Transporte de Órgãos com Temperatura Controlada

Arthur Faria Campos*, *16/0024242*, Sofia Consolmagno Fontes†, *16/0018234*
*† Engenharia Eletrônica, UNB-FGA, Brasília, Brasil

***Resumo*—Com o avanço da tecnologia as pastilhas de efeito Peltier estão se tornando alternativas interessante para sistemas de resfriamento. O artigo em questão descreve o desenvolvimento de um módulo eletrônico para o transporte de órgãos. Assim, utiliza-se células Peltier como sistema de refrigeração e sensores discretos associados a um microcontrolador para efetuar o controle de temperatura. Por conseguinte, para uma melhor preservação do órgão e um maior monitoramento, um software em linguagem em C que possibilita a obtenção dos dados do histórico da temperatura e o envio por bluetooth serial, para um aplicativo no celular.**

***Index Terms*—Transplante de órgãos, pastilhas termoelétricas, microcontroladores, msp430, bluetooth, sensor de temperatura.**

I. Introdução

De acordo com o Ministério da Saúde o transplante é a transferência de células, tecidos, órgãos, ou de partes do corpo de um doador para um receptor, com a finalidade de restabelecer uma função do corpo do receptor. Dessa forma, o projeto final da disciplina de Microcontroladores e Microprocessadores será a realização de um módulo eletrônico para o transplante de órgãos tendo sua temperatura monitorada e controlada pelo sensor DS18B20.

Hodiernamente, existe uma grande limitação para os transplantes de doações de órgãos, uma vez que, existe uma baixa taxa de autorização da família do doador. Assim, aproximadamente metade das famílias interrogadas não concorda que sejam retirados os órgãos e tecidos do ente falecido para doação. Conforme, a tabela 1 do Registro Brasileiro de Transplantes - Estatística de Transplantes do Ano de 2017 apresenta os dados da população brasileira relacionados a doação de órgãos [1].

Outras grandes dificuldades para a realização de transplantes são os prazos muito curtos e a dificuldade da conservação dos órgãos durante o transporte. O prazo entre a retirada do órgão do doador e o seu implante no receptor é chamado de tempo de isquemia. Os tempos máximos de isquemia normalmente aceitos para o transplante de diversos órgãos são mostrados a seguir:

Tabela I
Registro Brasileiro de Transplantes de 2017

		Necessidade anual estimada e nº de transplantes	Córnea	Rim	Fígado	Coração	Pulmão
População atual	206.081.432						
Extensão territorial (Km²)	8.514.876,60	Necessidade estimada	18.547	12.365	5.152	1.649	1.649
		Transplantes realizados	15.212	5.929	2.109	380	112

Número de Óbitos por ano	2010	2011	2012	2013	2014	2015	2016	2017
Todas as causas	1.136.947	1.170.498	1.181.166	1.220.678	1.227.039	1.264.175	Indisponível	Indisponível
Causas externas	143.256	145.842	152.013	151.683	156.942	152.136	Indisponível	Indisponível
Causas neurológicas	25.303	26.948	28.712	30.300	32.381	34.721	Indisponível	Indisponível
População (IBGE*)	190.755.799	190.755.799	190.755.799	190.755.799	190.755.799	202.768.562	204.450.649	206.081.432

IBGE - a partir do ano de 2015, o RBT passou a utilizar a estimativa da população. (antes era utilizado o CENSO)*

Tabela II
Tempo de Isquemia

Órgão	Horas
Coração	4
Figado	12
Pâncreas	20
Pulmão	6
Rim	48

Fonte: Adaptado de Associação Brasileira de Transplante de Órgãos (2009) e Saadi (2013).

O transporte de tecidos e enxertos é feito por meio da utilização de caixas térmicas compostas por material isolante, e preenchidas com gelo para manutenção do estado hipotérmico, em temperaturas próximas a 4°C, assim como os órgãos são imersos em solução isotônica e isolados por sacos plásticos [2]. Decorrente ao tempo de transporte, cuidados com o manuseio e armazenagem temporária influenciam a qualidade, a integridade, a efetivação do transplante e a diminuição da rejeição do órgão no paciente [3].

Em 2005, de acordo com a Revista Brasileira de Cirurgia Cardiovascular, o mau acondicionamento do órgão junto a solução estéril, acarretou na perda de cerca de 42 % de 1039 corações destinados para o transplante. Com tal característica, a utilização desse procedimento empregado atualmente, não existe um controle adequado e um monitoramento elaborado na refrigeração dos órgãos.

Portanto, o projeto visa um melhor aproveitamento dos órgãos doados, por meio do controle e da manutenção da faixa de temperatura interna o que garante que

as condições fisiológicas do órgão sejam preservadas, reduzindo assim as possibilidades de rejeição.

Outro benefício da utilização de um módulo eletrônico para refrigeração é a redução do peso e das dimensões das caixas térmicas do processo de transporte, auxiliando o trabalho das equipes de transplante e trazendo mais segurança ao sistema. Conforme que o Brasil apresenta vastas proporções territoriais, a funcionalidade do protótipo é recorrente em operações de longa distância e assim justifica a possibilidade de utilização de uma bateria, um adaptador no carro e uma fonte para alimentação em tomada 220V.

II. Objetivos

O projeto TOTC (transporte de Órgãos com temperatura controlada) tem como objetivo desenvolver um prototipo para o transporte de órgãos que se dará tanto por meio terrestre quanto pelo meio aéreo. Assim, com base nas pesquisas foi possível definir alguns parâmetros essenciais para o projeto.

A. Segurança

Para maior segurança no transporte haverá um monitoramento da temperatura do interior por meio de um sensor a prova d'água, o DS18B20, e a amostragem no display no exterior da caixa. Além da utilização da interface de um aplicativo de celular para o acompanhamento da temperatura e a apresentação do histórico em gráfico.

B. Versatilidade

O projeto contará com um sistema de alimentação versátil para o protótipo, uma vez que, utilizará alimentação elétrica do sistema de 12V do veículo, além de uma bateria para alimentação, em casos em que a caixa alterne entre os meios de transporte, e uma fonte para alimentação de uma tomada 220v .

