Universidade Federal de Minas Gerais
Engenharia de Controle e Automação

Controle de temperatura de núcleos de processamento por meio de placas de Peltier.

Projeto final de curso

Aluno: Israel Guido Fontana 2016068811

Professor orientador: Víctor Costa da Silva Campos
Departamento de engenharia eletrônica

Belo Horizonte, Julho de 2023

Monografia

Controle de temperatura de núcleos de processamento por meio de placas de Peltier.

Monografia submetida à banca examinadora designada pelo colegiado didático do curso de graduação em engenharia de controle e automação da Universidade Federal de Minas Gerais, como parte dos requisitos para aprovação na disciplina Projeto Final de Curso II.

Belo Horizonte, Julho de 2023

Resumo

O objetivo principal do trabalho é verificar a viabilidade de um sistema mais eficaz em relação aos métodos convencionais, com o intuito de diminuir a temperatura de um núcleo de processamento, como CPU *(Central Processing Unit)* e GPU *(Graphics Processing Unit)*. Neste caso, foi adotada uma CPU para realização dos experimentos, uma vez que é mais fácil adaptar a placa de Peltier e coletar os dados do sistema em tempo real, tanto no uso percentual do núcleo quanto na temperatura da unidade de processamento.

O protótipo foi desenvolvido baseado em um controle por histerese com *on-off* (liga-desliga) que é uma técnica utilizada para controlar sistemas que possuem um comportamento não linear ou imprevisível. Ele é baseado no princípio da histerese, que é a tendência de um sistema de apresentar diferentes respostas para o mesmo estímulo, dependendo do histórico dos eventos anteriores. No controle em histerese, são definidos dois limites ou níveis de referência, conhecidos como "limite superior"e "limite inferior".

O sistema é projetado para alternar entre dois estados de ação com base na posição da variável controlada em relação a esses limites. No entanto, há como desenvolver uma função que lineariza valores da saída de controle com base nesses limites.

A placa de Peltier é um componente que tem a propriedade de aquecer um lado e resfriar o outro quando fornecido energia para ela. Dessa forma, a parte fria fica em contato com o processador e a parte quente com o dissipador de calor. Essa energia pode ser controlada por um sistema, como uma placa Arduino em conjunto com um PWM (Pulse Width Modulation), realizando a função de monitorar, controlar e fornecer energia para a placa conforme programado.

Assim, os mesmos testes foram realizados em temperatura ambiente e no mesmo local, alternando apenas o uso da placa de Peltier e do dissipador convencional para comparar os resultados em condições semelhantes.

Com os resultados obtidos, observa-se que há vantagens no uso da placa de Peltier em algumas situações e desvantagens em outras. Ajustes nas características físicas da placa de Peltier podem melhorar algumas situações específicas, que serão abordadas posteriormente. Portanto, existe a possibilidade de tornar viável o uso desse sistema para auxiliar na troca de calor com o processador, embora sejam necessários alguns aperfeiçoamentos.

Palavras-chave: Peltier, controle, Arduíno, processador, temperatura.

Abstract

The main objective of the work is to verify the feasibility of a more efficient system compared to conventional methods, aiming to decrease the temperature of a processing core, such as CPU (Central Processing Unit) and GPU (Graphics Processing Unit). In this case, a CPU was adopted for conducting the experiments, as it is easier to adapt the Peltier plate and collect real-time data from the system, both in terms of core utilization percentage and temperature of the processing unit.

The prototype was developed based on hysteresis on-off control, which is a technique used to control systems that exhibit nonlinear or unpredictable behavior. It is based on the principle of hysteresis, which is the tendency of a system to exhibit different responses to the same stimulus, depending on the history of previous events. In hysteresis control, two limits or reference levels are defined, known as the "upper limit"and "lower limit."

The system is designed to switch between two action states based on the position of the controlled variable in relation to these limits. However, it is possible to develop a function that linearizes control output values based on these limits.

The Peltier plate is a component that has the property of heating one side and cooling the other when supplied with energy. Thus, the cold side is in contact with the processor, while the hot side is in contact with the heat sink. This energy can be controlled by a system, such as an Arduino board in conjunction with Pulse Width Modulation (PWM), performing the function of monitoring, controlling, and supplying power to the plate as programmed.

Therefore, the same tests were performed at room temperature and in the same location, only alternating the use of the Peltier plate and the conventional heat sink to compare the results under similar conditions.

Based on the results obtained, it is observed that there are advantages to using the Peltier plate in some situations and disadvantages in others. Adjustments to the physical characteristics of the Peltier plate can improve specific situations, which will be addressed later. Therefore, there is the possibility of making the use of this system feasible to assist in heat exchange with the processor, although some refinements are required.

Keywords: Peltier, control, Arduino, processor, temperature.

Agradecimentos

A profunda gratidão é o sentimento com Deus, pois ele sempre foi essencial em todas as minhas conquistas.

Sou imensamente grato à Universidade Federal de Minas Gerais por me proporcionar a oportunidade de cursar a engenharia de controle e automação e conseguir um título de tamanha importância.

Gostaria de expressar meu sincero agradecimento a todos os funcionários da universidade que, de alguma forma, contribuíram para o meu desenvolvimento ao longo dos anos, suas dedicações e serviços desempenharam um papel crucial no meu crescimento e aprendizado, bem como na ordem de toda a universidade.

Também gostaria de estender meu imenso obrigado a todos os professores desta instituição educacional, os quais contribuíram grandemente para a conclusão desta graduação. É por meio dos seus ensinamentos que minha jornada acadêmica se tornou possível. Sou especialmente grato ao meu orientador, Víctor Costa da Silva Campos, por me auxiliar neste projeto final, agradeço sua dedicação, tempo e o apoio que você prestou, incluindo os recursos úteis que me auxiliaram na conclusão deste trabalho.

Por fim, minha gratidão vai também para os meus pais, cujo apoio e assistência foram inestimáveis ao longo da minha jornada. Não é apenas a força deles, mas também a ajuda contínua que tornaram possível meu ingresso e conclusão bem-sucedida neste curso.

Lista de ilustrações

Sumário

1 Introdução

1.1 Motivação e justificativa

A proposta do projeto final de curso consiste em um sistema ativo de controle de temperatura por histerese com *on-off*, que será aplicado em um processador de computador convencional por meio de uma placa de Peltier e um dissipador de calor (resfriamento a água).

Nesse sistema, a potência fornecida para a placa de Peltier poderá ser controlada através de um módulo PWM o qual realizará a função *on-off*. O monitoramento e controle serão realizados por uma placa Arduino, que receberá do computador o valor atual de temperatura e atuará no sinal PWM.

A ideia para esse projeto tem o objetivo de manter as temperaturas dos componentes dos computadores em um nível baixo e estável. Isso é importante para prolongar a vida útil dos componentes eletrônicos, dentro da faixa de operação recomendada pelos fabricantes.

Ao longo do tempo, foram observadas novas soluções de resfriamento disponíveis no mercado, como o resfriamento a ar mais robusto (com dissipador metálico grande e uma ou duas ventoinhas) e o resfriamento a água, conhecido como *watercooler* (com dissipador metálico, bomba d'água, radiador e ventoinhas).

Também foram consideradas pastas térmicas de maior condutividade térmica e até mesmo os metais líquidos, que são usados para melhorar a transferência de calor entre o processador, pastilha de Peltier e o dissipador metálico. No entanto, o uso de metais líquidos no lugar da pasta térmica apresenta riscos de condutividade elétrica, logo o seu uso foi descartado.

Diante disso, surgiu a questão de utilizar a placa de Peltier como uma possível solução. Inicialmente, foi observado que a temperatura em sua parte fria pode chegar abaixo de 0 grau Celsius. Portanto, houve interesse em explorar seu uso para trocar calor com o processador do computador.

1.2 Problemas e premissas

Como a placa de Peltier é um componente ativo, ou seja precisa-se fornecer energia para aproveitar das suas características, isso é um fator que deve ser observado em relação a sua eficiência em relação aos métodos convencionais já disponíveis no mercado, mas a priori o intuito do trabalho é identificar se a temperatura do processador se beneficia da aplicação da pastilha de Peltier. O gasto energético é uma condição secundária, porém relevante e será abordado ao longo do trabalho.

A placa de Peltier utilizada no experimento tem um encapsulamento cerâmico, o qual não tem uma boa condutividade térmica em comparação com encapsulamentos metálicos. Na verdade a cerâmica é um isolante térmico, isso se deve à preocupação em isolar seus componentes internos devido as suas características de operação. Com isso, como a placa de Peltier faz a interface entre o processador e o dissipador, a baixa condutividade térmica da cerâmica é uma condição limitante para um melhor aproveitamento do sistema desenvolvido.

1.3 Objetivos do projeto

1.3.1 Objetivo Geral

Identificar através dos dados coletados se há a viabilidade da implementação do sistema em computadores convencionais.

1.3.2 Objetivos específicos

• Verificar através de um controle proporcional de potência utilizando-se de um modelo de histerese *on-off*, menores temperaturas em atividades comuns no computador.
• Limitar a potência em determinadas faixas de temperatura para evitar condensação de água.
• Encontrar o valor máximo e mínimo de temperatura que o processador atinge com o sistema.
• Viabilidade e diferenças para usuários convencionais e usuários que visam o uso extremo.

1.4 Estrutura da monografia

O trabalho está dividido em cinco capítulos, este primeiro aborda superficialmente o que será desenvolvido ao longo do texto. Por sua vez, o segundo capítulo relata os componentes e como foi feito a montagem e programação do sistema. O terceiro capítulo aborda teorias importantes que foram presentes no experimento bem como uma breve revisão bibliográfica de temas relacionados ao projeto. O quarto capítulo detalha as informações obtidas no experimento. E por fim o quinto capítulo contém a conclusão das premissas levantadas e possíveis melhorias do sistema.

2 Descrição do processo e desenvolvimento

O modelo do sistema completo é apresentado no diagrama abaixo, e em seguida a imagem do sistema real:

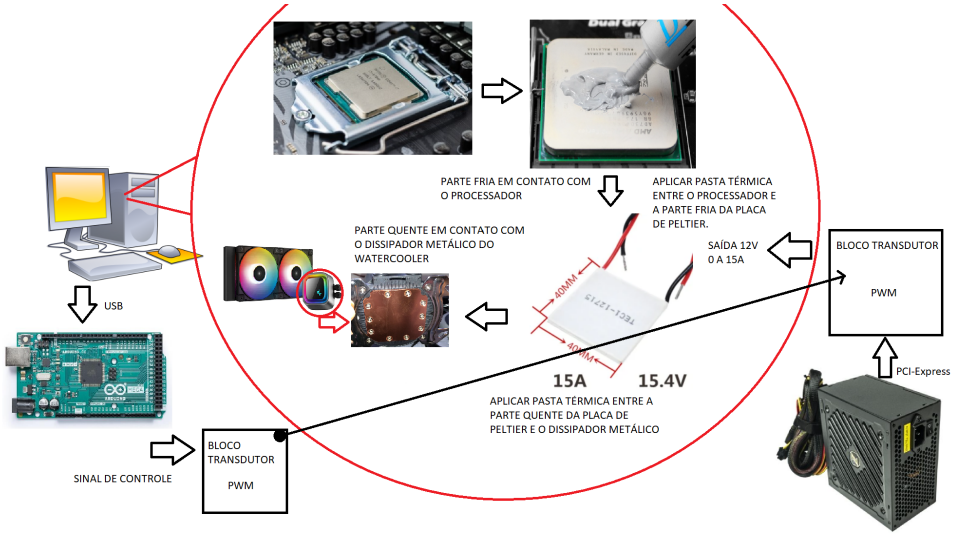

Figura 1 – Diagrama do sistema. Autoria própria.

