UNIVERSIDADE ESTADUAL PAULISTA

“Júlio de Mesquita Filho”

Faculdade de Engenharia - Campus de Ilha Solteira

Programa de Pós Graduação em Engenharia Elétrica

“Desenvolvimento de Hardware e Software para Tratamento e Visualização de Dados de Descargas Atmosféricas Aplicados à Segurança da Navegação Hidroviária”

Edmundo Beinecke

Orientador:
Prof. Dr. Aparecido Augusto de Carvalho
Co-Orientador:
Prof. Dr. Luiz Roberto Trovati

Dissertação submetida à Faculdade de Engenharia de Ilha Solteira – FEIS/UNESP – como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica.
Área de conhecimento: Controle e Automação.

Ilha Solteira (SP), Fevereiro de 2013.

FICHA CATALOGRÁFICA

Elaborada pela Seção Técnica de Aquisição e Tratamento da Informação
Serviço Técnico de Biblioteca e Documentação da UNESP - Ilha Solteira.

Beinecke, Edmundo.
B422d Desenvolvimento de hardware e software para tratamento e visualização de dados de descargas atmosféricas aplicados à segurança da navegação hidroviária / Edmundo Beinecke. – Ilha Solteira : [s.n.], 2013
148 f. : il.

Dissertação (mestrado) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira. Área de conhecimento: Controle e Automação, 2013

Orientador: Aparecido Augusto de Carvalho
Coorientador: Luiz Roberto Trovati
Inclui bibliografia

1. Sensores Stormscope. 2. Raio. 3. Descargas atmosféricas. 4. Segurança hidroviária. 5. Alerta de tempestades. 6. Tempestades.

unesp **UNIVERSIDADE ESTADUAL PAULISTA**

CAMPUS DE ILHA SOLTEIRA

FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA

CERTIFICADO DE APROVAÇÃO

TÍTULO: Desenvolvimento de hardware e Software para tratamento e visualização de Dados de Descargas Atmosféricas Aplicados à Segurança da Navegação Hidroviária

AUTOR: EDMUNDO BEINECKE

ORIENTADOR: Prof. Dr. APARECIDO AUGUSTO DE CARVALHO
CO-ORIENTADOR: Prof. Dr. LUIZ ROBERTO TROVATI

Aprovado como parte das exigências para obtenção do Título de Mestre em Engenharia Elétrica , Área: AUTOMAÇÃO, pela Comissão Examinadora:

Prof. Dr. APARECIDO AUGUSTO DE CARVALHO
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. ALEXANDRE CESAR RODRIGUES DA SILVA
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. WESLEY PONTES
Centro Universitário Toledo

Data da realização: 25 de fevereiro de 2013.

À Deus e a minha família, em especial aos meus pais que sempre acreditaram na minha capacidade.
DEDICO.

AGRADECIMENTOS

Ao Prof. Dr. Aparecido Augusto Carvalho pela orientação, pela dedicação e confiança no trabalho e por acreditar na minha capacidade ao longo do desenvolvimento do projeto.

Ao meu co-orientador Prof. Dr. Luiz Roberto Trovati, pelos grandes ensinamentos a mim transmitidos, conhecimentos estes que levarei por toda vida, pessoalmente e profissionalmente.

Agradeço aos meus pais Enio Beinecke e Gedi Rodrigues Beinecke, pela força e incentivo dados nos momentos difíceis e pelo tempo que abdicaram de minha companhia para que este trabalho pudesse ser concluído.

Aos meus colegas e grandes amigos do laboratório de Hidrologia e Hidrometria (LH^2) que me ajudaram no desenvolvimento do trabalho, e principalmente pelas amizades construídas durante este período.

A todos os funcionários e professores da pós-graduação que de forma direta ou indireta contribuíram para a realização deste trabalho.

A Universidade Estadual Paulista – Faculdade de Engenharia de Ilha Solteira por me acolher.

A FINEP – Financiadora de Estudos e Projetos pelo apoio financeiro concedido aos Projetos Ondisa 5 e 8, através dos convênios 01.07.0784-00 e 01.10.0699-00.

RESUMO

Neste trabalho aborda-se o desenvolvimento de rotinas de aquisição, interpretação e visualização de dados de descargas atmosféricas obtidas por sensores tipo Stormscope, visando o estabelecimento de um sistema de monitoramento de tempestades com ênfase à segurança da navegação hidroviária. O sistema é composto por dois modos operacionais, sendo um com sensor fixo em terra e outro com o sensor embarcado. O sensor Stomscope fixo em terra coleta e transmite os dados de descargas atmosféricas para uma central de dados. O sensor Stormscope embarcado é instalado em uma embarcação do tipo comboio Tietê – Paraná, no qual também opera um monitor para visualização dos dados de descargas atmosféricas e também um *hardware* dedicado para que se capture, catalogue e armazene os dados, todos de modo georreferenciados. São apresentados o desenvolvimento do *hardware* e do *software* para o monitoramento de tempestades, via acompanhamento e quantificação do número de descargas atmosféricas. O produto final é a visualização do início, desenvolvimento e fim do processo das descargas atmosféricas e sobretudo, o acompanhamento da sua evolução espacial e temporal. Simultaneamente à evolução espaço-temporal dos dados de descarga elétrica, os dados colhidos pelo sistema foram logados e catalogados para futuras análises referentes ao tema. Os dados obtidos com os sensores Stormscope foram comparados com a rede terrestre fixa RINDAT (Rede Integrada Nacional de Detecção de Descargas Atmosféricas) para fins de validação.

Palavras-chaves: Sensores Stormscope. Raios. Descargas atmosféricas. Segurança hidroviária. Alerta de tempestades.

ABSTRACT

This work discusses the development of acquiring, visualization and interpretation routines of atmospheric discharges data obtained by Stomscope sensors, aiming to establish a storm monitoring system with security alert purpose to the waterway navigation. The alert system is composed by two operational modes, on which one is a fixed sensor on ground and the other is a sensor aboard a ferry. The Stormscope sensor fixed on ground collects and transmits the atmospheric discharge data to a data central. The embedded system consists in a Stormscope sensor installed on a ferry of the Tietê-Paraná convoy type, in wich also operates a visualization system of atmospheric discharges data and also a dedicated hardware that captures, catalogs and store the data in a georeferenced way. We present the development of hardware and software for storms monitoring, via tracking and quantifying of the number of lightning. The final product is the visualization of the beginning, development and end of the process of lightning and especially the monitoring of their spatial and temporal evolution. Simultaneously to the spatio-temporal evolution of the electric discharge data, the data collected by the system are logged and cataloged for future analysis on the subject. The data obtained from the Stormscope sensor are compared to the fixed terrestrial network RINDAT (Integrated National Network of Atmospheric Discharges Detection) to validation purposes.

Keywords: Stormscope sensor. Lightning. Atmospheric discharges. Waterway security. Storm alerts.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SÍMBOLOS

ANTAQ	Agência Nacional de Transportes Aquaviários
ANTT	Agência Nacional de Transportes Terrestres
ASCII	American Standard Code for Information Interchange
CRC	Cheque de Redundância Cíclica
FATEC	Faculdade de Tecnologia de São Paulo
IPT	Instituto de Pesquisas Tecnológicas
LH²	Laboratório de Hidrologia e Hidrometria
LF	Low Frequency
LPATS	Lightning Position and Tracking System
NMEA	National Marine Electronics Association
P&D	Pesquisa e Desenvolvimento
RINDAT	Rede Integrada Nacional de Detecção de Descargas Atmosféricas
SPI	Serial Peripheral Interface
TCU	Tribunal de Contas da União
TPN	Tanque de Provas Numérico
UNESP	Universidade Estadual Paulista
USP	Universidade de São Paulo
UTC	Universal Time Coordinated
nm	Milha náutica (1nm = 1,852 km)
ω	Frequência Angular
ε_{rg}	Permissividade Elétrica Relativa do Solo
ε_0	Permissividade Elétrica do Vácuo
μ_g	Permeabilidade Magnética do solo
μ_0	Permeabilidade Magnética do Vácuo
σ_g	Condutividade Elétrica do solo
$dE_r(r,h,j\omega)$	Transformada de Fourier Diferencial de Campo Elétrico Horizontal
$dE_z(r,h,j\omega)$	Transformada de Fourier Diferencial de Campo Elétrico Vertical
$dH_\varphi(r,h,j\omega)$	Transformada de Fourier Diferencial de campo Magnético
$I(z',j\omega)$	Transformada discreta de Fourier da Distribuição Espaço-Temporal da Corrente ao Longo do Canal da Descarga de Retorno
J_0	Função de Bessel de ordem zero

SUMÁRIO

1 INTRODUÇÃO

As últimas décadas foram marcadas pelo grande avanço tecnológico da humanidade; estamos cada vez mais dependentes deste avanço para suprir nossas necessidades de alimentação, transporte, logística, saúde, comunicação, segurança, etc.

Um dos grandes desafios do século XXI é a conciliação do meio ambiente com o mundo moderno e suas formas logísticas de ida e vinda de bens de consumo, de forma segura. Em geral, meios que possam fornecer avanço tecnológico sustentável para o planeta, sem prejudicar sua fauna e flora. Para isto, pesquisas em diversos campos das ciências são desenvolvidas por todo globo, formas de produção mais eficientes de alimentos e energia, métodos de comunicação, novas formas de combustíveis não poluentes, etc.

Nas áreas de logística e segurança, investem-se milhões de reais anualmente no país em infraestrutura nos modais de transporte brasileiro, para que se possibilite o crescimento econômico projetado para as próximas décadas, de um PIB U$ 2 trilhões de dólares em 2009 para U$ 9,7 trilhões em 2050 (CHADE, 2011).

Para que estas previsões se concretizem, investimentos do setor privado e público devem ser realizados para melhoria do modal brasileiro, não somente na infraestrutura, mas também projetos em P&D, segurança e estudos relativos ao desenvolvimento e optimização dos modais.

Investimentos nestes setores tornarão a logística no país mais eficiente e de maior interconexão entre os modais, permitirá fácil escoamento de produção dos nichos produtores para as regiões consumidoras e portos, caso contrário, o setor logístico irá se tornar um dos grandes gargalos para futuras projeções econômicas. De nada adianta produzir em alto volume sem um baixo custo logístico e de grande eficiência; altos custos logísticos encarecem produtos tornando-os menos competitivos no mercado interno e externo.

Este trabalho tem um caráter científico e aplicado de monitoramento de descargas atmosféricas de modo a prover uma contribuição para o desenvolvimento da navegação hidroviária. Este monitoramento visa auxiliar os hidroviários na tomada de decisão referente à eventuais condições de navegabilidade adversa na sua rota ao longo da hidrovia e os portuários nos procedimentos de carga/descarga das embarcações. As previsões informadas pelo sistema são do tipo *nowcasting* (previsão de curtíssimo prazo) utilizando-se sensores Stormscope para detecção de descargas atmosféricas.

Especialmente no caso da Hidrovia Tietê – Paraná que é composta por lagos com grande espelho d'água, as tempestades severas estão associadas a ventos de alta intensidade que geram ondas cuja magnitude de altura e período, normalmente, dificultam sobremaneira a navegação. Assim, o sistema é de relevante importância para o país, pois, a vista da projeção para o futuro transporte hidroviária de cargas perigosas como combustíveis (Projeto Corredor do Etanol - Transpetro) um eventual acidente na hidrovia causada por efeitos atmosféricos pode contaminar ou mesmo extinguir fauna e flora ao longo do rio, interromper abastecimento de água de cidades ribeirinhas, por vidas de hidroviários em risco e ocasionar pesadas multas ambientais aos responsáveis pela embarcação.

Um aprofundamento sobre o assunto econômico e logístico brasileiro será feito no Apêndice C – Sistema Modal Brasileiro.

1.1 ESTADO DA ARTE NA SEGURANÇA HIDROVIÁRIA BRASILEIRA

Segundo auditoria realizada pelo TCU no ano de 2011, as hidrovias brasileiras no geral estão praticamente abandonadas, sete das oito hidrovias apresentam problemas sérios de sinalização e assoreamento, exceção é a Hidrovia Tietê-Paraná.

Hidrovias como a do Rio São Francisco, sofrem de problemas de sinalização, assoreamento, não possuem sistemas de dragagem, tampouco manutenção das eclusas existentes em seu percurso.

Ademais, no tema relativo aos alertas meteorológicos para a segurança da navegação hidroviária, apenas o Estado de São Paulo, recentemente iniciou as primeiras tratativas de abordagem dessa questão com o projeto de implantação de um centro de monitoramento no trecho da hidrovia do Rio Tietê.

Os fatores atmosféricos são responsáveis por diversos acidentes, tanto para transporte de passageiros, com especial atenção às hidrovias amazônicas, quanto para transporte de cargas, que podem gerar problemas ambientais sérios devido à natureza de sua carga.

Atualmente, há uma série de ações de cunho científico e tecnológico. Estas ações são lideradas pelas Universidades e fomentadas pelas agências governamentais de pesquisa e pela iniciativa privada com projetos P&D, com o propósito de tornar as hidrovias seguras e logisticamente eficientes.

O grupo de pesquisa TPN da Escola Politécnica da USP desenvolveu um simulador virtual para comboios fluviais, que permite a capitães de embarcações controlarem, através de manches e situações variadas de navegação, tais como, tempo adverso, vento, neblina, mapas reais em escala de diversos pontos de alta periculosidade ao longo da hidrovia Tietê-Paraná. Este simulador ajudará na formação e treinamento de novos capitães das embarcações, antes que tenham aulas práticas na hidrovia.

A FATEC de Jaú/SP, além da formação de recursos humanos com ênfase em navegação hidroviária, também realiza pesquisas relacionadas à modelagem hidrodinâmica de comboios em escala.

Pinto Jr. (2005), estuda o fenômeno dos raios e suas incidências no Brasil, baseados no livro milenar de Sun Tzu “A Arte da Guerra”. O autor discute métodos efetivos de prevenção de acidentes relacionados a descargas atmosféricas, para empresas e instituições, seus dados e análises são baseados no sistema RINDAT, e tem o intuito de diminuir prejuízos causados todos os anos por raios no país.

O Instituto de Pesquisas Tecnológicas (IPT) desenvolve e analisa soluções tecnológicas de engenharia naval para os setores de transportes marítimos, equipamentos navais e produção de petróleo. Outra importante linha de trabalho é a instrumentação e

monitoramento em escala real de navios (provas de mar, medições de forças e movimentos, extensometria e análise de desempenho de equipamentos navais, entre outros).

Na Faculdade de Engenharia de Ilha Solteira / UNESP vale destacar as contribuições do grupo de pesquisa do LH² - Laboratório de Hidrologia e Hidrometria, que há mais de 10 anos vem desenvolvendo pesquisas no tema vento x ondas em lagos e sistemas de sensoriamento e instrumentação para a segurança da navegação hidroviária.

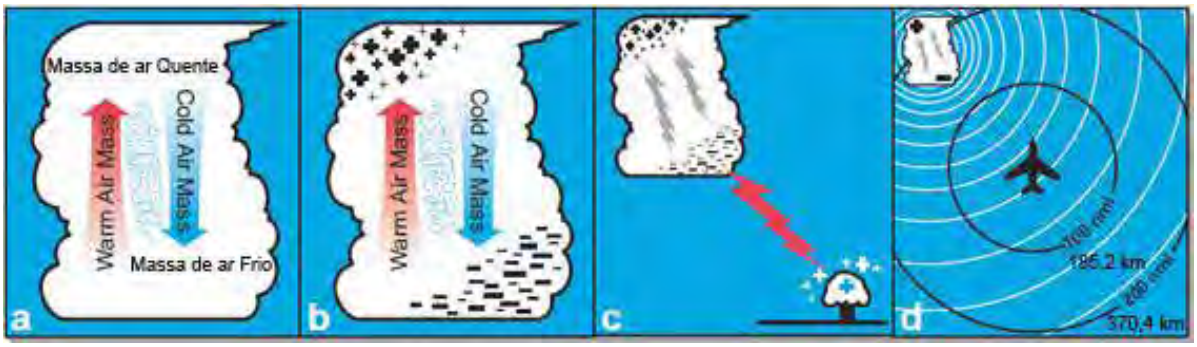
Naccarato e Pinto Jr. (2012), estudam a evolução espaço-temporal de tempestades utilizando uma rede de 48 sensores terrestres localizados em diversos pontos do país. O trabalho permite a quantização dos dados de descargas atmosféricas de forma a abranger praticamente todo o país.

A utilização de sensores Stormscope é algo inédito em trabalhos científicos voltados à detecção e acompanhamento espaço-temporal de tempestades. O trabalho é de grande valia na avaliação do sensor e ponto inicial para outros projetos de âmbito cientifico.

1.2 ANATOMIA DE UMA TEMPESTADE

Para a melhor compreensão do processo de detecção de descargas eletromagnéticas atmosféricas ou raios que é a base do funcionamento dos sensores utilizados neste trabalho, decidimos por, inicialmente, apresentar o mecanismo de produção dos raios. A figura Figura 1 exemplifica de forma resumida os estágios causadores de um raio e o efeito eletromagnético.

Figura 1 - Descargas Elétricas em uma Tempestade



Fonte: Adaptado de L-3 Communications Avionics Systems (1996)

a. A corrente de ar convectivo flui (ar quente indo pra cima e ar gelado indo pra baixo) leva a uma fricção entre as correntes opostas de ar e ventos cisalhantes. O quão mais próximo as correntes de ar são, maior será a força cisalhante das correntes de ar;

b. A fricção entre as correntes de ar opostas faz com que as cargas elétricas das moléculas do ar se separem. Como as cargas positivas (+) e negativas (-) estão separadas, há acúmulo de cargas de mesmo sinal, positivas no topo da nuvem e negativas perto da parte de baixo da mesma;

c. Descargas elétricas ocorrem quando as massas acumuladas de cargas positivas e negativas tentam se juntar, devido à atração eletromagnética que uma exerce sobre a outra. Estas descargas continuam a ocorrer repetidamente enquanto o cisalhamento do vento convectivo persiste. Algumas das descargas são visíveis como um relâmpago; a maioria das descargas elétricas ocorre dentro de uma nuvem ou entre elas. Apenas uma pequena porcentagem das descargas ocorre entre as nuvens e a terra. Descargas de nuvem-terra ocorrem quando a parte negativa da nuvem induz uma carga positiva em um objeto que está no chão. A imensa carga de separação quebra então o isolamento elétrico do ar, descarregando carga negativa da nuvem no objeto e nos arredores;

d. Todas as descargas elétricas irradiam sinais eletromagnéticos em todas as direções. Os sinais eletromagnéticos têm características únicas e taxas que variam de recorrência e força de sinal. L-3 Communications Avionics Systems (1996)

1.3 ESTÁGIOS DE UMA TEMPESTADE

A formação de uma tempestade convectiva tem início com a formação de nuvens do tipo Cumulus, que evoluem temporalmente até atingirem um intenso estágio de maturidade denominado Cumulonimbus, a partir do qual retrocedem rapidamente dissipando em chuva.

a. Estágio Cumulus

O estágio cumulus, ou estágio inicial da tempestade surge muitas vezes de céu limpo, se forma em massas de ar com alguma instabilidade, quando a humidade é relativamente baixa e a temperatura é relativamente elevada, quando o aquecimento desigual da superfície da Terra faz com que as parcelas de ar flutuantes ascendam por convecção acima do nível de ponto de orvalho, dando-se a condensação de gotículas. É usualmente livre de precipitação, este estágio pode provocar ventos verticais, congelamento e cisalhamento de ventos convectivos. Estas nuvens possuem base entre 2 a 7 km de altura em latitudes médias e diâmetros variando de 200m até 6 km.

Na Figura 2 pode-se observar um exemplo de nuvem cumulus.

Figura 2 - Nuvem Cumulus

Fonte: Universidade de Lisboa (2012).

b. Estágio Maduro ou Cumulonimbus

No estágio mais maduro e intenso de uma tempestade, as gotas de água das nuvem colidem e se combinam para formar chuva e granizo, e em temperaturas mais baixas, granizo e neve. Esta etapa representa muitos perigos para aeronaves, incluindo precipitação intensa, ventos fortes de cisalhamento convectivo, turbulência severa, granizo, gelo, tornados e relâmpagos.

Estas nuvens possuem bases entre 700 e 1.500 metros, com topos chegando a 24 e 35 km de altura, sua média está em torno de 9 a 12 km de altura e diâmetros variando de 4 a 6 km. Estas são caracterizadas por uma forma de “bigorna”, o topo apresenta expansão horizontal devido aos ventos superiores, que lembra uma bigorna de ferreiro. (INMET, 2012).

Na Figura 3 podemos observar um exemplo de nuvem cumulonimbus.

Figura 3 - Nuvem cumulonimbus

Fonte: Universidade de Lisboa (2012).

c. Estágio Dissipativo

Na fase de dissipação, a corrente de vento vertical enfraquece e, ao mesmo tempo, o vento convectivo cisalhante começa a diminuir. Normalmente, ocorrem taxas de precipitação elevada nesta fase.

Na Figura 4 podemos observar um exemplo de nuvem em estágio dissipativo.

Figura 4 - Nuvem em estágio dissipativo

Fonte: Universidade de Lisboa (2012).

1.4 O FENÔMENO DAS DESCARGAS ATMOSFÉRICAS

Os relâmpagos ou descargas atmosféricas são descargas elétricas de grande extensão (cerca de alguns quilômetros) e alta intensidade (picos de corrente acima de um quiloàmpere) que ocorrem na atmosfera.

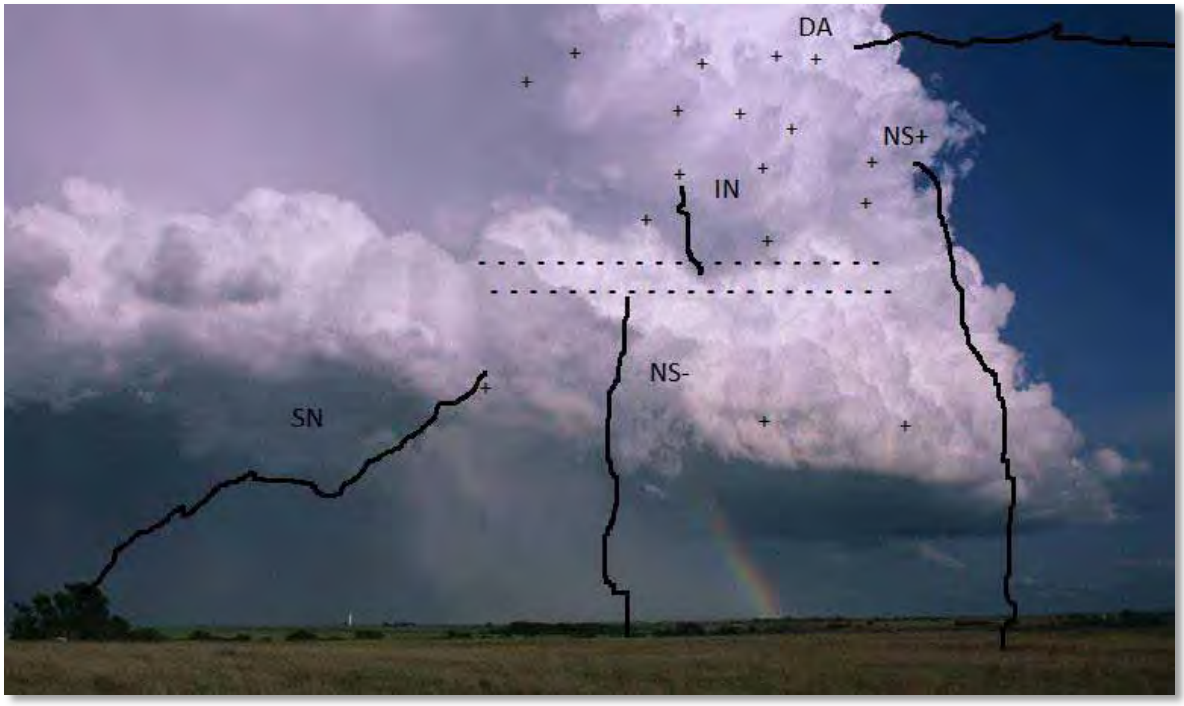
Estima-se que a incidência esteja entre cinquenta e cem relâmpagos por segundo em nosso planeta, 10 milhões por dia, chegando a impressionante casa de dois bilhões de eventos por ano. A grande maioria das ocorrências deste fenômeno fica restrita a atmosfera, a porcentagem de descargas que chegam ao solo é de 30% do total, e apesar da maior parte da superfície do planeta estar coberta por água, apenas 10% do total de descargas ocorrem nos oceanos. Quando um relâmpago atinge o solo é chamado de raio. (PINTOR JR, 2005).

Pintor Jr. (2005) explica que o fenômeno ocorre devido à concentração de cargas elétricas em regiões localizadas da atmosfera, na maior parte das vezes em tempestades. A descarga elétrica inicia quando o acumulo das cargas elétricas gera um campo elétrico de tal intensidade capaz de romper a rigidez dielétrica do ar, também chamada de capacidade isolante. Uma vez rompida a capacidade isolante do ar, ocorre o rápido deslocamento de elétrons de uma região de carga negativa que possui excesso de elétrons para uma região positiva que tem escassez destes.

Os relâmpagos são classificados de acordo com o local que se originam ou terminam, a Figura 5 ilustra os tipos existentes de relâmpagos, ela também demonstra a estrutura elétrica básica de uma nuvem de tempestade, conhecida como estrutura tripolar, esta estrutura é formada por três centros de carga: positivo na parte superior, negativo na parte intermediária e um positivo na parte inferior de menor intensidade.

Os relâmpagos que se originam da nuvem para o solo são chamados de relâmpagos nuvem-solo; os que se originam do solo para a nuvem são chamados de relâmpagos solo-nuvem; os de origem dentro da nuvem são chamados de intranuvem e os que se originam da nuvem e terminam em um ponto qualquer da atmosfera são chamados de descargas no ar.

Figura 5 - Estrutura típica de uma nuvem de tempestade e tipos de relâmpagos



Fonte: Adaptado de Pintor Jr. (2005)

Na Figura 5 estão apresentadas as representações de relâmpagos determinadas por: nuvem-solo positivos (NS+), nuvem-solo negativos (NS-), solo-nuvem (SN), intranuvem (IN) e descargas para o ar (DA).

O relâmpago do tipo intranuvem é o evento de maior ocorrência, devido a capacidade isolante do ar diminuir com a altura em função da redução da densidade do ar, e também as regiões de cargas opostas dentro da nuvem estão mais próximas em comparação com os outros tipos de relâmpagos. Os relâmpagos intranuvem representam cerca de 70% do total de descargas atmosféricas, mas este número pode sofrer alterações dependendo da localidade do globo; é de importância também ressaltar que estes eventos são de menor intensidade atingindo picos de corrente de poucos quiloàmperes.

Outros tipos de fenômenos mais frequentes são os relâmpagos nuvem-solo, dentre eles o mais comum é o nuvem-solo negativo que representa cerca de 90% dos eventos nuvem-solo, este se caracteriza pela descarga de elétrons da região negativa da nuvem em direção ao solo.

1.4.1 Características de um relâmpago

Um raio como definimos previamente é um relâmpago que atinge o solo. Tem duração média de um quarto de segundo, há registros de duração com variações de um décimo a dois segundos.

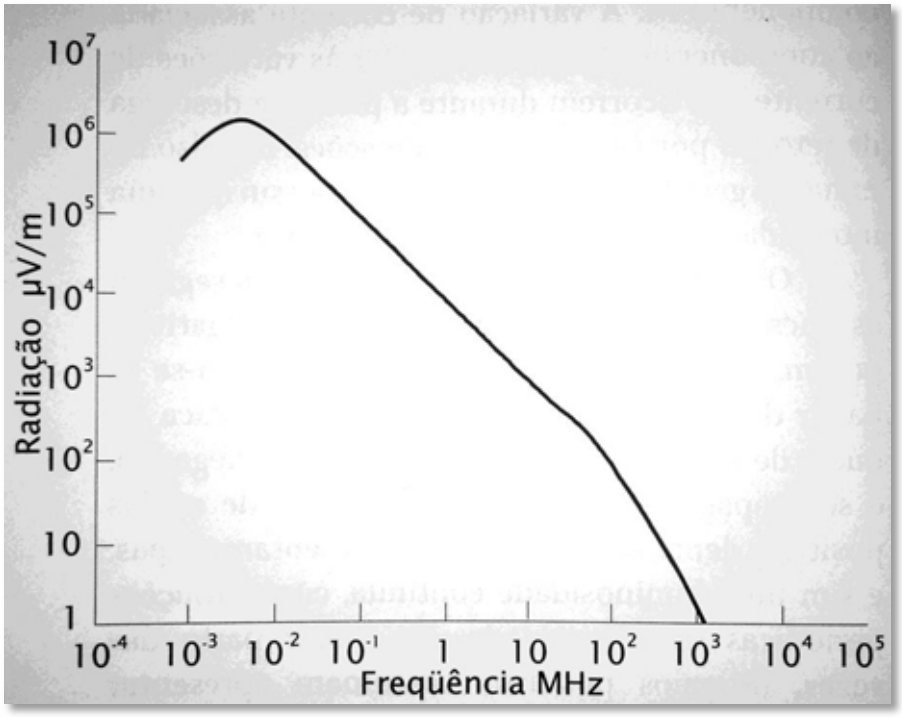
Com relação à extensão, os relâmpagos podem percorrer dezenas de quilômetros, a corrente elétrica por sua vez pode ter intensidade de algumas centenas de ampères até algumas centenas de quiloàmperes. A corrente flui por um canal de apenas alguns centímetros de diâmetro denominado de canal do relâmpago e sua temperatura pode chegar até dezenas de milhares de graus centígrados.

Quando observamos um raio, temos a impressão de ser um único evento contínuo de descarga da corrente elétrica, mas na verdade ele é a composição de vários eventos rápidos em um curtíssimo espaço de tempo. Estas são denominadas de descargas de retorno, e o número de descargas de multiplicidade do raio.

Em um raio negativo, quando a rigidez dielétrica é quebrada os elétrons se dirigem para a região de carência de elétrons através de um líder escalonado, acompanhado de uma radiação eletromagnética com variação de frequência de centenas de quilohertz a centenas de megahertz. O líder escalonado geralmente é uma descarga não visível que se propaga para fora da nuvem em direção ao solo com velocidade de cerca de 400 mil km/h.

A intensidade máxima de radiação produzida por um raio tem frequência em torno de 5 a 10 kHz, como podemos observar Figura 6. Fora da faixa visível a radiação é produzida pela aceleração dos elétrons presentes no canal do raio e representa cerca de 0,01% da energia total do raio, já na faixa visível, a radiação é devida à ionização e excitação dos átomos e moléculas de ar, representando cerca de 1% da energia do raio.

Figura 6 - Espectro da Componente Elétrica



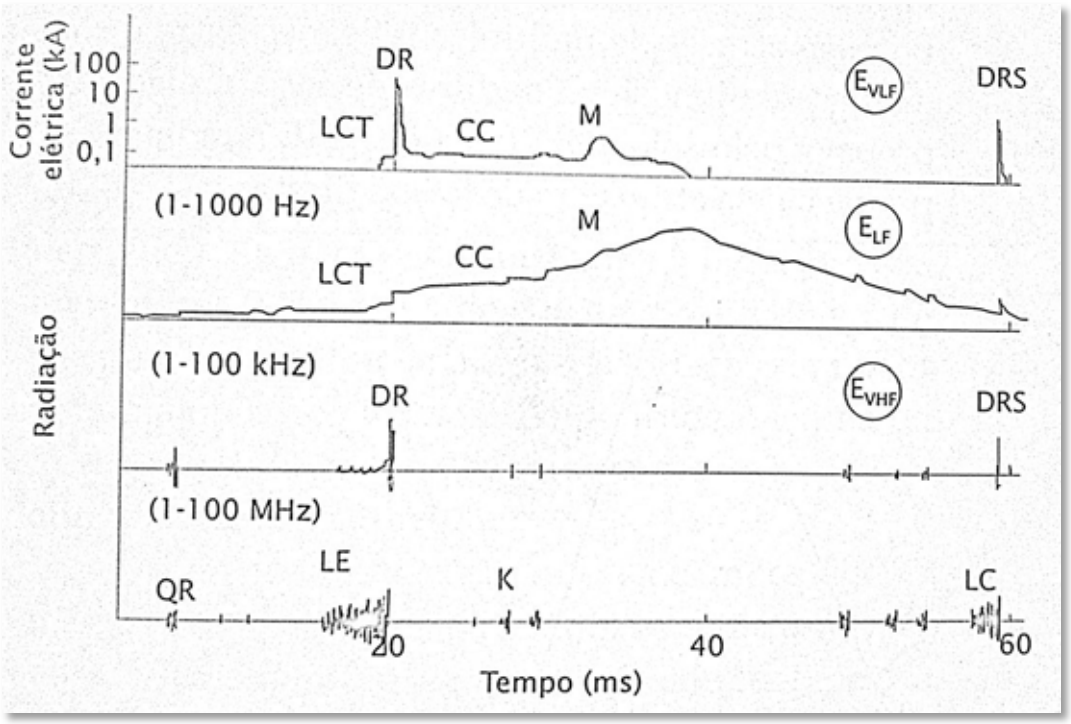
Fonte: Pintor Jr. (2005)

Na Figura 6, os resultados foram obtidos a uma distância de 20 km da base de um raio do tipo nuvem-solo negativo.

Pinto Jr. (2005), apresenta simulações sobre as radiações eletromagnéticas causadas por uma descarga atmosférica típica e suas faixas de radiações, bem como, um estudo sobre o comportamento da corrente elétrica durante as diversas fases do fenômeno.

Na Figura 7, podemos observar a corrente elétrica típica medida na base do canal para as diversas etapas de um raio nuvem-solo negativo, também podemos observar a componente elétrica da radiação observada de um ponto a cerca de 20 km de distância representadas nas faixas VLF, LF e VHF.

Figura 7 - Variação de corrente de um raio nuvem-solo negativo



Fonte: Pintor Jr. (2005)

Ainda na Figura 7 podemos observar as diversas etapas associadas como a quebra de rigidez (QR), líder escalonado (LE), líder conectante (LCT - em razão da proximidade do líder escalonado de um raio), primeira descarga de retorno (DR), corrente contínua (CC), componente M (M), descarga K (K), líder contínuo (LC) e descarga de retorno subsequente (DRS).

A corrente contínua de um raio carrega dezenas ou até mesmo centenas de Coulombs de carga para o solo, estas correntes produzem radiação principalmente na faixa de um a centenas de hertz, a luminosidade do canal aumenta por cerca de 1 ms devido ao aumento momentâneo da corrente no canal, a este aumento damos o nome de componente M.

As descargas K podem ocorrer dentro da nuvem após a primeira descarga de retorno, produzindo pulsos de radiação com variação de centenas de quilohertz a centenas de mega-hertz.

1.4.2 Equações dos campos eletromagnéticos gerados por descargas atmosféricas

É possível, em princípio, estimar os campos eletromagnéticos que seriam observados a uma determinada distância de um raio a partir das equações de Maxwell. Para isto, é necessário assumir um modelo para a corrente do canal e um modelo para a propagação da radiação ao longo da distância.

Romero (2006) descreve modelos teóricos para o cálculo da corrente de base do canal, campos elétricos e magnéticos gerados pelos eventos atmosféricos.

Em geral, a corrente na base do canal é dada pela equação 1 :

$$i(0,t) = \frac{I_o}{\eta} . \frac{\left(\frac{t}{\tau_1}\right)^n}{\left[1 + \left(\frac{t}{\tau_1}\right)^n\right]} . \exp\left(\frac{-t}{\tau_2}\right) \qquad (1)$$

sendo que:

$$\eta = \exp\left[-\left(\frac{\tau_1}{\tau_2}\right) . \left(n . \frac{\tau_2}{\tau_1}\right)^{1/n}\right]$$

Onde:

- I_0: Amplitude da corrente na base do canal (kA);
- τ_1: constante de tempo para a frente da onda (μs);
- τ_2: constante de tempo de decaimento de onda (μs);
- η: fator de correção da amplitude da corrente;
- n: expoente com valor entre 2 a 10.

A equação 1 geralmente é utilizada para o cálculo da corrente na base do canal devido à primeira descarga.

