UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ANA PAULA PRINCIVAL MACHADO
MATHEUS HENRIQUE SILVEIRA SANTANA

CONTROLADOR DE TEMPERATURA COM AQUECIMENTO E RESFRIAMENTO

CURITIBA
2023

ANA PAULA PRINCIVAL MACHADO
MATHEUS HENRIQUE SILVEIRA SANTANA

CONTROLADOR DE TEMPERATURA COM AQUECIMENTO E RESFRIAMENTO

Relatório final apresentado ao curso de Graduação em Engenharia Elétrica, Setor de Tecnologia, Universidade Federal do Paraná, como requisito para nota parcial para a disciplina de TE331 – Instrumentação Eletrônica.

Prof. Dr. Marlio Bonfim.

CURITIBA
2023

SUMÁRIO

1 - INTRODUÇÃO

A mensuração das grandezas físicas é de suma importância no cotidiano, pois atualmente o controle sobre elas é essencial para a manutenção da vida humana. Durante todo o dia estamos em direto contato com sistemas controlados, seja o forno elétrico para assar um bolo em determinada temperatura ou até mesmo o piloto automático do carro para conduzir em uma certa faixa de velocidade. A leitura de temperatura, velocidade, distância, luminosidade e outras grandezas é feita por meio de sensores: componentes desenvolvidos com certas especificações que são capazes de indicar a variação do ambiente que foram projetados para ler.

Assim, o presente projeto estuda particularmente a leitura de um sensor de temperatura, que é submetido a um sistema de aquecimento/resfriamento. A leitura do sensor é usada para limitar a faixa de temperatura do sistema, de -5 ºC até 75º. Tal variação de temperatura é provocada por uma pastilha de Efeito Peltier, que é controlada por meio de uma Ponte H: a inversão do sentido da corrente alterna o lado quente/frio da pastilha. Também é implementado um transistor a fim de controlar o PWM e, com isso, a intensidade da corrente.

Posteriormente a montagem do hardware e da certificação de baixo ruído, é desenvolvido um sistema de controle. Tanto o resfriamento quanto o aquecimento são de segunda ordem e necessitam de um controle PID. Dessa forma é necessária a identificação do sistema e o cálculo das constantes, a fim de que a pastilha atinja a temperatura indicada como SetPoint com a mínima oscilação possível.

2 – DESENVOLVIMENTO TEÓRICO

2.1. A instrumentação eletrônica

É uma área que explora o manuseio de equipamentos eletrônicos, sobretudo aqueles utilizados para ler e processar as grandezas físicas do mundo real. A partir dessa leitura pode-se efetuar um novo condicionamento, acionar um atuador e dessa forma obter-se um controle sobre o ambiente em estudo [1].

A eletrônica permite uma fácil manipulação do sinal elétrico (como amplificá-lo e filtrá-lo) e também um fácil processamento, o que possibilita a conversão analógica-digital ou digital-analógica [1].

2.2. Sensores e transdutores

Os sensores são dispositivos que são capazes de ler o ambiente a qual estão submetidos, convertendo o sinal de determinada grandeza (temperatura, pressão, umidade, luminosidade, velocidade, etc.) [1]. Os sensores são os elementos que leem o sinal de entrada do ambiente e o convertem em sinal elétrico, enquanto os atuadores são os que convertem o sinal elétrico em uma grandeza física [1].

No presente trabalho, o sensor utilizado é o de temperatura (converte a temperatura em sinal elétrico), enquanto o atuador é a pastilha de elemento Peltier (converte a tensão/corrente aplicadas em calor).

2.3. Temperatura

A temperatura é uma grandeza física envolvida em vários processos do cotidiano, e por isso muitas vezes é necessário mensurá-la e obter determinado controle sobre ela.

A temperatura é medida de acordo com o grau de agitação das moléculas de um material. Quanto maior a agitação, maior é a temperatura. Teoricamente, o estado de repouso absoluto das moléculas ocorreria em 0 K, mas fisicamente (pela Terceira Lei da Termodinâmica) isso é impossível [2].

2.3.1. Escala Fahrenheit

Foi desenvolvida por volta de 1700 por um físico alemão, que usou como referência para zero º F um gelo com cloreto de amônia, e o corpo humano como referência para 100 ºF. Essa escala possui maior variação do que a escala Celsius, por exemplo, já que 0 ºC corresponde a 32 ºF e 100 ºC corresponde a 212 ºF [2]. Atualmente é utilizada nos países de cultura inglesa.

2.3.2. Escala Celsius

Foi desenvolvida em 1742 por um físico sueco e é empregada em grande parte do mundo, que utiliza como referência mínima (0 ºC) a temperatura de congelamento da água em condições normais de pressão e como referência máxima a temperatura da ebulição da água (100 ºC) [2].

2.3.3. Escala Kelvin

Foi desenvolvida pelo físico inglês William Thompson (Lord Kelvin) por volta de 1850. Ele utilizou a escala Celsius para mensurar o mínimo de agitação de uma molécula (0 K), e a escala Kelvin também é conhecida como escala absoluta [2].

Em comparação, 0 ºC = 273 K e 100 ºC = 373 K [2].

2.3.4. Conversão de escalas

A escala Kelvin é utilizada como unidade de temperatura no Sistema Internacional e muitas vezes é necessário converter a escalas. Dessa forma, a conversão pode ser feita por meio de

$$\frac{TC}{5} = \frac{TF-32}{9} = \frac{TK-273}{5} \quad (1)$$

onde *TC* é a Temperatura Celsius, *TF* é a Temperatura Fahrenheit e *TK* é a Temperatura Kelvin.

3 – OBJETIVOS

3.1. Objetivo Geral

O objetivo geral do trabalho consiste em fazer um sistema de controle de temperatura com a pastilha Peltier, na faixa de 5ºC até 75ºC, por meio da leitura do sensor de temperatura TMP36 e do acionamento da ponte H.

3.2. Objetivos Específicos

- Fazer a conversão analógico-digital do sensor por meio de código no microcontrolador Arduino, exibindo os valores na tela do computador;
- Encontrar o ponto ótimo de operação da pastilha Peltier e acioná-la por meio da ponte H, juntamente com o controle PWM, analisando e eliminando ruídos;
- Fazer um controle PID do aquecimento e do resfriamento, a fim de que a pastilha seja mantida na temperatura desejada com a mínima variação possível.

4 – FUNDAMENTAÇÃO TEÓRICA DOS MATERIAIS

4.1. O controlador: Arduino

O controlador usado no projeto é o Arduino UNO. É uma plataforma de prototipagem eletrônica, que utiliza o microcontrolador da família Atmel AVR e linguagem de programação baseada em C/C++ [3]. O microcontrolador é o responsável por executar os programas e avaliar as entradas e as saídas, enviando e recebendo informações externas [3].

No Arduino UNO, o microcontrolador é o ATmega328p, cuja frequência de trabalho é 16 MHz. Possui 14 pinos I/O, sendo 6 analógicos e 6 com a função *Pulse Width Modulation* (PWM). Ela tem 32 KB de memória *flash*, e se conecta ao computador via USB [3]. A Fig. 1 mostra o microcontrolador, e a Fig.2 mostra a placa Arduino UNO.

Fig. 1 – Microcontrolador ATmega328p da placa Arduino UNO

Fonte: FilipeFlop [3]

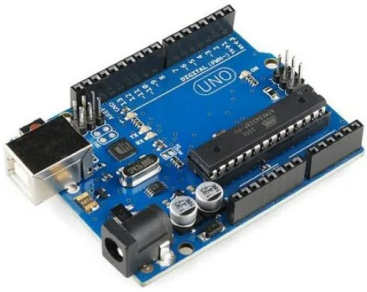


Fig. 2 – Placa Arduino UNO

Fonte: FilipeFlop [3]

4.2. O sensor de temperatura

Inicialmente, o sensor de temperatura proposto para a realização do trabalho foi o LM35, que é um componente analógico de fácil manuseio e implementação, exemplificado na Fig. 3. No entanto, ao testá-lo, verificou-se uma instabilidade em sua operação: valores que oscilavam muito a cada medição e uma leitura equivocada. Mesmo com duas unidades do componente, ambas apresentaram as mesmas falhas, que foram comprovadas em testes com o osciloscópio (verificando a presença de muito ruído no sinal).

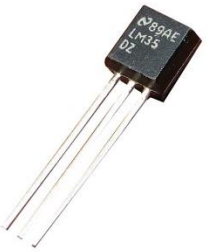


Fig. 3 – LM35

Fonte: Vida de Silício [4]

Então iniciou-se uma busca por um novo sensor que atendesse à demanda do trabalho. Em um primeiro momento foi testado o sensor digital Dallas 18b20 (mostrado na Fig. 4), que estava sendo utilizado como comparativo durante os testes com o LM35. Então também foi testado, por indicação, o sensor TMP36, um componente analógico muito semelhante ao LM35, como pode ser visto na Fig. 5. Testando os novos componentes digital e analógico sob as mesmas condições, foi verificado que o TMP36 apresentava melhor variação, atingindo valores mais altos ou mais baixos em curto intervalo de tempo. Dessa forma, este foi escolhido como sensor principal para a execução do trabalho.

Fig. 4 – Dallas 18b20

Fonte: Casa da Robótica [5]

Fig. 5 – TMP36

Fonte: Smart Kits [6]

4.2.1. Sensor TMP36

O TMP36 é um sensor analógico do tipo circuito integrado, com encapsulamento TO-92, aparentando um transistor de 3 terminais. É dito com alta precisão, funcionando entre 2,7 V até 5,5 V, fornecendo a tensão de forma linearmente proporcional à temperatura em graus Celsius, numa faixa de -40 ºC até 125 ºC. Ele não precisa de calibração, pois já possui uma precisão de 1º a 25 ºC e mais ou menos 2º em toda a sua faixa. Possui saída analógica e cada 10mV representa 1 ºC [7].

A Fig. 6 mostra a pinagem do sensor, de forma a orientar sua ligação correta, em que (visto de frente), os terminais correspondem à alimentação, à saída do sinal e ao GND.

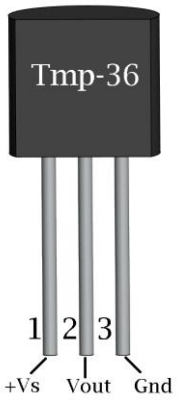


Fig. 6 – Terminais do sensor

Fonte: Components 101 [8]

No *datasheet* do componente pode-se encontrar um circuito equivalente para a família dos sensores TMP3x, como mostrado na Fig. 7. De acordo com o documento, no coração do sensor há um núcleo de banda, composto pelos transistores Q1 e Q2, polarizados por Q3 para aproximadamente 8 µA. O núcleo de *gap* de banda opera tanto Q1 quanto Q2 no mesmo nível de corrente do coletor. No entanto, como a área do emissor de Q1 é 10 vezes maior que a de Q2, a *VBE* de Q1 e a *VBE* de Q2 não são iguais pela seguinte relação [9]:

$$\Delta V_{BE} = V_T * \ln \frac{Ae,Q1}{Ae,Q2}. \quad (2)$$

Os resistores R1 e R2 são usados para dimensionar esse resultado para produzir a característica de transferência de tensão de saída do sensor. R2 e R3 são usados para dimensionar o *VBE* de Q1 como um termo de deslocamento em Vout [9].

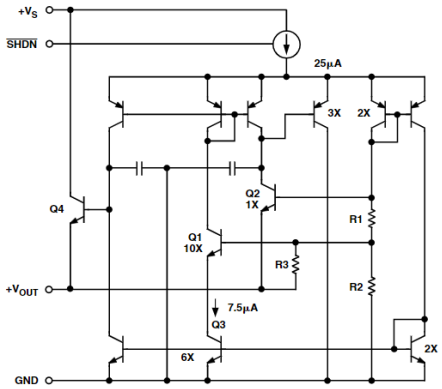


Fig. 7 – Circuito equivalente simplificado do sensor de temperatura

Fonte: Datasheet TMP36 [9]

4.3. Elemento Peltier

Como atuador do projeto é usado o elemento Peltier, que é uma pastilha com a tecnologia Peltier que possui um lado que esquenta e um lado que esfria. Quando ligada, o processo de aquecimento e resfriamento ocorrem simultaneamente, e os lados quente e frio podem ser alterados ao inverter a polaridade da tensão de alimentação [10].

O elemento é energeticamente eficiente, e de acordo com a referência [11], funciona da seguinte maneira: quando a tensão é aplicada, parte da energia é convertida em calor, de forma que um lado fica frio e outro fica quente. O lado de resfriamento fica frio à medida que a energia é extraída dele por meio da corrente elétrica, enquanto o lado do aquecimento fica quente ao liberar o calor extraído do lado frio para o ambiente [11]. A Fig. 8 exemplifica o funcionamento Peltier (na prática do nosso projeto as potências são bem menores do que as descritas na figura).

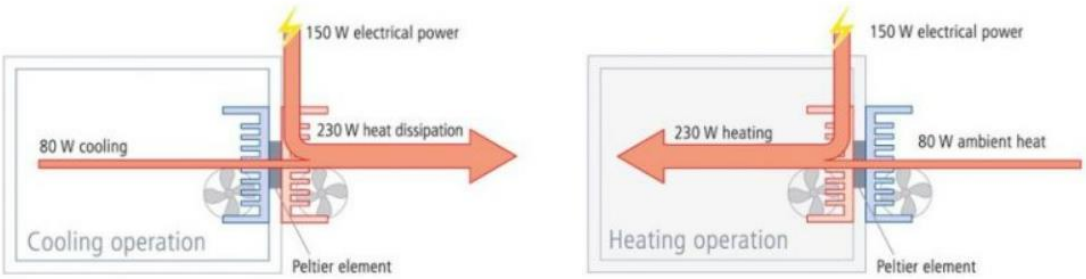


Fig. 8 – Funcionamento da Peltier

Fonte: Steq [11]

4.4. A Ponte H

Para controlar a reversão da polaridade da pastilha Peltier é utilizada uma ponte H, comumente usada para alterar o sentido de giro de um motor de corrente contínua.

O circuito mais simples da Ponte H é mostrado na Fig. 9, onde a corrente flui da esquerda para a direita se CH1 e CH4 forem fechadas, e flui da direita para a esquerda se CH3 e CH2 forem fechadas [12].

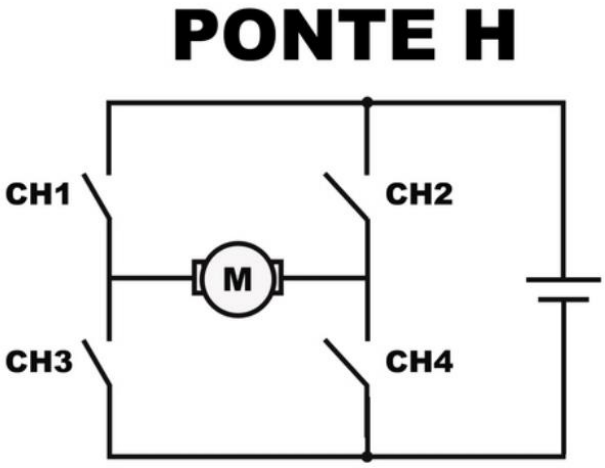


Fig. 9 – Funcionamento simples da ponte H

Fonte: Manual da Eletrônica [12]

Existem diferentes formas de realizar a Ponte H, sendo a mais comum com transistores. No entanto, sob orientação do professor, foi optado por realizá-la com relés, a fim de poder operar o sistema com maiores correntes.

4.5. PWM – *Pulse Width Modulation*

É uma técnica utilizada quando o ato de “ligar” ou “desligar” portas e componentes não satisfaz a aplicação, pois é necessário controlar a intensidade do atuador [13].

O PWM é um sinal totalmente digital, pois codifica níveis de sinal analógico e só apresenta saídas como totalmente ligado ou completamente desligado. Sua forma de atuar é pelo tempo em que a saída está em alta, denominada pulso. A taxa de ciclo é definida como (*pulso*/*ciclo*)*100, enquanto a frequência de modulação é definida por (*ciclos*/*segundo*)*100 [13].

5 – MATERIAIS E MÉTODOS

5.1. Conexão do sensor TMP36 ao Arduino

A conexão do sensor ao microcontrolador Arduino é feito de forma simples, como mostrado na Fig. 10, usando a porta analógica A0 para receber o sinal.

