UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

CARLOS HENRIQUE OLIVEIRA DE OLIVEIRA

DESENVOLVIMENTO DE DISPOSIVO DE RASTREAMENTO GPS COM COMUNICAÇÃO GSM

VITÓRIA – ES
DEZEMBRO/2014

CARLOS HENRIQUE OLIVEIRA DE OLIVEIRA

DESENVOLVIMENTO DE DISPOSIVO DE RASTREAMENTO GPS COM COMUNICAÇÃO GSM

Parte manuscrita do Projeto de Graduação do aluno **Carlos Henrique Oliveira de Oliveira**, apresentada ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

_____

Prof. Dr. Evandro Ottoni Teatini Salles
Orientador

_____

Eng. Philipe Rangel Demuth
Coorientador

_____

Profª. Drª. Raquel Frizera Vassalo
Examinadora

_____

Eng. Me. Fabio Louvatti do Carmo
Examinador

VITÓRIA – ES
DEZEMBRO/2014

RESUMO

Este trabalho propõe a construção de um dispositivo de rastreamento GPS (*Global Positioning System*) para ser usado na prática de esportes de aventura. Esse dispositivo além de oferecer a um usuário local as informações de posicionamento, disponibiliza essas informações a um usuário remoto através de uma mensagem SMS (*Short Message Service*). Para isso foi utilizado um módulo GSM (*Global System for Mobile Communications*), um módulo GPS e um microcontrolador PIC. O objetivo do dispositivo desenvolvido é facilitar operações de resgate, caso o usuário local se perca na prática da atividade esportiva.

Foram realizados testes que comprovam a precisão das medidas obtidas pelo módulo GPS e que comprovam a capacidade que o sistema tem de se comunicar através da rede GSM.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AS	Antispoofing
ASCII	American Standard Code for Information Interchange
AT	Attention
BSS	Base Station Subsystem
C/A	Coarse Aquisition
CSD	Circuit Switched Data
DoD	Departament of Defence
EEPROM	Electric Erasable Programmable Read-Only Memory
GPRS	Global Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communications
I2C	Inter-Integrated Circuit
ISDN	Integrated Services Digital Network
LCD	Liquid Crystal Display
MCS	Master Control Station
MDF	Medium Density Fiberboard
MS	Mobile Station
NMEA	National Marine Electronics Association
NSS	Network Switching Subsystem
OMS	Operation and Management Subsystem
P	Precise
PLMN	Public Land Mobile Network
PC	Personal Computer
PRN	Pseudorandom Noise
PWM	Pulse Width Modulation
RAM	Random Access Memory
SIM	Subscriber Identity Module
SMS	Short Message Service
SPI	Serial Peripheral Interface
TTFF	Time To First Fix
UART	Universal Asynchronous Receiver/Transmiter
USB	Universal Serial Bus

SUMÁRIO

1 INTRODUÇÃO

Dentre a grande variedade das motivações de vida do homem, uma que se faz presente em várias pessoas é o desejo de descobrir o novo, de conquistar. Esse desejo se mostra de maneiras diferentes e a história comprova isso, revelando muitos exploradores, descobridores e inventores. Como exemplo podemos citar as grandes navegações, a industrialização da produção, a invenção do avião, a ida do homem à lua. Todas essas realizações comprovam a curiosidade e a inventividade do ser humano, porém elas não seriam possíveis sem um simultâneo avanço tecnológico que as viabilizasse. Na verdade, em vários casos essas conquistas humanas é que foram o motivo para que esse avanço ocorresse. Não se pode negar que evolução tecnológica e desenvolvimento humano caminham lado a lado, sendo que essa relação de causa e efeito pode se alternar. O fato é que a evolução de instrumentos de medição e processamento de informações, juntamente à sua popularização influenciam diariamente a vida das pessoas propondo soluções cada vez mais eficientes e confiáveis para problemas práticos.

1.1 Motivação

Por mais que a maioria das pessoas não seja parte de grandes feitos, como os acima citados, suas conquistas pessoais cumprem importante papel no seu desenvolvimento enquanto ser humano. Para alguns, essas conquistas pessoais envolvem esportes de aventura, que frequentemente são praticados ao ar livre, como *trekking*, *mountain bike* e escalada.
Esses esportes trazem consigo, entretanto, riscos, dentre eles o risco de se perder. Esse risco pode ser mitigado através do uso de mapas, bússolas e observação do céu. Porém, esse é um caso em que o uso de uma tecnologia mais avançada, como o GPS (*Global Positioning System*), pode simplificar o problema, diminuindo a gravidade das possíveis consequências de se perder em uma trilha, por exemplo.

Além da localização em si, provida pelo GPS, pode ser interessante permitir que essa informação seja acessada por um usuário remoto que esteja monitorando um grupo que pratica *trekking*, por exemplo. Essa funcionalidade pode permitir que esse grupo, caso se perca, seja mais facilmente localizado, agilizando o resgate, principalmente em locais de difícil acesso, como mata fechada. Para isso, é possível usar um modem GSM (*Global System*

for Mobile Communications) que envie mensagens SMS (*Short Message Service*) contendo sua localização ao usuário remoto.

O dispositivo proposto neste trabalho aborda somente uma das soluções possíveis para o problema descrito. Outra solução, mais óbvia, seria desenvolver um aplicativo para algum *smartphone* que seja capaz de executar as mesmas tarefas, a saber, obter seu posicionamento GPS e enviá-lo através de uma mensagem SMS. Esse trabalho, porém, propõe um *hardware* dedicado à execução da função para a qual foi desenvolvido. Isso proporciona algumas vantagens em relação à utilização de um *hardware* multifuncional. Dentre essas vantagens é possível destacar: facilidade de integração de *hardwares* auxiliares (no caso deste trabalho foi utilizado um sensor de temperatura e um sensor pressão barométrica) e o baixo custo, quando comparado a dispositivos multifuncionais disponíveis no mercado.

1.2 Justificativa

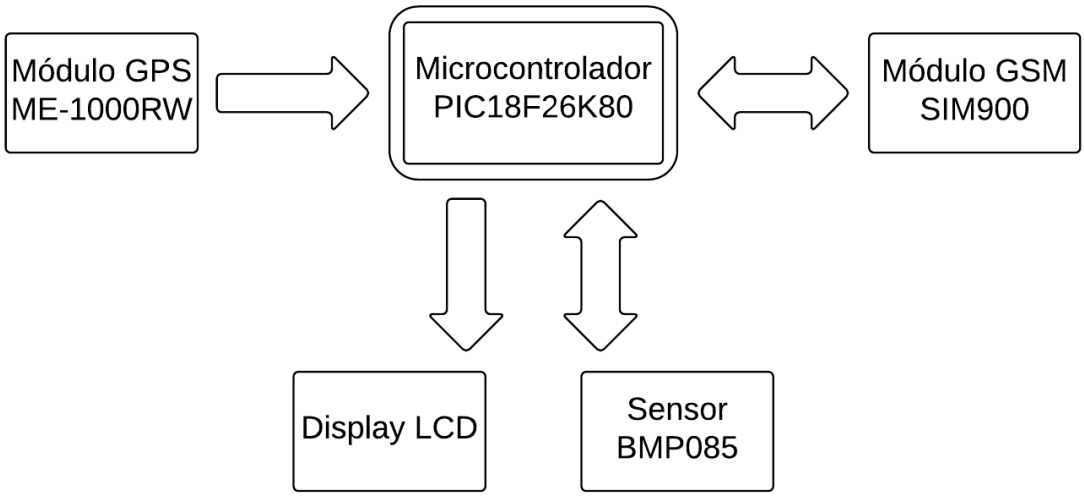
A principal justificativa deste trabalho se encontra na possibilidade de aumento da segurança das pessoas que utilizem o dispositivo por ele proposto. Além disso, o sistema pode ser utilizado em outras aplicações, como rastreamento de veículos e base de comunicação e localização de estações de monitoramento. Para esse último caso, o dispositivo pode ser útil no cumprimento da Lei Federal Nº 9.433, de 8 de janeiro de 1997, que prevê o monitoramento metereológico das bacias hidrográficas nacionais em acordo com a Política Nacional de Recursos Hídricos.

1.3 Objetivos

1.3.1 Objetivo Geral

Neste trabalho é feito um esforço para aumentar a segurança de pessoas na prática de atividades esportivas em locais onde pode haver o risco de se perder. Sendo assim, ele tem como objetivo o desenvolvimento de um dispositivo que seja capaz de obter sua localização informando-a a um usuário local, e que seja também capaz de enviar as informações dessa localização a um usuário remoto. Assim, pretende-se agilizar uma possível operação de resgate, o que torna a prática dessas atividades esportivas mais segura. Na Figura 1 é apresentada uma ideia básica do sistema proposto.

Figura 1 - Diagrama de blocos do sistema proposto



Fonte: Produção do próprio autor

Esse sistema é composto por um microcontrolador (PIC18F26k80) responsável por receber os dados do módulo GPS (ME-1000RW) e por controlar o módulo GSM (SIM900) para que a posição GPS seja enviada como uma mensagem de texto. O microcontrolador também é responsável por se comunicar com o sensor de pressão barométrica (BMP085) a fim de, através da pressão do atmosférica medida, calcular a altitude em que o dispositivo se encontra em relação ao nível do mar. Finalmente, o microcontrolador deve dispor todas as informações obtidas através do módulo GPS e através do sensor de pressão barométrica em um display LCD.

1.3.2 Objetivos Específicos

- Desenvolver uma unidade central de controle com tela utilizando um microcontrolador PIC.
- Implementar comunicação serial entre o módulo GSM e a unidade central de controle.
- Implementar comunicação serial entre o módulo GPS e a unidade central de controle.
- Ler e interpretar os dados obtidos do módulo GPS.
- Desenvolver um programa que permita a unidade central de controle enviar mensagens SMS.

1.3.3 Divisão da Monografia

Este trabalho está dividido nos seguintes capítulos

- Capítulo 2: *Embasamento Teórico* – aborda os conceitos básicos e os princípios de funcionamento do sistema GPS e da tecnologia GSM;

- Capítulo 3: *Desenvolvimento* – apresenta os *hardwares* utilizados e também o *hardware* e o *firmware* desenvolvidos nesse trabalho;
- Capítulo 4: *Testes e Resultados* – descreve os testes realizados no sistema desenvolvido e apresenta seus resultados;
- Capítulo 5: *Conclusão* – apresenta a conclusão final do projeto.

2 EMBASAMENTO TEÓRICO

2.1 GPS

O Sistema de Posicionamento Global (GPS) é uma utilidade de propriedade dos Estados Unidos que fornece aos seus usuários informações de posicionamento, de navegação e de tempo. O sistema consiste de três segmentos: o segmento espacial, o segmento de controle e o segmento de usuário. A força aérea norte americana desenvolve, mantém e opera os segmentos espacial e de controle [1].

2.1.1 Segmento Espacial

O segmento espacial é composto por 24 satélites dispostos em seis planos orbitais, de maneira que no mínimo quatro desses satélites estão visíveis de qualquer ponto da terra. A órbita dos satélites GPS, que estão a aproximadamente 20.200 km da terra, são praticamente circulares, com inclinação de 55º em relação ao equador, e possuem um período orbital de 12 horas siderais (aproximadamente 11 horas e 58 minutos) [2]. A Figura 2 ilustra a cobertura de satélites GPS.

Figura 2 - Disposição dos satélites GPS em órbitas

Fonte: McNamara (2004)

Cada um desses satélites é composto essencialmente por transceptores de rádio, relógios atômicos, computadores e equipamentos auxiliares que permitem a emissão de sinais em duas radiofrequências [3].

Um satélite GPS transmite um sinal de radiofrequência composto por duas frequências portadoras moduladas por dois códigos digitais e uma mensagem de navegação. A portadora L1 é gerada a 1575,42 MHz e a portadora L2 a 1227,60 MHz, o que equivale a comprimentos de onda de 19 cm e 24 cm, respectivamente. A transmissão de duas frequências permite corrigir o pior erro a que o sinal GPS está submetido, a refração ionosférica – explicada com maiores detalhes na Seção 3.1.5 deste trabalho.

Os códigos digitais usados na modulação das portadoras são o código C/A (*Coarse Aquisition*) e o código P (*Precise*). Cada código consiste em uma sequência binária de zeros e uns. Esses códigos são conhecidos como códigos pseudorrandômicos, ou PRN (*Pseudorandom Noise*), pois parecem ser gerados aleatoriamente, mas, na verdade, são gerados por algoritmos, presentes tanto no satélite como no receptor. O código C/A é usado para modular a portadora L1 e o código P é usado para modular a portadora L2.

O código C/A é uma sequência de 1023 bits que se repetem a cada milissegundo. Cada satélite possui seu próprio código C/A, assim o receptor de sinal GPS pode identificar qual satélite está transmitindo um código em particular.

O código P é uma sequência de, aproximadamente, $2{,}35 \times 10^{14}$ bits dividida em 38 segmentos, cada um com uma semana de duração (é taxa de transmissão é de 10.23Mbit/s). Cada satélite é identificado por transmitir uma dessas 38 sequências transmitindo, então, uma sequência própria com 6×10^{12} bits, aproximadamente. O código P foi projetado para ser usado pelo exército dos Estados Unidos da América e, por esse motivo, ele é encriptado pela adição de um código chamado código W, gerando o código Y. Essa encriptação é chamada de *antispoofing* (AS). O código Y é o que de fato modula a portadora L2, portanto é necessário que o receptor conheça o código W para que possa extrair o código P.

A mensagem de navegação GPS é adicionada tanto à portadora L1 como à L2.
Ela é dividida em três partes. A primeira parte contem a hora e a data GPS, além do status do satélite e de sua condição de operação. A segunda parte contem as informações orbitais do satélite conhecida como Efemérides, o que permite que o receptor calcule a posição do satélite. A terceira parte contem informações de posição geocêntrica (WGS84) e status de todos os satélites [2].

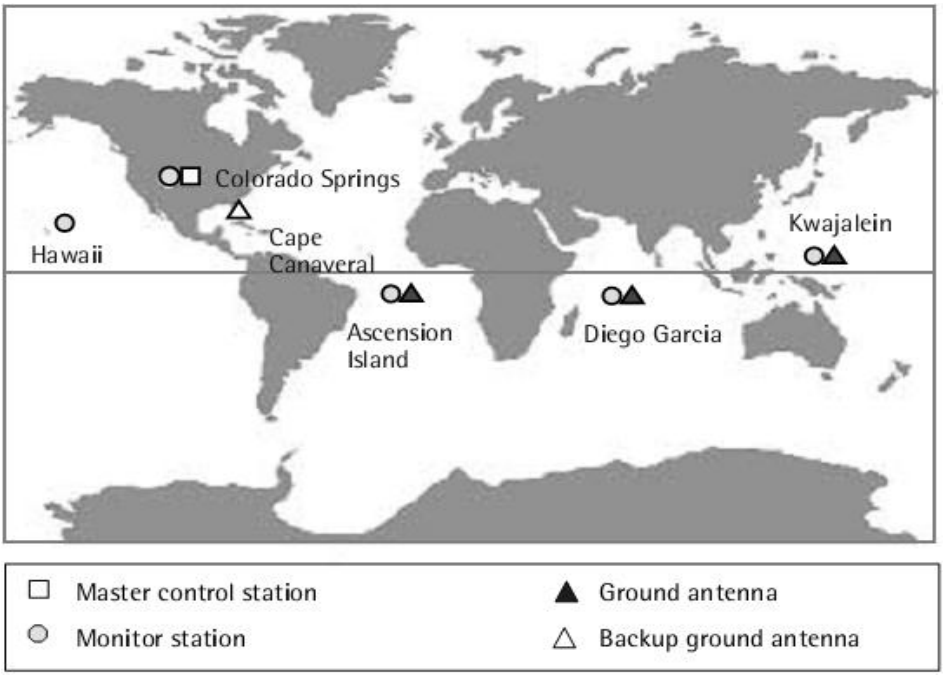
2.1.2 Segmento de Controle

O segmento de controle tem como objetivo monitorar os satélites a fim de calcular as correções de relógio, predizer as coordenadas e atualizar as mensagens de navegação deles [3], [4].

