UNIVERSIDADE DO EXTREMO SUL CATARINENSE - UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

ANDRÉ FELTRIN BALDESSAR

CONTROLE DE TEMPERATURA POR MEIO DE COMPUTAÇÃO EMBARCADA

CRICIÚMA, JULHO DE 2008

ANDRÉ FELTRIN BALDESSAR

CONTROLE DE TEMPERATURA POR MEIO DE COMPUTAÇÃO EMBARCADA

Trabalho de Conclusão de Curso apresentado para obtenção do Grau de Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense.

Orientador: Prof. M.Eng. Evânio Ramos Nicoleit.

CRICIÚMA, JULHO DE 2008

Dedico este trabalho a todos os que compreenderam o quão importante é para minha satisfação profissional.

AGRADECIMENTOS

Agradeço a todas as pessoas do meu convívio que acreditaram e contribuíram, mesmo que indiretamente, para a conclusão deste curso.

A minha mãe, que foi compreensiva nas decisões unidirecionais que tomei no transcorrer dos estudos. Aos amigos, que souberam entender a importância da conclusão e do aperfeiçoamento profissional. A todos que sentiram minha falta nos momentos de confraternizações, em que a ausência foi justificada por si só.

A professora Divânia, que prontamente se dispôs em me auxiliar nas traduções para o Inglês. Ao meu orientador Evânio, que acreditou que era possível e indicou o melhor rumo para que, enfim, eu pudesse finalizar as pesquisas com resultado positivo.

"Uma pessoa inteligente resolve um problema,
um sábio o previne."
Albert Einstein

RESUMO

Esta pesquisa aborda o desenvolvimento de um protótipo, formado de hardware e software, cuja principal função é o controle de temperatura. Utiliza conceitos de computação embarcada, pois a lógica do controle é embutida em um *chip* que realiza esta função, proporcionando dimensões físicas reduzidas se comparado ao uso de uma plataforma de computador pessoal.

O hardware proposto é dividido em três partes principais: aquisição, controle e atuação. A aquisição acontece por meio de um sensor de temperatura, que gera um sinal elétrico proporcional à temperatura em que se encontra. Este sinal então é lido por um microcontrolador que abriga um software desenvolvido em linguagem de baixo nível e tem a função de calcular qual dos dois atuadores disponíveis será ligado e em qual potência, ou seja, realizar o controle. Estes atuadores são representados por uma resistência com o objetivo de aquecer, e um ventilador com o objetivo de resfriar o ambiente.

Uma *interface* desenvolvida para *Microsoft Windows* é responsável por adquirir os dados repassados pelo microcontrolador através da porta serial, e apresentá-los ao usuário por meio de um gráfico de histórico. Outra função importante é enviar, também pela porta serial, uma temperatura de estabilização escolhida pelo usuário.

O protótipo controla os atuadores com base no valor de temperatura estabelecido pelo usuário e no sinal do sensor. Quando mais distante a temperatura do sensor está da temperatura estabelecida, maior a potência de atuação.

Palavras-Chave: Comunicação Serial, Microcontroladores, Microchip PIC, Controle Proporcional, *Assembly*, Computação Embarcada, Temperatura.

ABSTRACT

This research approaches the development of a prototype, formed by hardware and software, with temperature control as main function. It uses concepts of embedded computing, as the logic of the control is inside the chip that it carries out this function, providing reduced physical dimensions if compared to a personal computer platform.

The considered hardware is divided into three main parts: acquisition, control and activation. The acquisition happens by means of a temperature sensor, that generates an electric signal proportional to the temperature. This signal is read by a microcontroller that shelters a software developed in low-level programming language and has the function to calculate which of the two available activators will be on and in which power, that is, to carry out the control. These actuators are represented by a resistance with the aim of heating, and a fan with the aim of cooling the environment. An interface developed to Microsoft Windows is responsible for acquiring the data repassed by the microcontroller through the serial port, and presenting them to the user by means of a historical graph. Another important function is to send, also through the serial port, a stabilization temperature chosen by the user.

The prototype controls the activators based on the value of temperature established by the user and in the signal of the sensor. The more distant the temperature of the sensor is of the established temperature, the bigger the performance power is.

Keywords: Serial Communication, Microcontrollers, Microchip PIC, Proportional Control, Assembly, Embedded Computing, Temperature.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

RAM	*Random Acess Memory*
USART	*Universal Synchronous Asynchronous Receiver Transmitter*
PC	*Personal Computer*
EPROM	*Erasable Programmable Read-Only Memory*
PWM	*Pulse Width Modulation*
Vcc	Volts Corrente Contínua
Vca	Volts Corrente Alternada
DC	*Direct Current*
AC	*Alternate Current*
TTL	*Time to Live*
RISC	*Reduced Instruction Set Computer*
P	Proporcional
PI	Proporcional Integral
PD	Proporcional Derivativo
PID	Proporcional Integral Derivativo
PE	Ponto de Equilíbrio
TL	Temperatura Lida
TRIAC	*Triode for Alternating Current*
CAD	*Computer-Aided Design*
HTML	*HyperText Markup Language*
WEB	*World Wide Web*

SUMÁRIO

1 INTRODUÇÃO

Com o avanço da tecnologia, é cada vez mais crescente o número de sistemas computacionais de controle e automação. Seja no dia a dia, seja na indústria, a importância disto pode influenciar diretamente no tempo e na qualidade dos serviços.

Na indústria aviária, a cadeia produtiva começa antes mesmo do ovo eclodir. E é principalmente nesta fase em que o desenvolvimento do frango adulto terá seu início. Diretamente ligado a temperatura, a proporção de peso e tamanho do pinto eclodido se mantém até a fase do abate, sendo necessário o uso de hormônios para corrigir este fator. Dada a importância desta fase, a computação vêm auxiliar de forma a melhor controlar a temperatura de incubação do ovo, por meio de técnicas de controles proporcionais e uso de microcontroladores.

Algumas aplicações de uso industrial que utilizam sistemas computacionais ainda utilizam plataformas completas de computadores, com sistemas operacionais completos e hardware desnecessário. Neste projeto, foi utilizado um microcontrolador com um hardware especialista, minimizando o custo final do aparelho. Foram abordadas algumas formas de controle, algumas melhores e mais eficientes, outras menos complexas.

Foi utilizado o microcontrolador Microchip PIC para controle do projeto. Ele possui conceitos semelhantes a plataformas de microprocessadores de computadores pessoais, com alguns hardwares e recursos adicionais.

1.1 OBJETIVO GERAL

Desenvolver um sistema de controle de temperatura em hardware e software embarcado.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste trabalho de conclusão de curso são:

a) compreender e utilizar programação em baixo nível para controle de equipamentos (A*ssembly*);

b) aplicar tecnologia de controle lógico de hardware via microcontroladores da família *Microchip PICFlash®;*

c) proporcionar uma alternativa para substituição do uso de computadores pessoais para a função de controle;

d) produzir um hardware para ativação de equipamentos eletrônicos;

e) interfacear software com hardware via porta serial.

1.3 JUSTIFICATIVA

Aplicações embarcadas sugerem o uso de microcontroladores especializados na função controle, fazendo com que haja necessidade de integração entre hardware e software. Neste sentido, fica evidenciado que um software vai além da programação para computadores pessoais. Há meios de implementar um programa para microcontroladores utilizando a linguagem C, mas segundo Marinoni (2003), esta linguagem possui uma otimização limitada, já que é de alto nível. Desta forma, implementar um programa em baixo nível, como o A*ssembly*, significa dar comandos ao microcontrolador de maneira direta e otimizável, resultando em menor complexidade computacional.

Com o objetivo de escolher um microcontrolador que se adapte à necessidade do projeto, foram analisadas algumas características. No momento da escolha, há que se levar em conta que para cada aplicação existe um modelo que pode oferecer a melhor relação custo-benefício (VAGLICA, 1990). A família PICFlash da fabricante americana Microchip possui características importantes e desejáveis, como memória de programa que dependendo do modelo possui capacidade para mais de 2048 instruções, memória RAM e oscilador internos, encapsulamento comum do tipo DIP utilizado principalmente em circuitos integrados, temporizadores e contadores internos, controle de erro, além de uma interface USART integrada para comunicação serial (MICROCHIP, 2005).

A comunicação serial torna-se interessante, pois os computadores novos e antigos possuem esta interface, facilitando tanto na forma de gravação do programa no microcontrolador quanto na comunicação com um computador pessoal para troca de informações.

A aplicação deste projeto, um processo de controle de temperatura, resultou um estudo de caso. Com estes dados, o microcontrolador, por meio dos comandos de programação implementados em *Assembly*, pode controlar a temperatura de um ambiente (ou forno, por exemplo) de forma a estabilizar o processo.

Computação embarcada aplicada na área de veterinária é uma especialidade em que o país não tem histórico de produção de conhecimento e tecnologia. Este projeto pode prover base para pesquisas no âmbito da veterinária, na área da avicultura. Uma linha de pesquisa nesta área poderá gerar trabalhos correlatos, contribuindo assim para que a comunidade científica produza pesquisas de domínio público.

1.4 ESTRUTURA DO TRABALHO

A estrutura do trabalho está organizada em 6 capítulos. Este Capítulo 1 contextualiza a introdução referente ao trabalho, com a definição do objetivo desse estudo e também os objetivos específicos, concluindo então com a justificativa do trabalho realizado. O Capítulo 2 apresenta um caso de uso que valida o projeto, relacionando a temperatura e avicultura. Para controlar a temperatura, o Capítulo 3 aborda um projeto microcontrolado, com detalhes de cada etapa, como aquisição e controle. Ao final de deste capítulo, grande parte dos conceitos envolvidos no projeto estarão claros, para que nos Capítulos 4 e 5, as alternativas para a função controle sejam apresentadas. No Capítulo 6 os detalhes do projeto desenvolvido são esclarecidos, e por meio dos Apêndices, todas as etapas são documentadas.

2 TECNOLOGIA E AVICULTURA

O processo produtivo no desenvolvimento de aves para abate reúne uma série de recursos tecnológicos utilizados para controle do meio, como climatização e iluminação. Segundo Jordan (2005) a tecnologia é um dos fatores responsáveis por colocar o Brasil na segunda posição mundial na produção e exportação de aves. Tal importância justifica o estudo mais eficiente do uso da tecnologia, não só no processo de engorda e abate dos frangos, mas também na fase de incubação do ovo.

2.1 A AVICULTURA E A TEMPERATURA

Uma pesquisa conduzida pela Embrapa analisou 61.920 ovos a fim de verificar os efeitos positivos e negativos que a temperatura exerce sobre o processo de eclosão. Os resultados indicam que o uso da temperatura ideal minimiza a mortalidade dos pintos.

Não há um consenso sobre a temperatura no qual o embrião começa a se desenvolver. Decuypere (1992) considera que a faixa de temperatura ideal de armazenamento do ovo antes do desenvolvimento está entre 19-28ºC, ou seja, a partir deste limite o embrião começa a se desenvolver culminando na eclosão.

> A utilização da temperatura de 28,6ºC, medida pelo termômetro de bulbo úmido, aplicada durante o período de incubação, propiciou a otimização dos resultados de eclosão e eclodibilidade, independentemente da idade da matriz e da categoria de peso do ovo, e também reduziu a mortalidade embrionária total. (ROSA, 2002, p. 1015).

A diferença de temperatura ideal e estagnação do desenvolvimento do pinto, por serem muito próximas, exigem um bom controle. Para esta tarefa, a computação oferece soluções diversas, algumas delas com grande e outras com menor precisão.

2.2 Soluções da Computação para Controle da Temperatura

Por vezes, o uso de um computador para controle de variáveis do meio é uma solução prática. No caso estudado, um computador monitoraria sensores ligados em um dos seus barramentos de entrada e por meio de um software específico, controlaria aquecedores ligados a um dos seus barramentos de saída. Considerando que o poder de processamento dos computadores atuais está em torno de 1GHz, possivelmente haveria um grande desperdício dos recursos da máquina.

O mercado oferece uma opção interessante para aplicações que exigem o controle de uma variável por meio de sensores e atuadores: os Microcontroladores. Segundo Chandler (2006) estes circuitos integrados em forma de *chip* programáveis têm sido usados em indústrias de larga escala, principalmente na China. Entre as aplicações, destacam-se soluções de controle embarcadas, como medidores de variáveis elétricas e aparelhos monitores de pressão e temperatura. Desta forma, o hardware e software tornam-se especialistas na área em que o aparelho atua. Utilizando o hardware com os recursos mais próximos ao necessário, espera-se reduzir o custo total do sistema e facilitar o processo em uma eventual industrialização.

3 COMPUTAÇÃO EMBARCADA

Nos anos quarenta, quando a eletrônica surgiu para auxiliar na solução de problemas do cotidiano, sistemas com funções específicas utilizavam basicamente componentes discretos. Isto originava máquinas que utilizavam grandes espaços físicos e conseqüentemente, grandes fábricas.

A partir dos anos sessenta, foi possível integrar funções de cálculo para que, nos anos oitenta, também grande parte das funções lógicas fossem embutidas em um único *chip*. Nos dias de hoje existem circuitos eletrônicos diminutos dotados de microcontroladores com lógica programável, que garantem eficiência no uso de espaço físico. Como resultado, é possível encontrar aparelhos portáteis de entretenimento do tamanho de uma moeda, ou sistemas de controle embarcados capazes de manobrar veículos.

Computação Embarcada pode ser definida como um sistema computacional com propósitos específicos, normalmente construído em dimensões reduzidas, que deve funcionar de forma autônoma. (JUNG, 2005), o que torna possível realizar algumas considerações a respeito das pesquisas até agora desenvolvidas.

As pesquisas computacionais regionais, para a função controle, originam sistemas que utilizam plataformas de computadores pessoais (PC) completas. Para esta função específica, isto representa ociosidade de algumas partes do hardware e processamento desnecessário com tarefas inerentes de um sistema operacional. Estes fatores excluem este tipo de sistema do conceito de computação embarcada. Um sistema só pode ser embarcado se a função controle se der por meio de um hardware que seja projetado especificamente para esta função e que deve funcionar de forma autônoma, no caso desta pesquisa, por meio do uso de microcontroladores.

4 PROJETO MICROCONTROLADO PARA CONTROLE DE TEMPERATURA

O projeto de controle de temperatura baseado em um microcontrolador, em uma visão geral, pode ser dividido em três partes: aquisição, controle e atuação. Cada uma das partes necessita da anterior para funcionar corretamente, embora cada uma tenha seu próprio sistema de funcionamento.

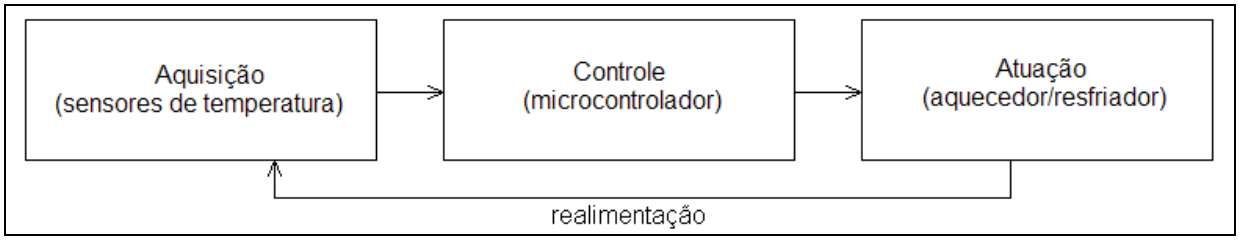


Figura 1. Diagrama do projeto proposto

Um fator importante em um projeto de leitura de temperatura é o fato do sistema se realimentar, demonstrado na Figura 1. Desta forma, o controle pode atuar de forma ágil e precisa sobre o meio, aquecendo ou resfriando.

4.1 AQUISIÇÃO DA TEMPERATURA

A temperatura é uma grandeza termodinâmica comum aos corpos em equilíbrio térmico. É devida à agitação térmica das partículas (átomos ou moléculas) que constituem um corpo. Quanto maior for à agitação destas partículas, maior será a temperatura do corpo. Quanto menor a agitação das partículas menor será a temperatura do corpo (GÜÉMEZ, 1998).

Transdutor é um dispositivo que converte uma forma de energia em outra. Utiliza para isso um elemento denominado sensor que recebe os dados e os transforma em energia elétrica. A finalidade de sua utilização em circuitos eletrônicos são: medição e controle (NATALE, 2000). Um transdutor de temperatura converte energia térmica

em tensão elétrica (KAUFMAN, 1984). Alguns transdutores para medida de temperatura são: termopares, termo resistências, termistores, pirômetros, circuitos integrados específicos, entre outros.

É por meio de sensores que a temperatura será convertida em sinal elétrico no qual o microcontrolador ‘entenda’. Comercialmente há inúmeras soluções.

Para medir a temperatura de um corpo (ou ambiente) utiliza-se uma escala[1] física. Dentre elas podem ser citadas as escalas Celsius, Kelvin e Fahrenheit. Esta última geralmente utilizada nos Estados Unidos e países do Norte da Europa. Geralmente, é necessária uma conversão de escala via software no microcontrolador.

Na escala Celsius, são definidos como 0 (zero) grau, para o ponto de congelamento da água, e como 100 (cem) graus, para o ponto de ebulição da água, considerando para altitude de referência o nível do mar. O referido intervalo é subdividido em 100 partes, correspondentes a um grau Celsius. Para converter a temperatura da escala Celsius para a escala Fahrenheit, pode-se utilizar a seguinte expressão:

$$°F = (1{,}8 \cdot °C) + 32.$$

Na escala Fahrenheit, o ponto de congelamento da água corresponde a 32 graus Fahrenheit e o ponto de ebulição da água a 212 graus Fahrenheit. Este intervalo é subdividido em 180 unidades corresponde a um grau Fahrenheit. Fialho (2005) demonstra como converter a temperatura da escala Fahrenheit para a escala Celsius utilizando a seguinte expressão:

$$°C = \frac{5}{9} \cdot (°F - 32)$$

[1] Conjunto ordenado de marcas, associado a qualquer numeração, que faz parte de um dispositivo indicador.

4.1.1 SENSOR DE TEMPERATURA DE PRECISÃO LM35

Fabricado pela National Semiconductor, o componente eletrônico LM35 é um circuito integrado que varia sua tensão de saída em aproximadamente 10mV por grau Celsius, de forma linear. Obviamente esta relação não é perfeita devido a particularidades inerentes ao processo de fabricação. Em um estudo realizado na Universidade de Caxias do Sul (BRUSAMARELLO, 2007), em três momentos obteve-se uma curva linear próxima do ideal, com pequenas variações entre eles. A Figura 2 ilustra o gráfico obtido com a terceira medição.

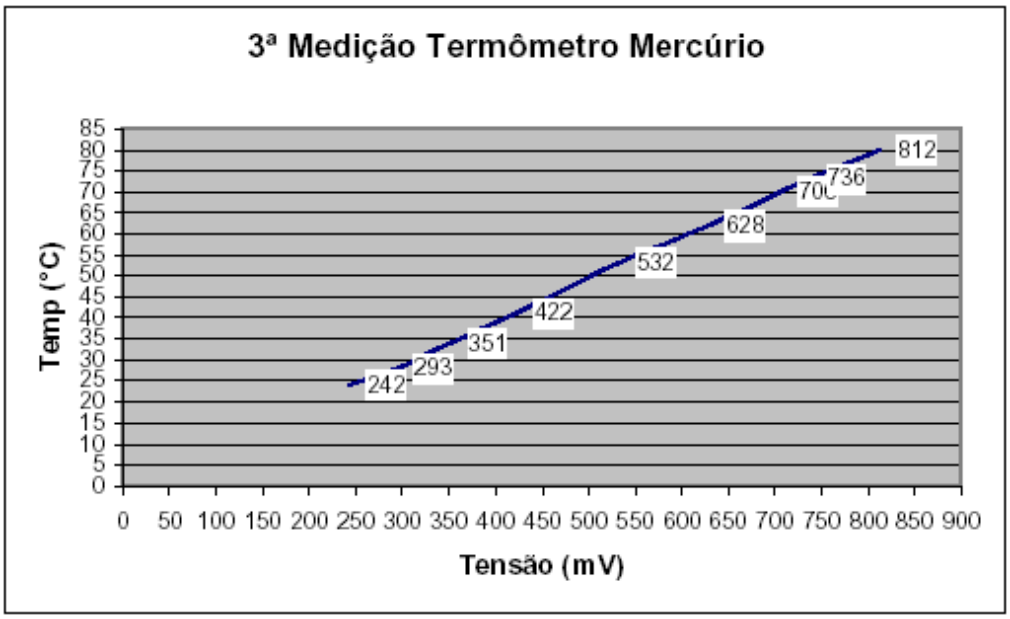


Figura 2. Última medição realizada em pesquisa de laboratório
Fonte: BRUSAMARELLO, V. (2007)

A partir dos dados apresentados, pode-se concluir que a curva de resposta do componente que mede a temperatura é bastante próxima do ideal, validando seu uso.

Encontra-se no mercado brasileiro o LM35 encapsulado em um invólucro denominado TO92. Este tipo de encapsulamento, indicado na Figura 3, tem como característica o tamanho reduzido, uma face achatada e três pinos de comunicação. Segundo fabricante, é possível alimentá-lo com uma tensão de até 35V entre os pinos

+Vs e GND. A medida que a temperatura começa a subir, a tensão entre os pinos Vout e GND acompanha o movimento, chegando até o limite de 180ºC, variando 0.01V por grau Celsius.

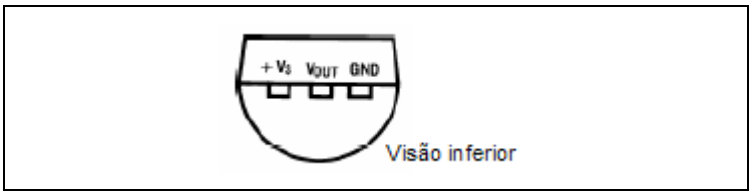


Figura 3. Componente LM35, sensor de temperatura
Fonte: NATIONAL SEMICONDUCTOR (1994)

O componente pode ser ligado diretamente ao microcontrolador, se este tiver a capacidade de decodificar o sinal analógico de forma digital. Geralmente os microcontroladores utilizam uma tensão de referência para comparar com o sinal do sensor e fazer a conversão de sinal analógico para digital. Se a tensão de referência for muito alta a conversão analógico-digital perde resolução, como mostra a Tabela 1. Usualmente utilizam uma tensão de referência não menor que 3.5Vcc. Os fabricantes recomendam utilizar a tensão da fonte.

