MINISTÉRIO DA DEFESA
EXERCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIAS E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE MESTRADO EM ENGENHARIA DE DEFESA

RODRIGO GUERATO SIQUEIRA

LOCALIZAÇÃO INERCIAL DE UM VEÍCULO QUATRO RODAS COM ESTERÇAMENTO UTILIZANDO FUSÃO SENSORIAL

Rio de Janeiro
2012

INSTITUTO MILITAR DE ENGENHARIA

RODRIGO GUERATO SIQUEIRA

LOCALIZAÇÃO INERCIAL DE UM VEÍCULO QUATRO RODAS COM ESTERÇAMENTO UTILIZANDO FUSÃO SENSORIAL

Dissertação de Mestrado apresentada ao Curso de Mestrado em Engenharia de Defesa do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Ciências em Engenharia de Defesa.

Orientador: Prof. Luciano Luporini Menegaldo, Dr.Sc.

Rio de Janeiro

2012

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 – Praia Vermelha
Rio de Janeiro - RJ CEP: 22290-270

519.2

S773L Siqueira, Rodrigo Guerato
LOCALIZAÇÃO INERCIAL DE UM VEÍCULO QUATRO RODAS COM ESTERÇAMENTO UTILIZANDO FUSÃO SENSORIAL / Rodrigo Guerato Siqueira - Rio de Janeiro: Instituto Militar de Engenharia, 2012.
116p. : il.

Dissertação: (mestrado) - Instituto Militar de Engenharia – Rio de Janeiro, 2012.

1. Engenharia de Defesa. 2. Localização Inercial. 3. Filtro de Kalman. 4. Veículos Autônomos.
I – Menegaldo, Luciano Luporini. II – Título III – Instituto Militar de Engenharia.

CDD 519.2

INSTITUTO MILITAR DE ENGENHARIA

RODRIGO GUERATO SIQUEIRA

LOCALIZAÇÃO INERCIAL DE UM VEÍCULO QUATRO RODAS COM ESTERÇAMENTO UTILIZANDO FUSÃO SENSORIAL

Dissertação de Mestrado apresentada ao Curso de Mestrado em Engenharia de Defesa do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Mestre em Ciências em Engenharia de Defesa.

Orientador: Prof. Luciano Luporini Menegaldo, D.Sc.

Aprovada em 13 de Agosto de 2012 pela seguinte Banca Examinadora:

Prof Luciano Luporini Menegaldo, D.Sc. da UFRJ – Presidente

Prof Jorge Audrin Morgado de Gois, Dr.-Ing do IME

Prof Max Suell Dutra, Dr.-Ing da UFRJ

Rio de Janeiro
2012

A todos aqueles que acreditaram em mim e me incentivaram para que eu pudesse alcançar meus objetivos. Especialmente à minha mãe Elisabete T. Guerato e a minha namorada Fernanda C. Molina.

AGRADECIMENTOS

Agradeço em primeiro lugar à minha família e amigos, especialmente à minha Mãe e a minha namorada Fernanda, que compreenderam minhas constantes ausências e falta de tempo e ainda assim sempre me incentivaram.

Ao meu orientador Prof. Dr. Luciano Luporini Menegaldo pela grande paciência, firme apoio e pela companhia nos diversos passeios de bicicleta.

Ao Prof. Dr. Jorge Audrin Morgado de Goes pelo apoio e por me honrar aceitando fazer parte de minha banca de defesa.

Ao Prof. Dr. Max Suell Dutra da COPPE por aceitar meu convite e compor a minha banca de defesa.

A CAPES que financiou a minha pesquisa e a grande maioria dos equipamentos comprados para o protótipo.

A SUBSIN na figura do Melqui Santos que me incentivou, apoiou e compreendeu as inúmeras vezes que tive que sair mais cedo, chegar mais tarde ou até mesmo não chegar para poder cumprir requisitos, entregar documentos e fazer experimentos.

Aos amigos Fernando (Donan), Alexandre (Pikachu), Leandro (Mineiro), Yuri, Roosevelt, TC Servilha de Oliveira, Maj Aquino, Ten Nina, Francisco, Tálita, Bianca, Geovanne, entre outros tantos que tive a oportunidade de conhecer e conviver dentro do IME, compartilhando muitos momentos bons e ruins.

Ao amigo Brunão que me ajudou a carregar o veículo inúmeras vezes para realizar os experimentos preliminares.

Aos amigos e colegas de trabalho Pedro Correa, Ian Esper, Mauricio Brito e Eduardo Ristow que tenho a oportunidade de conviver e de uma forma ou de outra também contribuíram para a realização deste.

Quando ouvires os aplausos do triunfo, que ressoem também aos teus ouvidos os risos que provocaste com os teus fracassos.”

S. Josemaría Escrivá

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

DARPA	–	*Defense Advanced Research Projects Agency*
EUA	–	Estados Unidos da América
CAPES	–	Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
UNICAMP	–	Universidade Estadual de Campinas
EPUSP	–	Escola Politécnica da Universidade de São Paulo
IME	–	Instituto Militar de Engenharia
VANT	–	Veículo Autônomo Não Tripulado
GPS	–	*Global Position System*
ABS	–	*Anti-lock Breaking System*
PWM	–	*Pulse Width Modulation*
MENS	–	*Micro Electro Mechanical Systems*
UMI	–	Unidade de Medidas Inerciais
NMEA	–	*National Marine Electronics Association*
UTM	–	*Universal Transverse Mercator*
UFRJ	–	Universidade Federal do Rio de Janeiro

RESUMO

O objetivo deste trabalho é o estudo de algoritmos de fusão sensorial apropriados para a combinação de sensores absolutos e relativos com características e taxas de aquisição diferentes com o fim de localizar um Veículo com a maior precisão e o menor número de interrupções possíveis.

Para gerar os dados utilizados no trabalho construiu-se uma plataforma robótica móvel em escala, onde foram instalados sensores de GPS, encoders e uma unidade de medidas inerciais (UMI). Tal plataforma dispõe de um sistema capaz de transmitir os dados dos sensores a um computador sem fios. Os dados são coletados por um programa desenvolvido em plataforma LABVIEW, capaz de coletar e armazenar os dados para posterior pós-processamento. Este pós-processamento foi realizado utilizando o *software* MATLAB e consta de algoritmos de fusão sensorial, baseadas em filtro de Kalman, que são capazes de processar os dados coletados nos experimentos, provenientes de sensores com diferentes características e diferentes taxas de amostragem, e reconstruir a trajetória realizada pelo veículo através de medidas ruidosas dos sensores.

ABSTRACT

The objective of this work is to study algorithms of sensory fusion appropriated for the combination of absolut and relatives sensors with characteristics and acquisition taxes different with the purpose of finding a vehicle with the highest accuracy and the lowest number of interruptions possible.

To generate the used data at the work a mobile robotic platform was created on scale, where GPS sensors, encoders and a inertial measurement unit were installed. Such plataform has a system capable of transmitting these data from the sensors to a wireless computer. The data are collected by a program developed in LABVIEW platform, capable of collecting and store these data for further post-processing. This post-processing was accomplished using the MATLAB software and consists of sensory fusion algorithms, based on Kalman's filter which are capable of processing the collected data at the experiments, from sensors with different characteristics and different collected samples, and reconstruct the path taken by a vehicle through noise measurements of the sensors.

1 INTRODUÇÃO

Com a crescente evolução tecnológica, as tarefas mais difíceis e perigosas tendem a ser executadas por máquinas ao invés de homens.

Em meados do século XVII, na Inglaterra, deu-se o início da hoje chamada revolução industrial, que se expandiu pelo mundo a partir do século XIX. Tal revolução produziu um profundo impacto no processo produtivo em nível econômico e social. Ao longo desse processo, o trabalho humano passou a ser substituído como nunca por máquinas que possuem um menor custo e uma maior capacidade produtiva.

Nas últimas décadas do século XX, o uso de computadores e robôs começou a tornar-se bastante expressivo. Muitas empresas e organizações passaram a despender de muitos recursos para o desenvolvimento dessas tecnologias e os robôs passaram a executar um grande número de tarefas, com menor probabilidade de erros, aumentando a produção e a qualidade da mesma (WALL, BENNETT, *et al.*, 2002).

Com a emergência dos robôs em meados dos anos 60, muitos pesquisadores já sonhavam em desenvolver veículos móveis autônomos, mas só em meados dos anos 90 é que esse assunto passou a ser estudado com maior seriedade (XIE, CHEN, *et al.*, 2007).

Nos tempos atuais os robôs substituem o homem em diversas atividades onde a vida do operário pode ser colocada em risco e, ao se tratar de guerra, não haveria de ser diferente.

Em 1958, motivada pelo lançamento do satélite *Sputnik* Soviético, foi criada nos Estados Unidos a DARPA (*Defense Advanced Research Projects Agency*). As comunidades políticas e de defesa reconheceram a necessidade de uma organização de defesa de alto nível para formular e executar projetos de pesquisa e desenvolvimento na área de defesa. A missão da DARPA é manter a superioridade tecnológica das forças militares dos EUA. Além disso, a agência visa evitar a surpresa tecnológica, patrocinando a pesquisa e desenvolvimento e fazendo a ponte entre as descobertas fundamentais e seu uso militar (DEFENSE ADVANCED RESEARCH PROJECTS AGENCY - DARPA, 2010).

Ao longo dos anos a DARPA vem trabalhando para melhorar a segurança nacional norte americana e financiando diversos projetos, como por exemplo, o *DARPA Urban Challenge*. Equipes do mundo inteiro tiveram a oportunidade de participar desse projeto. As 35 melhores equipes foram qualificadas através de trabalhos técnicos e vídeos de demonstrações e puderam participar de um evento nacional de qualificação. Essas equipes construíram veículos autônomos capazes de percorrer certa distância e transpor obstáculos definidos sem condutor ou controle remoto, navegando apenas através de sensores especiais e sistemas de posicionamento (DEFENSE ADVANCED RESEARCH PROJECTS AGENCY - DARPA, 2007).

Com a proposta de desenvolver e implantar uma rede de cooperação acadêmica brasileira na área de Defesa Nacional, possibilitando a produção de pesquisas científicas e tecnológicas e a formação de recursos humanos pós-graduados no tema, a Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES, em parceria com o Ministério da Defesa, deu inicio, no ano de 2005, ao programa Pró-Defesa. O programa dirige-se a instituições públicas e privadas brasileiras que possuam, em seus programas de pós-graduação, áreas de concentração ou linhas de pesquisa em Defesa Nacional (COORDENAÇÃO DE APERFEIÇOAMENTO DE PESSOAL DE NÍVEL SUPERIOR - CAPES, 2008).

Um dos projetos financiados pela CAPES/Pró-Defesa é o projeto intitulado: “DESENVOLVIMENTO DE UMA PLATAFORMA ROBÓTICA PARA ESTUDOS DE MOBILIDADE TERRESTRE (SEMI)-AUTÔNOMA” que é coordenado pela Universidade Estadual de Campinas (UNICAMP) e conta com a parceria da Escola Politécnica da Universidade de São Paulo (EPUSP) e o Instituto Militar de Engenharia (IME).

O projeto tem a intenção de construir um veículo autônomo semelhante aos desenvolvidos no *DARPA Urban Challenge.* Ademais, este projeto possui uma divisão definida dos trabalhos a serem realizados por cada uma das três universidades, sendo que a UNICAMP fica responsável pela navegação autônoma do veículo e integração de todos os subsistemas, a EPUSP pela estratégia de controle de trajetória baseada em análise de imagens e o IME responsável pelo posicionamento inercial.

1.1 VEÍCULOS AUTÔNOMOS MÓVEIS

Os veículos autônomos móveis são robôs móveis com a capacidade de planejar e executar suas ações a fim de realizar uma determinada tarefa sem intervenção humana, isto é, os sistemas computacionais que constituem o robô são capazes de executar as tarefas previstas usando apenas dados pré-definidos, por exemplo, mapas, e os dados adquiridos através de seus sensores, sem uso de controles remotos ou operadores. Esse veículo tem atraído a atenção de grande número de pesquisadores e diversos trabalhos podem ser encontrados falando sobre o assunto, como por exemplo (WALL, BENNETT, *et al.*, 2002) (XIE, CHEN, *et al.*, 2007) (JIN e ZHAN, 2007) (KEE, ZAIN e SALIMIN, 2008) (SARIFF e BUNIYAMIN, 2006) (SCHUBERT, MATTERN e WANIELIK, 2008). Devido ao desafio tecnológico que esse assunto impõe, são diversas as ferramentas e técnicas usadas e estudadas para modelar e documentar os sistemas e dotá-los de capacidade de interação com o ambiente.

Veículos autônomos possuem um grande número de aplicações possíveis com fins científicos, militares e comerciais (JIN e ZHAN, 2007). Eles podem atuar em diferentes ambientes. Na água, se tem como exemplo os Veículos Submarinos Autônomos – VSA adquiridos pela marinha portuguesa em 2011 e que podem ser usados no monitoramento e inspeção de objetos como *risers*, oleodutos, e instalações *off-shore*, no mapeamento de manchas de poluição ou no apoio a buscas de naufrágios entre muitos outros (REVISTA DA MARINHA, 2011). No ar existem os Veículos Aéreos Não Tripulados – VANT que podem ser utilizados para espionagem com fins militares, monitoramento de áreas de queimadas entre outras. Na terra existem os veículos de exploração, como por exemplo, os robôs *Spirit* e *Opportunity*, enviados a marte pela NASA em 2003, e até mesmo carros de passeio que já são capazes de fazer a manobra de estacionamento de forma autônoma.

Todos estes sistemas possuem em comum a capacidade de receber leituras de sensores que lhes provêm informações sobre o ambiente em que estão inseridos e de modo semi ou completamente autônomo, geram os comandos que fazem com que eles se desloquem em segurança por um ambiente.

A história dos carros autônomos teve início em 1977, no Japão, onde cientistas do *Tsukuba Mechanical Engineering Lab* construíram um carro que, seguindo

marcas brancas na pista, alcançava 30 km/h. No fim da década de 80, a *European Comission* (Comissão das Comunidades Europeias) fundou o projeto *Eureka Prometheus Project*, dedicado ao desenvolvimento de carros autônomos.

Na década de 90, diversas universidades americanas, europeias e asiáticas iniciaram pesquisas nessa área. Destacaram-se a *Carnegie Mellon University*, a *University of Michigan* e a *Stanford University.*

1.2 O PROBLEMA DE LOCALIZAÇÃO

Localizar um veículo terrestre consiste em determinar as coordenadas de sua posição e sua orientação em relação a um referencial fixo no ambiente onde ele se encontra.

Desde o inicio dos estudos em robótica, um dos problemas mais complexos e que sempre gera inúmeras discussões e trabalhos no meio acadêmico e cientifico é o problema de localização. A localização é a base para a locomoção autônoma. Se o veículo não tem informação sobre sua localização, decidir o que ele deve fazer se torna uma tarefa muito difícil (SANTOS, 2009).

Sistemas robóticos se valem de sensores dos mais diferentes tipos para obter algum conhecimento em relação a sua localização e já existem milhares de técnicas diferentes de localização utilizando sensores baseados em laser, ultrassom, GPS, plataformas Inerciais, etc.

1.3 NAVEGAÇÃO INERCIAL

Navegação inercial é definida como a determinação da posição, velocidade e orientação angular (atitude) de um veículo em relação a um sistema de coordenadas (TITTERTON e WESTON, 1997). Sistemas de Navegação Inercial são sistemas que utilizam sensores de movimentos ligados a computadores que são responsáveis por calcular continuamente a posição, a orientação e a velocidade de um veículo sem a necessidade de referências externas como, por exemplo, receptores de GPS.

Os sistemas de navegação inercial têm uma vasta gama de aplicações nos mais diferentes tipos de veículos, mas, como todo sistema de localização relativo, sofre muito com o acúmulo e a propagação de erros de medição e com os erros de integração numérica.

Estes erros podem ser minimizados com a utilização de técnicas de fusão sensorial onde se combinam a informação proveniente de sensores não inerciais, absolutos ou não, a fim de se obter uma boa estimativa das variáveis em questão.

1.4 VEÍCULOS EM ESCALA

Tradicionalmente, a indústria automobilística executa seus testes veiculares em protótipos. Esses protótipos são unidades reais do veículo a ser testado. A larga escala de fabricação garante que os elevados custos dessa metodologia sejam diluídos pelas grandes quantidades de unidades comercializadas. Recentemente, porém, aspectos ligados à segurança e à facilidade de locais apropriados para os testes, aliados ao menor custo, têm direcionado esforços no sentido do uso da teoria da semelhança para predição do comportamento dinâmico de veículos através do uso de modelos em escala. Os aspectos relacionados a locais de teste, segurança na realização dos testes e custo são mais relevantes à medida que o tamanho do veículo em questão aumenta e a escala de produção diminui (MOREIRA, 2011). Assim sendo, esses aspectos que têm uma determinada relevância nos estudos realizados por fabricantes de veículos, passam a ser muito mais relevantes ao se tratar de estudos acadêmicos onde, normalmente, os recursos são muito mais escassos.

1.5 OBJETIVOS DO TRABALHO

Veículos robóticos normalmente são operados remotamente. No entanto, esforços de pesquisa atuais buscam agregar capacidades sensoriais, de percepção – tanto do veículo em si quanto do meio onde ele evolui – e de tomada de decisão, visando o estabelecimento de estratégias de operação substancialmente autônomas

(BUENO, AZEVEDO, *et al.*, 2009). A maioria dos impeditivos relacionados a se ter um veículo robótico funcionando sem problemas advém do fato de não saber com relativa precisão a sua real localização. Diversos sistemas de localização já foram desenvolvidos baseando-se em diferentes sensores, cada um deles com suas características boas e más.

Um sistema de navegação inercial, por exemplo, baseia-se na integração no tempo das leituras de seus acelerômetros e girômetros para obter velocidade, posição e orientação. Esse sistema se mostra bastante eficiente para algumas aplicações, no entanto, o mesmo é um sistema relativo, ou seja, a posição calculada por meio desse sistema é relativa a um ponto de partida ou de referência e por isso sofre com o problema de acúmulo de erros de integração, deriva e outros tipos de ruído.

O GPS (*Global Position System*), ao contrário do sistema de navegação inercial, é um sistema de navegação absoluto, ou seja, é capaz de fornecer uma coordenada com alguma precisão do local onde ele se encontra. Esse sistema possui diversas limitações, como por exemplo, não funciona em ambientes fechados, baixa precisão, possíveis perdas de sinal dos satélites, baixa taxa de atualização em comparação com os outros sensores utilizados e a indisponibilidade de informações de orientação angular.

Para contornar os problemas de precisão do GPS existe a alternativa de usar, entre outras, a técnica conhecida como DGPS (*Differential Global Positioning System*). Utiliza-se nesse caso um sinal de correção para melhorar a estimativa de posição fornecida pelo GPS. Com essa técnica o problema de precisão é contornado, mas os problemas de perda de sinal e falta de orientação angular ainda persistem.

Sistemas de localização baseados em encoders rotativos são muito fáceis e baratos de implementar e por isso são dos mais usados em aplicações de robótica. Assim como os inerciais, são sistemas relativos e também sofrem com acúmulos de erros, além daqueles causados por deslizamentos laterais e longitudinais dos pneus.

A proposta desse trabalho é localizar um veículo autônomo com a maior precisão e o menor número de interrupções possíveis, por meio do estudo de algoritmos de fusão sensorial apropriados para a combinação de sensores absolutos e relativos com características e taxas de aquisição diferentes.

Técnicas de fusão sensorial serão utilizadas para contornar os já citados problemas de acúmulos de erros dos sensores relativos. Dentre elas, destaca-se o filtro de Kalman baseado no modelo cinemático planar. Problemas como a falta de orientação angular, baixa taxa de atualização e possíveis perdas de sinal dos sistemas de GPS são minimizados pela suplência mútua entre os sensores, proporcionando, com isso, uma medida de posição mais precisa, confiável e com maior taxa de atualização.

Para o levantamento de dados foi construída uma plataforma robótica móvel em escala. Tal plataforma é baseada em um carro de controle remoto em escala 1:8 que receberá diversas modificações necessárias para se tornar uma plataforma robótica. Além de possibilitar o levantamento de dados para o desenvolvimento do trabalho, a plataforma robótica atende ao projeto CAPES/Pró-Defesa, que visa desenvolver uma plataforma robótica móvel como citado no Capítulo 2**Erro! Fonte de referência ão encontrada.** desse trabalho. Nessa plataforma robótica, foram instalados sensores como: encoders, medidor de ângulo de esterçamento, GPS e sistema de navegação inercial. Foram desenvolvidos alguns circuitos eletrônicos que, embarcados, são capazes de ler e processar os sinais dos sensores instalados, gerenciar a plataforma robótica e fazer a interface entre os sensores e a base fixa.

A base fixa consiste em um *notebook* conectado ao veículo através de um sinal de rede sem fio e, utilizando um *software* especificamente desenvolvido para coletar os dados de todos os sensores, apresentá-los na tela e armazená-los em arquivos para futuro processamento.

1.6 ORGANIZAÇÃO DO TEXTO

O texto desse trabalho está organizado em cinco capítulos que abordam os seguintes temas: o Capítulo 2 traz uma descrição do projeto e montagem da plataforma robótica móvel, descrevendo o veículo utilizado, os sensores e suas configurações, os componentes responsáveis pela transmissão dos dados sem fio e o *software* de aquisição desenvolvido.

No Capítulo 3 foi discutido o problema de localização, bem como os tipos de localização, uma breve explicação sobre fusão sensorial e filtro de Kalman além de apresentado o modelo utilizado para o filtro.

O Capítulo 4 apresenta os resultados experimentais dividindo-os em testes preliminares realizados no laboratório, testes preliminares realizados em campo e os testes finais em campo.

No Capítulo 5 são encontradas discussões e conclusões pertinentes ao tema do trabalho.

2 PLATAFORMA ROBÓTICA MÓVEL

Nesse capítulo é apresentada a plataforma robótica móvel desenvolvida como suporte ao levantamento dos dados utilizados para o desenvolvimento desse trabalho, bem como todos os sensores e circuitos aplicados, os *softwares* desenvolvidos e a interface entre os sensores e o computador da base fixa.

Para facilitar o entendimento, o capítulo foi dividido em quatro sessões principais onde foram explorados cada um dos sensores utilizados, os circuitos utilizados para comunicação entre os sensores e o computador bem como o *software* desenvolvido para o sistema.

2.1 O VEÍCULO

A fim de atender a proposta do projeto CAPES de construir uma plataforma robótica para estudos de mobilidade terrestre (semi)-autônoma, optou-se por selecionar um veículo em escala com a máxima similaridade em relação a um veículo convencional de passeio: quatro rodas, suspenção por molas e amortecedores, tração diferencial e esterçamento nas rodas dianteiras. Considerando esses fatores, definiu-se que a melhor opção seria a da utilização de um automodelo em escala.

Existem no mercado diversos automodelos em escalas que variam entre 1:5 a 1:18 que são vendidos para o uso como brinquedo. Algumas marcas e modelos possuem suspensões e sistemas de transmissão bastante semelhantes, geometricamente, aos veículos reais. Foram pesquisadas e analisadas várias marcas e modelos de veículos em escala onde definiu-se pelo modelo Inferno GT2 fabricado pela *Kyosho*. O automodelo, que pode ser visto na FIG 2.1, é um veículo em escala 1:8 (480 mm de comprimento, 310 mm de largura e 135 mm de altura, maiores detalhes na TAB 2.1) com tração diferencial nas quatro rodas, suspenção de braços sobrepostos (conhecida como “duplo A”) que conta com molas e amortecedor, seguindo geometria semelhante à de veículos reais, motor a combustão, câmbio de duas marchas, rádio controle original de fábrica e custo

acessível ao orçamento do projeto. Entretanto, foi necessário realizar algumas alterações no automodelo, que serão descritas a seguir.

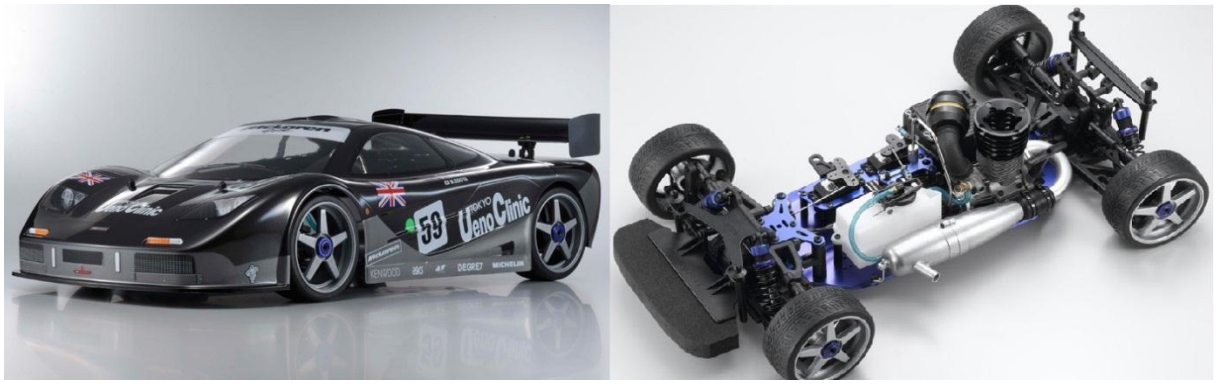

FIG 2.1: Automodelo Kyosho Inferno GT2.

TAB 2.1: Especificações do Automodelo Kyosho Inferno GT2.

Especificações	
Comprimento	480 mm
Largura	310 mm
Altura	135 mm
Distancia entre eixos	325 mm
Altura do Solo	10 mm
Distancia entre as rodas (D/T)	262 mm – 265 mm
Tamanho da Roda	91 mm x 48 mm
Relação do Câmbio	10.8:1 (1ª) / 8.37:1 (2ª)
Peso Total	3,3 Kg

2.1.1 MOTORIZAÇÃO

Originalmente o automodelo Inferno GT2 conta com um motor a combustão interna de metanol, como pode ser visto na FIG 2.2. Além de produzir elevados níveis de ruído, o motor movido a metanol produz, durante o seu funcionamento, gases nocivos à saúde, restringindo seu uso a ambientes abertos e ventilados. Além dos problemas citados, o motor de combustão interna precisa de uma constante e cara manutenção.

FIG 2.2: Detalhes do motor a combustão interna original do automodelo Inferno GT2.

A fim de facilitar os testes dentro do laboratório, antes inviáveis devido aos altos níveis de ruído e emissão de gases, visando diminuir a quantidade de manutenções e facilitar o controle através do computador, uma adaptação foi realizada onde o motor a combustão interna deu lugar a um motor elétrico sem escovas. Essa “conversão” é bastante comum nesse modelo a ponto de se encontrar kits de conversão à venda em lojas do ramo. Um kit de conversão composto por um conjunto de transmissão de apenas uma marcha, um motor elétrico sem escovas e um controlador de velocidade foi instalado no veículo por um especialista, transformando-o em um veículo movido a baterias, bem mais silencioso e livre de gases tóxicos conforme as fotos da FIG 2.3 (A) e (B).

O controlador de velocidade do motor elétrico utiliza o mesmo sinal proveniente do rádio controle que anteriormente controlava o servo-motor responsável pelo controle do carburador do motor a combustão. Isso facilita a conversão, pois basta conectar o fio sem nenhuma alteração nem configuração no sistema já existente.

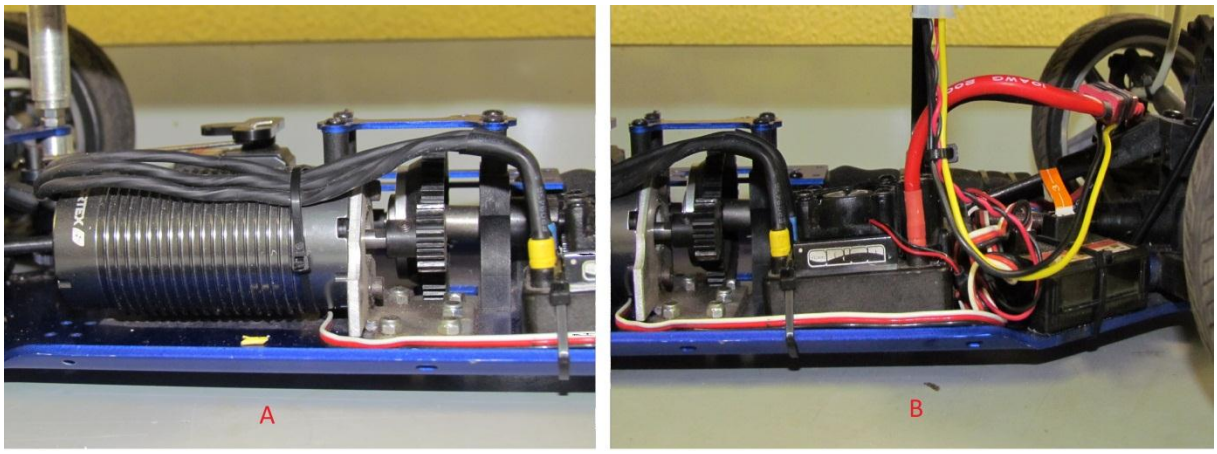

FIG 2.3: Em (A) Detalhes do motor elétrico e em (B) Detalhes do controlador de velocidade.

2.1.2 CONTROLE VIA *SOFTWARE*

Para aumentar a confiabilidade dos testes e conseguir uma maior repetitividade nas trajetórias percorridas com o veículo foi desenvolvido um sistema que possibilita o controle da aceleração, frenagem e esterçamento do veículo por um computador.

Analisando o controle remoto que compõe o kit vendido com o veículo, verificou-se que os comandos executados pelo operador do controle eram lidos através de dois potenciômetros funcionando como divisores de tensão.

A FIG 2.4 mostra um potenciômetro ligado como divisor de tensão. Nessa configuração o pino 1 é ligado à uma tensão de VCC, o pino 2 é ligado ao terra e o pino 3 apresenta uma tensão de saída proporcional à tensão de entrada e à posição do cursor do potenciômetro. Essa configuração é muito utilizada em robótica para monitorar a posição de um determinado eixo girante.

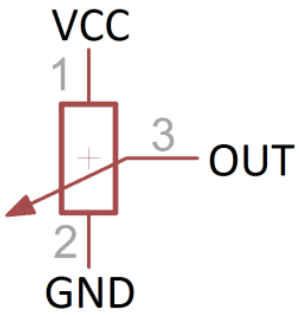

FIG 2.4: Representação esquemática de um potenciômetro.

Considerando essa a configuração dos potenciômetros empregados no circuito do controle remoto e sendo a tensão de alimentação do circuito (VCC) de 5 volts, projetou-se um circuito capaz de receber instruções de um microcomputador e gerar tensões de referência variando de 0 a 5 Volts para que fosse possível substituir os potenciômetros do controle. O circuito projetado é apresentado na FIG 2.5.

Foi utilizado um microcontrolador Microchip PIC18F252 que é o responsável pela parte lógica do circuito. Cujo *firmware* implementado está apresentado no anexo 7.2. O microcontrolador recebe instruções do computador através de uma porta serial padrão (RS-232) e gera uma tensão de referência semelhante à que seria gerada através do divisor de tensão formado pelo potenciômetro original do controle remoto. Isso permite que o computador controle, através de um *software* apropriado, as variáveis de aceleração, frenagem e esterçamento do veículo.

Além do circuito, uma placa de circuito impresso foi projeta e o desenho de fabricação dela é apresentado na FIG 2.6. Nesse desenho é possível verificar as trilhas do lado superior (em vermelho), e as trilhas do lado inferior (em azul) da placa de circuito impresso.