A seguir apresentaremos equações resultantes sobre os estudos modernos referentes ao tema, em coordenadas cilíndricas (ROMERO, 2006).

$$dE_r(r,h,j\omega) = \frac{j.\omega.I(z',j\omega).\mu_0.dz'}{4.\pi.{k_2}^2}.\left[\frac{\partial^2}{\partial r.\partial z}.(G_{22} - G_{21} + {k_1}^2.V_{22})\right] \quad (1)$$

$$dE_z(r,h,j\omega) = \frac{j.\omega.I(z',j\omega).\mu_0.dz'}{4.\pi.{k_2}^2}.\left[\left(\frac{\partial^2}{\partial z^2} + {k_2}^2\right).(G_{22} - G_{21} + {k_1}^2.V_{22})\right] \quad (2)$$

$$dH_\varphi(r,h,j\omega) = \frac{I(z',j\omega).dz'}{4.\pi}.\left[\frac{\partial}{\partial r}.(G_{22} - G_{21} + {k_1}^2.V_{22})\right] \quad (3)$$

Na qual:

$$G_{21} = G_{22} = \frac{exp(j.k_2.R')}{R'} \quad (4)$$

$$V_{22} = \int_0^\infty \frac{2.exp(-\gamma_2.|z'+h|)}{\gamma_2}.J_0.(\lambda r).\lambda d\lambda \quad (5)$$

Com,

$$R' = \sqrt{r^2 + (z'+h)^2}; \qquad R = \sqrt{r^2 + (z'-h)^2}; \qquad \gamma_2 = \sqrt{\lambda^2 - {k_2}^2};$$

$$k_1 = \sqrt{\left(\omega^2.\mu_g.\varepsilon_{rg}\right) + \left(j.\omega.\mu_0.\sigma_g\right)}; \qquad k_2 = \omega.\sqrt{\mu_0\varepsilon_0}$$

Nas equações cima $dE_r(r,h,j\omega)$, $dE_z(r,h,j\omega)$, $dH_\varphi(r,h,j\omega)$ correspondem, respectivamente, à transformada discreta de Fourier do campo elétrico horizontal, campo elétrico vertical e do campo magnético, associados a corrente, ao longo do canal; ε_{rg}, μ_g,

σ_g são a permissividade, permeabilidade e condutividade do solo respectivamente, $I(z', j\omega)$ é a transformada discreta de Fourier da distribuição espaço-temporal da corrente ao longo do canal $i(z', t)$, e finalmente J_0 representa a função de Bessel de ordem zero.

A equação (5) é também chamada de integral de Sommerfeld, esta integral converge lentamente, o que a torna difícil de ser avaliada numericamente. Para se determinar o campo elétrico e magnético, as equações (1), (2) e (3) devem ser integradas ao longo do canal de descarga. Feito isto, aplica-se a transformada inversa de Fourier para demonstrar o comportamento do campo eletromagnético em função do tempo.

1.5 DETECTORES DE DESCARGAS ATMOSFÉRICAS

Um detector de raios é um dispositivo que detecta descargas atmosféricas produzidas por raios na atmosfera. De maneira geral, eles captam a variação do campo elétrico, do campo magnético ou de ambos produzidos por relâmpagos.

Os primeiros sistemas de detecção de descargas atmosféricas surgiram na década de 1920 e consistiam de sensores formados por um par de espiras ortogonais. Estas espiras mediam a componente magnética dos pulsos de radiação gerada pelas descargas, na faixa de frequência em torno de 10 kHz. Nessa faixa, são registrados os pulsos de radiação associados às descargas de retorno, que são gerados próximo ao solo quando o líder escalonado da descarga se encontra com o líder conectante.

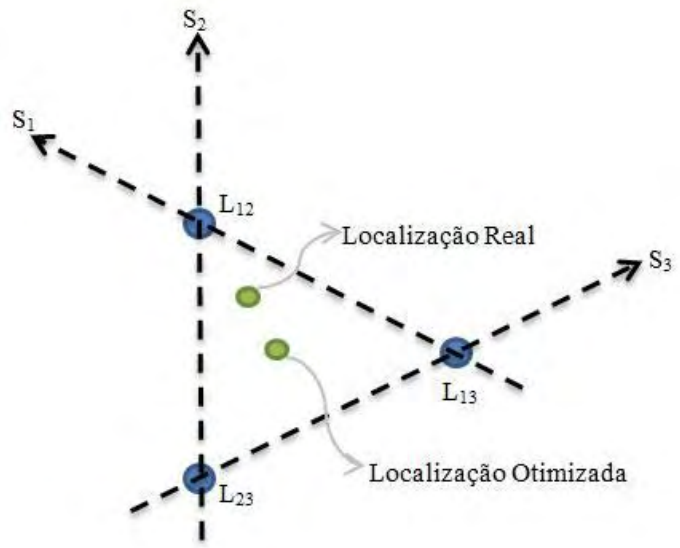
Atualmente existem três principais tipos de detectores: sistemas terrestres usando múltiplas antenas (por meio de triangulação de frequências de rádio), sistemas móveis usando uma direção e um sentido de antena no mesmo local (a bordo de um avião ou uma embarcação) e sistemas espaciais por meio de radares.

Há também sistemas satelitais com propósitos metereológicos, prevendo a evolução espaço-temporal de tormentas, a previsão do clima, alerta de furacões bem como outras previsões oceanográficas, solar-geofísicas e fenômenos terrestres.

1.5.1 Sistemas terrestres de detecção

Nos sistemas terrestres a radiação medida pelas duas espiras permite determinar seu ponto de origem, ou seja, o ponto de contato da descarga de retorno com o solo. O uso de três sensores possibilita determinar a localização da descarga atmosférica por meio do processo de otimização dos mínimos quadrados, que são obtidas a cada par de sensores. Este processo possibilita determinar a localização mais provável de ocorrência da descarga, por meio de processo de triangulação.

Figura 8 - Método da direção magnética



Fonte: Próprio autor

A Figura 8 ilustra o processo de otimização da localização do raio, os três sensores representados por (S_1, S_2, S_3) e a localização estimada para cada par de sensores (L_{12}, L_{13}, L_{23}), a localização real do evento e a localização otimizada.

1.5.2 Sistemas móveis de detecção

São sensores passivos, formados por uma antena e um processador capaz de processar as informações captadas pela antena; a antena dos sistemas móveis de detecção de descargas atmosféricas aguarda por sinais eletromagnéticos emitidos por raios. O

dispositivo capta campos elétricos e magnéticos gerados por relâmpagos intranuvem, nuvem-solo positivo e negativo e internuvem ou como definimos previamente descargas para o ar.

O processador analisa as características únicas dos sinais eletromagnéticos, quanto à força do sinal e a variação de ocorrência dos eventos, determinando assim a localização e intensidade das tempestades que geram as descargas elétricas.

1.5.3 Sistemas de detecção espaciais

São sistemas compostos por radares meteorológicos. Estes operam emitindo ondas de rádio UHF numa determinada região de interesse, recebendo de volta ecos das ondas emitidas provenientes das gotículas de água. Estes ecos são processados e transmitidos para um centro de comando dedicado onde serão analisados conforme a ocorrência e intensidade dos ecos das ondas de radio.

Este sistema é especialmente interessante para aeronaves de grande porte, pois, disponibiliza ao comandante saber a posição de nuvens no radar presente no painel da aeronave, não excluindo portanto o uso de sensores stormscope, pois, estes fazem detecção de raios complementando o sistema espacial.

1.5.4 Limitações dos sistemas

Cada sistema de detecção de relâmpagos tem suas próprias limitações.

Uma rede de detecção terrestre deve ser capaz de detectar um ataque com pelo menos três antenas, para localizá-lo com uma margem de erro aceitável. Isso, muitas vezes, leva à rejeição de um relâmpago nuvem-nuvem (ou intranuvem).

Uma antena pode detectar a posição da ocorrência de descarga elétrica na nuvem de partida e outra antena na nuvem receptora do mesmo relâmpago. Como resultado, as redes terrestres têm uma tendência a subestimar o número de ocorrências, principalmente no início das tempestades em um relâmpago nuvem-nuvem é prevalente.

Detectores móveis, como usam atenuação de sinal, em vez de triangulação, podem por vezes erroneamente indicar um raio fraco nas proximidades, um forte mais longe, ou vice-versa.

Sistemas de radares não usam atenuação, o que pode levar, às vezes a não visualização de uma tempestade dentro de outra tempestade. Geralmente no estágio cumulus, que é livre de precipitação, radares não conseguem detectar as descargas elétricas presentes neste estágio.

As informações fornecidas por radares têm às vezes muitos minutos de idade, e no momento que está amplamente disponível, torna-se de uso limitado para aplicações em tempo real, como na navegação aérea. (WILLIAMS, 1995).

Neste trabalho, utilizaremos detectores do tipo móvel de alta tecnologia, que não é sujeito a atenuações de sinal, conforme está abordado na seção 3.1.

1.6 A FILOSOFIA DO SISTEMA DESENVOLVIDO

O apoio de terra é importante para um navegante e sua função é prover informações de qualidade em tempo real para permitir decisões especialmente em situações adversas.

No caso específico deste trabalho, cuja filosofia está centrada num sistema de monitoramento de descargas atmosféricas a partir do qual se pode inferir a severidade de tempestades, o objetivo foi desenvolver produtos e *softwares* para aquisição de dados e visualização das informações de um sensor de descargas atmosféricas comercial, de modo a torná-las eficaz para os propósitos de apoio e segurança na navegação hidroviária.

Em síntese, os produtos gerados pelo sistema devem permitir informar ao comandante da embarcação, a área ou região de ocorrência das descargas atmosféricas e a direção e velocidade de deslocamento da respectiva tempestade associada àquelas descargas.

O sistema de monitoramento, definido na fase deste trabalho é composto por dois sensores de descargas atmosféricas idênticos, do tipo Stormscope, sendo um deles fixo em terra e o outro móvel, dito embarcado.

Ambas as centrais de coleta de dados permitem o acompanhamento dos eventos em tempo real e permitem ações antecipadas para a tomada de decisão.

As informações de descargas colhidas na antena do sensor são instantaneamente apresentadas através de um display multifunção, o qual indica a localização em azimute e distância da tempestade. No modo móvel embarcado no comboio, o *display* de visualização está na cabine da embarcação e no modo fixo, em terra. A central opera no LH². Um *software* dedicado desenvolvido neste trabalho permite observar a evolução dos eventos de descargas atmosféricas e estimar, com base nas informações antecedentes de azimute e distância, o vetor direção de deslocamento da tempestade.

Desse modo, o monitoramento pode operar como um sistema de alerta para o navegante hidroviário, permitindo inferir se a severidade de condição de mal tempo está indo de encontro com a embarcação. Essa informação é suporte decisivo para o comandante da embarcação realizar a escolha em prosseguir a viagem, esperar até a tempestade dissipar, ou ir para um ponto de abrigo existente ao longo da hidrovia.

Diferente de outros sistemas, o sistema desenvolvido permite observar os dados no momento de ocorrência do fenômeno, isto é importante, pois sistemas como o RINDAT (descrito no Anexo A – RINDAT) têm taxa de atualização de ocorrências de 15 minutos. Às vezes este período de tempo em países tropicais como o Brasil já foi suficiente para formação e dissipação de uma tempestade de alta intensidade e curta duração.

2 METODOLOGIA

O sistema de alerta de descargas atmosféricas, é composto por duas centrais de aquisição de dados para acompanhar a evolução de tempestades. Uma central localizada no LH² - Laboratório de Hidrologia e Hidrometria localizado na UNESP – Universidade Estadual Paulista, Campus Ilha Solteira/SP e outra central instalada numa embarcação.

2.1 CENTRAL LH²

A central localizada no LH² possui um computador dedicado ligado a um sensor detector de descargas elétricas, do tipo Stormscope. O computador executa um *software* desenvolvido neste trabalho, conforme apresentado na seção 5, recebe os dados brutos do sensor, os decodifica e descriptografa. Em seguida, o *software* armazena e cataloga os dados de descarga para futuras análises e estudos referentes ao tema a partir do banco de dados criado.

Ademais, com o processamento e armazenamento de dados, as informações de descarga eletromagnética são plotados na tela do *software* em tempo real. Junto com os dados instantâneos, o programa plota sumariamente, em intervalos de 15 minutos, as descargas atmosféricas em cores diferentes, em função da idade de ocorrência.

Considerando a variação espaço-tempo, este modo de representação dos dados por cores diferentes em intervalos de tempos definidos, permite acompanhar a evolução e a direção das tempestades. O vetor de sentido do deslocamento da tempestade é obtido com base nas informações precedentes.

2.2 CENTRAL EMBARCADA

A central embarcada pressupõe o monitoramento das descargas atmosféricas diretamente na embarcação com o propósito de agregar segurança para o comboio, pois, permite ao comandante visualizar em tela a posição relativa das descargas eletromagnéticas e o deslocamento das células de tempestades.

Em síntese, a embarcação é instrumentada com um sensor Stormscope acoplado a um *display* multifunção no qual são visualizados os pontos de ocorrência das descargas. Essa visualização, dada em azimute e distância da embarcação, é georeferenciada e corrigida dinamicamente no percurso.

Paralelamente, há um *hardware* que foi desenvolvido com a finalidade de armazenamento e processamento das informações de descargas elétricas. O *hardware* possui como CPU um microcontrolador do tipo dsPIC, onde os dados enviados pelo sensor são catalogados conforme hora, minuto e localidade dos eventos. Estes dados são salvos em arquivo de texto comum, em um cartão de memória tipo SD.

O funcionamento do *hardware* desenvolvido, bem como suas funcionalidades, componentes e esquemas elétricos estão apresentados na seção 6.

2.3 VALIDAÇÃO DOS DADOS

O processo de validação quantitativo dos dados de descargas atmosféricas obtidos em posição de ocorrência e seus desvios de posicionamento de local de incidência em azimute e distância são comparados com os dados da rede RINDAT. Esta é a rede de sensores terrestres de descargas atmosféricas padrão no país, que opera por triangulação de amostragem entre sensores, o que maximiza a precisão na detecção dos eventos.

Outro método de comparação qualitativo pode ser testado pelo acompanhamento visual das descargas elétricas, comparando, em tempo real, os dados mostrados no *display* multifunção e no *software* desenvolvido para a central LH².

3 MATERIAIS UTILIZADOS

O sistema de alerta de tempestades em curtíssimo prazo, desenvolvido neste trabalho teve como propósito a estruturação de um projeto de baixo custo, para possibilitar aplicabilidade operacional em todas as hidrovias brasileiras.

A central de dados em terra é composta por um microcomputador para visualização e tratamento dos dados, um sensor de descarga atmosférica e um GPS para georreferenciamento do sensor.

A central embarcada é formada por um *hardware* microcontrolado dedicado, ligado a um sensor de descarga atmosférica, um *display* multifunção para a visualização das ocorrências de descarga pelo comandante da embarcação e um GPS para georreferenciamento do sensor.

Na sequência estão descritos os principais componentes do sistema.

3.1 SENSOR DE DESCARGAS ATMOSFÉRICAS STORMCOPE WX-500

O Stormcope, modelo WX-500, é um sistema móvel de detecção de descarga atmosféricas fabricado pela L-3 Communications. De fato, trata-se de um sensor específico para uso aviônico capaz de detectar descargas elétricas num raio de 100 nm ou 185,2 km. Possui uma interface serial com taxa de 9600 bps na qual pode ser feita a comunicação com um computador por exemplo.

Apesar de ter sido desenvolvido para uso exclusivo na aviação, o equipamento é perfeitamente passível de emprego em outras aplicações, como por exemplo na navegação marítima e hidroviária. O princípio das medidas do instrumento está em coletar os impulsos das assinaturas elétricas e magnéticas específicas das descargas elétricas, através de sua antena na qual detecta tanto as descargas intranuvem quanto o encontro do líder

conectante com o líder escalonado, não importando portando a altura onde a antena está disposta.

As assinaturas eletromagnéticas aqui citadas são verificadas conforme a seção 1.4.1 deste trabalho.

Na Figura 9 pode-se observar o equipamento que é composto de uma CPU responsável pelo tratamento dos dados e uma antena responsável pela amostragem dos dados de descarga atmosférica. É importante ressaltar que para um correto posicionamento da antena ao norte da embarcação, a mesma deve obedecer aos procedimentos de instalação que devem ser consultados no manual do fabricante.

Figura 9 – WX-500 e sua antena

Fonte: Adaptado de L-3 Communications Avionics Systems (1996)

Por se tratar de um equipamento comercial, a obtenção de dados referentes à construção elétrica do dispositivo é de uso exclusivo do próprio fabricante, mas segundo o manual, o dispositivo não sofre atenuação devido à construção de sua antena, que possui tecnologia patenteada.

Por não sofrer atenuação de sinal, o sensor pode amostrar tempestades na mesma posição de latitude e longitude ocorrendo em diferentes altitudes.

Há uma relação direta entre a severidade da tempestade com relação ao número de descargas por minuto.

3.2 DISPLAY MULTIFUNÇÃO – BENDIX/KING KMD 150

Este dispositivo também aviônico integra a central embarcada, é conectado ao WX-500 e tem como função permitir a visualização das descargas elétricas em sua tela.

O *display* multifunção (MFD) tem acoplado em si tem cartão de memória com os mapas cartográficos de toda a América do Sul. Pelo posicionamento cartográfico podemos traçar as rotas de navegação com o apoio de um GPS.

O MFD recebe dados de posicionamento de um GPS para orientar constantemente a posição do cursor no mapa. A apresentação dos dados de descarga elétrica pode ser ajustada nos modos *cell* e *strike* discutidos nas seções 4.1.1 e 4.1.2, respectivamente.

Por se tratar de equipamento com propósitos operacionais para aviões, os dados de descarga podem ser apenas visualizados no MFD, de modo quase que instantâneo. Isto é, o equipamento atualiza suas informações após alguns minutos descartando os eventos remotos anteriores. Para os fins científicos e de desenvolvimento desse trabalho houve a necessidade de desenvolvimento de um *hardware* dedicado para se compor um banco de dados das descargas

A Figura 10 ilustra a tela do *display* MFD – KMD 150.

Figura 10 – Bendix/King KMD 150

Fonte: Adaptado de L-3 Communications Avionics Systems (1996)

3.3 GPS GARMIN 18X PC

Este instrumento é conectado ao *display* KMD 150 através de interface serial com taxa de transmissão de 4800 bps. O *display* multifunção e o Stormscope WX-500 utilizam os dados de posição obtidas pelo GPS, posicionando o cursor de localização do Stormscope sobre o mapa cartográfico.

Este equipamento foi escolhido devido à facilidade e qualidade das informações enviada. Utiliza o protocolo NMEA 0183, que é padrão na maioria dos equipamentos comerciais, sendo de fácil busca na literatura por informações sobre o funcionamento do protocolo.

Na Figura 11 apresenta-se uma foto do GPS – Garmin 18x-PC.

Figura 11 – GPS Garmin 18x PC

Fonte: GARMIN, 2012.

4 O PROCESSO DE FILTRAGEM DE DADOS

O Stormscope WX-500 não é um sensor desenvolvido para fins científicos e sim comerciais; em seu manual de instalação e operação não é detalhado o funcionamento das palavras ou *strings* enviados pelo sensor para o *display* multifunção, nem mesmo o funcionamento de seu *hardware*.

Diante disso, para obter as informações necessárias para o desenvolvimento do trabalho foi necessário entrar em contato com a empresa fabricante do sensor Stormscope, a L3-Communications. A empresa L3-Communications, gentilmente cedeu cópia do manual de desenvolvedores, após explicações que se tratava de desenvolvimento de trabalho de pesquisa, condicionado à assinatura de um contrato de não divulgação das informações sigilosas.

Alguns pontos base de operação das strings enviadas para o Stormscope cruciais para este desenvolvimento serão discutidas na seção 4.2 deste trabalho.

4.1 MODOS DE DESCARGAS ATMOSFÉRICAS

O Stormscope envia dois tipos de dados de descarga atmosférica para o *display* multifunção, onde são selecionáveis através de um menu de operação do próprio equipamento; são eles: *cell mode* e *strike mode.*

4.1.1 Cell Mode

Em grosso modo, é um localizador condensado de das descargas elétricas. O moto é utilizado para identificar a localização das células de tempestades. É mais útil durante períodos de atividade de descarga elétrica pesada. Exibindo dados da célula durante estes períodos permite ao comandante “peneirar” uma tela cheia de pontos de descarga para melhor determinar onde as células de tempestade estão localizadas.

4.1.2 Strike Mode

Ao contrário do *cell mode* que plota um resumo das descargas, este modo plota todas as ocorrências que estão acontecendo.

O *strike mode* é mais útil durante períodos de atividade elétrica de *flash*, ou relâmpagos, pois pode mapear as descargas iniciais associadas a uma tempestade que ainda está se formando. O Stormscope plota os pontos de descarga em azimute e distância, em tempo real, na tela do *display*, permitindo ao comandante localizar e desviar das tempestades em sua rota.

4.2 DECODIFICAÇÃO DAS PALAVRAS

A partir do manual de desenvolvedores (L-3 COMMUNICATIONS AVIONICS SYSTEMS, 1996) do Stormscope WX-500 obteve-se a informações necessárias sobre o processamento de dados do sensor.

De forma geral, a palavra ou *string* transmitida pelo Stormscope para o *display* multifunção através do protocolo serial RS232 para o *hardware* tem a seguinte forma:

<STX><id><ddd><CR>(<id><ddd><CR>).<cc><ETX>

na qual:

STX é o código ASCII de início de *string*; código ASCII (02h);

id é o designador de item, ou seja, o tipo de dado enviado pelo sensor;

CR é o código ASCII para mover o cursor para a primeira posição da linha código (0Dh);

ddd é o valor do dos dados;

cc são dois caracteres maiúsculos ASCII hexadecimais obtidos pela soma de todos os caracteres depois de STX e antes de (“.”);

ETX é o código ASCII de fim de texto (03h);

() indica campos opcionais para adição de elementos de dados. Até 65 campos são permitidos entre uma combinação de STX e ETX.

Um exemplo de *string* pode ser observado por um dado colhido pelo Stormscope, apresentado a seguir:

%IPA_X

%H0000

%S153259134

%E51.03

De forma simplificada, a primeira linha define o modo de operação e de transmissão de dados referentes à descarga, no caso %I é o status do sistema, em seguida P significa “OK”, A é o modo de tempo atmosférico,

%H é o *Aircraft Heading* (rumo da proa), este valor varia de 0000 a 3599, e representa o Azimute, de 0 a 359,9 graus em sentido horário, sendo 359,9° norte.

%S modo de descarga elétrica, os três primeiros valores (153 no exemplo) indicam distância do raio em quilômetros em *cell mode*, relativa ao nariz a aeronave, ou, nosso norte embarcado, tendo valor máximo de 200 milhas náuticas de diâmetro (370,4 quilômetros), os próximos três valores (259) indicam o ângulo ou azimute em graus com o norte, variando de 000 a 359. Os três finais (134) indicam o raio no *strike mode*, também apresentando um valor máximo de 200 (quilômetros).

%E é o código de erro especificado na referência L-3 Communications Avionics Systems (1996) e .03 indica um código CRC para checagem de checksum do sinal. Este erro é utilizado geralmente para que um algoritmo do programa verifique se a mensagem foi entregue corretamente, sem erro no envio.

A partir dos dados amostrados e processados, o arquivo de Log salvo pelo *software*, o programa executa uma rotina para filtrar os dados, um exemplo de dados filtrados é apresentado a seguir:

S153,259,134

Note que foi adicionada uma vírgula entre os dados de *cell mode*, ângulo e *strike mode* de forma a ser mais perceptível à análise dos dados e os dados de *Heading* não são necessários devido ao posicionamento do sensor ser realizado por GPS.

Na próxima seção será apresentado o *software* desenvolvido, que é responsável pela amostragem, visualização e armazenamento dos dados provenientes do sensor.

5 O SOFTWARE WX-500 STORMSCOPE LOGGER

A partir da interpretação dos sinais enviados pelo Stormscope, desenvolveu-se um *software* capaz de realizar a aquisição de dados e a filtragem destas informações, salvando ambas a informações de dados e dados filtrados em arquivos comum de texto, criando um banco de dados para futura manipulação dos mesmos.

O programa também é capaz de plotar dados de descarga elétrica através de um mapa global do plugin do Google Earth®. Esta funcionalidade é responsável pelo acompanhamento espaço-temporal das tormentas ao longo do raio de atenuação do do sistema.

O sistema operacional escolhido para o desenvolvimento do programa foi o ambiente Windows, devido à popularização do mesmo, fácil acesso ao usuário comum e praticamente sem necessidade de treinamento prévio.

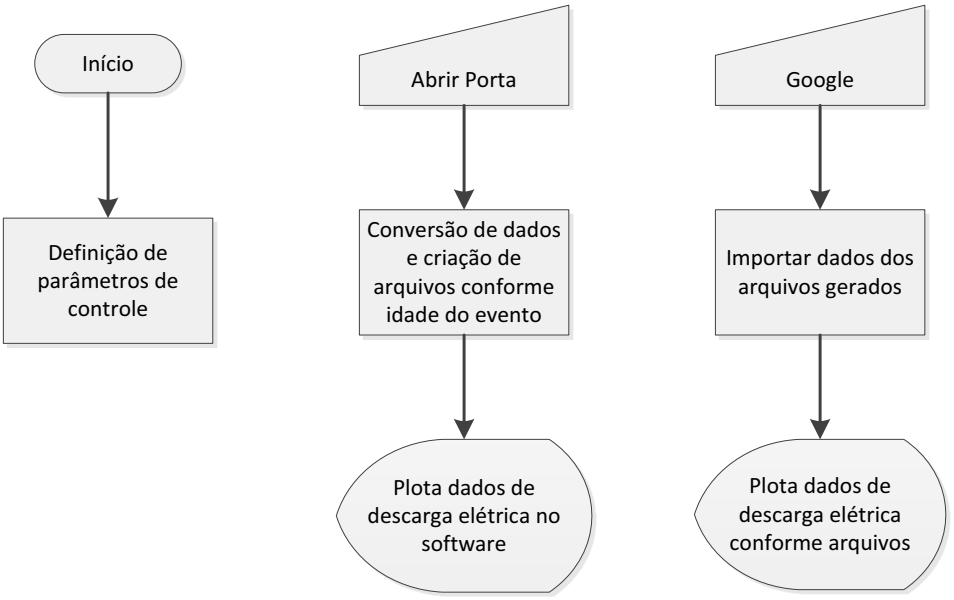
O *software* de aquisição de dados foi desenvolvido utilizando-se o *software* Embarcadero C++ Builder 2010 com direcionamento e consultas de Kolachina (2003).

5.1 DIAGRAMA EM BLOCOS

Na Figura 12, pode-se observar o diagrama em blocos do *software* desenvolvido neste trabalho.

De forma geral, o *software* ao iniciar configura suas bibliotecas internas e funções internas, ao iniciar a comunicação com o sensor os dados são capturados e convertidos conforme as opções do usuário, simultaneamente a conversão, arquivos de log são criados e dados referentes às descargas elétricas são plotadas na tela do *software* em um campo específico. Quando chamado pelo usuário o *software* chama um *browser* externo que executará em *Java Script* as rotinas de visualização das descargas através do plugin Google Earth®.

Figura 12 – Diagrama em blocos do software



Fonte: Próprio autor

5.2 APRESENTAÇÃO GERAL DO SOFTWARE

O *software* foi elaborado para ser operado da forma mais simples possível e possui as seguintes características e funcionalidades:

- seleção automática de porta serial;
- sistema de seleção de modo de captura strike ou cell;
- plotar dados de descarga em campo de log ou via plugin do Google Earth®;
- importar logs anteriores;
- importar logs provenientes do *hardware*;

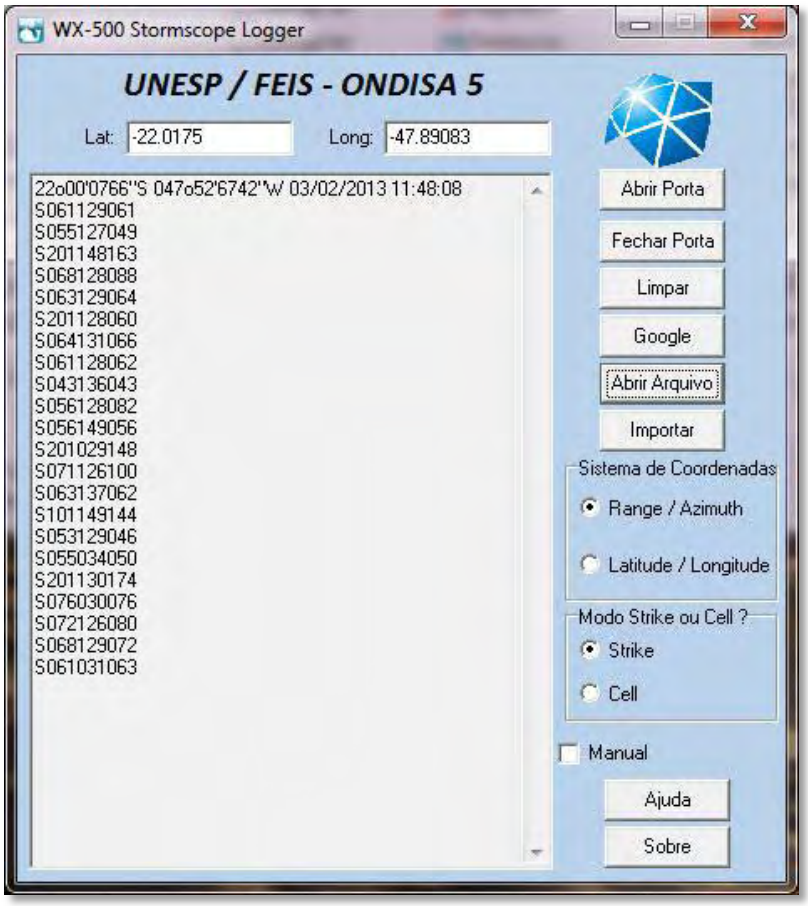
Também há funções referentes a abrir ou fechar porta de comunicação, limpar campo de log, botões de ajuda e sobre o autor.

O operador tem opção de observar os dados brutos do sensor ou mesmo a posição no momento que ocorrem. Estes dados podem ser por latitude e longitude, ou por distância

e azimute, em relação ao ponto de localização da coordenada de posição do sensor de amostragem, selecionados através do campo “Sistema de Coordenadas”.

Na Figura 13 pode-se observar a tela do *software* para importação dos dados do *hardware*.

Figura 13 – Tela principal do programa WX-500 Stormscope Logger



Fonte: Próprio autor

No desenvolvimento do *software*, optou-se pelo plugin Google Earth® para observar a evolução espaço-temporal das tempestades devido a sua abrangência, detalhamento e precisão de seus mapas. O plugin também se mostrou amigável ao usuário, por ter fácil manuseio de seus mapas.

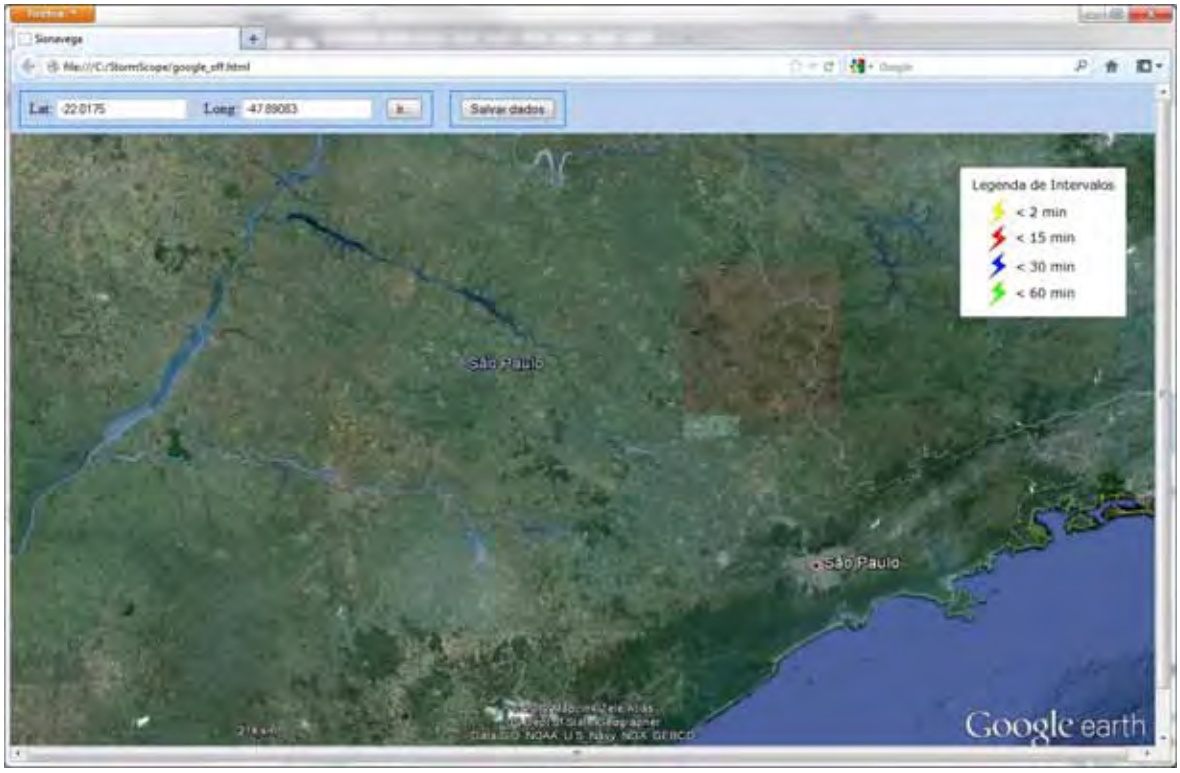
Clicando no botão Google do WX-500 Stormscope Logger, o *software* abre um navegador, na qual, através de scripts na linguagem Java, envia os dados diretamente para

o navegador, este por sua vez plota os dados em tempo real em um mapa fornecido online através do plugin, que fica alocado nos servidores do Google.

Na Figura 14 apresenta-se um exemplo do navegador com o plugin operando sem os dados de descarga elétrica.

No navegador há uma área responsável pela geolocalização manual, caso o usuário queira visualizar algum ponto específico do mapa, também há um botão para salvar o log da última hora de registros das descargas.

Figura 14 - Exemplo da tela do navegador com o plugin Google Earth®



Fonte: Próprio autor

A legenda de intervalos demonstra quatro cores distintas representando o período de tempo de ocorrência dos eventos de descargas atmosféricas. Esta diferença entre cores permite prever e observar a direção do deslocamento da tempestade. Resultados referentes aos deslocamentos das tempestades serão apresentados na seção 7.

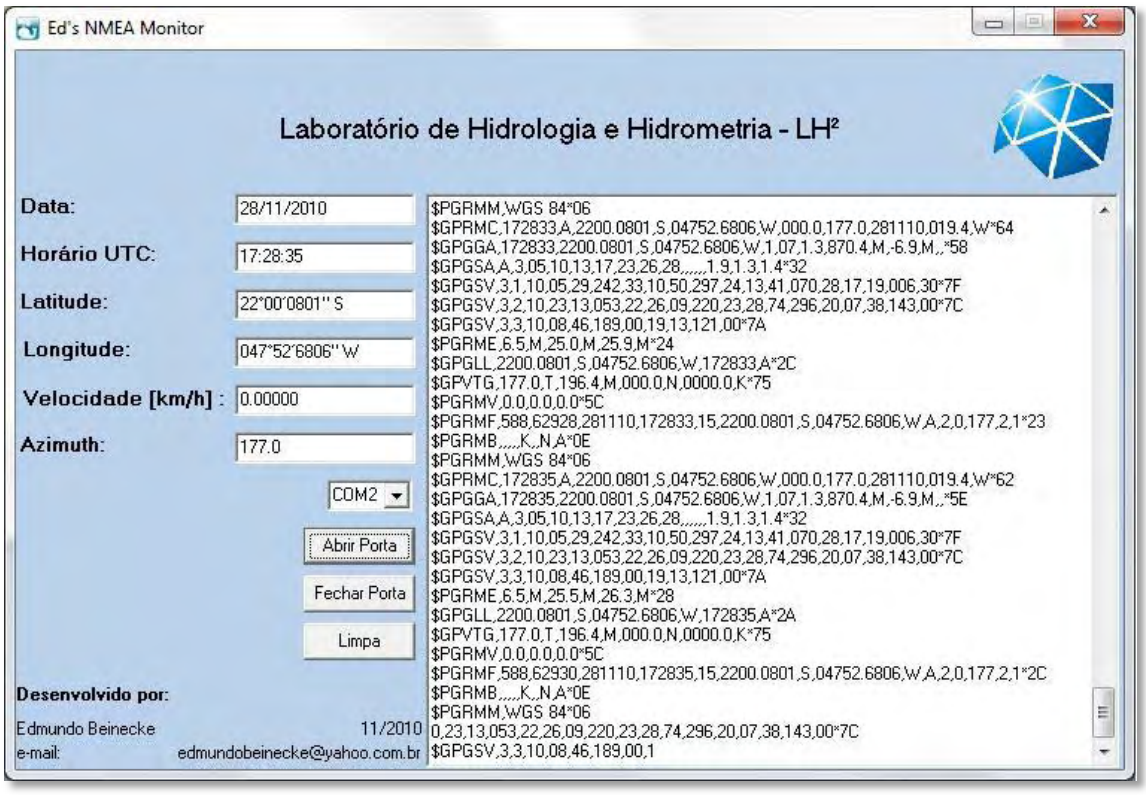
5.3 SOFTWARE DE MONITORAMENTO DE DADOS DE GPS

Este *software* foi uma adição ao projeto, para uso dos desenvolvedores e pesquisadores do laboratório LH². O programa foi desenvolvido com o propósito de inspeção rápida de dados de posição, data, hora, velocidade e azimute. Como o *software* foi desenvolvido com o intuito de ajudar em pesquisas georreferênciadas, foi adicionado um espaço onde se pode observar os dados brutos captados pelo GPS, facilitando aos pesquisadores obter outras informações relevantes.

