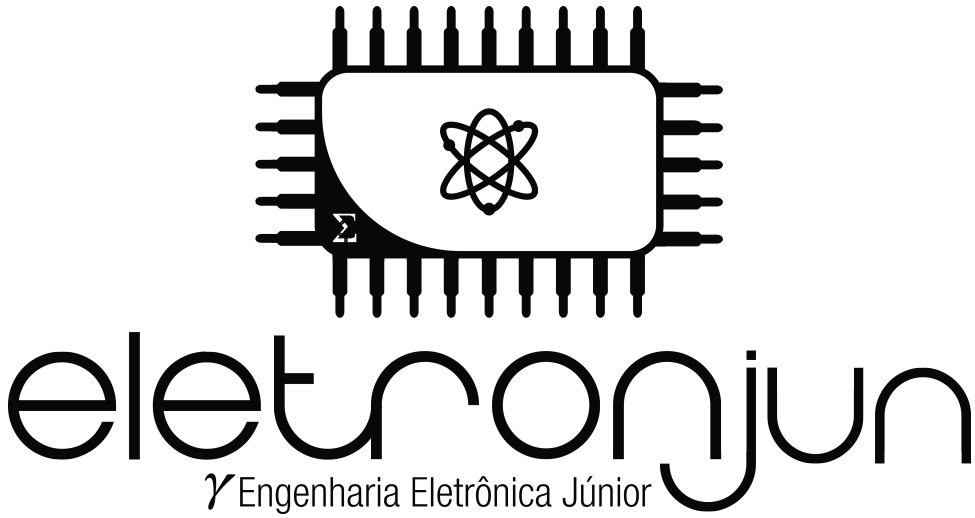


Percepção Térmica

Membros responsáveis:	Amanda Henriques, Bruna Medeiros, Gustavo Linhares, Mateus Gomes, Matheus Gois, Natália Ferreira e Vitor Guedes
Data de Início:	[dd/mm/aa]
Data de Entrega:	[dd/mm/aa]

1 Introdução

O projeto Percepção térmica consiste na construção de um sistema de controle e coleta de temperatura corporal para monitoramento médico da sensibilidade epitelial.

Com os quatro subsistemas que constituem o projeto (controle e leitura de temperatura, alimentação, interface gráfica), o protótipo será capaz de aumentar a temperatura na pele enquanto a monitora. Ao atingir uma temperatura ”quente”, ou seja, temperatura na qual o paciente sente essa sensação, o botão é apertado e os valores são armazenados e expostos no display. Dessa forma, o profissional de saúde será capaz de avaliar a situação do usuário de forma eficiente.

Este documento tem como objetivo listar todos os componentes relacionados ao hardware, software e modelagem do sistema para o desenvolvimento do projeto, bem como sua integração e funcionamento. Finalizando, assim, a fase de prototipação transcorrida até aqui.

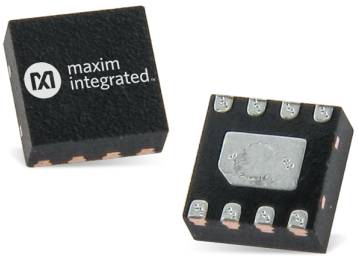
2 Hardware

2.1 Sensor MAX30205

O sensor de temperatura MAX30205 mede com precisão temperatura e fornece um alarme de superaquecimento / saída de interrupção / desligamento. Este dispositivo converte as medições de temperatura para o formato digital usando um conversor analógico-digital (ADC) de alta resolução, sigma-delta.

A precisão atende às especificações clínicas de termometria de o ASTM E1112 quando soldado na PCB final. Neste projeto, o mesmo será aplicado na palma da mão, antebraço, peito pé e coxa.

2.1.1 Especificações Técnicas



Tensão de operação	2,7 a 3,3 V
Corrente de operação	600μA
Tipo de comunicação	I2C
Precisão	0.1 °C

2.2 OLED

O Display OLED com tela de 0,96 conta com tecnologia que o possibilita ter uma iluminação própria, não sendo necessária a utilização de *backlight*, intensificando o contraste no conteúdo exibido e consecutivamente apresentando maior economia de energia.

Por já estar integrado em um módulo e possuir comunicação I2C, o Display OLED é ideal para aplicação em projetos de prototipagem baseados em microcontroladores, entre eles, o Arduino, possuindo pinagem fixa para alimentação e entrada de dados I2C.

2.2.1 Especificações Técnicas

Tensão de operação	3,3 a 5V
Corrente de operação	24mA
Tipo de comunicação	I2C
Preço Unitário	R$ 33,90

2.3 LED RGB

O LED (*light-emitting diode*) RGB, como o próprio nome indica, é um led capaz de gerar 3 cores: vermelho (*(R) red*), verde (*(G) green*) e azul (*(B) blue*). Esse componente se torna muito útil quando precisamos de uma indicação visual do que está acontecendo com o nosso sistema. Além dessas 3 cores, ligando 2 ou 3 delas simultaneamente, podemos gerar as cores de sua combinação, aumentando assim o

número de possibilidades que podemos representar.