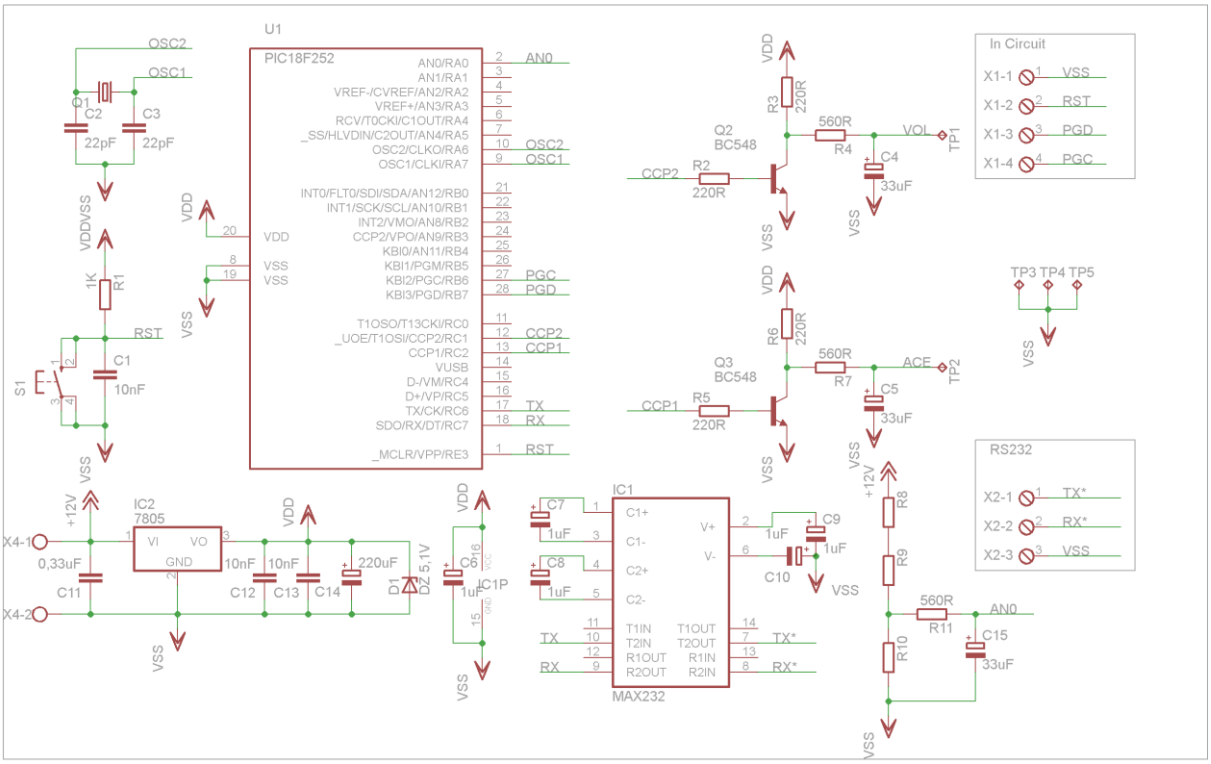

FIG 2.5: Esquema elétrico do circuito utilizado para controlar o veículo através de comandos gerados por um computador.

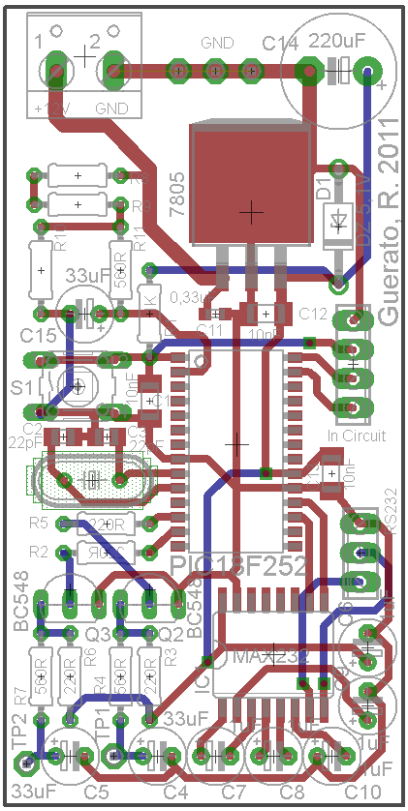

FIG 2.6: Placa de circuito impresso utilizada para controlar o veículo através de comandos gerados por um computador

Para a produção dos circuitos impressos utilizados nesse trabalho utilizou-se o seguinte processo: Os desenhos de fabricação foram encaminhados a uma empresa especializada que, confecciona as placas preparadas para a soldagem dos componentes eletrônicos. Já no laboratório do IME, os componentes foram soldados e o microcontrolador programado pelo próprio autor do trabalho. A FIG 2.7 apresenta a placa fabricada pronta para a soldagem dos componentes.

Depois de montado, o circuito foi instalado dentro da caixa do controle remoto original do veículo, na qual também foram montados: um conector DB9, para a ligação da porta serial utilizada na comunicação entre o controle e o computador, uma chave seletora que permite ao operador optar pelo uso do controle por computador ou pelo controle manual original, assim como um cabo para alimentação externa do circuito através de baterias 12 Volts.

Uma vez conectado ao computador, o controle remoto recebe os comandos de um pequeno *software* (FIG 2.8) onde são programadas sequências de comandos que possibilitam o veículo executar trajetórias pré-programadas.

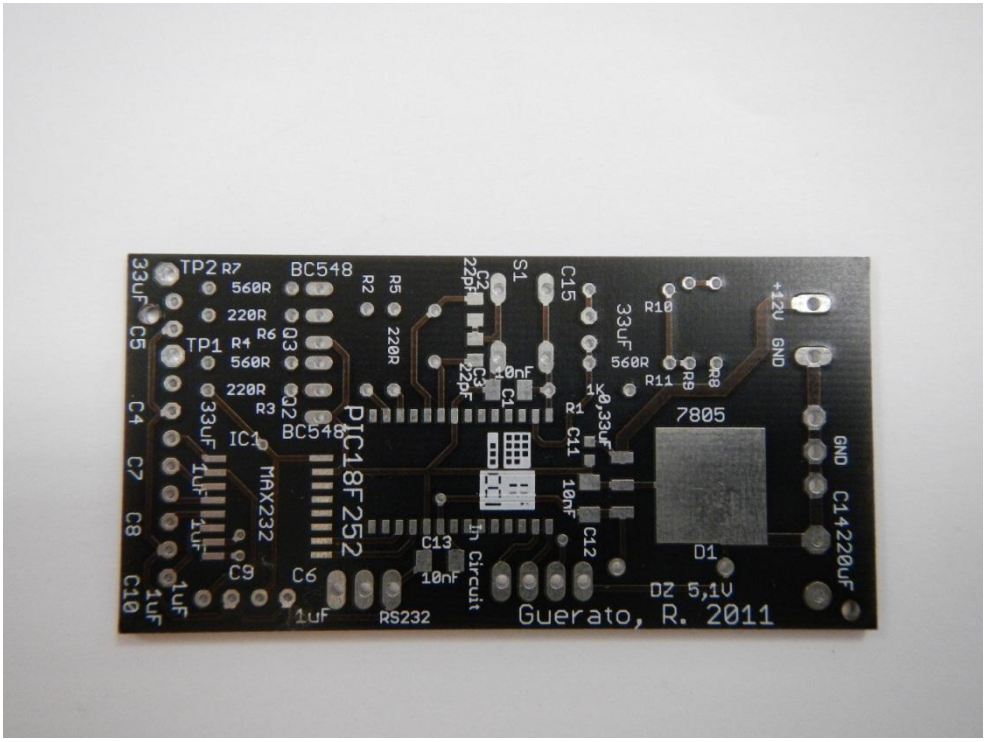

FIG 2.7: Placa de circuito Impresso pronta para a soldagem dos componentes.

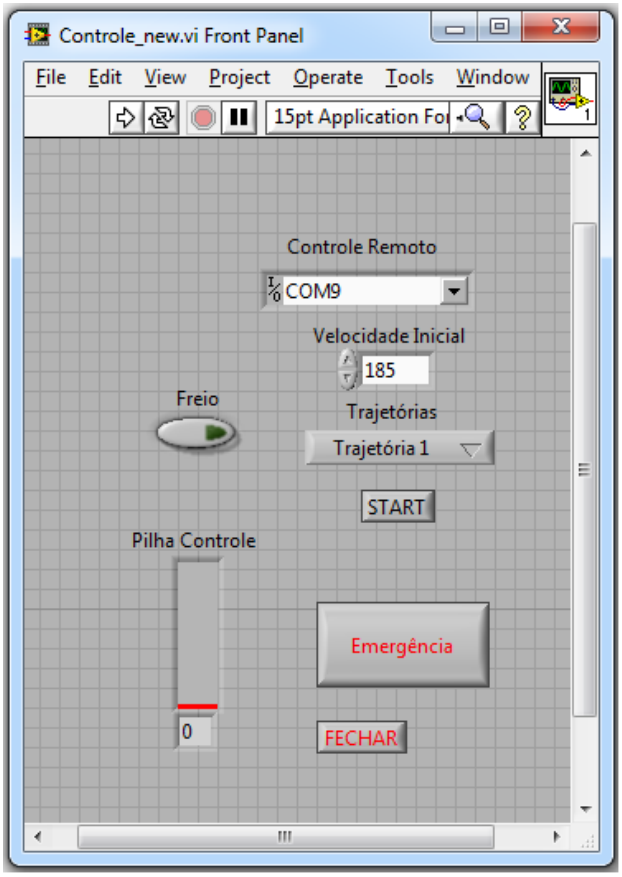

FIG 2.8: Janela do software que opera o controle remoto do veículo.

2.2 SENSORES UTILIZADOS

Com base na revisão bibliográfica apresentada anteriormente, observou-se a necessidade de se obter os seguintes dados, como um conjunto de medidas que poderiam ser integradas no algoritmo de fusão de dados: Aceleração, velocidade angular, posição angular e velocidade de cada uma das rodas, ângulo de esterçamento das rodas dianteiras e posição absoluta (adquirida através de um sistema de GPS).

Para que seja possível a obtenção de cada uma das variáveis citadas acima, foi necessário instalar uma série de sensores no veículo. Detalhes do funcionamento de cada um dos sensores utilizados foram explorados nos tópicos a seguir.

2.2.1 ODÔMETROS

A odometria é um dos métodos mais utilizados para estimar a posição de robôs móveis, devido à sua facilidade de implementação e ao seu baixo custo.

Trabalhos como (SANTOS, 2009) utilizam sensores presentes nas rodas do veículo, originalmente integrantes do sistema de freios ABS *(Anti-lock Breaking System*) para realizar medidas de odometria.

Em geral, sistemas de odometria normalmente são implementados através de encoders rotativos. O encoder gera um pulso para um determinado incremento de rotação do eixo (encoder rotativo), ou um pulso para uma determinada distância linear percorrida (encoder linear). Um exemplo de encoder rotativo pode ser visualizado na FIG 2.9. Encoders rotativos podem ser acoplados às rodas de um veiculo ou ao eixo diferencial. Monitorando-se os pulsos gerados pelo mesmo e sabendo o diâmetro da roda e resolução do encoder pode-se calcular a velocidade e a distância percorrida por cada uma das rodas.

FIG 2.9: Encoder Rotativo

O encoder ótico incremental é formado por um disco segmentado e um ou mais sensores óticos conforme ilustra a FIG 2.10. O sensor ótico é dotado de um emissor e um receptor de luz. O sensor emite um feixe de luz infravermelha que alcança o disco e pode ser refletido ou não, conforme a região do disco atingida. Quando o disco reflete o feixe de luz, o receptor é excitado provocando uma saída de nível lógico alto, enquanto que quando a luz é emitida numa região opaca, o receptor não é excitado e a saída do sensor é um nível lógico baixo (CLARK e OWINGS, 2003).

Contando-se o número de transições entre esses sinais lógicos altos e baixos, é possível calcular a posição angular relativa do encoder.

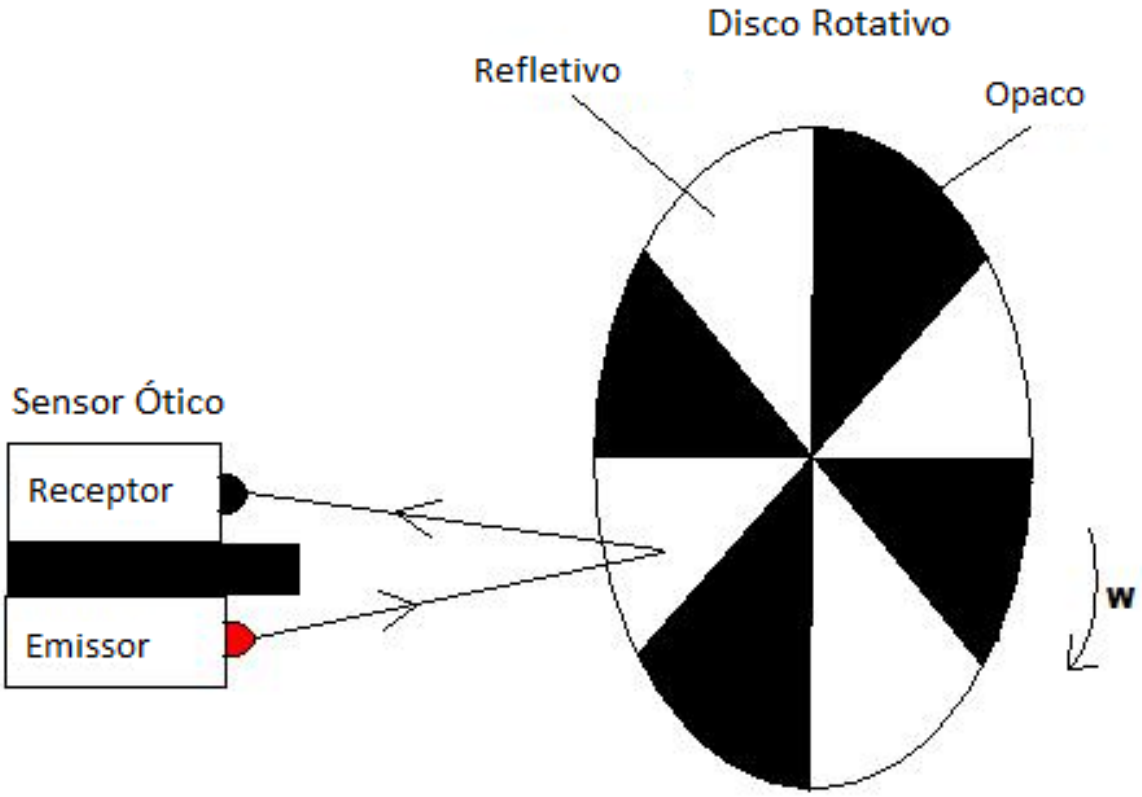

FIG 2.10: Representação didática de um encoder ótico rotativo.

No protótipo desenvolvido foram instalados quatro encoders óticos rotativos, um em cada uma das rodas do veículo. Esses encoders foram construídos artesanalmente utilizando um sensor ótico comercial modelo enc01a da Pololu, instalado no interior de cada uma das rodas, e um adesivo dotado de faixas pretas e brancas colado ao longo do perímetro interno da roda conforme se pode verificar na FIG 2.11.

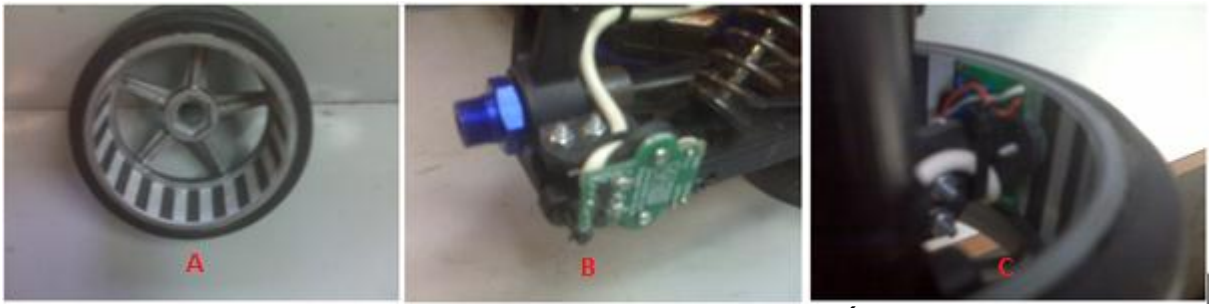

FIG 2.11: (A) Roda do veículo com adesivo colado; (B) Sensor Ótico; (C) Roda montada.

A fim de coletar os sinais gerados por cada um dos sensores óticos, foram projetados e construídos circuitos providos de um microcontrolador Microchip PIC 18F252 onde, igualmente ao circuito mostrado no item 2.1.2, é gravado um *firmware* que tem a função de contar os pulsos provenientes do encoder e calcular a posição e velocidade da roda. Esses dados de posição e velocidade calculados pelo circuito podem ser lidos por um computador através de uma porta de comunicação serial padrão RS-485. Tal padrão de comunicação serial foi adotado devido à possibilidade de criação de redes de sensores. Num mesmo canal de comunicação serial RS-485 foram ligados cinco circuitos como o apresentado na FIG 2.12. Quatro deles possuem a função descrita acima, isso é, de ler o encoder e calcular os dados de posição e velocidade de cada uma das rodas. O quinto circuito foi utilizado para a leitura do ângulo de esterçamento das rodas dianteiras, como é exposto no item 2.2.2.

A FIG 2.13 apresenta o layout da placa de leitura dos encoders no sistema de odometria. As trilhas desenhadas em vermelho se encontram na parte superior da placa enquanto as trilhas representadas em azul encontram-se na face inferior da placa. A FIG 2.14 apresenta a placa de circuito impresso pronta para a soldagem de seus componentes.

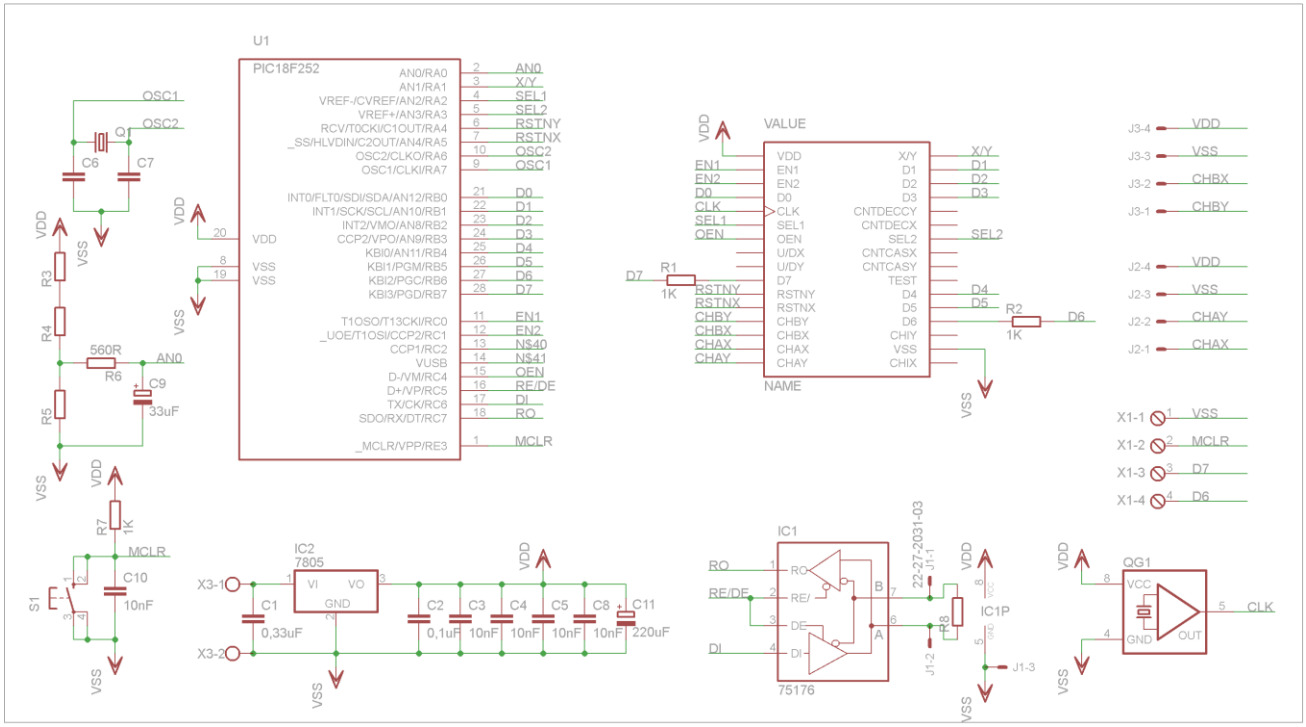

FIG 2.12: Esquema elétrico do circuito utilizado para ler os encoders.

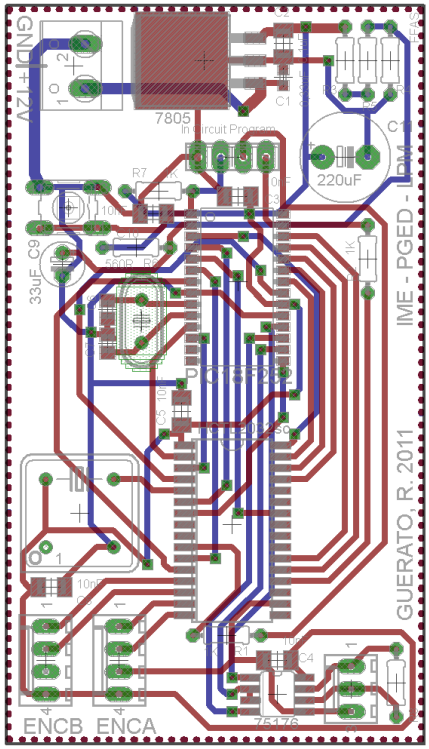

FIG 2.13: Placa de Circuito Impresso do Circuito projetado para leitura dos encoders

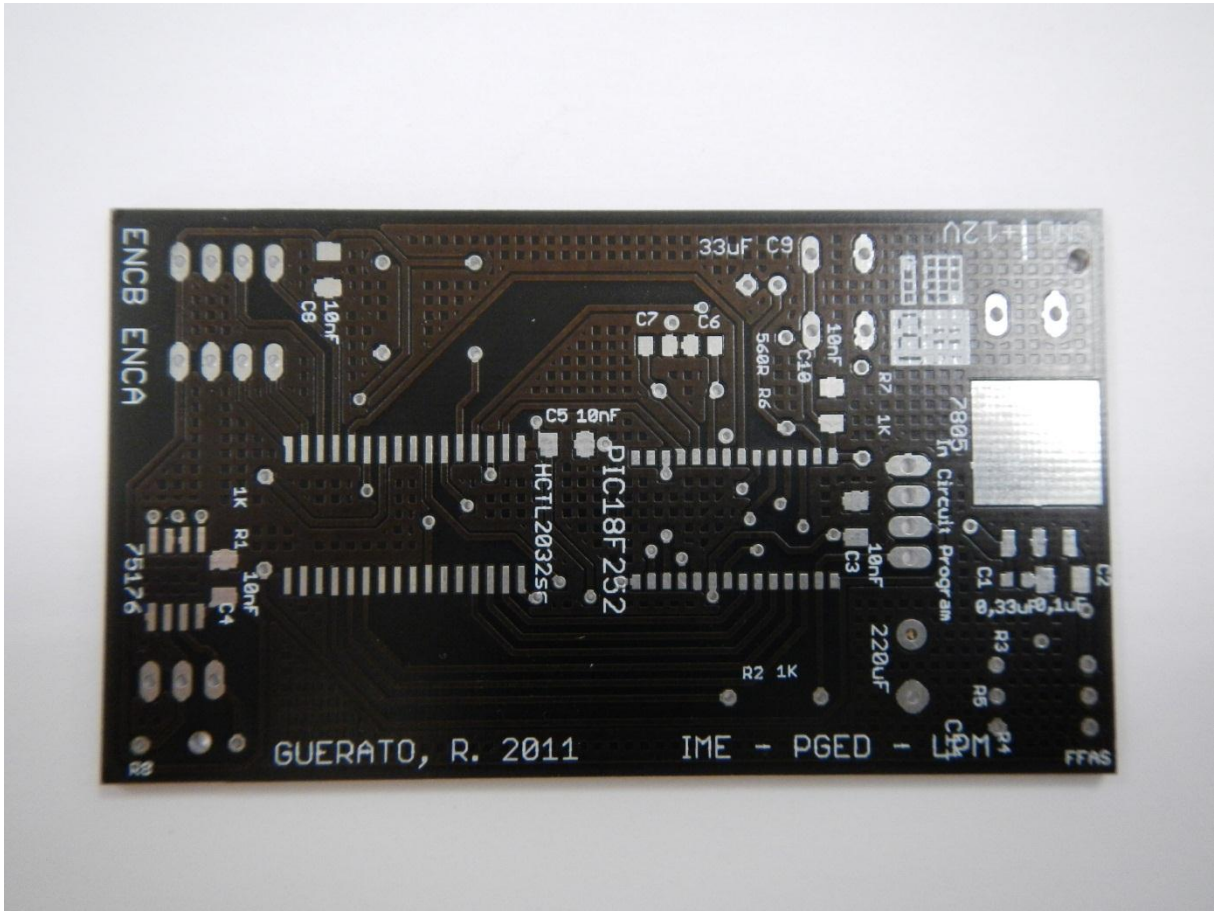

FIG 2.14: Placa de circuito Impresso pronta para a soldagem dos componentes.

2.2.2 LEITURA DO ÂNGULO DE ESTERÇAMENTO DAS RODAS

Durante o desenvolvimento do trabalho, percebeu-se a necessidade de monitorar o ângulo de esterçamento das rodas dianteiras. Um novo circuito foi necessário para atender a essa necessidade. Como circuitos semelhantes já haviam sido projetados e fabricados, optou-se por fazer uma adaptação em uma das placas de circuito impresso projetada anteriormente.

Foram feitas pequenas modificações no circuito da placa dos encoders já fabricadas e gravou-se um novo *firmware* capaz de ler e processar os sinais necessários.

No veículo, as rodas são esterçadas através da atuação de um servo-motor (FIG 2.15) que é acionado por um sinal proveniente do rádio controle.

FIG 2.15: Servo motor utilizado para o esterçamento das rodas.

Esses servo-motores de modelismo são dispositivos que possuem a capacidade de posicionamento angular dentro de certo intervalo, mantendo-se fixo na posição desejada. Eles trabalham em malha fechada, ou seja, internamente eles são capazes de determinar suas posições atuais e ao receberem um sinal de controle, comparam-no com a posição atual e atuam no sistema até atingir a posição desejada. Diferentemente de motores contínuos que giram seu eixo indefinidamente, os servo-motores possuem uma faixa de operação de poucos graus.

Como pode ser observado na foto da FIG 2.15, os servo-motores possuem três fios em seu conector: Um deles é ligado à alimentação de 5 Volts, o segundo é ligado ao terra e o terceiro recebe o sinal de controle.

Esse é modelado por PWM (*Pulse Width Modulation*), tal que o ângulo do eixo de saída do servo-motor é determinado a partir da duração da largura de pulso enviado. Esse sinal de PWM é uma onda quadrada de frequência fixa e largura de pulso variável entre 1 e 2 milissegundos, conforme ilustra a FIG 2.16. A posição do eixo do motor é sempre proporcional ao período do pulso em alto do PWM, ou seja, para um período de 1ms o servo leva seu eixo ao mínimo ângulo e para um período de 2 ms o servo leva seu eixo para a posição de máximo ângulo.

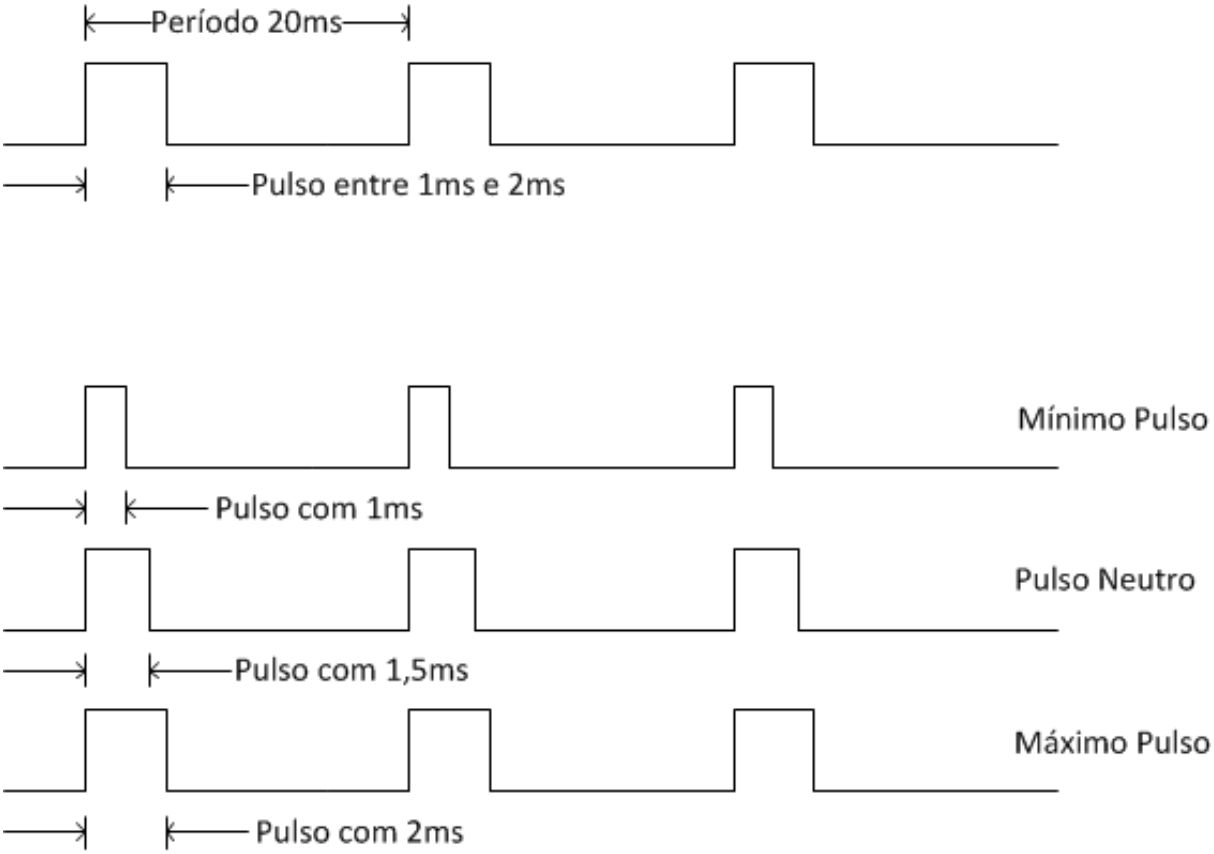

FIG 2.16: Exemplos de PWM

Conhecendo as características do servo-motor utilizado no sistema de esterçamento do veículo, foi desenvolvido um *firmware* capaz de ler a largura do pulso aplicado ao servo e entregar uma informação de contagem proporcional a esse tempo através de uma porta serial.

De posse desse sistema, um experimento simples foi realizado com o intuito de traçar uma relação entre a contagem do microcontrolador e o ângulo de esterçamento de cada uma das rodas. A roda do veículo foi posta em várias posições aleatórias anotando-se os ângulos de esterçamento, medidos através de um transferidor, e relacionando com a leitura do microcontrolador. Os dados obtidos nesse experimento foram tabulados e podem ser verificados na

TAB 2.2.

Através dos dados dessa tabela foram traçados dois gráficos de dispersão no Microsoft Excel, uma para cada roda. Em cada um dos gráficos foram traçadas linhas de tendência através de uma regressão linear, o que permitiu o cálculo de

uma equação de primeiro grau capaz de relacionar a leitura do microcontrolador com o ângulo de esterçamento de cada uma das rodas.

