

# Transporte de Órgãos com Temperatura Controlada

Arthur Faria Campos\*, 16/0024242, Sofia Consolmagnó 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 termoeletrônicas, microcontroladores, msp430, bluetooth, sensor de temperatura.

## I. INTRODUÇÃO

**D**E 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 imagem abaixo 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

População atual	206.081.432	Necessidade anual estimada e nº de transplantes						
		Córnea						
		Rim	Fígado	Coração	Pulmão			
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. Fontes: era utilizado o CENSO

Tabela II  
TEMPO DE ISQUEMIA

Órgão	Horas
Coração	4
Fígado	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 utilizado 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 protótipo 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 20,3×16,6×26,4 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, 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

Esta área ficará responsável pela elaboração do circuito que alternará entre as diferentes formas de alimentação do protótipo e também das potência 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 °C e um desvio aceitável de  $\pm 2$  °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 análise.

## 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 regularmente 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) *Discordância de dados*: Informar o usuário quando os parâmetros de temperatura estiverem fora dos parâmetros propostos, por meio de um aviso por LED.

### 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 hipotérmico, 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, sendo assim, para eles utilizarem um aplicativo, um visor de LED usar e um adaptador da fonte no carro durante o transporte dificultaria o traslado.

## 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 o cooler junto com o módulo eletrônico será instalado na tampa da caixa térmica, uma vez que precisa-se forçar a convecção.

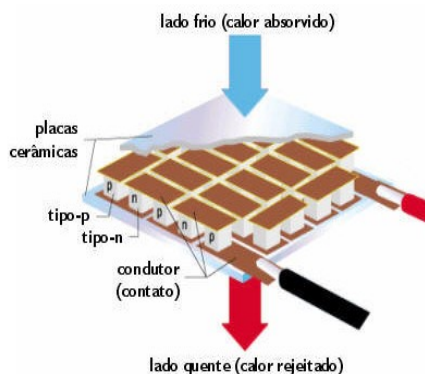


Figura 1. Pastilhas termoeletricas

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:

$$Q = \frac{V^2}{R} = V \times I \quad (1)$$

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.

$$Q = 12V \times 5A = 60W \quad (2)$$

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.

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

Em média 600 BTU são suficiente para gelar uma área de 1 m². Como a caixa térmica tem apenas 5 litros. É suficiente essa célula para refrigerar a caixa.

## B. Descrição do hardware

1) *Bill of Materials:* A partir de uma análise de custos inicial, foi acrescentado alguns materiais para serem utilizados para esse 3º ponto de controle. Assim como o previsto o orçamento se manteve viável com uma variação de aproximadamente R\$ 60,00 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	5,62×2
Bateria	2	64,76
Módulo de memória para cartão SD	1	6,90
Cartão de Memória SD 2GB-4GB	1	5,79
Display LCD	1	15,00
Fontes de alimentação	1	-
Adaptador para o carro	1	-
Custo Total		R\$237,04

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 o histórico de temperatura do módulo de transporte, formando assim um gráfico dos dados, desenvolvido no MIT App inventor.

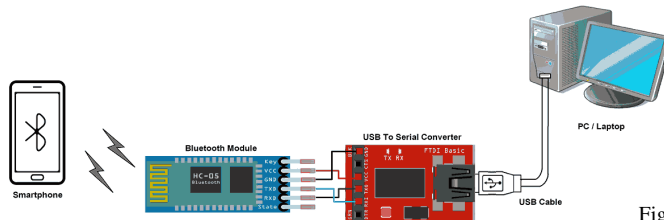
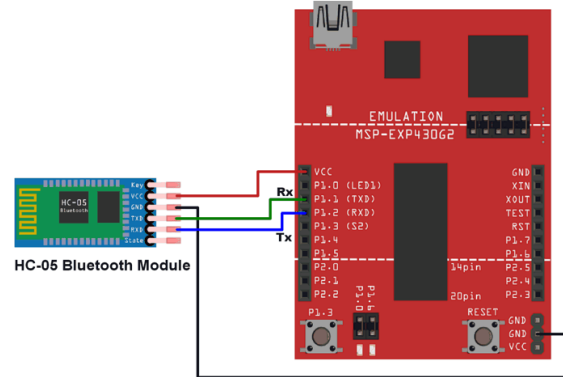


Figura 2. Conexão do Módulo Bluetooth com a MSP430 e o aplicativo.