C. Portabilidade

O protótipo contará com dimensões e pesos menores que as utilizadas atualmente. Assim, a caixa térmica utilizada tem proporções de $20,3{\times}16,6{\times}26,4$ cm, fabricada de polietileno e isolada por isopor, dessa forma, tem-se garantia que o tempo de conservação de produtos frios são de até 8 horas, da mesma forma que quanto maior for o volume de líquido armazenado, maior será o tempo de manutenção da temperatura.

Consequentemente, a caixa pesa 0,576 kg e com a utilização pastilhas de efeito peltier ao contrário do gelo seco terá uma redução ainda maior do peso, comparada com as utilizadas atualmente, e assim facilitará o transporte.

III. Metodologia

Para facilitar o desenvolvimento do protótipo o projeto será dividido em três áreas de trabalho: Controle, estrutura e alimentação. Sendo que, na etapa final do projeto realizaremos testes de viabilidade.

Também contará com o controle de repositórios e arquivos do projeto feitos através da plataforma GitHub a fim de facilitar a organização e armazenagem dos produtos e documentos do projeto.

A. Controle

A área de controle será o foco principal do projeto, contará com um microcontrolador MSP430 para realizar toda a comunicação entre os módulos e cálculos necessários.

B. Alimentação

Está área ficara responsável pela elaboração do circuito que alternará entre as diferentes formas de alimentação do prototipo e também da atividade do sistema de resfriamento.

C. Estrutura

O foco da área de estruturas é elaborar toda a parte mecânica do projeto, principalmente onde será alocado os controladores e o sistema de refrigeração. Assim como a análise de custos.

D. Testes

Serão realizados testes com órgãos simulados usando carne bovina, mimetizando órgãos humanos. Tendo como set point o valor de 4 ºC e um desvio aceitável de ± 2 °C, com a verificação dos dados por meio do sensor DS18B20. Dessa forma, os principais dados a serem obtidos nessas simulações são:

- O tempo que a caixa térmica leva para resfriar até a temperatura de set point;
- O tempo que essa caixa permanece com essa faixa de temperatura;
- Dados do sensor enviados por uma comunicação serial com a MSP;
- Plotagem do gráfico do histórico para analise.

IV. Requisitos

A. Requisitos técnicos

a) Formatação dos documentos: A elaboração e manutenção dos documentos produzidos no projeto deverá utilizar LaTeX de forma que a apresentação das informações fique organizada. Assim como, representará as instruções para a construção do protótipo.

b) Custo: O projeto deve ser viável economicamente para o escopo da disciplina e restrições da universidade.

B. Requisitos funcionais

a) Temperatura: Aferir a temperatura regulamente por meio do sensor DS18B20, e o sistema deve periodicamente atualizar os novos dados;

b) Disposição: Informar por meio do Display e pelo aplicativo a temperatura;

C. Requisitos de qualidade

a) Protótipo: O protótipo resultante do projeto deve ser robusto, portátil e funcional.

b) Funcionalidade: O sistema deve ser capaz de manter a temperatura controlada por volta de 4°C em estado hiportérmico, assim como seu histórico.

V. Ameaças

Uma das principais dificuldades é isolamento entre as placas da célula de peltier, dessa forma o lado quente da célula não pode entrar em contato com o lado frio. Assim como o isolamento da caixa térmica, que depois de cortada para inserção da Peltier e do cooler, deve armazenar o ar resfriado. De acordo com o fabricante a caixa térmica não aguenta fortes impactos, vibrações, contato com produtos químicos nocivos ao plástico, excesso de calor e de exposição a luz solar.

Outro problema que pode acontecer é o sistema parar de funcionar e assim não conseguir realizar o resfriamento colocando em risco o órgão transportado. Portanto, um fator muito limitante para o projeto em questão é a falta de treinamento especializado dos motoristas do transporte no acondicionamento de órgãos.

VI. Desenvolvimento

A. Sistema de Resfriamento

O efeito Peltier ocorre quando uma corrente elétrica passa por dois condutores, fazendo assim aquecer ou resfriar o ambiente. A tensão aplicada aos polos de dois materiais distintos cria uma diferença de temperatura, resultando no movimento do calor de um lado ao outro.

Consequentemente, uma pastilha de Peltier contém uma série de elementos semicondutores do tipo-p e tipo-n, conforme a Figura VI-A, agrupados como pares, os quais são soldados entre duas placas cerâmicas, eletricamente em série e termicamente em paralelo. Quando uma corrente DC passa por um ou mais pares de elementos de tipo-n e tipo-p, há uma redução na temperatura da junta (”lado frio - que é voltada para o interior do módulo”) resultando em uma absorção do calor do ambiente. Este calor é transferido pela pastilha por transporte de elétrons e emitido no outro lado (”quente - voltada para o ambiente externo”) via elétrons que movem de um estado alto para um estado baixo. A capacidade de bombeamento de calor de um resfriador é proporcional à corrente e o número de pares de elementos tipo-n e tipo-p.

Visando um melhor rendimento da célula de Peltier, foi-se convencionado que os coolers e as pastilhas serão instalados na tampa da caixa térmica, uma vez que precisa-se forçar a convecção.