O protocolo de comunicação utilizado para o desenvolvimento do programa foi o mundialmente adotado NMEA 0183. Este protocolo é o mais comum e disseminado entre os GPS de uso comercial. O GPS é conectado a uma porta serial de um microcomputador, ou a um conversor USB/Serial e transmite a uma taxa de 4800 bps.

A tela principal do programa é mostrada em exemplo na Figura 15.

Figura 15 – Exemplo da tela do software de monitoramento GPS



Fonte: Próprio autor

O modo de operação é semelhante ao *software* WX-500 Stormscope Logger, basta escolher a porta COM a se comunicar e clicar no botão Abrir Porta, em seguida os dados são preenchidos automaticamente nos campos de suas respectivas informações.

Na próximo seção será apresentado o *hardware* desenvolvido, que é responsável pela amostragem, e armazenamento dos dados provenientes do sensor presente na embarcação.

6 O HARDWARE DESENVOLVIDO

A outra frente de operação do sistema de alerta consiste de um eletrônica, composta por um sistema microcontrolado ligado ao sensor de descargas atmosféricas e a um GPS para georreferenciamento da embarcação.

O dispositivo tem operação automática, com função de catalogar os eventos de descarga que estão ocorrendo. O operador não necessita de forma alguma interagir com o equipamento. As strings enviadas pelo Stormscope serão direcionadas em conjunto para o *hardware* e para o *display* multifunção.

O *hardware* é responsável por decodificar e catalogar os dados de descargas atmosféricas por idade, tempo e forma, criando um banco de dados para futuras análises referentes ao tema.

O *display* multifunção instalado na cabine da embarcação, permite que o comandante observe os dados de descargas atmosféricas e sua evolução temporal, bem como sua localização justaposto ao mapa cartográfico presente no *display* multifunção.

6.1 DIAGRAMA EM BLOCOS DO HARDWARE

Antes do início do desenvolvimento do *software* embarcado, desenhou-se um diagrama em blocos no qual deveria conter a estrutura padrão de funcionamento do sistema.

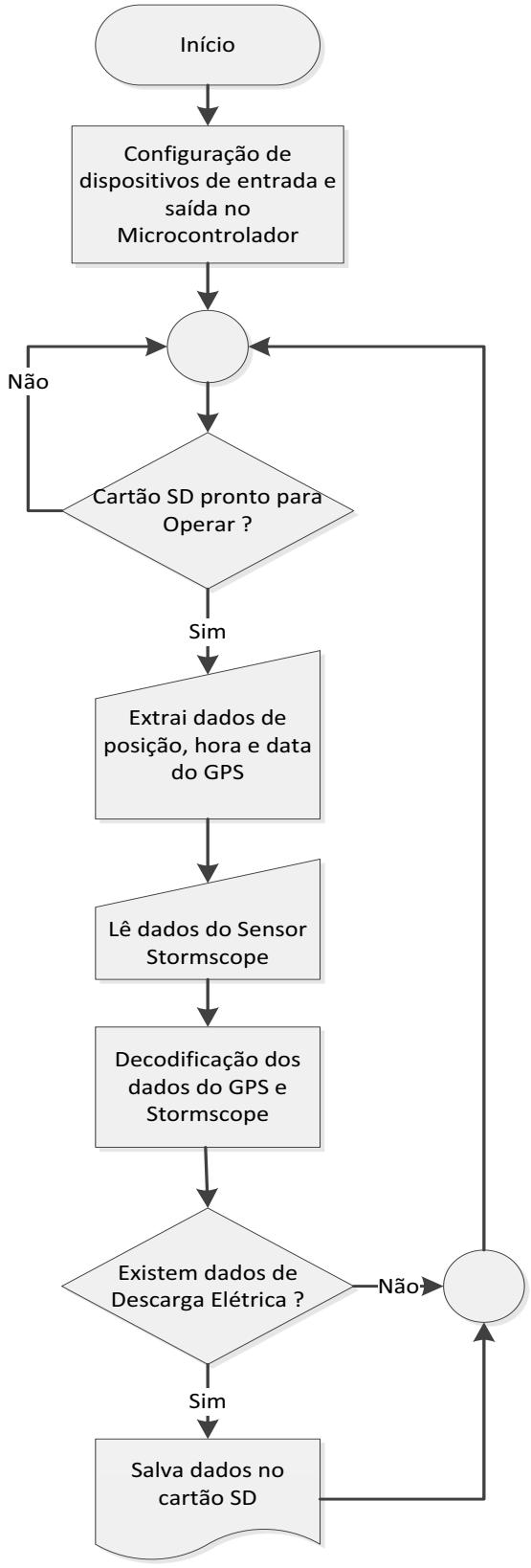
O microcontrolador do *hardware* tem um programa em sua memória que configura os equipamentos conectados, em seguida testa se há um cartão de memória SD conectado ao dispositivo através da interface SPI do CPU, caso negativo ela espera até que seja conectado um cartão de memória.

Quando um cartão SD for detectado e ele estiver funcionando adequadamente, o microcontrolador se conecta serialmente ao GPS, captura informações de posição data e hora, em seguida extrai estes dados e classifica-os para uso posterior.

No próximo passo, o microcontrolador inicia a conexão com a porta serial do Stormscope, capturando as informações de descarga se houver. Como etapa final, a eletrônica filtra os dados recebidos do WX-500, armazenando-os no cartão de memória SD, retornando ao teste do cartão para iniciar um novo ciclo de operação.

O arquivo é gravado minuto à minuto, classificado por data e hora de correspondência em um arquivo de texto comum, podendo ser utilizado por qualquer computador pessoal.

Figura 16 – Diagrama em blocos do software embarcado



Fonte: Próprio autor

6.2 DESCRIÇÃO DO DISPOSITIVO DE CAPTURA

O microcontrolador presente no *hardware* desenvolvido é responsável por receber os dados do GPS e do Stormscope através do protocolo serial RS232.

A rotina de captura das *strings* comunica primeiramente com o GPS pela porta serial número 1 do microcontrolador, com uma taxa de recepção de 4800 bps. Uma rotina desenvolvida no *software* embarcado é responsável pela aquisição e decodificação das palavras referentes a posição, em latitude e longitude, data e hora através da palavra GPRMC enviada pelo GPS de protocolo NMEA 0183. Após a decodificação dos dados a rotina devolve para o programa principal os dados prontos para serem catalogados.

Exemplo da palavra GPRMC enviada pelo GPS e utilizada pelo *hardware*:

$GPRMC,225446,A,4916.45,S,12311.12,W,000.5,054.7,160912,020.3,E*68

Onde:

225446	Hora UTC 22:54:46
A	Receptor de navegação A = OK, V = Cuidado
4916.45,S	Latitude 49 graus 16.45 min Sul
12311.12,W	Longitude 123 graus 11.12 min Oeste
000.5	Velocidade em nós por hora
054.7	Azimute em graus
160912	Data 16/09/2012
020.3,E	Desvio Magnético 20.3 graus leste
*68	Checagem de erro da palavra

De posse dos dados de posição, data e hora, o microcontrolador ativa a porta serial número 2 com taxa de 9600 bps. O programa principal executa uma rotina responsável pela aquisição e decodificação dos dados de descarga elétrica, que os separa em dados de *strike, cell* e azimute, em outras palavras, distância e ângulo do ponto central, retornando os dados capturados para o programa principal.

Caso seja encontrado algum dado de descarga elétrica, um arquivo é criado com nome intuitivo para fácil localização, o nome será sequencial em ano, mês, dia, hora e minuto, como exemplo 12090812.07S. Os arquivos são salvos em um cartão tipo SD comercial, através do protocolo SPI. Este tipo de memória foi escolhida devido ao uso comum no dia-a-dia, baixo custo e facilidade de operação.

Os arquivos serão gerados somente quando ocorrerem dados de descarga elétrica, para evitar o acúmulo de informações contendo somente dados de posição no cartão de memória.

Exemplo de conteúdo do arquivo “12082901.39S”.

22º00’0781”S 047º52’6801”W 29/08/2012 01:39:12

S109297108

S108115109

A primeira linha de cada arquivo sempre contém as informações de posição, seguidos de data e hora. Em seguida dados de descarga elétrica coletados e salvos, seguindo o padrão descrito na seção 4.2.

A comunicação, tratamento de *strings*, interface com usuário e controle dos dados é executada por um microcontrolador de 16-bits da Microchip®, dsPIC33FJ128MC202.

A escolha deste microcontrolador ocorreu devido a sua velocidade de operação, operando a 40 MIPS, pois, se faz necessário uma alta velocidade para que fosse possível captar as informações do GPS e do Stormscope, processá-las e armazená-las no dispositivo de memória, sem perder dados de descarga que são enviados a cada 2 segundos pelo sensor e gravados em arquivos de texto em intervalos de 1 minuto.

Outros motivos da escolha do microcontrolador se devem pela facilidade de se encontrar exemplos teóricos e práticos de aplicações na literatura, e pelo equipamento possuir duas portas seriais, necessárias ao desenvolvimento do trabalho, na comunicação do GPS e do Stormscope WX-500.

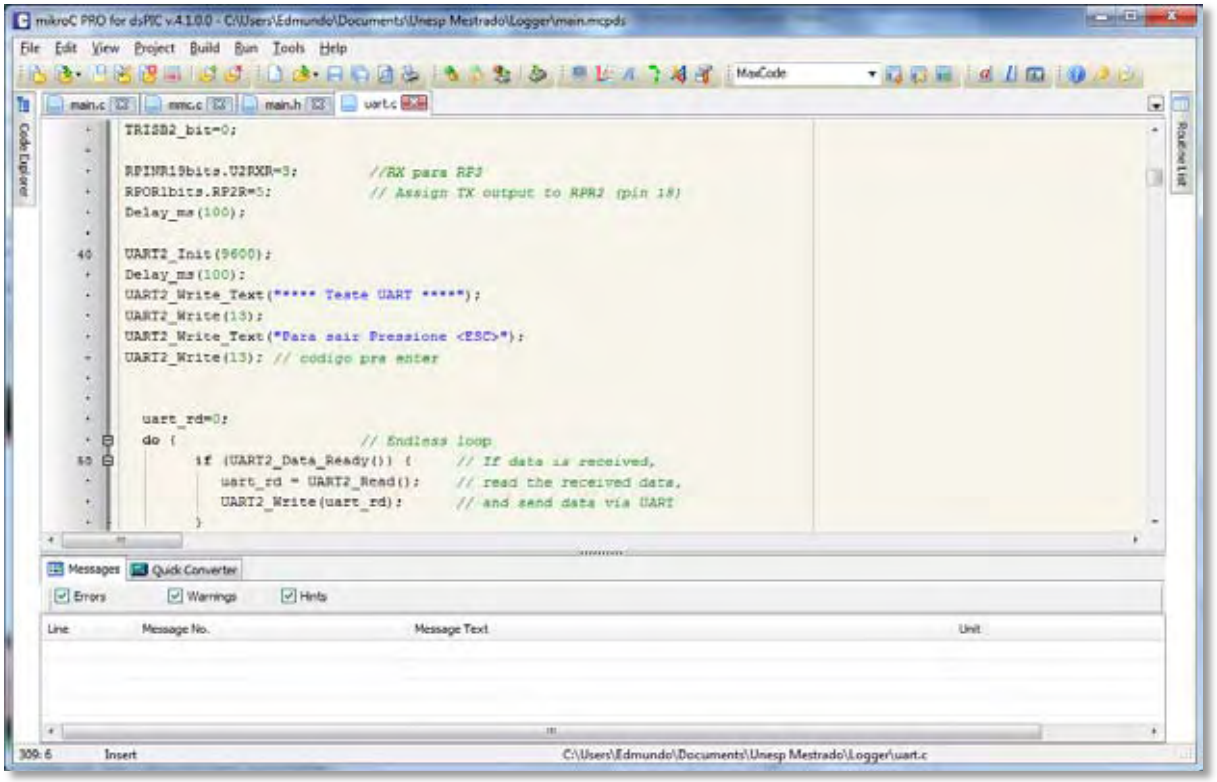
O *software* escolhido para o desenvolvimento do programa embarcado foi o mikroC PRO for dsPIC, da empresa Mikroelektronica, que forneceu uma licença de operação, bem como uma ferramenta de gravação do dispositivo, o LvPICFlash with MikroICD apresentado na Figura 17.

Figura 17 – LvPICFlash utilizado no projeto

Fonte: Mikroelektronica (2012).

Na Figura 18 visualizamos a tela de operação do programa com alguns dos trechos do *software* embarcado desenvolvido.

Figura 18 – Tela principal do mikroC Pro for dsPIC



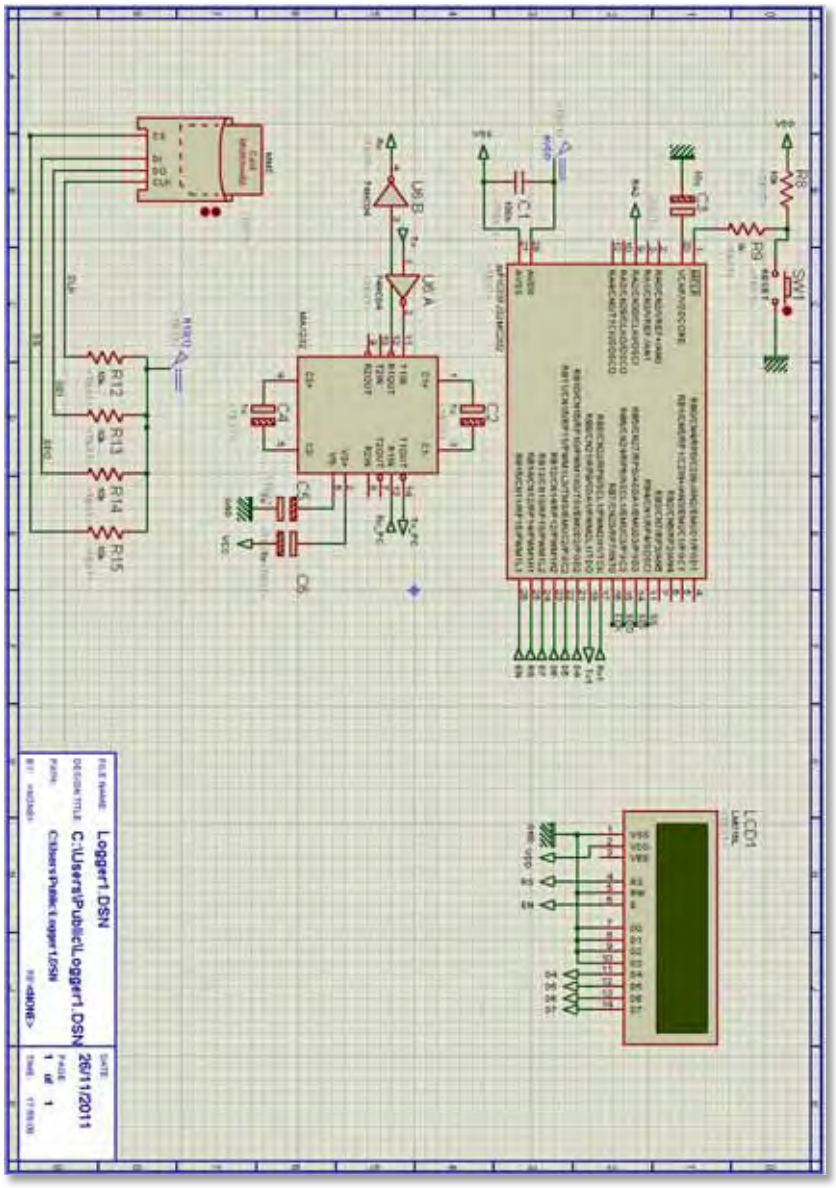
Fonte: Próprio autor

6.3 PROTÓTIPO DESENVOLVIDO

O esquema elétrico do *hardware* está apresentado na figura Figura 19, sendo que o *software* utilizado para se desenvolver e simular o sistema em operação foi o Proteus 7 Professional da Labcenter Electronics.

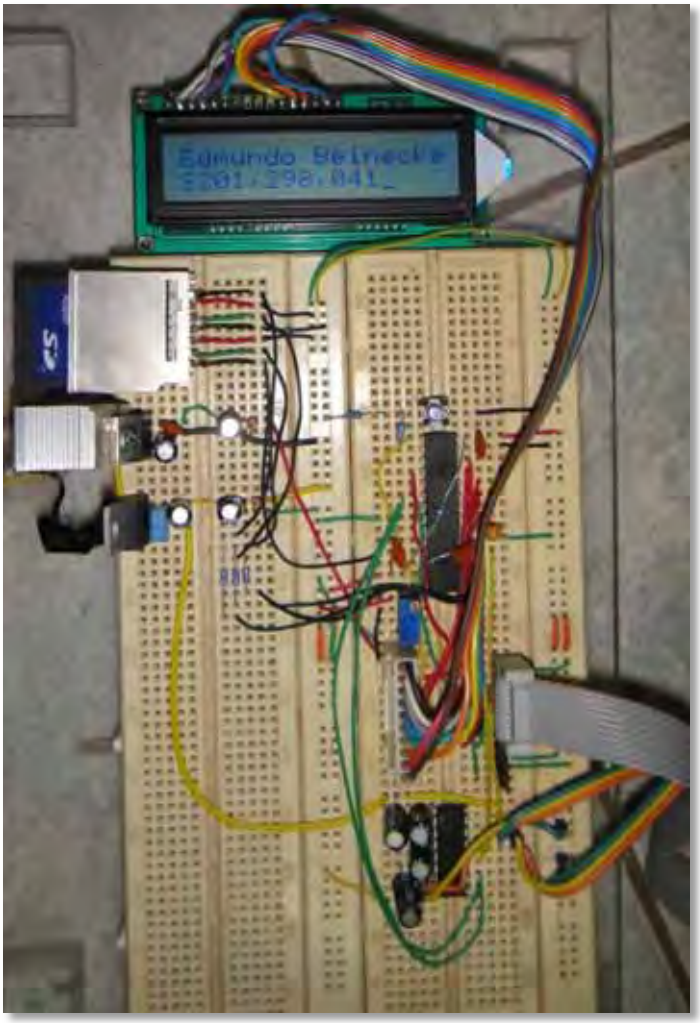
As imagens referentes ao protótipo montado em placa protoboard podem ser visualizadas nas figuras Figura 20 à Figura 22 pode-se observar o protótipo montado em protoboard e alguns dados de descarga elétrica. Alguns dados aferidos pelo GPS apresentado no *display* LCD na Figura 21 e dados aferidos pelo sensor Stormscope na Figura 22.

Figura 19 – Esquema elétrico do protótipo desenvolvido no Proteus



Fonte: Próprio autor

Figura 20 – Protótipo do *Hardware*



Fonte: Próprio autor

Figura 21 – Dados de latitude e longitude aferidos

Fonte: Próprio autor

Figura 22 – Dados de descarga elétrica aferido

Fonte: Próprio autor

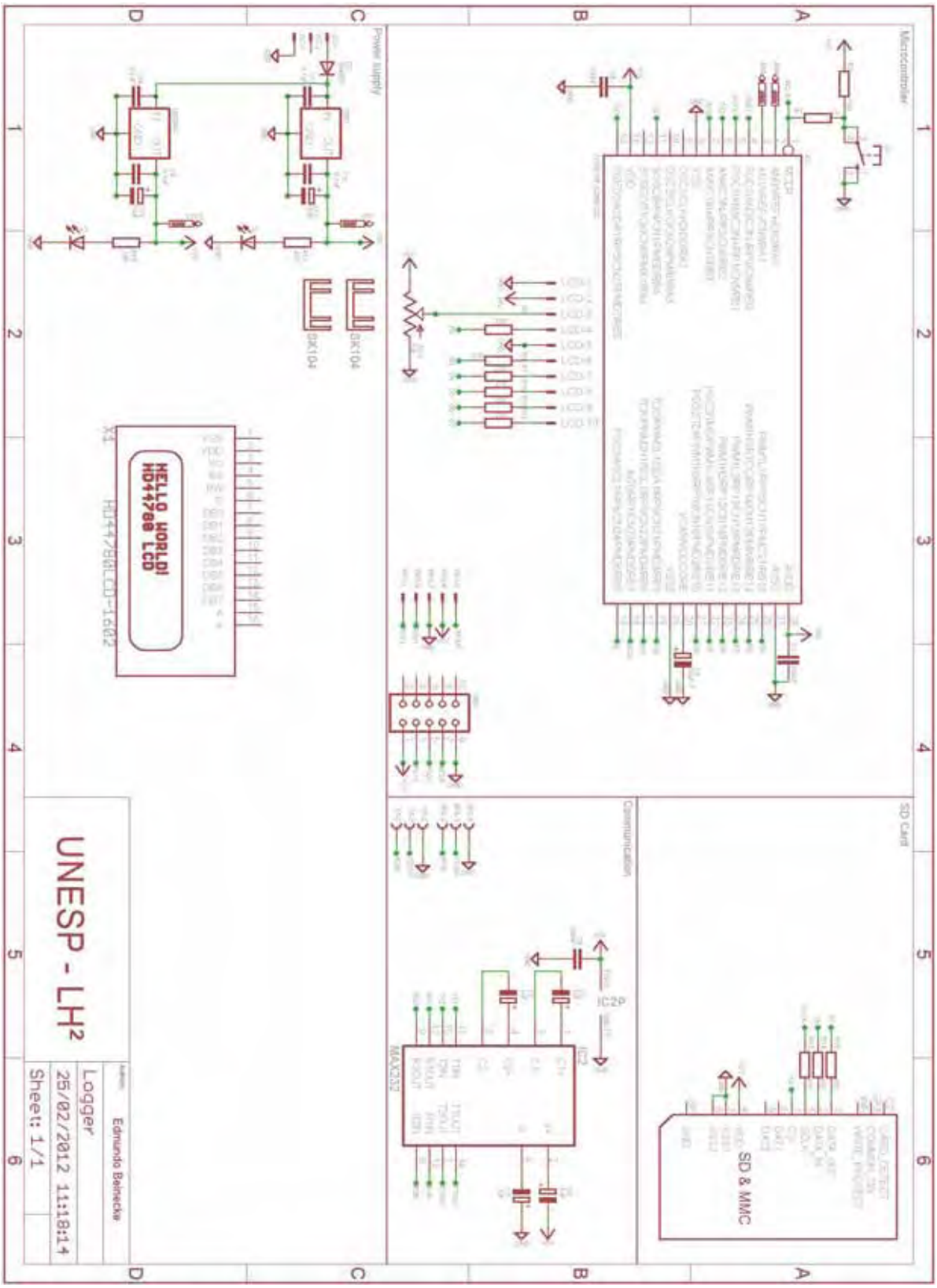
O GPS e o sensor Stormscope se conectam em um MAX232, que converte o sinal digital de 0 à 5V em um sinal utilizado pelo padrão serial RS232 que utiliza tensões entre -7 à 7V.

O cartão SD é conectado através das portas SPI do microcontrolador, este também se conecta ao *display* LCD com 4 bits paralelos responsáveis pelo envio das palavras.

6.4 HARDWARE FINAL

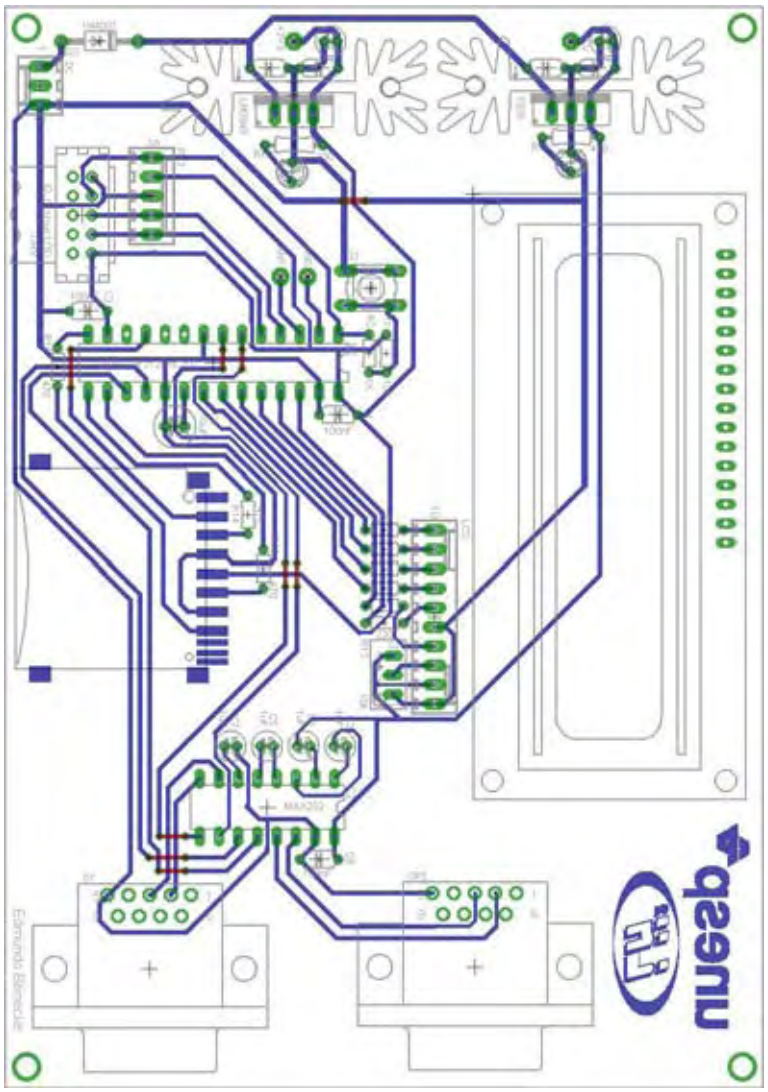
Após os testes com o protótipo terem sido concluídos, desenvolveu-se a versão final do sistema, o esquema elétrico e a placa de circuito impresso foram desenvolvidos utilizando o software EAGLE 6.20, conforme mostra as Figura 23 à Figura 25.

Figura 23 – Esquema elétrico desenvolvido no Eagle



Fonte: Próprio autor

Figura 24 – Placa de circuito impresso desenvolvido



Fonte: Próprio autor

Figura 25 – *Hardware* em funcionamento

Fonte: Próprio autor

7 RESULTADOS EXPERIMENTAIS

Uma dúvida no desenvolvimento do trabalho ocorreu por hipótese quando a embarcação estivesse de transposição de uma barragem, na eclusa de uma hidrelétrica, onde o campo eletromagnético gerado é relativamente alto. Levantou-se então a questão se essas emissões iriam interferir no funcionamento do sistema.

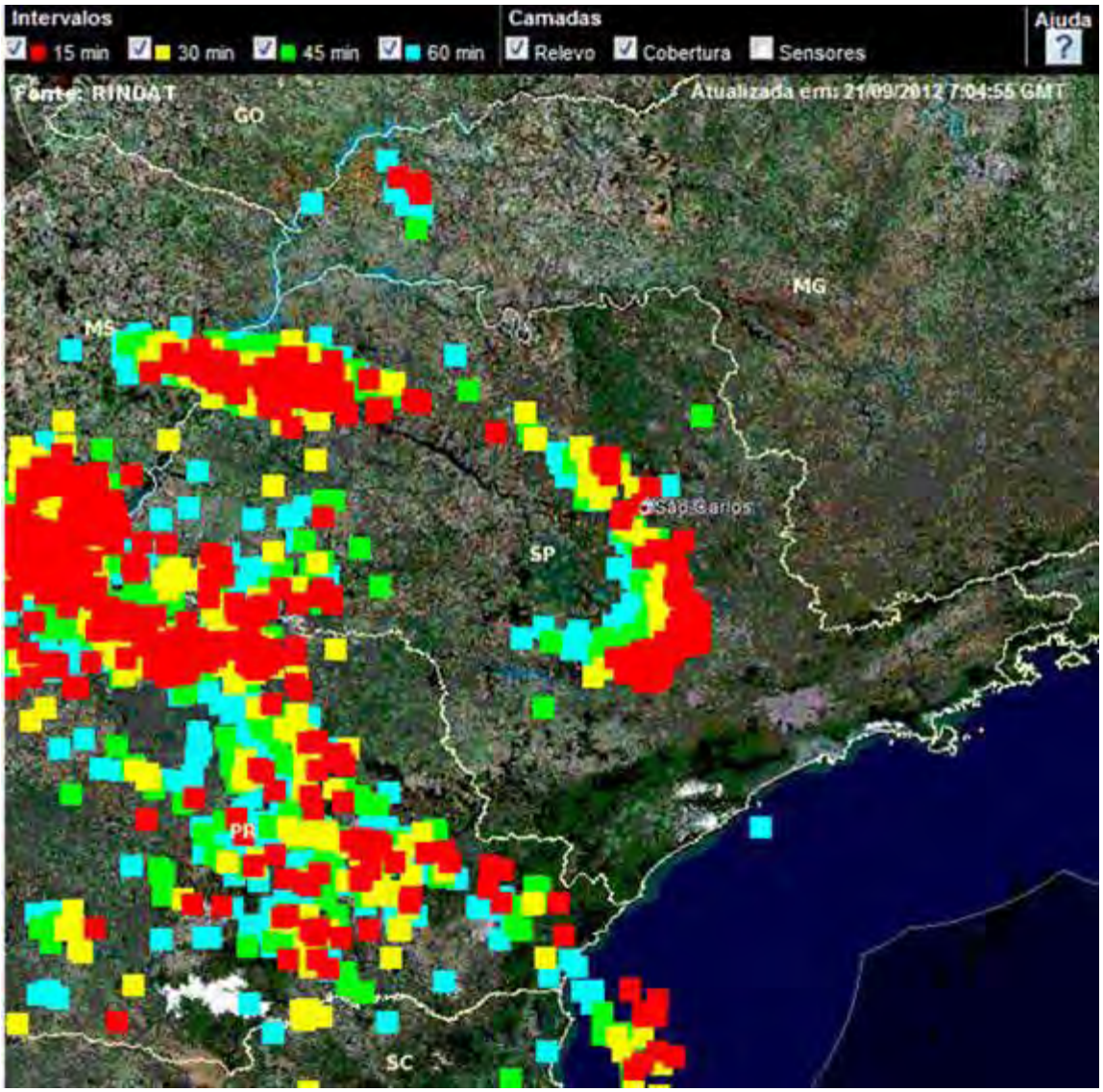
Este tipo de travessia por comboios chamada de eclusagem é comum ao longo das hidrovias brasileiras, devido a matriz energética do país ser composta principalmente por energia proveniente de hidroelétricas ao longo dos rios.

O sistema foi testado na hidroelétrica de Barra Bonita/SP. Observou-se que de fato ocorre uma interferência devido às emissões eletromagnéticas, mas estas não passavam de poucos pontos randômicos observados no *display* multifunção. Este erro ou sinais adicionais não devem causar dificuldades de interpretação, nem ao funcionamento do sistema, sendo irrelevante ao sistema como um todo.

A seguir serão apresentados resultados do software WX-500 Stormscoppe Logger apresentado na seção 5. Os dados foram cotejados com as informações do sistema RINDAT no momento de atuação da tormenta. A tormenta ocorreu em 21/09/2012 e os dados amostrados ao longo do dia.

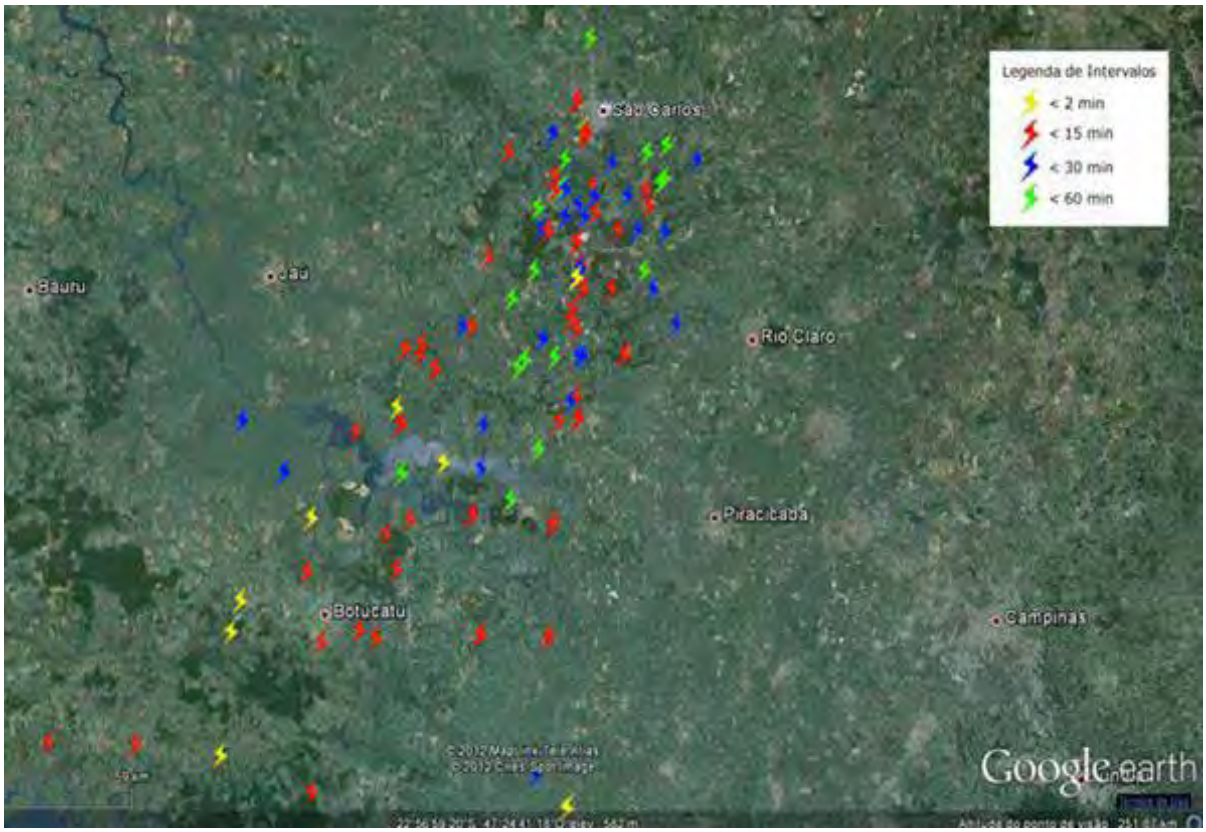
No momento de operação do dispositivo o centro de localização foi direcionado para São Carlos, cidade onde o sistema de alerta fixo desenvolvido estava operante.

Figura 26 – Descargas amostradas pelo sistema RINDAT em 21/09/2012



Fonte: RINDAT (2005).