2.3.1 Especificações Técnicas

Tensões de operação	2.0 V (Vermelho) 3.4 V (Verde) 3.4 V (Azul)
Corrente de operação	20mA
Preço Unitário	R$ 0,95

2.4 Botões

Os botões são componentes simples, porém muito bem utilizados em diversas aplicações do nosso dia-a-dia: usando um controle remoto, acessando sites pela internet, utilizando um micro-ondas etc. Esse pequeno componente possui um princípio bem simples de funcionamento, permitindo ou não a passagem de corrente. Com isso, quando apertamos ou soltamos um botão, fazemos com que alguma operação dentro do sistema ocorra, gerando assim o resultado que desejamos.

2.4.1 Especificações Técnicas

Chave Táctil

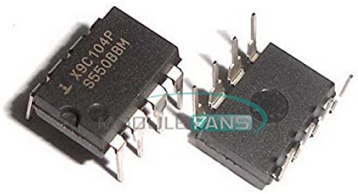
Tensão de operação	XX
Corrente de operação	50mA
Preço Unitário	R$ 0,15

Chave Botão

Tensão de operação	XX
Corrente de operação	3A - 250VCA
Preço Unitário	R$ 1,80

2.5 Potenciometro Digital X9C104

Um potenciômetro é um componente capaz de variar sua resistência. O mais comumente usado é o potenciômetro analógico, no qual possuímos um eixo que, quando girado, gera um aumento ou uma redução da resistência resultante. No caso do projeto, está sendo utilizado um potenciômetro digital, ou seja, ao invés de girarmos o eixo do componente, nós conectamos os terminais do mesmo em um microcontrolador e, utilizando suas portas de entrada e saída e um pouco de programação, nós controlamos essa resistência sem precisar fazer qualquer alteração no circuito.



Tensão de operação	5V
Corrente de operação	3mA
Resistência Máxima:	100 KΩ
Preço Unitário	$6.35

2.6 Reguladores

Um regulador de tensão é um componente capaz de realizar a manutenção da tensão de saída de um circuito. Geralmente ele é utilizado na entrada de circuitos, dentro de um sistema de controle de alimentação, com o objetivo principal de limitar sua tensão que passa por ele, evitando assim que passem tensões muito altas, capazes de danificar ou inutilizar um componente e/ou um circuito inteiro. Sendo assim, sua utilidade está intimamente atrelada à faixa de tensão em que ele consegue operar.

2.6.1 LM317

Máxima diferença entre tensão de entrada e saída	40V
Tensão de saída	1,2V a 37V
Corrente de saída	1,5A
Preço Unitário	R$1,42

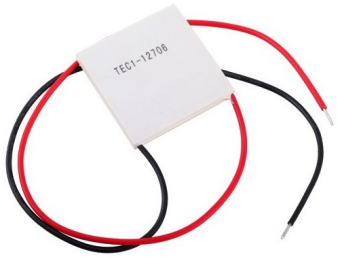
2.6.2 LM7805

Mínima diferença entre tensão de entrada e saída	2,4V
Tensão de entrada	7,4 a 28V
Tensão de saída	5V
Corrente máxima de saída	1,5A
Preço Unitário	R$0,99

2.7 Peltier

O Peltier é uma pastilha termoelétrica que, através de alimentação controlada e adequada, possui a capacidade de aquecer e resfriar sua superfície em questão de alguns segundos ou minutos, a depender da alimentação.

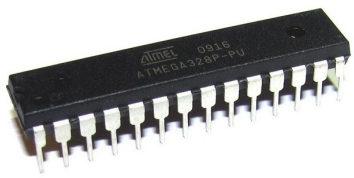
Ao colocar tensão nos terminais do Peltier, um lado irá aquecer rapidamente, enquanto o outro resfriará. Ao inverter a polaridade de ligação, a Pastilha irá efetuar o processo contrário, entrando em estado de aquecimento. Desse modo, torna-se possível uma vasta aplicação em projetos com arduino e microcontroladores no geral, como por exemplo: Coolers de CPU, mini-refrigeradores ou bebedouros e até aquecedores de água.



Nome técnico	TEC1-12709
Faixa de temperatura	-30 a 70^{o} C
Tensão de operação	0 a 15V
Preço Unitário	R$29,90

2.8 ATMega328P

O ATMega328P é um microprocessador extremamente comum em projetos eletrônicos. Ele faz parte da família de microprocessadores ATMega, que é amplamente utilizada em diversos microcontroladores do Arduino, que é muito conhecido para aplicações simples e práticas dentro da eletrônica. O ATMega328P, em específico, é utilizado no Arduino Uno R3, que é o modelo ”padrão”da marca. Com ele temos portas dedicadas a comunicação I2C e UART, portas de entrada e saída analógica e digital, possibilidade de comunicação *OneWire*, comunicação com cartão SD, clock de 16MHz e muito mais.



Memória	32Kb
Resolução do conversor AD/DA	10 Bits
Tensão de Operação	1.8V a 5.5V
Preço Unitário	R$20,66

2.9 Buzzer

O Buzzer é um componente capaz de gerar um sinal sonoro. Seu ”volume”depende da tensão em que opera. Alguns tipos de buzzer também possuem a ferramenta de alteração de frequência, possibilitando o ajuste dessa características para a produção de músicas ou bips simples como aqueles de alarmes de carro, por exemplo.