Tabela 1. Tabela de resolução com referência alta.

Valor analógico do Sensor	**Valor digital Convertido**
0 Vcc / 0°C	0
0.01 Vcc / 1°C	0
0.08 Vcc / 8°C	4
0.2 Vcc / 20°C	10
0.5 Vcc / 50°C	25
5 Vcc / 500°C	255

A solução para resolver o problema da resolução é amplificar o sinal do sensor. Amplificar significa multiplicar o valor obtido por um valor chamado ganho. E para definir o ganho desta solução, basta definir uma temperatura que corresponda ao valor de referência.

No caso estudado, a temperatura será definida em 50ºC, pois é aproximadamente duas vezes o valor da temperatura de eclosão do ovo. Portanto, antes

da amplificação 50ºC corresponde a 0,5V, e depois da amplificação a 5V. Calcula-se então um ganho de 10 vezes.

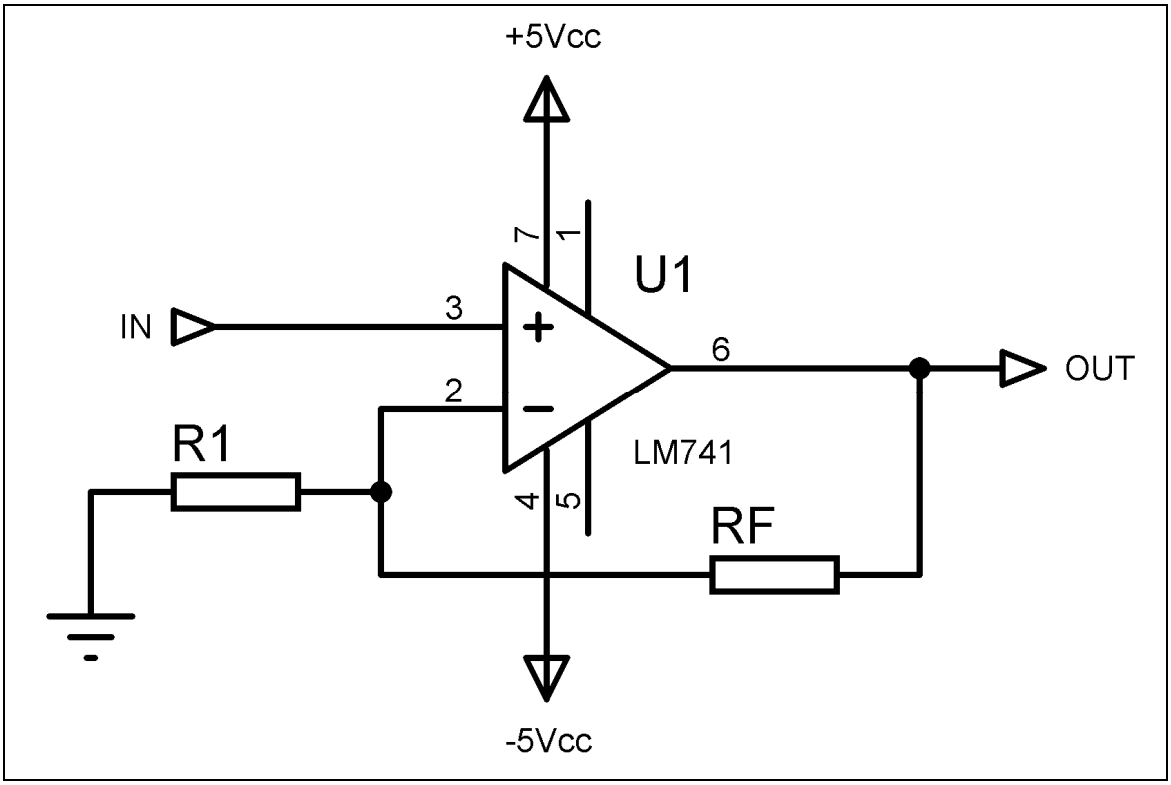


Figura 4. Circuito amplificador com LM741
Fonte: Adaptado de NATIONAL SEMICONDUCTOR (2000)

O circuito integrado LM741 é um amplificador operacional em que é possível configurar o ganho através de resistores externos. O fabricante nomeia o resistor de realimentação de RF, e estipula uma fórmula para cálculo do ganho:

$$Ganho = 1 + (RF / R1)$$

Como o ganho já é um valor conhecido (10), os valores de resistores para RF e R1 que possibilitam esta relação são, respectivamente, 9kΩ e 1kΩ.

4.2 CONTROLE

Fazendo uma analogia com o corpo humano, esta parte do projeto é o cérebro do sistema e também a mais importante. Utilizando um microcontrolador, a

temperatura fornecida em sinal analógico pelo conjunto sensor e amplificador será convertida pelo hardware em um número binário que poderá ser manipulado por um software. Cada microcontrolador possui recursos diferentes que variam de acordo principalmente com o preço. Para este projeto o fabricante escolhido foi a Microchip e o modelo, PIC, por apresentar suporte técnico no Brasil e uma ampla gama de bibliografia abordando-o.

4.2.1 MICROCONTROLADORES

O microcontrolador é um componente eletrônico dotado de uma lógica programável. Segundo Souza (2006) o microcontrolador é programável, pois toda a lógica de operação é estruturada na forma de um programa e gravada dentro do componente eletrônico. A Unidade Lógica Aritmética do componente fica responsável por todas as operações lógicas e aritméticas.

4.2.2 O MICROCHIP PIC

PIC é o nome de uma família de microcontroladores fabricada pela *Microchip Technology*. O fabricante não utiliza acrônimos para definir *PIC*. Inicialmente o fabricante produzia este microcontrolador para substituir o tratamento de tarefas de entradas e saídas de circuitos microprocessados, liberando recursos. Com o passar dos anos, foi sendo aprimorado com periféricos e tipos de memórias como *EPROM*, e mais recentemente memória *FLASH*.

Os recursos de hardware fabricados dentro do *chip*, segundo o fabricante Microchip (2002), minimizam o uso de componentes externos, reduzindo o custo do sistema. A medida que os recursos foram sendo compactados no *chip*, a família de

microcontroladores foi dividida em três partes: Linha Básica, Linha Intermediária e Linha de Alto Desempenho.

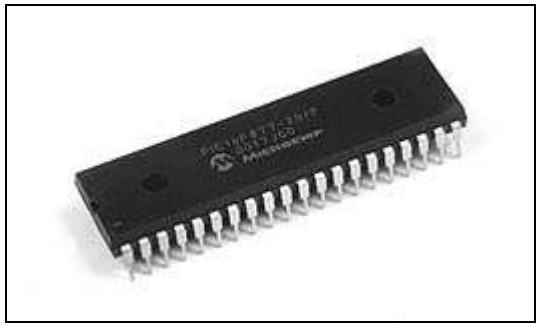


Figura 5. *Chip* PIC 16F877A
Fonte: MICROCHIP (2005)

O programa lógico é escrito em uma linguagem específica e opera com níveis de tensão nos pinos do *chip*. Assim, é possível obter informações do meio em que irá operar, e conseqüentemente controlá-lo.

4.2.3 A ESCOLHA DO MICROCONTROLADOR PIC

O custo acompanha os recursos oferecidos. Portanto, para definir custo, é necessário mapear quais os recursos serão utilizados no projeto.

Um ponto importante a ser considerado é a forma de gravação dos microcontroladores. Quando a memória de programa é do tipo *Erasable Programmable Read-Only Memory* (*EPROM*), é necessário haver um gravador de memória especial, já que para que seja reutilizada, é necessário aplicar radiação ultravioleta no orifício da parte superior do *chip*.

Há uma subfamília de microcontroladores PIC na Linha Intermediária denominados 16F, que utilizam memória *FLASH* para gravação dos dados, de forma elétrica, eliminando a necessidade do uso de gravadores com luz ultavioleta e tornando o processo mais simples.

As duas opções disponíveis no mercado brasileiro da Linha Intermediária 16F são o PIC16F628A e PIC16F877A. Podem-se comparar estes dois modelos em suas seguintes características:

Tabela 2. Comparativo de dois modelos de PIC com memória *FLASH*

	16F628A	**16F877A**
Pinos	18	40
Portas de I/O	16	33
Programação	14bits/35 instruções	14bits/35 instruções
Memória de Programa	2048 palavras	8192 palavras
Interrupções	10	15
Pulse Width Modulation	Não	Sim
Conversor A/D Interno	Não	Sim
Custo	~R$10,00	~R$25,00

Fonte: SOUZA, D. (2006), LAVINIA, N. (2006)

Os aspectos construtivos do *PIC* 16F877A mostram um conversor analógico-digital interno. Para fins construtivos, este fator é desejável, pois não será necessária uma porção de software no *chip* que garanta a aquisição e integridade dos dados. Também há um módulo de *Pulse Width Modulation* (PWM), que é útil no que diz respeito a exteriorizar sinais digitais em forma de sinais analógicos, requisito este também desejável.

4.2.4 A ARQUITETURA DO PIC 16F877A

Os microcontroladores *PIC* utilizam uma arquitetura diferente das utilizadas por microprocessadores usuais. A principal diferença é que a arquitetura Von-Neumann utilizada em microprocessadores compartilha o mesmo caminho físico interno para transportar dados da memória de programa e da memória de dados. Em uma máquina com a arquitetura de Harvard, o processador pode ler e escrever instruções e dados para

a memória ao mesmo tempo, resultando em velocidade e por outro lado, complexidade (SANCHEZ, 2006).

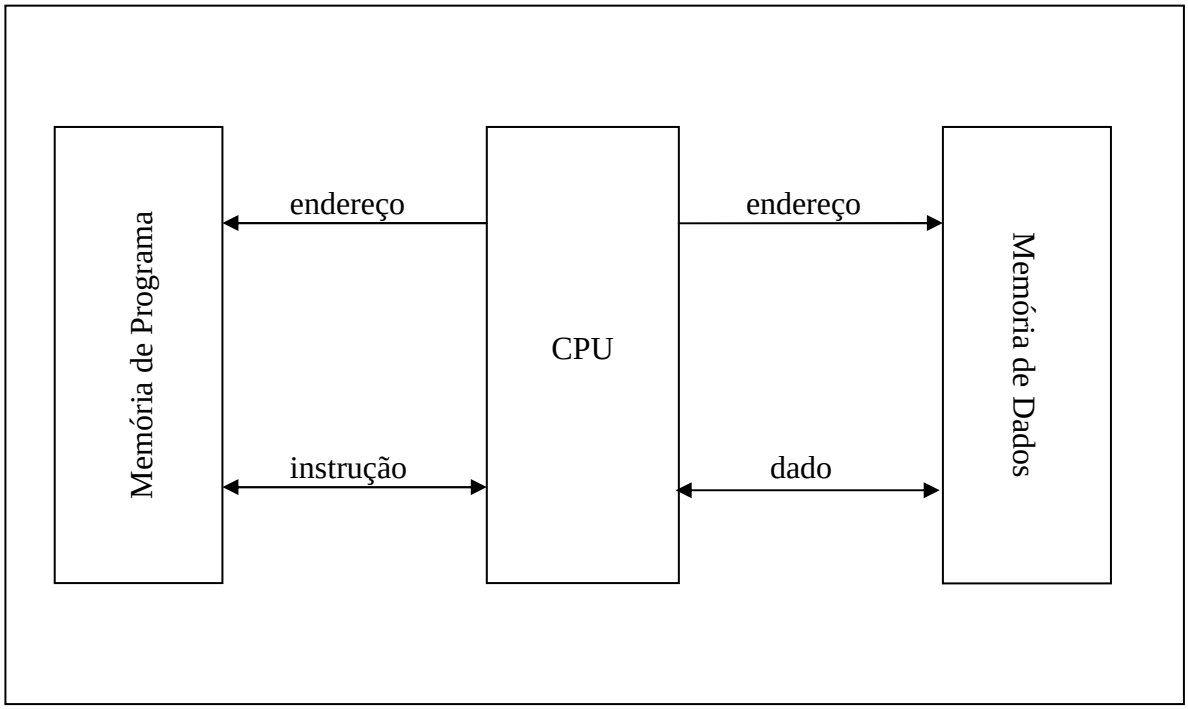


Figura 6. Arquitetura de Harvard
Fonte: Adaptado de SANCHEZ, J. (2006)

A arquitetura é um fator que influencia no modo com que o recurso de PWM funciona, pois será com base nos pulsos do *clock* que o sinal analógico será modulado. Internamente o *clock* da máquina é dividido por quatro, assim como as operações referentes à execução e busca da instrução na memória. Assim, é necessário 4 pulsos para executar uma instrução. Isto possibilita a aplicação da técnica de *pipeline*, que consiste no fato de que enquanto uma instrução está sendo executada, outra instrução pode ser buscada na memória de programa e carregada na CPU. Com um *clock* de 4MHz, os pulsos ocorrem a cada 0,25µs e cada instrução simples é executada em 1µs. Demais fatores referentes ao cálculo do período do PWM serão apresentados em outro momento.

Outros recursos especiais podem ser visualizados pelo diagrama de blocos da Figura 7 a seguir, como três temporizadores, dois comparadores (entradas analógicas),

uma saída analógica, possibilidade para uso de memória EEPROM externa e uma interface para comunicação serial. Este último recurso é especialmente importante, pois poderá ser utilizado para comunicação com um Microcomputador para troca de dados.

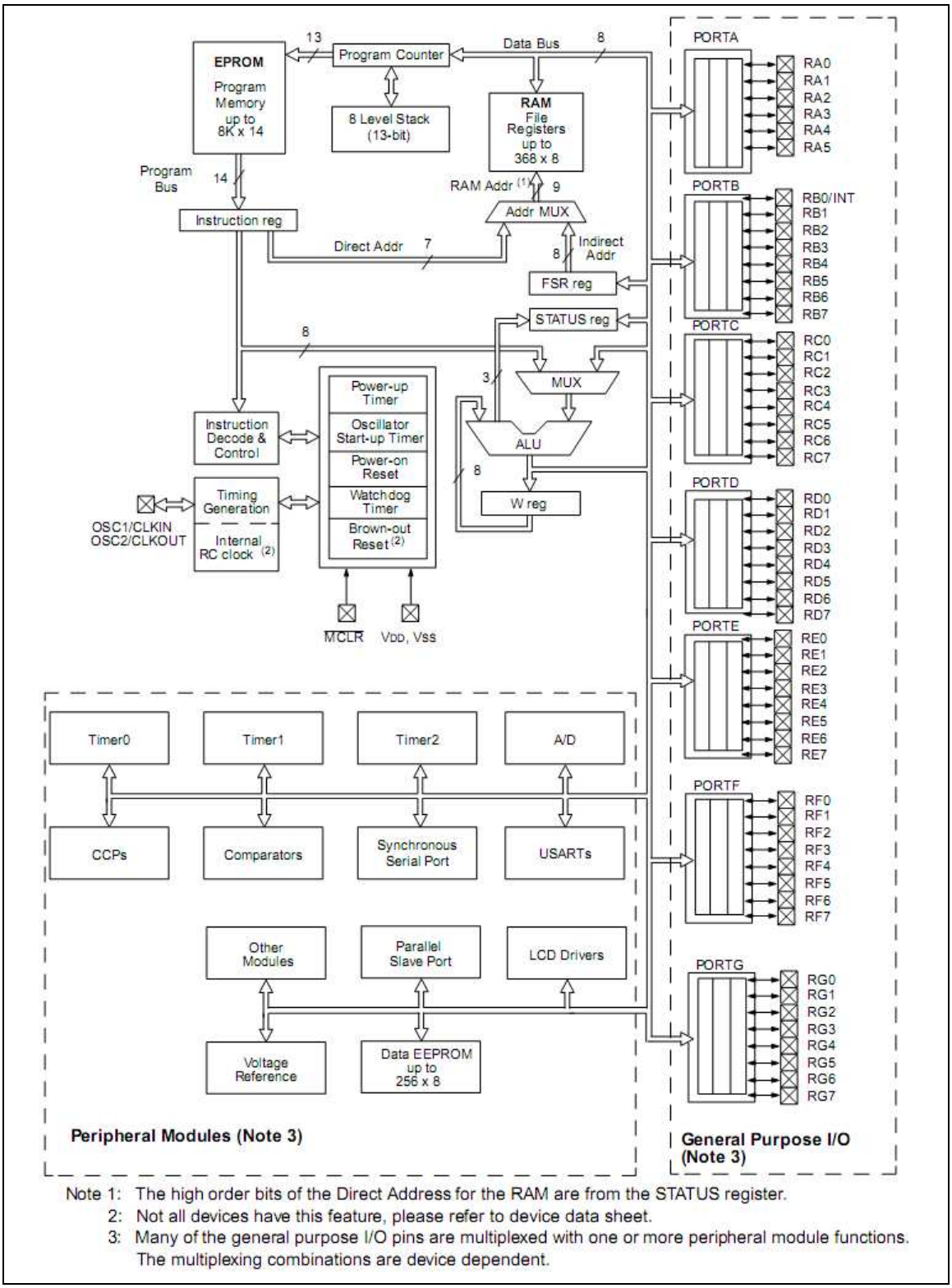


Figura 7. Diagrama de blocos geral de microcontroladores da família PICmicro
Fonte: MICROCHIP (2005)

4.2.5 COMUNICAÇÃO SERIAL

A comunicação serial é útil para comunicação entre *chips* ou entre *chip* e microcomputador. Quando não há sincronismo, ou seja, não há uma via de dados para informar aos dois lados da comunicação sobre o tempo correto para transmissão de cada bit, é necessário utilizar circuitos integrados externos para comunicação e para estabelecer no transmissor e receptor o tamanho dos dados transmitidos. Este tamanho é um dos atributos do protocolo *RS-232-C,* que é utilizado na grande maioria dos microcomputadores (SANCHEZ, 2006). É por meio deste atributo que os dados são sincronizados, sendo ele o divisor de uma unidade de tempo para obter-se a freqüência.

Comercialmente disponível no mercado brasileiro, o MAX-232, circuito integrado fabricado pela Texas Instruments, promove uma interface entre o PIC 16F877A, que funciona com o protocolo *TTL* e uma entrada serial que implementa o protocolo RS-232-C. Ele também se torna importante protegendo o barramento *Universal Synchronous Asynchronous Receiver Transmitter* (USART) do microcontrolador, caso o cabo até o receptor rompa. O esquema de ligação na Figura 8 exemplifica o uso deste circuito integrado.

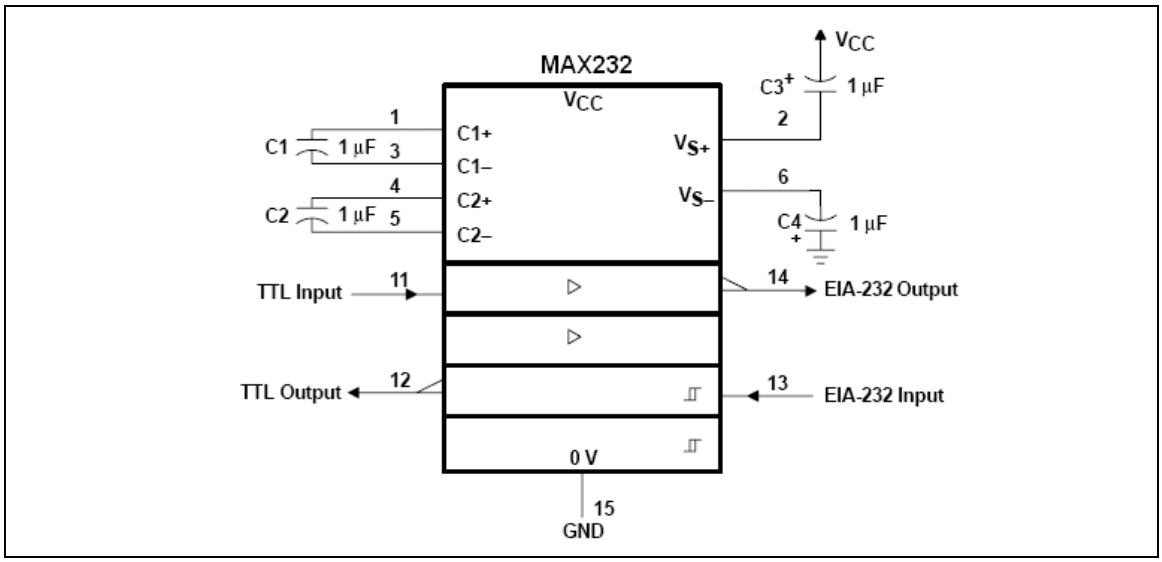


Figura 8. Esquema de Ligação do MAX232.
Fonte: TEXAS INSTRUMENTS (2002).

4.2.6 COMUNICAÇÃO COM O MEIO EXTERNO

Um microcontrolador que não se comunica com o meio externo, seja por entradas ou saídas, não tem motivos para funcionar. São por meio destes recursos que os sinais elétricos de sensores são captados e atuadores, como aquecedores e resfriadores, são ativados. O manual do PIC 16F877A lista as funções dos 40 pinos do *chip*, sendo as de uso comum demonstradas na Tabela 2.

Tabela 3. Funções principais dos pinos do *chip* PIC 16F877A.

Pinos	Funções Principais	Funções Secundárias
1~7	Port A (6 entradas ou saídas digitais), Reset	- Entradas analógicas, - Porta serial síncrona, - Entrada de pulso timer 0.
8~10	Port E (3 entradas ou saídas digitais)	- Entradas analógicas - Comunicação porta paralela
11,32	+Vcc (positivo da fonte de alimentação)	-
12,31	GND (negativo da fonte de alimentação)	-
13,14	Entrada e saída do oscilador externo (cristal)	- Pulso de clock externo - Saída de ¼ do tempo do oscilador
15~18 23~26	Port C (8 entradas ou saídas digitais); - TX (envio) do módulo serial; - RX (recebimento) do módulo serial.	- Entrada e saída do oscilador do timer 1 - Saídas dos dois módulos de PWM internos, entrada do módulo de captura e comparação - Clock da entrada serial síncrona.
19~22 27~30	Port D (8 entradas ou saídas digitais)	- 8 bits de comunicação com a porta paralela
33~40	Port B (8 entradas ou saídas digitais)	- Interrupção externa; - Controle de verificação de erro no programa.