TAB 2.2: Leitura do Microcontrolador x ângulos de esterçamento direito e esquerdo

Leitura Microcontrolador	Roda Esquerda	Roda Direita
6285	75,0	79,0
6735	78,3	82,2
7035	80,0	84,0
7823	86,4	89,9
8386	91,0	94,0
9007	95,2	98,4
9500	98,0	101,0
9967	102,3	105,2
10205	104,0	107,0
10800	108,4	111,1

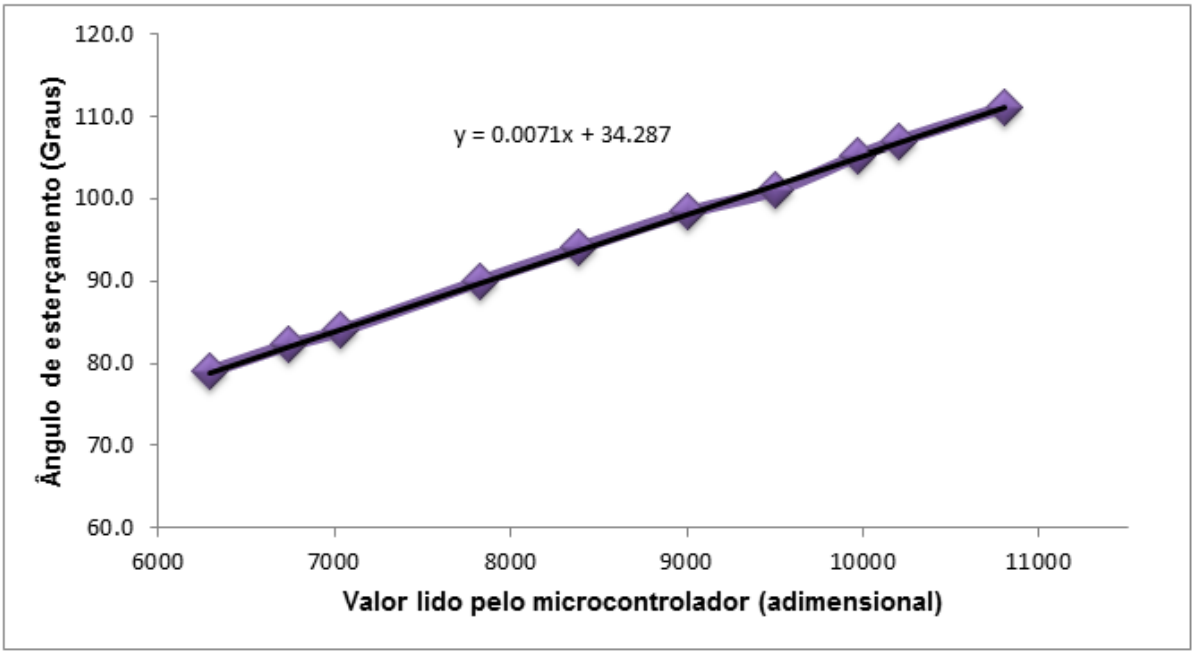

FIG 2.17: Gráfico e regressão para a roda direita.

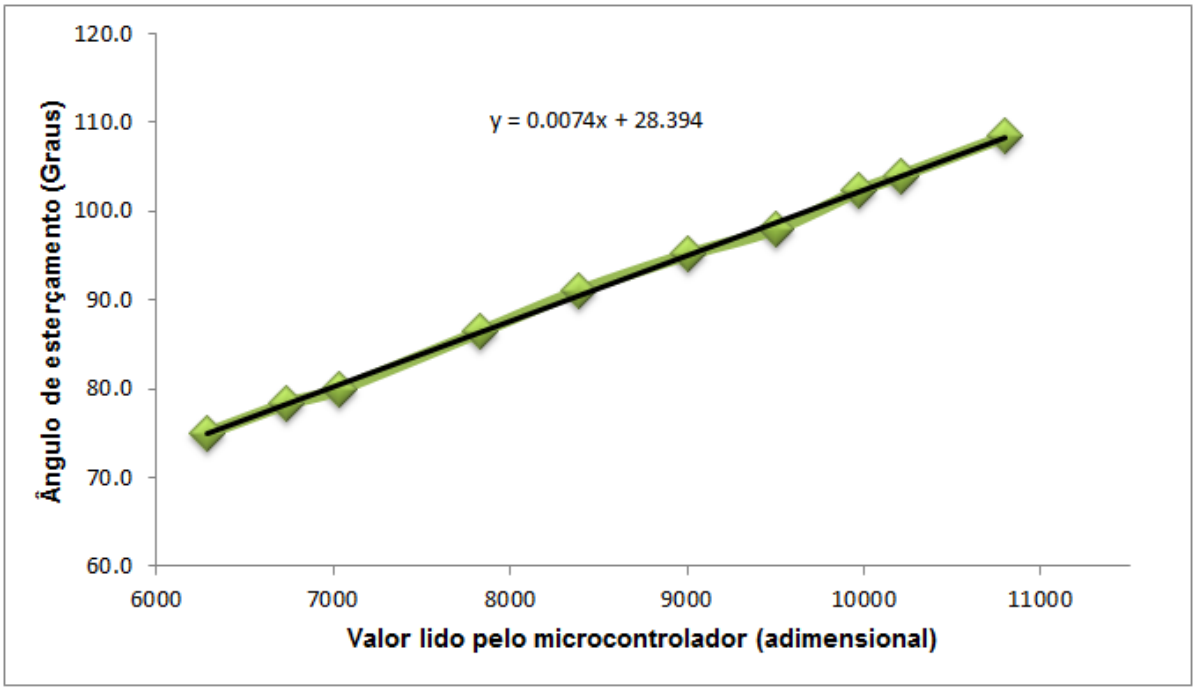

FIG 2.18: Gráfico e regressão para a roda esquerda.

As equações 2.1 e 2.2 obtidas através dos gráficos apresentados nas FIG 2.17 e FIG 2.18 foram incorporadas dentro do *firmware* do microcontrolador. Com isso, este microcontrolador tornou-se capaz de transmitir através da sua porta serial os valores dos ângulos de esterçamento de cada uma das rodas.

$$y = 0{,}0071x + 34{,}287 \quad (2.1)$$

$$y = 0074x + 28{,}394 \quad (2.2)$$

2.2.3 GPS

O GPS, (do inglês *Global Positioning System)* ou sistema de posicionamento Global, é um sistema de localização por satélites capaz de fornecer as informações necessárias a um aparelho receptor móvel para que ele mesmo determine sua posição em qualquer lugar da terra, em qualquer momento e, teoricamente, sob quaisquer condições atmosféricas.

O Sistema GPS é um sistema de propriedade do Governo dos Estados Unidos e operado através do seu Departamento de Defesa. Declarado totalmente

operacional em 1995, o sistema consiste numa “constelação” de 34 satélites que circundam o planeta Terra em trajetórias definidas a uma altitude de 20200 Km e a uma velocidade de 11265 Km/h. A combinação dessas propriedades garante que em qualquer instante de tempo, pelo menos quatro satélites estejam “visíveis” em qualquer ponto da Terra. Esses satélites possuem a bordo relógios atômicos, capazes de garantir uma informação horária com elevada precisão. Essa, por sua vez, é transmitida em conjunto com informações orbitais e de movimento. De posse das informações de quatro ou mais satélites e de informações de velocidade de propagação dos sinais, o receptor calcula a distância entre ele e cada um dos satélites “visíveis”, chamado pseudodistâncias, e utiliza-se de algoritmos matemáticos para calcular suas próprias coordenadas.

2.2.3.1 DGPS

O sistema denominado GPS foi criado para fins militares, para garantir que não fossem utilizados pelas tropas inimigas aos EUA. Até meados do ano 2000 o Departamento de Defesa Americano impunha uma chamada “disponibilidade seletiva” no sinal de GPS. Essa intervenção, que consistia em um erro pseudoaleatório introduzido ao sinal, fazia com que receptores civis operassem com erros em torno de 90 metros. Para se obter maior precisão nas aplicações civis, foi desenvolvida uma técnica denominada DGPS (*Differential Global Positioning System*), ou simplesmente GPS Diferencial. Com essa técnica os efeitos provenientes do erro introduzido no sinal são minimizados (DALBERTO, 2010). Com a desativação dessa “disponibilidade seletiva”, a técnica de DGPS continua a ser utilizada com o intuito de melhorar ainda mais a precisão das coordenadas determinadas.

O funcionamento do DGPS baseia-se na suposição de que os erros no cálculo de uma determinada coordenada são semelhantes para todos os receptores situados dentro de uma mesma área. Utilizando-se dessa suposição, são instalados receptores de GPS, chamados de receptor base, em locais georreferenciados, ou seja, lugares onde as coordenadas são muito bem conhecidas. Isso permite calcular

as diferenças entre a posição obtida através do receptor de GPS e a posição real do mesmo, estimando-se o erro do GPS. Através dessas diferenças, são gerados sinais de correção que são transmitidos ao receptor móvel que, por sua vez, os considera no cálculo das suas coordenadas.

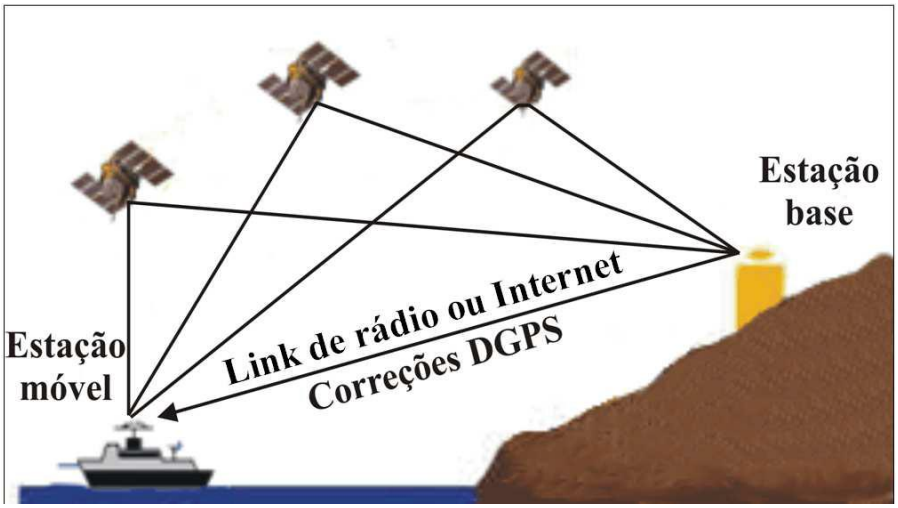

FIG 2.19: Conceito de DGPS (DALBERTO, 2010).

A transmissão dessas correções podem ser feitas por diferentes meios de comunicação como sinais de rádio, satélite e até mesmo internet. Quando a transmissão das correções é dada por meio da internet, o serviço recebe o nome de NTRIP.

2.2.3.2 NTRIP

O NTRIP do inglês (*Networked Transport of RTCM via Internet Protocol*) é um serviço para posicionamento em tempo real a partir da RBMC (Rede Brasileira de Monitoramento Contínuo), que utiliza a internet para enviar, aos receptores móveis ou estacionários, dados de correção.

A transmissão dos dados é realizada da seguinte forma: um receptor de GPS envia continuamente mensagens no padrão RTCM (*Radio Technical Commission for Maritime Services*) até um servidor “caster” localizado no IBGE (Instituto Brasileiro de Geografia e Estatística). Um usuário do serviço, dispondo-se de um computador

com um aplicativo “cliente” instalado e uma conexão com a internet, se conecta ao servidor do IBGE e escolhe a estação da RBMC cujos dados de correção diferenciais deseja receber. As correções são enviadas ao GPS do usuário através de uma porta serial padrão e desta forma obtêm-se as posições corrigidas. Atualmente o servidor do IBGE recebe dados de 27 estações localizadas nas principais capitais dos estados brasileiros (IBGE, 2011).

A RTCM é uma organização que desenvolve recomendações e critérios para a transmissão das correções das pseudodistâncias (DALBERTO, 2010). O padrão de mensagens RTCM possui diversas versões que ditam o formato em que os dados são enviados.

O padrão mais comum, com o qual a maioria dos receptores de GPS disponíveis no mercado funcionam, inclusive os de baixo custo, é a versão RTCM 2.3. No entanto, essa é uma versão mais antiga e não permite a realização de aplicações utilizando uma rede de estações de referência, como é o caso do sinal disponibilizado pelo IBGE. O padrão RTCM 3.0 é mais eficiente e suporta o transporte de um número de informações muito maior e, por isso, é o padrão adotado pelo IBGE para transmitir, de forma gratuita, as informações de correções por ele geradas.

Após uma extensiva pesquisa verificou-se que até então não existiam conversores via *software* ou *hardware*, capazes de transformar uma informação do padrão RTCM 2.3 em uma informação do padrão RTCM 3.0, o que impossibilitou o uso de DGPS através de NTRIP com um receptor de GPS de baixo custo.

FIG 2.20: GPS Novatel SMART V1.

O protótipo construído conta com um receptor de GPS (FIG 2.20) compatível com correções diferenciais RTCM nas versões 2.x e 3.0. O equipamento fabricado pela Novatel (Calgary, Canadá), modelo SMART V1, é um conjunto de Antena + receptor de GPS integrados em um mesmo encapsulamento. Ele conta com 14 canais de código L1 e rastreamento de fase e a capacidade de fornecer sinais a uma taxa de 20 Hz (NOVATEL, 2011).

2.2.4 SISTEMA DE NAVEGAÇÃO INERCIAL

Navegação inercial é o processo pelo qual se adquirem informações sobre a posição, velocidade e atitude de um veículo com relação a um dado referencial, utilizando informações fornecidas por sensores inerciais (JUNIOR, 2009).

Sistemas de navegação inercial são equipamentos compostos por girômetros, acelerômetros e bússolas magnéticas, com o propósito de fornecer dados como aceleração linear, velocidade angular, entre outros, do objeto em que ele esteja fixado. Esses sistemas são amplamente utilizados em navegação de robôs móveis (SANTOS, 2009).

O equipamento utilizado nesse trabalho é a Unidade de Medidas Inerciais (UMI) 3DM-GX2 fabricado pela Microstrain (Williston, EUA) (FIG 2.21).

FIG 2.21: Unidade de Medidas Inerciais 3DM-GX2 da Microstrain (MICROSTRAIN, 2011).

O 3DM-GX2 é uma UMI que utiliza sensores baseados na tecnologia MEMS *(Micro Electro Mechanical Systems)* que apresentam os menores custos e,

consequentemente, a pior qualidade de medidas dentre as tecnologias utilizadas para a produção de sistemas como esse. Esta unidade combina, em um pequeno encapsulamento, três acelerômetros ortogonais, três girômetros ortogonais, três magnetômetros ortogonais e sensores de temperatura. Todos esses sensores são ligados através de um conversor analógico/Digital a um microprocessador dotado de um algoritmo que permite o fornecimento de medidas em tempo real através de uma porta serial padrão RS-232 (MICROSTRAIN, 2011). Com uma taxa configurável entre 1 e 250 Hz, ele oferece diversos dados, brutos ou pré-processados.

A TAB 2.3 apresenta as especificações técnicas da unidade utilizada.

TAB 2.3: Especificações do Microstrain 3DM-GX2 (MICROSTRAIN, 2011).

Faixa de Orientação (pitch, roll, yaw)	360° em todos os eixos
Faixa de Aceleração	5 g
Estabilidade do Acelerômetro	‡0.005 g
Não linearidade do acelerômetro	0.20%
Faixa do Magnetômetro	‡1.2 Gauss
Resolução do A/D	16 bits
Acurácia da Orientação	0.5° típicos para testes estáticos; ‡2.0° típico para testes dinâmicos cíclicos e para orientações arbitrarias
Resolução da Orientação	<0.1° mínima
Repetitividade	0.20°
Modos de Saída	Aceleração e Taxa angular, Variação Angular e Variação de velocidade, Ângulos de Euler, Matriz de Rotação
Velocidade da porta Serial	115200
Alimentação	5.2 a 9 V
Corrente Consumida	90 mA
Conector	micro DB9
Temperatura de operação	-40 a 70°C
Dimensões	41 mm x 63 mm x 32 mm
Peso	39 g
Limite de impacto	1000 g desligado, 500 g ligado

2.3 INTEGRAÇÃO DOS SENSORES

Vários dos sensores utilizados neste trabalho devem ser conectados a portas seriais para que seus dados possam ser adquiridos por um computador. Como a plataforma robótica consiste em um automodelo adaptado, a instalação de um computador embarcado tornou-se desaconselhável devido ao tamanho e ao peso que isso acrescentaria ao veículo. Para contornar esse problema, foram utilizados os módulos conversores Ethernet-Serial MatchPort b/g da marca Lantronix (Irvine, EUA) (FIG 2.22). Esses módulos conversores possuem duas portas seriais configuráveis entre os padrões RS-232, RS-422 e RS-485 e podem trabalhar em redes cabeadas ou sem fio.

FIG 2.22: Conversor Ethernet Serial MachPort b/g da Lantronix.

Para facilitar a integração de tais módulos foram adquiridas duas placas de desenvolvimento do próprio fabricante. Nessas placas, os módulos instalados já possuem todos os periféricos necessários para seu funcionamento, como por exemplo, os conectores DB9 para as portas seriais, o conector RJ45 para a rede cabeada e a antena para a internet sem fio, entre outros circuitos necessários.

Utilizando esse “kit” composto pelo modulo e a placa de desenvolvimento, e um *software* especial, também fornecido pelo fabricante, as portas seriais dos módulos podem ser usadas por um computador conectado à rede de forma “transparente” ao usuário, ou seja, o usuário pode “ver” as portas seriais do módulo como se as mesmas estivessem instaladas fisicamente em seu computador. Com a utilização de dois módulos combinados somam-se 4 portas seriais disponíveis. (FIG 2.23).

Uma rede sem fio, sem a utilização de pontos de acesso - também conhecida como rede tipo *Ad-hoc* - foi criada e utilizada para conectar o computador aos dois conversores ethernet-serial.

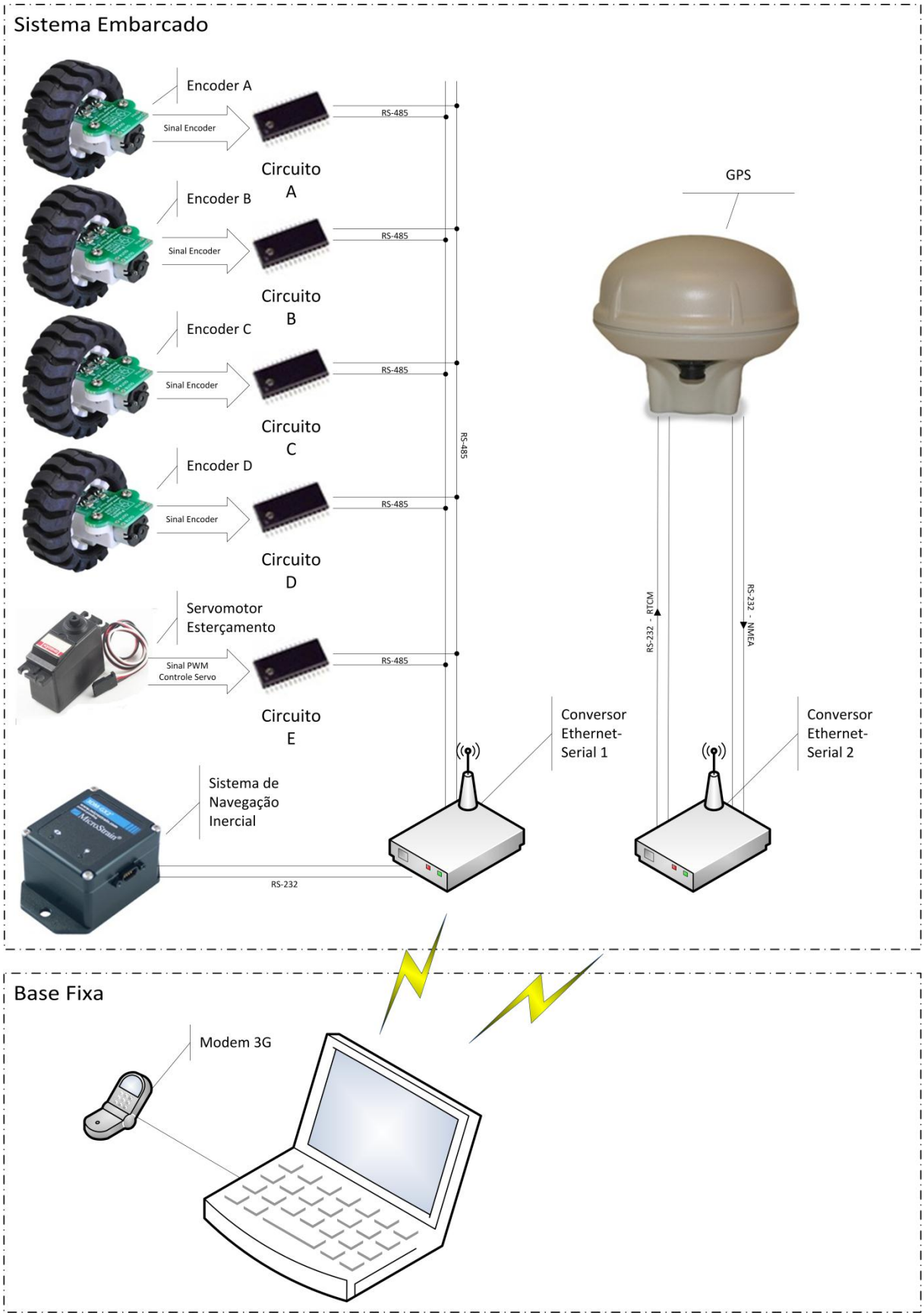

FIG 2.23: Layout da comunicação entre o computador e os sensores.

No primeiro módulo, uma das portas foi configurada para utilizar o padrão RS-485 e foi utilizada para fazer a aquisição dos dados dos circuitos contadores dos quatro encoders e do circuito responsável pela leitura do ângulo de esterçamento das rodas. Esse padrão foi escolhido para esses circuitos por permitir a formação de uma rede onde diversos circuitos podem ser conectados em paralelo. A segunda porta serial, configurada com o padrão RS-232, foi utilizada para a leitura dos sinais provenientes da plataforma inercial.

No segundo módulo, ambas as portas seriais foram configuradas com o padrão RS-232 e ligadas às duas portas seriais do GPS. A primeira porta é a responsável pela a recepção dos sinais de leitura do GPS recebidas no padrão NMEA enquanto a segunda porta é a responsável por enviar ao GPS os sinais de correção adquiridos pela internet através do modem 3G.

Todos os sensores utilizados foram acondicionados em uma caixa plástica que por sua vez foi instalada na parte superior do veículo conforme se pode ver na FIG 2.24, FIG 2.25 e FIG 2.26.

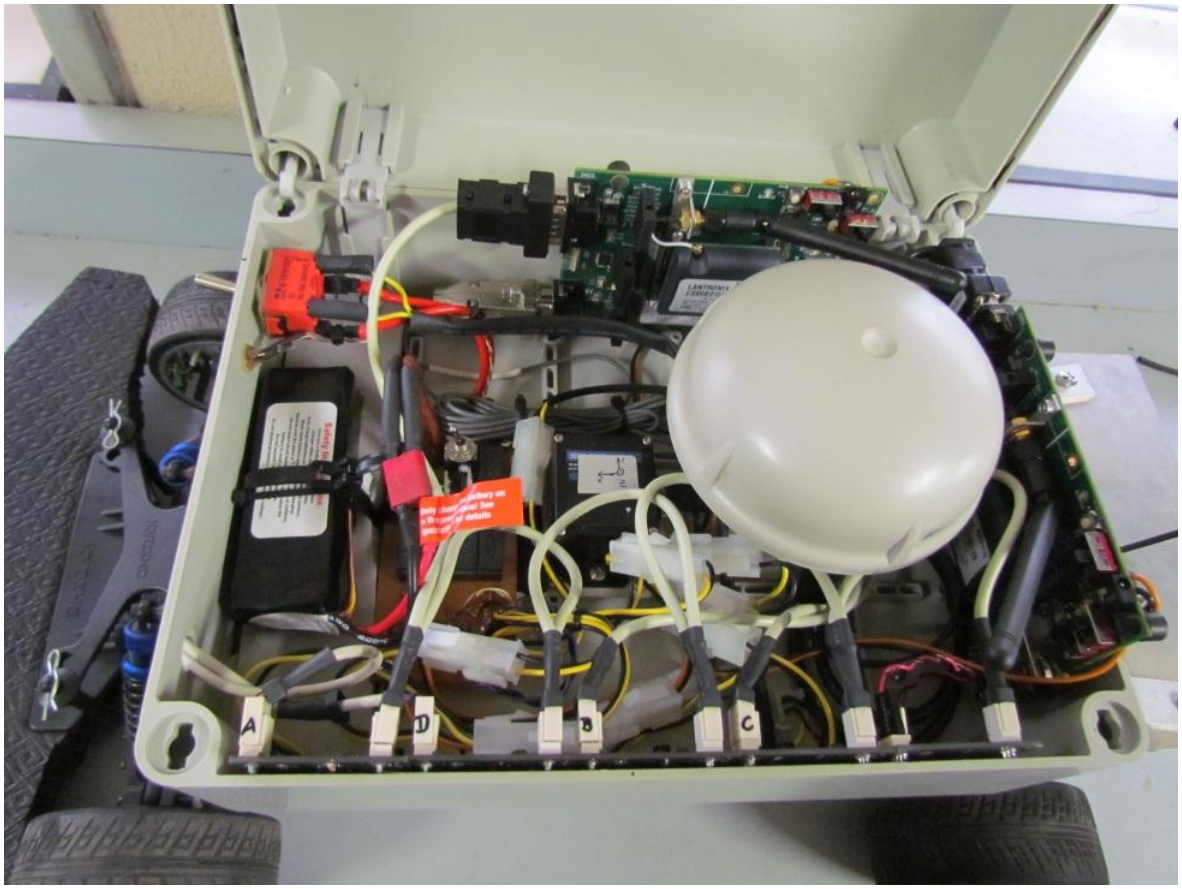

FIG 2.24: Caixa plástica contendo os sensores instalados.

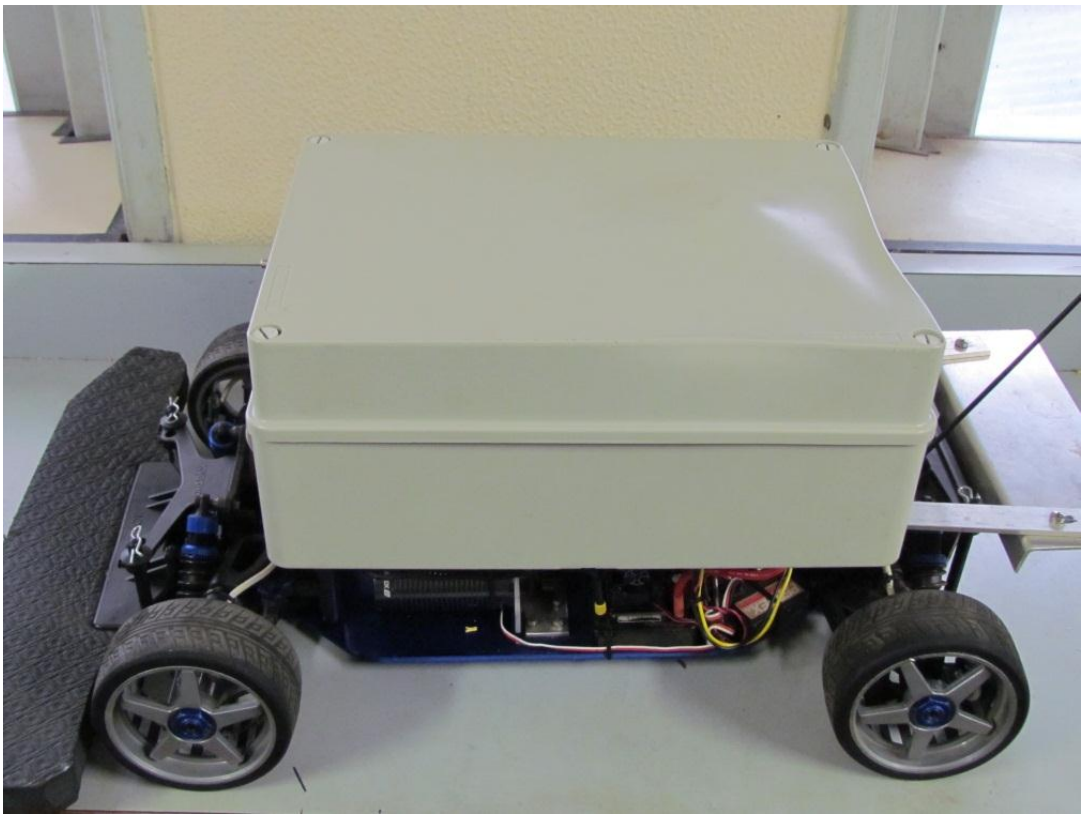

FIG 2.25: Veículo com a caixa plástica instalada.

FIG 2.26: Veículo pronto para os testes preliminares.

2.4 SOFTWARE DE AQUISIÇÃO

Desenvolvido em ambiente LABVIEW, da *National Instruments*, o *software* de aquisição de dados (FIG 2.27) tem a finalidade de gerenciar a aquisição dos dados de todos os sensores, apresentá-los na tela para a visualização e ainda gravar os dados adquiridos em um arquivo de texto (FIG 2.28), para o posterior processamento.

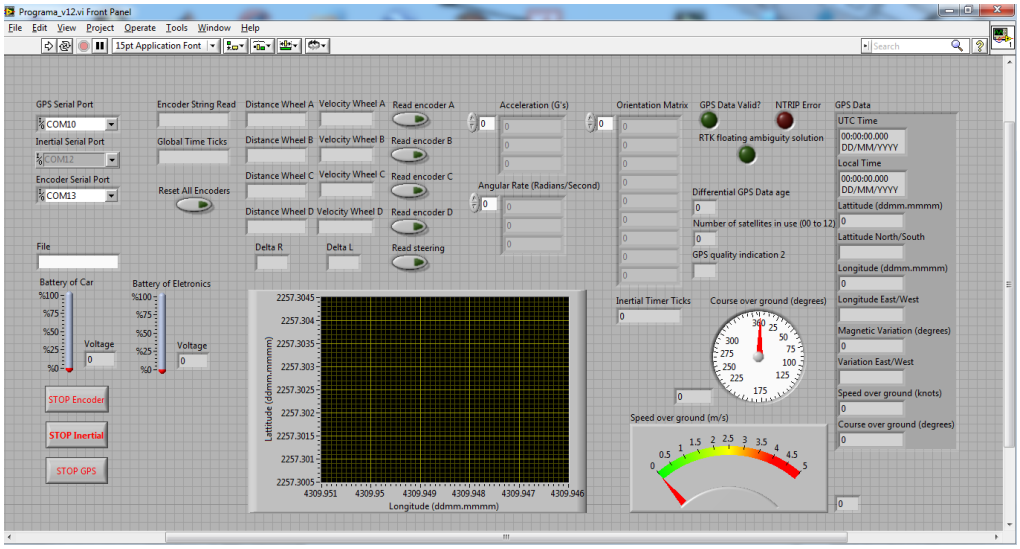

FIG 2.27: Software de aquisição de dados.