Para a placa MSP4230g 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.



e outro inferior e entre eles e a peltier foi aplicada uma pasta térmica para melhor 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 a inserção do sensor de temperatura.

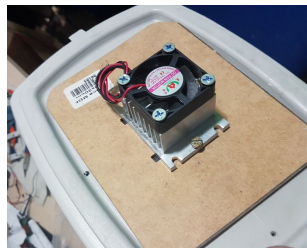


Figura 5. Cooler inferior para realizar a condução térmica.

Em seguida na lateral da caixa foi instalada uma placa mais rígida para ajudar na sustentação da bateria, e posteriormente um circuito impresso em PCB. Assim como, na parte frontal, foi fixado um LCD.



Figura 6. Protótipo funcional.

### C. Descrição do software

1) *Código Principal:* O MSP se inicializará em modo Low-Power ("LPM4"), e ficará ocioso até que o botão On/Off seja pressionado. Caso seja pressionado ele inicializará as todas as variáveis globais, depois ativando o display, com a tela de inicio. Após, irá ativar o sensor e testar se esta 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 e mostrando-as no display.

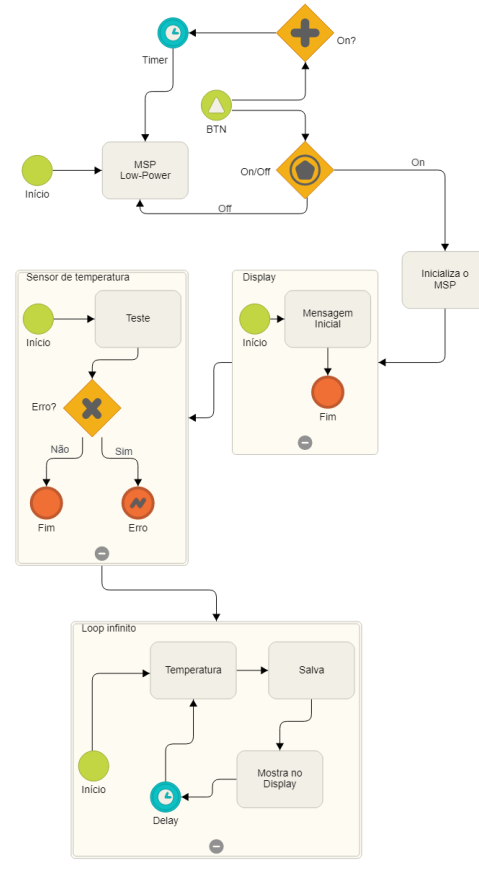


Figura 7. Fluxograma Inicial

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, quando o botão for pressionado 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 a Baund rate 32400.

São realizadas 3 funções para 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.

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.

## VII. RESULTADOS

Na segunda etapa de desenvolvimento conseguimos implementamos 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 VII. Também desenvolvemos o código do bluetooth mais com erros devido ao clock de 16MHz.



Figura 8. Protótipo mostrando no display 16x2 a temperatura do sensor

## VIII. CONSIDERAÇÕES FINAIS

Foram apresentadas as propostas de organização, os requisitos elicitados e o cronograma a ser seguido IV.

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. 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.

Tendo em vista a entrega do protótipo, o projeto ainda precisa de melhorias, sendo elas, principalmente, orçamentais e técnicas. Da mesma forma, que ocorreu pequeno atraso devido ao tempo de entrega dos principais componentes, dessa forma a realização do protótipo físico foi prejudicada.

