SISTEMA DE CONTROLE EM TEMPO REAL RECONFIGURAVEL DE EXPERIMENTO – ASTROBIOLOGIA

Tiago Sanches da Silva [1] ; Sérgio Ribeiro Augusto [2]

[1] Aluno de Iniciação Científica da Escola de Engenharia Mauá (EEM/CEUN-IMT);
[2] Professor da Escola de Engenharia Mauá (EEM/CEUN-IMT)

Resumo. *Este trabalho visa a concepção e implementação de um sistema operacional com características de tempo real e reconfigurável, para aplicações em experimentos espaciais, sendo parte do projeto de pesquisa "Sistema de Controle de Tempo Real Reconfiguravel em Astrobiologia", financiado pela Agência Espacial Brasileira (AEB). O sistema operacional será implementado em processador LEON [1] com arquitetura ternária adequada a aplicações que envolvem alta confiabilidade. A arquitetura de software proposta tem como base dois tipos de serviços: o PBS (Primary Boot Service), o qual é responsável pela inicialização do sistema bem como sua reconfiguração e atualização remota, e o APS (Aplication Service), responsável por controlar experimentos, sensores e atuadores. Este último tem como objetivo preparar os dados para enviar a uma base terrestre.*

Introdução

Os projetos espaciais são resultado da compilação de conhecimentos acumulados que resultam em experiências ousadas e ao mesmo tempo conservadoras. Desta forma a construção deste sistema operacional foi inspirada no satélite CoRoT [2], o qual possui arquitetura similar do ponto de vista de protocolos e sub sistemas, com bom histórico em missões espaciais. Esta inspiração garante uma boa escolha inicial e direcionamento para eventuais inovações.

Um sistema espacial pode geralmente ser dividido em dois grandes sistemas: carga útil e plataforma. A carga útil representa o experimento que se deseja colocar em órbita e a plataforma o veículo que transporta a carga útil. Esta proposta implica em definir claramente as interfaces de comunicação e o sistema operacional da carga útil. Este sistema é geralmente implementado com funcionalidade bem definida visando aumentar a robustez da aplicação. Entende-se por aumento de robustez a capacidade do software desenvolvido de não recorrer a situações que possam interromper o funcionamento definitivo do instrumento.

Para que esta confiabilidade seja obtida em um sistema que depende de sensores para tomar decisões de controle sobre os atuadores do experimento, é necessário que o *software* de controle seja desenvolvido em uma plataforma de tempo real [3], de maneira a se obter elevada velocidade de processamento e baixo tempo de resposta, levando a ações praticamente imediatas. Para a criação deste tipo de sistema são necessário conceitos de escalonamento de tarefas e definição de algum tipo de comunicação entre tarefas.

Este trabalho visa criar uma versão reduzida do sistema operacional desenvolvido para o CoRoT.

Material e Métodos

Hardware

Será utilizado a placa de desenvolvimento GR-PCI-XC5V adquirida com recursos do CNPQ, sob coordenação do programa Astrobiologia, para implementação do estudo. Esta placa de desenvolvimento é fruto da cooperação entre as empresas Aeroflex Gaisler [4] e Pender Electronic Design [5], sendo concebida especialmente para apoiar o desenvolvimento e prototipagem rápida de sistemas baseados no processador LEON, o qual foi escolhido neste projeto. Esta placa utiliza uma FPGA [6] Virtex-5 da Xilinx [7] de grande densidade de

células lógicas e capacidade de operar em barramento PCI [8]. A figura 1 fornece um diagrama de blocos da placa em questão.

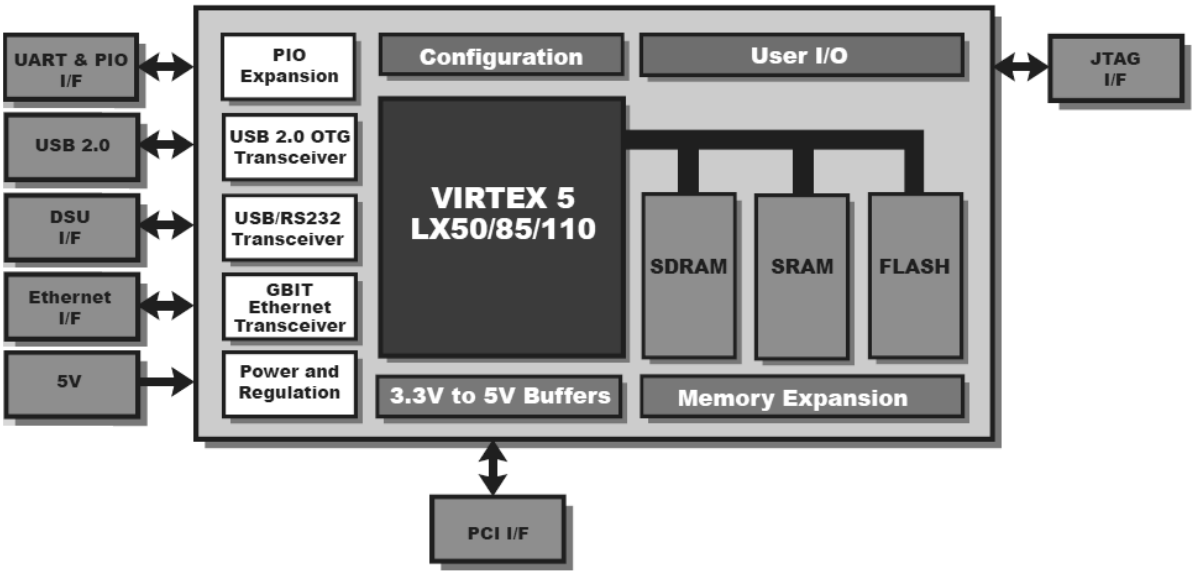

Figura 1 – Blocos funcionais e tecnologias da placa GR-PCI-XC5V

Como dito anteriormente, para unidade central de processamento será utilizado o processador LEON, baseado na arquitetura SPARC V8 [4], homologado para missões espaciais e também utilizado no CoRoT. Este processador será embarcado na FPGA Virtex 5 disponível na placa GR-PCI-XC5V.

