UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ENGENHARIA DE SÃO CARLOS

DEPARTAMENTO DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

MURILO SILVEIRA ROCHA

DESENVOLVIMENTO DE UM ANALISADOR DE REDE PROFIBUS DE BAIXO CUSTO

Trabalho de Conclusão de Curso

São Carlos

2016

MURILO SILVEIRA ROCHA

DESENVOLVIMENTO DE UM ANALISADOR DE REDE PROFIBUS DE BAIXO CUSTO

Trabalho de Conclusão de Curso apresentado

à Escola de Engenharia de São Carlos da

Universidade de São Paulo

Curso de Engenharia Elétrica com ênfase em

Sistemas de Energia e Automação

Orientador: Dennis Brandão

São Carlos

2016

Ficha catalográfica preparada pela Seção de Tratamento
Da informação do Serviço de Biblioteca – EESC/USP

Rocha, Murilo Silveira
R672d Desenvolvimento de um analisador de rede Profibus de baixo custo / Murilo Silveira Rocha; orientador Dennis Brandão. São Carlos, 2016.

Monografia (Graduação em Engenharia Elétrica com ênfase em Sistemas de Energia e Automação) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2016.

1. Profibus. 2. Analisador de rede. 3. BeagleBone. 4. Redes industriais. I. Título.

FOLHA DE APROVAÇÃO

Nome: Murilo Silveira Rocha

Título: "Desenvolvimento de um analisador de rede Profibus de baixo custo"

Trabalho de Conclusão de Curso defendido e aprovado em 18 / 11 / 16 ,

com NOTA 10,0 (dez) **, pela Comissão Julgadora:**

Prof. Associado Dennis Brandão - Orientador - SEL/EESC/USP

Prof. Associado Adilson Gonzaga - SEL/EESC/USP

Mestre Guilherme Serpa Sestito - Doutorando-SEL/EESC/USP

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado José Carlos de Melo Vieira Júnior

AGRADECIMENTOS

Agradeço primeiramente aos meus pais por todo o apoio, motivação e por sempre me proporcionarem as melhores oportunidades para alcançar tudo que alcancei.

Ao meu irmão Alex pelos ensinamentos de programação que foram um grande diferencial para minha vida.

À minha irmã Cibele por todo o companheirismo e amizade que sempre tivemos.

À Elisa por estar ao meu lado nos bons e maus momentos, sempre me apoiando e motivando a continuar.

A todos meus amigos e familiares que me acompanharam nessa jornada até o momento.

Ao professor Dr. Dennis Brandão pela orientação, apoio e oportunidade de realizar essa pesquisa.

Ao pessoal que conheci no Laboratório de Automação Industrial, que contribuíram para o meu aprendizado na área em que o projeto foi desenvolvido. Em especial ao Guilherme Serpa Sestito, que me aconselhou por todo o desenvolvimento deste trabalho, além de compartilhar seus conhecimentos e experiências do ramo acadêmico.

RESUMO

ROCHA, M. S.; **Desenvolvimento de um analisador de rede Profibus de baixo custo**. 2016. 111f. Monografia (Trabalho de Conclusão de Curso) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2016.

Este trabalho apresenta o desenvolvimento de um analisador de rede Profibus que coleta dados digitais a qualquer velocidade da rede e amostras analógicas do sinal de um dos canais para velocidades da rede abaixo de 500kbps. O desenvolvimento priorizou o menor custo possível, escolhendo assim o hardware que permitisse essa funcionalidade com um custo muito menor que os equipamentos similares do mercado. Houve também a preocupação da disponibilização dos dados coletados em tempo real, de maneira em que o usuário não ficasse atrelado a softwares fechados para realizar a coleta, portanto essa ferramenta poderá ser utilizada em pesquisas com processamento de dados em tempo real para redes Profibus em qualquer linguagem com suporte a comunicação TCP/IP.

Palavras chave: Profibus. Analisador de rede. Beaglebone. Redes Industriais.

ABSTRACT

ROCHA, M. S.; Development of a low cost Profibus network analyzer. 2016. 111p. Monografia (Trabalho de Conclusão de Curso) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2016.

This paper presents the development of a Profibus network analyzer that collects digital data from the network on any baud rate and analog samples from one of the network's channels running on baud rates below 500kbps. The project prioritized the lowest cost possible, choosing the hardware that allows this functionality at a much lower cost than similar market equipment. There was also the concern of the availability of the data collected in real time, so that the user would not be tied to closed software for the capture, so this tool could be used in research with real-time data processing for Profibus networks in any programming language with TCP/IP support.

Keywords: Profibus. Network analyzer. Beaglebone. Industrial network.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Desde a década de 90 as redes de automação industrial vêm se tornando mais essenciais para a manufatura e controle de processos. O número de fabricantes de sensores e atuadores aumentaram significativamente, assim como o desenvolvimento de novas redes para atender às novas necessidades. [1]

A rede Profibus é uma das redes industriais mais utilizadas no mundo, com aproximadamente 50 milhões de dispositivos já instalados, de acordo com a Profibus International, associação que regulamenta o protocolo, demandando assim cada vez mais profissionais qualificados para realizar instalações adequadas de novas redes, manutenções, certificações e diagnósticos de redes já operantes, garantindo sua funcionalidade com o mínimo de interrupções possíveis. [1]

Esses profissionais necessitam de várias ferramentas para realizar análises na rede Profibus, sendo que esses equipamentos e cada tipo de diagnóstico serão mais detalhados durante a apresentação do trabalho. Um dos métodos mais completos de se realizar um diagnóstico é com o uso de uma ferramenta chamada analisador de rede. Um analisador de rede consegue capturar informações dos dados digitais transmitidos pela rede, fazendo assim uma interpretação de cada telegrama da rede Profibus. Esses telegramas possuem campos específicos que podem indicar diagnósticos vindo dos equipamentos ligados na rede, que contém informações mais detalhadas sobre os problemas que estão acontecendo. [2]

Outra funcionalidade dos analisadores de rede é a captura analógica dos canais de transmissão da rede Profibus. Com a captura analógica é possível descobrir defeitos da instalação física da rede, como a medida da qualidade do sinal transmitido por ela, indicando interferências eletromagnéticas, reflexões de onda ou até mesmo curtos-circuitos entre linhas. Aliando com o conhecimento do profissional qualificado para realizar o diagnóstico, o problema pode ser rapidamente descoberto ou até mesmo prevenido antes que cause alguma interrupção da rede. [2]

1.1 Motivação

Existem diversos modelos comerciais de analisadores de rede Profibus, a maioria com as funções básicas de captura de telegramas e capturas da forma de onda do sinal transmitido pela rede. Existem outros com análises mais avançadas dos dados coletados na rede, que incrementam muito seu valor de mercado.

Além do preço elevado, outro problema enfrentado é que esses produtos são vendidos juntamente com o *driver* e o *software* que realiza a comunicação com o produto. Muitas vezes não é possível encontrar o *driver* para todos os sistemas operacionais, principalmente atualizações que permitem a compatibilidade com os sistemas mais novos. Outro problema enfrentado é que esses *softwares* não disponibilizam a informação coletada da maneira que o usuário deseja. Um exemplo disso seria o armazenamento dos pontos analógicos em arquivo: os *softwares* permitem a visualização de uma tela parecida com um osciloscópio, porém não permitem que esses pontos sejam salvos em um formato de arquivo de texto para a utilização deles em outros programas, como Excel ou Matlab.

1.2 Objetivos

Como objetivos desse trabalho ficou definido que o projeto teria em vista a solução desses dois problemas encontrados nos analisadores de rede Profibus comerciais já existentes: buscar o melhor *hardware* que realize as tarefas de captura dos telegramas da rede Profibus e capture as amostras analógicas da rede por um preço acessível, além também que a disponibilidade desses dados independa de drivers e softwares fechados, deixando o usuário livre para escolher como processar o dado coletado da rede.

1.3 Estruturação do trabalho

A organização deste trabalho foi feita da seguinte forma:

- Capítulo 2: Apresenta a rede Profibus e as características do protocolo, além de mostrar os métodos e ferramentas de diagnósticos disponíveis para essa rede.
- Capítulo 3: Apresenta os conceitos básicos do *hardware* escolhido: o computador de placa BeagleBone Black Rev. C, mostrando os básicos de sua

arquitetura de sistema e também os subsistemas disponíveis em seu processador que foram utilizados para a realização deste trabalho.

- Capítulo 4: Apresenta a arquitetura do software desenvolvido para a BeagleBone, definindo suas principais funções e explicando seu funcionamento por meio de diagramas de fluxo de dados separados em quatro módulos.
- Capítulo 5: Apresenta um manual de instruções para o uso da ferramenta desenvolvida, indicando ao usuário da ferramenta qual o protocolo que deve ser seguido para realizar a comunicação TCP/IP e assim conseguir configurar e iniciar a ferramenta, além de interpretar corretamente os dados recebidos.
- Capítulo 6: Apresenta um software desenvolvido para demonstrar a utilização da ferramenta para a captura de dados, exibindo uma *live-list* da rede Profibus conectada, além de exibir as amostras analógicas referentes a cada endereço dos equipamentos conectados.

1.4 Lastros da pesquisa

Essa pesquisa rendeu o prêmio de melhor trabalho apresentado no V SICEEL, Simpósio de Iniciação Científica da Engenharia Elétrica – EESC/USP e UFSCAR, no ano de 2016.

2 REDE PROFIBUS

Foi fundamental para o desenvolvimento deste trabalho o conhecimento do protocolo Profibus, desde o modo no qual os bits são transmitidos pela rede até o entendimento de cada tipo de mensagem e suas propriedades. Neste capítulo será abordado um resumo da teoria do funcionamento dessa rede, assim como a função exercida pelo equipamento desenvolvido neste trabalho, destacando algumas diferenças entre as versões Profibus-DP e Profibus-PA, porém focando na primeira devido a maior utilização no trabalho.

2.1 História do protocolo Profibus

A rede Profibus é uma das redes industriais mais utilizada do mundo, com uma estimativa de 53,7 milhões de dispositivos instalados [1]. Seu surgimento se dá a partir de um projeto de pesquisa de diversas empresas e institutos de pesquisas apoiado pelo governo alemão, entre os anos de 1987 e 1990. [3]

Foi desenvolvido como resultado dessa pesquisa o padrão Profibus FMS, que continha muitas funções sofisticadas e permitia uma comunicação complexa entre os dispositivos da rede. Em 1993 foi publicada uma nova parte dessa norma, que inclui o protocolo Profibus DP, que contém somente as funções básicas presentes no Profibus FMS, tornando-se assim um protocolo mais simples, de fácil entendimento e com a capacidade plena de realizar a troca de dados que os fabricantes de equipamentos buscavam naquele tempo. [3]

Nos anos posteriores surgiu uma nova versão da rede Profibus, chamada de Profibus PA. Esta versão possui requisitos que atendem as necessidades das indústrias de processos como, por exemplo, a segurança intrínseca de seus equipamentos, e assim podem ser utilizados em zonas classificadas, muito comum neste tipo de indústria. [3]

A Profibus International (PI) é responsável por gerenciar as atualizações do protocolo, realizar o *marketing* e dar suporte necessário aos usuários da tecnologia. Este suporte é realizado diretamente com a PI ou através de centros instalados em vários países que tem a função de realizar treinamentos de capacitação em Profibus. [3]

Uma outra característica da PI é a certificação de produtos em laboratórios certificados para garantir que eles atendem a todas as especificações exigidas pelas normas vigentes. Esse produto recebe então um selo de qualidade, garantindo seu funcionamento na rede. [3]

2.2 Camadas de rede da Profibus

O Profibus está baseado no modelo de protocolos de comunicação ISO/OSI, no qual apenas três camadas deste modelo são implementadas para Profibus: a camada física (1), a camada de enlace (2) e a camada de aplicação (7). Há ainda a implementação da camada de usuário, que não é abordada pelo modelo ISO/OSI, porém nessa camada há o acesso da camada de enlace por meio do *Direct Data Link Mapper* (DDLM). Isso torna-se possível ao usuário ou às ferramentas na rede de trocar dados de configuração, parametrização, diagnóstico e outros entre os dispositivos dessa rede. Pode-se visualizar essas camadas implementadas na rede conforme mostrado na Figura 2-1. [2]

	Usuário	
7	Camada de Aplicação	DP-V0, DP-V1, DP-V2
6	Camada de Apresentação	Não usado
5	Camada de Sessão	Não usado
4	Camada de Transporte	Não usado
3	Camada de Rede	Não usado
2	Camada de Enlace	FDL: Fieldbus Data Link Mestre e Escravo Token
1	Camada Física	RS-485, Fibra Ótica e MBP

Figura 2-1 - Camadas da Rede Profibus [2]

2.2.1 Camada física

A camada física da rede Profibus é responsável pela descrição de normas técnicas que devem ser cumpridas em relação à tecnologia de transmissão de dados, a posição e funções de todas as conexões dos conectores do protocolo e também de parâmetros técnicos que devem ser seguidos [2]. Todas essas normas estão apresentadas em [4].

No Profibus-DP o meio físico mais utilizado é o padrão EIA-485, que é o termo atual de se referenciar o padrão RS-485. Esse meio físico conta com a transmissão via cabo de par trançado, que devido às características elétricas como resistência e indutância impostas pela norma, permite que até 32 equipamentos sejam ligados em um mesmo barramento sem o uso

de repetidor. Utilizando-se repetidores é possível obter uma rede de até 126 equipamentos, respeitando o limite máximo de dispositivos em cada trecho e também o tamanho do cabo utilizado em cada trecho em função da taxa de transmissão da rede. [2]

Na Profibus-DP existem dez taxas de transmissão que podem ser definidas no projeto de uma nova rede. Conforme citado, a taxa de transmissão tem relação direta ao comprimento máximo de cabo ao longo do segmento de uma rede Profibus, sendo essa uma relação inversa entre as duas variáveis. Como observado na Tabela 2-1 os valores entre taxa de transmissão da rede, chamado também de *Baud Rate*, em relação ao comprimento máximo de cabo ao longo de um segmento, dado entre dois repetidores.

Taxa de Transmissão	Comprimento máximo do cabo por segmento (m)
9.6 kbps	1200
19.2 kbps	1200
45.45 kbps	1200
93.7 kbps	1200
187.5 kbps	1000
500 kbps	400
1.5 Mbps	200
3 Mbps	100
6 Mbps	100
12 Mbps	100

Tabela 2-1 - Comprimento máximo do Cabo por segmento em relação à taxa de transmissão. [3]

Os dados transmitidos pelo padrão EIA-485 passam por dois canais, nomeados na Profibus pelos fios A e B, que possuem valores de tensão iguais, porém com polaridade oposta. Um sinal dentro das normas da Profibus é dado por tensões em módulo entre esses dois terminais de 4V a 7V pico a pico [2]. Um sinal transmitido pelo meio físico EIA-485 se assemelha ao sinal mostrado na Figura 2-2.

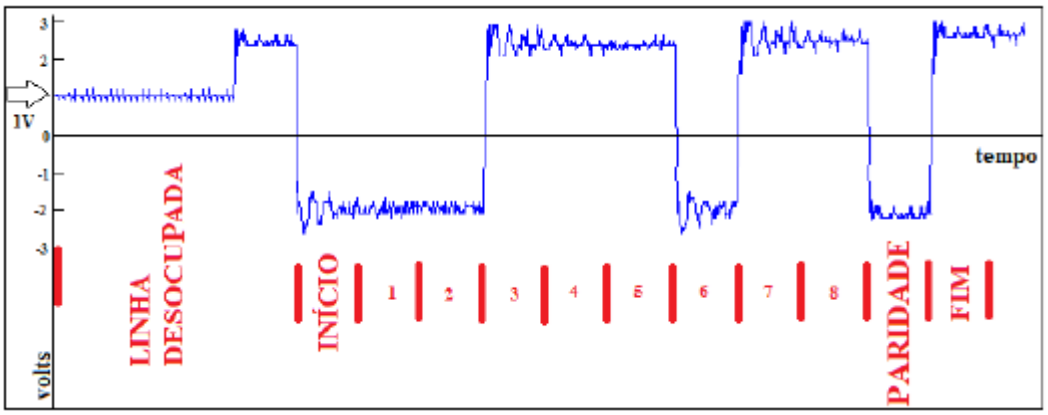


Figura 2-2 - Sinal analógico de um dos canais do padrão EIA-485. [2]

Os equipamentos que estão conectados à rede Profibus são ligados fisicamente a ela por meio de um conector, que possuem recomendações técnicas sobre especificações mecânicas e também de proteção elétrica, segundo a *Profibus Design Guideline (2015)*. Uma dessas especificações necessárias em um conector Profibus é a presença de terminadores ligados nas duas extremidades de cada segmento da rede, ou então o uso de um equipamento chamado terminador ativo em cada segmento [2]. Ambos possuem a função de evitar a reflexão de onda nas extremidades do segmento, que causa nas regiões centrais do segmento o efeito de superposição da onda, gerando deformação no sinal original, chegando até a comprometer a leitura correta de cada bit [3]. Uma representação desse efeito de reflexão devido ao desligamento ou uso incorreto dos terminadores em um segmento pode ser visto na Figura 2-3.

Outros problemas de transmissão pelo padrão EIA-485 podem aparecer e por meio da obtenção das amostras analógicas do segmento analisado é possível diagnosticar qual a causa do problema. Esses problemas mais comuns que aparecem devido à defeitos no meio físico são: [2]

- Falta de terminadores nas extremidades de cada segmento
- Excesso de terminador por segmento
- Falta de energização dos terminadores
- Interferência eletromagnética
- Cabeamento muito longo para o *Baud Rate* escolhido
- Curto circuito entre as linhas A e B ou entre uma linha e a malha de blindagem do cabo

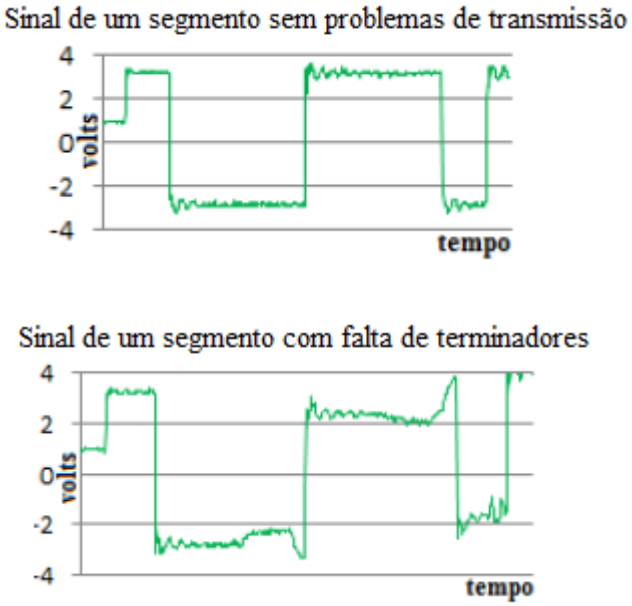


Figura 2-3 - Diferença entre um sinal sem problemas na transmissão e um com falta de terminadores no segmento amostrado. [2]-Adaptado

2.2.2 Camada de Enlace

A camada de enlace é a segunda camada presente no modelo ISO/OSI, logo acima da camada física. Nessa camada é descrita as regras de transmissão de dados na camada física e como atender cada equipamento ligado à rede [2]. A rede Profibus é uma rede do tipo:

- **Multidrop**: todos os equipamentos do segmento estão conectados no mesmo barramento que passa o sinal elétrico da transmissão. [3]
- **Assíncrona**: não é necessário um cabeamento específico para a sincronização do clock de transmissão entre os equipamentos. [3]
- **Half-Duplex**: É uma rede bidirecional, ou seja, todos os equipamentos podem receber informações e transmitir pela rede, porém ocorre apenas um deles de cada vez, nunca simultaneamente. [3]
- **Mestre-Escravo com passagem de *token***: existem dois tipos de equipamentos ligados na rede: os mestres e os escravos. Os escravos transmitem na rede apenas quando são solicitados por um mestre, sem exceção. Cada mestre possui um tempo de ciclo, no qual ele transmite dados para os escravos associados a ele e recebe de volta os dados requisitados, para então passar o *token* para o próximo

mestre. Essa passagem de *token* também é cíclica, chamada de *Token Ring*, que faz com que assim que o último mestre realiza o seu ciclo ele volta o *token* para o primeiro mestre, assim como exibido na Figura 2-4. [3]

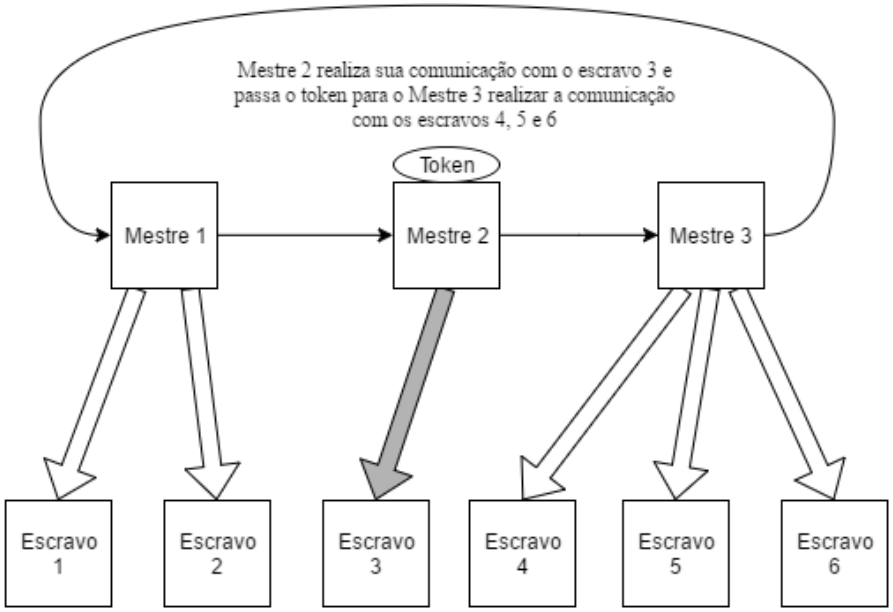


Figura 2-4 - Token Ring representado entre os mestres 1, 2 e 6. [5] Adaptado

Conforme explicado, em um sistema Mestre-Escravo somente o mestre tem a capacidade de iniciar a transmissão na rede. Caso seja feita uma rede somente com equipamentos escravos, ela permanecerá sem comunicação até que um mestre seja incluído. É importante lembrar também que dois equipamentos não podem transmitir ao mesmo tempo. Para evitar transmissões simultâneas entre mestres, por exemplo assim que uma rede é configurada e iniciada, os mestres aguardam um tempo proporcional ao endereço que é atribuído a ele. [3]

Cada equipamento possui um endereço único dentro de uma rede Profibus. A faixa de endereço vai do valor 0 ao valor 127, porém os endereços que podem ser configurados para um dispositivo são da faixa de 0 a 125. O endereço 126 fica reservado para equipamentos que não possuem um sistema mecânico de configuração de endereço, como um *dip-switch* binário ou rotativo. Esses equipamentos possuem o endereço 126 como configuração padrão vinda do fabricante, que deve ser alterado via ferramentas de rede Profibus. O endereço 127 fica reservado para mensagens de *Broadcast*, as quais podem transmitir dados de parametrização e operação da rede, além também de mensagens destinadas a grupos de equipamentos pré-configurados. [3]

O protocolo Profibus define que os dados transmitidos entre equipamentos na rede sejam encapsulados em telegramas. Cada telegrama é composto por um conjunto de caracteres formados por 11 bits cada, sendo 8 deles os bits de dado, 2 para sinalização do início e término do caractere (*start-bit* e *stop-bit*) e um bit de paridade, realizando a primeira checagem de validação do dado transmitido [2]. O modo de transmissão é pelo padrão UART (*Universal Asynchronous Receiver/Transmitter*), sendo que a parametrização de transmissão para a Profibus é:

- 8 bits de dados
- Start-bit com nível lógico 0
- Stop-bit com nível lógico 1
- Paridade Par
- LSB: Transmissão do bit menos significativo primeiro. [3]

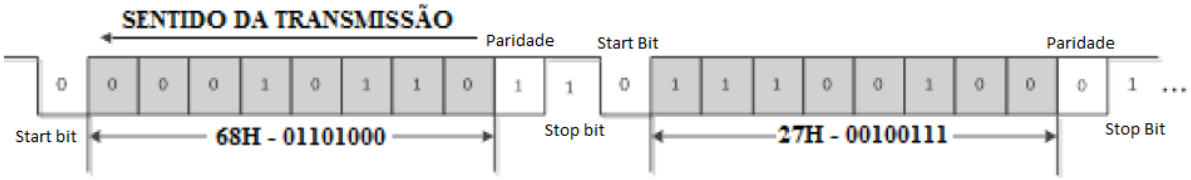


Figura 2-5 - Transmissão dos caracteres na rede Profibus. [6]-Adaptado

Conforme vemos na Figura 2-5, o bit de paridade terá o valor 1 se o número de bits com valor 1 dentro dos 8 bits de dados for ímpar. No exemplo da transmissão do caractere 68h, tem-se o valor 01101000 em binário. Assim, o bit de paridade deve ter o valor 1 para validar um número par de valores 1 no caractere. Já no exemplo da transmissão do caractere 27h, o binário 00100111 já possui um número par de bits em 1, sendo assim o bit de paridade com o valor zero [6]. Caso seja detectado um bit de paridade diferente do esperado, não apenas o caractere será descartado, mas todo o telegrama que ele faz parte. [2]

Antes do início de um telegrama e depois de seu término a rede se encontra em um estado chamado *idle*, ou linha desocupada. Esse estado também é dado em uma rede sem nenhum mestre. Nesse estado, é transmitido sempre o nível lógico 1, cuja tensão é mantida pelos terminadores do segmento. Pode-se observar a diferença da tensão entre a linha desocupada e o início de transmissão de um equipamento na Figura 2-2. Apesar dessa diferença de tensão assim que o equipamento dá a sua entrada na rede, a transmissão de bits 1 não é

interrompida imediatamente, e é dado um tempo para então o *start-bit* de nível lógico 0 começar a ser transmitido. [2]

Um telegrama é composto de 1 a 255 caracteres. Os caracteres são transmitidos seguidos um do outro, não permitindo *idle* na transmissão entre caracteres de um mesmo telegrama. Caso isso ocorra é necessário descartar todo o telegrama. Foi definido pela *Normative Parts of Profibus FMS, DP and PA (1998)* que os telegramas seriam diferenciados pelo seu primeiro caractere, dado pelo cabeçalho do telegrama. São cinco tipos diferentes de telegramas determinados pelos *Start Delimiters*, representados na Tabela 2-2, sendo cada um com a sua função e uma natureza do campo de dados. [2]

O primeiro *Start Delimiter* (SD1, valor 10h) é utilizado para mensagens que não possuem campo de dados, chamada também de mensagens sem campo de dados. Esse tipo de mensagem faz a busca por endereços ativos na rede, sendo assim possível montar a *live-list* com os endereços que respondem a essa mensagem. [7]

O segundo *Start Delimiter* (SD2, valor 68h) define um telegrama com campo de dados de tamanho variável. A maioria dos equipamentos utilizam esse tipo de telegrama para realizar a troca de dados [2]. O byte LE é transmitido após o SD2 e indica o tamanho em bytes do campo de dados da mensagem, que pode assumir um valor de 0 a 249, dado pela quantidade de bytes que do campo DATA_UNIT, que são os dados enviados propriamente dito, o endereço do destinatário do telegrama (DA) e o endereço do remetente do telegrama (SA). [2]

O SD3, com valor A2h não é mais utilizado, porém ele exercia a função da transmissão de telegramas de tamanho fixo com campo de dados. [7]

O *Start Delimiter* SD4, com valor de DCh, é utilizado para a passagem de *token* entre os mestres de uma mesma rede. As informações depois desse caractere são apenas o endereço que irá receber o *token* (DA) e qual endereço está enviando o *token* (SA). Caso uma rede possua apenas um mestre, o valor de SA e DA será o mesmo. [2]

O último *Start-Delimiter* (SC, valor E5h) define um telegrama de reconhecimento ou resposta curta, que é utilizado para equipamentos que necessitam de um tempo maior para processar seus dados antes de enviá-los por meio de um telegrama de campo de dados de tamanho variável. A única informação desse telegrama é o próprio *Start-Delimiter*, que informa que a mensagem foi recebida, porém não será respondida imediatamente. [7]

TELEGRAMA	FORMATO DO TELEGRAMAS
Telegrama de tamanho fixo sem campo de dados	 SD1 – Delimitador de início do tipo 1 (10H). Serviço de manutenção da rede (*Request FDL Status*) e troca de dados quando não há valores de escrita.
Telegrama de resposta curta ou reconhecimento	 SC – *Single Caracter* (E5H). Resposta curta a requisições do mestre onde o escravo não precisa ou não pode responder imediatamente
Telegrama com campo de dados de tamanho variável	 SD2 – Delimitador de início do tipo 2 (68H). A grande maioria dos serviços utiliza esse tipo de comando devido à flexibilidade no tamanho do campo de dados.
Telegrama de *token*	 SD4 – Delimitador de início do tipo 4 (DCH) Utilizado somente para transmissão do *token*. No caso de apenas um mestre na rede: DA=SA.