## REFERÊNCIAS

- [1] A. B. de Trasplante 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](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 DESENVOLVIMENTO



Figura 9. Sensor de Temperatura



Figura 10. Prototipo com bateria acoplada



Figura 11. Display LDC 16x2 no prototipo



Figura 12. Bateria, PCB e MSP

## APÊNDICE B PROJETOS JA FEITOS

Figura 13. Projeto do protótipo e sua construção em EPS [4]



Figura 14. Projeto TERMOVIDA [5]



## APÊNDICE C CRONOGRAMA

Tabela IV  
CRONOGRAMA

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

```

1 #include <msp430g2533.h>
2 #include "ishan.h"
3 #include "ds18x20.h"
4 #include "hd44780.h"
5 #include "UART.h"
6
7 float temperature=0;
8 void main()
9 {
10
11     WDTCTL = WDTPW + WDTHOLD; // Stop
12         watchdog timer
13
14     //setup DCO to 1MHZ
15     BCSCCTL1 = CALBC1_16MHZ;
16     DCOCTL = CALDCO_16MHZ;
17
18     //General GPIO Defines
19     LED_DIR |= (LED_0 + LED_1); // Set P1
20     .0 and P1.6 to output direction
21     LED_OUT &= ~(LED_0 + LED_1); // Set
22     the LEDs off
23
24     InitDS18B20();
25     InitializeLcm();
26     ClearLcmScreen();
27     Init_P1();
28     Init_UART(BAUD_9600);
29
30     for(;;)
31     {
32         temperature=GetData();
33         delay_ms(100);
34         LcmSetCursorPosition(0,0);
35         PrintStr("DS18B20-2");
36         LcmSetCursorPosition(1,0);
37         PrintStr("Temp:");
38         LcmSetCursorPosition(1,5);
39         HD44780_outdec(temperature*100, 2);
40
41         Send_Int(temperature*100);
42
43
44         Send_Int(100);
45
46         Send_String("OK");
47         P1IFG &= ~BTN;
48         P1OUT &= ~LED;
49
50     }
51
52 }
```

### DS18B20.h

```

1 #ifndef DS18X20_H_
2 #define DS18X20_H_
3
4 #define DS1820_OUT P1OUT
5 #define DS1820_DIR P1DIR
6 #define DS1820_SEL P1SEL
7 #define DS1820_IN P1IN
8 #define DS1820_DATA_IN_PIN BIT5
9 // #define DS1820_VCC
10     BIT3
11 // #define DS1820_GND BIT1
12
13
14 #define DS1820_SKIP_ROM 0xCC
15 #define DS1820_READ_SCRATCHPAD 0xBE
16 #define DS1820_CONVERT_T 0x44
17
18 void InitDS18B20(void);
19 unsigned int ResetDS1820 ( void );
20 void DS1820_HI(void);
21 void DS1820_LO(void);
22 void WriteZero(void);
23 void WriteOne(void);
24 unsigned int ReadBit(void);
25 void WriteDS1820 (unsigned char,int );
26 unsigned int ReadDS1820 ( void );
27 float GetData(void);
28 #endif /*DS18X20_H_*/
```