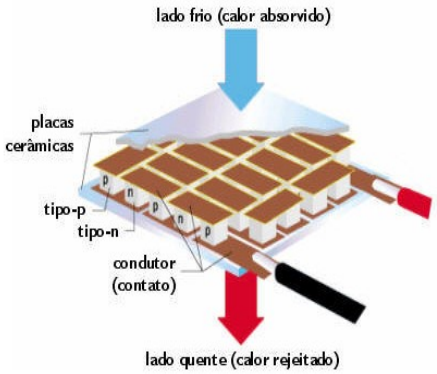

Figura 1. Pastilhas termoelétricas

1) Cálculos: A equação abaixo é utilizada para a dissipação de uma carga ativa, dessa forma é possível adequar qual célula de peltier é necessária para resfriar o projeto:

$$Q = \frac{V^2}{R} = V \times I \tag{1}$$

Q= Carga térmica ativa em watts.
V= Tensão aplicada ao sistema resfriado em volts.
R= Resistência da aplicação em ohms.
I= Corrente da aplicação em Ampére.

$$Q = 12V \times 5A = 60W \tag{2}$$

Consequentemente, 1 Watt é aproximadamente 3,41 BTU/h. Então como calculado acima, temos 60 Watts dissipados pela célula de peltier, se usada nessa configuração.

$$60Watts \times \left| \frac{3,41BTU/h}{1Watts} \right| = 204,6BTU/h. \tag{3}$$

Em média 600 BTU são suficiente para gelar uma área de 1 m², como a caixa térmica tem apenas 5 litros, uma célula é suficiente para refrigerar a caixa. Entretanto, como será utilizada uma bateria foi preciso colocar duas peltier em série para diminuir a corrente.

B. Descrição do hardware

1) Bill of Materials: Abaixo estao listados os materiais utilizados no protótipo e seus respectivos valores, sendo alguns itens retirados e outros adicionados em comparação ao ponto de contole 2. Assim como o previsto, o orçamento se manteve viável com uma variação de aproximadamente R$ 50,00 adicionados.

Tabela III
MATERIAIS PREVISTOS

Material	Quant.	Custo(R$)
MSP-EXP430FR2433	1	47,00
Sensor de temperatura DS18B20	1	5,35
Caixa térmica	1	35,00
Kit resfriamento	1	46,00
Pastilha Peltier 5A-60W	2	12,00
Bateria	1	33,00
PCB Furada (10x10)	1	9,00
Cartão de Memória SD 2GB-4GB	1	5,79
Display LCD	1	15,00
Conectores/Plugs	Div.	5,00
Rele 1 polo 5V	1	2,71
Fontes de alimentação	1	-
Adaptador para o carro	1	-
	Custo Total	R$215,85

2) Hardware Bluetooth: Para realizar a comunicação serial da MSP com o Módulo bluetooth HC-05, segue a conexão ilustrada na figura 2. Consequentemente, o smartphone requer um aplicativo que conecte com Bluetooth para receber os dados do sensor. Assim a placa MSP envia os dados do sensor e o aplicativo recebe essas informações e cria o histórico de temperatura pelos valores obtidos no sensor, formando assim um gráfico dos dados por meio do aplicativo desenvolvido no MIT App inventor.

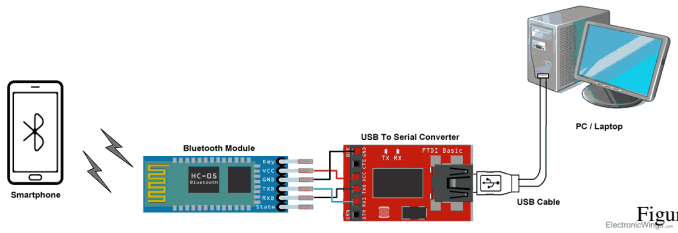

Figura 2. Conexão do Modúlo Bluetoth com a MSP430 e o aplicativo.

Para a placa MSP430G2 TI Launchpad, P1.1 é o pino Rx e P1.2 é o pino Tx, assim para conectar os jumpers com o Módulo bluetooth eles devem ser posicionados inversamente aos do microcontrolador para usar o Hardware Serial.Conforme a Figura 04 para realizar a montagem.

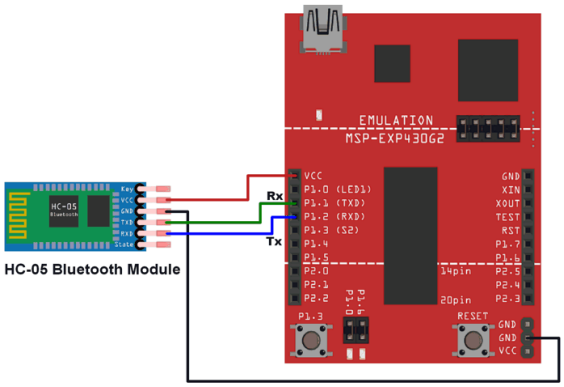

Figura 3. Interface da MSP430 conectada com o HC-05.

3) Protótipo funcional: Primeiramente, para inserir o cooler na tampa da caixa térmica foi necessária a realização de um corte do tamanho do dissipador de calor e assim por se tratar de uma tampa oca é imprescindível a vedação com Durepox e silicone nas partes laterais do corte. Logo depois foi inserido o dissipador no recorte e aplicou mais uma camada de vedação para evitar o máximo que altere a temperatura no interior da caixa.

Figura 4. Tampa da caixa tármica com vedação nos dissipadores.

Posteriormente, duas peltier foram ligadas em série pressionadas por dois dissipadores de calor, um superior

e outro inferior e entre eles e a peltier foi aplicada uma pasta térmica para melhorar a condução do calor. Dessa forma, para as peltier ficarem mais próximas, foi relevante o emprego de uma pequena tábua de MDF parafusada com o dissipador superior. Por ultimo foram adicionados os coolers sobre os dissipadores, e realizado os furos: para passarem os fios da peltier e da inserção do sensor de temperatura no interior da caixa.

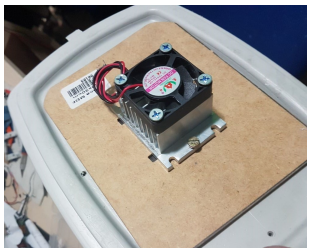

Figura 5. Cooler inferior para realizar a condução tármica.

Além do mais, a caixa foi revestida por papel alumínio, tanto na camada de isolamento em conjunto com o isopor quanto na parte interna da caixa e da tampa. A finalidade da utilização do papel alumínio é para ajudar na condução do calor e na refrigeração.

Figura 6. Interior revestido por alumínio

Em seguida na lateral da caixa foi instalada uma placa mais rígida para ajudar na sustentação da bateria, do msp e da PCB. Da mesma forma que na parte frontal, foi fixado um LCD.

Figura 7. Prototipo funcional

Posteriormente, um botão foi acoplado ao lado do Display LCD para ligar e desligar o sistema.

