TRABALHO DE GRADUAÇÃO

IMPLEMENTAÇÃO DO PROTOCOLO INDUSTRIAL DE CAMPO DEVICENET EM PLATAFORMA RASPBERRY PI 3 PARA AUTOMAÇÃO DE MAQUETE FERROVIÁRIA

Eduardo Fonseca

William Jun Yamada

Brasília, Julho de 2018

ENGENHARIA
MECATRÔNICA
UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

IMPLEMENTAÇÃO DO PROTOCOLO INDUSTRIAL DE CAMPO DEVICENET EM PLATAFORMA RASPBERRY PI 3 PARA AUTOMAÇÃO DE MAQUETE FERROVIÁRIA

Eduardo Fonseca

William Jun Yamada

Relatório submetido como requisito parcial de obtenção
de grau de Engenheiro de Controle e Automação

Banca Examinadora

Prof. Guilherme Caribé de Carvalho, ENM/UnB _____
Orientador

Prof. Carlos Humberto Llanos Quintero, ENM/UnB _____
Examinador interno

Prof. Jones Yudi Mori Alves da Silva, ENM/UnB _____
Examinador interno

Brasília, Julho de 2018

FICHA CATALOGRÁFICA

JOÃO, DA SILVA
Título do trabalho dividido em mais de uma linha para títulos realmente longos como este,

[Distrito Federal] 2015.

x, 101p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2015). Trabalho de Graduação – Universidade de Brasília.Faculdade de Tecnologia.

1. Bla 2.Ble
3. Bli

I. Mecatrônica/FT/UnB II. Título (Série)

REFERÊNCIA BIBLIOGRÁFICA

PEREIRA, EDUARDO DA FONSECA; YAMADA, WILLIAM JUN (2018). Implementação do protocolo industrial de campo DeviceNet em plataforma Raspberry Pi 3 para automação de maquete ferroviária. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-$n^{\circ}022$, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 101p.

CESSÃO DE DIREITOS

AUTORES: Eduardo da Fonseca Pereira e William Jun Yamada

TÍTULO DO TRABALHO DE GRADUAÇÃO: Implementação do protocolo industrial de campo DeviceNet em plataforma Raspberry Pi 3 para automação de maquete ferroviária.

GRAU: Engenheiro ANO: 2018

Eduardo da Fonseca Pereira

William Jun Yamada

RESUMO

Palavras Chave:

ABSTRACT

Keywords:

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Siglas

CAN	*Controller Area Network*
CI	Circuito Integrado
CID	*Connection ID*
CIP	*Common Industrial Protocol*
CLP	Controlador Lógico Programável
CRC	*Cyclic Redudancy Check*
CSMA/CD	*Carrier Sense Multiple Access with Collision Detection*
DO	Saída Digital - *Digital Out*
DI	Entrada Digital - *Digital Input*
ECU	*Electronic Control Unit*
EDS	*Electronic Data Sheet*
I/O	Entrada/Saída - *Input/Output*
ISO	*International Organization for Standardization*
LAN	*Local Area Network*
OSI	*Open Systems Interconnection*
RTR	*Remote Time Request*
SAE	*Society of Automotive Engineers*
SPI	*Serial Peripheral Interface*
UCMM	*Unconnected Conection Message Manager*

Capítulo 1

Introdução

O presente capítulo apresenta sequencialmente aspectos relacionados à contextualização do projeto, definição do problema, objetivos gerais e metodologia empregada.

1.1 Contextualização

O aprofundamento da integração econômica, social, cultural e política gerada pela globalização têm exigido bastante do sistema de transporte como um todo. Para acompanhar tal avanço e atender a crescente demanda, a automação e o controle de processos vêm sendo incorporados aos diversos sistemas para obtenção de melhores resultados e para a solução dos novos desafios.

Levando em conta o sistema ferroviário, processos que vão desde a simples abertura e fechamento automático de portas em metrôs até o controle total das vias, horários, fluxo, velocidade das locomotivas e supervisionamento total por uma central, são exemplos da inserção da automação nesse sistema.

A Universidade de Brasília possui em suas dependências um modelo de circuito de uma linha ferroviária que permite implementar sistemas de automação e controle monitorados por computador para simular situações reais (figura 1.1).

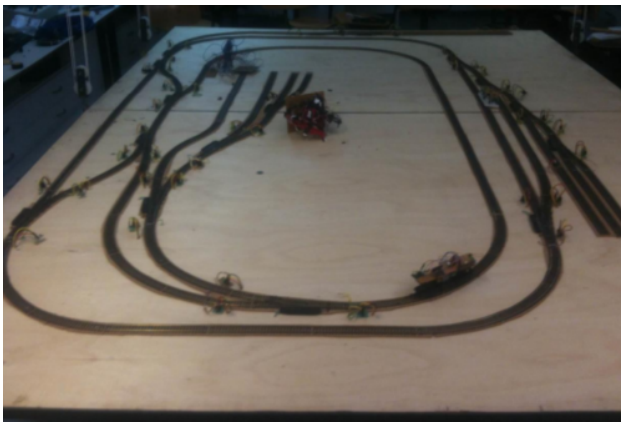

Figura 1.1: Modelo ferroviário desenvolvido no laboratório GRACO-UnB [1]

O modelo foi realizado em 2013 e encontra-se em desuso, pois a forma de comunicação entre o sistema e o computador não faz uso de uma rede muito eficaz.

1.2 Definição do problema

O modelo original [5] dispunha de 40 I/O's, ou seja, para as entradas e saídas em uma CLP eram necessárias 40 portas digitais. A necessidade de uma grande quantia de cabos para se conectar as entradas do sistema ao módulo I/O do laboratório tornava inviável a manutenção e organização do modelo. O modelo também não podia ser expandido para sistemas de maior complexidade e simular contextos mais realísticos sem aumentar, consequentemente, a quantia de cabos.

O segundo projeto [1] estabeleu uma solução utilizando um circuito multiplexador de 8 bits. A necessidade de menos portas de entradas e saídas tornou o modelo ferroviário mais viável, porém, a programação exigia uma lógica complexa de decodificação dos 8 bits em linguagem *ladder* ou *Sequential Function Chart*.

A utilização de um protocolo de rede industrial se tornou necessária, pois possibilitará a redução de fios sem aumentar a complexidade da programação da CLP.

