BRUNO SOUZA BASTOS
VIVIANE OLIVEIRA LIMA

Dispositivo Médico Inteligente para Monitoramento de Temperatura

Trabalho de Conclusão de Curso
apresentado à Escola Politécnica da
Universidade de São Paulo para obtenção
do título de Bacharel em Engenharia

Orientador:
Prof. Dr. Leopoldo Yoshioka

São Paulo
2015

BRUNO SOUZA BASTOS
VIVIANE OLIVEIRA LIMA

Dispositivo Médico Inteligente para Monitoramento de Temperatura

Trabalho de Conclusão de Curso
apresentado à Escola Politécnica da
Universidade de São Paulo para obtenção
do título de Bacharel em Engenharia

Área de Concentração:
Eletrônica embarcada

Orientador:
Prof. Dr. Leopoldo Yoshioka

São Paulo
2015

RESUMO

Em virtude do progresso da tecnologia na área de saúde e de um eventual aumento da demanda, pela população, por melhorias e novos equipamentos no ramo clínico, avanços da Engenharia em pesquisa e desenvolvimento voltados à medicina tomaram um grande impulso nas últimas décadas.

Com o crescimento do número de mulheres que trabalham fora de casa e a difusão da internet por todo o mundo, torna-se necessária e conveniente a criação de equipamentos que possam ser monitorados a longa distância, especialmente tratando-se de crianças e recém-nascidos enfermos.

Pensando nisso foi criado o Easyterm, uma solução inovadora que visa a auxiliar nos cuidados médicos dos primeiros anos após o nascimento, que são fundamentais para o desenvolvimento da criança.

Ao longo das próximas páginas, uma compreensão detalhada do projeto e do funcionamento dessa nova tecnologia será atingida. Inicialmente, serão apresentados os requisitos do projeto devidamente embasados nos conceitos médicos pertinentes. Em seguida, o processo de implementação do protótipo é explanado, e finaliza-se com uma conclusão sobre a adequação do dispositivo frente aos objetivos inicialmente determinados.

Palavras-Chave: Engenharia. Engenharia elétrica. Dispositivos médicos. Telemedicina. Termometria. Arduino.

ABSTRACT

By virtue of the progress of technology in healthcare and an occasional increase in demand, by the population, for improvements and new equipments in the clinical field, engineering advances in research and development that are focused in medicine took a great boost in the last decades.

With the increasing number of women who work outside the home and the diffusion of internet around the world, the creation of equipment that can be long distance monitored becomes necessary and convenient, especially with regard to ill children and newborn.

With that in mind, the Easyterm, an innovative solution that aims at helping medical care in the first months after birth (which are fundamental for the development of the child), was created.

Throughout the next pages, a detailed comprehension of the project and functioning of this new technology will be achieved. Initially, the project requirements will be presented, properly based on the pertinent medical concepts. Following, the implementation process of the prototype is explained, and finally, a conclusion is drawn about the device's adequation with regard to the initial objectives.

Keywords: Engineering. Electrical engineering. Medical devices. Telemedicine. Thermometry. Arduino.

SUMÁRIO

1 DEFINIÇÃO DE HIPOTERMIA E HIPERTERMIA 8

1.1 FEBRE 8

2 IDENTIFICAÇÃO E DECLARAÇÃO DAS NECESSIDADES 10

2.1 ENTREVISTA COM MÉDICA PEDIATRA 10
2.2 PROBLEMAS E NECESSIDADES IDENTIFICADAS 12

3 OBJETIVOS DO PROJETO 14

3.1 OBJETIVO 14
3.2 REQUISITOS DE MARKETING 14
3.3 PESQUISA DE LEVANTAMENTO DA SITUAÇÃO 14
3.3.1 Visão Geral 14
3.3.2 Tecnologias Relevantes 15
3.3.3 Árvore de Objetivos 17
3.4 REQUISITOS DE ENGENHARIA 19
3.5 ESPECIFICAÇÃO DE REQUISITOS 20
3.5.1 Análise avançada de requisitos 22

4 GERAÇÃO DE CONCEITOS 24

4.1 SENSOR DE TEMPERATURA (1) 26
4.2 PROCESSAMENTO DE DADOS (2 E 3) 27
4.3 BATERIA 28
4.4 TRANSMISSOR (4) 28
4.5 DECOMPOSIÇÃO FUNCIONAL 29
4.5.1 Projeto Nível 0 29
4.5.2 Projeto Nível 1 30
4.5.3 Acoplamento 31
4.5.4 Coesão 31

5 GERENCIAMENTO DE PROJETO 32

5.1 ETAPAS E ATIVIDADES 32
5.2 CUSTO 32
5.3 RISCOS 33
5.3.1 Planejamento 33
5.3.2 Identificação 33
5.3.2.1 Riscos internos 33
5.3.2.2 Riscos externos 33
5.3.3 Análise 34

6 IMPLEMENTAÇÃO 36

6.1 HARDWARE 36
6.1.1 Sensor de Temperatura 36
6.1.2 Microcontrolador 40
6.1.3 Bateria - Gerenciamento de Energia 41
6.1.3.1 Metodologia 42
6.1.4 Implementação Final 49
6.2 SOFTWARE 50
6.2.1 Plataforma Android 54

7 CONCLUSÃO **58**

8 APÊNDICES **59**

Apêndice A – Código hardware 60
Apêndice B – Código software 64

9 BIBLIOGRAFIA **72**

1 DEFINIÇÃO DE HIPOTERMIA E HIPERTERMIA

Segundo a Organização Mundial de Saúde (OMS), a faixa de normalidade de temperatura em crianças é entre 36,5 °C a 37 °C e hipotermia é classificada conforme a gravidade:

- Potencial estresse do frio (hipotermia leve): temperatura entre 36 °C e 36,4 °C.
- Hipotermia moderada: temperatura entre 32,0 °C e 35,9 °C
- Hipotermia grave: temperatura menor que 32,0 °C

Hipertermia é definida como temperatura corporal acima de 37,5 °C e mais popularmente conhecida como febre (MINISTÉRIO DA SAÚDE, 2011).

1.1 FEBRE

Em condições normais o corpo humano controla a temperatura de forma a manter os órgãos internos em torno de 37°C. Quando o organismo é sujeito a alguma condição anômala por causas internas ou externas pode ocorrer uma reação de elevação de temperatura de dois a três graus acima dos valores habituais para o indivíduo, caracterizando a ocorrência da febre.

O instrumento padrão para a medição da temperatura do corpo é o termômetro clínico de vidro com mercúrio. As formas de medições mais comuns são a colocação do bulbo do termômetro, por um período de cinco minutos, na parte interior dos seguintes locais:

- boca
- reto
- dobras das axilas

Embora o método mais preciso seja o retal, a forma mais utilizada é a axilar. A medição temperatura através da axila possui alguns inconvenientes: O primeiro é a pouca precisão, o segundo é a dificuldade de manter o termômetro em contato com o corpo durante um período mínimo quando se tratam de crianças (principalmente bebês), que normalmente resistem à colocação de objetos estranhos em seus corpos.

Adicionalmente, deve-se considerar o termômetro é constituído de vidro, um material frágil, podendo se quebrar com certa facilidade, liberando o mercúrio que é uma substância tóxica à saúde, pois é volátil em temperatura ambiente.

2 IDENTIFICAÇÃO E DECLARAÇÃO DAS NECESSIDADES

2.1 ENTREVISTA COM MÉDICA PEDIATRA

Inicialmente buscou-se compreender melhor a respeito da febre. Com este intuito o grupo agendou uma entrevista com a Dra. Cristina Ryoka Miyao Yoshioka, médica pediatra e vice-chefe da Enfermaria Pediátrica do Hospital Universitário da Universidade de São Paulo (HU-USP).

A seguir apresentamos um resumo dos principais aspectos a serem considerados em relação à febre em crianças:

- De acordo com a Dra. Cristina, os tipos de termômetro para medição de temperatura em hospitais são os termômetros: axilar, retal, oral, timpânico e o adesivo na artéria temporal (usado em casa). O mais utilizado no Hospital Universitário é o termômetro digital axilar.
- Quanto à confiabilidade ela enfatizou que, na medição da temperatura, o objetivo é ler a temperatura central que se encontra na artéria pulmonar, entretanto, não é possível medir essa temperatura. A melhor alternativa para essa medição seria o termômetro retal, porém, existe o desconforto do paciente e a dificuldade da mensuração da temperatura nessa região.
- Além disso, a precisão da medição depende do ambiente e para isso o termômetro ideal seria o que não sofresse interferência da temperatura externa como, por exemplo, no termômetro oral (onde o paciente está cansado para respirar e pode interferir na medida). Na pele, na fase inicial da febre, o vaso fica constrito ocasionando diminuição da temperatura não permitindo a identificação da febre em seu início.
- Segundo a pediatra, a partir de 38°C no termômetro retal, 37,8°C (ou 37,6°C dependendo do valor adotado pelo médico) no axilar é considerado que o paciente encontra-se em estado febril, podendo dar início ao uso de medicação. Contudo, a febre pode causar convulsão febril na faixa etária de 5 meses a 5 anos de idade, sendo muito comum crianças com esse quadro no pronto socorro.

- Adicionalmente, a febre é popularmente ligada à infecção, mas doenças reumatológicas, ancológicas, e tudo que dá processo inflamatório pode causar febre, sendo o mais frequente a infecção.
- Ainda segundo a Dra. Cristina, existem alguns tipos de doença onde a criança tem febre recorrente, como a tuberculose, onde a febre é sempre no final da tarde. Outro tipo de doença que apresenta febre periódica é a malária, onde os quadros febris ocorrem a cada 3 ou 4 dias. Além disso, ela ressalta que alguns pacientes que estão internados e tomam antibióticos apresentam febre contínua, nesses casos a febre não é gerada pela infecção, mas sim por conta de reações medicamentosas.
- Durante a entrevista perguntamos à doutora se utilizaria o dispositivo proposto neste projeto no Hospital Universitário. Ela disse afirmativamente que poderia sim utilizar o dispositivo dependendo do caso. Disse que o dispositivo seria de grande utilidade para acompanhamento do pico e da curva de temperatura. Ela enfatizou que no Hospital Universitário existe um protocolo para crianças com febre sem sinais (criança apenas com sintoma de febre) no qual, além de alguns exames para descobrir se existem riscos de infecção é necessária também uma monitoração de 24 horas do paciente onde ele teria de permanecer na enfermaria.
- Destacou que em alguns casos específicos são necessários rápidos diagnósticos e investigações da doença, como estão listados a seguir:

a) **Faixa etária de risco**: recém-nascido (investigação obrigatória), dois primeiros meses de vida (investigação recomendada) e terceiro mês de vida (desde que a impressão geral seja satisfatória, é aceitável manter em observação atenta). A partir dos três meses, é válida a observação em ambulatório com acesso (telefônico, retorno) facilmente disponível e programado.

b) **Febre de mais de 39,4 °C**: especialmente se acompanhada de tremores de frio, sugere infecção bacteriana/bacteriemia. Suspeitar também em casos de temperatura abaixo de 36 °C em criança abatida.

c) **Estado infeccioso/toxêmico acentuado**: má impressão geral, aspecto abatido, inapetência, irritabilidade alternada com sonolência, letargia, apatia, fácies de sofrimento, choro inconsolável ou choramingas, gemência (sinal de alarme) e a disposição da criança.

d) **Duração da febre maior que três dias (mais de 72 horas):** quando a duração da febre supera esse período, contados a partir do momento presumido do início da febre, é necessário monitorar com a maior precisão possível

2.2 PROBLEMAS E NECESSIDADES IDENTIFICADAS

Analisando-se os métodos e os dispositivos existentes atualmente para a medição de temperatura corporal, o grupo identificou os seguintes problemas com relação ao uso do termômetro clínico de vidro:

- é mecanicamente frágil;
- não possui boa precisão (exatidão), principalmente quando mede-se nas axilas;
- causa incômodo nas crianças, principalmente em bebês;
- não é adequado para se fazer um acompanhamento contínuo, principalmente em casos críticos (longa duração ou de picos);
- entre outras.

Tendo-se em vista os problemas com os termômetros clínicos atualmente utilizados, o grupo identificou as seguintes necessidades:

- detectar pequenas variações de temperatura–principalmente em bebês, pois na faixa etária entre 0 a 3 meses de vida encontram-se os maiores riscos associados à febre;
- acompanhar picos de temperatura;
- acompanhar febres prolongadas;

- detectar a condição de febre;
- detectar a ocorrência de uma situação crítica;
- não causar incômodos às crianças – principalmente bebês;
- gerar avisos ou alarmes;
- registrar o histórico da febre;
- entre outras.

3 OBJETIVOS DO PROJETO

3.1 OBJETIVO

Desenvolver e *construir* um dispositivo de medição de temperatura *especialmente para crianças*, que possa *monitorar a temperatura* durante 24 horas, mostrando uma curva de temperatura como *dados finais*, permitindo que o usuário saiba se o paciente apresenta febre.

3.2 REQUISITOS DE MARKETING

Os Requisitos de Marketing são os seguintes:

1. Detectar febre em bebês e crianças
2. Alertar o médico imediatamente quando necessário
3. Alertar a hora certa de dar medicação ao paciente
4. Método não abrasivo
5. Conforto do paciente
6. Fácil de usar
7. Possível utilização em dispositivos móveis
8. Método de precisão
9. Método de acuidade

3.3 PESQUISA DE LEVANTAMENTO DA SITUAÇÃO

3.3.1 Visão Geral

Esse projeto visa a construção de um termômetro que consiga aliar conforto, precisão, confiabilidade e comodidade ao usuário no monitoramento constante de temperatura do paciente.

- **Conforto:** para que haja conforto é necessária a criação de um produto compacto, leve e sem fios, características presentes nos sensores eletrônicos com capacidade de transmissão wireless. Para maior precisão

e confiabilidade é primordial que o sistema sofra o mínimo de interferências externas, necessitando assim de um circuito projetado especificamente para o uso em questão, com baixo consumo de energia (uso de bateria) e transmissão de dados confiável (dados emitidos pelo sensor sejam corretamente interpretados pelo receptor).

- **Comodidade**: a comodidade é uma condição inerente ao cotidiano do usuário. Menos intervenções na rotina do cliente proporcionam maior comodidade. Segundo dados divulgados pela analista Mary Meeker em 2013 o número de smartphones no Brasil era de 70 milhões e segundo o PNAD mais da metade da população brasileira possui acesso à internet, portanto, internet e smartphones já são uma realidade na rotina dos brasileiros. Então uma maneira usual de garantir maior conveniência ao usuário é associar o produto a essas duas plataformas citadas, para isso pode-se vincular os dados obtidos pelo sensor a uma plataforma embarcada que poderá disponibilizar os resultados na rede ou comunicar-se diretamente com um smartphone.

3.3.2 Tecnologias Relevantes

Os tipos de termômetro existentes atualmente são os seguintes:

- Termômetro Oral
- Termômetro artéria temporal
- Termômetro axilar
- Termômetro de ouvido
- Termômetro anal

O termômetro anal é considerado um método abrasivo e normalmente mais utilizado nos EUA, portanto, serão considerados apenas os métodos não abrasivos (itens 1 a 4).

A tabela abaixo mostra alguns fatores negativos de cada tecnologia utilizada atualmente e maneiras de corrigí-la:

Tabela 1: Fatores que afetam a acurácia e precisão de variados métodos de medição da temperatura (BRIDGES; THOMAS, 2009).