Tensão de operação	5V
Corrente máxima	42mA
Intensidade de saída	$\geq$ 85dB
Preço Unitário	R$1,52

2.10 Mosfet IRF540n

Um transistor MOSFET(*Metal Oxide Semiconductor Field Effect Transistor*) é um componente de 3 terminais que possui diversas aplicações dentro do mundo de circuitos eletrônicos e afins. Um uso comum para esse tipo de componente é utilizá-lo como uma chave, fazendo-o ”desligar”em alguma situação específica. No nosso caso, nós utilizamos essa funcionalidade para evitar gastos desnecessários com níveis de tensão e corrente muitos baixos, que não chegam nem a fazer nosso sistema funcionar totalmente. Com isso, é possível evitar o aquecimento dos componentes e gastos de energia desnecessários.

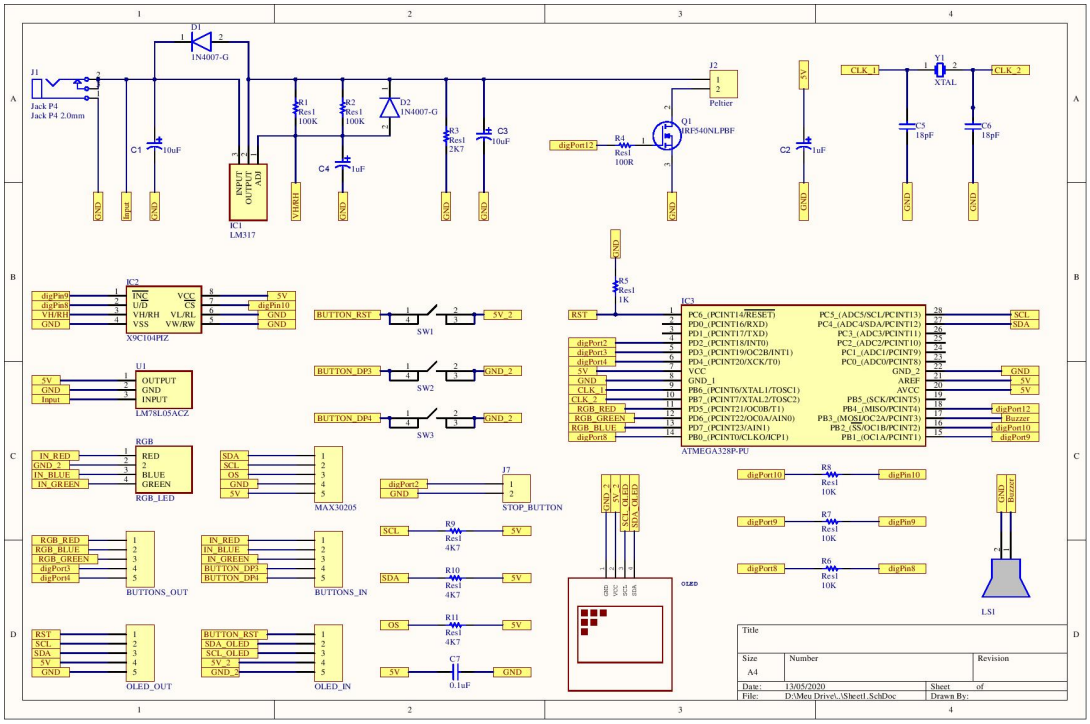
Corrente máxima no dreno (a 25^{o}C)	33 A
Dissipação máxima de potência	130W
V_{GS} máximo	$\pm$ 20V
Preço Unitário	R$3,23

2.11 Integração

A etapa de integração do projeto é onde juntamos todos os módulos, componentes, CIs (circuitos integrados) e tudo que compõe o circuito e o sistema do projeto no geral. O objetivo final dessa integração é fazer com que todos os módulos do projeto, que foram previamente testados e tidos como funcionais, continuem funcionando cor-

retamente, sem qualquer tipo de defeito ou erro inesperado causado pela união do sistema.