DA – *Destination Address* – Endereço de destino.
ED – *End Delimiter*, valor 16H
SA – *Source Address* – Endereço de origem.
LE – Tamanho do campo de informação.
FCS – *Frame Check Sequence*
DATA_UNIT – Campo de dados com tamanho variável (1 a 246)
DATA_UNIT8 – Campo de dados com tamanho fixo igual a 8.
FC – *Frame Control*

Tabela 2-2 - Tipos de telegramas do protocolo Profibus. [2]

Nos campos relacionados ao endereço de destinatário e endereço do remetente, DA e SA respectivamente, os sete bits menos significativos são aproveitados para o endereço, cobrindo assim a faixa já especificada de atuação de endereços disponíveis na Profibus, que é de 0 a 127. O bit mais significativo é utilizado apenas para mensagens de tamanho variável, ou seja, mensagens que iniciam com o SD2, conforme vemos na Figura 2-6. Quando o bit mais significativo de DA ou SA está em 1, aparece logo após o FC (*function code*) a extensão do endereço no qual o bit mais significativo estava ativo, ou até mesmo os dois, na ordem de DAE e SAE, Destination Adress Extension e Sender Adress Extension respectivamente. Esses dois

bytes já fazem parte da porção da mensagem chamada DATA_UNIT, que é a carga útil da mensagem propriamente dita, chamado também de *payload*. A Figura 2-7 mostra a estrutura completa de um telegrama do tipo SD2. [2]

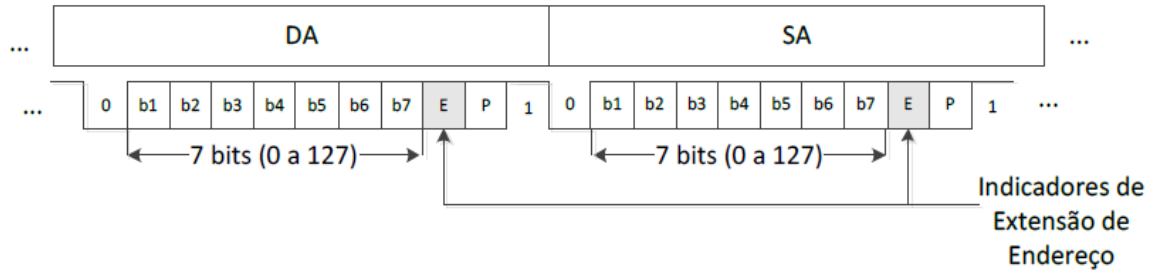


Figura 2-6 - Representação do bit indicador de extensão de endereço. [6]

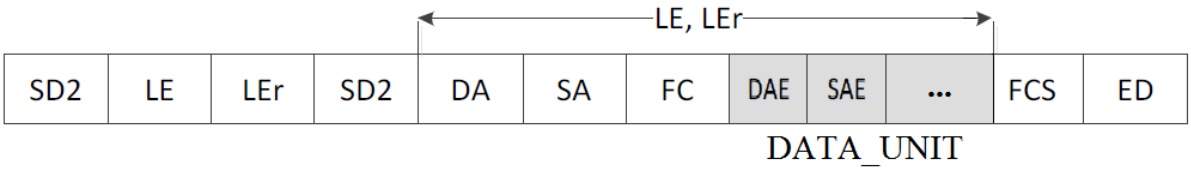


Figura 2-7 - Estrutura de um telegrama SD2 com extensões de endereços. [6] - Adaptado

O uso das extensões de endereço é dado na camada de usuário, não contemplada pelo modelo ISO/OSI. Nessa camada é possível implementar serviços especiais de configuração e parametrização dos equipamentos, cuja função é determinada pelo código de serviço, chamado de *Services Access Points* (SAP). O valor de DAE portanto referencia ao SAP de destino e o SAE referencia ao SAP de origem. Existem valores já pré-definidos para esses serviços, nos quais os equipamentos protocolados para uma certa versão da Profibus deverá responder corretamente quando solicitado um desses SAP. Na Profibus DP (versão DPV0) possuímos os SAP definidos na Tabela 2-3. [2]

Um exemplo do uso do SAP seria um pedido do mestre por um diagnóstico de seu escravo. Em um telegrama SD2 que o mestre envia para o escravo, no campo DA estará presente o valor do endereço do escravo e o bit mais significativo estará com o valor 1. O campo SA irá possuir o endereço do mestre que está enviando a mensagem, também com o bit mais significativo em 1. Logo depois virá o byte que representa o *function code* (FC), que será explicado logo abaixo. Depois do FC, virá o DAE, que é o SAP de destino, com o valor 3Ch.

SERVIÇO	SAP MESTRE	SAP ESCRAVO
Data Exchange	Não possui	Não possui
Set Slave Address	3Eh (62)	37h (55)
Read Inputs	3Eh (62)	38h (56)
Read Outputs	3Eh (62)	39h (57)
Global Control	3Eh (62)	3Ah (58)
Get Configuration	3Eh (62)	3Bh (59)
Slave Diagnostics	3Eh (62)	3Ch (60)
Set Parameters	3Eh (62)	3Dh (61)
Check Configuration	3Eh (62)	3Eh (62)

Tabela 2-3 - Serviços disponíveis na Profibus DPV0. [2]

Depois virá o campo SAE, que é o SAP de origem, contendo o valor 3Eh. Esse conjunto de bytes irá sinalizar o escravo que ele deverá retornar um diagnóstico de seu funcionamento. [8]

O campo FC, ou *function code*, aparece nos telegramas do tipo SD1 e SD2. Ele é responsável por controlar diversos tipos de indicações sobre o telegrama que foi transmitido, como por exemplo sinalizar se a mensagem é de requisição ou resposta, se o equipamento que a enviou é do tipo mestre ou escravo, bits que fazem a verificação se o frame não foi transmitido mais de uma vez e na versão mais nova da Profibus (DVP2) possui até mesmo uma indicação que o frame está sendo utilizado para a sincronização dos clocks dos equipamentos. A estrutura básica do FC está representada na Figura 2-8. [2]

Figura 2-8 - Representação da estrutura do FC. [2]

Na versão DPV2, o bit 8 representado na Figura 2-8 é responsável por indicar que o telegrama transferido faz parte da sincronização dos *clocks* entre os equipamentos. Porém na versão DPV0 e DPV1, que são a Profibus DP e PA respectivamente, esse bit não foi utilizando, sendo seu valor sempre zero [8]. O bit 7 indica se a mensagem é uma requisição ou é uma resposta. Caso indique que o telegrama seja uma resposta, nos bits 5 e 6 serão determinados de acordo com o tipo da estação, com os possíveis valores apresentados na Tabela 2-4. Porém, caso o telegrama seja uma requisição, os bits 5 e 6 são chamados de FCV (*Frame Count Valid*) FCB (*Frame Count Bit*) respectivamente.

O bit FCB é responsável por evitar a recepção de mensagens duplicadas e também pelo reenvio da última mensagem se for detectado que o escravo não respondeu a última requisição feita. Quando um escravo entra em operação, o mestre envia um telegrama para ele contendo o FCB com o valor 1 e o FCV com o valor zero. O escravo que recebeu esta mensagem guarda o valor de FCB igual a 1 e então responde a requisição feita. O mestre recebendo o telegrama pedido escravo, passa a enviar mensagens com FCV com o valor 1, porém sempre alterando o valor de FCB de 1 para 0 e 0 para 1.

Caso por algum motivo o mestre não receba a resposta de um escravo, ele envia novamente o telegrama com o mesmo FCB do último pedido, indicando que foi uma tentativa de *retry*. Se o escravo tiver recebido as duas mensagens (a original e o *retry*), ele deverá responder somente uma vez. Os bits 1 a 4 do FC definem códigos de funções específicas para requisições e respostas, nas quais é possível entre elas um escravo indicar que ele está com um diagnóstico pendente, sinalizando para o mestre requisitar o diagnóstico no próximo ciclo, entre outras funções. [8]

Bit 6	Bit 5	Tipo da estação
0	0	Estação do tipo escravo
0	1	Estação do tipo mestre. Não está pronta para entra no ciclo de *token*.
1	0	Estação do tipo mestre. Está pronta para entra no ciclo de *token*.
1	1	Estação do tipo mestre. Já operando no ciclo de *token*.

Tabela 2-4 - Tipo de estação para o caso de resposta no FC. [2]

Outro campo que aparece nos telegramas SD1 e SD2 é o FCS, chamado de *Frame Check Sequence*. Ele funciona como um *checksum*, onde no telegrama SD1 ele é definido pela soma dos campos SA, DA e FC. No telegrama SD2 ele é formado pela soma de todos os bytes que

compõem o *payload*, ou seja, os que fazem parte da contabilização do tamanho da mensagem (LE). Caso a soma exceda a capacidade de um byte (overflow), apenas a parte baixa da soma é utilizada. Se o equipamento que recebeu a mensagem verificar que a soma resultou em um valor diferente do recebido pelo campo FCS, o telegrama é descartado inteiro. [8]

O último campo que aparece nos telegramas SD1 e SD2 é o ED, nomeado *End Delimiter*. Esse campo é de valor constante 16h e apenas sinaliza que o telegrama acabou de ser transmitido. Logo após um ED sempre há um tempo *idle* na rede. [8]

2.3 Diferenças entre versões da Profibus

A Profibus atualmente conta com três versões principais, além de vários *profiles* que introduzem funções específicas para cada ramo de atuação da rede, como por exemplo o *ProfiEnergy* [9], que conta com funções para controle da alimentação de energia de equipamentos para realizar desligamentos ou ligamentos pré-definidos em horários de almoço, feriados, além de obter medidas sobre o consumo. Outro *profile* existente na Profibus é o *ProfiSafe* [10], que é especializado para equipamentos relacionados a segurança dos processos, como botões de emergência, cortinas de luz e outros tipos de sensores. O *profile ProfiDrive* [11] possui funções para a sincronização de motores a partir de *encoders* e também uma grande aplicação no controle desses.

Esses *profiles* estão na camada de usuário, não contemplada pelo modelo ISO/OSI. Na camada de aplicação desse modelo estão as versões da Profibus. Atualmente existem três versões, são elas:

- DPV0: Versão contemplada pelo padrão Profibus DP. Possui funções básicas de troca de mensagens e alguns serviços, conforme listados na Tabela 2-3. [2]
- DPV1: Versão contemplada pelo padrão Profibus PA. Mais utilizado em indústrias de processos, farmacêuticas, tratamento de água, entre outros, no qual o protocolo possui mais funções de configuração e parametrização, alarmes, além da transmissão com segurança intrínseca, adquirindo a possibilidade de ser utilizada em áreas classificadas. [2]

- DPV2: Versão da Profibus que contempla a sincronização dos *clocks* entre todos os equipamentos, utilizado para fazer o controle de motores juntamente com *encoders*. [1]

Todas as versões possuem a retrocompatibilidade, ou seja, um dispositivo da Profibus PA conseguirá comunicar e trocar dados normalmente com um dispositivo da Profibus DP, não conseguindo apenas acessar os serviços específicos da versão DPV1, assim como dispositivos DPV2 conseguem comunicar normalmente com dispositivos DPV1 e DPV0. [3]

2.4 Arquivo GSD

Todos os equipamentos aprovados pelos testes dos laboratórios da *Profibus International* ganham um *Identity Number* único, ou ID, que identificam o modelo do equipamento. Esse ID é enviado em mensagens do tipo SD2 que possuem dados de diagnóstico e parametrização. Para o mestre ter certeza que o equipamento ajustado no projeto para um dado endereço é o mesmo fisicamente instalado, durante a parametrização do equipamento ou algum diagnóstico esse ID é comparado com o ID referente ao equipamento do projeto. [3]

Os programas que fazem o projeto da rede precisam de um arquivo GSD (*General Station Description*), que podem vir com outras extensões como .gse, .gsf, .gsg, .gsi, .gsp e .gss, que possuem a mesma função do .gsd porém em linguagens diferentes como inglês, francês, alemão, italiano, português e espanhol respectivamente. Esses arquivos são como um *datasheet* eletrônico do equipamento, com a função de informar ao programa responsável pelo projeto da rede qual o ID do equipamento, seu nome, modelo, propriedades de projeto para ele (como a possibilidade ou não de operação em certos *Baud Rates*), além de informações sobre o campo de diagnóstico, onde cada conjunto de mensagens é traduzido em uma *string* que contém a descrição do que está ocorrendo com o equipamento. [3]

Outros equipamentos que utilizam o GSD para a identificação de cada estação são os analisadores de rede, que por meio da identificação do endereço que enviou um diagnóstico ele faz a busca pela biblioteca de GSD disponibilizada pelo usuário e consegue assim exibir as informações do equipamento em uma *live list*, além de relatar a mensagem de erro relacionado ao diagnóstico enviado pelo equipamento. [3]

2.5 Equipamentos de Análise de uma rede Profibus

Existem vários métodos para realizar a manutenção e o diagnóstico em uma rede Profibus. O usuário responsável por essa tarefa deverá estar ciente que muitas vezes é necessário a utilização de mais de um método para ter certeza de qual problema está acontecendo. Nessa seção serão abordados apenas os métodos mais utilizados e também suas principais características, todos referenciados à tese *Diagnóstico Automático de redes Profibus* (MOSSIN, 2012), sendo possível obter mais informações de equipamentos de mercado e suas funcionalidades por ele.

2.5.1 Inspeção Visual

O responsável pela manutenção percorre toda a instalação física da rede observando falhas como:

- Raio mínimo de curvaturas do cabo Profibus abaixo do recomendado pelo fabricante.
- Cabeamentos maiores que o estabelecido pela norma em relação ao *Baud Rate*, conforme Tabela 2-1.
- Todos os dispositivos devem suportar o *Baud Rate* configurada no projeto.
- Malha de blindagem do cabo com partes externas ao conector.
- Falta de aterramento dos dispositivos ligados à rede.
- Passagem do cabo Profibus em canaletas recomendadas, respeitando o limite mínimo de distâncias entre cabos de potência. Problema exibido na Figura 2-9.
- Falta de terminadores no começo e fim de cada segmento.

2.5.2 Multímetro

Utilizando um multímetro padrão é possível verificar problemas principalmente com o cabeamento. Todavia, o uso deste equipamento não é uma tarefa simples, pois necessita de conhecimento do usuário e demanda tempo. Esse método é utilizado com a rede desligada, geralmente na fase de montagem dos conectores:

- Curto-circuito entre as linhas A e B.
- Curto-circuito de alguma linha (A ou B) com a malha de blindagem do cabo.
- Inversão das linhas na montagem dos conectores.
- Falta de contato entre a malha de aterramento e o conector.
- Verificação se os parâmetros de resistência, indutância e capacitância dos cabos estão dentro das faixas permitidas pela norma.

Figura 2-9 - Exemplo de cabos Profibus próximos a cabos de potência. [12]

2.5.3 Testadores de barramento portáteis (*Bus testers*)

Os testadores de barramento, conforme mostrado na Figura 2-10, realizam algumas funções que foram descritas no multímetro, além de outras que são específicas da rede quando em funcionamento, como a identificação de reflexão de ondas por falta de terminadores. Geralmente também são utilizados na fase de montagem dos conectores, para as seguintes funções:

- Identificar perda de terminação ou mal funcionamento desse componente em um conector instalado.
- Identificar reflexões causadas não somente pela falta de terminação do cabeamento, mas também por algum problema de fabricação no interior de cabo ou um possível dano na manipulação dele.
- Realiza medidas nos níveis de tensão entre as linhas A e B para garantir os mesmos parâmetros entre os dois.
- Os que permitem a operação com a rede em funcionamento, é possível realizar medidas para testar os drivers de EIA-485 de cada equipamento verificando o comportamento do sinal transmitido, além de listar os endereços dos equipamentos ligados na rede, chamada de *live list*.

Figura 2-10 – Modelo de um testador de barramento portátil comercial. (TS Testador Profibus/FF Toledo & Souza). [13]

2.5.4 Osciloscópio

Poderá ser utilizado um osciloscópio para analisar a forma de onda do sinal que percorre nas linhas A e B da rede Profibus em funcionamento para identificar os seguintes problemas:

- Reflexão nas linhas A e B, seja por falta de terminador nas extremidades do segmento ou por algum dano no cabeamento.
- Curto-circuito entre as linhas ou entre uma linha e a malha de aterramento.
- Número de equipamentos acima do permitido por segmento.
- Cabeamento maior que o permitido pela norma em relação ao *Baud Rate* escolhido no projeto da rede.
- Interpretação dos caracteres e formação dos telegramas.

Toma-se o cuidado de isolar o pino de terra do osciloscópio do terra da rede de alimentação para não aterrar a referência do canal e consequentemente, aterrar um dos condutores de sinal da rede Profibus.

2.5.5 Analisadores de Rede

Os analisadores de rede Profibus são equipamentos que atuam na rede já em operação, porém não necessariamente em funcionamento correto. A maioria dos equipamentos comerciais possuem softwares que são instalados em computadores, que coletam as informações por meio de um driver que comunica com uma porta externa do computador, sendo a USB a mais usual.

Esses equipamentos possuem diversas funções que abrangem a possibilidade de fazer toda a análise de problemas dos equipamentos já citados acima. São elas:

- Captura dos telegramas enviados pela rede, os quais são identificados os equipamentos que estão na *live list* da rede por meio de seus GSDs, permitindo que os diagnósticos sejam interpretados pelo próprio programa e exibido ao usuário. Caso a rede esteja em funcionamento correto, é possível visualizar os telegramas contendo os dados de entradas e saídas trocadas entre os mestres e seus escravos.
- Testes de qualidade do sinal enviado por cada equipamento presente na rede a partir de um ponto dela. Com essa função do equipamento é possível descobrir o trecho do cabeamento que possui alguma falha estrutural ou descobrir falha do driver EIA-485 de algum equipamento da rede.
- Captura de amostras analógicas, para serem exibidas em uma tele similar à de um osciloscópio. Por meio dessa função é possível identificar os problemas que

são possíveis na identificação por meio de um osciloscópio. A Figura 2-11 mostra a interface de um dos modelos disponíveis no mercado.

- Análise do uso do barramento, mostrando por meio de gráficos qual a relação e entre *idle* e transmissão na rede, além também da relação do uso de cada versão Profibus (DPV0, DPV1 e DPV2).
- Estatísticas sobre os telegramas recebidos, como o número de *retries* ocorridos durante a captura, número e informações de diagnósticos e telegramas corrompidos.

Figura 2-11 - Interface do analisador de rede ProfiTrace. [14]

2.5.6 Monitores de rede e Ferramentas de Engenharia

Os monitores de rede e as ferramentas de engenharia são equipamentos ligados temporariamente ou permanentemente em uma rede Profibus, que podem ou não exercer a função de comunicação dentro da rede. Quando o equipamento comunica na rede, ele entra na fase acíclica da comunicação, realizando assim as operações requisitadas pelo o usuário, como

troca de endereço de um equipamento que não possui um sistema mecânico para realizar essa troca, ou outros serviços que podem ser realizados dentro da rede, listados na Tabela 2-3. Este equipamento é chamado de ferramenta de engenharia.

Quando o equipamento apenas coleta as informações da rede ele é chamado de monitor de rede. Esse monitor pode exercer funções como a montagem da *live* list e a criação de um *log* de diagnósticos e estatísticas que aconteceram na rede. Um modelo de mercado deste equipamento pode ser visto na Figura 2-12.

Figura 2-12 - Monitor de rede TH-Link Profibus. [13]

3 BEAGLEBONE

O hardware escolhido para cumprir o objetivo do projeto foi o computador de placa BeagleBone Black Rev. C. Essa placa foi desenvolvida pela *BeagleBoard.org Foundation*, a partir do projeto inicial chamado BeagleBoard, que teve a intenção de ser utilizado com fins educacionais em universidades, sendo inteiramente *open-source* [15]. A partir da *Beagleboard* foi então criado nos mesmos moldes a BeagleBone, em outubro de 2011. Em 2013 foi lançada uma nova versão, a BeagleBone Black, a qual no ano de 2014 já possuía a terceira versão, titulada como *Revision C*. [15]

Será descrito nesse capítulo as características principais da placa utilizada, enfatizando as funções que foram usadas para a realização do projeto.

3.1 Hardware da BeagleBone Black

A BeagleBone Black, exibida na Figura 3-1, possui as dimensões de 8,63cm x 5,46cm x 1,6cm e pode ser alimentada por uma fonte externa de 5V ou por meio de um cabo mini USB ligado a um computador. Ela necessita de uma corrente de até 500mA para o funcionamento sem nenhum equipamento ligado à sua porta USB. Caso seja necessário o uso de um equipamento ligado a essa porta, é obrigatório o uso de uma fonte externa de 5V que suporte no mínimo o fornecimento de 1A de corrente. [16]

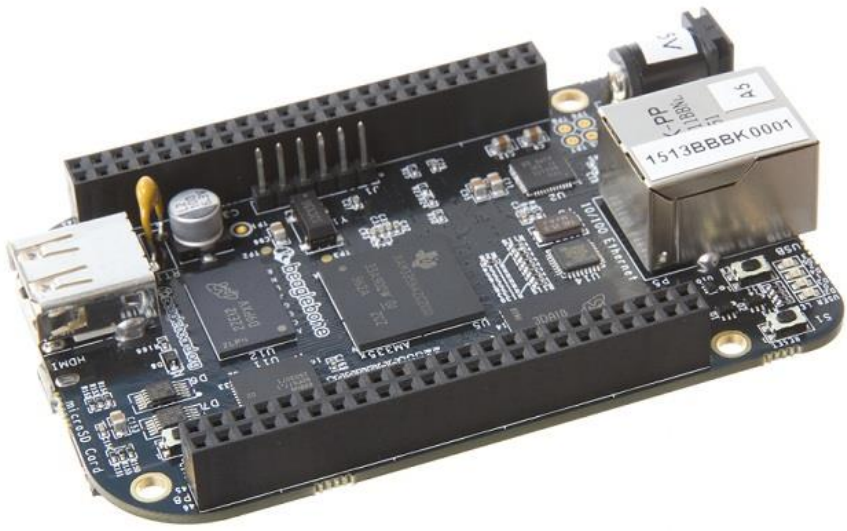


Figura 3-1 - BeagleBone Black Rev. C. [17]

A placa conta com um processador AM335x Sitara ARM Cortex-A8, fabricado pela *Texas Instruments*, 512MB de memória RAM, um eMMC (*Embedded Multimedia Card*) de 4GB de armazenamento, com um Linux Angstrom totalmente compatível com a placa já armazenado nessa memória FLASH nas configurações padrão de fábrica. A placa ainda conta com periféricos como porta USB que pode agir como *host*, uma saída micro HDMI, um conector de rede Ethernet RJ-45, um slot para cartão micro SD, que possibilita aumentar o armazenamento, além de dois *GPIO Expansion Headers*, que possuem conexão com 46 pinos cada, totalizando 96 pinos utilizáveis na placa. [16]

Cada pino possui uma função, como saídas de tensão 3.3V, 5V e 5V direto da fonte externa, chamado de 5V_RAW e o GND do circuito da placa. Dos 96 pinos, 65 são pinos de GPIOs, abreviação para *General Purpose Input/Output*, nos quais a maioria pode exercer mais de uma função. Quando isso acontece, é necessário criar uma configuração de multiplexação para esse pino, selecionando a saída para a função desejada que ele exerça. Oito pinos são de entradas analógicas, ligados a um conversor AD que suporta tensões de até 1.8V. A Figura 3-2 representa a disposição dos pinos com as suas principais funções secundárias e terciárias. [16]

3.2 Software da BeagleBone Black

O computador de placa BeagleBone Black acompanha com a sua configuração de fábrica o sistema operacional Angstrom, que é uma distribuição livre do Linux. Essa versão é compilada para processadores ARM e possui algumas características que permitem a comunicação do sistema operacional com os pinos de GPIO, além de seus outros periféricos. É possível utilizar na placa outro sistema operacional, como outras distribuições do Linux, como exemplo o Fedora, Ubuntu, Debian e até mesmo Android, também oficialmente compiladas para a versão ARM do processador da placa e também possuem a capacidade de comunicação com os pinos GPIO da placa. [18]

A linguagem de programação oficial para a BeagleBone é o BoneScript, que é uma biblioteca para o NodeJS, um interpretador da linguagem JavaScript. Com o BoneScript é possível realizar tarefas simples como obter o valor de entrada de um pino, ou definir seu estado de saída, até tarefas mais complexas, como a criação de *webservers* que facilitam principalmente o desenvolvimento para as aplicações de Internet of Things. [16]