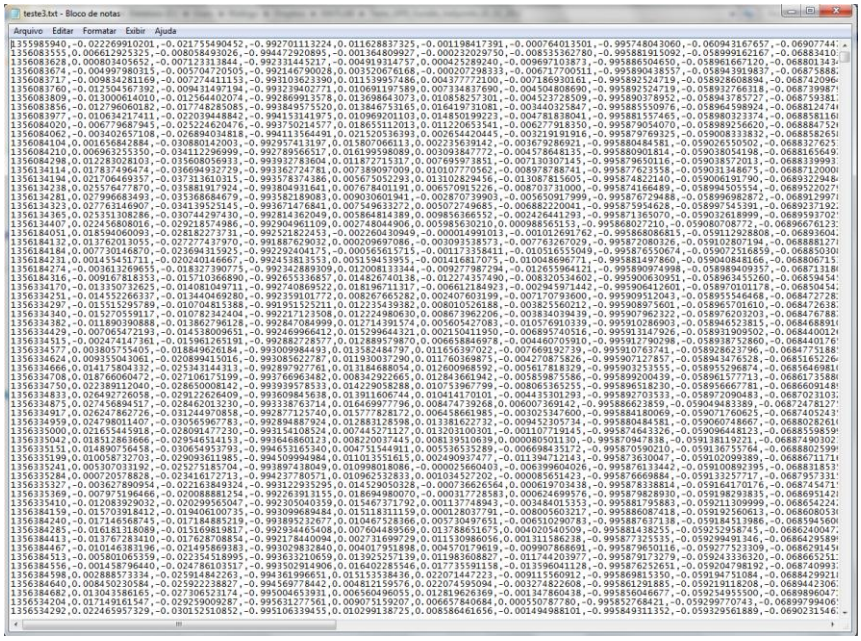

FIG 2.28: Exemplo de aquivo de texto com os dados de um teste.

3 FORMULAÇÃO DO PROBLEMA DE LOCALIZAÇÃO

O problema de localização pode ser definido como a possibilidade de determinação das coordenadas de posição e orientação angular (atitude) de um determinado corpo rígido em relação a um sistema de coordenadas. Para uma grande classe de problemas de navegação robótica, a localização é um pré-requisito.

Os sensores utilizados em um sistema de localização podem ser comparados aos sentidos nos seres vivos, ou seja, é através deles que o sistema é capaz de recolher informações do ambiente onde o mesmo está inserido e, com isso, determinar sua localização. Estes sensores podem ser divididos basicamente em dois tipos bem definidos, os sensores relativos e os sensores absolutos. Os sensores absolutos são aqueles que são capazes de fornecer informações da localização do objeto em relação ao ambiente a qualquer momento, sem qualquer relação com observações anteriores. Os sensores relativos, por outro lado, são aqueles que necessitam do conhecimento do estado anterior para gerar informações de sua posição atual. Estes fornecem valores internos ao móvel, como, por exemplo, velocidade das rodas ou ângulo de esterçamento do volante para o caso de um automóvel.

3.1 LOCALIZAÇÃO RELATIVA E LOCALIZAÇÃO ABSOLUTA

As técnicas de localização relativa são, em geral, pouco robustas, por basearem-se na medida da velocidade e direção do movimento de um determinado corpo e o tempo decorrido desde a última leitura. Nesse tipo de abordagem, acumulam-se erros de medida e erros sistemáticos a cada leitura. Como as estimativas são sempre feitas com base nas estimativas anteriores, os erros vão sendo propagados de tal forma que, em pouco tempo, as medidas estão fortemente contaminadas por erros.

As medidas de posição em um sistema de localização absoluta são independentes das informações de posição em instantes de tempo anteriores. Isto é,

para obter a posição com um sensor absoluto basta considerar a medida naquele instante. Sensores absolutos podem fornecer a localização completa de um determinado sistema ou apenas um parâmetro isolado como, por exemplo, a inclinação dada por um inclinômetro.

3.2 FUSÃO SENSORIAL

A fim de combinar as diferentes características dos sensores que existem no mercado e, com isso, conseguir uma estimativa mais confiável do que a medida de um sensor separada, foram desenvolvidas diversas técnicas de fusão sensorial. Tais técnicas têm como objetivo combinar as informações provenientes de diferentes sensores, com diferentes características e diferentes frequências de amostragem valendo-se de modelos matemáticos que determinam a evolução temporal das variáveis de estado de um determinado sistema. Seu objetivo é balancear as características de cada sensor para que a estimativa obtida seja mais confiável do que cada medição obtida isoladamente.

A fusão sensorial pode dar-se entre sensores que observam a mesma variável de um determinado processo, entre sensores complementares, que observam variáveis diferentes, ou ainda entre sensores cooperativos, ou seja, a variável observada por um dos sensores serve de base para as observações do outro sensor.

Os sensores utilizados no desenvolvimento deste trabalho fornecem todos os dados em formato digital e em diferentes frequências de amostragem. Os dados fornecidos pelos sensores são: aceleração nos três eixos, variação angular em três eixos, matriz de orientação, velocidade em cada uma das rodas, posição de cada uma das rodas, ângulo de esterçamento das rodas dianteiras, latitude, longitude, altitude, velocidade em relação ao solo e atitude.

3.3 FILTRO DE KALMAN

Criado em meados de 1960 por Rudolph E. Kalman, o filtro de Kalman foi inicialmente desenvolvido como uma solução recursiva para a filtragem linear de dados discretos. Ele consiste essencialmente num conjunto de equações matemáticas que implementam um estimador de estados preditivo, buscando corrigir interativamente a resposta de um determinado sistema através de variáveis relacionadas a ele. O filtro pode ser utilizado em processamento de imagens, processamento de sinais, supervisores de eventos discretos, sistemas de interferência, entre outros.

Este filtro foi originalmente estruturado com base num modelo de espaço de estados discreto para sistemas lineares sujeitos a ruídos nos sensores e incerteza de modelagem, ambos com características Gaussianas. Os estados são influenciados por seus próprios valores passados x_{k-1} e pelas entradas do sistema, u_k, e por sua vez influenciam as saídas do sistema z_k.

Existem duas equações que definem o modelo de espaço de estados: a equação de processo $x_k \in \Re^n$

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \tag{3.1}$$

e a equação de medição $z \in \Re^m$ que é dada por:

$$z_k = Hx_k + v_k \tag{3.2}$$

A equação de processo (3.1) é usada para estimar os estado atual x_k, através da soma de três parcelas matemáticas. A parcela inicial contém uma matriz A, relacionando o estado atual com o estado anterior (x_{k-1}) do processo. A segunda parcela contém uma Matriz B que é responsável por relacionar o estado atual com as variáveis de controle do processo, que são representados por um vetor $u_k \in \Re^l$. A terceira parcela matemática w_{k-1} representa a incerteza de modelagem do processo, considerada como um ruído branco com distribuição de probabilidade normal.

A equação de medida (3.2) é responsável por associar o estado de entrada à saída do sistema através do cálculo de um novo valor de medida z_k gerado pelo produto da matriz de correlação H com das medidas de entrada pelo estado atual. Além disso, considera-se a adição de um ruído de medida v_k, que possui qualitativamente as mesmas características do ruído de processo, mas não depende do estado do mesmo (estatisticamente não relacionados).

Integrando-se um modelo dessa natureza é possível gerar uma sequência de valores para os estados do sistema, prevendo estados futuros utilizando para isso os estados atuais.

O Filtro de Kalman estima os estados do processo em um determinado instante de tempo e então, os compara com as medições, atualizando os estados estimados. Desta forma, suas equações podem ser definidas como atualização temporal (predição) e atualização da medição (correção). As equações de predição são as responsáveis por calcular uma estimativa do estado futuro e também a covariância do erro para obter uma estimativa do erro para o instante de tempo imediatamente à frente. As equações de medição incorporam novas medições ao estado estimado *a priori* para obter uma estimativa melhorada *a posteriori*.

A variável $\hat{x}_k^-$ representa o estado *a priori* do processo em um determinado instante de tempo k. Já a variável $\hat{x}_k$ representa o estado *a posteriori*, também no tempo k, dado um valor de medida z_k.

As equações de atualização temporal para um filtro de Kalman discreto são:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{3.3}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{3.4}$$

As equações de atualização da medição para um filtro de Kalman discreto são:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{3.5}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{3.6}$$

$$P_k = (I - K_k H)P_k^- \tag{3.7}$$

O primeiro passo para executar o algoritmo de filtro de Kalman discreto é estimar um valor de medida inicial que representará o estado anterior $\hat{x}_{k-1}$ e sua estimativa

P_{k-1}. Depois de estimados estes valores, as condições para a realização da etapa de atualização de tempo já estão satisfeitas. Aplicando a equação de processo e a projeção do erro de covariância é possível calcular a projeção do estado *a priori*.

Em seguida, é executada a etapa de atualização de medida. Esta etapa é a responsável por calcular o ganho de Kalman, estimar o estado *a posteriori* e atualizar a covariância do erro. A FIG 3.1 ilustra o ciclo formado pelo algoritmo do filtro de Kalman.

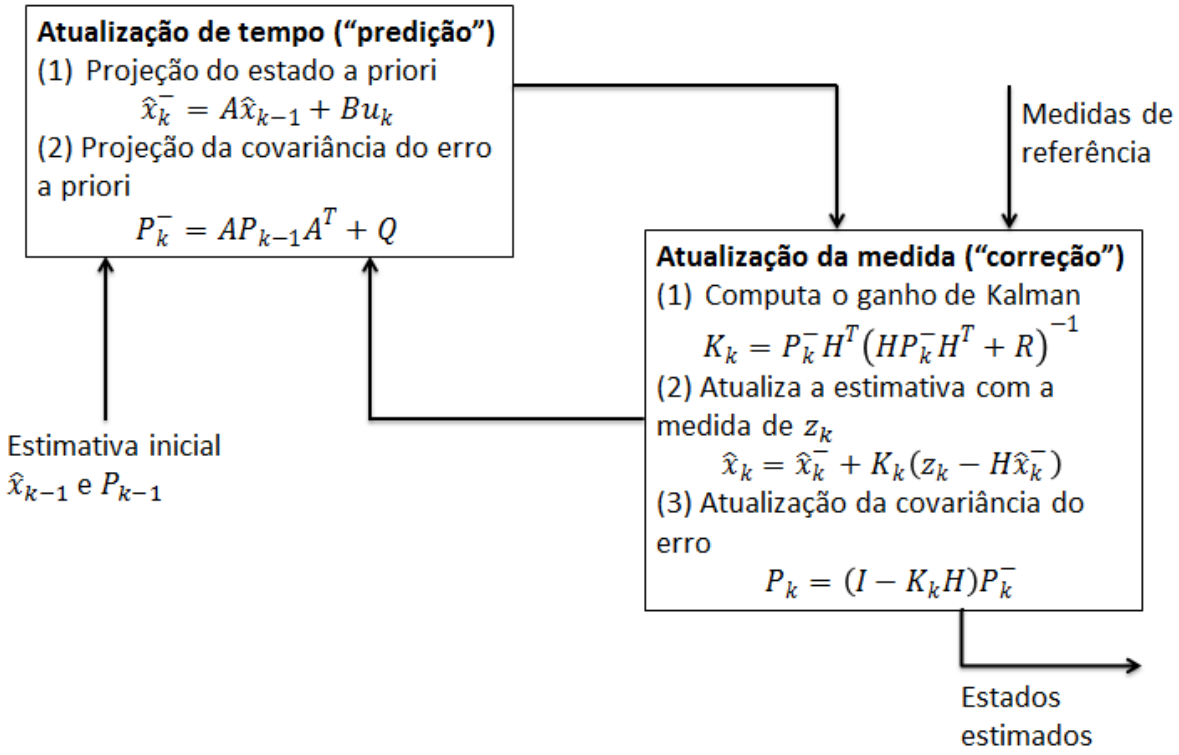

FIG 3.1: Algoritmo do filtro de Kalman

3.4 MODELO CINEMÁTICO

O modelo cinemático utilizado é baseado nas leis de movimento de Newton, onde é possível obter-se a velocidade e a posição de um corpo a partir de sua aceleração. Pelas leis de Newton, a aceleração pode ser expressa como sendo a derivada da velocidade em relação ao tempo, sendo assim:

$$\frac{dv}{dt} = a \tag{3.8}$$

Considerando que esta aceleração seja a fornecida pela UMI e que tenha erros devido à imprecisão dos acelerômetros (δa), pode-se escrever que:

$$a_{medido} = (a_{real} + \delta a) \tag{3.9}$$

Isolando-se a aceleração real, substituindo na equação (3.8) e integrando, pode-se dizer que a velocidade pode ser dada por:

$$\begin{aligned} \frac{dv}{dt} &= (a_{medida} - \delta a), \\ \int_{v_o}^{v} dv &= \int_{0}^{t} (a_{medida} - \delta a) dt \end{aligned} \tag{3.10}$$

$$v = v_0 + (a_{medida} - \delta a)t + w \tag{3.11}$$

A velocidade pode ser expressa como sendo a derivada da posição em relação ao tempo por

$$v = \frac{dp}{dt} \tag{3.12}$$

Substituindo a equação (3.11) na equação (3.12) e integrando, é obtida a posição

$$\begin{aligned} \frac{dp}{dt} &= v_0 + (a_{medida} - \delta a)t + w \\ \int_{p_0}^{p} dp &= \int_{0}^{t} v_0 dt + \int_{0}^{t} (a_{medida} - \delta a) t dt \end{aligned} \tag{3.13}$$

$$p = p_0 + v_0 t \frac{1}{2} (a_{medida} - \delta a)t^2 + w \tag{3.14}$$

Considerando que a velocidade e a aceleração sejam constantes para pequenos intervalos de tempo, as equações (3.11) e (3.14) podem ser discretizadas em

pequenos intervalos de tempo igual a T e representadas vetorialmente pelas equações (3.15) e (3.16).

$$\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + \boldsymbol{v}_k T + \frac{1}{2}\boldsymbol{a}_k T^2 - \frac{1}{2}\delta\boldsymbol{a}_k T^2 + \boldsymbol{w}_{pk} \tag{3.15}$$

$$\boldsymbol{v}_{k+1} = \boldsymbol{v}_k + \boldsymbol{a}_k T - \delta\boldsymbol{a}_k T + \boldsymbol{w}_{vk} \tag{3.16}$$

As equações (3.15) e (3.16) definem o modelo discreto cinemático do veículo e podem ser combinadas para fornecer uma representação no espaço de estados discreto como o da equação (3.17). Nelas, o vetor a_k contém as acelerações coletadas através da UMI nas direções x e y, já transportadas para o referencial da Terra. Desta forma, o vetor a_k será o vetor de entradas. O erro de aceleração δa é desconhecido, ou seja, é uma grandeza que se deseja estimar e, portanto fará parte do vetor de estados. Matricialmente tem-se:

$$x_{k+1} = Ax_k + Bu_k + Cw_k \tag{3.17}$$

Realocando os termos convenientemente, as equações (3.15) e (3.16) resultam no modelo da equação (3.18). Nesse modelo, o vetor de estados $\boldsymbol{x}$ é composto por seis elementos sendo eles: c_x e c_y as coordenadas de posição, v_x e v_y as componentes da velocidade e δa_x e δa_y as componentes do erro de aceleração. O vetor de entradas $\boldsymbol{u}$ é composto por dois elementos que são as componentes em x e em y da aceleração coletada através da UMI. O vetor $\boldsymbol{w}$ é composto por seis elementos independentes que descrevem os ruídos aleatórios do processo. A constante T é o período entre duas medições consecutivas.

$$\begin{bmatrix} c_x \\ C_y \\ v_x \\ v_y \\ \delta a_x \\ \delta a_y \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & 0 & T & 0 & \frac{T^2}{2} & 0 \\ 0 & 1 & 0 & T & 0 & \frac{T^2}{2} \\ 0 & 0 & 1 & 0 & T & 0 \\ 0 & 0 & 0 & 1 & 0 & T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} c_x \\ c_y \\ v_x \\ v_y \\ \delta a_x \\ \delta a_y \end{bmatrix}_k + \begin{bmatrix} \frac{T^2}{2} & 0 \\ 0 & \frac{T^2}{2} \\ T & 0 \\ 0 & T \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} a_x \\ a_y \end{bmatrix}$$
$$+ \begin{bmatrix} \frac{T^3}{6} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{T^3}{6} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{T^3}{6} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{T^3}{6} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{T^3}{6} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{T^3}{6} \end{bmatrix} \cdot \begin{bmatrix} w^1 \\ w^2 \\ w^3 \\ w^4 \\ w^5 \\ w^6 \end{bmatrix}_k \quad (3.18)$$

3.5 FILTRO DE KALMAN UTILIZADO PARA ESTIMAR OS ESTADOS DO SISTEMA

Após poucos segundos de simulação, elevados erros de posição são gerados, devido à presença de ruídos nas medidas fornecidas pelos acelerômetros e giroscópios da UMI. Para se estimar as variáveis com uma menor influencia dos erros pode-se adotar um filtro de Kalman, que utiliza como medidas de referência a velocidade fornecida pelos encoders instalados nas rodas do veículo, a posição fornecida por um receptor de GPS ou ambas.

Como o ruído da UMI é um ruído acumulativo e o ruído das medidas de referência de velocidade que são obtidas através dos encoders e de posição pelo receptor de GPS não são acumulativos, o filtro de Kalman pode ser usado para combinar as diferentes características dos sensores e, através deles, estimar a posição do veículo.

Os modelos das medições utilizados com o filtro de Kalman serão apresentados a seguir. Em ambos os modelos $\boldsymbol{e}$ é um vetor de erros de medidas, no primeiro modelo (3.19), $\boldsymbol{z}$ é um vetor de medidas de referencia de velocidade e no segundo modelo (3.20) $\boldsymbol{z}$ é um vetor de coordenadas de posição.

$$\begin{bmatrix} z_x \\ z_y \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} c_x \\ c_y \\ v_x \\ v_y \\ \delta a_x \\ \delta a_y \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} e^1 \\ e^2 \end{bmatrix} \tag{3.19}$$

$$\begin{bmatrix} z_x \\ z_y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} c_x \\ c_y \\ v_x \\ v_y \\ \delta a_x \\ \delta a_y \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} e^1 \\ e^2 \end{bmatrix} \tag{3.20}$$

O modelo representado por (3.19) foi utilizado para os casos onde apenas foram consideradas as velocidades provenientes dos encoders como medidas de referência. Para os casos onde o filtro de Kalman foi aplicado utilizando os sinais de velocidade (proveniente dos encoders) e de posição (proveniente do GPS) como medidas de referência, o algoritmo de correção seguiu a seguinte lógica: Em cada ciclo de correção, o algoritmo verifica se existe um novo dado proveniente do GPS. Se houver um novo dado, ele o utiliza como entrada de medida de referência, ou seja, utiliza o modelo representado por (3.20), caso contrario, a correção acontece utilizando como entrada de medida de referência o dado de velocidade e o modelo representado por (3.19).

3.6 SINTONIA DO FILTRO DE KALMAN

Para uma correta sintonia de um filtro de Kalman é necessário determinar as matrizes de covariância do ruído de processo Q e do ruído de medição R.

Em geral, o ruído de processo Q é de difícil caracterização e demanda um grande esforço experimental e de modelagem. Por esse motivo, existem várias

técnicas descritas na literatura para se chegar a uma estimativa inicial, passível ainda de ajustes por tentativa e erro (SANTANA, 2005).

Na execução da sintonia do filtro de Kalman utilizado neste trabalho, optou-se por sintonizar as matrizes Q e R por tentativa e erro. Notou-se que ao aplicar as matrizes como matrizes identidade os resultados já se mostraram satisfatórios e, portanto, os valores não foram alterados.

3.7 TRATAMETO DOS SINAIS

Como já apresentado nas sessões anteriores, os sensores utilizados geram sinais digitais que são enviados através das portas seriais ao computador que grava-os em arquivos de texto para posterior processamento dos mesmos.

Para processar os dados adquiridos nos experimentos que serão expostos no Capitulo 4, foi necessário submeter os mesmos a alguns tratamentos iniciais, os quais serão expostos a seguir.

3.7.1 GPS

Os dados provenientes do GPS são enviados ao computador em forma de mensagens do padrão NMEA *(National Marine Electronics Association).* O padrão NMEA é um protocolo de comunicação desenvolvido por uma associação de fabricantes e outros interessados nos instrumentos de navegação marítima.

Dentro do padrão existem diversas mensagens contendo diferentes tipos de informações geradas pelo equipamento receptor de GPS. Entre as diversas mensagens disponíveis no dispositivo, foram selecionadas duas para serem monitoradas pelo software de aquisição. A $GPGGA *(Global Positioning System Fix Data)* que é uma mensagem de posicionamento fixo, ou seja, ela apresenta informações sobre o tempo, latitude, longitude, altitude, número de satélites, qualidade do sinal (se o sinal é GPS ou DGPS), idade da correção (para o caso do sinal DGPS), entre outros, e a $GPRMC *(Recommended minimum specific GPS/Transit data)* que é uma mensagem contendo os mínimos dados necessários a

navegação por GPS. Assim como a mensagem $GPGGA, a $GPRMC apresenta informações de hora, latitude e longitude, bem como informações de velocidade sobre a terra (em nós) e atitude.

Em ambas as mensagens a latitude e a longitude são informadas em graus, minutos e segundos Para utilizar esses dados, é necessário convertê-los para metros. Essa conversão é feita com a utilização do método de projeção UTM (*Universal Transverse Mercator).* O sistema UTM é baseado numa projeção cilíndrica transversa perpendicular ao eixo de rotação da terra. Ele divide a terra em 60 zonas de 6 graus (FIG 3.2). Esse sistema de projeção planifica o mapa da Terra permitindo converter os dados de graus para metros mais facilmente. Um algoritmo foi implementado para a conversão em MATLAB e o código fonte dele pode ser visto na sessão 7.5.

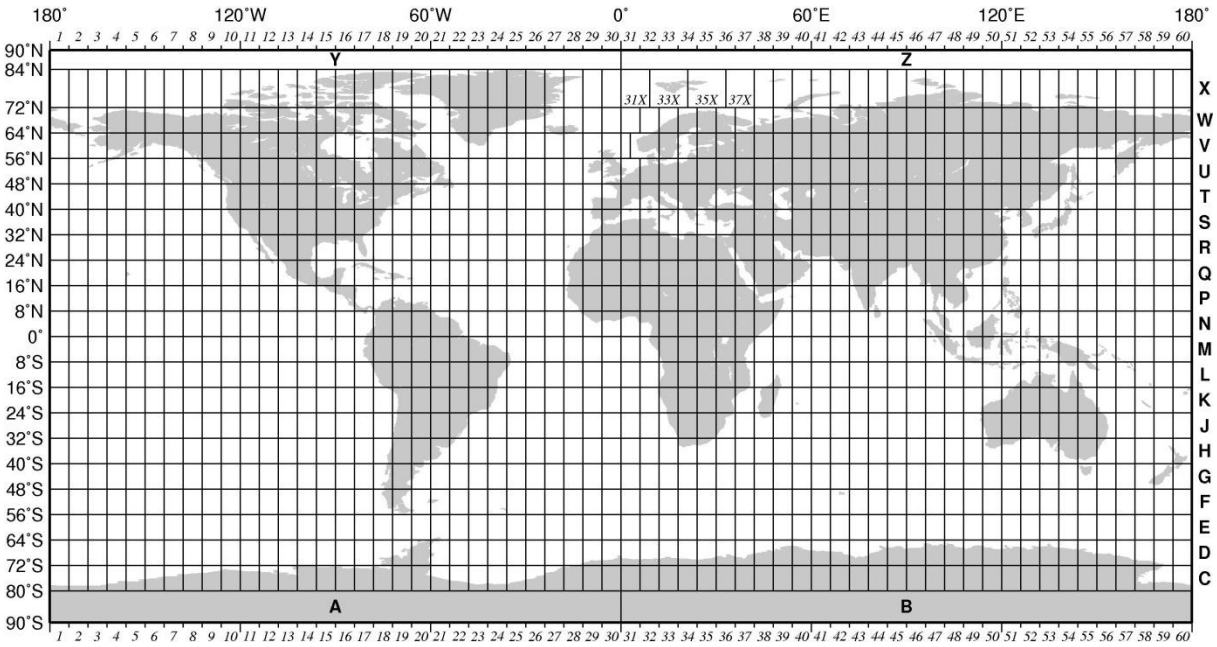

FIG 3.2: Zonas UTM [Wessel, 2010]

3.7.2 UNIDADE DE MEDIDAS INERCIAIS (UMI)

A UMI utilizada pode fornecer dados em diversos formatos diferentes. O manual do equipamento estabelece uma mensagem diferente para cada conjunto de dados que ela é capaz de fornecer e, podendo ainda trabalhar de duas maneiras

diferentes: modo discreto e modo contínuo. No primeiro, a unidade envia uma mensagem sempre que a uma requisição é enviada através de sua porta serial e a mensagem que é enviada é definida através do comando enviado. Na segunda, inicialmente são definidas a mensagem e a taxa de aquisição desejada. Em seguida, uma requisição de início do modo contínuo é enviada e a unidade passa a enviar a mensagem definida na frequência programada.

Neste trabalho a UMI foi utilizada no modo contínuo com uma frequência de 160 Hz e a mensagem utilizada foi a 0xC8. Esta é uma mensagem de 67 bytes onde são entregues a aceleração em x, y e z, a taxa angular ao redor desses mesmos eixos e a matriz de orientação, além de uma contagem utilizada como timer e um *checksum* (MICROSTRAIN, 2010).

O vetor de acelerações apresenta a direção e a magnitude das acelerações que a UMI é exposta. Os dados de aceleração que compõem esse vetor são provenientes dos acelerômetros e já se encontram processados a fim de compensar possíveis erros inerentes às variações de temperatura. Esses dados são entregues ao computador expressos em unidades de g $(1g = 9{,}80665\ m/s^2)$ e referenciado nas coordenadas locais do dispositivo. Esse valor de g utilizado pelo fabricante da UMI corresponde à gravidade ao nível do mar num local com aproximadamente 45,5 graus de latitude.

A seguinte equação foi utilizada para a transformação de unidades dos dados de aceleração:

$$g_l = 9{,}780318[1 + 0{,}0053024 \cdot \sin^2(L) - 0{,}0000058 \cdot \sin^2(2L)] - 3{,}086 \cdot 10^{-6} h \quad (3.21)$$

Onde: g_l= aceleração da gravidade local em (m/s^2)

L= latitude

h= altitude em metros em relação ao nível do mar

Para transformar os dados recebidos pela UMI do sistema de coordenadas locais do dispositivo para o sistema de coordenadas fixo na terra é necessária a utilização dos dados indicados pela matriz de transformação entregue pela própria UMI.

A matriz de transformação descreve a orientação da UMI e, com isso, ela permite transportar os dados de aceleração para um sistema de coordenadas fixo na Terra. Ela é uma matriz quadrada de nove elementos e satisfaz a seguinte equação:

$$V_{UMI} = M_{ij} \cdot V_T \tag{3.22}$$

Onde: V_{UMI} é um vetor expresso no sistema de coordenadas da UMI

V_T é o mesmo vetor expresso no sistema de coordenadas fixo na terra

Nesse trabalho, todos os dados de aceleração utilizados foram previamente processados a fim de adequar os dados às necessidades do problema de localização. Inicialmente os vetores de aceleração em cada instante de tempo foram multiplicados por suas respectivas matrizes de orientação conseguindo, com isso, a transformação dos dados para o sistema de coordenadas fixo na Terra. Em seguida, cada valor de aceleração foi multiplicado pela razão entre a gravidade local calculada e a gravidade considerada pelo fabricante para que os dados fossem apresentados em unidades de m/s^2.

3.7.3 ENCODER

Cada um dos circuitos responsáveis pelo monitoramento e contagem dos encoders das rodas fornece, através de sua porta serial, a velocidade instantânea e uma contagem de pulsos que pode ser utilizada no cálculo da distância percorrida.

Para calcular a distância percorrida através da contagem do encoder, a contagem de pulsos apresentada pelo circuito é multiplicado por uma constante que é calculada da seguinte forma:

$$K_{roda} = \frac{\pi \cdot d}{n} \tag{3.23}$$

onde: d é o diâmetro do pneu do veículo expresso em metros e n é a resolução do encoder utilizado, ou seja, o número de pulsos apresentados pelo encoder em uma revolução.

4 RESULTADOS EXPERIMENTAIS

Nesse capítulo serão apresentados os experimentos preliminares realizados com os sensores isoladamente e em seguida os testes realizados com a plataforma robótica. Serão apresentados também alguns resultados obtidos e as modificações implementadas na plataforma durante os testes.

4.1 TESTES PRELIMINARES EM LABORATÓRIO

Foram realizados testes isolados com cada um dos sensores com o intuito de familiarizar com as características dos mesmos e, com isso, definir a melhor forma de trabalhar com os dados obtidos por cada um deles.

4.1.1 GPS

O GPS apresenta em sua saída informações de latitude e longitude durante o seu funcionamento e, portanto, para avaliar o sinal coletado para um GPS, deve-se conhecer as coordenadas com a melhor precisão possível do local onde ocorre tal teste. No Brasil existem diversos pontos onde as coordenadas geográficas foram determinadas com bastante precisão e, nestes pontos, foram erguidos pilares padronizados pelo IBGE e denominados Marcos Geodésicos.

Dois dos diversos Marcos Geodésicos disponíveis na cidade do Rio de Janeiro encontram-se instalados no telhado do prédio do IME. Isso possibilitou avaliar a precisão do sinal coletado pelo GPS.

Para os testes preliminares o receptor de GPS foi instalado num dos marcos localizados no telhado do IME e ligado a um computador capaz de coletar os dados gerados e gravá-los em um arquivo para posterior processamento.

O sensor permaneceu ligado por uma hora antes dos testes para assegurar uma boa resolução e a sintonia do maior número de satélites possível. Passado esse

tempo, foram realizadas cinco coletas de dados utilizando apenas GPS e mais 5 coletas de dados utilizando a técnica de DGPS. Cada uma dessas coletas teve uma duração de aproximadamente dez minutos com o GPS trabalhando a uma frequência de 0,5Hz.

Após as coletas os dados foram processados. Foram calculadas as distâncias em linha reta entre cada ponto coletado e o ponto definido pelo IBGE e, em seguida, uma média dessa distância foi computada. Os resultados das médias das médias das distâncias calculadas podem ser verificadas na FIG 4.1.

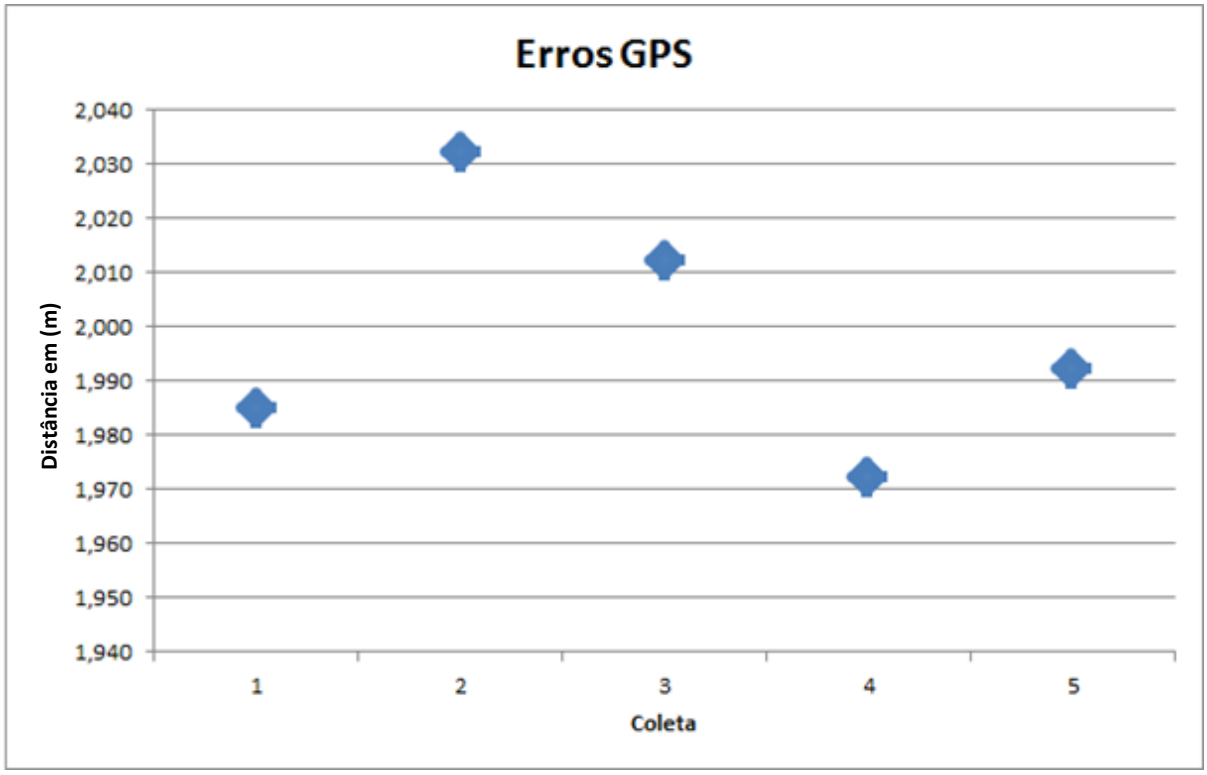

FIG 4.1: Gráfico com erro médio calculado em cada coleta GPS.