Método	Fatores	Como corrigir
Oral	• Tuboendotraqueal • Posição incorreta da sonda • Modo • Ao beber líquidos quentes ou frios • Mucoseouestomatite oral	• Posicionar a sonda no lado oposto ao tubo endotraqueal pois o tubo pode aumentar a temperatura medida • Posicionar a sonda na parte posterior sublingual - não na frente da boca • Definir o termômetro no modo "core" • Necessário espera de 30min após o consumo de líquidos • Contraindicação geral - informalmente reportado para aumentar a temperatura
Artéria Temporal	• Diaforese • Medicações vasopressoras • Lente suja • Fluxo de ar pelo rosto	• Manter o botão de SCAN pressionado e uma medição da temperatura adicional é tirada atrás da orelha • Limitada pesquisa • Necessário limpeza a cada 2 semanas • Limitada pesquisa
Timpânico	• Excesso de cerume • Ouvido encostado no travesseiro • Repetidas temperaturas • Usando o ouvido oposto • Lentes sujas	• A temperatura diminui entre 0,13°C a 0,3°C. Necessário retirar o cerume • Não medir temperatura no ouvido dos pacientes que estavam deitados • Esperar pelo menos 2 min entre as medições repetidas para impedir resfriamento do tecido após o contato com a sonda • Não ficar por cima do corpo do paciente para usar o ouvido contra lateral • Limpar de acordo com as especificações do produto
Axilar	• Posição incorreta da sonda	• Levantar o braço do paciente até que toda a axila possa ser vista • Posicionara sonda o mais alto

possível na axila sem deixá-lo entrar em contato com a pele do paciente
- Posicionar o braço do paciente confortavelmente do lado dele e segure-o nessa posição durante o ciclo de medição

Através de análise da acurácia e das deficiências de cada método de medição, e também considerando a opinião da Dra. Cristina e dos requisitos de marketing voltados ao paciente, foi decidido que o método de medição que melhor se encaixa ao projeto será o axilar, onde será utilizado um sistema que permanecerá em contato com a superfície da criança próximo à axila.

3.3.3 Árvore de Objetivos

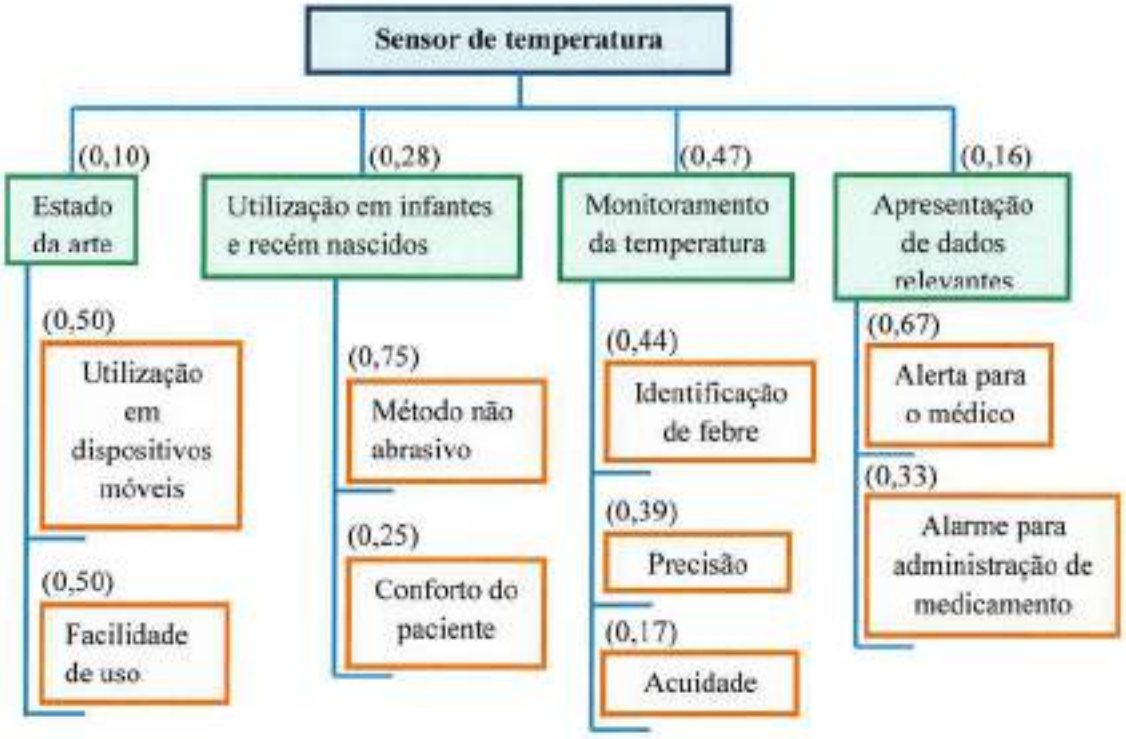

Figura 1: Árvore de objetivos

Tabela 2: Ponderação dos pesos das necessidades do cliente

	Estado da arte	Utilização em infantes e recém nascidos	Monitoramento da temperatura	Apresentação de dados relevantes	Média Geométrica	Peso
Estado da arte	1,00	0,33	0,25	0,50	0,45	0,10
Utilização em infantes e recém nascidos	3,00	1,00	0,50	2,00	1,32	0,28
Monitoramento da temperatura	4,00	2,00	1,00	3,00	2,21	0,47
Apresentação de dados relevantes	2,00	0,50	0,33	1,00	0,76	0,16

Tabela 3: Ponderação dos pesos das necessidades de estado da arte

	Utilização em dispositivos móveis	Facilidade de uso	Média Geométrica	Peso
Utilização em dispositivos móveis	1,00	1,00	1,00	0,50
Facilidade de uso	1,00	1,00	1,00	0,50

Tabela 4: Ponderação dos pesos das necessidades do paciente

	Método não abrasivo	Conforto do paciente	Média Geométrica	Peso
Método não abrasivo	1,00	3,00	1,73	0,75
Conforto do paciente	0,33	1,00	0,57	0,25

Tabela 5: Ponderação dos pesos das necessidades de precisão

	Identificação de febre	Precisão	Acuidade	Média Geométrica	Peso
Identificação de febre	1,00	1,00	3,00	1,44	0,44
Precisão	1,00	1,00	2,00	1,26	0,39
Acuidade	0,33	0,50	1,00	0,55	0,17

Tabela 6: Ponderação dos pesos das necessidades para o médico

	Alerta para o médico	Alarme para administração de medicamento	Média Geométrica	Peso
Alerta para o médico	1,00	2,00	1,41	0,67
Alarme para administração de medicamento	0,50	1,00	0,71	0,33

3.4 REQUISITOS DE ENGENHARIA

Os requisitos de Engenharia foram baseados no artigo (LUNA, 2008) e estão relacionados abaixo:

- Usabilidade

  A interface do usuário será fácil de entender.
- Confiabilidade

  A leitura do sensor deverá ter precisão de ±0,5°C.
- Desempenho

  O sensor de temperatura deverá funcionará por um período de 24horas.
- Segurança

  O sistema deve proteger os dados do paciente e divulgá-los apenas ao seu médico, quando o usuário permitir, seguindo os padrões éticos da Medicina*
- Implementação

  O conjunto do sistema, englobando: sensor, bateria, microcontrolador e transmissor; será leve e pequeno, não atrapalhando o paciente.
- Operações

  A interface do usuário funcionará inicialmente em computadores que suportem o Windows.

  O sistema operará na faixa de temperatura de 32,0°C a 42,0°C .

  A bateria será recarregável.
- Restrições de hardware e software

  O transmissor deverá ser capaz de se comunicar com o computador via Wireless ANT.

  Após a transmissão de dados para computador, eles serão tratados por Matlab e então enviados para a interface do usuário.

*Legais

Segundo (SANT'ANNA; CARDOSO; SANT'ANNA, 2005), no Brasil ainda não existe uma ampla jurisprudência quanto a cuidados por telemedicina, portanto, consideraremos os fatos éticos relativos ao paciente.

*Ética (SANT'ANNA; CARDOSO; SANT'ANNA, 2005)

Privacidade e confidencialidade: o paciente (no caso do nosso produto, o responsável pelo paciente, pois o mesmo é menor de idade) terá controle de quem terá acesso à informação sobre à sua saúde.

Consentimento: o usuário deve ter consentimento informado.

Responsabilidade do paciente: o paciente será o responsável pela coleta e transmissão de dados ao médico.

3.5 ESPECIFICAÇÃO DE REQUISITOS

Tabela 7: Especificação de requisitos

Requisitos Marketing	Requisitos de Engenharia	Justificativa
1,6	A interface do usuário será de fácil compreensão para o usuário.	O sistema deve permitir uma interface simples com o usuário, imprimindo na tela uma tabela e um gráfico com a temperatura do paciente.
1,8,9	A leitura do sensor deverá ter precisão de ±0,5°C.	Baseado nas tecnologias existentes, é possível chegar à essa precisão.
1,8	O sensor de temperatura deverá funcionar por um periodo de 24horas.	Baseado em tecnologias equivalentes já existentes.

2,3,5	O sistema deve proteger os dados do paciente e divulgá-los apenas ao seu médico, quando o usuário permitir, seguindo os padrões éticos da Medicina.	Segundo as leis éticas de Medicina.
4,5	O conjunto do sistema será leve e pequeno, não atrapalhando o paciente.	O conjunto final apresentará um tamanho confortável ao paciente (10cmx10cm).
6	A interface do usuário funcionará inicialmente em computadores que suportem o Windows.	Sabendo-se que a maioria da população brasileira utiliza ambiente Windows.
1	O sistema operará na faixa de temperatura 32,0°C - 42,0°C .	Baseado no artigo com sensor semelhante ao utilizado no projeto.
5	A bateria deverá possuir um tamanho adequado ao projeto e produzir corrente suficiente para alimentação do hardware.	O dispositivo utilizará uma bateria de no mínimo 1,2V.
7	O transmissor deverá ser capaz de se comunicar com o computador via Wireless.	Utilizando um módulo de comunicação Wireless.
8	Após a transmissão de dados para computador, eles serão tratados por outro software por comunicação Wireless.	Utilização de um software capaz de gerar executável para fácil utilização do usuário.

3.5.1 Análise avançada de requisitos

Tabela 8: Matriz de compromissos Engenharia - Engenharia

		Interface	Precisão	Tempo	Privacidade	Tamanho	Operador	Bateria	Transmissor
		+	-	+	+	-	+	-	-
Interface	+				↑↑		↑↑		
Precisão	-			↓		↓		↓	
Tempo	+					↓		↑↑	↑
Privacidade	+						↑		
Tamanho	-							↓↓	
Operador	+								↑
Bateria	-								↑
Transmissor	-								

Tabela 9: Matriz de compromissos Engenharia - Marketing

		Interface	Precisão	Tempo	Privacidade	Tamanho	Operador	Bateria	Transmissor
		+	-	+	+	-	+	-	-
1) Detectar Febre	+	↑	↑↑	↑			↑	↑	
2) Alerta	+	↑					↑		
3) Método não abrasivo	+					↓↓		↓	
5) Conforto	+					↓		↓	
6) Facilidade	+	↑↑		↑	↑		↑		
7) Utilização em smartphones	+	↑		↑	↑		↑		
8) Precisão	-	↑	↑↑						
9) Acuidade	+	↑↑	↑	↑					

Tabela 10: Benchmark Competitivo

Projeto	Termômetro Axilar/Oral Modelo MC-343F*	Termômetro Timpânico Modelo 29838**	Termômetro Artéria Temporal Modelo TAT-5000***

Fonte de Alimentação	1,5 V	3,0 V	9,0 V	5,0 V
Faixa de medição	32,0°C a 42,0°C	34,0°C a 42,2°C	16,0°C a 43,0°C	32,0°C a 42,0°C
Precisão	±0,2°C	±0,2°C	±0,1°C	±0,5°C
Bateria	Pilha alcalina não recarregável	Bateria de lítio (CR2032) não recarregável	Pilha alcalina não recarregável	Recarregável
Transmissor	-	-	-	Wireless ANT
	*(OMRON HEALTHCARE BRASIL, 2011)	**(INCOTERM)	***(EXERGEN CORPORATION)	

4 GERAÇÃO DE CONCEITOS

O projeto do dispositivo médico requer duas unidades principais, que foram denominadas: Módulo Adesivo e Módulo Tratar Dados. As figuras 2 e 3 abaixo explicitam as funções, que irão suprir as necessidades do projeto, de cada módulo:

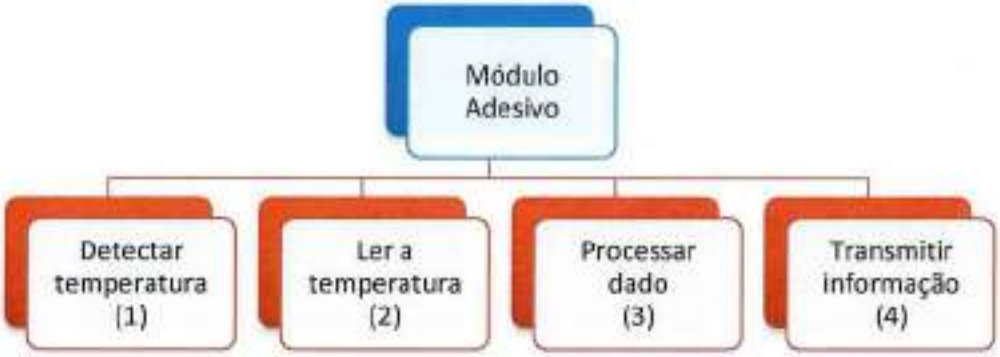

Figura 2: Módulo Adesivo - sistema que ficará acoplado com o paciente.

Figura 3: Módulo Tratar Dados - sistema que receberá os dados vindo do Módulo Adesivo e os passará para a interface do usuário.

Inicialmente, focaremos na implementação do Módulo Adesivo, que será o passo mais importante para a implementação do projeto. Para atender as especificações do módulo adesivo, os seguintes componentes são necessários:

- Sensor de temperatura;
- Microcontrolador;
- Bateria;
- Transmissor

Portanto, foram consideradas as opções detalhadas na tabela 11 abaixo:

Tabela 11: Tabela de conceitos do sistema

Sensor Térmico	Processamento de Dados	Bateria	Transmissor
Sensor lm35d	Microcontolador Arduino	Não recarregável	Bluetooth
Transistor	Microprocessador Raspberry	Recarregável	ANT Wireless Network Protocol

Inicialmente foi considerado um circuito de microeletrônica que já tinha sido utilizado por professores da Unifesp, com a mesma intenção que o nosso projeto (sensor de temperatura) . Entretanto, por falta de recursos para impressão, descartamos essa possibilidade.

Além disso, termopar e RTDs (termorresistores) também foram considerados, entretanto, esses sensores geralmente são usados para coletar altas temperaturas e em um intervalo muito grande não possuindo uma boa precisão ao ser utilizado no nosso projeto, portanto, foram desconsiderados.

Ademais, após uma extensa pesquisa e algumas reuniões com o orientador, foram escolhidas as dadas opções de componentes como as que mais se adaptam para o projeto.

Tabela 12: Ponderação dos pesos das especificações de engenharia

	Precisão	Tempo	Tamanho	Bateria	Transmissor	Média geométrica	Peso
Precisão	1.0	1.0	2.0	1.0	2.0	1.3	0.25
Tempo	1.0	1.0	2.0	1.0	0.5	1.0	0.19
Tamanho	0.5	0.5	1.0	0.3	0.5	0.5	0.10
Bateria	1.0	1.0	3.0	1.0	2.0	1.4	0.27
Transmissor	0.5	2.0	2.0	0.5	1.0	1.0	0.19

4.1 SENSOR DE TEMPERATURA (1)

O primeiro passo do projeto é escolher um sensor de temperatura, capaz de detectar a temperatura. As duas opções foram o sensor LM35 (CI) e um transistor.

