Universidade do Estado do Rio de Janeiro

Centro de Tecnologia e Ciências

Faculdade de Engenharia

Rafael Vida de Castro Rosario

Sistema para Monitoração de uma Embarcação não Tripulada

Rio de Janeiro

2013

Rafael Vida de Castro Rosario

Sistema para Monitoração de uma Embarcação não Tripulada

Projeto de graduação apresentado, como requisito parcial para obtenção do grau de Engenheiro Eletricista, à Faculdade de Engenharia, da Universidade do Estado do Rio de Janeiro.

Orientador: Prof. Dr. Tiago Roux de Oliveira

Coorientador: Prof. D.Sc. José Paulo Vilela Soares da Cunha

Rio de Janeiro

2013

CATALOGAÇÃO NA FONTE

UERJ / REDE SIRIUS / BIBLIOTECA CTC/B

P348 Rosario, Rafael Vida de Castro.
Sistema para monitoração de uma embarcação não tripulada / Rafael Vida de Castro Rosario. – 2013.
65f.

Orientador: Tiago Roux de Oliveira.
Projeto Final (Graduação) - Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.
Bibliografia p.50-54.

1. Engenharia eletrônica. 2. Aquisição de dados. 3. Sinais e sinalização. 4. Instrumentação. Oliveira, Tiago Roux de. II. Universidade do Estado do Rio de Janeiro. III. Título.

CDU 621.38+39

Rafael Vida de Castro Rosario

Sistema para Monitoração de uma Embarcação não Tripulada

Projeto de graduação apresentado, como requisito parcial para obtenção do grau de Engenheiro Eletricista, à Faculdade de Engenharia, da Universidade do Estado do Rio de Janeiro.

Aprovado em 11 de março de 2013.

Banca Examinadora:

_____

Prof. Dr. Tiago Roux de Oliveira (Orientador)

Faculdade de Engenharia - UERJ

_____

Prof. Dr. José Paulo Vilela Soares da Cunha

Faculdade de Engenharia - UERJ

_____

Prof. Dr. Jorge Luís Machado do Amaral

Faculdade de Engenharia - UERJ

Rio de Janeiro

2013

AGRADECIMENTOS

A Deus, princípio de tudo.

Aos meus pais, meus grandes amores, obrigado pelo carinho e compreensão.

Aos admiráveis orientadores Prof. José Paulo Vilela Soares da Cunha e Prof. Tiago Roux de Oliveira, exemplos de ética e dedicação.

Aos professores e funcionários do Departamento de Eletrônica e Telecomunicações da UERJ, pelo excelente convívio.

Aos colegas do curso de engenharia, pela troca de ideias e de experiências.

Aos amigos dos Buffalo's Engenharia UERJ, pois nem só de aulas se vive uma faculdade, mas de diversão também.

Eu sou patrão, não funcionário.

Menor do Chapa

RESUMO

ROSARIO, Rafael Vida de Castro. *Sistema para Monitoração de uma Embarcação não Tripulada*. 65f. Projeto Final (Graduação em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2013.

Este trabalho apresenta a construção de um sistema para monitoração de uma embarcação não tripulada, utilizando sensores para a medição das variáveis. Destinado à monitoração ambiental e de segurança, este projeto utiliza o microcontrolador Arduino para integrar os sensores à embarcação. Este projeto objetiva a análise das variáveis a serem medidas e a melhor forma de monitorá-las. Os dados da embarcação são disponibilizados na Internet através de uma integração entre as linguagens C, PHP e Arduino. As conclusões analisam a evolução do projeto, as dificuldades na alimentação do sistema e sugestões para trabalhos futuros.

Palavras-chave: USVs. Sistemas de monitoração. Aquisição de dados. Embarcação.

Teleoperação.

ABSTRACT

This work presents system construction to ship monitorization not shipped yet, using sensors to variables measurement. Designated to environmental monitorization and safety, this product uses microcontroller Arduino to integrate sensors to ship.This project aims variables analysis to be measured and greater ways of monitorizing them. Ship datas are available at Internet through integration among languages C, PHP and Arduino. Conclusions analize project evolution, difficulties in power system and suggestions to future works.

Keywords: GIS. Environmental. Protection areas. Geoinformation Ship. Teleoperation

LISTA DE FIGURAS E QUADROS

Figura 1.1: Embarcação utilizada no projeto. 13
Figura 1.2: Diagrama de blocos do sistema de aquisição. 13
Figura 2.1: Exemplo de motor utilizado no projeto. 16
Figura 3.1: Ligação Elétrica do Sensor de Corrente ACS756. 21
Figura 3.2: Diagrama do Circuito Sensor de Corrente ACS 756. 22
Figura 3.3: Esquema do sensor de inundação. 22
Figura 3.4: Diagrama do Circuito do Sensor de Inundação. 23
Figura 3.5: Diagrama do Circuito do Sensor de Umidade. 24
Figura 3.6: Diagrama de Blocos do LM35. 25
Figura 3.7: Diagrama do Circuito do LM35. 26
Figura 3.8: Diagrama de blocos do sensor ISO122JP. 27
Figura 3.9: Diagrama do Circuito do Amplificador de Isolação ISO122JP. 33
Figura 3.10: Diagrama do Circuito do Conversor DC/DC MEA1D0515SC. 34
Figura 4.1: Arduino. 36
Figura 4.2: Fluxo de dados. 38
Figura 5.1: Estrutura de um cabo USB. 42
Figura 5.2: Barra de ferramentas da IDE do Arduino. 44
Figura 5.3: Tela do programa “le_serial”. 45
Figura 5.4: Página da *Web* “index.php” que exibe as variáveis do barco. 46
Figura 5.5: Arquivo “sensores.txt”. 46
Figura 5.6: Arquivo “sensores.log”. 47
Quadro 2.1: Variáveis internas da embarcação a serem analisadas. 17
Quadro 3.1: Características técnicas do sensor LM35D. 25
Quadro 3.2: Características técnicas do amplificador ISO122JP. 28
Quadro 3.3: Dados Técnicos do Conversor DC/DC. 31
Quadro 3.4: Consumo de corrente de cada componente. 30
Quadro B.1: Orçamento. 65

LISTA DE SIGLAS

USV	*Unmanned Surface Vehicles*
ROV	*Remotely operated underwater vehicle*
DC	*Direct Current*
USB	*Universal Serial Bus*
HTML	*HyperText Markup Language*
PHP	*Hypertext Preprocessor*
Ajax	*Asynchronous JavaScript and XML*
CSS	*Cascading Style Sheets*
GND	*Ground*
IP	*Internet Protocol*
IDE	*Integrated Development Environment*
CSV	*Comma-Separated Values*
SQL	*Structured Query Language*

SUMÁRIO

1 INTRODUÇÃO 11
1.1 Sistema de monitoração 12
1.2 Objetivos 13
1.3 Estrutura do Texto 14
2 ANÁLISE DAS VARIÁVEIS 15
2.1 Variáveis do barco 15
2.2 Variáveis externas 18
3 SENSORES 20
3.1 Sensor de corrente 21
3.2 Sensor de inundação 22
3.3 Sensor de umidade 23
3.4 Sensor de temperatura 24
3.5 Sensor de tensão 26
4 AQUISIÇÃO DE DADOS 35
4.1 O microcontrolador Arduino 35
4.2 Apresentação dos dados 36
4.2.1 Aquisição de dados 39
4.2.2 Comunicação com o sistema de aquisição 40
4.2.3 Apresentação final dos dados 40
5 TESTES COM O ARDUINO 42
5.1 Resultados experimentais 44
5.2 Telas de programas 44
6 CONCLUSÃO 48
REFERÊNCIAS 50
APÊNDICE A - Programas 55
A.1 - Aquisição de dados 55
A.2 - Comunicação com o sistema de aquisição 58
A.3 - Apresentação final dos dados 60
APÊNDICE B - Orçamento 65

1 INTRODUÇÃO

A utilização de embarcações para monitoração tem principalmente dois grandes motivos básicos: ambiental e segurança.

Por se tratar de uma embarcação autônoma, seu uso é mais prático, rápido, eficiente e menos intrusivo que a utilização de monitoração convencional – com uso de especialistas no local.

> Para estudar os intrincados processos associados ao meio ambiente, em especial os mares, rios e atmosfera é necessário dispor de uma complexa gama de informações e ferramentas de aquisição de dados. Muitos destes dados sobre os fenômenos naturais são obtidos através de observações remotas com o uso de satélites ou através de observações experimentais numa escala de tempo muito grande, proporcionando dados válidos e úteis, porém imprecisos e onerosos. As coletas de dados *in loco* são necessárias para adicionar precisão e também para a calibração dos instrumentos de monitoração remota (Higinbotham et al., 2009 apud CUNHA, 2010, p. 4).
>
> Nos mares, rios e lagos, medições *in loco* têm sido usualmente coletadas por boias e embarcações de pesquisa. Contudo, a manutenção e operação de embarcações específicas para este fim têm custo muito alto. Por outro lado, as boias têm o inconveniente de serem fixas ou "nômades" (conduzidas pelas correntes e ventos) e, assim sendo, sua região de operação é muito limitada ou inflexível (CUNHA, 2010, p. 4).
>
> Os avanços da automação e o interesse em um sistema global de observação dos mares, rios e oceanos têm estimulado um progresso considerável na área de veículos de pesquisa não tripulados, incluindo aviões, submarinos e embarcações de superfície. Estes últimos, também denominados USV (*unmanned surface vehicles*), são o foco deste Projeto de Pesquisa (CUNHA, 2010, p. 4).

Dentre as aplicações de USVs estão "estudos marinhos e fluviais, tais como: batimetria, medição de correntes, medição de temperatura, oceanografia física, observação da vida marinha e biodiversidade" (CUNHA, 2010, p. 5), além dos "estudos atmosféricos sobre a qualidade do ar, insolação, presença de gases, temperatura, velocidade do vento e índices pluviométricos" (CUNHA, 2010, p. 5), e o "estudo de impactos ambientais utilizando um USV auto-ancorado [sic] que permite a busca e análise de amostras no local e de forma contínua" (CUNHA, 2010, p. 5).

Os USVs ainda são utilizados para "busca submarina e de superfície [com] câmeras subaquáticas e sonares permitindo a visualização do fundo do mar tendo-se em vista a busca de destroços e objetos suspeitos" (Leonessa et al. 2003; Betram e Veers, 2006; Wang et al. 2009 apud CUNHA, 2010, p. 5), a "inspeção de cascos de navios ancorados, com o auxilio de um ROV [(*Remotely operated underwater vehicle*)], podendo (...) ser feita a inspeção em casos que apresentam riscos para mergulhadores" (Leonessa et al. 2003; Betram e Veers, 2006 apud CUNHA, 2010, p. 5), a "vigilância costeira contínua e sem a presença humana" (CUNHA, 2010, p. 5), a caça de "minas subaquáticas em regiões que apresentem muito risco

para as tripulações de embarcações, quando o uso de USVs seria muito conveniente" (Leonessa et al. 2003; Betram e Veers, 2006; Djapic e Nad, 2010 apud CUNHA, 2010, p. 5). Além do mais, "os USVs podem ser usados como alvos em testes de novos armamentos e treino de tripulações da marinha de forma muito realista" (Betram e Veers, 2006 apud CUNHA, 2010, p. 5).

A faixa de fronteira do território nacional brasileiro com os outros países da América do Sul – excluindo-se apenas Chile e Equador - é estimada na ordem de 16.688 km de largura (LNCC, 2012). Também é através dessa larga extensão de terra que a segurança nacional é extremamente exposta. O Brasil viu-se obrigado a exercer vigilância constante em seus limites territoriais depois que percebeu que são pelas fronteiras terrestres que entram ilegalmente no país, durante todo o ano, armas, drogas e contrabando.

A realidade é que, atualmente, as forças de policiamento são insuficientes para atender a uma área de dimensões continentais, tornando assim a vida de contrabandistas, traficantes e vendedores de armamento mais fácil. Além disso, para aumentar a dimensão do problema, existe o fato de que a entrada de mercadoria é feita em locais de difícil acesso e circulação, como rios e riachos em regiões remotas e inabitadas da fronteira.

Além desses fatos, as máfias que atuam na fronteira, possuem armamento muitas vezes exclusivo das forças militares e tendem sempre a reagir quando o confronto é direto. Como são poucas a unidades de policiamento, quando as mesmas se defrontam com uma violação da fronteira, estas são obrigadas a recuar ou pedir reforço – dando tempo suficiente para os contraventores lograrem êxito na sua empreitada.

Com todas essas questões acima apresentadas, a utilização de uma embarcação não tripulada para monitoramento da segurança da fronteira é extremamente válida, pois permite uma monitoração não intrusiva - ou seja, não é possível detectar com facilidade a monitoração, portanto, o Estado ganha o elemento surpresa no combate, há a possibilidade de se fazer um histórico e realizar um banco de dados e constitui uma ferramenta de combate à criminalidade como um todo.

1.1 Sistema de monitoração

Para que uma embarcação navegue de forma autônoma é necessário um bom funcionamento dos seus componentes, compreendendo motores, bateria, entre outros.

Entretanto, esses itens podem apresentar falhas que se não corrigidas rapidamente, podem resultar em perdas muito maiores, como a própria embarcação.

Levando em consideração a realidade apresentada acima, faz-se necessária a monitoração das variáveis de funcionamento do barco, com a finalidade de assegurar o bom funcionamento e a segurança da sua operação.

O casco utilizado para a embarcação será o caiaque Brudden Hunter Fishing, que é do tipo moldado em plástico (SCHULTZE, 2012, p. 30). Na Figura 1.1 segue a embarcação utilizada no projeto.

Figura 1.1: Embarcação utilizada no projeto.

Na Figura 1.2 segue o diagrama de blocos completo do sistema, que será mais bem entendido ao longo dos capítulos.

Figura 1.2: Diagrama de blocos do sistema de aquisição.

