Universidade Federal do Rio de Janeiro

Escola Politécnica

Departamento de Eletrônica e de Computação

Rastreamento de posição por satélite com monitoração e transmissão de dados.

Autor:

Claudio Romero

Orientador:

Prof. Gelson Vieira Mendonça

Orientador:

José Otavio Goulart Pecly

Examinador:

Prof. Antônio Cláudio Gómez de Sousa

Examinador:

Prof. Mauros Campello Queiroz

DEL

Abril de 2013

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

Escola Politécnica – Departamento de Eletrônica e de Computação

Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária

Rio de Janeiro – RJ CEP 21949-900

À minha família e amigos.

Agradecimentos

Agradeço à minha família, por todo seu apoio e também por toda a sua contribuição, não somente para minha formação acadêmica mas principalmente para a minha formação com ser humano. Este projeto, assim como outras conquistas no passado e no futuro, é também de vocês.

Agradeço também aos meus amigos, pelo bom humor e pela sua grande amizade que continua expandindo meus horizontes. Espero que eu já tenha feito ou que possa vir a fazer o mesmo por vocês.

Aos membros do Laboratório de Traçadores agradeço pelas experiências e oportunidades desfrutadas, além é claro do companheirismo, todos essenciais para a realização deste trabalho.

Agradeço a todos aqueles que contribuíram, direta ou indiretamente, para minha formação acadêmica, sejam eles professores, que partilharam um pouco do seu conhecimento, ou funcionários que mantém a universidade em movimento.

Finalmente, faço minhas as palavras de um grande amigo, e agradeço ao leitor que leu até os agradecimentos, muito obrigado.

RESUMO

Este trabalho trata da integração de dispositivos eletrônicos para a implementação de um protótipo capaz de rastrear a sua posição (utilizando o Sistema de Posicionamento Global – GPS), delimitar uma área de interesse (*geofence*), realizar a leitura de sensores, estimar o estado de sua fonte de alimentação (bateria e carregador) e transmitir mensagens (SMS) através de uma rede de telefonia celular (GSM).

Inicialmente são apresentados o Sistema de Posicionamento Global (GPS) e conceitos relacionados, focando então nas sentenças, estabelecidas pelo padrão NMEA 0183, que são utilizadas pelos receptores GPS. Um diagrama de blocos do protótipo é apresentado seguido da avaliação de alguns dispositivos eletrônicos (microcontrolador, sensores, modem, conversor DC-DC etc.) necessários para a implementação do mesmo. Finalmente o trabalho trata do desenvolvimento do *firmware* do microcontrolador, incluindo as funcionalidades desejadas e as rotinas necessárias para a integração dos dispositivos.

O protótipo foi testado com sucesso demonstrando a capacidade de estabelecer a *geofence*, monitorar os sensores e transmitir mensagens (SMS), utilizando uma rede de telefonia celular, de acordo eventos pré-determinados.

Palavras-Chave: GPS, GSM, *geofence*, sensores, rastreamento.

ABSTRACT

This work deals with the integration of electronic devices for the implementation of a prototype capable of tracking it's position (using the Global Positioning System – GPS), establishing a geofence, reading sensors, estimating the status of it's power source (battery and charger) and transmitting messages (SMS) through a cellular network (GSM).

Initially the Global Positioning System and related concepts are presented, focusing then on the sentences, established by the NMEA 0183, used by GPS receivers. A block diagram of the prototype is then presented, followed by an evaluation of a few electronic devices (microcontrollers, sensors, modems, DC-DC converters etc.) necessary for it's implementation. Finally the development of the microcontroller's firmware is presented, including the desired functionalities and the routines necessary for the integration of the devices.

The prototype was successfully tested demonstrating it's abilty to establish a geofence, monitor it's sensors and transmit messages (SMS), using a cell phone network, related to previously determined events.

Key-words: GPS, GSM, geofence, sensors, tracking.

SIGLAS

ASCII – *American Standard for Information Interchange*

ADC – *Analog to Digital Converter*

CDMA – *Code Division Multiple Access*

DGPS – *Differential GPS*

DOP – *Dilution Of Precision*

EEPROM – *Electrically Erasable Programmable Read-Only Memory*

EIA – *Electronic Industries Alliance*

Fix – Posição obtida pelo receptor GPS

GGA – *Global Positioning System Fix Data*

GPRS – *Global Packet Radio Service*

GPS – *Global Positioning System*

GSA – *Satellite Status*

GSM – *Global System for Mobile Communications*

GSV – *Satellites in View*

HDOP – *Horizontal Dilution Of Precision*

IDE – *Integrated Development Enviroment*

ISP – *In-System Programming*

LT – Laboratório de Traçadores – COPPE/UFRJ

NMEA – *National Marine Electronics Association*

PDOP – *Positional Dilution of Precision*

ppm – Partes por milhão

PWM – *Pulse-Width Modulation*

RMC – *Recommended Minimum Sentence C*

SMS – *Short Message Service*

SNR – *Signal to Noise Ratio*

SRAM – *Static Random-Access Memory*

UFRJ – Universidade Federal do Rio de Janeiro

USART – *Universal Synchronus Asynchronus Receiver Transmitter*

UTC – *Universal Time Coordinated*

VDOP – *Vertical Dilution Of Precision*

ZDA – *Zulu Date & Time*

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 – Tema

Integração de dispositivos eletrônicos para aplicações nos meios acadêmico e científico, para aplicações não atendidas, completa ou parcialmente, por dispositivos comerciais.

Este trabalho utiliza-se de conhecimentos de eletrônica embarcada, especificamente os conhecimentos de eletrônica digital, programação e instrumentação.

1.2 – Delimitação

O objeto de estudo deste trabalho é a integração de dispositivos eletrônicos para o rastreamento de instrumentos científicos utilizados em trabalhos de campo; mais especificamente, serão integrados microcontroladores, GPS, modems (GSM/GPRS, rádio) além de sensores de corrente e tensão.

Atualmente estão disponíveis comercialmente dispositivos capazes de monitorar utilizando sistemas de navegação por satélite (GPS, GLONASS, Galileo) e modems (GSM/GPRS). Estes dispositivos porém são encontrados como sistemas prontos permitindo pouca ou nenhuma flexibilidade para alterações em seu modo de operação de acordo com as necessidades dos usuários. Outra classe de dispositivos, também disponíveis comercialmente, são capazes de fornecer individualmente funcionalidades como o rastreamento de posição (GPS), comunicação (modems), medição de grandezas físicas (sensores), processamento de dados (microcontroladores), conversão de sinais (A/D), etc. Este trabalho trata da utilização em conjunto desta segunda classe de dispositivos, permitindo obter funcionalidades específicas como a comunicação com determinados instrumentos científicos.

1.3 – Justificativa

Durante a realização de estudos de campo milhares ou até dezenas de milhares de reais/dólares em equipamentos são utilizados. Esses equipamentos estão sujeitos a defeitos, à ação do tempo e até a vandalismo, o que torna essencial monitorar tanto a localização quanto o estado de funcionamento dos equipamentos, seja para uma simples manutenção (troca de bateria/painel solar/carregador), quanto para uma substituição, ou até quem sabe a agregação de instrumentos evitando assim a perda de dias de trabalho ou até que o estudo tenha que ser refeito.

1.4 – Objetivos

O objetivo geral deste trabalho é a integração dos seguintes dispositivos: microcontrolador, GPS, modems (um ou mais) e sensores (corrente e tensão) de modo a monitorar a localização e o estado dos equipamentos.

Objetivos específicos:

1 → Criação de uma *geofence* utilizando um módulo GPS;

2 → Transmissão das coordenadas quando em posição externa à *geofence*;

3 → Transmissão do estado da bateria.

4 → Transmissão do estado do carregador da bateria.

1.4.1 – *Geofence*

Uma g*eofence,* nos termos deste trabalho, se refere a uma cerca imaginária criada com a ajuda de um dispositivo de rastreamento de posição. A cerca delimita uma área de interesse, quando tal área é ultrapassada um evento é gerado, um sistema pode então reagir, por exemplo, transmitindo uma mensagem referente a este evento. Para este trabalho deseja-se também que a área de interesse seja um quadrado de 400m de lado, cujo centro é definido pela primeira posição obtida pelo dispositivo.

1.5 – Metodologia

Inicialmente foi realizada uma revisão da literatura, particularmente no que diz respeito ao funcionamento do sistema de posicionamento global (GPS) e ao tipo e formato das informações (sentenças NMEA) disponibilizadas pelos dispositivos que o utilizam. Em um segundo momento foi escolhida uma plataforma que serviu de base para o desenvolvimento do protótipo, o que envolveu a escolha de uma família de microcontroladores, ambiente de desenvolvimento e finalmente o modelo a ser utilizado. A terceira etapa consistiu no desenvolvimento de um diagrama de blocos com o objetivo de explicitar os diferentes componentes necessários para o desenvolvimento do projeto, além da realização de uma pesquisa de componentes disponíveis, preferencialmente no Brasil, com valores razoáveis considerando sua aplicação e também na pesquisa de componentes equivalentes (ou idênticos) disponíveis no exterior. A quarta etapa envolveu o desenvolvimento e a simulação de rotinas utilizando a plataforma previamente escolhida e finalmente realizaram-se a montagem e os testes do protótipo.

1.6 – Descrição

No capítulo 2 serão apresentadas informações sobre o sistema de posicionamento global (GPS), sentenças utilizadas por receptores GPS e outros conceitos relacionados.

O capítulo 3 apresenta o projeto inicial e a seleção de componentes utilizados no protótipo.

O capítulo 4 trata do desenvolvimento do *firmware* do microcontrolador selecionado, apresentando um diagrama de estados, as bibliotecas utilizadas e as desenvolvidas, além das funcionalidades implementadas.

Os testes e resultados são apresentados no capítulo 5.

As conclusões serão apresentadas no sexto e último capítulo.

Capítulo 2

GPS/Sentenças NMEA

2.1 – Sistema de Posicionamento Global (GPS)

De acordo com [1], o Sistema de Posicionamento Global (GPS) é um sistema de navegação por satélite desenvolvido e operado pelo Departamento de Defesa do Governo dos Estados Unidos da América (EUA) que permite a usuários em terra, mar ou no ar determinar sua posição tridimensional, sua velocidade e o horário, 24 horas por dia, em todo tempo, em qualquer lugar do mundo, com uma precisão e exatidão superiores às de quaisquer outros sistemas de radionavegação atualmente disponíveis. O sistema é dividido em três segmentos: O Segmento Espacial, O Segmento de Controle e o Segmento de Usuários.

2.1.1 – O Segmento Espacial

O Segmento Espacial consiste de no mínimo 24 satélites em 6 órbitas a uma altitude de 20.200km acima da terra com uma inclinação de 55º e um período de aproximadamente 12 horas circulares. Os satélites de GPS transmitem continuamente uma mensagem contendo dados de tempo (provenientes de seus relógios atômicos), suas precisas órbitas (*ephemeris)*, um *Almanac* contendo informações rudimentares das órbitas de toda a constelação, e informações de correção de erro e saúde (estado dos satélites) de toda a constelação de satélites [2]. Todos os satélites transmitem na mesma frequência sendo eles distinguíveis graças a utilização de técnicas de multiplexação por código (CDMA – *Code Division Multiple Access*).

2.1.2 – O Segmento de Controle

O Segmento de Controle consiste de uma estação de controle central (localizada em Colorado Springs), cinco estações de monitoração e 3 antenas espalhadas pelo planeta. As estações de monitoração rastreiam os satélites coletando dados que são repassados para a estação de controle central que os utiliza para calcular com alta precisão as órbitas dos satélites, estas informações são transmitidas para os satélites

atualizando a mensagem que cada satélite transmite. Além disso a estação de controle central também atua ajustando o relógio dos satélites em relação à UTC e informações de saúde (estado) dos satélites da constelação.

2.1.3 – O Segmento de Usuários

O Segmento de Usuários consiste de receptores (processadores e antenas) que utilizam os dados transmitidos pelos satélites do sistema para obter sua posição, velocidade, tempo, etc.

Os receptores utilizam um processo de trilateração aonde, teoricamente, se busca um ponto presente na superfície de 3 esferas (centralizadas em 3 satélites) cujos raios, conhecidos como *pseudoranges*, são obtidos calculando a diferença do tempo entre o momento que a mensagem foi transmitida pelo satélite e o momento em que ela foi recebida, multiplicando esta diferença pela velocidade da luz (c). Na prática, porém, os receptores utilizam relógios com precisão insuficiente (alguns ppm), pois os valores obtidos serão multiplicados pela velocidade da luz (c), de modo que erros pequenos (por exemplo da ordem de 100ns) acarretariam em erros de dezenas (da ordem de 30m no exemplo anterior) de metros. Desse modo é necessário determinar uma quarta variável (o erro entre os relógios dos satélites e o relógio do receptor Δt) e por isso são necessários no mínimo quatro satélites, permitindo assim que sejam estimados a posição e o erro (Δt) utilizando um método de mínimos quadrados [3].

Muitos receptores GPS fornecem as informações obtidas utilizando sentenças definidas pela especificação NMEA 0183 (*National Marine Electronics Association* 0183).

2.2 – Conceitos relacionados

As sentenças definidas no NMEA 0183 e apresentadas na Seção 2.3 contém conceitos como diluição de precisão vertical (*VDOP*), horizontal (*HDOP*) e posicional (*PDOP*), que serão rapidamente abordados aqui.

2.2.1 – Diluição de Precisão (*DOP*)

A diluição de precisão (*DOP*) é uma medida determinada pela geometria relativa dos satélites visíveis em relação ao receptor. A informação enviada por cada satélite tem precisão limitada, se os satélites se encontram próximos, relativamente ao receptor (como representado na Figura 2.1), haverá uma maior interseção entre as esferas consideradas para determinar a posição e portanto uma diluição da precisão, relativa à posição obtida. Da mesma maneira se os satélites se encontram mais afastados (como representado na Figura 2.2) haverá uma menor interseção, ocorrendo também uma diluição da precisão, porém em grau menor. Os valores de diluição de precisão fornecidos não apresentam unidade podendo ser relacionados a graus de confiança, por exemplo valores menores que 1 são considerados ideais, entre 1 e 2 excelentes, entre 2 e 5 bons, etc. [4]. Para este projeto consideraram-se valores abaixo de 4 como confiáveis.

Figura 2.1 DOP I

Figura 2.2 DOP II

A Diluição de precisão vertical (VDOP) se refere ao grau de confiança relativo ao eixo vertical, ou seja aos valores de altitude obtidos pelo receptor, de maneira análoga a diluição de precisão horizontal (HDOP) refere-se ao grau de confiança relativo ao plano horizontal o que implica no grau de confiança das medidas de latitude e longitude. A diluição de precisão posicional (PDOP) engloba as duas outras medidas fornecendo um grau de confiança geral das informações obtidas.

2.3 – NMEA 0183

O padrão NMEA 0183 é um conjunto de especificações que abrange padrões de comunicação serial considerando níveis elétricos (EIA-422), taxa, paridade e até a definição do formato dos dados a serem transmitidos [5].

Os receptores GPS não aderem, de forma geral, ao padrão NMEA 0183, utilizando-se de comunicação serial, em diversos padrões (TTL, RS-232, LVTTL, etc...); fornecendo dados em código ASCII no formato de sentenças, como especificado no padrão.

2.3.1 – Sentenças NMEA 0183

De modo geral receptores GPS enviam as informações obtidas utilizando sentenças definidas pelo padrão NMEA 0183, o documento completo é vendido pela NMEA, porém partes significativas, como várias das sentenças, estão disponíveis na Inernet e em manuais de receptores GPS. Alguns receptores também enviam outras sentenças, criadas pelos seus fabricantes, que são similares às sentenças do padrão. As regras de formação das sentenças são apresentadas abaixo [5]:

1 → As sentenças se iniciam com o caractere "$";

2 → Os próximos caracteres identificam a fonte das sentenças, por exemplo, "GP" para GPS, "GL" para GLONASS e "PGRM" para sentenças proprietárias da empresa Garmin;

3 → Os próximos 3 caracteres, antes da primeira vírgula, identificam a sentença propriamente dita (ex.: "GGA" → *Fix Information,* "GSA*"* → *Overall Satellite Data*);

4 → Todos os campos seguintes, exceto o último, encontram-se delimitados por vírgulas;

5 → O último campo é terminado com um asterisco, que é imediatamente precedido por dois caracteres que representam um número hexadecimal de dois dígitos referente a uma soma de verificação (*checksum*).

6 → As sentenças são terminadas por caracteres CR (*Carriage Return*) e LF (*Line Feed*).

Exemplo – Sentença GLL(*Geographic Latitude and Longitude*)

$GPGLL,4916.45,N,12311.12,W,225444,A,*1D

"$" : Inicio da sentença.

"GP" : Fonte de sentença (receptor GPS).

"GLL" : Identificação da sentença.

"4916.45" : Latitude em graus (49) e minutos (16.45).

"N" : Latitude referente ao hemisfério norte .

"12311.12" : Longitude em graus (123) e minutos (11.12).

"W" : Longitude situada a oeste de Greenwich.

"225444" : Horário em que a posição foi obtida (22h54m44s - UTC).

"A" : Indica validade da posição (A → válido, V → inválido).

"*" : Indica o último campo.

"1D" : Soma de verificação (*Checksum*).

A soma de verificação (*checksum*) é realizada utilizando uma operação XOR (Ou exclusivo) na representação binária de cada caractere (cada byte) encontrado entre o caractere inicial ("$") e o asterisco [6] e [7].

2.3.2 – Sentenças "GP"

Dentre as diversas sentenças definidas no padrão NMEA0183, aquelas cujas fontes são receptores GPS (identificadas pelos caracteres "GP") são as de interesse para este trabalho, mais especificamente, as sentenças "GGA" (*Fix Information*), "GSA" (*Overall Satellite Data*), "RMC" (*Recommended Minimum Data for GPS)*, "GSV" (*Detailed Satellite Data e* "ZDA" (*Date and Time*), além da sentença "GLL" serão utilizadas. Estas sentenças foram selecionadas por fornecerem informações relevantes aos objetivos e por serem oferecidas por diversos receptores GPS considerados para este projeto.

A Sentença GLL apresentada acima fornece a posição representada pela latitude e longitude, além de indicar o horário, relativo à UTC (sigla em inglês de Tempo Universal Coordenado), em que tal posição foi obtida. O último campo também indica se aquela posição obtida é válida, ou seja, se o o receptor GPS já foi capaz de encontrar 4 ou mais satélites e já recebeu as informações necessárias para calcular sua posição.