O sensor Modelo LM35 possui um circuito simples, tendo uma saída com baixa impedância, tensão linear, faixa de temperatura de -55°C a 150°C e calibração inerente precisa, necessitando apenas de um amplificador de sinal e uma interface que realize a leitura.

Adicionalmente, a temperatura do sensor estará 0,01°C abaixo da temperatura da superfície que estará colado (pressupondo que a superfície esteja na temperatura ambiente).

O transistor necessita de uma baixa tensão de entrada, porém, antes de exercer a sua função de sensor, ele deve entrar em equilíbrio térmico com a superfície, tendo, assim, uma velocidade máxima de resposta e um certo atraso na medição da temperatura.

Por este motivo, e pela análise de Pugh abaixo (Tabela 13), foi decidido que a melhor opção seria o uso de um Circuito Integrado (CI) como sensor de temperatura, no caso escolhido, o sensor LM35.

Tabela 13: Análise de Pugh da opções do sensor de temperatura

	Opção 1: Sensor lm35d	Opção 2: Transistor
Precisão	1	-1
Tempo	1	-1
Tamanho	1	1

Bateria	-1	1
Transmissor	0	0
Net Score	2	0
Continuar?	Sim	Não

4.2 PROCESSAMENTO DE DADOS (2 E 3)

Após o sensor detectar temperatura, precisaremos transformar esse sinal analógico para digital, para então lermos a temperatura e processarmos os dados. Para isso, escolhemos duas opções, o microcontrolador Arduino e o microprocessador Raspberry para a análise.

A grande vantagem do Raspberry em comparação com o Arduino, é o fato de que ao invés de escrever uma programação para rodar o hardware corretamente, o Raspberry conecta-se ao computador, permitindo rodar o programa direto do sistema operador.

Entretanto, a performance do Raspberry se adéqua a dispositivos de vídeo e mídia, enquanto a do Arduino a motores e sensores. Na tabela 14 abaixo temos uma descrição das principais funcionalidades de ambos:

Tabela 14: Tecnologias do Arduino e Raspberry.

	Arduino Uno	**Raspberry Pi**
Processador	AVR ATmega328p	Broadcom ARM1176JZF-S
Frequência	16MHz	700MHz
Registrador	8 bit	32 bit
RAM	2k	512MB
GPIO	20	8
Corrente máxima	40 mA	5-10 mA
Potência	175 mW	700 mW

Ademais, na análise de Pugh (Tabela 15), o Arduino apresenta uma melhor pontuação final que o Raspberry. Portanto, para esse projeto em específico, o microcontrolador Arduino UNO será uma melhor opção pois iremos utilizar um sensor de temperatura.

Tabela 15: Análise de Pugh das opções para processamento de dados

		Opção 1: Arduino	Opção 2: Raspberry
Precisão	0.25	0.25	0.25
Tempo	0.19	0.19	0.19
Tamanho	0.10	0.10	-0.10
Bateria	0.27	0.27	-0.27
Transmissor	0.19	0.19	0.19
Net Score		1	0.26
Continuar?		Sim	Não

4.3 BATERIA

Para a bateria, temos a opção de utilizar uma bateria descartável (não recarregável) e uma recarregável. Para esse requisito, a diferença principal entre ambas é o tempo útil da bateria (a bateria recarregável dura mais) e o fato de que pilhas descartáveis de 1,2V, por exemplo, não permanecem no pico de voltagem após um período de tempo. Portanto, pela análise de Pugh (Tabela 16), a melhor opção será uma bateria recarregável.

Tabela 16: Análise de Pugh das opções da bateria

		Opção 1: Descartável	Opção 2: Recarregável
Precisão	0.25	0.00	0.00
Tempo	0.19	-0.19	0.19
Tamanho	0.10	0.10	0.10
Bateria	0.27	-0.27	0.27
Transmissor	0.19	0.00	0.00
Net Score		-0.36	0.56
Continuar?		Sim	Não

4.4 TRANSMISSOR (4)

Após o processamento dos dados, será necessário transmitir os dados para um plataforma (computador) que possa interpretar os dados e imprimir os relatórios finais na interface para o usuário. Para tanto, temos a opção de transmissão Bluetooth e via Wireless Network Protocol ANT.

A opção do serial Bluetooth de baixa energia será uma forma mais complexa de protocolo, porém, o dispositivo ANT, que é conectado via USB, é uma forma mais segura de transmissão e mais garantida, dado que o Bluetooth pode vir a falhar. Na tabela 17 abaixo estão apresentadas as diferenças de tecnologia entre os dois componentes:

Tabela 17: Tecnologias do ANT e Bluetooth

Tecnologia	ANT	Bluetooth Low Energy
Frequência	2.4 to 2.483 GHz	2.4 to 2.483 Ghz
Topologia suportada	P2P, estrela, árvore, malha	P2P, estrela
Modulação	GFSK	GFSK
Canal	1 MHz	2 Mhz
Protocolo	Simples	Mais complexa
Data	1 Mbit/s	1 Mbit/s
Range	50m	50m
Segurança	64 bits	128 bits

Portanto, a escolha inicial para teste de transmissão será via Wireless Network Protocol ANT (o modelo será 2.4GHz NRF24AP2-USB ANT + Wireless Communication Module AP3000).

4.5 DECOMPOSIÇÃO FUNCIONAL

4.5.1 Projeto Nivel 0

Tabela 18: Requisitos funcionais do Módulo Adesivo

Módulo	Adesivo
Entradas	Temperatura Corporal: 32,0°C a 42,0°C. Fonte de alimentação baixa: 5,0 V.
Saídas	Transmissão dos dados da temperatura processada para um computador.
Funcionalidade	Ler a temperatura, tratar os dados e transmiti-la para outra plataforma (computador).

Figura 4: Diagrama Funcional Nível 0

4.5.2 Projeto Nível 1

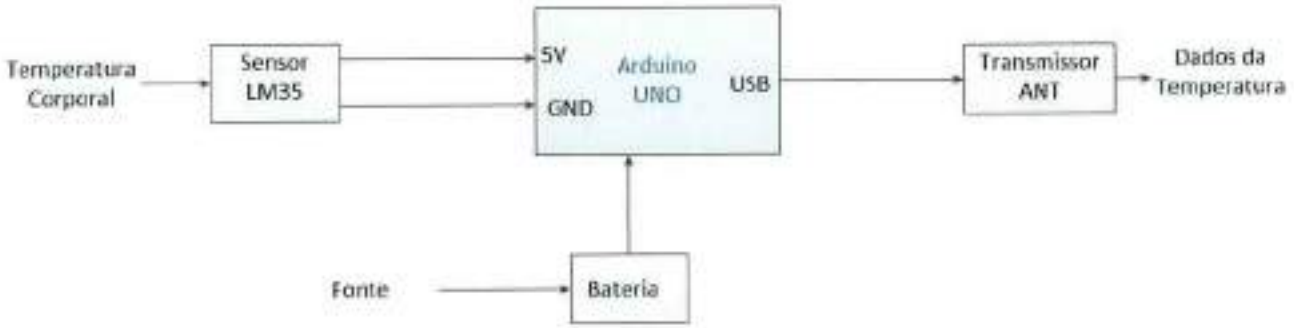

Figura 5: Diagrama Funcional Nível 1

Tabela 19: Requisitos funcionais do Módulo Sensor de Temperatura

Módulo	Sensor
Entradas	Temperatura Corporal: 32,0°C a 42,0°C.
Saídas	Voltagem linearmente proporcional à temperatura ($V = \frac{10mV}{°C} T$).
Funcionalidade	Ler a temperatura e produzir na saída uma voltagem proporcional, com acurácia de ±0,5°C.

Tabela 20: Requisitos funcionais do Módulo Bateria

Módulo	Bateria
Entradas	Alimentação de 5,0V.
Saídas	Voltagem de 5,0V.
Funcionalidade	Responsável pela fonte de energia do sistema.

Tabela 21: Requisitos funcionais do Módulo Arduino

Módulo	Arduino
Entradas	Voltagem provinda do sensor Voltagem vinda da bateria.
Saídas	Dados convertidos para temperatura.
Funcionalidade	Fazer a conversão de Volts para °C e tratar esses dados.

Tabela 22: Requisitos funcionais do Módulo Transmissor

Módulo	Transmissor ATN
Entradas	Dados da temperatura.
Saídas	Dados da temperatura.
Funcionalidade	Fazer a transmissão dos dados coletados para outra plataforma.

4.5.3 Acoplamento

Desconsiderando-se o módulo Bateria pois não indicará erros, teremos 3 módulos dentro do Módulo Adesivo, possuindo um total de conexões:

$$Cmax = \frac{n(n-1)}{2} = 3$$

O acoplamento do sistema não é alto, e cada módulo está correlacionado com o outro.

4.5.4 Coesão

É possível testar os módulos de maneira independente, se conectarmos o Arduino no computador e programarmos para fazer um teste de cada vez. Portanto, o sistema possui boa coesão.

5 GERENCIAMENTO DE PROJETO

Para o primeiro semestre a execução do projeto estará focada na etapa 1 representada na figura 6, pode-se notar que o objetivo para este estágio é a construção do módulo do sensor de temperatura com os componentes necessários para a leitura correta dos dados. O cronograma para esta fase do trabalho encontra-se na figura 7 com detalhamento dos dias e dos vínculos das tarefas a serem realizadas.

5.1 ETAPAS E ATIVIDADES

Figura 6: EAP

5.2 CUSTO

O custo do projeto (Figura 2) encontra-se atualizado com os componentes finais utilizados, as mudanças feitas ao longo da implementação estão detalhadas no item 6.

Material	Descrição	Preço (unidade)	Quantidade	Preço Total
Microcontrolador	Arduino Pro Mini Atmega328p	$ 1.50	1	$ 1.50
Sensor de temperatura	DS18B20 TO-92 DALLAS	$ 5.99	1	$ 5.99
Módulo Bluetooth	HC 05 RF sem fio Bluetooth Transceiver escravo módulo RS232	$ 2.92	1	$ 2.92
Bateria	Bateria eletrônica de lítio 3 V / 220 mAh CR2032	$ 0.29	3	$ 0.88
Chave on/off		$ 0.54	1	$ 0.54
Resistor		$ 0.02	3	$ 0.06
Outros	Placa de cobre, etc	$ 1.00	1	$ 1.00
Total*		$ 12.26	11	$ 12.89

Figura 7: Orçamento total do Hardware.

*Preços baseados no site AliExpress.

*Valor em dólares americanos.

5.3 RISCOS

5.3.1 Planejamento

Risco é a chance que um evento indesejável pode ocorrer e suas possíveis consequências (NASCIMENTO, 2003). Os riscos se dividem em:

- Riscos de Projeto: riscos diretamente ligados ao projeto. Os fatores que estão intimamente ligados a estes riscos são: requisitos, pessoal, recursos, cliente, orçamento e cronograma.
- Riscos Técnicos: riscos relacionados à qualidade do produto a ser desenvolvido. Eles envolvem problemas de: design, implementação, interface, verificações e manutenção.
- Riscos de negócios: riscos relacionados à viabilidade do projeto. Entre os riscos de negócios estão: produção de um produto excelente, mas que não tem demanda; troca de gerente do projeto; produção de um produto que não se encaixa no mercado.

5.3.2 Identificação

Nesse projeto, a ênfase é dada aos riscos internos e externos.

5.3.2.1 Riscos internos

O risco interno é controlado pelo líder, que pode reduzí-lo mediante ações diretas. Riscos internos constituem uma parte das limitações estabelecidas para o projeto por meio da criação de metas. Dentre os riscos internos, estão:

- Custo
- Prazo
- Gerenciais
- Perda de potencial
- Fluxo de caixa

5.3.2.2 Riscos externos

O risco externo encontra-se fora do controle dos líderes de projeto, como, por exemplo, as interfaces do projeto desconhecidas e cujas definições são feitas por terceiros. Dentre os riscos externos previsíveis para esse projeto, pode-se citar:

- Taxa de câmbio: risco associado a operações internacionais em um mundo no qual o valor relativo das moedas varia.
- Inflação: risco macroeconômico da perda de renda disponível de pessoas físicas e jurídicas, acarretando prejuízos ao fluxo de caixa e recusa do projeto.
- Impactos sociais: relacionado aos efeitos sociais, econômicos, culturais, sobre pessoas, grupos de pessoas ou comunidades.
- Impactos ambientais: impactam diretamente no aumento dos custos do projeto.
- Riscos operacionais: referem-se às perdas potenciais, má administração, controles defeituosos ou falha humana. Além disso, fraudes e riscos de modelo (imperfeições).
- Riscos de Mercado:geram resultados adversos por conta da instabilidade em taxas de juros, taxas de câmbio, preços de ações, etc.

Além disso, considera-se também os riscos externos imprevisíveis, como:

- Medidas reguladoras
- Efeitos colaterais

5.3.3 Análise

Dentro do escopo do projeto serão considerados os riscos descritos na Tabela 23 abaixo:

Tabela 23: Descrição dos riscos do projeto

Riscos internos	Riscos externos previsíveis	Riscos externos não previsíveis
Variação nos custos causadas pelo fornecedor	Variação dos custos por conta de variações do câmbio e/ou inflação	Produto causa efeitos colaterais (alergia, coceira, queimaduras, etc.)
Atrasos no cronograma devido a dificuldades de integração dos sistemas	Crescimento na utilização de materiais que impactam o meio ambiente	Reprovação do governo para distribuição comercial

Aprovação de patentes	Problemas de qualidade de produção
Alteração do escopo inicial	Novos regimes de impostos
Erro na análise do sistema	Falta de aceitação pela sociedade

Com isso, pode-se construir a matriz de risco do projeto (Tabela 24):

Tabela 24: Matriz de Riscos

Probabilidade		Baixo	Médio	Alto	Muito alto
Frequente	1			Problemas de qualidade de produção	
Provável	0.8			Atrasos no cronograma devido à dificuldades de integração dos sistemas	Variação nos custos causadas pelo fornecedor
Alto	0.6	Erro na análise do sistema	Novos regimes de impostos	Variação dos custos por conta de variações do câmbio e/ou inflação	Produto causa efeitos colaterais (alergia, coceira, queimaduras, etc.)
Médio	0.4	Falta de aceitação pela sociedade	Reprovação do governo para distribuição comercial		
Baixo	0.2	Utilização de materiais que impactam o meio ambiente		Alteração do escopo inicial	Aprovação de patentes

Impacto

6 IMPLEMENTAÇÃO

6.1 HARDWARE

6.1.1 Sensor de Temperatura

Para a primeira fase do projeto foram feitos, inicialmente, testes com o sensor LM35d acoplado ao Arduino UNO. O sensor LM35 possui 3 pinos (Figura 9), onde cada pino (da esquerda para direita, olhando de frente para o lado plano do CI) representa (TEXAS INSTRUMENTS, 2015):

- Pino 1: Alimentação (Entre 2,7 e 5,5V)
- Pino 2: Saída analógica (Valor analógico a ser interpretado pelo Arduino)
- Pino 3: Terra (GND)

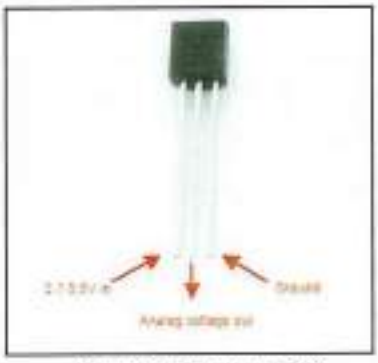

Figura 8: Sensor LM35d

Para os testes, os componentes foram ligados em uma protoboard, de acordo com a Figura 9, e o Arduino foi conectado ao computador via USB para envio do código de programação e para fonte de alimentação do Arduino.