1.2 Objetivos

O objetivo deste Projeto de Graduação é analisar as variáveis a serem monitoradas, definir a melhor forma para monitorar esses dados e a melhor forma de apresentar essa informação. Em seguida, será feita a escolha dos componentes, a busca de fornecedores,

cotação de preços e compra dos componentes requeridos para a construção deste Projeto de Graduação. Por fim, será realizado o estudo do funcionamento dos mesmos, montagem, ensaio e testes com o protótipo.

Para verificar o desempenho das propostas em condições reais, será usado um catamarã de porte médio para experimento em águas calmas.

1.3 Estrutura do Texto

A organização deste Projeto de Graduação foi estabelecida como se segue. No Capítulo 2 é feita a análise das variáveis do barco e variáveis externas a serem monitoradas. O Capítulo 3 apresenta a escolha dos sensores mais adequados, bem como os circuitos para que os mesmos funcionem de maneira correta.

No Capítulo 4 é descrito como será feita a aquisição dos dados através do Arduino, incluindo toda a programação necessária para que o sistema funcione. No Capítulo 5 são exibidos os resultados experimentais do sistema proposto.

No Capítulo 6 são discutidas conclusões sobre o sistema desenvolvido neste Projeto de Graduação.

2 ANÁLISE DAS VARIÁVEIS

A análise das variáveis presentes no barco é de fundamental importância para o projeto de embarcações não tripuladas, porque a mesma serve como uma forma de se fazer a prevenção de problemas no barco. Esse monitoramento é uma forma de monitorar e prevenir falhas, que pode evitar perdas e danos mais significativos ao equipamento que é frágil e de alto custo financeiro.

Com a finalidade de tornar mais fácil a análise das variáveis a serem dimensionadas no barco, foi feita a divisão das variáveis em dois tipos distintos, as variáveis do barco e as externas.

2.1 Variáveis do barco

No barco, nós temos a presença dos motores/propulsores e o microcomputador de controle, juntamente com a bateria dentro de uma caixa fechada e a prova d'água. Com essas informações em mente, os principais problemas possíveis na embarcação seriam o mau funcionamento ou o superaquecimento dos motores, o superaquecimento e a umidade do microcomputador e da bateria, a inundação no casco e a estabilidade do barco. Para cada problema, foi pensada uma solução.

Como o motor do barco se trata de um motor elétrico Marine Sports Phantom 44 lbs, de 12 V por 30 A (SCHULTZE, 2012, p. 45, 46, 48), cuja foto é mostrada na Figura 2.1, a absoluta maioria dos problemas que possam acontecer nele, são primeiramente detectadas pelas variáveis de corrente e tensão. Como no barco existem 3 motores idênticos, serão 6 variáveis a serem analisadas nessa parte.

Figura 2.1: Exemplo de motor utilizado no projeto.

A água é um dos melhores refrigeradores que conhecemos no nosso planeta e o fato dos motores estarem dentro da mesma, nos leva a reconsiderar se o superaquecimento dos motores é realmente um problema, já que com o barco em movimento, a tendência da temperatura do motor é se aproximar a temperatura d'água, o que é no máximo 30°C - ou seja, com uma boa margem de segurança de operação do motor.

Esse problema se torna mais irrelevante ainda se levarmos em consideração que uma situação desse porte, seria ordinariamente detectada pelas variáveis de corrente e tensão. Por fim, os motores são lacrados e fechados, tornando mais difícil ainda conseguir determinar a sua temperatura. Com todas essas premissas, verificamos que o superaquecimento do motor não é realmente um problema a ser considerado. Porém, mesmo assim, como o barco também será utilizado para monitoração ambiental, a temperatura da água e do ar também serão variáveis a serem medidas.

O superaquecimento do microcomputador é um problema real, visto que, a mesma libera calor em sua operação normal. Acredita-se que a caixa de proteção do microcomputador consiga se resfriar com a operação do barco – já que o mesmo em

movimento ajuda na troca de calor – mas existe, de fato, a necessidade de se avaliar a temperatura dentro da caixa. Com esse fato, teremos uma variável de temperatura na caixa.

A umidade dentro da caixa onde fica o microcomputador é um problema considerável, já que aparelhos eletrônicos não funcionam bem com umidade e até mesmo podem estragar devido à oxidação que é intensificada em exposição à salinidade d'água do mar – que é uma das possíveis áreas de atuação do barco. Dessa forma, temos mais uma variável a ser medida no barco.

A bateria será responsável por alimentar os motores do barco. Sendo assim, é interessante verificarmos o bom funcionamento dela e medirmos sua tensão e corrente.

Com esses dados, podemos fazer cruzamento das tensões e correntes dos três motores e descobrir a presença de alguma ligação com alta impedância ou ainda tensões e correntes sendo desviadas e etc.

Além disso, se a bateria descarregar ou apresentar qualquer outro tipo de problema, conseguiremos rastrear através dessas duas variáveis. Sendo assim, as variáveis de tensão e corrente também serão aferidas.

A inundação do casco é um dos problemas mais primários e óbvios da navegação porque levaria a uma perda imediata da embarcação. Portanto, é necessária uma variável que faça a medição do nível de água dentro do casco.

O Quadro 2.1 mostra um resumo de todas as variáveis internas da embarcação a serem analisadas.

Quadro 2.1: Variáveis internas da embarcação a serem analisadas.

Variável	**Local**
Tensão	Bateria e motores
Corrente	Motores
Temperatura	Caixa, ar e água
Umidade	Caixa
Inundação	Casco

2.2 Variáveis externas

Dependendo do ambiente onde a embarcação fará o monitoramento e a finalidade do mesmo – como apresentado no Capítulo 1 – as variáveis a serem medidas poderão mudar drasticamente.

O escopo deste Projeto de Graduação se resume em analisar as variáveis internas que garantam o bom funcionamento do barco, deixando as outras a cargo de um projeto que seja mais específico com a utilização final do catamarã.

Com a finalidade apenas de produzir material informativo para um estudo mais aprofundado para fim de monitoração, apresentaremos abaixo as principais variáveis a serem medidas no caso ambiental e no de segurança, que foram pensadas ao longo do desenvolvimento deste Projeto de Graduação e que foram deixadas a parte pelos motivos expostos acima:

- *Ambiental*

- Acidez da água – Para o desenvolvimento correto e natural do ecossistema de qualquer lago, rio, mar e oceano, é necessário que a água possua o pH adequado.

- Direção e velocidade do vento – São variáveis importantes para aplicações meteorológicas.

- *Segurança*

- Câmera de vídeo – É a forma mais simples de se conseguir observar o que acontece na região monitorada.

- Sensor de proximidade – O equipamento de monitoração custa caro e deve ser protegido contra contraventores. Um sensor de aproximação poderia evitar que um corpo externo e estranho se aproximasse a uma distância perigosa do barco.

- Microfone – A utilização desse dispositivo sonoro poderia alertar sobre qualquer atividade suspeita de uma forma antecipada a própria imagem do local.

3 SENSORES

Através do estudo realizado anteriormente sobre quais variáveis são plausíveis e interessantes para monitoração do barco e do ambiente – lembrando que isso varia de acordo com as possíveis aplicações – chegou-se à conclusão que a melhor maneira de medir as variáveis com agilidade, confiança e custo razoável é através do uso de sensores.

Um sensor pode ser definido como um dispositivo que recebe um estímulo e responde com um sinal elétrico. A finalidade de um sensor é responder a algum tipo de entrada física (estímulos) e convertê-lo num sinal elétrico. Podemos dizer que ele traduz um valor geralmente não elétrico em um valor elétrico. O sinal de saída pode ser canalizado, amplificado e modificado por dispositivos eletrônicos, podendo estar sob a forma de tensão, corrente ou carga. Este ainda pode ser descrito em termos de amplitude, frequência, fase ou código digital. Esse conjunto de características é chamado formato do sinal da saída (FRADEN, 2003, p. 2).

O sensor deve ser distinguido do transdutor. Este último é um conversor de um tipo de energia em outro, enquanto o sensor converte qualquer tipo de energia em elétrica apenas. Um exemplo de transdutor é um alto-falante, que converte um sinal elétrico em um campo magnético variável e, consequentemente, em ondas acústicas (FRADEN, 2003, p. 3).

Os transdutores podem ser parte de sensores complexos. Por exemplo, um sensor químico pode ter uma parte que converte a energia de uma reação química em calor (transdutor) e outra parte que converte o calor em um sinal elétrico. A combinação dos dois forma um dispositivo que produz um sinal elétrico em resposta a uma reação química. Muitos sensores complexos incorporam pelo menos um tipo de sensor e certo número de transdutores (FRADEN, 2003, p. 3).

Ainda com base na averiguação das variáveis de interesse para monitoração, podemos fazer uma conexão direta entre elas e os tipos de sensores que será preciso usar. Por exemplo, se precisarmos monitorar corrente, utilizaremos um sensor de corrente. Sendo essa mesma ideia aplicada para todos os tipos de variáveis que serão medidas na embarcação.

Desta forma, cada variável será comentada nas seções a seguir.

3.1 Sensor de corrente

A ideia inicial no projeto é ter 4 sensores de corrente – um para cada motor e um para a bateria do barco. Entretanto, como o da bateria teria que aguentar a soma da corrente dos 3 motores, seria necessário utilizar um sensor que suportasse uma corrente da ordem de 200 A, considerando os transitórios de corrente. Porém, um sensor que aguente essa quantidade de amperagem não é um sensor barato. Portanto, a ideia de utilizar 4 sensores foi abandonada, passando-se a utilizar somente 3 – um para cada motor. Para medir a corrente de cada motor será usado um sensor de corrente de atua segundo o Efeito *Hall*. O sensor mais indicado neste caso é o ACS756SCA-100B-PFF-T (COUTINHO, não publicado).

A ligação elétrica do sensor de corrente é mostrada na Figura 3.1.

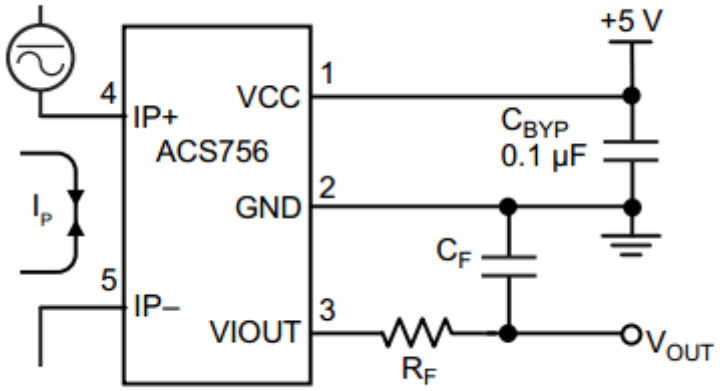

Figura 3.1: Ligação Elétrica do Sensor de Corrente ACS756.

Como esse sensor é bipolar, a tensão na saída quando a corrente for 0 A é de $V_{cc}/2$, sendo a tensão maior que esse valor para correntes positivas e menor para correntes negativas. O capacitor de desacoplamento C_{BYP} é usado para eliminar o efeito de ruído de alta frequência vindo da fonte de +5 V do Arduino (Capítulo 4), já que, com uma variação da tensão de alimentação, teríamos uma variação da tensão de saída, além de também alterar a sensibilidade do sensor (ALLEGRO MICROSYSTEMS, 2011, p. 4), desajustando totalmente o mesmo.

O resistor R_F e o capacitor C_F atuam como um filtro passa-baixas, que suaviza o ruído da saída. Através de testes realizados, concluiu-se que o ruído sem esse filtro é pequeno, portanto esse filtro é opcional (COUTINHO, não publicado).

O circuito completo do sensor, incluindo seu circuito de proteção (COUTINHO, não publicado), é mostrado na Figura 3.2.

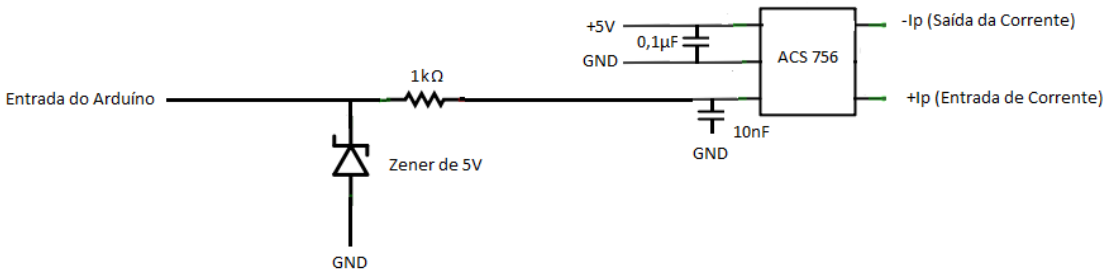

Figura 3.2: Diagrama do Circuito Sensor de Corrente ACS 756.

3.2 Sensor de inundação

O barco e o equipamento presente dentro do mesmo possuem um alto custo, portanto não podemos correr o risco de perdê-los através de um dos acidentes mais comuns na navegação, o naufrágio. Dessa forma, será feita a medição constante do nível de água no casco.

Para medir o nível de água, será construído um sensor com a finalidade de diminuir o custo do projeto. Para tal, será utilizada a técnica de um circuito de alarme em aberto e com as duas pontas conectadas em pontos extremamente opostos de uma esponja vegetal, como pode ser visto na Figura 3.3. Como a esponja seca não é um bom condutor, esse circuito ficará em aberto e o alarme não será acionado.

Figura 3.3: Esquema do sensor de inundação.

A esponja absorve rapidamente água, então, se no casco existir uma quantidade de água capaz de fazer a condução elétrica entre as duas extremidades, o circuito se fechará e dessa forma entenderemos que o nível de água dentro do barco é preocupante para o seu funcionamento.

A ligação elétrica do sensor de inundação, junto com seu circuito de proteção (COUTINHO, não publicado) é mostrada na Figura 3.4.

