Projeto e Análise de um Sistema de Aquisição de Dados Distribuído para Aplicações em Tempo Real

Jadsonlee da Silva Sá

Dissertação de Mestrado submetida à Coordenadoria do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande - Campus de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Instrumentação e Controle

Orientadores
José Sérgio da Rocha Neto, D.Sc.
Antonio Marcus Nogueira Lima, Dr.

Campina Grande, Paraíba, Brasil

Projeto e Análise de um Sistema de Aquisição de Dados Distribuído para Aplicações em Tempo Real

Jadsonlee da Silva Sá

Dissertação de Mestrado apresentada em Março de 2007

Orientadores
José Sérgio da Rocha Neto, D.Sc.
Antonio Marcus Nogueira Lima, Dr.

Banca Examinadora
Alexandre Cunha Oliveira, D.Sc.
Eurico Bezerra Filho, D.Sc.

Campina Grande, Paraíba, Brasil, Março de 2007

Dedicatória

Dedico este trabalho aos meus pais, Adaildo Jeremias de Sá e Maria Auxiliadora da Silva Sá, aos meus irmãos, Jancylee da Silva Sá e Jemylee da Silva Sá, e a todos que contribuiram de alguma forma para a realização deste trabalho

Agradecimentos

- Agradeço a Deus pela minha vida.
- Aos meus pais, Adaildo Jeremias Sá e Maria Auxiliadora da Silva Sá, pelo apoio incondicional.
- Aos meus irmãos, Jancylee e Jemylee, pelo incentivo e pela força.
- Aos professores José Sérgio e Antonio Marcos pela orientação e ajuda, sem a qual este trabalho não seria realizado.
- Ao técnico Simões Toledo, pela grande ajuda na confecção das placas utilizadas neste trabalho.
- Aos amigos e amigas: Manabu Oguro, Simões Toledo, Cláudia Delgado, Tiago Marinho, Tomás Victor, Jaidilson Jó, Alfranque Amaral, José Luis, Danilo, Darlan, Pedrosa, Lucas, Luciano Lisboa, Milena Albuquerque, Airam e Paulo Sausen, Valter Vermherem, Wálber Medeiros, Vinícius Cavalcanti, Luiz Felipe, Renan, Dimas Delgado, Camila Caetano, Loly Liu, Genildo de Moura, Saulo, Olímpio, Maria Ester, Diana Keyla, Martapoliana, Tércia Batista, Hadassa Rodrigues, Guilherme Lócio, Fábio Marques, Wullo Mariute, Virlandro Nobre, Fernando Menezes e João Paulo.
- Aos funcionários da COPELE e dos laboratórios LIEC, LEMCAD e LEIAM.
- Ao CNPq, pela bolsa de pesquisa.
- Aos projetos GEBRA/BRASYMPE e iSensor, pelo apoio financeiro oferecido a este projeto.

Resumo

Nesta dissertação descreve-se o projeto e análise de um sistema de aquisição de dados distribuído, que foi desenvolvido para adquirir em tempo real as informações convertidas por vários sensores distribuídos no Laboratório de Geração Termoelétrica, o qual está instalado nas dependências do Departamento de Engenharia Elétrica na Universidade Federal de Campina Grande. O principal componente deste Laboratório é um sistema eletro-mecânico constituído por um motor do ciclo diesel de 250 HP acoplado a um gerador elétrico com capacidade de 150 kW.

O sistema de aquisição de dados é constituído por oito nós sensores e um nó supervisor. Todos os nós sensores foram projetados e desenvolvidos de acordo com os requisitos da aplicação. Cada nó sensor é responsável pela aquisição e transmissão das informações coletadas pelos seus respectivos sensores para o nó supervisor, conforme os requisitos de tempo exigidos pelo sistema. Os nós comunicam-se por meio de um barramento com 25 m de comprimento de acordo com o protocolo de comunicações CAN (*Controller Area Network*) a uma taxa de transmissão de 1 Mbps. Uma versão *Time-Triggered* do protocolo CAN foi implementada por *software* no mesmo *chip* CAN, utilizando um algoritmo de sincronização baseado na especificação TTCAN (*Time-Triggered* CAN) nível 1.

Abstract

In this dissertation is presented the project and analysis of a Distributed Acquisition Data System that it was development to acquire in real time the information converted for some distributed sensors into the Laboratory of Thermoelectrical Generation which is installed in the Department of Electric Engineering in the Federal University of Campina Grande. The main component of this Laboratory is an electromechanical system consisting by an motor of the cycle diesel of 250 HP connected to an electric generator to capacity of 150 kW.

The acquisition data system is composed by eight sensor nodes and one supervisor node. All sensor nodes have been projected and development in accordance with the requirement of the application. Each in sensor node is responsible for the acquisition and transmission of the information collected for its respective sensors to the supervisor node, as the requirements of time demanded by the system. These nodes communicate via a bus with 25 meters in length with a transmission rate of 1 Mbps in accordance with the CAN (Controller Area Network) protocol. A version time-triggered of the CAN protocol was implemented by software in the CAN chip, having used an algorithm of synchronization based on specification TTCAN (Time-Triggered CAN) level 1.

Índice

Glossário

μC	Microcontrolador
τ_{bit}	Tempo de *Bit*
L_{max}	Máximo comprimento do barramento
q_{CeN0}	Quantidade de ciclos do relógio 0 do escravo do tempo N
q_{CeN1}	Quantidade de ciclos do relógio 1 do escravo do tempo N
q_{Cm0}	Quantidade de ciclos do relógio 0 do mestre do tempo
q_{Cm1}	Quantidade de ciclos do relógio 1 do mestre do tempo
t_p^*	Constante de propagação específica do barramento
t_{CAN}	Tempo de atraso do *chip* CAN
t_{CeN}	Tempo do ciclo do nó escravo do tempo N
t_{Cm}	Tempo do ciclo do nó mestre do tempo
t_d	Tempo máximo de propagação
t_{el}	Tempo de atraso do sinal elétrico
t_{Fase_1}	Tempo do Segmento de Fase 1
t_{Fase_2}	Tempo do Segmento de Fase 2
t_L	Tempo gasto para o sinal elétrico chegar ao *transceiver* do nó mais distante
t_N	Período do relógio do nó N derivado pelo respectivo cristal
t_{Prop}	Tempo do Segmento de Propagação
t_{ReN}	Período de ciclo dos relógios do escravo N
t_{Rm}	Período de ciclo do relógio do mestre
t_{Sinc}	Tempo do Segmento de Sincronização
t_{TeN}	*Tx-Trigger* do nó escravo do tempo N
t_{Tm}	*Tx-Trigger* do nó mestre do tempo
$t_{transceiver}$	Tempo de atraso do *transceiver*

ACK	*Acknowledgment*
API CANLIB	*Application Programmer Interface CAN LIBrary*
ASI	*Actuator Sensor Interface*
b	Número de *bytes* do campo de dados
BCH	*Bose Chaudhuri-Hocquenghem*
C	Tempo gasto para transmitir uma mensagem de dados
CAL	*CAN Application Layer*
CAN	*Controller Area Network*
CRC	*Cyclic Redundancy Check*
CSMA	*Carrier-Sense Multiple Access*
CSMA/CA	*Carrier Sense Multiple Access with Collision Avoidance*
CSMA/CD	*Carrier-Sense Multiple Access/Collision Detect*
DEE	Departamento de Engenharia Elétrica
df	Fator de ajuste do TUR
DLC	*Data Lenght Control*
DSP	Processador Digital de Sinais
e	Erro de Fase
EOF	*End of Frame*
FIFO	*First In First Out*
FTTCAN	*Flexible Time-Triggered Controller Area Network*
GNV	Gás Natural Veicular
ISO	*International Standardization Organization*
LPS	Largura do Pulso de Sincronização
MAC	*Medium Access Control*
MR	Marco de Referêncial
MRM	Marco de Referência do Mestre
MS	Marco de Sincronismo
NRZ	*Non-Return to Zero*
NTU	Unidade de Tempo da Rede
OL	*Offset Local*

OSI	*Open Systems Interconnections*
PCI	*Peripheral Component Interconnect*
PROFIBUS	*PROcess FIeld BUS*
REC	Contador de Erro de Recepção
RTR	*Remote Transmit Request*
SADC	Sistema de Aquisição de Dados Centralizado
SADDs	Sistemas de Aquisição de Dados Distribuído
SADs	Sistemas de Aquisição de Dados
SJW	Syncronization Jump Width
SOF	*Start of Frame*
SPI	*Serial Peripheral Interface*
TEC	Contador de Erro de Transmissão
TPI	Tempo de Processamento da Informação
TQ	*Time Quanta*
TTP	*Time-Triggered Protocol*
TUR	Razão da Unidade de Tempo Local
UFCG	Universidade Federal de Campina Grande
USB	*Universal Serial Bus*

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Sistemas de Aquisição de Dados (SADs) são utilizados para adquirir as informações das grandezas físicas provenientes de processos do mundo real por meio de sensores. As informações adquiridas por um SAD são analisadas para determinar o comportamento do processo em estudo. Exemplos de SADs são encontrados em diversas aplicações, tais como: monitoramento de máquinas (SILVA et al., 2005) (PANHAN, 2002); análise de sinais de tensão e corrente da rede elétrica (NETO; GIACOMIN, 2004); determinação de parâmetros mecânicos dos solos (LIBONATI; OLIVEIRA; CAMPOS, 2003) (FONSECA; XAVIER; CARDOSO, 2003); aplicações automotivas e aeroespaciais (JOHANSSON; TöRNGREN; NIELSEN, 2005); na robótica (LIMA et al., 2004) entre outros.

Durante muitos anos, SADs com arquiteturas centralizadas predominaram nas diversas aplicações. Nestes sistemas todos os sensores são conectados por meio de ligações ponto a ponto a uma única unidade de processamento, onde as informações são coletadas e transmitidas para um computador. Devido aos avanços tecnológicos, as aplicações tornaram-se mais complexas, necessitando de mais sensores distribuídos no processo. Como conseqüência, houve um aumento na quantidade dos cabos conectando os sensores à unidade de processamento. Problemas de confiabilidade, expansibilidade e aumento do custo de instalação e manutenção começaram a surgir. Uma solução para resolver esses problemas foi a utilização de sistemas com arquiteturas de processamento distribuído baseadas em redes de comunicações. A evolução dos dispositivos semicondutores viabilizou a disponibilidade de *hardware* de baixo custo e das tecnologias de redes de comunicações/barramentos de campo[1], as quais possibilitaram a implementação de SADs com arquiteturas de processamento distribuído, ou simplesmente, Sistemas de Aquisição de Dados Distribuídos (SADDs).

A implementação de um SADD envolve a especificação de vários componentes de *hard-*

[1]Barramento serial digital que possibilita a comunicação em tempo real entre dispositivos de campo tais como, sensores, atuadores, controladores, reguladores e etc (ALMEIDA, 1999).

ware e *software*. Estes componentes devem interagir de forma que os requisitos funcionais e temporais de operação da aplicação sejam satisfeitos. Os requisitos funcionais estão relacionados com o funcionamento correto dos componentes do sistema. Já os requisitos temporais referem-se aos instantes em que determinadas tarefas devem ser iniciadas e finalizadas.

Uma arquitetura típica de um SADD é constituída por vários subsistemas chamados de nós ou módulos. Os nós são distribuídos no processo de acordo com a localização dos sensores, e comunicam-se através de um meio compartilhado, normalmente na topologia barramento, conforme um protocolo de comunicação em rede. Basicamente, existem dois tipos de nó: nó sensor e o nó supervisor. O nó sensor é constituído pelos seguintes componentes: sensores; circuitos de condicionamento; unidade de processamento (μC ou DSP); dispositivos de rede; e um *firmware*[2] para gerenciamento e controle do nó. O nó supervisor é constituído por um ou mais dispositivos de rede dedicados com alguma *interface* para comunicação com um *software* de aplicação no computador. Na Figura 1.1 representa-se um diagrama de uma arquitetura típica de um SADD e seus componentes de *hardware* e *software*.

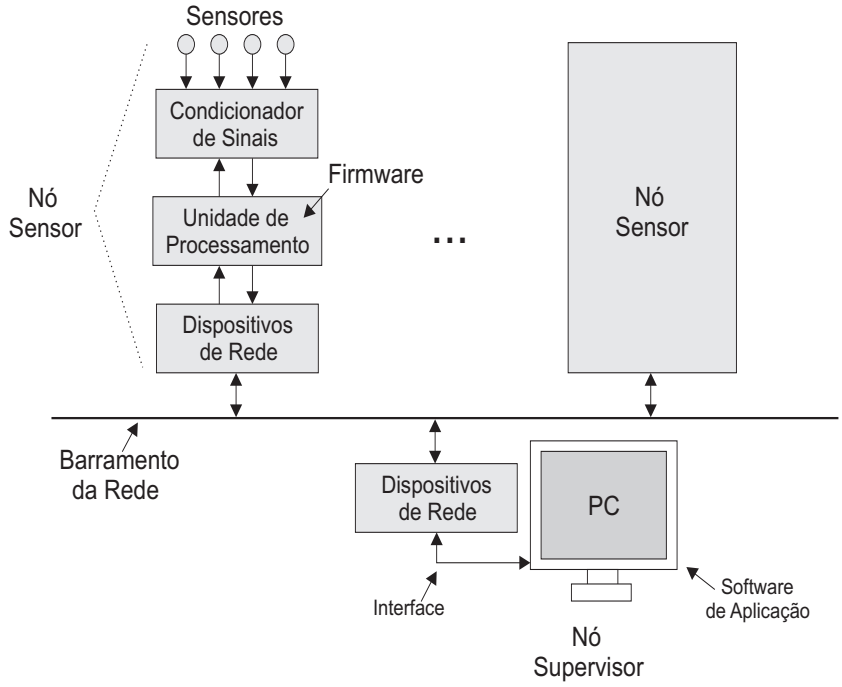

Figura 1.1: Diagrama de uma arquitetura típica de um SADD.

Conforme representado na Figura 1.1, os sensores captam as grandezas físicas do processo em estudo e as convertem em sinais elétricos analógicos (tensão ou corrente elétrica). Estes sinais são adequados por circuitos condicionadores (filtros, amplificadores,

[2] *Software* que gerencia e controla diretamente um *hardware*

isoladores e outros) para serem convertidos em sinais digitais pelo conversor analógico digital embarcado na unidade de processamento (μC ou DSP), a uma determinada taxa de amostragem. Estas informações são enviadas para o dispositivo de rede, que as transmitem em forma de mensagem pelo barramento da rede até o dispositivo de rede do nó supervisor. Em seguida, essas informações são enviadas via uma *interface* para um computador, onde são processadas, armazenadas e analisadas pelo *software* de aplicação.

Para determinar corretamente o comportamento do processo em estudo, todas as informações das grandezas físicas devem ser adquiridas pelos nós sensores e transmitidas corretamente para o nó supervisor, conforme os requisitos de tempo real do sistema. Para satisfazer esses requisitos, cada um dos nós executam uma ou mais tarefas periódicas utilizando seus recursos de hardware. Essas tarefas são gerenciadas e controladas pelo *firmware* armazenado na memória da unidade de processamento (Nó sensor) e pelo *software* de aplicação do computador (Nó supervisor).

Basicamente, um nó sensor executa uma tarefa periódica que realiza em seqüência as seguintes atividades: conversão analógico digital dos sinais provenientes dos sensores; envio dessas informações para o *buffer* de transmissão do dispositivo de rede; requisição de transmissão da mensagem; e transmissão da mensagem contendo essas informações até o nó supervisor.

O nó supervisor executa uma tarefa que faz a retirada das mensagens do *buffer* de recepção do dispositivo de rede, e executa atividades específicas com as informações dessas mensagens, tais como: processamento, armazenamento em arquivos de texto e visualização por meio de *interface* gráfica. As mensagens são armazenadas e removidas da fila do *buffer* de recepção de acordo com o protocolo FIFO (*First In First Out*) pelo *software* de aplicação. Este *buffer* possui comprimento limitado, o qual impõe restrições de tempo ao *software* de aplicação. Deve-se garantir que não haverá estouro de capacidade do *buffer*, e consequentemente, todas as mensagens serão recebidas. Caso contrário, as mensagens que chegarem durante a sobrecarga do *buffer* serão perdidas. Devido o *software* de aplicação e o sistema operacional compartilharem os mesmos recursos de *hardware*, o comportamento temporal do sistema operacional afeta diretamente o comportamento temporal do *software* de aplicação. Deste modo, a utilização de um sistema operacional capaz de satisfazer esses requisitos de tempo são cruciais para o desempenho da aplicação.

Em um SADD, os nós comunicam-se via um meio de comunicação compartilhado. Para solucionar possíveis colisões entre as mensagens enviadas pelos nós, o protocolo de comunicação implementa um mecanismo para controlar o acesso ao meio. Basicamente, esses mecanismos podem ser classificados em aleatórios (não-determinísticos) e determinísticos.

Os mecanismos aleatórios seguem o paradigma de comunicação *Event-Triggered* (ALBERT, 2004). Neste tipo de comunicação, as mensagens são escalonadas dinamicamente

no barramento durante a execução do sistema. Algum método de arbitragem é utilizado para solucionar possíveis colisões entre mensagens. A vantagem deste tipo de comunicação é a capacidade de reagir à eventos externos aleatórios tais como, erros em mensagens provocados por interferências eletromagnéticas, visto que estas mensagens serão re-transmitidas. Por outro lado, esses sistemas podem apresentar uma alta variabilidade nos atrasos na transmissão das mensagens. Para garantir que as mensagens chegarão ao seu destino respeitando os requisitos de tempo, deve-se realizar uma análise dos tempos de resposta de cada uma das mensagens, considerando o tráfego e as condições de erros no barramento. CAN (*Controller Area Network*) (BOSCH, 1991) é um exemplo de protocolo *Event-Triggered* baseado no mecanismo CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*).

Os mecanismos determinísticos seguem o paradigma de comunicação *Time-Triggered* (KOPETZ; BAUER, 2003) (ALBERT, 2004). Neste tipo de comunicação, as mensagens são escalonadas estaticamente de acordo com a progressão de um tempo sincronizado globalmente entre os diversos nós da rede. Janelas de tempo são atribuídas a cada uma das mensagens, semelhante ao mecanismo TDMA (*Time Division Multiple Access*). Portanto, colisões entre mensagens são evitadas. Sistemas baseados neste paradigma de comunicação possuem um comportamento estritamente determinístico no tempo. Como desvantagem, estes sistemas são incapazes de reagir à eventos externos aleatórios. Exemplos de redes que seguem este conceito de comunicação são TTP (*Time-Triggered Protocol*) (TTTECH, 1999) e TTCAN (*Time-Triggered Controller Area Network*) (FüHRER et al., 2000) (LEEN; HEFFERNAN., 2002).

A especificação dos componentes de um SADD depende exclusivamente do processo em estudo. Questões como, qual o ambiente em que o processo está localizado, quais tipos de grandezas físicas serão monitoradas e quais os requisitos de tempo real do sistema são fundamentais durante a fase de projeto. Com essas informações, pode-se decidir quais sensores serão utilizados, os circuitos necessários para um adequado condicionamento dos sinais, a taxa de amostragem dos sinais, a resolução do conversor analógico digital e os requisitos para a escolha da rede de comunicação, tais como: taxa de transmissão; controle de acesso ao meio (tipo de comunicação); meio de transmissão; número máximo de nós; comprimento da mensagem; tempo de latência da mensagem entre outros.

A rede de comunicação é o componente principal em qualquer sistema distribuído. Apesar dos requisitos da aplicação serem fundamentais na escolha da tecnologia de rede, questões como custo de instalação, manutenção e disponibilidade de *hardware* são cruciais no processo de escolha do sistema. Atualmente, existem no mercado diversas tecnologias de rede. As redes mais utilizadas em aplicações de monitoração e controle são: ASI (*Actuator Sensor Interface*) (KRIESEL; MADELUNG, 1999), CAN (ETSCHBERGER, 2001)

(LAWRENZ, 1997), PROFIBUS (*PROcess FIeld BUS*) (PROFIBUS, 2000), Interbus-S (INTERBUS, 2007), WorldFip (WORLDFIP, 2007) e DeviceNet (DEVICENET, 2007). Uma das redes que tem se destacado durante as duas últimas décadas é a rede/protocolo de comunicações CAN[3].

CAN é uma rede de comunicações utilizada em aplicações onde se deseja monitorar e controlar sistemas distribuídos em tempo real com um alto nível de confiabilidade das informações. Embora seja uma rede com controle de acesso ao meio não-determinístico, é possível implementar uma versão determinística, chamada de TTCAN(*Time-Triggered* CAN) (FüHRER et al., 2000) (HARTWICH et al., 2000).

Devido as suas características, tais como: taxa de transmissão de 1 Mbps; protocolo orientado a mensagens; curto comprimento da mensagem; eficientes mecanimos para detecção e sinalização de erros; detecção e desativação de nós defeituosos, a disponibilidade de *chips* de baixo custo, a grande aceitação no mercado e a possibilidade de implementar dois conceitos de comunicações, escolhemos a rede CAN para ser utilizada neste trabalho.

1.1 Objetivos

Neste trabalho apresenta-se a implementação de um sistema de aquisição de dados distribuído em tempo real, que foi projetado para ser utilizado no Laboratório de Geração Termoelétrica instalado nas dependências do Departamento de Engenharia Elétrica (DEE) na Universidade Federal de Campina Grande (UFCG). Este sistema tem como objetivo resolver alguns dos problemas apresentados no atual sistema de aquisição de dados centralizado instalado no laboratório, tais como: redução da grande quantidade de cabos; capacidade de analisar grandezas físicas com altos requisitos de tempo real; e maior capacidade de expansão do sistema.

A implementação do sistema de aquisição de dados foi baseado na rede de comunicações CAN. Uma versão com acesso ao meio determinístico baseado no TTCAN nível 1 foi implementado por *software* utilizando um algoritmo de sincronização e algumas funções em *hardware*, no mesmo *chip* CAN. Deste modo, o sistema será capaz de funcionar em um dos dois tipos de comunicações.

O sistema de aquisição de dados deverá ser capaz de adquirir as informações de vários sensores distribuídos no laboratório, respeitando as restrições de tempo. Todos os nós sensores, os *firmware* e o *software* de aplicação foram especificados, projetados e desenvolvidos no DEE pelo autor deste trabalho.

[3]Os termos rede CAN e protocolo CAN são utilizados pela maioria dos autores de forma intercambiável.

1.2 Sinopse dos Capítulos

Esta dissertação é composta por mais sete capítulos, os quais são descritos a seguir:

- No Capítulo 2 descreve-se brevemente o Laboratório de Geração Termoelétrica e o sistema de aquisição de dados centralizado. Os principais problemas deste sistema serão apresentados. Um sistema com arquitetura distribuída baseado na rede CAN/TTCAN será proposto. Finalmente, alguns conceitos e definições básicas envolvendo redes de comunicações serão apresentados.

- As principais características, funcionamento e formas de implementação da rede de comunicações CAN são apresentadas no Capítulo 3.

- No Capítulo 4 apresenta-se uma breve descrição do funcionamento do protocolo TTCAN (nível 1 e nível 2). Uma estratégia de implementação por *software* deste protocolo será apresentada.

- O projeto do sistema de aquisição de dados distribuído para o laboratório de geração termoelétrica é apresentado no Capítulo 5.

- No Capítulo 6 são apresentados os resultados obtidos neste trabalho.

- As conclusões e sugestões para trabalhos futuros são apresentadas no capítulo 7.

- No Capítulo Anexos são apresentados os diagramas dos circuitos elétricos das placas e dos circuitos eletrônicos que foram desenvolvidos para a implementação do sistema de aquisição de dados distribuído.

Capítulo 2

Laboratório de Geração Termoelétrica

2.1 Introdução

No Laboratório de Geração Termoelétrica, instalado nas dependências do Departamento de Engenharia Elétrica (DEE) na Universidade Federal de Campina Grande (UFCG), são desenvolvidas pesquisas com o intuito de otimizar e controlar o uso de Gás Natural Veicular (GNV) em motores do ciclo diesel para geração de eletricidade. O principal componente do laboratório é um sistema eletro-mecânico constituído por um motor do ciclo diesel de 250 HP acoplado a um gerador elétrico com capacidade de 150 KW. Uma fotografia do sistema eletro-mecânico está representada na Figura 2.1.

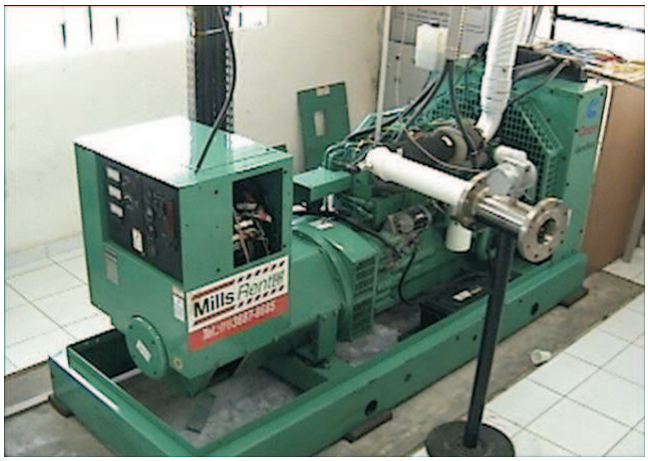

Figura 2.1: Fotografia do sistema eletro-mecânico utilizado no Laboratório de Geração Termoelétrica.

Para utilizar o GNV como combustível é necessário realizar modificações nos kits de GNV disponíveis comercialmente (SILVA et al., 2005). Deste modo, é necessário ter um conhecimento detalhado dos parâmetros elétricos, mecânicos e químicos do sistema eletro-mecânico para realizar as modificações no kit de GNV. Visando esta parametrização, foi

projetado um Sistema de Aquisição da Dados Centralizado (SADC) para adquirir as informações de diversas grandezas físicas distribuídas no sistema eletro-mecânico (SILVA et al., 2005). No entanto, alguns requisitos exigidos pelo sistema não foram alcançados.

2.2 Sistema de Aquisição de Dados Centralizado

O SADC é constituído por uma única unidade de processamento baseado no microcontrolador PIC16F877 (MICROCHIP, 2001), o qual realiza a aquisição dos sinais provenientes de 27 sensores distribuídos em diversos pontos do laboratório. Cada um dos sensores está conectado individualmente ao seu respectivo circuito de condicionamento por meio de cabos com comprimentos variando em torno de 5 à 15 m. Após o condicionamento, os sinais elétricos são amostrados por um *sample/hold* e dois multiplexadores (cada um com 16 canais) garante que essas informações serão digitalizadas com resolução de 10 *bits*, de forma seqüenciada e a uma taxa de amostragem de um segundo pelo microcontrolador. Essas informações são enviadas via *interface* serial para um computador com sistema operacional *Windows* XP, onde um *software* de aplicação feito em MATLAB realiza o armazenamento, processamento e análise prévia dessas informações por meio de *interface* gráfica. Uma fonte de alimentação fornece energia elétrica para os dispositivos do sistema (Para alguns sensores). Na Figura 2.2 representa-se um diagrama de blocos deste sistema.

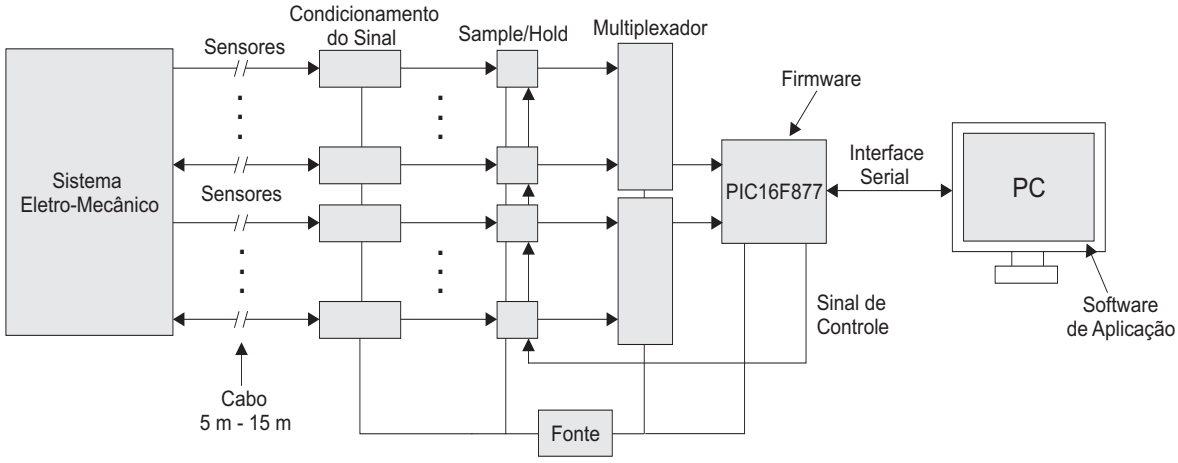

Figura 2.2: Diagrama de blocos do SADC.

Este sistema adquire as informações de quinze sensores de temperatura (Termopares J e K), quatro sensores de pressão, quatro sensores de vazão, três sensores de corrente, e um sensor de rotação. No Capítulo 9 apresenta-se uma descrição sobre esses sensores e seus respectivos circuitos de condicionamento.

2.2.1 Problemas do SADC

Apesar de ter apresentado um bom funcionamento, a taxa de amostragem de um segundo permitiu que apenas as grandezas de temperatura, pressão e vazão fossem analisadas corretamente, pois as variações dessas grandezas são lentas. No entanto, as grandezas elétricas e mecânicas, as quais necessitam de uma taxa de amostragem muito menor (cerca de centenas de microsegundos) não puderam ser analisadas de forma precisa. Além da incapacidade de satisfazer os requisitos de tempo real da aplicação, este sistema apresenta os seguintes problemas:

- Grande quantidade de cabos, ocasionada pela conecção ponto a ponto de cada um dos 27 sensores;
- Sobrecarga da unidade de processamento. A grande quantidade de computação necessária para o processamento torna o sistema incapaz de realizar atividades de controle em tempo real;
- Limitação da quantidade de sensores. O sistema é capaz de adquirir no máximo as informações de 32 sensores.

Deste modo, é providencial a utilização de um sistema de aquisição de dados capaz de solucionar estes problemas. Devido a natureza distribuída dos sensores em torno do sistema termo-elétrico, e por apresentar os requisitos necessários para resolver os problemas apresentados anteriormente, uma solução com arquitetura de processamento distribuído baseado em uma rede de comunicações torna-se viável.

2.2.2 Vantagens de um SAD com Arquitetura Distribuída

SAD com arquiteturas distribuídas apresentam as seguintes vantagens quando comparadas aos SAD com arquiteturas centralizadas (FREDRIKSSON, 1994), (ETSCHBERGER, 2001), (BROSTER, 2003) e (Sá; BARROS; NETO, 2005):

- Redução da quantidade de fios - Os nós da rede são colocados próximos aos sensores e atuadores, e comunicam-se por meio de um único cabo. Em alguns casos, a quantidade de fios pode ser reduzida em 90% ou mais;
- Redução da complexidade dos nós - Com a adição de vários nós no sistema, a quantidade de funcionalidade que cada nó deverá oferecer diminui bastante. Isto resulta em um *hardware* e *software* bem mais simples, possibilitando uma maior previsibilidade e precisão do controle dos recursos;

- Maior capacidade de processamento - Visto que cada processador será submetido a uma menor carga, existe a possibilidade de implementar *software* de controle mais sofisticados;
- Capacidade de tolerância à falhas - Facilidade de implementar redundância modular a nível dos nós por replicação deles;
- Escalabilidade - É garantido que modificações e a adição de novas funções não limitem a expansibilidade do sistema até um limite superior;
- Composabilidade - Capacidade de manter constantes as propriedades que foram estabelecidas nas várias partes do sistema. É importante esclarecer que do ponto de vista temporal, sistemas de comunicações (*Event-Triggered*) não são composáveis. No entanto, a propriedade de composabilidade temporal é garantida nos sistemas de comunicações (*Time-Triggered*);
- Facilidade de instalação e manutenção - A eletrônica em vários nós podem ser idênticas e intercambiáveis;
- Baixo custo - Apesar de uma arquitetura distribuída requisitar mais *hardware*, à medida que o sistema aumenta, o custo de implementação utilizando uma arquitetura centralizada torna-se mais caro, principalmente por causa da grande quantidade e dos longos comprimentos de cabo. Possíveis reduções de custo foram estimadas na ordem de 40% quando comparadas com arquiteturas centralizadas (ETSCHBERGER, 2001).

2.3 Sistema de Aquisição de Dados Proposto

O sistema de aquisição de dados proposto possui uma arquitetura distribuída, baseada na rede de comunicações CAN/TTCAN. O sistema será constituído por vários nós distribuídos no laboratório conforme a localização dos sensores. Um diagrama de blocos do sistema proposto está representado na Figura 2.3.

Um dos requisitos deste SAD é utilizar os sinais provenientes dos sensores já instalados no laboratório. No entanto, foi possível aproveitar apenas os sinais dos sensores de pressão, vazão, corrente e rotação. Os sensores de temperatura (Termopares) apresentaram comportamentos indesejados, principalmente quanto à calibração. Deste modo, decidimos instalar alguns sensores de temperatura redundantes e adquirir as informações de temperatura em outros pontos de medição. Com isso, o número de sensores inicialmente foi reduzido para 14, dos quais: quatro sensores de temperatura; três sensores de pressão; três sensores de vazão; três sensores de corrente; e um sensor de rotação.