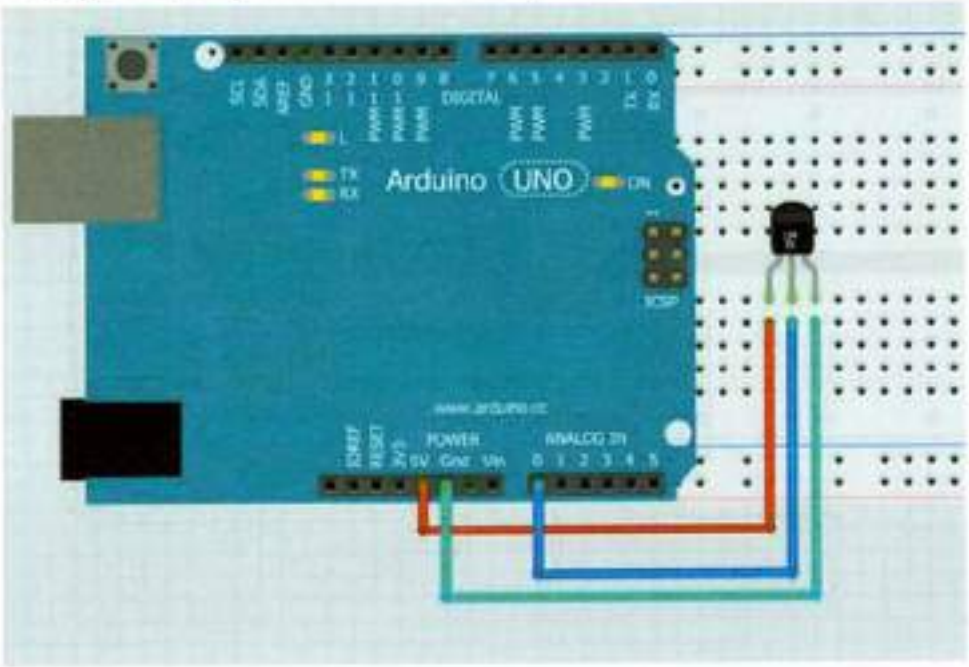

Figura 9: Circuito para teste do sensor LM35d

Para verificação de erros de leitura do circuito, os testes foram feitos com 2 sensores ao mesmo tempo comparando-se a leitura final (impressa no Arduino 1.6.4) de cada um deles.

Os 2 sensores leram a temperatura ambiente na hora testada e foi observado que existia uma grande discrepância (considerando a precisão do projeto em questão) entre as medidas feitas entre eles, além do fato de não atingirem uma temperatura constante. Ao analisar a curva de erro por temperatura (Figura 10) no datasheet do LM35, percebe-se que o erro (em temperaturas diferentes de 25°C) será maior que 0,5°C, o que não se encaixa nos requisitos do projeto.

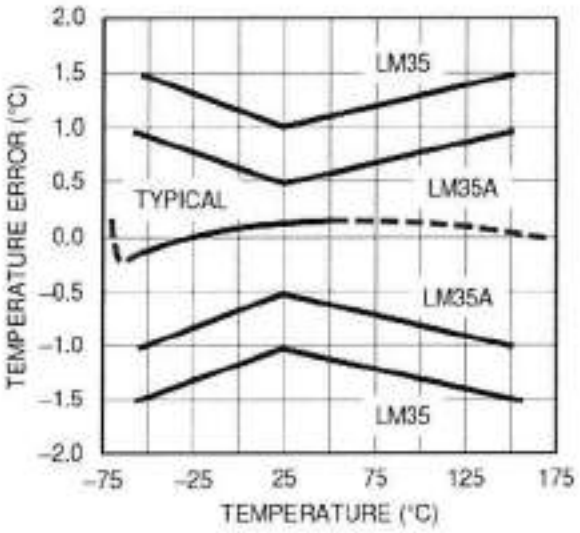

Figura 10: Curva de erro do sensor LM35 (TEXAS INSTRUMENTS, 2015)

Portanto, foi necessária a mudança do sensor de temperatura para obter uma melhor precisão nas medidas. O sensor escolhido foi o DS18B20, da Dallas Semiconductor. Na tabela 24 abaixo, foi feita uma comparação entre as principais características dos dois sensores (TEXAS INSTRUMENTS, 2015) e(DALLAS SEMICONDUCTOR):

Tabela 25: Caracteriticas dos sensores LM35 e DS18B20

LM35	DS18B20
Diretamente calibrado em graus Celsius	Interface 1-Wire que exige apenas uma porta para comunicação
Fator de escala linear +10mV/°C	Não necessita de componentes externos

Erro de precisão de 0,5°C (para temperatura de 25C)	Opera entre 3,0V e 5,0V
Faixa de leitura de -55°C a 150°C	Não necessita de alimentação na função standby
Pode ser utilizado em aplicações remotas	Faixa de temperatura lida de -55°C a 125°C
Baixo custo	Erro de precisão de +-0,5°C na faixa de temperatura -10°C a 85°C
Opera de 4V a 30V	Resolução da temperatura programável de 9 a 12bits
Corrente de fuga menor que 60uA	Converte temperatura de 12bits para digital em um máximo de 750ms
Baixo aquecimento, 0,08C em ar parado	Configuração de alarmes de temperatura
Baixa impedância, 0,1Ω para 1mA	Comando de alarme identifica e endereça componentes cujo temperatura está fora dos limites programados
	Aplicações em controles termoestáticos, sistemas industriais, termômetros, etc

Como pode-se notar na tabela acima, o sensor DS18B20 será uma escolha mais apropriada para o projeto pois apresenta um erro de leitura de 0,2°C (Figura 11), menor que o erro do LM35 na faixa de temperatura desejada.

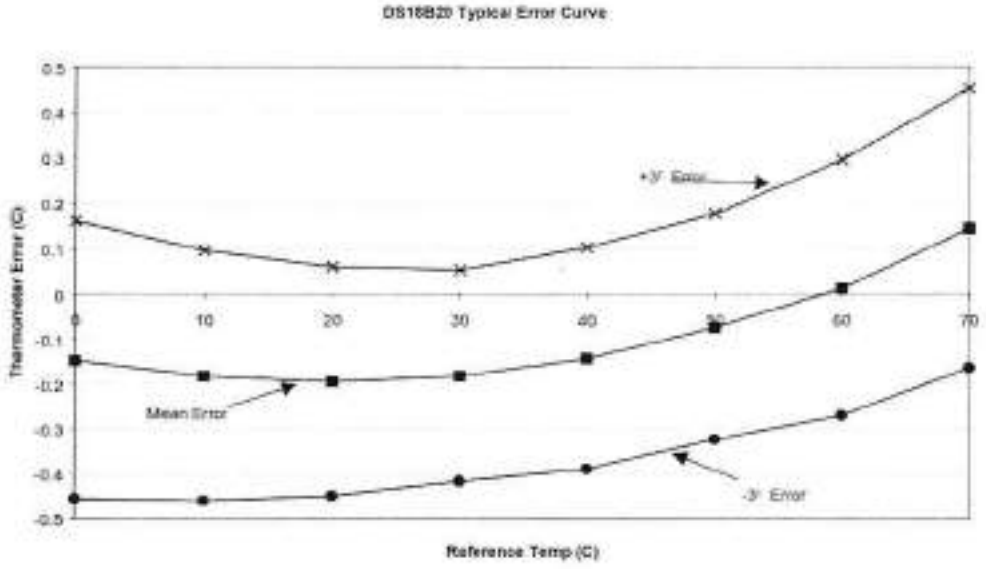

Figura 11: Curva de erro do sensor DS18B20 (DALLAS SEMICONDUCTOR)

Além disso, ele possui a tecnologia 1-Wire, que permite utilizar vários sensores conectados a apenas um pino de leitura do Arduino pois cada sensor possui serial number específico. Essa tecnologia será de grande utilidade no nosso projeto se desejarmos ter uma média da leitura de mais de um sensor.

Figura 12: Sensor ds18b20.

O sensor DS18B20 também possui 3 pinos (Figura 13) e as ligações feitas do sensor com o Arduino para impressão dos testes estão apresentadas na Figura 13. Também foi necessário a utilização de um resistor de 4,7kΩ, pois sem ele o sensor não será detectado pelo programa.

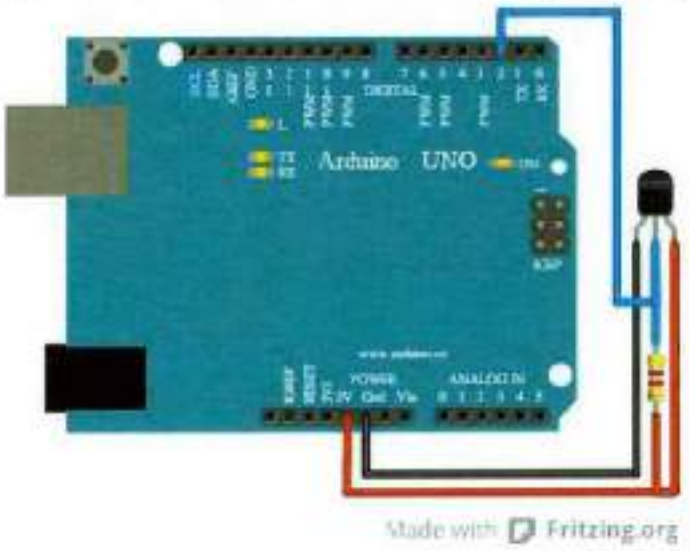

Figura 13: Ligação sensor com Arduino e representação dos pinos do sensor

Para o programa utilizado no Arduino, foi necessária a inclusão das bibliotecas OneWire e Sketch. O teste foi feito colocando-se o DS18B20 em contato com a pela na região próxima à axila ao mesmo tempo que foi medida a temperatura por um termômetro axilar digital e o tempo de cada leitura foi de 1000ms (1s).

Para a prova de conceitos foram impressas as temperaturas lidas pelo sensor no próprio programa do Arduino, sua representação encontra-se no gráfico abaixo (Figura 14):

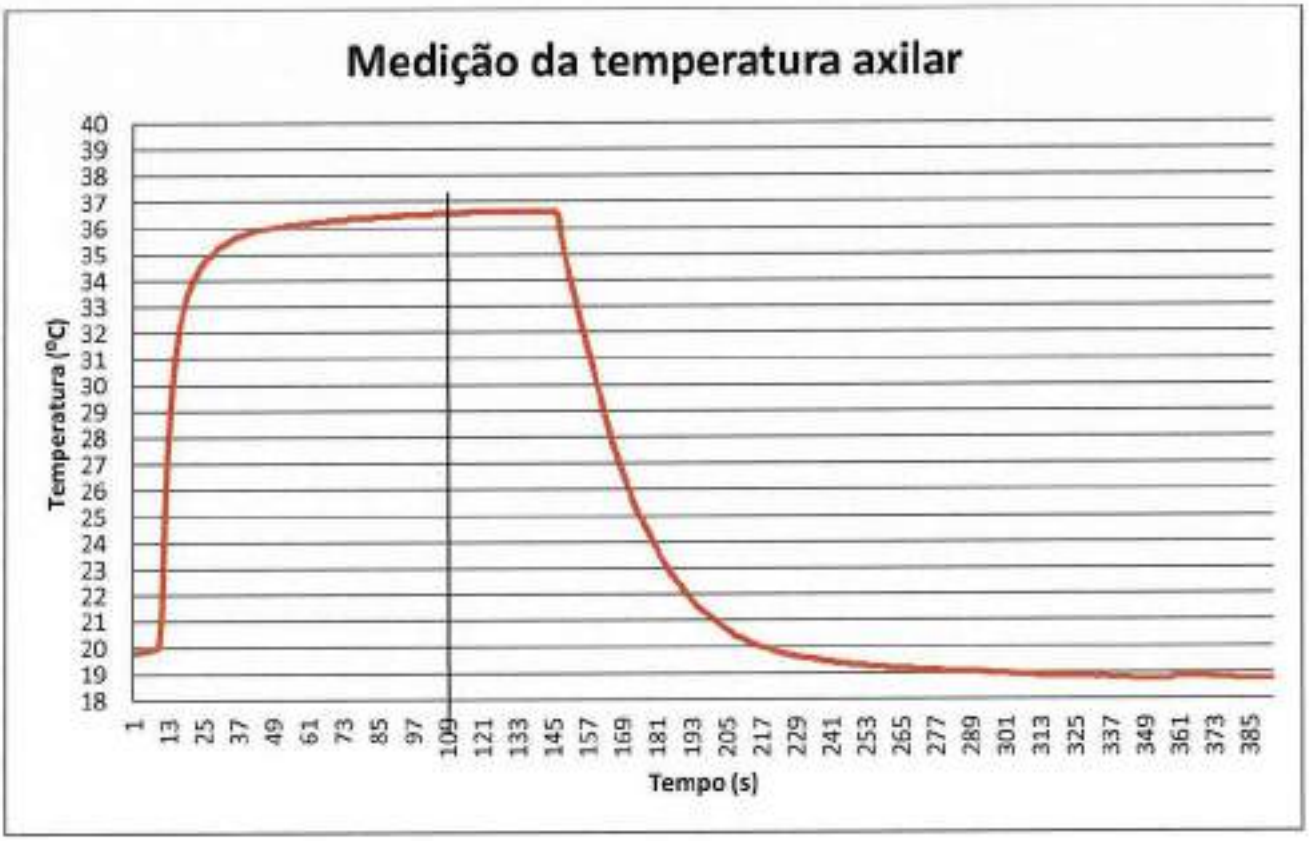

Figura 14: Temperatura axilar medida pelo sensor DS18B20

A temperatura lida no termômetro digital foi de 36,6°C e a temperatura de estabilidade do sensor foi de 36,63°C, provando que o DS18B20 apresenta uma alta precisão. É importante notar também o tempo de resposta do sensor pois após aproximadamente 100 segundos ele já havia chegado à estabilidade (tempo inferior aos termômetros digitais que levam cerca de 3 minutos).

6.1.2 Microcontrolador

Para os teste iniciais do projeto, foi-se utilizado o Arduino Uno. Entretanto, para atendermos um dos principais requisitos do projeto (segundo foi explicado no item 3.5), o tamanho, foi necessária a troca para um microcontrolador menor.

Portanto, foi-se escolhido o Arduino Nano. O Arduino Nano é uma placa pequena, completa e compatível com ATmega328P, podendo funcionar via conexão Mini-B USB, por uma fonte externa irregular de 6-20V (pelo pin30), ou por uma fonte regular de 5V.

Cada um dos 14 pinos digitais do Nano pode ser usado como input ou output, através de funções específicas, sendo operadas a 5 Volts. Cada pin recebe ou envia um máximo de 40 mA e tem uma resistência pull-up interna de 20-50 kOhms (ARDUINO).

Entretanto, para melhores adequações em relação ao gerenciamento de energia do hardware, o Arduino Pro Mini será a melhor escolha para o projeto, como será melhor detalhado no item 6.1.3 de Gerenciamento de Energia. As característica de cada Arduino estão descritas na Tabela 26.

Tabela 26: Comparação entre as características das placas de Arduino (Arduino).