A sentença GGA também fornece a posição representada por latitude e longitude, o horário em que a posição foi obtida e sua validade, representada por

diferentes valores, cada um referente a um método utilizado para a obtenção da posição. Além disso a sentença fornece também o número de satélites rastreados pelo receptor, a diluição de precisão horizontal (HDOP), a altitude acima do nível médio do mar, a altura do Geoide e finalmente a diluição de precisão horizontal (HDOP), dois campos vazios (exceto para o método DGPS de obtenção da posição) e o *checksum.*

```
$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

Where:
     GGA          Global Positioning System Fix Data
     123519       Fix taken at 12:35:19 UTC
     4807.038,N   Latitude 48 deg 07.038' N
     01131.000,E  Longitude 11 deg 31.000' E
     1            Fix quality: 0 = invalid
                               1 = GPS fix (SPS)
                               2 = DGPS fix
                               3 = PPS fix
                               4 = Real Time Kinematic
                               5 = Float RTK
                               6 = estimated (dead reckoning) (2.3 feature)
                               7 = Manual input mode
                               8 = Simulation mode
     08           Number of satellites being tracked
     0.9          Horizontal dilution of position
     545.4,M      Altitude, Meters, above mean sea level
     46.9,M       Height of geoid (mean sea level) above WGS84
                      ellipsoid
     (empty field) time in seconds since last DGPS update
     (empty field) DGPS station ID number
     *47          the checksum data, always begins with *
```

A sentença GSA fornece informações sobre a qualidade da posição obtida, indicando se a posição é válida e fornecendo dados para estimar sua precisão, como o número de satélites utilizados para obtê-la, as diluições de precisão (PDOP, HDOP e VDOP) e se a posição foi obtida baseando-se em informação anterior da altitude (2D *fix*) ou não (3D *fix*) [6].

```
  $GPGSA,A,3,04,05,,09,12,,,24,,,,,2.5,1.3,2.1*39

Where:
     GSA      Satellite status
     A        Auto selection of 2D or 3D fix (M = manual)
     3        3D fix - values include: 1 = no fix
                                       2 = 2D fix
                                       3 = 3D fix
     04,05... PRNs of satellites used for fix (space for 12)
     2.5      PDOP (dilution of precision)
     1.3      Horizontal dilution of precision (HDOP)
     2.1      Vertical dilution of precision (VDOP)
     *39      the checksum data, always begins with *
```

A sentença RMC é similar à sentença GLL fornecendo, além da posição, informações de hora, validade, data, velocidade e declinação magnética [6].

```
$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A

Where:
     RMC          Recommended Minimum sentence C
     123519       Fix taken at 12:35:19 UTC
     A            Status A=active or V=Void.
     4807.038,N   Latitude 48 deg 07.038' N
     01131.000,E  Longitude 11 deg 31.000' E
     022.4        Speed over the ground in knots
     084.4        Track angle in degrees True
     230394       Date - 23rd of March 1994
     003.1,W      Magnetic Variation
     *6A          The checksum data, always begins with *
```

A sentença GSV não é particularmente útil para os objetivos deste projeto, fornecendo exclusivamente informações sobre os satélites detectados pelo receptor. As informações fornecidas porém permitem um melhor entendimento de como o receptor recebe dados sobre a posição dos satélites e suas órbitas. As informações do número de satélites visíveis e o SNR do sinal de cada satélite podem vir a ser utilizadas para estimar indiretamente o quão exata é a posição obtida [6].

```
$GPGSV,2,1,08,01,40,083,46,02,17,308,41,12,07,344,39,14,22,228,45*75

Where:
      GSV          Satellites in view
      2            Number of sentences for full data
      1            sentence 1 of 2
      08           Number of satellites in view

      01           Satellite PRN number
      40           Elevation, degrees
      083          Azimuth, degrees
      46           SNR - higher is better
           for up to 4 satellites per sentence
      *75          the checksum data, always begins with *
```

A sentença ZDA fornece apenas informações de data e hora, fornecendo porém a hora local da zona referente a posição atual, o que pode ser interessante para a aplicação do projeto [6].

```
$GPZDA,hhmmss.ss,dd,mm,yyyy,xx,yy*CC
 $GPZDA,201530.00,04,07,2002,00,00*60

where:
        hhmmss    HrMinSec(UTC)
        dd,mm,yyy Day,Month,Year
        xx        local zone hours -13..13
        yy        local zone minutes 0..59
        *CC       checksum
```

Capítulo 3

Desenvolvimento do Diagrama de Blocos e Seleção de Hardware

3.1 – Diagrama de Blocos Inicial

A concepção deste projeto se baseou nas funcionalidades de um dispositivo *TraceME,* da empresa KCS, já utilizado em trabalhos de campo pelo Laboratório de Traçadores(LT). Segundo a empresa KCS um *TraceME* é um rastreador GPS com funcionalidades GPRS e capacidades avançadas de Entrada e Saída (I/O), que permite que você rastreie e/ou controle uma grande diversidade de dispositivos [8]. Em vários aspectos as funcionalidades deste dispositivo ultrapassam aquelas almejadas neste trabalho porém sua configuração e aplicação prática se mostraram complicadas, seja por uma documentação precária, seja por um comportamento errático onde o dispositivo, por exemplo, se mostra incapaz de se comunicar com a rede de telefonia celular, incapaz de obter uma posição válida ou até a utilização de todos os créditos associados à linha telefônica enviando seguidas mensagens de texto em poucos minutos.

A aplicação que inspirou este projeto consiste no rastreamento da posição de uma boia, na qual encontram-se um painel solar, uma bateria, um carregador para a bateria, instrumentos científicos e o dispositivo desenvolvido. A Figura 3.1 apresenta o diagrama de blocos inicial deste trabalho, em preto estão os blocos já utilizados pelo laboratório: painel solar (Pmax 10W, Vmax 17V, Imax 0,58A), uma bateria de moto, carro ou similares (12V) e um carregador/controlador de carga próprio para painéis solares e bateria de 12V. Em cinza encontram-se os blocos que compõem o protótipo: o receptor GPS é responsável pelo rastreamento da posição; o bloco GSM/GPRS permite a transmissão de dados através de rede de telefonia celular; o sensor de tensão permite avaliar o estado da bateria; o sensor de corrente permite que se avalie indiretamente o estado do carregador da bateria e finalmente o microcontrolador permite integrar todos os dispositivos e implementar a funcionalidade de *geofence*.

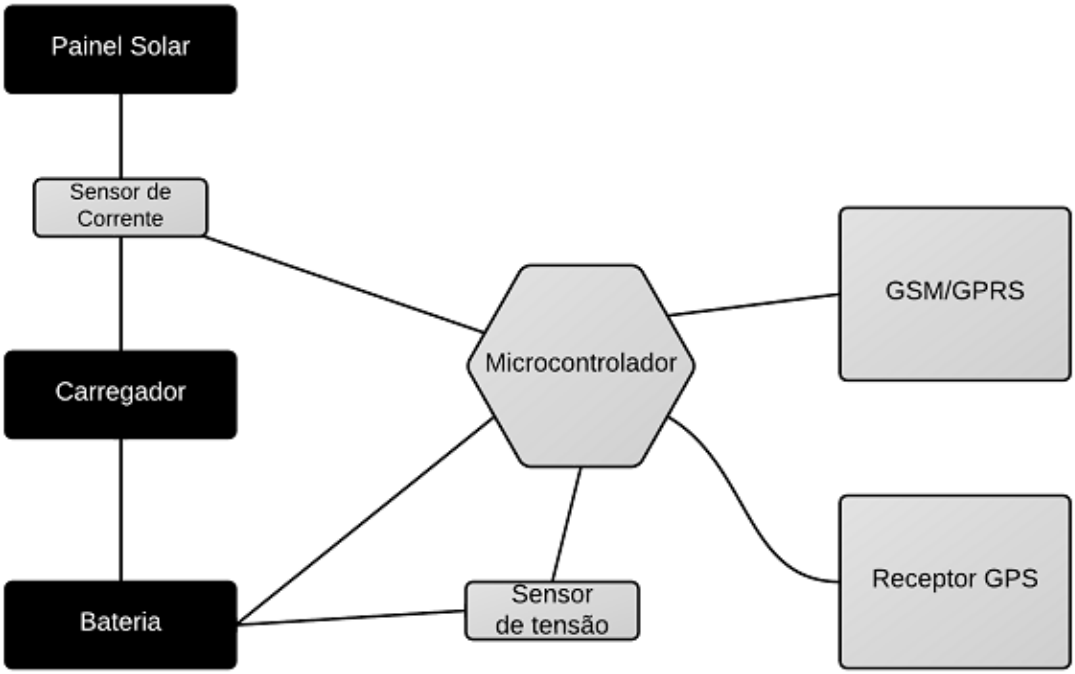

Figura 3.1 Diagrama de blocos inicial

3.2 – Microcontroladores

Para o desenvolvimento deste projeto foi necessária a escolha de uma família de microcontroladores. A princípio esta escolha pode parecer igual a de qualquer outro componente, como o receptor GPS ou os sensores a serem utilizados, porém o microcontrolador tem um papel central no processo de integração dos diversos componentes e a sua escolha implica, ou ao menos direciona, a determinação do ambiente de desenvolvimento. Para este trabalho foram considerados principalmente microcontroladores de 8-bits da Microchip (PIC) e da Atmel (AVR). A escolha considerou custo, ferramentas disponíveis para o desenvolvimento do *firmware*, ferramentas necessárias para a programação do microcontrolador (programador), as linguagens de programação disponíveis e a experiência prévia com microcontroladores. Os microcontroladores de ambas as empresas oferecem ambientes de desenvolvimento (MPLAB e AVR) que permitem utilizar linguagens de programação de mais alto nível como C. Ambos necessitam de programadores (ex: PICKit, AVRDragon) porém também aceitam *bootloaders* que permitem uma programação direta e no sistema (ISP – *In-System Programming*). Os custos são similares, porém relativos pois devido à grande

variedade de microcontroladores oferecidos e da compatibilidade dos programadores (PICkit limitado a determinados modelos PIC e AVR Dragon abrangendo a maioria dos AVR) os custos para uma aplicação específica podem variar em favor da família PIC ou da família AVR. Por outro lado alguns microcontroladores da Atmel são utilizados na plataforma de desenvolvimento Arduino, esta plataforma utiliza um *bootloader* e como suas placas já possuem circuitos integrados que permitem a comunicação com computadores, não necessitam de um programador. Na verdade é bastante comum utilizar tais placas como programadores ISP.

3.2.1 – Arduino

De acordo com [9], Arduino é uma plataforma aberta (*Open Source*) de prototipagem de dispositivos eletrônicos. A plataforma consiste de um ambiente de desenvolvimento, bibliotecas e placas de desenvolvimento, utilizando o compilador AVR-GCC (C e C++) e o utilitário ISP AVRDUDE, sendo baseada principalmente na família de microcontroladores ATmega da Atmel. A plataforma é dita aberta pois tanto as bibliotecas, o *bootloader* e até o projeto das placas estão livremente disponíveis. Esta plataforma é particularmente interessante pois, como já mencionado, não necessita de um programador, suas placas de desenvolvimento são versáteis e relativamente baratas, além de suas bibliotecas facilitarem a utilização das capacidades do microcontrolador (USART, ADC, PWM, etc.). O ambiente de desenvolvimento integrado (IDE sigla em inglês) apesar de apresentar limitações (algumas das quais possivelmente intencionais já que a plataforma almeja uma baixa barreira de entrada), como por exemplo a falta de suporte para depuradores (*debuggers*) e a falta de um simulador, é bastante funcional permitindo um início rápido do desenvolvimento, o que também é facilitado pela inclusão de um pseudoterminal que permite testes simples e rápidos do código. Apesar de utilizar seu próprio ambiente de desenvolvimento, as placas e as bibliotecas podem ser utilizadas com outros ambientes de desenvolvimento como o da própria Atmel (AVR Studio) ou o da fundação Eclipse (Eclipse IDE).

Existem várias placas de desenvolvimento Arduino oficiais e, devido a sua natureza aberta, inúmeras placas não oficiais compatíveis. Além disso existem também também inúmeras placas de expansão conhecidas como *Shields* como a placa na Figura 3.4. Uma rápida pesquisa aponta dois modelos mais recentes, Arduino Uno (Figura 3.2) e Arduino MEGA 2560 (Figura 3.3), como opções mais interessantes. Abaixo são apresentadas algumas especificações de ambas as placas:

Tabela 3.1 – Comparação de características das placas de desenvolvimento.

	Arduino Uno [10]	Arduino Mega 2560 [11]
Microcontrolador	ATmega328	ATmega2560
Tensão de operação	5V	5V
Tensão de Alimentação(recomendada)	7 – 12V	7 – 12V
Tensão de Alimentação(limites)	6 – 20V	6 – 20V
Terminais I/O Digitais	14 (6 capazes de PWM)	54 (15 capazes de PWM)
Entradas Analógicas	6	16
Corrente DC Terminais I/O	40mA	40mA
Corrente DC terminal 3.3V	50mA	50mA
Memória Flash	32KB (0.5KB para *bootloader*)	256KB (8KB para *bootloader*)
SRAM	2KB	8KB
EEPROM	1KB	4KB
Frequência de operação	16MHz	16MHz

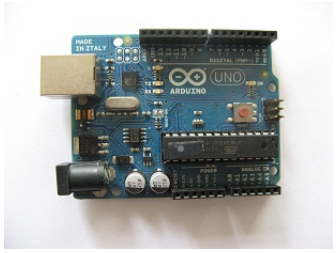

Figura 3.2 Arduino Uno.

Figura 3.3 Arduino MEGA 2560.

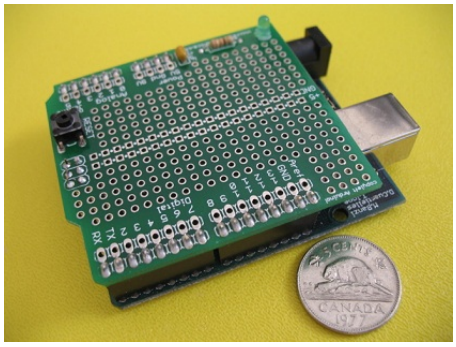

Figura 3.4 *Prototyping Shield* compatível com placas Arduino.

As especificações acima indicam placas bastante similares, existindo basicamente diferenças entre o número de pinos e a a memória disponível, diferenças diretamente relacionadas ao microcontrolador utilizado.

O diagrama de blocos inicial (Figura 3.1) indica que além de ser capaz de interagir com sensores, o microcontrolador deve ser capaz de se comunicar com um modem GSM/GPRS e um receptor GPS o que implica a presença de dois *USART* (*Universal Synchronous and Asynchronous serial Receiver and Transmitter*). Examinando os *datasheets* ([12] e [13]) de ambos os microcontroladores verifica-se a existência de apenas um "*USART*" no ATmega328 e de 4 no Atmega2560, apesar de já existirem bibliotecas capazes de implementar tal funcionalidade em software, a existência de 4 no segundo microcontrolador facilita uma possível expansão das funcionalidades do projeto. Além disso considerando a pequena quantidade de *SRAM* (*Static Ramdom-Access Memory*) disponível no microcontrolador ATmega328 (2KB) a placa Mega 2560 se mostrou a mais interessante, vale ressaltar porém que o microcontrolador Atmega328 esta disponível no encapsulamento *PDIP* (Plastic Dual In-line Package) o que facilitaria a construção de uma versão futura do projeto utilizando uma placa com apenas os componentes necessários, consequentemente reduzindo custos.

3.3 – Receptores GPS

Existem vários receptores GPS disponíveis no mercado. Analisando especificações de desempenho nenhum dos modelos se destacou, isto ocorre pois os fabricantes nem sempre fornecem as mesmas informações, o que já dificulta uma comparação, e especificações como a de sensibilidade não deixam claro ao que se referem (implicitamente parecem se referir à pior relação sinal ruído em que o receptor consegue obter os dados sem erros) ou em que situação foram auferidas. Outras especificações como o consumo de energia, limites operacionais, tempo de aquisição, etc. são muito similares ou idênticas. A escolha do receptor considerou então outros aspectos como a tensão de alimentação, a interface de comunicação, a interface mecânica e o custo. Para cada um destes aspectos foram consideradas mais interessantes as especificações abaixo:

1) Tensão de alimentação:	5V (compatível com a placa selecionada)
2) Interface de comunicação:	Serial TTL (compatível com a placa selecionada)
3) Interface mecânica:	Compatível com a placa selecionada
4) Custo:	O menor possível

Dentre os receptores encontrados o SkyNav SKM53 (SKM53) e o Sparkfun Venus GPS se destacaram. Segundo seu *datasheet* [14], o SKM53 atende a 1ª e 4ª especificações, além de parcialmente a 2ª (LVTTL.), já o Sparkfun Venus, de acordo com seu respectivo *datasheet* [15], atende a 2ª e 3ª especificações parcialmente, destacando-se porém por utilizar uma antena externa. A maioria dos outros receptores encontrados não atendiam nem a 1ª nem a 3ª especificações, alguns deles atendendo parcialmente a 2ª e outros nem isto. A especificação de custo foi considerada por último, sendo que outros receptores apresentaram custo similar ao do SKM53, porém atendendo a menos especificações que este. O SKM53 foi selecionado pois o outro modelo que se destacou se tornou indisponível durante a avaliação dos diferentes receptores. Após a seleção do receptor descobriu-se também a possibilidade de configurar a taxa de transmissão, as sentenças transmitidas e a periodicidade em que as sentenças são transmitidas de acordo com o número de posições obtidas pelo receptor (ex: enviar a sentença GSA a cada duas posições obtidas).

3.3 – Modem GSM/GPRS

A variedade de modems GSM/GPRS disponíveis localmente para este tipo de aplicação é limitada. Além disso dentre os disponíveis vários apresentam características mecânicas indesejáveis, seja pela utilização na interface de cabos não padronizados, por necessitarem de um encaixe difícil de ser encontrado ou até pela necessidade de desenvolver uma placa para sua utilização. Os modelos mais interessantes encontrados são *Shields* compatíveis com as placas Arduino, baseados no módulo SIM900 da empresa SIMCom. A Figura 3.5 apresenta o modelo selecionado: *GPRS Shield* da empresa *Seedstudio*. Segundo [16], o *GPRS Shield* permite que uma rede de telefonia celular GSM seja utilizada para receber e transmitir dados de três maneiras diferentes, SMS (*Short Message Service*), áudio e serviço GPRS. O modelo foi selecionado devido

a maior documentação disponibilizada [16] e [17]. Para a transmissão de dados será necessário também avaliar a cobertura das redes de celular disponíveis no local de utilização. Em casos de falta cobertura, o que é bem possível já que segundo [18] a tecnologia GSM tem um alcance que varia de 1km (áreas urbanas) até 35km (áreas rurais), pode-se utilizar um par de rádio modems que se comunicam com o microcontrolador utilizando um conversor de níveis apresentado na Seção 3.6.

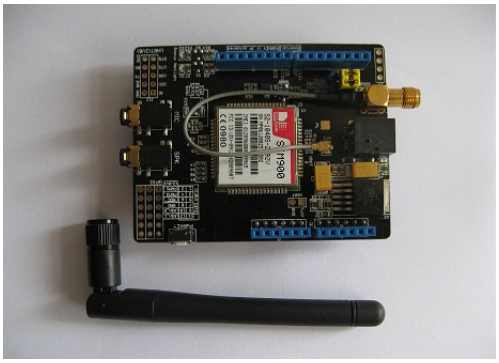

Figura 3.5 Seedstudio GPRS Shield

3.4 – Sensores

O diagrama de blocos inicial indica a necessidade de dois sensores, um de tensão que fornecerá o estado da bateria e outro de corrente que auxiliará na estimação do estado do carregador da bateria. Além disso o microcontrolador selecionado apresenta um conversor analógico-digital (ADC) multiplexado que permitirá a leitura destes sensores.

3.4.1 – Sensor de tensão

A leitura da tensão da bateria é simples considerando que o microcontrolador estará na mesma referência de tensão, sendo necessário apenas a utilização de um divisor de tensão para adequar a tensão da bateria (12V) à referência utilizada pelo ADC do microcontrolador. O divisor utiliza resistores de 1% com valores 10KΩ e 4K99Ω. Estes valores foram escolhidos considerando uma referência do ADC de 5V e que a tensão da bateria poderia chegar a 15V.