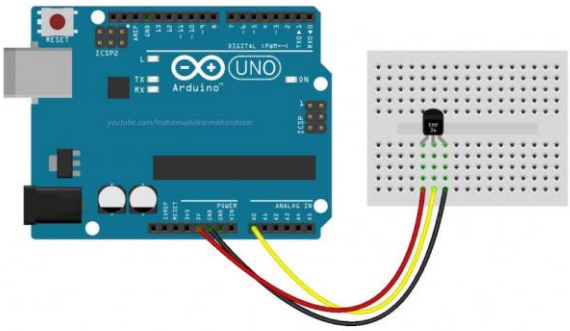


Fig. 10 – Conexão TMP36 ao Arduino

Fonte: Project Hub [10]

O código para a execução é mostrado na Fig. 11, também incluído no Anexo I. De modo simplificado, lemos o sinal do sensor pela porta A0 (linha 8), então multiplicamos por 5, pois estamos conectando o sensor na alimentação de 5 V (linha 9), e então dividimos por 1024 para a conversão AD. Na linha 12 há a subtração de 0,5 e a multiplicação por 100 para ajustar a faixa de medição do sensor para a temperatura em Celsius, que então é exibida na tela. O código foi extraído da referência [14], com suporte pelas referências [15] e [16].

TMP36 §

```
#define Pino A0 //Define a porta A0 como entrada

//Esquerda: VCC, meio: sinal, direita: GND

void setup() {
Serial.begin(9600);
}

void loop() {
int leitura = analogRead(Pino); //Lê a entrada
float tensao = leitura * 5; //multiplica por 5V
tensao /= 1024.0; //Conversão AD

float Temperatura = (tensao - 0.5) *100; //Subtraímos 0,5 para ajustar a faixa de medição do sensor
Serial.print(Temperatura); //Printa a temperatura
Serial.println(" Graus");
Serial.print(leitura);
Serial.println(" leitura");
delay (1000);
}
```

Fig. 11 – Código para a leitura do sensor

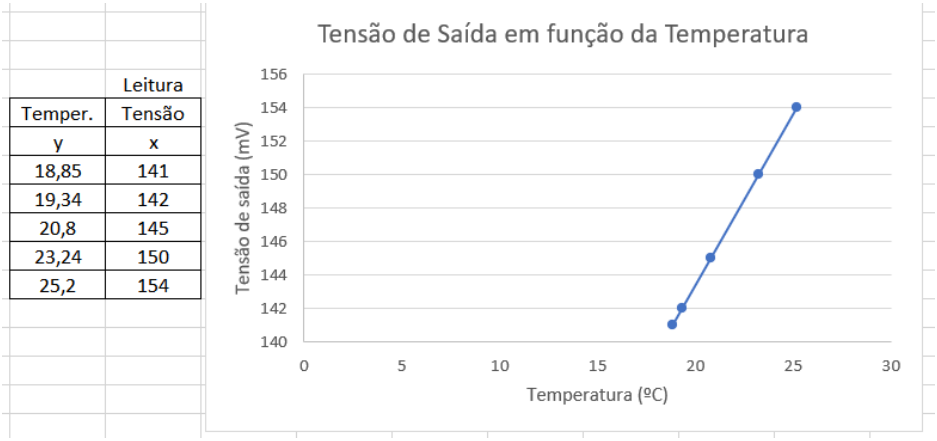
Fonte: Referência [14]

Dessa forma, a conversão pode ser resumida pela equação

$$Temperatura = \left(\frac{leitura*5}{1024} - 0{,}5\right) * 100 \quad (3)$$

sendo esta a equação principal do funcionamento do sensor.

O código foi testado, variando a temperatura próxima ao sensor (simplesmente aproximando a mão) e foram coletados alguns pontos do *monitor serial*. Estes foram plotados em um gráfico no Excel, como mostra a Fig. 12.



	Leitura
Temper.	Tensão
y	x
18,85	141
19,34	142
20,8	145
23,24	150
25,2	154

Fig. 12 – Tensão de saída em função da temperatura (5 V)

Fonte: Os autores

No entanto, sob sugestão do professor, foi feito um teste alterando a tensão de referência interna do Arduino para 1,1 V, a fim de buscar por uma melhor precisão. Isso é feito como mostrado na Fig. 13, onde é acrescentada a função *analogReference(INTERNAL)* na linha 8, e mudando a multiplicação para 1,1 V, na linha 10.

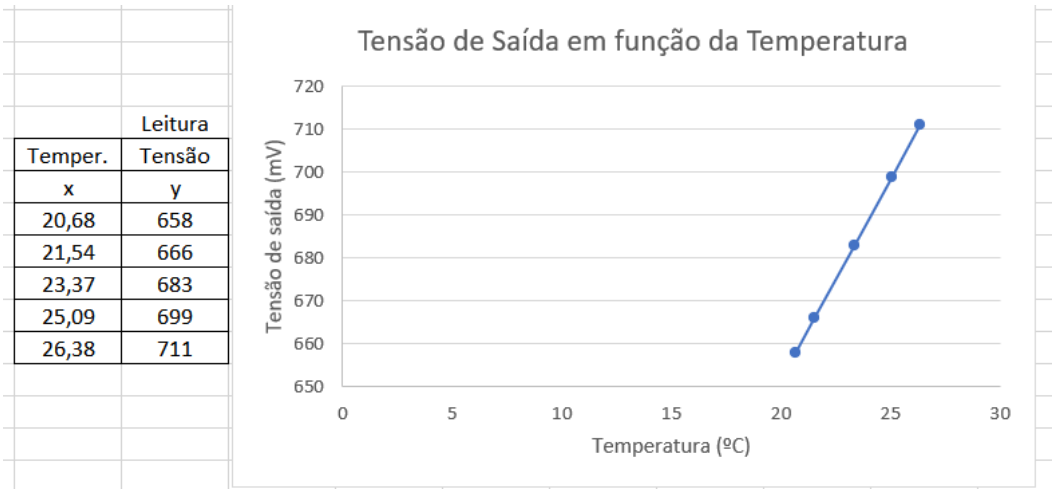
TMP36 §

```
#define Pino A0 //Define a porta A0 como entrada

//Esquerda: VCC, meio: sinal, direita: GND

void setup() {
Serial.begin(9600);
}

void loop() {
analogReference(INTERNAL); //Muda a V interna de ref do Arduino para 1,1V
int leitura = analogRead(Pino); //Lê a entrada
float tensao = leitura * 1.1; //multiplica por 1,1V
tensao /= 1024.0; //Conversão AD

float Temperatura = (tensao - 0.5) *100; //Subtraímos 0,5 para ajustar a faixa de medição do sensor
Serial.print(Temperatura); //Printa a temperatura
Serial.println(" Graus");
Serial.print(leitura);
Serial.println(" leitura");
delay (1000);
}
```

Fig. 13 – Mudança na referência interna de tensão

Fonte: Os autores

Da mesma forma, foram plotados alguns pontos no Excel ao variar a temperatura próxima ao sensor, como mostrado na Fig. 14.



	Leitura
Temper.	Tensão
x	y
20,68	658
21,54	666
23,37	683
25,09	699
26,38	711

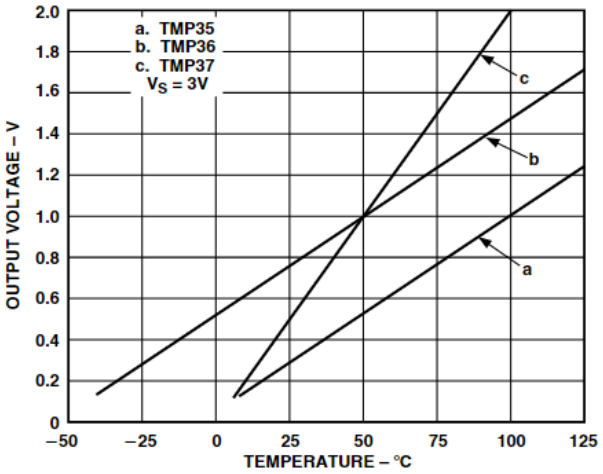
Fig. 14 – Tensão de saída em função da temperatura (1,1 V)

Fonte: Os autores

Neste caso, a equação (2) é adaptada para

$$Temperatura = \left(\frac{leitura * 1{,}1}{1024} - 0{,}5\right) * 100. \quad (4)$$

De forma geral, a curva de comportamento do TMP36 pode ser observada na Fig. 15, apresentada em seu *datasheet* [9], semelhante aos gráficos obtidos experimentalmente.



TPC 1. Output Voltage vs. Temperature

Fig. 15 – Tensão de saída em função da temperatura

Fonte: *Datasheet* TMP36 [9]

Na Fig. 16 é possível ver a temperatura no *Serial Plotter* da IDE do Arduino, onde variamos a temperatura próxima do sensor alternando entre um copo com água quente e um cubo de gelo por alguns instantes, a fim de alcançar temperaturas mais altas e mais baixas.

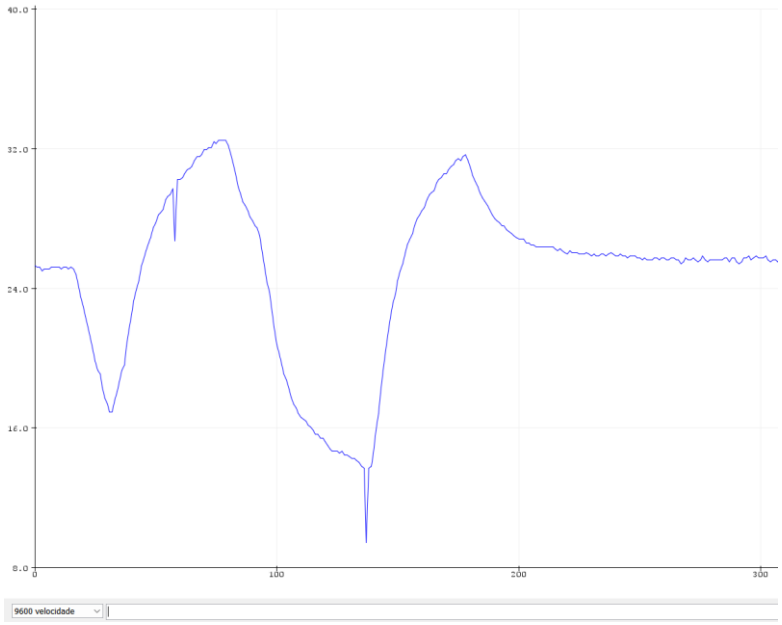


Fig. 16 – Temperatura no *Serial Plotter*

Fonte: Os autores

A Fig. 17, por fim, mostra a montagem na protoboard onde o sensor foi testado.

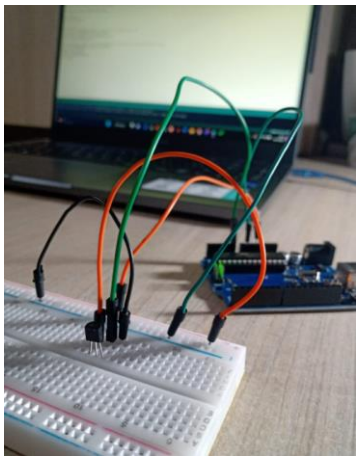


Fig. 17 – Montagem para os experimentos

Fonte: Os autores

5.2. Testes com o elemento Peltier

A Fig. 18 mostra a pastilha utilizada no trabalho. Inicialmente foram feitos testes com ela, a fim de encontrar seu ponto ótimo de funcionamento.

Fig. 18 – Pastilha Peltier

Fonte: Os autores

Para isso, o a pastilha foi grudada a um dissipador e *cooler*, a fim de que evitar a queima do componente por meio da troca de calor eficiente. Então o sensor foi colocado ao lado exposto, sendo esse o lado de resfriamento, como mostrado na Fig. 19.

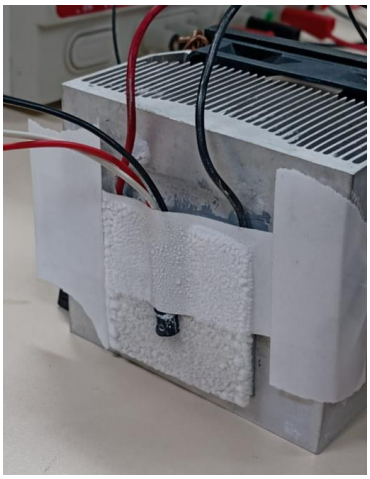


Fig. 19 – Esquema de montagem para teste da Peltier

Fonte: Os autores

O teste compreendeu em variar a tensão de funcionamento de 4V até cerca de 12 V ao passo de 0,5 V, analisando o lado frio da Peltier, a fim de encontrar o valor de tensão que possibilitasse a menor temperatura. Os resultados obtidos são mostrados na Tabela I, onde pode-se observar que a tensão ótima de operação é 12 V, que é quando o sensor lê -4,24 ºC. A Fig. 20 mostra um gráfico desse comportamento.

Logo após o teste de resfriamento foi feito um teste de aquecimento (apresentado também na Tabela I) onde foi encontrado um problema: a saturação o sensor a partir de 6 V, onde ele passa a ler continuamente 59,89 ºC, apesar do aumento da tensão.

Tabela I – Teste de aquecimento e resfriamento da Peltier

	Esfriando			Esquentando
Tensão (V)	Corrente (A)	Mínimo ºC	Máximo ºC	Corrente (A)
4	1,05	7,36	42	0,88
4,5	1,19	6,4	44,32	1,01
5	1,34	5	50,76	1,14
5,5	1,48	3,5	55,8	1,18
6	1,59	2,6	59,89	1,26
6,5	1,77	1,56	59,89	1,34
7	1,88	1,13	59,89	1,4
7,5	2,02	0,27		
8	2,16	-0,69		
8,5	2,3	-1,45		
9	2,41	-2,09		
9,5	2,52	-2,63		
10	2,66	-2,84		
10,5	2,8	-3,16		
11	2,92	-3,5		
11,5	3,01	-4,02		
11,8	3,08	-4,24		

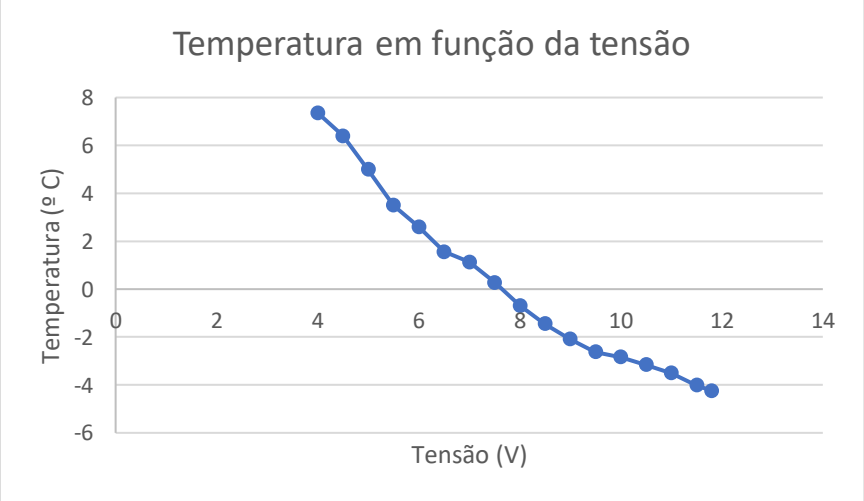


Fig. 20 – Temperatura em função da tensão

Fonte: Os autores

Foi verificado, então, que o sensor não poderia ser utilizado com a referência interna de 1,1 V, pois em determinadas temperaturas ele chegaria à sua saturação. De acordo com seu *datasheet*, em 5 ºC o TMP36 possui uma tensão de 510 mV. Dessa forma, em 75 ºC (a temperatura máxima proposta) ele chegaria a 1,21 V.

5.3. Referência externa de tensão

Dessa forma, foi necessário fazer uma referência externa de tensão para utilizar o sensor, pois o 1,1 V seria insuficiente e os 5 V causariam uma imprecisão significativa.

Assim foi construído um divisor de tensão a fim de obter na saída cerca de 1,4 V. Para isso foi utilizado um resistor de 470 Ω e um de 1200 Ω, chegando assim a

$$\text{Vout} = \text{Vin} * \frac{R2}{R1+R2} = 5 * \frac{470}{1200 + 470} = 1{,}407\ V \qquad (5)$$

a referência externa de tensão é feita conforme a Fig. 21, e o código de leitura do sensor é alterado conforme a Fig. 22, atualizado no Anexo I.