Nome	Uno	Nano	Pro Mini
Processador	ATmega328P	ATmega168 ATmega328P	ATmega328P
Operating/Input Voltage	5 V / 7-12 V	5 V / 7-9 V	3.3 V / 3.35-12 V 5 V / 5-12 V
CPU Speed	16 MHz	16 MHz	8 MHz 16 MHz
Analógico In/Out	6/0	8/0	6/0
Digital IO/PWM	14/6	14/6	14/6
EEPROM [KB]	1	0.512 1	0.512
SRAM [KB]	2	1 2	1
Flash [KB]	32	16 32	16
USB	Regular	Mini	-
UART	1	1	1

6.1.3 Bateria - Gerenciamento de Energia

Tratando-se de um dispositivo wearable é indispensável e de suma importância a realização de um estudo do consumo de energia do módulo adesivo. Para isso, os componentes escolhidos foram analisados de forma teórica e empírica com o objetivo de validar a proposta inicial de duração da bateria, assim serão propostas modificações de software e de hardware para otimização do projeto.

No item 4.3, foi inicialmente considerado a utilização de uma bateria recarregável como a mais apropriada para o projeto. Entretanto, como será explicado no item 6.1.3.1, a fonte de energia mais adequada para o projeto será a Bateria eletrônica de lítio 3 V / 220 mAh CR2032.

6.1.3.1 Metodologia

Por padrão, as baterias disponíveis no mercado não disponibilizam diretamente a informação da medida de sua energia, o fabricante fornece a sua capacidade em termos de mAh (unidade usada para identificar a transferência de carga elétrica por meio de uma corrente estável de 1 mA ao longo de uma hora). Assim sendo, os testes foram realizados colocando um amperímetro em série com a fonte de alimentação para, a partir de sua leitura, poder se estimar o tempo de duração da bateria. É relevante salientar também que o consumo de energia no Arduino e no Bluetooth representa grande parcela do consumo total, por conseguinte, a análise será feita basicamente nesses dois componentes.

Foram realizados testes para o primeiro modelo proposto sem modificações, para esse caso a leitura para os momentos anteriores ao pareamento foi em média de 70 mA, durante o pareamento foi de 48 mA (valor também encontrado ao longo da transmissão de dados) e após o pareamento foi de 32 mA (esses são os valores médios pois a medida das correntes apresentaram pequenas flutuações) . O consumo do módulo Bluetooth pode ser visto no datasheet (HC), onde a corrente durante o pareamento se situa entre 30 e 40 mA e após o pareamento é de aproximadamente 8 mA, justificando as divergências nos dados de corrente encontradas nas diferentes situações.

Supondo uma bateria comum de 9V com capacidade de 440 mAh e considerando que nessas configurações ocorra transmissões com duração de 5 segundos a cada 15 minutos, constata-se que o ciclo da corrente se repete nesse intervalo de tempo e é representado pela Figura 15 (desconsidera-se a corrente de pareamento e pré pareamento pois só acontecem no primeiro ciclo, esse critério também será utilizado para os testes posteriores). Portanto, a corrente média será de aproximadamente 32,1 mA, gerando uma estimativa para a duração da bateria de 13,7 horas, valor aquém dos resultados esperados.

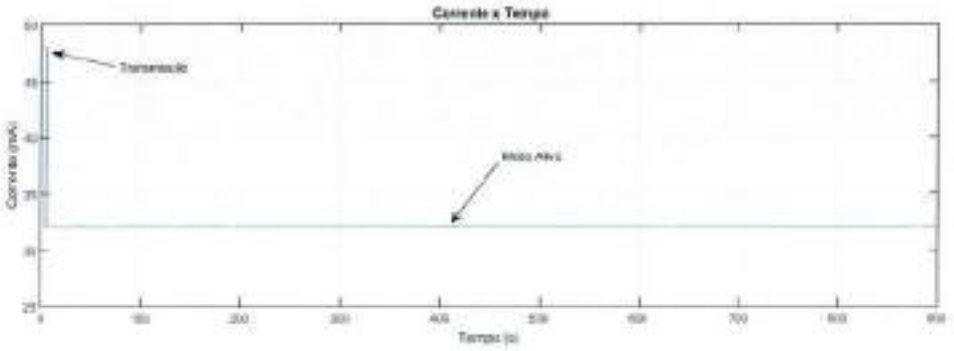

Figura 15: Gráfico da Corrente medida x Tempo sem nenhuma modificação.

Uma das alternativas para redução do consumo de energia é modificar o software do Arduino. Sabendo-se que o objetivo do produto é fazer a leitura de temperatura do paciente deve-se levar em conta que é necessária a obtenção de dados apenas em intervalos de tempo, pois mesmo em casos mais agudos a variação de temperatura não é instantânea. Assim sendo, a transmissão de dados pode ser realizada apenas nesses momentos, ou seja, pode-se tornar o módulo adesivo inativo durante esse hiato (mantendo apenas o pareamento). Essa função pode ser implementada aplicando a biblioteca "Narcoleptic.h" ao programa do embarcado, ela é responsável por executar o uso do "sleep" e manter o Arduino em modo de economia.

Existem várias alternativas para a utilização dessa função, a partir da Tabela 27 nota-se que cada possibilidade representa quais componentes serão mantidos ativos, podendo "despertar" novamente após um tempo determinado ou a partir de um trigger externo. No caso da biblioteca escolhida o modo usado é o power-down onde é usado um intervalo de tempo para tornar o processador ativo novamente.

Essa função só pode ser usada para períodos de no máximo 8 segundos, a implementação para intervalos de tempo maiores são possíveis apenas usando funções de loop, como o "while" ou o "for". O trecho do código responsável pela execução do que foi colocado anteriormente é o seguinte:

```
for (int i=0;i<=104;i++) {
        Narcoleptic.delay(8000);
    }
```

Especificamente nesse caso o módulo adesivo permanece em power-down durante 14 minutos, terminado esse tempo o código segue seu fluxo normal de execução.

Tabela 27: Clocks ativos e Wape-up Sources em diferentes modos "sleep"(ATMEL, 2014).

Table 10-1. Active Clock Domains and Wake-up Sources in the Different Sleep Modes.

	Active Clock Domains					Oscillators		Wake-up Sources							
	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Oscillator Enabled	INT1, INT0 and Pin Change	TWI Address Match	Timer2	SPM/EEPROM Ready	ADC	WDT	Other I/O	Software BOD Disable
Idle			X	X	X	X	X(2)	X	X	X	X	X	X	X	
ADC Noise Reduction				X	X	X	X(2)	X(3)	X	X(2)	X	X	X		
Power-down								X(3)	X				X		X
Power-save					X		X(2)	X(3)	X	X			X		X
Standby(1)						X		X(3)	X				X		X
Extended Standby					X(2)	X	X(2)	X(3)	X	X			X		X

Com essas características a nova leitura de corrente após a sincronização dos dispositivos foi de 24 mA nos momentos em que não estava ativo. Supondo que permaneça ligado em full power por 1 minuto e em power down por 14 minutos o ciclo da corrente será dado pela Figura 16. A corrente média será de 24,6 mA acarretando uma estimativa de duração de bateria de 17,9 horas, valor ainda abaixo dos requisitos exigidos.

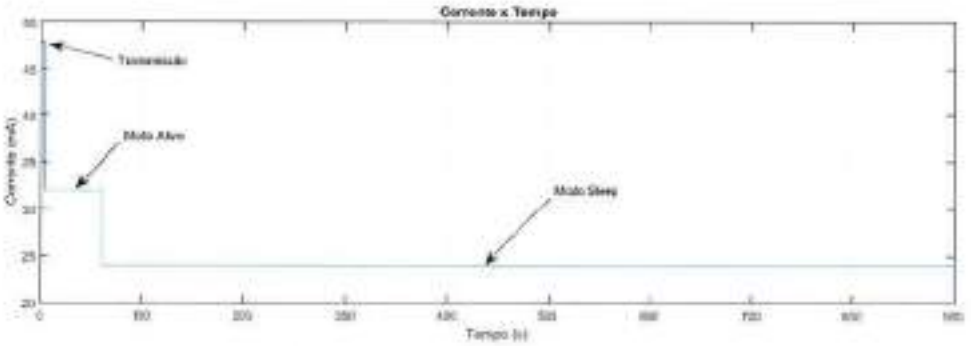

Figura 16: Corrente medida após a implementação da função sleep.

Observando a Figura 16, nota-se que o consumo durante o modo sleep continua sendo alto para os requisitos propostos, consequentemente, mais modificações são necessárias para atingir o objetivo. Como apenas alterações no software não foram suficientes, foram feitas também mudanças no hardware. No formato atual do projeto, o componente HC-05 e os demais periféricos estão conectados diretamente ao Vcc do Arduino, que está sempre em HIGH, sendo a principal causa do consumo excessivo do módulo adesivo mesmo em power-down. Como solução ligou-se a alimentação dos periféricos a um pino digital do embarcado, por conseguinte, o Arduino adquire a capacidade de desligá-los quando necessário.

Sendo assim, modifica-se o código de execução do embarcado para que, nos momentos em que estiver em power-down, sejam desligados todos os componentes periféricos. O trecho de código da implementação dessa função (complementar ao citado previamente) está disposto abaixo:

```
digitalWrite(Vcc,LOW);
        digitalWrite(2,LOW);
        digitalWrite(6,LOW);
        digitalWrite(7,LOW);
         for (int i=0;i<=104;i++) {
            Narcoleptic.delay(8000);
          }
        digitalWrite(Vcc,HIGH);
        digitalWrite(2,HIGH);
        digitalWrite(6,HIGH);
        digitalWrite(7,HIGH);
```

Antes da utilização da função sleep define-se os pinos digitais empregados no código em LOW, dessa forma o consumo dos periféricos será nulo enquanto o módulo estiver em power-down, ao voltar para o estado ativo define-se novamente os pinos em HIGH para que a leitura possa ser feita e o ciclo se reinicie.

Além da "Narcoleptic.h" também foi incluída a biblioteca "avr/power.h", com ela é possível desativar módulos ociosos do processador durante o estado ativo do Arduino. Como citado anteriormente o sensor ds18b20 faz a leitura da temperatura e realiza a conversão analógico-digital, possibilitando inativar o conversor presente no processador com os comandos:

```
ADCSRA = 0;  //desabilita a conversao analogico digital no arduino
power_adc_disable(); // fixa a configuração do conversor, que no caso está em 0
```

Nas configurações utilizadas até o momento o embarcado não recebe clocks externos e também não é responsável por fornecer o clock para algum periférico, portanto, o SPI (Serial Peripheral Interface, que realiza a comunicação serial síncrona do Arduino com outros componentes) é desnecessário para esse projeto, sendo desabilitado a partir do comando:

```
power_spi_disable(); // desabilita SPI
```

Como não serão transmitidos ou recebidos dados via cabo serial o USART0 (Universal asynchronous receiver/transmitter) também não será utilizado, é desativado pelo comando:

power_usart0_disable();// Serial (USART)

Assim como o SPI o TWI (variação do I2C, Inter-Integrated Circuit) não é útil nesse projeto pois não existe comunicação com protocolo master-slave entre o embarcado e os periféricos, pode-se inativá-lo a partir do comando:

power_twi_disable(); // TWI (I2C)

Essas modificações propostas são detalhadas nas seções 24.9, 19.5, 20.11 e 22.9 respectivamente(ATMEL, 2014).

Para essa nova configuração a leitura de corrente durante o pareamento e transmissão de dados foi de 47 mA (duração de 5 segundos cada), para antes do pareamento a leitura foi de 65 mA (duração de 2 segundos), para o modo ativo foi de 31 mA e para o estado power-down 6,4 mA (nesse caso o pareamento e o pré-pareamento devem ser considerados pois como há o desligamento completo do módulo bluetooth é necessário que haja a sincronização em todos os ciclos). Assim, o período da corrente é dado pela Figura 17 a seguir:

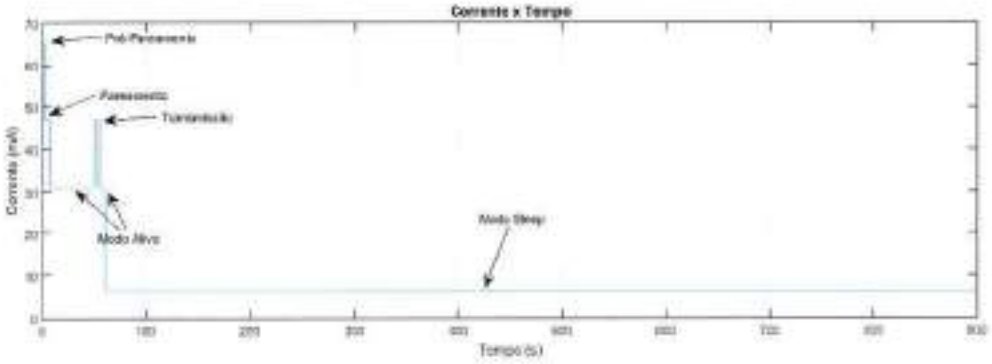

Figura 17: Corrente medida após a implementação da alimentação pelo pino digital

Com essa estrutura a corrente média é de 8,3 mA, proporcionando uma estimativa de duração da bateria de 53 horas, valor coerente com os requisitos estipulados.

Foi considerado até esse momento para os cálculos uma bateria comum de 9V, no entanto, suas dimensões afetam diretamente outro requisito considerável do projeto: o tamanho. Por esse motivo, e tendo em vista que a corrente média atingiu valores que possibilitam o uso de baterias com cargas menores, optou-se por substituir esse modelo

por 3 baterias em série do tipo moeda (CR2032), que possuem uma carga de 220 mAh, menor custo, menor dimensão e juntas proporcionam a mesma tensão da bateria comum, nesse caso o tempo de duração estimado passa a ser de 26,5 horas.

Apesar de atingir os requisitos, a troca de bateria reduziu pela metade a estimativa de uso sem a sua troca e ainda existem fatores que amplificam o consumo de bateria. Analisando o esquemático na Figura 18 do Arduino nano usado para os testes nota-se que ele possui uma interface USB que depende apenas da alimentação do regulador de tensão (+5V), isto é, independente do modo em que o embarcado estiver operando não haverá economia de energia nessa interface.

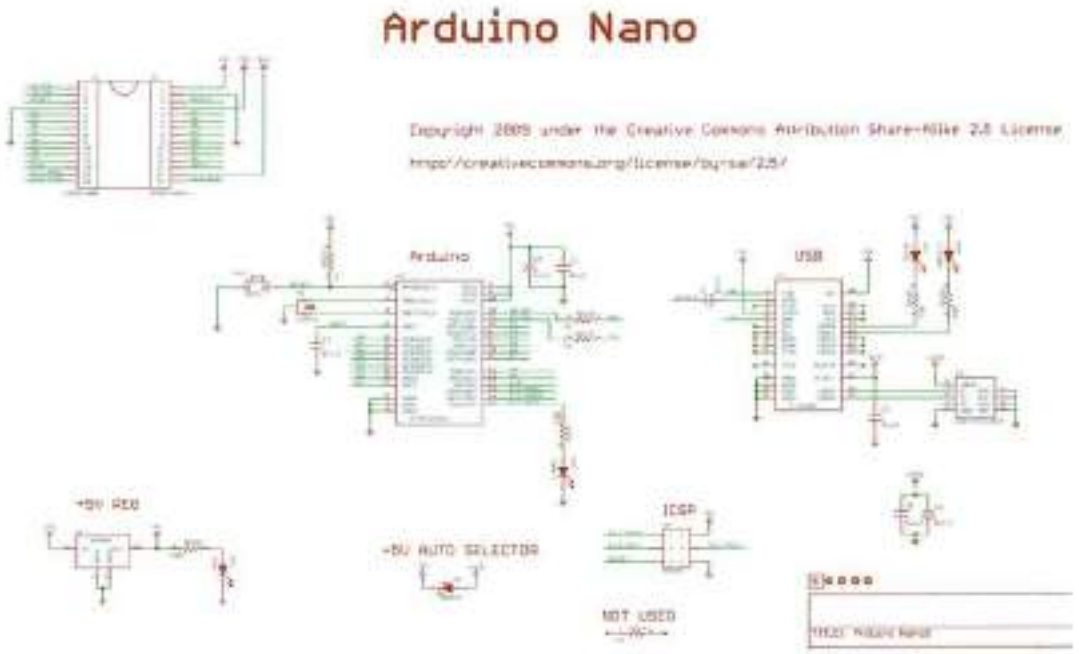

Figura 18: Esquemático do Arduino nano

Como alternativa pode-se utilizar um outro modelo de embarcado: o Arduino pro mini. Nesse componente pode-se observar a partir do seu esquemático (Figura 19) que não existe uma interface USB, o que representa uma redução no consumo de energia, além de possuir menores dimensões e custo, ou seja, é favorável aos requisitos do projeto (itens 3.2 e 3.4).