3.4.2 – Sensor de corrente

A leitura da corrente fornecida pelo painel solar para o carregador da bateria pode ser realizada de várias maneiras, por exemplo, utilizando um resistor de potência com pequeno valor colocado em série entre o painel e o carregador. Esta implementação apresenta alguns problemas, principalmente no que se refere a leitura pelo ADC do microcontrolador que aceita em cada entrada valores de tensão (que seriam subtraídos) de até VCC+0.3V (5.3V) e cada entrada neste cenário estaria conectada a tensões consideravelmente superiores fornecidas pelo painel solar. Algumas soluções foram consideradas e optou-se pelo sensor ACS756SCA-050B (ACS756), apresentado nas Figuras 3.6 e 3.7, que utiliza o efeito Hall para medir a corrente e fornece como saída uma tensão entre VCC/2 e VCC para correntes que o percorrem do seu pino 4 para o 5 e de VCC/2 a GND para correntes no outro sentido. O *datasheet* indica uma sensibilidade de 40mV/A para uma alimentação (VCC) de 5V e estima-se uma corrente máxima de 5A o que geraria tensão máxima de 2.7V (VCC/2 + 40mV*5) [19]. Como não se espera que a corrente percorra o sensor nas duas direções é possível subtrair a tensão constante VCC/2, obtendo apenas a variação da saída do sensor e amplificar esta, melhor utilizando a faixa de atuação do ADC. A Figura 3.8 apresenta um amplificador operacional na configuração de subtrator com ganho de aproximadamente 25 (R1 = 6k2Ω e R2 = 158kΩ) e outro na configuração de seguidor de tensão (ambos encontrados no CI TLV272 [20]), desacoplando um divisor de tensão com dois resistores de 1% com o valor de 1k05Ω, assim a relação de escala será de 1V/A de 0 a 5V.

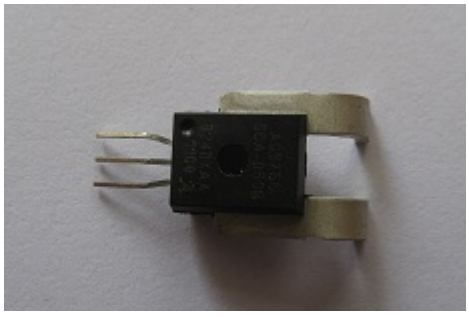

Figura 3.6 ACS756

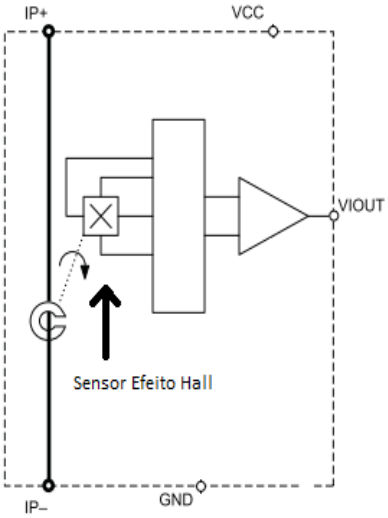

Figura 3.7 Diagrama do sensor ACS756 (modificado de [19]).

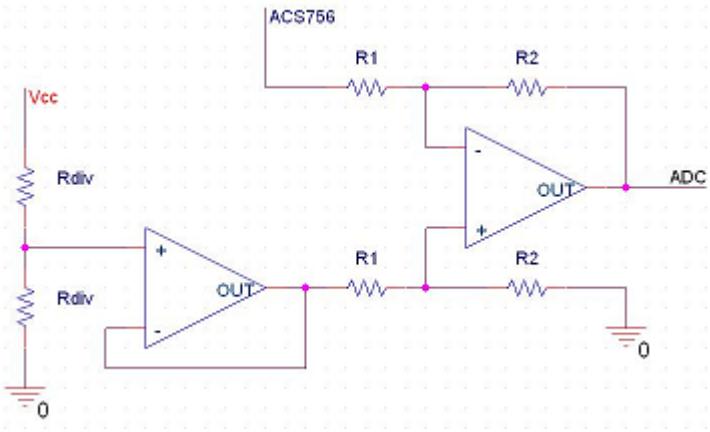

Figura 3.8 Amplificadores

3.5 – Conversor DC-DC (5V)

A bateria indicada no diagrama de blocos será uma bateria comum de carro (ou de moto) que tipicamente fornece uma tensão de 12V, a maioria dos componentes porém operam com uma tensão de 5V. A placa Arduino selecionada contém um regulador linear capaz de fornecer tal tensão, porém devido à natureza do funcionamento dos reguladores lineares, a corrente que ele é capaz de fornecer é bastante limitada (800 mA), já que ele teria que dissipar 7W/A e literalmente se utilizaria 240% da energia realmente necessária. Para o desenvolvimento inicial utilizando apenas o microcontrolador e o receptor GPS, a corrente necessária teoricamente estaria em torno de 150mA, especificação que o regulador linear foi capaz de atender. Em um segundo momento do desenvolvimento adicionou-se o modem GSM/GPRS, o regulador linear foi capaz de alimentar os dispositivos porém rapidamente se aqueceu de tal forma que tornou clara a necessidade de uma outra fonte de 5V.

Na concepção do projeto se imaginava que seria possível encontrar facilmente fontes de baixo custo e compactas que atendessem às necessidades deste projeto. Após uma pesquisa encontrou-se apenas fontes com relativo baixo custo no exterior ou fontes maiores, dimensionadas para grandes potências (ex.: 50W) com custo relativo a sua capacidade. A primeira solução imaginada para este problema foi utilizar um regulador linear com um dissipador. Esta solução não era desejável devido a sua baixa eficiência. Felizmente na pesquisa desta solução foi encontrada a família *SIMPLE SWITCHER* de reguladores chaveados de tensão da Texas Instruments, mais especificamente os circuito

integrado LM2676-5.0, que permite construir uma fonte chaveada com apenas alguns componentes externos (diodo, indutor e capacitores) [21]. A Figura 3.9 abaixo apresenta um exemplo de aplicação extraído do *datasheet* do LM2676.

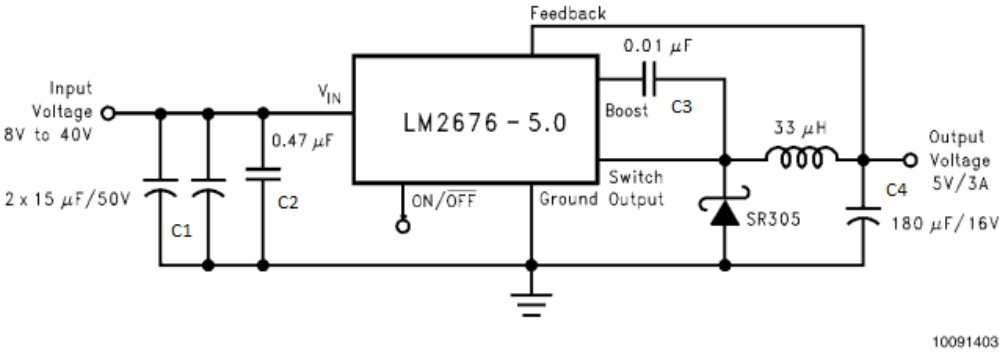

Figura 3.9 Exemplo de fonte de alimentação chaveada *step-down* utilizando LM2676 – 5.0 (modificado de [21] Fonte: Texas Intruments, 2012).

Baseando-se nas informações do *datasheet* foram selecionados o seguintes componentes para montar a fonte de 5V:

– Capacitor de Tântalo 22uF, 35V (C1).

– (2x) Capacitor de Tântalo 100uF, 16V (C4).

– Capacitor Cerâmico 0.47uF (C2).

– Capacitor Cerâmico 0.01uF (C3).

– Retificador Schottky MBR1545CT (apenas um dos diodos encapsulados).

– Indutor 33uH, 3A.

A fonte foi montada sobre um *Shield* ProtoshieldBR (Figura 3.10) para utilização mais simples com a placa de desenvolvimento.

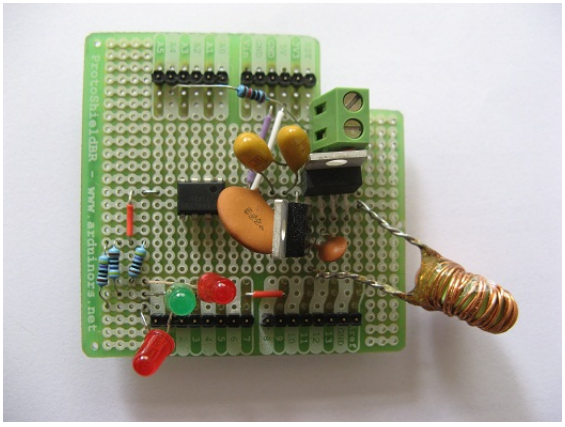

Figura 3.10 Fonte chaveada – LM2676 – 5.0

3.6 – Outros componentes

Considerando a possibilidade de expansão das funcionalidades do projeto adicionou-se um conversor de níveis lógicos MAX232N e os capacitores necessários (1uF) para seu funcionamento, visando seu uso com os dois *USARTS* ainda disponíveis do microcontrolador. Foi adicionado também um *DIP Switch* que poderá ser utilizado para propósitos de configuração.

3.7 – Diagrama de Blocos Final

A Figura 3.11 apresenta o diagrama de blocos final. Este diagrama detalha os blocos previstos no diagrama inicial, de acordo com as avaliações apresentadas anteriormente. O diagrama final também apresenta outros blocos como a fonte de 5V, o conversor de níveis MAX232N e um simples *DIP Switch*. A Figura 3.12 apresenta a placa criada para abrigar o receptor GPS, o circuito integrado MAX232N e o *DIP Switch*. Esta placa se encaixa sobre a placa de desenvolvimento Arduino MEGA 2560 e também permite o encaixe do *GPRS Shield.* A Figura 3.13 apresenta o protótipo montado, incluindo o *Shield*, encaixado sobre o *GPRS Shield,* que contém a fonte de 5V, e o sensor de corrente e o divisor de tensão.

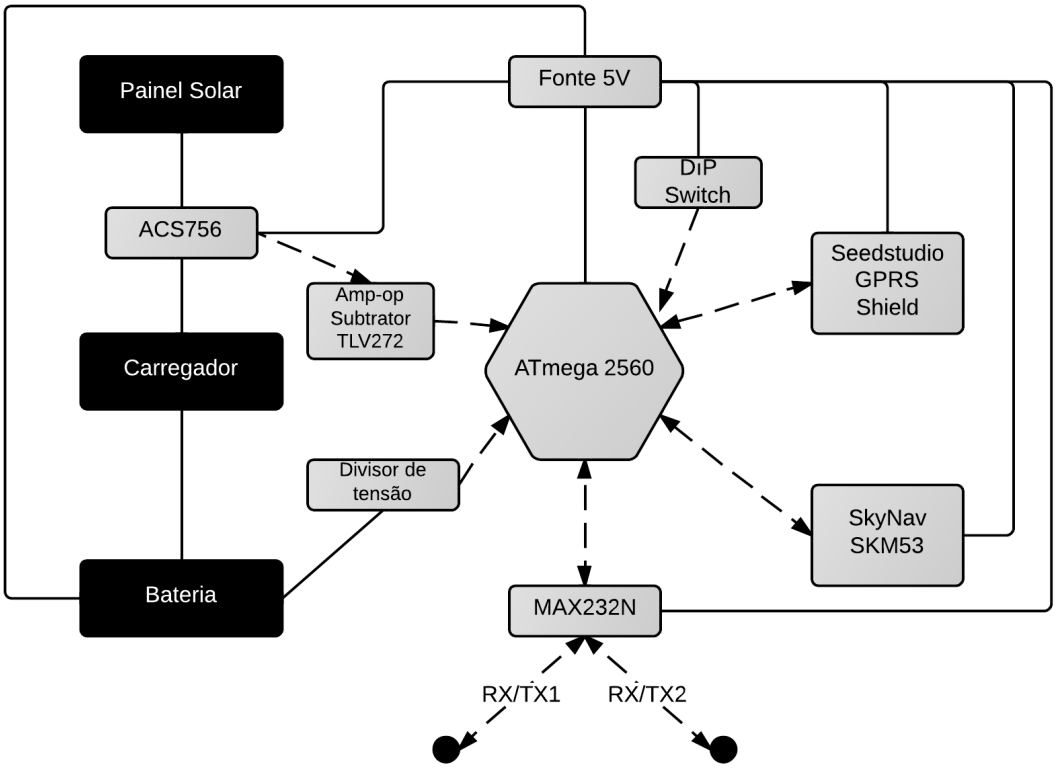

Figura 3.11 Diagrama de Blocos final

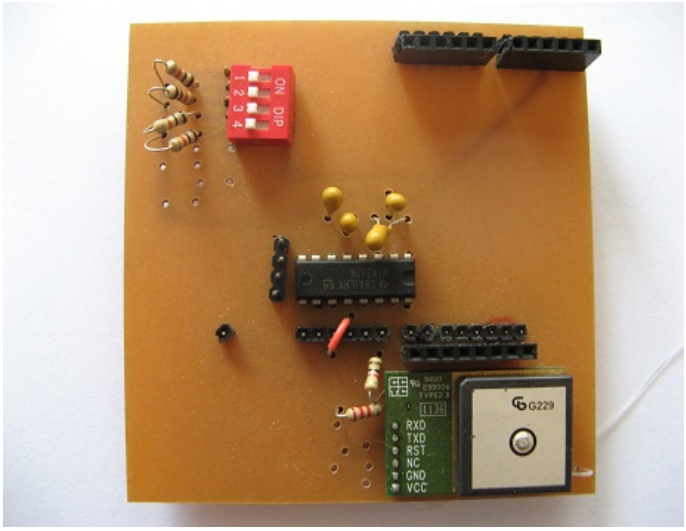

Figura 3.12 Placa criada para abrigar o receptor GPS e o conversor de níveis MAX232

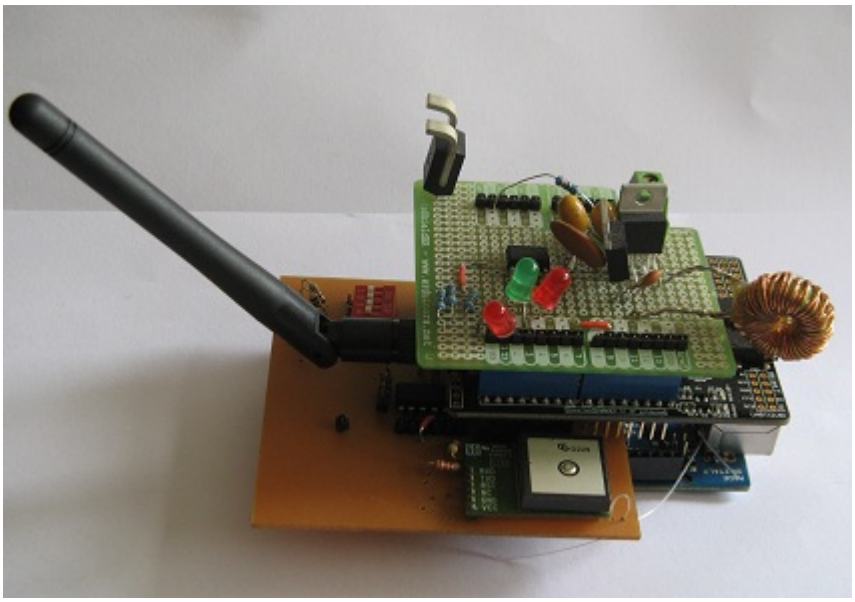

Figura 3.13 Protótipo Montado

Capítulo 4

Projeto do *Firmware (*ATmega2560)

4.1 – Fluxograma do *Firmware*

Antes de iniciar o desenvolvimento do *firmware* é importante avaliar as funcionalidades desejadas. Para ajudar nessa avaliação e esclarecer o funcionamento do *firmware*, criaram-se fluxogramas que facilitaram a identificação de métodos, variáveis e até funcionalidades inicialmente não previstas.

Como definido nos objetivos deseja-se monitorar a posição, o estado da bateria, o estado do carregador e transmitir essas informações. A transmissão de dados, que será realizada com base em eventos (saída da *geofence*, bateria com problemas, carregador com problemas, etc.), e a aferição dos estados da bateria e do carregador podem ser realizados esporadicamente sem prejuízos. Por outro lado a funcionalidade de rastreamento requer uma constante atualização dos dados, o que é refletido também no receptor GPS que envia um grupo de sentenças a cada 1 segundo.

Outra característica importante é a estrutura adotada pelo ambiente de desenvolvimento (Arduino), que disfarça a estrutura comum de um programa em C ou C++, implementando na função *main()* uma função *setup()*, executada apenas uma vez quando o microcontrolador é ativado, e uma função *loop()*, que nada mais é que um iteração infinita, sendo o código desenvolvido executado dentro destas duas funções.

O fluxograma final apresentado na Figura 4.1 reflete esta constante atualização dos dados e a estrutura adotada pela plataforma. Dentre os vários blocos é interessante destacar o bloco de configuração inicial, o bloco de sincronização Serial-GPS e o bloco de identificação das sentenças: O bloco de configuração inicial será abordado na Seção 4.3. O bloco de sincronização Serial-GPS descarta caracteres até encontrar o inicio de uma sentença sincronizando o microcontrolador e o receptor GPS. O bloco de identificação das sentenças representa um método relativamente simples porém importante pois reflete a estrutura S*witch-Case* adotada para o tratamento das diferentes sentenças recebidas.

