Pontifícia Universidade Católica do Rio Grande do Sul

Faculdade de Engenharia – Faculdade de Informática

Concepção do *On-Board Data Handling* com Sensor Inercial para Aplicações Espaciais

Trabalho de Conclusão Curso

Autores
Cristiano Garré Ferreira
Felipe Augusto da Silva
Paulo Ricardo Cechelero Villa

Orientador
Prof. Dr. Eduardo Augusto Bezerra

Porto Alegre, junho de 2009

Resumo

A indústria espacial mundial movimenta bilhões de doláres anualmente. **Segundo a publicação "The Space Report 2009", dos US$257 bilhões gastos no ano passado, dois terços desse valor foram movimentados pela iniciativa privada**. Treze governos, incluindo o Brasil, realizaram juntos em 2008 despesas em torno de US$83 bilhões, sendo que apenas os USA são responsáveis por 80% desses gastos. **Essa tendência da iniciativa privada participar cada vez mais de atividades comerciais na área espacial já é uma realidade no Brasil**. O Brasil é um dos poucos países no mundo que possui um programa espacial atuante. Nos últimos anos a Agência Espacial Brasileira lançou diversos satélites, porém em quantidade ainda bastante inferior ao se considerar programas espaciais tais como a Nasa, Agência Espacial Europeia (ESA) e a China. Por outro lado, ao se comparar com o resto do mundo, os números brasileiros são bastante significativos, e a tendência para os próximos anos é um aumento considerável nas atividades nesse setor, com a participação cada vez mais intensa da iniciativa privada. O Instituto Nacional de Pesquisas Espaciais (INPE) está a cerca de dez anos preparando a "Missão Espacial Completa", e o primeiro satélite desenvolvido inteiramente no Brasil, o Amazônia I, será lançado nos próximos três anos.

O projeto e desenvolvimento desse satélite causou uma revolução na indústria nacional, uma vez que o INPE sozinho não possui recursos humanos, intelectuais e capacidade produtiva para essa iniciativa. **A indústria brasileira precisou se adequar as exigências de inovação, alta tecnologia, controle de qualidade e confiabilidade características do setor espacial**. A maioria dos fornecedores do INPE são empresas localizadas em São Paulo, mais exatamente, na região de São José dos Campos. A cerca de cinco anos a PUCRS iniciou pesquisas nessa área junto ao INPE, o que culminou com a criação da empresa Innalogics Ltda na Incubadora Raiar do TecnoPUC. **Em 2008, a Innalogics formou um consórcio com a empresa Aeroeletrônica, também de Porto Alegre, e esse consórcio foi o vencedor de uma licitação do INPE para fornecimento de um dos principais módulos do satélite Amazônia I, o módulo de comunicação com a Terra**. Esse módulo está em fase final de produção e estará sendo entregue ao INPE nos meses de julho e agosto de 2009. No segundo semestre de 2009, o INPE estará lançando o edital de uma nova licitação, dessa vez para o desenvolvimento do On-Board Data Handling (OBDH), com computador de bordo (On-Board Computer, OBC), da plataforma multi-missão a ser utilizada no Amazônia I.

O presente trabalho de conclusão de curso se antecipa a essa licitação, desenvolvendo a arquitetura básica do OBDH proposto pelo INPE, resultando em uma importante vantagem competitiva para as empresas gaúchas envolvidas. O projeto e desenvolvimento do OBDH foi realizado seguindo sugestões, instruções e acompanhamento do INPE, contando com o apoio da Innalogics e Aeroeletrônica. Esse relatório de trabalho de conclusão de curso descreve detalhes técnicos e apresenta os resultados alcançados.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Siglas e Abreviações

ACDH	Attitude Control and Data Handling
AD	Analog to Digital
AEB	Agência Espacial Brasileira
AMBA	Advanced Microcontroller Bus Architecture
ASIC	Application-Specific Integrated Circuit
BCH	Bose-Chaudhuri, Hocquenghem
CAN	Controller Area Network
CCSDS	Consultative Committee for Space Data Systems
CLCW	Command Link Control Word
CLTU	Command Link Transmission Unit
CNPQ	Conselho Nacional de Desenvolvimento Científico e Tecnológico
COTS	Commercial Off-The-Self
CPDU	Command Pulse Distribution Unit
CRC	Cyclic Redundancy Check
DA	Digital to Analog
DPU	Data Processing Unit
DSU	Debug Support Unit
E2PROM	Electrically Erasable Programmable Read-Only Memory
ESA	European Space Agency
ESE	Electrical Support Equipment
EDAC	Error Detection and Correction
FACIN	Faculdade de Informática
FARM	Frame Acceptance and Reporting Mechanism
FPGA	Field-Programmable Gate Array
GAPH	Grupo de Apoio ao Projeto de Hardware
GNAT	Modified General Public License
GPL	General Public License
GPS	Global Positioning System
GRLIB	Gaisler Research Library
GRMON	Gaisler Research Monitor
GSE	Grupo de Sistemas Embarcados
HDLC	High Level Data Link Control
IDE	Integrated Drive Electronics
INPE	Instituto Nacional de Pesquisas Espaciais
INS	Inertial Navigation System
LED	Light Emitting Diode
LEP	Laboratório de Ensino e Pesquisa
LGA	Land Grid Array
MAC	Medium Access Control
MEMS	MicroElectroMechanical Systems
MISO	Master Input, Slave Output
MMC	Multi Media Card
MOSI	Master Output, Slave Input

MRO	Mars Reconnaissance Orbiter
NASA	National Aeronautics and Space Administration
OBC	On-Board Computer
OBDH	On-Board Data Handling
PC	Personal Computer
PMM	Plataforma Multi-Missão
PNAE	Programa Nacional Espacial Brasileiro
PUCRS	Pontifícia Universidade Católica do Rio Grande do Sul
RCC	RTEMS Cross Compiler
RS	Reed-Solomon
RTEMS	Real Time Executive for Multiprocessor Systems
RSoC	Reconfigurable System-on-a-Chip
SCL	Serial Clock
SD	Security Digital
SDRAM	Single Data Rate Random Access Memory
SISCAO	Sistema de Controle de Atitude e Órbita de um satélite em três eixos
SoC	System-on-a-Chip
SPI	Serial Peripheral Interface
SO	Sistema Operacional
SS	Slave Select
SSTL	Surrey Sattelite Technology Ltd.
TC	Telecomando
TCC	Trabalho de Conclusão de Curso
TCD	Telecomando Direto
TCR	Telecomando Roteado
TM	Telemetria
TTL	Transistor-Transistor Logic
UART	Universal Asynchronous Receiver Transmitter
UTMC	Unidade de Telemetria e Telecomando
VHDL	Very High Speed Integrated Circuit Hardware Description Language

1. Introdução

O desenvolvimento da tecnologia espacial possibilita um melhor entendimento não somente do nosso planeta, mas também de soluções a serem utilizadas em problemas do cotidiano. Desde o lançamento do primeiro objeto ao espaço seguiram-se pesquisas visando a utilização de satélites em missões de exploração, aprimoramento dos meios de comunicação, localização, observação, defesa entre outros. Atualmente satélites são lançados com uma grande diversidade de objetivos [1], como por exemplo:

- Satélites de Astronomia: para observação do espaço, galáxias, planetas distantes e outros objetos do espaço;
- Satélites Científicos: desenvolvidos para realizar experimentos científicos no ambiente espacial. Exemplos desses satélites incluem aqueles para estudo do Sol (radiação, vento solar), experimentos em gravidade zero entre outros;
- Satélites de Comunicação: satélites estacionários para fins de telecomunicação;
- Satélites de Navegação: enviam sinais de tempo e localização para dispositivos móveis na Terra, para determinar a localização dos mesmos;
- Satélites de Observação da Terra: utilizados para monitoração de meio ambiente, meteorologia, criação de mapas entre outros;
- Satélites de uso militar.

Atualmente alguns satélites com tecnologia Brasileira estão orbitando a Terra, existindo também diversos projetos em andamento. Porém, para que a participação do Brasil se torne mais expressiva e economicamente viável o Instituto Nacional de Pesquisas Espaciais (INPE) está adotando um padrão com o objetivo de facilitar o reaproveitamento de módulos de satélites entre diversas missões. Para isso o INPE vem solicitando apoio de terceiros (empresas e instituições de pesquisa) com o intuito de desenvolver uma estrutura modular nomeada Plataforma Multi-Missão (PMM) [2].

A PMM é uma plataforma genérica para satélites na classe de 500 kg de peso total. A principal motivação do INPE para o desenvolvimento da PMM é fornecer um serviço modular compatível com um conjunto de *payloads* (cargas úteis do satélite) pré-estabelecidos pelo Programa Nacional Espacial Brasileiro (PNAE), enfatizando aplicações de observação na região Amazônica. Essa motivação surgiu visando a redução de custos e escalabilidade, já que com a PMM, o conceito de reuso seria praticado quase por completo, necessitando apenas realizar alterações que fossem específicas de cada missão. O projeto PMM tem como meta o lançamento de quatro satélites, o satélite de observação Amazônia-1, o satélite científico Lattes-1, o satélite de observação da Terra MAPSAR e o satélite meteorológico de medidas de precipitação GPM-Br[2].

O satélite Amazônia-1 tem data de lançamento prevista para 2011, será o primeiro satélite de observação da Terra desenvolvido inteiramente pelo Brasil e o primeiro a utilizar a PMM. O Amazônia-1 permitirá ao Brasil, juntamente com os satélites CBERS-3 (lançamento previsto para 2009) e CBERS-4 (lançamento previsto para 2011) desenvolvidos em parceria com a China, uma cobertura completa da terra em cinco dias, tornando o país autônomo para obtenção de imagens de

média resolução e eliminando o risco de descontinuidade de fornecimento desse tipo de imagem por parte de satélites estrangeiros.

Com o lançamento agendado de satélites nacionais e o desenvolvimento da PMM, a área espacial no Brasil torna-se cada vez mais promissora. Projetos na área espacial são de reconhecida importância para os avanços da tecnologia como um todo, sendo esta uma das motivações para o desenvolvimento desse trabalho.

O presente trabalho de conclusão de curso abrange as áreas: espacial, microeletrônica, sistemas de tempo real e arquitetura de computadores. Possuindo como objetivo central o projeto e a implementação de um modelo básico para gerência e manipulação de dados a bordo de um satélite (*On-Board Data Handling* - OBDH).

Além de uma implementação básica do OBDH, o trabalho contempla a integração deste com os demais subsistemas que compõem a Plataforma de Serviços de um satélite. Um satélite é dividido em três grandes partes[51], uma delas é a Plataforma de Serviço, responsável por realizar a manipulação dos dados provenientes dos sensores e o controle de atitude e órbita do satélite. As outras partes são o *payload*, que possui equipamentos específicos da missão e o sistema de propulsão, responsável pela navegação.

A integração foi desenvolvida para exemplificar o fluxo completo das mensagens transmitidas entre a Estação Terrestre e o veículo espacial. As mensagens transmitidas da Estação Terrestre são recebidas, verificadas (integridade dos dados) e desempacotadas pelo subsistema de comunicação. Esse repassa as informações para o OBDH, que realiza a interpretação e manipulação das mensagens. Quando o conteúdo da mensagem representa um comando de requisição de dados, o OBDH executa as rotinas de busca nos sensores para capturar as medições aferidas naquele instante. Ao capturar e manipular os dados, o OBDH envia para o Subsistema de Comunicação que se encarrega de transmitir a mensagem do satélite para a Estação Terrestre.

Os dados providos dos sensores têm como finalidade localizar o satélite espacialmente, seja com o intuito de informar a Estação Terrestre ou em alguns casos para tomar atitudes para evitar que um evento indesejado ocorra. Sensores de estrela, sensores solar, sensores inerciais, entre outros fazem parte dos sistemas de controle de atitude e órbita da maioria dos satélites. Para aproximar esse trabalho de uma aplicação real serão utilizados acelerômetros e giroscópios (sensores inerciais) comunicando-se com o OBDH, as informações providas pelos sensores serão enviadas para a Estação Terrestre quando solicitado pela mesma.

Adicionalmente, é importante mencionar que devido ao espaço de tempo disponível para implementação e teste do trabalho proposto, nesse projeto não estão sendo implementados os requisitos de confiabilidade, tolerância a falhas e radiação que um projeto espacial exige. Esse tratamento poderá ser implementado em trabalhos futuros com o intuito de tornar o trabalho mais próximo ao idealizado pelo INPE.

O texto do presente relatório está organizado como segue. O Capítulo 2 descreve alguns trabalhos relacionados com o tema abordado. O Capítulo 3 realiza uma descrição do Projeto PUC#SAT onde o presente trabalho se insere. O Capítulo 4 descreve a Plataforma de

Desenvolvimento, ou seja, a arquitetura e configuração de hardware utilizada para desenvolver o OBDH, detalhando também o funcionamento da Unidade de Telemetria e Telecomando (UTMC). O Capítulo 5 descreve como foi implementada a Plataforma de Serviços e como foram estruturadas as funções de aquisição e manipulação de dados do OBDH. O Capítulo 6 relata os detalhes da implementação, tanto da Plataforma de Desenvolvimento, quanto da Plataforma de Serviços. Já o Capítulo 7 realiza a integração das duas Plataformas e demonstra os testes e resultados obtidos com o funcionamento do OBDH. Finalizando, o Capítulo 8 apresenta as conclusões e os trabalhos futuros. O trabalho possui ainda alguns anexos, que auxiliam na compreensão do que foi abordado no trabalho.

2. Trabalhos Relacionados e Contextualização do Desenvolvimento

Este capítulo apresenta alguns trabalhos na área espacial relacionados ao projeto proposto. Os trabalhos descritos no presente capítulo foram desenvolvidos em universidades e centros de pesquisa em diversos países e foram utilizados para realizar o estudo e adquirir o embasamento necessário para o desenvolvimento desse trabalho.

2.1 Centro Espacial Surrey - Reino Unido

O Centro Espacial de Surrey [30] desenvolve satélites de pequeno porte utilizando o padrão de pilha de protocolos do *Consultative Committee for Space Data Systems* (CCSDS)[5][6][7] nos seus módulos de comunicação. Por exemplo, no projeto ChipSat [31] um computador de bordo de um satélite é implementado na forma de um System-on-Chip (SoC). Núcleos de propriedade intelectual escritos na linguagem de descrição de hardware VHDL são usados para construir sistemas do computador de bordo. Os principais blocos do SoC consistem em: microprocessador (LEON SPARC), unidade de detecção e correção de erros (EDAC) em memória, sistemas de inicialização (*boot*), controlador HDLC (*High Level Data Link Control*), interface de rede CAN (*Controller Area Network*), interface IDE (*Integrated Drive Electronics*), co-processador aritmético e interface de barramento de periféricos AMBA. Um FPGA Xilinx Virtex é usado para prototipação do SoC. Na Figura 1 é apresentado o diagrama de blocos do ChipSat e suas ligações nos pinos do FPGA.

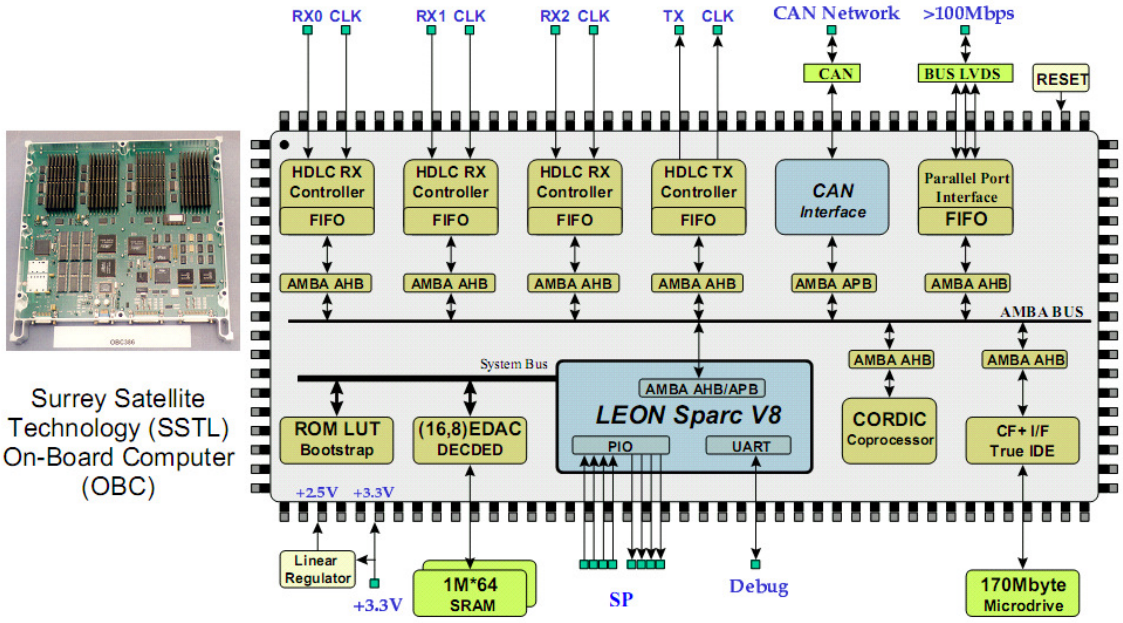

Figura 1 - Diagrama de blocos do ChipSat.

O projeto visa implementar um sistema OBDH para um satélite de pequeno porte (cerca de 100 a 200 kg). O OBDH conta com sensoriamento remoto e capacidade de coleta de dados tudo integrado em um único FPGA. A especificação do SoC é baseada em requisitos de futuras missões de satélites da empresa *Surrey Satellite Technology Ltd*. (SSTL), que desenvolve e manufatura satélites de pequeno porte a mais de 20 anos. O SoC abordado consiste em 4 subsistemas: um processador RISC modificado para utilização no espaço; um subsistema de

manipulação de imagens capaz de capturar e comprimir imagens; um subsistema para comunicação bi-direcional do satélite; e suporte a subsistemas periféricos. O estudo mostrou que é possível implementar as funcionalidades de um OBDH para pequenos satélites em FPGAs, o trabalho utilizou o FPGA da Xilinx XCV800 na placa de prototipação XSV-800 da empresa XESS *Corporation*.

2.2 OBDH da Empresa LABEN

O trabalho de Aranci, Maltecca e Ranieri [41] descreve o OBDH desenvolvido na empresa Italiana LABEN para a Agência Espacial Européia (*European Space Agency* - ESA). O artigo descreve a experiência obtida pela equipe no projeto e desenvolvimento do OBDH, projetado para utilização em dois satélites da ESA. Uma característica importante do projeto considera a utilização do mesmo sistema em duas missões com requisitos distintos, levando a necessidade de desenvolvimento de um OBDH re-utilizável.

O artigo inicia descrevendo as características das missões, incluindo a plataforma de serviço e o *payload*. A Figura 2 apresenta o diagrama de blocos do OBDH proposto, onde o módulo CDMU realiza a função do OBC, gerenciando toda a transferência de dados, servindo como o computador servidor da rede do OBDH. Os instrumentos do *payload* e todos os módulos da plataforma de serviços estão conectados nessa rede. No projeto da CDMU foi utilizado o processador 31750, no qual posteriormente foi identificado um erro em algumas operações numéricas. Por essa razão, nas versões seguintes desse projeto foi utilizado o processador ERC32[42], o mesmo em uso no INPE e considerado para o presente trabalho, conforme discutido no Capítulo 4.

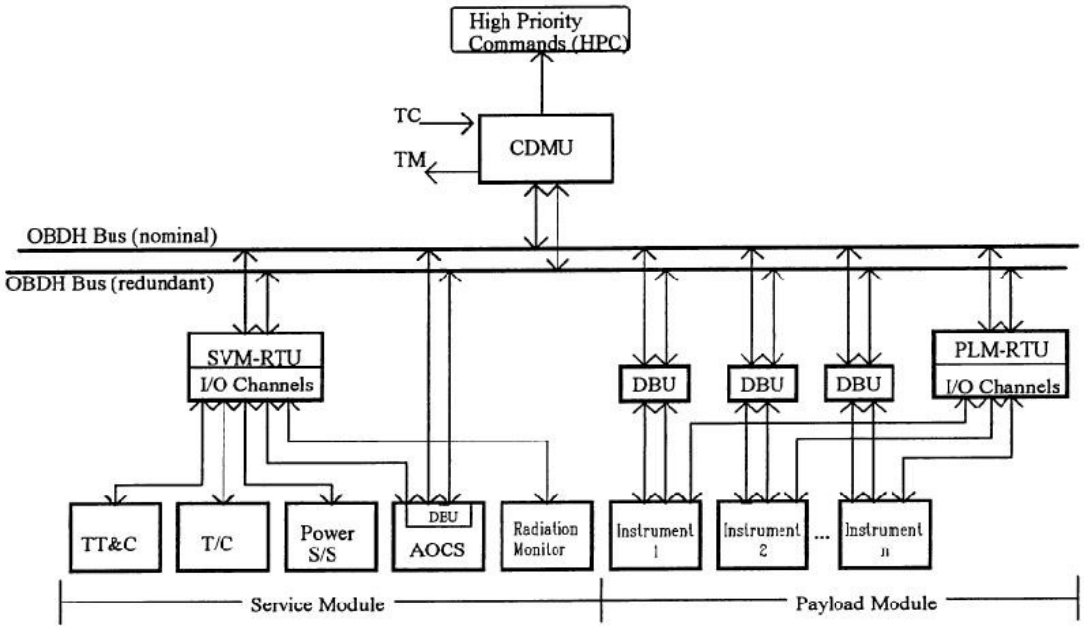

Figura 2 - Diagrama de Blocos do OBDH da Empresa LABEN.

2.3 Universidade Técnica IDA - Alemanha

Pesquisadores da Universidade Técnica IDA em Braunschweig [37], realizaram um estudo comparativo entre modelos de unidades de processamento de dados (DPU) utilizados em aplicações espaciais. As DPUs são usadas como interface entre o satélite e os instrumentos sensoriais, realizando o processamento dos dados em casos específicos que necessitem de um robusto pré-processamento antes de serem enviados para a Estação Terrestre. O estudo de caso do trabalho foram os instrumentos de aquisição de imagens em tempo real, aplicação que exige um alto poder computacional. Isso porque as imagens após serem capturadas necessitam ser manipuladas utilizando algoritmos de compressão. Uma DPU além de ser robusta necessita atender outros requisitos como: mecanismos de tolerância a radiação, baixo consumo de potência e volume/massa reduzidos.

Os autores do trabalho citam as três principais abordagens de implementações para sistemas de processamento de dados:

- Modelo tradicional, baseado em hardware físico e componentes discretos, ou *Aplication-Specific Integrated Circuits* (ASICs) de alto padrão de qualidade.
- *Commercial Off-The-Self* (COTS), modelos que utiliza soluções integradas.
- Reconfigurable System-on-Chip (RSoC), solução com processador e funções especiais, implementados em um FPGA com tolerância a radiação.

A Figura 3, ilustra um modelo de DPU em SoC, suas atividades de processamento e como essa se comunica com os demais componentes do satélite.