Fonte: Adaptado de MICROCHIP (2005).

Cada função do microcontrolador corresponde a um pino no *chip*. Por questões de espaço físico, a maioria dos pinos não possuem função exclusiva, sendo configuráveis via software. É comum um projeto necessitar utilizar um recurso cujo

pino já está sendo utilizado. Nestes casos, recomenda-se o uso de outro microcontrolador.

Os pinos importantes para este projeto são os pinos 25 e 26 (RX e TX da USART), 16 e 17 (saídas PWM), 3 (entrada analógica com conversor A/D), e demais pinos de entradas e saídas.

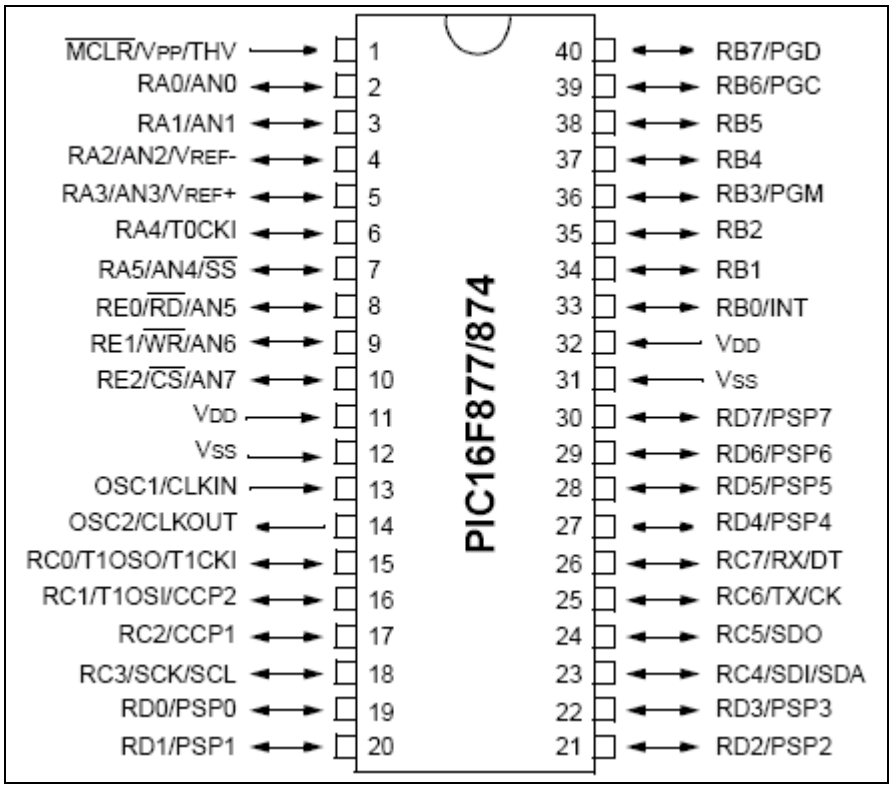


Figura 9. Disposição das funções no *chip*
Fonte: MICROCHIP (2005)

Observa-se na Figura 9 várias funções. Para controlá-las, é necessário que o microcontrolador seja dotado de um software. Para este modelo, o programa é escrito na linguagem *Assembly*.

4.2.7 O ASSEMBLY DO PIC 16F877A

Cada aplicação necessita de um software. Isto é o que diferencia circuitos eletrônicos comuns de circuitos microcontrolados, pois neste último é possível

programar como os atuadores devem se comportar com base nos dados de entrada. No caso de microcontroladores, o software é formado por instruções simples em código *Assembly*, que é a linguagem de programação mais baixa existente.

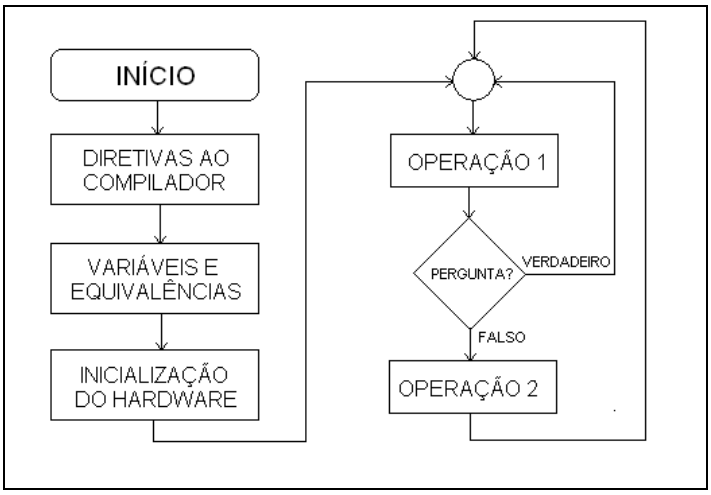


Figura 10. Diagrama demonstrativo do programa principal
Fonte: Adaptado de SOUZA, D. (2006)

O fabricante do *chip* optou por utilizar uma *lista de instruções de máquina reduzido* (RISC). Desta maneira, a programação fica reduzida a 35 palavras reservadas, que realizam as operações básicas do microcontrolador. “Isto torna o aprendizado muito mais fácil e dinâmico, mas, por outro lado, implica no fato de que muitas funções devem ser ‘construídas’, pois não possuem uma instrução direta, exigindo maior habilidade do programador” (SOUZA, 2006, p. 23).

Observa-se na Figura 10 algumas operações presentes no programa em *Assembly* escrito para o microcontrolador PIC. Algumas operações dizem respeito ao compilador e variáveis do sistema, não sendo gravados no *chip*. Já outras, são necessárias para configurar pinos com mais de uma função. Geralmente os compiladores mais modernos incluem interfaces de desenvolvimento que facilitam a programação, contendo recursos como abstração de endereços de memória para variáveis inteligíveis. Porém a forma de ‘escrever’ instruções não muda.

As instruções do *chip* ficam guardadas na memória de programa e são executadas de modo seqüencial. É necessário adicionar ao início da memória algumas operações de configuração de hardware, para que o *chip* possa identificar qual o modo de operação irá trabalhar.

Tabela 4. Lista de instruções completa do PIC 16F877A

Comando	Parâmetros	Descrição	Ciclos de Clock Gastos
ADDWF	f, d	Add W and f	1
ANDWF	f, d	AND W with f	1
CLRF	f	Clear f	1
CLRW	—	Clear W	1
COMF	f, d	Complement f	1
DECF	f, d	Decrement f	1
DECFSZ	f, d	Decrement f, Skip if 0	1 ou 2
INCF	f, d	Increment f	1
INCFSZ	f, d	Increment f, Skip if 0	1 ou 2
IORWF	f, d	Inclusive OR W with f	1
MOVF	f, d	Move f	1
MOVWF	f	Move W to f	1
NOP	—	No Operation	1
RLF	f, d	Rotate Left f through Carry	1
RRF	f, d	Rotate Right f through Carry	1
SUBWF	f, d	Subtract W from f	1
SWAPF	f, d	Swap nibbles in f	1
XORWF	f, d	Exclusive OR W with f	1
BCF	f, b	Bit Clear f	1
BSF	f, b	Bit Set f	1
BTFSC	f, b	Bit Test f, Skip if Clear	1 ou 2
BTFSS	f, b	Bit Test f, Skip if Set	1 ou 2
ADDLW	K	Add literal and W	1
ANDLW	k	AND literal with W	1
CALL	k	Call subroutine	2
CLRWDT	—	Clear Watchdog Timer	1
GOTO	k	Go to address	2
IORLW	k	Inclusive OR literal with W	1
MOVLW	k	Move literal to W	1
RETFIE	—	Return from interrupt	2
RETLW	k	Return with literal in W	2
RETURN	—	Return from Subroutine	2
SLEEP	—	Go into Standby mode	1
SUBLW	k	Subtract W from literal	1
XORLW	k	Exclusive OR literal with W	1

Fonte: Adaptado de MICROCHIP (2005).

Conforme mostra a Tabela 4, uma instrução é composta por: operação e argumentos. Um exemplo é a instrução BTFSC *f,b*. Ela testa (BT, *bit test*) o *bit b* do registrador (F) *f* e salta (S) a próxima linha de programa se for 0 (C).

Utilizando esta lista de instruções, é possível realizar uma lógica de controle e atuação para o microcontrolador, ou seja, para que ele leia a temperatura e decida qual atuador deve utilizar e em qual potência.

4.3 Formas de Controle da Temperatura

Existem algumas formas de controle que são usualmente aplicadas. Algumas com mais eficiência e com um sistema de controle mais complexo, e outras com menor eficiência, porém com um sistema de controle simplificado. Serão apresentadas as principais, para que ao fim se possa escolher uma forma adequada a esta pesquisa. Como forma de ilustração, a variável temperatura foi escolhida para explicação dos controles.

4.3.1 Liga/Desliga (Controle ON/OFF)

Esta é a forma mais simples de controle e é utilizada em diversas aplicações, desde sistemas de arrefecimento em veículos até chuveiros elétricos. Quando o sensor de temperatura informa ao controlador uma temperatura inferior à programada, imediatamente é acionado um dispositivo, que aquece o meio. O ponto principal deste modo de controle é o fato de que, como o próprio nome sugere, há somente dois estados: ligado e desligado.

Potência = 100% ou 0%.

4.3.2 Controle Proporcional

Uma evolução da forma Liga/Desliga, este tipo utiliza um valor pré-determinado multiplicado pela variação da temperatura para determinar a potência (ou

tensão) a ser aplicada ao dispositivo de aquecimento ou resfriamento. Seu uso é delicado em situações onde a temperatura varia bruscamente, pois a potência do dispositivo tenta seguir a variação.

Potência = Ganho * Variação

4.3.3 CONTROLE DERIVATIVO PROPORCIONAL (PD)

Difere-se do Controle Proporcional pois possui uma taxa de variação ajustável. Este fator corrige em parte o problema de variação brusca da temperatura, fazendo com que a potência aplicada ao dispositivo siga adequadamente a variação.

Potência = (Ganho * Variação) + Taxa de correção

4.3.4 CONTROLE INTEGRAL DERIVATIVO PROPORCIONAL (PID)

Com aplicações científicas e industriais, esta forma de controle envolve além dos fatores Proporcional e Derivativo, também o fator Integral. Deste modo, a carga será energizada com uma potência até que a temperatura se estabilize ao nível de programado. Quando isto acontece, a potência do sinal tende a zero.

Potência = (Ganho * Variação) + Taxa de correção + Correção Integral.

3.3.5 CONTROLE INTEGRAL PROPORCIONAL (PI)

Em um sistema de controle tipo PD, quando há ruído elétrico no sensor de temperatura, o controle derivativo é ativado e faz com que a potência aplicada ao

dispositivo varie de modo equivocado. Neste caso é sugeria a substituição do controle derivativo pelo controle PI. Neste caso, as variações devida a ruído são eliminadas.

4.3.6 O CONTROLE APLICADO AO PROJETO

Como esta pesquisa envolve programação em baixo nível (*Assembly*), optou-se por utilizar o controle proporcional por razões de baixa complexidade, eficiência alcançada e estabilidade de variação por parte dos dispositivos de atuação a serem utilizados.

A temperatura é uma grandeza variável em um sistema de controle. À medida que a temperatura sobe ou desce, o microcontrolador define qual atuador deve ser acionado e a sua potência. Quem define a inclinação das retas de acionamento é o ponto de equilíbrio (reta pontilhada no gráfico da Figura 11). Quanto mais perto do ponto de equilíbrio, menor a potência que será aplicada ao atuador.

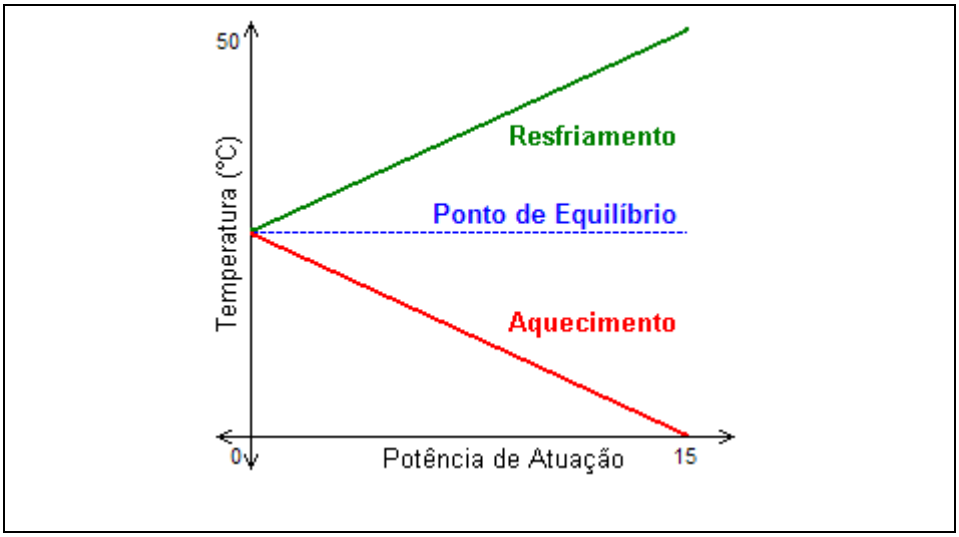


Figura 11. Gráfico de um controle de temperatura ideal

Como é necessário converter o sinal digital de potência de atuação calculado pelo controle (retas inclinadas) em dado analógico, a Figura 11 representa uma condição que não se aplica ao microcontrolador. Isto acontece pois assim como o conversor

analógico-digital utilizado para converter a temperatura, um conversor digital-analógico utilizado para converter a potência digital e potência analógica, possui uma resolução fixa. Para fins de pesquisa, 16 níveis de potência para cada atuador são satisfatórios.

Fazendo uma releitura do gráfico apresentado, aplicando o conceito da resolução, pode-se obter um gráfico em níveis, seguindo a mesma lógica a situação ideal.

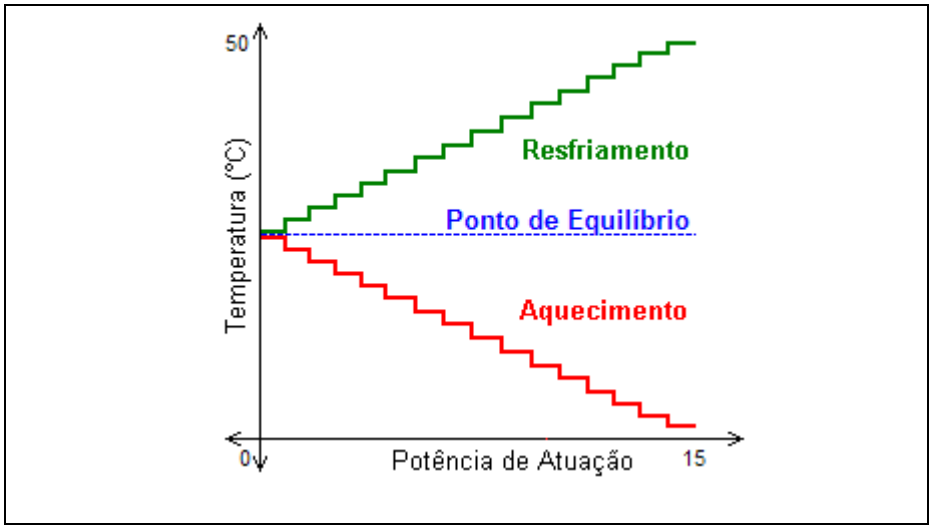


Figura 12. Gráfico de um controle com resolução

Por considerar a potência zero como um nível, a potência máxima de aquecimento não é relativa a temperatura mínima da escala. Na verdade, ela é alcançada antes. Isto vale para o resfriamento também.

Matematicamente, pode-se verificar a lógica do processo de escolha da potência do atuador por meio dos testes abaixo definidos.

$$\textit{Aquecimento:} \quad (PE/16)*PT < TL < (PE/16)*(PT+1)$$

$$\textit{Resfriamento:} \quad PE+(((50-PE)/16)*PT) < TL < PE+(((50-PE)/16)*(PT+1))$$

Onde PE é o Ponto de Equilíbrio representado pela linha pontilhada na Figura 12, PT é a potência do atuador e TL é a temperatura lida. Esta lógica matemática

será transformada em lógica *Assembly*, que é a linguagem de programação que o microcontrolador trabalha.

O hardware do projeto é quem definirá o quanto a potência será perceptível. Se a intenção será utilizar atuadores com potências maiores, uma porção de software adicional será necessária.

4.3.7 Controle e Atuação

O microcontrolador PIC possui uma função especial chamada *Modulação de Largura de Pulso* (PWM). Com ela, é possível realizar uma conversão digital-analógica, pois gera uma onda quadrada, cuja parte alta pode ser controlada. Pode-se ver a operação do PWM na Figura 13 e 14.

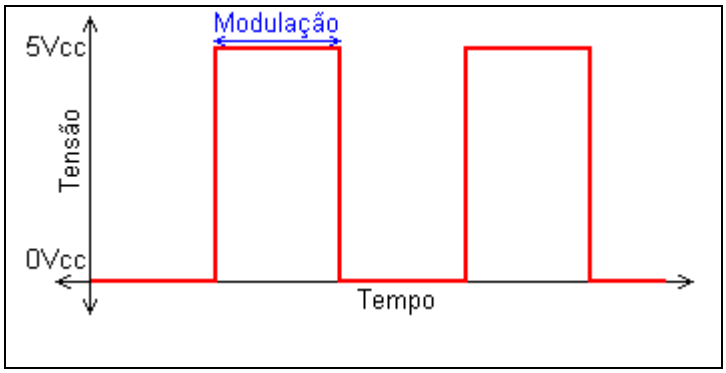


Figura 13. PWM operando em 50%.

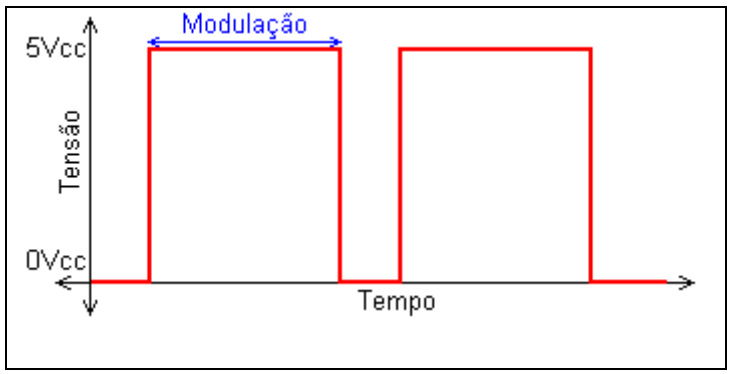


Figura 14. PWM operando em 75%.

Como resultado, a saída no pino do microcontrolador é um sinal pulsante, cuja parte alta é de maior duração, ou seja, a carga recebe tensão o tempo todo, ou ainda de menor duração, onde a carga não recebe tensão e portanto o atuador não atua.

Com o modelo de microcontrolador utilizado, é possível modular até 1024 pontos, ou seja, pode-se variar a modulação em intervalos de até 0,01% . Como neste projeto este não é um fator crítico, a resolução será minimizada para intervalos de 6,25%, ou seja, 16 pontos. A freqüência dos pulsos, por outro lado, depende de fatores externos, como a freqüência de pulsos de *clock*. Utilizando um cristal de 4MHz, a freqüência mínima de pulsos do PWM é de 250Hz.

Espera-se poder controlar um equipamento, linearmente, através deste pulso de PWM. Lavinia (2006) comprova que é possível controlar cargas *direct current* (DC), ou seja, cargas que trabalham com corrente contínua. Segundo ele, é mais simples o uso deste tipo de sinal, pois os componentes eletrônicos envolvidos já realizam a conversão digital analógico. Na Figura 17 será apresentado o esquema eletrônico.

Por vezes, como neste projeto, é requerido controle AC para cargas de corrente alternada (AC), ou seja, que trabalham em corrente alternada. Isto acontece porque este tipo de corrente corresponde a um sinal sinusoidal que oscila sessenta vezes por segundo, na rede elétrica comum.

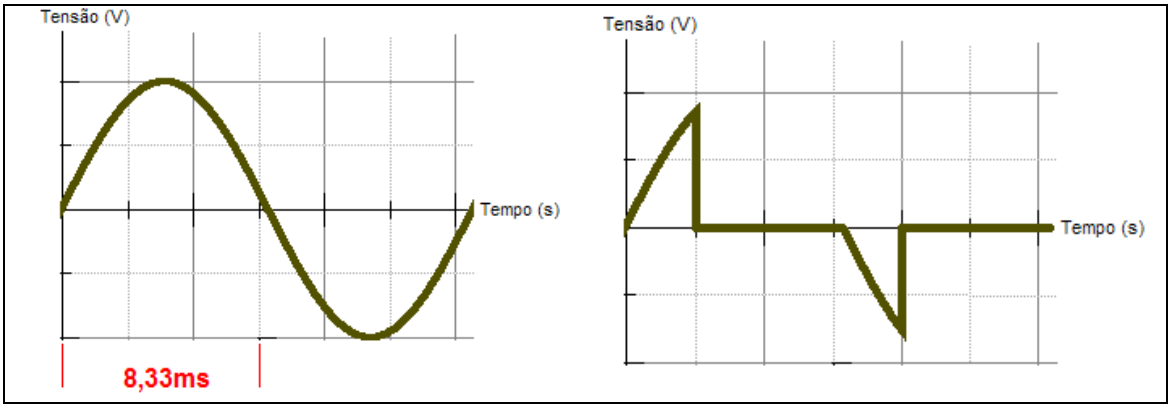


Figura 15. Senóide da rede elétrica representando o controle

É necessário que o controlador detecte a passagem da senóide pela tensão zero, para então liberar um pulso com duração de 0 a 8,33ms, que corresponde a potência de 0 a 15 calculada por software. Na Figura 15 pode-se visualizar a senóide completa, com a potência 15 e a senóide controlada, com uma potência menor. Caso esta passagem por zero não for detectada por software, o PWM irá disparar a carga em qualquer momento, não sendo possível um controle linear efetivo.