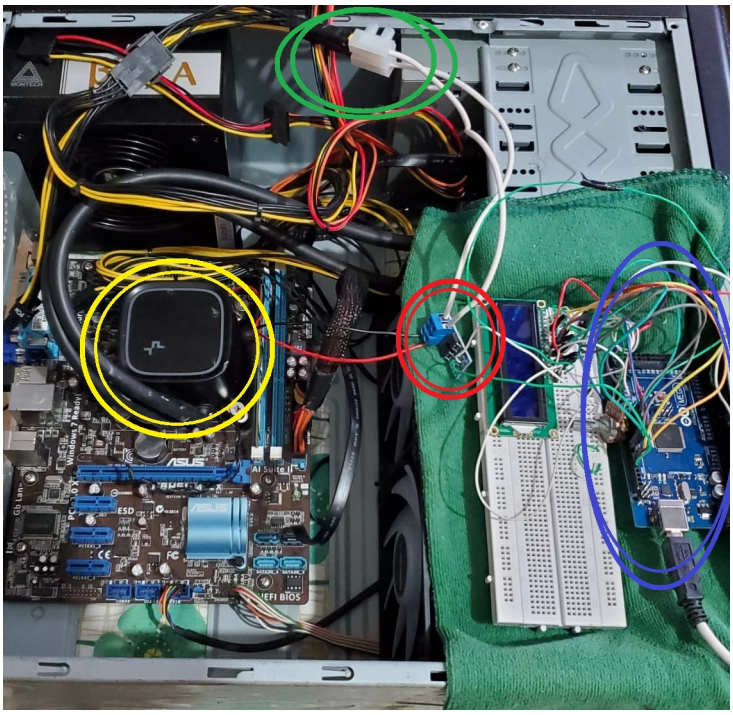

Figura 2 – Sistema real do trabalho. Em amarelo o sistema de resfriamento em conjunto com a placa de Peltier, em vermelho o módulo PWM, em azul o Arduíno, e de verde a entrada de energia para controle do PWM à placa de Peltier. Autoria própria.

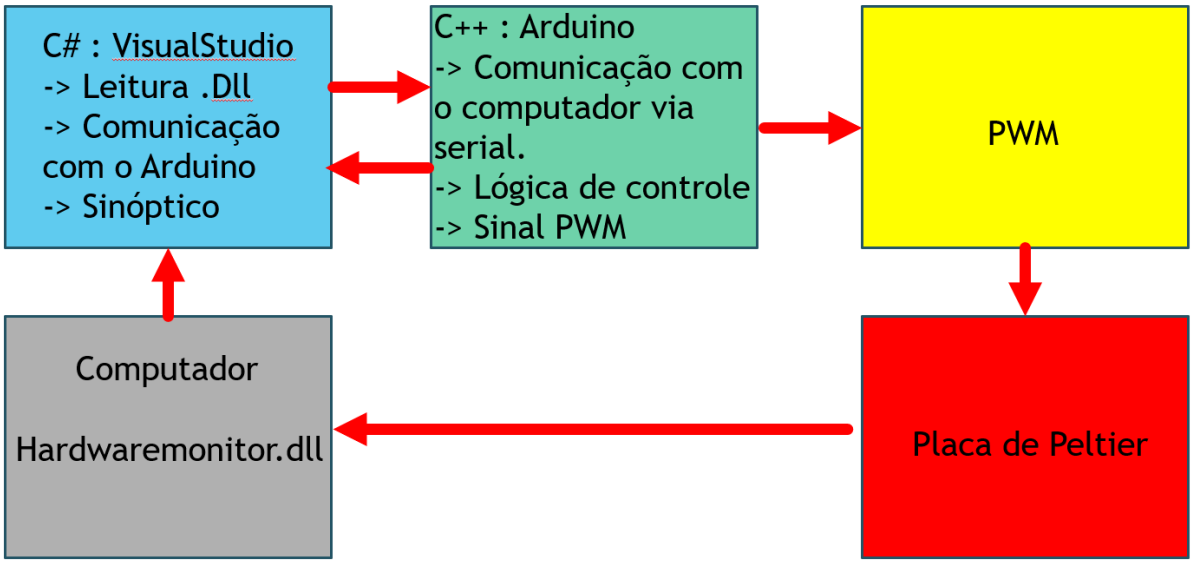

Figura 3 – Arquitetura de comunicação e controle do projeto.

2.1 Processo de montagem e análise

A partir das imagens 1 e 2 tem-se o sistema de estudo do trabalho, com isso os componentes do computador são :

- CPU Intel Core i5-2310 (4 núcleos) com potência térmica máxima dissipada de 95W.
- Placa mãe Asus P8H61-M LX.
- Armazenamento de 256GB SSD SATA.
- 8GB de memória RAM.
- Sistema operacional Windows 10 PRO.
- Fonte de energia Montech Beta Bronze de 650W.
- WaterCooler (resfriamento a água) DeepCool LE500.
- Pasta térmica da marca Thermal Compound ZF-12.

Componentes para realizar o controle de temperatura

- Placa de Peltier Modelo TEC1 - 12715.
- Arduino Mega 2560.
- Módulo De Potência Mosfet PWM 15A 400W Arduino.

Com os componentes explicitados, o objeto de controle é a temperatura do pacote do CPU Intel Core i5-2310, conhecido do termo em inglês *package temperature*, que se refere à temperatura física do encapsulamento do processador mostrada na figura 1, na parte onde mostra a aplicação da pasta térmica. Os componentes como a placa de Peltier, o Arduíno e o PWM serão apresentados com mais detalhes na próxima seção.

A partir dos seguintes componentes primariamente instalados no computador ; placa mãe, memória ram, fonte de energia e armazenamento com o sistema operacional, segue-se para as etapas de montagem do sistema do trabalho;

• Colocar o processador no local de instalação (*socket*).
• Aplicar uma fina camada de pasta térmica sobre o encapsulamento do processador.
• Colocar a parte fria da placa de Peltier sobre a pasta térmica. Deve-se analisar a polaridade e guardar qual fio é o positivo e o negativo, pois se inverter a polaridade o lado frio se torna quente e o lado quente se torna frio.
• Aplicar uma fina camada de pasta térmica sobre a parte quente da placa de Peltier.
• Instalar o resfriamento a água tomando cuidado em relação a placa de Peltier que se torna escorregadia devido à pasta térmica, sendo assim ter a certeza que a placa está cobrindo toda a superfície do encapsulamento do processador. Apertar os parafusos de fixação do resfriamento a água de maneira cautelosa, não deve-se pressionar muito, pois pode quebrar a porcelana da placa de Peltier e danificar o componente.
• Selecionar os fios positivos e negativos da fonte de alimentação com o auxílio de um voltímetro, para juntar todos em apenas dois fios, um positivo 12V+ e outro 0V, assim levando-os para a entrada de alimentação do Módulo PWM.
• Colocar os fios da placa de Peltier nas respectivas saídas do Módulo PWM obedecendo a polaridade.
• Conectar os fios do módulo PWM, do sinal PWM e GND (*Ground*), terra em português, no Arduíno, na respectiva saída de acordo com a programação.
• Conectar via USB a placa Arduíno a uma entrada USB do computador.

2.2 Componentes da instrumentação do processo

2.2.1 Placa de Peltier (Atuador)

A placa de Peltier é basicamente um encapsulamento cerâmico o qual contém células semicondutoras em seu interior, quando é aplicado uma corrente elétrica nas células, uma parte da junção das células esfria, conhecido como lado frio da placa, e a outra parte esquenta, chamado de lado quente.

A pastilha de Peltier do modelo utilizado (TEC1-12715) pode trabalhar com tensões de 12V a 15.4V e com uma corrente máxima de 15A, podendo operar o lado frio e o lado quente na faixa de -30°C a 70°C. Logo, há particularidades as quais devem ser observadas, como condensação de água em certas faixas de temperatura e boa dissipação do lado quente.

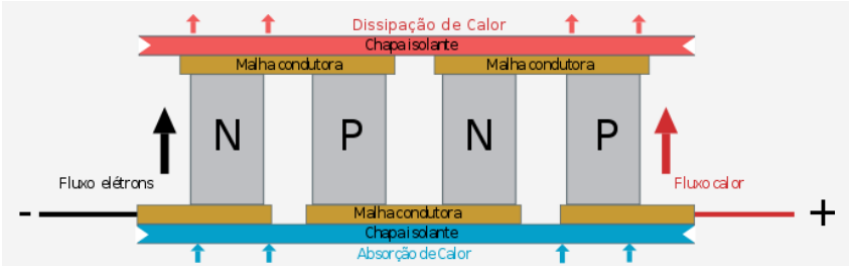

Figura 4 – Diagrama interno de uma placa de Peltier. (YHAEFFNER, 2016)

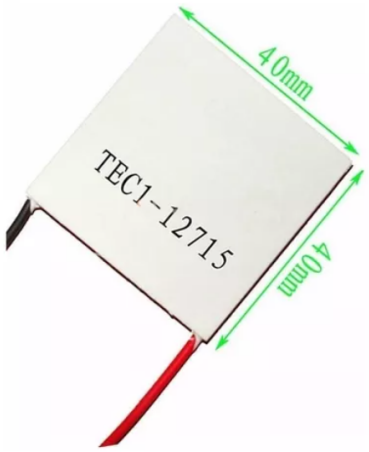

Figura 5 – Placa de Peltier e modelo a ser usado no trabalho (TEC1-12715). Imagem genérica.

À medida que a placa de Peltier troca calor com o processador, consegue-se analisar a eficiência térmica e de consumo em comparação aos sistemas convencionais que há hoje no mercado, como refrigeração a ar (*aircoolers*) ou resfriamento a água (*watercoolers*).

A atuação para que a placa de Peltier ajude a dissipar o calor do processador é basicamente esfriar mais enquanto o processador está mais quente, ou seja fornecer mais

energia para a placa, e quando o processador estiver em baixa carga de uso, diminui-se então a potência na placa de Peltier. Logo, quanto mais potência aplicada à placa de Peltier mais calor ela irá dissipar também, portanto é imprescindível uma boa dissipação da parte quente da placa.

Sendo assim, a placa arduíno vai receber a informação da temperatura em tempo real por uma API (Interface de Programação de Aplicações em português) pelo barramento USB. Com isso um código interno de controle no arduíno vai atribuir um sinal de saída de controle em PWM, o qual irá para o transdutor (módulo PWM).

A fonte utilizada para alimentar a placa de Peltier será uma fonte padrão de computador, utilizando o barramento PCI-Express (6+2 pinos) o qual consegue alimentar até 150 Watts na faixa média de 12 Volts, caso necessário pode-se utilizar dois barramentos PCI para ter uma maior tolerância energética.

O *datasheet*[1] pode ser acessado em (RAJGURUELECTRONICS, 2023), o modelo apresentado no *datasheet*[1] é o de dimensões 50x50x4mm, mas tem o mesmo número de células que o 40x40x4mm.

***Datasheet*[1]** = Folha de dados que contém as especificações técnicas de um componente.

2.2.2 Módulo PWM (transdutor)

O módulo mosfet PWM é utilizado em diversos projetos eletrônicos para controle de motores, lâmpadas, bombas e outros dispositivos.

Este módulo trabalha com tensões entre 5 e 36 VDC (tensão contínua) na entrada e é controlado por microcontroladores e placas como Arduino e Raspberry Pi.

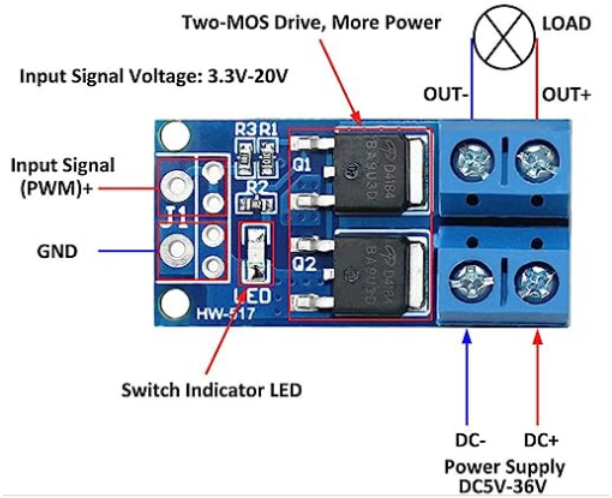

Figura 6 – Módulo PWM utilizado no trabalho. Imagem genérica obtida em (WAYINTOP, 2020).

Especificações básicas :

- Módulo com dois Mosfets.
- Tensão de entrada: 5 a 36 VDC.
- Tensão de operação PWM: 3.3 a 20 VDC.
- Corrente máxima: 15A.
- Potência máxima: 400W.
- Dimensões: 40 x 40 x 3,3 mm. Largura, comprimento e altura respectivamente.

De acordo com a figura 6 representativa do módulo, tem-se alguns campos para correta ligação no sistema :

- *Input Signal* [2] (PWM+) = O sinal que será recebido do Arduíno.
- *Two-Mos Drive, More Power* [3] = Mosfets da placa.
- *Load* = Carga a qual a placa irá alimentar, neste caso a placa de Peltier.
- *Out- Out+* = Saídas para a carga, com a polaridade positiva e negativa.
- *Switch indicator led* = Luz a qual indica chaveamento de sinal. • *Power Supply* [4] *DC-DC+*[5] = Entradas, com a polaridade positiva e negativa, aqui sendo utilizada a fonte de alimentação do próprio computador.