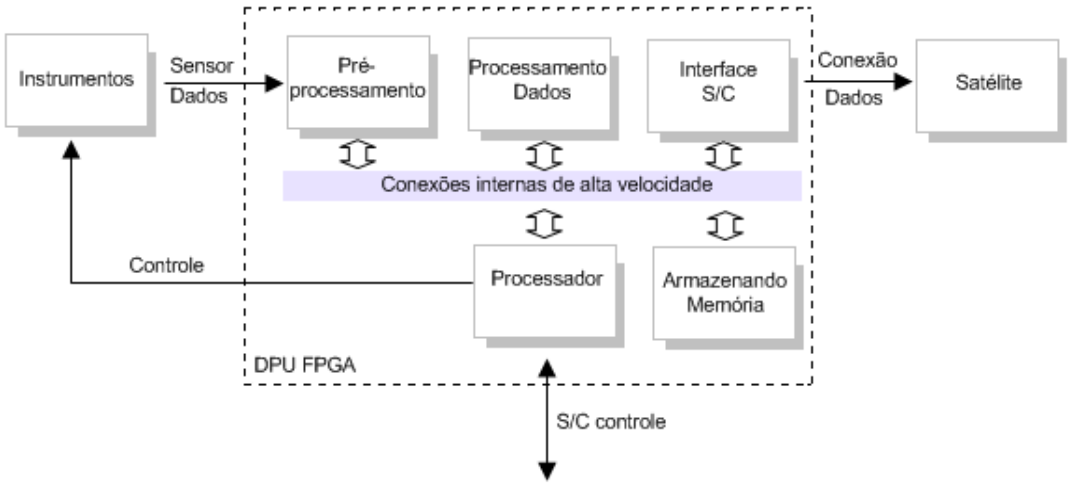

Figura 3 - Arquitetura de uma DPU System-on-Chip.

O objetivo do trabalho foi demonstrar através da análise do desempenho e da granularidade das características dos modelos, as novas tendências de implementações de DPUs. Para quantificar o desempenho e as vantagens dos modelos, foram utilizadas as premissas citadas anteriormente e parâmetros de desenvolvimento de projeto, como: custo, tempo de projeto e

flexibilidade. A Tabela 1 demonstra as vantagens de desenvolver uma DPU em SoC, ao invés, de utilizar os modelos tradicionais de implementação.

Tabela 1 - Parâmetros Importantes de diferentes modelos.

Modelos	**Tradicional**	**COTS**	**RSoC**
DPUs em operação	**Rosetta Rosina**	**µDPU**	**VEX VMC**
Processador	TSC21020	DSP56302	LEON-2 em Virtex-1
Volume – sem encapsulamento [cm^3]	850	70	250
Massa - sem encapsulamento [g]	800	80	280
Consumo de potência [W]	4 – 6	0,4 – 4	3,5
Poder processamento [MIPS]	6 – 20	0 – 100	20
Memória Cache	16 Mbit	6 Mbit	16 Mbit
Memória de Dados	128 Mbit	1 Gbit	1 Gbit
DPUs em desenvolvimento	**AT697 / ASIC**	**MSC8101**	**LEON-3 em Virtex-II**
Consumo de potência [W]	2 – 4	1 – 4	2 – 4
Máx. Poder processamento [MIPS]	86 / ~300	3000	~80
Tolerância permanente (TID, LU)	alta	média	alta
Tolerância transiente (SEU, SEFI)	alta	média	média
Qualidade de confiabilidade	alta	média	alta
Custo dos componentes	alto / muito alto	baixo	médio
Tempo de desenvolvimento	médio / longo	médio	curto
Flexibilidade	muito baixa	baixa	alta

Analisando a Tabela 1 é possível fazer diversas análises sobre os modelos de arquiteturas de desenvolvimento. Uma das análises faz referência aos parâmetros de dimensão dos modelos, observa-se que as novas abordagens tendem a possuir menores parâmetros de volume e massa do que os modelos tradicionais. Essa tendência se reforça a medida que observa-se também o aumento da capacidade de processamento e armazenamento dos modelos. Outra característica relevante dos modelos é o balanço entre o custo e a flexibilidade. Os modelos tradicionais possuem padrões de custo elevado e oferecem uma flexibilidade reduzida, portanto essa abordagem se torna uma solução específica. Já o modelo em RSoC, que possui um custo considerado mediano, possui alta flexibilidade o que a torna uma solução com aspectos genéricos.

2.4 PUCRS - Brasil

O trabalho de mestrado realizado por Gabriel Marchesan Almeida [13], no âmbito do projeto PUC#SAT e orientado pelo professor Eduardo Bezerra, também é destinado à pesquisa espacial no contexto de tratamento de Telecomando (TC) e Telemetria (TM) para satélites. O principal objetivo do trabalho foi investigar técnicas de desenvolvimento de hardware para codificação de Telemetria e Telecomando utilizando algoritmos de codificação e correção de erros. A pesquisa realizada naquele trabalho deu ênfase apenas na codificação de TM e decodificação de TC. O trabalho apresentou o projeto de um SoC para codificação de telemetria e decodificação de Telecomando CCSDS utilizando, respectivamente, os algoritmos de correção de erros RS (*Reed-Solomon*) e BCH (*Bose, Chaudhuri and Hocquenghem*) em linguagem VHDL.

Além do projeto em hardware foram apresentadas pesquisas científicas que utilizam o algoritmo de RS em diferentes tipos de aplicações.

Os circuitos implementados em VHDL foram validados através de simulação e também por meio de um aplicativo desenvolvido na linguagem Java executando em um PC. Esse aplicativo envia dados de Telecomando e Telemetria a serem codificados/decodificados nos circuitos prototipados em um dispositivo FPGA, recebendo dados resultantes provenientes do mesmo. Todo o processo de projeto, implementação e validação, visa otimização de área e diminuição no tempo de processamento de codificação e decodificação.

Este trabalho foi à primeira pesquisa do projeto PUC#SAT realizada pelo Grupo de Sistemas Embarcados (GSE). A partir deste trabalho foi possível dar continuidade ao estudo do protocolo CCSDS e desenvolver o protótipo de um Subsistema de Comunicação. Devido a importância do projeto PUC#SAT para o presente trabalho, no Capítulo 3 será realizada uma descrição mais detalhada.

2.5 Contribuição para o Trabalho

Os trabalhos apresentados serviram como base para elaborar e embasar a arquitetura desenvolvida. O estudo comparativo entre as tecnologias, descrito na seção 2.3, foi de grande importância para a escolha da plataforma de desenvolvimento. Os resultados obtidos na comparação realizada entre as tecnologias foram determinantes na validação da sugestão do INPE de utilizar um sistema embarcado em um FPGA ao invés de um processador ou ASIC.

Além dos trabalhos citados, outros estudos da área espacial agregaram conhecimento e foram importantes para o desenvolvimento das atividades. As fontes utilizadas nesses estudos são citadas ao longo do texto, sendo devidamente referenciadas nas “Referências Bibliográficas”.

3. Projeto PUC#SAT

O trabalho proposto foi desenvolvido como parte das pesquisas em andamento no GSE da Faculdade de Informática (FACIN) da Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS). Mais especificamente, no âmbito do projeto PUC#SAT [3], o qual dispõe de financiamento da Agência Espacial Brasileira (AEB)[4] e do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), supervisionado pelo INPE. O ponto central do PUC#SAT, são os aspectos relacionados à implementação em hardware da pilha de protocolos do CCSDS[5][6][7] para utilização no Subsistema de Comunicação[8] de um veículo espacial.

O desenvolvimento do módulo de comunicação, para uso em satélites do INPE, teve inicio no GSE em 2005 no âmbito do programa Uniespaço da AEB. A previsão de término dessa etapa do Uniespaço é para o final de Julho de 2009. De forma a auxiliar na conclusão dos trabalhos em andamento no GSE, o INPE contratou o consórcio formado pelas empresas Aeroeletrônica (AEL) e Innalogics. O objetivo principal da atuação dessas empresas no projeto é fornecer o auxilio necessário ao GSE de forma a transformar os resultados das pesquisas realizadas em um produto que atenda os rigorosos padrões de qualidade do INPE. Outro objetivo é o apoio financeiro necessário para o pagamento da equipe envolvida, uma vez que o financiamento da AEB não possui esse intuito. Dessa forma, a Innalogics e a AEL tem trabalhado junto ao GSE na produção da UTMC [11] que pertence ao Subsistema de Comunicação do Computador de Bordo do “Sistema de Controle de Atitude e Órbita de um Satélite Estabilizado em Três Eixos (SISCAO)”. O modelo segue o fluxo completo de implementação, o qual inclui o projeto mecânico e elétrico, e a codificação em VHDL da descrição de hardware que irá programar o FPGA. Sua arquitetura será detalhada seção 4.1.

O Subsistema de Comunicação em desenvolvimento no PUC#SAT, a UTMC, aparece no canto inferior esquerdo da Figura 4, onde é representado um diagrama de blocos simplificado do ACDH, que é responsável pelo controle de atitude, órbita, e gerência de dados (*Attitude Control and Data Handling* - ACDH)[9]. O ACDH ilustrado é uma Plataforma de Serviços semelhante aquela idealizada pelo INPE para a PMM[10] do PNAE. Essa plataforma viabilizará o reaproveitamento de módulos entre as diversas missões, o que reduzirá não apenas o tempo de projeto e desenvolvimento, mas também os custos associados.

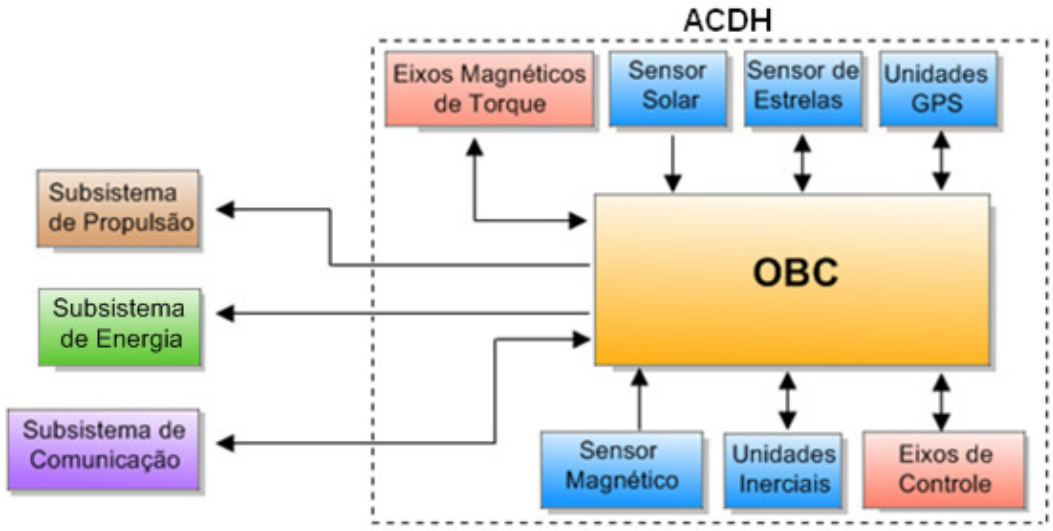

Figura 4 - Diagrama de Blocos do ACDH da Plataforma Multi-Missão.

O ACDH representado na Figura 4 pelo retângulo pontilhado é composto por um Computador de Bordo (*On-Board Computer* - OBC), atuadores e instrumentos responsáveis pelo controle sensorial do satélite. Os subsistemas são gerenciados e configurados através do ACDH, ou seja, existe um canal de comunicação entre o OBC e todos os subsistemas presentes na Plataforma de Serviços do satélite. A Plataforma é formada por quatro subsistemas principais: de propulsão, de comunicação, de potência e o ACDH.

A seguir é realizada uma descrição da interação do subsistema de comunicação com o OBC, e como é realizado o gerenciamento dos dados provenientes dos instrumentos sensoriais.

Na Plataforma de Serviços do satélite, o Subsistema de Comunicação é o módulo responsável pela comunicação com a Estação Terrestre. Através deste os controladores da missão são informados sobre aspectos referentes à "saúde" do veículo espacial e capazes de enviar requisições que serão executadas pelo OBC[9]. Exemplos de comandos enviados incluem os de correção de órbita e atitude do veículo espacial, os de ligar e desligar motores, entre outros. Em contrapartida o satélite necessita enviar respostas a um determinado comando, informações de posição do satélite, situações de exceção ou erro e informações de perda de mensagens. Por essa razão, esse módulo do ACDH é considerado um componente crítico, pois uma falha neste módulo resulta no fim da missão. Além do Subsistema de Comunicação em um satélite existem outras formas de trocar mensagens, uma delas é a transmissão de informações provenientes do *payload*, tais com imagens coletadas e dados de sensoriamento remoto.

As mensagens que partem da Estação Terrestre são chamadas de Telecomando (TC). Um TC pode seguir dois fluxos distintos dentro do Subsistema de Comunicação: ser repassado para que o OBC realize o seu processamento (Telecomando Roteado - TCR), ou realizar o acionamento de algum instrumento diretamente, sem passar pelo OBC (Telecomando Direto - TCD)[7][11]. Essa distinção existe, pois em algumas situações é desnecessário o OBC realizar o processamento do pacote, além disso, em alguns casos o equipamento pode ter sofrido alguma avaria e estar impossibilitado de processar a informação. Nesse caso, o acionamento procede de forma direta. Este fluxo de TC está

representado na Figura 5, onde o bloco “TC” representa a pilha de protocolos implementada em VHDL para a manipulação de Telecomandos, e “TM” a pilha de protocolos implementada de Telemetria. A *Command Link Transmission Unit* (CLTU) que aparece na entrada do bloco de TC contém TCs encapsulados em quadros de TC.

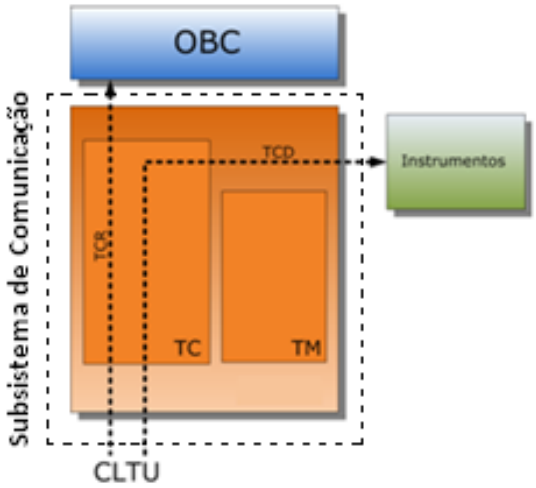

Figura 5 - Fluxo de Telecomandos no Subsistema de Comunicação.

Realizando o fluxo contrário ao do TC estão as mensagens de Telemetria (TM). As TMs possuem caráter informativo, ou seja, relatam com mensagens periódicas aspectos comportamentais relevantes do satélite, ou respondem uma solicitação que foi requisitada por TC [12][13]. Em um caso particular, quando a Telemetria não está enviando nenhuma informação específica, esta é responsável por enviar pacotes vazios, esse mecanismo é utilizado para manter a Estação Terrestre informada que o satélite está respondendo e o canal de comunicação está funcionando corretamente, evitando assim possíveis falhas de comunicação. Este fluxo de Telemetria está representado na Figura 6.

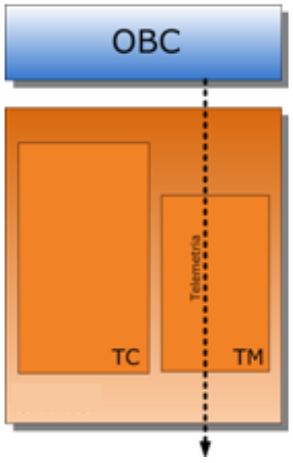

Figura 6 - Fluxo de Telemetria no Subsistema de Comunicação.

Na Figura 7, é possível observar como estão dispostas as camadas do protocolo CCSDS, nos fluxos de Telecomando e Telemetria. Os TCs são processados pela Camada de Codificação, e em seguida repassados a Camada de Transferência, na qual os pacotes são retirados dos quadros e classificados como TCDs ou TCRs. Se for um TCD os pacotes são enviados para a Camada de Empacotamento que realiza a verificação dos dados. Caso o pacote seja válido, seu conteúdo é enviado para a unidade de distribuição de pulso (*Command Pulse Distribution Unit* - CPDU) que aciona o instrumento correspondente e envia para a TM um pacote de confirmação de recepção do TC, porém caso seus dados sejam inválidos é enviado um pacote informando que o TC recebido não atende as especificações. Essas mensagens são encapsuladas em quadros de TM e enviadas para a Estação Terrestre. Entretanto, se for um TCR, o pacote é enviado para o OBC através de uma interface serial síncrona[11][14][6][3].

O Subsistema de Comunicação recebe na Camada de Empacotamento do bloco de TM os pacotes formados pelo OBC, e os armazena para enviá-los assim que seja recebida uma quantidade adequada para envio. Esses dados são repassados para a Camada de Codificação que adiciona os seus dados de controle de acordo com a codificação utilizada, para que após isso, o quadro seja enviado para a Estação Terrestre[13].

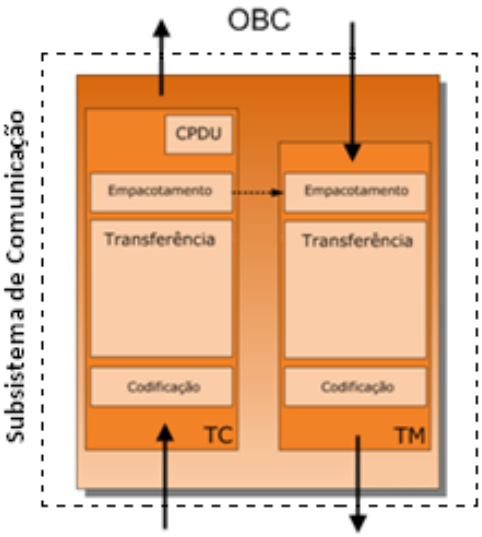

Figura 7 - Fluxo de funcionamento da pilha de protocolos CCSDS.

Para garantir a segurança e a integridade das informações trocadas entre o satélite e os controladores da missão são implementados nos fluxos de TC/TM algoritmos de codificação e decodificação capazes de realizar detecção e correção de erros. O algoritmo BCH está implementado no fluxo de TC, e na TM estão implementados dois algoritmos, o RS[16] e o Convolucional[15], que podem ser aplicados de forma isolada ou combinados, o que garante maior confiabilidade na troca de informações.

Existem diversas arquiteturas de projeto de OBC, uma das mais investigadas atualmente[37] e considerada foco do trabalho é a que utiliza um processador embarcado em FPGA, o qual executa um sistema operacional de tempo real que implementa as rotinas pertinentes ao ACDH. Seguindo as

premissas da arquitetura mencionada, em nível de sistema operacional, são implementados aplicativos que se classificam em três grandes conjuntos: os que atuam sobre o controle de atitude e órbita, os que realizam a manipulação e aquisição dos dados e os que realizam a verificação e controle de energia do satélite [33]. O diagrama de blocos do OBC é ilustrado na Figura 8.

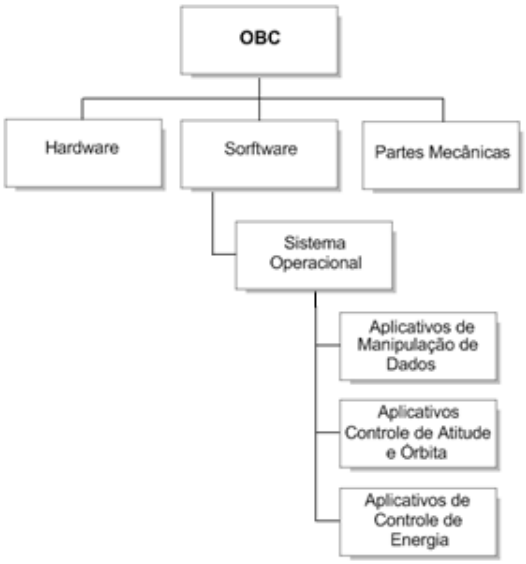

Figura 8 - Diagrama de Blocos do OBC.

Os Aplicativos de Manipulação de Dados realizam o controle do ACDH, ou seja, são as tarefas que gerenciam os módulos funcionais da Plataforma de Serviços, e um dos módulos gerenciados é o Subsistema de Comunicação. Já os Aplicativos de Controle de Atitude e Órbita são as rotinas que implementam os algoritmos de posicionamento inercial, esses aplicativos realizam: controle de órbita, requisitam informações dos sensores e acionam os atuadores [9]. Essas camadas de aplicações do OBC serão detalhadas no Capítulo 5.

Derivado do ACDH existe outra nomenclatura para a Plataforma de Serviços chamada de *On-Board Data Handling* (OBDH), essa nomenclatura se aplica quando o controle de atitude e órbita é realizado por outro módulo de forma independente [41]. A arquitetura do OBDH se restringe apenas a um computador de bordo que se comunica com os demais subsistemas e *payload* realizando o gerenciamento e processamento da comunicação entre os subsistemas.

Nesse contexto, o INPE e os responsáveis pelo desenvolvimento da UTMC foram consultados de forma a identificar aspectos de maior interesse para a continuação do projeto PUC#SAT. O INPE sugeriu que uma contribuição importante para o desenvolvimento da PMM seria elaborar uma arquitetura de hardware do ACDH com processador e sistema operacional de tempo real embarcados em FPGA. Em nível de software a sugestão foi implementar alguns módulos dos Aplicativos de Manipulação de Dados e dos Aplicativos de Controle de Atitude e Órbita. Além disso, foi sugerido que o projeto fosse integrado com a UTMC, para que fosse possível realizar a interface com um Subsistema de Comunicação real e não apenas em condições simuladas. A partir destes aconselhamentos foi elaborado o presente Trabalho de Conclusão de Curso.

Devido ao grau de complexibilidade das Aplicações de Controle de Atitude e Orbita do ACDH, o trabalho concentrou o foco no desenvolvimento de um OBDH bastante simplificado, que tem como principal função realizar a interatividade com o Subsistema de Comunicação. Para agregar e aproximar o projeto ao funcionamento de um ACDH foi utilizado um acelerômetro e um giroscópio, dispositivos utilizados no sensoriamento inercial de satélites. A arquitetura completa do projeto e suas funcionalidades serão detalhadas nos Capítulos 4,5 e 6.

4. Arquitetura Proposta de OBDH para o PUC#SAT

Este capítulo apresenta os componentes utilizados no projeto e implementação da versão simplificada do OBDH proposto, justificando as escolhas realizadas. Após uma visão geral da arquitetura do sistema, os módulos desenvolvidos e adaptados de outros projetos relacionados são devidamente descritos.

4.1 Arquitetura

Existe uma tendência mundial de busca por novas técnicas de projeto visando os patamares de aplicações com necessidades de alto desempenho, considerando baixo consumo de potência e redução de área. Atualmente, grande parte dos equipamentos embarcados em satélites são baseados em processadores comerciais[37]. Esses dispositivos realizam o processamento e distribuem as tarefas para os demais componentes do sistema. Porém essa topologia de arquitetura utiliza porções consideráveis de área em uma placa, além de consumo de potência considerável[37].

Para equacionar esse problema estão sendo adotadas arquiteturas que utilizam processadores embarcados em FPGA[37]. Com essa técnica é possível atingir níveis satisfatórios em relação ao consumo de potência e área de ocupação do circuito. Já que em FPGA é possível embarcar um ou mais processadores em um único circuito integrado. Além disso, as arquiteturas que utilizam um FPGA permitem embarcar periféricos agregados ao processador o que aumenta o desempenho da aplicação.

Partindo dessas considerações o presente projeto do OBDH foi desenvolvido utilizando um processador embarcado em FPGA e um sistema operacional de tempo real, que gerencia as aplicações do OBC, conforme ilustrado na Figura 9.