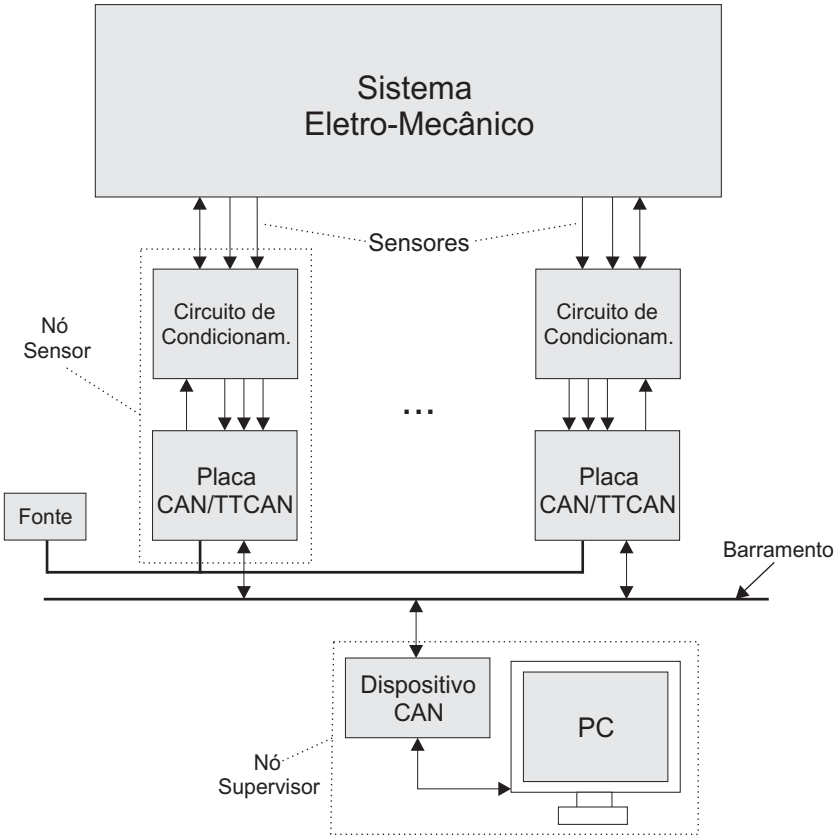

Figura 2.3: Diagrama de blocos do sistema de aquisição de dados proposto.

A especificação e projeto dos componentes do sistema de aquisição de dados proposto será detalhado no Capítulo 5.

2.4 Redes de Comunicações

A rede de comunicação é o principal componente de um sistema distribuído. Atualmente, existem diversas redes com diferentes características disponíveis no mercado. Nas seções seguintes serão apresentados alguns conceitos e definições básicas envolvendo uma rede de comunicação. Algumas das principais características serão apresentadas.

2.4.1 Modelo de Referência

A especificação da maioria das redes de comunicações é baseada no modelo de referência OSI (*Open Systems Interconnections*) arquitetado pela ISO (*International Standardization Organization*), para suportar o desenvolvimento e implementação de padrões de comunicações abertos. Isto significa que o sistema de comunicação está aberto à comunicação com outros equipamentos de diferentes classes, fabricantes, modelos e etc (STEMMER, 2001).

O modelo OSI consiste em uma arquitetura dividida em sete camadas, conforme representado na Figura 2.4. Cada camada possui funções próprias e bem definidas. Uma camada fornece seus serviços para a camada imediatamente acima e utiliza os serviços da camada imediatamente abaixo. Estas camadas são organizadas em duas classes distintas: as camadas superiores, que compreendem desde a camada de sessão até a camada de aplicação, cujos serviços são orientados a resolução de questões envolvendo as aplicações; e as camadas inferiores, que compreendem desde a camada física até a camada de transporte, cujos serviços estão relacionados com a transmissão das informações pela rede.

Figura 2.4: Esquema do modelo de referência OSI.

Embora seja um modelo com sete camadas, algumas delas não são relevantes para sistemas de comunicações voltados para aplicações de automação e controle. A maioria dos protocolos projetados para estas aplicações são baseados apenas nas camadas física e de enlace de dados, e em alguns protocolos a camada de aplicação, a qual fornece os meios para o programa do usuário acessar a rede. O uso apenas dessas camadas aumenta a eficiência do protocolo e reduz consideravelmente o tempo de latência causado pelo *software* envolvido na aplicação (ETSCHBERGER, 2001).

Camada Física

A camada física tem como objetivo assegurar o transporte de dados, representados por um conjunto de *bits* via um meio de transmissão. Esta camada fornece as características mecânicas, elétricas, funcionais e de procedimento para ativar, manter e finalizar a comunicação entre os nós da rede. Questões como a representação, codificação e sincronização do *bit*, a taxa de transmissão, tipo de comunicação e a *interface* e unidade de acesso ao meio físico (*Transceiver*), são definidas nesta camada.

Camada de Enlace

A camada de enlace é responsável basicamente por duas funções: controle de acesso ao meio, o qual é utilizado para resolver o problema de mais de um nó tentar transmitir uma mensagem no mesmo instante; e a definição das mensagens, as quais contém informações de dados e controle. Os mecanismos de detecção, sinalização e correção de erros são especificados nesta camada, assim como os mecanismos de reconhecimento das mensagens.

2.4.2 Tipos de Protocolos de Comunicações

Os protocolos de comunicações podem ser diferenciados entre protocolos orientados a mensagem, e protocolos orientados ao nó.

Orientado a Mensagem

Neste tipo de protocolo, a troca de informações é baseada nos identificadores das mensagens. Uma mensagem transmitida por um nó é identificada por um único identificador. Este identificador caracteriza a informação contida na mensagem e o seu nível de prioridade, caso o mecanismo de acesso ao meio seja não-determinístico.

Orientado ao Nó

Neste tipo de protocolo, a troca de mensagens entre dois ou mais nós é baseada no endereçamento do nó. Geralmente, as mensagens transmitidas no meio de comunicação contém informações da fonte de endereço dos nós. Então, uma mensagem é enviada para um nó específico ou para um grupo de nós. Endereços especiais são reservados para a transmissão de mensagens direcionadas para um grupo ou todos os nós (*Broadcasting*).

2.5 Controle de Acesso ao Meio

Em sistemas de comunicações em rede é possível que dois ou mais nós tentem acessar o meio de transmissão ao mesmo tempo. Para contornar essa situação, é necessário que o protocolo de rede utilize algum mecanismo para controlar o acesso ao meio de transmissão. Esse mecanismo é normalmente chamado de MAC (*Medium Access Control*) - Controle de Acesso ao Meio. O mecanismo MAC determina o desempenho de uma rede em termos de latência de tempo e comportamento em tempo real. Deste modo, o mecanismo MAC pode ser considerado como um critério decisivo na escolha de um barramento de campo. Os mecanismos de controle de acesso ao meio podem ser classificados como aleatórios ou determinísticos. Os mecanismos aleatórios seguem o paradigma de comunicação *Event-*

Triggered. Já os mecanismos determinísticos seguem o paradigma de comunicação *Time-Triggered*.

2.5.1 Mecanismos de Acesso ao Meio Aleatórios

Em uma rede com mecanismo de acesso ao meio aleatório, os nós antes de tentarem transmitir, observam se o meio de transmissão está ou não ocupado. Caso esteja livre, qualquer nó pode tentar transmitir a sua mensagem. Devido a capacidade de detectar a ausência ou não de atividade no meio de transmissão, os mecanismos de acesso ao meio aleatório são chamados de CSMA (*Carrier-Sense Multiple Access*) - Acesso Múltiplo com Detecção de Portadora. Os mecanismos CSMA podem ser classificados quanto a presença ou não de colisão. Uma colisão ocorre quando mais de um nó tenta transmitir uma mensagem ao mesmo tempo.

Acesso ao Meio com Colisão - CSMA/CD

Mecanismos onde colisões podem ocorrer e serem detectadas, são chamadas de CSMA/CD (*Carrier-Sense Multiple Access/Collision Detect*). Por exemplo, a Ethernet é um protocolo de comunicação que utiliza o mecanismo CSMA/CD. Nesse caso, um nó pode transmitir somente se o meio de transmissão estiver livre. Caso dois nós tentem transmitir ao mesmo tempo, ambos abortam a transmissão e esperam por um tempo aleatório, o qual é normalmente menor que o tempo de transmissão de uma mensagem, e em seguida, observam se o meio de transmissão está livre. Caso esteja livre, o nó tentará transmitir novamente a sua mensagem (KUROSE; ROSS, 2003). Devido ao tempo de espera aleatório após a detecção de uma colisão, a Ethernet não é usada em aplicações em tempo real. Mesmo conhecendo o tráfego da rede com antecedência, não é possível determinar a latência no pior caso de uma mensagem.

Acesso ao Meio sem Colisão - CSMA/CA

Mecanismos em que as colisões são minimizadas ou completamente evitadas são chamadas de CSMA/CA (*Carrier-Sense Multiple Access/Collision Avoidance*). O protocolo CAN é um exemplo de rede que utiliza o método CSMA/CA. Nesse mecanismo, um nó pode tentar transmitir uma mensagem somente quando o meio de transmissão estiver livre. No entanto, caso dois ou mais nós tentem transmitir ao mesmo tempo, a disputa pelo meio de transmissão é solucionada de acordo com a prioridade da mensagem durante a fase de arbitração. No final da fase de arbitração, somente o nó que estiver transmitindo a mensagem com maior prioridade permanecerá no barramento. Neste caso, o tempo de latência no pior caso pode ser determinado.

2.5.2 Mecanismos de Acesso ao Meio Determinístico

Em sistemas com controle de acesso ao meio determinístico, o direito de acessar o meio de transmissão é definido com antecedência durante a fase de projeto do sistema. Deste modo, garante-se que apenas um nó acessará o barramento em um determinado instante de tempo. O direito de acessar o meio de transmissão pode ser determinado por meio de um controle centralizado ou descentralizado.

Acesso ao Meio com Controle Centralizado

Em um acesso ao meio com controle centralizado, uma entidade central (mestre ou unidade de sincronização), o qual é um nó da rede, determina o instante de tempo que cada nó acessará o meio de transmissão. Exemplos de mecanismos com controle centralizado são:

- Mestre-Escravo;
- *Delegated Token.*

Mestre-Escravo

No mecanismo mestre-escravo, um nó da rede chamado de mestre, assume o controle de acesso ao meio. O mestre requisita por meio de uma mensagem direcionada a um determinado escravo, que o mesmo pode transmitir uma mensagem naquele instante de tempo. Um esquema do funcionamento de uma rede de acordo com o princípio mestre-escravo está representado na Figura 2.5. O nó mestre M envia uma mensagem de requisição ao nó escravo E_1, em seguida, o nó E_1 responde enviando uma mensagem com as informações requisitadas. O procedimento segue da mesma forma para os demais escravos da rede, até que o ciclo de tempo seja finalizado. Deste modo, a latência de tempo dos nós escravos pode ser facilmente determinada.

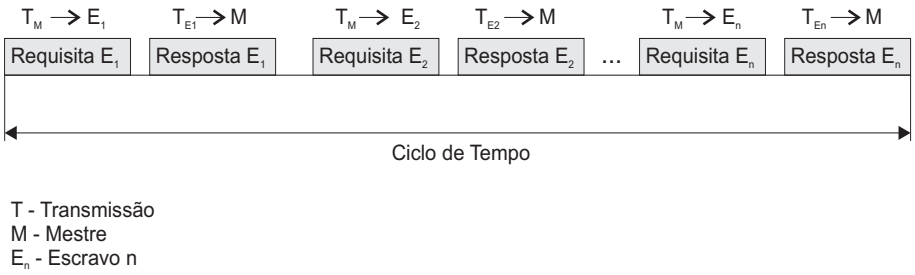

Figura 2.5: Representação do esquema de funcionamento do princípio mestre-escravo.

As principais vantagens desse mecanismo são a simplicidade de implementação e a determinação do tempo de latência máximo de cada nó da rede. Devido o máximo tempo

de latência ser limitado e proporcional a transmissão de uma mensagem, a quantidade de escravos na rede é também limitada. Como desvantagens, os escravos devem esperar uma requisição do mestre para iniciarem a transmissão, e a comunicação é feita apenas de um nó para vários nós, ou seja, a troca de dados entre os escravos pode apenas ser realizada por meio do mestre. Conseqüentemente, um defeito no mestre provoca uma perda total de comunicação no sistema. A rede ASI (KRIESEL; MADELUNG, 1999) e o PROFIBUS-DP (PROFIBUS, 2000) são exemplos de barramentos de campo que utilizam o mecanismo mestre-escravo.

Delegated Token

Neste mecanismo, uma entidade central chamada de mestre ou árbitro do barramento, induz os nós produtores a transmitirem as mensagens requisitadas, chamadas de variáveis, de acordo com uma escala pré-definida no tempo. As mensagens transmitidas pelos produtores podem ser aceitas por todos os nós (consumidores) que tiverem interesse. O mestre atribui o direito de acesso ao barramento transmitindo aos nós uma ficha de requisição da mensagem, que possui um identificador idêntico ao identificador da mensagem requisitada. Após a mensagem requisitada ser enviada por um nó, o direito de acesso ao meio volta para o mestre da rede. Um esquema do funcionamento do mecanismo *delegated token* está representado na Figura 2.6. O mestre M envia uma mensagem de requisição para todos os nós T com uma ficha que possui o identificador da mensagem requisitada. O nó N_1 possui a mensagem requisitada e, conseqüentemente, o direito de acesso ao meio. Em seguida, N_1 transmite a mensagem para todos os nós da rede. Após isso, o direito de acesso ao meio retorna para o mestre da rede. Conforme a escala pré-determinada no tempo, o mestre envia uma nova mensagem de requisição com uma determinada ficha. O procedimento segue como citado anteriormente.

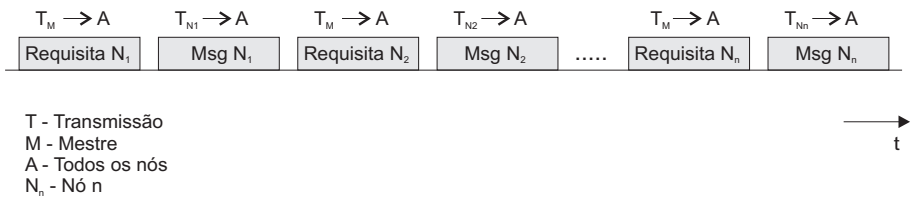

Figura 2.6: Representação do esquema de funcionamento do mecanismo *delegated token*.

Ao contrário do mecanismo mestre-escravo, a estrutura de comunicação é realizada de vários para vários nós da rede, e portanto, é possível realizar a troca de dados entre nós arbitrários de um sistema distribuído.

A grande vantagem desse mecanismo é o alto grau de determinísmo no tempo, associado ao modelo de comunicação orientado a mensagem. Assim como no mecanismo

mestre-escravo, a principal desvantagem é a possibilidade de bloqueio total do sistema ocasionado por um defeito no mestre. Uma solução imediata para evitar a perda total de comunicação do sistema é fornecer uma redundância do mestre. O conceito de comunicação *delegated token* é utilizado pelos barramentos WorldFIP (WORLDFIP, 2007) e ControlNet (CONTROLNET, 2007).

Acesso ao Meio com Controle Descentralizado

Mecanismos com controle descentralizado são mais complexos. No entanto, eles são mais flexíveis, e permanecem em operação mesmo quando um nó do sistema falha ou é desativado. Exemplos de mecanismos com controle descentralizado são:

- Passagem de Ficha;
- TDMA - *Time Division Multiple Access.*

Passagem de Ficha

Neste mecanismo, o direito de acesso ao meio é transferido de nó para nó da rede por meio da passagem de uma ficha. Assim que um nó receber a mensagem que representa a passagem da ficha, ele poderá acessar o meio de transmissão e trocar dados com os demais nós da rede durante um período de tempo limitado, chamado de tempo de posse da ficha. Após o término desse tempo, ou quando não existir mais dados a serem transmitidos pelo nó, a ficha será transferida do atual nó para o nó seguinte, por meio do envio de uma mensagem. Na Figura 2.7 está representado um esquema do funcionamento do mecanismo passagem de ficha.

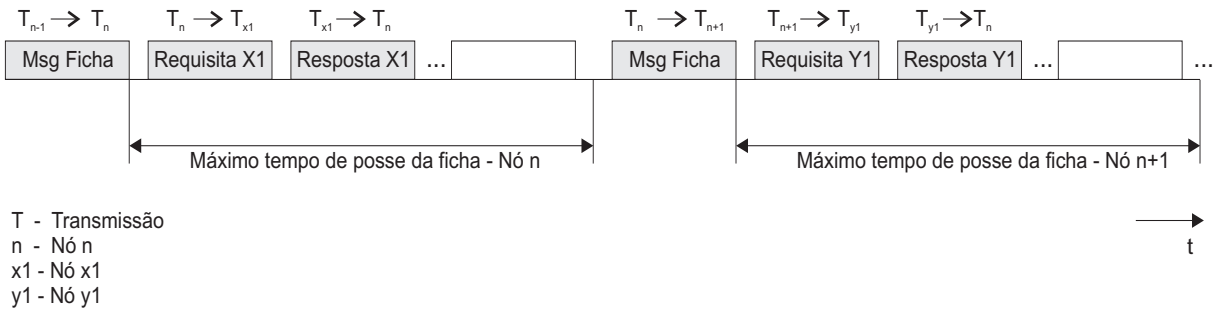

Figura 2.7: Representação do esquema de funcionamento do mecanismo passagem de ficha.

O nó n recebe a ficha do nó n-1, e portanto, possui o direito de acesso ao meio de transmissão. Nesse ponto, o nó n envia uma mensagem de requisição ao nó X1 que, em seguida, retorna por meio de uma mensagem de resposta as informações requisitadas pelo

nó N. A ficha permanece com o nó n até o final do tempo de posse da ficha, conforme indicado na Figura 2.7. Ao término desse tempo, o nó n envia por meio de uma mensagem a ficha para o nó n+1. Em seguida, um procedimento semelhante ocorrerá.

O princípio de passagem de ficha é relativamente complexo, particularmente quando a integração dinâmica e a exclusão de nós for implementada. Além disso, medidas complexas são requisitadas para estabelecer o círculo lógico da passagem de ficha, monitoramento do processo de passagem de ficha ou re-inicialização do círculo lógico após a perda da ficha.

Nesse mecanismo a estrutura de comunicação pode ser realizada de vários para vários nós. A latência de tempo é determinada de acordo com o tempo de posse da ficha e da quantidade de nós da rede. O mecanismo de passagem de ficha é aplicado no barramento ARCNET (ARCNET, 2007) e, no PROFIBUS (PROFIBUS, 2000), quando mais de um dispositivo mestre estiver envolvido.

TDMA - *Time Division Multiple Access*

No mecanismo TDMA, o direito de acesso ao meio é determinado pela progressão de um tempo sincronizado globalmente entre os diversos nós da rede. A sincronização pode ser garantida por meio de uma mensagem específica transmitida por uma unidade central de sincronização. Um ciclo de tempo (espaço de tempo) fixado é dividido em várias janelas de tempo, as quais são atribuídas aos respectivos nós. Deste modo, a colisão entre mensagens transmitidas pelos nós é evitada. Cada ciclo de tempo possui um tamanho fixado e repete-se indefinidamente. Um esquema com um ciclo de tempo e suas janelas está representado na Figura 2.8. Um ciclo de tempo de 400 μs é dividido em cinco janelas de tempo. O ciclo de tempo inicia com a transmissão de uma mensagem pelo nó A e termina com a transmissão de uma mensagem pelo nó E.

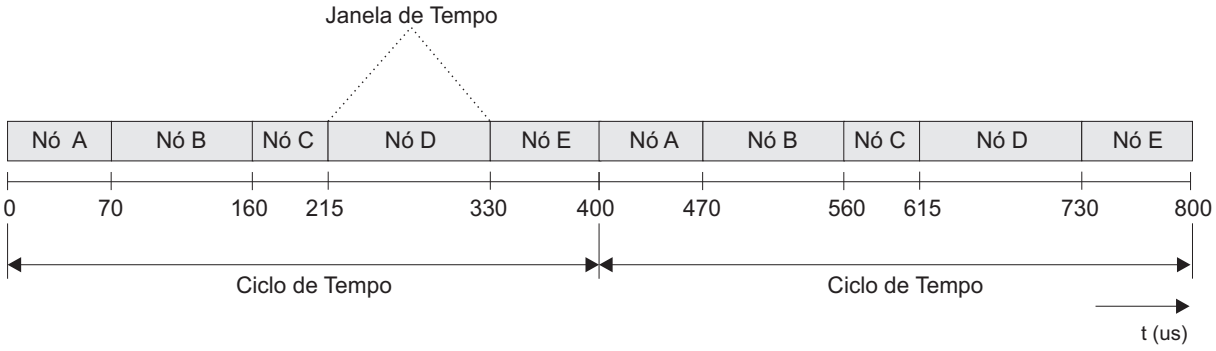

Figura 2.8: Representação do ciclo de tempo e das janelas de tempo no mecanismo TDMA.

Neste mecanismo, todo escalonamento é conhecido com antecedência. A atribuição de janelas para as respectivas mensagens, conforme o ciclo de tempo, é realizado durante a fase de projeto do sistema por meio de alguma ferramenta. O conhecimento prévio

do tráfego no meio de transmissão resulta em uma previsibilidade do sistema. Deste modo, pode-se realizar facilmente uma análise dos tempos de resposta no pior caso das mensagens, e mostrar que os prazos das mensagens serão satisfeitos sob certas condições. A latência de tempo é determinada de acordo com o número de mensagens transmitidas e a taxa de transmissão da rede. Exemplos de barramentos que utilizam o esquema TDMA são o WorldFip (WORLDFIP, 2007), SERCOS (SERCOS, 2007), TTP (TTTECH, 1999).

Algumas redes tentam combinar a comunicação *Event* e *Time-Triggered* em seus protocolos. Exemplos de algumas delas são: Byteflight (AG et al., 2001), FlexRay (FLEXRAY, 2004), FTTCAN (*Flexible Time-Triggered* CAN) (ALMEIDA; FONSECA; FONSECA, 1998) (ALMEIDA; PEDREIRAS; FONSECA, 2002) e *Time-Triggered* CAN (FüHRER et al., 2000) (LEEN; HEFFERNAN., 2002).

2.6 Conclusões

Neste Capítulo descreveu-se o Laboratório de geração termoelétrica e o seu sistema de aquisição de dados centralizado. Os principais problemas deste sistema foram citados, assim como, os requisitos desta aplicação. Um sistema de aquisição de dados distribuído baseado no protocolo CAN/TTCAN foi proposto como solução para estes problemas. Em seguida, apresentou-se alguns conceitos e definições sobre sistemas de comunicações voltados para aplicações de automação e controle.

Capítulo 3

Protocolo de Comunicação CAN

3.1 Introdução

CAN (*Controller Area Network*) é um protocolo de comunicação serial utilizado para controlar sistemas distribuídos em tempo real. Este protocolo foi projetado pela empresa Robert Bosch GmbH na década de 80, para satisfazer os requisitos da engenharia automotiva.

O protocolo CAN pode ser descrito em termos do modelo de referência OSI por três camadas: a camada de enlace de dados, a camada física, e opcionalmente a camada de aplicação que pode ser especificada pelo projetista do sistema ou por alguns dos protocolos da camada superior baseados no CAN como por exemplo, o CAL (*CAN Application Layer*)/CANopen (PFEIFFER; AYRE; KEYDEL, 2003).

A especificação desenvolvida pela Bosch implementa apenas a camada de enlace e parte da camada física, a qual é descrita pelo padrão ISO 11898-1 (ISO, 1999a). O restante da camada física é definida na ISO 11898-2 para aplicações em alta velocidade (ISO, 1999b), e na ISO 11898-3 para aplicações em baixa velocidade (ISO, 1999c). Recentemente, a camada de sessão para a comunicação CAN, designado por protocolo TTCAN (*Time-Triggered* CAN) foi definida na ISO 11898-4 (ISO, 2004). Na Figura 3.1 representa-se o modelo OSI e as camadas utilizadas pelo protocolo CAN.

Os dispositivos CAN implementam em *hardware* apenas o padrão ISO 11898-1. Existem no mercado alguns dispositivos CAN que possuem os requisitos para implementar parte do TTCAN por *software*, por exemplo, o controlador CAN MCP2515 (MICROCHIP, 2005). Atualmente, a Bosch está desenvolvendo um *chip* que implementa em *hardware* o protocolo TTCAN (BOSCH, 2007).

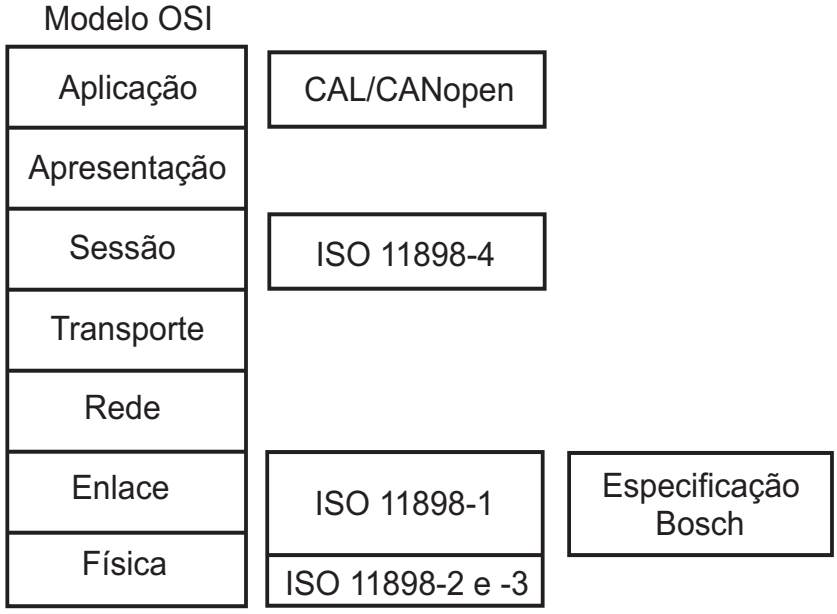

Figura 3.1: Camadas do modelo OSI utilizadas pelo protocolo CAN.

A especificação Bosch (versão CAN 2.0) (BOSCH, 1991) é constituída por duas partes: a parte 2.0A e a parte 2.0B. A diferença entre as duas partes, é que a parte A descreve a mensagem com formato padrão e a parte B com formato padrão e estendido. No entanto, elas são compatíveis.

3.1.1 Aplicações

Apesar de ter sido projetado para aplicações em sistemas embarcados automotivos (BAILEY, 1996) e (JOHANSSON; TöRNGREN; NIELSEN, 2005), pode-se encontrar diversas aplicações utilizando o CAN, tais como: na robótica (BOURDON; RUAUX; DELAPLACE, 1996), (KAISER; SCHAEFFER, 2001) e (MARTíNEZ et al., 2001); em máquinas (FREDRIKSSON, 1994); em helicópteros (SANTOS; STEMMER, 2002); na domótica (MORAES; AMORY; JúNIOR, 2001); em telescópios (MANCINI et al., 2001) e (BROOKS, 2000); no setor de transmissão de energia elétrica (BENEDETTI; NETTO, 2003); na geofísica (FONSECA; XAVIER; CARDOSO, 2003) entre outras.

Em geral, utiliza-se o protocolo CAN em sistemas embarcados distribuídos, onde dispositivos inteligentes precisam se comunicar. As razões pela vasta utilização em diversos domínios de aplicação são devido as suas características, a grande disponibilidade de *chip* de baixo custo e de *interface* integradas ao protocolo. Em 2003 foram vendidos cerca de 400 milhões de dispositivos CAN (CIA, 2007).

3.2 Rede CAN

Uma rede CAN consiste basicamente de nós que utilizam um meio compartilhado com dois fios (CAN_H e CAN_L) para trocar informações, como representado na Figura 3.2.

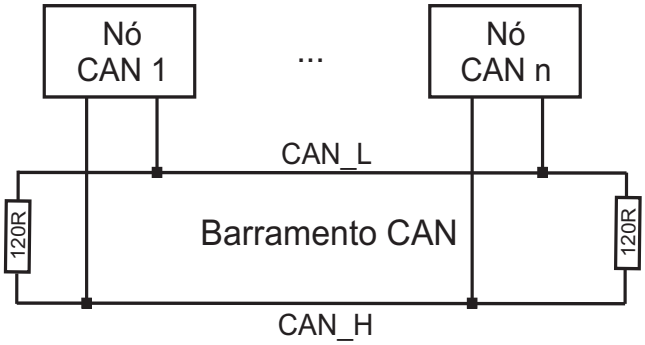

Figura 3.2: Representação de uma rede CAN.

3.2.1 Nó CAN

Para implementar um nó CAN são necessários três componentes básicos: um *transceiver*, um controlador CAN e um microcontrolador. Na Figura 3.3 representa-se um diagrama de blocos de um nó CAN.

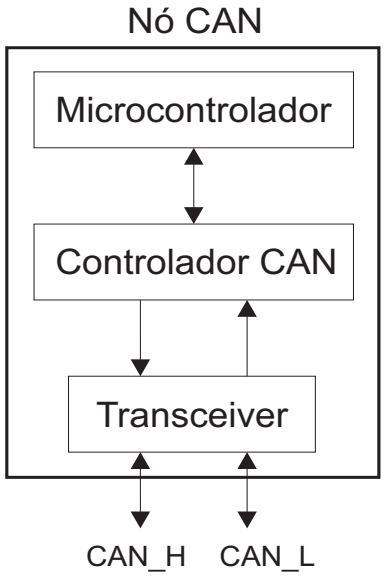

Figura 3.3: Diagrama de blocos de um nó CAN.

O microcontrolador tem a função de configurar e gerenciar os seus periféricos e o controlador CAN, de acordo com a aplicação. Ele realiza tarefas como: aquisição das

informações obtidas pelos sensores; envio dos sinais de controle para os atuadores conectados a ele; recepção das informações das mensagens CAN enviadas pelo controlador entre outras. O controlador basicamente dispõe das funções necessárias para implementar a especificação CAN. Ele é constituído basicamente por *buffers* de transmissão e recepção, registradores para configuração das máscaras, filtros e temporização dos *bits* das mensagens, e módulos para execução em hardware do protocolo CAN. O *transceiver* é responsável pela conexão do controlador com o meio de transmissão.

3.2.2 Controlador CAN

O mercado oferece diversos controladores CAN. Eles podem ser classificados quanto ao nível de integração com os demais dispositivos que constituem um nó. Basicamente, pode-se encontrar as seguintes arquiteturas:

- *Stand-Alone*;
- Integrada;
- *Single-Chip*.

Em uma arquitetura *Stand-Alone*, o controlador é separado do microcontrolador e do *transceiver*. A comunicação entre o controlador e o microcontrolador é realizada através de alguma *interface* simples, como a SPI (*Serial Peripheral Interface*). Na Figura 3.4 representa-se um diagrama de blocos de um nó CAN com a arquitetura *Stand-Alone*.

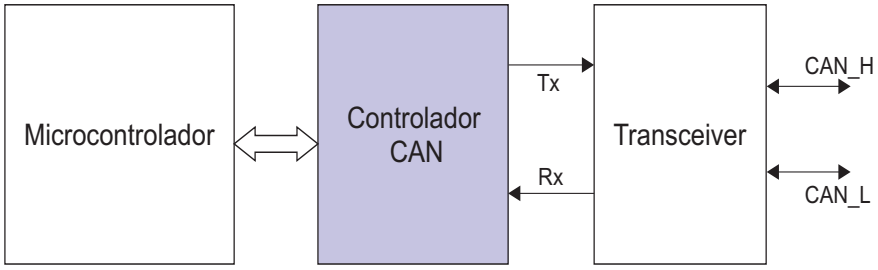

Figura 3.4: Diagrama de blocos de um nó CAN com a arquitetura *Stand-Alone*.

Na arquitetura Integrada, o controlador e o microcontrolador estão integrados em um único *chip* e separados do *transceiver*. A representação em diagrama de blocos de um nó com essa arquitetura está ilustrada na Figura 3.5.

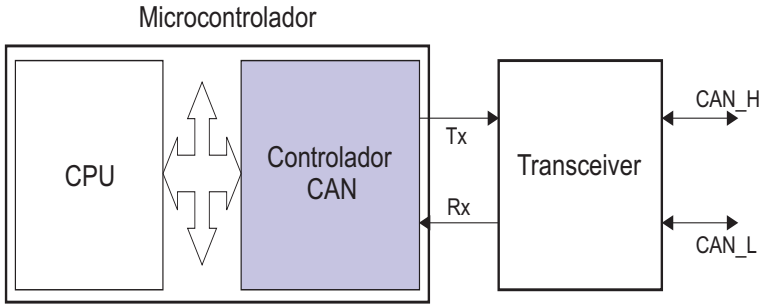

Figura 3.5: Diagrama de blocos de um nó CAN com a arquitetura integrada.

A arquitetura *Single-Chip* dispõe de todos os dispositivos integrados em um único *chip*, como representado na Figura 3.6.

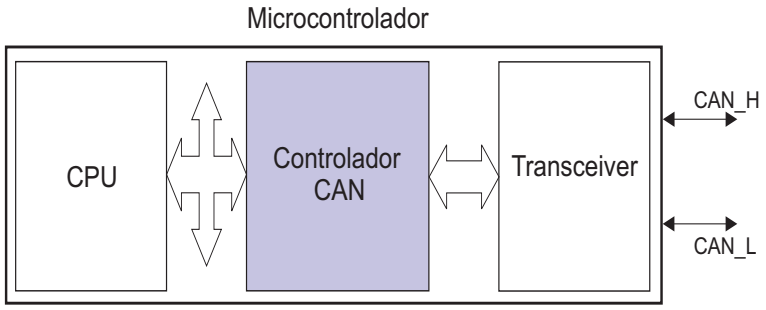

Figura 3.6: Diagrama de blocos de um nó CAN com a arquitetura *Single-Chip*.

Em uma arquitetura *Stand-Alone*, existe uma flexibilidade na escolha do microcontrolador. Além disso, é possível reutilizar o código em diferentes tipos de processadores. No entanto, o fato da comunicação entre o controlador e o microcontrolador ser realizada por meio de um barramento externo, reduz consideravelmente a taxa de comunicação quando comparada aos outros dois tipos de arquitetura, visto que elas utilizam um barramento de dados e de endereço interno de alta velocidade.

É possível também encontrar no mercado controladores CAN integrados em placas com *interface* RS232, PCI (*Peripheral Component Interconnect*) e USB (*Universal Serial Bus*). Estes dispositivos são normalmente utilizados para implementação de nós supervisores em um sistema de aquisição de dados.

3.2.3 *Transceiver*

A conexão entre um controlador CAN e o meio físico de transmissão é realizada pelo *transceiver*. Basicamente, o *transceiver* converte um nível lógico em um sinal compatível com o meio de transmissão utilizado. No mercado existem diversos *transceiver* compatíveis com os padrões ISO 11898-2 e ISO 11898-3. Na Figura 3.7 representa-se um esquema de

funcionamento de um *transceiver* conectado a um meio de transmissão com dois fios de acordo com o padrão ISO 11898-2.

