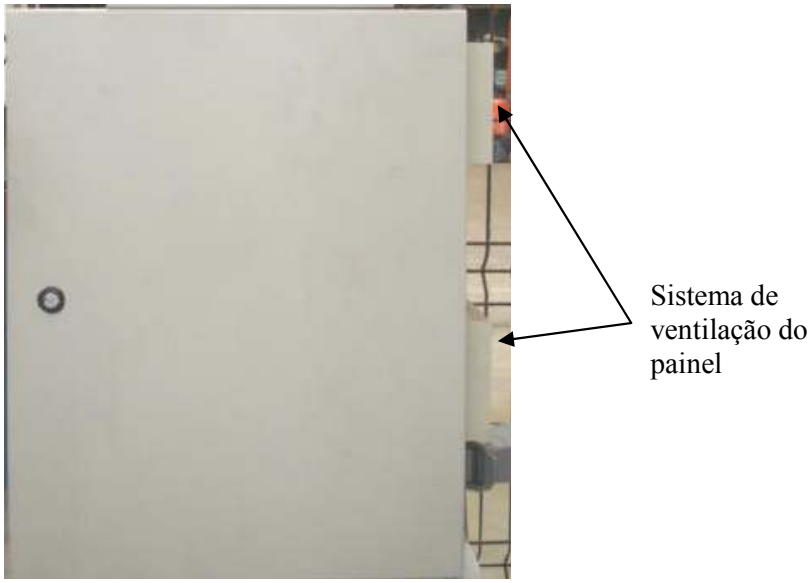


Figura 3-10: Vista externa etapa de potência

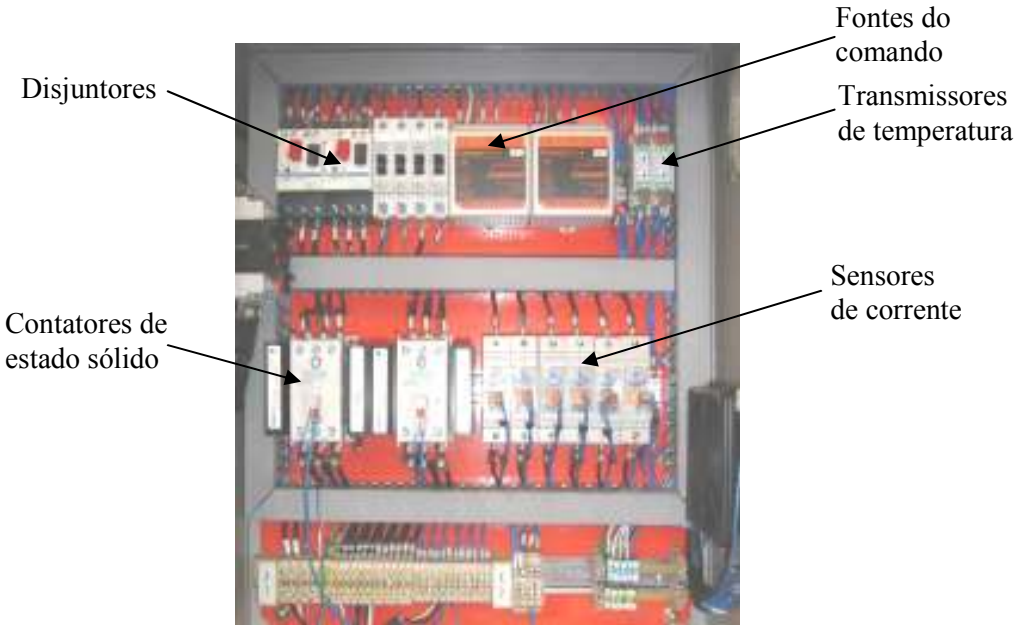


Figura 3-11: Placa de montagem etapa de potência

A etapa de potência é composta por disjuntores, contatores de estado sólido e relés de supervisão de corrente, respectivamente, para dois sistemas de aquecimento independentes. Também estão na etapa de potência as fontes de tensão que alimentam o sistema de controle.

As variáveis de temperatura, obtidas da planta, têm seus dados condicionados na etapa de potência, ou seja, a variação de resistência do sensor PT100, é ligado a um condicionador de sinal, o qual converte em saída de tensão (0-10Vcc) para o microprocessador. Foi utilizado o transmissor de temperatura TxRail da NOVUS produtos eletrônicos. O datasheet se encontra no anexo A.

3.4. Planta de controle

A planta de controle é a mesa de uma prensa de conformação mecânica, na qual são montadas as matrizes para forjamento a quente.

Cada mesa (inferior e superior) é composta por três resistências de aquecimento 380V/3800W ligadas em delta, um sensor para medição da temperatura do sistema de aquecimento e um sensor para monitorar a temperatura da caixa de ligação das resistências na mesa.

As resistências de aquecimento inseridas na mesa são do tipo cartucho, de modo a transmitir a energia térmica para as matrizes instaladas sobre a mesa.

A mesa contém uma chapa de aquecimento montada sobre o centro da mesa onde no interior desta estão inseridas as resistências, e sobre ela ficam as matrizes em contato. A figura 3-12 ilustra a chapa de aquecimento.

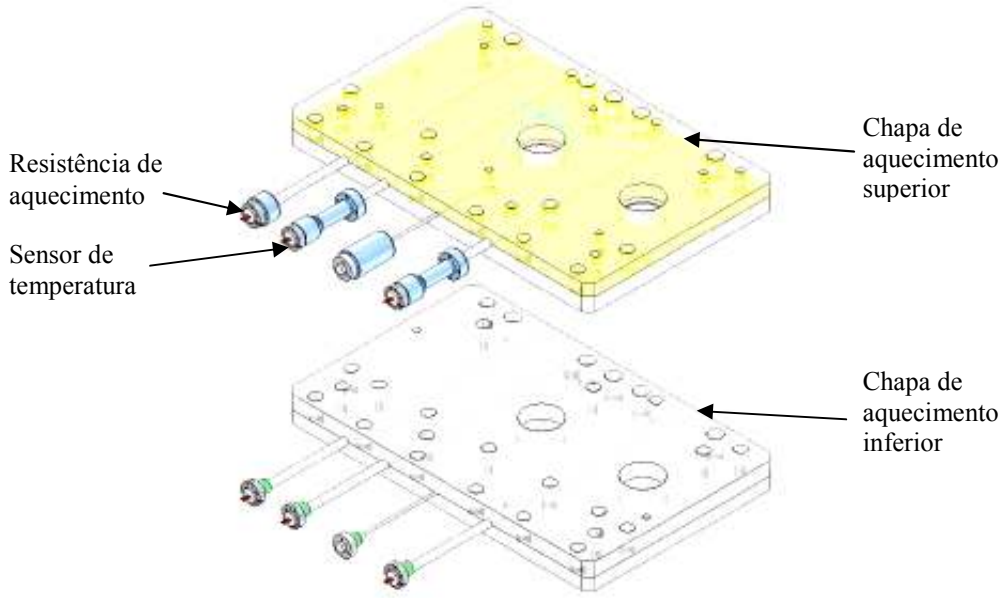


Figura 3-12: conjunto chapa de aquecimento

O conjunto, pode ser considerado duas plantas de controle independentes, sendo que cada uma tem um sistema de aquecimento, um sensor para monitorar a temperatura do aquecimento e outro para monitorar a temperatura da caixa de ligação das resistências.

Na figura 3-13, é mostrada uma vista explodida do conjunto mesa superior e inferior. Cada mesa tem massa aproximada de 3800Kg de aço, sendo o conjunto total de 7600Kg.

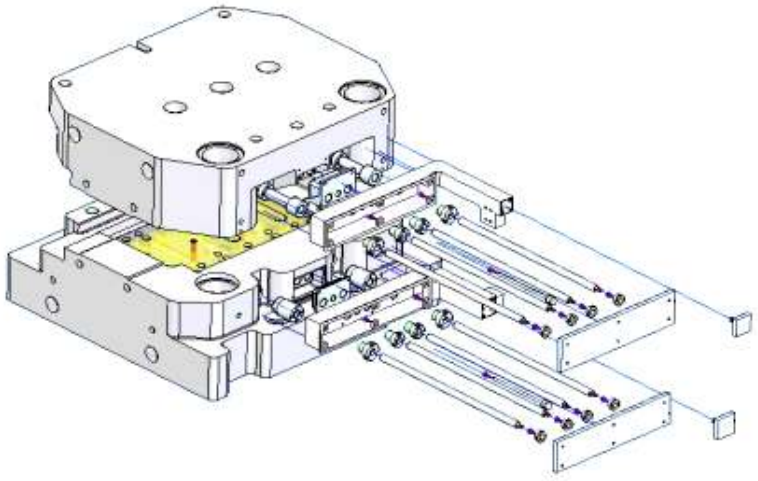


Figura 3-13: Vista explodida da planta de controle. Chapa de aquecimento em amarelo

3.5. *Software*

3.5.1. Compilador

O software foi desenvolvido no ambiente de programação do "M-IDE Studio for MCS-51". O M-IDE é um compilador C, com suporte à biblioteca da família AT89C52. O programa compilado gera o código fonte do processador (.hex).

O ambiente de programação é mostrado na figura 3-14.

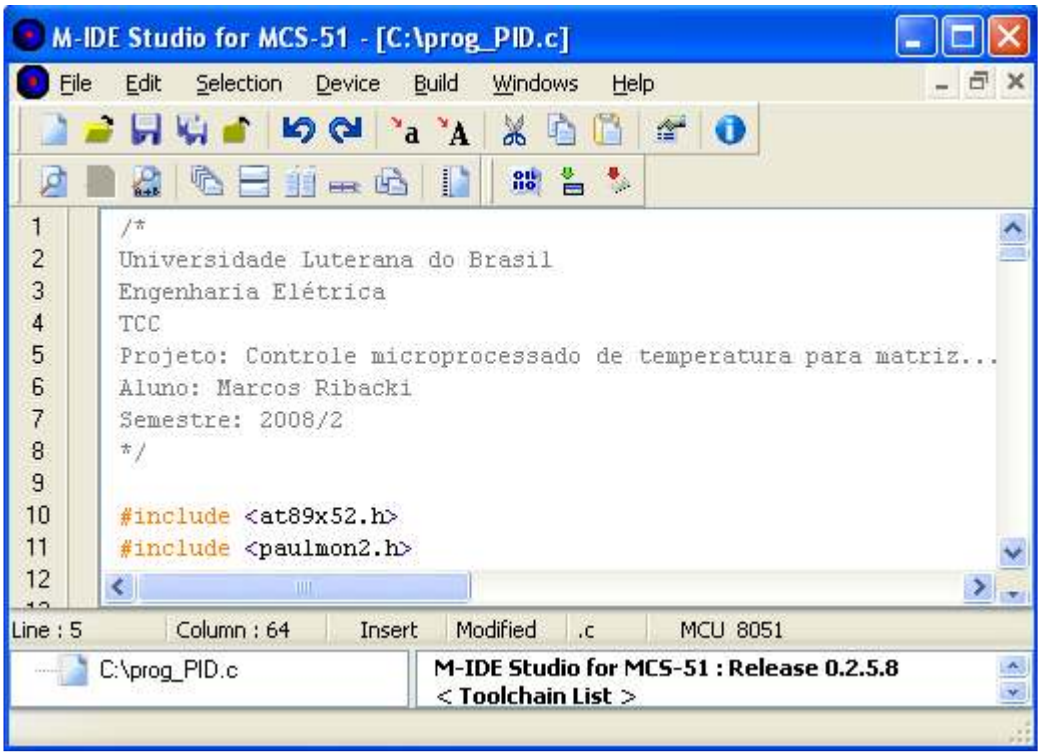


Figura 3-14: Ambiente de programação do MIDE.

3.5.2. Estrutura geral do programa

O programa do microcontrolaor foi desenvolvido com base na complexidade das variáveis do sistema a serem controladas. O sistema precisa monitorar entradas digitais que fornecem status de funcionamento do hardware, entradas analógicas para leitura dos valores de temperatura e saídas para atuação do sistema de controle PWM. Permite ao usuário a visualização dos alarmes, configurar os set-points de temperatura de forma flexível e monitorar as variáveis de temperatura e potência de cada mesa, como sendo as principais variáveis que devem ficar na tela de visualização principal.

Partindo desta necessidade, foi elaborado um fluxograma que ilustra de forma simplificada a navegação nas telas do display, mostrado na figura 3-15.

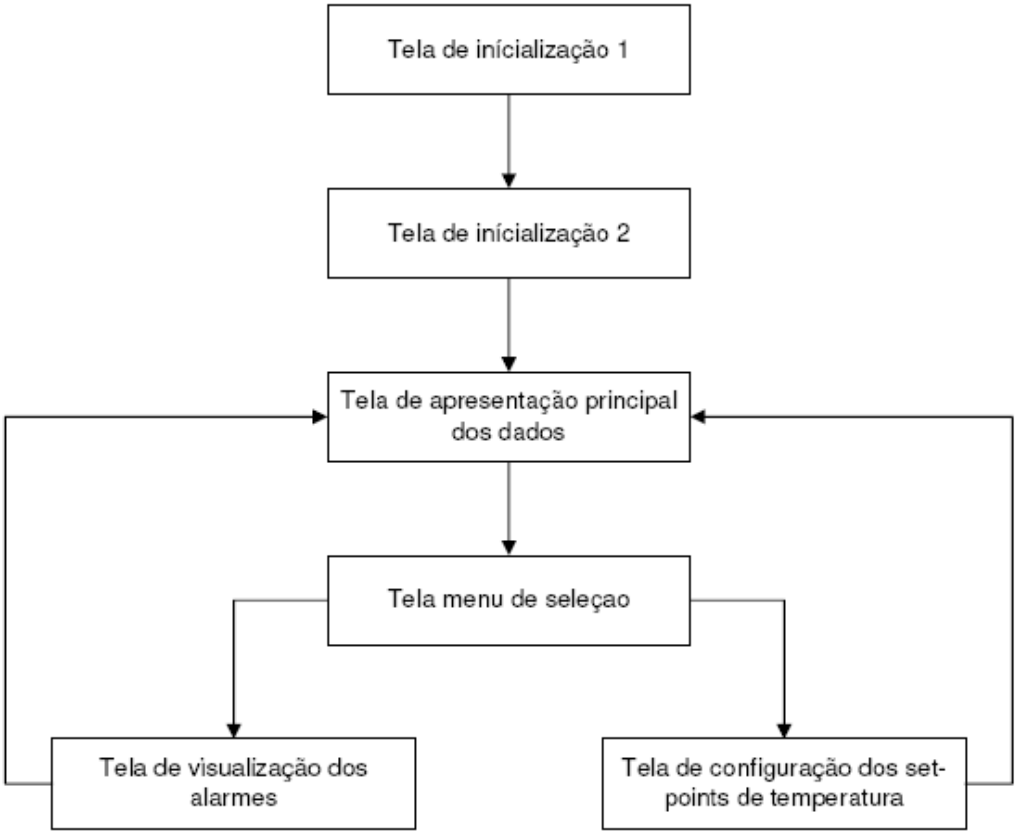


Figura 3-15: Diagrama simplificado de navegação nas telas do display

Tela de inicialização 1

A tela de inícialização 1 é a primeira tela mostrada quando da energização do sistema, nela estão contidas algumas informações do projeto, conforme a figura 3-16.

Figura 3-16: Tela de inicialização 1

Tela de inicialização 2

A tela de inicialização 2 é mostrada após um tempo aproximado de 5 segundos de amostra da tela de inicialização 1. Ela contém alguns dados sobre a execução do projeto.

Figura 3-17: Tela de inicialização 2

Tela de apresentação pricipal dos dados

Durante a operação normal do sistema, a tela de apresentação principal do dados fica ativa, nela, são mostrados os principais dados do sistema, como temperatura da mesa inferior e superior e potência consumida para cada mesa, como pode ser visto na figura 3-18.

Figura 3-18: Tela de apresentação principal dos dados.

Tela menu de seleção

Ainda durante a operação normal do sistema, se o usuário desejar alterar algum valor de set-point ou visualizar os alarmes do sistema, ele pode acessar através do botão ENTER a tela menu que direcionará para as telas de alarmes ou configuração de set-ponts conforme as instruções nela contidas, como mostra a figura 3-19.

Figura 3-19: tela menu de seleção

Tela de visualização dos alarmes

Partindo da tela menu, ou então da primeira ocorrência de alguma alarme, a tela mostrada é a tela de visualização dos alarmes, nela podem ser visualizados os 2 primeiros alarmes que ocorreram e o último. A figura 3-20 mostra a ocorrência de 2 alarmes simultaneamente de forma simulada. A tela de apresentação principal dos dados é mostrada automaticamente após 5 segundos.

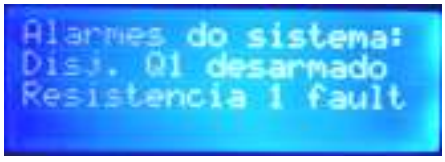


Figura 3-20: Tela de visualização dos alarmes

Tela de configuração dos parâmetros.

Do mesmo modo que partindo da tela menu para a tela de alarmes, pode-se partir para a tela de configuração dos parâmetros de temperatura. A figura 3-21 mostra o primeiro set-point de temperatura, onde mais três set-points podem ser

acessados através da navegação UP & DOWN. Estando na tela do parâmetro desejado, para alterar, basta pressionar uma vez o botão ENTER, para então os botões de UP & DOWN passarem a incrementar ou decrementar o valor do set-point dentro da faixa permitida que vai de 0 a 400°C. Para salvar o valor e retornar para a navegação de parâmetros, basta pressionar o botão ENTER novamente. Para retornar à tela de apresentação principal dos dados, aguardar aproximadamente 5 segundos sem pressionar nenhum botão.

Figura 3-21: Tela de configuração dos set-points de temperatura

4. IMPLEMENTAÇÃO DO SOFTWARE DE CONTROLE PID E RESULTADOS

Para implementação do software de controle, é necessário conhecer a planta a ser controlada.

Tendo em vista a necessidade da validação do software, o sistema foi testado previamente em um protótipo com constantes de tempo mais rápidas.

O sistema completo demora em torno de 3 horas para estabilizar, isto tornaria inviável em termos de tempo os testes.

Para testar, utilizou-se uma base de aquecimento de ferro elétrico comum, pois trata-se de uma massa aquecida em alguns pontos e com grande área para dissipar o calor.

Para o conhecimento da resposta da planta, foi aplicado uma excitação do tipo degrau, e coletado a curva de aquecimento do protótipo, a qual pode ser vista na figura 4-1.

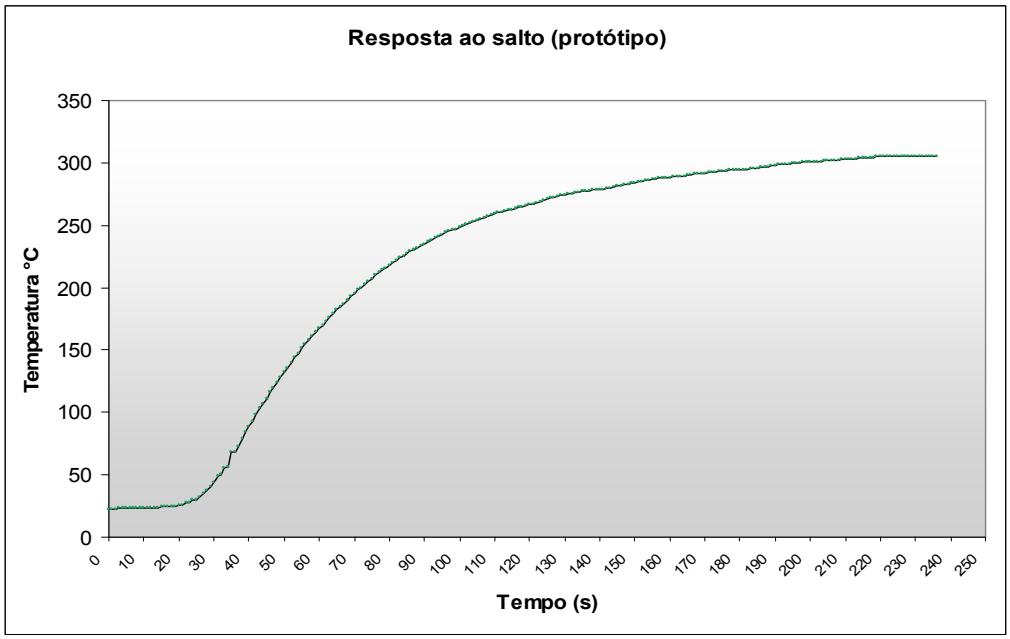


Figura 4-1: Gráfico de aquecimento do sistema, resposta ao salto, com 25% de potência

A partir da curva de aquecimento, resposta ao degrau, foi aplicado o método da resposta ao salto de Ziegler-Nichols, para a obtenção dos parâmetros do controlador PID, pelo seu comportamento atender a característica da curva em "S", como exige este método.

4.1. Obtenção dos parâmetros do controlador

Sobre a curva do sistema é traçada a reta tangente, no ponto de inflexão conforme mostra a figura 4-2.