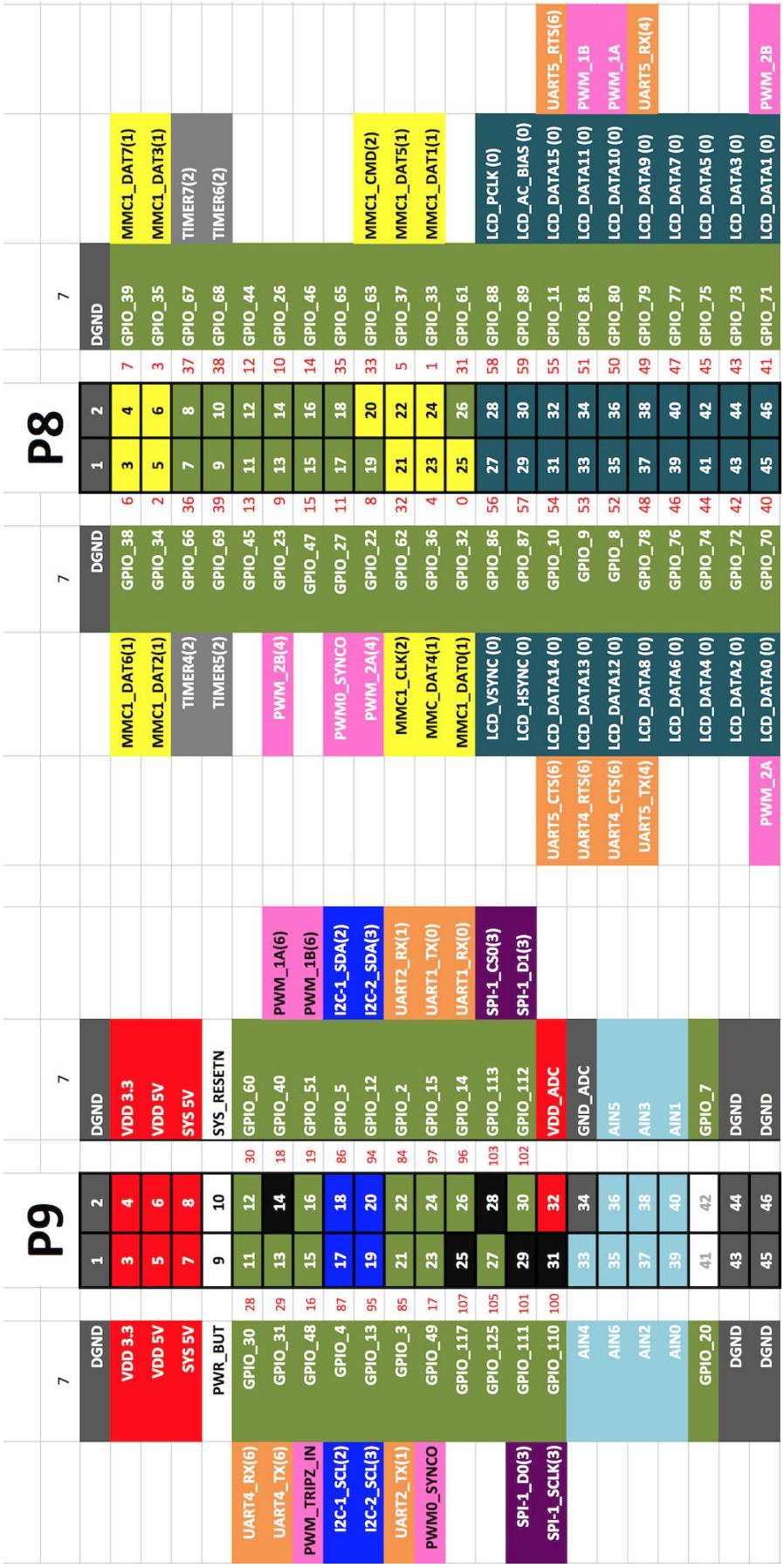


Figura 3-2 - Disposição espacial dos pinos da BeagleBone e suas principais funções. [19]

Como é utilizado um sistema Linux para a execução e controle da placa, o usuário não fica preso apenas na linguagem BoneScript. É possível programar e executar softwares em qualquer linguagem compilável para Linux, contanto que seja adequadamente compilada para a versão ARMv7. Muitos usuários também programaram bibliotecas em diversas linguagens que permitiram que a iteração entre *software* e GPIOs não dependesse somente do BoneScript, como por exemplo bibliotecas para C, C++, Java, Python, Qt, entre muitas outras. [20]

Um problema que aparece porém nas programações em qualquer linguagem que dependa do sistema Linux é a sincronia exata com os pinos de GPIO e também da falta de precisão dos temporizadores (*timers*). Um sistema operacional fica encarregado de realizar várias tarefas ao mesmo tempo, porém a arquitetura do processador permite que apenas uma tarefa seja executada por vez. Isso faz com que quando um programa executa uma função para esperar um determinado tempo, esse tempo nunca será preciso, pois o sistema operacional pode demorar mais para verificar se a espera já foi concluída ou não. O erro relativo dessa espera aumenta inversamente proporcional ao tempo de espera, ou seja, em funções que determinam um *delay* de alguns microssegundos causará um erro relativo muito maior do que funções que geram *delays* de segundos. Para contornar esse problema é necessário utilizar subsistemas que estão presentes dentro do processador AM335x. [20]

3.3 Subsistemas do processador AM335x

O processador AM335x é produzido pela *Texas Instruments* e de acordo com seus manuais e referências ele é um processador voltado para aplicações na área industrial. Além de possuir o núcleo ARM principal no qual o sistema Linux opera, dentro dele há vários subsistemas que operam independente do sistema operacional ou dos comandos do ARM. Alguns desses principais subsistemas que foram utilizados para esse projeto e que serão explicados mais detalhadamente são o PRU-ICSS, Timers, eCAP e ADC. Os outros sistemas e uma ideia básica de como eles estão dispostos dentro do processador estão representados na Figura 3-3. [21]

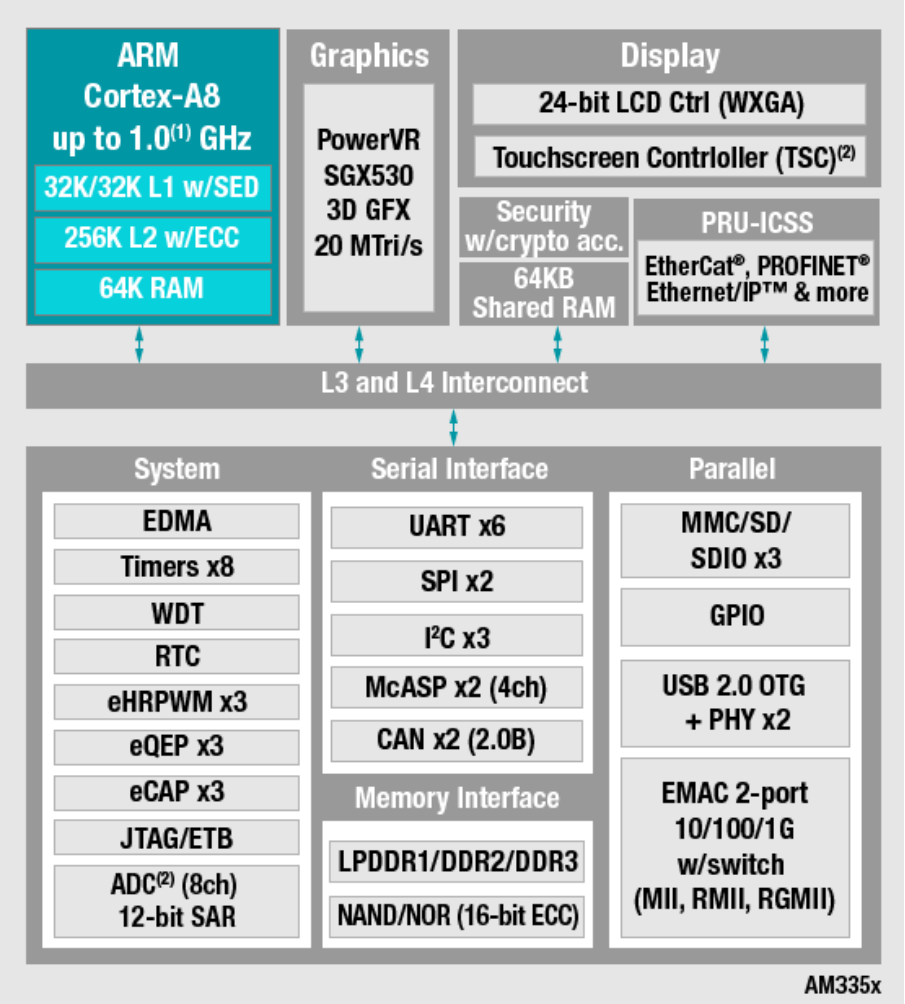


Figura 3-3 - Organização dos subsistemas disponíveis dentro do processador AM335x. [21]

3.3.1 Programmable Real-Time Unit Subsystem (PRU-ICSS)

O primeiro subsistema descrito será o mais importante para a realização deste trabalho. Os núcleos chamados PRU-ICSS, abreviação para *Programmable Real-Time Unit Subsystem and Industrial Communication Subsystem*, ou simplesmente PRU, são núcleos de processamento independente do núcleo principal (ARM) do processador. Dentro do processador AM335x há dois núcleos PRU, onde cada um consiste em um núcleo RISC de 32 bits, cada um com a sua memória de programa de 8KB, uma memória de dados de 8KB e acesso a uma área de memória compartilhada chamada SHARED RAM de 12KB. Há instruções que acessam a memória de dados do outro núcleo PRU, com permissão de leitura e escrita. Eles possuem uma conexão com os barramentos principais do processador, o L3 e L4, representados

também na Figura 3-3, tornando possível a escrita e leitura também na área de memória do núcleo ARM e vice-versa. Essas memórias, subsistemas e barramentos ligados ao PRU-ICSS podem ser visualizadas na Figura 3-4. [22]

Os núcleos PRUs possuem um clock próprio de 200MHz. Devido a arquitetura RISC abordada neles, a maioria das instruções que o PRU é capaz de realizar é executada com apenas um ciclo de clock. As exceções são algumas operações que envolvem leitura e escrita de memória, que demoram 2 ciclos para as operações nas memórias de dados das próprias PRUs, porém para a leitura e escrita na região de memória que depende do barramento L3 e L4, o tempo de execução não pode ser determinado, pois depende do nível de tráfego no barramento, que possui diversas funções dentro do processador. [22]

Cada núcleo PRU possui 32 registradores de uso geral, numerados de R0 a R31. Cada um desses registradores possui 32 bits, que podem ser divididos em 4 bytes ou 3 words. Cada byte possui 8 bits, com o byte0 abrangendo do bit 0 ao bit 7, o byte1 abrangendo do bit 8 ao bit 15, e assim por diante. Os words possuem o tamanho de 16 bits, com o word0 iniciando no bit 0 e terminando no bit 15, o word1 começando no bit 8 e terminando no bit 23 e o word2 iniciando no bit 16 e terminando no bit 31, conforme mostrado na Figura 3-5. [20]

Os dois últimos registradores de uso geral, o R30 e R31, possuem funções especiais como controlar o estado de saída dos pinos GPIO, realizar leituras nesses pinos, além de possuírem bits que estão ligados diretamente ao sistema de controle de interrupções, o INTC. Há uma área de memória compartilhada entre os dois PRUs chamado de *Scratch Pad*, que permite a troca dos valores desses 32 registradores de uso geral, trocando apenas o banco utilizado por meio da configuração de certos registradores. [22]

3.3.2 Interrupt Controller (INTC)

Os PRUs possuem uma conexão com um controlador de interrupções chamado de INTC (*Interrupt Controller*). Esse controlador é responsável por receber interrupções vindas de outros subsistemas do processador e também enviar e receber interrupções para o núcleo ARM do processador, chamado de host. Não há interrupções assíncronas na PRU, somente por *pooling*. Em processos de interrupção por *pooling* o processador deve verificar periodicamente se houve alguma requisição de interrupção, sendo que essa verificação deverá ser programada pelo projetista do software que executará na PRU. A diferença para uma interrupção assíncrona seria

que assim que uma flag de interrupção fosse gerada, o processador interrompe a execução do código em que ele está, redireciona o ponteiro de programa para a região determinada para a execução da interrupção e então quando finalizada ele voltaria a execução do código que estava sendo feita antes da interrupção. Para a máxima garantia de precisão no tempo de execução de acordo com o que foi programado pelo usuário, o método de interrupção *pooling* foi adotado no PRU. Para tornar essa verificação mais eficiente, o INTC faz a configuração das interrupções que o usuário deseja habilitar em duas etapas: canais (*channels*) e *hosts*, conforme mostrado pela Figura 3-6. [22]

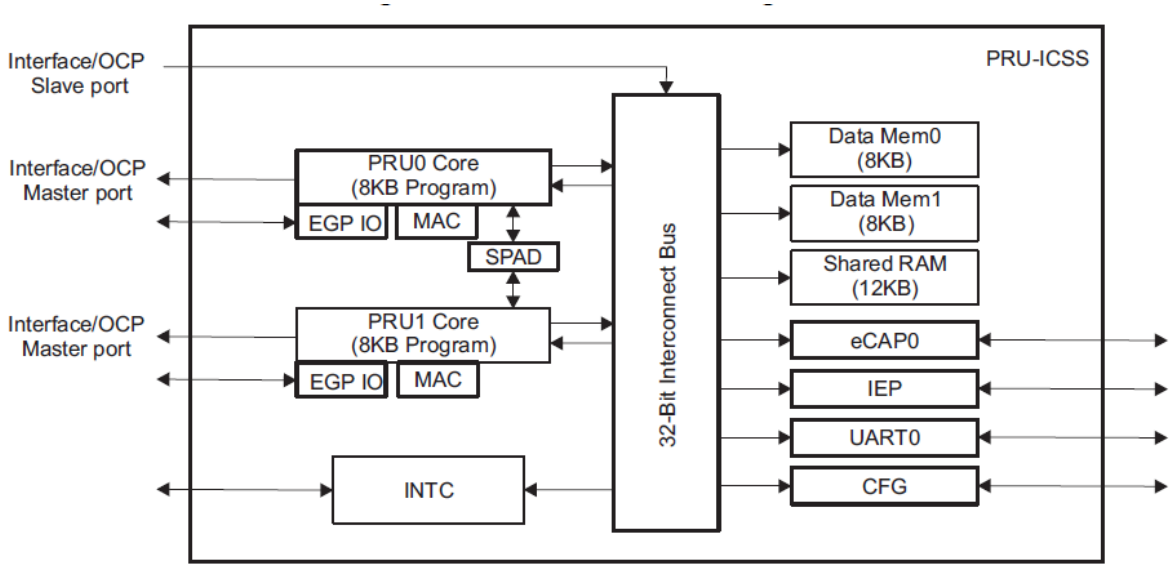


Figura 3-4 - Diagrama de bloco do PRU-ICSS. [22]

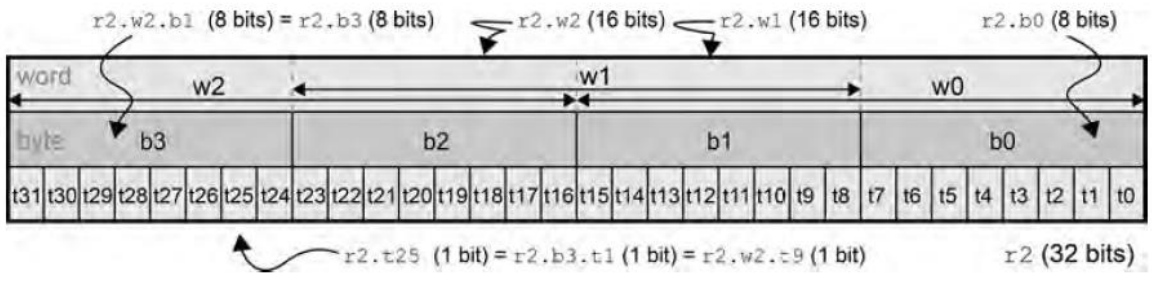


Figura 3-5 - Maneiras de acessar os bits, bytes e words dentro de um registrador de uso geral. [20]

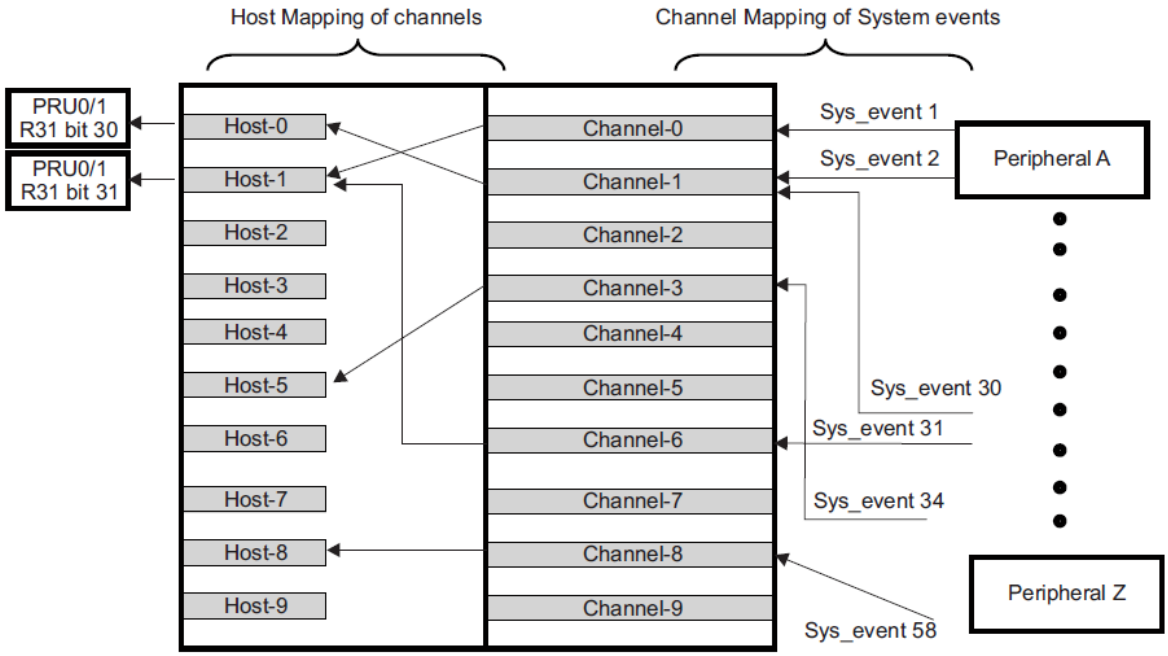


Figura 3-6 - Funcionamento do Interrupt Controller (INTC). [22]

O usuário possui a capacidade de configurar por meio de registradores do INTC quais eventos do sistema serão ligados a quais canais, com a numeração de 0 a 9. Esses eventos dos sistemas são flags de interrupções geradas pelos subsistemas presentes dentro do processador AM335x, sendo identificadas por um número de evento de interrupção, como por exemplo os módulos de UART (sendo a UART0 com o número de evento de interrupção 51, a UART 1, UART 2 e UART 3 com número de evento de interrupção de 32, 52 e 6 respectivamente). A tabela com todos os 64 números de eventos de interrupção pode ser encontrada na página 227 do documento *AM335x Sitara Processors – Technical Reference Manual* (2011). O usuário sabendo qual subsistema ele quer checar por interrupções, ele realiza a configuração de algum dos 10 canais do INTC para receber as informações sobre aquele número de evento de interrupção. Um canal pode receber mais de um número de evento de interrupção. Logo após a configuração do canal do *Interrupt Controller*, é necessário configurar qual *host* esse canal irá ativar caso alguma interrupção configurada para ele for identificada. Os *hosts* também são numerados de 0 a 9, que fazem a coleta de informações dos canais configurados para eles. Os *hosts* 0 e 1 estão ligados aos bits 30 e 31 do registrador de uso geral R31 de cada núcleo PRU, enquanto os demais hosts podem ser configurados para ativar interrupções no núcleo ARM do processador [22]. Abaixo segue os passos para a configuração de uma interrupção dentro do PRU:

- O usuário deve ativar as interrupções do subsistema que deseja utilizar nos registradores dele. Exemplo: ativar a interrupção de recebimento de um novo dado e a ativar a interrupção da conclusão de envio de um dado na UART1.
- Configurar o canal desejado que receba as interrupções desse subsistema por meio do número de evento de interrupção. Exemplo: configurar que o canal 1 recebe os eventos da UART1. Portanto será configurado que o canal 1 terá o número de evento de interrupção configurado para 32.
- Configurar qual host será ativado por um determinado canal. Exemplo: configurar que o host 0 será ativado pelo canal 1.
- Ativar o registrador que habilita o funcionamento do INTC, chamado de *Global Enable Register* (GER).

Realizando os passos de configuração acima, as interrupções desejadas pelo usuário já estão ativas e funcionando. Durante a execução do programa, o usuário deverá programar passos que fazem a conferência dessas interrupções, para determinar se alguma está ativa, chamado também de *pooling* de interrupção [22]. Os passos para descobrir qual interrupção foi ativa são:

- Verificar se os bits 30 ou 31 do registrador R31 está com o valor 1. Caso positivo, significa que uma interrupção foi gerada nos canais ligados ao *host* 0 ou *host* 1 respectivamente. Exemplo: no caso da configuração da UART1 feita acima, deveremos verificar o estado do bit 31 do registrador R31.
- Se o usuário tem certeza de evento de interrupção geraria essa interrupção do host que foi ativado, ele pode buscar diretamente os detalhes da interrupção no evento configurado. Porém se mais de um canal gerar interrupção em um host, é necessário ler qual foi o valor do número de evento de interrupção responsável pela ativação no registrador *Global Prioritized Index Register* (GPIR). Exemplo: no caso configurado, apenas o subsistema da UART1 geraria uma ativação do *host* 1. Portanto, se no *pooling* dentro do programa for identificado uma ativação no *host* 1, poderá ser consultado diretamente o registrador de interrupção do subsistema.
- Encontrado qual subsistema gerou a interrupção no *host* verificado, o usuário verifica dentro dos registradores de interrupção do subsistema qual foi o responsável por gerar o evento de interrupção do subsistema, direcionando assim o seu código para a execução do trecho responsável pelo processamento daquela

interrupção. Exemplo: o usuário lê o registrador de status de interrupção da UART1 e verifica que o bit correspondente ao recebimento de um novo dado está ativo. Portanto seu código irá realizar uma sub-rotina para processar esse dado que foi recebido.

Acima estão os passos básicos para a configuração e identificação de interrupções no sistema da PRU por meio de *pooling*. Existem registradores mais complexos no INTC que realizam tarefas como priorizar certos canais a outros, criando uma espécie de empilhamento de interrupções de acordo com a prioridade configurada, chamado *nesting*. Além disso, a cada interrupção detectada no ciclo de *pooling* é necessário limpar o estado da interrupção correspondente, para que no próximo ciclo ela não seja detectada novamente. [22]

3.3.3 Timers

O processador AM335x possui um subsistema chamado de *Industrial Ethernet Peripheral*, ou IEP, que realiza trabalhos necessários para a ligação de uma rede *Ethernet* industrial, como *Profinet*, *EtherCAT* ou até o mesmo o *Ethernet*/IP por meio da ligação dos cabos de conexão de até 8 pinos GPIO ligados ao IEP. Porém, para a realização desse trabalho não foram utilizadas essas funções principais desse subsistema, mas apenas os *timers* (ou temporizadores) que são capazes de gerar interrupções em certos eventos de comparação. [22]

O clock que é ligado ao subsistema IEP tem base no clock gerado para a PRU, de 200MHz, podendo ser configurado para ter esse valor dividido em múltiplos de dois. A configuração padrão de operação é usar sem divisor de clock, fazendo assim que o valor do contador desse timer (registrador COUNT) incremente um valor a cada borda de subida do clock. Isso nos dá uma precisão de 5 nanossegundos utilizando o clock de entrada sem divisão em relação ao clock do PRU. Esse timer possui 32 bits de capacidade de contagem, gerando nessas condições uma interrupção de overflow a cada 21,47 segundos. [22]

É possível também configurar oito interrupções nesse timer por valores de comparação, os quais são configurados a partir dos registradores CMP0 a CMP7. Ao atribuir um valor de 32 bits em um desses registradores e ativar o registrador de interrupção do IEP, toda vez que o valor de COUNT for igual ao valor do CMP configurador causará uma interrupção no subsistema IEP. [22]

3.3.4 Enhanced Capture Module (eCAP)

Outro subsistema presente dentro do processador AM335x é o *Pulse-Width Modulation Subsystem* (PWMSS), que possui três módulos no interior desse subsistema: Enhanced PWM Module (ePWM), Enhanced Capture Module (eCAP) e Enhanced Quadrature Encoder Pulse Module (eQEP). [22]

Os registradores desse subsistema são compartilhados entre os módulos, portanto é possível ativar somente um dos módulos por vez no PWMSS. No primeiro citado, o ePWM, é possível configurar os registradores para gerar pulsos de saída nos pinos GPIO ligados ao PWMSS que geram PWMs com precisão de 5 nanossegundos, sendo capaz de operar em frequências de até 1.25MHz. O segundo módulo, eCAP, será mais detalhado na explicação abaixo devido ao uso dele no projeto. O terceiro módulo, eQEP, possui a capacidade de ler a entrada de dois GPIO e em um ciclo de clock calcular várias informações sobre os dois sinais, como o período do PWM gerado em cada porta e a sua razão cíclica, além de realizar operações entre os dois sinais para permitir o uso de *encoders* de disco ótico, que fornece informação de velocidade do eixo do motor ao qual está acoplado, além também da posição em ângulo em relação a um referencial de ângulo zero. [22]

O segundo módulo comentado no parágrafo acima, o eCAP, fica encarregado de realizar a contagem em um timer de 32 bits exclusivo desse subsistema (não é o mesmo timer do subsistema IEP) e por meio de eventos pré-definidos do sinal ligado ao pino de entrada do eCAP ele realiza a captura do valor do timer e salva em um dos registradores de captura, que vai do CAP1 ao CAP4.

O registrador do *timer* que realiza a contagem do tempo nesse subsistema é chamado de *Time-Stamp Counter Register* (TSCTR). Assim como no timer do IEP, esse timer é alimentado pelo clock da PRU de 200MHz, sendo possível aplicar divisores múltiplos de 2 nele. Se nenhum valor de divisão for utilizado, a cada 5 nanossegundos uma unidade será acrescida no TSCTR. Por meio de registradores de configuração desse módulo é possível definir os eventos que realizarão a captura do tempo que está sendo contado pelo TSCTR e também qual o modo de operação: em tempo absoluto ou tempo relativo. [22]

Os eventos possíveis para ser configurado são a borda de subida ou a borda de descida do sinal. É atribuído a cada registrador CAPx um evento, que realizará a captura de tempo e salvará o resultado do modo escolhido nesse CAPx assim que o evento acontecer. É possível

configurar que assim que o valor de um desses registradores for alterado (CAP1 a CAP4), uma interrupção será gerada por esse subsistema. Outra interrupção que pode ser habilitada é o overflow do registrador TSCTR. [22]

O modo de operação do eCAP também é definido via configuração a partir de registradores. No modo de tempo absoluto, o valor salvo nos registradores de captura (CAPx) é o valor presente no próprio registrador TSCTR. Combinando essa função com a interrupção de overflow é possível criar um sistema de *time-stamp* com contadores de 64 bits, que levariam mais de 2900 anos para serem completamente preenchidos. Se o modo de operação de tempo relativo for escolhido, sempre que um evento for gerado o valor salvo no registrador CAPx atual é a diferença entre o valor de TSCTR com o valor salvo no CAPx anterior, pois o valor de TSCTR é zerado na ocorrência de um evento. Ou seja, é um modo de operação que salva quando tempo levou entre um evento configurado e outro. [22]

A Figura 3-7 mostra a operação em modo de tempo absoluto, onde todos os CAPx foram configurados para os eventos de borda de subida. Ou seja, no sinal ligado ao GPIO de entrada, sempre que houver uma borda de subida os CAPx irão sendo preenchidos, gerando interrupções para o programa coletar essas informações no momento correto.

Na Figura 3-8 temos o modo de operação em tempo relativo, com todos os eventos configurados para a captura da borda de subida. Notamos que a diferença em relação ao tempo absoluto é que o registrador TSCTR é zerado a cada evento, já salvando o valor de tempo que decorreu entre os dois eventos. Esse modo de operação é usado para a obtenção de períodos diretamente, sem se preocupar em realizar a diferença de *time-stamps* para obtê-los.