Figura 8. Display com botão

Um Relé de 5 pinos foi associado na placa com furos com o objetivo de controlar a temperatura a cima da condição de estado hipótermico de 4°C. Contudo, esse valor da condição hipótermica pode ser alterado dependendo da temperatura que melhor acondiciona cada órgão. Sendo assim, caso a temperatura fique inferior a esse valor o cooler é desligado, e quando aumentar a temperatura e ultrapassar esse limite, o cooler volta a funcionar.

A fonte de sinal de 5V que controla o relé, pelos os pinos digitais do msp, pode não “aguentar” a corrente necessária para ativar o relé e portanto, queimará o dispositivo. Então é necessária a utilização de um transistor para acionar o relé.

Figura 9. Sistema de Controle

C. Descrição do software

1) Código Principal: O MSP se inicializará desativando o watch dog time, e setando o system master clock para 16MHz, apos isso inicializará o sensor, seguido do display, depois as comunicações seriais do bluetooth.

Posteriormente, irá ativar o sensor e testar se está efetuando medidas, nesta etapa caso ocorra erro o sistema se reiniciará ativando um led de erro. Passado por estas duas etapas com sucesso o MSP entrará em um loop, medindo a temperatura em um determinado intervalo de tempo, mostrando-as no display e as enviando por comunicação serial.

2) Código Bluetooth: O código em C realizado na plataforma IO do software Atom, utiliza a launchpad para enviar os dados do sensor como Data para o computador via UART. Dessa forma, quanto o botão for pressionando o módulo do bluetooth começa uma conversão, e assim realiza um loop infinito. Na MSP deve-se alterar duas conexões entre o RX e o TX para que o código execute.

Consequentemente, o sensor DS18B20 motiva a realização do requisito em que ambos os dispositivos devem operar com a mesma velocidade, sendo assim utiliza-se na comunicação bluetooth a Baund rate de 32400.

São realizadas 3 funções para enviar enviar dados :

- Na função Send_data apenas um único byte é enviado, assim como as outras funções ela espera o TX buffer estar preparado para um novo valor e escreve o caractere na localização determinada pelo pointer e o incremente pointer.
- A função Send_int envia um número inteiro de bit a bit por meio de uma divisão modular.
- A função Send_string envia um número específico de bytes dependendo do tamanho do Array

Posteriormente, deve-se habilitar os pinos para transmissão serial UART quando o número de Baunds forem iguais e assim configura a transmissão serial UART com 8 bits de dados, sem paridade, começando pelo bit menos significativo e com um bit de STOP.

De modo consequente, o módulo bluetooth deve receber os valores escolhidos pelo usuário e transmitir para a msp.

Assim, quando os dados são recebidos, eles são armazenados na flag de interrupção de recebimento. Os dados são mantidos nesse registro até serem lidos por software ou quando outro quadro é recebido, caso em que é sobrescrito e o UCAxSTAT é definido. Quando o UCAxRXBUF é lido pelo software, o UCAxRXIFG é limpo. Entretanto, ocorre um erro não notificado que não é capaz de transferir um valor de temperatura selecionado no aplicativo para a msp.

3) Código do sensor DS18B20: Para lermos a temperatura do sensor de acordo com o esquema de one-wire, deve-se notar que, após a emissão de um comando ROM, é necessário emitir um comando de reset. Então a sequência de comandos será:

- Reset
- Ignorar ROM
- Converter T
- Espere por 750us
- Reset
- Ignorar ROM
- Leia o *Scratchpad*

Comandos de Função: Esses comandos permitem que o mestre grave e leia a partir da memória do rascunho do DS18B20, inicie conversões de temperatura e determine o modo de fornecimento de energia. É importante observar que o mestre pode emitir um dos comandos de função do DS18B20. A seguir, lista de comandos de função relevantes para o DS18B20

- **CONVERT T:** Usado pelo mestre para instruir o escravo para iniciar a conversão de temperatura. Se o DS18B20 for alimentado por uma fonte externa, o mestre pode emitir intervalos de tempo de leitura após o comando Convert T e o DS18B20 responderá transmitindo um 0 enquanto a conversão de temperatura estiver em andamento e 1 quando a conversão estiver concluída. A conversão de temperatura leva um mínimo de 750 ms. Sim, isso é milissegundo, você leu isso corretamente. Então, depois de emitir o comando, o comandante tem que esperar pelo mínimo de 750 ms antes que o bus busque a resposta do escravo.
- **Skip ROM:**Nós estaremos usando este comando. Usado quando há apenas um dispositivo no barramento. Instrui o Escravo que não deve ser endereçado exclusivamente.

- **READ SCRATCHPAD:** Este comando permite ao mestre ler o conteúdo do scratchpad. Os dados são lidos primeiro em LSB. O mestre pode emitir um reset para encerrar a leitura a qualquer momento se apenas uma parte dos dados do rascunho for necessária. Ele faz isso emitindo um comando de redefinição.

Interpretação de Dados: Uma vez que os dados são recebidos a bit pelo mestre, o próximo passo é tratar os dados de tal forma que você obtenha a temperatura atual lida pelo dispositivo. A resolução padrão dos dados de temperatura na energização é de 12 bits e é calibrada em graus Celsius e não em Fahrenheit. No entanto, opções estão disponíveis para obter dados em resoluções mais baixas em 9,10,11 bits, correspondendo a incrementos de 0,5 ° C, 0,25 ° C, 0,125 ° C e 0,0625 ° C, respectivamente.

Mas estaremos mantendo uma resolução de 12 bits,a mais alta.

Os dados de temperatura são armazenados como um número de complemento de dois estendido de sinal de 16 bits no registro de temperatura no dispositivo escravo com os últimos 4 bits do MSB contendo o bit de sinal para a leitura. Isso é útil durante a leitura de temperaturas abaixo de zero. Para temperaturas positivas, o bit de sinal não está definido, mas é definido como 1 para temperaturas abaixo de zero.