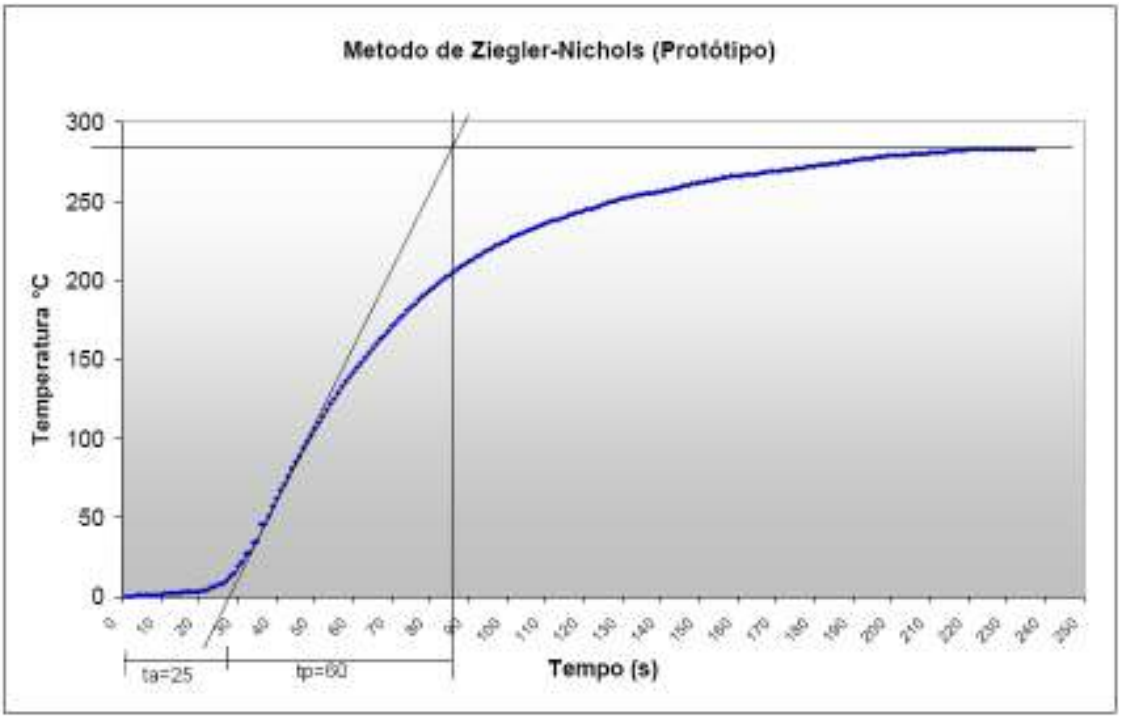


Figura 4-2: Reta tangente ao ponto de inflexão.

τa = 25 (tempo de retardo)

Tp= 60 (constante de tempo)

Com base nestes dados, pode-se obter os valores de Kp, Ti, e Td conforme segue:

Para o controle PID:

$$G_c(s) = K_p(1 + \frac{1}{T_i s} + T_d s)$$

[Equação - 4.1]

$$Kp = 1{,}2.\frac{Tp}{\tau a} = 1{,}2.\frac{60}{25} = 2{,}9$$

[Equação - 4.2]

$$Ti = 2.\tau a = 2.25 = 50$$

[Equação - 4.3]

$$Td = 0{,}5.\tau a = 0{,}5.25 = 12{,}5$$

[Equação - 4.4]

Substituindo os valores na equação 4.1, obtém-se a função de malha aberta.

$$Gc(s) = 2{,}9.(1 + \frac{1}{19.s} + 4{,}75.s) = \frac{36{,}25.s^2 + 2{,}9.s + 0{,}027}{s}$$

[Equação - 4.5]

$$Ki = \frac{Kp}{Ti} = \frac{2{,}9}{50} = 0{,}058$$

[Equação - 4.6]

$$Kd = Kp.Td = 2{,}9.12{,}5 = 36{,}25$$

[equação - 4.7]

A partir da equação do PID, obtém-se o sinal de atuação $a(t)$ através da equação 4.8,

$$a(t) = Kp.e(t) + Ki.\int_0^t e(t).dt + Kd.\frac{d.e(t)}{dt}$$

[Equação - 4.8]

Onde:
$e(t)$ = Erro entre a saída e a referência
Kp = Constante proporcional
Kd = Constante derivativa
Ki = Constante integral

A discretização é obtida substituindo a variável t por:

$$t = n.Ts$$

Ts = Período de amostragem
n = Numero da atuação.

A equação discretizada será então:

$$a(nTs) = Kp.e(nTs) + Ki.Ts\sum_{j=0}^{n} e(j.Ts) + Kd.\frac{e((n-1)Ts)}{Ts}$$

[Equação - 4.9]

Como $n.Ts$ representa o número da atuação, pode ser substituída simplesmente por n **:**

$$a(n) = Kp.e(n) + Ki.Ts\sum_{j=0}^{n} e(j) + Kd.\frac{e(n) - e(n-1)}{Ts}$$

[Equação - 4.10]

$$a(n) = (Kp + \frac{Kd}{Ts}).e(n) + \frac{Kd}{Ts}.e(n-1) + Ki.Ts.\sum_{j=0}^{n} e(j)$$

[Equação - 4.11]

Fazendo:

$$a(n-1) = (Kp + \frac{Kd}{Ts}).e(n-1) + \frac{Kd}{Ts}.e(n-2) + Ki.Ts.\sum_{j=0}^{n-1} e(j)$$

[Equação - 4.12]

E calculando:

$$a(n) - a(n-1) = (Kp + \frac{Kd}{Ts}).e(n) + \frac{Kd}{Ts}.e(n-1) + Ki.Ts.\sum_{j=0}^{n} e(j)$$
$$-(Kp + \frac{Kd}{Ts}).e(n-1) + \frac{Kd}{Ts}.e(n-2) + Ki.Ts.\sum_{j=0}^{n-1} e(j)$$

[Equação - 4.13]

Considerando:

$$\sum_{j=0}^{n} e(j) - \sum_{j=0}^{n-1} e(j) = e(n)$$

[Equação - 4.14]

Obtêm-se uma outra formula de implementação do controlador digital:

$$a(n) = a(n-1) + (Kp + \frac{Kd}{Ts} + Ki.Ts).e(n) + (-Kp - 2.\frac{Kd}{Ts}).e(n-1) + (\frac{Kd}{Ts}).e(n-2)$$

[Equação - 4.15]

4.2. Desenvolvimento do programa de controle PID

4.2.1. Fluxograma:

A estrutura do programa de atendimento a interrupção do microcontrolador e controle do PWM, que é responsável pela execução da rotina do controlador PID e atuação nos relés de estado sólido para controle da potência de aquecimento, é mostrada na figura 4-3. O fluxograma completo do programa está no apêndice B.

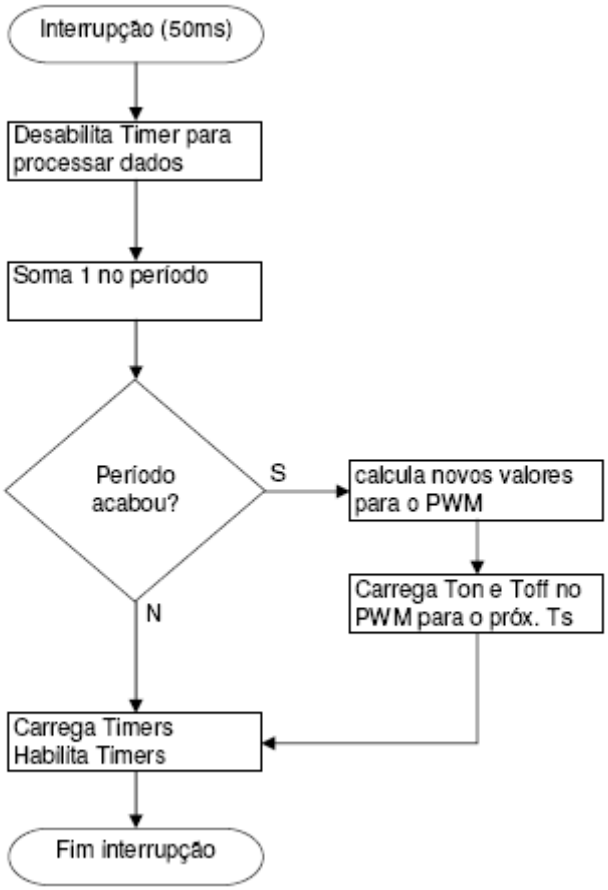


Figura 4-3: Estrutura da rotina do programa de atendimento a interrupção

Interrupção 50ms: A cada 50ms é gerada uma interrupção, em função do timer 0 estar parametrizado para esta base de tempo, deste modo é possível gerar um tempo regular de amostragem para o controlador PID.

Desabilita timer para processar dados: Durante o processamento da coleta de dados da ação de controle para o próximo período de atuação da saída PWM, o timer é desabilitado.

Soma 1 no período: a cada interrupção gerada (50ms), o programa vai passar por esta variável que soma 1 no seu acumulador, isto serve de referência para o período do PWM, que é de 2 segundos, não é possível fazer uma interrupção com este tempo, já que o limite do processador é de 64536 ciclos de processamento onde cada ciclo demora 1 μs , logo, o tempo máximo é de 64,5 ms.

Período acabou ?: verifica se o acumulador contém o valor correspondente ao final do período e direciona o programa para obtenção de novo valor de controle a parametrização do PWM, se o período não acabou, apenas direciona para carregar o timer novamente e iniciar nova contagem.

Calcula novos valores para o PWM: executa a rotina do PID que será mais detalhada no item 4.2.4.

Carrega Ton e Toff no PWM para o próx. Ts: Com base na nova ação de controle obtida na rotina PID, são carregados os novos valores de Ton e Toff para o PWM durante o próximo período.

Carreta timer / Habilita timers: ativa novamente o timer para inciar uma nova contagem de 50ms.

4.2.2. Rotina de controle do PWM

A rotina *controle_tempo_pwm*, simplesmente atualiza o valor de atuação com base na saída do controle PID, que é chamada na linha 220 conforme pode ser visto na figura 4-4. A seguir detalha-se a mesma.

```
210  void controle_tempo_pwm (void) interrupt 1
211  {
212
213          TR0 = 0;
214          Tempo_avalia_PWM++;
215          if(Tempo_avalia_PWM >= 2)
216          {
217                  T_inf++;
218                  if(T_inf >= 20)
219                  {
220                          controle_PID ();
221                          inf_Ton = inf_Vc;
222                          T_inf = 0;
223                  }
224                  if(inf_Ton >= 0)
225                  {
226                          PWM_MAT_INF = 0;
227                          inf_Ton--;
228                  }
229                  if(inf_Ton < 0)
230                  {
231                          PWM_MAT_INF = 1;
232                  }
233                  Tempo_avalia_PWM = 0;
234          }
235          TL0 = 0x0A;
236          TH0 = 0x3C;
237          TR0 = 1;
238  }
```

Figura 4-4: Rotina de controle da potência

4.2.3. Controlador PID

A rotina controlador PID quando solicitada pelo programa de controle do PWM, avalia o estado do erro atual e calcula a saída de controle para o próximo período em função da equação discreta do PID. A rotina é apresentada na figura 4-5 e foi implementada com base na equação 4.15, que foi obtida através do método da resposta ao salto de Ziegler-Nichols, descrita no capitulo 3.

```
240  void controle_PID (void)
241  {
242          int q0,q1,q2;
243          inf_erro = temperatura[1] - valor_AN0;
244          q0 = inf_Kp + (inf_Kd/inf_Ts) + inf_Ki*inf_Ts;
245          q1 = -inf_Kp - 2*(inf_Kd/inf_Ts);
246          q2 = inf_Kd/inf_Ts;
247          inf_Vc = inf_Vc_ant + q0*inf_erro + q1*inf_erro_ant_1 + q2*inf_erro_ant_2;
248          inf_erro_ant_2 = inf_erro_ant_1;
249          inf_erro_ant_1 = inf_erro;
250          inf_Vc_ant = inf_Vc;
251  }
```

Figura 4-5: Rotina de controle PID.

4.3. Resultados com o protótipo.

Como o desenvolvimento do projeto foi em duas etapas, primeiramente no protótipo de aquecimento e controle, e posteriormente na planta final de controle, os dados são apresentados em duas formas distintas.

O testes práticos com o protótipo para os parâmetros calculados de Kp=2,9, Kd = 36,25 e Ki=0,058, que foram obtidos no item 4.1, apresentaram uma resposta um pouco lenta, uma tentativa de melhora foi feita baixando o ganho derivativo para Kd=15, mas o overshoot ficou elevado. A resposta satisfatória, foi obtida com os valores de Kp e Ki brevemente alterados sendo 3 e 0,1 os novos valores respectivamente.

As figuras 4-6, 4-7 e 4-8 ilustram os resultados obtidos com o protótipo.

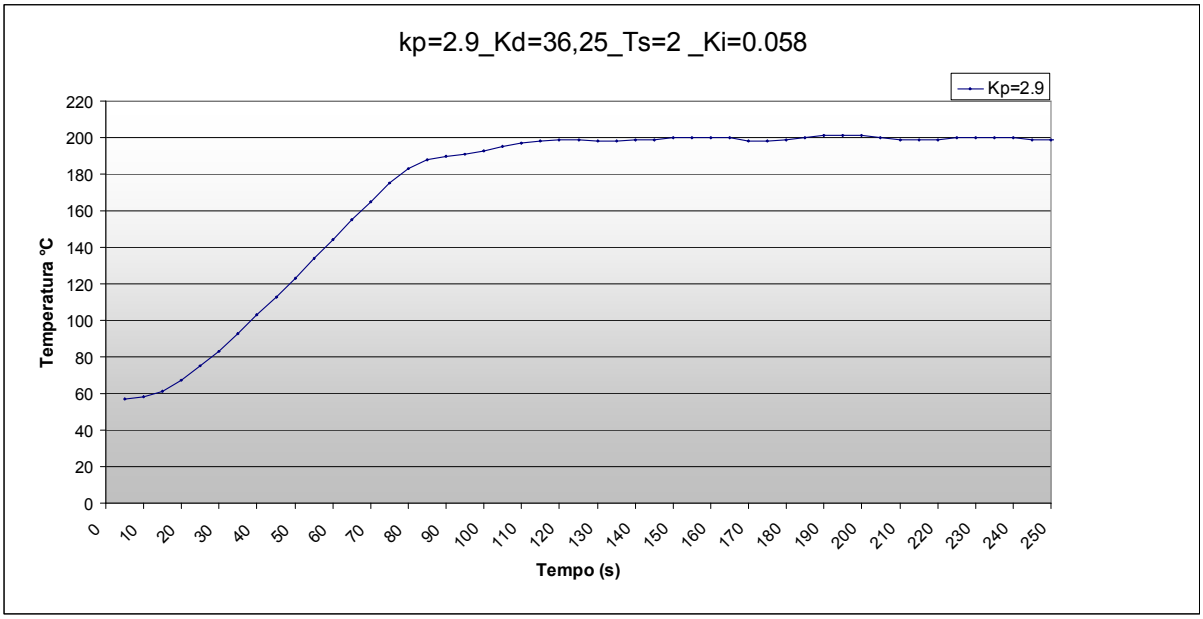


Figura 4-6: Resposta da planta protótipo com os valores calculados.

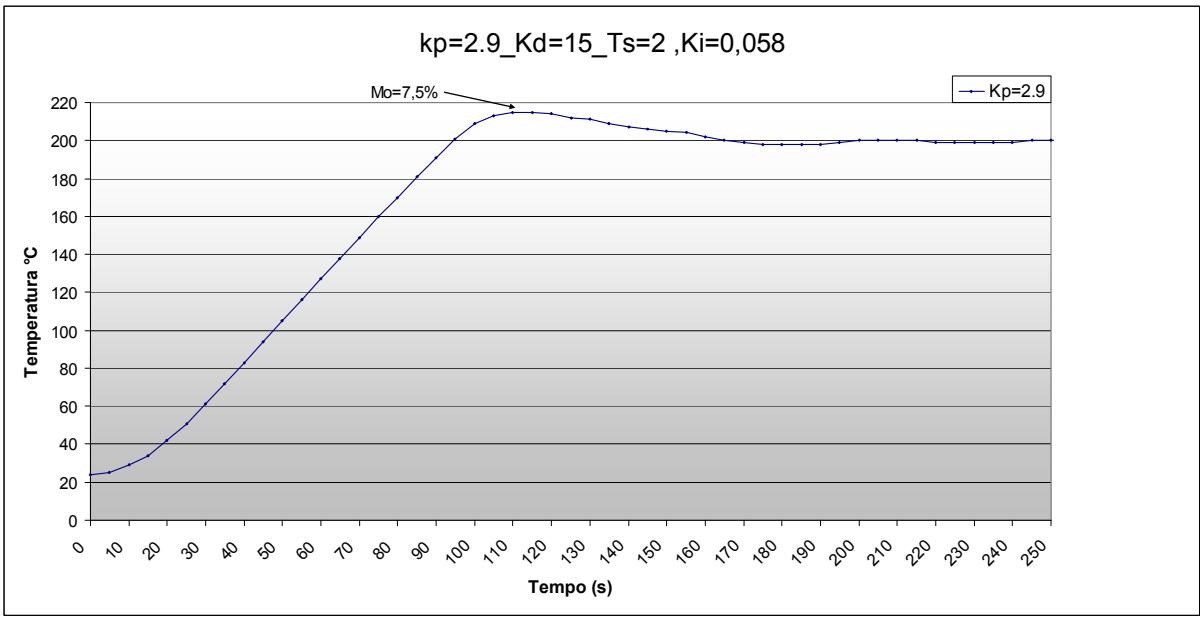


Figura 4-7: Resposta da planta protótipo com alteração do valor de Kd.

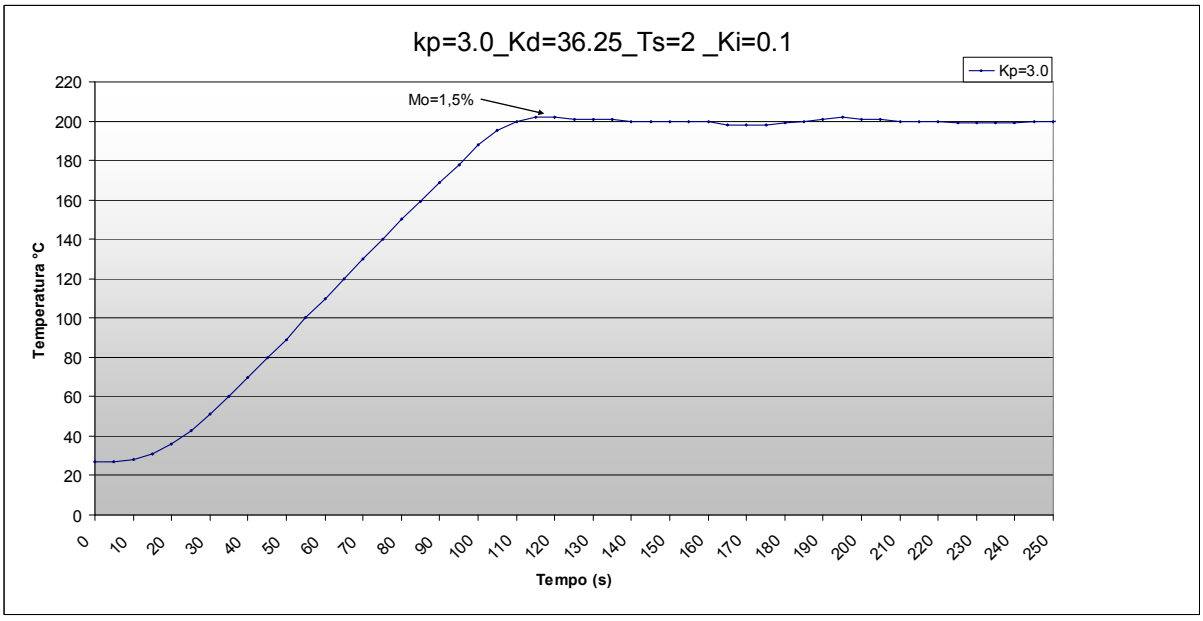


Figura 4-8: Resposta da planta protótipo com os novos valores de Kp e Ki.

4.4. Resultados com a planta final (sistema com as matrizes)

A implantação do sistema final aconteceu após o ajuste de todos os controles através do protótipo e comprovação dos resultados. Deste modo, após a montagem do sistema final, a planta foi submetida a resposta ao salto, para novo cálculo dos parâmetros do controlador PID. Como a resposta do sistema é lenta,

não foi possível a comprovação dos parâmetros do controlador PID. A figura 4-9 mostra a eficiência do aquecimento elétrico do sistema através de uma imagem termográfica, e na figura 4-10, tem-se uma foto da planta de controle no momento em que foi obtida a imagem termográfica. Salienta-se a pouca variação de temperatura em toda a matriz.

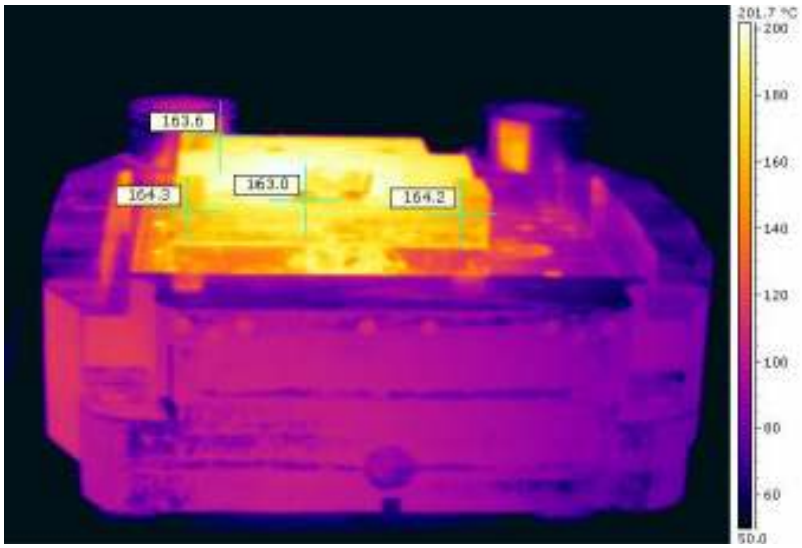


Figura 4-9: Imagem termográfica do sistema aquecido

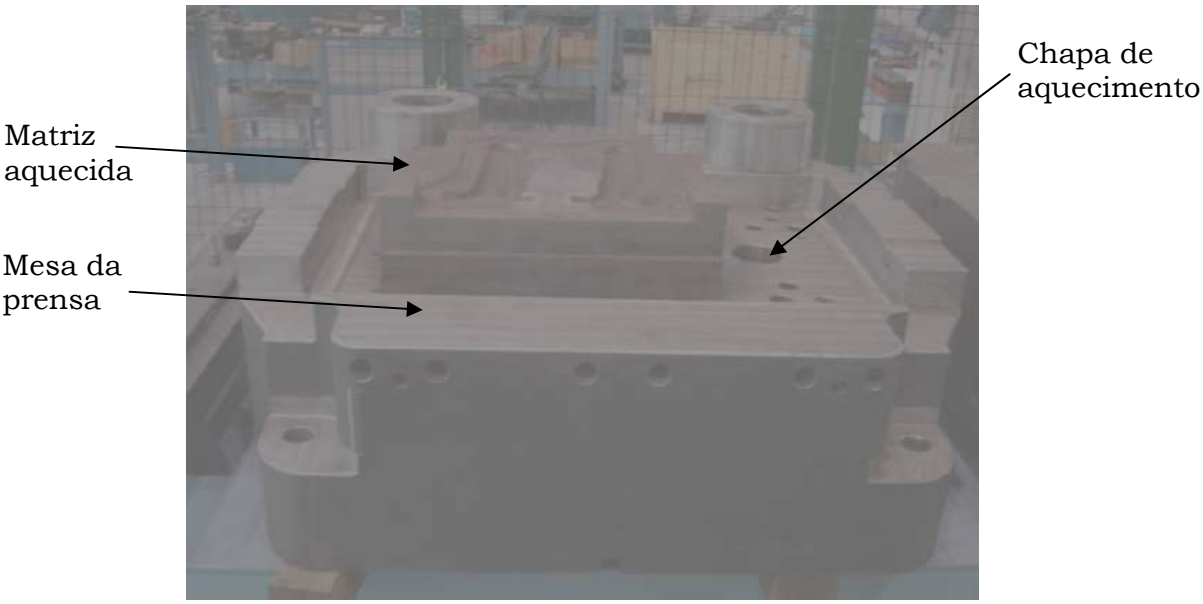


Figura 4-10: Imagem da planta de controle.