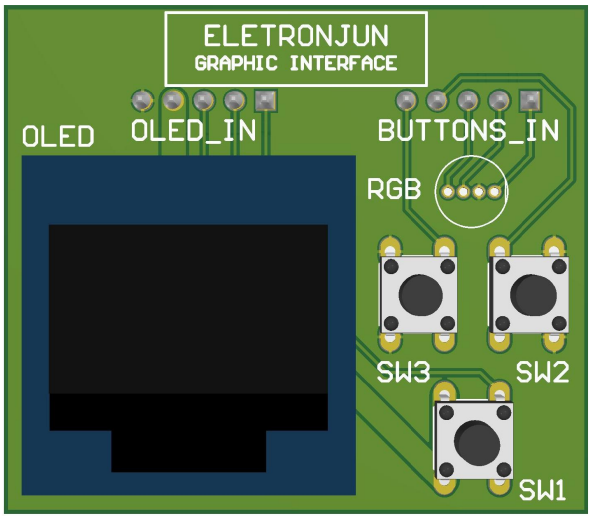
2.11.1 Esquemático



2.11.2 Layout PCI

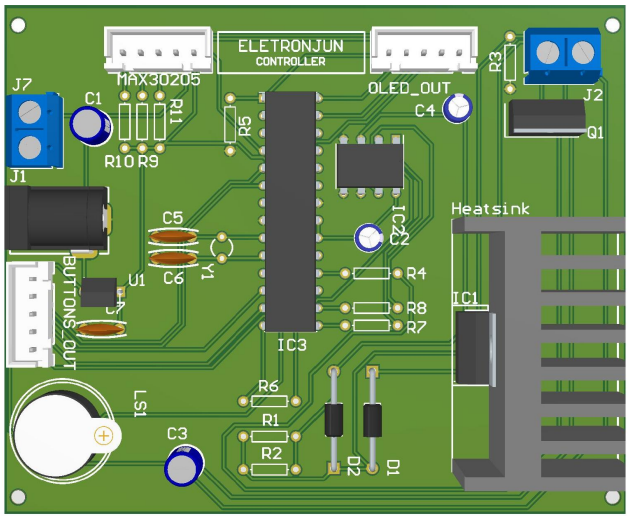
PCI (placa de circuito impresso) ou, do inglês, PCB (*Printed Circuit Board*) é a placa física em que contruímos nosso circuito. Utilizando o esquemático como um guia de como devemos ligar cada componente, realizamos o processo de layout (”desenho”da PCI, realizado por meio de softwares no computador); de fabricação, onde são feitas as trilhas e os furos do circuito, onde serão colocados os componentes durante a etapa de soldagem. Com isso, temos uma PCB/PCI completa e pronta para uso.

Placa de Interface Nesse placa estão presentes todos os componentes com os quais o usuário do sistema deverá interagir ou que deverão ser vistos por ele para indicar algo sobre o circuito: LEDs (o LED RGB, no caso), OLED e botões.



Placa de Controle Na placa de controle se encontram todos os demais componentes do circuito: aqueles responsáveis pelo circuito de alimentação, pelo circuito de controle de tensão e corrente, pela interface com software e outros. Nessa placa se encontra presente também o dissipador (*heatsink* na imagem), que é utilizado para reduzir a temperatura do LM317, que pode chegar a temperaturas de até 125^{o} C.

Além disso, o *buzzer*, responsável pelos avisos sonoros dentro do sistema, também está na placa de controle. Visto que o mesmo não necessita ser visto pelo usuário para que funcione e que sua potência sonora é capaz de ser ouvida mesmo quando isolado, não foi necessário colocá-lo na placa de interface gráfica.



3 Software

3.1 Orientação a Objetos

No projeto, o software foi orientado a objetos, visto que é um técnica que visa o reaproveitamento do software caso um subsistema futuramente precisar de upgrades. Além disso o código tem características de responsividade e consistência.

Somando a isso, foram criadas classes, no desenvolvimento de firmware, classes nesse contexto são chamadas de bibliotecas, nas quais controlam o X9C104, LED RGB, Buzzer, OLED e MAX30205.

3.2 Coleta de Dados

A coleta de informações medidas pelo sensor de temperatura é feita a partir da biblioteca ”Protocentral MAX30205”. Usando o método ”getTemperature()”da classe MAX30205, é possível pegar o valor da temperatura atual lida pelo sensor e armazená-lo em uma variável.

Exemplo de aplicação da coleta abaixo:

```
1 MAX30205 tempSensor; //instanciamento do objeto tempsensor da
     classe MAX30205
2 float temperature_coletada; //declaracao da variavel para coletar
      a temperatura
3 temperatura_coletada = tempSensor.getTemperature(); //armazena
     valor da temperatura (lida pelo sensor) na variavel '
     temperatura_coletada'
```

3.3 Controle de Periféricos

O controle de periféricos é feito por meio de classes, sendo que cada periférico possui uma classe com seus métodos e atributos.

3.3.1 Buzzer

A classe do Buzzer possui um atributo privado para armazenar o pino que o mesmo está conectado ao arduino, um método construtor e um método para ativar seu som por um período específico.

É possível ver a declaração da classe no código abaixo:

```
class Buzzer
{
private:
    int _pin; //atributo privado que armazena o pino
public:
    Buzzer(int pin); //metodo construtor que inicializa o buzzer
       com o pino certo
    void beep(int temp); //metodo para ativar o som do buzzer
       durante o periodo dado por "temp"
};
```

3.3.2 LED RGB

A classe do LED RGB possui três atributos privados para armazenar os pinos das cores verde, vermelho e azul, um método construtor e um método que acende uma cor específica.

É possível ver esta classe no código abaixo:

```
class LedRGB
{
private:
    int green; //pino verde
    int blue; //pino azul
    int red; //pino vermelho
public:
    LedRGB(int p1,int p2,int p3); //metodo construtor que
       inicializa o led com os pinos certos
    void turnOn(char colour); //metodo que acende o LED com a cor
        armazenada na variavel 'colour'
};
```

3.3.3 Botões

A classe dos Botões possui um atributo privado para armazenar o pino do botão que está sendo instanciado, dois atributos públicos para armazenar a situação atual do botão (se está pressionado ou não), um método construtor e um método para interromper o botão.