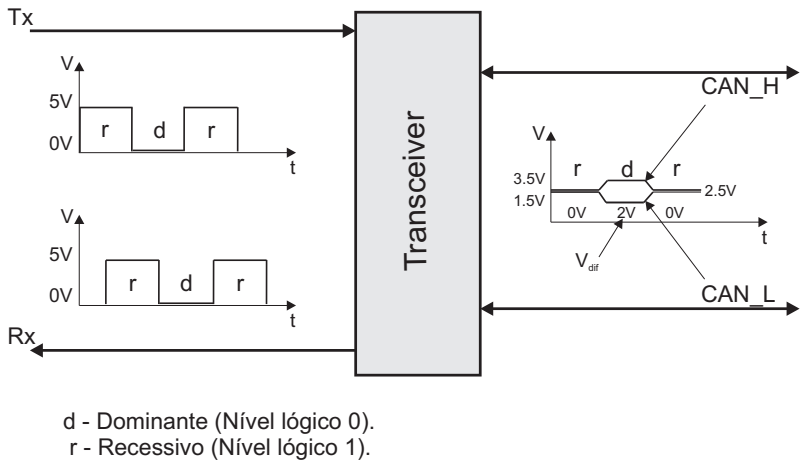

Figura 3.7: Representação do funcionamento de um *transceiver* conectado a um barramento diferencial com dois fios.

3.2.4 Meio de Transmissão

É possível utilizar alguns meios físicos de transmissão, tais como: par de fios, um único fio[1], rádio freqüência e fibra óptica. Geralmente, utiliza-se um par de fios com sinalização diferencial, comumente chamado de barramento CAN.

Neste meio de transmissão, as linhas do barramento devem ser terminadas com uma resistência de 120 Ohms para evitar reflexões do sinal. O nível lógico do sinal é indicado pela diferença de tensão entre os dois fios, CAN_H e CAN_L, conforme a Equação 3.1.

$$V_{dif} = V_{CAN\_H} - V_{CAN\_L} \tag{3.1}$$

O nível lógico recessivo é representando por $V_{dif} < 0.5$ e o nível lógico dominante por $V_{dif} > 0.9$. A sinalização por diferença de tensão consegue minimizar efeitos de interferências eletromagnéticas. Como a interferência afeta ambos os fios com a mesma intensidade, a diferença de tensão entre eles permanecerá constante (Figura 3.8).

[1]Especificado pelo padrão SAE J2411. Utilizado na parte eletrônica do motor de veículos.

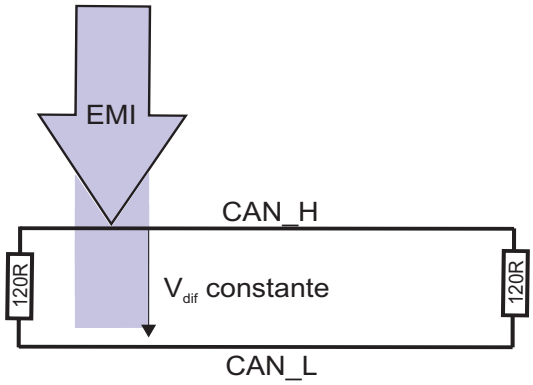

Figura 3.8: Representação do processo de minimização do efeito de interferências eletromagnéticas.

3.3 Características Básicas

A maioria das implementações de uma rede CAN utiliza a topologia barramento, mas é possível implementar redes na topologia árvore e estruturas hierárquicas através de repetidores ou roteadores. O número de nós por rede é teoricamente limitado pelo protocolo, mas depende principalmente das características elétricas dos *transceivers* utilizados. Por exemplo, é possível ter em uma mesma rede CAN em que os nós utilizem o *transceiver* MCP2551 (MICROCHIP, 2003), até 112 nós. Além disso, a utilização de repetidores permite aumentar o número de nós da rede.

O comprimento máximo do barramento em uma determinada taxa de transmissão é limitada pelo tempo de propagação do sinal ao longo do meio de transmissão. A taxa máxima de transmissão permitida é de 1 Mbps para um barramento com comprimento de até 40 m.

Possui a propriedade *broadcast* atômico, ou seja, uma mensagem será transmitida para todos os nós da rede uma única vez, e na ordem em que foi transmitida.

É um protocolo baseado em mensagens. Em uma rede CAN a troca de informações é baseada no identificador da mensagem. Conforme representado na Figura 3.9, o microcontrolador do nó que deseja transmitir uma mensagem envia para o controlador CAN os dados a serem transmitidos (1). Então, o controlador forma uma mensagem com um determinado identificador (Definido pelo projetista do sistema), as informações de controle e os dados. Assim que o barramento estiver livre, o controlador envia a mensagem para o barramento (2), realizando a comunicação *broadcast* (3). Em seguida, todos os nós da rede verificam se a mensagem é relevante ou não para ele de acordo com o identificador da mensagem recebida (4). O processo de verificação da relevância da mensagem é realizado

através de mecanismos de filtragem dos identificadores, os quais são implementados no controlador do nó, e que também são definidos durante o projeto do sistema. Portanto, uma mensagem pode ser aceita por um, vários ou todos os nós da rede. Caso a mensagem seja aceita, o controlador indicará por meio de uma interrupção ao microcontrolador, que em seguida receberá as informações desta mensagem (5). Se a mensagem não for aceita pelo mecanismo de filtragem, ela será automaticamente rejeitada.

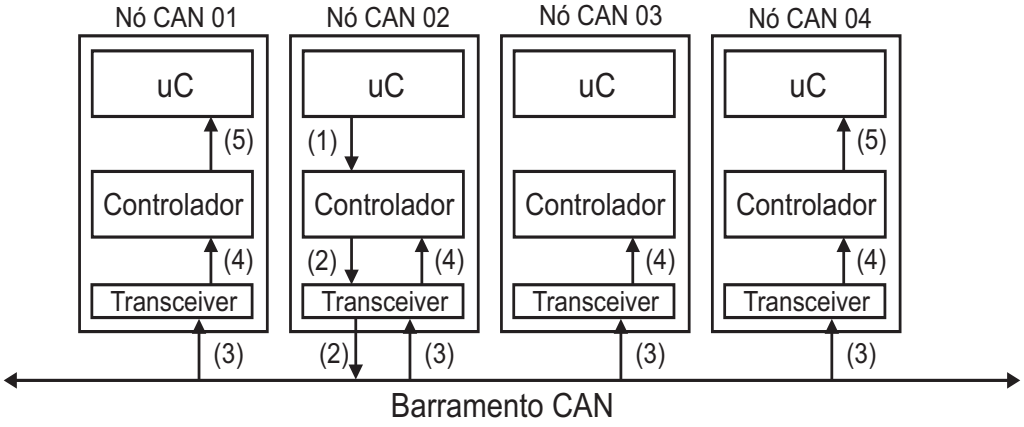

Figura 3.9: Representação da comunicação *broadcast* em uma rede CAN.

As mensagens possuem prioridade fixas com relação ao acesso do barramento, indicadas pelo identificador da mensagem. Além disso, as mensagens não sofrem preempção.

O protocolo possui capacidade multi-mestre. O acesso ao barramento não é controlado por uma unidade de controle mestre. Vários nós da rede podem iniciar em simultâneo a transmissão de uma mensagem assim que o barramento estiver livre. Porém, apenas a mensagem com a maior prioridade acessará primeiro o barramento. O controle de acesso ao barramento é solucionado pelo mecanismo CSMA/CA (*Carrier Sense Multiple Acess with Collision Avoidance*)[2].

O comprimento máximo de dados de uma mensagem CAN é limitada em 8 *bytes*. Para garantir a integridade das informações, existem mecanismos para detecção e recuperação dos erros. Os controladores CAN implementam dois contadores de erros que classificam os nós em ativo, passivo ou desconectado[3]. Então, é possível detectar e desativar nós defeituosos da rede. Inicialmente, todo nó é classificado como ativo.

3.4 Identificador da Mensagem CAN

Cada mensagem possui um identificador único. Dependendo do formato da mensagem é possível ter identificadores com 11 *bits* (formato padrão - CAN 2.0A) ou 29 *bits* (formato

[2]Ver seção 3.8

[3]Ver seção 3.10

estendido - CAN 2.0B). Portanto, é possível ter no máximo 2048 (2^{11}) identificadores diferentes no formato padrão, e mais de 536 milhões (2^{29}) de identificadores no formato estendido. Em geral, as aplicações utilizam apenas mensagens no formato padrão, pois elas possuem um melhor fator de utilização da mensagem.

O identificador possui duas importantes funções: caracterizar o conteúdo da mensagem de forma que ela seja filtrada nos nós receptores e determinar a prioridade das mensagens relacionada com o acesso ao barramento.

De acordo com as características elétricas implementadas pelos controladores CAN, os nós da rede estão conectados ao barramento por meio do mecanismo *wired-AND*. Isso significa que o nível lógico '0' é dominante e o nível lógico '1' é recessivo. De acordo com esse mecanismo, a mensagem que tem o identificador com menor valor numérico possui a maior prioridade, e a mensagem que tem o maior valor numérico possui a menor prioridade. Então, mensagens que tenham maior importância no sistema ou tenham um *deadline* curto, podem acessar o barramento com uma pequena latência de tempo, atribuindo a elas identificadores com alta prioridade.

3.5 Codificação dos *Bits*

Os *bits* das mensagens CAN são representados fisicamente de acordo com a codificação NRZ (*Non-Return to Zero*). Nesse esquema de codificação, o nível do sinal permanece constante durante um determinado tempo, chamado de tempo de *bit*[4], conforme representado na Figura 3.10.

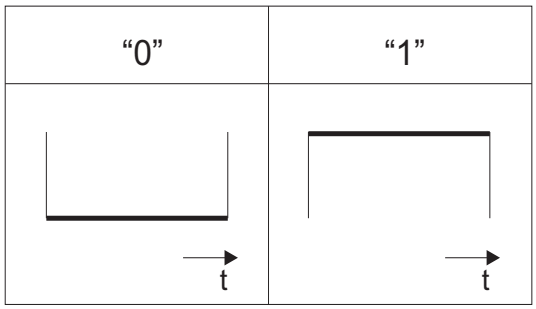

Figura 3.10: Representação de um *bit* pela codificação NRZ.

[4]Ver seção 3.11

3.6 Mecanismo de Inserção de *Bits*

Na codificação NRZ, é possível que uma seqüência grande de *bits* com mesmo nível lógico seja transmitida durante um longo período de tempo, causando a inexistência de bordas de transição, as quais são usadas na sincronização da rede. Portanto, medições adequadas devem ser realizadas para garantir que o intervalo de tempo máximo permissível entre duas bordas de sinal não seja ultrapassado. Para resolver esse problema, o protocolo CAN utiliza o mecanismo de inserção de *bits*. Esse mecanismo provoca uma borda de transição no máximo a cada cinco *bits*. Quando uma seqüência de cinco *bits* com mesmo nível lógico for transmitida, o mecanismo insere automaticamente um *bit* com nível lógico diferente, provocando uma transição de borda. Portanto, a seqüência de *bits* transmitidas em uma mensagem fornecerá um número suficiente de bordas para sincronização dos receptores. Quando os receptores recebem a mensagem, eles automaticamente retiram os *bits* que foram inseridos antes de serem processados pelo microcontrolador. Na Figura 3.11 apresenta-se o princípio do mecanismo de inserção de *bits*.

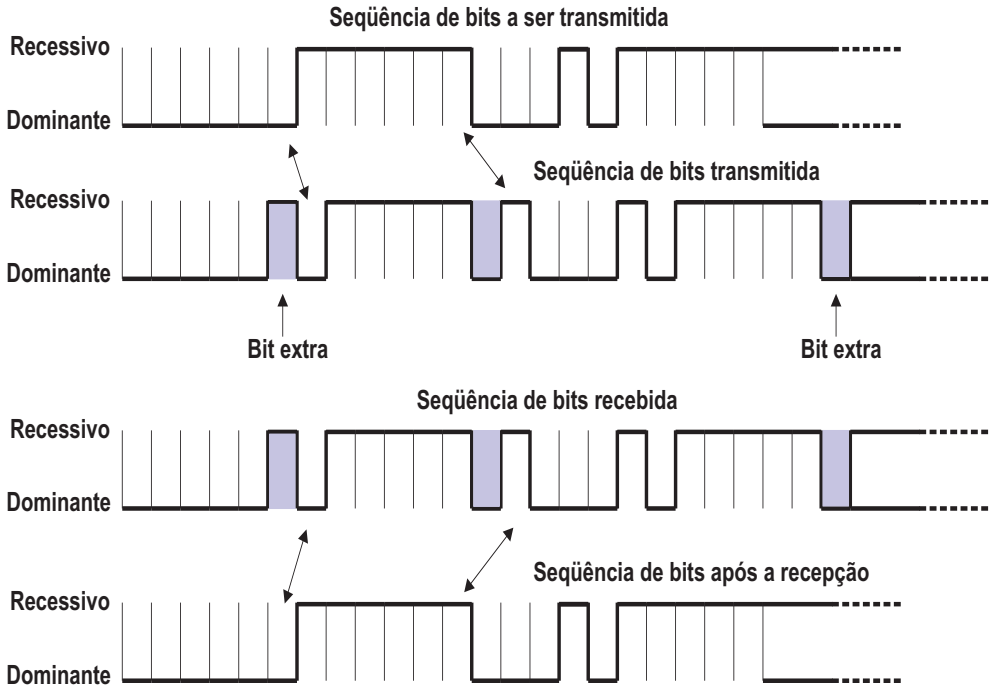

Figura 3.11: Representação do mecanismo de inserção de *bits* utilizado pelo protocolo CAN.

3.7 Tipos das Mensagens

A transmissão e recepção das informações em uma rede CAN são efetuadas e controladas através de quatro tipos diferentes de mensagens:

- Mensagem de Dados;
- Mensagem de Requisição Remota de Dados;
- Mensagem de Erro;
- Mensagem de Sobrecarga.

A mensagem de dados e a mensagem de requisição remota de dados são utilizadas durante a operação normal da rede. A mensagem de dados é utilizada para transferir dados de um nó para os demais nós da rede, e a mensagem de requisição de dados é usada para requisitar dados de um determinado nó da rede.

A mensagem de erro e a mensagem de sobrecarga são utilizadas para sinalizar um estado anormal da rede. A mensagem de erro sinaliza a existência de um erro gerado no barramento, e a mensagem de sobrecarga sinaliza que um nó particular não está pronto para receber mensagens naquele momento. Apesar da existência do mecanismo de sinalização de sobrecarga, a maioria dos controladores CAN não requerem um atraso na recepção da próxima mensagem (ETSCHBERGER, 2001).

Mensagens de dados e de requisição de dados são separadas de mensagens anteriores (dados, requisição remota de dados, erro ou sobrecarga) por um intervalo de tempo chamado de espaço entre mensagens, que será explicado na seção 3.7.3. Nas seções seguintes serão apresentadas apenas as mensagens de dados no formato padrão, a qual será utilizada neste trabalho, e a mensagem de erro.

3.7.1 Mensagem de Dados

Uma mensagem de dados, representada na Figura 3.12, é formada pelos seguintes campos: SOF (*Start of Frame*), arbitração, controle, dados CRC (*Cyclic Redundancy Check*), ACK (*Acknowledgement*) e EOF (*End of Frame*).

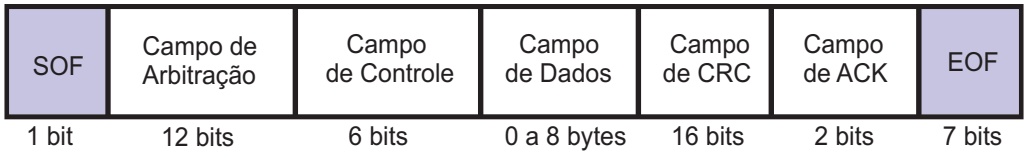

Figura 3.12: Esquema de uma mensagem de dados.

SOF

Este *bit* marca o início de uma mensagem de dados, o qual possui valor fixo em dominante (nível lógico '0'). Todos os nós da rede são sincronizados na transição de borda causada

por esse *bit*. Durante a operação da rede, o *bit* SOF é sempre precedido por *bits* recessivos, os quais são provenientes do intervalo entre mensagem.

Campo de Arbitração

O campo de arbitração é constituído por um identificador com 11 *bits* e pelo *bit* RTR (*Remote Transmit Request*), como representado na Figura 3.13. O *bit* RTR indica se a mensagem é de dados ou de requisição de dados. Se RTR for dominante (nível lógico '0'), a mensagem será de dados. Caso tenha valor recessivo (nível lógico '1'), a mensagem será de requisição de dados.

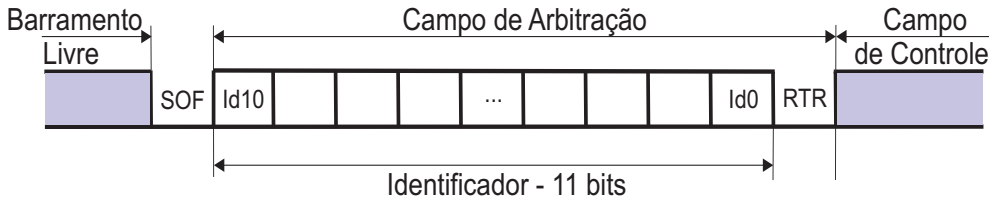

Figura 3.13: Esquema do campo de arbitração de uma mensagem de dados.

Campo de Controle

O campo de controle (Figura 3.14) consiste de seis *bits*, o *bit* IDE, um *bit* reservado 'r0', o qual é sempre transmitido com um valor dominante e quatro *bits* de DLC (*Data Lenght Control*). O *bit* IDE indica se o identificador possui 11 bits (IDE = 0) ou 29 *bits* (IDE = 1).

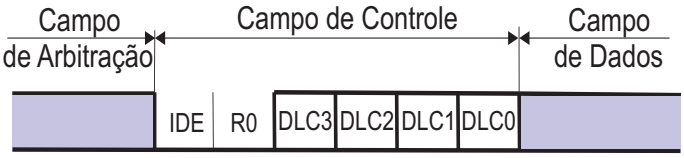

Figura 3.14: Esquema do campo de controle de uma mensagem de dados.

Os quatro *bits* DLC informam a quantidade de *bytes* no campo de dados que serão transmitidos na mensagem. Na Tabela 3.1 apresenta-se todas as possíveis configurações dos *bits* DLC.

Campo de Dados

O campo de dados contém os dados que serão transmitidos em uma mensagem. É possível transmitir por mensagem até 8 *bytes* de dados. Os *bits* mais significativos são transferidos primeiro.

Tabela 3.1: Possíveis configurações dos *bits* DLC.

Número de Bytes de Dados	**DLC3**	**DLC2**	**DLC1**	**DLC0**
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	x (0 ou 1)	x	x

Campo CRC

O campo CRC consiste de 15 *bits* de verificação (sequência CRC) e do *bit* delimitador de CRC, o qual é sempre transmitido com um valor recessivo. Por meio da sequência CRC, um receptor pode verificar se determinados *bits* da mensagem foram corrompidos. A sequência de verificação da mensagem é derivada de um código de redundância cíclica bem aplicado em mensagens com quantidade de *bits* menores que 127, chamado de código BCH (Bose Chaudhuri-Hocquenghem) (KUROSE; ROSS, 2003). O código CRC fornece uma distância de *hamming* de 6, ou seja, é possível detectar até 5 erros em *bits* distribuídos aleatoriamente do SOF até o campo de dados. Um diagrama do campo de CRC de uma mensagem de dados está representado na Figura 3.15.

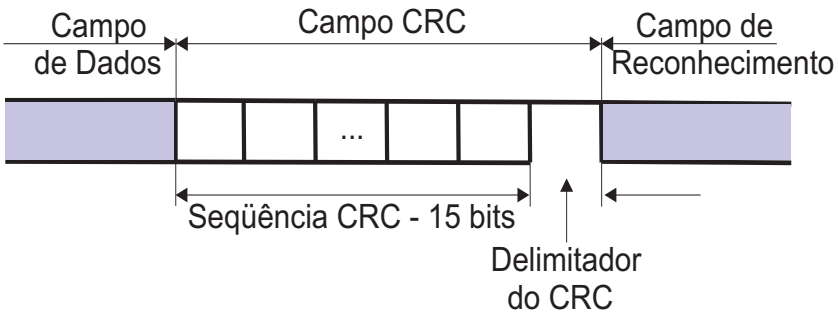

Figura 3.15: Esquema do campo CRC de uma mensagem de dados.

O algoritmo CRC funciona da seguinte forma: os dados da mensagem do SOF até o campo de dados são tratados como um número binário. Este número binário é dividido em módulo-2 por outro número binário, chamado de polinomial geradora. O resto desta

divisão é chamado de CRC *checksum*, o qual é transmitido juntamente com a mensagem na seqüência CRC de 15 *bits*. Ao receber a mensagem, o receptor realiza o mesmo procedimento feito pelo transmissor. Se o resto desta divisão for igual a seqüência CRC recebida, a mensagem foi recebida sem erro. Caso contrário, a mensagem contém ao menos um erro. Na Equação 3.2 apresentra-se a equação polinomial utilizada no algoritmo de CRC pelo protocolo CAN.

$$P(x) = x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1 \tag{3.2}$$

Campo de Reconhecimento (ACK)

O campo de reconhecimento consiste de dois *bits*, o *bit ACK Slot* e o *bit* Delimitador ACK. O transmissor da mensagem envia ambos os *bits* com valor recessivo. Ao receber a mensagem, o receptor verifica se a seqüência CRC recebida está correta. Caso esteja correta, o receptor envia o reconhecimento para o transmissor desta mensagem sobrescrevendo o valor do *bit ACK-Slot* de recessivo para dominante. O transmissor de uma mensagem espera o reconhecimento de pelo menos um receptor. Na Figura 3.16 representa-se o campo de reconhecimento de uma mensagem de dados.

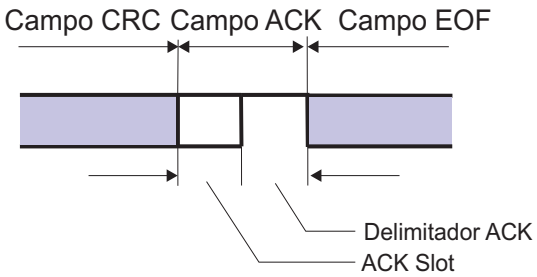

Figura 3.16: Esquema do campo de reconhecimento de uma mensagem de dados.

O mecanismo de reconhecimento indica a recepção sem erros de pelo menos um nó. No entanto, isso não é suficiente para garantir uma consistência dos dados transmitidos no sistema. Um erro detectado por algum nó será sinalizado pelo envio de uma mensagem de erro. Um nó que detecta um erro de transmissão por meio da verificação do CRC, sinaliza o erro somente após o campo de reconhecimento. Então, é possivel que um transmissor receba o reconhecimento de um nó receptor e em seguida, detecte uma mensagem de erro transmitida por outro nó receptor indicando que ocorreu um erro local. Neste caso, o transmissor será forçado a transmitir novamente a mensagem.

Campo EOF

O campo EOF (*End Of Frame*) indica o final da mensagem. Este campo é composto por 7 *bits* recessivos, conforme representado na Figura 3.17.

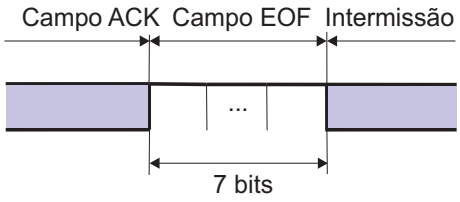

Figura 3.17: Esquema do campo EOF de uma mensagem de dados.

3.7.2 Mensagem de Erro

A detecção de algum erro durante a transmissão ou recepção de um dos quatro tipos de mensagens é sinalizada por meio de uma mensagem de erro, a qual é constituída pelos campos *flag* de erro e delimitador de erro, como representado na Figura 3.18.

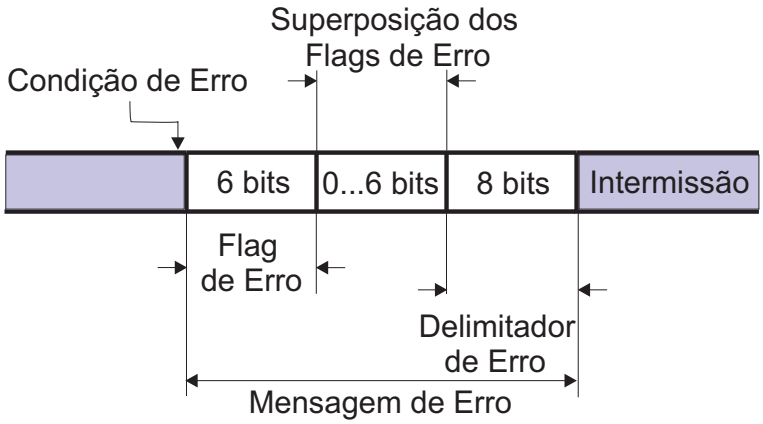

Figura 3.18: Esquema de uma mensagem de erro.

Flag de Erro

Existem dois tipos de *flag* de erro: o ativo, que é transmitido por nós ativos e consiste de seis *bits* dominantes, e o passivo, que é transmitido por nós passivos e consiste de seis *bits* recessivos.

Quando um nó ativo (transmissor ou receptor) detecta um erro na atual mensagem que está sendo enviada ao barramento, ele sinaliza esta condição de erro enviando um *flag* de erro ativo, o qual sobrescreve a mensagem com erro. Assim que os outros nós da rede, ativos ou passivos, detectarem um erro na mensagem, eles enviam os seus respectivos

flags de erro. Como um *flag* de erro viola a regra de inserção de *bits*, um nó detectará um erro no barramento no máximo no final do sexto *bit* do *flag* de erro transmitido pelo nó que iniciou a sinalização do erro. Dependendo do ponto no tempo em que os outros nós detectaram a condição de erro, é possível ter uma seqüência entre 6 e 12 *bits* nesse campo.

Caso um nó passivo e transmissor da atual mensagem no barramento detectar um erro, ele enviará um *flag* de erro passivo. No entanto, os receptores somente detectarão um erro, se o *flag* de erro estiver sendo transmitido no lugar de um campo da mensagem codificado pelo método da inserção de *bits*(SOF até a seqüência CRC), exceto no campo de arbitração e nos últimos seis *bits* do campo CRC. *Flags* de erro passivo enviados por nós receptores não sobrescrevem mensagens com erro. Portanto, não são capazes de sinalizar um erro no barramento.

Um nó inicia a transmissão de um *flag* de erro, assim que ele detectar uma condição de erro, exceto no caso de um erro de CRC. Neste caso, a transmissão de um *flag* de erro é iniciada no próximo *bit* após o delimitador ACK, de modo a não atrapalhar a função de reconhecimento da mensagem.

Delimitador de Erro

Uma mensagem de erro é finalizada por uma seqüência de oito *bits* recessivos. Após a transmissão do seu *flag* de erro, os nós transmitem *bits* recessivos e monitoram o barramento até reconhecer um *bit* recessivo. Quando um *bit* recessivo for detectado, eles enviam mais sete *bits* recessivos, finalizando o campo delimitador de erro. Com este mecanismo, um nó é capaz de determinar se ele foi o primeiro nó a transmitir o *flag* de erro e consequentemente, foi o primeiro a detectar uma condição de erro. Neste caso, a transmissão do primeiro *bit* recessivo do delimitador de erro não é afetado imediatamente, devido aos outros nós que ainda não enviaram os seus *flags* de erro. Então, após transmitir o seu *flag* de erro ativo, o nó que sinalizou primeiro o erro detectará no barramento no mínimo mais um *bit* dominante proveniente dos *flags* de erro dos demais nós. O processo de identificação do nó que detectou primeiro o erro é a base para o confinamento de erros dos nós, que será explicado na seção 3.10.

Na Figura 3.19 representa-se um esquema da formação de uma mensagem de erro no caso de um nó ativo detectar um erro local pela verificação da seqüência CRC. Todos os nós são ativos. O nó Y detecta um erro pela verificação da seqüência CRC (1). O nó Z recebe a mensagem corretamente e portanto, faz o reconhecimento no *bit slot-ACK*, enviando um *bit* dominante que sobrescreve o *bit* recessivo transmitido pelo nó X (2). Devido o mecanismo de reconhecimento ser preservado, o nó Y só sinaliza o erro detectado após o campo ACK, no primeiro *bit* do campo EOF (3). Em seguida, os nós X e Y detectam

um erro de formato no ponto (4), pois eles esperavam um *bit* recessivo (campo EOF). Então, eles enviam os seus *flags* de erro no próximo *bit* (5). Um *flag* de erro ativo é formado no barramento com um comprimento total de 7 *bits* dominantes. No ponto (6), o nó Y detecta que ele foi o primeiro nó que sinalizou o erro. Após o término dos oito *bits* recessivos do delimitador de erro e o fim do campo intermissão, o nó X tenta retransmitir a mensagem (7).

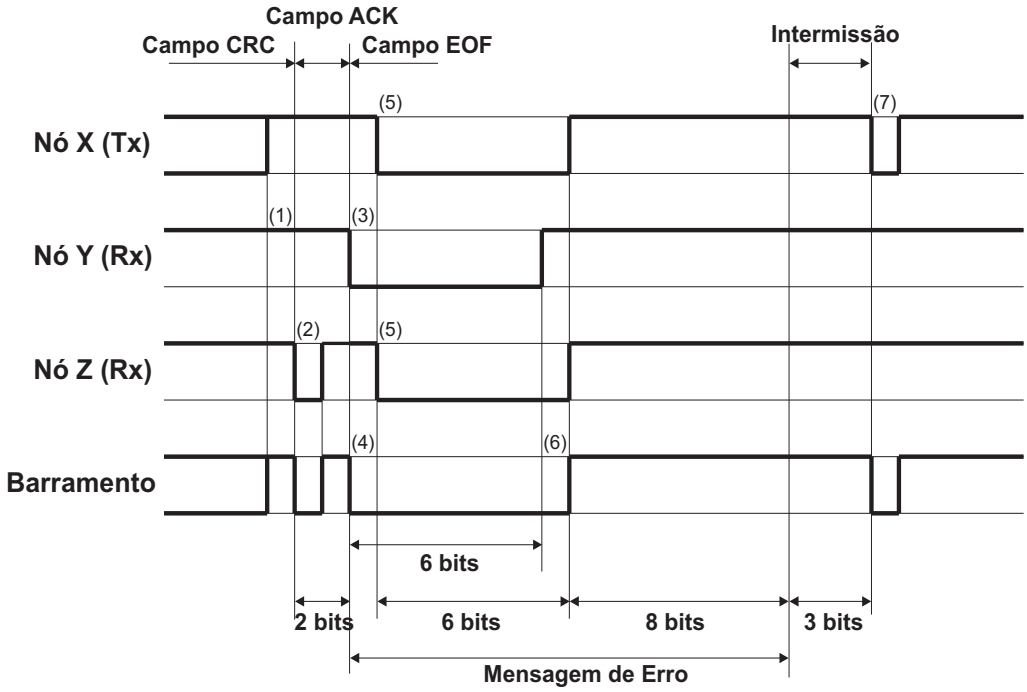

Figura 3.19: Esquema da formação de uma mensagem de erro no caso de um nó ativo detectar um erro local pela verificação da seqüência CRC.

3.7.3 Intervalo entre Mensagens

Mensagens de dados e de requisição remota de dados são separadas de mensagens precedentes (mensagens de dados, requisição, erro ou sobrecarga) por um espaço de tempo, chamado de intervalo entre mensagens.

O intervalo entre mensagens possui dois formatos, e dependem da atual classificação do nó: ativo ou passivo. O formato do intervalo entre mensagens para um nó ativo e para um nó passivo estão representados, respectivamente, nas Figuras 3.20 e 3.21. Ele consiste dos campos: intermissão, barramento livre, e para os nós passivos, o campo suspensão de transmissão.

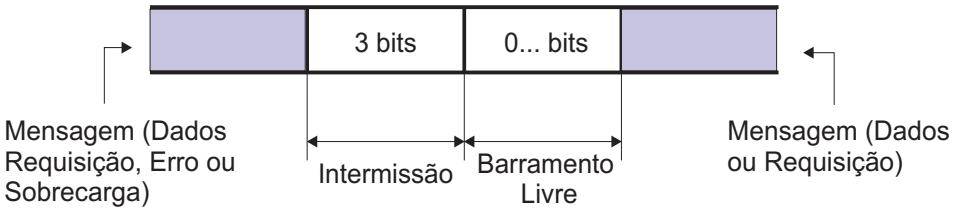

Figura 3.20: Esquema do formato do intervalo entre mensagens de um nó ativo.

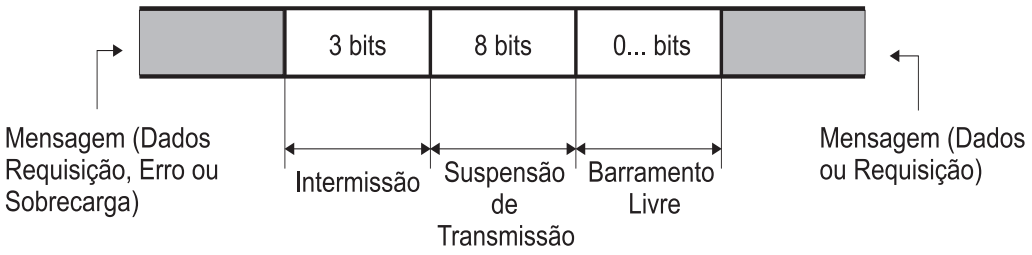

Figura 3.21: Esquema do formato do intervalo entre mensagens de um nó passivo.

O campo intermissão consiste de três *bits* recessivos, os quais representam a distância mínima entre mensagens. O campo barramento livre possui um comprimento arbitrário com os níveis recessivos no barramento, e o campo suspensão de transmissão é constituído por um tempo de inibição de transmissão de oito *bits*.

Os nós passivos estão habilitados a iniciar a transmissão de uma mensagem, somente após o término do campo suspensão de transmissão, e somente se o barramento estiver livre.

3.8 Controle de Acesso ao Meio

Os nós de uma rede CAN acessam o meio de transmissão de acordo com o princípio de alocação do barramento baseado em uma contenção não-centralizada. O acesso ao barramento é permitido somente se ele estiver livre. O barramento é considerado livre por algum nó da rede, apenas se o campo de intermissão não for interrompido por um *bit* dominante (SOF). Durante o barramento livre, o nível lógico lido pelos nós é sempre recessivo.