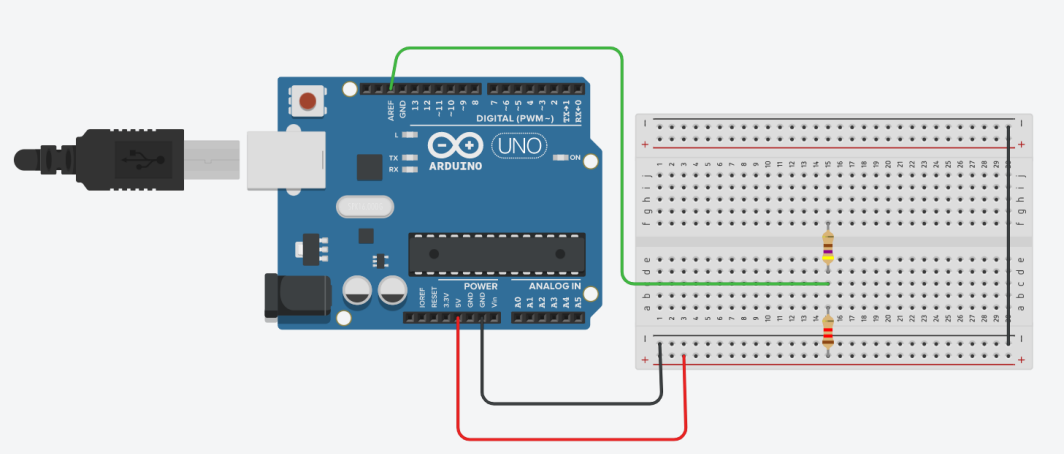


Fig. 21 – Esquema de montagem de AREF

Fonte: Os autores

```
void setup() {
Serial.begin(9600);
analogReference(EXTERNAL);
```

```
}

void loop(){
tempo = millis();      //armazena o valor do tempo na variável "tempo" (em ms)
int leitura = analogRead(Pino); //Lê a entrada
float tensao = leitura * 1.4; //multiplica por 1,4V
tensao /= 1024.0; //Conversão AD

float Temperatura = (tensao - 0.5) *100; //Subtraímos 0,5 para ajustar a faixa de medição do sensor
Serial.print(Temperatura); //Printa a temperatura
```

Fig. 22 – Mudança no código para a referência Externa

Fonte: Os autores

5.4. Montagem da Ponte H

Para a montagem da ponte H com dois relés foi utilizada a Fig. 23 como base [17], inicialmente projetada para a reversão do giro do motor. Para testes, na prática a ponte H foi montada com dois LEDs no lugar do motor (a Peltier requer uma alimentação regulada e acionamento com *cooler* ligado em alimentação separada em 12V, montagem que não é possível de fazer em casa), como mostra a Fig. 24.

Na Fig. 24, o acionamento de um botão faz com que a corrente flua num sentido e acione o LED verde. O acionamento do segundo botão faz com que a corrente flua no outro sentido e acenda o LED vermelho. Ambos os relés (AX1RC2) possuem tensão de acionamento em 12 V, bem como a Peltier. A ponte H será acionada (determinado relé será alimentado) conforme a faixa de temperatura programada. Isso é feito por meio de programação de portas digitais do Arduino, conectadas à base dos transistores BC548.

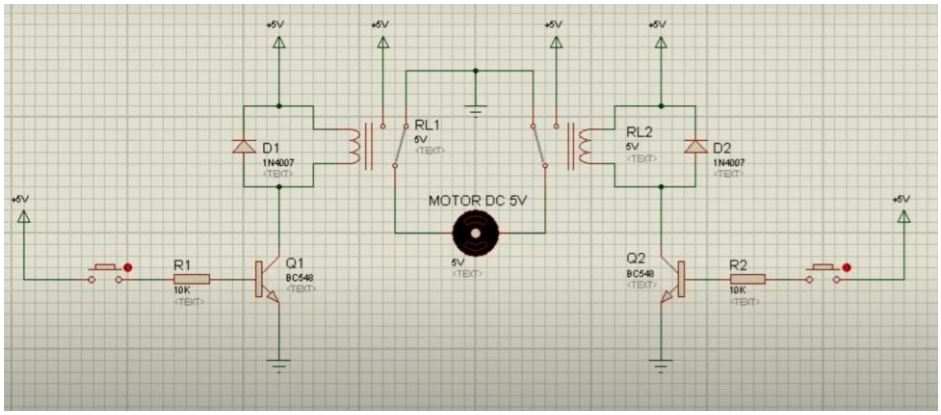


Fig. 23 – Circuito de simulação da Ponte H

Fonte: JV Eletrônica [17]

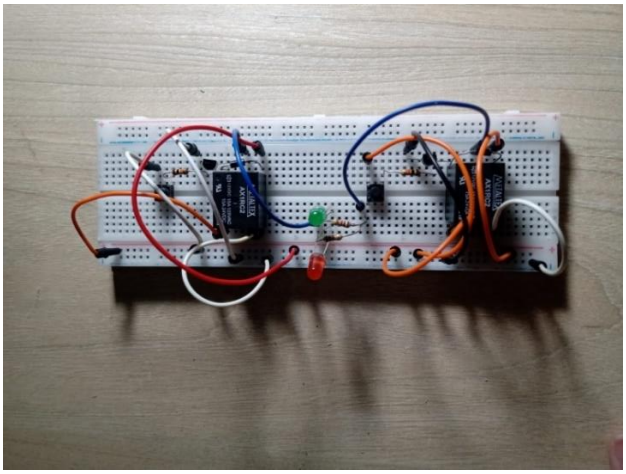


Fig. 24 – Teste da ponte H com LEDs

Fonte: Os autores

Para isso são utilizadas as portas 4 e 7, que são setadas como OUTPUT. Para um teste sem o uso da Peltier, são escolhidas duas faixas de temperatura: abaixo de 30 ºC e acima de 30 ºC, a fim de verificar o acionamento das portas. Quando configuradas em LOW, as portas não fornecem tensão; enquanto em HIGH fornecem 5 V, que é a tensão máxima fornecida pelo Arduino. Neste caso, 5 V são suficientes para o acionamento da ponte H. A Fig. 25 mostra a montagem do circuito, enquanto a Fig. 26 mostra o código utilizado para teste.

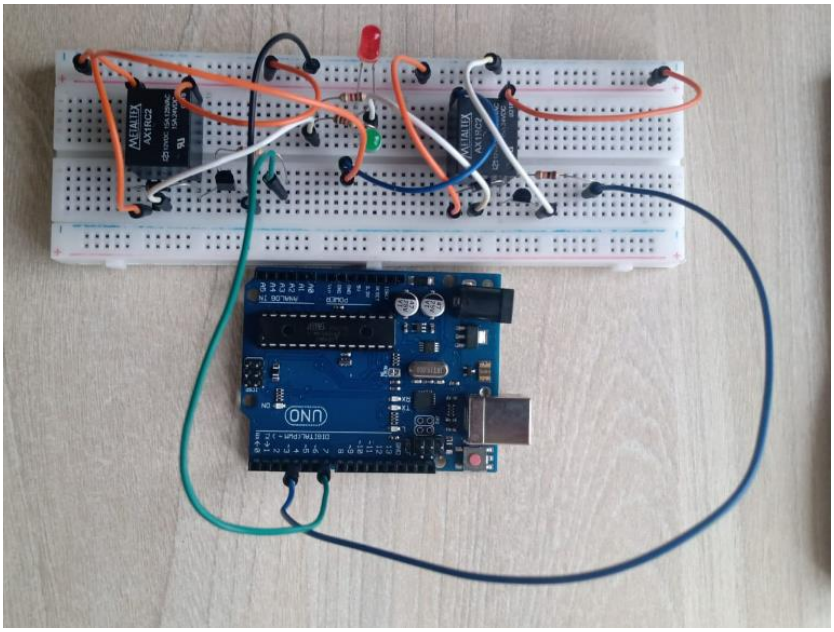


Fig. 25 – Montagem do circuito para acionamento da ponte H

Fonte: Os autores

```
void setup() {
  pinMode(4, OUTPUT);
  pinMode(7, OUTPUT);

void loop() {
if(Temperatura <= 30){
  digitalWrite(4, HIGH);
  digitalWrite(7, LOW);
}
if(Temperatura>30){
  digitalWrite(4, LOW);
  digitalWrite(7, HIGH);
```

Fig. 26 – Código para teste de acionamento da ponte H

Fonte: Os autores

5.5. Programação PWM

Então é necessário fazer um controle PWM a fim de controlar a intensidade da corrente que flui para a Peltier, com alteração instantânea de acordo com o valor inserido pelo usuário.

Isso é feito por meio de um MOSFET, neste caso, o IRFZ44N. O GND da ponte H é conectada ao dreno do transistor, enquanto o *gate* é conectado à porta digital 3 do Arduino. O *source* é conectado ao terra, como mostra a Fig. 27.

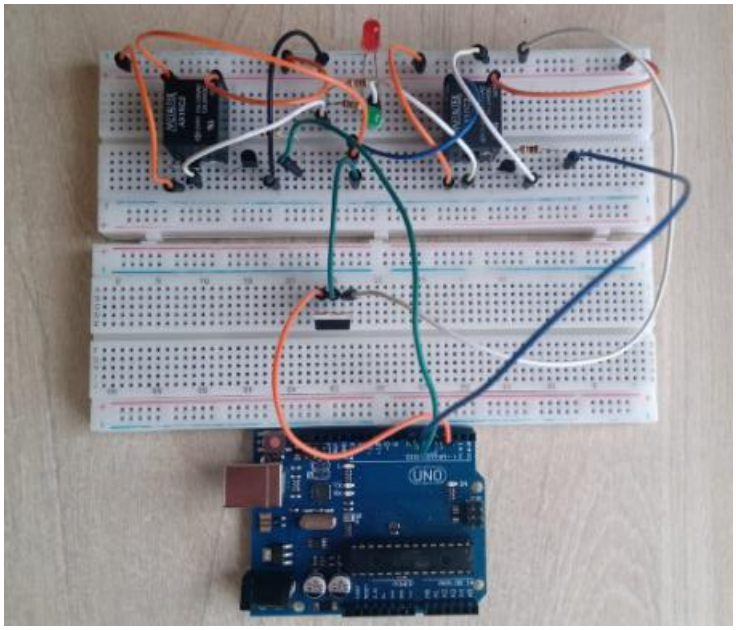


Fig. 27 – Conexão MOSFET para PWM

Fonte: os autores

No entanto, como existe a condição “esquenta” ou “esfria”, existem duas condições para a atuação do PWM. A fim de coletar somente uma variável da serial, faz-se uma faixa de atuação de 0 a 511, em que de 0 a 255 o PWM é definido no modo de resfriamento, enquanto de 256 até 511 ele é definido no modo de aquecimento. Esta última faixa possui uma subtração de 256 do valor inserido para manter o PWM entre 0 a 255, como mostra a Fig. 28.

```
//Esquerda: VCC, meio: sinal, direita: GND

#define Pino A0 //Define a porta A0 como entrada
int media[100];
float temp=0.0;
int PWM = 0;  //variável PWM (0 a 255);
unsigned int Ts = 500;  //tempo de amostragem em
unsigned long tempo = 0; //tempo de contagem em ms
int Aux_PWM = 0;


Serial.println("Digite o PWM de 0 a 255 para esfriar; 256 a 511 para esquentar: ");


void loop(){
tempo = millis();      //armazena o valor do tempo na variável "tempo" (em ms)
int leitura = analogRead(Pino); //Lê a entrada
float tensao = leitura * 1.4; //multiplica por 1,4V
tensao /= 1024.0; //Conversão AD

float Temperatura = (tensao - 0.5) *100; //Subtraímos 0,5 para ajustar a faixa de medição do sensor
Serial.print(Temperatura); //Printa a temperatura
Serial.print(" ");//envia um espaço de separação dos dados
Serial.println(PWM);
if (Serial.available() > 1) {  //verifica se há dados na serial
Aux_PWM = Serial.parseInt(); //lê caracteres numéricos ASCII da serial e converte em um número inteiro
Aux_PWM = constrain(Aux_PWM, 0, 511);  //limita os valores da variável PWM entre 0 a 511

if (Aux_PWM <= 255){

  PWM = Aux_PWM;

  digitalWrite(4, HIGH);
  digitalWrite(7, LOW);
  }
if(Aux_PWM > 255){
  PWM = Aux_PWM - 256;
  digitalWrite(4, LOW);
  digitalWrite(7, HIGH);
  }

}
```

Fig. 28 – Código de controle PWM

Fonte: Os autores

5.6. Sistema completo

Dessa forma, o sistema completo é mostrado na Fig. 29, onde consta o acionamento da Peltier por meio da ponte H (com o controle pelas portas digitais

do Arduino), o controle PWM por meio do MOSFET, a conexão para leitura do TMP36 e o divisor de tensão utilizado em AREF.

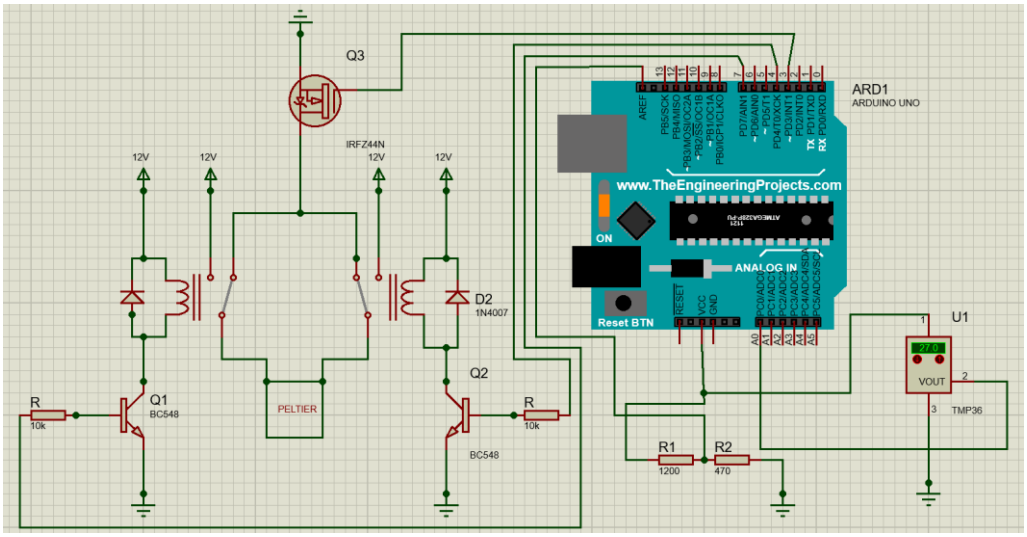


Fig. 29 – Sistema completo

Fonte: Os autores

E o código completo é descrito a seguir, também colocado no Anexo II.

Inicialmente são definidos pinos, portas e variáveis, conforme Fig. 30. Então é feita uma leitura inicial do ambiente, a fim de definir o sentido da corrente para a Peltier esquentar ou esfriar. Como leva um certo tempo para a estabilização da temperatura no início da leitura, é feito uma iteração para definir a temperatura ambiente como a média das primeiras 100 leituras.

```
//Esquerda: VCC, meio: sinal, direita: GND

#define Pino A0 //Define a porta A0 como entrada
int media[100];
float temp=0.0;
int PWM = 0;  //variável PWM (0 a 255);
unsigned int Ts = 500;  //tempo de amostragem em
unsigned long tempo = 0; //tempo de contagem em ms
int Aux_PWM = 0;

void setup() {
Serial.begin(9600);
analogReference(EXTERNAL);
Serial.setTimeout(50); //limita o tempo de espera de caracteres de entrada da serial
pinMode(3, OUTPUT);//define pino 3 como saida PWM para Gate do MOSFET que irá acionar uma carga
analogWrite (3, 0); //escreve o valor 0 na saída PWM
pinMode(4, OUTPUT);
pinMode(7, OUTPUT);

for (int i=0; i<100; i++){
int leitura = analogRead(Pino); //Lê a entrada
float tensao = leitura * 1.4; //multiplica por 1,4V
tensao /= 1024.0; //Conversão AD
media[i] = (tensao - 0.5) *100; //Subtraímos 0,5 para ajustar a faixa de medição do sensor
temp = temp+media[i];
delay(10);
}
float TemperaturaAmbiente = temp/100.0;
```

Fig. 30 – Primeira parte do código

Fonte: Os autores

Após isso a temperatura ambiente é informada ao usuário, e são informadas as faixas de valor de esfriamento e aquecimento, a fim de obter o PWM, como mostrado na Fig. 31.