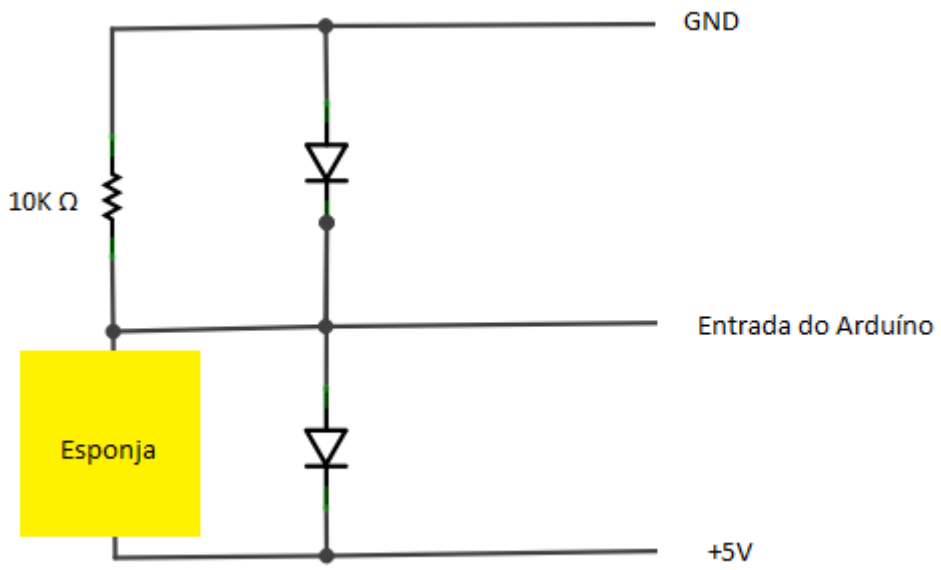

Figura 3.4: Diagrama do Circuito do Sensor de Inundação.

3.3 Sensor de umidade

A umidade dentro da caixa de controle deve ser baixa, porque lá dentro está o microcomputador do barco com diversos componentes eletrônicos que oxidam facilmente em contato com umidade, ainda mais na presença de umidade marítima. O sensor mais indicado é o HIH-4000 (COUTINHO, não publicado).

Sua ligação elétrica é bem simples – apenas três pinos. Para eliminar o efeito de ruído de alta frequência vindo da alimentação, colocamos um capacitor de *bypass* entre a alimentação positiva de 5 V e o GND (*Ground*) do circuito. De acordo com o manual do

fabricante, é recomendado utilizar um resistor de 80 kΩ como carga (HONEYWELL, 2010, p. 5), porém como este não é um valor comercial, foi utilizado um resistor de 82 kΩ.

Na Figura 3.5 é mostrada a ligação completa do circuito do sensor de umidade.

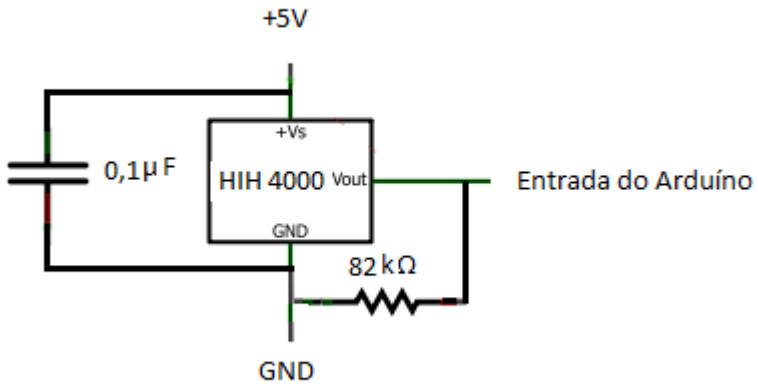

Figura 3.5: Diagrama do Circuito do Sensor de Umidade.

3.4 Sensor de temperatura

Hoje em dia, existe a disponibilidade de diversos tipos de sensores de temperatura, que vão desde os termistores, termopar, termoressistores, eletrônicos, pirômetros, dentre outros. Entretanto, estima-se que talvez nenhum dos citados anteriormente seja de tão simples manuseio e exija tão poucos aparatos eletrônicos para que funcione, quanto o modelo LM35, da Texas Instruments. O seu circuito usual é bastante simples, necessitando apenas do sensor propriamente dito, e de uma interface que realize a leitura do sinal, quem sabe até mostrando um valor de temperatura diretamente em um visor ou *display* ou até mesmo disparando algum elemento eletrônico como, por exemplo, um transistor quando a situação for apropriada. (CRESPI E CERON, 20??).

Esse sensor apresenta na sua saída uma tensão linear relativa à temperatura que ele afere no momento da medição. Ele funciona com a relação de para cada 1° Celsius medido na entrada, ele apresentará na saída o incremento de 10 mV.

Assim, o LM35 tem uma vantagem sobre os sensores lineares calibrados em Kelvin, já que o usuário não é obrigado a fazer a conversão para Celsius. Ele não requer nenhum tipo de calibração ou aparato para fornecer precisões típicas de $^{1}/_{4}$º C a temperatura ambiente. Sua

impedância de saída é baixa, possui saída linear, não precisa de uma calibração e possui uma faixa de operação de -55º C a 150º C – atendendo dessa forma, a necessidade deste Projeto de Graduação. Com esta faixa de operação, a amplitude do seu sinal de saída pode variar entre 0,55 V e 1,5 V. O auto aquecimento do sensor não é um problema, visto que ele drena apenas 60 µA para sua alimentação, gerando no máximo 0,1° C – o que é irrelevante para a nossa aplicação (NATIONAL SEMICONDUCTOR CORPORATION, 2000, p. 2).

Na Figura 3.6 é mostrado o diagrama de blocos do LM35.

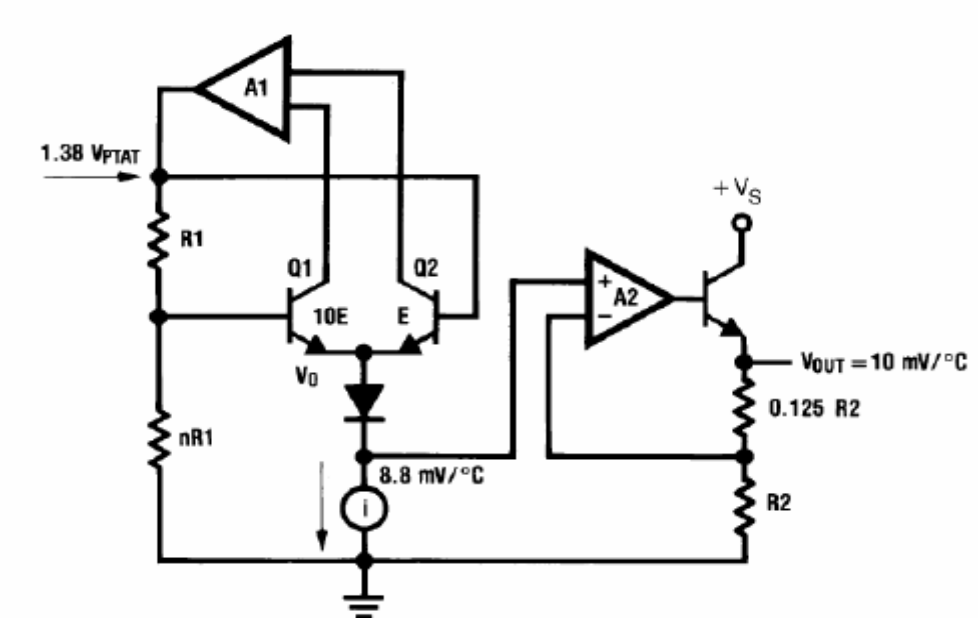

Figura 3.6: Diagrama de Blocos do LM35.

No Quadro 3.1 é mostrado um resumo das características técnicas do sensor LM35D.

Quadro 3.1: Características técnicas do sensor LM35D.

Característica	**Valor**
Precisão	±0,6 ºC
Sensibilidade	10 mV/ºC
Corrente quiescente	56 µA
Coeficiente de temperatura da corrente quiescente	0,39 µA/ºC
Faixa de operação	-55 a 150 ºC

Apesar de ser um sensor de fácil ligação, foi notado durante os testes (COUTINHO, não publicado) um ruído demasiado, certas vezes com oscilações de 2º C, que com certeza não se referiam à variação real de temperatura. Diversas alternativas, como o filtro média-móvel (ALCÂNTARA, 2008) e filtro passa-baixas foram utilizadas, porém nenhuma delas reduziu o ruído do sistema satisfatoriamente.

Então, após averiguarmos o manual do fabricante, concluiu-se que o LM35 pode sofrer interferência de fontes eletromagnéticas intensas, tais como relés, ondas de rádio, transitórios, etc., já que seu circuito interno pode se comportar como uma antena. Para melhores resultados, é recomendado utilizar um amortecedor RC com um resistor de 75 Ω e um capacitor de 1 µF na saída do sensor (NSC, 2000, p. 7). Como não há resistor comercial com esse valor, ele foi substituído por um de 82 Ω. Sendo assim, a ligação elétrica do sensor de temperatura é mostrada na Figura 3.7.

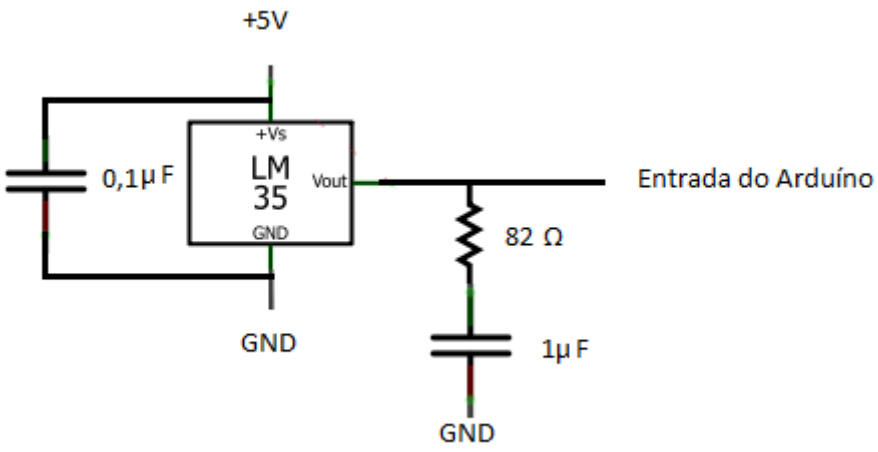

Figura 3.7: Diagrama do Circuito do LM35.

Para os sensores de temperatura da água e do ar, faz-se necessário um circuito de proteção (COUTINHO, não publicado).

3.5 Sensor de tensão

A maioria dos sensores usados transformará a grandeza a ser medida em tensão. Neste caso, como queremos medir diretamente a tensão – seja nos motores ou baterias, etc. - não será necessário o uso de nenhum tipo adicional de sensor.

Entretanto, para não ocorrer erro durante a medição e nem o comprometimento do funcionamento do controlador, não podemos deixar que a ligação elétrica fosse feita diretamente na placa do mesmo, já que os equipamentos a serem monitorados (motores, baterias) devem estar eletricamente isolados do sistema de aquisição[1].

Portanto, utilizaremos quatro amplificadores de isolação (também conhecidos como isoladores galvânicos), um para a bateria e um para cada motor, de modo a garantir a segurança do equipamento eletrônico, assim como a confiabilidade dos dados mensurados.

O amplificador de isolação escolhido foi o ISO122JP da TEXAS INSTRUMENTS (COUTINHO, não publicado). Este sensor foi escolhido porque, dentre os disponíveis no mercado, é bem robusto, apesar de ser o mais barato, e atende aos requisitos deste Projeto de Graduação. Seu diagrama de blocos é mostrado na Figura 3.8.

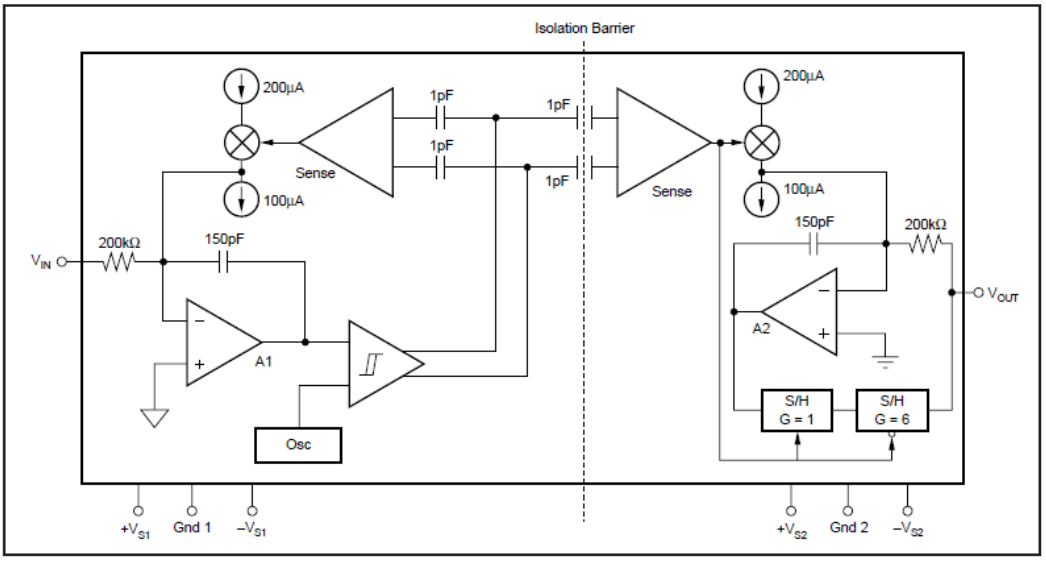

Figura 3.8: Diagrama de blocos do sensor ISO122JP.

Este amplificador tem a entrada e saída isoladas galvanicamente por dois capacitores de 1 pF. A entrada é modulada e transmitida digitalmente através da barreira capacitiva. O sinal é recebido do outro lado, convertido em tensão analógica, e removida a componente oscilante inerente à demodulação (BURR-BROWN, 1993, p. 6).

O amplificador A1 (Figura 3.8), integra a diferença entre a corrente de entrada (V_{IN} / 200 kΩ) e a fonte de corrente chaveada de ±100 µA. Para exemplificar, digamos que a tensão de entrada seja de 0 V. A saída do integrador será uma rampa, que crescerá até o limiar do

[1] Comunicação pessoal de J.P.S.V. Cunha, em 2011.