Devido o acesso ao meio ser realizado de forma aleatória, é possível que vários nós tentem transmitir simultaneamente, gerando um conflito no meio de transmissão. Para controlar o acesso ao meio, o CAN utiliza o método CSMA/CA (*Carrier-Sense Multiple Acess with Collision Avoidance*). Este método garante que durante a fase de arbitração, ou seja, durante a disputa pelo acesso ao barramento, apenas o nó que tiver a mensagem

com maior prioridade, permanecerá transmitindo a mensagem após o final do campo de arbitração.

Na Figura 3.22 representa-se um exemplo do processo de arbitração. Após o campo de intermissão o barramento torna-se livre (nível lógico recessivo), e portanto, pronto para ser acessado pelos nós da rede. Os nós que desejam transmitir uma mensagem enviam o *bit* SOF e em seguida os *bits* do campo de arbitração. Durante a fase de arbitração, todos os nós que estão transmitindo monitoram o atual nível do barramento e comparam com o nível que ele transmitiu. Se um nó enviar um '0' e monitorar um '0', ele continua no processo de arbitração e envia o próximo *bit* do seu identificador. Caso ele envie um '1' e monitore um '0', ele perde a disputa pelo acesso ao meio, pára de transmitir e torna-se apenas um receptor da mensagem, como pode ser observado no *bit* 5 (Nó 2) e no *bit* 2 (Nó 1) do identificador, na Figura. Neste exemplo, o nó 3 venceu o processo de arbitração, visto que ele contém a mensagem com maior prioridade.

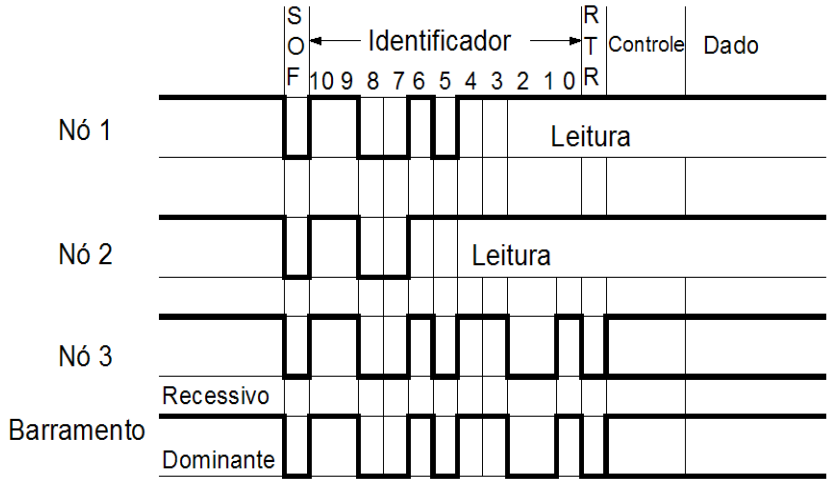

Figura 3.22: Representação de um exemplo do processo de arbitração em uma rede CAN.

Como o princípio de arbitração é baseado em uma comparação do *bit* transmitido com o nível do *bit* monitorado no barramento, o máximo tempo de propagação do sinal entre os nós da rede, assim como os atrasos internos, devem ser levados em consideração para a determinação da amostragem do *bit*. Portanto, o tempo mínimo de transmissão de um *bit* em uma rede CAN, e consequentemente a taxa de transmissão, são limitados por esse mecanismo.

3.9 Mecanismos para Detecção de Erros

Para permitir a globalização de erros locais e a consistência dos dados, o protocolo CAN implementa cinco mecanismos para detecção de erros:

- Verificação do *bit*;
- Verificação do formato;
- Verificação da regra de inserção de *bits*;
- Verificação da seqüência CRC;
- Verificação do reconhecimento.

Após detectar um erro, os seguintes eventos ocorrerão na seguinte ordem:

1. Uma mensagem de erro será enviada pelo nó que detectou o erro;
2. A mensagem com erro será descartada por todos os nós;
3. Os contadores de erro de cada um dos nós será incrementado por um valor específico;
4. A mensagem será retransmitida.

3.9.1 Verificação do *Bit*

Todo nó quando está transmitindo, monitora o barramento para saber se o valor do *bit* transmitido difere do atual valor lido no barramento. Caso ocorra uma diferença, um erro será detectado. Sobrescrever um *bit* recessivo por um *bit* dominante durante a fase de arbitração e durante o *slot-ACK*, não é interpretado como um erro de *bit*. Durante a fase de arbitração, isto indica que uma mensagem com maior prioridade está tentando transmitir uma mensagem, e no *slot-ACK* indica que ao menos um receptor recebeu corretamente a mensagem. Apenas, o *bit* SOF e os campos de controle, dados e CRC são submetidos a este mecanismo.

3.9.2 Verificação do Formato

Existem campos das mensagens CAN que possuem formato fixo (*bits* delimitadores recessivos e campo EOF, ambos fixos em recessivo), os quais são verificados por todos os nós. Um erro de formato será detectado quando um *bit* de um campo submetido a este mecanismo possui um valor diferente do esperado.

3.9.3 Verificação da Regra de Inserção de *Bits*

Um erro de inserção de *bits* ocorre quando um nó detecta uma seqüência de seis *bits* com mesmo nível lógico em um campo de uma mensagem que seja codificado pelo método de inserção (SOF até a seqüência CRC).

3.9.4 Verificação da Seqüência CRC

Para garantir a integridade das informações transmitidas, o protocolo CAN implementa o mecanismo de verificação da seqüência CRC. Este método geralmente é aplicado em sistemas de transmissão de dados para garantir uma alta probabilidade de detecção de erros. Um erro de CRC acontece quando um receptor verifica que a seqüência CRC recebida é diferente da seqüência CRC calculada por ele. Os *bits* a partir do SOF até o campo de dados são submetidos a este mecanismo.

3.9.5 Verificação do Reconhecimento

Após a verificação da seqüência CRC, se a mensagem tiver sido recebida corretamente, o receptor envia um *bit* de reconhecimento com valor dominante, o qual sobrepõe o *bit Slot-ACK*. Uma ausência de reconhecimento (nível recessivo durante o *bit Slot-ACK*) é interpretado pelo transmissor da mensagem como um erro de reconhecimento.

3.10 Confinamento de Falhas

Devido ao princípio de sinalização de erros utilizado no protocolo CAN, existe a possibilidade de um nó defeituoso bloquear toda rede por meio do envio contínuo de um *flag* de erro. Para evitar este problema, um mecanismo de confinamento de falhas foi implementado. Por meio deste, é possível detectar um nó defeituoso e até desconectá-lo do barramento.

O mecanismo de confinamento de falhas consiste em dois contadores de erro, o REC (*Receive Error Counter*) e o TEC (*Transmit Error Counter*), os quais são implementados no controlador CAN de cada nó. Estes contadores são incrementados por um valor específico toda vez que um nó transmitir ou receber uma mensagem com erro, e são decrementados uma vez quando um nó transmitir ou receber uma mensagem sem erro. O valor do incremento dos contadores de um nó depende dele ter sido ou não o primeiro nó a detectar o erro. Se o nó que detectou primeiro o erro for o transmissor da mensagem com erro, o seu TEC será incrementado oito vezes. Caso ele seja o receptor, o REC será incrementado nove vezes. Os nós que detectam o erro depois, terão o REC ou o TEC

incrementados apenas uma vez, caso ele seja um receptor ou o transmissor da mensagem, respectivamente.

Caso um nó detecte primeiro um erro constantemente por um longo período de tempo, pode-se assumir que existe algum defeito nele. Quando limites específicos dos contadores de erro são ultrapassados, medidas são tomadas contra esse nó. Dependendo do valor dos seus contadores de erro, um nó pode estar em um dos três possíveis estados de erro:

- Ativo ao erro;
- Passivo ao erro;
- Desconectado.

Inicialmente, todo nó da rede é classificado como um nó ativo ao erro. Portanto, ele envia um *flag* de erro ativo para sinalizar a ocorrência de um erro. Caso o REC ou o TEC do nó ultrapasse o valor 127, ele será classificado como um nó passivo ao erro. No estado passivo ao erro, o nó ainda é capaz de comunicar-se com os outros nós da rede. No entanto, erros detectados por ele serão agora sinalizados por um *flag* de erro passivo, o qual não sobrepõe a mensagem que está sendo transmitida, exceto no caso dele ser o próprio transmissor da mensagem com erro. Além disso, um nó passivo ao erro deverá esperar por oito tempos de *bit* (campo de suspensão de transmissão) para tentar retransmitir uma mensagem. Ambas as medidas garantem que um nó que tenha detectado erros a uma taxa elevada sobre um longo período de tempo, não atrapalhe a comunicação no barramento. No entanto, é possível que um nó passivo volte a ser um nó ativo ao erro. Para que isso ocorra, ambos os contadores de erro deverão estar abaixo do valor 128.

Quando o TEC de um nó ultrapassar o valor 255, ele será imediatamente desconectado do barramento. Um nó nesse estado não poderá transmitir nenhum tipo de mensagem e nem enviar reconhecimentos. Estando neste estado, o nó poderá retornar ao estado ativo somente após um *reset*, configuração e a detecção de 128 seqüências de 11 *bits* recessivos consecutivos[5]. Na Figura 3.23 representa-se o diagrama de estados de erro de um nó CAN.

[5]Esta seqüência de 11 *bits* recessivos correspondem aos últimos 8 *bits* de uma mensagem de dados ou de requisição de dados (bit delimitador do ACK e campo EOF), mais os 3 *bits* seguintes correspondentes ao intervalo entre mensagens de um nó ativo.

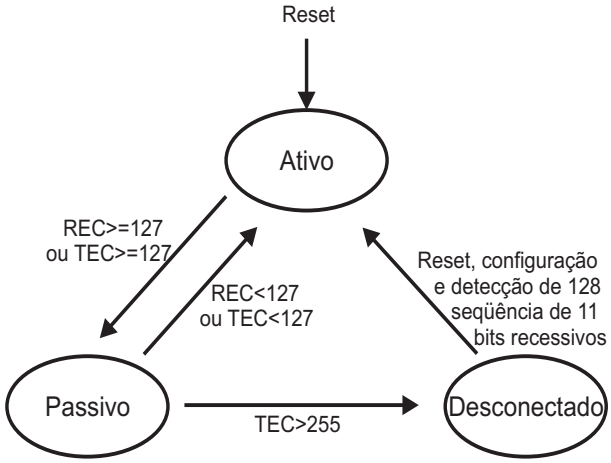

Figura 3.23: Diagrama de estados de erro de um nó CAN.

3.11 Temporização do *Bit*

A transmissão dos dados é realizada de forma síncrona. Isso significa que a capacidade de transmissão é usada com mais eficiência. No entanto, esse método requer um mecanismo sofisticado para sincronização do *bit*.

Em um protocolo de transmissão síncrono existe apenas um *bit* de início no começo da mensagem (*bit* SOF no protocolo CAN). No entanto, isso não é suficiente para saber se a amostragem do *bit* no receptor é suficientemente síncrona com o transmissor. Então, uma contínua re-sincronização do receptor deve ser realizada a cada período de *clock* de um *bit* recebido.

O período de transmissão e recepção de um *bit* em um barramento CAN, na ausência de re-sincronização é chamado de Tempo de *Bit* (τ_{bit}). O tempo de *bit* é dividido em quatro segmentos de tempo:

- Segmento de Sincronização - t_{Sinc};
- Segmento de Propagação - t_{Prop};
- Segmento de Fase 1 - t_{Fase_1};
- Segmento de Fase 2 - t_{Fase_2}.

Além dos segmentos de tempo, existem alguns parâmetros associados ao tempo de *bit*:

- Largura do Pulso de Sincronização - LPS (Em inglês, SJW);
- Ponto de Amostragem;

- Tempo de Processamento da Informação - TPI.

Na Figura 3.24 representa-se o esquema de um tempo de *bit* e seus respectivos segmentos de tempo.

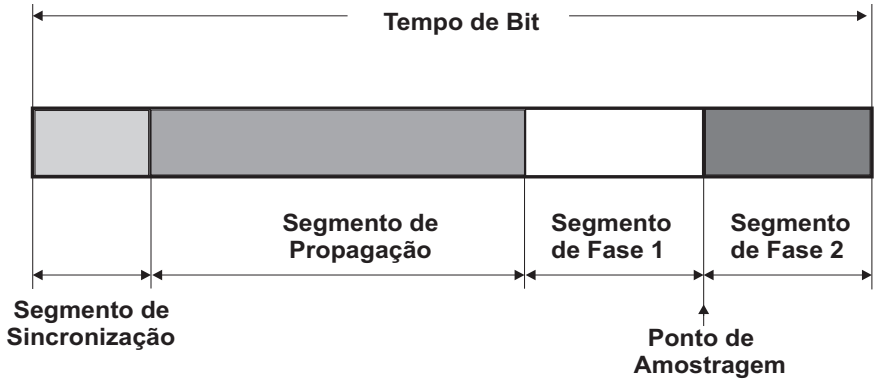

Figura 3.24: Representação do esquema de um tempo de *bit*.

Os comprimentos dos segmentos de tempo são especificados como múltiplos de uma unidade de tempo básica, denominada de *Time Quanta* (*TQ*). O *time quanta* é derivado do período do oscilador do controlador CAN.

O tempo de *bit* deve ter no mínimo uma duração de 8 TQ e no máximo de 25 TQ, e pode ser descrita pela Equação 3.3.

$$\tau_{bit} = t_{Sinc} + t_{Prop} + t_{Fase_1} + t_{Fase_2} \tag{3.3}$$

A taxa de *bit* é definida na especificação CAN, como o número de *bits* por segundo transmitidos por um transmissor ideal sem re-sincronização, descrita pela Equação 3.3.

$$f_{bit} = \frac{1}{\tau_{bit}} \tag{3.4}$$

3.11.1 Segmento de Sincronização

O segmento de sincronização é usado para sincronizar os vários nós do barramento. Espera-se que a borda de um sinal na recepção ocorra neste segmento (caso ideal). Este segmento é fixado em 1 TQ.

A distância entre uma borda que ocorre fora do segmento de sincronização e o início deste segmento, é chamado de erro de fase (e).

3.11.2 Segmento de Propagação

O segmento de propagação é utilizado para compensar atrasos físicos entre os nós. Estes atrasos consistem nos tempos de propagação do sinal no barramento e interno aos nós.

Este segmento pode ser ajustado entre 1 e 8 TQ.

3.11.3 Segmento de Fase 1

O segmento de fase 1 é usado para compensar o erro de fase. Este segmento pode ter o seu comprimento aumentado durante a re-sincronização por LPS pulsos de tempo. O tempo de duração deste segmento pode ser programado entre 1 e 8 TQ.

3.11.4 Segmento de Fase 2

O segmento de fase 2 também é usado para compensar o erro de fase. Ao contrário do segmento de fase 1, esse segmento pode ter o seu comprimento reduzido durante a re-sincronização por LPS pulsos de tempo. O tempo de duração desse segmento pode ser programado entre 2 e 8 TQ.

3.11.5 Largura do Pulso de Sincronização

A largura do pulso de sincronização (LPS) é o tempo usado para compensar um erro de fase. O tempo de duração pode ser programado entre 1 e o $\min(4 * TQ, t_{Fase_1})$.

3.11.6 Ponto de Amostagem

O ponto de amostragem é um ponto do tempo de *bit* em que um *bit* é lido e interpretado. Ele está localizado no fim do segmento de fase 1. Exceto se o controlador estiver configurado para fazer três amostras do sinal. Neste caso, o *bit* é amostrado até o fim do segmento de fase 1. O resultado majoritário das três amostras será interpretado como o nível lógico recebido pelo controlador.

3.11.7 Tempo de Processamento da Informação

O Tempo de Processamento da Informação (TPI) é o tempo gasto para determinar o nível lógico de um *bit*. O TPI começa no ponto de amostragem, e a sua duração depende do controlador, mas não poderá ser maior que 2 TQ.

3.11.8 Dimensionamento dos Segmentos de Tempo

O dimensionamento dos segmentos de temporização do *bit* são determinados pelos requisitos do sistema. Se o comprimento máximo do barramento for requisitado em uma taxa específica ou a máxima taxa de transmissão for requisitada para um determinado comprimento do barramento, os segmentos de fase deverão ser programados com seus valores

mínimos. Com os segmentos de fase programados em seus valores mínimos, apenas erros de fase com $|e| = 1$ poderão ser corrigidos durante a re-sincronização.

Caso a taxa de transmissão e o comprimento do barramento forem baixos, os segmentos de fase poderão ter valores maiores. Deste modo, é possível corrigir erros de fase com $|e| = 4$ durante a re-sincronização.

3.12 Sincronização do *Bit*

Existem dois mecanismos de sincronização para manter um sincronismo entre os nós da rede: sincronização forçada e a re-sincronização. A sincronização forçada é acionada na borda de descida quando o barramento está livre, que é interpretado como o *bit* SOF. Essa sincronização inicia a lógica interna do tempo de *bit*. A re-sincronização é usado para ajustar o tempo de *bit*, devido a erros de fase, enquanto um nó CAN está recebendo uma mensagem.

A sincronização ocorre apenas nas bordas de recessivo para dominante. As bordas são detectadas pela comparação do atual nível do barramento com o nível do barramento no ponto de amostragem anterior. Uma borda é síncrona, se ela for detectada dentro do segmento de sincronização. Caso contrário, existe um erro de fase (e). O erro de fase será positivo se a borda ocorrer depois do segmento de sincronização, e será negativo quando uma borda ocorrer antes do segmento de sincronização.

Quando o erro de fase for positivo (transmissor é lento em relação ao receptor), o tempo de *bit* será aumentado por meio do segmento de fase 1 por LPS pulsos de tempo, como representado na Figura 3.25.

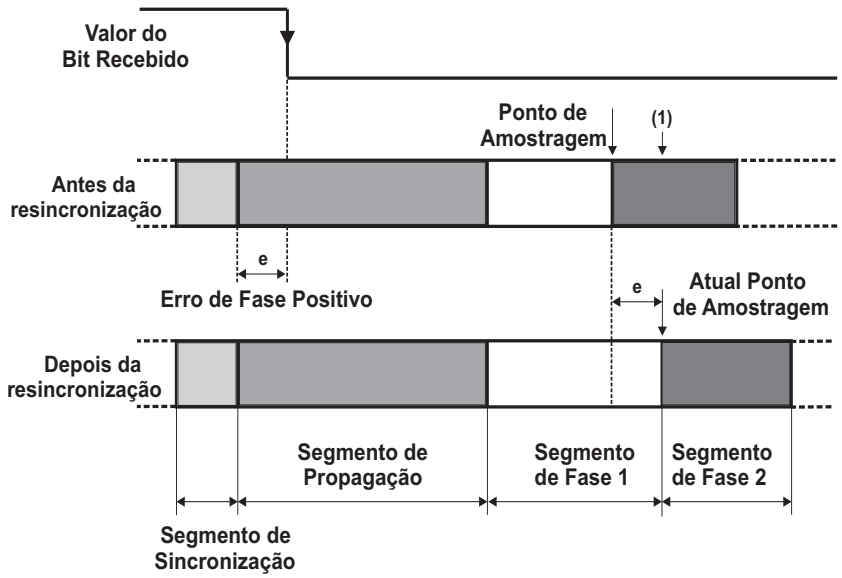

Figura 3.25: Representação do processo de re-sincronização com Erro de Fase Positivo.

Na ausência de re-sincronização a amostragem do *bit* seria realizada no ponto (1) da Figura 3.25.

Quando o erro fase for negativo (transmissor é rápido em relação ao receptor), o tempo de *bit* será diminuido por meio do segmento de fase 2 por LPS pulsos de tempo, como representado na Figura 3.26.

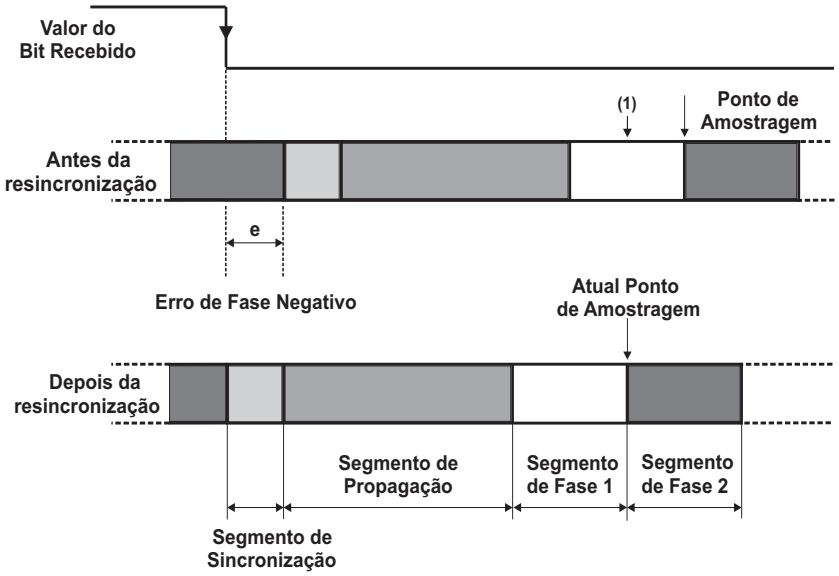

Figura 3.26: Representação do processo de re-sincronização com Erro de Fase Negativo.

Na ausência de re-sincronização a amostragem do *bit* seria realizada no ponto (1) da Figura 3.26.

Se a magnitude do erro de fase for menor ou igual ao valor programado do LPS, o segmento de fase será aumentado ou diminuido pela quantidade exata do erro de fase. Caso a magnitude do erro de fase for maior que LPS, o processo de re-sincronização não poderá compensar o erro de fase completamente. Então um erro (e - LPS) permanecerá.

Os segmentos de fase são alterados temporariamente. No próximo tempo de *bit*, os tempos programados para cada segmento serão restaurados. Apenas uma sincronização é permitida entre dois pontos de amostragem.

Durante a re-sincronização o erro de fase e pode ser reduzido no máximo pelo comprimento de LPS. Caso o erro de fase seja maior que LPS, um erro de fase residual permanecerá. Os erros de fase residuais podem se acumular. Deste modo, é necessário ter osciladores com uma tolerância máxima respeitando os requisitos do sistema. Em geral, utilizam-se osciladores de quartzo para fornecer esses requisitos.

3.13 Taxa de Transmissão x Comprimento do Barramento

O dimensionamento do segmento de propagação define o máximo comprimento possível para uma taxa de dados específica, ou a maior taxa de dados possível para um determinado comprimento do barramento. Na Figura 3.27 representa-se o cenário de pior caso entre os dois nós mais distantes em uma rede[6].

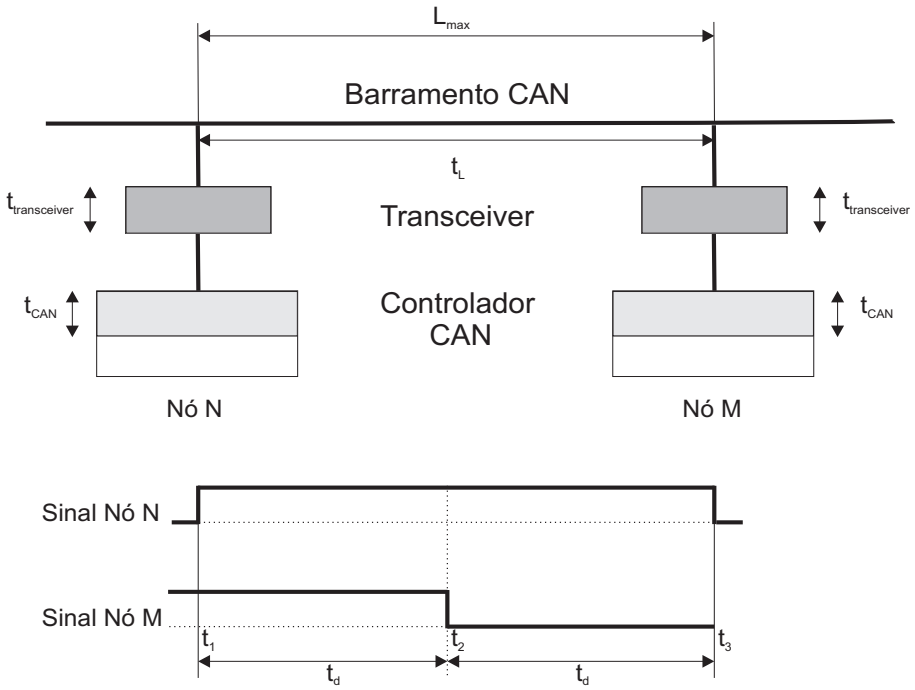

Figura 3.27: Representação do cenário de pior caso para o cálculo do comprimento do segmento de propagação.

Neste caso, observamos o nó N transmitindo um *bit* recessivo no ponto do tempo t_1. O sinal atinge o nó M no ponto do tempo t_2 atrasado pelo tempo de propagação do sinal t_d. O máximo tempo de atraso é definido então pelo tempo de propagação do sinal entre os nós que estão mais distantes. Baseado no tempo de propagação existente, a sincronização do nó M com o nó N é também atrasada por esse tempo, tal que o próprio nó M possa iniciar a transmissão de um novo *bit* no ponto do tempo t_2 (*bit* dominante). Este nível, no entanto não pode ser detectado pelo nó N até o ponto do tempo t_3. Isto significa que até este tempo, o nó N não é capaz de decidir se o nível do sinal que ele transmitiu foi retido ou foi substituído por um *bit* dominante enviado pelo nó M, como apresentado na Figura 3.27. O dimensionamento do segmento de propagação ($t_{SegProp}$) pode ser realizado

[6]O cenário de pior caso ocorre durante a arbitração do barramento ou durante o reconhecimento da mensagem.

da seguinte forma:

$$\begin{cases} t_{SegProp} \geq 2 \times t_d \\ t_d = t_{el} + t_L \\ t_{el} = t_{CAN} + t_{transceiver} \\ t_L = L_{max} \times t_p^* \end{cases} \tag{3.5}$$

Onde:

t_d - Tempo máximo de propagação;
t_{el} - Tempo de atraso do sinal elétrico;
t_L - Tempo gasto para o sinal elétrico chegar ao *transceiver* do nó mais distante;
t_{CAN} - Tempo de atraso do *chip* CAN;
$t_{transceiver}$ - Tempo de atraso do *transceiver*;
L_{max} - Máximo comprimento do barramento;
t_p^* - Constante de propagação específica do barramento.

O máximo comprimento L_{max} de um barramento com um par de fios, conforme indicado na Equação 3.6, é obtido com uma constante de propagação do sinal específico t_p^* de aproximadamente 5 ns/m. A partir da Equação 3.5, temos que $L_{max} = \frac{t_L}{t_p^*}$, onde $t_L = t_d - t_{el}$ e $t_d \leq \frac{t_{SegProp}}{2}$. Para obter L_{max}, assume-se o maior valor possível de t_d, ou seja, $\frac{t_{SegProp}}{2}$.

$$L_{max} = \frac{0.5 \times t_{SegProp} - t_{el}}{5ns/m} \tag{3.6}$$

Os atrasos provocados pelos dispositivos CAN são em torno de 300 ns. Na Tabela 3.2 apresenta-se os valores máximos recomendados do comprimento do barramento e as respectivas taxas de transmissão.

Tabela 3.2: Taxa de transmissão e máximo comprimento do barramento.

Taxa de Transmissão (kbps)	**Máximo Comprimento do Barramento (m)**
1000	40
500	110
250	280
125	620
100	790
50	1640

3.14 Tempo de Transmissão de uma Mensagem de Dados

O tempo (C) gasto para transmitir uma mensagem de dados no formato padrão, indicado na Equação 3.7, é determinado em função do comprimento da mensagem (em *bits*) e do tempo de transmissão de um *bit* (τ_{bit})[7]. O comprimento da mensagem depende do número de *bytes* de dados (b) e da quantidade de *bits* inseridos devido ao mecanismo de inserção de *bits*[8]. Referindo-se a Equação 3.7, o valor $44 + (8 \times b)$ corresponde a todos os *bits* da mensagem, exceto os *bits* que podem ser inseridos pelo mecanismo de inserção de *bits*, que correspondem ao termo entre $\lfloor\ \rfloor$. Observe que apenas os *bits* do SOF até o campo CRC são submetidos ao mecanismo de inserção de *bits*, que equivale ao termo $(34 + (8 \times b) - 1)$.

$$C = \left(44 + (8 \times b) + \left\lfloor \frac{34 + (8 \times b) - 1}{4} \right\rfloor\right) \times \tau_{bit} \tag{3.7}$$

Onde, o termo entre $\lfloor x \rfloor$ é igual ao maior inteiro menor que o valor de x. Na Tabela 3.3 apresenta-se o tempo gasto para transmitir uma mensagem de dados em função do número de *bytes* no campo de dados no pior e melhor caso, considerando $\tau_{bit} = 1\mu s$ (1 Mbps). O pior caso ocorre quando o número máximo de *bits* é inserido, e o melhor caso ocorre quando nenhum *bit* é inserido. Nolte et alli (NOLTE; HANSSON; NORSTRöM, 2002) (NOLTE, 2003) apresentam um método para reduzir o número de *bits* inseridos em uma mensagem.

Tabela 3.3: Tempo de transmissão de uma mensagem de dados.

Número de *Bytes* de Dados (b)	**Tempo no Pior Caso** (μs)	**Tempo no Melhor Caso** (μs)
0	52	44
1	62	52
2	72	60
3	82	68
4	92	76
5	102	84
6	112	92
7	122	100
8	132	108

[7]Ver seção 3.11
[8]Ver seção 3.6

3.15 Tempo de Sinalização de um Erro

O tempo gasto na sinalização de um erro varia entre 14 tempos de *bit* (Melhor caso) e 20 tempos de *bit* (Pior caso). O melhor caso ocorre quando todos os nós detectam um erro no mesmo ponto. Então, cada nó transmitirá os seis *bits* correspondentes ao *flag* de erro, e em seguida, mais oito *bits* correspondentes ao delimitador de erro. O pior caso ocorre quando apenas um nó detecta um erro, e os demais nós detectarão o erro somente no término do *flag* de erro do nó que detectou o erro, resultando em um tempo de recuperação 20 *bits*.

3.16 Implementação de uma Rede CAN

O implementação de uma rede CAN envolve a escolha de vários componentes de *hardware*, o desenvolvimento de *software* para configurar e controlar os dispositivos, e executar as atividades específicas de cada nó de acordo com a aplicação. As seguintes tarefas devem ser realizadas pelo projetista do sistema:

- Escolha da arquitetura e do controlador CAN, e consequentemente, os dispositivos que formam um nó. A decisão de qual microcontrolador será utilizado é fundamental para a aplicação;
- Atribuição de identificadores fixos e únicos para as mensagens enviadas pelos nós. Esta decisão deve ser tomada de acordo com as informações transmitidas por cada mensagem. Deve-se atribuir altas prioridades para as mensagens com maior importância no sistema, como por exemplo, mensagens com altos requisitos de tempo real;
- Definição e configuração das máscaras e filtros de aceitação conforme a relevância das mensagens para cada nó. A utilização correta das máscaras pode reduzir o tempo de processamento interno ao controlador CAN no processo de recepção;
- Configurar corretamente os *buffers* de transmissão, de acordo com o tipo e formato da mensagem, o identificador, a quantidade de *bytes* que serão transmitidos na mensagem, e o nível de prioridade do *buffer* com relação aos demais *buffers* do mesmo controlador. Neste caso, deve-se tomar como referência a ordem de prioridade atribuída pelo identificador da mensagem;
- Configuração dos parâmetros do tempo de *bit* conforme a aplicação. A escolha do tipo, freqüência e tolerância dos cristais conectados aos controladores são importantes para garantir a comunicação entre os nós da rede;

- Desenvolvimento de um *software* embarcado em cada nó capaz de realizar as tarefas dos itens anteriores. O *software* deve ser otimizado para garantir as restrições de tempo. Deve existir um gerenciamento correto dos serviços de interrupções de cada um dos periféricos, assim como, as prioridades das interrupções.

3.17 Conclusões

Neste Capítulo foram apresentadas as principais características, funcionamento e formas de implementação de uma rede CAN. O fato deste protocolo ter sido desenvolvido para aplicações automotivas, justifica a sua escolha como protocolo a ser utilizado no sistema de aquisição de dados proposto neste trabalho. Além disso, CAN possui uma elevada taxa de transmissão, eficientes mecanismo para detecção e sinalização de erros, disponibilidade de *hardware* de baixo custo no mercado, e a possibilidade de implementar dois conceitos de comunicações, os quais foram desenvolvidos neste trabalho utilizando um único *chip*.

Capítulo 4

Protocolo Time-Triggered CAN

4.1 Introdução

CAN é um protocolo *event-triggered* baseado no mecanismo de controle de acesso ao meio CSMA/CA. Neste mecanismo, um nós pode transmitir uma mensagem somente se o barramento estiver livre. No entanto, é possível que mais de um nó tente transmitir uma mensagem ao mesmo tempo, conforme apresentado na seção 3.8. O mecanismo de arbitração garante que a mensagem com maior prioridade que estiver na disputa pelo acesso ao meio, sempre vencerá o processo de arbitração. As mensagens com menores prioridades serão automaticamente re-escalonadas assim que o barramento estiver livre. Portanto, as mensagens podem chegar aos seus destinos com atrasos, os quais são variáveis (*Jitter*). É possível que até a mensagem com maior prioridade da rede sofra um atraso. Este atraso ocorrerá quando uma mensagem estiver ocupando o barramento durante o seu escalonamento.

Estes atrasos podem comprometer o desempenho do sistema em aplicações que necessitam de um comportamento estritamente determinístico no tempo, tais como, aplicações de controle distribuído em malha fechada e aquisição de dados com altos requisitos de tempo real, onde normalmente os eventos ocorrem periodicamente. Nestas aplicações, sistemas de comunicações baseados no paradigma *time-triggered* são mais convenientes, pois neste tipo de comunicação as mensagens são transmitidas em janelas de tempo exclusivas, ou seja, as mensagens não disputam o acesso ao meio de transmissão. Por outro lado, existem aplicações em que os eventos ocorrem de forma periódica e esporádica. Nestas aplicações, o sistema de comunicação deverá ser capaz de satisfazer os altos requisitos de tempo real e ao mesmo tempo ser flexível, de modo a satisfazer os eventos que ocorrem esporadicamente.