Input Signal [2] = Sinal de entrada.
Two-MOS Drive, More Power [3] = Dois módulos mosfet, mais potência.
Power Supply [4] = Fonte de alimentação.
DC [5]= (*Direct Current*) que significa corrente contínua.

A partir dos dados do módulo PWM, o comportamento dele é modular o sinal contínuo que vem da fonte de alimentação do computador, isto é feito chaveando o sinal em ligado e desligado conforme figura representativa abaixo :

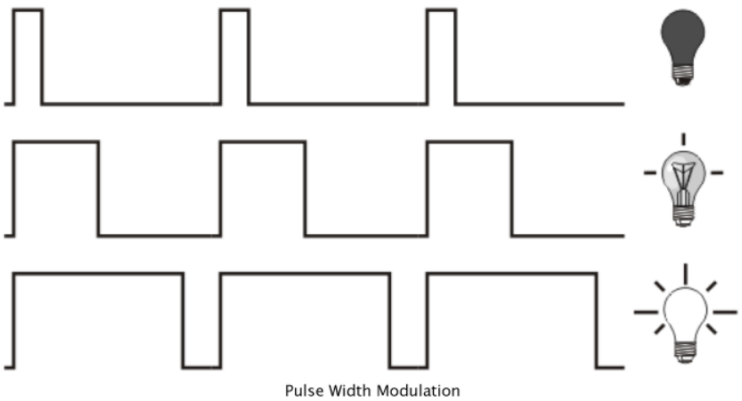

Figura 7 – Gráfico representativo de um sinal PWM. Imagem retirada de (IFSC, 2018)

Utilizando-se da figura 7, há o primeiro gráfico e uma lâmpada mais apagada, mostrando que o interruptor (mosfet) fica mais tempo desligado do que ligado em uma determinada faixa de tempo, com isso se produz um comportamento em que a lâmpada tem pouquíssimo brilho, logo menor potência e gasto energético, pois entrega-se menos energia nesta faixa de tempo.

Em segundo vemos que o interruptor fica mais tempo ligado em relação ao primeiro gráfico, ou seja o mosfet está entregando mais tempo de energia para a lâmpada, logo observa-se um brilho médio. Por fim, no último gráfico o interruptor passa mais tempo ligado do que desligado, sendo assim um brilho mais intenso, e uma maior potência é entregue para a lâmpada.

Analogamente é o que acontece com a placa de Peltier, o arduíno vai controlar o mosfet, e com isso tem-se mais ou menos tempo de entrega de energia para a placa em uma mesma faixa de tempo, desligando e ligando o sinal de energia para a placa, o que será mostrado nos códigos da programação.

2.2.3 Fonte de alimentação

O modelo utilizado foi a fonte Montech BETA Bronze 650W para alimentação geral do computador, e foi aproveitado a saída PCI-Express (6+2) o qual consegue alimentar até 150W na faixa média de 12 Volts para fornecer energia para a placa de Peltier.

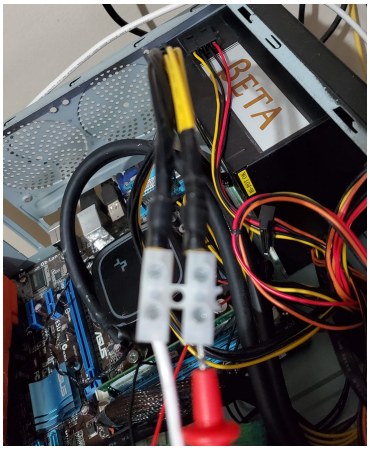

Figura 8 – Separação dos fios positivos e negativos do conector PCI-Express. Autoria própria.

A separação dos fios do cabo PCI-Express deve se respeitar a questão das cores como na imagem 8, onde os cabos amarelos são os positivos e os pretos os negativos, pela questão da polaridade da pastilhade Peltier. Após separados os fios eles foram conectados na barra de conexão, e em sequência foi verificado os 12 Volts, assim levados pelos dois fios brancos mais grossos na figura 9 para o módulo PWM, onde também pode-se observar o fio vermelho e preto que é da placa de Peltier, o fio verde é o sinal PWM vindo do Arduíno e ao lado dele um fio branco que é o terra.

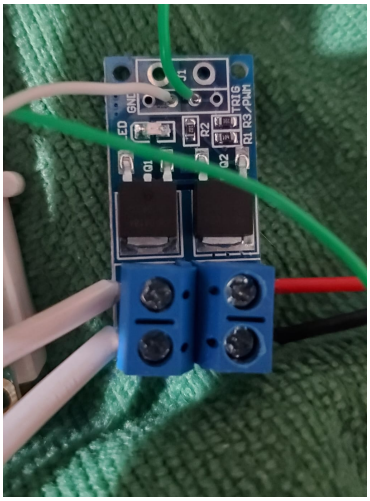

Figura 9 – Ligação da fonte no módulo PWM. Autoria própria.

2.2.4 Sensor de temperatura

O sensor de temperatura utilizado foi o interno do processador.

2.3 Códigos da programação do sistema

2.3.1 Programação de interface com o Arduíno

Como mostrado na seção anterior 2.2.4, o sensor de temperatura está interno ao processador. Para conseguir usar essa informação (temperatura do processador) em tempo real, e com isso atuar no sistema, foi necessário usar uma biblioteca (*OpenHardwareMonitor*) na linguagem Csharp (C#) para enviar esta informação para o Arduíno.

Ao passo que foi necessário utilizar a linguagem C# no Microsoft Visual Studio para aquisição da informação de temperatura, foi desenvolvido uma tela (interface com o usuário) para acompanhar os dados (temperatura do pacote e uso de cada núcleo) em tempo real, conectar o computador com o arduíno, registrar o valor da temperatura de segundo em segundo e guardar em um arquivo texto.

```
1   using System;
2   using System.Collections.Generic;
3   using System.ComponentModel;
4   using System.Data;
5   using System.Drawing;
6   using System.Linq;
7   using System.Text;
8   using System.Threading.Tasks;
9   using System.Windows.Forms;
10  using OpenHardwareMonitor.Hardware;
11  using OpenHardwareMonitor;
12  using System.IO.Ports;
13  using System.IO;
14
15  namespace C__Arduino_temperature_control_project
16  {
        3 references
17      public partial class Form1 : Form
18      {
19          Computer myComputer;
20          List<string> temps_Arduino = new List<string>();
21          string caminhoArquivo = "C:/Users/Public/Desktop/temps.txt";
```

Figura 10 – 1° parte do código em C#. Autoria própria.

A figura 10 mostra as bibliotecas utilizadas, a criação de uma instância da classe *Computer* da biblioteca *OpenHardwareMonitor*, uma lista para armazenar os valores de temperatura recebidos do Arduíno junto ao caminho em que os dados seriam guardados. Utilizou-se o dado vindo do Arduíno em vez de utilizar da própria biblioteca para implementar um código mais completo, assim o computador envia e recebe informações do Arduíno, projetado assim para uma melhoria futura do projeto.

```
24          // Inicializa o componente da classe Form.
            1 reference
25          public Form1()
26          {
27              InitializeComponent();
28          }
29
30          // Manipulador de eventos para o evento Form Load
            1 reference
31          private void Form1_Load(object sender, EventArgs e)
32          {
33              //Inicializa a instância do computador e habilita o monitoramento do CPU
34              myComputer = new Computer() { CPUEnabled = true};
35              myComputer.Open();
36          }
37
38          //Manipulador de evento para o evento Timer Tick
            1 reference
39          private void Timer1_Tick(object sender, EventArgs e)
40          {
41              string hardwareName = "";
42              float temp = 0;
43              byte count = 0;
44              float totalLoad = 0, load1 = 0, load2 = 0, load3 = 0, load4 = 0;
```

Figura 11 – 2° parte do código em C#. Autoria própria.

A imagem 11 apresenta a inicialização de um *form* (classe para tela gráfica da linguagem) bem como um manipulador de eventos para o evento *Form Load*, e também a inicialização da instância do computador que habilita o monitoramento do CPU, junto a um temporizador para atualização das informações.

```
45
46              //Faz iterações pelos itens de hardware no computador.
47              foreach (var hardwareItem in myComputer.Hardware)
48              {
49                  if (hardwareItem.HardwareType == HardwareType.CPU)
50                  {
51                      //Atualiza informações de hardware.
52                      hardwareItem.Update();
53                      foreach (IHardware subHardware in hardwareItem.SubHardware)
54                      {
55                          //Atualiza informações de sub-hardware (como núcleos do processador).
56                          subHardware.Update();
57                      }
58
59                      //Constrói a string do nome do hardware.
60                      foreach (var name in hardwareItem.Name)
61                      {
62                          hardwareName += name;
63                      }
64
```

Figura 12 – 3° parte do código em C#. Autoria própria.

A imagem 12 mostra a varredura pelos sensores os quais são monitorados pela biblioteca e também a construção do nome de cada sensor.

```
//Faz aquisição dos valores de temperatura e carga dos sensores.
foreach (var sensor in hardwareItem.Sensors)
{
    if (sensor.SensorType == SensorType.Temperature)
    {
        temp = sensor.Value.Value;
    }

    else if (sensor.SensorType == SensorType.Load)
    {
        count++;
        switch(count)
        {
            case 1:
                totalLoad = sensor.Value.Value;
                break;
            case 2:
                load1 = sensor.Value.Value;
                break;
            case 3:
                load2 = sensor.Value.Value;
                break;
            case 4:
                load3 = sensor.Value.Value;
                break;
            case 5:
                load4 = sensor.Value.Value;
                break;

        }
    }
}
```

Figura 13 – 4° parte do código em C#. Autoria própria.

A figura 13 contém parte do código referente à atribuição dos valores do sensor para as variáveis criadas.

```
98          //Atualiza a interface de usuário com os valores de hardware e sensor.
99          label_hardwareName.Text = "Processor : " + hardwareName;
100
101
102         progressBar_cpuTemp.Value = Convert.ToInt32(Math.Round(temp));
103         label_cpuTemp.Text = string.Format("{0:0.00} °C", temp);
104
105         progressBar_totalLoad.Value = Convert.ToInt32(Math.Round(totalLoad));
106         progressBar_core1Load.Value = Convert.ToInt32(Math.Round(load1));
107         progressBar_core2Load.Value = Convert.ToInt32(Math.Round(load2));
108         progressBar_core3Load.Value = Convert.ToInt32(Math.Round(load3));
109         progressBar_core4Load.Value = Convert.ToInt32(Math.Round(load4));
110
111
112
113         label_totalLoad.Text = string.Format("{0:0.00} %", totalLoad);
114         label_core1Load.Text = string.Format("{0:0.00} %", load1);
115         label_core2Load.Text = string.Format("{0:0.00} %", load2);
116         label_core3Load.Text = string.Format("{0:0.00} %", load3);
117         label_core4Load.Text = string.Format("{0:0.00} %", load4);
118
119
120         if (serialPort1.IsOpen)
121         {
122             serialPort1.Write(totalLoad + "-" + temp + "\n");
123             LerVariavelSerial();
124         }
125     }
126 }
127 }
```

Figura 14 – 5° parte do código em C#. Autoria própria.

O código referente à figura 14 é responsável por atualizar os dados na tela gráfica que faz a interface com o usuário.

```
119                 // Verifica se a porta serial está aberta
120                 if (serialPort1.IsOpen)
121                 {
122                     // Escreve os valores de carga e temperatura no formato "carga-temperatura" na porta serial
123                     serialPort1.Write(totalLoad + "-" + temp + "\n");
124                     LerVariavelSerial();
125                 }
126             }
127         }
128     }
129
130     //Inicializa uma caixa de seleção na interface gráfica para
131     //escolher dentre as portas COM disponíveis.
        1 reference
132     private void comboBox1_DropDown(object sender, EventArgs e)
133     {   // Obtém uma lista de nomes de portas seriais disponíveis
134         string[] portLists = SerialPort.GetPortNames();
135         // Limpa os itens da combobox
136         comboBox_portLists.Items.Clear();
137         // Adiciona os nomes das portas seriais à combobox
138         comboBox_portLists.Items.AddRange(portLists);
139     }
```

Figura 15 – 6° parte do código em C#. Autoria própria.