4.4 ATUADORES

Um equipamento que deseja controlar uma variável necessita de um atuador ou carga. Dependendo da grandeza a ser controlada e da direção que ela assume, o atuador muda. A grandeza temperatura pode ser controlada por uma resistência que aquece com relação à corrente elétrica que passa por ela e/ou por um resfriador, que resfria um sistema quando as rotações de sua hélice aumentam. Portanto, exemplos de atuadores podem ser resistências e ventoinhas.

Geralmente os atuadores necessitam, para acionamento, de uma corrente elétrica muito maior do que a disponibilizada por equipamentos de controle baseados em microcontroladores. Dependendo do tipo de corrente e da potência da carga, o hardware de relacionamento muda.

4.4.1 ATUAÇÃO POR CARGAS RESISTIVAS (AQUECEDORES)

Na situação de aquecimento, as cargas resistivas são adequadas, pois tendem a aquecer baseados na corrente e tensão que passa sobre o componente. Resistores e lâmpadas incandescentes são exemplos comuns.

*Potência Dissipada = Tensão * Corrente*

É importante observar que a tensão de trabalho do controlador não passa de 5Vcc e alguns miliamperes, sendo que a do atuador pode chegar a 220Vca e vários amperes. Devido a esta diferença, é necessário um hardware adicional para que se possa fazer um acoplamento.

Uma medida proposta por Braga (2008) para atuadores AC consiste em utilizar um acoplador ótico para isolar o circuito de controle e potência.

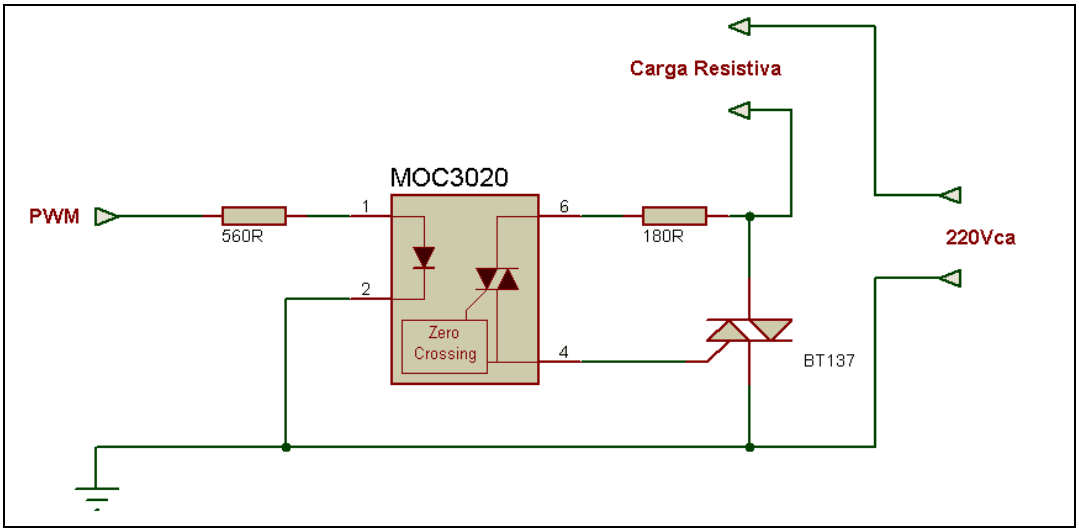


Figura 16. Circuito para acionamento de cargas resistivas
Fonte: Adaptado de BRAGA (2008)

Pode-se observar na Figura 16 que o acoplador ótico MOC3020 divide o circuito em duas partes distintas. A esquerda está a parte de controle e a direita está a parte de potência. Como o PWM gera pulsos que variam seu período, a tensão sobre a carga resistiva irá ser aplicada de forma gradual. É importante observar que o software deve ser capaz de gerar os pulsos a cada passagem por zero. Souza (2006) sugere utilizar as interrupções geradas por pulso de *clock* externo, sendo esta função dedicada do microcontrolador, através de um simples resistor de 4k7 ligado da rede elétrica ao pino do *chip*.

Como esta parte do projeto já foi testada pelo autor, os componentes envolvidos não serão abordados de forma detalhada. O acoplador ótico que aparece no diagrama funciona com lógica ON/OFF. Enquanto é alimentado, libera a saída. Como

este chaveamento se dá internamente com luz, ele é ideal para fins de isolamento. Já o TRIAC BT137 é um tiristor cujo chaveamento pode ser controlado por meio do DIAC interno que o MOC3020 possui.

Interfacear o microcontrolador através de tensões DC é relativamente mais simples. O fator importante neste modo é a potência dissipada pelo transistor Q1. Ela é calculada através da fórmula *(144/R2)*.

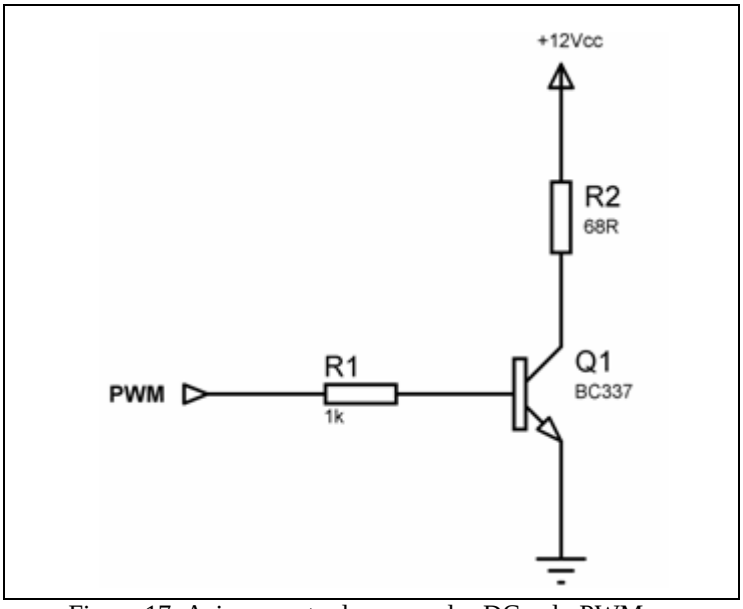


Figura 17. Acionamento de aquecedor DC pelo PWM
Fonte: LAVINIA (2006)

4.4.2 ATUAÇÃO POR CARGAS INDUTIVAS (RESFRIADORES)

Este tipo de carga consiste em ventiladores e bobinas, geralmente usados para resfriamento. Mesmo funcionando similar a cargas resistivas, geram uma influência diferente sobre os componentes envolvidos no modo AC. No momento em que a tensão é aplicada sobre a carga, um pico de corrente acontece e pode danificar o TRIAC.

Como se vê na Figura 18, a forma de comutação requer a adição de um filtro que ‘amorteça’ este impacto. Simplesmente chamado de *Snubber*, configura-se de um

resistor e de um capacitor em paralelo com o TRIAC. Para reduzir o risco de dano ao circuito, o resistor deve possuir uma potência de 2W e o capacitor deverá ser de poliéster e 400V.

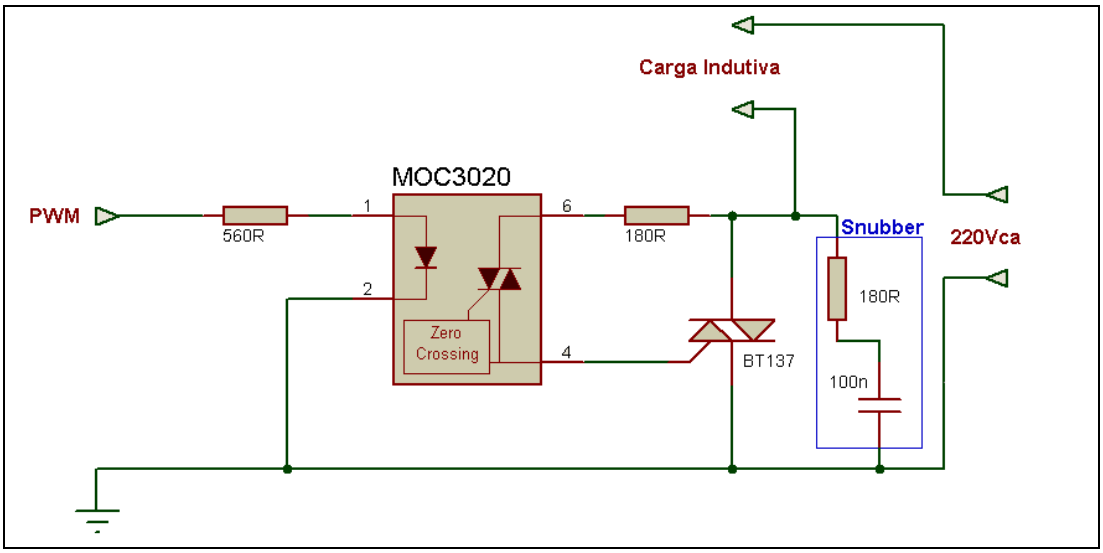


Figura 18. Circuito para acionamento de cargas indutivas
Fonte: Adaptado de BRAGA (2008)

Do mesmo modo, para o modo DC pode-se utilizar um circuito similar, com um diodo em paralelo com a carga. A função é a mesma do filtro *Snubber*.

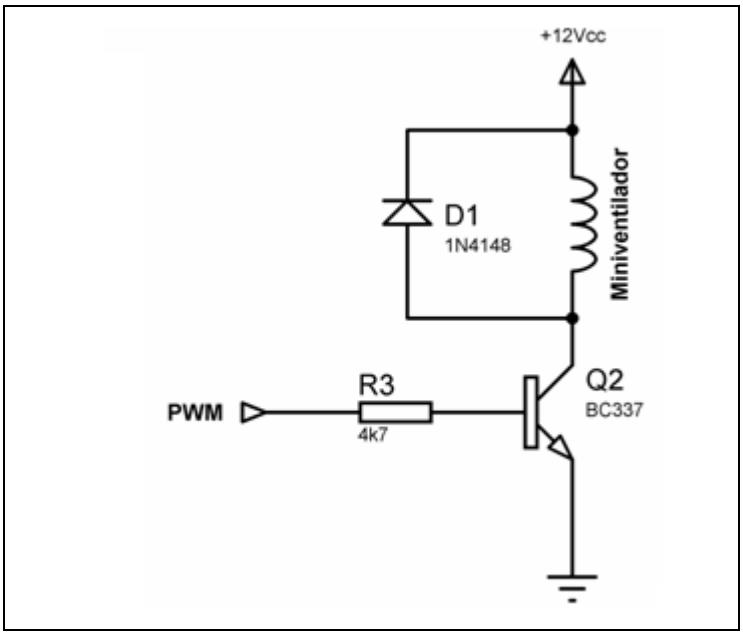


Figura 19. Acionamento de aquecedor DC pelo PWM
Fonte: LAVINIA (2006)

5 TRABALHOS CORRELATOS

No decorrer dos estudos referentes a este trabalho, foram analisados alguns trabalhos no âmbito da tecnologia. Por se tratar de uma área não abordada na graduação de Ciência da Computação, a maioria das pesquisas existentes encontram-se de posse da iniciativa privada, porém também será abordada.

5.1 AQUISIÇÃO E CONTROLE DE TEMPERATURA EM 8 CANAIS

Trabalho de conclusão de curso de graduação em Ciência da Computação, Sistema de Monitoramento e Controle da Temperatura em 8 Canais por meio de Hardware e Software, desenvolvido na Universidade do Extremo Sul Catarinense publicado em 2007 que aborda a aquisição e controle de temperatura em uma extrusora de polímeros. Neste trabalho foi utilizado um microcomputador completo e a forma de controle da temperatura foi Liga-Desliga.

Como objeto final, obteve-se um circuito eletrônico que tem a capacidade de se comunicar com a porta paralela do microcomputador a fim de transmitir e receber dados de um sistema computacional responsável pelo monitoramento, análise, gravação e controle da temperatura.

5.2 PROTÓTIPO PARA MONITORAMENTO DE TEMPERATURA NO PROCESSO DE INCUBAÇÃO

Pesquisa de conclusão de curso de graduação em Sistemas de Informação pela Universidade do Sul de Santa Catarina, este trabalho visa o desenvolvimento de um

conjunto de hardware e software que possa monitorar a temperatura no processo de incubação de ovos de aves.

O protótipo de sistema fazia aquisição e registro de dados. Entretanto, o controle não fora implementado. Era necessário a intervenção do usuário. Mesmo assim, o sistema demonstrou a precisão necessária no registro dos dados da temperatura no processo crítico de incubação.

5.3 Características do desenvolvimento embrionário de Gallus Gallus Domesticus, em temperaturas e períodos diferentes de incubação

Pesquisa realizada na Universidade Federal de Santa Catarina que expões os resultados de análises relacionando a temperatura no período de incubação de aves com o tamanho e peso do embrião e ave eclodida. Dentre as conclusões apresentadas, uma vem de encontro ao trabalho proposto: a temperatura é um fator decisivo na qualidade dos futuros frangos, influenciando diretamente no tamanho final e consequentemente na quantidade de hormônios que será aplicada para que ele atinja o tamanho ideal de abate.

5.4 Aplicação Prática na Iniciativa Privada

A empresa Tecnnic Eletrônica Industrial, de Criciúma trabalha com microcontroladores PIC desde 2003. Dentre as aplicações para os equipamentos fabricados por eles, está um controlador de temperatura em aviários, do tipo Liga-Desliga e Proporcional. O código fechado é vendido para empresas fabricantes de equipamentos que comercializam os produtos.

Este tipo de iniciativa é aceitável, pois recebe investimento alto. Porém não beneficia o pequeno produtor associado a cooperativas, que se vê obrigado a comprar o produto final por um preço cuja mão de obra especializada no desenvolvimento está agregada.

6 ESTUDO DE CASO

A fim de por em prova o uso de um microcontrolador para controle proporcional de temperatura, utilizou-se um caso de uso que vem de encontro à realidade da economia regional.

Na avicultura moderna, que é baseada essencialmente em lucro, a quantidade de hormônios presentes nos frangos tende a crescer se o processo de eclosão do pinto no ovo não for devidamente assistido. As indústrias avicultoras inserem hormônios na alimentação do frango para que o peso final do produto seja lucrativo.

Para auxiliar o produtor no sentido de que a qualidade do pinto seja a melhor possível, este trabalho visa desenvolver um protótipo que integra o uso de um microcontrolador para realizar um controle proporcional da temperatura de um modo prático.

7 TRABALHO DESENVOLVIDO

O objetivo desta pesquisa foi gerar um protótipo que possa controlar a temperatura de um determinado corpo. Ao final, obteve-se um equipamento no qual foi possível testar as teorias estudadas e comprovar seu uso.

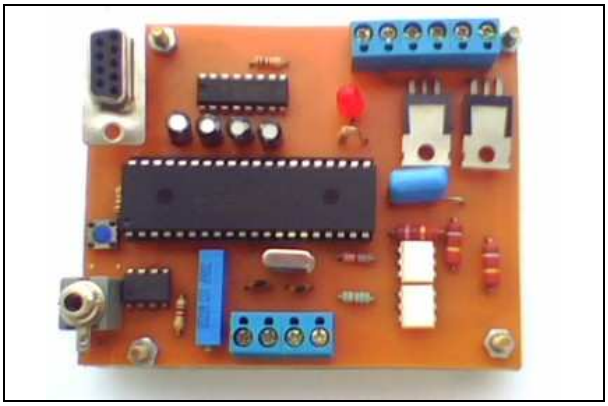


Figura 20. Protótipo final desta pesquisa

O equipamento demonstrado na Figura 20 é formado de hardware, que abriga uma parte do software. Este conjunto é responsável pelo controle de dois atuadores. Há também outra porção de software que serve para registro dos dados obtidos e interface para que o usuário especifique a temperatura ideal planejada.

Suficientes para provar os objetivos deste trabalho, foram utilizadas cargas para corrente contínua. Este fator influenciou diretamente no software, pois não foi necessário programar uma função PWM específica para detectar a passagem por zero da rede elétrica, uma vez que não foi utilizada. Porém, para facilitar futuras implementações e continuação das pesquisas, a placa de circuito impressa confeccionada, que abriga o circuito eletrônico, foi preparada para controlar cargas para corrente alternada, bastando apenas uma modificação no software para isto.

7.1 SOFTWARE DO MICROCONTROLADOR

As bibliografias pesquisadas orientam a produzir um fluxograma de processos antes de iniciar a programação em baixo nível. Segundo Souza (2006), torna-se então menos custoso ao programador transcrever o fluxograma em programa *assembly*, comparado à transcrição de idéias.

7.1.1 FLUXOGRAMA DO PROGRAMA

O objetivo do fluxograma é representar a lógica do programa. Com isto é possível que até mesmo alguém que não tenha familiaridade com o código, entenda os modos de operação. No caso do controle de temperatura, deve representar as configurações iniciais de hardware, a aquisição da temperatura, o cálculo do controle, a

comunicação pela porta serial e exteriorização do controle. Na Figura 21, está representado a parte do fluxograma do sistema embarcado responsável pelo cálculo da potência do resfriador, uma das funções principais do sistema. Todos os fluxogramas encontram-se no Apêndice B.

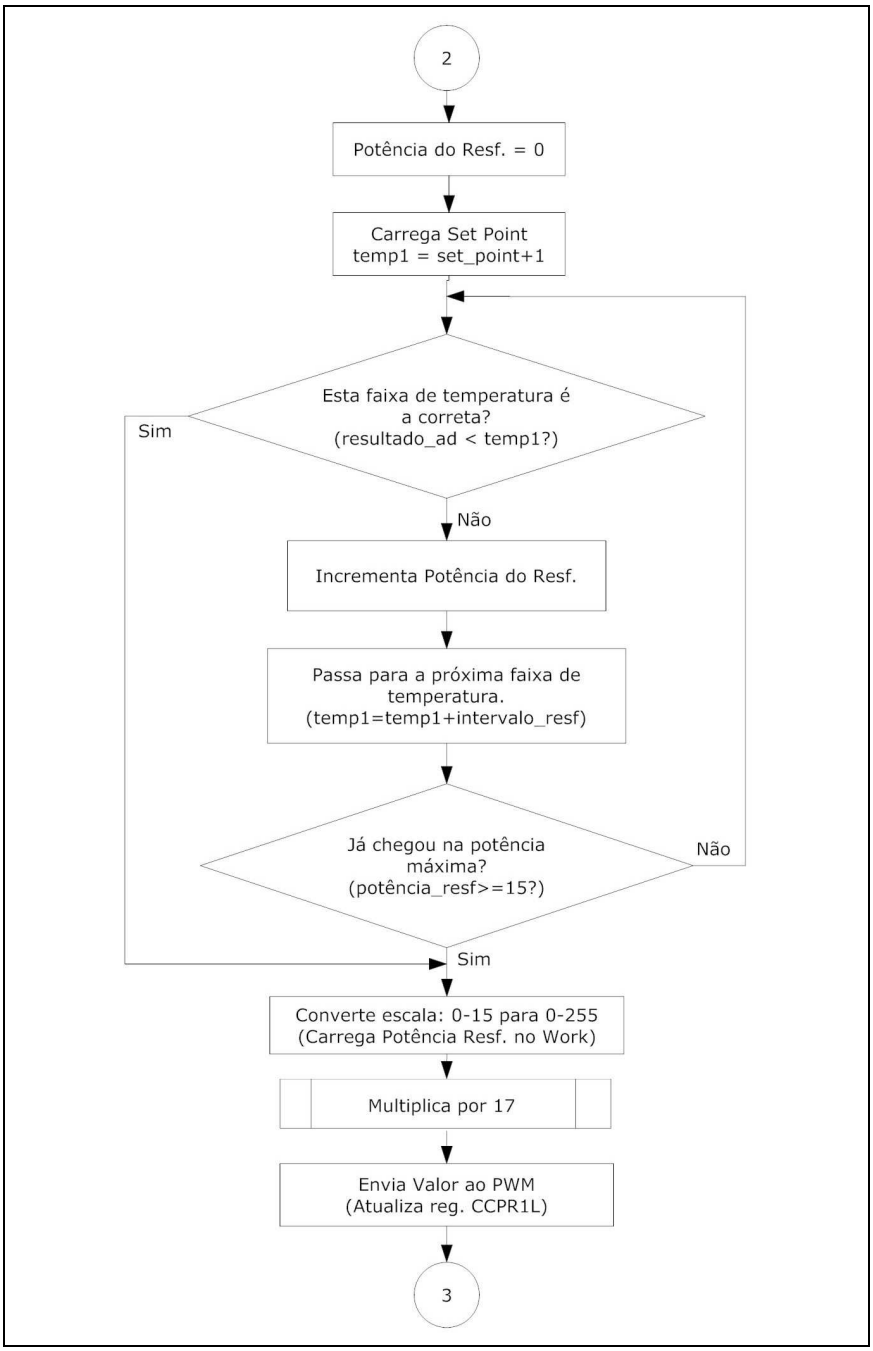


Figura 21. Fluxograma que representa um cálculo da potência

Neste diagrama de fluxo, a potência do resfriador é calculada com base em faixas de temperatura, que representam potência. São 15 faixas, sendo que o *loop*

principal, ao centro da Figura 21, é responsável por encontrar em qual delas a temperatura ambiente se encontra. Ao final desta função, a potência estará definida.

Inicialmente, as idéias do projeto foram rabiscadas em papel, através de reflexão sobre os temas pesquisados. Após, elas foram expressadas em fluxograma utilizando como base a estrutura dos exemplos presentes na bibliografia de Lavinia (2006) e as limitações do microcontrolador estudadas. Através das pesquisas, soube-se que não é possível realizar testes lógicos diretamente, bem como operações como multiplicação e divisão, através da linguagem *Assembly* do PIC. Portanto, o fluxograma foi feito abordando todas estas temáticas.

Foram comuns acertos e ajustes no fluxograma à medida que a programação em *Assembly* evoluía. Isto aconteceu devido a média complexidade na transcrição das fórmulas de controle utilizando comandos limitados da linguagem, como adição e subtração.