Schmitt Trigger. Quando isto acontecer, o amplificador Sense (Figura 3.8) e o *Schmitt Trigger* forçarão a fonte de corrente a mudar de polaridade, gerando uma rampa negativa na saída do integrador, resultando numa onda triangular com *duty cycle* de 50%. O oscilador interno força a corrente a chavear a 500 kHz (BB, 1993, p. 6).

O amplificador Sense (Figura 3.8), do outro lado detecta as transições do sinal através da barreira capacitiva e direciona a fonte de corrente chaveada ao integrador A2 (Figura 3.8). O estágio de saída balanceia o ciclo de trabalho da corrente modulada contra a corrente de realimentação através do resistor de 200 kΩ, resultando num valor médio da tensão de saída igual ao da tensão de entrada (BB, 1993, p. 6).

O ISO122JP possui uma tensão de *offset* de 50 mV – que não representará problema para as nossas medições que são na escala de Volt. Seu limite de banda é de 50 kHz – o que é indiferente, visto que utilizaremos somente corrente DC (*Direct current*). Sua isolação de tensão é de 1500 Vrms – o que garante uma boa faixa de segurança para nossa utilização. Seu ganho não linear é de 0,01% - o que é irrelevante, visto que não precisaremos do ganho do mesmo e somente da sua capacidade de isolação. Sua tensão de alimentação é de ±12 V até ±18 V, porém o ideal é que seja alimentado com ±15V (BB, 1993, p. 2). No Quadro 3.2 é mostrado um resumo das características técnicas do amplificador ISO122JP.

Quadro 3.2: Características técnicas do amplificador ISO122JP.

Característica	**Valor**
Impedância da barreira	$10^{14}\ \Omega \parallel 2$ pF
Corrente de fuga a 60 Hz	0,18 μA_{RMS}
Ganho	1
Erro de ganho	±0,05 %
Ruído	4 $\mu V/Hz^{1/2}$
Faixa de operação	-12,5 a 12,5 V
Resistência de entrada	200 kΩ
Corrente de saída	± 15 mA
Ripple da tensão de saída	20 mV_{PP}
Largura de banda	50 kHz
Tensão de alimentação	±12 a ±18 V
Corrente quiescente do lado da entrada	± 5 mA
Corrente quiescente do lado da saída	± 5,5 mA
Temperatura de operação	-25 a 85 ºC

O fato de este amplificador de isolação necessitar de uma alimentação de ± 15 V mostrou-se um problema, visto que só teremos como fonte de alimentação a bateria com +12 V e o controlador que fornece uma saída de +5 V.

A solução seria utilizar um conversor DC/DC. Os conversores DC/DC são circuitos eletrônicos que convertem uma tensão ou corrente contínua, que tem uma determinada amplitude, em outra tensão ou corrente contínua com outra amplitude diferente (MOHAN, 2002).

A primeira solução pensada foi ligar o conversor na bateria. Porém, essa ideia foi prontamente descartada, pois, se a bateria fosse usada para alimentar o conversor, caso ela começasse a descarregar, haveria um problema na alimentação dos mesmos, e

consequentemente na alimentação dos amplificadores, fazendo com que as medidas das tensões da bateria e dos motores sejam falhas.

Sendo assim, a decisão foi alimentar o conversor DC/DC com os 5 V do Arduino (Capítulo 4), que por sua vez é alimentado pela porta USB (*Universal Serial Bus*), não havendo mais o problema de descarregamento da bateria, pois o computador seria o último componente a falhar no pior dos casos.

Sabendo que tipo de componente seria preciso, foi buscado um modelo que se adequasse à exigência deste Projeto de Graduação. Nesse ponto, houve grande dificuldade de se achar um conversor DC/DC que possuísse um valor razoável e que não estivesse fora de linha – foram encontrados muitos conversores que não eram recomendados nem pelos vendedores para projetos atuais, por se tratarem de modelos com tecnologia defasada.

A fim de manter o isolamento galvânico, deve-se usar um conversor DC/DC com isolação entre sua entrada e a sua saída para alimentar o estágio de entrada e outro para alimentar o estágio de saída de cada amplificador de isolação. Um único conversor DC/DC pode alimentar todos os estágios de saída de todos os amplificadores de isolação, uma vez que as saídas desses amplificadores serão ligadas ao mesmo terra do conversor A/D.

Como era sabido que os conversores DC/DC geralmente consomem muita corrente, então, se tornou uma preocupação o fato de a porta USB do computador só fornecer 500 mA. No Quadro 3.3 é mostrado o consumo de corrente de todos os componentes do projeto. Vários conversores foram encontrados, mas todos consumiam muita corrente.

Quadro 3.3: Consumo de corrente de cada componente.

Componente	**Quantidade**	**Corrente por unidade (mA)**	**Corrente total (mA)**
Arduino	1	10	10
LM35D	3	<1	<1
ACS756	4	10	40
HIH-4000	1	0,5	1
MEA1D0515SC	4	45	180
MEA1D0515SC	1	134	134
Esponja	1	0,5	0,5
Total			**365**

Para escolher o conversor adequado, primeiro foi necessário comprar o amplificador de isolação e testá-lo, para verificar qual alimentação seria utilizada (ele pode ser alimentado de ± 4,5 a ± 18 V), e qual o consumo real de corrente. Após diversos testes (COUTINHO, não publicado), o conversor escolhido foi o MEA1D0515SC, da Murata Power Solutions, cujas especificações são descritas no Quadro 3.4, as quais se encaixavam nas requeridas.

Quadro 3.4: Dados Técnicos do Conversor DC/DC.

Característica	**Valor**
Tensão de isolamento	1 kV
Potência de saída	1 W
Tensão de entrada	4,5 V a 5,5 V
Tensão de entrada (nominal)	5 V
Corrente de entrada	225 mA
Número de saídas	2
Tensão de saída (Canal 1)	15 V
Corrente de saída (Canal 1)	33 mA
Tensão de saída (Canal 2)	- 15 V
Corrente de saída (Canal 2)	- 33 mA

Como a tensão da bateria pode atingir mais de 12 V em certas condições, além de futuramente haver a possibilidade de trocá-la, juntamente com os motores, por modelos de 24 V, e o amplificador de isolação ISO122JP só responde até tensões de 12,5 V, tornou-se necessário atenuar o sinal de tensão antes que o mesmo chegasse ao amplificador. Após diversos testes (COUTINHO, não publicado), concluiu-se que a melhor forma seria utilizar um resistor em série com a entrada, já que o ISO122JP possui uma impedância de entrada de 200 kΩ, como pode ser visto na Figura 3.8. A equação abaixo mostra o cálculo do resistor, onde R_{in} é a impedância de entrada do amplificador, R é a resistência que será colocada em série com a entrada, e V é a tensão máxima a ser lida:

$$V_{in} = \frac{R_{in}}{R_{in} + R}\ x\ V$$
$$5 = \frac{200}{R + 200}\ x\ 30$$
$$5R + 1000 = 6000$$
$$5R = 5000$$
$$R = 1000\ k\Omega$$

(3.1)

Sendo assim, a resistência usada para atenuar a tensão da bateria e dos motores será de 1 MΩ, que realizará uma atenuação de 6 vezes.

Como o amplificador não gera na saída um sinal de tensão contínuo, e sim uma onda triangular com valor médio igual à tensão da entrada com frequência de 500 kHz e *ripple* de 20 mV_{PP} (BB, 1993, p. 2), seria necessário utilizar um filtro passa-baixas na saída do amplificador para remover essa componente oscilante.

Para projetar o filtro, levamos em consideração que, como o conversor A/D do Arduino (Capítulo 4) possui tensão de referência de 5 V e 10 bits (ATMEL, 2012, p. 377), a resolução do mesmo é de aproximadamente 5 mV ($5 / 2^{10}$). Sendo assim, temos que realizar um filtro em que, na frequência de 500 kHz, o *ripple* seja muito menor que 5 mV, para não haver influência na medição. Ou seja, haverá uma atenuação de 40 vezes (20 mV / 0,5 mV) em 500 kHz. A equação abaixo mostra o cálculo da frequência de corte do filtro:

$$\frac{f_c}{500} = \frac{1}{40}$$
$$40 f_c = 500$$
$$f_c = 12{,}5\ kHz$$

(3.2)

Para realizar o filtro, escolhemos um resistor arbitrário de 1 kΩ, e o capacitor foi calculado a fim de obter a frequência de corte desejada, como mostra a equação abaixo:

$$f_c = \frac{1}{2\pi RC}$$
$$12{,}5\ x\ 10^3 = \frac{1}{2\pi\ x\ 1000\ x\ C}$$
$$C \cong 12\ nF \tag{3.3}$$

Sendo assim, o capacitor utilizado no filtro será de 12 nF. Além disso, foram utilizados capacitores de *bypass* de 1 µF em todas as alimentações. O circuito completo do amplificador de isolação, incluindo seu circuito de proteção (COUTINHO, não publicado) é mostrado na Figura 3.9.

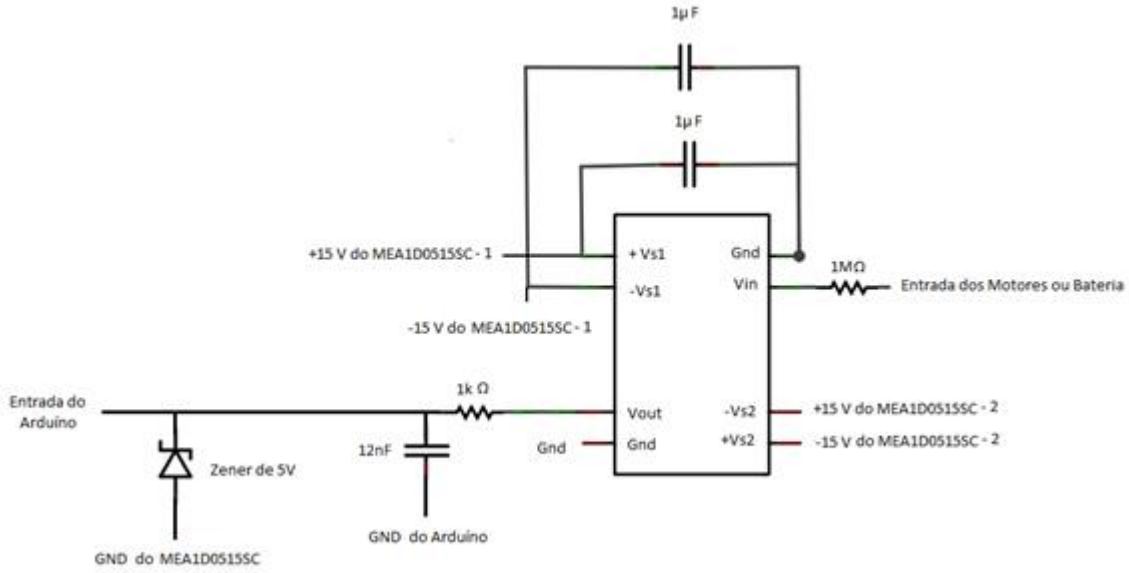

Figura 3.9: Diagrama do Circuito do Amplificador de Isolação ISO122JP.

Na Figura 3.10 é mostrada a ligação do conversor DC/DC. O terra digital do Arduino (Capítulo 4) será utilizado como referência para a entrada do conversor.

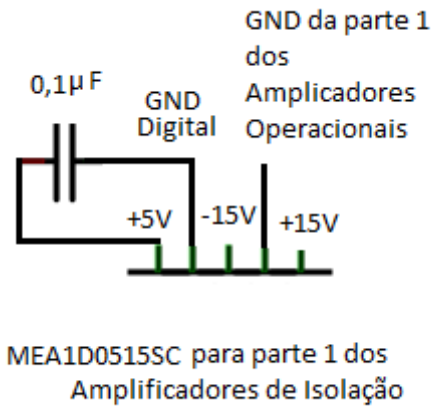

Figura 3.10: Diagrama do Circuito do Conversor DC/DC MEA1D0515SC.

O orçamento do sistema de aquisição é apresentado no Apêndice B.

4 AQUISIÇÃO DE DADOS

Para realizar a aquisição das variáveis mensuradas através dos sensores é necessária a utilização de um microcontrolador que seja capaz de organizar os dados e disponibilizá-los de forma prática, simples e eficiente.

4.1 O microcontrolador Arduino

Nossa faculdade já dispunha de um sistema microcontrolado Arduino, que possui um microcontrolador Atmel, que atende plenamente as exigências deste Projeto de Graduação, por isso ele foi escolhido para este projeto.

O Arduino é uma plataforma eletrônica baseada em hardware flexível e um *software* de fácil utilização. O microcontrolador Arduino pode perceber o ambiente ao receber a entrada de uma variedade de sensores e responder por luzes de controle, motores e outros atuadores. O microcontrolador na placa é programado usando a linguagem de programação Arduino e o ambiente de desenvolvimento Arduino. Os projetos de Arduino podem ser *stand-alone* ou podem comunicar-se com *software* rodando em um computador.

O microcontrolador Arduino pode ainda ser utilizado no desenvolvimento de objetos interativos, admitindo entradas de uma série de sensores ou chaves, e controlando outras saídas físicas. Os circuitos podem ser montados à mão ou comprados pré-montados; o *software* de programação de código-livre pode ser baixado gratuitamente.

A linguagem de programação do Arduino é uma implementação do *Wiring*, uma plataforma computacional física semelhante, que é baseada no ambiente multimídia de programação *Processing* (BACELLAR, 2012). Se tornando meramente um conjunto de funções C/C++ que podem ser chamadas em seu código. Seu esboço sofre pequenas mudanças (como geração automática de protótipos de funções) e então é passado diretamente para um compilador C/C++ (avr-g++). Todas as construções padrão C e C++ suportadas pelo avr-g++ devem funcionar no Arduino. Para mais detalhes, veja a página Processo de construção do Arduino (ARDUINO, 2012).

O Arduino será ligado na porta USB de um *netbook* LG X140 rodando Linux Mandriva, que emulará uma porta serial. Na Figura 4.1 é mostrada a placa do Arduino.