A figura 15 mostra um trecho do código que verifica se há alguma porta serial aberta e também inicializa uma caixa de seleção para selecionar qual porta COM o usuário deseja usar.

```
141     private void button1_Click(object sender, EventArgs e)
142     {
143         try
144         {
145             // Configura o nome da porta serial com base na seleção do usuário
146             serialPort1.PortName = comboBox_portLists.Text;
147             serialPort1.BaudRate = 9600;
148             // Abre a porta serial
149             serialPort1.Open();
150
151             MessageBox.Show("Sucess Connected to Arduino Board" + "\n");
152         }
153
154         catch (Exception error)
155         {
156             MessageBox.Show(error.Message);
157         }
158     }
        1 reference
159     private void button2_Click(object sender, EventArgs e)
160     {
161         try
162         {
163             // Fecha a porta serial
164             serialPort1.Close();
165
166             MessageBox.Show("Sucess Disconnected to Arduino Board" + "\n");
167         }
168
169         catch (Exception error)
170         {
171             MessageBox.Show(error.Message);
172         }
173     }
```

Figura 16 – 7° parte do código em C#. Autoria própria.

Quando se cria uma caixa de seleção é necessário criar onde o usuário tem a opção de clicar, com isso a imagem 16 mostra a parte do código onde após selecionar a porta serial desejada, ele pode clicar em conectar ou desconectar da porta, representado na interface como *Open* para conectar e *Close* para desconectar.

```
174     private void Form1_FormCLosing(object sender, FormClosingEventArgs e)
175     {
176         try
177         {   // Fecha a porta serial antes de fechar o formulário
178             serialPort1.Close();
179         }
180
181         catch (Exception error)
182         {
183             MessageBox.Show(error.Message);
184         }
185     }
186
        1 reference
187     private void LerVariavelSerial()
188     {
189         // Lê a linha recebida pela porta serial e adiciona à lista temps_Arduino
190         temps_Arduino.Add(serialPort1.ReadLine());
191
192
193         using (StreamWriter escritor = new StreamWriter(caminhoArquivo))
194         {
195             // Escreve cada valor da lista temps_Arduino em um arquivo
196             foreach (string valor in temps_Arduino)
197             {
198                 escritor.WriteLine(valor);
199             }
200
201         }
```

Figura 17 – 8° parte do código em C#. Autoria própria.

Por fim, a figura 17 finaliza a tela de interface quando o usuário clica no X para sair dela e faz a leitura das temperatura enviadas pelo arduíno e as armazena em um arquivo de texto.

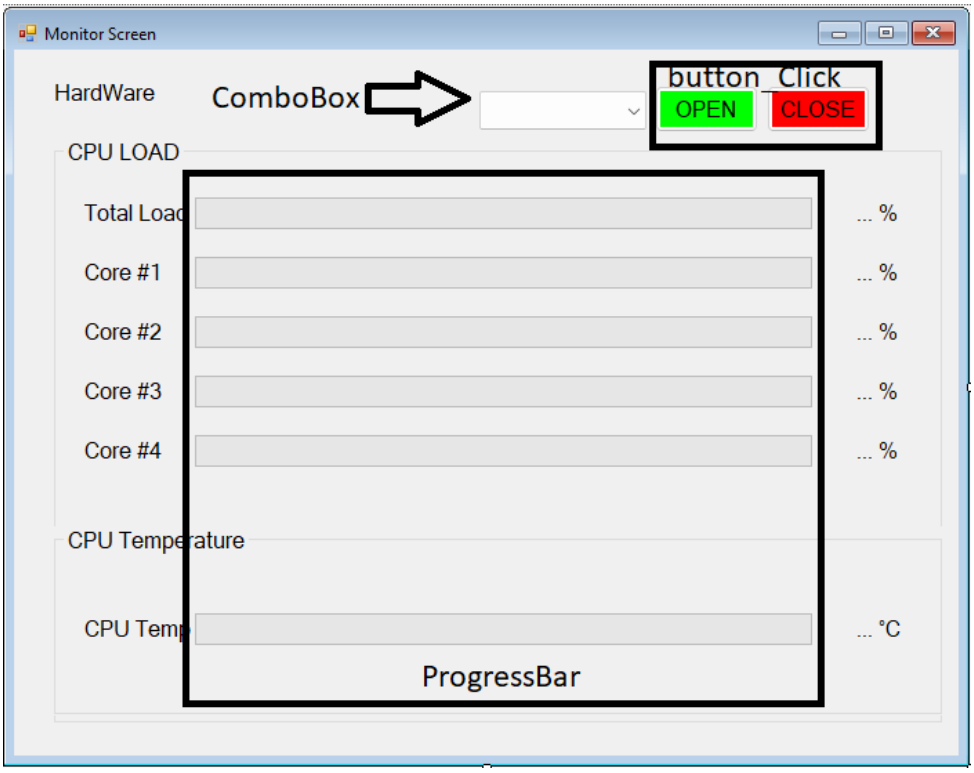

Figura 18 – Interface com o usuário (Form). Autoria própria.

A imagem 18 mostra o que se refere cada campo na linha de código.

2.3.2 Programação do Arduíno

Uma vez criado o código para conseguir enviar os dados via porta serial para o Arduíno, um outro código deve ser implementado no microcontrolador do arduíno na linguagem C++, para que assim consiga-se ler os dados do processador, tratar esses dados, e a partir disto tomar a decisão do sinal a ser enviado para o PWM.

```
1  //Carrega a biblioteca LiquidCrystal
2  #include <LiquidCrystal.h>
3
4  #include <string.h>
5
6  //Define os pinos que serão utilizados para ligação ao display
7  LiquidCrystal lcd(12, 11, 6, 7, 8, 9, 5, 4, 3, 2); //Modo 8 bits
8
9  int pinoPWM = 13; // Define o pino PWM no arduino.
10 int valorPWM = 200; // Variavel para o valor do PWM.
11
12 char c;
13 String dataIn,cpuL,cpuT;
14 float cpuLoad, cpuTemp, currentSignal;
15 int ctrl=0;
```

Figura 19 – 1° parte do código para o Arduíno em C++. Autoria própria.

Na primeira parte do código de acordo com a figura 19, tem-se as duas bibliotecas usadas, uma para tratamento de *strings* e outra para um *display* LCD. A tela LCD foi utilizada para facilitar na sincronização dos dados no início do projeto, mas que nada interfere ou se faz necessário ao controle do projeto.

A variável de controle é a valorPWM a qual já é inicializada em 200 para garantir uma potência alta, para evitar em uma eventual trava do código e tempo até começar receber os dados de temperatura um superaquecimento do processador e placa de Peltier.

```
void setup() {
    Serial.begin(9600);// Inicializa a porta a uma taxa definida e já conhecida para sincronizar com o código em C#
    lcd.begin(16, 2); // Inicializa o display LCD com 16 colunas e 2 linhas
    pinMode(pinoPWM, OUTPUT); // Define que o pino 13 da placa é um pino de saída.
    analogWrite(pinoPWM, valorPWM); // Define a potência para a placa no momento da inicialização completa dos sinais e sistema.
}

void loop() {

  while (Serial.available()>0)// Verifica se há dados disponíveis na porta serial
  {
      c=Serial.read(); Lê o próximo caractere disponível na porta serial
      if (c=='\n'){break;}// Se o caractere lido for uma quebra de linha sai do loop.
      else                {dataIn+=c;}// Concatena o caractere lido à string dataIn
  }

for (int i = 0 ; i < dataIn.length() ; i++ ){// Percorre cada caractere da string dataIn
    if (dataIn.charAt(i) != '-') ctrl+=1; // Se o caractere atual for diferente de '-' incrementa o contador ctrl
    else break; // Sai do loop
}

cpuL = dataIn.substring(0,ctrl); // Obtém a substring que representa a carga do CPU
cpuT = dataIn.substring((ctrl+1),dataIn.length()); // Obtém a substring que representa a temperatura do CPU

cpuLoad = cpuL.toFloat(); // Converte a carga do CPU para um valor decimal
cpuTemp = cpuT.toFloat(); // Converte a temperatura do CPU para um valor decimal

 if (cpuTemp == 0) {valorPWM = 255 ; analogWrite(pinoPWM, valorPWM);}// Se não estiver obtendo dados seta o PWM pra um valor fixo de segurança.
 else if (cpuTemp <= 55) {valorPWM = map(cpuTemp, 0, 55, 25, 255); analogWrite(pinoPWM, valorPWM);} // Mapeia a temperatura do CPU para o intervalo de valores do PWM
 else if (cpuTemp > 55 ) analogWrite(pinoPWM, 255); // Define o valor máximo para o PWM no pino pinoPWM


Serial.println(cpuT); // envia via serial o valor da variável para ser armazenado no arquivo txt pelo código em C#

 c=0; // Reinicia a variável c
```

Figura 20 – 2° parte do código para o Arduíno em C++. Autoria própria.

O código mostrado na figura 20 apresenta a parte em que o Arduíno lê o conjunto de caracteres vindo do C#, em um tempo de amostragem de 1 segundo, converte esses valores para numéricos, pois eles vem como cadeia de caracteres, e a partir disso consegue-se comparar esses valores para mapear e assim definir qual a potência será enviada para a placa de Peltier através do valor de temperatura daquele instante de tempo.

O sinal PWM trabalha com valores na faixa de 0 a 255, 0 significa nenhuma energia, analogamente a um interruptor sempre aberto, e 255 se traduz como entregar toda a energia disponível na entrada para a saída, ou seja interruptor sempre fechado, e valores intermediários nessa faixa é comparável à explicação da imagem 7.

Sendo assim, no código há três condições, primeiro quando não se tem dados disponíveis, assim o arduíno interpreta que a temperatura do processador é 0, com isso ele mantém a potência máxima para evitar sobreaquecimento. A segunda condição é quando o Arduíno detecta que tem informação serial, a partir disso ele entra na condição de mapeamento, representado pela função *map*. Por último a terceira condição é quando a temperatura ultrapassa o valor de limite máximo determinado na programação, configurando a saída PWM para o máximo.

A função *map* (A,B,C,D), representa o controle por histerese onde A, B representa a faixa de temperatura a ser mapeada (limite inferior e superior respectivamente), C e D são os valores a atribuir em uma escala linear para a saída PWM. Sendo assim, a função map (A,B,C,D) tem a seguinte função afim para a saída de controle: $\mathrm{Y(X)} = \frac{D-C}{B-A} \cdot (X - A) + C$ com X podendo assumir valores entre A e B, incluindo A e B.

No caso do código da figura 20 tem-se para (A,B,C,D) os seguintes parâmetros (0,55,25,255) ou seja $Y = 230/55 \cdot X + 25$. Caso a temperatura na variável CpuTemp chegue a 0°C a função deveria atribuir 25 na saída, porém a primeira condição será acionada com 255 na saída, logo quando controlado a única forma do CPU chegar nessa temperatura é utilizando a potência máxima, então na prática e fisicamente não há problemas.

Por outro lado, quando a temperatura chegar em 55°C ou acima o controle atribui os 255, liberando assim potência máxima, e por exemplo quando a temperatura for 25°C tem-se $Y = 230/55 \cdot 25 + 25$, logo $Y \approx 130$. Por fim, sempre que a temperatura for maior que 55°C o valor do PWM fica configurado no máximo, em 255.

```
53    delay(1000);
54    dataIn="";cpuL="";cpuT="";
55
56    ctrl=0;
57
58
59  }
```

Figura 21 – 3° parte do código para o Arduíno em C++. Autoria própria.

Fim do código mostrado pela figura 21 que limpa as variáveis de controle e de dados para uma nova aquisição de valores em um tempo de atualização de 1 segundo.

2.4 Resumo do capítulo

Com os componentes e a programação do sistema explicitados, tem-se agora como extrair os dados para aplicar uma metodologia para comparar se a aplicação da placa de Peltier é vantajosa em relação aos métodos convencionais, quais os pontos negativos e positivos, e analisar se há a possibilidade de melhorar o sistema para proporcionar mais eficiência térmica e energética.

Por consequência, há a necessidade de sincronizar os dados recebidos e enviados entre o computador e o Arduíno, e assim fazer o *hardware* trabalhar de forma satisfatória com os *softwares* elaborados, visando comportamentos responsivos e notáveis na prática e nos resultados.