7.1.2 TRADUÇÃO PARA ASSEMBLY E TRANSFERÊNCIA PARA O CHIP

A partir do fluxograma e dos dados de programação pesquisados, como estrutura e comandos, foi possível a transcrição do programa. Os cabeçalhos de configuração presentes nas cinco primeiras linhas do Apêndice C foram extraídos sem modificações da bibliografia de Lavinia (2006), pois estes dados geralmente não mudam de projeto para projeto. Entre outras estruturas presentes no código fonte no Apêndice C, os procedimentos tornam-se importantes no sentido de organizar o código em blocos. Geralmente são formados de instruções que se repetem durante a execução do programa, como operações de divisão e multiplicação. Estes procedimentos equivalem a

funções em uma linguagem de alto nível. Os principais procedimentos estão relacionados na Tabela 5, bem como suas funções específicas.

Tabela 5. Principais procedimentos do código fonte

Procedimento	Registradores Envolvidos	Função Específica
delayy	tempo1, tempo0, work	Gasta (work*2) milisegundos.
multiplica	mult_x, mult_xr	Multiplica o conteúdo de mult_x por 17, guarda o resultaod em mult_xr.
divide	div_x, div_y, div_i, div_s	Divide o conteúdo de div_x pelo conteúdo de div_y, guarda a parte inteira do resultado em div_i e a sobra em div_s.
calcula_intervalo_resf	set_point, range_resf, intervalo_resf	Calcula quantas unidades AD equivalem 1 unidade de potência do resfriador. Guarda o cálculo em intervalo_resf.
calcula_intervalo_aque	set_point, range_aque, intervalo_aque	Calcula quantas unidades AD equivalem 1 unidade de potência do aquecedor. Guarda o calculo em intervalo_aque.
guarda_setpoint	set_point	Guarda o conteúdo de set_point na memória não-volátil.
recupera_setpoint	set_point	Recupera o valor da memória não-volátil e salva em set_point.
receber_dado	set_point	Lê o valor na porta serial e guarda em set_point. Chama os procedimentos calcula_intervalo_resf e calcula_intervalo_aque. Chama o procedimento guarda_setpoint.
enviar_dado	work	Envia work pela serial.
main	todas	Liga e desliga o LED de atividade. Instruções de cálculo de potência. Chama as funções receber_dado e enviar_dado.

Percebe-se, portanto, que ao traduzir os fluxogramas, deve-se manter a estrutura. Isto facilita a compreensão e manutenção do código, além de evitar códigos redundantes.

O software utilizado para programar foi o *MpLab*, produzido pelo mesmo fabricante do microcontrolador. Há embutido um compilador, que identifica erros de sintaxe e semântica antes de gerar o arquivo hexadecimal que é transferido para a memória não volátil do *chip.*

Em conjunto, utilizou-se um programador de baixo custo fabricado pela representante do fabricante no Brasil, LabTools. Este programador constitui-se de uma placa de circuito impresso com alguns componentes, que é responsável por ‘traduzir’ os sinais da porta serial do computador para a lógica TTL utilizada na gravação ‘in-circuit’

do *chip*. Então, basta acoplá-lo na placa, ligar o programador e o compilador identifica ambos. Logo, a transferência do programa já pode ser efetuada.

7.2 PROTÓTIPO ELETRÔNICO

Existem três etapas no qual é possível elaborar um protótipo físico. Deve-se elaborar um esquema eletrônico onde é demonstrado como todos os componentes eletrônicos se relacionam. Por meio dele é possível desenvolver uma placa de circuito impresso onde serão soldados. Por fim, há a etapa da solda dos componentes.

Para este projeto, foram executadas todas estas etapas. É possível pular algumas delas, mas é aconselhável que se faça todas para que se tenha registro e qualidade no protótipo final.

7.2.1 ESQUEMA ELETRÔNICO

O esquema eletrônico é importante na medida em que quem se interessar pela pesquisa, utilize este recurso para visualizar como os componentes eletrônicos e o microcontrolador estão se relacionando.

Com um software CAD específico, neste caso *ISIS 6*, foi possível montar o circuito esquemático com base nas pesquisas realizadas. As teorias foram juntadas, os esquemáticos oriundos de manuais e bibliografias foram acoplados e por fim obtiveram-se os diagramas presentes no Apêndice A.

7.2.2 LAYOUT DA PLACA DE CIRCUITO IMPRESSO

A placa de circuito impresso serve para organizar e ligar os componentes eletrônicos envolvidos no projeto. Gerar o layout significa transcrever o esquema eletrônico em um desenho no qual se possa gerar uma placa eletrônica física, em que é possível soldar os componentes eletrônicos.

Este desenho, ou seja, a disposição destes componentes, é planejada em um software CAD específico, neste caso, *Ivex WinBoard*. Todo o processo é manual e deve ser feito visando o lado dos componentes na placa, tal qual o esquema eletrônico. O objetivo principal é que os componentes e ligações sejam dispostos no menor espaço físico possível. Assim, a miniaturização do protótipo fortalece o título deste trabalho, que diz respeito à computação embarcada.

Ao final, o desenho deve ser impresso invertido em um dos eixos para que as ligações sejam desenhadas no lado da placa em que vai a solda. Este desenho foi dividido nas versões para cargas DC e AC, nos Apêndices D e E, respectivamente.

7.2.3 Placa de Circuito Impresso

Existem empresas especializadas na elaboração de placas de circuito impresso. Porém, para conduzir o processo de pesquisa, costuma-se desenvolver protótipos.

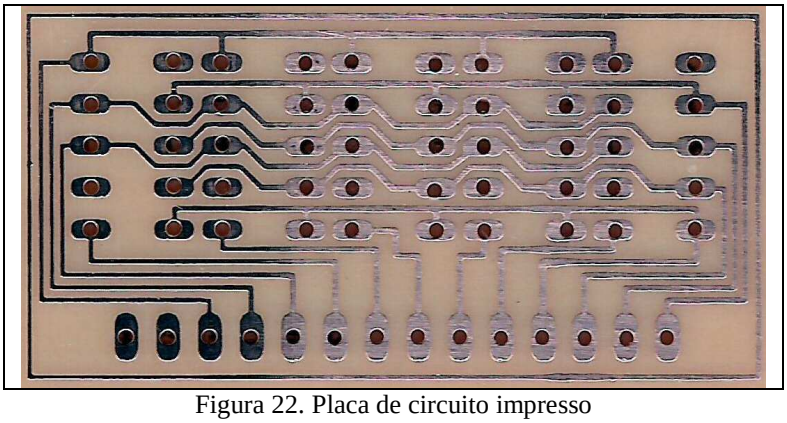


Figura 22. Placa de circuito impresso

Na Figura 22 encontra-se um exemplo de placa de circuito impresso. Fica evidente que a qualidade visual depende do modo com que o desenho feito no CAD é colocado na placa. Se o processo for manual, certamente o resultado será inferior ao exemplificado.

7.2.4 SOLDAGEM

Os instrumentos necessários para realizar esta atividade são ferro de solda e estanho em fio. O microprocessador foi montado em soquete para facilitar a manutenção e evitar que a temperatura da solda o danificasse.

7.3 SOFTWARE PARA AQUISIÇÃO DOS DADOS

Este projeto não possui nenhum artifício de visualização de dados, ou seja, não há nenhum *display* em que o usuário possa ter acesso aos dados obtidos. Por isto, foi gerada uma interface que é responsável por apresentar os dados de temperatura lidos pelo protótipo e também enviar o *set point* estabelecido pelo usuário. O objetivo desta não é controlar a temperatura, pois o software do protótipo já realiza esta função.

Na prática, o protótipo é conectado a um computador através da porta serial, sendo que há um cabo específico para isto. Então, os dados são apresentados em tempo real em forma de gráfico, a medida em que chegam do protótipo via porta serial, conforme demonstrado na Figura 23.

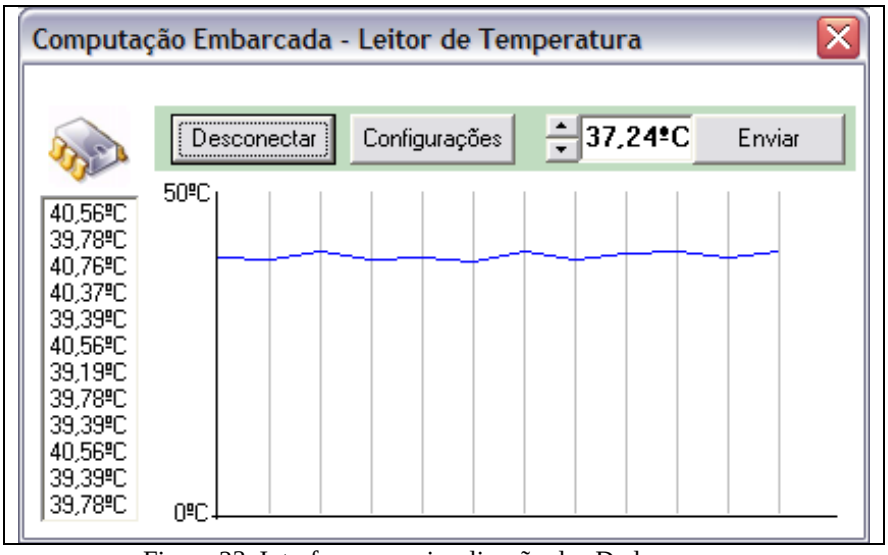


Figura 23. Interface para visualização dos Dados

Foi programado na suíte Borland Delphi 7 e feito para o sistema operacional Windows XP. Esta escolha foi feita pela agilidade na programação e por existir um componente pronto (*ComPort*) e próprio para realizar a comunicação serial, mas o conceito pode ser aplicado em qualquer linguagem de programação. No Apêndice F encontra-se o diagrama de fluxo referente a esta interface.

CONCLUSÃO

Com o desenvolvimento tecnológico, cada vez mais são possíveis sistemas de controle efetivos para temperatura em ambientes industriais. A tecnologia dos microcontroladores possui recursos amplos e direcionados especificamente para este função, o que estimula seu uso nestes ambientes e diminui o custo total de projetos e produtos, pois a necessidade de computadores diminui e em alguns casos, se extingue.

No projeto desenvolvido de um sistema de controle de temperatura para avicultura, as bibliografias apontaram que com o microcontrolador *PIC* é possível obter uma resolução de aquisição de até 10 *bits*, ou seja, uma precisão de meio décimo de grau, o que valida seu uso tanto em fornos industriais quanto em aplicações críticas de laboratório.

Foram desenvolvidos hardware e software, em um protótipo de funcionamento independente, no que diz respeito à função controle. No hardware, componentes eletrônicos fazem aquisição da temperatura através de sensores, estabelecem a comunicação com um computador tipo PC – se for desejado e controlam os atuadores. No software, a lógica de controle linear foi implementada em linguagem *Assembly*, incluindo técnicas de conversão *AD*, modulação *PWM* e comunicação serial para comunicação de dados.

Foi desenvolvida também uma interface para computadores com sistema operacional *Microsoft Windows XP*, no qual dados recebidos do protótipo são apresentados em forma de gráfico.

Dentre as dificuldades encontradas para o desenvolvimento do protótipo cita-se a escassa bibliografia. Esta dificuldade foi contornada por meio de orientação, contatos com profissionais da área, estudo das folhas de dados (DataSheets) dos

componentes envolvidos. Outro fator crítico é que o microcontrolador utilizado PIC trabalha com 8 *bits*, não sendo possível a expansão do código de controle para atingir uma resolução de 10 *bits*. Uma dificuldade prática apresentada é que o oscilador simplificado, por vezes, tornava-se não confiável, resultando em um sistema por vezes instável no que diz respeito a aquisição da temperatura. Este contratempo foi contornado pela utilização de filtros capacitivos.

Há alguns hardwares disponíveis especificamente para desenvolvimento e teste, dispensando o uso de *protoboards* e remoção do microcontrolador para programação. Entretanto, por questões de custo, não foi adquirido este tipo de equipamento. Isto ocasionou a danificação de alguns *chips* por eletricidade estática. Cuidados adicionais devem ser tomados para prevenir estes tipos de inconvenientes. É recomendável, nestes casos, o uso de pulseira anti-estática.

Todos os objetivos foram alcançados com sucesso ao final desta pesquisa, inclusive com a produção de um protótipo em hardware e software. Como sugestão para trabalhos futuros e continuidade dessa pesquisa sugere-se:

a) adicionar canais de controle;

b) substituir a função PWM do *chip* pela implementação de uma específica com detecção de passagem por zero da rede elétrica, para controle de cargas para corrente alternada;

c) adicionar funcionalidades à interface para armazenar os valores lidos (histórico);

d) ampliar o circuito eletrônico afim de proporcionar uma opção de gravação 'in circuit';

e) estudar e proporcionar alternativas para sensores de temperatura;

f) aumentar a faixa de trabalho de temperatura, ampliando a gama de atuação, visando fornos industriais;

g) tornar a resolução do conversor AD mais refinada;

h) ampliar o circuito com um microcontrolador trabalhando como servidor de páginas HTML, para monitoramento a distância via WEB;

i) implementar tolerância a falha.

REFERÊNCIAS

BRAGA, Newton C.. **Coleção de relés de estado sólido**. Saber Eletrônica, Taubaté, n. 412, p.32, maio 2007. Disponível em: <http://www.sabereletronica.com.br/secoes/leitura/367>. Acesso em: 11 jan. 2008.

BRUSSAMARELO, Valner. CRESPI, Roger, et al. **Sensor de Temperatura LM35**. UCS, Caxias do Sul: 2006.

CHANDLER, Ariz. **Power-Meter Maker Linyang Takes Delivery of Five-Billionth PIC Microcontroller**. Microchip Press News, 2007. <http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2018&mcparam=en028627>Acesso em 20.nov.2007.

DECUYPERE, Eddy, et al. **Manejo da Incubação** (Biologia das Aves – Fisiologia do Embrião). 537p. Facta: Campinas, 2003.

DIAS, Paulo Fernando, MÜLLER, Yara Maria Rauh. **Características do desenvolvimento embrionário de Gallus Gallus Domesticus, em temperaturas e períodos diferentes de incubação**. Departamento de Biologia Celular , Embriologia e genética do Centro de Ciências Biomédicas da UFSC. Florianópolis, SC. 1998. SciELO <http://www.scielo.br/pdf/bjvras/v35n5/35n5a10.pdf> Acesso em 20.nov.2007.

GÜÉMEZ, Julio; FIOLHAIS, Carlos; FIOLHAIS, **Manuel. Fundamentos de Termodinâmica do Equilíbrio**. Lisboa: Fundação Calouste Gulbernkian, 1998.

JORDAN, Rodrigo A., TAVARES, Maria H. F. **Análise de diferentes sistemas de iluminação para aviários de produção de ovos férteis**. UNICAMP, Campinas, SP. 2005. SciELO <http://www.scielo.br/pdf/rbeaa/v9n3/v9n3a19.pdf> Acesso em 20.nov.2007.

JUNG, Cláudio Rosito et al. **Computação Embarcada**: Projeto e Implementação de Veículos Autônomos Inteligentes. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, Não use números Romanos ou letras, use somente números Arábicos., 2005, São Leopoldo. Anais do XXV Congresso da Sociedade Brasileira de Computação. São Leopoldo: Sbc, 2005. v. 1, p. 1358 - 1406. Disponível em: <http://www.sbc.org.br/bibliotecadigital/download.php?paper=138>. Acesso em: 06 jul. 2008.

KAUFMAN, Milton; WILSON, J.A.. **Eletrônica Básica**: teoria e prática. V.1.São Paulo: Rideel Ltda. 1984.

LAVINIA, Nicolás César, SOUZA, David José de. **Conectando o PIC** - Recursos Avançados. 3 ed. 384p. Érica: São Paulo, 2006. ISBN 85-7194-7376.

MARINONI, Mauro, et al. **Six-Legged Robot**: Real-Time Issues and Architecture. Proceedings of the 5th IFAC International Symposium on Intelligent Components and Instruments for Control Applications (SICICA, 2003), Aveiro, Portugal. 245-250, 2003.

MATIC, Nebojsa, ANDRIC, Dragan. **The Pic Microcontroller** - Book 1. Trad. Alberto Jerônimo. 252 p.. mikroElektronika: Belgrade, 2000.

MICROCHIP TECHNOLOGY INC. **PIC16F87X**: Flash-Based 8-Bit CMOS Microcontrollers. Chandler, Arizona, 2005. Disponível em: <http://ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf> Acesso em 13.nov.2006.

NATALE, Ferdinando. **Automação Industrial**. São Paulo: Érica, 2000.

NATIONAL SEMICONDUCTOR. **LM35:** Precision Centigrade Temperature Sensors. Santa Clara, Califórnia, 2000. Disponível em: < www.national.com/ds/LM/LM35.pdf > Acesso em 13.nov.2006.

NATIONAL SEMICONDUCTOR. **LM741: Operational Amplifier.** Santa Clara, Califórnia, 2004. Disponível em: <http://cache.national.com/ds/LM/LM741.pdf> Acesso em 15.maio.2008.

SALVARO, Everson Bez Birolo, ROCHO, Wagner da Silva. **Protótipo para Monitoramento de Temperatura do Processo de Incubação**. Universidade do Sul de Santa Catarina. Departamento de Sistemas de Informação. ARARANGUÁ, 2006.

ROSA, Paulo Sérgio, GUIDONI, Antônio Lourenço. **Influência da temperatura de incubação em ovos de matrizes de corte com diferentes idades e classificados por peso sobre os resultados de incubação**. Revista Brasileira de Zootecnia, v.31 n.2 supl. Viçosa. abr. 2002. SciELO http://www.scielo.br/pdf/rbz/v31n2s0/21291.pdf> Acesso em 20.nov.2007.

SANCHEZ, Julio. **Microcontroller Programming**: The Microchip PIC. 799 p. CRC Press: Northwest: 2006. ISBN: 978-0-8493-7189-9.

SILVA, Ricardo Pereira e. **Eletrônica básica**: um enfoque voltado à informática. Florianópolis: UFSC, 1995. ISBN 8532800211.

SMITH, Tom W.. Care and Incubation of Hatching Eggs. Mississippi State University. Disponível em: <http://www.msstate.edu/dept/poultry/hatch.htm>. Acesso em: 15 maio 2008.

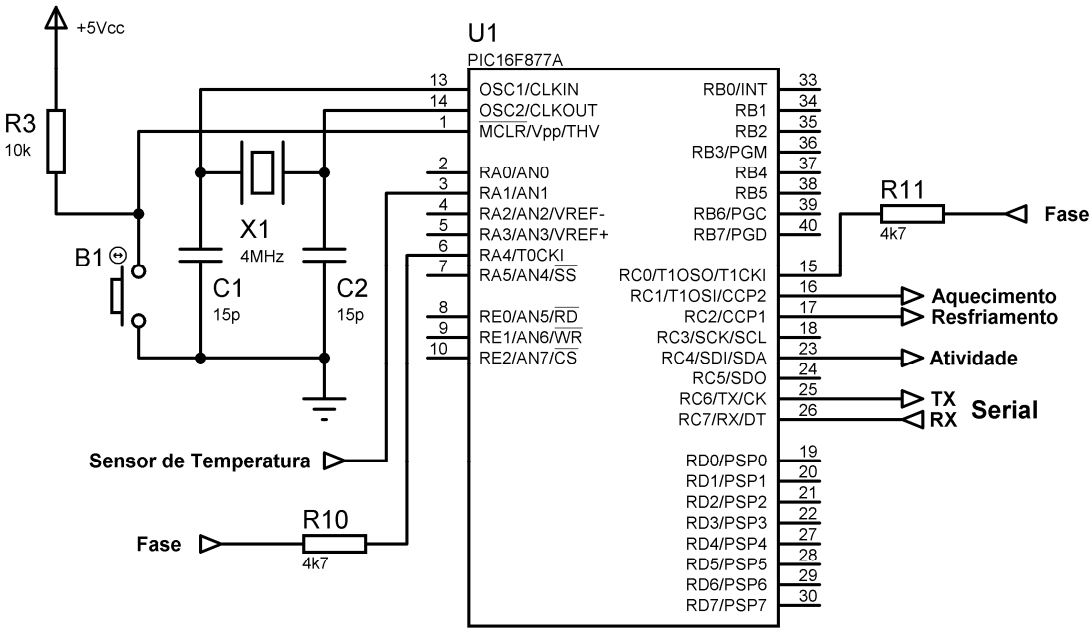
SOUZA, David José de. **Desbravando o PIC** - Ampliado e Atualizado para PIC 16F628A. 10 ed. 272p. Érica: São Paulo, 2006. ISBN 85-7194-8674.

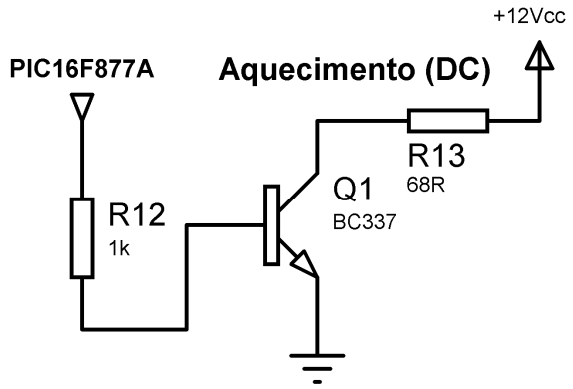
TAUB, Herbert. **Digital Integrated Electronics**. Trad. Paulo Elyot Meirelles Villela. São Paulo: McGraw-Hill, 1982.

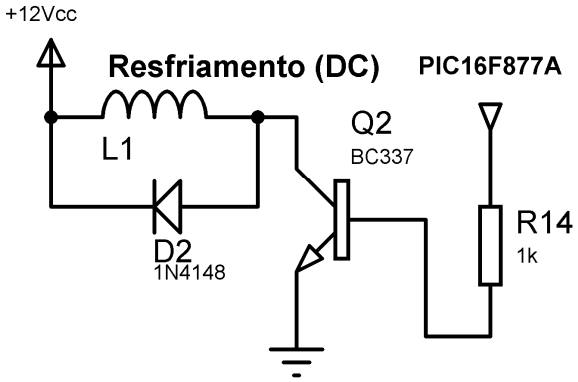
TEXAS INSTRUMENTS. **MAX232, MAX232I:** DUAL EIA-232 DRIVERS/RECEIVERS. Dallas, Texas, 2002. Disponível em: <http://www.datasheetcatalog.org/datasheet/texasinstruments/max232.pdf> Acesso em 25.mai.2008.