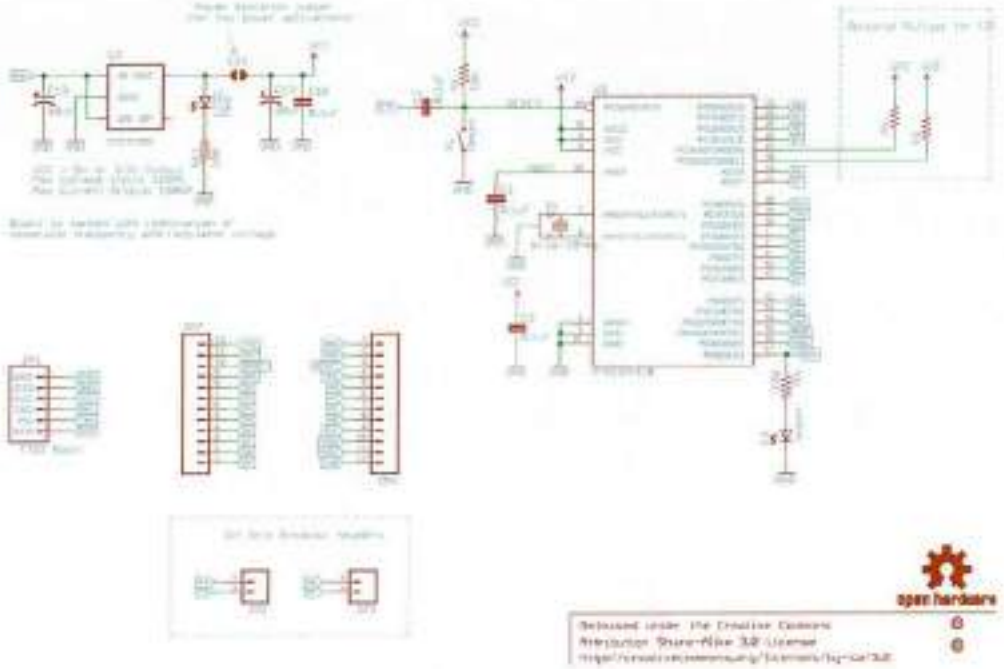

Figura 19: Esquemático do Arduino pro mini

Implementou-se as mesmas configurações do teste mais eficiente do Arduino nano no Arduino pro mini, dessa forma, a leitura de corrente pré-pareamento foi de 60 mA, durante o pareamento foi de 45 mA, para o modo ativo foi de 29 mA e para o modo power-down foi de 3,3 mA. Assim sendo, o ciclo da corrente para esse modelo é representado pela Figura 20.

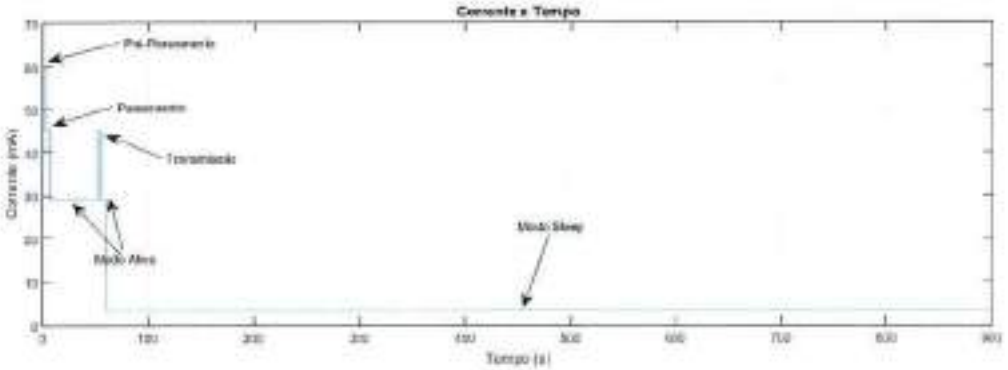

Figura 20: Corrente medida no Arduino pro mini

Portanto, para esse embarcado a corrente média será de aproximadamente 5,3 mA, gerando uma estimativa de duração da bateria de 41,5 horas, valor que cumpre de maneira satisfatória os requisitos exigidos. A Tabela 28 mostra um resumo das correntes médias obtidas de acordo com as configurações.

Tabela 28: Resumo dos resultados obtidos.

Configuração	Corrente Média(mA)	Tempo Estimado da Bateria (horas)
Nenhuma modificação (Arduino nano)	32,1	Bateria comum de 9V 13,7 Bateria moeda 6,8
Inclusão da biblioteca Narcoleptic (Arduino nano)	24,6	Bateria comum de 9V 17,9 Bateria moeda 8,9
Inclusão da biblioteca avr/power.h + Uso do pino digital para alimentação dos periféricos (Arduino nano)	8,3	Bateria comum de 9V 53,0 Bateria moeda 26,5
Todas as modificações implementadas no Arduino pro mini	5,3	Bateria comum de 9V 83,0 Bateria moeda 41,5

6.1.4 Implementação Final

Na Figura 21 abaixo, temos a representação em 3D final do Hardware, desenhado pelo software Kicad 4.0.0 RC1. O código final utilizado para a programação do Arduino encontra-se no Apêndice A –.

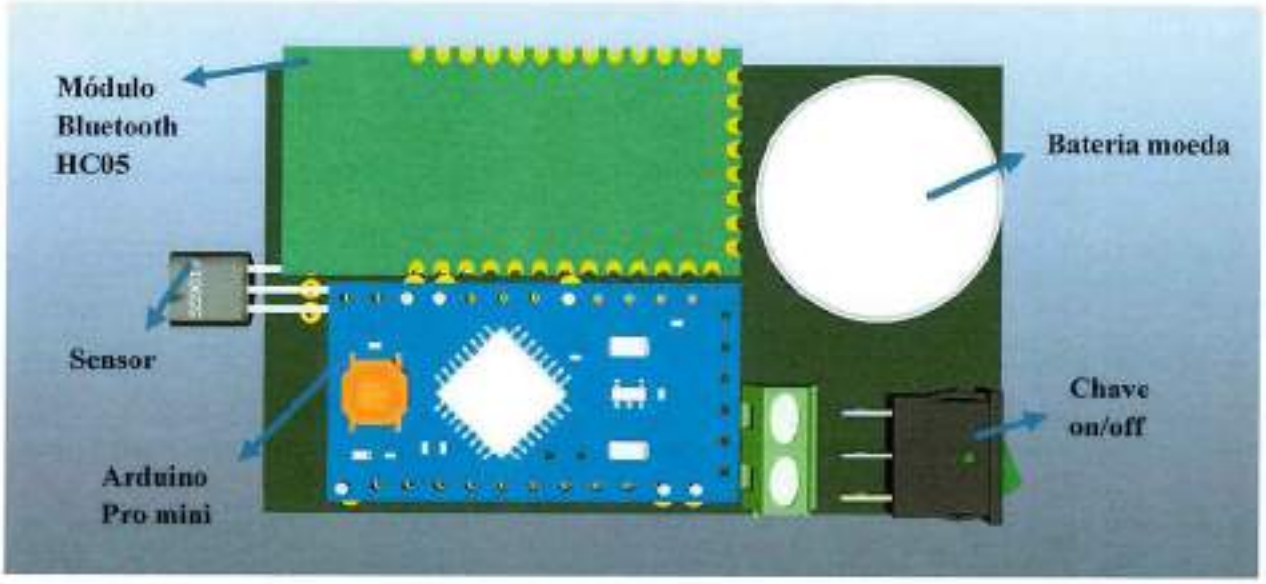

Figura 21: Visão frontal do módulo final implementado.

E na Figura 22, encontra-se o circuito impresso, de apenas uma camada, que será utilizado para confecção da placa de cobre. A dimensão final da placa será de 6x4x1cm³.

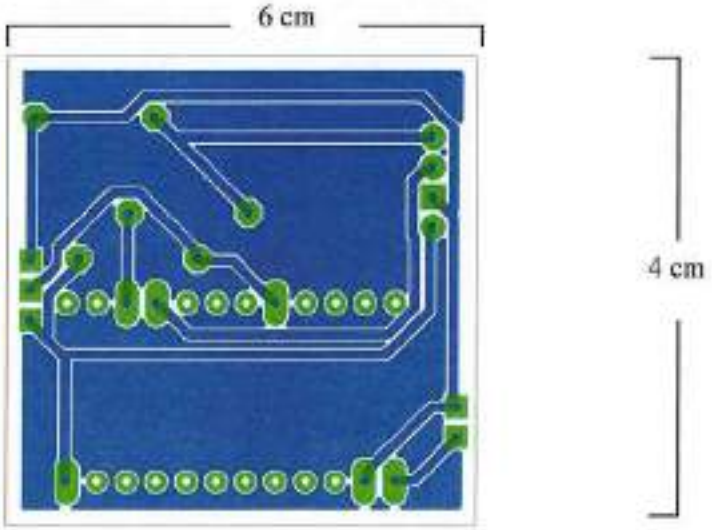

Figura 22: Circuito impresso do módulo final implementado.

6.2 SOFTWARE

Na 2ª fase do projeto do Módulo Adesivo, é necessária a criação de uma interface para comunicação do computador pessoal com o *hardware*, via Bluetooth, e que seja amigável ao usuário. Dentre as várias opções de linguagens de programação que possibilitavam a criação desta interface, a linguagem Java foi a escolhida, devido à abundância de bibliotecas com funções pertinentes a esta aplicação.

Após uma vasta pesquisa e análise das horas/homem necessárias para implementação da interface, foi decidido a utilização do Software Processing 3.0b4 (PROCESSING FOUNDATION), que é uma ferramenta de linguagem e desenvolvimento aberta para escrita de programas em Java, utilizada quando deseja-se que um computador se comunique com o Arduino, por exemplo para mostrar ou salvar dados coletados deste.

O Processing foi utilizado para se comunicar com o Módulo Bluetooth através de uma porta serial específica do computador (COM). Para a implementação das funções propostas foi necessário incluir as seguintes bibliotecas:

processing.serial

org.gicentre.utils.stat

ddf.minim

A primeira é responsável pela comunicação serial através do Bluetooth dos dois dispositivos, é encarregada do protocolo de interação entre eles, definindo assim bits de paridade, velocidade de transmissão, bits de dados e stop bits. A utilidade da segunda é prover um método prático de construção de gráficos a partir do arquivo .csv gerado, complementando assim duas funcionalidades oferecidas pelo programa. A terceira biblioteca citada possibilita a execução de arquivos de áudio, item necessário para a criação do alarme sonoro.

Para que seja possivel a implementação do fluxograma apresentado na Figura 23, são necessárias modificações na biblioteca "*processing.serial*". Isso se deve ao fato de que as funções executadas por ela interrompem o curso do programa caso a conexão não seja bem sucedida ou caso não encontre nenhum dispositivo para comunicação, ou seja, sem essa alteração o executável sofreria um "crash" em qualquer momento que o módulo Bluetooth desligasse, impossibilitando a efetivação do que foi sugerido no item 6.1.3 (Bateria - Gerenciamento de Energia). Dessa forma, o trecho de código modificado é apresentado da seguinte maneira:

```
try {
    // the native open() call is not using O_NONBLOCK, so this might block for certain
operations (see write())
    port.openPort();
    port.setParams(baudRate, dataBits, stopBitsIdx, parity);
    // we could register more events here
    port.addEventListener(this, SerialPort.MASK_RXCHAR);
  } catch (SerialPortException e) {
    // this used to be a RuntimeException before, so stick with it
    //throw new RuntimeException("Error opening serial port " + e.getPortName() + ": " +
e.getExceptionType());

  }
```

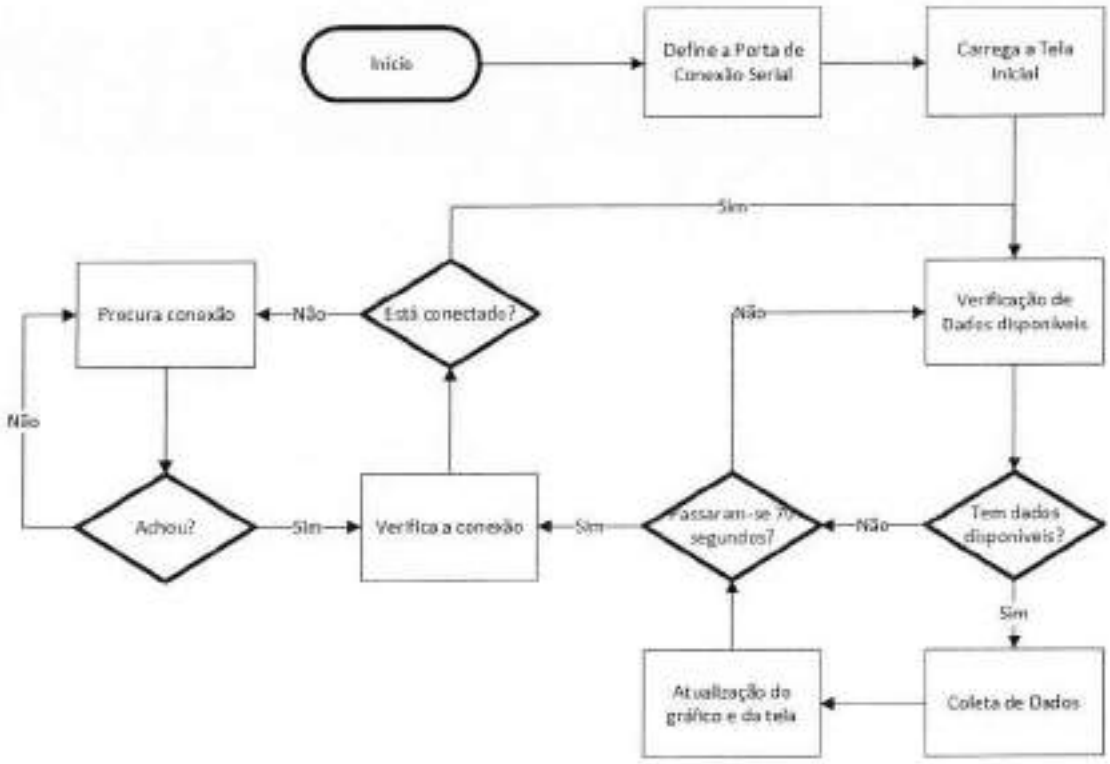

Figura 23: Fluxo de lógica para interface do usuário.