Portanto, temos a placa de Peltier como o atuador do sistema, recebendo uma potência fornecida por um módulo PWM baseado em um mosfet que se comporta basicamente como um interruptor, o qual liga e desliga a energia fornecida para a placa determinadas vezes por segundo de acordo com a programação, resultando assim na potência real entregue à pastilha. O sensor por sua vez, foi utilizado o interno do processador para ter maior fidelidade com os resultados obtidos, e evitar improvisos, podendo assim atrapalhar o contato entre a placa de Peltier e o encapsulamento do processador.

Os limites utilizados na função *map* se relacionam com curvas de calibração, as quais são usadas para relacionar os valores de entrada e saída de um sistema, permitindo uma correspondência precisa entre os dois. Por exemplo, em um sistema de controle de temperatura, uma curva de calibração pode relacionar a leitura do sensor de temperatura com o valor real da temperatura. Isso permite que o sistema de controle tome decisões com base nas informações precisas do sensor.

Por outro lado, o controle por histerese é uma técnica de controle que leva em consideração a história passada do sistema para tomar decisões de controle. Ele utiliza dois limites ou níveis de referência, conhecidos como "limite superior"e "limite inferior". O sistema alterna entre dois estados de ação com base na posição da variável controlada em relação a esses limites.

A relação entre as curvas de calibração e o controle por histerese ocorre quando a calibração do sistema é usada para determinar os limites ou níveis de referência para o controle por histerese. Por exemplo, no controle de temperatura, a calibração pode fornecer informações sobre os valores ótimos para o limite superior e limite inferior com base na faixa de operação desejada. Esses valores podem ser definidos para garantir que o sistema permaneça dentro dos limites desejados e evite comutações frequentes entre os estados de ação e repouso.

Portanto, as curvas de calibração fornecem as informações necessárias para definir os limites de controle, permitindo que o sistema tome decisões com base em dados confiáveis e precisos, considerando a história passada do sistema. Com isso, o modelo de controle de histerese com um limite superior e inferior é mapeado para que um chaveamento *on-off* atue de formas diferentes e proporcionais para cada valor de temperatura dentro dos limites selecionados.

3 Metodologia

3.1 Fundamentação teórica

3.1.1 Efeito Peltier

Em 1821, o físico estoniano Thomas Johann Seebeck fez uma descoberta inovadora que ficou conhecida como o efeito termoelétrico. Enquanto realizava um experimento para gerar uma diferença de potencial, ele observou o fenômeno que ocorria na junção de dois corpos metálicos diferentes com temperaturas distintas, como cobre e ferro. Quando Seebeck alterava a temperatura dos dois corpos metálicos, ele notou uma mudança no desempenho. A partir do trabalho de Seebeck, o físico francês Jean Charles Athanase Peltier fez uma observação significativa em 1834. Ao controlar a direção da corrente elétrica em relação à força eletromotriz, Peltier descobriu o efeito reverso do experimento de Seebeck, resultando em resfriamento ou aquecimento na junção dos corpos metálicos.

Um modelo de circuito com uma fonte de energia de corrente contínua pode conter semicondutores positivos e negativos. O semicondutor positivo apresenta uma corrente elétrica convencional que flui de temperaturas decrescentes, enquanto o semicondutor negativo mostra uma corrente elétrica convencional que flui em direção a temperaturas crescentes. A combinação desses conjuntos de semicondutores permite a criação de uma bateria termoelétrica ou de um sistema de refrigeração termoelétrica (KAKIMOTO, 2013).

3.1.2 Condutividade térmica

A condução térmica é um processo fundamental pelo qual a energia térmica é transferida dentro de um corpo através de colisões moleculares. No entanto, deve-se observar que esse tipo de condução térmica, embora presente, não é tão significativo quanto a condução térmica que ocorre através do movimento de elétrons de um átomo para outro em todo um material.

Esse fenômeno é devido à estrutura eletrônica dos átomos. Em certos átomos, os elétrons não estão fortemente ligados ao núcleo, o que permite que esses elétrons se separem e se movam para átomos vizinhos. Esses elétrons móveis, frequentemente chamados de "elétrons livres", desempenham um papel crucial na transferência de energia térmica dentro de um material.

Durante a condução térmica, os elétrons livres ganham energia cinética das regiões circundantes de temperatura mais alta. Eles então se difundem através da estrutura cristalina, colidindo com outros átomos e transferindo a energia adquirida para esses átomos a cada colisão. Dessa forma, a energia térmica é efetivamente transportada através do material, permitindo a propagação do calor.

Esse processo de migração de elétrons e transferência de energia é comumente chamado de "condução térmica". Vale ressaltar que a capacidade de um material conduzir calor está intimamente relacionada à sua condutividade elétrica. Materiais com alta condutividade elétrica também tendem a ter boa condutividade térmica, pois a presença de numerosos elétrons livres possibilita uma transferência eficiente de energia.

Por outro lado, materiais com baixa condutividade elétrica geralmente têm condutividade térmica limitada, pois sua mobilidade eletrônica e a consequente transferência de energia são restritas. Essa relação entre condutividade elétrica e térmica é evidente em vários materiais, incluindo metais, que são conhecidos por suas excelentes propriedades de condução térmica e elétrica.

Em resumo, a condução térmica ocorre quando a energia térmica é transferida por meio de colisões moleculares, embora em menor medida do que a condução habilitada pelo movimento de elétrons livres dentro de um material. A presença de elétrons livres e sua capacidade de migrar e transferir energia desempenham um papel fundamental na determinação da adequação de um material como condutor de calor e eletricidade (ROSA et al., 2016).

O conceito de fluxo de calor quantifica a quantidade de calor que atravessa o corpo em um determinado intervalo de tempo. É importante destacar que o fluxo de calor, ou condução, ocorre apenas quando existe uma diferença de temperatura em pontos espacialmente distribuídos ao longo do material. No caso de um fluxo de calor unidimensional em um material isotrópico, a relação entre o fluxo de calor e outras variáveis pode ser expressa matematicamente, incluindo a seguinte relação (ROSA et al., 2016):

$$\Phi = \frac{\Delta Q}{\Delta t} = K \cdot A \cdot \frac{\Delta T}{\Delta x}$$

Significado dos símbolos :
Φ = fluxo de calor [cal/s].
Q = quantidade de calor, medida em Joule [J]
K = condutividade térmica do material [cal/s $\cdot$ cm $\cdot$ °C].
A = área da seção transversal à direção do fluxo, [cm^2]
$\frac{\Delta T}{\Delta x}$ = gradiente de temperatura, ou seja, a razão entre a variação de temperatura e a distância na direção do fluxo de calor.

3.1.3 Outros trabalhos com efeito Peltier

Muitos trabalhos já foram feitos envolvendo o efeito Peltier e utilizando da pastilha de Peltier, e alguns se aproximam do tema como o elaborado por Cássio Santos Oki Okumura, Felipe Felix Santos, Ricardo Alves Pereira Junior e João Alves Bento com o título : Sistema de refrigeração termoelétrica de Peltier usado para arrefecer o processador de computador no curso de engenharia mecânica pela UniEvangélica (OKUMURA et al., 2018).

Neste trabalho os autores tiveram o intuito de analisar a capacidade de resfriamento de um processador usando a placa de Peltier, analisando vários cenários, como um dissipador de alumínio com aletas acoplado à uma ventoinha, conhecido pela expressão em inglês *aircooler*, chapas de cobre de espessuras diferentes e um sistema de resfriamento líquido (*watercooler*), com isso foram realizados vários ensaios utilizando-se de programas que utilizam a capacidade máxima do processador, porém a placa de Peltier ficava sempre em regime de potência máxima, ligada diretamente na fonte de alimentação do computador.

A partir disso os integrantes do trabalho anteriormente citado concluíram que "foi analisado que é possível a aplicação do dispositivo com placa Peltier no sistema de arrefecimento do processador. Porém, foram identificadas as variáveis que podem interferir no valor da resistência térmica dos componentes que compõem o sistema de refrigeração do processador. Como a espessura da chapa de cobre, a altura e material do dissipador aletado, a rotação dos coolers, a área de contato entre os componentes, a qualidade da pasta térmica utilizada e a temperatura externa do gabinete.Por esses motivos, deve-se analisar os melhores componentes para adequar o sistema de arrefecimento sem danificar ou reduzir a vida útil do computador."Transcrição *ipsis litteris* dos autores.

Outro trabalho realizado abordando o tema refere-se a prática realizado por Alisson Luiz, Leandro Cesar e Rafael Dalbello da Universidade Tecnológica Federal do Paraná pelo departamento de eletrônica e tecnologia em automação industrial com o título : Aplicação de pastilhas Peltier para fabricação de hidromel (BUENO; DALOSKI; ALMEIDA, 2018).

Neste trabalho ouve a aplicação da placa de Peltier para controlar a temperatura dentro de um recipiente com o intuito de favorecer assim aspectos desejáveis em bebidas fermentadas, visando um baixo custo de produção e de uso. Porém de acordo com os autores, foi inviável a utilização das placas de Peltier pois há no mercado melhores tecnologias para o mesmo desempenho, e o gasto energético em relação à capacidade térmica não é eficiente em comparação com refrigeradores disponíveis atualmente.

3.2 Planejamento experimental

A fim de observar qual seria o comportamento da placa de Peltier e sua capacidade de resfriar o processador foi feito uma comparação com o mesmo sistema, porém sem a própria placa de Peltier, logo usa-se a parte de dissipação do resfriamento a água, item destacado em amarelo na figura 2, em contato com o processador, como é a instalação convencional.

Todos os ensaios foram realizados utilizando o mesmo sistema, ora utilizando a placa de Peltier e ora não a utilizando, todas as aquisições de dados em 10 minutos, sempre no mesmo ambiente e temperatura local, o estresse do processador foi feito com o mesmo programa (CPU-Z) e todas as análises comparativas obedecendo a mesma ordem para evitar interferência de outras variáveis.

Junto a isso, foi adicionado na entrada de energia da fonte um potenciômetro de modelo P4400 KILL A WATT para obter a potência total do sistema em determinadas situações do experimento.

Figura 22 – Potenciômetro utilizado para medir o gasto energético total do sistema. Autoria própria.

Figura 23 – Etiqueta com especificações do potenciômetro utilizado. Autoria própria.

Com isso pôde-se observar as temperaturas em uma mesma aplicação, e determinar os pontos positivos e negativos da pastilha de Peltier em cada situação, bem como o gasto energético.

4 Resultados e discussão

4.1 Resultados do primeiro ensaio

O primeiro ensaio consiste em um controle com uma maior faixa de temperatura, onde a função *map* do Arduíno mapeia esta faixa devolvendo valores de trabalho para o PWM linearmente relacionados conforme explicado no item 2.3.2.

Com isso tem-se a função map com os parâmetros (0,55,25,255), que se traduz basicamente quanto mais frio o processador está menos energia entrega-se para a pastilha de Peltier, e quanto mais quente mais energia.

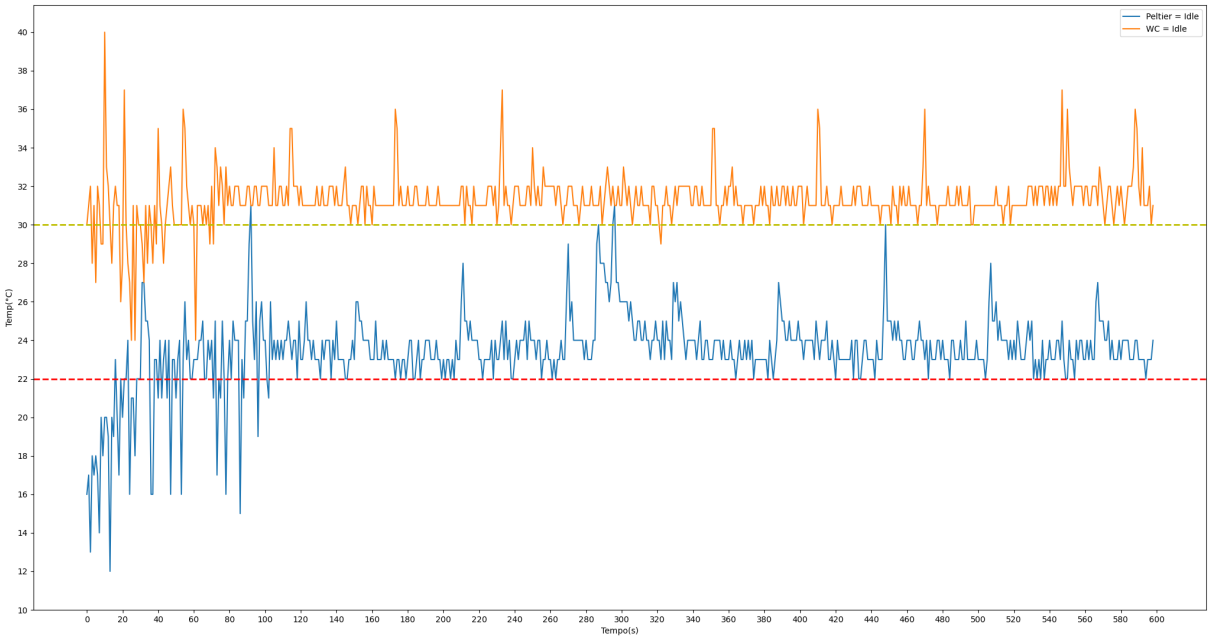

Figura 24 – Gráfico temperatura(°C) x tempo(s) dos dados coletados no primeiro ensaio em *Idle*[1]. Autoria própria.