Para resolver estes problemas, um protocolo baseado no CAN, designado por TTCAN - *Time-Triggered CAN*, foi especificado recentemente na ISO 11898-4 (ISO, 2004). TTCAN,

além de utilizar a especificação CAN, implementa a camada de sessão do modelo OSI, a qual controla a transmissão de mensagens sincronizadas no tempo de acordo com um escalonamento estático pré-definido, semelhante ao esquema TDMA (*Time Division Multiple Access*). A especificação TTCAN pode ser implementada em dois níveis. O nível 1 é restrito apenas a transmissão cíclica das mensagens. O nível 2 é uma extensão do nível 1, o qual implementa um tempo global altamente sincronizado. Neste nível, um mecanismo de sincronização é implementado para compensar as variações de tempo provocadas pelos diferentes cristais dos nós.

A principal vantagem do TTCAN é a possibilidade de combinar os paradigmas de comunicações *event-triggered* e *time-triggered* (METZNER; STIERAND, 2006), e implementar mecanismos de tolerância à falhas (MüLLER et al., 2002). Além disso, este protocolo evita variações de atrasos nas mensagens, garante uma comunicação padrão e determinística no barramento e utiliza com mais eficiência a largura de banda da rede (FüHRER et al., 2000) (LEEN; HEFFERNAN., 2002).

4.2 Comunicação TTCAN

A comunicação TTCAN é baseada na transmissão periódica de uma mensagem de referência por um nó da rede chamado de mestre do tempo. Todos os nós da rede são sincronizados durante a recepção desta mensagem. Cada período iniciado por uma mensagem de referência é chamado de ciclo básico. O tempo de duração de um ciclo básico é chamado de tempo do ciclo. O ciclo básico é constituído por várias janelas de tempo, onde as mensagens de dados serão transmitidas. Existem três tipos de janelas de tempo: exclusiva, arbitração e livre.

Janelas exclusivas são usadas para a transmissão de mensagens periódicas, as quais são transmitidas sem competir pelo acesso ao barramento. Apenas um nó da rede pode iniciar uma transmissão em uma janela exclusiva.

Janelas de arbitração são usadas para a transmissão de mensagens esporádicas. Várias mensagens esporádicas poderão compartilhar a mesma janela de tempo. Caso mais de uma mensagem esporádica tente transmitir ao mesmo tempo dentro de uma janela de arbitração, o conflito gerado pelo acesso simultâneo será solucionado de acordo com o processo de arbitração do protocolo CAN. No entanto, o mecanismo de re-transmissão automática das mensagens é desabilitado[1]. Estas mensagens poderão ser transmitidas em outras janelas de arbitração.

Janelas livres podem ser utilizadas para a transmissão de mensagens de erro, ou reservadas para futuras extensões da rede. Na Figura 4.1 representa-se um esquema de um

[1]O mecanismo de re-transmissão automática é habilitado somente para mensagens de referência.

ciclo básico com um exemplo de alocação das janelas.

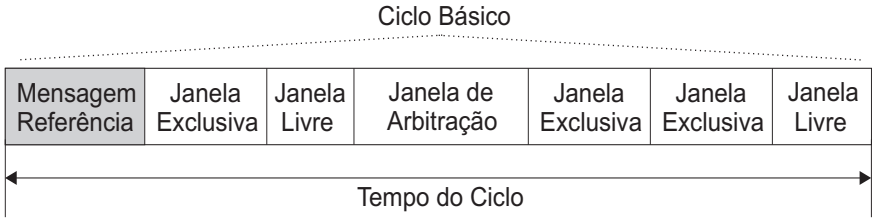

Figura 4.1: Representação do esquema de um ciclo básico.

Desde que o ciclo básico não fornece flexibilidade suficiente para especificar os diferentes períodos de transmissão de um sistema, a especificação TTCAN permite que vários ciclos básicos sejam conectados em série, formando o chamado ciclo matriz. Todas as mensagens de todos os nós de uma rede são organizadas nas janelas de tempo do ciclo matriz. Cada ciclo básico de um ciclo matriz consiste de uma seqüência de janelas de tempo com mesmo comprimento, começando sempre pela mensagem de referência. A especificação TTCAN permite que um ciclo matriz tenha no máximo 64 ciclos básicos. Na Figura 4.2 representa-se um ciclo matriz com quatro ciclos básicos. Uma mensagem de referência é transmitida pelo mestre do tempo assim que o ciclo básico for finalizado, iniciando desta forma um novo ciclo básico. Após o término do último ciclo básico, o ciclo matriz é re-inicializado.

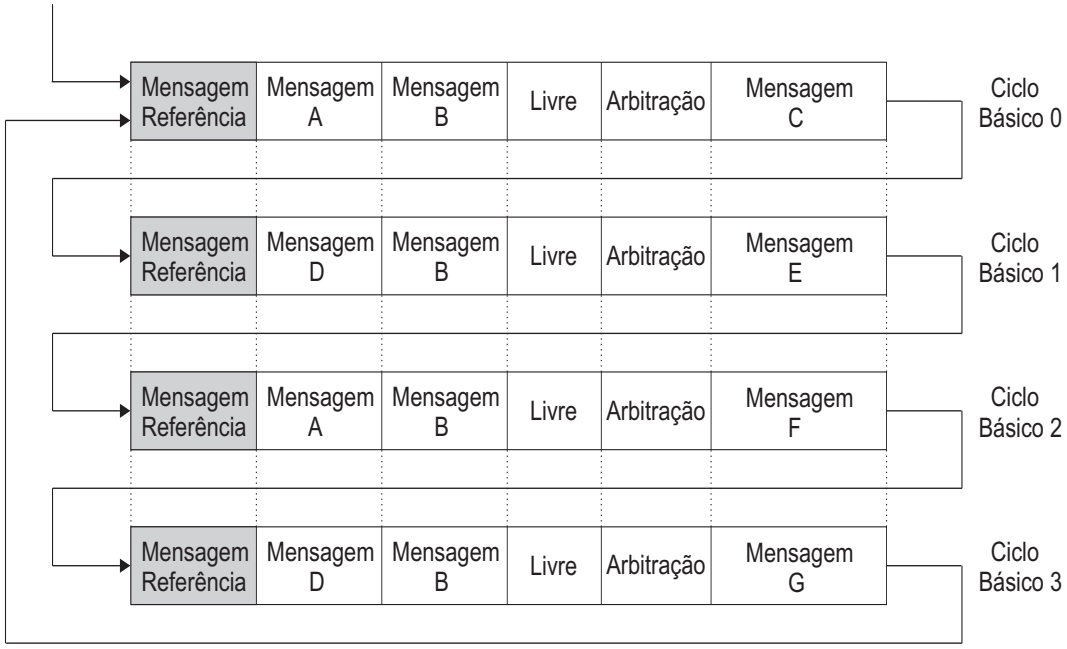

Figura 4.2: Representação do esquema de um ciclo matriz.

O número de ciclos básicos, o comprimento e o tipo das janelas, e as respectivas men-

sagens atribuídas a cada uma dessas janelas, devem ser especificados durante a fase de projeto do sistema. Para realizar este projeto é necessário saber informações sobre o período de transmissão e o comprimento das mensagens. Assim, pode-se definir uma estratégia de escalonamento ótimo das mensagens. Albert (ALBERT; HUGEL, 2005) apresenta alguns conceitos de escalonamento heurístico utilizados na construção do ciclo matriz. Normalmente, o período do ciclo matriz é igual ao maior período de uma mensagem. Enquanto que o período do ciclo básico é igual ao menor período de uma mensagem.

4.2.1 Mensagem de Referência

As mensagens de referência são utilizadas para sincronizar e calibrar as bases de tempo de todos os nós, de acordo com a base de tempo do mestre do tempo, fornecendo um tempo global para a rede.

Uma mensagem de referência é reconhecida pelos nó da rede pelo seu identificador, o qual deve ter um valor específicado durante a fase do projeto. No TTCAN nível 1, a mensagem de referência possui apenas 1 *byte* de dados, que fornece informações sobre número do atual ciclo básico. No TTCAN nível 2, a mensagem de referência possui quatro *bytes* de dados, que também fornecem as mesmas informações de controle do nível 1 e informações sobre a base de tempo do nó mestre, que será utilizado para garantir uma base de tempo global altamente sincronizada (MüLLER et al., 2002).

Normalmente, a mensagem de referência é transmitida em intervalos de tempo eqüidistantes. Opcionalmente, é possível que a mensagem de referência seja gatilhada por um evento específico, o qual é determinado pelas informações do campo de dados da mensagem de referência transmitida no ciclo básico anterior.

4.2.2 Mestre do Tempo

Desde que, a presença do mestre do tempo é fundamental para a comunicação TTCAN, a especificação fornece mecanismos de tolerância à falhas por meio da redundância do mestre. É possível ter no máximo oito nós mestres do tempo. Cada mestre transmite uma mensagem de referência com um identificador específico, onde os oito primeiros *bits* identificam a mensagem de referência e os três *bits* restantes identificam o nível de prioridade dos oito possíveis mestres do tempo.

4.2.3 Marco do Tempo

A seqüência de janelas de tempo do ciclo básico são controladas pelos Marcos do Tempo, os quais são comparados com o atual valor do tempo do ciclo. Uma determinada janela de tempo é "aberta"quando um correspondente marco do tempo ocorrer, e será "fechada"pouco

antes da próxima janela ser "aberta". Um marco do tempo especifica o início das janelas exclusiva e arbitração. Existem dois marcos do tempo: *Tx-Triggers* e *Rx-Triggers*.

Os *tx-triggers* indicam o instante em que mensagens periódicas ou esporádicas serão transmitidas. Já os *rx-triggers* indicam o instante em que as mensagens serão recebidas.

4.3 Bases de Tempo

Cada nó tem sua própria base de tempo, chamado de tempo local, o qual é implementado por um relógio que é incrementado pela NTU (Unidade de Tempo da Rede) (HARTWICH et al., 2002). O tempo local de cada nó é sincronizado no ponto de amostragem do *bit* SOF (*Start Of Frame*) das mensagens de referência. O comprimento da NTU é definido durante o projeto da rede, e deve ser igual para todos os nós. A NTU é baseada no período do relógio de cada nó e em um valor numérico chamado de TUR (Razão da Unidade de Tempo Local), conforme indicado pela Equação 4.1. Onde, t_N é o período do relógio do nó N derivado do respectivo cristal.

$$NTU = t_N \times TUR \tag{4.1}$$

No TTCAN nível 1, TUR é um valor constante e o tempo local é informado por um relógio com 16 *bits* incrementado uma vez a cada NTU. Portanto, as informações do tempo local são números inteiros com resolução igual ao NTU.

No TTCAN nível 2, o tempo local é informado por um relógio com 16 *bits* estendido por uma parte fracionária com N *bits* (No máximo três *bits*). O tempo local é incrementado 2^N vezes a cada NTU, fornecendo uma maior resolução do tempo comparado ao nível 1. Neste nível, o TUR é um valor não-inteiro variável, que é utilizado para compensar variações dos diferentes cristais de cada nó, garantindo uma maior precisão do tempo entre os nós.

4.3.1 Tempo do Ciclo

A sincronização entre os nós é mantida pela transmissão periódica da mensagem de referência, que é reconhecida sincronamente por todos os nós[2]. Cada mensagem de referência válida, inicia um novo ciclo básico e causa uma re-inicialização do tempo do ciclo de cada nó. O valor do tempo local capturado no ponto de amostragem do *bit* SOF da mensagem de referência, é chamado de Marco do Sincronismo (MS). Quando a mensagem de referência for reconhecida como uma mensagem de referência válida, o MS desta mensagem torna-se o novo Marco de Referência (MR). O tempo do ciclo é igual a atual diferença

[2]Desconsiderando o tempo de propagação do sinal

entre o tempo local e o MR, o qual é re-iniciado no começo de cada ciclo básico quando o MR for carregado, conforme representado na Figura 4.3.

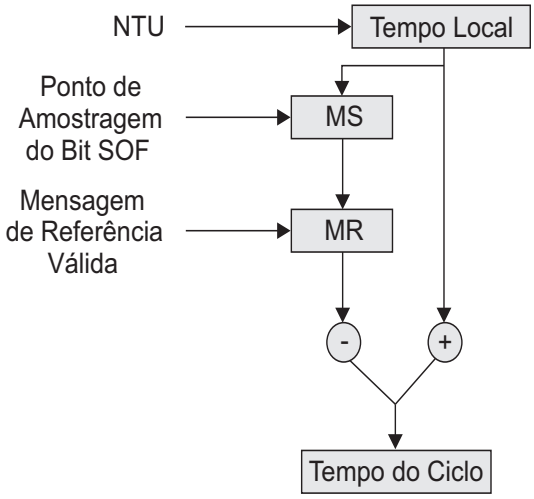

Figura 4.3: Representação do esquema de geração do tempo de ciclo.

4.3.2 Tempo Global

A noção de tempo global depende do nível de implementação do TTCAN. No nível 1, a base de tempo comum para todos os nós da rede é o tempo do ciclo, que é re-iniciado no começo de cada ciclo básico, e é baseado no tempo local de cada nó, conforme apresentado na seção 4.3.1. Neste nível, o tempo local pode ser visto como o tempo global da rede.

No nivel 2, existe um tempo global que é utilizado como a referência para calibração de todas as bases de tempo local. Este tempo global é derivado do tempo local do mestre do tempo. Após a fase de inicialização, um possível mestre do tempo transmite uma mensagem de referência, e todos os nós inclusive ele, capturam o seu tempo local no MS (Marco de Sincronismo), e assim que a mensagem for considerada válida, o MS torna-se o MR (Marco de Referência). No caso do mestre do tempo, este MR é chamado de MRM (Marco de Referência do Mestre). Em seguida, o mestre do tempo envia novamente uma mensagem de referência contendo no campo de dados o valor do MRM capturado na mensagem de referência anterior. Todos os nós inclusive o mestre do tempo, realizam o mesmo procedimento, sendo que eles retiram desta mensagem de referência o MRM do mestre. Então, cada nó calcula a diferença entre o seu MR e o MRM, da mensagem de referência anterior, onde o resultado é chamado de *Offset Local* (OL). Observe que o OL do atual mestre será zero. No entanto, se outro nó assumir o posto de mestre do tempo, o valor do seu OL será diferente de zero. Para todos os nós, o tempo global é a soma entre o seu tempo local e o *offset* local calculado, como representado na Figura 4.4.

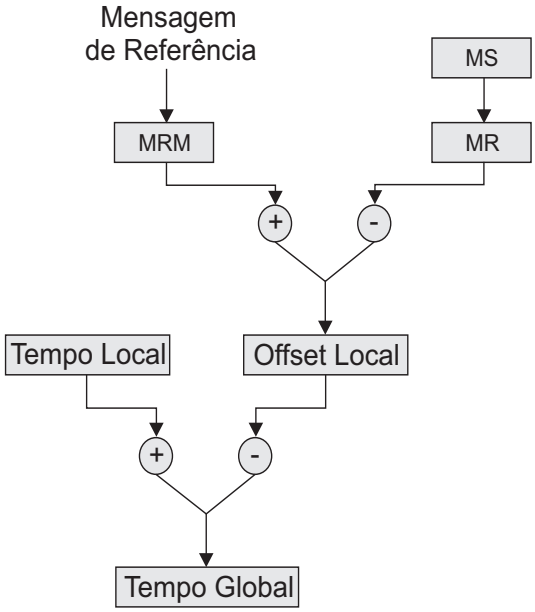

Figura 4.4: Representação do esquema de geração do tempo global.

As variações entre o tempo global e o tempo local de cada nó são compensadas na recepção das mensagens de referência pela atualização do OL. Alterações do OL mostram as diferenças entre o NTU dos nós e o NTU do atual mestre. Isto ocorre devido as variações físicas entre os cristais, resultando em pequenas diferenças entre o termo t_N de cada nó, conforme indicado na Equação 4.1. A atual diferença de velocidade do relógio é calculada pela divisão entre dois consecutivos MRM e dois consecutivos MR. A variação da velocidade do relógio é compensada pela atualização do TUR que gera o NTU local de cada nó.

O fator df pelo qual o TUR do nó deve ser ajustado é calculado de acordo com a Equação 4.2.

$$df = \frac{MR_{atual} - MR_{anterior}}{MRM_{atual} - MRM_{anterior}} \tag{4.2}$$

O novo TUR é determinado pela Equação 4.3.

$$TUR = df \times TUR_{anterior} \tag{4.3}$$

Na Figura 4.5 representa-se um diagrama com o esquema de compensação da variação dos relógios dos nós. Antes da sincronização da rede, cada nó atribui como tempo global o seu próprio tempo local, e assume como *offset* local o valor zero.

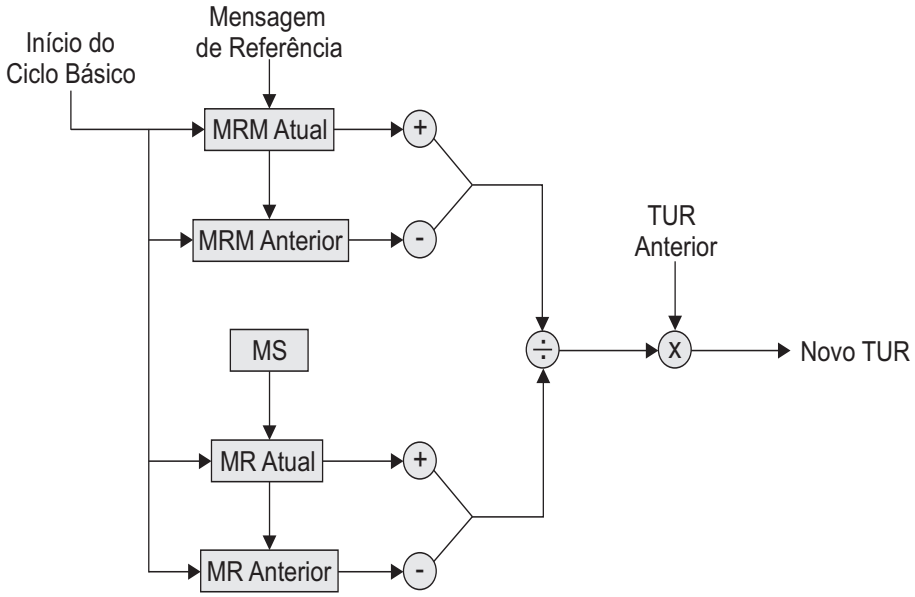

Figura 4.5: Representação do esquema de compensação da variação dos relógios.

4.4 Implementação de uma Rede TTCAN

Uma rede TTCAN é composta pelos mesmos componentes utilizados na implementação de uma rede CAN, conforme apresentado no Capítulo 3. A principal e única diferença é que os controladores TTCAN executam um protocolo de sincronização e calibração dos relógios, que permitem a comunicação *time-triggered*, como descrito nas seções anteriores deste Capítulo. Nós TTCAN são completamente compatíveis com nós CAN. Portanto, nós TTCAN podem utilizar o mesmo barramento e o mesmo *transceiver* CAN. Os controladores CAN disponíveis no mercado podem receber mensagens enviadas por um nó TTCAN, assim como, controladores TTCAN podem operar em redes CAN (HARTWICH et al., 2002) (ALBERT; STRASSER; TRäCHTLER, 2003).

Atualmente, a Bosch GmbH está desenvolvendo um controlador TTCAN, chamado de TTCAN *testchip*, o qual é disponivel por um preço de 600 euros. No entanto, este *chip* é considerado uma amostra não qualificada, intencionada apenas para propósitos de desenvolvimento (BOSCH, 2007). O TTCAN *testechip* é um controlador com arquitetura *stand-alone* com 44 pinos, que implementa os dois níveis TTCAN por *hardware*, ou seja, toda comunicação *time-triggered* não depende de um *software* de controle. É necessário apenas que o *firmware* do microcontrolador configure-o corretamente por meio de uma *interface* externa, de acordo com a aplicação. A configuração envolve o ajuste de registradores que indicam a base de tempo, os identificadores dos mestre do tempo, os marcos de tempo para as respectivas mensagens entre outros.

Apesar de não existir um controlador TTCAN qualificado para propósitos de aplicação, alguns controladores CAN disponíveis recentemente no mercado, possuem os requisitos básicos para implementar uma versão por *software* do TTCAN nível 1. Um exemplo destes controladores é o MCP2515 (MICROCHIP, 2005). Este controlador dispõe das seguintes funções em *hardware*:

- Capacidade de gerar um evento de sinalização (um pulso) em um dos seus pinos, no ponto de amostragem do *bit* SOF;
- Capacidade de desabilitar o mecanismo de re-transmissão automática (Modo *one-shot*).

Baseado nestas funções, é possível desenvolver uma estratégia de sincronização por *software* dos relógios de cada nó da rede e fornecer uma comunicação *time-triggered* com uma precisão na ordem de μs. Aproveitando-se destas características, foi proposto neste trabalho o desenvolvimento de uma rede TTCAN, onde os nós são baseados no controlador CAN *stand-alone* MCP2515, que implementa uma comunicação *time-triggered* de acordo com algumas características do TTCAN nível 1. Na seção seguinte descreve-se a solução proposta que será utilizada no sistema de aquisição de dados distribuído.

4.5 Comunicação TTCAN Proposta

A comunicação TTCAN desenvolvida neste trabalho é baseada na especificação TTCAN nível 1 com pequenas diferenças, as quais são citadas a seguir:

- Existe apenas um nó mestre na rede;
- A mensagem de referência possui 0 *bytes* de dados;
- O mecanismo de re-transmissão automática é desabilitado em todos os nós, inclusive no mestre do tempo.
- Não existe um registrador que capture o atual valor de tempo do temporizador (relógio) que marca o tempo do ciclo;
- Não existem *Rx-Triggers*;
- Não existe limitação quanto ao número de ciclos básicos por ciclo matriz. A quantidade mínima de ciclos básicos necessários para alocar todas as mensagens da rede é igual a divisão entre o maior período de uma mensagem e o tempo de duração de um ciclo básico (menor período de uma mensagem da rede)[3].

[3]O período de uma mensagem é o tempo gasto entre duas subseqüentes transmissões desta mensagem.

O microcontrolador utilizado no nó TTCAN deve ter os seguintes requisitos de *hardware*:

- Duas interrupções externa;
- Dois temporizadores (16 *bits*) com interrupções, por exemplo, os temporizadores[4] 0 e 1 do microcontrolador 80C51 (NICOLOSI, 2004);
- *Interface* SPI.

O pino do MCP2515 que gera um pulso no ponto de amostragem do *bit* SOF, deve ser conectado ao pino de interrupção externa do microcontrolador (INT1/SOF), de acordo com a Figura 4.6. Este evento gera a sincronização dos nós da rede, os quais executam uma simples rotina para acionar os relógios, escalonar as mensagens nas respectivas janelas de tempo do ciclo básico equivalente, e incrementar uma variável que indica o atual ciclo básico. A interrupção externa INT0 é utilizada para indicar ao microcontrolador que o MCP2515 recebeu uma mensagem válida. Como pode ser observado na Figura 4.6, a comunicação entre o MCP2515 e o microcontrolador é realizada por meio da *interface* SPI, a qual será explicada na seção 5.3.1. A utilização do MCP2515 impõe a restrição do uso da *interface* SPI.

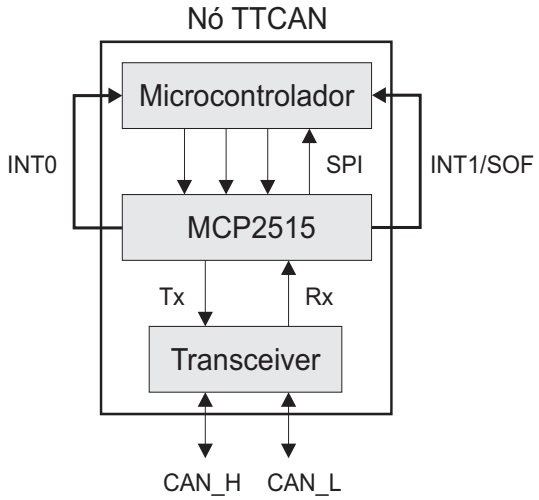

Figura 4.6: Representação do diagrama de blocos do nó TTCAN proposto.

4.5.1 Ciclo do Tempo e Marco do Tempo

Cada nó possui dois relógios, os quais são utilizados para gerar o ciclo do tempo e o marco do tempo *tx-trigger*.

[4]A partir deste ponto, o temporizadores 0 e 1 serão chamados de relógios 0 e 1

No nó mestre do tempo, o relógio 0 é utilizado para marcar o tempo do ciclo (t_{Cm}) conforme indicado na Equação 4.4. Onde: t_{Rm} é o período de ciclo dos relógios (0 e 1) do mestre derivado da CPU, e q_{Cm0} é a quantidade de ciclos do relógio 0 do nó mestre. O valor máximo de q_{Cm0} é 65536 (2^{16}).

$$t_{Cm} = t_{Rm} \times q_{Cm0} \tag{4.4}$$

Este relógio pode ser visto como o relógio de tempo global da rede. Uma mensagem de referência sempre será enviada na interrupção de *overflow* deste relógio, a qual re-inicia um ciclo básico e incrementa a variável que indica qual o atual ciclo básico do ciclo matriz. Caso o ciclo matriz tenha finalizado, a variável de contagem do ciclo básico será zerada. O relógio 1 é utilizado para gerar o evento de transmissão da mensagem de dados (Marco do tempo *Tx-Trigger*) do nó mestre. O *Tx-Trigger* do mestre (t_{Tm}) é determinado pela Equação 4.5. Onde: q_{Cm1} é a quantidade de ciclos do relógio 1 do nó mestre. O valor máximo de q_{Cm1} é 65536 (2^{16}).

$$t_{Tm} = t_{Rm} \times q_{Cm1} \tag{4.5}$$

Os nós restantes (Escravos do tempo) também utilizam os dois relógios e com as mesmas funções do nó mestre. Exceto que uma mensagem de referência não será transmitida na interrupção de *overflow* do relógio 0. O relógio 0 é utilizado apenas para indicar que o ciclo básico foi finalizado. As Equações 4.6 e 4.7 determinam, respectivamente, o tempo do ciclo (t_{CeN}) e o *Tx-Trigger* (t_{TeN}) do escravo N. t_{ReN} é o período de ciclo dos relógios (0 e 1) do escravo N, q_{CeN0} e q_{CeN1} são as quantidades de ciclos dos relógios 0 e 1 do escravo N, respectivamente.

$$t_{CeN} = t_{ReN} \times q_{CeN0} \tag{4.6}$$

$$t_{TeN} = t_{ReN} \times q_{CeN1} \tag{4.7}$$

4.5.2 Variações entre os Relógios

Pode-se considerar que os relógios (0 e 1) de cada nó não possuem variações, visto que existe um único cristal que fornece o período de ciclo destes relógios, t_{Rm} no nó mestre e t_{ReN} nos N nós escravos. Na Figura 4.7 representa-se um esquema de geração dos tempo de ciclo e marco do tempo *tx-trigger* do nó mestre. Um esquema semelhante é utilizado para gerar esses tempos nos nós escravos do tempo.

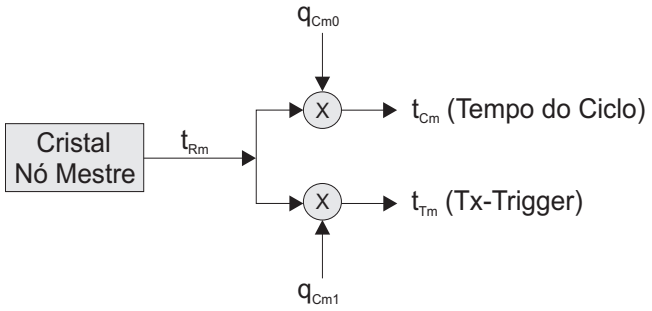

Figura 4.7: Representação do esquema de geração dos tempo de ciclo e *tx-trigger* no nó mestre.

Por outro lado, existe uma variação entre os relógios de cada nó, visto que cada nó possui o seu próprio cristal. A maioria dos métodos de sincronização baseados no protocolo CAN, (GERGELEIT; STEICH, 1994), (TURSKI, 1994), (RODRIGUES; GUIMARãES; RUFINO, 1998) e (LINDQVIST, 2003) utilizam algum algoritmo para compensação das variações dos diferentes cristais dos nós, por meio de equações lineares baseadas no relógio do mestre da rede. Estes cristais variam entre 10^{-6} e 10^{-5} em torno da freqüência nominal, ou seja, um relógio baseado nesses cristais variam na ordem de microsegundos à dezenas de microsegundos a cada segundo (LINDQVIST, 2003). Para minimizar este problema, utilizamos um ciclo básico com período na ordem de centenas de microsegundos, o que proporcionou uma variação entre os diferentes relógios abaixo de 1 μs.

De modo a evitar que o tempo do ciclo (t_{CeN}) dos nós escravos do tempo terminem depois do tempo do ciclo do mestre do tempo, deve-se programar q_{CeN0} com um valor pouco menor que q_{Cm0}.

Para sincronizar os nós, o mestre transmite no início de todo ciclo básico a mensagem de referência. De acordo com o método proposto, a única informação útil desta mensagem é o primeiro *bit* da mensagem, *bit* SOF. As chances de ocorrer um erro neste *bit* são muito pequenas, justificando a ausência de redundância do nó mestre.

4.5.3 Algoritmo de Execução dos Nós TTCAN

Um fluxograma simplificado do algoritmo do nó mestre e dos nós escravos estão representados, respectivamente, nas Figuras 4.8 e 4.9. O algoritmo de execução do nó mestre é formado por quatro interrupções (INT0, INT1, Relógio 0 e Relógio 1), e nos nós escravos por três interrupções (INT1, Relógio 0 e Relógio 1).

A interrupção INT0 ocorre somente na recepção da mensagem de inicialização da rede, que é enviada pelo nó supervisor da aplicação. Devido esta interrupção ocorrer somente uma vez e no início do funcionamento do sistema, atribuiu-se a ela a menor prioridade.

A interrupção INT1 indica que o *bit* SOF de uma mensagem foi recebida. Esta interrupção é fundamental no processo de comunicação. Portanto, atribui-se a ela a maior prioridade entre as interrupções.

As interrupções dos relógios 0 e 1 ocorrem no *overflow* destes relógios, conforme a configuração realizada durante o projeto do sistema. A interrupção do relógio 1 ocorrerá sempre primeiro, pois este relógio é utilizado para gerar o marco do tempo *Tx-Trigger* do ciclo básico. Esta interrupção possui a segunda maior prioridade, enquanto que, a interrupção do relógio 0 possui a terceira maior prioridade.

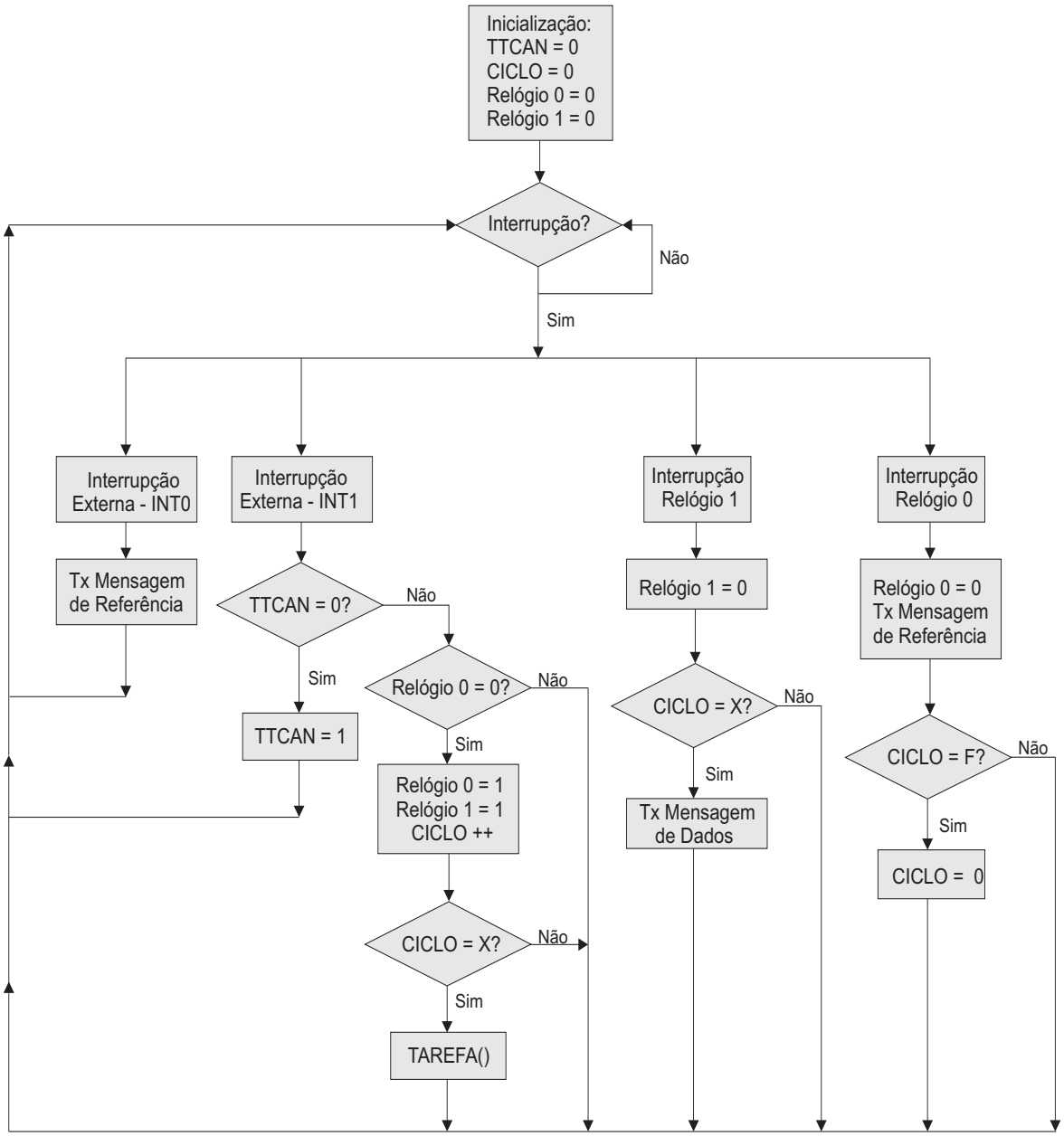

Figura 4.8: Fluxograma simplificado do algoritmo do nó mestre do tempo.