Com isso, mesmo que não haja conexão o programa continua sendo executado, o computador continuará monitorando o ambiente em busca do dispositivo médico até que ele esteja disponivel para pareamento e posterior envio de dados. Após a recepção dos valores de temperatura codificados em ASCII o programa faz o tratamento dos dados e os disponibiliza na interface para o usuário em forma de tabela e gráfico, atualizando a tela e o arquivo *.csv* para os novos valores recebidos.

A conexão dura em torno de 1 minuto, em seguida o componente HC-05 desliga pois o embarcado entra no modo power-down (no qual permanece por 14 minutos), todavia, mesmo com essa situação o software continua reconhecendo a conexão como ativa inviabilizando futuras recepções de dados. Por esse motivo utiliza-se o seguinte comando:

```
if ( millis() - lastTime > 70000 ) {
      //println( "do things every 70 seconds" );
       myPort.stop();
        while(myPort.active()==false){
          String portName = Serial.list()[2];
          myPort=new Serial(this, portName, 9600);
          }
        lastTime = millis();
      }
```

Este trecho é responsável por encerrar a conexão antiga e procurar por uma nova, continua em loop até que possa parear com o módulo adesivo, reiniciando o ciclo. Esses comandos são sempre executados após a recepção e tratamento dos dados (por isso é indispensável que se tenha uma referência de tempo, representada por "millis()").

Dentro da interface espera-se passar para o usuário as funções:

- gráfico de temperatura *vs.* tempo;
- tabela com últimas leituras de temperatura;
- alarme;
- envio email direcionado ao médico do paciente;
- temperatura real da última leitura;
- geração de arquivo *.csv* com os dados coletados.

O layout final desenhado pelo Microsoft Visio 2010 está representado na Figura 24.

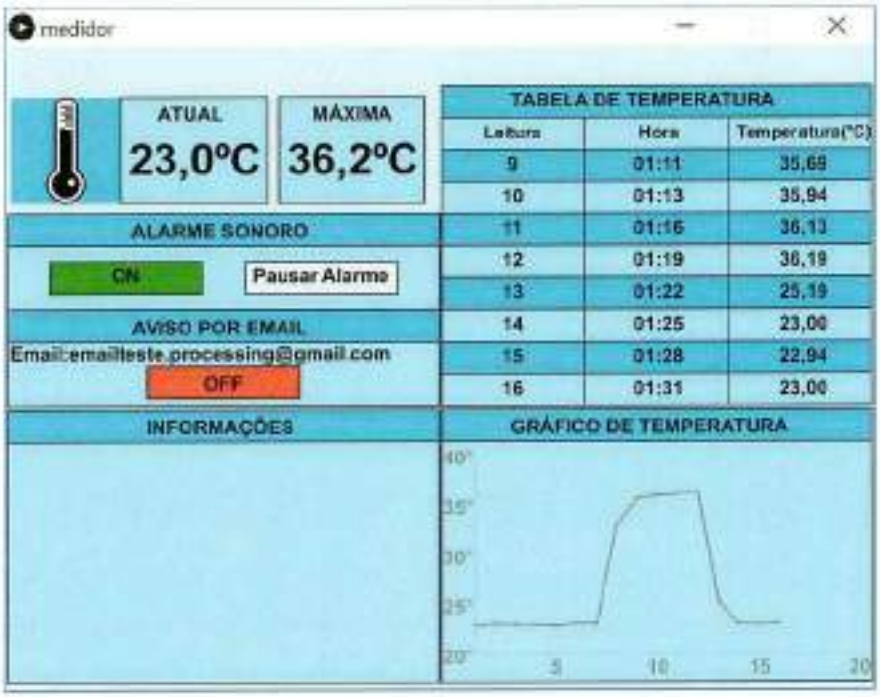

Figura 24: Projeção da interface final do usuário.

O código utilizado no Processing encontra-se no Apêndice B –. O fluxo de comandos será definido a seguir:

1. instanciação de variáveis e bibliotecas;
2. verificação de conexão com porta serial;
3. abertura de arquivo *.csv* para coleta e gravação de dados da leitura;
4. função "void draw", contendo as funções principais.

A função "void draw" possui os principais comandos para impressão da interface final, sendo implementado o gráfico de temperatura, além da função alarme para o usuário, atendendo aos requisitos de Marketing (item 3.2).

6.2.1 Plataforma Android

Devido ao grande mercado de portáteis (como smartphones e tablets) considerou-se também a implementação de um software direcionado a esse público, que apresenta a vantagem de ser de fácil acesso e de uso frequente e cotidiano. Nessa área existem muitas opções de sistemas operacionais, dentre elas as principais são o iOS, o Android e o Windows Phone. Tendo em vista que o sistema operacional Android domina grande parte do mercado de portáteis (nos smartphones esse domínio atingiu 90% do mercado brasileiro no primeiro trimestre de 2015 segundo informações da Kantar WorldPanel ComTech) optou-se por utilizá-lo, possibilitando maior abrangência para o projeto.

Sendo assim, criou-se um aplicativo para Android que disponibiliza as seguintes funções para comodidade do usuário:

- alarme sonoro em caso de febre;
- email de aviso com anexo das temperaturas lidas caso seja constatada situação de febre;
- temperatura real da última leitura;
- geração de arquivo *.txt* com os dados coletados

É possível notar a partir da interface com o usuário ilustrada na Figura 25 os itens listados acima, além da alternativa de ativar ou desativar os serviços de aviso por email e de alarme sonoro.

Figura 25: Interface para usuário para smartphone Android.

Para a comunicação do aplicativo com o módulo bluetooth foi utilizada a mesma lógica citada neste capítulo para o caso do programa para Windows, divergindo apenas na forma em que procura novamente pela conexão. O Processing (no momento em que encerra a conexão para tentar uma nova) busca constantemente o dispositivo bluetooth, no caso do app Android (onde a bateria também possui prioridade), essa busca é feita em intervalos de tempo justamente para que haja um menor consumo de energia.

Para a implementação do aplicativo foram consideradas as seguintes condições:

- A leitura de temperatura deve ser constante, independente se o usuário está com o aplicativo ativo na tela.
- O usuário deve estar ciente de que o aplicativo está em funcionamento
- A interface deve facilitar a interação do cliente, inclusive na ativação e pareamento dos dispositivos.

Para que um aplicativo continue rodando em background (ou seja, para que continue ativo mesmo que não seja a atividade principal) é necessária a criação de um serviço, no entanto, um serviço simples está predisposto a muitas ações do sistema para o seu fechamento (o SO pode optar por encerrar o serviço para liberar memória) , o que inviabilizaria uma conexão confiável entre o módulo adesivo e o dispositivo portátil.

Dessa forma, é vital que se atribua uma prioridade maior ao aplicativo para que só seja encerrado em casos extremos, sendo para isso executado um Service Foreground, esse tipo de serviço confere ao programa preferência no sistema e está demonstrado a seguir:

```
PendingIntent pendingIntent = PendingIntent.getActivity(this, 0,
        notificationIntent, 0);
    Bitmap icon = BitmapFactory.decodeResource(getResources(),
        R.drawable.termometro);
    Notification notification = new NotificationCompat.Builder(this)
        .setContentTitle("Easy Term")
        .setTicker("Easy Term")
        .setContentText("Monitoramento de Temperatura")
        .setSmallIcon(R.mipmap.ic_launcher)
        .setLargeIcon(Bitmap.createScaledBitmap(icon, 128, 128, false))
        .setContentIntent(pendingIntent)
        .setOngoing(true).build();
    startForeground(101,notification);} //inicio da execução de Foreground
```

Além de atender a condição estabelecida no primeiro item essa função também cumpre o segundo item, pois gera uma notificação ao usuário que pode ser vista com o programa em segundo plano (Figura 26).

Figura 26: Notificação de funcionamento do programa na tela principal do smartphone.

Para uma interface mais amigável ao usuário e maior facilidade no uso, foram implementados botões para que a descoberta e pareamento do dispositivo bluetooth sejam feitos pelo próprio aplicativo. Além disso, o usuário possui o controle sobre os

alertas gerados pelo programa, as variáveis que controlam o envio de email, e o alarme sonoro em caso de febre. Eles são dados pelo próprio cliente a partir de flags geradas pelos switchs da interface, podendo ser observado a partir do código abaixo:

```
if((temp>37.8)||(temp<35.9)){   //verifica se a temperatura está fora do normal
        flagSendEmail++;
        if(flagAlarmeSonoro==1))
        sound.start();          }
```

Isto posto, garante-se o cumprimento das condições estabelecidas proporcionando ao usuário facilidade de uso e conforto.

7 CONCLUSÃO

O protótipo criado neste Projeto de Formatura tentou seguir os Requisitos de Marketing e Engenharia propostos no inicio da identificação das necessidades. Para isso, várias mudanças foram necessárias no escopo do projeto.

Inicialmente, foi-se trocado o sensor LM35 da Texas Instrument para o sensor DS18B20 da Dallas para suprir o requisito de precisão, pois o primeiro sensor escolhido apresentava um erro de leitura maior que 0,5°C. Depois, o tipo do Arduino utilizado foi modificado, sendo escolhido o Arduino Promini, atendendo ao requisito de tamanho.

Ainda atendendo o requisito de tamanho, o protótipo final utilizou uma bateria de Lítio e um módulo Bluetooth como escolha de transmissão mais confiável de dados.

Ao tratarmos os dados pelo programa Processing no computador, foi possível obter uma interface clara e de fácil compreensão para o usuário final, assim como poder informá-lo se o paciente apresenta febre com uma precisão menor que 0,2°C.

Portanto, tanto o módulo Hardware como o módulo Software se adequaram às necessidades do projeto, além de apresentarem uma alta performance e durabilidade, satisfazendo o objetivo de monitorar a temperatura de crianças de 0 a 5 anos durante 24 horas, permitindo avisar ao usuário se o paciente apresentar febre.

8 APÊNDICES

APÊNDICE A – CÓDIGO HARDWARE

Código utilizado no Software Arduino 1.6.4

```
#include <OneWire.h>          //inclusão dos bibliotecas
#include <SoftwareSerial.h>
#include <Narcoleptic.h>
#include <avr/power.h>

SoftwareSerial bt(6, 7); // RX  TX
OneWire  ds(2);  // define o pino digital 2 para recepção dos dados do sensor
const int Vcc=10; //define o pino 10 como Vcc
long lastTime = 0;  //define uma variável para funções com o tempo decorrido de execução

void setup(void) {
 lastTime = millis();  //usa a função millis para leitura do tempo
 pinMode(Vcc, OUTPUT);
 digitalWrite(Vcc,HIGH);
 ADCSRA = 0;  //desabilita a conversao analogico digital no arduino
 power_adc_disable(); // ADC
 power_spi_disable(); // SPI
 power_usart0_disable();// Serial (USART0)
 power_twi_disable(); // TWI (I2C)
 //power_timer0_disable();// Timer 0
 //power_timer1_disable();// Timer 1
 //power_timer2_disable();// Timer 2
}

void loop(void) {
 byte i;          //definição de variáveis para a recepção dos dados do ds18b20
 byte present = 0;
 byte type_s;
 byte data[12];
 byte addr[8];
 float celsius, fahrenheit;  //variáveis que definirão qual foi a leitura da temperatura

//  digitalWrite(Vcc,HIGH);
```

```
 if ( ids.search(addr)) {
//   Serial.println("Todos os sensores ja foram detectados");
//   Serial.println(); //muitos comandos são utilizados para os testes, principalmente os
   ds.reset_search(); //que utilizam a Serial (que depende da conexão via cabo)
   delay(250);
   return;
 }

// Serial.print("ROM =");
// for( i = 0; i < 8; i++) {
//   Serial.write(' ');
//   Serial.print(addr[i], HEX);
// }

// if (OneWire::crc8(addr, 7) != addr[7]) {
//    Serial.println("CRC invalido!");
//    return;
// }
// Serial.println();

 // abordagem dos diferentes casos e modelos de sensor
 switch (addr[0]) {
   case 0x10:
    //Serial.println("  Chip = DS18S20");
    type_s = 1;
    break;
   case 0x28:
    //Serial.println("  Chip = DS18B20");
    type_s = 0;
    break;
   case 0x22:
    // Serial.println("  Chip = DS1822");
    type_s = 0;
    break;
   default:
    //Serial.println("O dispositivo nao pertence a familia DS18x20.");
    return;
 }
```

```
  ds.reset();
  ds.select(addr);
  ds.write(0x44);       // função de escrita do sensor da biblioteca onewire

  delay(1000);    // espera para que as ações anteriores sejam executadas por completo


  present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);

// Serial.print(" Data = ");
// Serial.print(present, HEX);
// Serial.print(" ");
 for ( i = 0; i < 9; i++) {           // são necessários 9 bytes
   data[i] = ds.read();          //leitura e armazenamento dos dados do sensor
//   Serial.print(data[i], HEX);
//   Serial.print(" ");
 }
// Serial.print(" CRC=");
// Serial.print(OneWire::crc8(data, 8), HEX);
// Serial.println();

 // Converter os dados para a temperatura real
 // porque o resultado é um sinal de 16 bits, ele deve
 // ser armazenado em um "int16_t", que é sempre de 16 bits
 // mesmo quando compilado por um processador de 32 bits.
 int16_t raw = (data[1] << 8) | data[0];
 if (type_s) {
  raw = raw << 3;
  if (data[7] == 0x10) {
   // resolução máximo de 12 bits para a conversão A/D
   raw = (raw & 0xFFF0) + 12 - data[6];
  }
 } else {
  byte cfg = (data[4] & 0x60);
  // para menores resoluções alguns bits são indefinidos, então zera-se eles
```

```
    if (cfg == 0x00) raw = raw & ~7; // resolução de 9 bits, tempo de conversão: 93.75 ms
    else if (cfg == 0x20) raw = raw & ~3; // resolução de 10 bits, tempo de conversão: 187.5 ms
    else if (cfg == 0x40) raw = raw & ~1; // resolução de 11 bits, tempo de conversão: 375 ms
    //// o padrão é a resolução de 12 bits com tempo de conversão de 750 ms
   }
   celsius = (float)raw / 16.0;  //definição da temperatura em Celsius
   fahrenheit = celsius * 1.8 + 32.0;  //definição da temperature em Fahrenheit
  // Serial.print(" Temperature = ");
  // Serial.print(celsius);
  // Serial.print(" Celsius, ");
  // Serial.print(fahrenheit);
  // Serial.println(" Fahrenheit");

   delay(48750); //tempo de espera para que haja o pareamento entre o computador
                 //e o módulo bluetooth
   bt.begin(9600);  //início da transmissão de dados para o computador
   bt.println(celsius); //dados a serem transmitidos
   delay(10000);     //espera para que se execute as funções anteriores por completo
  //   for (int i=0;i<=1;i++) {
  //     Narcoleptic.delay(8000); //
  //   }
                                  //verifica se ocorreu a passagem de 50 segundos desde
   if ( millis() - lastTime > 50000 ) {    //o início da leitura para iniciar o processo de power-down
         digitalWrite(Vcc,LOW);    //defini-se os pinos usados em LOW para economia de energia
         digitalWrite(2,LOW);
         digitalWrite(6,LOW);
         digitalWrite(7,LOW);
          for (int i=0;i<=104;i++) {
              Narcoleptic.delay(8000); //execução do sleep em ciclos de 8 segundos, no total
            }                          // a duração é de 14 minutos
         digitalWrite(Vcc,HIGH); //o embarcado volta a ficar ativo e define-se os pinos em HIGH
         digitalWrite(2,HIGH);
         digitalWrite(6,HIGH);
         digitalWrite(7,HIGH);
         lastTime = millis(); //leitura do tempo para repetir o processo
       }
}
```