3.3.5 Analog-to-Digital Converter (ADC)

O processador AM335x conta com um subsistema que realiza amostragens analógicas e as convertem em valores digitais. Esse subsistema é chamado de *Touchscreen Controller and Analog-to-Digital Converter Subsytem* (TSC_ADC_SS). Esse módulo possui funções que permitem a ligação de telas que possuem a função de detecção de toque por meio de uma malha resistiva, podendo ser de modelos que utilizam 4, 5 ou 8 fios para a transferência de dados. Porém não é necessário utilizar esse módulo apenas como um controlador para esses tipos de tela, podendo ser utilizado para amostragem de sinais analógicos genéricos, desde que fiquem dentro da faixa de operação do conversor. [22]

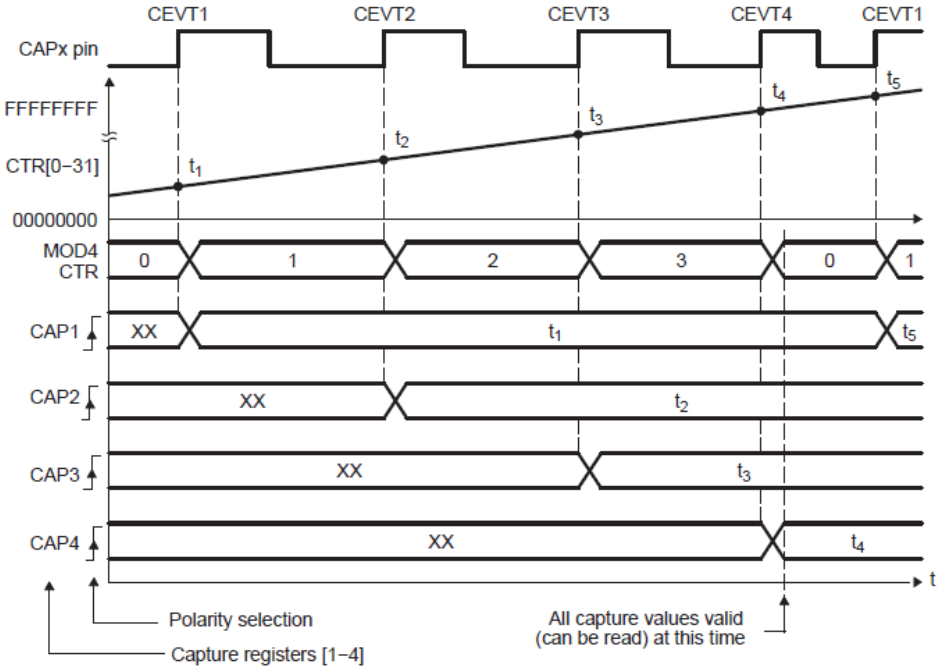


Figura 3-7 - eCAP com modo de operação em tempo absoluto e todos es eventos configurados para borda de subida. [22]

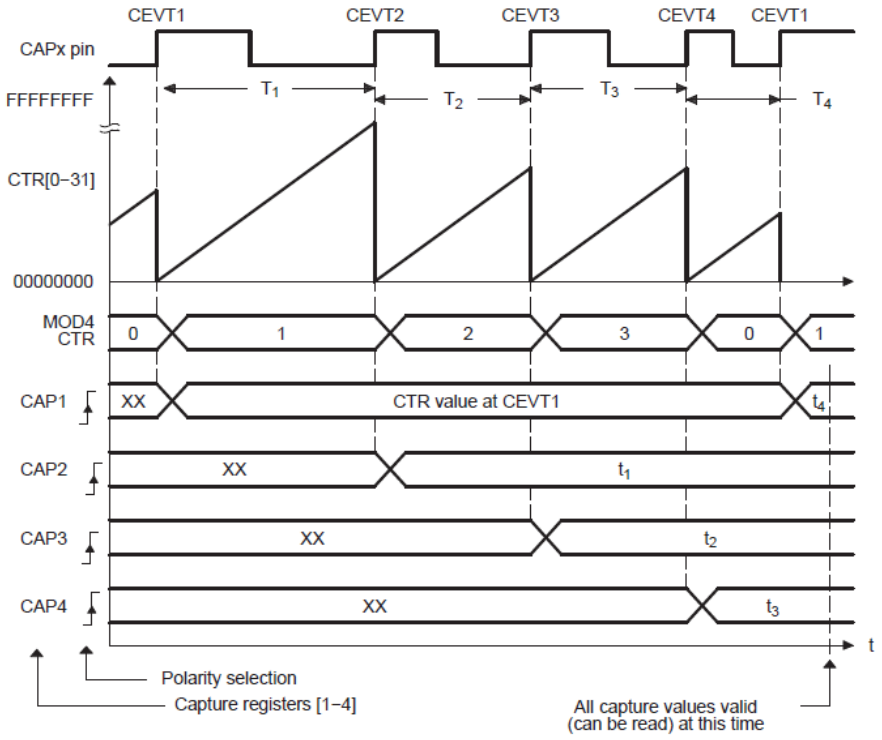


Figura 3-8 - eCAP com modo de operação em tempo relativo, com todos os eventos configurados em borda de subida. [22]

Esse módulo conta com 8 entradas analógicas, chamadas de AN0 a AN7, que suportam tensões de 0 a 1.8V, com polaridade apenas positiva. Essa tensão será medida entre esses pinos ANx e o pino de referência do conversor ADC, chamado de GND_ADC. Caso seja necessário utilizar o valor máximo de 1.8V como alimentação para capturar a medida (como medir o valor de uma resistência em um divisor de tensão), o pino chamado VDD_ADC poderá ser utilizado para fornecer essa tensão máxima suportada pelo ADC. Cada amostra é armazenada em um valor inteiro de 12 bits, portanto com valores que vão de 0 a 4095. O acréscimo de uma unidade nesse valor representa cerca de 0.44mV na amostra analógica lida. [22]

Apesar de possuir 8 entradas analógicas, dentro do módulo ADC existe apenas um núcleo que realiza a amostragem, conforme visto na Figura 3-9. Dentro do módulo ADC há um multiplexador, representado pelo AMUX 9:1 na Figura 3-9, que realiza a multiplexação das entradas analógicas ANx para a entrada do módulo de conversão, representado pelo ADC da imagem. Vários outros componentes são representados na imagem, porém a maioria são relacionados à configuração utilizando o controlador de telas *touchscreen*, não abordado nesse trabalho. Para mais informações sobre essa função e outros componentes não abordados nessa seção, consultar o manual *AM335x Sitara Processors – Technical Reference Manual* (2011). [22]

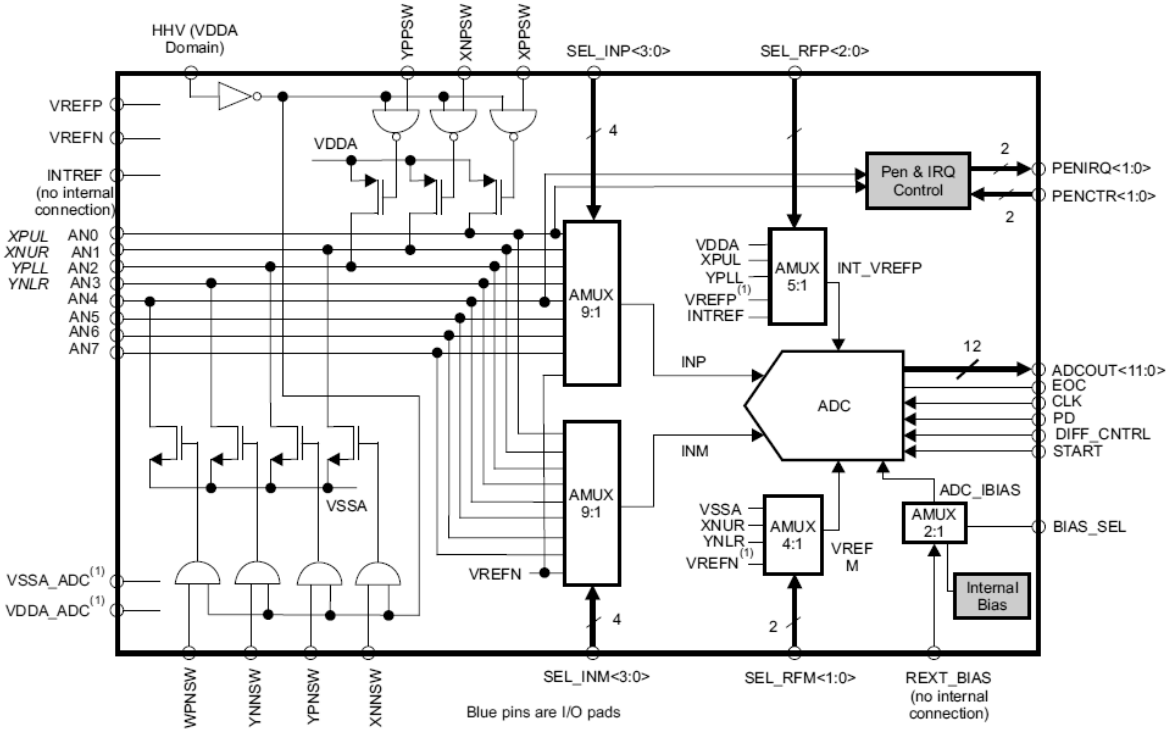


Figura 3-9 - Representação de funcionamento do módulo ADC. [22]

O módulo de conversão ADC realiza uma amostragem e a disponibiliza utilizando 15 ciclos de clock do módulo ADC. O clock padrão que chega até esse módulo é de 24MHz, porém é possível configurar divisores na base desse valor, para diminuir a velocidade de amostragem. O módulo funciona a partir de passos, denominados no manual como *steps*. É possível configurar o módulo ADC para operar continuamente, realizando o reinicio do ciclo automaticamente, ou operar em modo *single-shot*, que executa o ciclo configurado apenas uma vez, necessitando do programa da PRU ativar novamente o módulo para então realizar mais um ciclo. [22]

Um ciclo pode consistir dos seguintes passos: *idle*, *charge*, *samples* e *open*. No primeiro passo do ciclo, chamado *idle*, é possível aplicar uma configuração de aterramento através de transístores presentes dentro desse módulo para gerar um evento de detecção em uma tela *touchscreen*. Esse passo não será detalhado pois não será utilizado no projeto. O segundo passo, chamado *charge*, serve para amostrar o evento de carregamento dos capacitores de uma tela *touchscreen*, também não utilizado neste projeto. Os últimos passos, chamados de *sample steps* são os eventos de captura analógica propriamente ditos. No registrador STEPENABLE do módulo ADC é possível escolher quais passos, serão ativados em cada ciclo, sendo possível desativar por meio desse registrador o passo *charge*. Cada passo de *sample step* ativado, assim como o *idle*, *open* e *charge* gastam 15 ciclos de clock para serem executados. Os sample steps disponíveis são numerados de STEP1 ao STEP16, cada um com um registrador para configurar qual pino de ANx será utilizado no *sample step* escolhido, além de quantos clocks adicionais serão utilizados para gerar uma média de amostragem e qual o delay esperado em clocks antes de iniciar a amostragem. É possível em cada STEP configurar um passo adicional chamado *open*, no qual o conversor coloca uma impedância muito alta na entrada analógica que será amostrada, usada também no modo de amostragem para telas *touchscreen*. [22]

Para configurar o módulo ADC afim de realizar amostragens de propósito geral (sem a utilização do controlador de *touchscreen*), devemos configurar os passos da seguinte maneira: [22]

- Desabilitar o passo *idle* por meio do registrador IDLECONFIG
- Desabilitar o passo *charge* e habilitar quais *sample steps* (STEP1 a STEP16) serão utilizados por meio do registrador STEPENABLE.
- Configurar os respectivos *sample steps* habilitados, definindo a porta de amostragem (AN0 a AN7), além também o número acional de *clocks* para a

amostragem e o número de *clocks* que a porta permanecerá no passo *open* antes de iniciar a amostragem.

Depois do módulo ADC ser configurado conforme as instruções acima, é necessário habilitá-lo por meio do registrador CTRL desse subsistema. Assim ele começará a percorrer somente os passos habilitados, sempre na seguinte ordem: *idle*, *charge*, STEP1, STEP2, ..., STEP16, voltando então para o início do ciclo caso o modo contínuo esteja habilitado. A cada amostragem feita nos passos *sample steps*, a amostra é armazenada em um dos registradores presentes nesse módulo, denominados como FIFO0 e FIFO1. A definição de qual dos dois será utilizada é configurada em cada *sample step*, juntamente com a porta de amostragem utilizada. Cada campo de registrador FIFO possui a capacidade de armazenar 128 *words* de 32 bits, sendo que os dois primeiros bytes menos significativos dessa *word* estão as amostras de 12 bits vindas do ADC. Os últimos dois bytes registram informações sobre qual o STEP responsável pela aquela amostra. Sempre que uma nova amostra chega ao registrador FIFO, ela é armazenada no final do campo reservado para esse registrador. Para o usuário saber quantas amostras estão esperando para serem lidas, basta consultar o registrador FIFO0COUNT ou FIFO1COUNT. Ao ler uma amostra do registrador FIFO, ocorre um deslocamento em que a amostra mais antiga se torna a próxima amostra lida. O usuário deve ficar atento para não permitir que o limite desse registrador (128 amostras) seja ultrapassado, senão as novas amostras não serão salvas. O funcionamento da amostragem, armazenagem no FIFO e leitura dos dados desse registrador estão representados na Figura 3-10. [22]

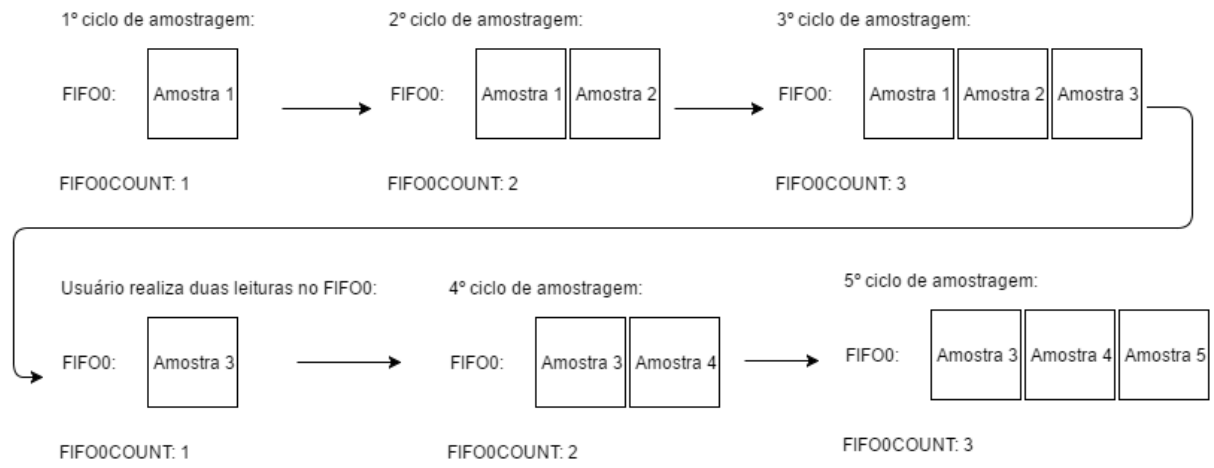


Figura 3-10 - Representação dos ciclos de amostragem, armazenamento na FIFO0 e leitura desse registrador.

Para conseguir a maior taxa de amostragem possível do módulo ADC, as seguintes configurações foram adotadas: desabilitar o passo *idle* e o passo *charge*. Habilitar apenas o STEP1, direcionado a uma porta ANx que se deseja fazer a amostragem. Definir a configuração do STEP1 para 0 *clocks* adicionais para a amostragem e 0 *clocks* no passo de open. Definir o divisor de *clock* de entrada do módulo ADC para 0, desabilitando assim o divisor de *clock*. Ativar o módulo ADC em modo contínuo. Feito essa configuração, é possível obter 1 milhão e 600 mil amostras por segundo da porta ANx escolhida, devido a definição de um ciclo sendo apenas o passo STEP1. Conforme dito, a entrada de clock máximo do conversor ADC é 24MHz e, ao utilizar 15 ciclos por amostragem no STEP1, obtem-se uma frequência de amostragem de 1.6MHz. [22]

3.4 Comunicação entre o núcleo PRU e o núcleo ARM

Conforma já mostrado pela Figura 3-3, o núcleo ARM do processador AM335x comunica com os núcleos PRU e os outros subsistemas presentes no processador por meio de dois barramentos: L3 e L4. É possível o acesso direto de memória entre os dois núcleos de processadores: o processador ARM consegue acessar as memórias de programa e memórias de dados do PRU0 e PRU1, além da região chamada de SHARED RAM. Na outra via, os PRUs conseguem acessar diretamente uma região de memória alocada na memória DDR, utilizada pelo núcleo ARM, porém é importante lembrar que todas as operações que utilizam o barramento L3 ou L4 não possuem a garantia de tempo de execução nos núcleos PRU. [22]

Para o núcleo PRU conseguir executar o software programado para ele são necessários dois passos: a gravação do código traduzido pelo *assembler* na área de memória de programa do núcleo PRU que irá ser utilizado e a configuração do registrador CTRL do núcleo desejado para o modo *ENABLE*. Essa configuração faz com que o núcleo PRU execute sequencialmente o programa armazenado na sua memória de programa, até que uma instrução de *sleep* ou de *halt* seja encontrada, ou então o registrador CTRL do PRU seja configurado para o modo *DISABLE*. [22]

3.4.1 Assembler PRU

A *Texas Instruments*, fabricante do processador AM335x, disponibiliza um *assembler* que traduz o código escrito em Assembly pelo o usuário para a linguagem de máquina do PRU. Essa ferramenta tem o nome de *PASM Tool* e possui licença de uso livre. Para facilitar o uso na BeagleBone, foi desenvolvido em um formato *opensource* um pacote de ferramentas chamado de *AM335x PRU Package*, que inclui uma biblioteca para a linguagem C, um driver para o Linux chamado *uio_pruss* e o *PASM Tool*. [20]

Ao instalar corretamente esse pacote, o usuário poderá compilar seus programas escritos em Assembly em arquivos binários, resultantes da ferramenta PASM. Por meio do driver *uio_pruss* o Linux consegue alocar uma região de memória com o tamanho de até 8MB que poderá ser acessado também pelos núcleos PRU, além de estabelecer a conexão com os núcleos através dos barramentos L3/L4. Esse driver deverá ser carregado toda vez que a BeagleBone for iniciada e o usuário for utilizar os núcleos PRU para a execução de alguma tarefa. A biblioteca em C, chamada *prussdrv*, permite ao usuário a escrita de um programa nesta linguagem para ser executado pelo núcleo ARM, porém faz a definição de funções que realizam a comunicação com os núcleos PRU [20]. Essas funções são:

- Carregamento dos arquivos binários gerados pela ferramenta PASM para a região de memória de programa do núcleo PRU escolhido. [20]
- Configurar o núcleo PRU escolhido para o modo ENABLE ou DISABLE, realizando assim o início ou a parada da execução do código escrito na memória de programa do núcleo escolhido. [20]
- Mapear a memória de dados do núcleo PRU escolhido e retornar o seu endereço para ser utilizado pelo programa em C. [20]
- Buscar o endereço de início da região alocada pelo driver *uio_pruss*, além de buscar o tamanho desse buffer alocado. [20]
- Esperar eventos de interrupção vindas dos núcleos PRU. [20]
- Gerar sinais de interrupção para os núcleos PRU. [20]

O usuário precisa, portanto, programar sua rotina de software que será executado em um dos núcleos da PRU, para então programar um software na linguagem C que irá utilizar a biblioteca citada para realizar a comunicação entre o núcleo ARM e o núcleo PRU escolhido. É possível também programar duas rotinas para serem executadas independente em cada um

dos núcleos PRU, com o mesmo software em C, que irá descarregar na memória de programa de cada núcleo PRU um arquivo binário diferente.

3.4.2 Device Tree Overlay (DTO)

Conforme foi falado na seção 3.1 deste capítulo, cada pino de GPIO da BeagleBone pode possuir uma ou mais função, sendo necessário realizar uma multiplexação dessa função antes de utilizá-lo para um certo propósito. Essa multiplexação tem o nome de *pinmux*. [20]

Assim que o sistema Linux é iniciado os pinos encontram-se no estado padrão de *pinmux*, que nos pinos de GPIO é o próprio estado de *input* e *output* digital. Na Figura 3-2 pode-se ver as outras funções que os pinos podem assumir, como por exemplo a UART4_RX e UART4_TX para os pinos P9_11 e P9_13 respectivamente. Para realizar o *pinmux* correto e fornecer a esses pinos as funções de UART, é necessário carregar um *Device Tree Overlay*, ou DTO, no sistema do Linux. [20]

Esse DTO fornece ao Linux a informação de como se comunicar com o *hardware* e assim realizar o *pinmux* para a função do pino definido dentro do DTO carregado no sistema. O usuário deverá compilar escrever um arquivo de extensão *.dts* com os formatos especificados para um arquivo de DTO, explicitando nesse arquivo quais pinos serão utilizados, qual o modo eles serão definidos e suas propriedades, que podem variar entre *Pull-Up*, *Pull-Down* e desabilitado. O modo do pino deverá ser consultado em tabelas especificas ou ferramentas da BeagleBone, que fornecerão quais as funções disponíveis para aquele pino e qual o modo de cada um. As funções de um pino podem ser: [20]

- GPIO: que será utilizado pelo sistema Linux por meio de programas executados em diversas linguagens, como por exemplo o BoneScript. A velocidade de leitura e escrita nos pinos neste modo é de até 200kHz.
- PRU_GPIO: pinos de entrada e saída, porém controlados pelos PRU. Esses pinos possuem a capacidade de leitura/escrita de até 100MHz.
- Subsistemas: uma ou mais saídas de subsistemas presentes dentro do processador AM335x podem aparecer em cada pino, como por exemplo o ePWM, eCAP, eQEP, UART, entre outros.

Esse arquivo DTO, após compilado, poderá ser carregado pelo sistema Linux, para então comunicar ao hardware qual o *pinmux* deverá ser feito para os pinos configurados nesse arquivo. É importante lembrar que caso for carregado um arquivo DTO que possuía um pino definido para uma função, é necessário descarregá-lo para poder carregar outro arquivo que possua o mesmo pino em outra função. Caso contrário, o Linux detectará o conflito e informará que não é possível carregar o segundo arquivo. [20]

4 DESENVOLVIMENTO DO ANALISADOR DE REDE PROFIBUS

Nesse capítulo serão descritas as etapas para o desenvolvimento do *software* visando alcançar o objetivo final do projeto: a elaboração de uma ferramenta para análise de uma rede Profibus.

Os dois capítulos anteriores mostraram conceitos necessários para o entendimento para a captura adequada da rede Profibus e também sobre o hardware BeagleBone, no qual foi mostrado as principais funções e quais os subsistemas que foram utilizados para o desenvolvimento deste *software*.

4.1 Lista de materiais utilizados no desenvolvimento

Para o desenvolvimento do software que realizará a captura dos telegramas e também coletará as amostras analógicas da rede Profibus foi utilizado o computador de placa BeagleBone Black Rev. C, com o sistema operacional Angstrom/Linux instalado.

Para a transformação do sinal diferencial Profibus em sinal digital apto a ser ligado em uma porta GPIO da BeagleBone, que suporta tensões de até 3.3V, foi utilizado a placa PA006301 da Profichip, que possui um *transceiver* EIA-485/TTL, que já realiza essa conversão de modo apropriado e com isolamento elétrico entre a rede Profibus e a saída digital, oferendo uma melhor proteção.

Para a implementação da rede Profibus para aprendizado do funcionamento e testes foram utilizados equipamentos do Laboratório de Automação Industrial (LAI), no qual foi utilizado um Controlador Lógico Programável (CLP) da Siemens, modelo S7-1200 CPU1214c DC/DC/DC, juntamente com escravos de diferentes fabricantes, como Altus, Siemens e DLG.

4.2 *Transceiver* Profibus

O EIA-485, a camada física mais utilizada para as redes Profibus, transmite as informações da rede por meio de dois canais, denominados como A e B, que possuem valores

de tensão iguais, porém em polaridades diferentes. A associação desse modo de transmissão junto com a utilizado de cabos de par trançado melhoram a robustez da transmissão contra ruídos eletromagnéticos. Segundo a *Normative Parts of Profibus FMS, DP and PA (1998)*, o valor de tensão entre esses dois canais deve permanecer entre 4V e 7V.

A BeagleBone suporta em seus pinos GPIO a tensão de até 3.3V com polaridade positiva, onde valores superiores a esse limite podem causar danos permanentes à placa. Para isso, não se pode ligar os canais da Profibus diretamente no pino GPIO.

A ferramenta que realiza essa conversão é chamada de *transceiver*. Para esse projeto, foi utilizado um *transceiver* EIA-485 que converte o sinal Profibus em níveis lógicos de tensão com amplitude de 5V. Foi feito um divisor de tensão utilizando seis resistores de 1 kΩ em série, fornecendo uma resistência equivalente total de 6 kΩ. O circuito apresentado na Figura 4-1 mostra a ligação realizada para obter uma tensão de 3.3V para o pino GPIO ligado à BeagleBone.

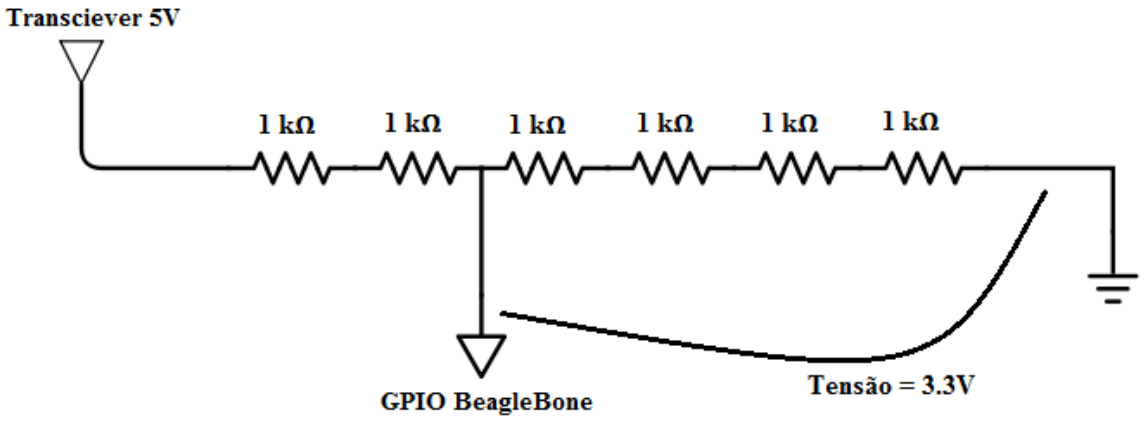


Figura 4-1 - Divisor de tensão utilizado para realizar a conexão entre o transceiver EIA-485 e um pino GPIO da BeagleBone.

Existem circuitos integrados (CI) que fazem a conversão do sinal EIA-485 para sinais digitais em diferentes níveis de tensão, como o MAX485, que possui a limitação de *Baud Rate* de até 200kbps [23], ou o ISO1176 da *Texas Instruments* [24], que possui o funcionamento completo voltado para a Profibus. A justificativa da escolha da placa PA006301 no desenvolvimento foi que ela já estava disponível para o uso no Laboratório de Automação Industrial, não necessitando assim a compra de algum CI para realizar essa tarefa.