1.3 Objetivos do projeto

Este projeto visa tornar uma placa Raspberry Pi em um dispositivo de entrada e saída através de um protocolo industrial de rede de campo para realizar a interface entre o modelo de um sistema ferroviário com o controlador. Desta forma, será possível realizar a diminuição de fios sem aumento de complexidade da programação da CLP.

1.4 Apresentação do manuscrito

Capítulo 2

Fundamentos

Neste capítulo serão abordados os conceitos necessários para a compreensão do projeto. Desta forma serão apresentados os conceitos de modelo OSI, FieldBus, CAN, CIP e DeviceNet.

2.1 Modelo OSI

O Modelo OSI (*Open System Interconnection*) é um modelo de rede de computador referência da ISO dividido em camadas de funções, criado em 1971 e formalizado em 1983, com objetivo de ser um padrão, para protocolos de comunicação entre os mais diversos sistemas em uma rede, garantindo a comunicação entre dois sistemas computacionais (end-to-end).

Este modelo divide as redes de computadores em 7 camadas, de forma a se obter camadas de abstração. Cada protocolo implementa uma funcionalidade assinalada a uma determinada camada.

Esse modelo possui 3 conceitos essenciais:

1. Protocolo: este termo denota um conjunto de regras que governam na comunicação entre camadas de mesmo nível.
2. Serviço: representa qualquer serviço disponível de uma camada (provedor de serviços) para a camada logo acima (usuário de serviços).
3. Interface: especifica quais serviços estão disponíveis, como eles podem ser acessado, quais parâmetros são transferidos e resultados esperados.

2.1.1 Modelo de Camadas

O modelo OSI faz uma distinção clara entre comunicações horizontais e verticais. A figura 2.1 mostra a estrutura de camadas e como o dado é passado de um processo para outro. O sistema começa com a transmissão de dados através da camada 7 (camada de aplicação). A camada 7 prepara os dados e faz requisição de serviços à camada 6 (camada de apresentação). A camada 6 faz o mesmo: prepara os dados e requisita serviço à camada 5 (camada de sessão), e assim continua até a camada 1 (camada física). A

cada camada, são adicionados cabeçalhos específicos aos dados específicos de cada protocolo. Ao chegar a camada física o dado é de fato transmitido pelo meio físico.

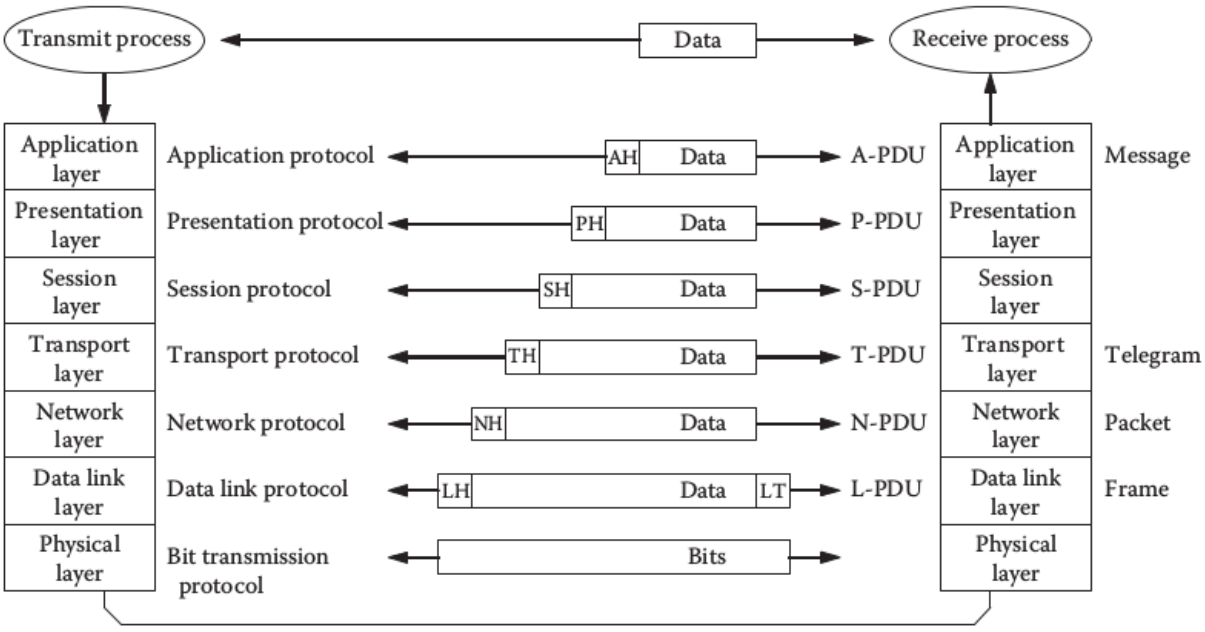

Figura 2.1: Estrutura de camadas do modelo OSI

Uma breve descrição de cada camada será realizada:

1. **Camada física**: esta camada representa as propriedades mecânicas, elétricas, ótico, físico e lógico de um sistema de comunicação.

2. **Camada de enlace de dados**: realiza a pura conexão ponto-a-ponto entre duas redes com a tarefa de garantir a integridade dos dados. Nesta camada é produzido o data frame (estrutura dos dados) e verifica erros por meio de algoritimos como o CRC.

3. **Camada de rede**: realiza roteamento de funções e também pode realizar a fragmentação e remontagem.

4. **Camada de transporte**: responsável por receber os dados enviados pela camada de sessão e segmentá-los para que sejam enviados a camada de rede, que por sua vez, transforma esses segmentos em pacotes. No receptor, a camada de Transporte realiza o processo inverso, ou seja, recebe os pacotes da camada de rede e junta os segmentos para enviar à camada de sessão.

5. **Camada de sessão**: responsável pela troca de dados e a comunicação entre hosts, a camada de Sessão permite que duas aplicações em computadores diferentes estabeleçam uma comunicação, definindo como será feita a transmissão de dados, pondo marcações nos dados que serão transmitidos.

6. **Camada de apresentação**: também chamada camada de Tradução, converte o formato do dado recebido pela camada de Aplicação em um formato comum a ser usado na transmissão desse dado, ou seja, um formato entendido pelo protocolo usado.