Então, no loop principal é feita a leitura constante da temperatura atual, bem como a leitura instantânea do PWM pela serial. Há uma sintaxe de condição para definir o sentido da corrente e a sua intensidade, de acordo com o digitado.

Também nesse trecho há a leitura do PWM, restrito entre 0 e 255, a fim de permitir o controle instantâneo da intensidade da corrente.

```
void loop(){
tempo = millis();      //armazena o valor do tempo na variável "tempo" (em ms)
int leitura = analogRead(Pino); //Lê a entrada
float tensao = leitura * 1.4; //multiplica por 1,4V
tensao /= 1024.0; //Conversão AD

float Temperatura = (tensao - 0.5) *100; //Subtraímos 0,5 para ajustar a faixa de medição do sensor
Serial.print(Temperatura); //Printa a temperatura
Serial.print(" ");//envia um espaço de separação dos dados
Serial.println(PWM);
if (Serial.available() > 1) {  //verifica se há dados na serial
Aux_PWM = Serial.parseInt(); //lê caracteres numéricos ASCII da serial e converte em um número inteiro
Aux_PWM = constrain(Aux_PWM, 0, 511);  //limita os valores da variável PWM entre 0 a 511

if (Aux_PWM <= 255){

  PWM = Aux_PWM;

  digitalWrite(4, HIGH);
  digitalWrite(7, LOW);
  }
if(Aux_PWM > 255){
  PWM = Aux_PWM - 256;
  digitalWrite(4, LOW);
  digitalWrite(7, HIGH);
  }

}
analogWrite (3, PWM);  //atribui o valor do PWM à porta de saída 3
Serial.read(); //lê e limpa dados restantes da serial
//delay(1000);
while ((tempo+Ts) > millis()) {}     //espera o tempo necessário para completar o tempo de amostragem Ts
}
```

Fig. 31 – Segunda parte do código

Fonte: Os autores

6 – CONDICIONAMENTO E ESTATÍSTICA

6.1. Circuito de condicionamento do atuador

Após soldar o circuito em uma placa de circuito universal como mostrado na Fig. 32, foram realizados novos testes a fim de mensurar a precisão do sistema.

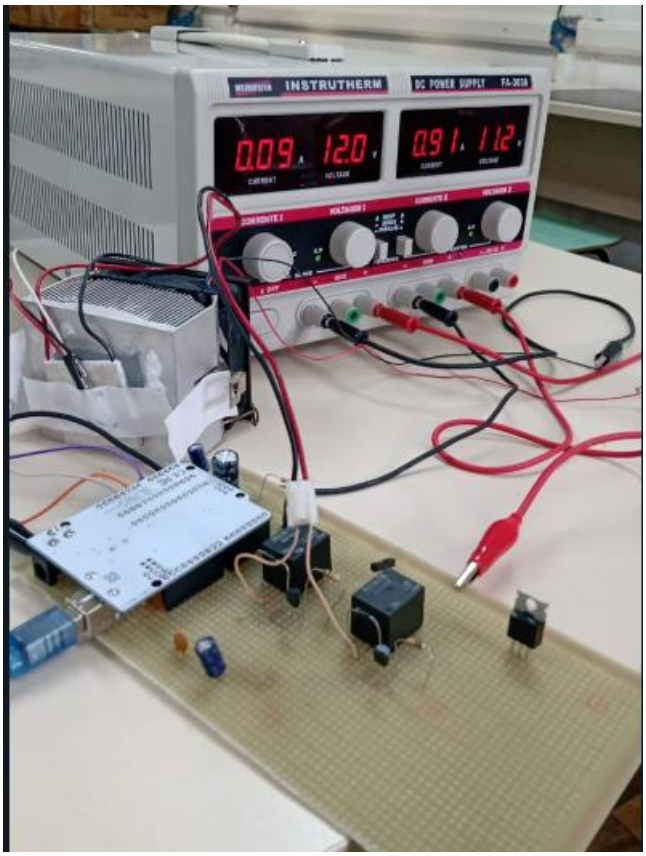


Fig. 32 – Circuito soldado na placa universal

Fonte: Os autores

Mantendo a Peltier por volta de 35 ºC (aproximadamente metade da sua faixa de operação) com 12 V de alimentação e um PWM constante de 25, foram analisados aproximadamente 100 dados mostrados na serial da IDE, que foram colocados no Excel a fim de análise. Nesses dados, vê-se que a menor variação entre um dado e outro é de aproximadamente 0,14 º C, como mostra a Fig. 33.

	35.18 25
	35.18 25
	35.31 25
	35.31 25
	35.31 25
	35.18 25
	35.18 25
	35.31 25
	35.31 25
	35.31 25
	35.31 25
	35.31 25
	35.31 25
	35.31 25
	35.31 25
	35.31 25
	35.31 25
	35.31 25
	35.31 25
	35.31 25
	35.31 25
	35.31 25
	35.18 25
	35.18 25
	35.18 25

Fig. 33 – Análise da variação

Fonte: Os autores

Analisando a faixa dinâmica, obtém-se

$$\frac{0,14}{70} = 0,002 \tag{6}$$

que resulta em uma resolução de 8 a 9 bits.

6.2. Análise estatística dos dados do sensor

Então foram feitos alguns testes para medir o SNR do sistema, sendo este definido por

$$SNR = \frac{Valor\ médio}{Desvio\ Padrão} \tag{7}$$

No início, verificou-se que o SNR estava muito ruim, na ordem de 30 ou 40. Então foram realizados vários testes com diferentes capacitores ao longo do circuito em busca de melhoras. Um arranjo que pareceu provocar melhoras foi um capacitor eletrolítico de 1000 µF, um de 100 µF e um de 100 nF entre o 5 V e o terra; um de 100 µF entre o 12 V e o terra; e um de 47 µF entre o pino analógico A0 (que recebe o sinal do sensor) e o terra. Com esse circuito, foram feitas novas medições na temperatura de 35 ºC, cujos resultados são mostrados na Fig. 34.

Capacitor no A0			Capacitor no A0			Capacitor no A0		
Capacitores na alimentação			Capacitores na alimentação			Capacitores na alimentação		
Com a Peltier			Com a Peltier			Com a Peltier		
	Temp			Temp			Temp	
35.72 25	35,72	35,28991 média	35.59 25	35,59	35,52246 média	35.45 25	35,45	35,18628 média
35.59 25	35,59	0,346685 desvio	35.72 25	35,72	0,123645 desvio	35.45 25	35,45	0,133717 desvio
35.59 25	35,59	101,7923 SNR	35.72 25	35,72	287,2945 SNR	35.45 25	35,45	263,1391 SNR
35.59 25	35,59		35.72 25	35,72		35.45 25	35,45	
35.59 25	35,59		35.59 25	35,59		35.45 25	35,45	
35.45 25	35,45		35.72 25	35,72		35.45 25	35,45	
35.59 25	35,59		35.59 25	35,59		35.45 25	35,45	
35.45 25	35,45		35.72 25	35,72		35.45 25	35,45	
35.59 25	35,59		35.59 25	35,59		35.45 25	35,45	
35.59 25	35,59		35.59 25	35,59		35.31 25	35,31	
35.59 25	35,59		35.59 25	35,59		35.31 25	35,31	
35.45 25	35,45		35.72 25	35,72		35.45 25	35,45	
35.45 25	35,45		35.59 25	35,59		35.31 25	35,31	
35.45 25	35,45		35.72 25	35,72		35.31 25	35,31	
35.45 25	35,45		35.59 25	35,59		35.31 25	35,31	
35.45 25	35,45		35.59 25	35,59		35.31 25	35,31	
35.45 25	35,45		35.59 25	35,59		35.18 25	35,18	
35.31 25	35,31		35.59 25	35,59		35.31 25	35,31	
35.31 25	35,31		35.59 25	35,59		35.31 25	35,31	
35.31 25	35,31		35.59 25	35,59		35.31 25	35,31	
35.31 25	35,31		35.72 25	35,72		35.45 25	35,45	

Fig. 34 – Resultados de SNR

Fonte: Os autores

Ressalta-se que o segundo SNR e o terceiro SNR foram obtidos alguns minutos após o primeiro, o que mostra que quanto maior o tempo em um mesmo PWM e uma mesma temperatura resultam maior estabilidade nas medições. De forma geral, nesta composição o SNR está satisfatório, maior que 100.

7 – SISTEMA DE CONTROLE PID

Nesta etapa será realizado um sistema de controle sobre o trabalho. Como existem duas regiões de operação (aquecimento e resfriamento), serão projetados 2 sistemas. O código finalizado está presente na íntegra no Anexo III.

7.1. Aquecimento

Inicialmente foram feitos alguns testes com o sistema, em que foi identificado um grande ruído. Apesar de termos obtido um bom SNR na segunda etapa, a montagem fixa do sistema pode ter provocado alguma oscilação. Dessa forma, a fim de corrigir isso sem ter que mexer a nível de hardware, foram aplicados dois filtros digitais: a média sobre amostragem e média móvel exponencial (EWMA), conforme a Fig. 35, em que “media_ruido” torna-se a variável de saída.

```
//Filtro de sobreamostragem
temp2 = 0;
for (int k=0; k<10; k++){
  int leitura2 = analogRead(Pino); //Lê a entrada
  float tensao2 = leitura2 * 1.4; //multiplica por 1,4V
  tensao2 /= 1024.0; //Conversão AD
  media2[k] = (tensao2 - 0.5) *100; //Subtraímos 0,5 para ajustar a faixa de medição do sensor
  temp2 = temp2+media2[k];
}
float Temperatura = temp2/10.0;

//alfa = (n-1)/n, em que n = 5.
//EWMA = EWMA * alfa+(1-alfa)*floatAD
media_ruido = media_ruido*0.8 + (0.2)*Temperatura;
```

Fig. 35 – Aplicação de filtros digitais no código

Fonte: Os autores

A fim de montar o sistema de controle, foi necessário dividir a operação do aquecimento em baixa e alta temperatura, dividindo a faixa em até 42ºC e após 42ºC.

7.1.1 Operação em baixa temperatura no aquecimento

Para identificar a forma do sistema e posteriormente inserir os dados no MATLAB, é necessário inicialmente fazer a coleta de temperatura após a inserção de um degrau. No entanto, a fim de melhorar a identificação, é subtraído de todos os valores a temperatura ambiente (temperatura inicial do sistema), para que ele comece em 0. Assim, a partir da temperatura ambiente foi aplicado um degrau com PWM em 50, o que resultou no comportamento mostrado na Fig. 36, identificando um sistema de segunda ordem.

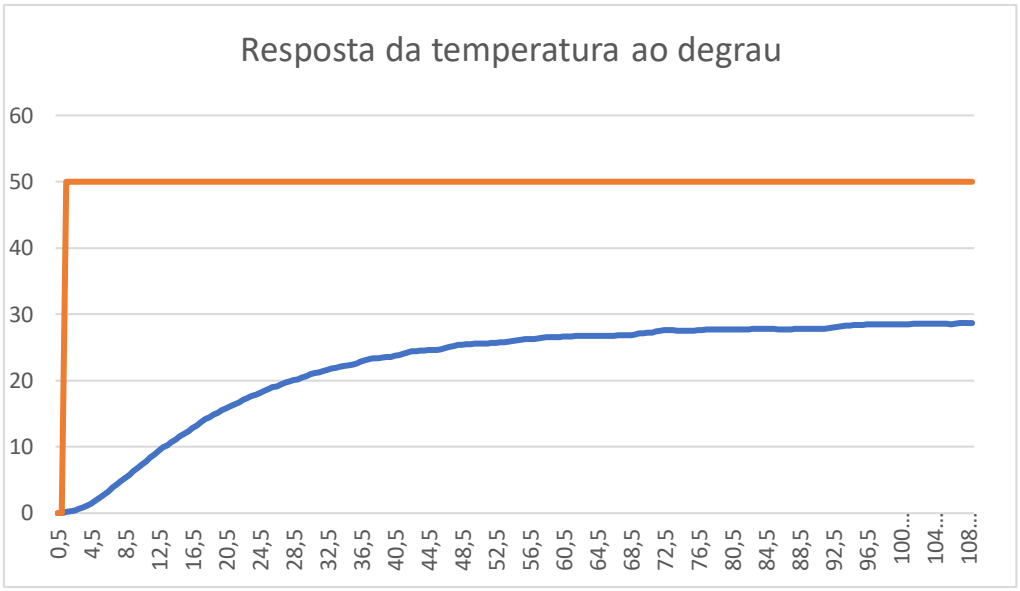


Fig. 36 – Resposta ao degrau da primeira faixa de temperatura

Fonte: Os autores

Sendo um sistema de segunda ordem, podemos identificá-lo pela equação

$$H(s) = \frac{k}{(1+Tp1s)(1+Tp2s)}, \tag{8}$$

em que k, Tp1 e Tp2 podem ser estimados pelo MATLAB, como mostrado na Fig. 37.

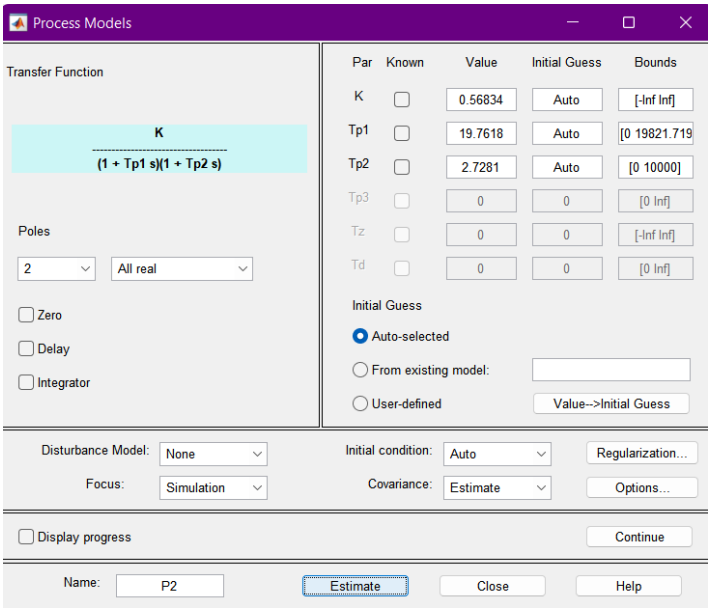


Fig. 37 – Parâmetros MATLAB

Fonte: Os autores

A Fig. 38 mostra o model output, que é a identificação da simulação com os dados reais.

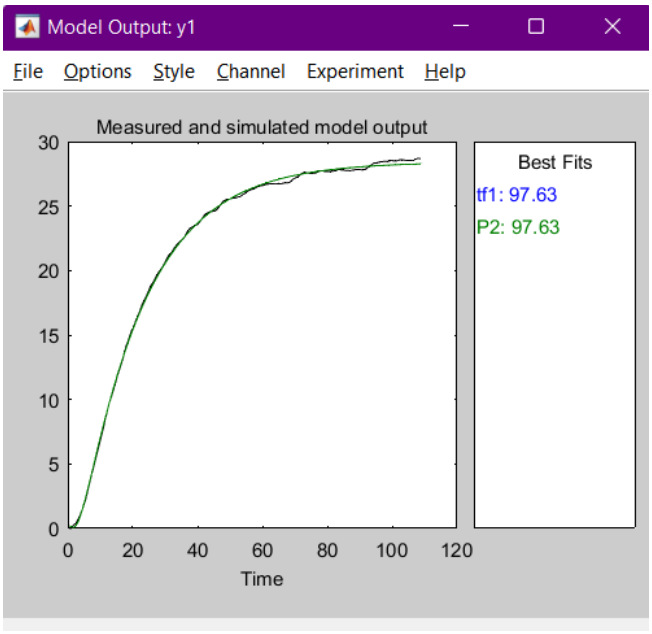


Fig. 38 – Model output

Fonte: Os autores

A partir disso foram calculados o T e L conforme orientado (a partir da reta tangente ao ponto de inflexão da curva de resposta ao sistema) e posteriormente os parâmetros do PID (Ki, Kp, Kd). O tempo de amostragem Ts é constante em 0,5 s.