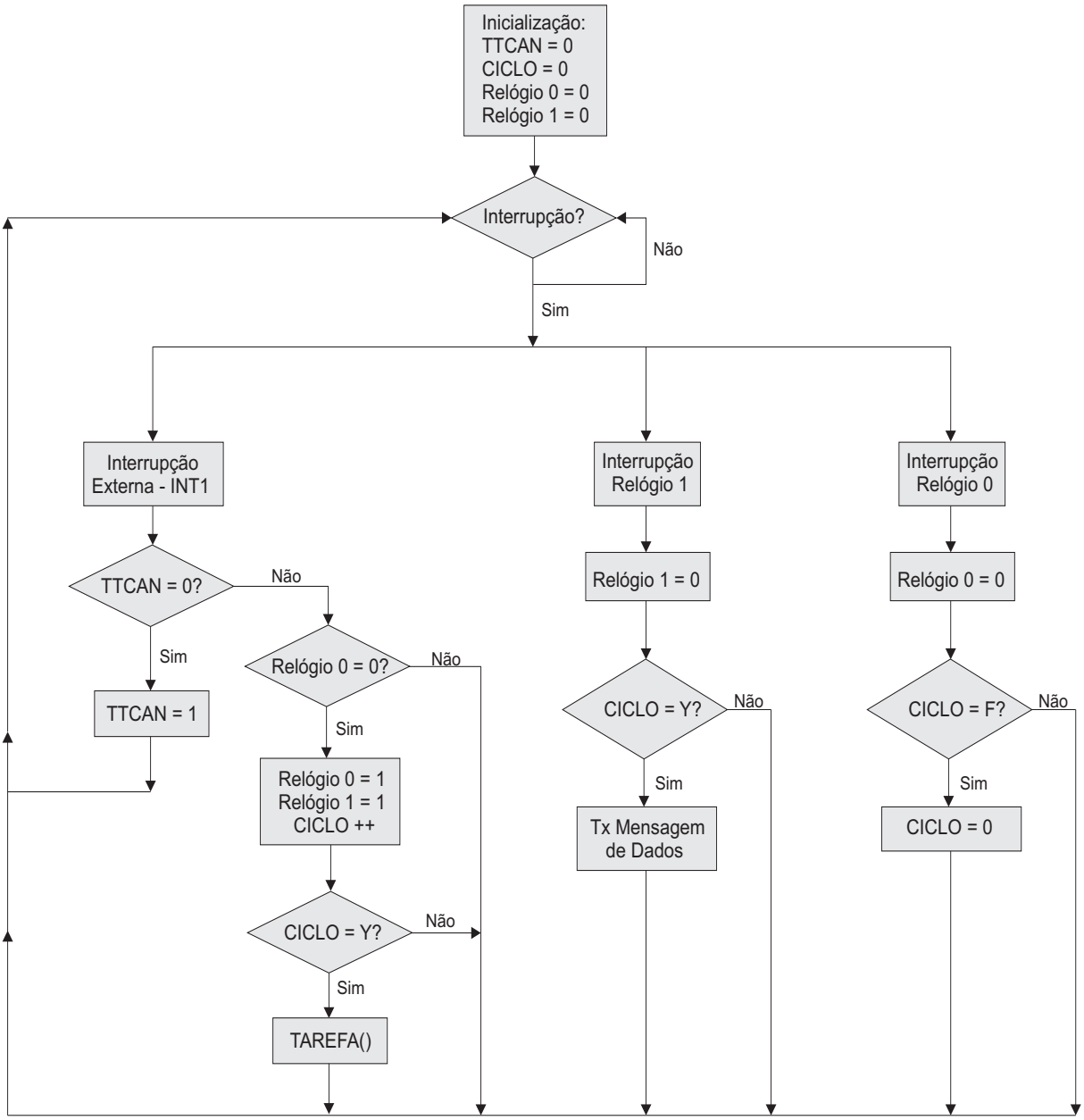

Figura 4.9: Fluxograma do algoritmo do nó escravo do tempo.

Inicialmente, todos os nós esperam a recepção do *bit* SOF de uma mensagem enviada pelo nó supervisor da rede para iniciar a comunicação. Neste instante, os relógios de todos os nós estão desligados (Relógio 0 = 0 e Relógio 1 = 0), e as variáveis TTCAN e CICLO possuem valor '0'. Na recepção do *bit* SOF desta mensagem, a interrupção INT1 ocorrerá em todos os nós. Neste instante, todos os nós ajustam a variável TTCAN para o valor '1'. Esta variável indica que a comunicação TTCAN foi iniciada. Após a recepção e aceitação desta mensagem, somente pelo nó mestre do tempo, a interrupção INT0 ocorrerá. Na rotina de serviço de interrupção, o nó mestre transmite a primeira mensagem de referência, iniciando o ciclo básico do sistema. Na recepção do *bit* SOF

desta mensagem de referência, a interrupção INT1 ocorrerá em todos os nós. Visto que em todos os nós a variável TTCAN está ajustada em '1' e o relógio 0 está desligado '0', a rotina de serviço de interrupção executa as seguintes atividades: aciona os relógios 0 e 1; e incrementa a variável CICLO que inicialmente estava ajustada em '0'. Caso este seja o ciclo básico em que o nó transmitirá uma mensagem de dados, a função TAREFA() será executada. Esta função realiza a conversão analógico digital dos sinais provenientes dos sensores conectados aos respectivos nós e armazena-os no *buffer* de transmissão da mensagem para serem enviados na respectiva janela de tempo do ciclo básico, quando ocorrer a interrupção no relógio 1.

A interrupção do relógio 1 ocorrerá sempre no *overflow* deste relógio. Na rotina desta interrupção, o relógio 1 é desligado '0' e caso o valor da variável CICLO seja igual a X (Figura 4.8) ou igual a Y (Figura 4.9), uma mensagem de dados será transmitida.

A rotina do serviço de interrupção executada na interrupção do relógio 0 depende do tipo de nó. No nó mestre, as seguintes atividades são realizadas: o relógio 0 é desligado '0'; uma mensagem de referência será transmitida; e caso a variável CICLO for igual ao número de ciclos básicos do ciclo matriz (CICLO = F nas Figuras 4.8 e 4.9), esta variável será zerada. Nos nós escravos, as mesmas tarefas são executadas, exceto o envio de uma mensagem de referência.

4.5.4 Projeto do Ciclo Matriz

O projeto de um ciclo matriz envolve a especificação dos seguintes parâmetros:

- Tempo de duração (Período) do ciclo básico;
- Tempo de duração (Período) do ciclo matriz;
- Quantidade de ciclos básicos;
- Tempo de duração das janelas de tempo;
- Quantidade de janelas de tempo do ciclo básico;
- Determinação dos marcos do tempo *Tx-Trigger*.

Para especificar estes parâmetros é necessário ter um conhecimento prévio das informações do período e do tempo gasto na transmissão de cada uma das mensagens da rede. Cada mensagem possui um período fixo. Atribui-se como tempo de duração de um ciclo básico, o menor período entre o conjunto de períodos das mensagens que serão escalonadas. O tempo de duração do ciclo matriz é igual ao maior período entre os períodos do mesmo conjunto de mensagens. Deste modo, a quantidade de ciclos básicos do ciclo

matriz é determinada pela divisão entre o maior e o menor período deste conjunto de mensagens.

O tempo de duração de uma janela de tempo depende do comprimento da mensagem e do tempo de *bit* da rede. Conforme representado na Figura 4.2, um ciclo matriz pode ser visto como uma matriz, onde as linhas são os ciclos básicos e as colunas são as janelas de tempo. Deste modo, a coluna de cada ciclo básico deve ter o mesmo comprimento. No entanto, pode existir a possibilidade de mensagens com diferentes tamanhos serem escalonadas na mesma coluna (diferentes ciclos básicos). Neste caso, atribui-se como comprimento da janela o tempo gasto para transmitir a maior mensagem que será enviada nesta coluna do ciclo matriz.

Após o término de uma janela exclusiva ou arbitrária, existe uma janela de tempo livre, a qual é utilizada para alocar pelo menos uma mensagem de erro e o intervalo entre mensagens considerando o pior caso. Portanto, esta janela possui um comprimento de no mínimo $31 * \tau_{bit}$.

A quantidade de janelas de tempo depende do comprimento das mensagens e do ciclo básico, e do período das mensagens. Deve-se garantir que todas as mensagens da rede serão escalonadas no ciclo matriz.

Os marcos do tempo *Tx-Trigger* indicam o início de cada janela de tempo. Cada nó gera seu *Tx-Trigger* localmente baseado no relógio 1, conforme apresentado anteriormente.

4.6 Conclusões

Neste Capítulo descreveu-se brevemente o protocolo de comunicações TTCAN (nível 1 e nível 2). As razões para o desenvolvimento deste protocolo foram explicadas inicialmente. Devido a não disponibilidade de um *chip* qualificado para propósitos de aplicações, foi proposto uma implementação por *software* do TTCAN nível 1, baseado em alguns requisitos de *hardware* do controlador CAN MCP2515. Apresentou-se um fluxograma de execução do nó mestre do tempo e dos escravos do tempo, e um método para projetar um ciclo matriz conforme o algoritmo proposto.

Capítulo 5

Projeto do Sistema de Aquisição de Dados Distribuído

5.1 O Sistema de Aquisição de Dados Distribuído

O sistema de aquisição de dados é constituído por oito nós sensores e um nó supervisor, os quais comunicam-se por meio de um barramento com comprimento de 25 m, de acordo com o protocolo de comunicações CAN ou TTCAN (*software*), a uma taxa de transmissão de 1 Mbps. Na Figura 5.1 representa-se um esquema com a arquitetura do sistema de aquisição de dados no laboratório de geração termoelétrica.

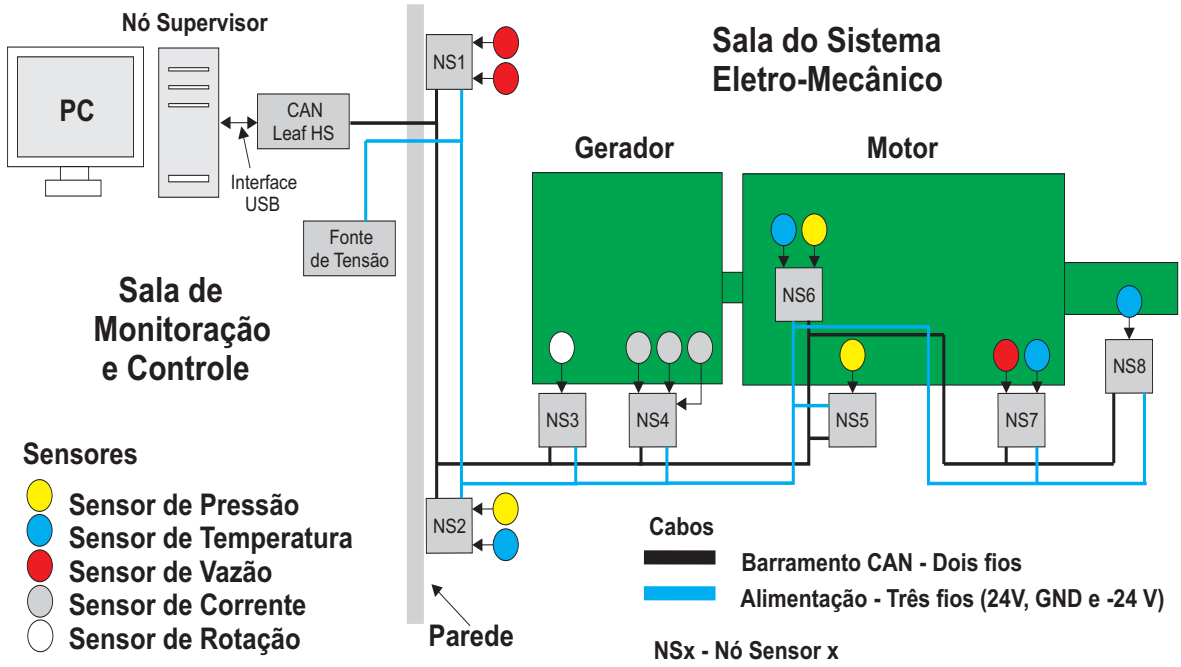

Figura 5.1: Representação da arquitetura do sistema de aquisição de dados no laboratório de geração termoelétrica.

A quantidade e o posicionamento dos nós sensores foram determinados de acordo com a localização dos sensores no laboratório de geração termoelétrica. Cada nó sensor é responsável pela aquisição e transmissão das informações convertidas pelos seus respectivos sensores para o nó supervisor, conforme os requisitos de tempo. É possível conectar no máximo oito sensores por nó[1]. Uma fonte de tensão ($\pm$24 V e GND) fornece energia elétrica para todos os nós sensores, por meio de um cabo (22 m) com três fios. A razão para este nível de tensão é devido aos sensores de vazão necessitarem de uma tensão de alimentação de 24 V. Além disso, a maioria dos circuitos integrados utilizados no condicionamento dos sinais necessitam de uma tensão de $\pm 15V$. Um detalhamento de toda parte de alimentação dos nós sensores é apresentado no Capítulo 9 seção 9.2.1.

5.2 Nó Supervisor

O nó supervisor é constituído por um computador Pentium 4 com velocidade de 2 GHz e 2 GB de memória RAM, um dispositivo CAN *leaf professional* HS (KVASER, 2007) e um *software* de aplicação. O computador e o dispositivo CAN comunicam-se via *interface* USB a uma taxa de transmissão de 12 Mbps. A escolha do dispositivo CAN *leaf professional* foi baseada nas seguintes características: capacidade de processar até 20 000 mensagens por segundo; precisão do relógio de 1 μs; CPU interna ao dispositivo; *transceiver* interno baseado no padrão ISO 11898-2 (alta velocidade); disponibilidade de *drivers* para linux; facilidade na programação por meio de API; e *interface* USB 2.0.

O sistema operacional utilizado no computador é o Linux 2.6 com a distribuição Ubuntu 6.06 LTS (UBUNTU, 2007). A utilização do linux garantiu um rápido processamento das informações pelo *software* de aplicação, evitando sobrecargas nos *buffers* de recepção do *hardware* e *software*. Além disso, existe a possibilidade de instalar um kernel em tempo real para aplicações de tempo real crítico, como o RTLinux (FSMLABS, 2007), RTAI (RTAI, 2007) entre outros. Um esquema com os componentes do nó supervisor está representado na Figura 5.2.

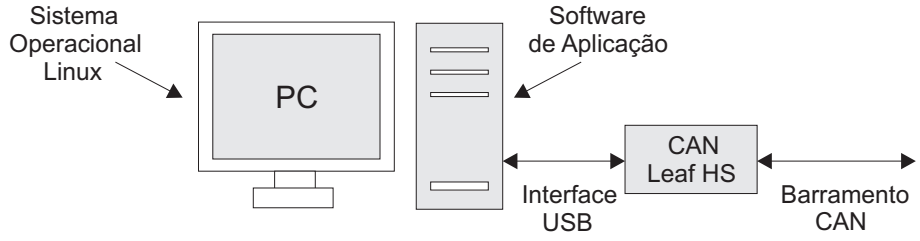

Figura 5.2: Esquema com os componentes do nó supervisor.

[1]O conversor analógico digital do microcontrolador utilizado possui apenas oito canais multiplexados.

A configuração e controle do dispositivo CAN foi realizada por meio da API CANLIB (*Application Programmer Interface CAN LIBrary*), a qual é fornecida pelo fabricante do dispositivo *leaf professional* HS, assim como o *driver* de instalação do dispositivo. API CANLIB é formada por várias funções, as quais utilizamos no projeto do *software* de aplicação. Na Figura 5.3 apresenta-se um diagrama com os componentes utilizados na comunicação entre o dispositivo CAN e o *software* de aplicação.

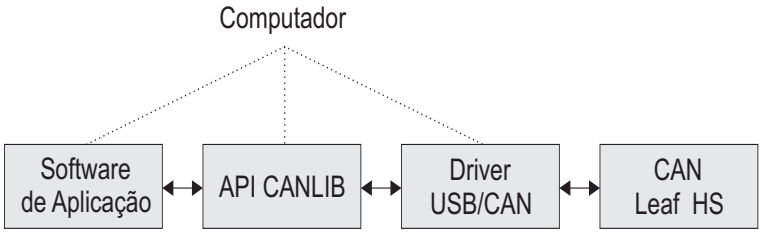

Figura 5.3: Diagrama com os componentes utilizados na comunicação entre o dispositivo CAN e o *software* de aplicação.

5.2.1 Software de Aplicação

O *software* de aplicação foi programado em linguagem C e compilado no gcc. Para ambos os modos de operação (CAN ou TTCAN), o algoritmo de execução do *software* de aplicação é idêntico. Devido o nó supervisor receber somente mensagens após realizar a inicialização do sistema, não existe nenhum algoritmo de sincronização deste nó. O nó supervisor funciona somente no modo CAN. No entanto, ele é capaz de receber mensagens com a rede (nós sensores) funcionando no modo CAN e no modo TTCAN. Um fluxograma simplificado do nó supervisor está representado na Figura 5.4. Inicialmente, o *software* realiza a definição das variáveis locais e globais, cria dois arquivos de textos, os quais serão utilizados para armazenar as informações das mensagens enviadas por cada um dos nós sensores. Em seguida, é definida uma função baseada na biblioteca signal.h para interromper o algoritmo de execução e finalizar a execução do sistema após digitar "crtl-c"no terminal linux. Logo após, é realizada a configuração do controlador e dos parâmetros do tempo de *bit*.

A operação da rede é iniciada com o envio da mensagem de inicialização. Inicialmente, a variável FIM é definida com valor '0'. O *software* realiza duas tarefas: verifica se alguma mensagem chegou; e se ocorreu a interrupção SIGINT. Após a recepção de uma mensagem (dados, sobrecarga ou erro), todas as informações desta mensagem são armazenadas no arquivo de texto 1 e impressas no terminal linux, e uma variável que indica o tipo de mensagem é incrementada (DADOS, SOBRECARGA ou ERRO). Quando a interrupção SIGINT ocorre, a variável FIM é ajustada para o valor '1' e as informações

das variáveis DADOS, SOBRECARGA, ERRO e o tempo total de execução do sistema são armazenadas no arquivo de texto 2.

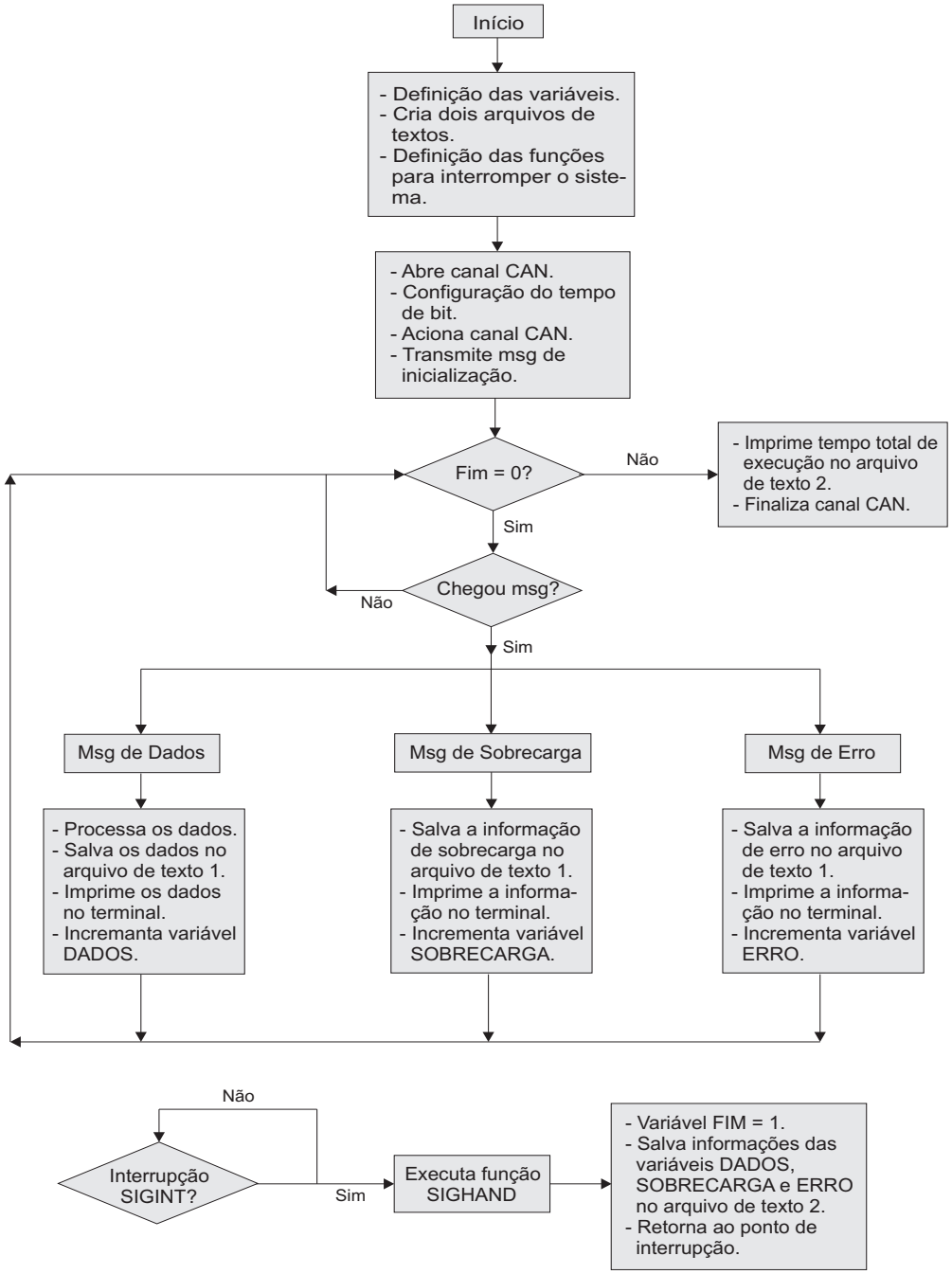

Figura 5.4: Fluxograma de execução do *software* de aplicação do nó supervisor.

5.3 Nó Sensor

Cada nó sensor é constituído por uma placa CAN/TTCAN, uma placa para condicionamento de sinais dos sensores e um *firmware* embarcado no microcontrolador da placa CAN/TTCAN. Todas as placas e *firmware* foram projetadas pelo autor deste trabalho durante o projeto de dissertação. Cada nó sensor está embarcado em uma caixa de plástico do tipo PB114 com as dimensões 55 mm × 97 mm × 147 mm. A caixa dispõe externamente de quatro conectores (Alimentação do nó, barramento CAN/TTCAN, *interface* serial (conector DB9) e entrada dos sinais dos sensores), dois botões (*Reset* e modo de programação do microcontrolador) e quatro LEDs. Internamente a caixa, existem dois cabos que fazem a conexão entre as duas placas: o cabo 1 é utilizado para distribuir energia elétrica para a placa circuito de condicionamento, enquanto que o cabo 2, é utilizado para enviar os sinais provenientes dos sensores (após serem condicionados) para a placa CAN/TTCAN, onde esses sinais serão convertidos digitamente pelo conversor A/D do microcontrolador. Uma fotografia de um nó sensor está representada na Figura 5.5.

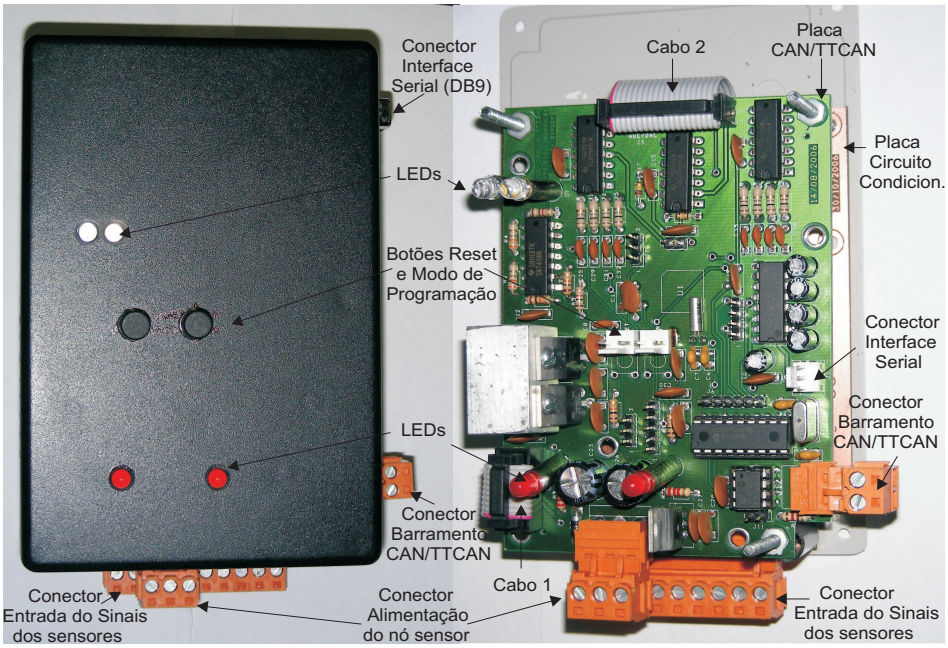

Figura 5.5: Fotografia de um nó sensor.

5.3.1 Placa CAN/TTCAN

A placa é constituída por três blocos principais: alimentação; nó CAN/TTCAN *stand-alone*; e circuitos para condicionamento dos dispositivos da placa. Um diagrama de blocos desta placa está representado na Figura 5.6. No Capítulo 9 seção 9.2.1 apresenta-se uma descrição detalhada de todos os circuitos desta placa.

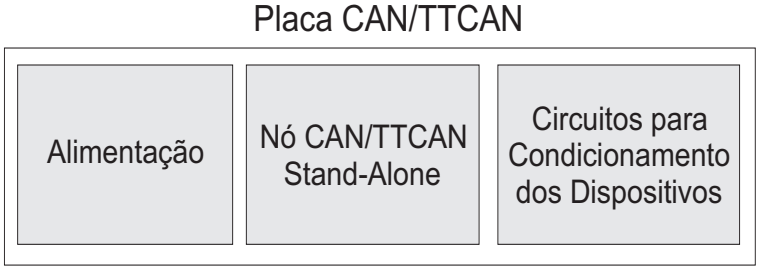

Figura 5.6: Representação em diagrama de blocos da placa CAN/TTCAN.

Nó CAN/TTCAN

O nó CAN/TTCAN é baseado em uma arquitetura *stand-alone* constituída pelos seguintes dispositivos:

- Microcontrolador ADuC842 (DEVICE, 2003);
- Controlador CAN MCP2515 (MICROCHIP, 2005);
- *Transceiver* MCP2551 (MICROCHIP, 2003).

A comunicação entre o ADuC842 e o MCP2515 é realizada por meio da *interface* SPI (*Serial Peripheral Interface*), a uma taxa de transmissão de 8,33 Mbps. A SPI é uma *interface* serial síncrona que transfere e recebe dados (8 *bits*) simultaneamente de um dispositivo mestre para um ou vários dispositivos escravos. A *interface* consiste de quatro linhas: SCK ou SCLOCK (*Clock*), SI ou MOSI (*Master Out Slave In*), SO ou MISO (*Master In Slave Out*) e $\overline{CS}$ (*Chip Select*). Neste circuito, o ADuC842 é o dispositivo mestre e o MCP2515 é o dispositivo escravo. Na Figura 5.7 representa-se um diagrama de blocos do nó CAN/TTCAN.

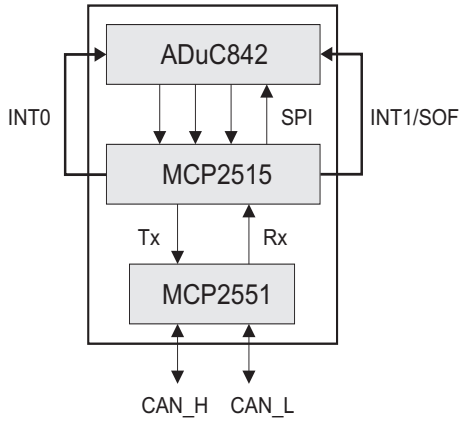

Figura 5.7: Representação em diagrama de blocos do nó CAN/TTCAN.

A escolha do controlador CAN MCP2515 foi baseada no fato deste dispositivo possuir os requisitos básicos para implementar uma versão por *software* do TTCAN nível 1, conforme citado na seção 4.4. Já o microcontrolador ADuC842 foi escolhido devido as suas características, como: *interface* SPI com elevada taxa de transmissão; freqüência do relógio do *core* de 16,78 MHz; tempo de conversão A/D na ordem de unidades de microsegundos; disponibilidade de temporizadores (relógio) entre outras. Essas características permitiram que os requisitos da aplicação fossem cumpridos.

Placa Condicionamento dos Sinais dos Sensores

Existem oito diferentes placas de condicionamento, uma para cada um dos oito nós sensores. Cada placa possui os circuitos necessários para condicionar os sinais provenientes dos respectivos sensores associados a ela. Os circuitos de condicionamento utilizados para cada um dos sensores e o funcionamento deles, são idênticos aos circuitos apresentados na seção 9.

Cada placa possui três conectores. Um destes conectores recebe a alimentação da placa CAN/TTCAN, que é utilizado para alimentar os circuitos integrados e os sensores quando necessário. O segundo é utilizado para conectar os sinais provenientes dos sensores antes do condicionamento e o último é utilizado para conectar os sinais dos sensores já condicionados para a placa CAN/TTCAN. Uma fotografia da placa de condicionamento do nó sensor 01 está representada na Figura 5.8.

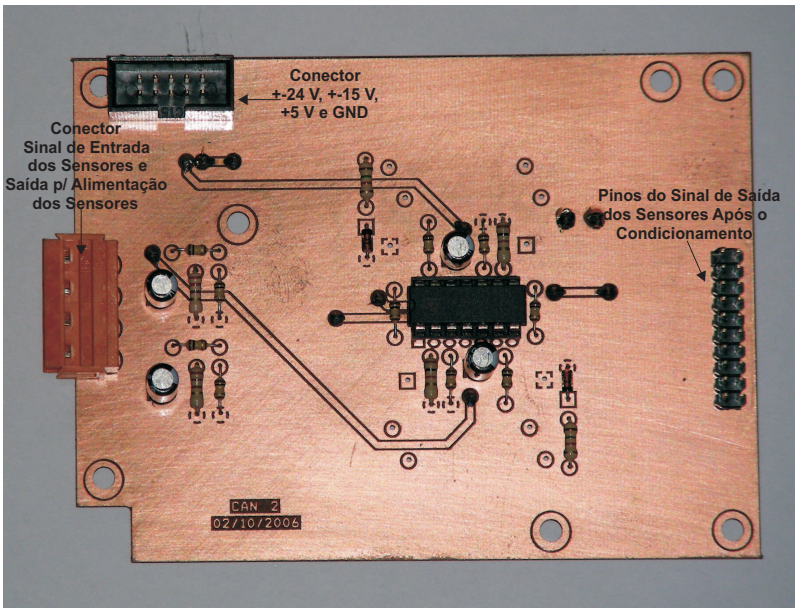

Figura 5.8: Fotografia da placa circuito de condicionamento do nó sensor 01.

5.3.2 Firmware

Cada nó sensor possui dois tipos de *firmware*, um para o modo CAN e o outro para o modo TTCAN. Os *firmware* de cada nó foram programados em linguagem C e compilados no Keil μVision (DEVICE, 2007), assim como a geração do arquivo executável. O *download* do arquivo executável é realizado por meio do aplicativo WSD (*Windows Serial Download*(DEVICE, 2007)) via a *interface* serial. Ambas as ferramentas de *software* funcionam somente no sistema operacional *Windows*.

No modo CAN, o *firmware* de cada nó sensor executa tarefas idênticas, conforme o fluxograma representado na Figura 5.9.

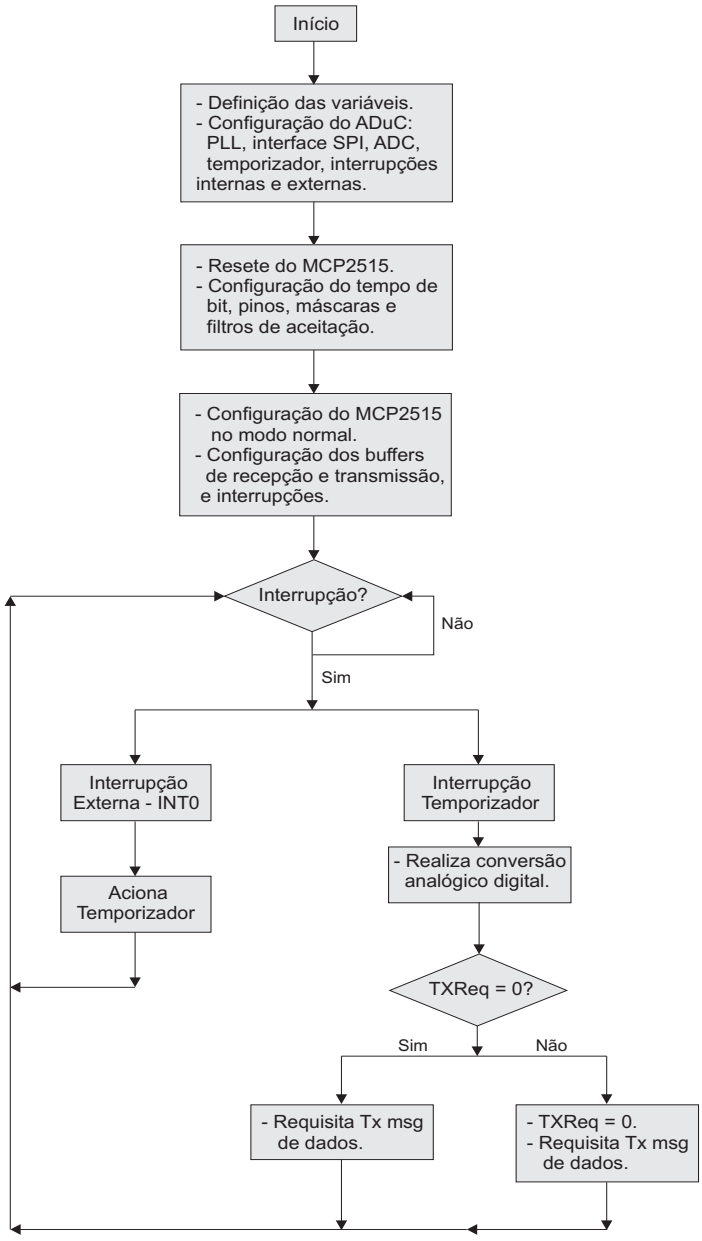

Figura 5.9: Fluxograma de execução do *firmware* de um nó sensor.