4.3 Conexões com os GPIO da BeagleBone

Neste projeto foi definido alguns pinos GPIO utilizados na placa BeagleBone que devem ser conectados corretamente para o funcionamento do software desenvolvido. São eles:

- **P9_1**: O pino 1 do *Pin Header* 9 é o pino de referência (GND) do circuito geral da BeagleBone. Ele deve ser ligado ao GND do *transceiver* utilizado para realizar a captura do sinal digital corretamente.
- **P9_27**: Esse pino deverá ser conectado ao RX do transceiver. Como foi utilizado um divisor de tensão de resistências para adequar a tensão de entrada do GPIO da placa, esse pino deverá ser conectado no terminal GPIO BeagleBone representado na Figura 4-1.
- **P9_34**: Esse pino, nomeado de GND_ADC, deve ser ligado à referência do circuito que coletará o sinal de um dos canais da rede Profibus. Será especificado posteriormente como esse circuito deverá ser montado.
- **P9_39**: O pino AN0 será a entrada analógica para o canal 0 do subsistema ADC do processador AM335x, que será conectado no local de amostragem de um dos canais da rede Profibus.

4.4 *Device Tree Overlay* utilizado no desenvolvimento

Definido os pinos que seriam utilizados no desenvolvimento do projeto, foi necessário realizar a programação do *Device Tree Overlay* (DTO), para realizar o *pinmux* correto nos GPIO da BeagleBone. O pino P9_27 possui o offset de identificação com o valor 01A4h e foi configurado para operar no modo 6, que garante o modo de input para os núcleos PRU. As propriedades adicionais desse pino foram: *Fast Slew Rate, Pull-Up,* Modo 6. O pino P9_39 também foi incluso nas configurações do DTO, para habilitar o acesso ao pino. Esse pino não pode ter outra função, portanto não é necessário informar o modo de operação e nem as propriedades adicionais.

O resultado ao compilar esse arquivo é um arquivo de extensão .dtbo, que posteriormente vai ser utilizado para carregar essa configuração no Linux.

4.5 Organização do software desenvolvido

O software desenvolvido para alcançar os objetivos do projeto conta com três módulos diferentes, que fornecerão os dados coletados da rede Profibus para um quarto módulo, projetado pelo usuário do analisador de rede desenvolvido.

O primeiro módulo é o software produzido pela linguagem C++ , com as bibliotecas Qt e *prussdrv*, executado no sistema operacional pelo núcleo ARM do processador. A biblioteca Qt é utilizada nesse módulo para a criação de um servidor TCP/IP que que irá receber a conexão do software do usuário que deseja coletar os dados fornecidos pela ferramenta. A biblioteca *prussdrv* inclui funções que realizam a comunicação entre o sistema operacional Linux e o PRU, como as funções de gravar o código na memória de programa do PRU e detectar interrupções. Neste trabalho, esse módulo será referenciado como o módulo *host*.

O segundo módulo é responsável pela captura digital da rede Profibus, através do pino que recebe esse sinal do *transceiver* utilizado, ligado na BeagleBone através do pino P9_27. Ele ficará encarregado de encontrar o Baud Rate que a transmissão pela rede Profibus se encontra, além de detectar erros de transmissão nos caracteres lidos da rede. Esse módulo, chamado de módulo digital, irá separar os telegramas para serem salvos em blocos na memória, comunicando com o módulo *host* e também com o módulo analógico.

O módulo analógico, o terceiro dos módulos inclusos no software desenvolvido, é responsável pela captura de amostras analógicas diretamente de um dos canais da rede Profibus. Ele receberá informações do módulo digital, para fazer a associação da amostra capturada com o endereço do equipamento que enviou a mensagem, para atualizar as amostras salvas em intervalos de tempo configurados pelo software do usuário.

O quarto módulo, que não está incluso dentro do software desenvolvido, será chamado de software do usuário. Ele ficará a cargo de implementação do usuário, nas quais serão dadas instruções para o uso correto da comunicação TCP/IP para obter os dados e configurar o analisador de rede para realizar as capturas desejadas pelo usuário, dentro dos objetivos propostos do trabalho. No capítulo 6 será mostrado um exemplo de aplicação para que o usuário possa ter uma noção de quais as capacidades da ferramenta.

4.6 Diagrama de fluxo de dados

Nas próximas seções serão apresentadas o diagrama do fluxo de dados (DFD) entre os módulos e também seus subníveis, que fornecerá informações sobre quais módulos trocam informações entre si e quais os tipos de dados utilizados nessa troca. Serão apresentadas e descritas também as principais funções presentes em cada módulo e seus respectivos subníveis, seguindo os modelos padrões utilizados em Engenharia de Software.

4.6.1 Nível zero

Exibido na Figura 4-2, o nível zero de um DFD é chamado também de nível de contexto, que mostra as informações trocadas entre as entidades externas de cada módulo e também os dados trocados entre os módulos. A explicação geral de qual a função de cada módulo já foi explicada na seção 4.5 deste capítulo.

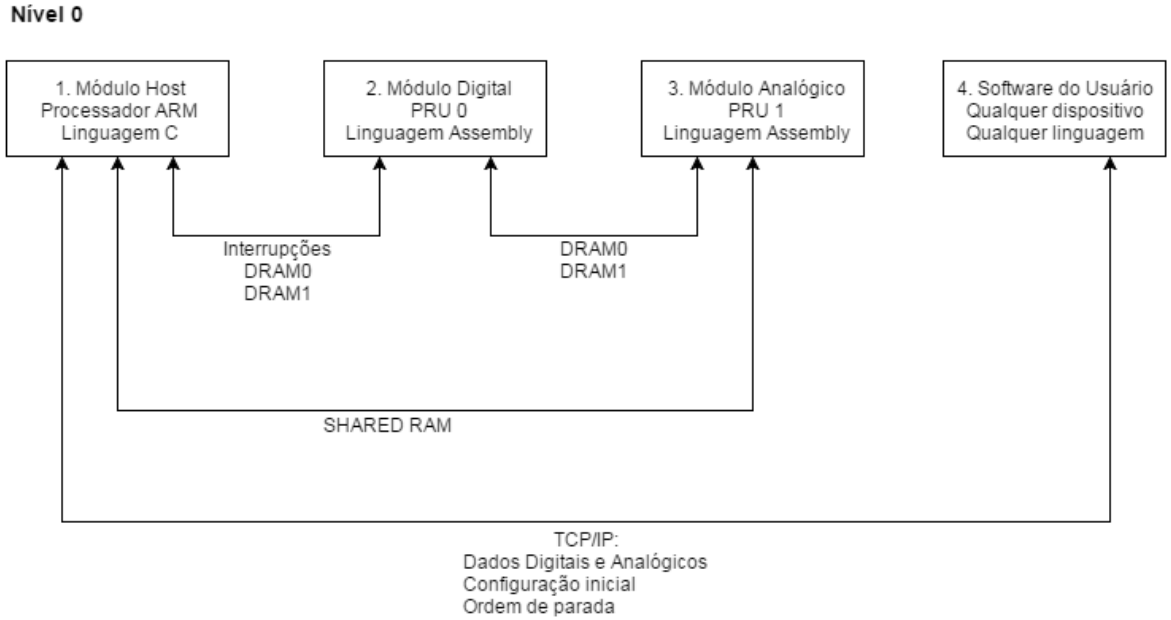


Figura 4-2 - Nível zero do diagrama de fluxo de dados do programa.

O usuário da ferramenta ficará encarregado de programar um software que realizará a comunicação com o módulo *host* via TCP/IP, seguindo o protocolo definido no capítulo de instruções de utilização desse trabalho. Dentro da inicialização da ferramenta, o usuário poderá

configurar quantos pacotes de telegramas ele deseja capturar da rede, se deseja habilitar o módulo analógico se possível e também qual o tempo de atualização das amostras analógicas associadas a cada endereço. O módulo *host* irá enviar para o software do usuário a informações coletadas da rede.

O módulo digital fará leituras na porta P9_27, obtendo assim o sinal que passa pela rede Profibus, já filtrado pelo *transceiver*. O módulo analógico realizará leituras na porta AN0 de um dos canais da Profibus.

Entre o módulo *host* e o módulo digital serão trocadas informações pelos bancos de memória DRAM0 e DRAM1, que possuem espaço de 8192 bytes cada uma. Serão utilizados os 20 primeiros bytes dos dois bancos como cabeçalho de configuração, onde informações sobre qual banco está sendo utilizado, quantos telegramas foram salvos naquele banco, entre outros que serão melhores descritos abaixo. O módulo digital ativará um evento de interrupção que fará com que o módulo *host* identifique o banco que acabou de ser preenchido e realize a cópia da memória do PRU para a memória DDR, para assim prosseguir no envio desses dados via TCP/IP. Entre o módulo host e o módulo analógico haverá a troca de dados via o banco de memória SHARED RAM, que possui 12kB de espaço. O módulo host verificará periodicamente se existem amostras novas salvas no banco, realizando assim a cópia desse banco de memória para uma área da memória DDR, para assim também prosseguir no envio desses dados via TCP, caso a configuração inicial do usuário habilite o módulo analógico.

Entre o módulo digital e o módulo analógico há a troca de dados via a área definida como cabeçalho de configuração, que são os 20 primeiros bytes de cada banco de memória de dados dos núcleos PRU, o DRAM0 e DRAM1.

4.6.2 Nível um

No nível um de um DFD são apresentadas as funções internas de cada módulo, representando quais informações chegam nelas e quais informações saem delas. As funções não necessariamente representam funções reais dentro do código programado, mas sim uma divisão de uma única tarefa.

4.6.2.1 Nível 1: Módulo *Host*

O nível 1 do módulo *host*, representado na Figura 4-3, apresenta as principais tarefas que compõe este módulo. Assim que o software é iniciado, o módulo *host* começa a ser executado. Sua primeira tarefa é a criação do servidor TCP/IP, o qual será configurado a porta que será escutada, esperando uma conexão do software do usuário. Após o servidor TCP/IP ser iniciado, são criados dois threads: um que executa o monitoramento da porta escolhida, esperando pela conexão do software do usuário, e outra que será responsável pela comunicação com os núcleos PRU.

Após a criação dos *threads*, o módulo *host* inicia a comunicação com o driver *uio_pruss*, estabelecendo uma conexão com os núcleos PRU. É então mapeado vários ponteiros de diferentes tipos de dados, como *shorts* e *ints*, apontando para as regiões dos bancos de memória DRAM0, DRAM1 e SHARED RAM, que farão a leitura e a escrita de dados diretamente nessas memórias.

O módulo *host* espera então uma conexão pela porta escolhida. Assim que uma conexão por essa porta é estabelecida, o módulo *host* recebe do software do usuário as configurações que o usuário definiu, como a habilitação do módulo analógico, qual o intervalo de atualização das amostras analógicas em cada endereço e quantos telegramas o módulo digital deve capturar. Essa configuração é escrita no cabeçalho de configuração da DRAM0 e DRAM1. É então chamada uma instrução que grava os programas do módulo digital e o módulo analógico nas memórias de programa do PRU0 e PRU1 respectivamente, iniciando também a comunicação de interrupções entre os núcleos PRU e o núcleo ARM.

Após os dois programas serem gravados com sucesso nos núcleos PRU, o módulo *host* entra em um ciclo denominado como ciclo principal. Esse ciclo coletará os dados do módulo digital e analógico e os enviará via TCP/IP para serem utilizados pelo software do usuário. Esse ciclo será mais detalhado no nível 2. Ele permanecerá ativo até que a conexão TCP/IP estiver estabelecida. Assim que cessada a conexão, o programa desabilitará os núcleos PRU e voltará ao primeiro passo, ou seja, a criação de um novo servidor TCP/IP, executando os passos já explicados nessa seção novamente.

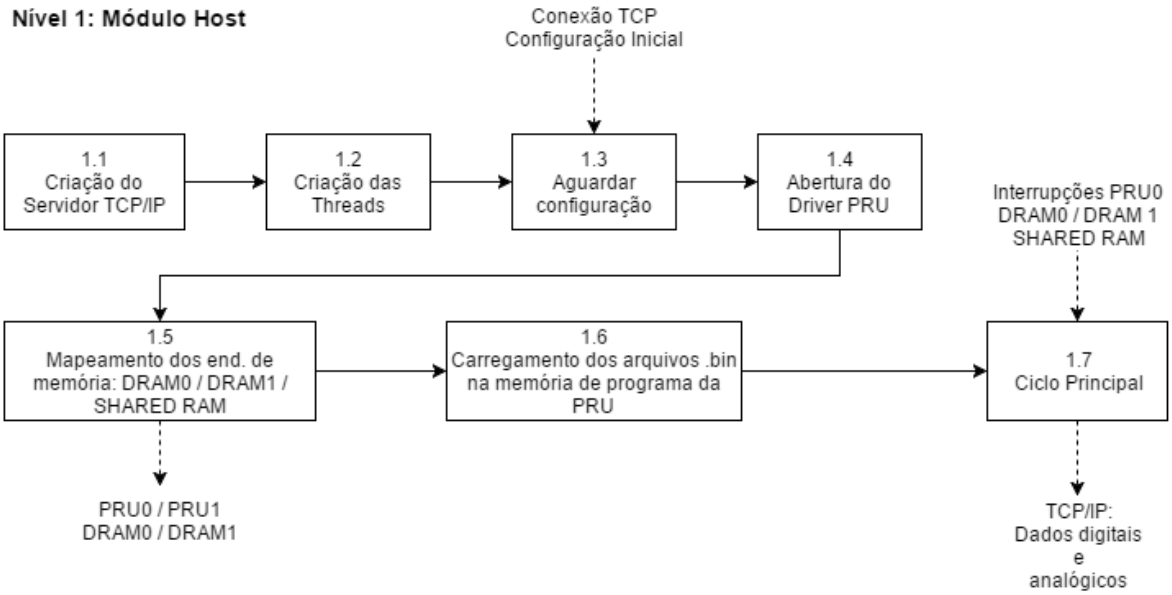


Figura 4-3 - DFD Nível 1 - Módulo Host.

4.6.2.2 Nível 1: Módulo Digital

Serão apresentadas nessa seção as principais tarefas do módulo digital, conforme vistas na Figura 4-4. O módulo digital inicia com o carregamento do programa no núcleo PRU0 feito pelo módulo host no passo descrito como “Carregamento dos arquivos .bin na memória de programa da PRU”. Devido ao carregamento correto do *Device Tree Overlay* no sistema Linux, descrito na seção 4.4 deste capítulo, os núcleos PRU ganham acesso ao pino P9_27 para realizar leituras do estado digital que está ligado a ele, permitindo assim ler as informações que estão passando na rede Profibus.

A primeira tarefa realizada ao iniciar o módulo digital é a configuração dos registradores que habilitam o barramento L3/L4, permitindo que o módulo host acesse as informações dos bancos de memórias dos núcleos PRU. É configurado também o timer utilizado para o timestamp da captura. Todos os registradores de uso geral são zerados, para não gerar problemas com valores que estavam já armazenados em uma utilização prévia.

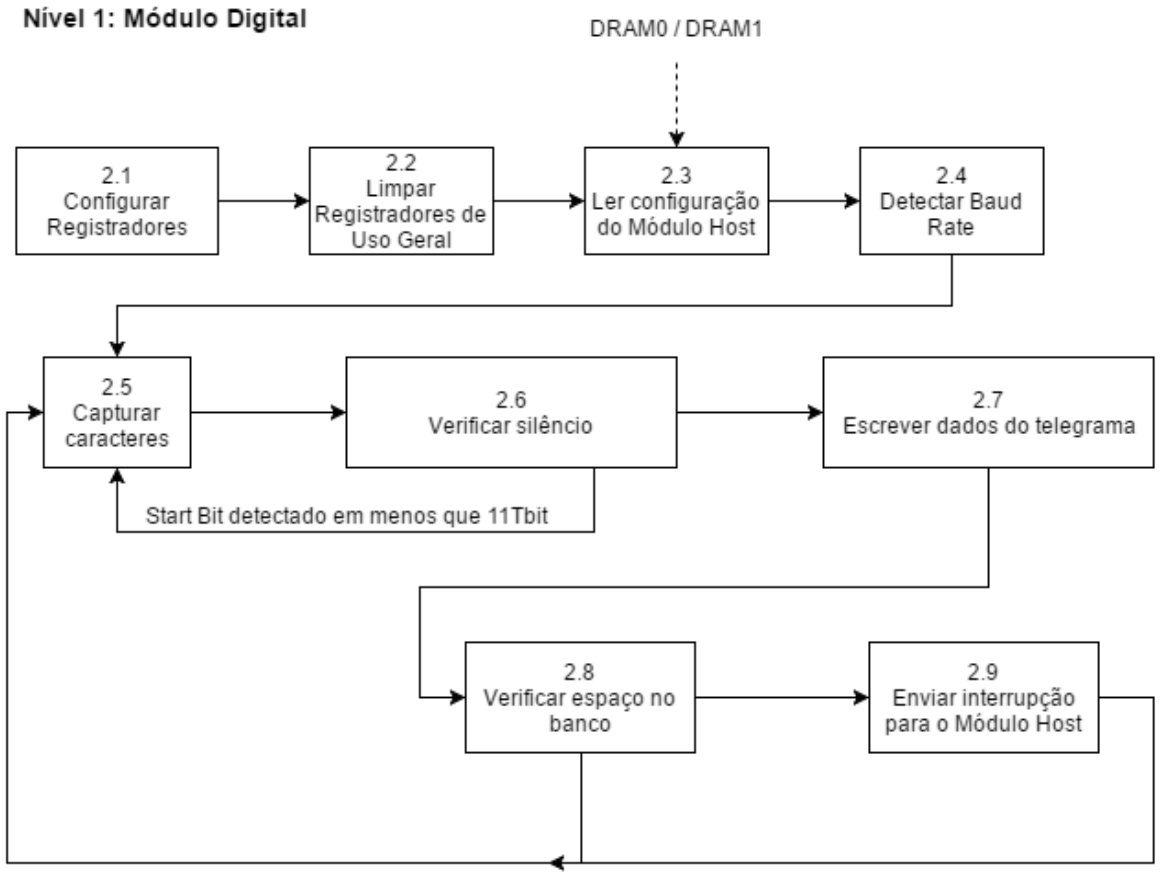


Figura 4-4 - DFD Nível 1 - Módulo Digital.

Logo após essa etapa de configuração, o módulo digital entra no modo de detecção de *Baud Rate*. Esse processo será melhor detalhado no nível 2. O resultado dessa tarefa será salvo no cabeçalho de configuração, com um valor de 1 a 11, representando os valores de *Baud Rate* em ordem decrescente, iniciando do 12Mbps. O valor 0 salvo no cabeçalho de configuração diz ao módulo host e ao módulo analógico que nenhum Baud Rate válido foi detectado ainda, seja por falta de transmissão na rede ou seja pela utilização de uma Baud Rate não válida para a Profibus.

Assim que uma Baud Rate válida for detectada, o modo digital entra no modo de telegramas. A tarefa de esperar o término do *idle* da rede espera até que o estado lido no pino P9_27 saia do nível lógico 1 e passe para o nível lógico 0. Isso significa que um telegrama começou a ser transmitido, devido ao start bit, portanto cada caractere desse telegrama passará a ser lido, bit a bit. É salvo no cabeçalho de configuração um dado que será interpretado pelo módulo analógico como início de transmissão de um telegrama.

A função de ler um caractere também será melhor detalhada no nível 2. Em resumo, assim que o *start bit* é detectado o *timestamp* de início do telegrama é salvo. Um período de meio tempo de bit é aguardado, tempo esse definido pela detecção do Baud Rate, a partir do primeiro start bit transmitido. Depois o programa sempre aguarda o tempo equivalente a um tempo de bit, realizando leituras do nível lógico presente na porta P9_27. Com isso os caracteres vão sendo montados, bit a bit, e quando completos passam pela verificação da paridade. Se nenhum erro na transmissão do caracter ocorreu, o módulo digital verifica se depois da transmissão do caractere houve um silêncio. Caso ainda haja transmissão de outro caractere, esse caractere passa pelo mesmo processo descrito, até que um silêncio seja detectado. Detectado um silêncio, o módulo digital entende que o envio do telegrama acabou, salvando assim o *timestamp* de término de transmissão daquele telegrama.

No banco que está sendo utilizado (DRAM0 ou DRAM1) é salvo o telegrama montado, juntamente com o seu tamanho e seus *timestamp* de início e término e o status de validação do telegrama, indicando se em algum caractere houve algum erro de transmissão (start bit, paridade ou stop bit). É verificado se é possível salvar mais um telegrama no banco utilizado, fazendo a verificação se ainda há no mínimo 272 bytes disponíveis na memória do banco utilizado. Esse número vem do tamanho máximo que um telegrama da Profibus pode ter, juntamente com as informações de tamanho do telegrama e do *timestamp* capturado pelo módulo. Se não houver memória suficiente, o banco de dados é trocado, alterando um valor presente no cabeçalho de configuração. Uma interrupção para o módulo host é enviada, sinalizando que um banco de memória foi totalmente preenchido.

Para evitar uma propagação de erros muito grande caso a Baud Rate se altere durante a execução do software, sempre que um erro de transmissão é detectado por esse módulo é somado um no contador de erros seguidos. Se houverem mais de 3 caracteres com erro de transmissão seguidos, o programa volta na etapa de detecção de Baud Rate. Toda vez que um caractere é lido sem erros, o contador de erros seguidos é zerado.

4.6.2.3 Nível 1: Módulo Analógico

O nível 1 do diagrama do fluxo de dados do módulo analógico está apresentado na Figura 4-5, no qual estão presentes as principais tarefas realizadas por esse módulo.

O módulo analógico é iniciado pelo módulo host, na tarefa “Carregamento dos arquivos .bin na memória de programa da PRU”. Esse programa será executado pelo núcleo PRU1.

Assim que iniciado, o núcleo realiza os passos de configuração do registrador que aponta para o endereço de memória inicial do banco SHARED RAM. Com isso, torna-se possível acessar a SHARED RAM por meio de um único comando em *assembly*, por meio de um recurso chamado *Constant Table*, ao invés de passar junto com o comando de acesso a memória o endereço que queremos ler ou escrever.

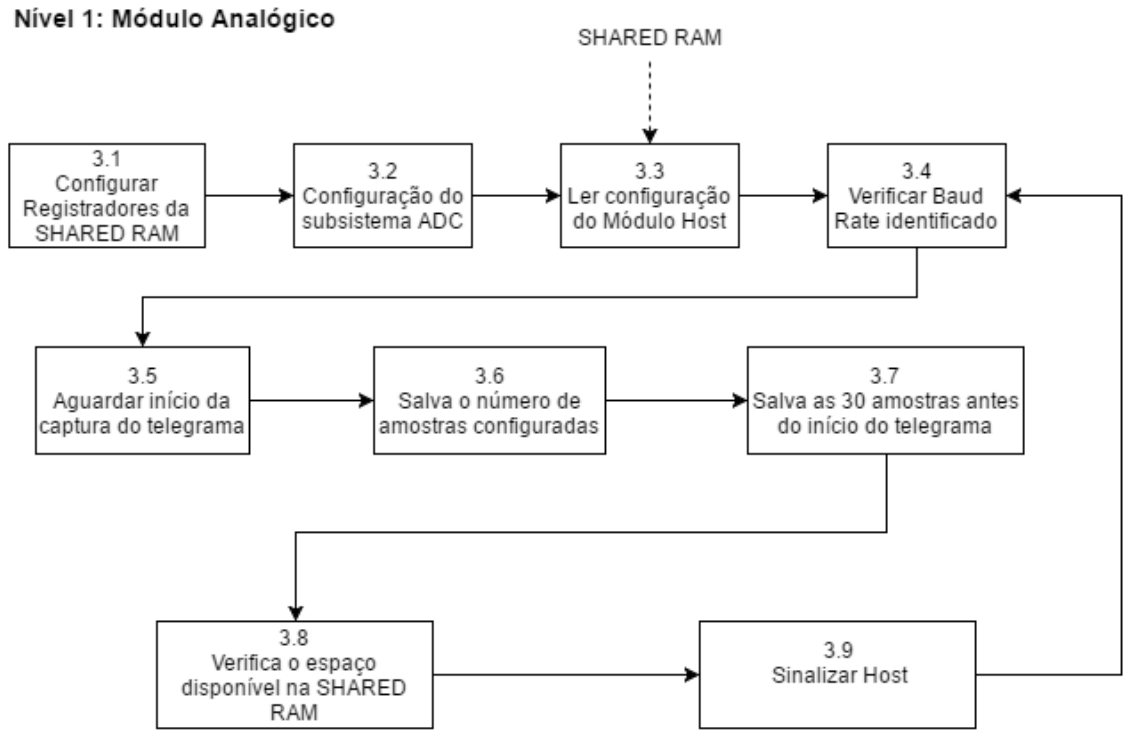


Figura 4-5 - DFD Nível 1 - Módulo Analógico.

Depois a configuração dos registradores do subsistema ADC é feita, ativando assim o módulo analógico para realizar amostragens. São lidos também os parâmetros de configuração vindos do módulo *host*. A próxima etapa checa os valores do cabeçalho de configuração, verificando se um Baud Rate válido foi identificado e também se esse *Baud Rate* é menor ou igual a 500kbps. Devido a velocidade máxima de amostramento do subsistema ADC de 1.6MSPS, essa limitação foi necessária, pois já garanta no mínimo 3 amostras por bit. Para valores de Baud Rate maiores que esse, a velocidade de amostramento não é suficiente, gerando resultados com erro.

Se um *Baud Rate* maior que 500kbps for identificada, as amostras ADC lidas são sempre descartadas. Senão, as amostras passam por um processo de chaveamento entre os registradores de uso geral, criando uma espécie de buffer que armazena as 30 ultimas amostras lidas da rede, sendo que as amostras mais velhas são descartadas se esse buffer se encher. Esse processo ocorre enquanto a rede está em *idle*. Juntamente com isso, a verificação do cabeçalho de configuração é feita, esperando a mudança do byte que está reservado para sinalizar que um telegrama começou a ser transferido, obtendo também o endereço da estação que está enviando esse telegrama. Assim que o telegrama começou a ser transferido começa a ser gravado no banco de memória SHARED RAM o número de amostras definido pela configuração vinda do módulo *host*. Depois que esse número configurado de amostras analógicas foi gravado na memória, é feito a gravação do buffer das 30 amostras, antes do endereço inicial que armazenou as amostras depois do início do telegrama. Isso nos dá a sequência das 30 amostras antes do início do telegrama e o número de amostras configurado pelo módulo *host*. É verificado então se o banco SHARED RAM ainda terá espaço para armazenar mais amostras, caso contrário o endereço que irá gravar nesse banco volta a zero. É sinalizado para o módulo *host* que um novo conjunto de amostras já está presente na memória, iniciando novamente o ciclo de amostragem.

4.6.2.4 Nível 1: Software do Usuário

Os objetivos do software do usuário está fora do contexto desse trabalho, no qual o usuário dessa ferramenta poderá usar os dados coletados para processa-los da maneira que desejar. Porém, para a obtenção desses dados é necessário seguir certos passos, que estão descritos no nível 1 desse módulo, representados na Figura 4-6.

Para iniciar a comunicação, o software do usuário deve comunicar com o servidor TCP/IP criado pelo módulo host, sendo executado pela BeagleBone. É necessário configurar corretamente o IP e a porta informado pelo software executado pela BeagleBone.

Depois da conexão estabelecida com o servidor, é necessário enviar para o módulo *host* o pacote de dados correspondente a configuração da ferramenta, especificado no capítulo referente ao modo de uso da ferramenta. Essas configurações irão definir se o módulo analógico será utilizado, quantas amostras serão salvas após o início do telegrama, quanto tempo é necessário esperar para atualizar as amostras de um endereço e também quantos telegramas serão salvos pelo modulo digital, tendo a opção de um número indeterminado, que só irá parar

a operação dos módulos quando a conexão entre o software do usuário e o módulo *host* for perdida.