O sinal é lido com o LSB primeiro, teremos que invertê-lo para obter o MSB primeiro. Convertendo o padrão de bits lido em um hexadecimal de 16 bits, obtemos o valor de 0x244h = 580 Como a resolução de bits é de 0,0625 graus / bit no modo de 12 bits, multiplicamos o valor por 0,0625 para obtermos a temperatura em °C.

4) Aplicativo: O equipamento possui um aplicativo que associa um órgão à uma temperatura correspondente, alterada pelo usuário. Conforme, a proposição que cada órgão deve estar submetido a uma temperatura diferente de acordo com as melhores condições individuais de manutenção.

Figura 10. Aplicativo elaborado

Primeiramente, deve-se sincronizar o aplicativo com o módulo bluetooth apertando o botão de coração. Logo depois é possível preencher dados de grande influência para o transplante, como o nome do doador, o tipo sanguíneo, a idade e o órgão transportado. Outra funcionalidade do aplicativo é expor o tempo que a caixa térmica está sendo utilizada, e com isso é concebível plotar um gráfico da Temperatura X Tempo.

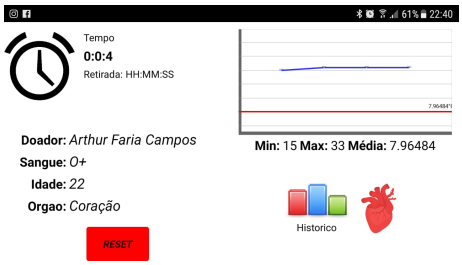

Figura 11. Aplicativo em Funcionamento

VII. Resultados

Portanto, tem-se como resultado obtido desde etapas de desenvolvimento anteriores, conseguiu-se implementar o Sensor de temperatura DS18B20 juntamente com o display 12x2 com códigos em C no MSP430G2553. Também foi concluída a estrutura do cooler como pode ser visto nas figuras 15 a 18, o prototipo funcional está finalizado. Também desenvolveu-se o codigo do bluetooth que envia dados para o celular por meio do módulo Hc-05. Assim como concluiu a elaboração do aplicativo para controle e monitoramento.

Durante a fase de testes, para avaliar a taxa de resfriamento no interior da caixa térmica, foi realizado um teste, com duração de 20 minutos, utilizando a bateria, e com a caixa vazia. Para o acompanhamento da temperatura, foi empregado o sistema de sensoriamento desenvolvido para o projeto, através de um sensor disposto no meio da lateral da caixa. O teste, é caracterizado por uma temperatura inicial de 26ºC, e após passado o tempo proposto teve uma diminuição de 6º C. Sendo assim, a eficiência do sistema de controle será em média após 1 hora de funcionamento.

Não foram realizados testes com órgãos simulados, mimetizando orgãos humanos, como previamente indicado no planejamento, uma vez que, as pastilhas peltier não atingiram o resultado esperado. Conforme as pesquisas realizadas para elaboração do projeto, o tempo para demostrar os parâmetros relativos a trocas térmicas, como o calor especíco,seriam necessário aproximadamente 200 minutos, entretanto a bateria utilizada nesse prototipo não é suficiente.

Um dos requisitos básicos para a realização do projeto era a diminuição das dimensões das caixas para transporte de órgãos utilizadas hodiernamente. Sendo assim, a caixa térmica pesava iniciantemente 0,576 kg e com a implementação da bateria, do display, da PCB furada com a MSP e o restante dos componentes inseridos pesa aproximadamente 1,800 kg. Portanto,as dimensões para locomoção do projeto continuam viáveis, uma vez que tinha-se como estimativa inicial de pesar até 12 kg.

Figura 12. Peso da caixa térmica após adição de componentes

Durante a execução do aplicativo, ele recebe os dados enviados pelo sensor, com um pequeno atraso quase imperceptível. Entretanto, para períodos maiores e uma quantidade superior de amostas o gráfico para de marcar a temperatura ao longo do tempo.

VIII. Considerações finais

Este documento visou apresentar uma base do projeto a ser desenvolvido da disciplina de Microprocessadores e Microcontroladores, do campus Gama da Universidade de Brasília com uma definição técnica mais profunda do projeto a ser desenvolvido IV. Além das propostas de organização, requisitos elicitados, cronograma a ser seguido, também foram denotadas as especificações técnicas de quase todos os materiais necessários para a implementação do protótipo.

Um fator limitante é a autonomia da bateria, uma vez que, tem a durabilidade de aproximadamente 20 minutos. Para elaboração de um projeto mais elaborado, era necessário alterar essa bateria.

Os componentes utilizados no interior da composição do módulo eletrônico para o transporte de órgãos foram basicamente produzidos à base de alumínio.Todavia, como se trata de um material que vai circular na área hospitalar, ele deve ser feito em aço inox, até mesmo por questões de higienização, contaminação e esterilização. Entretanto, depois de diversos aperfeiçoamentos, seria possível comercialização do TOTC.

O principal objetivo era a manutenção da faixa de temperatura, a qual traz impactos diretos na sobrevida do receptor, assim como também evita o descarte de tecidos e permite um maior número de transplantes. Assim sendo, a funcionalidade do protótipo é preferível em vez do método tradicional.

Referências

[1] A. B. de Trasnplante de Órgãos, *Dimensionamento dos Transplantes no Brasil e em cada estado*, V. D. Garcia, Ed., 2017.

[2] ANVISA, “Agência nacional de vigilância sanitária. transporte de órgãos é padronizado,” *Revista Liberato*, 2009. [Online]. Available: http://www.anvisa.gov.br/divulga/noticias/2009/220509_2%28link1%29.htm

[3] R. C. S. W. V. PEREIRA, W. A.; FERNANDES, “Diretrizes básicas para captação e retirada de múltiplos órgãos e tecidos. são paulo,” *ABTO*, 2009.

[4] L. E. Bohn, M. B. Haag, and A. B. Mombach., “Módulo eletrônico para transporte de órgãos em estado hipotérmico,” *Revista Liberato*, vol. 17, no. 27, pp. 01–118, 2016.