Figura 27 – Descargas amostradas pelo sistema desenvolvido em 21/09/2012



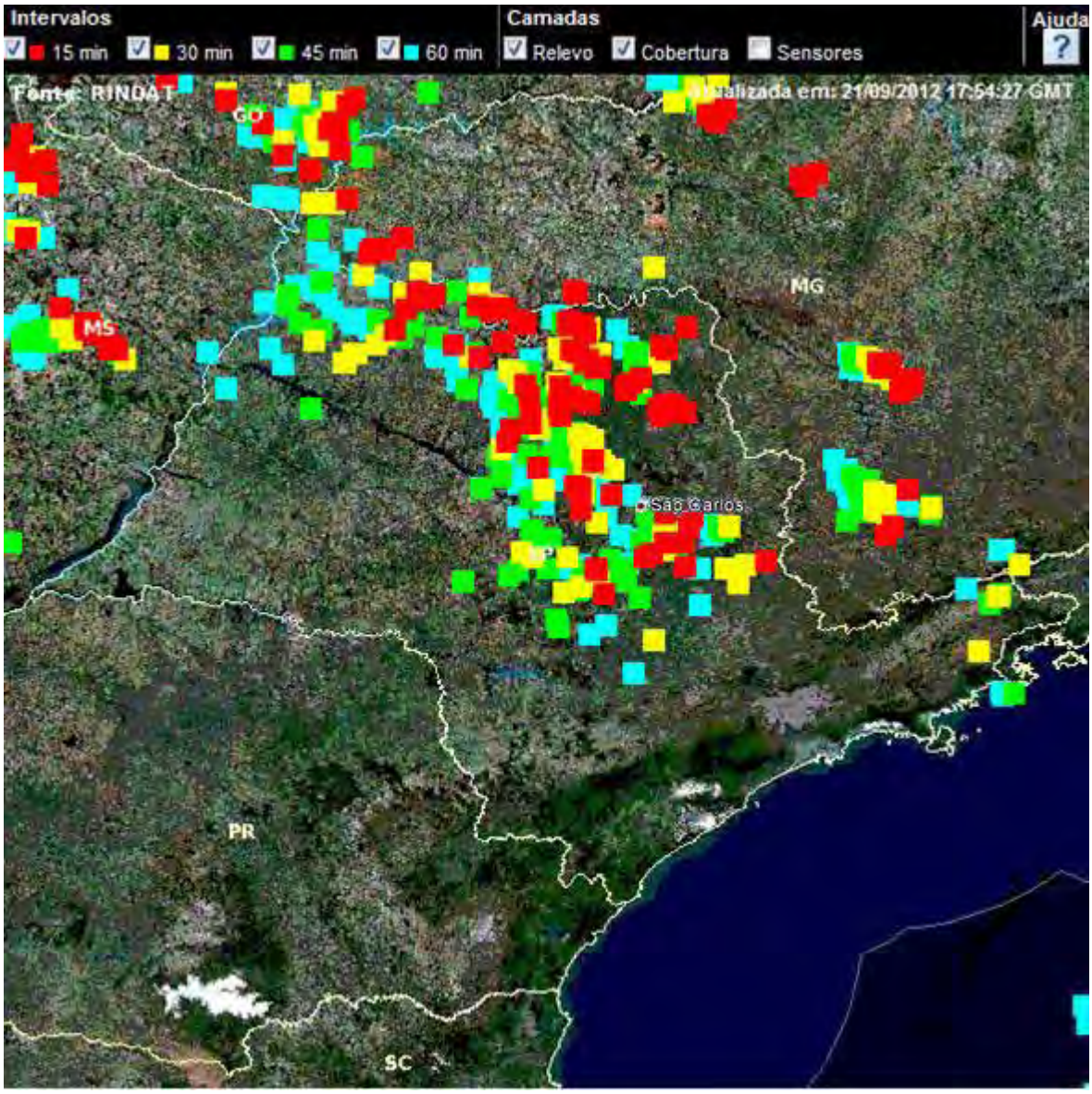
Fonte: Próprio autor

Na Figura 26 pode-se observar o deslocamento da tempestade na direção sudeste do estado de São Paulo pelo sistema RINDAT. Na Figura 27 observa-se o mesmo tipo de deslocamento da tempestade em direção sudeste do estado de São Paulo.

Comparando-se os resultados com os dados de monitoramento RINDAT, verificou-se o mesmo padrão de incidência de descargas para a região de operação.

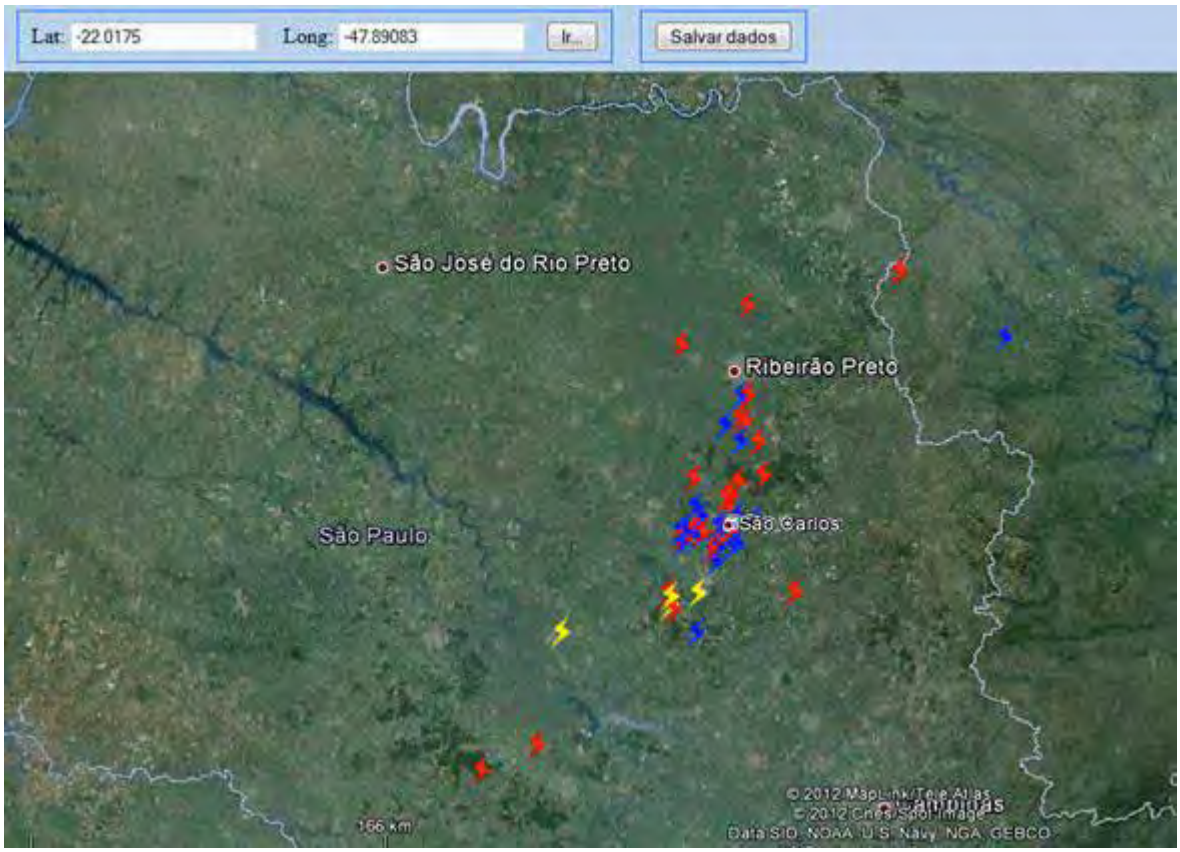
Outra ocorrência no mesmo dia da tormenta do dia 21/09/2012 por volta das 14 horas e serão relatadas entre as Figura 28 à Figura 30.

Figura 28 – Descargas amostradas pelo Sistema RINDAT , em 21/09/12



Fonte: RINDAT (2005).

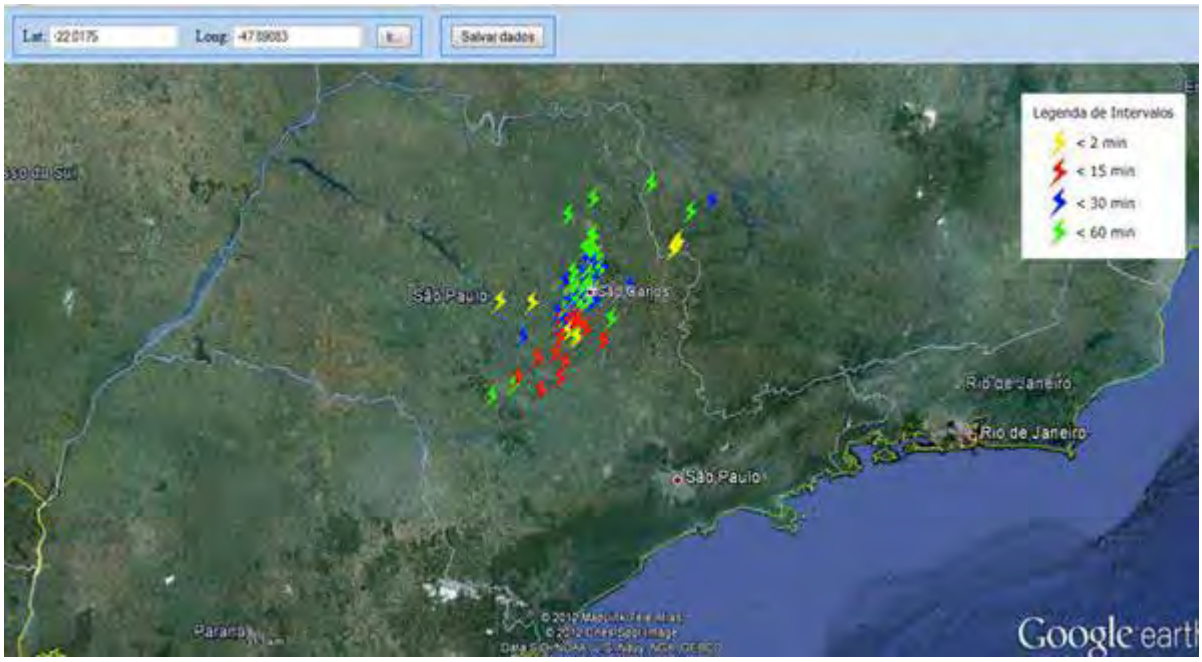
Figura 29 – Descargas amostradas pelo sistema desenvolvido em 21/09/12



Fonte: Próprio autor

Novamente podemos observar nas Figura 29 e Figura 30 o mesmo tipo de deslocamento, desta vez a tormenta está deslocando para o nordeste da região do estado.

Figura 30 – Descargas amostradas pelo sistema desenvolvido em 21/09/12

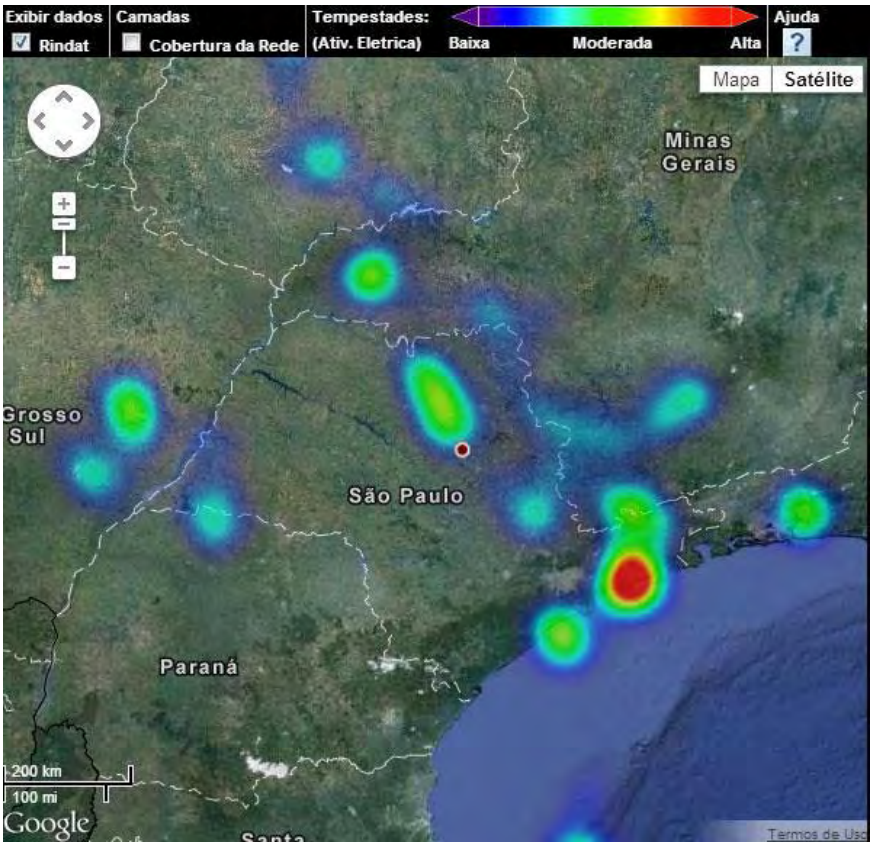


Fonte: Próprio autor

Em meados de janeiro de 2013, o modo de apresetanção das descargas elétricas do sistema RINDAT mudou de aparência, ao invés de pontos de descarga sendo plotados no mapa, o sistema agora plota de hora em hora um gradiente de concentração de descargas.

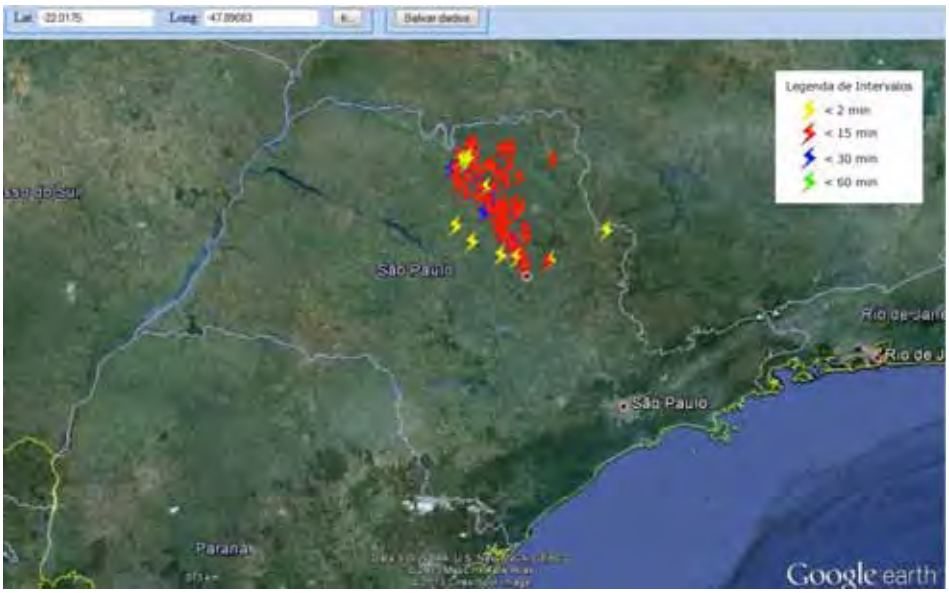
As figuras Figura 31 à Figura 36 mostram a atividade elétrica ao longo de 3 horas seguidas do dia 10/02/2013 equiparando-se os dois sistemas em sequência hora à hora.

Figura 31 – RINDAT 10/02 às 20hrs



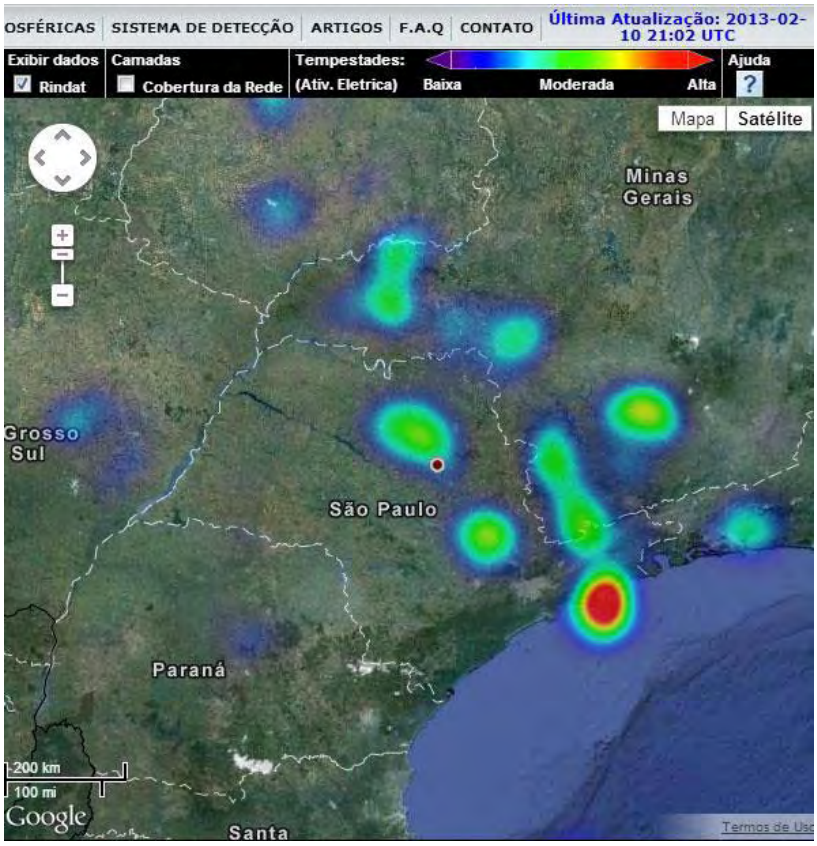
Fonte: Próprio autor

Figura 32 – Aquisição pelo sistema 10/02/2013 às 20hrs.



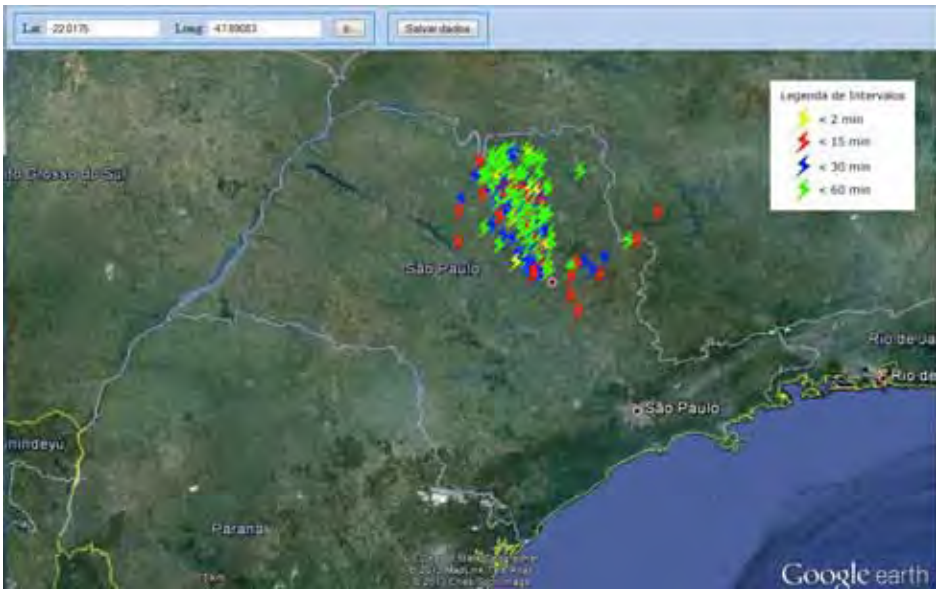
Fonte: Próprio autor

Figura 33 – RINDAT 10/02 às 21hrs



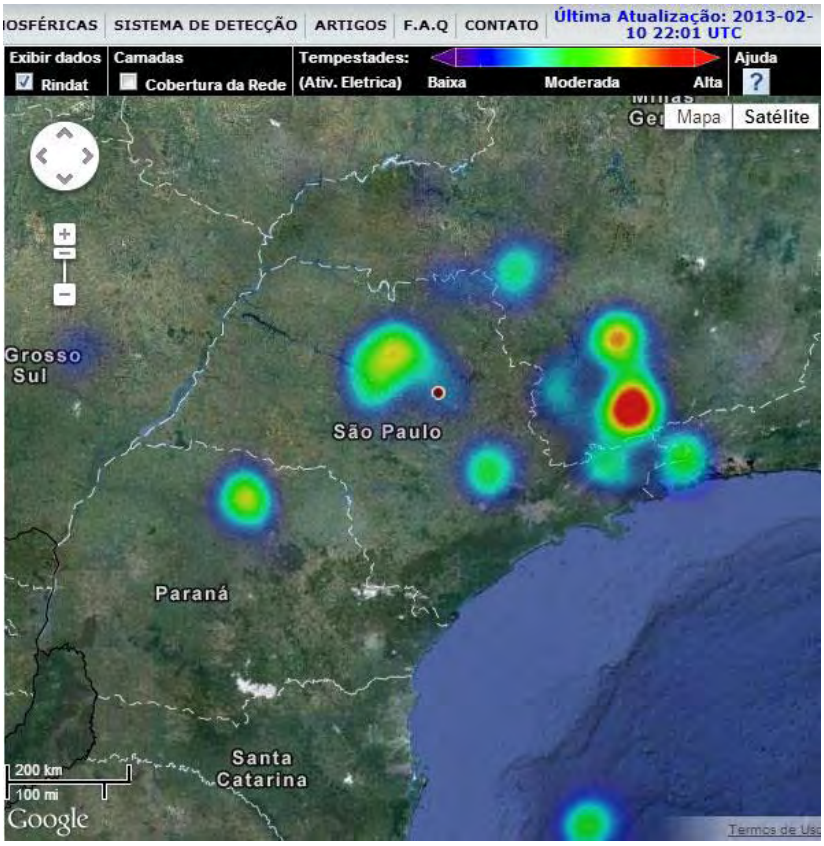
Fonte: Próprio autor

Figura 34 – Aquisição pelo sistema 10/02/2013 às 21hrs.



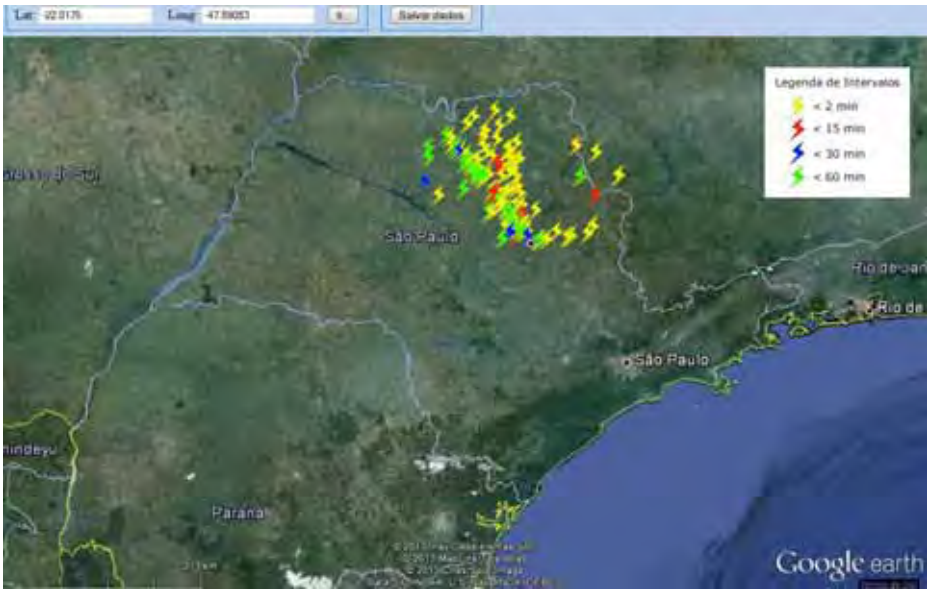
Fonte: Próprio autor

Figura 35 – RINDAT 10/02 às 22hrs



Fonte: Próprio autor

Figura 36 – Aquisição pelo sistema 10/02/2013 às 22hrs.



Fonte: Próprio autor

No novo sistema de visualização RINDAT, apesar de haverem melhorias nos aspectos de visualização da intensidade das tempestades, perdeu-se o sentido de deslocamento das tormentas, não é mais possível prever para onde uma tempestade está se direcionando. Este fato trás mais valia ao trabalho, visto que o sistema é atualizado de minuto a minuto e permite a visualização do deslocamento das tormentas.

8 CONCLUSÃO

Os softwares desenvolvidos atenderam as especificações que lhe foram designadas de forma a possibilitar análises futuras de dados sobre descargas atmosféricas, bem como a criação de um banco de dados referentes a estas informações.

Com o modo monitor do software, juntamente com dados relativos à Rede Integrada Nacional de Detecção de Descargas Atmosféricas (RINDAT), criou-se a possibilidade de montar um sistema integrado de segurança referente à malha hidroviária do Estado de São Paulo e também a possibilidade de se monitorar e acompanhar o desenvolvimento e evolução de tempestades.

O *hardware* desenvolvido possibilita a formação de um banco de dados ao longo das hidrovias. Posteriormente, estes dados poderão ser analisados através do software desenvolvido e cotejados com dados recentes de descargas elétricas. Para o navegante a instrumentação de descargas elétricas que estará presente em sua cabine será de grande valor na tomada de decisões sobre suas ações na embarcação.

Considerando que na região da Hidrovia Tietê – Paraná as tempestades provocadas pelos sistemas convectivos são de difícil previsibilidade, pois crescem e retrocedem rapidamente, é recomendável empregar o esforço técnico-cientifico desenvolvido neste trabalho, de forma a 75torm-lo um sistema operacional em futuro próximo.

Especialmente, no momento em que se iniciarem as operações de transporte de cargas perigosas na Hidrovia Tietê-Paraná (Corredor do Etanol – Transpetro) um sistema de monitoramento e alerta de descargas atmosféricas dedicado ao transpote hidroviário tem aplicação direta nos processos de carga e descarga e na passagem das embarcações em pontos críticos da hidrovia.

Um sistema de previsão de curtíssimo prazo (*nowcasting)* ancorado em dados de descargas atmosféricas deve representar um extraordinário avanço a questão da segurança hidroviária, diminuindo e prevenindo a ocorrência de acidentes.

Uma possível continuação deste trabalho seria o desenvolvimento de vários sistemas interligados, formando um sistema de maior abrangência e baixo custo de forma a abranger estados e regiões do país e do mundo que não possuem este tipo de sistema de detecção de tormentas.

A parte relativa ao desenvolvimento dos softwares de controle, aquisição de dados e visualização das informações estão concluídas e apresentadas nos apêndices que se seguem.

REFERÊNCIAS

BRASIL. Agência Nacional de Transportes Terrestres - ANTT. **Transporte ferroviário**. Brasília, DF, 2012. Disponível em: <http://appweb2.antt.gov.br/carga/ferrroviario/ferroviario.asp>. Acesso em: 4 mar. 2012.

CHADE, J. Estudo prevê o Brasil como o 4. PIB mundial em 2050. **O Estado de São Paulo**, São Paulo, Seção Economia, 07 jan. 2011. Disponivel em: <http://economia.estadao.com.br/noticias/economia%20brasil,estudo-preve-o-brasil-como-o-4-pib-mundial-em-2050,50225,0.htm>. Acesso em: 12 nov. 2011.

GARMIN. **Garmin 18x PC GPS navigator uni**t. Salem, 2012. Disponível em: <http://www.amazon.com/Garmin-18x-GPS-Navigator-Unit/dp/B0016NYHVS>. Acesso em: 4 mar. 2012.

INSTITUTO NACIONAL DE METEREOLOGIA - INMET. **Atlas de nuvens.** Brasília, DF, 2012. Disponivel em: <http://www.inmet.gov.br/html/informacoes/sobre_meteorologia/atlas_nuvens/atlas_nuvens.html>. Acesso em: 4 mar. 2012.

KOLACHINA, S. S. **C++ Builder 6:** developer's guide. Plano: Wordware Publishing, 2003.

L-3 COMMUNICATIONS AVIONICS SYSTEMS, I. **Stomrscope WX-500 Interface:** developer's guide. Grand Rapids: BFGoodrich, 1996.

MIKROELEKTRONIKA. LV24-33 programmer. Beograd, 2012. Disponível em: <http://www.mikroe.com/products/view/50/lvpicflash-with-mikroicd/>. Acesso em: 4 mar. 2012.

NACCARATO, K. P.; PINTO JR., O. Lightning detection in Southeastern Brazil from the new Brazilian Total Lightning Network (BrasiIDAT). In: INTERNATIONAL CONFERENCE ON LIGHTNING PROTECTION - ICLP, 2012, Vienna. **Proceedings…** Vienna: ALDIS, 2012.

OLIVA, J. A. B. Cenário atual do transporte hidroviário brasileiro. In: SEMINÁRIO INTERNACIONAL EM LOGÍSTICAC AGROINDUSTRIAL - “O Transporte Hidroviário(Fluvial e Cabotagem)de Granéis Agrícolas", 5., 2008, Piracicaba. **Anais...** Piracicaba: Esalq, 2008. p. 8-13.

OLIVEIRA, B. M.; TROVATI, L. R. Rastreamento satelital de embarcações e modelagem. In: CONFERENCIA IBEROAMERICANA EN SISTEMAS, CIBERNÉTICA E INFORMÁTICA, 10., 2011, Orlando. **Anais...** Orlando: [s.n.], 2011.

PINTO JR., O. **A arte da guerra contra os raios**. São Paulo: Oficina de Textos, 2005.

RIBEIRO, P. C. C.; FERREIRA, K. A. Logística e transporte: uma discussão sobre os modais de transporte e o panorama brasileiro. In: ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO, 22., 2002, Curitiba. **Anais...** Curitiba: [s.n.], 2002.

REDE INTEGRADA NACIONAL DE DETECÇÃO DE DESCARGAS ATMOSFÉRICAS - RINDAT. São José dos Campos, 2012. Disponível em: <http://www.rindat.com.br/>. Acesso em: 26 nov. 2011.

ROMERO, F. Avaliação do comportamento dos campos eletromagnéticos gerados por descargas atmosféricas nuvem-terra. In: SIMPÓSIO BRASILEIRO DE SISTEMAS ELÉTRICOS, 2006, Campina Grande. **Anais...** Campina Grande: UFCG, 2006. p. 35-53.

UNIVERSIDADE DE LISBOA. Faculdade de Ciências. Departamento de Engenharia Geográfica, Geofísica e Energia. **As nuvens**. Lisboa, 2012. Disponível em: <http://geofisica.fc.ul.pt/informacoes/curiosidades/nuvens.htm>. Acesso em:

WILLIAMS, E. R. **Meteorological aspects of thunderstorms**. Boca Raton: CRC Press, 1995.

APÊNDICE A – SOFTWARE EMBARCADO

Códigos fontes do software embarcado e seus respectivos arquivos serão adicionados a seguir nos tópicos à frente.

- Arquivo main.c é o programa principal responsável pela inicialização do *hardware*, chamada de rotinas de GPS, Stormscope e gravação de dados em cartão SD.

- Arquivo main.h é o elo entre as rotinas e o programa principal.

- Arquivo mmc.c é responsável pelo configuração da inicialização e rotinas refrente a gravação dos dos em um cartão SD.

- Arquivo uart.c é responsável pela incilização das portas seriais do Stormscope e GPS, também responsável pela aquisição e tratamento dos dados do Stormscope e GPS.

APÊNDICE A1 – ARQUIVO MAIN.C

```
#include “main.h”
#include “__Lib_FAT32.h”


 gpslog posi;
 79tormscope storm;


void main() {

 short status=-1;
 short fhandle=-1;
 unsigned long swapStartSc=0;
 unsigned char buf[1024];
 unsigned int d;
 char bytes_in_cluster;unsigned long freeClusters;unsigned long freeBytes;
 unsigned long totalClusters,badClusters;

 unsigned int i=0;
```

```
int e=0;
unsigned char ed[4];


// Configure Oscillator to operate the device at 40Mhz
// Fosc= Fin*M/(N1*N2), Fcy=Fosc/2
// Fosc= 7.37*43/(2*2)=80Mhz for 7.37 input clock
//
  PLLFBD=43;                    // M=43
  CLKDIVbits.PLLPOST=0;          // N1=2
  CLKDIVbits.PLLPRE=0;           // N2=2
  OSCTUN=0;                      // Tune FRC oscillator, if FRC is used

PORTB = 0;
TRISB = 0;
PORTA = 0;
TRISA = 0;
//LATA2_bit = 0;
ADPCFG = 0xFFFF;                 // Configura AN como digital

//AD1PCFGL = 0x3F;

LCD_init();
Delay_ms(20);

//uart_off();
//gps_off();

gpsinit();      // aparentemente a uart precisa ser ligada antes do mmc apos da pau
uart2_inicio();


mmcinit();
Spi1_Init_Advanced(_SPI_MASTER,                _SPI_8_BIT,                _SPI_PRESCALE_SEC_1,
_SPI_PRESCALE_PRI_16,
_SPI_SS_DISABLE,               _SPI_DATA_SAMPLE_MIDDLE,               _SPI_CLK_IDLE_HIGH,
_SPI_ACTIVE_2_IDLE);
Delay_ms(20);

do {
    Delay_ms(750);
    e=FAT32_init();
    inttostr(e,ed);

    Lcd_Cmd(_LCD_CLEAR);             // Clear display
    Lcd_Out(1,1,"Cartao SD");
    Lcd_Out(2,1,"Presente ?");
    if (e == 0) {
        //Lcd_Out(2,2,ed);
        Lcd_Out(2,14,"SIM");
        Delay_ms(1000);
    }
    else if (e == -1) {
        //Lcd_Out(2,2,ed);
        Lcd_Out(2,14,"NAO");
        Delay_ms(1000);
    }
}while (e != 0);
```

```
Spi1_Init_Advanced(_SPI_MASTER, _SPI_8_BIT, _SPI_PRESCALE_SEC_1,
_SPI_PRESCALE_PRI_16,
_SPI_SS_DISABLE, _SPI_DATA_SAMPLE_MIDDLE, _SPI_CLK_IDLE_HIGH,
_SPI_ACTIVE_2_IDLE);
//Spi1_Init_Advanced(_SPI_MASTER, _SPI_8_BIT, _SPI_PRESCALE_SEC_1,
_SPI_PRESCALE_PRI_4,
//_SPI_SS_DISABLE, _SPI_DATA_SAMPLE_MIDDLE, _SPI_CLK_IDLE_HIGH,
_SPI_ACTIVE_2_IDLE);
Delay_ms(20);

Lcd_Cmd(_LCD_CLEAR);              // Clear display
Lcd_Out(1,1,"Verificando");
Lcd_Out(2,1,"Cartao SD ");
FAT32_ScanDisk(&totalClusters, &freeClusters, &badClusters);       // retrieve cluster info
Lcd_Out(2,11,"OK !");
Delay_ms(1000);

Lcd_Cmd(_LCD_CURSOR_OFF);         // Cursor off

 while(1){
      //gpsinit();
      for (i=0;i<16;i++) {
         posi.lat[i]=0;
         posi.lat1[i]=0;
         posi.lon[i]=0;
         posi.lon1[i]=0;
         posi.date[i]=0;
         posi.date1[i]=0;
         posi.hora[i]=0;
         posi.hora1[i]=0;
         posi.nomearquivo[i]=0;
      }

      //UART_Set_Active(&UART1_Read, &UART1_Write, &UART1_Data_Ready, &UART1_Tx_Idle);
      //Delay_ms(50);
      posi=gprmc();                  // capta dados do gps (lat,lon,data,hora)
      //gps_off();
      //for (i=0; i<801; i++) { storm.ret[i] = 0; } // Limpar buffer da leitura anterior
      i=0;
      while (i<1024) {
           buf[i] = 0;
           i++;
      }
      //for (i=0; i<1024; i++) { buf[i] = 0; } // Limpar buffer da leitura anterior
      //uart2_inicio();
      //UART_Set_Active(&UART2_Read, &UART2_Write, &UART2_Data_Ready, &UART2_Tx_Idle);
      //Delay_ms(50);
      //for (i=0; i<801; i++) { storm.ret[i] = 0; } // Limpar buffer da leitura anterior
      storm=stormlogger();
      //uart_off();
      i=0;
      Lcd_Cmd(_LCD_CLEAR);
      Lcd_Out(1,1,"Logando");
      d=0;
      i=0;
      status=FAT32_Exists(posi.nomearquivo);
      while(d<801 && i!=2) {
          if (storm.ret[d]=='S') {
             i++;
          }
```

```
            d++;
        }
        if (i==2) { // se achou algum dado de storm cria arquivo e salva

          if (status==0) {          //se o arquivo nao existe grava coodernadas
                strcat(buf,posi.lat1);
                strcat(buf,posi.lon1);
                strcat(buf,posi.date1);
                strcat(buf,posi.hora1);
          }
          strcat(buf,storm.ret);
          i=0;
          while (buf[i]!=0) { i++; }

          fhandle = FAT32_Open(posi.nomearquivo, FILE_APPEND);
          Delay_ms(25);
          FAT32_Write(fhandle, buf, i+2);
          //Delay_ms(50);
          FAT32_Close(fhandle);

        }
        Delay_ms(350);
        Lcd_Cmd(_LCD_CLEAR);              // Clear display
        posi.lon1[13]=0;
        posi.lon1[14]=0;
        Lcd_Out(1,1,posi.lat1);
        Lcd_Out(2,1,posi.lon1);

    }
}
```