7. **Camada de aplicação**: é uma camada de inteface entre a unidade de comunicação e a aplicação real.

Através destas definições é possível realizar comunicações complexas de forma estruturada.

2.2 FieldBus

Em meados de 1960 o sinal analógico de 4-20mA foi introduzido para realizar o controle de dispositivos industriais. Aproximadamente em 1980, os sensores inteligentes começaram a ser desenvolvidos e implementados usando um controle digital. Isso motivou a necessidade de integrar vários tipos de instrumentações digitais em campos de comunicações, visando otimizar a performance dos sistemas. Dessa forma se tornou óbvio que um padrão era necessário para formalizar o controle dos dispositivos inteligentes.

Fieldbus é um termo genérico empregado para descrever tecnologias de comunicação industrial; o termo fieldbus abrange muitos diferentes protocolos para redes industriais (como Modbus, Profibus, CAN, DeviceNet). Segundo a definição dada pela norma IEC 61158 [6] "FieldBus é um barramento de dados digital, serial, multicomponentes, de dados para comunicação com controles industriais e dispositivos como, por exemplo, transdutores, atuadores e controladores locais".

Em uma hierarquia de protocolos, o FieldBus atua em nível de campo (nível de sensores) e processos (CLPs) como indicado na figura 2.2.

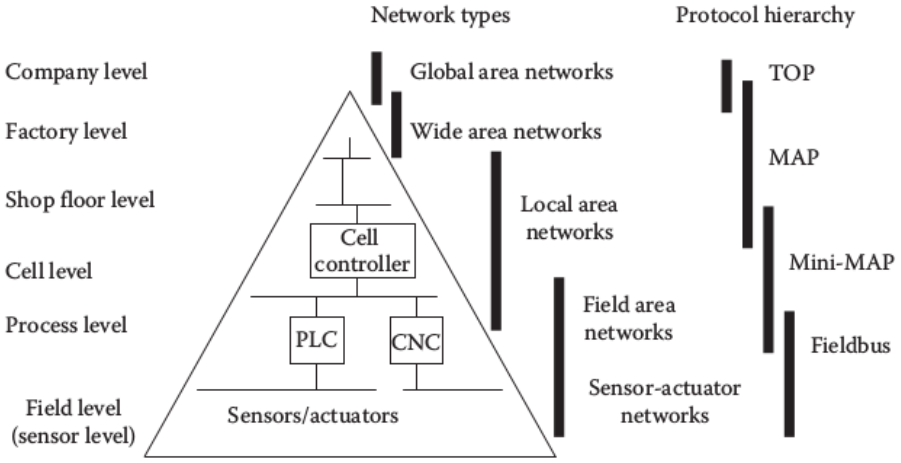

Figura 2.2: Nível hierárquico de redes de automação [2]

Apesar das LANs estarem cada vez mais presentes e estarem reduzindo os níveis da hierarquia na automação, nunca irão deixar o fieldbus obsoleto. A LAN é uma rede de propósito geral enquanto o Fieldbus é de propósito específico. Por sua alta especificidade, o protocolo de campo obtém melhor performance em redes do mais baixo nível (que envolvem diretamente com sensores e atuadores).

Os protocolos fieldbus são modelados, em sua essência, através do modelo OSI. Porém, apenas as camadas 1, 2 e 7 são de fato utilizadas [7]. Desta forma, o modelo OSI fica reduzido à estrutura da figura

2.3. Algumas funcionalidades das camadas 3 a 6 ainda existem, porém implementadas nas camadas 2 ou 7. O padrão IEC 61158 estabelece a versão de modelo de camadas do fieldbus.

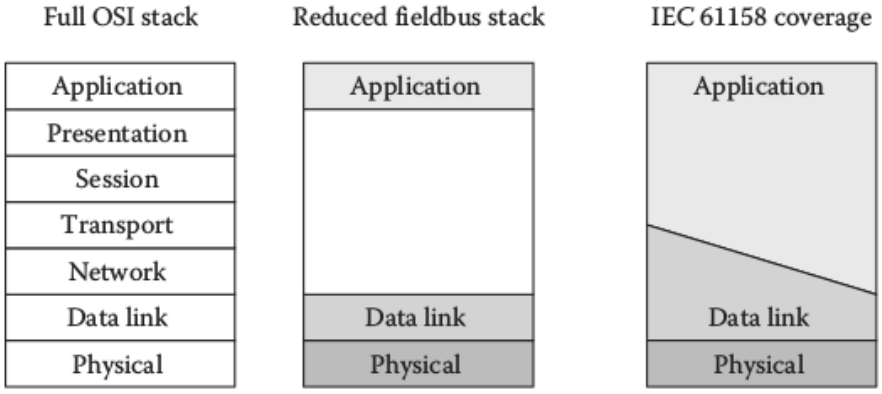

Figura 2.3: Estrutura de camadas do fieldbus definido por IEC 61158 [2]

As camadas de apresentação de um fieldbus devem ser compreensíveis. São indispensáveis para sistemas abertos e para gerar uma base grande interoperabilidade. Desta forma, a programação de algumas abstrações de funções de certas aplicações pode ficar muito complexa. Por esta razão, alguns protocolos fieldbus não possuem a camada de apresentação. Um exemplo é o protocolo CAN, que serve como base para os protocolos CANopen, SDS e DeviceNet.

2.3 Protocolo CAN

O protocolo CAN foi desenvolvido por Robert Bosh em 1986 para aplicação na indústria autombilística, com o objetivo de simplificar os complexos sistemas de fios em veículos com sistemas de controlo compostos por múltiplos microcontroladores/microcomputadores para gestão do motor, sistema ABS, controlo da suspensão, etc. A sua especificação base anunciava elevada taxa de transmissão, grande imunidade a interferências eléctricas e capacidade de detectar erros.

O CAN implementa apenas as duas camadas inferiores do modelo OSI (camada física e camada de enlace de dados) e é regulamentado pela normas ISO11898 para aplicações de alta velocidade, ISO11519 para aplicações de baixa velocidade e SAE J1939 para veículos pesados (caminhões e ônibus) [8].