A seguir mostra-se na figura 4-11 o resultado da aplicação de um salto unitário de 100% de potência na matriz. Com isso, pode-se obter os parâmetros Kp, Ki e Kd, que ainda não foram implementados.

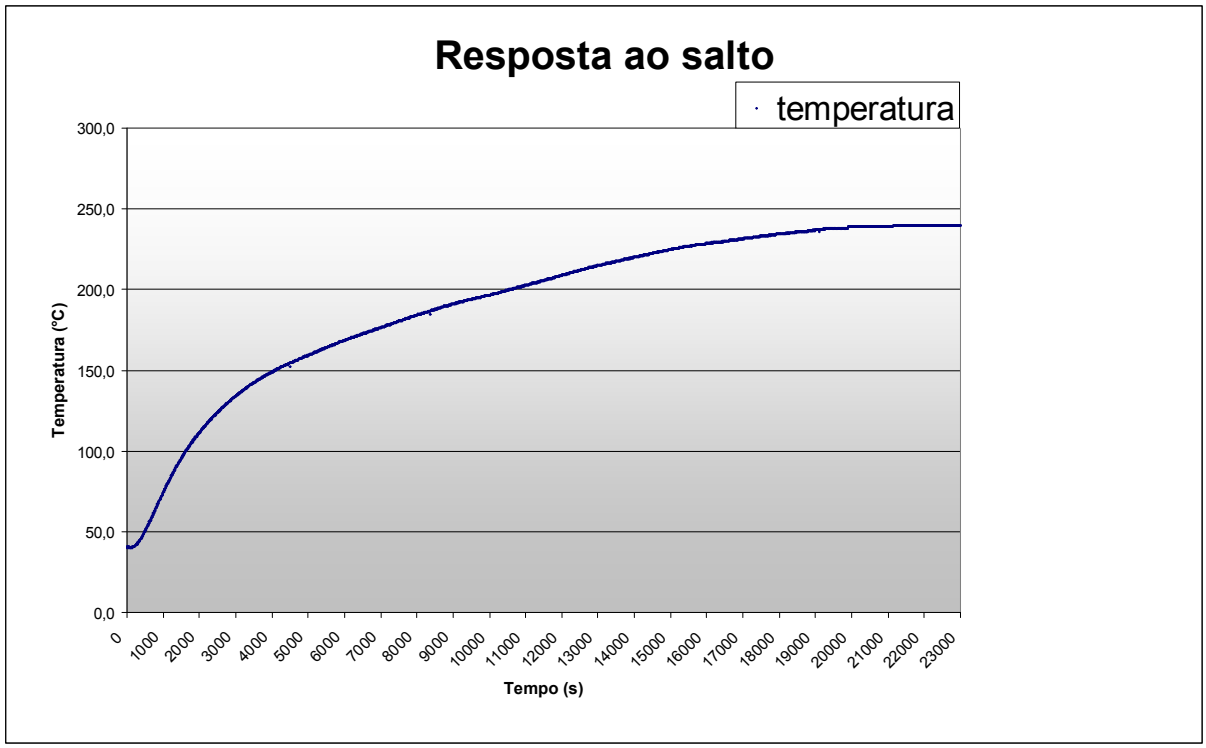


Figura 4-11: Resposta ao salto matriz.

5. CONSIDERAÇÕES FINAIS

O objetivo deste trabalho foi desenvolver um sistema de controle automático de temperatura, através de um sistema PID, incluindo o monitoramento das variáveis de feedback do hardware, num sistema flexível a acessível ao usuário.

A partir do protótipo de controle onde os testes foram realizados, foi comprovado o funcionamento do sistema de monitoramento das variáveis de controle PID. Os resultados obtidos do sistema, sujeito a ação de controle calculada, que embora não apresentou um resultado satisfatório na primeira tentativa, podem ser justificados pela obtenção pouco precisa dos parâmetros do controlador através do método de Ziegler Nichols. Contudo, com uma pequena variação do valor de Kp e Ki se conseguiu um resultado excelente para a aplicação.

Frente aos objetivos propostos, obteve-se um *software* eficiente para o controle de temperatura através do PID digital. O sistema controlou a temperatura de forma eficiente com $Mo \leq 5\%$.

O funcionamento da interface com o usuário através do display ficou conforme o que era necessário para a aplicação do sistema no processo produtivo.

Para dar continuidade à proposta deste trabalho, propõe-se a continuação dos testes de controle da planta final e implantação final do sistema, além de testes exaustivos de operabilidade com os usuários finais.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] BAZANELLA, ALEXANDRE SANFELICE ; SILVA, JOÃO MANOEL DA. - **Ajuste de Controladores PID**. - www.ece.ufrgs.br/~jmgomes/pid/Apostila/apostila/.

[2] ALEXANDRE BALBINOT & VALNER BRUSAMARELLO. - **Instrumentação e Fundamentos de Medida**. - Vol.1, 1º ed, 2006, LTC.

[3] OGATA, KATSUHIKO. - **Engenharia de Controle Moderno.** - Editora Prentice-Hall do Brasil – RJ – 1998.

[4] HANG, C. C. ; ASTRÖM, K. J.; HO, W. K. - **Refinements of the Ziegler–Nichols tuning formula.** - Proc Inst. Elect. Eng.—Part D: Control Theory and Applications, vol. 138, no. 2, 1991.

[5] Novus Produtos Eletrônica, www.novus.com.br.

[6] Siemens, www.siemens.com.br.

[7] SEBORG, D. E. ; EDGAR, T. F.; MELLICHAMP, D. A. - **Process dynamics and control.**- New York: John Wiley & Sons, 1989

OBRAS CONSULTADAS

1 - PRUDENTE, FRANCESCO. – **Automação Industrial.** - Editora LTC – RJ – 2007.

2 – CARRO, LUIGI. – **Projeto e prototipação de sistemas digitais**. – Ed. Universidade/ UFRGS – 2001.

GLOSSÁRIO

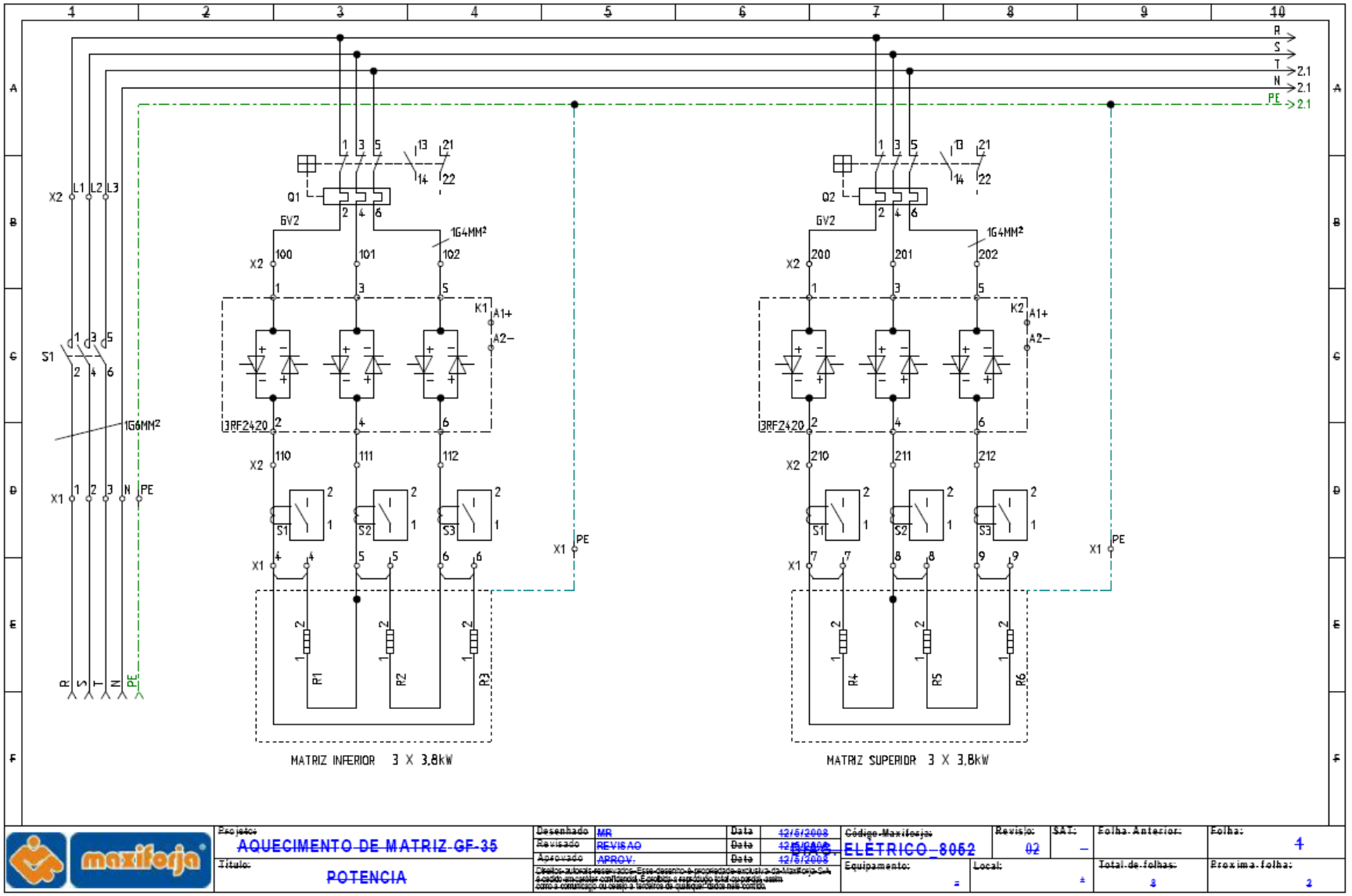
Matriz/Ferramenta: Ferramenta utilizada no forjamento para conformar o material deixando-o com a geometria final da peça.

Desgaste: Dano progressivo nas ferramentas através do movimento relativo entre suas superfícies.

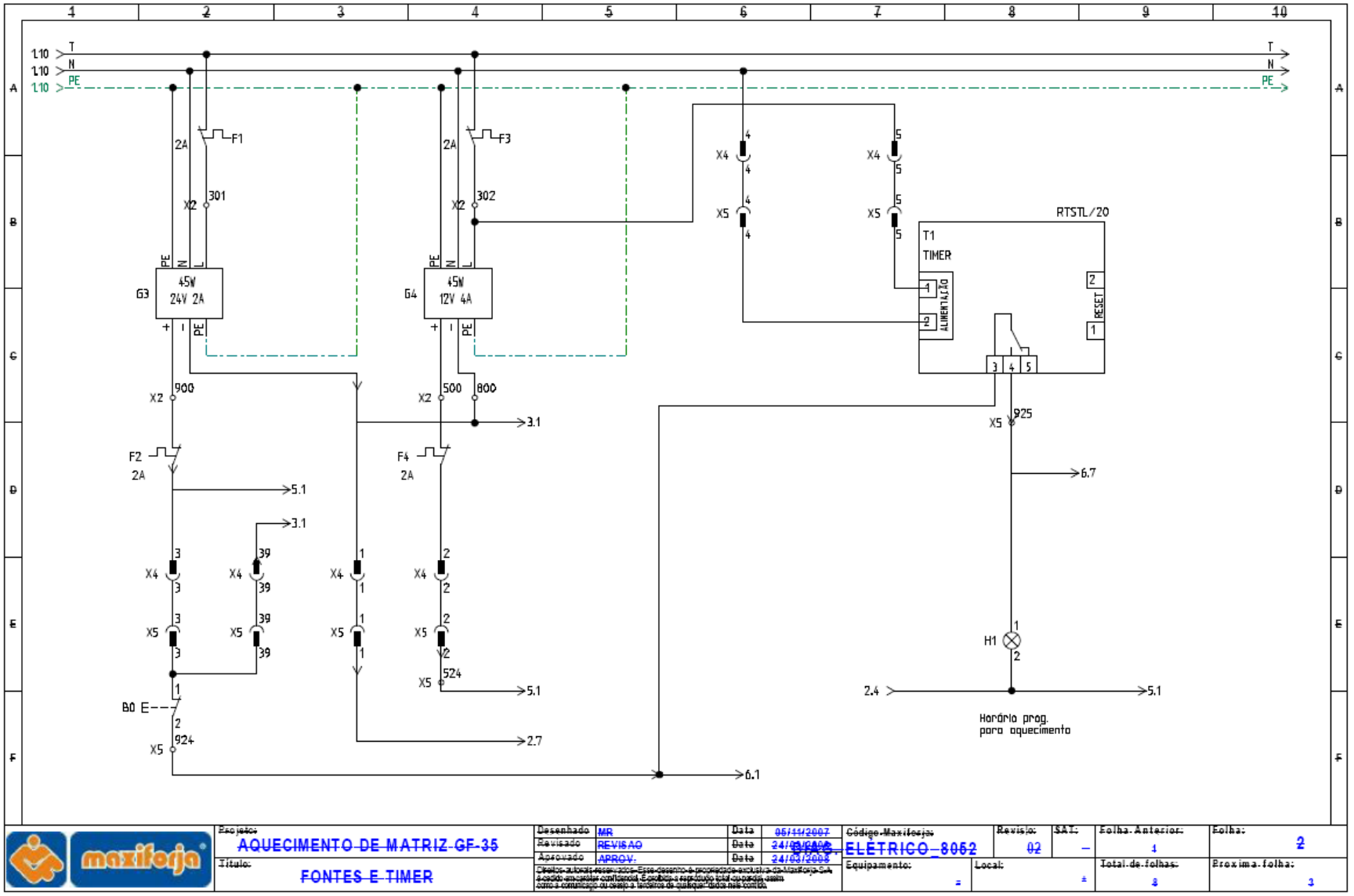
Tenacidade: Energia que o material absorve antes de fraturar.

Forjamento: É o nome genérico de operações de conformação mecânica efetuadas com esforços de compressão sobre o material, de tal modo que ele tende a assumir o perfil da ferramenta de trabalho.