O mesmo procedimento foi aplicado aos dados coletados utilizando a técnica de DGPS. Os resultados obtidos estão apresentados na FIG 4.2.

Considerando que a distância entre o ponto determinado pelo IBGE e a leitura do receptor de GPS seja igual ao erro de leitura do mesmo, podemos concluir que o receptor utilizado, em condições ideais de trabalho, consegue fornecer dados com um erro médio em torno de 2 metros para GPS e 0,3 metros para o DGPS.

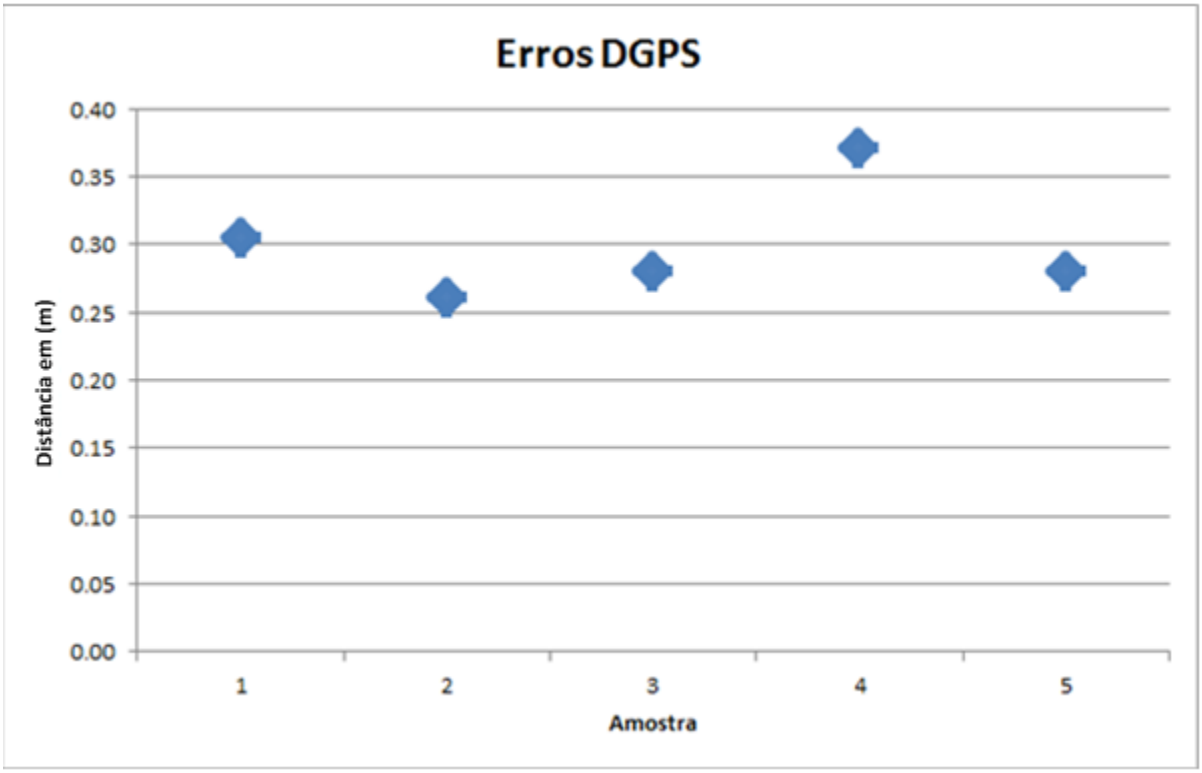

FIG 4.2: Gráfico com erro médio calculado em cada coleta DGPS.

4.1.2 UMI

A unidade de medidas inerciais foi ligada a um microcomputador com software capaz de ler e gravar os dados coletados pela unidade e gravá-los em um arquivo de texto para posterior processamento.

Inicialmente foram realizados testes onde os dados foram coletados em diferentes frequências, sempre utilizando o modo contínuo (modo onde o dispositivo realiza leituras numa frequência definida previamente pelo usuário e as envia continuamente). Foi analisado o número de dados recebidos e comparado com o previsto (calculado através do tempo decorrido) e notou-se que para frequências acima de 160 Hz, ocorrem muitas perdas de pacote. Através das observações obtidas com os testes realizados, definiu-se que o sensor seria utilizado configurado para trabalhar a uma frequência de no máximo 160Hz.

Depois de definida a frequência de amostragem, foram realizadas tomadas de dados com a UMI parada sobre o chão do laboratório a fim de se verificar a estabilidade e a qualidade dos sinais obtidos.

A FIG 4.3 apresenta as acelerações coletadas durante um teste com duração de aproximadamente oito horas. Em vermelho as acelerações do eixo x e em azul as acelerações do eixo y.

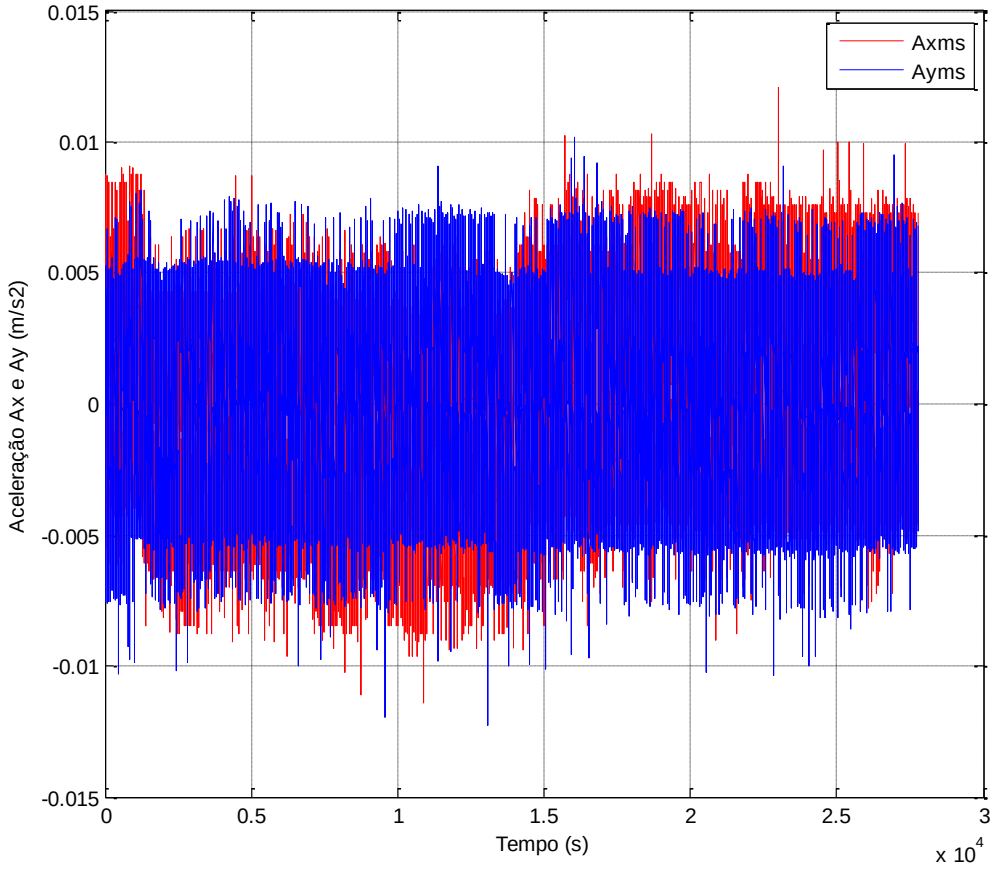

FIG 4.3: Acelerações em x e y para teste parado durante oito horas.

As acelerações apresentadas no gráfico podem ser consideradas como ruídos, pois durante o referido teste a UMI permaneceu estática. Para obter a posição da UMI através das leituras de aceleração, pôde-se utilizar a técnica conhecida como duplo integrador. A posição de um ponto no espaço é dada por:

$$p = p_0 + v_0 t + \frac{1}{2} a t^2 \tag{4.1}$$

Fazendo-se o período de amostragem igual a T e admitindo que T é muito pequeno temos:

$$p_{k+1} = p_k + v_k T + \frac{1}{2} a_k T^2 \tag{4.2}$$

As variações de velocidade e de posição, para esse conjunto de dados de aceleração integradas, estão apresentadas na FIG 4.4 e a trajetória no plano horizontal na

FIG 4.5. Evidentemente essa trajetória é o resultado da integração dos ruídos dos sensores, pois, como se sabe, o sensor não foi movimentado durante o teste.

Esse teste é interessante para mostrar que os sinais provenientes da UMI são contaminados com erros que devem ser levados em consideração ao processar esses sinais.

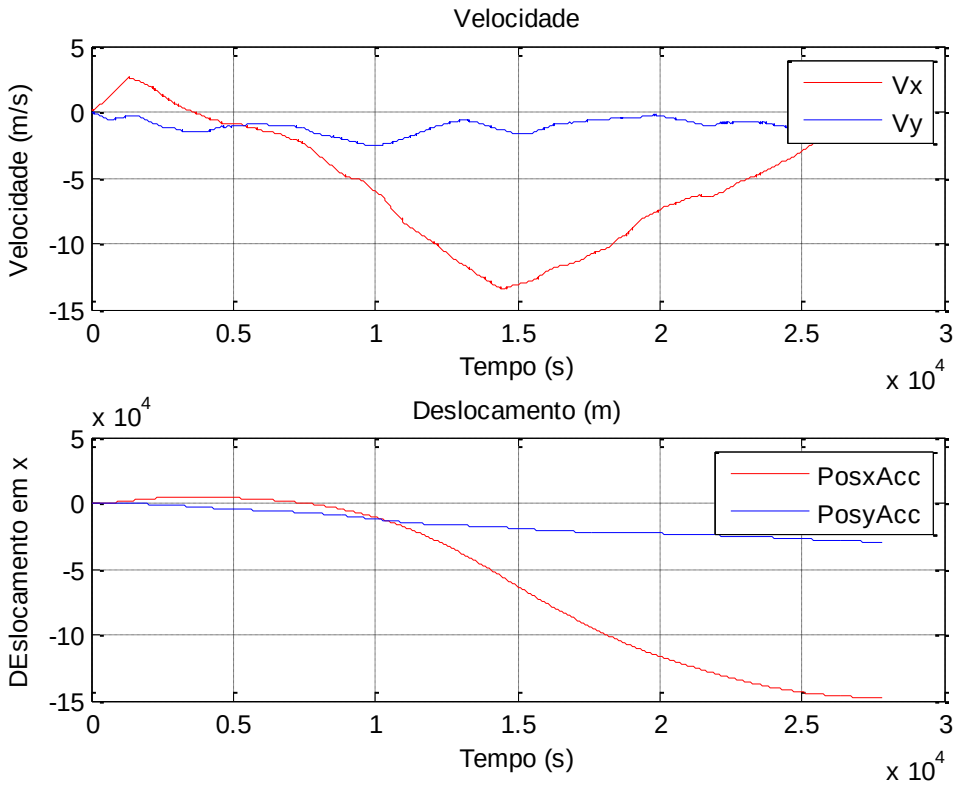

FIG 4.4: Variação da velocidade e da posição

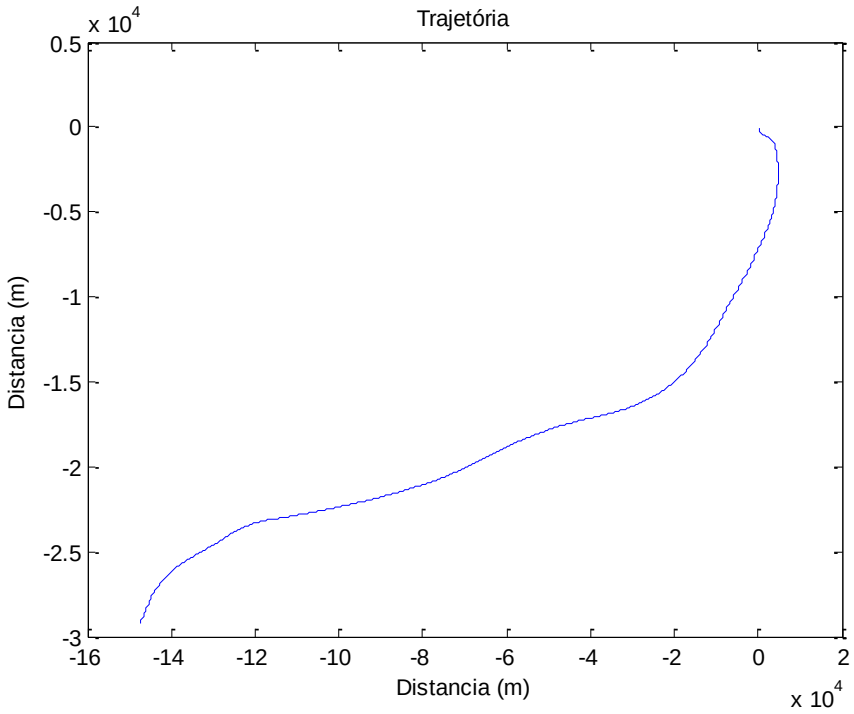

FIG 4.5: Trajetória da UMI.

Foram realizadas algumas tentativas de aplicar filtros nos sinais provenientes da UMI com o intuito de reduzir os ruídos presentes nos sinais de aceleração. Foi realizado um teste onde a UMI ficou parada durante um período de aproximadamente setenta segundos e os sinais coletados foram apresentados na FIG 4.6. Como se pode verificar no gráfico, tais acelerações possuem flutuações provenientes de ruídos dos sensores, além de componentes DC e derivas. A trajetória reconstruída através da técnica do duplo integrador pode ser vista na FIG 4.7. Novamente, a trajetória esperada deveria ser um ponto fixo.

Com o intuito de tentar minimizar esse erro na reconstrução da trajetória, o sinal foi submetido a um filtro passa-baixa do tipo *Butterworth* de sexta ordem e frequência de corte na ordem de 5Hz. Além disso foram subtraídas as componentes DC do sinal.

Analisando o sinal obtido após a etapa de filtragem, foi verificado uma tendência (deriva) no mesmo. Segundo (AGUIRRE, 2007), uma forma simples de eliminar a tendência é ajustar um polinômio (por exemplo uma reta) usando-se técnicas de regressão linear e tomar a diferença entre os dados medidos (com tendência) e o polinômio estimado. Tais diferenças passam a ser os novos dados, sem tendência.

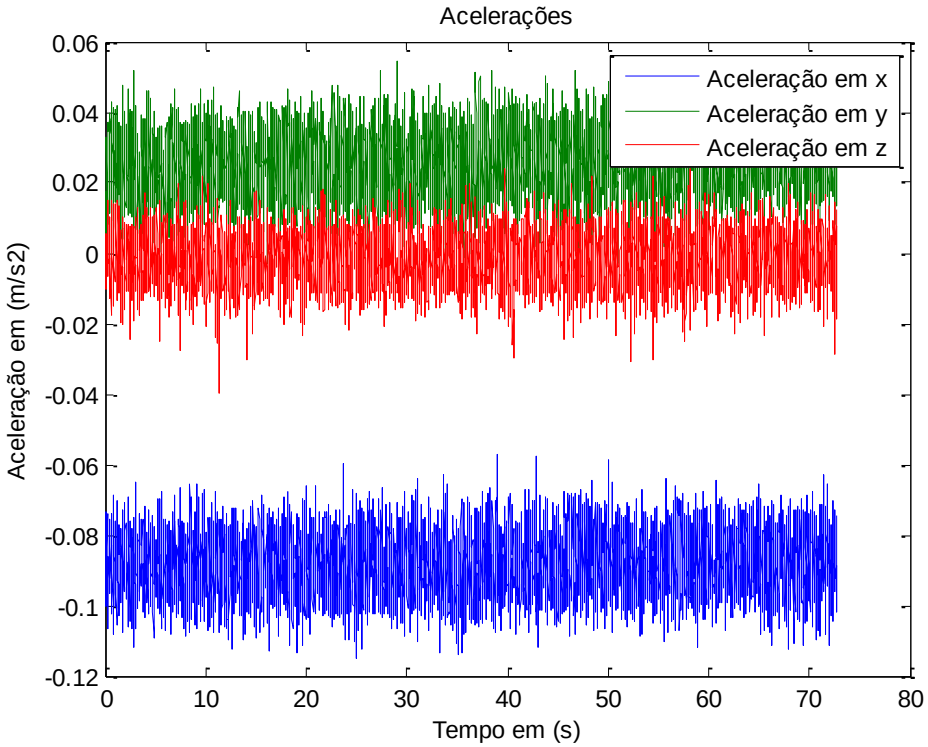

FIG 4.6: Acelerações para aproximadamente setenta segundos parados.

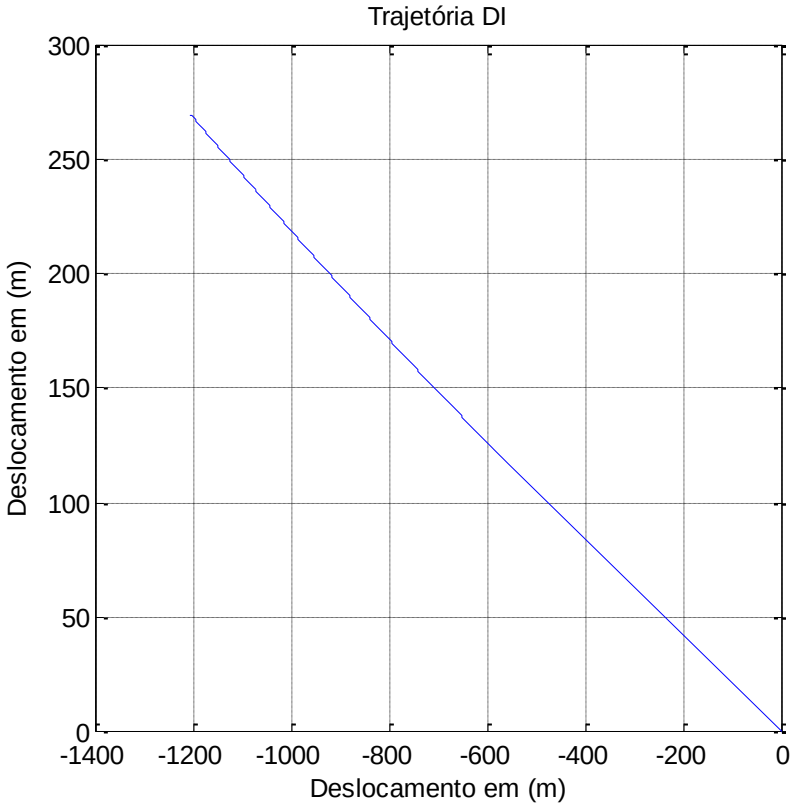

FIG 4.7: Trajetória reconstruída a partir dos dados de aceleração.

De posse desta informação, para se retirar a deriva do sinal filtrado, foram ajustados polinômios de primeiro grau através de cada um dos vetores de aceleração e tomadas às diferenças entre os dados medidos e o polinômio, conforme a equação (4.3). Esses polinômios foram ajustados utilizando a função POLYFIT do MATLAB.

Depois de calculados os coeficientes, eles foram utilizados para determinar os pontos das retas que, por sua vez, foram subtraídos dos vetores de aceleração, conforme a equação (4.3). O sinal resultante (FIG 4.8) foi submetido à técnica do duplo integrador e a trajetória obtida pode ser observada na FIG 4.9.

$$A_k = B_k - R_k \tag{4.3}$$

Onde: A_k é a aceleração sem a deriva no instante k.
B_k é a aceleração com a deriva no instante k.
R_k é o ponto da reta ajustada no instante k.

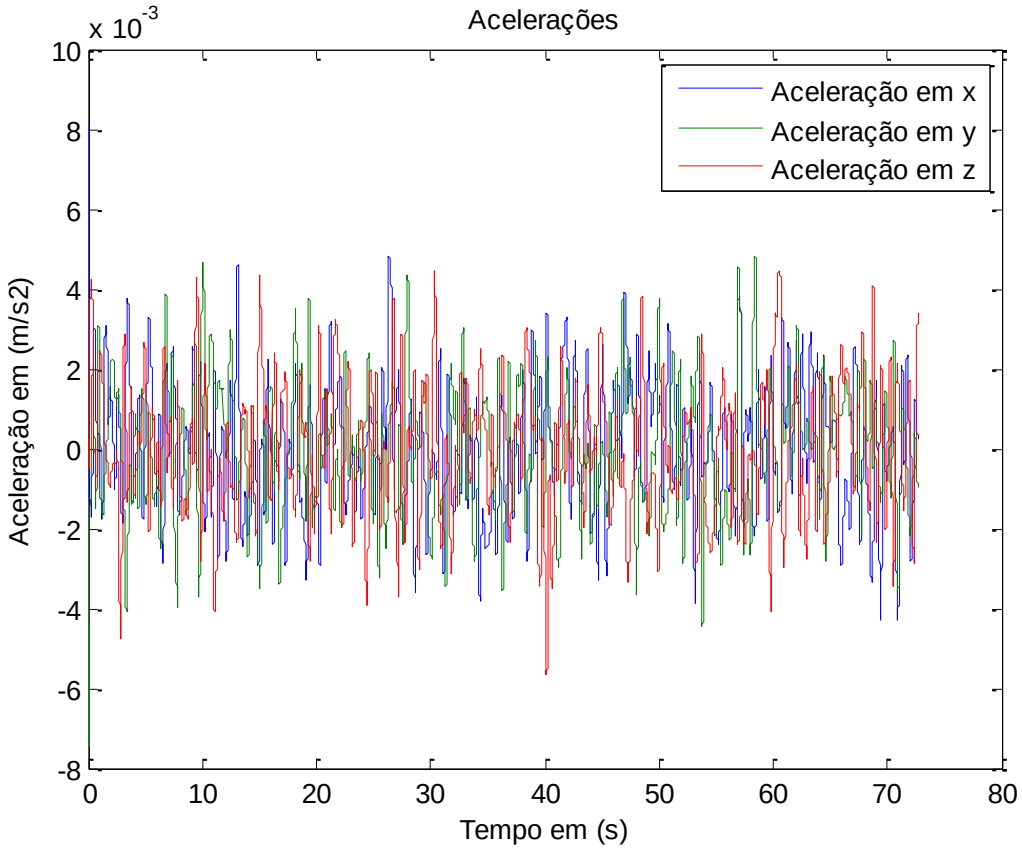

FIG 4.8: Sinal filtrado.

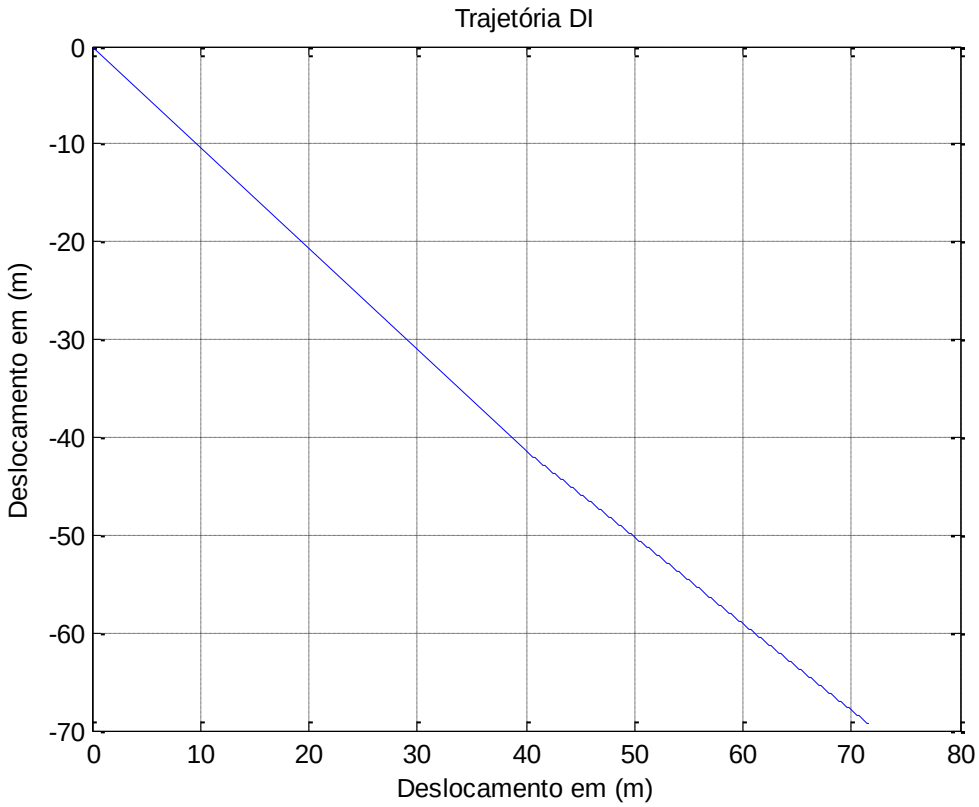

FIG 4.9: Trajetória reconstruída a partir dos dados de aceleração filtrados.

Comparando os gráficos apresentados nas FIG 4.7 e FIG 4.9, pode-se notar que a reconstrução da trajetória com o sinal filtrado apresenta um erro bastante inferior quando comparada com a reconstrução utilizando o sinal original. Mesmo assim, ainda é uma reconstrução bastante ruim, pois a trajetória apresenta um deslocamento de vários metros enquanto a UMI ficou parada.

4.2 TESTES DE LOCOMOÇÃO

O sistema descrito no capitulo 2.1.2 responsável pelo controle de trajetória através do computador é um sistema de malha aberta, ou seja, o sistema atua no veículo sem que haja algum sinal de retorno para verificar se o mesmo executou de maneira correta a tarefa. Nos testes preliminares realizados, notou-se que o mesmo poderia ser influenciado por um grande número de variáveis não controladas, que fazia com que as trajetórias não fossem executadas da maneira correta. Um

exemplo disso é a rotação do motor que é proporcional ao nível de carga da bateria. Conforme o veículo se locomove pela pista e a carga da sua bateria é consumida, a rotação do motor, para um mesmo valor de referência, diminui fazendo com que o mesmo não consiga repetir a trajetória satisfatoriamente. Outros fatores como possíveis irregularidades no solo e pequenas derrapagens dos pneus também influenciam na repetibilidade dos testes.

Tais fatores tornaram pouco prático o controle de uma trajetória em malha aberta. Embora o sistema para controlar o veículo através do microcomputador tenha sido implementado, optou-se pela execução das trajetórias de forma manual, ou seja, o veículo foi comandado por um operador através de seu controle remoto original.

4.3 ARRANJO EXPERIMENTAL

Os experimentos foram realizados em um heliponto localizado dentro do campus da Universidade Federal do Rio de Janeiro – UFRJ (FIG 4.10). As marcações geométricas pintadas no chão, a regularidade do solo e a ausência de qualquer tipo de interferência à visada do céu fizeram do heliponto o melhor lugar dentre os disponíveis para realização dos testes. As tomadas de dados foram realizadas em trajetórias definidas pelas marcas pintadas no solo do heliponto, mais precisamente no quadrado interno (em destaque na FIG 4.11) que possui 13,30 metros de lado. O veículo foi conduzido de forma que sempre permanecesse com as suas quatro rodas passando por cima das linhas do quadrado. Foram realizadas trajetórias ao longo de uma das retas (lado destacado em verde na FIG 4.11) e também trajetórias ao longo de todo o quadrado. Os dados coletados foram catalogados e guardados para posterior processamento e reconstrução das trajetórias.

FIG 4.10: Vista aérea do local dos testes.

FIG 4.11: Heliponto (google, 2012)

4.4 RESULTADOS OBTIDOS

4.4.1 TESTE PRELIMINAR EM CAMPO

Depois de processados os dados dos primeiros testes notou-se que o filtro de Kalman apresentava uma resposta enviesada, onde a trajetória reconstruída sempre tendia a se desviar para um lado.

A FIG 4.12 apresenta o resultado de reconstrução da trajetória através do filtro de Kalman de uma trajetória de ida e volta em uma das retas do heliponto.

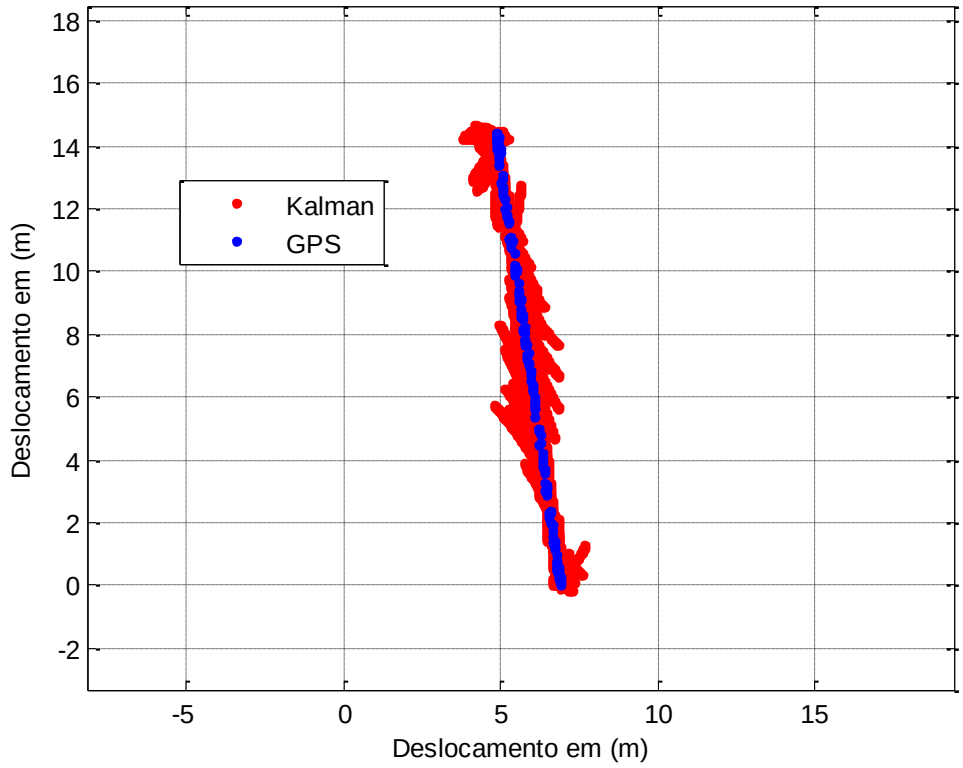

FIG 4.12: Trajetória de ida e volta na reta.