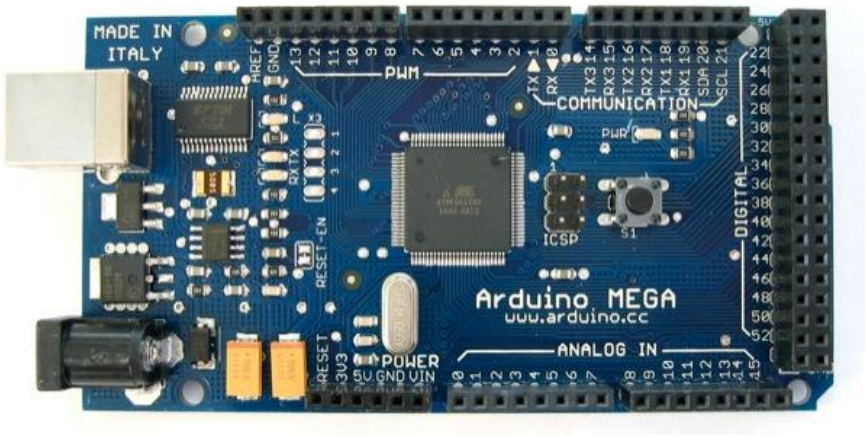

Figura 4.1: Arduino.

Anteriormente, foi falado que o conversor DC/DC seria alimentado pela porta USB do computador. Na verdade, o Arduino que será alimentado pela USB, e ele, por sua vez, disponibilizará esta tensão pelo pino 5 V. O pino GND, logicamente é o terra.

Os pinos ANALOG IN são onde serão ligadas as saídas de todos os sensores, onde a tensão máxima nesses pinos não pode exceder 5 V. Como o computador não trabalha com grandezas analógicas, o Arduino possui um conversor A/D de 10 bits integrado, que será usado para converter a saída analógica dos sensores para um valor binário (Arduino, 2009).

4.2 Apresentação dos dados

Como já são sabidas quais variáveis são interessantes de se monitorar e foi identificado o melhor meio de medi-las e como controlar esse processo, só resta encontrar uma maneira de como esses dados possam ser expostos de uma maneira eficiente e que atinja a nossa principal diretriz nesse Projeto de Graduação.

Quando os manuais e materiais sobre esse microcontrolador eram estudados, foi descoberto que é possível fazer uma interface entre Arduino e uma página de Internet através da utilização das linguagens de programação do microcontrolador e das linguagens de programação HTML (*HyperText Markup Language*) e PHP (*Hypertext Preprocessor*) (SOARES, 2009).

> O PHP é uma linguagem para a criação de scripts para a Web do lado do servidor embutidos em HTML, cujo código-fonte é aberto, e que é compatível com os mais importantes servidores *Web* (especialmente o Apache). O PHP permite incorporar fragmentos de código em páginas de HTML normais – código esse que é interpretado à medida que suas páginas são oferecidas aos usuários. O PHP também serve como uma linguagem de "cola", facilitando a conexão de suas páginas web com o banco de dados do lado do servidor (CONVERSE e PARK, 2002).

Sabendo-se disso, conseguiremos atingir plenamente e de uma forma bem robusta, direta, segura e inteligente o objetivo do Projeto de Graduação que é a monitoração. Através do uso da Internet, essa monitoração das variáveis pode ser feita, com segurança, bem longe do ponto de operação da embarcação não tripulada, além de garantir que a mesma seja *online* com a medição – garantindo velocidade na averiguação de possíveis problemas com a embarcação.

Basicamente, o sistema funcionaria da seguinte forma: o usuário acessaria a página PHP, que escreveria algum caractere para a porta serial do servidor. O Arduino, por sua vez, detectaria esse caractere e escreveria a leitura dos sensores na porta serial. O PHP então leria a porta e apresentaria o valor na tela ao usuário (SOARES, 2009).

Entretanto, esse sistema só daria suporte a um único usuário acessando a página. Foi pensado então em rodar-se uma página em PHP no servidor que leria a porta serial e guardaria os dados num arquivo de texto. A página que o cliente acessaria trabalharia apenas com o arquivo de texto, sendo possível que várias pessoas usassem o sistema ao mesmo tempo.

Por outro lado, seria necessário que o browser ficasse aberto continuamente no servidor, não havendo a possibilidade de trocar de usuário no *netbook*. A solução definitiva foi, em vez de ler a porta serial com uma página em PHP, ler através de um programa escrito em linguagem C.

> [A linguagem C] é frequentemente chamada de linguagem de médio nível para computadores. Isso não significa que C seja menos poderosa, difícil de usar ou menos desenvolvida que uma linguagem de alto nível como BASIC e Pascal, tampouco [sic] implica que esta seja similar à linguagem *assembly* e apresente problemas correlatos (...). [A linguagem de programação] C é tratada como uma linguagem de médio nível porque combina elementos de linguagens de alto nível com a funcionalidade da linguagem *assembly* (...). Como uma linguagem de médio nível, C permite a manipulação de bits, bytes e endereços – os elementos básicos com os quais o computador funciona (SCHILDT, 1997, p. 4, 5).

O programa em C rodaria em plano de fundo no computador continuamente. Ele escreveria os dados num arquivo de texto: um com apenas a última leitura, que seria lido pela página que o cliente acessaria, e outro arquivo de *log*, ao qual seria adicionada uma nova linha a cada leitura, a fim de monitorar o histórico da embarcação. Além da leitura dos sensores, o programa também escreveria a data e a hora da medida. O fluxo de dados é mais bem ilustrado no diagrama de blocos da Figura 4.2:

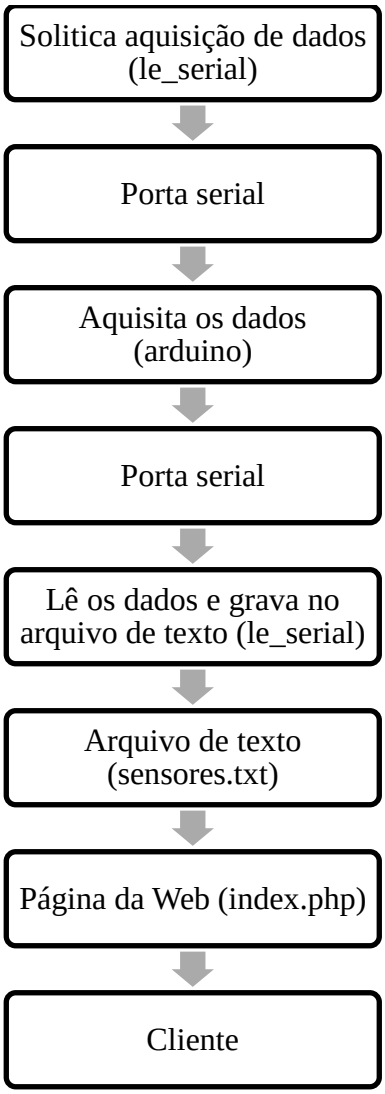

Figura 4.2: Fluxo de dados.

Cada leitura dos sensores será enviada em apenas uma linha pela serial, sendo que a leitura de cada sensor será separada por uma barra vertical da seguinte. Além disso, no início da linha constará a data, no seguinte formato: "Dia da semana", "Mês", "Dia", "Hora:Minuto:Segundo", "Ano". A ordem dos sensores será: temperatura do circuito, temperatura da água, temperatura do ar, tensão no motor 1, tensão no motor 2, tensão no

motor 3, tensão na bateria, corrente no motor 1, corrente no motor 2, corrente no motor 3, umidade do ar e inundação. Exemplo:

Thu Mar 8 20:00:51 2012|29.4|25.2|24.2|11.75|11.91|11.83|12.29|39.4|39.8|39.7|35|0.

Ou seja:

- Data da medida: 20h51m, quarta-feira, 8 de março de 2012
- Temperatura do circuito: 29,4 ºC
- Temperatura da água: 25,2 ºC
- Temperatura do ar: 24,2 ºC
- Tensão no motor 1: 11,75 V
- Tensão no motor 2: 11,91 V
- Tensão no motor 3: 11,83 V
- Tensão na bateria: 12,29 V
- Corrente no motor 1: 39,4 A
- Corrente no motor 2: 39,8 A
- Corrente no motor 3: 39,7 A
- Umidade do ar: 35 %
- Inundação: Não

4.2.1 Aquisição de dados

O programa de aquisição de dados rodará no próprio Arduino e será responsável basicamente por ler os dados dos sensores e enviá-los através da porta serial. Como na biblioteca do Arduino não há uma função nativa para arredondamento, inicialmente é criada uma função chamada “arred” para esta função. Após isto, é definida como 9600 bps a velocidade de comunicação da porta serial. Em seguida, são definidas constantes que serão usadas para o ajuste dos sensores.

Após verificar se a porta serial está disponível, é verificado também se o caractere “1” foi lido na porta. Daí em diante, o Arduino lê os dados de todos os sensores, realizando todas as conversões e ajustes necessários. Por fim, a leitura das medidas é impresso na serial.

O arquivo do programa em Arduino será chamado arduino.pde, e será armazenado na pasta home/rafaelvida/public_html, bem como todos os arquivos deste Projeto de Graduação. O código-fonte do programa está disponível no Apêndice A.1.

4.2.2 Comunicação com o sistema de aquisição

O programa para se comunicar com o sistema de aquisição (Arduino) será escrito em linguagem C e sua responsabilidade será enviar um pedido ao Arduino para ler os sensores, receber a leitura e escrevê-la num arquivo de texto. Inicialmente, o mesmo abre a porta serial e define a velocidade de comunicação como a mesma que foi definida no *software* do Arduino, que é 9600 bps. O caractere “1” é enviado pela porta serial, o que inicia o processo de leitura dos sensores pelo Arduino. Após aguardar 1 segundo até o Arduino obter a leitura de todas as medidas, a resposta é obtida, então essa leitura, juntamente com a data atual, é gravada num arquivo de texto, sobrescrevendo o arquivo anterior, além de adicionar uma nova linha em um arquivo de *log*.

O arquivo que contém o código do programa será chamado arduino.c, o arquivo que conterá a última leitura, sensores.txt, e o arquivo de *log*, sensores.log. O código-fonte do programa está disponível no Apêndice A.2.

4.2.3 Apresentação final dos dados

Como a página que apresentará os dados será usada para monitoração da embarcação, é interessante que ela seja recarregada automaticamente sem nenhuma ação do usuário. Inicialmente, isto era feito através da inclusão do atributo HTTP-EQUIV=“REFRESH” na tag META da página, porém, dessa forma, seria necessário carregar toda a página novamente, além de causar lentidão no navegador, em computadores mais lentos. A solução encontrar foi atualizar a página via *Ajax* (*Asynchronous JavaScript and XML*).

> O *Ajax* é uma nova tecnologia em ascensão (...), [embora não seja] realmente novo. *Ajax* não é por si só tecnologia, mas sim uma forma de mesclar outras tecnologias que já conhecemos a fim de tornar páginas *web* mais interativas (...). O *Ajax* reúne várias tecnologias *Web* bem estabelecidas e as usa de maneira nova e interessante. Os princípios do *Ajax* separam elegantemente o cliente do servidor e podem ser utilizados com qualquer linguagem *server-side*. (CARVALHO, 2008).

Primeiramente, a função *JavaScript* que atualiza a página é chamada e é feita uma requisição assíncrona à página que lê o arquivo de texto a cada 1 segundo. Em seguida, o arquivo de texto onde estão as medidas dos sensores é aberto, e sua única linha é transformada em um *array*, onde cada índice se refere a um sensor. Por fim, os dados são exibidos na tela.

A página que recarrega automaticamente com o tempo será chamada de index.php, e a que lê o arquivo sensores.txt, será chamada de status.php. Já o arquivo *JavaScript* ficará localizado na subpasta "js", e será chamado de ajax.js. Ainda há um arquivo CSS (*Cascading Style Sheets*), que formatará o visual da página, cujo nome será css.css e ficará localizado na subpasta "css". O código-fonte da programação está disponível nos Apêndices A.3.

5 TESTES COM O ARDUINO

Com os ajustes e testes refeitos com o circuito todo montado, o último passo para conclusão deste Projeto de Graduação são os testes da placa com o Arduino conectado.

O circuito impresso foi conectado ao microcontrolador e esse último ao *notebook* através de seu cabo USB. Ao tentarmos ler os dados na página HTML percebemos que muitos deles não condiziam com os valores esperados. Estudando razões para tais ocorrências, percebeu-se que o Arduino só estava fornecendo 4,3 V para alimentação do circuito impresso. Com esse valor não seria possível alimentar todos os componentes e o funcionamento deste Projeto de Graduação estaria severamente comprometido.

Percebeu-se logo que o sistema não conseguia manter a tensão de 5 V com tanta carga (cerca de 365 mA (COUTINHO, não publicado)), porém a grande dúvida era se o Arduino que não estava conseguindo fornecer essa tensão, ou se era a própria USB que não conseguia fazer isso.

A primeira solução pensada foi utilizar uma fonte externa feita a partir da tensão da bateria, mas a alta corrente mostrou-se um problema complexo para ser superado.

Supondo-se que o problema estava no Arduino, outra solução mais simples seria a ligação direta de uma outra porta USB para alimentar o circuito e manter o Arduino sendo alimentado em separado. Utilizamos um *mouse* USB e abrimos o seu cabo principal que contém 4 fios: um vermelho (V_{CC}), um preto (GND) , um verde e um branco (dados), além de uma blindagem interna de alumínio, e outra de estanho (LOBÃO, 2011). Na Figura 5.1 podemos ver a estrutura de um cabo USB.