APÊNDICE A2 – ARQUIVO MAIN.H

```
sbit Mmc_Chip_Select          at LATB4_bit;  // for writing to output pin always use latch
sbit Mmc_Chip_Select_Direction at TRISB4_bit;


// LCD Conexoes do modulo
sbit LCD_D4 at LATB10_bit;
sbit LCD_D5 at LATB11_bit;
sbit LCD_D6 at LATB12_bit;
sbit LCD_D7 at LATB13_bit;
sbit LCD_RS at LATB14_bit;  // Register Select
sbit LCD_EN at LATB15_bit;  // Enable


sbit LCD_D4_Direction at TRISB10_bit;
sbit LCD_D5_Direction at TRISB11_bit;
sbit LCD_D6_Direction at TRISB12_bit;
sbit LCD_D7_Direction at TRISB13_bit;
sbit LCD_RS_Direction at TRISB14_bit;
sbit LCD_EN_Direction at TRISB15_bit;


void uart_init(void);
void gpsinit(void);
void gps_off(void);
```

```
void uart_off(void);
//void gpslat(void);
//void gpsdata(void);

void uart2_inicio(void);
//void stormlog(void);

typedef struct {
/*    char lat[13];
    char lat1[16];
    char lon[13];
    char lon1[16];
    char date[12];
    char date1[12];
    char hora[12];
    char hora1[12];
    char nomearquivo[13];*/
    unsigned char lat[16];
    unsigned char lat1[16];
    unsigned char lon[16];
    unsigned char lon1[16];
    unsigned char date[16];
    unsigned char date1[16];
    unsigned char hora[16];
    unsigned char hora1[16];
    unsigned char nomearquivo[16];
} gpslog;

typedef struct {
    unsigned char ret[801];
    //char treated[601];
} 83tormscope;

gpslog gprmc(void);
83tormscope stormlogger(void);


void mmcinit(void);
//void Create_Log(char hora[12]);
```

APÊNDICE A3 – ARQUIVO MMC.C

```
#include “main.h”

extern sbit Mmc_Chip_Select           at LATB4_bit;  // for writing to output pin always use latch
extern sbit Mmc_Chip_Select_Direction at TRISB4_bit;

unsigned char o;
unsigned char ed1[4];

void mmcinit(){

 //AD1PCFGL = 0x3F;
/*
 Unlock_IOLOCK();
// RPINR21bits.SS1R = 4;
 RPINR20bits.SDI1R = 5;
```

```
 RPOR3bits.RP7R = 8;  //CLk
 RPOR3bits.RP6R = 7;   //SDO
// RPINR20bits.SCK1R = 7;
 Lock_IOLOCK(); */



 //o=PPS_Mapping(7, _INPUT, _SCK1IN);   // Sets pin 7 to be Input, and maps SCK (Clock Input)
 //o=PPS_Mapping(4, _OUTPUT, _SS1OUT);    // Sets pin 4 to be Input, and maps SS (Chip Select)

 o=PPS_Mapping(7, _OUTPUT, _SCK1OUT);   // Sets pin 7 to be Input, and maps SCK (Clock Output)
 inttostr(o,ed1);
 Lcd_Out(1,1,"Remapeando SCK");
 if (o == 255) {
    Lcd_Out(2,1,ed1);
    Lcd_Out(2,9,"OK !");
 }
 else if (o != 255) {
    Lcd_Out(2,1,ed1);
    Lcd_Out(2,9,"Falha !");
 }
 Delay_ms(1000);


 inttostr(o,ed1);
 Lcd_Out(1,1,"Remapeando SS ");
 if (o == 255) {
    Lcd_Out(2,1,ed1);
    Lcd_Out(2,9,"OK !");
 }
 else if (o != 255) {
    Lcd_Out(2,1,ed1);
    Lcd_Out(2,9,"Falha !");
 }
 Delay_ms(1000);

 o=PPS_Mapping(5, _INPUT, _SDI1);     // Sets pin 5 to be Input, and maps SDI (Data Input)
 inttostr(o,ed1);
 Lcd_Out(1,1,"Remapeando SDI");

 if (o == 255) {
    Lcd_Out(2,1,ed1);
    Lcd_Out(2,9,"OK !");
 }
 else if (o != 255) {
    Lcd_Out(2,1,ed1);
    Lcd_Out(2,9,"Falha !");
 }
 Delay_ms(1000);

 o=PPS_Mapping(6, _OUTPUT, _SDO1);    // Sets pin 6 to be Output, and maps SDO (Data Output)
 inttostr(o,ed1);
 Lcd_Out(1,1,"Remapeando SDO");

 if (o == 255) {
    Lcd_Out(2,1,ed1);
    Lcd_Out(2,9,"OK !");
 }
 else if (o != 255) {
    Lcd_Out(2,1,ed1);
```

```
    Lcd_Out(2,9,"Falha !");
  }
 Delay_ms(1000);

}
```

APÊNDICE A4 – ARQUIVO UART.C

```
// ***** UART ***** //

#include "main.h"


// Conector azul ordem nos pinos de uart 1 é : pin 13 laranja pin 12 amarelo

unsigned char uart_rd;


void uart_init(void){

//     PPS_Mapping(8, _INPUT, _U1RX);    // Sets pin 8 to be Input, and maps UART1 RX
//     PPS_Mapping(9, _OUTPUT, _U1TX);   // Sets pin 9 to be Output, and maps UART1 TX

   LATA2_bit=1;              // (Liga Teclado e uart)
   TRISB8_bit=1;
   TRISB9_bit=0;
   RPINR18bits.U1RXR=8;       // Assign RX input to RPR8 (pin 17)
   RPOR4bits.RP9R=3;          // Assign TX output to RPR9 (pin 18)
     //RPOR1bits.RP3R=3;       // Tx em RP3
     //RPINR18 = 0x09;        // Make Pin RP9 U1RX
     //RPOR4bits.RP8R = 0x03;  // Make Pin RP8 U1TX

   //UART1_Init(9600);
   UART1_Init_Advanced(9600,              _UART_8BIT_NOPARITY,              _UART_ONE_STOPBIT,
_UART_HI_SPEED);
   Delay_ms(100);    // (Necessário para estabilizar UART)
}

void uart2_inicio(void){

 LATA2_bit=1;
 TRISB3_bit=1;
 TRISB2_bit=0;

 RPINR19bits.U2RXR=3;       //RX para RP3
 RPOR1bits.RP2R=5;          // Assign TX output to RPR2 (pin 18)
 Delay_ms(100);

 //UART2_Init(9600);
 UART2_Init_Advanced(9600,              _UART_8BIT_NOPARITY,              _UART_ONE_STOPBIT,
_UART_HI_SPEED);

 Delay_ms(100);
}
```

```
void gpsinit(){

  LATA2_bit=1;              // (Liga Teclado e uart)
  TRISB8_bit=1;
  TRISB9_bit=0;
  RPINR18bits.U1RXR=8;      // Assign RX input to RPR8 (pin 17)
  RPOR4bits.RP9R=3;         // Assign TX output to RPR9 (pin 18)

  UART1_Init_Advanced(4800,              _UART_8BIT_NOPARITY,              _UART_ONE_STOPBIT,
_UART_HI_SPEED);
  //UART1_Init(4800);

  Delay_ms(100);    // (Necessário para estabilizar UART)

  UART1_Write_Text("$PGRMO,,2");    //desabilita todos os comandos do gps
  //UART1_Write(13);
  UART1_Write(0x0D);
  //UART1_Write(10);
  UART1_Write(0x0A);
  UART1_Write_Text("$PGRMO,GPRMC,1"); //habilita apenas GPRMC
  UART1_Write(0x0D);
  UART1_Write(0x0A);
  //UART1_Write(13);
  //UART1_Write(10);
}

gpslog gprmc(void){
 unsigned                                                                                                    char
output[851],buffer[16];//,lat[13],lat1[16],lon1[16],lon[13],date[12],date1[12],hora[12],hora1[12];
 unsigned int i=0,j=0,z=0;
 gpslog dados;
 U1MODEbits.UARTEN=1;    // liga UART1

 for (i=0 ; i<851 ; i++) { output[i]=0; }
 for (i=0 ; i<16 ; i++) {    //era 12 antes
    dados.lat[i]=0;
    dados.lat1[i]=0;
    dados.lon[i]=0;
    dados.lon1[i]=0;
    dados.date[i]=0;
    dados.date1[i]=0;
    dados.hora[i]=0;
    dados.hora1[i]=0;
    dados.nomearquivo[i]=0;
    buffer[i]=0;
 }

// ****************** Rotina para Tirar coordenadas ******************
 U1STAbits.OERR = 0;
 z=0;
 while(z<851){
      if (UART1_Data_Ready()) {     // Recebeu dados ?
        output[z] = UART1_Read();
        z++;
        //U1STAbits.URXDA = 0;    //limpa buffer se nao me engano olhar
        Delay_us(25);
        U1STAbits.OERR = 0;     //datasheet      comentado 31/07
      }
```

```
}
U1MODEbits.UARTEN=0;
output[850]=0;
i=0;
j=0;
z=0;

 while (output[z] != 0 ) { // nao esquecer z++ no final
     if (output[z] == '$') {
       z++;
       for (i=0 ; i<5 ; i++) {
          buffer[i]=output[z];
          z++;
       }
       buffer[5]=0;                    // Encerra string
       i=0;
       j=0;
       if (strcmp(buffer,"GPRMC") == 0) {  // Entra se reconhecer GPRMC
         i=0;
         do {
            z++;
            if (output[z] == ',') { i++; } // procura pela primeira string
         }while (i != 2);            // de dados de 87tormsc 87tormsc de ','
         i=0;
         j=0;
         do{
           z++;
           dados.lat[j]=output[z];
           if (output[z] == ',') {
             z++;
             dados.lat[j]=output[z];
             j++;
             dados.lat[j]=0;
             i=1;
           }
           j++;
         }while (i != 1);
         i=0;
         j=0;
         z++;
         if (output[z] == ',') {
           do {
              z++;
              dados.lon[j]=output[z];
              if (output[z] == ',') {
                z++;
                dados.lon[j]=output[z];
                j++;
                dados.lon[j]=0;
                i=1;
              }
              j++;
           }while (i != 1);
           i=0;
         }
       }
     }
     z++;
 }
 dados.lat1[0]=dados.lat[0];
```

```
 dados.lat1[1]=dados.lat[1];
 dados.lat1[2]= ‘o’;
 dados.lat1[3]=dados.lat[2];
 dados.lat1[4]=dados.lat[3];
 dados.lat1[5]=’\’’;
 dados.lat1[6]=dados.lat[5];
 dados.lat1[7]=dados.lat[6];
 dados.lat1[8]=dados.lat[7];
 dados.lat1[9]=dados.lat[8];
 dados.lat1[10]=’””;
 dados.lat1[11]=dados.lat[9];
 dados.lat1[12]=’ ‘;
 dados.lat1[13]=0;


 dados.lon1[0]=dados.lon[0]; //editacao de hora pra format hh:mm:ss
 dados.lon1[1]=dados.lon[1];
 dados.lon1[2]=dados.lon[2];
 dados.lon1[3]= ‘o’;
 dados.lon1[4]=dados.lon[3];
 dados.lon1[5]=dados.lon[4];
 dados.lon1[6]=’\’’;
 dados.lon1[7]=dados.lon[6];
 dados.lon1[8]=dados.lon[7];
 dados.lon1[9]=dados.lon[8];
 dados.lon1[10]=dados.lon[9];
 dados.lon1[11]=’””;
 dados.lon1[12]=dados.lon[10];
 dados.lon1[13]=’ ‘;
 dados.lon1[14]=0;

// ****************** Rotina para Tirar Data ******************
  for (i=0 ; i<16 ; i++) {
    //dados.date[i]=0;
    //dados.date1[i]=0;
    buffer[i]=0;
  }
  i=0;
  j=0;
  z=0;

  while (output[z] != 0) {                        // nao esquecer z++ no final
       if (output[z] == ‘$’) {
          z++;
          for (i=0 ; i<5 ; i++) {
             buffer[i]=output[z];
             z++;
          }
          buffer[5]=0;                            // Encerra string
          if (strcmp(buffer,”GPRMC”) == 0) {       // Entra se reconhecer GPRMC
            i=0;
            do {
               z++;
               if (output[z] == ‘,’) { i++; } // procura pela primeira string
            }while (i != 8);                 // de data 88tormsc de ‘,’
            i=0;
            j=0;
            do {
               z++;
               dados.date[j]=output[z];
```

```
                if ( output[z] ==',') {
                  z++;
                  dados.date[j]=output[z];
                  j++;
                  dados.date[j]=0;
                  i=1;
                }
                j++;
              }while (i != 1);
            }
          }
          z++;
      }
      dados.date[6]=0;

      dados.date1[0]=dados.date[0]; //editacao de hora pra format hh:mm:ss
      dados.date1[1]=dados.date[1];
      dados.date1[2]='/';
      dados.date1[3]=dados.date[2];
      dados.date1[4]=dados.date[3];
      dados.date1[5]='/';
      dados.date1[6]='2';
      dados.date1[7]='0';
      dados.date1[8]=dados.date[4];
      dados.date1[9]=dados.date[5];
      dados.date1[10]=' ';
      dados.date1[11]=0;

// ****************** Rotina para Tirar Hora UTC ******************

      for (i=0 ; i<16 ; i++) {
        //dados.hora[i]=0;
        //dados.hora1[i]=0;
        buffer[i]=0;
      }
      i=0;
      j=0;
      z=0;
      while (output[z] != 0) {                    // nao esquecer z++ no final
          if (output[z] == '$') {
            z++;
            for (i=0 ; i<5 ; i++) {
              buffer[i]=output[z];
              z++;
            }
            buffer[5]=0;                          // Encerra string
            if (strcmp(buffer,"GPRMC") == 0) {       // Entra se reconhecer GPRMC
              i=0;
              j=0;
                do{
                  z++;                          // rotina para extrair hora
                  dados.hora[j]=output[z];
                  if (output[z] == ',') {
                    dados.hora[j]=0;
                    i=1;
                  }
                  j++;
                }while(i != 1);
            }
          }
```

```
        z++;
    }
    dados.hora1[0]=dados.hora[0]; //editacao de hora pra format hh:mm:ss
    dados.hora1[1]=dados.hora[1];
    dados.hora1[2]=':';
    dados.hora1[3]=dados.hora[2];
    dados.hora1[4]=dados.hora[3];
    dados.hora1[5]=':';
    dados.hora1[6]=dados.hora[4];
    dados.hora1[7]=dados.hora[5];
    dados.hora1[8]=13;
    dados.hora1[9]=10;
    dados.hora1[10]=0;

    dados.nomearquivo[0]=dados.date[4];
    dados.nomearquivo[1]=dados.date[5];
    dados.nomearquivo[2]=dados.date[2];
    dados.nomearquivo[3]=dados.date[3];
    dados.nomearquivo[4]=dados.date[0];
    dados.nomearquivo[5]=dados.date[1];
    dados.nomearquivo[6]=dados.hora[0];
    dados.nomearquivo[7]=dados.hora[1];
    dados.nomearquivo[8]=dados.hora[2];
    dados.nomearquivo[9]=dados.hora[3];
    dados.nomearquivo[10]='s';
    dados.nomearquivo[11]=0;
    //dados.nomearquivo[10]='.';
    //dados.nomearquivo[11]='e';

    return dados;     // retorna todos os dados de uma vez
}


90tormscope stormlogger (void) { //nao esquecer de setar baud rate pra 9600
  90tormscope storm;
  unsigned char treated[801]; //melhor 800 se der
  unsigned int i=0,z=0,w=0;
  U2MODEbits.UARTEN=1;
  //while(storm.ret[i]!=0) { //(i<801)
  //while(i<801) {
  //     storm.ret[i]=0;
  //     treated[i]=0;
  //     i++;
  //} //ver se nao da lag
  i=0;
  uart_rd=0;
  U2STAbits.OERR = 0;     //comentei no dia 31/07 testar
  while (uart_rd != 2) {
      if (UART2_Data_Ready()) {  //adicionei dia 21/08
        //  if (UART2_Data_Ready() == 1) {         // le enquanto não acha a 90tormsc string de STX (0d02)
        uart_rd = UART2_Read();
        Delay_us(25);
        U2STAbits.OERR = 0;     //comentei no dia 31/07 testar
        //UART2_Read_Text(output,2,255);    //lento
      }
  //  }
  }
  z=0; i=0;
  U2STAbits.OERR = 0;     //comentei no dia 31/07 testar
  do {          //le enquanto não acha o fim da string ETX (0d03)
```

```
    if (UART2_Data_Ready()) {
      storm.ret[z] = UART2_Read();
      if (storm.ret[z] == 3 || z==800) { i=1; }
      z++;
      Delay_us(25);
      U2STAbits.OERR = 0;     //comentei no dia 31/07 testar
    }
  } while (i != 1);

  U2MODEbits.UARTEN=0;

  storm.ret[800]=0;
  treated[800]=0;

  i=0;  z=0;  w=0;

  while ((storm.ret[z] != 3) && (w < 800)){ //mudei de z pra w z<600
      if (storm.ret[z] == '%'){
        z++;
        if (storm.ret[z] == 'S'){
          //Lcd_Chr(2,16,'E');
          //Delay_ms(50);
          treated[w] = storm.ret[z];   //comentado 29/07
          w++;                         //comentado 29/07
          for (i=0; i<=8; i++) {
             //if (i==3 || i==6) {      //comentado 29/07
             //   treated[w] = ',';
             //   w++;
             //}
             z++;
             treated[w] = storm.ret[z];
             w++;
             if (i == 8) {
               //for (j=0; j<=8 ; j++) {
                  treated[w] = 13;   //Carriage return
                  w++;
                  treated[w]=10;     //Line feed
                  w++;
               //}
             }
          }
        }
      }
      z++;
  }
  storm.ret[800]=0;
  treated[800]=0;

 //z=0;
 //w=0;

// while ((treated[w] != 'S') && w<588){
//       w++;
// }

 //for (i=0 ; i<12 ; i++) {
 //    teste[i] = treated[w];
 //    w++;
 //}
 //teste[12]=0;
```

```
// Lcd_Out(2,1,teste);
 //Delay_ms(2000);



 i=0;
 while(i<801) {   // criei no dia 21/08
     storm.ret[i]=0;
     i++;
 }

 strcpy(storm.ret,treated);
 return storm;
}

void uart_off(void){
   U2STAbits.UTXEN = 0;
   U2MODEbits.UARTEN=0;       //Desabilita UART
   RPINR19bits.U2RXR=0;       //RX para RP3
   RPOR1bits.RP2R=0;          // Assign TX output to RPR2 (pin 18)
   LATA2_bit=0;               //desacopla teclado e 92tormscope92 serial
}

void gps_off(void){
   U1STAbits.UTXEN = 0;
   U1MODEbits.UARTEN=0;       //Desabilita UART
   RPINR18bits.U1RXR=0;       //Desassocia RPN8 com U1RX
   RPOR4bits.RP9R=0;          //Desassocia RPN9 com U1TX
   LATA2_bit=0;               //desacopla teclado e 92tormscope92 serial
}
```

APÊNDICE B – WX-500 STORMSCOPE LOGGER

Códigos fontes do software e seus respectivos arquivos serão adicionados a seguir nos tópicos à frente.

- Unit1.cpp – Programa principal responsável pela conexão serial, aquisições, tratamento dos dados do Stormscope e conexão com os navegadores para executar os 93tormsco do Google Earth®.
- Unit2.cpp – Responsável geração dos logs de dados.
- Unit3.cpp – Tela de sobre do programa.
- Google_off.html – Responsável pelo gerenciamento online dos dados enviados pelo arquivo Unit1.cpp fazendo a conexão com o plugin Google Earth®.
- Google_open.html – Responsável pela importação dos dados de logs prévios e plotagem via plugin.
- Google_open2.html – Responsável pela importação dos dados de logs provenientes do *hardware* desenvolvido.

APÊNDICE B1 - ARQUIVO UNIT1.CPP

```
#ifdef __BORLANDC__
#pragma hdrstop       // 93tormsc specific
#include <condefs.h>
#pragma argsused
#endif
USEUNIT("Tserial.cpp");
#include <vcl.h>
#include "Unit2.cpp" // Monitor
#include "Unit3.cpp" //AboutBox
```

```
#include “Unit5.cpp” //Google
#include “geo.h”
#include “geomag.c”
#include “geoPoint.c”
#include “geoEllips.c”

#pragma hdrstop

#include <iomanip.h>

#include <iostream.h>
//#include <stdio.h>
#include <conio.h>
#include <time.h>
#include <ctime>
//#include <windows.h>
#include <cstdlib.h>
//#include <stdlib.h>
#include <string.h>
#include <fstream.h>
#include “Tserial.h”
#include <stddef.h>

//Edmundo
#include <locale>
#include <sstream>
#include <dir.h>
#include “listports.c”
#include <vector>
#include <windows.h>
//#include <stxutif.h>

#include “Unit1.h”

//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource “*.dfm”
Tform1 *Form1;

#define GEO_TEST_LAT -20.254345                    // Latitude Ilha Solteira -20.254345
#define GEO_TEST_LON -47.89083          // Longitude Ilha Solteira  -51.202971
#define GEO_TEST_HGT   377.0                    // Altura Ilha Solteira
#define GEO_TEST_DATUM GEO_DATUM_DEFAULT          //!< WGS84 Datum

//---------------------------------------------------------------------------
// Variaveis Globais
Tserial *com;
char letter,letra,tempo_off=0,tempo=0,out[100];
long timermin=60000;
char bufferstring[30],bufferstring1[30];
char buffer[80] = “SisNavega”;
wchar_t buffergeo[13];
vector<string> portas;
char porta = ‘COM5’;
int cntporta = 1000;

time_t rawtime; //94tormscop de tempo
struct tm *timeinfo;
int baud=0,i=0,j=0,cxy=0,contador=0,contador_limpa=0,linecounter=0;//cbuff15=0,cbuff30=0,cbuff45=0;
double firstpass=0; //primeira passada para abrir logs
```

```
#define arrlength(a) ( sizeof ( a ) / sizeof ( *a ) )

static BOOL CALLBACK callback(LPVOID lpCallbackValue,LISTPORTS_PORTINFO* lpPortInfo)
{
  string Items = lpPortInfo->lpPortName;
  portas.push_back(Items);
  //ShowMessage(Items.c_str());
  return TRUE;
}

void sleep(unsigned int mseconds)
{
        clock_t goal = mseconds + clock();
        while (goal > clock());
}

__fastcall Tform1::Tform1(Tcomponent* Owner)
                : Tform(Owner)
{
// Initialize the dialog filters to open/save *.txt files
// and also files with arbitrary extension.
OpenDialog1->Filter = "Any file (*.*)|*.*";
// Call ListPort function
ListPorts(callback,NULL);
sleep(2000);
for (j=0 ; j < portas.size( ) ; j++) {
                int kk = 0;
                com = new Tserial();
                sleep(10);
                char *cpy = new char[portas[j].size()+1];
                strcpy(cpy, portas[j].c_str());
                com->connect(cpy, 9600, spNONE);
                for (i=0 ; i < 10 ; i++) {
                        char leitura = com->getChar();
                        if (leitura == '%') {
                          kk = 1;
                        }
                        if (kk == 1 && leitura == 'I') {
                          cntporta = j;
                        }
                        sleep(10);
                }
                com->disconnect();
                sleep(10);
}
}

void Automode(void) {

        double xyz[3],rae[3],efg[3],lat,lon,hgt;
        wchar_t stri[4],azi[4],celula[4];

        GEO_LOCATION loc;
        geoInitLocation(&loc, GEO_TEST_LAT,
                                                GEO_TEST_LON,
                                                GEO_TEST_HGT,
                                                GEO_TEST_DATUM,
                                                "Pointing Site");
```

```
        if (com == 0) {
          Form1->Memo1->Cursor=crDefault;
          com = new Tserial();
          sleep(10);
        }
          if (com != 0) {
                Form1->Memo1->Cursor=crAppStart; // Muda Cursor para ocupado
          }

        time ( &rawtime );
        timeinfo = localtime ( &rawtime );

        //extern char buffer[80],bufferstring[30];
        for (i=0 ; i <= 80 ; i++) {
                buffer[i]=0;
        }
        for (i=0 ; i <= 30 ; i++) {
                bufferstring[i]=0;
        }

        ofstream log,latlog;
        if (Form1->RadioGroup1->ItemIndex==0) {
                strftime(buffer,80,"%Y%m%d%H%M.txt",timeinfo);      //nome      do      arquivo      por
anomesdiahoramin.txt
        }

        log.open(buffer, ofstream::app);
        i = 0;
        j = 0;

                for (j=0; j <= 100 ; j++) {
                        out[j]=0;
                }
                j=0;

                do{ cxy = com->getChar(); }while(cxy !=2);
                if (Form1->RadioGroup1->ItemIndex == 0) {
                  Form1->RadioGroup2->Visible = FALSE;
                  //if (cxy == 2) { //codigo STX
                        while (cxy != 3) {
                                if (cxy == 13) {   //codigo de CR
                                        log << out;


                                        Form1->Memo1->Lines->Add(UnicodeString(out));
                                        //Form1->Memo1->Lines->Add("");
                                        log << "\n";
                                                for (j = 0; j < 100 ; j++) { out[j] = 0; }
                                        j=0;
                                }
                                cxy = com->getChar();
                                out[j] = cxy;
                                j=j+1;
                                if (cxy == 3) {
                                        Form1->Memo1->Lines->Add("");
                                        log << "\n";
                                }
                        }
                  //}
```

```
                }
        log.close();

//*****************************Lat/Lon*****************************

                if (Form1->RadioGroup1->ItemIndex == 1) {
                        latlog.open("latlog.txt", ofstream::app);
                        Form1->RadioGroup2->Visible = TRUE;
                        while (cxy != 3) { // Faça enquanto diferente de End of TX
                                cxy = com->getChar();
                                if (cxy == 'S') {
                                        for (i = 0; i < 3; i++) {
                                                cxy = com->getChar();
                                                stri[i] = cxy;
                                        }
                                        stri[3] = 0;
                                        for (i = 0; i < 3; i++) {
                                                cxy = com->getChar();
                                                azi[i] = cxy;
                                        }
                                        azi[3] = 0;
                                        for (i = 0; i < 3; i++) {
                                                cxy = com->getChar();
                                                celula[i] = cxy;
                                        }
                                        celula[3] = 0;
                                        if (Form1->RadioGroup2->ItemIndex == 0) {               //
lat/lon para Strike
                                                rae[0] = _wtof(stri)*1000;
        //distância em metros
                                                rae[1] = _wtof(azi)*DEG_TO_RAD;
        //inserir dado em graus
                                                rae[2] = 0.0;
                        //não da pra calcular 97tormsco por falta de dados do Stormscope

                                                time_t now = std::time(NULL);                   //Calcula o
Timestamp
                                                tm *ltm = localtime(&now);

                                                geoRae2Efg (&loc, rae, efg);

geoEfg2Llh(GEO_TEST_DATUM,efg,&lat,&lon,&hgt);
                                                lat=lat*RAD_TO_DEG;
                                //convertendo para graus
                                                lon=lon*RAD_TO_DEG;
                                                if (_wtof(stri)*1000 < 200000) {
                                                        latlog << linecounter;
                                                        latlog << " ";
                                                        linecounter++;
                                                        latlog << setprecision(11);
                                                        latlog     << lat;
                                                        latlog << " ";
                                                        latlog << lon;
                                                        latlog << " ";
                                                        latlog << now;
                                                        latlog << " ";
                                                        latlog << ltm->tm_hour << ":";
                                                        latlog << ltm->tm_min << ":";
                                                        latlog << ltm->tm_sec;
                                                        latlog << "\n";
```

```
                                        }
                                    }
                                    else if (Form1->RadioGroup2->ItemIndex == 1) { // lat/lon
para Cell
                                        rae[0] = _wtof(98torms)*1000;
//distância em metros
                                        rae[1] = _wtof(azi)*DEG_TO_RAD;          //inserir
dado em graus
                                        rae[2] = 0.0;
            //não da pra calcular 98tormsco por falta de dados do Stormscope

             time_t now = std::time(NULL);          //Calcula o Timestamp
                                        tm *ltm = localtime(&now);

                                        geoRae2Efg (&loc, rae, efg);

geoEfg2Llh(GEO_TEST_DATUM,efg,&lat,&lon,&hgt);
                                        lat=lat*RAD_TO_DEG;
                        //convertendo para graus
                                        lon=lon*RAD_TO_DEG;
                                        if (_wtof(celula)*1000 < 200000) {
                                        latlog << linecounter;
                                        latlog << " ";
                                        linecounter++;
                                        latlog << setprecision(11);
                                        latlog << lat;
                                        latlog << " ";
                                        latlog << lon;
                                        latlog << now;
                                        latlog << " ";
                                        latlog << ltm->tm_hour << ":";
                                        latlog << ltm->tm_min << ":";
                                        latlog << ltm->tm_sec;
                                        latlog << "\n";
                                        }
                                    }
                                    swprintf(buffergeo,L"%13.6f",lat);
//transforma double em wchar_t
                                    Form1->Memo1->SetSelTextBuf(L"Lat:");
                                    Form1->Memo1->SetSelTextBuf(buffergeo);
                                    swprintf(buffergeo,L"%13.6f",lon);
                                    Form1->Memo1->SetSelTextBuf(L"    Lon:");
                                    Form1->Memo1->SetSelTextBuf(buffergeo);
                                    Form1->Memo1->Lines->Add("");


                                }
                        }
                        latlog.close();
                }
}

void strings_auto(void) {

// ********** Daqui pra baixo parte de tratamento de Strings

  //extern char buffer[80],bufferstring[30]; // importa variaveis declaradas

  Form1->Timer1->Enabled = FALSE;
```

```
ifstream heading,mon0;//,mon1,mon2,mon3;
ofstream string,log0;//,log1,log2,log3;  // Salva arquivo de string corrigido


Form1->Memo1->Lines->Add("");

if (Form1->RadioGroup1->ItemIndex == 0) {   ////////////////////////////////
heading.open(buffer); //abre arquivo strings.txt
99torms(bufferstring, "string_%s",buffer); // Nome do arquivo string_data.txt

string.open(bufferstring, ofstream::app); //Cria ou abre arquivo para gravacao strings

/*if (!heading.is_open()) {
        Application->MessageBox(L"Nao foi possível abrir arquivo! Programa será terminado!\n" ,L"Erro"
,MB_OK | MB_ICONERROR);
        heading.clear(); //reseta o objeto leitura, para limpar memória do sistema
}

if (!string.is_open()) {
        Application->MessageBox(L"Nao foi possível abrir arquivo! Programa será terminado!\n" ,L"Erro"
,MB_OK | MB_ICONERROR);
        string.clear(); //reseta o objeto leitura, para limpar memória do sistema
}  */


// ***** Reconhecimento de %H e %S para heading a partir de um arquivo de log

        for (j=0 ; j <= 100 ; j++) {
                out[j]=0;
        }
        while (heading.get(letra)){
                j=0;
                if (letra == '%') { //99torms decimal para % = 37
                  heading.get(letra);
                  if (letra == 'H') { //Codigo para H=72
                          //Form1->Memo1->Lines->Add("");
                          //Form1->Memo1->SetSelTextBuf("H");
                          //string << "\n";
                          //string << "H";
                          for (i=0; i<=3; i++) {
                                  heading.get(letra);
                                  j=0;
                                  out[j]=letra;
                          //      string << letra;
                                  if (i==3) {
                                          for (j=0; j<=3; j++) {
                                                  if (j == 3) {
                                                    //Form1->Memo1->Lines->Add("");
                          //                                  string << "\n";
                                                  }
                                          }
                                  }
                           }
                  }
        if (letra == 'S') { //Codigo para S=83
          //Form1->Memo1->SetSelTextBuf("S");
          string << "S";
          for (i=0; i<=8; i++) {
             if (i==3 || i==6) {
```

```
                //Form1->Memo1->SetSelTextBuf(“,”);
                string << “,”;
              }
                                        heading.get(letra);
              j=0;
              out[j]=letra;
              //Form1->Memo1->SetSelTextBuf(out);
                                        string << letra;
              if (i==8) {
                for (j=0; j<=8; j++) {
                                                        if (j==8) {
                                                                //Form1->Memo1->Lines->Add(“”);
                    string << “\n”;
                  }
                }
              }
            }
          }
        }
   }

  heading.close();
  string.close();

  if (Form1->RadioGroup1->ItemIndex == 0) {
            mon0.open(bufferstring);
            log0.open(“Log.txt”, ofstream::app);   //colentando informacoes
            while (!mon0.eof()) {                 // atualiza arquivo de Log
                              mon0.get(letra);
                              log0 << letra;
            }
  mon0.close();
  log0.close();
  }
  } ////////////////////////////////

  Form1->Timer1->Enabled = TRUE;
}


//---------------------------------------------------------------------------
void __fastcall Tform1::Button1Click(Tobject *Sender)
{
            //Tserial *com;  //declarei nas 100tormscop globais

            com = new Tserial();
            sleep(10);

            remove(“Log1.txt”);
            remove(“Log2.txt”);
            remove(“Log3.txt”);
            remove(“corlog1.txt”);
            remove(“corlog2.txt”);
            remove(“corlog3.txt”);
            firstpass=0;

            //if (com!=0)
            //{
                        //if      (ComboBox1->ItemIndex == 0) { com->connect(“COM1”, 9600, spNONE); }
                        //else if (ComboBox1->ItemIndex == 1) { com->connect(“COM2”, 9600, spNONE); }
```

```
                    //else if (ComboBox1->ItemIndex == 2) { com->connect("COM3", 9600, spNONE); }
                    //else if (ComboBox1->ItemIndex == 3) { com->connect("COM4", 9600, spNONE); }
                    //else if (ComboBox1->ItemIndex == 4) { com->connect("COM5", 9600, spNONE); }
                    //else if (ComboBox1->ItemIndex == 5) { com->connect("COM6", 9600, spNONE); }
                    if (cntporta!=1000) {
                      char *cpy = new char[portas[cntporta].size()+1];
                      strcpy(cpy, portas[cntporta].c_str());
                      //KARELcom->connect(cpy, 9600, spNONE);
                    }
                    if (com != 0) { Form1->Memo1->Cursor=crAppStart; } // Muda Cursor para ocupado
            //}
            //else if (com == 0){ Form1->Memo1->Cursor=crDefault; }

    Form1->Button1->Enabled = FALSE;
            baud=ComboBox1->ItemIndex;

            Form1->Elat->Visible=False;
            Form1->Llat->Visible=False;
            Form1->Elong->Visible=False;
            Form1->Llong->Visible=False;

            Form1->Timer1->Interval= 2000;
            Form1->Timer2->Interval= 900000;  //timermin = 60000 (tempo em 101tormscope101os)
            Form1->Timer3->Interval= 1800000;
            Form1->Timer4->Interval= 2700000;

            Form1->Timer1->Enabled = TRUE;
    Form1->Timer2->Enabled = TRUE;
    Form1->Timer3->Enabled = TRUE;
            Form1->Timer4->Enabled = TRUE;

            contador=0;

}
//---------------------------------------------------------------------------
void __fastcall Tform1::Button2Click(Tobject *Sender)
{
 if (com != 0) {        // se já estiver desconectado não faz nada
    com->disconnect();
    delete com;
    sleep(10);
    //com=0;
 }
 Form1->Button1->Enabled = TRUE;
 Form1->Timer1->Enabled = FALSE;
 Form1->Timer2->Enabled = FALSE;
 Form1->Timer3->Enabled = FALSE;
 Form1->Timer4->Enabled = FALSE;
 Form1->Elat->Visible=True;
 Form1->Llat->Visible=True;
 Form1->Elong->Visible=True;
 Form1->Llong->Visible=True;
 firstpass=0;
}
//---------------------------------------------------------------------------
void __fastcall Tform1::FormClose(Tobject *Sender, TcloseAction &Action)
{
 if (com != 0) {  // se já estiver desconectado não faz nada
            com->disconnect();
            delete com;
```

```
            //com=0;
            sleep(10);
            Form2->Timer1->Enabled = FALSE;
 }
}
//---------------------------------------------------------------------------
void __fastcall Tform1::Button3Click(Tobject *Sender)
{
  AboutBox->Visible = True;
 //ShowMessage("UNESP – Universidade Estadual Paulista"sLineBreak"Laboratório de Hidrologia e Hidrometria"sLineBreak sLineBreak"Autor: Engº. Edmundo Beinecke"sLineBreak"e-mail: edmundobeinecke@yahoo.com.br");
}
//---------------------------------------------------------------------------
void __fastcall Tform1::Button4Click(Tobject *Sender)
{
  Application->MessageBox(L" 1- Escolha a porta Serial a se comunicar (COM1, COM2, etc)\n 2- Clique em Abrir Porta para começar a logar\n 3- Limpa: Limpa as informações mostradas no campo de log\n 4- Abrir Monitor: Abre modo visual, para acompanhar deslocamento de descargas atmosféricas\n\n Os arquivos de Log e Strings serão gravados no mesmo diretório do programa" ,L"Ajuda" ,MB_OK | MB_ICONASTERISK);
}
//---------------------------------------------------------------------------
void __fastcall Tform1::Timer1Timer(Tobject *Sender)
{
 Automode(); //Auto Mode
 contador=contador+1;

 if (contador >= 9) {
            Form1->Memo1->Clear();
            //if (Form1->RadioGroup1->ItemIndex == 0) {
                        strings_auto();
            //}
            contador=0;
 }
}
//---------------------------------------------------------------------------


void __fastcall Tform1::Button5Click(Tobject *Sender)
{ Form1->Memo1->Clear(); }
//---------------------------------------------------------------------------


void __fastcall Tform1::Button6Click(Tobject *Sender)
{
 Form2->Visible = TRUE;
}
//---------------------------------------------------------------------------


void __fastcall Tform1::Timer2Timer(Tobject *Sender)
{
// TIMER DE 15 MINUTOS

  if (Form1->RadioGroup1->ItemIndex == 1) {  // selecionado para lat/lon
              Form2->Timer1->Enabled = FALSE;
              Form2->Button1->Visible = FALSE;
              //remove("corlog1.txt");
```

```
            CopyFile("latlog.txt","corlog1.txt", FALSE); //copia arquivo false indica para substituir o arquivo
se exister
                        //lat0.open("latlog.txt");
                        //corlog0.open("corlog1.txt", ofstream::app);
                        //while (!lat0.eof()) {
                        //      lat0.get(letra);
                        //      corlog0 << letra;
                        //}
                        //lat0.close();
                        //corlog0.close();
//      sleep(100);
            remove("latlog.txt");
//      sleep(10);
            linecounter = 0;
  }
  if (Form1->RadioGroup1->ItemIndex == 0) {  // selecionado azimuth/distancia
            Form2->Button1->Visible = TRUE;

            CopyFile("Log.txt","Log1.txt" , FALSE);

            remove("Log.txt");
            if (firstpass == 0) { Form2->Visible = TRUE; }
                        firstpass=firstpass+1;
                        Form2->Timer1->Enabled = TRUE;
            }
  //Form1->Timer2->Enabled = FALSE;
  sleep(20);
}
//---------------------------------------------------------------------------

void __fastcall Tform1::Timer3Timer(Tobject *Sender)
{
// TIMER DE 30 MINUTOS

  if (Form1->RadioGroup1->ItemIndex == 1) {  // selecionado para lat/lon
             Form2->Timer1->Enabled = FALSE;
             Form2->Button1->Visible = FALSE;
             CopyFile("corlog1.txt","corlog2.txt",  FALSE);  //copia  arquivo  false  indica  para  substituir  o
arquivo se exister
  //     sleep(100);
  }

  if (Form1->RadioGroup1->ItemIndex == 0) {  // selecionado azimuth/distancia
            Form2->Timer1->Enabled = TRUE;
            Form2->Button1->Visible = TRUE;
            CopyFile("Log1.txt","Log2.txt" , FALSE);
  //   sleep(100);
  }
  //sleep(40); //pequeno delay para start (evitar problemas de sincronismo)
  //Form1->Timer3->Enabled = FALSE;
}
//---------------------------------------------------------------------------

void __fastcall Tform1::Timer4Timer(Tobject *Sender)
{
// TIMER DE 45 MINUTOS

  if (Form1->RadioGroup1->ItemIndex == 1) {  // selecionado para lat/lon
             Form2->Timer1->Enabled = FALSE;
             Form2->Button1->Visible = FALSE;
```

```
            CopyFile("corlog2.txt","corlog3.txt", FALSE); //copia arquivo false indica para substituir o
arquivo se exister
            //sleep(100);
  }

  if (Form1->RadioGroup1->ItemIndex == 0) {  // selecionado azimuth/distancia
            Form2->Timer1->Enabled = TRUE;
            Form2->Button1->Visible = TRUE;
            CopyFile("Log2.txt","Log3.txt" , FALSE);
            //sleep(100);
  }
  //Form1->Timer4->Enabled = FALSE;
  //sleep(60); //pequeno delay para start (evitar problemas de sincronismo)
}
//---------------------------------------------------------------------------


std::string getexepath()
{
  char result[ MAX_PATH ];
  return std::string( result, GetModuleFileName( NULL, result, MAX_PATH ) );
}

bool replace(std::string& str, const std::string& from, const std::string& to) {
    size_t start_pos = str.find(from);
            //if(start_pos == std::string::npos)
                    //return false;
    str.replace(start_pos, from.length(), to);
    return true;
}

void replaceAll(std::string& str, const std::string& from, const std::string& to) {
            size_t start_pos = 0;
    while((start_pos = str.find(from, start_pos)) != std::string::npos) {
        str.replace(start_pos, from.length(), to);
        start_pos += to.length(); // In case 'to' contains 'from', like replacing 'x' with 'yx'
    }
}

void __fastcall Tform1::Button7Click(Tobject *Sender)
{
 string result = getexepath();
 replace(result, "StormScope.", "\google_off.html");
 string result2 = result.substr(0,result.length()-3);
 replaceAll(result2, "\\", "/");
 string Mpath = "C:\\Program Files\\Mozilla Firefox\\firefox.exe file:///*123";
 replace(Mpath, "*", result2);
 string Path = Mpath.substr(0,Mpath.length()-3);
 WinExec(Path.c_str(), SW_SHOWNORMAL);
 //WinExec("C:\\Program                    Files\\Mozilla                    Firefox\\firefox.exe
file:///D:/Edmundo/StormScope2/google_off.html", SW_SHOWNORMAL);
}
//---------------------------------------------------------------------------
std::string ws2s(const std::wstring& s)
{
int len;
int slength = (int)s.length() + 1;
len = WideCharToMultiByte(CP_ACP, 0, s.c_str(), slength, 0, 0, 0, 0);
char* buf = new char[len];
WideCharToMultiByte(CP_ACP, 0, s.c_str(), slength, buf, len, 0, 0);
```

```
std::string r(buf);
delete[] buf;
return r;
}

void __fastcall Tform1::Button8Click(Tobject *Sender)
{
  wchar_t szFileName[MAXFILE+4];
  wchar_t extent[MAXEXT+4];
  wchar_t FilePath[MAXPATH+4];
  wchar_t FileDisk[MAXDRIVE+4];
  // Execute an open file dialog.
  If (OpenDialog1->Execute())
            // First, check if the file exists.
            If (FileExists(OpenDialog1->FileName)) {
              // If it exists, load the data into the memo box.
              Memo1->Lines->LoadFromFile(OpenDialog1->FileName);
              _wfnsplit(OpenDialog1->FileName.w_str(), 0, 0, szFileName, extent);
              string result = getexepath();
              replace(result, "StormScope.", "\google_open.html");
              string result2 = result.substr(0,result.length()-3);
              replaceAll(result2, "\\", "/");
              string Mpath = "C:\\Program Files\\Mozilla Firefox\\firefox.exe file:///*?f=";
              replace(Mpath, "*", result2);
              Mpath += ws2s(szFileName);
              Mpath += ws2s(extent);
              WinExec(Mpath.c_str(), SW_SHOWNORMAL);
   } else {
              // Otherwise, throw an exception.
              Throw(Exception("File does not exist."));
   }
}
//---------------------------------------------------------------------------


void __fastcall Tform1::Button9Click(Tobject *Sender)
{
  using namespace std;

  wchar_t szFileName[MAXFILE+4];
  wchar_t extent[MAXEXT+4];
  wchar_t FilePath[MAXPATH+4];
  wchar_t FileDisk[MAXDRIVE+4];
  double xyz[3],rae[3],efg[3],lat,lon,hgt;
  wchar_t stri[4],azi[4],celula[4];

  int random_integer = rand();

  char bufk[21];

  // convert 123 to string [buf]
  itoa(random_integer, bufk, 10);

  ofstream latlog1;
  //latlog1.open(random_integer.str(), ofstream::app);

  //latlog1.open(bufk, ofstream::app);
  latlog1.open(bufk, ofstream::105torm);
  // Execute an open file dialog.
  If (OpenDialog1->Execute())
```

```
// First, check if the file exists.
If (FileExists(OpenDialog1->FileName)) {
  // If it exists, load the data into the memo box.
  Memo1->Lines->LoadFromFile(OpenDialog1->FileName);
  string str = Memo1->Lines->Strings[0].t_str();

  string buf; // Have a buffer string
  stringstream ss(str); // Insert the string into a stream
        vector<string> tokens; // Create vector to hold our words
        while (ss >> buf)
              tokens.push_back(buf);

  string latOR, longOR;
  latOR = tokens[0].c_str();

  /* Locate the substring to replace. */
  latOR.replace(latOR.find("o"), 1, ".");
  latOR.replace(latOR.find("'"), 1, "");
  latOR.replace(latOR.find("'"), 1, "");
  if (latOR.find('S') != -1) {
        latOR.insert(0,"-");
  }
  latOR.replace(latOR.length()-1, 1, "");

  longOR = tokens[1].c_str();
  /* Locate the substring to replace. */
  longOR.replace(longOR.find("o"), 1, ".");
  longOR.replace(longOR.find("'"), 1, "");
  longOR.replace(longOR.find("'"), 1, "");
  if (longOR.find('W')!= -1) {
        longOR.insert(0,"-");
  }
  longOR.replace(longOR.length()-1, 1, "");

  latlog1 << latOR.c_str();
  latlog1 << " ";
  latlog1 << longOR.c_str();
  latlog1 << " ";
  latlog1 << tokens[2].c_str();
  latlog1 << " ";
  latlog1 << tokens[3].c_str();
  latlog1 << "\n";

  GEO_LOCATION loc;
  geoInitLocation(&loc, atof(latOR.c_str()),
                                          atof(longOR.c_str()),
                                          GEO_TEST_HGT,
                                          GEO_TEST_DATUM,
                                          "Pointing Site");


  for (int i=1; i< Memo1->Lines->Count; i++)
        {
                    string str2 = Memo1->Lines->Strings[i].t_str();

                    if (str2.length()== 0) {
                    continue;
                    }
```

```
                                        rae[0] = atof(str2.substr(8,3).c_str())*1000;
            //distância em metros
                                        rae[1] = atof(str2.substr(5,3).c_str())*DEG_TO_RAD;
            //inserir dado em graus
                                        rae[2] = 0.0;

                                        //rae[0] = _wtof(stri)*1000;                                    //distância
em metros
                                        //rae[1] = _wtof(azi)*DEG_TO_RAD;                 //inserir    dado    em
graus
                                        //rae[2] = 0.0;

                                        geoRae2Efg (&loc, rae, efg);
                                        geoEfg2Llh(GEO_TEST_DATUM,efg,&lat,&lon,&hgt);
                                        lat=lat*RAD_TO_DEG;
              //convertendo para graus
                                        lon=lon*RAD_TO_DEG;
                                        latlog1 << setprecision(11);
                                        latlog1   << lat;
                                        latlog1 << " ";
                                        latlog1 << lon;
                                        latlog1 << "\n";
                                        //latlog1 << Memo1->Lines->Strings[i];
                        }


              latlog1 << "\n";
              latlog1.close();
              _wfnsplit(OpenDialog1->FileName.w_str(), 0, 0, szFileName, extent);
              string result = getexepath();
              replace(result, "StormScope.", "\google_open2.html");
              string result2 = result.substr(0,result.length()-3);
              replaceAll(result2, "\\", "/");
              string Mpath = "C:\\Program Files\\Mozilla Firefox\\firefox.exe file:///*?f=";
              replace(Mpath, "*", result2);
              Mpath += bufk;

              ShowMessage(bufk);
              //Mpath += ws2s(szFileName);
              //Mpath += ws2s(extent);
              WinExec(Mpath.c_str(), SW_SHOWNORMAL);
   } else {
              // Otherwise, throw an exception.
              Throw(Exception("File does not exist."));
   }
}
//---------------------------------------------------------------------------
```