[5] L. P. E. A. T. D. Eduardo A. Di Marzo, Antonio M. Pavone, “Termovida – caixa térmica para transporte de órgãos para transplantes,” *uspdigital*, 2008. [Online]. Available: https://uspdigital.usp.br/siicusp/cdOnlineTrabalhoVisualizarResumo?numeroInscricaoTrabalho=1731&numeroEdicao=16

APÊNDICE A
DENSENVOLVIMENTO

Figura 13. Sensor de Temperatura

Figura 14. Prototipo com bateria acoplada

Figura 15. Display LDC 16x2 no prototipo

Figura 16. Bateria, PCB e MSP

Figura 17. Display com botão

Figura 18. Sistema de Controle

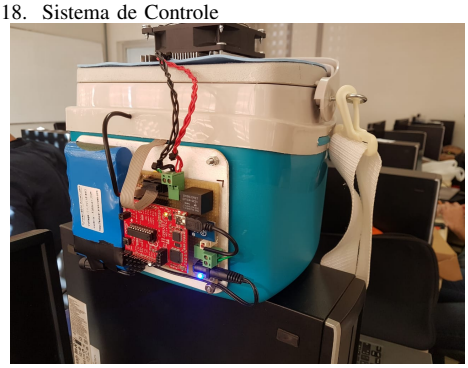

Figura 19. Visão Geral

Figura 20. Interior revestido por alumínio

APÊNDICE B
PROJETOS JÁ FEITOS

Figura 21. Projeto do protótipo e sua construção em EPS [4]

Figura 22. Projeto TERMOVIDA [5]

APÊNDICE C
CRONOGRAMA

Tabela IV
CRONOGRAMA

		Abril		Maio		Junho
T		Pesquisa do Estado da Arte e Desenvolvimento do Relatório	1			
Q			2	Ponto de Controle #2		
Q			3			
S			4		1	
S			5		2	
D	1		6		3	
S	2		7	Refinamento dos Codigos implemntados	4	
T	3		8		5	
Q	4	Ponto de Controle #1	9		6	
Q	5		10		7	
S	6		11		8	
S	7		12		9	
D	8		13		10	
S	9	Desenvolvimento Inicial dos Códigos com a biblioteca	14		11	
T	10		15		12	
Q	11		16	Prova #2	13	Ponto de Controle #4
Q	12		17		14	
S	13		18		15	
S	14		19		16	
D	15		20		17	
S	16		21	Aprimoramento do Protótipo	18	
T	17		22		19	
Q	18	Prova #1	23		20	
Q	19		24		21	
S	20		25		22	
S	21		26		23	
D	22		27		24	
S	23	Montagem Inicial do protótipo físico e Testes	28		25	
T	24		29		26	
Q	25		30	Ponto de Controle #3	27	**Entrega Final**
Q	26		31		28	
S	27				29	
S	28	Desenvolvimento do Relatório			30	
D	29					
S	30					

APÊNDICE D
CÓDIGOS

Main.c

```
#include <msp430g2533.h>
#include "totc.h"
#include "ds18b20.h"
#include "lcd16x2.h"
#include "uart.h"

float temperature=0;
int T_Limite = 10;

# define RLY1 BIT7

int main()
{
  WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer
  //------------------- Configure the Clocks -------------------//

    DCOCTL  = 0;              // Select lowest DCOx and MODx settings
    BCSCTL1 = CALBC1_16MHZ;   // Set range
    DCOCTL  = CALDCO_16MHZ;   // Set DCO step + modulation

  //---------------- Configuring the LED's ----------------------//

    P1DIR  |=  BIT0 + BIT6 + RLY1;  // P1.0 and P1.6 output
    P1OUT  &= ~BIT0 + BIT6;  // P1.0 and P1.6 = 0

  //---------------- Inicializando os Modulos ----------------------//
    Initialize_LCD();
    Clear_LCDScreen();
      Init_UART();

  //---------------- Loop Principal ----------------------//
    for(;;)
    {

        temperature=GetData();
        delay_ms(100);

    // Display LCD
    LCDSet_CursorPosition(0,2);
    Print_String("Projeto TOTC");
    LCDSet_CursorPosition(1,0);
    Print_String("Temp:");
    LCDSet_CursorPosition(1,6);
    LCD_outdec(temperature*100, 2);

    // Bluetooth UART
    // Send_String("Temp:");
    Send_Int((int)temperature*100);
    Send_String("\n");
    P1OUT &= ~LED_UART;

    // Controle de Temperatura
    if ( temperature< T_Limite) //Ativa o rele
      P1OUT |= RLY1;
    else  // desliga
      P1OUT &=~ RLY1;
    }
  return 0;
}
```

DS18B20.h

```
#ifndef DS18B20
#define DS18B20

#define DS18B20_OUT
    P1OUT
#define DS18B20_DIR
    P1DIR
#define DS18B20_SEL
    P1SEL
#define DS18B20_IN
    P1IN
#define DS18B20_DATA_IN_PIN
    BIT4
// #define DS18B20_VCC
    BIT3
// #define DS18B20_GND                BIT1

#define DS18B20_SKIP_ROM              0xCC
#define DS18B20_READ_SCRATCHPAD       0xBE
#define DS18B20_CONVERT_T             0x44

void Init_DS18B20(void);
unsigned int Reset_DS18B20 ( void );
void DS18B20_HI(void);
void DS18B20_LO(void);
void WriteZero(void);
void WriteOne(void);
unsigned int ReadBit(void);
void Write_DS18B20 (unsigned char,int );
unsigned int Read_DS18B20 ( void );
float GetData(void);
#endif
```