### DS18B20.c

```

1 #include <msp430g2452.h>
2 #include "ishan.h"
3 #include "ds18x20.h"
4
5 void DS1820_HI()
6 {
7     DS1820_DIR|=DS1820_DATA_IN_PIN; //set
8     port as output
9     DS1820_OUT|=DS1820_DATA_IN_PIN; //set
10    port high
11 }
12 void DS1820_LO()
13 {
14     DS1820_DIR|=DS1820_DATA_IN_PIN; //set
15     port as output
16     DS1820_OUT&=~DS1820_DATA_IN_PIN; //set
17     port low
18 }
19 void InitDS18B20(void)
20 {
21     //General GPIO Defines
22     // DS1820_DIR |= (DS1820_VCC +
23     DS1820_GND);
24     // DS1820_OUT|=DS1820_VCC;
25     // DS1820_OUT&=~DS1820_GND;
26     LED1_OFF;
27 }
28 unsigned int ResetDS1820 ( void )
29 {
30 }
```

```

25  /* Steps to reset one wire bus          68
26  * Pull bus low                          69
27  * hold condition for 480us
28  * release bus
29  * wait for 60us                        70
30  * read bus                            71
31  * if bus low then device present set
    / return var accordingly
32  * wait for balance period (480-60)    72
33  */
34  int device_present=0;
35  DS1820_LO();                          73
                                         //
    Drive bus low
36  delay_us (480);                      74
                                         //
    hold for 480us
37  DS1820_DIR &= ~DS1820_DATA_IN_PIN;    76
    //release bus. set port
    in input mode
38  if(DS1820_IN & DS1820_DATA_IN_PIN)    77
39  {
40      device_present=0;
41  }
42  delay_us (480);                      83
                                         //wait
    for 480us
43  return device_present;                86
44  }
45  void WriteZero(void)                  87
46  {
47      /*Steps for master to transmit logical
        zero to slave device on bus
48      * pull bus low                    92
49      * hold for 60us                  93
50      * release bus                    94
51      * wait for 1us for recovery      95
52      */
53
54      DS1820_LO();                      98
                                         //
    Drive bus low
55  delay_us (60);                        101
                                         //
    sample time slot for the slave
56  DS1820_DIR &= ~DS1820_DATA_IN_PIN;    103
    //release bus. set port
    in input mode
57  delay_us (1);                        106
                                         //
    recovery time slot
58
59  }
60
61  void WriteOne(void)                   111
62  {
63      /*Steps for master to transmit logical
        one to slave device on bus
64      * pull bus low                    113
65      * hold for 5us                    114
66      * release bus                    115
67      * wait for 1us for recovery
                                         116
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

```

116         hold for 5us
DS1820_DIR &= ~DS1820_DATA_IN_PIN;
        //release bus. set port
        in input mode
117 delay_us (10);
        //
        wait for slave to drive port
        either high or low
118 if(DS1820_IN & DS1820_DATA_IN_PIN)
        //read bus
119 {
120     bit=1;
        //if read high set bit high
121 }
122 delay_us (45);
        //
        recovery time slot
123 return bit;
124
125 }
126 unsigned int ReadDS1820 ( void )
127 {
128     unsigned char i;
129     unsigned int data=0;
130     DS1820_DIR &= ~DS1820_DATA_IN_PIN;
131     //release bus. set port
132     in input mode
133
134     for(i=16;i>0;i--)
135     {
136         data>>=1;
137         if(ReadBit())
138         {
139             data |=0x8000;
140         }
141     }
142
143 }
144
145 return(data);
146 }
147
148 float GetData(void)
149 {
150     unsigned int temp;
151     ResetDS1820();
152     WriteDS1820(DS1820_SKIP_ROM,0);
153     WriteDS1820(DS1820_CONVERT_T,1);
154     delay_ms(750);
155     ResetDS1820();
156     WriteDS1820(DS1820_SKIP_ROM,0);
157     WriteDS1820(DS1820_READ_SCRATCHPAD,0);
158     LED0_ON;
159     temp = ReadDS1820();
160     LED0_OFF;
161     if(temp<0x8000)
162     {
163
164         return(temp*0.0625);

```

```

165     }
166     else
167     {
168         temp=(~temp)+1;LED0_OFF;
169         return(temp*0.0625);
170     }
171 }
172

```

## LCD.h

```

1 /*
2  * hd44780.h
3  *
4  * Created on: 28.08.2012
5  * Author: user
6  */
7
8 #ifndef HD44780_H_
9 #define HD44780_H_
10
11 #define LCM_DIR P2DIR
12 #define LCM_OUT P2OUT
13
14 //
15 // Define symbolic LCM - MCU pin mappings
16
17 // We've set DATA PIN TO 4,5,6,7 for easy
18 // translation
19 #define LCM_PIN_D4 BIT0
20 // P1.4
21 #define LCM_PIN_D5 BIT1
22 // P1.5
23 #define LCM_PIN_D6 BIT2
24 // P1.6
25 #define LCM_PIN_D7 BIT3
26 // P1.7
27 #define LCM_PIN_RS BIT4
28 // P1.0
29 #define LCM_PIN_EN BIT5
30 // P1.1
31
32 #define LCM_PIN_MASK ((LCM_PIN_RS |
33 LCM_PIN_EN | LCM_PIN_D7 | LCM_PIN_D6 |
34 LCM_PIN_D5 | LCM_PIN_D4))
35
36 #define FALSE 0
37 #define TRUE 1
38
39 void LcmSetCursorPosition(char Row, char
40 Col);
41 void ClearLcmScreen();
42 void InitializeLcm(void);
43 void PrintStr(char *Text);
44 void HD44780_outdec(long data, unsigned
45 char ndigits);
46 void SendByte(char ByteToSend, int IsData)
47 ;
48 #endif /* HD44780_H_ */

```

## LCD.c