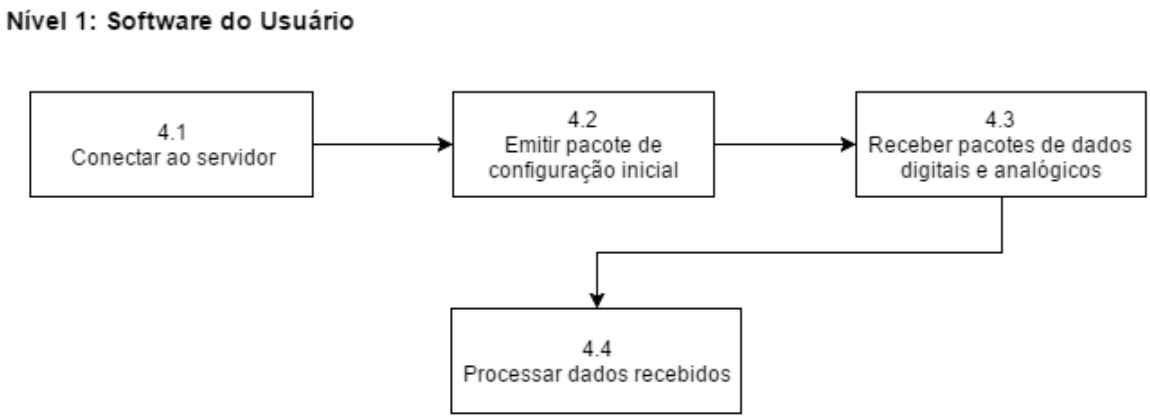


Figura 4-6 - DFD Nível 1 - Software do usuário.

4.6.3 Nível Dois

O nível dois de um DFD detalhe os processos do nível 1 em mais tarefas, expandindo seu funcionamento. Muitas das tarefas já foram bem detalhadas no texto da seção referente ao nível 1, porém muitas ainda não tiveram todo seu funcionamento explicado. Essas serão explicadas individualmente abaixo.

4.6.3.1 Nível 2: Módulo Host – Ciclo Principal

O processo “Ciclo Principal” apresentado no nível 1 do módulo host pode ser expandido em subprocessos, apresentados no nível 2 do DFD pela Figura 4-7.

O processo “Ciclo Principal” irá aguardar alguma das interrupções vindas dos núcleos PRU. Quando recebida a interrupção, o programa identificará se ela veio do módulo digital ou do módulo analógico.

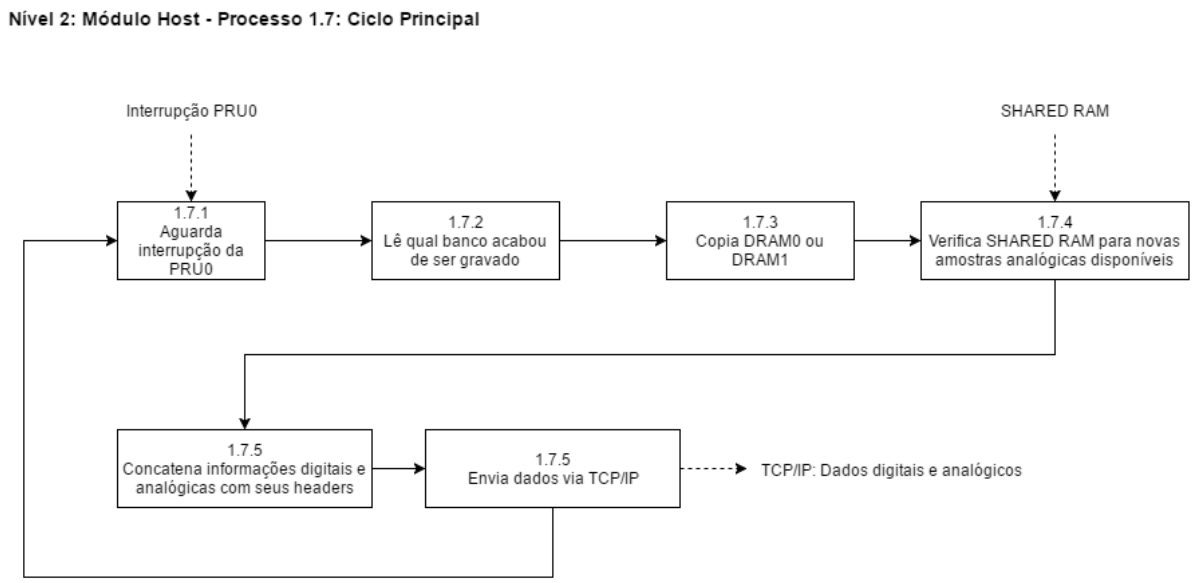


Figura 4-7 - DFD Nível 2: Módulo Host, processo "Ciclo Principal".

Se a interrupção veio do módulo digital, o processo verifica no cabeçalho de configuração qual banco de memória acabou de ser preenchido (DRAM0 ou DRAM1). Esse banco é inteiro copiado para uma variável declarada no módulo *host*, afim de evitar o sobrecarregamento do barramento L3/L4, executando todos os acessos de memória de uma vez, ao invés de vários fragmentados pela execução.

Como todo o banco de memória utilizado foi copiado, as informações presentes no cabeçalho de configuração dele também foram copiadas. É então verificado nesse cabeçalho de configuração até qual endereço do banco de memória copiado foi escrito. Todos os dados presentes entre o endereço inicial de escrita, até esse endereço final são então enviados via TCP/IP para o software do usuário, acrescentando o protocolo descrito no capítulo de instruções de uso dessa ferramenta. Isso irá informar ao software do usuário que a mensagem recebida contém informações sobre telegramas coletados pelo módulo digital.

Porém, se a interrupção detectada veio do módulo analógico, a tarefa do módulo host irá copiar o banco de memória SHARED RAM, verificando também o espaço reservado para o cabeçalho de configuração deste banco. É verificado qual foi o endereço de início e qual o endereço de fim que acabou de ser atualizado, para então reservar os dados presentes dentro dessa faixa e encaminhar para o software do usuário via TCP/IP, acrescentando o protocolo definido nas instruções de uso nessa ferramenta para identificar uma mensagem contendo informação de amostras analógicas.

Essa operação continua em ciclo contínuo até que a conexão TCP/IP estiver aberta. Ao acontecer a interrupção da conexão, os módulos PRU são desativados e o programa volta para a fase de espera da conexão, aguardando um novo cliente requisitar conexão do servidor TCP/IP.

4.6.3.2 Nível 2: Módulo Digital – Detectar Baud Rate

O processo “Detectar Baud Rate” apresentado no nível 1 do módulo digital possui a função de descobrir por meio da transmissão de mensagens na rede Profibus qual a taxa de transmissão que a rede opera. Existe a possibilidade de reconfiguração da rede durante a execução dessa ferramenta, portanto essa função poderá ser chamada novamente caso mais de 3 caracteres com algum erro for detectado pelos processos posteriores a este. Seu funcionamento está apresentado na Figura 4-8.

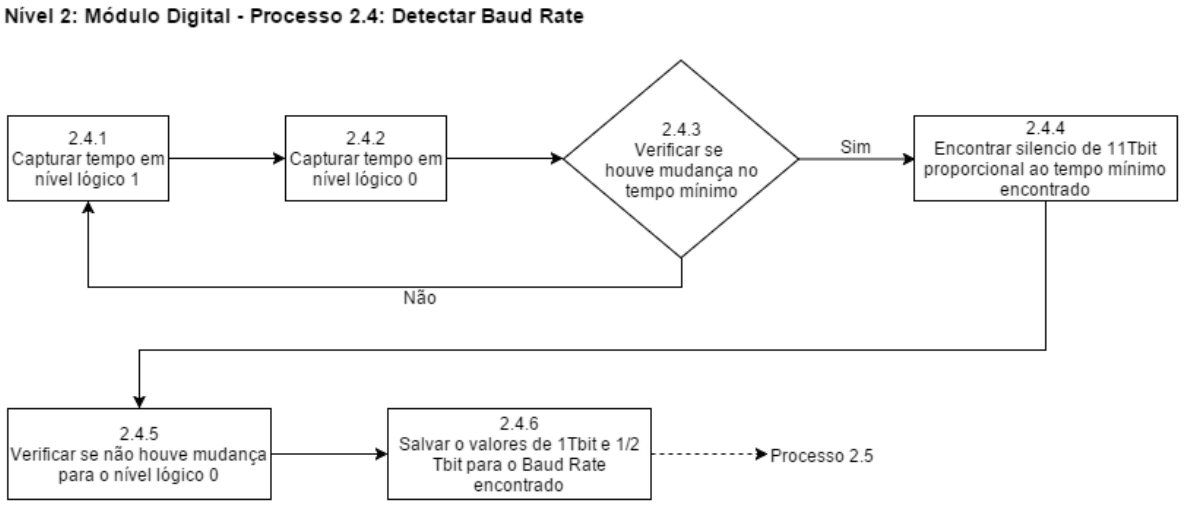


Figura 4-8 - DFD Nível 2: Módulo Digital, processo "Detectar Baud Rate".

O processo inicia esperando o nível lógico do sinal que chega no GPIO P9_27 ficar em 1. Um *timer* é disparado, e só irá recomeçar sua contagem quando ocorrer a mudança de nível lógico, de 1 para 0. Quando essa mudança ocorrer, o valor que o *timer* continha é comparado com um registrador de uso geral, que salva sempre o menor valor entre o valor já salvo nele e o valor contabilizado pelo *timer*. O mesmo processo é feito para o nível lógico 0, contando o tempo por meio de um *timer* até ocorrer a mudança do nível lógico do sinal ligado ao pino. Assim que a alteração de nível lógico ocorre, agora de 0 para 1, o valor do timer é novamente

comparado com o registrador de uso geral, sendo salvo neste se o valor obtido no *timer* for menor que o valor presente. O próximo passo compara o valor mínimo que estava salvo com o valor atual de mínimo. Se ocorreu mudança, volta-se ao primeiro passo desse processo, até não haver mais mudança no valor mínimo encontrado. Se não houve mudança, é aguardado então encontrar um estado de *idle* na rede proporcional ao valor mínimo encontrado.

O menor tempo de *idle* em uma rede Profibus é dado como o tempo de 11 T_{bit}, calculado pela equação abaixo:

$$T_{bit} = \frac{1}{Baud\ Rate}$$

Esse processo não só espera até encontrar esse tempo de intervalo em *idle* da rede, mas também continua procurando um valor mínimo diferente do valor já encontrado. Se nessa etapa for encontrado um valor mínimo diferente, o processo volta ao primeiro passo. Se o tempo de *idle* de no mínimo 11 T_{bit} foi encontrado, significa que o Baud Rate foi encontrado com sucesso. Isso fará com que o processo salve no cabeçalho de configuração o número do Baud Rate encontrado, que varia de 1 a 11 em ordem decrescente, sendo 1 o Baud Rate de 12Mbps. O valor 0 será salvo se o Baud Rate encontrado não pertencer aos padrões da Profibus, voltando para o passo 1 desse processo. São salvos em registradores de uso geral o tempo equivalente a meio T_{bit}, 1 T_{bit} e 11 T_{bit}, que serão utilizados pelo processo que faz a captura dos caracteres.

Sempre será utilizado apenas um telegrama completo para a detecção do Baud Rate. Isso devido aos valores constantes escolhidos para o protocolo que simbolizam os *Start Delimiters* e o *End Delimiter*, sendo os valores 10h, 68h, A2h, DCh e 16h os possíveis valores dos campos citados. Conforme mostrado pela Figura 4-9, todos esses valores possuem pelo menos um bit que o estado é alterado em um T_{bit}. Portanto, esse processo irá detectar esse menor valor em algum dos *Start Delimiters*, caso o início do processo se dê dentro do estado idle da rede, ou então ele detectará o *End Delimiter*, que logo depois possui um silêncio, que é utilizado também nesse processo de detecção.

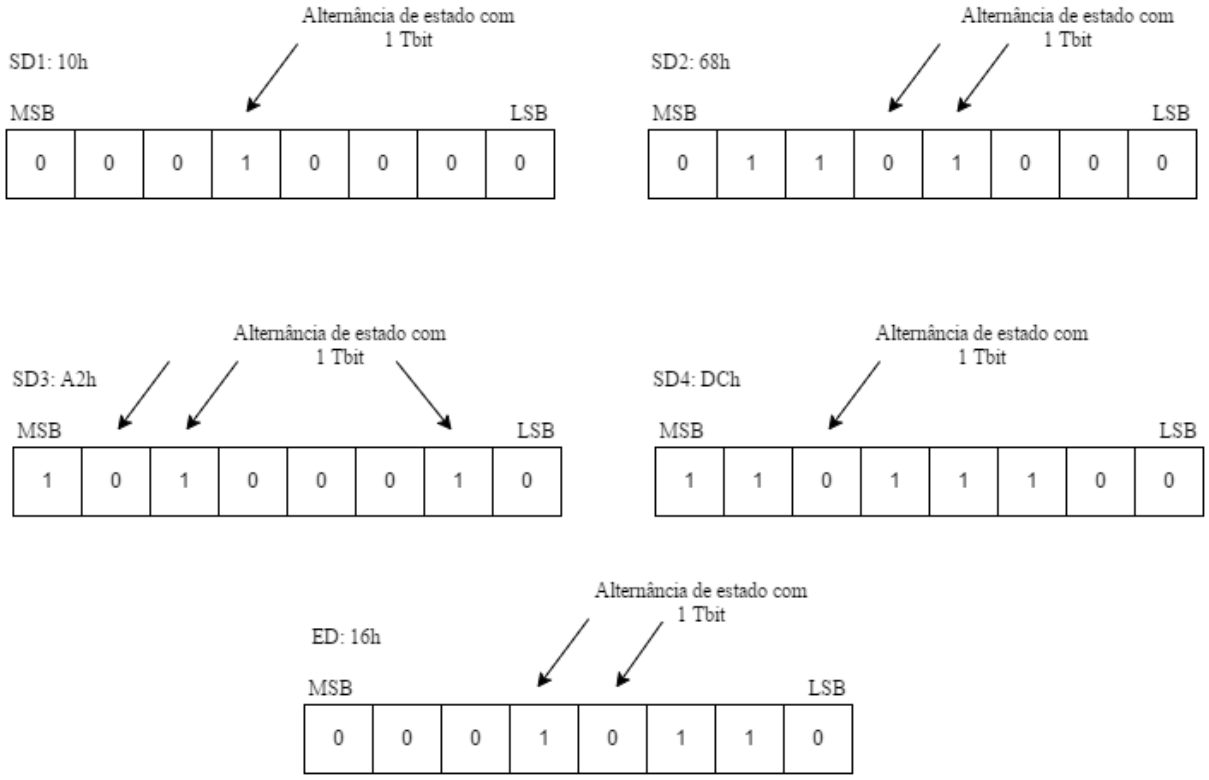


Figura 4-9 - Alternâncias de nível lógico nos SD1, SD2, SD3, SD4 e ED.

4.6.3.3 Nível 2: Módulo Digital – Captura caracteres

O processo "Captura caracteres" contido no módulo digital, apresentado no nível 1 desse DFD, possui um funcionamento que necessita de mais um nível de detalhamento para a sua compreensão. Este funcionamento detalhado está representado na Figura 4-10.

A primeira vez que esse processo se inicia, um silêncio acabou de ser encontrado pelo processo "Detectar Baud Rate". Portanto, é necessário que um Baud Rate válido esteja salvo no cabeçalho de configuração, juntamente com os valores de tempo de meio T_{bit}, 1 T_{bit} e 11 T_{bit}. O processo espera que o nível lógico lido na porta P9_27 saia de 1 e passe para zero, indicando um start bit, significando também o início do primeiro caractere de um telegrama. Após essa detecção, é salvo o *timestamp* equivalente ao início da transmissão do telegrama e também é feita a alteração no cabeçalho de configuração, indicando ao módulo analógico que foi identificado o início da transmissão de um telegrama. É aguardado o tempo equivalente a meio T_{bit} do Baud Rate detectado. Finalizada essa espera, é lido se o estado do pino ainda continua em 0. Caso tenha mudado para 1, a captura do telegrama é interrompida e começa-se o processo novamente, além de somar um no contador de erros seguidos. Se o valor permaneceu em zero, prossegue-se com a leitura do campo de dados do caractere transmitido.

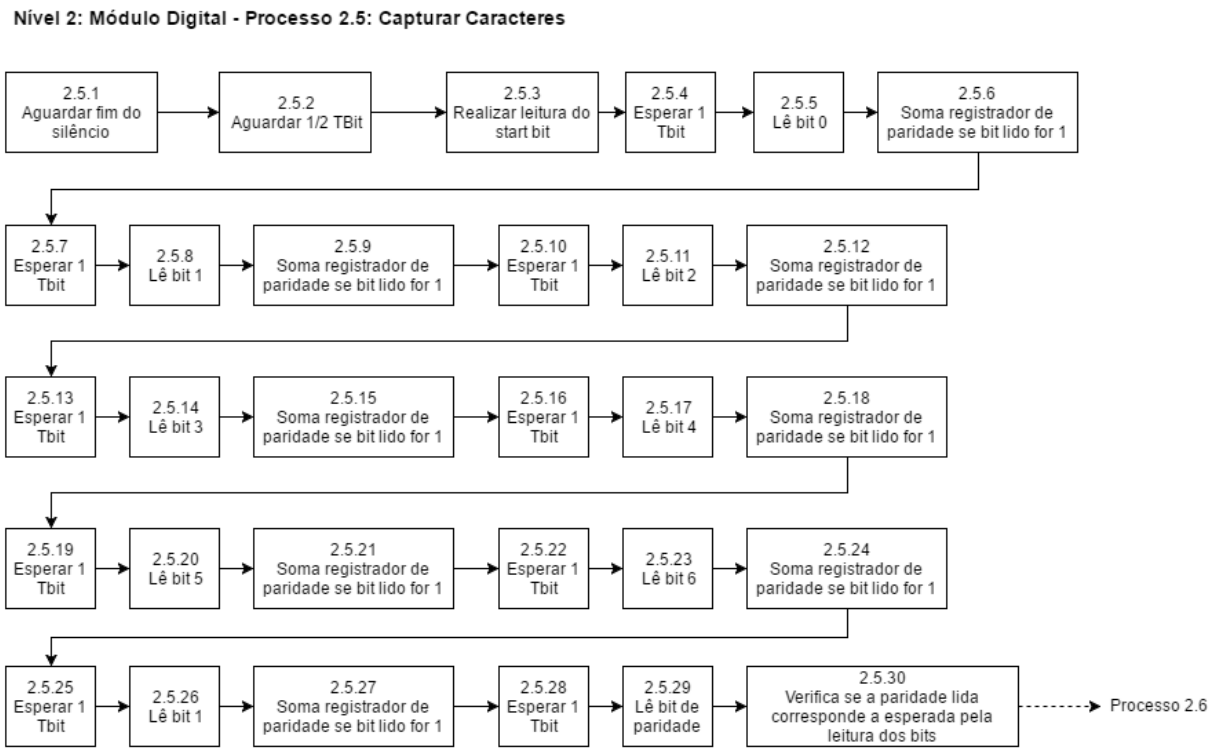


Figura 4-10 - DFD Nível 2: Módulo Digital, processo "Captura caracteres".

Cada passo agora é aguardado o tempo equivalente a 1 T_{bit}, e então a leitura do estado do pino é realizada, fazendo o preenchimento da posição equivalente do bit em um registrador de uso geral que irá montando o byte transmitido. Isso é feito dos bits 0 a 7. Juntamente com a leitura, cada vez que um bit possui o valor binário 1, é somado o valor 1 em um registrador de uso geral, que será utilizado para a detecção do bit de paridade. Após a leitura do bit 7 e sua soma caso o valor seja 1, o próximo bit lido será o de paridade. É realizada a leitura do valor do bit de paridade, e então comparada com o bit menos significativo do registrador geral responsável pela detecção do bit de paridade. O valor contido nesse bit menos significativo deverá ter o mesmo valor do bit de paridade lido. Se os valores não forem os mesmos, a etapa de erro será chamada, indicando erro de paridade e somando um no contador de erros seguidos. Finalmente é aguardado mais 1T_{bit}, e a leitura realizada deve possuir o valor 1, caso contrário é sinalizado um erro de stop bit.

Depois do stop bit, o processo entra em um estado que ele conta quanto tempo levará a troca de estado do bit 1 para o bit 0. Se essa troca ocorrer em menos de 11 T_{bit}, significa que a transmissão do telegrama ainda não acabou, e outro caractere dele será lido, voltando a etapa dois desse processo. Se o valor de tempo alcançar 11 T_{bit}, significa que a rede está em um

estado de *idle*, e o envio do telegrama foi encerrado. Isso fará com que o processo “Captura caracteres” termine, dando continuidade ao ciclo.

4.6.3.4 Nível 2: Módulo Analógico – Aguarda início da Captura do Telegrama

O detalhamento do processo “Aguarda início da Captura do Telegrama”, presente no nível 1 do módulo analógico está representado na Figura 4-11. Sua primeira tarefa é verificar o cabeçalho de configuração para identificar se um telegrama começou a ser transferido. Essa informação é alterada pelo processo “Captura caracteres” do módulo digital. Enquanto esse valor permanece em zero, indicando que a rede Profibus permanece em estado de *idle*, esse processo entra em *loop*, sendo o primeiro passo a alternância entre os *words* dos registradores de uso geral entre R10 e R24, totalizando 30 *words*. O *word* menos significativo do registrador R24 passa para o a faixa mais significativa desse mesmo registrador, enquanto o valor que estava na faixa mais significativa de R24 passa para o *word* menos significativo de R23, e assim por diante, até o valor mais significativo de R10 ser descartado.

Após essa alternância entre as amostras salvas nos registradores R10 até R24, é feita a captura da amostra analógica vinda do subsistema ADC e salvando no *word* menos significativo do registrador de uso geral R24. Então o processo volta ao início do ciclo e é verificado se a transmissão de um telegrama já teve início. Caso afirmativo, o processo salva o valor zero no campo do cabeçalho de configuração que registra o início da transmissão de um telegrama e começa assim o processo “Salvar o número de amostras configuradas”, já descrita no nível 1 do DFD do módulo analógico.

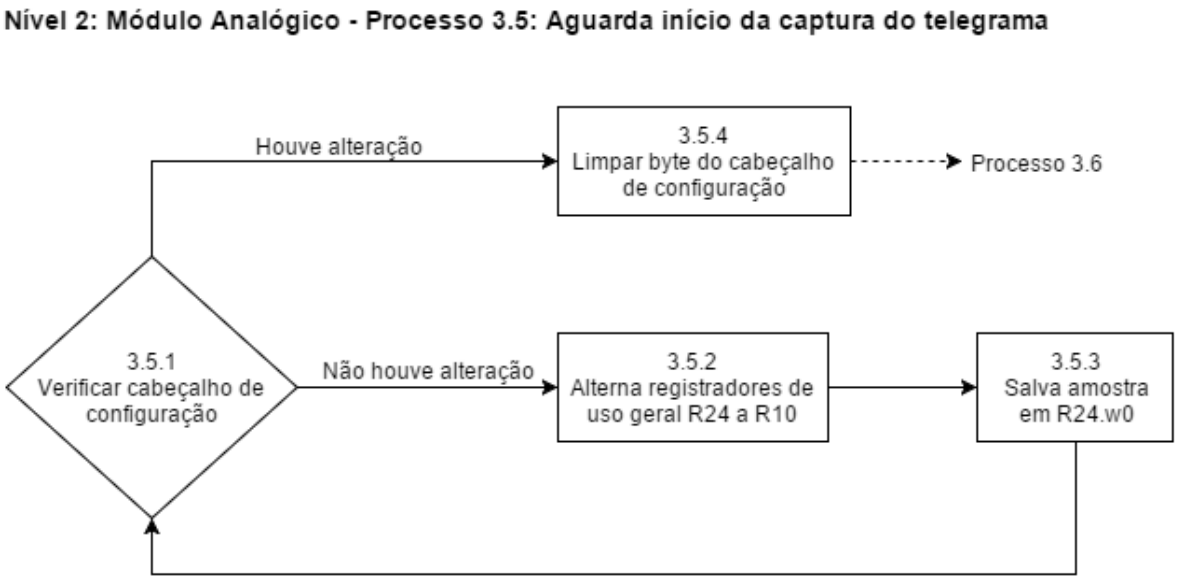


Figura 4-11 - DFD Nível 2: Módulo Analógico, processo "Aguarda início da Captura do Telegrama".

5 INSTRUÇÕES DE USO DA FERRAMENTA

Este capítulo fornecerá as instruções necessárias para os usuários que desejam utilizar a ferramenta para suas próprias aplicações. Recomenda-se que a ferramenta seja executada na mesma rede local da aplicação, pois será usado constantemente até 6Mbps dependendo das configurações escolhidas para a ferramenta, podendo causar grande perda de pacotes se for utilizada a transmissão de dados fora da mesma rede local. O nome escolhido para a ferramenta desenvolvida foi ProfiLAI.

5.1 Requisitos para a BeagleBone

Conforme já especificado em outros capítulos, para a execução da ferramenta de captura da rede Profibus na BeagleBone são necessários os seguintes requisitos:

- BeagleBone Black Rev. C, conectada a uma rede Ethernet local.
- *AM335x PRU Package* e seus drivers devidamente instalados.
- Qt5 para Linux-ARM instalado.

Com essas ferramentas já em funcionamento na BeagleBone, o programa ProfiLAI poderá ser executado. Os endereços locais de rede disponíveis na BeagleBone para aquela conexão serão exibidos assim que a ferramenta for executada. O software desenvolvido pelo usuário deverá conectar a um desses IPs locais, na porta 3100.

5.2 Requisitos para o software do usuário

O usuário deverá verificar se os requisitos abaixo são cumpridos, para conseguir receber todos os dados enviados pela ferramenta de captura:

- Desenvolvimento em uma linguagem de programação que dê suporte ao protocolo TCP/IP. Todos os dados devem ser enviados e recebidos na ordenação de Little-Endian (bytes menos significativo primeiro).
- Rede Ethernet local estável que suporte comunicações de 10Mbps.

- Seguir todas as regras de configuração, início e leitura da ferramenta, descritas abaixo.

5.3 Protocolo de dados da ferramenta

Se o usuário confirmar que os requisitos descritos acima forem cumpridos, a comunicação de dados entre a ferramenta de captura e o software que processará os dados será realizada de forma correta. Porém, para o início da coleta dos dados o software do usuário deve primeiramente configurar a ferramenta ProfiLAI, que começará então a enviar os dados coletados para o usuário. Os passos para essa configuração e para a interpretação dos dados estarão descritos nas seções abaixo.

5.3.1 Conexão com a ferramenta

O software do usuário deverá estabelecer uma conexão TCP/IP com a ferramenta por meio de um *socket*. A ferramenta funciona como um servidor TCP/IP, escutando e aceitando todos os pedidos de conexão recebidos na porta 3100. Apenas uma conexão é permitida por vez, portanto o usuário deve ficar atento de estabelecer a conexão apenas no início de seu programa e mantê-la ativa durante todo o seu funcionamento.

Caso a conexão seja interrompida ou uma nova conexão for feita, a captura será parada e a conexão antiga será cortada, dando agora a vez para a nova conexão.

5.3.2 Configuração e início da captura

Assim que uma nova conexão for estabelecida, a ferramenta entra em modo de espera até receber uma ordem de configuração, para então iniciar sua captura na rede. A configuração da ferramenta é feita a partir de um pacote de dados de 17 bytes, enviados do software do usuário para a ferramenta ProfiLAI com a ordenação dos bytes em Little-Endian. O pacote deverá conter as seguintes informações:

- 4 bytes de Header: O header de configuração indica ao programa que o pacote recebido é uma configuração de início de captura. O número que representa esse header em hexadecimal é o **0xDC686816**.

- 4 bytes para o número de pacotes digitais capturados: O usuário tem a possibilidade de configurar quantos pacotes de dados digitais deseja receber. Cada pacote de dados digitais contém cerca de 50 telegramas da rede Profibus. Quando a ferramenta concluir o envio desse número de pacotes, ela entrará no modo de espera novamente, esperando uma nova configuração para capturar e enviar novos dados. Se o usuário deseja receber indefinidamente, basta enviar o número 4.294.967.295 (ou -1), que corresponderá ao maior valor que pode ser salvo em 4 bytes.
- 1 byte indicando a utilização do módulo analógico: O usuário deverá enviar o número 0 para desabilitar a captura analógica e o número 1 para habilitar a captura analógica. Independente de desabilitar o módulo analógico, os próximos 8 bytes deverão ser enviados, podendo contar qualquer número.
- 4 bytes para o número de amostras feitas por captura analógica: se o usuário habilitou a captura analógica e a rede estiver em um Baud Rate menor ou igual a 500kbps, no primeiro start bit do telegrama será capturado 30 amostras antes desse start bit (período de *idle* da rede) mais o número configurado neste parâmetro. Recomenda-se não utilizar um número maior que 2000 nesse parâmetro, pois a ocupação da rede Ethernet e o número de dados para o processamento pelo software do usuário aumenta drasticamente com esse parâmetro.
- 4 bytes indicando o tempo em milissegundos entre amostras analógicas de um mesmo endereço: esse parâmetro irá ignorar as amostras feitas para um mesmo endereço até que o tempo aqui definido não seja atingido. Por exemplo: mesmo que uma estação transmita a cada 400 microssegundos, o pacote de amostras analógicas atualizado só será enviado ao software do usuário em intervalos definido por esse parâmetro.

Quando o software do usuário enviar ao programa ProfiLAI esse pacote de 17 bytes, a ferramenta irá analisar o header e as informações contidas. Caso todas elas forem válidas, a ferramenta começará a captura da rede conforme as configurações passadas e irá mandar para o software do usuário conectado os pacotes contendo as informações coletadas da rede. Abaixo está dois casos de pacotes de configuração que podem ser enviados para a ferramenta:

Exemplo 1: Número de pacotes digitais indefinidos (enviar os pacotes até receber um pedido de parada). Módulo analógico habilitado, enviando 100 amostras analógicas a cada

captura após o primeiro start bit do telegrama, com intervalo de 2 milissegundos a cada captura analógica por endereço. O pacote de configuração que o software de usuário deve enviar a ferramenta está exibido na Figura 5-1.

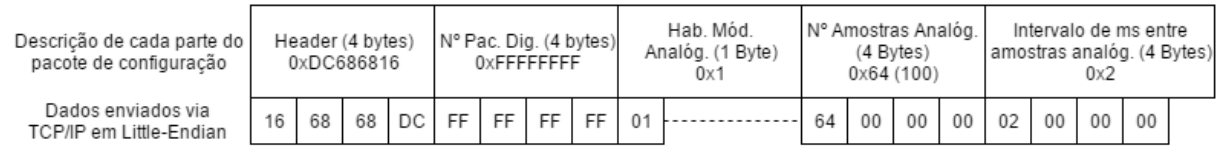


Figura 5-1 - Exemplo do pacote de configuração com o número indefinido de pacotes digitais, módulo analógico habilitado, com 100 amostras analógicas coletadas após o primeiro start-bit do telegrama e intervalos de atualização de 2ms para cada endereço.

Exemplo 2: Enviar apenas 50 pacotes digitais. Módulo analógico desabilitado. Os parâmetros de número de amostras e intervalo entre capturas podem apresentar qualquer valor, porém necessariamente devem ser enviados em seus tamanhos originais (4 bytes cada). O pacote de configuração enviado para esse caso está representado na Figura 5-2.

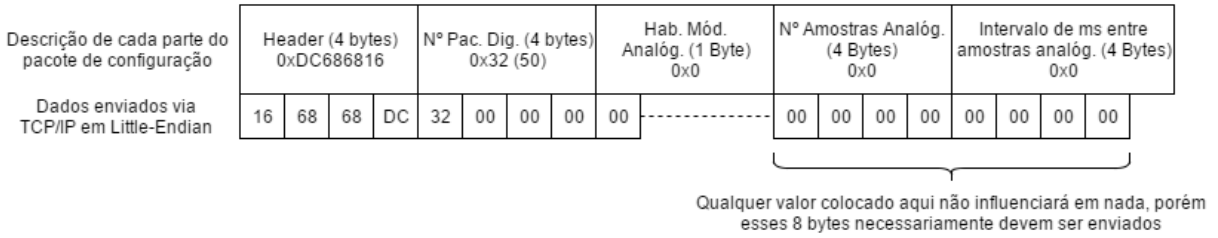


Figura 5-2 - Exemplo do pacote de configuração que fará a captura de 50 pacotes digitais, com o módulo analógico desabilitado.

5.3.3 Parada da captura

Se o software do usuário deseja parar de receber os pacotes de dados coletados da rede (digitais e analógicos), ele poderá enviar o pacote que indica para a ferramenta parar de capturar os dados da rede. Após essa parada, a ferramenta volta para o modo de espera. É necessário enviar apenas os 4 bytes com o valor de **0xE51010A2**, conforme mostrado na Figura 5-3. Esse comando deve ser usado também caso o usuário deseje trocar os parâmetros de configuração enquanto está acontecendo uma captura, enviando os 4 bytes indicando a parada e depois enviando os 17 bytes da nova configuração.

Descrição de cada parte do pacote de parada de captura	Header de parada (4 bytes) 0xE51010A2			
Dados enviados via TCP/IP em Little-Endian	A2	10	10	E5

Figura 5-3 - Pacote de parada de captura. Deverá ser enviado caso o software do usuário deseje parar de receber pacotes de dados ou deseje iniciar uma captura com configurações diferentes da que está acontecendo.

5.3.4 Interpretação do pacote de dados

Quando um pacote de configuração de início de captura válido é enviado à ferramenta ProfiLAI, a ferramenta inicia os núcleos PRU com os parâmetros passados pela configuração recebida, começando assim a captura dos pacotes digitais e, caso ativado, a captura das amostras analógicas. Esses dados serão enviados em grandes pacotes da ferramenta ProfiLAI para o software de usuário, que irá receber esse fluxo de dados, ordenados também em Little-Endian.

Os quatro primeiros bytes recebidos indicam o header do pacote que irá ser interpretado. Esse pacote poderá conter amostras digitais ou amostras analógicas, cada um possuindo o seu modo para serem interpretados. São eles:

5.3.4.1 Pacote de dados digitais

Recebendo o header **0xDCDCA216**, o software do usuário deverá entrar no modo de interpretação de um pacote de dados digitais. Logo após esses primeiros quatro bytes de header, virá dois bytes indicando o tamanho do pacote digital que irá ser interpretado, que será chamado nessa etapa de *nBytes*. Sabemos, portanto, que os próximos *nBytes* pertencem a esse pacote. Dentro desse pacote recebido, estarão contidos os telegramas da rede Profibus, que contém informações sobre o Baud Rate quando ele foi coletado, se aconteceu algum erro de transmissão, qual o tamanho do telegrama, os dados desse telegrama, um timestamp do início da transmissão desse telegrama e um timestamp de final do telegrama. Portanto, a etapa de interpretar o pacote de dados digitais deve realizar essa leitura de telegramas até acabar os *nBytes* do pacote digital que está sendo lido. Sempre o número de *nBytes* corresponderá exatamente o número total de bytes dos telegramas, sem faltar e nem sobrar bytes dentro desse

pacote de dados digitais. Caso isso aconteça, um erro na transmissão TCP/IP aconteceu ou um erro na interpretação do pacote digital aconteceu. Após os dois bytes indicando o tamanho desse pacote de dados digital, é enviado 4 bytes indicando o identificador desse pacote digital. Esse campo serve para identificar se nenhum pacote digital foi perdido, pois sempre depois de uma nova configuração esse identificador inicia em 1 e é acrescido a cada pacote digital enviado. Portanto se o programa identificar a alteração maior que um entre dois pacotes digitais, significa que um ou mais pacotes digitais não foi lido ou recebido pela transmissão TCP/IP. Após esse identificador do pacote digital, inicia-se assim a transmissão dos telegramas, que seguem os padrões de dados que serão especificados abaixo, juntamente com seus tipo e tamanho.

- 1 byte: o valor do primeiro byte representa duas informações: nos 4 bits mais significativos ficará em 0 caso nenhum erro de transmissão (start bit, paridade ou stop bit) na rede Profibus aconteceu durante esse telegrama. Os 4 bits menos significativos representam o Baud Rate, no qual o número formado por ele representa um dos valores da tabela abaixo:

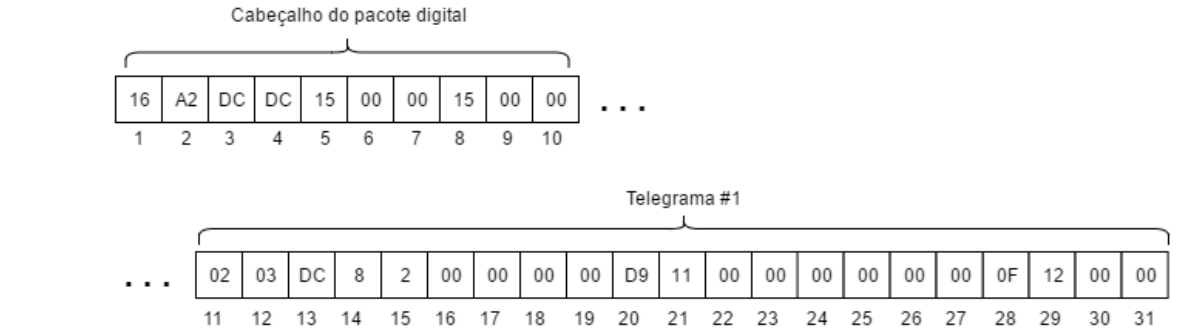
Número lido nos 4 bits menos significativos	Baud Rate
0	Baud Rate desconhecido
1	12 Mbps
2	6 Mbps
3	3 Mbps
4	1.5 Mbps
5	500 kbps
6	187.5 kbps
7	93.7 kbps
8	45.45 kbps
9	19.2 kbps
10	9.6 kbps

Tabela 5-1 - Relação dos 4 bits menos significativos do primeiro campo de cada telegrama com o Baud Rate.

- 1 byte indicando o tamanho do telegrama que foi transmitido na rede.
- N bytes indicando os dados transmitidos no telegrama lido na rede, sendo N o número de bytes que foi lido no passo anterior.
- 8 bytes que representam um inteiro de 64 bits, correspondente ao timestamp de início de envio do telegrama.

- 8 bytes que representam um inteiro de 64 bits, correspondente ao timestamp de término de envio do telegrama.

Exemplo de um telegrama lido da rede: o telegrama enviado é de passe de token (SD4), com 8 de endereço de destino e 2 de endereço de envio, sem erros de transmissão e com Baud Rate de 6Mbps. Supondo que um pacote digital contenha somente esse telegrama, iriamos ler esse pacote da maneira apresentada na Figura 5-4. O timestamp representa valores desde o início da captura configurada, sendo que cada valor acrescido nele representa o valor de 5 nanossegundos.



Cabeçalho do pacote digital:

Byte 1 a 4: 0xDCDCA216
Header dizendo que as próximas informações são dados digitais

Byte 5 e 6: 0x0015
Sinaliza que após o cabeçalho terão 21 bytes de telegramas

Byte 7 a 10: 0x00001500
Indica que esse é o pacote 5376
Usado para a verificação se o último pacote foi o 5375 e o próximo será o 5377, garantindo que nenhum pacote foi perdido

Telegrama #1:

Byte 11: 0x02
Bit 0-3: 0x2: Baud Rate de 6Mbps
Bit 4-7: 0x0: Nenhum erro de transmissão no telegrama

Byte 12: 0x3
Os três próximos bytes fazem parte dos dados do telegrama

Byte 13, 14 e 15: 0xDC, 0x8, 0x2
Telegrama transmitido pela rede Profibus
Passe de Token do endereço 2 para o endereço 8

Byte 16 a 19: 0x00000000
Parte alta do timestamp de início da transmissão do telegrama

Byte 20 a 23: 0x000011D9
Parte baixa do timestamp de início da transmissão do telegrama

Byte 24 a 27: 0x00000000
Parte alta do timestamp de término da transmissão do telegrama

Byte 28 a 31: 0x0000120F
Parte baixa do timestamp de término da transmissão do telegrama

Figura 5-4 - Exemplo de um pacote digital que possui apenas um telegrama de troca de token.

No caso de haver mais telegramas dentro de um mesmo pacote de dados digitais, ele viria logo depois do primeiro telegrama e assim por diante, sendo a soma do tamanho de todos os campos de telegrama o mesmo tamanho especificado no cabeçalho do pacote digital.

5.3.4.2 Pacote de dados analógicos

Seguindo a mesma lógica do pacote de dados digitais, o pacote analógico inicia com um header de valor **0x10166816**. Logo após esses 4 bytes de header, é enviado 4 bytes que representam o tamanho do pacote de dados analógicos que estará depois do cabeçalho.

Inicia-se assim a leitura desse pacote de dados. Ele é dividido em conjunto de amostras analógicas, enviadas pelo mesmo endereço. Os primeiros quatro bytes lidos é o ID do conjunto de amostra, que será incrementado por um a cada conjunto enviado, servindo para garantir que nenhum conjunto foi perdido pela transmissão TCP/IP. Logo depois é lido em apenas um byte o endereço do dispositivo que enviou o telegrama correspondente as amostras coletadas. Em seguida, é lido os 8 bytes correspondentes ao timestamp de início do envio do telegrama. Esse valor pode ser comparado com as amostras de dados digitais, para encontrar exatamente o telegrama que corresponde às amostras analógicas que serão lidas. Após esse timestamp, é lido 4 bytes que correspondem ao número de amostras que estão gravadas a partir do próximo campo. São lidas as 30 primeiras amostras, totalizando 60 bytes, que correspondem às amostras do estado de *idle* da rede, logo antes do primeiro *start-bit* do telegrama correspondente. Depois dessas 30 amostras, virão as N amostras definidas na configuração de início da captura. Se o número de bytes lidos ainda não totalizou o tamanho lido no cabeçalho do pacote de dados analógicos, isso significa que outro conjunto de amostra virá logo após a última amostra, repetindo até o número de bytes lidos de todos os conjuntos ser igual ao tamanho lido no cabeçalho desse pacote.

Os valores de cada amostra do conversor AD da BeagleBone podem ir de 0 a 4095, sendo proporcionais à tensão máxima do divisor de tensão ligado no circuito de captura das amostras analógicas.

A Figura 5-5 exemplifica a transmissão de um pacote de dados analógicos que só possui um conjunto de amostras, enviadas pelo endereço 2. Foi escolhida na configuração que o número de amostras após o start-bit seria 100 amostras, totalizando 130 amostras transmitidas.

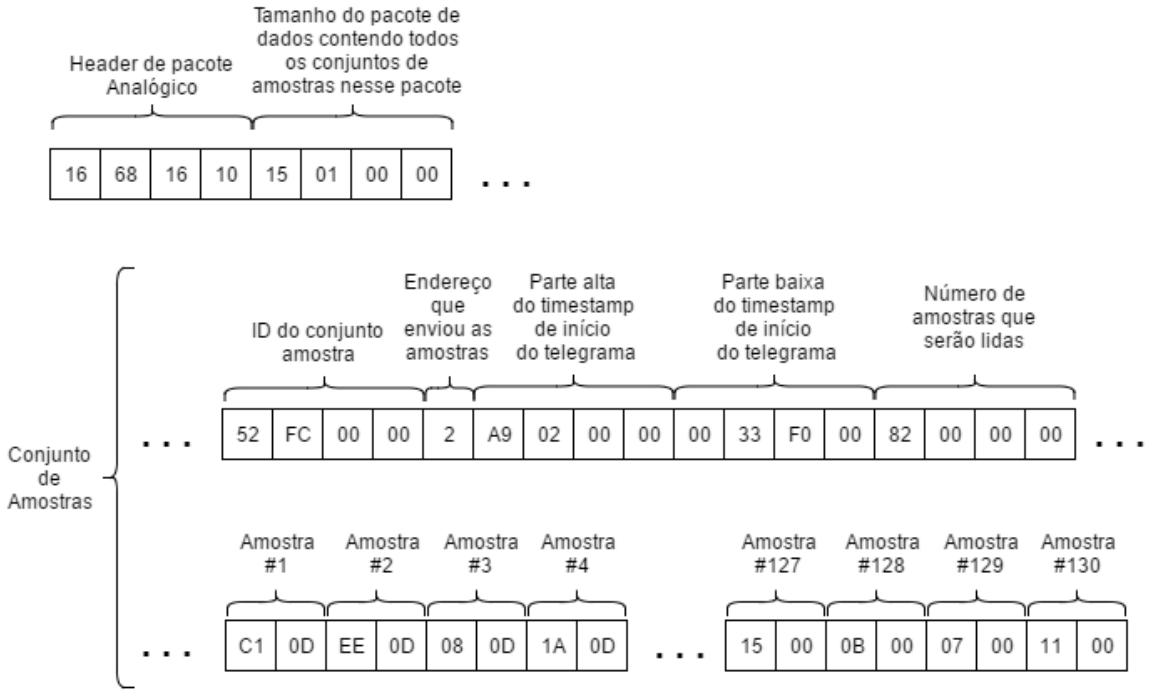


Figura 5-5 - Exemplo de um pacote de dados analógico que contém apenas um conjunto de amostras, enviado pelo endereço 2, contendo 130 amostras (0x82).

5.4 Resumo dos dados

A Tabela 5-2 mostra os dados que podem ser enviados para o programa ProfiLAI, com as suas respectivas funções, enquanto as Tabela 5-3 e Tabela 5-4 mostram os dados que são recebidos pelo software do usuário por meio dos pacotes de dados digitais e pacotes de dados analógicos respectivamente.

Valor do campo de dado	Tipo do campo de dado	Tamanho em bytes	O que representa o campo de dados
0xDC686816	Unsigned Int	4	Header de configuração.
0 a 4.294.967.295	Unsigned Int	4	Número de pacotes digitais que serão lidos.
0 ou 1	Unsigned Char	1	Habilitar/desabilitar o módulo analógico.
0 a 4.294.967.295	Unsigned Int	4	Número de amostras analógicas lidas por conjunto (recomendado menos de 2000).
0 a 4.294.967.295	Unsigned Int	4	Intervalo em milissegundos entre atualização de conjunto de dados analógicos para um mesmo endereço.
0xE51010A2	Unsigned Int	4	Comando de parada de captura.

Tabela 5-2 - Possíveis campos de dados que o software do usuário pode enviar ao ProfiLAI.

Valor do campo de dado	Tipo do campo de dado	Tamanho em bytes	O que representa o campo de dados
0xDCDCA216	Unsigned Int	4	Header do pacote digital.
0 a 65535	Unsigned Short	2	Número de bytes após o cabeçalho do pacote de dados digital.
0 a 4.294.967.295	Unsigned Int	4	ID do pacote digital recebido.
0 a 256	Unsigned Char	1	Byte que representa nos 4 bits menos significativos o Baud Rate que foi transmitido o telegrama e os 4 bits mais significativos se ocorreu um erro de transmissão (start-bit, paridade ou stop-bit)
0 a 256	Unsigned Char	1	Tamanho em bytes dos dados do telegrama
0 a 256	Unsigned Char	1	Bytes de dados do telegrama.
-	Unsigned Long Int	8	Timestamp de início do telegrama.
-	Unsigned Long Int	8	Timestamp de término do telegrama.

Tabela 5-3 - Possíveis campos de dados recebidos por um pacote de dados digitais.

Valor do campo de dado	Tipo do campo de dado	Tamanho em bytes	O que representa o campo de dados
0x10166816	Unsigned Int	4	Header do pacote analógico.
0 a 4.294.967.295	Unsigned Int	4	Número de bytes após o cabeçalho do pacote de dados analógico.
0 a 4.294.967.295	Unsigned Int	4	ID do conjunto de amostras digitais.
0 a 256	Unsigned Char	1	Endereço que enviou o conjunto de amostras analógicas.
-	Unsigned Long Int	8	Timestamp de início do telegrama.
0 a 4.294.967.295	Unsigned Int	4	Número de amostras que serão lidas nesse conjunto de amostras analógicas.
0 a 65565	Unsigned Short	2	Amostras analógicas (os valores lidos estarão na faixa de 0 a 4095).

Tabela 5-4 - Possíveis campos de dados recebidos por um pacote de dados analógicos.

6 RESULTADOS EXPERIMENTAIS

Utilizando a ferramenta desenvolvida, que teve todos seus passos de funcionamento descritos no capítulo 4 e as instruções de uso no capítulo 5, foi montado uma rede industrial com alguns dispositivos presentes no Laboratório de Automação Industrial (LAI). Foi programado um software que utiliza os dados coletados da rede para montar uma *live-list*, exibindo também as amostras analógicas capturadas de cada endereço, sendo este *software* o processo descrito como *software* do usuário descrito no nível zero do diagrama de fluxo de dados, na seção 4.6.1.

Abaixo estão as fotos e descrições da configuração da rede montada e também do funcionamento do software desenvolvido para a coleta dos dados.

6.1 Rede Profibus experimental

A rede Profibus montada para realizar os testes da ferramenta desenvolvida possui a seguinte configuração:

- Siemens S7-1200 CPU 1214C DC/DC/DC, com o cartão Siemens CM 1243-5 Profibus. Este CLP foi configurado como mestre da rede, no endereço 2, com os escravos descritos abaixo. Este equipamento está marcado com o número 1 na Figura 6-1.
- Siemens ET200, com um módulo de entradas digitais e um módulo de saídas digitais, no endereço 60 da rede Profibus. Dispositivo marcado com o número 2 na Figura 6-1.
- Altus 100V, com módulos de saída por relê, no endereço 33 da rede Profibus. Representado pelo número 3 na Figura 6-1.
- Escravo Profibus na placa Profichip utilizada como transciever Profibus, no endereço 7. Indicado pelo número 4 na Figura 6-1.

O item marcado com o número 5 na Figura 6-1 é a BeagleBone Black utilizada.

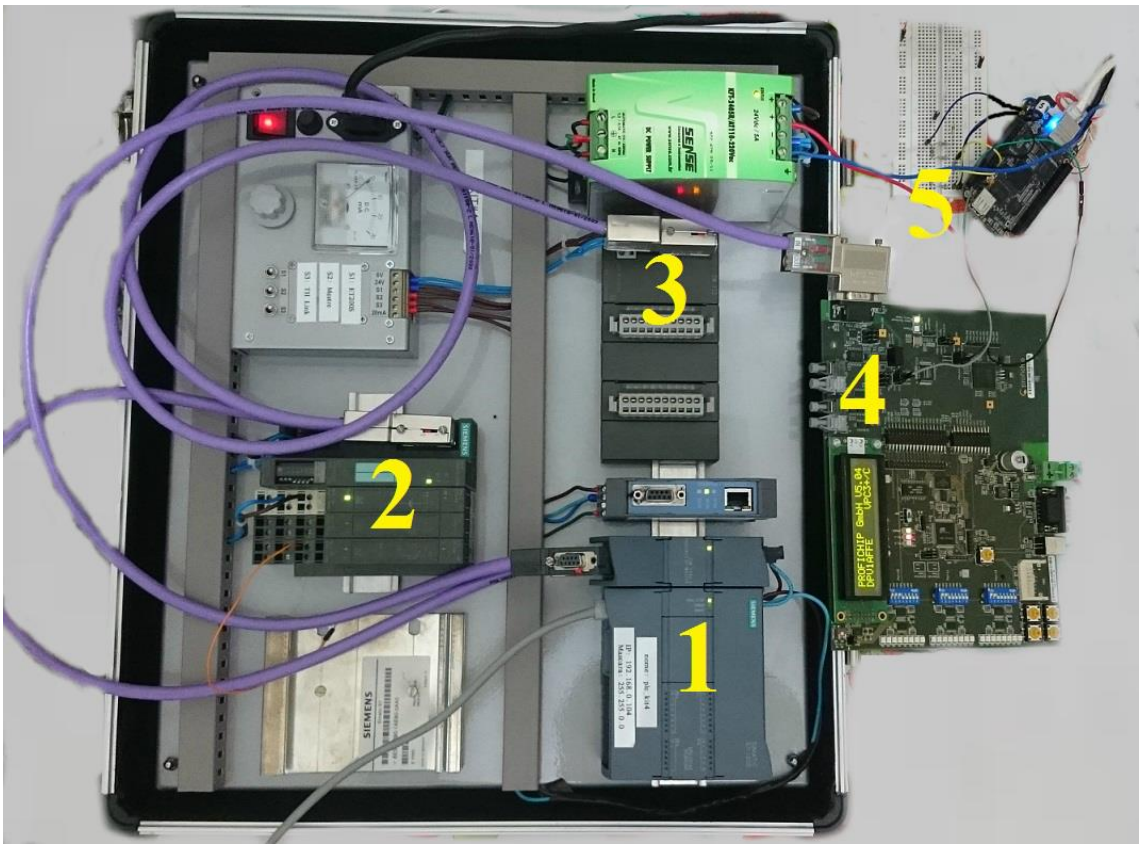


Figura 6-1 - Foto da rede configurada com os equipamentos listados acima.

6.2 Software experimental desenvolvido

Foi desenvolvido um software utilizando a linguagem C++ e a biblioteca Qt5 para Windows, que comunica com o servidor TCP/IP criado pelo software executado pela BeagleBone e segue os padrões de protocolo para interpretar os dados vindos dessa comunicação.

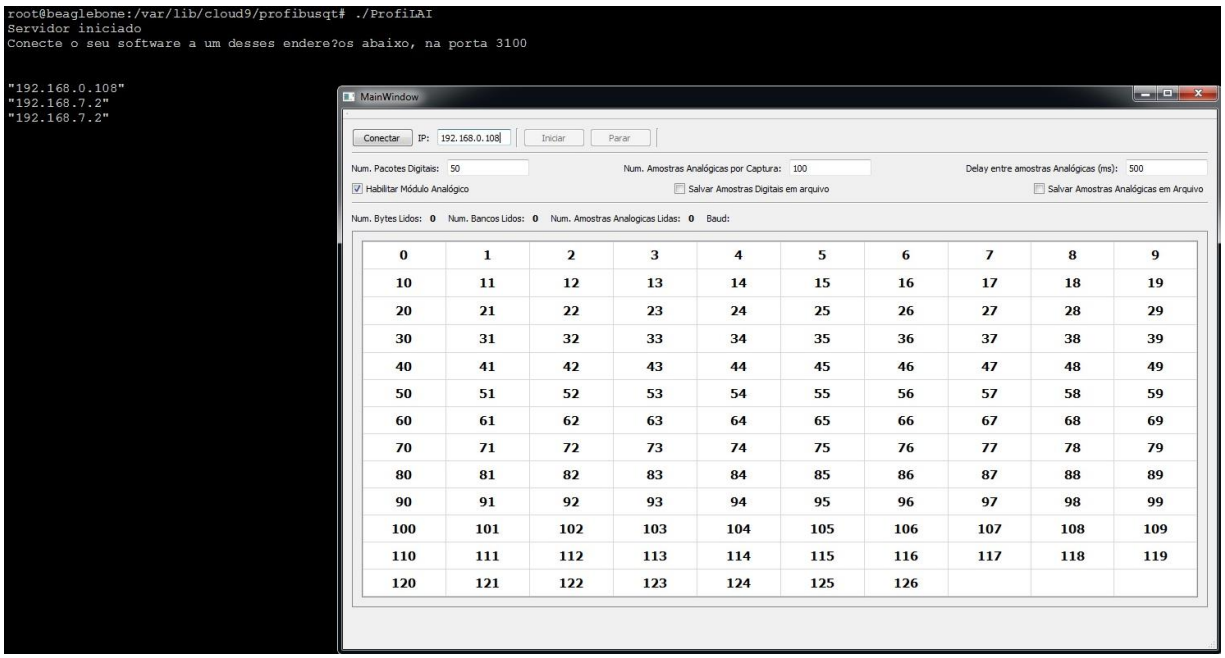
Esse software possui o objetivo de montar uma *live-list* a partir dos telegramas coletados da rede Profibus. A *live-list* mostra todos os endereços que comunicam na rede Profibus, tendo a possibilidade também de classificar esses dispositivos em mestres e escravos pelo campo FC dos telegramas SD1 e SD2. Os endereços pertencentes aos equipamentos energizados e conectados à rede são inseridos em uma tabela, seguindo a seguinte coloração: os mestres online são marcados na cor azul, os escravos online são marcados na cor verde e os endereços que foram detectados online em alguma captura, porém não estão mais comunicando, são marcados com a cor vermelha.