APÊNDICE B2 - ARQUIVO UNIT2.CPP

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
//#include "Unit1.cpp"
//#include "Unit3.cpp"
#include "Unit2.h"
#include <math.h>
```

```
#include <string.h>
#include <time.h>
#include <fstream.h>
#define pi 3.14159265
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"

using namespace std;
Tform2 *Form2;

// **** Variaveis Globais ****

  double cx=0,cy=0;
  extern double firstpass;
  unsigned int raio=0,ang=0,corr=0;
  char
cell1[10000][3],cell[3],angulo1[10000][3],angulo[3],strike1[10000][3],strike[3],head1[10000][4],head[4];

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

double polar_x(int raio,108tormsc,int corr){
        double x,rad,108tormsco,corrad;

        108tormsco = corr/10;    // angulo de 0 a 3599
        corrad   = (pi*108tormsco)/180;
        rad      = (pi*ang)/180;
        x        = raio*cos(rad-(pi/2)-1/2*corrad);
        //x = raio*cos(rad-(pi/2));
        return(x);
}

double polar_y(int raio,108tormsc,int corr){
        double y,rad,correcao,corrad;

        108tormsco = corr/10;    // angulo de 0 a 3599
        corrad   = (pi*108tormsco)/180;
        rad      = (pi*ang)/180;
        y        = raio*sin(rad-(pi/2)-1/2*corrad);
        //y = raio*sin(rad-(pi/2));
        return(y);
}



__fastcall Tform2::Tform2(Tcomponent* Owner)
                : Tform(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall Tform2::FormPaint(Tobject *Sender)
{
  HDC hDC = this->Canvas->Handle;
  MoveWindowOrg(hDC, ClientWidth/2, ClientHeight/2); //muda ponto 0,0 para o centro da tela

  Form2->Canvas->Pen->Color = clBlack;
  Form2->Canvas->Brush->Color = clWhite;
  Form2->Canvas->Ellipse(-200, -200, 200, 200);
  this->Canvas->TextOut(165, -12, "200km");
```

```
  Form2->Canvas->Pen->Style = psDot;
  Form2->Canvas->Ellipse(-150, -150, 150, 150);
  this->Canvas->TextOut(130, -12, “150”);
  Form2->Canvas->Ellipse(-100, -100, 100, 100);
  Form2->Canvas->Ellipse(-50, -50, 50, 50);
  this->Canvas->TextOut(80, -12, “100”);
  this->Canvas->TextOut(35, -12, “50”);
  Form2->Canvas->MoveTo(0, 200);
  Form2->Canvas->LineTo(0, -200);
  Form2->Canvas->MoveTo(-200, 0);
  Form2->Canvas->LineTo(200, 0);
  Form2->Canvas->MoveTo(0, 0);
  Form2->Canvas->Pen->Style = psSolid;

  this->Canvas->Rectangle(173, -150, +298, -240);
  this->Canvas->TextOut(183, -235, “Legenda de Intervalos”);
  Form2->Canvas->Pen->Color = clRed;
  Form2->Canvas->MoveTo(183, -205);
  Form2->Canvas->LineTo(205, -205);
  Form2->Canvas->MoveTo(183, -204);
  Form2->Canvas->LineTo(205, -204);
  this->Canvas->TextOut(215, -212, “15 minutos”);
  Form2->Canvas->Pen->Color = clBlue;
  Form2->Canvas->MoveTo(183, -185);
  Form2->Canvas->LineTo(205, -185);
  Form2->Canvas->MoveTo(183, -184);
  Form2->Canvas->LineTo(205, -184);
  this->Canvas->TextOut(215, -192, “30 minutos”);
  Form2->Canvas->Pen->Color = clGreen;
  Form2->Canvas->MoveTo(183, -165);
  Form2->Canvas->LineTo(205, -165);
  Form2->Canvas->MoveTo(183, -164);
  Form2->Canvas->LineTo(205, -164);
  this->Canvas->TextOut(215, -172, “45 minutos”);
  Form2->Canvas->Pen->Color = clBlack;


}
//---------------------------------------------------------------------------



void __fastcall Tform2::Timer1Timer(Tobject *Sender)
{
  char i,let;
  Form2->Timer1->Enabled = FALSE;

  HDC ed = this->Canvas->Handle;
  MoveWindowOrg(ed, ClientWidth/2, ClientHeight/2); //muda ponto 0,0 para o centro da tela */

  Form2->Canvas->Pen->Color = clBlack;
  Form2->Canvas->Brush->Color = clWhite;
  Form2->Canvas->Ellipse(-200, -200, 200, 200);
  this->Canvas->TextOut(165, -12, “200km”);
  Form2->Canvas->Pen->Style = psDot;
  Form2->Canvas->Ellipse(-150, -150, 150, 150);
  this->Canvas->TextOut(130, -12, “150”);
  Form2->Canvas->Ellipse(-100, -100, 100, 100);
  Form2->Canvas->Ellipse(-50, -50, 50, 50);
  this->Canvas->TextOut(80, -12, “100”);
```

```
 this->Canvas->TextOut(35, -12, “50”);
 Form2->Canvas->MoveTo(0, 200);
 Form2->Canvas->LineTo(0, -200);
 Form2->Canvas->MoveTo(-200, 0);
 Form2->Canvas->LineTo(200, 0);
 Form2->Canvas->MoveTo(0, 0);
 Form2->Canvas->Pen->Style = psSolid;
 Form2->Canvas->Pen->Color = clBlack;

 this->Canvas->Rectangle(173, -150, +298, -240);
 this->Canvas->TextOut(183, -235, “Legenda de Intervalos”);
 Form2->Canvas->Pen->Color = clRed;
 Form2->Canvas->MoveTo(183, -205);
 Form2->Canvas->LineTo(205, -205);
 Form2->Canvas->MoveTo(183, -204);
 Form2->Canvas->LineTo(205, -204);
 this->Canvas->TextOut(215, -212, “15 minutos”);
 Form2->Canvas->Pen->Color = clBlue;
 Form2->Canvas->MoveTo(183, -185);
 Form2->Canvas->LineTo(205, -185);
 Form2->Canvas->MoveTo(183, -184);
 Form2->Canvas->LineTo(205, -184);
 this->Canvas->TextOut(215, -192, “30 minutos”);
 Form2->Canvas->Pen->Color = clGreen;
 Form2->Canvas->MoveTo(183, -165);
 Form2->Canvas->LineTo(205, -165);
 Form2->Canvas->MoveTo(183, -164);
 Form2->Canvas->LineTo(205, -164);
 this->Canvas->TextOut(215, -172, “45 minutos”);
 Form2->Canvas->Pen->Color = clBlack;

ifstream monitor,monitor1,monitor2;

head[0] = 0; head[1] = 0; head[2] = 0; head[3] = 0;

 if (firstpass != 0 && firstpass != 1) { // se primeira passada diferente de 0 e 1 abre log 45 min
  monitor2.open(“Log3.txt”);
  while (!monitor2.eof()){ // varre o arquivo até o final
                        monitor2.get(let);
                        //if (let == ‘H’) { //Codigo para S=72
                        //      monitor2.getline(head,5);
                        //}
                        if (let == ‘S’) { //Codigo para S=83
                          for (i=0 ; i<3 ; i++) {
                                        monitor2.get(let);
                                        cell[i]=let; //grava cell
                          }
                          monitor2.get(let);
                          for (i=0 ; i<3 ; i++) {
                                        monitor2.get(let);
                                        angulo[i]=let; //grava angulo
                          }
                          monitor2.get(let);
                          for (i=0 ; i<3 ; i++) {
                                        monitor2.get(let);
                                        strike[i]=let; //grava strike
        }

        //HDC ed = this->Canvas->Handle;
        //MoveWindowOrg(ed, ClientWidth/2, ClientHeight/2); //muda ponto 0,0 para o centro da tela
```

```
        if (Form2->RadioGroup1->ItemIndex == 0) {
          raio=atoi(cell);
          ang=atoi(angulo);
          corr=atoi(head);
          if (raio < 201) { // se raio = 201 modo teste
            cx=polar_x(raio,ang,corr);
            cy=polar_y(raio,ang,corr);
            this->Canvas->Font->Size = 4;
            this->Canvas->Font->Color = clGreen;
            this->Canvas->TextOut(cx, cy, "o");
          }
          this->Canvas->Font->Color = clBlack;
        }
        if (Form2->RadioGroup1->ItemIndex == 1) {
          raio=atoi(strike);
          ang=atoi(angulo);
          corr=atoi(head);
          if (raio < 201) { // se raio = 201 modo teste
            cx=polar_x(raio,ang,corr);
            cy=polar_y(raio,ang,corr);
            this->Canvas->Font->Color = clGreen;
            this->Canvas->TextOut(cx, cy, "x");
          }
          this->Canvas->Font->Color = clBlack;
        }
  }
}
 monitor2.close();
 //sleep(1);
}


 if (firstpass != 0) {
  monitor1.open("Log2.txt");
  while (!monitor1.eof()){ // varre o arquivo até o final
      monitor1.get(let);
                    //if (let == 'H') { //Codigo para S=72
                    //      monitor1.getline(head,5);
      //}
      if (let == 'S') { //Codigo para S=83
        for (i=0 ; i<3 ; i++) {
           monitor1.get(let);
           cell[i]=let; //grava cell
        }
        monitor1.get(let);
        for (i=0 ; i<3 ; i++) {
           monitor1.get(let);
           angulo[i]=let; //grava angulo
        }
        monitor1.get(let);
        for (i=0 ; i<3 ; i++) {
           monitor1.get(let);
           strike[i]=let; //grava strike
        }

        //HDC ed = this->Canvas->Handle;
        //MoveWindowOrg(ed, ClientWidth/2, ClientHeight/2); //muda ponto 0,0 para o centro da tela */

        if (Form2->RadioGroup1->ItemIndex == 0) {
```

```
            raio=atoi(cell);
            ang=atoi(angulo);
            corr=atoi(head);
            if (raio < 201) { // se raio = 201 modo teste
              cx=polar_x(raio,ang,corr);
              cy=polar_y(raio,ang,corr);
              this->Canvas->Font->Size = 4;
              this->Canvas->Font->Color = clBlue;
              this->Canvas->TextOut(cx, cy, “o”);
            }
            this->Canvas->Font->Color = clBlack;
          }
          if (Form2->RadioGroup1->ItemIndex == 1) {
            raio=atoi(strike);
            ang=atoi(angulo);
            corr=atoi(head);
            if (raio < 201) { // se raio = 201 modo teste
              cx=polar_x(raio,ang,corr);
              cy=polar_y(raio,ang,corr);
              this->Canvas->Font->Color = clBlue;
              this->Canvas->TextOut(cx, cy, “x”);
            }
            this->Canvas->Font->Color = clBlack;
          }

        }
      }
      monitor1.close();
     }

    monitor.open(“Log1.txt”);
    if (!monitor.is_open()) {
            Application->MessageBox(L”Nao foi possível abrir arquivo! Programa será terminado!\n” ,L”Erro”
    ,MB_OK | MB_ICONERROR);
            monitor.clear(); //reseta o objeto leitura, para limpar memória do sistema
    }

     while (!monitor.eof()){ // varre o arquivo até o final
        monitor.get(let);
                        //if (let == ‘H’) { //Codigo para S=72
                        //      monitor.getline(head,5);
                        //}
        if (let == ‘S’) { //Codigo para S=83
          for (i=0 ; i<3 ; i++) {
            monitor.get(let);
            cell[i]=let; //grava cell
          }
          monitor.get(let);
          for (i=0 ; i<3 ; i++) {
            monitor.get(let);
            angulo[i]=let; //grava angulo
                          }
                          monitor.get(let);
          for (i=0 ; i<3 ; i++) {
            monitor.get(let);
            strike[i]=let; //grava strike
          }

          //HDC ed = this->Canvas->Handle;
          //MoveWindowOrg(ed, ClientWidth/2, ClientHeight/2); //muda ponto 0,0 para o centro da tela */
```

```
        if (Form2->RadioGroup1->ItemIndex == 0) {
          raio=atoi(cell);
          ang=atoi(angulo);
          corr=atoi(head);
          if (raio < 201) { // se raio = 201 modo teste
            cx=polar_x(raio,ang,corr);
            cy=polar_y(raio,ang,corr);
            this->Canvas->Font->Size = 4;
            this->Canvas->Font->Color = clRed;
            this->Canvas->TextOut(cx, cy, "o");
          }
          this->Canvas->Font->Color = clBlack;
        }
        if (Form2->RadioGroup1->ItemIndex == 1) {
          raio=atoi(strike);
          ang=atoi(angulo);
          corr=atoi(head);
          if (raio < 201) { // se raio = 201 modo teste
            cx=polar_x(raio,ang,corr);
            cy=polar_y(raio,ang,corr);
            this->Canvas->Font->Color = clRed;
            this->Canvas->TextOut(cx, cy, "x");
          }
          this->Canvas->Font->Color = clBlack;
        }

      }
  }
  monitor.close();
}


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

void __fastcall Tform2::Button1Click(Tobject *Sender)
{
  char i,let;
  head[0] = 0; head[1] = 0; head[2] = 0; head[3] = 0;

  //Form2->Timer1->Enabled = FALSE;
  HDC fu = this->Canvas->Handle;
  MoveWindowOrg(fu, ClientWidth/2, ClientHeight/2); //muda ponto 0,0 para o centro da tela */

  Form2->Canvas->Pen->Color = clBlack;
  Form2->Canvas->Brush->Color = clWhite;
  Form2->Canvas->Ellipse(-200, -200, 200, 200);
  this->Canvas->TextOut(165, -12, "200km");
  Form2->Canvas->Pen->Style = psDot;
  Form2->Canvas->Ellipse(-150, -150, 150, 150);
  this->Canvas->TextOut(130, -12, "150");
  Form2->Canvas->Ellipse(-100, -100, 100, 100);
  Form2->Canvas->Ellipse(-50, -50, 50, 50);
  this->Canvas->TextOut(80, -12, "100");
  this->Canvas->TextOut(35, -12, "50");
  Form2->Canvas->MoveTo(0, 200);
  Form2->Canvas->LineTo(0, -200);
  Form2->Canvas->MoveTo(-200, 0);
  Form2->Canvas->LineTo(200, 0);
```

```
Form2->Canvas->MoveTo(0, 0);
Form2->Canvas->Pen->Style = psSolid;
Form2->Canvas->Pen->Color = clBlack;
this->Canvas->Rectangle(173, -150, +298, -240);
this->Canvas->TextOut(183, -235, “Legenda de Intervalos”);
Form2->Canvas->Pen->Color = clRed;
Form2->Canvas->MoveTo(183, -205);
Form2->Canvas->LineTo(205, -205);
Form2->Canvas->MoveTo(183, -204);
Form2->Canvas->LineTo(205, -204);
this->Canvas->TextOut(215, -212, “15 minutos”);
Form2->Canvas->Pen->Color = clBlue;
Form2->Canvas->MoveTo(183, -185);
Form2->Canvas->LineTo(205, -185);
Form2->Canvas->MoveTo(183, -184);
Form2->Canvas->LineTo(205, -184);
this->Canvas->TextOut(215, -192, “30 minutos”);

Form2->Canvas->Pen->Color = clGreen;
Form2->Canvas->MoveTo(183, -165);
Form2->Canvas->LineTo(205, -165);
Form2->Canvas->MoveTo(183, -164);
Form2->Canvas->LineTo(205, -164);
this->Canvas->TextOut(215, -172, “45 minutos”);

Form2->Canvas->Pen->Color = clBlack;

ifstream monitor3,monitor4,monitor5;

if (firstpass != 0 && firstpass != 1) { // se primeira passada diferente de 0 e 1 abre log 45 min

monitor5.open(“Log3.txt”);
while (!monitor5.eof()){ // varre o arquivo até o final
    monitor5.get(let);
                //if (let == ‘H’) { //Codigo para S=72
                //      monitor5.getline(head,5);
                //}
    if (let == ‘S’) { //Codigo para S=83
      for (i=0 ; i<3 ; i++) {
         monitor5.get(let);
         cell[i]=let; //grava cell
      }
      monitor5.get(let);
      for (i=0 ; i<3 ; i++) {
         monitor5.get(let);
         angulo[i]=let; //grava angulo
      }
      monitor5.get(let);
      for (i=0 ; i<3 ; i++) {
         monitor5.get(let);
         strike[i]=let; //grava strike
      }

      //HDC ed = this->Canvas->Handle;
      //MoveWindowOrg(ed, ClientWidth/2, ClientHeight/2); //muda ponto 0,0 para o centro da tela

      if (Form2->RadioGroup1->ItemIndex == 0) {
        raio=atoi(cell);
        ang=atoi(angulo);
        corr=atoi(head);
```

```
            if (raio < 201) { // se raio = 201 modo teste
              cx=polar_x(raio,ang,corr);
              cy=polar_y(raio,ang,corr);
              this->Canvas->Font->Size = 4;
              this->Canvas->Font->Color = clGreen;
              this->Canvas->TextOut(cx, cy, “o”);
            }
            this->Canvas->Font->Color = clBlack;
          }
          if (Form2->RadioGroup1->ItemIndex == 1) {
            raio=atoi(strike);
            ang=atoi(angulo);
            corr=atoi(head);
            if (raio < 201) { // se raio = 201 modo teste
              cx=polar_x(raio,ang,corr);
              cy=polar_y(raio,ang,corr);
              this->Canvas->Font->Color = clGreen;
              this->Canvas->TextOut(cx, cy, “x”);
            }
            this->Canvas->Font->Color = clBlack;
          }
        }
 }
 monitor5.close();
}

 if (firstpass != 0) {
  monitor4.open(“Log2.txt”);
  while (!monitor4.eof()){ // varre o arquivo até o final
                        monitor4.get(let);
                        //if (let == ‘H’) { //Codigo para S=72
                        //      monitor4.getline(head,5);
                        //}
       if (let == ‘S’) { //Codigo para S=83
          for (i=0 ; i<3 ; i++) {
             monitor4.get(let);
             cell[i]=let; //grava cell
          }
          monitor4.get(let);
          for (i=0 ; i<3 ; i++) {
             monitor4.get(let);
             angulo[i]=let; //grava angulo
          }
          monitor4.get(let);
          for (i=0 ; i<3 ; i++) {
             monitor4.get(let);
             strike[i]=let; //grava strike
          }

          //HDC ed = this->Canvas->Handle;
          //MoveWindowOrg(ed, ClientWidth/2, ClientHeight/2); //muda ponto 0,0 para o centro da tela */


          if (Form2->RadioGroup1->ItemIndex == 0) {
            raio=atoi(cell);
            ang=atoi(angulo);
            corr=atoi(head);
            if (raio < 201) { // se raio = 201 modo teste
              cx=polar_x(raio,ang,corr);
              cy=polar_y(raio,ang,corr);
```

```
            this->Canvas->Font->Size = 4;
            this->Canvas->Font->Color = clBlue;
            this->Canvas->TextOut(cx, cy, “o”);
          }
          this->Canvas->Font->Color = clBlack;
        }
        if (Form2->RadioGroup1->ItemIndex == 1) {
          raio=atoi(strike);
          ang=atoi(angulo);
          corr=atoi(head);
          if (raio < 201) { // se raio = 201 modo teste
            cx=polar_x(raio,ang,corr);
            cy=polar_y(raio,ang,corr);
            this->Canvas->Font->Color = clBlue;
            this->Canvas->TextOut(cx, cy, “x”);
          }
          this->Canvas->Font->Color = clBlack;
        }

      }
  }
  monitor4.close();
 }

 monitor3.open(“Log1.txt”);

 if (!monitor3.is_open()) {
   Application->MessageBox(L”Nao foi possível abrir arquivo! Programa será terminado!\n” ,L”Erro” ,MB_OK | MB_ICONERROR);
   monitor3.clear(); //reseta o objeto leitura, para limpar memória do sistema
 }

  while (!monitor3.eof()){ // varre o arquivo até o final
      monitor3.get(let);
                    //if (let == ‘H’) { //Codigo para S=72
                    //      monitor3.getline(head,5);
                    //}
      if (let == ‘S’) { //Codigo para S=83
        for (i=0 ; i<3 ; i++) {
          monitor3.get(let);
          cell[i]=let; //grava cell
        }
        monitor3.get(let);
        for (i=0 ; i<3 ; i++) {
          monitor3.get(let);
          angulo[i]=let; //grava angulo
        }
        monitor3.get(let);
        for (i=0 ; i<3 ; i++) {
          monitor3.get(let);
          strike[i]=let; //grava strike
        }

        //HDC ed = this->Canvas->Handle;
        //MoveWindowOrg(ed, ClientWidth/2, ClientHeight/2); //muda ponto 0,0 para o centro da tela */


        if (Form2->RadioGroup1->ItemIndex == 0) {
          raio=atoi(cell);
          ang=atoi(angulo);
```

```
            corr=atoi(head);
            if (raio < 201) { // se raio = 201 modo teste
              cx=polar_x(raio,ang,corr);
              cy=polar_y(raio,ang,corr);
              this->Canvas->Font->Size = 4;
              this->Canvas->Font->Color = clRed;
              this->Canvas->TextOut(cx, cy, "o");
            }
            this->Canvas->Font->Color = clBlack;
          }
          if (Form2->RadioGroup1->ItemIndex == 1) {
            raio=atoi(strike);
            ang=atoi(angulo);
            corr=atoi(head);
            if (raio < 201) { // se raio = 201 modo teste
              cx=polar_x(raio,ang,corr);
              cy=polar_y(raio,ang,corr);
                                        this->Canvas->Font->Color = clRed;
              this->Canvas->TextOut(cx, cy, "x");
            }
                                  this->Canvas->Font->Color = clBlack;
          }

        }
   }
   monitor3.close();

}
//---------------------------------------------------------------------------
```

APÊNDICE B3- ARQUIVO UNIT3.CPP

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Unit3.h"
//---------------------------------------------------------------------------
#pragma resource "*.dfm"
TaboutBox *AboutBox;
//---------------------------------------------------------------------------
__fastcall TaboutBox::TaboutBox(Tcomponent* Aowner)
        : Tform(Aowner)
{
}
//---------------------------------------------------------------------------

void __fastcall TaboutBox::OKButtonClick(Tobject *Sender)
{
 AboutBox->Close();
}
//---------------------------------------------------------------------------
```