DS18B20.c

```
#include <msp430g2533.h>
#include "ds18b20.h"
#include "totc.h"

void DS18B20_HI()
{
    DS18B20_DIR|=DS18B20_DATA_IN_PIN; // set port as output
    DS18B20_OUT|=DS18B20_DATA_IN_PIN;   // set port high
}

void DS18B20_LO()
```

```
{
    DS18B20_DIR|=DS18B20_DATA_IN_PIN; // set port as output
    DS18B20_OUT&=~DS18B20_DATA_IN_PIN;// set port low
}

//    ------------------------------------------
//                Usar apenas Ligado Diretamente          //
//    ------------------------------------------

// void Init_DS18B20(void)
// {
//  // General GPIO Defines
//  // DS18B20_DIR |= (DS18B20_VCC + DS18B20_GND);
//  // DS18B20_OUT|=DS18B20_VCC;
//  // DS18B20_OUT&=~DS18B20_GND;
//  // P1REN |= BIT5;
//  LED1_OFF;
// }

unsigned int Reset_DS18B20 ( void )
{
    /* Steps to reset one wire bus
     * Pull bus low
     * hold condition for 480us
     * release bus
     * wait for 60us
     * read bus
     * if bus low then device present set / return var accordingly
     * wait for balance period (480-60)
     */
    int device_present=0;
    DS18B20_LO();                                   // Drive bus low
    delay_us (480);                                 // hold for 480us
    DS18B20_DIR &= ~DS18B20_DATA_IN_PIN;            //release bus. set port in input mode
    if(DS18B20_IN & DS18B20_DATA_IN_PIN)
    {
        device_present=0;
    }
    delay_us (480);                                 //wait for 480us
    return device_present;
}

void WriteZero(void)
{
    /*Steps for master to transmit logical zero to slave device on bus
     * pull bus low
     * hold for 60us
     * release bus
     * wait for 1us for recovery
     */
    DS18B20_LO();                                   // Drive bus low
    delay_us (60);                                  // sample time slot for the slave
    DS18B20_DIR &= ~DS18B20_DATA_IN_PIN;            //release bus. set port in input mode
    delay_us (1);                                   // recovery time slot
}

void WriteOne(void)
{
    /*Steps for master to transmit logical one to slave device on bus
     * pull bus low
     * hold for 5us
     * release bus
     * wait for 1us for recovery
     */
    DS18B20_LO();                                   // Drive bus low
    delay_us (5);
    DS18B20_DIR &= ~DS18B20_DATA_IN_PIN;            //release bus. set port in input mode
    delay_us (55);                                  // sample time slot for the slave
    delay_us (1);                                   // recovery time slot

}


void Write_DS18B20 (unsigned char data,int power )
{
    unsigned char i;
    for(i=8;i>0;i--)
    {

        if(data & 0x01)
        {
            WriteOne();
        }
        else
        {
            WriteZero();
        }
```

```
        data >>=1;

    }/*
    if(power == 1)
    {
        DS1820_HI();
        delay_ms(10);
    }
    */
}

unsigned int ReadBit (void)
{
    /*Steps for master to issue a read
        request to slave device on bus aka
         milk slave device
     * pull bus low
     * hold for 5us
     * release bus
     * wait for 45us for recovery
     */
    int bit=0;
    DS18B20_LO();                          //
        Drive bus low
    delay_us (5);                          //
        hold for 5us
    DS18B20_DIR &= ~DS18B20_DATA_IN_PIN;
                    //release bus. set port
         in input mode
    delay_us (10);                        //
        wait for slave to drive port
        either high or low
    if(DS18B20_IN & DS18B20_DATA_IN_PIN)
                    //read bus
    {
        bit=1;

            //if read high set bit high
    }
    delay_us (45);                        //
        recovery time slot
    return bit;
}

unsigned int Read_DS18B20 ( void )
{
    unsigned char i;
    unsigned int data=0;
    DS18B20_DIR &= ~DS18B20_DATA_IN_PIN;
                    //release bus. set port
         in input mode

     for(i=16;i>0;i--)
    {
        data>>=1;
        if(ReadBit())
        {
            data |=0x8000;
        }
    }
    return(data);
}

float GetData(void)
{
    unsigned int temp;
    Reset_DS18B20();
    Write_DS18B20(DS18B20_SKIP_ROM,0);
      Write_DS18B20(DS18B20_CONVERT_T,1);
    delay_ms(750);

    Reset_DS18B20();
    Write_DS18B20(DS18B20_SKIP_ROM,0);
    Write_DS18B20(DS18B20_READ_SCRATCHPAD
        ,0);

    LED0_ON;
    temp = Read_DS18B20();

      LED0_OFF;
    if(temp<0x8000)
    {
        return(temp*0.0625);
    }
    else
    {
        temp=(~temp)+1;LED0_OFF;
        return(temp*0.0625);
    }

}
```

LCD16x2.h

```
#ifndef LCD16x2
#define LCD16x2

#define     LCD_DIR                 P2DIR
#define     LCD_OUT                 P2OUT

#define     LCD_PIN_D4              BIT0
             // P1.4
#define     LCD_PIN_D5              BIT1
             // P1.5
#define     LCD_PIN_D6              BIT2
             // P1.6
#define     LCD_PIN_D7              BIT3
             // P1.7
#define     LCD_PIN_RS              BIT4
             // P1.0
#define     LCD_PIN_EN              BIT5
             // P1.1

#define     LCD_PIN_MASK   ((LCD_PIN_RS |
    LCD_PIN_EN | LCD_PIN_D7 | LCD_PIN_D6 |
     LCD_PIN_D5 | LCD_PIN_D4))

#define     FALSE                   0
#define     TRUE                    1

```

```
void LCDSet_CursorPosition(char Row, char Col);
void Clear_LCDScreen();
void Initialize_LCD(void);
void Print_String(char *Text);
void LCD_outdec(long data, unsigned char ndigits);
void SendByte(char ByteToSend, int IsData);

#endif
```