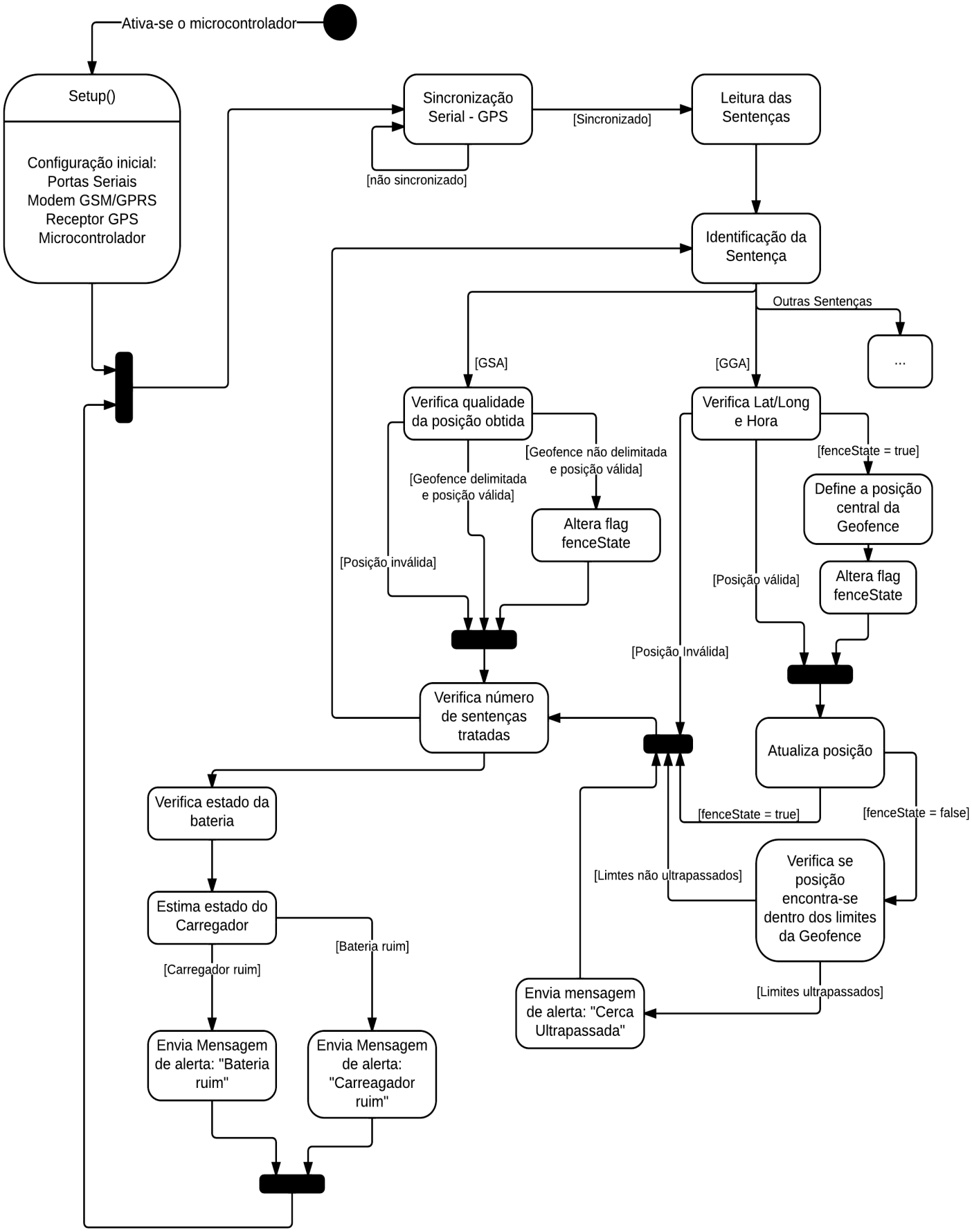

Figura 4.1 Fluxograma final

4.2 – Simulador GPS

O início do desenvolvimento do *firmware* ocorreu de forma paralela ao desenvolvimento do *hardware* e a seleção de componentes. Isto foi possível pois utilizou-se o programa (disponível numa versão *free trial* [22]) GpsSimul da empresa *SailSoft*. Este programa gera sentenças NMEA-0183 e as envia utilizando portas seriais disponíveis no computador. Utilizando um conversor USB-Serial-TTL que se apresenta para o sistema operacional como uma porta serial comum, foi possível alimentar o microcontrolador com sentenças NMEA e iniciar o desenvolvimento rapidamente, o que também facilitou a compreensão das sentenças NMEA, pré-requisito essencial para o desenvolvimento da classe **GPSDataSKM53.**

4.3 – Rotinas de Inicialização

O fluxograma da Figura 4.1 apresenta uma etapa de configuração inicial que é executada dentro da função *setup*(). Além da configuração de portas seriais e dos terminais do microcontrolador, são também realizadas a leitura de um *DIP switch,* o envio de uma sentença proprietária de configuração do receptor GPS e o acionamento do *GPRS Shield*.

O *DIP switch* apresenta quatro chaves que conectam ou não quatro terminais do microcontrolador a fonte de 5V. A leitura destes terminais é armazenada em variáveis que podem ser utilizadas para configuração do dispositivo.

A sentença enviada foi encontrada com pouquíssima documentação e segue o seguinte modelo: $PMTK314,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0*28<CR><LF>, aonde o 1º zero se refere a GLL, o 2º a RMC, o 3º a VTG, o 4º a GGA, o 5º a GSA, o 6º a GSV e o 18º a ZDA. Os zeros devem ser substituídos por 1,2,3,4 ou 5 para determinar o número de posições que devem ser recebidas antes que o receptor envie a sentença.

4.4 – Estado da Bateria e do Carregador

A estimativa do estado do carregador e da bateria é análoga a solução de um problema subdeterminado. Medindo apenas a tensão da bateria é impossível determinar se o problema é da bateria, do carregador ou até do painel solar. Para melhorar a estimativa podem-se adicionar outras medidas, neste caso mede-se também a corrente entre o painel solar e o carregador, sendo possível também medir a tensão gerada pelo painel solar.

A medida de corrente entre o painel e o carregador apresenta algumas dificuldades (assim como uma possível medida da tensão gerada pelo painel) pois a energia elétrica gerada pelo painel solar varia de acordo com o tempo e também com o horário, desta forma é necessário verificar a validade da medida antes da utilização da mesma na determinação do estado do carregador. O método de validação utilizado é simples, são considerados válidos apenas valores obtidos dentro um determinado intervalo de tempo, a cada dia (ex.:10:00 a 16:00). Além disso o valor medido da corrente não é utilizado diretamente, verificando-se apenas a existência de corrente acima de um determinado limiar. Este limiar não está necessariamente associado a uma carga esperada e sim a uma carga mínima e ao ruído inerente ao sensor. A validação e o valor medido estão intrinsecamente associados aos intervalos e limiares utilizados, estes sendo escolhidos visando evitar falsos positivos. Provavelmente ambos serão alterados de acordo com o uso, melhorando assim sua pertinência.

A utilização da corrente medida, mesmo considerando hipotéticos intervalos e limiares perfeitos é limitada, já que a sua pertinência depende de múltiplas variáveis (tempo, carga, intervalos, etc.) o que potencialmente geraria múltiplos falsos positivos, por isso é interessante adotar também uma validação do conjunto (painel, carregador e bateria), de forma que enquanto o conjunto estiver saudável a medida da corrente não é utilizada. Felizmente isto pode ser inferido a partir da tensão da bateria, ou seja, se a tensão da bateria estiver acima de um nível (11,5V) considera-se que ela esteja saudável e que esteja sendo carregada, abaixo deste nível utiliza-se a medida de corrente para inferir se há alguma problema com o carregador e finalmente se a bateria atinge um nível inferior(10,5V) considera-se que a bateria esteja com problemas (no caso de um problema com o carregador um aviso já teria sido enviado antes do aviso da bateria ser enviado).

O estado do carregador poderá também ser inferido pelo usuário avaliando a tensão da bateria ao longo do tempo, por exemplo, se a tensão da bateria estiver em contínua queda é possível que ela não esteja sendo carregada. Para permitir esta avaliação atribui-se à variável **dailyMsgTime** o valor da hora do dia (ex.: 12:00:00 → 120000) em que deseja-se que o protótipo transmita a mensagem.

4.5 – Transmissão de Dados

A transmissão de dados é realizada através de uma mensagem SMS, utilizando o modem GSM(*GPRS Shield*). Para isso foi desenvolvida a classe **GsmRT** que encapsula, dentre outras funcionalidades, os comandos AT necessários para a transmissão de uma mensagem SMS. A transmissão, de forma opcional, também pode ser realizada utilizando um modem que atue de forma "invisível", ou seja, que tanto para o transmissor quanto para o receptor ele se comporte como um simples cabo. Neste caso a transmissão ocorre como uma simples mensagem enviada pela porta serial do microcontrolador.

Para evitar que mensagens do mesmo tipo de evento (estado da bateria e carregador ou estado da *geofence*) sejam continuamente transmitidas, e portanto utilizem todos os créditos associados à linha telefônica, um contador é associado a cada tipo de evento o que permite configurar um intervalo entre as mensagens.

Para qualquer método de transmissão de dados, uma rotina gera uma mensagem contendo as informações de latitude, longitude, tensão da bateria, corrente entrando no carregador e um alerta referente ao evento que gerou a mensagem: Falha na Bateria, Cerca Delimitada, Cerca Ultrapassada e Falha do Carregador.

4.6 – Bibliotecas

Além das bibliotecas da plataforma Arduino foi utilizada também a biblioteca **PString**, e foram desenvolvidas as classes **GpsDataSKM53** e **GsmRT**, que encapsulam os dados e as funções referentes a interação com o receptor GPS SKM53 e o "*GPRS Shield*".

4.6.1 – Bibliotecas Arduino

Serial: Desenvolvida como uma classe C++, encapsula a configuração e utilização das "*USART*" incluindo vários métodos como:

begin(*baud***)** – Configura a utilização dos respectivos pinos como TX e RX e taxa baud;

available() – Retorna o número de bytes disponíveis para serem lidos da porta serial;

print(*variável***)** e p**rintln(***variável***)** – envia pela porta serial os dados contidos na *variável* como caracteres ASCII, adicionando, no caso da função println caracteres CR (*carriage return*) e LF (*line feed*) após os dados.

read() – Retorna o primeiro byte disponível no buffer de recepção.

ADC: **analogReference(***referência***)** – Permite selecionar a referência a ser utilizada pelo ADC passando no lugar da *referência* as seguintes palavras chave:

"DEFAULT" → VCC

"INTERNAL" → referência interna (varia com a placa utilizada - não disponível no Arduino MEGA).

"INTERNAL1V1"→ referência interna de 1.1V (apenas Arduino MEGA).

"INTERNAL2V56"→ referência interna de 2.56V(apenas Arduino MEGA).

analogRead(*pino***)** – Retorna um inteiro de 0 a 1023 referente a conversão A/D e a referência utilizada no respectivo pino.

Temporização: **delay(***tempo***)** – Paralisa a execução do programa pelo valor da variável *tempo* em milissegundos.

Terminais Digitais E/S: **pinMode(***pino***)** – Configura um terminal do microcontrolador como de Entrada ou Saída.

digitalRead(*pino***)** – Realiza a leitura de um terminal retornando 1 ou 0.

digitalWrite(*pino***) –** Modifica a tensão do terminal para o equivalente ao valores lógicos 1 (5V) ou 0 (0V).

4.6.2– Biblioteca PString

PString é uma biblioteca desenvolvida por Mikal Hart disponível no site Arduiniana que facilita a manipulação de *strings* (do tipo da linguagem C) sem apresentar os custos de memória relacionados ao uso de *Strings* (do tipo da linguagem C++). A biblioteca funciona operando sobre buffers que nada mais são do que vetores de caracteres (*string* da linguagem C) previamente alocados, não realizando alocação dinâmica de memória. A biblioteca garante que nunca ocorrerá um estouro do *buffer* (potencias dados que causariam este estouro são descartados), o *buffer* sempre consistirá de memória válida (o string previamente alocado) e que o *buffer* sempre apresenta um caractere *NULL* que garante sua validade como string da linguagem C. Segundo [23], o uso da biblioteca implica num aumento do tamanho do programa de 100 a 600 bytes (o que não é uma grande limitação), porém os objetos **PString** utilizam apenas 8 bytes além da memória *SRAM* alocada para seus respectivos *buffers*. A biblioteca se comporta como uma classe C++ apresentando os seguintes métodos:

Print(*texto*) **println**(*texto*) – Funcionam de forma análoga aos métodos acima da biblioteca **Serial**, porém guardando os dados em *buffer* (string do tipo C).

"=" – sobrecarrega o operador permitindo que ele seja utilizado como para outras variáveis básicas. Ex.: PString str = "teste";

"+=" – sobrecarrega o operador implementando a operação de concatenação.

"==" – sobrecarrega o operador implementando a operação de comparação.

begin() – determina que o próximo caractere seja armazenado na posição inicial do *buffer*.

4.6.3 – Classe GpsDataSKM53

A classe **GpsDataSKM53** foi desenvolvida principalmente para encapsular dados como a latitude, longitude, hora (UTC-3), validade da posição e altitude. Foram também desenvolvidos métodos que permitem extrair estes dados das sentenças GLL, GGA e GSA; métodos *getters* para os dados; e métodos para lidar com a *geofence:*

setLatLongGLL(char* inData**)** – Extrai a Latitude e Longitude de uma sentença GLL.

readGGA(char* inData**)** – Extrai Latitude, longitude, hora e altitude de uma sentença GGA.

getNmeaType(char* inData**)** – Determina o tipo da sentença NMEA, retornando um inteiro referente a cada sentença: 1– GSA, 2 – GGA, 3 – GLL.

chkFixGSA(char* inData, int tamanho**)** – Extrai as informações de validade da posição e as diluições de precisão (VDOP, HDOP) e as utiliza para determinar se a posição recebida é considerada válida.

A Figura 4.2 apresenta o formato das informações de latitude e longitude fornecidas pelo receptor GPS.

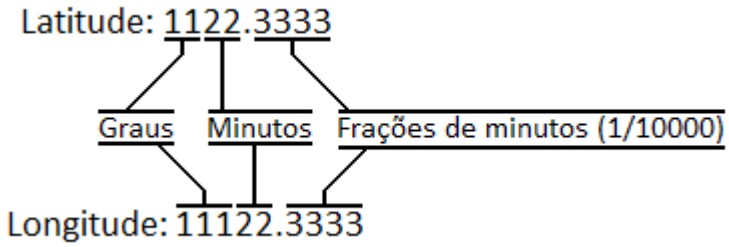

Figura 4.2 Formato da Latitude e Longitude – receptor GPS.

Para simplificar sua manipulação, armazena-se nas variáveis "**latUtil" e "loUtil**", respectivamente, a latitude e a longitude convertidas para frações de minutos:

Valor = 60x10 000x(graus) + 10 000x(minutos) + (frações de minutos)

getLatUtil() – Retorna o valor da variável "**latUtil**"

getLoUtil() – Retorna o valor da variável "**loUtil**"

getLaDeg() – Retorna a parcela da latitude referente aos graus

Assim como a função acima existem funções análogas para as parcelas de minutos e fração (1/10000) de minutos, existindo também as mesmas funções referentes à longitude (**getLaMin()**,**getLaSec()**,**getLoDeg()**,**getLoMin()**,**getLoSec()**).

getFixState() – Retorna o estado do *flag fix* que é utilizado para indicar a validade da posição.

s**etFixState(***boolean fixState***)** – Permite que o *flag fix* seja alterado.

chkGeoFenceOk(*long latitude, long longitude***) –** Compara a posição fornecida com a posição original, determinando se os limites da *geofence* foram ultrapassados.

4.6.4 – Classe GsmRT

A classe **GsmRT** foi desenvolvida para interagir com o *GPRS Shield* (modem GSM/GPRS utilizado), e apresenta os seguintes métodos:

onOff() – Liga/Desliga o modem variando a tensão em uma de suas portas seguindo uma temporização específica.

off() – Desliga o modem enviando pela porta serial um comando AT.

setPhoneNumber(*char opt, char* number***) –** Armazena um número de telefone como uma *string* (lingaugem C) de acordo com o caractere *opt*.

sendSMS(*char opt, char*message***)** – Envia através de SMS a mensagem contida em *message* para o telefone referido pelo caractere *opt.*

4.7 – *Geofence*

A implementação da *geofence*, como definida na Seção 1.4.1, depende apenas do armazenamento de uma posição inicial (centro da área de interesse) e da constante atualização da posição. Esta funcionalidade foi implementada utilizando dois objetos da classe **GpsDataSKM53**: **"gps***"* e **"gfence**". O primeiro é constantemente atualizado com as informações de posição, hora, diluição de precisão, validade da posição, etc. Já o segundo contém as informações que delimitam a posição inicial, obtidas assim que o receptor envia uma posição válida, e os valores de latitude e longitude se estabilizam. Este processo de estabilização não está refletido no fluxograma da Figura 4.1 por

motivos de clareza e consiste em descarte das primeiras posições válidas fornecidas pelo receptor GPS.

A área de interesse é delimitada como uma variação (para mais ou para menos) de 200 metros, na direção da latitude ou da longitude, em relação à posição inicial. Por se tratar de uma variação relativamente pequena em relação a curvatura da terra, podemos aproximar a área por um plano. A classe **GpsDataSKM53** porém, armazena informações de posição, como latitude (ex.: "**latUtil"**) e longitude (ex.: "**loUtil"**), e não informações de distância, que permitiriam avaliar diretamente se o protótipo se encontra na área de interesse.

A relação entre a distância e a variação de uma grau de latitude (mantendo-se a mesma longitude) é aproximadamente linear, variando menos de 1% ao longo das diferentes latitudes. Por outro lado, a relação entre a distância e a variação de um grau de longitude (mantendo-se a mesma latitude) é não linear, como pode ser observado na Tabela 4.1 que contém valores extraídos da "*Latitude/Longitude Distance Calculator*" (Calculadora de distância Latitude/Longitude) disponibilizada pelo "*National Hurricane Center*" dos Estados Unidos da América [24]. Isto ocorre pois a latitude divide o planeta em "círculos" paralelos ao equador que por sua vez são divididos em diferentes longitudes (vide Figura 4.3), como para cada valor de latitude os "círculos" tem raios diferentes, as distâncias referentes a variações de longitude também se alteram, como está exemplificado na Tabela 4.1

Tabela 4.1 – Valores de distância aproximada para uma variação de $\Delta = 1º$ de longitude.

Latitude	Distância (km)
0º	111
45º	79
89º	2
90º	0

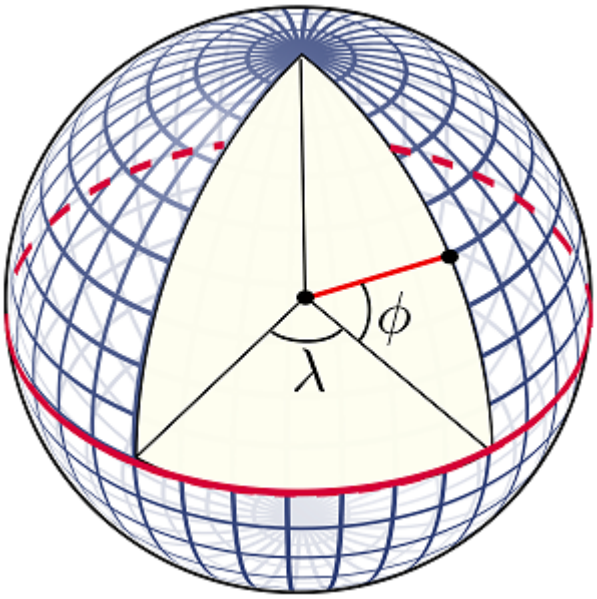

Figura 4.3 Representação de Latitude(Φ) e Longitude(λ) em uma esfera.

Para determinar se a posição atual encontra-se dentro da área de interesse, é necessário determinar um delta de latitude e outro de longitude que sejam equivalentes à variação máxima permitida (200 metros). Convertendo estes valores para o mesmo formato das variáveis "**latUtil**" e "**loUtil**", podemos então compará-los às diferenças entre os valores de "**latUtil**" nos objetos "**gps** e "**gfence**" e entre "**loUtil**" nos mesmos objetos. Para a latitude, que apresenta uma relação aproximadamente linear com a distância, optou-se se por um delta fixo ($\Delta = 1091 \rightarrow$ ~200m). Por outro lado, para a longitude adotou-se um delta variável, dependente da atual latitude. Isto foi implementado calculando-se a distância entre graus de longitude para cada grau de latitude (utilizando a calculadora já mencionada), encontrando assim, aproximadamente, variações de longitude equivalentes a 200 metros. Estes valores foram então armazenados em um vetor de inteiros, de modo que a diferença entre os valores de longitude presentes nos objetos "**gps"** e "**gfence**" seja comparada ao valor correspondente armazenado neste vetor, de acordo com a atual latitude.

A implementação da "*geofence*" implica também um tratamento de eventos, especificamente algum tipo de aviso quando a área de interesse é ultrapassada, neste caso gera-se uma mensagem contendo este aviso e as informações atuais de posição, data, bateria, etc. Esta mensagem é então repassada para a classe "**GsmRT**" através do método **sendSMS** que se encarrega de enviar a mensagem utilizando o modem

GSM/GPRS. Para fins de testes (ao menos inicialmente) é gerada e enviada também uma mensagem avisando que a área de interesse foi delimitada.