Com o intuito de verificar o real motivo dessa distorção na trajetória, foram investigados os ângulos de Euler do sinal obtido pela UMI. Os ângulos de Euler foram calculados através da matriz de orientação fornecida pela UMI da seguinte forma:

Considerando que a matriz de orientação seja uma matriz quadrada 3x3 M tal que:

$$M = \begin{bmatrix} M11 & M12 & M13 \\ M21 & M22 & M23 \\ M31 & M32 & M33 \end{bmatrix} \quad (4.4)$$

Os ângulos de Euler podem ser calculados través das equações (4.5), (4.6) e (4.7) (MICROSTRAIN, 2010).

$$Pitch = \theta = arcsen(-M13) \quad (4.5)$$

$$Roll = \phi = \arctan\left(\frac{M23}{M33}\right) \quad (4.6)$$

$$Yall = \psi = \arctan\left(\frac{M12}{M11}\right) \quad (4.7)$$

Os ângulos de Euler calculados para uma massa de dados coletados com o veículo parado sobre um cavalete e com o motor em funcionamento podem ser vistos na FIG 4.13. Esses ângulos mostram a existência de ruídos contaminando o sinal. Para que o sinal pudesse ser aproveitado no algoritmo, a fonte de ruído deveria ser eliminada e então foram investigadas as possíveis fontes de ruído. Analisando a documentação disponibilizada pelo fabricante da UMI, verificou-se que a matriz de orientação é calculada utilizando dados provenientes, entre outros, de uma bússola magnética instalada dentro do componente. Verificou-se também que o local escolhido para a instalação da unidade estaria muito próxima ao motor elétrico de propulsão do veículo.

É sabido que motores elétricos, durante o seu funcionamento, são uma grande fonte de ruídos eletromagnéticos e por esse motivo, deveriam estar influenciando nas bússolas internas da UMI. A solução encontrada foi trocar a UMI de lugar para verificar se as suspeitas se confirmavam. A unidade foi instalada no topo de uma espécie de mastro que por sua vez foi fixado na traseira do veículo conforme ilustra a FIG 4.14.

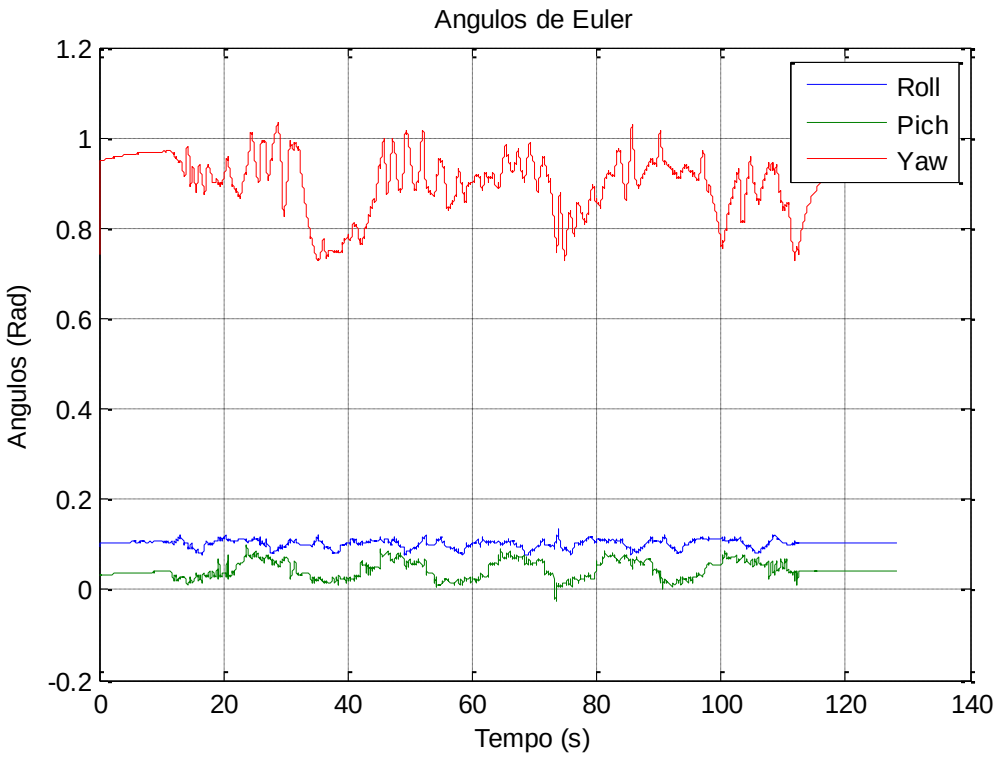

FIG 4.13: Ângulos de Euler calculados.

FIG 4.14: Mastro instalado com UMI em seu topo.

4.4.2 TESTES FINAIS EM CAMPO

Depois das alterações feitas a fim de minimizar os problemas relacionados aos ruídos induzidos pelo motor na UMI, foram realizadas novas tomadas de dados e novos processamentos. Os resultados relevantes serão apresentados a seguir.

4.4.2.1 TRAJETÓRIA ABERTA (RETA)

A tomada de dados foi realizada executando uma trajetória reta de ida e volta em marcha a ré, e todos os dados coletados foram guardados em um arquivo para posterior processamento. Os dados de deslocamento e velocidade coletados pelos encoders podem ser vistos na FIG 4.15.

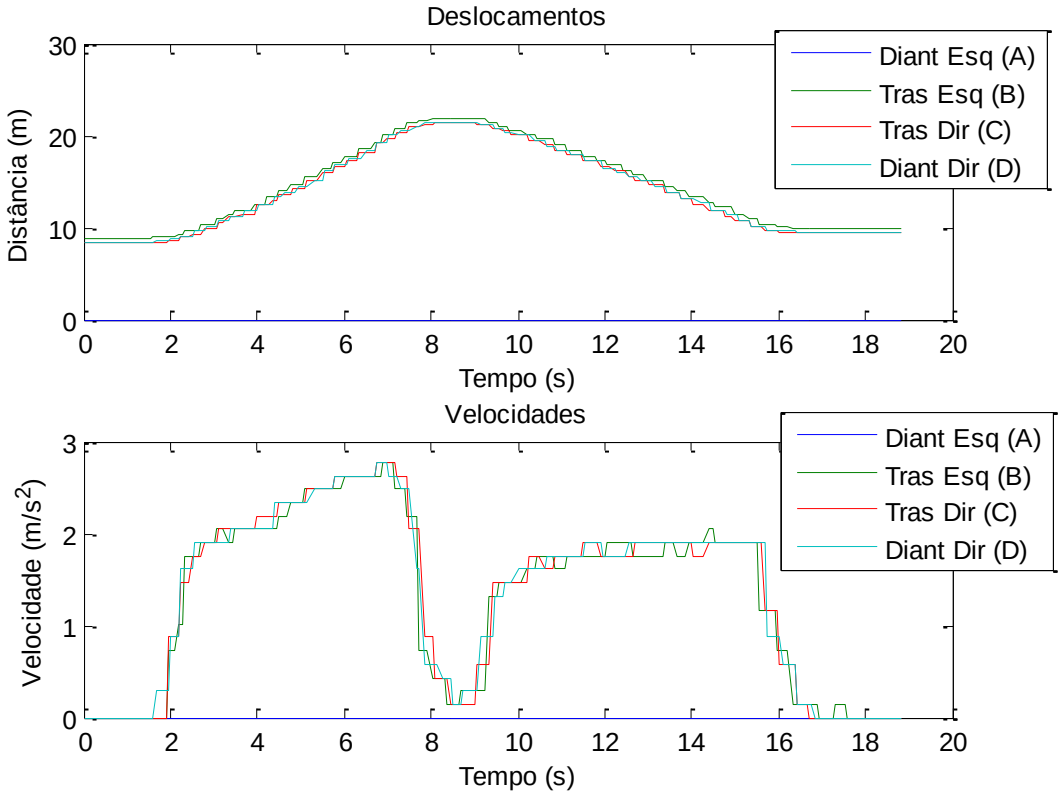

FIG 4.15: Dados de deslocamento e velocidade coletados pelos encoders.

Os dados de aceleração nos eixos x e y coletados através da UMI são apresentados na FIG 4.16 e os pontos coletados pelo GPS e convertidos para o sistema de coordenadas UTM são apresentados na FIG 4.17.

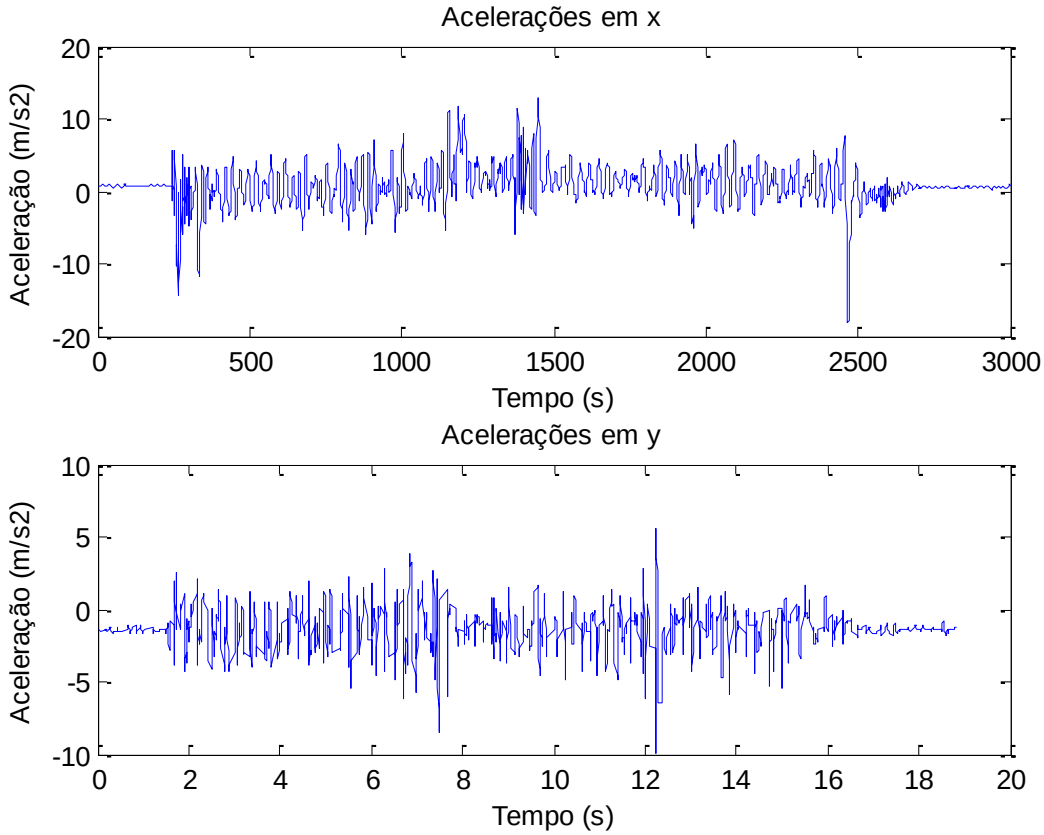

FIG 4.16: Dados de aceleração nos eixos x e y coletados através da UMI.

Os dados apresentados foram processados através de três métodos diferentes, apresentados a seguir. No primeiro método foram utilizados apenas os dados das acelerações e matriz de orientação disponibilizados pela UMI. Esses dados foram submetidos ao método de reconstrução de trajetória do duplo integrador e a trajetória reconstruída é apresentada na FIG 4.18.

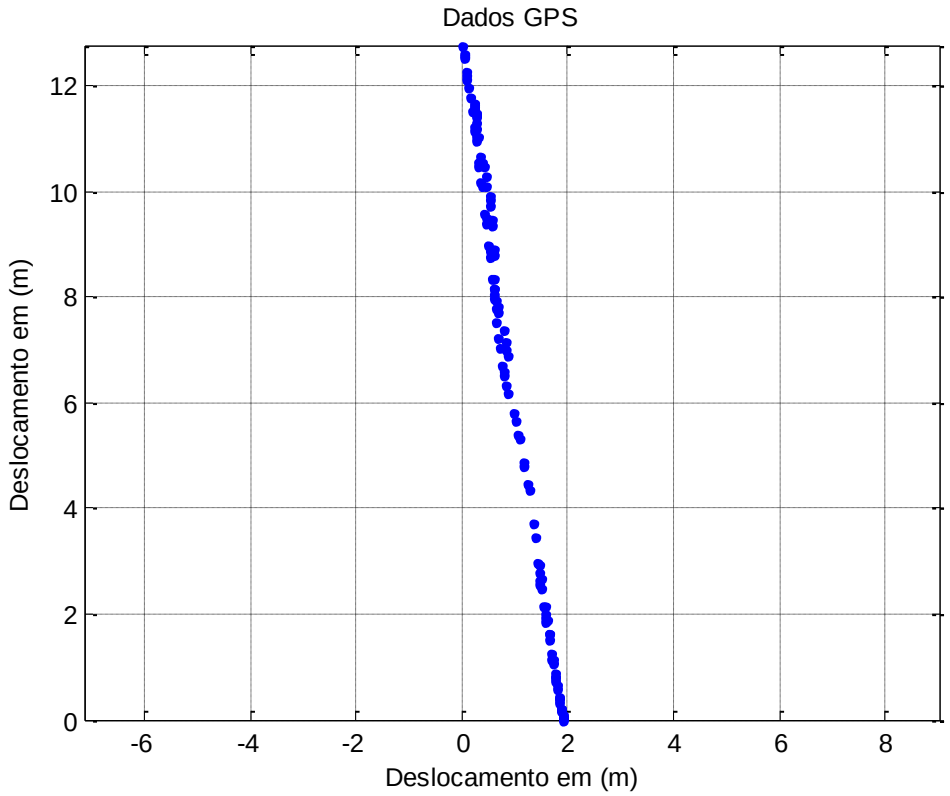

FIG 4.17: Pontos coletados pelo GPS e convertidos para o sistema de coordenadas UTM.

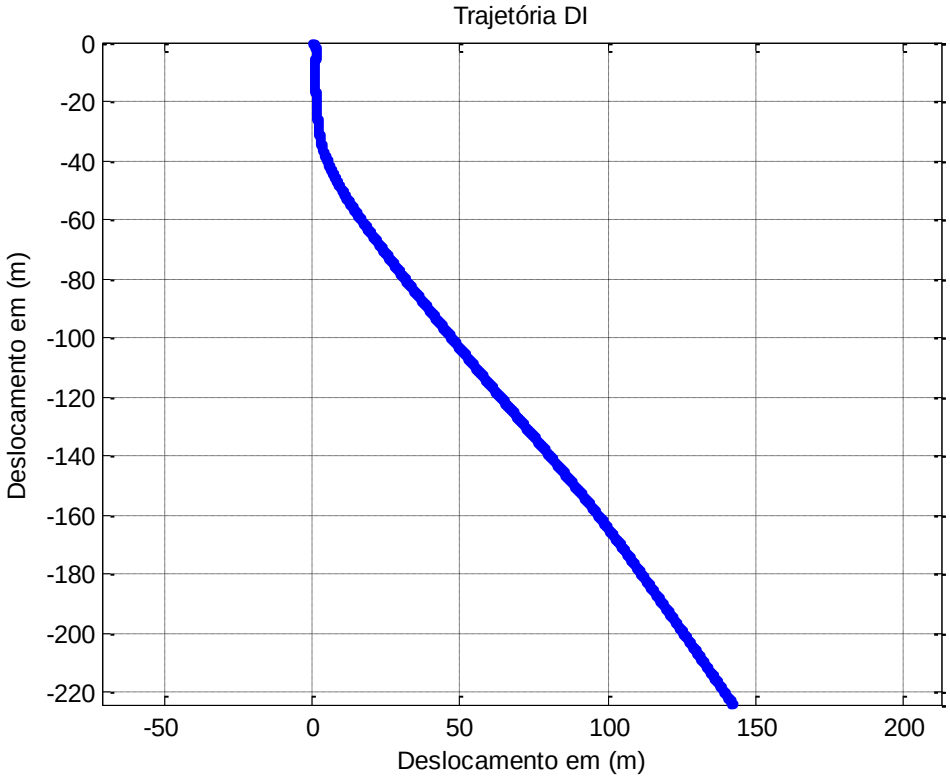

FIG 4.18: Trajetória reconstruída a partir do metodo do duplo integrador.

O segundo método consistiu na reconstrução da trajetória através de um filtro de Kalman utilizando os dados de aceleração e matriz de orientação provenientes da UMI, e os dados de velocidade dos encoders traseiros como medidas de referência. A trajetória reconstruída é apresentada na FIG 4.19.

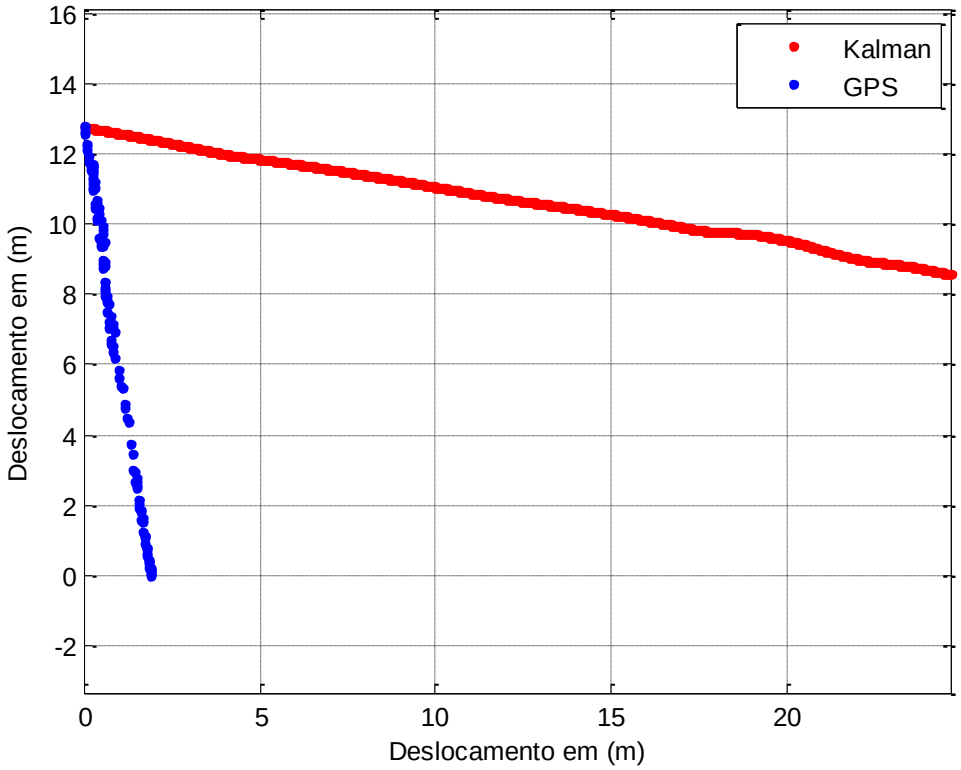

FIG 4.19: Trajetória reconstruída através do filtro de Kalman com encoder

O terceiro método consistiu na reconstrução da trajetória através de um filtro de Kalman utilizando além dos dados do encoder os dados do GPS como medida de referência. A trajetória reconstruída é apresentada na FIG 4.20.

As trajetórias reconstruídas através dos três métodos diferentes apresentaram resultados de baixa qualidade. Isso posto, decidiu-se por investigar melhor os resultados obtidos em outros formatos de trajetórias para tentar determinar a razão desses resultados.

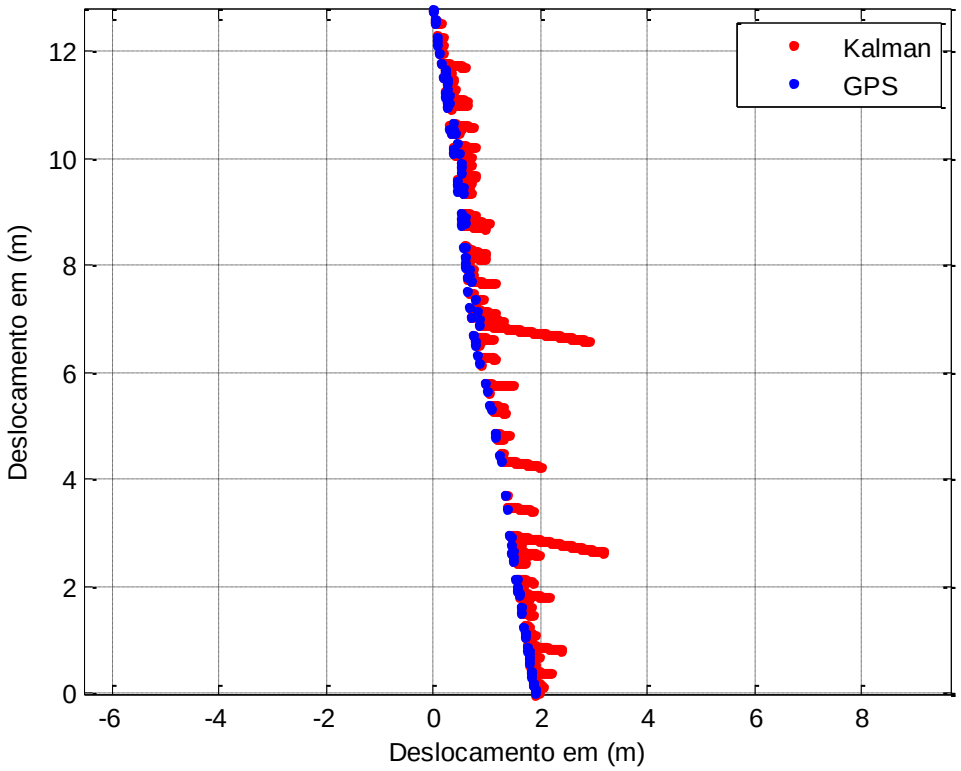

FIG 4.20: Trajetória aberta reconstruída através do filtro de Kalman com encoder e GPS.

4.4.2.2 TRAJETÓRIA FECHADA (QUADRADO)

A tomada de dados foi realizada executando uma trajetória fechada ao longo do quadrado interno demarcado no solo do heliponto. Analogamente ao item 4.4.2.1, todos os dados coletados foram guardados em um arquivo para posterior processamento e serão apresentados a seguir.

Os dados de deslocamento e velocidades coletados pelos encoders podem ser verificados nos gráficos apresentados na FIG 4.21.

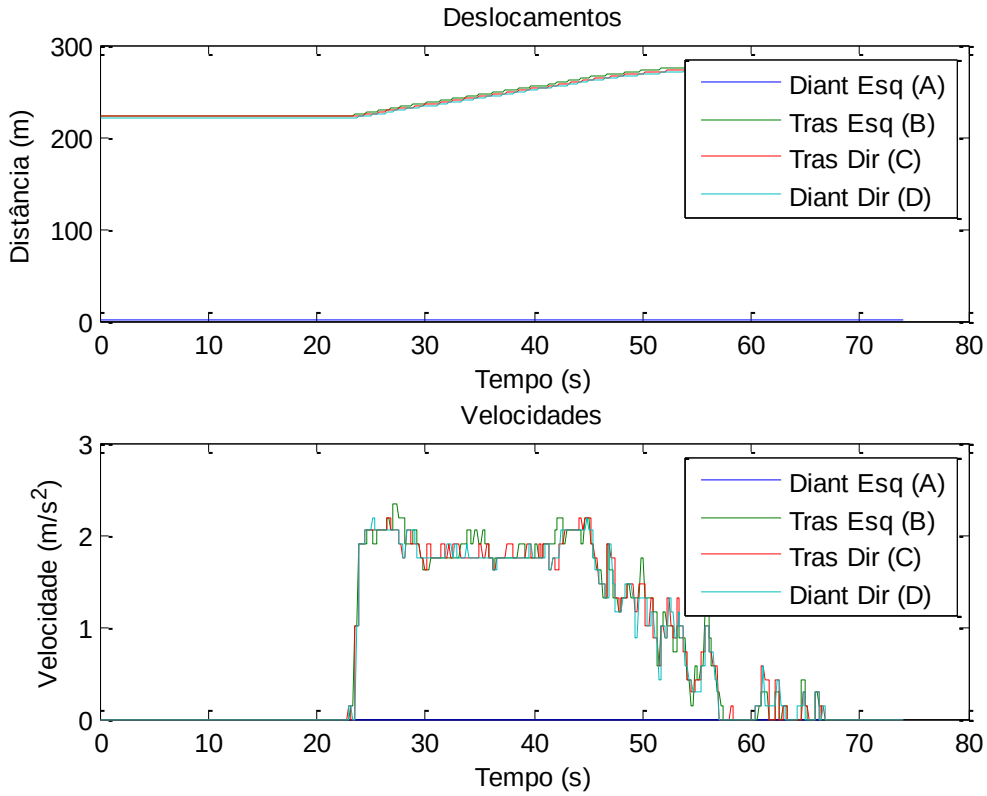

FIG 4.21: Dados de deslocamento e velocidades coletados pelos encoders.

Os dados de aceleração nos eixos x e y coletados através da UMI estão apresentados na FIG 4.22 e os pontos coletados pelo GPS convertidos para o sistema de coordenadas UTM estão apresentados na FIG 4.23.

Analogamente ao caso da trajetória aberta, os dados apresentados acima foram processados através de três métodos diferentes e serão apresentados a seguir. No primeiro método foram utilizados apenas os dados das acelerações e matriz de orientação entregues pela UMI. Esses dados foram submetidos ao método de reconstrução de trajetória do duplo integrador e a trajetória reconstruída é apresentada na FIG 4.24.

No segundo método a trajetória foi reconstruída através do filtro de Kalman utilizando os dados coletados pela UMI e os dados de velocidade coletados através dos encoders traseiros foram empregados como atualização de medida. A trajetória reconstruída através do filtro de Kalman é apresentada na FIG 4.25.

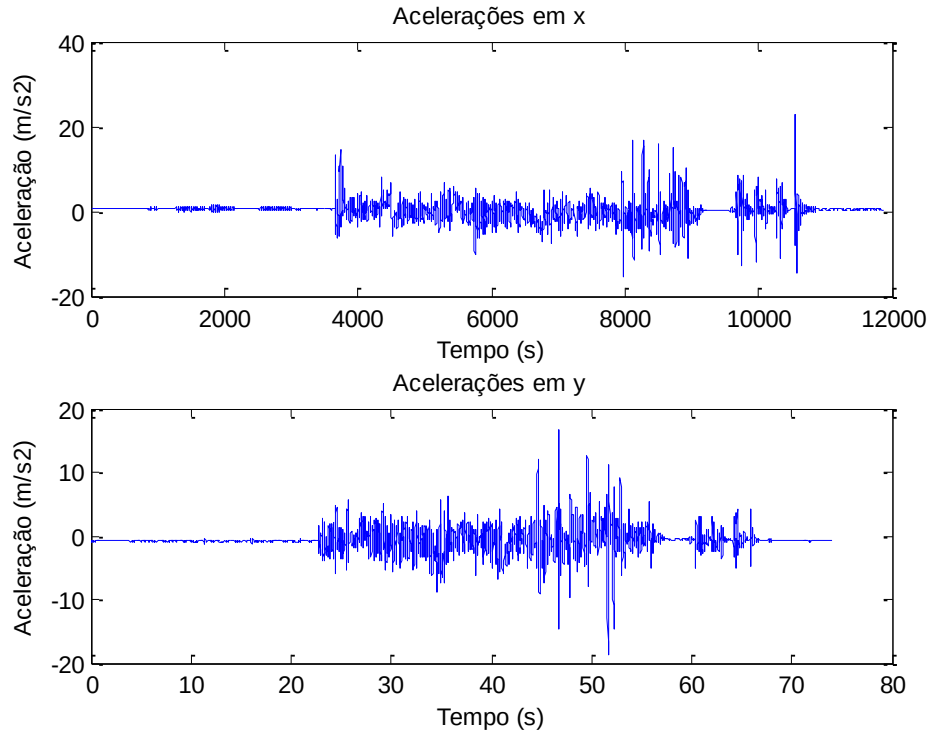

FIG 4.22: Dados de acelerações em x e y para trajetória fechada.

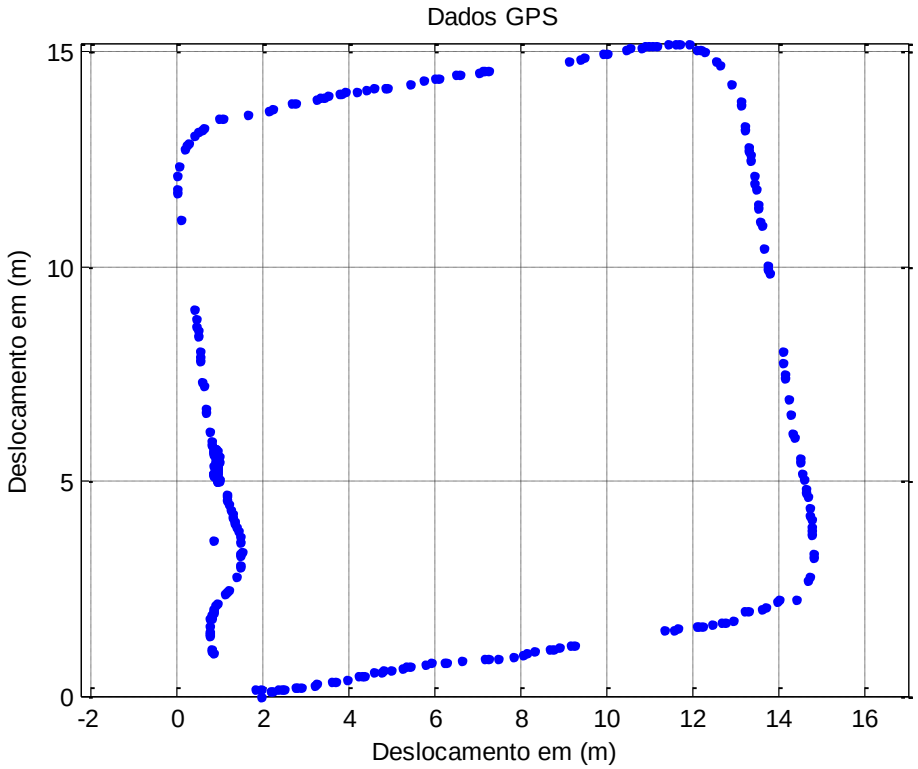

FIG 4.23: Dados de GPS para trajetória fechada.

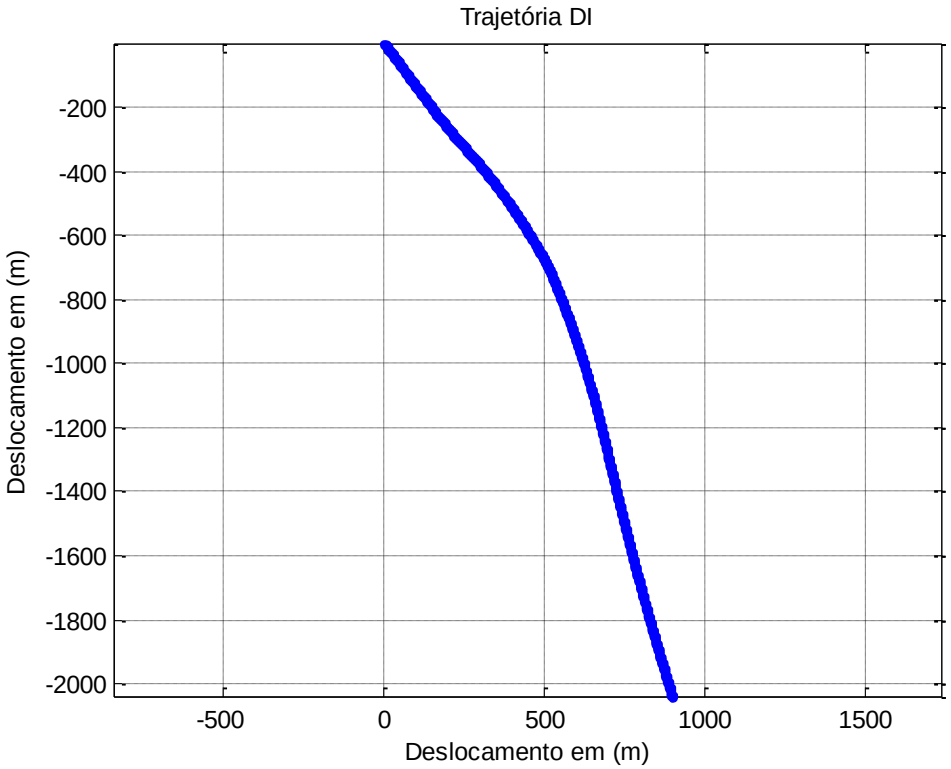

FIG 4.24: Trajetória fechada reconstruída através do método do duplo integrador.

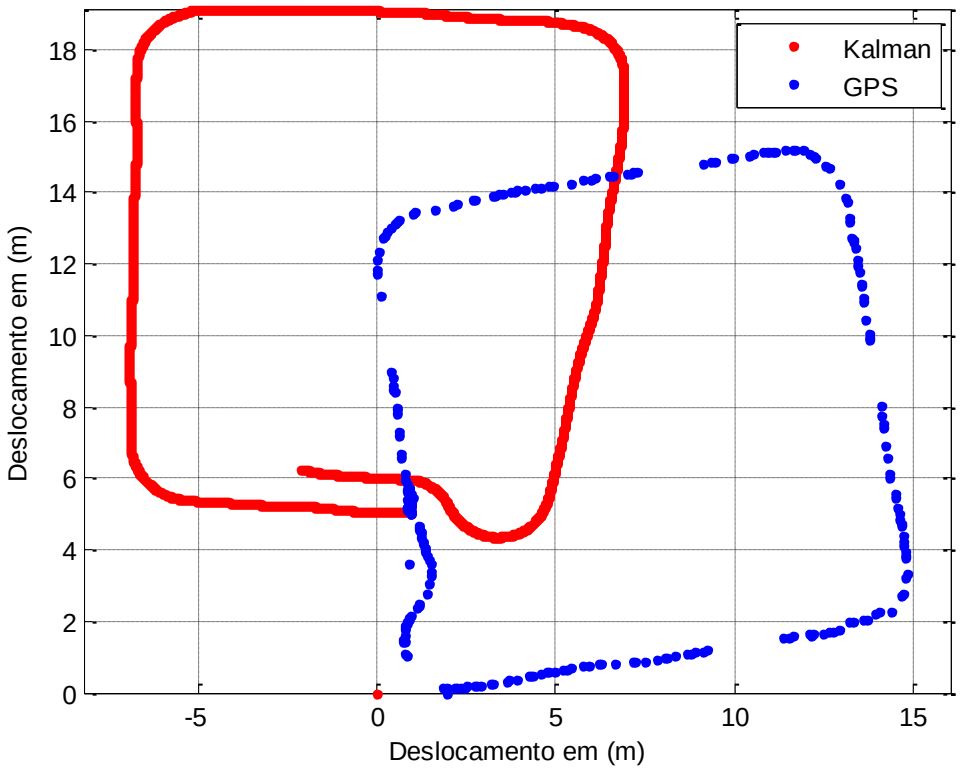

FIG 4.25: Trajetória fechada reconstruída através do filtro de Kalman.

Com a reconstrução da trajetória apresentada na FIG 4.25, foi possível verificar que havia um desalinhamento presente na matriz de orientação entregue pela UMI. O motivo deste desalinhamento não é conhecido, contudo, acredita-se ser uma composição de desalinhamento da fixação da UMI no veículo, uma pequena parcela de declinação magnética (inconstância que ocorre entre as marcações da bússola e a geográfica definido pela posição astronômica) e outra parcela devido aos erros de leitura inerentes a qualidade da bússola magnética interna da UMI.

Para resolver este problema de alinhamento acrescentou-se ao algoritmo responsável pela transformação de referencial definido na equação (3.9), uma matriz de rotação responsável por alinhar os dados de aceleração e de velocidade como a apresentada na equação (4.8), onde ϕ é o ângulo em radianos em que a rotação vai ser executada.

$$\begin{bmatrix} \cos(\phi) & -sen(\phi) & 0 \\ sen(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.8}$$

O valor de ϕ foi definido por tentativa e erro, sempre comparando a trajetória reconstruída com os pontos recebidos pelo GPS. O resultado da trajetória reconstruída depois do alinhamento e utilizando apenas os dados dos encoders como atualização podem ser vistos na FIG 4.26.

O terceiro método aplicado consiste na reconstrução da trajetória através de um filtro de Kalman utilizando além dos dados do encoder os dados do GPS como medida de referência. A introdução dos dados do GPS visa minimizar o acúmulo de erros dos sensores relativos, visto que o mesmo apresenta medidas absolutas.

A trajetória reconstruída através do filtro de Kalman utilizando dados de GPS é apresentada na FIG 4.27.

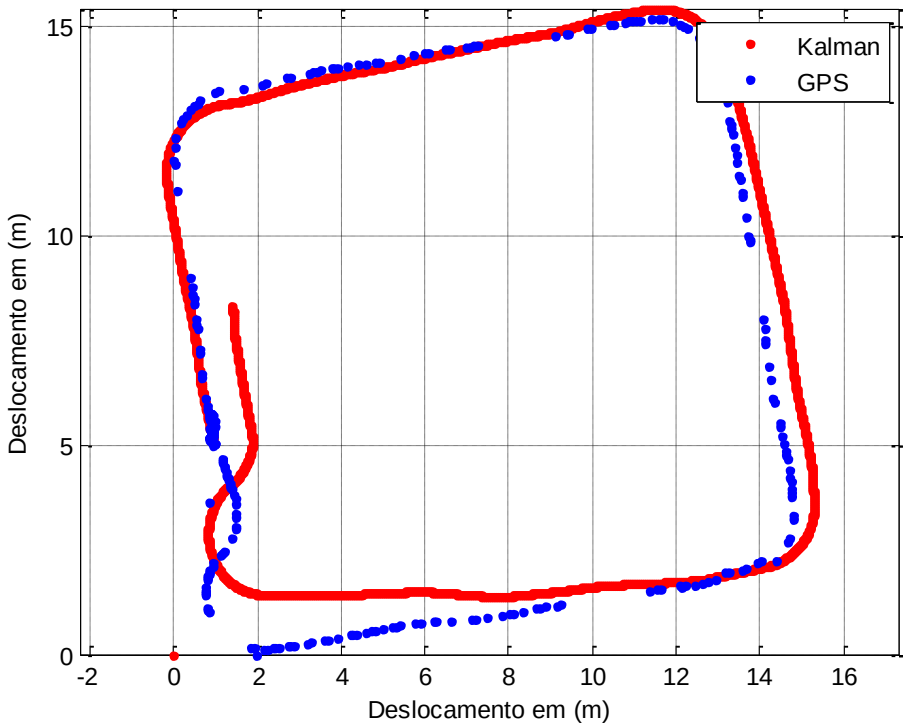

FIG 4.26: Trajetória fechada reconstruída através do filtro de Kalman com encoder (caso1).

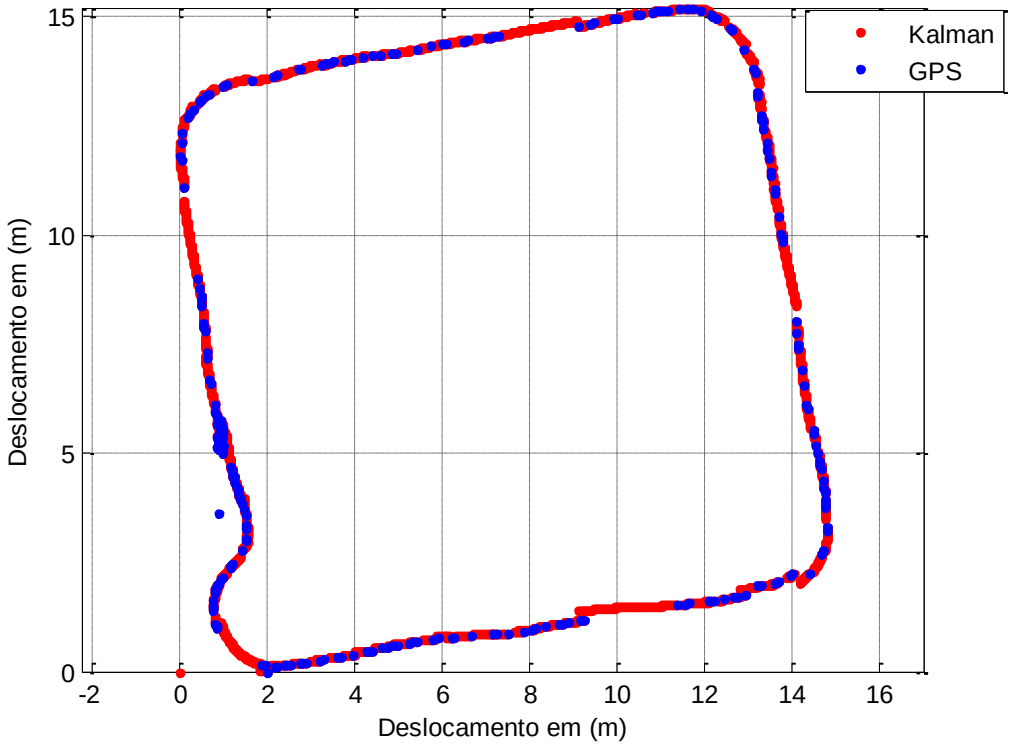

FIG 4.27: Trajetória fechada reconstruída através do filtro de Kalman com encoder e GPS (caso 1).

Os mesmos algoritmos de reconstrução de trajetórias por filtro de Kalman utilizando como entradas de referência apenas os dados do encoder e os dados do encoder e do GPS foram repetidos para diversas tomadas de dados e serão apresentadas nas FIG FIG 4.28, FIG 4.29, FIG 4.30, FIG 4.31,FIG 4.32 e FIG 4.33.

As tomadas de dados foram realizadas nas mesmas condições das apresentadas anteriormente com exceção da última tomada, apresentada nas FIG FIG 4.32 e FIG 4.33, onde o veículo realizou a trajetória no sentido anti-horário e completou duas voltas ao invés de apenas uma como nos casos anteriores.

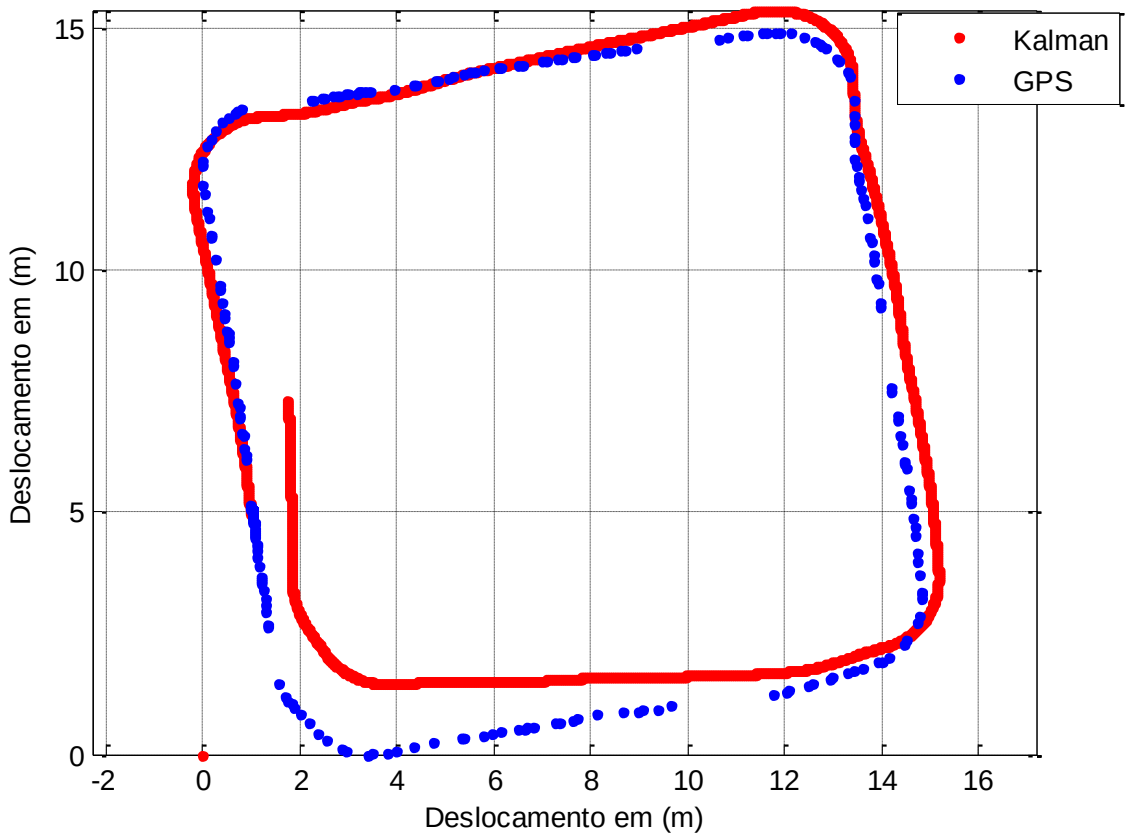

FIG 4.28: Trajetória quadrada com apenas encoder como medida de referência (caso 2).

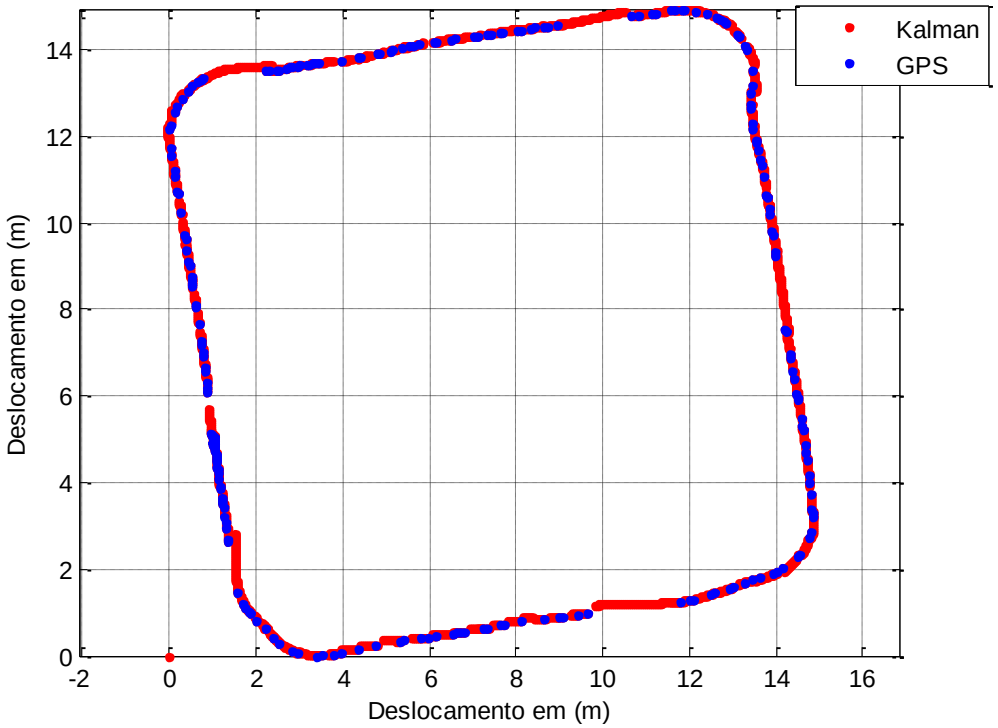

FIG 4.29: Trajetória quadrada com encoder e GPS como medidas de referência (caso 2).

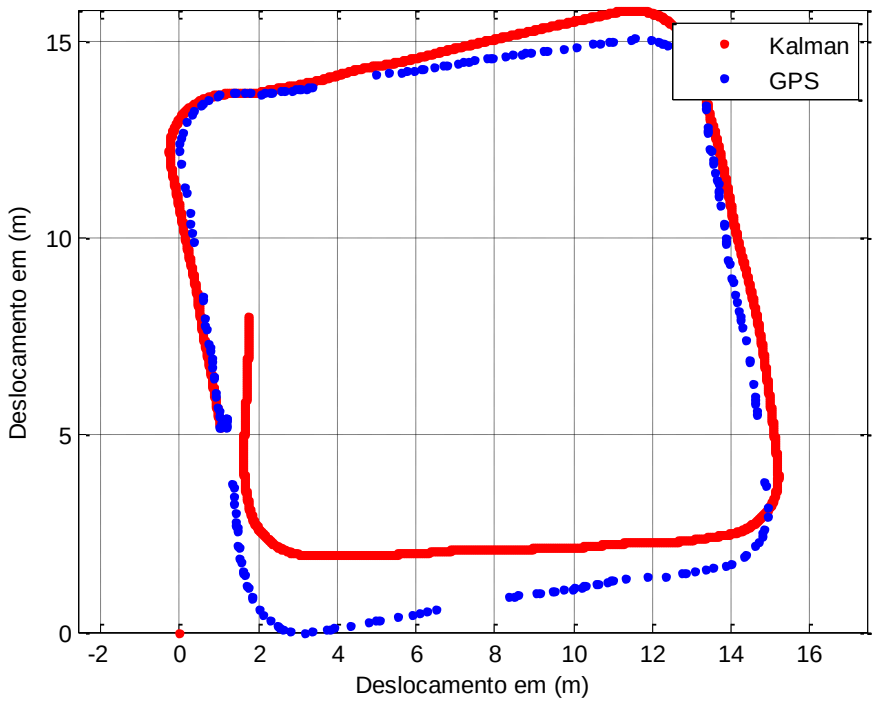

FIG 4.30: Trajetória quadrada com apenas encoder como medida de referência (caso 3).

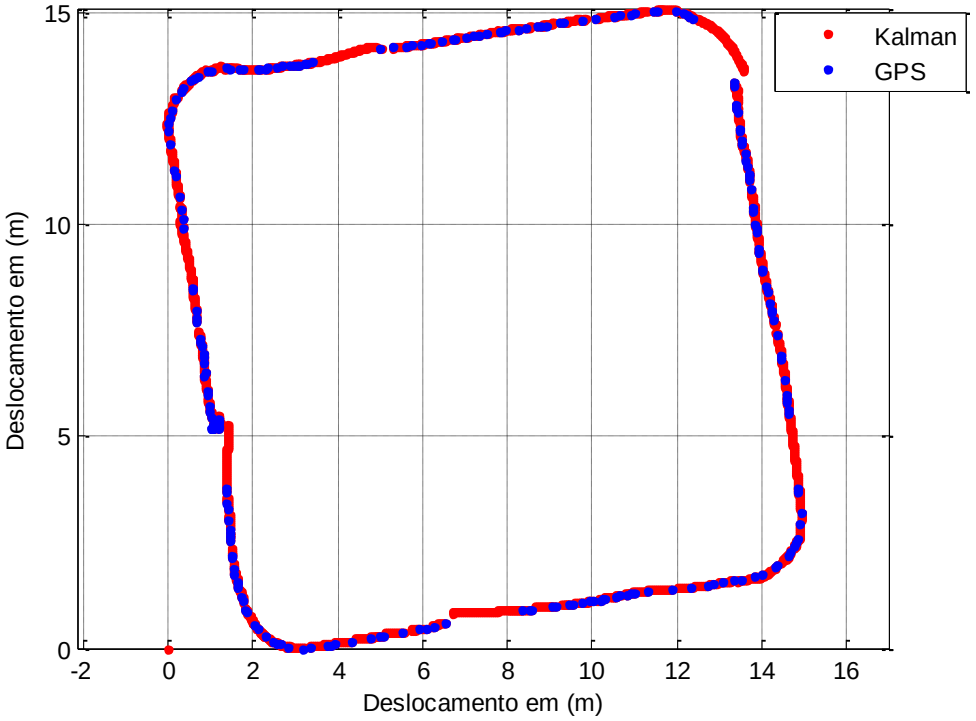

FIG 4.31: Trajetória quadrada com encoder e GPS como medidas de referência (caso 3).

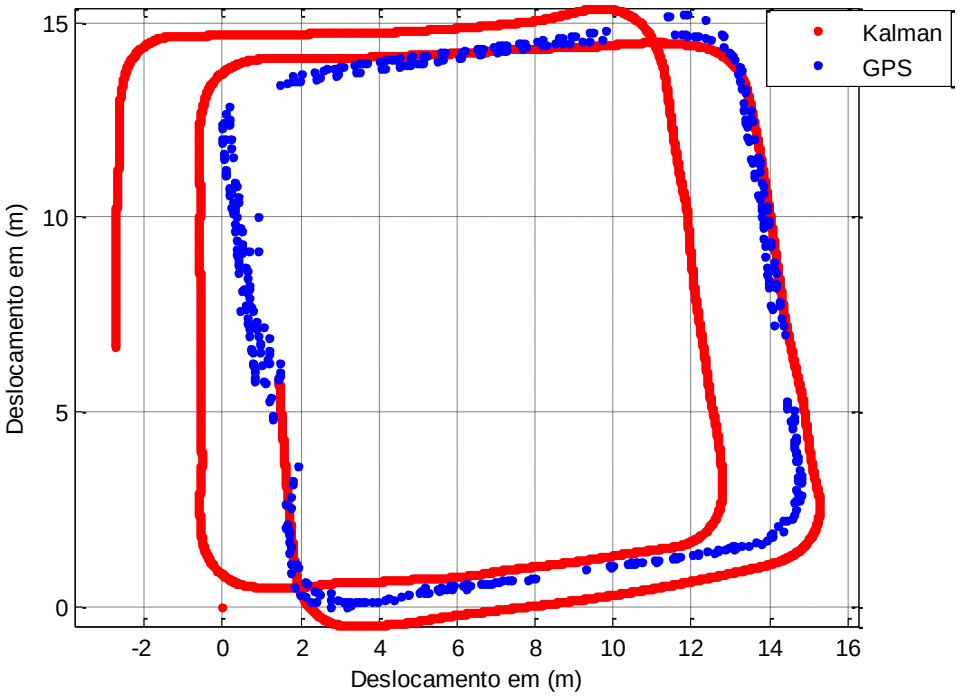

FIG 4.32: Trajetória quadrada com apenas encoder como medida de referência (caso 4).

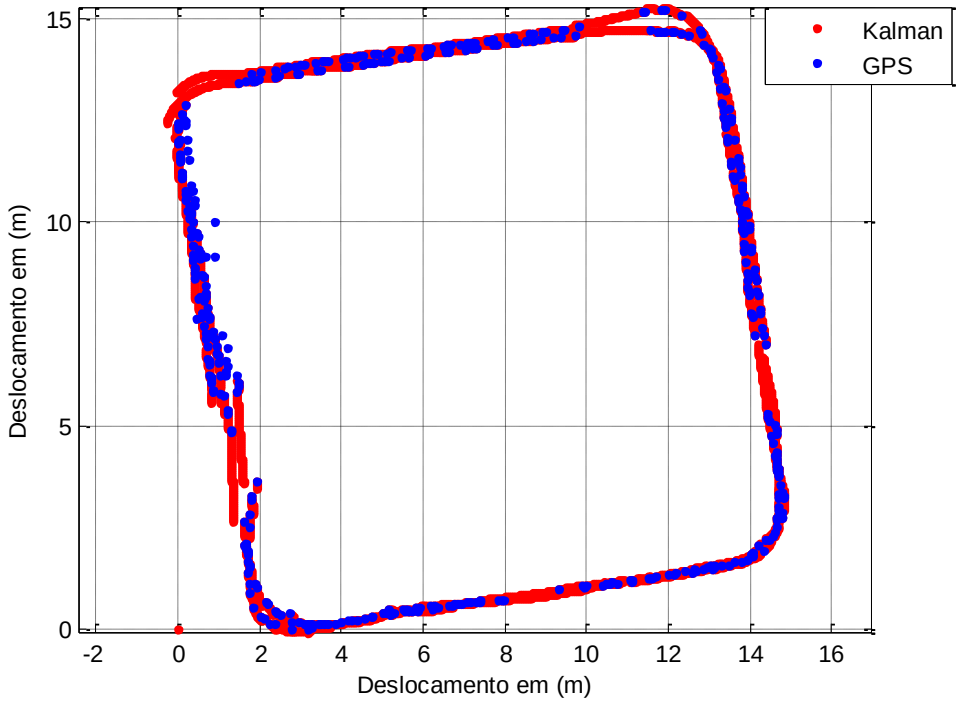

FIG 4.33: Trajetória quadrada com encoder e GPS como medidas de referência (caso 4).

Os resultados obtidos após o alinhamento inicial se mostraram satisfatórios. Os erros apresentados nos casos onde não foram utilizadas as medidas de GPS como medidas de atualização provêm da utilização de apenas sensores relativos. Tais sensores sofrem com o acúmulo de erros e estes erros podem ser minimizados com a introdução de uma entrada de algum sensor cujos dados sejam absolutos, como é o caso do GPS. Comparando as trajetórias reconstruídas a partir da combinação dos dados coletados pelo encoder, UMI e GPS com os pontos coletados pelo GPS, é possível notar que neste caso os erros acumulativos foram bastante minimizados e pode-se afirmar que a mesma é uma boa estimativa da real trajetória do veículo.

5 DISCUSSÃO E CONCLUSÕES

5.1 VISÃO GERAL

Esse trabalho teve como objetivo o projeto e construção de uma plataforma robótica móvel e o estudo de algoritmos de fusão sensorial apropriados para a combinação de sensores absolutos e relativos com características e taxas de aquisição diferentes, a fim de implementar um sistema computacional de reconstrução de trajetórias de um automóvel a partir de dados ruidosos. As etapas de desenvolvimento desse trabalho passaram pela escolha das grandezas a serem medidas, especificação dos sensores capazes de medir essas grandezas, projeto de circuitos dedicados à integração dos sensores, especificação do veículo utilizado, instalação física dos sensores, integração com o microcomputador, desenvolvimento de software capaz de coletar e armazenar os dados e a implementação de algoritmos para a fusão sensorial desses dados.

Os sensores utilizados para a medição das variáveis foram um receptor de GPS com a capacidade de receber correções diferenciais, uma unidade de medição inercial, que fornece orientação, aceleração linear e velocidade angular do veículo, um circuito dedicado à leitura do ângulo de esterçamento das rodas dianteiras e mais quatro encoders (um em cada roda) que, ligados aos seus respectivos circuitos eletrônicos, são capazes de fornecer a velocidade e o deslocamento de cada uma das rodas do veículo.

5.2 RESULTADOS ALCANÇADOS

Os resultados das reconstruções de trajetórias apresentados no Capítulo 4 mostram que é possível se estimar a localização de um veículo utilizando sensores com medidas ruidosas e baixas taxas de amostragem.

O primeiro método apresentado, o do duplo integrador, apresentou um desempenho muito ruim para todos os casos testados. Seus resultados apresentam

trajetórias com erros de milhares de metros devido à dependência unicamente dos sinais proveniente da UMI que se mostraram muitos ruidosos. A estimação da trajetória através do filtro de Kalman utilizando apenas os dados de velocidade provenientes do encoder como medidas de referência, se mostrou eficiente apenas por um curto espaço de tempo. Depois de algum tempo e sucessivas integrações, o ruído proveniente do sinal da UMI contamina o sinal desviando a trajetória.

O terceiro método apresentado utiliza o encoder como medida de referência no intervalo entre cada uma das leituras do GPS, que possui uma frequência de amostragem bem menor que os outros sensores (20Hz). Ou seja, em cada iteração o sistema verifica se existe uma nova leitura de GPS: se houver, a mesma é utilizada, caso contrário, o sistema emprega a leitura de velocidade proveniente dos encoders como medida de referência.

Esse método se mostrou viável para a reconstrução de trajetórias, permitindo que fossem estimados pontos entre cada leitura de GPS e, com isso, obter uma informação de posição com uma frequência superior do que apenas o GPS isoladamente.

Por algum motivo ainda desconhecido os dados gerados pelo receptor de GPS sofriam algumas interrupções temporárias que podem ser observadas na FIG 4.23. Essas interrupções podem ser resultantes de alguma característica do sensor, do *software* de aquisição utilizado e até mesmo do sistema utilizado para a transmissão dos dados entre o veículo e o computador.

Essas interrupções não prejudicaram o trabalho, pelo contrário, elas serviram para verificar o bom desempenho do filtro de Kalman durante períodos mais longos sem recepção dos dados de GPS.

5.3 SUGESTÃO DE TRABALHOS FUTUROS

Várias melhorias poderiam ser incorporadas à plataforma robótica móvel, como a adição de novos sensores ou a troca de sensores de baixa qualidade por sensores de mais alta qualidade, além disso, poderiam ser feitas modificações no algoritmo de fusão sensorial.

O uso do GPS como único sensor de posição absoluta faz com que o sistema dependa fortemente das medidas desse sensor, que, por sinal, é bastante suscetível a interferências. A inclusão de novos sensores de posição absoluta tornaria o sistema mais robusto frente aos erros provenientes do sistema de GPS.

A utilização de encoders mais precisos permitiria utilizar modelos baseados nas diferenças de rotação de cada uma das rodas para determinar o deslocamento relativo do veículo. A desvantagem desse método é a alta sensibilidade aos deslizamentos das rodas e a variação do diâmetro dos pneus que acontecem com o seu desgaste natural.

A utilização de um sensor de orientação que não seja baseado no campo magnético da Terra proporcionaria uma melhoria à estimação da orientação do veículo e, consequentemente, melhoraria a estimação da posição.

Outros algoritmos de fusão sensorial aliados a novas propostas de modelos poderiam ser utilizados para a localização do veículo, como por exemplo, o filtro de Kalman estendido. O filtro de Kalman é uma solução para o problema de estimar os estados de um sistema dinâmico estocástico de dimensão infinita *linear*. Para sistemas não lineares, o filtro de Kalman não é, a rigor aplicável, pois a hipótese de linearidade é necessária para que a estimação seja ótima. O filtro de Kalman estendido procura transpor esta dificuldade utilizando uma linearização em torno da estimativa do estado.

Outra possibilidade seria a de utilização de mapas com informações geográficas pré-carregados no sistema. Nesse caso, o mapa seria utilizado como mais uma informação de entrada para a fusão sensorial.

6 REFERÊNCIAS BIBLIOGRAFICAS

AGUIRRE, L. A. **Introdução à identificação de sistemas:** técnicas lineares e não-lineares aplicadas a sistemas reais. 3. ed. Belo Horizonte: Editora UFMG, 2007.

BUENO, S. S. et al. **UMA PLATAFORMA PARA PESQUISA E DESENVOLVIMENTO EM ROBÓTICA TERRESTRE DE EXTERIOR**. Simpósio Brasileiro de Automação Inteligente. Brasilia: [s.n.]. 2009.

CLARK, D.; OWINGS, M. **Building Robot Drive Trains**. 1ª. ed. New York: McGraw-Hill, 2003.

COORDENAÇÃO DE APERFEIÇOAMENTO DE PESSOAL DE NÍVEL SUPERIOR - CAPES. CAPES Pró Defesa, 2008. Disponivel em: <http://www.capes.gov.br/bolsas/programas-especiais/pro-defesa>. Acesso em: 2010 Outubro 2010.

DALBERTO, L. F. A. **DGPS EM REDE: DESENVOLVIMENTO E IMPLANTAÇÃO VIA INTERNET UTILIZANDO A REDE GNSS DO ESTADO DE SÃO PAULO**. Universidade Estadual Paulista. Presidente Prudente, p. 92. 2010.

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY - DARPA. DARPA Urban Challenge, 2007. Disponivel em: <http://www.darpa.mil/grandchallenge/index.asp>. Acesso em: 1 Outubro 2010.

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY - DARPA, 2010. Disponivel em: <http://www.darpa.mil>. Acesso em: 01 Outubro 2010.

GUSTAVO EMMENDOERFER, R. A. L.; SANTOS, W. E. D.; GULES, R. SISTEMA DE RECONSTRUÇÃO DE TRAJETÓRIAS DE PIG'S BASEADO EM MEDIDAS GPS/GIROSCÓPIOS/ODÔMETROS. **Congresso Brasileiro de Automática**, Juiz de Fora, 2008.

IBGE. Instituto Brasileiro de Geografia e Estatística, 2011. Disponivel em: <http://www.ibge.gov.br/home/geociencias/geodesia/rbmc/ntrip/>. Acesso em: 28 Janeiro 2011.

JIN, W.; ZHAN, X. **A Modified Kalman Filtering via Fuzzy Logic System for ARVs Location**. International Conference on Mechatronics and Automation. Harbin, China: IEEE. 2007. p. 711 - 716.

JUNIOR, C. J. A. F. **SIMULAÇÃO E IMPLEMENTAÇÃO EM TEMPO REAL DE SISTEMAS DE NAVEGAÇÃO INERCIAL INTEGRADOS INS/GPS**. Instituto Militar de Engenharia. Rio de janeiro, p. 112. 2009.

KEE, G. M.; ZAIN, Z. M.; SALIMIN, R. H. **Design and Development PIC-Based Autonomous Robot**. International Conference on Robotics, Automation and Mechatronics. Chengdu, CHINA: IEEE. 2008. p. 1-5.

MICROSTRAIN. **3DM-GX2 Data Communications Protocol**. Version 1.19. ed. Williston: MicroStrain, Inc., 2010.

MICROSTRAIN, 2011. Disponivel em: <http://www.microstrain.com/inertial/sensors>. Acesso em: 1 dezembro 2011.

MOREIRA, A. L. **Metodologia de Avaliação de Modelos em Escala de Veículos Militares**. Instituto Militar de Engenharia. Rio de janeiro, p. 224. 2011.

NOVATEL, 2011. Disponivel em: <http://www.novatel.com/assets/Documents/Papers/SMARTV1_ant.pdf>. Acesso em: 15 Abril 2011.

REVISTA DA MARINHA. SEACON – Marinha incorpora 3 novos submarinos autónomos. **Revista da Marinha**, Portugal, Julho 2011.

SANTANA, D. D. S. **Estimação de trajetórias terrestres utilizando unidade de medição inercial de baixo custo e fusão sensorial**. Escola Politécnica da Universidade de São Paulo. São Paulo, p. 138. 2005.

SANTOS, M. M. **Desenvolvimento de um Sistema de Localização e Recostrução de Trajetórias para um Veículo Terrestre**. Universidade Federal de Minas Gerais. Minas Gerais, p. 136. 2009.

SARIFF, N.; BUNIYAMIN, N. **An Overview of Autonomous Mobile Robot Path Planning Algorithms**. 4th Student Conference on Research and Development (Scored 2006). [S.l.]: [s.n.]. 2006. p. 183-188.

SCHUBERT, R.; MATTERN, N.; WANIELIK, G. **Comparison of Low-Cost GPS/INS Sensors for Autonomous Vehicle Applications**. Position, Location and Navigation Symposium, 2008 IEEE/ION. Monterey, CA: [s.n.]. 2008. p. 1133-1144.

TITTERTON, D. H.; WESTON, J. L. Strapdown Inertial Navigation Technology. **IEE Radar, Sonar, Navigation and Avionics Series 5**, London, 1997.

XIE, M. et al. **Development of Navigation System for Autonomous Vehicle to Meet the DARPA Urban Grand Challenger**. Intelligent Transportation Systems Conference. Seattle, WA, USA: IEEE. 2007. p. 767-772.

7 ANEXOS

7.1 FIRMWARE DE LEITURA DOS ENCODERS

encoder.c

```
#include "encoder.h"
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#define ADDR   'D'
#define cte_roda 0.01523672437

int32 count, count_old, delta;
float velocity;

#int_EXT
void  EXT_isr(void)
{
  if(input(CHB)==0)
    count ++;
  else
     count --;
}

#int_RTCC
void  RTCC_isr(void)
{
  delta = (count-count_old);
  if (count < count_old)
     delta = -(count-count_old);

  velocity = ((delta * cte_roda)/0.104);
  count_old = count;
}


void main()
{

  setup_adc_ports(AN0);
  setup_adc(ADC_CLOCK_INTERNAL);
  setup_spi(SPI_SS_DISABLED);
  setup_wdt(WDT_OFF);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_8);
  setup_timer_1(T1_DISABLED);
  setup_timer_2(T2_DISABLED,0,1);
  setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
  enable_interrupts(INT_EXT);
  enable_interrupts(INT_RTCC);
```

```
        enable_interrupts(GLOBAL);
        set_adc_channel(0);

        count = 0;
        count_old = 0;

        char COMMAND;
        int16  value;

        while(true)
        {
          char dado;
          rs232_errors=0;
          while(getch() != ADDR) restart_wdt();

          COMMAND=getch();
          switch (COMMAND)  // verifica a tecla
          {
            case 'd' :     // Envia contagem do encoder.
                     printf("contagem = %ld\r\n",count);
                     printf("velocidade = %f\r\n",velocity);
                     /*printf("delta = %ld     count = %ld     count_old = %ld\r\n",
delta,count,count_old);
                     value = read_adc();
                     delay_ms(50);
                     printf("ADC = %ld\r\n", value);*/
            break;

            case 'e' :     // Envia contagem do encoder.
                     printf("$%cE,%ld\n",ADDR,count);
            break;

            case 'v' :     // Envia valor de velocidade.
                     printf("$%cV,%f\n",ADDR,velocity);
            break;

            case 'f' :     // Envia contagem do encoder e velocidade.
                     printf("$%cF,%ld,%f\n",ADDR,count,velocity);
            break;

            case 't' :
                     value = read_adc();
                     delay_ms(5);
                     float tension;
                     tension = ((0.0145*value)+0.0736);
                     printf("$%cT,%f\n",ADDR,tension);
            break;

            case 'r' :     // Envia contagem do encoder e velocidade.
```

```
                count=0;
          break;
        }

      }
    }

    //  Diametro da Roda 97mm
    //  Intervalo de interrupção T0 = 104ms
    //  encoder 20 ppv
    //  equação de ajuste do AD   y = 0.0145x + 0.0736
```

encoder.h

```
#include <18F252.h>
#device adc=10

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES WDT128                   //Watch Dog Timer uses 1:128 Postscale
#FUSES HS                       //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for
PCD)
#FUSES NOPROTECT                //Code not protected from reading
#FUSES NOOSCSEN                 //Oscillator switching is disabled, main oscillator is
source
#FUSES NOBROWNOUT               //No brownout reset
#FUSES BORV20                   //Brownout reset at 2.0V
#FUSES PUT                      //Power Up Timer
#FUSES STVREN                   //Stack full/underflow will cause reset
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18)
used for I/O
#FUSES NOWRT                    //Program memory not write protected
#FUSES NOWRTD                   //Data EEPROM not write protected
#FUSES NOWRTB                   //Boot block not write protected
#FUSES NOWRTC                   //configuration not registers write protected
#FUSES NOCPD                    //No EE protection
#FUSES NOCPB                    //No Boot Block code protection
#FUSES NOEBTR                   //Memory not protected from table reads
#FUSES NOEBTRB                  //Boot block not protected from table reads

#define CHB PIN_B5

#use delay(clock=20000000,RESTART_WDT)
#use
rs232(baud=57600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8,enable=PIN_C5,errors
,restart_wdt)
```

7.2 FIRMWARE CIRCUITO DO CONTROLE REMOTO

Controle.c

```
#include "Controle.h"
#include <stdlib.h>
//#include <input.c>

void get_string(char* s, unsigned int8 max) {
  unsigned int8 len;
  char c;

  --max;
  len=0;
  do {
    c=getc();
    if(c==8) {  // Backspace
       if(len>0) {
         len--;
       }
    } else if ((c>=' ')&&(c<='~'))
      if(len<=max) {
        s[len++]=c;
        //putc(c);
      }
  } while(c!=13);
  s[len]=0;
}

signed int16 get_long()
{
  char s[7];
  signed int16 l;

  get_string(s, 7);
  l=atol(s);
  return(l);
}

void main()
{
   int16 d_value, a_value;
   setup_adc_ports(AN0);
   setup_adc(ADC_CLOCK_INTERNAL);
```

```
setup_spi(SPI_SS_DISABLED);
setup_wdt(WDT_OFF);
setup_timer_0(RTCC_INTERNAL);
setup_timer_1(T1_DISABLED);
setup_timer_2(T2_DIV_BY_4,124,1);
setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
setup_ccp1(CCP_PWM);
setup_ccp2(CCP_PWM);

d_value = 206;
a_value = 204;

set_pwm1_duty(d_value);
set_pwm2_duty(a_value);
set_adc_channel(0);

output_low(pin_b6);
output_low(pin_b7);


while(true)
{
   int16 value = 0;
   char command;

   rs232_errors = 0;
   command = getch();

   if ((command == 'a')||(command == 'd'))
      value=get_long();

   if (command == 'a')
   {
      if (value << 110)
         value = 110;
      if (value >> 290)
         value = 290;
      set_pwm2_duty(value);
   }

   else if(command == 'd')
   {
      if (value << 50)
         value = 50;
      if (value >> 380)
         value = 380;
      set_pwm1_duty(value);
   }
   else if(command == 'b')
```

```
        {
          value = read_adc();
          delay_ms(50);
          printf("%ld", value);
        }
      }
    }
```

Controle.h

```
#include <18F252.h>
#device adc=10

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES WDT128                   //Watch Dog Timer uses 1:128 Postscale
#FUSES HS                       //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for
PCD)
#FUSES NOPROTECT                //Code not protected from reading
#FUSES NOOSCSEN                 //Oscillator switching is disabled, main oscillator is
source
#FUSES BROWNOUT                 //Reset when brownout detected
#FUSES BORV27                   //Brownout reset at 2.7V
#FUSES PUT                      //Power Up Timer
#FUSES STVREN                   //Stack full/underflow will cause reset
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18)
used for I/O
#FUSES NOWRT                    //Program memory not write protected
#FUSES NOWRTD                   //Data EEPROM not write protected
#FUSES NOWRTB                   //Boot block not write protected
#FUSES NOWRTC                   //configuration not registers write protected
#FUSES NOCPD                    //No EE protection
#FUSES NOCPB                    //No Boot Block code protection
#FUSES NOEBTR                   //Memory not protected from table reads
#FUSES NOEBTRB                  //Boot block not protected from table reads

#use delay(clock=20000000,RESTART_WDT)
#use
rs232(baud=19200,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8,restart_wdt,errors)
```

7.3 FIRMWARE LEITURA ESTERÇAMENTO

Estercamento.c

```
#include "estercamento.h"
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#define ADDR   'E'

unsigned int16 rise, fall, pulse_width, value;
float time, E, D;

#int_CCP1
void  CCP1_isr(void)
{
  rise = CCP_1; //CCP_1 is the time the pulse went high
}

#int_CCP2
void  CCP2_isr(void)
{
  fall = CCP_2; //CCP_2 is the time the pulse went low
  pulse_width = fall - rise; //pulse width

}



void main()
{

  setup_adc_ports(AN0);
  setup_adc(ADC_CLOCK_INTERNAL);
  setup_spi(SPI_SS_DISABLED);
  setup_wdt(WDT_OFF);
  setup_timer_0(RTCC_INTERNAL);
  setup_timer_1(T1_INTERNAL|T1_DIV_BY_1);
  setup_timer_2(T2_DISABLED,0,1);
  setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
  setup_ccp1(CCP_CAPTURE_RE);
  setup_ccp2(CCP_CAPTURE_FE);
  enable_interrupts(INT_CCP1);
  enable_interrupts(INT_CCP2);
  enable_interrupts(GLOBAL);

  set_adc_channel(0);

  char COMMAND;

  while (true)
  {
    rs232_errors=0;
```

```
        while(getch() != ADDR) restart_wdt();
        COMMAND=getch();
        switch (COMMAND)  // verifica a tecla
        {
          case 'd' :
                    time = (pulse_width-1)*0.2;
                    printf("pulse_width = %lu\r\n",pulse_width);
                    printf("time = %f\r\n",time);
                    printf("Roda D = %f, Roda E = %f\r\n",D=(0.0071*pulse_width)+34.404,E=(0.0074*pulse_width)+28.501);
          break;

          case 'f' :     // Envia contagem do encoder e velocidade.

printf("$%cF,%f,%f\n",ADDR,D=(0.0071*pulse_width)+34.404,E=(0.0074*pulse_width)+28.501);
          break;

          case 't' :
                    value = read_adc();
                    delay_ms(5);
                    float tension;
                    tension = ((0.0145*value)+0.0736);
                    printf("$%cT,%f\n",ADDR,tension);
          break;

        }

      }

    }

    /*
    Ajuste curva
    Leitura   roda E   Roda D
    6285   75   79
    7035   80   84
    8386   91   94
    9500   98   101
    10205   104   107

    Roda esquerda
    y=0.0074*x+28.501

    Roda Direita
    y=0.0071*x+34.404
```

```
*/
```

estercamento.h"

```
#include <18F252.h>
#device adc=10

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES WDT128                   //Watch Dog Timer uses 1:128 Postscale
#FUSES HS                       //High speed Osc (> 4mhz for PCM/PCH) (>10mhz for
PCD)
#FUSES NOPROTECT                //Code not protected from reading
#FUSES NOOSCSEN                 //Oscillator switching is disabled, main oscillator is
source
#FUSES NOBROWNOUT               //No brownout reset
#FUSES BORV20                   //Brownout reset at 2.0V
#FUSES PUT                      //Power Up Timer
#FUSES STVREN                   //Stack full/underflow will cause reset
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18)
used for I/O
#FUSES NOWRT                    //Program memory not write protected
#FUSES NOWRTD                   //Data EEPROM not write protected
#FUSES NOWRTB                   //Boot block not write protected
#FUSES NOWRTC                   //configuration not registers write protected
#FUSES NOCPD                    //No EE protection
#FUSES NOCPB                    //No Boot Block code protection
#FUSES NOEBTR                   //Memory not protected from table reads
#FUSES NOEBTRB                  //Boot block not protected from table reads

#use delay(clock=20000000)
#use
rs232(baud=56700,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8,enable=PIN_C5,restar
t_wdt,errors)
```

7.4 CÓDIGO EM MATLAB PARA PROCESSAMENTO DOS DADOS

```
%*********************************************************************
% Autor: Rodrigo Guerato Siqueira
%
% Implementação de um filtro de Kalman
% entrando com dados de GPS no Filtro Kalman
% Plot dos dados de GPS
```

```
% Plot dos dados de Encoder
% Recusntrução de trajetório modelo Bicicleta
%**************************************************************************

clc
clear all
close all

[what, where]=uigetfile('*.txt','File to read:');
A = load (fullfile(where, what),'-ascii');
what
N=length(A);
cte_roda = 0.01523672437;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Aquisição do tempo - Global Time Ticks (us)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
t=A(1:N,1);
t=t*0.000001;
t=t-t(1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Aquisição dos dados do Giro
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Gloc = 9.780318 * (1+0.0053024*(sin(-24))^2-0.0000058*sin(2*-24)^2)-
3.086*10^(-6)*1;
ajuG = Gloc/9.80665;

%Aceleração em x
Ax = A(1:N,2);
%Aceleração em y
Ay = A(1:N,3);
%Aceleração em z
Az = A(1:N,4);

%Taxa Angular em x
Wx = A(1:N,5);
%Taxa Angular em y
Wy = A(1:N,6);
%Taxa Angular em z
Wz = A(1:N,7);

%Matriz de Orientação
M11 = A(1:N,8);
M12 = A(1:N,9);
M13 = A(1:N,10);
M21 = A(1:N,11);
M22 = A(1:N,12);
M23 = A(1:N,13);
M31 = A(1:N,14);
M32 = A(1:N,15);
M33 = A(1:N,16);

%Inertial Ticks
it = A(1:N,17);
it = it - min(it);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%Aquisição dos dados das rodas
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Roda A - Dianteira esquerda
ContA = A(1:N,18);
ContA = ContA.*cte_roda;
VelA = A(1:N,19);
%Roda B - Traseira esquerda
ContB = A(1:N,20);
ContB = ContB.*cte_roda;
VelB = A(1:N,21);
%Roda C - Traseira direita
ContC = A(1:N,22);
ContC = ContC.*cte_roda;
VelC = A(1:N,23);
%Roda D - Dianteira direita
ContD = A(1:N,24);
ContD = ContD.*cte_roda;
VelD = A(1:N,25);
%Angulo Roda esquerda
AngE = A(1:N,26);
%Angulo Roda direita
AngD = A(1:N,27);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Aquisição dos dados do GPS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Latitude
lattitude = A(1:N,28);
lattitude = lattitude*(-1);
%Longitude
longitude = A(1:N,29);
longitude = longitude*(-1);
%Velocidade sobre a terra (knots)
spogr = A(1:N,30);
%Velocidade sobre a terra (m/s)
velgps = spogr*0.51444444;
%Couse over Groung
Cog = A(1:N,31);
%Idade da correção Diferencial
difage = A(1:N,32);
clear A;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Fim da aquisição dos dados
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Plot dos dados de Encoder
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:N
        if abs(VelA(i)) > 50    %retira Velocidade maluca que aparece
quando
            VelA(i) = 0;        %o encoder passa de 0 para -1;
        end
        if abs(VelB(i)) > 50    %retira Velocidade maluca que aparece
quando
            VelB(i) = 0;        %o encoder passa de 0 para -1;
        end
```

```
        if abs(VelC(i)) > 50    %retira Velocidade maluca que aparece
quando
            VelC(i) = 0;        %o encoder passa de 0 para -1;
        end
        if abs(VelD(i)) > 50    %retira Velocidade maluca que aparece
quando
            VelD(i) = 0;        %o encoder passa de 0 para -1;
        end
end

figure('Name','Dados dos Encoders','NumberTitle','off');
title('Dados de Encoder')
subplot(2,1,1)
plot(t,ContA,t,ContB,t,ContC,t,ContD)
title('Deslocamentos')
legend('Diant Esq (A)','Tras Esq (B)','Tras Dir (C)','Diant Dir (D)')
xlabel('Tempo (s)')
ylabel('Distância (m)')
subplot(2,1,2)
plot(t,VelA,t,VelB,t,VelC,t,VelD)
title('Velocidades')
legend('Diant Esq (A)','Tras Esq (B)','Tras Dir (C)','Diant Dir (D)')
xlabel('Tempo (s)')
ylabel('Velocidade (m/s^2)')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Plot dos dados de GPS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:N

    lat(i) = floor(lattitude(i)/100)+(rem(lattitude(i),100))/60;
    lon(i) = floor(longitude(i)/100)+(rem(longitude(i),100))/60;

    [xGPS(i),yGPS(i)] = ConvertGedtoUTM(lat(i),lon(i));

end




minX=min(xGPS);
minY=min(yGPS);

xGPS=xGPS-minX;
yGPS=yGPS-minY;

figure('name','Pontos GPS','NumberTitle','off')
plot(xGPS,yGPS,'.')
title('Dados GPS')
grid on
xlabel('Deslocamento em (m)')
ylabel('Deslocamento em (m)')
axis equal

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Multiplica aceleração pela Matriz de Orientação
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
rot=0.85*(pi/2);
%rot=0;

mat_rot = [cos(rot) -sin(rot) 0
            sin(rot) cos(rot) 0
             0 0 1];
for i=1:N
        Acc = [Ax(i) Ay(i) Az(i)];

        MatOrient = inv([M11(i) M12(i) M13(i)
                     M21(i) M22(i) M23(i)
                     M31(i) M32(i) M33(i)]);

        VetAcc(i,:) = Acc * MatOrient * mat_rot;
        VetVel(i,:) = [((VelB(i)+VelC(i))/2) 0 0] * MatOrient*mat_rot;

        theta(i) = asin(-M13(i));  %Pich
        fi(i) = atan2(M23(i),M33(i)); %Roll
        psi(i) = atan2(M12(i),M11(i)); %Yaw

end

VetAcc = [VetAcc(:,1) VetAcc(:,2)];
VetVel = [VetVel(:,1) VetVel(:,2)];

VetAcc = VetAcc * ajuG; %Ajusta Acelerações para o Brasil e
VetAcc = VetAcc * Gloc; %transforma elas em m/s2

figure('Name','Angulos de Euler','NumberTitle','off')
plot(t,fi,t,theta,t,psi)
legend('Roll','Pich','Yaw')
title('Angulos de Euler')
xlabel('Tempo (s)')
ylabel('Angulos (Rad)')

z=1:N;
figure('Name','Dados do Inercial','NumberTitle','off');
subplot(2,1,1)
plot(z,VetAcc(:,1))
title('Acelerações em x')
xlabel('Tempo (s)')
ylabel('Aceleração (m/s2)')

subplot(2,1,2)
plot(t,VetAcc(:,2))
title('Acelerações em y')
xlabel('Tempo (s)')
ylabel('Aceleração (m/s2)')


%*********************************************************************
% Adequa os nomes dos dados para fazer o Kalman
%*********************************************************************
clear A Acc Ax Ay Az Wx Wy Wz B M11 M12 M13 M21 M22 M23 M31 M32 M33 ...
    MatOrient ajuG Gloc i;
```

```
%*************************************************************************
% dados Iniciais
%*************************************************************************
%Gera vetores de posição, velocidade e aceleraçao inicial
%*************************************************************************

Po = [mean(xGPS(1500:1700)) mean(yGPS(1500:1700))];

% inicialização do vetor de estados

Xest = [Po VetVel(1,:) VetAcc(1,:)]';

%inicialização da matriz de covariancia do erro

P = eye(6)*1e0;

%inicialização do vetor de entradas

U = [0 0]';

%declaração do ruido de medição da velocidade Rv

Rv=eye(2)*1e0;

%declara desvio padrão do ruido de medição da posição
%(erro de posiçao ep = 0,3m)

Rp=eye(2)*1e0;

%matrizes que representam a medição
%H_Coord: Quando se tem medidas de referencia das coordenadas
% entrada dos dados de GPS se houverem
H_Coord = [1 0 0 0 0 0
           0 1 0 0 0 0];

H_Vel = [0 0 1 0 0 0
         0 0 0 1 0 0 ];

xGPS_old = xGPS(1);
kke=0;
kkg=0;
%=========================================================================
%=========================================================================
% Inicio do Loop - Filtro de Kalman
%=========================================================================
%=========================================================================

%calculo do intervalo de amostragem

for (i=1500:N)
    if(i==1)
        T = it(i+1)-it(i);
    else
        T = it(i)-it(i-1);
    end

%matrizes da dinamica do sistema ==> x(k+1) = A.x(k)+B.u(k)+C.w(k)
```

```
A = [1 0 T 0 0.5*T^2 0
     0 1 0 T 0 0.5*T^2
     0 0 1 0 T 0
     0 0 0 1 0 T
     0 0 0 0 1 0
     0 0 0 0 0 1];

 B = [0.5*T^2 0
      0 0.5*T^2
      T 0
      0 T
      0 0
      0 0];

 % Matriz do ruído associado ao processo (erro de posiçao proporcional ao
 % cubo do tempo, erro de velocidade proporcional so quadrado do tempo e
 % erro de aceleração proporcional ao tempo.

 C = [T^3/6 0 0 0 0 0
      0 T^3/6 0 0 0 0
      0 0 T^2/2 0 0 0
      0 0 0 T^2/2 0 0
      0 0 0 0 T 0
      0 0 0 0 0 T];

  % declara ruido de processo Q


%    Q = 1*C*C';
   Q=[1 0 0 0 0 0
       0 1 0 0 0 0
       0 0 1 0 0 0
       0 0 0 1 0 0
       0 0 0 0 1 0
       0 0 0 0 0 1];

  % entradas do sistema u(k) => (leituras dos acelerometros transformadas
  % para NED)

  U = VetAcc(i,:)';

  %*************************************************************************
  %verifica se utiliza medidas de coordenadas ou medidas de velocidade
  %*************************************************************************

   if xGPS_old == xGPS(i)
      H= H_Vel;
      Z = VetVel(i,:)';
      R = Rv;
      kke = kke+1;

   else
       H = H_Coord;
       Z = [xGPS(i) yGPS(i)]';
       R = Rp;
       kkg = kkg+1;
   end
```

```
   xGPS_old = xGPS(i);

  %*************************************************************************
  % Atualização temporal (Predição)
  %*************************************************************************
  % projeta o estado a frente

  Xest = A*Xest + B*U;

  % projeta a covariancia do erro a frente

  P = A*P*A'+Q;

  %*************************************************************************
  % Atualização da medição (Correção)
  %*************************************************************************
  %computa o ganho K do filtro de Kalman

  K = P*H'*inv(H*P*H'+R);

  % atualiza a estimativa Xk com a medição Zk

  Xest = Xest + K*(Z-H*Xest);


  % atualiza a covariancia do erro

  P = P-K*H*P;

  Var_P = [P(1,1) P(2,2)];
  STD_(i) = norm(Var_P);

  % simetriza a matriz P

  P = 0.5*(P+P');


  %armazena estados estimados (gera uma matriz onde cada coluna da matriz
  %representa uma estimativa dos estados no instante (i))

  M_Xest(:,i) = Xest;


end

figure('Name','Filtro de Kalman','NumberTitle','off')
plot(M_Xest(1,:),M_Xest(2,:))
grid on
title('Trajetória FK')
xlabel('Deslocamento em (m)')
ylabel('Deslocamento em (m)')
axis equal

[Vx, Vy, Posx, Posy] = integra_xy(VetAcc(:,1),VetAcc(:,2),it,N);

figure('Name','Duplo Integrador','NumberTitle','off')
```

```
plot (Posx,Posy,'b.')
grid on
title('Trajetória DI')
xlabel('Deslocamento em (m)')
ylabel('Deslocamento em (m)')
axis equal


figure ('Name','GPS + Kalman','NumberTitle','off')
% plot(M_Xest(1,(100:N)),M_Xest(2,(100:N)),'r.',xGPS,yGPS,'b.')
plot(M_Xest(1,:),M_Xest(2,:),'r.',xGPS,yGPS,'b.')

legend('Kalman','GPS')
xlabel('Deslocamento em (m)')
ylabel('Deslocamento em (m)')
axis equal
grid on


function [Vx, Vy, Posx, Posy] = integra_xy(Axms,Ayms,it,N)

Vx=[];
Vy=[];
Vx(1)=0;
Vy(1)=0;

for i=2:N
    dt=(it(i)-it(i-1));
    Vx(i)=Vx(i-1)+(((Axms(i)+Axms(i-1))/2)*dt);
    Vy(i)=Vy(i-1)+(((Ayms(i)+Ayms(i-1))/2)*dt);
end

Posx=[];
Posy=[];
Posx(1)=0;
Posy(1)=0;

for i=2:N
    dt=(it(i)-it(i-1));
    Posx(i)=Posx(i-1)+Vx(i)*dt+(1/2)*Axms(i)*(dt)^2;
    Posy(i)=Posy(i-1)+Vy(i)*dt+(1/2)*Ayms(i)*(dt)^2;
end
```

7.5 CÓDIGO EM MATLAB PARA CONVERSÃO DE COORDENADAS GEODÉSICAS PARA UTM.

```
function [x,y] = ConvertGedtoUTM(lat,lon)
%   Rodrigo Guerato Siqueira
%   function para converter coordenadas Geodesicas para UTM.
%   22/09/2011
%   Adaptado do Java script para MATLAB
```

```
%   Original em:
http://home.hiwaay.net/~taylorc/toolbox/geography/geoutm.html
%   Reference: Hoffmann-Wellenhof, B., Lichtenegger, H., and Collins, J.,
%   GPS: Theory and Practice, 3rd ed.  New York: Springer-Verlag Wien,
1994.

%Ellipsoid model constants (actual values here are for WGS84) */
 sm_a = 6378137.0;      %Ellipsoid model major axis.
 sm_b = 6356752.314;    %Ellipsoid model minor axis.
 sm_EccSquared = 6.69437999013e-03;

 UTMScaleFactor = 0.9996;

% Compute the UTM zone.

zone = floor((lon + 180.0) / 6) + 1;

phi=(lat/ 180.0 * pi);      %Latitude of the point, in radians.
lambda=(lon/ 180.0 * pi);   %Longitude of the point, in radians.

%Determines the central meridian for the given UTM zone.
%Longitude of the central meridian to be used, in radians.
lambda0 = ((-183.0 + (zone * 6.0))/ 180.0 * pi);

%Precalculate ep2
ep2 = ((sm_a^2) - (sm_b^2)) / (sm_b^2);

%Precalculate nu2
nu2 = ep2 * (cos(phi)^2);

%Precalculate N
N = (sm_a^2) / (sm_b * sqrt(1 + nu2));

% Precalculate t
t = tan(phi);
t2 = t * t;
tmp = (t2 * t2 * t2) - (t^6.0);

%Precalculate l
l = lambda - lambda0;

%Precalculate coefficients for l**n in the equations below
%so a normal human being can read the expressions for easting
%and northing
%-- l**1 and l**2 have coefficients of 1.0 */
l3coef = 1.0 - t2 + nu2;

l4coef = 5.0 - t2 + 9 * nu2 + 4.0 * (nu2 * nu2);

l5coef = 5.0 - 18.0 * t2 + (t2 * t2) + 14.0 * nu2 - 58.0 * t2 * nu2;

l6coef = 61.0 - 58.0 * t2 + (t2 * t2) + 270.0 * nu2 - 330.0 * t2 * nu2;

l7coef = 61.0 - 479.0 * t2 + 179.0 * (t2 * t2) - (t2 * t2 * t2);

l8coef = 1385.0 - 3111.0 * t2 + 543.0 * (t2 * t2) - (t2 * t2 * t2);
```

```
% Calculate easting (x)
xy(1) = N * cos(phi) * l + (N / 6.0 * (cos (phi)^3.0) * l3coef ...
        * (l^3.0)) + (N / 120.0 * (cos (phi)^5.0) * l5coef * (l^5.0)) ...
        + (N / 5040.0 * (cos (phi)^7.0) * l7coef * (l^7.0));

%Computes the ellipsoidal distance from the equator
%to a point at a given latitude.

%Precalculate n
n = (sm_a - sm_b) / (sm_a + sm_b);

%Precalculate alpha
alpha = ((sm_a + sm_b) / 2.0) * (1.0 + ((n^2.0) / 4.0) + ((n^4.0) / 64.0));

%Precalculate beta
beta = (-3.0 * n / 2.0) + (9.0 * (n^3.0) / 16.0) + (-3.0 * (n^5.0) / 32.0);

%Precalculate gamma
gamma = (15.0 * (n^2.0) / 16.0) + (-15.0 * (n^4.0) / 32.0);

%Precalculate delta
delta = (-35.0 * (n^3.0) / 48.0) + (105.0 * (n^5.0) / 256.0);

%Precalculate epsilon */
epsilon = (315.0 * (n^4.0) / 512.0);

%Now calculate the sum of the series (ALOM = Arc Length Of Meridian)
ALOM = alpha * (phi + (beta * sin(2.0 * phi)) + ...
       (gamma * sin (4.0 * phi)) + (delta * sin (6.0 * phi)) ...
       + (epsilon * sin (8.0 * phi)));

%Calculate northing (y)
xy(2) = ALOM + (t / 2.0 * N * (cos (phi)^2.0) * (l^2.0)) ...
        + (t / 24.0 * N * (cos (phi)^4.0) * l4coef * (l^4.0)) ...
        + (t / 720.0 * N * (cos (phi)^6.0) * l6coef * (l^6.0)) ...
        + (t / 40320.0 * N * (cos (phi)^8.0) * l8coef * (l^8.0));

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Adjust easting and northing for UTM system.
xy(1) = xy(1) * UTMScaleFactor + 500000.0;
xy(2) = xy(2) * UTMScaleFactor;
    if xy(2) < 0
        xy(2) = xy(2) + 10000000;
    end

x = xy(1);
y = xy(2);
```