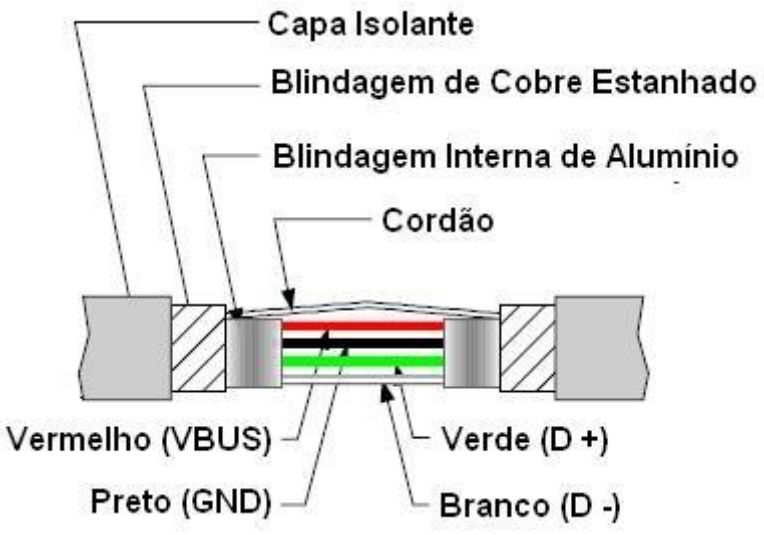

Figura 5.1: Estrutura de um cabo USB.

Utilizamos os fios de Vcc e terra para alimentar o circuito da placa sem ainda ligar o Arduino, e o resultado foi ainda pior: 3,6 V.

A solução então seria alimentar parte do circuito diretamente pela porta USB, e outra parte através da USB ligada ao Arduino. O componente que mais consome corrente desde Projeto de Graduação é o conversor MEA1D0515SC, que alimenta a saída dos quatro amplificadores de isolação, cerca de 134 mA (COUTINHO, não publicado). Sendo assim, inicialmente a outra porta USB foi utilizada para alimentar apenas ele.

Dessa forma, com o conversor alimentado por uma porta USB e o Arduino alimentado pela outra e conectado ao circuito, foi verificado que a alimentação do conversor só chegava a 4,39V. Com esta alimentação, ele forneceria ±13,24 V aos amplificadores de isolação. Porém, como, apesar da alimentação ideal dos amplificadores ser de ±15 V, ele pode ser alimentado com, no mínimo, ±12 V. Logo, devido a todas as circunstâncias, essa queda na alimentação do conversor seria aceitável.

Por outro lado, no Arduino, a tensão fornecida era de 4,87 V, o que já estava bem próximo dos ideais 5 V. Porém, durante os testes verificou-se que a leitura apresentada era mais alta que o valor real. Isso acontece porque a tensão de referência do conversor analógico-digital do chip Atmel é a mesma que a tensão de alimentação (ATMEL, 2012, p. 377). Embora houvesse apenas uma diferença de 0,13 V em relação à alimentação ideal, isso se tornou um problema.

Levando em consideração que a leitura de cada entrada analógica em volts é dada pela equação abaixo, onde V_{ref} é a tensão de referência do conversor A/D do Arduino, n é o número de bits do conversor e analogRead(A0) é sua leitura digital, o problema foi facilmente resolvido definindo-se uma variável chamada “vcc” com o valor da tensão de alimentação no início do programa do Arduino e utilizando-a em todas as conversões das leituras:

$$\left({V_{ref}}/{2^n}\right) * analogRead(A0) \tag{5.1}$$

Em relação aos sensores, essa queda na alimentação não se mostra tão problemática, sendo que inclusive há sensores que podem ser alimentados com menos de 5 V.

5.1 Resultados experimentais

Finalmente, as medidas dos sensores foram verificadas. Nos casos onde havia vários sensores do mesmo tipo, apenas um deles foi verificado.

Como exemplo, o sensor que medirá a tensão do segundo motor foi ligado à bateria, apresentando em tela um resultado de 13,1 V. Com o multímetro, a tensão foi medida e o valor encontrado foi de 12,4V. Sendo assim, foi necessário definir a constante de ajuste dos amplificadores de isolação no programa em Arduino (c_tensao_motor_1, c_ tensao _motor_2, c_ tensao _motor_3, c_tensao_bateria) como -0,7 V.

Já o sensor que medirá a corrente do segundo motor foi ligado em aberto, para ajuste do seu zero, já que, como dito anteriormente no Capítulo 3, sua tensão de saída a 0 A é $V_{cc}/2$ V. O valor apresentado em tela foi de -0,7 A. Dessa forma, foi necessário definir a constante de ajuste dos sensores de corrente no programa em Ardruino (c_corr_motor_1, c_corr _motor_2, c_ corr_motor_3) como 0,7 A.

O sensor de inundação foi testado com a esponja dentro de um balde, com os fios conectados em suas extremidades e saindo do balde até o Arduino. Como esperado, quando o nível de água dentro do balde ficou relativamente alto, foi informado que o sistema estava inundado.

A temperatura medida foi, em média, de 25,9 ºC, e a umidade do ar, 52%. Não foi possível ajustar estes dois sensores, já que não existe equipamento disponível para o mesmo no Laboratório de Engenharia Elétrica, além de os valores obtidos serem bem razoáveis.

5.2 Telas de programas

Para que o Arduino leia os sensores, primeiramente é necessário enviar o código para o seu *chip*. Isto é feito através do botão *Upload*, na barra de ferramentas da IDE (*Integrated Development Environment*) do Arduino, como é mostrado na Figura 5.2.

Figura 5.2: Barra de ferramentas da IDE do Arduino.

Após isso, é necessário compilar o código-fonte do programa em linguagem C. Para isso, basta ir a *Application Launcher Menu* > Konsole. Após ser aberta a tela do Konsole, digitar:

> cd public_html

> gcc -o “le_serial” “arduino.c”

> ./arduino

O segundo comando compila o arquivo “arduino.c” e gera o executável “le_serial”. Caso o programa já esteja compilado, ele não será necessário.

A Figura 5.3 mostra a leitura de cada sensor separada por uma barra. Nesta figura, assim como nas a seguir, somente a leitura da temperatura da água, da tensão e corrente do segundo motor, da umidade e da inundação que estarão corretas, já que estes foram os únicos sensores a serem conectados.

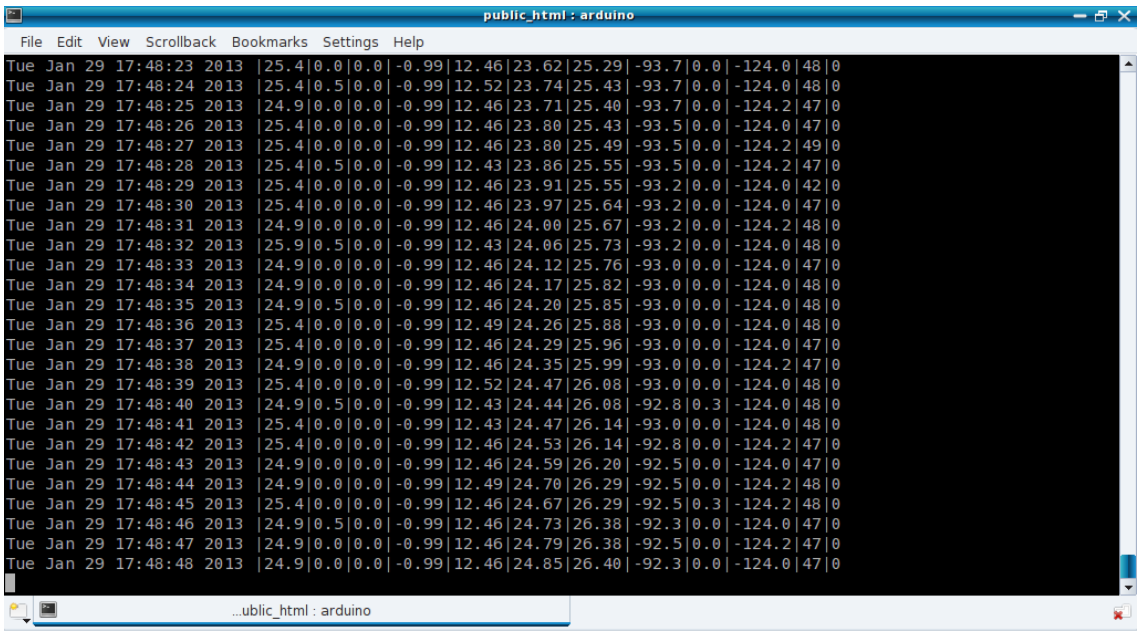

Figura 5.3: Tela do programa “le_serial”.

Após isso, basta abrir a página http://localhost/~rafaelvida, que a página da *Web* com todas as informações é exibida, como mostra a Figura 5.4.

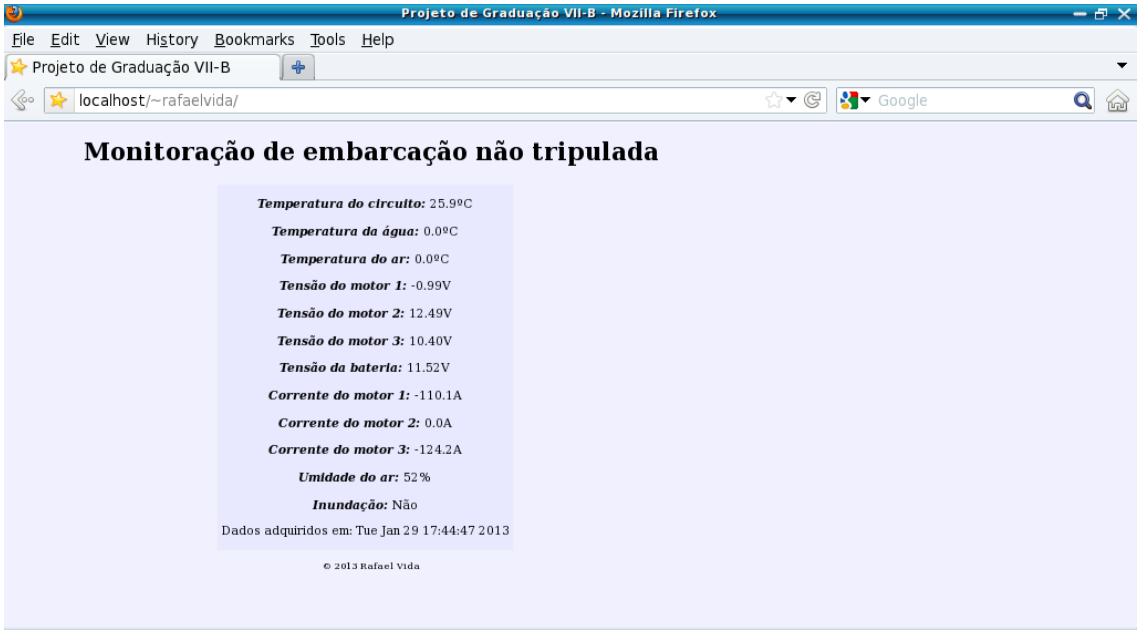

Figura 5.4: Página da *Web* "index.php" que exibe as variáveis do barco.

Para quem acessa a página de outro computador, basta trocar *localhost* pelo IP (*Internet Protocol*) da máquina.

Como ilustração é mostrada na Figura 5.5 o arquivo sensores.txt que armazena a última leitura dos dados.

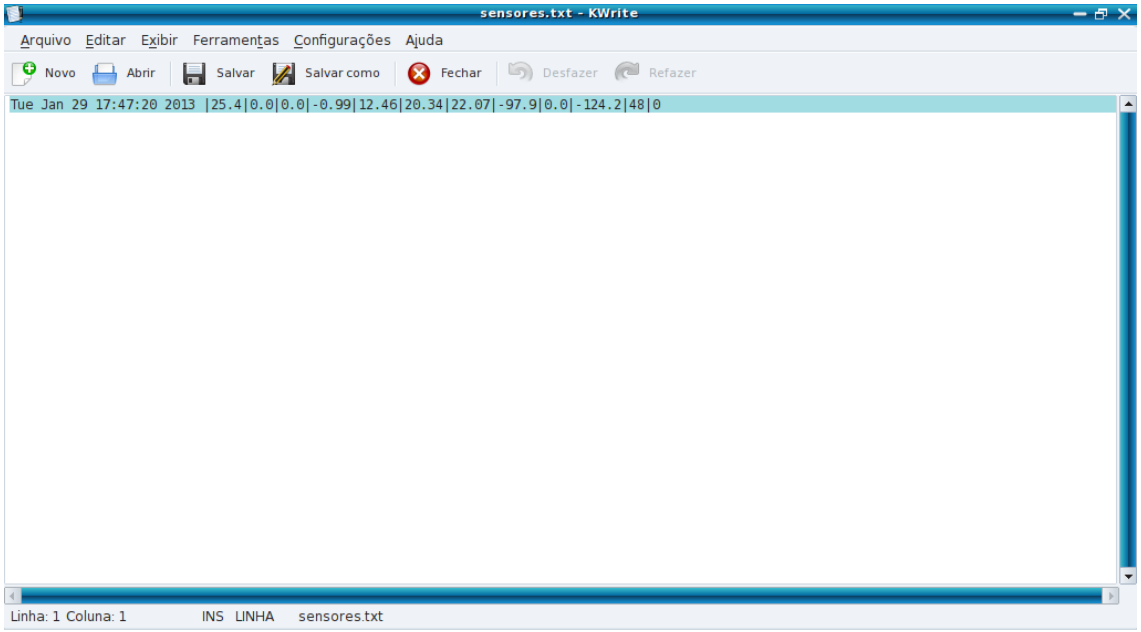

Figura 5.5: Arquivo "sensores.txt".

Já na Figura 5.6 é mostrado o arquivo sensores.log, que contem o histórico de todas as medidas.