Este segmento é composto de estação central de controle (MSC), conforme mostrado na Figura 3, quatro antenas dedicadas e uma rede de estações de monitoramento [2].

A rede de estações de monitoramento é composta por seis estações localizadas no Havai, Cabo Canaveral, Ilhas Ascension, Kwajalein, Diego Garcia e nos EUA, em Colorado Springs, onde se encontra a estação central de controle.

Figura 3 - Locais de controle GPS



Fonte:Rabbany (2002)

As estações de monitoramento captam os sinais de cada satélite GPS e repassam as informações obtidas para a estação central de controle. Através dessas informações a estação central de controle calcula os fatores de correção de relógio e as coordenadas de cada satélite como uma função do tempo. Os parâmetros calculados são repassados aos satélites através das antenas em Cabo Canaveral, em Kwajalein, em Diego Garcia e nas Ilhas Ascension.

2.1.3 Segmento de Usuário

Este segmento consiste de dispositivos eletrônicos espalhados por todo o mundo que são capazes de captar e processar sinais de satélites GPS para, assim, estimar sua posição no globo (longitude, latitude e altitude). Alguns exemplos podem ser vistos na Figura 4.

Figura 4 - Exemplos de dispositivos GPS

Fonte: Site da Topcon(2014)

Fonte: Site da Garmin (2014)

Fonte: Site da Topcon (2014)

Fonte: Site da Unistrong (2014)

Inicialmente, esses receptores de sinal GPS eram classificados em três grupos, que são caracterizados pelo tipo de observável (Código C/A ou P) a ser processada [3].

Em um desses grupos estão os receptores de Código P, que foram projetados exclusivamente para uso militar. Seu uso é feito somente mediante autorização de DoD (Departamento de Defesa dos EUA).

Nos outros dois grupos estão os receptores de Código C/A, que podem ser usados livremente pela sociedade civil. Os dispositivos de um desses dois grupos se utilizam da frequência da portadora do sinal GPS para calcular pseudodistâncias. Essa técnica gera resultados de medição mais precisos do que o resultado gerado pelos receptores do outro grupo que utiliza somente a informação contida na portadora L1. Atualmente, existem mais de 500 tipos diferentes de receptores GPS disponíveis comercialmente [2].

2.1.4 Funcionamento

O GPS funciona baseado em uma idéia muito simples. Os satélites GPS enviam constantemente, através de *broadcast*, as mensagens de navegação e, juntamente a elas, o exato momento em que foram enviadas.

Suponha que em um dado instante t o receptor de GPS recebe sinais de três satélites. As informações do primeiro satélite revelam que ele estava na posição (x_1, y_1, z_1) no instante t_1. Dessa forma é possível afirmar que o receptor se encontra na superfície de uma esfera centrada em (x_1, y_1, z_1) e com raio $c(t - t_1)$, ou seja, representa uma solução da equação

$$(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 = c^2(t - t_1)^2 \qquad (1)$$

Onde c é a velocidade da luz no vácuo, (x, y, z) é um ponto na esfera em torno do satélite, centrado em (x_1, y_1, z_1), e t é o instante em que os dados do satélite chegaram ao receptor.

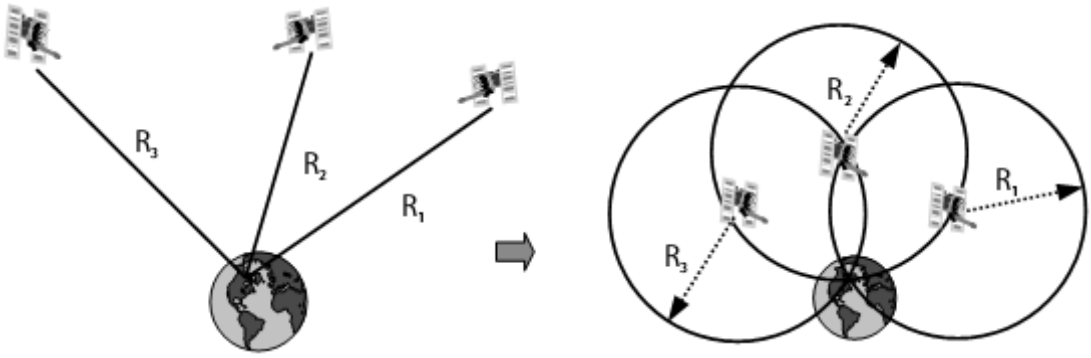
Tratando de maneira análoga as informações dos outros dois satélites, a posição do receptor pode ser vista como uma das duas soluções do seguinte sistema de equações quadráticas:

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 = c^2(t - t_1)^2 \\ (x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 = c^2(t - t_2)^2 \\ (x - x_3)^2 + (y - y_3)^2 + (z - z_3)^2 = c^2(t - t_3)^2 \end{cases} \qquad (2)$$

Onde (x_1, y_1, z_1), (x_2, y_2, z_2) e (x_3, y_3, z_3) representam a posição de cada um dos três satélites, e t_1, t_2 e t_3 representam o instante em que cada uma das três posições chegaram ao receptor.

A Figura 5 representa visualmente o que acima foi descrito analiticamente.

Figura 5 - Idéia básica do posicionamento GPS



Fonte: Rabbany (2002)

Das duas soluções possíveis uma delas deverá estar claramente fora da superfície da terra, sendo descartada.

Essa idéia básica, porém, não apresentaria bons resultados por si só, pois erra ao assumir que o relógio do receptor tem precisão suficiente para o cálculo das distâncias. Isso porque esse cálculo se utiliza da velocidade da luz que é de $300.000\ km/s$.

Um receptor comercial tem seu relógio baseado em um oscilador de quartzo, que pode gerar erros de 1 segundo por dia. A multiplicação desse erro pela velocidade da luz compromete a localização do receptor calculada. Suponha que o erro do relógio é de apenas 1ms, a posição do receptor seria calculada com erro de 300 km.

Alem disso, o receptor GPS não está com seu relógio sincronizado com os relógios dos satélites. Assim, independentemente do erro gerado pela imprecisão do oscilador do receptor, as distâncias entre receptor e cada satélite não podem ser calculadas, pois sem sincronismo não é possível saber o tempo que o sinal demorou para viajar desde o satélite até o receptor.

Felizmente, a defasagem de tempo entre relógio do receptor e o relógio do sistema GPS (sincronizado pela Estação Central de Controle) acrescida do erro gerado pela imprecisão do oscilador é a mesma para as três medidas. Assim, esse problema é resolvido acrescentando as informações de um quarto satélite e resolvendo um sistema com quatro equações quadráticas, em que o erro do relógio de quartzo em relação ao relógio atômico presente no satélite mais a

defasagem temporal em relação à hora GPS é uma das incógnitas (Δ_t) e aparece somada ao tempo de recepção t.

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 = c^2(t + \Delta_t - t_1)^2 \\ (x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 = c^2(t + \Delta_t - t_2)^2 \\ (x - x_3)^2 + (y - y_3)^2 + (z - z_3)^2 = c^2(t + \Delta_t - t_3)^2 \\ (x - x_4)^2 + (y - y_4)^2 + (z - z_4)^2 = c^2(t + \Delta_t - t_4)^2 \end{cases}$$

2.1.5 Fontes de Erros

Além do erro de relógio, dito anteriormente, existem outras fontes de erro que interferem na precisão da medida GPS, as quais estão relacionadas, principalmente, com a qualidade do sinal.

a) Ionosfera

A ionosfera é uma região composta por elétrons livres e que está situada entre 100 e 1000 km acima da superfície terrestre. Essa camada ionizada é capaz de afetar diretamente a portadora do código C/A. O pior efeito causado pelos elétrons livres é o atraso na modulação da portadora, que também pode ser visto como um aumento do caminho a ser percorrido pelo sinal desde o satélite até o receptor.

A maior parte do erro causado pelo atraso da ionosfera é minimizada se a distância entre as estações for da ordem de 10km, pois os sinais ao atravessar a mesma região da ionosfera sofrem os mesmos atrasos na modulação da portadora. Para distâncias superiores a 100 km é necessário o uso de receptores de dupla frequência, porque a portadora L2 tem maior comprimento de onda, sofrendo menos com esse efeito.

A intensidade dos efeitos da refração ionosférica depende da hora do dia, da estação do ano, da atividade solar, da latitude.

b) Troposfera

A região troposférica é uma região gasosa que está entre a superfície da terra e aproximadamente 40 km de altitude. O atraso gerado pela refração troposférica não depende da frequência do sinal, mas sim da densidade da atmosfera e do ângulo de elevação do satélite, podendo causar erros de até 30m.

A troposfera pode ser considerada como a mistura de dois gases ideais: vapores de água e ar seco. A parte seca representa cerca de 90% de toda a refração troposférica, porém pode ser modelada com precisão, e assim seu efeito na medição GPS pode ser drasticamente reduzido. Os outros 10% desse tipo de refração são devidos à parte úmida e não podem ser modelados. O atraso troposférico é influenciado pela temperatura, umidade e pressão, que variam com a altitude do local.

c) Perda de ciclos

Durante uma observação do sinal GPS, se houver alguma descontinuidade na medida da fase da portadora, ocorre a perda de ciclo. Isso pode ocorrer quando há aceleração da antena do receptor, algum problema no receptor seja de *hardware* ou *software*, variação nas condições atmosféricas, bloqueio do sinal, dentre outros motivos.

d) Multicaminhos

Esse erro é devido a reflexões indesejadas do sinal GPS em superfícies próximas ao receptor. Essas reflexões podem ser horizontais, verticais ou uma combinação dessas duas. O efeito dessas reflexões é que, em relação ao sinal que percorre um caminho direto, o sinal que é refletido sempre percorre um caminho aumentado, o que representa um atraso em sua recepção e, consequentemente, uma interpretação errada da distância entre a antena receptora e o satélite que gerou o sinal. A omissão de sinais gerados por satélites com baixa elevação pode diminuir o efeito causado por esse erro, já que os sinais de satélites nessa condição são mais susceptíveis a multicaminhamento.

e) Antispoofing

Como citado anteriormente, o código P é criptografado antes de ser enviado pelos satélites, sendo transformado em código Y, não acessível a usuários civis. O AS faz com que receptores do código P necessitem de um AOC (*Auxiliary Output Chip*) para funcionarem corretamente. Esse AOC possui a chave de decriptografia para a correta interpretação do código Y. Não utilizar o código P significa perder precisão do sinal GPS, devido à falta de condições de eliminar grande parte dos efeitos ionosféricos somente utilizando o código C/A. Alguns receptores utilizam modelos ionosféricos

para aumentar a precisão das medidas, outros tentam extrair o código P a partir do código Y por meio de um processo de correlação [5].

2.1.6 Protocolo NMEA

O protocolo de comunicação serial utilizado pelos receptores de GPS é baseado no padrão da *National Marine Electronic Association*, conhecido como NMEA-0183. Todas as mensagens desse padrão são iniciadas com um "$" e terminadas com um CR/LF (*carriage return*/*line feed*). As mensagens são compostas por caracteres ASCII (*American Standard Code for Information Interchange*). As informações contidas em cada mensagem estão em uma única linha, separadas por vírgulas. Cada sentença possui um prefixo GP, seguido de três letras que identificam o seu conteúdo. Neste trabalho foi usada a versão 3.01 desse protocolo.

As sentenças presentes no protocolo NMEA-0183 v3.01 são:

- GPGGA – *Global Positioning System Fized Data*;
- GPGLL – *Geographic Position – Latitude/Longitude*;
- GPVTG – *Course Over Ground and Ground Speed*;
- GPRMC – *Recommended Minimum Specific GPS/Transit Data*;
- GPGSA – *GNSS DOP and Active Satellites*;
- GPGSV – *GNSS Satellites in View*;
- GPMSS – *MSK Receiver Signal*;
- GPZDA – *SiRF Timing Message.*

No projeto desenvolvido nesse trabalho foram utilizadas somente as seguintes sentenças: GPGGA, da qual são obtidas as informações de hora, latitude e longitude, modo de operação do GPS e número de satélites dos quais o módulo está recebendo os dados; GPVTG, que oferece informação de velocidade; e, GPZDA, de onde são obtidas informação de data e hora. Nas Tabelas 1, 2, 3 e 4 estão detalhados o formato e o conteúdo de cada uma das sentenças utilizadas neste trabalho.

Exemplo de mensagem GPGGA:

$GPGGA,011727.817,2019.1828,S,04019.7811,W,1,05,1.5,51.9,M,-9.9,M,,0000*74<CR><LF>

Tabela 1 - Formato da Sentença GPGGA

Nome	**Exemplo**	**Unidade**	**Descrição**
Prefixo	$GPGGA		Cabeçalho do Protocolo
Hora UTC	011727.817		hhmmss.sss
Latitude	2019.1828		ggmm.mmmm
Indicador N/S	S		N = Norte ou S = Sul
Longitude	04019.7811		ggmm.mmmm
Indicador E/W	W		E = Leste ou W = Oeste
Indicador de tipo	1		Ver tabela 2
Satélites utilizados	05		De 0 a 14
HDOP	1.5		Diluição horizontal da precisão
Altitude MSL	51.9	metros	
Unidade	M	metros	
Separação do Geóide	-9.9	metros	
Unidade	M	metros	
Age of Diff. Corr.		segundos	Campo vazio se DGPS não utilizado
Identificação da estação de referência diferencial	0000		
Checksum	*74		
<CR><LF>			Fim da Mensagem

Tabela 2 - Tipo de posicionamento

Valor	**Descrição**
0	Posição não disponível ou inválida
1	Sistema padrão de posicionamento, posição válida
2	GPS diferencial, posição válida
3-5	Não utilizado
6	Modo *Dead Reckoning*, posição válida

Exemplo de uma mensagem GPVTG:

$GPVTG,026.0,T,,M,000.0,N,000.0,K,A*09<CR><LF>

Tabela 3 - Formato da Sentença GPVTG

Nome	**Exemplo**	**Unidade**	**Descrição**
Prefixo	$GPVTG		Cabeçalho do Protocolo
Curso	026.0	Graus	Sentido em relação norte
Referencia	T		Verdadeira
Curso		Graus	Sentido em relação norte
Referência	M		Magnética
Velocidade	000.0	Nós	Velocidade horizontal medida
Unidade	N		
Velocidade	000.0	Km/h	Velocidade horizontal medida
Unidade	K		Quilômetros por hora
Modo	A		A=Autônomo, D=DGPS, E=DR
Checksum	*09		
<CR><LF>			Fim da Mensagem

Exemplo de Mensagem GPZDA:

$GPZDA,011727.817,23,12,2014,00,00*5F<CR><LF>

Tabela 4 - Formato da Sentença GPZDA

Nome	**Exemplo**	**Unidade**	**Descrição**
Prefixo	$GPZDA		Cabeçalho do Protocolo
Hora UTC	011727.817		hhmmss.sss
Dia	23		01 a 31
Mês	12		01 a 12
Ano	2014		1980 a 2079
Hora Local	00		Em relação a hora UTC
Minuto Local	00		Em relação a hora UTC
Checksum	*5F		
<CR><LF>			Fim da Mensagem

2.2 GSM

Antes da década de 1980, o mercado de telefonia celular na Europa possuía vários padrões analógicos. A falta de compatibilidade gerada por esse fato inviabilizava a utilização de um mesmo serviço em diferentes regiões. Por conta disso, um esforço de uniformização dos padrões levou à criação do GSM (*Global System for Mobile Communications*), um serviço de telefonia digital de alta capacidade, que rapidamente se tornou o padrão mais utilizado no mundo. No Brasil essa tecnologia entrou em funcionamento em 2002.

O sistema GSM, também chamado de sistema de segunda geração, tem como principal característica a introdução de protocolos de transmissão de dados que adaptam o canal de voz para transmitir bits. Isso permitiu que essa tecnologia pudesse se desenvolver à medida que novas demandas foram surgindo e novas aplicações se tornaram necessárias, introduzindo no serviço de telefonia celular, por exemplo, o envio de mensagens de texto e serviço de dados a altas taxas.

2.2.1 Arquitetura

A arquitetura GSM é composta por quatro subsistemas, a saber: Estação Móvel (MS, *Mobile Station*), Sistema de Estação Base (BSS, *Base Station Subsystem*), Sistema de Comutação de Rede (NSS, *Network Switching Subsystem*) e Sistema de Operação e Manutenção (OMS, *Operation and Management Subsystem*). A junção desses quatro subsistemas compõe uma Rede Móvel Pública Terrestre (PLMN, *Public Land Mobile Network*). A PLMN é capaz de realizar somente chamadas locais entre seus assinantes móveis. Na maioria das vezes, é necessário o uso de uma rede de telefonia fixa para rotear as chamadas. Normalmente, uma operadora de telefonia celular fornece tanto a PLMN como a rede de telefonia fixa aos seus clientes [6].

Dentro de uma PLMN, o BSS provê e gerencia os caminhos de transmissão entre as MSs e o NSS. O NSS, além de gerenciar as conexões, conecta uma MS a outra MS ou a alguma outra rede de interesse. Assim, o NSS não entra diretamente em contato com uma MS. Da mesma forma que uma BSS não entra em contato direto com uma rede externa. Portanto, MS, BSS e NSS formam a parte operacional da rede GSM, provendo e estabelecendo caminhos de

transmissão. Já o OMS é usado pelo provedor do serviço para controlar e gerenciar o sistema GSM.

2.2.2 Estação Móvel

A Estação Móvel (MS) é o equipamento físico usado pelo assinante para acessar os serviços da PLMN, podendo ser um celular, um computador ou qualquer outro sistema de comunicação de voz ou dados.

Uma MS é dividida em duas partes: um equipamento móvel, que contém o *hardware* e o *software* que compõem o rádio e a interface com o usuário; e, o módulo de identidade do assinante (SIM, *Subscriber Identity Module*). Para que uma MS se comunique com o BSS, o SIM *card* deverá estar conectado ao equipamento móvel, pois sem SIM *card* não há usuário associado à rede.

2.2.3 Sistema de Estação Base

O Sistema de Estação Base (BSS) é responsável por conectar a MS ao NSS. Um sinal é enviado pela MS ao BSS, este capta o sinal e extrai dele as informações. Essas informações são enviadas à rede (PLMN). O BSS envia à MS os dados que recebe da rede em um sinal que ela é capaz de interpretar.

O BSS é composto por três elementos, sendo um responsável por trocar sinais com as MSs, outro responsável por controlar esse primeiro e se comunicar com a rede, e um terceiro que faz uma conversão do sinal para que ele seja transmitido à MS. O primeiro desses três elementos é a Estação Transceptora, que é composto basicamente por antenas de rádio frequência. O segundo, Controlador de Estação Base, controla as conexões e as comutações de um grupo de Estações Transceptoras. O terceiro elemento é o Transcodificador, que converte o sinal de voz enviado pela MS de 64kbits/s para 16kbits/s a fim de otimizar o uso da banda de transmissão. Chamadas de dados não são comprimidas pelo Transcodificador.

2.2.4 Sistema de Comutação de Rede

O NSS gerencia as comutações entre usuários GSM e a rede, controla e administra a mobilidade dos usuários, e armazena e consulta a base de dados dos assinantes. O NSS executa as funções de comutação necessárias entre MSs localizadas em determinada região geográfica. O NSS identifica quando uma MS sai de sua região geográfica de origem e vai

para outra, e associa a essa MS uma nova identificação, a fim de que ela seja encontrada nessa nova região. A nova identificação é concedida dinamicamente à MS através de um processo de registro. Nesse processo, o NSS é capaz de identificar a qual PLMN a MS pertence, verificar se o serviço de *roaming* é permitido para essa MS, gerar uma nova identificação para MS e atualizar a base de dados. O NSS é também responsável por conectar usuários GSM a redes de outra natureza, como a ISDN (*Integrated Service Digital Network*).

2.2.5 Sistema de Operação e Manutenção

O OMS é responsável pela segurança da rede, validando as identificações das MSs. Três funções são executadas pelo OMS: acessar a NSS a fim de determinar se uma MS poderá se conectar à rede; atualizar a lista de MSs válidas, fraudulentas ou com algum tipo de problema; e, dar manutenção e operar a PLMN, podendo atuar em cada uma das outras três estruturas da rede GSM citadas anteriormente.

3 DESENVOLVIMENTO

Este trabalho se propôs a desenvolver um dispositivo utilizando um microcontrolador PIC, integrado a módulos GPS e GSM, capaz de informar sua localização tanto ao usuário local como a um usuário remoto.

Para que o usuário local tenha acesso às informações do GPS foi utilizado um *display* LCD (*Liquid Crystal Display*) 20x4, ou seja, um *display* que exibe até 20 caracteres em cada uma de suas 4 linhas. Além das informações do GPS, o dispositivo desenvolvido conta com um sensor de pressão atmosférica e de temperatura, o BMP085, através do qual é possível extrair a altitude em que o dispositivo se encontra. As informações obtidas pelo usuário podem ser enviadas para um número da lista de contatos desse dispositivo através de uma mensagem SMS. Para isso, é utilizado o modem GSM SIM900. O microcontrolador PIC é responsável por coordenar e executar todas as tarefas fazendo o uso de cada módulo citado. O sistema é alimentado por duas baterias recarregáveis de Íon-Lítio de 3000mA/h, que são recarregadas através de uma porta USB.

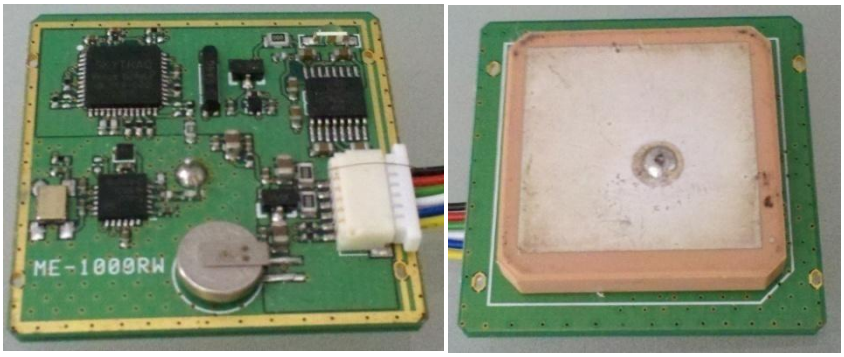
3.1 *Hardwares* Utilizados

A seguir, será apresentado como cada um desses módulos funciona e como o microcontrolador faz uso de suas próprias funcionalidades e das funcionalidades de cada módulo.

3.1.1 Módulo GPS

O módulo GPS utilizado é o ME-1000RW, que é baseado no *chipset* SkyTraq Venus6 (vide Figura 6). Esse módulo possui uma antena GPS de cerâmica integrada, 51 canais de aquisição e 14 canais de rastreamento, sendo capaz de receber sinais de 65 satélites GPS e transferir as informações através de sua porta serial.

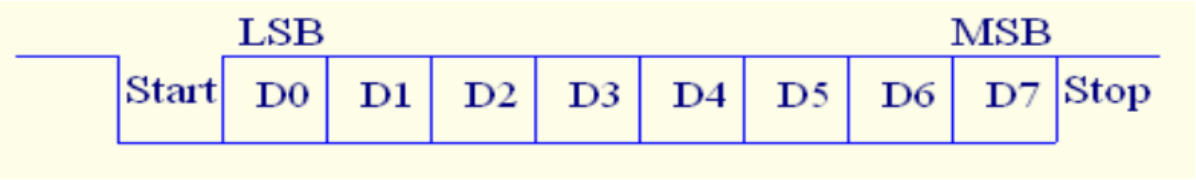
Figura 6 - À direita, vista frontal e, à esquerda, vista traseira do módulo GPS.



Fonte: Produção do próprio autor

Este módulo possui interface RS232 e UART (*Universal Asynchronous Receiver/Transmiter*), ambas podendo operar *full duplex*. Para este trabalho a UART foi configurada para operar a 9600bps, mas esse módulo é capaz de operar também com outras taxas de bits (4800, 38400 e 115200bps). Outras características da interface serial são: 1 *start bit*, 1 *stop bit*, 8 bits de dados, sem bits de paridade. Na Figura 7 há um exemplo genérico de um *byte* de dados enviado.

Figura 7 - Sequencia de Bits da comunicação serial



Fonte: Datasheet do módulo GPS ME-1000RW

O módulo ME-1000RW é capaz de interpretar somente a portadora L1 (1.575,42MHz), ou seja, trabalha com o código C/A. Sua antena lhe confere sensibilidade de -161dBm. De acordo com o *datasheet*, a precisão do receptor é de cerca de 5 metros. As mensagens transmitidas por ele estão de acordo com o protocolo NMEA-0183 v3.01, porém a sentença GPMSS não é suportada. É possível configurar o módulo para enviar a cada segundo quantas e quais mensagens forem necessárias, dentre as que são suportadas. Neste projeto, o módulo foi configurado para emitir as mensagens GPGGA, GPVTG e GPZDA, descritas na Seção 3.1 deste trabalho.

O módulo pode operar tanto em *cold start*, quanto com *hot start*. Isso significa que ele possui uma pequena bateria embutida, o que lhe permite que, quando operando em *hot start*, ao

iniciar, sejam usadas as informações do Almanac e das Efemérides de quando ele esteve pela última vez conectado aos satélites. Quando o módulo opera em *cold start*, essas informações devem ser baixadas a cada nova inicialização. A operação em *hot start* reduz o chamado *Time-To-First-Fix* (TTFF) em relação à operação em *cold start*. O fabricante promete TTFF de 10 segundos em *hot start* e 32 segundos em *cold start*.

A alimentação deve ser feita dentro da faixa de 3.6V a 6V, havendo um consumo de cerca de 23mA [7].

3.1.2 Modulo GSM

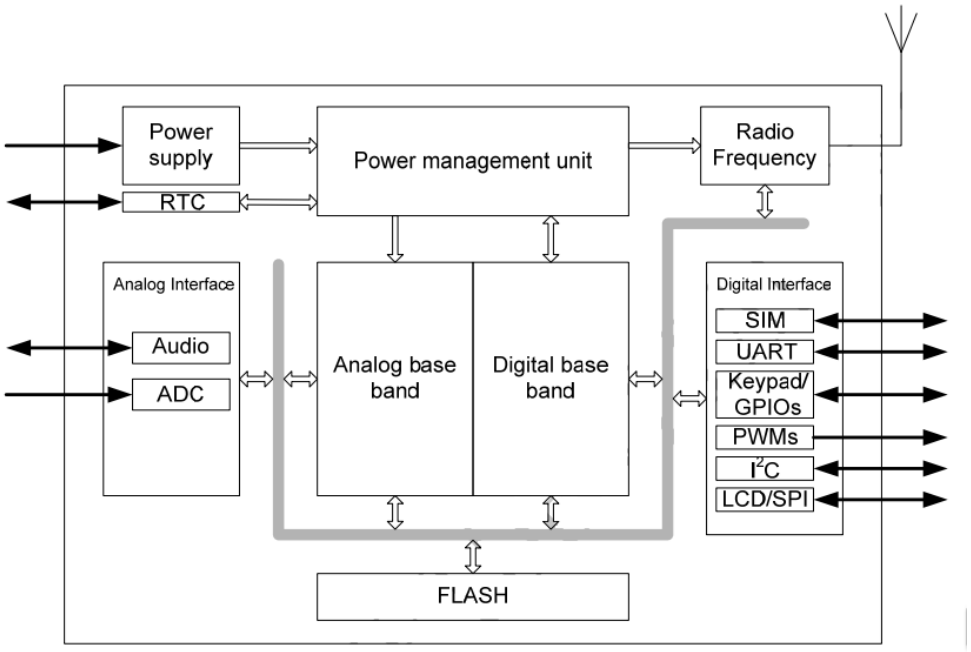
Para o desenvolvimento deste trabalho foi utilizado o módulo GSM IcomSat v1.1 produzido pela Itead Studio e baseado no modem SIM900 fabricado pela SIMCom (vide Figura 8). Este é um modem GSM/GPRS *quadriband* que opera nas frequências GSM 850MHz, EGSM 900MHz, DSC 1800MHz e PCS 1900MHz.

Figura 8 - Modem GSM SIM900

Fonte: Site da SimCom (2014)

O SIM900 é caracterizado pelo baixo consumo, reduzido tamanho e alta confiabilidade. Funciona com alimentação de 3.2V a 4.8V com consumo típico de 10mA, consumindo somente 1mA em *sleep mode* e podendo chegar consumir 440mA durante uma chamada de dados. A Figura 9 mostra um diagrama funcional do SIM900.

Figura 9 - Diagrama funcional do SIM900



Fonte: Datasheet do SIM900

O modem é controlado através de sua UART, que opera em todas as velocidades desde 1200bps até 115200bps. A comunicação é baseada nos comando AT. Neste trabalho, dentre todas as funcionalidades desse modem, somente o envio de mensagens SMS é usado.

Para enviar um SMS o comando utilizado é o +CMGS, que é usado como explicado a seguir.

Envia-se para o módulo, através da UART, a seguinte mensagem, caractere a caractere:

```
>AT+CMGS = "número de celular de destino" <CR>
>"mensagem que se deseja enviar" <ctrl+Z>
```

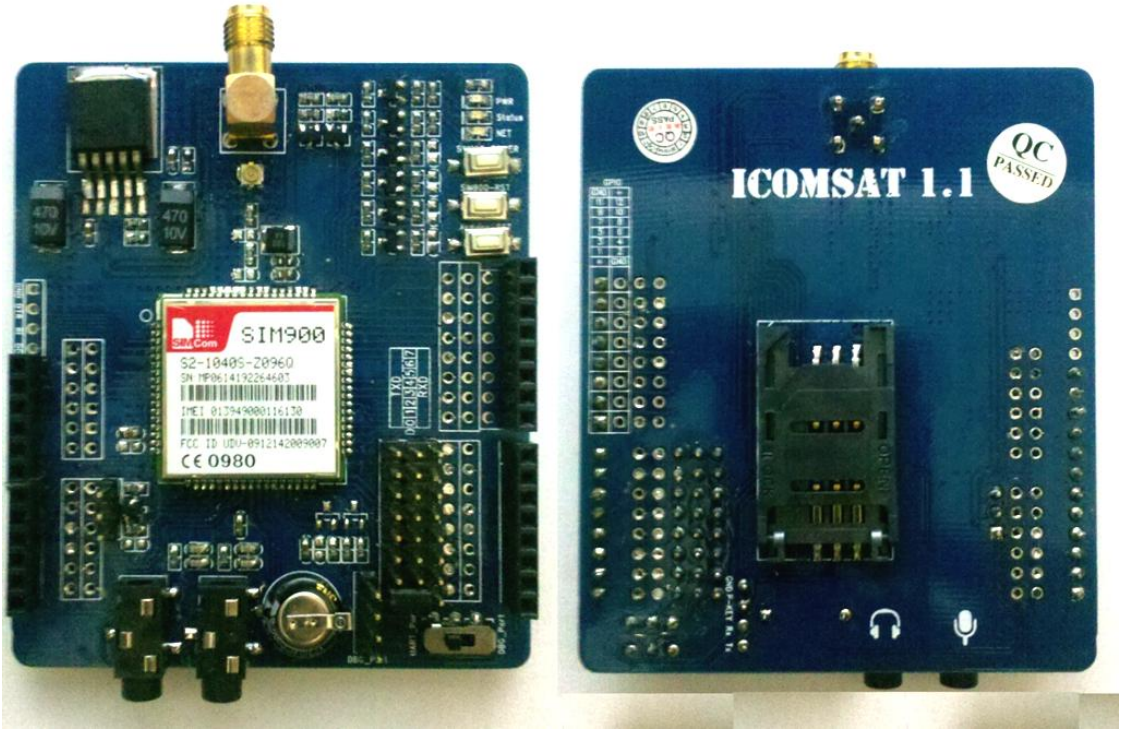
Quando a mensagem é efetivamente enviada o módulo responde:

```
>+CMGS: <nr>
```

Em que, `CR` significa *carriage return*; `ctrl+Z` significa *control* Z, que é equivalente a enviar 1A em hexadecimal; e, `nr` é um identificador da mensagem que vai de 0 a 255.

Como dito anteriormente, o SIM900 utilizado é parte do módulo IcomSat v1.1, mostrado na Figura 10. Esse módulo, basicamente, tem um conector para antena, um *socket* para o SIM *card*, conectores para fone e microfone, um circuito regulador de tensão e leds indicadores de funcionamento, além de dar fácil acesso a todos os pinos do SIM900.

Figura 10 - À direita, vista superior e à esquerda, vista inferior do módulo IcomSatv1.1



Fonte: Produção do próprio autor

A escolha do módulo IcomSat v1.1 é justificada por sua disponibilidade no mercado, baixo custo e facilidade de integração a outros circuitos.

3.1.3 Sensor de Pressão barométrica BMP085

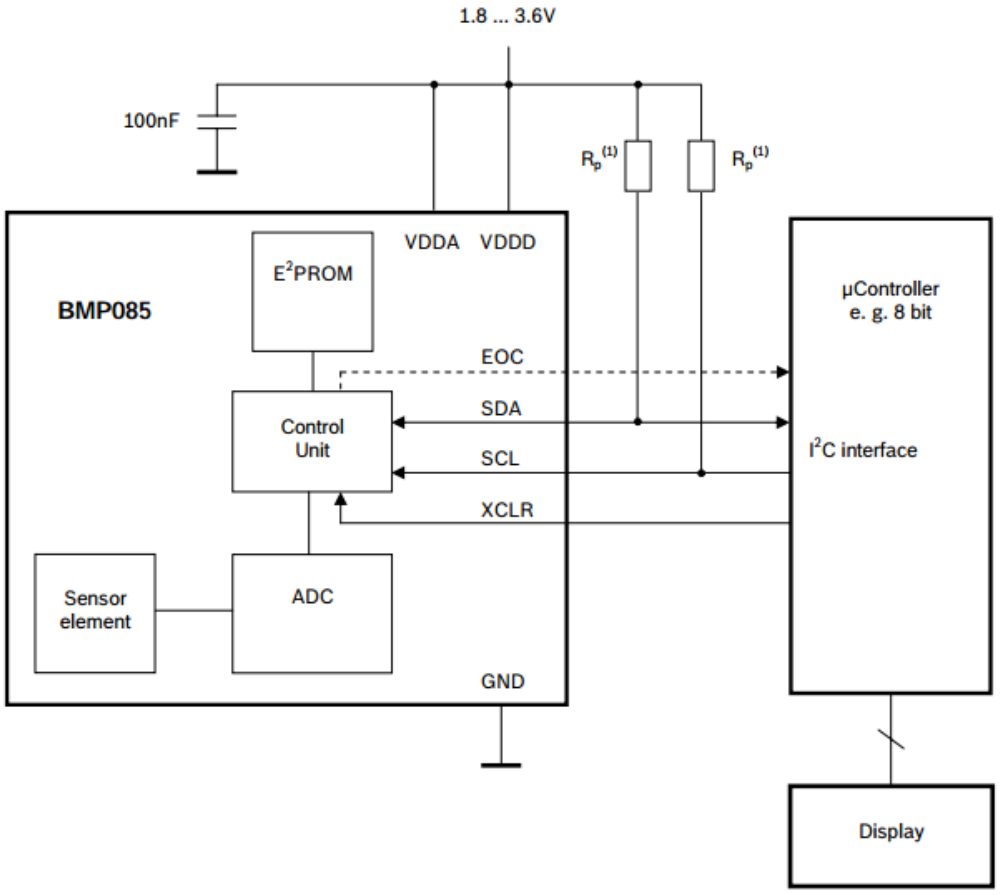
O BMP085 é um sensor desenvolvido pela Bosch Sensortec. Seu desempenho o faz adequado para o uso em dispositivos móveis, como *smart phones*, *tablets* e dispositivos esportivos.

Esse sensor promete baixo consumo de energia. De acordo com seu *datasheet*, quando alimentado com 2.5 V, a corrente consumida varia entre 3uA (em modo de economia de energia) e 12uA (em modo de alta resolução). Neste trabalho o sensor é configurado para trabalhar em modo de alta resolução, de acordo com o *datasheet* a precisão nesse modo de operação é de 0.03 hPa, o que corresponde a 0.25 metros[1]. Este sensor se comunica com o microcontrolador através do barramento I2C (*Inter-Integrated Circuit*).

[1] Ou seja, uma variação na pressão de 1hPa equivale a uma variação de 8.43m, no nível do mar.

O BMP085 consiste em um sensor piezo-resistivo, um conversor analógico/digital e uma unidade de controle com EEPROM e interface I2C. A EEPROM tem armazenados 176 bits de dados de calibração, que são utilizados para compensar as medidas de pressão em relação à temperatura e a outros parâmetros do sensor. A Figura 11 apresenta um diagrama funcional do sensor sendo utilizado em uma aplicação genérica.

Figura 11 - Diagrama esquemático do BMP085



Fonte: *Datasheet* do Sensor

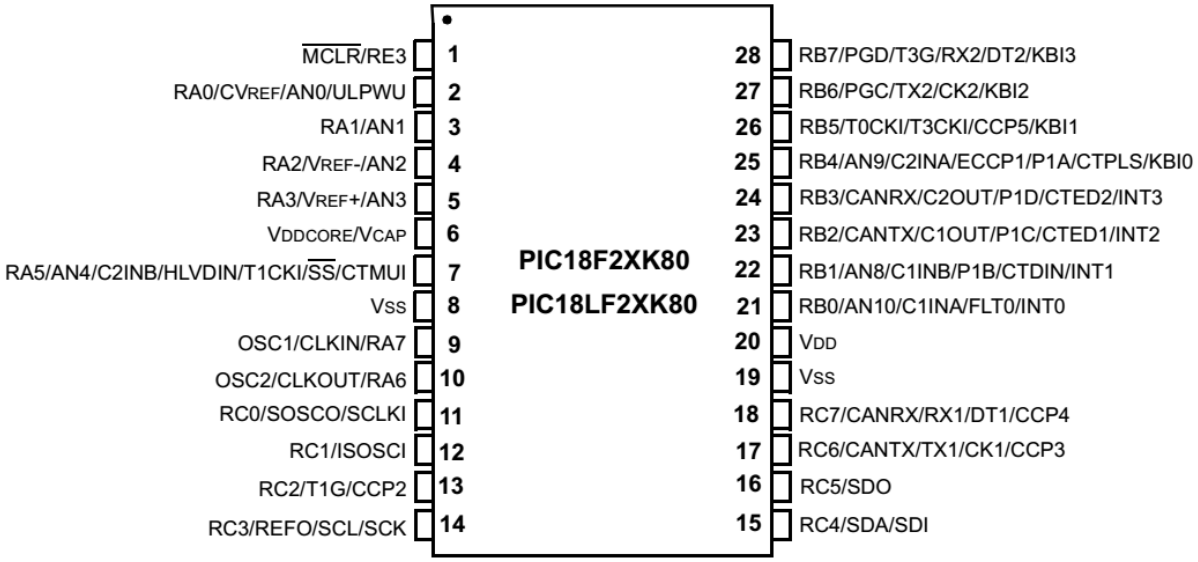
3.1.4 Microcontrolador

O microcontrolador utilizado neste trabalho foi o PIC18F26K80 desenvolvido pela Microchip, que possui arquitetura Harvard com memória flash de 64kbytes para armazenar o programa, uma RAM de 3648 bytes para execução do programa e uma EEPROM de 1024 bytes para armazenar dados. Esse PIC conta com o conjunto padrão de instruções PIC18

contendo 75 instruções, mais um conjunto de extensão de 8 instruções para otimização de código recursivo ou código que utilize uma pilha via *software* [8].
Além disso, o 18F26K80 conta com a tecnologia *nanoWatt*, que garante um baixo consumo de energia. A alimentação é feita a 3.3V sendo exigida uma corrente de, em média, 230nA. Para geração do *clock* foi usado um cristal ressonante de 4MHz.

A principal característica desse microcontrolador que o levou a ser escolhido para o desenvolvimento desse trabalho foi o fato de possuir duas UARTs, uma para se comunicar com o módulo GSM e outra para se comunicar com o módulo GPS. Além disso, possui comunicação SPI e I2C, a qual é usada para se comunicar com o sensor de pressão barométrica BMP085. O que foi descrito até aqui corresponde ao uso de 13 dos 28 pinos disponíveis. Os 15 pinos restantes são utilizados como portas I/O ou simplesmente não foram utilizados. A pinagem completa do microcontrolador pode ser vista na Figura 12. Na Figura 13 pode-se observar seu diagrama de blocos.

Figura 12 - Diagrama de pinos do PIC18F26K80

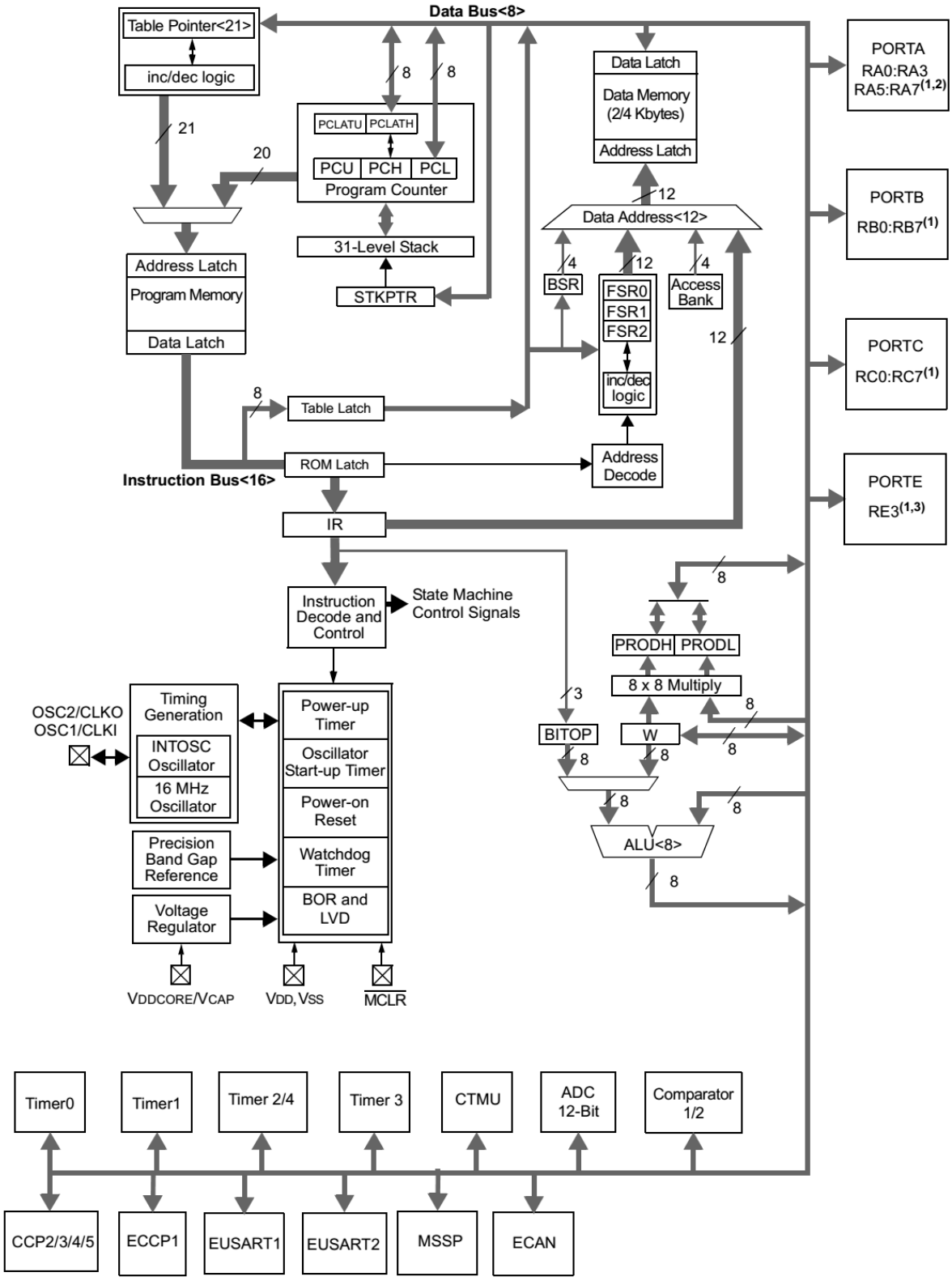


Fonte: Datasheet do PIC18F26K80

Esse microcontrolador apresenta outras funcionalidades não exploradas neste trabalho, como:

- 8 Conversores A/D de 12 bits;
- 4 Módulos PWM;
- 5 Timers;
- 2 Módulos Comparadores;
- 31 Fontes de Interrupção, sendo 4 por *hardware*.

Figura 13 - Diagrama de blocos do PIC 18F26K80



Fonte: Datasheet do PIC18F26K80

3.2 Descrição do *Hardware* Desenvolvido

O *hardware* desenvolvido consiste de um circuito que integra um regulador de tensão LM7805, o microcontrolador PIC18F26K80, o módulo GPS, o módulo GSM, um *display* LCD 20X4, o sensor de pressão BMP085, duas baterias de íon-lítio, um recarregador de baterias baseado no chip 4065E e dois botões para controlar o sistema, além de tudo o que é necessário para que os itens supracitados funcionem adequadamente. Na Figura 14 é apresentado o esquemático do *hardware*, desenvolvido no *software* Eagle 7.1.0 versão de estudante, que é gratuitamente distribuído em www.cadsoftusa.com, página do desenvolvedor (CadSoft Computer).

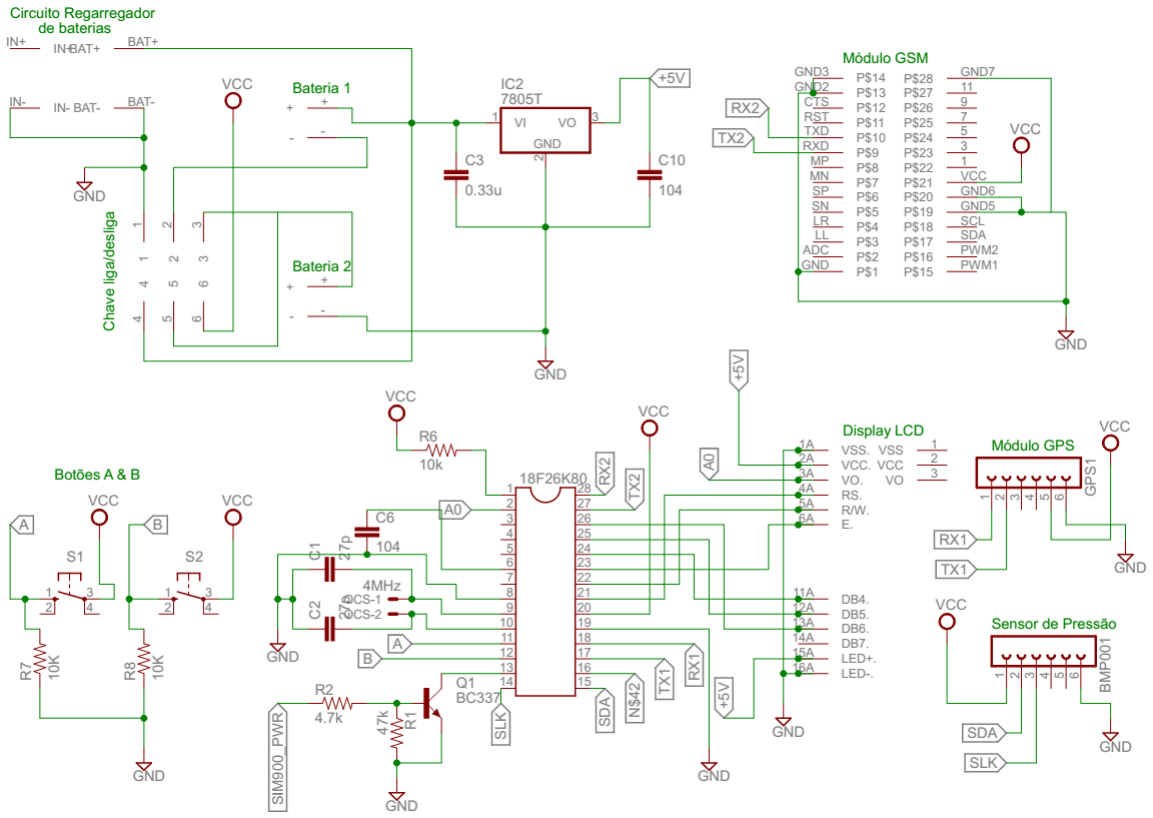
Na parte superior da Figura 14 está representado o circuito de acionamento e alimentação do sistema. A chave liga/desliga, quando pressionada, conecta o pino 2 ao pino 3 e também conecta o pino 5 ao pino 6, desconectando o pino 1 do pino 2 e o pino 4 do pino 5. A conexão entre os pinos 2 e 3 coloca as duas baterias de 3.7V em série, fornecendo uma tensão de 6.0V a 7.4V ao circuito regulador de tensão LM7805[2], dependendo do nível de carga das baterias. Na saída do regulador de tensão são fornecidos 5V somente ao LCD. A conexão entre os pinos 5 e 6 alimenta o restante do circuito com a tensão da bateria 2 através do pino VCC. Portanto, quando a chave liga/desliga está pressionada, o sistema está em funcionamento. Quando a chave é colocada na posição de repouso o circuito é desligado e as baterias colocadas em paralelo afim de que possam ser recarregadas. Nessa posição, a conexão entre os pinos 1 e 2 conecta o polo negativo da bateria 1 ao GND, e a conexão entre os pinos 4 e 5 conecta o polo positivo da bateria 2 ao polo positivo da bateria 1. Assim, quando um cabo USB é conectado ao recarregador de baterias o sistema é alimentado através de IN+ e IN- com os 5V da porta USB e fornece tensão e corrente constantes para alimentar as baterias.

No restante da Figura 14 estão representados os circuitos funcionais do sistema. Primeiro, a alimentação do PIC, seu circuito oscilador e sua conexão com o LCD, e logo abaixo as conexões dos botões, dos módulos GSM e GPS e do sensor de pressão.

[2] Observe que o LM7805 é um regulador de tensão linear e não chaveado, portanto de elevado consumo de energia, o que pode ser comprometedor para uma aplicação embarcada. O intuito deste trabalho é fazer uma prova de conceito, porém para uma versão profissional esse regulador deve ser, obrigatoriamente, substituído por uma regulador chaveado.

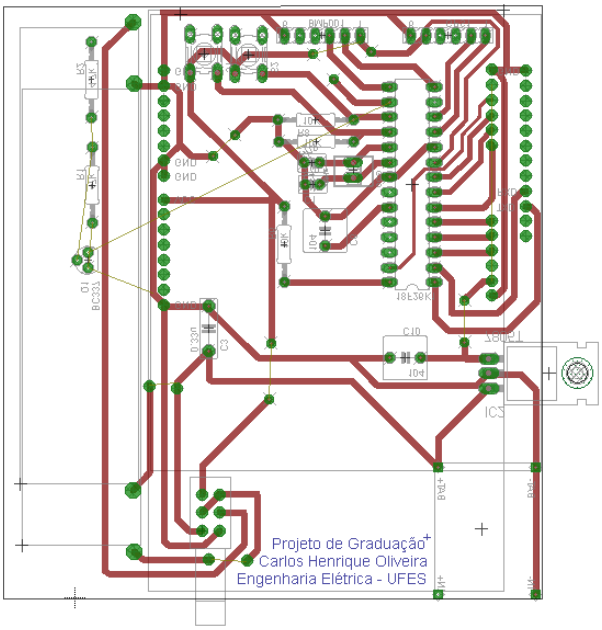
Na Figura 15 está o desenho da placa de circuito impresso desenvolvido. Este desenho foi gerado no mesmo programa em que foi feito o esquemático, descrito anteriormente. O projeto dessa placa foi pensado de maneira que o sistema ficasse o mais compacto possível. Assim, o sistema como um todo está dividido fisicamente em três níveis. No primeiro nível está o módulo GSM IcomSat v1.1. Através de *pin headers* o segundo nível, composto pela maioria dos componentes, se conecta ao nível 1. E, finalmente, o display LCD é conectado, também através de *pin headers*, ao segundo nível. Mais detalhes podem ser observados nas Figuras 16 e 17.

Figura 14 - Esquemático do hardware desenvolvido



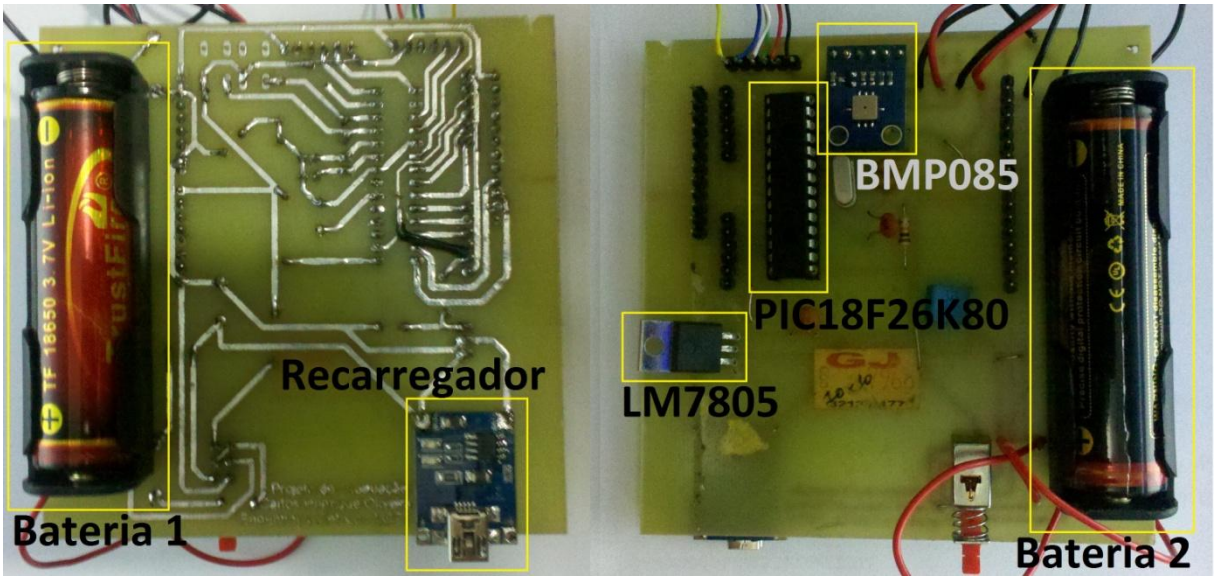
Fonte: Produção do próprio autor

Figura 15 - Desenho da placa de circuito impresso desenvolvida



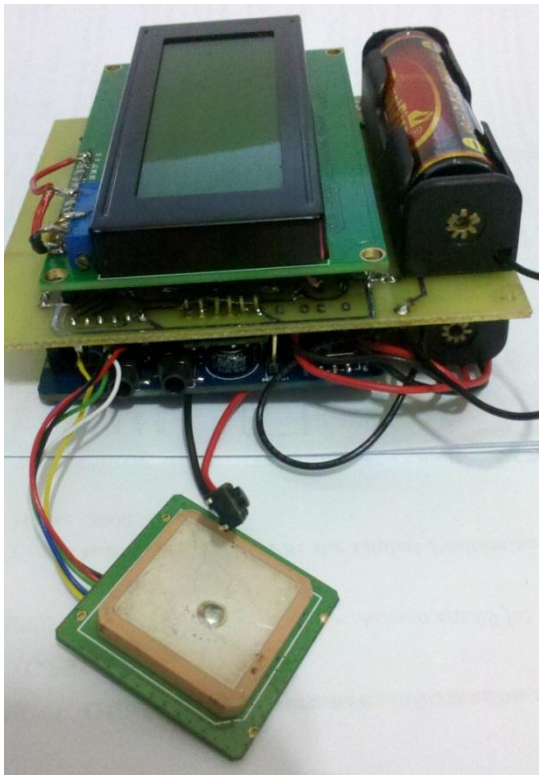
Fonte: Produção do próprio autor

Figura 16 - Foto do circuito desenvolvido. À direita a parte superior e à esquerda a parte inferior



Fonte: Produção do próprio autor

Figura 17 - Foto do sistema montado



Fonte: Produção do próprio autor

3.3 Descrição do *Firmware* Desenvolvido

O sistema desenvolvido possui três funções principais: mostrar ao usuário as informações de posicionamento, velocidade, temperatura, altitude, data e hora; permitir que o usuário salve na memória do dispositivo um *WayPoint*, que é composto de posição, data e hora atuais, de maneira amigável; e, permitir que o usuário envie um *WayPoint* para um dos números de celular salvo na agenda. As duas primeiras funções são executadas no Menu GPS (menu 1) do *firmware*, e a terceira é executada no Menu GSM (menu 2).

O *firmware* desenvolvido tem como objetivo oferecer ao microcontrolador uma interface com os módulos GPS e GSM, realizando todos os procedimentos necessários para isso, e controlar a interface do sistema com o usuário, o *display* LCD. Nas Figuras 18 e 19 está um fluxograma que explica a idéia básica de como o *firmware* funciona.

A inicialização do programa consiste de configurar, baseado na freqüência de oscilação do cristal, o *clock* do sistema (4MHz), configurar as UARTs (taxa de comunicação, pinos do microcontrolador para TX e RX), declarar variáveis, definir o comportamento das

interrupções de comunicação, configurar as portas I/O do microcontrolador, habilitar as interrupções de comunicação, ligar o modem GSM e testar sua comunicação, iniciar o *display* LCD e calibrar o sensor de pressão atmosférica.

Após a inicialização, o programa entra em um *loop* infinito, que é interrompido uma vez por segundo para tratar a interrupção gerada pela UART que recebe os dados do GPS. Nesse *loop* infinito o programa fica basicamente testando qual menu e qual submenu deve ser executado. Essa conferência é feita pelas funções *TestSubmenu()* e *TestMenu()*, que podem ser encontradas no Apêndice A deste trabalho. A função *TestMenu()* está representada no fluxograma pela pergunta "Botão A foi pressionado?", que está presente somente no início do submenu 1.1 e do submenu 2.1. A função *TestSubmenu()* aparece no fluxograma representada pela pergunta "Botão B foi pressionado?" e está presente no início de cada submenu.

Dentro de cada submenu são executadas funções específicas. O Submenu 1.1 chama a função *GPS()* que é responsável por receber as mensagens enviadas pelo módulo GPS e tratar esses dados, retirando de cada tipo de mensagem recebida a informação que se deseja e exibindo-a no *display*. Se, por acaso, não houver comunicação entre o módulo GPS e o microcontrolador, então uma mensagem na tela informa o problema ao usuário.

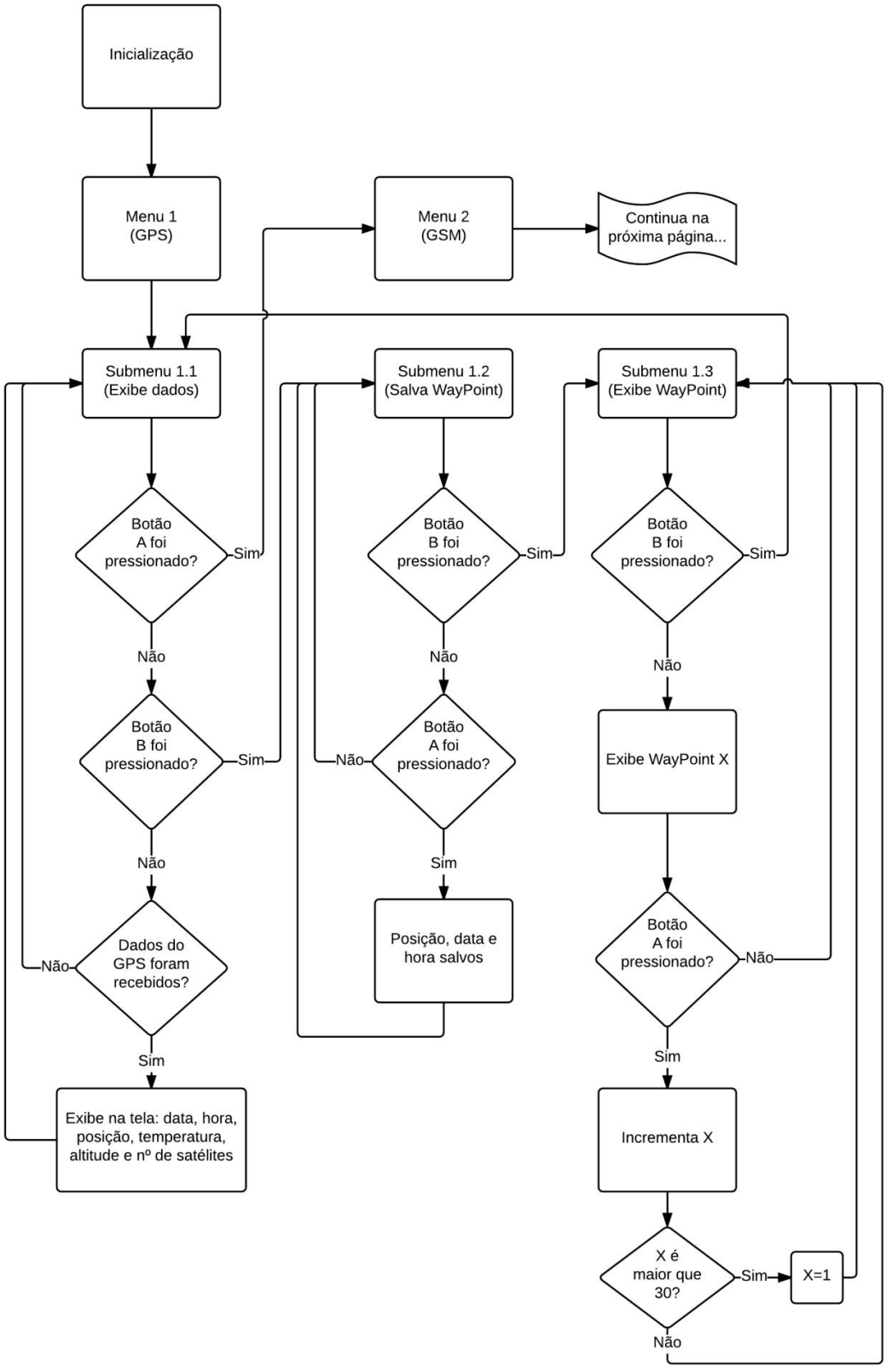
No Submenu 1.2 a função que é chamada é a *SalvaWayPoint()*. Essa função, assim como a *GPS()* só está disponível para o usuário se estiver havendo comunicação com o módulo GPS. O objetivo principal desta função é tratar os dados recebidos, retirando de cada tipo de mensagem a informação que se deseja, organizar numa estrutura essas informações e salvar o *WayPoint* na memória do microcontrolador.