```

1 //
2 // MSP430 LCD Code
3 //
4
5 #include "msp430g2553.h"
6 #include "hd44780.h"
7
8 //
9 // Routine Desc:
10 //
11 // This is the function that must be
12 // called
13 // whenever the LCM needs to be told to
14 // scan it's data bus.
15 // Parameters:
16 //
17 // void.
18 //
19 // Return
20 //
21 // void.
22 //
23 void PulseLcm()
24 {
25     //
26     // pull EN bit low
27     //
28     LCM_OUT &= ~LCM_PIN_EN;
29     __delay_cycles(200);
30
31     //
32     // pull EN bit high
33     //
34     LCM_OUT |= LCM_PIN_EN;
35     __delay_cycles(200);
36
37     //
38     // pull EN bit low again
39     //
40     LCM_OUT &= (~LCM_PIN_EN);
41     __delay_cycles(200);
42 }
43
44
45 //
46 // Routine Desc:
47 //
48 // Send a byte on the data bus in the 4
49 // bit mode
50 // This requires sending the data in two
51 // chunks.
52 // The high nibble first and then the low
53 // nibble
54 // Parameters:
55 //
56 // ByteToSend - the single byte to send
57 //
58 // IsData - set to TRUE if the byte is
59 // character data
60 // FALSE if its a command
61 // Return
62 //
63 // void.
64 void SendByte(char ByteToSend, int IsData)
65 {
66     //
67     // clear out all pins
68     //
69     LCM_OUT &= (~LCM_PIN_MASK);
70
71     //
72     // set High Nibble (HN) -
73     // usefulness of the identity mapping
74     // apparent here. We can set the
75     // DB7 - DB4 just by setting P1.7 - P1
76     // .4
77     // using a simple assignment
78     //
79     LCM_OUT |= (ByteToSend & 0xF0);
80     LCM_OUT |= ((ByteToSend & 0xF0) >>
81         4);
82
83     if (IsData == TRUE)
84     {
85         LCM_OUT |= LCM_PIN_RS;
86     }
87     else
88     {
89         LCM_OUT &= ~LCM_PIN_RS;
90     }
91
92     //
93     // we've set up the input voltages to
94     // the LCM.
95     // Now tell it to read them.
96     //
97     PulseLcm();
98
99     //
100     // set Low Nibble (LN) -
101     // usefulness of the identity mapping
102     // apparent here. We can set the
103     // DB7 - DB4 just by setting P1.7 - P1
104     // .4
105     // using a simple assignment
106     //
107     LCM_OUT &= (~LCM_PIN_MASK);
108     LCM_OUT |= ((ByteToSend & 0x0F) );
109
110     if (IsData == TRUE)
111     {
112         LCM_OUT |= LCM_PIN_RS;
113     }
114     else
115     {
116         LCM_OUT &= ~LCM_PIN_RS;
117     }
118
119     //

```