LCD16x2.c

```
#include <msp430g2533.h>
#include "lcd16x2.h"

void Pulse_LCD()
{
    // pull EN bit low
    LCD_OUT &= ~LCD_PIN_EN;
    __delay_cycles(200);

    // pull EN bit high
    LCD_OUT |= LCD_PIN_EN;
    __delay_cycles(200);

    // pull EN bit low again
    LCD_OUT &= (~LCD_PIN_EN);
    __delay_cycles(200);
}

//
    ----------------------------------------

// Envie um byte no barramento de dados no modo de 4 bits
// Isso requer o envio dos dados em dois trechos.
// O nibble alto primeiro e depois o nible baixo
//
// Parametros:
//
// ByteToSend - o unico byte para enviar
//
// IsData - definido como TRUE se o byte for um dado de caractere
// FALSE se for um comando
                          //
 //
     ----------------------------------------

void SendByte(char ByteToSend, int IsData)
{
    // clear out all pins
    // set High Nibble (HN) -
    LCD_OUT &= (~LCD_PIN_MASK);
    LCD_OUT |= ((ByteToSend & 0xF0) >> 4);

    if (IsData == TRUE)
        LCD_OUT |= LCD_PIN_RS;
    else
        LCD_OUT &= ~LCD_PIN_RS;

    Pulse_LCD();

    // clear out all pins
    // set Low Nibble (LN) -
    LCD_OUT &= (~LCD_PIN_MASK);
    LCD_OUT |= ((ByteToSend & 0x0F) );

    if (IsData == TRUE)
        LCD_OUT |= LCD_PIN_RS;
    else
        LCD_OUT &= ~LCD_PIN_RS;

    Pulse_LCD();
}

// Set the position of the cursor on the screen
void LCDSet_CursorPosition(char Row, char Col)
{
    char address;
    // construct address from (Row, Col) pair
    if (Row == 0)
        address = 0;
    else
        address = 0x40;

    address |= Col;
    SendByte(0x80 | address, FALSE);
}

// Clear the screen data and return the
// cursor to home position
void Clear_LCDScreen()
{
    SendByte(0x01, FALSE);
    SendByte(0x02, FALSE);
}

// Initialize the LCD after power-up.
void Initialize_LCD(void)
{
    // set the MSP pin configurations
    // and bring them to low
    LCD_DIR |= LCD_PIN_MASK;
    LCD_OUT &= ~(LCD_PIN_MASK);

    // wait for the LCD to warm up and reach
    // active regions. Remember MSPs can power
    // up much faster than the LCD.
    __delay_cycles(100000);

    // initialize the LCD module
    // 1. Set 4-bit input
    LCD_OUT &= ~LCD_PIN_RS;
    LCD_OUT &= ~LCD_PIN_EN;
    LCD_OUT = 0x20;
```

```
    Pulse_LCD();

    // set 4-bit input - second time.
    // (as reqd by the spec.)
    SendByte(0x28, FALSE);

    // 2. Display on, cursor on, blink cursor
    SendByte(0x0E, FALSE);

    // 3. Cursor move auto-increment
    SendByte(0x06, FALSE);
}

// Print a string of characters to the screen
void Print_String(char *Text)
{
    char *c;
    c = Text;

    while ((c != 0) && (*c != 0))
    {
        SendByte(*c, TRUE);
        c++;
    }
}

void LCD_outdec(long data, unsigned char ndigits)
{
    unsigned char sign, s[6];
    unsigned int i;
    sign = ' ';

    if(data < 0)
    {
        sign='-';
        data = -data;
    }

    i = 0;

    do {
        s[i++] = data % 10 + '0';
        if(i == ndigits) {
            s[i++]='.';
        }
    } while( (data /= 10) > 0);

    s[i] = sign;
    for (i = 0; i<5; i++)
    {
        SendByte(s[4-i], TRUE);
    }
}
```

totc.c

```
#include "totc.h"

void delay_ms(int ms)
{
    while (ms--)
    {
        __delay_cycles(16000); // set for 16Mhz change it to 1000 for 1 Mhz
    }

}

void delay_us(int us)
{
    while (us--)
    {
        __delay_cycles(8); // set for 16 Mhz change it to 1000 for 1 Mhz
    }
}
```

UART.h

```
#ifndef UART
#define UART

#define     RX                BIT1
#define     TX                BIT2
#define     LED_UART              BIT6

void Send_Data(volatile unsigned char c);
void Send_Int(int n);
void Send_String(char str[]);
void Init_UART();

#endif
```

UART.c

```
#include <msp430g2553.h>
#include  "uart.h"


void Send_Data(volatile unsigned char c)
{
    while((IFG2&UCA0TXIFG)==0);
    UCA0TXBUF = c;
}

void Send_Int(int n)
{
    int casa, dig;
    if(n==0)
    {
        Send_Data('0');
        return;
    }

    if(n<0)
    {
        Send_Data('-');
        n = -n;
    }

    for(casa = 1; casa<=n; casa *= 10);
```

```
27
28      casa /= 10;
29      while(casa>0)
30      {
31          dig = (n/casa);
32          Send_Data(dig+'0');
33          n -= dig*casa;
34          casa /= 10;
35      }
36  }
37
38  void Send_String(char str[])
39  {
40      int i;
41      for(i=0; str[i]!= '\0'; i++)
42          Send_Data(str[i]);
43  }
44
45  void Init_UART()
46  {
47    //--------- Setting the UART function
          for P1.1 & P1.2 --------//
48
49     P1SEL  |=  RX + TX;  // P1.1 UCA0RXD
          input
50     P1SEL2 |=  RX + TX;  // P1.2 UCA0TXD
          output
51
52    //------------ Configuring the UART(
          USCI_A0) ----------------//
53
54     UCA0CTL1 |=  UCSSEL_2 + UCSWRST;  //
          USCI Clock = SMCLK,USCI_A0 disabled
55     UCA0BR0   =  130;                 //
          104 From datasheet table-
56     UCA0BR1   =  6;                   // -
          selects baudrate =9600,clk = SMCLK
57     UCA0MCTL  =  UCBRS_1;             //
          Modulation value = 1 from datasheet
58     // UCA0STAT |=  UCLISTEN;            //
           loop back mode enabled
59     UCA0CTL1 &= ~UCSWRST;             //
          Clear UCSWRST to enable USCI_A0
60
61    //---------------- Enabling the
          interrupts ------------------//
62
63     // IE2 |= UCA0TXIE;                  //
           Enable the Transmit interrupt
64     IE2 |= UCA0RXIE;                  //
          Enable the Receive  interrupt
65     _BIS_SR(GIE);                     //
          Enable the global interrupt
66  }
```