A imagem 24 se refere à comparação dos sistemas. A curva sem a placa de Peltier de cor laranja, denominado no gráfico de WC (*watercooler*), que se refere ao elemento de troca de calor direto com o processador, neste caso é o WC, e a outra curva na cor azul denominada Peltier a qual é refente ao componente direto de troca de calor, a pastilha de Peltier. A expressão *idle* em inglês se refere a um comportamento ocioso, à espera de uma tarefa ou sobre carga mínima de funcionamento do sistema.

A partir disso, pode-se observar que com a placa de Peltier, após as temperaturas se estabilizarem por volta dos 100 segundos, a curva com a pastilha oscilou entre 22°C com picos em 30°C, e sem a pastilha oscilou entre entre 30°C com picos em 36°C, esses picos

se devem a processos de rotina do sistema que entram rapidamente para serem executados no processador, aumentando assim a carga momentânea e temperatura.

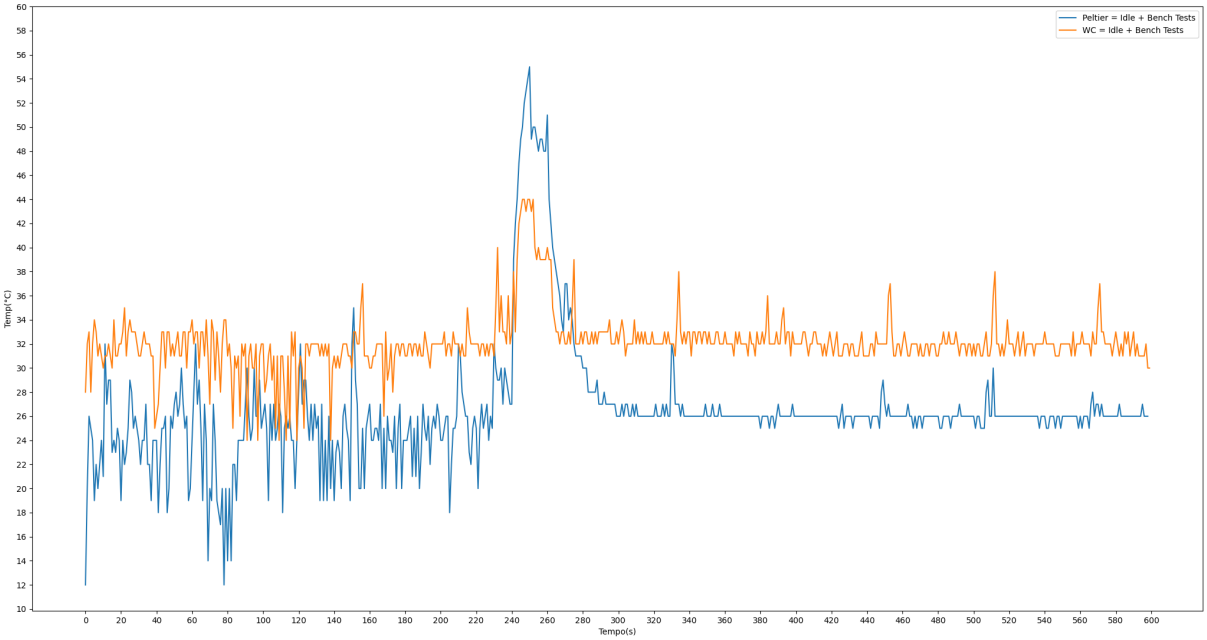

Figura 25 – Gráfico temperatura(°C) x tempo(s) dos dados coletados no primeiro ensaio em *idle* um teste curto de estresse . Autoria própria.

A imagem 25 é referente ao comparativo de um teste rápido de estresse em um determinado período de tempo, inicia-se em estado ocioso, aplica-se o estresse e volta para o estado ocioso, este teste foi efetuado pelo programa CPU-Z. Nota-se que em estado de menor carga, a pastilha se sai melhor na troca de calor com o CPU, porém em maiores cargas a dissipação do calor começa a ficar pior em relação ao não uso da pastilha.

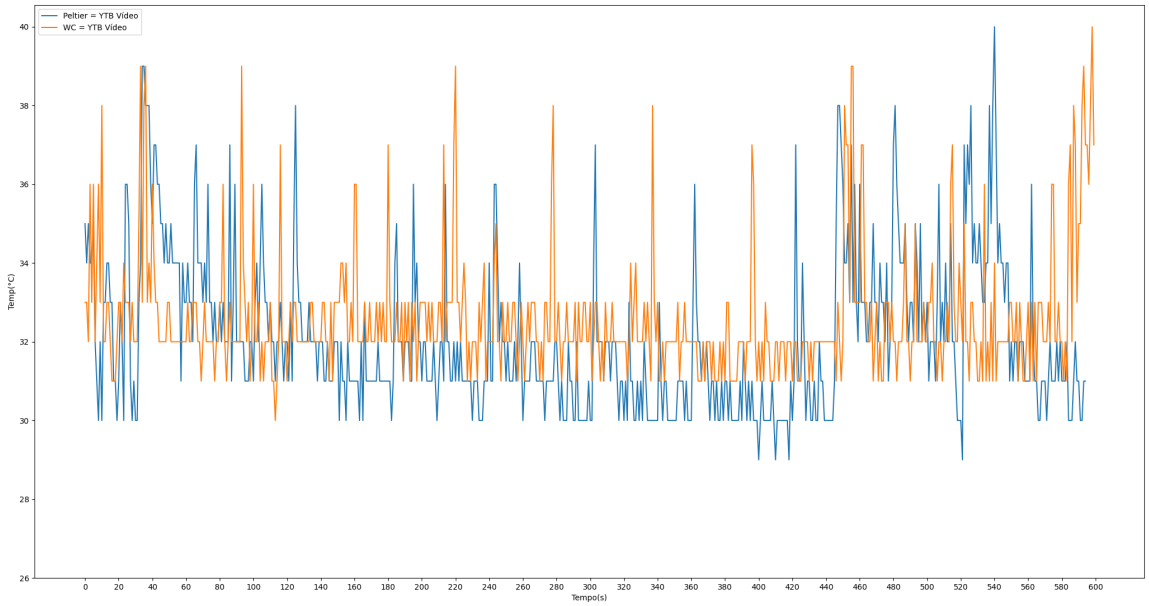

Figura 26 – Gráfico temperatura(°C) x tempo(s) com a carga de um vídeo. Autoria própria.

A figura 26 é referente ao comparativo o qual simula o cotidiano de um usuário, assistindo a um vídeo no Youtube, o mesmo vídeo foi colocado para ambos os testes, na mesma resolução e em tela cheia. Identifica-se que com um controle em uma faixa ampla de temperatura mínima e máxima, o uso da placa de Peltier leva pouca vantagem em relação ao uso convencional.

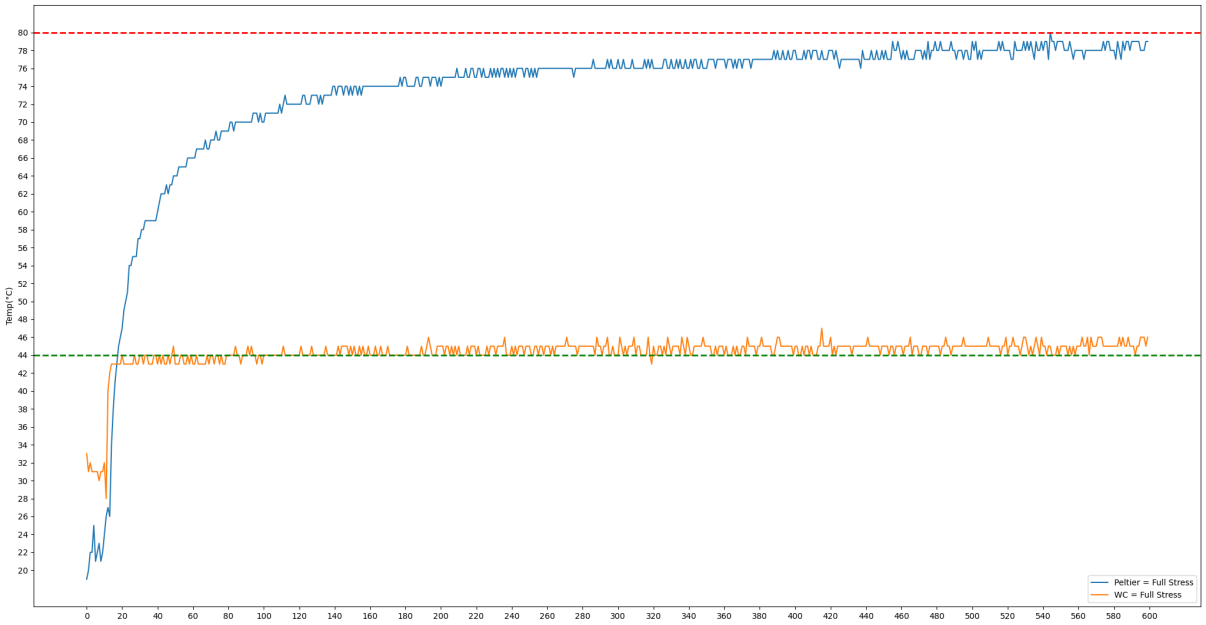

Figura 27 – Gráfico temperatura(°C) x tempo(s) com estresse máximo ao longo do tempo. Autoria própria.

A figura 27 mostra que em usos extremos do CPU, a dissipação da pastilha de Peltier não consegue manter temperaturas baixas em relação ao método convencional, mesmo em regime de potência máxima. A temperatura ultrapassou os 55°C e o PWM ficou com o sinal de 255, demonstrando que a pastilha não tem uma eficiência térmica de troca de calor com o processador comparado com o WC em estado de estresse máximo.

Logo, é importante salientar que o resfriamento a água, no método convencional, está com a superfície de cobre do dissipador, excelente material para dissipar calor, como elemento secundário de transferência de calor, passando pelo elemento primário que é a pasta térmica, no caso do sistema com a placa de Peltier, a pastilha é o elemento secundário, porém ela tem um encapsulamento de porcelana, que é um isolante térmico, desfavorecendo a troca de calor.

Os gráficos para os valores PWM serão equivalentes em relação à temperatura *versus* o tempo, dentro da temperatura mapeada, pois estão linearmente relacionados. O gráfico de PWM *versus* Temperatura discorrerá basicamente em um gráfico de dispersão, relacionando cada valor da temperatura ao seu valor equivalente em PWM numa proporção linear, para cada valor de temperatura da faixa mapeada, há um valor diferente para a modulação *on-off* do PWM.

4.2 Resultados do segundo ensaio

Neste experimento houve um controle com uma menor faixa de temperatura e menor faixa de trabalho para o módulo PWM, alterou-se a função map no Arduíno com os seguintes parâmetros (0,40,100,255), tem-se agora a seguinte função linear para este map $Y(x) = 155/40 \cdot x + 100$. No experimento anterior a função relacionada ao map (0,55,25,255) era $Y(x) = 230/55 \cdot x + 25$, um valor intermediário de temperatura de 22°C no processador, o PWM assumiria o valor de $Y(22) = 230/55 \cdot 22 + 25 \approx 117$, no novo map $Y(22) = 155/40 \cdot 22 + 100 \approx 185$, logo para um mesmo valor de temperatura será fornecido mais potência para a placa de Peltier.

Esse intervalo para a função map foi escolhido com o intuito de manter o processador na faixa de 20°C de temperatura em *idle*, para evitar condensação de água em temperaturas mais baixas. O ajuste para essa faixa foi feito manualmente forçando valores PWM no Arduíno de 10 em 10 para observar em qual faixa aproximada o processador tenderia a manter os 20°C.