```

114 // we've set up the input voltages to 172
115 // the LCM. 173
116 // Now tell it to read them. 174
117 // 175
118 PulseLcm(); 176
119 } 177
120 178
121 // 179
122 // Routine Desc: 180
123 // 181
124 // Set the position of the cursor on the 182
125 // screen 183
126 // Parameters: 184
127 // 185
128 // Row - zero based row number 186
129 // 187
130 // Col - zero based col number 188
131 // 189
132 // Return 190
133 // 191
134 // void. 192
135 // 193
136 void LcmSetCursorPosition(char Row, char 194
137 Col) 195
138 { 196
139     char address; 197
140 198
141 // 199
142 // construct address from (Row, Col) 200
143 // pair 201
144 // 202
145 if (Row == 0) 203
146 { 204
147     address = 0; 205
148 } 206
149 else 207
150 { 208
151     address = 0x40; 209
152 } 210
153 211
154 address |= Col; 212
155 213
156 SendByte(0x80 | address, FALSE); 214
157 } 215
158 216
159 // 217
160 // Routine Desc: 218
161 // 219
162 // Clear the screen data and return the 220
163 // cursor to home position 221
164 // Parameters: 222
165 // 223
166 // void. 224
167 // 225
168 // Return 226
169 // 227
170 // void. 228
171 //

```

```

172 void ClearLcmScreen()
173 {
174     //
175     // Clear display, return home
176     //
177     SendByte(0x01, FALSE);
178     SendByte(0x02, FALSE);
179 }
180
181 //
182 // Routine Desc:
183 //
184 // Initialize the LCM after power-up.
185 //
186 // Note: This routine must not be called
187 // twice on the
188 // LCM. This is not so uncommon
189 // when the power
190 // for the MCU and LCM are
191 // separate.
192 //
193 // Parameters:
194 //
195 // void.
196 //
197 // Return
198 //
199 // void.
200
201 void InitializeLcm(void)
202 {
203     //
204     // set the MSP pin configurations
205     // and bring them to low
206     //
207     LCM_DIR |= LCM_PIN_MASK;
208     LCM_OUT &= ~(LCM_PIN_MASK);
209
210     //
211     // wait for the LCM to warm up and
212     // reach
213     // active regions. Remember MSPs can
214     // power
215     // up much faster than the LCM.
216     //
217     __delay_cycles(100000);
218
219     //
220     // initialize the LCM module
221     //
222     // 1. Set 4-bit input
223     //
224     LCM_OUT &= ~LCM_PIN_RS;
225     LCM_OUT &= ~LCM_PIN_EN;
226
227     LCM_OUT = 0x20;
228     PulseLcm();
229 }

```



```

229 // set 4-bit input - second time.
230 // (as reqd by the spec.)
231 //
232 SendByte(0x28, FALSE);
233
234 //
235 // 2. Display on, cursor on, blink
    cursor
236 //
237 SendByte(0x0E, FALSE);
238
239 //
240 // 3. Cursor move auto-increment
241 //
242 SendByte(0x06, FALSE);
243 }
244
245 //
246 // Routine Desc
247 //
248 //
249 // Print a string of characters to the
    screen
250 //
251 // Parameters:
252 //
253 // Text - null terminated string of
    chars
254 //
255 // Returns
256 //
257 // void.
258 //
259 void PrintStr(char *Text)
260 {
261     char *c;
262
263     c = Text;
264
265     while ((c != 0) && (*c != 0))
266     {
267         SendByte(*c, TRUE);
268         c++;
269     }
270
271 }
272
273 void HD44780_outdec(long data, unsigned
    char ndigits){
274     unsigned char sign, s[6];
275     unsigned int i;
276     sign = ' ';
277     if(data < 0) {
278         sign='-';
279         data = -data;
280     }
281     i = 0;
282     do {
283         s[i++] = data % 10 + '0';
284         if(i == ndigits) {
285             s[i++]='.';
286

```

```

287     } while( (data /= 10) > 0);
288     s[i] = sign;
289     for (i = 0; i<5; i++){
290         SendByte(s[4-i], TRUE);
291     }
292 }
293
294 //
295 // Routine Desc
296 //
297 // main entry point to the sketch
298 //
299 // Parameters
300 //
301 // void.
302 //
303 // Returns
304 //
305 // void.
306 //
307 void hd44780_test(void)
308 {
309     WDTCTL = WDTPW + WDTHOLD;
        // Stop watchdog timer
310
311     InitializeLcm();
312
313     ClearLcmScreen();
314
315     PrintStr("Hello World!");
316     LcmSetCursorPosition(1,0);
317     PrintStr("Welcome to MSP430");
318     while (1)
319     {
320         __delay_cycles(1000);
321     }
322
323 }

```

#### Delay.c