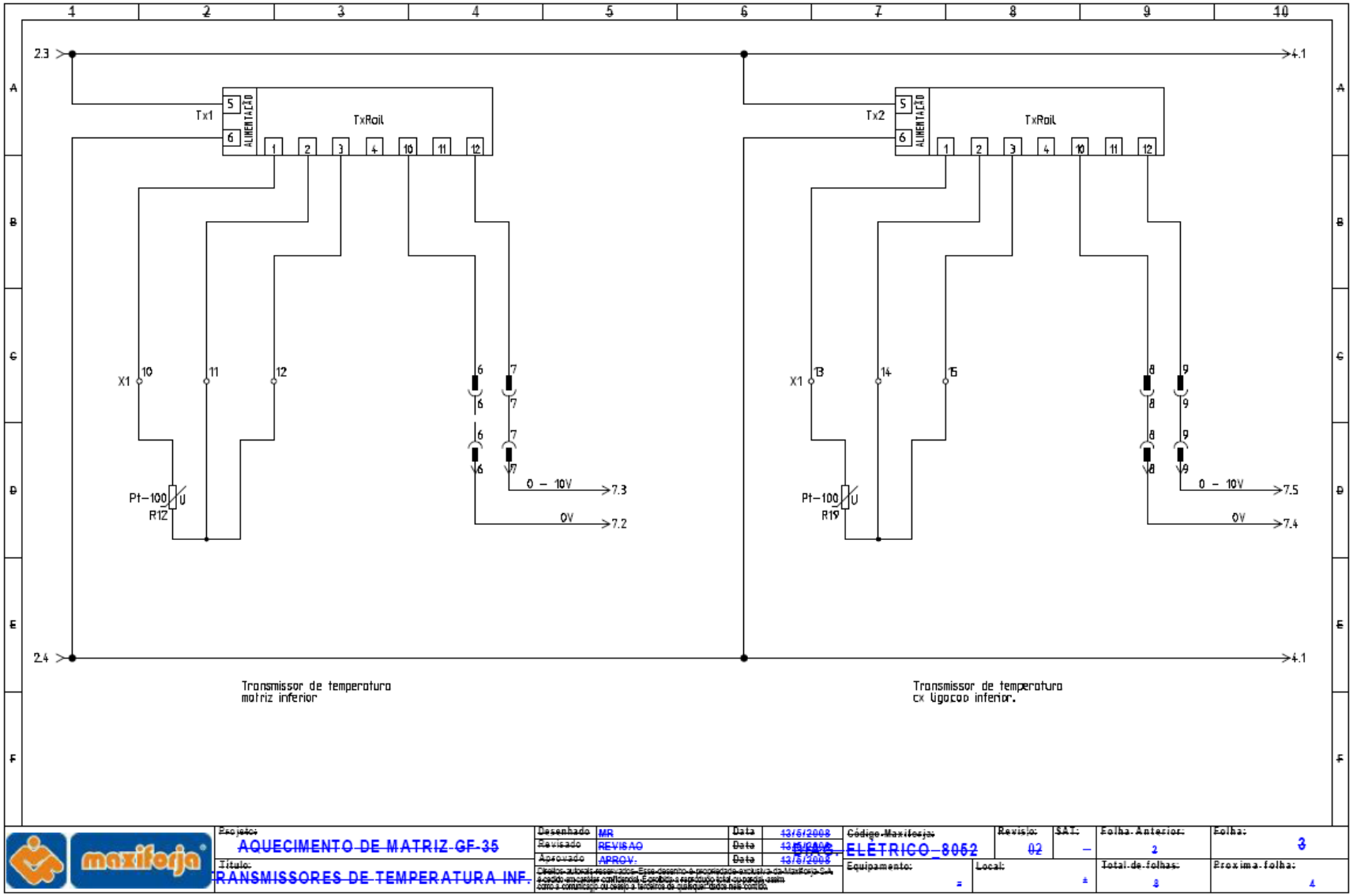
APÊNDICE A – DIAGRAMA ELÉTRICO

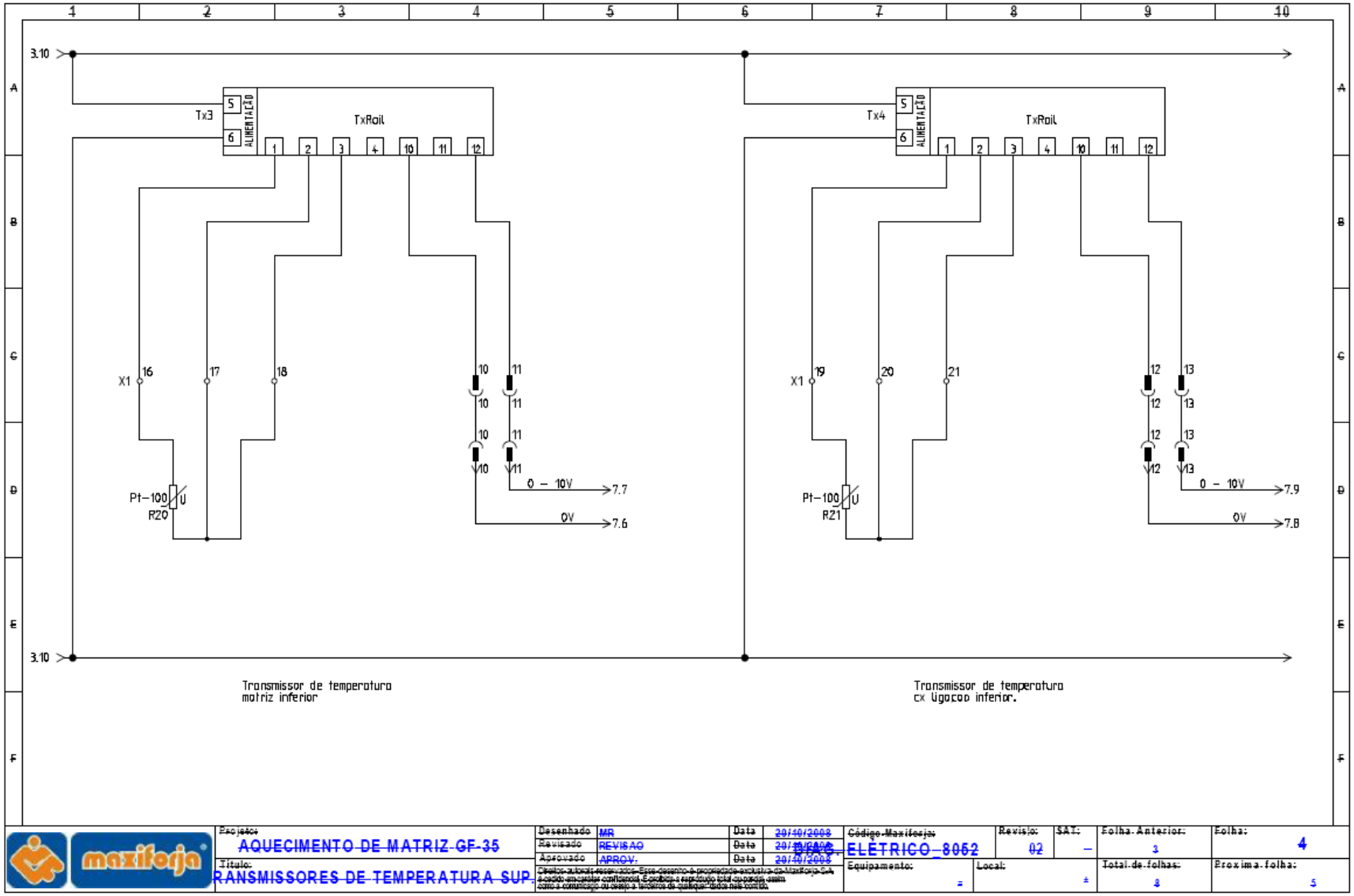


MARCOS HENRIQUE RIBACKI
Universidade Luterana do Brasil

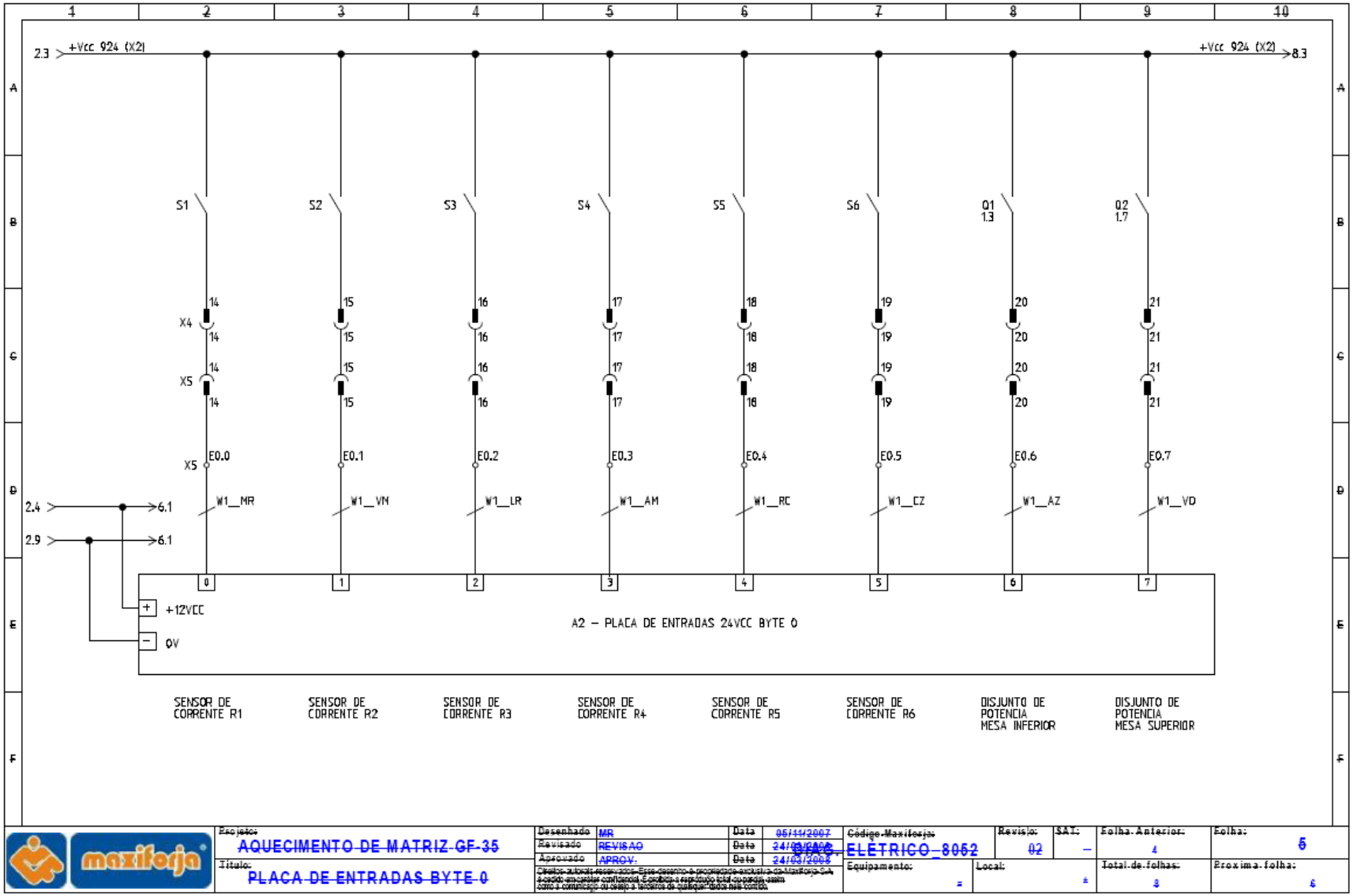


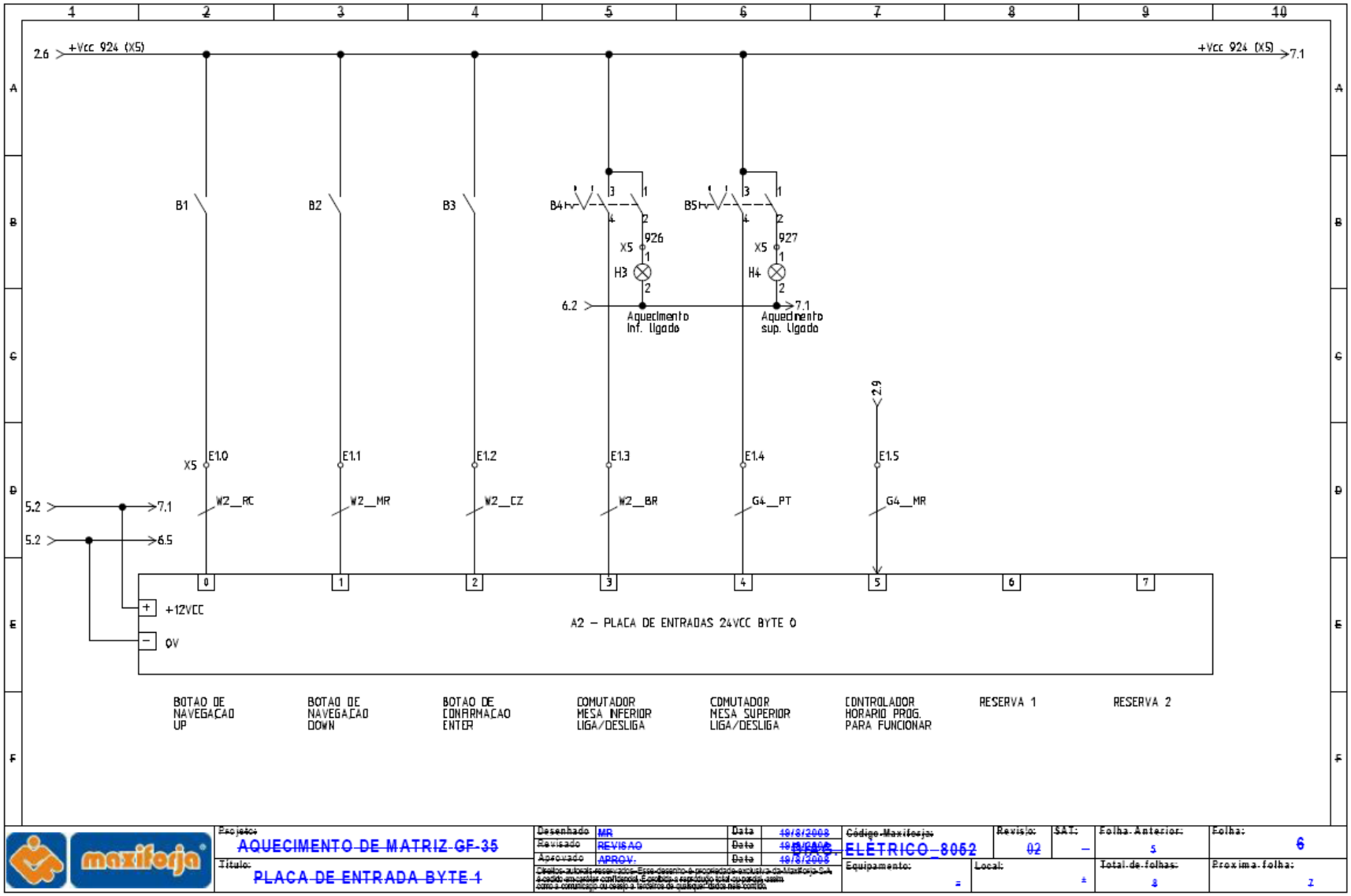
MARCOS HENRIQUE RIBACKI
Universidade Luterana do Brasil


2.3
2.4
4.1
Tx1
Tx2
TxRail
ALIMENTAÇÃO
X1
Pt-100
R12
R19
0 - 10V
0V
7.3
7.2
7.5
7.4
Transmissor de temperatura
matriz inferior
Transmissor de temperatura
cx ligacao inferior.
AQUECIMENTO DE MATRIZ GF-35
TRANSMISSORES DE TEMPERATURA INF.
ELETRICO_8052
maxiforja

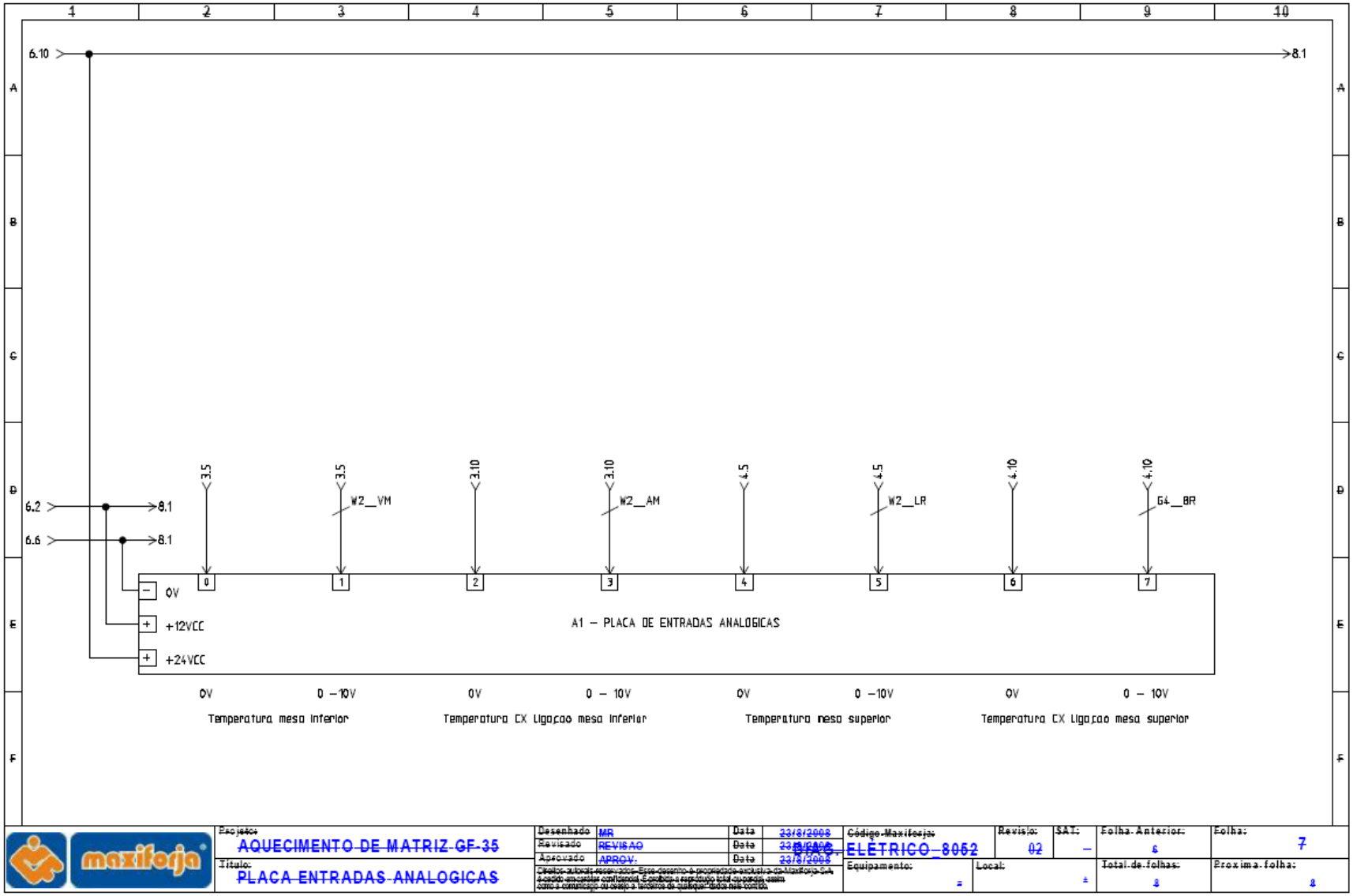


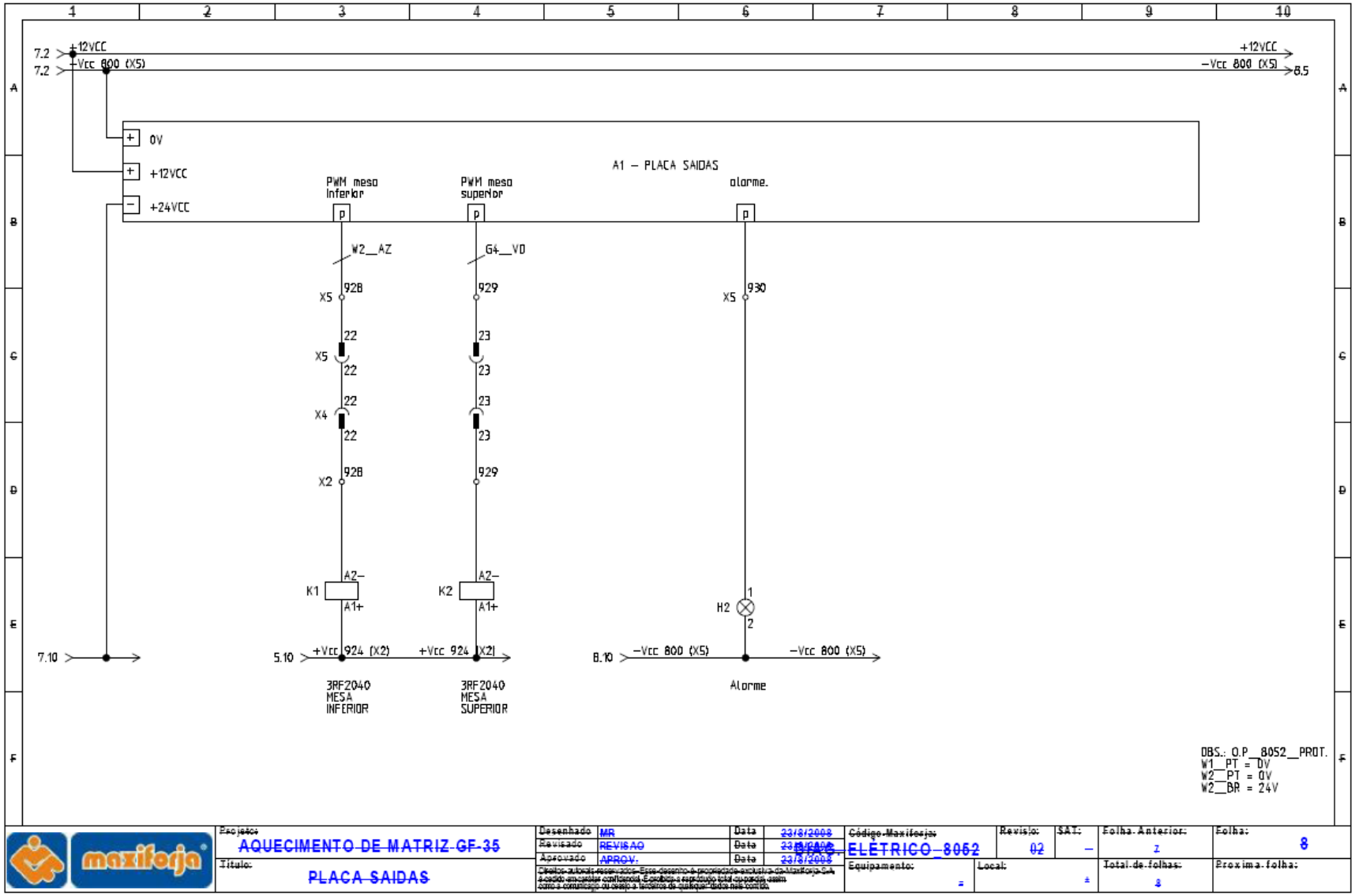
MARCOS HENRIQUE RIBACKI
Universidade Luterana do Brasil


+Vcc 924 (X2)
+Vcc 924 (X2)
S1
S2
S3
S4
S5
S6
Q1
Q2
X4
X5
E0.0
E0.1
E0.2
E0.3
E0.4
E0.5
E0.6
E0.7
W1__MR
W1__VM
W1__LR
W1__AM
W1__RC
W1__CZ
W1__AZ
W1__VD
+12VCC
0V
A2 – PLACA DE ENTRADAS 24VCC BYTE 0
SENSOR DE CORRENTE R1
SENSOR DE CORRENTE R2
SENSOR DE CORRENTE R3
SENSOR DE CORRENTE R4
SENSOR DE CORRENTE R5
SENSOR DE CORRENTE R6
DISJUNTO DE POTENCIA MESA INFERIOR
DISJUNTO DE POTENCIA MESA SUPERIOR
AQUECIMENTO DE MATRIZ GF-35
PLACA DE ENTRADAS BYTE 0
ELETRICO_8052



MARCOS HENRIQUE RIBACKI
Universidade Luterana do Brasil

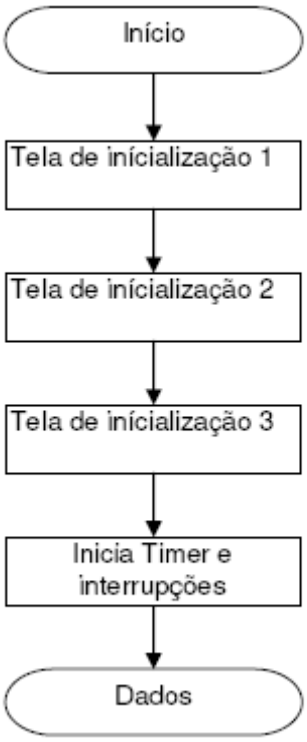

6.10
8.1
6.2
8.1
6.6
8.1
3.5
W2__VM
3.10
W2__AM
4.5
W2__LR
4.10
G4__BR
0V
+12VCC
+24VCC
A1 – PLACA DE ENTRADAS ANALOGICAS
0V
0 – 10V
Temperatura mesa inferior
0V
0 – 10V
Temperatura CX Ligação mesa inferior
0V
0 – 10V
Temperatura mesa superior
0V
0 – 10V
Temperatura CX Ligação mesa superior
maxiforja
Projeto: AQUECIMENTO DE MATRIZ GF-35
Título: PLACA ENTRADAS ANALOGICAS
Desenhado MR Data 23/8/2008
Revisado REVISAO Data 23/8/2008
Aprovado APROV. Data 23/8/2008
Código Maxiforja: ELETRICO_8052
Revisão: 02
SAT: –
Folha Anterior: 6
Folha: 7
Equipamento:
Local:
Total de folhas: 8
Proxima folha: 8



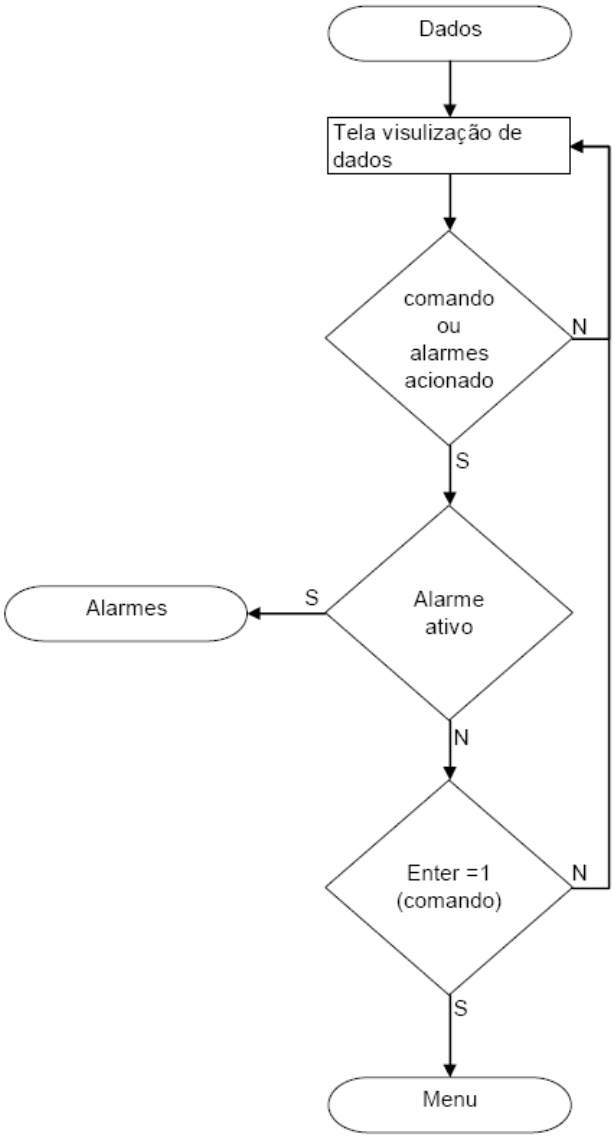
MARCOS HENRIQUE RIBACKI
Universidade Luterana do Brasil

APÊNDICE B – FLUXOGRAMA DO PROGRAMA

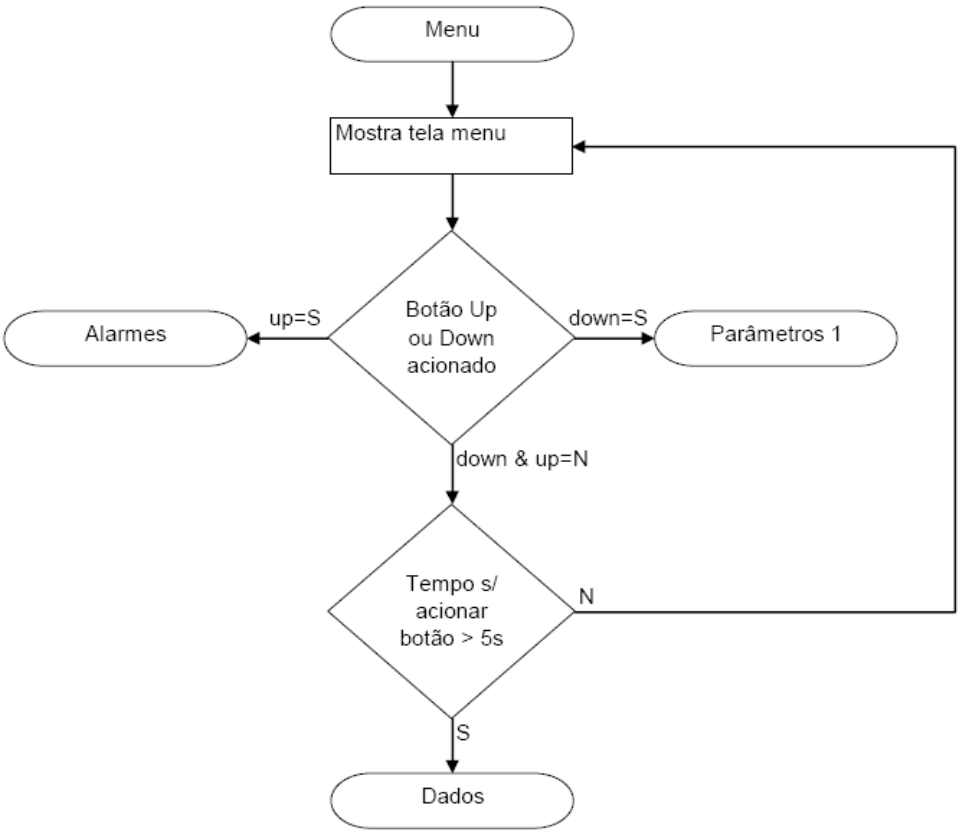
Telas de inicialização.