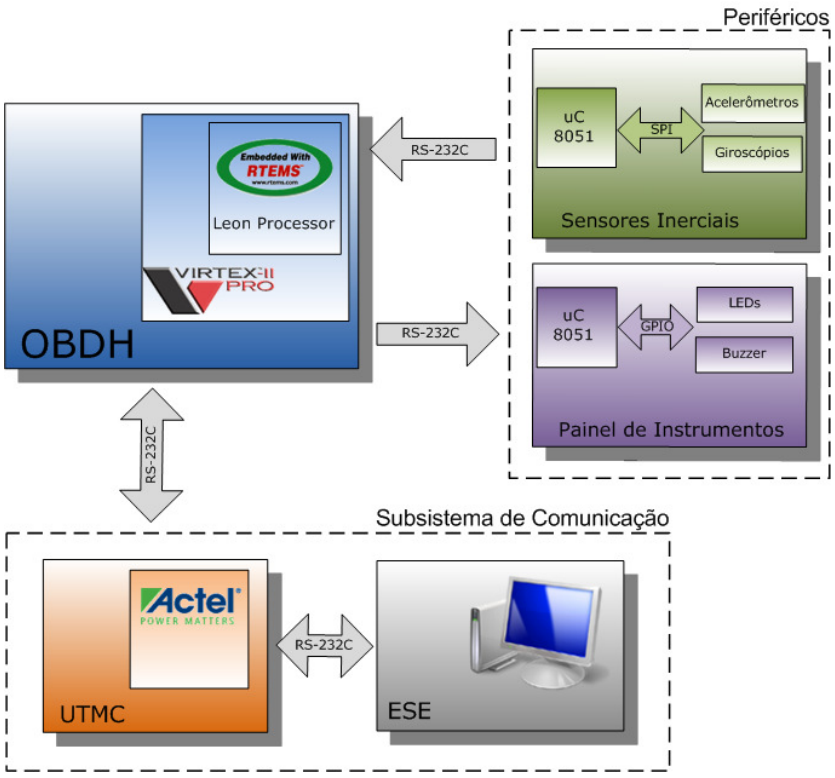

Figura 9 - Arquitetura da Plataforma de Desenvolvimento do OBDH.

Como elemento principal da Plataforma de Desenvolvimento foi utilizado o processador LEON3 embarcado em um FPGA Virtex-II da Xilinx, ilustrado no canto superior esquerdo da Figura 9. Sobre o LEON3 [19] foi utilizado o sistema operacional de tempo real RTEMS [17] que serve de suporte ao desenvolvimento das aplicações do OBDH. Nas seções 4.2 e 4.3 é realizada uma descrição sobre o LEON3 e o RTEMS, respectivamente.

Foram implementados dois periféricos para o LEON3 com funcionalidades pertinentes aos instrumentos do OBDH:

- Sensoriamento Inercial do satélite: Este periférico possui dois sensores, um acelerômetro e um giroscópio. Ambos utilizam o protocolo SPI para enviar dados sensoriais. Um microcontrolador 8051 foi utilizado para receber dados dos sensores e enviar de forma serial para o LEON3. No microcontrolador foram implementadas as funções do protocolo SPI para leitura do sensor e do protocolo RS-232C para enviar os dados para o LEON3.
- Painel de Instrumentos do satélite: Este periférico utiliza um microcontrolador 8051 para simular a ocorrência de eventos no satélite. Basicamente, foram implementados dois eventos: LEDs de visualização da potência dos propulsores e um sinal sonoro indicando falha no sistema. Estes eventos possuem caráter fictício no funcionamento do satélite, porém foram idealizados visando agregar funcionalidades ao OBDH e aumentar o número de possibilidades de Telecomandos

disponíveis no projeto. A comunicação do processador com o microcontrolador também é através do protocolo serial RS-232C.

A interface do OBDH com o subsistema de comunicação, bloco UTMC na Figura 9 e Figura 10 também utiliza o protocolo RS-232C. Graças à conclusão do protótipo da UTMC do PUC#SAT foi possível utilizá-la no projeto, dispensando a necessidade de desenvolver os simuladores do Subsistema de Comunicação inicialmente planejados. A UTMC irá se comunicar com o OBDH e o com o *Electrical Support Equipment* (ESE).

O ESE é um equipamento de teste desenvolvido em paralelo ao projeto do protótipo da UTMC, e seu propósito é implementar rotinas de automação de testes para validação do protocolo que está sendo executado na UTMC. Para um fim específico o ESE desempenha a função da Estação Terrestre, portanto o presente trabalho irá utilizar o ESE para interagir com a UTMC enviando os Telecomandos e recebendo as Telemetrias. O ESE será detalhado na seção 4.6.7.

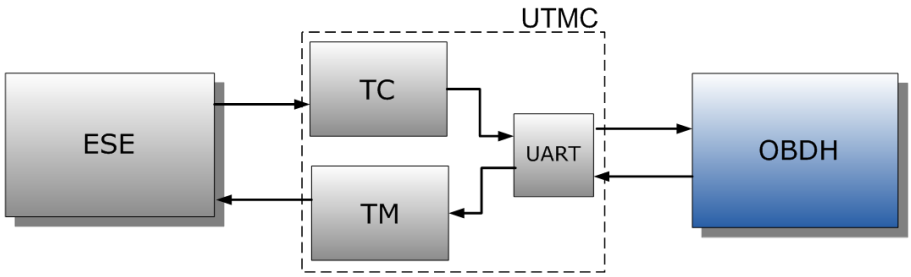

Figura 10 - Diagrama de Fluxo de Comunicação.

A Figura 10 representa o diagrama de fluxo da comunicação do projeto. Assumindo o ESE como ponto de partida da Comunicação, desempenhando a função da Estação Terrestre, os quadros de TC são montados e enviados para a UTMC. Na UTMC, o quadro passa pelas camadas da pilha de protocolos de TC, extraindo pacotes que são enviados para o OBDH. O OBDH irá desempacotar esses dados, e interpretar os comandos existentes na área de dados. A partir do comando, o OBDH irá tomar uma atitude e enviará para a UTMC um pacote de TM. Na UTMC esse pacote é encapsulado em um quadro de TM e enviado para o ESE, completando assim o fluxo de comunicação.

4.2 Processador

A escolha do processador utilizado nesse trabalho de conclusão levou em consideração a opinião dos engenheiros do INPE, que aconselharam a utilização do processador LEON3. O grupo de TCC seguiu essa recomendação embarcando o LEON3 no FPGA Virtex II da placa de desenvolvimento Xilinx XUP V2-Pro. O INPE planejou inicialmente a utilização do processador ERC32 [42] para as futuras missões da PMM. Porém, devido à experiência na utilização do processador LEON3 por parte da agência espacial européia, existe interesse nesse processador que acabou sendo sugerido para a implementação desse projeto.

A placa de desenvolvimento Xilinx XUP V2-Pro disponibiliza uma plataforma avançada de hardware, que consiste em um FPGA Virtex-II Pro de alto desempenho cercado por um conjunto

de componentes, que podem ser utilizados para criar um sistema complexo e demonstrar toda a capacidade do FPGA. Entre os componentes da placa existe uma porta serial RS-232C e conectores de expansão que serão utilizados para comunicação com os periféricos. A compatibilidade da placa de desenvolvimento Xilinx XUP V2-Pro com o processador LEON3 e os periféricos disponíveis na placa, são fatores importantes na escolha dos componentes utilizados, sendo que estão disponíveis na Internet diversos projetos utilizando essa combinação de processador e FPGA.

O processador LEON3 é uma descrição VHDL sintetizável de um processador de 32-bits compatível com a arquitetura SPARC V8 [19]. O modelo é altamente configurável, o código fonte completo está disponível para download sob a licença GPL, permitindo que o processador seja utilizado sem restrições em aplicações não comerciais. O processador possui as seguintes características:

- Conjunto de instruções do SPARC V8;
- *Pipeline* avançado de sete estágios;
- Unidades de MAC;
- *Cache* de instruções e de dados separadas (Arquitetura Harvard);
- AMBA-2.0 AHB *bus interface*;
- Suporte a *on-chip debug*;
- *Symmetric Multi-processor support* (SMP);
- Até 125 MHz em FPGA e 400 MHz em ASIC com tecnologia de 0.13µm;

O LEON3 é razoavelmente parametrizável através de configurações em seu VHDL, sem alterar nenhum pacote de configuração global. Assim, é possível instanciar vários núcleos em um mesmo projeto, com diferentes configurações. As características do LEON3 podem ser definidas através de uma ferramenta gráfica, facilitando a configuração do processador. A ferramenta de configuração pode alterar grande parte de suas características, e permite adicionar periféricos *on-chip* como memórias, interfaces de rede entre outros.

O processador LEON3 pode ser sintetizado com as principais ferramentas comerciais como Synplify, Synopsys DC, Cadence RC, Xilinx XST e Altera Quartus, utilizando *scripts* ou a interface gráfica das ferramentas.

O LEON3 foi certificado pela SPARC *International* como sendo compatível com o SPARC V8, tornando possível que aplicações desenvolvidas para a arquitetura SPARC V8 sejam utilizadas no LEON3.

4.3 Sistema Operacional

Em aplicações críticas, que necessitem de um grau elevado de confiabilidade e com requisitos de tempo real, é necessário que o sistema operacional garanta que todas as tarefas sejam cumpridas em um tempo pré-determinado, evitando que eventos críticos passem despercebidos e causem falhas na aplicação. A utilização de um sistema operacional de tempo real é importante devido à complexidade de um OBDH, que necessita responder aos sensores,

comandos e solicitações de acordo com prazos bem definidos, garantindo a integridade da aplicação.

Sistemas de tempo real caracterizam-se pela necessidade fundamental de manter um sincronismo constante entre as tarefas, isto é, o sistema deve atuar de acordo com a dinâmica de estados do processo, levando em conta os tempos de execução e as prioridades de cada tarefa [38].

Sistemas de tempo real não dependem somente do resultado lógico de computação, mas também do tempo em que os resultados são produzidos, onde diversas tarefas são executadas e o escalonamento em função das restrições temporais é um grande problema. Tarefas recebem dados de entrada, executam um algoritmo e geram saídas. A tarefa está logicamente correta se gerar uma saída correta em função dos dados de entrada em um prazo temporal satisfatório [39]. Devido a isso se pode afirmar que, o tempo é o principal objetivo dos sistemas de tempo real, pois as tarefas executadas pelo processador devem ser concluídas antes de seu limite temporal (*deadline*) [40].

A escolha do sistema operacional levou em consideração o aconselhamento dos Engenheiros do INPE, que habituados a trabalhar com aplicações espaciais sugeriram a utilização de um sistema operacional de tempo real comumente utilizado nesse tipo de aplicação, o RTEMS (*Real Time Executive for Multiprocessor Systems*).

O RTEMS é um sistema operacional de tempo real desenvolvido para sistemas embarcados. Foi criado no início dos anos 80, sendo a primeira versão comercial disponibilizada em 1993. Atualmente o projeto RTEMS é gerenciado pela *OAR Corporation*[17]. Esse sistema operacional é amplamente utilizado em aplicações espaciais, pois dá suporte a vários microprocessadores empregados nesse tipo de aplicação, entre eles SPARC, ERC32, LEON, MIPS Mongoose-V, Coldfire e PowerPC. Como exemplo de utilização é possível citar um dos satélites de reconhecimento que está orbitando o planeta Marte, o qual utiliza o RTEMS integrado ao módulo responsável pela comunicação. O satélite *Mars Reconnaissance Orbiter* (MRO) é uma espaçonave produzida e lançada pela NASA com o intuito de reconhecer e explorar a orbita de Marte[18].

4.4 Microcontroladores

Microcontroladores são usados em sistemas embarcados que não necessitam de poder computacional e de recursos existentes em desktops e notebooks. Microcontroladores são circuitos integrados que agregam as principais funcionalidades de um computador, acrescidos de funcionalidades específicas para uso em sistemas embarcados. Estão presentes nas mais diversas aplicações tais como em automóveis, televisões, rádios, máquinas fotográficas digitais, microondas, elevadores, entre outros.

Por possuírem integrados em um único chip recursos de memória, processamento e controle, os microcontroladores têm certas limitações de recursos computacionais quando comparados aos microprocessadores [27].

Os microcontroladores mais comumente encontrados no mercado são baseados nas arquiteturas 8051, Zilog Z80, PIC e ARM. Exemplos largamente utilizados de microcontroladores incluem o Intel P8051, baseado na arquitetura 8051, e o TMS470 baseado na arquitetura ARM. O preço dos chips possui uma grande variação, podendo custar desde alguns centavos a centenas de dólares.

Neste trabalho foram utilizadas duas placas com o microcontrolador 8051 confeccionados na disciplina de Laboratório de Processadores. O uso destas é devido ao conhecimento de programação e manuseio adquirido durante a disciplina. Estas placas utilizam o conversor AD/DA MSC1213 da Texas Instruments, que possui integrado um microcontrolador baseado no 8051[43]. As placas possuem diversos recursos como LEDs, conversores analógico/digital, comunicação serial, driver de potência, entre outros.

Os microcontroladores atuais possuem diversas interfaces de comunicação, permitindo a utilização dos mais variados periféricos incluindo sensores e atuadores. Alguns dos principais sensores disponíveis utilizam o protocolo *Serial Peripheral Interface* (SPI)[28]. Este protocolo é bastante usado para acesso a memórias (*flash*, E2PROM, SD, MMC) e para comunicação entre diversas controladoras, por ser um protocolo bastante rápido e que requer poucos fios para sua implementação.

No protocolo SPI, a comunicação é síncrona, ou seja, há um sinal de relógio comum a todos os dispositivos ligados no mesmo barramento SPI. A freqüência do relógio geralmente varia entre 1Mhz e 70Mhz, usufrui da comunicação full duplex, onde se envia e recebe dados no mesmo ciclo de relógio. A comunicação é realizada no modo mestre-escravo, existindo um sinal adicional para cada dispositivo ligado no barramento, o qual habilita a comunicação do dispositivo com o mestre. Um exemplo de comunicação SPI é demonstrado na Figura 11.

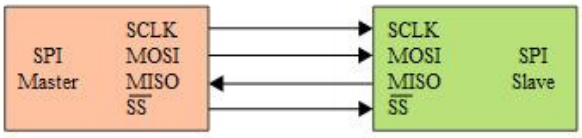

Figura 11 - Interface SPI Mestre/Escravo.

Em suma, há um sinal de relógio, gerado pelo mestre, chamado *Serial Clock* (SCLK); um barramento unidirecional de um bit para envio de dados do mestre para os escravos, chamado *Master Output*, *Slave Input* (MOSI); outro barramento unidirecional de um bit para envio de dados dos escravos para o mestre, *Master Input*, *Slave Output* (MISO); e mais um sinal adicional para habilitar a comunicação com cada um dos escravos, chamado *Slave Select* (SS). Caso existam diversos escravos, estes são denominados SS1, SS2, etc.

4.5 Periféricos

Essa seção tem como objetivo apresentar os conceitos das tecnologias que são utilizadas nos periféricos de sensoriamento da presente implementação, facilitando a compreensão do trabalho. Inicia-se explicitando o funcionamento de sensores e a tecnologia dos transdutores

atuais. Após, é apresentada a teoria de construção dos sensores de aceleração e rotação. Por fim, uma descrição do painel de instrumento utilizado.

4.5.1 Sensores, MEMS e INS

Sensores utilizados em sistemas computacionais são dispositivos que percebem sinais e os transformam em níveis elétricos (de forma a tornar possível fazer medições eletronicamente) [22]. Existem vários tipos de sensores, podendo-se citar os indutivos (percebem variação do campo magnético), capacitivos (percebem variação na capacitância), resistivos (percebem variação da resistência), ópticos (percebem variação da luz), de gravidade (percebem alteração da gravidade sofrida), dentre outros que podem perceber radiação, vibração, elementos químicos, entre outros. A partir destes sensores, é possível medir luz, som, temperatura, pressão, umidade, resistência elétrica, tensão elétrica, corrente elétrica, proximidade, velocidade, posição, vazão, e outras grandezas.

Alguns dos sensores atuais utilizam em sua construção uma tecnologia denominada de Sistemas Microeletromecânicos (*MicroElectroMechanical Systems* - MEMS) a qual é resultado da integração de elementos mecânicos e eletrônicos em um mesmo substrato de silício [23]. Os MEMS são produzidos com o mesmo processo utilizado na fabricação dos circuitos integrados modernos, assim é possível a construção, por exemplo, de motores elétricos menores que o diâmetro de um fio de cabelo. Existem, entretanto, alguns processos que não são derivados da tecnologia de circuitos integrados. Sensores MEMS são dispositivos como sensores de pressão, acelerômetros, girômetros e giroscópios que observam uma grandeza do ambiente e produzem um sinal elétrico proporcional a medida desta grandeza. Atualmente, MEMS são utilizados em diversas aplicações, como alarmes de automóveis, controle de movimento de robôs, controle de elevadores, sistemas de navegação inercial entre outros.

Sensores do tipo acelerômetros e giroscópios são normalmente utilizados em sistemas de navegação inercial (*Inertial Navigation Systems* - INS). O INS foi demonstrado pela primeira vez em 1949 por C. S. Draper, e desde então tem sido um método de navegação utilizado para fins militares e comerciais[26].

Todo objeto livre para se mover no espaço tem seis “graus de liberdade” – ou meios para se mover. Há três graus lineares de liberdade (x, y, z) que especificam sua posição e três graus de liberdade rotacionais, theta (*pitch*), psi (*yaw*), e phi (*roll*), que especificam sua atitude. Sendo conhecidas essas seis variáveis, então é possível saber sua localização – algo muito útil para controle de veículos espaciais. Se for possível obter esses dados em determinados períodos de tempo, então também é possível determinar quão rápido está se movendo, e qual a sua aceleração. Para tanto são utilizados acelerômetros e giroscópios para medir como um veículo espacial está acelerando e/ou rotacionando no espaço.

Sistemas de navegação inercial são utilizados em diversas situações onde o uso de referência externa para medir posição é impraticável. Estes sistemas, tipicamente, são muito utilizados na área marítima e aeronáutica e são compostos por alta tecnologia, custando algumas centenas e até milhares de dólares. No entanto, é possível a elaboração

de um sistema semelhante por um custo mais acessível através de acelerômetros e giroscópios do tipo MEMS.

4.5.2 Giroscópios

O giroscópio é um dispositivo para medir ou manter a orientação, baseado nos princípio de conservação de momento angular de um corpo[24]. Representa a grandeza física que relaciona a distribuição da massa do corpo ao redor de um eixo de rotação com sua velocidade angular.

A essência deste dispositivo consiste em uma roda livre, ou várias rodas, para girar em qualquer direção e com uma propriedade: estas se opõem a qualquer tentativa de mudar sua direção original. Exemplo disso é girar a roda de uma bicicleta no ar e tentar mudar a sua posição bruscamente, uma reação a este movimento será sentida.

Existem giroscópios completos, que atuam em todas as direções, e giroscópios simples, que atuam apenas em uma direção. Os três graus de liberdade rotacionais *pitch*, *yaw*, e *roll* estão representados na Figura 12.

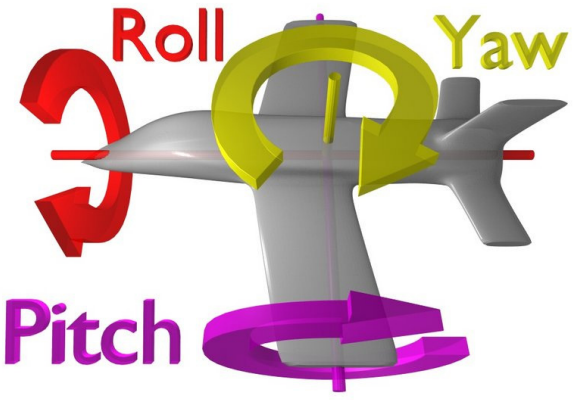

Figura 12 - Movimentos do Giroscópio.

No presente trabalho é utilizado o giroscópio ADIS16250 da empresa Analog Devices [35]. Este componente foi adquirido pela empresa Innalogics há algum tempo e foi disponibilizado para ser utilizado neste trabalho. Porém, foi necessário desenvolver as placas de circuito impresso e o código para comunicação com o dispositivo.

4.5.3 Acelerômetros

Acelerômetros são sensores capazes de converter a aceleração ou o movimento em sinais elétricos [25]. Para realizar essa conversão é utilizado um sistema massa e mola onde uma massa é sustentada por pontos de apoio fixos, como representado na Figura 13.

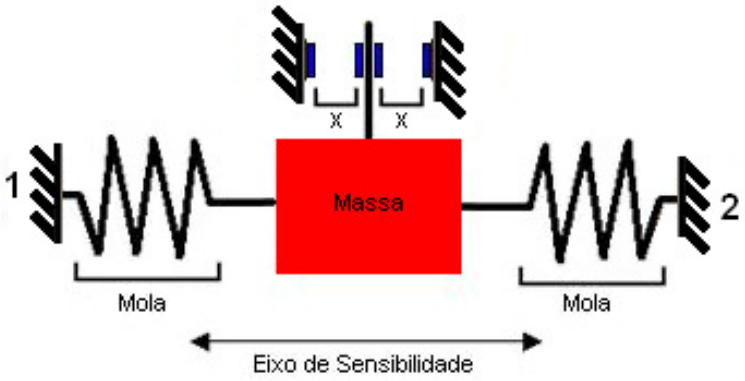

Figura 13 - Princípio de funcionamento de um acelerômetro.

Quando essa massa sofre uma aceleração, ela se desloca em relação aos pontos 1 e 2 ilustrados na Figura 13. Junto dessa massa são adicionadas placas criando um acoplamento capacitivo conforme apresentado na parte superior da Figura 13. A capacitância de um capacitor é dada por C=k/x, onde 'x' é a distância entre essas placas que formam os capacitores e 'k' é a propriedade do material colocado entre as placas. Portanto, se conhecermos a capacitância 'C' e a constante 'k', podemos determinar a distância entre as placas.

Quando a massa está em repouso, a distância entre as placas é a mesma e ambos capacitores possuem capacitância equivalente. Quando o sistema sofre uma aceleração a capacitância do conjunto capacitivo varia. Monitorando essa variação no tempo é possível medir a aceleração sofrida pela massa.

No presente trabalho é utilizado o acelerômetro ADIS16201 da empresa Analog Devices [34]. Este componente também foi adquirido pela empresa Innalogics e foi disponibilizado para ser utilizado neste trabalho. Também foi necessário desenvolver as placas de circuito impresso e o código para comunicação com o dispositivo.

4.5.4 Painel de Instrumentos

O painel de instrumentos visa aumentar a possibilidade de TCs interpretados pelo OBDH e exercer o papel de um periférico adicional, representando visual e sonoramente os TCs recebidos. Este painel é utilizado nesse trabalho para simular o acionamento dos instrumentos de um satélite. Os instrumentos reais são, na maioria, do tipo ON/OFF, ou seja, de forma a serem ativados recebem um pulso “ON” com uma duração pré-definida de acordo com os requisitos da missão. Os componentes selecionados para o painel de instrumentos possibilitam a visualização da ativação/desativação dos sinais em questão.

Para representar o painel de instrumentos foi utilizada uma das placas mencionadas na seção 4.4. A placa disponibiliza uma série de componentes conectados nas entradas e saídas do microcontrolador. Destes componentes foi utilizado um barramento de 8 LEDs e o *buzzer*. Os componentes serão ativados através do envio de um TC para o OBDH, no caso do TC ser destinado aos instrumentos, o OBDH enviará para o painel via serial RS-232C os dois bytes representando o comando a ser utilizado. Os comandos possíveis são: ligar LEDs,

representado por dois números no formato hexadecimal que indicam quais os LEDs a serem ligados e ligar *buzzer*, representado por um caractere G seguido de um valor hexadecimal que indica o tempo em segundos que deve permanecer ligado.