```

1 #include "ishan.h"
2
3 void delay_ms(int ms)
4 {
5     while (ms-->0)
6     {
7         __delay_cycles(16000); // set for
            16Mhz change it to 1000 for 1
            Mhz
8     }
9 }
10
11
12
13 void delay_us(int us)
14 {
15     while (us-->0)
16     {
17         __delay_cycles(8); // set for 16
            Mhz change it to 1000 for 1
            Mhz
18     }
19 }

```

```

19 }
20
UART.h
1 #define RX BIT1
2 #define TX BIT2
3 #define BTN BIT3
4 #define LED BIT6
5 #define BAUD_9600 0
6 #define BAUD_19200 1
7 #define BAUD_38400 2
8 #define BAUD_56000 3
9 #define BAUD_115200 4
10 #define BAUD_128000 5
11 #define BAUD_256000 6
12 #define NUM_BAUDS 7
13
14 void Send_Data(volatile unsigned char c);
15 void Send_Int(int n);
16 void Send_String(char str[]);
17 void Init_P1(void);
18 void Init_UART(unsigned int
    baud_rate_choice);

UART.c
1 #include <msp430g2553.h>
2 #include <legacymsp430.h>
3 #include "UART.h"
4
5
6 void Send_Data(volatile unsigned char c)
7 {
8     while ((IFG2&UCA0TXIFG)==0);
9     UCA0TXBUF = c;
10 }
11
12 void Send_Int(int n)
13 {
14     int casa, dig;
15     if(n==0)
16     {
17         Send_Data('0');
18         return;
19     }
20     if(n<0)
21     {
22         Send_Data('-');
23         n = -n;
24     }
25     for(casa = 1; casa<=n; casa *= 10);
26     casa /= 10;
27     while(casa>0)
28     {
29         dig = (n/casa);
30         Send_Data(dig+'0');
31         n -= dig*casa;
32         casa /= 10;
33     }
34 }
35
36 void Send_String(char str[])
37 {
38     int i;
39     for(i=0; str[i]!='\0'; i++)
40         Send_Data(str[i]);
41 }
42
43 void Init_UART(unsigned int
44     baud_rate_choice)
45 {
46     unsigned char BRs[NUM_BAUDS] = {104,
47         52, 4, 17, 8, 7, 3};
48     unsigned char MCTLs[NUM_BAUDS] =
49     {UCBRF_0+UCBRS_1,
50     UCBRF_0+UCBRS_0,
51     UCBRF_0+UCBRS_0,
52     UCBRF_0+UCBRS_7,
53     UCBRF_0+UCBRS_6,
54     UCBRF_0+UCBRS_7,
55     UCBRF_0+UCBRS_7};
56
57     if(baud_rate_choice<NUM_BAUDS)
58     {
59         // Habilita os pinos para
60         // transmissao serial UART
61         P1SEL2 = P1SEL = RX+TX;
62         // Configura a transmissao serial
63         // UART com 8 bits de dados,
64         // sem paridade, começando pelo
65         // bit menos significativo,
66         // e com um bit de STOP
67         UCA0CTL0 = 0;
68         // Escolhe o SMCLK como clock para
69         // a UART
70         UCA0CTL1 = UCSSEL_2;
71         // Define a baud rate
72         UCA0BR0 = BRs[baud_rate_choice];
73         UCA0BR1 = 0;
74         UCA0MCTL = MCTLs[baud_rate_choice
75             ];
76     }
77 }
78
79 void Init_P1(void)
80 {
81     P1OUT |= BTN;
82     P1REN |= BTN;
83     P1DIR &= ~BTN;
84     P1IE = P1IES = BTN;
85     P1OUT &= ~LED;
86     P1DIR |= LED;
87 }

```