Software

Para o desenvolvimento do Sistema Operacional de tempo real é necessário o desenvolvimento de conceitos de tarefas (*tasks*), prioridades entre tarefas, comunicação interna entre processos e principalmente escalonamento entre tarefas. Para agilizar o processo de criação deste sistema optou-se por utilizar o *kernel* de tempo real RTEMS (*Real-Time Executive for Multiprocessor System*)[9] com suporte a sistemas multitarefa. Este Kernel já possui algoritmos de escalonamento de tarefas e comunicação *inter-task* utilizando o conceito de *mailbox*.Também, o RTEMS é compatível com a arquitetura SPARC-V8.

Ferramentas de Desenvolvimento

Como ambiente de desenvolvimento é utilizado o LEON *Integrated Development Environment for Eclipse* [10], o qual possui todas as ferramentas necessárias para o desenvolvimento de um sistema em tempo real utilizando RTEMS embarcado em processadores LEON. A estrutura do ambiente de desenvolvimento é ilustrada na figura 2.

Para simulação é utilizado o *Plugin* TSIM para o ambiente Eclipse IDE. Quanto ao *debug* e ao monitoramento do sistema, o mesmo é realizado através do aplicativo GRMON[10], o qual interage diretamente com a placa de desenvolvimento utilizada.

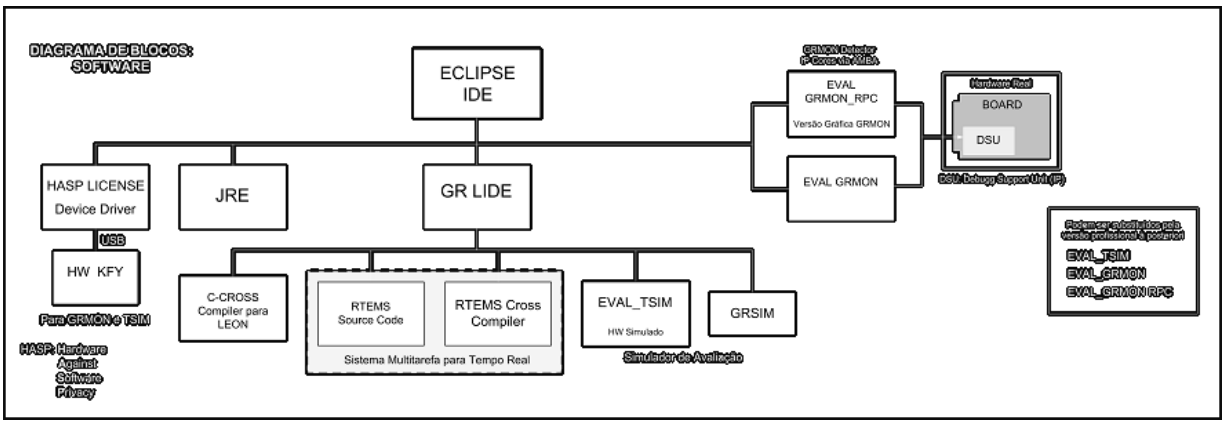

Figura 2 – Diagrama de ferramentas e softwares utilizados no projeto

Sistema Operacional de tempo real

O modelo geral do experimento pode ser observado na figura 3. A arquitetura proposta permite a comunicação do sistema de controle com o experimento utilizando uma rede CAN [11]. As informações coletadas do experimento a serem enviados ao solo (Telemetria - TM) e os comandos recebidos do solo (Telecomandos - TC) são disponibilizadas em uma rede MIL 1553[12], um protocolo bastante confiável para a comunicação entre o instrumento e a plataforma. O *software* proposto é inspirado na experiência do satélite CoRoT com um número reduzido de TC/TM. Sua estrutura é modular, permitindo a utilização de dois a trinta e dois sensores.

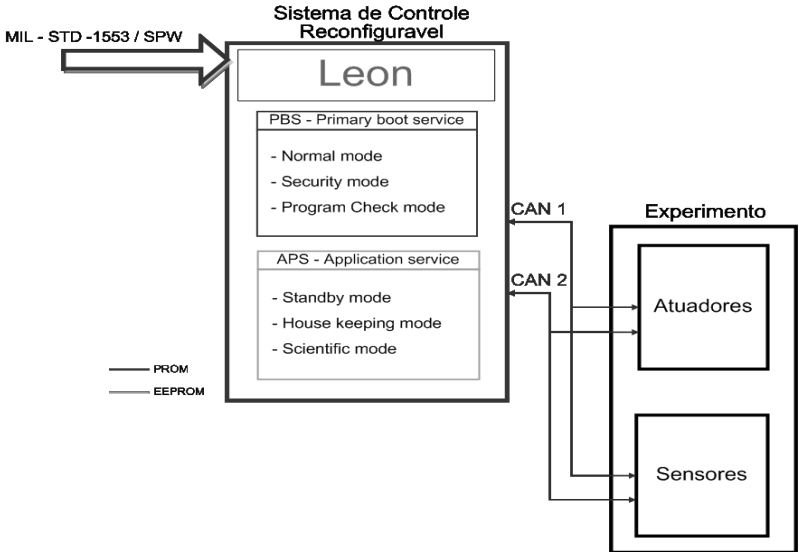

Figura 3- Diagrama geral do Sistema de Controle

Além de ser em tempo real o *software* é reconfigurável. Entende-se como reconfigurável a possibilidade de atualizar o *software* ou parte dele remotamente, e também a possibilidade de a qualquer hora, através de comandos enviados da base, realizar uma atualização de sua versão.

A arquitetura de software em estudo tem como base dois tipos de serviços: o PBS (*Primary Boot Service*), o qual é responsável pela inicialização do sistema bem como sua reconfiguração e atualização remota, e o APS (*Application Service*), responsável por controlar experimentos, sensores e atuadores. Este último tem como objetivo preparar os dados para enviar a uma base terrestre. Este trabalho focou no estudo da arquitetura do PBS e suas funções.

Resultado e Discussões

Este trabalho de iniciação científica modelou os estados dos serviços PBS e APS bem como sua estrutura multitarefa utilizando o diagrama de Harel [13]. Foram desenvolvidos também datagramas de comunicação para envio de comandos, reconhecimentos e respostas. Construíram também dois diagramas UML de atividades detalhados da comunicação entre plataforma e experimento, e algumas tabelas de controle utilizadas no gerenciamento da comunicação e comandos.

Serviços

Este sistema de controle possui dois serviços que são executados em paralelo pelo processador.

O APS é o serviço relativo a aplicação,ou seja, o gerenciamento do experimento propriamente dito. O mesmo possibilita telemetria de aquisição de dados dos sensores e varredura do instrumento para verificação de sua integridade. Este serviço pode ser atualizado e reconfigurado pela base, através do PBS que gerencia esta atualização do *firmware*.

O serviço PBS é responsável pelo sistema de boot após *reset* físico, permite uma manipulação mínima do sistema de processamento e inclui o *kernel* das funções elementares do *software* bem como a verificação da integridade do código no APS e suas atualizações. Toda comunicação é centralizada neste serviço, caso exista comando que seja direcionado ao APS, ele utiliza a comunicação *inter task* para entregá-lo.

PBS – Primary Boot Service	APS – Application Service
- Normal Mode	- Standby Mode
- Security Mode	- House Keeping Mode
- Program Check Mode	- Scientific Mode

Figura 4 – Estados possíveis dos serviços do Sistema Operacional

Como podemos observar na figura 4, cada um dos serviços possuem três estados de operação, onde é executado apenas um estado de cada vez em cada serviço.

O diagrama de Harel na figura 5 demonstra como as tarefas que são executadas, deixando bem explicito o paralelismo entre elas.

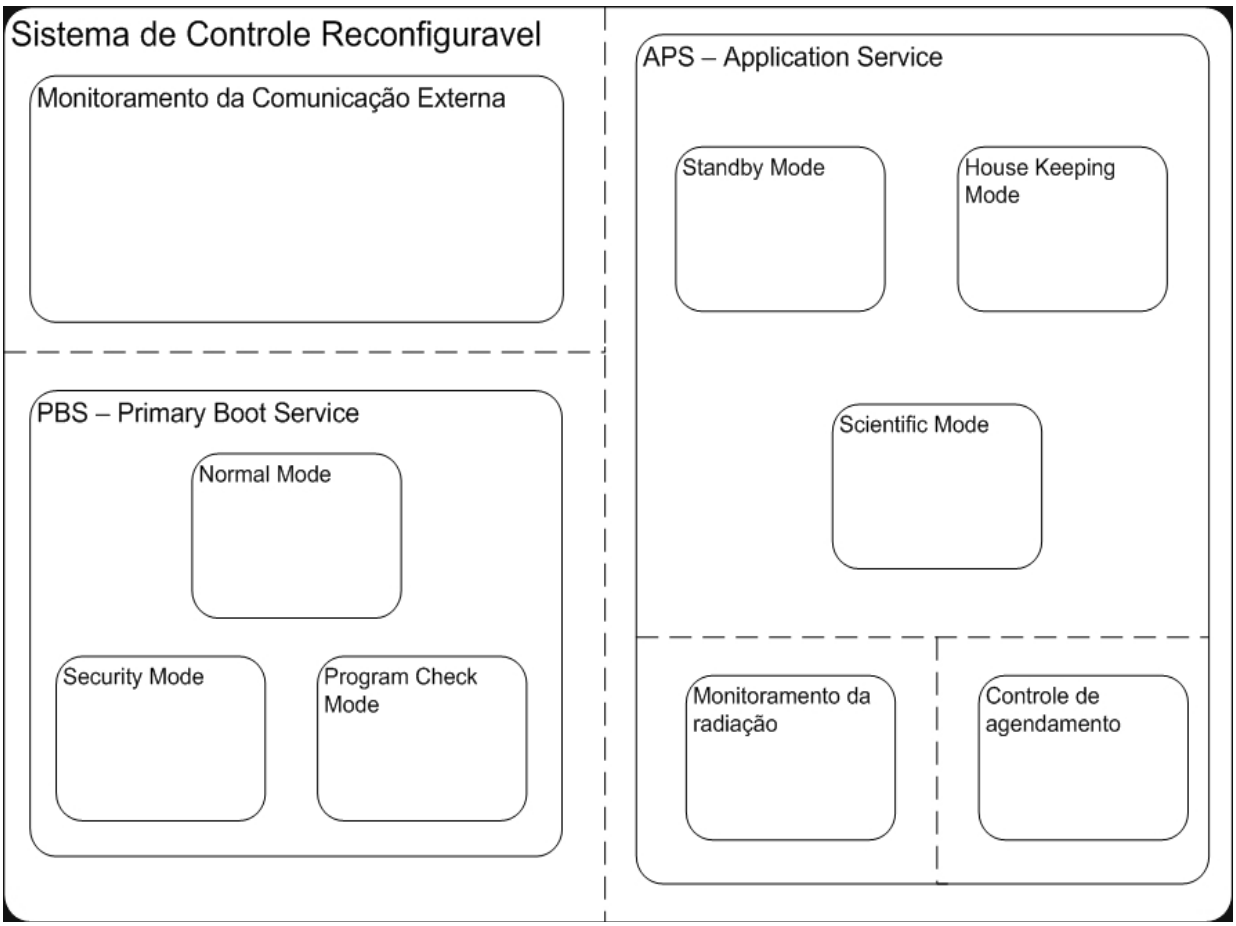

Figura 5 – Diagrama de Harel do Sistema de Tempo Real

As tarefas que estão sendo executadas em paralelo estão representadas por uma separação de linhas tracejadas, por exemplo, o Monitoramento da Comunicação Externa, o APS e o PBS estão sendo executados em paralelo. As tarefas que não estão separados por estas linhas são chamadas de estado e são executadas uma de cada vez.

A seguir são resumidas as funções de cada um dos estados pertencentes ao PBS e ao APS:

Normal mode (PBS): Executa uma versão do APS na inicialização do sistema e recebe comandos externos da comunicação, sendo que estes comandos servem para mudar o PBS ou o APS de estado.

Security Mode (PBS): O PBS entra neste modo para atualizar o código do APS ou parte dele. Este modo não pode ser acionado no meio de uma operação de risco do APS, ou seja, quando APS estiver interagindo diretamente com os sensores e atuadores.

Program Check Mode (PBS): Modo onde é verificado se o código gravado na EEPROM (APS) está integro e sem alterações.

Standby Mode: Este modo monitora a comunicação *inter task*, de maneira a verificar a existência de comandos enviados da base para o APS.

House keeping Mode: Este modo é responsável pela varredura dos sensores e atuadores do experimento, verificando se estão comunicáveis e íntegros.

Scientific Mode: Responsável pela telemetria do experimento. Também realiza o empacotamento das informações geradas pela telemetria para que seja enviada para a base ou armazenada em memória.

Módulo de Comunicação

Foi desenvolvido um modelo de comunicação visando a segurança na identificação do comando a ser realizado pelo experimento. Para tal, foi montada uma estrutura onde a

plataforma de solo envia o comando, o experimento interpreta o mesmo e envia um sinal de reconhecimento/interpretação (*ack*). A plataforma de solo verifica se o experimento interpretou corretamente o comando e envia um *reply acknowledge*, validando a interpretação ou retificando a interpretação do comando. Na figura 6 pode ser observado o modelo resumido da comunicação.