No último submenu do menu 1, a função executada é a *VerWayPoint()*, que é responsável por mostrar a lista dos *WayPoints* salvos.

Como já dito anteriormente, é no Menu 2 que se envia um *WayPoint* através de uma mensagem SMS. No Submenu 2.1, o sistema aguarda que o usuário comece a escolher o conteúdo e o destino da mensagem. No Submenu 2.2 é escolhido o *WayPoint* a ser enviado, no Submenu 2.3 é escolhido o número de destino da mensagem e, finalmente, no Submenu 2.4 o usuário pode escolher confirmar o envio do SMS ou abortar a operação.

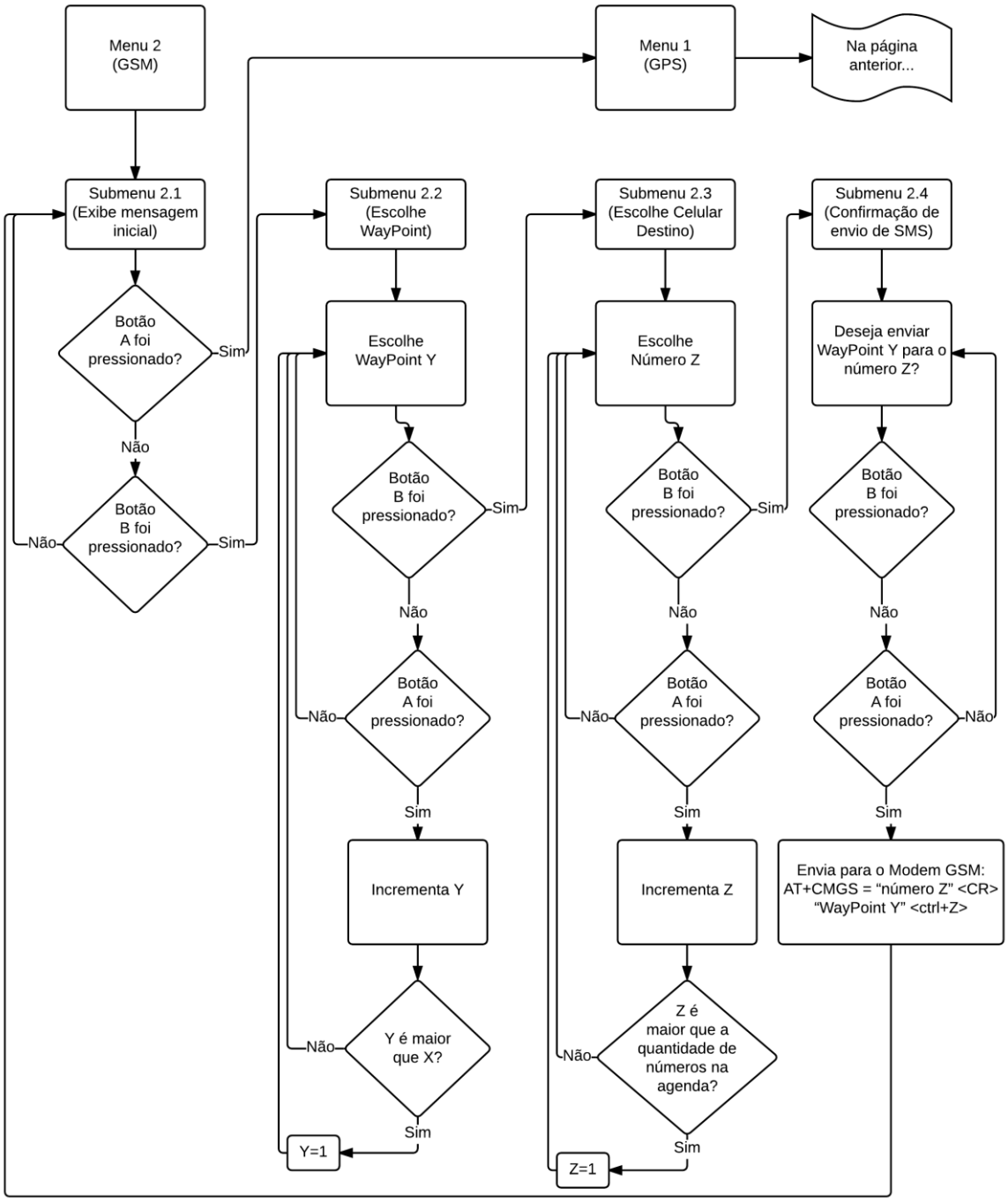
Todo o *software* aqui descrito está disposto no Apêndice A deste trabalho.

Figura 18 - Fluxograma do Firmware Desenvolvido



Fonte: Produção do próprio autor

Figura 19 - Fluxograma do *Firmware* Desenvolvido (continuação)

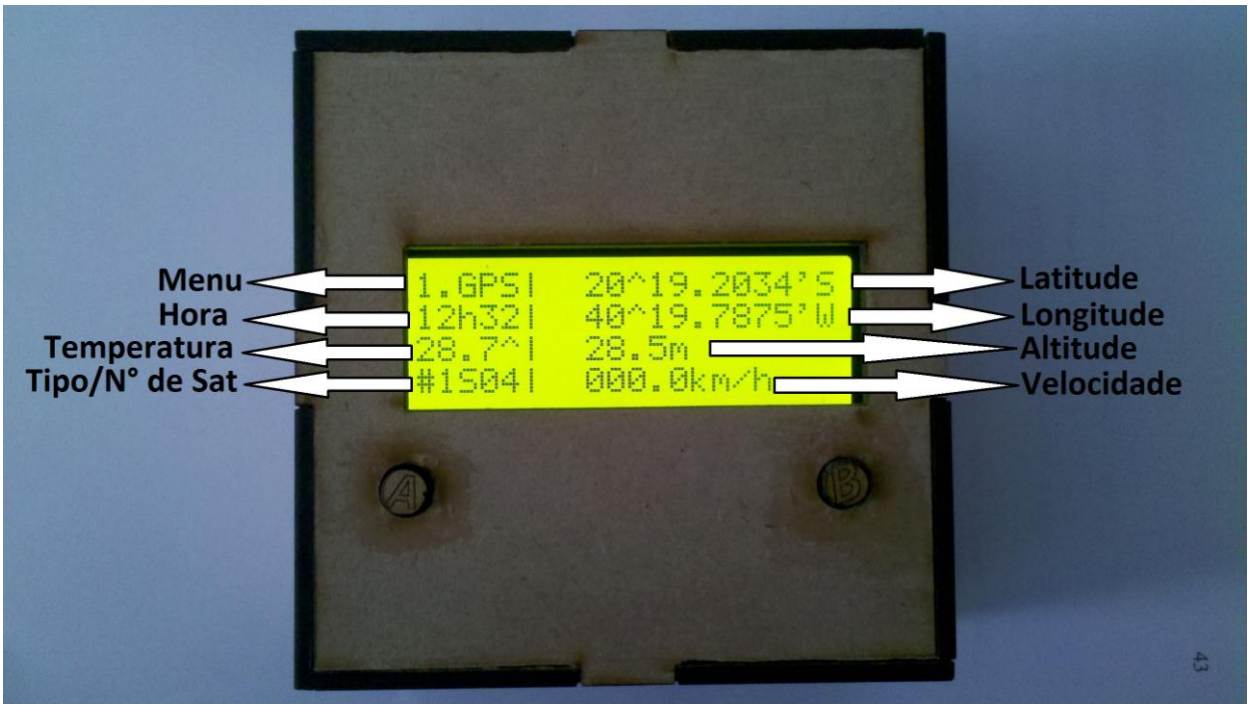


Fonte: Produção do próprio autor

4 TESTES E RESULTADOS

O circuito desenvolvido foi acomodado em uma caixa de MDF (*Medium Density Fiberboard*). Na Figura 20 está uma foto do dispositivo de rastreamento proposto em sua forma final. Nesta figura é possível ver a execução do Menu 1, em que são exibidas informações de latitude, longitude, altitude, velocidade, hora, temperatura, tipo de posicionamento (vide Tabela 2) e número de satélites GPS dos quais o dispositivo recebe sinais.

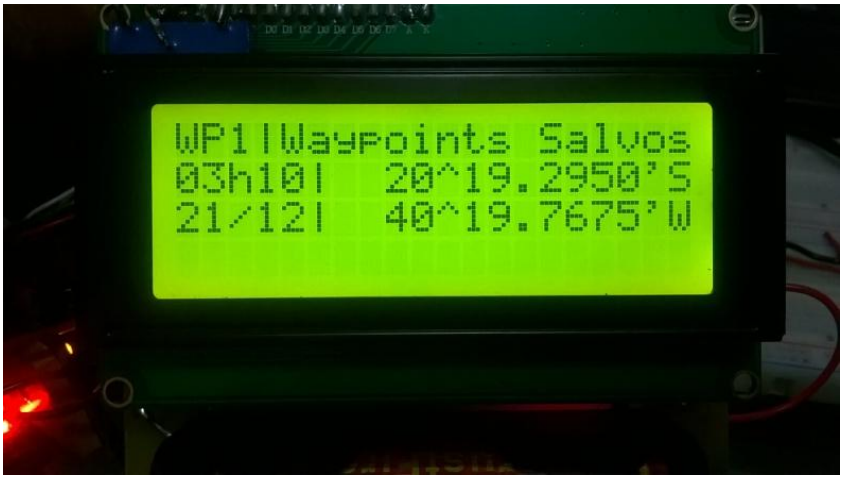
Figura 20 - Projeto Finalizado



Fonte: Produção do próprio autor

Os testes realizados compreendem ao envio de mensagem SMS pelo dispositivo contendo um *WayPoint* e à obtenção de uma sequência de *WayPoints* ao longo de um percurso. Inicialmente, foi salvo um WayPoint no dispositivo, conforme mostrado na Figura 21.

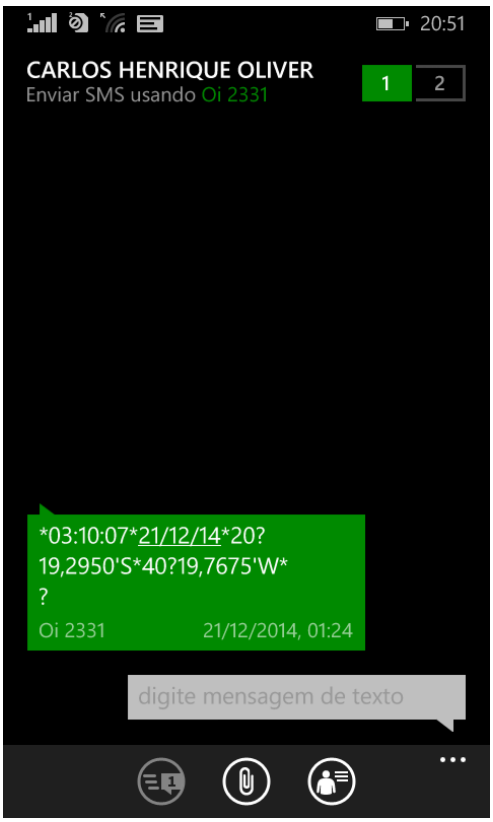
Figura 21 - Waypoint salvo no dispositivo



Fonte: Produção do próprio autor

Esse WayPoint salvo foi enviado para um número de celular através do menu GSM. Na Figura 22 pode-se ver a imagem da tela do celular que recebeu essa mensagem.

Figura 22 - SMS contendo o waypoint enviado



Fonte: Produção do próprio autor

Comparando as Figuras 21 e 22 é possível ver que o *WayPoint* com latitude 20°19.2950’S e longitude 40°19.7675’W, obtido às 03h10 do dia 21 de dezembro de 2014 (seguindo o horário

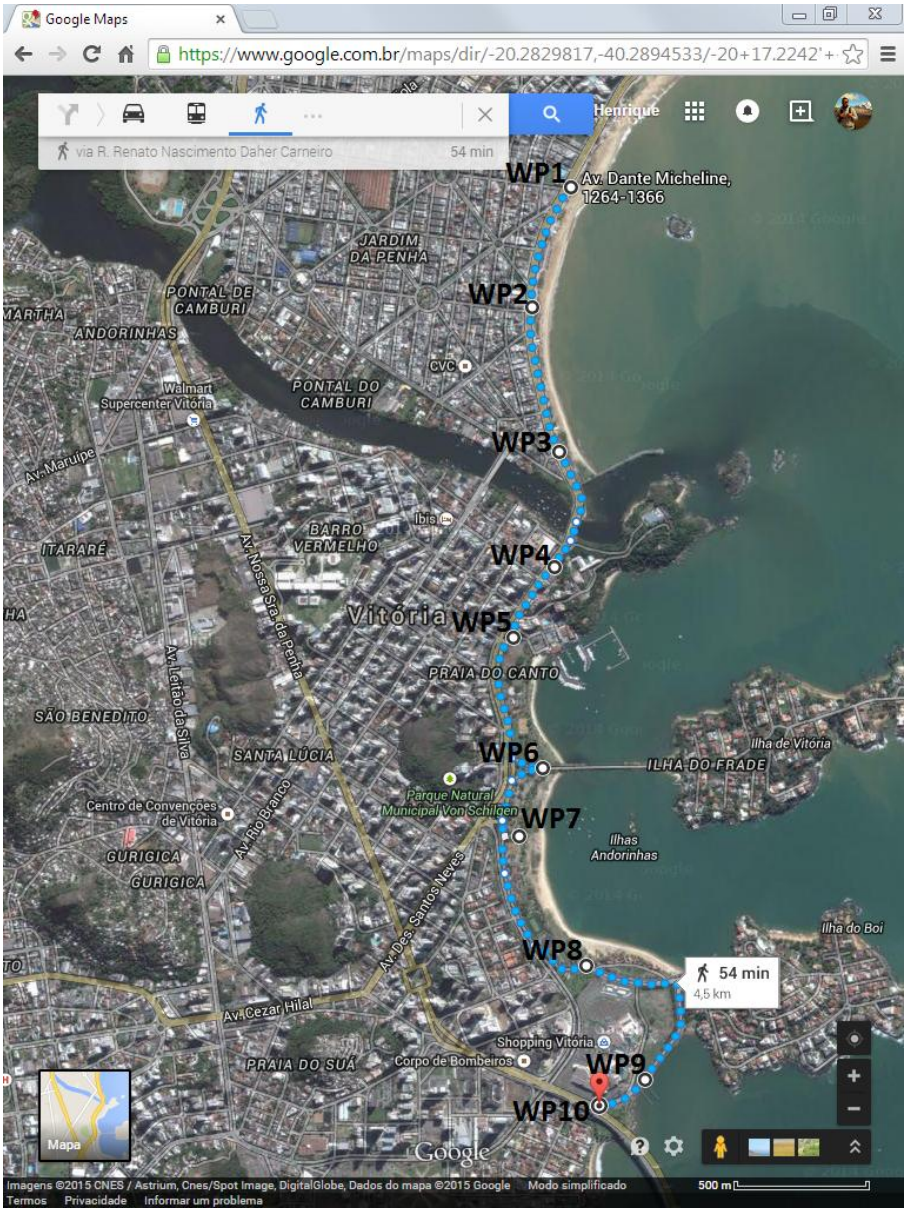
UTC, *Universal Time Coordinated*) foi enviado com sucesso pelo dispositivo. O que comprova que o sistema funcionou como desejado no que diz respeito ao envio de SMS.

O objetivo do segundo teste é comprovar a precisão e a confiabilidade dos dados obtidos pelo dispositivo. Para isso foram obtidos 10 *WayPoints* e, com o auxílio do Google Maps, esses pontos foram marcados no mapa. Os pontos obtidos foram os seguintes:

- WP1: Praia de Camburi/Clube dos Oficiais – 20°16.9789'S – 40°17.3672'W
- WP2: Praia de Camburi/Banco do Brasil – 20°17.2242'S – 40°17.4525'W
- WP3: Praia de Camburi/Estacionamento 1 – 20°17.5207'S – 40°17.3916'W
- WP4: Posto de gasolina/Ponte de Camburi – 20°17.7592'S – 40°17.4026'W
- WP5: Início da Praça dos Namorados – 20°17.9041'S – 40°17.4924'W
- WP6: Ponte da Ilha do Frade – 20°18.1712'S – 40°17.4275'W
- WP7: Praça dos Desejos – 20°18.3120'S – 40°17.4807'W
- WP8: Curva da Jurema/Início da ciclovia – 20°18.5776'S – 40°17.3321'W
- WP9: Curva da Jurema/Final da ciclovia – 20°18.5776'S – 40°17.3321'W
- WP10: Shopping Vitória/Pier – 20°18.8121'S – 40°17.2035'W

Na Figura 23 é possível ver o resultado da marcação dos pontos no mapa. Os pontos obtidos são representados por pontos brancos com um circulo em volta.

Figura 23 - Waypoints obtidos pelo dispositivo marcados no mapa



Fonte: www.google.com.br/maps

Os pontos obtidos pelo dispositivo parecem apresentar precisão dentro da faixa de erro de 5m especificada no *datasheet* do módulo GPS, porém mesmo que o erro seja um pouco maior o resultado obtido na Figura 23 é satisfatório para a aplicação pretendida.

5 CONCLUSÃO

Nesse projeto foi desenvolvido um dispositivo de rastreamento GPS (*Global Positioning System*) para ser utilizado na prática de esportes de aventura. O dispositivo desenvolvido funciona conforme foi proposto: é capaz de armazenar os *WayPoints* e enviá-los via mensagem SMS.

As funcionalidades apresentadas pelo dispositivo são úteis para um usuário que esteja praticando *trekking*, escalada, ciclismo, ou qualquer outra atividade física ao ar livre onde haja o risco de se perder. Caso isso ocorra, uma mensagem contendo as coordenadas GPS pode ser enviada para um usuário remoto. Essa informação pode ser de muita utilidade, principalmente quando um resgate em mata fechada se faz necessário. Além disso, o dispositivo apresenta características úteis para o monitoramento de atividades físicas, como velocidade, temperatura e altitude.

O foco dado às funcionalidades do dispositivo desenvolvido está em torno da prática de atividades ao ar livre, porém o dispositivo de rastreamento pode ser usado genericamente para outras finalidades, como segurança. Como exemplo, é possível citar o rastreamento de veículos roubados. Para este caso deverá ser acrescentada uma função de solicitação do posicionamento do dispositivo através de uma mensagem SMS.

No desenvolvimento de trabalhos futuros, algumas melhorias podem ser feitas no projeto do dispositivo apresentado. Essas melhorias devem levar em consideração a redução do consumo de energia e a redução do tamanho físico do dispositivo. No que diz respeito ao consumo de energia: substituição do LM7805 por um regulador de tensão chaveado e substituição do *display* LCD por um *display* LED. Em relação ao tamanho: utilização do modem GSM soldado diretamente no *hardware* desenvolvido, eliminando toda a estrutura do módulo GSM que não é utilizada; substituição das baterias utilizadas por baterias que ocupem menos espaço; e, utilização de um *display* mais compacto. Além dessas melhorias pode ser desenvolvida a comunicação do dispositivo diretamente com um computador, a fim de transferir os *WayPoints* salvos.

Além das melhorias de *hardware* citadas, pode ser interessante desenvolver uma opção no *firmware* que permita o funcionamento do dispositivo em modo automático. Dessa forma, os *WayPoints* poderiam ser salvos e enviados automaticamente para um usuário remoto. Isso permitiria que, por exemplo, caso o usuário local sofra um acidente na prática de uma atividade esportiva que o leve a fica inconsciente, o usuário remoto teria acesso às ultimas posições do dispositivo, facilitando o resgate mesmo que o usuário local não seja capaz de solicitar ajuda.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] U.S. GOVERNMENT. **Official U.S. Government information about the Global Positioning System (GPS) and related topics.**
Disponível em: <http://www.gps.gov/systems/gps/>. Acesso em:09 jul. 2014

[2] EL-RABBANY, A. **Introduction to GPS: the Global Positioning System.** 1ª Edição. Norwood: Artech House, 2002.

[3] HOFMANN-WELLENHOF, B; LICHTENEGGER, H; COLLINS, J. **Global Positioning System: Theory and Practice**. 1ª Edição. Rockville: Springer-Verlag Wien, 1992.

[4] LOGSDON, T. **Understanding the Navstar GPS, GIS, and IVHS.** 2ª Edição. Estados Unidos: Chapman & Hall, 1995.

[5] ROCHA, C. **Geoprocessamento: Tecnologia Transdisciplinar.** Edição do autor. Juiz de Fora, MG: Ed. do Autor, 2000.

[6] GARG, V; WILKES, J. **Principles and Applications of GSM.** 3ª Edição. India: Pearson Education, 2004.

[7] ME COMPONENTES. **ME-1000RW, 65 channels with ultra-high sensitive Smart GPS Antenna module: Technical Data Sheet.** Versão 1.2.

[8] MICROCHIP. **PIC18F66K80 Family Data Sheet.** Microchip Technology Inc., 2001.

APÊNDICE A

```
//Firmware do Dispositivo de Rastreamento GPS com Comunicação GSM
//Carlos Henrique Oliveira de Oliviera
//Dezembro de 2014
#include <18f26k80.h>
#include <string.h>
#fuses HSM
#use delay(clock=16000000)

#use standard_io(a)
#use standard_io(b)
#use fast_io(c)
#include <LCD4_20.c>
#include <BMP085.c>
#include <math.h>

#use rs232(baud=9600, rcv=pin_c7, xmit=pin_c6, parity=n,
stream=gps_data)
#use rs232(baud=9600, rcv=pin_b7, xmit=pin_b6, parity=n,
stream=gsm_data)

void GPS(void);
void BMP085(void);
void SalvaWayPoint(void);
void VerWayPoint(void);
void SelecionaWaypoint(void);
void SelecionaNumero(void);
void EnviaSMS(void);
void DisplayMenu(void);
void TestMenu(int1);
void TestSubmenu(int1);

int menu=1;
int submenu=1;
int1 modo=0;

struct wp{
   char lati[9];
   char lonj[9];
   char data[6];
   char hora[6];
}  waypoint[50];

char numero[14];
char GpsData[150];

char msg[50];
int i=0;
int j=0;
int index_salvar=0;
int index_ver=1;
```

```
int index_seleciona=1;
int index_numero=0;
int1 wps=0;//wps=0, WP não salvo. wps=1, WP salvo.
int1 wps2=0;
int1 c0ouc1=0;


float temperatura=0;
float pressao=0;
float P_zero=1017.18;
float f = 5.225;
float alt=0,a=0,b=0;

#INT_RDA
void RDA_isr()
   {
      i++;
      GpsData[i]=getc(gps_data);
   }

#INT_RDA2
void RDA2_isr()
   {
      j++;
      msg[j]=getc(gsm_data);
   }

void main(void){
   set_tris_a(0b11111111);
   set_tris_b(0b11111111);
   set_tris_c(0b11111011);

   enable_interrupts(GLOBAL);
   enable_interrupts(INT_RDA);
   enable_Interrupts(INT_RDA2);


   lcd_init();
   printf(lcd_putc,"\fProjeto de Graduacao");
   printf(lcd_putc,"\nEngenharia Eletrica");
   printf(lcd_putc,"\nCarlos Henrique Oli");
   printf(lcd_putc,"\nveira de Oliveira");
   output_high(pin_c2);
   delay_ms(3000);
   output_low(pin_c2);
   //bmp085Calibration();
   delay_ms(1000);



while(true)
   {
//MENU1-------------------------------------------------------------
   DisplayMenu();
```

```
   while(menu==1)
      {
      if(submenu==1)
         {
         GPS();
         TestSubmenu(0);
         TestMenu(0);
         }

      if(submenu==2)
         {
         SalvaWayPoint();
         TestSubmenu(0);
         }

      if(submenu==3)
         {
         VerWayPoint();
         TestSubmenu(1);
         }
      }

//MENU2-----------------------------------------------------------------
   DisplayMenu();
   while(menu==2)
      {
      if(submenu==1)
         {
         printf(lcd_putc,"\f2.GSM | Envio de WP");
         delay_ms(100);
         TestSubmenu(0);
         TestMenu(0);
         }
      if(submenu==2)
         {
         SelecionaWaypoint();
         TestSubmenu(0);
         }

      if(submenu==3)
         {
         SelecionaNumero();
         TestSubmenu(0);
         }
      if(submenu==4)
         {
         EnviaSMS();
         TestSubmenu(1);
         }
      }

//MENU3-----------------------------------------------------------------
   DisplayMenu();
   while(menu==3)
      {
```

```
      if(submenu==1)
         {
         printf(lcd_putc("\f3.Selecione o modo\nde operacao"));
         delay_ms(100);
         TestSubmenu(0);
         TestMenu(1);
         }
      if(submenu==2)
         {
         lcd_gotoxy(1,3);
         if(modo==0)
            {
            printf(lcd_putc,"Modo Manual         ");
            if(input(pin_c0))
               modo=1;
            while(input(pin_c0))
               delay_ms(50);
            }
         if(modo==1)
            {
            printf(lcd_putc,"Modo Automatico     ");
            if(input(pin_c0))
               modo=0;
            while(input(pin_c0))
               delay_ms(50);
            }
         delay_ms(100);
         TestSubmenu(1);
         }

      }


   }



}


void GPS(void)
   {
      if(GpsData[i]=='$')//garante que está havendo comunicação com
o gps
         {
         i=0;
         if(GpsData[3]=='G')//lê somente os dados da GGA
            {
            lcd_gotoxy(1,1);
            printf(lcd_putc,"1.GPS|
%c%c^%c%c.%c%c%c%c'%c",GpsData[18],GpsData[19],GpsData[20],GpsData[2
1],GpsData[23],GpsData[24],GpsData[25],GpsData[26],GpsData[28]);
            lcd_gotoxy(1,2);
            printf(lcd_putc,"%c%ch%c%c|
%c%c^%c%c.%c%c%c%c'%c",GpsData[7],GpsData[8],GpsData[9],GpsData[10],
```

```
GpsData[31],GpsData[32],GpsData[33],GpsData[34],GpsData[36],GpsData[
37],GpsData[38],GpsData[39],GpsData[41]);
            BMP085();
            lcd_gotoxy(1,4);
            printf(lcd_putc,"#%cS%c%c| %3.1g
%fm",GpsData[43],GpsData[45],GpsData[46], temperatura, alt);
            }
         if(GpsData[3]=='Z')//lê somente os dados da ZDA
            {
            lcd_gotoxy(1,3);

printf(lcd_putc,"%c%c/%c%c|",GpsData[18],GpsData[19],GpsData[21],Gps
Data[22]);
            }
         if(GpsData[3]=='V')//lê somente os dados da VTG
            {
            lcd_gotoxy(7,3);
            printf(lcd_putc,"  %c%c%c%c%ckm/h ", GpsData[26],
GpsData[27],GpsData[28],GpsData[29],GpsData[30]);
            }
         }

   }

void BMP085(void)
   {
    a=0;
    b=0;
    temperatura = BMP085Temperature();
    pressao = BMP085Pressure(false);
    a = pressao/P_zero;
    f = 1/f;
    b = pow(a,f);
    alt = 44330*(1-b);

   }


void SalvaWayPoint(void)
   {
   if(GpsData[i]=='$')
      {
      i=0;
      if(GpsData[3]=='G')
         {
         printf(lcd_putc,"\fPara gravar Waypoint\npressione A por
2sg");
         if(input(pin_c0))
            {
            while(input(pin_c0))
               {
               delay_ms(50);
               }
            if(index_salvar>49   )
               {
```

```
                while(c0ouc1==0)
                   {
                   if((input(pin_c1)==1) || (input(pin_c0==1)))
                      c0ouc1=1;
                   printf(lcd_putc,"\f   Memoria cheia!   \nPara
sobreescrever");
                   printf(lcd_putc,"\npressione A. Para\nsair
pressione B");
                   delay_ms(100);
                   }
               if(input(pin_c0))
                   {
                   while(input(pin_c0))
                      delay_ms(50);
                   index_salvar=0;
                   }
               if(input(pin_c1))
                   {
                   while(input(pin_c1))
                      delay_ms(50);
                   wps=1;//evita que o último WP seja sobreescrito

                   }
                }
             printf(lcd_putc,"\fSalvando Waypoint...");
             while(wps==0)//Enquanto WP não for salvo, GpsData
procura o início da mensagem, "$"
                {
                if(GpsData[i]=='$')
                   {
                   i=0;
                   if(GpsData[3]=='G')
                      {
                      index_salvar++;

                      waypoint[index_salvar].lati[0]=GpsData[18];
                      waypoint[index_salvar].lati[1]=GpsData[19];
                      waypoint[index_salvar].lati[2]=GpsData[20];
                      waypoint[index_salvar].lati[3]=GpsData[21];
                      waypoint[index_salvar].lati[4]=GpsData[23];
                      waypoint[index_salvar].lati[5]=GpsData[24];
                      waypoint[index_salvar].lati[6]=GpsData[25];
                      waypoint[index_salvar].lati[7]=GpsData[26];
                      waypoint[index_salvar].lati[8]=GpsData[28];

                      waypoint[index_salvar].lonj[0]=GpsData[31];
                      waypoint[index_salvar].lonj[1]=GpsData[32];
                      waypoint[index_salvar].lonj[2]=GpsData[33];
                      waypoint[index_salvar].lonj[3]=GpsData[34];
                      waypoint[index_salvar].lonj[4]=GpsData[36];
                      waypoint[index_salvar].lonj[5]=GpsData[37];
                      waypoint[index_salvar].lonj[6]=GpsData[38];
                      waypoint[index_salvar].lonj[7]=GpsData[39];
                      waypoint[index_salvar].lonj[8]=GpsData[41];
```

```
                     wps2=1;//garante que a informação da ZDA vai
                            //ser salva no index_salvar certo
                     }
                  if((GpsData[3]=='Z')&&(wps2==1))
                     {
                     waypoint[index_salvar].data[0]=GpsData[18];
                     waypoint[index_salvar].data[1]=GpsData[19];
                     waypoint[index_salvar].data[2]=GpsData[21];
                     waypoint[index_salvar].data[3]=GpsData[22];
                     waypoint[index_salvar].data[4]=GpsData[26];
                     waypoint[index_salvar].data[5]=GpsData[27];

                     waypoint[index_salvar].hora[0]=GpsData[7];
                     waypoint[index_salvar].hora[1]=GpsData[8];
                     waypoint[index_salvar].hora[2]=GpsData[9];
                     waypoint[index_salvar].hora[3]=GpsData[10];
                     waypoint[index_salvar].hora[4]=GpsData[11];
                     waypoint[index_salvar].hora[5]=GpsData[12];

                     printf(lcd_putc,"\f Waypoint %i salvo!",
index_salvar);
                     delay_ms(1000);

                     submenu=1;//sai do submenu de gravar waypoint
                     wps=1;//determina o fim deste while, pois WP já
foi salvo
                     wps2=0;
                     }
                  }
               }
               wps=0;//reset de wps para salvar o próximo WP



            }
         }
      }
   }

void VerWayPoint(void)
   {
   if(index_salvar==0)
      printf(lcd_putc,"\fNao ha Waypoints\nsalvos...");
   else
      {
      printf(lcd_putc,"\fWP%i|Waypoints Salvos",index_ver);
      printf(lcd_putc,"\n%c%ch%c%c|  %c%c^%c%c.%c%c%c%c'%c",
waypoint[index_ver].hora[0], waypoint[index_ver].hora[1],
waypoint[index_ver].hora[2], waypoint[index_ver].hora[3],
waypoint[index_ver].lati[0], waypoint[index_ver].lati[1],
waypoint[index_ver].lati[2], waypoint[index_ver].lati[3],
waypoint[index_ver].lati[4], waypoint[index_ver].lati[5],
waypoint[index_ver].lati[6], waypoint[index_ver].lati[7],
waypoint[index_ver].lati[8]);
```

```
      printf(lcd_putc,"\n%c%c/%c%c|  %c%c^%c%c.%c%c%c%c'%c",
waypoint[index_ver].data[0], waypoint[index_ver].data[1],
waypoint[index_ver].data[2], waypoint[index_ver].data[3],
waypoint[index_ver].lonj[0], waypoint[index_ver].lonj[1],
waypoint[index_ver].lonj[2],waypoint[index_ver].lonj[3],
waypoint[index_ver].lonj[4], waypoint[index_ver].lonj[5],
waypoint[index_ver].lonj[6], waypoint[index_ver].lonj[7],
waypoint[index_ver].lonj[8]);
      if(input(pin_c0))
         {
         while(input(pin_c0))
            delay_ms(50);
         index_ver++;
         if(index_ver>index_salvar)
            index_ver=1;
         }
      }
   delay_ms(100);
   }

void SelecionaWaypoint()
   {
   lcd_gotoxy(1,2);
   if(index_salvar==0)
      {
      printf(lcd_putc,"Nao ha WPs salvos");
      }
   else
      {
      printf(lcd_putc,"Enviar Waypoint: %i", index_seleciona);
      if(input(pin_c0))
         index_seleciona++;
      while(input(pin_c0))
         delay_ms(50);
      if(index_seleciona>index_salvar)
            index_seleciona = 1;
      }
   delay_ms(100);
   }

void SelecionaNumero()
   {
   lcd_gotoxy(1,3);
   switch(index_numero)
      {
      case 0:

      printf(lcd_putc,"Para: (27)999659594");
      numero = "+5527999659594";
      break;

      case 1:
      printf(lcd_putc,"Para: (27)988372331");
      numero = "+5527988372331";
      break;
```

```
      case 2:
      printf(lcd_putc,"Para: (27)988372333");
      numero = "+5527988372333";
      break;

      case 3:
      printf(lcd_putc,"Para: (27)988372334");
      numero = "+5527988372334";
      break;

      case 4:
      printf(lcd_putc,"Para: vazio!       ");
      numero = "0";
      break;
      }
      if(input(pin_c0))
         index_numero++;
      while(input(pin_c0));
         delay_ms(50);
      if(index_numero>4)
         index_numero=0;
      delay_ms(100);
   }

void EnviaSMS(void)
   {
   printf(lcd_putc,"\fPressione A para\nenviar Waypoint %i\npara
%c%c%c%c%c%c%c%c%c%c%c%c%c", index_seleciona, numero[0],
numero[1], numero[2], numero[3], numero[4], numero[5], numero[6],
numero[7], numero[8], numero[9], numero[10], numero[11], numero[12],
numero[13]);
   if(input(pin_c0))
      {
      while(input(pin_c0))
         delay_ms(50);
      printf(lcd_putc,"\fEnviando Mensagem...");
      fprintf(gsm_data,"AT+CMGS=\"%c%c%c%c%c%c%c%c%c%c%c%c%c\"\r",
numero[0], numero[1], numero[2], numero[3], numero[4], numero[5],
numero[6], numero[7], numero[8], numero[9], numero[10], numero[11],
numero[12], numero[13]);
      delay_ms(2000);

      putc('*',gsm_data);
      putchar(waypoint[index_seleciona].hora[0],gsm_data);
      putchar(waypoint[index_seleciona].hora[1],gsm_data);
      putc(':',gsm_data);
      putchar(waypoint[index_seleciona].hora[2],gsm_data);
      putchar(waypoint[index_seleciona].hora[3],gsm_data);
      putc(':',gsm_data);
      putchar(waypoint[index_seleciona].hora[4],gsm_data);
      putchar(waypoint[index_seleciona].hora[5],gsm_data);

      putc('*',gsm_data);
```

```
      putchar(waypoint[index_seleciona].data[0],gsm_data);
      putchar(waypoint[index_seleciona].data[1],gsm_data);
      putc('/',gsm_data);
      putchar(waypoint[index_seleciona].data[2],gsm_data);
      putchar(waypoint[index_seleciona].data[3],gsm_data);
      putc('/',gsm_data);
      putchar(waypoint[index_seleciona].data[4],gsm_data);
      putchar(waypoint[index_seleciona].data[5],gsm_data);

      putc('*',gsm_data);

      putchar(waypoint[index_seleciona].lati[0],gsm_data);
      putchar(waypoint[index_seleciona].lati[1],gsm_data);
      putc('^',gsm_data);
      putchar(waypoint[index_seleciona].lati[2],gsm_data);
      putchar(waypoint[index_seleciona].lati[3],gsm_data);
      putc(',',gsm_data);
      putchar(waypoint[index_seleciona].lati[4],gsm_data);
      putchar(waypoint[index_seleciona].lati[5],gsm_data);
      putchar(waypoint[index_seleciona].lati[6],gsm_data);
      putchar(waypoint[index_seleciona].lati[7],gsm_data);
      putc('\'',gsm_data);
      putchar(waypoint[index_seleciona].lati[8],gsm_data);

      putc('*',gsm_data);

      putchar(waypoint[index_seleciona].lonj[0],gsm_data);
      putchar(waypoint[index_seleciona].lonj[1],gsm_data);
      putc('^',gsm_data);
      putchar(waypoint[index_seleciona].lonj[2],gsm_data);
      putchar(waypoint[index_seleciona].lonj[3],gsm_data);
      putc(',',gsm_data);
      putchar(waypoint[index_seleciona].lonj[4],gsm_data);
      putchar(waypoint[index_seleciona].lonj[5],gsm_data);
      putchar(waypoint[index_seleciona].lonj[6],gsm_data);
      putchar(waypoint[index_seleciona].lonj[7],gsm_data);
      putc('\'',gsm_data);
      putchar(waypoint[index_seleciona].lonj[8],gsm_data);

      putc('*',gsm_data);

      puts("\n\f",gsm_data);
      delay_ms(2000);
      putchar(0x1A,gsm_data);
      while(input(pin_c0))
         delay_ms(50);
      printf(lcd_putc,"\fMensagem enviada!!!");
      delay_ms(500);
      submenu = 1;
      }
   delay_ms(300);
   //j++;
   }

void DisplayMenu(void)
```

```
   {
    printf(lcd_putc,"\f|------------------|");
    if(menu==1)
      printf(lcd_putc,"\n|      1.GPS       |");
    if(menu==2)
      printf(lcd_putc,"\n|      2.GSM       |");
    if(menu==3)
      printf(lcd_putc,"\n|3.MODO DE OPERACAO|");
    printf(lcd_putc,"\n|------------------|");
    printf(lcd_putc,"\n                    ");
    //while(input(pin_c0))
      delay_ms(1000);
      if(input(pin_c0))
         delay_ms(50);
      printf(lcd_putc,"\f");
   }


void TestMenu(int1 ultimo_menu)
   {
   if(input(pin_c0))
      {
      if(ultimo_menu==1)
         menu=1;
      else
         menu=menu+1;
      }
   }


void TestSubmenu(int1 ultimo_submenu)
   {
   if(input(pin_c1))
      {
      if(ultimo_submenu==1)
         submenu=1;
      else
         submenu=submenu+1;

      //delay_ms(50);
      while(input(pin_c1))
         delay_ms(50);
      }
   }
```

ANEXO 1

```
//***********************************************
//  BMP085 Barometric Pressure Sensor
//  Al Testani
//  08/17/12
//***********************************************

// place a #use i2c statement in the main program and comment this
out if not applicable
#use i2c(master, sda=PIN_C4, scl=PIN_C3, FAST, FORCE_HW)

#include <math.h>

const int8 OVS_S = 3; // Oversampling Setting (0,1,2,3 from ultra
low power, to ultra hi-resolution)

#define BMP085_ADDRESS 0xEE          // I2C address of BMP085
#define P_CORRECTION   1.5           // in mBars - factor to adjust
for elevation to match local weather station pressure



// Calibration values
signed int16 ac1;
signed int16 ac2;
signed int16 ac3;
int16 ac4;
int16 ac5;
int16 ac6;
signed int16 b1;
signed int16 b2;
signed int16 mb;
signed int16 mc;
signed int16 md;

// floating point cal factors
float _c3;
float _c4;
float _b1;
float _c5;
float _c6;
float _mc;
float _md;

// polynomomial constants
float _x0;
float _x1;
float _x2;
```

```
float _y0;
float _y1;
float _y2;
float _p0;
float _p1;
float _p2;

float _s;     // T-25, used in pressure calculation - must run
temperature reading before pressure reading
float _Temp;  // set after every temperature or temperature/pressure
reading


//-----------------------------------------------
int8 BMP085ReadByte(int8 address)
//-----------------------------------------------
{
int8 data;

   i2c_start();
   i2c_write(BMP085_ADDRESS);
   i2c_write(address);
   i2c_start();
   i2c_write(BMP085_ADDRESS | 0x01 );
   data=i2c_read(0);
   i2c_stop();
   return(data);
}


//-----------------------------------------------
int16 BMP085ReadInt(int8 address)
//-----------------------------------------------
{
int8 msb, lsb;
int16 temp;

   i2c_start();
   i2c_write(BMP085_ADDRESS);
   i2c_write(address);
   i2c_start();
   i2c_write(BMP085_ADDRESS | 0x01 );
   msb = i2c_read();
   lsb = i2c_read(0);
   i2c_stop();
   temp = make16(msb, lsb);
   return ( temp );
}


//-----------------------------------------------
void BMP085WriteByte(int8 address, int8 data)
//-----------------------------------------------
{
   i2c_start();
```

```
   i2c_write(BMP085_ADDRESS);
   i2c_write(address);
   i2c_write(data);
   i2c_stop();
}


//-----------------------------------------------
void bmp085Calibration()
//-----------------------------------------------
{
   // read BMP085 EEPROM cal factors
   ac1 = bmp085ReadInt(0xAA);
   ac2 = bmp085ReadInt(0xAC);
   ac3 = bmp085ReadInt(0xAE);
   ac4 = bmp085ReadInt(0xB0);
   ac5 = bmp085ReadInt(0xB2);
   ac6 = bmp085ReadInt(0xB4);
   b1  = bmp085ReadInt(0xB6);
   b2  = bmp085ReadInt(0xB8);
   mb  = bmp085ReadInt(0xBA);
   mc  = bmp085ReadInt(0xBC);
   md  = bmp085ReadInt(0xBE);

    // calculate floating point cal factors
   _c3 = 0.0048828125 * ac3;            // 160 * pow2(-15) * ac3;
   _c4 = 0.000000030517578125 * ac4;    // 1E-3 * pow2(-15) * ac4;
   _c5 = 0.00000019073486328125 * ac5;  // (pow2(-15)/160) * ac5;
   _c6 = (float)ac6;
   _b1 = 0.00002384185791015625 * b1;   // 25600 * pow2(-30) * b1;
   _mc = 0.08 * mc;                     // (pow2(11) / 25600) * mc;
   _md = (float)md / 160;

   // calculate polynomial constants
   _x0 = (float)ac1;
   _x1 = 0.01953125 * ac2;             // 160 * pow2(-13) * ac2;
   _x2 = 0.000762939453125 * b2;       // 25600 * pow2(-25) * b2;
   _y0 = _c4 * 32768;                  //_c4 * pow2(15);
   _y1 = _c4 * _c3;
   _y2 = _c4 * _b1;
   _p0 = 2.364375;
   _p1 = 0.992984;
   _p2 = 0.000004421;
}


// Read the uncompensated temperature value
//-----------------------------------------------
int16 BMP085ReadUT()
//-----------------------------------------------
{
int16 ut;

  // Write 0x2E into Register 0xF4
  BMP085WriteByte(0xF4, 0x2E);
```

```
  delay_ms(5); // Wait at least 4.5ms
  // Read two bytes from registers 0xF6 and 0xF7
  ut = BMP085ReadInt(0xF6);
  return((float)ut);
}


// Read the uncompensated pressure value
//-----------------------------------------------
int32 bmp085ReadUP()
//-----------------------------------------------
{
int8 msb, lsb, xlsb;
float p;

  // Write 0x34+(OSS<<6) into register 0xF4
  // Request a pressure reading w/ oversampling setting
  BMP085WriteByte(0xF4, (0x34 + (OVS_S<<6)) );

  // Wait for conversion, delay time dependent on OSS
  switch (OVS_S)
  {
     case 0: delay_ms(5);  break;
     case 1: delay_ms(8);  break;
     case 2: delay_ms(14); break;
     case 3: delay_ms(26); break;
  }

  // Read register 0xF6 (MSB), 0xF7 (LSB), and 0xF8 (XLSB)
   msb  = BMP085ReadByte(0xF6);
   lsb  = BMP085ReadByte(0xF7);
   xlsb = BMP085ReadByte(0xF8);
   p = (256*msb) + lsb + (xlsb/256);
   return(p);
}


//-----------------------------------------------
float BMP085GetTemp(float _tu)
//-----------------------------------------------
{
float alpha, T;

   alpha = _c5 * (_tu - _c6);
   T = alpha + (_mc/(alpha + _md));
   _s = T - 25;
   return(T);
}


//-----------------------------------------------
float BMP085GetPressure(float _pu)
//-----------------------------------------------
{
float x, y, z;
```

```
float P;

   x = _x2*_s*_s + _x1*_s + _x0;
   y = _y2*_s*_s + _y1*_s + _y0;
   z = ((float)_pu - x) / y;
   P = _p2*z*z + _p1*z + _p0;
   P += P_CORRECTION;
   return(P);
}


//------------------------------------------------
float BMP085Pressure(boolean getTemp)
//------------------------------------------------
{
   if (getTemp)
      _Temp = BMP085GetTemp(BMP085ReadUT());  // creates _s required
for pressure calculation
   return(BMP085GetPressure(BMP085ReadUP()));
}


//------------------------------------------------
float BMP085Temperature(void)
//------------------------------------------------
{
   _Temp = BMP085GetTemp(BMP085ReadUT());
   return(_Temp);
}
```