APÊNDICE B4 - ARQUIVO GOOGLE_OFF.HML

```
<!–saved from url=(0022)http://internet.e-mail →
<html>
<head>
 <title>Sisnavega</title>
<META HTTP-EQUIV=”PRAGMA” CONTENT=”NO-CACHE”>
<script
src=”http://www.google.com/jsapi?key=ABQIAAAAiGji94Y3GAXfuMb7sVeqcxS9upoiAcxYqv2Ry8hHG
3ZOdfgTBhSU1f25naQkmuw5wiJxuozWo11OYA”> </script>
<script type=”text/javascript” src=”http://www.kvasst.com/dossier/stormscope/gears_init.js”></script>
<script type=”text/javascript”>

    var ge;
            google.load(“earth”, “1”);
            google.setOnLoadCallback(init);
            var placemark = new Array(30);
            var placemarkk = new Array();
            var Array0 = new Array();
            var pontos_rota = new Array(10);
            var icon = new Array(30);
            var point = new Array(30);
            var style = new Array(30);
            var eclusa = new Array(7);
            var lineStringPlacemark = new Array(10);
            var req;
            var ray = 0;
            var store = 0;
            //Variables para salvar arquivo
            var latitude = -22.0175;
            var longitude = -47.89083;
            var height = 40000;
            var angle = 65.35;
            var rotation = 0;
            var call = 0;

    function init() {
    google.earth.createInstance(‘map3d’, initCB, failureCB); }

    function initCB(instance) {
      ge = instance;
      ge.getWindow().setVisibility(true);
                    ge.getLayerRoot().enableLayerById(ge.LAYER_TERRAIN, true);
                    ge.getLayerRoot().enableLayerById(ge.LAYER_BUILDINGS, true);
                    ge.getLayerRoot().enableLayerById(ge.LAYER_BORDERS, true);
                    //add_eclusa(“Usina Promissão”,”http://www.dec.feis.unesp.br/ondisa/usina.png”,-
21.2996612,-49.783376,”Eclusa de Promissão”,1);
                    //add_eclusa(“Usina Ibitinga”,”http://www.dec.feis.unesp.br/ondisa/usina.png”,-
21.758294, -48.991390,”Eclusa de Ibitinga”,2);
                    //add_eclusa(“Usina Bariri”,”http://www.dec.feis.unesp.br/ondisa/usina.png”,-22.154374,-
48.755125,”Eclusa de Bariri”,3);
                    //add_eclusa(“Usina Barra Bonita”,”http://www.dec.feis.unesp.br/ondisa/usina.png”,-
22.520733,-48.536487,”Eclusa de Barra Bonita”,4);
                    //add_eclusa(“Usina NAV”,”http://www.dec.feis.unesp.br/ondisa/usina.png”,-21.117446,-
50.201479,”Eclusa de NAV”,5);
                    //add_eclusa(“COH AES Bauru”,”http://www.dec.feis.unesp.br/ondisa/coh_aes.png”,-
22.325152,-49.063305,”Centro de Operações AES”,6);
                    //add_eclusa(“COH UNESP”,”http://www.dec.feis.unesp.br/ondisa/coh_aes.png”,-
20.42881,-51.341509,”Centro de Operações UNESP”,7);
```

```
                //vai_para(-21.2996612,-49.783376,40000,65.35,-30.47);
                vai_para(latitude,longitude,height,angle,rotation);
                document.getElementById("latitude").value = "-22.0175";
                document.getElementById("longitude").value = "-47.89083";
                var options = ge.getOptions();
                options.setStatusBarVisibility(true);
                options.setScaleLegendVisibility(true);
                ge.getNavigationControl().setVisibility(ge.VISIBILITY_HIDE);
                createScreenOverlay();
                }

        function reloadData() {
                var now = new Date();
                url = 'latlog.txt';
                try {
                        req = new XMLHttpRequest();
                } catch (e) {
                        try {
                                req = new ActiveXObject("Msxml2.XMLHTTP");
                                } catch (e) {
                                        try {
                                                req = new ActiveXObject("Microsoft.XMLHTTP");
                                                } catch (oc) {
                                                        alert("No AJAX Support");
                                                        return;
                                                }
                                }
                }
                req.onreadystatechange = processReqChange;
                req.open("GET", url, true);
                req.send(null);
  }


        function processReqChange() {
        // If req shows "complete"
                if (req.readyState == 4)
                        {
                                dataDiv = document.getElementById('currentData');
                                // If "OK"
                                if (req.status == 0)
                                        {
                                        initLight();
                                        }
                                        else
                                        {
                                        // Flag error
                                        dataDiv.innerHTML = '<p>There was a problem retrieving data:
' + req.status + '</p>';
                                        }
                        }
        }


        function failureCB(errorCode) {     }

        function createScreenOverlay() {
                var screenOverlay = ge.createScreenOverlay('');
                screenOverlay.setIcon(ge.createIcon(''));
                screenOverlay.getIcon().
```

```
            setHref("http://www.kvasst.com/dossier/stormscope/legenda.png");

            // Set the point inside the overlay that is used as the positioning
            // anchor point.
            screenOverlay.getOverlayXY().setXUnits(ge.UNITS_FRACTION);
            screenOverlay.getOverlayXY().setYUnits(ge.UNITS_FRACTION);
            screenOverlay.getOverlayXY().setX(.945);
            screenOverlay.getOverlayXY().setY(.900);

            // Set screen position in fractions.
            screenOverlay.getScreenXY().setXUnits(ge.UNITS_FRACTION);
            screenOverlay.getScreenXY().setYUnits(ge.UNITS_FRACTION);
            screenOverlay.getScreenXY().setX(.8);  // Random x.
            screenOverlay.getScreenXY().setY(.8);  // Random y.

            // Rotate around object's center point.
            screenOverlay.getRotationXY().setXUnits(ge.UNITS_FRACTION);
            screenOverlay.getRotationXY().setYUnits(ge.UNITS_FRACTION);
            screenOverlay.getRotationXY().setX(0.5);
            screenOverlay.getRotationXY().setY(0.5);

            // Set object's size in pixels.
            screenOverlay.getSize().setXUnits(ge.UNITS_PIXELS);
            screenOverlay.getSize().setYUnits(ge.UNITS_PIXELS);
            screenOverlay.getSize().setX(182);
            screenOverlay.getSize().setY(164);

            // Rotate by a random number of degrees.
            screenOverlay.setRotation(0);

            ge.getFeatures().appendChild(screenOverlay);
        }


        function vai_para(lat,longi,altura,tilt,heading) {
        //ge.getOptions().setFlyToSpeed(ge.SPEED_TELEPORT);
var lookAt = ge.getView().copyAsLookAt(ge.ALTITUDE_RELATIVE_TO_GROUND);
        lookAt.setLatitude(lat);
        lookAt.setLongitude(longi);
        lookAt.setRange(altura);
        lookAt.setTilt(tilt);
        lookAt.setHeading(heading);
ge.getView().setAbstractView(lookAt);
        }

        function movemap() {
        //ge.getOptions().setFlyToSpeed(ge.SPEED_TELEPORT);
var lookAt = ge.getView().copyAsLookAt(ge.ALTITUDE_RELATIVE_TO_GROUND);
        latitude = document.getElementById("latitude").value;
        longitude = document.getElementById("longitude").value;
        lookAt.setLatitude(parseFloat(latitude));
        lookAt.setLongitude(parseFloat(longitude));
        height = 40000;
        angle = 65.35;
        rotation = -30.47;
        lookAt.setRange(height);
        lookAt.setTilt(angle);
        lookAt.setHeading(rotation);
ge.getView().setAbstractView(lookAt);
        }
```

```
    function CriaMuro(v1,v2,raio,cor,i) {
    lineStringPlacemark[i] = ge.createPlacemark('');
    var lineString = ge.createLineString('');
                lineStringPlacemark[i].setGeometry(lineString);
                lineString.setExtrude(true);
lineString.setAltitudeMode(ge.ALTITUDE_RELATIVE_TO_GROUND);
                var steps = 100;
                var pi2 = Math.PI * 2;
                for (var k = 0; k < steps; k++) {
                        var lat = v1 + raio * Math.cos(k / steps * pi2);
                        var lng = v2 + raio * Math.sin(k / steps * pi2);
                        lineString.getCoordinates().pushLatLngAlt(lat, lng, 0);
                        }
                // Create a style and set width and color of line.
 lineStringPlacemark[i].setStyleSelector(ge.createStyle(''));
 var lineStyle = lineStringPlacemark[i].getStyleSelector().getLineStyle();
 lineStyle.setWidth(10);
 lineStyle.getColor().set(cor);  // aabbggrr format
                //var polyStyle = lineStringPlacemark.getStyleSelector().getPolyStyle();
                //polyStyle.getColor().set('ddffffff');  // aabbggrr format
                //polyStyle.setColorMode(ge.COLOR_RANDOM);

                 ge.getFeatures().appendChild(lineStringPlacemark[i]);
 }

                 function CriaRota(lat1,lat2,lat3,lat4,lat5,long1,long2,long3,long4,long5,cor,i) {
                lineStringPlacemark[i] = ge.createPlacemark('');
                var lineString = ge.createLineString('');
                lineStringPlacemark[i].setGeometry(lineString);
                lineString.setExtrude(true);
lineString.setAltitudeMode(ge.ALTITUDE_RELATIVE_TO_GROUND);
                lineString.getCoordinates().pushLatLngAlt(lat1, long1, 0);
                lineString.getCoordinates().pushLatLngAlt(lat2, long2, 0);
                lineString.getCoordinates().pushLatLngAlt(lat3, long3, 0);
                lineString.getCoordinates().pushLatLngAlt(lat4, long4, 0);
                lineString.getCoordinates().pushLatLngAlt(lat5, long5, 0);
                // Create a style and set width and color of line.
 lineStringPlacemark[i].setStyleSelector(ge.createStyle(''));
 var lineStyle = lineStringPlacemark[i].getStyleSelector().getLineStyle();
 lineStyle.setWidth(5);
 lineStyle.getColor().set("ff007FFF");  // aabbggrr format
                ge.getFeatures().appendChild(lineStringPlacemark[i]);

                }

    function add_pontosrota(icone,lat,longi,i) {
    // Create the placemark.
    Pontos_rota[i] = ge.createPlacemark('');
    // Define a custom icon.
    Icon[i] = ge.createIcon('');
    icon[i].setHref(icone);
    style[i] = ge.createStyle(''); //create a new style
    style[i].getIconStyle().setIcon(icon[i]); //apply the icon to the style
    pontos_rota[i].setStyleSelector(style[i]); //apply the style to the placemark
    // Set the placemark's location.
    Point[i] = ge.createPoint('');
    point[i].setLatitude(lat);
    point[i].setLongitude(longi);
    pontos_rota[i].setGeometry(point[i]);
```

```
        // Add the placemark to Earth.
        Ge.getFeatures().appendChild(pontos_rota[i]);
        }


        function add_eclusa(nome_eclusa,icone,lat,longi,descricao,i) {
        // Create the placemark.
        Eclusa[i] = ge.createPlacemark('');
        eclusa[i].setName(nome_eclusa);
        eclusa[i].setDescription(descricao);
// Define a custom icon.
        Icon[i] = ge.createIcon('');
        icon[i].setHref(icone);
        style[i] = ge.createStyle(''); //create a new style
        style[i].getIconStyle().setIcon(icon[i]); //apply the icon to the style
        eclusa[i].setStyleSelector(style[i]); //apply the style to the placemark
        // Set the placemark's location.
        Point[i] = ge.createPoint('');
        point[i].setLatitude(lat);
        point[i].setLongitude(longi);
        eclusa[i].setGeometry(point[i]);
        // Add the placemark to Earth.
        Ge.getFeatures().appendChild(eclusa[i]);
        }

        function initLight() {
        var div = Math.round((new Date().getTime())/1000.0);
        div = div – 3600;
        var lines =  req.responseText.split("\n");
        if (ray == 0) {
                var counter = 0;
                for (var k = 0; k < lines.length – 1; k++) {
                        var split = lines[k].split(" ");
                        var diff = div – parseFloat(split[3]);
                        if (diff >= 3600) {
                        continue;
                        } else {
                        placemarkk[counter] = ge.createPlacemark('');
                        icon = ge.createIcon('');
                                if ( diff >= 1800 && diff < 3600 ) {

        icon.setHref('http://www.kvasst.com/dossier/122tormscope/usina_g.png');
                                } else if ( (diff < 1800) && (diff >= 900) ) {

        icon.setHref('http://www.kvasst.com/dossier/122tormscope/usina_b.png');
                                } else if ( (diff < 900) && (diff >= 121) ) {

        icon.setHref('http://www.kvasst.com/dossier/122tormscope/usina_r.png');
                                } else {

        icon.setHref('http://www.kvasst.com/dossier/122tormscope/usina_y.png');
                                }
                        style = ge.createStyle(''); //create a new style
                        style.getIconStyle().setIcon(icon); //apply the icon to the style
                        placemarkk[counter].setStyleSelector(style);

                        var point = ge.createPoint('');
                        var lat = parseFloat(split[1]);
                        var longit = parseFloat(split[2]);
                        var timestp = parseFloat(split[3]);
```

```
                Array0[counter] = [lat,longit,timestp];
                point.setLongitude(longit);
                point.setLatitude(lat);
                placemarkk[counter].setGeometry(point);
                // Add the placemark to Earth.
                Ge.getFeatures().appendChild(placemarkk[counter]);
                counter = counter + 1;
                }
        }
        ray = Array0.length;
} else {
        var posI = 10000000;
        var posF = -1;
        var diff = 0;
        var lat_f = Array0[ray – 1][0];
        var longit_f = Array0[ray – 1][1];
        var timestp_f = Array0[ray – 1][2];

        //Eliminamos os placemark com time >= 60 min
        for (var j = 0; j < Array0.length; j++) {
                diff = div – Array0[j][2];
                if ( diff >= 3600 ) {
                        ge.getFeatures().removeChild(placemarkk[j]);

                        posI = j;
                }
        }

        //Eliminamos os elementos do array com time >= 60 min
        if ( posI != 10000000 ) {
                Array0.splice(0,posI + 1);
        } else {
                posI = -1;
        }

        //Desplazamos o resto dos placemarks
        for (var k = posI + 1; k < Array0.length;  k++) {
                        ge.getFeatures().removeChild(placemarkk[k]);
                        placemarkk[k – posI – 1] = ge.createPlacemark(‘’);
                        icon = ge.createIcon(‘’);
                        diff = div – Array0[k – posI – 1][2];
                                        if ( diff >= 1800 && diff < 3600 ) {

icon.setHref(‘http://www.kvasst.com/dossier/123tormscope/usina_g.png’);
                                        } else if ( (diff < 1800) && (diff >= 900) ) {

icon.setHref(‘http://www.kvasst.com/dossier/123tormscope/usina_b.png’);
                                        } else if ( (diff < 900) && (diff >= 121) ) {

icon.setHref(‘http://www.kvasst.com/dossier/123tormscope/usina_r.png’);
                                        } else {

icon.setHref(‘http://www.kvasst.com/dossier/123tormscope/usina_y.png’);
                                        }
                        style = ge.createStyle(‘’); //create a new style
                        style.getIconStyle().setIcon(icon); //apply the icon to the style
                        placemarkk[k – posI – 1].setStyleSelector(style);

                        var point = ge.createPoint(‘’);
                        var lat = Array0[k – posI – 1][0];
```

```
                        var longit = Array0[k – posI – 1][1];
                        point.setLongitude(longit);
                        point.setLatitude(lat);
                        placemarkk[k – posI – 1].setGeometry(point);

                        ge.getFeatures().appendChild(placemarkk[k – posI – 1]);

        }

        //Encontramos a posição do ultimo elemento de Array0 na nova data
        for (var k = 0; k < lines.length – 1; k++) {
                var split = lines[k].split(" ");
                if (parseFloat(split[1]) == lat_f) {
                        if (parseFloat(split[2]) == longit_f) {
                                if (parseFloat(split[3]) == timestp_f) {
                                        posF = k;

                                }
                        }
                }
        }

        //Creamos os novos placemarks
        var larr0 = Array0.length;
        for (var k = posF + 1; k < lines.length – 1;  k++) {
                        var split = lines[k].split(" ");
                        var lat = parseFloat(split[1]);
                        var longit = parseFloat(split[2]);
                        var timestp = parseFloat(split[3]);
                        placemarkk[larr0 + k – posF – 1] = ge.createPlacemark('');
                        icon = ge.createIcon('');
                        diff = div – timestp;
                                        if ( diff >= 1800 && diff < 3600 ) {

icon.setHref('http://www.kvasst.com/dossier/124tormscope/usina_g.png');
                                        } else if ( (diff < 1800) && (diff >= 900) ) {

icon.setHref('http://www.kvasst.com/dossier/124tormscope/usina_b.png');
                                        } else if ( (diff < 900) && (diff >= 121) ) {

icon.setHref('http://www.kvasst.com/dossier/124tormscope/usina_r.png');
                                        } else {

icon.setHref('http://www.kvasst.com/dossier/124tormscope/usina_y.png');
                                        }
                        style = ge.createStyle(''); //create a new style
                        style.getIconStyle().setIcon(icon); //apply the icon to the style
                        placemarkk[larr0 + k – posF – 1].setStyleSelector(style);

                        var point = ge.createPoint('');

                        point.setLongitude(longit);
                        point.setLatitude(lat);
                        placemarkk[larr0 + k – posF – 1].setGeometry(point);

                        ge.getFeatures().appendChild(placemarkk[larr0 + k – posF – 1]);
                        Array0[larr0 + k – posF – 1] = [lat,longit,timestp];

        }
        ray = Array0.length;
```

```
}
call = call + 1;
}

function add_barco(nome_barco,125torm,lat,longi,125tormscop,i) {
// Create the placemark.
Placemark[i] = ge.createPlacemark(‘’);
placemark[i].setName(nome_barco);
placemark[i].setDescription(descricao);

// Define a custom icon.
Icon[i] = ge.createIcon(‘’);
icon[i].setHref(icone);
style[i] = ge.createStyle(‘’); //create a new style
style[i].getIconStyle().setIcon(icon[i]); //apply the icon to the style
placemark[i].setStyleSelector(style[i]); //apply the style to the placemark
// Set the placemark’s location.
Point[i] = ge.createPoint(‘’);
point[i].setLatitude(lat);
point[i].setLongitude(longi);
placemark[i].setGeometry(point[i]);
// Add the placemark to Earth.
Ge.getFeatures().appendChild(placemark[i]);
}

function del_barco(quant) {
var i ;
for (i=1;i<= quant ;i++) {
ge.getFeatures().removeChild(placemark[i]);
}
}

function del_raios(quant) {
var i ;
for (i=1;i<= quant ;i++) {
ge.getFeatures().removeChild(lineStringPlacemark[i]);
}
}

function del_rota(quant) {
var i ;
ge.getFeatures().removeChild(lineStringPlacemark[1]);
for (i=1;i<= quant ;i++) {
ge.getFeatures().removeChild(pontos_rota[i]);
}

}

function passeio() {
var href = ‘http://www.dec.feis.unesp.br/ondisa/passeio3.kml’;
google.earth.fetchKml(ge, href, kmlFinishedLoading);
}

function kmlFinishedLoading(object) {
ge.getTourPlayer().setTour(object);
ge.getTourPlayer().play();
}

function Ativa_Controle() {
var options = ge.getOptions();
```

```
        options.setMouseNavigationEnabled(true);

        }

        function Desativa_Controle() {
        var options = ge.getOptions();
        options.setMouseNavigationEnabled(false);
        }

    function PegaCamera() {
        var lookAt = ge.getView().copyAsLookAt(ge.ALTITUDE_RELATIVE_TO_GROUND);
        document.getElementById("id_lat_la").innerHTML = lookAt.getLatitude().toFixed(5);
        document.getElementById("id_lng_la").innerHTML = lookAt.getLongitude().toFixed(5);
        document.getElementById("id_rng_la").innerHTML = lookAt.getRange().toFixed(2);
        document.getElementById("id_alt_la").innerHTML = lookAt.getAltitude().toFixed(2);
        document.getElementById("id_head_la").innerHTML = lookAt.getHeading().toFixed(2);
        document.getElementById("id_tilt_la").innerHTML = lookAt.getTilt().toFixed(2);
        }

        netscape.security.PrivilegeManager.enablePrivilege('UniversalXPConnect');
        Components.utils.import("resource://gre/modules/NetUtil.jsm");
        Components.utils.import("resource://gre/modules/FileUtils.jsm");

        //
        function save(output){
        // Request file write privs
        netscape.security.PrivilegeManager.enablePrivilege("UniversalXPConnect");
        // Open the save file dialog
        const nsIFilePicker = Components.interfaces.nsIFilePicker;
        var fp = Components.classes["@mozilla.org/filepicker;1"].createInstance(nsIFilePicker);
        fp.init(window, "Save File…", nsIFilePicker.modeSave);
        fp.appendFilters(nsIFilePicker.filterAll);
        var rv = fp.show();
        if(rv == nsIFilePicker.returnOK || rv == nsIFilePicker.returnReplace){
                // Open the file and write to it
                var file = fp.file;
                alert (fp.file.path);
                if(file.exists() == false){//create as necessary
                        file.create( Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 420 );
                }
                var outputStream = Components.classes["@mozilla.org/network/file-output-stream;1"]
                        .createInstance( Components.interfaces.nsIFileOutputStream );
                outputStream.init( file, 0x04 | 0x08 | 0x20, 640, 0 );
                // Save data to file
                var result = outputStream.write( output, output.length );
                outputStream.close();
        }
        }

        //Função para concatenar dados e chama função salvar
        function fileop() {
        var data = latitude + " " + longitude + " " + height + "\n";
                                                        //Armazena dados de latitude, longitude e
altura da vista do google
                data = data + angle + " " + rotation + " " + (Math.round((new Date().getTime())/1000.0) –
3600) + "\n";           //Armazena dados de angulo da vista, rotação respeito oi norte e o tempo em
timestamp
                for (var k = 0; k < Array0.length; k++) {
                data = data + Array0[k][0] + " " + Array0[k][1] + " " + Array0[k][2] + "\n";
                                                //Concatena valores dos dados
```

```
                }
                save(data);
        }

        function getFileFromPath(path) {
                        var file =
Components.classes['@mozilla.org/file/local;1'].createInstance(Components.interfaces.nsILocalFile);
                        file.initWithPath(path);
                        return file;
        }

        function filelog() {
        var data = latitude + " " + longitude + " " + height + "\n";
                                                        //Armazena dados de latitude, longitude e
altura da vista do google
                data = data + angle + " " + rotation + " " + (Math.round((new Date().getTime())/1000.0) –
3600) + "\n";           //Armazena dados de angulo da vista, rotação respeito oi norte e o tempo em
timestamp
                for (var k = 0; k < Array0.length; k++) {
                data = data + Array0[k][0] + " " + Array0[k][1] + " " + Array0[k][2] + "\n";
                                                //Concatena valores dos dados
                }
                        var localpath=document.location.pathname;
                        var directory=localpath.substr(1,localpath.lastIndexOf('/'));
                        var Cdir = directory.replace(/\//g, "\\\\");
                        var Cdir = Cdir + "Logs\\";
                        // Request file write privs
                        netscape.security.PrivilegeManager.enablePrivilege("UniversalXPConnect");

                        var myfile=getFileFromPath(Cdir);
                        var currentTime = new Date();
                        var year = currentTime.getFullYear();
                        var month = currentTime.getMonth() + 1;
                        if (month < 10) month = '0' + month;
                        var day = currentTime.getDate();
                        if (day < 10) day = '0' + day;
                        var hour = currentTime.getHours();
                        if (hour < 10) hour = '0' + hour;
                        var minute = currentTime.getMinutes();
                        if (minute < 10) minute = '0' + minute;
                        var name = "" + year + month + day + hour + minute + ".txt";
                        myfile.append(name);
                        myfile.create(Components.interfaces.nsIFile.NORMAL_FILE_TYPE, 420);

                        var outputStream = Components.classes["@mozilla.org/network/file-output-
stream;1"].createInstance(Components.interfaces.nsIFileOutputStream);
                        outputStream.init(myfile, 0x04 | 0x08 | 0x20, 640, 0);
                        // Save data to file
                        var result = outputStream.write( data, data.length );
                        outputStream.close();
        }

interval = setInterval('reloadData()', 13000);
interval2 = setInterval('filelog()', 3600000);
</script>

</head>
<body bgcolor="#b9d1ea" marginwidth="0">
<table height="40" width="100%" cellpadding="0" cellspacing="0" style="margin-bottom:7px;margin-
top:1px;margin-left:10px;"><tr><td width="190" style="border-bottom:1px;border-left:1px;border-
```

```
top:1px;border-right:0;border-style:solid;border-color:#06F;padding-left:10px;”>Lat: <input type=”text” name=”lat” id=”latitude” /></td><td width=”190” style=”border-bottom:1px;border-left:0px;border-top:1px;border-right:0;border-style:solid;border-color:#06F;padding-left:2px;”>Long: <input type=”text” name=”long” id=”longitude” /></td>
<td style=”border-bottom:1px;border-left:0px;border-top:1px;border-right:1px;border-style:solid;border-color:#06F;padding-right:10px;padding-left:10px;” width=”35”><input type=”button” value=”Ir…” onClick=”movemap();” /></td><td><div style=”width:10px;display:block;” /></td><td width=”90” style=”border-bottom:1px;border-left:1px;border-top:1px;border-right:1px;border-style:solid;border-color:#06F;padding-right:10px;padding-left:10px;”>
<input type=”button” value=”Salvar dados” onClick=”fileop();”></td><td><div style=”width:350px;display:block;” /> </td></tr></table>
<div id=”currentData” align=”center”>
</div>
  <div id=”map3d” width=100%></div>
</body>
</html>
```

APÊNDICE B5- ARQUIVO GOOGLE_OPEN.HTML

```
<!–saved from url=(0022)http://internet.e-mail →
<html>
<head>
 <title>Sisnavega</title>
<META HTTP-EQUIV=”PRAGMA” CONTENT=”NO-CACHE”>
<script
src=”http://www.google.com/jsapi?key=ABQIAAAAiGji94Y3GAXfuMb7sVeqcxS9upoiAcxYqv2Ry8hHG3ZOdfgTBhSU1f25naQkmuw5wiJxuozWo11OYA”> </script>
<script type=”text/javascript” src=”http://www.kvasst.com/dossier/stormscope/gears_init.js”></script>
<script type=”text/javascript”>

    var ge;
            google.load(“earth”, “1”);
            google.setOnLoadCallback(init);
            var placemark = new Array(30);
            var placemarkk = new Array();
            var Array0 = new Array();
            var pontos_rota = new Array(10);
            var icon = new Array(30);
            var point = new Array(30);
            var style = new Array(30);
            var eclusa = new Array(7);
            var lineStringPlacemark = new Array(10);
            var req;
            var ray = 0;
            var store = 0;
            //Variables para salvar arquivo
            var latitude = -22.0175;
            var longitude = -47.89083;
            var height = 40000;
            var angle = 65.35;
            var rotation = 0;

    function init() {
    google.earth.createInstance(‘map3d’, initCB, failureCB); }

    function initCB(instance) {
```

```
        ge = instance;
        ge.getWindow().setVisibility(true);
                        ge.getLayerRoot().enableLayerById(ge.LAYER_TERRAIN, true);
                        ge.getLayerRoot().enableLayerById(ge.LAYER_BUILDINGS, true);
                        ge.getLayerRoot().enableLayerById(ge.LAYER_BORDERS, true);
                        //add_eclusa("Usina Promissão","http://www.dec.feis.unesp.br/ondisa/usina.png",-
21.2996612,-49.783376,"Eclusa de Promissão",1);
                        //add_eclusa("Usina Ibitinga","http://www.dec.feis.unesp.br/ondisa/usina.png",-
21.758294, -48.991390,"Eclusa de Ibitinga",2);
                        //add_eclusa("Usina Bariri","http://www.dec.feis.unesp.br/ondisa/usina.png",-22.154374,-
48.755125,"Eclusa de Bariri",3);
                        //add_eclusa("Usina Barra Bonita","http://www.dec.feis.unesp.br/ondisa/usina.png",-
22.520733,-48.536487,"Eclusa de Barra Bonita",4);
                        //add_eclusa("Usina NAV","http://www.dec.feis.unesp.br/ondisa/usina.png",-21.117446,-
50.201479,"Eclusa de NAV",5);
                        //add_eclusa("COH AES Bauru","http://www.dec.feis.unesp.br/ondisa/coh_aes.png",-
22.325152,-49.063305,"Centro de Operações AES",6);
                        //add_eclusa("COH UNESP","http://www.dec.feis.unesp.br/ondisa/coh_aes.png",-
20.42881,-51.341509,"Centro de Operações UNESP",7);
                        //vai_para(-21.2996612,-49.783376,40000,65.35,-30.47);
                        //vai_para(latitude,longitude,height,angle,rotation);
                        var options = ge.getOptions();
                        options.setStatusBarVisibility(true);
                        options.setScaleLegendVisibility(true);
                        ge.getNavigationControl().setVisibility(ge.VISIBILITY_HIDE);
                        createScreenOverlay();
                        }

                function reloadData() {
                        var now = new Date();
                        var first = getUrlVars()["f"];
                        url = first;
                        try {
                                req = new XMLHttpRequest();
                        } catch (e) {
                                try {
                                        req = new ActiveXObject("Msxml2.XMLHTTP");
                                        } catch (e) {
                                                try {
                                                        req = new ActiveXObject("Microsoft.XMLHTTP");
                                                        } catch (oc) {
                                                                alert("No AJAX Support");
                                                                return;
                                                        }
                                        }
                        }
                        req.onreadystatechange = processReqChange;
                        req.open("GET", url, true);
                        req.send(null);
  }

                function getUrlVars() {
                        var vars = {};
                        var parts = window.location.href.replace(/[?&]+([^=&]+)=([^&]*)/gi,
function(m,key,value) {
                                vars[key] = value;
                        });
   return vars;
                }
```

```
function processReqChange() {
// If req shows “complete”
        if (req.readyState == 4)
                {
                        dataDiv = document.getElementById(‘currentData’);
                        // If “OK”
                        if (req.status == 0)
                                {
                                // Set current data text
                                //dataDiv.innerHTML = req.responseText;
                                //google.earth.createInstance(‘map3d’, initLight, failureCB);
                                //removeall();
                                var lines =  req.responseText.split(“\n”);
                                var split = lines[0].split(“ “);
                                latitude = parseFloat(split[0]);
                                longitude = parseFloat(split[1]);
                                height = parseFloat(split[2]);
                                split = lines[1].split(“ “);
                                angle = parseFloat(split[0]);
                                rotation = parseFloat(split[1]);
                                vai_para(latitude,longitude,height,angle,rotation);
                                initLight();
                                clearInterval(interval);
                                // Start new timer (1 min)
                                //interval = setInterval(“MoveToLatest()”, 6000);
                                //imeoutID = setTimeout(‘reloadData()’, 60000);
                                }
                                else
                                {
                                // Flag error
                                dataDiv.innerHTML = ‘<p>There was a problem retrieving data:
‘ + req.status + ‘</p>’;
                                }
                        }
        }


        function createScreenOverlay() {
                var screenOverlay = ge.createScreenOverlay(‘’);
                screenOverlay.setIcon(ge.createIcon(‘’));
                screenOverlay.getIcon().
                setHref(“http://www.kvasst.com/dossier/stormscope/legenda.png”);

                // Set the point inside the overlay that is used as the positioning
                // anchor point.
                screenOverlay.getOverlayXY().setXUnits(ge.UNITS_FRACTION);
                screenOverlay.getOverlayXY().setYUnits(ge.UNITS_FRACTION);
                screenOverlay.getOverlayXY().setX(.945);
                screenOverlay.getOverlayXY().setY(.900);

                // Set screen position in fractions.
                screenOverlay.getScreenXY().setXUnits(ge.UNITS_FRACTION);
                screenOverlay.getScreenXY().setYUnits(ge.UNITS_FRACTION);
                screenOverlay.getScreenXY().setX(.8);  // Random x.
                screenOverlay.getScreenXY().setY(.8);  // Random y.

                // Rotate around object’s center point.
                screenOverlay.getRotationXY().setXUnits(ge.UNITS_FRACTION);
                screenOverlay.getRotationXY().setYUnits(ge.UNITS_FRACTION);
```

```
                screenOverlay.getRotationXY().setX(0.5);
                screenOverlay.getRotationXY().setY(0.5);

                // Set object's size in pixels.
                screenOverlay.getSize().setXUnits(ge.UNITS_PIXELS);
                screenOverlay.getSize().setYUnits(ge.UNITS_PIXELS);
                screenOverlay.getSize().setX(182);
                screenOverlay.getSize().setY(164);

                // Rotate by a random number of degrees.
                screenOverlay.setRotation(0);

                ge.getFeatures().appendChild(screenOverlay);
        }


        function failureCB(errorCode) {     }

        function vai_para(lat,longi,altura,tilt,heading) {
        //ge.getOptions().setFlyToSpeed(ge.SPEED_TELEPORT);
var lookAt = ge.getView().copyAsLookAt(ge.ALTITUDE_RELATIVE_TO_GROUND);
        lookAt.setLatitude(lat);
        lookAt.setLongitude(longi);
        lookAt.setRange(altura);
        lookAt.setTilt(tilt);
        lookAt.setHeading(heading);
ge.getView().setAbstractView(lookAt);
        }

        function movemap() {
        //ge.getOptions().setFlyToSpeed(ge.SPEED_TELEPORT);
var lookAt = ge.getView().copyAsLookAt(ge.ALTITUDE_RELATIVE_TO_GROUND);
        latitude = document.getElementById("latitude").value;
        longitude = document.getElementById("longitude").value;
        lookAt.setLatitude(parseFloat(latitude));
        lookAt.setLongitude(parseFloat(longitude));
        height = 40000;
        angle = 65.35;
        rotation = -30.47;
        lookAt.setRange(height);
        lookAt.setTilt(angle);
        lookAt.setHeading(rotation);
ge.getView().setAbstractView(lookAt);
        }

        function CriaMuro(v1,v2,raio,cor,i) {
        lineStringPlacemark[i] = ge.createPlacemark('');
        var lineString = ge.createLineString('');
                lineStringPlacemark[i].setGeometry(lineString);
                lineString.setExtrude(true);
   lineString.setAltitudeMode(ge.ALTITUDE_RELATIVE_TO_GROUND);
                var steps = 100;
                var pi2 = Math.PI * 2;
                for (var k = 0; k < steps; k++) {
                        var lat = v1 + raio * Math.cos(k / steps * pi2);
                        var lng = v2 + raio * Math.sin(k / steps * pi2);
                        lineString.getCoordinates().pushLatLngAlt(lat, lng, 0);
                        }
                // Create a style and set width and color of line.
    lineStringPlacemark[i].setStyleSelector(ge.createStyle(''));
```

```
    var lineStyle = lineStringPlacemark[i].getStyleSelector().getLineStyle();
    lineStyle.setWidth(10);
    lineStyle.getColor().set(cor);  // aabbggrr format
                    //var polyStyle = lineStringPlacemark.getStyleSelector().getPolyStyle();
                    //polyStyle.getColor().set('ddffffff');  // aabbggrr format
                    //polyStyle.setColorMode(ge.COLOR_RANDOM);

                     ge.getFeatures().appendChild(lineStringPlacemark[i]);
    }

                     function CriaRota(lat1,lat2,lat3,lat4,lat5,long1,long2,long3,long4,long5,cor,i) {
                    lineStringPlacemark[i] = ge.createPlacemark('');
                    var lineString = ge.createLineString('');
                    lineStringPlacemark[i].setGeometry(lineString);
                    lineString.setExtrude(true);
   lineString.setAltitudeMode(ge.ALTITUDE_RELATIVE_TO_GROUND);
                    lineString.getCoordinates().pushLatLngAlt(lat1, long1, 0);
                    lineString.getCoordinates().pushLatLngAlt(lat2, long2, 0);
                    lineString.getCoordinates().pushLatLngAlt(lat3, long3, 0);
                    lineString.getCoordinates().pushLatLngAlt(lat4, long4, 0);
                    lineString.getCoordinates().pushLatLngAlt(lat5, long5, 0);
                    // Create a style and set width and color of line.
    lineStringPlacemark[i].setStyleSelector(ge.createStyle(''));
    var lineStyle = lineStringPlacemark[i].getStyleSelector().getLineStyle();
    lineStyle.setWidth(5);
    lineStyle.getColor().set("ff007FFF");  // aabbggrr format
                    ge.getFeatures().appendChild(lineStringPlacemark[i]);

                    }

        function add_pontosrota(icone,lat,longi,i) {
        // Create the placemark.
        Pontos_rota[i] = ge.createPlacemark('');
        // Define a custom icon.
        Icon[i] = ge.createIcon('');
        icon[i].setHref(icone);
        style[i] = ge.createStyle(''); //create a new style
        style[i].getIconStyle().setIcon(icon[i]); //apply the icon to the style
        pontos_rota[i].setStyleSelector(style[i]); //apply the style to the placemark
        // Set the placemark's location.
        Point[i] = ge.createPoint('');
        point[i].setLatitude(lat);
        point[i].setLongitude(longi);
        pontos_rota[i].setGeometry(point[i]);
        // Add the placemark to Earth.
        Ge.getFeatures().appendChild(pontos_rota[i]);
        }


        function add_eclusa(nome_eclusa,icone,lat,longi,descricao,i) {
        // Create the placemark.
        Eclusa[i] = ge.createPlacemark('');
        eclusa[i].setName(nome_eclusa);
        eclusa[i].setDescription(descricao);
// Define a custom icon.
        Icon[i] = ge.createIcon('');
        icon[i].setHref(icone);
        style[i] = ge.createStyle(''); //create a new style
        style[i].getIconStyle().setIcon(icon[i]); //apply the icon to the style
        eclusa[i].setStyleSelector(style[i]); //apply the style to the placemark
```

```
// Set the placemark's location.
Point[i] = ge.createPoint('');
point[i].setLatitude(lat);
point[i].setLongitude(longi);
eclusa[i].setGeometry(point[i]);
// Add the placemark to Earth.
Ge.getFeatures().appendChild(eclusa[i]);
}

function initLight() {

var lines =  req.responseText.split("\n");
var split = lines[1].split(" ");
var div = parseFloat(split[2]);
        for (var k = 2; k < lines.length – 1; k++) {
                var split = lines[k].split(" ");
                var diff = div – parseFloat(split[2]);
                placemarkk[k-2] = ge.createPlacemark('');
                icon = ge.createIcon('');
                        if ( diff >= 1800 && diff < 3600 ) {

icon.setHref('http://www.kvasst.com/dossier/133tormscope/usina_g.png');
                        } else if ( (diff < 1800) && (diff >= 900) ) {

icon.setHref('http://www.kvasst.com/dossier/133tormscope/usina_b.png');
                        } else if ( (diff < 900) && (diff >= 121) ) {

icon.setHref('http://www.kvasst.com/dossier/133tormscope/usina_r.png');
                        } else {

icon.setHref('http://www.kvasst.com/dossier/133tormscope/usina_y.png');
                        }
                style = ge.createStyle(''); //create a new style
                style.getIconStyle().setIcon(icon); //apply the icon to the style
                placemarkk[k-2].setStyleSelector(style);
                var point = ge.createPoint('');
                var lat = parseFloat(split[0]);
                var longit = parseFloat(split[1]);
                point.setLongitude(longit);
                point.setLatitude(lat);
                placemarkk[k-2].setGeometry(point);
                // Add the placemark to Earth.
                Ge.getFeatures().appendChild(placemarkk[k-2]);

        }
}


function autoRotate() {
alert ("clico");
}

function removeall() {
        //for (var g = 0; g < ray; g++) {
        //              eval("ge.getFeatures().removeChild(placemark" + g +");");
        //}
return true;
}

function add_barco(nome_barco,icone,lat,longi,descricao,i) {
```

```
// Create the placemark.
Placemark[i] = ge.createPlacemark(‘’);
placemark[i].setName(nome_barco);
placemark[i].setDescription(descricao);

// Define a custom icon.
Icon[i] = ge.createIcon(‘’);
icon[i].setHref(icone);
style[i] = ge.createStyle(‘’); //create a new style
style[i].getIconStyle().setIcon(icon[i]); //apply the icon to the style
placemark[i].setStyleSelector(style[i]); //apply the style to the placemark
// Set the placemark’s location.
Point[i] = ge.createPoint(‘’);
point[i].setLatitude(lat);
point[i].setLongitude(longi);
placemark[i].setGeometry(point[i]);
// Add the placemark to Earth.
Ge.getFeatures().appendChild(placemark[i]);
}

function del_barco(quant) {
var i ;
for (i=1;i<= quant ;i++) {
ge.getFeatures().removeChild(placemark[i]);
}
}

function del_raios(quant) {
var i ;
for (i=1;i<= quant ;i++) {
ge.getFeatures().removeChild(lineStringPlacemark[i]);
}
}

function del_rota(quant) {
var i ;
ge.getFeatures().removeChild(lineStringPlacemark[1]);
for (i=1;i<= quant ;i++) {
ge.getFeatures().removeChild(pontos_rota[i]);
}

}

function passeio() {
var href = ‘http://www.dec.feis.unesp.br/ondisa/passeio3.kml’;
google.earth.fetchKml(ge, href, kmlFinishedLoading);
}

function kmlFinishedLoading(object) {
ge.getTourPlayer().setTour(object);
ge.getTourPlayer().play();
}

function Ativa_Controle() {
var options = ge.getOptions();
options.setMouseNavigationEnabled(true);

}

function Desativa_Controle() {
```

```
        var options = ge.getOptions();
        options.setMouseNavigationEnabled(false);
        }

   function PegaCamera() {
        var lookAt = ge.getView().copyAsLookAt(ge.ALTITUDE_RELATIVE_TO_GROUND);
        document.getElementById("id_lat_la").innerHTML = lookAt.getLatitude().toFixed(5);
        document.getElementById("id_lng_la").innerHTML = lookAt.getLongitude().toFixed(5);
        document.getElementById("id_rng_la").innerHTML = lookAt.getRange().toFixed(2);
        document.getElementById("id_alt_la").innerHTML = lookAt.getAltitude().toFixed(2);
        document.getElementById("id_head_la").innerHTML = lookAt.getHeading().toFixed(2);
        document.getElementById("id_tilt_la").innerHTML = lookAt.getTilt().toFixed(2);
        }

var interval = setInterval('reloadData()', 4000);
//interval = setInterval('reloadData()', 30000);
//imeoutID = setTimeout('reloadData()', 60000); <p>Loading Data…</p>

</script>

</head>
<body>
<div id="currentData" align="center">
  </div>
   <div id="map3d" width=100%></div>
</body>
</html>
```