Outra função implementada nesse software é mostrar a forma de onda formada a partir das amostras analógicas capturadas da rede, se ela operar em um Baud Rate que permite essa captura. Com isso, ao clicar em um endereço disponível na *live-list*, as últimas amostras analógicas recebidas referentes ao endereço selecionado serão exibidas em forma de gráfico.

Nas seções a seguir serão mostradas as telas presentes no programa identificando suas funcionalidades.

6.2.1 Tela principal

Pode-se observar na Figura 6-2 o início de ambos os programas: o ProfiLAI, sendo executado na BeagleBone e visto no fundo por um terminal SSH e o software do usuário desenvolvido, com o campo de IP preenchido com o mesmo IP local exibido pelo servidor criado na BeagleBone.



0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109
110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126			

Figura 6-2 - Estado inicial do ProfiLAI e tela inicial do software do usuário desenvolvido.

Após estabelecer a conexão, a ferramenta ProfiLAI indica que a conexão foi recebida com sucesso, assim como visto na Figura 6-3. O software do usuário permite agora que os comandos de início de captura e de parada de captura sejam enviados para a ferramenta. As

configurações podem ser alteradas por meio dos campos de textos e *checkboxes* autoexplicativos no programa desenvolvido.

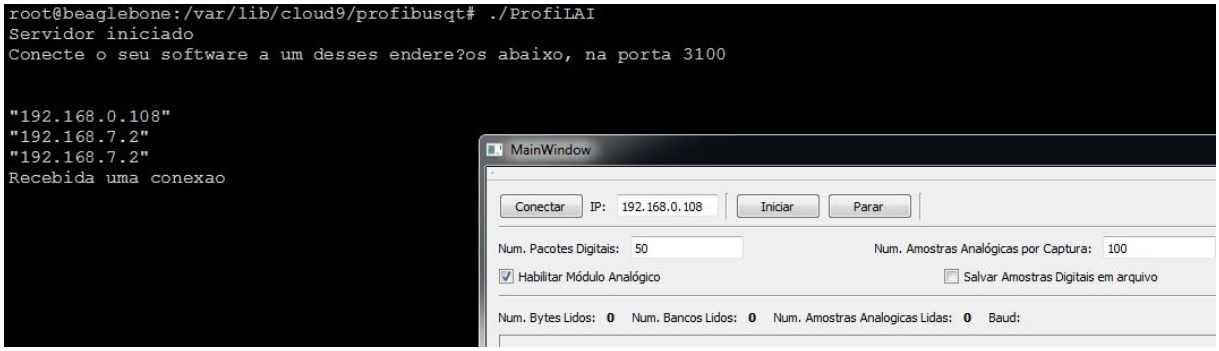


Figura 6-3 - Conexão estabelecida pelo software do usuário e o módulo host do ProfiLAI.

Após configurar os parâmetros de captura corretamente, a ferramenta ProfiLAI executa a captura até atingir o número de pacotes de dados digitais estabelecidos, ou então até receber um comando de parada. As telas apresentadas na Figura 6-4, Figura 6-5 e Figura 6-6 apresentam situações pós-captura de ambos os softwares.

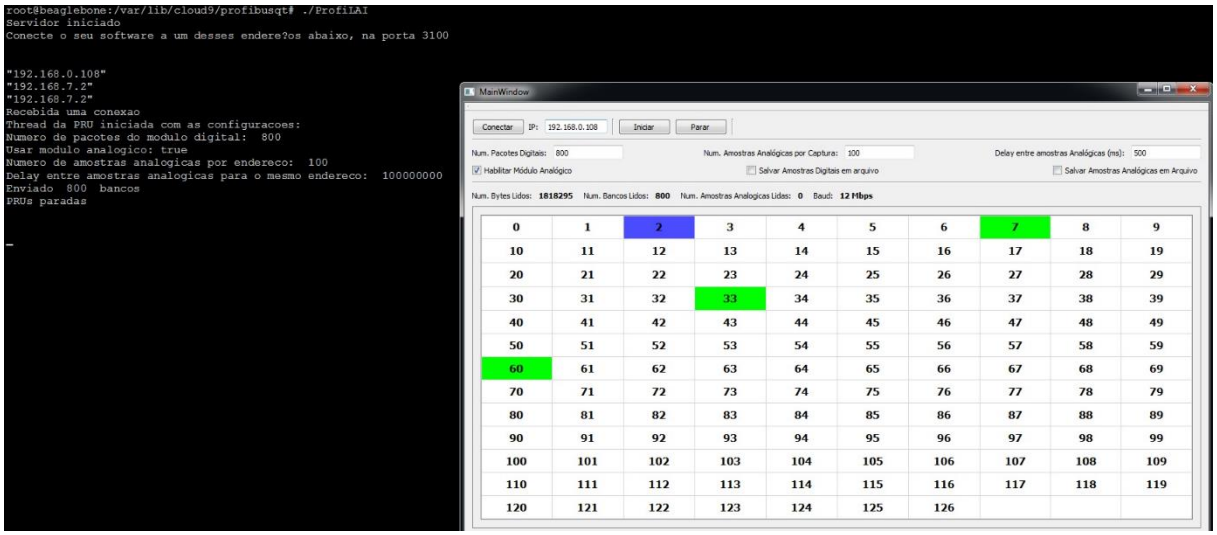


Figura 6-4 - Captura de 800 pacotes digitais na rede Profibus operando no Baud Rate de 12Mbps.

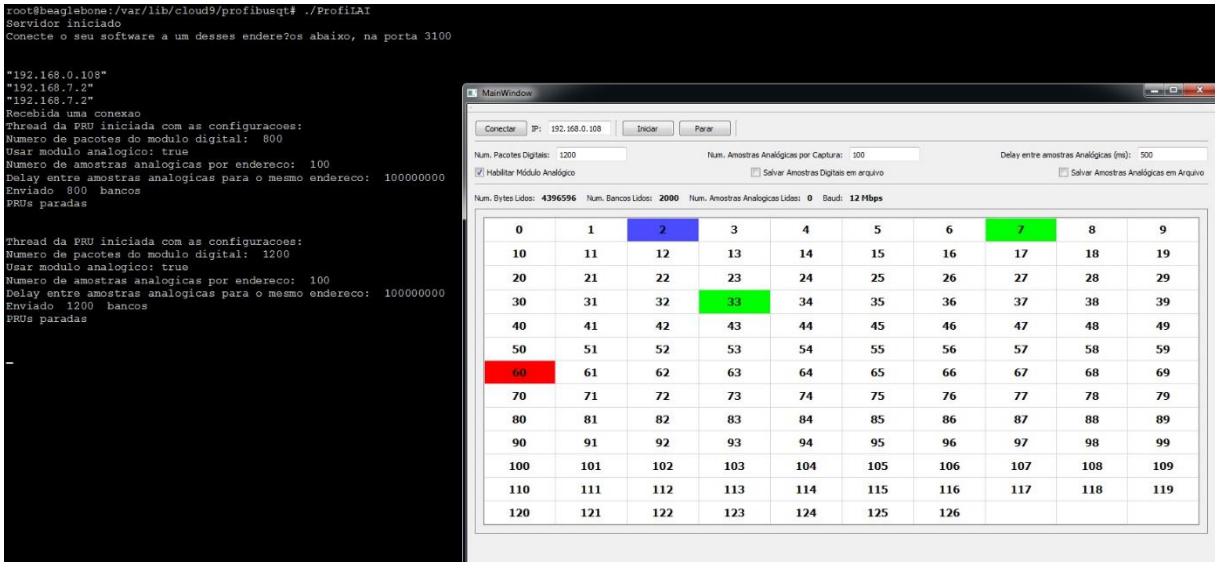


Figura 6-5 - Captura de mais 1200 pacotes digitais depois do termino da captura anterior. O equipamento do endereço 60 foi desligado, indicando perda de comunicação.

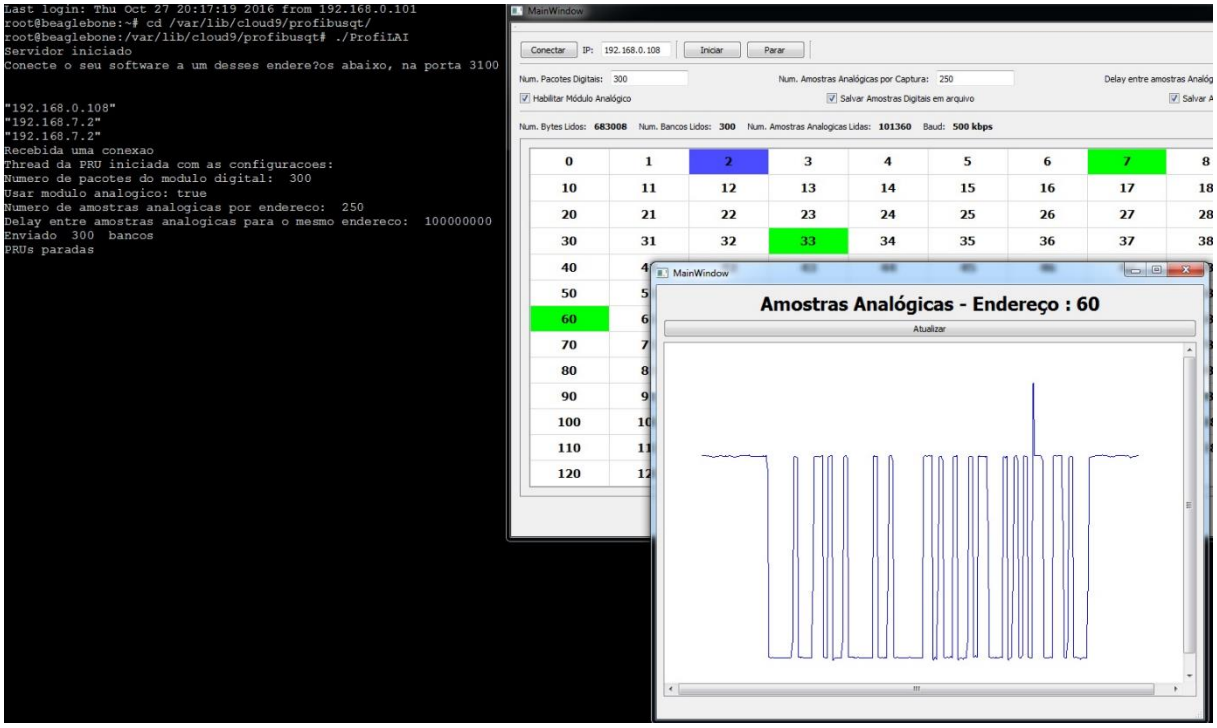


Figura 6-6 - Captura de 300 pacotes digitais em uma rede Profibus operando no Baud Rate de 500kbps. 101360 amostras analógicas foram capturadas, sendo que o último pacote de 250 amostras do endereço 60 estão exibidas na tela de osciloscópio.

6.2.2 Arquivos gerados

Neste software de usuário desenvolvido, o usuário tem a opção de salvar os dados digitais e analógicos recebidos em arquivos separados. Esses arquivos possuem os campos dos pacotes de dados digitais e pacotes dos dados analógicos descritos juntamente com os valores recebidos, para que o usuário possa ver seu funcionamento caso tenha alguma dúvida. Dentro desse arquivo é possível também coletar as informações enviadas pelos telegramas e também os valores das amostras analógicas lidas da rede, conforme visto na Figura 6-7 e Figura 6-8.

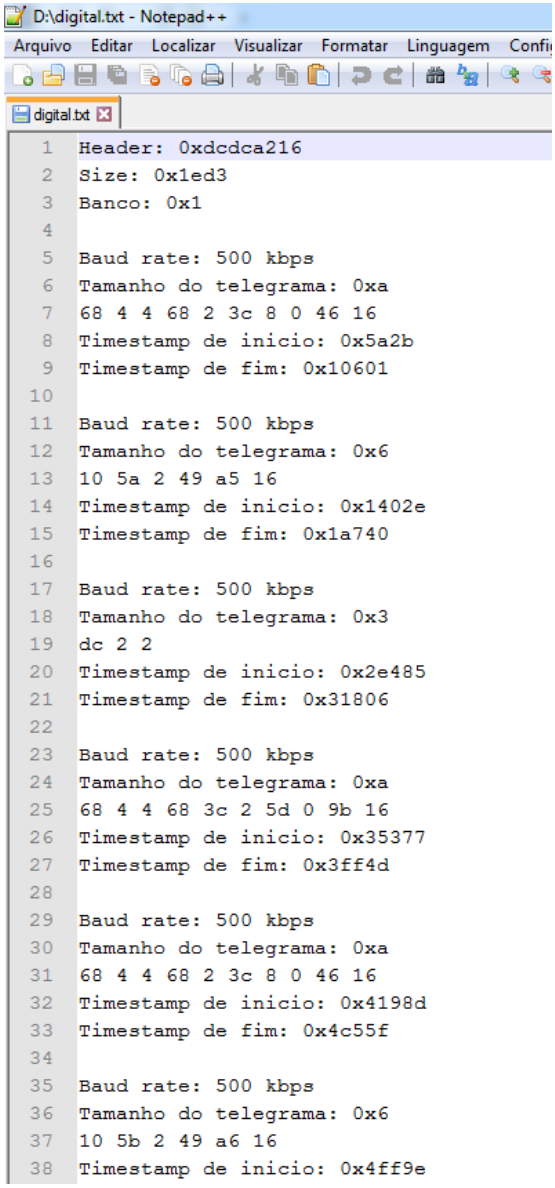


Figura 6-7 - Arquivo dos pacotes de dados digitais.

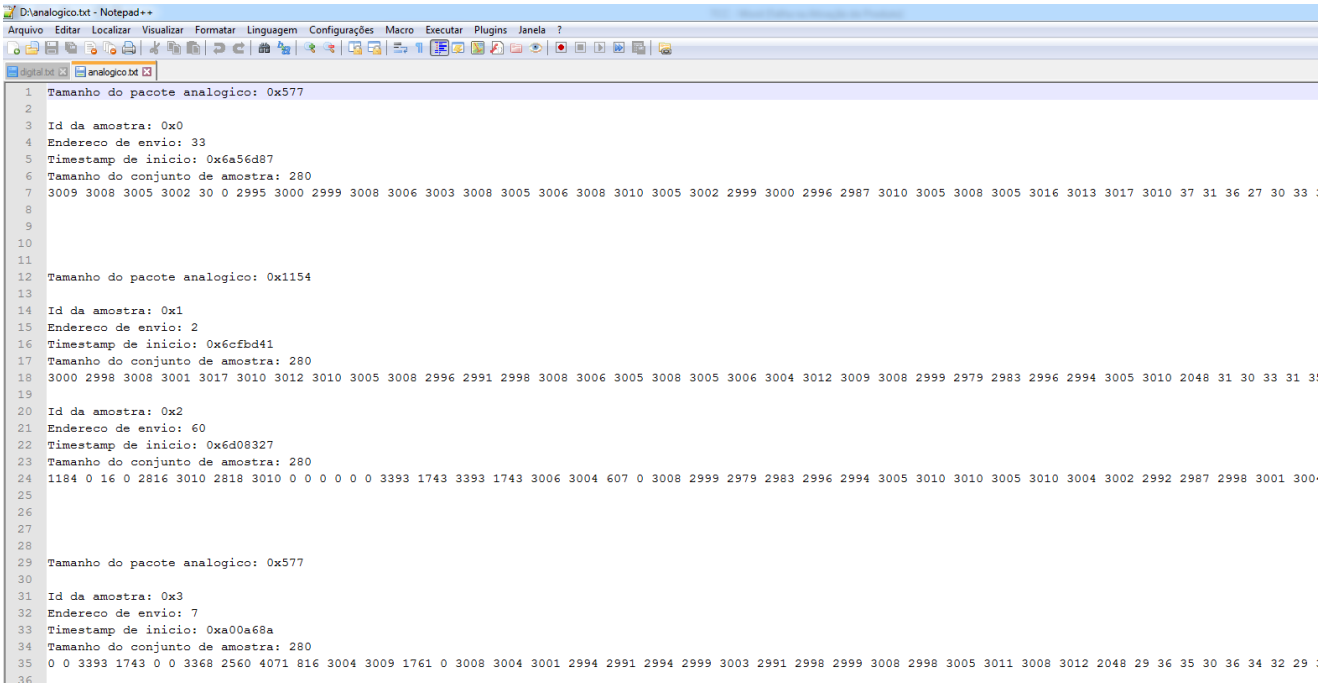


Figura 6-8 - Arquivo dos pacotes de dados analógicos.

6.3 Resultados obtidos variando a configuração da rede

Vários testes foram feitos com a rede em operação em diversas situações para garantir o funcionamento contínuo da ferramenta de forma adequada. As situações testadas e os resultados da operação da ferramenta em cada uma foram:

1) Rede em operação normal: a ferramenta ProfiLAI executa o envio de todos os pacotes digitais requeridos pela configuração inicial da captura. O teste mais extenso realizado foi de 3 horas e 22 minutos de duração, onde foram transmitidos um milhão de pacotes digitais, lidos da rede Profibus operando no Baud Rate de 12Mbps. Foram lidos 1.7GB de dados dos telegramas da rede e transmitidos para o software do usuário desenvolvido para esse teste. Interrupções nos escravos e no mestre eram esporadicamente causadas, retirando a alimentação desses dispositivos e depois alimentando-os novamente.
2) Troca de Baud Rate da rede durante a operação: assim que a ferramenta detecta três erros seguidos de transmissão, causados pela troca do Baud Rate durante a operação, a captura é reiniciada automaticamente pela ferramenta ProfiLAI, sem causar a perda de conexão com o software do usuário. Depois do reinicio da

captura, o novo Baud Rate é detectado corretamente e a captura continua sem erros.

3) Início da captura sem mestres na rede: a ferramenta ProfiLAI não envia nenhum pacote até que seja coletado alguma informação da rede Profibus. Portanto, enquanto nenhum mestre é conectado na rede, a comunicação entre a ferramenta e o software do usuário fica aberta, porém sem nenhuma transmissão de dados.
4) Interrupção de equipamento (mestre ou escravo) durante a rede: conforme descrito no primeiro item, essa interrupção é considerada como uma operação normal. Mesmo que um dispositivo fique *offline*, a transmissão para o software do usuário continua se ainda houver dados sendo transmitidos na rede Profibus. Se todos os mestres forem interrompidos, a transmissão dos dados continuará quando a transmissão da rede Profibus retornar.
5) Interrupção da conexão TCP/IP: se por algum motivo a conexão TCP/IP for interrompida entre a ferramenta ProfiLAI e o software do usuário, a ferramenta para a captura se alguma estiver sendo realizada e entra no modo de espera de conexão.
6) Conexão de outro software do usuário: se uma nova conexão for requisitada enquanto uma já estiver aberta, a ferramenta ProfiLAI irá parar a captura caso alguma estiver sendo realizada e interromperá a conexão antiga, aguardado a configuração inicial de captura da nova conexão.

6.3.1 Tabela de Resultados

Todos os Baud Rates possíveis da Profibus foram testados diversas vezes para garantir que nenhum erro de captura iria acontecer. Alguns dos resultados foram separados e exibidos na Tabela 6-1 e Tabela 6-2.

Baud Rate da rede:	9,6kbps	19,2kbps	500kbps	1,5Mbps
Baud Rate detectado	9,6kbps	19,2kbps	500kbps	1,5Mbps
Nº de pacotes digitais requisitados	200	500	1500	5000
Nº de amostras analógicas por captura	500	500	300	-
Delay entre amostras analógicas de mesmo endereço (ms)	500	200	1000	-
Nº de pacotes digitais lidos	200	500	1500	5000
Bytes lidos	20704	80616	3415178	11372334
Nº de amostras analógicas lidas	86380	381070	397980	-
Throughput médio da rede Ethernet	54,8 kbps	121,2kbps	709kbps	4,1Mbps
Tempo	43,6 s	1m 23s	2m 4s	2m 15s

Tabela 6-1 - Resultados dos testes aplicados na ferramenta.

Baud Rate da rede:	6Mbps	12 Mbps	12 Mbps
Baud Rate detectado	6Mbps	12 Mbps	12 Mbps
Nº de pacotes digitais requisitados	5000	25000	1000000
Nº de amostras analógicas por captura	-	-	-
Delay entre amostras analógicas de mesmo endereço (ms)	-	-	-
Nº de pacotes digitais lidos	5000	25000	1000000
Bytes lidos	11372334	56829155	1,73GB
Nº de amostras analógicas lidas	-	-	-
Throughput médio da rede Ethernet	4,1 Mbps	5,3 Mbps	5,3 Mbps
Tempo	1m 19s	5m 2s	3h 22m

Tabela 6-2 - Segunda parte dos resultados dos testes aplicados na ferramenta.

7 CONCLUSÃO

Buscou-se nesse trabalho utilizar o máximo do hardware escolhido para alcançar os objetivos propostos. Atualmente, o preço de uma BeagleBone Black Rev. C no Brasil está em torno de R$400,00. Apenas utilizando essa placa e a ferramenta desenvolvida é possível coletar as informações da rede Profibus, funcionalidade essa feita por equipamentos comerciais com preços que chegam a algumas dezenas de milhares de reais, cumprindo assim o objetivo do desenvolvimento de uma ferramenta de baixo custo.

Para a realização desse trabalho, o aprendizado obtido em disciplinas da graduação que abordavam o assunto de Assembly, como Introdução à Organização de Computadores e Aplicações de Microprocessadores I e II foram amplamente utilizados, acrescentando agora o aprendizado profundo da arquitetura do processador AM335x e o sistema da BeagleBone. As disciplinas de Automação e Redes Industriais também foram fundamentais para o desenvolvimento deste trabalho, uma vez que a finalidade da rede trabalhada é para o ramo industrial, sendo necessário o conhecimento de todo o funcionamento desta rede e seus equipamentos.

A ferramenta desenvolvida será utilizada para novas pesquisas na área de Profibus pelos alunos do Laboratório de Automação Industrial, principalmente no processamento dos dados em tempo real, que antes não era possível devido a impossibilidade dessa coleta dos dados e transmissão para o software desenvolvido pelo pesquisador, devido à necessidade dos drivers e softwares autorais dos analisadores de rede existentes.

Os resultados coletados com a ferramenta ProfiLAI tiveram uma confiabilidade de 100% nos testes aplicados, até me

smo com a execução da captura por mais de três horas seguidas sem perder nenhuma informação transmitida pela rede.

REFERÊNCIAS

1. INTERNATIONAL, P. Overview. **Profibus International**, 2016. Disponivel em: <http://www.profibus.com/technology/profibus/overview/>. Acesso em: 27 set. 2016.

2. MOSSIN, E. A. **Diagnóstico Automático de redes Profibus**. Tese de Doutorado. Universidade de São Paulo. São Carlos. 2012.

3. BRANDÃO, D.; SESTITO, G. S. **Instalador Certificado Profibus, v1.7**. [S.l.]: [s.n.], 2016.

4. GUIDELINE, P. **Installation Guideline for Profibus-DP/FMS**. [S.l.]: [s.n.], Version 1.0, 1998.

5. HANELT, C. Online-Optimierung des Übertragungsverhaltens eines SPS-Verbundes unter PROFIBUS mit Hilfe von Fuzzy Logik. Disponivel em: <http://hanelt.de/dipl_ver.htm>. Acesso em: 28 set. 2016.

6. TORRES, R. V. **Simulador de redes Profibus**. Dissertação de Mestrado. Universidade de São Paulo. São Carlos. 2013.

7. Introduction to Profibus DP. Technical Reference - Acromag. [S.l.]. 2002.

8. FELSER, M. **PROFIBUS Handbuch**, 2013. Disponivel em: <http://profibus.felser.ch/en/index.html?service_access_point.htm>. Acesso em: 28 set. 2016.

9. PROFIBUS International. **ProfiEnergy**. Disponivel em: <http://www.profibus.com/technology/profienergy/>. Acesso em: 29 set. 2016.

10. PROFIBUS International. **ProfiSafe**. Disponivel em: <http://www.profibus.com/technology/profisafe/>. Acesso em: 29 set. 2016.

11. PROFIBUS International. **ProfiDrive**. Disponivel em: <http://www.profibus.com/technology/profidrive/>. Acesso em: 29 set. 2016.

12. SMAR. **EMI - Interferência Eletromagnética em instalações industriais e muito mais**. Disponivel em: <http://www.smar.com/brasil/artigo-tecnico/emi-interferencia-eletromagnetica-em-instalacoes-industriais-e-muito-mais>. Acesso em: 29 set. 2016.

13. **Toledo & Souza**. Disponivel em: <http://www.toledoesouza.com/>. Acesso em: 29 set. 2016.

14. PROFIBUS Troubleshooting Kit. Disponivel em: <http://www.idxonline.com/Default.aspx?tabid=365>. Acesso em: 25 out. 2016.

15. **About BeagleBoard**. Disponivel em: <http://beagleboard.org/about>. Acesso em: 29 set. 2016.

16. MONK, S. **Programming the BeagleBone Black:** Getting Started with JavaScript and BoneScript. 1ª. ed. [S.l.]: McGraw-Hill, 2014.

17. **Beagleboard:** BeagleBoneBlack. Disponivel em: <http://elinux.org/Beagleboard:BeagleBoneBlack>. Acesso em: 29 set. 2016.

18. **BeagleBone Operating Systems**. Disponivel em: <http://elinux.org/BeagleBone_Operating_Systems>. Acesso em: 29 set. 2016.

19. **GPIO Pinout**. Disponivel em: <https://github.com/uraimo/SwiftyGPIO/wiki/GPIO-Pinout>. Acesso em: 29 set. 2016.

20. MOLLOY, D. **Exploring BeagleBone:** Tools and Techniques for Building with Embedded Linux. [S.l.]: Willey, 2014.

21. **AM335x Overview - Texas Instruments**. Disponivel em: <http://www.ti.com/lsds/ti/processors/sitara/arm_cortex-a8/am335x/overview.page>. Acesso em: 30 set. 2016.

22. INSTRUMENTS, T. **AM335x Sitara Processors - Technical Reference Manual**. [S.l.]. 2011.

23. INTEGRATED, M. **Datasheet MAX481/MAX483/MAX485/MAX487/MAX491/MAX1487**. [S.l.]. 2014.

24. INSTRUMENTS, T. **ISO1176 Isolated RS-485 Profibus Transceiver Datasheet**. [S.l.]. 2015.