Capítulo 5

Testes e Calibração

5.1 – Testes de Carga Conversor DC-DC 5V

Os testes de carga foram realizados utilizando resistores de potência de 10Ω associados de forma a gerar cargas de 20Ω (250mA), 10Ω (500mA) e 5Ω (1A), a fonte foi alimentada com 8V e um osciloscópio Tektronix TDS 1012C-EDU foi utilizado para medir a tensão sobre os resistores. A Figura 5.1 apresenta a imagem da tela do osciloscópio com o acoplamento na posição "terra" e serve apenas de referência.

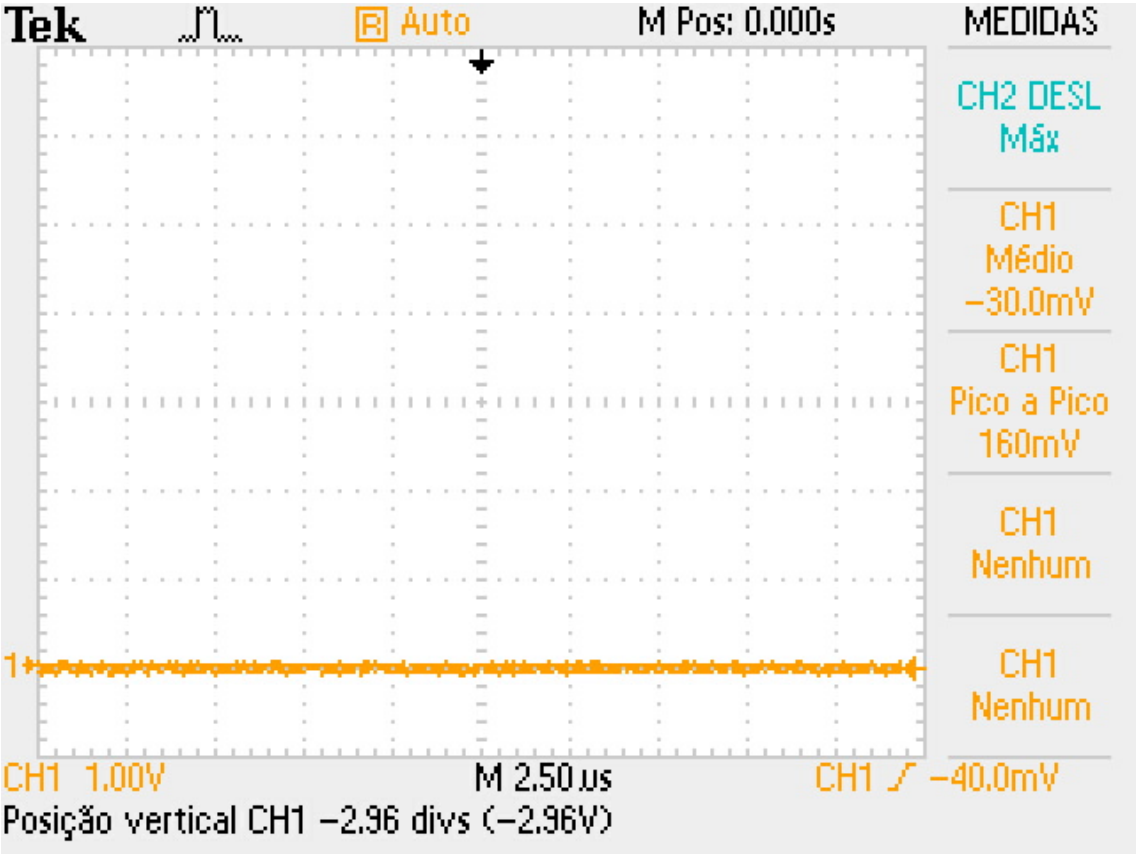

Figura 5.1 – Acoplamento "terra"

A Figura 5.2 apresenta a imagem da tela do osciloscópio com o acoplamento na posição “DC” da fonte sem carga. Observa-se que a tensão está um pouco acima da esperada e a tensão pico-a-pico é relativamente elevada.

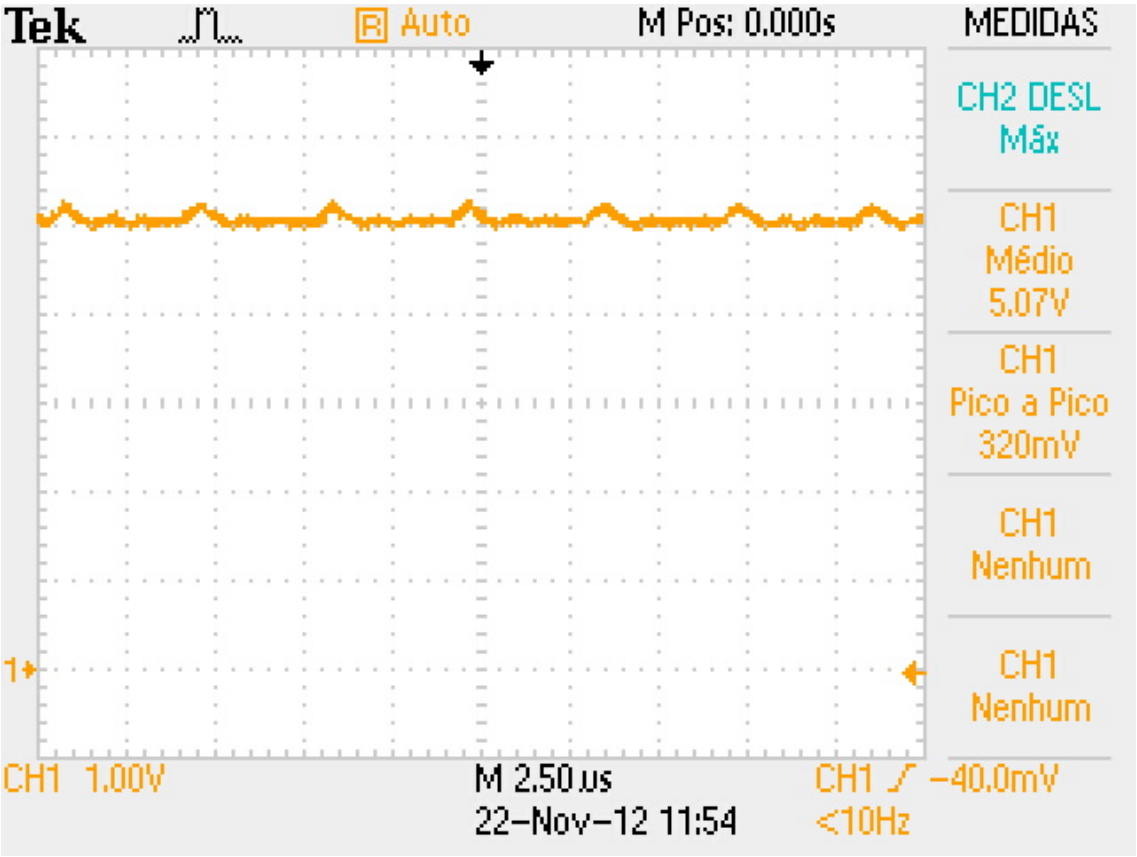

Figura 5.2 – Acoplamento “DC” sem carga

A Figura 5.3 apresenta o mesmo teste agora com o acoplamento na posição “AC” e a escala adequada. Pode-se observar claramente o chaveamento e picos de aproximadamente 150mv.

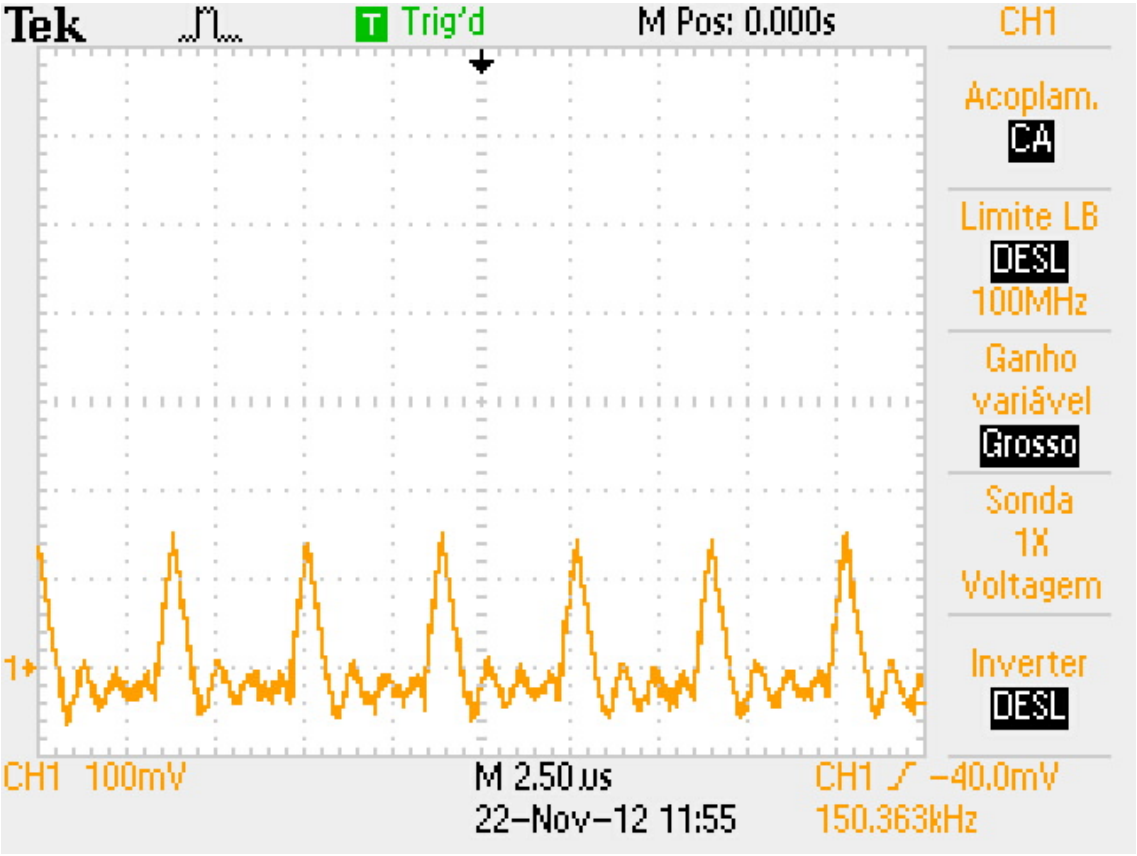

Figura 5.3 – Acoplamento "AC" sem carga

As Figuras 5.4 e 5.5 referem-se ao teste de carga de 20Ω (250mA), com acoplamentos "DC" e "AC" respectivamente. Na Figura 5.4 observa-se uma tensão média de 4.88V e uma tensão de pico-a-pico 560mV. Mesmo na escala utilizada nesta Figura está claro o elevado *ripple* presente para esta carga. Na Figura 5.5 o *ripple* fica ainda mais claro e podem ser observados picos de até 250mV.

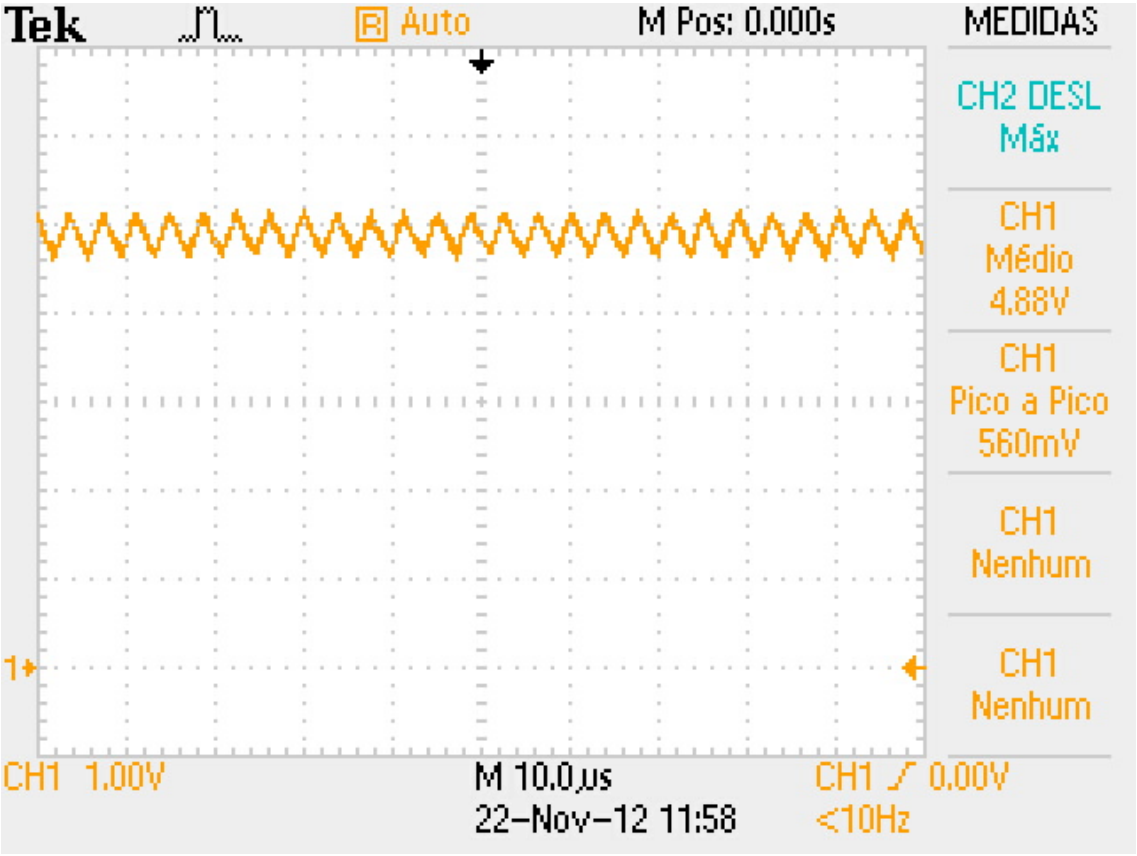

Figura 5.4 – Acoplamento "DC" carga 20Ω (250mA)

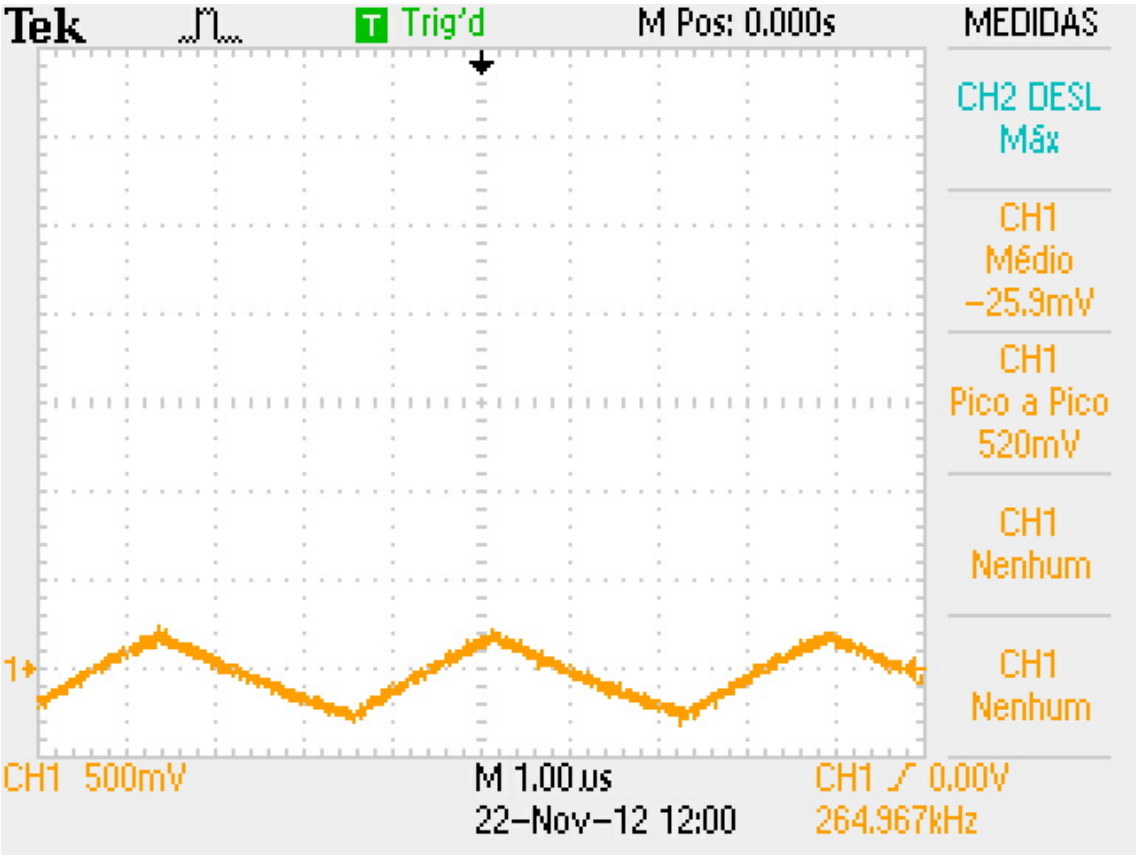

Figura 5.5 – Acoplamento "AC" carga 20Ω (250mA)

Dobrando a carga (500mA) a tensão média caiu minimamente (4.86V) e a tensão de pico a pico se manteve, como pode ser observado nas Figura 5.6 (acoplamento “DC”) e 5.7 (acoplamento “AC”).

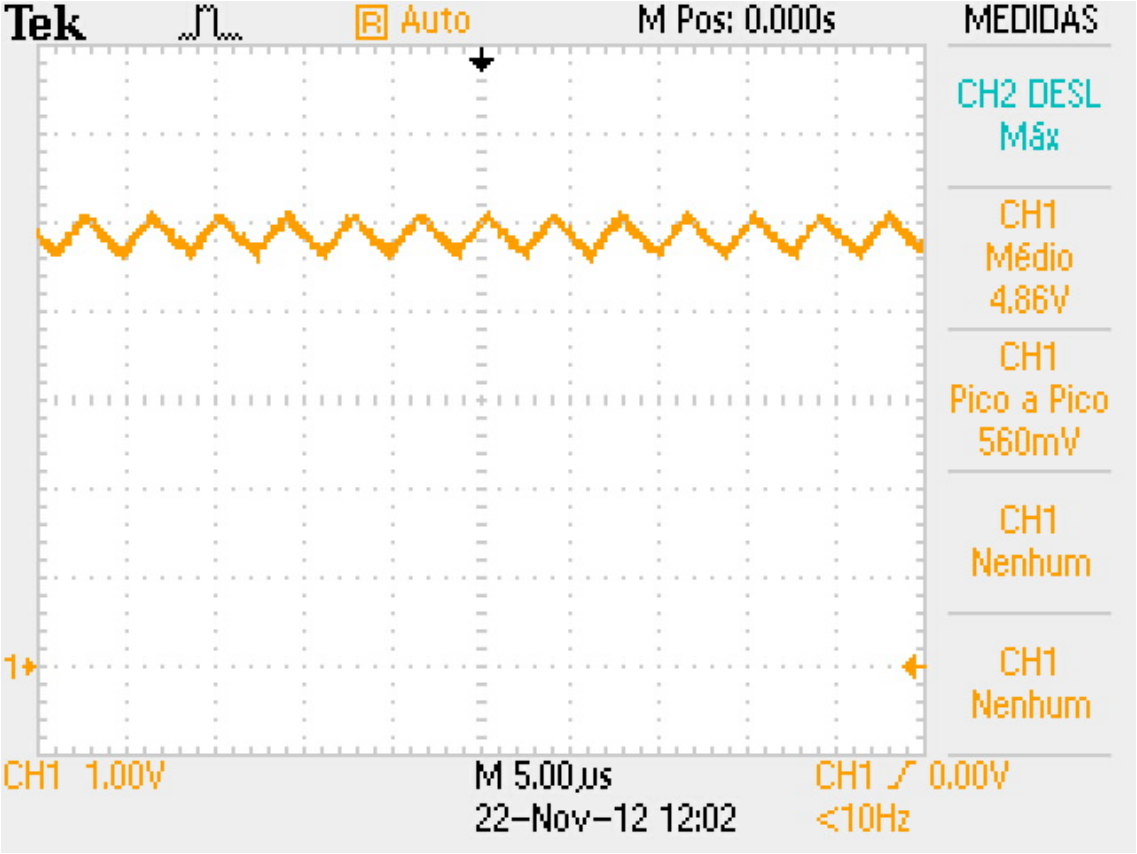

Figura 5.6 – Acoplamento “DC” carga 10Ω (500mA)

Dobrando novamente a carga (1A) observa-se comportamento similar (Figura 5.8), a tensão média diminui (4.73V), desta vez significativamente, e a tensão de pico a pico diminui minimamente (520mV). A Figura 5.9 refere-se ao mesmo teste, desta vez utilizando o acoplamento “AC”. Percebe-se novamente um elevado *ripple*, porém os picos são claramente menores do que os 250mV encontrados nos testes anteriores.