É possível ver a declaração da classe no código abaixo:

```
1  class PushButton
2  {
3  private:
4      int _pin; //variavel para armazenar pino
5
6  public:
7      PushButton(int pin); //construtor que inicializa o botao com
           seu pino
8      boolean pressed(void); //atributo para saber o estado do
           botao ao ser pressionado
9      boolean isPressed(void); //atributo para saber o estado do
           botao ao ser interrompido
10     void interrupt(void); //metodo para definir o pino de
           interrupcao
11 };
```

3.4 OLED

O display OLED é controlado a partir da biblioteca ”U8glib”, que possui funções que permitem escrever e mostrar valores na tela do display. As telas desenhadas após a etapa de prototipação sofreram pequenas mudanças em comparação ao projeto conceitual, porém, atendem as especificações feitas no documento de requisitos.

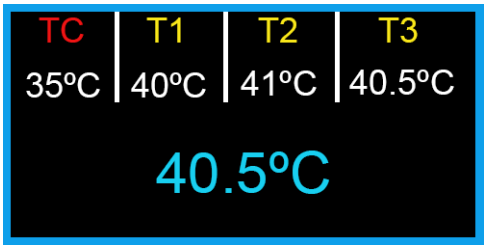
3.4.1 Layout

Aqui serão exibidas as telas conceituais apresentadas no relatório de requisitos, juntamente com as telas reais.

- Menu Inicial

- Tela da Temperatura Corporal

- Tela do Teste



3.4.2 Código de Controle

A partir das funções da biblioteca ”U8glib”, é possível desenhar cada tela de acordo com as específicações necessárias. Exemplo da montagem de uma tela do sistema:

```
1     u8g.setFont(u8g_font_8x13B); //determina a fonte que sera
          usada
2     u8g.drawRBox(0, 0, 127, 16, 2); //desenha uma caixa na parte
          superior da tela
3     u8g.setColorIndex(0); //determina a cor dos elementos
4     u8g.drawStr(17, 13, "Menu Inicial"); //escreve "Menu Inicial"
          na tela
```

3.4.3 Controle de Telas

Para o controle de telas, foi feita uma struct de funções, em que cada função representa uma tela do sistema. No código abaixo, é possível ver as principais partes do código de controle de telas:

```
1  // Struct com as funcoes das telas
2  typedef struct oled
3  {
4    void menuInit() //tela do menu inicial
5    {
6      ...
7    }
8
9    void menuTC() //tela da temperatura corporal atual
10   {
11     ...
12   }
13
14   void menuTest() //tela dos testes
15   {
16     ...
17   }
18 }
```

3.5 Controle do Peltier

O controle do peltier está diretamente ligado ao controle do potenciometro digital X9C104, para entender seu funcionamento é necessário explicar o controle do X9C104.

3.5.1 X9C104

Para o controle do X9C104 foi elaborado a classe ”x9c104”que implementa métodos que tem como função, habilitar, definir pinos de conexão com o microcontrolador, parametrizar resistência inicial e final e controlar a velocidade da mudança da resistência.

```
1  class x9c10x{
2      public:
3          x9c10x(int cs, int inc, int ud);     // Define CS, INC and
               U/D
4          void set(int init, int lt);          // Init = initial
              resistance | lt = limit resistance
5          void up(int ohms, int temp);         // Speed of Ohms/Temp
               UP
6          void down(int ohms, int temp);       // Speed of Ohms/Temp
               DOWN
7          void downOnce(int ohms, int temp);       // Speed of Ohms/
              Temp DOWN
8          void upOnce(int ohms, int temp);       // Speed of Ohms/
              Temp DOWN
9          void stop(void);                 // Define sudden stop
              button
10         void readpot(int delay_temp, String analog_pin);     //
              Read resistance
11 };
```

Os métodos upOnce e downOnce foi utilizados no projetos, definem uma iteração de subida ou descida da resistência, ou seja, quando chamados, o upOnce e downOnce, aumentam e diminuem a resistência de saída do terminal VW, respectivamente.

3.5.2 Mosfet IRF540n

O software implementado para o mosfet IRF540n, tem como função habilitar ou desabilitar a tensão nos terminais do peltier.

```
1 #define cutPeltier 12 // Definindo a palavra cutPultier como o
     inteiro 12
2 digitalWrite(cutPeltier, HIGH); // Funcao para pertimir o peltier
     receber tensao
3 digitalWrite(cutPeltier, LOW); // Fucao que nega ao peltier
     tensao em seus terminais
```

4 Integração

4.1 PCB

Fotos reais da placa:

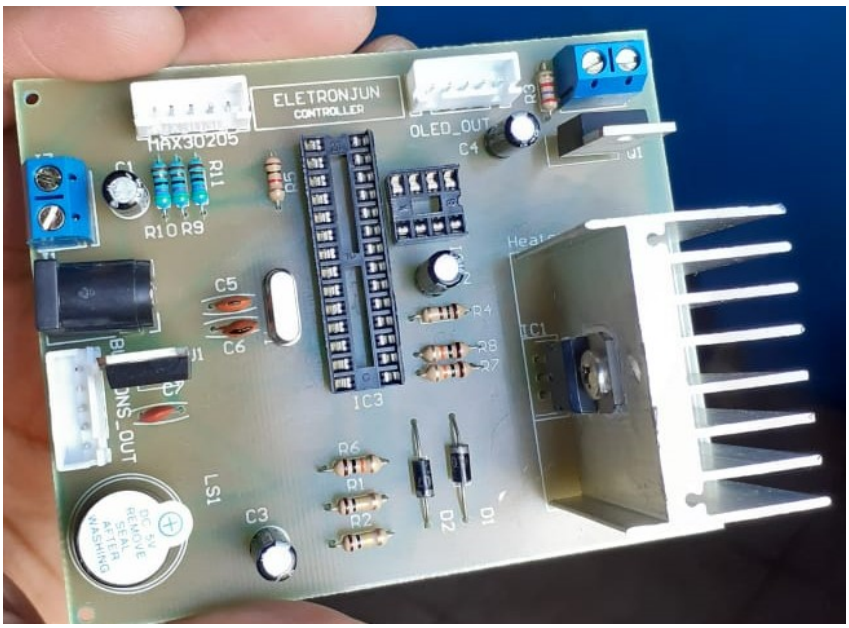
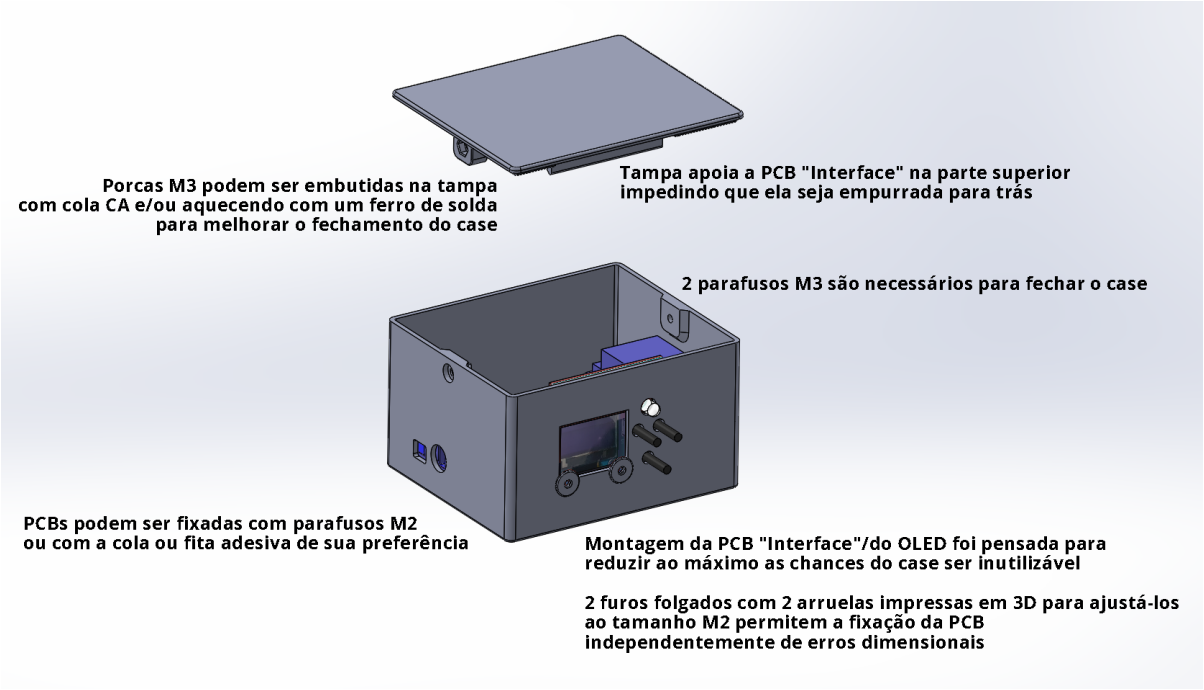