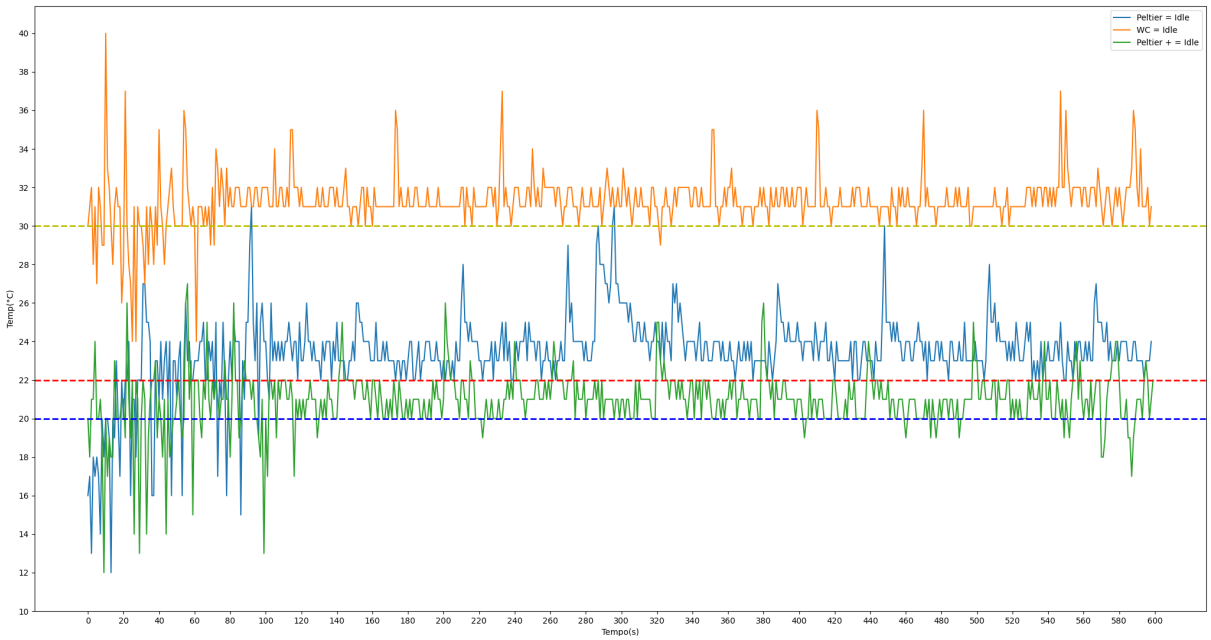

Figura 28 – Gráfico temperatura(°C) x tempo(s) dos dados coletados no segundo ensaio em *Idle*. Autoria própria.

A partir da imagem 28 na curva de cor verde e legenda Peltier +, com uma faixa de trabalho menor, observa-se que houve uma queda na temperatura em relação ao map anterior na cor azul a qual trabalhava com um intervalo de temperatura maior.

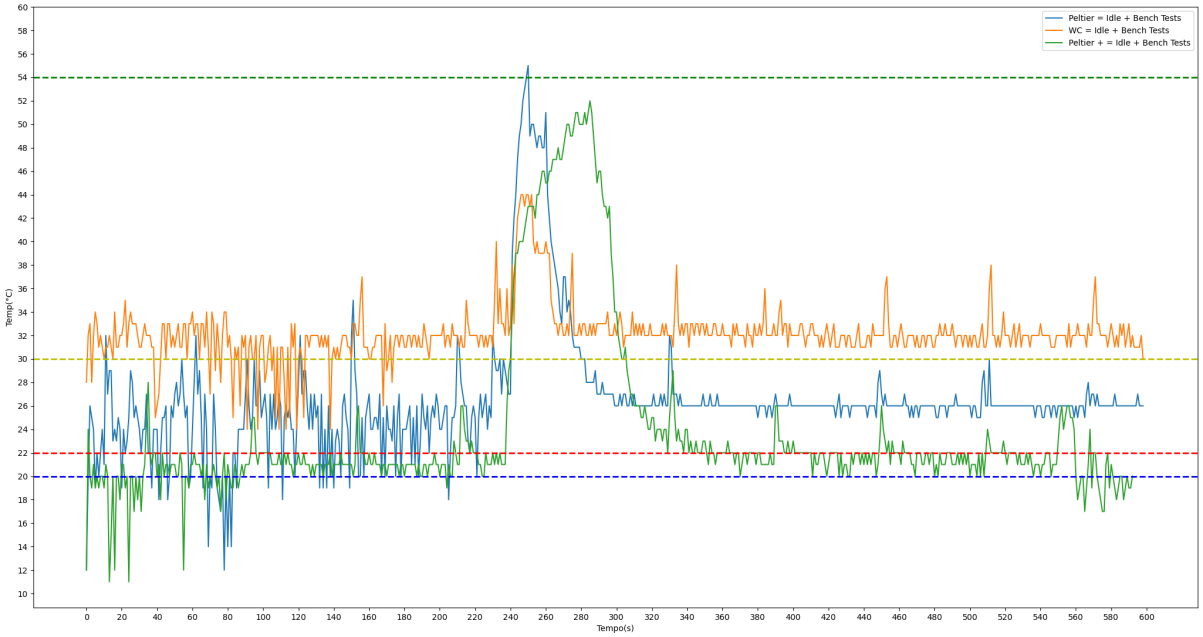

Figura 29 – Gráfico temperatura(°C) x tempo(s) dos dados coletados no segundo ensaio em *idle* um teste curto de estresse . Autoria própria.

Conforme a figura 29 que corresponde a um estresse em um curto intervalo de tempo, mostrou-se que a temperatura aumentou, porém mais devagar, e chegou a um pico menor que o map do ensaio anterior, entretanto ficou termicamente menos eficiente do que a curva em laranja.

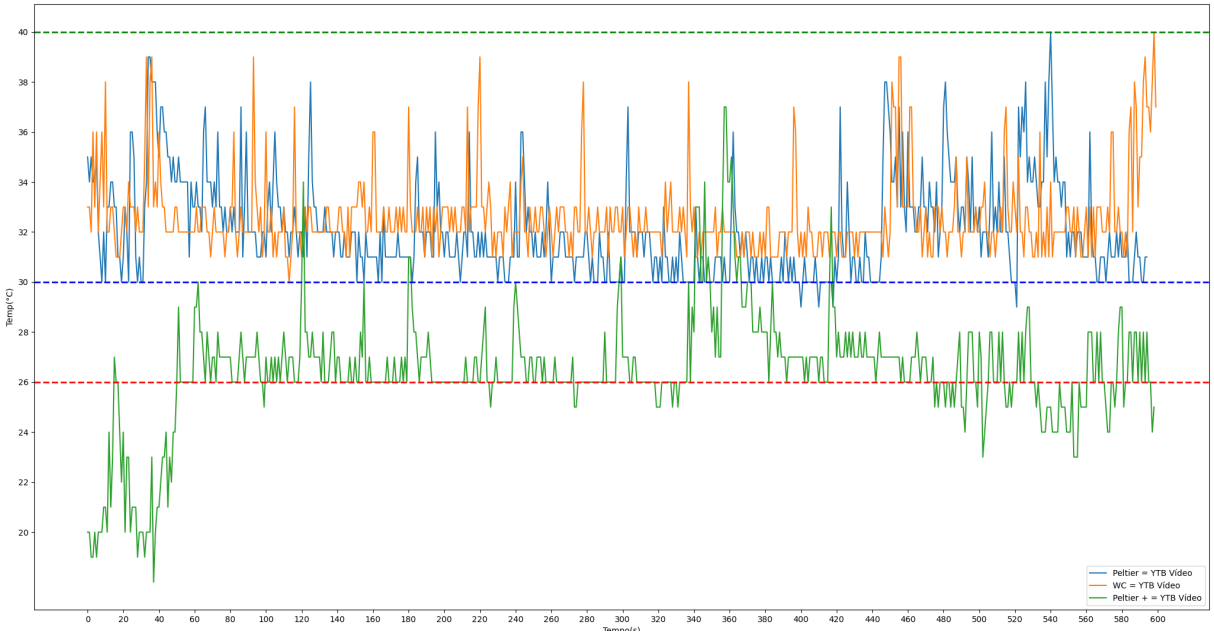

Figura 30 – Gráfico temperatura(°C) x tempo(s) com a carga de um vídeo. Autoria própria.

Com o gráfico da imagem 30 fica exemplificado que para tarefas de pouca e média carga no processador, a pastilha de Peltier se sai bem para manter a temperatura do processador menor.

4.3 Resultado do terceiro ensaio

Por último foi colocado a potência máxima da pastilha de Peltier no processador em estado ocioso, para observar qual a temperatura mínima que ele poderia chegar.

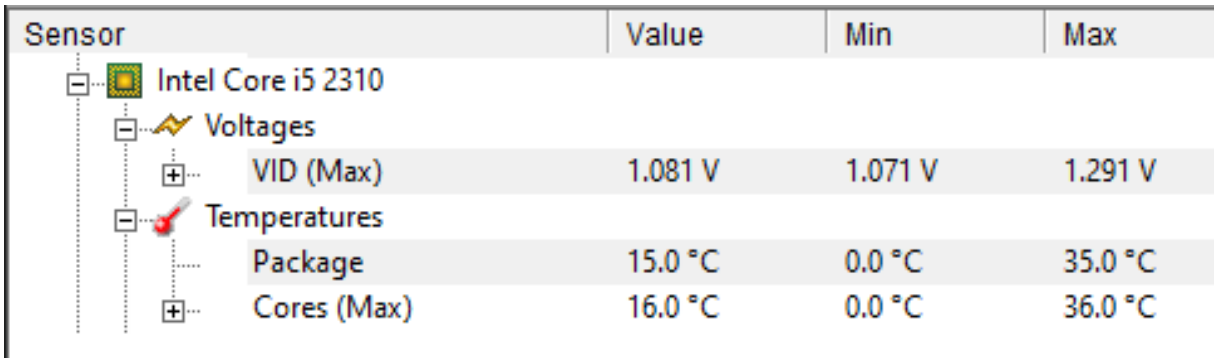

Figura 31 – Temperaturas com o PWM no máximo. Autoria própria.

Na figura 31 pode-se observar que o valor mínimo atingido foi 0°C, porque o sensor interno do processador só identifica até 0 grau, não identifica valores negativos, mas provavelmente se tivesse a escala negativa a temperatura chegaria a níveis menores que zero.

4.4 Análise da potência

Para verificar o consumo, foi utilizado um potenciômetro na entrada da fonte de energia, e a cada passagem por um teste no controle suave, anotava-se de minuto em minuto o valor correspondente na tela do aparelho de medição.

Em média a potência do sistema em *idle* + controle textitmap (0,55,25,255) em relação ao *idle* do sistema sem a placa de Peltier foi de 44 Watts contra 26 Watts. A potência no regime assistindo a um vídeo no Youtube com e sem a placa de Peltier ficou 57 Watts para o sistema com a placa de Peltier e 34 Watts para o sistema sem a pastilha, no estado de estresse máximo foi em torno de 120 Watts para o sistema controlado e 72 Watts para o resfriado somente com o resfriamento líquido.

Com isso, o consumo médio em cada regime foi em torno de 68% a mais de potência utilizando o sistema de controle, no regime ocioso ficou $44/26 \approx 1{,}69$, no sistema assistindo a um vídeo $57/34 \approx 1{,}68$ e no regime em estresse máximo $120/72 \approx 1{,}67$.

No caso do ensaio em regime *idle* no controle do primeiro ensaio, a temperatura era em torno de 30°C com o sistema convencional de resfriamento, e aproximadamente 22°C com o controle, sendo assim tem-se $(22/30 - 1) \approx 27\%$ de melhora térmica a um custo de 68% a mais de energia, evidenciando que energeticamente não compensa.