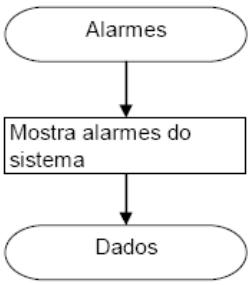
Apresentação principal dos dados.



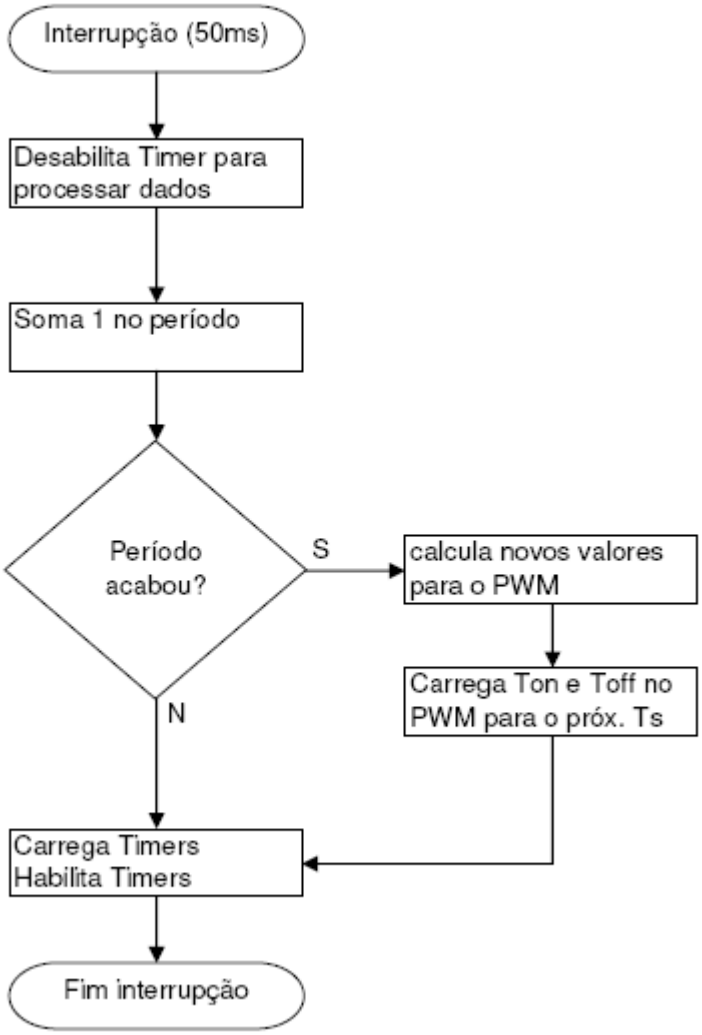
Tela menu de seleção



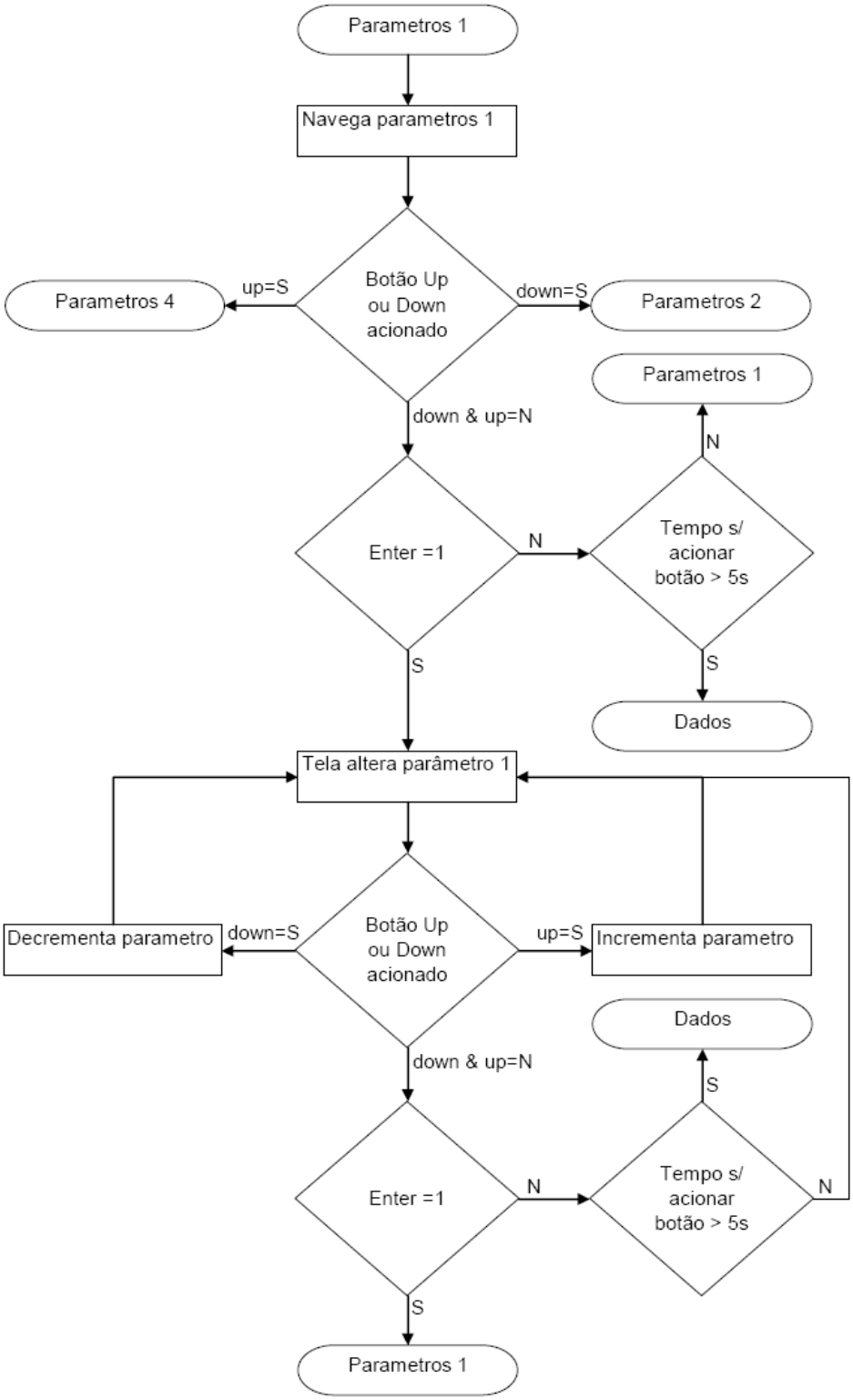
Tela de visualização dos alarmes



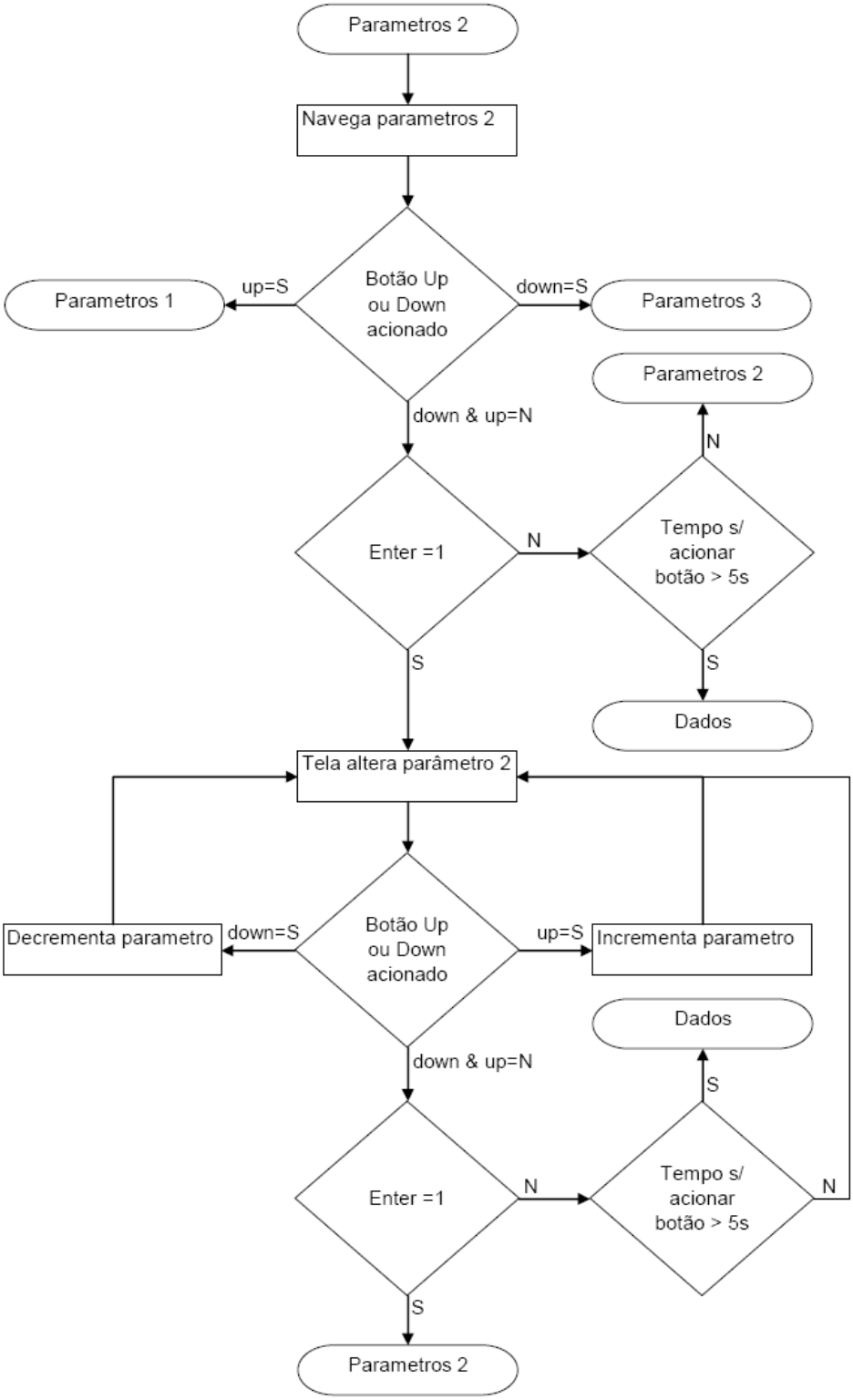
Atendimento a interrupção.



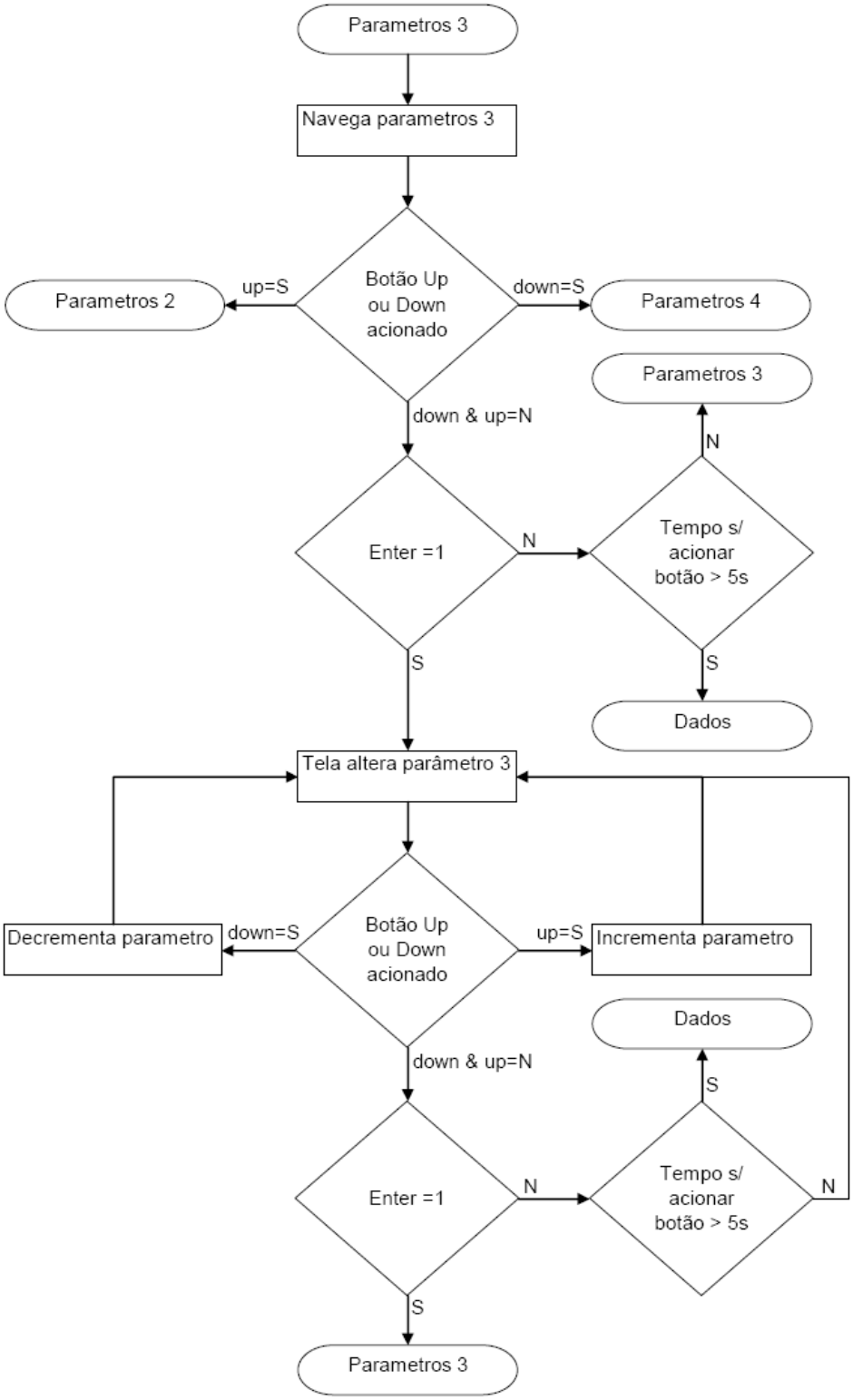
Navegação telas de alteração dos parâmetros 1/4.



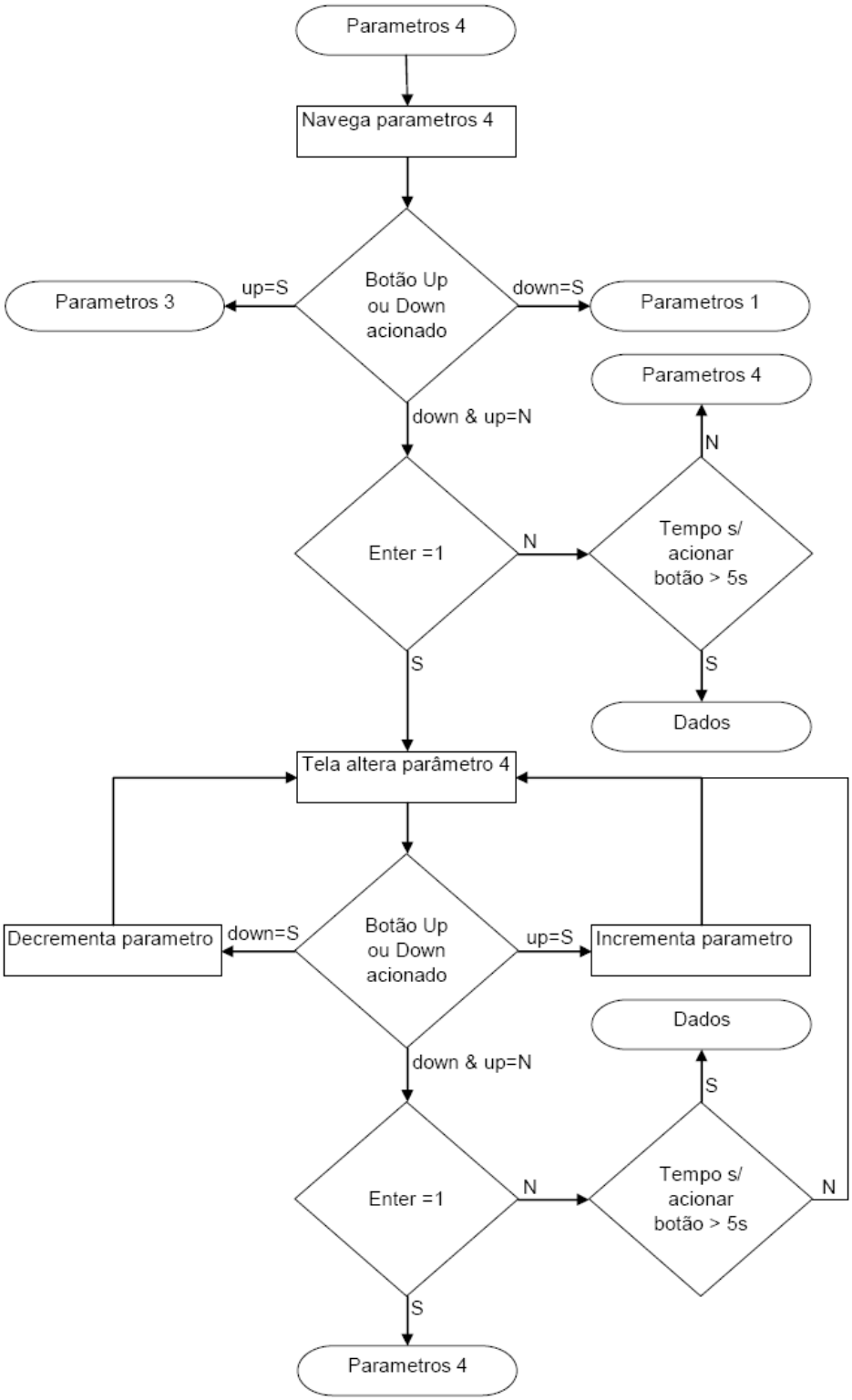
Navegação telas de alteração dos parâmetros 2/4.



Navegação telas de alteração dos parâmetros 3/4.

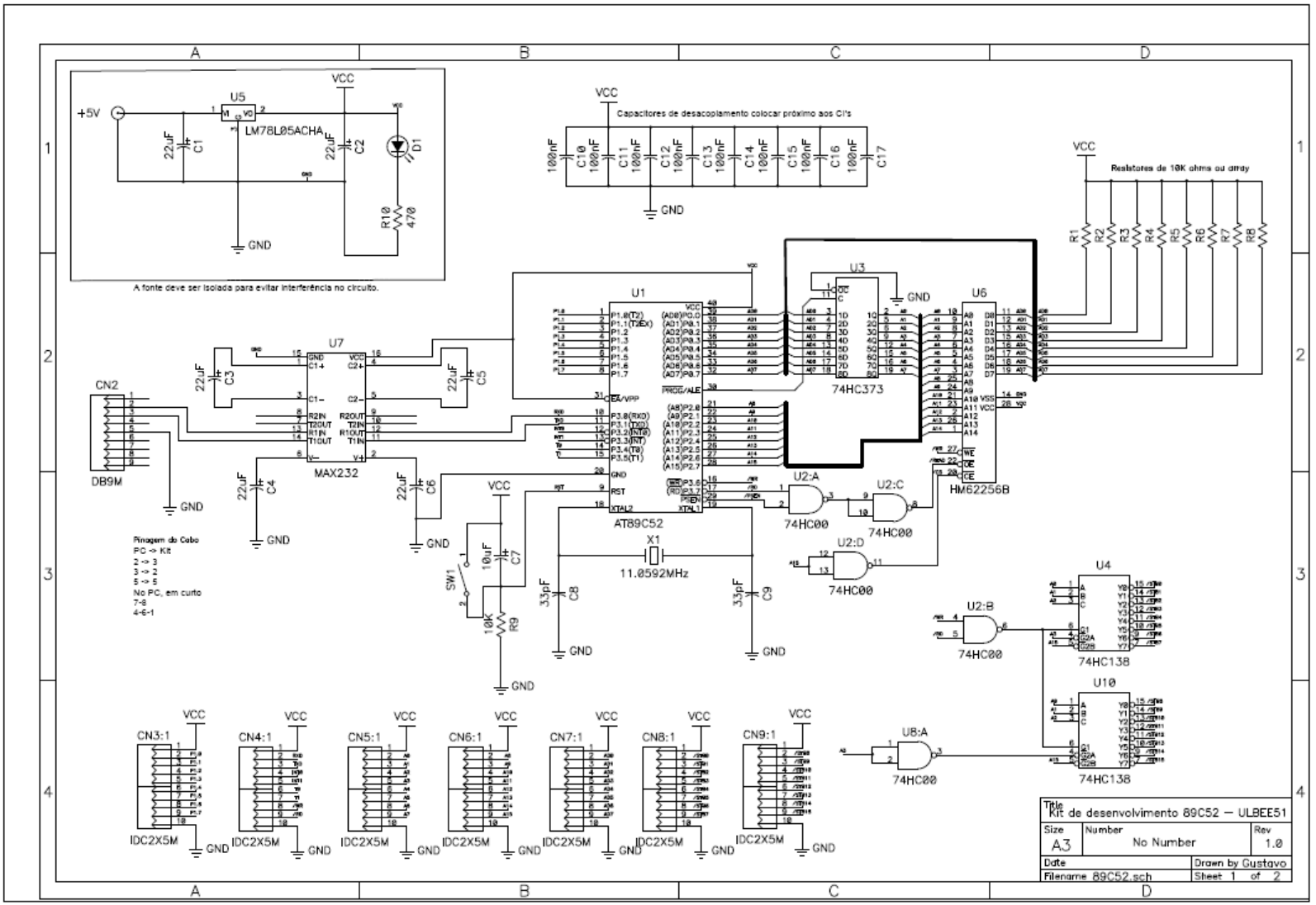


Navegação telas de alteração dos parâmetros 4/4.



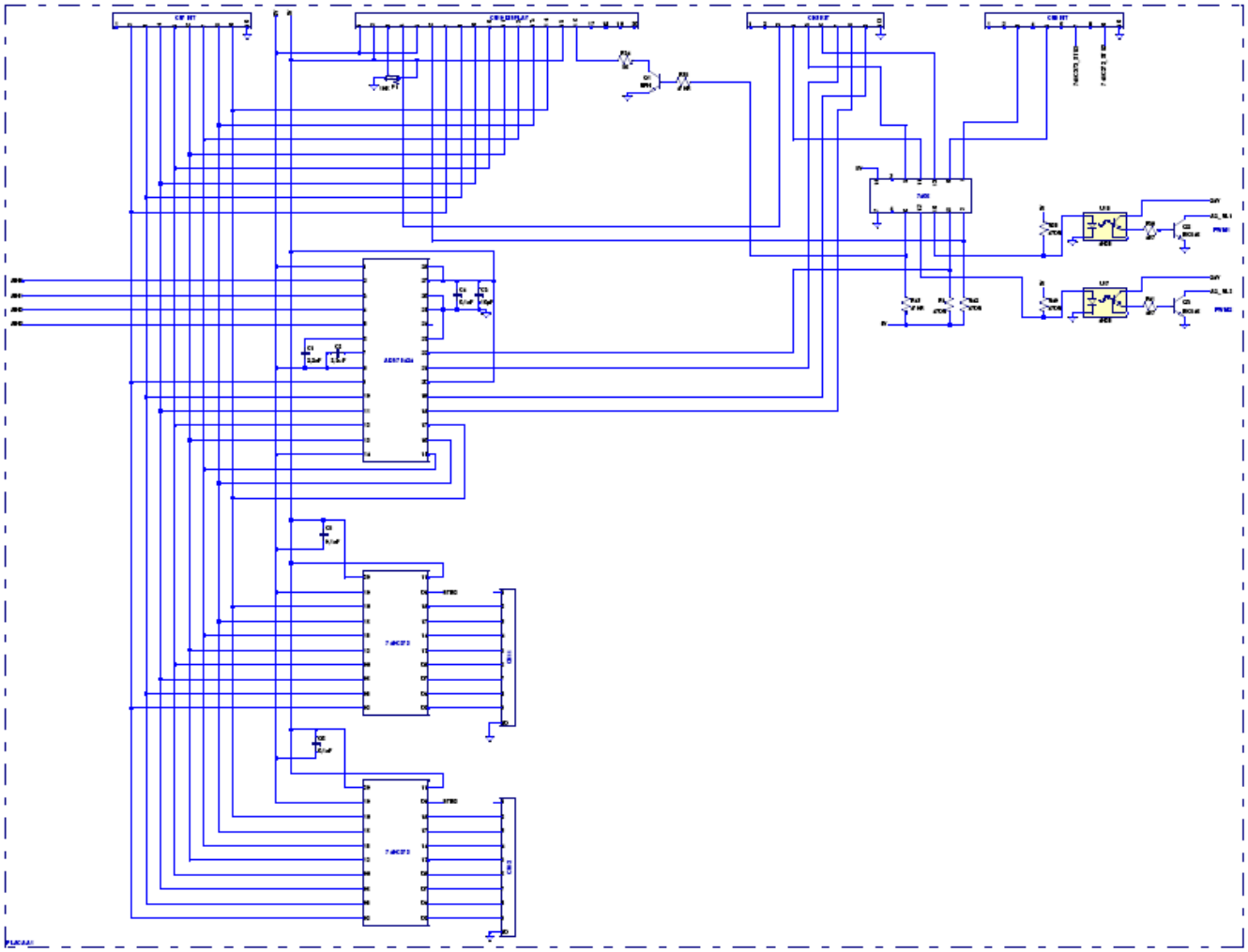
APÊNDICE C – DIAGRAMAS ELETRÔNICOS

Kit de desenvolvimento 89C52 ULBEE51.

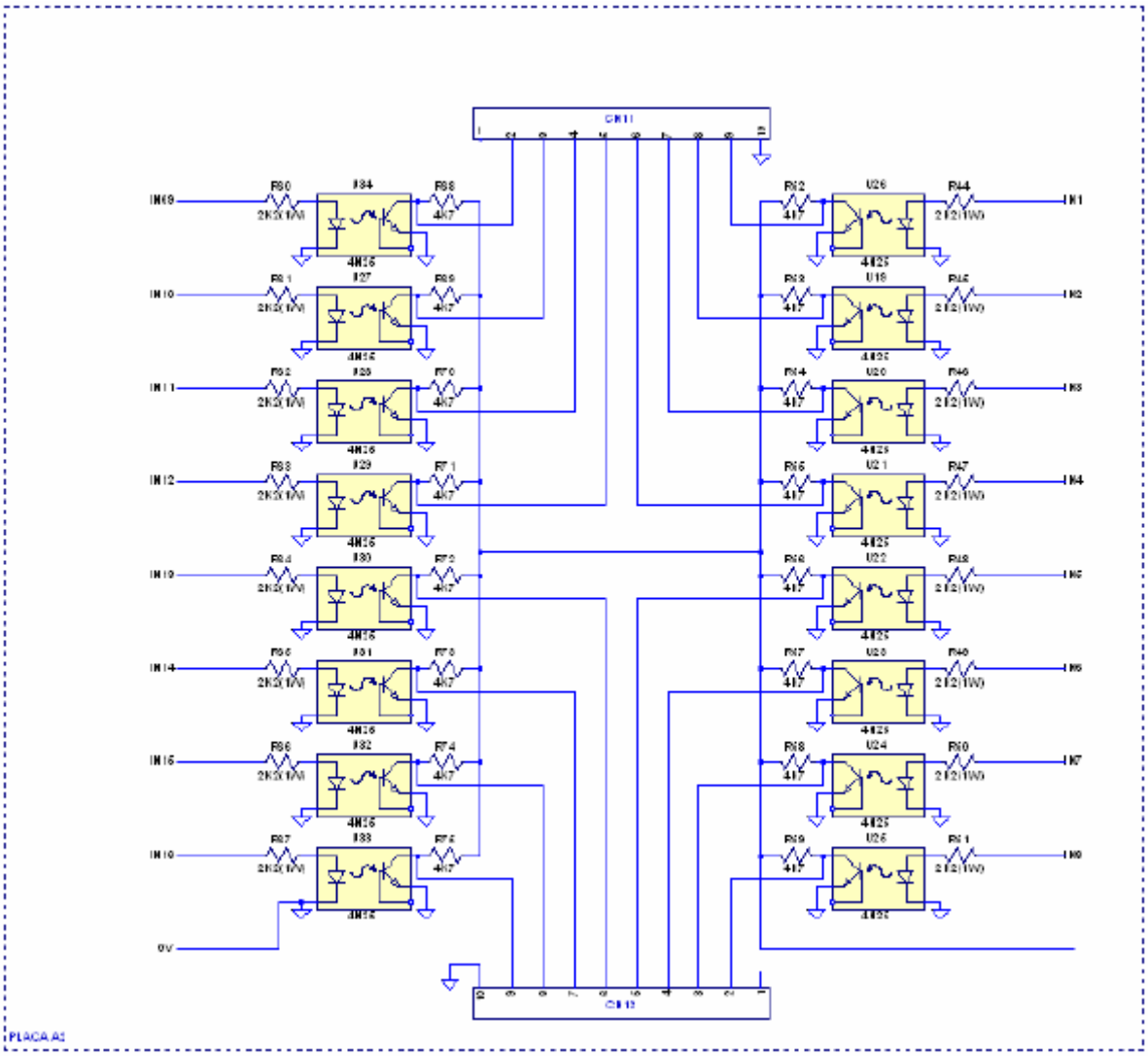


MARCOS HENRIQUE RIBACKI
Universidade Luterana do Brasil

Placa de entradas analógicas A1.



Placa de entradas digitais (A2).



APÊNDICE C – ROTINA DO MICROCONTROLADOR

```
/*
Universidade Luterana do Brasil
Engenharia Elétrica
TCC
Projeto: Controle microprocessado de temperatura para matriz de forjamento a quente.
Aluno: Marcos Ribacki
Semestre: 2008/2    */
#include <at89x52.h>
#include <paulmon2.h>
// Conexão dos pinos do microcontrolador Atmel AT89C52, porta P1
#define DISPLAY_RS              P1_1        //Seleciona dado ou istrução no display, 0=instrução 1=dado
#define PWM_MAT_SUP             P1_2        //Aciona contator matriz superior
#define DISPLAY_BACKLIGHT       P1_3        //habilita Backlight do dispaly, habilita em 0
#define PWM_MAT_INF             P1_4        //Aciona contator matriz inferior
#define AD_B_select             P1_5        //Seleção do byte 1=D0-D3 or 0=D4-D11
#define AD_A0                   P1_6        //Seleção da entrada analógica converssor AD
#define AD_A1                   P1_7        //Seleção da entrada analógica converssor AD
/*Tabela de seleção das entradas analógicas
A0      A1      Channel
0       0       AIN0
0       1       AIN1
1       0       AIN2
1       1       AIN3        */
// Conexão dos pinos do microcontrolador Atmel AT89C52, strobe.
#define DISPLAY_E               0x0001      //Ativa LCD, ativa em 0
#define AD_RC                   0x0003      //Abilita leitura de dados do AD
#define LACH_1                  0x0005      //Abilita leitura de dados do LACH_1 byte 0
#define LACH_2                  0x0007      //Abilita leitura de dados do LACH_2 byte 1
volatile xdata at DISPLAY_E unsigned char lcd;
volatile xdata at AD_RC unsigned char AD_READ;
volatile xdata at LACH_1 unsigned char BYTE0;
volatile xdata at LACH_2 unsigned char BYTE1;
void inicia_lcd (void);
void tela_intro(void);
void tela_01(void);
void tela_02(void);
void tela_exe_textos(void);
void tela_exe_dados(void);
void tela_menu (void);
void tela_alarmes (void);
void tela_parametros (void);
void rotina_compara_dados (void);
void rotina_define_action1(void);
void tela_imprime_byte(void);
void atualiza_vetor_alarmes (void);
void atualiza_vetor_comandos (void);
void ler_ad_an0_only (void);
void ler_ad_an2_only (void);
void ler_byte_0 (void);
void ler_byte_1 (void);
void escreve_lcd (char *texto);
void tempo(float);
void delayms(char c);
void segunda_linha (void);
void limpa_lcd (void);
void terceira_linha (void);
void quarta_linha (void);
void espaco (int quantidade);
void ler_ad_an0 (void);
void ler_ad_an2 (void);
void tela_parametros_1 (void);
void tela_parametros_2 (void);
void tela_parametros_3 (void);
void tela_parametros_4 (void);
void tela_ler_funcao_parametros (void);
void tela_exe_funcao_parametros (void);
void tela_navega_parametros (void);
void controle_tempo_pwm (void)interrupt 1;
void inicia_pwm (void);
void controle_PID (void);
int valor_byte_0 = 0;                       //memória valor entrada alarmes
```

```
int valor_byte_1 = 0;                          //memória valor entrada comandos
int valor_AN0 = 0;                             //memória valor temperatura matriz inferior
int valor_AN1 = 0;                             //memória valor temperatura caixa de ligação mesa
inferior
int valor_AN2 = 0;                             //memória valor temperatura matriz superior
int valor_AN3 = 0;                             //memória valor temperatura cx de ligação mesa superior
int contador_2 = 0;                            //tempo tela menu
int contador_3 = 0;                            //tempo tela parametros
int contador_4 = 0;                            //tempo coleta de dados hiper terminal

int T_inf = 0;                                 //tempo período PWM matriz inferior
int Tempo_avalia_PWM = 0;
int Ton = 0;
int inf_Ton = 0;

//<<variáveis PWM matriz inferior>>
int inf_Vc = 0;                                //Sinal de controle PWM
int inf_Vc_ant = 0;                            //Sinal de controle PWM anterior
int inf_erro = 0;                              //Erro atual
int inf_erro_ant_1 = 0;                        //Erro anterior
int inf_erro_ant_2 = 0;                        //Erro anterior anterior
//parametros PID
int inf_Kp = 2.99;                             //constante proporcional
int inf_Kd = 30;                               //Constante derivativa
int inf_Ki = 0.1;                              //Constante integral
int inf_Ts = 2;                                //Tempo período de amostragem (s)

char menu = 1;                                 //case seleção fluxo do programa principal
char tela = 1;                                 //case seleção das telas de programação de parametros
char down_up = 0;                              //contador controle botton UP or DOWN acionado
navegação rotina parametros
char enter = 1;                                //contador controle botton enter acionado rotina
parametros
char bite_0 [8];                               //vetor bite alarmes
char bite_1 [8];                               //vetor bite comandos
int temperatura [5];                           //vetor set point de temperatura

void main(void)                                //rotina principal
{       switch(menu)                           //controle fluxo programa principal
        {
        case 1: inicia_lcd();                  //chama rotina de inicialização do display
                tela_intro();
                tempo(1);
                limpa_lcd();
                tela_01();                     //chama rotina da primeira tela de apresentação
                tempo(1);                      //tempo para mostrar a primeira tela de apresentação
                limpa_lcd();                   //chama rotina para limpar LCD
                tela_02();                     //chama rotina da segunda tela de apresentação
                tempo(1);                      //tempo para mostrar a segunda tela de apresentação
                rotina_le_dados();
                inicia_o_pwm();
                temperatura [1]=200;
                menu = 3;                      //define main ir para rotina ler dados
        break;
        case 2: rotina_le_dados();
                if(valor_byte_0 == 255 && valor_byte_1 == 255)
                {
                        tela_exe_textos();
                        do
                        {
                                tela_exe_dados();
                                contador_4++;
                                if(contador_4 > 43)
                                {
                                        pm2_pint16u(valor_AN0);
                                        tempo(1);
                                        pm2_newline();
                                        contador_4 = 0;
                                }
                        }while(valor_byte_0 == 255 && valor_byte_1 == 255);
                }
                menu = 3;                      //define main ir comparar dados
        break;
        case 3: if(valor_byte_0 != 255)
                {
```

```
                atualiza_vetor_alarmes ();
                menu = 8;                       //define main ir para tela menu
        }
        else if(valor_byte_1 != 255)
        {
                        atualiza_vetor_comandos ();
                        if(bite_1[2] == 0)
                        {
                                menu = 5;       //define main ir para tela menu
                                tempo(2);
                        }
                }
                else if(menu == 3)
                {
                        menu = 2;               //define main ir para ler dados
                }
        break;
        case 4: tela_alarmes ();
                menu = 2;                       //define main ir para ler dados
        break;
        case 5: tela_menu();
                valor_byte_1 = 255;
                contador_2 = 0;                 //resset contador_2 tempo tela menu
                while(valor_byte_1 == 255 && contador_2 < 75)
                {
                ler_byte_1();
                atualiza_vetor_comandos ();
                delayms(3);                     //tempo para sair da tela menu, aprox. 5s.
                contador_2++;
                }
                if(bite_1[0] == 0)
                {
                        menu = 4;               //define main ir para tela alarmes
                }
                else if(bite_1[1] == 0)
                {
                        tela = 1;
                        menu = 6;               //define main ir para navega_parametros
                }
                else if(menu == 5)
                {
                        menu = 2;               //define main ir para ler dados
                }
        break;
        case 6: tela_navega_parametros();
                menu = 2;                       //define main ir para ler dados
        break;
        }
        main();                                 //chama o programa main novamente
}

void controle_tempo_pwm (void) interrupt 1
{
        TR0 = 0;                                //desabilita contagem o timer 0
        Tempo_avalia_PWM++;
        if(Tempo_avalia_PWM >= 2)
        {
                T_inf++;
                if(T_inf >= 20)
                {
                        controle_PID ();
                        inf_Ton = inf_Vc;
                        T_inf = 0;
                }
                if(inf_Ton >= 0)
                {
                        PWM_MAT_INF = 0;
                        inf_Ton--;
                }
                if(inf_Ton < 0)
                {
                        PWM_MAT_INF = 1;
                }
                Tempo_avalia_PWM = 0;
        }
```

```
        TL0 = 0x0A;
        TH0 = 0x3C;
        TR0 = 1;                                        //Habilita inicio do timer
}

void controle_PID (void)
{
        int q0,q1,q2;
        inf_erro = temperatura[1] - valor_AN0;
        q0 = inf_Kp + (inf_Kd/inf_Ts) + inf_Ki*inf_Ts;
        q1 = -inf_Kp - 2*(inf_Kd/inf_Ts);
        q2 = inf_Kd/inf_Ts;
        inf_Vc = inf_Vc_ant + q0*inf_erro + q1*inf_erro_ant_1 + q2*inf_erro_ant_2;
        inf_erro_ant_2 = inf_erro_ant_1;
        inf_erro_ant_1 = inf_erro;
        inf_Vc_ant = inf_Vc;
}
void tela_01(void)                                      //sub rotina da tela 1 de apresentação
{
        escreve_lcd("*******PRR-03******");             //escreve string no LCD primeira linha
        escreve_lcd("do ferramental      ");             //escreve string no LCD terceira linha
        segunda_linha();                                //posiciona cursor na segunda linha
        escreve_lcd("manutencao termica ");             //escreve string no LCD segunda linha
}

void tela_intro(void)                                   //sub rotina da tela 1 de apresentação
{
        escreve_lcd("TCC ENG ELETRICA   ");             //escreve string no LCD primeira linha
        escreve_lcd("2008_2 ULBRA       ");             //escreve string no LCD terceira linha
        segunda_linha();                                //posiciona cursor na segunda linha
        escreve_lcd("MARCOS RIBACKI     ");             //escreve string no LCD segunda linha
        quarta_linha();                                 //posiciona cursor na segunda linha
        escreve_lcd("ORIENTADOR: AUGUSTO");
}

void ler_ad_an0 (void)
{
        char nibble_alto = 0;
        char nibble_baixo = 0;
        int conversao = 0;
        int  vala, valb, valc;
        AD_A1=0;
        AD_A0=0;
        AD_B_select=0;
        delayms(2);
        nibble_alto=AD_READ;                            // Lê a conversão do AD    e coloca na variável
        AD_B_select=1;
        delayms(2);
        nibble_baixo=AD_READ;
        delayms(2);
        AD_B_select=0;
        delayms(2);
        nibble_alto=AD_READ;
        AD_B_select=1;
        delayms(2);
        nibble_baixo=AD_READ;
        delayms(2);
        nibble_baixo >>= 4;
        nibble_baixo &= 0x0f;
        conversao = nibble_alto;                        //salva byte alto
        conversao <<= 4;                                //desloca 4 bits pra poder salvar o nibble baixo depois
        conversao &= 0x0ff0;                            //mascara pra garantir
        conversao |= nibble_baixo;                      //soma nibble baixo
        conversao = (conversao/5.12);
        valor_AN0 = conversao;
        vala=conversao/100;
        valb=(conversao%100)/10;
        valc=((conversao%100)%10)/1;
        lcd=vala + 0x30;
        delayms(2);
        lcd=valb + 0x30;
        delayms(2);
        lcd=valc + 0x30;
        delayms(2);
        lcd=0xDF;
```

```
	delayms(2);
}

void ler_ad_an2 (void)
{
	char nibble_alto = 0;
	char nibble_baixo = 0;
	int conversao = 0;
	int  vala, valb, valc;
	AD_A1=0;
	AD_A0=1;
	AD_B_select=0;
	delayms(2);
	nibble_alto=AD_READ;			// Lê a conversão do AD    e coloca na variável
	AD_B_select=1;
	delayms(2);
	nibble_baixo=AD_READ;
	delayms(2);
	AD_B_select=0;
	delayms(2);
	nibble_alto=AD_READ;			// Lê a conversão do AD    e coloca na variável
	AD_B_select=1;
	delayms(2);
	nibble_baixo=AD_READ;
	delayms(2);
	nibble_baixo >>= 4;
	nibble_baixo &= 0x0f;
	conversao = nibble_alto;		//salva byte alto
	conversao <<= 4;			//desloca 4 bits pra poder salvar o nibble baixo depois
	conversao &= 0x0ff0;			//mascara pra garantir
	conversao |= nibble_baixo;		//soma nibble baixo
	conversao = (conversao/5.12);
	valor_AN2 = conversao;
	vala=conversao/100;
	valb=(conversao%100)/10;
	valc=((conversao%100)%10)/1;
	lcd=vala + 0x30;
	delayms(2);
	lcd=valb + 0x30;
	delayms(2);
	lcd=valc + 0x30;
	delayms(2);
	lcd=0xDF;
	delayms(2);
}

void tela_navega_parametros (void)
{
	switch(tela)
	{
		case 1: tela_parametros_1();
			tela_ler_funcao_parametros ();
			tela_exe_funcao_parametros ();
			break;
		case 2: tela_parametros_2();
			tela_ler_funcao_parametros ();
			tela_exe_funcao_parametros ();
		break;
		case 3: tela_parametros_3();
			tela_ler_funcao_parametros ();
			tela_exe_funcao_parametros ();
		break;
		case 4: tela_parametros_4();
			tela_ler_funcao_parametros ();
			tela_exe_funcao_parametros ();
		break;
	}
	tela_navega_parametros();
}

void tela_exe_funcao_parametros (void)
{
	int  valr, vala, valb, valc;
	{
		tela = tela + down_up;
```

```
                if(tela > 4)
                {
                        tela = 1;
                }
                else if(tela < 1)
                {
                        tela = 4;
                }
        }
        if(down_up != 0)
        {
                tela_navega_parametros ();
        }
        else if(enter == 0)
        {
                enter = 1;
                tempo(2);
                while(enter ==1)
                {
                        tela_ler_funcao_parametros ();
                        temperatura[tela]=temperatura[tela]+down_up;
                        if(temperatura[tela] > 300)
                        {
                                temperatura[tela] = 300;
                        }
                        else if(temperatura[tela] < 0)
                        {
                                temperatura[tela] = 0;
                        }
                        quarta_linha();
                        valr=temperatura[tela];
                        delayms(2);
                        vala=valr/100;
                        valb=(valr%100)/10;
                        valc=((valr%100)%10)/1;
                        delayms(2);
                        lcd=vala + 0x30;
                        delayms(2);
                        lcd=valb + 0x30;
                        delayms(2);
                        lcd=valc + 0x30;
                        delayms(2);
                        lcd=0xDF;
                        delayms(2);
                }
                tela_navega_parametros();
        }
}

void tela_ler_funcao_parametros (void)
{
        contador_3=0;
        ler_byte_1();
        while(valor_byte_1 == 255 && contador_3 < 200)
        {
                ler_byte_1();
                delayms(21);                    //tempo para sair da tela de parametros, aprox 5s
                contador_3++;
        }
    if(valor_byte_1 != 255)
    {
        atualiza_vetor_comandos ();
        down_up=0;
        delayms(100);                           //tempo incrementar/decrementar parametro
        if(bite_1[1] == 0 )
        {
                down_up = -1;
        }
        else if(bite_1[0] == 0)
        {
                down_up = 1;
        }
                else if(bite_1[2] == 0)
                {
                        enter = 0;
```

```
                        }
        }
        else if(contador_3 > 199)
        {
            menu = 2;
            main();
        }
}

void tela_parametros_1 (void)
{
            int  valr, vala, valb, valc;
            limpa_lcd();
            escreve_lcd("Parametro:");
            segunda_linha();
            escreve_lcd("Temperatura mesa");
            terceira_linha();
            escreve_lcd("inferior:");
            quarta_linha();
            valr=temperatura[tela];
            delayms(2);
            vala=valr/100;
            valb=(valr%100)/10;
            valc=((valr%100)%10)/1;
            delayms(2);
            lcd=vala + 0x30;
            delayms(2);
            lcd=valb + 0x30;
            delayms(2);
            lcd=valc + 0x30;
            delayms(2);
            lcd=0xDF;
            delayms(2000);
}

void tela_parametros_2 (void)
{
            int  valr, vala, valb, valc;
            limpa_lcd();
            escreve_lcd("Parametro:");
            segunda_linha();
            escreve_lcd("Temperatura mesa");
            terceira_linha();
            escreve_lcd("superior:");
            quarta_linha();
            valr=temperatura[tela];
            delayms(2);
            vala=valr/100;
            valb=(valr%100)/10;
            valc=((valr%100)%10)/1;
            delayms(2);
            lcd=vala + 0x30;
            delayms(2);
            lcd=valb + 0x30;
            delayms(2);
            lcd=valc + 0x30;
            delayms(2);
            lcd=0xDF;
            delayms(2000);
}

void tela_parametros_3 (void)
{
            int  valr, vala, valb, valc;
            limpa_lcd();
            escreve_lcd("Parametro:");
            segunda_linha();
            escreve_lcd("Temperatura caixa de");
            terceira_linha();
            escreve_lcd("ligacao inferior:");
            quarta_linha();
            valr=temperatura[tela];
            delayms(2);
            vala=valr/100;
            valb=(valr%100)/10;
```

```
		valc=((valr%100)%10)/1;
		delayms(2);
		lcd=vala + 0x30;
		delayms(2);
		lcd=valb + 0x30;
		delayms(2);
		lcd=valc + 0x30;
		delayms(2);
		lcd=0xDF;
		delayms(2000);
}

void tela_parametros_4 (void)
{
		int  valr, vala, valb, valc;
		limpa_lcd();
		escreve_lcd("Parametro:");
		segunda_linha();
		escreve_lcd("Temperatura caixa de");
		terceira_linha();
		escreve_lcd("ligacao superior:");
		quarta_linha();
		valr=temperatura[tela];
		delayms(2);
		vala=valr/100;
		valb=(valr%100)/10;
		valc=((valr%100)%10)/1;
		delayms(2);
		lcd=vala + 0x30;
		delayms(2);
		lcd=valb + 0x30;
		delayms(2);
		lcd=valc + 0x30;
		delayms(2);
		lcd=0xDF;
		delayms(2000);
}

void tela_menu (void)
{
		limpa_lcd();
		escreve_lcd("Precione UP ou DOWN");
		segunda_linha();
		escreve_lcd("UP = Ver alarmes");
		terceira_linha();
		escreve_lcd("DOWN = Parametros");
}

void tela_exe_textos(void)
{
		limpa_lcd();
		escreve_lcd("Temperatura Matriz:");
		escreve_lcd("Inferior =");
		terceira_linha();
		espaco(15);
		escreve_lcd("C");
		segunda_linha();
		escreve_lcd("Superior =");
		segunda_linha();
		espaco(15);
		escreve_lcd("C");
		espaco(3);
		quarta_linha();
		escreve_lcd("Pi=");
		espaco(1);
		escreve_lcd("Kw");
		espaco(3);
		escreve_lcd(" Ps=");
		espaco(1);
		escreve_lcd("Kw");
}

void inicia_pwm (void)                          //rotina para inicalização do timer 0, interrupções
{
```

```
	EA = 1;                                  //Permite que cada interrupção seja habilitada
individualmente
	ET0 = 1;                                 //Habilita a interrupção pedida pelo timer 0
	TMOD = 0x21;                             //Habilita a timer 0, modo 1 (16bits) cotrole por software
	TL0 = 0x0A;
	TH0 = 0x3C;
	TR0 = 1;                                 //Habilita inicio do timer
}



void tela_02(void)
{
	escreve_lcd("Engenharia de P&D  ");
	escreve_lcd("Projeto AEMF-GF35   ");
	segunda_linha();
	escreve_lcd("Maxiforja 08/2008   ");
	escreve_lcd("            V1.1");
}

void tela_alarmes (void)
{
	limpa_lcd();
	escreve_lcd("Alarmes do sistema: ");
	if(bite_0[0] == 0 )
	{
		segunda_linha();
		escreve_lcd("Resistencia 1 fault ");
	}
	tempo(25);                               //tempo para sair da tela alarmes, aprox 5s
}

void atualiza_vetor_alarmes (void)
{
	int val_enter =0 , contador = 0;
	delayms(5);
	val_enter=valor_byte_0;
	delayms(5);
	while (contador < 8)
	{
		bite_0[contador] = val_enter%2;
		val_enter = val_enter/2;
		contador++;
	}
}

void atualiza_vetor_comandos (void)
{
	int val_enter =0 , contador = 0;
	delayms(5);
	val_enter=valor_byte_1;
	delayms(5);
	while (contador < 8)
	{
		atualiza_vetor_alarmes();
		bite_1[contador] = val_enter%2;
		val_enter = val_enter/2;
		contador++;
	}
}

void tela_imprime_byte(void)
{
	int contador = 0;
	limpa_lcd();
	delayms(10);
	while(contador < 8)
	{
		lcd=bite_0[contador] + 0x30;
		delayms(10);
		contador++;
	}
	delayms(10);
	segunda_linha();
	contador =0;
```

```
		while(contador < 8)
		{
			atualiza_vetor_comandos();
			lcd=bite_1[contador] + 0x30;
			delayms(10);
			contador++;
		}
}

void rotina_le_dados(void)
{
		ler_ad_an0_only();
		ler_ad_an2_only();
		ler_byte_0();
		ler_byte_1();
}

void ler_byte_0 (void)
{
		int val_enter;
		val_enter =0;
		val_enter=BYTE0;
		delayms(2);
		valor_byte_0 = val_enter;
}

void ler_byte_1 (void)
{
		int val_enter;
		val_enter =0;
		val_enter=BYTE1;
		delayms(2);
		valor_byte_1 = val_enter;
}

void tela_exe_dados(void)
{
		 int val_enter =0;
		segunda_linha();
		DISPLAY_RS=0;
		delayms(2);
		lcd=0xCB;                                   //Desloca cursor para coluna 12
		delayms(2);
		DISPLAY_RS=1;
		delayms(2);
		ler_ad_an2();
		terceira_linha();
		espaco(11);
		delayms(2);
		ler_ad_an0();
		delayms(5);
		val_enter =0;
		val_enter=BYTE0;                            // Lê lê byte 0      e coloca na variável
		delayms(2);
		valor_byte_0 = val_enter;
		val_enter =0;
		val_enter=BYTE1;
		delayms(2);
		valor_byte_1 = val_enter;
}

void ler_ad_an0_only (void)
{
		char nibble_alto = 0;
		char nibble_baixo = 0;
		int conversao = 0;
		AD_A1=0;
		AD_A0=0;
		AD_B_select=0;
		delayms(2);
		nibble_alto=AD_READ;                        // Lê a conversão do AD     e coloca na variável
		AD_B_select=1;
		delayms(2);
		nibble_baixo=AD_READ;
		delayms(2);
```

```
        AD_B_select=0;
        delayms(2);
        nibble_alto=AD_READ;
        AD_B_select=1;
        delayms(2);
        nibble_baixo=AD_READ;
        delayms(2);
        nibble_baixo >>= 4;
        nibble_baixo &= 0x0f;
        conversao = nibble_alto;                //salva byte alto
        conversao <<= 4;                        //desloca 4 bits pra poder salvar o nibble baixo depois
        conversao &= 0x0ff0;                    //mascara pra garantir
        conversao |= nibble_baixo;              //soma nibble baixo
        conversao = (conversao/5.12);
        valor_AN0 = conversao;
}

void ler_ad_an2_only (void)
{
        char nibble_alto = 0;
        char nibble_baixo = 0;
        int conversao = 0;
        AD_A1=0;
        AD_A0=1;
        AD_B_select=0;
        delayms(2);
        nibble_alto=AD_READ;
        AD_B_select=1;
        delayms(2);
        nibble_baixo=AD_READ;
        delayms(2);
        AD_B_select=0;
        delayms(2);
        nibble_alto=AD_READ;
        AD_B_select=1;
        delayms(2);
        nibble_baixo=AD_READ;
        delayms(2);
        nibble_baixo >>= 4;
        nibble_baixo &= 0x0f;
        conversao = nibble_alto;                //salva byte alto
        conversao <<= 4;                        //desloca 4 bits pra poder salvar o nibble baixo depois
        conversao &= 0x0ff0;                    //mascara pra garantir
        conversao |= nibble_baixo;              //soma nibble baixo
        conversao = (conversao/5.12);
        valor_AN2 = conversao;
}

void segunda_linha (void)                       //sub rotina posiciona cursor na segunda linha
{
        int aux = 0;                            //resset valor da variavel int
        DISPLAY_RS=0;                           //habilita display para receber instrução
        delayms(2);
        lcd=0xC0;                               //posiciona cursor na segunda linha
        delayms(2);
        DISPLAY_RS=1;                           //habilita display pra receber dados
        delayms(2);
}

void terceira_linha (void)                      //sub rotina posiciona cursor na terceira linha
{
        DISPLAY_RS=0;                           //habilita display para receber instrução
        delayms(2);
        lcd=0x80;                               //posiciona cursor na primeira linha
        delayms(2);
        espaco(20);                             //desloca cursor 21 posicições a direita = terceira linha
}

void quarta_linha (void)                        //sub rotina posiciona cursor na quarta linha
{
        int aux = 0;                            //inicia variável aux com valor "0"
        DISPLAY_RS=0;                           //habilita display para receber instrução
        delayms(2);
        lcd=0xC0;                               //posiciona cursor na segunda linha
        delayms(2);
```

```
        espaco(20);                         //desloca cursor 20 posicões a direita = quarta linha
}

void espaco(int quantidade)                 //sub rotina posiciona cursor quantidade de espaço para
direita
{
        int aux =0 ;                        //inicia variável aux com valor "0"
        DISPLAY_RS=0;                       //habilita display para receber instrução
        delayms(2);
        while(aux < quantidade)             //condição
        {
                aux++;                      //soma 1 na variável aux
                lcd=0x14;                   //desloca cursor 1 posição para direita
                delayms(1);
        }
        DISPLAY_RS=1;                       //habilita display para receber instrução
        delayms(2);
}

void limpa_lcd (void)                       //sub rotina limpa LCD
{
        DISPLAY_RS=0;                       /habilita display para receber instrução
        delayms(2);
        lcd=0x01;                           //limpa display com home cursor
        delayms(2);
        DISPLAY_RS=1;                       //habilita display pra receber dados
        delayms(2);
}

void escreve_lcd (char *texto)              //sub rotina escreve string no LCD
{
        char dado;
        dado=*texto;
        while (dado!=0)
                {
                        dado=*texto;
                        lcd=dado;
                        delayms(1);
                        texto ++;
                }
        DISPLAY_RS=0;
        delayms(2);
        lcd=0x10;
        delayms(2);
        DISPLAY_RS=1;
        delayms(2);
        lcd=0x20;
        delayms(2);
}

void inicia_lcd (void)                      //Liga o LCD_clean com cursor em home
{
        DISPLAY_BACKLIGHT=0;                //Liga backlight do LCD
        delayms(10);
        DISPLAY_RS=0;                       //habilita display para receber instrução
        delayms(10);
        lcd=0x3C;                           //parametriza display 4 linhas matriz 5 x 10 8bits
        delayms(10);
        lcd=0x0C;                           //Liga display sem cursor
        delayms(10);
        lcd=0x01;                           //Limpa display com home cursor
        delayms(10);
        DISPLAY_RS=1;                       //habilita display para receber dados
        delayms(10);
}

void tempo(float temp)
{
        float i=0;
        int j;
        float a;
        a = temp*250;
        for(i=0;i<a;i++)
        {
                j++;
```

```
                j--;
        }
}

void delayms(char c)                    //rotina de tempo para processamento de dados
{
        c;
        _asm
                mov r0,dpl
        0100$:
                mov R2,#8
        0200$:
                mov R1,#55
        0300$:
                djnz R1,0300$
                djnz R2,0200$
                djnz R0,0100$
                ret
        _endasm;
}
```

ANEXO A – MANUAL TXRAIL, NOVUS PRODUTOS ELETRONICOS

INSTALAÇÃO MECÂNICA

O transmissor tem gabinete próprio para ser instalado em trilho de 35 mm.

Dimensões:

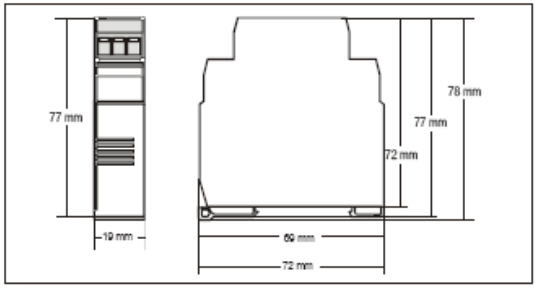


Figura 04 – Dimensões do transmissor

INSTALAÇÃO ELÉTRICA

Invólucro dos Terminais em poliamida.

Secção do fio utilizado: 0,14 a 1,5 mm².

Torque recomendado: 0,8 Nm.

Recomendações para a Instalação

- Condutores de sinais de entrada devem percorrer a planta do sistema separados dos condutores de saída e de alimentação, se possível em eletrodutos aterrados.
- A alimentação dos instrumentos deve vir de uma rede própria para instrumentação.
- Em aplicações de controle e monitoração é essencial considerar o que pode acontecer quando qualquer parte do sistema falhar.
- É recomendável o uso de FILTROS RC (47 Ω e 100 nF, série) em bobinas de contactoras, solenóides, etc.

Conexões Elétricas

A figura abaixo mostra as conexões elétricas necessárias. Os terminais 1, 2 e 3 são dedicados à conexão de entrada (sensores, sinais, etc). Quando Pt100 2 fios os terminais 2 e 3 devem ser interligados.

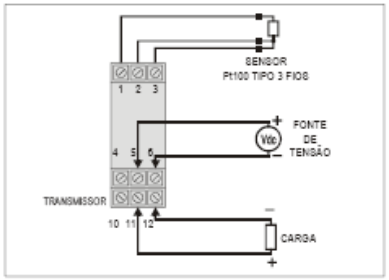


Figura 05 – Conexões elétricas do transmissor – Pt100

MARCOS HENRIQUE RIBACKI
Universidade Luterana do Brasil

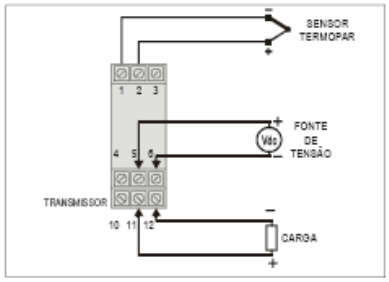


Figura 06 – Conexões elétricas do transmissor – Termopar

OPERAÇÃO

O transmissor vem de fábrica perfeitamente calibrado com sensores padronizados, não necessitando nenhum ajuste por parte do usuário.

Quando necessário, pequenas correções no sinal de saída podem ser feitas diretamente no transmissor pelo usuário. Para isso basta pressionar as teclas identificadas como ZERO+ e ZERO- no frontal do transmissor. Estas duas teclas estão localizadas logo abaixo da etiqueta frontal para evitar ajustes acidentais. Com uma pequena ferramenta de 2mm de diâmetro é possível acessar as teclas. Após o tempo de dois segundos da tecla pressionada, a tensão de saída começa a ser alterada para mais (ajuste ZERO+) ou para menos (ajuste ZERO-). O usuário deve monitorar a tensão de saída e, ao alcançar o valor desejado, soltar a tecla.

Estas pequenas correções podem ser feitas também através do software TxConfig (agora em unidades de temperatura). Ver Figura 03 e campo **Correção de Zero** na tela principal do software TxConfig. A interface TxConfig pode ser conectada ao transmissor mesmo com este ligado ao processo e operando.

O usuário deve escolher sensor e faixa mais adequados ao seu processo. A faixa escolhida não deve ultrapassar a faixa máxima de medição definida para o sensor e não deve ser menor que a faixa mínima para este mesmo sensor.

É importante observar que a precisão do transmissor é sempre baseada na faixa máxima do sensor utilizado, mesmo quando uma faixa intermediária foi configurada. Exemplo:

O sensor Pt100 tem faixa máxima de –200 a +650 °C e precisão total de 0,2 %.

Logo, podemos ter um erro de até 1,7 °C (0,2 % de 850 °C)

Este erro é possível em uma faixa ampla com a máxima (-200 a 650 °C) ou em uma faixa mais estreita definida pelo usuário com 0 a 100 °C.

Nota: Quando efetuadas aferições no transmissor, observar se a corrente de excitação de Pt100 exigida pelo calibrador utilizado é compatível com a corrente de excitação de Pt100 usada no transmissor: 0.18 mA.

TRANSMISSOR DE TEMPERATURA

TxRail 0-10Vdc

MANUAL DE OPERAÇÃO – V1.2x

NOVUS
WWW.NOVUS.COM.BR

Rua Álvaro Chaves, 155 CEP 90220-040 Porto Alegre - RS
Fone: (51) 3323-3600 Fax: (51) 3323-3644

Rua Paris, 856 Vila Madalena CEP 01257-040 - São Paulo SP
Fone: (11) 3675-0366 Fax: (11) 3675-0377

e-mail: info@novus.com.br

GARANTIA

O fabricante assegura ao comprador de seus equipamentos, identificados pela nota fiscal de compra, uma garantia de doze meses, nos seguintes termos:

- O período de garantia inicia a partir da data de emissão da Nota Fiscal, fornecida pela Novus.
- Dentro do período de garantia, mão-de-obra e componentes aplicados em reparos de defeitos ocorridos em uso normal, serão gratuitos.
- Para os eventuais reparos, enviar o equipamento, juntamente com as notas fiscais de remessa para conserto, para o endereço do fabricante. Despesas e riscos de transporte, ida e volta, correrão por conta do comprador.
- Mesmo no período de garantia serão cobrados os consertos de defeitos causados por choques mecânicos ou exposição do equipamento a condições impróprias de temperatura e umidade.

INTRODUÇÃO

O TxRail é um transmissor de temperatura com saída de tensão de 0 a10 Vdc, para montagem em trilhos tipo 32 mm e 35 mm. Permite ao usuário configurar facilmente o sensor e a faixa de medição de temperatura que serão utilizados no processo.
O sinal de saída tem comportamento linear em relação a temperatura medida pelo sensor. **Não** tem isolação elétrica entre entrada e saída.

ESPECIFICAÇÕES

Entrada de sensor: Configurável. Os sensores aceitos estão listados na Tabela 1, com as respectivas faixas de medida.

Termopares: Tipos J, K, R, S, T, N e E, conforme NBR 12771.
Impedância >> 1 MΩ

Pt100: Tipo 3 fios, Excitação de 0.18 mA, α= 0.00385, Conforme NBR 13773.
Para utilizar Pt100 2 fios, interligar terminais 2 e 3.

Tensão: 0 a 50 mVdc. Impedância >> 1 MΩ

Tipo de Sensor	Faixa Máxima de Medição	Faixa Mínima de Medição de:
Termopar K	0 a 1370 °C	100 °C
Termopar J	0 a 760 °C	100 °C
Termopar R	0 a 1760 °C	400 °C
Termopar S	0 a 1760 °C	400 °C
Termopar T	0 a 400 °C	100 °C
Termopar N	0 a 1300 °C	100 °C
Termopar E	0 a 720 °C	100 °C
Pt100	-200 a 650 °C	40 °C
Tensão	0 a 50 mV	5 mV

Tabela 01 – Sensores aceitos pelo transmissor

Saída: Tensão elétrica de 0 a 10 Vdc, tipo 3 fios; linear em relação a temperatura medida pelo sensor selecionado.

Resolução da Saída: 0,0025 V (12 bits)

Corrente Máxima na saída: 2 mA

Precisão Total: Erro máximo 0,3 % da faixa máxima para termopares, 0,2 % da faixa máxima para Pt100 e tensão (para tensão de saída acima de 100 mV);

Tempo de Resposta: <100 ms

Alimentação: 18 a 35 Vdc, tensão sobre o transmissor;

Temperatura de Operação: -40 a 85 °C

Umidade Ambiente: 0 a 90 % UR

Grau de Proteção: IP40

Compatibilidade Eletromagnética: EN 50081-2, EN 50082-2

Não apresenta isolamento elétrico entre entrada, saída e alimentação.

Proteção interna contra inversão da polaridade da tensão de alimentação.

Compensação interna de junta-fria para termopares.

CONFIGURAÇÃO

Para o modelo já configurado com sensor e faixa adequados não é necessária nenhuma intervenção e sua instalação pode ser executada imediatamente. Quando uma alteração na configuração é necessária, esta é realizada no **software TxConfig** e então enviada ao transmissor com o auxílio da **Interface TxConfig**.

A interface TxConfig contém um circuito eletrônico que converte os níveis dos sinais da RS232 do PC para níveis aceitos pelo produto. Não utilize qualquer outra interface ou cabo de ligação à RS232, pois o produto será danificado e este dano não é coberto pela garantia.

Interface e software TxConfig compõem o **Kit de Configuração do Transmissor** que pode ser adquirido junto a Novus Produtos Eletrônicos Ltda. ou em seus representantes autorizados. O software poder ser atualizado gratuitamente no website do fabricante. Para a instalação executar o arquivo **Tx_setup.exe** e seguir as instruções indicadas.

Erro de configuração da porta serial pode ocorrer quando outros softwares utilizam a mesma porta serial. Finalizar todos os softwares que utilizam a porta serial especificada para o TxConfig antes de carregá-lo.

O cabo da interface TxConfig tem 1,5 metro de comprimento. Uma de suas extremidades é conectada ao transmissor conforme Figura 01. A outra extremidade possui conector DB9 fêmea, que deve ser conectado à porta serial disponível no computador.

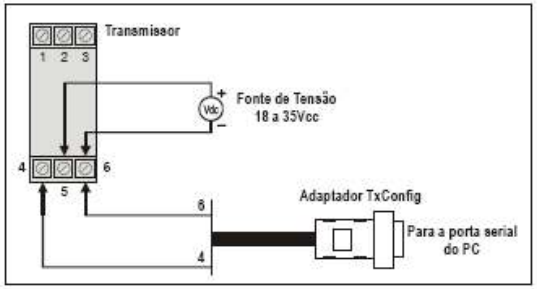


Figura 01 – Conexões do Adaptador TxConfig

Durante a configuração, o transmissor precisa ser alimentado por fonte de tensão elétrica de 18 a 35 Vdc.

Após estas conexões o usuário deve executar o software **TxConfig** e, se necessário, utilizar o tópico *Ajuda* para providenciar a configuração do transmissor.

A Figura 03 mostra a tela principal do software TxConfig.

Figura 03 – Tela principal do software TxConfig

Os campos desta tela têm as seguintes finalidades:

1. **Seleção do sensor**: Selecionar o sensor a ser utilizado. Ver Tabela 01.
2. **Faixa de medida:** Definir a faixa de medição do transmissor.

   Limite Inferior de Faixa corresponde à temperatura desejada para a tensão de **0** Vdc.

   Limite Superior de Faixa corresponde à temperatura desejada para tensão de **10** Vdc.

   Quando o Limite Inferior é definido com valor maior que o valor de Limite Superior, a tensão de saída tem variação reversa, operando de 10 a 0 Vdc.

   Os valores escolhidos não podem ultrapassar a **Faixa do Sensor** mostrada neste mesmo campo e, também, não podem estabelecer faixa com largura (*span*) menor que o valor de **Faixa Mínima** indicada mais abaixo neste mesmo campo. Ver Tabela 1 deste manual.
3. **Otimização de Filtragem**: Filtrar as medidas feitas pelo transmissor eliminando interferências vindas da rede elétrica que alimenta o processo.
4. **Falha de Sensor**: Estabelecer o comportamento da saída diante de problemas apresentados pelo sensor. Quando selecionado **Mínimo** a tensão de saída vai para 0 Vdc (*down-scale*), tipicamente utilizado em refrigeração. Quando selecionado **Máximo**, vai para 10 Vdc (*up-scale*), tipicamente utilizado em aquecimento.
5. **Correção de Zero**: Corrigir pequenos erros apresentados pelo transmissor, por exemplo, quando da troca de sensor. Ver item **Operação** neste manual.
6. **Informações do transmissor**: Neste campo constam dados que identificam o transmissor. Estas informações devem ser repassadas ao fabricante em eventuais contatos técnicos.
7. **Ler Configuração**: Quando pressionado, permite leitura da configuração presente no transmissor conectado.
8. **Enviar Configuração**: Quando pressionado, permite enviar a nova configuração ao transmissor conectado.

Nota: Se no pedido de compra o usuário não define uma configuração específica, a seguinte configuração será adotada:

- Sensor Pt100, faixa 0 a 100 °C, 0 °C de correção de zero.
- Filtro para 60 Hz e saída em máximo para falhas de sensor.

ANEXO B – MANUAL SENSOR DE CORRENTE,CRK AUTOMAÇÃO INDUSTRIAL

CRK AUTOMAÇÃO

Catálogo Técnico

Sensor Detector de Corrente
CRK–SC–001

1. DESCRIÇÃO

O sensor detector de corrente CRK foi desenvolvido para proteger sistemas onde a fase é de extrema importância, evitando comprometer resultados finais em processos caso houver anomalias na rede de alimentação, detectando falta de corrente elétrica. *Exemplos: resistências elétricas queimadas, etc.* Seu relé interno comutará, desligando o sistema sob proteção sempre que houver anormalidade na rede monitorada.

2. NOMENCLATURA

CRK–SC–001

- 001 = Versão
- SC = Sensor Detector de Corrente
- CRK = Fabricante

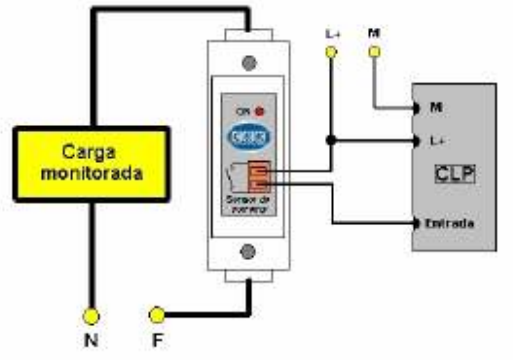
3. MODO DE OPERAÇÃO

Trabalha com correntes de 3, 10, 15 e 20 A. (Correntes especificadas pelo cliente). Conectando-se a fase nas extremidades do sensor, o relé arma. Liga um led vermelho no indicador "ON".

Quando ocorrer a queda de uma das fases conectadas ao sensor (queda de mais ou menos 30%), o relé de saída desarma, abrindo seu contato. Quando restabelecida a anomalia o relé rearma automaticamente.

4. ESQUEMA DE LIGAÇÃO

Para exemplificar uma aplicação o exemplo de ligação mostra uma carga sendo monitorada pelo sensor e um CLP (Controlador Lógico Programável) verificando a presença de corrente elétrica na carga.



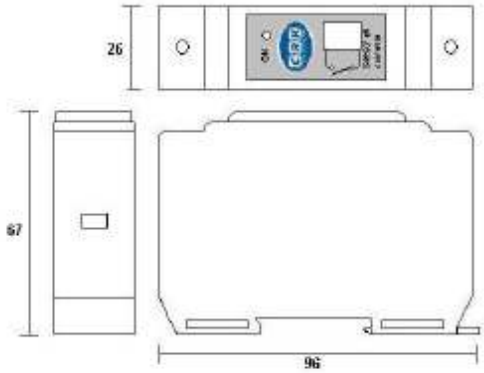
5. ESPECIFICAÇÕES TÉCNICAS

Correntes de monitoramento	3, 10, 15, 20 A
Tensão de alimentação	Componente alimentado pela carga monitorada
Corrente do contato	3 A
Tensão do contato	250 VDC / 230 VAC
Potência do contato	60 W / 120 VA
Ambiente de operação	0 a 55° C, umidade 20 a 85%
Peso	120 g

6. DIMENSÕES

Caixa plástica em ABS; medidas 26 X 67 X 96 mm (comprimento X largura X altura); Bornes com parafuso imperdível M5; encaixe para fixação em trilho Din 35 mm ou com presilhas tipo disjuntor.

MARCOS HENRIQUE RIBACKI
Universidade Luterana do Brasil



*Escala em milímetros

7. GARANTIA

A CRK Automação Industrial LTDA, concede garantia a este produto por ela fabricado pelo período de 12 meses contados a partir da data de emissão da nota fiscal de venda ao consumidor, desde que o mesmo tenha sido instalado e utilizado conforme as orientações contidas no manual de instruções.

Durante o período estipulado, a garantia cobre a mão de obra e as peças utilizadas no reparo de defeitos devidamente constatados como sendo de fabricação. Somente um técnico credenciado e pertencente à CRK Automação Industrial LTDA está habilitado a reparar defeitos cobertos pela garantia.

A GARANTIA PERDE QUANDO:

A instalação ou a utilização do produto estiver em desacordo com as recomendações existentes no manual de instruções.

O produto sofrer qualquer dano provocado por acidente, queda, agente da natureza, maus tratos, ou ainda alterações ou consertos realizados por pessoas não autorizadas pelo fabricante.

Os defeitos forem provocados pela utilização de alimentação elétrica imprópria, ou sujeita a flutuação excessiva.

A GARANTIA NÃO COBRE:

Transporte e remoção de produtos para o serviço técnico autorizado.

Despesas com viagens e estadias decorrentes do atendimento técnico.

Reparo necessário, incluindo troca de peças em caso de utilização de peças de reposição não original de fábrica e que venha prejudicar o perfeito funcionamento do produto, ou quando o defeito for causado pela alimentação com condições inadequadas.

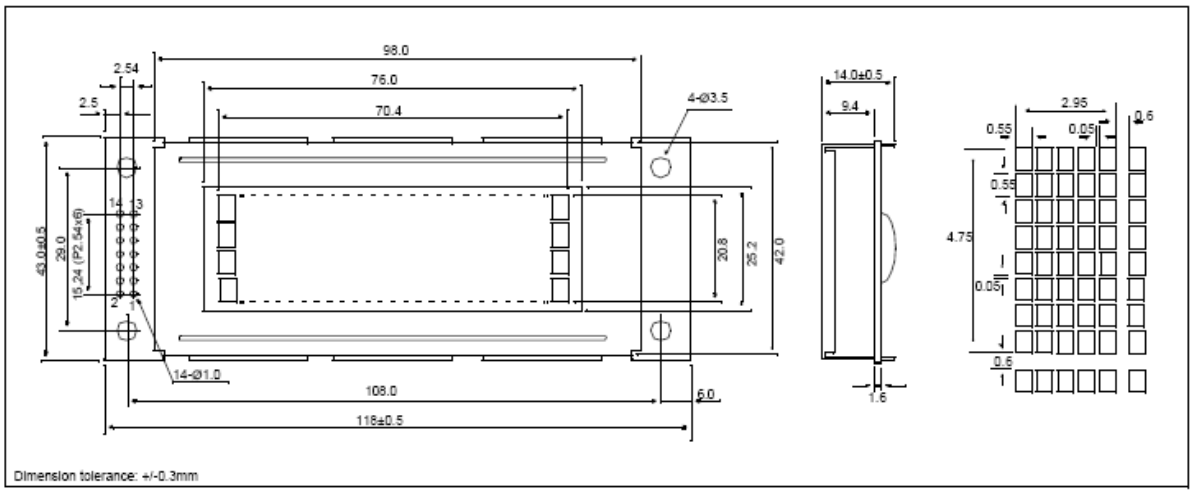
Desempenho insatisfatório do produto devido à conexão do mesmo com equipamentos de padrões e configurações incompatíveis.

MARCOS HENRIQUE RIBACKI
Universidade Luterana do Brasil

ANEXO C – DATASHEET DO DIPLAY LCD

HDM20416L-B

Dimensional Drawing — 20 Character x 4 Lines, LED Backlight



Dimension tolerance: +/-0.3mm

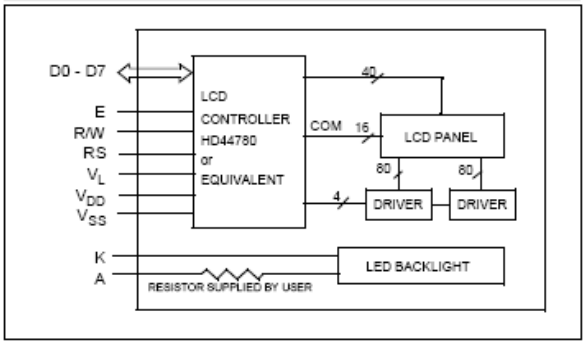
Features

Character Format ..... 5x7 Dots with Cursor
Backlight.....LED
Options.TN/Gray STN/Yellow STN, 12 o'Clock/6 o'Clock View
Normal/Extended Temperature
Normal/Negative Displays

Physical Data

Module Size.....118.0W x 43.0H x 14.0T mm
Viewing Area Size.....76.0W x 25.2H mm
Weight.....60g

Block Diagram



Absolute Maximum Ratings

PARAMETER	SYMBOL	MIN	MAX	UNIT
SUPPLY VOLTAGE	V_{DD}-V_{SS}	-0.3	7.0	V
SUPPLY VOLTAGE FOR LCD	V_{DD}-V_L	0	13.5	V
INPUT VOLTAGE	V_{IN}	V_{SS}	V_{DD}	V
OPERATING TEMPERATURE	T_{OP}	0	50	℃
STORAGE TEMPERATURE	T_{STG}	-20	60	℃

Electrical Characteristics (VDD=5.0±0.25V 25°C)

PARAMETER	SYM	CONDITION	MIN	TYP	MAX	UNIT
INPUT HIGH VOLTAGE	V_{IH}	-	2.2	-	-	V
INPUT LOW VOLTAGE	V_{IL}	-	-	-	0.6	V
OUTPUT HIGH VOLTAGE	V_{OH}	I_{OH}=0.2mA	2.4	-	-	V
OUTPUT LOW VOLTAGE	V_{OL}	I_{OL}=1.2mA	-	-	0.4	V
POWER SUPPLY CURRENT	I_{DD}	V_{DD}=5.0V	-	0.35	0.6	mA
POWER SUPPLY FOR LCD	V_{DD}-V_L	T_A=25°C	3.0	-	11.0	V
LED FORWARD VOLTAGE	V_F	I_F=210mA	3.9	4.1	4.5	V
BRIGHTNESS	L	I_F=210mA	-	250	-	cd/m²
DRIVE METHOD	1/16 Duty					

Pin Connections

PIN NO.	SYMBOL	LEVEL	FUNCTION
1	V_{SS}	-	0V — Power supply
2	V_{DD}	-	5V — Power supply
3	V_L	-	- — Power supply
4	RS	H/L	H: Data input L: Instruction data input
5	R/W	H/L	H: Data read L: Data write
6	E	H,H→L	Enable signal
7	D0	H/L	Data bus
8	D1	H/L	Data bus
9	D2	H/L	Data bus
10	D3	H/L	Data bus
11	D4	H/L	Data bus
12	D5	H/L	Data bus
13	D6	H/L	Data bus
14	D7	H/L	Data bus
15	K	-	Anode for LED backlight
16	A	-	Cathode for LED backlight

ANEXO D – DATASHEET ADS 7824

BURR-BROWN®

ADS7824

www.burr-brown.com/databook/ADS7824.html

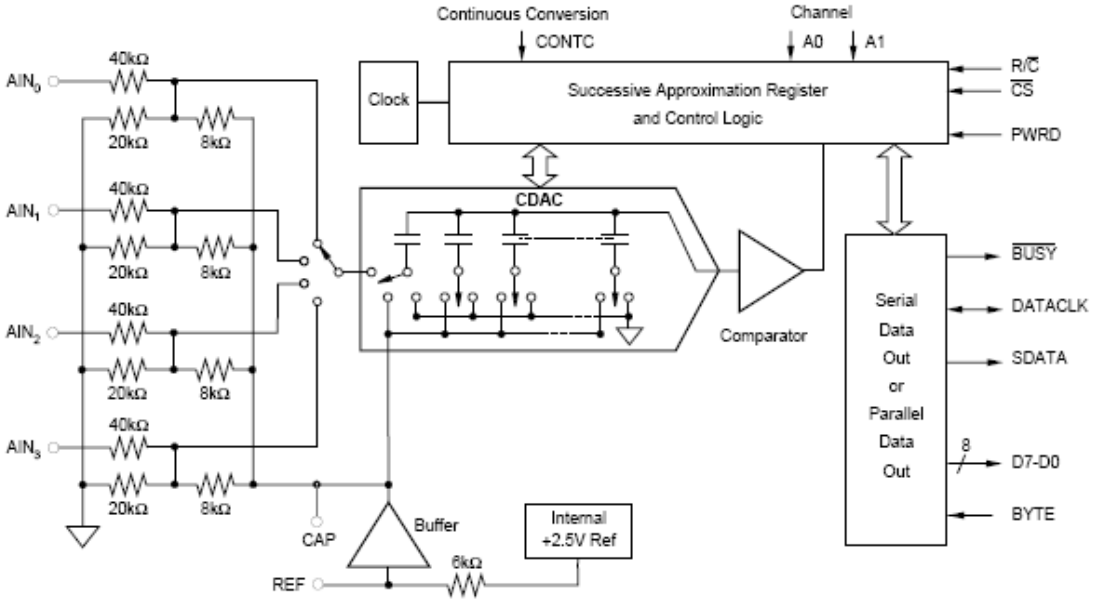
4 Channel, 12-Bit Sampling CMOS A/D Converter

FEATURES

- 25µs max SAMPLING AND CONVERSION
- SINGLE +5V SUPPLY OPERATION
- PIN-COMPATIBLE WITH 16-BIT ADS7825
- PARALLEL AND SERIAL DATA OUTPUT
- 28-PIN 0.3" PLASTIC DIP AND SOIC
- ±0.5 LSB max INL AND DNL
- 50mW max POWER DISSIPATION
- 50µW POWER DOWN MODE
- ±10V INPUT RANGE, FOUR CHANNEL MULTIPLEXER
- CONTINUOUS CONVERSION MODE

DESCRIPTION

The ADS7824 can acquire and convert 12 bits to within ±0.5 LSB in 25µs max while consuming only 50mW max. Laser-trimmed scaling resistors provide the standard industrial ±10V input range and channel-to-channel matching of ±0.1%. The ADS7824 is a low-power 12-bit sampling A/D with a four channel input multiplexer, S/H, clock, reference, and a parallel/serial microprocessor interface. It can be configured in a continuous conversion mode to sequentially digitize all four channels. The 28-pin ADS7824 is available in a plastic 0.3" DIP and in a SOIC, both fully specified for operation over the industrial –40°C to +85°C range.



International Airport Industrial Park • Mailing Address: PO Box 11400, Tucson, AZ 85734 • Street Address: 6730 S. Tucson Blvd., Tucson, AZ 85706 • Tel: (520) 746-1111 • Twx: 910-952-1111
Internet: http://www.burr-brown.com/ • FAXLine: (800) 548-6133 (US/Canada Only) • Cable: BBRCORP • Telex: 066-6491 • FAX: (520) 889-1510 • Immediate Product Info: (800) 548-6132

PDS-1303B Printed in U.S.A. October, 1997