INSTITUTO FEDRAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO
CAMPUS SÃO JOÃ DA BOA VISTA
ENGENHARA DE CONTROLE E AUTOMAÇÃO
LABORATÓRIO DE LINGUAGEM, PROGRAMAÇÃO E SENSORES

CONTROLE DE TEMPERATURA E LUMINOSIDADE DE UMA ESTUFA

Camila da Silva Bertazzi
Cezar Henrique Pimenta Gregório
Emerson Castelhano Voltarelli
Fernanda Chiod Luzetti Lima
Gustavo Matavelli

SÃO JOÃO DA BOA VISTA
JUNHO DE 2019

Camila da Silva Bertazzi
Cesar Henrique Pimenta Gregório
Emerson Castelhano Voltarelli
Fernanda Chiod Luzetti Lima
Gustavo Matavelli

Controle De Temperatura E Luminosidade De Uma Estufa

Trabalho apresentado na disciplina de LB7, LB8, CLP e CPR, do curso de Engenharia de Controle e Automação do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, como requisito para conclusão das referidas disciplinas.
Professores: Emerson dos Reis, Daniel Espanhol Razera, Muriell de Rodrigues e Freire e André Luiz Gontijo.

São João da Boa Vista
Junho de 2019

RESUMO

O uso de estufas está cada vez mais comum entre as pessoas, seja por falta de espaço ou tempo para se ter/cuidar de uma horta ou até mesmo por hobby. Por mais que seja terapêutico e divertido mexer em uma horta, não é possível controlar os agentes ideais para uma boa plantação, um solo sem nutrientes e sem a humidade correta pode fazer com que as plantas não crescem corretamente, assim como não se ter uma boa iluminação e um bom controle de temperatura, além de que ainda é possível se contrair pragas. Visando corrigir esses problemas, opta-se pela confecção de uma estufa, onde se pode controlar alguma dessas variáveis, como por exemplo a iluminação e a temperatura ideal para o crescimento saudável das plantas, além de não precisar se preocupar com possíveis pragas que poderiam se instalar nas plantas. Tendo em mente as variáveis a serem controladas e uma maquete simulando a estufa, estuda-se maneiras de se fazer esse controle, para isso realiza-se testes para ver se o aquecedor e a iluminação satisfazem as condições de necessidade das plantas, para posteriormente, fazer o controle automático dessas variáveis utilizando softwares e equipamentos próprios de controle. Após pronta, pode-se utilizar a estufa para o plantio, seria interessante observar o crescimento de uma mesma planta dentro e fora da estufa, para notar, se nesse caso, a estufa ajudaria a planta a se desenvolver de forma mais rápida e saudável, porém, devido ao tempo de projeto disponível, não realizou-se o teste em si.

Palavras-chaves: Estufa. Planta. Controle. Aquecedor. Iluminação.

ABSTRACT

The use of greenhouses are increasing commonly among people, either because of lack of space or time to take care of a vegetable garden or even for a hobby. Although the therapeutic and fun gardening work, it is not possible to control the ideal variables for a good plantation, a soil without nutrients and without the correct humidity can cause the plants not to grow properly, as well as not to have a good lighting and good temperature control, and it is still possible to contract pests. In order to correct these problems, a greenhouse is chosen, where one can control some of these variables, such as the lighting and the ideal temperature for healthy plant growth, besides not having to worry about possible pests that could installed in plants. Keeping in mind the variables to be controlled and a mock-up model simulating the greenhouse, with the studie of ways to make this control, testing to see if the heater and the lighting satisfy the conditions of need of the plants, among the development of an automatic control of these variables using software and control equipment. After the greenhouse is ready to be used for planting, it would be interesting to observe the growth of the same plant inside and outside the greenhouse, to note if in this case the greenhouse would help the plant to growth in a faster and healthier way, however, due to the available project time, the test itself was not performed.

Keywords: Greenhouse. Plant. Control. Heater. Lighting.

Sumário

LISTA DE FIGURAS

LISTA DE TABELAS

1- INTRODUÇÃO

A soma dos processos de crescimento e desenvolvimento da planta recebe o nome de ontogênese. Dentre os fatores que podem interferir no desenvolvimento de um vegetal podem ser:

- Internos, como alterações genéticas, taxas de hormônios e vitaminas.
- Externos, como intensidade da iluminação e temperatura do ambiente em que a planta se encontra (KAMLER, 2002).

O presente trabalho busca expor o desenvolvimento do projeto de uma estufa que visa o controle automático de dois fatores externos ao desenvolvimento de um vegetal, que estão expostos mais à frente.

1.1. A Influência da Temperatura no Cultivo de Plantas

A temperatura exerce grande influência na atividade fisiológica de todos os seres vivos, por controlar as taxas de reações metabólicas nas células.

No caso das plantas, a temperatura é fundamental para o desenvolvimento dos vegetais, podendo interferir de forma benéfica ou maléfica no desenvolvimento da planta. A temperatura do ar e do solo afetam todos os estágios do processo do crescimento vegetal (EMBRAPA, 2014).

Segundo Sentelhas et al (2009, p. 25):

> As culturas possuem limites térmicos mínimo, ótimo e máximo, os cultivos tropicais exigem altas temperaturas o ano todo. Por outro lado, o centeio tem a necessidade de baixa temperatura e pode até suportar as temperaturas de congelamento […]. Geralmente, as altas temperaturas não são tão destrutivas para as plantas como as baixas temperaturas.

1.2. A influência da Luz no Cultivo de Plantas

A luminosidade tem papel fundamental na vida das plantas, interferindo na fotossíntese e, portanto, no desenvolvimento de cada espécie (KLUGE et al, 2015).

A fotossíntese permite a entrada de energia na cadeia alimentar. Através desse processo, os seres vivos autótrofos fotossintetizantes capturam energia luminosa e transformam em energia química para ligar átomos e formar moléculas energéticas (GOVINDJEE, 1975).

No entanto, não é possível ter sol abundante em todo o ano, por conta de solstício e equinócio, além disso, existem lugares que recebem pouca luz solar ao longo do ano. A iluminação artificial é uma forma de substituir a luz natural. Um LED pode se assemelhar a luz do sol em termos de espectro e intensidade, garantindo o crescimento saudável das plantas, visto que ele possui um amplo espectro de cores que variam do vermelho ao violeta (SANTOS; PEREIRA, 2002).

1.3. Estufas Elétricas

A função principal da estufa é proteger as plantas e manter as melhores condições para o perfeito desenvolvimento do cultivo (MUIJZENBERG, 1980).

Numa estufa elétrica a fonte do calor é dada pela transformação da energia elétrica em energia térmica, que se acumula dentro de um ambiente fechado (REVISTA AGROPECUÁRIA, 2018).

1.4. Arduino

Arduino é uma plataforma de prototipagem eletrônica de hardware livre e de placa única, projetada com um microcontrolador Atmel AVR com suporte de entrada/saída embutido, uma linguagem de programação padrão, a qual tem origem em Wiring, e é essencialmente C/C++. O objetivo do projeto é criar ferramentas que são acessíveis, com baixo custo, flexíveis e fáceis de se usar por principiantes e profissionais. Principalmente para aqueles que não teriam alcance aos controladores mais sofisticados e ferramentas mais complicadas (ARDUINO, 2019).

O Arduino utilizado no projeto foi o Arduino Uno, cujas entradas, saídas e funções adicionais estão ilustradas pela Figura 1 [9] abaixo.

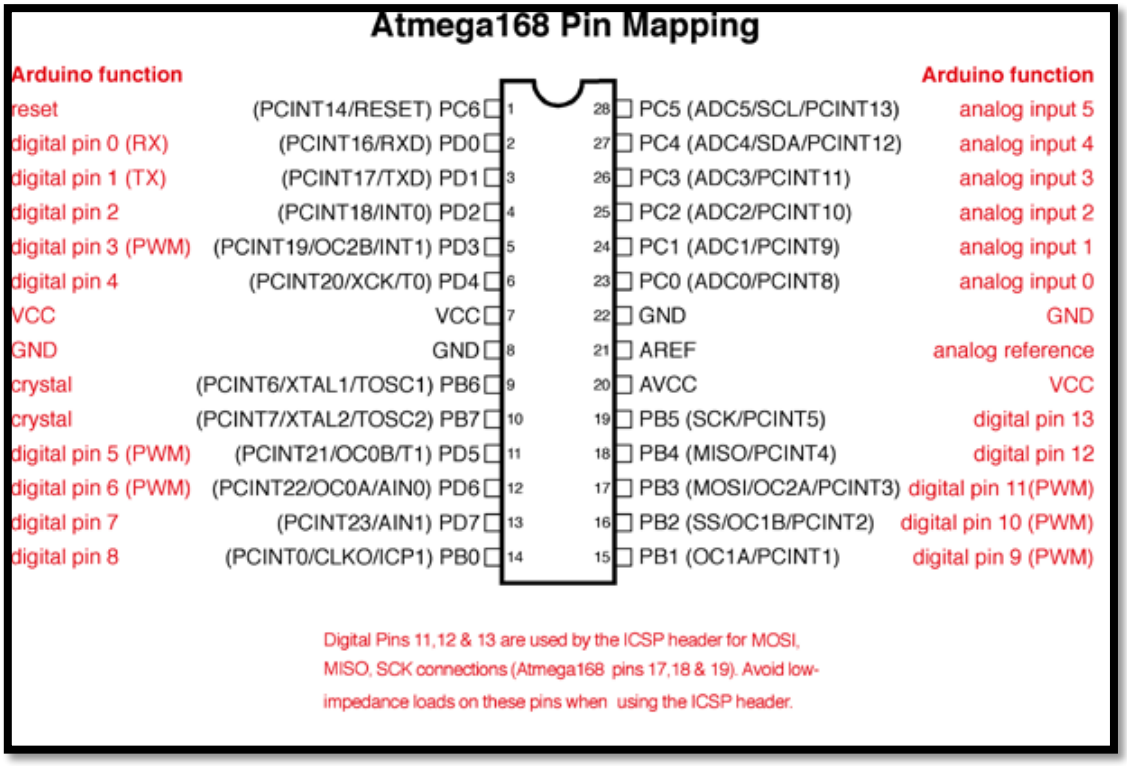


Figura 1 - Diagrama de Pinos do Arduino Uno.

1.5. LDR

O LDR (*Light Dependent Resistor*), ilustrado pela Figura 2 [11], é um componente eletrônico passivo do tipo resistor variável, mais especificamente, é um resistor cuja resistência varia conforme a intensidade da luz que incide sobre ele. Tipicamente, à medida que a intensidade da luz aumenta, a sua resistência diminui.

O LDR é construído a partir de material semicondutor com elevada resistência elétrica. Quando a luz que incide sobre o semicondutor tem uma frequência suficiente, os fótons que incidem sobre o semicondutor liberam elétrons para a banda condutora que irão melhorar a sua condutividade e assim diminuir a resistência (RADIO ELETRONICS, 2013).

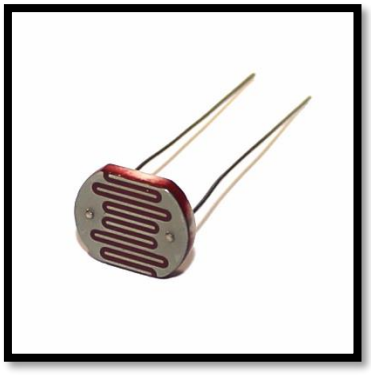


Figura 2 - LDR.

1.6. LM35

Os dispositivos da série LM35 são sensores de temperatura de circuito integrado de precisão, com uma tensão de saída linear proporcional à temperatura em graus Celsius. Os dispositivos LM35 não requerem qualquer calibração externa para fornecer precisões típicas de ± ¼ °C à temperatura ambiente e ± ¾ °C ao longo de uma gama de temperaturas entre -55 °C e 150 °C.

A baixa impedância de saída, saída linear e calibração inerente precisa do dispositivo LM35 torna a leitura e controle de circuitos especialmente fácil. Como o dispositivo LM35 utiliza apenas 60 µA da fonte, tem baixo auto aquecimento de menos de 0,1 °C, evitando influências na leitura da temperatura por auto aquecimento (TEXAS INSTRUMENTS, 2017). O diagrama de pinos do LM35 está ilustrado pela Figura 3 [Anexo III] abaixo.

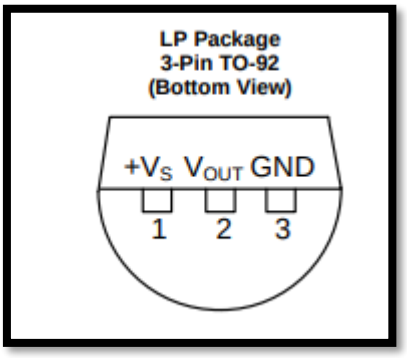


Figura 3 - Diagrama de pinos do LM35 com vista inferior.

1.7. TRIAC

Um TRIAC, *(Triode for Alternating Current)* é um componente eletrônico equivalente a dois retificadores controlados de silício (SCR/tiristores) ligados em antiparalelo e com o terminal de disparo (ou gatilho - *gate*) ligados juntos. Este tipo de ligação resulta em uma chave eletrônica bidirecional que pode conduzir a corrente elétrica nos dois sentidos.

Um TRIAC pode ser disparado por uma corrente alternada aplicada no terminal de disparo (*gate*). Uma vez disparado, o dispositivo continua a conduzir até que a corrente elétrica caia abaixo do valor de corte, como o valor da tensão final da metade do ciclo de uma corrente alternada. Também pode-se controlar o início da condução do dispositivo, aplicando um pulso em um ponto pré-determinado do ciclo de corrente alternada, o que permite controlar a percentagem do ciclo que estará alimentando a carga (também chamado de controle de fase) (SCILLC, 2005). A Figura 4 [12] ilustra simbologia do TRIAC.

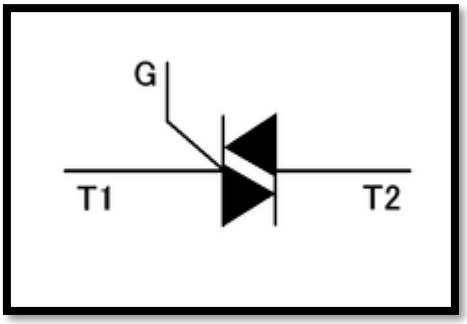


Figura 4 - Simbologia de um TRIAC.

1.8. Opto acopladores

O opto acoplador é um componente eletrônico que transfere sinais elétricos entre dois circuitos isolados usando luz. Opto acopladores impedem que altas tensões afetem o sistema que recebe o sinal.

Um opto acoplador contém uma fonte emissora de luz, quase sempre um diodo emissor de luz infravermelho (LED), que converte o sinal elétrico de entrada em luz, um canal dielétrico e um fotossensor, que detecta a luz que entra e gera energia elétrica diretamente, ou modula a corrente elétrica que flui de uma fonte de alimentação externa (KINCAID, 2010). O diagrama elétrico de um opto acoplador é ilustrado pela Figura 5 [13].

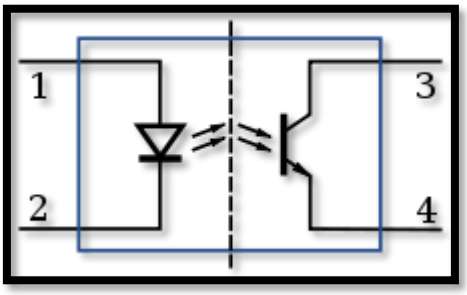


Figura 5 - Diagrama elétrico de um opto acoplador.

2- MATERIAIS

2.1- BANCADA

- Aquário simulando uma estufa;
- Estrutura de madeira para apoio;
- Tampa;
- Arduino;
- Aquecedor (chapinha de cabelo);
- 1 dissipador de calor;
- 2 ventoinhas;
- Terra;
- Plantas artificiais;
- 1 fonte regulável de bancada;
- 1 LM35.

2.2- AQUECEDOR

- 1 resistor de 220 Ω;
- 2 resistores de 330 Ω;
- 1 resistor de 39 Ω;
- 1 capacitor de 10 nF;
- 1 BT137;
- 1 MOC 3021;
- 1 resistência de chapinha 40 W.

2.3- ILUMINAÇÃO

- Fitas de LED;
- 1 4N25;
- 1 resistor de 220 Ω;
- 1 resistor de 1 kΩ;
- 1 TIP31;
- 2 LDR;
- 2 resistores de 10 kΩ.

2.4- RESFRIADOR

- 1 ventoinha;
- 1 4N25;
- 1 resistor de 220 Ω;
- 1 resistor de 1 kΩ;
- 1 TIP31.

3- PROCEDIMENTO METODOLÓGICO

3.1- Circuitos e interfaces

3.1.1- Sensores

Devido ao uso de sensores lineares, de fácil aplicação para o Arduino, a ligação para o LM35 foi direta, como pode ser vista na Figura 6 abaixo.

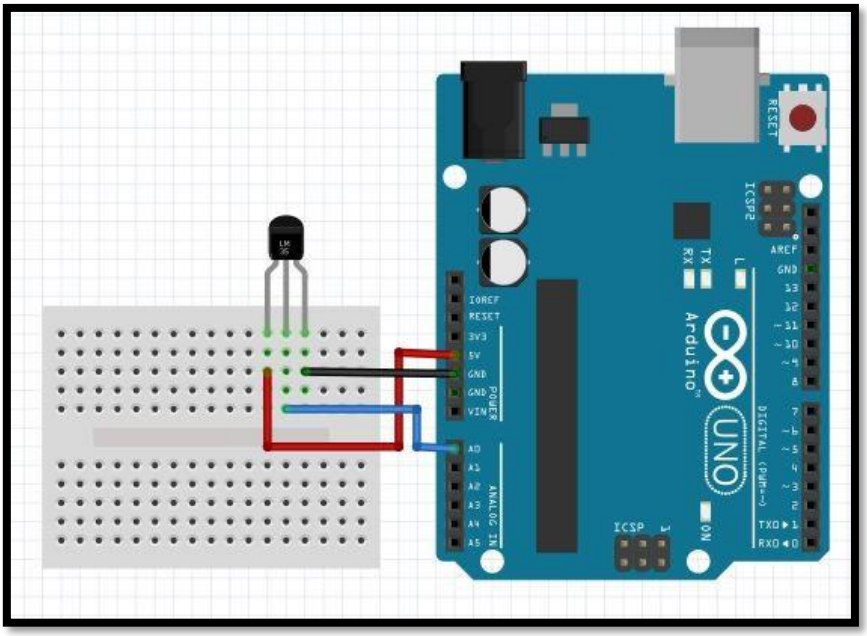


Figura 6 - Circuito para leitura de temperatura pelo Arduino com o LM35.

Para os LDR, a ligação necessita de um divisor de tensão para que o LDR não cause um curto entre a alimentação (5V) e o terra (GND) do Arduino, como pode ser visto na Figura 7 a seguir.

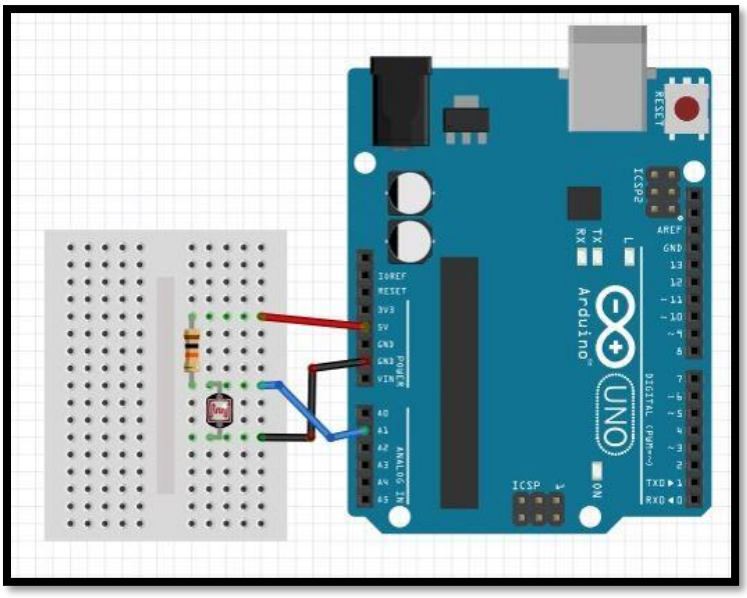


Figura 7 - Circuito para leitura de temperatura pelo Arduino com o LDR.

3.1.2- Iluminação

O circuito de iluminação foi utilizado um circuito de controle pelo PWM do Arduino, onde um opto acoplador ativa a base de um transistor que controla a potência nas fitas de LED, representado pelo LED verde na Figura 8 abaixo.

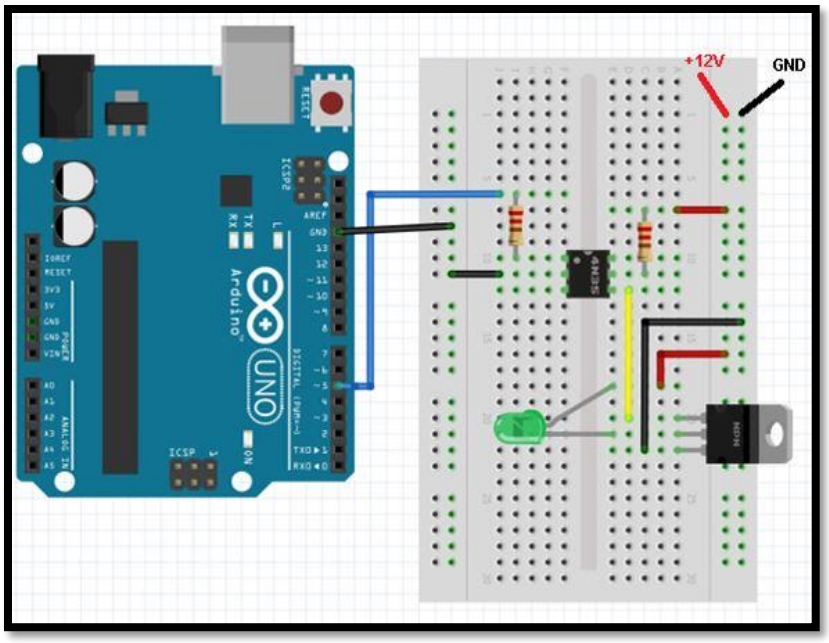


Figura 8- Circuito de controle PWM de luminosidade.

3.1.3- Resfriador

Para o resfriador, com o atuador sendo uma ventoinha, utilizou-se do mesmo driver de potência do sistema de iluminação, como pode ser visto na Figura 9 abaixo.

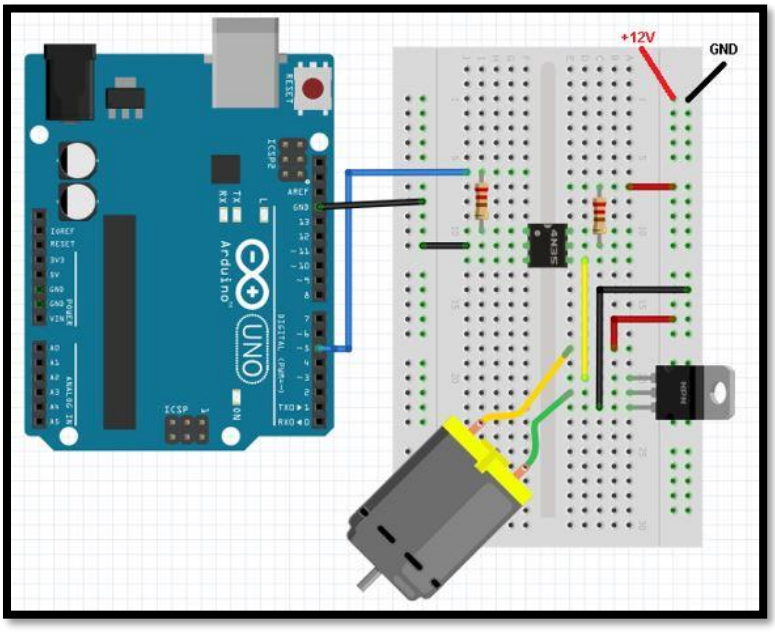


Figura 9 - Driver de potência do resfriador DC.

3.1.4- Aquecedor

O aquecedor sendo representado por uma carga resistiva AC, foi controlado por um circuito de chaveamento de um Triac, direcionado pelo opto acoplador MOC3021, onde o Arduino chavear uma porcentagem das ondas por um PWM de 6Hz (período de 166,66 ms), como pode ser visto na Figura 10 a seguir.

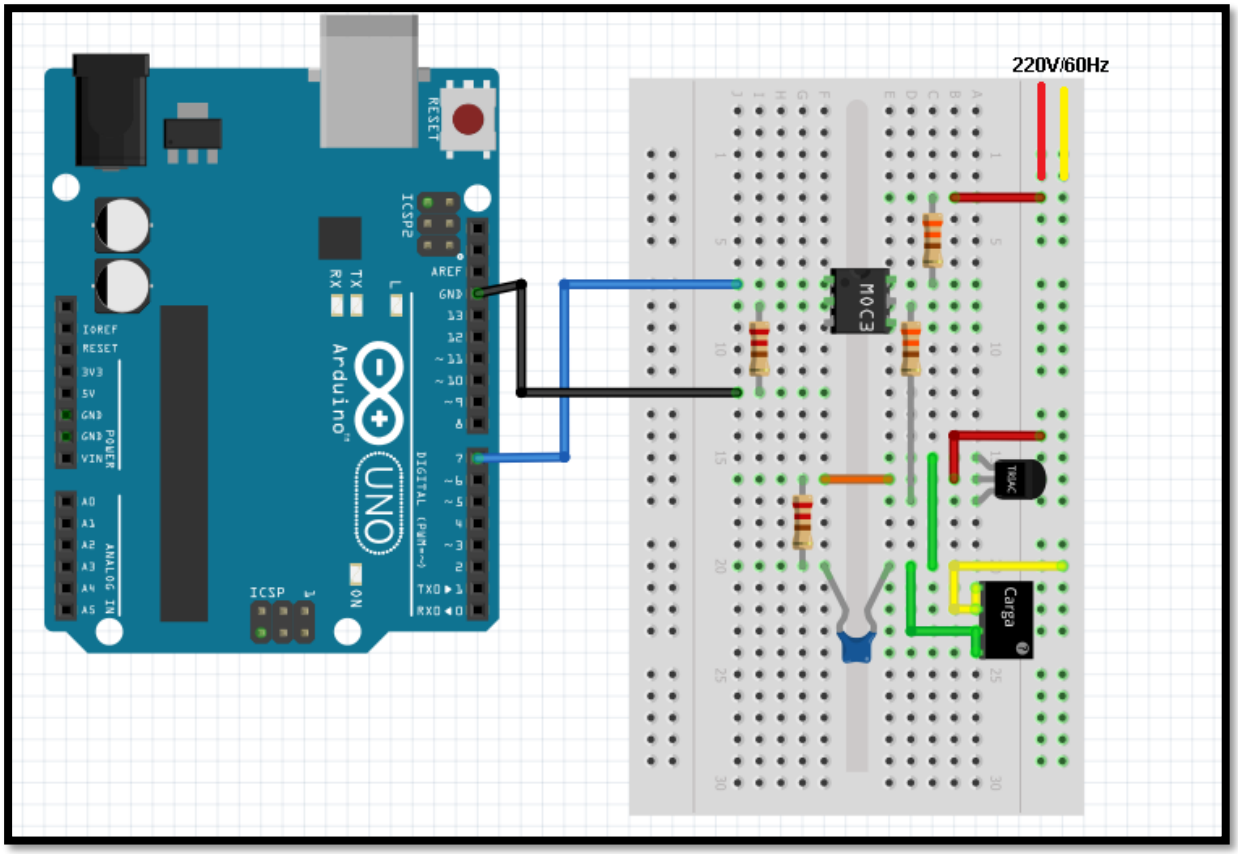


Figura 10 - Driver de potência do aquecedor AC.

3.2 - Modelo das funções de transferência.

3.2.1 - Aquecedor

Para obtenção do modelo da função de transferência do aquecedor, foi necessário o uso do Arduino junto a uma comunicação serial como software Processing 3, onde o Arduino iria ler os valores de temperatura a cada segundo e enviar via serial. O código no Arduino pode ser visto na Figura 11 abaixo.

```
1  int sensor = A0; //LM35
2  unsigned long tempoAnterior=0;
3  //variavel do ultimo tempo marcado, geralmente usada unsigned long
4  const long intervalo=1000;//valor constante, não pode ser mudado
5
6  void setup() {
7    pinMode(sensor, INPUT);
8    Serial.begin(9600);
9  }
10 float lertemp(int porta)//le 10 vezes a temperatura e faz a media
11 {
12   float valor=0;
13   int i=0;
14   for(i=0; i<10;i++)
15   {valor = valor + analogRead(porta);}
16   //Equação para passar de 8bits para a temperatura real
17   return valor*50/1023;}
18
19 void loop() {
20   unsigned long tempo = millis();
21   if (tempo - tempoAnterior >= intervalo)
22   {//se der o invervalo de 1s
23     tempoAnterior = tempo;//iguala os tempos
24     float valor = lertemp(sensor);
25     Serial.print(tempoAnterior/1000);
26     Serial.print(" ");
27     Serial.println(valor);
28   }}
```

Figura 11 - Código do Arduino para salvar a temperatura lida.

O código basicamente conta o tempo até dar o intervalo de um segundo, onde faz uma média de 10 valores lidos pela entrada analógica A0, onde está conectado o sensor e envia o tempo em segundos, junto ao valor para a porta serial, que será recebida pelo programa do Processing. Para o código que irá receber e salvar um arquivo de texto com os dados, desenvolveu-se o código da Figura 12 abaixo.

```
import processing.serial.*; //Biblioteca importante para essa aplicação
Serial mySerial; //Funcionalidade serial
PrintWriter output; //Funcionalidade de escrita em arquivos
void setup() {
 size(500,300);  // Abre uma janela no tamanho 500x300
 smooth(); // Habilita o anti-aliasing
 background(0); // Define fundo na cor preta
 textFont(createFont("Arial Bold",18)); //Carrega a fonte de texto a ser utilizada
 textAlign(CENTER);
  String portName = Serial.list()[0]; //configura a porta serial
  mySerial = new Serial(this, portName, 9600); //Configura a porta serial
  output = createWriter( "data.txt" ); //Cria o objeto arquivo para gravar os dados
}
void draw() { //Mesma coisa que a funçao loop do Arduino
  if (mySerial.available() > 0 ) { //Se receber um valor na porta serial
    String value = mySerial.readStringUntil('\n'); //Le o valor recebido
    if ( value != null ) { // Se o valor nao for nulo
      output.print(value); //value é o valor recebido pela porta serial
      fill(255);
      rect(111,126,273,28);
      fill(0);
      text("temperatura : " + value, width/2, height/2);
    }}}
void keyPressed() { //Se alguma tecla for pressionada
  output.flush(); // Termina de escrever os dados pro arquivo
  output.close(); // Fecha o arquivo
  exit(); // Para o programa
}
```

Figura 12- Código do Processing para salvar a temperatura lida pelo Arduino.

A função setup da Figura 7 cria a janela, inicia a criação do arquivo data.txt, onde os dados serão salvos e habilita a porta serial que irá se comunicar com o Arduino. Na função draw, essa análoga ao Loop do Arduino, ele verifica a conexão serial, lê os dados até o final da linha, imprime os valores no arquivo de texto e depois na janela criada. A função keyPressed verifica se alguma tecla foi pressionada enquanto a janela está no primeiro plano do computador e, caso for pressionado, ela salva o arquivo de texto e fecha a janela.

Com os dois códigos abertos, já enviado ao Arduino, conectou-se a ventoinha a alimentação 12V e ligou-se as resistências na rede 220V/60Hz, ao mesmo tempo que executou-se o código do Processing, como pode ser visto na Figura 13 a seguir.

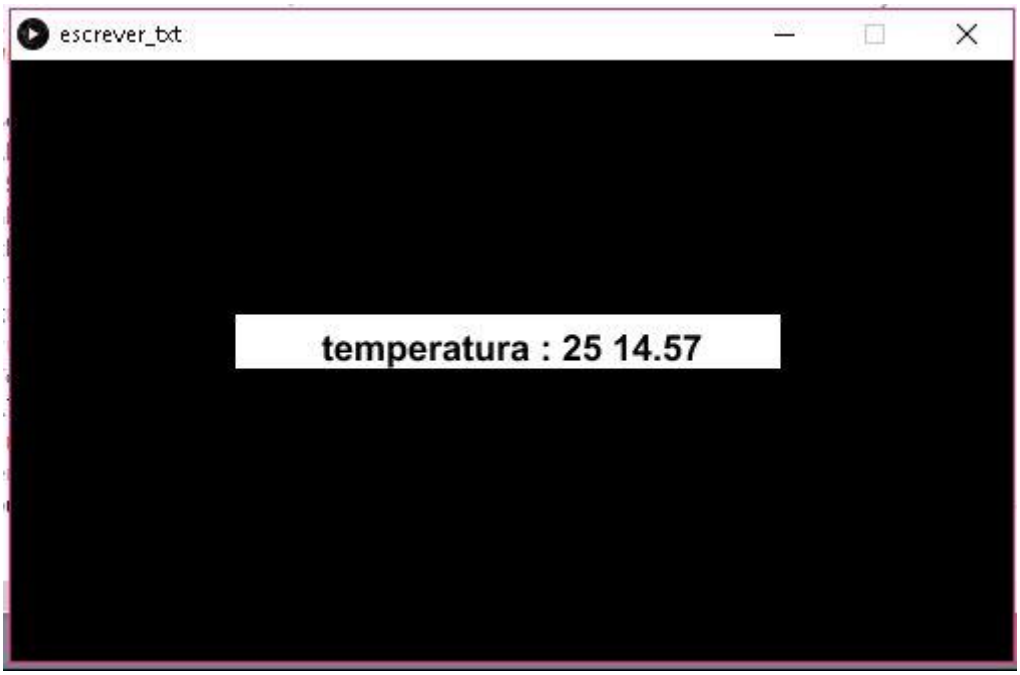


Figura 13 - Tela de monitoramento do Processing 3.

Após a temperatura estabilizar, fechou-se o código e o arquivo de texto foi movido para outra pasta, para não ser sobrescrito e iniciou-se o teste de modelo do resfriador.

3.2.2 – Resfriador

Para o resfriador, com a temperatura máxima atingida pelo aquecedor, desligou-se o sistema de aquecimento, ligou-se o resfriador e abriu-se novamente o código do Processing, agora visando salvar os dados de temperatura do resfriador, onde esperou-se até que a temperatura se estabiliza-se com a temperatura ambiente, para fechar o programa e salvar os dados.

3.2.3 - Luminosidade

Para o modelo do sistema de iluminação, foi necessário um outro código no Arduino, devido a resposta das fitas de Led serem bem mais rápidas que a do sistema de aquecimento e resfriamento da estufa, sendo os testes realizados a cada 20 micro segundos e depois de terminado, o Arduino envia os valores via serial, como pode ser visto na Figura 14.

```
int sensor = A0; //LDR
unsigned long tempoAnterior=0;
const long intervalo=20;//função constante, não pode ser mudada
int liga = 0,i = 0,valores[550];
void setup() {
  pinMode(sensor, INPUT);
  pinMode(2, INPUT);//PINO DO BOTAO
  pinMode(8, OUTPUT);
  Serial.begin(9600);}
void loop() {
  if(digitalRead(2)==1)
  {//verifica se apertou o botão
    liga = 1;
    digitalWrite(8,HIGH);}
  if(liga==1){
    unsigned long tempo = micros();
    if (tempo - tempoAnterior >= intervalo) {//se der o invervalo de 20us
      tempoAnterior = tempo;//iguala os tempos
      valores[i] = analogRead(sensor);//le o LDR
      i++;}}
  else{
    digitalWrite(8,LOW);
    Serial.print("");}
  if(i>=500){
    liga = 0;
    for (int j=0;j<=i;j++){
    Serial.print((20*j));
    Serial.print(" ");
    Serial.println(valores[j]);}}}
```

Figura 14 - Código do Arduino para salvar a temperatura lida.

3.3 – Bancada

Com o corpo da bancada adquirido entre as sucatas da faculdade, removeu-se as partes que não iriam ser utilizadas, resultando no corpo da Figura 15.

Figura 15 - Bancada utilizada para montagem.

Tendo a bancada limpada, colou-se um tubo para evitar vazamento da terra no furo pré-existente e em seguida, foi preenchido cerca de 2 centímetros de altura de terra, como pode ser visto na Figura 16.

Figura 16 - Preenchimento do fundo com terra.

Para montagem do sistema de aquecimento, foi passado pasta térmica nas resistências de chapinha de cabelo e fixou-se as resistências com uma chapa metálica e 2 parafusos no dissipador, como observado na Figura 17.

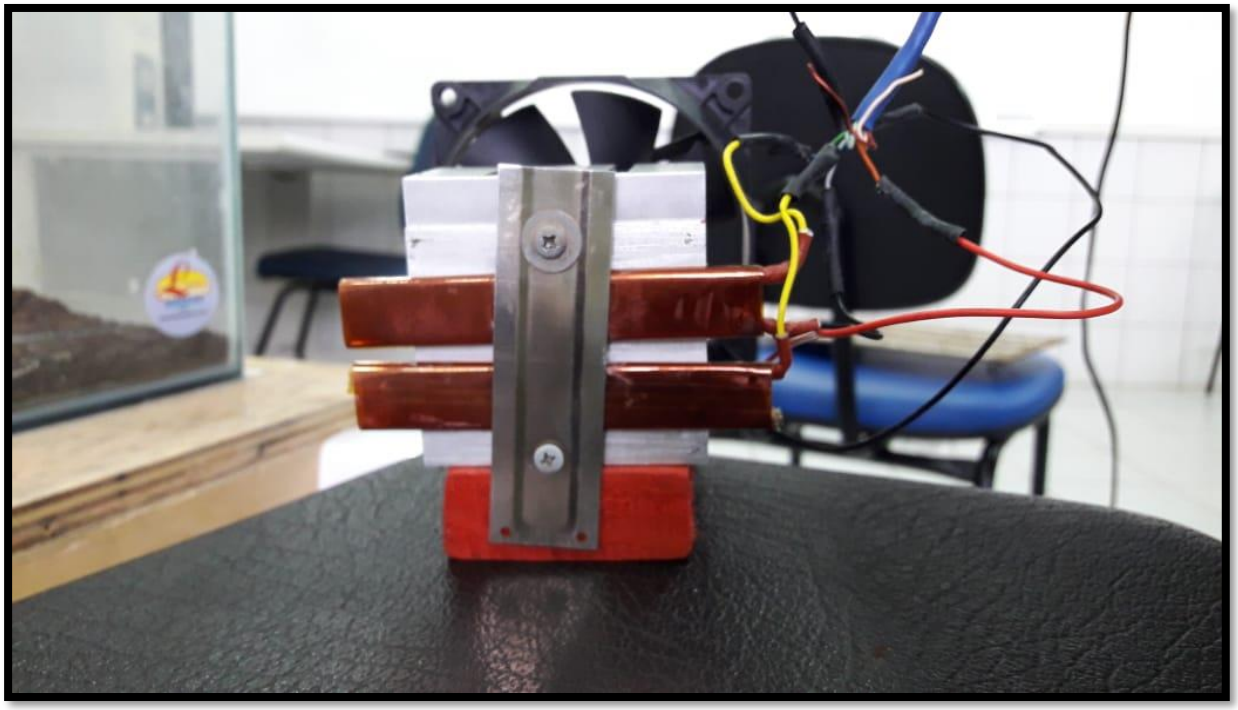


Figura 17 - Fixação das resistências no dissipador do sistema de aquecimento.

Assim, colou-se a ventoinha e o dissipador em um suporte de madeira e fixou-se o sistema de aquecimento no canto da estufa, representado na Figura 18 abaixo.

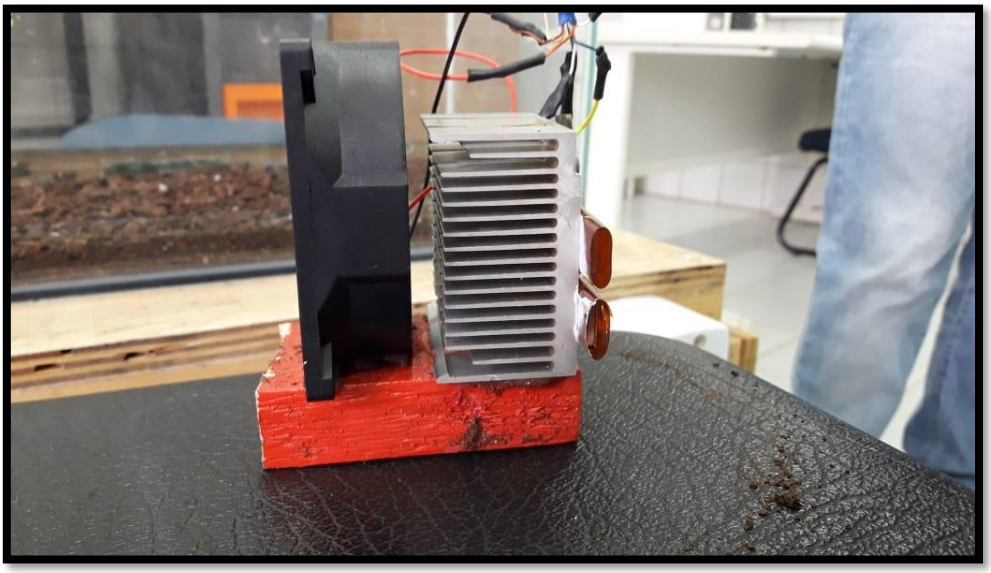


Figura 18 - Sistema de aquecimento montado.

Para o sistema de resfriamento, com uma tampa de Duratex, fez-se um furo circular de raio 80 mm, para encaixe da ventoinha de resfriação, mostrado na Figura 19.

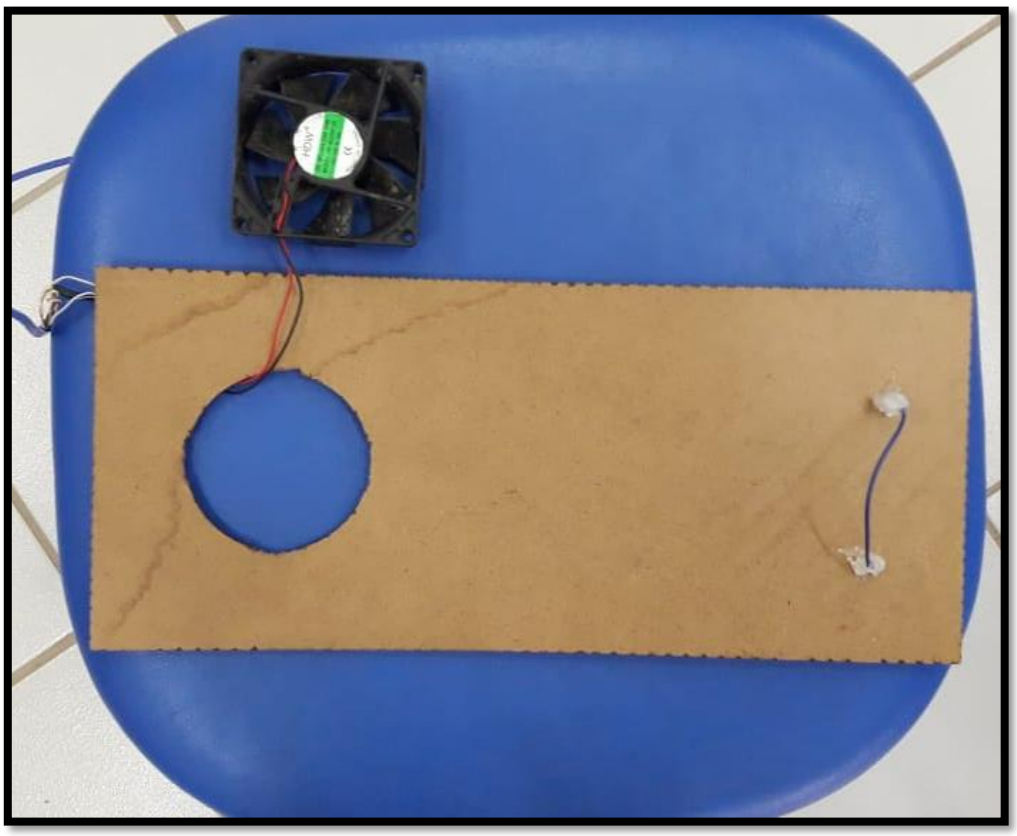


Figura 19 - Tampa com furo para ventoinha.

Com a ventoinha encaixada, colou-se as fitas de LED na parte de baixo da tampa, como pode ser visto na Figura 20.

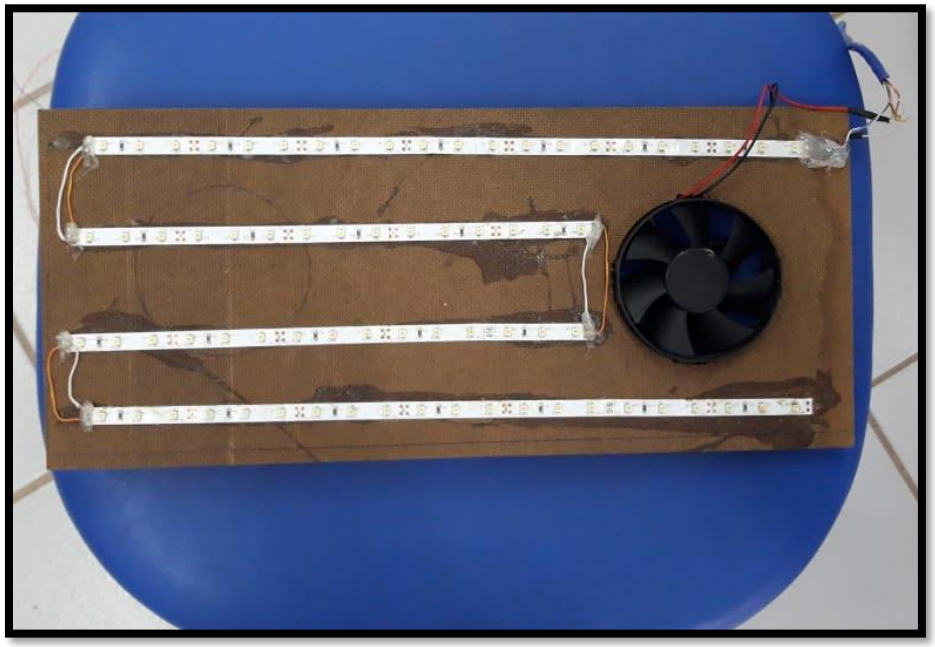


Figura 20 - Tampa montada com a ventoinha e as fitas de LED.

Assim, com os sistemas prontos, utilizou-se o tubo fixado anteriormente para facilitar a manipulação dos fios e alocou-se os sensores em suas respectivas posições, mostrado na Figura 21.

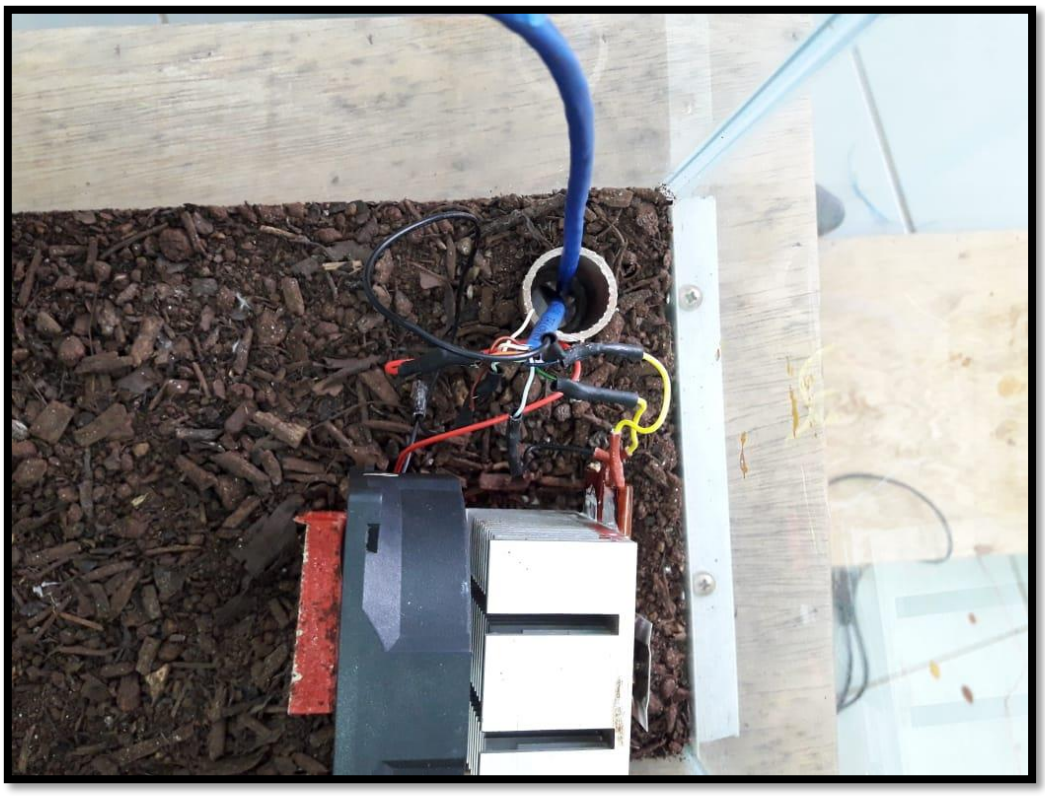


Figura 21 - Tubo utilizado para passagem do cabeamento.

Por último, para representar o plantio, utilizou-se de pequenas plantas artificiais, como gramas e ramos de pimenta, como pode ser notado na Figura 22.

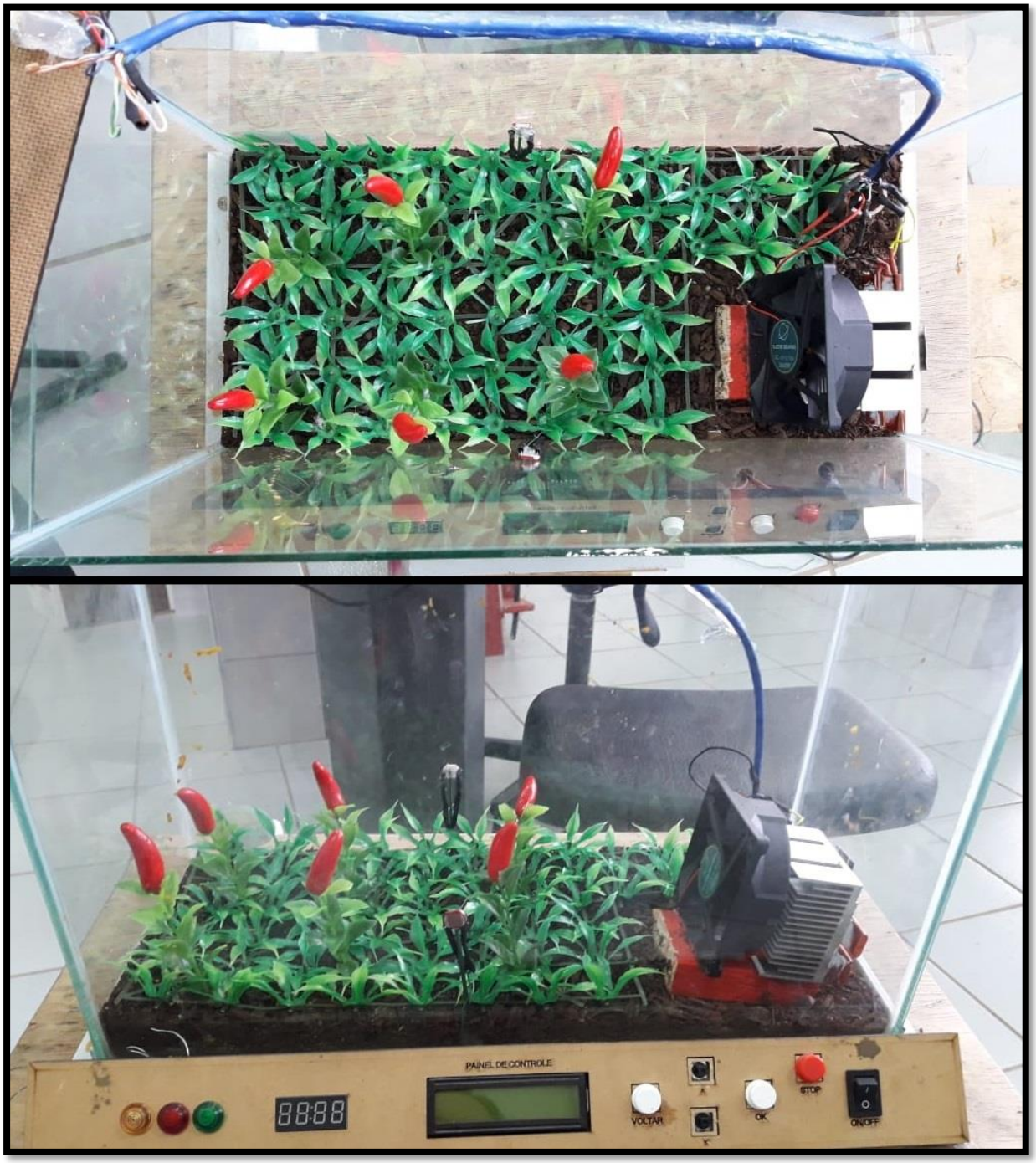


Figura 22 - Bancada montada e pronta para utilização.

4- RESULTADOS E DISCUSSÕES

4.1- Resultado dos modelos

4.1.1 – Luminosidade

Após o procedimento para obtenção dos dados, com a ajuda do Matlab importou-se os arquivos e tratou-se os dados para eliminar leituras extras e repetições do ciclo. Assim, com o resultado dos dois testes importados plotou-se a curva de resposta do sistema, como pode ser visto na Figura 23.

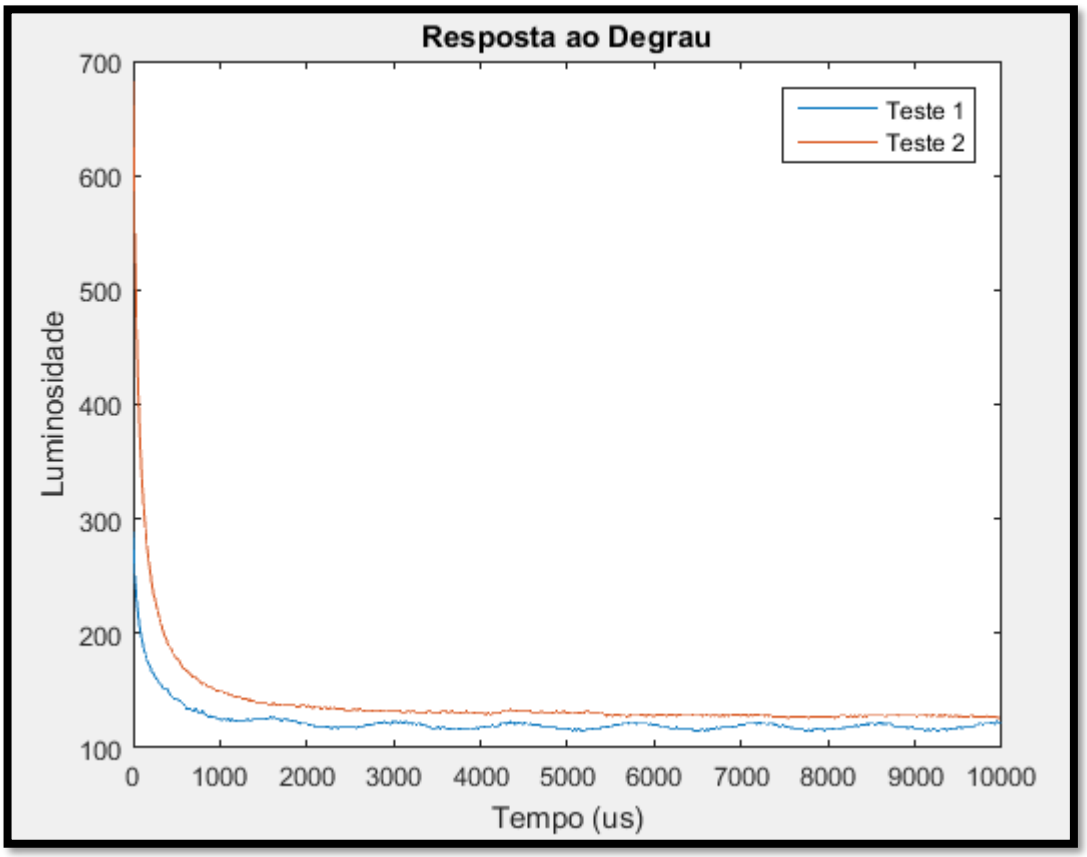


Figura 23- Resposta ao degrau dos testes de luminosidade.

Como pode ser visto, a resposta para os dois testes foi extremamente rápida, com um tempo de estabilização menor que dois mili segundos. A resposta decai, pois o LDR diminui sua resistência com o aumento da luminosidade, fazendo com que o valor que chegue pelo divisor de tensão, diminua.

Depois disso, pelo comendo Ident, importou-se os dois dados para a janela de identificação de modelo e optou-se por uma função de transferência com um polo e delay, resultando em um modelo com 82,62% de aproximação, que corresponde a uma boa função de transferência, como pode ser visto na Figura 24.

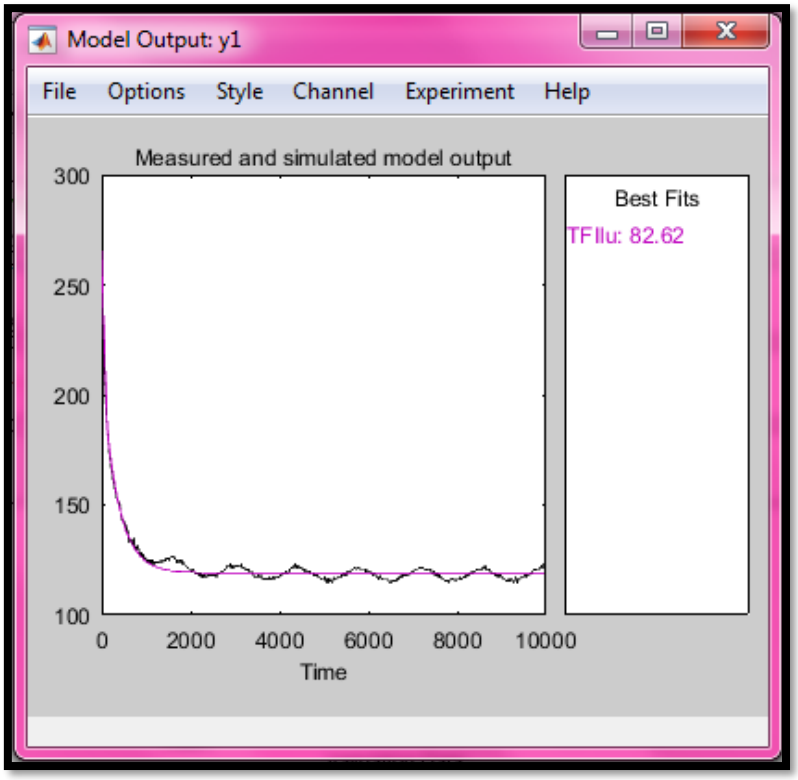


Figura 24- Validação da função de transferência da luminosidade com primeiro teste.

Para validação com segundo teste, obteve-se uma aproximação de 62,77%, como pode ser visto na Figura 25.

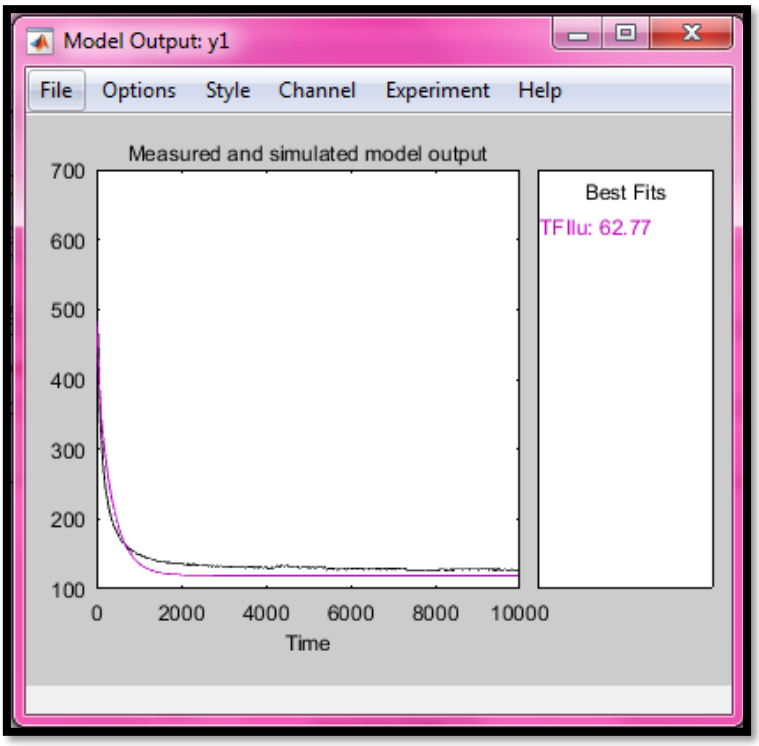


Figura 25- Validação da função de transferência da luminosidade com segundo teste.

Os valores obtidos são aceitáveis, podendo passar essa função de transferência para o Workspace do Matlab, obtendo então, sua função de transferência, como pode ser visto na Figura 26.

```
GIlu =

  From input "u1" to output "y1":
                    119.1
  exp(-129*s) * -----------
                348.4 s + 1
```

Figura 26- Função de transferência da luminosidade.

Como, nesse caso, a resposta é muito rápida, não é preciso um controle PID, optando-se por deixar um ganho proporcional (kp) igual a um e os ganhos integral (ki) e derivativo (kd) iguais a zero, ou seja, a resposta ao degrau do sistema de luminosidade é vista na Figura 27.

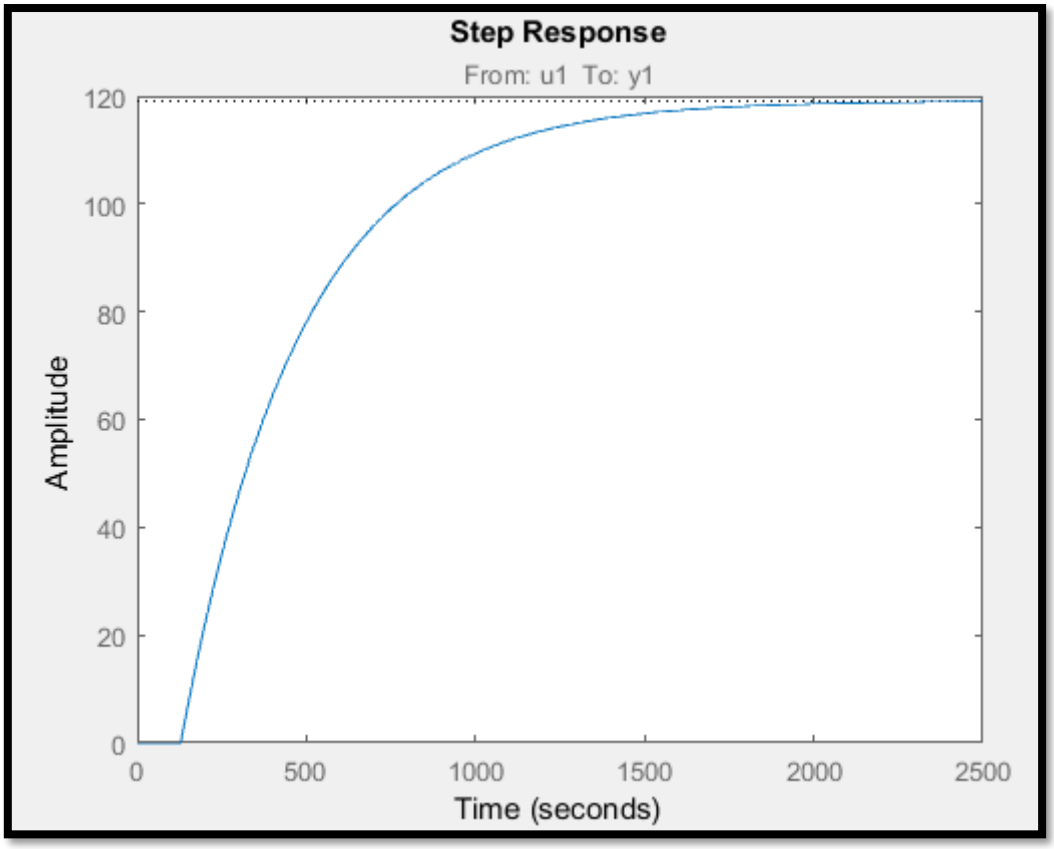


Figura 27- Resposta ao degrau da função de transferência do sistema de iluminação.

4.1.2- Aquecimento

Após o procedimento para obtenção dos dados, com a ajuda do Matlab importou-se os arquivos e tratou-se os dados obtidos no comando Ident. Assim, com o resultado dos dois testes importados plotou-se a curva de resposta do sistema, como pode ser visto na Figura 28.

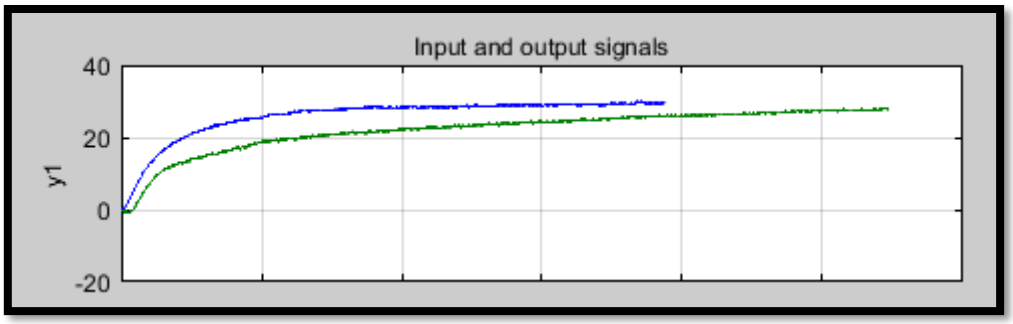


Figura 28- Resposta ao degrau do teste de aquecimento.

Depois disso, ainda pelo comando Ident, importou-se os dois dados de aquecimento para a janela de identificação de modelo e optou-se por uma função de transferência com dois polos e um zero, resultando em um modelo com 94,05% de aproximação, que corresponde a uma ótima função de transferência, como pode ser visto na Figura 29.

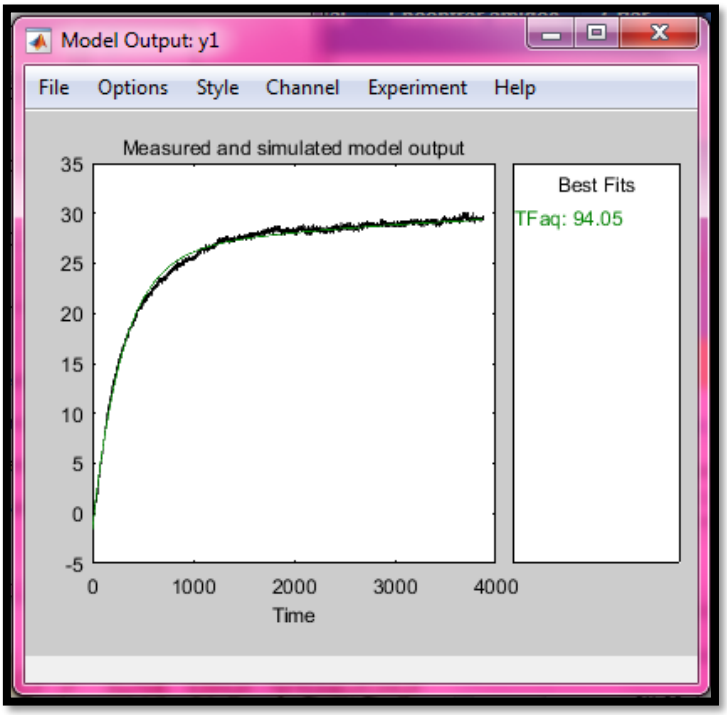


Figura 29- Validação da função de transferência do aquecimento com primeiro teste.

Já para o segundo teste, obteve-se uma aproximação de 93,72%, como pode ser visto na Figura 30.

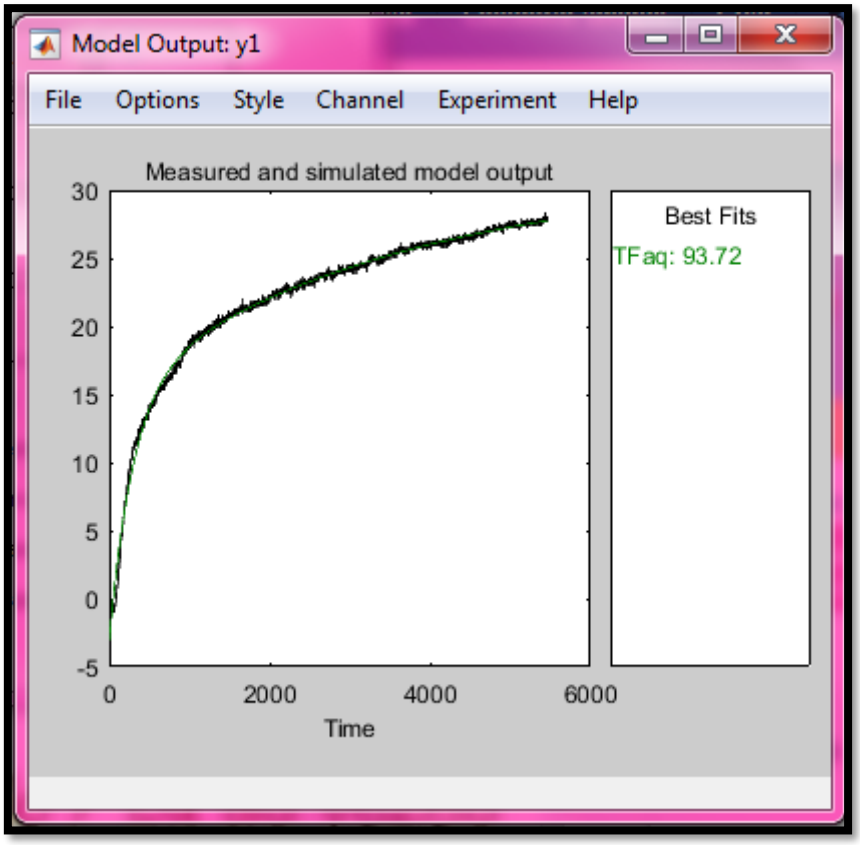


Figura 30- Validação da função de transferência do aquecimento com segundo teste.

Como os valores obtidos são aceitáveis, pôde-se passar essa função de transferência para o Workspace do Matlab, obtendo então, sua função de transferência, como pode ser visto na Figura 31.

```
Gaq =

      5.957e04 s + 31.2
  ------------------------
  1.055e06 s^2 + 3849 s + 1
```

Figura 31- Função de transferência do aquecedor.

Após obter-se a função de transferência, pode-se ver sua resposta ao degrau na Figura 32. Como a resposta corresponde aos dois testes com excelentes aproximações e como ela parece uma resposta de primeira ordem, devido a uma aproximação do Ident, obteve-se um polo e um zero muito próximos.

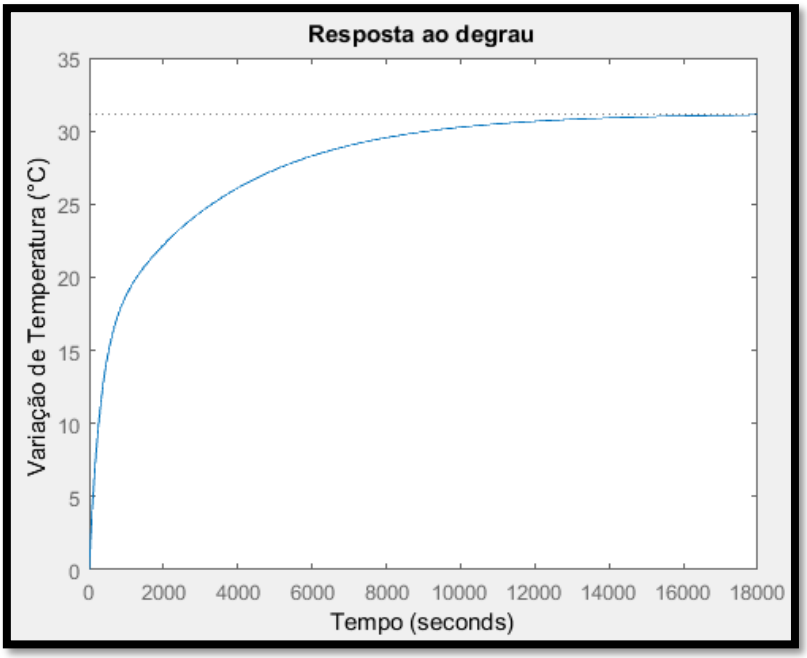


Figura 32- Resposta ao degrau da função de transferência do sistema de aquecimento.

4.1.3- Resfriador

Após a obtenção dos dados, com a ajuda do Matlab importou-se os arquivos e tratou-se os dados. Assim, com o resultado dos dois testes importados plotou-se a curva de resposta do sistema, como pode ser visto na Figura 33.

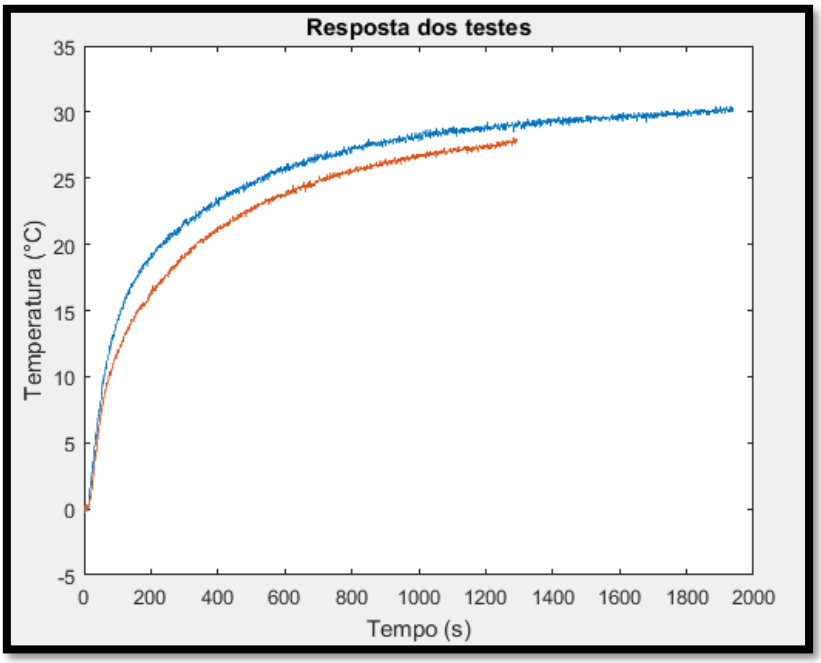


Figura 33- Resposta ao degrau do teste de resfriamento.

Após obtida a resposta ao degrau dos testes, no comando Ident, importou-se os dois dados do teste de resfriamento para a janela de identificação de modelo e optou-se por uma função de transferência com dois polos e um zero, resultando em

um modelo com 96,27% de aproximação, que corresponde a uma excelente função de transferência, como pode ser visto na Figura 34.

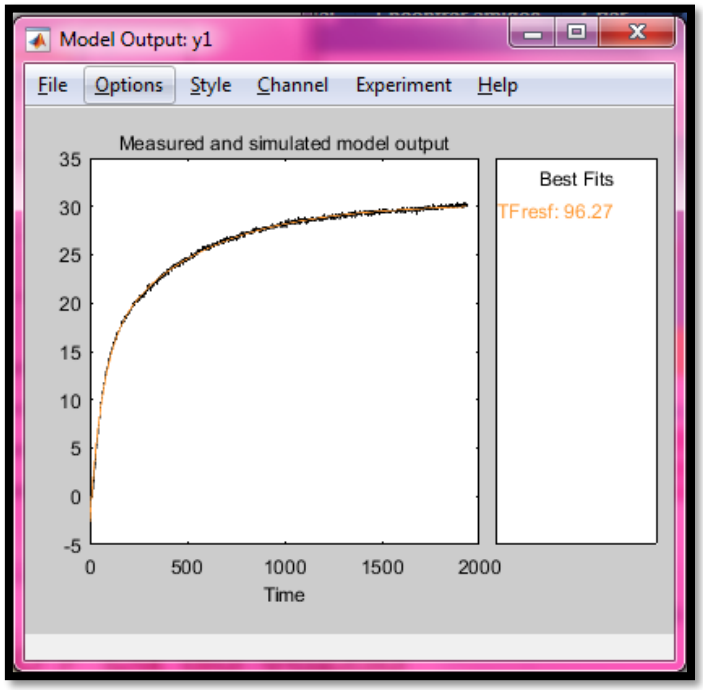


Figura 34- Validação da função de transferência do resfriamento com primeiro teste.

Para o segundo teste obteve-se uma aproximação de 89,57%, como pode ser visto na Figura 35.

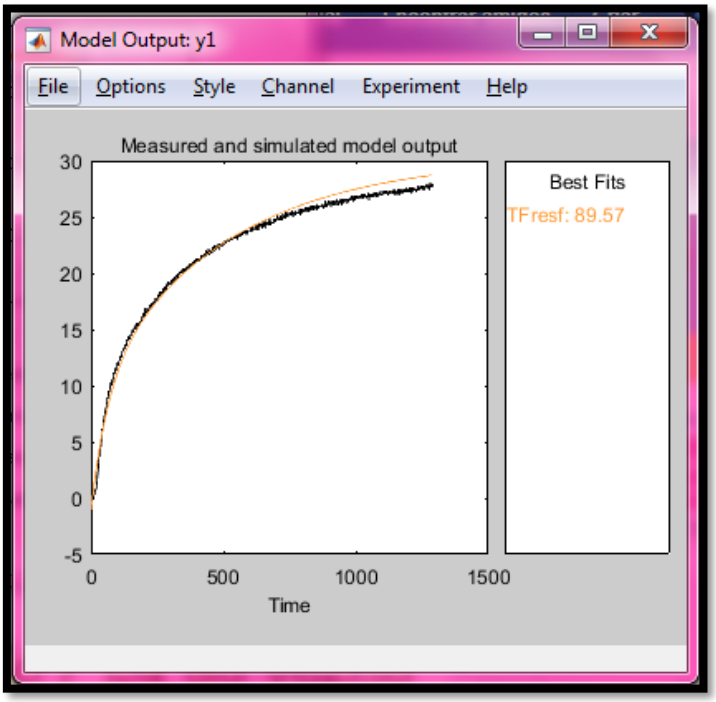


Figura 35- Validação da função de transferência do resfriamento com segundo teste.

Como os valores obtidos são aceitáveis, pôde-se passar essa função de transferência para o Workspace do Matlab, obtendo então, sua função de transferência, como pode ser visto na Figura 36.

```
Gres =

        8899 s + 30.3
  --------------------------
  3.223e04 s^2 + 567.3 s + 1
```

Figura 36- Função de transferência para o sistema de resfriamento.

Após obter-se a função de transferência, pode-se ver sua resposta ao degrau na Figura 37.

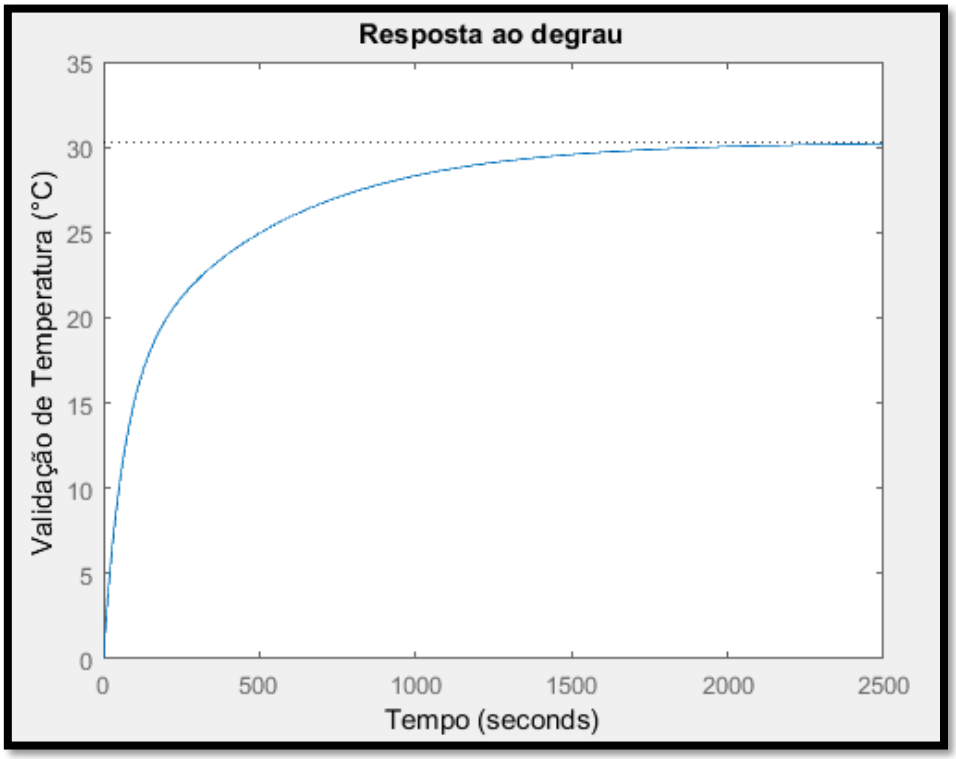


Figura 37- resposta o degrau do sistema de resfriamento.

4.1.4 – Estratégia de Controle

Tanto para a função de transferência de aquecimento quanto para a de resfriamento, foram obtidos 2 polos e 1 zero, mas mesmo assim, a resposta ao degrau se assemelha a uma função com apenas um polo e nenhum zero. Isso se deve pelo fato de as duas funções possuírem um polo muito próximo do zero, fazendo com que o polo que sobrou tenha predominância na resposta.

Assim, pelo fato do sistema de iluminação ser único, utilizou-se de um controle com realimentação comum e, como o esquema de aquecimento e resfriamento consistia em duas ventoinhas, uma fornecendo ar quente e a outra retirando ar quente do sistema, a melhor estratégia seria a estratégia Split-Range com uma histerese de 2ºC, como pode ser visto na Figura 38 abaixo.

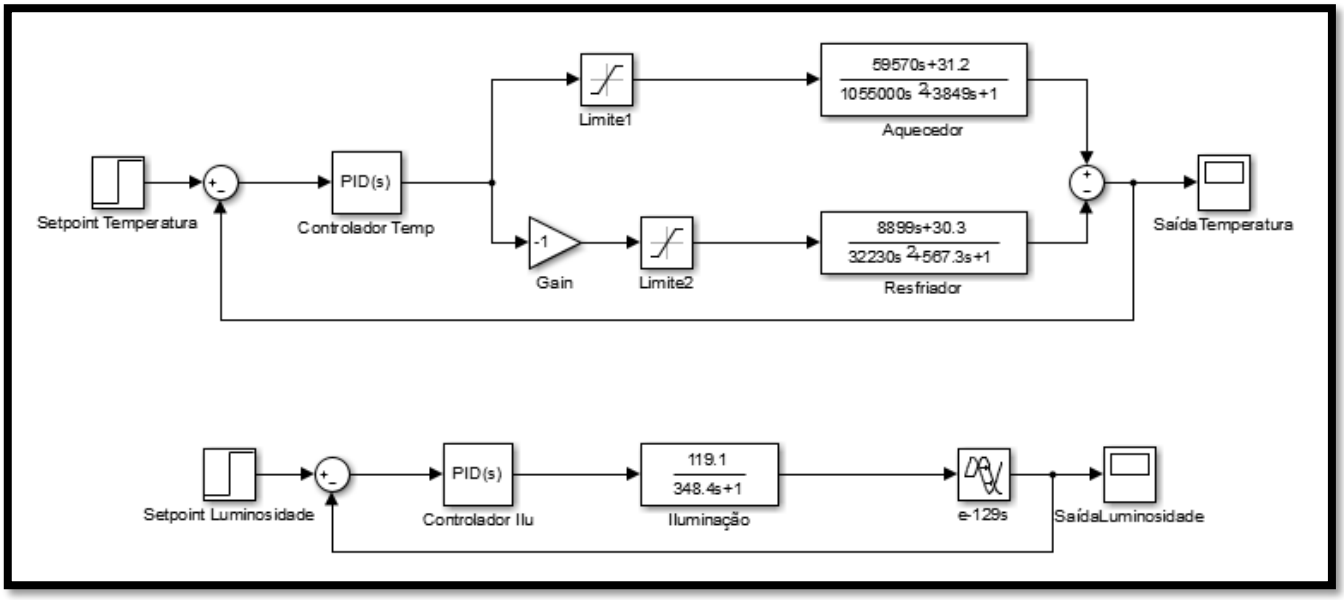


Figura 38 - Controle Split-Range de temperatura e PID simples de Iluminação no Simulink.

Pelo bloco do PID de temperatura, com auxilio do PID tunning do Simulink, obteve-se então os ganhos de:

$$K_P = 0.0556581594865022$$
$$K_D = -0.693052286053374$$
$$K_i = 0.000778346310258697$$

Com os ganhos ajustados realizou-se vários testes, como por exemplo a resposta ao degrau de 30ºC, que pode ser vista na Figura 39.

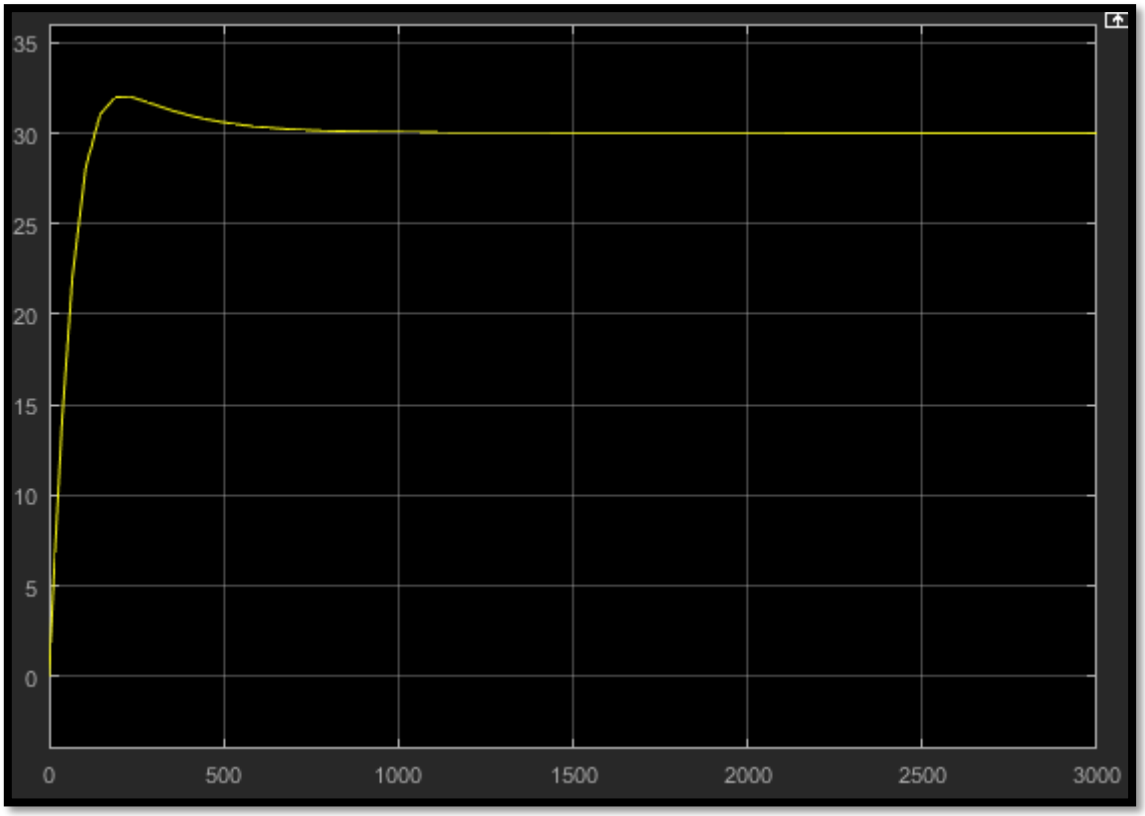


Figura 39 - Resposta setpoint de 30 ºC.

Como observa-se na Figura 39, a resposta foi muito boa, sendo mais rápida que os testes realizados, mesmo com um pequeno overshoot, mas nada que pudesse comprometer o sistema, não sendo necessário nenhum ajuste fino dos parâmetros do PID.

4.2- Comparação entre cálculos e modelos

Para o sistema de aquecimento, partindo da equação geral dos gases, representada pela Equação 1 abaixo, tem-se:

$$P_{atm}.V = n.R.T \quad (1)$$

Onde P é a pressão em Pa, V o volume m³, n o número de mols, R a constante universal dos gases em Pa.m³/mol.K e T a temperatura em Kelvin. Sabe-se que o número de mols é dado pela relação entre a massa total e a massa molar, resultando na Equação 2:

$$P_{atm}.V = \frac{m}{M}.R.T \quad (2)$$

Sabe-se que a relação entre a constante universal dos gases e a massa molar do gás é a constante particular do gás, que para o ar é dada (3):

$$\overline{R}_{ar} = 287\,\frac{J}{kg.K} \quad (3)$$

Assim, substituindo (3) em (2), tem-se a Equação 4:

$$\frac{m}{V} = \frac{P_{atm}}{\overline{R}_{ar}.T_{media}} \quad (4)$$

A relação entre massa e volume é a massa específica do ar, como mostra a Equação (5):

$$\rho = \frac{m}{V} \quad (5)$$

De (5) em (4), tem-se a Equação 6:

$$\rho = \frac{P_{atm}}{\overline{R}_{ar}.T_{media}} \quad (6)$$

Pela calorimetria, sabe-se que a quantidade de calor no aquecimento de uma substancia é dada pela Equação 7:

$$Q = m.c_p.\Delta T \quad (7)$$

Onde Q é a quantidade de calor em J, m é a massa em kg, c_p é o calor específico em J/Kg.K e ΔT a variação de temperatura em ºC ou K. Por definição a potência é a variação de energia em função do tempo, ou seja, a derivada da energia em relação ao tempo, resultando na Equação 8:

$$P = \frac{dQ}{dt} = \frac{d}{dt}\left[m.c_p.\Delta T\right] = \frac{dm}{dt}.c_p.\Delta T = \dot{m}.c_p.\Delta T \quad (8)$$

Onde ṁ é a vazão em massa de ar que passa pela ventoinha em kg/s.Como a vazão em massa se relaciona com a vazão volumétrica pela Equação 9 e a vazão volumétrica é dada pela Equação 10 abaixo, resulta-se na Equação 11 a seguir:

$$\dot{m} = V_{vol}.\rho \quad (9)$$

$$V_{vol} = V.A \quad (10)$$

$$\dot{m} = V.A.\rho \quad (11)$$

Sendo V a velocidade do ar em m/s, A a área da seção transversal da ventoinha em m² e ρ a massa específica do ar em kg/m³, de (11) em (8), tem-se a Equação 12:

$$P = \rho.V.A.c_p.\Delta T \quad (12)$$

De (6) em (12), obtém-se finalmente a potência, dada pela Equação 13:

$$P = \frac{P_{atm}}{\overline{R}_{ar}.T_{media}}.V.A.c_p.\Delta T \quad (13)$$

Assim, pelo Anemômetro de fio quente, obteve-se uma velocidade média para a ventoinha de:

$$V = 0{,}2\frac{m}{s} \quad (14)$$

Pela embalagem da resistência de chapinha de cabelo, tem-se que a potência da chapinha é:

$$P = 40W \quad (15)$$

Medindo a seção transversal da ventoinha, obteve-se o diâmetro D=80,0 ± 0,5 mm, resultando na área:

$$A = \pi(\frac{d}{2})^2 = \pi(\frac{0{,}080 \pm 0{,}005}{2})^2 = 0{,}00503 m^2$$

O erro para a área pode ser obtido pelo método RSS, ou seja:

$$\Delta A = A.\sqrt{2.(\frac{\Delta D}{D})^2} = 0{,}00503\sqrt{2.(\frac{0{,}5}{80})^2} = 0{,}00004\ m^2$$

Ou seja, a área, com o erro propagado, será dada por:

$$A = 0{,}00503 \pm 0{,}00004\ m^2 \quad (16)$$

A diferença de temperatura medida pelo LM35 nos testes de obtenção do modelo da função de transferência pode ser vista na Tabela 1 abaixo:

Tabela 1 - Variações de temperatura nos dois testes obtidos.

Teste	Temperatura (ºC)
Teste 1	29,5 ± 0,5
Teste 2	29,5 ± 0,5

O erro considerado foi o erro adotado pelo datasheet do fabricante, de 0,5ºC, como pode ser visto no Anexo III. Baseado em [14], a pressão atmosférica média em São João da Boa Vista é de:

$$P_{atm} = 898\ hPa = 89800\ Pa \quad (17)$$

Por [15], a temperatura média ao longo do ano de São João da Boa Vista é dada por:

$$T_{media} = 22ºC = 295{,}15\ K \quad (18)$$

Baseado em [16], o calor específico do ar é dado por:

$$c_p = 1007\frac{J}{kg.K} \quad (19)$$

Assim, de (3), (14), (16), (17), (18) e (19) em (13), tem-se a relação entre potência e variação de temperatura dada pela Equação 20:

$$P = \frac{89800}{287.295{,}15}.0{,}2.(0{,}00503 \pm 0{,}00004).1007.\Delta T$$

$$P = (1{,}073 \pm 0{,}005).\Delta T \quad (20)$$

Com a Equação 20, junto a (15) e aos dados da Tabela 1, montou-se a Tabela 2, com os valores de temperatura medidos dos testes e o calculado pela potência da chapinha, junto com a potência calculada pelos testes.

Tabela 2 - Variação de temperatura e potência do sistema de aquecimento.

	Temperatura (ºC)			Potência (W)		
Teste 1	29,5	±	0,5	31,7	±	0,6
Teste 2	28,0	±	0,5	30,0	±	0,6
Nominal	37,3	±	0,2	40		

Com os erros obtidos pelo método RSS, nota-se que o a diferença de potência utilizada em relação a potência total das chapinhas, houve uma perda média de 9,2±0,6 W, devido principalmente a perda de calor pelo vidro ao ambiente externo, junto a estufa não ser totalmente vedada, perdendo um pouco do ar aquecido pelos vãos da tampa e da ventoinha de resfriação. Realizando os mesmos cálculos para o sistema de resfriação, montou-se a Tabela 3 abaixo.

Tabela 3 - Variação de temperatura e potência do sistema de resfriamento.

	Temperatura (ºC)			Potência (W)		
Teste 1	-30,0	±	0,5	-32,2	±	-0,6
Teste 2	-28,0	±	0,5	-30,0	±	-0,6
Média	-29,0	±	0,5	-31,1	±	-0,6

É importante salientar que os valores negativos da Tabela 3 se devem pelo fato de que o resfriador remove ar quente da estufa, ou seja, ele retira energia de dentro do sistema, fazendo a temperatura final ser menor que a inicial. Os valores de potência para os 2 sistemas foram próximos, uma vez que a variação de temperatura foi aproximadamente as mesmas.

Para obter os valores de resistência no dissipador, com a tensão da rede sendo $220V_{RMS}$/60Hz, por (15), espera-se que o valor de resistência seja:

$$P = \frac{U^2}{R} \rightarrow R = \frac{U^2}{P} = \frac{220^2}{40} = 1210\Omega$$

Medindo os valores das resistências, foi obtido os valores de 560 ± 1 Ω e 600 ± 1 Ω, que somadas, chegam a 1160 ± 2 Ω, um desvio de 4% do esperado, que está dentro de uma faixa aceitável.

4.3 – Circuitos

Para os drivers de corrente continua, ou seja, que irão operar os sistemas de resfriamento e de iluminação, pelo datasheet do CI 4N25 (Anexo II), obtém-se a Tabela 4.

Tabela 4 - Valores máximos entrada e saída do 4N25.

4N25	
Input	**Output**
I_F = 60 mA	I_{Cmax} = 50 mA
V_F = 1,3 V	P_{max} = 150 mW
P_{max} = 100 mW	

Pela malha de entrada, pela lei de Kirchoff das tensões, tem-se a Equação 21:

$$V_{cc} = R.I + V_F$$
$$5 = R.I + 1{,}3$$
$$R.I = 3{,}7$$
$$R = \frac{3{,}7}{I} \quad (21)$$

Assim, o calor da resistência na saída do Arduino pode ser definido pela corrente de ativação do opto acoplador. Baseado na Tabela 4, a corrente máxima direta na entrada do opto acoplador é de 60mA, escolhendo uma corrente de operação como 20 mA, pela Equação 21, o valor da resistência deve estar próximo a:

$$R = \frac{3{,}7}{0{,}020} = 185\Omega$$

O valor mais próximo disponível era de 220 Ω, sendo esse então utilizado para a saída PWM do Arduino. Como agora tem-se uma corrente que pode variar de 17 a 22 mA e uma tensão de polarização direta de 1,3V, com o caso mais extremo sendo 22mA, calcula-se a potência dissipada pela entrada como:

$$P = V_F.I_F = 1{,}3.0{,}022 = 28{,}6\,mW$$

Ou seja, o valor de potência dissipada pela LED de entrada não excede a potência máxima. Para a saída, com a alimentação de 12V e uma corrente que deve ser menor que 50mA, adotou-se o valor de 12mA, resultando então em uma resistência de:

$$V = R.I \rightarrow R = \frac{V}{I} = \frac{12}{0{,}012} = 1000\,\Omega$$

Com os valores de corrente (Ic) e tensão entre emissor e coletor (V_{CE}=0,5V), a potência dissipada pela saída do opto acoplador pode ser calculada em:

$$P = V.I = 0{,}5.0{,}012 = 6\,mW$$

O valor calculado está bem abaixo do limite, o que satisfaz as necessidades do projeto. Para o transistor utilizado, o modelo TIP31 possui os valores de corrente e tensão mostrados na Tabela 5 [Anexo I].

Tabela 5 - Dados do transistor TIP31.

TIP31	
I_{Bmax}	1A
I_{Cmax}	3A
V_{CEO}	40V
P_{max}	40W
$V_{CE(sat)}$	1,2V

Como já calculado, a corrente de base (I_B) no transistor é a mesma que a corrente de saída no 4N25, ou seja, 12mA, que atende as especificações. Por estimativa das cargas, sendo o sistema de iluminação aquele que irá consumir maior corrente, estimada em 300mA, enquanto a ventoinha tem um consumo dado de 150 mA, a corrente de coletor (Ic) não excederá os limites estabelecidos pelo fabricante. Com a tensão de saturação ($V_{CE(sat)}$) entre o coletor e o emissor do transistor, junto a corrente de coletor, tem-se que a potência dissipada pelo transistor é:

$$P = V.I = 1{,}2 * 0{,}3 = 360\ mW$$

Assim, a potência dissipada está dentro dos parâmetros máximos estabelecidos pelo fabricante. Para o circuito de controle de potência AC, pelo datasheet do MOC3021, pode-se compilar os dados necessários para dimensionamento do circuito na Tabela 6 [Anexo IV].

Tabela 6 - Valores máximos de corrente, tensão e potência para o MOC3021.

MOC3021	
Input	**Output**
I_F= 60 mA	V_{DRM} = 400V
V_F = 1,3 V	I_{TSM} = 1A
P_{max} = 120 mW	P_{max} = 150W

Nota-se que para a entrada do opto acoplador, o resistor pode ser de mesma magnitude do 4N25, uma vez que os valores de corrente e tensão direta são iguais e a potência máxima é maior, portanto, será usado R = 220 Ω. Seguindo o datasheet do componente, para os resistores de saída, recomenda-se utilizar 1 resistor de 330 Ω pela alimentação e outro de igual valor entre a o sinal de saída e o gate do triac.

Para o triac que será ativado pelo opto acoplador, optou-se pelo BT137, com valores máximos de corrente exibidos na Tabela 7 [Anexo V].

Tabela 7 - Valores máximos de corrente e tensão do triac BT137.

BT137	
V_{DRM}	600 V
I_{RMS}	8 A
I_{GM}	2 A
V_{GM}	5 V
P_{GM}	5 W

Como a tensão máxima de isolação (V_{DRM}) não ultrapassa o valor de pico da rede e, como obtido em (15), a potência da chapinha é de 40W, o que traz uma corrente de:

$$I_{RMS} = \frac{P}{V} = \frac{40}{220} = 182\ mA$$

Assim, a corrente não ultrapassar os valores máximos. Finalmente, para a corrente de ativação do gate (I_{GM}) é a mesma que na saída do opto acoplador, de aproximadamente 450mA, caso a corrente toda passe pelo gate, ainda não irá ultrapassar o valor máximo. Com o valor de tensão de disparo (V_{GT}=1,5V), junto a corrente de coletor, obtém-se a potência dissipada pelo gate do triac.

$$P_{GT} = V_{GT}.I_{GT} = 1{,}5.0{,}45 = 675\ mW$$

Completando então o dimensionamento dos circuitos, a potência dissipada pelo gate não ultrapassará a potência máxima. Para o circuito de driver PWM dimensionado e montado, pode-se medir os respectivos valores desejados, que resultou na Tabela 8 a seguir.

Tabela 8 - Valores calculados, medidos e os respectivos desvios para os drivers DC.

4N25				TIP31			
Valor	**Calculado**	**Medido**	**Desvio**	**Valor**	**Calculado**	**Medido**	**Desvio**
V_F (V)	1,3	1,262 ± 0,001	3%	**I_B (mA)**	12	7,88 ± 0,01	34%
I_F (mA)	16,818	15,82 ± 0,01	6%	**I_C (mA)**	300	230 ± 10	23%
P_{input} (mW)	21,864	19,96 ± 0,02	9%	**$V_{CE(sat)}$ (V)**	1,2	0,0546 ± 0,0001	95%
I_C (mA)	12	7,88 ± 0,01	34%	**P (mW)**	360	12,6 ± 0,5	97
$V_{CE(sat)}$ (V)	0,5	0,5 ± 0,1	0%				
P_{output} (mW)	6	3,9 ± 0,8	34%				

Pelos dados da Tabela 8, pode-se observar que houveram alguns desvios em relação aos valores calculados, mas nada que atrapalhe no funcionamento do circuito ou que comprometa a integridade dos dispositivos, uma vez que pela característica não ideal e pelos dados do datasheet dos componentes serem obtidos em condições extremamente especificas, traz uma variação nos valores medidos, assim como ocorre a propagação da incerteza pelos cálculos. Finalmente com o

circuito do driver AC, pode-se montar a Tabela 9, com os dados para os dois principais componentes.

Tabela 9 - Valores calculados, medidos e os respectivos desvios para o driver AC.

MOC3021				BT137			
Valor	**Calculado**	**Medido**	**Desvio**	**Valor**	**Calculado**	**Medido**	**Desvio**
V_F (V)	1,3	1,212 ± 0,001	7%	**I_{RMS} (mA)**	182	189,7 ± 0,1	4,21%
I_F (mA)	16,818	16,18 ± 0,01	4%	**I_{GT} (mA)**	450	430 ± 10	4,44%
P_{input} (mW)	21,86	19,61 ± 0,02	10%	**V_{GT} (V)**	1,5	0,35 ± 0,01	76,67%
I (mA)	450	430 ± 10	4%	**P_{GT} (W)**	675	151 ± 6	77,70%
V_{TM} (V)	3	0,545 ± 0,001	81%				
P_{output} (W)	1,35	0,234 ± 0,005	82 %				

Com os dados da Tabela 9, observa-se que os valores medidos correspondem aos calculados, exceto quando comparou-se os valores nominais dos datasheets, como V_{TM} do opto acoplador e o V_{GT} do triac, devido a esses valores serem obtidos pelo fabricante em condições muito especificas.

4.4 – Programação

O controle da temperatura, controle da iluminação e interface com os circuitos elétricos é feito através do Arduino, que se comunica com o programa supervisório desenvolvido em C# permitindo que o usuário modifique os parâmetros de configuração e obtenha informações sobre o processo de maneira intuitiva.

4.4.1 – Programação do Arduino

Definição das portas a serem utilizadas e das variáveis:

```
#include <PID_v1.h>

#define sensor_temp A0
#define ldr_1 A1
#define ldr_2 A2

#define vent_ex 11
#define led 06
#define vent_aq 8

int pwm_temp = 0;
int pwm_luz = 0;

String dado;

int ligado = 0;
int res_ligada = 0;
int ex_ligado = 0;
int controle_led = 2; // 0 - desligado, 1 - ligado, 2 - auto (ldr)
int sp_led_sup;

double setpoint_tC = 30.0;
double setpoint_t;
double setpoint_led;
double temperatura;
double temperatura_p1d;
double luminosidade;
double temperatura_C;
double saida_aq;
double saida_led;

int media_ldr;

unsigned long tempoAnterior = 0;
unsigned long tempo_inicial = 0;
unsigned long tempo_final = 0;
unsigned long intervalo_pid = 0;
const long intervalo = 166.7;
```

Figura 40 - Portas e variáveis do Arduino.

Definição das variáveis que são utilizadas para fazer o controle PID do aquecedor:

```
float erro = 0;
float erroAnt = 0;

float kp = 0.0556581594865022;
float kd = -0.693052286053374;
float ki = 0.000778346310258697;
float valorD;
float valorI;
float valorP;
float valorad;
float U = 0;
float T;
float integralAnt = 0;
```

Figura 41 - Variáveis do PID.

Foi utilizada uma função para fazer a leitura da temperatura com um filtro de média móvel, a fim de eliminar ruídos que influenciariam na medida:

```
float lertemp( int porta )
{
  float valor = 0;
  for (int i = 0; i < 10; i++)
  {
    valor = valor + analogRead(porta);
  }
  return valor / 10.0;
}
```

Figura 42 - Função para média das temperaturas.

O controle PID da iluminação é feita através de uma biblioteca, não foi possível utiliza-la para o controle do aquecedor devido a problemas com a saturação do controle:

```
PID PIDled(&luminosidade, &saida_led, &setpoint_led, 1, 0, 0, DIRECT);
```

Figura 43 - Configuração PID pela biblioteca do Arduino.

A configuração das portas e primeiros ajustes são feitos na parte de setup

```
void setup() {
  // saidas

  pinMode(vent_aq, OUTPUT); // vent aquecedor
  pinMode(led, OUTPUT); // pwm iluminacao ~
  pinMode(vent_ex, OUTPUT); // pwm ventoinha exaustor ~

  pinMode(07, OUTPUT);
  pinMode(sensor_temp, INPUT);

  temperatura = lertemp(sensor_temp);
  temperatura_C  = temperatura * 500.0 / 1023.0;


  media_ldr = (analogRead(ldr_1) + analogRead(ldr_2)) / 2.0;
  luminosidade = media_ldr;

  tempo_inicial = 0;
  tempo_final = 1;

  PIDled.SetMode(AUTOMATIC);

  // inicia comunicacao serial
  Serial.begin(9600);
}
```

Figura 44 - Função Setup().

Na parte do loop o programa faz o cálculo do PID para o aquecedor após ler a temperatura:

```
void loop() {
  temperatura = lertemp(sensor_temp);
  temperatura_C  = temperatura * 500.0 / 1023.0;

  intervalo_pid = tempo_final - tempo_inicial;
  T = intervalo_pid / 1000;

  erro = setpoint_tC - temperatura_C;
  valorP = kp * erro;
  valorI = integralAnt + ki * (erroAnt + erro) * T / 2;
  valorD = kd * (erro - erroAnt) / T;
  U = valorP + valorD + valorI;
  if (U > 255)
  {
    U = 255;
  }
  if (U < 0)
  {
    U = 0;
  }

  if (erro * erroAnt < 0) {
    integralAnt = 0;
  }
  else {
    integralAnt = valorI;
  }
  erroAnt = erro;


tempo_inicial = millis();
saida_aq = U;
```

Figura 45 - Cálculo do PID para o aquecedor.

O programa se comunica com o supervisório enviando uma string alguns valores que serão utilizados por ele. O supervisório envia o caractere ‘W’ toda vez que quer realizar essa leitura. Os demais caracteres mostrados servem para alterar variáveis do programa do Arduino, podendo receber seu valor logo em seguida, como é o caso do ‘I’, ‘S’ e ‘J’.

```
// supervisorio
if (Serial.available()) {

  switch (Serial.read()) {
    case 'W': // ligado # temp # ldr1 # ldr2 # aq # ex #

      dado = String(ligado, DEC) + "#"; // ligado
      dado += String(temperatura_C, DEC) + "#"; // temperatura
      dado += String(analogRead(ldr_1), DEC) + "#"; // ldr1
      dado += String(analogRead(ldr_2), DEC) + "#"; // ldr1
      dado += String( res_ligada, DEC) + "#"; // valor pwd saida aq
      dado += String(ex_ligado, DEC) + "#"; // valor pwd saida ex
      dado += String(setpoint_tC, DEC) + "#"; // valor pwd saida ex

      Serial.println(dado);

      break;

    case 'L':
      ligado = 1;
      break;

    case 'D':
      ligado = 0;
      break;

    case 'I':
      controle_led = Serial.parseInt();
      break;

    case 'S':
      setpoint_tC =  Serial.parseFloat();
      break;

    case 'J':
      sp_led_sup =  Serial.parseInt();
      break;

  }
}
```

Figura 46 - Comunicação serial.

Caso esteja ligado, o sistema manda pulsos com intervalo de tempo proporcional à saída calculada pelo controlador PID para o triac realizar o chaveamento da resistência:

```
if (ligado) {

  unsigned long valor = (saida_aq / 255) * 166.7;
  unsigned long tempo = millis();
  if (tempo - tempoAnterior >= intervalo)
  { //se der o invervalo de 1s
    tempoAnterior = tempo;//iguala os tempos
    if (res_ligada == 1) {
      digitalWrite(07, HIGH);
      delay(valor);
    }
    digitalWrite(07, LOW);
  }
```

Figura 47 - Sistema ligado.

O sistema também verifica se a temperatura encontra-se na faixa de histerese do setpoint e ativa o aquecedor ou resfriador conforme necessário. É feito também o cálculo da saída para o driver dos LEDs usando o PID da biblioteca.

```
if (temperatura_C >= (setpoint_tC + 2.0)) {
  res_ligada = 0;
  ex_ligado = 1;
  analogWrite(vent_ex, saida_aq);
  digitalWrite(vent_aq, LOW);
} else if (temperatura_C < (setpoint_tC - 2.0)) {
  res_ligada = 1;
  ex_ligado = 0;
  analogWrite(vent_ex, 0);
  digitalWrite(vent_aq, HIGH);
}

media_ldr = (analogRead(ldr_1) + analogRead(ldr_2)) / 2.0;
luminosidade = media_ldr;
if (controle_led == 0) {
  setpoint_led = 0;
} else if (controle_led == 1) {
  setpoint_led = 1023;
} else if (controle_led == 2) {
  setpoint_led = sp_led_sup;
}

PIDled.Compute();
analogWrite(led, (int) (255- saida_led));
```

Figura 48 - Verificação do setpoint e da saída.

Caso o sistema seja desligado as saídas são todas desligadas. Ao final do loop o programa atualiza o tempo final para ser calculado pela próxima interação do PID do aquecedor.

```
}
else { // desligado
  res_ligada = 0;
  digitalWrite(07, LOW);
  digitalWrite(vent_aq, LOW);
  analogWrite(vent_ex, 0);
  analogWrite(led, 0);

}
tempo_final = millis();
}
```

Figura 49 - Sistema desligado.

4.4.2 – Programação do Supervisório

O supervisório permite que o usuário controle os parâmetros do processo e obtenha informações sobre o que está acontecendo. A figura da estufa é animada para mostrar ao usuário se o aquecedor ou resfriador está ligado, o controle de iluminação pode ser sempre ligado, sempre desligado ou definido através do nível de luz desejado (o valor dos sensores de luminosidade é usado pelo controlador e o nível de luminosidade ideal é atingido). O usuário pode acompanhar a temperatura atual do processo através do gráfico.

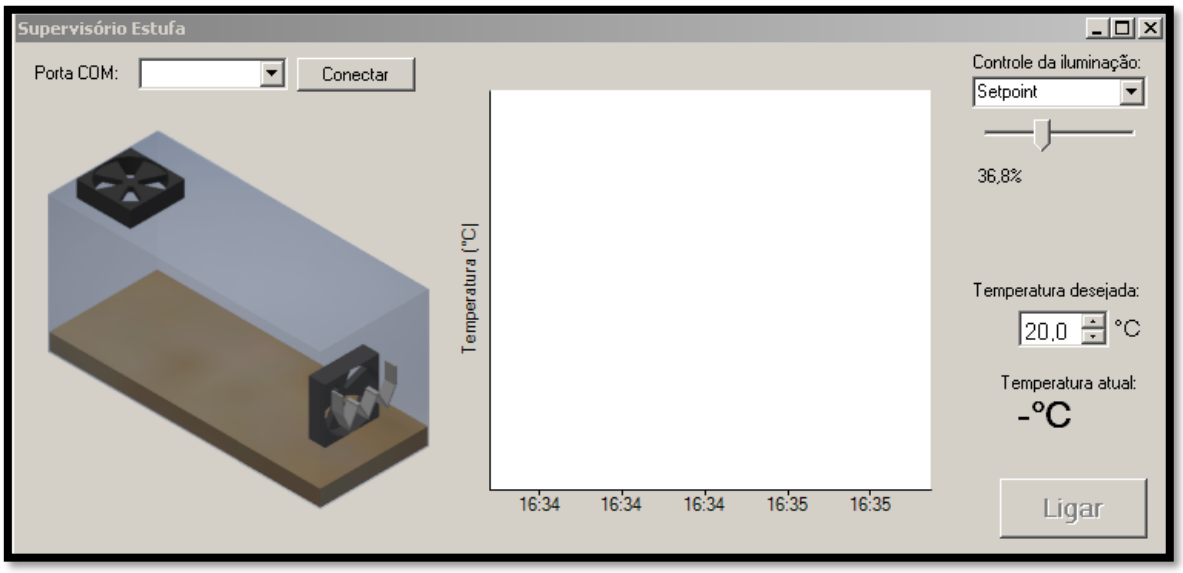


Figura 50 - Tela do supervisório.

No início do programa são declaradas variáveis globais que serão usadas por outras classes e funções.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO.Ports;  // necessário para ter ac
using System.Globalization;
using System.Timers;

namespace Estufa
{

    public partial class Form1 : Form
    {

        public static class Globals{
            public static int ligado = 0;
            public static float temperatura;
            public static float setpoint;
            public static int sp_led;
            public static int controle_led = 2;

            public static int pwm_aq;
            public static int pwm_ex;

            public static bool anim_aq;
            public static bool anim_ex;

            public static bool enviaw = true;
        }
```

Figura 51 - Variáveis do supervisório.

Quando o programa é inicializado são ajustadas algumas opções e é verificada a existência das portas seriais:

```
public Form1()
{
    InitializeComponent();
    cmb_iluminacao.SelectedIndex = 2;
    cmb_conn.SelectedItem = 0;
    chart.ChartAreas[0].AxisX.MajorGrid.LineWidth = 0;
    chart.ChartAreas[0].AxisY.MajorGrid.LineWidth = 0;
    chart.ChartAreas[0].AxisY.Title = "Temperatura (°C)";
    chart.ChartAreas[0].AxisY.TitleFont = new Font("Segoe UI", 8, FontStyle.Regular);
    chart.ChartAreas[0].AxisX.IsLabelAutoFit = false;

}

private void Form1_Load(object sender, EventArgs e)
{


    //Rotina para atualizar comboBox com as portas COMs:
    foreach (string comStr in SerialPort.GetPortNames())
    {   //Adiciona todas as COM diponíveis na lista
        cmb_conn.Items.Add(comStr);
    }
    //Seleciona a primeira posição da lista
    if (cmb_conn.Items.Count > 0)
    {
        cmb_conn.SelectedIndex = 0;
    }
}
```

Figura 52 - Ajustes iniciais.

Alguns métodos são criados para que outras classes possam alterar os valores das labels:

```
    private void SetTempText(string temp_txt)
    {

        if (this.lbl_temp_atual.InvokeRequired)
        {
            SetTempCallback d = new SetTempCallback(SetTempText);
            this.Invoke(d, new object[] { temp_txt });
        }
        else
        {
            this.lbl_temp_atual.Text = temp_txt;
        }
    }

    delegate void SetLigarCallback(string ligar_txt);

    private void SetLigarText(string ligar_txt)
    {

        if (this.btn_ligar.InvokeRequired)
        {
            SetLigarCallback d = new SetLigarCallback(SetLigarText);
            this.Invoke(d, new object[] { ligar_txt });
        }
        else
        {
            this.btn_ligar.Text = ligar_txt;
        }
    }

delegate void SetTrackTextCallback(string ligar_txt);

private void SetTrackText(string track_txt)
{

    if (this.lbl_track.InvokeRequired)
    {
        SetTrackTextCallback d = new SetTrackTextCallback(SetTrackText);
        this.Invoke(d, new object[] { track_txt });
    }
    else
    {
        this.lbl_track.Text = track_txt;
    }
}
```

Figura 53 - Métodos criados.

O botão “conectar” estabelece a comunicação com o Arduino e trata de alguns eventos que ocorrerão ao desligar.

```
    private void btn_conn_Click(object sender, EventArgs e)
    {
        //Verifica se a porta COM do Arduino foi selecionada:
        if (cmb_conn.SelectedIndex < 0)
        {
            MessageBox.Show("Adicionar/Selecionar uma COM!");
            return;
        }

        if (serialPort1.IsOpen == false)
        {   //Se a conexao serial estiver fechada... abre uma conexao com o Arduino:
            try
            {
                serialPort1.PortName = cmb_conn.Items[cmb_conn.SelectedIndex].ToString();
                serialPort1.Open();
                timerLeitura.Enabled = true;
                btn_ligar.Enabled = true;
                cmb_conn.Enabled = false;
                btn_conn.Text = "Desconectar";
            }
            catch (Exception ex)
            {   //Erro ao conectar com o Arduino:
                MessageBox.Show(ex.Message);
            }
        }
      else
      {   //Se a conexao serial estiver aberta... fecha a conexao com o Arduino:
          try
          {
              serialPort1.Write("D");
              serialPort1.Close();
              timerLeitura.Enabled = false;
              btn_ligar.Enabled = false;
              cmb_conn.Enabled = true;

              Globals.anim_ex = false;
              Globals.anim_aq = false;
              lbl_aq.Image = global::Estufa.Properties.Resources.aq0;
              btn_conn.Text = "Conectar";
          }
          catch (Exception ex)
          {   //Erro ao fechar a conexao com Arduino:
              MessageBox.Show(ex.Message);
          }
      }
}
```

Figura 54 - Conexão com a porta serial.

O programa faz o mesmo quando é fechado pelo usuário.

```
private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{   //Se a conexao serial estiver aberta... fecha a conexao com o Arduino:
    try
    {
        serialPort1.Write("D");
        serialPort1.Close();
        timerLeitura.Enabled = false;
        btn_ligar.Enabled = false;
        cmb_conn.Enabled = true;

        Globals.anim_ex = false;
        Globals.anim_aq = false;
        lbl_aq.Image = global::Estufa.Properties.Resources.aq0;
        btn_conn.Text = "Conectar";
    }
    catch (Exception ex)
    {   //Erro ao fechar a conexao com Arduino:
        MessageBox.Show(ex.Message);
    }
}
```

Figura 55 - Função para fechar o supervisório.

O programa envia periodicamente pedidos de informação para o Arduino, que são tratados quando recebidos pela porta serial para serem usados no programa (o parâmetro CultureInfo.InvariantCulture do float.Parse() garante o valor será processado independe do caractere separador de dos decimais).

```
private void serialPort1_DataReceived(object sender, SerialDataReceivedEventArgs e)
{   //receber valor pela serial...
    string RxStr = serialPort1.ReadLine();  //le o dado disponível na serial
    RxStr = RxStr.Replace("\r", "");        //retirar codigo de nova linha...

    string[] valores = RxStr.Split('#');    //divide a string pelo #
    // ligado # temp # ldr1 # ldr2 # aq # ex #

    Globals.ligado = int.Parse(valores[0]);
    Globals.temperatura = float.Parse(valores[1], CultureInfo.InvariantCulture);
    Globals.pwm_aq = int.Parse(valores[4]);
    Globals.pwm_ex = int.Parse(valores[5]);
```

Figura 56 - Informações enviadas para o Arduino.

O programa então ajusta seus valores de acordo com o recebido do Arduino, habilitando as animações se necessário.

```
if (Globals.ligado == 1)
{
    SetLigarText("Desligar");
}
else
{
    SetLigarText("Ligar");
}


SetTempText(Globals.temperatura.ToString("#0.0") + "°C");

if (Globals.pwm_aq == 1 && (Globals.ligado==1)) //valor pode ser ajustado
{
    Globals.anim_aq = true;

}
else
{
    Globals.anim_aq = false;
    lbl_aq.Image = global::Estufa.Properties.Resources.aq0;
}

if (Globals.pwm_ex == 1 && (Globals.ligado == 1)) //valor pode ser ajustado
{
    Globals.anim_ex = true;

}
else
{
    Globals.anim_ex = false;
    lbl_ex.Image = global::Estufa.Properties.Resources.re1;
}
```

Figura 57 - Ajustes dos valores recebidos.

Envia-se dados para o Arduino quando são modificados pelo usuário (durante o envio o pedido de informações através do envio do caractere W é pausado).

```
private void cmb_iluminacao_SelectedIndexChanged(object sender, EventArgs e)
{
    Globals.controle_led = cmb_iluminacao.SelectedIndex;
    //MessageBox.Show(Globals.controle_led.ToString());
    if (serialPort1.IsOpen == true)
    {

        Globals.enviaw = false;
        serialPort1.Write("I");
        serialPort1.Write(Globals.controle_led.ToString());
        Globals.enviaw = true;

    }
}
```

```
private void ud_sp_ValueChanged(object sender, EventArgs e)
{
    Globals.setpoint = (float) ud_sp.Value;
    if (serialPort1.IsOpen == true)
    {

        Globals.enviaw = false;
        serialPort1.Write("S");
        serialPort1.Write(Globals.setpoint.ToString());
        Globals.enviaw = true;

    }
}

 private void track_led_Scroll(object sender, EventArgs e)
 {
     Globals.sp_led = 1023 - track_led.Value;

     float valor_pc = (((float) track_led.Value/1023)*100);

     SetTrackText(valor_pc.ToString("#0.0") + "%");


     if (serialPort1.IsOpen == true)
     {

         Globals.enviaw = false;
         serialPort1.Write("J");
         serialPort1.Write(Globals.sp_led.ToString());
         Globals.enviaw = true;

     }
```

Figura 58 - Dados enviados ao Arduino.

Valores de temperatura recebidos são periodicamente adicionados no gráfico.

```
private void timerGraf_Tick(object sender, EventArgs e)
{

    DateTime timeValue = DateTime.Now;


        chart.Series[0].Points.AddXY(timeValue, Globals.temperatura);



}
```

Figura 59 - Animação do gráfico.

As animações são atualizadas se os sistemas de aquecimento e resfriamento são ligados:

```
int cont_aq = 0;
private void timerAnimAq_Tick(object sender, EventArgs e)
{
    if (Globals.anim_aq)
    {
        if (cont_aq == 0) lbl_aq.Image = global::Estufa.Properties.Resources.aq1;
        if (cont_aq == 1) lbl_aq.Image = global::Estufa.Properties.Resources.aq2;
        if (cont_aq == 2) lbl_aq.Image = global::Estufa.Properties.Resources.aq3;
        if (cont_aq == 3)
        {
            lbl_aq.Image = global::Estufa.Properties.Resources.aq4;
            cont_aq = 0;
        }
        cont_aq++;
    }
}

int cont_ex = 0;
private void timerAnimEx_Tick(object sender, EventArgs e)
 {

     if (Globals.anim_ex)
     {
         if (cont_ex == 0) lbl_ex.Image = global::Estufa.Properties.Resources.re1;
         if (cont_ex == 1) lbl_ex.Image = global::Estufa.Properties.Resources.re2;
         if (cont_ex == 2) lbl_ex.Image = global::Estufa.Properties.Resources.re3;
         if (cont_ex == 3)
         {
             lbl_ex.Image = global::Estufa.Properties.Resources.re4;
             cont_ex = 0;
         }
         cont_ex++;
     }
}
```

Figura 60 - Animações da estufa no supervisório.

5- CONCLUSÃO

Através desse projeto foi possível realizar o controle de temperatura e luminosidade de uma estufa utilizando Arduino como plataforma de prototipagem eletrônica. Para o controle de luminosidade foi utilizado O LDR (Light Dependent Resistor), um resistor cuja resistência varia conforme a intensidade da luz que incide sobre ele. Já para o controle de temperatura foi utilizado o LM35, os dispositivos da série LM35 são sensores de temperatura de circuito integrado de precisão, com uma tensão de saída linear proporcional à temperatura em graus Celsius. Componentes eletrônicos utilizados: Um TRIAC, (Triode for Alternating Current) e optoacopladores. Devido ao uso de sensores lineares, a ligação para o LM35 foi direta.

O circuito de iluminação foi utilizado um circuito de controle pelo PWM do Arduino, onde um optoacoplador ativa a base de um transistor que controla a potência nas fitas de led. Para o resfriador, com o atuador sendo uma ventoinha, utilizou-se do mesmo driver de potência do sistema de iluminação. O aquecedor sendo representado por uma carga resistiva AC, foi controlado por um circuito de chaveamento de um Triac, direcionado pelo optoacoplador MOC3021, onde o Arduino chavear uma porcentagem das ondas por um PWM de 6Hz (período de 166,66 ms), para obtenção do modelo da função de transferência do aquecedor, foi necessário o uso do Arduino junto a uma comunicação serial como software Processing 3, onde o Arduino iria ler os valores de temperatura a cada segundo e enviar via serial.

Depois disso, pelo comando Ident, importou-se os dois dados para a janela de identificação de modelo e optou-se por uma função de transferência com um polo e delay, resultando em um modelo com 82,62% de aproximação, que corresponde a uma boa função de transferência. Para validação com segundo teste, obteve-se uma aproximação de 62,77%. Os valores obtidos são aceitáveis, podendo passar essa função de transferência para o Workspace do Matlab, obtendo então, sua função de transferência. Como, nesse caso, a resposta é muito rápida, não é preciso um controle PID.

Depois disso, ainda pelo comando Ident, importou-se os dois dados de aquecimento para a janela de identificação de modelo e optou-se por uma função de transferência com dois polos e um zero, resultando em um modelo com 94,05% de aproximação, que corresponde a uma ótima função de transferência. Já para o segundo teste, obteve-se uma aproximação de 93,72%. Como os valores obtidos são aceitáveis, pôde-se passar essa função de transferência para o Workspace do

Matlab, obtendo então, sua função de transferência. Como a resposta corresponde aos dois testes com excelentes aproximações e como ela parece uma resposta de primeira ordem, devido a uma aproximação do Ident, obteve-se um polo e um zero muito próximos.

Após a obtenção dos dados, com a ajuda do Matlab importou-se os arquivos e tratou-se os dados. Assim, com o resultado dos dois testes importados plotou-se a curva de resposta do sistema. Após obtida a resposta ao degrau dos testes, no comando Ident, importou-se os dois dados do teste de resfriamento para a janela de identificação de modelo e optou-se por uma função de transferência com dois polos e um zero, resultando em um modelo com 96,27% de aproximação, que corresponde a uma excelente função de transferência. Para o segundo teste obteve-se uma aproximação de 89,57%.

6- REFERÊNCIAS BIBLIOGRÁFICAS

[1] REVISTA AGROPECUÁRIA. **Fique por dentro de alguns detalhes sobre a utilização de estufas agrícolas**. 2018. Disponível em: <http://www.revistaagropecuaria.com.br/2018/07/04/fique-por-dentro-de-alguns-detalhes-sobre-a-utilizacao-das-estufas-agricolas/>. Acesso em 15 jun. 2019

[2] MUIJZENBERG, E. W. **A history of greenhouses Institute for Agricultural Engineering**. 1980, Wageningen, Netherlands.

[3] EMPRESA BRASILEIRA DE PESQUISA AGROPECUÁRIAS – EMBRAPA. **Zoneamento agrícola de risco climático: instrumento de gestão de risco utilizado pelo**
seguro agrícola do Brasil. Disponível em: <http://w.agencia.cnptia.embrapa.br/Repositorio/Zoneameno_agricola_000fl7v6vox02wyiv80ispcrruh04mek.pdf>. Acesso em 15 de jun. 2019.

[4] SENTELHAS , P . C.; MONTEIRO, J. E. B. A. **Agrometeorologia dos cultivos - O fator meteorológico na produção agrícola.** INMET, 2009

[5] KAMLER, E. (1 de março de 2002). **Ontogeny of yolk-feeding fish: an ecological perspective.** *Reviews in Fish Biology and Fisheries* (em inglês). 12 (1): 79–103. ISSN 0960-3166. doi:10.1023/a:1022603204337

[6] KLUGE, R. A.; TEZOTTO U.; DA SILVA, P. P. M. **Aspectos Fisiológicos e Ambientais da Fotossíntese.** Rev. Virtual Quim., 2015, 7 (1), 56-73. Data de publicação na Web: 30 de novembro de 2014

[7] Govindjee (1975). *Bioenergetics of photosynthesis*. Boston: Academic Press. ISBN 0-12-294350-3

[8] SANTOS, Lucas Fugikawa; PEREIRA, Clayton José. Composição de cores através da calibração radiométrica e fotométrica de LEDs: teoria e experimento. **Rev. Bras. Ensino Fís.**, São Paulo , v. 35, n. 2, p. 1-8, June 2013 . Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1806-11172013000200014&lng=en&nrm=iso>. Acesso em: 15 June 2019.

[9] ARDUINO. **Arduino – PinMapping168.** Disponível em: <https://www.arduino.cc/en/Hacking/PinMapping168>. Acesso em: 15 jun. 2019.

[10] ELETRONICS NOTES. **Light Dependent Resistor LDR, Photoresistor.** Disponível em: <https://www.electronics-notes.com/articles/electronic_components/resistors/light-dependent-resistor-ldr.php>. Acesso em: 15 jun. 2019.

[11] MERCADO LIVRE. **10 Peças De Ldr Fotoresistor Pic Arduino Sensor de Luz.** Disponível em: <https://produto.mercadolivre.com.br/MLB-896896860-10-pecas-de-ldr-fotoresistor-pic-arduino-sensor-de-luz-_JM>. Acesso em: 15 jun. 2019.

[12] WIKIPÉDIA. **TRIAC.** Disponível em: <https://pt.wikipedia.org/wiki/TRIAC>. Acesso em: 15 jun. 2019.

[13] WIKIPÉDIA. **Opto-isolator.** Disponível em: <https://en.wikipedia.org/wiki/Opto-isolator>. Acesso em: 15 jun. 2019.

[14] CLIMATEMPO. **Previsão do tempo para São João da Boa Vista.** Disponível em: <https://www.climatempo.com.br/previsao-do-tempo/cidade/554/sao-joao-da-boa-vista>. Acesso em: 15 jun. 2019.

[15] WEATHER SPARK. **Clima característico em São João da Boa Vista, Brasil durante o ano.** Disponível em: <https://pt.weatherspark.com/y/30338/Clima-característico-em-São-João-da-Boa-Vista-Brasil-durante-o-ano>. Acesso em: 15 jun. 2019.

[16] MORAN, Michael J. et al. **Introdução à engenharia de sistemas térmicos: termodinâmica, mecânica dos fluidos e transferência de calor**, pg 562. Rio de Janeiro: LTC, 2005.

[17] GITHUB. **Arduino-PID-Library.** Disponível em: <https://github.com/br3ttb/Arduino-PID-Library/blob/master/PID_v1.h> . Acesso em: 14 jun. 2019.

ANEXO I

TIP31 Series(TIP31/31A/31B/31C)

Medium Power Linear Switching Applications

- Complementary to TIP32/32A/32B/32C

TO-220

1.Base 2.Collector 3.Emitter

NPN Epitaxial Silicon Transistor

Absolute Maximum Ratings T_C=25°C unless otherwise noted

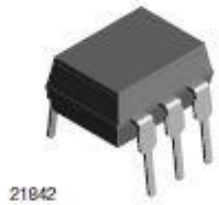
Symbol	Parameter	Value	Units
V_{CBO}	Collector-Base Voltage : TIP31	40	V
	: TIP31A	60	V
	: TIP31B	80	V
	: TIP31C	100	V
V_{CEO}	Collector-Emitter Voltage : TIP31	40	V
	: TIP31A	60	V
	: TIP31B	80	V
	: TIP31C	100	V
V_{EBO}	Emitter-Base Voltage	5	V
I_C	Collector Current (DC)	3	A
I_{CP}	Collector Current (Pulse)	5	A
I_B	Base Current	1	A
P_C	Collector Dissipation (T_C=25°C)	40	W
P_C	Collector Dissipation (T_a=25°C)	2	W
T_J	Junction Temperature	150	°C
T_{STG}	Storage Temperature	- 65 ~ 150	°C

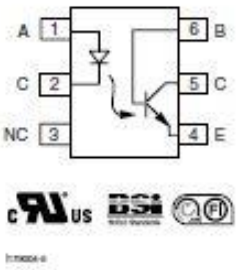
Electrical Characteristics T_C=25°C unless otherwise noted

Symbol	Parameter	Test Condition	Min.	Max.	Units
V_{CEO}(sus)	* Collector-Emitter Sustaining Voltage				
	: TIP31	I_C = 30mA, I_B = 0	40		V
	: TIP31A		60		V
	: TIP31B		80		V
	: TIP31C		100		V
I_{CEO}	Collector Cut-off Current				
	: TIP31/31A	V_{CE} = 30V, I_B = 0		0.3	mA
	: TIP31B/31C	V_{CE} = 60V, I_B = 0		0.3	mA
I_{CES}	Collector Cut-off Current				
	: TIP31	V_{CE} = 40V, V_{EB} = 0		200	µA
	: TIP31A	V_{CE} = 60V, V_{EB} = 0		200	µA
	: TIP31B	V_{CE} = 80V, V_{EB} = 0		200	µA
	: TIP31C	V_{CE} = 100V, V_{EB} = 0		200	µA
I_{EBO}	Emitter Cut-off Current	V_{EB} = 5V, I_C = 0		1	mA
h_{FE}	* DC Current Gain	V_{CE} = 4V, I_C = 1A	25		
		V_{CE} = 4V, I_C = 3A	10	50	
V_{CE}(sat)	* Collector-Emitter Saturation Voltage	I_C = 3A, I_B = 375mA		1.2	V
V_{BE}(sat)	* Base-Emitter Saturation Voltage	V_{CE} = 4V, I_C = 3A		1.8	V
f_T	Current Gain Bandwidth Product	V_{CE} = 10V, I_C = 500mA	3.0		MHz

* Pulse Test: PW≤300µs, Duty Cycle≤2%

ANEXO II





FEATURES

- Isolation test voltage 5000 V_{RMS}
- Interfaces with common logic families
- Input-output coupling capacitance < 0.5 pF
- Industry standard dual-in-line 6 pin package
- Compliant to RoHS directive 2002/95/EC and in accordance to WEEE 2002/96/EC

RoHS COMPLIANT

APPLICATIONS

- AC mains detection
- Reed relay driving
- Switch mode power supply feedback
- Telephone ring detection
- Logic ground isolation
- Logic coupling with high frequency noise rejection

AGENCY APPROVALS

- UL1577, file no. E52744
- BSI: EN 60065:2002, EN 60950:2000
- FIMKO: EN 60950, EN 60065, EN 60335

DESCRIPTION

The 4N25 family is an industry standard single channel phototransistor coupler. This family includes the 4N25, 4N26, 4N27, 4N28. Each optocoupler consists of gallium arsenide infrared LED and a silicon NPN phototransistor.

ORDER INFORMATION	
PART	**REMARKS**
4N25	CTR > 20 %, DIP-6
4N26	CTR > 20 %, DIP-6
4N27	CTR > 10 %, DIP-6
4N28	CTR > 10 %, DIP-6

ABSOLUTE MAXIMUM RATINGS (1)				
PARAMETER	**TEST CONDITION**	**SYMBOL**	**VALUE**	**UNIT**
INPUT				
Reverse voltage		V_R	5	V
Forward current		I_F	60	mA
Surge current	t ≤ 10 µs	I_{FSM}	3	A
Power dissipation		P_{diss}	100	mW
OUTPUT				
Collector emitter breakdown voltage		V_{CEO}	70	V
Emitter base breakdown voltage		V_{EBO}	7	V
Collector current		I_C	50	mA
	t ≤ 1 ms	I_C	100	mA
Power dissipation		P_{diss}	150	mW

ANEXO III

1 Features

- Calibrated Directly in Celsius (Centigrade)
- Linear + 10-mV/°C Scale Factor
- 0.5°C Ensured Accuracy (at 25°C)
- Rated for Full −55°C to 150°C Range
- Suitable for Remote Applications
- Low-Cost Due to Wafer-Level Trimming
- Operates From 4 V to 30 V
- Less Than 60-μA Current Drain
- Low Self-Heating, 0.08°C in Still Air
- Non-Linearity Only ±¼°C Typical
- Low-Impedance Output, 0.1 Ω for 1-mA Load

2 Applications

- Power Supplies
- Battery Management
- HVAC
- Appliances

3 Description

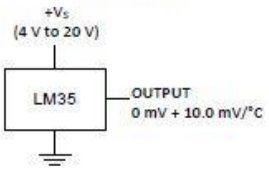
The LM35 series are precision integrated-circuit temperature devices with an output voltage linearly-proportional to the Centigrade temperature. The LM35 device has an advantage over linear temperature sensors calibrated in Kelvin, as the user is not required to subtract a large constant voltage from the output to obtain convenient Centigrade scaling. The LM35 device does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55°C to 150°C temperature range. Lower cost is assured by trimming and calibration at the wafer level. The low-output impedance, linear output, and precise inherent calibration of the LM35 device makes interfacing to readout or control circuitry especially easy. The device is used with single power supplies, or with plus and minus supplies. As the LM35 device draws only 60 μA from the supply, it has very low self-heating of less than 0.1°C in still air. The LM35 device is rated to operate over a −55°C to 150°C temperature range, while the LM35C device is rated for a −40°C to 110°C range (−10° with improved accuracy). The LM35-series devices are available packaged in hermetic TO transistor packages, while the LM35C, LM35CA, and LM35D devices are available in the plastic TO-92 transistor package. The LM35D device is available in an 8-lead surface-mount small-outline package and a plastic TO-220 package.

Device Information[1]

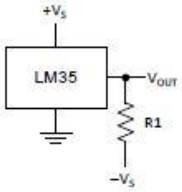
PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM35	TO-CAN (3)	4.699 mm × 4.699 mm
	TO-92 (3)	4.30 mm × 4.30 mm
	SOIC (8)	4.90 mm × 3.91 mm
	TO-220 (3)	14.986 mm × 10.16 mm

(1) For all available packages, see the orderable addendum at the end of the datasheet.

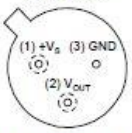
Basic Centigrade Temperature Sensor (2°C to 150°C)



Full-Range Centigrade Temperature Sensor



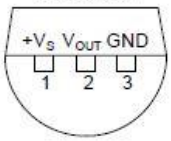
NDV Package
3-Pin TO-CAN
(Top View)



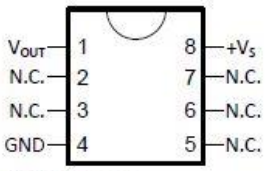
Case is connected to negative pin (GND)

Refer the second NDV0003H page for reference

LP Package
3-Pin TO-92
(Bottom View)

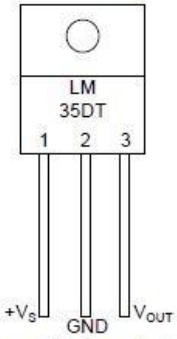


D Package
8-PIN SOIC
(Top View)



N.C. = No connection

NEB Package
3-Pin TO-220
(Top View)



Tab is connected to the negative pin (GND).

NOTE: The LM35DT pinout is different than the discontinued LM35DP

ANEXO IV

*Motorola Preferred Device

The MOC3041, MOC3042 and MOC3043 devices consist of gallium arsenide infrared emitting diodes optically coupled to a monolithic silicon detector performing the function of a Zero Voltage Crossing bilateral triac driver.

They are designed for use with a triac in the interface of logic systems to equipment powered from 115 Vac lines, such as solid–state relays, industrial controls, motors, solenoids and consumer appliances, etc.

- Simplifies Logic Control of 115 Vac Power
- Zero Voltage Crossing
- dv/dt of 2000 V/µs Typical, 1000 V/µs Guaranteed
- ***To order devices that are tested and marked per VDE 0884 requirements, the suffix "V" must be included at end of part number. VDE 0884 is a test option.***

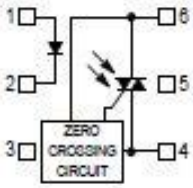
Recommended for 115/240 Vac(rms) Applications:

- Solenoid/Valve Controls
- Lighting Controls
- Static Power Switches
- AC Motor Drives
- Temperature Controls
- E.M. Contactors
- AC Motor Starters
- Solid State Relays

STYLE 6 PLASTIC

STANDARD THRU HOLE CASE 730A–04

COUPLER SCHEMATIC



1. ANODE
2. CATHODE
3. NC
4. MAIN TERMINAL
5. SUBSTRATE DO NOT CONNECT
6. MAIN TERMINAL

MAXIMUM RATINGS (T_A = 25°C unless otherwise noted)

Rating	Symbol	Value	Unit
INFRARED EMITTING DIODE			
Reverse Voltage	V_R	6	Volts
Forward Current — Continuous	I_F	60	mA
Total Power Dissipation @ T_A = 25°C Negligible Power in Output Driver Derate above 25°C	P_D	120 1.41	mW mW/°C
OUTPUT DRIVER			
Off–State Output Terminal Voltage	V_{DRM}	400	Volts
Peak Repetitive Surge Current (PW = 100 µs, 120 pps)	I_{TSM}	1	A
Total Power Dissipation @ T_A = 25°C Derate above 25°C	P_D	150 1.76	mW mW/°C
TOTAL DEVICE			
Isolation Surge Voltage(1) (Peak ac Voltage, 60 Hz, 1 Second Duration)	V_{ISO}	7500	Vac(pk)
Total Power Dissipation @ T_A = 25°C Derate above 25°C	P_D	250 2.94	mW mW/°C
Junction Temperature Range	T_J	–40 to +100	°C
Ambient Operating Temperature Range(2)	T_A	–40 to +85	°C
Storage Temperature Range(2)	T_{stg}	–40 to +150	°C
Soldering Temperature (10 s)	T_L	260	°C

1. Isolation surge voltage, V_{ISO}, is an internal device dielectric breakdown rating. For this test, Pins 1 and 2 are common, and Pins 4, 5 and 6 are common.

ANEXO V

GENERAL DESCRIPTION

Passivated triacs in a plastic envelope, intended for use in applications requiring high bidirectional transient and blocking voltage capability and high thermal cycling performance. Typical applications include motor control, industrial and domestic lighting, heating and static switching.

QUICK REFERENCE DATA

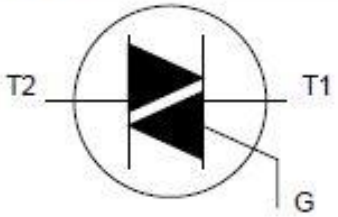
SYMBOL	PARAMETER		MAX.	MAX.	UNIT
		BT137-	600	800	
		BT137-	600F		
		BT137-	600G		
V_{DRM}	Repetitive peak off-state voltages		600	800	V
$I_{T(RMS)}$	RMS on-state current		8	8	A
I_{TSM}	Non-repetitive peak on-state current		65	65	A

PINNING - TO220AB

PIN	DESCRIPTION
1	main terminal 1
2	main terminal 2
3	gate
tab	main terminal 2

PIN CONFIGURATION

SYMBOL



LIMITING VALUES

Limiting values in accordance with the Absolute Maximum System (IEC 134).

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.		UNIT
V_{DRM}	Repetitive peak off-state voltages		-	-600 600[1]	-800 800	V
$I_{T(RMS)}$	RMS on-state current	full sine wave; $T_{mb} \leq 102$ °C	-	8		A
I_{TSM}	Non-repetitive peak on-state current	full sine wave; T_j = 25 °C prior to surge				
		t = 20 ms	-	65		A
		t = 16.7 ms	-	71		A
I^2t	I^2t for fusing	t = 10 ms	-	21		A^2s
dI_T/dt	Repetitive rate of rise of on-state current after triggering	I_{TM} = 12 A; I_G = 0.2 A; dI_G/dt = 0.2 A/µs				
		T2+ G+	-	50		A/µs
		T2+ G-	-	50		A/µs
		T2- G-	-	50		A/µs
		T2- G+	-	10		A/µs
I_{GM}	Peak gate current		-	2		A
V_{GM}	Peak gate voltage		-	5		V
P_{GM}	Peak gate power		-	5		W
$P_{G(AV)}$	Average gate power	over any 20 ms period	-	0.5		W
T_{stg}	Storage temperature		-40	150		°C
T_j	Operating junction temperature		-	125		°C