Com isso, k = 0,56834, L = 1,5831 e T = 26,9806, e os demais parâmetros são definidos por

$$Kp = \frac{1{,}2}{k} * \frac{T}{L} = \frac{1{,}2}{0{,}56834} * \frac{26{,}9806}{1{,}5831} = 35{,}9845 \quad (9)$$

$$Ki = Kp * \frac{Ts}{2L} = 35{,}9845 * \frac{0{,}5}{2*1{,}5831} = 5{,}6826 \quad (10)$$

$$Kd = Kp * \frac{L}{Ts} = 35{,}9845 * \frac{1{,}5831}{0{,}5} = 113{,}93 \quad (11)$$

No entanto, esses valores não apresentaram um bom sistema, causando um grande overshoot e grande erro em regime permanente. Dessa forma, foi feito um ajuste fino nos valores a partir de tentativa e erro, chegando a Kp = 29, Ki = 5,45 e Kd = 125, implementados no código conforme a Fig. 39.

```
      if(SetPoint <= 42){
      float KA = 0.56834;
      float KpA = 29;
      float KiA = 5.45;
      float KdA = 125;
        digitalWrite(4, HIGH);
        digitalWrite(7, LOW);
      PWM = (erro*KpA) + (int_erro*KiA) - (dif_temp*KdA);
      media_ruido_ant = media_ruido;
        if(PWM > 255){
          PWM = 255;
        }
        if(PWM <0){
          PWM = 0;
        }
 analogWrite (3, PWM); //atribui o valor do PWM à porta de saída 3
}
```

Fig. 39 – Ajuste dos parâmetros

Fonte: Os autores

A partir disso, são feitos testes da temperatura ambiente para 40ºC, como mostrado na Fig. 40 (overshoot) e Fig. 41 (regime permanente).

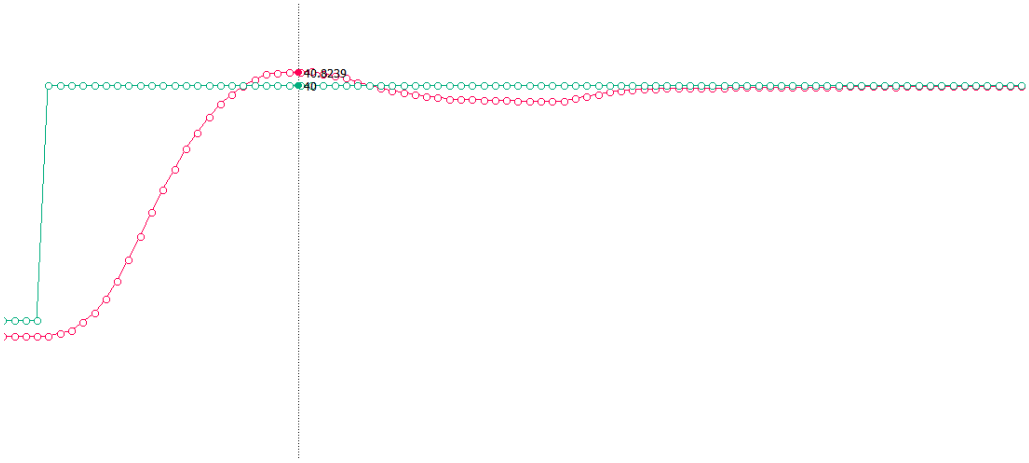


Fig. 40 – Overshoot em 40 ºC

Fonte: Os autores

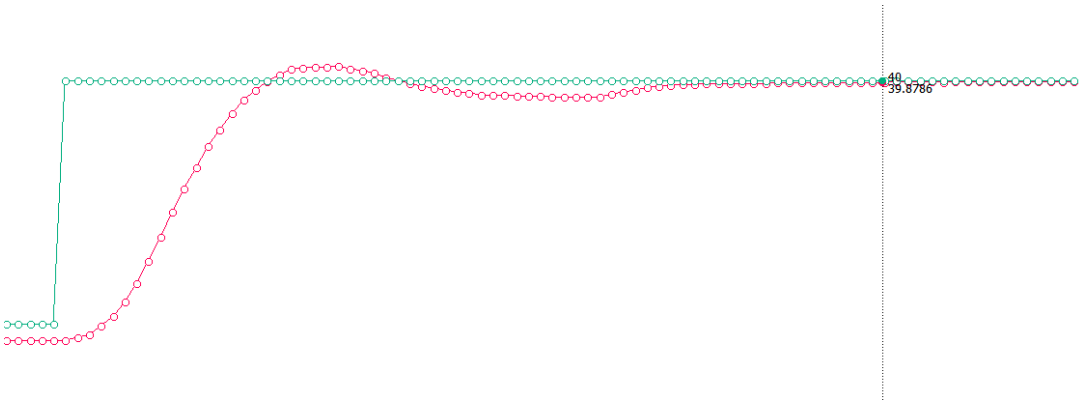


Fig. 41 – Regime permanente em 40ºC

Fonte: Os autores

E então testes da temperatura ambiente para 35ºC, conforme Fig. 42 (overshoot) e Fig. 43 (regime permanente).

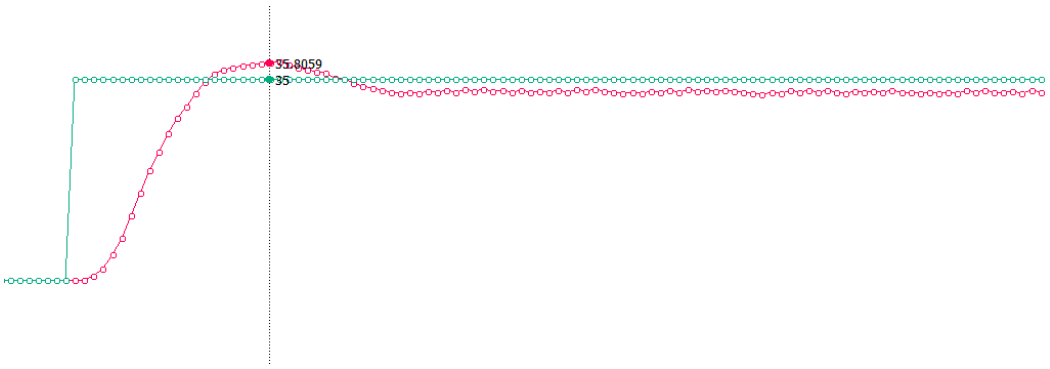


Fig. 42 – Overshoot em 35 ºC

Fonte: Os autores

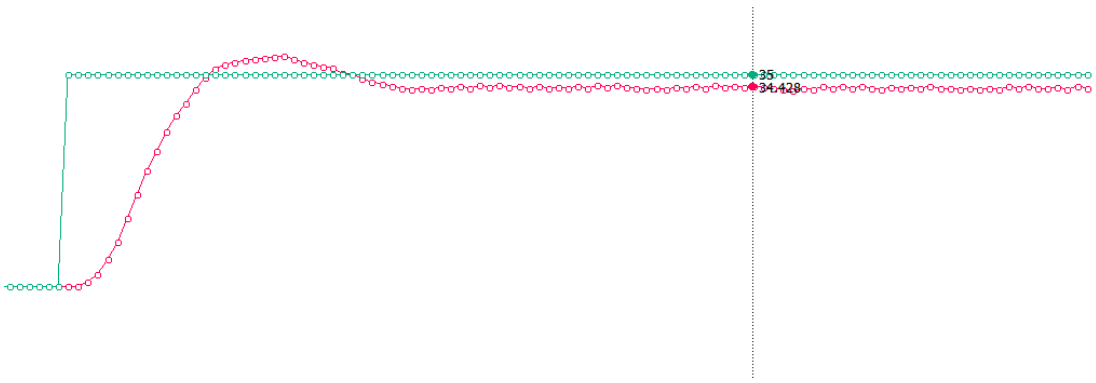


Fig. 43 – Regime permanente em 35 ºC

Fonte: Os autores

A Tabela II mostra a comparação entre os limites do sistema medido e desejado.

Tabela II – Precisão da operação para temperatura média

	Overshoot (5%)	Regime Permanente (2%)
Desejado 40ºC	42	39,2 - 40,8
Obtido 40ºC	40,82	39,87
Desejado 35ºC	36,75	34,3 – 35,7
Obtido 35ºC	35,81	34,44

7.1.2 Operação em alta temperatura no aquecimento

Da mesma forma foi prosseguido para o sistema com alta temperatura. No entanto, dessa vez o sistema foi iniciado em 40ºC, com PWM de 23, e então aquecido até aproximadamente 60ºC, com PWM em 60. Da mesma forma, de todos os valores foi descontada a temperatura inicial, o que também resultou em um sistema de segunda ordem, como mostrado na Fig. 44 (no MATLAB, pois devido aos valores o Excel não conseguiu plotar o gráfico formatado). Os parâmetros estimados são mostrados na Fig. 45, pelo MATLAB, e o model output é mostrado na Fig. 46.

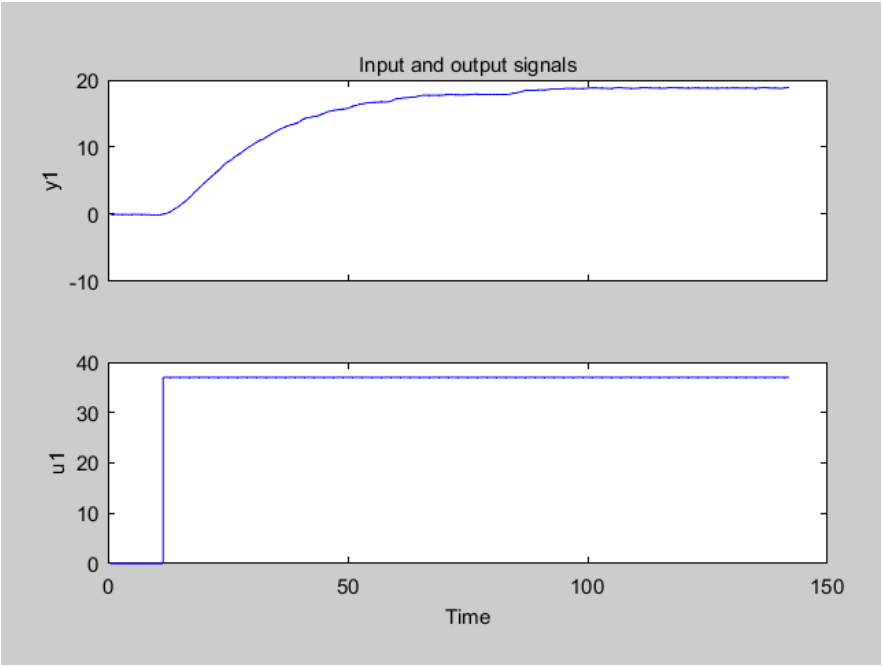


Fig. 44 – Resposta ao degrau para altas temperaturas

Fonte: Os autores

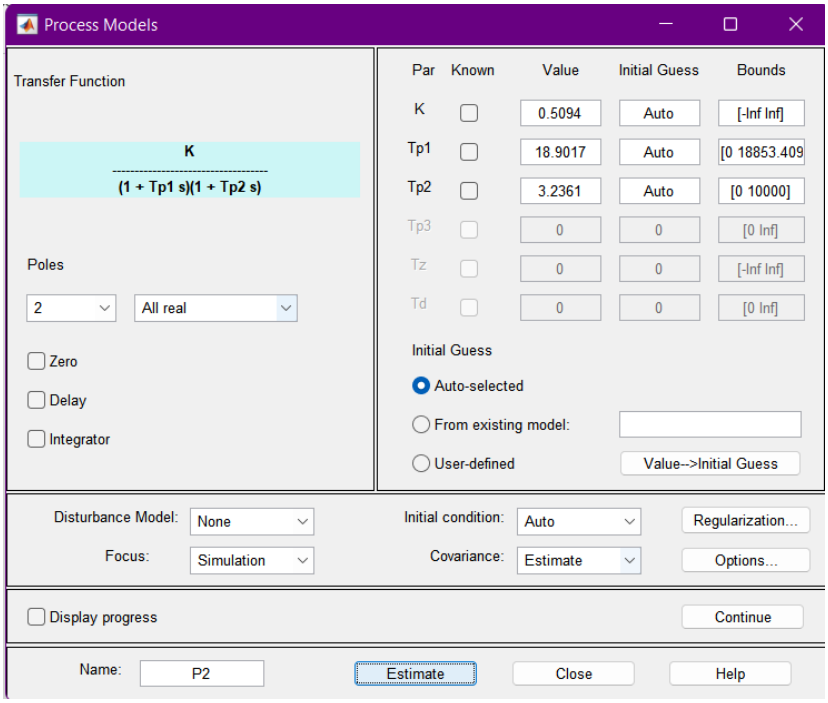


Fig. 45 – Parâmetros apresentados

Fonte: Os autores

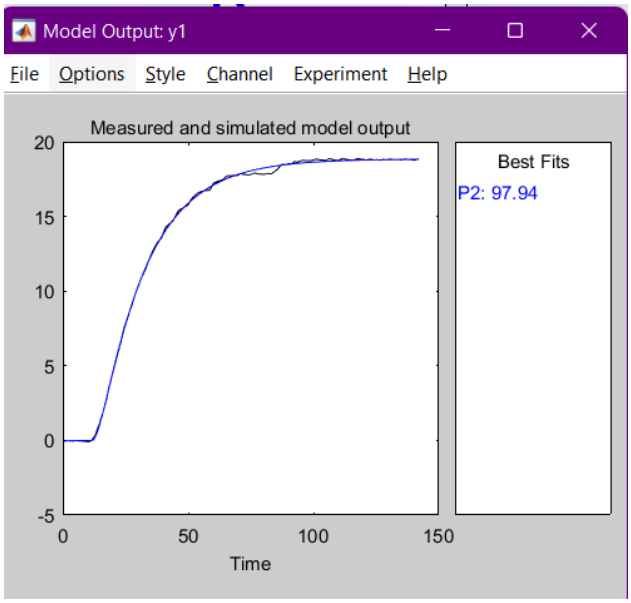


Fig. 46 – Model output

Fonte: Os autores

Com isso, foram calculados L = 1,8122657, T = 23,216683, e

$$Kp = \frac{1,2}{k} * \frac{T}{L} = \frac{1,2}{0,5094} * \frac{23,216683}{1,8122657} = 30,178 \quad (12)$$

$$Ki = Kp * \frac{Ts}{2L} = 30,1787 * \frac{0,5}{2*1,8122657} = 4,163 \quad (13)$$

$$Kd = Kp * \frac{L}{Ts} = 30,1787 * \frac{1,8122657}{0,5} = 109,3836 \quad (14)$$

No entanto, a fim de um ajuste fino, o Ki foi alterado para 4. Dessa forma o sistema foi montado conforme a Fig. 47.

```
if(SetPoint > 42){
 float LA = 1.8122657;
 float TA = 23.216683;
float KA = 0.56834;
float KpA = (1.2/KA)*(TA/LA);
float KiA = 4;
float KdA = KpA*(LA/0.5);
  digitalWrite(4, HIGH);
  digitalWrite(7, LOW);
PWM = (erro*KpA) + (int_erro*KiA) - (dif_temp*KdA);
media_ruido_ant = media_ruido;
  if(PWM > 255){
    PWM = 255;
  }
  if(PWM <0){
    PWM = 0;
  }
 analogWrite (3, PWM); //atribui o valor do PWM à porta de saída 3
}
```

Fig. 47 – Ajuste dos parâmetros

Fonte: Os autores

Com isso, foram obtidos os gráficos variando da temperatura ambiente para 50ºC, conforme a Fig. 48, Fig. 49 e Fig. 50. Então é feito o teste de temperatura ambiente para 60ºC, como exibido na Fig. 51 e Fig. 52.

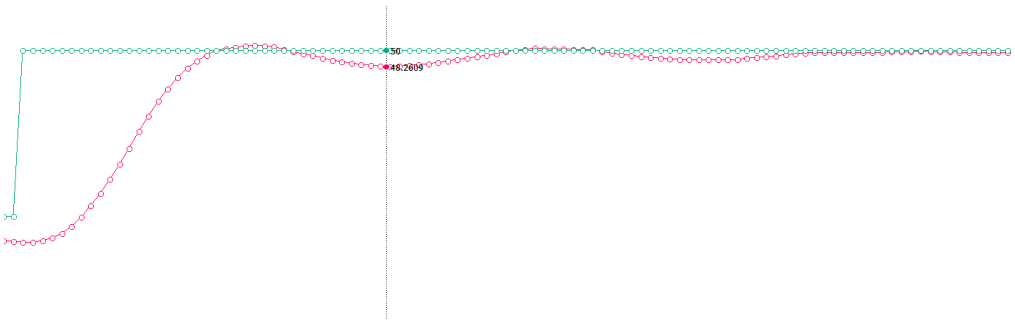


Fig. 48 – Undershoot em 50 ºC

Fonte: Os autores

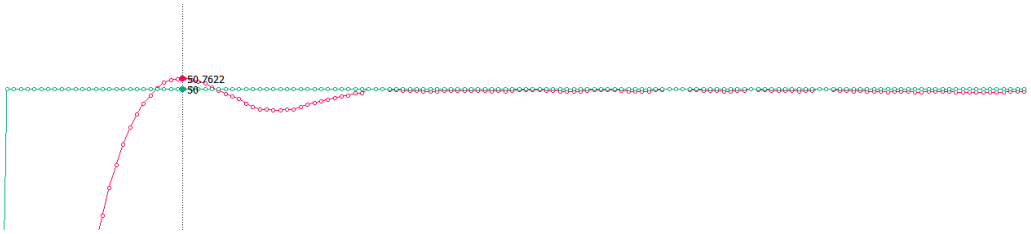


Fig. 49 – Overshoot em 50 ºC

Fonte: Os autores

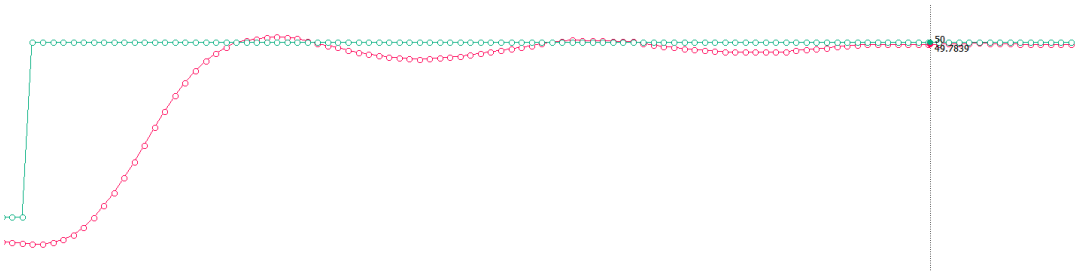


Fig. 50 – Regime permanente em 50 ºC

Fonte: Os autores

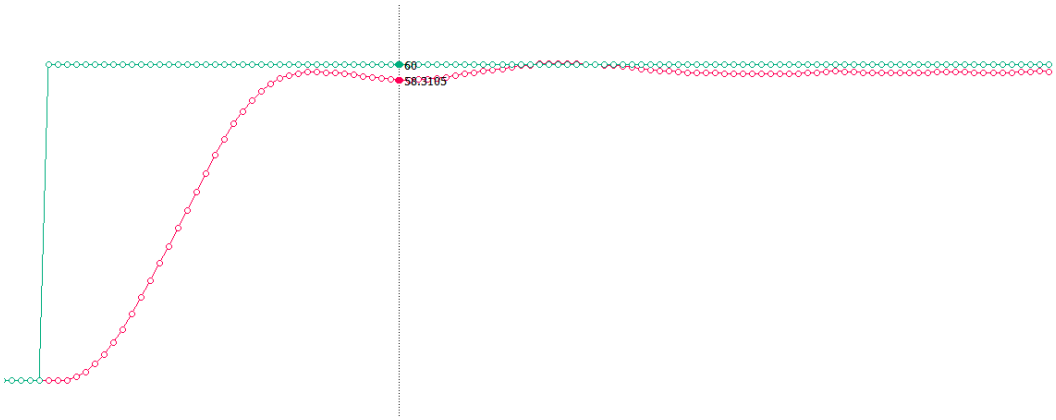


Fig. 51 – Undershoot em 60 ºC

Fonte: Os autores

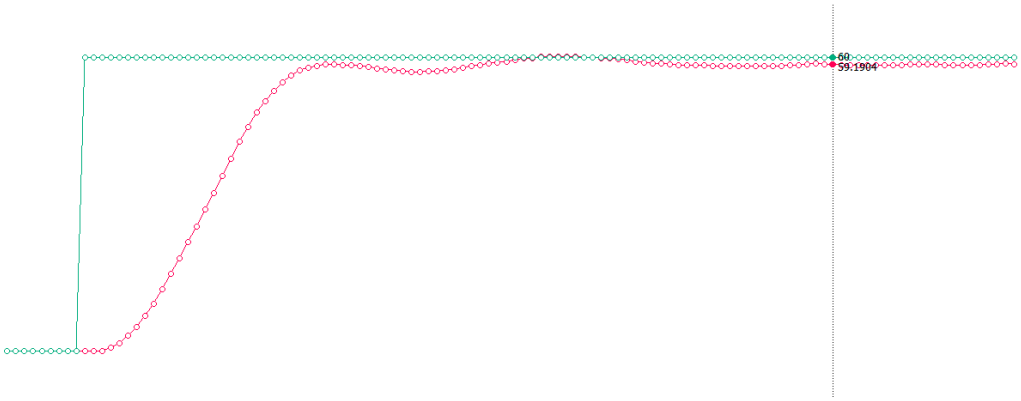


Fig. 52 – Regime permanente em 60 ºC

Fonte: Os autores

A Tabela III mostra a comparação entre o sistema obtido e desejado nessas faixas de operação.

Tabela III – Precisão da operação para temperatura alta

	Overshoot (5%)	Regime Permanente (2%)
Desejado 50ºC	47,5 - 52,5	49 - 51
Obtido 50ºC	48,26 – 50,76	49,78
Desejado 60ºC	57 - 63	58,8 – 61,2
Obtido 60ºC	58,3	59,19

Nota-se que todos os dados obtidos neste etapa estão dentro dos limites desejados.

7.2. Resfriamento

Para o resfriamento foi realizado o mesmo procedimento anterior, mas sem dividir a faixa de operação. Para a identificação do MATLAB, foram coletados dados após a aplicação de um degrau de PWM 100. Então foi subtraída a temperatura inicial de todos os valores, e em seguida multiplicado por -1. Após isso, a faixa ficou de 0 até aproximadamente 20ºC (que corresponde a 26ºC até 5ºC na realidade). O comportamento é mostrado na Fig. 53, identificando um sistema de segunda ordem.

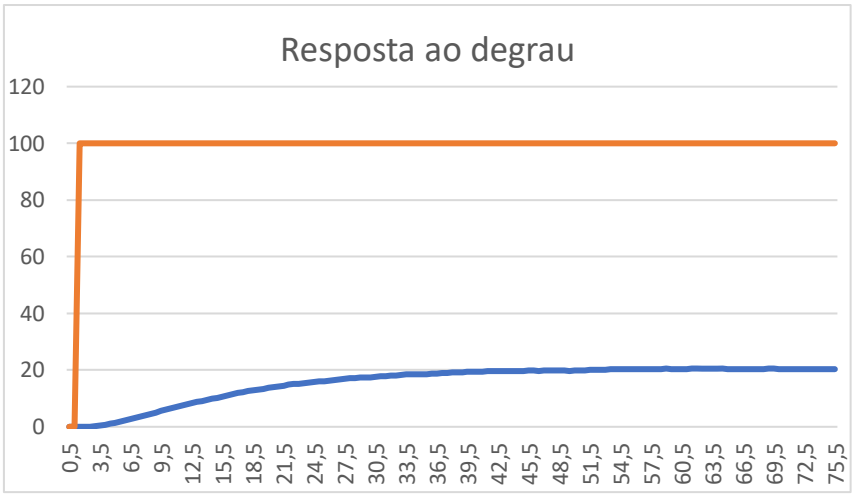


Fig. 53 – Resposta ao degrau no resfriamento

Fonte: Os autores

Esses dados foram inseridos no MATLAB, que resultou nos parâmetros apresentados na Fig. 54 e então plotados na Fig. 55.

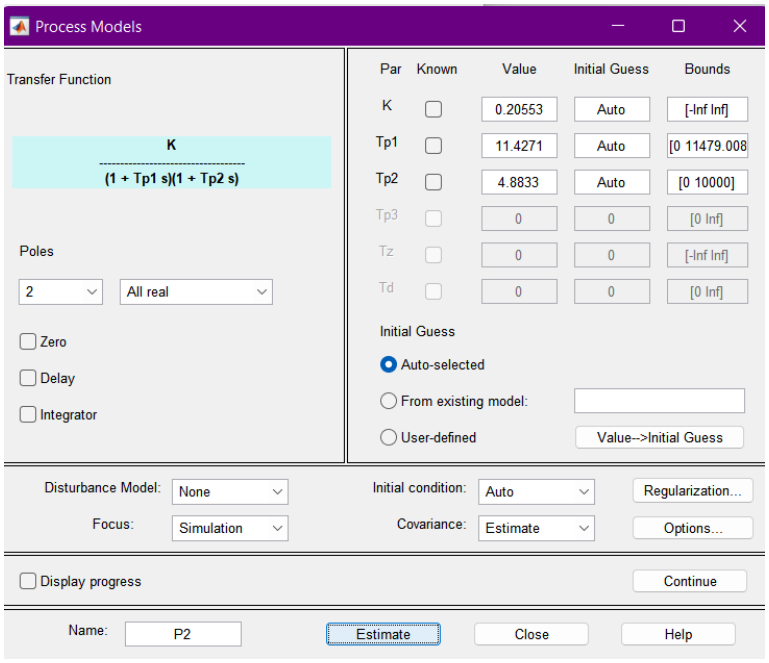


Fig. 54 – Parâmetros do sistema

Fonte: Os autores

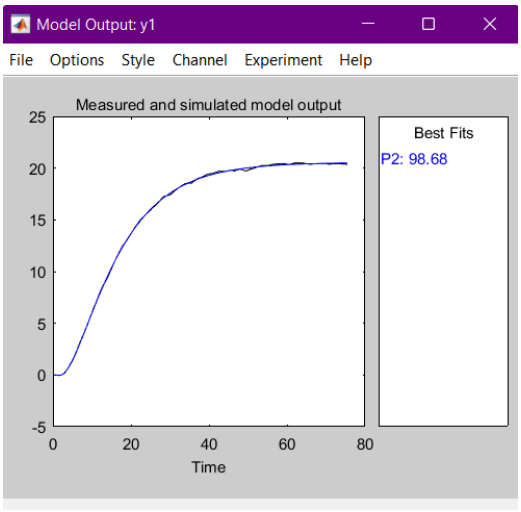


Fig. 55 – Model output

Fonte: Os autores

Com isso, foram calculados L = 2,009137 e T = 21,55101913, e

$$Kp = \frac{1{,}2}{k} * \frac{T}{L} = \frac{1{,}2}{0{,}20553} * \frac{21{,}55101913}{2{,}009137} = 62{,}6273 \quad (15)$$

$$Ki = Kp * \frac{Ts}{2L} = 62{,}6273 * \frac{0{,}5}{2*2{,}009137} = 7{,}79 \quad (16)$$

$$Kd = Kp * \frac{L}{Ts} = 62{,}6273 * \frac{2{,}009137}{0{,}5} = 252{,}653 \quad (17)$$

No entanto, a resposta a esse sistema ficou muito instável, com muitas oscilações em regime permanente, conforme a Fig. 56.

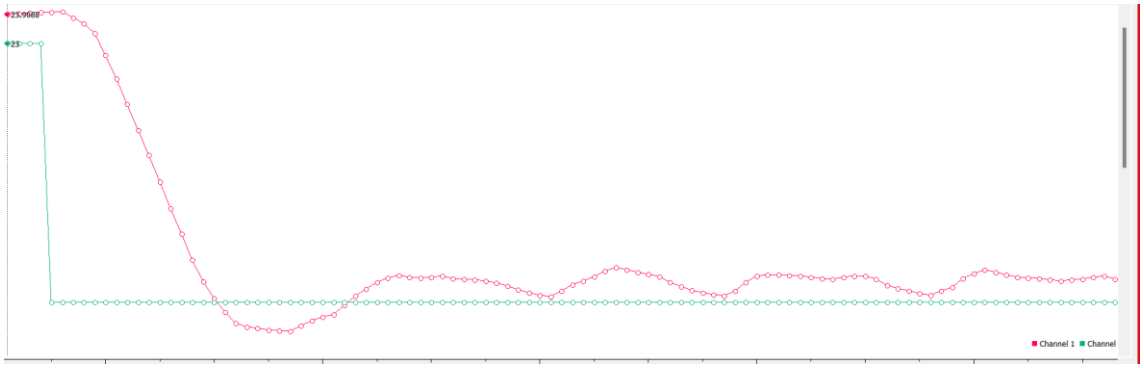


Fig. 56 – Oscilações em regime permanente

Fonte: Os autores

A partir disso, tentamos modificar as constantes Ki, Kp e Kd a fim de obter alguma melhora no sistema.

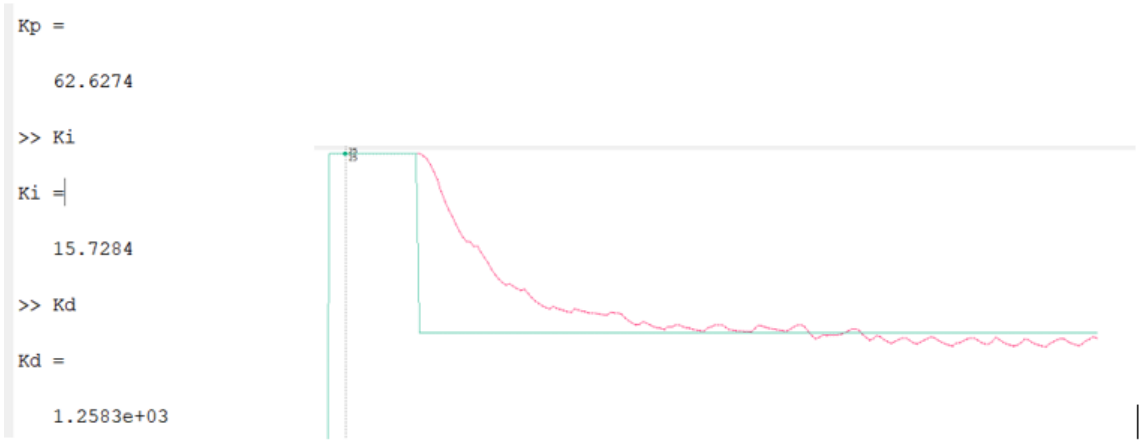


Fig. 57 – Alteração no Kd e Ki

Fonte: Os autores

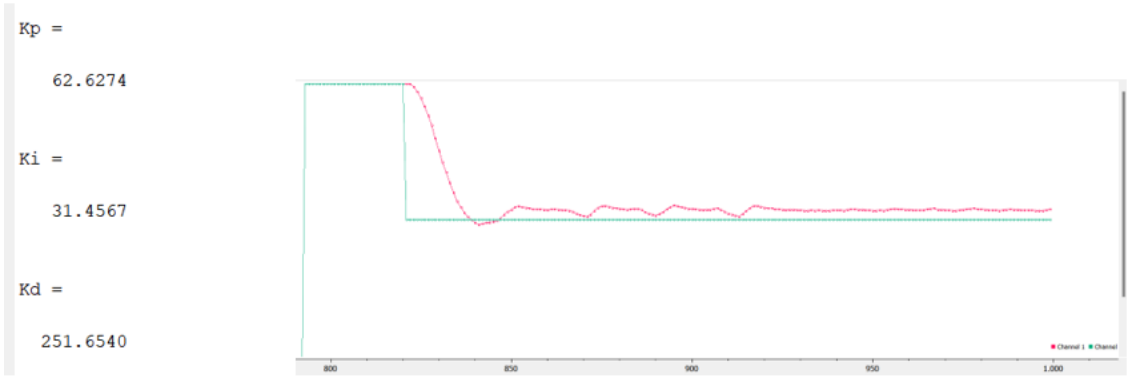


Fig. 58 – Alteração no Kd

Fonte: Os autores

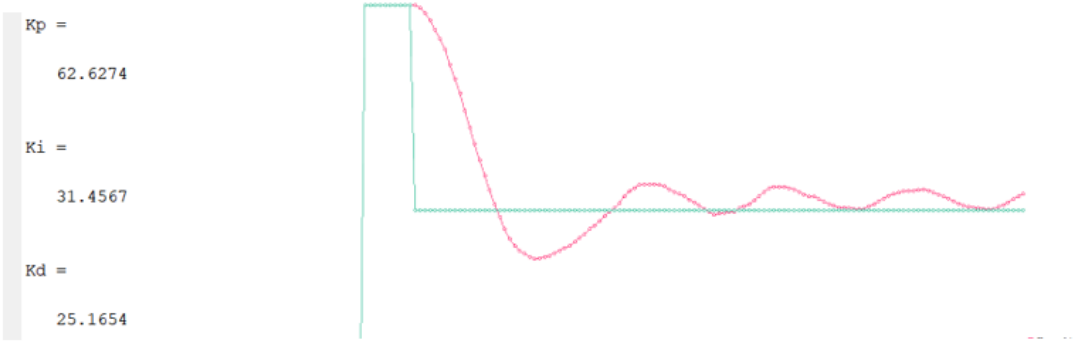


Fig. 59 – Alteração no Kd e Ki

Fonte: Os autores

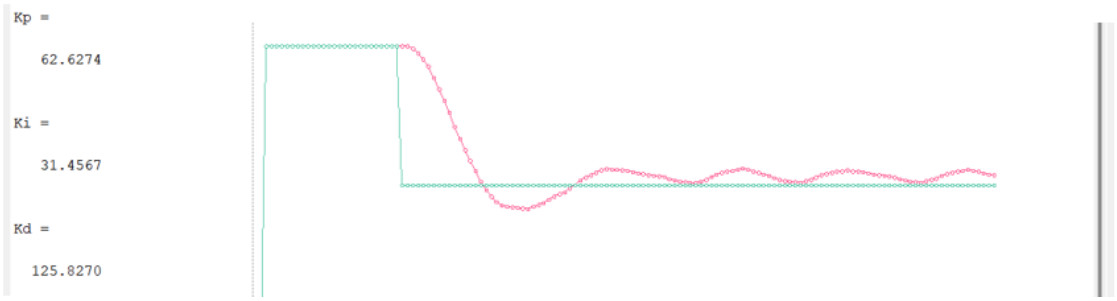


Fig. 60 – Alteração no Kd e Ki

Fonte: Os autores

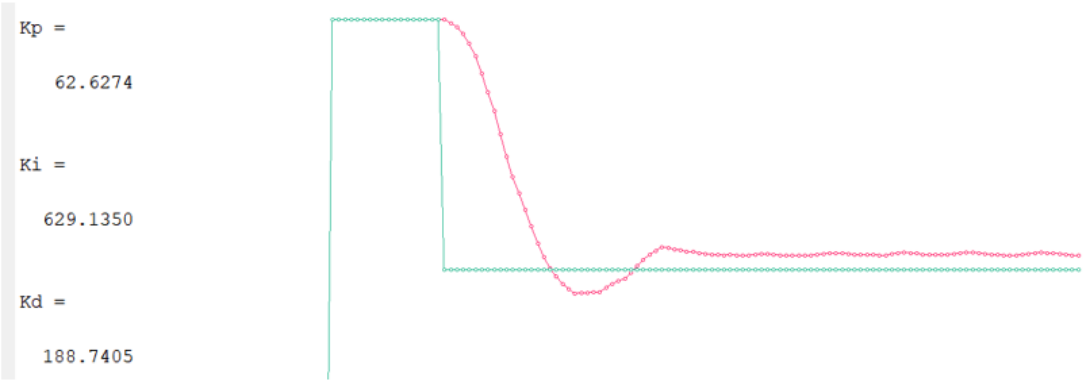


Fig. 61 – Alteração no Kd e Ki

Fonte: Os autores

Apesar de constantes mudanças, o erro em regime permanente nunca apresentou melhora, sempre apresentando oscilações ou um offset em relação ao SetPoint.

Um resultado que reduziu a oscilação foi colocar as mesmas constantes do aquecimento para alta temperatura no sistema de resfriamento, mas o offset permaneceu. Sob orientação do professor, foi aumentada a margem do AntiWindup para 20% e até 30%, a fim de acionar o erro integrativo e aproximar o resultado do SetPoint, mas o código e o resultado obtido (que foi insatisfatório) são mostrados na Fig. 62 e Fig. 63.

```
//ESFRIA
  if(SetPoint < TemperaturaAmbiente){
      digitalWrite(4, LOW);
      digitalWrite(7, HIGH);
    erro = SetPoint - media_ruido; //calcula o erro entre SetP e PV
if (erro> (SetPoint*0.3) || erro < (-SetPoint*0.3)) //reset do termo integrativo (anti-windup)
{
  int_erro = 0; //zera o erro integrativo se o erro for maior que 10% de SetP (em modulo). Isto reduz o
}
else {
  int_erro += erro; //habilita o erro integrativo se o erro for menor que 10% de SetP (em modulo)
}

dif_temp = media_ruido - media_ruido_ant;

        PWM = (erro*KpE) + (int_erro*KiE) - (dif_temp*KdE)+ offset;
       //PWM = (erro*KpE) + (int_erro*KiE);
        media_ruido_ant = media_ruido;
          if(PWM > 255){
            PWM = 255;
          }
          if(PWM <0){
            PWM = 0;
          }
```

Fig. 62 – Código a fim de corrigir o erro

Fonte: Os autores

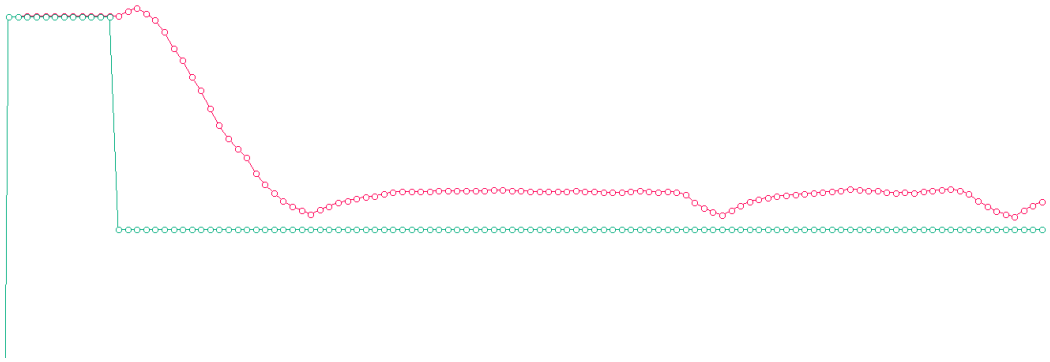


Fig. 63 – Resultado obtido

Fonte: Os autores

Assim, a única forma que encontramos de reduzir esse offset foi inclui-lo no cálculo do PWM. Em cada SetPoint, ao estabilizar a temperatura havia um valor de PWM “residual” que mantinha o valor lido afastado do desejado. Assim, de grau em grau (de 18ºC a 10ºC) foi feita uma “calibração”, somando esse valor ao controlador, conforme Fig. 64.

```
else if(SetPoint < 20 && SetPoint>=18){
  offset = 35;
}
else if(SetPoint < 18 && SetPoint>=17){
  offset = 38;
}
 else if(SetPoint < 17 && SetPoint>=16){
  offset = 42;
}
  else if(SetPoint < 16 && SetPoint>=15){
  offset = 45;
}
else if(SetPoint< 15 && SetPoint >=14){
  offset = 48;
}
else if(SetPoint< 14 && SetPoint >=13){
  offset = 55;
}
else if(SetPoint< 13 && SetPoint >=12){
  offset = 62;
}
else if(SetPoint< 12 && SetPoint >=11){
  offset = 70;
}
else if(SetPoint< 11 && SetPoint >=10){
```

Fig. 64 – Calibração de PWM

Fonte: Os autores

A partir disso, foi feito um teste da temperatura ambiente para 18ºC, como é apresentado na Fig. 65 e Fig. 66.

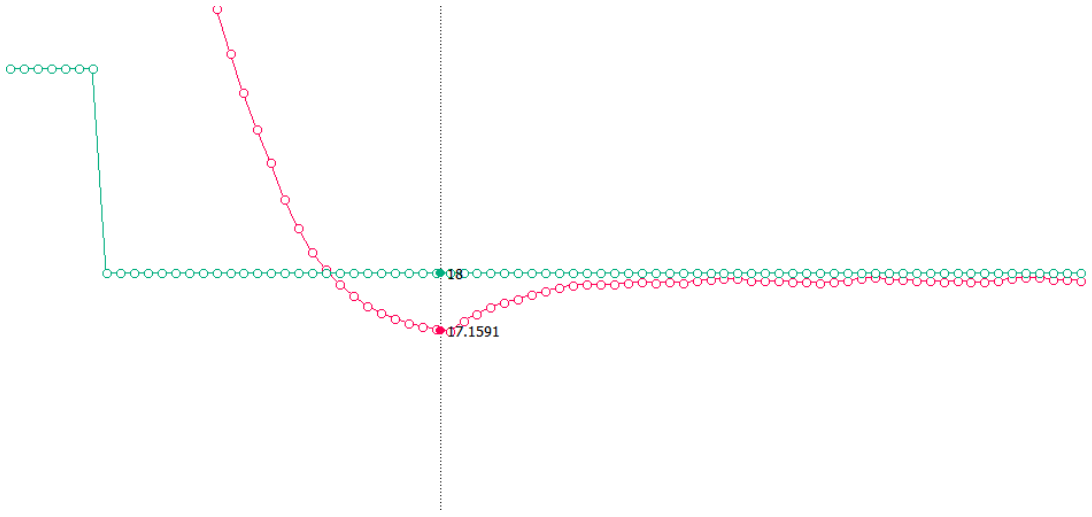


Fig. 65 – Overshoot em 18ºC

Fonte: Os autores

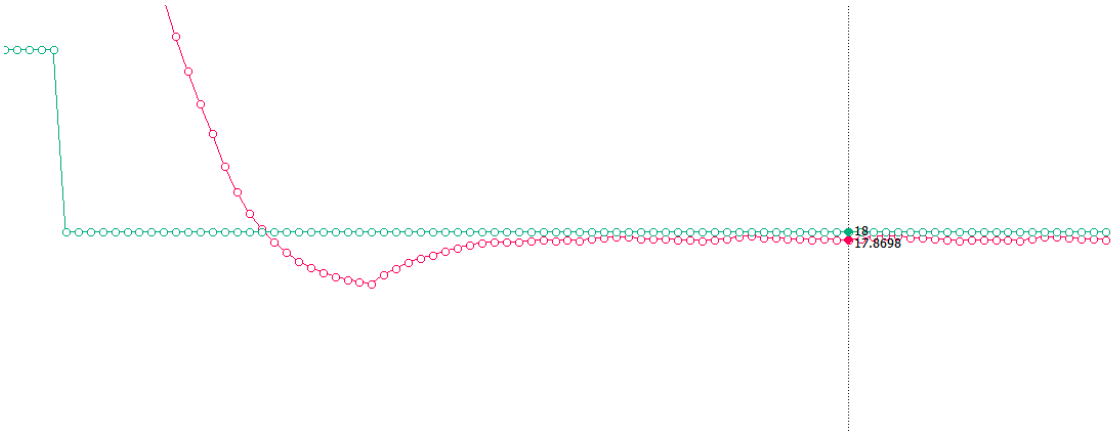


Fig. 66 – Regime permanente em 18ºC

Fonte: Os autores

Então é realizado o mesmo teste, da temperatura ambiente para 15ºC, conforme a Fig. 67 e Fig. 68.

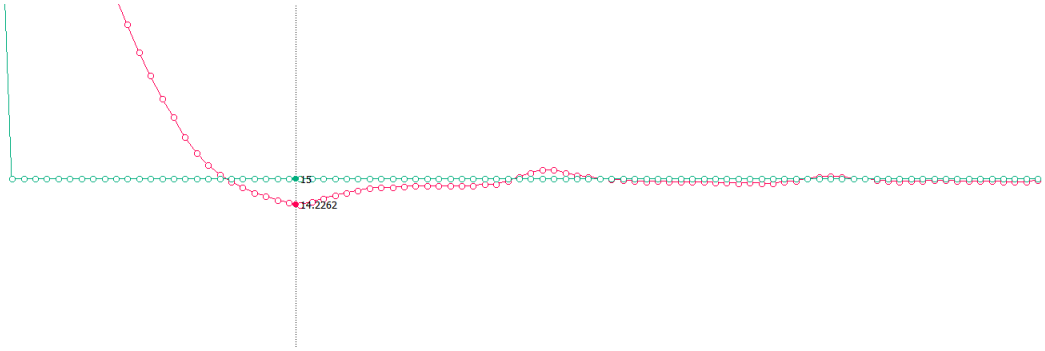


Fig. 67 – Overshoot em 15ºC

Fonte: Os autores

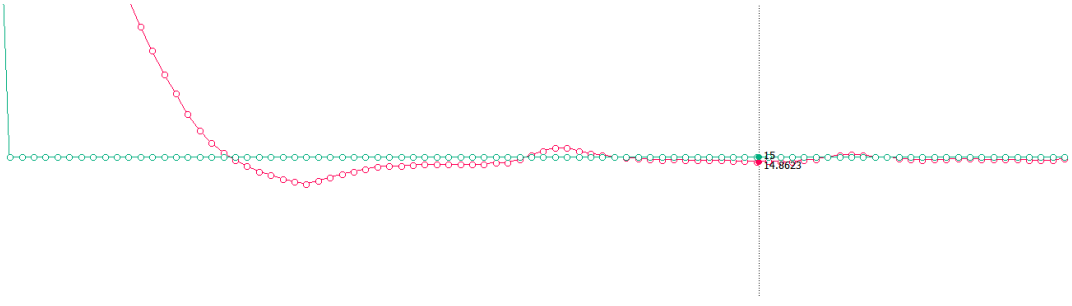


Fig. 68 – Regime permanente em 15ºC

Fonte: Os autores

A Tabela IV mostra as comparações entre o sistema desejado e obtido.

Tabela IV – Comparação entre sistemas no resfriamento

	Overshoot (5%)	Regime Permanente (2%)
Desejado 18ºC	17,10 – 18,9	17,64 – 18,36
Obtido 18ºC	17,16	17,87
Desejado 15ºC	14,25 – 15,75	14,7 – 15,3
Obtido 15ºC	14,22	14,86

Apesar a solução empregada provocar uma relativa melhora, nota-se que em 15ºC o overshoot foi maior que 5%. Isso ocorreu porque o método utilizado no resfriamento é relativamente instável, pois o offset escolhido causa maiores variações (tanto em regime permanente como em sobre-elevação), principalmente em temperaturas mais baixas.

Reitera-se que o método utilizado para o resfriamento não é ideal, pois ao variar as condições iniciais ou a fonte de alimentação, os offset escolhidos podem não ser mais ideais para cada faixa de temperatura. No entanto, melhorou significativamente o sistema, ao contrário das sucessivas alterações de Kp, Ki e Kd.

CONSIDERAÇÕES FINAIS

A temperatura é uma grandeza física muito presente no cotidiano, e assim são utilizados sistemas a fim de monitorar a sua operação. O presente trabalho foi proposto como a montagem e estudo de um sistema de aquecimento e resfriamento da pastilha de efeito Peltier, cuja temperatura é lida pelo sensor analógico TMP36.

Na primeira etapa do trabalho foi realizada a leitura e calibração do sensor, além da escrita do código para aquisição da tensão e conversão em grau Celsius. Foi verificada a estrutura do sensor e sua operação linear.

Na segunda etapa foi desenvolvida uma ponte H por meio de dois relés, que tem como finalidade alterar o sentido da corrente, o que altera o lado quente e lado frio da pastilha. Também foi inserido um MOSFET no circuito a fim de controlar o PWM, limitando a intensidade da corrente. Por fim, todo o sistema foi soldado em uma placa de circuito universal, juntamente vários capacitores (entre as alimentações e o terra) a fim de filtrar o ruído do sistema. Alguns testes mostraram um SNR superior a 100, o que indica um baixo nível de ruído na operação.

Na terceira etapa o sistema foi disposto sobre uma tábua a fim de iniciar o processo de controle. Um nível de ruído significante foi novamente encontrado, oriundo provavelmente pela disposição dos cabos ou até mesmo do desgaste dos componentes. Para eliminá-lo foram aplicados 2 filtros digitais, que o deixaram novamente mais estável.

O sistema de controle foi dividido em três faixas: duas para aquecimento e uma para resfriamento, ambos os sentidos como um sistema de segunda ordem.

No sistema de aquecimento foi necessário um ajuste fino das constantes (inicialmente calculadas no MATLAB), observando a redução do overshoot e do erro em regime permanente. Os resultados obtidos foram satisfatórios, com pouca oscilação.

O sistema de resfriamento, no entanto, apresentou maior instabilidade e dificuldade na definição das constantes, que levou à escolha de um sistema que corrige o offset e mantém a temperatura mais próxima da desejada, sem tanta oscilação. A solução não é ideal, pois é relativamente instável, mas apresentou significativa melhora, ao contrário do ajuste fino das constantes.

Por fim, o sistema de hardware (até a conclusão da segunda etapa) apresentou boa funcionalidade e funciona como proposto: a variação do PWM altera de fato a intensidade da corrente e o sistema é capaz de alterar o lado quente e frio da Peltier por meio da Ponte H. O sistema de controle PID funciona como proposto: a temperatura da pastilha se direciona para o SetPoint inserido, ainda que com alguma oscilação. O controle do aquecimento foi construído somente alterando as constantes do PID, enquanto que no resfriamento foi necessário um ajuste manual e não muito elegante, mas que supriu as necessidades imediatas do projeto. Reitera-se também que todos os testes funcionam idealmente partindo da temperatura ambiente para o SetPoint, o que significa que uma variação direta (por exemplo) de 60ºC para 15ºC ou de 15ºC para 60ºC pode não produzir os mesmos resultados apresentados.

REFERÊNCIAS

[1] Bonfim, Marlio. **Slides Instrumentação Eletrônica** – TE331. Disponível em <http://www.eletrica.ufpr.br/marliob/te149/aula1.pdf> Acesso em 07/11/2022.

[2] Conhecimento Científico. **O que é temperatura: conceito, tipos de medição e escalas**. Disponível em <https://conhecimentocientifico.com/o-que-e-temperatura/> Acesso em 07/12/2022.

[3] Filipe Flop. **O que é Arduino**. Disponível em < https://www.filipeflop.com/blog/o-que-e-arduino/> Acesso em 05/11/2022.

[4] Vida de Silício. LM35 **Sensor de Temperatura**. Disponível em < https://www.vidadesilicio.com.br/produto/lm35-sensor-de-temperatura/> Acesso em 04/11/2022.

[5] Casa da Robótica**. Sensor de Temperatura CI DS18b20 18B20.** Disponível em < https://www.casadarobotica.com/sensores-e-modulos/sensores/temperatura/sensor-de-temperatura-ci-ds18b20-18b20> Acesso em 04/11/2022.

[6] **Smart Kits. Sensor de Temperatura TMP36.** Disponível em < https://www.smartkits.com.br/sensor-de-temperatura-tmp36> Acesso em 04/11/2022.

[7] Blog Master Walker. **Como usar com Arduino – Sensor de Temperatura TMP36**. Disponível em < https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-sensor-de-temperatura-tmp36> Acesso em 04/11/2022.

[8] Components 101. TMP36 – **Temperature Sensor**. Disponível em <https://components101.com/sensors/tmp36-temperature-sensor> Acesso em 05/11/2022.

[9] Alldatasheet. **TMP36 Datasheet PDF – Analog Devices**. Disponível em <https://pdf1.alldatasheet.com/datasheet-pdf/view/49108/AD/TMP36.html> Acesso em 05/11/2022.

[10] Project Hub. **Using A Temp Sensor With Arduino TMP36 Temperature Sensor.** Disponível em < https://create.arduino.cc/projecthub/m_karim02/using-a-temp-sensor-with-arduino-tmp36-temperature-sensor-1e1d0b > Acesso em 05/11/2022.

[11] Steq. **Tecnologia Peltier: Economia de Energia e Precisão** de **Temperatura em Câmaras Climáticas**. Disponível em < https://www.steq.com.br/tecnologia-peltier-economia-de-energia-e-presica-de-temperatura-em-camaras-climaticas/ > Acesso em 21/11/2022.

[12] Manual da Eletrônica. **Ponte H – O que é e como funciona**. Disponível em <https://www.manualdaeletronica.com.br/ponte-h-o-que-e-como-funciona/> Acesso em 21/11/2022.

[13] CitiSystems. **O que é PWM e para que serve?** Disponível em <https://www.citisystems.com.br/pwm/> Acesso em 09/12/2022.

[14] YouTube. **TMP Temperature Sensor Arduino on TinkerCad**. Disponível em < https://www.youtube.com/watch?v=sFmuz4mU5w8 > Acesso em 05/11/2022.

[15] Sistemas interativos com Arduino. **Tutorial: Usando sensor de temperatura com Arduino.** Disponível em < https://www.sistemasinterativoscomarduino.net/2020/06/projeto-usando-sensores-de-temperatura.html > Acesso em 05/11/2022.

[16] Cap Sistema. **Guia de uso do sensor de temperatura analógico TMP36 com Arduino**. Disponível em < https://capsistema.com.br/index.php/2020/11/09/guia-de-uso-do-sensor-de-temperatura-analogico-tmp36-com-arduino/ > Acesso em 05/11/2022.

[17] JV Eletrônica. **Ponte H com relés para reversão de motor DC**. Disponível em <https://www.youtube.com/watch?v=srkXGgQwa3U&t=10s> Acesso em 09/12/2022.

ANEXO I

Código do funcionamento do sensor TMP36

```
#define Pino A0 //Define a porta A0 como entrada

//Esquerda: VCC, meio: sinal, direita: GND

void setup() {
Serial.begin(9600);
analogReference(EXTERNAL); //Muda a V interna de ref do Arduino para 1.22V

}

void loop() {
int leitura = analogRead(Pino); //Lê a entrada
float tensao = leitura * 1.22; //multiplica por 1.22V - Ref externa
tensao /= 1024.0; //Conversão AD

float Temperatura = (tensao - 0.5) *100; //Subtraímos 0,5 para ajustar a faixa de medição do sensor
Serial.print(Temperatura); //Printa a temperatura
Serial.println(" Graus");
Serial.print(leitura);
Serial.println(" leitura");
delay (1000);
}
```

ANEXO II

```
//Esquerda: VCC, meio: sinal, direita: GND


#define Pino A0 //Define a porta A0 como entrada
int media[100];
float temp=0.0;
int PWM = 0;  //variável PWM (0 a 255);
unsigned int Ts = 500;  //tempo de amostragem em
unsigned long tempo = 0; //tempo de contagem em ms
int Aux_PWM = 0;


void setup() {
Serial.begin(9600);
analogReference(EXTERNAL);
Serial.setTimeout(50); //limita o tempo de espera de caracteres de entrada da serial
pinMode(3, OUTPUT);//define pino 3 como saida PWM para Gate do MOSFET que irá acionar uma carga
analogWrite (3, 0); //escreve o valor 0 na saída PWM
pinMode(4, OUTPUT);
pinMode(7, OUTPUT);


for (int i=0; i<100; i++){
int leitura = analogRead(Pino); //Lê a entrada
float tensao = leitura * 1.4; //multiplica por 1,4V
tensao /= 1024.0; //Conversão AD
media[i] = (tensao - 0.5) *100; //Subtraímos 0,5 para ajustar a faixa de medição do sensor
temp = temp+media[i];
delay(10);
}
float TemperaturaAmbiente = temp/100.0;
```

```
Serial.print ("Temperatura Ambiente: ");

Serial.print(" ");//envia um espaço de separação dos dados

Serial.println(TemperaturaAmbiente);

Serial.println("Digite o PWM de 0 a 255 para esfriar; 256 a 511 para esquentar: ");




//while (Serial.available() == 0) {}

//

//if(Serial.available()>0){

//PWM = constrain(PWM, 0, 511);  //limita os valores da temperatura entre 5 e 75

//PWM = Serial.parseInt(); //lê caracteres numéricos ASCII da serial e converte em um número inteiro

//Aux_PWM = PWM;

//}


}


void loop(){

tempo = millis();      //armazena o valor do tempo na variável "tempo" (em ms)

int leitura = analogRead(Pino); //Lê a entrada

float tensao = leitura * 1.4; //multiplica por 1,4V

tensao /= 1024.0; //Conversão AD


float Temperatura = (tensao - 0.5) *100; //Subtraímos 0,5 para ajustar a faixa de medição do sensor

Serial.print(Temperatura); //Printa a temperatura

Serial.print(" ");//envia um espaço de separação dos dados

Serial.println(PWM);
```

```
if (Serial.available() > 1) {  //verifica se há dados na serial
Aux_PWM = Serial.parseInt(); //lê caracteres numéricos ASCII da serial e converte em um número inteiro
Aux_PWM = constrain(Aux_PWM, 0, 511);  //limita os valores da variável PWM entre 0 a 511

if (Aux_PWM <= 255){

  PWM = Aux_PWM;

  digitalWrite(4, HIGH);
  digitalWrite(7, LOW);
  }
if(Aux_PWM > 255){
  PWM = Aux_PWM - 256;
  digitalWrite(4, LOW);
  digitalWrite(7, HIGH);
  }

}
analogWrite (3, PWM);  //atribui o valor do PWM à porta de saída 3
Serial.read(); //lê e limpa dados restantes da serial
//delay(1000);
while ((tempo+Ts) > millis()) {}     //espera o tempo necessário para completar o tempo de amostragem Ts
}
```

ANEXO III

```
//Esquerda: VCC, meio: sinal, direita: GND
#define Pino A0 //Define a porta A0 como entrada
int offset = 0;
int media[100];
float temp=0.0;
float temp2 = 0.0;
float media_ruido;
int media2[10];
int PWM = 0; //variável PWM (0 a 255);
unsigned int Ts = 500; //tempo de amostragem em ms
unsigned long tempo = 0; //tempo de contagem em ms
int Aux_PWM = 0;
float SetPoint;
float TemperaturaAmbiente;
float erro;
float dif_temp = 0;
float media_ruido_ant;
int int_erro = 0;

//Aquece

//float LA = 1.6203703;
//float TA = 27.13674855;
//float KpA = (1.2/KA)*(TA/LA);
//float KiA = KpA*(0.5/2*LA);
//float KdA = KpA*(LA/0.5);
```

```
//float KpA = 1.2*(TA/LA);
//float KiA = 0.5/(LA);
//float KdA = 0.5*(LA);

//Esfria
//float KE = -0.20553;
//float LE = 2.009138;
//float TE = 21.55101913;
//float KpE = (1.2/KE)*(TE/LE);
//float KiE = KpE*(0.5/2*LE);
//float KdE = 0.5*KpE*(LE/0.5);

      float LE = 1.8122657;
      float TE = 23.216683;
      float KE = -0.56834;
      float KpE = (1.2/KE)*(TE/LE);
      float KiE = 4;
      float KdE = KpE*(LE/0.5);

//float KpE = -32.31926;
//float KiE = -(1/14.8728);
//float KdE = -3.20706;

//float KpE = 20/KE;
//float Tp1 = 23.24;
//float KiE = 2*((KpE*0.5)/Tp1);

void setup() {
Serial.begin(9600);
analogReference(EXTERNAL);
```

```
Serial.setTimeout(50); //limita o tempo de espera de caracteres de entrada da serial

pinMode(3, OUTPUT);//define pino 3 como saida PWM para Gate do MOSFET que irá acionar uma carga
analogWrite (3, 0); //escreve o valor 0 na saída PWM

pinMode(4, OUTPUT);
pinMode(7, OUTPUT);

for (int i=0; i<100; i++){
  int leitura = analogRead(Pino); //Lê a entrada
  float tensao = leitura * 1.4; //multiplica por 1,4V
  tensao /= 1024.0; //Conversão AD
  media[i] = (tensao - 0.5) *100; //Subtraímos 0,5 para ajustar a faixa de medição do sensor
  temp = temp+media[i];
}

TemperaturaAmbiente = temp/100.0;
SetPoint = TemperaturaAmbiente;
media_ruido = TemperaturaAmbiente;
media_ruido_ant = TemperaturaAmbiente;
}

void loop(){
tempo = millis(); //armazena o valor do tempo na variável "tempo" (em ms)

//Serial.print(auxiliar);

if (Serial.available()>0){
    SetPoint = Serial.parseInt();
```

```
    SetPoint = constrain(SetPoint, -5, 70);
    Serial.read();
}

//Filtro de sobreamostragem
temp2 = 0;
for (int k=0; k<10; k++){
  int leitura2 = analogRead(Pino); //Lê a entrada
  float tensao2 = leitura2 * 1.4; //multiplica por 1,4V
  tensao2 /= 1024.0; //Conversão AD
  media2[k] = (tensao2 - 0.5) *100; //Subtraímos 0,5 para ajustar a faixa de medição do sensor
  temp2 = temp2+media2[k];
}
float Temperatura = temp2/10.0;

//alfa = (n-1)/n, em que n = 5.
//EWMA = EWMA * alfa+(1-alfa)*floatAD
media_ruido = media_ruido*0.8 + (0.2)*Temperatura;

//Controlador PID
erro = SetPoint - media_ruido; //calcula o erro entre SetP e PV
if (erro> (SetPoint*0.05) || erro < (-SetPoint*0.05)) //reset do termo integrativo (anti-windup)
{
  int_erro = 0; //zera o erro integrativo se o erro for maior que 10% de SetP (em modulo). Isto reduz o overshoot
}
else {
  int_erro += erro; //habilita o erro integrativo se o erro for menor que 10% de SetP (em modulo)
}
```

```
dif_temp = media_ruido - media_ruido_ant;

//esquenta
  if(SetPoint > TemperaturaAmbiente){
       if(SetPoint <= 42){
       float KA = 0.56834;
       float KpA = 29;
       float KiA = 5.45;
       float KdA = 125;
         digitalWrite(4, HIGH);
         digitalWrite(7, LOW);
       PWM = (erro*KpA) + (int_erro*KiA) - (dif_temp*KdA);
       media_ruido_ant = media_ruido;
         if(PWM > 255){
           PWM = 255;
         }
         if(PWM <0){
           PWM = 0;
         }
   analogWrite (3, PWM); //atribui o valor do PWM à porta de saída 3
  }

       if(SetPoint > 42){
        float LA = 1.8122657;
        float TA = 23.216683;
       float KA = 0.56834;
       float KpA = (1.2/KA)*(TA/LA);
       float KiA = 4;
       float KdA = KpA*(LA/0.5);
```

```
        digitalWrite(4, HIGH);
        digitalWrite(7, LOW);
      PWM = (erro*KpA) + (int_erro*KiA) - (dif_temp*KdA);
      media_ruido_ant = media_ruido;
        if(PWM > 255){
          PWM = 255;
        }
        if(PWM <0){
          PWM = 0;
        }
  analogWrite (3, PWM); //atribui o valor do PWM à porta de saída 3
 }
 }

//ESFRIA

//fonte a 11 V

  if(SetPoint < TemperaturaAmbiente){

     digitalWrite(4, LOW);
     digitalWrite(7, HIGH);

     if(SetPoint < 24 && SetPoint>=22){
       offset = 22;
     }
     else if(SetPoint < 22 && SetPoint>=20){
       offset = 27;
     }
     else if(SetPoint < 20 && SetPoint>=18){
```

```
  offset = 35;
}
else if(SetPoint < 18 && SetPoint>=17){
  offset = 38;
}
 else if(SetPoint < 17 && SetPoint>=16){
  offset = 42;
}
  else if(SetPoint < 16 && SetPoint>=15){
  offset = 45;
}
else if(SetPoint< 15 && SetPoint >=14){
  offset = 48;
}
else if(SetPoint< 14 && SetPoint >=13){
  offset = 55;
}
else if(SetPoint< 13 && SetPoint >=12){
  offset = 62;
}
else if(SetPoint< 12 && SetPoint >=11){
  offset = 70;
}
else if(SetPoint< 11 && SetPoint >=10){
  offset = 72;
}
else{
  offset = 0;
  }
```

```
      PWM = (erro*KpE) + (int_erro*KiE) - (dif_temp*KdE)+ offset;
     //PWM = (erro*KpE) + (int_erro*KiE);
      media_ruido_ant = media_ruido;
        if(PWM > 255){
          PWM = 255;
        }
        if(PWM <0){
          PWM = 0;
        }
    analogWrite (3, PWM); //atribui o valor do PWM à porta de saída 3
 }

Serial.print(media_ruido); //Printa a temperatura
Serial.print(" ");//envia um espaço de separação dos dados
Serial.print(PWM);
Serial.print(" ");//envia um espaço de separação dos dados
Serial.println(SetPoint);
Serial.println(offset);
//Serial.print(int_erro);
//Serial.print(" ");
//Serial.println(erro);

//Serial.print(" ");
if(Temperatura > 70){
digitalWrite(4, LOW);
digitalWrite(7, LOW);
}

Serial.read(); //lê e limpa dados restantes da serial
//delay(1000);
```

```
while ((tempo+Ts) > millis()) {} //espera o tempo necessário para completar o tempo de amostragem Ts

}
```