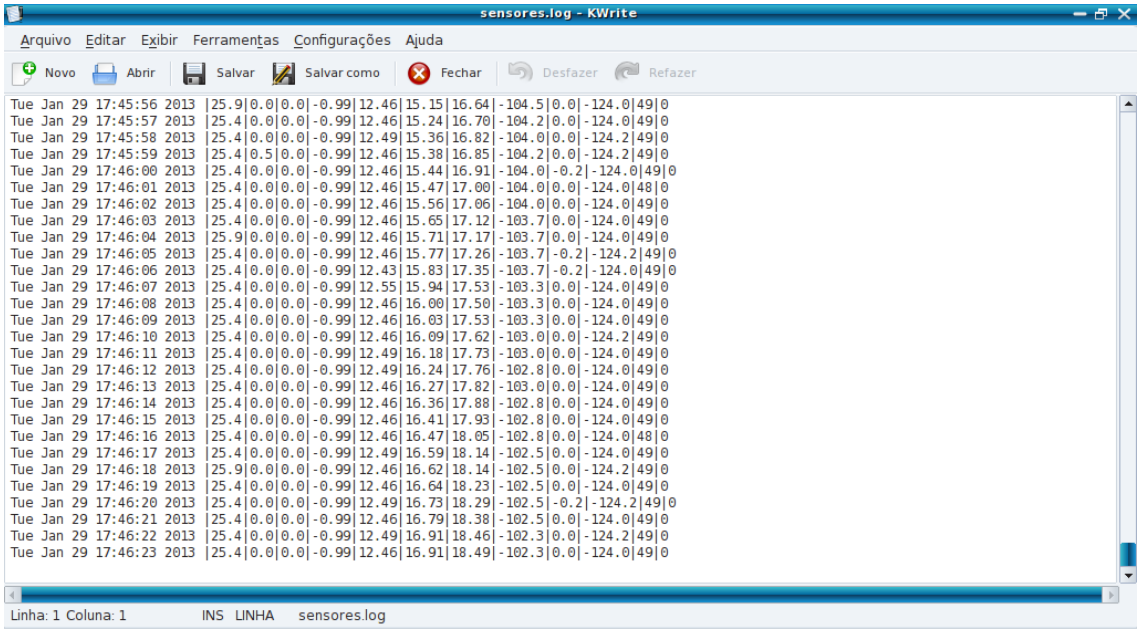

Figura 5.6: Arquivo “sensores.log”.

6 CONCLUSÃO

Ao início deste Projeto de Graduação, a única forma de ter acesso à leitura das medidas da embarcação seria através do botão *Serial Monitor* da IDE do Arduino. Porém, desta forma, para acessar as medidas era necessário estar dentro dela, o que foge ao principal intuito do Projeto: ser uma embarcação não tripulada. Entretanto, com a integração das linguagens Arduino, C e PHP, foi possível que a embarcação fosse monitorada em qualquer parte do mundo, bastando apenas ter acesso à Internet.

Durante o desenvolvimento deste Projeto, concluímos que, apesar de o sistema de aquisição de dados parecer simples, há vários fatores que complicam o desenvolvimento do mesmo.

A primeira dificuldade foi o fato de a porta USB só fornecer 500 mA de corrente, logo foi obrigatória a escolha de componentes mais econômicos. Caso o sistema fosse mais robusto, com mais sensores, por exemplo, haveria uma dificuldade adicional neste ponto.

Outra dificuldade foi que, mesmo a porta USB fornecendo 500 mA, ela não consegue manter a tensão de 5 V constante quando a corrente se aproxima deste limiar, fato que foi solucionado utilizando mais de uma porta para a alimentação do circuito, além de algumas correções no *software* para contornar este problema.

Futuramente, caso haja a necessidade de usar novos sensores, basta liga-los ao Arduino e adicionar novas linhas aos códigos dos programas, para que as leituras sejam feitas, além de tomar cuidado com as tensões de alimentação.

Se houver a necessidade de trocar os motores e bateria por modelos de 24 V também não haverá problema, pois a leitura da tensão dos mesmos já previu esta situação.

6.1 Sugestões para trabalhos futuros

Apesar deste Projeto de Graduação se intitular **SISTEMA PARA MONITORAÇÃO DE EMBARCAÇÃO NÃO TRIPULADA**, haveria a necessidade de se ter acesso à embarcação toda vez que a bateria do *netbook* descarregar, para recarregá-la. Uma sugestão para trabalhos futuros seria o desenvolvimento de um sistema que garantisse a autonomia do sistema de aquisição de dados sem intervenção humana, com o uso de energia solar, por exemplo.

Em nível de programação, há duas sugestões: uma simples e uma mais complexa. A simples seria, em vez de gravar os dados num arquivo de texto, gravá-los num arquivo CSV (*Comma-Separated Values*), o qual pode ser aberto através do Microsoft Excel ou manipulado através de alguma linguagem de programação. A mais complexa seria gravar os dados num banco de dados SQL (*Structured Query Language*) e desenvolver uma página onde fosse possível acessar as medidas em qualquer momento anterior e plotar gráficos com as variações entre diversos intervalos de tempo.

REFERÊNCIAS

ALCÂNTARA, R. Filtro Passa Baixa Digital Simples. **Eletronica.org**, 2008. Disponível em: <http://www2.eletronica.org/artigos/eletronica-digital/filtro-passa-baixa-digital-para-microcontroladores-de-8bit>.

ALLEGRO MICROSYSTEMS. **ACS756**, 2011. Disponível em: <http://www.allegromicro.com/~/Media/Files/Datasheets/ACS756-Datasheet.ashx>.

ARDUINO. **Arduino Build Process**, 2012. Disponível em: <http://arduino.cc/en/Hacking/BuildProcess>.

ARDUINO. **Arduino – ArduinoBoardMega**, 2009. Disponível em: <http://arduino.cc/en/Main/ArduinoBoardMega>.

ARDUINO, **Arduino – DigitalReadSerial**, 2009. Disponível em: <http://arduino.cc/en/Tutorial/DigitalReadSerial>.

ATMEL. **8-bit Atmel Microcontroller with 64K/128K/256K Bytes In-System Programmable Flash**, 2012. Disponível em: <http://www.atmel.com/Images/doc2549.pdf>.

BACELAR, G. Arduino – O que é?. **Bileras**, 2012. Disponível em: <http://www.bileras.com.br/arduino-o-que-e/>.

BOJORGE, N. Simbologia e Nomenclatura de Instrumentação. **Departamento de Engenharia Química e de Petróleo – UFF**, 2010. Disponível em: <http://www.professores.uff.br/controledeprocessos-eq/images/stories/Aula02_Instrumen_Nomenclat.pdf>.

BOYLESTAD, R.; NASHELSKY, L. **Dispositivos Eletrônicos e Teoria de Circuitos**. 6. ed. Rio de Janeiro: Livros Técnicos e Científicos Editora S.A., 1998.

BURR-BROWN. **ISO 122 JP - Precision Lowest Cost ISOLATION AMPLIFIER**, 1993. Disponível em: <http://www.datasheetcatalog.org/datasheet2/a/0s7p3k438f9c8x8q3kh4zzqacq7y.pdf>

C++ REFERENCE. **Asctime**, 2000. Disponível em: <http://www.cplusplus.com/reference/clibrary/ctime/asctime/>.

C++ REFERENCE. **Strcat**, 2000. Disponível em: <http://www.cplusplus.com/reference/clibrary/cstring/strcat/>.

CARVALHO, T. S. **Fundamentos de Ajax e o Modelo DOM**, 2008. Disponível em: <http://www.ebah.com.br/content/ABAAAAEYAAK/fundamentos-ajax>.

CONVERSE, T.; PARK, J. **PHP a Bíblia**. 2. ed. Rio de Janeiro: Campus, 2003.c

COUTINHO, T. G. **Montagem de um Sistema de Aquisição de Dados**, 2013. Projeto de Graduação em Engenharia Eletrônica - UERJ.

CRESPI, R.; CERON T. A. **Sensor de temperatura LM35**. Disponível em: <http://hermes.ucs.br/ccet/demc/vjbrusam/inst/temp51.pdf>.

CUNHA, J. P. V. S. **Embarcações Não Tripuladas para Monitoração Ambiental e Defesa**. Rio de Janeiro: FAPRJ, 2010.

CUNHA, J. P. V. S. **Projeto e Estudo de Simulação de um Sistema de Controle a Estrutura Variável de um Veículo Submarino de Operação Remota**. Rio de Janeiro: COPPE/UFRJ, 1992.

CUNHA, R. Eletrônica analógica: capacitores de desacoplamento. **Saber Eletrônica**, 2009. Disponível em: <http://www.sabereletronica.com.br/secoes/leitura/1147>.

DALLY, J. W.; RILEY, W. F.; MCCONNELL, K. G. **Instrumentation for Engineering Measurements**. 2. ed. Michigan: Wiley, 1993.

DOEBELIN, E. O. **Measurement Systems – Application und Design**. 5. ed. Michigan: Mcgraw-Hill College, 2003.

FRADEN, J. **Handbook of Modern Sensors**. 3. ed. Califórnia: Springer, 2003.

HALL E. H. **On a New Action of the Magnet on Electric Currents**. The Johns Hopkins University Press, 2011. Disponível em: <http://cc.ee.ntu.edu.tw/~thlin/E_Hall_paper_1879.pdf >.

HONEYWELL. **HIH-4000 Series**, 2010. Disponível em: <http://sensing.honeywell.com/index.php/ci_id/49922/la_id/1/document/1/re_id/0>.

KENNY. Viva o Linux. **Arredondamento**, 2007. Disponível em: <http://www.vivaolinux.com.br/topico/C-C++/Arredondamento>.

KIEREN, S. Remove trailing newline character from asctime output. **KBlog**, 2010. Disponível em: <http://www.ukdragon.com/index.php/2010/01/18/remove-trailing-newline-character-from-asctime-output>.

LNCC. **Fronteiras e Limites do Brasil**, 2012. Disponível em: <http://info.lncc.br/>.

LOBÃO, F. R. USB – Conexão Elétrica. **Fassi**, 2011. Disponível em: <http://www.fassi.com.br/artigos/usb000/usb002>.

MOHAN, N. **Power Electronics: Converters, Applications, and Design.** 2. ed. Michigan: Wiley, 2002.

NATIONAL SEMICONDUCTOR CORPORATION. **LM35 - Precision Centigrade Temperature Sensors**, 2010. Disponível em: <http://www.ti.com/lit/ds/symlink/lm35.pdf>.

OGATA, K. **Engenharia de Controle Moderno**. 4. ed. São Paulo: Pearson Prentice Hall, 2003.

PROJETO 39. **O que é Arduino?**, 2010. Disponível em: <http://projeto39.wordpress.com/o-arduino/>.

THE PHP GROUP. **PHP: fgets – Manual**, 2012. Disponível em: <http://php.net/manual/pt_BR/function.fgets.php>.

THE PHP GROUP. **PHP: fopen – Manual**, 2012. Disponível em: <http://php.net/manual/pt_BR/function.fopen.php>.

SCHILDT, H. C Completo e Total. 3. ed. São Paulo: Makron Books, 1996.

SCHULTZE, H. J. **Projeto e Construção de uma Embarcação Teleoperada**, 2012. Projeto de Graduação em Engenharia Eletrônica - UERJ.

SILVA, O. J. Aprenda a abrir ou criar arquivos usando a função fopen(). **Arquivo de Códigos, 2010**. Disponível em: <http://www.arquivodecodigos.net/dicas/c-aprenda-a-abrir-ou-criar-arquivos-usando-a-funcao-fopen-2276.html>.

SILVA, O. J. Escrevendo palavras ou textos em um arquivo usando a função fputs(). **Arquivo de Códigos, 2010**. Disponível em: <http://www.arquivodecodigos.net/dicas/c-escrevendo-palavras-ou-textos-em-um-arquivo-usando-a-funcao-fputs-2604.html>.

SILVA, O. J. Lendo o conteúdo de um arquivo uma linha de cada vez. **Arquivo de Códigos, 2010**. Disponível em: <http://www.arquivodecodigos.net/dicas/c-lendo-o-conteudo-de-um-arquivo-uma-linha-de-cada-vez-2280.html>.

SOARES, B. **Controlando a Arduino com PHP via porta serial**, 2009. Disponível em: <http://blog.bsoares.com.br/php/controlling-arduino-with-php>.

STEFFENS, C. Sensores. **Universidade Federal do Rio de Grande do Sul**, 2006. Disponível em: <http://www.if.ufrgs.br/mpef/mef004/20061/Cesar/SENSORES-Definicao.html>.

SWEET, M. R. **Serial Programming Guide for POSIX Operating Systems**, 2005. Disponível em: <http://www.easysw.com/~mike/serial/serial.html>.

XAVIER, D. W. Operações Matemáticas. **TI Expert**, 2010. Disponível em: <http://www.tiexpert.net/programacao/web/php/concatenacao-e-operacoes-matematicas.php>.

WIKIPEDIA. **Água (substância)**, 2012. Disponível em: <http://pt.wikipedia.org/wiki/%C3%81gua_(subst%C3%A2ncia)#Condutividade_el.C3.A9trica>.

APÊNDICE A - Programas

Neste apêndice são mostrados os códigos-fonte de todo o *software* usado na monitoração da embarcação não tripulada.