4.6 UTMC

A UTMC é o Subsistema de Comunicação utilizado na arquitetura do projeto do OBDH. Um protótipo da UTMC desenvolvido na PUCRS para ser utilizado pelo INPE na PMM. A partir desse protótipo o INPE poderá evoluir o seu funcionamento e produzir a versão de vôo que será embarcada nos próximos satélites. O projeto final será implementado com tecnologias anti-radiação e estratégias de desenvolvimento com alta confiabilidade. Essas técnicas são utilizadas com o intuito de evitar possíveis danos irrecuperáveis nesta unidade. Nesta seção será realizada uma descrição do funcionamento e serão abordados alguns detalhes da implementação da UTMC.

4.6.1 Hardware

A UTMC não possui elementos de processamento físicos, sua arquitetura baseia-se na utilização de uma FPGA, que implementa a pilha de protocolos CCSDS/ESA do Subsistema de comunicação. A FPGA utilizada é a ProASIC3E A3PE1500 com encapsulamento de pinos PQ208[44], essa FPGA possui 1.5 milhões de portas lógicas, memória interna em torno de 33 Kbytes, pinos de I/O diferenciais (LVDS, LVPECL, B-LVDS) entre outras características. A escolha desse FPGA levou em consideração as semelhanças com a FPGA que será utilizada no modelo de vôo da UTMC, com tecnologias *anti-fusion* e EDAC em memória[50].

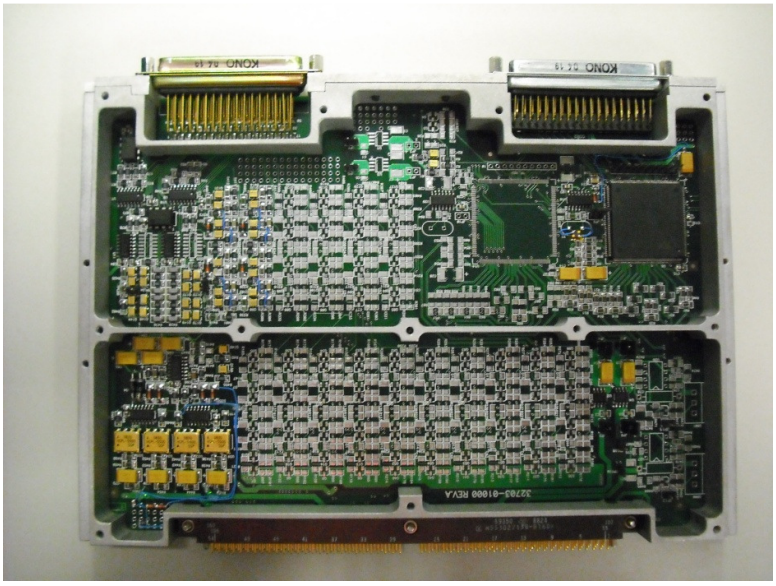

Figura 14 - Hardware da UTMC.

Utilizando a FPGA mencionada a AEL junto com a Innalogics desenvolveu o hardware do protótipo da UTMC. O hardware possui alimentação de 3,3V, 5V e 15V. Essas alimentações servem para energizar além da FPGA, outros componentes da placa. A Figura 14 exibe a estrutura e o hardware da UTMC, onde podem ser observados os dois FPGAs, principal e redundante, no canto direito superior da placa. Pode-se observar também os dois

conectores utilizados para realizar a comunicação com as portas seriais síncronas na parte superior da Figura 14, e o conector para conexão com o ESE, conforme explicado na seção 4.6.7, na parte inferior.

4.6.2 Pilha de Protocolos CCSDS/ESA

A UTMC implementa a pilha de protocolos de TC e TM baseados no CCSDS/ESA. A Figura 15 representa as camadas da pilha de protocolos.

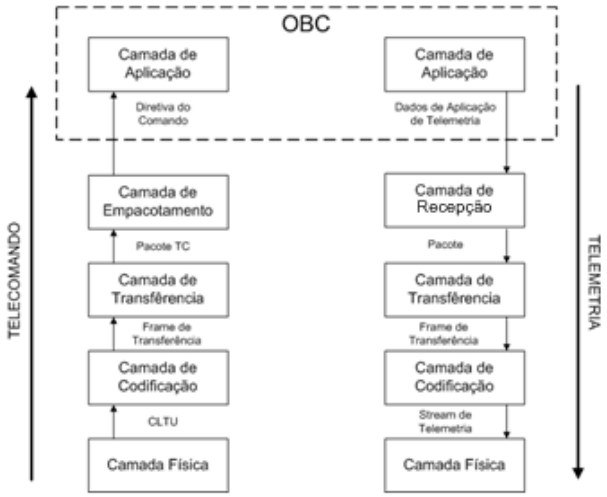

Figura 15 - Pilha de Protocolos CCSDS/ESA.

As seções 4.6.3 e 4.6.4 descrevem os fluxos de TC e TM respectivamente. O protótipo da UTMC possui quatro implementações de códigos VHDL, cada versão possuindo uma técnica de codificação de Telemetria diferente. Em uma versão o algoritmo de codificação é o RS, em outra o algoritmo é o Convolucional. Outra versão possui os dois algoritmos combinados. E existe uma quarta versão que não implementa nenhum algoritmo de codificação.

4.6.3 Telecomando

Inicialmente, os dados recebidos pelo módulo de telecomunicação (rádio-freqüência) do satélite são convertidos para o formato digital e repassados para a camada de codificação no formato de CLTUs. A camada de codificação possui um módulo BCH, desenvolvido no PUC#SAT, responsável por decodificar o quadro recebido e corrigir possíveis erros ocasionados durante o processo de transmissão do TC para o segmento espacial. Depois de decodificados, os dados são repassados para a camada de transferência, onde o módulo *Frame Acceptance and Reporting Mechanism* (FARM) indica a aceitação ou rejeição do quadro recebido. Isso é feito através da verificação do número de seqüência contido no cabeçalho do quadro do frame. Quando o número de seqüência corresponde ao esperado, um sinal de indicação é passado para a camada de transferência informando que o dado pode ser processado. Caso contrário o quadro deve ser rejeitado. O módulo FARM então rejeita o quadro e envia um pedido de retransmissão por intermédio de uma *Command Link Control Word* (CLCW) para o módulo de TM.

Ainda na camada de transferência, é identificado o tipo de TC recebido, ou seja, TC direto ou TC roteado. Os TCs roteados são repassados para o *On-Board Computer* (OBC), que desempacotará os comandos e os distribuirá para os demais subsistemas do satélite. Entende-se por TC direto todo comando que deverá ser processado com uma prioridade alta, diretamente, pelos instrumentos presentes no veículo espacial. A camada de empacotamento realiza o desempacotamento dos comandos diretos, e também o envio de ACKs (indicando recebimento de um pacote válido) e NACKs (indicando um erro na integridade do pacote) para o fluxo de TM. Os TCs diretos desempacotados são enviados para a CPDU, que é responsável por enviar esses TCs diretos para os instrumentos. Os bits de um TC direto são codificados em forma de pulsos e enviados de forma paralela para os instrumentos. Esse módulo é responsável, também, por garantir a duração de 13 milissegundos de um pulso enviado a um instrumento, conforme determinação do INPE.

É importante notar que a implementação em FPGA do fluxo de TC consiste nos módulos "Camada de Codificação", "Camada de Transferência", "FARM", "Camada de Empacotamento" e "CPDU". Esses módulos foram implementados em VHDL, inicialmente para configuração de FPGAs Xilinx, e posteriormente para FPGAs Actel.

4.6.4 Telemetria

A TM recebe os pacotes provenientes do OBC na Camada de Recepção, verifica se as camadas inferiores estão ocupadas, caso não estejam, o pacote é enviado para a Camada de Transferência, que é responsável por encapsular esse pacote em um quadro de Telemetria. Essa Camada preenche o cabeçalho do quadro, adiciona o CLCW recebido do TC e calcula o *Cyclic Redundancy Check* (CRC) do quadro. Terminando essa etapa o quadro é enviado para a Camada de Codificação que codifica o quadro utilizando os algoritmos RS, Convolucional ou ambos. O quadro pode também não ser codificado. Após passar pela Camada de Codificação os dados são enviados para a Estação Terrestre através de uma serial síncrona, com taxa de transmissão de 650 Kbps.

Quando o OBC não está enviando pacotes para a UTMC, a mesma se encarrega de enviar pacotes vazios para a Estação Terrestre. Esse mecanismo é utilizado para que sempre haja comunicação entre o satélite e a Estação Terrestre eliminando a hipótese do canal ficar "em silêncio".

4.6.5 Contribuição do Grupo ao Projeto UTMC

Um dos integrantes do trabalho de conclusão participou ativamente do processo de desenvolvimento da UTMC. Paralelo ao planejamento do OBDH esse integrante implementou o fluxo completo da pilha de protocolos CCSDS/ESA da Telemetria. Este fato foi determinante para a conclusão do projeto e tornou possível a sua utilização no presente trabalho.

Além disso, foi necessário realizar algumas adaptações na implementação da UTMC para integrar seu funcionamento ao do OBDH, fato que foi facilitado pois o grupo já conhecia o funcionamento e o código do projeto.

4.6.6 Integração com o Projeto

O Protótipo da UTMC necessitou sofrer algumas alterações para se adequar, de forma simplificada, ao projeto proposto no presente trabalho. Algumas alterações na codificação VHDL e em ligações físicas foram necessárias para integrá-la ao projeto.

A alteração mais significativa foi a utilização de uma serial assíncrona para comunicação da UTMC com o OBDH. Para isso, foi adicionado ao código VHDL o módulo serial UART, desenvolvido pelo Grupo de Apoio ao Hardware (GAPH), e implementado uma entidade que transforma as saídas seriais síncronas da UTMC em assíncronas.

Outro detalhe importante é que a versão de UTMC utilizada nesse trabalho não utiliza nenhum algoritmo de codificação na Telemetria.

4.6.7 ESE

O conceito do Equipamento de Suporte Elétrico (ESE) é prover hardware e software para testar de maneira eficiente os componentes de um equipamento elétrico. O ESE conceitualmente é composto de três partes principais: sistema de configuração e testes, sistema de execução de testes e sistema pós-processamento.

Para o projeto da UTMC, foi desenvolvido um ESE que é integrado eletricamente com a UTMC. Sua principal função elétrica é disponibilizar de forma acessível a comunicação com um dispositivo externo através de portas seriais síncronas. O hardware do ESE serve também como painel para indicar possíveis eventos, oriundos da UTMC, utilizando LEDs. Toda comunicação com a UTMC é realizada através do ESE que possui 2 entradas e 2 saídas de comunicação. Na Figura 16 o ESE é representado em maiores detalhes.

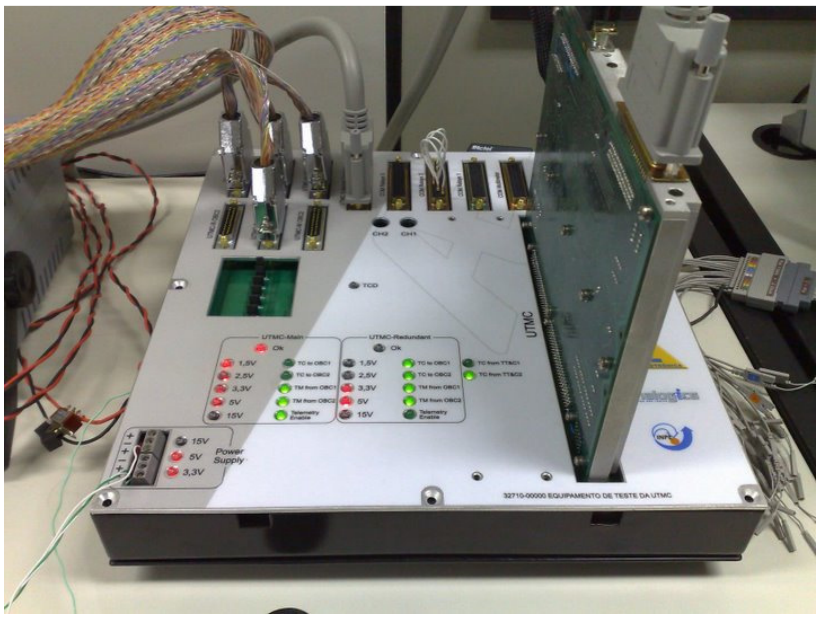

Figura 16 - Hardware do ESSE.

No canto inferior esquerdo da Figura 16 encontram-se as entradas de alimentação do ESE (3,3V, 5V e 15V). Logo acima estão conjuntos de LEDs utilizados para indicação de

eventos tais como: envio serial de Telecomandos, tensões de alimentação, recepção de Telemetria, entre outros. Na parte superior da Figura 16 aparece um conjunto de conectores utilizados para comunicação serial e ativação dos pulsos de Telecomandos Diretos. A placa da UTMC, contendo os FPGAs, encontra-se fixada verticalmente em um *slot* do ESE. Atrás da placa da UTMC, oculto na Figura 16, encontram-se a conexão do dispositivo FlashPro3 da Actel, utilizado para gravação dos FPGAs.

Aliado ao protótipo elétrico foi desenvolvido um software na linguagem de programação LabView. A principal função desse software é gerar vetores de entrada (Telecomandos e Telemetrias) que testam de forma automática o funcionamento da UTMC. Através de uma interface gráfica o usuário pode realizar a configuração e acompanhar o funcionamento da UTMC. Além disso, o ESE é responsável pela interpretação e verificação dos dados enviados e coletados da UTMC.

Na integração da UTMC com o OBDH, a versão utilizada do ESE possui menos funções do que a versão utilizada na UTMC, uma vez que não é mais necessário desempenhar as funções de OBC. Para integrar com o projeto do trabalho de conclusão, o ESE passou a gerar os Telecomandos e ficar na espera de um pacote de dados recebido através de uma Telemetria.

5. Componentes de Software do OBDH

Os módulos de software visam realizar a manipulação dos dados provenientes dos sensores, o controle de atitude e órbita e o controle de energia do satélite. Neste capítulo será realizada uma descrição dos módulos que compõem o software do OBDH, destacando aqueles que foram desenvolvidos no projeto simplificado deste presente trabalho.

5.1 Arquitetura

Os componentes de software necessitam da arquitetura de desenvolvimento, descrita no Capítulo 4, como suporte para a execução das aplicações. O principal requisito das aplicações é temporal. O Sistema Operacional (SO) deve ser capaz de cumprir os requisitos de tempo das aplicações, por essa razão a Arquitetura do OBDH deve ser projetada com um Sistema Operacional de Tempo Real que seja capaz de gerenciar aplicações multitarefa.

A arquitetura do software do OBDH é dividida em duas camadas: a Camada do Sistema Operacional e a Camada de Aplicação. A camada do SO faz referência às suas primitivas, funções nativas, estruturas, além de englobar as funções de acesso ao kernel, os drivers das interfaces de entrada e saída e alguns serviços. Já a Camada de Aplicação é onde se situam os módulos responsáveis pelo processamento e gerenciamento das informações, e integração com os demais subsistemas do satélite.

A Camada de Aplicação apresentada na Figura 17 é composta por quatro módulos de softwares: Software de Controle do OBDH, Software do Subsistema de Controle de Atitude e Órbita, o Software do Subsistema de Energia e o Software de Manipulação do *Payload*. Devido ao curto tempo para desenvolvimento do trabalho foram abordados apenas alguns módulos simplificados do Software de Controle do OBDH e do Software de Controle de Atitude e Órbita, que serão explicados nas seções 5.2 e 5.3.

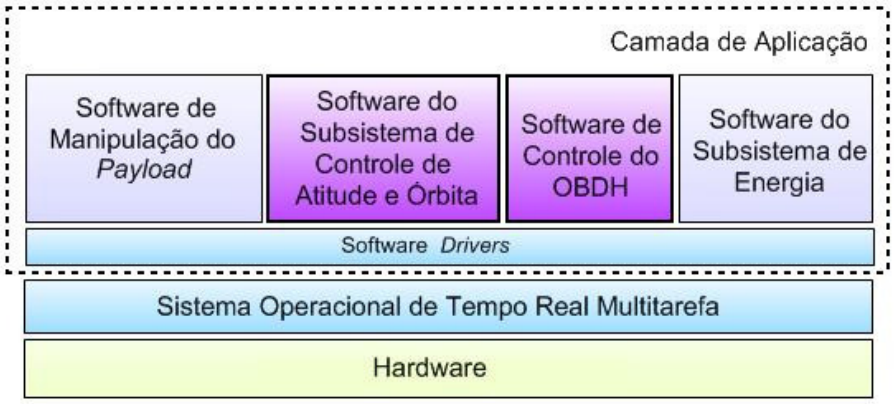

Figura 17 - Visão Geral da Arquitetura da Plataforma de Serviços.

A área pontilhada da Figura 17 representa a Camada de Aplicação da Plataforma de Serviços onde se situam os grupos de softwares que controlam e realizam o sensoriamento do satélite. Os retângulos grifados representam os módulos que serão descritos no presente trabalho e que foram implementados de forma simplificada no projeto do OBDH.

5.2 Software de Controle do OBDH

O módulo de Controle tem a finalidade de gerenciar os demais módulos do OBDH e realizar a interface com o Subsistema de Comunicação. Além disso, esse módulo é o responsável por gerenciar o sistema na sua totalidade. O módulo realiza o controle do tempo, a manipulação dos Telecomandos e das Telemetrias, o gerenciamento de memória e a manipulação dos dados provenientes do *Payload*. A Figura 18 demonstra as tarefas que são gerenciadas pelo Software de Controle do OBDH [41].

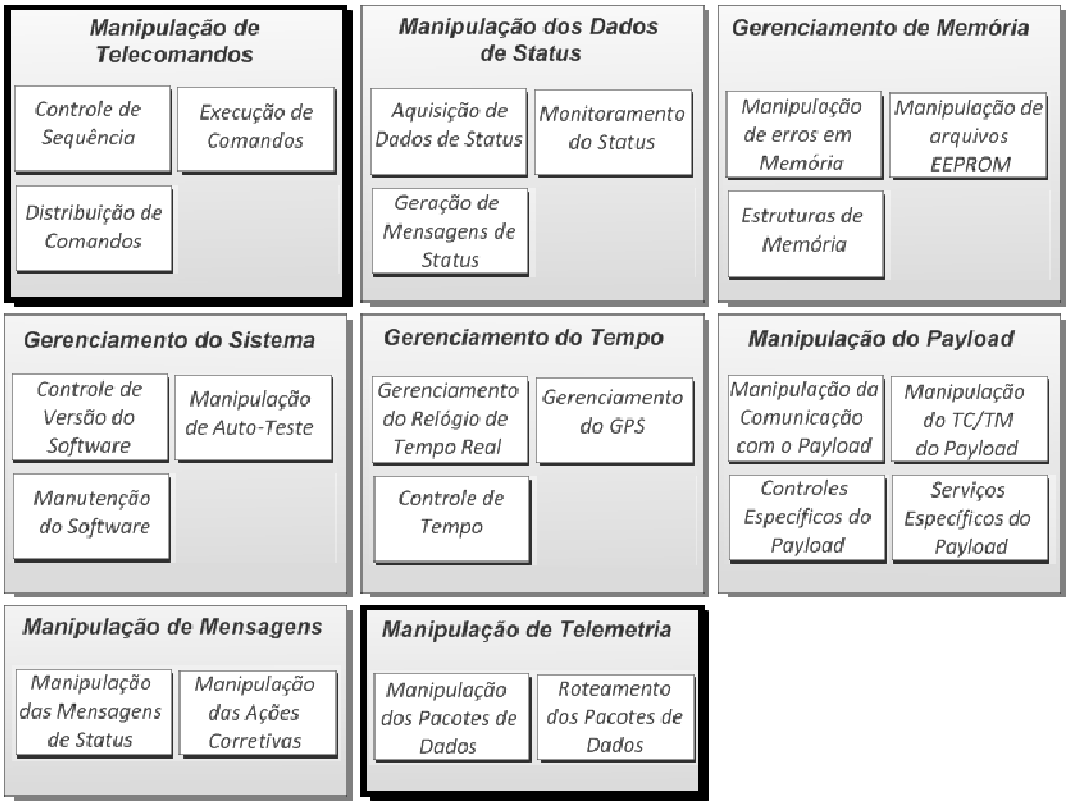

Figura 18 - Software do Subsistema OBDH.

Devido, principalmente, à restrição de tempo disponível para o desenvolvimento, no presente trabalho foram implementados dois módulos dos que foram citados no parágrafo anterior. Um que interpreta TCs provenientes do subsistema de comunicação e outro que encapsula e envia pacotes de TM para o canal de comunicação. Esses módulos estão em destaque na Figura 18. Os módulos implementados são descritos na seção 5.2.1 e 5.2.2.

5.2.1 Manipulação de Telecomandos

A tarefa do sistema de tempo real responsável pela recepção dos telecomandos tem como funções: verificar a chegada de novos telecomandos, executar os comandos recebidos e, quando necessário, ativa o módulo de manipulação de Telemetria. A tarefa inicia sua execução configurando a porta serial designada. Caso nenhum erro ocorra, a tarefa entra em um laço aguardando o recebimento de uma seqüência pré-definida de início de transmissão.

Através das diretivas do RTEMS existe o controle de ocupação do processador, onde cada tarefa possui um tempo limite de operação e no seu término o processador é alocado para outras tarefas.

No momento em que uma seqüência de início de transmissão válida é identificada um segundo contador de tempo é iniciado e a tarefa começa a armazenar todos os dados recebidos. Essa tarefa encerra somente quando uma seqüência de parada válida for identificada ou quando o tempo máximo de espera é alcançado e os dados são descartados.

Caso as seqüência de início e fim da transmissão estiverem corretas a tarefa inicia a validação do Telecomando através dos campos de seu cabeçalho, que estão ilustrados na Figura 19.

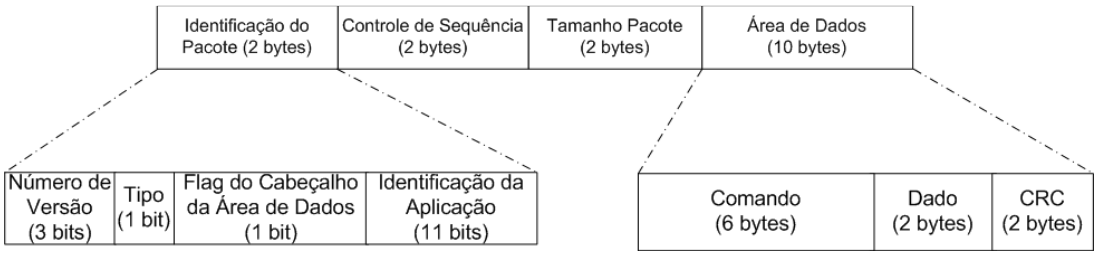

Figura 19 - Pacote de Telecomando.

A validação do Telecomando é realizada analisando todos os campos do cabeçalho do pacote recebido. No presente projeto do OBDH são verificado apenas o controle de seqüência do pacote e o CRC, ou seja, são descartados pacotes recebidos que não correspondem ao número de seqüência esperado pelo OBDH ou que o CRC do pacote esteja incorreto.

Para os fins do projeto do OBDH, foram implementados três Telecomandos que exigem uma Telemetria como resposta, são eles: “GET_GYRO” execução de leitura do Giroscópio e retorno de valores de rotação no eixo X em graus por segundo e o ângulo estimado em graus; “GET_ACEL” execução de leitura do acelerômetro e retorno de valores de aceleração nos eixos X e Y em unidades de gravidade (força G); “GET_INCL” execução de leitura do acelerômetro e retorno de valores de inclinação nos eixos X e Y em graus. Quando é identificado um desses Telecomandos a tarefa dispara uma Telemetria correspondente.

Para gerar as CLTUs que contêm os Telecomandos enviados para a UTMC está sendo utilizado um aplicativo desenvolvido pela equipe de Engenheiros do INPE. Esse aplicativo permite que seja carregado um arquivo de entrada com Telecomandos pré-definidos e a interface realiza a montagem da CLTU automaticamente. Após completar o processo, o aplicativo gera um arquivo com a CLTU contendo os Telecomandos a serem executados.

A Figura 20 ilustra o aplicativo fornecido pelo INPE para auxiliar no projeto do OBDH.

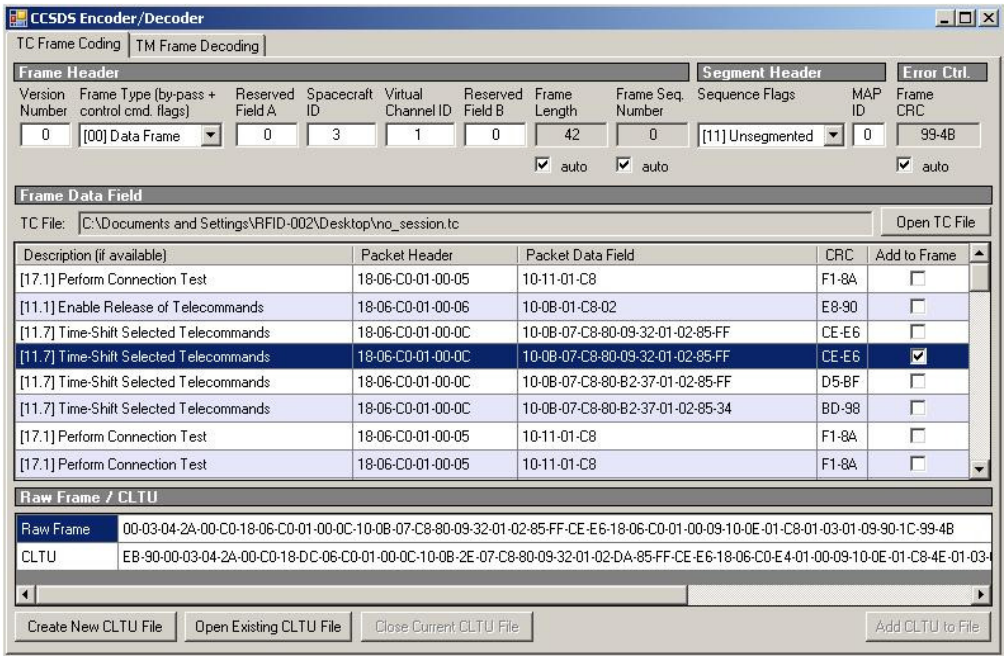

Figura 20 - Aplicativo que gera os Telecomandos para a UTMC.

Para utilizar o programa é necessário carregar um arquivo com Telecomandos pré-definidos. Esse arquivo é carregado quando pressionado o botão “Open TC File” localizado no lado direito superior da Figura 22. Depois de carregado o arquivo, os Telecomandos são apresentados na parte central da interface (*Frame Data Field*). Na parte inferior da interface são exibidas as informações de preenchimento de quadro e CLTU, de acordo com a seqüência de pacotes que estão sendo selecionados. O campo “*Raw frame”* exibe o formato do quadro encapsulado na CLTU. A parte superior da interface se refere a configuração dos quadros de Telecomandos gerados, ou seja, são campos que permitem parametrizar as informações a fim de atingir abranger todas as possíveis combinações de quadros possíveis. Por fim, o botão “*Create New CLTU File*” cria um arquivo de saída contendo a CLTU gerada pela aplicação.

5.2.2 Manipulação de Telemetrias

A tarefa responsável pela manipulação das Telemetrias tem a incumbência de empacotar e enviar as Telemetrias assim que um Telecomando for recebido. Isso porque em satélites a maioria dos Telecomandos enviados necessitam receber uma Telemetria como resposta para que a Estação Terrestre obtenha a confirmação que a ação solicitada foi executada.

Os campos da Telemetria estão ilustrados na Figura 21, que demonstra em detalhes o cabeçalho do pacote e área de dados específica do projeto do OBDH.

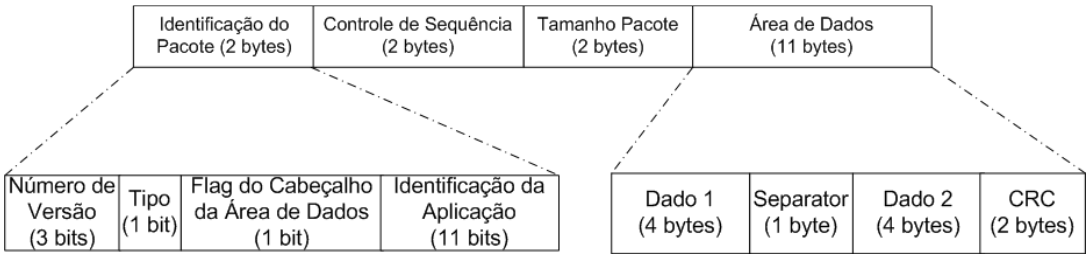

Figura 21 - Pacote de Telemetria.

Assim como nos Telecomandos, os pacotes de Telemetria também seguem um controle de seqüência. Esse controle é importante, pois possibilita que a Estação Terrestre possa observar quando houve perda de pacotes na transmissão. Além disso, nos pacotes de Telemetria existe também o CRC que permite a verificação da integridade das informações do pacote.

5.3 Software do Subsistema de Controle de Atitude e Órbita

O Software do Subsistema de Controle de Atitude e Órbita é responsável por realizar o controle dos instrumentos presentes no OBDH. Neste módulo são implementadas as tarefas que atuam diretamente sobre os sensores. Existem tarefas de aquisição e validação dos dados, tarefas que controlam a posição espacial e inercial do satélite. Além disso, este módulo é responsável por controlar os atuadores a fim de alterar a atitude e órbita do satélite.

Deste módulo apenas uma tarefa foi implementada no projeto do OBDH. Devido a complexibilidade do software foi implementado apenas o Gerenciamento dos Sensores de forma simplificada, que consiste em realizar a aquisição dos dados dos sensores e deixá-los disponíveis para as demais aplicações do OBDH. Esse módulo em um OBDH real controla uma quantidade maior de instrumentos, e a aquisição dos dados já são pré-processados para que o os módulos de controle de atitude e órbita apliquem os algoritmos de controle.

Na Figura 22 é ilustrada a arquitetura completa do Software de Controle de Atitude e Órbita, salientando a tarefa que foi desenvolvida no presente trabalho.

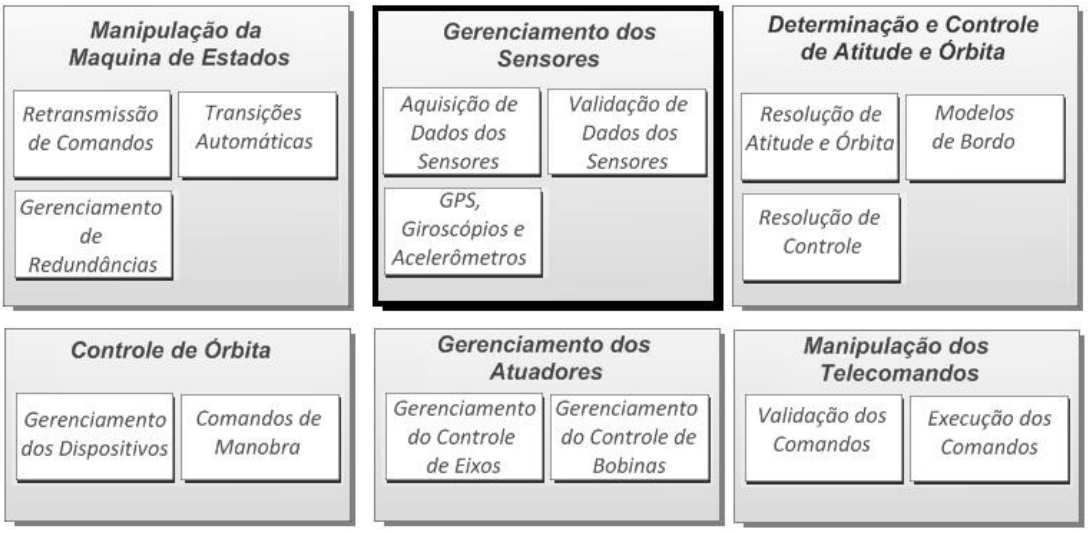

Figura 22 - Software do Subsistema de Controle de Atitude e Órbita.

Gerenciamento dos Sensores

Uma tarefa do sistema de tempo real foi desenvolvida para gerenciar os dados providos pelos sensores inerciais. Optou-se por uma tarefa exclusiva para executar essa função, pois os dados dos sensores são enviados pelo microcontrolador para o OBDH a cada segundo. Devido à temporização das tarefas do sistema de tempo real garante-se que nenhum dado será perdido. A tarefa "INS Task", como foi chamada, tem como única função receber os dados dos sensores via serial RS-232C e atualizar a estrutura responsável por armazená-las.

A execução da tarefa inicia configurando a interface serial responsável pela comunicação. Caso nenhum erro ocorra, a tarefa fica aguardando o recebimento de um dado pré-definido que indica o início de uma transmissão válida. Para o caso de nenhum dado ser recebido foi implementado um mecanismo de *timeout*, o que garante que a tarefa não fique ocupando o processador desnecessariamente. Caso o dado que indica início da transmissão seja recebida, a seqüência de dados é armazenada até que um dado pré-definido que indica o fim da transmissão seja recebido. Novamente, existe um contador de tempo para evitar que a tarefa fique esperando o dado que indica final de transmissão infinitamente. Caso os caracteres de início e fim da transmissão sejam válidos a tarefa verifica qual dos campos da estrutura devem ser atualizados e os atualiza. A Tabela 2 lista as mensagens válidas implementadas para comunicação com os sensores.

Tabela 2 - Mensagens de atualização dos sensores.

Mensagens	Descrição
$AAX<dado>AAY<dado>#	Novos dados de aceleração em X e em Y.
$AIX<dado>AIY<dado>#	Novos dados de inclinação em X e em Y.
$GYA<dado>GYG<dado>#	Novos dados de rotação no eixo X e ângulo estimado.

A partir das informações disponibilizadas pelos sensores seria possível realizar o controle inercial do satélite. Em trabalhos futuros é possível implementar os algoritmos que realizam o posicionamento do satélite agregando mais tarefas ao Software do Subsistema de Controle de Atitude e Órbita.

6. Detalhes da Implementação

Nesse capítulo são abordados os detalhes de implementação do sistema, os métodos adotados para seu desenvolvimento bem como as dificuldades encontradas. Nas seções a seguir é explicado como cada componente do sistema foi desenvolvido e testado.

6.1 Processador

Conforme colocado no Capítulo 4, o processador selecionado para o projeto foi o LEON3, da empresa Aeroflex Gaisler que fornece uma série de recursos para utilização desse processador. A Aeroflex Gaisler é uma empresa especializada em prover núcleos de propriedade intelectual (*Intelectual Property* - IP) e ferramentas para processadores embarcados baseados na arquitetura SPARC, tendo como produto principal o modelo do processador LEON3 sintetizável para uma grande variedade de FPGAs juntamente com a biblioteca de IPs (GRLIB) [19]. A biblioteca GRLIB consiste em uma variedade de IPs reutilizáveis portados para desenvolvimento em SoC, suportando uma grande variedade de ferramentas de CAD e FPGAs [45].

A preparação do ambiente de desenvolvimento de sistemas baseados no processador LEON3 inclui a detenção e instalação da biblioteca GRLIB a partir do site da Gaisler. O arquivo "grlib-gpl-1.0.20-b3403.tar.gz" deve ser extraído em uma pasta qualquer. Após, deve-se acessar grlib-gpl-1.0.20-b3403/designs/leon3-digilent-xup, que é a pasta com o projeto para a placa da Xilinx, utilizada nesse trabalho. Antes de realizar a síntese do VHDL é necessário configurar o projeto (*design*), para isso a GRLIB conta com uma ferramenta gráfica onde podem ser selecionadas as configurações desejadas, adicionados e removidos periféricos, configurada a quantidade de processadores entre outros. Essa ferramenta de configuração é basicamente um script que altera o arquivo “config.vhd”, responsável pela configuração do sistema como um todo, caso seja necessário o arquivo pode ser modificado manualmente. A Figura 23 ilustra a ferramenta gráfica de configuração.

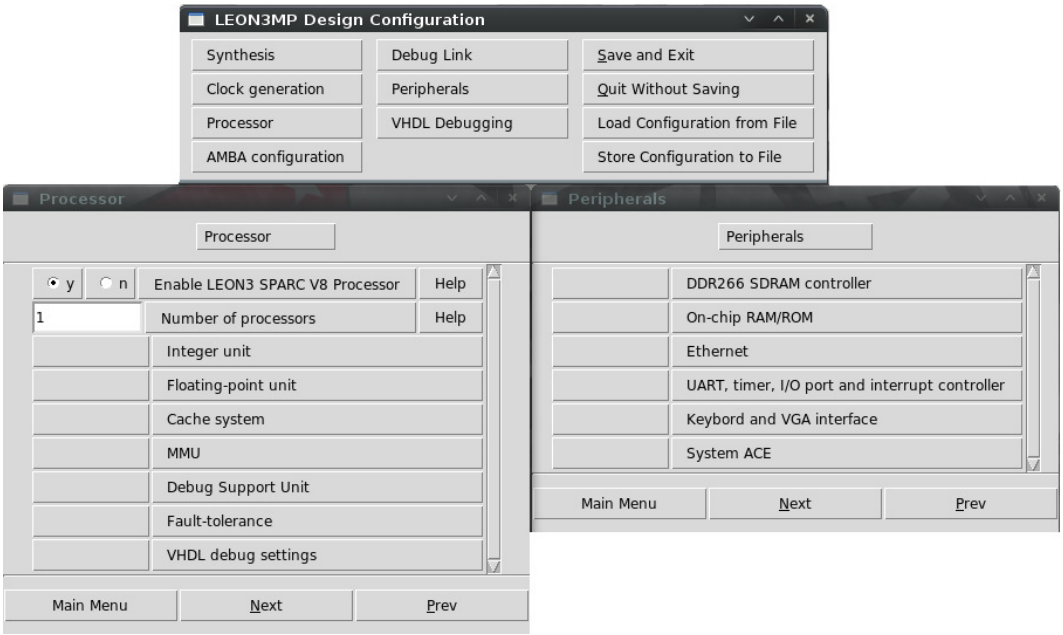

Figura 23 - Ferramenta Gráfica GRLIB.

Nas configurações padrão do projeto do LEON3 utilizado nesse trabalho estão disponíveis os seguintes periféricos: Controlador para SDRAM DDR266, Ethernet MAC, UART e Interface PS2. Como a placa XUP disponibiliza apenas um conector DB9, conector padrão para comunicação serial RS-232C, o design do processador para a placa de desenvolvimento não permite adicionar novas UARTs através da ferramenta de configuração ilustrada na Figura 23. Na arquitetura proposta, são necessárias três portas seriais (três UARTs): uma para possibilitar ao LEON3 o recebimento de informações dos sensores inerciais (placa com 8051); uma para envio de comandos para os instrumentos (painel de instrumentos); e outra para comunicação com a UTMC por onde o LEON3 irá receber Telecomandos e enviar Telemetrias. Dessa forma, foi necessário adicionar outras duas UARTs diretamente no código VHDL. O LEON3 conta com um barramento de periféricos on-chip, o AMBA-2.0 AHB/APB. Esse barramento é bastante utilizado, sendo o mesmo dos processadores ARM[45].

O código VHDL do processador LEON3 foi alterado adicionando-se as novas UARTs ao barramento AMBA-2.0 AHB/APB, em endereços livres e com posições próprias no vetor de interrupção. Os trechos do código VHDL alterado para que as novas UARTS funcionem corretamente estão no Anexo E. Os pinos de transmissão e recepção das UARTs adicionadas foram mapeados nos conectores de expansão da XUP. Como os pinos de expansão fornecem níveis de tensão TTL (5 volts) foi necessário utilizar um componente MAX232 [46] que converte os níveis de tensão de TTL para RS-232C, ou seja, de +12v para 0v e -12v para +5v. Assim é necessário utilizar dois MAX232, um para cada UART adicionada manualmente. Os *layouts* das placas utilizadas para os MAX232 estão no Anexo D.

6.2 Sistema Operacional

O RTEMS *Cross Compiler system* (RCC) é um sistema de desenvolvimento multiplataforma que gera, entre outros, executáveis para os processadores das famílias LEON e ERC32 com as diretivas de tempo real do sistema operacional RTEMS. Por se tratar de um compilador cruzado permite gerar códigos executáveis para o LEON3 utilizando outro sistema operacional, como por exemplo, o Linux.

Com um compilador próprio baseado no gcc, o RCC permite que códigos escritos em linguagem de programação C sejam compilados utilizando bibliotecas padrão para Linux e algumas bibliotecas exclusivas para o RTEMS. Os executáveis gerados pela ferramenta podem ser executados diretamente no processador LEON3, pois já incluem as diretivas do sistema operacional RTEMS, ou seja, a aplicação em linguagem C é compilada e ligada aos módulos do RTEMS gerando um código único em linguagem de máquina do processador, dispensando a instalação do sistema operacional. O processo de geração do código executável da aplicação, integrado ao RTEMS está ilustrado na Figura 24.

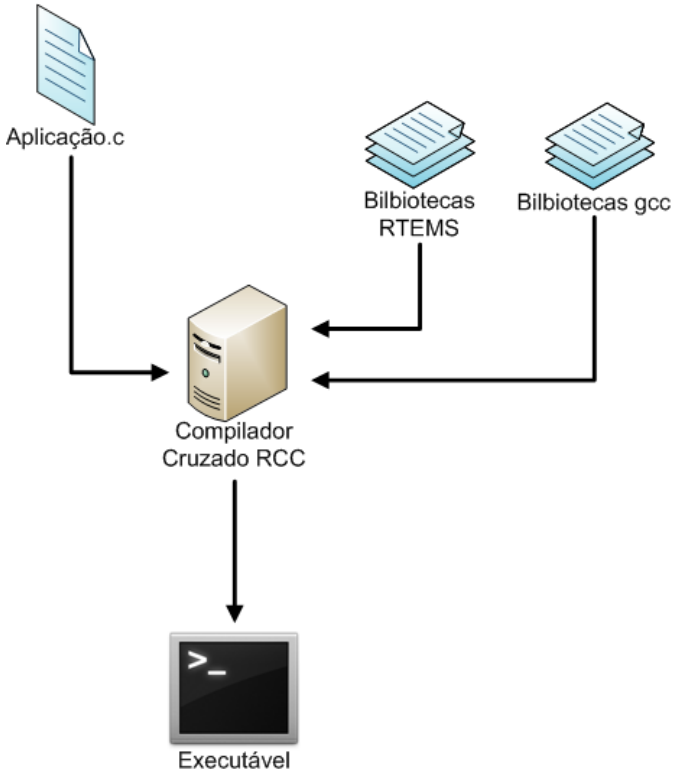

Figura 24 - Processo de geração do executável pelo RCC.

Por se tratar de um sistema operacional de tempo real o RTEMS exige que sejam definidas algumas configurações das tarefas executadas no sistema. Configurações como, por exemplo, número máximo de tarefas executadas, tamanho máximo de pilha das tarefas, número de descritores de arquivo gerenciados pela biblioteca de entrada e saída "libio" [47], entre outras.

O compilador cruzado RCC é distribuído de forma gratuita sob a licença GNU e pode ser obtido no site da Gaisler[48]. O RCC funciona em conjunto com a GRLIB facilitando o desenvolvimento deste trabalho. Os detalhes sobre a instalação e utilização do RCC no sistema operacional Linux estão no Anexo B.

6.3 Software OBDH

Nessa seção serão explicados os detalhes da implementação do software responsável por realizar as funcionalidades do OBDH. Como explicado nas seções anteriores, o software do OBDH foi desenvolvido em linguagem de programação C, utilizando as diretivas de tempo real do sistema operacional RTEMS. O compilador cruzado RCC foi utilizado para gerar códigos de máquina para o processador LEON3.

Por se tratar de um sistema operacional de tempo real todas as funções executadas pelo software necessitam ser executadas por tarefas, como explicado na seção 4.3. Cada tarefa possui características próprias que definem seu funcionamento. É necessário definir como as tarefas serão executadas pelo processador para manter as garantias de tempo real.

As tarefas desenvolvidas para implementação desse trabalho utilizam a função "*rtems_task_wake_after()*" no final de sua execução, assim é garantido que as tarefas são executadas até o fim antes de perder o processador. Cada tarefa é iniciada com declaração de variáveis locais e inicialização de interfaces de comunicação serial, caso nenhum erro ocorra a tarefa entra em um laço infinito onde suas funções principais são executadas. No final do laço a função "*rtems_task_wake_after()*" é utilizada, fazendo com que a tarefa entre em estado de espera pelo tempo em segundos indicado como parâmetro da função. Para calcular o tempo em que a tarefa permanece em estado de espera é utilizada a função "*get_ticks_per_second()*" que calcula o tempo em segundos baseado no número de ciclos de relógio do processador.

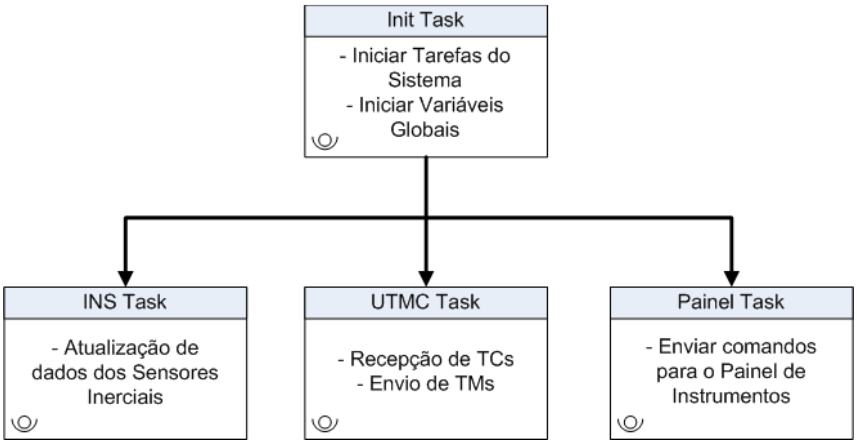

Figura 25 - Tarefas do Sistema.

Como ilustrado na Figura 25, as seguintes tarefas foram desenvolvidas para o software do OBDH:

- *Init Task*: tarefa padrão do RTEMS responsável pela declaração, criação e inicialização de todas as tarefas do sistema. Funções do RTEMS são utilizadas para definir parâmetros de cada tarefa, como por exemplo, o tamanho da pilha em memória disponível para a mesma. As estruturas globais utilizadas para armazenar os dados do programa são inicializadas nessa tarefa.
- *UTMC Task*: tem a função de receber os TCs, verificar a integridade do seu cabeçalho, em caso de um TC válido executar o comando indicado, montar o cabeçalho e a área de dados das TMs e as enviar.
- *INS Task*: responsável por receber os valores providos pelos sensores e atualizar a estrutura de dados responsável pelo armazenamento dos mesmos.
- *Painel Task*: verifica se um novo comando para o painel de instrumentos foi recebido, caso afirmativo envia o comando para o mesmo.

A tarefa UTMC Task é considerada a tarefa de maior prioridade do sistema, pois a perda de um TC recebido pode causar divergência entre o contador de seqüência de Telecomandos recebidos no OBDH e enviados na UTMC, fazendo com que os próximos TCs recebidos não sejam considerados válidos até que um pedido de retransmissão seja enviado pelo OBDH informando que um pacote de TC foi perdido. Para diminuir a probabilidade de ocorrência desse problema, a

tarefa UTMC Task é executada o dobro de vezes que as demais tarefas, para isso, o parâmetro utilizado na função "*rtems_task_wake_after()*", nessa tarefa, é a metade do utilizado nas outras tarefas. A Figura 26 ilustra a ocupação de processador pelas tarefas no tempo, não foram feitas medidas exatas de tempo das tarefas, portanto a Figura é meramente ilustrativa.

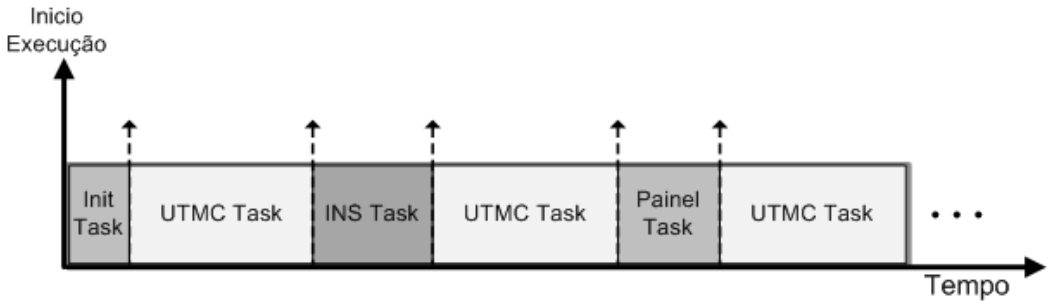

Figura 26 - Escalonamento de Tarefas.

6.4 Teste e Depuração do LEON3

Para efetuar os testes e depuração do software executado pelo LEON3 (SO e Aplicação) foi utilizada a ferramenta GRMON, distribuída de forma gratuita pela mesma empresa responsável pela GRLIB.

O GRMON é um monitor de depuração para os processadores da família LEON e para sistemas embarcados baseados na biblioteca GRLIB. A depuração do sistema embarcado no FPGA é possível devido à capacidade do GRMON de se comunicar com a unidade de suporte a depuração (*Debug Support Unit* - DSU) do LEON3, permitindo depuração não intrusiva do sistema.

Com essa ferramenta é possível adicionar executáveis criados com o RCC ao LEON3 e executá-los; visualizar as configurações do processador e os periféricos conectados ao barramento AMBA; permite conexão remota com o depurador GDB através das interfaces: Serial, Ethernet, JTAG, PCI e USB; tornando possível que o sistema seja depurado de maneira confiável.

Para os fins desse trabalho foi utilizado o cabo de testes JTAG para configurar o FPGA com o *bitstream* contendo a imagem do processador, para envio dos executáveis do sistema e depuração do mesmo. Os detalhes sobre a instalação e configuração do JTAG no sistema operacional Linux estão no Anexo C.

Quando a placa de desenvolvimento é ligada é necessário enviar o *bitstream* com a imagem previamente compilada do processador LEON3 ao FPGA, para isso foi utilizado o software Impact.

6.5 Periféricos

Nesta seção são demonstrados os procedimentos realizados no presente trabalho referentes aos componentes periféricos do OBDH. Mais especificamente, o procedimento de confecção de placas de circuito impresso, soldagem e codificação para o microcontrolador de forma a realizarem a comunicação com os sensores através do protocolo SPI.

6.5.1 Confecção de Hardware e Soldagem de Componentes

Como o acelerômetro e o giroscópio não haviam sido utilizados anteriormente, o processo de desenho e confecção das placas de circuito impresso foi realizado pelo grupo. Os componentes têm o encapsulamento do tipo *Land Grid Array* (LGA) o que significa que as conexões ficam na parte inferior do componente. Na Figura 27 é mostrado um componente LGA.

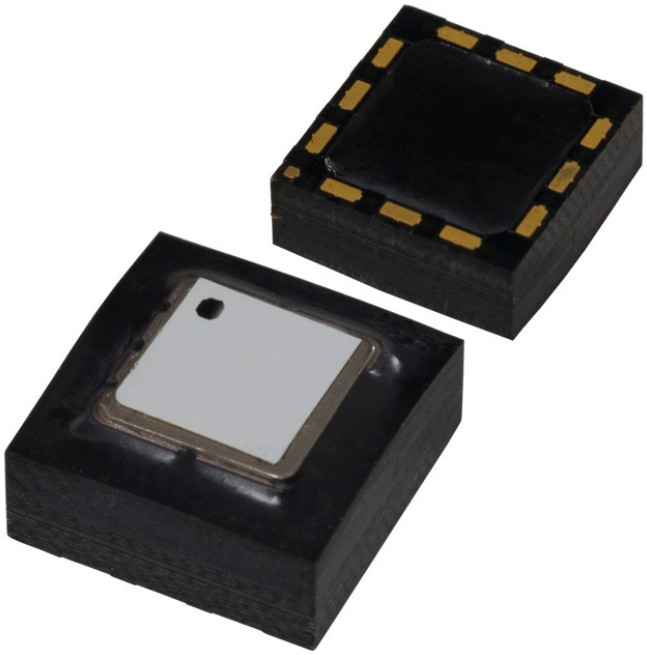

Figura 27 - Componente LGA.

Devido ao tipo de encapsulamento dos componentes foram feitas placas para disponibilizar as conexões de modo que estas fossem facilmente encaixadas em uma matriz de contatos. O processo de desenvolvimento das placas foi realizado como segue. Inicialmente foi feito um esquemático na ferramenta OrCAD da empresa Cadence [49]. A partir desse esquema elétrico, foi gerada uma *netlist* (lista de conexões de componentes elétricos) a qual serve de entrada para a ferramenta OrCAD Layout, também da empresa Cadence, que faz o roteamento das trilhas do circuito impresso. Finalmente a manufatura da placa é feita no Laboratório de Ensino e Pesquisa (LEP) da PUCRS em uma máquina de fresamento, a qual remove todo o cobre da placa deixando somente as trilhas e conexões desejadas.

Para realizar a soldagem dos componentes foi necessário um ajuste ao método tradicionalmente utilizado. Devido ao encapsulamento do componente, sua soldagem requer técnicas industriais, portanto, foi necessário um método diferenciado para soldagem dos componentes. Esse método consiste em modificar os *footprints* permitindo que seja possível externar os pinos do componente, para isso suas conexões foram alongadas. Tendo acesso a esses pinos maiores foi possível aplicar estanho nas conexões do componente e da placa. Ao aquecer as conexões da placa, a solda entre o contato do componente e da placa é

fundida, realizando a solda. A Figura 28 demonstra a placa desenvolvida com o componente soldado.

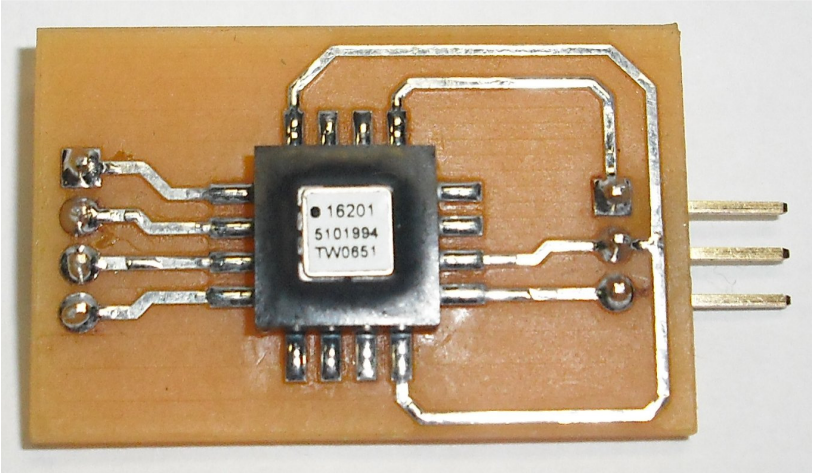

Figura 28 - Placa do Acelerômetro.

O responsável pela modificação do *footprint* do componente e pela idealização da técnica de soldagem foi o professor Dr. Júlio César Marques de Lima, responsável também pela soldagem dos componentes. Os *layouts* das placas desenvolvidas estão no Anexo D.

É importante ressaltar que as tensões de alimentação dos componentes são diferentes, o giroscópio utiliza 5 volts, a mesma tensão da placa com o microcontrolador, e o acelerômetro utiliza 3,3 volts, logo algumas adaptações tiveram de ser feitas para garantir os valores de tensão corretos para alimentação do acelerômetro.

O barramento de dados é comum aos dois componentes, para garantir que não ocorra problema com o acelerômetro, resistores de 1kohm foram adicionados ao barramento do mesmo. Para reduzir a tensão de alimentação de 5 volts para 3,3 volts foi utilizado um regulador de tensão e capacitores de filtro nas entradas e saídas do componente. Sem estes capacitores o acelerômetro não funcionou corretamente devido ao grande nível de ruído proveniente do regulador de tensão.

6.5.2 Comunicação com Acelerômetros e Giroscópios

Após ter as placas com o acelerômetro e giroscópio soldados, o próximo passo foi realizar a leitura dos registradores dos mesmos. O processo de leitura e escrita nos dois componentes é o mesmo, diferenciando apenas nos endereços dos registradores.

Para ler e escrever os dados dos componentes é utilizado o protocolo SPI, cada acesso ao componente é feito em um quadro de 16 bits de dados. As leituras dos registradores são executadas a partir do envio de dados no seguinte formato: o primeiro bit enviado informa se é leitura (nível lógico zero) ou escrita (nível lógico um), o segundo bit deve seguir em nível lógico zero e os seis bits que seguem, informam o endereço do registrador. Os últimos oito bits são de dados, no caso de uma escrita (byte a ser gravado no registrador previamente

endereçado) ou bits *don't care* no caso de uma leitura. Na Figura 29 é demonstrado o diagrama de tempo da comunicação com os componentes.

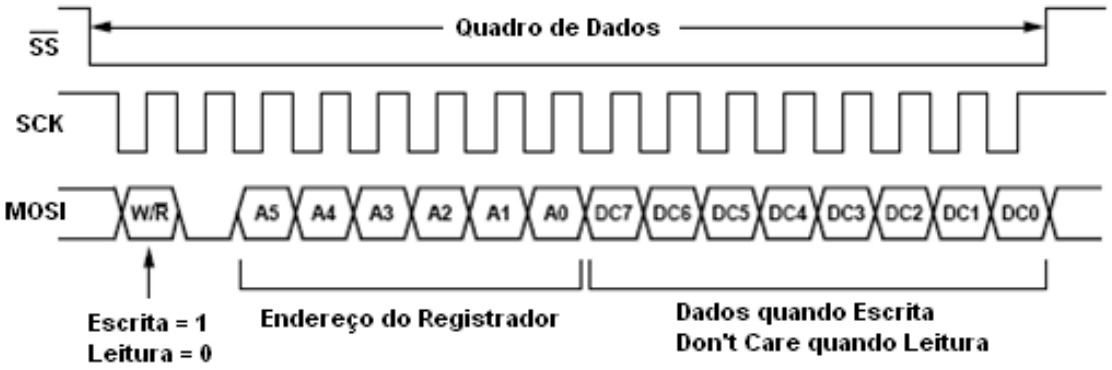

Figura 29 – Diagrama de Tempo para Leitura e Escrita no protocolo SPI.

Por se tratar de um protocolo serial, o dado do registrador endereçado é disponibilizado no próximo quadro de dados. A Figura 30 apresenta o diagrama do tempo da comunicação SPI com o componente.

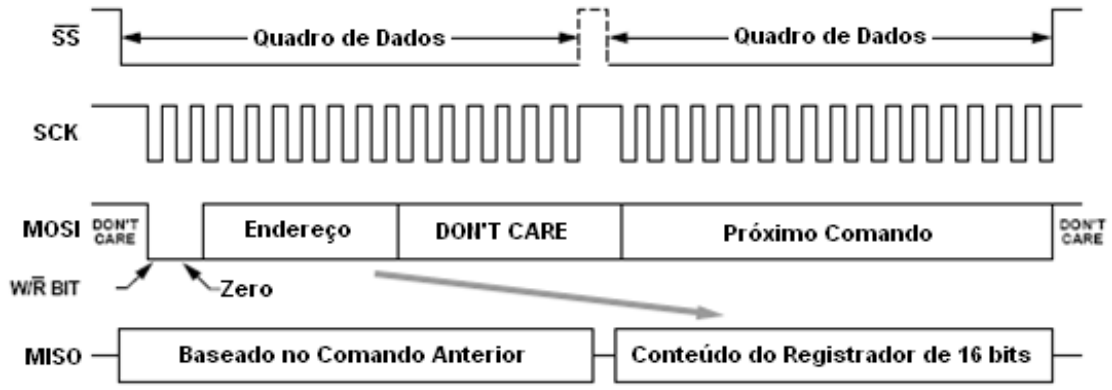

Figura 30 - Leitura de um registrador em SPI.

Os componentes possuem interface digital, assim os valores mensurados pelos componentes são armazenados em um conjunto de registradores. Esses registradores informam além da aceleração e rotação, também a inclinação, temperatura, tensão de alimentação, entre outros.

Os registradores disponíveis no acelerômetro modelo ADIS16201 são demonstrados na Tabela 3. O campo Formato do Dado da Tabela 3, deve ser destacado pois é importante na concepção do código de leitura e escrita nos componentes. O campo indica se o dado é representado em valores sem sinal (Binário sem sinal) ou em valores com sinal (Binário em Complemento de 2), para os casos em que podem existir medições negativas.

Tabela 3 - Registradores do Acelerômetro ADIS16201.

Nome	Descrição	Endereço	Tamanho	Formato do Dado	Escala (LSB)
SUPPLY_OUT	Tensão de Alimentação	0x03,	12 bits	Binário sem sinal	1,8315 mV
XACCL_OUT	Aceleração no eixo X	0x05,	14 bits	Complemento de	0,4625 mg
YACCL_OUT	Aceleração no eixo Y	0x07,	14 bits	Complemento de	0,4625 mg
AUX_DAC	Tensão Analógica Auxiliar	0x09,	12 bits	Binário sem sinal	0,61 mV
TEMP_OUT	Sensor de Temperatura	0x0B,	12 bits	Binário sem sinal	-0,47°C
XINCL_OUT	Inclinação no eixo X	0x0D,	12 bits	Complemento de	0,1°
YINCL_OUT	Inclinação no eixo Y	0x0F, 0x0E	12 bits	Complemento de	0,1°

Analogamente ao acelerômetro, o giroscópio ADIS16250 possui sua própria gama de registradores demonstrado na Tabela 4.

Tabela 4 - Registradores do Giroscópio ADIS16250.

Nome	Descrição	Endereço	Tamanho	Formato do Dado	Escala (LSB)
ENDURANCE	Contador da Memória	0x01,	16 bits	Binário	1 count
SUPPLY_OUT	Tensão de Alimentação	0x03,	12 bits	Binário	1,8315 mV
GYRO_OUT	Dado do Giroscópio	0x05,	14 bits	Complemento de	0,07326°/sec
AUX_DAC	Tensão Analógica Auxiliar	0x0B,	12 bits	Binário	0,6105 mV
TEMP_OUT	Sensor de Temperatura	0x0D,	12 bits	Complemento de	0,1453°C
ANGL_OUT	Ângulo de Saída	0x0F, 0x0E	14 bits	Binário	0,03663°

6.5.3 Placa de Sensor Inercial

A partir das informações do protocolo foi gerado o código correspondente para o microcontrolador 8051 na linguagem de programação C, Na geração do código de máquina que é carregado no microcontrolador. O compilador SDCC (*Small Device C Compiler*) foi utilizado. Este código realiza a leitura dos seguintes registradores no acelerômetro e giroscópio listados na Tabela 3 e Tabela 4: XACCL_OUT (dados de aceleração no eixo X), YACCL_OUT (dados de aceleração no eixo Y), XINCL_OUT (dados de inclinação no eixo X), YINCL_OUT (dados de inclinação no eixo Y), GYRO_OUT (dados de rotação no eixo X) e ANGL_OUT (ângulo estimado). Assim, os dados providos pelos componentes são enviados via RS-232C ao OBDH e os valores dos registradores são novamente lidos e enviados ao OBDH em um laço infinito. O código completo em linguagem de programação C está disponível no CD com os demais arquivos do trabalho de conclusão. A Figura 31 demonstra a placa em que os sensores estão montados e a placa com o microcontrolador 8051 responsável pelo envio dos dados.

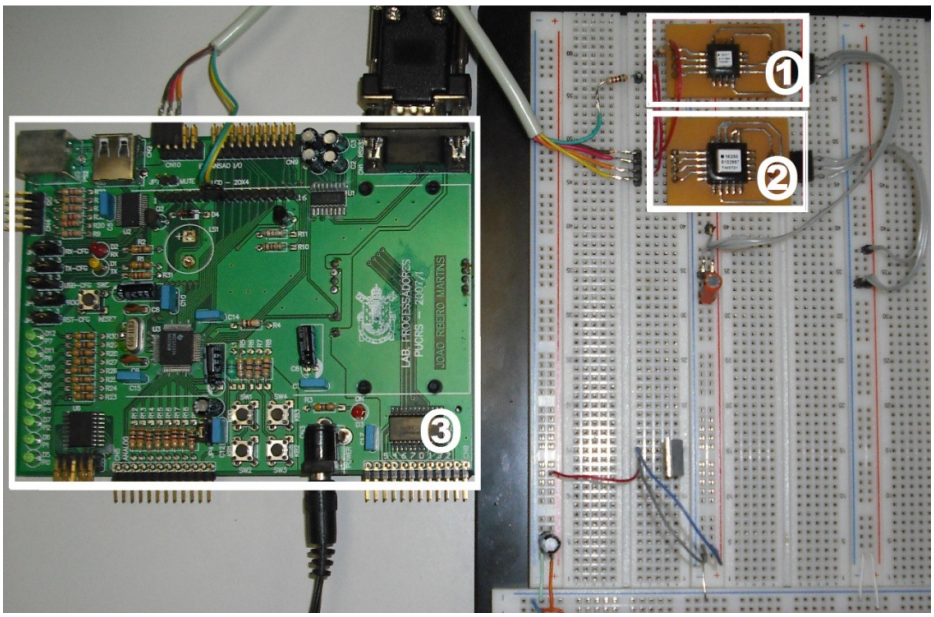

Figura 31 - Placa de Sensor Inercial.

A direita da Figura 31 está a protoboard contendo os sensores inerciais montados na mesma. No item número 1 da figura se encontra o acelerômetro, no item 2 está o giroscópio, estes dois componentes conectados a placa com o microcontrolador 8051 (indicada pelo item 3 da figura) através de um barramento de dados comum. Na parte superior ao centro da figura está o conector RS-232C conectado na placa com o microcontrolador.

6.5.4 Painel de Instrumentos

O painel de instrumentos é uma expressão visual ou sonora para os TCs recebidos pelo OBDH, o código consiste em um laço infinito que recebe dados pela porta serial. Ao ser recebido um caracter '$' é iniciado o processo de interpretação da leitura. Caso o segundo caracter seja um 'G' o buzzer é ativado. O tempo de ativação é o terceiro caracter recebido, podendo ser ligado de 1 (0x1) segundo até 15 segundos (0xF).

Caso o segundo caracter não for um 'G', os dois caracteres seguidos depois da inicialização representam o byte a ser escrito nos oito LEDs disponíveis na placa, sendo a seqüência de caracteres '00' significando nenhum LED aceso (0x00) e 'FF' todos os LEDs acesos (0xFF). O código completo em linguagem de programação C está disponível no CD com os demais arquivos do trabalho de conclusão. A Figura 32 demonstra a placa com o microcontrolador 8051 responsável por representar o painel de instrumentos.

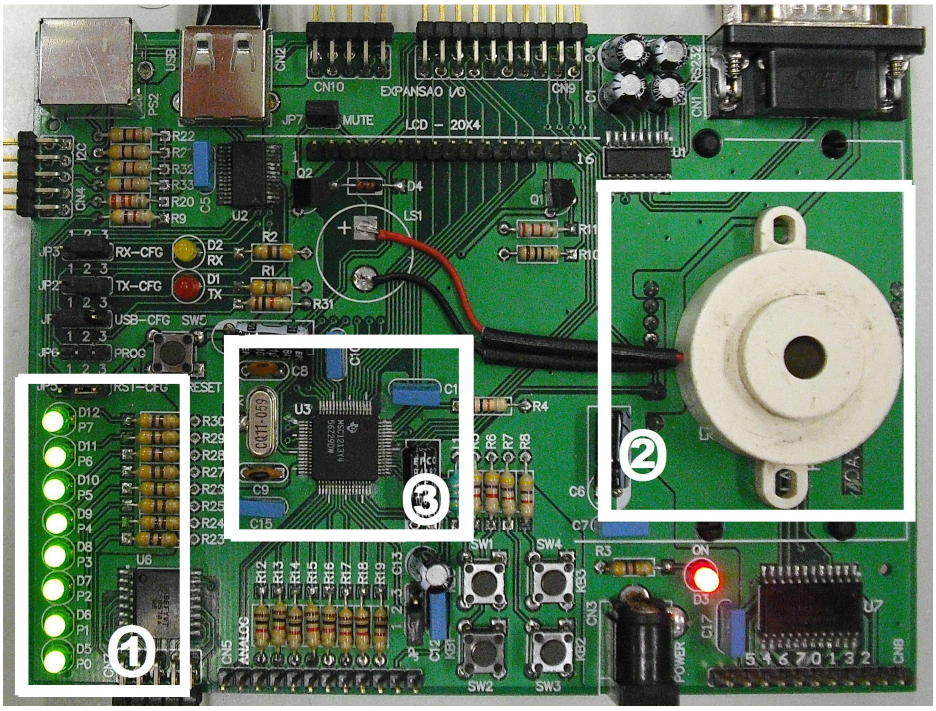

Figura 32 - Painel de Instrumentos.

No painel de instrumentos, os 8 LEDs são demonstrados no item número 1 da Figura 32, no item 2 da figura é mostrado o buzzer, o microcontrolador 8051 (comum as duas placas com esse controlador) é apresentado no item 3 da figura. No canto superior direito da figura é possível ver o conector da porta serial RS-232C.

7. Resultados Obtidos

Este capítulo apresenta os resultados obtidos na implementação desse trabalho. Serão demonstrados os métodos utilizados para os testes parciais de cada componente do sistema, e os resultados obtidos com os mesmos. As seções desse capítulo estão divididas cronologicamente, ou seja, na mesma seqüência em que os testes foram efetuados. As seções foram divididas em: Processador, que contém os detalhes de síntese do LEON3 e os acessos iniciais ao processador; Sistema Operacional, que demonstra os primeiros programas compilados e executados no sistema; Periféricos, que aborda a maneira como as placas dos Sensores e do Painel de Instrumentos foram validadas; OBDH e Instrumentos, que demonstra como o software do OBDH foi validado juntamente com os Periféricos; e Sistema Completo, com a validação final do sistema utilizando a UTMC e o ESE.

7.1 Processador

A síntese do processador LEON3 foi efetuada utilizando os scripts disponibilizados pela GRLIB[21], facilitando assim o processo de síntese do processador. Foi necessário passar como parâmetro do script a ferramenta a ser utilizada pelo mesmo, para os fins desse trabalho foram utilizadas as ferramentas da Xilinx. As configurações efetuadas antes da síntese foram abordadas na seção 6.1. A Tabela 5 demonstra os resultados de utilização de FPGA obtidos na síntese do processador.

Tabela 5 - Utilização do FPGA.

Componente	Utilização	Porcentagem
BSCAN	1 de 1 disponível	100%
BUFGMUX	11 de 16 disponíveis	68%
DCM	4 de 8 disponíveis	50%
External IOB	207 de 556 disponíveis	37%
MULT18X18	1 de 136 disponíveis	1%
RAMB16	34 de 136 disponíveis	25%
SLICES	13694 de 13696 disponíveis	99%

É importante ressaltar que esses resultados são referentes a síntese do processador LEON3 modificado para os fins desse trabalho, ou seja, com as UARTs adicionais para comunicação com os instrumentos. Apesar de a imagem gerada ocupar grande parte do FPGA não é necessário que nenhuma alteração seja efetuada para redução de área ocupada, pois essa imagem contém o processador e todos os periféricos necessários para realização desse trabalho. A imagem gerada pela síntese foi carregada no FPGA utilizando a ferramenta Impact, como explicado na seção 6.4.

Para verificar o funcionamento do processador embarcado no FPGA foi utilizado o software GRMON, abordado no Capítulo 6, seção 6.4. O GRMON permite verificar algumas configurações do processador e os periféricos conectados ao barramento AMBA-2.0 AHB/APB. Utilizando o cabo de testes JTAG, o GRMON se conecta com a unidade de depuração do processador (DSU) permitindo acesso não intrusivo ao processador. A Figura 33 demonstra o funcionamento inicial do GRMON, com as informações sobre o JTAG, conexão com a DSU e os principais módulos componentes encontrados no barramento. Assim é possível verificar que o processador embarcado no FPGA está funcionando corretamente.

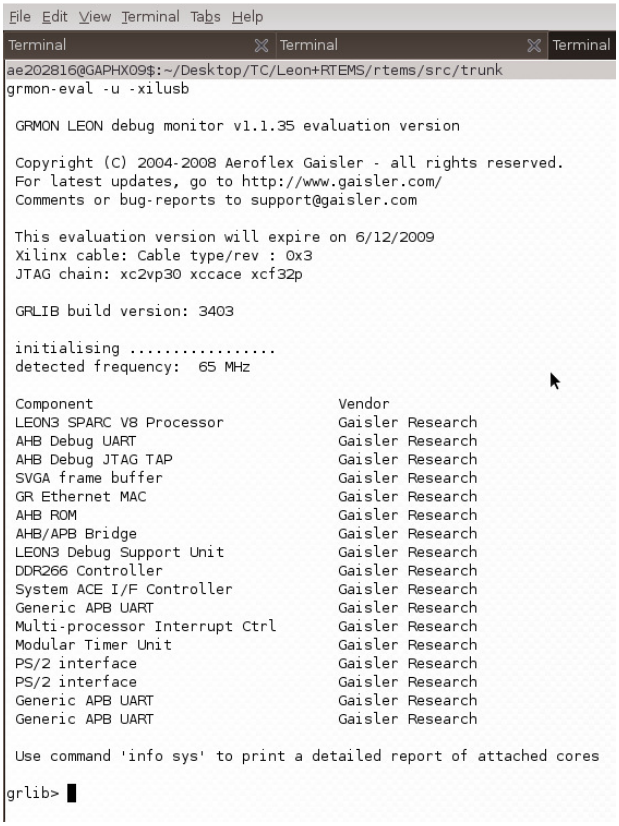

Figura 33 - Inicialização do GRMON.

O GRMON disponibiliza alguns comandos para verificar informações sobre o processador, entre eles, o de maior relevância para os fins desse trabalho é o “info sys”. Esse comando retorna informações detalhadas sobre os módulos conectados ao processador como, por exemplo: endereços no barramento de dados, endereços no vetor de interrupção, relógios utilizados pelo componente, buffers, entre outros. A Figura 34 demonstra o retorno da função “info sys”. Vale ressaltar que as UARTs adicionadas para os fins desse trabalho aparecem na lista de dispositivos, com os endereços de barramento previamente definidos pelo grupo.

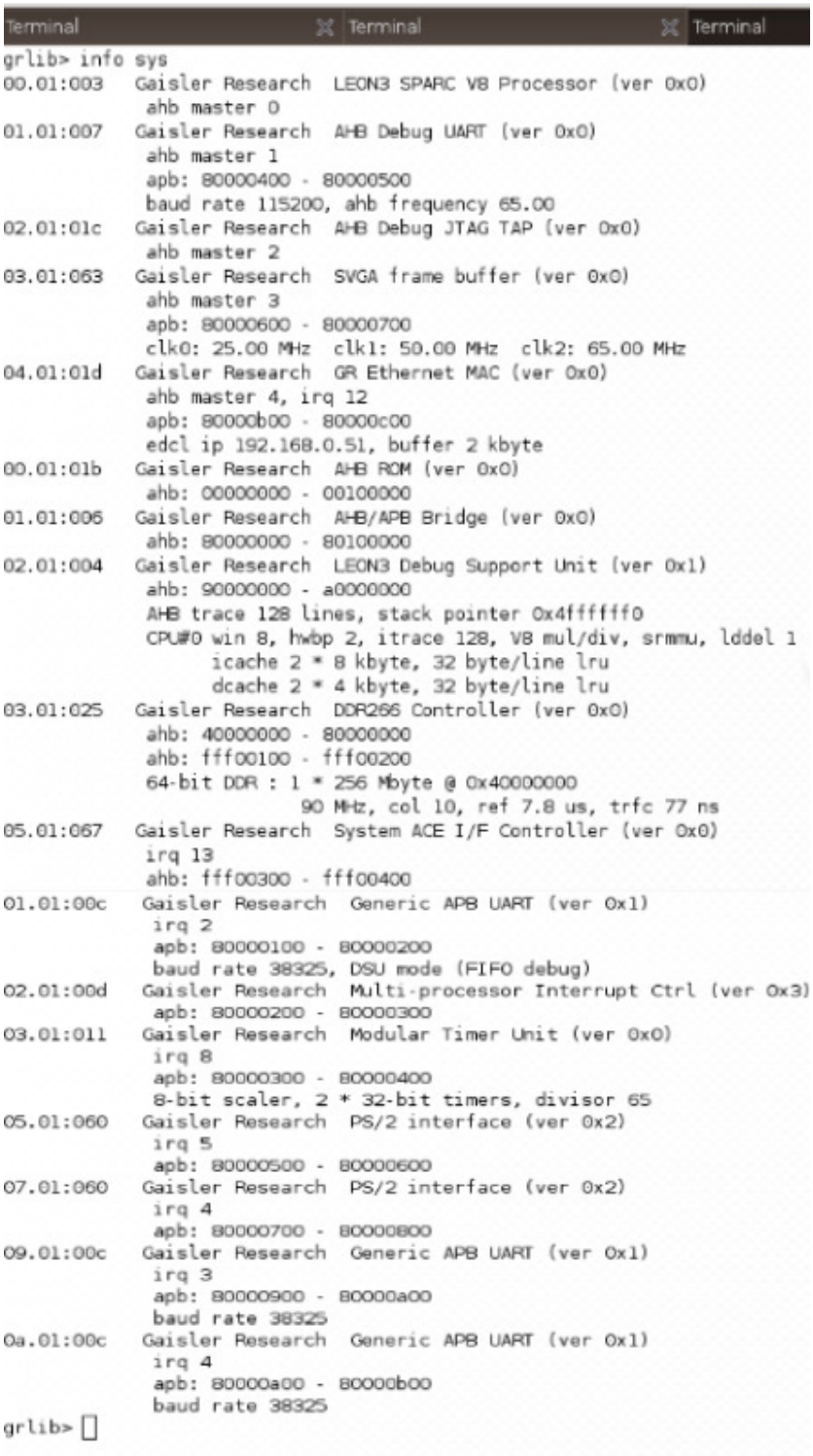

Figura 34 - Informações sobre o Sistema.

7.2 Sistema Operacional

O sistema operacional RTEMS, utilizado nesse trabalho, não necessita ser instalado, ou seja, é basicamente um conjunto de bibliotecas utilizadas na compilação do código fonte, maiores detalhes sobre o RTEMS e sua utilização podem ser encontrados nas seções 4.3 e 6.2.

Como demonstrado na seção 4.3 os códigos fonte desenvolvidos nesse trabalho são compilados com o RCC[48], compilador cruzado que utiliza as bibliotecas do sistema operacional RTEMS para gerar os códigos de máquina para o processador LEON3. Antes de iniciar o desenvolvimento do código fonte responsável por realizar as funções de um OBDH, foram executados códigos exemplo disponibilizados pela Aeroflex Gaisler para garantir o funcionamento do processador LEON3 e do sistema operacional embarcados no FPGA.

A utilização dos códigos exemplo disponíveis com o RCC facilitou o processo de aprendizagem na utilização do compilador cruzado e o inicio do desenvolvimento dos códigos fonte do OBDH. Entre os códigos exemplo utilizados vale ressaltar o "rtems-tasks.c" que demonstra o processo de inicialização e execução de duas tarefas de tempo real que alteram-se na utilização do processador.

Com o correto funcionamento dos programas exemplo testados, e as respostas aos comandos do GRMON, o grupo considerou que o processador e o sistema operacional estão prontos para inicialização do desenvolvimento dos códigos do OBDH.

7.3 Periféricos

Os testes de validação dos sensores inerciais foram aplicados em duas etapas, primeiramente, para garantir que não havia erros no processo de comunicação pelo protocolo SPI, o código gerado foi verificado observando a saída do barramento de dados (pinos SCK e MISO) no osciloscópio, os pontos principais a serem observados são a freqüência do relógio e o dado. Na Figura 35 é apresentada uma leitura do osciloscópio do registrador 0x0E na freqüência de 24,57kHz.

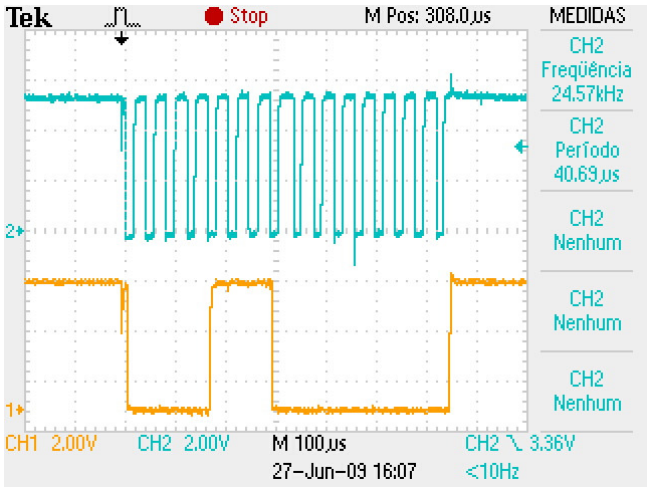

Figura 35 - Leitura da Saída SPI Gerada pelo Microcontrolador.

Como especificado no manual do fabricante dos componentes (acelerômetro e giroscópio), a freqüência de operação mínima permitida para o relógio responsável por sincronizar a comunicação é de 0,01MHz podendo variar até 2,5MHz. O dado a ser enviado deve estar estável quando o relógio está em nível lógico um (transição do dado ocorre no nível lógico zero), as temporizações da comunicação foram realizadas de acordo com essas informações e validadas através de um osciloscópio.

Garantindo que as temporizações da comunicação estão corretas, os sensores inerciais foram conectados a placa com o microcontrolador. Utilizando o código previamente gerado, foram realizadas leituras nos registradores de inclinação do acelerômetro. As leituras foram realizadas diversas vezes e os dados recebidos mostrados na tela de um computador. Para verificar a funcionalidade do componente, a protoboard onde os sensores se encontravam foi inclinada nos dois eixos de sensibilidade do acelerômetro, assim foi possível observar as alterações nos valores recebidos. As informações de aceleração e inclinação podem ser verificadas corretamente, devido a maneira como as grandezas, por eles medidas, são representadas. Porém, alguns dados de sensores necessitam de um ambiente próprio para validação, como por exemplo, dados de rotação, pois para se ter um valor significativo é necessário rotacionar o componente a uma velocidade constante, procedimento que se torna difícil devido aos diversos fios conectados as placas. Sendo assim os valores provenientes do giroscópio não foram devidamente validados, pois esse não era o objetivo principal desse trabalho.

Para o Painel de Instrumentos, o procedimento é mais trivial. Foi gerada uma rotina para o microcontrolador com a função de receber dados pela interface serial e executar um comando pré-definido, que pode ser acender os LEDs ou ativar o buzzer. Para fazer a verificação desse código, utilizando um PC, foram enviados ao microcontrolador caracteres pré-definidos, como explicado na seção 6.5.4. Assim foi possível verificar visualmente o funcionamento do Painel.

7.4 OBDH e Periféricos

Tendo os módulos do sistema validados separadamente, como explicado nas seções anteriores, foram realizadas as conexões entre os módulos. Os módulos se comunicam através do protocolo serial RS-232C, como explicado na seção 5.1. Para realizar a comunicação foi necessário utilizar dois componentes MAX232, isso se deve ao fato de os pinos utilizados para a comunicação no FPGA serem do tipo TTL, que utiliza níveis de tensão diferentes dos utilizados na comunicação serial RS-232C, uma explicação detalhada sobre a utilização dos componentes pode ser encontrada na seção 6.1.

Em um primeiro momento foram desenvolvidas rotinas simples de leitura e escrita para validação da comunicação serial com os periféricos. Essas rotinas foram utilizadas como base para o desenvolvimento das utilizadas na versão final deste trabalho. Assim foi possível validar as UARTs adicionadas ao VHDL do processador, as placas desenvolvidas para utilização dos componentes MAX232 e as rotinas de comunicação tanto nos microcontroladores como no processador LEON3.

Para que fosse possível depurar o sistema sem a utilização da UTMC foi necessário utilizar um PC fazendo o papel do Subsistema de Comunicação. Com esse intuído foi desenvolvido um programa em linguagem de programação C, que faz o papel da UTMC. O programa de teste gera todos os Telecomandos implementados nesse trabalho e recebe as Telemetrias correspondentes, para que seja possível verificar erros no funcionamento do OBDH. Os Telecomandos e as Telemetrias definidos para esse trabalho são demonstrados nas seções 5.2.1 e 5.2.2 respectivamente.

A Figura 36 ilustra a utilização do software de teste na seguinte situação: O software, fazendo o papel do Subsistema de Comunicação, enviou um Telecomando para o OBDH requisitando os dados de inclinação provenientes do acelerômetro; O OBDH respondeu ao Telecomando com uma Telemetria informando os dados atualizados do sensor; A Figura 36 demonstra a seleção do Telecomando a ser enviado (3, GET_INC), e a Telemetria recebida como resposta (CABEÇALHO, DADO e CRC).

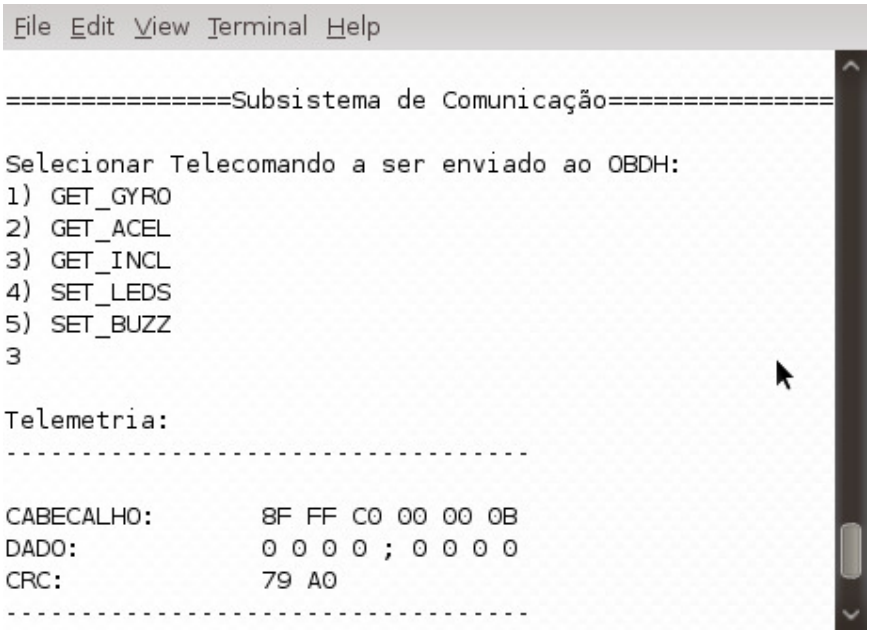

Figura 36 - Software de Teste.

A Figura 37 ilustra o recebimento do Telecomando no OBDH. Demonstrando o Telecomando recebido (Telecomando na primeira linha do terminal), uma breve descrição de sua função (“Telecomando Recebido”) e os campos de controle de erro verificados (Contador de Seqüência e CRC do Pacote) pelo OBDH.

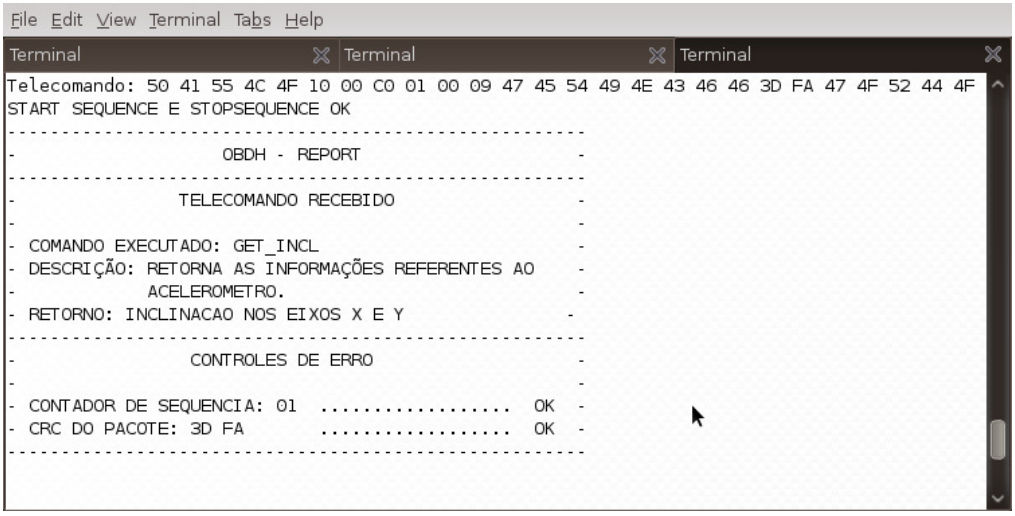

Figura 37 - Software de Teste no LEON3.

7.5 Sistema Completo

Com todas as etapas de teste explicadas nas seções anteriores devidamente concluídas, o sistema foi considerado estável para iniciar a integração com o Subsistema de Comunicação real. Para efetuar a comunicação entre o OBDH e a UTMC foi necessário adicionar uma UART a UTMC, tornando a comunicação serial síncrona utilizada, em serial assíncrona RS-232C, os detalhes sobre as modificações na UTMC estão na seção 4.6.6.

A comunicação entre o OBDH e a UTMC foi considerada válida no momento em que os pacotes de Telecomando e Telemetria foram reconhecidos pelo OBDH e pelo ESE respectivamente. Os Telecomandos utilizados foram criados com auxilio da ferramenta de geração de Telecomandos disponibilizada pelo INPE, como explicado na seção 5.2.1.

Assim, foi possível efetuar o envio dos Telecomandos previamente definidos a partir do ESE e verificar as Telemetrias recebidas no mesmo. Foi possível comprovar visualmente o funcionamento dos Telecomandos devido à utilização do Painel de Instrumentos, ou seja, foram enviados Telecomandos para ligar os LEDs e o buzzer do painel, e verificado se os itens do painel foram ativados corretamente. Os Telecomandos de aquisição de dados dos sensores foram validados através do ESE, ou seja, as Telemetrias com os dados dos sensores são exibidas na tela do ESE sendo possível verificar os dados providos pelos sensores na área de dados do pacote de Telemetria. Todos os Telecomandos criados para esse trabalho foram enviados diversas vezes, garantindo o funcionamento proposto.

A Figura 38 demonstra uma das telas do Labview (software utilizado no ESE) com o sistema completo em funcionamento. No primeiro campo de dados é demonstrado o Telecomando enviado pelo ESE, contendo os campos de cabeçalho e controle utilizados pela UTMC e a área de dados e cabeçalho que será enviada para o OBDH. No segundo campo é demonstrada a Telemetria recebida, que novamente passou pela UTMC e teve seus campos de cabeçalho e controle adicionados antes de retornar ao ESE.

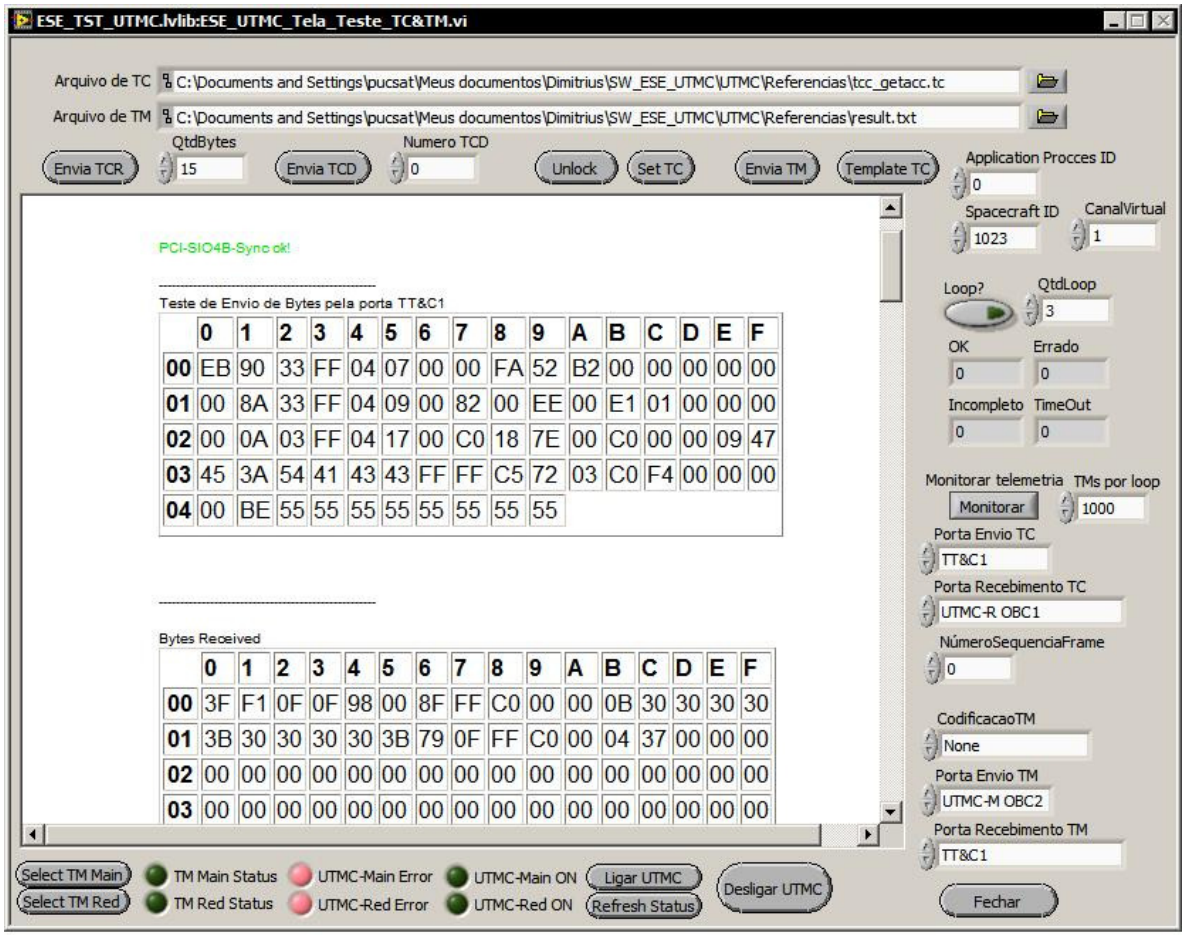

Figura 38 - TC e TM completos.

O campo de dados da Telemetria tem maior relevância para depuração do OBDH, pois nele estão contidos os dados providos pelos sensores. Na Figura 39 está ilustrado o campo de dados da Telemetria que contém as informações de inclinação requeridas pelo Telecomando.

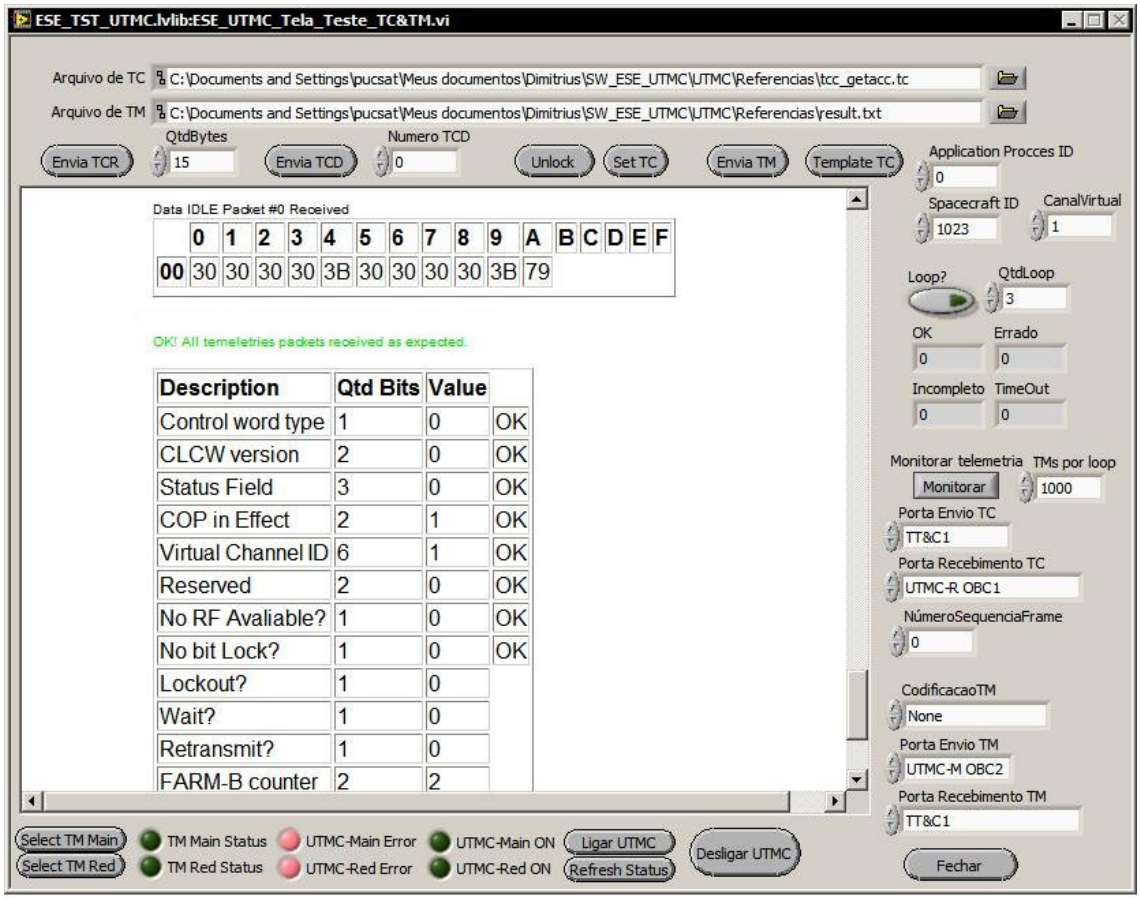

Figura 39 - Área de Dados da TM.

8. Conclusão e Trabalhos Futuros

No presente trabalho foram desenvolvidas as funcionalidades básicas de um OBDH para veículos espaciais. O projeto agrega ao PUC#SAT uma nova linha de pesquisas a fim de suprir a demanda do INPE para a PMM. Tendo em vista uma implementação mais próxima de aplicações reais, sensores inerciais e um painel de instrumentos foram adicionados ao projeto. O trabalho teve a duração de 5 meses, onde o grupo atingiu as metas inicialmente colocadas na Proposta de Trabalho de Conclusão, com exceção do algoritmo de navegação inercial que acabou por se tornar um tema para trabalhos futuros devido a sua complexidade. No decorrer desse período as tarefas, para cada um dos membros do grupo, foram realizadas em paralelo conforme cronograma proposto.

Além do conhecimento técnico, dado o espaço de tempo disponível para a pesquisa e desenvolvimento, o grupo pode afirmar que a quantidade de trabalho foi suficientemente grande, e exigiu dedicação e disciplina dos componentes do grupo. Isso é importante para o desenvolvimento da equipe no âmbito profissional como engenheiros. A organização das idéias, escrita do trabalho e a capacidade de trabalho em grupo podem ser citadas como as principais lições aprendidas no decorrer do trabalho motivando o grupo a continuar as atividades em um potencial mestrado na área espacial.

Destaca-se como uma das maiores dificuldades do trabalho a escrita do texto descritivo, visto que durante o curso de Engenharia o hábito de escrita não é muito exercitado. O grupo apresentou dificuldades ao implementar o processo para comunicação com os sensores inerciais, a integração com a UTMC e a manipulação das tarefas utilizando o sistema operacional de tempo real. Além disso, foi necessária muita pesquisa para acrescentar os periféricos de comunicação no LEON3, visto que o material disponível sobre o processador é extremamente escasso.

O presente trabalho tem como característica a multidisciplinaridade, requerendo do grupo a aplicação de Algoritmos, Programação, Arquitetura de Computadores, Sistemas Operacionais, Sistemas de Tempo Real, Programação de Periféricos, Processadores, Teste e Confiabilidade de Sistemas e Projeto de Sistemas Integrados, sendo estes assuntos abordados no curso de Engenharia de Computação.

Como trabalhos futuros podemos citar o algoritmo de navegação inercial, a aplicação de técnicas de confiabilidade de dados e a continuação da implementação do OBDH, completando os módulos que não foram contemplados no presente trabalho.

Referências Bibliográficas

[1] David S. F. Portree; Joseph P. Loftus, Jr. "**Orbital Debris: A Chronology"**. Lyndon B. Johnson Space Center. Extraído de http://ston.jsc.nasa.gov/collections/TRS/_techrep/TP-1999-208856.pdf em 31 de março de 2009.

[2] Instituto Nacional de Pesquisas Espaciais. **"INPE"**. Extraído de http://www.inpe.br/ em 31 de março de 2009.

[3] Bezerra, E. A.; Almeida, G. M., "**PUC#SAT - Interface de Telecomando e Telemetria CCSDS Visando Lógica Reconfigurável**", PROGRAMA UNIESPAÇO, II SEMINÁRIO Programa UNIESPAÇO - 2004/2006, São José dos Campos, SP, INPE, 24 Nov. 2006, pp. 54-60.

[4] Agência Espacial Brasileira, **"AEB"**. Extraído de http://www.aeb.gov.br em 1 de abril de 2009.

[5] CCSDS, **"TC Space Data Link Protocol. Blue Book. Issue 1"**.Washington, DC, USA, September 2003, CCSDS Recommended Standard.

[6] CCSDS, **"Communications Operation Procedure-1. Blue Book. Issue 1"**. Washington, DC, USA, September 2003, CCSDS Recommended Standard.

[7] CCSDS, **"Space Packet Protocol. Blue Book. Issue 1"**. Washington, DC, USA, September 2003, CCSDS Recommended Standard.

[8] CCSDS, **"Telecommand Summary of Concept and Rationale. Green Book. Issue 6"**. Washington, DC, USA, January 1987, CCSDS Recommended Standard.

[9] INPE, **"Multi-mission Platform Attitude Control and Data Handling (ACDH) Subsystem Specification"**, Documento de Especificação A822700-SPC-01/06, INPE, São José dos Campos, 129pp., Agosto 2001.

[10] INPE, **"Brasil e Reino Unido anunciam projeto de cooperação espacial"**. Extraído de http://www.inpe.br/noticias/noticia.php?Cod_Noticia=1513 em 4 de abril de 2009.

[11] EUROPEAN COOPERATION FOR SPACE STANDARDIZATION (ECSS), **"Ground systems and operations - Telemetry and telecommand packet utilization"**. Extraído de http://gtb.iasfbo.inaf.it/public/gtb_manuals/ECSS-E-70-41A(30Jan2003).pdf em 31 de março de 2009.

[12] CCSDS, **"Telemetry Summary of Concept and Rationale. Green Book. Issue 1"**. Washington, DC, USA, December 1987, CCSDS Recommended Standard.

[13] Gabriel Marchesan Almeida, **"Códigos Corretores de Erros em Hardware para Sistemas de Telecomando e Telemetria em Aplicações Espaciais",** Porto Alegre, Brasil, Março 2007, Dissertação de Mestrado.

[14] ESA, **"Packet Telecommand Standard"**, ESA PSS-04-107 – issue 1, Jan 1988.

[15] Gaisler Research, **"Packet Telemetry Encoder"**, Convolutional-ESA.

[16]Almeida, G. M.; Bezerra, E. A. ; Cargnini, Luis Vitorio ; Fagundes, Rubem Dutra Ribeiro ; Mesquita, D. G. . **"A Reed-Solomon algorithm for FPGA area optimization in space applications"**. In: Second IEEE NASA/ESA Conference on Adaptive Hardware and Systems, AHS 2007., 2007, Edinburgh. Second IEEE NASA/ESA Conference on Adaptive Hardware and Systems, AHS 2007.. Los Alamitos, California : IEEE, 2007. p. 243-249.

[17] OAR Corporation. **"RTEMS Web Site"**. Extraído de http://www.rtems.com/ em 22 de maio 2009.

[18] NASA, **"Real Time Mars Approach Navigation aided by the Mars Network"**. Extraído de http://trs-new.jpl.nasa.gov/dspace/bitstream/2014/39788/1/06-0474.pdf em 22 de maio de 2009.

[19] Aeroflex Gaisler. **"LEON3 Web Site"**. Extraído de http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=13&Itemid=53 em 31 de março de 2009.

[20] Aeroflex Gaisler. **"LEON/ERC32 RTEMS Cross Compilation System (RCC)"**. Extraído de http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=150&Itemid=31 em 14 de abril de 2009.

[21] Aeroflex Gaisler. **"GRMON Web Site"**. Extraído de http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=39&Itemid=128 em 25 de junho de 2009.

[22] THOMAZINI, Daniel e ALBUQUERQUE, Pedro Urbano Braga. **"Sensores Industriais: Fundamentos e Aplicações"**. São Paulo : Ed. Érica, 2007.

[23] MEMS Exchange, "About MEMS and Nanotechnology". Extraído de http://www.memsnet.org/mems/ em 25 de junho de 2009.

[24] Lucas, Alfred. **"Théorie élémentaire du compas gyroscopique : a l'usage des marins et des aviateurs"**. Paris : Société d'Éditions, 1940.

[25] Analog Devices. **"ADIS16201 Datasheet"**. Extraído de www.analog.com/static/imported-files/Data_Sheets/ADIS16201.pdf em 25 de junho de 2009.

[26] Sousa, Célia Marise Ferreira de. **"A Integração do Sistema GPS/INS para a Monitorização da Linha de Costa do Litoral do Algarve"**. Algarve : s.n., 2004.

[27] THOMSON, Delmar. **"The 8051 Microcontroller: Architecture, Programming and Applications"**. Eagan : West Publishing Company, 1991.

[28] Freescale Semiconductor. **"SPI Block Guide"**. Extraído de http://www.freescale.com/files/microcontrollers/doc/ref_manual/S12SPIV3.pdf em 27 de junho de 2009.

[29] Brown, S. et al. **"Field-programmable gate arrays"**. Boston: Kluwer Academic, 1992.

[30] Surrey Space Centre, **"SSC"**. Extraído de http://www.ee.surrey.ac.uk/SSC/ em 27 de junho de 2009.

[31] T.Vladimirova, A da Silva Curiel, MPD'04, ESTEC. **"A System-on-a-Chip for Small Satellite Data Processing and Control ("ChipSat")"**, ESA Contract 14894

[32] INPE, **"A822000 – PRR – Multi-mission plataform specification",** 10 de agosto de 2001.

[33] INPE, **"A12700-SPC-01 Rev 01 - Amazonia-1 Satellite ACDH Subsystem Specification"**, 24 de junho de 2008.

[34] Analog Devices, **"ADIS16201 Datasheet"**. Extraído de http://www.analog.com/static/imported-files/data_sheets/ADIS16201.pdf em 27 de junho de 2009.

[35] Analog Devices, **"ADIS16250 Datasheet"**. Extraído de http://www.analog.com/static/imported-files/data_sheets/ADIS16250_16255.pdf em 27 de junho de 2009.

[36] NASA, **"EGSE (Electrical Ground Support Equipment) for ESA VEGA Launcher"**. Extraído de http://adsabs.harvard.edu/abs/2004ESASP.558..423F em 27 de junho de 2009.

[37] Fiethe, B.; Michalik, H.; Dierker, C.; Osterloh, B.; Zhou, G.; "**Reconfigurable System-on-Chip Data Processing Units for Space Imaging Instruments**" pp.181, 2007 Design, Automation & Test in Europe Conference & Exhibition, 2007

[38] Malcolm, N.; Zhao, W.; "**The Timed-Token Protocol for Real-Time Communications**" IEEE Comp. Los Alamitos, CA, USA. 1994;27(1):35-41.

[39] Ramamritham K; Stankovic J A.; "**Scheduling Algorithms and Operating Systems Support for Real-Time Systems**" IEEE Proceedings. 1994;82(1):55-67.

[40] Shin, K. G.; Ramanathan, P. A.; "**New Discipline of Computer Science and Engineering. IEEE Proceedings**" Australian. 1994;82(1):6-24.

[41] Aranci, G.; Maltecca, L.; Ranieri., R.; "**The On Board Data Handling Subsystem for XMM/Integral Space Missions**". DASIA 97 Conference on 'Data System in Aerospace'. Sevilla, Spain, 26-29 de Maio de 1997.

[42] ATMEL, "**TSC695F SPARC 32-bit Space Processor User Manual**". Extraído de http://www.atmel.com/dyn/resources/prod_documents/doc4148.pdf em 27 de Junho de 2009.

[43] Texas Instruments, "**MSC121X Precision ADC and DACs with 8051 Microcontroller and Flash Memory User's Guide**". Extraído de http://focus.ti.com/lit/ug/sbau101a/sbau101a.pdf em 23 de Julho de 2009.

[44] Actel, "**ProASIC 3E Handbook**". Extraído de http://www.actel.com/documents/PA3E_HB.pdf em 23 de Julho de 2009.

[45] Aeroflex Gaisler, "**GRLIB IP Library**". Extraído de http://www.gaisler.com/cms/index.php?option=com_content&task=section&id=13&Itemid=125 em 23 de Julho de 2009.

[46] Texas Instuments, "**MAX232, MAX232I DUAL EIA-232 Drivers/Receivers**". Extraído de http://focus.ti.com/lit/ds/symlink/max232.pdf em 23 Julho de 2009.

[47] OAR Corporation, "**RTEMS CPU Kit with SuperCore Documentation**". Extraído de http://www.rtems.com/doxygen/cpukit/html/confdefs_8h.html em 25 de Julho de 2009.

[48] Aeroflex Gaisler, "**LEON/ERC32 RTEMS Cross Compilation System (RCC)**". Extraído de http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=150&Itemid=31 em 25 de Julho de 2009.

[49] Cadence, "**Cadence OrCAD Solutions**". Extraído de http://www.cadence.com/products/orcad/pages/default.aspx 25 de Julho de 2009.

[50] Innalogics Ltda. "**Descrição do Projeto da UTMC**", Relatório Técnico de Documentação de Projeto, Innalogics, Porto Alegre, Jun 2009, 84 p.

[51] Emadi, Ali. "**Vehicular electric power systems : land, sea, air, and space vehicles**", Extraído de http://www.engnetbase.com/ejournals/books/book_km.asp?id=1527 em 29 de Julho de 2009.

Anexos

Anexo A Cronograma de Atividades

O cronograma proposto contempla cinco meses de trabalho. As áreas para estudo foram divididas de maneira que cada membro do grupo fosse responsável por um determinado assunto.

Tabela 6 - Cronograma de atividades

	Março	Abril	Maio	Junho	Julho
Estudos e Proposta					
Estudo INS					
Estudo LEON / RTEMS					
Estudo OBC					
Estudo UTMC					
Definição do Projeto					
Escrita da Proposta					
Projeto INS					
Projeto das Placas					
Confecção Placas					
Integração 8051/INS					
Teste INS					
Projeto OBC					
Síntese Leon					
Integração RTEMS					
Teste OBC					
Implementação OBC					
Algoritmo INS					
Protocolo de Comunicação UTMC					
Interpretação de Telecomandos					
Integração Física da Comunicação					
Teste Comunicação					
Teste Funcional					
Resultados					
Escrita TC					
Apresentação					

Legenda
Paulo
Cristiano
Felipe
Todos

Anexo B Instalação RCC

Após obter o arquivo tar com a distribuição binária, descomprima no local adequado, o RCC é compilado pra ser instalado no diretório "/opt/rtems-4.10" para qualquer plataforma.

```
# cd /opt

# bunzip2 -c sparc-rtems-4.10-gcc-4.3.3-1.1.99.x-linux.tar.bz2 | tar xf -
```

Para ter acesso direto as ferramentas disponíveis deve-se adicionar o diretório "/opt/rtems-4.10/bin" ao caminho de busca de executaveis:

```
# export PATH=$PATH:/opt/rtems-4.10/bin
```

Na pasta "/opt/rtems-4.10/src/samples" estão disponíveis uma série de códigos fonte para o RTEMS. Utilizamos esses exemplos para garantir que o LEON3 está funcionando perfeitamente.

Para compilar os códigos fontes deve-se executar os seguintes comandos:

```
# cd /opt/rtems-4.10/src/samples

# make
```

É necessário que o caminho do compilador "sparc-rtems-gcc" seja adicionado a variável de ambiente PATH para que o Makefile de compilação dos arquivos exemplo funcione corretamente.

Anexo C Instalação JTAG

Os drivers do cabo JTAG utilizados na instalação do ISE não funcionam na maioria das distribuições Linux, pois são específicos para o Red Hat Enterprise Linux. Por isso, é necessário instalar um driver usb alternativo que pode ser obtido na internet (ACHAR REFERENCIA BOA).

Antes de instalar os drivers é necessario ter a biblioteca libusb-dev instalada, para isso utilizamos o comando:

apt-get install libusb-dev

Após deve-se extrair o arquivo do driver para a pasta onde a ferramenta ISE está instalada e executar o make para efetuar a instalação:

tar xzf usb-driver-HEAD.tar.gz -C /opt/Xilinx/10.1

cd /opt/Xilinx/10.1/usb-driver

make

Com o comando "ls" verificamos se o driver foi instalado:

ls libusb-driver.so

Para testar o dispositivo deve-se utilizar o comando lsusb e verificar se o endereço é igual ao de baixo, ou seja, que o firmware inicial do dispositivo foi iniciado corretamente:

lsusb | grep "Xilinx"

Bus 006 Device 017: ID 03fd:0008 Xilinx, Inc.

Finalizando é necessário setar a seguinte variavél de ambiente antes de executar o GRMON:

"export LD_PRELOAD=/opt/Xilinx/10.1/usb-driver/libusb-driver.so"

Referências:

http://www.rmdir.de/~michael/xilinx/

http://blog.veiga.eti.br/como-instalar-o-ise-101-no-linux-ubuntu-804/

Anexo D *Layouts* Desenvolvidos

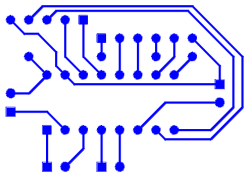

Figura 40 - Layout do Max232

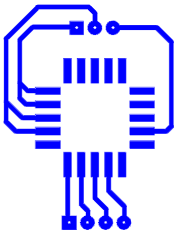

Figura 41 - Layout Giroscópio ADIS16250

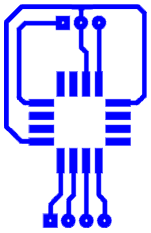

Figura 42 - Layout Acelerômetro ADIS16201

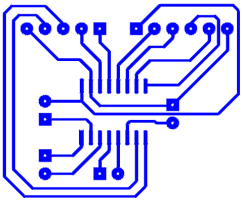

Figura 43 - Layout MAX3232

Anexo E VHDL do LEON3

File Edit View Terminal Help

```
-- ================== Modificação para adição de 2 UARTs ======================
  ua1 : if CFG_UART1_ENABLE /= 0 generate
    uart1 : apbuart                         -- UART 1
      generic map (pindex   => 1, paddr => 1, pirq => 2, console => dbguart,
                   fifosize => CFG_UART1_FIFO)
      port map (rstn, clkm, apbi, apbo(1), u1i, u1o);
    u1i.rxd <= rxd1; u1i.ctsn <= '0'; u1i.extclk <= '0'; --txd1 <= u1o.txd;
  end generate;
  noua0 : if CFG_UART1_ENABLE = 0 generate apbo(1) <= apb_none; end generate;

  ua2 : if CFG_UART2_ENABLE /= 0 generate
    uart2 : apbuart                         -- UART 2
    generic map (pindex => 9, paddr => 9,  pirq => 3, fifosize => CFG_UART2_FIFO)
    port map (rstn, clkm, apbi, apbo(9), u2i, u2o);
    u2i.rxd <= rxd2; u2i.ctsn <= '0'; u2i.extclk <= '0'; --txd2 <= u2o.txd;
  end generate;
  noua1 : if CFG_UART2_ENABLE = 0 generate apbo(9) <= apb_none; end generate;

  ua3 : if CFG_UART3_ENABLE /= 0 generate
    uart3 : apbuart                         -- UART 3
    generic map (pindex => 10, paddr => 10,  pirq => 4, fifosize => CFG_UART3_FIFO)
    port map (rstn, clkm, apbi, apbo(10), u3i, u3o);
    u3i.rxd <= rxd3; u3i.ctsn <= '0'; u3i.extclk <= '0'; --txd2 <= u2o.txd;
  end generate;
  noua2 : if CFG_UART3_ENABLE = 0 generate apbo(10) <= apb_none; end generate;
--============================================================================
                                                          347,85-81     58%
```

File Edit View Terminal Help

```
-- ================== Modificação para adição de 2 UARTs ======================
-- UART 1
  constant CFG_UART1_ENABLE : integer := 1;
  constant CFG_UART1_FIFO : integer := 8;
-- UART 2
  constant CFG_UART2_ENABLE : integer := 1;
  constant CFG_UART2_FIFO : integer := 32;
-- UART 3
  constant CFG_UART3_ENABLE : integer := 1;
  constant CFG_UART3_FIFO : integer := 32;
--============================================================================
                                                              132,3         80%
```