De acordo com a Figura 5.9, inicialmente é realizada a configuração dos periféricos do ADuC842 que serão utilizados na aplicação, e a configuração do controlador CAN MCP2515. Em seguida, o nó sensor espera o envio da mensagem de inicialização do nó supervisor. Na recepção desta mensagem, a interrupção externa INT0 ocorrerá. A rotina de serviço desta interrupção aciona o temporizador que será utilizado para gerar periodicamente a interrupção temporizador. Nesta interrupção, o ADuC842 realiza a conversão analógico digital dos sinais provenientes dos sensores conectados ao nó. Logo após, verifica se a mensagem anterior foi transmitida (TXReq = 0). Se TXReq for '0', o ADuC842 requista a transmissão de uma mensagem com os novos dados. Caso contrário, o algoritmo requisita o aborto da mensagem anterior ajustando a variável TXReq para o valor '0', e realiza a requisição de transmissão de uma mensagem com os novos dados.

No modo TTCAN, existe um *firmware* idêntico para cada nó sensor, exceto para o nó sensor 04, que é o mestre do tempo. O funcionamento desses algoritmos foram apresentados na seção 4.5.3 do Capítulo anterior.

Capítulo 6

Resultados Experimentais

Foram realizados experimentos no Laboratório de Geração Termoelétrica com a rede funcionando no modo de comunicação CAN e no modo de comunicação TTCAN. Para ambos os modos de comunicações, utilizou-se o nó supervisor com os Sistemas Operacionais (SO) Linux Ubuntu e *Windows* XP.

Cada nó transmite uma única mensagem periódica para o nó supervisor a uma taxa de transmissão de 1 Mbps via um barramento com 25 m de comprimento. As informações das mensagens foram visualizadas por meio de um terminal no computador do nó supervisor.

6.1 Experimentos - Rede CAN

Os experimentos com a rede CAN foram realizados com oito nós sensores e um nó supervisor. No entanto, somente alguns dos sensores foram conectados, pois o sistema termo-elétrico estava desativado durante o período de coleta dos resultados. Portanto, foi possível apenas adquirir as grandezas provenientes dos sensores de temperatura e de um sensor de pressão (ambiente), os quais não dependem do funcionamento do sistema termo-elétrico. Informações dos sensores de corrente, rotação e vazão não foram coletados. Para manter o trafégo na rede, todos os nós sem sensores conectados simularam a aquisição e transmissão dessas informações.

Todas informações dos nós sensores utilizados neste experimento estão indicadas na Tabela 6.1.

Tabela 6.1: Informações dos nós sensores.

Nó Sensor	**Identificador**	**Período de Tx (ms)**	**Sensores**	**Quantidade de bytes de dados**
Nó 01	597	500	-	4
Nó 02	853	500	01 Temperatura 01 - Pressão	4
Nó 03	341	1	-	2
Nó 04	85	0,5	-	6
Nó 05	1109	500	-	2
Nó 06	1877	500	01 Temperatura	4
Nó 07	1621	500	01 Temperatura	4
Nó 08	1365	500	01 Temperatura	2

Experimento com o SO Linux

Neste experimento, utilizou-se o SO Linux Ubuntu. Observou-se que os *buffers* de recepção do nó supervisor não sofreram nenhuma sobrecarga, visto que nenhuma notificação foi sinalizada. Uma tela do terminal do Linux Ubuntu com as informações durante um intervalo de tempo de 4009 μs de cada um dos nós está representada na Figura 6.1.

Figura 6.1: Tela do terminal Linux Ubuntu - Rede CAN.

Observe que em cada linha de texto visualizada na Figura, estão indicadas as informações do identificador (Id) do nó, os dados recebidos na mensagem e o instante de tempo em que a mensagem foi recebida no nó supervisor. As informações de temperatura estão em °C, pressão em bar e o tempo em μs.

Experimento com SO *Windows* XP

Uma tela do terminal no *Windows* XP com as informações do identificador das mensagens enviadas pelos nós sensores, os dados recebidos e os instantes em que as mensagens foram recebidas estão indicadas na Figura 6.2. As informações de temperatura estão em °C, pressão em bar e o tempo em ms.

Neste experimento, ocorreram algumas sobrecargas nos *buffers* de recepção do nó supervisor. A primeira sobrecarga indicada pela letra 'o', ocorreu no instante de tempo 525,650 ms. Esta sobrecarga causou um atraso de 18,9 ms na recepção da próxima mensagem. Estes atrasos ocorreram devido o *Windows* XP não ser considerado um sistema operacional em tempo real para uma aplicação com estes requisitos de tempo.

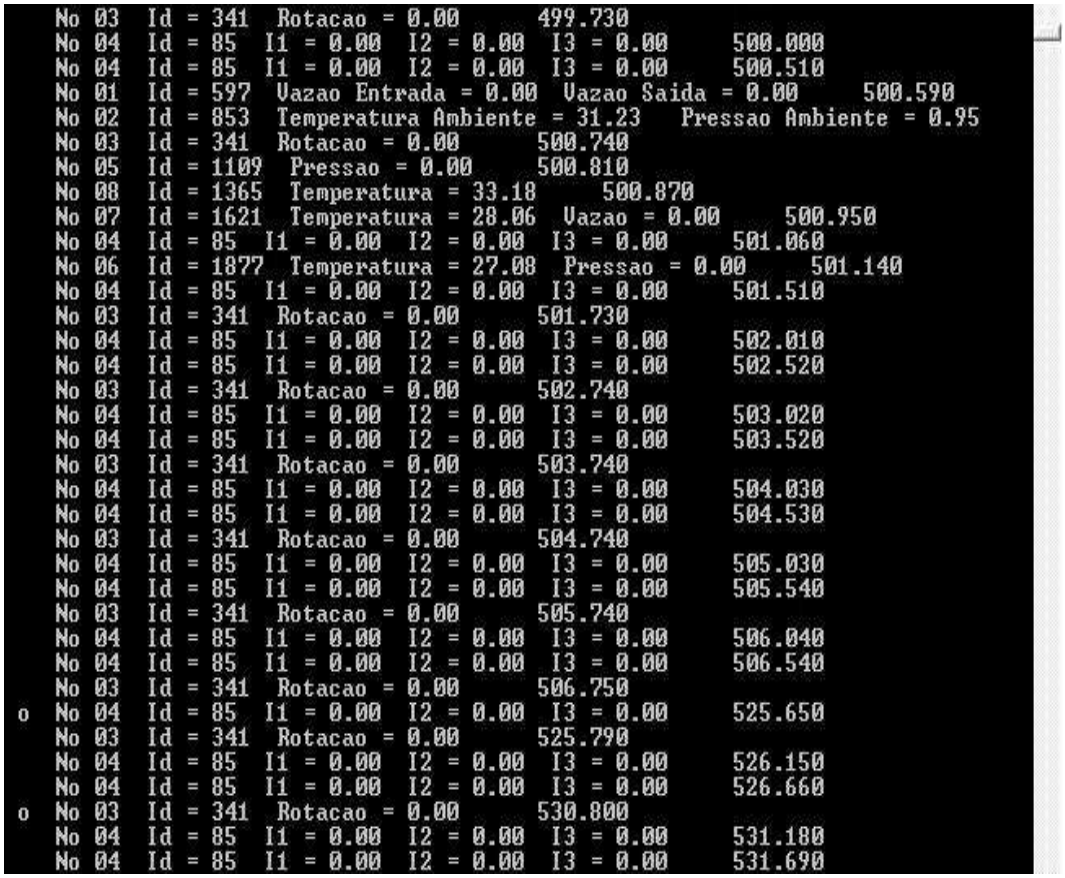

Figura 6.2: Tela do terminal *windows* XP - Rede CAN.

6.2 Rede TTCAN

Os experimentos com a rede TTCAN foram realizados com seis nós sensores e um nó supervisor. Neste experimento, nenhum sensor foi conectado. Assim como no experimento da rede CAN, o sistema termo-elétrico estava desativado. Para manter o trafégo na rede, todos os nós sensores simularam a aquisição e transmissão das informações do sensores.

Foi projetado um ciclo matriz com quatro ciclos básicos. Para permitir uma melhor visualização dessas informações no terminal do computador e no oscilóscopio, atribuiu-se pequenos períodos para as mensagens de cada nó. As informações do identificador e quantidade de *bytes* de dados de cada nó sensor, indicadas na Tabela 6.1, foram também utilizadas neste experimento. As informações do período, ciclo básico e *Tx-Trigger* de cada uma das mensagens dos respectivos nós estão indicados na Tabela 6.2. O nó 04 é o mestre do tempo.

Tabela 6.2: Informações do período, ciclo básico e *Tx-Trigger* das mensagens TTCAN.

Nó Sensor	**Identificador**	**Período de Tx (μs)**	**Ciclo Básico**	**Tx-Trigger (μs)**
Nó 04	869	510	1, 2, 3 e 4	0
Nó 01	597	2000	1	338
Nó 02	853	-	-	-
Nó 03	341	1000	1 e 3	86
Nó 04	85	510	1, 2, 3 e 4	212
Nó 05	1109	2000	2	338
Nó 06	1877	2000	3	338
Nó 07	1621	-	-	-
Nó 08	1365	2000	4	338

Uma representação do ciclo matriz utilizado neste experimento está representado na Figura 6.3. Janelas de tempo livre são abreviadas pela sigla JTL. As JTL menores possuem um espaço de tempo de 34 μs, o qual possibilita a transmissão de até uma mensagem de erro considerando o pior caso. Todas as janelas restantes são exclusivas.

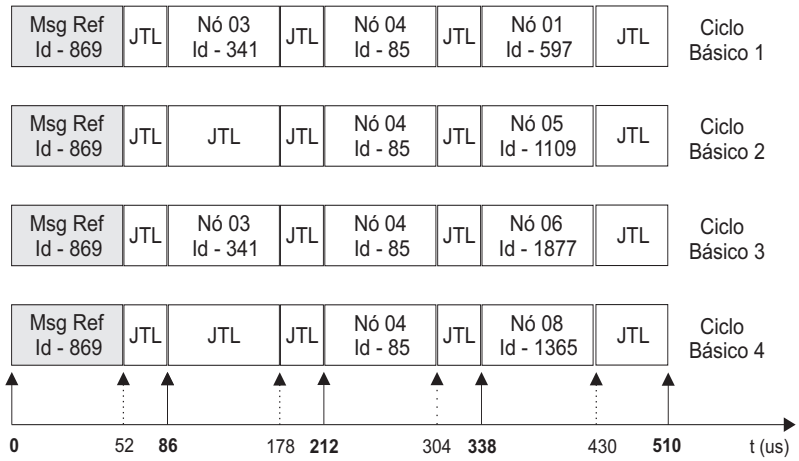

Figura 6.3: Ciclo matriz utilizado no experimento.

Experimento com o SO Linux

Na Figura 6.4 representa-se a tela do osciloscópio com os sinais das mensagens do ciclo matriz e os sinais gerados no pino SOF do controlador CAN de um nó. Observe que no início de cada mensagem existe um pulso SOF. Conforme indicado na Tabela 6.2, o ciclo matriz é formado por quatro ciclos básicos. As mensagens de referências de cada ciclo básico estão indicadas pela sigla MR.

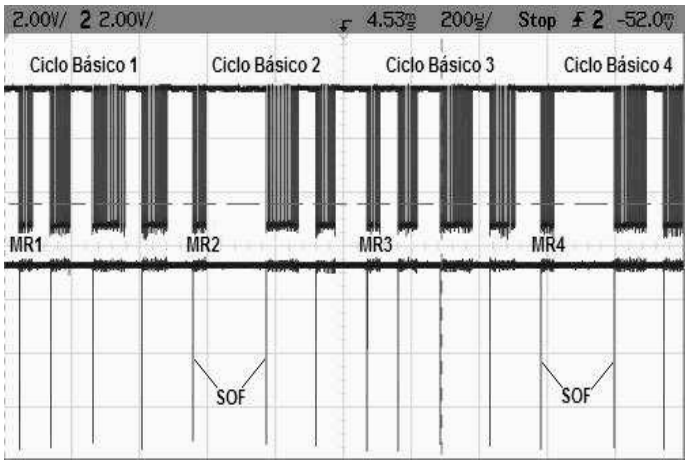

Figura 6.4: Tela do osciloscópio com os sinais das mensagens do ciclo matriz e os sinais SOF.

As informações das mensagens na tela do terminal Linux do nó supervisor estão representadas na Figura 6.5. O ciclo básico 1 é iniciado pelo envio da mensagem de referência no instante de tempo 1702569 μs. Observe na Figura que a duração de cada ciclo básico varia entre 511 e 512 μs.

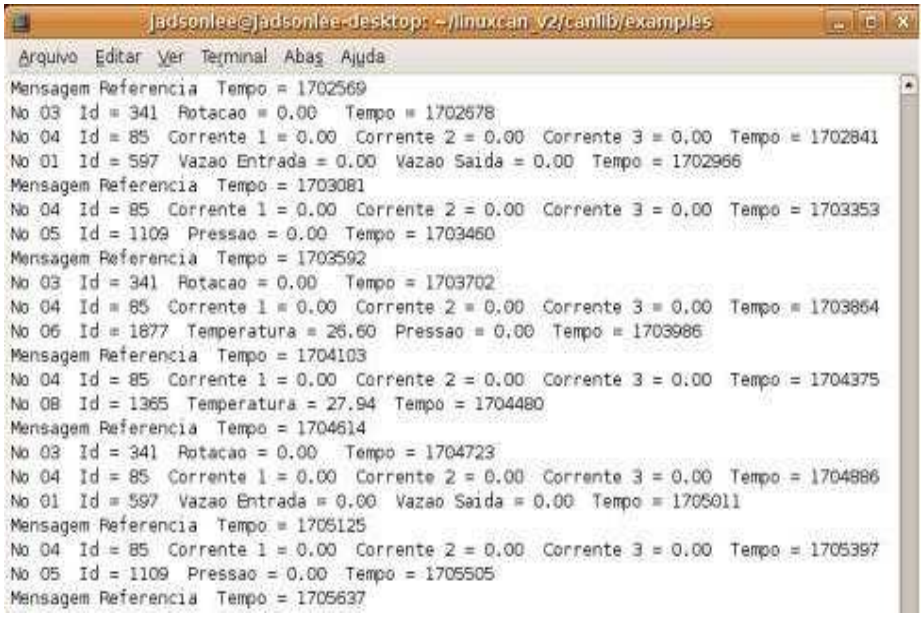

Figura 6.5: Tela do terminal linux com as informações das mensagens do ciclo matriz.

Experimento com SO *Windows* XP

Uma tela do terminal com as informações recebidas pelo nó supervisor está representada na Figura 6.6. Assim como no experimento da rede CAN, ocorreram algumas sobrecargas nos *buffers* de recepção do nó supervisor, conforme indicado na Figura nos instantes de tempo 1660,180 e 1661,330 ms e sinalizado pela letra 'o'. Apesar das sobrecargas o sistema foi capaz de retomar o sincronismo entre os nós, garantindo a comunicação TTCAN de acordo com o ciclo matriz projetado.

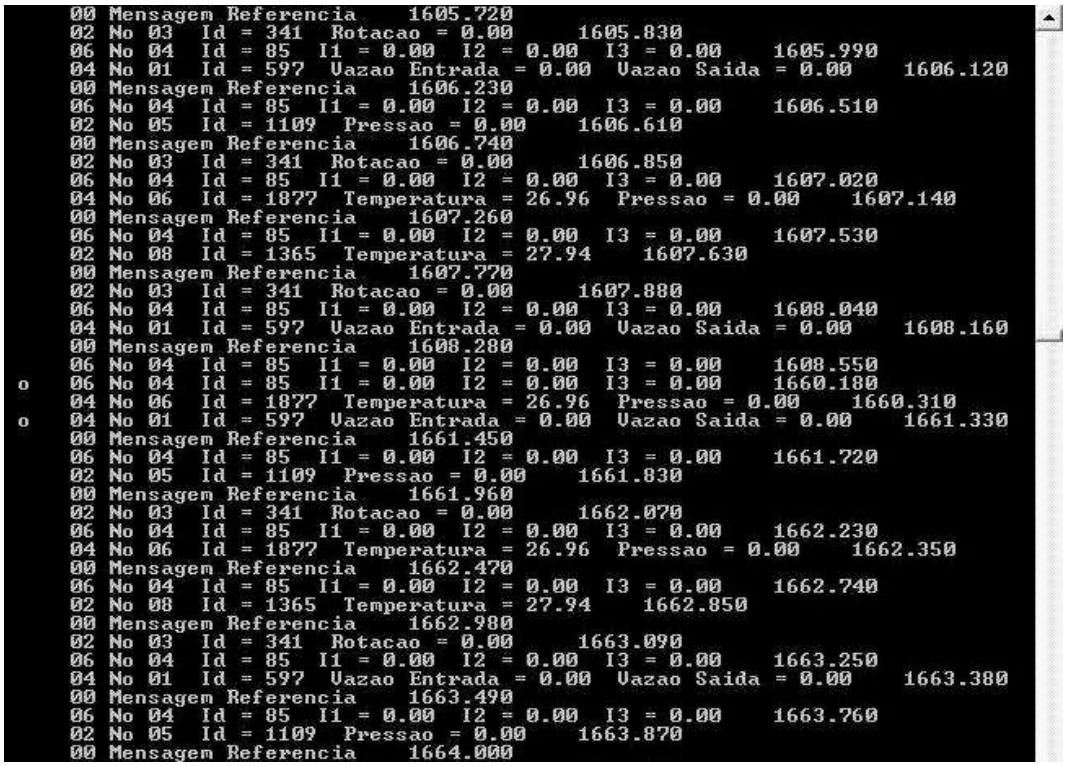

Figura 6.6: Tela do terminal *windows* XP - Rede TTCAN.

Capítulo 7

Conclusões e Sugestões para Trabalhos Futuros

7.1 Conclusões

Nesta dissertação foi desenvolvido um sistema de aquisição de dados distribuído, que foi projetado para ser utilizado no Laboratório de Geração Termoelétrica instalado nas dependências do Departamento de Engenharia Elétrica da Universidade Federal de Campina Grande.

O sistema de aquisição de dados foi baseado no protocolo de comunicações CAN (*Controller Area Network*). Uma versão *Time-Triggered* do protocolo CAN foi implementada por *software* no mesmo *chip* CAN, utilizando um algoritmo de sincronização baseado na especificação TTCAN (*Time-Triggered* CAN) nível 1. Foram desenvolvidos oito nós sensores e um nó supervisor. Todos os nós sensores apresentaram bom funcionamento. O nó supervisor apresentou problemas de sobrecarga nos *buffers* de recepção quando foi utilizado no computador o sistema operacional *Windows* XP. Isto ocorreu devido o alto requisito de tempo de alguns dos nós, visto que o Wundows não é um sistema operacional em tempo real. Para evitar este problema, utilizamos o sistema operacional Linux Ubuntu.

Os requisitos de tempo das mensagens para ambas as redes foram satisfeitos. A grande vantagem do sistema TTCAN é o comportamento determinístico no tempo. No entanto, existe a possibilidade das mensagens TTCAN não chegarem ao nó supervisor caso ocorram erros no barramento, pois o mecanismo de re-transmissão automática é desabilitado. O sistema CAN apresentou uma maior flexibilidade quando comparado a rede TTCAN. Este sistema reage melhor a eventos externos assíncronos tais como, erros no barramento, pois mesmo após um erro na mensagem, esta mensagem poderá ser re-transmitida automaticamente.

7.2 Sugestões para Trabalhos Futuros

Para continuação deste trabalho sugere-se:

- Desenvolver uma *interface* para visualização das informações no nó supervisor;
- Realizar uma análise dos tempos de resposta no pior caso das mensagens CAN sob condições ideais (sem erros no barramento);
- Determinar um modelo matemático capaz de descrever o comportamento de erros no barramento para o ambiente do laboratório, e estender a análise dos tempos de resposta das mensagens considerando este modelo de erros.
- Desenvolver mecanismos de tolerância à falhas no sistema TTCAN.
- Desenvolver aplicações de controle em malha fechada com os nós CAN/TTCAN desenvolvidos.

Referências Bibliográficas

AG, B. et al. *Byteflight Specification*. www.byteflight.com, 2001.

ALBERT, A. Comparison of event-triggered and time-triggered concepts with regard to distributed control systems. *Embedded World, Nümberg*, 2004.

ALBERT, A.; HUGEL, R. Heuristic scheduling concepts for ttcan networks. *Proceedings of The 10th international CAN Conference, Italy*, 2005.

ALBERT, A.; STRASSER, R.; TRäCHTLER, A. Migration from can to ttcan for a distributed control system. *Proceedings of The 9th international CAN Conference, Germany*, p. 05/9 – 05/15, 2003.

ALMEIDA, L.; PEDREIRAS, P.; FONSECA, J. The ftt-can protocol: Why and how. *IEEE Transactions on Industrial Electronics*, v. 49(6), p. 1198–1201, 2002.

ALMEIDA, L. M. P. de. *Flexibility and Timeliness in Fieldbus-Based Real-Time Systems*. Tese (Doutorado) — University of Aveiro, Portugal, 1999.

ALMEIDA, L. M. P. de; FONSECA, J. A.; FONSECA, P. Flexible time-triggered communication on a controller area network. *Proceedings of the 19th IEEE Real-Time Systems Symposium*, 1998.

ARCNET. Março 2007. Disponível em: <www.arcnet.com>.

BAILEY, M. Can in electric vehicles. *Proceedings of the 3rd international CAN Conference, France*, 1996.

BENEDETTI, M. R.; NETTO, J. C. Canopen aplicado a sistemas distribuídos em tempo real no setor de transmissão de energia elétrica. *Workshop de tempo real, WTR'2003, Natal, RN, Brasil*, 2003.

BOSCH. Ttcan testechip. Março 2007. Disponível em: <www.can.bosch.com>.

BOSCH, R. *CAN Specification Version 2.0*. Stuttgart, Germany, 1991.

BOSCH, R. Ttcan ip module. Março 2007. Disponível em: <www.semiconductors.bosch.de>.

BOURDON, G.; RUAUX, P.; DELAPLACE, S. Can for autonomous mobile robot. *Proceedings of The 3rd international CAN Conference, France*, 1996.

BROOKS, M. J. Controller area network for monitor and control in alma. *Advanced Telescope and Instrumentation Control Software*, v. 4009, p. 276–285, 2000.

BROSTER, I. *Flexibility in Dependable Real-Time Communication.* Tese (Doutorado) — University of York, England, August 2003.

CIA. Can in automation. Março 2007. Disponível em: <www.can-cia.org>.

CONTROLNET. Março 2007. Disponível em: <www.controlnet.org>.

DEVICE, A. *Ultraprecision Operational Amplifier - OP177.* USA, 1995.

DEVICE, A. *Monolithic Thermocouple Amplifiers with Cold Junction Compensation - AD594/595.* USA, 1999.

DEVICE, A. *ADuC842, microconverter, 12-bit ADCs with embedded 62 kB flash MCU.* USA, 2003.

DEVICE, A. Analog device, inc. Março 2007. Disponível em: <www.analog.com>.

DEVICENET. Devicenet an overview. Março 2007. Disponível em: <www.can-cia.org/devicenet>.

DWYLER, N. Medidor de vazão mtg. Março 2007. Disponível em: <www.nykondwyler.com.br/>.

ETSCHBERGER, K. *Controller Area Network: Basics, Protocols, Chips and Application.* First edition. Weingarten, Germany: IXXAT Press, 2001.

FüHRER, T. et al. Time-triggered communication on can (time-triggered can - ttcan). *Proceedings of 7th international CAN Conference, iCC, Amsterdam*, p. 92–98, 2000.

FLEXRAY. *FlexRay Communications Systems Protocol Specification Version 2.1.* www.flexray.com, 2004.

FONSECA, J. A.; XAVIER, C.; CARDOSO, S. Sas - sistema de aquisição de sismica com barramento de campo. *Actas do 3º Congresso Luso-Moçambicano de Engenharia, Maputo, Moçambique*, 2003.

FREDRIKSSON, L. B. Controller area network and the protocol can for machine control systems. *Mechatronics, Sweden*, v. 4(2), p. 159–192, 1994.

FSMLABS. Real-time linux. Março 2007. Disponível em: <www.fsmlabs.com>.

GERGELEIT, M.; STEICH, H. Implementing a distributed high-resolution real-time clock using the can bus. *Proceedings of the 1st international CAN conference*, 1994.

HARTWICH, F. et al. Can network with time triggered communication. *Proceedings of the 7th international CAN Conference, Holand*, 2000.

HARTWICH, F. et al. Timing in the ttcan network. *Proceedings of The 8th international CAN Conference, USA*, 2002.

HYTRONIC. *Transmissor de pressão model TM*. Brasil, 2005.

INCONTROL. Medidor de vazão para líquidos e gases tipo turbina série vtl. Março 2007. Disponível em: <www.incontrol.ind.br/medidoresvazao.htm>.

INTERBUS. Interbus. Março 2007. Disponível em: <www.phoenixcontact.com.br/en/prod1-6.html>.

ISO. *Road Vehicles - Controller Area Network (CAN) - Part 1: Data link layer and physical signalling*. ISO 11898-1, 1999.

ISO. *Road Vehicles - Controller Area Network (CAN) - Part 2: High-speed medium access unit and medium-dependent interface*. ISO 11898-2, 1999.

ISO. *Road Vehicles - Controller Area Network (CAN) - Part 3: Low-speed, fault-tolerant, medium-dependent interface*. ISO 11898-3, 1999.

ISO. *Road Vehicles - Controller Area Network (CAN) - Part 4: Time triggered communication*. ISO 11898-4, 2004.

JOHANSSON, K. H.; TöRNGREN, M.; NIELSEN, L. Vehicle applications of controller area network. *Handbook of Networked and Embedded Control Systems*, p. 741–766, 2005.

KAISER, J.; SCHAEFFER, P. Icu - a smart optical sensor for direct control. *Proceedings IEEE International Conference on Mechatronics and Machine Vision in Pratic, Hong Kong*, 2001.

KOPETZ, H.; BAUER, G. The time-triggered architecture. *In: IEEE*, v. 91, p. 112 – 126, 2003.

KRIESEL, W. R.; MADELUNG, O. W. *AS Interface - The Actuator-Sensor-Interface for Automation*. Múnchem, Germany: Carl Hanser Verlag, 1999.

KUROSE, J. F.; ROSS, K. W. *Redes de Computadores e a Internet: uma nova abordagem*. Primeira. Brasil: Addison Wesley, 2003.

KVASER. Kvaser - advanced can solutions. Março 2007. Disponível em: <www.kvaser.com>.

LAWRENZ, W. *CAN System Engineering: From Theory to Practical Applications*. First edition. Germany: Springer, 1997.

LEEN, G.; HEFFERNAN., D. Ttcan: A new time-triggered controller area network. *Microprocessors and Microsystems*, v. 26(2), p. 77–94, 2002.

LIBONATI, P. A. O.; OLIVEIRA, A.; CAMPOS, L. E. P. de. Sistema de aquisição de dados para laboratórios. *In: Encontro Para a Qualidade de Laboratórios, ENQUALAB, São Paulo*, v.1, p. 127–132, 2003.

LIMA, E. J. et al. Sensoring for retrofitting of an industrial robot. *11th IFAC Symposium on Information Control Problems in Manufacturing, Salvador, Brazil*, 2004.

LINDQVIST, D. *Clock Synchronization over CAN*. Dissertação — Chalmers University pf Technology - Göteborg, 2003.

MANCINI, D. et al. Active optics control of the vst telescope with the can field-bus. *Proceedings of International Conference on Accelerator and Large Experimental Physics Control System, ICALEPCS, Napoli, Italy*, 2001.

MARTíNEZ, R. et al. Arquitectura de control distribuida sobre bus can para robots moviles. *XXII Jornadas Automática, Barcelona*, 2001.

METZNER, A.; STIERAND, I. Analyzing mixed event triggered/time triggered systems. *Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2006.

MICROCHIP. *PIC16F7X Data Sheet*. USA, 2001.

MICROCHIP. *High-Speed CAN Transceiver - MCP2551*. USA, 2003.

MICROCHIP. *Stand-Alone CAN Cotroller with SPI Interface - MCP2515*. USA, 2005.

MüLLER, B. et al. Fault tolerant ttcan networks. *Proceedings of the 8th international CAN Conference, USA.*, 2002.

MORAES, F.; AMORY, A. M.; JúNIOR, J. P. Sistema integrado e multiplataforma para controle remoto de residências. *VII Workshop Iberchip, Montevidéu, Uruguai*, 2001.

NETO, P. R. de A.; GIACOMIN, J. C. Aquisição e análise de sinais elétricos. *Infocomp*, v.1, p. 7–12, 2004.

NICOLOSI, D. *Microcontrolador 8051*. Quinta. Múnchem, Germany: Érica, 2004.

NOLTE, T. *Reducing Pessism and Increasing Flexibility in The Controller Area Network*. Dissertação — Mälardalens University, Sweden, 2003.

NOLTE, T.; HANSSON, H.; NORSTRöM, C. Minimizing can response-time jitter by message manipulation. *IEEE Real Time Technology and Applications Symposium*, p. 197–206, 2002.

PANHAN, A. M. *Sistema de Aquisição de Dados e Monitoramento Remoto para Câmaras Frias e Sistemas de Refrigeração*. Dissertação — Universidade de Campinas - UNICAMP, Julho 2002.

PFEIFFER, O.; AYRE, A.; KEYDEL, C. *Embedded Networking eith CAN and CANopen*. RTC Books: ISBN 0-929392-78-7, 2003.

PROFIBUS. *PROFIBUS - Descrição Técnica*. Brasil, 2000.

RODRIGUES, L.; GUIMARãES, M.; RUFINO, J. Fault-tolerant clock synchronization in can. *Proceedings of the 19th real-time systems symposium, Spain*, p. 420–429, 1998.

RTAI. Real-time application interface for linux. Março 2007. Disponível em: <www.rtai.org>.

Sá, J. da S.; BARROS, P. R.; NETO, J. S. da R. Implementação e análise de uma rede can para controle de um sistema distribuído. *V Congresso Internacional de Automação, Sistemas e Instrumentação - ISA, Brasil*, 2005.

SANTOS, M. M. D.; STEMMER, M. R. Uma rede can aplicada ao controle de uma aeronave não tripulada - o helicóptero hélix. *XIV Congresso brasileiro de Automática, Natal, Brasil*, 2002.

SEMICONDUCTORS, N. *LF198/LF298/LF398, LF198A/LF398A Monolithic Sample-and-Hold Circuits*. Brasil, 2000.

SERCOS. Março 2007. Disponível em: <www.sercos.org>.

SILVA, L. P. da et al. Sistema de aquisição de dados de baixo custo para um grupo gerador do ciclo diesel. *Congresso Estudantil SAE Brasil*, 2005.

STEMMER, M. R. *Sistemas Distribuídos e Redes de Computadores para Controle e Automação Industrial*. Brasil, 2001.

TTTECH. *TTP/C Protocol Specification - Technical Report*. www.tttech.com, july 1999.

TURSKI, K. A global time system for can network. *Proceedings of the 1st international CAN conference*, 1994.

UBUNTU. Ubuntu: linux for human beings. Março 2007. Disponível em: <www.ubuntu.com>.

WORLDFIP. Março 2007. Disponível em: <www.worldfip.org>.

Capítulo 9

Anexos

9.1 Anexo A - Os Sensores

Os tipos, modelos, características e os respectivos circuitos de condicionamento utilizados no SADC serão apresentados nas seções seguintes.

9.1.1 Sensores de Temperatura

Todos os sensores de temperatura são termopares tipo J ou tipo K. Os circuitos de condicionamento são idênticos, exceto que o termopar tipo J utiliza o CI AD594 (DEVICE, 1999) e o tipo K o CI AD595 (DEVICE, 1999). Os CIs AD594 e AD595 são alimentados por uma tensão elétrica de ± 15 V. Na Figura 9.1 está representado o diagrama do circuito elétrico utilizado para condicionamento dos sensores de temperatura.

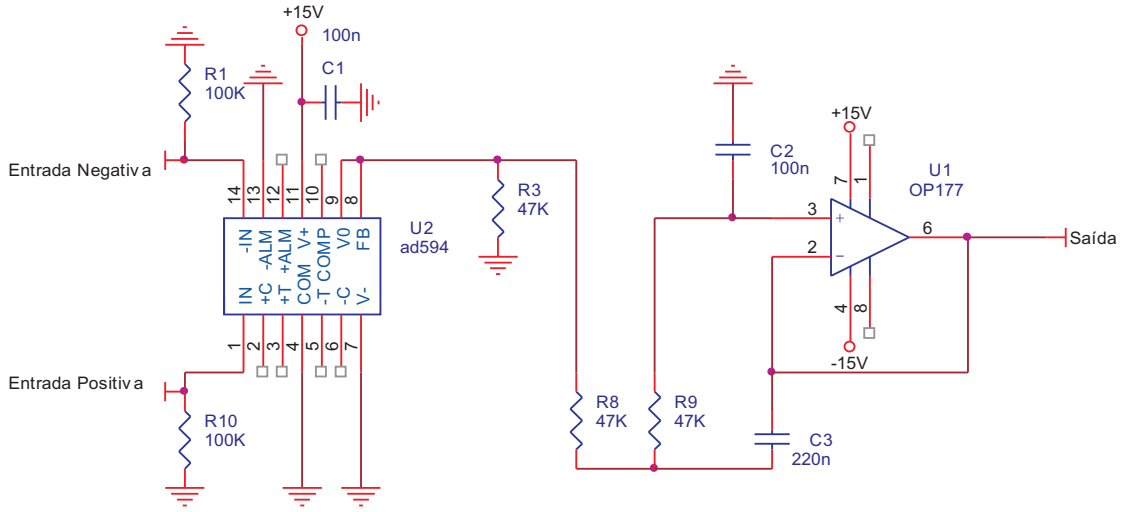

Figura 9.1: Diagrama do circuito elétrico utilizado no condicionamento dos sensores de temperatura.

O termopar é conectado nas entradas positiva (Tipo J - Ferro e Tipo K - Cromel) e negativa (Tipo J - Constantan e Tipo K - Alumel). O AD594/595 produz um sinal com nível de 10mV/°C no pino 9 e em seguida, esse sinal passa por um filtro passa-baixas de 2° ordem com freqüência de 50 Hz implementado pelo CI OP177 (DEVICE, 1995). Neste ponto, o sinal está pronto para ser convertido em um sinal digital.