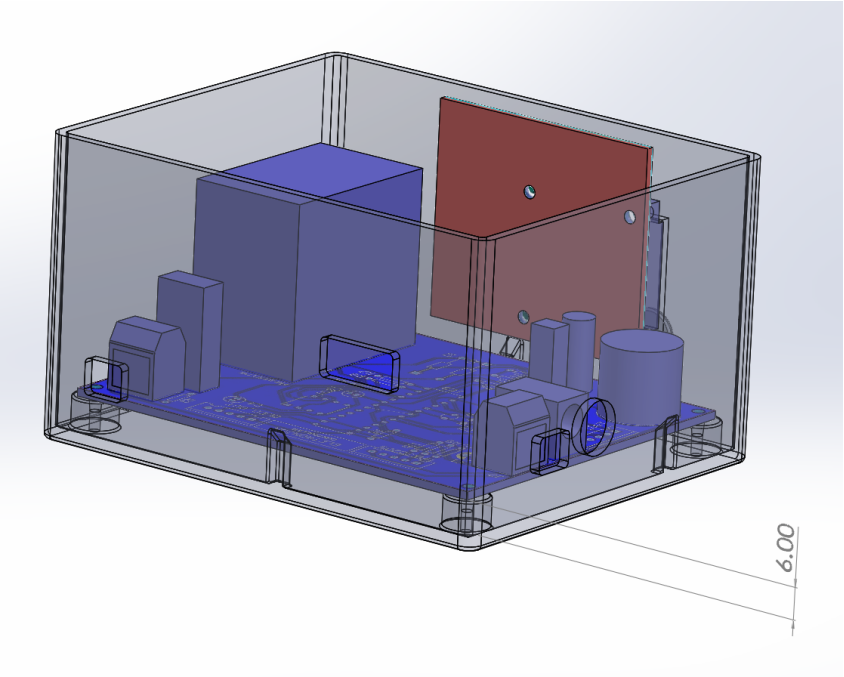
Figura 1: *Placa Controller*

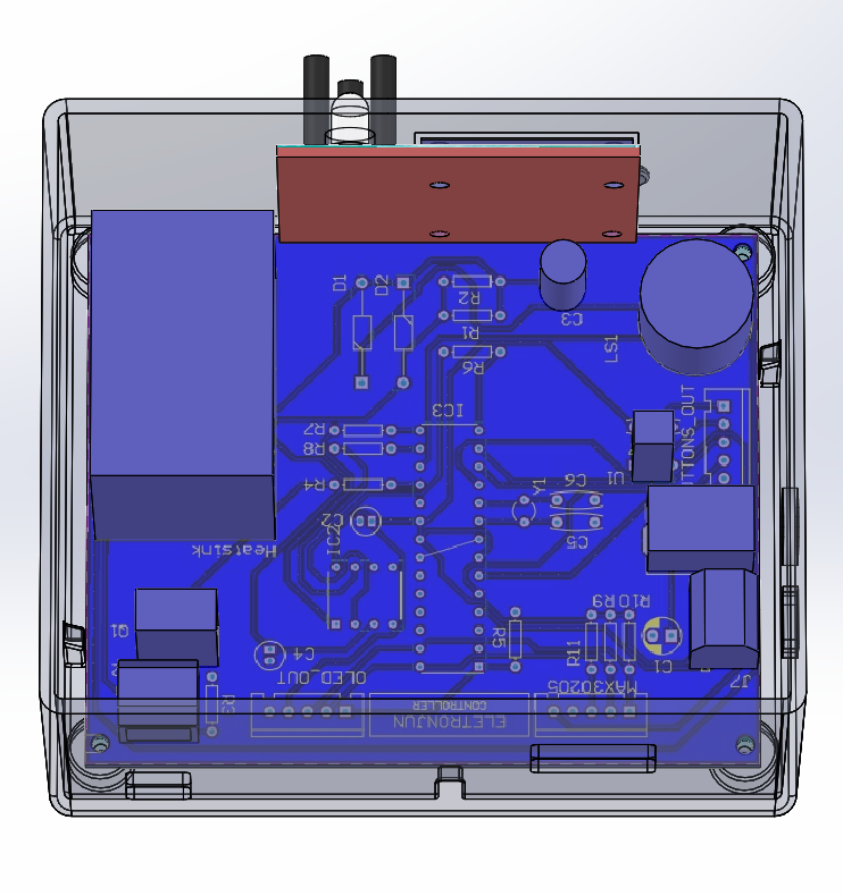
Figura 2: *Placa Graphic Interface*

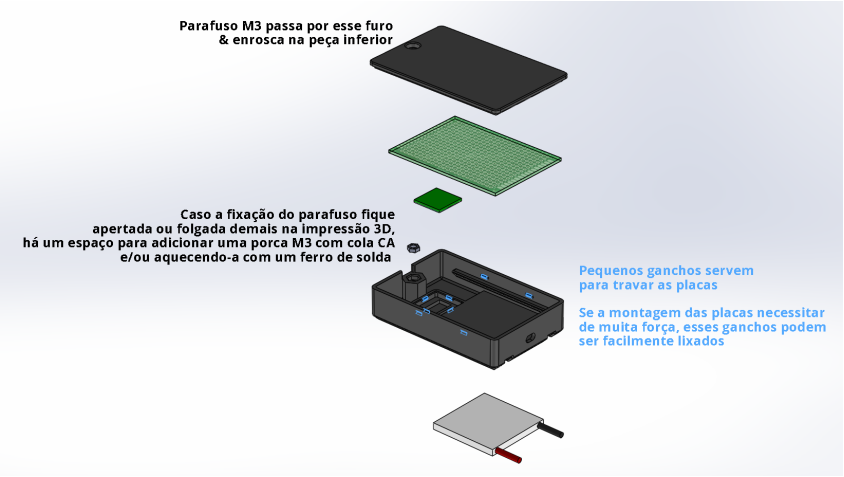
4.2 CAD

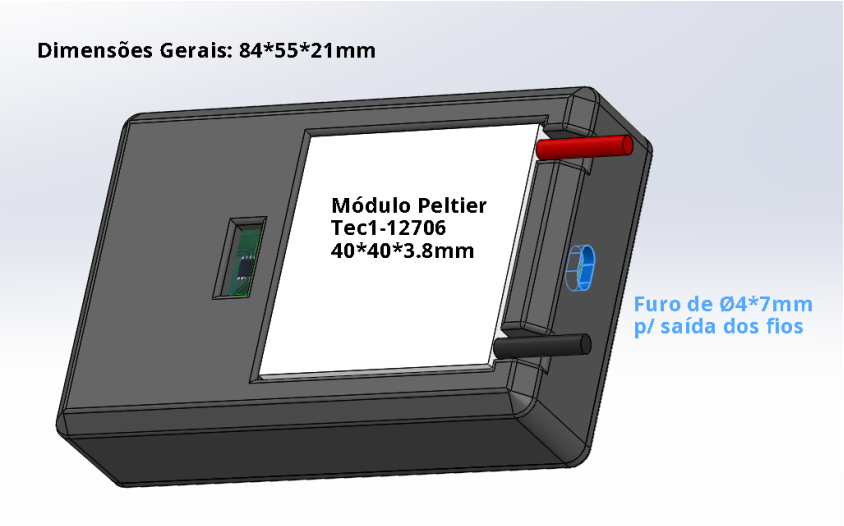
Fotos da modelagem do CAD e reais:

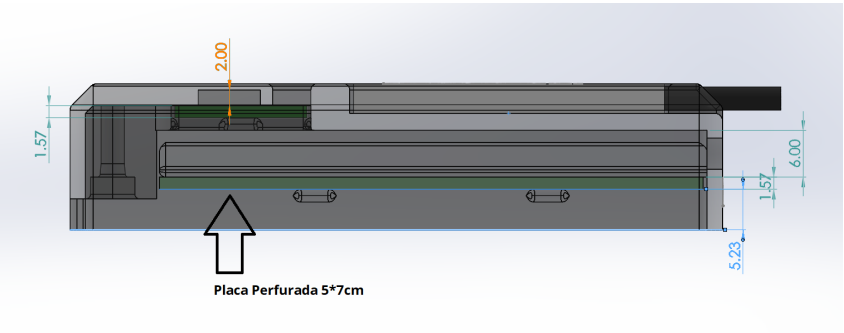



6.00


D1
D2
R2
R1
R6
C3
LS1
IC3
R7
R8
R4
C2
IC2
Y1
C6
C5
U1
BUTTONS_OUT
Heatsink
Q1
C4
OLED_OUT
R3
R10
R9
R11
C1
J7
MAX30205
ELETRONJUN
CONTROLLER


Parafuso M3 passa por esse furo
& enrosca na peça inferior
Caso a fixação do parafuso fique
apertada ou folgada demais na impressão 3D,
há um espaço para adicionar uma porca M3 com cola CA
e/ou aquecendo-a com um ferro de solda
Pequenos ganchos servem
para travar as placas
Se a montagem das placas necessitar
de muita força, esses ganchos podem
ser facilmente lixados


Dimensões Gerais: 84*55*21mm
Módulo Peltier
Tec1-12706
40*40*3.8mm
Furo de Ø4*7mm
p/ saída dos fios


2.00
1.57
6.00
1.57
5.23
Placa Perfurada 5*7cm

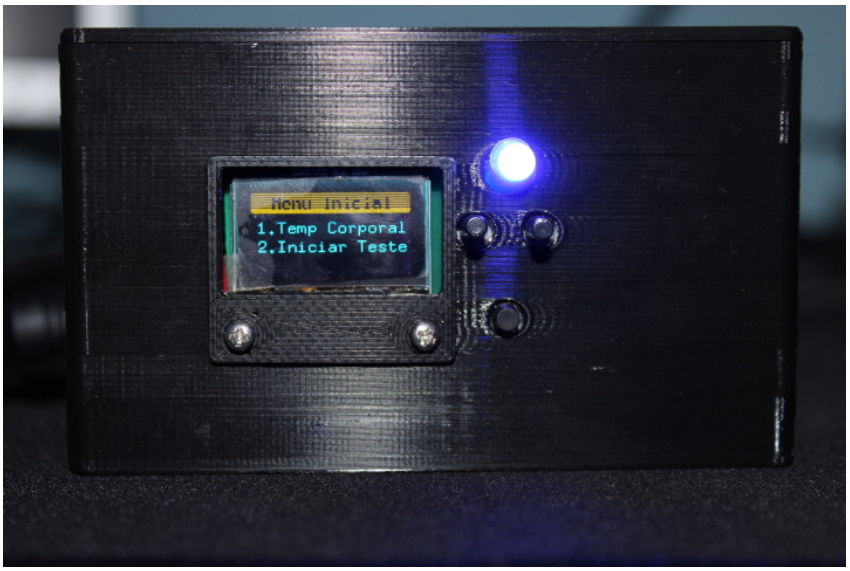


Figura 3: *Imagem frontal da Case Controller*

Figura 4: *Imagem superior do aplicador*

Figura 5: *Imagem do botão para iniciar o teste*

5 Conclusão

O projeto Percepção Térmica foi finalizado com exito, todas as técnicas e objetivos desenvolvidos no Projeto Conceitual e documento de requisitos foram atendidos, apesar de sofrerem modificações. A parte de controle de temperatura da célula de Peltier que foi alterado significativamente devido a falhas no sistema apresentado na parte conceitual, porém foi contornado com uma nova solução que se apresentou bastante eficaz na fase de teste de protótipo.

O projeto passou por dificuldades devido a época da pandemia por conta do Covid-19, materiais foram atrasados para chegar e a logística para a produção do protótipo foi complicado, com isso, o projeto não atendeu o prazo previamente planejado, porém foi finalizado.