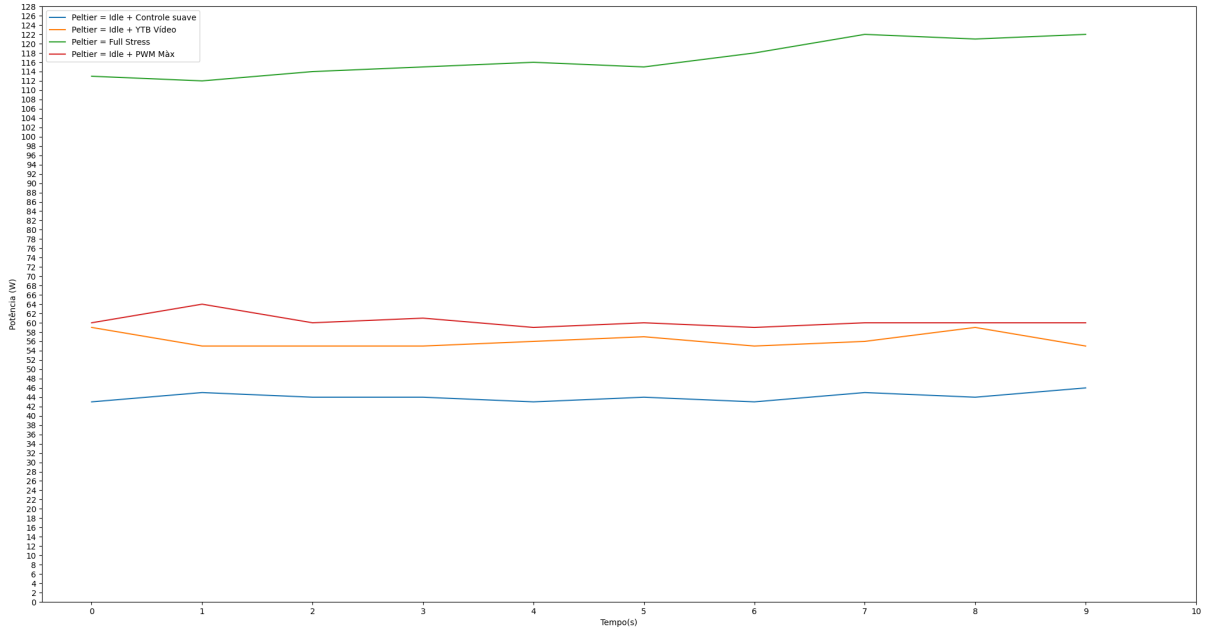

Figura 32 – Potências em cada tipo de situação de uso do processador com a placa de Peltier. Autoria própria.

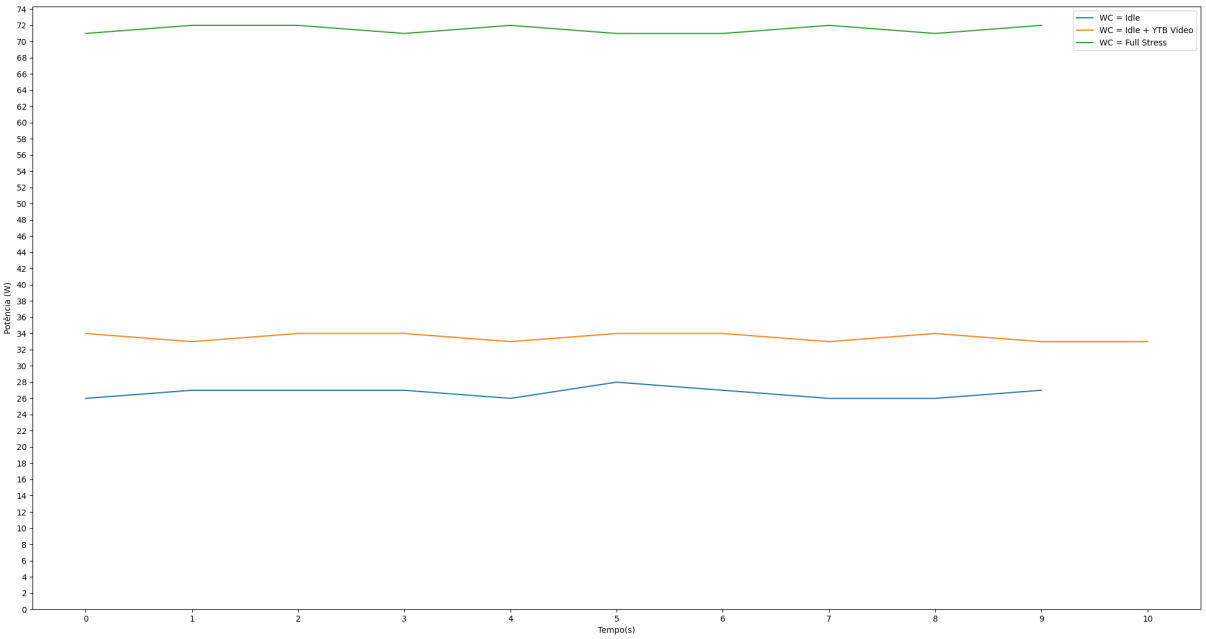

Figura 33 – Potências em cada tipo de situação de uso do processador sem a placa de Peltier. Autoria própria.

4.5 Análise geral

Por meio dos experimentos práticos foi possível analisar o comportamento físico do sistema, um fator intrínseco da placa de Peltier que impactou de forma direta no resultado é o fato da porcelana ter a condutividade térmica $\mathbf{K}$ [cal/s · cm · °C] $\approx 0{,}0011$ enquanto o cobre tem $\approx 0{,}922$, ou seja ≈ 830 vezes melhor condução térmica. A partir disso verifica-se que a porcelana é um mau condutor térmico, enquanto o cobre é um excelente condutor.

Com a observação supracitada ficou evidente que encostar um material gelado em outro mais quente não necessariamente é mais eficiente na condução térmica, é necessário estudar e aproveitar das características de condução térmica de cada material. Uma conclusão em conjunto com o observado pelo trabalho realizado sobre o tema : Sistema de refrigeração termoelétrica de Peltier usado para arrefecer o processador de computador (OKUMURA et al., 2018), foi que o método convencional ainda é mais eficiente no geral para ajudar a arrefecer um processador.

Em relação ao item 4.4 a respeito da análise de potência consumida para um determinado resultado, também tem-se um resultado de eficiência energética condizente com a prática sobre o tema : Aplicação de pastilhas Peltier para fabricação de hidromel (BUENO; DALOSKI; ALMEIDA, 2018) em que a placa de Peltier ainda é ineficiente energeticamente em relação a outros sistemas disponíveis no mercado.

5 Conclusões e sugestões

5.0.1 Conclusões

Por meio dos experimentos práticos e no andamento do projeto já foram notados os desafios para tornar o sistema viável, ao conectar a placa de Peltier diretamente à potência máxima fornecida pela fonte e ver o processador chegando a 0°C, e com a possibilidade de ter atingido temperaturas negativas, houve uma expectativa de conseguir resultados excelentes para todo o uso do processador. Com isso, ao longo dos testes e aumentando a carga do processador foram notadas características físicas do sistema as quais poderiam ser melhoradas, principalmente a questão de transferência de calor da placa de Peltier para o dissipador de cobre do *watercooler*.

Em uso baixo e médio de carga no processador, caso o usuário tenha um processador que trabalhe com temperaturas mais altas, como características do modelo e arquitetura, o sistema apresentado é uma viabilidade para deixar as temperaturas mais baixas, principalmente se o usuário possuir um modelo de arrefecimento do tipo *aircooler*, que costuma ser ainda menos eficiente que o resfriamento a água, principalmente os modelos simples que são originais de fábrica e acompanham os processadores. A prática deste trabalho teve por decisão já inicial comparar a dissipação por meio da placa de Peltier com um sistema mais eficiente no mercado, por isso escolhido um resfriamento a água, caso o sistema fosse um convencional com ventoinhas e dissipadores somente, haveria algum ganho a mais em relação ao *watercooler*.

Á medida que um uso mais extremo de um processador é exigido pelo usuário, melhorias no sistema devem ser implementadas para analisar a factibilidade do uso no cotidiano, mas em correlação ao esquema proposto nesta prática o uso do modelo convencional de resfriamento a água é ainda termicamente e energeticamente mais eficiente.

O aprendizado em relação à programação e pesquisa para a efetividade do estudo foi de muito proveito, etapas desenvolvidas foram um desafio para tornar o conjunto funcional, como sincronizar os dados do computador com o Arduíno. Embora haja material disponível em muitas fontes, na prática é um desafio, pois sempre há erros, problemas de *software*, incompatibilidades, no exemplo da fonte funciona mas na prática não, logo se estuda e extrai o que é útil e continua a pesquisa para o problema apresentado, e principalmente problemas que acontecem que nem são pensados inicialmente. A função *substring* no arduíno como um exemplo foi utilizada para sincronizar os dados recebidos do computador, algo que parece simples, inicialmente foi pensado em um laço de repetição, mas não foi possível sincronizar de nenhuma forma, foram horas de pesquisa por uma função que conseguiria colocar os dados de forma interpretável para a lógica.

5.0.2 Sugestões de continuidade

Melhorias e maneiras diferentes podem ser feitas para tornar o sistema mais eficiente, inicialmente tinha se pensado em desenvolver um sistema de controle digital e condicional, mas ao decorrer das práticas, notou-se que estabelecer um *setpoint* seria um desafio, pois a placa não teria potência e comportamento o suficiente para atingi-lo e iria saturar em um modelo de uso real. Ao aplicar o estresse máximo no processador e de acordo com a figura 27 haveria um comportamento a uma entrada degrau na curva em azul, e a partir disso obteria um modelo que é de fácil ajuste no arduíno, com a função da biblioteca PID_v1.

Este modelo de controle será implementado futuramente, pois há modelos robustos da placa de Peltier com encapsulamento metálico, com isso o problema com a transferência de calor com a porcelana será atenuado, podendo trazer respostas mais rápidas e factíveis a um sistema de controle PID, junto a isso também, outras placas de Peltier podem ser colocadas em contato com o radiador de água do resfriamento líquido, as quais iriam ajudar a refrigerar a pastilha em contato com o processador, ajudando assim a diminuir a temperatura da água no radiador.

Referências

BUENO; DALOSKI; ALMEIDA. *Aplicação de pastilhas Peltier para fabricação de hidromel*. 2018. Disponível em: <http://repositorio.utfpr.edu.br/jspui/bitstream/1/16952/1/PG_COELE_2018_2_03.pdf>. Acesso em: 26 de Maio de 2023. Citado 2 vezes nas páginas 23 e 32.

IFSC. *AULA 6 - Microcontroladores - Técnico*. 2018. Disponível em: <https://wiki.ifsc.edu.br/mediawiki/index.php/AULA_6_-_Microcontroladores_-_T%C3%A9cnico>. Acesso em: 18 de junho de 2023. Citado 2 vezes nas páginas 5 e 9.

KAKIMOTO, L. C. *Efeito Peltier-Seebeck: gerando eletricidade por diferença de temperatura*. 2013. Disponível em: <https://www.ifi.unicamp.br/~lunazzi/F530_F590_F690_F809_F895/F809/F609_2013_sem1/LuisC_Siervo_F609_RF3.pdf>. Acesso em: 21 de Maio de 2023. Citado na página 21.

OKUMURA et al. *Sistema de refrigeração termoelétrica de Peltier usado para arrefecer o processador de computador*. 2018. Disponível em: <http://45.4.96.19/bitstream/aee/478/1/1_Grupo_1_Cassio_Felipe_Ricardo.pdf>. Acesso em: 26 de Maio de 2023. Citado 2 vezes nas páginas 23 e 32.

RAJGURUELECTRONICS. *Specification of Thermoelectric Module*. 2023. Disponível em: <https://www.rajguruelectronics.com/Product/1522/thermoelectric-cooler-peltier-module-tec1-12715.pdf>. Acesso em: 18 de junho de 2023. Citado na página 8.

ROSA et al. *Experimento de condução térmica com e sem uso de sensores e arduíno*. 2016. Disponível em: <https://periodicos.ufsc.br/index.php/fisica/article/view/2175-7941.2016v33n1p292/31590>. Acesso em: 24 de Maio de 2023. Citado na página 22.

WAYINTOP. *WayinTop 4 x Módulo de controlador Mosfet FET PWM*. 2020. Disponível em: <https://www.amazon.es/WayinTop-controlador-Potencia-0-20KHz-Terminal/dp/B08GFD7F9V>. Acesso em: 18 de junho de 2023. Citado 2 vezes nas páginas 5 e 8.

YHAEFFNER. *Esquema de uma pastilha de Peltier*. 2016. Disponível em: <https://pt.wikipedia.org/wiki/Efeito_Peltier#/media/File:Esquema_Pastilha_de_Peltier.svg>. Acesso em: 18 de junho de 2023. Citado 2 vezes nas páginas 5 e 7.