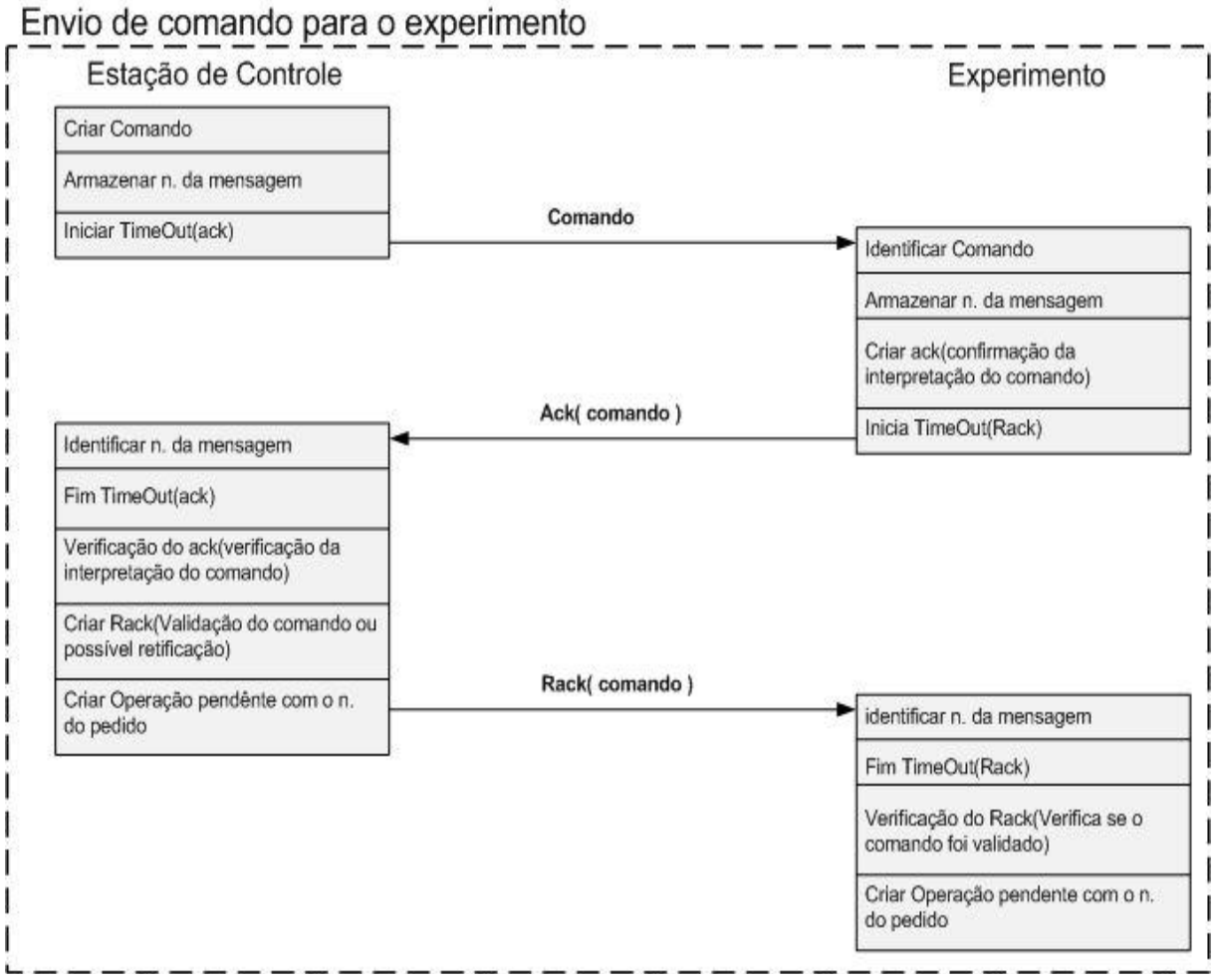

Figura 6 – Diagrama resumido da comunicação no envio de comandos

Foi criado um Diagrama *Unified Model Language* (UML) de Atividades completo deste esquema de comunicação que esta no anexo 1.

No fim da comunicação do comando é criada uma operação pendente que entra em uma fila no *firmware* do experimento para que seja realizada assim que possível. Após a realização da operação, sendo bem sucedida ou não, o experimento envia uma resposta a respeito da operação, o que chamamos neste trabalho de *OReply* (*Operational Reply*). O diagrama do modelo do *OReply* é mais simples que o de envio de comando, pois não existe a necessidade de reenvios e respostas, o OReply é encarado apenas como informativo, ou seja, não é um tipo de comunicação critica. Diagrama do *OReply* na figura 7.

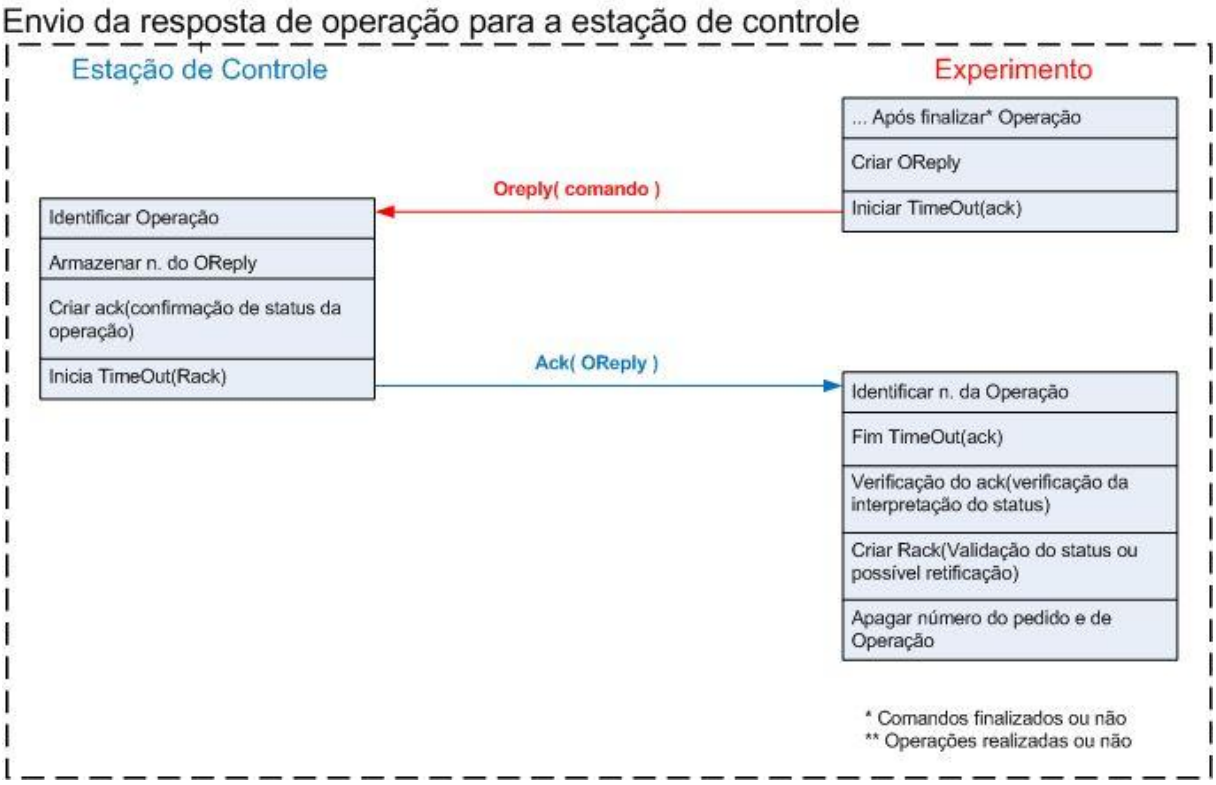

Figura 7 – Diagrama resumido do envio do *OReply*

O Diagrama UML de Atividades completo deste esquema do envio do *OReply* está no anexo 2.

Datagramas

O *datagrama* desenvolvido para a comunicação foi inspirado no *datagrama* do CoRoT. Segue o modelo geral dos *datagramas* desenvolvidos.