O CAN é considerado um sistema de barramento série, bom para ligar em rede subsistemas inteligentes, tais como sensores e atuadores. A informação transmitida possui tamanho curto. Assim, cada mensagem CAN pode conter um máximo de 8 bytes de informação útil, sendo no entanto possível transmitir blocos maiores de dados recorrendo a segmentação.

2.3.1 Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

O protocolo CAN é fundamentado no conceito CSMA/CD. CSMA significa *Carrier Sense Multple Access*, ou seja, cada nó da rede deve monitorar o barramento por um período de inatividade antes de

enviar uma mensagem (*Carrier Sense*). No período de inatividade, todos os nós tem mesma oportunidade para transmitir seus dados (*Multiple Access*). CD significa *Collision Detection*. Caso dois nós tentem acessar a rede ao mesmo tempo, será detectada uma colisão.

No CAN também é utilizada uma arbitragem de lógica binária não destrutiva que permite que a mensagem não se corrompa mesmo havendo colisão. Para definir essa arbitragem, define-se o estado dominante e o recessivo. O bit 0 é definido como bit dominante e 1 como recessivo. O bit dominante sempre terá prioridade sobre o barramento, desta forma, quando dois nós acessarem o barramento algum deles irá perder a arbitragem emitindo um bit recessivo em seu identificador de mensagem (que se encontra nos bits de arbitragem da mensagem). No momento que perder a arbitragem, o nó com menor prioriadade irá parar de transmitir os dados.

A comunicação é baseada em mensagens e realizada em broadcast, onde todos os dispositivos dos nós recebem a mensagem e cabe a cada um verificar se é uma mensagem a ser processada e reconhecida ou não. Desta forma uma mesma mensagem pode endereçar apenas um nó ou múltiplos. Outro benefício é a possibilidade de inserir nós à rede sem precisar reprogramá-la, basta o nó ser capaz de processar as mensagens a ele direcionadas.

2.3.2 Data frame

O CAN possui 4 tipos de frame, mas o mais utilizado é o standard data frame, ilustrado pela figura 2.4. Em seu primeiro bit há um sinalizador de início de frame (*Start of Frame - SoF*) seguido pelo campo de arbitragem de 12 bits. No campo de artbitragem se encontra 11 bits de identificador de mensagem e 1 bit de RTR. O RTR é usado quando um nó necessita da informação de outro nó da rede.

Após o campo de arbitragem, existem 6 bits de controle, que definem qual tipo de frame a ser usado e o tamanho do dado a ser transmitido na mensagem.

Em sequência, existe o campo de dados, cujo tamanho é definido pelo campo de controle, podendo variar de 0 a 8 bytes.

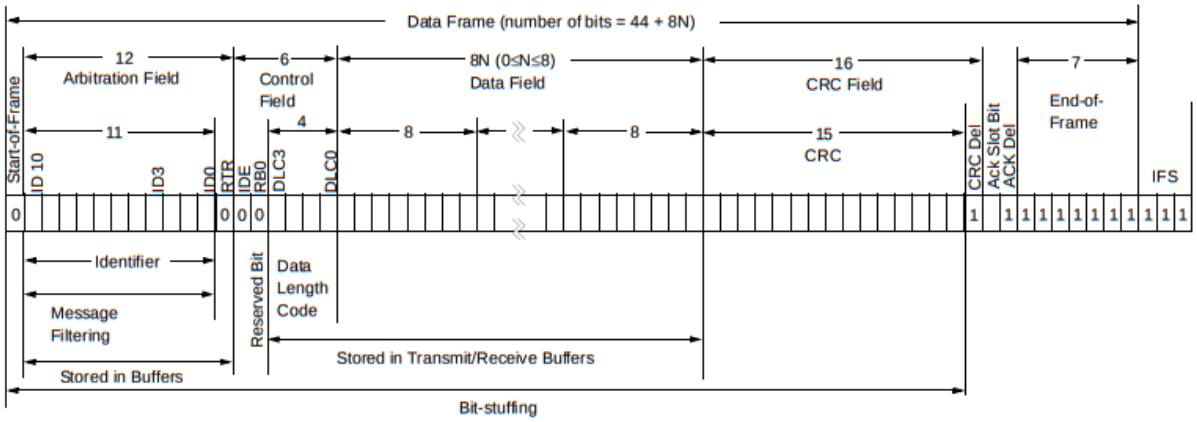

Figura 2.4: Standard CAN Data Frame [3]

Após o campo de dados existe o CRC. Este campo é utilizado para verificar a integridade da mensagem. Por fim existem 2 bits de reconhecimento e 7 bits para identificar o fim de frame. Entre uma mensagem e outra existe um espaço entre frames preenchidos com bits recessivos.

2.3.3 Tratamento de erros

Cada nó é capaz de identificar falhas e transitar para outros estados de operação. Desta forma, um nó pode ir de modo normal para o modo *bus-off* (desligado da rede). Todos os nós com falhas não são capazes de acessar o barramento para evitar derrubar a rede. Existem 5 condições de erros e 3 estados de erros no protocolo CAN.

2.3.3.1 Erros detectados

- *CRC Error* - o campo CRC no data frame contém um valor calculado por uma função hash com os dados da mensagem. Após transmitida a mensagem, todos os nós da rede recalculam o valor CRC da mensagem e verifica a integridade da mensagem. Caso o valor calculado pelos nós não coincida com o CRC da mensagem, um error frame é gerado. Como um nó não recebeu a sua devida mensagem por conta do erro, a mensagem é retransmitida após devido tempo.
- *Acknowledge Error* - se os bits de reconhecimento não forem dominantes, significa que nenhum nó recebeu a mensagem. Portanto, um erro é detectado e produz-se um error frame e a mensagem retransmitida.
- *Form Error* - se for detectado um bit dominante no espaço entre frames, no acknowledge delimiter ou no fim do frame, o protocolo CAN reconhece como uma forma de violação gerando um error frame.
- *Bit Error* - quando um nó estiver transmitindo a mensagem e em seu próprio monitoramento do barramento identificar que enviou um bit errado, ocorre um bit error e a mensagem é retransmitida.
- *Stuff Error* - como o protocolo é assíncrono, é necessária uma lógica de *bit-stuffing* para síncronizar os clocks de todos os nós. Caso haja um erro de *bit-stuffing* o frame error é gerado.

2.3.3.2 Estados de erro

- *Error-Active* - é o modo de operação normal de um nó, onde ele é permitido ler e escrever na rede.
- *Error-Passive* - ocorre quando o contador de erros exceder 127. Quando um nó está nesse modo, ele deve aguardar pelo menos 8 bits após o término de um frame para realizar a sua transmissão.
- *Bus-off* - ocorre quando o contador de erros exceder 255. Quando um nó está nesse modo, ele não pode enviar nem receber dados da rede.

2.4 Common Industrial Protocol - CIP

O *Common Industrial Protocol* é um protocolo de comunicação de redes industriais ponto-a-ponto que provê conexões entre dispositivos industriais (sensores, atuadores) e dispositivos de alto nível (CLP) [4]. Este protocolo possui uma grande versatibilidade, tendo sido integrado em vários outros protocolos de camada de aplicação como o EtherNet/IP™, DeviceNet™, ControlNet™, e CompoNet™. A figura 2.5 mostra a relação das principais adaptações do CIP.

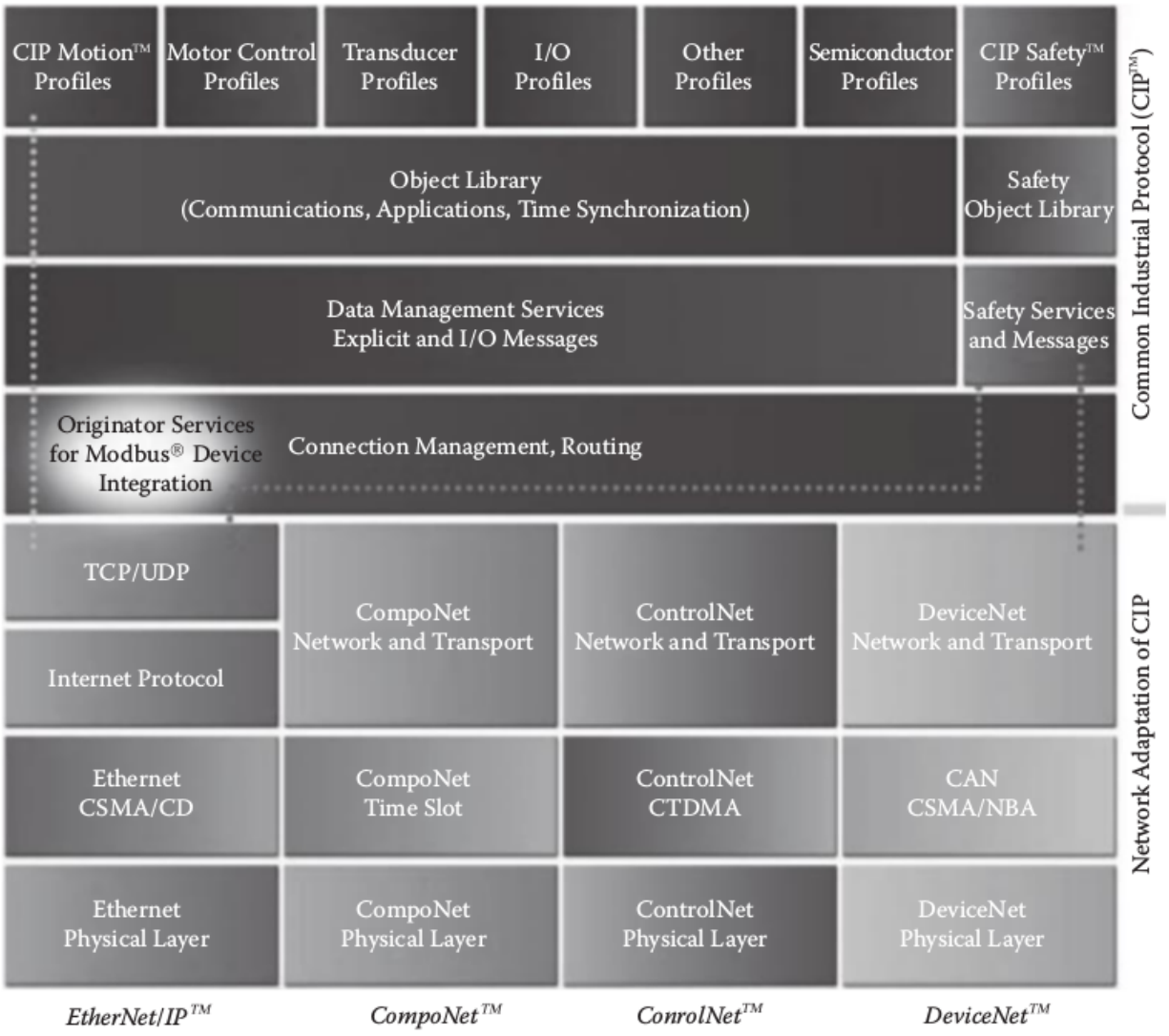

Figura 2.5: O Common Industria Protocol e suas principais adaptações [2]

Alguns termos importantes para a compreensão do CIP são

- *Cliente* - dentro de um modelo cliente/servidor, o cliente é o dispositivo faz uma requisição de serviço ao servidor. O cliente espera uma resposta do servidor.
- *Servidor* - dentro de um modelo cliente/servidor, o servidor é o dispositivo que presta um serviço ao

cliente. O servidor retorna uma resposta ao cliente.

- *Produtor* - dentro do modelo produtor/consumidor, o produtor é aquele que insere uma mensagem na rede para um ou mais consumidores.
- *Consumidor* - dentro do modelo produtor/consumidor, o consumidor é aquele que coleta uma mensagem da rede.
- *Modelo Produtor/Consumidor* - é um modelo multicast no qual os nós definem se irão consumir a mensagem na rede dependendo do *connection ID* (CID) no pacote de dados.
- *Mensagem Explícita* - uma mensagem que contém uma informação endereçamento e serviço que será direcionada a um dispositivo que irá realizar algum serviço oferecido por um de seus componentes.
- *Mensagem (I/O) Implícita* - uma mensagem que os nós consumidores já sabem o que fazer com o dado baseado no CID da mensagem. Em geral são utilizados para transportar dados de I/O.

2.4.1 Modelagem de objetos

Este protocolo tem como base a modelagem em objetos. Portanto, todo nó de uma rede é modelado como uma coleção de objetos que fará a abstração de cada componente do dispositivo. Os objetos CIP são estruturados em classe, instâncias e atributos.

Uma classe representa um conjunto de objetos que pertencem ao mesmo tipo de componente do sistema. Uma instância de um objeto é a representação de um objeto particular em uma classe. Todas instâncias de uma classe possui os mesmos conjunto de atributos, porém, cada um tem seus particulares valores de cada atributo. A figura 2.6 ilustra multiplas instâncias de objeto em uma classe que residem em um nó.Cada classe também pode conter atributos que a descreva.

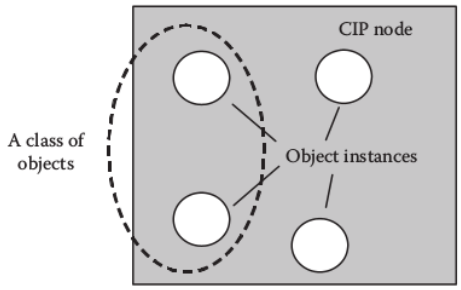

Figura 2.6: Uma classe de objetos [2]

Todos objetos e seus componentes são endereçados por um sistema de endereçamento uniforme que consiste em:

- *Media Access Control Identifier (MAC ID)* - identificação para cada nó da rede
- *Class Identifies (Class ID)* - endereço atribuido para cada classe de objeto acessível na rede

- *Instance Identifier (Instance ID)* - endereço para cada instância de objeto em uma classe
- *Attribute Identifier (Attribute ID)* - endereço dos atributos de uma classe ou instância
- *Service Code* - denota a função da instância de um objeto ou de uma classe de objeto

2.4.2 Serviços

Códigos de serviços são utilizados para definir uma ação que foi requisitada através de uma mensagem explícita. Além de serviços de leitura e escrita de dados, o CIP estabelece vários serviços de natureza similares que podem ser utilizadas por qualquer rede CIP e são uteis para vários objetos. Existem códigos de serviços que são específicos de um objeto, portanto, esse mesmo código pode ser utilizado por outro objeto para identificar um serviço diferente. Por fim, também pode-se fazer um serviço específico de acordo com os requisitos de um produto a ser desenvolvido. A criação de serviços oferece uma grande flexibilidade, porém o desenvolvedor necessita realizar uma documentação para futuros usos, uma vez que não é um serviço universal.

2.4.3 Protocolo de mensagens

CIP é uma rede baseada em conexão. Toda conexão realizada são atribuídas CIDs (*Connection ID*). Se a comunicação é bidirecional, são necessárias duas CIDs. Existem dois objetos de conexão: Conexão I/O (para comunicação de propósito especial também conhecido como conexão implícita) e conexão de mensagens explícitas (de propósito múltiplo/genérico). As mensagens explícitas tem uma estrutura pré-definida pelo protocolo de bits enquanto mensagens do tipo I/O não possuem uma estrutura pré-definida, cabe ao desenvolvedor definir a interpretação destas.

Para se estabelecer uma conexão pode-se utilizar a função do *Unconnected Message Manager* (UCMM) que oferece o serviço de `Forward_open` que estabelece uma conexão. Existem outras formas de se estabelecer a conexão além do UCMM, um exemplo é fazer uso de comunicações baseadas mensagens pré-estabelecidas em conjuntos Mestre-Escravo, que será descrito mais a frente.

2.4.4 Objetos de comunicação

Os objetos de comunicação gerenciam e providenciam o tempo de troca de mensagens. Cada objeto de comunicação possui uma parte de *link consumer*, ou uma de *link producer* ou ambos. As figuras 2.7 e 2.8 ilustram as conexões do tipo I/O e explícitas, respectivamente. Os atributos dos objetos de conexão estabelecem o maior tamanho de dado (em bytes) que pode ser consumido ou produzido, endereço de destino, tempo de resposta esperado, entre outros.

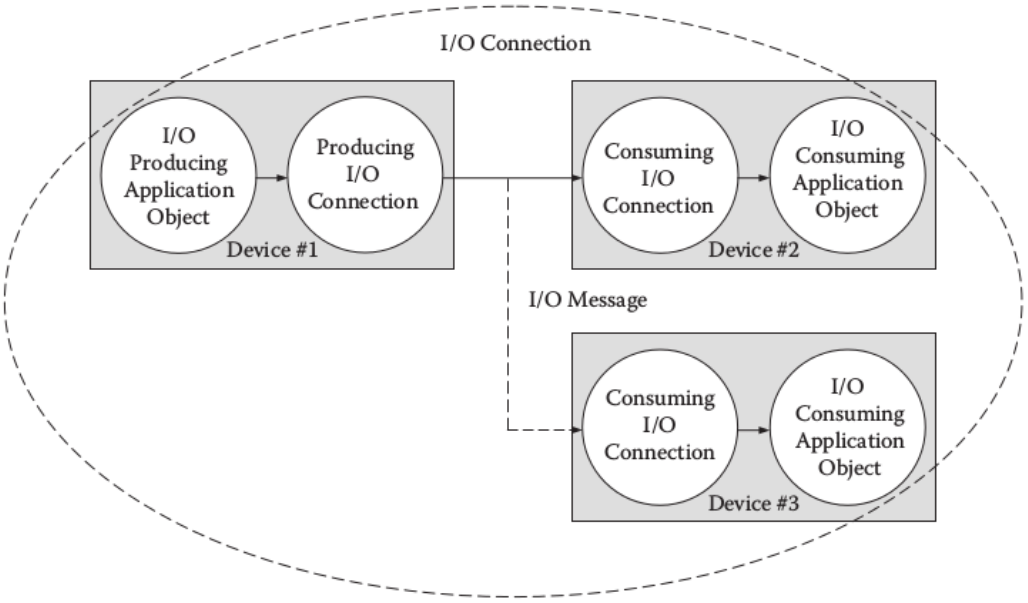

Figura 2.7: Conexão I/O [4]

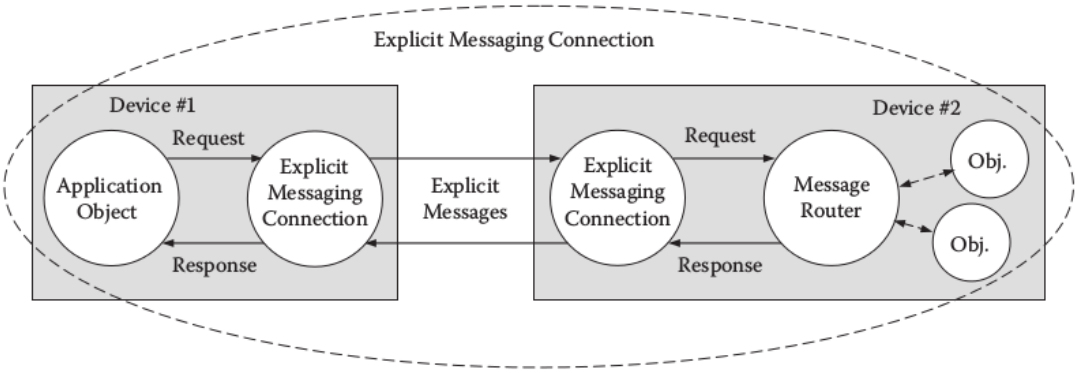

Figura 2.8: Conexão Explícita [4]

2.4.5 Biblioteca de objetos

O CIP possui uma coleção de objetos já definidos. Pode-se classificar os objetos em três grupos: de uso geral, de aplicação específica e de rede especifica.

Os objetos de uso geral são definidos na tabela 2.1 (os valores em hexadecimal entre parênteses corresponde ao Class ID):

Assembly (0x04)	Message Router (0x02)
Acknowledge Handler (0x2B)	Originator Connection List (0x45)
Connection (0x05)	Parameter (0x0F)
Connection Configuration (0xF3)	Parameter Group (0x10)
Connection Manager (0x06)	Port (0xF4)
File (0x37)	Register (0x07)
Identity (0x01)	Selection (0x2E)

Tabela 2.1: Objetos de uso geral definidos pelo CIP

Os objetos de aplicação específica são listados na tabela 2.2

AC/DC Drive (0x2A)	Analog Group (0x22)
Analog Input Group (0x20)	Analog Input Point (0x0A)
Analog Output Group (0x21)	Analog Output Point (0x0B)
Base Energy (0x4E)	Block Sequencer (0x26)
Command Block (0x27)	Control Supervisor (0x29)
Discrete Group (0x1F)	Discrete Input Group (0x1D)
Discrete Output Group (0x1E)	Discrete Input Point (0x08)
Discrete Output Point (0x09)	Electrical Energy (0x4F)
Event Log (0x41)	Group (0x12)
Motion Device Axis (0x42)	Motor Data (0x28)
Nonelectrical Energy (0x50)	Overload (0x2C)
Position Controller (0x25)	Position Controller Supervisor (0x24)
Position Sensor (0x23)	Presence Sensing (0x0E)
S-Analog Actuator (0x32)	S-Analog Sensor (0x31)
S-Device Supervisor (0x30)	S-Gas Calibration (0x34)
S-Partial Pressure (0x38)	S-Sensor Calibration (0x40)
S-Single Stage Controller (0x33)	Safety Analog Input Group (0x4A)
Safety Analog Input Point (0x49)	Safety Discrete Input Group (0x3E)
Safety Discrete Input Point (0x3D)	Safety Discrete Output Group (0x3C)
Safety Discrete Output Point (0x3B)	Safety Dual Channel Analog Input (0x4B)
Safety Dual Channel Output (0x3F)	Safety Supervisor (0x39)
Safety Validator (0x3A)	Softstart (0x2D)
Target Connection List (0x4D)	Time Sync (0x43)
Trip Point (0x35)	

Tabela 2.2: Objetos de aplicação específica definidos pelo CIP

Por fim, os objetos de rede específica são listados na tabela 2.3

Base Switch (0x51)	CompoNet Link (0xF7)
CompoNet Repeater (0xF8)	ControlNet (0xF0)
ControlNet Keeper (0xF1)	ControlNet Scheduling (0xF2)
Device Level Ring (DLR) (0x47)	DeviceNet (0x03)
Ethernet Link (0xF6)	Modbus (0x44)
Modbus Serial Link (0x46)	Parallel Redundancy Protocol (0x56)
Power Management (0x53)	PRP Nodes Table (0x57)
SERCOS III Link (0x4C)	SNMP (0x52)
QoS (0x48)	RSTP Bridge (0x54)
RSTP Port (0x55)	TCP/IP Interface (0xF5)

Tabela 2.3: Objetos de rede definidos pelo CIP

Todo dispositivo CIP possuí no mínimo um objeto de *Identity*, um objeto *Connection* ou *Connection Manager*, um *Message Router*, um objeto de rede específica. Outros objetos são adicionados para as funcionalidades desejada para o dispositivo.

2.4.6 Configuração e EDS (*Electronic Data Sheets*)

O CIP oferece diversas formas de configurar dispositivos:

- Data sheet impresso
- Objeto de parâmetros
- Arquivo EDS
- Combinação de EDS com objeto de parâmetros

O data sheet impresso não é muito eficaz pois necessita o usuário adicionar manualmente as informações do dispositivo. Este método não disponibiliza o contexto, conteúdo nem o formato dos dados.

O objeto de parâmetros disponibiliza todos os possíveis dados de configuração e é realizada a configuração de forma automática pela ferramenta de configuração. Porém, para projetos de pequeno porte, detalhes de todos os parâmetros não são necessários, tornando-se um inconveniente para o desenvolvedor.

O arquivo EDS supre todas as informações necessárias do dispositivo sem sobrecarregar de informações. Este é um arquivo que pode ser gerado em qualquer editor de texto ASCII, porém existem ferramentas de edição específica para produzir EDS's para facilitar a produção do arquivo de configuração.

As regras estabelecidas pelo CIP dita que os arquivos EDS deverão estar divididas em determinadas seções, que são identificadas por estarem entre colchetes []. As seções são:

- File: Descreve o conteúdo e a revisão do arquivo.

- Device: Usado para identificar o dispositivo.
- Device Classification: Descreve a quais redes ele pode ser conectado.
- ParamClass: Descreve as configurações além dos parâmetros das classes.
- Params: Identifica todas as configurações de cada dispositivo.
- Groups: Identifica todos os grupos de parâmetros do dispositivo e lista o nome do grupo e número dos parâmetros.
- Assembly: Descreve a estrutura dos dados.
- Connection Manager: Descreve as conexões que o dispositivo tem suporte.
- Connection ManagerN: O mesmo da seção [Connection Manager], porém para apenas algumas portas.
- Port: Descreve as várias portas de rede que o dispositivo pode se conectar.
- Capacity: Descreve a capacidade de comunicação das redes ControlNet e Ethernet/IP.
- Connection Configuration: define o Connection Configuration Object caso tenha sido implementado.
- Event Enumeration: Associa um evento ou código de status dentro de um dispositivo com uma string.
- Symbolic Translation: Traduz uma string simbólica para o valor real.
- Internationalization: Disponibilizar uma string em diversas linguagens.
- Modular: Descreve a estrutura modular interna do dispositivo.
- IO_Info: Descreve o método de conexão I/O e o tamanho do dado de I/O. Permitido apenas para o DeviceNet.
- Variant_IO_Info: Descreve multiplos IO_Info. Permitido apenas para o DeviceNet.
- EnumPar: Enumera uma lista de parâmetros de escolha para o usuário. Permitido apenas para o DeviceNet.
- Seções Object Class: Descreve os detalhes de cada classe de objetos.

Uma ferramenta com uma coleção de arquivos EDS usa a seção [Device] e tenta fazer a correspondência com todos os nós encontrados na rede, permitindo a interface entre o dispositivo e a ferramenta utilizada.

2.4.7 Gerenciamento de dados

O gerenciamento de dados (descrito no apêncide C de [4]) descreve o modelo de endereçamento para entidades CIP e as estruturas de dados das próprias entidades.

O modelo de gerenciamento é realizado em segmentos, que podem ser do tipo

- Port Segment - usado para rotear de uma sub-rede à outra
- Logical Segment - informação de referência lógica (endereço de classe/instância/atributos)
- Network Segment - especifica parâmetros de rede necessário para transmitir para outras redes.
- Symbolic Segment - nome simbólicos
- Data Segment - Dado embarcado (como, por exemplo, dados de configuração)

Entre esses, os mais importantes são o Logical Segment e o Data Segment, os quais serão melhor detalhados.

2.4.7.1 Logical Segment

O **Logical Segment** possuí em seu primeiro byte valores entre 0x20 e 0x3F pode ser utilizado para endereçar objetos e seus atributos em um dispositivo. A estrutura típica é colocada na sequência [Class ID] [Instance ID] [Attribute ID] (segundo [2]).

Esse tipo de endereçamento é utilizado tipicamente para endereçar objetos *Assembly*, *Parameter*, entre outros endereçáveis. É utilizado de forma extensa em arquivos EDS e mensagens explícitas.

2.4.7.2 Data Segment

O **Data Segment** providencia um mecanismo para entregar dados para uma aplicação. Seu primeiro byte tem valores entre 0x80 e 0x9F. Os dados podem ser estruturados ou elementares e cada um tem um tipo de codificação que segue os requerimentos da norma IEC 61131-3 [6]. Para o contexto deste projeto os dados elementares são os mais relevantes, pois são estes que são utilizados para especificar parâmetros do arquivo EDS. Os tipos mais relevantes são

- 1 bit
  - Boolean, BOOL, Código 0xC1
- 1 byte
  - Bit string, 8 bits, BYTE, Código 0xD1
  - Unsigned 8-bit integer, USINT, Código 0xC6
  - Signed 8-bit integer, SINT, Código 0xC2

- 2 bytes
  - Bit string, 16 bits, WORD, Código 0xD2
  - Unsigned 16-bit integer, UINT, Código 0xC7
  - Signed 16-bit integer, INT, Código 0xC3
- 4 bytes
  - Bit string, 32 bits, DWORD, Código 0xD3
  - Unsigned 32-bit integer, UDINT, Código 0xC8
  - Signed 32-bit integer, DINT, Código 0xC4

2.5 DeviceNet

[9]

Capítulo 3

Conclusões

Concluir

3.1 Perspectivas Futuras

Perspectivas futuras

REFERÊNCIAS BIBLIOGRÁFICAS

[1] ANDRADE, L. H. do E. S.; VEIGA, I. V. A. *PROJETO E IMPLEMENTAÇÃO DE UMA MAQUETE PARA ESTUDO DE PROBLEMAS RELACIONADOS À AUTOMAÇÃO DE SISTEMAS DE TRANSPORTE FERROVIÁRIO*. [S.l.]: Universidade de Brasília, 2013.

[2] ZURAWSKI, R. *The Industrial Communication Technology Handbook (Industrial Information Technology)*. 1. ed. [S.l.]: CRC Press, 2005. ISBN 0849330777,9780849330773.

[3] ODVA. *THE CIP NETWORKS LIBRARY, Volume 3, DeviceNet Adaption of CIP*. [S.l.]: ODVA and ControlNet International Ltd., 2007.

[4] ODVA. *THE CIP NETWORKS LIBRARY, Volume 1, Common Industrial Protocol*. [S.l.]: ODVA and ControlNet International Ltd., 2007.

[5] JESUS, T. R. de; AQUINO, J. A. V. de. *ESTRATÉGIA DE CONTROLE DE TRÁFEGO EM VIAS FÉRREAS SINGELAS, UTILIZANDO PROGRAMAÇÃO SFC SOBRE ARQUITETURA DE CONTROLADORES INDUSTRIAIS PROGRAMÁVEIS*. [S.l.]: Universidade de Brasília, 2009.

[6] IEC 61158 PROFIBUS: Physical Layer. [S.l.], mar. 2003.

[7] PLEINEVAUX P.; DECOTIGNIE, J.-D. Time critical communication networks: field buses. *IEEE Network*, IEEE, v. 2, 05 1988.

[8] CONTROLLER Area Network (CAN) Basics. [S.l.], 1999.

[9] DATASHEET MCP2515. [S.l.]: Microchip. `http://ww1.microchip.com/downloads/en/DeviceDoc/21801d.pdf`.

ANEXOS

I. DESCRIÇÃO DO CONTEÚDO DO CD

Descrever CD.

II. PROGRAMAS UTILIZADOS

Quais programas foram utilizados?