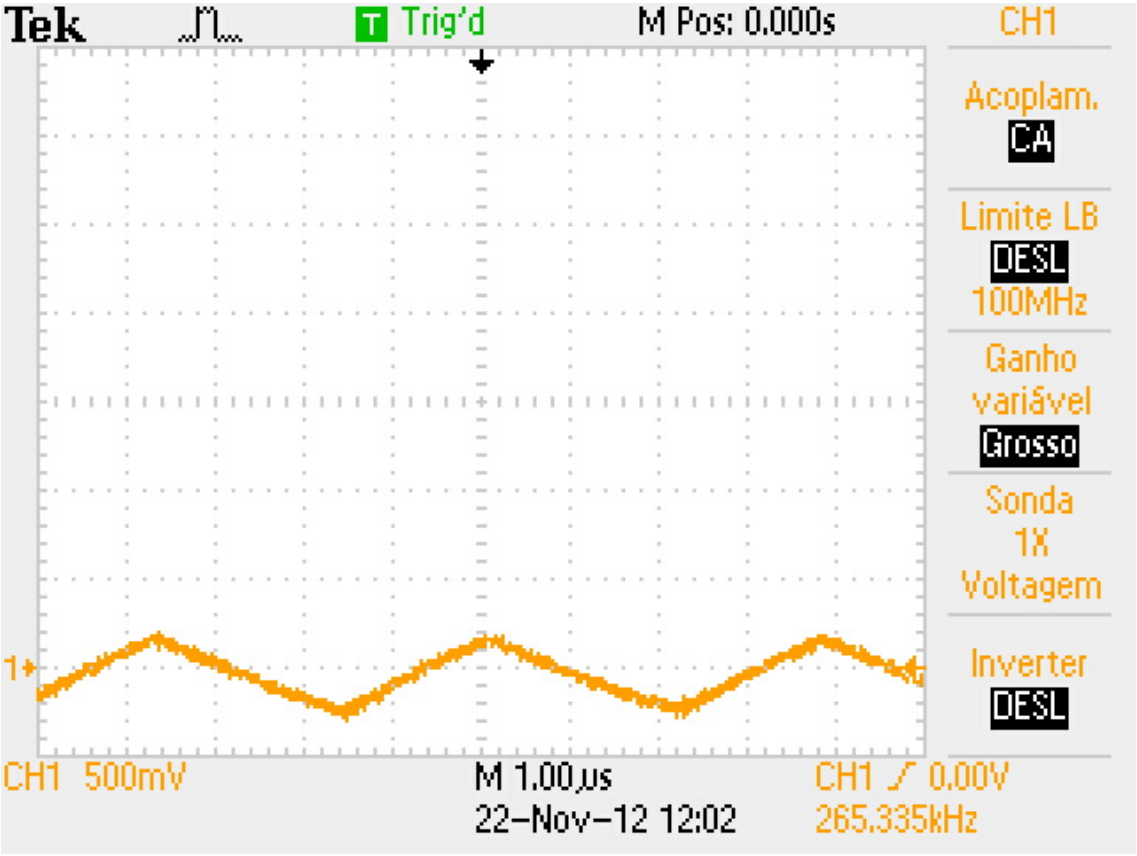

Figura 5.7 – Acoplamento "AC" carga 10Ω (500mA)

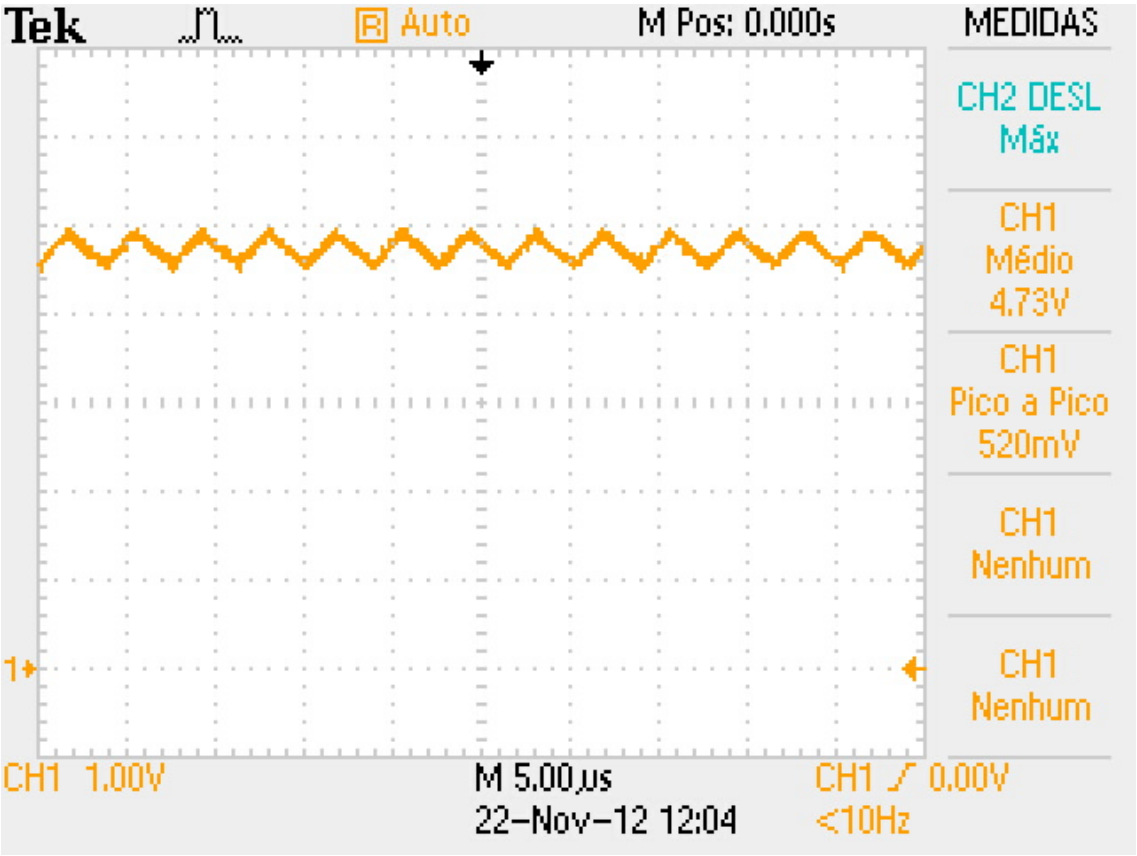

Figura 5.8 – Acoplamento "DC" carga 5Ω (1A)

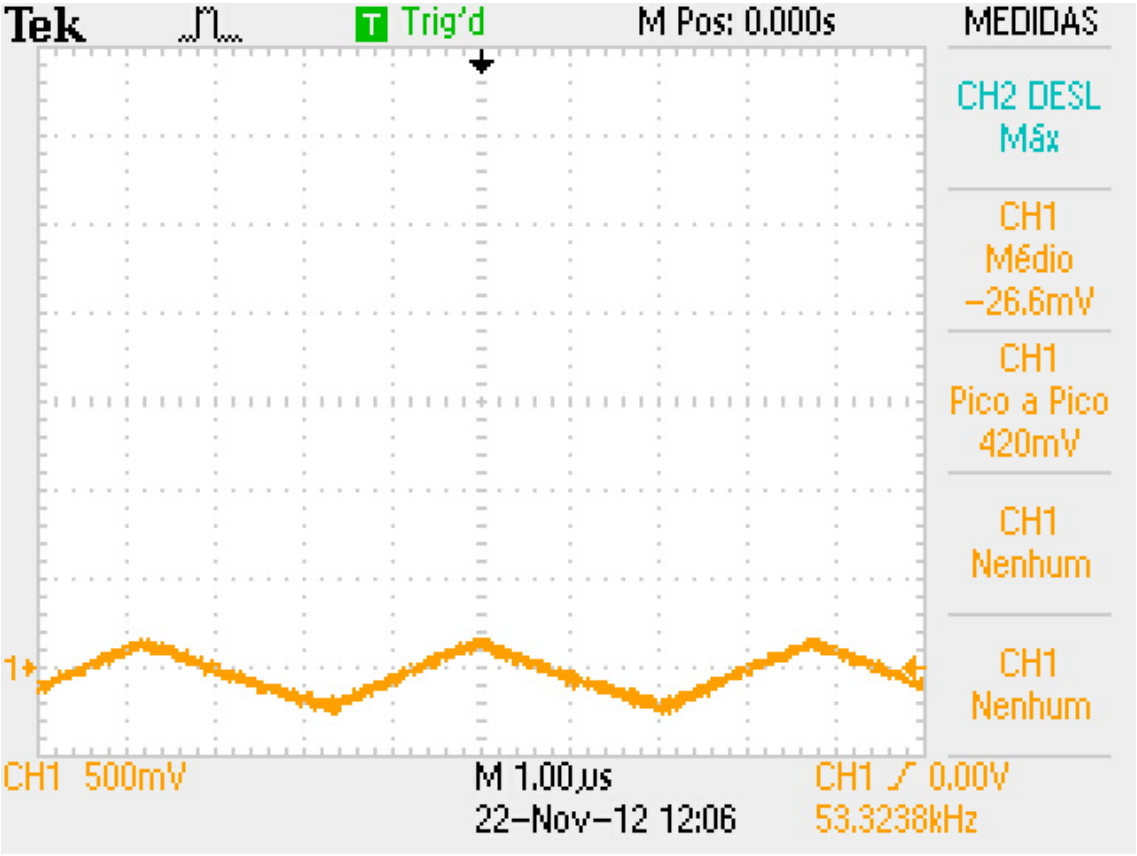

Figura 5.9 – Acoplamento "AC" carga 5Ω (1A)

Nos 3 testes de carga realizados a fonte apresentou tensão média abaixo da desejada e com o aumento da carga apresentou uma queda da tensão média. Além disso a tensão pico a pico observada se manteve significativamente elevada, mesmo diminuindo no último teste de carga realizado.

5.2 – Calibração dos Sensores

A calibração dos sensores consistiu em múltiplas leituras de cada medida registrando também valores máximos e mínimos. A partir dos dados obtidos foram geradas retas de calibração utilizando regressão linear.

5.2.1 – Tensão da Bateria

O protótipo foi alimentado com tensões variando entre 7,9V (tensão mínima de alimentação da fonte) e 15V (tensão máxima esperada da bateria), os dados foram então adicionados a uma planilha, gerando-se uma reta de calibração que é apresentada na Figura 5.10.

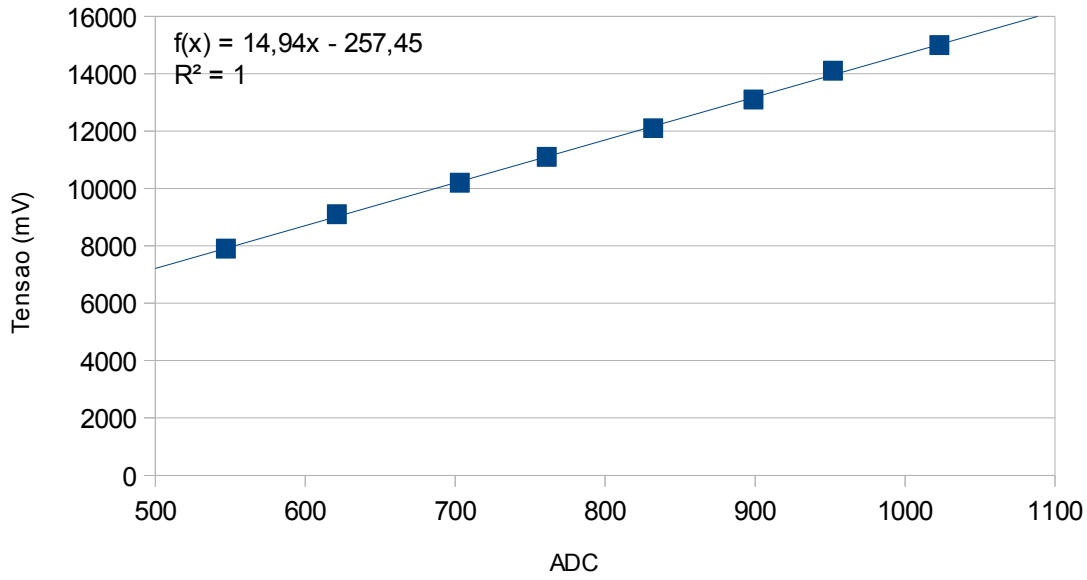

Figura 5.10 – Reta de calibração: Tensão da bateria em função da leitura do ADC

5.2.1 – Corrente entre painel e carregador

O sensor de corrente foi colocado em série com dois resistores de potência de 10Ω em paralelo, e a tensão sobre a carga foi variada entre 0V e 4,91V. A partir dos dados obtidos foi gerada um reta de calibração apresentada na Figura 5.11.

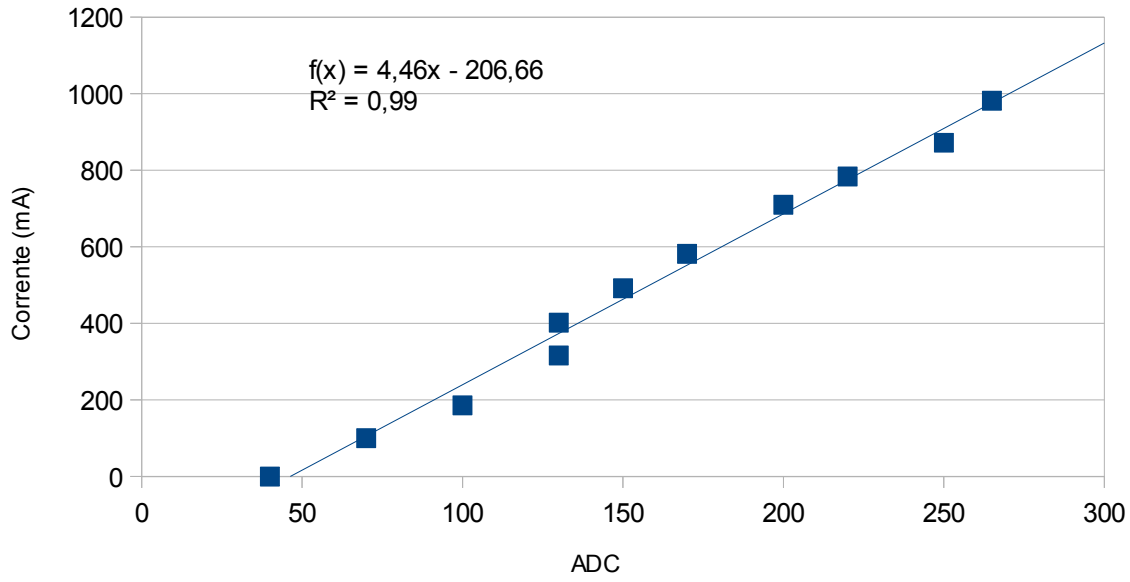

Figura 5.11 – Reta de calibração: Corrente entre o painel e o carregador em função da leitura do ADC.

Durante os testes o sensor de corrente apresentou um elevado nível de ruído, isto foi omitido da Figura 5.11, mas pode ser visualizado na Figura 5.12 onde as barras de erros representam os valores máximos e mínimos obtidos para cada leitura.

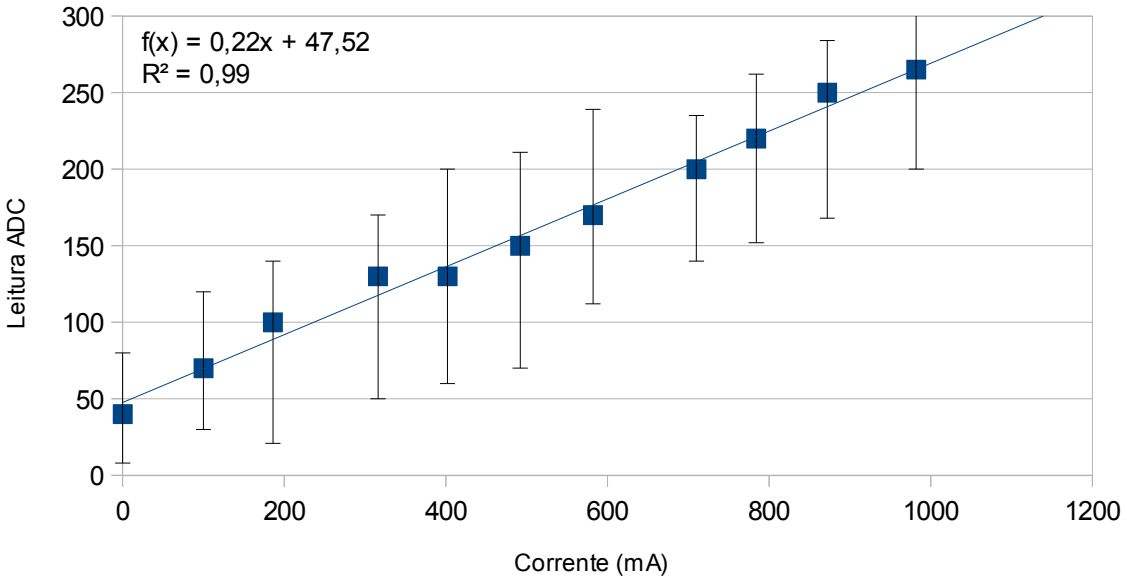

Figura 5.12 – Leitura do ADC em função da corrente percorrendo o sensor de corrente.

5.2 – Teste de funcionalidades

As funcionalidades foram testadas no Parque Brigadeiro Eduardo Gomes localizado em Botafogo na cidade do Rio de Janeiro. O teste foi realizado sem o painel solar ou o carregador, sendo o protótipo alimentado por uma bateria alcalina de 9V (o que já permitiu testar a funcionalidade de alerta do estado da bateria) e, utilizando um bicicleta para facilitar a locomoção, verificou-se sua capacidade de delimitar uma *geofence*. O procedimento adotado para o teste consistiu em 2 simples passos:

1 → Ativar o protótipo e esperar a recepção de uma mensagem SMS indicando que a cerca foi delimitada. O protótipo também ativa um LED quando isso ocorre.