	Conteúdo	Tamanho
Header	*Packet_identifier*	1 Byte
	Packet_type	1 Byte
	Packet_Length	1 Byte
Comand	*Comand_ID*	1 Byte
Parameter	*Service*	1 Byte
	Parameter 1	1 Byte
	Parameter 2	1 Byte
	Parameter 3	1 Byte

Packet_identifier: Identifica o pacote que esta sendo enviado ao sistema remoto, o *ack* se referencia a este i*d* quando enviado.

Packet_type: define o tipo de pacote: se comando, *acknowledge* ou *OReply*.

Cmd_ID: Identificação do comando a ser executado pelo sistema.

Service: Indica para qual Serviço (tarefa) o comando está sendo direcionado.

Parameters: Utilizado para complementar alguns comandos, como por exemplo, o comando inicializar, que necessita que envie como parâmetro a versão do *firmware* a ser inicializado.

Conclusão

Este trabalho utilizou muitos conceitos de *softwares* para sistemas embarcados (*firmware*), como por exemplo, sistemas de tempo real, modelos de comunicação, modelagem de serviços e de tarefas, arquiteturas baseadas na solução SPARC V8, *datagramas* e protocolos de comunicação. O foco foi o desenvolvimento teórico e modelamento dos sistemas envolvidos. Infelizmente não foi possível sua implementação prática, pois a placa de desenvolvimento necessária para o mesmo não estava disponível. Foram concluídos os diagramas e modelos de comunicação e de estados do PBS, estando pronto para implementação quando a placa GR-PCI-XC5V estiver disponível.

Referências Bibliográficas

[1] Aeroflex Gaisler. (2010) Grlib ip core user's manual. Version 1.0.22.

[2] Convection Rotation and Planetary Transits, Acesso em setembro, disponível em: http://www.esa.int/esaMI/COROT/index.html

[3] HANDSCHIN, Edmund, ed. Real-time control of electric power systems. Amsterdam: Elsevier, 1972. 296 p.

[4] Gaisler. Disponível em:.http://www.gaisler.com/. Acesso em: outubro de 2010

[5] Pender Electronic Design, Acesso em outubro de 2010, disponível em: http://www.pender.ch/

[6] Chu, Pong P. (2006) RTL HARDWARE DESIGN USING VHDL - coding for efficiency, Portability, and Scalability - Chu, Pong P. Wiley-Interscience

[7] Xilinx. Disponível em:.http://www.xilinx.com. Acesso em: Setembro de 2010

[8] ABBOTT, Doug:. (2004) PCI bus demystified. 2. ed. Elsevier.

[9] Coutinho, M.; Rufino, J. et AL. (2005) Control of Event Handling Timeliness in RTEMS, Parallel and Distributed Computing and Systems

[10] Gaisler. Disponível em: http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=115&Itemid=103 Acesso em: abril de 2010

[11] KAWAMOTO, Bruno Seiji et al. Aquisição de sinais de usinas usando o protocolo CAN. Orientador: DENIS, Everson. São Caetano do Sul, SP: CEUN-EEM, 2006.

[12] MIL-STD-1553B Interface Standard for Digital Time Division Command/Response Multiplex Data Bus. DoD, 21 Sep 1978

[13] Harel, D. (1987) *Statecharts: A visual formalism for complex systems*, Science of Computer Programming , 8.

Anexo 1:

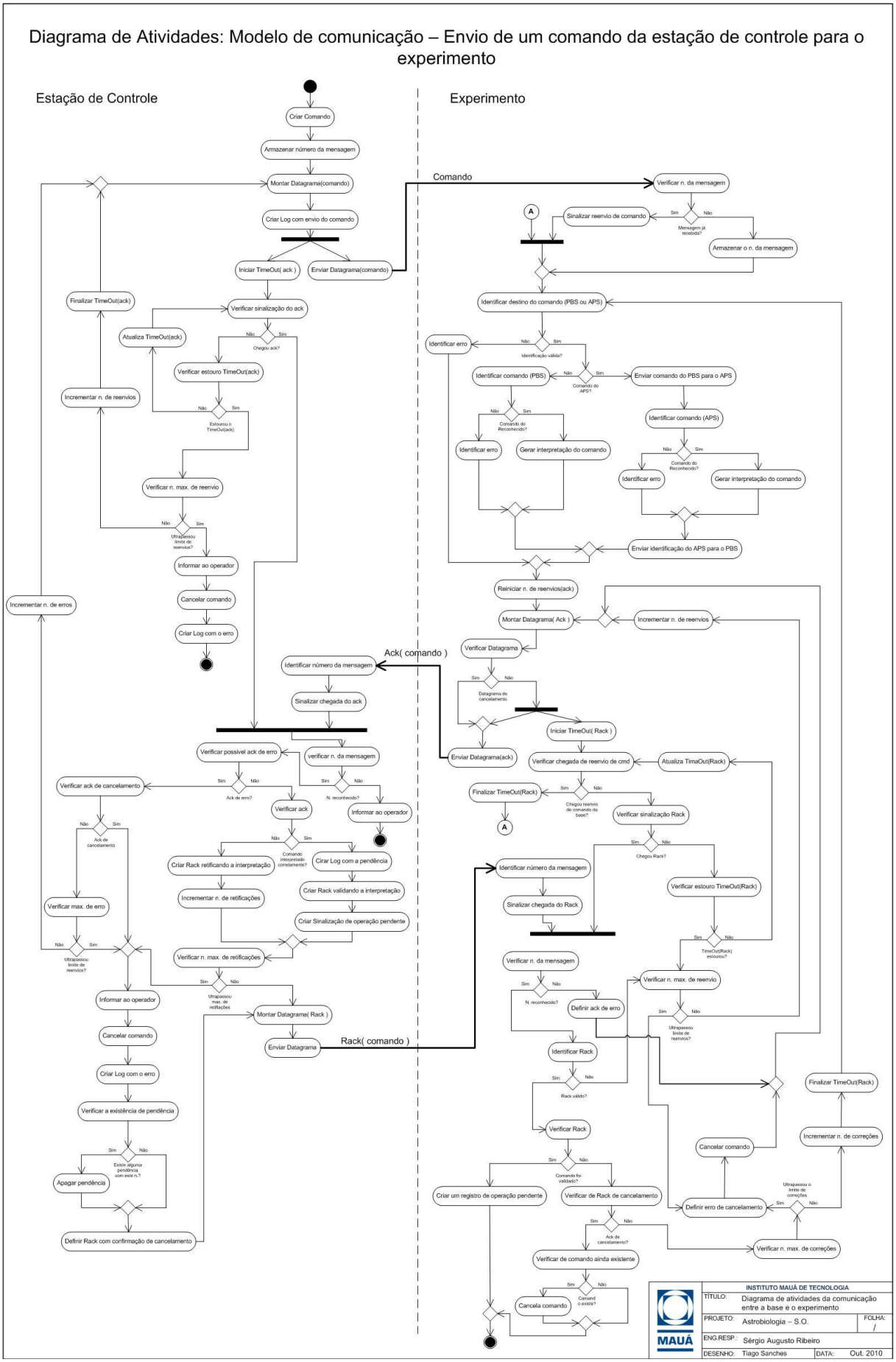
Diagrama de Atividades: Modelo de comunicação – Envio de um comando da estação de controle para o experimento
Estação de Controle
Experimento
Criar Comando
Armazenar número da mensagem
Montar Datagrama(comando)
Criar Log com envio do comando
Iniciar TimeOut(ack)
Enviar Datagrama(comando)
Comando
Finalizar TimeOut(ack)
Verificar sinalização do ack
Atualiza TimeOut(ack)
Chegou ack?
Verificar estouro TimeOut(ack)
Incrementar n. de reenvios
Estourou o TimeOut(ack)
Verificar n. max. de reenvio
Ultrapassou limite de reenvios?
Informar ao operador
Cancelar comando
Criar Log com o erro
Incrementar n. de erros
Verificar n. da mensagem
Sinalizar reenvio de comando
Mensagem já recebida?
Armazenar o n. da mensagem
Identificar destino do comando (PBS ou APS)
Identificar erro
Identificação válida?
Identificar comando (PBS)
Comando do APS?
Enviar comando do PBS para o APS
Identificar comando (APS)
Comando do Reconhecido?
Identificar erro
Gerar interpretação do comando
Identificar erro
Gerar interpretação do comando
Enviar identificação do APS para o PBS
Reiniciar n. de reenvios(ack)
Montar Datagrama(Ack)
Incrementar n. de reenvios
Verificar Datagrama
Datagrama de cancelamento
Iniciar TimeOut(Rack)
Ack(comando)
Identificar número da mensagem
Sinalizar chegada do ack
Enviar Datagrama(ack)
Verificar chegada de reenvio de cmd
Atualiza TimaOut(Rack)
Finalizar TimeOut(Rack)
Chegou reenvio de comando da base?
Verificar sinalização Rack
Verificar possível ack de erro
verificar n. da mensagem
Verificar ack de cancelamento
Ack de erro?
N. reconhecido?
Informar ao operador
Verificar ack
Ack de cancelamento
Comando interpretado corretamente?
Cirar Log com a pendência
Criar Rack retificando a interpretação
Incrementar n. de retificações
Criar Rack validando a interpretação
Criar Sinalização de operação pendente
Verificar max. de erro
Ultrapassou limite de reenvios?
Verificar n. max. de retificações
Ultrapassou max. de retificações
Informar ao operador
Cancelar comando
Criar Log com o erro
Verificar a existência de pendência
Existe alguma pendência com este n.?
Apagar pendência
Definir Rack com confirmação de cancelamento
Montar Datagrama(Rack)
Enviar Datagrama
Rack(comando)
Chegou Rack?
Identificar número da mensagem
Verificar estouro TimeOut(Rack)
Sinalizar chegada do Rack
TimeOut(Rack) estourou?
Verificar n. da mensagem
Verificar n. max. de reenvio
N. reconhecido?
Definir ack de erro
Ultrapassou limite de reenvios?
Identificar Rack
Rack valido?
Finalizar TimeOut(Rack)
Verificar Rack
Incrementar n. de correções
Comando foi validado?
Cancelar comando
Criar um registro de operação pendente
Verificar de Rack de cancelamento
Definir erro de cancelamento
Ultrapassou o limite de correções
Ack de cancelamento?
Verificar n. max. de correções
Verificar de comando ainda existente
Comand o existe?
Cancela comando
INSTITUTO MAUÁ DE TECNOLOGIA
TÍTULO: Diagrama de atividades da comunicação entre a base e o experimento
PROJETO: Astrobiologia – S.O.
FOLHA: /
ENG.RESP.: Sérgio Augusto Ribeiro
DESENHO: Tiago Sanches
DATA: Out. 2010
MAUÁ

Anexo 2:

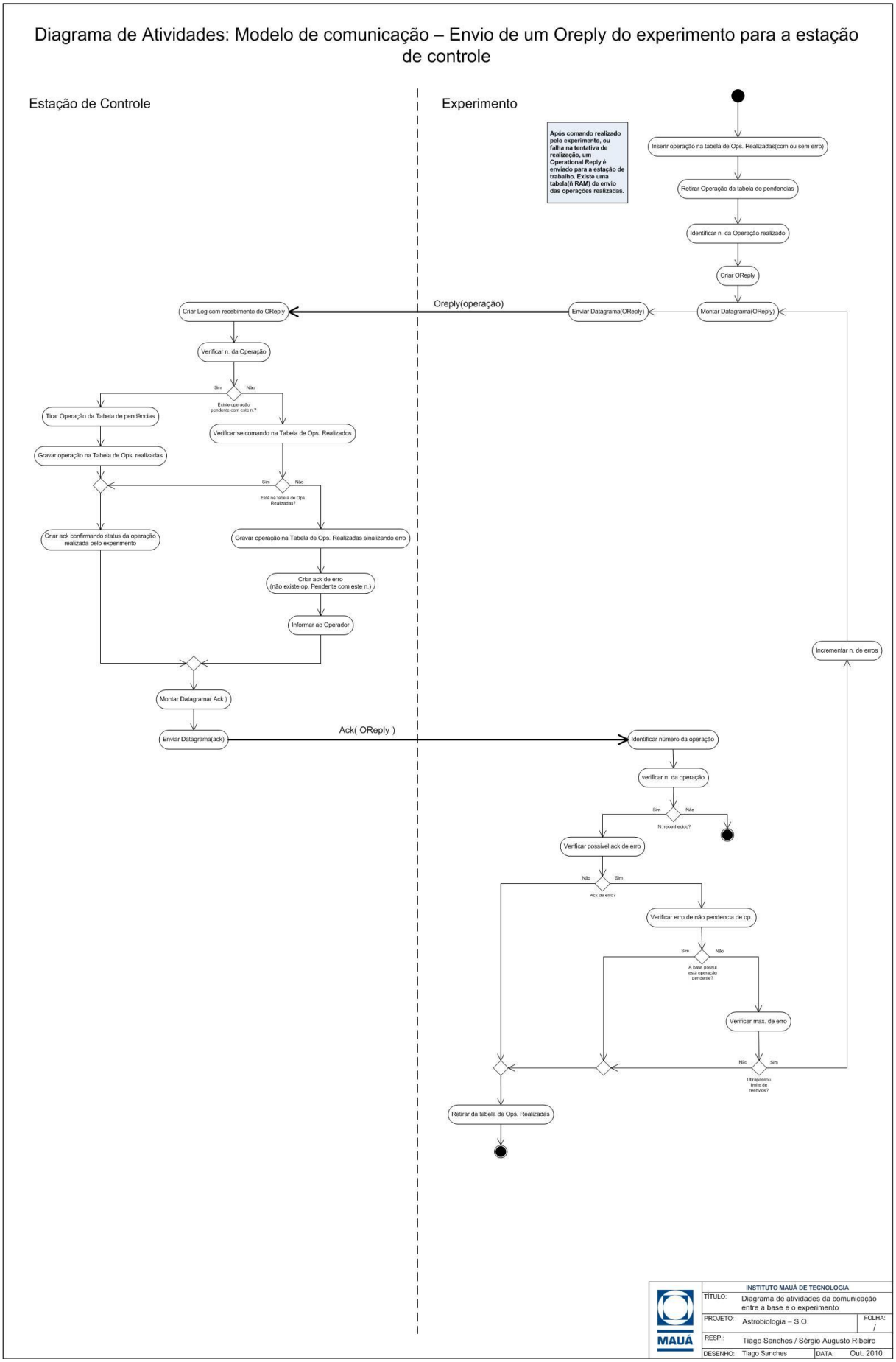
Diagrama de Atividades: Modelo de comunicação – Envio de um Oreply do experimento para a estação de controle
Estação de Controle
Experimento
Após comando realizado pelo experimento, ou falha na tentativa de realização, um Operational Reply é enviado para a estação de trabalho. Existe uma tabela(rt RAM) de envio das operações realizadas.
Inserir operação na tabela de Ops. Realizadas(com ou sem erro)
Retirar Operação da tabela de pendencias
Identificar n. da Operação realizado
Criar OReply
Montar Datagrama(OReply)
Enviar Datagrama(OReply)
Oreply(operação)
Criar Log com recebimento do OReply
Verificar n. da Operação
Sim
Não
Existe operação pendente com este n.?
Tirar Operação da Tabela de pendências
Verificar se comando na Tabela de Ops. Realizados
Gravar operação na Tabela de Ops. realizadas
Sim
Não
Está na tabela de Ops. Realizadas?
Criar ack confirmando status da operação realizada pelo experimento
Gravar operação na Tabela de Ops. Realizadas sinalizando erro
Criar ack de erro (não existe op. Pendente com este n.)
Informar ao Operador
Montar Datagrama(Ack)
Enviar Datagrama(ack)
Ack(OReply)
Identificar número da operação
verificar n. da operação
Sim
Não
N. reconhecido?
Verificar possível ack de erro
Não
Sim
Ack de erro?
Verificar erro de não pendencia de op.
Sim
Não
A base possui está operação pendente?
Verificar max. de erro
Não
Sim
Ultrapassou limite de reenvios?
Incrementar n. de erros
Retirar da tabela de Ops. Realizadas
INSTITUTO MAUÁ DE TECNOLOGIA
TÍTULO: Diagrama de atividades da comunicação entre a base e o experimento
PROJETO: Astrobiologia – S.O.
FOLHA: /
RESP.: Tiago Sanches / Sérgio Augusto Ribeiro
DESENHO: Tiago Sanches
DATA: Out. 2010
MAUÁ