APÊNDICE B6 - ARQUIVO GOOGLE_OPEN2.HTML

```
<!–saved from url=(0022)http://internet.e-mail →
<html>
<head>
 <title>Sisnavega</title>
<META HTTP-EQUIV="PRAGMA" CONTENT="NO-CACHE">
<script
src="http://www.google.com/jsapi?key=ABQIAAAAiGji94Y3GAXfuMb7sVeqcxS9upoiAcxYqv2Ry8hHG
3ZOdfgTBhSU1f25naQkmuw5wiJxuozWo11OYA"> </script>
<script type="text/javascript" src="http://www.kvasst.com/dossier/stormscope/gears_init.js"></script>
<script type="text/javascript">

    var ge;
        google.load("earth", "1");
        google.setOnLoadCallback(init);
        var placemark = new Array(30);
        var placemarkk = new Array();
        var Array0 = new Array();
        var pontos_rota = new Array(10);
        var icon = new Array(30);
        var point = new Array(30);
        var style = new Array(30);
        var eclusa = new Array(7);
        var lineStringPlacemark = new Array(10);
        var req;
        var ray = 0;
        var store = 0;
        //Variables para salvar arquivo
```

```
        var latitude = -22.0175;
        var longitude = -47.89083;
        var height = 40000;
        var angle = 65.35;
        var rotation = 0;

    function init() {
    google.earth.createInstance('map3d', initCB, failureCB); }

    function initCB(instance) {
      ge = instance;
      ge.getWindow().setVisibility(true);
                ge.getLayerRoot().enableLayerById(ge.LAYER_TERRAIN, true);
                ge.getLayerRoot().enableLayerById(ge.LAYER_BUILDINGS, true);
                ge.getLayerRoot().enableLayerById(ge.LAYER_BORDERS, true);
                //add_eclusa("Usina Promissão","http://www.dec.feis.unesp.br/ondisa/usina.png",-
21.2996612,-49.783376,"Eclusa de Promissão",1);
                //add_eclusa("Usina Ibitinga","http://www.dec.feis.unesp.br/ondisa/usina.png",-
21.758294, -48.991390,"Eclusa de Ibitinga",2);
                //add_eclusa("Usina Bariri","http://www.dec.feis.unesp.br/ondisa/usina.png",-22.154374,-
48.755125,"Eclusa de Bariri",3);
                //add_eclusa("Usina Barra Bonita","http://www.dec.feis.unesp.br/ondisa/usina.png",-
22.520733,-48.536487,"Eclusa de Barra Bonita",4);
                //add_eclusa("Usina NAV","http://www.dec.feis.unesp.br/ondisa/usina.png",-21.117446,-
50.201479,"Eclusa de NAV",5);
                //add_eclusa("COH AES Bauru","http://www.dec.feis.unesp.br/ondisa/coh_aes.png",-
22.325152,-49.063305,"Centro de Operações AES",6);
                //add_eclusa("COH UNESP","http://www.dec.feis.unesp.br/ondisa/coh_aes.png",-
20.42881,-51.341509,"Centro de Operações UNESP",7);
                 //vai_para(-21.2996612,-49.783376,40000,65.35,-30.47);
                //vai_para(latitude,longitude,height,angle,rotation);
                var options = ge.getOptions();
                options.setStatusBarVisibility(true);
                options.setScaleLegendVisibility(true);
                ge.getNavigationControl().setVisibility(ge.VISIBILITY_HIDE);
                createScreenOverlay();
                }

        function reloadData() {
                var now = new Date();
                var first = getUrlVars()["f"];
                url = first;
                try {
                        req = new XMLHttpRequest();
                } catch (e) {
                        try {
                                req = new ActiveXObject("Msxml2.XMLHTTP");
                                } catch (e) {
                                        try {
                                                req = new ActiveXObject("Microsoft.XMLHTTP");
                                                } catch (oc) {
                                                        alert("No AJAX Support");
                                                        return;
                                                }
                                }
                }
                req.onreadystatechange = processReqChange;
                req.open("GET", url, true);

                req.send(null);
```

```
  }

        function getUrlVars() {
                var vars = {};
                var parts = window.location.href.replace(/[?&]+([^=&]+)=([^&]*)/gi,
function(m,key,value) {
                        vars[key] = value;
                });
  return vars;
        }


        function processReqChange() {
        // If req shows "complete"
                if (req.readyState == 4)
                        {
                                dataDiv = document.getElementById('currentData');
                                // If "OK"
                                if (req.status == 0)
                                        {
                                        // Set current data text
                                        //dataDiv.innerHTML = req.responseText;
                                        //google.earth.createInstance('map3d', initLight, failureCB);
                                        //removeall();
                                        var lines =  req.responseText.split("\n");
                                        var split = lines[0].split(" ");
                                        latitude = parseFloat(split[0]);
                                        longitude = parseFloat(split[1]);
                                        height = 40000;
                                        angle = 65.35;
                                        rotation = 1341636009
                                        vai_para(latitude,longitude,height,angle,rotation);
                                        clearInterval(interval);
                                        initLight();
                                        // Start new timer (1 min)
                                        //interval = setInterval("MoveToLatest()", 6000);
                                        //imeoutID = setTimeout('reloadData()', 60000);
                                        }
                                        else
                                        {
                                        // Flag error
                                        dataDiv.innerHTML = '<p>There was a problem retrieving data:
' + req.status + '</p>';
                                        }
                        }
        }


        function createScreenOverlay() {
                var screenOverlay = ge.createScreenOverlay('');
                screenOverlay.setIcon(ge.createIcon(''));
                screenOverlay.getIcon().
                setHref("http://www.kvasst.com/dossier/stormscope/legenda.png");

                // Set the point inside the overlay that is used as the positioning
                // anchor point.
                screenOverlay.getOverlayXY().setXUnits(ge.UNITS_FRACTION);
                screenOverlay.getOverlayXY().setYUnits(ge.UNITS_FRACTION);
                screenOverlay.getOverlayXY().setX(.945);
                screenOverlay.getOverlayXY().setY(.900);
```

```
                // Set screen position in fractions.
                screenOverlay.getScreenXY().setXUnits(ge.UNITS_FRACTION);
                screenOverlay.getScreenXY().setYUnits(ge.UNITS_FRACTION);
                screenOverlay.getScreenXY().setX(.8);  // Random x.
                screenOverlay.getScreenXY().setY(.8);  // Random y.

                // Rotate around object's center point.
                screenOverlay.getRotationXY().setXUnits(ge.UNITS_FRACTION);
                screenOverlay.getRotationXY().setYUnits(ge.UNITS_FRACTION);
                screenOverlay.getRotationXY().setX(0.5);
                screenOverlay.getRotationXY().setY(0.5);

                // Set object's size in pixels.
                screenOverlay.getSize().setXUnits(ge.UNITS_PIXELS);
                screenOverlay.getSize().setYUnits(ge.UNITS_PIXELS);
                screenOverlay.getSize().setX(182);
                screenOverlay.getSize().setY(164);

                // Rotate by a random number of degrees.
                screenOverlay.setRotation(0);

                ge.getFeatures().appendChild(screenOverlay);
        }


        function failureCB(errorCode) {     }

        function vai_para(lat,longi,altura,tilt,heading) {
        //ge.getOptions().setFlyToSpeed(ge.SPEED_TELEPORT);
var lookAt = ge.getView().copyAsLookAt(ge.ALTITUDE_RELATIVE_TO_GROUND);
        lookAt.setLatitude(lat);
        lookAt.setLongitude(longi);
        lookAt.setRange(altura);
        lookAt.setTilt(tilt);
        lookAt.setHeading(heading);
ge.getView().setAbstractView(lookAt);
        }

        function movemap() {
        //ge.getOptions().setFlyToSpeed(ge.SPEED_TELEPORT);
var lookAt = ge.getView().copyAsLookAt(ge.ALTITUDE_RELATIVE_TO_GROUND);
        latitude = document.getElementById("latitude").value;
        longitude = document.getElementById("longitude").value;
        lookAt.setLatitude(parseFloat(latitude));
        lookAt.setLongitude(parseFloat(longitude));
        height = 40000;
        angle = 65.35;
        rotation = -30.47;
        lookAt.setRange(height);
        lookAt.setTilt(angle);
        lookAt.setHeading(rotation);
ge.getView().setAbstractView(lookAt);
        }

        function CriaMuro(v1,v2,raio,cor,i) {
        lineStringPlacemark[i] = ge.createPlacemark('');
        var lineString = ge.createLineString('');
                lineStringPlacemark[i].setGeometry(lineString);
                lineString.setExtrude(true);
```

```
lineString.setAltitudeMode(ge.ALTITUDE_RELATIVE_TO_GROUND);
                var steps = 100;
                var pi2 = Math.PI * 2;
                for (var k = 0; k < steps; k++) {
                        var lat = v1 + raio * Math.cos(k / steps * pi2);
                        var lng = v2 + raio * Math.sin(k / steps * pi2);
                        lineString.getCoordinates().pushLatLngAlt(lat, lng, 0);
                        }
                // Create a style and set width and color of line.
 lineStringPlacemark[i].setStyleSelector(ge.createStyle(‘’));
 var lineStyle = lineStringPlacemark[i].getStyleSelector().getLineStyle();
 lineStyle.setWidth(10);
 lineStyle.getColor().set(cor);  // aabbggrr format
                //var polyStyle = lineStringPlacemark.getStyleSelector().getPolyStyle();
                //polyStyle.getColor().set(‘ddffffff’);  // aabbggrr format
                //polyStyle.setColorMode(ge.COLOR_RANDOM);

                 ge.getFeatures().appendChild(lineStringPlacemark[i]);
 }

                 function CriaRota(lat1,lat2,lat3,lat4,lat5,long1,long2,long3,long4,long5,cor,i) {
                lineStringPlacemark[i] = ge.createPlacemark(‘’);
                var lineString = ge.createLineString(‘’);
                lineStringPlacemark[i].setGeometry(lineString);
                lineString.setExtrude(true);
lineString.setAltitudeMode(ge.ALTITUDE_RELATIVE_TO_GROUND);
                lineString.getCoordinates().pushLatLngAlt(lat1, long1, 0);
                lineString.getCoordinates().pushLatLngAlt(lat2, long2, 0);
                lineString.getCoordinates().pushLatLngAlt(lat3, long3, 0);
                lineString.getCoordinates().pushLatLngAlt(lat4, long4, 0);
                lineString.getCoordinates().pushLatLngAlt(lat5, long5, 0);
                // Create a style and set width and color of line.
 lineStringPlacemark[i].setStyleSelector(ge.createStyle(‘’));
 var lineStyle = lineStringPlacemark[i].getStyleSelector().getLineStyle();
 lineStyle.setWidth(5);
 lineStyle.getColor().set(“ff007FFF”);  // aabbggrr format
                ge.getFeatures().appendChild(lineStringPlacemark[i]);

                }

    function add_pontosrota(icone,lat,longi,i) {
    // Create the placemark.
    Pontos_rota[i] = ge.createPlacemark(‘’);
    // Define a custom icon.
    Icon[i] = ge.createIcon(‘’);
    icon[i].setHref(icone);
    style[i] = ge.createStyle(‘’); //create a new style
    style[i].getIconStyle().setIcon(icon[i]); //apply the icon to the style
    pontos_rota[i].setStyleSelector(style[i]); //apply the style to the placemark
    // Set the placemark’s location.
    Point[i] = ge.createPoint(‘’);
    point[i].setLatitude(lat);
    point[i].setLongitude(longi);
    pontos_rota[i].setGeometry(point[i]);
    // Add the placemark to Earth.
    Ge.getFeatures().appendChild(pontos_rota[i]);
    }


    function add_eclusa(nome_eclusa,icone,lat,longi,descricao,i) {
```

```
        // Create the placemark.
        Eclusa[i] = ge.createPlacemark('');
        eclusa[i].setName(nome_eclusa);
        eclusa[i].setDescription(descricao);
// Define a custom icon.
        Icon[i] = ge.createIcon('');
        icon[i].setHref(icone);
        style[i] = ge.createStyle(''); //create a new style
        style[i].getIconStyle().setIcon(icon[i]); //apply the icon to the style
        eclusa[i].setStyleSelector(style[i]); //apply the style to the placemark
        // Set the placemark's location.
        Point[i] = ge.createPoint('');
        point[i].setLatitude(lat);
        point[i].setLongitude(longi);
        eclusa[i].setGeometry(point[i]);
        // Add the placemark to Earth.
        Ge.getFeatures().appendChild(eclusa[i]);
        }

        function initLight() {

        var lines =  req.responseText.split("\n");
        var split = lines[1].split(" ");
        var div = parseFloat(split[2]);
                for (var k = 2; k < lines.length – 1; k++) {
                        var split = lines[k].split(" ");
                        placemarkk[k-2] = ge.createPlacemark('');
                        icon = ge.createIcon('');
                        icon.setHref('http://www.kvasst.com/dossier/140tormscope/usina_y.png');
                        style = ge.createStyle(''); //create a new style
                        style.getIconStyle().setIcon(icon); //apply the icon to the style
                        placemarkk[k-2].setStyleSelector(style);
                        var point = ge.createPoint('');
                        var lat = parseFloat(split[0]);
                        var longit = parseFloat(split[1]);
                        point.setLongitude(longit);
                        point.setLatitude(lat);
                        placemarkk[k-2].setGeometry(point);
                        ge.getFeatures().appendChild(placemarkk[k-2]);

                }
        }


        function autoRotate() {
        alert ("clico");
        }

        function removeall() {
                //for (var g = 0; g < ray; g++) {
                //                eval("ge.getFeatures().removeChild(placemark" + g +");");
                //}
        return true;
        }

        function add_barco(nome_barco,icone,lat,longi,descricao,i) {
        // Create the placemark.
        Placemark[i] = ge.createPlacemark('');
        placemark[i].setName(nome_barco);
        placemark[i].setDescription(descricao);
```

```
        // Define a custom icon.
        Icon[i] = ge.createIcon(‘’);
        icon[i].setHref(icone);
        style[i] = ge.createStyle(‘’); //create a new style
        style[i].getIconStyle().setIcon(icon[i]); //apply the icon to the style
        placemark[i].setStyleSelector(style[i]); //apply the style to the placemark
        // Set the placemark’s location.
        Point[i] = ge.createPoint(‘’);
        point[i].setLatitude(lat);
        point[i].setLongitude(longi);
        placemark[i].setGeometry(point[i]);
        // Add the placemark to Earth.
        Ge.getFeatures().appendChild(placemark[i]);
        }

        function del_barco(quant) {
        var i ;
        for (i=1;i<= quant ;i++) {
        ge.getFeatures().removeChild(placemark[i]);
        }
        }

        function del_raios(quant) {
        var i ;
        for (i=1;i<= quant ;i++) {
        ge.getFeatures().removeChild(lineStringPlacemark[i]);
        }
        }

        function del_rota(quant) {
        var i ;
        ge.getFeatures().removeChild(lineStringPlacemark[1]);
        for (i=1;i<= quant ;i++) {
        ge.getFeatures().removeChild(pontos_rota[i]);
        }

        }

        function passeio() {
        var href = ‘http://www.dec.feis.unesp.br/ondisa/passeio3.kml’;
google.earth.fetchKml(ge, href, kmlFinishedLoading);
        }

        function kmlFinishedLoading(object) {
        ge.getTourPlayer().setTour(object);
        ge.getTourPlayer().play();
        }

        function Ativa_Controle() {
        var options = ge.getOptions();
        options.setMouseNavigationEnabled(true);

        }

        function Desativa_Controle() {
        var options = ge.getOptions();
        options.setMouseNavigationEnabled(false);
        }
```

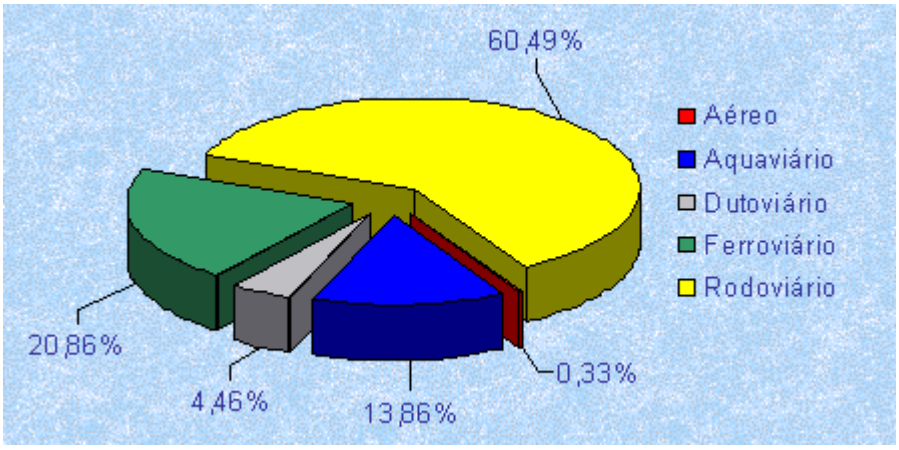
```
   function PegaCamera() {
         var lookAt = ge.getView().copyAsLookAt(ge.ALTITUDE_RELATIVE_TO_GROUND);
         document.getElementById("id_lat_la").innerHTML = lookAt.getLatitude().toFixed(5);
         document.getElementById("id_lng_la").innerHTML = lookAt.getLongitude().toFixed(5);
         document.getElementById("id_rng_la").innerHTML = lookAt.getRange().toFixed(2);
         document.getElementById("id_alt_la").innerHTML = lookAt.getAltitude().toFixed(2);
         document.getElementById("id_head_la").innerHTML = lookAt.getHeading().toFixed(2);
         document.getElementById("id_tilt_la").innerHTML = lookAt.getTilt().toFixed(2);
         }

var interval = setInterval('reloadData()', 4000);
//interval = setInterval('reloadData()', 30000);
//imeoutID = setTimeout('reloadData()', 60000); <p>Loading Data…</p>

</script>

</head>
<body>
<div id="currentData" align="center">
  </div>
   <div id="map3d" width=100%></div>
</body>
</html>
```

APÊNDICE C – SISTEMA MODAL BRASILEIRO

Uma das principais barreiras atualmente no Brasil para o desenvolvimento da logística está relacionada com as enormes deficiências encontradas na infraestrutura de transportes e comunicação. O transporte brasileiro como sabemos, apresenta uma exagerada dependência do modal rodoviário; este tipo de distorções na matriz de transportes causa o estrangulamento das vias devido ao alto fluxo de veículos, grandes custos de manutenção das estradas e rodovias. É fato a má conservação destas vias rodoviárias pelo governo brasileiro, o que causa insegurança para usuários das vias. (RIBEIRO; FERREIRA, 2002)

Segundo a ANTT a matriz de transportes brasileira está representada conforme o gráfico da Figura 37.

Figura 37 – Matriz de transporte brasileiro

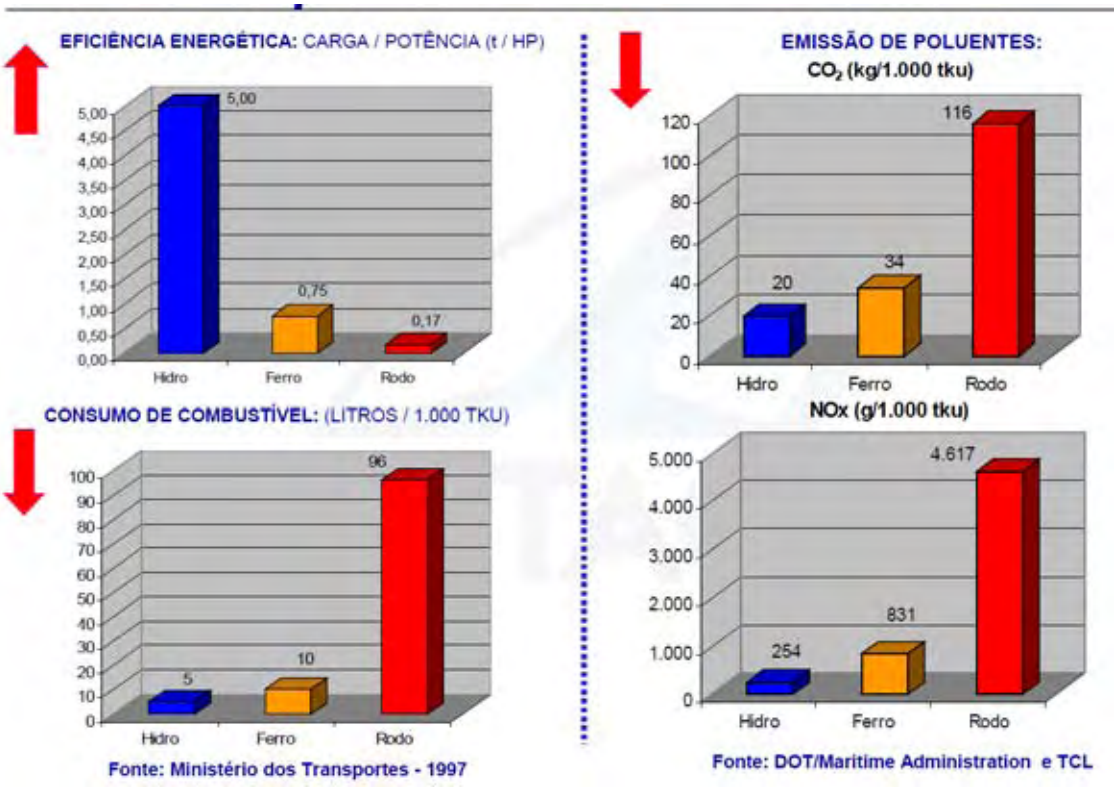


Fonte: Brasil (2012).

Um dos grandes trunfos do país no setor logístico é o modal hidroviário interior, que tem grande potencial de crescimento; o Brasil é naturalmente um país rico em hidrovias naturais, não são necessários grandes investimentos para navegação interior. Segundo a ANTAQ o país tem cerca de 13mil km de vias navegáveis, com possibilidade para expansão de 44 mil km, sendo, portanto um país privilegiado neste sentido.

Outro aspecto importante a ser ressaltado é a eficiência energética do transporte hidroviário, como pode ser observado na Figura 38.

Figura 38 – Aspectos ambientais relevantes



Fonte: Oliva (2008).

Analisando as comparações energéticas e ambientais chegamos à conclusão que o modal hidroviário deve ser mais explorado pelo como meio de transporte no país devido às diversas vantagens econômicas e ambientais apresentadas, além destas ainda podemos citar: (OLIVA, 2008)

- Menor custo operacional;
- Menor congestionamento de tráfego;
- Menor número de acidentes;
- Custo de Infraestrutura;
- Maior capacidade de concentração de cargas;
- Maior vida útil dos equipamentos e veículos;
- Maior segurança da carga e controle fiscal.

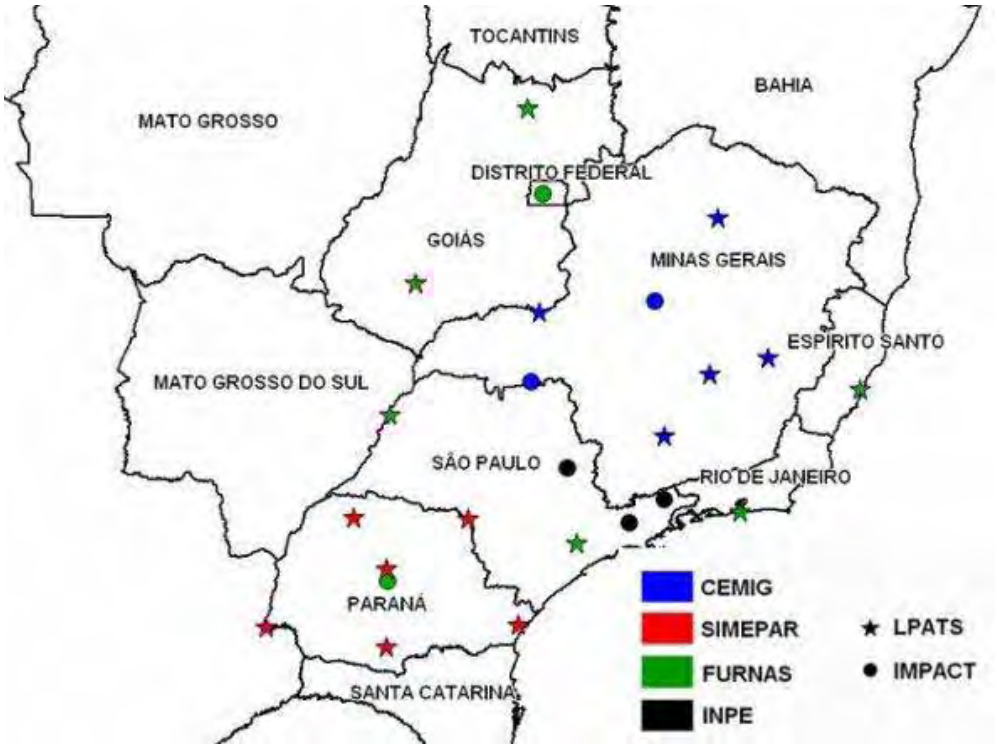
ANEXO A – RINDAT

A Rede Integrada Nacional de Detecção de Descargas Atmosféricas (RINDAT) é uma rede de sensores e centrais que permitem detectar em tempo real as descargas atmosféricas nuvem-solo, isto é, a maior parte das descargas que atingem o solo, em parte do território brasileiro.

A RINDAT foi criada a partir de um convênio de cooperação técnico-científico entre quatro instituições: a CEMIG (Companhia Energética de Minas Gerais), FURNAS (Furnas Centrais Elétricas), o INPE (Instituto Nacional de Pesquisas Espaciais) e o SIMEPAR (Sistema Meteorológico do Paraná).

Até o início de 2005, a RINDAT cobria cerca de um terço do país. Ao longo de 2005, a RINDAT esta passando por um processo de expansão, cuja meta é fazer com que a rede passe a cobrir dois terços do país, incluindo de forma integral as regiões sul, sudeste e centro-oeste. Em área de monitoramento, a RINDAT ocupa a terceira posição no mundo, sendo superada somente pelas redes existentes nos Estados Unidos e Canadá.

Figura 39 – Localização dos sensores pelo Brasil



Fonte: RINDAT (2005)

Principais Objetivos do RINDAT:

- Intercâmbio dos sinais obtidos pelos sensores das diferentes instituições envolvidas, com o objetivo de ampliar a área de cobertura e melhorar a qualificação das informações obtidas.

- Integração dos procedimentos de análise, manutenção e operação conjunta, de modo a otimizar os custos e minimizar as perdas de dados, aumentando a redundância da rede.

- Intercâmbio de informações técnico-científicas. Com base nas tecnologias atualmente disponíveis, a RINDAT apresenta as seguintes características principais: eficiência de detecção entre 70% e 90%, precisão de localização média entre 0,5 km e 2 km, precisão média da estimativa do pico de corrente das descargas de 20% a 50% e capacidade de discriminação entre as descargas nuvem-solo e intranuvem de cerca de 80% a 90%, sendo que estes valores variam regionalmente. (RINDAT, 2005)

ANEXO A1 - SISTEMA DE DETECÇÃO DE DESCARGAS ATMOSFÉRICAS

O Sistema de Detecção e Localização de Descargas Atmosféricas gera pesquisa científica e produtos destinados a aplicações na previsão de tempo, na análise e manutenção de sistemas elétricos de transmissão, de distribuição e na emissão de laudos de análise de eventos severos para seguradoras e empresas de engenharia.

O sistema utiliza as tecnologias denominadas "Sistema de Localização e Rastreio de Raios" ("Lightning Positioning and Tracking System" - LPATS) e "Localização da Direção Magnética" ("Magnetic Direction Finder" - MDF). A precisão das informações de localização de raios do sistema é, em média, de 500 metros dentro do perímetro definido pela posição das estações remotas de recepção. O sistema opera através do Sistema de Posicionamento Global ("Global Positioning System"- GPS), o qual proporciona informações de temporização de raios com resoluções de até 300 nanosegundos.

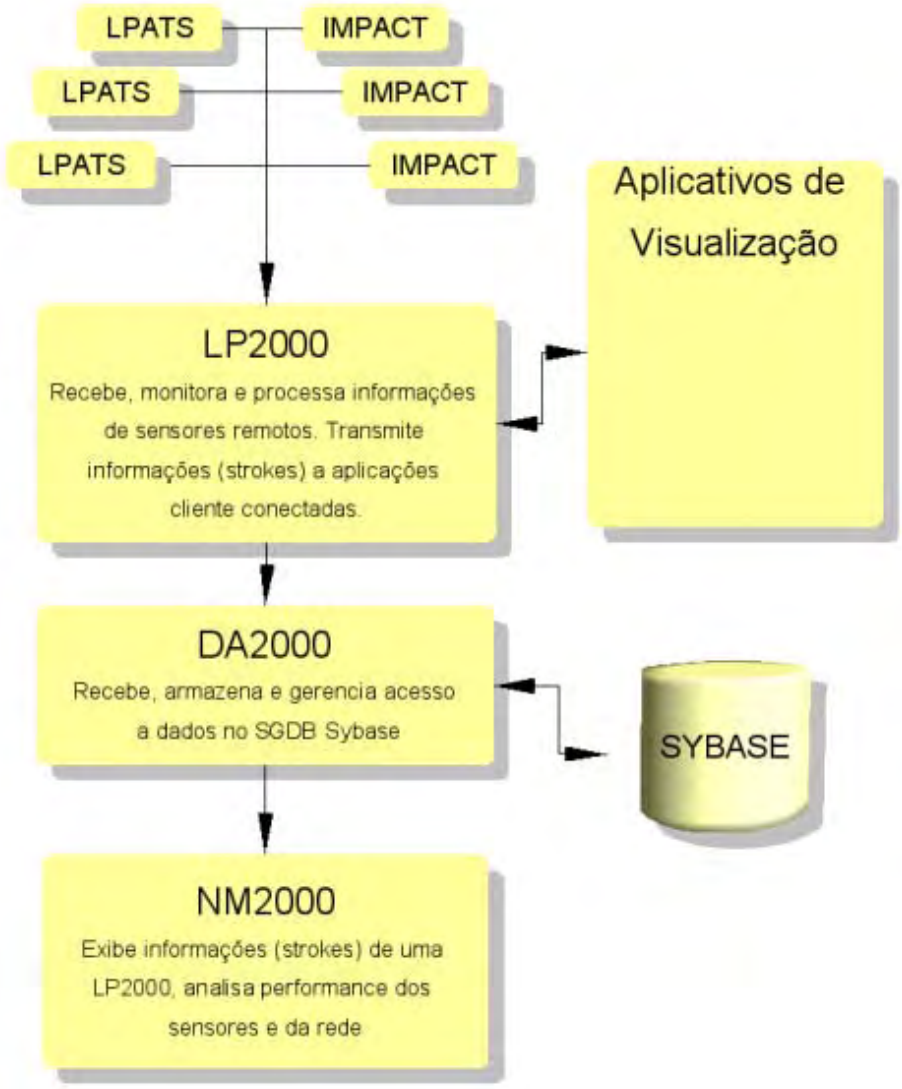
Entre os produtos de visualização gerados pelo sistema se destacam:

- Localização geográfica e temporal de descargas atmosféricas nuvem-terra;
- Localização de temporais
- Determinação de características de descargas como: valor estimado do pico da corrente de retorno, polaridade e número de componentes (multiplicidade) se a descarga for de natureza múltipla.

Após os sinais das descargas serem registrados pelos sensores, eles são enviados as centrais de processamento onde são então processados para obter-se a localização e características das descargas, e disponibilizados para visualização em tempo real ou armazenados para análises históricas. A RINDAT possui 5 centrais em: Belém, Belo Horizonte, Curitiba, Rio de Janeiro e São José dos Campos.

Os sinais dos sensores são transmitidos através de canal de comunicação dedicado para as centrais de processamento, onde são processados e distribuídos para unidades de visualização e armazenamento de dados. (RINDAT, 2005) .

Figura 40 – Funcionamento do sistema RINDAT



Fonte: RINDAT (2005)