2 → Uma vez delimitada a cerca, percorrer aproximadamente 200 metros (utilizaram-se como guias as marcações de distância presentes na ciclovia) até ultrapassar a área delimitada. Da mesma maneira que acima espera-se a recepção de uma mensagem SMS indicando que a os limites da cerca foram ultrapassados e também que o protótipo ative um outro LED.

Durante a realização dos testes espera-se também que ocorra a recepção de mensagens alertando sobre o estado da bateria. Não esperam-se, porém, mensagens alertando sobre o estado do carregador pois o teste do estado da bateria tem precedência e a bateria de 9V utilizada nunca apresenta o limiar de tensão mínimo esperado de uma bateria saudável.

As Figura 5.12, 5.13, 5.14 e 5.15 mostram as três mensagens recebidas em um telefone celular para cada repetição do procedimento acima.

Figura 5.13 Mensagens – 1

Figura 5.14 Mensagens – 2

A primeira mensagem é sempre a de alerta do estado da bateria e, assim como esperado (o protótipo espera a estabilização dos valores de latitude e longitude antes de delimitar a *geofence*), somente é enviada quando protótipo recebe a sua primeira posição válida. A segunda mensagem indica que a área de interesse foi delimitada e finalmente a última mensagem indica que esta área foi ultrapassada.

Figura 5.15 Mensagens – 4

Figura 5.16 Mensagens – 5

Para fins ilustrativos as coordenadas recebidas nas mensagens relativas à *geofence* foram inseridas no aplicativo web de mapas do Google. Uma vez marcados os pontos foram traçadas linhas ligando os pontos centrais das áreas delimitadas aos pontos em que as áreas foram ultrapassadas. Utilizando o link abaixo é possível acessar o mapa gerado, identificando os pontos iniciais e finais de cada teste, além da distância estimada, que em todos os casos foi muito próxima a 200m. Na Figura 5.16 apresenta-se uma imagem deste mapa gerado.

https://maps.google.com/maps/ms?msid=201392639139015259059.0004ceef66f231b0e74af&msa=0&ll=-22.945807,-43.17701&spn=0.008635,0.016512

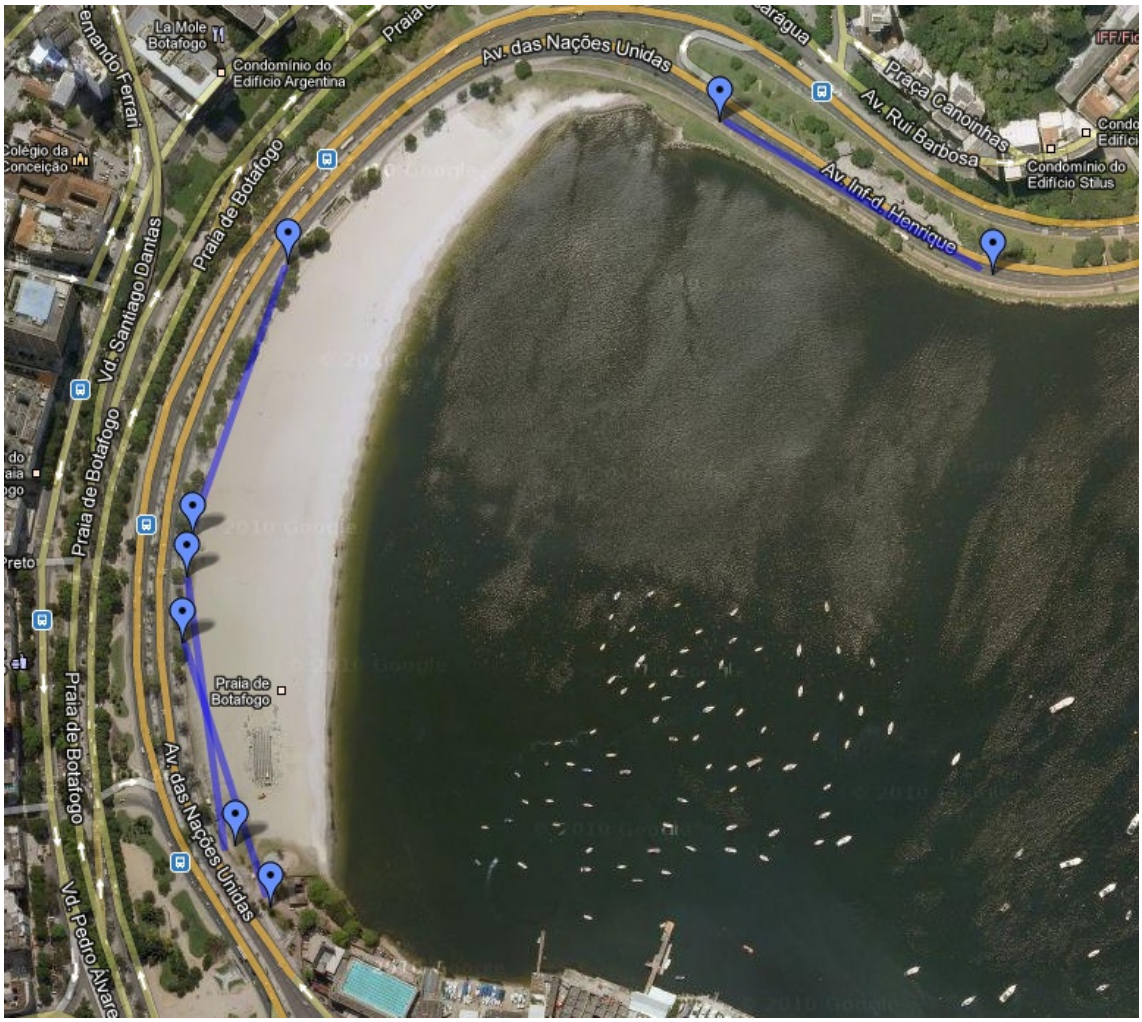

Figura 5.17 Mapa com as coordenadas recebidas durante o teste (adaptado de imagem do Google).

Capítulo 6

Conclusões

Os resultados apresentados no capítulo anterior mostram que não só o protótipo é capaz de rastrear a sua posição, como também é capaz de estabelecer uma região de interesse (centralizada no ponto aonde ele foi ativado) e detectar que tal região foi ultrapassada, atendendo assim o primeiro objetivo que é o da criação de uma *geofence*. Além disso o protótipo se utilizou de uma rede de telefonia celular para transmitir mensagens contendo informações sobre a situação da *geofence*, como também do estado da bateria, atendendo os objetivos 2 e 3.

Assim como o fez sobre o estado da bateria, o protótipo também é capaz de enviar mensagens referentes ao estado do carregador. Como já mencionado anteriormente, o estado do carregador é inferido indiretamente e a eficácia (ou ineficácia) de tal inferência ainda deverá ser estabelecida. Um aspecto que os testes deixaram claro é que o sensor de corrente selecionado não é ideal para esta aplicação. Foi considerada uma corrente máxima de 5A entre o carregador e painel solar, porém o protótipo em si raramente utilizará 1A o que representa menos de 2% da escala do sensor. Apesar disso como o sensor está sendo utilizado de forma discreta, ou seja, deseja-se apenas determinar se há ou não corrente percorrendo-o, sua utilização será válida desde que ele se mostre adequado em situações reais.

O conversor DC-DC utilizado no protótipo apresentou uma grande tensão pico a pico em todos os testes de carga e não foi capaz de manter a tensão média (5V) desejada. Segundo [25], o uso de indutores com valores menores aos recomendados pelo *datasheet* causam maiores tensões de *ripple* na saída da fonte. O indutor utilizado foi encomendando de acordo com os valores sugeridos, porém como trata-se de um indutor manualmente construído esta pode ser uma das causas do elevado *ripple* encontrado. Apesar dos problemas encontrados, o conversor DC-DC foi capaz de manter uma tensão relativamente próxima (>4.7V) aos 5V, atuando dentro da faixa de operação de todos os componentes digitais. Por outro lado o ruído e a variação na

tensão média, de acordo com a carga, são inadequados para os componentes analógicos, afetando a utilização do ADC do microcontrolador.

6.1 – Trabalhos Futuros

A partir dos resultados obtidos neste trabalho apresentam-se algumas vertentes para desenvolvimentos futuros:

→ Substituir ou reprojetar o conversor DC-DC utilizado.

→ Expansão das funcionalidades do protótipo atual adicionando, por exemplo, funcionalidades como a capacidade de configuração remota, obtenção e transmissão do estado de instrumentos científicos, novos sensores, etc.

→ Implementação de maneira mais definitiva do protótipo, utilizando uma placa desenvolvida especificamente para o protótipo, com apenas os componentes necessários, possivelmente substituindo o sensor de corrente.

→ Modificar o *firmware* de modo a minimizar a utilização de memória *SRAM*, e o protótipo de modo que possa se utilizar a mesma *USART* para se comunicar com o receptor GPS e o modem GSM/GPRS, permitindo assim a utilização um microcontrolador mais simples e barato (ex.: Atmega328P).

→ Reimplementar o protótipo utilizando dois ou mais microcontroladores mais simples, separando as funcionalidades entre eles, e possivelmente expandindo as funcionalidades do protótipo atual.

→ Avaliar uma nova implementação da *geofence,* agora determinando os valores de latitude e longitude referentes aos extremos da área de interesse, permitindo a comparação direta de valores de latitude e longitude, para determinar se o protótipo se encontra dentro da área.

Referências

[1] U.S COAST GUARD, "U.S. COAST GUARD NAVIGATION CENTER – General information on GPS" http://www.navcen.uscg.gov/?pageName=GPSmain (Acesso em 03 dezembro 2012).

[2] U.S. COAST GUARD, "NAVSTAR GPS USER EQUIPMENT INTRODUCTION" www.navcen.uscg.gov/pubs/gps/gpsuser/gpsuser.pdf, 1996 (Acesso em 31 Janeiro 2013).

[3] WIKIPEDIA, "Global Positioning System" http://en.wikipedia.org/wiki/GPS (Acesso em 03 dezembro 2012).

[4] LANGLEY, R.B, "Dilution of Precision", *GPS World*, pp 52 – 59 Maio 1999.

[5] WIKIPEDIA, "NMEA 0183" http://en.wikipedia.org/wiki/NMEA_0183 (Acesso em 07 dezembro 2012).

[6] DEPRIEST, D., "NMEA Data", http://www.gpsinformation.org/dale/nmea.htm#intro (Acesso em 04 dezembro 2012).

[7] PERSON, J., "Calculating and Validating NMEA checksums" http://codepedia.com/1/calculating+and+validating+NMEA+Checksums (Acesso em 31 Janeiro 2013).

[8] KOLFF COMPUTER SUPPLIES, "Welcome to KCS TraceME", http://www.traceme.eu/?page=home (Acesso em 04 dezembro 2012).

[9] ARDUINO, "Arduino – Introduction",http://www.arduino.cc/en/Guide/Introduction (Acesso em 04 dezembro 2012).

[10] ARDUINO, "Arduino – ArduinoBoardUno", http://arduino.cc/en/Main/ArduinoBoardUno (Acesso em 04 dezembro 2012).

[11] ARDUINO, "Arduino – ArduinoBoardMEGA2560", http://arduino.cc/en/Main/ArduinoBoardMega2560 (Acesso em 04 dezembro 2012).

[12] ATMEL, "Atmel 8-bit Microcontroller with 4/8/16/36KBytes In-System Programmable Flash", http://www.atmel.com/Images/doc8271.pdf , 2012 (Acesso em 31 de Janeiro 2013).

[13] ATMEL, "8-bit Atmel Microcontroller with 64/128/256KBytes In-System Programmable Flash", http://www.atmel.com/Images/2549S.pdf , 2012 (Acesso em 31 de Janeiro 2013).

[14] SKYLAB M&C TECHNOLOGY CO. LTD, " SkyNav SKM53 GPS Module Datasheet" , http://file01.up71.com/File/CorpDownFile/2012/08/18/0_skylab_20120818151614.pdf , 2010 (Acesso em 31 Janeiro 2013).

[15] SKYTRAQ TECHNOLOGY INC, "VENUS634FLPx 65 Channel Low Power GPS Receiver – Flash", http://www.sparkfun.com/datasheets/GPS/Modules/Skytraq-Venus634FLPx_DS_v051.pdf , 2009 (Acesso em 05 dezembro 2012).

[16] SEEDSTUDIO, "Seedstudio GPRS Shield – Wiki ", http://www.seeedstudio.-com/wiki/GPRS_Shield (Acesso em 07 novembro 2012).

[17] SIMCOM, "SIM900_AT Command Manual_ v1.03 ", http://garden.seeedstudio.com/images/a/a8/SIM900_AT_Command_Manual_V1.03.pdf , 2010 (Acesso em 31 Janeiro 2013)

[18] PERÉZ-FONTAN, H., *Introduction to Mobile Communications Engineering.* Massachusetts, Artech House, 1999.

[19] ALLEGRO MICROSYSTEMS INC, "ACS756 – Fully Integrated, Hall Effect0-Based Linear Current Sensor IC with 3kVRMS Voltage Isolation and a Low-Resistance Current Conductor", http://www.allegromicro.com/~/media/Files/Datasheets/ACS756-Datasheet.ashx , Massachusetts 2011 (Acesso em 31 Janeiro 2013).

[20] TEXAS INSTRUMENTS INCORPORATED, "TLV271, TLV272, TLV274 Family of 550-uA/Ch-3Mhz Rail-to-Rail Output Operational Amplifiers" http://www.ti.com/litv/pdf/slos351d , Texas 2004, (Acesso em 31 Janeiro 2013).

[21] TEXAS INSTRUMENTS INCORPORATED, "LM2676 SIMPLE SWITCHER High Efficeicy 3A Step-Down Voltage Regulator" , http://www.ti.com/litv/pdf/snvs031i , Texas 2012 (Acesso em 31 Janeiro 2013).

[22]SAILSOFT, "GpsSimul from Sailsoft", http://www.sailsoft.nl/gpssimul.htm (Acesso em 06 dezembro 2012).

[23] HART, M., "PString | Arduiniana ", http://arduiniana.org/libraries/pstring/ (Acesso em 05 dezembro 2012).

[24] NATIONAL HURRICANE CENTER, "Latitude/Longitude Distance Calculator", http://www.nhc.noaa.gov/gccalc.shtml (Acesso em 05 dezembro 2012).

[25] NATIONAL SEMICONDUCTOR, "Answers/FAQs – Power", http://www.nation-al.com/kbase/category/Power.html (Acesso em 06 dezembro 2012).