9.1.2 Sensores de Pressão

Os sensores de pressão são do tipo TM25 (HYTRONIC, 2005). A tensão de alimentação para funcionamento destes sensores é de 15 V. O sinal elétrico convertido pelo TM25 está no padrão 4-20 mA. Basicamente, o circuito de condicionamento converte o padrão 4-20 mA para o nível de tensão proporcional entre 0-5 V. O circuito garante que um sinal de 4 mA na entrada resultará em um sinal de tensão de 0 V na saída (O ajuste é feito por meio do potenciômetro J1), e um sinal de 20 mA na entrada, em um sinal de 5 V na saída (O ajuste é feito por meio do potenciômetro J2). O diagrama de circuito elétrico utilizado para condicionamento deste sensor está representado na Figura 9.2.

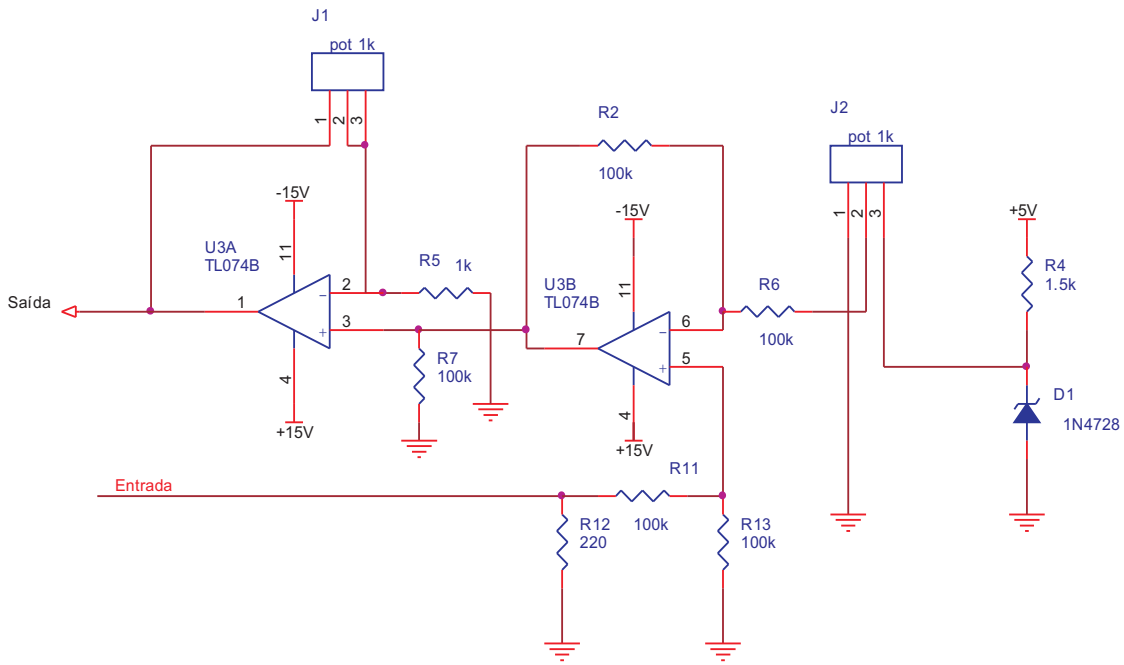

Figura 9.2: Diagrama do circuito elétrico utilizado no condicionamento dos sensores de pressão.

9.1.3 Sensores de Vazão

Foram utilizados três diferentes modelos de sensores de vazão:

- Modelo VTL da Incontrol (INCONTROL, 2007);
- Modelo MTG100F1S14RFN da Nykon Dwyler (DWYLER, 2007);
- Modelo MTG038RN4RFN da Nykon Dwyler (DWYLER, 2007).

A tensão de alimentação para ambos os modelos é de 24 V. O sinal elétrico de saída desses sensores está no padrão 4-20 mA. Portanto, o circuito de condicionamento utilizado nos sensores de vazão é idêntico ao condicionamento usado nos sensores de pressão.

9.1.4 Sensores de Corrente

Os sensores de corrente são constituídos por transformadores de corrente e uma resistência *shunt* de 1 Ω/50 V. O valor de tensão medido no resistor de 1 Ω é conectado nas entradas do subtrator implementado pelo CI TL084. Em seguida, este sinal passa por um retificador de onda completa e no estágio seguinte, por um filtro passa-baixas com freqüência de corte de 25 Hz e por um circuito *sample/hold* (LF398) (SEMICONDUCTORS, 2000) que faz a amostragem do sinal. Neste ponto, o sinal pode ser convertido digitalmente. O diagrama do circuito elétrico utilizado para condicionamento de um dos três sensores de corrente está representado na Figura 9.3.

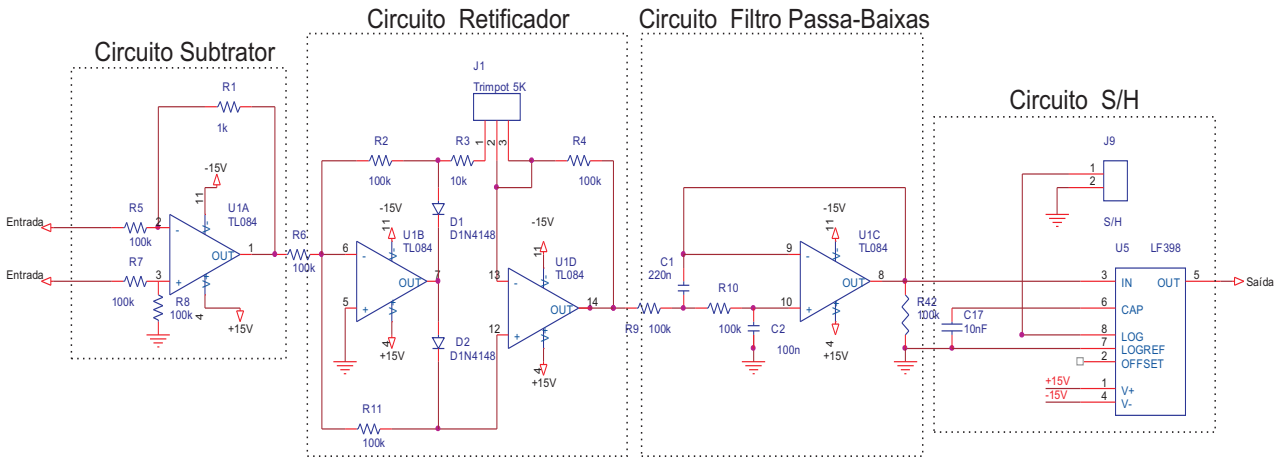

Figura 9.3: Diagrama do circuito elétrico utilizado no condicionamento dos sensores de corrente.

9.1.5 Sensor de Rotação

Existe apenas um sensor de rotação, que é baseado no CI LF2907. O sinal elétrico com a freqüência de rotação do gerador passa por um transformador que baixa a tensão para 9 V. Em seguida, o LF2907 gera uma tensão elétrica proporcional a freqüência deste sinal. Neste estágio, o sinal é amostrado e convertido digitalmente. O diagrama de circuito elétrico utilizado para condicionamento do sensor de rotação está representado na Figura 9.4.

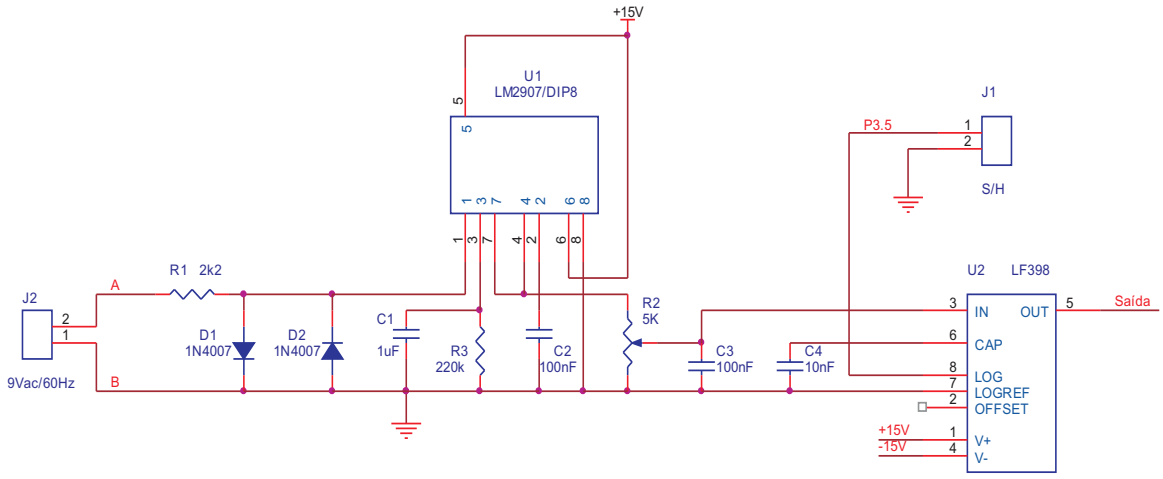

Figura 9.4: Diagrama do circuito elétrico utilizado no condicionamento dos sensores de rotação.

9.2 Anexo B - Nó Sensor

9.2.1 Placa CAN/TTCAN

A placa CAN/TTCAN é constituída por circuitos de condicionamento para o fornecimento adequado de energia elétrica para os diversos circuitos integrados da própria placa e da placa condicionamento de sinais, circuitos para o funcionamento do ADuC842 e um nó CAN/TTCAN *stand-alone* baseado no microconversor ADuC842, controlador MCP2515 e *transceiver* MCP2551. Na Figura 9.5 representa-se o diagrama de circuito elétrico da placa CAN/TTCAN. Nas seções seguintes será explicado o funcionamento dos circuitos elétricos desta placa.

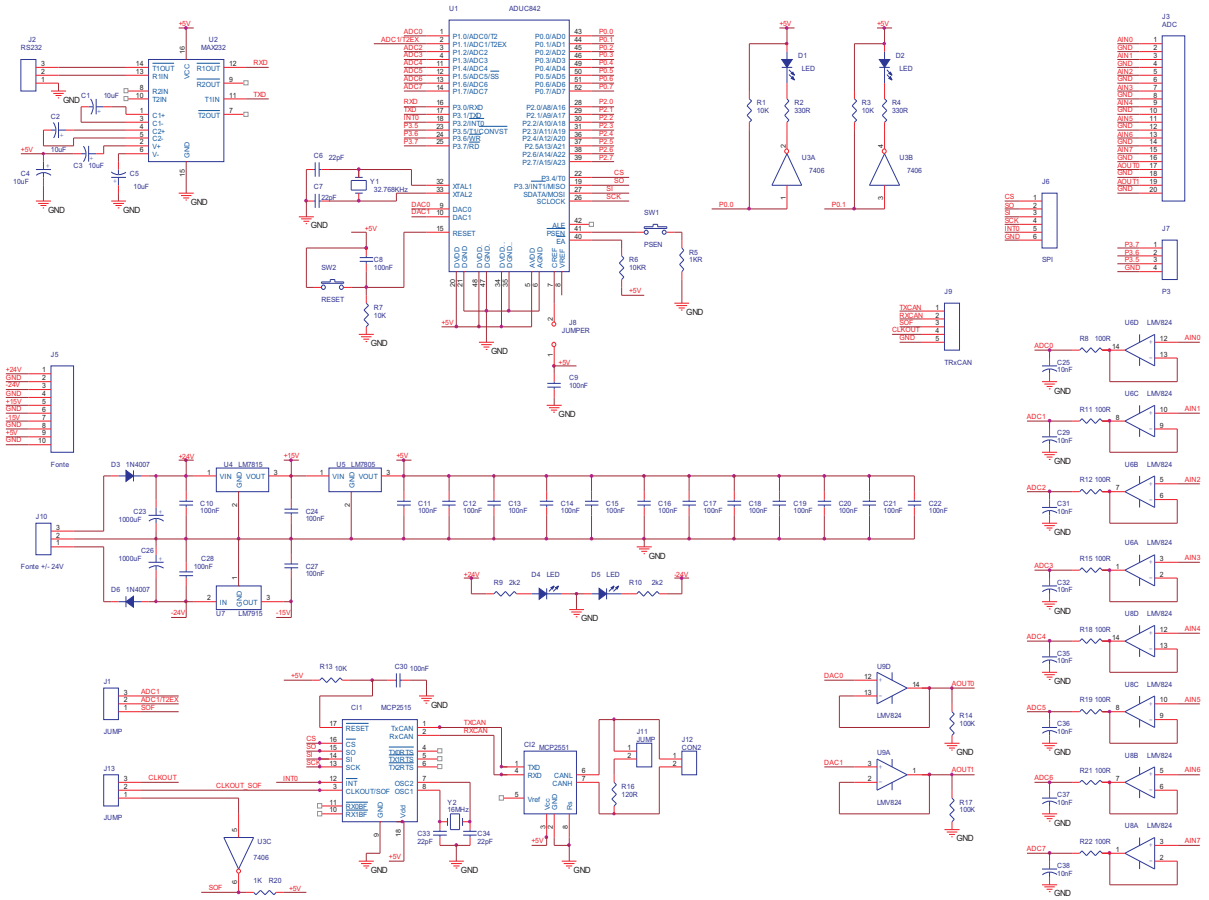

Figura 9.5: Diagrama do circuito elétrico de uma placa CAN/TTCAN.

Alimentação da Placa CAN/TTCAN

Os vários componentes dos nós sensores necessitam de diferentes níveis de tensão elétrica para um apropriado funcionamento. Para garantir estes níveis, foi projetado um circuito de alimentação que regula o nível de tensão de ±24 V fornecido por uma fonte externa, para as tensões de ±15 V e 5 V. Na Figura 9.6 apresenta-se o diagrama de circuito elétrico da alimentação da placa CAN/TTCAN.

As tensões elétricas fornecidas pela fonte são conectadas ao conector J10. Estas tensões são reguladas para +15 V, -15 V e +5 V por meio dos reguladores LM7815, LM7915 e LM7805, respectivamente. Estes níveis de tensão e os níveis ±24 V estão disponíveis no conector J5. Com um auxílio de um cabo, estes níveis de tensão são fornecidos para a placa circuito de condicionamento.

Os diodos D3 e D6 garantem a proteção do circuito contra uma eventual inversão de polaridade da fonte. Os capacitores C23 e C26, e os capacitores C10 e C28 são utilizados, respectivamente, para filtragem e desacoplamento DC (*Direct Current*) da fonte de ±24 V. Já os capacitores C24, C27 e C11 realizam o desacoplamento DC dos três reguladores

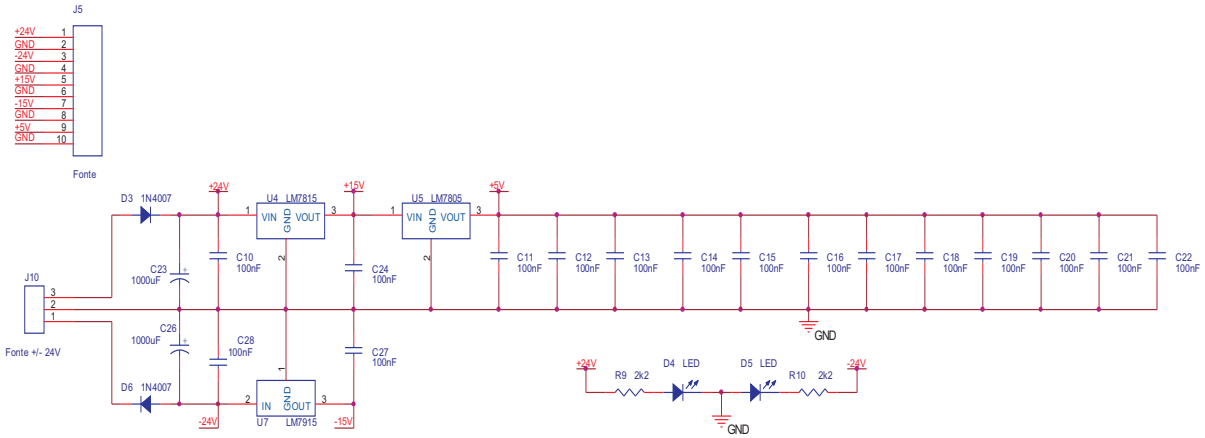

Figura 9.6: Diagrama do circuito elétrico da alimentação da placa CAN/TTCAN.

de tensão, garantindo uma alimentação de qualidade e sem ruídos de alta freqüência para os dispositivos que serão energizados por estas tensões. Os capacitores restantes são utilizados também para fazer o desacoplamento DC de cada um dos circuitos integrados que serão energizados por estas tensões. Os LEDs D4 e D5 serão acesos quando a placa for energizada.

Circuitos Básicos para o Funcionamento do ADuC842

Uma das configurações básicas para o funcionamento do ADuC842 está representada na Figura 9.7. Nesta configuração, o ADuC possui um cristal externo de 32,768 KHz. Por meio de um circuito PLL interno ao *chip*, é possível atingir a máxima freqüência do *core*, 16,78 MHz.

O ADuC é energizado pela conexão de uma tensão de +5 V nos pinos 5, 20, 34 e 48. Os pinos 6, 21, 35 e 47 devem ser aterrados. Uma tensão de referência interna de +2,5 V é disponível no *chip*. Esta tensão é utilizada como referência para o conversor A/D e D/A. Para obter uma tensão de referência maior (máximo +5 V), deve-se conectar a tensão desejada no pino de tensão de referência externa (pino 7) e realizar alguns ajustes no *firmware*. No circuito, uma tensão de referência externa de +5 V é garantida pela conexão de um *strap* no *jumper* J8. Tensões de referência negativas não podem ser conectadas ao dispositivo.

Os circuitos básicos conectados nos pinos 15 (*RESET*), 41 (PSEN) e 40 são especificados no *datasheet* do ADuC842. Basicamente, eles são usados para resetar, colocar o ADuC em modo de programação e habilitar o acesso a uma possível memória externa.

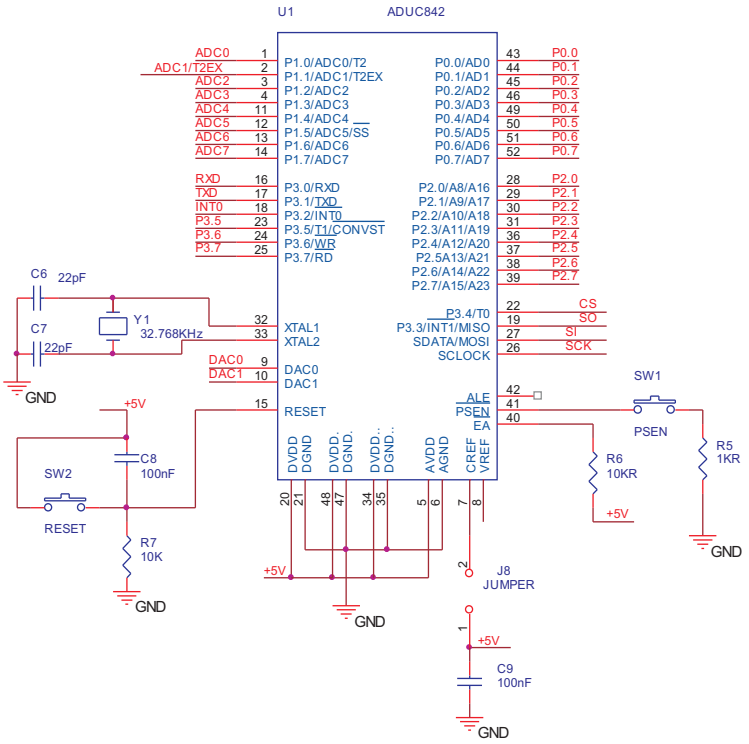

Figura 9.7: Diagrama do circuito elétrico básico para funcionamento do ADuC842.

Portas de Entrada e Saída

O ADuC842 dispõe de quatro portas de entrada e saída (Portas 0, 1, 2 e 3). Cada porta possui oito pinos, os quais são compartilhados com diversas funções.

Os pinos da porta 0 são pinos bidirecionais digitais do tipo coletor aberto. Utilizamos apenas os pinos P0.0 e P0.1, os quais estão conectados aos LEDs D1 e D2, respectivamente. Devido estes pinos serem do tipo coletor aberto é necessário fazer uma conexão externa com VCC (+5 V) por meio de um resistor *pull-up* (10 KΩ). Para garantir que os LEDs serão acesos com nível lógico '1' e apagados com o nível lógico '0', conectamos na saída dos pinos e em série com os LEDs um inversor baseado no CI 7406. Para limitar a corrente que passa pelos LEDs foi conectado em série um resistor de 330 Ω.

A porta P1 é utilizada apenas como entrada analógica ou digital. Na placa, todos os pinos são utilizados como entrada analógica, exceto o pino P1.1, que pode também ser utilizado como entrada digital no circuito TTCAN.

As portas 2 e 3 são portas bidirecionais digitais. A maioria dos pinos são compartilhados com outros periféricos. Na placa, os pinos P3.5, P3.6 e P3.7 estão disponíveis no conector J7 para serem utilizados como entrada ou saída digital. O pino P3.5 é utilizado para gatilhar os circuitos *sample/hold* na placa condicionamento de sinais. A porta 2 não está sendo utilizada nesta placa. Na Figura 9.8 está representado um diagrama do

circuito elétrico dos LEDs D1 e D2 e o conector J7.

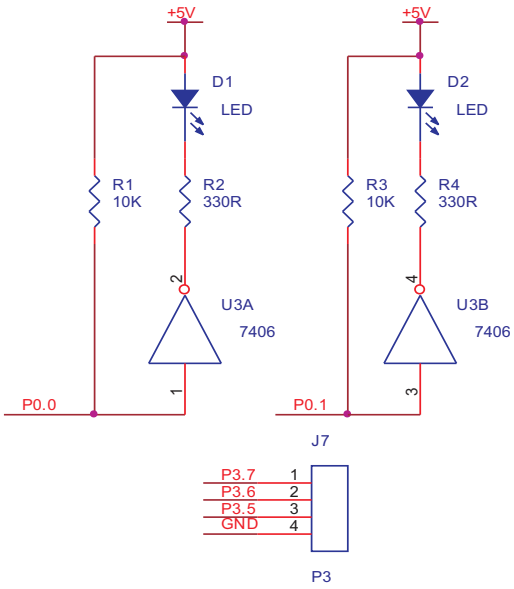

Figura 9.8: Diagrama do circuito elétrico dos LEDs D1 e D2 e o conector J7.

Conversores A/D e D/A

O ADuC842 possui um conversor analógico digital (A/D) com oito canais multiplexados e resolução de 12 *bits*, e dois conversores digitais analógicos (D/A) de 12 *bits*. Na Figura 9.9 apresenta-se um diagrama do circuito elétrico das entradas analógicas do conversor A/D e das saídas digitais do conversor D/A do ADuC842.

A conexão dos sinais analógicos com os oito canais de entrada do conversor A/D do ADuC842 são feitas através do conector J3. Além das entradas analógicas do conversor A/D, este conector possui os pinos de saída dos dois conversores D/A disponíveis no ADuC.

Conforme apresentado na Figura 9.9, cada um dos oito canais do conversor A/D estão protegidos por um circuito *buffer* utilizando amplificadores operacionais. Na saída de cada *buffer*, um filtro passa baixas com freqüência de corte de aproximadamente 160 KHz elimina os ruídos de alta freqüência que possam interferir na aquisição da tensão de entrada. Cada uma das duas saídas analógicas possuem também um *buffer* para evitar conexões diretas com os conversores D/A e limitar o fornecimento de corrente elétrica para o sistema que for conectado externamente.

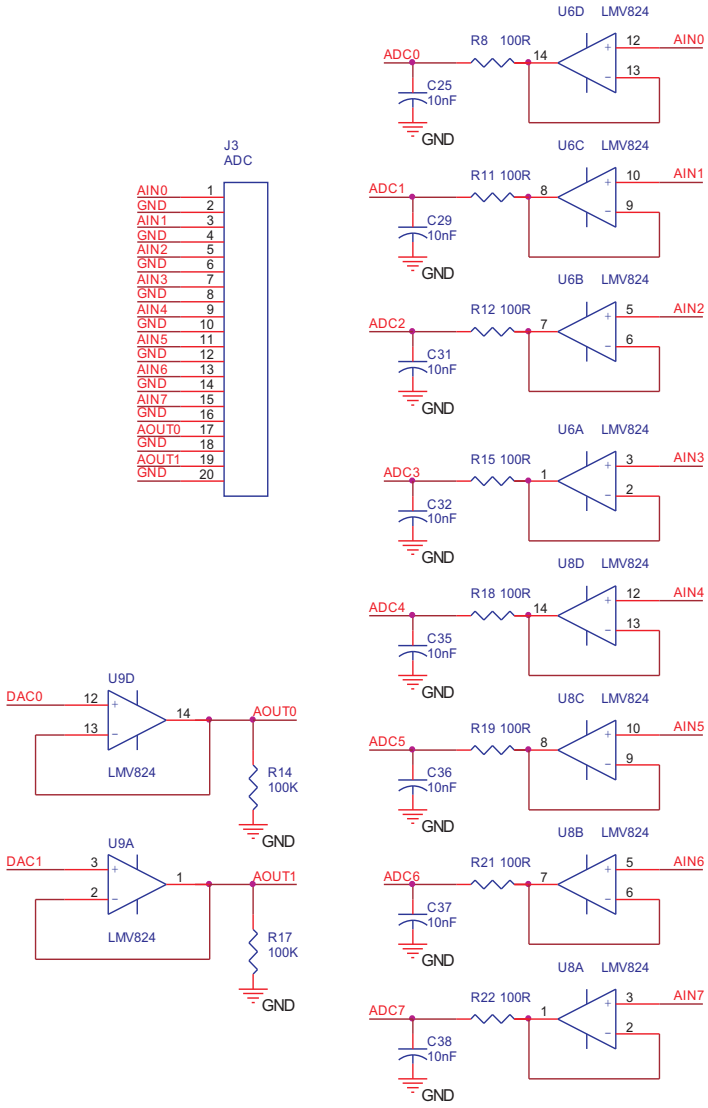

Figura 9.9: Diagrama do circuito elétrico das entradas analógicas do conversor A/D e das saídas digitais do conversor D/A do ADuC842.

Comunicação Serial RS-232

O ADuC842 dispõe da comunicação serial RS-232, que conectada na *interface* serial de um computador possibilita a gravação do *firmware* na sua memória. Além disso, pode-se utilizá-la na comunicação com outros dispositivos.

A comunicação serial é realizada por meio de um circuito que converte os níveis de tensão TTL provenientes do ADuC842 para os sinais do padrão RS-232. O circuito de *interface* é baseado no CI MAX232. Os sinais das linhas RXD e TXD do ADuC842 possuem níveis de tensão TTL (0V e 5V), os quais são convertidos para -10V (nível lógico 1) e +10V (nível lógico 0). A conexão com a *interface* RS-232 é realizada por meio do conector J2. Na Figura 9.10 arpesenta-se o diagrama de circuito elétrico da comunicação

serial RS-232 do ADuC842.

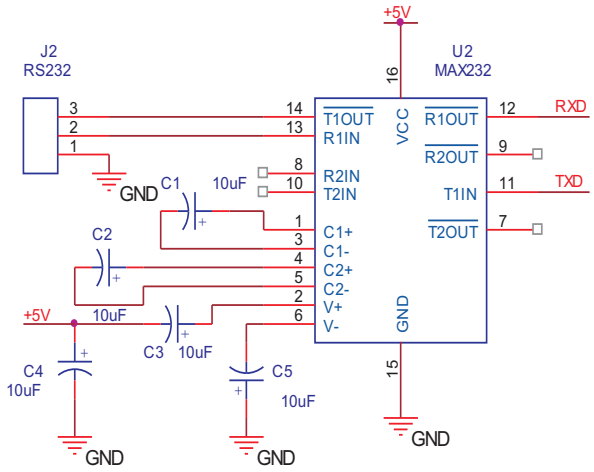

Figura 9.10: Diagrama do circuito elétrico da comunicação serial RS-232 do ADuC842.

Nó CAN/TTCAN

O nó CAN/TTCAN é constituído pelos seguintes dispositivos:

- Microconversor ADuC842;
- Controlador MCP2515;
- *Transceiver* MCP2551.

Um diagrama simplificado do circuito elétrico do nó CAN/TTCAN está representado na Figura 9.11.

O MCP2515 e o MCP2551 são energizados com uma tensão de +5 V nos pinos 18 e 3, e aterrados nos pinos 9 e 2, respectivamente. O pino 8 do MCP2551 conectado ao GND garante o funcionamento no modo *High-Speed*.

O MCP2515 foi projetado para funcionar com um cristal externo de até 25 MHz. Na placa, foi utilizado um cristal de 16 MHz conectado aos pinos 7 e 8 e aterrados via os capacitores C33 e C34 de 22 pF. Este cristal é utilizado para gerar a temporização dos *bits* das mensagens enviadas e recebidas pelo controlador CAN. Um circuito RC típico utilizado para resetar o controlador está conectado no pino 17. O resistor e o capacitor devem ter valores que mantenha o controlador em *reset* por no mínimo 2 μs após a alimentação. No circuito utilizamos R13 = 10KΩ e C30 = 100 ηF. Esses valores são mais que suficientes para garantir o *reset* do dispositivo, visto que a constante de tempo do circuito é 1 ms.

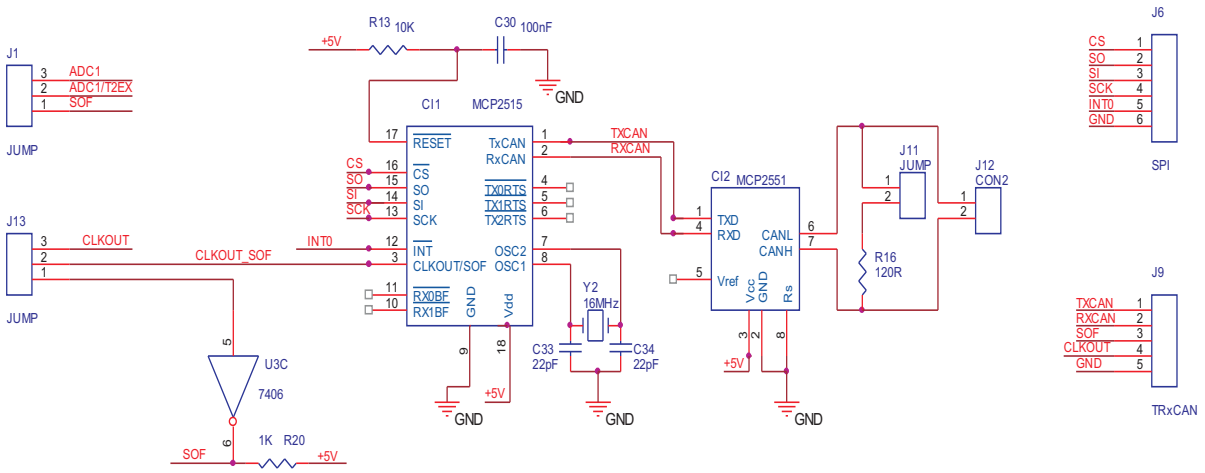

Figura 9.11: Diagrama simplificado do circuito elétrico do nó CAN/TTCAN.

A comunicação entre o MCP2515 e o ADuC842 é realizada por meio da *interface* SPI *Serial Peripheral Interface*) a uma taxa de 8,33 Mbps. A SPI é uma *interface* serial síncrona que transfere e recebe dados (8 *bits*) simultaneamente de um dispositivo mestre para um ou vários dispositivos escravos. A *interface* consiste de quatro linhas: SCK ou SCLOCK (*Clock*), SI ou MOSI (*Master Out Slave In*), SO ou MISO (*Master In Slave Out*) e $\overline{CS}$ (*Chip Select*). O linha SCK determina a velocidade e o sincronismo da transferência e recepção de dados. No dispositivo mestre, SCK é sempre uma saída, e no dispositivo escravo é sempre uma entrada. SI é a linha usada para transferir os dados do mestre para o escravo. SO é a linha onde os dados são transferidos do escravo para o mestre. A linha $\overline{CS}$ habilita a transferência dos dados quando o seu nível for baixo. No modo mestre esta linha é uma saída, e no modo escravo é uma entrada. Neste circuito, o ADuC842 é o dispositivo mestre e o MCP2515 é o dispositivo escravo. Na Tabela 9.1 estão listados os respectivos pinos do MCP2515 e ADuC842 utilizados na comunicação SPI.

Tabela 9.1: Pinos utilizados na comunicação SPI entre ADuC842 e o MCP2515.

Pino do ADuC842	**Pino do MCP2515**	**Função SPI**
26	13	SCK
27	14	SI
19	15	SO
22	16	CS

O MCP2515 possui os requisitos para implementar o protocolo CAN e o TTCAN nível 1. Deste modo, foi projetado um circuito capaz de funcionar em ambos os protocolos.

Para funcionar no modo TTCAN, os *jumpers* J1 e J13 devem está na posição 1-2. No entanto, o funcionamento no modo CAN também é garantido com essa configuração, apenas algumas modificações no *software* deverão ser realizadas. Para funcionar apenas no modo CAN, pelo menos o *jumper* J13 deve está na posição 2-3.

O MCP2515 gera um sinal de +5 V no pino 3 (CLKOUT/SOF) com duração de $2 * T_{OSC}$, quando detecta o ponto de amostragem do *bit* SOF de uma mensagem. Utilizamos este sinal para gerar uma interrupção no ADuC842 (Pino 2 − T2EX) e sincronizar os *clocks* baseados neste evento. No entanto, as interrupções do ADuC são sensíveis apenas a transições de borda de nível lógico alto para baixo. Então, foi conectado em série com o pino 3 do MCP2515 um inversor baseado no CI 7406.

A conexão do MCP2515 com o meio de transmissão é realizada por meio do *transceiver* MCP2551. A linha de transmissão TXCAN (pino 1) e a de recepção RXCAN (pino 2) estão ligadas aos pinos 1 e 4 do MCP2551, respectivamente. Os pinos 6 e 7 do MCP2551 estão ligados ao barramento CAN via o conector J12. Entre os pinos 6 e 7, temos um resitor de 120 Ω em série com o *jumper* J11. Um *strap* deverá ser conectado nesse *jumper* caso esse nó CAN/TTCAN esteja na extremidade da rede.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo