UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DIRETORIA DE GRADUAÇÃO E EDUCAÇÃO PROFISSIONAL
CURSO DE TECNOLOGIA EM MANUTENÇÃO INDUSTRIAL

TCHARLEY ALEXSANDER SEIDEL

CONTROLADOR DE TEMPERATURA UNIFORME

TRABALHO DE CONCLUSÃO DE CURSO

MEDIANEIRA
2012

TCHARLEY ALEXSANDER SEIDEL

CONTROLADOR DE TEMPERATURA UNIFORME

Trabalho de Conclusão de Curso apresentada como requisito parcial à obtenção do título de Tecnólogo em Tecnológica em Manutenção Industrial, da Coordenação do curso de Tecnologia em Manutenção Industrial, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Adriano de Andrade Bresolin

MEDIANEIRA

2012

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Graduação e Educação Profissional
Coordenação do Curso Superior de Tecnologia em Manutenção Industrial

TERMO DE APROVAÇÃO

CONTROLADOR DE TEMPERATURA UNIFORME

Por:

Tcharley Alexsander Seidel

Este Trabalho de Conclusão de Curso (TCC) foi apresentado às 19:05 h do dia 09 de Outubro de 2012 como requisito parcial para a obtenção do título de Tecnólogo no Curso Superior de Tecnologia em Manutenção Industrial, da Universidade Tecnológica Federal do Paraná, *Campus* Medianeira. Os acadêmicos foram argüidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho APROVADO.

Prof. Dr. Adriano de Andrade Bresolin UTFPR – *Campus* Medianeira (Orientador)	Prof. Me. Alberto Noboru Miyadaira UTFPR – *Campus* Medianeira (Convidado)
Prof. Me. Yuri Ferruzzi UTFPR – *Campus* Medianeira (Convidado)	Prof. Yuri Ferruzzi UTFPR – *Campus* Medianeira (Responsável pelas atividades de TCC)

A Folha de Aprovação assinada encontra-se na coordenação do Curso de Tecnologia em Manutenção Industrial.

AGRADECIMENTOS

Agradeço aos professores que sugeriram ideias relacionadas ao desenvolvimento deste trabalho, agradeço ao orientador pelo tempo e atenção dedicado ao trabalho e agradeço principalmente a minha família pela força que deram para estudar ao longo desses anos.

RESUMO

O controle de temperatura é muitas vezes necessário em alguns processos de produção. No meio fabril, geralmente, existem equipamentos conhecidos como controladores de temperatura que dispõem um controle na saída do tipo 0 e 1 (liga/desliga). Contudo, esse tipo de controle não é útil em alguns processos onde a função da carga e as temperaturas críticas são uniformes e precisam de um controle contínuo e não simplesmente liga/desliga. Este trabalho tem como objetivo o desenvolvimento de um controlador de potência automático que realize o controle da temperatura dos escamoteadores de maternidades de suínos, realizando uma variação gradual da potência cedida à carga, e conseqüentemente um controle uniforme da temperatura.

ABSTRACT

Temperature control is often required in some processes. In the middle factory, usually, there are devices known as temperature controllers that have a control on the output type 0 and 1 (on / off). However, this type of control is not useful in some cases where the function of the load and the critical temperatures are uniform and have a continuous control and not simply on / off. This work aims at the development of a power controller that performs automatic temperature control of maternity pens of pigs, performing a gradual variation of the power transferred to the load, and consequently a uniform temperature control.

LISTA DE SIGLAS

AC	Alternating Current;
A/D	Analógico / Digital;
ASES	Software;
BIT	"Binary digit", (dígito binário);
CCP	Compare/Capture/PWM;
CI	Circuito Integrado;
CPU	Central Processing Unit;
DC	Direct current;
EEPROM	Electrically-Erasable Programmable Read-Only Memory;
EPROM	Erasable Programmable Read-Only Memory;
EUA	United States of America;
GND	Graduated Neutral Density filter (filtro graduado de densidade neutra);
HZ	Hertz;
IDE	Integrated Development Environment;
INTEL	Multinacional americana de semicondutores;
ISIS	Software;
LED	Light-Emitting Diode;
LCD	Liquid Crystal Display;
NTC	Negative Temperature Coeficiente;
PIC	Programmable Interface Controller" (Controlador de Interface Programável);
PWM	Pulse-Width Modulation (modulação por largura de pulso);
RAM	Random Access Memory (Memória de acesso aleatório);
RMS	Root Mean Square;
UL	Underwriters Laboratories;
USB	Universal Serial Bus;

LISTA DE FIGURAS

LISTA DE TABELAS

Sumário

1 Introdução

A atual sociedade desfruta de um grande conhecimento tecnológico, proporcionando grande conforto ao consumidor. Contudo, a implementação dos recursos alcançados ainda não atende de forma ampla ao setor de criação de suínos.

Muitos granjeiros possuem em sua propriedade uma maternidade, e nesta um sistema de aquecimento para os leitões recém nascidos. Inicialmente, o controle da temperatura na maternidade era realizado ligando e desligando lâmpadas incandescentes, e ainda alternado as mesmas por potências diferentes. Ainda hoje existem granjas que seguem esse sistema. Outros buscaram alternativas para simplificar e facilitar o controle, instalando em suas maternidades *dimmer* de potência, mas o processo ainda é manual exigindo um acompanhamento dos funcionários ou proprietário.

Com base nessas informações, propõem-se um aparelho que efetue esse controle da temperatura dos escamoteadores automaticamente com o mesmo tipo de controle do *dimmer*, ou seja, variando a potência de forma gradual.

Este trabalho tem como objetivo geral construir um aparelho que realize o controle automático da temperatura dos escamoteadores de forma gradual. Também em relação ao objetivo do mesmo, pretende-se realizar o controle do disparo do Triac com microcontrolador, adaptar o circuito de controle manual com o automático, montar aparelho e realizar teste com carga de 1000W.

2 Motivação

Produtores de suínos relatam grande dificuldade nos cuidados com o leitão recém-nascido. O leitão recém-nascido não possui resistência térmica suficiente para manter a temperatura corporal, logo o mesmo irá tentar se aquecer junto à porca, correndo risco de ser esmagado.

Outro fato está relacionado a uma temperatura inferior da necessária no escamoteador para o leitão ficar em repouso, podendo o leitão ter uma queda na temperatura corporal iniciando uma série de conseqüências tais como, queda no consumo de colostro, queda na absorção de imunoglobulinas, queda na proteção imunitária, aumento na incidência de doenças e morte.

A figura 1 mostra um escamoteador de uma maternidade de suínos onde os leitões ficam em repouso quando não estão amamentando.

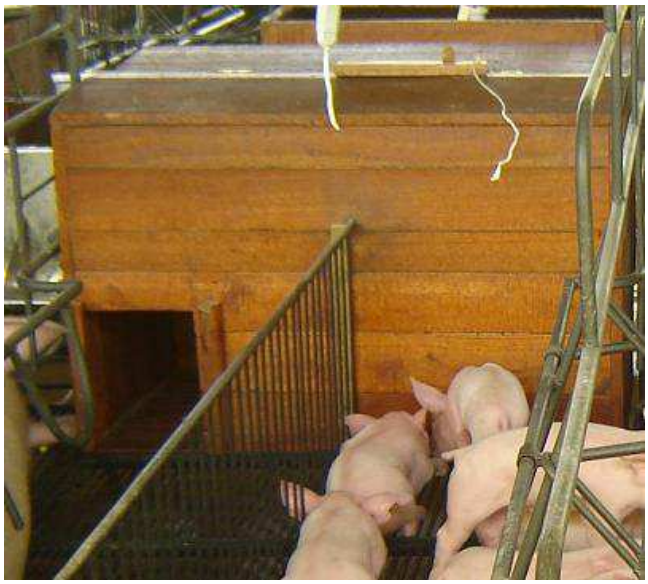

Figura 1. Escamoteador de uma maternidade de suínos
Fonte: Sossuinos (2007).

Também deve ser levado em consideração que se a temperatura do ambiente dos escamoteadores estiver muito elevada, o leitão não irá ficar dentro do escamoteador, e sim nas suas proximidades ou junto à porca. Estando o leitão nas proximidades do escamoteador, o mesmo pode perder calor corporal seguidas das conseqüências citadas acima, também mostrado de forma resumida na figura 2. [Nääs e Caldara; 2011]

Fatos como a freqüente troca de lâmpadas devido à variação de temperatura ambiente, a falta de monitoramento das condições térmicas dos escamoteadores, a falta de um controle mais amplo na potência elétrica cedida a resistência de aquecimento e a necessidade de luz permanente nos escamoteadores, são motivos de problemas freqüentes como a quebra de lâmpadas, rompimento do filamento das mesmas, desconforto térmico dos leitões entre outros problemas também relacionados à saúde do mesmo.

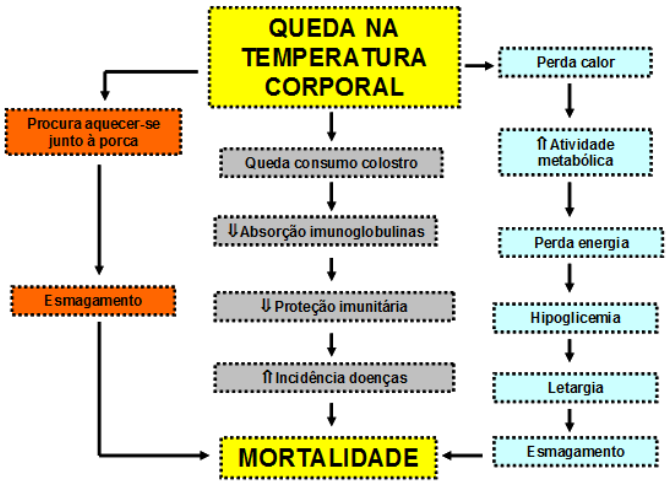

Figura 2. Esquema da influência da queda da temperatura corporal sobre a mortalidade de leitões na maternidade.
Fonte: Animalworld **(2011).**

Atualmente, com o fim da fabricação de lâmpadas incandescentes de até 150W, os suinocultores estão optando por resistências colocadas no piso e, por lâmpadas infravermelho de 250W, que até o momento, melhor atendem as condições construtivas e as necessidades existentes na maternidade.

2.1 Soluções existentes

O mercado atual dispõe de um aparelho de controle de temperatura apresentado na figura 3.

Figura 3. Controlador de temperatura
Fonte: Gsibrasil (2012)

O aparelho acima visto é fabricado para ser aplicado em aviários e pocilgas. São algumas características do mesmo:

· Possui 5 saídas de controle: 3 estágios de ventilação (mínima com o 1º grupo), 1 de nebulização (opera por ciclos de tempo) e 1 de alarme (por temperatura alta, baixa, falta de energia e falha do sensor);
· Fácil operação e programação por seletor rotativo e teclado;
· Controle somente de temperatura;
· Sonda de temperatura enclausurada (sensor NTC), à prova de imersão e com ótima resistência mecânica;
· Permite a calibração do sensor de temperatura;
· A programação é direta através da chave giratória realizando a seleção da função a alterar, com a tecla PROG se libera a alteração e com as teclas ACIMA e ABAIXO se altera o valor para o desejado.
· Tensão de alimentação (V) = 220/254V +/ -20%
· Freqüência (Hz) = 50/60 Hz
· Temperatura de controle (°C) = 10 a 40 resolução 0,1°C
· Temperatura de indicação (°C) = 0 a 45 resolução 0,1°C
· Comandos = 5 saídas independentes, a relé (2,5A por est.) [Gsibrasil,2012]

Contudo, o aparelho visto acima possui um controle baseado no acionamento de relé na saída, logo, não pode realizar um controle da potência cedida à carga de forma gradual.

Abaixo segue outros exemplos de controladores de temperatura disponíveis a venda no mercado, porém com o mesmo sistema de controle de saída. A figura 4 apresenta o controlador modelo DTB4848, com 8 rampas e 8 patamares, todas entrada de sinal configuráveis. Saída de controle de tensão à relé e relé de alarme.

A figura 5 apresenta o controlador de temperatura e umidade relativa N322RHT, possui duas saídas de controle do tipo relé que podem ser configuradas independentemente para atuar como controle ou alarme.

Figura 4. Controlador de temperatura modelo DTB4848.
Fonte: Presstemp **(2012)**

Figura 5. Controlador de Temperatura N1020.
Fonte: Novus (2012)

Atualmente já existe um aparelho que atende a essas necessidades, contudo o seu controle é realizado de forma manual, exigindo do suinocultor um monitoramento constante da variação térmica diária. Os suinocultores têm optado pela aplicação deste devido a sua forma de controle, onde, através de um ajuste no potenciômetro do mesmo, é realizado uma variação gradual da potência cedida á uma carga de até, aproximadamente, 30 A.

2.2 Solução apresentada

Buscando reduzir ao máximo os problemas acima citados, propõe-se um aparelho que realize o controle da temperatura ambiente dos escamoteadores com monitoramento contínuo e mais preciso, onde o mesmo realizará o controle de forma automática.

A solução encontrada que melhor atende a necessidade da maioria dos granjeiros se resume em um aparelho que realize o controle automático de duas linhas de escamoteadores individualmente, e que tenha como opcional também o controle manual para o caso de algum problema ocorrer. O mesmo terá de quatro botões principais onde poderá ser selecionada a faixa de temperatura que melhor atende a fase de crescimento do leitão e, de um display para comunicação com o operador.

Também parte do sistema de controle, os sensores de temperatura instalados em um ou mais escamoteadores, objetivando a leitura da temperatura do ambiente do escamoteador para posteriormente ser realizado o controle da potência cedida à carga que será uma resistência.

O controle da temperatura será realizado com base do sinal gerado pelo PWM do microcontrolador. A variação do tempo de duração dos pulsos gerados pelo PWM, resultará em uma tensão média, onde esta por sua vez, referenciará o ângulo do disparo do Triac realizado pelo TCA785.

3 Sistema de controle de temperatura

O funcionamento do controlador de temperatura parte do principio do modelo de um *dimmer* simples, sendo a potência variada com o uso de um Triac. Contudo, o controlador de temperatura compreende funções mais complexas, deste modo utilizará um microcontrolador para realizar todas as funções cabíveis ao processo do controle automatizado da temperatura.

O controlador de temperatura contará com um sensor instalado em um escamoteador selecionado, onde, por meio do mesmo, será identificada a temperatura do ambiente do escamoteador e, em comparação com os dados contidos no programa do microcontrolador, a execução do controle do disparo do gate do Triac.

Tabela 1. Tabela de temperatura x fase do suíno.

Fase	Temperatura de conforto (°C)	Temperatura critica inferior (°C)	Temperatura critica superior (°C)
Recém-nascidos	32-34	-	-
Leitões até a desmama	29-31	21	36
Leitões desmamados	22-26	17	27
Leitões em crescimento	18-20	15	26
Suínos em terminação	12-21	12	26
Fêmeas gestantes	16-19	10	24
Fêmeas em lactação	12-16	7	23
Fêmeas vazias e machos	17-21	10	25

Fonte: Animalworld (2011)

Como pode ser observado na tabela 1, para cada fase de crescimento do leitão, a faixa de temperatura que atende a zona de conforto do suíno varia. De acordo com a tabela, leitões recém-nascidos precisam de 32 a 34 graus centigrados, leitões até a desmama de 29 a 31 graus centigrados, desmamados de 22 a 26 graus centigrados e leitões em crescimento de 18 a 20 graus centigrados.

Nesse projeto, as faixas de temperaturas usadas são as respectivamente citadas acima. Ao contrário das demais fases de crescimento, os leitões na fase de recém-nascidos possuem uma faixa de temperatura de conforto mais rígida, onde esta não possui uma temperatura crítica inferior e superior.

Sendo bastante variável a fase dos suínos na maternidade, propõe-se o uso de quatro botões na interface do aparelho, onde cada botão representará uma faixa

de temperatura definida no programa, onde, conforme o crescimento do leitão pode-se escolher a faixa de temperatura que melhor se adapta a fase do mesmo.

3.1 Microcontrolador PIC

A história do microcontrolador começa no ano de 1969, uma equipe de engenheiros japoneses chega aos EUA com a encomenda de alguns circuitos integrados para calculadoras a serem implementados segundo os seus projetos. Sendo a proposta entregue a INTEL, a mesma apresentou uma solução substancialmente diferente. A solução pressupunha que a função do circuito integrado seria determinado por um programa nele armazenado.

Depois de algum tempo, embora os engenheiros japoneses tenham tentado encontra uma solução mais fácil, a idéia da INTEL venceu e o primeiro microprocessador nasceu. Em 1971, a INTEL adquiriu os direitos sobre a venda deste bloco integrado, e neste mesmo ano apareceu no mercado um microprocessador designado por 4004, sendo este o primeiro microprocessador de 4 bits com velocidade de 6000 operações por segundo. [Morellato,2012]

Os microcontroladores PIC são fabricados pela MICROCHIP, a qual iniciou seus negócios no Brasil em 1990, permanecendo o mesmo até hoje. A empresa foi fundada em 1987, quando a General Instrument separou sua divisão de microeletrônica como uma subsidiária integral. A Microchip Technology se tornou uma empresa independente em 1989, quando foi adquirida por um grupo de capitalistas ousado, e veio a público em 1993.[Cary, 2012]

O microcontrolador é uma única pastilha de silício encapsulada, nele tem-se todos os componentes necessários ao controle de um processo. Atualmente, muitos equipamentos de nosso uso diário, tais como: eletrodomésticos, CD Player, alarmes, celulares e brinquedos, entre outros, utilizam microcontroladores para execução de suas funções.

Os microcontroladores PIC apresentam uma estrutura de máquina interna do tipo Harvard, onde existem dois barramentos internos, sendo um de dados e outro de instruções. Esse tipo de arquitetura permite que, enquanto uma instrução é executada, outra seja buscada da memória, o que torna o processamento mais rápido. [Souza, 2008]

Um microcontolador é um sistema computacional completo, no qual estão incluídos uma CPU (Central Processor Unit), memória, um sistema de *clock*, sinais I/Os (Input/Output), além de outros possíveis periféricos, tais como, módulos de temporização e conversores A/D entre outros, integrados em um mesmo componente (chip). As partes mais integrantes de qualquer computador, e que também estão presentes, em menor escala, nos microcontroladores são:

a) Unidade Central de Processamento (CPU);
b) Sistema de Clock[1] para dar seqüência às atividades da CPU;
c) Memória para armazenamento de instruções e manipulação de dados;
d) Entradas para interiorizar na CPU informações do mundo externo;
e) Saídas para exteriorizar as informações processadas pela CPU para o mundo externo;
f) Programa (Software[2]) para que o sistema execute uma função. [Denardin,2008]

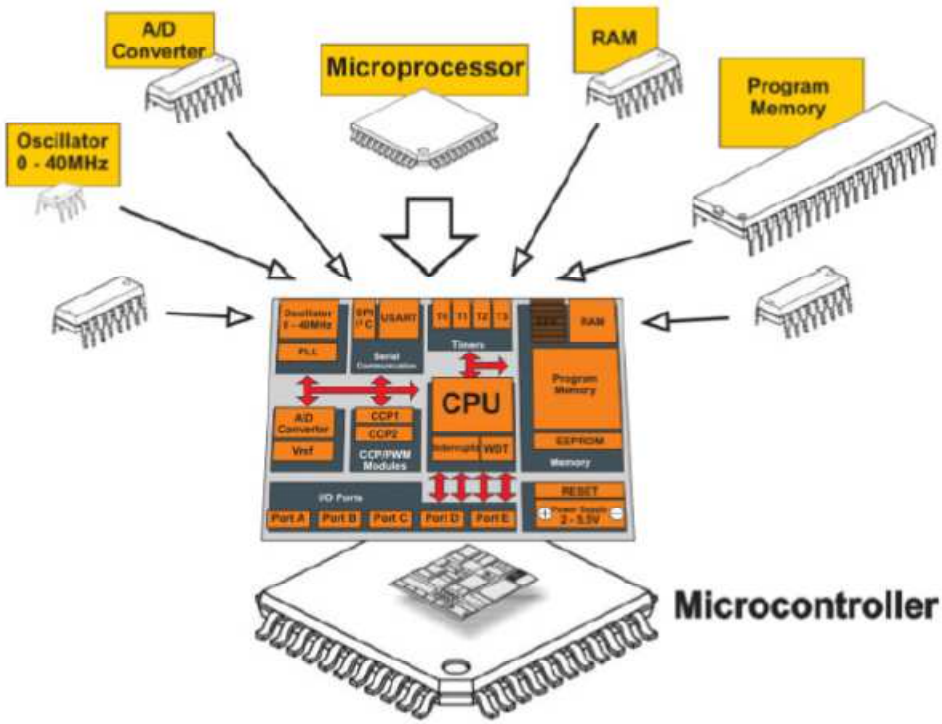

Figura 6. Arquitetura geral de um microcontrolador
Fonte: Seródio (2011)

[1] Fornece a velocidade de execução das instruções do programa pelo microprocessador.
[2] Sistemas microprocessados operam segundo a execução seqüencial de instruções e operando amarzenados em memória.

A figura 6 mostra de forma simplificada os componentes necessários ao controle de um processo incluso em uma única pastilha de silício, formando um sistema completo chamado microcontrolador.

O microcontrolador aplicado neste projeto é o PIC18F4550 que possui muitas funcionalidades. As características desse microcontrolador são:

· Memória de programação e de dados que permitem apagamento e reescrita de valores milhares de vezes;

· É auto programável, isto é, através de um controle interno por software, o microcontrolador pode escrever na sua própria memória de programa. Usando uma rotina de *bootloader*, alocada no bloco *Boot* no topo da memória de programa, é possível atualizar a aplicação em campo, sem a utilização de um gravador.

· 1 Módulo *Universal Serial Bus* (USB). O PIC18F4550 possui um USB SIE (*Serial Interface Engine*) compatível com *full-speed* e *low-speed* USB o que possibilita a rápida comunicação entre um *host* USB e o microcontrolador.

· 13 Conversores analógico-digitais de 10 bits;

· Memória de programa *Flash* com 32 *Kbytes*;

· Memória de dados EEPROM 256 *Bytes*;

· Memória RAM 2 *Kbytes*;

· Freqüência de operação até 48 MHz;

· Portas bidirecionais de entrada e saída: A, B, C, D e E;

· 4 *Timers*;

· 1 Módulo *Capture*/*Compare*/PWM;

· 2 Comparadores;

· 1 *Streaming Parallel Port* (SPP). [Labtools, 2012]

3.2 O CI TCA 785

O circuito integrado monolítico analógico TCA 785 com 16 pinos disponíveis é fabricado pela Icotron S/A Indústria de Componentes Eletrônicos. Entre várias aplicações gerais, é dedicado à aplicação de controle de ângulo de disparo de tiristores (triacs) continuamente de 0° a 180°. Sua configuração interna possibilita uma simplificada seleção de componentes externos para chaveamento, sem tornar muito volumoso o circuito final.

Devido a sua versatilidade, permite inúmeras aplicações dentro da eletrônica, apesar de um componente dedicado à construção de circuitos de disparos para tiristores em geral.

Descrição do seu Funcionamento

• Consumo interno de corrente, apenas 5 miliampéres;

• Possibilidades de inibição simultânea de todas as saídas;

• Operação em circuitos polifásicos, utilizando, mais de um TCA ligados em paralelo;

• Duas saídas principais (corrente até 55 miliampéres) e duas em coletor aberto (corrente até 1,5 miliampéres);

• Uma saída para controle de triacs;

• Duração dos pulsos de saída determinada pela colocação de um capacitor externo;

• Saída de tensão regulada em 3,1V.

Internamente, o integrado é alimentado por uma tensão regulada de 3,1V, independente das variações possíveis em sua alimentação externa, estimada entre 8 e 18V. Essa tensão de 3,1V pode ser obtida no pino 8. O sincronismo é obtido através de um detector de zero (pino 5), conectado a um registrador de sincronismo.

A figura 7 mostra a representação do CI TCA785 e a sua respectiva pinagem.

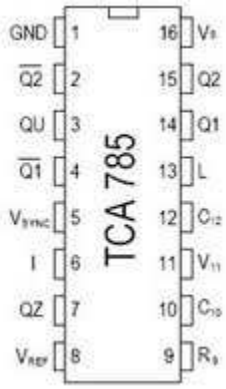

Figura 7. TCA 785 pinagem
Fonte: Guzella **(2008)**

A figura 8 mostra de forma resumida as principais formas de onda do TCA785. A senóide de V5 representa a tensão de referência para o carregamento do capacitor interno com tensão V10. O capacitor é carregado a cada semiciclo e, ao atingir a tensão de V11, envia pulsos nos pinos 14 e 15. A variação do ponto no

tempo do semiciclo em que é emitido um pulso está diretamente ligado a variação da tensão de V11.

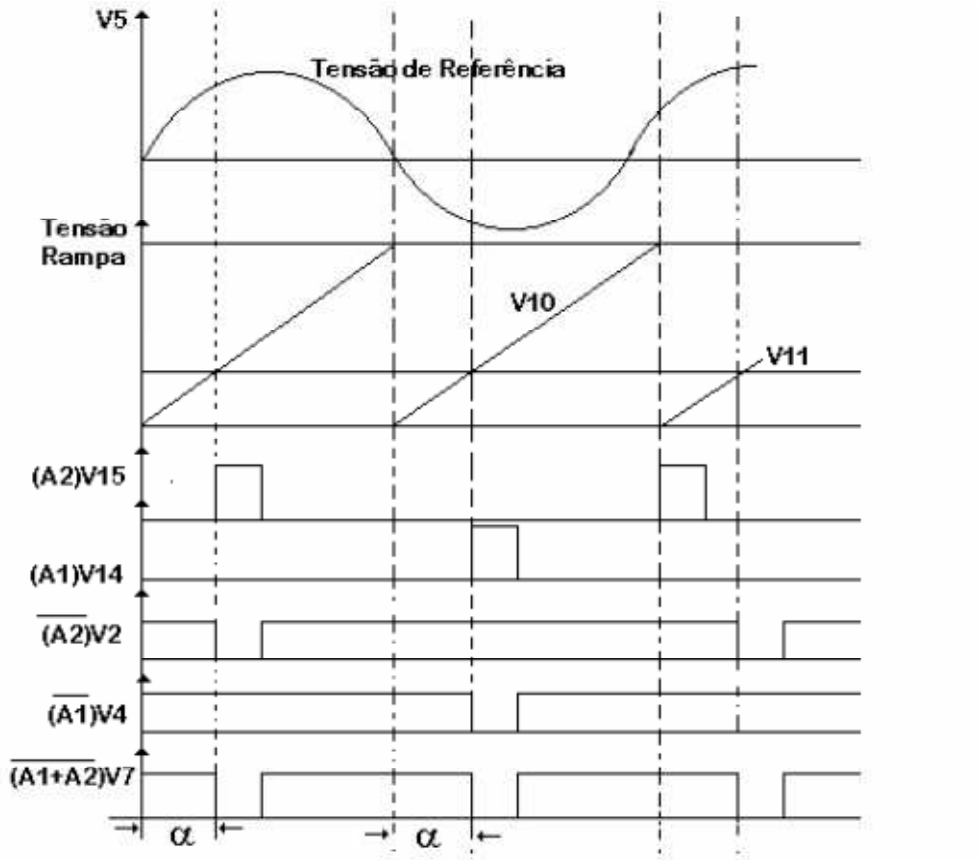

Figura 8. Principais formas de onda do TCA785
Fonte: Guima (2011)

O gerador de rampa, cujo controle está na unidade lógica, provém de uma fonte de corrente constante carregando o capacitor o C10, corrente essa controlada pelo potenciômetro P9, cuja finalidade é ajustar a amplitude da rampa, que vai a zero sempre que a tensão de sincronismo passa por zero, devido a saturação de um transistor em paralelo com o capacitor. O comparador de controle compara a tensão de rampa com a tensão de controle; quando essas forem iguais envia pulsos nas saídas via unidade lógica. Obtêm-se, então, nos pinos 14 e 15, pulsos positivos para os semiciclos negativo e positivo da tensão de sincronismo defasado entre si em 180º.

Esses pulsos têm suas larguras determinadas pela conexão de um capacitor externo, C12, entre o pino 12 e o terra, e amplitudes iguais à tensão de alimentação do pino 16. O pino 6 quando aterrado pôr um relé ou um transistor PNP inibe todas as saídas do TCA 785, servindo de proteção para o sistema. [Guima, 2011]

3.3 O CI LM35DZ

O sensor LM35 é um sensor de precisão, fabricado pela National Semiconductor[3], que apresenta uma saída de tensão linear relativa à temperatura em que ele se encontrar no momento em que for alimentado por uma tensão de 4-20Vdc e GND, tendo em sua saída um sinal de 10mV para cada Grau Celsius de temperatura. Apresenta uma boa vantagem com relação aos demais sensores de temperatura calibrados em “KELVIN”, não necessitando nenhuma subtração de variáveis para que se obtenha uma escala de temperatura em Graus Celsius.

A figura 9 mostra o CI LM35DZ, circuito integrado de precisão, que realiza o sensoriamento da temperatura do ar em escala termométrica Celsius.

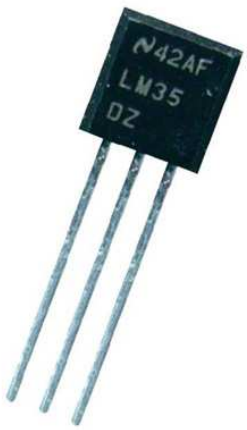

Figura 9. CI LM35DZ, sensor de temperatura.
Fonte: Multilogica (2012)

O LM35 não necessita de qualquer calibração externa para fornecer com exatidão, valores temperatura com variações de ¼ºC ou até mesmo ¾ºC dentro da faixa de temperatura de –55ºC à 150ºC. Este sensor tem saída com baixa impedância, tensão linear e calibração inerente precisa, fazendo com que o

[3] National Semiconductor: Empresa de semicondutores

interfaceamento de leitura seja especificamente simples, barateando todo o sistema em função disto.

Este sensor poderá ser alimentado com alimentação simples ou simétrica, dependendo do que se desejar como sinal de saída, mas independentemente disso, a saída continuará sendo de 10mV/ºC. Ele drena apenas 60µA para estas alimentações, sendo assim seu auto-aquecimento é de aproximadamente 0.1ºC ao ar livre. [Nicoleti, 2011]

3.4 CI LM358N

O CI LM358N é um chip que possui dois amplificadores operacionais (op amp) vide figura 10, cuja função principal é a amplificação da tensão. Podem ser alimentados por uma fonte simples (não simétrica) que pode ir de 3 a 32 volts. Sua saída fica sempre em uma faixa de voltagens que se estende de um valor um pouco acima de zero até cerca de 2 volts abaixo da tensão de alimentação +V.

Foram projetados especificamente para operar a partir de uma única fonte de alimentação sobre uma ampla gama de tensões. A baixa fuga de energia é independente da magnitude da tensão de alimentação.

Sua aplicação atual tem se tornado vasta devido a suas características de funcionamento se aproximar muito das ideais. Circuitos amplificadores de áudio, circuitos de instrumentação entre outros são exemplos de aplicação. [Ferreira,2003]

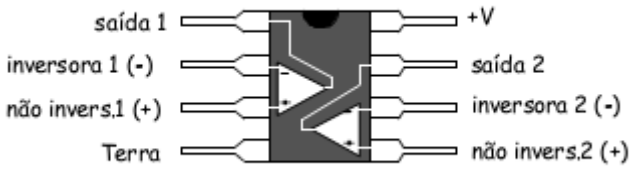

Figura 10. CI LM358 amplificador operacional
Fonte: Centelhas **(2009)**

3.5 O CI BTA41

Disponível em encapsulamento de alta potência, a série BTA / BTB40-41 é adequado para uso geral na comutação de energia AC. Eles podem ser usados na função ON/OFF, e no controle de potência através da condução parcial da senóide da rede, em aplicações tais como relés estáticos, regulação de aquecimento,

aquecedores de água, motores de indução, circuitos de partida, em equipamentos de solda ou para a operação de controle de alta potência.

Ao utilizar um bloco interno de cerâmica, a série BTA fornece guia isolado da alta tensão (avaliado em 2500V RMS), de acordo com as normas UL (ref. Arquivo.:E81734).

A figura 11 mostra o Triac BTA41 600B, desenvolvido para suportar até 40 ampér com isolação de 600V. [STMicroeletronics, 2001]

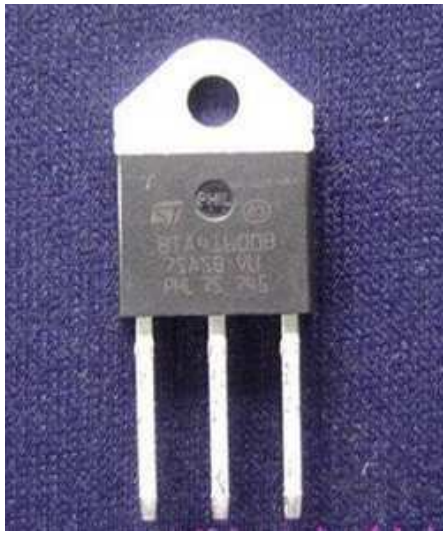

Figura 11. Triac BTA41 600B
Fonte: Ebayimg (2012)

3.6 O CI MOC 3021

Os óptos-acopladores possuem um receptor que funciona por polarização através de luz emitida por um emissor, geralmente um diodo emissor de luz (LED). Este tipo de componente isola os circuitos de outros com tensões elevadas (ordem dos 800V de pico) protegendo assim o circuito com tensões mais fracas.

O CI MOC3021 é composto por um diodo emissor de luz infravermelho (LED) que, através da luz que emite, coloca o foto-triac no estado de condução. A condução do Triac do CI MOC3021 deixa passar as duas alternâncias da fonte alternada.

A figura 12 mostra o CI MOC3021, aplicado geralmente em circuitos que se deseja isolar o circuito de controle do circuito de potência. [Pereira,Cordeiro,2007]

Figura 12. Opto-acoplador MOC3021
Fonte: Futurlec (2012)

3.7 LCD

O LCD caractere 16x2, ou seja, dezesseis caracteres por duas linhas é um dos LCDs mais utilizados em equipamentos eletrônicos. A Figura13 mostra um típico LCD de 16 x 2.

O HD4478 possui dois registradores de 8bits, um registrador de instrução (IR) e um registrador de dados (DR). O registrador de instrução é responsável pelas operações de configuração, rolagem de tela, posicionamento do cursor, entre outras. O registrador de dados contém o dado que deve ser escrito no LCD.

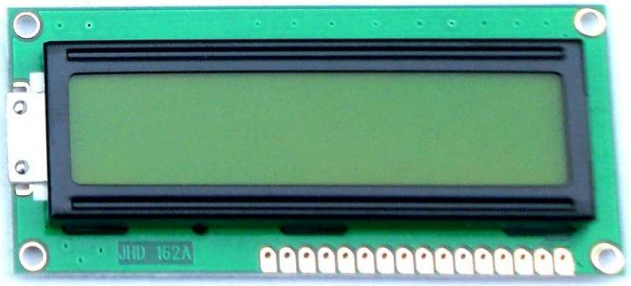

Figura 13. Display LCD 16x2
Fonte: FellipeMauricio (2012)

A função de cada pino esta descrita na tabela abaixo.

Tabela 2. Descrição dos pinos LCD

PIN NO	Symbol	Function
1	VSS	GND
2	VDD	power supply for logic
3	VO	Contrast adjustment
4	RS	H/L Reagister select signal
5	R/W	H/L Read/write signal
6	E	H/L Enable signal
7	DB0	H/L Data bus line
8	DB1	H/L Data bus line
9	DB2	H/L Data bus line
10	DB3	H/L Data bus line
11	DB4	H/L Data bus line
12	DB5	H/L Data bus line
13	DB6	H/L Data bus line
14	DB7	H/L Data bus line
15	A	Power supply for BKL(+)
16	K	Power supply for BKL(-)

Fonte: Central AVR (2012)

A linha Enable (E) permite a ativação do display e a utilização das linhas R/W e RS. Quando a mesma esta em nível baixo, o LCD fica inibido e ignora os Sinais RS e R/W.

A linha Read/Write determina o sentido dos dados entre o LCD e o microcontrolador. Quando esta em nível baixo, os dados serão escritos no LCD, e quando esta em nível alto os dados serão lidos no LCD.

A linha de Register Select (RS), o LCD interpreta o tipo de dados presente nas Linhas de dados. Quando esta em nível lógico baixo, uma instrução será escrita no LCD e quando em nível lógico alto, será escrito um caráter no LCD.

Os pinos de DB0 a DB7 equivalem ao barramento de dados paralelo. Este é um barramento bidirecional, pois ele pode ser efetuado tanto para a escrita quanto para leitura dos dados armazenados na memória RAM do LCD. Apesar de existirem oito vias de dados, esses *displays* também podem operar com quatro vias (DB4 a DB7), ficando assim as demais vias sem função. Neste caso, as informações são enviadas em dois pacotes de quatro bits cada um. [Muniz. 2009]

3.8 Fonte

O CI LM7805 circuito regulador de tensão linear pertence à família LM78XX de circuitos integrados. Isto significa basicamente que o LM7805 é apenas um dos vários tipos de circuitos integrados com tensão linear fixa que pode ser instalado em vários dispositivos eletrônicos. A família LM78XX de circuito integrado é

frequentemente usada em dispositivos eletrônicos que necessitam de uma alimentação controlada.

A família LM78XX tem vários subtipos, com os dois últimos números designam a sua tensão de funcionamento. As famílias LM78XX de circuitos reguladores de tensão são todos os reguladores de tensão positiva. Há outra série de circuitos do regulador de tensão designadas como a série LM79XX, mas estes conduzem tensão negativa.

Os LM7805 variantes e outros membros da família LM78XX de circuitos reguladores de tensão têm um processo de construção simples e, sobretudo, não requerem qualquer componente extra para executar sua função básica - a tensão de regulação.

O LM7805 é um circuito regulador de tensão compacto, permitindo assim ser instalado em muitos tipos diferentes de sistemas eletrônicos. A família inteira de LM78XX circuitos reguladores de tensão também tem proteção integrada que os impede de ir ao seu limite de potência. [Lm7805, 2012]

O CI LM7805 pode ser energizado com tensão entre 7 e 25V no pino V-IN, obtendo em V-OUT, uma tensão linear de 5V, como mostra a Figura 14.

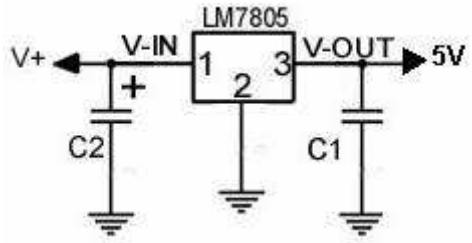

Figura 14. LM7805 regulador de tensão.
Fonte: Eletronica **(2012)**

4 Montagem e testes

Para melhor compreensão dos acontecimentos relacionados à parte prática, optou-se em dividir os mesmos em subtítulos.

4.1 Linguagem C

Na obtenção de conhecimento referente à linguagem C, recorreu-se a vários exemplos prontos para simulações e estudos. Com o auxilio de um kit de simulação da empresa Mosaico[4], foi realizado simulações entre outros testes até se obter conhecimento suficiente para a criação do programa a ser usado no projeto em questão.

O desenvolvimento do programa em linguagem C foi realizado no ambiente de desenvolvimento MPLAB[5] IDE mostrado na figura 15. Fornecido pela empresa Microchip_Technology, integra diversos ambientes de trabalho para programação, simulação e gravação de microcontroladores.

Figura 15. Imagem software MPLAB
Fonte: Tonieletrônica (2010)

a) Software Gratuito – www.microchip.com;
b) Versão Atual: 7.00;
c) Linguagem de programação: Assembly, Linguagem C;
d) Gerenciamento de projetos;
e) Compilação;

[4] Mosaico:Empresa especializada em processamento digital
[5] Software de desenvolvimento de programas.

f) Simulação;
g) Emulação;
h) Gravação do chip; {24}

4.2 Fluxograma

Em relação às primeiras ideias, o processo do aparelho se resume ao seguinte fluxograma mostrado na figura 16:

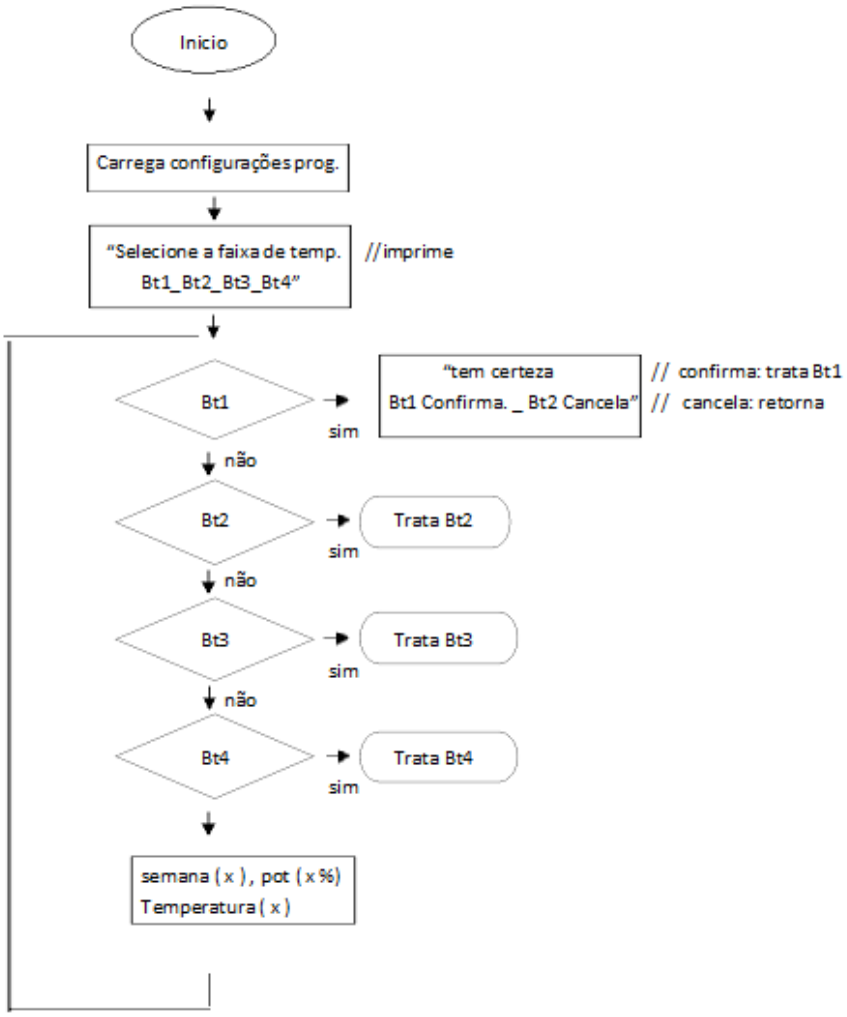

Figura 16. Primeiro fluxograma gerado
Fonte: Autoria própria

Inicialmente foi realizado uma pesquisa com o proprietário de uma granja, onde o mesmo sugeriu alternativas que melhor atenderiam o sistema de criação de suínos dele.

Entre as ideias sugeridas, propôs um controle duplo separado em uma mesma sala de criação. Esse controle compreenderia duas saídas, sendo separado o controle de potência das duas saídas, onde cada saída trabalharia com a faixa de temperatura selecionada individualmente.

No debate também foi discutido a variação de temperatura nas escamoteadores, onde a conclusão se deu em uma variação de dois graus centigrados por semana.

Após a aquisição dos dados da visita realizada na granja, gerou-se outro fluxograma mostrado na figura 17 incluindo as novas idéias até aqui obtidas.

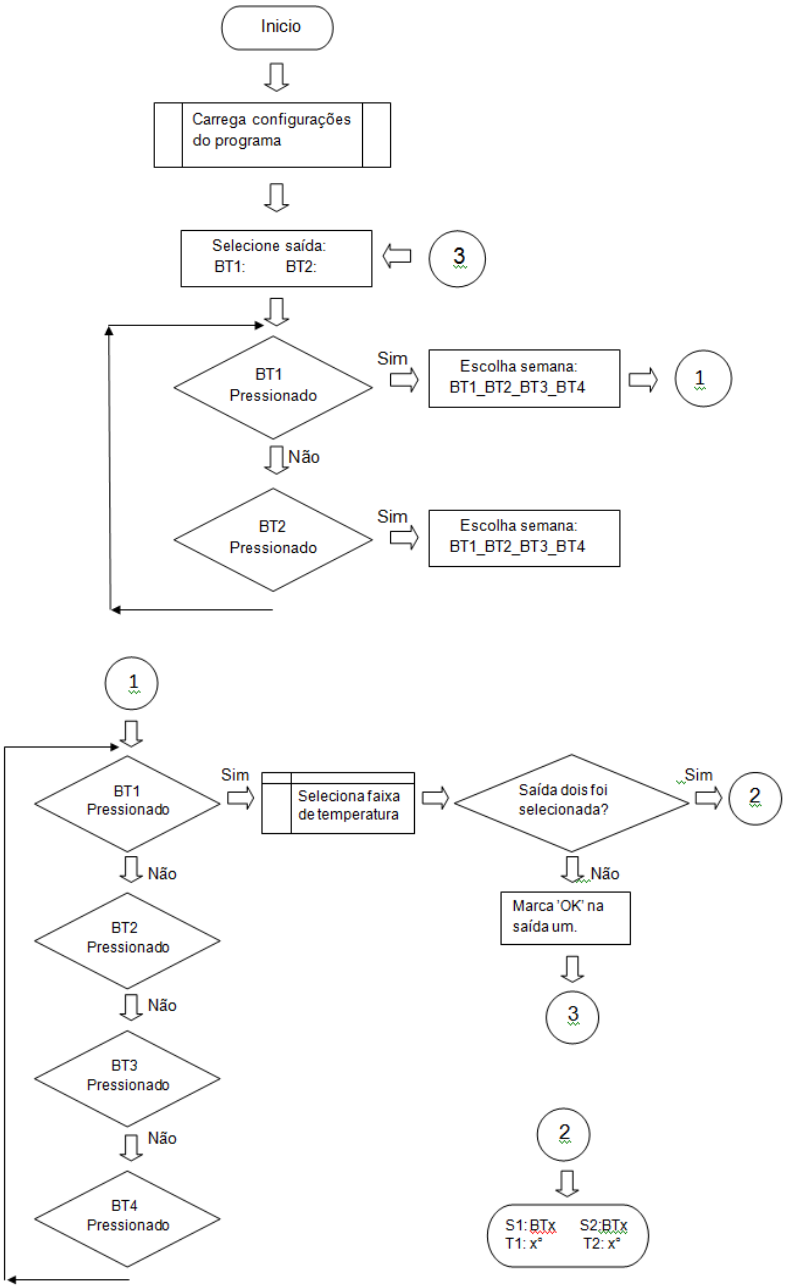

Figura 17. Fluxograma gerado após a visita.
Fonte: Autoria própria

Na primeira tentativa, foi criado o programa de forma que se efetua o controle do disparo do Triac, usando o próprio microcontrolador para mandar os pulsos no tempo de sincronismo com a rede.

O primeiro programa apresentou vários problemas, se destacando entre eles, a lógica de funcionamento. Esse programa foi desenvolvido para seguir uma lógica de processos de acordo com a lógica de controle, contudo quando a rotina principal do mesmo chamava uma função, enquanto esta aguardava a mudança do valor de uma variável, a execução da rotina principal parava e em conseqüência não executava a leitura das portas, finalizando a lógica.

Com o auxilio do software de simulação, foi possível identificar os erros contidos ao longo do programa assim como também compreender o motivo dos mesmos e, por conseguinte, refazer o mesmo de forma funcional e adequada.

Adotou-se a aplicação do componente chamado TCA785, geralmente usado para controle de potência onde, com ele é possível controlar o ângulo de disparo de um Triac com referência a um sinal contínuo no pino 11 do mesmo. Logo, no projeto, o mesmo irá controlar o tempo de disparo de um Triac com referência ao sinal do PWM do microcontrolador.

A figura 18 mostra o circuito montado com parte dos componentes para simulação. A simulação realizada executa o programa criado no software MPLAB após ser compilado e convertido em hexadecimal, através do endereçamento pelo componente virtual microcontrolador.

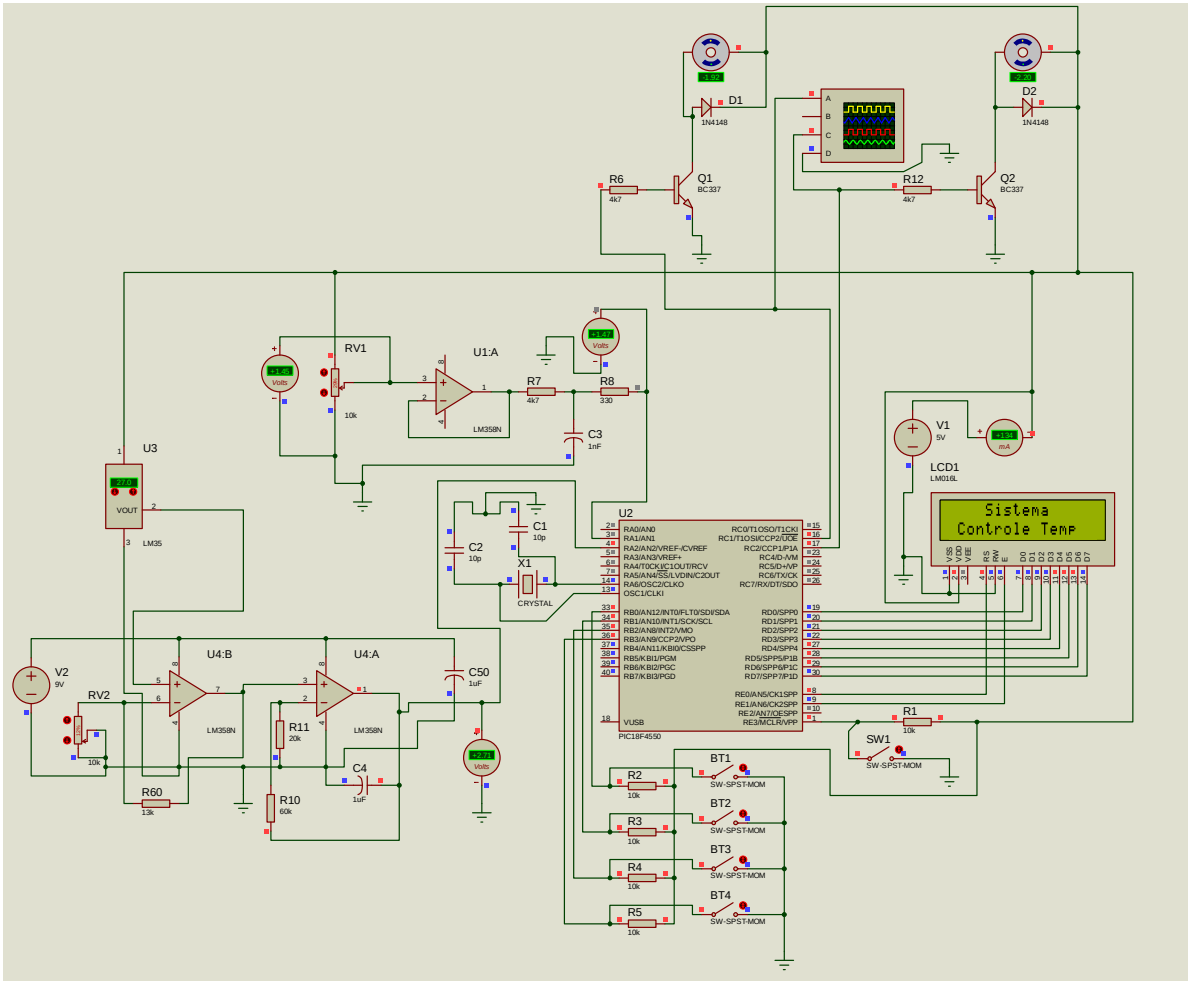

Figura 18. Circuito eletrônico montado para simulação
Fonte: Autoria própria

A figura 19 mostra o componente TCA785 e suas ligações em uma simulação. Nesse caso foi colocado um potenciômetro no pino 11, simulando o sinal do PWM do microcontrolador.

Após simulações, foi realizado alterações no programa do microcontrolador para que o controle seja realizado com as duas saídas de PWM do microcontrolador.

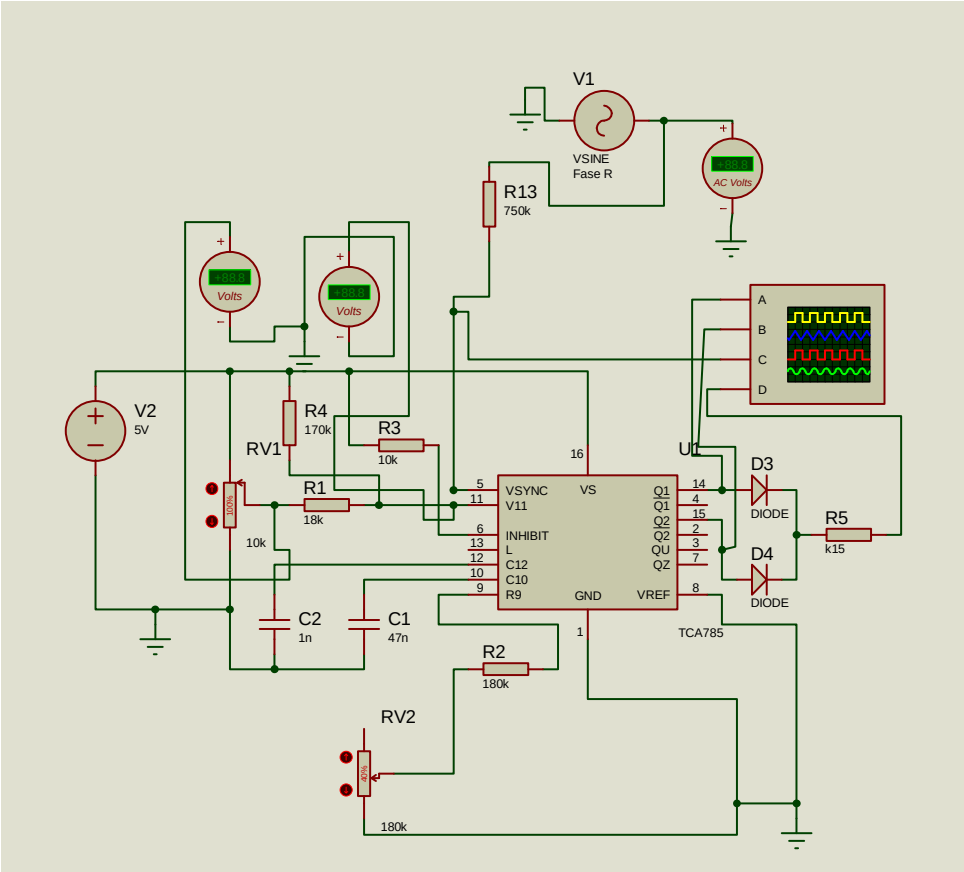

Figura 19. Circuito com TCA785 para simulação
Fonte: Autoria própria

No segundo programa foi encontrado problemas com o tratamento do estouro dos TIMER1 e TIMER2. Esse problema deixou de ocorrer ao colocar a função de tratamento na rotina principal, onde as mesmas eram verificadas a cada varredura da função WHILE. Para realizar o tratamento dos TIMER1 e 2, ativou-se o TIMER0, onde a cada interrupção do mesmo, é feita a verificação do flag dos TIMER1 e TIMER2.

4.3 Simulação no Software

A simulação do programa criado é feita com o arquivo gerado em hexadecimal na compilação do mesmo no software MPLAB. O arquivo em hexadecimal é então selecionado pelo software através do microcontrolador virtual como mostra a figura 20.

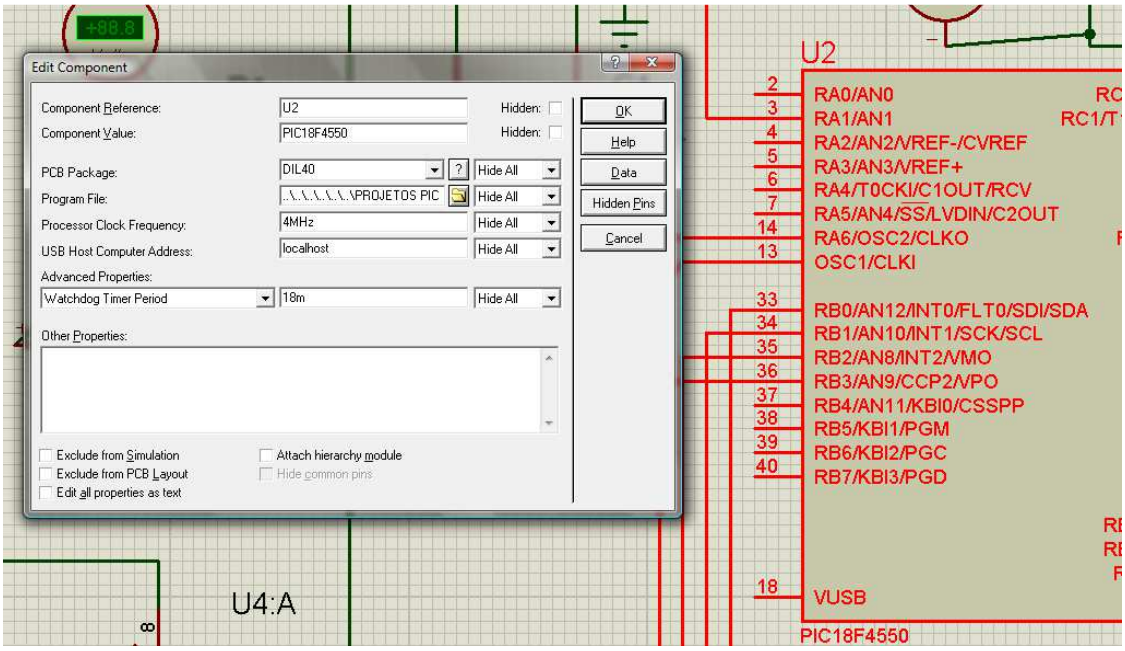

Figura 20. Seleção de arquivo hexadecimal
Fonte: Autoria própria

Selecionando a opção *program file*, abrirá uma janela para procurar o diretório do arquivo, e após selecionar o mesmo, é só confirmar em OK e simular.

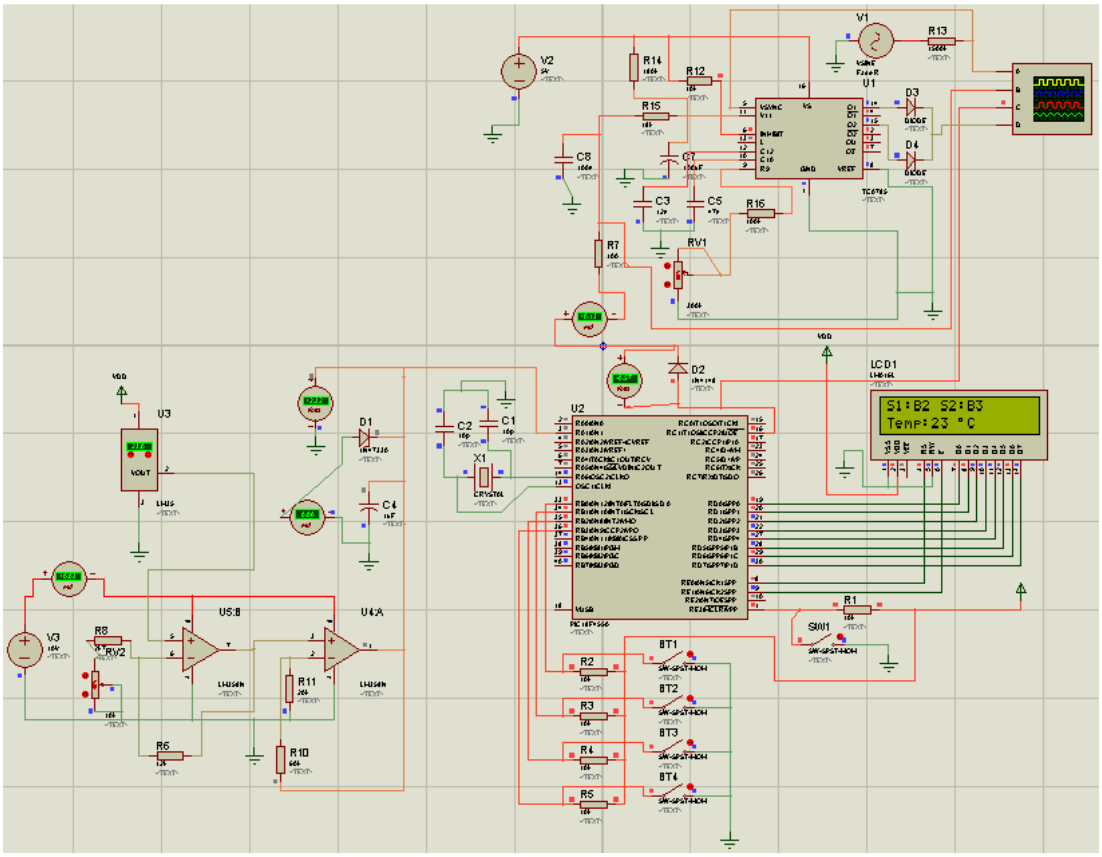

Figura 21. Programa sendo executado na simulação
Fonte: Autoria própria

A figura 21 mostra o programa em simulação juntamente com o circuito amplificador do sensor de temperatura, LCD, botões e TCA785.

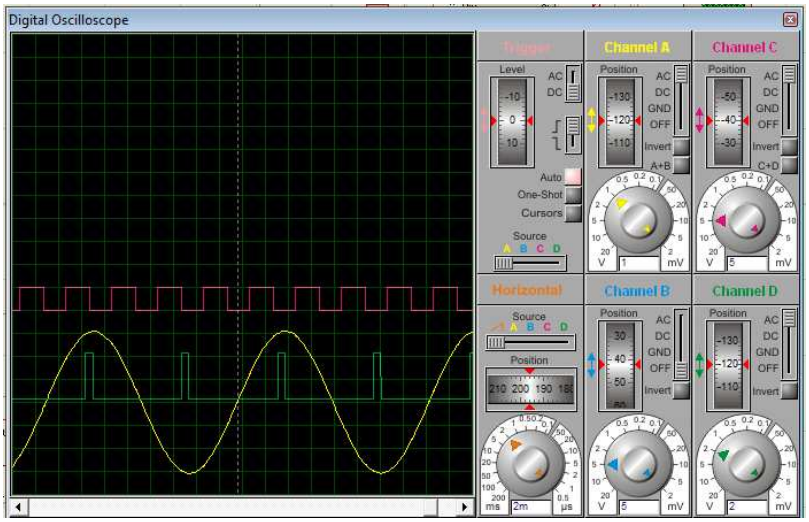

Figura 22. Osciloscópio virtual
Fonte: Autoria própria

A figura 22 mostra a leitura realizada no osciloscópio virtual do software onde, o canal A representa a onda senoidal da rede elétrica, o canal B a leitura do PWM do microcontrolador, e o canal D os pulsos gerados nos pinos 14 e 15 do circuito integrado TCA785.

4.3.1 Simulação na placa de testes MCLAB2

Também foi realizado simulações do programa na prática, usando a placa de testes MCLAB2[6]. A placa de testes desenvolvida pela LABTOOLS[7] possui um microcontrolador PIC18F4550 e um conjunto de periféricos que permite realizar simulações diversificadas.

Em nosso projeto, o programa enviará mensagens ao LCD e através dos botões que a placa possui, pode ser realizado a seleção da faixa de temperatura do programa e, este por sua vez, realizará o controle do PWM. O PWM ligará um pequeno cooler, variando a velocidade de forma proporcional a potência cedida pelo mesmo, para melhor visualização do controle.

Na figura 24 pode ser visualizado a execução do programa na placa MCLAB2. A temperatura indicada no lcd esta referenciado a tensão variavel de um trimpot, onde este simula o sensor de temperatura.

[6] MCLAB2: Placa para simulações.
[7] LABTOOLS: Uma divisão da Mosaico High Performance Solutions.

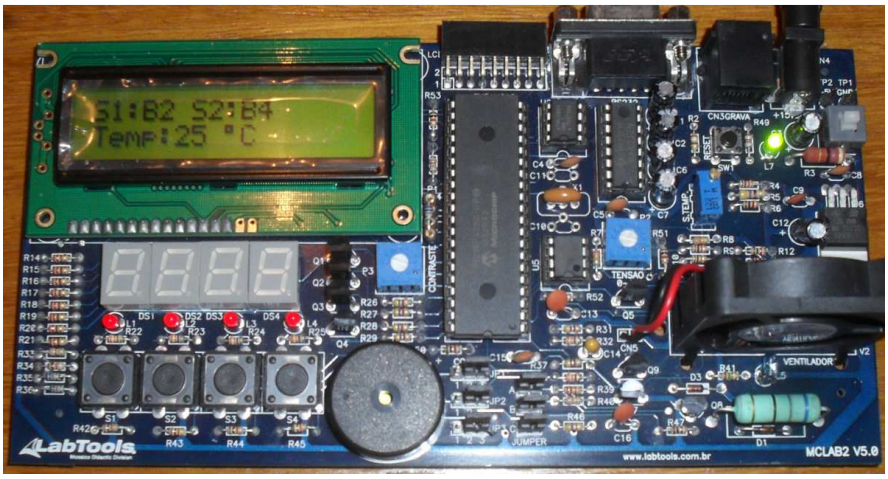

Figura 23. Simulação do programa na placa MCLAB2
Fonte: Autoria própria

4.4 Confecção da placa com circuito impresso principal

Para a confecção da placa de circuito impresso, determinou-se uma borda onde esta compreende as dimensões da placa a ser confeccionada. Após posicionar os componentes na área determinada pelas bordas, organizaram-se as trilhas como mostra a figura 24.

Optou-se por fazer preenchimento entre as trilhas, preenchendo os espaços que no contrário seria corroído.

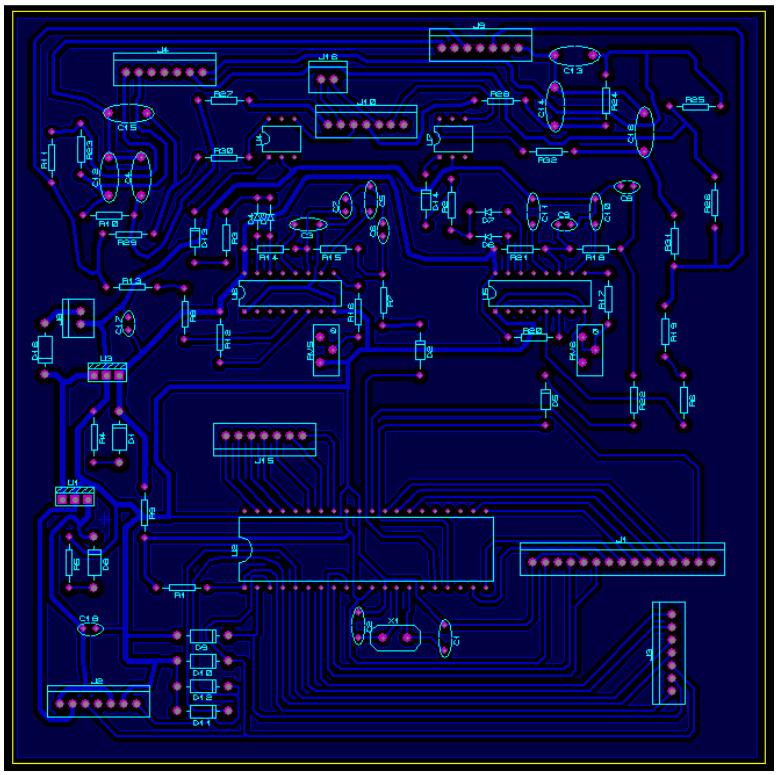

Figura 24. Placa de circuito impresso com microcontrolador
Fonte: Autoria própria

Concluído o desenho da placa, realizou-se a impressão da mesma em pdf como mostra a figura 25.

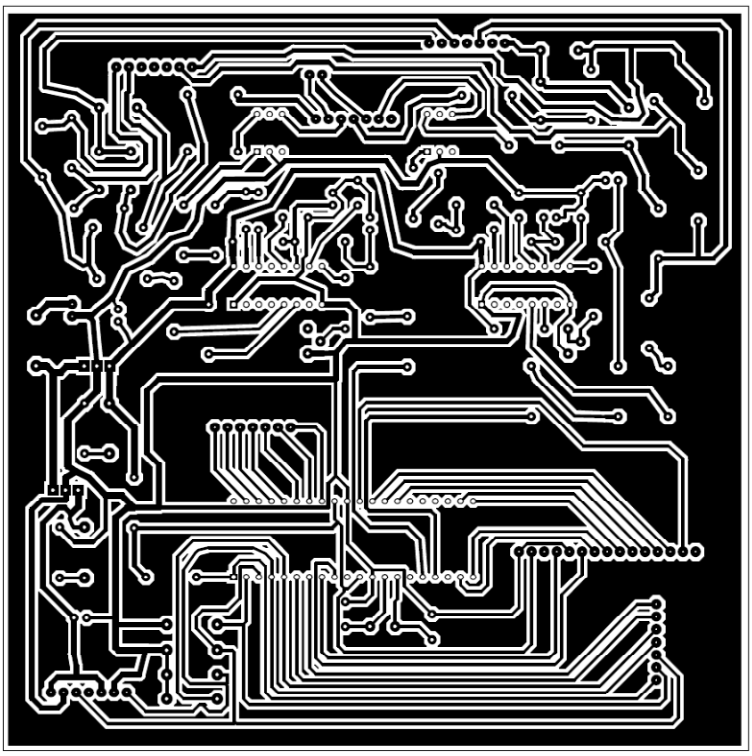

Figura 25. Circuito impresso em PDF
Fonte: Autoria própria

Após impresso em papel foto com tone, inicia-se o processo de transferência do tone para a superfície cobreada do fenolite. Primeiramente realizou-se a limpeza da superfície cobreada com álcool e, em seguida, foi posicionado o desenho sobre a placa e fixado o mesmo com fita crepe para não mexer ao passar o ferro quente. Nesse passo foi usado um ferro de passar roupa, e com o mesmo passou-se sobre o papel foto em toda a área do desenho.

No passo seguinte, após esfriar, mergulhou-se a placa fenolite com o papel foto em água e, depois de passado algum tempo, removeu-se com cuidado o papel foto. Após a remoção do papel, verificou-se a ausência de falhas na transferência do tone e posteriormente colocou-se para corroer mergulhando-a em ácido per cloreto de ferro. Após lavar em água corrente e remover o tone, realizou-se a furação da mesma com broca 0,7mm e 1 mm, como mostra a figura 26.

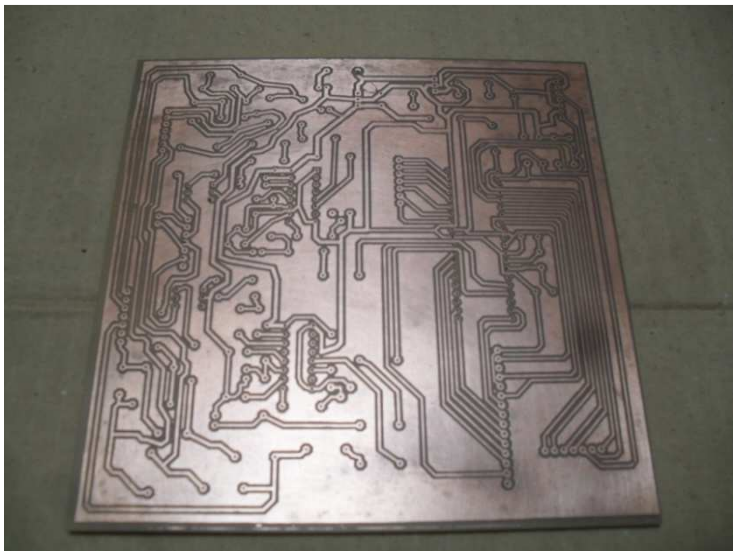

Figura 26. Placa de circuito impresso
Fonte: Autoria própria

Posteriormente, instalaram-se os componentes fixando-os através da solda tipo 60% estanho e 40% chumbo. Para o microcontrolador, TCA785 e o optoacoplador foram usados alojamentos, permitindo fácil remoção e evitando aquecimento do mesmo na soldagem. Abaixo, a figura 27 mostra a placa com os componentes soldados.

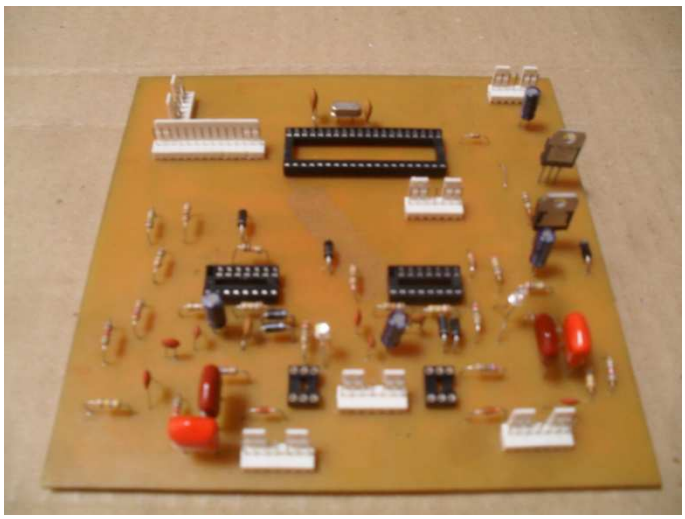

Figura 27. Placa circuito impresso com componentes soldados
Fonte: Autoria própria

4.5 Simulações com TCA785 no protoboard

Após realizado testes na placa mostrado na figura 27, ocorreram problemas no controle de disparos do Triac. O controle de disparo ocorria de forma aleatória, tornando o mesmo integralmente instável.

Usando uma matriz de contato mostrada na figura 28, realizou-se várias simulações com circuitos diferentes alternando o valor dos componentes usados até se obter o controle preciso do disparo do Triac.

Para alimentar do circuito do aparelho, foi usado com uma fonte chaveada de 500ma. Contudo a mesma estava causando interferências no funcionamento do CI TCA785. Como medida de correção para eliminar as interferências, foi instalado capacitores em paralelo com a alimentação da fonte, obtendo uma tensão de alimentação do circuito filtrada e mais estável.

Durante os testes realizados, verificou-se que com a alimentação de 5V para o CI TCA785, a tensão de disparo dos pinos 14 e 15 do mesmo não era suficiente para acionar o CI MOC3021. Posteriormente a pesquisa, optou-se por aplicar o transistor BC548 para amplificar o sinal dos respectivos pinos, adequando as tensões de funcionamento de ambos com resistores.

Ainda durante os testes, foi constatado que o sinal do PWM do microcontrolador não estava sendo suficientemente filtrado, gerando instabilidade no controle de disparo do Triac em função do sinal de referência do CI TCA785. A solução encontrada foi a adição de capacitor de 100uF, sendo a corrente máxima do PWM cerca de trinta por cento da corrente que o PWM do microcontrolador pode suportar.

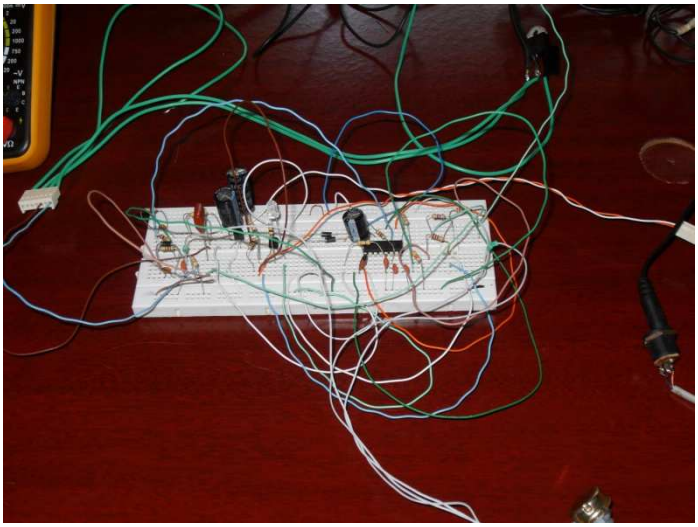

Figura 28. Circuito sendo simulado no protoboard
Fonte: Autoria própria

4.6 Circuitos refeitos e divididos

Em função das alterações do novo circuito, foi necessário a reconstrução do circuito impresso, e ainda, buscando alojar o mesmo em uma caixa de menores dimensões, o mesmo foi dividido ao meio como mostra a figura 29.

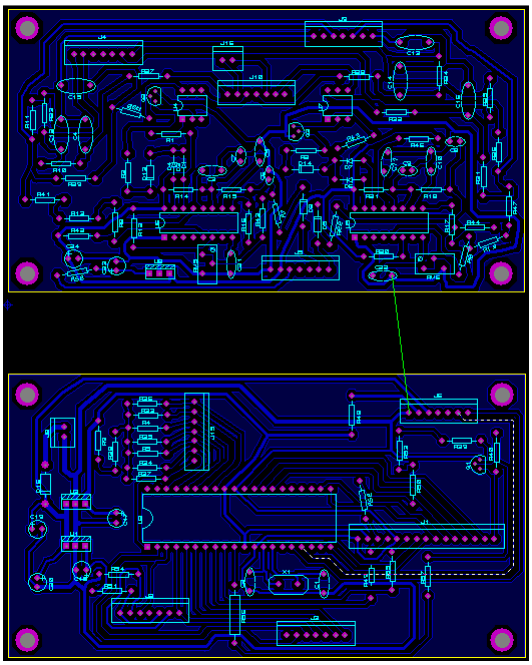

Figura 29. Circuito impresso para confecção da placa
Fonte: Autoria própria

Abaixo, a figura 30 mostra a placa já confeccionada e com os componentes soldados. Embora definido o circuito, foi necessário algumas alterações para melhorar do controle.

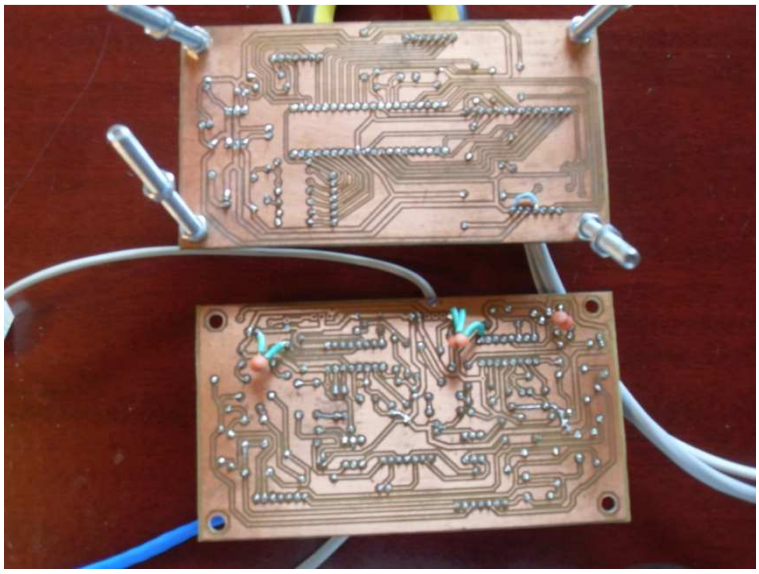

Figura 30. Placas eletrônicas corroídas e com os componentes eletrônicos
Fonte: Autoria própria

As placas eletrônicas foram montadas sobrepostas separadas e fixadas por quatro parafusos como mostra a figura 31, onde estes por sua vez, são fixados no fundo da caixa.

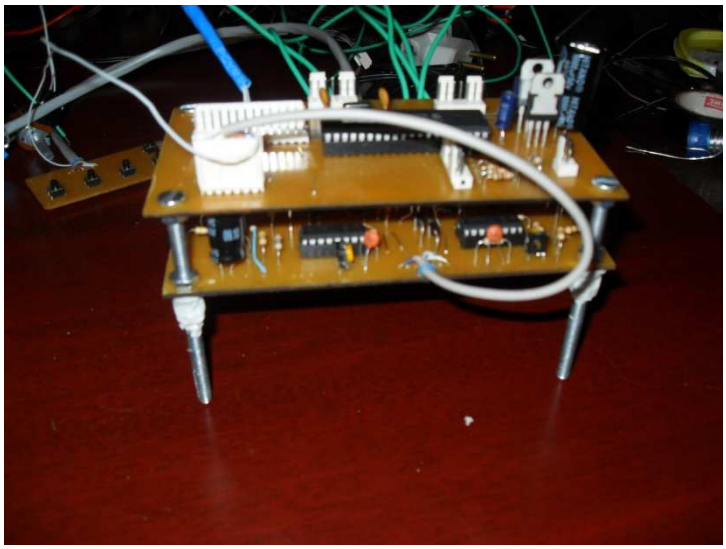

Figura 31. Conjunto de placas eletrônicas prontas para fixação na caixa
Fonte: Autoria própria.

4.7 Testes finais

Terminando a montagem do aparelho, realizou-se um último teste verificando o seu funcionamento.

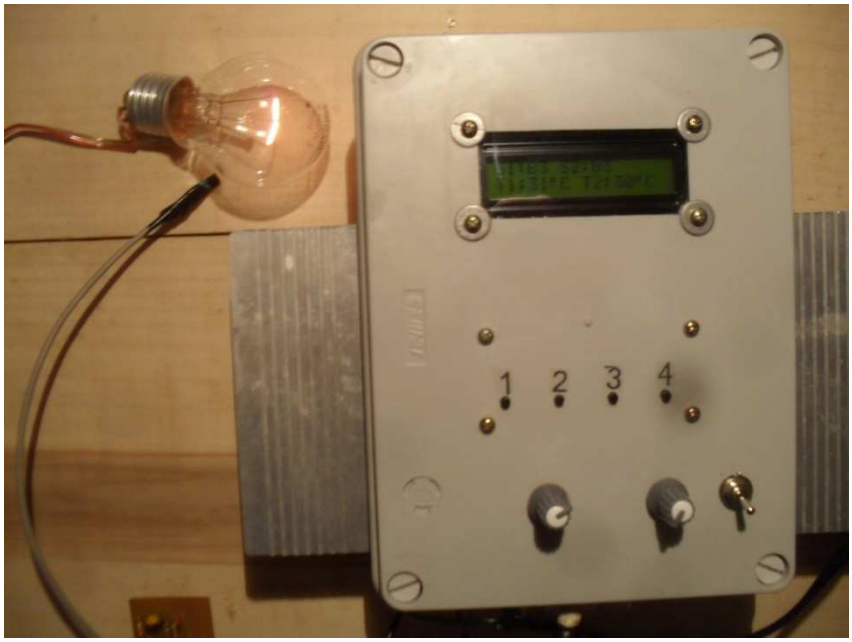

Figura 32. Aparelho montado sendo testado
Fonte: Autoria própria

A figura 32 mostra o aparelho em funcionamento controlando a potência cedida á carga, que nesse caso é uma lâmpada de 100W. Com o sensor próximo da lâmpada, a tendência é de o aparelho controlar a potência cedida à lâmpada até que a mesma aqueça o sensor com uma temperatura que fique dentro da faixa de temperatura do programa do aparelho.

Durante os testes também foi realizado o controle da temperatura com uma carga resistiva de 1000W, obtendo como resposta um controle estável e preciso.

5 Conclusão

A solução apresentada nesse trabalho propõe um controle automático de potência cedida á carga para a adequação da temperatura dos escamoteadores em maternidade de suínos.

Durante o desenvolvimento do projeto, surgiram varias dificuldades relacionadas com a lógica de programação e ao funcionamento dos componentes eletrônicos, em especial o TCA785. Depois de seguidas simulações e pesquisas, obteve-se um circuito funcional, realizando o controle estável da potência.

Concluído a construção do aparelho, foi realizado um teste com uma carga resistiva de 1000W, alimentado com tensão de 127V gerando aproximadamente uma corrente de 7,8 Ampères. Durante o teste, observou-se que o controle da potência cedida à carga ocorria de forma estável de acordo com o programa do microcontrolador.

Para o funcionamento do aparelho, é necessário a instalação de proteções elétricas (disjuntores ou fusíveis), pois o mesmo poderá será danificado em curtos ou sobrecargas que ultrapassem a corrente de 35 Ampères.

Para trabalhos futuros, pode-se implementar o programa de forma que o usuário altere as faixas de temperatura sendo estas gravadas na memória não-volátil para não serem perdidos no desligamento do aparelho.

6 Referências

1. SOSSUINOS. **Equipamentos para suinocultura**. Disponível em: http://www.sossuinos.com.br/equipamentos.htm Acesso em 19/10/2012.
2. NÄÄS, Irenilza de Alencar e CALDARA, Fabiana Ribeiro, **Ambiência para suínos no inverno.** Disponível em: http://www.porkworld.com.br/artigos/post/ambiencia-para-suinos-no-inverno Acesso em 19/10/2012.
3. ANIMALWORLD. **Queda na temperatura corporal**. Disponível em: http://editora-animalworld.com.br/blog/wp-content/uploads/2011/07/12.png Acesso em 19/10/2012.
4. GSIBRASIL. **Controlador de ambientes INTEGRA-PRO CA-05**. Disponível em: http://www.gsibrasil.ind.br/pdf_prods/catalogo_tp001_lg01_Controlador_de_Ambientes_Integra-Pro_CA-05.pdf Acesso em 19/10/2012.
5. Presstemp. **Controlador DTB4848.** Disponível em: http://www.presstemp.com.br/index.asp?idproduto=257673 Acesso em 19/10/2012.
6. Novus. **Controlador de temperatura N1020.** Disponível em: http://www.novus.com.br/site/default.asp Acesso em 19/10/2012
7. ANIMALWORLD. **Temperatura de conforto**. Disponível em: http://editora-animalworld.com.br/blog/wp-content/uploads/2011/07/Sem-t%C3%ADtulo12.png Acesso em 19/10/2012.
8. VITAGLIANO Luiz Antonio, BARBOSA Fellipe Freitas e PEREIRA Francisco Alves, **Estresse térmico pelo frio sobre o desempenho de suínos.** Disponível em: http://www.porkworld.com.br/artigos/post/estresse-termico-pelo-frio-sobre-o-desempenho-de-suinos Acesso em 19/10/2012
9. MORELLATO, Fernando César. **Microcontroladores para TV.** Disponível em: http://dc167.4shared.com/doc/vhwoWK81/preview.html Acesso em 17/10/2012.
10. CARY, David. **Microchip Technology.** Disponível em: http://en.inforapid.org/index.php?search=General%20Instrument%20CP1600

Acesso em 18/10/2012.

11. DERNANDIN,Gustavo Weber. **Microcontroladores.** Disponível em: http://pessoal.utfpr.edu.br/gustavo/apostila_micro.pdf Acesso em 17/10/2012.
12. SERÓDIO Anderson Augusto. **Noções sobre microcontroladores.** Disponível em: http://projetosembarcados.blogspot.com.br/2011/01/nocoes-sobre-microcontroladores.html Acesso em 19/10/2012.
13. LABTOOLS. **Curso Módulo 4 – Família High End PIC 18 e MPLAB C18**. Disponivel em: http://www.mosaico.com.br/index.asp?canal=5&pg=showProduto&id=37&path=Produtos Acesso em 19/10/2012
14. Guzella Luis. **Circuitos retificadores de onda.** Disponível em: http://amigonerd.net/trabalho/42417-circuitos-retificadores-de-onda Acesso em 19/10/2012.
15. GUIMA, Lucas. **Explicacao sobre TCA-785.** Disponível em: http://pt.scribd.com/doc/85484181/Explicacao-sobre-TCA-785 Acesso em 19/10/2012.
16. MULTILOGICA. **Sensor de temperatura LM35DZ.** Disponível em: http://multilogica-shop.com/sensor-de-temperatura-lm35dz Acesso em 19/10/2012.
17. NICOLETI Thales. **Projetos eletrônicos.** Disponível em: http://thalesnicoleti.blogspot.com.br/2011_01_01_archive.html Acesso em 19/10/2012.
18. CENTELHAS. **Amplificadores.** Disponível em: http://www.centelhas.com.br/biblioteca/amplificadores.pdf Acesso em 19/10/2012.
19. FERREIRA Franlim F. **Amplificadores operacionais.** Disponível em: http://paginas.fe.up.pt/~fff/Homepage/Ficheiros/E1_Cap2.pdf Acesso em 19/10/2012.
20. STMICROELETRONICS**. Datasheet.** Disponível em: http://www.datasheetcatalog.org/datasheet/stmicroelectronics/7469.pdf Acesso em 19/10/2012.
21. EBAYIMG. **BTA41-600B.** Disponível em: http://i.ebayimg.com/t/2pcs-BTA41-600B-BTA41-600-BTA41-600-Triac-600V-40A-ST-

/24/!Byt2(rgEGk~$(KGrHqZ,!joEw5N3DToGBMS-IimKzQ~~_35.JPG Acesso em: 20/10/2012

22. PEREIRA Filipe e CORDEIRO Francisco. **Relatório de projeto final.** Disponível em: http://ltodi.est.ips.pt/aabreu/WattimetroDigitalRelatorio.pdf Acesso em 19/10/2012.
23. FUTURLEC**. Moc3021.** Disponível em: http://www.futurlec.com/Pictures/MOC3021.jpg Acesso em 19/10/2012
24. FellipeMauricio. **LCD 16x2**. Disponível em: http://fellipemauricio.com.br/wp-content/uploads/2012/05/lcd.jpg Acesso em 19/10/2012.
25. CENTRAL AVR**. Display lcd** Disponível em: http://centralavr.blogspot.com.br/2011/04/utilizando-um-display-de-lcd-com-o.html Acesso em 19/10/2012.
26. MUNIZ Laurentino Borges. **Sistema de controle digital da temperatura da água do chuveiro elétrico.** Disponível em: http://www.readbag.com/aldeia3putacao-greenstone-collect-trabalho-index-assoc-hash01cd-dir-doc Acesso em 19/10/2012
27. LM7805. **Vantagens da faminia 78XX**. Disponível em: http://lm7805.net/ Acesso em 19/10/2012.
28. ELETRONICA. **Regulador LM.** Disponível em: http://www2.eletronica.org/hack-s-dicas/regulador-lm7805/conexao_tipica_lm7805 Acesso em 19/10/2012.
29. TONIELETRÔNICA. **Mplab ide 8.5.** Disponível em: http://www.te1.com.br/wp-content/uploads/2010/05/mplab-ide-8.5.jpg Acesso em 19/10/2012.
30. SOUSA David José de. **Desbravando o PIC: ampliado e atualizado para PIC 16F628A** / -- 12.ed. -- São Paulo:Érica, 2008.

Anexo I

```
/*****************************************************
 *      Programação em C18 - TCC usando o PIC18F4550 e McLab2          *
 *                    primeiro programa                                *
 *                                                                     *
 *          UTFPR- Tecnologia em Manutenção Industrial                 *
 *                                                                     *
 * TEL: (045) 9948-4535        EMAIL: alex.tcharleysander@hotmail.com  *
 *****************************************************
 *  VERSÃO : 1.0                                       *
 *  DATA : 27/10/2011                                  *
 ***************************************/

/************************************
 *                  Descrição geral                  *
 ************************************/
/*
          ESTE SOFTWARE ESTÁ PREPARADO PARA LER QUATRO BOTÕES E CONFIGURAR A SAIDA
DE ACORDO COM A FORMULA ASSOCIADO A CADA BOTÃO, ONDE ESTE TERA DEFINIDO UMA FAIXA
DE TEMPERATURA PARA A EXECUÇÃO DO CONTROLE, ALEM DE MOSTRAR NO LCD A CONFIGURAÇÃO
ESCOLHIDA, A SAIDA SELECIONADA E MOSTRAR A TEMPERATURA ATUAL.
*/
/************************************
 *                          DEFINIÇÃO PIC          *
 ************************************/

#include <p18F4550.h>          // Register definitions

/************************************
 *          INCLUDES DAS FUNÇÕES DE PERIFÉRICOS DO PIC          *
 ************************************/
#include <pwm.h>                        //PWM library functions
#include <adc.h>                        //ADC library functions
#include <timers.h>                     //Timer library functions
#include <delays.h>                     //Delay library functions
#include <i2c.h>                        //I2C library functions
#include <stdlib.h>                     //Library functions
#include <usart.h>                      //USART library functions
#include <stdio.h>
/************************************
 *                Configurações para gravação              *
 ************************************/

#pragma config FOSC = HS
#pragma config CPUDIV = OSC1_PLL2
#pragma config WDT = ON
#pragma config WDTPS = 128
#pragma config LVP = OFF
#pragma config PWRT = ON
#pragma config BOR = ON
#pragma config BORV = 0


/************************************
 *          Definição e inicialização das variáveis Globais        *
 ************************************/
//Neste bloco estão definidas as variáveis globais do programa.
//Este programa não utiliza nenhuma variável de usuário

unsigned short      FILTRO1;            //FILTRO BOTÃO 1
unsigned short      FILTRO2;            //FILTRO BOTÃO 2
unsigned short      FILTRO3;            //FILTRO BOTÃO 3
unsigned short      FILTRO4;            //FILTRO BOTÃO 4
unsigned char contador = 2;             //Contador para somar igual um segundo
unsigned char BT_um = 0;
unsigned char BT_dois = 0;
unsigned char BT_treis = 0;
unsigned char BT_quatro = 0;
unsigned char mensagem = 1;
unsigned char frase_triac = 0;
unsigned char triac_um = 0;
unsigned char triac_dois = 0;
```

```
unsigned char um_saida = 0;
unsigned char dois_saida = 0;
unsigned char temp_one = 0;
unsigned char temp_two = 0;
unsigned char tela_um = 0;
unsigned char tela_dois = 0;
unsigned char PL_frasetr = 0;
unsigned char PL_trum = 0;
unsigned char PL_trdois = 0;
unsigned char pronto_um = 0;
unsigned char pronto_dois = 0;
unsigned int conversao = 0;
unsigned int convdois = 0;
unsigned int TCone = 180;
unsigned int TCtwo = 180;
unsigned char unidade = 0;
unsigned char dezena = 0;
unsigned char centena = 0;
unsigned char tela_valor = 0;
unsigned int resultado = 0;
unsigned int resultdois = 0;
unsigned int termum = 0;
unsigned int termdois = 0;
unsigned char desconta = 2;

/************************************
 *                Constantes internas                *
 ************************************/
//A definição de constantes facilita a programação e a manutenção.

/************************************
 *            Declaração dos flags de software            *
 ************************************/
//A definição de flags ajuda na programação e economiza memória RAM.
//Este programa não utiliza nenhum flag de usuário

struct
{
        unsigned BIT0:1;
   unsigned BIT1:1;
   unsigned BIT2:1;
   unsigned BIT3:1;
   unsigned BIT4:1;
   unsigned BIT5:1;
   unsigned BIT6:1;
   unsigned BIT7:1;
}FLAGSbits;                                //ARMAZENA OS FLAGS DE CONTROLE

#define   ST_BT1          FLAGSbits.BIT0          //STATUS DO BOTÃO 1
#define   ST_BT2          FLAGSbits.BIT1          //STATUS DO BOTÃO 2
#define   ST_BT3          FLAGSbits.BIT2          //STATUS DO BOTÃO 3
#define   ST_BT4          FLAGSbits.BIT3          //STATUS DO BOTÃO 4
#define delta_timer1      (65536 - 62500)
#define AN0_AN1
/************************************
 *              PROTOTIPAGEM DE FUNÇÕES              *
 ************************************/

void comando_lcd(unsigned char caracter);
void escreve_lcd(unsigned char caracter);
void limpa_lcd(void);
void inicializa_lcd(void);
void escreve_frase(const rom char *frase);
void tela_principal(void);
void TRATA_HIGH_INT(void);
void TRATA_INT_TIMER1(void);
void TRATA_botoes(void);
void decrementa_timer(void);
void frase_partida(void);
void selecao(void);
void confirma_um(void);
void confirma_dois(void);
void reinicia_bt(void);
void servo_um(void);
void servo_dois(void);
void contra(void);
```

```
void conch(void);
void converte_bcd(unsigned char aux);

/************************************
 *                ENTRADAS          *
 ************************************/
// As entradas devem ser associadas a nomes para facilitar a programação e
//futuras alterações do hardware.

#define BT1 PORTBbits.RB0       //BOTÃO 1
                                                        //0 -> PRESSIONADO
                                                        //1 -> LIBERADO

#define BT2 PORTBbits.RB1       //BOTÃO 2
                                                        //0 -> PRESSIONADO
                                                        //1 -> LIBERADO

#define BT3 PORTBbits.RB2       //BOTÃO 3
                                                        //0 -> PRESSIONADO
                                                        //1 -> LIBERADO

#define BT4 PORTBbits.RB3       //BOTÃO 4
                                                        //0 -> PRESSIONADO
                                                        //1 -> LIBERADO

/************************************
 *                SAÍDAS            *
 ************************************/
// AS SAÍDAS DEVEM SER ASSOCIADAS A NOMES PARA FACILITAR A PROGRAMAÇÃO E
//FUTURAS ALTERAÇÕES DO HARDWARE.

   #define rs                   PORTEbits.RE0     // via do lcd que sinaliza recepção de dados ou comando
   #define enable    PORTEbits.RE1     // enable do lcd


/************************************
 *      Rotina que envia um COMANDO para o LCD      *
 ************************************/

void comando_lcd(unsigned char caracter)
{
        rs = 0;                                         // seleciona o envio de um comando
        PORTD = caracter;                               // carrega o PORTD com o caracter
        enable = 1 ;                                    // gera pulso no enable
        Delay10TCYx(1);                                 // espera 10 microsegundos
        enable = 0;                                     // desce o pino de enable
        Delay10TCYx(4);                                 // espera mínimo 40 microsegundos
}

/************************************
 *      Rotina que envia um DADO a ser escrito no LCD       *
 ************************************/

void escreve_lcd(unsigned char caracter)
{
        rs = 1;                                         // seleciona o envio de um comando
        PORTD = caracter;                               // carrega o PORTD com o caracter
        enable = 1;                                     // gera pulso no enable
        Delay10TCYx(1);                                 // espera 10 microsegundos
        enable = 0;                                     // desce o pino de enable
        Delay10TCYx(4);                                 // espera mínimo 40 microsegundos
}

/************************************
 *          Função para limpar o LCD            *
 ************************************/

void limpa_lcd(void)
{
        comando_lcd(0x01);                              // limpa lcd
        Delay1KTCYx(2);
}

/************************************
 *      Inicialização do Display de LCD         *
 ************************************/
```

```
void inicializa_lcd(void)
{
	comando_lcd(0x30);				// envia comando para inicializar display
	Delay1KTCYx(4);					// espera 4 milisengundos

	comando_lcd(0x30);				// envia comando para inicializar display
	Delay10TCYx(10);				// espera 100 microsengundos

	comando_lcd(0x30);				// envia comando para inicializar display

	comando_lcd(0x38);				// liga o display, sem cursor e sem blink

	limpa_lcd();					// limpa lcd

	comando_lcd(0x0c);				// display sem cursor

	comando_lcd(0x06);				// desloca cursor para a direita
}

/*************************************
 *	Função para escrever uma frase no LCD	*
 *************************************/

void escreve_frase(const rom char *frase)
{
	do
	{
		escreve_lcd(*frase);
	}while(*++frase);
}

/*********************************
 *	Funcao de decremento do Timer		*
 *********************************/
void decrementa_timer(void)
{
	desconta --;					// DECREMENTA UNIDADE
	if (desconta == 0)				// desconta = 2?
	{
		desconta = 2;				// UNIDADE = 0
		servo_um();					// atualiza saida
		servo_dois();				// atualiza saida
		Delay10TCYx(1);				// espera 10 microsegundos
		SetDCPWM1(TCone);			// ATUALIZA PWM1
		SetDCPWM2(TCtwo);			// ATUALIZA PWM2
	}
}

/*********************************
 *		Servo um			*
 *********************************/

void servo_um(void)
{
switch(temp_one)					// atualizacao e formulas de conversao
		{
		case 1:
		if (TCone < 610)			// permite incrementar maximo 85 = 20% saida
			{
if (termum > 34)					// +tempo - potencia saida
	{	TCone ++;	}				// -tempo + potencia saida

if (termum > 35)
{	TCone ++;	}	//aumenta mais e intervalo pequeno de tempo entre ajuste

			}
if (TCone > 5)	// permite minimo 1
	{
	if (termum < 32)
	{	TCone --; }

	if (termum < 30)
	{	TCone --; }		//aumenta mais e intervalo pequeno de tempo entre ajuste

				if (termum < 28)
```

```
                                        {       TCone --; }                     //aumenta mais e intervalo pequeno
de tempo entre ajuste

                        }       break;

                                case 2:
                if (TCone < 610)                        //permite incrementar maximo 85 = 20% saida
                        {
                                if (termum > 31)
                                        {       TCone ++;       }       //aumenta mais e intervalo pequeno
de tempo entre ajuste

                                if (termum > 32)
                                        {       TCone ++;       }

                        }
                if (TCone > 5)  // permite minimo 1
                        {
                                if (termum < 29)
                                        {       TCone --; }

                                if (termum < 27)
                                        {       TCone --; }     //aumenta mais e intervalo pequeno de tempo
entre ajuste

                                if (termum < 25)
                                        {       TCone --; }     //aumenta mais e intervalo pequeno de tempo
entre ajuste

                        }       break;

                                case 3:
                if (TCone < 610)                        //permite decrementar
                        {
                                if (termum > 28)
                                        {       TCone ++;       }

                                if (termum > 29)
                                        {       TCone ++;       }
                        }

                if (TCone > 5)  // permite minimo 1
                        {
                                if (termum < 26)
                                        {       TCone --; }

                                if (termum < 24)
                                        {       TCone --; }     //aumenta mais e intervalo pequeno de tempo
entre ajuste

                                if (termum < 22)
                                        {       TCone --; }

                        }       break;

                                case 4:
                if (TCone < 610)                                        //permite
                        {
                                if (termum > 24)
                                        {       TCone ++;       }

                                if (termum > 25)
                                        {       TCone ++;       }

                        }
                if (TCone > 5)                                          // permite minimo 1
                        {
                                if (termum < 18)
                                        {       TCone --; }

                                if (termum < 17)
                                        {       TCone --; }                     //aumenta mais e intervalo pequeno
de tempo entre ajuste

                                if (termum < 16)
                                        {       TCone --; }
```

```
			}		break;
		}										//		finaliza
comando sw
	}											//finaliza um


/************************************
 *					Servo dois				*
 ************************************/

 void servo_dois(void)
{
	switch(temp_two)
	{
		case 1:
		if (TCtwo < 610)			//permite incrementar maximo 85 = 20% saida
			{
				if (termdois > 34)				// +tempo - potencia saida
					{		TCtwo ++;		}					//	-
tempo + potencia saida

				if (termdois > 35)
					{		TCtwo ++;		}		//aumenta mais e intervalo pequeno
de tempo entre ajuste

			}
		if (TCtwo > 5)		// permite minimo 1
			{
				if (termdois < 32)
					{		TCtwo --; }

				if (termdois < 30)
					{		TCtwo --; }			//aumenta mais e intervalo pequeno
de tempo entre ajuste

				if (termdois < 28)
					{		TCtwo --; }			//aumenta mais e intervalo pequeno
de tempo entre ajuste

			}		break;
case 2:
		if (TCtwo < 610)			//permite incrementar maximo 85 = 20% saida
			{
				if (termdois > 31)
					{		TCtwo ++;		}		//aumenta mais e intervalo pequeno
de tempo entre ajuste

				if (termdois > 32)
					{		TCtwo ++;		}

			}
		if (TCtwo > 5)		// permite minimo 1
			{
				if (termdois < 29)
					{		TCtwo --; }

				if (termdois < 27)
					{		TCtwo --; }		//aumenta mais e intervalo pequeno de tempo
entre ajuste

				if (termdois < 25)
					{		TCtwo --; }		//aumenta mais e intervalo pequeno de tempo
entre ajuste

			}		break;

				case 3:
		if (TCtwo < 610)			//permite decrementar
			{
				if (termdois > 28)
					{		TCtwo ++;		}

				if (termdois > 29)
					{		TCtwo ++;		}
			}
```

```
			if (TCtwo > 5)		// permite minimo 1
				{
					if (termdois < 26)
						{		TCtwo --; }

					if (termdois < 24)
						{		TCtwo --; }		//aumenta mais e intervalo pequeno de tempo
entre ajuste

					if (termdois < 22)
						{		TCtwo --; }

				}		break;

					case 4:
			if (TCtwo < 610)							//permite
				{
					if (termdois > 24)
						{		TCtwo ++;		}

					if (termdois > 25)
						{		TCtwo ++;		}

				}
			if (TCtwo > 5)								// permite minimo 1
				{
					if (termdois < 18)
						{		TCtwo --; }

					if (termdois < 17)
						{		TCtwo --; }				//aumenta mais e intervalo pequeno
de tempo entre ajuste

					if (termdois < 16)
						{		TCtwo --; }

				}		break;
		}											// finaliza comando sw
}													//finaliza dois

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *					Tela Principal						*
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void tela_principal(void)
{
		limpa_lcd();					// limpa lcd
		comando_lcd(0x80);				// posiciona o cursor na linha 0, coluna 0
										// imprime mensagem no lcd
		escreve_frase("S1:");

		comando_lcd(0x86);				// posiciona o cursor na linha 0, coluna 6
										// imprime mensagem no lcd
		escreve_frase("S2:");

		comando_lcd(0xC0);				// posiciona o cursor na linha 0, coluna 3
		escreve_frase("T1:");

		comando_lcd(0xC8);				// posiciona o cursor na linha 0, coluna 3
		escreve_frase("T2:");
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *						informa								*
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

		void informa(void)
		{
			tela_principal();
			tela_um   = temp_one;
			tela_dois = temp_two;
			mensagem = 1;
			tela_valor = 1;
			switch(tela_um)
				{
```

```
                        case 1:
                                comando_lcd(0x83);          // posiciona o cursor na linha 1,
coluna 1
                                escreve_frase("B1");        // imprime mensagem no lcd
                        break;

                        case 2:
                                comando_lcd(0x83);          // posiciona o cursor na linha 1,
coluna 6
                                escreve_frase("B2");        // imprime mensagem no lcd
                        break;

                        case 3:
                                comando_lcd(0x83);          // posiciona o cursor na linha 1,
coluna 6
                                escreve_frase("B3");        // imprime mensagem no lcd
                        break;

                        case 4:
                                comando_lcd(0x83);          // posiciona o cursor na linha 1,
coluna 6
                                escreve_frase("B4");        // imprime mensagem no lcd
                        break;
                        }

                switch(tela_dois)
                        {
                        case 1:
                                comando_lcd(0x89);          // posiciona o cursor na linha 1,
coluna 1
                                escreve_frase("B1");        // imprime mensagem no lcd
                        break;

                        case 2:
                                comando_lcd(0x89);          // posiciona o cursor na linha 1,
coluna 6
                                escreve_frase("B2");        // imprime mensagem no lcd
                        break;

                        case 3:
                                comando_lcd(0x89);          // posiciona o cursor na linha 1,
coluna 6
                                escreve_frase("B3");        // imprime mensagem no lcd
                        break;

                        case 4:
                                comando_lcd(0x89);          // posiciona o cursor na linha 1,
coluna 6
                                escreve_frase("B4");        // imprime mensagem no lcd
                        break;
                        }
        }

/*************************************
 *                  Funcao                                                  *
 *************************************/

void funcao(void)                               // mensagem que mostra a funcao do aparelho
{
        comando_lcd(0x84);                              // posiciona o cursor na linha 0, coluna 1

        escreve_frase("Sistema");       // imprime mensagem no lcd
        comando_lcd(0xC1);                              // posiciona o cursor na linha 1, coluna 2
        escreve_frase("Controle Temp");  // imprime mensagem no lcd
}

/*************************************
 *                  confirma um                                             *
 *************************************/
void confirma_um(void)                  //confirma a selecao saida triac um
{
                reinicia_bt();                          // coloca botoes nivel logico zerro
                Delay10TCYx(1);                         // espera 10 microsegundos
                if(!dois_saida)                         //se tempo dois for igual a zero
                {
                pronto_um = 1;
```

```
                frase_partida();                // mesagem de escolha de botoes
                }
                        else
                        {
                                informa();              // imprime mensagem de bt selecionado + tela principal
                        }
}

/************************************
 *                                  confirma dois                                                   *
 ************************************/
void confirma_dois(void)
{
                reinicia_bt();                          // coloca botoes nivel logico zerro
                Delay10TCYx(1);                         // espera 10 microsegundos
                if(!um_saida)                           //se tempo dois for igual a zero
                {
                        pronto_dois = 1;
                        frase_partida();        // mensagem selecao de saida
                }
                        else
                        {
                                informa();              //mostra a mensagem principal
                        }
}


/************************************
 *                                  reinicia bt                 *
 ************************************/
void reinicia_bt(void)
{
        BT_um = 0;
        BT_dois = 0;
        BT_treis = 0;
        BT_quatro = 0;
}

/************************************
 *                                  selecao                                                 *
 ************************************/
void selecao(void)
{
                                                        limpa_lcd();
                                                        comando_lcd(0x80);
        // posiciona o cursor na linha 0, coluna 0
                                                        escreve_frase("Escolha Semana");        // imprime mensagem no
lcd
                                                        comando_lcd(0xC0);
                                                        escreve_frase("B1_B2_B3_B4");
                                                        frase_triac = 0;                                        // fecha
acesso a frase triac
}

/************************************
 *                                  TRATA BOTOES                *
 ************************************/

void TRATA_botoes(void)
{
        if(mensagem)                                                                    //vetor de partida
                {
                        tela_valor = 0;
                        frase_partida();                                        // mensagem selecao de saida mais...
                }                                                                                       //... abilitacao de
frase triac
                        if(frase_triac)                                                 //seleciona saida a ser
tratada
                        {
                                        if(!BT_treis)                                   // se bt treis nao entra e
pula para reiniciabt
                                        {
                                                if(!BT_quatro)                  // se bt quatro nao entra e
pula para reiniciarbt
                                                        {
```

```
                                                                selecao();              //escreve          mensagem
selecao de semana e nivel logico baixo frase treis
                                                                if(BT_um)
                                                                {
                                                                        PL_trum = 1;                    //seta
pula ciclo triac um
                                                                }
                                                                if(BT_dois)
                                                                {
                                                                        PL_trdois = 1;                  //seta
pula ciclo triac dois
                                                                }
                                                        }
                                                }
                                        reinicia_bt();                                          // coloca nivel baixo nas
quatro variaveis
                        }
                if(triac_um)                                                                    // se triac um igual um
entao seleciona faixa temp saida um
                {
                        if(BT_um)
                        {
                                temp_one = 1;
                        }
                        if(BT_dois)
                        {
                                temp_one = 2;
                        }
                        if(BT_treis)
                        {
                                temp_one = 3;
                        }
                        if(BT_quatro)
                        {
                                temp_one = 4;
                        }

                        um_saida = 1;
                        confirma_um();
                        triac_um = 0;
                }

                if(triac_dois)                                                                  // se triac igual dois entao
seleciona faixa temp saida dois
                {
                        if(BT_um)
                        {
                                temp_two = 1;
                        }
                        if(BT_dois)
                        {
                                temp_two = 2;
                        }
                        if(BT_treis)
                        {
                                temp_two = 3;
                        }
                        if(BT_quatro)
                        {
                                temp_two = 4;
                        }

                        dois_saida = 1;
                        confirma_dois();
                        triac_dois = 0;
                }

                if(PL_frasetr)                          // funcao pula ciclo
                {
                        frase_triac = 1;
                        PL_frasetr = 0;
                }

                if(PL_trum)                                                     // funcao pula ciclo triac um
                {
                        triac_um = 1;                   // seleciona saida
```

```
                PL_trum = 0;
        }
        if(PL_trdois)                           // funcao pula ciclo triac dois
        {
                triac_dois = 1;                 // seleciona saida
                PL_trdois = 0;                  // as funcoes possuem posicao significativa no programa
        }
}

/**************************************
 *                                    frase partida                                    *
 **************************************/
void frase_partida(void)
{
        reinicia_bt();
        Delay10TCYx(1);                         // espera 10 microsegundos
        limpa_lcd();
        comando_lcd(0x80);                                      // posiciona o cursor na linha 0, coluna 0
        escreve_frase("Selecione Saida");       // imprime mensagem no lcd
        comando_lcd(0xC0);
        escreve_frase("Bt1:   Bt2:");
        if(pronto_um)
        {
                comando_lcd(0xC4);                      // posiciona o cursor na linha 0, coluna 4
                escreve_frase("Ok");            // imprime mensagem no lcd
        }
        if(pronto_dois)
        {
                comando_lcd(0xCb);                      // posiciona o cursor na linha 0, coluna 4
                escreve_frase("Ok");            // imprime mensagem no lcd
        }
        pronto_um = 0;
        pronto_dois = 0;
        mensagem = 0;                                   // fecha acesso funcao mensagem
        PL_frasetr = 1;                         // permite execução da funcao escolha semana

}

/**************************************
 *                          FUNÇÃO PARA SEPARAR DÍGITOS                          *
 **************************************/

void converte_bcd(unsigned char aux)
{
        unidade = 0;
        dezena = 0;
        centena = 0;
        if (aux == 0)return;  //SE AUX. FOR IGUAL A ZERO, SAI
        while(aux--)
        {
                unidade++;
                if(unidade != 10)continue;
                unidade = 0;
                dezena++;
                if(dezena != 10) continue;
                dezena = 0;
                centena++;
                if(centena != 10)continue;
                centena = 0;
        }
}
/**************************************
 *                          FUNÇÃO PARA converter pos ch um                          *
 **************************************/
void contra(void)                                       //realiza conversao apos atualizacao
        {                                                                               //con verte t emperatu ra
                SetChanADC(ADC_CH1);                    //seleciona ch um porta A0
                Delay10TCYx(1);
                ConvertADC();                                   //Inicia conversão AD
        if(!BusyADC());                                 //Aguarda fim da conversão AD
                Delay10TCYx(1);
                conversao = ADRESH;                             //lê resultado da conversão AD
                resultado = (conversao * 50); //faz regra de 3 para converter o valor,
                Delay10TCYx(1);                                 // espera 10 microsegundos
                termum = (resultado / 255);
        if(tela_valor)
```

```
                {
                Delay10TCYx(1);                                         // espera 10 microsegundos
                converte_bcd((unsigned char)termum);
                comando_lcd(0xC3);                                      //posiciona o cursor na linha 1, coluna 5
                escreve_lcd (dezena + 0x30); //escreve no display de LCD
                escreve_lcd (unidade + 0x30);           //escreve no display de LCD
                comando_lcd(0xC5);
                escreve_lcd(0xDF);
                escreve_lcd ('C');
                }
        }
/*************************************
 *                      FUNÇÃO PARA converter pos ch dois                  *
 *************************************/

void conch(void)                                                        //realiza conversao apos atualizacao
        {
                SetChanADC(ADC_CH2);                                    //seleciona ch um porta A0
                Delay10TCYx(1);                                         // espera 10 microsegundos
                ConvertADC();                                           //Inicia conversão AD
        if(!BusyADC());                                                 //Aguarda fim da conversão AD
                Delay10TCYx(1);                                         // espera 10 microsegundos
                convdois = ADRESH;                                      //lê resultado da conversão AD
                resultdois = (convdois * 50);   //faz regra de 3 para converter o valor,
                Delay10TCYx(1);                                         // espera 10 microsegundos
                termdois = (resultdois / 255);
        if(tela_valor)
                {
                Delay10TCYx(1);                                         // espera 10 microsegundos
                converte_bcd((unsigned char)termdois);
                comando_lcd(0xCB);                                      //posiciona o cursor na linha 1, coluna 5
                escreve_lcd (dezena + 0x30); //escreve no display de LCD
                escreve_lcd (unidade + 0x30);           //escreve no display de LCD
                comando_lcd(0xCD);
                escreve_lcd(0xDF);
                escreve_lcd ('C');
                }
        }

/*************************************
 *          Rotina de Tratamento de interrupção de TMR0          *
 *************************************/
// Esta interrupção ocorrerá a cada 256us.

void TRATA_INT_TIMER0(void)
{
        INTCONbits.TMR0IF = 0;      //LIMPA FLAG DE INTERRUPÇÃO
}
/*************************************
 *          Funcao de tratamento de interrupcao de Timer1        *
 *************************************/
// Esta interrupcao ocorrera a cada 0,5 segundos.

void TRATA_INT_TIMER1(void)
{
        PIR1bits.TMR1IF = 0;                            // limpa flag de interrupção
        WriteTimer1(delta_timer1);              // carrega timer1
        contador --;                                                    // decrementa contador de interrupções
        if (contador == 0)
        {
                WriteTimer1(delta_timer1);    // carraga timer1
                contador = 2;                                           // carrega contador de int
                decrementa_timer();
                if(BT_um)
                {
                        TRATA_botoes();
                }
                if(BT_dois)
                {
                        TRATA_botoes();
                }
                if(BT_treis)
                {
                        TRATA_botoes();
                }
                if(BT_quatro)
```

```
                {
                        TRATA_botoes();
                }
                contra();
                conch();
        }
}

 void TRATA_INT_TIMER2(void)
{
        PIR1bits.TMR2IF = 0;
}

/************************************
 *                Função Principal                *
 ************************************/

void main ()
{
        PORTA = 0x00;                   //Limpa PORTA
        PORTB = 0x00;                   //Limpa PORTB
        PORTC = 0x00;                           //Limpa PORTC
        PORTD = 0x00;                   //Limpa PORTD
        PORTE = 0x00;                   //Limpa PORTE

        LATA = 0x00;                    //Limpa PORTA
        LATB = 0x00;                   //Limpa PORTB
        LATC = 0x00;                            //Limpa PORTC
        LATD = 0x00;                   //Limpa PORTD
        LATE = 0x00;                   //Limpa PORTE


        TRISA = 0b11111111;             //CONFIG DIREÇÃO DOS PINOS PORTA UM ENTRADA ZERO SAIDA
        TRISB = 0b00001111;             //CONFIG DIREÇÃO DOS PINOS PORTB
        TRISC = 0b11111001;                     //CONFIG DIREÇÃO DOS PINOS PORTC
        TRISD = 0b00000000;             //CONFIG DIREÇÃO DOS PINOS PORTD
        TRISE = 0b00000100;             //CONFIG DIREÇÃO DOS PINOS PORTE

        while(RCONbits.NOT_TO);                                 //AGUARDA ESTOURO DO WDT

        OpenPWM1(249);
//      CCP1CON = 0b00001111;                   //CONFIGURAÇÃO DO MÓDULO CCP1 -> PWM
//      PR2 = 249;
        OpenPWM2(249);
//      CCP2CON = 0b00001111;                   //CONFIGURAÇÃO DO MÓDULO CCP2 -> PWM

//       setup_adc_ports;
        OpenADC(ADC_FOSC_8 & ADC_LEFT_JUST & ADC_0_TAD, ADC_CH0 & ADC_INT_OFF &
ADC_VREFPLUS_VDD & ADC_VREFMINUS_VSS, ADC_3ANA);
        //CONFIGURAÇÃO DO AD
        OpenTimer0(TIMER_INT_ON & T0_8BIT & T0_SOURCE_INT & T0_PS_1_2);
        //CONFIGURAÇÃO DO TIMER0
        OpenTimer1(TIMER_INT_OFF & T1_SOURCE_INT & T1_16BIT_RW & T1_PS_1_4 & T1_OSC1EN_OFF &
T1_SYNC_EXT_OFF);
        //CONFIGURAÇÃO DO TIMER1
        OpenTimer2(TIMER_INT_ON & T2_PS_1_16 & T2_POST_1_1);
        //CONFIGURA O TIMER 2


        INTCONbits.GIE = 1;                     //LIGA A CHAVE GERAL
/************************************
 *              Inicialização do Sistema                *
 ************************************/

        inicializa_lcd();
        funcao();                                       //mensagem mostra a funcao

/************************************
 *                Rotina Principal                *
 ************************************/

        while(1)
        {
                ClrWdt();                                                       //LIMPA O WDT

                if(!BT1)                                                        //BOTÃO 1 PRESSIONADO?
```

```
		{
			if (--FILTRO1 == 0)			//TERMINOU FILTRO DE BOTÃO?
			{
				if (!ST_BT1)				//AÇÃO DO BOTÃO FOI EXECUTADA?
				{
					BT_um = 1;				// BOTÃO 1
					ST_BT1 = 1;			//SETA FLAG DE ESTADO DO BOTÃO 1

				}
			}
			continue;
		}
		else ST_BT1 = 0;				// BOTÃO LIBERADO, LIMPA O FLAG


		if(!BT2)					//BOTÃO 2 PRESSIONADO?
		{
			if (--FILTRO2 == 0)			//TERMINOU FILTRO DE BOTÃO?
			{
				if (!ST_BT2)				//AÇÃO DO BOTÃO FOI EXECUTADA?
				{
					BT_dois = 1;			// BOTÃO 2
					ST_BT2 = 1;			//SETA FLAG DE ESTADO DO BOTÃO 2
				}
			}
			continue;
		}
		else ST_BT2 = 0;				// BOTÃO LIBERADO, LIMPA O FLAG

		if(!BT3)					//BOTÃO 3 PRESSIONADO?
		{
			if (--FILTRO3 == 0)			//TERMINOU FILTRO DE BOTÃO?
			{
				if (!ST_BT3)				//AÇÃO DO BOTÃO FOI EXECUTADA?
				{
					BT_treis = 1;			// BOTÃO 3
					ST_BT3 = 1;			//SETA FLAG DE ESTADO DO BOTÃO 3

				}
			}
			continue;
		}
		else ST_BT3 = 0;				// BOTÃO LIBERADO, LIMPA O FLAG

		if(!BT4)					//BOTÃO 4 PRESSIONADO?
		{
			if (--FILTRO4 == 0)			//TERMINOU FILTRO DE BOTÃO?
			{
				if (!ST_BT4)				//AÇÃO DO BOTÃO FOI EXECUTADA?
				{
					BT_quatro = 1;			// BOTÃO 4
					ST_BT4 = 1;			//SETA FLAG DE ESTADO DO BOTÃO 4

				}
			}
			continue;
		}
		else ST_BT4 = 0;				// BOTÃO LIBERADO, LIMPA O FLAG

		FILTRO1 = 400;				//RECARREGA FILTRO BT1
		FILTRO2 = 400;				//RECARREGA FILTRO BT2
		FILTRO3 = 400;				//RECARREGA FILTRO BT3
		FILTRO4 = 400;				//RECARREGA FILTRO BT4

	}// FIM LAÇO PRINCIPAL
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// *		ROTINA DE TRATAMENTO DE INT DE ALTA PRIORIDADE		*
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

#pragma code VETOR_INT_HIGH = 0x0008	//VETOR DE ALTA PRIORIDADE
void VETOR_INT_HIGH (void)
{
  _asm goto TRATA_HIGH_INT _endasm
}
#pragma code
```

```
#pragma interrupt TRATA_HIGH_INT
void TRATA_HIGH_INT(void)
{
	if(INTCONbits.TMR0IF)	TRATA_INT_TIMER0();
	if(PIR1bits.TMR1IF)	TRATA_INT_TIMER1();
	if(PIR1bits.TMR2IF)	TRATA_INT_TIMER2();
}
```