Apêndice A

A1 – Código Fonte Firmware Atmega2560

```
#include <GpsDataSKM53.h>
#include <PString.h>
#include <GsmRT.h>

char inData[81]; //Tamanho máximo de sentença fornecida pelo SKM53
PString inDataString(inData,sizeof(inData));

char inData1[81];
PString inDataString1(inData1,sizeof(inData1));

char inChar;

long timeout;

GpsDataSKM53 gps;
GpsDataSKM53 gFence;

GsmRT celular;
char numero[11] = "88954692";

boolean ok;
boolean fenceState;
boolean chargerOk = true;
boolean batteryOk = true;
boolean dailyMsgOk = true;

boolean configPin1 = 52;
boolean pin1State;
boolean configPin2 = 50;
boolean pin2State;
boolean configPin3 = 48;
boolean pin3State;
boolean configPin4 = 46;
boolean pin4State;


byte led4State = LOW;
byte led3State = LOW;
byte led2State = LOW;

int settled;

////Delay for sending SMS abou geofence////
int smsDelay1;
////////////////////////////////////////////////////////////////////////////////

////Delay for sensing SMS about battery and charger////
int smsDelay2;
////////////////////////////////////////////////////////////////////////////////

// Time interval between sms messages (in seconds) //////
int smsTimeLimit = 3600;
// Time of day to send daily message (hhmmss)/////////////////////////////////
long dailyMsgTime = 120000;
////////////////////////////////////////////////////////////////////////////////
```

```
float batteryVolts;
float chargerAmps;

void setup()
{
  Serial.begin(19200);  //Modem GSM/GPRS
  Serial3.begin(9600);  //GPS

  ////Configura SKM53 para somente enviar sentenças GSA e GGA/////////
  Serial3.println("$PMTK314,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0*28");
  ////////////////////////////////////////////////////////////////////

////LED pins ////////////////////////////////////////////////////////////////
  pinMode(4,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(2,OUTPUT);
//////////////////////////////////////////////////////////////////////////////

////DIP Switch pins, setup and read state/////////////////////////////////////
  pinMode(configPin1,INPUT);
  pin1State = digitalRead(configPin1);
  pinMode(configPin2,INPUT);
  pin2State = digitalRead(configPin2);
  pinMode(configPin3,INPUT);
  pin3State = digitalRead(configPin3);
  pinMode(configPin4,INPUT);
  pin4State = digitalRead(configPin4);
//////////////////////////////////////////////////////////////////////////////

  celular.onOff();
  celular.setPhoneNumber('1',numero);

  smsDelay1 = smsTimeLimit + 1;
  smsDelay2 = smsTimeLimit + 1;

  ok = false;
}


void loop(){
  //timeout = 0;
  ok = false;
  inDataString.begin();

///Sincronização da entrada serial do GPS ////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
  while(ok == false/*timeout < 14000*/)
  {
    if(Serial3.available() > 0)
    {
      inChar = Serial3.read();
      if(inChar == '$')
      {
        inDataString += inChar;
        //timeout = 14000;
        ok = true;
      }
    }
    //timeout++;
    //Serial.println(timeout);
  }
//////////////////////////////////////////////////////////////////////////////
```

```
  //timeout = 0;
  ok = false;

///Recebimento das setenças/////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
  while(ok == false/*timeout < 14000*/)
  {
    if(Serial3.available() > 0)
    {
      inChar = Serial3.read();
      if(inChar == '\n')
      {
        //timeout = 14000;
        ok = true;
      } else inDataString +=inChar;
    }
    //timeout++;
  }


  ok = false;
  inDataString1.begin();

  while(ok == false/*timeout < 14000*/)
  {
    if(Serial3.available() > 0)
    {
      inChar = Serial3.read();
      if(inChar == '\n')
      {
        //timeout = 14000;
        ok = true;
      } else inDataString1 +=inChar;
    }
    //timeout++;
  }
////////////////////////////////////////////////////////////////////////////////


///Switch que trata as diferentes sentenças/////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
  timeout = 0;
  while(timeout < 2)
  {
    //Serial.println(inData);
    //Serial.println(timeout);
    switch(gps.getNmeaType(inData)) {
      case 1://Sentença GSA
          //Serial.println("Sentenca: GSA");
          digitalWrite(4,HIGH);
          gps.chkFixGSA(inData, inDataString.length());
          if(gFence.getFixState() == false)
          {
            if(led4State = LOW)
            {
              led4State = HIGH;
            } else led4State = LOW;
            digitalWrite(4,led3State);
            gFence.chkFixGSA(inData, inDataString.length());
            if(gps.getFixState() == true)//if(gFence.getFixState() == true)
            {
              //Serial.println("Fix Ok");
```

```
              digitalWrite(4,HIGH);
              fenceState = true; //Necessário????
              settled = 0;
            }
          }

          //Serial.println(inData);
          //Serial.println(" ");
          break;

      case 2://Sentença GGA
          //Serial.println("Sentenca : GGA");

//fenceState == true indica que a posição atual deve ser guardada como referência
da geofence/
          if(fenceState == true)
          {
            settled++;

            if(settled > 70)
            {
              gFence.readGGA(inData);//obtem-se a posição central da geofence
              gFence.setFixState(true);
              fenceState = false;
              digitalWrite(3,HIGH);
              gerarMsgSMS(1);
              smsDelay1 = smsTimeLimit;
            }
           }

          if(gps.getFixState() == true)
          {
            gps.readGGA(inData);
            if(fenceState == false)
            {
                if(gFence.chkGeoFenceOk(gps.getLatUtil(),gps.getLoUtil()) ==
false)
                {
                  digitalWrite(2,HIGH);
                  gerarMsgSMS(2);
                } else digitalWrite(2,LOW);
            }
            readSensors();

//Daily message and it`s allowed time interval////////////////////////////////
////////////////////////////////////////////////////////////////////////////
            if((gps.getTime() > dailyMsgTime) && (gps.getTime() < (dailyMsgTime +
500)))
            {
              if(dailyMsgOk == false)
              {
                gerarMsgSMS(5);
                dailyMsgOk = true;
              }
            }
            if(gps.getTime() > dailyMsgTime + 500)
            {
              dailyMsgOk = false;
            }
////////////////////////////////////////////////////////////////////////////
          break;
      default:
```

```
    }

///clears inData and then copies inData1 string to inData//////////////////////
////////////////////////////////////////////////////////////////////////////////
    inDataString.begin();
    inDataString.print(inData1);
////////////////////////////////////////////////////////////////////////////////

    if(batteryOk == false)
    {
        gerarMsgSMS(4);
    }
    if(chargerOk == false)
    {
        gerarMsgSMS(3);
    }
    smsDelay1++;
    smsDelay2++;
    timeout++;
  }
}



////Geração e envio de SMS no formato:////////////////////////////////////////
/*Mensagem -- Lat:gg graus mm.mmmm minutos -- Long:gg graus mm.mmmm --
Hora:hhmmss
opt: 1-> Cerca Delimitada, 2-> Cerca Ultrapassada, 3-> Falha Carregador, 4->
Falha Bateria 5-> Mensagem Diaria */
////////////////////////////////////////////////////////////////////////////////
void gerarMsgSMS(int opt){

      char smsBuffer[141];
      PString smsString(smsBuffer,sizeof(smsBuffer));
      smsString.begin();
      boolean smsOk1 = false;
      boolean smsOk2 = false;
      boolean smsOk3 = false;


      switch(opt) {
        case 1:
            smsString.print("Cerca Delimitada -- Lat:");
            if(smsDelay1 > smsTimeLimit)
            {
                smsOk1 = true;
            }
            break;
        case 2:
            smsString.print("Cerca Ultrapassada -- Lat:");
            if(smsDelay1 > smsTimeLimit)
            {
                smsOk1 = true;
            }
            break;
        case 3:
            smsString.print("Falha do Carregador -- Lat:");
            if(smsDelay2 > smsTimeLimit)
            {
                smsOk2 = true;
            }
            break;
```

```
        case 4:
            smsString.print("Falha da Bateria -- Lat:");
            if(smsDelay2 > smsTimeLimit)
            {
                smsOk2 = true;
            }
            break;
        case 5:
            smsString.print("Mensagem Diaria -- Lat:");
            smsOk3 = true;
            break;
        default:
            break;
      }
      smsString.print(gps.getLaDeg());
      smsString.print(" graus ");
      smsString.print(gps.getLaMin());
      smsString.print(".");
      smsString.print(gps.getLaSec());
      smsString.print(" minutos");
      smsString.print("--Long:");
      smsString.print(gps.getLoDeg());
      smsString.print(" graus ");
      smsString.print(gps.getLoMin());
      smsString.print(".");
      smsString.print(gps.getLoSec());
      smsString.print("--Hora:");
      smsString.print(gps.getTime());
      smsString.print("--Bateria:");
      smsString.print(batteryVolts);
      smsString.print("mV");
      smsString.print("--Carregador:");
      smsString.print(chargerAmps);
      smsString.print("mA");

//If a message hasn't been sent in the last "smsTimeLimit" seconds sends a SMS///
////////////////////////////////////////////////////////////////////////////////
      if(smsOk1 == true)
      {
        celular.sendSMS('1',smsBuffer);
        //Serial.println("Mandei SMS");
        smsDelay1 = 0;
      }

      if(smsOk2 == true)
      {
        celular.sendSMS('1',smsBuffer);
        //Serial.println("Mandei SMS");
        smsDelay2 = 0;
      }

      if(smsOk3 == true)
      {
        celular.sendSMS('1',smsBuffer);
      }
////////////////////////////////////////////////////////////////////////////////


////If configInput is active send message to Serial port 2//////////////////////
////////////////////////////////////////////////////////////////////////////////
      if(pin1State == true)
      {
```

```
         Serial2.println(smsString);
      }
////////////////////////////////////////////////////////////////////////////////

}

////reads sensors according to opt: '1' battery, '2' current////////////////////
////////////////////////////////////////////////////////////////////////////////
void readSensors(){

  int index = 0;
  int tempInt = 0;
  boolean alerta = false;

////Read battery voltage -> output directly in mV//////////////////////////////
////////////////////////////////////////////////////////////////////////////////
  while(index < 16)
  {
    tempInt += analogRead(A0);
    index++;
  }
  tempInt = tempInt/16;
  batteryVolts = ((14.93525*tempInt) - 257.45264);
  if(batteryVolts < 11,5)
  {
     alerta = true;
  }
  if(batteryVolts < 10,6)
  {
    batteryOk = false;
  }
////////////////////////////////////////////////////////////////////////////////
  index = 0;
  tempInt = 0;
////Read current sensor output -> directly in mA///////////////////////////////
////////////////////////////////////////////////////////////////////////////////

  while(index < 16)
  {
    tempInt += analogRead(A3);
    index++;
  }
  tempInt = tempInt/16;
  chargerAmps = ((4.46336*tempInt) -206.66313);
  if(alerta == true)
  {
      if((gps.getTime()  > 100000) && (gps.getTime() < 190000))
      {
          if(chargerAmps < 200)
          {
              chargerOk = false;
          }else chargerOk = true;
      }else chargerOk = true;
  }
////////////////////////////////////////////////////////////////////////////////
}
```

A2 – Código Fonte Classe GpsDataSKM53

```
#include "Arduino.h"


#ifndef GpsDataSKM53_h
#define GpsDataSKM53_h

class GpsDataSKM53
{
  public:
    GpsDataSKM53();
    void setLatLongGLL(char* inData);
       void readGGA(char* inData);
    int getNmeaType(char* inData); //1-GSA, 2-GGA, 3-GLL,
       void chkFixGSA(char* inData,int tamanho);
       long getLaDeg();
       long getLaMin();
       long getLaSec();
       long getLoDeg();
       long getLoMin();
       long getLoSec();
       long getLoUtil();
       long getLatUtil();
       long getTime();
       int getAltitude();
       boolean getFixState();
       void setFixState(boolean fixState);
       boolean chkGeoFenceOk(long latitude,long longitude);

  private:
       int tempInt;
       char tempChar;

       char tempCharArray[10];

       long latDeg;
       long latMin;
       long latSec;
       long loDeg;
       long loMin;
       long loSec;


       long latUtil;
       long loUtil;
       long gpsUtcMinus3Time;
       int    altitude;
       boolean fix;


};
#endif
```

```
#include "Arduino.h"
#include "GpsDataSKM53.h"

GpsDataSKM53::GpsDataSKM53()
{
       latDeg = 0;
       latMin = 0;
       latSec = 0;
       loDeg = 0;
       loMin = 0;
       loSec = 0;
       fix = false;
}

void GpsDataSKM53::setLatLongGLL(char* inData){

    if(inData[15] == 'S')   // Somente necessário para o "degree"
    {
      tempCharArray[0] = '-';
    } else tempCharArray[0] = '+';

    tempCharArray[1] = inData[7];
    tempCharArray[2] = inData[8];
    tempCharArray[3] = '\0';

    latDeg = atoi(tempCharArray);

    tempCharArray[1] = inData[9];
    tempCharArray[2] = inData[10];
    tempCharArray[3] = '\0';

    latMin = atoi(tempCharArray);

    tempCharArray[1] = inData[12];
    tempCharArray[2] = inData[13];
    tempCharArray[3] = '\0';

    latSec = atoi(tempCharArray);

    latUtil = (60*1000)*latDeg + 1000*latMin + latSec;

    if(inData[28] == 'W')   // Somente necessário para o "degree"
    {
      tempCharArray[0] = '-';
    } else tempCharArray[0] = '+';

    tempCharArray[1] = inData[18];
    tempCharArray[2] = inData[19];
    tempCharArray[3] = inData[20];
    tempCharArray[4] = '\0';

    loDeg = atoi(tempCharArray);

    tempCharArray[1] = inData[21];
    tempCharArray[2] = inData[22];
```

```
    tempCharArray[3] = '\0';

    loMin = atoi(tempCharArray);

    tempCharArray[1] = inData[24];
    tempCharArray[2] = inData[25];
    tempCharArray[3] = '\0';

    loSec = atoi(tempCharArray);

    loUtil = (60*1000)*loDeg + 1000*loMin + loSec;
}

void GpsDataSKM53::readGGA(char* inData){

//$GPGGA,182211.000,2256.1530,S,04310.8633,W,2,10,0.83,-8.2,M,-5.6,M,0000,0000*56
       if(inData[43] == '0')
       {
              fix = false;
       } else {
              tempCharArray[0] = inData[7];
              tempCharArray[1] = inData[8];
              tempCharArray[2] = inData[9];
              tempCharArray[3] = inData[10];
              tempCharArray[4] = inData[11];
              tempCharArray[5] = inData[12];
              tempCharArray[6] = '\0';
////////////////////////////////////////////////////////////////////////////
//UTC time is in format hhmmss, we correct for utc-3 subtracting
//30000. If gpsUtcMinus3Time is negative it means different days
//so we correct the time by adding 240000 -> 01h -> -2h -> 22h//////////////////
              gpsUtcMinus3Time = (atol(tempCharArray) - 30000);
              if(gpsUtcMinus3Time < 0)
              {
                     gpsUtcMinus3Time = (gpsUtcMinus3Time + 240000);
              }
////////////////////////////////////////////////////////////////////////////
              if(inData[28] == 'N')
              {
                     tempCharArray[0] = '+';
              } else tempCharArray[0] = '-';

              tempCharArray[1] = inData[18];
              tempCharArray[2] = inData[19];
              tempCharArray[3] = '\0';

              latDeg = atol(tempCharArray);

              tempCharArray[1] = inData[20];
              tempCharArray[2] = inData[21];
              tempCharArray[3] = '\0';

              latMin = atol(tempCharArray);

              tempCharArray[1] = inData[23];
              tempCharArray[2] = inData[24];
              tempCharArray[3] = inData[25];
              tempCharArray[4] = inData[26];
              tempCharArray[5] = '\0';

              latSec = atol(tempCharArray);
```

```
		latUtil = 600000*latDeg + 10000*latMin + latSec;

		if(inData[41] == 'E')   // Somente necessário para o "degree"
		{
			tempCharArray[0] = '+';
		} else tempCharArray[0] = '-';

		tempCharArray[1] = inData[30];
		tempCharArray[2] = inData[31];
		tempCharArray[3] = inData[32];
		tempCharArray[4] = '\0';

		loDeg = atol(tempCharArray);

		tempCharArray[1] = inData[33];
		tempCharArray[2] = inData[34];
		tempCharArray[3] = '\0';

		loMin = atol(tempCharArray);

		tempCharArray[1] = inData[36];
		tempCharArray[2] = inData[37];
		tempCharArray[3] = inData[38];
		tempCharArray[4] = inData[39];
		tempCharArray[5] = '\0';

		loSec = atol(tempCharArray);

		loUtil = 600000*loDeg + 10000*loMin + loSec;

		tempChar = inData[52];
		tempInt = 0;
		while(tempChar != ',')
		{
			tempCharArray[tempInt] = tempChar;
			tempInt++;
			tempChar = inData[52+tempInt];
		}
		altitude = atoi(tempCharArray);
	}

}

void GpsDataSKM53::chkFixGSA(char* inData,int size){
	if(inData[9] == '1')
	{
		fix = false;
	} else {

		fix = true;

		tempCharArray[0] = inData[size-12];
		tempCharArray[1] = '\0';
		if((tempInt = atoi(tempCharArray)) >= 4)
		{
			fix = false;
		}
		tempCharArray[0] = inData[size-8];

		if((tempInt = atoi(tempCharArray)) >= 4)
		{
			fix = false;
```

```
            }
        }
}



int GpsDataSKM53::getNmeaType(char * inData){


        tempCharArray[0] = inData[3];
        tempCharArray[1] = inData[4];
        tempCharArray[2] = inData[5];
        tempCharArray[3] = '\0';

        if(strcmp(tempCharArray,"GSA") == 0)
        {
               tempInt = 1;
        } else if (strcmp(tempCharArray,"GGA") == 0){
               tempInt = 2;
        } else if (strcmp(tempCharArray,"GLL") == 0){
               tempInt = 3;
        }else tempInt = -1;

        return(tempInt);
}

boolean GpsDataSKM53::chkGeoFenceOk(long latitude,long longitude){

        boolean ok;

//A variável "longitudeDelta" se refere à variação de um delta qualquer
//de longitude em função da latitude, neste caso o vetor contém os
//os valores (aproximados) referentes a 200m em milésimos de minutos.
////////////////////////////////////////////////////////////////////////////////
       int longitudeDelta[] = {1081,1081,1081,1081,1081,1081,1081,1091,
1091,1091,1101,1101,1101,1111,1121,1121,1121,1132,1143,1154,1154,1165,
1176,1176,1188,1200,1212,1224,1237,1250,1263,1277,1290,1304,1319,1333,
1348,1364,1395,1412,1429,144,1481,1500,1519,1558,1579,1622,1644,1690,
1714,1765,1791,1846,1875,1935,1967,2034,2105,2143,2222,2308,2400,2449,
2553,2667,2791,2857,3000,3158,3333,3529,3750,3871,4138,4444,4800,5217,
5714,6316,7059,8000,8571,10000,12000,15000,20000,30000,0,0};
////////////////////////////////////////////////////////////////////////////////


    if(fix == true) //Geo-Fence já estabelecida
    {
////Verifica se a posiçao variou +- 200m(1100) da posicao referência/////////////
////Formato latUtil/loUtil: 60000*graus + 10000*minutos + milésimos de minutos///
      ok = true;
         tempInt = latitude - latUtil;
         tempInt = abs(tempInt);
         //Serial.println(tempInt);
      if(tempInt > 1091 /*longitudeDelta[abs((int)latDeg)]*/)
      {
        Serial.println("ALERTA");
        ok = false;
      }
         tempInt = longitude - loUtil;
         tempInt = abs(tempInt);
         //Serial.println(tempInt);
      if(tempInt > longitudeDelta[abs((int)latDeg)])
```

```
      {
        Serial.println("ALERTA");
        ok = false;
      }
     }
        return(ok);
}
long GpsDataSKM53::getLaDeg(){
       return(latDeg);
}

long GpsDataSKM53::getLaMin(){
       return(latMin);
}

long GpsDataSKM53::getLaSec(){
       return(latSec);
}

long GpsDataSKM53::getLoDeg(){
       return(loDeg);
}

long GpsDataSKM53::getLoMin(){
       return(loMin);
}

long GpsDataSKM53::getLoSec(){
       return(loSec);
}

long GpsDataSKM53::getLatUtil(){
       return(latUtil);
}

long GpsDataSKM53::getLoUtil(){
       return(loUtil);
}

long GpsDataSKM53::getTime(){
       return(gpsUtcMinus3Time);
}

int GpsDataSKM53::getAltitude(){
       return(altitude);
}

boolean GpsDataSKM53::getFixState(){
       return(fix);
}


void GpsDataSKM53::setFixState(boolean fixState){
       fix = fixState;
}
```

A3 – Código Fonte Firmware Atmega2560

```
#include "Arduino.h"

#ifndef GsmRT_h
#define GsmRT_h

class GsmRT
{
  public:
    GsmRT();
       void onOff();
       void off();
       void setPhoneNumber(char opt,char* number); //opt define qual o número de
telefone
       int sendSMS(char opt,char* message);//opt define qual o número de telefone

  private:
       char phoneNumber1[14];
       char phoneNumber2[14];
       char phoneNumber3[14];
       char buffer[24];
       boolean ok;
};
#endif
```

```
#include "Arduino.h"
#include "GsmRT.h"


GsmRT::GsmRT(){
       pinMode(7,OUTPUT);
}

void GsmRT::onOff(){
       digitalWrite(7,LOW);
       delay(1000);
       digitalWrite(7,HIGH);
       delay(2000);
       digitalWrite(7,LOW);
       delay(3000);
}

void GsmRT::off(){
       Serial.println("AT+CPOWD=1");
}

void GsmRT::setPhoneNumber(char opt,char* number){
       int index = -1;

       switch(opt){
        case '1':
                    do{
```

```
                        index++;
                        phoneNumber1[index] = number[index];
                        } while(number[index] != '\0');
                        break;
         case '2':
                        do{
                        index++;
                        phoneNumber2[index] = number[index];
                        } while(number[index] != '\0');
                        break;
         case '3':
                        do{
                        index++;
                        phoneNumber3[index] = number[index];
                        } while(number[index] != '\0');
                        break;
         default:
        }
}


int GsmRT::sendSMS(char opt,char* message){

        //Sets SMS mode to text
        Serial.println("AT+CMGF=1");
        delay(100);


        //Generates : AT+CMGS="phoneNumberX" X = opt
        buffer[0] = '\0';
        strcat(buffer,"AT+CMGS=\"");

        switch(opt){
         case '1':
                        strcat(buffer,phoneNumber1);
                        break;
         case '2':
                        strcat(buffer,phoneNumber2);
                        break;
         case '3':
                        strcat(buffer,phoneNumber1);
                        break;
         default:
        }

        strcat(buffer,"\"");//\\0");

        //Sends generated command
        Serial.println(buffer);
        delay(100);
        //Sends contents of SMS message
        Serial.println(message);
        delay(100);
        //Sends "ctrl+z" indicating the end of the message and sends message
        Serial.println((char)26);
        return (0);
}
```

Anexo I

Abaixo seguem informações relacionadas a Figura 3.8 retirada de datasheet da Texas Instruments.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI ' s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI ' s standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or " enhanced plastic. " Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer ' s risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products Applications

Audio www.ti.com/audio Automotive and Transportation www.ti.com/automotive
Amplifiers amplifier.ti.com Communications and Telecom www.ti.com/communications
Data Converters dataconverter.ti.com Computers and Peripherals www.ti.com/computers
DLP ® Products www.dlp.com Consumer Electronics www.ti.com/consumer-apps
DSP dsp.ti.com Energy and Lighting www.ti.com/energy
Clocks and Timers www.ti.com/clocks Industrial www.ti.com/industrial
Interface interface.ti.com Medical www.ti.com/medical
Logic logic.ti.com Security www.ti.com/security
Power Mgmt power.ti.com Space, Avionics and Defense www.ti.com/space-avionics-defense
Microcontrollers microcontroller.ti.com Video and Imaging www.ti.com/video
RFID www.ti-rfid.com
OMAP Mobile Processors www.ti.com/omap
Wireless Connectivity www.ti.com/wirelessconnectivity
TI E2E Community Home Page e2e.ti.com
Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265