A.1 - Aquisição de dados

Abaixo segue o código do programa de aquisição de dados rodando no Arduino que lê os sensores e envia os dados para a serial.

```
float arred(float numero, int casas) // Cria uma funçao que arredonda o numero definido por "numero" ate a casa decimal de numero "casas"...
{
  return floor(numero * pow(10, casas) + 0.5) / pow(10, casas);
}

void setup()
{
  Serial.begin(9600); // Define a velocidade de comunicaçao da porta serial...
  pinMode(2, OUTPUT); // Define o pino digital 2 como saida...
}

int serial;
float vcc = 4.87; // Tensao de alimentaçao do Arduino...
float temp_circ = 0; // Define o desvio da medida do sensor da temperatura do circuito em relaçao a medida correta...
float temp_agua = 0; // Define o desvio da medida do sensor da temperatura da agua em relaçao a medida correta...
float temp_ar = 0; // Define o desvio da medida do sensor da temperatura do ar em relaçao a medida correta...
float c_tensao_motor_1 = -0.7; // Define o desvio da medida do sensor de tensao do motor 1 em relaçao a medida correta...
float c_tensao_motor_2 = -0.7; // Define o desvio da medida do sensor de tensao do motor 2 em relaçao a medida correta...
```

```
float c_tensao_motor_3 = -0.7; // Define o desvio da medida do sensor de tensao do motor 3 em relaçao a medida correta...
float c_corr_motor_1 = 0.7; // Define o desvio da medida do sensor da corrente do motor 1 em relaçao a medida correta...
float c_corr_motor_2 = 0.7; // Define o desvio da medida do sensor da corrente do motor 2 em relaçao a medida correta...
float c_corr_motor_3 = 0.7; // Define o desvio da medida do sensor da corrente do motor 3 em relaçao a medida correta...
float c_tensao_bateria = -0.7; // Define o desvio da medida do sensor de tensao da bateria em relaçao a medida correta...
int c_umidade = 0; // Define o desvio da medida do sensor de umidade em relaçao a medida correta...

void loop()
{
  if (Serial.available() > 0) { // Verifica se a porta serial esta disponivel...
    if (Serial.read() == 1); { // Le a porta serial...
      float temp_circ = arred(vcc / 0.01 / 1024.0 * analogRead(A0), 1); // Le a temperatura do circuito...
      float temp_agua = arred(vcc / 0.01 / 1024.0 * analogRead(A1), 1); // Le a temperatura da agua...
      float temp_ar = arred(vcc / 0.01 / 1024.0 * analogRead(A11), 1); // Le a temperatura do ar...
      float tensao_motor_1 = 6.0 * arred((vcc / 1024.0 * analogRead(A2)), 3) + c_tensao_motor_1; // Le a tensao do motor 1...
      float tensao_motor_2 = 6.0 * arred((vcc / 1024.0 * analogRead(A3)), 3) + c_tensao_motor_2; // Le a tensao do motor 2...
      float tensao_motor_3 = 6.0 * arred((vcc / 1024.0 * analogRead(A4)), 3) + c_tensao_motor_3; // Le a tensao do motor 3...
      float tensao_bateria = 6.0 * arred((vcc / 1024.0 * analogRead(A5)), 3) + c_tensao_bateria; // Le a tensao da bateria...
      float corr_motor_1 = arred(vcc / 0.02 / 1024.0 * (analogRead(A6) - 512) + c_corr_motor_1, 1); // Le a corrente do motor 1...
      float corr_motor_2 = arred(vcc / 0.02 / 1024.0 * (analogRead(A7) - 512) + c_corr_motor_2, 1); // Le a corrente do motor 2...
      float corr_motor_3 = arred(vcc / 0.02 / 1024.0 * (analogRead(A8) - 512) + c_corr_motor_3, 1); // Le a corrente do motor 3...
      int umidade = round((vcc / 1024.0 * analogRead(A9) - 0.9237 + 0.0041 * temp_circ - 0.00004 * pow(temp_circ, 2)) / (0.0305 + 0.000044 * temp_circ - 0.0000011 * pow(temp_circ, 2))); // Le a umidade da caixa...
```

```
      int inundacao = analogRead(A10); // Le o resultado da conversao A/D
do sensor de inundaçao...
      Serial.print("|");
      Serial.print(temp_circ, 1); // Imprime na serial a temperatura do
circuito com 1 casa decimal de precisao...
      Serial.print("|"); // Imprime um espaço na serial...
      Serial.print(temp_agua, 1); // Imprime na serial a temperatura da
agua com 1 casa decimal de precisao...
      Serial.print("|"); // Imprime um espaço na serial...
      Serial.print(temp_ar, 1); // Imprime na serial a temperatura do ar
com 1 casa decimal de precisao...
      Serial.print("|"); // Imprime um espaço na serial...
      Serial.print(tensao_motor_1); // Imprime na serial a tensao do motor
1 com 2 casas decimais de precisao...
      Serial.print("|"); // Imprime um espaço na serial...
      Serial.print(tensao_motor_2); // Imprime na serial a tensao do motor
2 com 2 casas decimais de precisao...
      Serial.print("|"); // Imprime um espaço na serial...
      Serial.print(tensao_motor_3); // Imprime na serial a tensao do motor
3 com 2 casas decimais de precisao...
      Serial.print("|"); // Imprime um espaço na serial...
      Serial.print(tensao_bateria); // Imprime na serial a tensao da
bateria com 2 casas decimais de precisao...
      Serial.print("|"); // Imprime um espaço na serial...
      Serial.print(corr_motor_1, 1); // Imprime na serial a corrente do
motor 1 com 1 casa derintcimal de precisao...
      Serial.print("|"); // Imprime um espaço na serial...
      Serial.print(corr_motor_2, 1); // Imprime na serial a corrente do
motor 2 com 1 casa decimal de precisao...
      Serial.print("|"); // Imprime um espaço na serial...
      Serial.print(corr_motor_3, 1); // Imprime a corrente do motor 3 com 1
casa decimal de precisao...
      Serial.print("|"); // Imprime um espaço na serial...
      Serial.print(umidade); // Imprime na serial a umidade com 1 casa
decimal de precisao...
      Serial.print("|"); // Imprime um espaço na serial...
      if (inundacao > 300) { // Se o valor da leitura A/D do sensor de
inundaçao for maior que 300 (aproximadamente 1,5 V), imprime "1" na serial.
Caso contrario, imprime "0";
        Serial.println("1");
      }
```

```
      else {
        Serial.println("0");
     }
    }
  }
}
```

A.2 - Comunicação com o sistema de aquisição

Abaixo segue o código do programa que se comunica com o sistema de aquisição (Arduino), escrito em linguagem C, que lê a serial e guarda os dados em um arquivo de texto.

```
#include <stdio.h>
#include <fcntl.h>
#include <termios.h>
#include <time.h>
#include <string.h>

int main()
{
    char porta_serial[12] = "/dev/ttyUSB0"; // Define a qual porta serial está conectado o Arduino...
    int tamanho = 100; // Define o tamanho máximo da linha a ser lida da porta serial...
    int delay = 1000000; // Define o tempo que o programa vai esperar o Arduino...
    int fd = open(porta_serial, O_RDWR | O_NOCTTY | O_NDELAY); // Abre a porta...
                    struct termios options;
    tcgetattr(fd, &options); // Obtém as opções atuais para a porta...
                        cfsetispeed(&options, B9600);
    cfsetospeed(&options, B9600); // Configura a velocidade de comunicação com a serial para 9600 bps...
    options.c_cflag |= (CLOCAL | CREAD); // Habilita o receptor e configura o modo local...
    tcsetattr(fd, TCSANOW, &options); // Configura as novas opções da porta...
    int n; // Cria uma variavel para o loop da linha de baixo...
```

```
        for (n=1;;n++) // Cria um loop infinito...
                                                                {
     FILE *porta = fopen(porta_serial, "w+"); // Abre a porta para leitura
e                                                               escrita...
                char caracter[1] = "1";
     fwrite(&caracter, 1, 1, porta); // Escreve o caractere "1" na porta
serial...
      usleep(delay); // Aguarda 1 segundo até o Arduino responder...
                              char valores[tamanho];
     fgets(valores, tamanho, porta); // Guarda o que está na porta serial
na                            variável "valores"...
            fclose(porta); // Fecha a porta...
                                time_t rawtime;
                     struct tm *timeinfo;
                                                    time(&rawtime);
                 timeinfo = localtime(&rawtime);
     char *data = asctime(timeinfo); // Armazena a data atual na variável
"data"...
     data[strlen(data)-1] = '\0'; // Substitui o último caracter da data,
que é uma quebra de linha, por um caracter nulo...
      strcat(data, " "); // Adiciona um espaço depois da data...
     strcat(data, valores); // Concatena a data com os dados obtidos da
porta serial e guarda na variável "data"...
     if (n > 1) // Na primeira iteração (n = 1), descarrega dados inválidos
da                                                              serial...
                                                                {
        printf("%s", data); // Exibe a data mais os dados obtidos da porta
serial                          na                              tela...
     FILE *sensores = fopen("sensores.txt", "w+"); // Abre o arquivo
sensores.txt                    para                            escrita...
    fputs(data, sensores); // Guarda a data mais o que estava na porta
serial                          no                              arquivo...
            fclose(sensores); //Fecha o arquivo...
    sensores = fopen("sensores.log", "a"); // Abre o arquivo sensores.log
para                                                            inserção...
    fputs(data, sensores); // Adiciona uma linha contendo a data mais o que
estava na porta serial ao arquivo...
           fclose(sensores); //Fecha o arquivo...
                                                                }
                                                                }
}
```

Abaixo segue o código da página PHP que exibe os dados na tela:

```
<html>
	<head>
		<title>Projeto de Graduação VII-B</title>
		<script type="text/javascript" src="js/ajax.js"></script> <!-- Chama o arquivo de JavaScript "ajax.js"... -->
		<style type="text/css">
			@import url("css/css.css"); <!-- Chama a folha de estilos CSS "css.css"... -->
		</style>
	</head>
	<body>
		<script type="text/javascript">
		refreshdiv_timediv(); <!-- Chama a função refreshdiv_timediv() do script... -->
		</script>
		<h1>Monitoração de embarcação não tripulada</h1>
		<div id="corpo"></div> <!-- Cria uma div que conterá o conteúdo da página que será atualizada constantemente... -->
		<p id="rodape">© 2013 Rafael Vida</font></p>
	</body>
</html>
```

A.3 - Apresentação final dos dados

Abaixo segue a página PHP que lê os dados dos sensores:

```
<?php
	header("Content-Type: text/html; charset=ISO-8859-1",true);
	$port = fopen('sensores.txt','r'); // Abre o arquivo "sensores.txt" para leitura...
	$serial = fgets($port); // Guarda o conteúdo do arquivo na variável $serial...
	fclose($port); // Fecha o arquivo...
	// A linha do arquivo a ser lida tem o seguinte formato: "Dia da semana" "Mês" "Dia" "Hora:Minuto:Segundo" "Ano" "Temp. circuito" "Temp. água"...
```

```
	// Ou seja, inicia com a data, seguida pelas medidas dos sensores, separados por um espaço...
	// Os 25 primeiros caracteres representam a data...
	$data = explode("-", $serial);
	if($data[12] == 1) {
		$inundacao = "Sim";
	} else $inundacao = "Não";
	echo "<p class='medida'><span>Temperatura do circuito: </span>", $data[1], "°C";
	echo "<p class='medida'><span>Temperatura da água: </span>", $data[2], "°C";
	echo "<p class='medida'><span>Temperatura do ar: </span>", $data[3], "°C";
	echo "<p class='medida'><span>Tensão do motor 1: </span>", $data[4], "V";
	echo "<p class='medida'><span>Tensão do motor 2: </span>", $data[5], "V";
	echo "<p class='medida'><span>Tensão do motor 3: </span>", $data[6], "V";
	echo "<p class='medida'><span>Tensão da bateria: </span>", $data[7], "V";
	echo "<p class='medida'><span>Corrente do motor 1: </span>", $data[8], "A";
	echo "<p class='medida'><span>Corrente do motor 2: </span>", $data[9], "A";
	echo "<p class='medida'><span>Corrente do motor 3: </span>", $data[10], "A";
	echo "<p class='medida'><span>Umidade do ar: </span>", $data[11], "%";
	echo "<p class='medida'><span>Inundação: </span>", $inundacao;
	?>
	<p id="data">Dados adquiridos em: <?php echo $data[0]; ?></p>
```

Abaixo segue o código JavaScript que faz a requisição assíncrona da leitura dos sensores:

```
function AJAX(){ // A função Ajax...
try{
xmlHttp=new XMLHttpRequest(); // Firefox, Opera 8.0+, Safari...
return xmlHttp;
```

```
}
catch (e){
try{
xmlHttp=new ActiveXObject("Msxml2.XMLHTTP"); // Internet Explorer...
return xmlHttp;
}
catch (e){
try{
xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
return xmlHttp;
}
catch (e){
alert("Your browser does not support AJAX.");
return false;
}
}
}
}

function fetch_unix_timestamp() // Função para prevenir solicitações GET do
IE...
{
 return parseInt(new Date().getTime().toString().substring(0, 10))
}

function refreshdiv_timediv(){ // Função que atualiza a div

var seconds = 1; // Define o intervalo que a página será atualizada...
var divid = "corpo"; // Define a id da div que será atualizada...
var url = "status.php"; // Define o endereço da página que será colocada na
div...

var xmlHttp_one = AJAX(); // Cria o xmlHttp...

var timestamp = fetch_unix_timestamp();
var nocacheurl = url+"?t="+timestamp; // Sem cache...

xmlHttp_one.onreadystatechange=function(){ // O código...
if(xmlHttp_one.readyState==4){
document.getElementById(divid).innerHTML=xmlHttp_one.responseText;
setTimeout('refreshdiv_timediv()',seconds*1000);
```

```
}
}
xmlHttp_one.open("GET",nocacheurl,true);
xmlHttp_one.send(null);
}

window.onload = function startrefresh(){ // Inicia o processo de
atualização...
setTimeout('refreshdiv_timediv()',seconds*1000);
}
```

Abaixo segue a folha de estilos CSS que formata o layout e estilo da página:

```
* {
      font-family:"trebuchet ms";
}
body {
      width:980px;
      background-color:#F0F0FF;
}
h1 {
      text-align:center;
}
p#data {
      text-align:center;
}
p#rodape {
      text-align:center;
      font-size:12px;
}
div#corpo {
      text-align:center;
      border:1px solid ActiveBorder;
      width:400px;
      position:relative;
      margin-left:280px;
      background-color:#E8E8FF;
}
span {
      font-style:italic;
```

```
      font-weight:bold;
}
p.medida:hover {
      background-color:#D8D8FF;
}
```

APÊNDICE B - Orçamento

No Quadro B.1 segue o orçamento parcial deste Projeto de Graduação.

Quadro B.1: Orçamento parcial.

Sensor	**Fabricante**	**Quantidade**	**Preço Unitário (R$)**	**Valor Total (R$)**
ISO122JP	Texas Instruments	4	60,99	243,96
MEA1D0515SC	Mouser Eletronics	5	16,76	83,82
ACS756SCA-100B-PFF-T	Allegro Microsystems	3	17,31	51,93
HIH 4000-001	Honeywell	1	48,63	48,63
LM35	Texas Instruments	3	3,80	11,40
Arduino Mega	Arduino	1	139,90	139,90
Total (R$)				579,64