APÊNDICE B – CÓDIGO SOFTWARE

Código utilizado no Software Processing 3.0b4

```
import processing.serial.*; //Importa a biblioteca para abrir uma comunicação
import org.gicentre.utils.stat.*;   // bliblioteca para construção do gráfico
import ddf.minim.*;    //biblioteca para execução do áudio

// definição dos objetos das bibliotecas
XYChart lineChart;
Minim minim;
AudioPlayer player;
Serial myPort; //Instancia a biblioteca para a comuniação Serial
//Cria uma instância para cada imagem da interface

PImage fundo; //Background
PImage ponteiro; //Ponteiro
PImage termometro; //Termometro
PrintWriter outputtemp;
int valor_recebido; //Cria uma variável para armazenas o valor recebido pela serial
int[] vetor_asc = new int[7]; //vetor com valores em código ASCII
int[] vetor_hora = new int[8]; //vetor com valores da hora de leitura
int[] vetor_minuto = new int[8];  //vetor com valores dos minutos de leitura
float[] vetor_temperatura = new float[8];  //vetor com valores de temperatura
int a,b,c,d; //variáveis de auxilio para a leitura dos dados recebidos
int hora, minuto, segundo, i, j, flag, leitura;  //variáveis de auxilio
String valor;
boolean teste;
float  temp, eixotempo, eixotemperatura;
long lastTime = 0;

void setup()
{
 lastTime = millis();  //referência de tempo
// println(lastTime);

 //**************************************VERIFICA AS CONEXOES DO
COMPUTADOR************************//
 String portName = Serial.list()[2]; //Lista as portas COM (Serial) encontradas e
```

```
                    //armazena a escolhida na variável portName
          printArray(Serial.list());

          //myPort=new Serial(this, portName, 9600);//Abre uma comunicação Serial com baud rate de
9600 //<>//
          //println(portName);
          //teste=myPort.active();
          //println("teste=",teste);

          //************************************ABERTURA DO ARQUIVO
CSV**********************************//
          outputtemp = createWriter("Temperatura.csv");
          outputtemp.print("TEMPERATURA"+"\n");
          outputtemp.flush();


          //************************DEFINICOES DE
VARIAVEIS*****************************************//
          lineChart = new XYChart(this);
          minim = new Minim(this);
          player = minim.loadFile("groove.mp3"); //definição do arquivo de áudio
          i=0;
          leitura = 1;
          flag=1;

          //************************************DEFINICOES DA
INTERFACE*****************************//
          fundo = loadImage("fundo.jpg"); //imagens a serem carregadas
          ponteiro = loadImage("ponteiro.png");
          termometro = loadImage("termometro.png");
          size(1280, 720); //Define o tamanho da tela
          background(255); //Define o background
          //image(termometro, 2,2);
          //Carrega a fonte de texto a ser utilizada
             textFont(createFont("Arial Bold",18));
             textAlign(CENTER);

//*****************************TABELA*****************************************************
```

```
*****//
        noFill(); //preenchimento do retângulo
        rect(width/2,height/16,width/2-width/30,height/2); //criação do retângulo
        for(j=1;j<10;j++){  //criação de linhas para a tabela
        line(width/2,j*height/20+height/16,width-width/30,j*height/20+height/16);
        }
        line(width/2+width*14/90,height/16+height/20,width/2+width*14/90,height/16+height/2);
        line(width/2+width*28/90,height/16+height/20,width/2+width*28/90,height/16+height/2);
        fill(0);
        text("TABELA DE TEMPERATURA", width/2+width*3*14/180, height/16+height/30);
        text("Leitura", width/2+width*1*14/180, height/16+height/20+height/30);
        text("Hora", width/2+width*3*14/180, height/16+height/20+height/30);
        text("Temperatura(°C)", width/2+width*5*14/180, height/16+height/20+height/30);

        //**********************************CONTORNO
GRAFICO***********************************************//
        fill(255);
        rect(width/2,height/2+height/16+4,width/2-width/30,height/2-height/16-10);
        fill(0);
        text("GRÁFICO DE TEMPERATURA", width/2+width*3*14/180, height/2+height/16+height/30);

        //translate(width/2, height/2+ajuste_y); //Posiciona o ponto 0 da interface no centro da tela
        //                        //Com ajuste de 14 pixels em y (height)
        //rotate(radians(-90));
        //image(ponteiro, -(ponteiro.width/2), -ponteiro.height+(7)); //Posiciona o centro do furo do
ponteiro
        //                                  //Na posição central do gráfico na imagem de fundo

        //**********************************CONDICOES                                    DO
PACIENTE*************************************//
        fill(255);
        rect(2,2,256,120);
        image(termometro, 5,5); //carrega a imagem do termometro
        rect(105,7,148,110);

        //********************PRIMEIRA                    TENTATIVA                    DE
CONEXAO*************************************************//
        myPort=new Serial(this, portName, 9600);//Abre uma comunicação Serial com baud rate de
```

```
9600
        }

        void draw()
        {
         if (myPort.available() > 0) //Se algo for recebido pela serial
         {
          valor_recebido = myPort.read(); //Armazena o que foi lido dentro da variável valor recebido

          vetor_asc[i]=valor_recebido;

           i++;
           if(i==7){
              a=vetor_asc[0]-48;          //a,b,c,d sao os valores recebidos em codigos de ASCII pelo
arduino convertidos
              b=(vetor_asc[1]-48);        //para os valores reais, cada um representa um digito da
temperatura
              c=(vetor_asc[3]-48);
              d=((vetor_asc[4])-48);
          valor=""+a+b+"."+c+d;         //construcao da string contendo a temperatura
          temp=Float.parseFloat(valor);  //converte string pra float
          println(temp);
          i=0;
          hora = hour();
          minuto = minute();
          segundo = second();
          //outputtemp.print(hora+":"+minuto+",");     //faz a escrita do tempo no arquivo
          outputtemp.print(leitura+",");
          outputtemp.print(temp+"\n");    //faz a escrita da temperatura no arquivo
          outputtemp.flush();

          /************************INICIO ALARME*********************/
          if(temp>37){  //define a temperatura de disparo do alarme
          if ( player.position() == player.length() ) {
            player.rewind();
            player.play();
          }
          else {
```

```
            player.play();
           }
           //sendMail();
          }
          /************************FIM ALARME************************/


          /**********************************INICIO
INTERFACE*****************************************************************************/
          /***********PLANO DE FUNDO***********/
           size(1280,720);
           background(255); //Atualiza a imagem de fundo (backgroud) da interface
          //Carrega a fonte de texto a ser utilizada
           textFont(createFont("Arial Bold",18));
           textAlign(CENTER);

          fill(255);
          rect(width/2,height/16,width/2-width/30,height/2);
          for(j=1;j<10;j++){
          line(width/2,j*height/20+height/16,width-width/30,j*height/20+height/16);
          }
          line(width/2+width*14/90,height/16+height/20,width/2+width*14/90,height/16+height/2);
          line(width/2+width*28/90,height/16+height/20,width/2+width*28/90,height/16+height/2);
          fill(0);
          text("TABELA DE TEMPERATURA", width/2+width*3*14/180, height/16+height/30);
          text("Leitura", width/2+width*1*14/180, height/16+height/20+height/30);
          text("Hora", width/2+width*3*14/180, height/16+height/20+height/30);
          text("Temperatura(°C)", width/2+width*5*14/180, height/16+height/20+height/30);

          if(leitura<9){      // 8 leituras dispostas na tabela com informação do horário
              vetor_hora[flag-1] = hora;
              vetor_minuto[flag-1] = minuto;
              vetor_temperatura[flag-1] = temp;
            for(j=1;j<flag+1;j++){
              text(j, width/2+width*1*14/180, height/16+height/20+j*height/20+height/30);
              text(vetor_hora[j-1]+":"+vetor_minuto[j-1],                    width/2+width*3*14/180,
height/16+height/20+j*height/20+height/30);
              text(String.format("%.1f",vetor_temperatura[j-1]),             width/2+width*5*14/180,
```

```
height/16+height/20+j*height/20+height/30);}
                }
               else{
                for(j=0;j<7;j++){
                 vetor_hora[j]=vetor_hora[j+1];
                 vetor_minuto[j]=vetor_minuto[j+1];
                 vetor_temperatura[j]=vetor_temperatura[j+1];
                }
                 vetor_hora[7] = hora;
                 vetor_minuto[7] = minuto;
                 vetor_temperatura[7] = temp;
                 flag = leitura-7;
                 for(j=1;j<9;j++){
                   text(flag, width/2+width*1*14/180, height/16+height/20+j*height/20+height/30);
                   text(vetor_hora[j-1]+":"+vetor_minuto[j-1], width/2+width*3*14/180,
height/16+height/20+j*height/20+height/30);
                   text(String.format("%.2f",vetor_temperatura[j-1]), width/2+width*5*14/180,
height/16+height/20+j*height/20+height/30);
                   flag = flag+1;
                   //println("flag=",flag);}
                }

            fill(255);
            rect(width/2,height/2+height/16+4,width/2-width/30,height/2-height/16-10);
            fill(0);
            text("GRÁFICO DE TEMPERATURA", width/2+width*3*14/180,
height/2+height/16+height/30);
            flag++;
            leitura++;

            textFont(createFont("Arial Bold",45));
            textAlign(CENTER);
            fill(255);
            rect(2,2,256,120);
            image(termometro, 5,5);
            rect(105,7,148,110);
            fill(0);
            text(String.format("%.1f",temp), 179, 62);
```

```
///***********INICIO GRÁFICO***************/
textFont(createFont("Arial Bold",18));
textAlign(CENTER);
String[] data = loadStrings("Temperatura.csv"); //leitura dos dados do arquivo
float[] eixotempo  = new float[data.length-1];
float[] eixotemperatura = new float[data.length-1];

for (int i=0; i<data.length-1; i++) {
String[] tokens = data[i+1].split(","); //separação dos dados dos eixos x e y
eixotempo[i]  = Float.parseFloat(tokens[0]); //definição do eixo x
eixotemperatura[i] = Float.parseFloat(tokens[1]); //definição do eixo y
}

lineChart.setData(eixotempo,eixotemperatura);

// Formatação dos eixos
lineChart.showXAxis(true);
lineChart.showYAxis(true);
lineChart.setMinY(20);
lineChart.setMinX(1);

lineChart.setYFormat("##,##"); // define o eixo Y (Temperatura) com duas casas decimais
lineChart.setXFormat("###");   // define o eixo x como um valor inteiro

// formatações do gráfico a ser gerado
lineChart.setPointColour(color(180,50,50,100));
lineChart.setPointSize(5);
lineChart.setLineWidth(2);

lineChart.draw(width/2+2,height/2+50,width/2-width/30,height/2-height/16-15);
/*****************************FIM GRAFICO*****************************/

//translate(width/2, height/2+ajuste_y); //Posiciona o ponto 0 da interface no centro da
tela
//                    //Com ajuste de 14 pixels em y (height)
//rotate(radians((90/12)*(temp-20)-90)); //Pega o valor lido referendo ao potenciômetro,
//                    //Subtrai 90 desse valor, converte para radianos e
```

```
            //                        //Executa a função de rotação da tela
            //image(ponteiro, -(ponteiro.width/2), -ponteiro.height+(7)); //Posiciona o centro do furo do
ponteiro
            //                                  //No posição central do gráfico na imagem de fundo
            /***********************************************FIM
INTERFACE***********************************************/

          } //fim do if(i==7)

        }//fim do if (myPort.available() > 0)

          if ( millis() - lastTime > 70000 ) {   //faz a contagem do tempo para reinicializar a conexão
          //println( "do things every 70 seconds" );
          //println( "entrada:"+hour()+":"+minute()+":"+second());
           myPort.stop();  //corta a conexão
            while(myPort.active()==false){ //procura por nova conexão
             String portName = Serial.list()[2];
             myPort=new Serial(this, portName, 9600);
             }
          // println( "saida:"+hour()+":"+minute()+":"+second());
           lastTime = millis();
          }
      }
```

9 BIBLIOGRAFIA

ARDUINO. **Arduino**. Disponivel em: <https://www.arduino.cc/>. Acesso em: 11 Novembro 2015.

ARDUINO. **Arduino Nano User Manual**. [S.l.], p. 1-5.

ATMEL. **ATmega48A/PA/88A/PA/168A/PA/328/P [DATASHEET]**. Atmel. San Jose, p. 1-650. 2014.

BRIDGES, E.; THOMAS, K. Noninvasive Measurement of Body Temperature in Critically Ill Patients. **Critical Care Nurse**, v. 29, n. 3, p. 94-97, June 2009.

COSTA, C. M. A. **Técnicas de mensuração da temperatura corporal: uma especial atenção para as variações da temperatura da pele mensuradas por tomografia ao longo do dia**. Viçosa, Minas gerais, p. 12-28. 2012.

DALLAS SEMICONDUCTOR. Electronic Components Datasheet Search. **Alldatasheet**. Disponivel em: <http://www.alldatasheet.com/datasheet-pdf/pdf/58557/DALLAS/DS18B20.html>. Acesso em: 20 jun. 2015.

EXERGEN CORPORATION. Exergen - Temporal Scanner (TAT-5000). **Exergen Corporation**. Disponivel em: <http://www.exergen.com/ww/Exergen-TAT5000-Portuguese.pdf>. Acesso em: 15 maio 2015.

GARRISON, G. T.; JONHSTON, J. A.; PONCIROLI, K. M. Fever in Children. **St. Clair Pediatrics**.

HC. **HC Serial Bluetooth Products - User Instructional Manual**. [S.l.], p. 1-16.

INCOTERM. Baby Care Termômetro Clínico - Digital Auricular (TH809). **Incoterm - Soluções em medição**. Disponivel em: <http://www.incoterm.com.br/saude/29838+babycare+termometro+digital+auricular+infravermelho>. Acesso em: 15 maio 2015.

LUNA, A. J. H. D. O. **Abordagem da engenharia de requisitos em projetos de desenvolvimento de software para telessaúde/telemedicina**. Universidade Federal de Pernambuco - Centro de Informática. Recife, p. 26-29. 2008.

MINISTÉRIO DA SAÚDE. Atenção à Saúde do Recém-Nascido. **Guia para os Profissionais de Saúde**, Brasilia, v. 4, n. 1, 2011.

MURAHOVSCHI, J. A criança com febre no consultório. **Jornal da Pediatria**, v. 79, n. 1, 2003.

NASCIMENTO, V. M. **Gerenciamento de Risco em Projetos: Como**. Universidade Veiga de Almeida. Rio de Janeiro, p. 8-20. 2003.

OMRON HEALTHCARE BRASIL. Omron - Termômetro Digital Modelo MC-343F. **Omron Healthcare**, 2011. Disponivel em: <https://www.omronhealthcare.la/uploads/attachment/17942034fa6f0d86103454e7e87ee850c085a806033111-MC-343BR-IM-9493614-9A-pdf.pdf>. Acesso em: 15 maio 2015.

PROCESSING FOUNDATION. Processing. **Processing**. Disponivel em: <https://www.processing.org/>. Acesso em: 11 Novembro 2015.

SANT'ANNA, R. T.; CARDOSO, A. K.; SANT'ANNA, J. R. M. Aspectos Éticos e Legais da Telemedicina Aplicados a Dispositivos de Estimulação Cardíaca Artificial. **Reblampa**, v. 18, n. 3, p. 103-110, 2005.

TEXAS INSTRUMENTS. Texas Instruments. **Texas Instruments**, January 2015. Disponivel em: <http://www.ti.com/>. Acesso em: 15 maio 2015.