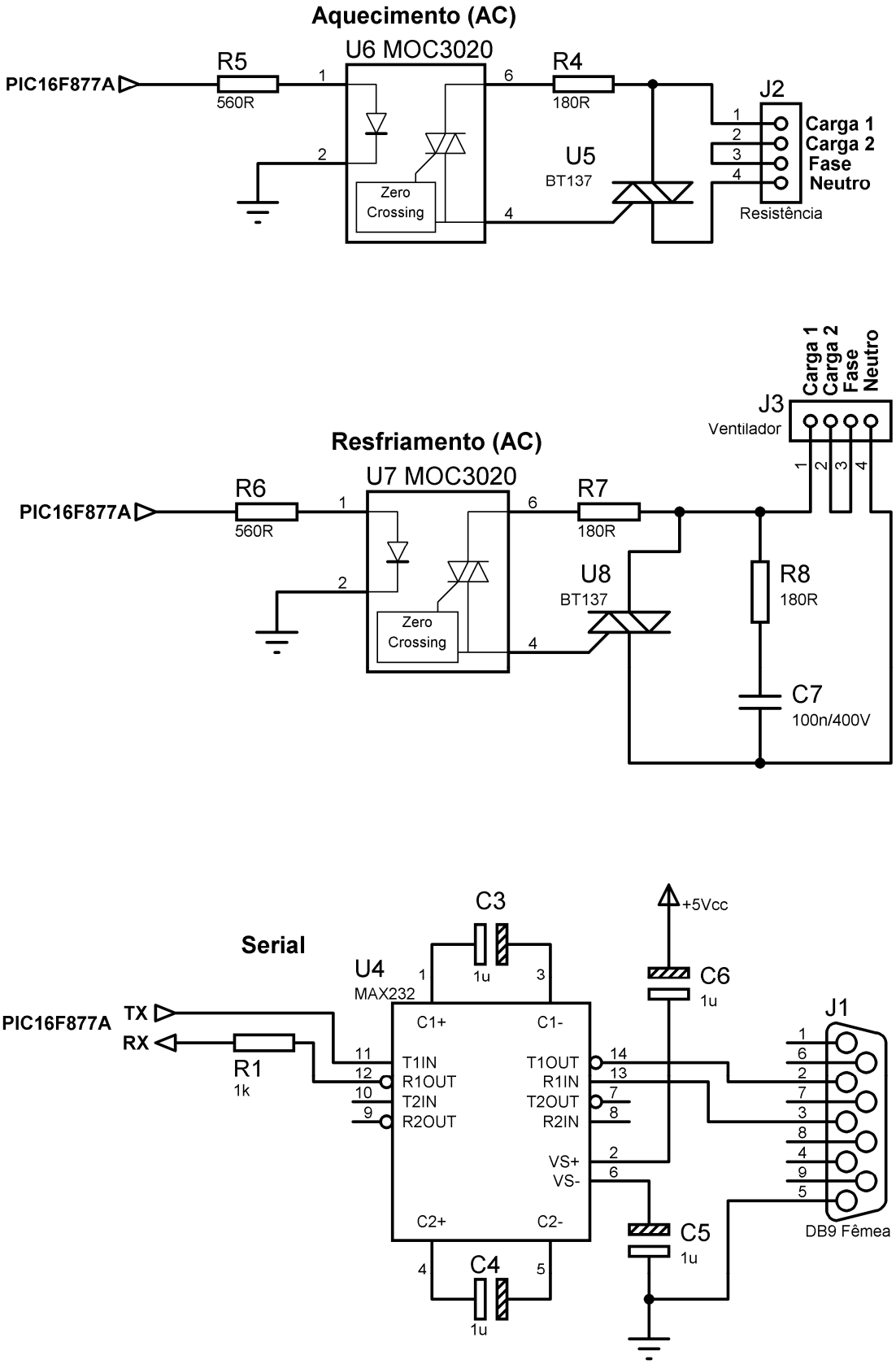
VAGLICA, John, GILMOUR, Peter. How to select a microcontroller. IEEE Spectrum, p.106-109. Nov. 1990.

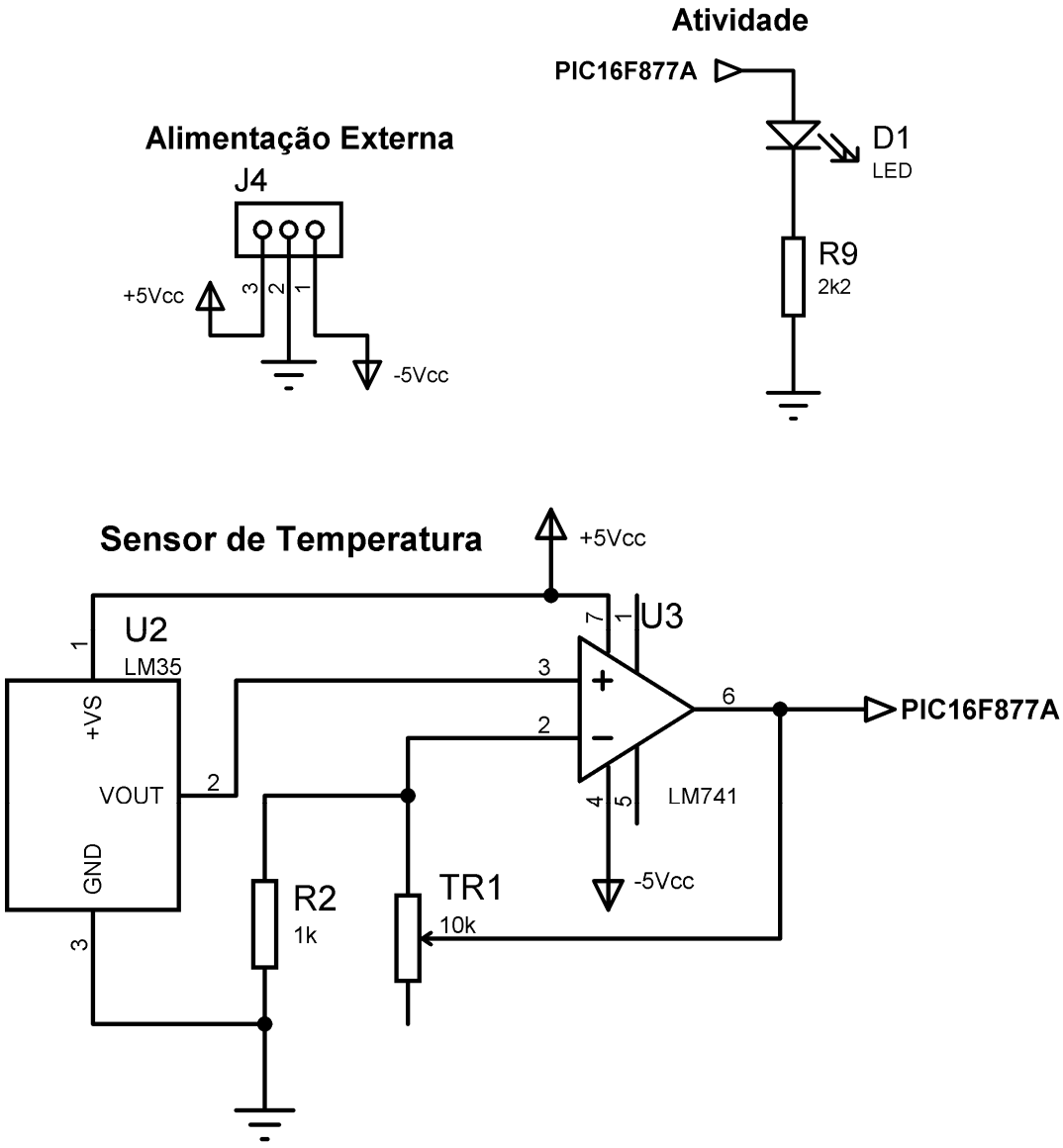
APÊNDICE A – ESQUEMA ELETRÔNICO DO PROTÓTIPO



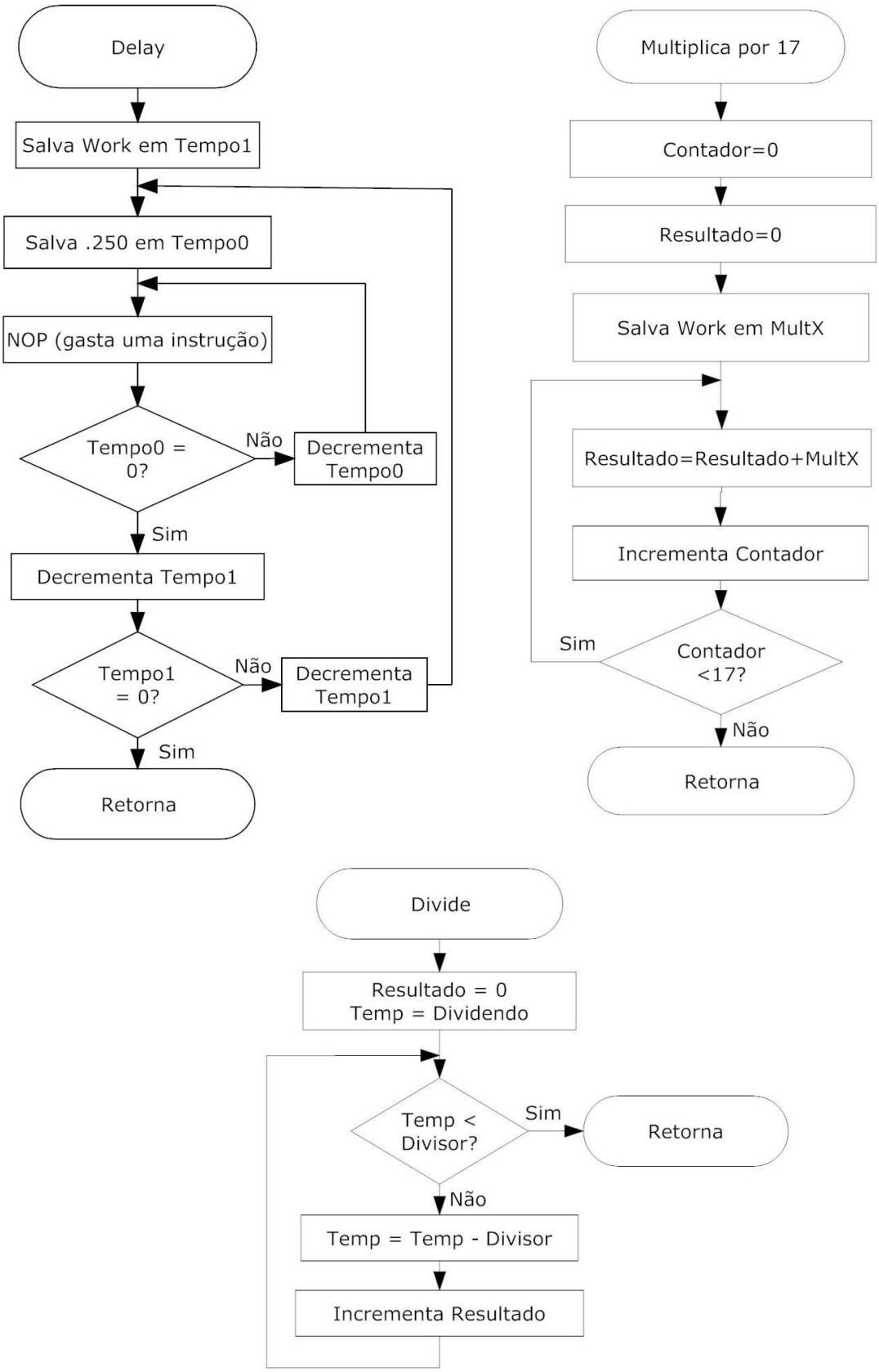


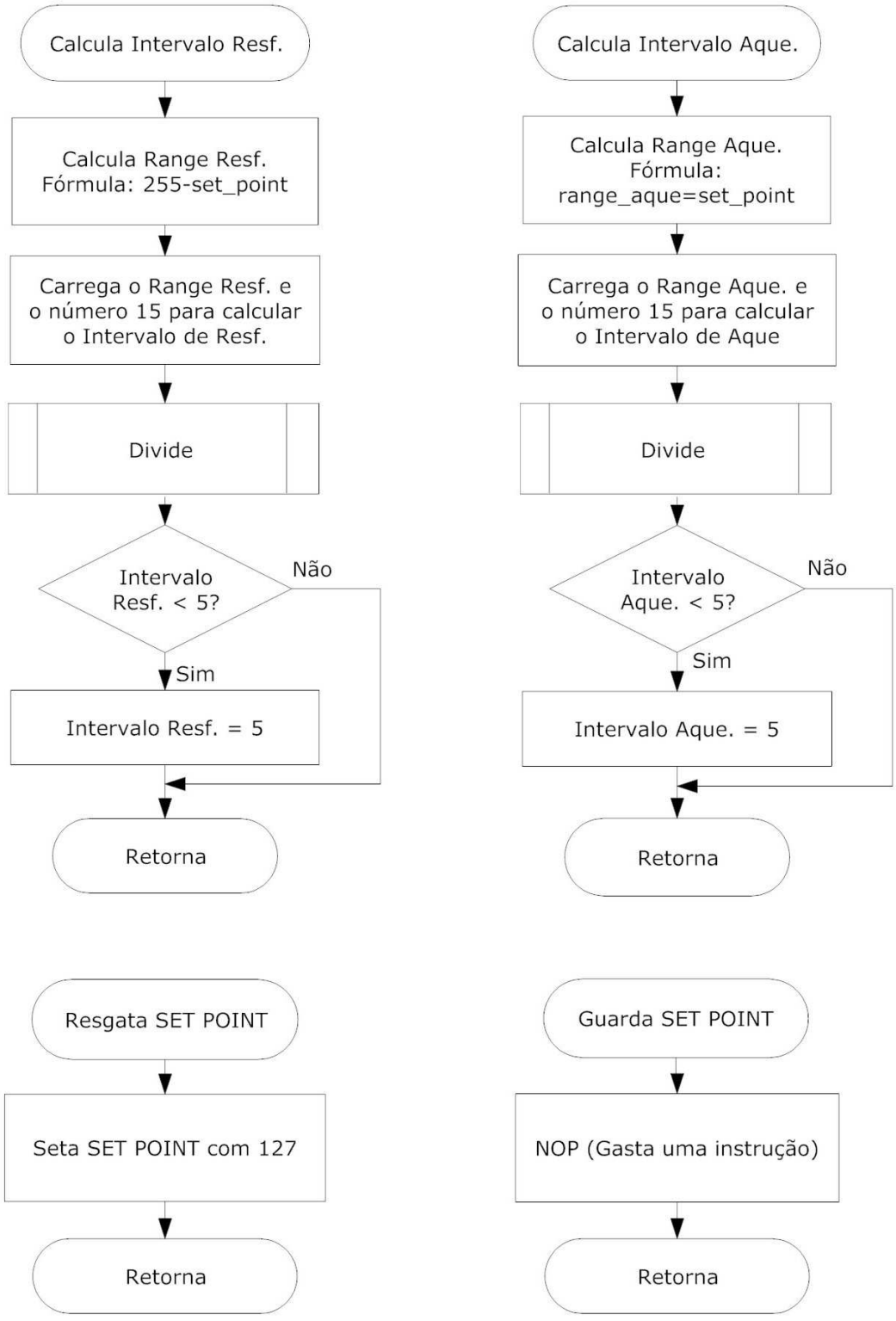


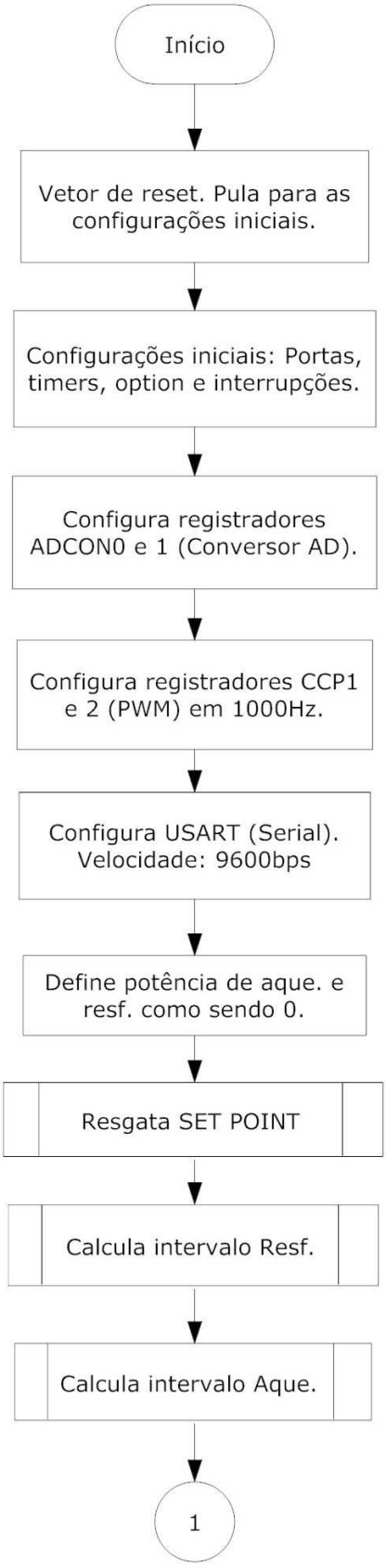

Aquecimento (AC)
PIC16F877A
R5
560R
U6 MOC3020
Zero Crossing
R4
180R
U5
BT137
J2
Carga 1
Carga 2
Fase
Neutro
Resistência
Resfriamento (AC)
PIC16F877A
R6
560R
U7 MOC3020
Zero Crossing
R7
180R
U8
BT137
R8
180R
C7
100n/400V
J3
Ventilador
Carga 1
Carga 2
Fase
Neutro
Serial
PIC16F877A
TX
RX
R1
1k
U4
MAX232
C1+
C1-
T1IN
R1OUT
T2IN
R2OUT
T1OUT
R1IN
T2OUT
R2IN
VS+
VS-
C2+
C2-
C3
1u
C4
1u
C5
1u
C6
1u
+5Vcc
J1
DB9 Fêmea


Atividade
PIC16F877A
D1
LED
R9
2k2
Alimentação Externa
J4
+5Vcc
-5Vcc
Sensor de Temperatura
+5Vcc
U2
LM35
+VS
VOUT
GND
U3
LM741
PIC16F877A
R2
1k
TR1
10k
-5Vcc

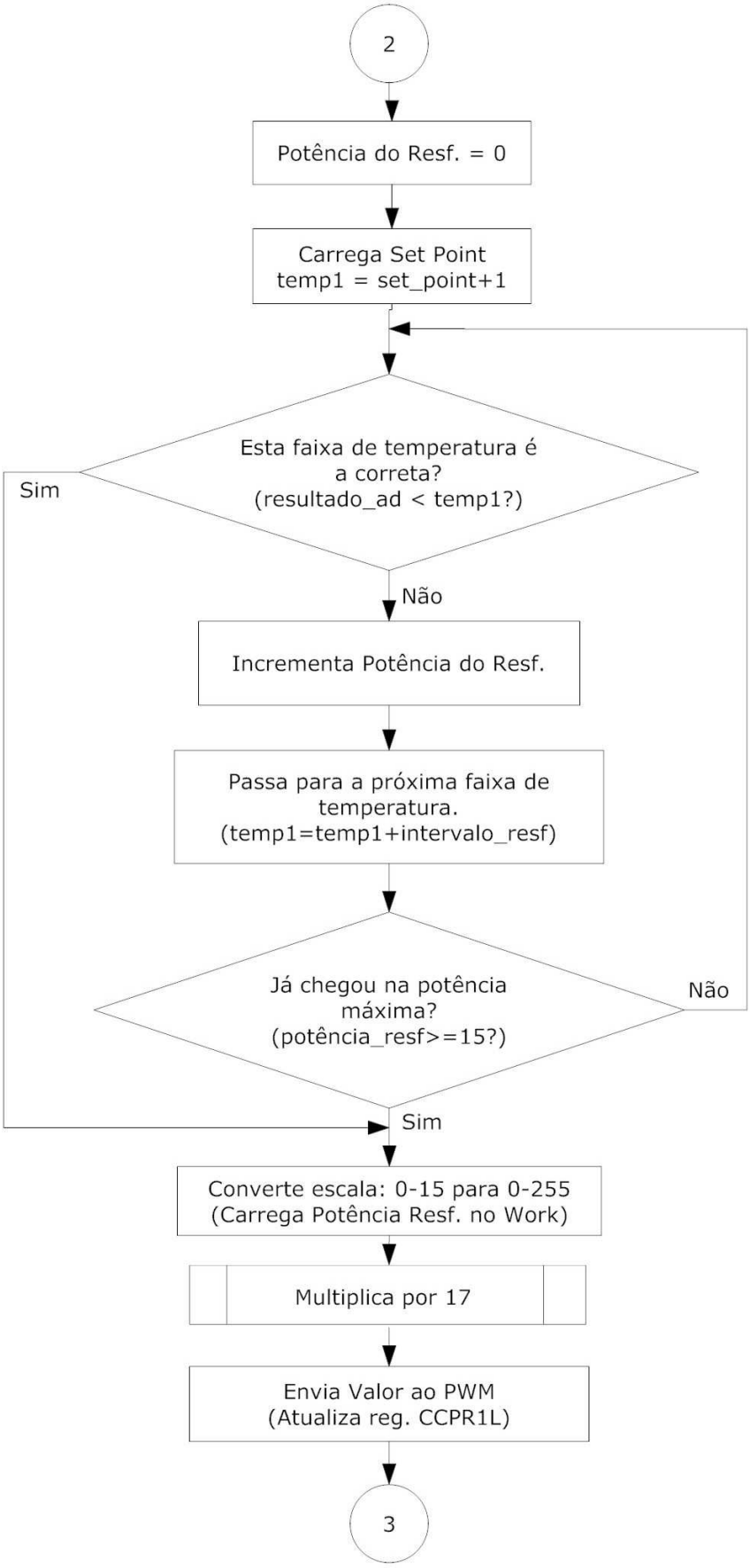
APÊNDICE B – FLUXOGRAMA DO PROGRAMA

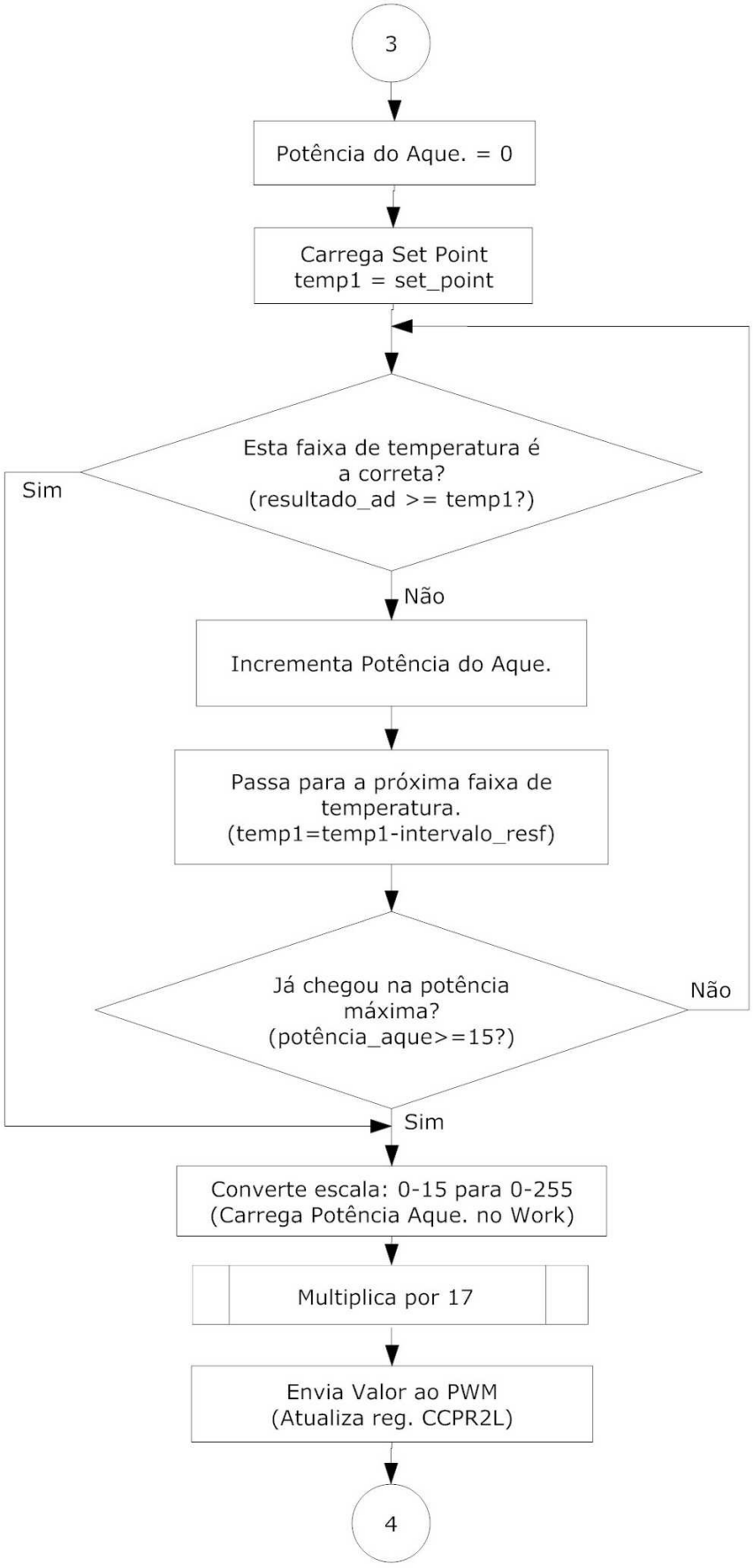


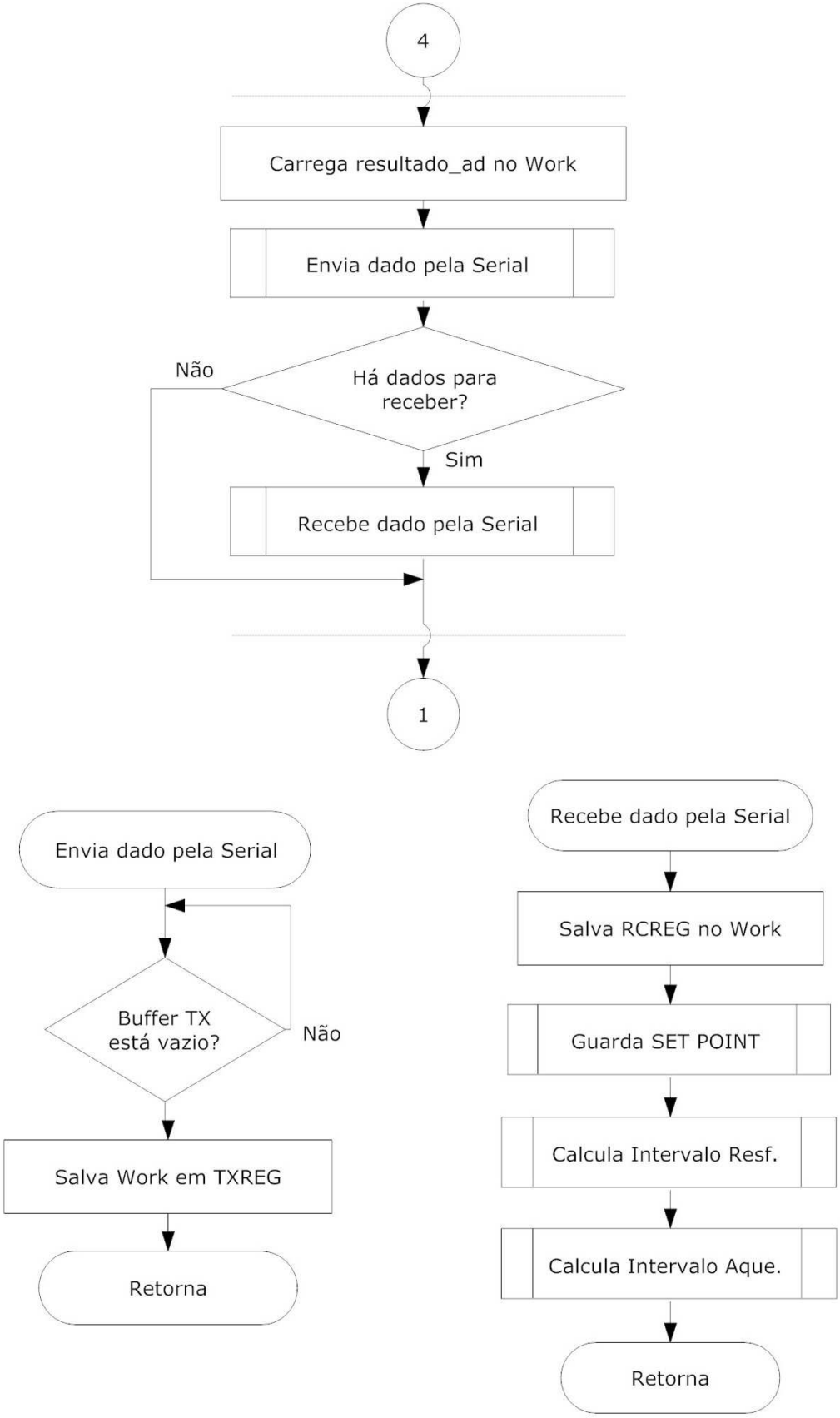

Calcula Intervalo Resf.
Calcula Range Resf.
Fórmula: 255-set_point
Carrega o Range Resf. e
o número 15 para calcular
o Intervalo de Resf.
Divide
Intervalo
Resf. < 5?
Não
Sim
Intervalo Resf. = 5
Retorna
Calcula Intervalo Aque.
Calcula Range Aque.
Fórmula:
range_aque=set_point
Carrega o Range Aque. e
o número 15 para calcular
o Intervalo de Aque
Divide
Intervalo
Aque. < 5?
Não
Sim
Intervalo Aque. = 5
Retorna
Resgata SET POINT
Seta SET POINT com 127
Retorna
Guarda SET POINT
NOP (Gasta uma instrução)
Retorna


Início
Vetor de reset. Pula para as configurações iniciais.
Configurações iniciais: Portas, timers, option e interrupções.
Configura registradores ADCON0 e 1 (Conversor AD).
Configura registradores CCP1 e 2 (PWM) em 1000Hz.
Configura USART (Serial). Velocidade: 9600bps
Define potência de aque. e resf. como sendo 0.
Resgata SET POINT
Calcula intervalo Resf.
Calcula intervalo Aque.
1


1
Inicia Conversão AD
(Seta ADCON0,GO)
Terminou
Conversão AD?
Não
Salva Valor em
resultado_ad
Liga o LED de Atividade
Carrega .250 no Work
Delay
Desliga o LED de Atividade
Carrega .250 no Work
Delay
2


2
Potência do Resf. = 0
Carrega Set Point
temp1 = set_point+1
Esta faixa de temperatura é
a correta?
(resultado_ad < temp1?)
Sim
Não
Incrementa Potência do Resf.
Passa para a próxima faixa de
temperatura.
(temp1=temp1+intervalo_resf)
Já chegou na potência
máxima?
(potência_resf>=15?)
Não
Sim
Converte escala: 0-15 para 0-255
(Carrega Potência Resf. no Work)
Multiplica por 17
Envia Valor ao PWM
(Atualiza reg. CCPR1L)
3


3
Potência do Aque. = 0
Carrega Set Point
temp1 = set_point
Esta faixa de temperatura é
a correta?
(resultado_ad >= temp1?)
Sim
Não
Incrementa Potência do Aque.
Passa para a próxima faixa de
temperatura.
(temp1=temp1-intervalo_resf)
Já chegou na potência
máxima?
(potência_aque>=15?)
Não
Sim
Converte escala: 0-15 para 0-255
(Carrega Potência Aque. no Work)
Multiplica por 17
Envia Valor ao PWM
(Atualiza reg. CCPR2L)
4


4
Carrega resultado_ad no Work
Envia dado pela Serial
Não
Há dados para receber?
Sim
Recebe dado pela Serial
1
Envia dado pela Serial
Buffer TX está vazio?
Não
Salva Work em TXREG
Retorna
Recebe dado pela Serial
Salva RCREG no Work
Guarda SET POINT
Calcula Intervalo Resf.
Calcula Intervalo Aque.
Retorna

APÊNDICE C – PROGRAMA ESCRITO EM *ASSEMBLY*

```
#include <P16F877A.INC>      ; Arquivo de include do uC usado, no caso PIC16F877A.


__CONFIG _CP_OFF & _CPD_OFF & _DEBUG_OFF & _LVP_OFF & _WRT_OFF & _BODEN_OFF
& _PWRTE_OFF & _WDT_OFF & _XT_OSC

#DEFINE BANK0 BCF STATUS,RP0      ;SETA BANK 0 DE MEMÓRIA
#DEFINE BANK1 BSF STATUS,RP0      ;SETA BANK 1 DE MAMÓRIA

;-----------------------------------------------------------------
;-------------------------VARIAVEIS LOCAIS
;-----------------------------------------------------------------

        cblock 0x20
        tempo0              ;Variaveis rotina de delay
        tempo1

        resultado_ad        ;Guarda o resultado da conversao AD, ou seja,
                            ; a temperatura lida pelo sensor
        set_point           ;Fica guardado o limite de temperatura imposto pelo usuário

        range_resf
        intervalo_resf
        potencia_resf       ;Guarda a potência calculada de resfriamento

        range_aque
        intervalo_aque
        potencia_aque       ;Guarda a potência calculada de aquecimento

                             ;Variáveis da rotina de divisão
        DIV_X               ;dividendo
        DIV_Y               ;divisor
        DIV_I               ;parte inteira
        DIV_S               ;sobra da divisao

                            ;Variáveis da rotina multiplicação
        MULT_X              ;multiplicador
        MULT_XR             ;resultado
        mult_temp

        temp1               ;Variáveis de comparação para testes lógicos
        temp2

        endc


;-----------------------------------------------------------------
;-------------------------DEFINES DAS SAÍDAS
; Neste lugar as saídas são identificadas para facilitar o
; entendimento do programa por parte do programador. Como não é
; obrigatório fazê-lo, há apenas uma saída para constar.
;-----------------------------------------------------------------
#DEFINE        LED     PORTC,4              ;PORTA DO LED
;-----------------------------------------------------------------


;-----------------------------------------------------------------
;-------------------------VETOR DE RESET DO MICROCONTROLADOR
```

```
; Este é o primeiro endereço de memória que o chip lê quando
; é energizado.
;-----------------------------------------------------------
        org 0x00                                ; Vetor de reset do uC.
        goto CONFIGURAR
;-----------------------------------------------------------


;-----------------------------------------------------------
;--------------------------VETOR DE INTERRUPÇÃO
; Sempre que uma interrupção acontece, o microcontrolador aponta
; para este endereço de memória e executa os códigos até que
; encontre a linha RETFIE.
;-----------------------------------------------------------
        ORG    0x04             ;ENDEREÇO INICIAL DA INTERRUPÇÃO
        RETFIE                  ;RETORNA DA INTERRUPÇÃO
;-----------------------------------------------------------



;*************************************************************
;--------------------------ROTINAS
;-----------------------------------------------------------

;-----------------------------------------------------------
;--------------------------Rotina de delay 2ms
;-----------------------------------------------------------
; Esta rotina gasta 2 ms.
; Uso: move-se um inteiro para o work e chama-se esta função.
;      Por exemplo, se deseja-se gastar 500ms:
;      movlw .250
;      call DELAYY
;-----------------------------------------------------------
DELAYY
        movwf  tempo1
        movlw  .250
        movwf  tempo0
        clrwdt
        decfsz  tempo0,f
        goto    $-2
        decfsz  tempo1,f
        goto    $-6
        return
;-----------------------------------------------------------


;-----------------------------------------------------------
;--------------------------Rotina de Multiplicação
;-----------------------------------------------------------
; Esta rotina multiplica um número por 17. Isto é útil na
; conversão da potência calculada para o valor em PWM.
; Uso: Move-se o valor a ser multiplicado para o reg. MULT_X,
;      chama-se a função. O resultado é guardado em MULT_XR.
;      Por exemplo, se deseja multiplicar 5 por 17:
;      movlf .5
;      movwf MULT_X
;      CALL MULTIPLICA
;      movfw MULT_XR
;-----------------------------------------------------------
MULTIPLICA
        movlw .0
        movwf mult_temp
        movwf MULT_XR
```

```
MULT_LOOP:
        movfw MULT_X
        addwf MULT_XR,MULT_XR

        incf mult_temp
        movlw .17                       ;faz 17 vezes
        subwf mult_temp,w
        btfss STATUS,C
        goto MULT_LOOP

        return
;-----------------------------------------------------------------


;-----------------------------------------------------------------
;--------------------------Rotina de Divisão
;-----------------------------------------------------------------
; Esta rotina divide um valor pelo outro. Útil no cálculo dos
; 'ranges' de atuação do resfriador e aquecedor.
; Uso: Move-se o dividendo para DIV_X e o divisor para DIV_Y.
;      A parte inteira do resultado é guardada em DIV_I, e a
;      sobra da divisão fica em DIV_S.
;      Por exemplo, se deseja fazer a operação (10/2)
;      movlw .10
;      movwf DIV_X
;      movlw .2
;      movwf DIV_Y
;      call DIVIDE
;-----------------------------------------------------------------
DIVIDE
        movlw .0
        movwf DIV_I                     ;zera parte inteira
        movfw DIV_X
        movwf DIV_S                     ;move o dividendo para a sobra

DIV_LOOP:
        movfw DIV_Y                     ;carrega o work com o valor do divisor
        subwf DIV_S,w                   ; w=DIV_R-DIV_Y
        btfss STATUS,C
        goto DIV_FIM                    ; x<y
        movwf DIV_S                     ; x>=y
        incf DIV_I                      ;incrementa a parte inteira
        goto DIV_LOOP

DIV_FIM:
        return
;-----------------------------------------------------------------


;-----------------------------------------------------------------
;--------------------------Rotina que calcula o intervalo do
;--------------------------resfriador
;-----------------------------------------------------------------
; Esta rotina serve para calcular qual o tamanho de passos do
; resfriador, ou seja, de quantos em quantos graus a potência irá
; mudar.
; Uso: com o set_point estabelecido, basta chamar a rotina.
;      Por exemplo:
;      CALL CALCULA_INTERVALO_RESF
;-----------------------------------------------------------------
CALCULA_INTERVALO_RESF

;Calcula range
        movfw set_point
```

```
        sublw .255
        movwf range_resf


;Calcula intervalo
        movfw range_resf                ; range é o dividendo
        movwf DIV_X
        movlw .15                       ; 15 é o divisor
        movwf DIV_Y
        call DIVIDE                     ;faz DIV_X / DIV_Y
        movfw DIV_I
        movwf intervalo_resf            ;coloca a parte inteira da divisao em intervalo

;Limita intervalo em .5
        movlw .5
        subwf intervalo_resf,w          ; w=intervalo-y
        btfss STATUS,C
        goto $+2                        ; intervalo<5
        goto $+3                        ; intervalo>=5
        movlw .5
        movwf intervalo_resf

        return
;-----------------------------------------------------------------


;-----------------------------------------------------------------
;-------------------------Rotina que calcula o intervalo do
;-------------------------aquecedor
;-----------------------------------------------------------------
; Esta rotina serve basicamente para o mesmo propósito da feita
; para o resfriador.
; Uso: com o set_point estabelecido, basta chamar a rotina.
;      Por exemplo:
;      CALL CALCULA_INTERVALO_AQUE
;-----------------------------------------------------------------
CALCULA_INTERVALO_AQUE

;Calcula range
        movfw set_point
        movwf range_aque

;Calcula intervalo
        movfw range_aque        ; range é o dividendo
        movwf DIV_X
        movlw .15               ; 15 é o divisor
        movwf DIV_Y
        call DIVIDE             ; faz DIV_X / DIV_Y
        movfw DIV_I
        movwf intervalo_aque    ; coloca a parte inteira da divisao em intervalo

; Limita intervalo em .5
        movlw .5
        subwf intervalo_aque,w  ; w=intervalo-y
        btfss STATUS,C
        goto $+2                ; intervalo<5
        goto $+3                ; intervalo>=5
        movlw .5
        movwf intervalo_aque

        return
;-----------------------------------------------------------------
```

```
;-------------------------------------------------------------
;-MEMORIA-----------------Rotina que guarda set_point na memória
;-------------------------não volátil
;-------------------------------------------------------------
; Esta rotina serve para guardar o valor do set_point na memória
; não volátil, ou seja, que não é limpa com a desernegização do
; chip. Esta rotina ainda não foi implementada, porém consta para
; facilitar uma futura implementação.
; Uso: com o set_point estabelecido, basta chamar a função.
;      movlw .200
;      movwf set_point
;      CALL GUARDA_SETPOINT
;-------------------------------------------------------------
GUARDA_SETPOINT
        nop
;imprementação futura
        return
;-------------------------------------------------------------


;-------------------------------------------------------------
;-MEMORIA-----------------------Rotina que recupera set_point da
;--------------------------------memória não volátil
;-------------------------------------------------------------
; Esta rotina recupera o set_point da memória não volátil. Esta
; rotina também não foi implementada, mas consta para facilitar
; uma futura implementação.
; Uso: no início do programa, basta chamar esta função:
;      CALL RECUPERA_SETPOINT
;-------------------------------------------------------------
RECUPERA_SETPOINT
        movlw .127
        movwf set_point
;imprementação futura
        return
;-------------------------------------------------------------


;-------------------------------------------------------------
;-SERIAL--------------------Rotina que carrega valor recebido na
;-------------------------variável set_point
;-------------------------------------------------------------
; Esta rotina serve para regatar o valor que se encontra no
; barramento serial do chip e colocá-lo na variável set_point.
; Uso: depois de testar se há algum valor na serial, chamar esta
;      rotina. Por exemplo:
;           BTFSC PIR1,RCIF
;           CALL RECEBER_DADO
;-------------------------------------------------------------
RECEBER_DADO
        MOVF   RCREG,W ; CARREGA DADO RECEBIDO NO WORK
        MOVWFset_point                  ; guarda valor na ram
        call GUARDA_SETPOINT            ; guarda valor na memoria nao volatil
        call CALCULA_INTERVALO_RESF
        call CALCULA_INTERVALO_AQUE
        return
;-------------------------------------------------------------


;-------------------------------------------------------------
;-SERIAL--------------------Rotina que envia o work pela serial
;-------------------------------------------------------------
; Esta rotina serve para enviar o valor de work pela serial.
; Uso: mover o valor para work e depois chamar esta função.
```

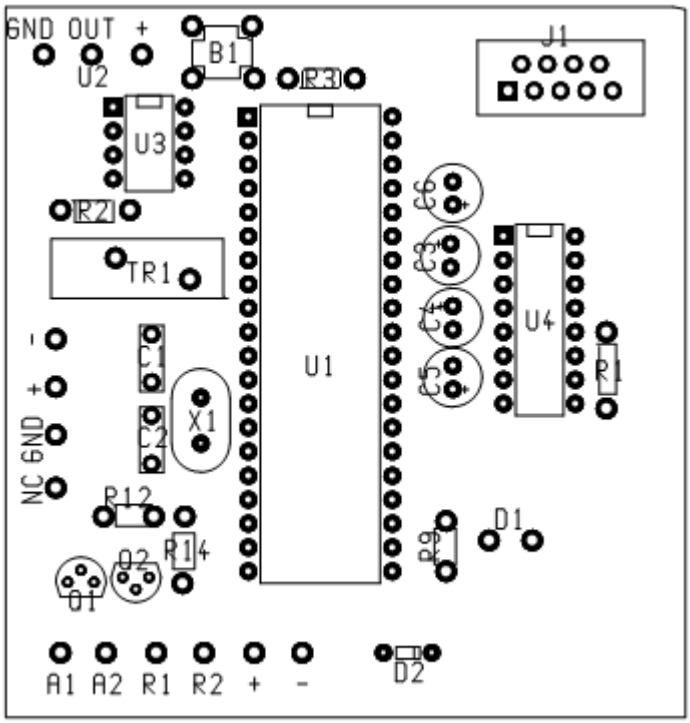
```
;     movlw .200
;     CALL ENVIAR_DADO
;-------------------------------------------------------------
ENVIAR_DADO
        BANK1                   ; ALTERA P/ BANCO 1 DA RAM
        BTFSS  TXSTA,TRMT       ; O BUFFER DE TX ESTÁ VAZIO ?
        GOTO   $-1              ; NÃO - AGUARDA ESVAZIAR
        BANK0                   ; SIM - VOLTA P/ BANCO 0 DA RAM
        MOVWFTXREG              ; SALVA WORK EM TXREG (INICIA TX)
        return
;-------------------------------------------------------------

;*************************************************************



;-------------------------------------------------------------
;-------------------------CONFIGURAÇÕES DOS RECURSOS DO CHIP
;-------------------------------------------------------------

CONFIGURAR
        clrf    PORTA           ; Inicializa os Port's.
        clrf    PORTB
        clrf    PORTC
        clrf    PORTD
        clrf    PORTE

        BANK1               ; Seleciona banco de memória 1
        movlw  0xff
        movwf  TRISA            ; Configura PortA como entrada.
        movlw  0x00
        movwf  TRISB            ; Configura PortB como saída.
        movlw  b'11101001'
        movwf  TRISC            ; Configura PortC como saída e saída.
        movlw  0x00
        movwf  TRISD            ; Configura PortD como saída.
        movlw  0x07
        movwf  TRISE            ; Configura PortE como entrada e desliga Posta Paralela.

        movlw  0x00
        movwf  OPTION_REG       ; Configura Opções:
                                ; Pull-Up habilitados.
                                ; Interrupção na borda de subida do sinal no pino B0.
                                ; Timer0 incrementado pelo oscilador interno.
                                ; Prescaler WDT 1:1.
                                ; Prescaler Timer0 1:2.

        movlw  0x80
        movwf  INTCON           ; Desabilita todas as interrupções.

        movlw  0x00
        movwf  PIE1             ; Desabilita interrupções periféricas.

        movlw  0x00
        movwf  PIE2             ; Desabilita interrupções periféricas.

        movlw  0x04
        movwf  ADCON1           ; Configura conversor A/D.
                                ; A0,A1,A3 como entradas analógicas.
                                ; A2,A4,A5 como I/O digital.
                                ; PortE como I/O digital.
                                ; Justificado á esquerda.
                                ; Vref+ = VDD (+5V).
                                ; Vref- = GND (0V).
```

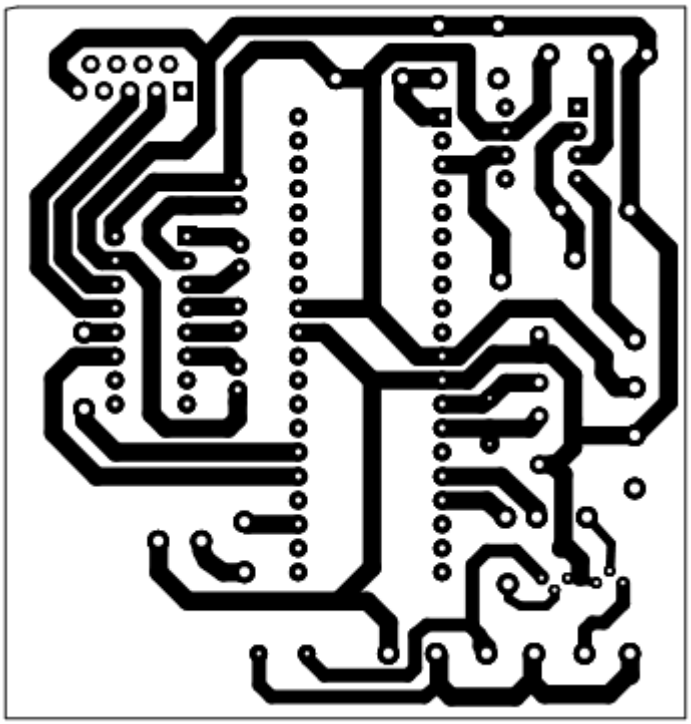
```
        movlw  0x07
        movwf  CMCON        ; Desliga comparadores internos.

        movlw  0x00
        movwf  CVRCON       ; Desliga módulo de refêrencia interna de tensão.

        MOVLW B'00100100'
        MOVWF TXSTA         ; Configura usart
                            ; Habilita tx
                            ; Modo assincrono
                            ; Transmissão de 8 bits
                            ; High speed baud rate
        MOVLW .25
        MOVWF SPBRG         ; ACERTA BAUD RATE -> 9600bps

        MOVLW .249
        MOVWF PR2

        BANK0               ; Seleciona banco de memória 0.

        MOVLW B'10010000'
        MOVWF RCSTA         ; CONFIGURA USART
                            ; HABILITA RX
                            ; RECEPÇÃO DE 8 BITS
                            ; RECEPÇÃO CONTÍNUA
                            ; DESABILITA ADDRESS DETECT

        movlw  0x49
        movwf  ADCON0       ; Configura coversor A/D.
                            ; Velocidade -> Fosc /8.
                            ; Canal 1.
                            ; Módulo ligado.

        movlw b'01001101'   ; Configura PWM
        movwf T2CON         ; Frequência: 1000Hz

        clrf CCPR2L         ; inicia PWM do aquecedor em 0%
        movlw b'00001111'
        movwf CCP2CON

        clrf CCPR1L         ; inicia PWM do resfriador em 0%
        movlw b'00001111'
        movwf CCP1CON


;-----------------------------------------------------------------
;--------------------------INICIO DO PROGRAMA PRINCIPAL
;-----------------------------------------------------------------
; Os valores de temperaturas são trabalhados na escala de 8 bits
; para permanescer igual aos limites dos registradores internos do
; chip. Logo, 0ºC corresponde a 0 e 50ºC corresponde a 255.

movlw .0                    ;inicializa as potencias de aquecimento e resf. em 0%
movwf potencia_aque
movwf potencia_resf
call RECUPERA_SETPOINT      ;recupera set_point armazenado na memória não volátil
call CALCULA_INTERVALO_RESF ;calcula o intervalo por unidade de potência de resfrimaneto
call CALCULA_INTERVALO_AQUE ;calcula o intervalo por unidade de potência de
aquecimento

loop:

;****** Faz a conversao AD (resgata valor do sensor)
```

```
        bsf ADCON0,GO           ; Inicia conversão A/D.
        btfsc ADCON0,GO         ; Fim da conversão ?
        goto $-1                ; Não - Volta 1 instrução.
                                ; Sim.
        movf ADRESH,W           ; Salva valor da conversão no work.
        movwf resultado_ad      ; Carrega work em resultado_ad.


;****** LED de atividade
        BSF     LED             ; ASCENDE O LED
        movlw .250              ; Aguarda 500ms
        call DELAYY

        BCF     LED             ; APAGA O LED
        movlw .250              ;Aguarda 500ms
        call DELAYY


;****** Calcula potência do resfriador
; É basicamente a implementação da fórmula
; { set_point+(((50-set_point)/16)*potencia_resf) < resultado_ad
; { resultado_ad < set_point+(((50-set_point)/16)*(potencia_resf+1))
; sendo que algumas partes como (set_point/16) já foram calculadas
; no recebimento do set_point

        movlw .0
        movwf potencia_resf     ; A Potência do resfriador é desconhecida,
                                ; então é 0, a princípio.
        movfw set_point
        movwf temp1             ; Carrega set_point
        incf temp1              ; Evita o set_point (considera valores maiores)

        RESF_LOOP:
           movfw temp1
           subwf resultado_ad,w ; w=resultado_ad-temp1
           btfss STATUS,C
            goto RESF_FIM       ; resultado_ad<temp1
                                ; resultado_ad>=temp1

           incf potencia_resf   ;incrementa potencia em uma unidade
           movfw intervalo_resf
           addwf temp1,w        ;incrementa temp1 com o valor do intervalo
           movwf temp1

           movlw .15
           subwf potencia_resf,w ; w=potencia_resf-15
           btfss STATUS,C
           goto RESF_LOOP       ; potencia_resf<15
                                ; potencia_resf>=15
        RESF_FIM:

;****** Aciona atuador (resfriador) através do PWM
        movfw potencia_resf
        movwf MULT_X            ;converte a escala de 0-15 para 0-255
        CALL MULTIPLICA
        movfw MULT_XR
        bcf CCP1CON,5           ;envia valor para o PWM
        bcf CCP1CON,4
        movwf CCPR1L


;****** Calcula potencia do aquecedor
; É basicamente a implementação da fórmula
; { (set_point/16)*potencia_aque < resultado_ad
```

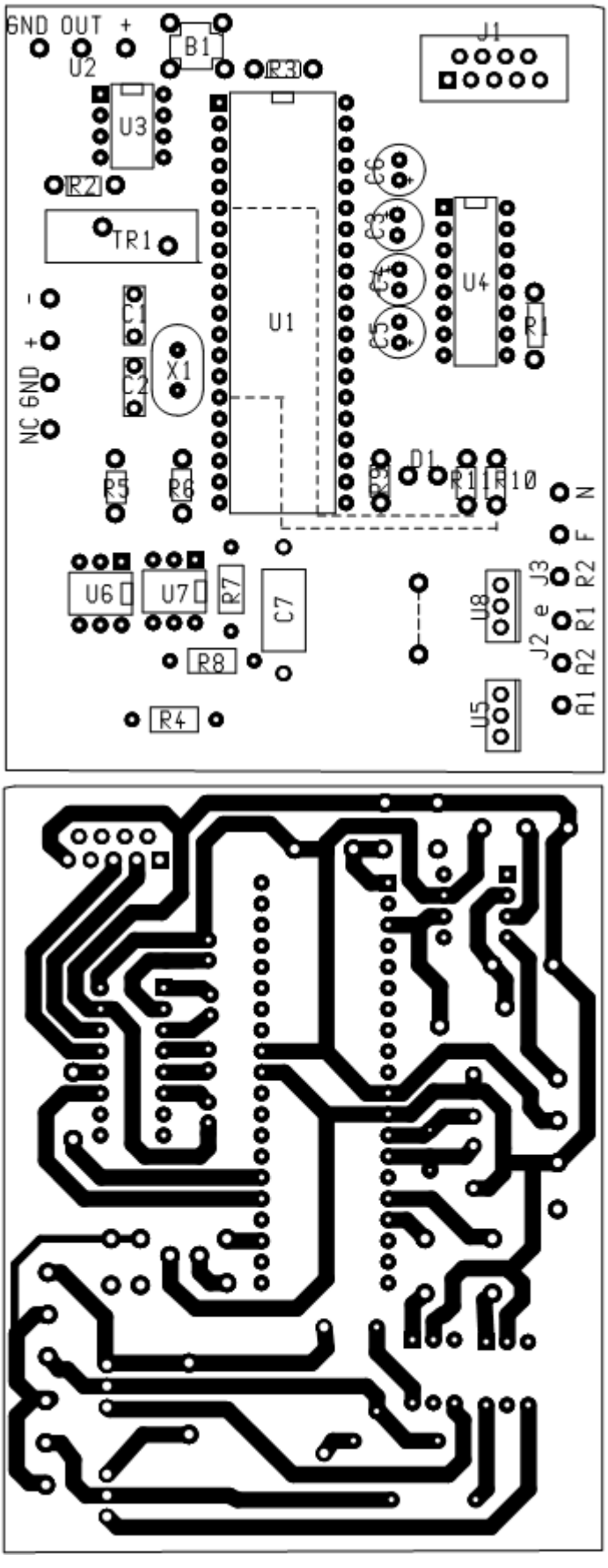
```
; { resultado_ad < (set_point/16)*(potencia_aque+1)
; sendo que algumas partes como (set_point/16) já foram
; calculadas no recebimento do set_point

        movlw .0
        movwf potencia_aque         ;A Potência do resfriador é aquecedor,
                                    ; então é 0, a princípio.
        movfw set_point
        movwf temp1                 ;carrega set_point

        AQUE_LOOP:
           movfw temp1
           subwf resultado_ad,w     ; w=resultado_ad-temp1
           btfss STATUS,C
            goto $+2                ; resultado_ad<temp1
            goto AQUE_FIM           ; resultado_ad>=temp1

           incf potencia_aque       ;incrementa potencia em uma unidade
           movfw intervalo_aque
           subwf temp1,w            ;decrementa temp1 com o valor do intervalo
           movwf temp1

           movlw .15
           subwf potencia_aque,w    ; w=potencia_aque-15
           btfss STATUS,C
            goto AQUE_LOOP          ; potencia_aque<15
                                    ; potencia_aque>=15
        AQUE_FIM:

;****** Aciona atuador (aquecedor) através do PWM
        movfw potencia_aque
        movwf MULT_X                ;converte a escala de 0-15 para 0-255
        CALL MULTIPLICA
        movfw MULT_XR
        bcf CCP2CON,5               ;envia valor para o PWM
        bcf CCP2CON,4
        movwf CCPR2L


;****** SERIAL - trasmite valor da conversao
        movfw  resultado_ad
        call ENVIAR_DADO

;****** SERIAL - verifica e recebe os dados de set_point
        BTFSC  PIR1,RCIF            ; Recebeu algum dado na serial ?
        call RECEBER_DADO           ; SIM
                                    ; NAO

;****** Repete o Loop principal
        goto     loop               ; Vai para loop e repete o programa principa.

        end                         ; Fim do Programa.
```

APÊNDICE D – DESENHO PARA PLACA DE CIRCUITO IMPRESSO – MODELO DC

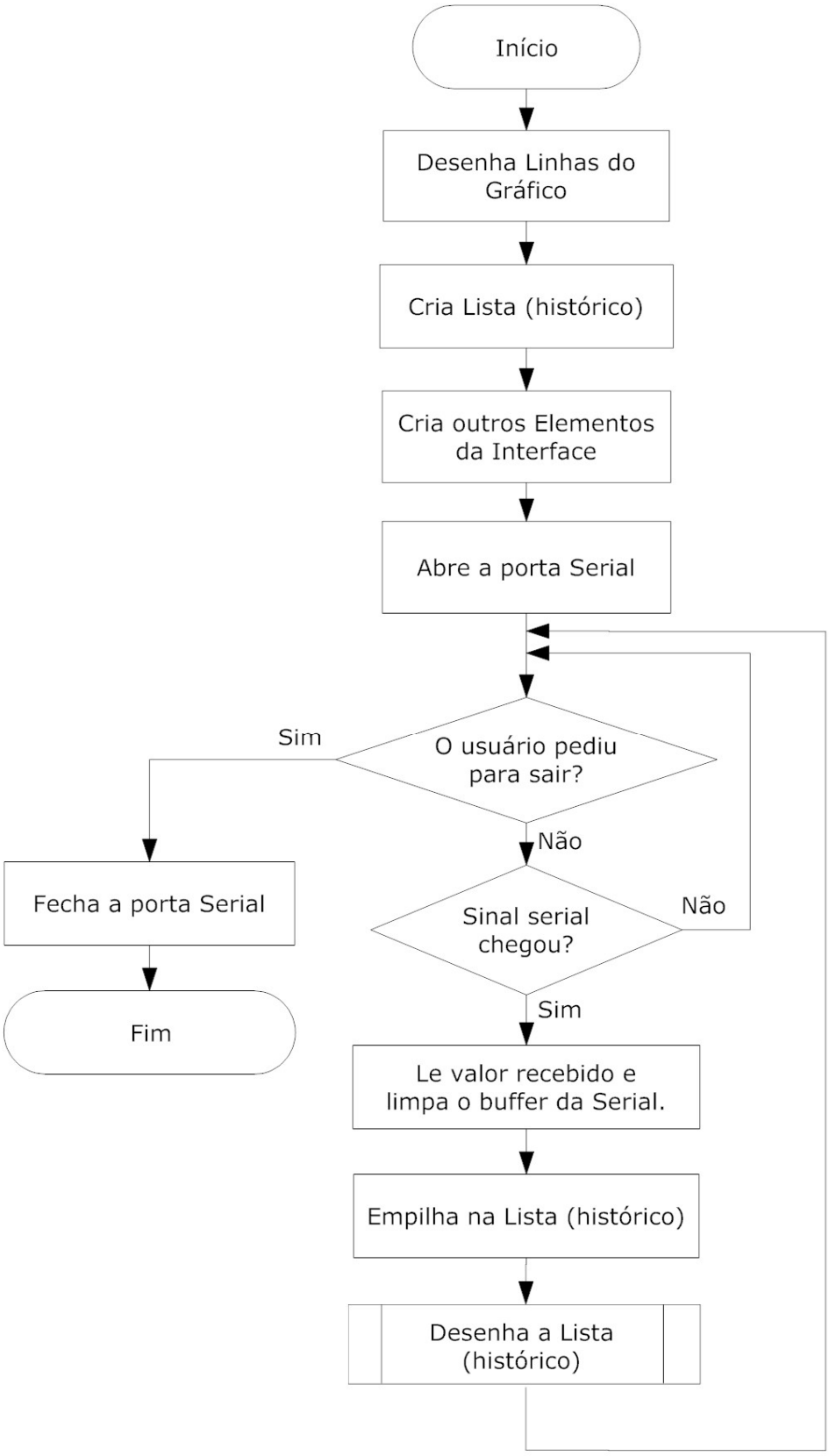


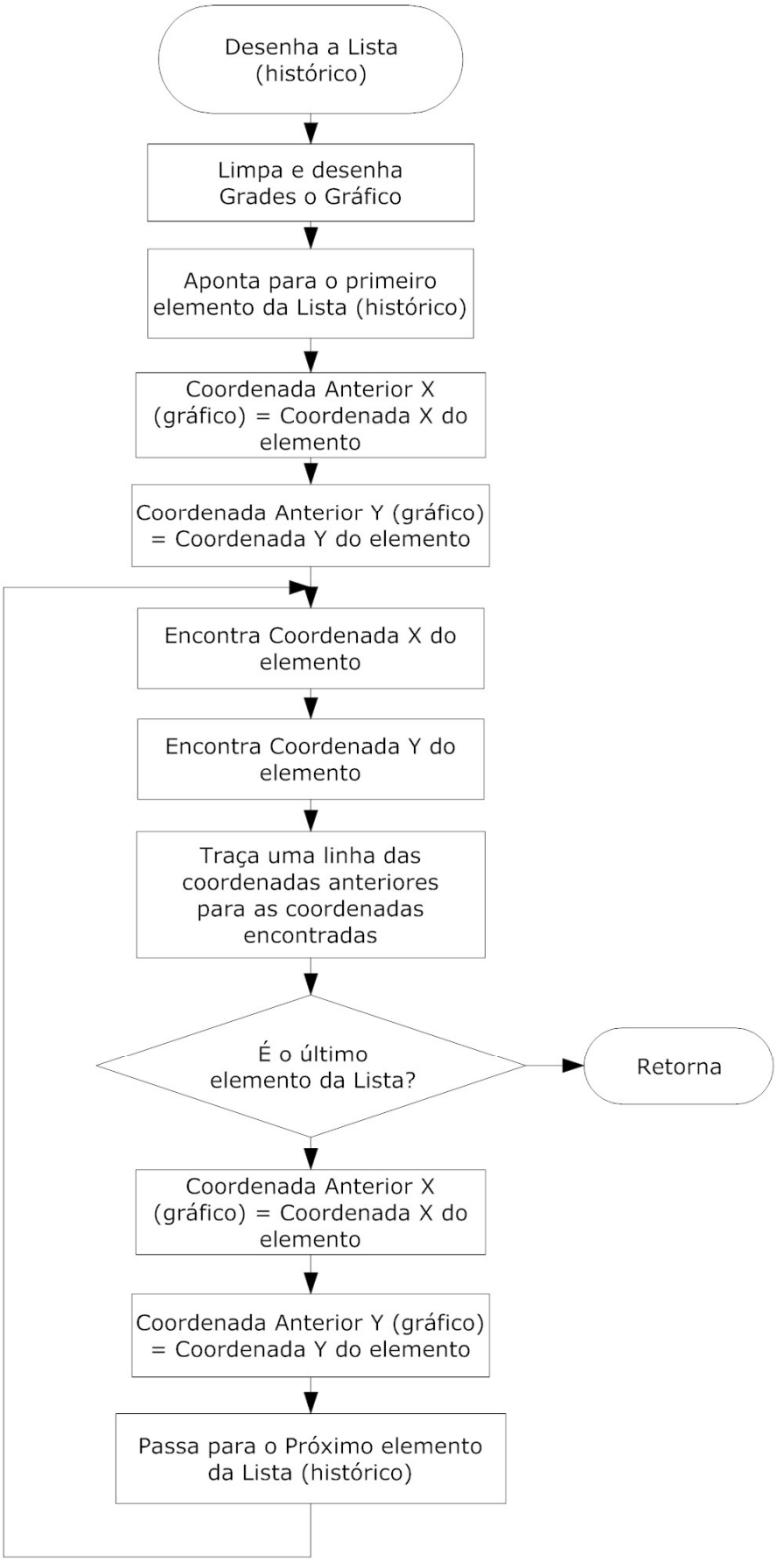


APÊNDICE E – DESENHO PARA PLACA DE CIRCUITO IMPRESSO – MODELO AC



APÊNDICE F – FLUXOGRAMA DA INTERFACE




Desenha a Lista (histórico)
Limpa e desenha Grades o Gráfico
Aponta para o primeiro elemento da Lista (histórico)
Coordenada Anterior X (gráfico) = Coordenada X do elemento
Coordenada Anterior Y (gráfico) = Coordenada Y do elemento
Encontra Coordenada X do elemento
Encontra Coordenada Y do elemento
Traça uma linha das coordenadas anteriores para as coordenadas encontradas
É o último elemento da Lista?
Retorna
Coordenada Anterior X (gráfico) = Coordenada X do elemento
Coordenada Anterior Y (gráfico) = Coordenada Y do elemento
Passa para o Próximo elemento da Lista (histórico)